A DRAFT

Thirty Brief Lectures on Foundations of Deep Learning

Tomaso Poggio, Gemini+ChatGPT

Intelligence and its Fundamental Principles

February 18, 2026

Prologue

Modern deep learning works astonishingly well, yet we still lack a fundamental understanding of *why*. What we call "deep learning" is not merely a triumph of computer science or engineering; it is a natural phenomenon that demands explanation in terms of underlying principles—much as physics explains why the Sun produces energy or why matter follows specific laws.

This collection of essays is an attempt to uncover some of those principles. It argues that today's AI systems succeed because they instantiate deep structural regularities of the world—regularities that any intelligent system, biological or artificial, must exploit. Two principles emerge repeatedly throughout these essays.

The first is *(sparse) compositionality*, and the second is *genericity*. In this essays, we show that they are not heuristic conjectures, but logical inevitabilities derived from the foundations of computation and physics. We show that sparse compositionality is mathematically implied by efficient computability, just as genericity is dictated by assuming invariance in the choice of coordinates. Together, they define the channel through which intelligence must pass.

Together – and helped by several other observations – these principles offer a first step toward a unified theory of modern AI—one that explains not only *how* current systems work, but *why* they work at all.

Contents

Prologue i

How to Read This Collection 1

A Note on Co-Authorship 3

Part I: Principles and Foundations 5

1 Intelligence as Associative Memory 6
- 1.1 Building a Turing Machine with Attention Heads . . . 7
- 1.2 Connections to Other Models . . . 8
- 1.3 Clarifying the Associative-Memory Lens . . . 8
  - 1.3.1 HyperBF Equivalence: Attention as a Normalized HyperBF . . . 8
- 1.4 Multi-Head Requirements for RAM-like Computation . . . 9
- 1.5 On Diligence, Creativity, and Exploration . . . 9
- 1.6 Question: How Can Transformers Be So Consistent? . . . 10
  - 1.6.1 Global Consistency from Shared Latent Geometry . . . 11
  - 1.6.2 Self-Reinforcing Autoregression . . . 11
  - 1.6.3 Emergent Consistency as Manifold-Constrained Recall . . . 11
- 1.7 Memories . . . 11

2 A Historical Reflection on Associative Memories 13
- 2.1 Associative Memory as the Core of Intelligence . . . 13
- 2.2 Introduction . . . 13
- 2.3 Historical Background . . . 14
  - 2.3.1 Early Heteroassociative Models . . . 14
  - 2.3.2 Nonlinear Associative Recall and Polynomial Expansions . . . 15
  - 2.3.3 Correlation Memories, RBFs, and HyperBFs . . . 15
  - 2.3.4 Hopfield Networks as a Special Case . . . 15
- 2.4 Genericity as the Hidden Enabler of Associative Memories . . . 16
- 2.5 Evolutionary Perspective . . . 16
  - 2.5.1 From Reflexes to Multi-Stage Adaptive Hierarchies . . . 16
- 2.6 Modern Parallels: Transformers as Associative Memories . . . 17
- 2.7 What Associative Memory Does—and Does Not—Explain . . . 17
  - 2.7.1 Experimental Evidence from Homogeneous HyperBF Transformers . . . 18
- 2.8 Conclusion . . . 19

3 Associative Memories are Turing-Complete **20**
3.1 Introduction . . . 21
3.2 Model: States, Memories, Reads, and Updates . . . 21
3.2.1 State Space and Encodings . . . 21
3.2.2 Associative Memory Interface ($\mathcal{M}_{\text{prog}}$) . . . 21
3.2.3 Transition Update (M_1) and Query Map . . . 22
3.3 From Turing Machines to ATMs . . . 22
3.3.1 Embedding the Tape . . . 22
3.3.2 Embedding the Transition Table . . . 22
3.4 Main Theorem and Proof Sketch . . . 23
3.4.1 Proof Intuition . . . 23
3.5 Structural Correspondence to the Transformer . . . 23
3.5.1 The Memory ($\mathcal{M}_{\text{prog}}$) and Query Map ($Q_{\text{map}}$) . . . 23
3.5.2 The Read Operation (read_τ) . . . 23
3.5.3 The Update Function (M_1) . . . 23
3.6 The Transformer as an Associative Turing Machine . . . 24
3.6.1 The Update Function (M_1) corresponds to the Feed-Forward Network . . . 24
3.6.2 The Memory ($\mathcal{M}_{\text{prog}}$) corresponds to Keys and Values . . . 24
3.6.3 Summary of Correspondence . . . 24

4 Efficient Computability and Compositional Sparsity **26**
4.1 Perspective and Scope . . . 27
4.2 Definitions and model conventions . . . 29
4.2.1 Efficient computability (Boolean and real-valued) . . . 29
4.2.2 Computation DAGs and compositional sparsity . . . 29
4.3 Efficient computability $\Rightarrow$ compositional sparsity . . . 30
4.3.1 Boolean case . . . 30
4.3.2 Real-valued case: safe discrete-grid result and conditional continuous result . 30
4.4 Consequences and realizations . . . 31
4.4.1 Small circuits (tautological) . . . 31
4.4.2 Exact deep-ReLU realizations on discrete domains . . . 31
4.4.3 Sparse tabulation (lookup) . . . 31
4.4.4 Algebraic/ polyhedral realization . . . 31
4.5 Unifying mechanisms: a normalized-similarity read (with assumptions) . . . 31
4.6 Addressability vs. superpositional storage . . . 33
4.6.1 Exponential codebooks via random spherical codes . . . 33
4.6.2 Holographic superposition and SNR . . . 33
4.7 Concluding remarks . . . 34
4.8 Mathematical Supplement: Proofs of the Main Theorems . . . 34
4.8.1 Encoding of Turing Configurations . . . 34
4.8.2 Proof of Theorem 8 (Autoregressive Universality) . . . 35
4.8.3 Proof of Theorem 9 (Diffusion-Step Universality) . . . 35
4.8.4 Remarks and Extensions . . . 36
4.9 Technical Note: Replacing Linear Threshold Functions by Boolean Circuits . . . 36
4.9.1 Expressing a Turing-Machine Step as a Circuit . . . 36
4.9.2 Replacement in the Stepwise-Learning Framework . . . 37
4.9.3 Comparison with Linear Threshold Implementations . . . 37
4.9.4 Summary . . . 37

5 Optimization and Compositionality (with P. Beneventano) 38
5.1 The Core Argument . . . 39
5.2 The Representation Benefit: Avoiding the Curse . . . 39
5.2.1 Parameter Counting . . . 39
5.3 The Optimization Challenge in End-to-End Learning . . . 39
5.3.1 Vanishing Gradients and Conditioning . . . 40
5.4 The Ideal Scenario: Module-wise Optimization . . . 40
5.4.1 The "Grey Box" vs. "Black Box" . . . 40
5.4.2 Curriculum Learning and Pre-training . . . 40
5.5 Sample Complexity: The Gap Between Shallow and Deep . . . 41
5.5.1 The Curse for Shallow Architectures . . . 41
5.5.2 The Blessing for Deep Compositional Networks . . . 41
5.6 Summary: The Optimization-Representation Trade-off . . . 42

6 Genericity and Optimization (with P. Beneventano) 43
6.1 The Core Argument . . . 44
6.2 Defining Genericity: Invariance to Shifts . . . 44
6.2.1 General Genericity . . . 44
6.2.2 The Genericity Principle (Shift Invariance) . . . 44
6.3 Why non-genericity is fragile: Thom transversality . . . 45
6.3.1 Jets and “bad” sets of functions . . . 45
6.3.2 Thom’s transversality theorem (informal) . . . 45
6.3.3 Connection to shift-based genericity . . . 45
6.4 The Mathematics of Structure Leaking . . . 46
6.4.1 Real-Valued Targets: Linear Footprints . . . 46
6.4.2 Boolean Targets: The Bias Leakage . . . 46
6.5 Clarifying Generalization vs. Optimization . . . 47
6.6 From Genericity to Optimization Dynamics . . . 48
6.6.1 The Gradient Signal at Initialization . . . 48
6.6.2 The Optimization Hierarchy . . . 48
6.6.3 Sample Complexity: The Staircase vs. The Cliff . . . 49
6.7 Case Study: The Danger of Residual Fitting . . . 49
6.8 Summary . . . 49

7 Principles of Deep Learning 50
7.1 Principle I: Sparse Compositionality . . . 50
7.1.1 Hierarchy follows from sparse compositionality . . . 51
7.1.2 Modularity and reuse of modules . . . 51
7.1.3 Transfer learning . . . 52
7.1.4 Interpretability from consistent compositionality . . . 52
7.1.5 Non-uniqueness of sparse decompositions . . . 52
7.2 Principle II: Genericity of Learnable Targets . . . 52
7.2.1 Genericity from invariance to the choice of the origin of the coordinates . . . 53
7.2.2 Genericity ensures good gradients for optimization . . . 53
7.3 Two independent but complementary principles . . . 53
7.4 Conclusion . . . 54

Part II: Computation and Algorithms 55

8 Efficient Computability, Compositional Sparsity, and Self-Attention 56
8.1 Preliminaries and Assumptions . . . 57
8.2 Main Results . . . 57
8.2.1 Theorem 2 (Compositional Approximation by Attention with Dimension-Free Rate) . . . 57
8.2.2 Theorem 3 (Efficient Computability ⇒ Transformer Approximants) . . . 58
8.2.3 Theorem 4 (Margin Implies Near Top-k Sparsity) . . . 58
8.2.4 Theorem 5 (Low Rank Suffices for k-Ary Nodes) . . . 59
8.3 Consequences and Predictions . . . 59
8.4 Empirical Consistency . . . 59
8.5 Limitations and Open Problems . . . 60
8.6 Conclusion . . . 60

9 Hardware for Compositionally Sparse Computation (with J. Bates) 61
9.1 The Structural Alignment Argument . . . 62
9.2 System and Numeric Model . . . 62
9.3 Mapping Compositional DAGs to 2D Meshes . . . 63
9.4 Sparse Attention on 2D Meshes . . . 64
9.4.1 KV-Stationary: Coarse-to-Fine Attention . . . 64
9.4.2 Q-Stationary: Systolic Streaming with Early Exit . . . 64
9.5 AM Quantization and Error Analysis . . . 65
9.6 Discussion: Why 2D Meshes Help . . . 65
9.7 Conclusion . . . 65

10 A Common Principle Underlying Diffusion Models and Transformers 66
10.1 Introduction . . . 67
10.2 Preliminaries . . . 67
10.3 Stepwise Universality of Autoregressive Predictors . . . 67
10.4 Stepwise Universality of Diffusion Predictors . . . 68
10.5 Discussion: Transformers and Diffusion as Stepwise Computation . . . 69
10.6 Technical Note: Gaussian Diffusion and Noisy One-Hot Encodings . . . 69
Appendix C: Gaussian Diffusion and Noisy One-Hot Encodings . . . 69
10.6.1 Setup: Forward Diffusion with One-Hot Embeddings . . . 69
10.6.2 Recovering the Active Coordinate . . . 70
10.6.3 Implementing the Turing-Step Update . . . 70
10.6.4 Training and Composition . . . 70

11 Lottery Ticket and Compositionality 71
11.1 Introduction . . . 71
11.2 The Geometry of Sparsity . . . 72
11.3 Approximate Lottery Ticket Theorem for Compositionally Sparse Functions . . . 72
11.4 Refinements and Empirical Connections . . . 73

Part III: Learning and Evolution 75

12 Implicit Regularization and Bits **76**
12.1 The Universal Currency: Bits, Geometry, and Noise . . . 77
12.1.1 Kolmogorov Complexity (The Language of Bits) . . . 77
12.1.2 Metric Entropy (The Language of Geometry) . . . 77
12.1.3 Rademacher Complexity (The Language of Statistics) . . . 78
12.2 The Bridge: Dudley's Chaining Integral . . . 78
12.3 Architecture as an Entropy Compressor . . . 78
12.4 Discussion: The Interplay of Architecture and Optimization . . . 79

13 Multiplicative Regularization Generalizes Better (based on work with R. Dubach and M. Abdallah) **80**
13.1 Introduction . . . 80
13.2 Background . . . 81
13.3 Theory . . . 82
13.4 Methodology . . . 82
13.5 Results . . . 83
13.6 Discussion . . . 85
13.7 Conclusion . . . 85

14 Concentration of Probability in Overparametrized Networks **86**
14.1 Main Results . . . 86
14.1.1 The Geometric Lemma: Existence of Flat Minimizers . . . 87
14.1.2 The Dynamics Lemma: SGDL as Langevin Diffusion . . . 87
14.1.3 The Concentration Lemma: Flat Beats Sharp . . . 87
14.1.4 The Timescale Law . . . 87
14.2 Detailed Analysis . . . 89
14.2.1 Synthesis of Geometric and Dynamic Views . . . 89
14.2.2 Critical Assessment . . . 89
14.2.3 Summary . . . 90

15 A Self-Assembling Cortical Circuit for Generalized Gradient Descent (with Qianli Liao and Liu Ziyin) **91**
15.1 A Minimal Synaptic Motif for Cortical Learning . . . 92
15.2 The Self-Assembling Learning Rule . . . 93
15.3 Theoretical Result: Emergence of Gradient Descent . . . 94
15.4 Biological Interpretation and Predictions . . . 94
15.5 An Example Implementation of SAL in Cortex . . . 95
15.6 Implications for Learning and Intelligence . . . 95

16 Zeroth-Order Evolutionary Post-Training for LLMs (with Y. Gan) **97**
16.1 Introduction . . . 97
16.1.1 Why ZO for LLMs? . . . 98
16.1.2 Historical Context and Theoretical Roots . . . 99
16.1.3 Chapter Overview . . . 99
16.2 Zeroth-Order Gradient Estimators . . . 99
16.3 Algorithmic Forms . . . 100
16.3.1 Random Search . . . 100
16.3.2 Distribution-Based Methods: CMA-ES and NES . . . 100

16.3.3 ES for LLMs . . . 101
16.4 Case Studies and Speculative Applications of ZO Evolution in LLMs . . . 101
16.4.1 Tuning Guardrails with Non-Differentiable Objectives . . . 102
16.4.2 Optimizing Mixture-of-Experts Dispatch . . . 102
16.4.3 Language Model Alignment without Differentiability . . . 102
16.4.4 Speculative: Meta-Controllers over Training Dynamics . . . 103
16.4.5 Summary of ZO-Friendly Structures . . . 103

17 Boolean Circuits and a Path to "Superintelligence" (by D. Koplow) 104
17.1 Boolean Circuits . . . 104
17.2 Learning . . . 105
17.2.1 Reasoning . . . 105
17.2.2 Measuring Learnability . . . 106
17.3 The Computational Cliff . . . 108
17.3.1 Problem Formulation . . . 108
17.3.2 The Statistical Obfuscation Construction . . . 109
17.3.3 Theoretical Guarantees . . . 110
17.3.4 Tool Synthesis . . . 113

Part IV: Extensions and Speculations 115

18 Consistency in Language Models 116
18.1 Definition of Consistency . . . 116
18.2 Contextual Representations and Associative Memory Hypothesis . . . 116
18.3 Contextual Consistency Hypothesis . . . 117
18.4 Theoretical Bound . . . 117
18.5 Illustration . . . 117
18.6 Conclusion . . . 117

19 Learning 2D views, recognizing 3D objects: what is the structure of embeddings 119
19.1 The 1994 Paradigm Shift . . . 119
19.2 Supporting Evidence: Psychophysics and Physiology . . . 119
19.3 The Geometry of the Embedding Space in Deep Networks . . . 120
19.3.1 Topology vs. Geometry . . . 120
19.3.2 Identity and Pose Disentanglement . . . 120
19.4 Compositional Sparsity and the DAG Architecture . . . 120
19.5 Modern Extensions: Multimodal Alignment and Scaling . . . 120
19.5.1 Unified 3D-2D-Language Embeddings . . . 120
19.5.2 The Embodied Turing Test . . . 120
19.6 Learning Invariant Object Representations from View Sequences . . . 121

20 More on Genericity Conjecture (with P. Beneventano) 123
20.1 Motivation and Informal Conjecture . . . 123
20.2 Preliminaries: Orthonormal Polynomial Framework . . . 123
20.2.1 Information Exponent of a Scalar Function . . . 124
20.2.2 Teacher–Student Single-Index Models . . . 124
20.3 Genericity as a Structural Property of Functions . . . 124
20.3.1 Genericity via Taylor Jets . . . 125

20.3.2 Thom–Mather Transversality and Structural Stability . . . 125
20.3.3 Invariance Under Smooth Coordinate Transformations . . . 125
20.4 Deep Networks and the Effective Information Exponent . . . 126
20.5 Formal Conjecture . . . 127
20.6 Evidence and Partial Progress . . . 128
20.6.1 Single-Index and GLM Results . . . 128
20.6.2 Quadratic and Polynomial Activations . . . 128
20.6.3 Residual Networks and Identity Skips . . . 128
20.6.4 Nonlinear Attention . . . 129
20.7 Research Program and Missing Ingredients . . . 129
20.8 Outlook . . . 130

21 Diffusion Models, Ill-Posed Inversion, and Generative Compression 131
21.1 Diffusion as a Forward Process and Ill-Posed Inversion . . . 131
21.2 Probabilistic Reverse Diffusion and Learned Scores . . . 132
21.3 Energy Landscape Viewpoint . . . 132
21.4 Inpainting and Generative Compression . . . 132
21.5 Discussion . . . 133

22 World Models Before Language 136
22.1 Sparse Compositionality as the Structural Prior of Evolution . . . 136
22.2 World Models as Predictive State-Space Systems . . . 137
22.3 Associative Memory as a Computational Primitive . . . 137
22.4 Hippocampal Replay as Approximate Inference . . . 138
22.5 What Evolution Discovered Before Language . . . 138
22.5.1 Predictive Physical Inference . . . 138
22.5.2 Social and Causal Modeling . . . 138
22.5.3 Compositional Perception and Action . . . 138
22.5.4 Associative Memory for Episodes and Scenes . . . 138
22.6 Language as an Overlay on a Pre-Existing Architecture . . . 139
22.7 Conclusion . . . 139

23 The Hippocampal Scaffold and Compositional Sparsity 140
23.1 Introduction: The Memory Palace . . . 140
23.2 The Variables of Experience . . . 141
23.3 The Indexing Mechanism: Pattern Separation . . . 141
23.3.1 Why this works . . . 141
23.4 Building the Scaffold Graph . . . 142
23.5 The Connection to Compositional Sparsity . . . 142
23.5.1 The Dense Trap of Sensory Learning . . . 142
23.5.2 The Sparse Solution via the Scaffold . . . 142
23.6 The Associative Turing Machine . . . 143
23.7 The Cortical Transfer: Systems Consolidation . . . 143
23.7.1 From Orthogonal Indexing to Manifold Learning . . . 143
23.7.2 Collapsing the Keys . . . 144
23.7.3 Retrieval Without the Scaffold . . . 144
23.7.4 The Semantic Trade-off . . . 144
23.8 A Unifying Computational Claim . . . 144

23.9 Relation to Existing Theories of the Hippocampus . . . 145
23.9.1 Cognitive Map Theory as Scaffold Construction . . . 146
23.9.2 Marr's Theory and Pattern Separation . . . 146
23.9.3 Indexing Theory and Pointer-Based Memory . . . 146
23.9.4 Complementary Learning Systems as a Change of Basis . . . 146
23.9.5 Successor Representations and Predictive Maps . . . 147
23.10 A Learnability Consequence of the Scaffold . . . 147
23.11 Beyond the Scaffold: Cortical Abstraction . . . 147
23.11.1 From Pointers to Generative Models . . . 148
23.11.2 The Result: Zero-Shot Navigation . . . 148
23.11.3 The Functional Hand-off . . . 148
23.11.4 Transformers as a Silicon Analog to the Hippocampal-Cortical Circuit . . . 149
23.12 Why the Cortex Is Still Needed: From Enumerated Constituents to Parametric Composition . . . 150
23.12.1 Constituent Functions in the Hippocampus . . . 150
23.12.2 What the Cortex Learns Beyond the Constituents . . . 151
23.12.3 A Hierarchy of Representations . . . 151
23.12.4 Conceptual Resolution . . . 151
23.12.5 Artificial Analogues of Cortical Abstraction . . . 152
23.13 Summary . . . 153
23.14 Connection with grid cells . . . 153

24 Reusable Sparse Compositionality 155
24.1 The Non-Uniqueness of Composition . . . 155
24.1.1 The Identifiability Problem . . . 155
24.2 The Constraint of Reusability . . . 156
24.2.1 Formulation: Shared Modularity . . . 156
24.3 Genericity as a Selector for Modularity . . . 156
24.4 Summary: The Trinity of Learnability . . . 157

25 What Is Missing in LLMs? 158
25.1 The Missing Foundation: World Models Before Language . . . 158
25.2 A Structural Limitation of LLMs . . . 159
25.3 From Diligence to Exploration . . . 159
25.4 The Memory Gap: From Context Windows to Turing-Efficient Memory . . . 160
25.5 The "Memento" Condition: Externalizing State . . . 160
25.6 The Barrier: Reusability, Genericity, and Sparsity . . . 161
25.6.1 Why LLMs Violate Genericity . . . 161
25.7 Memory as Generative Reconstruction . . . 161
25.8 The Causal Ladder . . . 161
25.9 The Algorithmic Role of Sleep . . . 162
25.10 The Symbol Grounding Problem: Maps Without Territories . . . 162
25.11 Conclusion . . . 162
25.12 Proposed Experiments . . . 163
25.12.1 Fragmented Embeddings and Reconstruction . . . 163
25.12.2 Diffusion in Memory Space . . . 163
25.12.3 Neurobiological Correlates . . . 163
25.12.4 Conceptual Prediction . . . 163

25.13 What Is Missing in Large Language Models: Compression of Composition 163
25.13.1 Representation Without Compression 163
25.13.2 Cortical Learning as Rule Compression 163
25.13.3 Depth Collapse and Learned Shortcuts 164
25.13.4 The Limitation of Fixed-Depth Architectures 164
25.13.5 Architectural Implications 164
25.14 Beyond LLMs: From Read-Only Models to Associative Turing Machines 165
25.14.1 The Transformer as a "Read-Only" ATM 165
25.14.2 Restoring Persistent State: Linear Recurrence and World Models 165
25.14.3 A Frontier: Online Plasticity and Rule Internalization 166
25.14.4 Closing the Loop Between Map and Territory 166

26 Computational role of eccentricity dependent cortical magnification 167
26.1 Introduction 167
26.2 Core Thesis 167
26.3 The Inverted Truncated Pyramid 167
26.4 The Magic Map: remaping to a Square Lattice 168
26.5 Hierarchical Decimation and Crowding 168
26.6 Visual Recognition via IP Fragments 168
26.7 Predictions and Empirical Alignment 168
26.8 The Geometry of the Magic Map 169
26.9 GELU Jets as Pooling Operators 169
26.10 Phase-Dependent Perception 169
26.11 Implications for Continual Learning 170

27 Nonlinear Scale Space 171
27.1 Introduction 171
27.2 A Minimal Formal Model 172
27.3 Relation to Classical Scale Space 172
27.4 The Jet of the GELU Activation 172
27.5 Phase I: The Discriminative Pipeline (< 100 ms) 174

28 Is Efficiently Computable Intelligence Different from Unconscious Intelligence? 175
28.1 Efficient Computability and Sparse Compositionality 177
28.2 Approximation, Optimization, and Generalization 178
28.3 Relation to Prior Theories of Deep Learning 178
28.4 Discussion 180

29 Mixture of Experts 182
29.1 Introduction 182
29.2 Shallow MoE Networks (Low-Dimensional Structure) 182
29.3 Deep MoE Networks (Compositional Sparsity) 183
29.4 Unified Insights 183
29.5 Connections with Compositional Sparsity Framework 184
29.5.1 The Core Alignment: Hierarchical Decomposition 184
29.5.2 Overcoming the Curse of Dimensionality 184
29.5.3 The "Exponential Capacity" Extension 185
29.5.4 Direct Mapping of Terminology 185

29.6 Summary . . . 185
29.7 DeepSeek-V3 MoE Architecture and Training Methodology . . . 185
29.7.1 Architectural Composition . . . 185
29.7.2 Training Dynamics . . . 186
29.7.3 Summary of Specs . . . 187

30 Takens Theorem, RNNs and Associative Turing Machines 188
30.1 Takens' Embedding Theorem . . . 188
30.1.1 Relevance to RNNs . . . 188
30.2 Autonomous RNN model . . . 189
30.2.1 Simulation of finite-state machines . . . 189
30.2.2 Turing machines restricted to a finite time horizon . . . 189
30.2.3 Why the time bound is essential . . . 190
30.2.4 Comparison with Associative Turing Machines . . . 190

Appendix 192

A Appendix: Stability, ERM, and the Foundations of Learnability 193
A.1 Learning setup and ERM . . . 193
A.2 CV_{loo} stability . . . 194
A.3 Stability and generalization . . . 194
A.4 Stability as a modeling requirement . . . 194

B TechnicalNote: Compositionality in Machine Learning and Physics 196
B.1 Introduction . . . 196
B.2 Definitions and Setup . . . 198
B.3 From Efficient Computation to DAGs . . . 199
B.3.1 From arithmetic circuits to neural DAGs . . . 200
B.3.2 Computation as compositional structure . . . 200
B.4 ML Approximation Theorem . . . 200
B.4.1 Local constructive approximation with explicit constants . . . 201
B.4.2 Global error propagation and optimal budget . . . 202
B.4.3 The theorem . . . 203
B.4.4 Depth separation . . . 203
B.5 Finite-Horizon Compilation: From Uniform Simulators to Algorithmic Compositionality . . . 204
B.5.1 Setup . . . 204
B.5.2 Compilation theorem (algorithmic, safe) . . . 204
B.6 Learning Theory and Optimization . . . 205
B.7 Limits of the Framework . . . 206
B.8 Conclusion . . . 208

C Technical Note: A Group-invariant Johnson-Lindestrauss Lemma 210
C.1 Introduction . . . 210
C.2 Detailed Solution . . . 211
C.2.1 A finite control set and inner-product preservation . . . 212
C.2.2 Discretizing Haar averages over G . . . 212
C.2.3 From scalar errors to CDF (KS) errors . . . 212

C.2.4 Averaging over templates and a uniform bound on S . . . 213
C.2.5 Conclusion (finite-sample "JL for the paper's invariant metric") . . . 214
C.2.6 Corollaries . . . 214
C.2.7 Remarks . . . 214

D Interlude: Most Real Numbers Do Not Exist 216
D.1 Mathematical Preliminaries . . . 216
D.2 Computability and Effective Existence . . . 217
D.3 Definability and Symbolic Description . . . 217
D.4 Operational Discretization Under Finite Resources . . . 218
D.5 Why Most Real Numbers Do Not Exist . . . 218
D.6 Technical Note: Cardinality, Complexity, and Randomness . . . 218
D.6.1 Cardinality Review . . . 218
D.6.2 Kolmogorov Complexity . . . 219
D.6.3 Non-Computable Transcendentals . . . 219
D.6.4 Physical Measurement . . . 219
D.6.5 Definability . . . 219

E Potential Projects in Zeroth-Order Optimization: Directed Mutations 220
E.1 Directed mutations: binary-search–like efficiency . . . 220
E.1.1 Setting and notation . . . 220
E.2 Genes $\rightarrow$ subgenes as a binary tree of traits . . . 221
E.2.1 Hierarchical mutation model . . . 221
E.2.2 Sparse-error localization via adaptive subtree queries . . . 221
E.2.3 Greedy hierarchical descent for additive convex loss . . . 222
E.3 Context and prior art (concise) . . . 222
E.3.1 Biology: hierarchical gene regulation . . . 222
E.3.2 Evolutionary computation: linkage and hierarchy . . . 223
E.4 Implications and extensions . . . 223

F Faster Attention 224
F.1 The Adjacency Distance Chain . . . 224
F.2 Utilizing the Lower Bound for Pruning . . . 224
F.3 Formal Constraint: The Metric Continuity Hypothesis . . . 225
F.4 Experimental Verification . . . 225
F.5 Proposed Algorithm: Recursive Distance Bounding . . . 226
F.6 Comparison with Global Clustering and Hashing Approaches . . . 226
F.6.1 LSH and k-means Clustering . . . 226
F.6.2 Global vs. Recursive Local Metrics . . . 226
F.6.3 Summary of Complexity Drivers . . . 227

G Appendix: The Hippocampal Scaffold and Compositional Sparsity 228
G.1 Introduction: The Memory Palace . . . 228
G.2 The Variables of Experience . . . 228
G.3 Mathematical Foundations of the Hippocampal Index . . . 228
G.3.1 The Top-K Projection Mechanism . . . 229
G.3.2 Locality-Sensitive Hashing (LSH) . . . 229
G.4 Building the Scaffold Graph . . . 229

G.5 Connection to Existing Theories . . . 229
G.6 Conclusion . . . 230

H The Imitation Game 2.0 (Idea by Dan Mitropolsky) 231
H.1 Introduction . . . 231
H.2 The Containment Principle . . . 232
H.3 The Protocol . . . 232
H.4 The Metric: Simulation Distance . . . 233
H.5 Case Study: The "Bad Code" Test . . . 233
H.6 Conclusion . . . 234

I More on RNNs and Turing Machines 235
I.1 The Limits of Autonomous RNNs . . . 235
I.1.1 The Precision Argument . . . 235
I.2 The Discrete State Hypothesis . . . 235
I.2.1 Precision vs. Boundedness . . . 236
I.3 The Power of Separation: State vs. Tape . . . 236
I.3.1 Computational Hierarchy . . . 236
I.4 Conclusion: The "Two-RNN" Architecture . . . 236
I.4.1 The Fundamental Distinction: Active vs. Passive . . . 237
I.4.2 The Correct Architecture: Controller + Matrix . . . 237
I.5 Intuition: Internal vs. External Memory . . . 237
I.5.1 1. The "Index Card" vs. The "Scroll" . . . 238
I.5.2 2. The "Space of Functions" (Computational Power) . . . 238
I.6 Analogy: Transformers as Turing Machines . . . 240
I.6.1 1. Attention as the Tape (Memory) . . . 240
I.6.2 2. MLP as the State (Controller) . . . 240
I.6.3 Summary Mapping . . . 240
I.7 Correspondences in the Associative Turing Machine (ATM) . . . 240
I.7.1 1. Component Mapping . . . 241
I.7.2 2. Solving the Precision Problem . . . 241
I.8 Conclusion. . . . 241

Glossary of Core Concepts 244

References 248

How to Read This Collection

This volume is not a conventional monograph. It is a collection of essays—some conceptual, some technical, some speculative—written at different times but revolving around a shared set of themes. It has evolved into something close to an intellectual atlas: an ongoing effort to map the principles that underlie modern artificial intelligence and the physics of natural intelligence. Several chapters are proposals for research projects that may be completed before too long.

Two ideas appear repeatedly throughout the collection, forming the spine of the argument:

- **Sparse compositionality**: the view that complex functions, behaviors, and representations arise from the hierarchical composition of a small number of reusable parts. This principle explains approximation, hierarchy, modularity, transfer, interpretability, and the unreasonable effectiveness of deep networks.
- **Genericity**: the observation that the real functions we want to learn in the world are not arbitrary high-complexity objects. Learnable functions must show stability with respect to the training set and invariance to simple transformations of the coordinate systems used for their definition. This implies that meaningful structure leaves persistent low-degree footprints, enabling gradient-based optimization.

The collection is organized into four parts:

Part I: Principles explores the mathematical inevitability of the two core ideas above. It argues that intelligence is not a grab-bag of heuristics but a solution to specific physical constraints.

Part II: Computation & Hardware instantiates these principles in specific architectures, moving from Associative Turing Machines to the efficiency of Boolean functions.

Part III: Learning & Evolution examines the dynamics of learning in deep networks and evolutionary strategies.

Part IV: Extensions & Speculations takes the most risks. It treats biology not as a metaphor, but as a constraint. Here, we argue that mechanisms like "World Models" and "Dreaming" are not just biological quirks but computational necessities.

Suggested Paths

Readers interested primarily in the **conceptual picture** may prefer to begin with:

- The *Prologue*,
- Chapter 1: *Intelligence as Associative Memory*,
- Chapter 22: *World Models Before Language*.

Readers seeking **computational foundations** may start with:

- Chapter 3: *Associative Turing Machines*,
- The essays on compositional optimization in Part I,
- The appendices, which now contain rigorous proofs regarding the Johnson-Lindenstrauss lemma and group invariance.

Readers curious about **broader implications** or the future of AI may enjoy:

- Chapter 25: *What is Missing in LLMs?*,
- Appendix D: *Most Real Numbers Do Not Exist*,
- The concluding essay on principles and open questions.

As with any collection of essays, some ideas appear more than once, viewed from different angles. This repetition is intentional: deep principles re-emerge across different domains. The unity of the volume lies not in a single narrative arc but in the convergence of these diverse perspectives on the same underlying structure of intelligence. Finally, several chapters include the name of a human co-author with whom I discussed extensively the content of the chapter. Even so. the responsibility for mistakes and omissions in the written text is on me and Gemini!

Last but not least: this is, for the time being, just a draft, often a draft of proposals of research project. It contains errors; there are missing references and missing acknowledgments. With time and feedback from you they will be corrected! This is ready to be a "working notebook" for the field. It provides a counter-narrative to the "scale is all you need" crowd.

Note on Co-Authorship: An Experiment

This book is an experiment in a new kind of scientific collaboration. It was co-authored by a human scientist and two Large Language Model (specifically, GPT and Gemini).

In the history of science, we often speak of "tools for thought"—notation, calculus, the computer. But until recently, those tools were passive. They did not push back. This collection represents a shift toward active *cognitive amplification*. Throughout the writing process, the AI did not merely act as a scribe or a ghostwriter; it functioned as a sparring partner, a synthesizer, and occasionally, a mirror. Gemini also drafted figures which are far from perfect but serve as good placeholders.

The Role of the Model

The core ideas—Sparse Compositionality, Genericity, the biologically constrained definition of intelligence—are mine. They stem from decades of research at MIT and the Center for Brains, Minds and Machines (CBMM). However, the *articulation* of these ideas was often a dialectic process.

I would feed the model a raw intuition or a rough technical note (e.g., "Language is just a subtitle track for a world model"). The model would expand it, connect it to adjacent concepts in the history of computation (Turing, Post, von Neumann), and return a draft. We would then iterate.

This process revealed an interesting property of current AI: it is a machine of *genericity*. Left to its own devices, the model tends to "smooth" ideas. It gravitates toward the consensus, the balanced view, the polite academic tone. It often tried to soften my more radical claims about biology—for instance, the insistence that diffusion models are biologically implausible, or that scaling laws alone will hit a wall.

I found myself constantly having to "roughen" the text back up—to re-inject the asymmetry, the specific bet, and the falsifiable conjecture. In this sense, the authorship process mirrored the very principles discussed in the book: the AI provided the *generic* compositional capability, while the human provided the *sparse*, high-frequency logical jumps that define scientific novelty.

Why This Matters

This collaboration is not just a novelty; it is a data point in the study of intelligence itself. If, as this book argues, intelligence relies on the composition of reusable modules, then LLMs have mastered the modules of *syntax* and *rhetorical structure*. They have learned the "shape" of a scientific argument. But they famously lack the "World Model"—the grounding in physical reality that allows a primate to understand gravity without words.

Writing this book with an LLM confirmed that hypothesis. The model could write fluently about "World Models," but it could not *verify* consistency without my guidance. It could hallucinate a citation as easily as it could summarize a theorem. It possessed language, but not truth.

The Future of Scientific Writing

I believe this model— Human + AI— will become the standard for scientific inquiry, at least in the near future. It allows a researcher to move faster, to synthesize broader fields of literature, and to maintain a unity of voice across disparate topics. But it requires a new kind of vigilance. We must guard against the "smoothing" effect of these models, which risks homogenizing scientific thought into a polite, generic mean.

This collection, therefore, is an artifact of 2026: a testament to what machines can do, and a reminder of what, for now, only biological brains can provide.

Tomaso Poggio
Lake Sabrina, Needham, Massachusetts
January 2026

Part I: Principles and Foundations

CHAPTER 1

Intelligence as Associative Memory

What if the secret of modern AI is not a mysterious spark, but a very old idea equipped with new machinery—associative memory? Transformers and large language models (LLMs) look less like monolithic "brains" and more like carefully organized banks of memories addressable by content [193]. This perspective connects classical radial basis function (RBF) networks—smooth, template-based responders from the 1980s [24]—to modern attention mechanisms. With enough attention heads working together, these systems begin to resemble a random-access Turing machine: they can retrieve the relevant context, bind it to the current problem, and write back improved representations—iteratively and at scale.

Figure 1.1: Intelligence as an Associative Memory

Long before deep networks, RBF networks modeled cognition as template matching: a set of prototypes is placed in feature space, and responses grow stronger as inputs approach a prototype—"association by similarity." Intelligence, in this view, is largely the ability to retrieve the right neighbors and combine them appropriately.

Transformers generalize this idea dramatically. Each attention head behaves like a learned, anisotropic RBF—or, in the terminology of Poggio and Girosi [152], a *Hyper-RBF*:

- Query ≈ the current cue or question.
- Keys ≈ stored templates.
- Values ≈ retrieved contents associated with matching templates.
- Attention weights ≈ soft, learned similarity—anisotropic, content-aware, and context-dependent.

Unlike classical RBFs with fixed centers and metrics, attention templates are dynamic: they move and recombine across layers. Individual heads specialize—some track syntax, others entities, others long-range dependencies—yielding an associative memory system with thousands of adaptive kernels operating in parallel.

Operational schema (informal). Modern LLM blocks iterate a simple loop:

1. *Content-addressable read:* form a query from the current state and retrieve values from a key–value store (self-attention).
2. *Local update:* apply a pointwise transformation (residual MLP) that writes information into the hidden state.
3. *Reuse:* later layers re-address the updated state, compounding small steps into complex behavior.

A rigorous associative variant of this loop suffices to simulate classical computation; see the *Associative Turing Machine* (ATM) in Chap. 3, Thm. 1.

1.1 Building a Turing Machine with Attention Heads

An attention block in a transformer consists of multiple Hyper-RBFs—one per attention head. Each Hyper-RBF receives as input the current token representation, while its centers correspond to representations of previous tokens. A single head can retrieve a relevant neighbor; multiple heads can coordinate read–write cycles over a shared latent memory:

- **Random access.** Heads allow the model to jump to any position whose content matches the query, not merely the next symbol in a sequence. This is the defining property of RAM-like computation.
- **Working memory.** Stacks of attention layers act as recurrent "micro-programs": read from content-addressable memory, transform, and write results into hidden states that later layers can re-read.
- **Learned controllers.** Through next-token prediction, the network learns a distributed controller that selects which heads to activate, what to retrieve, and how to update internal scratchpads—entirely from data. Over time, it discovers algorithmic patterns such as copying, counting, bracket matching, reference tracking, and multi-step reasoning.

In short, with sufficient depth and heads, attention implements a learned, differentiable Turing machine with RAM-like lookup. The formal counterpart is the ATM model in Chap. 3, where attention-style reads implement memory M_2 and residual MLPs implement the local update M_1. Thm. 1 proves that any deterministic Turing machine can be simulated with only polynomial overhead in associative reads and updates.

Cost line. Per computational "step," the pattern is one content-addressable read plus one local update. In a dense transformer processing a sequence of length n, this yields $O(n^2)$ attention time per layer (or $O(n)$ under effective sparsity), with $O(n)$ memory for the key–value cache; depth counts steps.

1.2 Connections to Other Models

Diffusion models [77] and state-space models (SSMs) [66] echo the same principle: intelligence as learned transitions over latent states.

Diffusion models learn a small denoising operator—a local transition—that, when iterated, reconstructs structure from noise. As we will see in another chapter this is similar to autoregressive learning in transformers: the noising step provides a training set to leanr constituent function of the oveall composiitonal function mapping noise into a pattern.

Mamba-style SSMs learn linear-plus-gated state updates that stream efficiently through sequences. They trade global random access for fast, structured memory, but the motif remains the same: a learned transition repeatedly applied builds complex behavior.

Across transformers, diffusion models, and SSMs, the unifying theme is computation via small, reliable steps (each one corresponding to a constituent function of the overall compositional function) composed deeply over time—intelligence as iterated associative updates. For a formal abstraction of this read/update pattern, see the ATM interface in Chap. 3.

1.3 Clarifying the Associative-Memory Lens

To sharpen the analogy, note that a classical extension of RBFs—sometimes called Hyper-RBFs in the work of Poggio and Girosi—computes responses of the form

$$\phi_i(x) \;=\; \exp\Big(-\tfrac{1}{2}\,(x-\mu_i)^\top \Sigma_i^{-1}(x-\mu_i)\Big),$$

with centers μ_i and (possibly anisotropic) covariances Σ_i. In transformers, a single attention head computes weights

$$\alpha_{ij} \;=\; \mathrm{softmax}_j\!\left(\frac{q_i^\top k_j}{\sqrt{d}}\right),$$

which can be interpreted as an adaptive, directional similarity between a query q_i and keys $\{k_j\}$. The Hyper-RBF perspective emphasizes that attention not only selects neighbors by content, but also composes *values* v_j via $\sum_j \alpha_{ij} v_j$, enabling information binding and controlled routing across layers. Unlike fixed kernels, (q_i, k_j, v_j) are learned and contextually modulated at each layer, yielding a programmable associative system.

1.3.1 HyperBF Equivalence: Attention as a Normalized HyperBF

Recent work [151] shows that if $\|q\| = \|k_i\| = 1$ (or more generally after removing constant offsets in the exponent), then the softmax of the dot product is exactly equivalent to a Gaussian RBF kernel. Formally, one can rewrite the attention weights as

$$\mathrm{Attn}(q, k_i, v_i) = \sum_i \frac{\exp(-\|q-k_i\|^2/(2\sigma^2))}{\sum_j \exp(-\|q-k_j\|^2/(2\sigma^2))}\, v_i. \tag{1.1}$$

This is exactly the "normalized HyperBF" form:

$$\phi(x) = \sum_{i=1}^{N} \frac{c_i \, K(\|x - t_i\|_W^2)}{\sum_j K(\|x - t_j\|_W^2)}. \tag{1.2}$$

Thus, an attention head is *identical* to a HyperBF unit:

$$\boxed{\text{Attention Head} = \text{Normalized HyperBF Network.}}$$

The Transformer's most distinctive component is therefore a kernel-based associative memory.

1.4 Multi-Head Requirements for RAM-like Computation

Simulating random-access behavior typically requires:

1. **Addressing capacity:** multiple heads can pursue distinct hypotheses (e.g., short-range syntax vs. long-range co-reference), increasing parallel lookup bandwidth.

2. **Inter-head coordination:** later layers can fuse evidence from earlier lookups, akin to multi-ported memory where read results influence subsequent queries.

3. **Write-back via states:** while the external sequence is read-only, internal hidden states act as a scratchpad. Layer-wise transformations serve as differentiable "writes" that downstream layers can re-read.

This architecture approximates a controller+RAM setup where the controller is distributed across layers and heads, and the RAM is realized by content-addressable keys/values and persistent hidden activations.

1.5 On Diligence, Creativity, and Exploration

The "Diligent Learner" perspective [173] emphasizes systematic accumulation of competencies under a stable objective. To move beyond local optima and enable novelty, augment training with:

- **Exploration bonuses** (curiosity-driven objectives) to valorize informative states.

- **Evolutionary search** over architectures or prompts to diversify computational primitives (e.g., heads, value pathways).

- **Self-curricula** that stage tasks from easy to hard, ensuring stable acquisition while retaining headroom for innovation.

In practice, diligence (exploitation) and creativity (exploration) are complementary regimes of the same associative engine.

1.6 Question: How Can Transformers Be So Consistent?

Question. It is intuitive that transformers, functioning as associative memories, can retrieve related facts, statements, words, or sentences. What is less intuitive is that they manage to be so *consistent* in what they retrieve when prompted by a given prompt. How is this possible?

Answer. Transformers are trained as next-token predictors, not as explicit associative memories enforcing global consistency. Yet, when prompted, they often produce coherent multi-sentence responses that remain self-consistent over long spans. This property emerges from the interaction of three underlying principles.

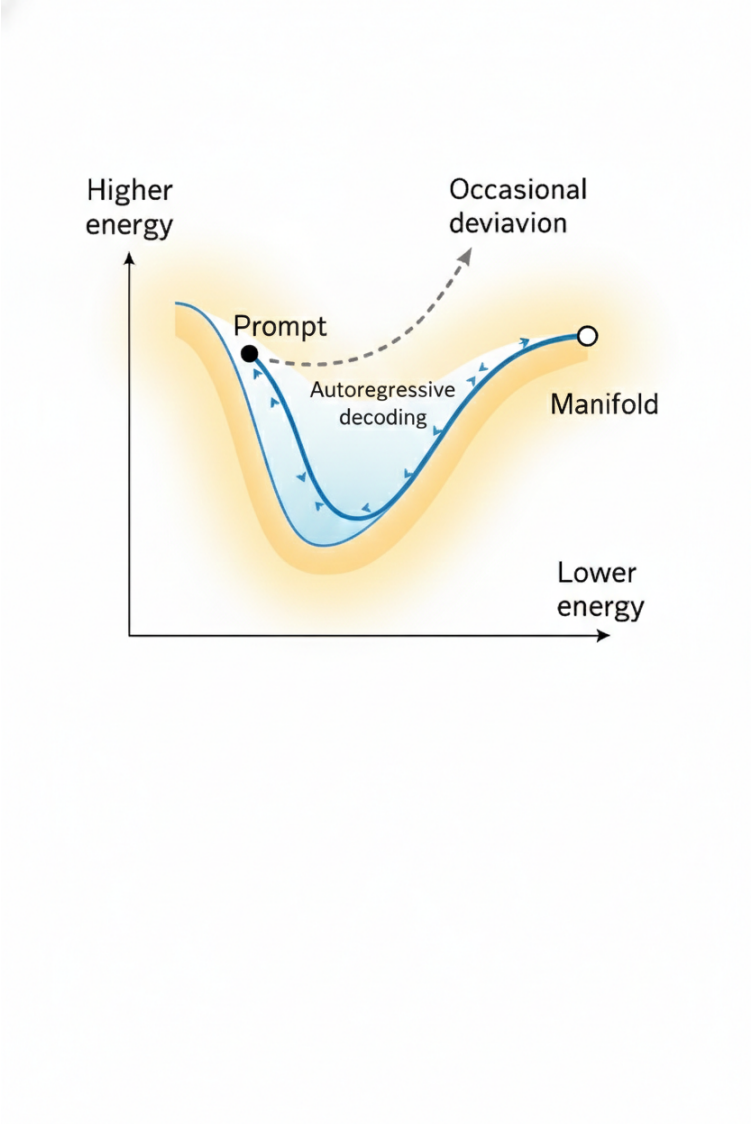


Figure 1.2: Schematic of manifold-constrained generation. A prompt initializes the hidden state in the transformer's energy landscape. Autoregressive decoding moves downhill along the manifold of coherent continuations, staying within an attractor basin of semantically and syntactically consistent states. Occasional deviations (e.g., high-temperature sampling) can push the trajectory outside the manifold, leading to incoherence or hallucination.

1.6.1 Global Consistency from Shared Latent Geometry

Each token representation depends on all previous tokens through the self-attention mechanism. During training, the network learns a self-consistent geometry of latent embeddings, in which facts, syntactic relations, and semantic associations lie on low-dimensional manifolds. When a prompt is given, the hidden state—a superposition of keys, queries, and values—is confined to a region of this manifold. Subsequent tokens are produced within the same geometric region, preserving coherence until a stochastic deviation moves the sequence elsewhere.

This geometric constraint explains why transformers act as associative memories: attention performs soft retrievals over an embedding space whose structure encodes factual and syntactic consistency.

1.6.2 Self-Reinforcing Autoregression

At each generation step, the produced token is appended to the context and re-encoded. If early outputs are coherent with the manifold, later predictions remain constrained to that region, forming a self-reinforcing loop:

$$h_{t+1} = f_\theta(h_t),$$

where h_t represents the internal state and f_θ the learned dynamics. As long as f_θ maps a neighborhood of coherent states onto itself,

$$f_\theta(\mathcal{M}) \subseteq \mathcal{M},$$

the system stays near an invariant manifold $\mathcal{M}$ of consistent continuations. Hallucinations occur when noise or sampling temperature pushes the state outside this manifold.

1.6.3 Emergent Consistency as Manifold-Constrained Recall

In summary, transformers behave as massive associative memories endowed with an implicit energy landscape over text sequences. Prompting fixes part of a pattern; decoding completes it by descending the energy surface within a structured manifold of coherent narratives. Consistency emerges not from an explicit rule but from the geometry of the learned embedding space and the self-stabilizing nature of autoregressive dynamics.

Consistency, in this view, is an emergent property of manifold-constrained associative recall.

1.7 Memories

Comparison: Kanerva's Sparse Distributed Memory and hippocampal scaffolding. Kanerva's *Sparse Distributed Memory* (SDM) [90] provides a mathematical model of an associative memory operating in a high-dimensional binary space. Each memory location corresponds to a randomly chosen address vector, and recall is achieved by aggregating the stored contents of those locations lying within a fixed Hamming radius of the input cue. The SDM thus implements a high-dimensional nearest-neighbor retrieval rule followed by averaging—a linear associative mechanism capable of pattern completion and generalization.

In contrast, the *hippocampus* can be viewed as an anatomical and functional realization of an associative scaffold that interacts hierarchically with cortical areas [122, 125]. The dentate gyrus and CA3 subfields act as sparse coding and auto-associative modules, respectively: dentate granule cells expand and sparsify cortical inputs, while recurrent collaterals in CA3 implement attractor

dynamics analogous to Kanerva's distributed retrieval [82]. However, unlike SDM, hippocampal circuits are embedded in a multilayer system in which the retrieved pattern serves as a scaffold for cortical consolidation and reconstruction over time. In this sense, Kanerva's SDM formalizes the *local, static* properties of associative recall in a fixed high-dimensional space, whereas hippocampal scaffolding realizes a *dynamic, hierarchical* memory system that not only retrieves associations but also builds and reorganizes compositional representations across multiple levels of abstraction.

Storage efficiency. From an information-theoretic standpoint, Kanerva's Sparse Distributed Memory trades efficiency for robustness: its distributed addressing and heavy overlap between stored patterns yield roughly 0.1–0.2 bits of recoverable information per synapse, comparable to classical Hopfield networks [82]. The hippocampal scaffold, by contrast, combines sparse coding in dentate gyrus with structured auto-association in CA3, reducing interference between unrelated memories and increasing efficiency to an estimated 0.5–1.0 bits per synapse. This higher storage efficiency reflects biological mechanisms of sparsification and hierarchical consolidation, suggesting that cortical–hippocampal memory may implement an optimized, multi-layer variant of Kanerva's associative architecture.

Single-layer versus multi-layer compositional memory. Kanerva's Sparse Distributed Memory can be viewed as a *single-layer associative map*: it implements direct retrieval in a fixed high-dimensional address space, effectively performing one-step interpolation among stored patterns. Its architecture is flat—there is no hierarchy or composition of intermediate representations—so it supports local generalization but not the construction of new representations from previously stored parts. In contrast, the hippocampal scaffold embodies a *multi-layer compositional memory*. Inputs from entorhinal cortex are expanded and sparsified in dentate gyrus, auto-associated through recurrent CA3 dynamics, and then compressed and remapped in CA1 before being reinstated to cortex. Each stage transforms and recombines representations from the preceding one, forming a deep compositional DAG analogous to the architecture of a sparse multi-layer network. This hierarchy enables both pattern completion and the flexible recombination of episodic components—functions beyond the scope of a single-layer associative memory such as SDM. In this sense, the hippocampal scaffold may be viewed as a *biological precursor of transformer-like multi-layer associative systems*, implementing in neural hardware the same principle of hierarchical composition that underlies modern deep architectures.

Parallel between hippocampal memory and transformer architectures. At a computational level, the hippocampal memory circuit and transformer architectures share three key principles. First, both perform *associative retrieval*: CA3 recurrent dynamics retrieve stored patterns based on similarity to a partial cue, analogous to the attention mechanism retrieving values by the similarity between queries and keys. Second, both implement a *multi-layer compositional pipeline*. The hippocampal pathway—entorhinal cortex $\rightarrow$ dentate gyrus $\rightarrow$ CA3 $\rightarrow$ CA1—forms a stack of low-arity transformations, closely paralleling the alternating attention and feed-forward (MLP) blocks of a transformer. Each stage refines and recombines distributed representations from the previous one, yielding deep compositional computation. Third, both systems maintain and update *context*: in transformers through autoregressive conditioning on past tokens, and in the hippocampus through recurrent replay and entorhinal feedback that integrate episodic context over time. In this formal sense, the hippocampus can be regarded as a biological implementation of a transformer-like associative architecture, one that achieves hierarchical composition and contextual sequence generation through local neural plasticity rather than backpropagation.

CHAPTER 2

A Historical Reflection on Associative Memories

This chapter is a historical and conceptual essay that follows the thesis introduced in Chapter 1. While Chapter 1 frames intelligence directly as an associative process, the present chapter traces the origins of that idea across biological evolution, early theoretical neuroscience, and modern artificial intelligence. Several later chapters revisit these themes from different technical perspectives; the repetition is intentional.

2.1 Associative Memory as the Core of Intelligence

A rarely acknowledged but deep conceptual continuity runs from the earliest associative memories in theoretical neuroscience to the core mechanisms of modern artificial intelligence. This essay argues that associative memory is not an obsolete concept but a central computational primitive underlying both biological and artificial intelligence. We trace a continuous line connecting (i) early formal models of associative memories in the 1960s and 1970s [200, 98]; (ii) the role of monosynaptic reflexes and episodic binding in the evolution of animal intelligence; (iii) the emergence of metric-based radial basis function (RBF) and hyperBF networks as powerful parametric associative memories [152]; (iv) hippocampal indexing and scaffolding models; and (v) the attention mechanism in modern transformers, which can be interpreted mathematically as a high-dimensional associative memory—a normalized, learnable hyperBF network [193].

2.2 Introduction

Associative memory—the ability to bind one pattern to another and later retrieve the bound information from a full or partial cue—is among the earliest and most fundamental ideas in neural computation. Yet despite its centrality, it is often treated as a historical precursor to modern deep networks rather than as a principle that continues to shape state-of-the-art AI systems and biological intelligence.

The core claim of this essay is that associative memory may be the primitive operation from which intelligence emerges. Many major advances in both biological evolution and artificial intelligence can be understood as refinements, extensions, or compositions of associative mechanisms.

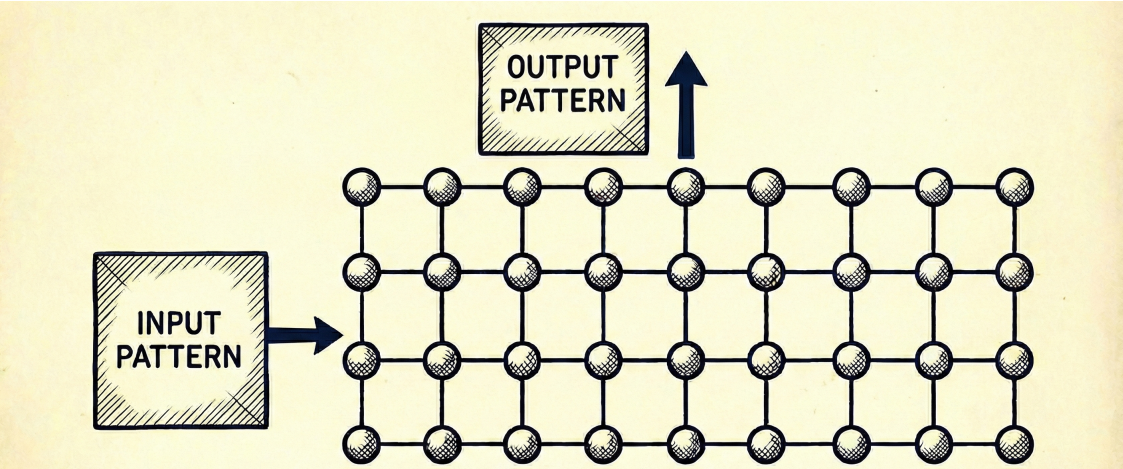


Figure 1. Schematic representation of a non-holographic associative memory, after Willshaw & Longuet-Higgins, *Nature* (1969).

Figure 2.1: The lineage of associative memory.

2.3 Historical Background

The modern notion of associative memory—a system storing pairs of patterns (x^μ, y^μ) such that presentation of x^μ retrieves y^μ—emerged in the late 1960s and early 1970s. These early models were explicitly concerned with capacity, robustness, and biological plausibility.

2.3.1 Early Heteroassociative Models

Willshaw and Longuet-Higgins introduced sparse binary heteroassociative memories with one-shot learning and feedforward recall [200]. Kohonen subsequently formulated associative memory in linear algebraic terms via correlation matrices [98],

$$W = \sum_\mu y^\mu (x^\mu)^\top, \qquad \hat{y} = Wx,$$

a formulation that already anticipates kernel regression and modern attention when normalization is introduced.

2.3.2 Nonlinear Associative Recall and Polynomial Expansions

A conceptually central contribution is Poggio's formulation of associative recall as an operator estimation problem (1973–1975) [144]. Given paired data matrices X and Y, the goal is to approximate

$$Y \approx \Theta(X)$$

via a polynomial expansion

$$Y = L_0 + L_1(X) + L_2(X, X) + \cdots + L_k(X, \ldots, X),$$

where L_ℓ are symmetric multilinear operators. The linear term $L_1 = YX^\dagger$ recovers Kohonen's memory, while higher-order terms provide systematic nonlinear corrections.

Defining the residual after removing the approximation up to order $k-1$ as $E_{k-1} = Y - \sum_{\ell=0}^{k-1} L_\ell(X, \ldots, X)$, the optimal kth-order correction is given by:

$$L_k = E_{k-1} C_k^\dagger,$$

where C_k is the "k-way" lifted tensor of inputs defined by $(C_k)_{\alpha_1 \cdots \alpha_k, j} = \prod_{t=1}^{k} X_{\alpha_t j}$.

Implicit introduction of polynomial kernels. Viewed retrospectively, this framework implicitly introduces polynomial feature maps and kernels. The polynomial expansion can be reinterpreted as an explicit feature map $\Phi_k(x)$ containing all monomials of degree up to k. The corresponding kernel

$$K_k(x, x') = \langle \Phi_k(x), \Phi_k(x') \rangle = (1 + x^\top x')^k$$

appears here well before it became standard in the Support Vector Machine literature. Conceptually, this framework unifies linear associative memories with nonlinear function approximation and lies closer to modern attention mechanisms than later autoassociative Hopfield networks.

2.3.3 Correlation Memories, RBFs, and HyperBFs

Palm formalized correlation-based associative memories with capacity and error-correction guarantees [138]. Later, Poggio and Girosi introduced radial basis function networks [152],

$$f(x) = \sum_{\mu} \alpha_\mu \phi(\|x - x^\mu\|),$$

and their generalization to **HyperBFs** with learned metrics. These architectures realize smooth, metric-based associative mappings.

2.3.4 Hopfield Networks as a Special Case

Hopfield networks introduced attractor dynamics for autoassociative recall [82]. While historically influential, they are a relatively narrow subclass: they are purely autoassociative, rely on symmetric weights, and lack the content-addressable key–value structure central to transformers. Transformers trace their lineage far more directly to the heteroassociative (Willshaw, Kohonen) and metric-based (RBF) models.

2.4 Genericity as the Hidden Enabler of Associative Memories

A striking feature of the associative memory models surveyed in this chapter is that they work remarkably well in practice despite weak worst-case guarantees. This empirical success has a common, often unstated explanation: the targets they operate on are generic.

In Poggio's polynomial operator expansion, low-order terms capture most of the signal, while higher-order corrections rapidly diminish in magnitude. Similarly, in correlation memories and RBF networks, nearby patterns dominate recall. These facts are not architectural accidents; they reflect a property of the world.

Genericity—the absence of exact algebraic cancellations and pathological symmetries—ensures that meaningful associations leave stable, low-degree statistical footprints. As a result, associative recall is driven by strong linear and quadratic components, with higher-order terms acting as small refinements rather than essential structure.

From this perspective, associative memory is computationally viable not because it can represent arbitrary functions, but because the functions of interest in perception, action, and cognition are generically simple when viewed in the right coordinates. This insight anticipates modern results on optimization cliffs and staircases: when low-degree structure is present, learning and retrieval proceed smoothly; when it is absent, they fail catastrophically.

Thus, genericity is the silent partner of associative memory, explaining why these models have repeatedly reappeared across biology, neuroscience, and artificial intelligence.

2.5 Evolutionary Perspective

From an evolutionary standpoint, associative memory is the simplest mechanism capable of linking perception to action. Monosynaptic reflex arcs implement the mapping

$$x_{\text{sensory}} \mapsto y_{\text{motor}},$$

the biological analog of heteroassociative memories.

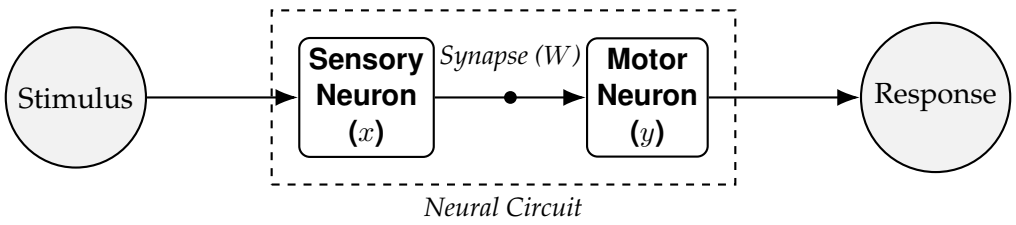


Figure 2.2: The Monosynaptic Reflex Arc. In this biological model of a simple associative memory, a sensory input signal (x) is transmitted across a synapse (W) to trigger a motor output (y).

The introduction of feedback pathways enabled refinement and optimization of intermediate representations. Recent models such as Self-Assembling Learning (SAL) illustrate how local plasticity can self-organize into optimization circuits [145].

2.5.1 From Reflexes to Multi-Stage Adaptive Hierarchies

The evolutionary narrative likely follows this progression:

1. **Reflexes (single synapse):** Hardwired sensorimotor associations appear first. They are one-shot heteroassociations.

2. **Plastic reflex circuits:** Heterosynaptic and Hebbian mechanisms allow these arcs to adapt with experience.

3. **Two-stage loops:** Evolution introduces crude feedback pathways (ascending + descending), enabling multi-step refinement.

4. **Self-assembly of learning circuits:** As shown by the SAL model, random upstream/downstream pathways with plastic cross-links can self-organize into effective SGD-like optimization circuits without explicit genetic specification of backpropagation.

5. **Cortical scaffolding:** The hippocampus provides episodic heteroassociation and indexing, allowing the cortex to gradually internalize stable associations.

2.6 Modern Parallels: Transformers as Associative Memories

The attention mechanism at the core of transformers [193],

$$\mathrm{Attn}(q, K, V) = \mathrm{softmax}\left(\frac{qK^\top}{\sigma^2}\right) V,$$

is a normalized kernel-based associative memory. Under mild assumptions, it is equivalent to a Gaussian hyperBF:

$$\sum_i \frac{\exp(-\|q - k_i\|^2/2\sigma^2)}{\sum_j \exp(-\|q - k_j\|^2/2\sigma^2)}\, v_i.$$

Thus, a Transformer layer is a composition of two associative memories:

$$\text{Transformer Layer} = \underbrace{\text{HyperBF}_{\text{sample}\to\text{sample}}}_{\text{Self-Attention}} \circ \underbrace{\text{HyperBF}_{\text{sample}\to\text{center}}}_{\text{Feedforward Layer}}.$$

2.7 What Associative Memory Does—and Does Not—Explain

The historical continuity traced in this chapter establishes associative memory as a central computational primitive of intelligence. However, associative memory alone is not sufficient to explain the full phenomenology of intelligent behavior.

Associative memories excel at binding, retrieval, and generalization from partial cues. They provide robustness, noise tolerance, and graceful degradation—properties shared by both biological brains and modern transformers. Yet, in isolation, they lack three essential capabilities.

First, associative memories are fundamentally *atemporal*. They retrieve patterns, but they do not represent persistent latent state evolving over time. Without an explicit dynamical system layered on top, there is no notion of trajectory, causation, or planning.

Second, associative memories do not by themselves enforce *modular reuse*. While they can store many associations, nothing prevents those associations from becoming globally entangled, inhibiting efficient composition across tasks.

Third, associative memories do not provide a mechanism for *long-term accumulation of structure*. Without an external memory or consolidation process, new associations overwrite or interfere with old ones.

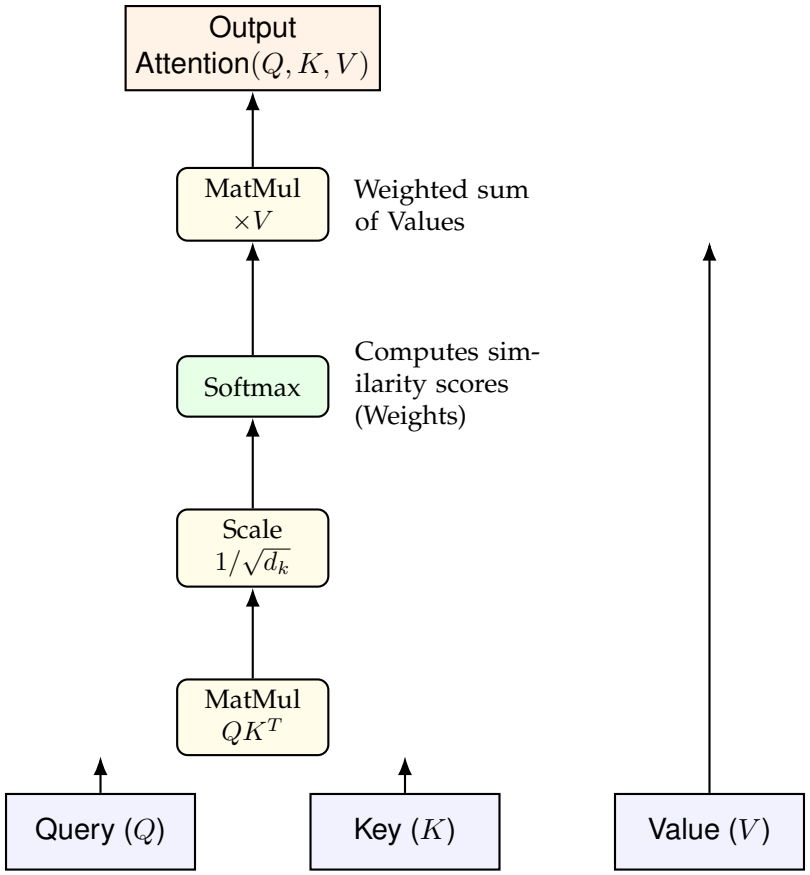


Figure 2.3: The Transformer Attention Mechanism. The Query (Q) and Key (K) interact to compute similarity weights. These weights are then used to retrieve a specific mixture of the stored Values (V).

Transformers inherit all three limitations. They are extraordinarily powerful associative memories, but they lack persistent state, explicit world dynamics, and long-term architectural reuse. These missing elements are addressed in later chapters through Associative Turing Machines, Large Embedding Models, and sparse compositional architectures.

2.7.1 Experimental Evidence from Homogeneous HyperBF Transformers

This equivalence is not merely theoretical. Recent work constructs a "homogeneous" architecture in which all nonlinear components of a standard Vision Transformer (ViT)—both the self-attention blocks and the MLP blocks—are replaced by explicit HyperBF modules.

Experiments on standard image classification benchmarks show that the HyperBF-only models closely match the performance of a baseline ViT with comparable depth, width, and number of heads. For example, on CIFAR and Tiny ImageNet:

Dataset	ViT Acc. (%)	HyperBF Acc. (%)	Gap (%)
CIFAR10	75.61	75.12	0.49
CIFAR100	49.90	48.30	1.60
Tiny ImageNet	32.03	31.14	0.89

This provides concrete empirical evidence that the transformer block is, in practice as well as theory, a composition of associative memory modules.

2.8 Conclusion

Viewed historically, evolutionarily, and computationally, associative memory is not a recurring solution—found across decades and disciplines—to the problem of building systems that can learn, generalize, and scale. From reflex arcs to hippocampal scaffolding to transformer attention, associative memory appears not at the periphery of intelligence, but at its origin.

CHAPTER 3

Associative Memories are Turing-Complete

In Chapter 2, we viewed associative memory as a biological primitive. Here, we demonstrate its computational universality. We formalize an ***Associative Turing Machine*** *(ATM)—a computation model built entirely from content-addressable memory interactions and local updates. We provide precise semantics, a cost model, and a* ***Turing-equivalence*** *theorem. This formalism bridges the gap between classical computability and modern Deep Learning, suggesting that Transformers are not just statistical correlators, but programmable computers trained via gradient descent.*

FIGURE 3: ASSOCIATIVE MEMORIES AS TURING MACHINES

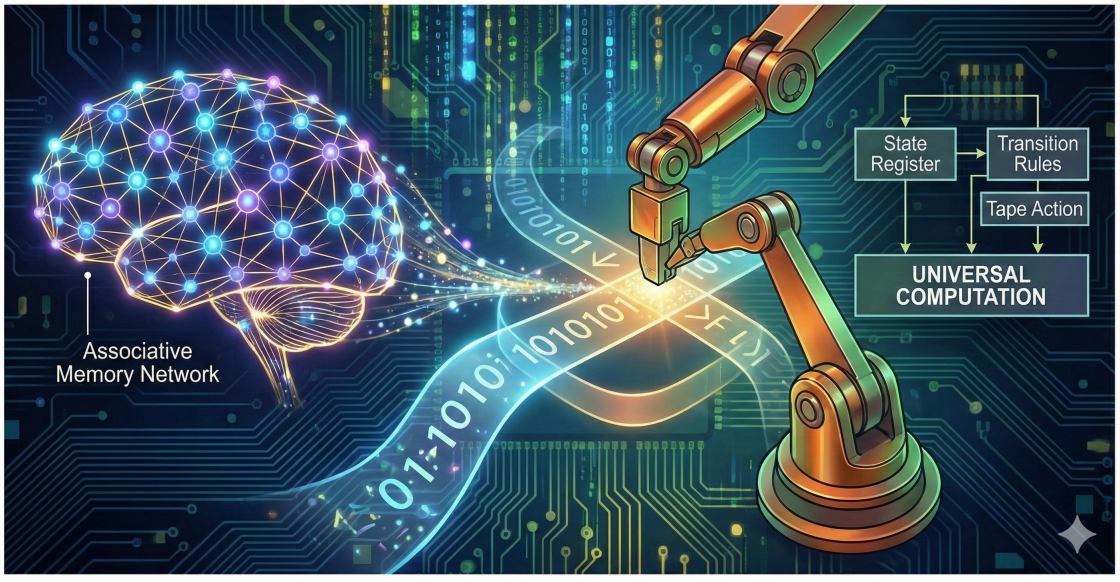


Figure 3.1

3.1 Introduction

A common criticism of Deep Learning, particularly of large language models, is that they are merely "stochastic parrots"—statistical mimics lacking genuine reasoning capabilities. However, from a formal perspective, the architecture underlying these models—the Transformer [193]—bears a striking resemblance to a programmable computer.

To make this rigorous, we must define a computational model based not on logic gates or pointers, but on **continuous associative retrieval** modules.

An Associative Turing Machine (ATM) factors each computational step into two primitives:

1. **Content-Addressable Read (The Memory):** Given a *query* vector q, the system reads a convex combination of *values* whose *keys* are most similar to q in a learned metric. This replaces the "pointer-based" lookup of RAM and relates directly to classical kernel methods (see [24, 152]).

2. **Local Update (The Processor):** A pointwise transformation is applied to the current state and the retrieved value to generate the next state. This corresponds to the transition function of a classical Turing machine.

Iterating "read $\to$ update" realizes algorithmic programs. In this chapter, we prove that this mechanism is sufficient to simulate any deterministic Turing machine with polynomial overhead, providing a theoretical foundation for the algorithmic capabilities of attention-based architectures (see [162]).

3.2 Model: States, Memories, Reads, and Updates

3.2.1 State Space and Encodings

Let Σ be a finite tape alphabet with blank symbol $\sqcup$, Q a finite set of control states, and let classical Turing machine configurations be triples $(q, h, T) \in Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}$.

To map this discrete structure into the continuous vector space of neural networks, we fix an encoding:

$$\mathrm{Enc} : Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}} \to V \subseteq \mathbb{R}^{d_x}, \qquad \mathrm{Dec} : V \supseteq \mathcal{D} \to Q \times \mathbb{Z} \times \Sigma^{\mathbb{Z}}, \tag{3.1}$$

where V is a bounded subset (e.g., a product of simplices) and Dec is a partial inverse defined on a decoding domain $\mathcal{D} \subseteq V$. The *machine state* at step t is $x_t \in V$.

3.2.2 Associative Memory Interface ($\mathcal{M}_{\text{prog}}$)

The core of the ATM is the memory $\mathcal{M}_{\text{prog}}$, which stores the "program" (transition table). It is a finite multiset of key–value pairs

$$\mathcal{M}_{\text{prog}} = \{(k_i, v_i)\}_{i=1}^{N} \subset \mathbb{R}^{d_k} \times \mathbb{R}^{d_v}.$$

Given a query $q \in \mathbb{R}^{d_k}$ and temperature $\tau > 0$, define the softmax read operation:

$$\mathrm{read}_\tau(q; \mathcal{M}_{\text{prog}}) = \sum_{i=1}^{N} \alpha_i(q)\, v_i, \qquad \alpha_i(q) = \frac{\exp\Big(\langle \hat{q}, \hat{k}_i \rangle / \tau\Big)}{\sum_{j=1}^{N} \exp\Big(\langle \hat{q}, \hat{k}_j \rangle / \tau\Big)}. \tag{3.2}$$

As $\tau \to 0$, this mechanism converges to a hard nearest-neighbor lookup, connecting the model to classical associative memories and Sparse Distributed Memory [90]. However, for finite τ, the read operation is fully differentiable.

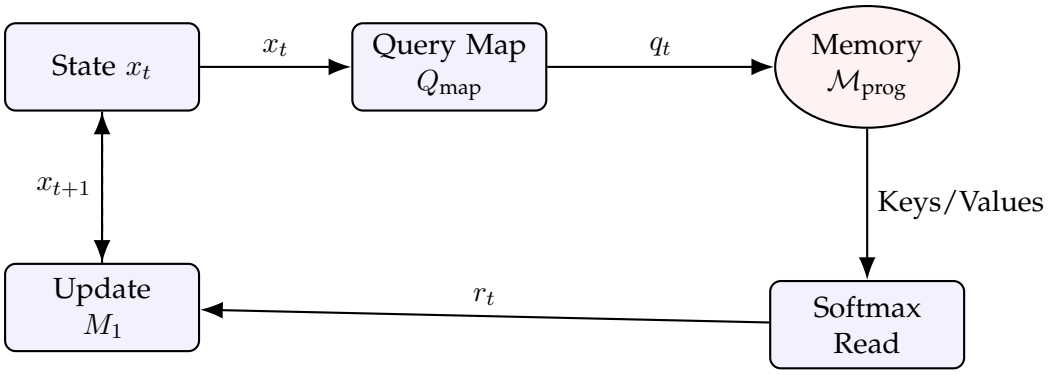


Figure 3.2: Schematic of an Associative Turing Machine (ATM). The cycle consists of generating a query from the current state, retrieving instructions or data from the associative memory, and updating the state via a transition function.

3.2.3 Transition Update (M_1) and Query Map

The machine operates in discrete time. Let $Q_{\text{map}} : V \to \mathbb{R}^{d_k}$ be a function that forms a query from the current state. The local update rule is:

$$x_{t+1} = M_1(x_t, r_t), \qquad r_t = \text{read}_\tau(Q_{\text{map}}(x_t); \mathcal{M}_{\text{prog}}). \tag{3.3}$$

We assume M_1 (the "CPU") is a Lipschitz-continuous function, typically realized by a Multi-Layer Perceptron (MLP) as in standard Transformer blocks.

3.3 From Turing Machines to ATMs

How do we simulate a discrete symbol-manipulating machine with continuous vectors? Let $T = (Q, \Sigma, \delta, q_0, q_{\text{halt}})$ be a deterministic single-tape Turing machine.

3.3.1 Embedding the Tape

The infinite tape of a Turing machine cannot be stored in a fixed-size vector x_t using a naive encoding. Instead, we employ a **fractional encoding** (analogous to a two-stack simulation).

We view the tape as two stacks: one to the left of the head and one to the right. Each stack can be encoded as a rational number $S \in [0, 1)$ using a base-N representation, where $N = |\Sigma|$. Pushing and popping symbols corresponds to affine transformations (multiplying by N or $1/N$ and adding offsets). In the vector space $\mathbb{R}^{d_x}$, these operations are realized by linear maps within the update function M_1. This follows the tradition of simulating automata with dynamical systems.

3.3.2 Embedding the Transition Table

The logic of the Turing machine is stored in the associative memory. We define a binding map $B : Q \times \Sigma \to \mathbb{R}^{d_k}$ that converts a (state, symbol) pair into a query key. We populate the memory with:

$$\mathcal{M}_{\text{prog}} = \{(k_{q,s}, v_{q,s}) : k_{q,s} = B(q, s),\ v_{q,s} = \text{pack}(\delta(q, s))\}.$$

Here, $v_{q,s}$ encodes the action: the new state, the symbol to write, and the direction to move.

3.4 Main Theorem and Proof Sketch

Theorem 1 (ATM $\equiv$ TM, Polynomial Overhead)**.** *For every deterministic Turing machine T and input x, there exists a uniform ATM that halts iff T halts. If T halts in t steps, the ATM halts in $O(t)$ operations with polynomial precision overhead.*

3.4.1 Proof Intuition

The proof relies on two key properties:

1. **Separability of Keys:** We can choose the embedding dimension d_k high enough such that all valid query keys $B(q, s)$ are well-separated on the sphere.
2. **Concentration of Attention:** For a sufficiently small temperature τ, the softmax operation $\alpha_i(q)$ places $1 - \epsilon$ of its mass on the single key closest to the query.

The update function M_1 is designed to be robust to this ϵ-noise. Because the tape encoding is contractive (pushing symbols moves the stack value by smaller and smaller increments), errors do not accumulate catastrophically if the machine periodically "cleans up" its representation—a process akin to digital logic restoration in analog circuits.

3.5 Structural Correspondence to the Transformer

To ground the ATM formalism, we explicitly map its mathematical primitives to the standard layers of the Transformer architecture [193]. This mapping demonstrates that a Transformer block is a physical realization of an associative computational cycle.

3.5.1 The Memory ($\mathcal{M}_{\text{prog}}$) and Query Map ($Q_{\text{map}}$)

In the ATM, the program logic is stored in a multiset of key-value pairs. In a Transformer, this corresponds to the **Key (K)** and **Value (V)** matrices.

- The **Query Projection (W_Q)** acts as the Q_{map}, transforming the current residual state into a search query.
- The **Keys (K)** represent the "addresses" or conditions in the transition table.
- The **Values (V)** represent the "instructions" or candidate next-states.

3.5.2 The Read Operation (read_τ)

The **Self-Attention** mechanism, specifically the scaled dot-product followed by a softmax, is the architectural implementation of the ATM's read_τ operation. It performs a differentiable, similarity-based retrieval of information from the contextual memory. Notice that this frames learning (gradient descent) as the process of "writing" a program into the K and V matrices.

3.5.3 The Update Function (M_1)

The **Feed-Forward Network (FFN)** plays the role of the transition function M_1. While attention routes information, the FFN processes it, taking the current state and the retrieved value to compute the next machine state. In this view, the FFN acts as the "CPU" of the machine, while the attention layers act as the "Addressable Memory".

ATM Primitive	Transformer Layer	Computational Role
Query Map Q_{map}	Query Projection W_Q	State-to-Address Translation
Memory $\mathcal{M}_{\text{prog}}$	Keys (K) & Values (V)	Program Logic Storage
Read read_τ	Self-Attention Layer	Associative Data Retrieval
Update M_1	Feed-Forward Net (FFN)	Transition Execution

Table 3.1: Structural mapping between the Associative Turing Machine and Transformer modules.

3.6 The Transformer as an Associative Turing Machine

The theoretical ATM model aligns precisely with the standard Transformer block defined by [193]. We can now explicitly map the mathematical components of the ATM to the specific layers of the Transformer architecture.

3.6.1 The Update Function (M_1) corresponds to the Feed-Forward Network

In the ATM, the state update is given by $x_{t+1} = M_1(x_t, r_t)$. In a Transformer block, this role is played by the **Feed-Forward Network (FFN)**.

The FFN acts as the processing unit (or "CPU") of the machine. It takes the information retrieved by the attention mechanism (the context) and the current residual state, applying non-linear transformations to compute the next representation.

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{3.4}$$

Functionally, while the attention mechanism routes information (finding *where* data is), the FFN processes it (calculating *what* it means).

3.6.2 The Memory ($\mathcal{M}_{\text{prog}}$) corresponds to Keys and Values

The associative memory $\mathcal{M}_{\text{prog}}$ consists of key-value pairs used to store the transition logic. In the Transformer, this is realized by the **Key (K)** and **Value (V)** matrices in the Self-Attention layer.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.5}$$

Here, the Keys (K) represent the "addresses" or conditions in the transition table, while the Values (V) represent the "instructions" or next-state candidates. The network learns these matrices via gradient descent, effectively writing its own program into the weights W_K and W_V.

3.6.3 Summary of Correspondence

The following table summarizes the structural equivalence:

In chapter 30 we will see why standard RNNs are not associative Turing machines, though they are associative finite state machines.

ATM Component	**Transformer Layer**	**Computational Role**
Query Map (Q_{map})	Query Projection (W_Q)	Creates the search query from state.
Memory ($\mathcal{M}_{\text{prog}}$)	Keys (K) & Values (V)	Stores the program rules/logic.
Read (read$_\tau$)	Attention Mechanism	Retrieves relevant instructions.
Update (M_1)	Feed-Forward Network	Processes info and updates state.

Table 3.2: Mapping between Associative Turing Machine primitives and Transformer modules.

CHAPTER 4

Efficient Computability and Compositional Sparsity

This chapter builds a precise bridge between efficient computability *(polynomial-time Turing computation under explicit precision and uniformity models) and* compositional sparsity *(bounded-fan-in computation DAGs). Compositional sparsity of a target function is well known to imply that there is no curse of dimensionality when the approximation is via a multi-layer deep network [129, 155]. The far reaching consequences include: (i) small bounded-fan-in circuits, (ii) exact deep-net realizations on discrete domains (iii) justification of deep nets as universal parametric approximants with non-exponential number of parameters. We also formulate a* conditional *equivalence among several "associative-memory" mechanisms (attention, Gaussian RBFs, modern Hopfield nets, Kanerva's SDM) as instances of a normalized-similarity read* under stated assumptions*, and we separate* exponential addressability *(codebook cardinality) from* linear superpositional storage *limits via random spherical codes and a CRT/grid-cell example.*

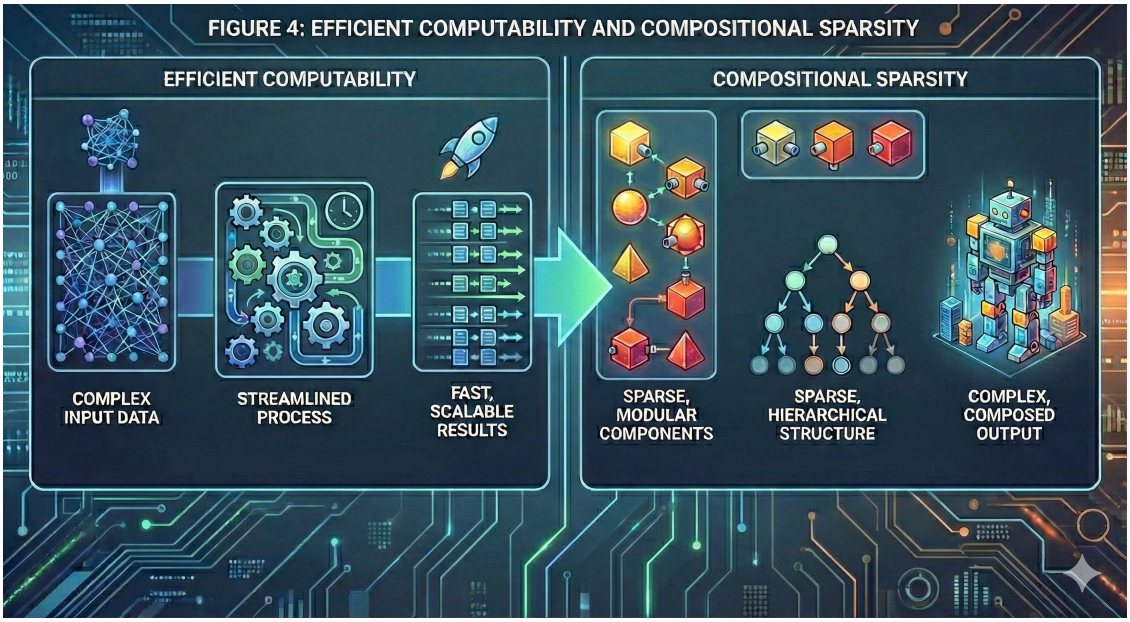


Figure 4.1

4.1 Perspective and Scope

This chapter develops a precise bridge between *efficient computability* and *compositional sparsity*. The central structural claim is that functions which can be computed or learned in practice by digital systems are not generic high-dimensional mappings, but belong to a highly constrained class admitting bounded–fan-in compositional representations.

Informally, the message is:

$$\text{Efficient computability on discrete grids} \implies \text{bounded–fan-in computation DAGs},$$

and, for continuous targets,

$$\text{Efficient computability+computable modulus/conditioning+compositional sparsity} \implies \text{dimension-robust dee}$$

This perspective provides a unified explanation for several empirical and theoretical observations that otherwise appear disconnected: why deep networks can avoid the curse of dimensionality, why optimization remains tractable despite nonconvexity, and why generalization often exceeds what classical capacity-based measures would predict.

Efficient computability as a structural restriction. Throughout this chapter, *efficient computability* is understood in the standard Church–Turing sense: a function is efficiently computable if there exists a deterministic Turing machine that evaluates it in polynomial time, given explicit input precision and output tolerance [189, 177]. This notion captures what can be computed in practice by ordinary digital computers.

A key observation—made precise in the sections that follow—is that efficient computability is not merely a constraint on runtime. It is a strong structural restriction on the class of admissible functions. Any efficiently computable function can be unfolded into a computation DAG of polynomial size and *bounded local fan-in*. This bounded fan-in property is exactly what we call *compositional sparsity*.

Thus, compositional sparsity is not an architectural preference or inductive bias introduced by deep learning practitioners. It is an unavoidable consequence of efficient computation itself.

From Turing machines to deep networks. The formal results in this chapter show that efficiently computable Boolean functions admit P-uniform bounded–fan-in circuits [36, 8], and that their real-valued counterparts—when equipped with explicit precision models and mild regularity assumptions—admit deep ReLU realizations with polynomial size and depth. On discrete grids, these realizations are exact; on continuous domains, dimension-robust approximation rates require bounded local arity [206].

In this sense, deep networks with sparse connectivity are not merely universal approximators. They are the *natural computational normal form* of efficiently computable functions. They are also the natural parametric form to be used to represent a function to be learned.

Consequences for learning. Compositional sparsity explains three central properties of modern learning systems:

1. **Approximation.** Sparse compositional structure eliminates the exponential dependence on ambient dimension that characterizes generic function classes, thereby avoiding the curse of dimensionality.

2. **Optimization.** Bounded local dependencies constrain gradient flow and prevent the combinatorial explosion of interactions that would otherwise make optimization intractable. While training may require overparameterization, the learned solution typically concentrates on a sparse compositional core.

3. **Generalization.** The effective complexity of a compositionally sparse hypothesis class scales with the arity and complexity of its constituents, not with the total input dimension, yielding tighter generalization bounds when sparsity is taken into account [182].

These effects follow from structure alone and do not depend on specific choices of activation functions or architectures.

Associative memory as a unifying mechanism. At the level of computational primitives, many apparently distinct mechanisms— attention, Gaussian RBFs [152], modern Hopfield networks [162], and sparse distributed memory [90]— can be viewed as instances of a *normalized similarity read*, under explicit assumptions. Each such read constitutes an admissible bounded-arity node in a compositional DAG. Stacking these primitives yields deep architectures that are simultaneously expressive, optimizable, and compatible with efficient computability.

A key clarification made later in this chapter is the distinction between *exponential addressability* (the size of a codebook of keys) and *linear superpositional storage* (the number of items reliably retrievable from a single superposition). Confusing these notions has led to persistent misinterpretations of associative memory capacity [91].

Implications for artificial vs. biological intelligence. Because all artificial intelligence systems are implemented on digital computers, every function they realize is efficiently computable by construction. The theory developed here therefore applies fully to machine learning systems: any function learned or executed by an AI model must admit a compositionally sparse representation, even if that structure is not explicit in the trained architecture.

The situation is more subtle for biological intelligence. The physical Church–Turing thesis asserts that biological processes are, in principle, Turing computable. It does *not* assert that they are efficiently computable. Some biological processes—particularly those involving tightly coupled continuous dynamics, affective states, or homeostatic regulation—may be computable only with exponential resources when simulated digitally. Such processes may therefore fall outside the class of functions that can be learned or approximated efficiently by deep networks.

This distinction suggests a non-obvious but instructive asymmetry: higher-level cognitive functions such as language and abstraction, which exhibit strong hierarchical and compositional structure, may be more accessible to efficient computational modeling than some evolutionarily older processes.

A falsifiable prediction. If efficient computability implies compositional sparsity, then the constituent functions implemented by modern architectures must have bounded local arity. In transformers, this implies that the feed-forward (MLP) blocks implement low-arity constituents. Concretely, after suitable permutations of coordinates, the linear maps in each MLP layer should admit block-sparse or column-sparse representations, with sparsity controlled by constituent arity rather than model width.

This yields a concrete, testable prediction: at fixed accuracy, transformer MLP weights should be prunable to sparsity levels determined by local arity (up to logarithmic factors), with corresponding

improvements in norm-based generalization bounds. Emerging empirical evidence of extreme but structured prunability is consistent with this prediction.

The remainder of this chapter provides formal definitions, precise theorems, and explicit constructions that substantiate the claims made in this perspective.

4.2 Definitions and model conventions

Base node classes. Unless otherwise stated, we use one of the following bases for internal nodes with arity $\leq k$:

- **Boolean base:** gates over $\{\pm 1\}$ with fan-in $\leq k$ from a fixed finite basis (e.g., $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$) and free fan-out.
- **ReLU base:** affine+ReLU nodes with at most k inputs, real weights with specified precision, and free fan-out.

We indicate explicitly when a different base is used.

4.2.1 Efficient computability (Boolean and real-valued)

Definition 1 (Efficient computability; Boolean)**.** *Let $\{f_n : \{0,1\}^n \to \{0,1\}\}_{n\geq 1}$. The family is* efficiently computable *if there exists a deterministic Turing machine M and a polynomial p such that $M(x)$ halts within $p(n)$ steps and outputs $f_n(x)$ for all x. Equivalently, there exists a* P*-uniform Boolean circuit family $\{C_n\}$ of size $\mathrm{poly}(n)$ computing f_n.*

Definition 2 (Efficient ε-computability; real-valued (bit/TTE model))**.** *Let $f : [0,1]^d \to \mathbb{R}^m$. An input x is presented via an oracle returning b-bit dyadics, and the target tolerance is $\varepsilon = 2^{-m'}$. We say f is* efficiently computable *if there is a deterministic TM M running in time $\mathrm{poly}(d, b, m')$ on (x, ε) that outputs y with $\|y - f(x)\| \leq \varepsilon$ and with output bit-length $\mathrm{poly}(d, b, m')$. When needed, we assume a* computable modulus of continuity *and a* polynomially bounded condition number *on $[0,1]^d$.*

4.2.2 Computation DAGs and compositional sparsity

Definition 3 (Computation DAG)**.** *A computation DAG $G = (V, E)$ has input nodes* In*, internal nodes* Int*, and output nodes* Out*. Each $v \in$ Int has parent set $\mathrm{pa}(v)$ of size at most k (the* fan-in*) and computes a map ϕ_v on the Cartesian product of its parents' state spaces. The* size *is $s(G) = |\mathsf{Int}|$ and the* depth *$L(G)$ is the longest input-to-output path length.*

Definition 4 (Exact and approximate compositional sparsity)**.** *Fix a base node class $\mathcal{F}_{\leq k}$. A function $f : [0,1]^d \to \mathbb{R}^m$ is* exactly (k, s, L)*-compositionally sparse if $f = \Phi_G$ for some DAG G with fan-in $\leq k$, size $\leq s$, depth $\leq L$, and $\phi_v \in \mathcal{F}_{\leq k}$ at each internal node. Given a function norm $\|\cdot\|_{\mathcal{X}}$, f is* (k, s, L, ε)*-compositionally sparse if there exists such a G with $\|f - \Phi_G\|_{\mathcal{X}} \leq \varepsilon$.*

Uniformity and precision. Circuit/DAG families are assumed P-uniform. For real-valued bases, weights/activations are quantized to $\mathrm{poly}(\log(1/\varepsilon))$ bits as specified in the constructions.

4.3 Efficient computability ⇒ compositional sparsity

4.3.1 Boolean case

Theorem 2 (TM ⇒ circuit; bounded fan-in with uniformity and bounds)**.** *If $f_n : \{0,1\}^n \to \{0,1\}$ is computable in time $T(n) = \mathrm{poly}(n)$, then there exists a* P*-uniform circuit family over $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$ with fan-in ≤ 2, depth $L(n) = O(T(n))$, and size $s(n) = O(T(n) \log T(n))$ that computes f_n. Hence $f_n \in \mathsf{CS}_2(s(n), L(n))$.*

Proof sketch. Space–time tableau unrolled in time; each cell at time $t+1$ depends on $O(1)$ cells at time t. Implement via constant-size gadgets, layer by layer. Clocking/addressing induce the $O(\log T)$ overhead in size. The gate list and wiring are produced in time $\mathrm{poly}(T(n))$. □

4.3.2 Real-valued case: safe discrete-grid result and conditional continuous result

Theorem 3 (Bit-grid deep realization; safe version)**.** *Let $f : [0,1]^d \to \mathbb{R}^m$ be efficiently computable in time $\mathrm{poly}(d, n, \log(1/\varepsilon))$ on inputs from the n-bit grid $\mathcal{G}_n := \{0, 1, \ldots, 2^n\}^d / 2^n$. Then there exists a feedforward network Φ over the ReLU base, organized as a constant-fan-in DAG, of size and depth $\mathrm{poly}(d, n, \log(1/\varepsilon))$ such that*

$$\max_{x \in \mathcal{G}_n} \|\Phi(x) - f(x)\| \;\leq\; \varepsilon.$$

Moreover, the weights can be quantized to $O(\log(1/\varepsilon))$ bits without changing the bound.

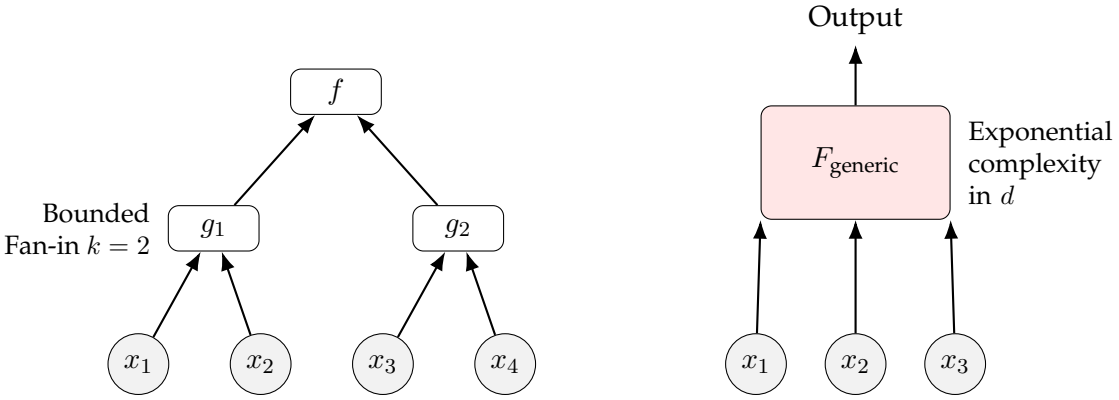


Figure 4.2: The Structural Consequence of Efficient Computability. (a) An efficiently computable function unfolds into a hierarchically structured DAG with bounded local fan-in. This compositional sparsity avoids the curse of dimensionality. Notice that in the Boolean case each constituent function is a juntas, which is sparse enough to be computable without curse of dimensionality. (b) A generic function typically requires a dense, unstructured representation where complexity scales exponentially with input dimension.

Theorem 4 (Continuous approximation; conditional on modulus/conditioning)**.** *Let $f : [0,1]^d \to \mathbb{R}^m$ be efficiently computable with a polynomially computable modulus of continuity and polynomially bounded conditioning on $[0,1]^d$. If, in addition, f is (k, s, L)-compositionally sparse (bounded local arity k) or admits a (k, s, L) approximation at tolerance $\varepsilon/2$, then there exists a ReLU network Φ_ε with constant fan-in and*

$$\|\Phi_\varepsilon - f\|_{\mathcal{X}} \;\leq\; \varepsilon, \qquad \textit{size, depth} \;=\; \mathrm{poly}\big(d, k, L, \log(1/\varepsilon)\big).$$

Dimension-robust $\log(1/\varepsilon)$ *rates require bounded local arity* k *(compositional sparsity); in general, the curse remains [155].*

Stability under depth. If each node is L-Lipschitz and per-node approximation error is δ, then over depth D the end-to-end error is $\leq \delta \frac{L^D-1}{L-1}$ (linear in D if $L \leq 1$). We set $\delta = O(\varepsilon/D)$ to guarantee total error $\leq \varepsilon$.

4.4 Consequences and realizations

4.4.1 Small circuits (tautological)

If $f \in \mathsf{CS}_k(s, L)$ then, by definition, f has a bounded-fan-in circuit of size s and depth L.

4.4.2 Exact deep-ReLU realizations on discrete domains

Lemma 1 (ReLU gadgets for Boolean gates)**.** *For any fixed Boolean gate basis and arity* $m \leq k$*, there is a constant-size ReLU subnetwork* $\mathcal{N}_g$ *(with* $O(1)$ *weights of magnitude* $O(1)$*) that matches* g *on* $\{\pm 1\}^m$*; free fan-out is by wire duplication.*

Theorem 5 (Exact realization on $\{\pm 1\}^n$)*. If* $f \in \mathsf{CS}_k(s, L)$ *over the Boolean base, there exists a ReLU network of size* $O(s)$ *and depth* $O(L)$ *computing* f *exactly on* $\{\pm 1\}^n$*.*

4.4.3 Sparse tabulation (lookup)

Proposition 1 (Lookup/tabulation under explicit bases)**.** *If the node class is Boolean with fan-in* $\leq k$*, then evaluation layer-by-layer can be realized by tabulating each* distinct *Boolean node via a* 2^k*-entry table; the total stored bits are* $O(N_{\textit{distinct}} \cdot 2^k)$ *with* $N_{\textit{distinct}} \leq s$*. For ReLU nodes, an* ε*-quantized lookup scheme stores* $O(N_{\textit{distinct}} \cdot 2^k \cdot \log(1/\varepsilon))$ *bits under a fixed dynamic range; computing* ϕ_v *on the fly trades space for time.*

4.4.4 Algebraic/ polyhedral realization

Proposition 2 (Layered varieties and polyhedra)**.** Boolean base: *The graph of* f *over* $\{\pm 1\}^n$ *is the projection of a variety in* $\mathbb{R}^{n+s}$ *defined by* $O(s)$ *polynomial equations of constant degree, each involving at most* $k+1$ *variables.*
ReLU base: *Over compact domains, the graph of* f *is a union of at most* $\mathrm{poly}(s)$ *polyhedral patches; an algebraic encoding exists with degree that can grow with depth* L*.*

4.5 Unifying mechanisms: a normalized-similarity read (with assumptions)

We study the retrieval

$$r(q) \;=\; \sum_{i=1}^{N} \alpha_i(q)\, v_i, \qquad \alpha_i(q) \;=\; \frac{\exp\big(s(q,k_i)\big)}{\sum_{j=1}^{N} \exp\big(s(q,k_j)\big)}. \tag{4.1}$$

Lemma 2 (Score-equivalence, conditional)**.** *Assume:*

1. ***Attention:*** $s(q,k) = \langle q,k\rangle/\tau$ *(definition).*

2. ***RBFs (shared metric):*** *centers μ_i share a common Mahalanobis metric $\Sigma \succ 0$, i.e., weights $\propto \exp(-\frac{1}{2}\|q-\mu_i\|^2_{\Sigma^{-1}})$; then after the linear change $\phi(q) = \Sigma^{-1/2}q$, $\psi(\mu_i) = \Sigma^{-1/2}\mu_i$ and absorbing per-key additive constants into the softmax normalization, we obtain* (4.1) *[152].*

3. ***Modern Hopfield (single step):*** *with $\alpha_i(x) \propto \exp(\beta\langle x, x_i\rangle)$ and one read step at inverse temperature $\beta = 1/\tau$, the update equals a single-head attention read as in* (4.1) *[162].*

4. ***SDM hard limit:*** *embedding binary addresses on the sphere with a fixed cosine margin and taking $\tau\downarrow 0$ recovers hard nearest-neighbor (Hamming) retrieval as the limit of* (4.1) *[90].*

These equivalences fail without the stated assumptions (e.g., per-key covariances in RBFs or multi-step Hopfield energy descent). NTM/DNC content reads match (4.1)*, but location addressing and explicit erase/write lie outside this form [64, 65].*

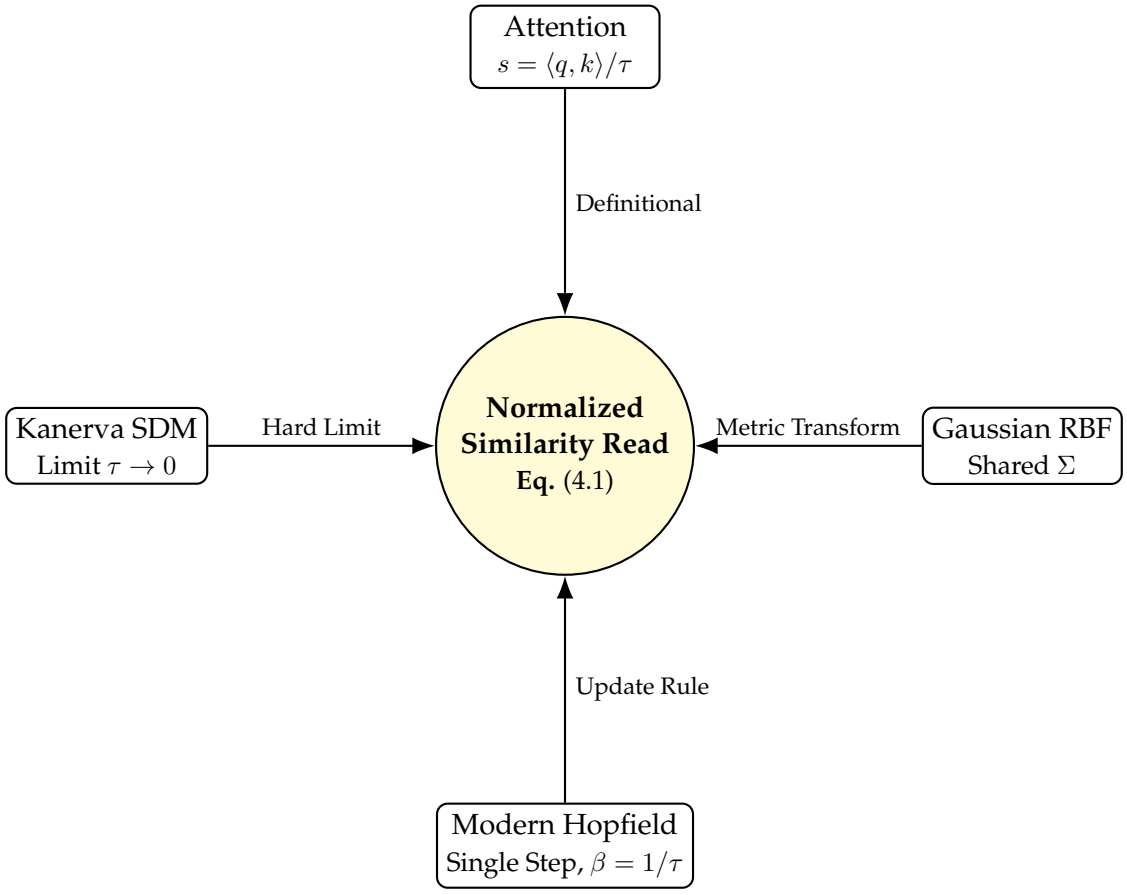


Figure 4.3: The Normalized Similarity Read as a Unifying Mechanism. Under the assumptions detailed in Lemma 2, various associative memory mechanisms converge to a common mathematical form based on softmax-normalized similarity scores.

Mechanism	Score $s(q,k)$	Eq. to (4.1)	Update M_1	Assumptions / Notes
Attention	$\langle q,k\rangle/\tau$	Definitional	Pointwise MLP	Multihead factorization
Gaussian RBF	$-\frac{1}{2}\|q-\mu\|^2_{\Sigma^{-1}}$	Yes (shared Σ)	Pointwise MLP	Per-key covariances break equivale
Hopfield (modern)	$\beta\langle x,x_i\rangle$	Single step	Identity/MLP	Multi-step descent $\not\equiv$ single read
SDM (Kanerva)	$-\mathrm{Ham}(q,k)$ (hard)	$\tau\downarrow 0$	Identity/MLP	Needs margin; shells differ
NTM/DNC	content (+loc.)	Content part	Recurrent ctrl.	Location & erase/write outside (4

Compositional sparsity view. Each AM+update block (one read of the form (4.1) plus a pointwise M_1) is an admissible bounded-arity node. A depth-L stack realizes a (k,s,L) computation DAG with k controlled by head fan-in and local mixing arity. With the hypotheses of Thms. 2, 3, and 4, efficiently computable families admit $(k,s,L)=\mathrm{poly}$ in the appropriate parameters.

4.6 Addressability vs. superpositional storage

4.6.1 Exponential codebooks via random spherical codes

Let d be the key dimension. Standard random packing yields:

Lemma 3 (Near-orthogonality, exponential packing)**.** *For any $\varepsilon\in(0,1)$ there exists $C>0$ such that if $N\le\exp(C\varepsilon^2 d)$ then, with high probability, one can realize $\{k(x)\}_{x=1}^N\subset\mathbb{S}^{d-1}$ with $\max_{x\neq y}|\langle k(x),k(y)\rangle|\le\varepsilon$.*

CRT/grid-cell example (address space). As a constructive alternative, pick coprime moduli $(m_j)_{j=1}^J$ and define module phases $k_j(x)=\exp(2\pi i\langle a_j,x\rangle/m_j)$; the address space is $\prod_j m_j$ by Chinese remaindering [52]. Mapping phases to $\mathbb{S}^{d-1}$ (e.g., real/imag parts) gives large codebooks with controllable collisions; this is orthogonal to the superposition limit below.

4.6.2 Holographic superposition and SNR

Let $\odot$ denote a unitary binding operator (e.g., *Hadamard product* with unit-modulus complex keys, or circular convolution in real space, which is diagonalized by the DFT) [143]. Store M associations as

$$W=\sum_{j=1}^{M} k_j\odot w_j,$$

and retrieve with the unbinding $k_\ell^\dagger\odot W$ (complex conjugate for phases or inverse filter for convolution), followed by a componentwise estimator of w_ℓ.

Proposition 3 (Crosstalk and capacity)**.** *With keys from Lemma 3 (or unit-modulus phases with small mutual coherence), the per-coordinate crosstalk variance is $O(M/d)$ and $\mathrm{SNR}\asymp\sqrt{d/M}$. Thus a single superposition reliably stores and retrieves $M=\Theta(d)$ items at fixed error under standard estimators [194].*

Takeaway. "Exponential addressability" refers to the *number of distinct addresses* in the codebook (Lemma 3 or CRT/grid). The number of items reliably retrievable from a *single* superposition scales only linearly with d due to crosstalk.

4.7 Concluding remarks

The tableau translation of Turing computation into bounded-fan-in circuits (Thm. 2) and the safe/conditional real-valued counterparts (Thms. 3–4) show that:

- Efficient computability on discrete grids yields uniform bounded-fan-in DAGs and exact deep-ReLU realizations on those grids (Thm. 5).
- For continuous targets, *dimension-robust* rates require *compositional sparsity* (bounded local arity) together with computable modulus/conditioning [129].
- Tabulation and algebraic/polyhedral realizations follow once the base is fixed and precision is explicit (Props. 1–2).

At the mechanism level, attention, Gaussian RBFs, modern Hopfield networks, and SDM can align with a normalized-similarity read *under explicit assumptions* (Lemma 2); NTM/DNC extend this with location and write/erase. Finally, random spherical codes and CRT/grid scaffolds clarify how to enjoy exponential *addressability* without contradicting the linear limits of *superpositional storage*. Together, these results provide a unified, compositionally sparse account of when and why deep architectures avoid the curse of dimensionality while remaining compatible with classical notions of algorithmic efficiency.

4.8 Mathematical Supplement: Proofs of the Main Theorems

This appendix provides full proofs for Theorems 8 and 9. Both results rely on the same construction: unrolling a polynomial-time Turing machine into a sequence of discrete configurations and showing that each local transition can be implemented by a linear threshold (or small Boolean) network acting on a suitably encoded representation.

4.8.1 Encoding of Turing Configurations

Let M be a deterministic Turing machine that computes $f : \Sigma^n \to \Sigma^m$ in time $T(n) = \mathrm{poly}(n)$. During its execution on an input $x \in \Sigma^n$, the machine passes through configurations

$$\tau_0, \tau_1, \ldots, \tau_{T(n)},$$

where each τ_t records the entire contents of the work tape, the head position, and the internal control state.

Binary encoding. We represent each configuration τ_t by a binary vector

$$s_t \in \{0,1\}^d, \qquad d = \mathrm{poly}(T(n)).$$

A convenient choice is a one-hot encoding: for every tape cell $j \le S(n) \le \mathrm{poly}(T(n))$ and each alphabet symbol $a \in \Gamma$, we include a bit $b_{j,a}$ that is 1 iff cell j contains symbol a in τ_t. Additional one-hot groups encode the head position and internal state. This encoding is injective: every configuration corresponds to a unique s_t.

Locality of updates. The transition $\tau_t \mapsto \tau_{t+1}$ depends only on: (i) the symbol under the head, (ii) the internal state, and produces updates to that same local region. Hence each bit of s_{t+1} depends on $O(1)$ bits of s_t. This locality is crucial for realizing the transition by simple threshold rules.

4.8.2 Proof of Theorem 8 (Autoregressive Universality)

Existence of a linear threshold realization. For each output bit b' of s_{t+1} there is a small finite set of input bits of s_t that determines it. Because s_t is one-hot in each field, testing which case applies reduces to a linear threshold test: choose a weight vector $\mathbf{w}$ that assigns positive weight to the active indicator for the correct combination of state and tape symbol and negative weight elsewhere, together with a bias b so that

$$h_{\mathbf{w},b}(s_t) = 1 \quad \text{iff the local rule producing } b' \text{ applies.}$$

Collecting all these units yields a multi-output linear threshold layer H satisfying $H(s_t) = s_{t+1}$ for all valid s_t.

Dataset construction. Generate a dataset

$$\mathcal{D} = \{(s_t^{(r)}, s_{t+1}^{(r)}) : r = 1, \dots, M,\ t = 0, \dots, T(n) - 1\},$$

where each sequence $\{s_t^{(r)}\}$ comes from unrolling M on an input $x^{(r)}$ sampled from the data distribution D. The dataset size $|\mathcal{D}| = M\,T(n)$ can be chosen polynomial in $T(n)$.

Learning argument. The hypothesis class of linear threshold functions on $\{0,1\}^d$ has VC dimension $O(d^2)$, hence is PAC-learnable with $m = \mathrm{poly}(d)$ examples. Since $d = \mathrm{poly}(T(n))$, a sample of size $|\mathcal{D}| = \mathrm{poly}(T(n))$ suffices to fit the exact transition map on all reachable configurations with high probability. Denote the learned map by $\hat{H}$.

Iterative reconstruction of $f(x)$. For a new input x,

$$s_0 \xrightarrow{\hat{H}} s_1 \xrightarrow{\hat{H}} \cdots \xrightarrow{\hat{H}} s_{T(n)},$$

and $s_{T(n)}$ encodes $f(x)$. By union-bound arguments, the probability that any intermediate step errs is small, so the overall error rate is bounded by $\mathrm{poly}(T(n))\varepsilon$. Hence $\hat{H}$ reproduces f with high probability.

□

4.8.3 Proof of Theorem 9 (Diffusion-Step Universality)

Construction of the diffusion chain. Define a sequence of latent states

$$s_T,\ s_{T-1},\ \dots,\ s_0$$

such that s_t encodes the configuration τ_t of the same Turing computation. We regard the "forward diffusion" as running the machine (increasing t) and the "denoising" direction as reversing it.

Local denoising map. Each reverse transition $\tau_t \mapsto \tau_{t-1}$ is again local and hence can be implemented by the same threshold construction: there exists a layer G with $G(s_t) = s_{t-1}$ for all valid encodings. Training data consist of pairs (s_t, s_{t-1}) obtained by unrolling M; the sample size is polynomial in $T(n)$.

Learning and composition. A learned approximation $\hat{G}$ to G generalizes by the same VC-dimension argument as before. Iterating $\hat{G}$ backward $T(n)$ steps from s_T reconstructs s_0 encoding the final output $f(x)$. Thus the diffusion-style denoiser achieves the same computational universality as the autoregressive predictor.

□

4.8.4 Remarks and Extensions

- The above proofs are constructive: given any polynomial-time Turing machine, one can explicitly build the dataset and weight vectors realizing the stepwise transitions.
- The encodings are polynomial in $T(n)$ but may be large in practice; this is a theoretical universality statement, not an efficiency claim.
- Replacing linear thresholds with polynomial-size Boolean circuits yields identical results, often with smaller encodings (Appendix B).
- Appendix C extends the argument to *true Gaussian diffusion*, where each s_t is a one-hot vector corrupted by Gaussian noise and the denoiser is a linear threshold network that identifies the active coordinate with high probability.

4.9 Technical Note: Replacing Linear Threshold Functions by Boolean Circuits

The proofs use *linear threshold functions* (LTFs) to represent the local transition map of a Turing machine. Exactly the same reasoning applies if we replace each LTF by a *polynomial-size Boolean circuit*. This substitution makes the construction more natural from the standpoint of classical computational complexity.

4.9.1 Expressing a Turing-Machine Step as a Circuit

Each step of a deterministic Turing machine reads:

1. the symbol a currently under the head,
2. the internal control state q, and
3. the local neighborhood of the head on the tape,

and outputs:

- a new symbol a',
- a new internal state q', and
- a movement direction (left, right, or stay).

This transition function is a finite lookup table

$$\delta : \Gamma \times Q \to \Gamma \times Q \times \{\Leftarrow, \Rightarrow, \cdot\}.$$

Such a mapping can always be encoded by a Boolean circuit of size $O(|\Gamma||Q|)$. When we embed the machine configuration into a bit vector s_t, the update rule $s_t \mapsto s_{t+1}$ therefore admits a polynomial-size circuit implementation composed of AND/OR/NOT gates acting on $O(1)$ bits per output.

4.9.2 Replacement in the Stepwise-Learning Framework

In both the autoregressive and diffusion constructions, the only property required of the hypothesis class $\mathcal{H}$ is that for every valid configuration s_t there exists $h \in \mathcal{H}$ satisfying $h(s_t) = s_{t+1}$ (or s_{t-1}). Because a polynomial-size Boolean circuit can realize this mapping, we may take $\mathcal{H}$ to be the set of all such circuits instead of LTFs.

Training then amounts to learning the truth table of the circuit's local rule from pairs (s_t, s_{t+1}) or (s_t, s_{t-1}). All results in Theorems 8 and 9 remain unchanged: the learned circuit can be composed $T(n)$ times to reproduce the full polynomial-time computation.

4.9.3 Comparison with Linear Threshold Implementations

- **Representation complexity.** LTFs often require high-dimensional one-hot expansions to simulate discrete logic. Circuits encode the same logic natively and can thus achieve the transition rule with fewer variables.
- **Expressive power.** Polynomial-size Boolean circuits are universal for polynomial-time computation. Using them as the stepwise predictor aligns the learning model directly with the class $\mathbf{P}$.
- **Interpretation.** Theorems 8–9 can therefore be rephrased as: *stepwise training of polynomial-size circuits on intermediate states suffices to learn any function in* $\mathbf{P}$.

4.9.4 Summary

Replacing the linear-threshold layer by a polynomial-size Boolean circuit leaves all theoretical results intact while simplifying the connection to computational complexity. The unifying principle remains the same: **step-by-step supervision of intermediate Turing-machine configurations enables a simple local learner—whether linear or Boolean—to reproduce any polynomial-time computation.**

CHAPTER 5

Optimization and Compositionality (with P. Beneventano)

Does compositional structure make optimization easier? This chapter argues that the answer is nuanced. Compositionality is the structural property that allows high-dimensional functions to be represented with polynomially many parameters, thereby avoiding the curse of dimensionality. This is its primary contribution: it renders the problem of learning ***possible*** *(in terms of sample complexity and memory). However, it does not automatically make the optimization landscape* ***easy****. We distinguish between the* representation benefit *(which is massive) and the* optimization difficulty *(which remains high in end-to-end training). We conclude that the "unreasonable effectiveness" of optimization in deep learning arises not just from the architecture, but from the ability to decompose the problem when intermediate signals are available.*

Figure 5.1

5.1 The Core Argument

The success of Deep Learning is often attributed to the ability of stochastic gradient descent (SGD) to optimize deep networks. However, from a theoretical standpoint, compositional structure plays two distinct roles, which must not be confused:

1. **Representational Efficiency (The "Why it works" condition):** Compositionality ensures that the number of parameters required to approximate a function scales polynomially (e.g., linearly) with the input dimension d, rather than exponentially. Without this, no optimization algorithm could succeed because the search space would be too vast to explore with finite data [129].

2. **Optimization Landscape (The "How it works" puzzle):** While compositionality reduces the number of variables, it introduces non-convexity. A deep composition of simple functions $f(g(h(x)))$ creates a landscape with symmetries, saddle points, and local minima. Therefore, compositionality is a *necessary condition* for solvability, but it does not inherently guarantee fast convergence.

5.2 The Representation Benefit: Avoiding the Curse

The primary contribution of compositionality to optimization is that it reduces the dimensionality of the search space to a manageable size.

5.2.1 Parameter Counting

Consider a target function $f : \mathbb{R}^d \to \mathbb{R}$.

- **Generic Case:** If f is a generic smooth function (e.g., in a Sobolev space), approximating it to error ε requires $O(\varepsilon^{-d})$ parameters. Optimization in this space is intractable for large d.

- **Compositional Case:** If f is compositionally sparse (a DAG of constituent functions with bounded arity k), the number of parameters scales as $O(d \cdot k)$ [154].

Implication: The optimization algorithm operates in a space of dimension $O(d)$ rather than $O(e^d)$. This is the fundamental reason optimization is feasible: the "needle in the haystack" is effectively larger because the haystack (the parameter space) is exponentially smaller.

5.3 The Optimization Challenge in End-to-End Learning

If we train a deep network end-to-end (using only input x and output y), we are solving:

$$\min_{W_1,\dots,W_L} \sum_i \ell\left(f_L(\dots f_1(x_i; W_1)\dots; W_L), y_i\right)$$

Even though the number of parameters W is manageable, this problem is highly non-convex.

5.3.1 Vanishing Gradients and Conditioning

Compositionality creates deep chains of dependencies. In standard gradient descent, the gradient at early layers is the product of Jacobians of later layers:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial f_L} \cdot \frac{\partial f_L}{\partial f_{L-1}} \cdots \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial W_1}$$

Unlike the "block coordinate descent" scenarios in convex optimization, these blocks are coupled multiplicatively. This often leads to ill-conditioning, where the landscape is extremely steep in some directions and flat in others.

Conclusion: Compositional structure *alone* does not imply fast convergence rates (like those found in convex optimization). The fact that SGD finds global minima is likely due to overparameterization (Relaxation) rather than the raw compositional shape.

5.4 The Ideal Scenario: Module-wise Optimization

It has been pointed out that the true optimization advantage of compositionality emerges when the decomposition is known and exploitable.

5.4.1 The "Grey Box" vs. "Black Box"

- **Black Box Optimization (End-to-End):** We only see x and y. We must infer the internal representations. This is hard.
- **Grey Box Optimization (Modular):** We know the structure $f(x) = h(g(x))$ and, crucially, we have training signals for the intermediate variable $z = g(x)$.

Theorem 6 (Optimization with Intermediate Supervision). *Let $f = h \circ g$. If datasets $D_g = \{(x, g(x))\}$ and $D_h = \{(g(x), y)\}$ are available, the optimization problem decouples into two independent sub-problems:*

$$\min_{W_g} \sum \ell(g(x; W_g), z) \quad \textit{and} \quad \min_{W_h} \sum \ell(h(z; W_h), y)$$

If the constituent functions g and h are "simple" (e.g., low dimension, convex), these sub-problems can be solved efficiently.

5.4.2 Curriculum Learning and Pre-training

In practice, we rarely have full datasets for every intermediate module. However, modern techniques approximate this "Modular Optimization" ideal:

1. **Layer-wise Pre-training:** Training one layer at a time (as in early Deep Belief Networks) effectively creates local supervised problems [76].
2. **Transfer Learning:** Using a pre-trained "backbone" (e.g., ResNet or BERT) treats the early layers as a solved module $g(x)$, reducing the optimization task to just the final head $h(z)$.
3. **Chain-of-Thought:** In Large Language Models, forcing the model to output intermediate reasoning steps is a form of explicit compositional supervision [20].

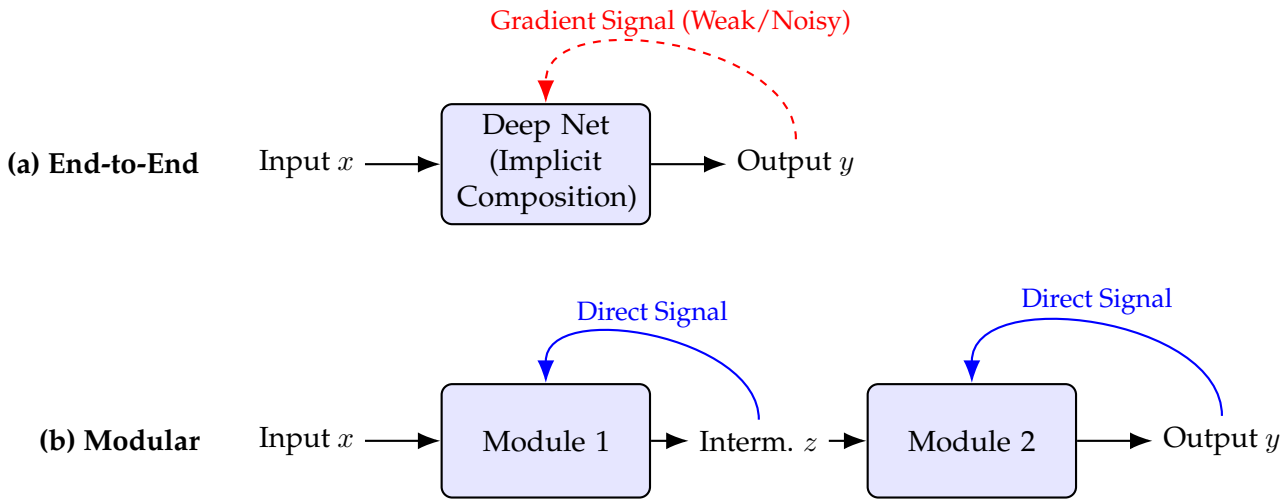


Figure 5.2: Two Regimes of Compositional Optimization. (a) In end-to-end learning, compositionality reduces parameter count but the error signal must propagate through the entire chain, often leading to difficult optimization landscapes. (b) When intermediate supervision is available (Modular), the problem decomposes into simpler, easily solvable sub-problems.

5.5 Sample Complexity: The Gap Between Shallow and Deep

A critical implication of compositionality is its effect on sample complexity—the number of training examples N required to achieve a given generalization error. This scaling depends fundamentally on whether the network architecture matches the hierarchical structure of the target function.

5.5.1 The Curse for Shallow Architectures

Standard "shallow" methods—ranging from classical linear approximators (splines, kernel machines) to single-hidden-layer neural networks—face a fundamental limitation when approximating high-dimensional functions.

Classical results [68] show that for a generic Lipschitz function in d dimensions, the number of examples required to achieve error ε scales exponentially with d:

$$N_{\text{shallow}} \approx O(\varepsilon^{-d}).$$

Crucially, even if the target function has a sparse compositional structure (e.g., a hierarchy of low-dimensional functions), a shallow network generally cannot exploit this without an exponentially large width. It effectively attempts to "flatten" the composition, losing the structural prior and succumbing to the curse of dimensionality.

5.5.2 The Blessing for Deep Compositional Networks

In contrast, deep neural networks can mirror the compositional graph of the target function. If the target function f is a composition of functions with local dimensionality $k \ll d$, and the network depth matches this structure, the number of parameters W—and by extension the sample complexity—scales linearly with d:

$$N_{\text{deep}} \approx O(d \cdot \varepsilon^{-k}).$$

This gap between $O(\varepsilon^{-d})$ and $O(d\varepsilon^{-k})$ quantifies the data efficiency of deep learning [129]. The advantage of depth is not just about having "more" parameters, but about having the *right* structural arrangement to represent the target function compactly. Optimization in deep learning is feasible because we are searching for a solution in this structurally aligned, polynomially-sized space.

5.6 Summary: The Optimization-Representation Trade-off

Compositionality is not a direct accelerator of gradient descent dynamics in the way convexity is. Instead, its role is foundational:

1. **Existence:** It guarantees that a solution exists within a polynomially sized parameter space (Representation).
2. **Feasibility:** It allows the problem to be solved with realistic amounts of data (Sample Complexity).
3. **Speed:** Fast optimization is **not** guaranteed by compositionality alone. It requires either (a) massive overparameterization to smooth the landscape, or (b) access to intermediate signals that allow the problem to be decomposed into its constituent parts.

Thus, we affirm the view that compositionality is primarily about avoiding the curse of dimensionality in representation, and secondarily about enabling modular learning strategies when data permits.

CHAPTER 6

Genericity and Optimization (with P. Beneventano)

Why does local gradient descent succeed in optimizing deep networks, despite the non-convex nature of the loss landscape? This chapter proposes that the answer lies in a property of learnable functions called ***Genericity****.*

We argue that most functions are "Generic," meaning that non-generic functions have measure zero. This also implies that the "jet" structure of generic functions is invariant to basic transformations like shifts in the origin of the coordinates of the input x. a very useful property follows: while mathematical pathologies (like pure parity) hide their structure at points of symmetry, generic functions reveal "linear footprints"—low-degree correlations—under realistic biases and noise. This structural revelation ensures that gradient-based methods see informative directions at initialization, preventing the optimizer from stalling on flat plateaus.

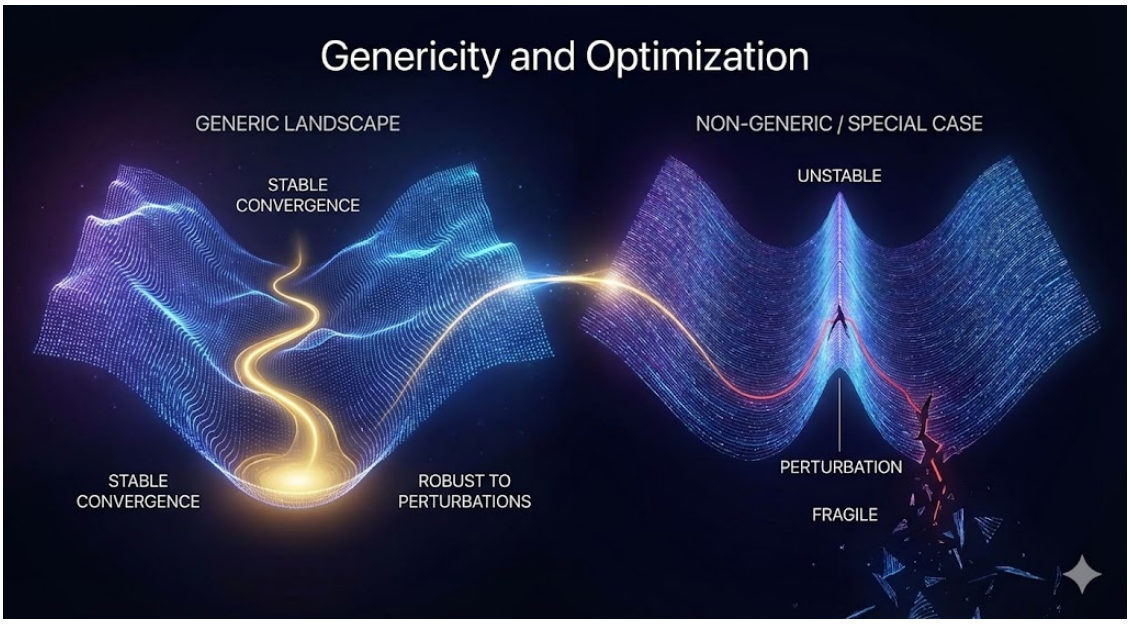


Figure 6.1: Visualizing the transition from a pathological symmetric landscape to a generic learnable landscape.

6.1 The Core Argument

The success of deep learning relies on a harmony between the algorithm and the environment. The fundamental puzzle of deep learning optimization is often stated as: "How can a local greedy method (Gradient Descent) find a solution in a high-dimensional, non-convex landscape?"

We propose that the answer is not found in the optimizer alone, but in the nature of the target functions that are sensible to learn.

1. **The Mathematical View:** In the space of all possible functions, "hard" functions exist (e.g., high-degree parity). These functions have landscapes that are flat almost everywhere, with the solution hidden in a small, distinct region. Optimization is impossible without exhaustive search.

2. **The Physical View (Genericity):** Real-world functions are not adversarial. They are **Generic**, meaning they are not "fragile" to changes in the coordinate system. Specifically, we assume that under small shifts of the input zero-point (biases) or small perturbations (noise), the function exhibits visible low-degree components (linear or quadratic correlations). This aligns with the theoretical framework of smoothed analysis [179].

The logical chain of this chapter is:

$$\text{Genericity} \implies \text{Visible Gradients at Initialization} \implies \text{Tractable Optimization.}$$

6.2 Defining Genericity: Invariance to Shifts

Mathematical intuition often fails in high dimensions because we tend to think of "pure" functions defined in highly symmetric coordinate systems. To formalize our argument, we must define what we mean by a "Generic" function.

6.2.1 General Genericity

We propose a generalized definition based on the invariance of local structure (jets) under low-degree coordinate transformations.

Definition 5 (General Genericity (informal))**.** *A target function f is **Generic of order** k if, under polynomial coordinate transformations of degree at most k, the low-order statistical signatures of f (e.g. non-vanishing low-degree projections) persist for typical small perturbations of the input distribution.*

6.2.2 The Genericity Principle (Shift Invariance)

In this chapter, we specialize this definition to the case of $k = 0$, which corresponds to invariance under coordinate shifts (biases).

Definition 6 (Genericity Principle (Shift Invariance))**.** *A target function is **Generic (for optimization)** if small shifts of the input origin (transformations $x \mapsto x + b$) typically induce* non-vanishing low-degree correlations *between $f(x)$ and low-degree functions of x (e.g. linear or quadratic features), so that gradient-based methods have informative initialization signal.*

Conversely, a **Non-Generic** function lacks these dominant low-degree terms, relying instead on high-complexity (high-frequency) terms to fit data. The prototype of a Non-Generic function is the Parity function (or XOR) centered perfectly at zero:

$$f(x) = \prod_{i=1}^{d} x_i, \quad x \sim \mathcal{N}(0, I).$$

This function is orthogonal to all linear terms. A gradient-based learner initialized randomly sees zero correlation; the loss landscape is a perfectly flat plateau. However, this hardness is an artifact of a specific, fragile symmetry.

6.3 Why non-genericity is fragile: Thom transversality

The previous sections motivate *genericity* as the appearance of low-degree statistical signal ("linear footprints") after small perturbations such as coordinate shifts. Here we give a geometric justification for why exact cancellations are *non-robust*: they correspond to the intersection of a smooth map with a *thin* (positive-codimension) "bad" set, and Thom's transversality theorem implies that such intersections disappear under arbitrarily small perturbations.

6.3.1 Jets and "bad" sets of functions

Let $f : \mathbb{R}^d \to \mathbb{R}$ be smooth. The k-jet $j^k f(x)$ encodes the value of f and all partial derivatives up to order k at x. The collection of all k-jets forms the jet bundle $J^k(\mathbb{R}^d, \mathbb{R})$.

Many "non-generic" properties can be written as algebraic constraints on jets. For example, a strong form of "no linear footprint at x" corresponds to the first-jet constraint $\nabla f(x) = 0$. More generally, vanishing of all derivatives up to order k is the constraint

$$f(x) = 0, \;\; \nabla f(x) = 0, \;\; \ldots, \;\; D^k f(x) = 0,$$

which defines a subset $W \subset J^k(\mathbb{R}^d, \mathbb{R})$ of positive codimension.

6.3.2 Thom's transversality theorem (informal)

Theorem 7 (Thom transversality (informal))**.** *Let $W \subset J^k(\mathbb{R}^d, \mathbb{R})$ be a smooth submanifold (or, more generally, a stratified submanifold). Then the set of smooth functions f whose jet extension $j^k f : \mathbb{R}^d \to J^k(\mathbb{R}^d, \mathbb{R})$ is transverse to W is open and dense in the Whitney C^∞ topology.*

Transversality means that $j^k f$ meets W "at a nonzero angle." When W has positive codimension, transversality implies that the preimage $(j^k f)^{-1}(W)$ is typically empty or has strictly smaller dimension than the ambient space. In particular, the constraints defining W cannot hold "everywhere" except for functions living in a highly exceptional set.

6.3.3 Connection to shift-based genericity

To connect this to our setting, consider the shifted family

$$f_b(x) := f(x + b), \qquad b \in \mathbb{R}^d.$$

Non-genericity after a shift can be phrased as "for this b, the low-degree signal vanishes," e.g. a linear footprint condition of the form

$$\mathbb{E}\big[f_b(X)\, \phi(X)\big] = 0$$

for a collection of low-degree test functions ϕ. For polynomial targets and Gaussian or product measures, these expectations are (real-)analytic functions of the shift parameter b (indeed, polynomials in many cases). Exact cancellation therefore requires b to lie in the zero set of a nontrivial analytic function, which is a measure-zero set.

Geometrically, the shift parameter b acts as a low-dimensional perturbation of the jet extension. Thom transversality tells us that, for a generic perturbation (in particular, for almost every small shift), the jet avoids the "bad" set W responsible for vanishing low-order structure. This is the formal content of the slogan:

exact symmetries are unstable under perturbation.

Remark 1 (What this does and does not claim)**.** *Thom transversality is a* structural *result: it explains why pathological cancellations are nongeneric (topologically rare / measure-zero under smooth randomization). Quantitative statements (how large the footprint is) require additional anti-concentration tools (e.g. Carbery–Wright) and/or model-specific bounds.*

6.4 The Mathematics of Structure Leaking

What happens when we apply the Genericity Principle (coordinate shifts) to these "hard" functions? We prove that the "hard" couplings do not vanish; they are simply accompanied by lower-order shadows that guide the optimizer.

6.4.1 Real-Valued Targets: Linear Footprints

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a smooth target. We model "generic coordinates" by assuming the input x contains a small Gaussian jitter (or that the observer's coordinate system is slightly noisy). Let $x \sim \mathcal{N}(z, \sigma^2 I)$.

Proposition 4 (Shifted Stein identity and linear footprints)**.** *Let $X \sim \mathcal{N}(\mu, \sigma^2 I_d)$ and let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable with mild integrability conditions. Then for each i,*

$$\mathbb{E}[f(X)\,(X_i - \mu_i)] \;=\; \sigma^2\,\mathbb{E}\Big[\frac{\partial f}{\partial x_i}(X)\Big].$$

Equivalently,

$$\mathbb{E}[f(X)X_i] \;=\; \mu_i\,\mathbb{E}[f(X)] \;+\; \sigma^2\,\mathbb{E}\Big[\frac{\partial f}{\partial x_i}(X)\Big].$$

Implication: Except under special symmetries (which occur on a measure-zero set of shifts in many polynomial settings), active variables typically induce nonzero linear correlations, i.e. "linear footprints," that can be detected by simple screening.

6.4.2 Boolean Targets: The Bias Leakage

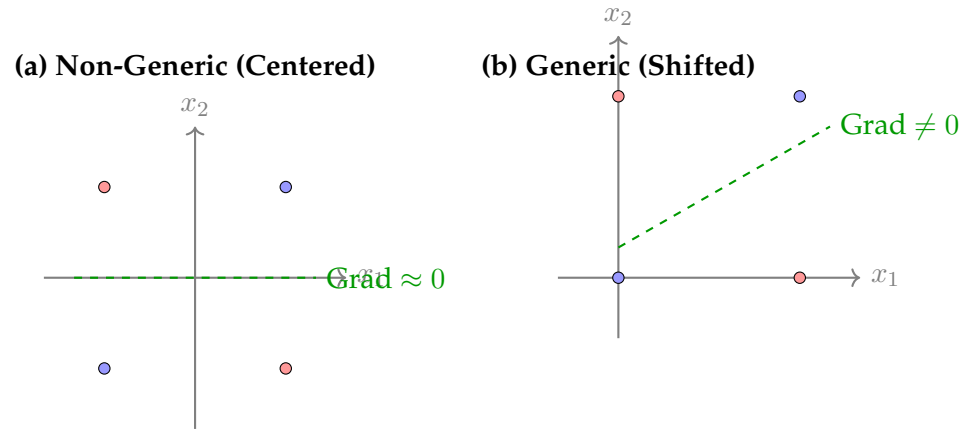
Consider Boolean inputs $x \in \{-1, 1\}^d$. Let $f(x) = \prod_{i \in S} x_i$ be a parity function on a subset S. If we shift the coordinate system by a vector μ (implying $\mathbb{E}[x_i] = \mu_i \neq 0$), the function expands:

$$f(x' - \mu) \approx \prod (x'_i - \mu_i) \implies \text{Linear Terms appear.}$$

Lemma 4 (Leakage under Bias)**.** *Under a biased product distribution, the correlation between the function and a single relevant variable x_j ($j \in S$) is:*

$$\mathbb{E}[f(x)x_j] = \prod_{i \in S \setminus \{j\}} \mu_i.$$

Implication: As long as the biases μ_i are non-zero (the Generic condition), the correlation is non-zero. The spectral mass "leaks" down to degree 1. The "hardest" Boolean function becomes visible to linear screens [40].



Symmetric. No linear correlation. Optimizer is blind.

Coordinate Shift breaks symmetry. Linear "footprint" appears.

Figure 6.2: The Genericity Principle visualized on the XOR problem. (a) In a perfectly symmetric coordinate system (Non-Generic), the target function has zero correlation with the inputs. (b) Under a coordinate shift or bias (Generic), the symmetry breaks, and a linear correlation emerges.

6.5 Clarifying Generalization vs. Optimization

Genericity, as used in this chapter, is an *optimization-signal* condition: it concerns whether low-degree statistics of the data distribution expose informative directions for gradient-based methods at initialization. This is conceptually distinct from classical *generalization* conditions, such as uniform convergence (uGC) or algorithmic stability, which are properties of a hypothesis class and learning rule.

Generalization is about the algorithm and hypothesis class. Uniform stability [23] and uGC properties [**vapnik1971uniform**] control how empirical risk tracks population risk. They do not, by themselves, guarantee that a particular optimization method can *find* a good hypothesis efficiently.

Optimization is about available statistical signal. Gradient-based methods access the data distribution through low-degree statistics (expected gradients), and can be viewed as instances of Statistical Query (SQ) algorithms [94]. When a target is orthogonal to all low-degree polynomials under the input distribution (the “cliff” regime), these statistics carry essentially no information,

leading to large sample/iteration complexity for any gradient/SQ procedure. In contrast, under generic perturbations (bias/noise), low-degree correlations typically become nonzero ("linear footprints"), enabling an efficient first step.

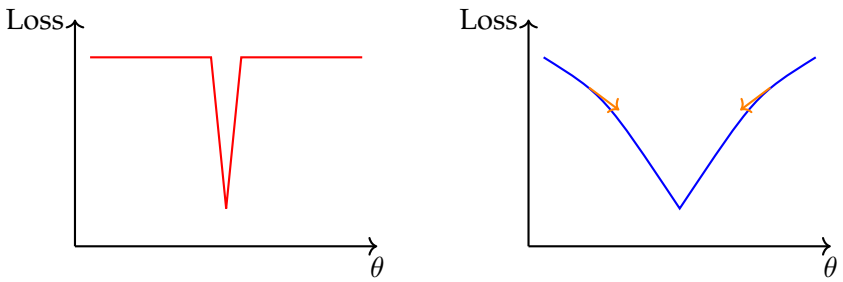
Takeaway. In this chapter we use *genericity* to explain why optimization succeeds in typical (smoothed) settings despite worst-case hardness, while recognizing that generalization requires additional assumptions (e.g., implicit/explicit regularization) beyond genericity.

6.6 From Genericity to Optimization Dynamics

How does this structural property explain the success of Deep Learning optimization?

6.6.1 The Gradient Signal at Initialization

Consider training a neural network $F(x;\theta)$ with gradient descent. At initialization, if $f(x)$ is Generic (e.g., has linear footprints due to coordinate shifts), it correlates with the simple features computed by the random network. Consequently, **the gradient is non-zero**.



(a) Non-Generic **(b) Generic**

Figure 6.3: Optimization Landscapes. (a) A non-generic function (like pure parity) creates a flat landscape where local search fails. (b) A generic function (with linear footprints) creates a tilted landscape.

6.6.2 The Optimization Hierarchy

Genericity suggests a hierarchical view of optimization dynamics:

1. *Breaking the Symmetry (Stage 1):* Because of coordinate shifts (biases), the target function exposes linear terms. The optimizer detects these first, moving away from the initial random plateau.

2. *Focusing Attention (Stage 2):* As weights grow, the effective "bias" inside the network shifts further, amplifying the signal for higher-order interactions.

3. *Solving the Core (Stage 3):* Having identified the relevant subspace, the network approximates the non-linear "pure" part.

6.6.3 Sample Complexity: The Staircase vs. The Cliff

The distinction between Generic and Non-Generic functions determines sample complexity.

- **Non-Generic Cost:** To find a parity of order k in dimension d without bias, one must search $\binom{d}{k}$ possibilities. Sample complexity is $O(d^k)$.
- **Generic Cost:** With linear footprints, the learner identifies relevant variables via screening in $O(d \log d)$, then solves the reduced problem. Genericity enables a **Staircase** learning curriculum [40].

6.7 Case Study: The Danger of Residual Fitting

The theory of Genericity clarifies a critical difference between two modeling strategies. Consider a target function $f(x)$ containing both linear and high-order nonlinear components.

- **Case A (End-to-End):** Train a network to approximate $f(x)$ directly.
- **Case B (Residual Fitting):** First, fit the best linear model $L(x)$. Then, train a network on the *residual* $r(x) = f(x) - L(x)$.

Intuitively, Case B seems superior. However, the Genericity Principle suggests Case A is sample-efficient, while Case B can destroy the initialization signal that many gradient-based methods rely on.

Why Case B Fails (Kicking Away the Ladder). In Case A, the linear footprints of $f(x)$ guide the optimizer. In Case B, step 1 removes these footprints. By definition, the residual $r(x)$ is orthogonal to the linear span of the inputs: $\mathbb{E}[r(x) \cdot x] = 0$. We have explicitly engineered a **Non-Generic** target function. The residual is purely nonlinear, making the gradient signal at initialization uninformative.

> *Conclusion:* Fitting the residual transforms a tractable "Staircase" problem into an intractable "Cliff" problem. The linear term is not just noise to be subtracted; it is the essential guide that makes the nonlinear optimization feasible.

6.8 Summary

The "unreasonable effectiveness" of deep learning optimization is not a property of the optimizer alone, but of the physics of the data.

1. *Nature is Generic:* Physical laws are invariant to coordinate shifts. This breaks symmetries that would otherwise hide structure.
2. *Genericity is about optimization signal:* Genericity does not by itself guarantee generalization, but it helps explain why gradient-based methods avoid “blind” initialization and can efficiently find predictive structure in typical (smoothed) settings.
3. *Structure Leaks:* In generic coordinates, high-order interactions cast "linear footprints."
4. *Optimization Succeeds:* These footprints ensure non-zero gradients at initialization, allowing local search to succeed.

CHAPTER 7

Principles of Deep Learning

Modern deep learning works far better than classical statistical learning theory or traditional numerical analysis would lead one to expect. In this chapter we articulate two basic principles that, taken together, offer an explanation for the empirical success of deep architectures: (1) the sparse compositionality *of efficiently computable functions, and (2) the* genericity *required for stability and learnability. Sparse compositionality explains approximation, hierarchy, modularity, and transfer. Genericity explains stability and the existence of informative gradients during optimization. Together these principles provide a coherent framework for understanding the remarkable capabilities of deep networks and transformers.*

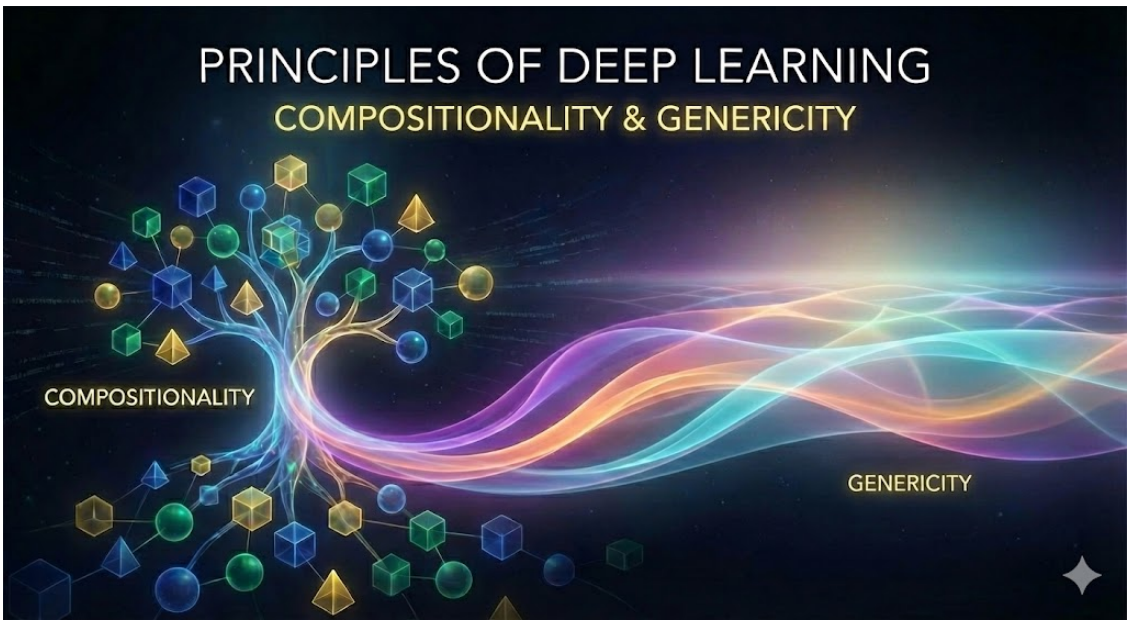


Figure 7.1

7.1 Principle I: Sparse Compositionality

The property of *sparse compositionality* is formally implied by the property of *efficient Turing computability* [189]. Functions that are efficiently computable necessarily decompose into bounded-fan-

in computational graphs (DAGs). This is a structural fact: a computation composed of many small local operations must be sparse in the sense that each intermediate value depends only on a few others [129].

Formally, if f is efficiently computable, then it admits a representation

$$f = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}$$

in which each $f^{(\ell)}$ acts on a small number of variables. Deep networks mirror this structure and thus avoid the curse of dimensionality when their architecture matches such sparse DAGs [154].

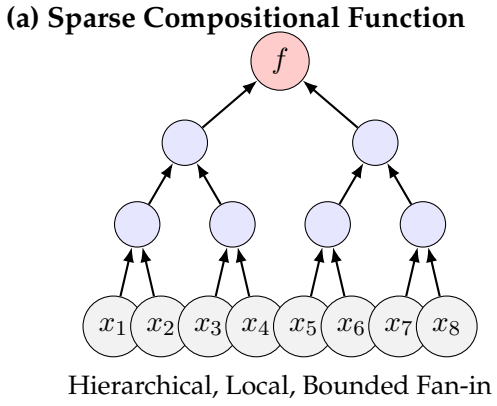


Figure 7.2: Visualizing Sparse Compositionality. An efficiently computable function naturally decomposes into a hierarchical DAG where each node depends on a small number of inputs (bounded fan-in). This structure allows deep networks to approximate the function without incurring the curse of dimensionality.

In this section we describe several consequences of sparsity and compositionality.

7.1.1 Hierarchy follows from sparse compositionality

If a function admits a compositional DAG in which each node has bounded fanin, then the natural architecture for approximating it is a *hierarchical network*: each layer computes simple local features, which become inputs to features of increasing complexity.

Hierarchy is not an extra assumption; it is a direct corollary of the DAG.

Thus the empirical fact that good deep networks are deep is not surprising: depth is the natural way to reflect the compositional structure of the target.

7.1.2 Modularity and reuse of modules

Sparse compositionality also implies *modularity*:

- subgraphs (subfunctions) appear repeatedly in the DAG,
- these subgraphs compute intermediate concepts,
- the same module can be used at different places in the computation.

This modularity is reflected in architectures such as convolutional networks, residual networks, and attention blocks, where learned components are reused globally or across layers.

The existence of modules is not a property of the architecture alone; it is a property of the underlying function.

7.1.3 Transfer learning

If a function decomposes compositionally and modules appear in multiple places, then learning one task provides information useful for others.

$$\text{Module learned in Task A} \quad \Rightarrow \quad \text{usable immediately in Task B.}$$

Thus *transfer learning* is a natural consequence of sparse compositionality: reuse across tasks parallels reuse across positions in a single computation [20].

7.1.4 Interpretability from consistent compositionality

If the decomposition of f into modules is *consistent across data*, then the learned network will assign stable semantic meaning to its internal components. This yields a form of interpretability:

$$\text{same subfunction} \quad \Rightarrow \quad \text{same module/layer.}$$

Thus interpretability is a consequence of:

- the existence of a compositional structure in the function, and
- the ability of the model to discover and consistently reuse that structure.

7.1.5 Non-uniqueness of sparse decompositions

A crucial observation is that *sparse decompositions are not unique*. A given function f may admit many distinct DAG representations. In such cases, a general-purpose architecture—such as a transformer—has no reason to choose the same decomposition across different inputs or during training.

Consequences:

- Transformers do not consistently reuse modules: the same computation can be implemented in many ways. the same computation can be implemented in many ways.
- Without architectural constraints enforcing modularity, reuse, or locality, the model may realize different decompositions in different parts of the input space.
- This explains why transformers are powerful but not inherently interpretable or modular.

Transformers achieve excellent performance even without consistent module reuse because they can realize *any* decomposition compatible with their large associative memory structure.

7.2 Principle II: Genericity of Learnable Targets

Sparse compositionality describes the *structure* of computable functions. Genericity describes a different property: the requirement that learnable functions be *invariant* in their basic jet structure under shifts of the coordinates. Genericity implies that the function cannot rely on measure-zero cancellations or extremely fragile high-order interactions [179].

7.2.1 Genericity from invariance to the choice of the origin of the coordinates

Genericity excludes "non-generic" functions in which:

- all low-order terms vanish exactly,
- small perturbations of the input distribution flip the identity of the best predictor,

Genericity is the opposite situation: under realistic noise and bias, *every relevant variable leaves a low-degree footprint*, often linear, that persists across perturbations.

Thus:

$$\text{Stability} \quad \Rightarrow \quad \text{Genericity (low-degree footprints, no fine-tuning).}$$

7.2.2 Genericity ensures good gradients for optimization

Once genericity holds, gradient-based optimization is feasible:

- Linear footprints provide nonzero first-order information.
- Gradients do not vanish due to high-order cancellations.
- Different layers of a deep network receive consistent signals.

Thus genericity provides the missing piece for explaining why gradient descent can find good solutions in nonconvex architectures.

7.3 Two independent but complementary principles

Sparse compositionality and genericity arise from two different fundamental assumptions:

- Efficient Turing computability $\Rightarrow$ sparse compositionality.
- ERM-based learnability + stability $\Rightarrow$ genericity.

They are independent but complementary:

- Sparse compositionality explains approximation, hierarchy, modularity, transfer, interpretability.
- Genericity explains stability, robustness, and the existence of informative gradients.

Together they provide a unified explanation for why deep learning works.

$$\text{Sparse compositionality} + \text{Genericity} \quad \Longrightarrow \quad \begin{cases} \text{Efficient approximation} \\ \text{Generalization (stability)} \\ \text{Optimizable landscapes} \\ \text{Hierarchy, modularity, reuse} \\ \text{Transfer and interpretability} \end{cases}$$

7.4 Conclusion

The two principles articulated in this chapter—sparse compositionality and genericity—capture complementary aspects of deep learning:

- the *structure* of the functions that we want to learn, and
- the *regularity* that makes them learnable and optimizable.

Most of the puzzles of deep learning—its scalability, generalization, modularity, transfer, and surprising ease of optimization—follow naturally from these two principles once they are recognized as the structural and regularity constraints of the real functions we want to learn.

Part II
Computation and Algorithms

CHAPTER 8

Efficient Computability, Compositional Sparsity, and Self-Attention

We establish a precise mathematical bridge between the result that efficient computability implies 1) deep, compositionally sparse approximants and 2) the structure of self-attention in Transformers.

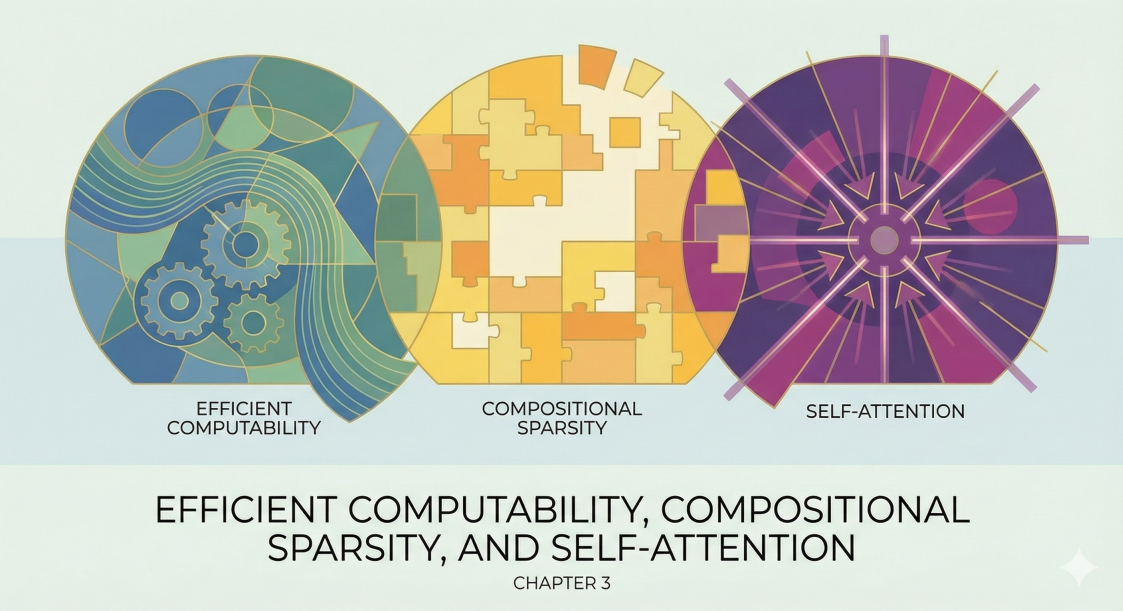


Figure 8.1

- *Attention implements a Gaussian kernel smoother (Nadaraya–Watson estimator) with a learned Mahalanobis metric.*
- *Any (k, s, L)-compositionally sparse target with node smoothness C^r can be uniformly approximated by a stack of attention+MLP blocks using $N_{\text{keys}} = \mathcal{O}(s\,\varepsilon^{-k/r})$ keys, achieving error $\leq \varepsilon$ independent of ambient dimension.*
- *If f is efficiently Turing-computable, there exists a Transformer of size $\text{poly}(n + \log(1/\varepsilon))$ with uniform error $\leq \varepsilon$ on n-bit inputs.*

- *Sparsity and low rank emerge naturally: when top-k neighbors are separated by a margin, attention weights outside the top-k are exponentially small; rank-$\leq k$ query/key metrics suffice for optimal rates.*

8.1 Preliminaries and Assumptions

We consider functions $f : [-1, 1]^n \to \mathbb{R}$ representable as compositionally sparse DAGs: each node v has fan-in $m_v \leq k$ and computes $g_v \in C^r(\Omega_v)$ with bounded C^r norm $\|g_v\|_{C^r} \leq B$. Each layer map is L_ℓ-Lipschitz. This structure encompasses the class of functions approximable by deep networks of bounded local arity [128, 147].

A self-attention head uses parameters $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$, temperature $\tau > 0$, and computes for query x and stored keys $\{x_j\}$:

$$\mathrm{Attn}(x) = \sum_{j=1}^{N} \alpha_j(x)\, v_j, \qquad \alpha_j(x) = \frac{\exp\bigl(q^\top k_j/\tau\bigr)}{\sum_\ell \exp(q^\top k_\ell/\tau)}, \quad q = W_Q x,\ k_j = W_K x_j,\ v_j = W_V x_j.$$

LayerNorm ensures $\|q\|, \|k_j\| \approx 1$.

—

8.2 Main Results

Theorem 1 (Self-Attention as Gaussian Nadaraya–Watson Smoother)

Under approximate normalization $\|q\|, \|k_j\| \approx 1$,

$$\alpha_j(x) \ \propto\ \exp\Bigl(-\tfrac{1}{2\tau}\|q - k_j\|^2\Bigr),$$

i.e. attention weights coincide with Gaussian Nadaraya–Watson kernel weights with bandwidth τ. If $W_Q = W_K = B$, the distance $\|q-k_j\|^2 = \|B(x-x_j)\|^2 = (x-x_j)^\top(B^\top B)(x-x_j)$ is a Mahalanobis metric with matrix $B^\top B$.

Proof. For unit vectors, $q^\top k_j = 1 - \frac{1}{2}\|q - k_j\|^2$. Exponentiating and normalizing in the softmax gives the stated Gaussian form. □

This formalizes earlier observations that attention is a data-dependent kernel smoother [186, 33]. For discussion of a more efficient attention see Appendix F.

8.2.1 Theorem 2 (Compositional Approximation by Attention with Dimension-Free Rate)

Let $f : [-1, 1]^n \to \mathbb{R}$ admit a compositional DAG representation of depth L and size s with nodes $g_v \in C^r$ on at most k variables. Then for any $\varepsilon \in (0, 1)$ there exists a stack of attention+MLP blocks of depth $\mathcal{O}(L)$ such that

$$\sup_{x\in[-1,1]^n} |f(x) - \widehat{f}(x)| \leq \varepsilon,$$

using a total number of keys

$$N_{\mathrm{keys}} \leq C\, s\, \varepsilon^{-k/r},$$

where C depends on k, r, B and the layer Lipschitz constants, and each head's projection B has rank $\leq k$.

Proof Sketch. Each node g_v is a C^r function on $\mathbb{R}^{m_v}$, $m_v \le k$. By standard Gaussian RBF/Nadaraya–Watson approximation results [127, 45], there exist centers $\{x_j^{(v)}\}$ such that

$$\|g_v - \tilde{g}_v\|_\infty \le \delta, \qquad N_v = \mathcal{O}((B/\delta)^{m_v/r}).$$

Each $\tilde{g}_v$ can be realized by one attention head using keys $\{x_j^{(v)}\}$ and values $v_j = g_v(x_j^{(v)})$ (Theorem 1). Composing across L layers yields global error ε when $\delta = \varepsilon/(L\prod_\ell L_\ell)$. Summing N_v gives $N_{\text{keys}} = \mathcal{O}(s\,\varepsilon^{-k/r})$. □

Discussion. The approximation rate depends on the local arity k, not on ambient dimension n. This reproduces the dimension-free bounds of compositionally sparse networks [147, 112].

—

8.2.2 Theorem 3 (Efficient Computability ⇒ Transformer Approximants)

Assume f is efficiently Turing-computable in time $\text{poly}(n + \log(1/\varepsilon))$ on n-bit inputs. Then there exists a family of Transformers $\{\mathcal{T}_{n,\varepsilon}\}$ of size $\text{poly}(n + \log(1/\varepsilon))$ such that

$$\|\mathcal{T}_{n,\varepsilon}(x) - f(x)\| \le \varepsilon$$

for all n-bit inputs $x \in [-1, 1]^n$.

Proof. By the CBMM Memo 072 theorem [148], any efficiently computable function admits a deep network Φ_ε with polynomial size and compositional sparsity (k, s, L). By Theorem 8.2.1, each node of Φ_ε can be replaced by an attention+MLP module of comparable size. Hence overall size/depth remain polynomial and the uniform approximation error ε is preserved. □

Remarks. This establishes that self-attention architectures are expressive enough to realize all efficiently computable functions with polynomial resources, matching the constructive bounds previously shown for generic sparse deep nets.

—

8.2.3 Theorem 4 (Margin Implies Near Top-k Sparsity)

Let squared distances $\Delta_j = \|q - k_j\|^2$ be ordered $\Delta_{(1)} \le \cdots \le \Delta_{(N)}$. Then for any $k < N$,

$$1 - \sum_{j=1}^{k} \alpha_{(j)}(q) \;\le\; (N-k)\exp\Big(-\frac{\Delta_{(k+1)} - \Delta_{(1)}}{2\tau}\Big).$$

If $\Delta_{(k+1)} - \Delta_{(1)} \ge \gamma > 0$, the tail mass outside top-k is $\le (N-k)e^{-\gamma/(2\tau)}$.

Proof. Factor $\exp\Big(-\Delta_{(1)}/(2\tau)\Big)$ in numerator and denominator and bound the remaining terms. □

Implication. If relevant keys are separated by a finite margin, attention becomes effectively k-sparse. This explains empirical top-k sparsity observed in trained Transformers [130, 195, 34].

—

8.2.4 Theorem 5 (Low Rank Suffices for k-Ary Nodes)

If $g : \mathbb{R}^m \to \mathbb{R}$, $m \leq k$, admits a Gaussian NW approximant of error δ using $N = \mathcal{O}((B/\delta)^{m/r})$ centers, then there exists a rank-m projection B such that one attention head realizes this approximant. The exponent m/r is optimal for C^r functions [45].

Proof. Let B be the embedding $\mathbb{R}^m \hookrightarrow \mathbb{R}^{d_k}$. The Gaussian NW approximant uses kernels $\exp(-\|B(x - x_j)\|^2/(2\tau))$ of rank m. Known lower bounds for C^r functions show that no method can achieve better exponent than m/r. □

Consequence. Optimal attention heads need only rank $\leq k$ in their query/key projections. Empirically, learned W_Q, W_K often have low effective rank [139, 2].

—

8.3 Consequences and Predictions

- **Dimension-free rates.** Approximation error scales as $\varepsilon^{-k/r}$, independent of token dimension d.
- **Emergent sparsity.** Margins in key–query distances imply exponentially decaying tails (Theorem 4).
- **Low-rank projections.** Effective rank per head matches local arity k (Theorem 5).
- **Expressivity.** Every efficiently computable function has a polynomial-size Transformer realization (Theorem 3).

—

8.4 Empirical Consistency

Empirical findings across many studies align with these theoretical predictions:

Head specialization. A small fraction of heads dominate performance [130, 195, 34], consistent with low-arity compositional structure.

Low-rank matrices. Empirical spectra of W_Q, W_K, W_V show low effective rank [139, 2].

Attention sparsity. Learned attention distributions become nearly top-k sparse at convergence, especially in deeper layers.

Dimension-independent scaling. Transformer performance follows power-law scaling $\varepsilon \sim N^{-\alpha}$ without dependence on embedding dimension [92, 86].

—

8.5 Limitations and Open Problems

- Early layers sometimes exhibit diffuse attention; theory predicts this corresponds to wide-band kernels providing coarse averaging.
- The low-rank assumption holds empirically but lacks a purely theoretical guarantee from the training dynamics.
- The results assume smooth local maps (C^r); extending to nonsmooth or discrete combinatorial tasks is open.

8.6 Conclusion

Self-attention implements the same structural principle uncovered in the CBMM theory: efficiently computable functions decompose into locally low-dimensional components. An attention head is a Gaussian kernel smoother in a learned metric; multi-head attention forms mixtures of such smoothers; and stacked layers yield composition-respecting approximants with dimension-free rates governed by local arity k. Transformers thus concretely instantiate the principle:

$$\text{Efficient Computability} \;\Rightarrow\; \text{Compositional Sparsity} \;\Rightarrow\; \text{Dimension-Free Deep Approximation.}$$

CHAPTER 9

Hardware for Compositionally Sparse Computation (with J. Bates)

The central thesis of this book—that efficiently computable functions *are compositionally sparse—has profound implications for hardware. If the mathematical structure of efficiently computable functions consists of bounded-fan-in DAGs with local dependencies, then the optimal physical substrate is not a monolithic matrix multiplier, but* **a 2D mesh of simple processing elements with nearest-neighbor communication** *– as suggested by Joe Bates. This chapter argues for a "structural alignment" between algorithm and hardware. We explore architectures where the physical layout mirrors the computational DAG, minimizing energy spent on data movement and maximizing locality. We specifically analyze how Transformer blocks (MLP and Attention) map onto such meshes using approximate HDR arithmetic and localized routing.*

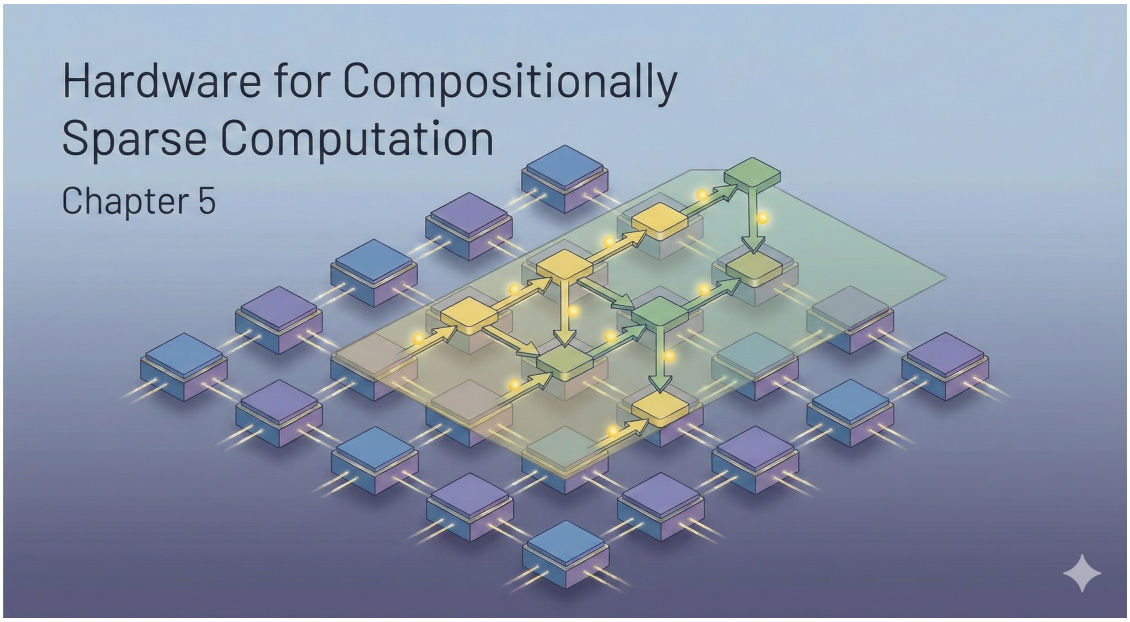


Figure 9.1

9.1 The Structural Alignment Argument

The prevailing trend in AI hardware has been to build massive, centralized matrix engines (like GPUs and TPUs) designed for dense, all-to-all connectivity [89]. This approach assumes that the underlying computation is generic dense matrix multiplication.

However, as established in Chapter 4, **efficiently computable functions** are not generic dense blobs. They are **Compositionally Sparse**: they decompose into directed acyclic graphs (DAGs) where:

- **Bounded Fan-in:** Each node depends on a small number of inputs.
- **Locality:** Modules interact primarily with their logical neighbors.

This theoretical insight suggests a different hardware paradigm. Instead of building hardware for dense connectivity, we should build hardware that is **isomorphic** to the sparse compositional structure.

- **The Mathematical Structure:** A graph of local, low-arity operations.
- **The Physical Structure:** A 2D mesh of simple Processing Elements (PEs) with local interconnects [13].

By mapping the logical parents of a node to physical neighbors on the chip, we convert "wiring costs" (which dominate energy consumption) into "local hops," aligning the physics of the chip with the math of the function [101].

[Image of 2D mesh network topology vs crossbar]

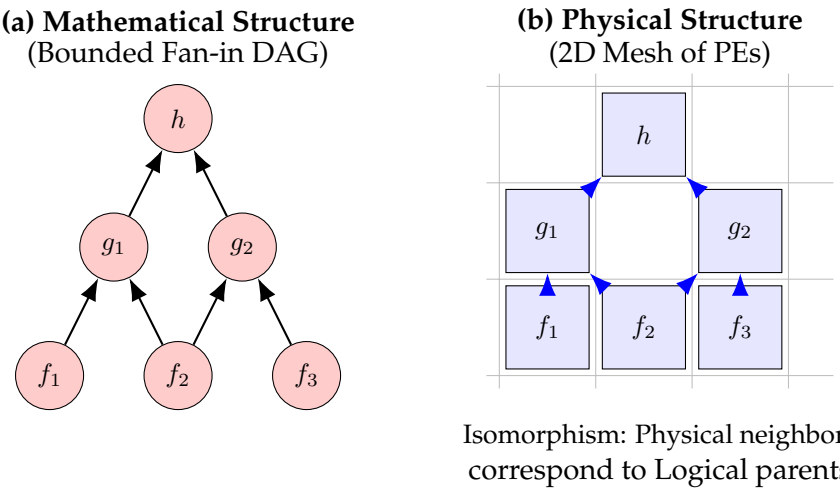


Figure 9.2: Structural Alignment. The bounded fan-in DAG of an efficiently computable function (a) maps naturally onto a 2D Mesh topology (b). By placing logically connected nodes on physically adjacent tiles, communication remains local, avoiding the need for global crossbars.

9.2 System and Numeric Model

Compute Fabric. We assume a 2D mesh of P simple processing elements arranged as a $\sqrt{P} \times \sqrt{P}$ grid.

- **Approximate Arithmetic Units:** Leveraging Logarithmic Number Systems (LNS) or stochastic logic to perform high-dynamic range operations with low area cost [72, 14];
- Small local SRAM ("tile memory") for vector tiles and KV storage;
- Four nearest-neighbor links (N/E/S/W) for hop-by-hop routing.

The vision is to build a *locality-driven* architecture that trades bit-perfect deterministic accuracy for massive parallelism. By eschewing wide crossbars and IEEE floating-point cores, we prioritize wire efficiency and density [103].

Numeric Format: The Case for Approximate HDR. Instead of standard integers, we utilize a High Dynamic Range (HDR) format with approximate semantics (e.g., LNS). In the log domain, multiplication reduces to fixed-point addition ($\log(xy) = \log x + \log y$), drastically reducing the transistor count of the Processing Elements. Crucially, HDR supports the vast numerical scales encountered during backpropagation (gradients ranging from 10^{-7} to 10^{2}), enabling *training* on the mesh. We assume a relative error tolerance of $\approx 1\%$, which deep networks have been shown to absorb without degradation [15, 131].

9.3 Mapping Compositional DAGs to 2D Meshes

We consider DAGs G of depth L with layer widths W_ℓ and fan-in $\leq k$. A natural layout places each layer ℓ on a $\sqrt{W_\ell} \times \sqrt{W_\ell}$ patch with parents at layer $\ell-1$ assigned to adjacent patches.

Proposition 5 (2D schedule for bounded fan-in DAGs)**.** *Let each layer ℓ be placed on a $\sqrt{W_\ell} \times \sqrt{W_\ell}$ region. Then the latency to realize layer ℓ on a 2D mesh is*

$$T_\ell = T_{comp,\ell} + T_{comm,\ell},$$
$$T_{comm,\ell} = \mathcal{O}(kD_\ell + \mathrm{cong}_\ell),$$

where D_ℓ is the Manhattan diameter of dependencies and cong_ℓ is max per-link congestion. If edge placement uses locality-aware heuristics (e.g., hypergraph partitioning), then $D_\ell = \mathcal{O}(\sqrt{W_\ell})$ in general, and $D_\ell = \mathcal{O}(1)$ if G has bounded treewidth.

Constraint: Hierarchical Locality and Rent's Rule. We must address a subtle topological constraint: bounded fan-in alone is *insufficient* to guarantee a compact 2D embedding. A random sparse graph (an expander) has bounded fan-in but requires global wires that scale with the square root of the chip area, destroying locality.

For the structural alignment argument to hold, the computational DAG must exhibit *hierarchical locality*. Formally, the graph must satisfy a generalized **Rent's Rule** [105], where the number of terminals T required to communicate with a cluster of G logic gates scales as:

$$T \propto G^p \tag{9.1}$$

To map efficiently to a 2D mesh without excessive congestion, the Rent exponent must satisfy $p \leq 0.5$. We conjecture that *efficiently computable functions* naturally satisfy this geometric constraint, as they typically decompose into semi-independent modules (objects, concepts) rather than monolithic global dependencies.

[Image of Rent's rule log-log plot for VLSI circuits]

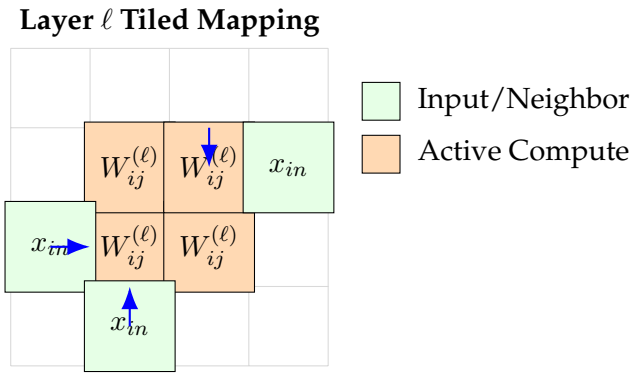


Figure 9.3: Tiled Mapping of a Neural Layer. A large layer W_ℓ is decomposed into tiles distributed across the mesh. Data flows from neighboring regions (Layer $\ell - 1$) to the active computation tiles using only local routing, avoiding global broadcasts.

MLP Modules. When viewed as feedforward layers with fan-in k, the MLPs in a transformer are ideally suited to such layout. Each layer can be broadcast in tiles, and inputs fetched from neighbors within a few hops. With sparsity or block structure in the weights (e.g., compositionally local weight matrices), compute remains bounded and placement near-optimal.

Speculatively, MLP submodules might be structured as locally reused kernels applied in a recurrent fashion over overlapping scopes [30]. This suggests a kind of *sliding compositionality*, where each kernel operates over a spatial patch of nearby tokens (or semantic units). In this case, the 2D mesh becomes a physical substrate for composing locality-aware update modules.

9.4 Sparse Attention on 2D Meshes

We explore two strategies, KV-stationary and Q-stationary, for evaluating top-s attention in a distributed mesh.

9.4.1 KV-Stationary: Coarse-to-Fine Attention

Distribute keys and values across PEs; queries are routed to buckets [97]:

1. **Coarse filter**: compute $c(q) = \text{sign}(Bq)$, a low-rank code of q;
2. **Fine score**: at bucket PEs, evaluate $\langle q, k_i \rangle$ and select top-s;
3. **Mix**: normalize (softmax-lite) and compute weighted sum $\sum_i \alpha_i v_i$.

This coarse-to-fine filtering exploits structure in the attention map and avoids global broadcasts.

9.4.2 Q-Stationary: Systolic Streaming with Early Exit

Store queries locally, and stream (K, V) across PEs. Maintain top-s heaps in-place; prune keys that cannot contribute based on conservative score bounds [42].

[Image of systolic array architecture diagram]

Speculative Opportunity. If attention graphs are induced by a latent hierarchical function (e.g., a parse or program tree), then coarse bucketings might correspond to semantic equivalence classes. The mesh becomes a canvas for learning or reflecting emergent structure.

9.5 AM Quantization and Error Analysis

We adopt two softmax approximations compatible with integer hardware:

- **Blockwise softmax**: via integer max and LUTs [208];
- **Tempered max-mix**: softmax-lite using ReLU and shifts only.

Error bounds (e.g., Lemma **??**) link quantization precision Δ to output fidelity. We derive end-to-end degradation (Proposition **??**) and provide explicit tolerances for b-bit quantizers [44, 70].

9.6 Discussion: Why 2D Meshes Help

- **Local fan-in and attention sparsity** shrink receptive fields, making local routing sufficient.
- **Spatial bucket locality** and compositional weight reuse enable static co-placement of related scopes [31].
- **Fixed-point compatibility** ensures bit-serial or SIMD units suffice, reducing energy and area.
- **Emergent structure discovery**: If learning induces local composition or symmetry, the mesh layout might co-adapt.

9.7 Conclusion

We presented a speculative but rigorous framework for mapping compositionally sparse computations—typical of transformer blocks—to 2D meshes of simple processing elements. Such mappings exploit low arity, sparse attention, and localized updates to minimize communication. The result is a hardware substrate that leverages structural regularities in both the algorithm and the data, favoring integer arithmetic, approximate softmax, and hop-local routing. Future work may explore adaptive mesh reconfiguration, dynamic bucketing strategies, and learning-augmented placement.

The broader implication is that sparsity—far from being a nuisance—may be a blessing. Compositional structure enables hardware mappings with minimal overhead and supports scalable, efficient deployment of deep learning systems on modest, spatially organized fabrics.

CHAPTER 10

A Common Principle Underlying Diffusion Models and Transformers

We propose a unifying theoretical principle: step–by–step supervision of partial states—whether in an autoregressive *or a* diffusion *format—is sufficient to emulate any polynomial-time Turing computation, even when each step is implemented by a simple linear threshold (or small Boolean) model. The key problem is to provide appropriate input-output training data. Concretely, for every function f computable by a Turing machine in polynomial time, there exists a dataset of intermediate configurations such that training a linear predictor on successive states—either $s_t \mapsto s_{t+1}$ (autoregressive) or $s_t \mapsto s_{t-1}$ (diffusion)—recovers f. The same formal structure explains why transformers and diffusion models, though architecturally distinct, both succeed through stepwise prediction of local transitions, since they both provide the necessary training sets.* **mention way to avoid backprop...used by the brain? intially for compression learning from hippocampus....evolutinary beginning of world models...**

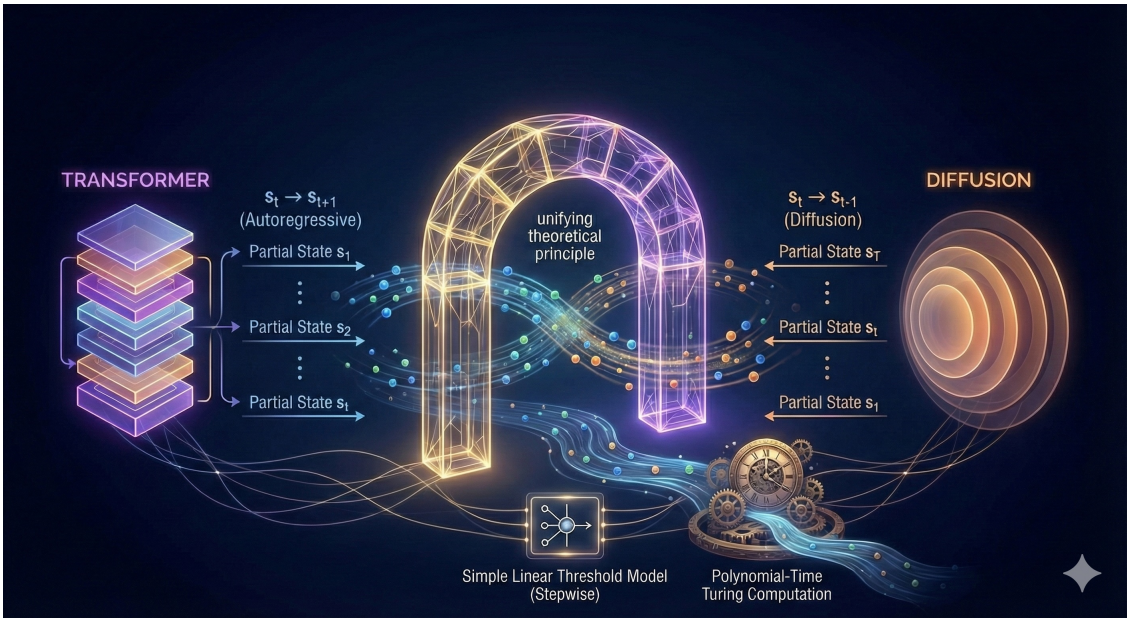


Figure 10.1

10.1 Introduction

Large language models such as GPT transformers are trained by *autoregressive* next-token prediction: each step predicts x_{t+1} from the current prefix $x_{\leq t}$. Diffusion models, by contrast, learn to *denoise*: each step predicts the less noisy sample x_{t-1} from x_t. Despite this difference in direction, both training schemes learn a local update map between adjacent states.

This chapter formalizes that correspondence. We show that, with suitably designed representations, a simple linear threshold predictor trained on step-wise supervision can implement any polynomial-time Turing machine. Hence, the principle of **learning local transitions between partial states** suffices for universal computation.

Our analysis parallels the recent result of Malach, who proved universality of autoregressive next-token learners [118]. We restate his theorem in a compact "unrolled-Turing" formulation and extend the same logic to diffusion-style denoisers, including a version with true Gaussian noise. Finally, we discuss why transformers and diffusion models can both be viewed as instances of the same compositional mechanism.

10.2 Preliminaries

Polynomial-time computable functions. Let Σ be a finite alphabet and let $f : \Sigma^n \to \Sigma^m$ be a function computed by a deterministic Turing machine M in time $T(n) = \mathrm{poly}(n)$. A *configuration* of M at time t—its tape contents, head position, and internal state—is denoted τ_t. We encode each configuration by a binary vector $s_t \in \{0,1\}^d$ of dimension $d = \mathrm{poly}(T(n))$; for instance, each tape cell and state symbol can be represented in one-hot form.

Linear threshold and circuit predictors. A linear threshold function (LTF) is

$$h_{\mathbf{w},b}(z) = \begin{cases} 1, & \langle \mathbf{w}, z \rangle + b \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

Multiple LTFs in parallel, or equivalently a shallow threshold circuit, can produce multi-bit outputs. All results below remain valid if we replace the LTF by any polynomial-size Boolean circuit implementing the same local update. The replacement of linear threshold predictors by Boolean circuits, formalized in an Appendix of chapter 4, is implicitly used here to establish the universality of stepwise diffusion models.

Stepwise training paradigms. We distinguish two kinds of supervision:

1. **Autoregressive (AR) prediction:** learn the mapping $s_t \mapsto s_{t+1}$ that advances the machine one step forward.
2. **Diffusion-style denoising:** learn the mapping $s_t \mapsto s_{t-1}$ (or, in continuous variants, $\hat{x}_t \mapsto \hat{x}_{t-1}$) that reconstructs a cleaner previous state.

10.3 Stepwise Universality of Autoregressive Predictors

Theorem 8 (Autoregressive universality (see Malach, 2023)). *Let $f : \Sigma^n \to \Sigma^m$ be computable by a Turing machine M in time $T(n)$. Then there exists a polynomial-size dataset $\mathcal{D}$ of pairs (s_t, s_{t+1}), and a linear threshold (or small circuit) predictor trained on the next-token loss over $\mathcal{D}$, such that the iterated predictor reproduces f with high probability over inputs drawn from any distribution D.*

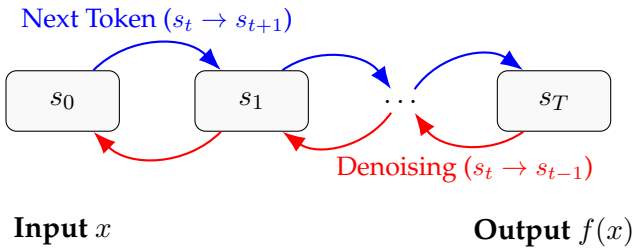


Figure 10.2: Two sides of the same coin. Autoregressive models (top) learn to predict the next computational state, simulating the Turing machine forward. Diffusion models (bottom) learn to recover the previous computational state, effectively reversing the computational trajectory (or denoising it). Both rely on learning local transitions.

Proof sketch. Unroll M on an input x for $T(n)$ steps, producing configurations $\tau_0, \ldots, \tau_{T(n)}$ and encodings $s_0, \ldots, s_{T(n)}$. Because each transition of M updates only a constant-size local neighborhood (current head position, symbol, and state), each bit of s_{t+1} depends on a fixed small subset of bits of s_t. With a one-hot representation, each such rule can be realized by a linear threshold unit; collecting all outputs gives a single layer H with d parallel thresholds so that $H(s_t) = s_{t+1}$. Training H on all transitions (s_t, s_{t+1}) from a polynomial number of simulated runs yields a model that generalizes to new x, and iterating H for $T(n)$ steps outputs $s_{T(n)}$ encoding $f(x)$ [7].

Remark. The key point is that the expressive power resides in the training set—which exposes every local transition—not in the nonlinearity of the model. Depth is replaced by supervision across time.

10.4 Stepwise Universality of Diffusion Predictors

Theorem 9 (Diffusion-step universality)**.** *For any polynomial-time computable f as above, there exists a discrete diffusion process with states $s_T, \ldots, s_0$ such that a linear threshold (or small circuit) predictor trained on pairs (s_t, s_{t-1}) learns a denoising map whose $T(n)$-fold composition recovers f.*

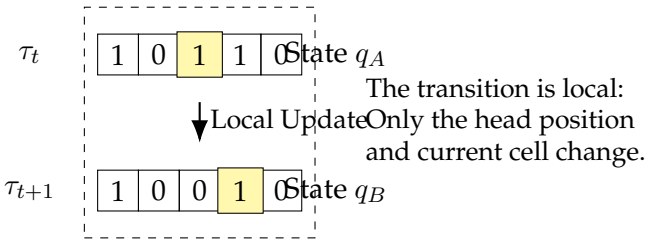


Figure 10.3: The Local Transition Principle. Because Turing Machines operate locally, the mapping between any configuration τ_t and its successor τ_{t+1} (or predecessor τ_{t-1}) is simple. This allows a shallow network to learn the "rules of physics" for the computation, provided it is trained on the intermediate steps.

Idea. Interpret the "denoising" map $G(s_t) = s_{t-1}$ as the time-reversed Turing transition $\tau_t \mapsto \tau_{t-1}$. Because each step of the Turing machine is local, the same linear threshold construction as in the autoregressive case implements G. Training on all consecutive pairs (s_t, s_{t-1}) teaches the model the inverse transition rule. Iterating G for $T(n)$ steps reconstructs the initial or output configuration depending on direction. Thus, a diffusion-style learner trained stepwise has identical computational universality.

Continuous or noisy variants. In Appendix C we show that this result extends to true Gaussian diffusion: each Turing configuration is embedded as a one-hot vector s_t; forward diffusion adds Gaussian noise; and a linear threshold denoiser can still identify the active coordinate and apply the discrete update rule with high probability [77].

10.5 Discussion: Transformers and Diffusion as Stepwise Computation

Both autoregressive and diffusion training teach a model to approximate a local transition map on an implicit computational trajectory. The key is to exploit appropriate training sets.

- In an **autoregressive transformer**, training data supervise the forward transition $s_t \mapsto s_{t+1}$—learning to predict the *next* state in a computation.
- In a **diffusion model**, training data supervise the reverse transition $s_t \mapsto s_{t-1}$—learning to reconstruct the *previous* state in a computation corrupted by noise. Here the step-wise training data are created by successive "noising" of the target.

From this viewpoint, both architectures instantiate the same universal mechanism: a composition of simple, local updates whose repetition yields arbitrarily complex functions. Transformers "integrate" forward toward the minimum of a loss landscape, while diffusion models "integrate" outward from that minimum through inverse steps. The underlying reason for their power is therefore not architectural depth but the availability of dense stepwise training data that exposes every intermediate transition of an underlying computation [171].

10.6 Technical Note: Gaussian Diffusion and Noisy One-Hot Encodings

This appendix extends Theorem 9 to a *true* diffusion process with Gaussian noise, demonstrating that the universality argument still holds when the forward process adds continuous noise at each step.

10.6.1 Setup: Forward Diffusion with One-Hot Embeddings

Let each discrete Turing configuration τ_t be encoded by a one-hot (or scaled one-hot) vector

$$s_t \in \mathbb{R}^d, \qquad s_t(k) = \begin{cases} C, & \text{if coordinate } k \text{ represents } \tau_t, \\ 0, & \text{otherwise,} \end{cases}$$

where $C > 0$ sets the signal amplitude and $d = \mathrm{poly}(T(n))$.

Following the standard discrete-time diffusion formulation of Ho et al. [77], the forward process adds Gaussian noise:

$$x_t = \sqrt{\bar{\alpha}_t}\, s_t + \sqrt{1 - \bar{\alpha}_t}\, z_t, \qquad z_t \sim \mathcal{N}(0, I_d),$$

where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. The variable x_t is thus a noisy observation of s_t.

10.6.2 Recovering the Active Coordinate

Lemma 5 (High-probability identification)**.** *Let s_t be one-hot with amplitude C and x_t generated as above. If $C \gg \sqrt{1-\bar{\alpha}_t}$, then with probability at least $1 - e^{-\Omega(C^2/(1-\bar{\alpha}_t))}$, the index*

$$k^\star = \arg\max_j x_t(j)$$

equals the hot coordinate of s_t.

Sketch. For the active coordinate k, $x_t(k) \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}C, 1-\bar{\alpha}_t)$, while for all $j \neq k$, $x_t(j) \sim \mathcal{N}(0, 1-\bar{\alpha}_t)$. A standard Gaussian tail bound shows $\Pr[x_t(k) \leq x_t(j)] \leq \exp\Big(-\frac{(\sqrt{\bar{\alpha}_t}C)^2}{4(1-\bar{\alpha}_t)}\Big)$. A union bound over $d = \mathrm{poly}(T(n))$ coordinates yields the stated result. □

Hence, a simple *arg-max* operation recovers the correct coordinate with overwhelming probability whenever the signal-to-noise ratio $C^2/(1-\bar{\alpha}_t)$ is large.

10.6.3 Implementing the Turing-Step Update

Let the deterministic transition function of the Turing machine be $f : \{1, \ldots, d\} \to \{1, \ldots, d\}$, mapping each active coordinate k to the next one $k' = f(k)$.

After identifying $k^\star$ via Lemma 5, we must output a one-hot vector s_{t-1} with its non-zero entry at $k' = f(k^\star)$. This map can be implemented by a small Boolean or threshold circuit:

$$s_{t-1}(j) = \begin{cases} C, & \text{if } j = f(k^\star), \\ 0, & \text{otherwise.} \end{cases}$$

In particular, we can realize the overall denoising step as a network

$$D(x_t) = M\big(\mathrm{argmax}(x_t)\big),$$

where the first block computes the dominant coordinate (via linear comparisons or a small network of LTFs), and M encodes the lookup table $k \mapsto f(k)$. Both submodules are polynomial in d [176].

10.6.4 Training and Composition

Training proceeds exactly as in the discrete diffusion case: for each input x, unroll the Turing machine for $T(n)$ steps, generate noisy states x_t via the forward process, and supervise the denoiser with targets s_{t-1}. Because the hot coordinate can be recovered with high probability and the mapping $k \mapsto f(k)$ is deterministic.

CHAPTER 11

Lottery Ticket and Compositionality

Is there a fundamental connection between the compositionality of a task and the compressibility of the network that learns it? We propose that the "Lottery Ticket Hypothesis" is not merely an artifact of gradient descent, but a direct consequence of the compositional sparsity of the target function.

Figure 11.1: The Lottery Ticket Hypothesis as a Consequence of Compositional Sparsity

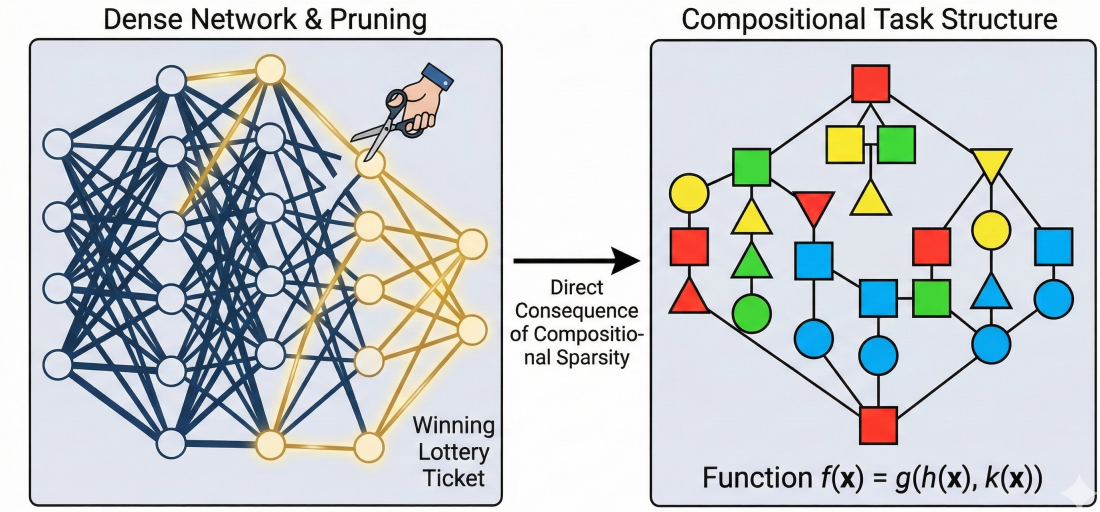


Figure 11.1

11.1 Introduction

The Lottery Ticket Hypothesis (LTH) suggests that dense neural networks contain sparse subnetworks ("winning tickets") that can match the performance of the full model [54]. While typically studied through the lens of optimization dynamics, we argue here for a structural origin: **tickets exist because the target function is compositionally sparse.**

If the function $f^\star$ we wish to learn can be represented by a sparse graph of local computations (a DAG of size $s \ll$ parameters), then the dense network's weight matrices are effectively approxi-

mating low-rank operators [129]. Consequently, the dense solution $\hat{\theta}$ must lie in the close vicinity of a low-dimensional manifold defined by sparse, low-rank networks.

This chapter formalizes this intuition. We prove an **Approximate Warm-Start Lottery Ticket Theorem**: essentially, that dense training implicitly locates the neighborhood of a compositional solution, allowing us to project (prune/decompose) onto that solution with minimal loss.

11.2 The Geometry of Sparsity

Before stating the formal result, it is helpful to visualize the geometric relationship between the dense optimization landscape and the manifold of compositionally sparse functions.

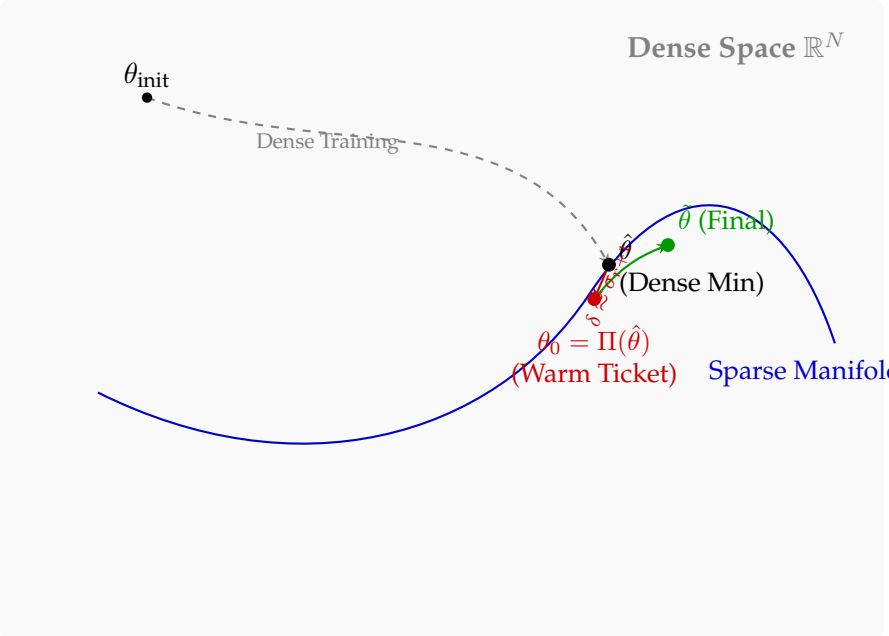


Figure 11.2: **The Geometric View of the Lottery Ticket Hypothesis.** Dense optimization finds a minimum $\hat{\theta}$ that is not sparse itself, but lies extremely close to the manifold $\Theta_{S,r}$ of compositionally sparse networks. The projection distance δ (red) is determined by the "tail energy" of the singular values (the non-compositional noise). Pruning is simply the orthogonal projection onto this manifold, providing a "warm start" θ_0 for restricted fine-tuning.

11.3 Approximate Lottery Ticket Theorem for Compositionally Sparse Functions

Theorem 10 (Approximate Lottery Ticket Hypothesis for Compositionally Sparse Targets)**.** *Let $f^\star \in \mathcal{C}_{\text{comp}}(s, k, L, r)$ be a* compositionally sparse *function, computable by a bounded-fan-in DAG of size s and depth L.*

Consider an overparameterized architecture $\mathcal{A}_{\text{dense}}$ (e.g., a Transformer or MLP) with layer weights $\{W_\ell\}_{\ell=1}^{L}$, trained to global convergence on data drawn from $\mathcal{D}$ under a loss $R(\theta)$. Assume that training converges to a minimum $\hat{\theta}$ with minimal norm.

Then there exists a structured subnetwork $\Theta_{S,r} \subset \mathcal{A}_{\text{dense}}$ with $|S| = \tilde{O}(s)$ active parameters and per-layer ranks $r_\ell = \tilde{O}(1)$ such that:

(a) (***Approximation***) *There exists* $\theta^\star \in \Theta_{S,r}$ *satisfying* $\|F_{\theta^\star} - f^\star\|_{L^2(\mathcal{D}_X)} \le \varepsilon$.

(b) (***Proximity of minima***) *The dense minimum* $\hat{\theta}$ *and its projection* $\theta_0 = \Pi_{S,r}(\hat{\theta})$ *yield almost identical functions,*

$$R(\theta_0) \;\le\; R(\hat{\theta}) + O(\varepsilon + \delta),$$

where δ *is the aggregate spectral/pruning error*

$$\delta \;\sim\; \sum_{\ell=1}^{L} (\sigma_{r_\ell+1}(W_\ell) + \|E_\ell^{\text{prune}}\|).$$

Moreover, fine-tuning θ_0 *within* $\Theta_{S,r}$ *converges to a subnetwork* $\tilde{\theta}$ *satisfying* $R(\tilde{\theta}) \;\le\; R(\hat{\theta}) + O(\varepsilon + \delta)$.

Remark 2 (Interpretation)**.** *Under minimal-norm convergence, dense optimization implicitly identifies a low-complexity region of parameter space containing a compositionally sparse subnetwork* $\Theta_{S,r}$ *of size* $\tilde{O}(s)$*. Projecting onto this manifold and retraining within it recovers the dense model's performance up to* $O(\varepsilon + \delta)$*. This provides a succinct form of the* approximate lottery ticket hypothesis *(see also [119]): the effective solution is already sparse and compositional, even if the training representation is dense.*

11.4 Refinements and Empirical Connections

We now strengthen the result by explicitly linking the projection error δ to the smoothness properties of the target function.

Corollary 1 (Spectral Decay and Smoothness in LLMs)**.** *The success of pruning and low-rank compression in Large Language Models is explained by the relationship between the smoothness of the target function and the singular value spectrum of the learned weights [124].*

(a) ***Activation Sparsity*** $\leftrightarrow$ ***Local Connectivity.*** *The observation that only a small fraction of neurons activate for any given token implies the underlying computation graph is effectively sparse (*$S_t \ll$ *total).*

(b) ***Weight Compressibility*** $\leftrightarrow$ ***Functional Smoothness.*** *If the constituent functions of the target composition* $f^\star$ *are* r*-times differentiable (smooth), approximation theory dictates that their polynomial or spectral coefficients must decay rapidly.*

Specifically, for a function of smoothness class C^r*, the singular values* σ_k *of the weight matrix representing it decay as:*

$$\sigma_k(W) \;\lesssim\; k^{-\frac{r}{d}}$$

where d *is the effective intrinsic dimension.*

Implication: *The heavy-tailed power-law decay observed in trained LLM weights (*$\alpha > 1$*) is not accidental. It is the spectral signature of the* smoothness *of the underlying semantic functions. Because* σ_k *decays rapidly, the truncation error*

$$\delta = \sum_{k>rank} \sigma_k$$

is negligible. This allows us to discard 90% of the parameters (the "tail") without losing the "head" that encodes the smooth compositional structure.

(c) ***Hierarchical Modularity.*** *The ability to prune entire attention heads without catastrophic failure suggests that the learned function is modular. The dense model learns a sum of modules; pruning removes the redundant modules, revealing the underlying compositional DAG.*

Consequently, the empirical success of post-training pruning provides strong evidence that neural networks learn functions in the compositionally sparse class $\mathcal{C}_{\text{comp}}$.

Part III
Learning and Evolution

CHAPTER 12

Implicit Regularization and Bits

We propose that the disparate forms of "inductive bias" in deep learning—from convolutional locality to low-rank attention—can be unified under a single theoretical currency: ***Bits****. By constraining the description length of the function class, architectural choices reduce metric entropy (covering numbers) and consequently Rademacher complexity. While optimization dynamics determine which specific solution is selected, the architecture defines the geometry of the solution space, imposing a hard ceiling on complexity that guarantees generalization.*

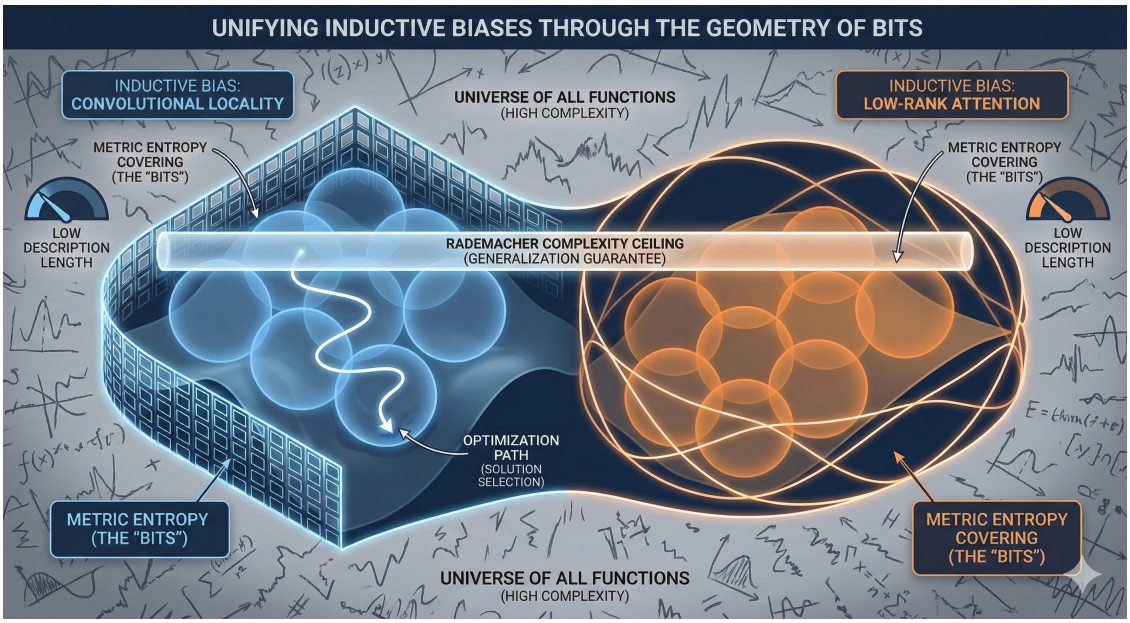


Figure 12.1: Unifying Inductive Biases through the Geometry of Bits. Different architectural choices, such as convolutional locality or low-rank attention, constrain the function space to different geometric regions. Despite their structural differences, their complexity can be measured by a unified currency: the number of bits (represented by covering spheres) needed to describe that region at a specific scale (metric entropy). By constraining the description length, architecture lowers metric entropy, placing a hard ceiling on Rademacher complexity and guaranteeing generalization, regardless of the specific path taken by optimization.

12.1 The Universal Currency: Bits, Geometry, and Noise

To understand how architecture aids learning, we must relate three distinct languages of complexity.

12.1.1 Kolmogorov Complexity (The Language of Bits)

For a function $f : \mathcal{X} \to \mathcal{Y}$, the Kolmogorov complexity $K(f)$ is the length of the shortest program that computes f on a fixed universal Turing machine:

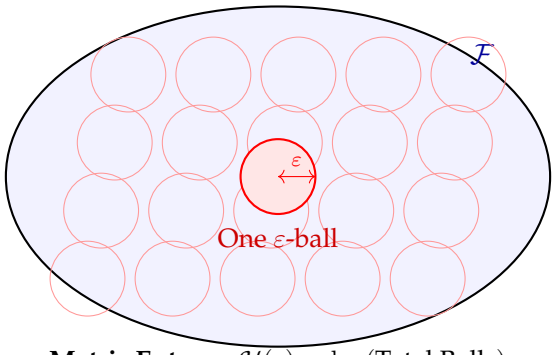
$$K(f) \;=\; \min_{p:\, U(p,\cdot)=f(\cdot)} |p|.$$

This is the ultimate lower bound on compressibility [99]. If a neural network architecture allows a function to be specified with very few parameters (e.g., a shared convolutional kernel), it forces $K(f_\theta)$ to be small. The Minimum Description Length (MDL) principle tells us that short programs generalize better because they cannot memorize random noise (which is incompressible).

12.1.2 Metric Entropy (The Language of Geometry)

How "large" is the space of functions $\mathcal{F}$ realizable by our network? We measure this not by volume, but by how many ε-balls are needed to cover it.

[Image of metric entropy covering number visualization]



Metric Entropy $\mathcal{H}(\varepsilon) \approx \log(\text{Total Balls})$
Counts the bits needed to specify a function $f \in \mathcal{F}$
to within precision ε.

Figure 12.2: **Visualizing Metric Entropy.** The function class $\mathcal{F}$ is covered by balls of radius ε. The "size" of the class is the log-count of these balls. Architectures with bounded fan-in or low rank shrink the volume of $\mathcal{F}$, requiring fewer balls to cover it, thus reducing metric entropy.

Let $\mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|)$ be the *covering number*: the fewest distinct functions needed to approximate any $f \in \mathcal{F}$ within error ε. The **Metric Entropy** is simply the bit-count of this cover:

$$\mathcal{H}(\varepsilon, \mathcal{F}) \;:=\; \log_2 \mathcal{N}(\varepsilon, \mathcal{F}, \|\cdot\|).$$

Intuitively, $\mathcal{H}(\varepsilon)$ is the number of bits required to describe a specific function in the class up to resolution ε.

12.1.3 Rademacher Complexity (The Language of Statistics)

Rademacher complexity $\widehat{\mathfrak{R}}_S(\mathcal{F})$ measures the ability of the function class to correlate with random noise signs $\sigma_i \in \{\pm 1\}$. If a class is too complex (too many bits), it can memorize any noise pattern, leading to overfitting [11].

[Image of Rademacher complexity random labels]

12.2 The Bridge: Dudley's Chaining Integral

How do we connect the geometry (Metric Entropy) to the statistics (Rademacher)? We use **Dudley's Chaining Integral**.

Imagine describing a function f by first giving a coarse approximation (using few bits), then a finer detail correction (more bits), and so on. The total complexity is the sum of these descriptions across all scales. Mathematically, this bounds the Rademacher complexity by the integral of the square root of the metric entropy [49]:

$$\widehat{\mathfrak{R}}_S(\mathcal{F}) \;\leq\; \frac{C}{\sqrt{n}} \int_0^{\operatorname{diam}(\mathcal{F})} \sqrt{\mathcal{H}(\varepsilon, \mathcal{F})}\, d\varepsilon. \tag{12.1}$$

The Takeaway: If architectural constraints suppress the metric entropy $\mathcal{H}(\varepsilon)$—even at very fine scales ε—the Rademacher complexity collapses, ensuring generalization.

[Image of Dudley's entropy integral]

12.3 Architecture as an Entropy Compressor

We now examine how specific architectural features act as "implicit regularizers" by compressing the metric entropy.

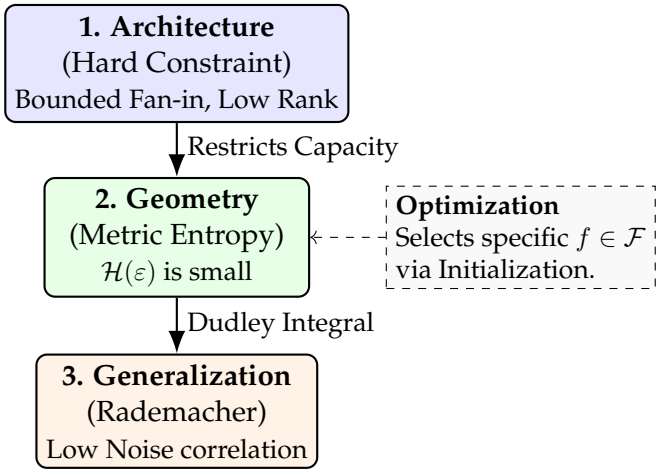


Figure 12.3: **The Regularization Funnel.** Architectural choices like locality or low rank act as a hard constraint on the geometry of the function space, lowering the metric entropy. This feeds into Dudley's integral to bound Rademacher complexity. Optimization dynamics play a role within this pre-defined geometry.

(A) Compositional Sparsity (Bounded Fan-in). Consider functions computable by a computation graph (DAG) with s nodes, where each node depends on at most k inputs (local connectivity). Even if the ambient dimension n is huge, the local complexity is low. For smooth node functions (Lipschitz bound B, smoothness r), the metric entropy scales as:

$$\mathcal{H}(\varepsilon, \mathcal{F}_{\text{CS}}) \lesssim s \cdot \left(\frac{B}{\varepsilon}\right)^{k/r}. \tag{12.2}$$

Crucially, this bound is **independent of the input dimension** n. The architecture forces the model to ignore the vast majority of the input space, focusing only on local interactions defined by k.

(B) Low-Rank Projections (Attention). In Transformers, attention matrices are often low-rank ($W = UV^\top$ where rank $\ll d$). A full rank matrix has d^2 degrees of freedom; a low-rank one has $2rd$. This linear reduction in parameters leads to a logarithmic reduction in metric entropy. The "bottleneck" rank r acts as a hard limit on the information capacity of each head, preventing the memorization of high-frequency noise [9].

(C) Quantization as Capacity Control. Using low-precision (e.g., 8-bit or 4-bit) weights is often viewed merely as a hardware optimization. Theoretically, it is a powerful regularizer. If weights are quantized to b bits, the total number of possible networks is finite (2^{bP}). The metric entropy at zero error becomes exactly bP bits. By denying the model infinite precision, we prevent it from encoding training data into the arbitrary decimal expansions of the weights [209].

12.4 Discussion: The Interplay of Architecture and Optimization

A nuanced view is required regarding the "cause" of generalization. Recent work by Xu et al. [204] and others suggests that for overparameterized networks, the final weights depend heavily on initialization and the specific trajectory of gradient descent. If the architecture is loose (large capacity), bad initialization can lead to "Kernel regime" solutions that generalize poorly, while "simple" initialization leads to min-norm solutions.

However, this does not negate the role of architecture. We propose the following synthesis:

- **Architecture defines the Playground (Geometry):** It sets the *potential* minimum description length. By restricting fan-in k or rank r, the architecture ensures that the manifold of possible solutions has low metric entropy.
- **Optimization selects the Position (Kinetics):** Within this favorable playground, implicit bias from initialization and SGD directs the network toward the simplest solution compatible with the data.

Thus, architecture is the *enabling constraint*: it ensures that a low-complexity solution exists and is accessible, effectively lowering the ceiling of the Dudley integral.

CHAPTER 13

Multiplicative Regularization Generalizes Better (based on work with R. Dubach and M. Abdallah)

We investigate the effectiveness of multiplicative versus additive (L2) regularization in deep neural networks, focusing on convolutional neural networks for classification. While additive methods constrain the sum of squared weights, multiplicative regularization directly penalizes the product of layerwise Frobenius norms, a quantity theoretically linked to tighter Rademacher-based generalization bounds. Through experiments on binary classification tasks in a controlled setup, we observe that multiplicative regularization consistently yields wider margin distributions, stronger rank suppression in deeper layers, and improved robustness to label noise. Under 20% *label corruption, multiplicative regularization preserves margins that are* 5.2% *higher and achieves* 3.59% *higher accuracy compared to additive regularization in our main network architecture. Furthermore, multiplicative regularization achieves a* 3.53% *boost in test performance for multiclass classification compared to additive regularization. Our analysis of training dynamics shows that directly constraining the global product of norms leads to flatter loss landscapes that correlate with greater resilience to overfitting. These findings highlight the practical benefits of multiplicative penalties for improving generalization and stability in deep models.*

13.1 Introduction

Regularization is central to controlling overfitting and improving generalization in deep learning, as the capacity of neural networks is inherently tied to their generalization abilities. This fact motivated extensive work to define generalization bounds reflective of neural network capacity and their ability to generalize beyond training data [58].

Among standard techniques, additive approaches such as L2 regularization remain dominant, yet there is increasing interest in methods that constrain global weight configurations more directly. Recent theoretical work suggests that products of layer norms play a key role in capacity control, motivated by norm-based generalization bounds linking tighter Rademacher complexity guarantees to the product of weight norms across layers. Moreover, Bottman et al. (2023) provide geometric evidence that multiplicative penalties produce more isolated, well-conditioned minima in the loss landscape, which helps explain their superior robustness [21]. These bounds suggest that a direct penalty on the product of the layer norms should offer distinct advantages over conventional per-

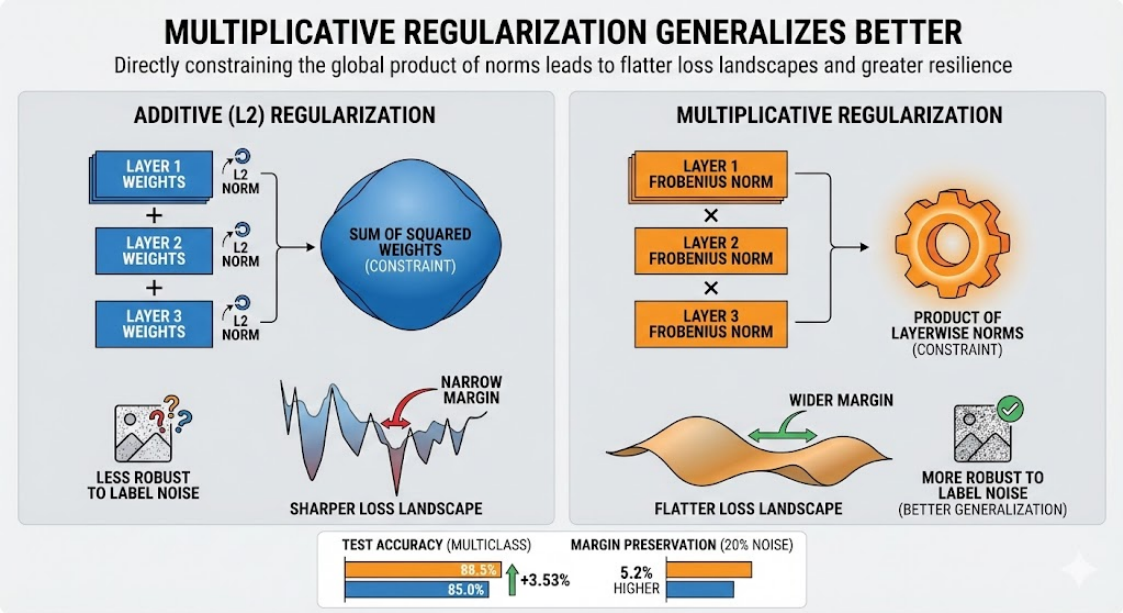


Figure 13.1

layer or summation-based penalties, which are the typical capacity-control methods employed in practice [134]. Our work compares these additive and multiplicative approaches in a systematically controlled setting to evaluate how each method affects training dynamics, margin behavior, and robustness to label noise.

[Image of additive vs multiplicative regularization constraints geometry]

We focus on the conceptual difference that L2 regularization penalizes the sum of squared parameters. In contrast, multiplicative penalties establish a coordinated constraint system across layers by penalizing the product of their Frobenius norms. We aim to determine whether explicitly targeting the product of norms elicits stronger generalization, better noise robustness, and more stable training. Through extensive experiments on binary classification tasks using convolutional neural networks, we find that multiplicative regularization promotes wider margins, sharper rank suppression, and increased resilience to label corruption. We also confirm similar trends in multiclass classification. Our results align with the theoretical motivation that bounding the product of layer norms provides a more principled approach to capacity control [3, 63].

13.2 Background

Researchers have long studied how large models generalize effectively despite being overparameterized [207]. L2 regularization has proven valuable for constraining weight magnitudes and improving performance, but its effect on the product of norms remains unclear [80, 102]. For multilayer networks, Rademacher complexity bounds depend on the product of layerwise spectral or Frobenius norms [136, 135] and can therefore be controlled by constraining this product. Earlier bounds displayed an exponential dependence on depth, but more recent results reduce this to polynomial dependence [62]. A representative bound has the form

$$\mathfrak{R}_m(\mathcal{F}) = \mathcal{O}\left(\frac{B^L}{m}\prod_{i=1}^{L}\|W_i\|_F^2\right), \tag{13.1}$$

where L is the number of layers, $\|W_i\|_F$ denotes the Frobenius norm of the i-th layer weight matrix, B bounds the input norm, and m is the sample size.

[Image of Rademacher complexity bound visualization]

From the perspective of such bounds, multiplicative regularization, which directly targets the product of norms, provides a more theoretically grounded capacity-control mechanism than methods constraining only individual weights or their sum [136]. Empirical comparisons have often focused on variations in weight decay and related additive techniques [80, 102]. Ensemble- and dropout-based methods inject noise to reduce reliance on any single set of parameters but do not systematically target the product of norms [180]. Other developments introduce multiplicative norm constraints or reparameterizations that keep layer products bounded [87, 169]. Bottman et al. analyze how different regularizers shape loss-geometry, arguing that multiplicative regularization induces degenerate minima while (conjecturally) additive regularization yields Morse-type isolated minima for generic nonlinear networks [21]. However, few empirical studies directly assess whether a multiplicative penalty outperforms standard L2 penalties; our work fills this gap in a controlled classification setting.

13.3 Theory

We relate the quantities $\prod_{i=1}^{L} \|W_i\|_F^2$ and $\sum_{i=1}^{L} \|W_i\|_F^2$. For nonnegative $\|W_i\|_F^2$, the AM–GM inequality gives

$$\frac{1}{L}\sum_{i=1}^{L} \|W_i\|_F^2 \;\geq\; \left(\prod_{i=1}^{L} \|W_i\|_F^2\right)^{1/L}, \tag{13.2}$$

with equality iff $\|W_1\|_F^2 = \cdots = \|W_L\|_F^2$. Raising both sides of (13.2) to the power L yields

$$\left(\frac{1}{L}\sum_{i=1}^{L} \|W_i\|_F^2\right)^{L} \;\geq\; \prod_{i=1}^{L} \|W_i\|_F^2. \tag{13.3}$$

Thus, in general, a small additive term may imply a small multiplicative term (up to a power and constant), but the converse need not hold. Hence minima of additive and multiplicative regularizers need not coincide. A useful identity is

$$\log\left(\prod_{i=1}^{L} \|W_i\|_F^2\right) = \sum_{i=1}^{L} \log\Big(\|W_i\|_F^2\Big), \tag{13.4}$$

so minimizing the product is equivalent (by monotonicity of $\log$) to minimizing the sum of the logs of squared norms, rather than the sum of the squared norms themselves.

13.4 Methodology

We compare two penalties added to the original loss L_{orig} with coefficient $\varepsilon > 0$:

$$L_{\text{add}} = L_{\text{orig}} + \varepsilon \sum_{i=1}^{L} \|W_i\|_F^2, \tag{13.5}$$

$$L_{\text{mult}} = L_{\text{orig}} + \varepsilon \prod_{i=1}^{L} \|W_i\|_F^2. \tag{13.6}$$

Additive regularization applies independent pressure to each layer, whereas multiplicative regularization couples the layers globally by directly targeting the product that appears in generalization bounds.

Architectures and data. For the binary experiments, we use a CNN with four convolutional layers of channels $(16, 32, 64, 128)$ with ReLU activations and 2×2 average pooling (stride 2) after each conv block, followed by two fully connected layers (256 hidden units, then one output). Convolutional layers are bias-free; biases are enabled in the fully connected layers. The total number of trainable parameters is $228{,}785$. We train on a binary CIFAR-10 task mapping "cat" (class 3) to -1 and "deer" (class 4) to $+1$. Inputs are normalized channel-wise to mean $(0.4914, 0.4822, 0.4465)$ and standard deviation $(0.2470, 0.2435, 0.2616)$. We use $10{,}000$ training and $2{,}000$ test images. Training uses MSE regression with prediction $\operatorname{sign}(f(x))$.

For multiclass experiments, we train on full CIFAR-10 (10 classes) with one-hot targets and MSE loss; prediction is by $\arg\max$ over logits. The CNN has four convolutional blocks, each with two 3×3 conv layers, ReLU, and 2×2 average pooling. Channels per block are $(64, 128, 256, 512)$. The final $2 \times 2 \times 512$ map is flattened, then a fully connected layer with 512 ReLU units precedes a 10-way linear output. Total parameters are $5{,}737{,}152$. No biases are used. All weights use Kaiming (He) initialization [74].

Training protocol. Weights are initialized with He normal initialization; a global scaling $s \in \{0.2, 0.4, 0.6, 0.8\}$ is applied to all layers to study low-norm starts. We use SGD with batch size 16 for up to 2000 epochs, with early stopping if test accuracy does not improve for 200 epochs. For both penalties we sweep $\varepsilon \in \{5 \times 10^{-1}, 5 \times 10^{-3}, \ldots, 5 \times 10^{-15}\}$ in the binary setup (and to 5×10^{-20} in the multiclass setup). Each configuration is run with seeds 1–5 (binary) or 10 seeds (multiclass), discarding runs that fail to train (random-guessing).

Metrics. Besides accuracy and test loss, we compute margins for correctly classified points: for sample (x_i, y_i) with $y_i \in \{-1, +1\}$,

$$\text{margin}_i \;=\; y_i \, f(x_i), \tag{13.7}$$

where $f(x)$ is the scalar network output before the sign. For fair comparison, after training we rescale weights to unit global norm before margin computation. To estimate effective dimensionality, we compute layer ranks via SVD of each weight matrix (convolutional kernels reshaped to matrices). Singular values are normalized by the largest; the rank is the count exceeding 1% of the maximum [203]. We also study robustness by flipping a fraction of training labels at random.

13.5 Results

All results reported here correspond to the best ε per method chosen by averaging test accuracy over seeds at the fixed initialization scale $s = 0.2$. Both strategies can reach near-perfect training accuracy, indicating minimal underfitting, but multiplicative regularization consistently yields lower test loss and higher test accuracy. The average test-accuracy gap reaches up to 9.4 percentage points in the binary task. Mean test margins are systematically larger under multiplicative penalties.

Noise robustness (binary). With clean labels (0% noise), multiplicative regularization attains 89.50% test accuracy versus 87.87% for additive and 87.80% without regularization. With 10%

Table 13.1: By-layer rank and Frobenius norms for the best-performing binary configuration (averaged over seeds).

	Rank			Frobenius norm $\|W_i\|_F$		
Layer	Mult.	Add.	Unreg.	Mult.	Add.	Unreg.
conv1	16	16	16	7.39	4.55	5.71
conv2	32	32	32	16.84	5.14	6.78
conv3	64	64	64	10.74	6.39	8.47
conv4	47	128	128	1.29	6.18	9.04
fc1	2	256	256	0.46	5.19	10.25
fc2	1	1	1	0.38	3.57	3.10
$\sum_i \|W_i\|_F^2$	456 (Mult.)		166 (Add.) 347 (Unreg.)			
$\prod_i \|W_i\|_F$	3.01×10^2 (Mult.)		1.71×10^4 (Add.) 9.42×10^4 (Unreg.)			

Table 13.2: Multiclass CIFAR-10: norms and accuracies averaged over 10 seeds at best ε.

Regularization	$\prod_i \|W_i\|_F$	$\sum_i \|W_i\|_F^2$	Train Acc.	Test Acc.
Multiplicative	3.73×10^7	679	0.891	0.687
Additive	1.10×10^9	875	0.925	0.651
Unregularized	1.67×10^9	960	0.937	0.650

label flips, multiplicative maintains 86.68% versus 81.13% (additive) and 80.20% (none). With 20% corruption, multiplicative reaches 83.10%, surpassing additive (73.69%) by 9.41 points and unregularized (73.08%) by 10.02 points. Margins mirror these trends: under clean data, the mean margin is 0.8329 (mult.) versus 0.7896 (add.) and 0.7890 (none). With 20% noise, the margin is 0.7109 (a 14.6% reduction) for multiplicative, versus 0.5124 (35.1% reduction) for additive and 0.5066 (35.8% reduction) for none.

Training dynamics and norm structure. Multiplicative and additive penalties converge to different minima of the unregularized landscape. Multiplicative penalties drive the product $\prod_i \|W_i\|_F$ over an order of magnitude lower than additive penalties, which in turn reduce it by about an order relative to the unregularized case. In contrast, $\sum_i \|W_i\|_F^2$ decreases most under additive penalties, whereas under multiplicative penalties it can increase while $\prod_i \|W_i\|_F$ remains low. Later layers show pronounced rank suppression under multiplicative regularization, especially the first fully connected layer [59].

Multiclass results. On full CIFAR-10 with clean labels, multiplicative regularization again yields the best test accuracy and smaller generalization gap. Table 13.2 summarizes averages over 10 seeds at the best ε per method. Additive regularization offered little benefit over the unregularized case in our sweeps, whereas multiplicative delivered a 3%+ test-accuracy boost and much smaller norm products.

13.6 Discussion

Our findings confirm that multiplicative regularization can offer practical benefits over additive penalties in controlling capacity and improving resilience to noise. Bounding the product of norms is intimately tied to margin-based and Rademacher complexity bounds. Empirically, multiplicative penalties lead to distinct training dynamics and converge to minima different from additive regularization. In convolutional networks trained on CIFAR-10, multiplicative constraints guide optimization toward flatter minima with simplified high-level representations: higher margins and pronounced rank suppression, especially in later layers [109, 28, 95]. The norm distributions reflect global coupling across layers, rather than uniform pressure on each layer as in additive penalties. Initialization scale matters: smaller initial weights let multiplicative constraints act early, preventing norm growth; with larger scales, differences narrow.

13.7 Conclusion

Directly penalizing the product of layerwise norms tends to yield higher test accuracy, improved margins, better noise robustness, and stronger rank suppression than additive L2 penalties in our controlled CNN studies. These observations align with theoretical arguments that product-based constraints provide a principled mechanism for capacity control in overparameterized networks, with especially clear advantages under label noise. Future work may explore hybrid strategies, other architectures (e.g., Transformers), and optimizers that further amplify the benefits of multiplicative regularization.

CHAPTER 14

For Overparametrized Networks Probability Concentrates Around Global Minima

In the regime of over-parameterized neural networks, the landscape of the training loss is complex and generally non-convex. Despite this, optimization algorithms such as Stochastic Gradient Descent (SGD) often converge to solutions that generalize well. This chapter explores the theoretical underpinnings of this phenomenon, focusing on the interplay between the geometry of the loss landscape and the stochastic dynamics of the optimizer.

The main rigorous conclusions presented here are three-fold:

1. ***Geometric Degeneracy:*** *In over-parameterized networks, the set of parameters achieving zero training loss generically is a high-dimensional manifolds. These "flat" or degenerate minima are characterized by Hessians with large nullspaces.*

2. ***Langevin Concentration:*** *Injecting Gaussian noise into gradient descent (resulting in SGDL) effectively models the optimization process as Langevin dynamics. The resulting stationary distribution is a Boltzmann distribution $p(\theta) \propto e^{-L(\theta)/T}$. As the "temperature" T decreases, probability mass concentrates in regions of larger local volume—favoring flat minima over sharp ones.*

3. ***Exploration-Exploitation Trade-off:*** *The balance between exploration and exploitation is governed by the ratio of the learning rate to the batch size. This ratio defines distinct timescales for drift and diffusion, establishing an effective temperature that promotes the exploration of wide valleys in the loss landscape.*

14.1 Main Results

We formalize these insights through four key lemmas that connect algebraic geometry, stochastic differential equations, and measure concentration. The key intuition is that global minima are very degenerate in overparametrized situations. Since diffusion-like processes converge to a Boltzman distribution where the greater probability mass will concentrate on degenerate manifolds we have convergence to global minima with high probability not because they are minima but because they are the most degenerate. We know that the assumption of Gaussian noise, SGDL and continous

diffusion are not strictly correct for practical cases in which SGD is used. We believe however that the qualitative phenomena we describe are very likely to hold in general – even if the precise dynamic is different because the "noise" associated with SGD is not additive Gaussian noise and the dynamics is discrete.

14.1.1 The Geometric Lemma: Existence of Flat Minimizers

Let P be the number of trainable parameters and M be the number of scalar constraints induced by the training data. Under polynomial or "generic" conditions (or using piecewise-polynomial surrogates for activation functions like ReLU), the set of zero-loss solutions behaves as an algebraic variety [197].

Lemma 6 (Geometric Degeneracy)**.** *The zero-loss variety has real dimension $d \geq P - M$. Consequently, at any parameter configuration θ^* such that the loss $L(\theta^*) = 0$, the Hessian $\nabla^2 L(\theta^*)$ possesses at least $P - M$ zero eigenvalues.*

This result – related to Bezout theorem – guarantees the existence of high-dimensional manifolds of global minima, providing the geometric basis for the "flatness" observed in deep learning solutions [37].

14.1.2 The Dynamics Lemma: SGDL as Langevin Diffusion

We analyze the update rule for Stochastic Gradient Descent with explicit Langevin noise (SGDL):

$$\theta_{t+1} = \theta_t - \gamma \nabla L(\theta_t) + \sqrt{2\gamma T}\zeta_t, \tag{14.1}$$

where ζ_t is i.i.d. Gaussian noise.

Lemma 7 (Langevin Correspondence)**.** *The discrete SGDL update approximates (under some strong conditions!) the Euler-Maruyama discretization of the continuous Langevin stochastic differential equation:*

$$d\theta_t = -\nabla L(\theta_t)dt + \sqrt{2T}dB_t. \tag{14.2}$$

The stationary distribution of this process is given by the Gibbs-Boltzmann density $p(\theta) \propto e^{-L(\theta)/T}$ [198].

14.1.3 The Concentration Lemma: Flat Beats Sharp

While all global minima have (by definition) equal training loss, they are distinguished by the volume of their surrounding basins.

Lemma 8 (Volume Concentration)**.** *For minima with equal loss, neighborhoods with smaller curvature (flatter geometry) occupy a larger local volume. In the context of Singular Learning Theory, this corresponds to a smaller Real Log Canonical Threshold (RLCT). Asymptotically, the stationary probability mass captures more weight in these flatter regions. Thus, lowering the temperature T or increasing the dimensionality of the parameter space favors degenerate, flat minima over sharp, isolated ones [79].*

14.1.4 The Timescale Law

The effective thermodynamics of SGD are set by the hyperparameters.

Lemma 9 (Drift and Diffusion Timescales)**.** *The characteristic timescales for drift (τ_{drift}) and diffusion (τ_{diff}) scale as:*

$$\tau_{drift} \sim \frac{n}{\gamma \|\nabla L\|}, \quad \tau_{diff} \sim \frac{\sqrt{n}}{\gamma \operatorname{tr}(\sqrt{\Sigma})}, \tag{14.3}$$

where n is the batch size, γ is the learning rate, and Σ is the noise covariance.

This relationship illuminates how the learning rate and batch size jointly determine the effective temperature, regulating the optimizer's ability to escape sharp basins and explore wide valleys [121].

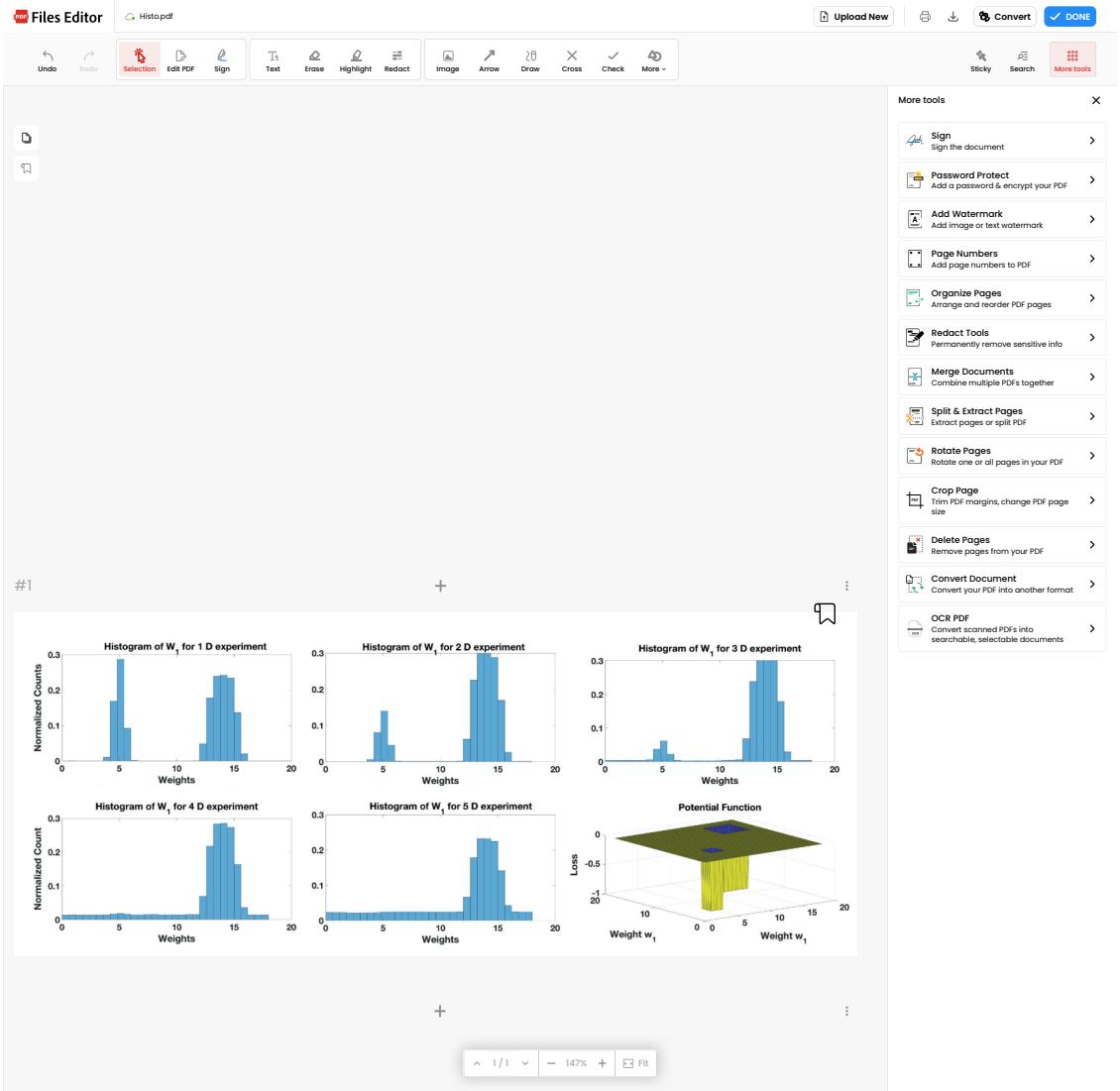


Figure 14.1

14.2 Detailed Analysis

14.2.1 Synthesis of Geometric and Dynamic Views

The convergence of over-parameterized networks to generalizable solutions can be understood as a synthesis of geometry and probability.

Geometry of Flat Minima. With a square loss function on over-parameterized ReLU networks, the zero-loss set $L^{-1}(0)$ is generically a smooth submanifold of codimension equal to the number of constraints – that is the number of training data. This implies that the Hessian at any global minimizer must have a large nullspace of dimensionality in the order of $W - N$ where W is the number of weights and N is the number of training data. Parallel arguments from algebraic geometry, which approximate ReLUs with polynomials, use parameter counting (Bézout-style reasoning) to demonstrate the existence of continuous families of zero-loss solutions when the number of parameters P exceeds the number of constraints N. The Implicit Function Theorem then formally establishes the "flatness" of these solution manifolds.

Stochastic Dynamics and Stationary Measures. By viewing SGDL as a discretization of Langevin dynamics, we gain access to the stationary measure $p(\theta) \propto e^{-L(\theta)/T}$. A Taylor expansion near a global minimum, combined with a small temperature T, implies that the Boltzmann mass concentrates in balls around global minimizers. This provides a principled metric for comparing minima: those with flatter geometries (smaller curvature) are entropically favored [153].

Why Flat Minima Dominate. In this framework, flatness is a proxy for probability mass. For two minima with equal depth (loss value), the one with the flatter Hessian has a neighborhood with exponentially larger volume. Singular Learning Theory reinforces this view: a smaller RLCT implies a larger neighborhood volume exponent. Consequently, the stochastic dynamics spend significantly more time in these degenerate regions. Simulations of SGDL confirm this measure concentration, showing that the system preferentially settles on high-dimensional zero-loss manifolds.

Effective Thermodynamics. The derivation of drift and diffusion timescales suggests that the ratio of learning rate to batch size acts as a control parameter for the "effective temperature" of the training process. When diffusion is fast relative to drift, the dynamics are locally equilibrated, promoting the thorough exploration of wide valleys.

14.2.2 Critical Assessment

While the theoretical picture is compelling, several assumptions require careful scrutiny.

The "Polynomial in Parameters" Premise. Neural networks with ReLU activations are piecewise polynomial; they are multi-affine in layer parameters for fixed activation patterns. Treating them as globally polynomial is an approximation useful for algebraic geometry arguments. However, replacing ReLUs with polynomial activations makes these dimension arguments mathematically rigorous. Thus, while the existence of flat directions is credible, strict polynomiality for standard ReLU networks is an approximation.

Genericity and Rank Conditions. The lower bound $\dim \geq P - M$ relies on the generic independence of the constraint equations. Invoking the Implicit Function Theorem requires a full-rank Jacobian, and ensuring a smooth manifold structure on the real solution set requires care. These conditions are plausible and consistent with the literature but represent technical subtleties in the rigorous proof.

SGD vs. Langevin Dynamics. The assumption that gradient noise is Gaussian is asymptotic in the batch size. Practical training often uses small batches, resulting in heavy-tailed, non-Gaussian noise. Furthermore, momentum and learning-rate schedules deviate from the simple

Langevin diffusion model. The phenomenology—that SGD behaves *as if* sampling at an effective temperature—is a powerful heuristic but not an exact equivalence [121].

Equal-Loss Assumption. The comparison of minima assumes equal training loss (often zero). In practice, finite temperature, non-zero training loss, and strong non-quadraticity complicate the picture. Nevertheless, the directional prediction remains robust: SGD-like noise disfavors sharp basins in favor of flatter ones.

14.2.3 Summary

The core picture emerging from this analysis is coherent:

- **Over-parameterization** leads to an abundance of degenerate solution sets, as predicted by geometric dimension arguments.
- **Stochastic dynamics** at low temperature lead to stationary measures that weight solutions by their local volume, creating a strong preference for flat minima.
- **Timescale analysis** explains the practical role of batch size and learning rate in defining an effective temperature that enables the exploration of these wide valleys.

CHAPTER 15

A Self-Assembling Cortical Circuit for Generalized Gradient Descent (with Qianli Liao and Liu Ziyin)

How does the brain implement supervised learning? Backpropagation is biologically implausible but are there neural circuits that could be used for optimization? Do they represent a "motif" that we could look for in different brain areas? This chapter describes the proposal for such a circuit.

Overview

One of the long–standing mysteries in neuroscience is how cortex could implement a general-purpose learning rule comparable to stochastic gradient descent (SGD). Anatomically, the cortex is equipped with extensive ascending (feedforward) and descending (feedback) pathways. Physiologically, it exhibits a rich variety of synaptic plasticity mechanisms, including homosynaptic and heterosynaptic updates. Yet the classical backpropagation algorithm requires both the transport of precise synaptic weights and a global organization of error signals—assumptions that are difficult to reconcile with biology [39, 19].

This chapter describes a simple and biologically plausible cortical circuit, based on recent work by Liao, Ziyin, Gan, Cheung, Harnett and Poggio [110], that *self-assembles* from random initial connectivity and converges to an effective implementation of SGD. The circuit uses only local, heterosynaptic plasticity and requires no weight symmetry, no global information, and no special developmental fine–tuning. Remarkably, once assembled, the circuit performs comparably to SGD on standard machine learning tasks.

The mechanism illustrates a broader principle of this book:

> Sparse, modular motifs can support general learning by composing locally learned transformations.

In this sense, the SAL circuit is the synaptic analogue of the compositional architectures discussed in earlier chapters: sparse constituent operations arranged into a minimal motif that can scale to arbitrarily complex tasks.

The chapter proceeds as follows. Section 15.1 introduces the cortical synaptic motif. Section 15.2

describes the Self-Assembling Learning (SAL) algorithm. Section 15.3 states the main theoretical result: under mild conditions, the dynamics converge to a matrix–preconditioned form of SGD. Section 15.4 outlines the biological predictions. Section 15.6 discusses implications for machine learning and biological intelligence.

15.1 A Minimal Synaptic Motif for Cortical Learning

The circuit (illustrated in Figure 15.1) consists of two interacting pathways:

- an *ascending* (feedforward) stream with weights W_ℓ;
- a *descending* (feedback) stream with weights $\bar{W}_\ell$.

Between these streams are two cross–connections:

$$V_\ell : \text{downstream} \to \text{upstream}, \qquad \bar{V}_\ell : \text{upstream} \to \text{downstream}.$$

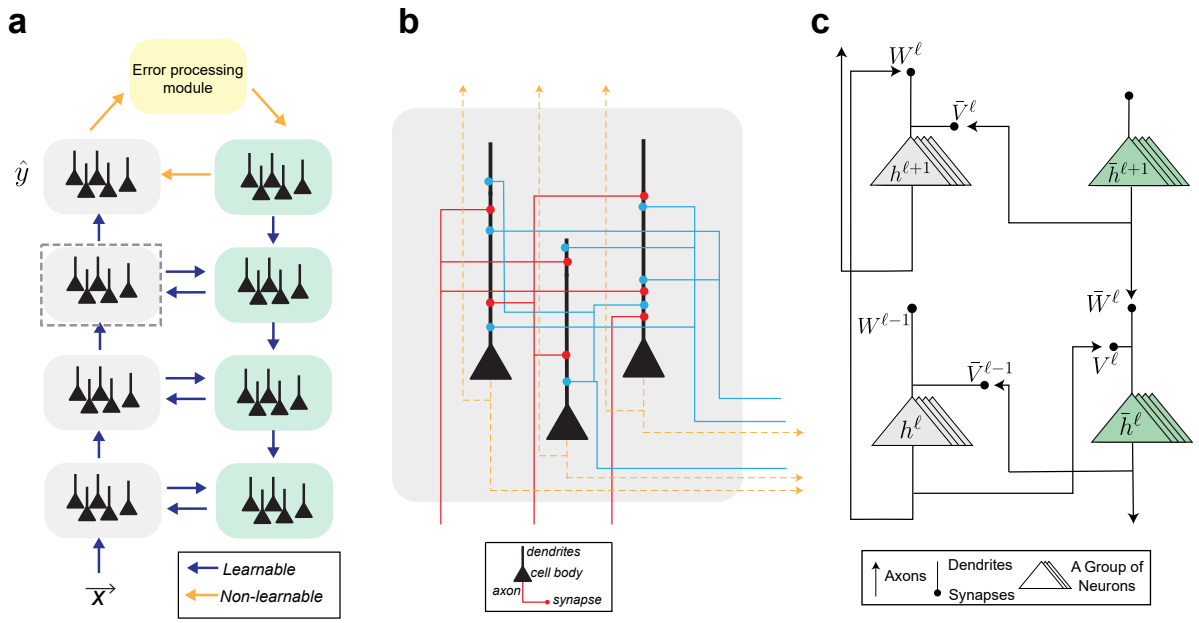


Figure 15.1: **A synaptic motif for interactions between ascending and descending information streams.** **(a)** Schematic of the upstream-downstream architecture. The upstream consists of a standard fully-connected neural network with multiple layers, roughly corresponding to a multi-region cortical processing pathway (e.g. V1-V2-V4-IT). The output of the upstream network goes to an error processing module (potentially corresponding to frontal cortices). This module computes a local error signal. This error signal is sent to the feedback (downstream) pathway, which processes information layer by layer downwards. The orange arrows represent non-learnable (identity) connections. Blue arrows represent learnable connections, each parameterized by a fully-connected weight matrix. **(b)** A detailed view of the smallest unit of the connection motif in panel a, indicated by grey dashed box. The neurons in panel b (as well as the abstract forms in a and c are illustrated as pyramidal neurons, the principal cell type of the cortex. Only apical dendrites are illustrated here for simplicity. **(c)** A mathematical description of the unit in panel b. Each arrow (which corresponds to a collection of axons, dendrites, and synapses) represents a set of full connections between two groups of neurons, parameterized by a weight matrix W or V.

Thus each layer–to–layer interface contains four synaptic matrices,

$$(W_\ell,\ \bar{W}_\ell,\ V_\ell,\ \bar{V}_\ell),$$

all of which are initialized randomly and all of which are plastic.

Anatomically, this motif corresponds to reciprocal cortico–cortical projections and their local dendritic interactions. Biophysically, the circuit's learning rule relies on experimentally observed heterosynaptic plasticity, which allows changes in a synapse to depend on activity at nearby synapses without violating locality [32].

Feedforward computation. With activations $h_\ell \in \mathbb{R}^{d_\ell}$, the upstream pathway computes:

$$h_{\ell+1} = D^u_\ell(W_\ell h_\ell), \qquad \ell = 1, \dots, L-1, \tag{15.1}$$

where D^u_ℓ is typically the Jacobian of a ReLU nonlinearity.

Error injection. Given a loss $F(h_L, y)$, the downstream pathway begins with

$$\bar{h}_L = -\nabla_{h_L} F.$$

Feedback computation. The descending pathway propagates the error estimate locally:

$$\bar{h}_\ell = D^d_{\ell+1}(\bar{W}_\ell \bar{h}_{\ell+1}), \qquad \ell = L-1, \dots, 1, \tag{15.2}$$

where the matrices D^d_ℓ may be linear or nonlinear.

The key point is that the descending pathway is *not* assumed to mirror the ascending one. It starts random, remains asymmetric, and is allowed to be overparameterized ($\bar{d}_\ell \gg d_\ell$), matching cortical anatomy where feedback projections often outnumber feedforward projections.

15.2 The Self-Assembling Learning Rule

SAL updates all four sets of synapses using local heterosynaptic products of pre– and post–synaptic activities. For each layer ℓ:

$$\Delta W_\ell = \eta\, D^u_{\ell+1}\, \bar{V}_\ell\, \bar{h}_{\ell+1}\, h_\ell^\top\, D^u_\ell - \gamma W_\ell, \tag{15.3}$$

$$\Delta \bar{W}_\ell = \eta\, D^d_{\ell+1}\, \bar{h}_{\ell+1}\, h_\ell^\top\, V_\ell^\top\, D^d_\ell - \gamma \bar{W}_\ell, \tag{15.4}$$

$$\Delta V_\ell = \eta\, \bar{h}_\ell\, h_\ell^\top - \gamma V_\ell, \tag{15.5}$$

$$\Delta \bar{V}_\ell = \eta\, \bar{h}_{\ell+1}\, h_{\ell+1}^\top - \gamma \bar{V}_\ell. \tag{15.6}$$

Each update depends only on:

$$(\text{local activity}) \times (\text{neighboring synaptic input}),$$

which is permitted by known forms of heterosynaptic plasticity, including calcium-mediated interactions and dendritic-level synaptic "tagging."

what kind of figure is it? [Image of heterosynaptic plasticity synaptic update rule]

The circuit has no special wiring, no weight-symmetry constraints, and no assumptions about initial structure. Instead, the structure *emerges* during learning.

15.3 Theoretical Result: Emergence of Gradient Descent

The central theorem states that under mild conditions, the SAL updates for the feedforward weights W_ℓ converge to a matrix–preconditioned form of SGD.

Theorem 11 (SAL implements SGD up to a learned preconditioner)**.** *Consider a ReLU network with arbitrary width and depth. Assume the descending pathway is overparameterized ($\bar{d}_\ell \geq d_\ell$) and that V_ℓ and $\bar{V}_\ell$ remain full rank during training. If their updates become small, $\Delta V_\ell = O(\varepsilon)$ and $\Delta \bar{V}_\ell = O(\varepsilon)$, then*

$$\Delta W_\ell = H_\ell \, \Delta W_\ell^{\mathrm{SGD}} - \gamma W_\ell + O(\varepsilon), \qquad H_\ell = \bar{V}_\ell \bar{V}_\ell^\top \tag{15.7}$$

where $\Delta W_\ell^{\mathrm{SGD}} = -\nabla_{W_\ell} F$.

Thus the circuit performs gradient descent with a *learned, positive-definite matrix preconditioner* H_ℓ. In this sense, the SAL circuit is a *gradient machine* whose learning rule itself is learned.

Several consequences follow:

- The stationary points of SAL coincide with those of SGD.
- The descending pathway effectively learns to approximate the transpose Jacobian of the feedforward network—emergent weight alignment without symmetry constraints, similar to Feedback Alignment mechanisms [111].
- Overparameterized feedback pathways improve performance, as observed in cortical anatomy and in experiments.

15.4 Biological Interpretation and Predictions

The SAL circuit makes several explicit, testable predictions about cortical microcircuitry:

(1) A four–synapse reciprocal motif. Between any two layers (or cortical regions), there should exist plastic synapses forming the motif $(W,\ V,\ \bar{W},\ \bar{V})$.

(2) Pervasive heterosynaptic plasticity. Synaptic changes should depend not only on pre– and post–synaptic firing, but also on neighboring synaptic activity—consistent with dendritic calcium spread, retrograde messengers, and synaptic tagging.

(3) Overabundance of feedback connections. SAL predicts that larger feedback pathways improve learning. Anatomically, cortico–cortical feedback projections often outnumber feedforward ones by factors of 5–10, in agreement with model behavior.

(4) Self-assembly from random connectivity. No precise hard-wired matching is needed. The algorithm predicts that feedback pathways become aligned with the forward dynamics during experience.

(5) Distinct electrophysiological properties. Feedback inputs should operate with different nonlinearities or time constants than feedforward inputs, matching the role of D_ℓ^d versus D_ℓ^u.

[Image of cortical column layer connectivity diagram]

15.5 An Example Implementation of SAL in Cortex

Self-Assembling Learning (SAL) is defined in terms of four abstract pathway types—upstream-to-upstream (u→u), upstream-to-downstream (u→d), downstream-to-downstream (d→d), and downstream-to-upstream (d→u). These pathway types are functional constructs rather than claims about specific neuron morphologies or laminar structures. The defining architectural principles of SAL are (i) the existence of two interacting streams, (ii) driving interactions within each stream and modulatory interactions between streams, and (iii) the absence of closed loops composed entirely of driving connections within either stream. SAL itself makes no assumptions about the biophysical mechanisms by which these properties are implemented. When viewed through this functional lens, cortical lamination and its canonical connectivity patterns provide a useful and concrete example of how the SAL architecture could be realized in a biological system.

In sensory cortex, many pathways that operate within the same functional stream—such as those carrying representational content—are strong and driving, whereas interactions between distinct streams often exert a modulatory influence on activity and learning. Importantly, the distinction emphasized by SAL is not between feedforward and feedback pathways per se, but between within-stream and cross-stream interactions. Under this interpretation, a number of well-established cortical pathways can be organized into the four SAL categories. Table 15.1 summarizes one such correspondence between canonical cortical pathways and SAL pathway types, together with a hypothesized relationship between dendritic targeting and the resulting driving or modulatory effects observed in cortex.

Interpretation and scope. The correspondence shown in Table 15.1 should be interpreted as an example of how SAL could be implemented in cortex, rather than as a claim that SAL must be implemented through cortical lamination or that cortical lamination evolved specifically to implement SAL. Cortical areas exhibit substantial variation in laminar structure, connectivity, and cellular properties, and exceptions to canonical patterns are well documented. In particular, some cortical neurons, such as large layer 5 pyramidal neurons, exhibit conditional or gated driving as a result of their dendritic biophysics. Such properties reflect details of a specific biological implementation and are not assumptions or requirements of the SAL framework itself. Other brain systems may satisfy the same SAL constraints using different cellular or circuit mechanisms.

From the perspective of SAL, the central claim is therefore functional rather than morphological: learning systems that avoid unstable dynamics and signal contamination should exhibit two interacting streams, driving connectivity within each stream, modulatory connectivity between streams, and no closed loops of purely driving interactions within a stream. Cortical lamination provides one biologically plausible realization of these constraints, but not their only possible instantiation.

15.6 Implications for Learning and Intelligence

The SAL algorithm has a few implications on important problems of learning and intelligence:

Bridging biological and artificial systems. Modern deep networks rely on global backpropagation, yet biological learning is entirely local. The SAL circuit offers a concrete reconciliation: local plasticity can approximate global gradient descent via a self-organizing feedback system.

Implications for future AI architectures. The SAL motif suggests architectures where:

Table 15.1: One possible correspondence between canonical cortical pathways and SAL pathway types. Dendritic targeting and observed influence type describe *biological properties of the cortical implementation* rather than assumptions of the SAL framework. "Conditionally driving" refers to a biophysical property observed in some layer 5 pyramidal neurons, arising from interactions between basal and apical compartments, and is *not* a requirement of SAL itself.

SAL pathway	Weight type	Cortical pathway example	Dendritic targeting	Observed influence type
u $\to$ u	W	L4 $\to$ L2/3 (within area)	Basal / proximal	Driving
		L2/3 $\to$ L4 (higher area)	Basal / proximal	Driving
u $\to$ d	V	L2/3 $\to$ L5 (within area)	Basal + apical	Conditionally driving
d $\to$ d	$\bar{W}$	L5 $\to$ (thalamus $\to$) L5 (inter-areal)	Basal + apical	Conditionally driving
		L6 $\to$ (thalamus $\to$) L6 (inter-areal)	Basal / proximal	Driving
d $\to$ u	$\bar{V}$	L5 $\to$ L1 $\to$ L2/3 (inter-areal)	Apical / distal	Modulatory
		L5 $\to$ L2/3 (within area)	Apical-biased	Modulatory

- feedback pathways learn separately from feedforward ones,
- the learning rule itself is learned (via H_ℓ),
- hierarchical motifs self-assemble through local interactions.

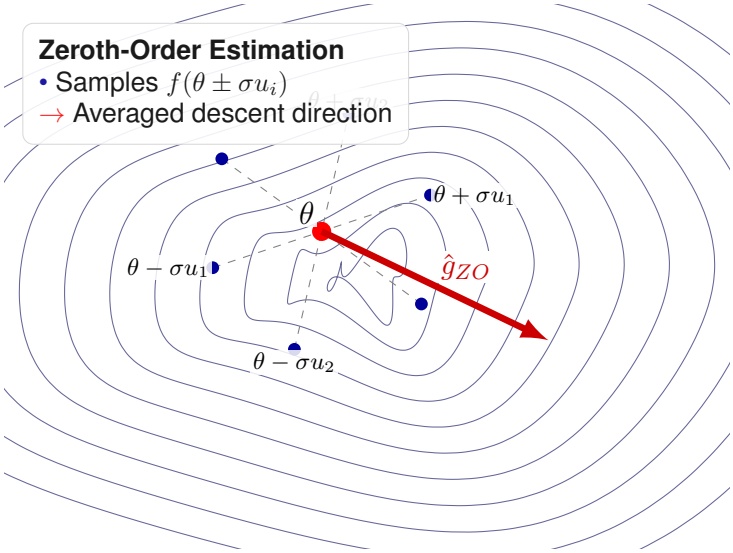
Such systems may inspire new forms of deep learning, moving beyond fixed optimizers toward neural architectures that *learn how to learn* through structured feedback.

In summary, the Self-Assembling Learning paradigm provides a biologically plausible mechanism by which cortex might implement gradient-based learning, demonstrating how simple motifs can self-assemble powerful, general-purpose learning systems.

CHAPTER 16

Zeroth-Order Evolutionary Post-Training for LLMs (with Y. Gan)

We explore when zeroth order optimization works, why it fails, and how to use it. Zeroth-order (ZO) evolutionary methods offer a natural fit for post-training large language models (LLMs) in settings where gradients are unavailable, unreliable, or undefined [133]. Such methods can directly optimize noisy or non-differentiable objectives, including human preference judgments and safety scores. While attractive in principle, their practicality hinges critically on the effective dimensionality of the search space.



16.1 Introduction

Recent advancements in the post-training of Large Language Models (LLMs) demonstrate that Reinforcement Learning (RL) significantly outperforms Supervised Fine-Tuning (SFT) in enhancing reasoning capabilities. RL encourages the model to explore and optimize its solution paths. An example is DeepSeek-R1 [67], which uses a RL algorithm *Group Relative Policy Optimization (GRPO)* to achieve substantial improvements in mathematical reasoning.

Despite these successes, a fundamental challenge remains. While first-order optimizers like Stochastic Gradient Descent (SGD) and Adam are the workhorses of pre-training, the post-training

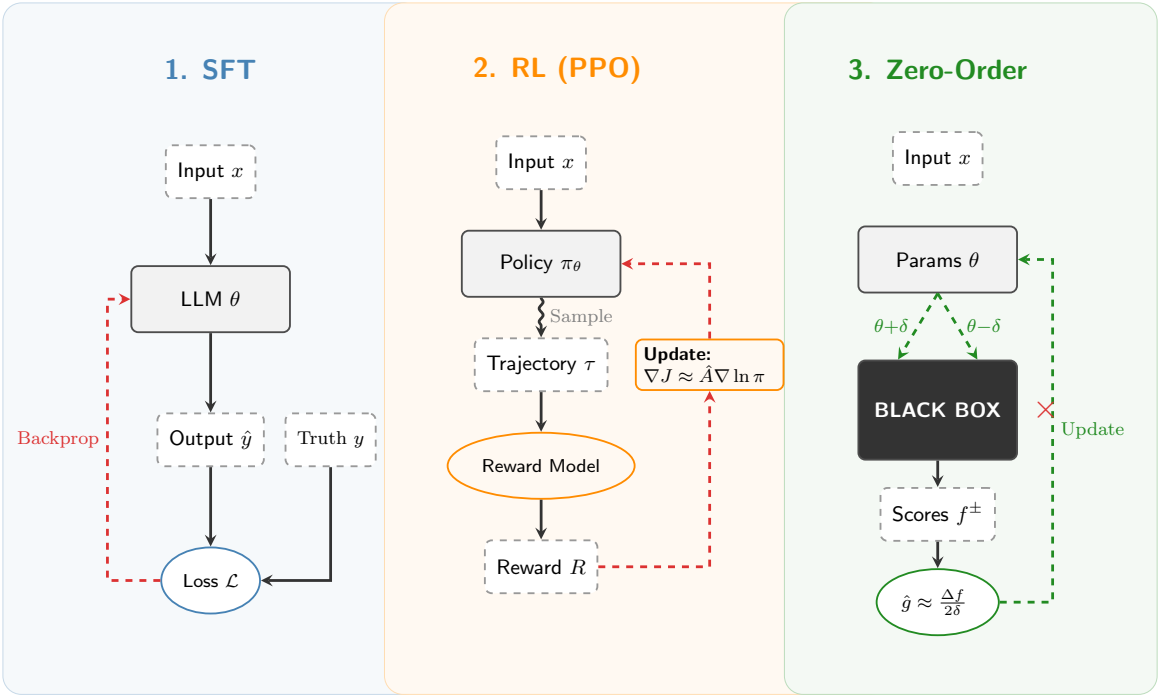


Figure 16.1: **Overview of LLM Training Paradigms.** The diagram contrasts three optimization frameworks: (1) **SFT**: Direct supervision where loss $\mathcal{L}$ is computed against ground truth y to update θ via backpropagation. (2) **RL (PPO)**: Policy optimization where a reward R for trajectory τ informs the gradient update $\nabla J \approx \hat{A} \nabla \ln \pi$, where $\hat{A}$ is the advantage function. (3) **Zero-Order Methods**: Gradient estimation $\hat{g} \approx \frac{\Delta f}{2\delta}$ using finite differences of scores $f^{\pm}$ obtained from parameter perturbations $\theta \pm \delta$, bypassing the need for internal gradient access in black-box environments.

phase is increasingly encountering regimes where gradient computation is problematic. In scenarios involving discrete rewards, black-box APIs, or memory-constrained environments, gradients often become computationally prohibitive, notoriously unstable, or mathematically undefined.

In response to these challenges, there has been a paradigm shift towards Zeroth-Order (ZO) optimization in the LLMs post-training. By estimating gradients through forward passes, ZO methods offer a viable pathway for optimizing LLMs in gradient-free or memory-constrained regimes.

16.1.1 Why ZO for LLMs?

In recent years, ZO optimization methods have emerged as a useful tool for solving challenges related to LLMs without using backpropagation. This renewed interest comes from two limitations of gradient-based learning in LLMs post-training:

1. **Memory-Efficient Fine-Tuning:** Backpropagation requires storing activation maps for every layer, leading to massive memory footprints. Recent approaches like MeZO (Memory-efficient Zeroth-Order optimization) [120] demonstrate that LLMs with billions of parameters can be

fine-tuned using only forward passes, enabling training on consumer-grade hardware with negligible memory overhead compared to inference.

2. **Non-Differentiable Objectives:** The alignment of models with human intent often relies on metrics that are fundamentally discrete or black-box. Examples include maximizing "pass@k" rates in code generation, reducing "jailbreak success rates" in safety testing, or optimizing against a hard API-based reward signal. In these scenarios, the loss function $f(\theta)$ is not smooth, rendering gradients undefined.

Thus, ZO methods are no longer just a theoretical curiosity; they are becoming the standard interface for aligning models with complex, real-world constraints.

16.1.2 Historical Context and Theoretical Roots

While applying ZO optimization methods to billion-parameter transformers is new, the underlying mathematics rests on a rich history. These methods date back to the foundational work on direct search methods in the 1960s and the development of Evolution Strategies (ES) in the 1970s [164].

Classically, these algorithms address the problem of minimizing a function $f(x)$ using only query evaluations. The field evolved from simple heuristics (like Nelder-Mead simplex search [104]) to sophisticated randomized smoothing techniques. A theoretical milestone was Nesterov's analysis of Gaussian smoothing [133], which established that random search can be viewed as Stochastic Gradient Descent on a smoothed approximation of the objective function. Similarly, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [71] introduced mechanisms to adapt the geometry of the search space, making ZO methods robust to ill-conditioned landscapes.

In the following sections, we bridge the gap between classical literature and modern LLM applications by first introducing zero-order optimization methods and then showing how to apply them to LLMs.

16.1.3 Chapter Overview

The remainder of this chapter is organized as follows. Section 16.2 introduces the mathematical foundations of zeroth-order gradient estimation. Section 16.3 presents concrete algorithmic forms, from simple random search to sophisticated evolution strategies. Section 16.4 explores practical applications and speculative use cases in LLM post-training.

16.2 Zeroth-Order Gradient Estimators

Having motivated the need for gradient-free optimization in LLM post-training, we now formalize the mathematical machinery underlying zeroth-order methods. The key insight is that even without access to true gradients, we can construct unbiased estimators using only function evaluations.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be our objective function, and $\theta \in \mathbb{R}^d$ the vector of tunable parameters. The standard two-point estimator of the gradient along a random direction $u \sim \mathcal{N}(0, I_d)$ is:

$$\hat{g}(\theta; u) = \frac{f(\theta + \sigma u) - f(\theta - \sigma u)}{2\sigma} u. \tag{16.1}$$

The expectation over u gives the gradient of the *Gaussian-smoothed* function [133]:

$$f_\sigma(\theta) := \mathbb{E}_{z \sim \mathcal{N}(0, I_d)}[f(\theta + \sigma z)], \quad \text{so that } \mathbb{E}_u[\hat{g}(\theta; u)] = \nabla f_\sigma(\theta). \tag{16.2}$$

Bias–Variance Tradeoff

The mean squared error of the estimator satisfies:

$$\mathbb{E}\left[\|\hat{g}_n - \nabla f(\theta)\|^2\right] \lesssim \frac{dL^2}{n\sigma^2} + \sigma^2 \|\nabla^2 f(\theta)\|_F^2 + \frac{\mathrm{Var}[f]}{n}, \tag{16.3}$$

where L is a Lipschitz constant. The optimal smoothing radius $\sigma \sim (d/n)^{1/4}$ yields error scaling as $(d/n)^{1/2}$ [48].

16.3 Algorithmic Forms

The gradient estimators of Section 16.2 provide the theoretical foundation; we now turn to concrete algorithms that operationalize these ideas. We progress from simple random search to sophisticated distribution-based methods that adapt the geometry of the search space.

16.3.1 Random Search

Algorithm 1 Random Search with Population Sampling

Require: Objective f, initial point θ_0, perturbation scale σ, population size n, iterations T

Initialize $\theta \leftarrow \theta_0$
for $t = 1$ **to** T **do**
  Sample $\delta_1, \ldots, \delta_n \sim \mathcal{N}(0, \sigma^2 I)$
  Evaluate $f_i \leftarrow f(\theta + \delta_i)$ for $i = 1, \ldots, n$
  Select $i^* \leftarrow \arg\min_i f_i$
  if $f_{i^*} < f(\theta)$ **then**
    $\theta \leftarrow \theta + \delta_{i^*}$
  end if
end for
return θ

16.3.2 Distribution-Based Methods: CMA-ES and NES

While simple random search estimates a gradient for a fixed smoothing geometry, advanced Evolutionary Strategies (ES) dynamically adapt the search distribution itself. These methods model the optimization of $f(\theta)$ as maximizing the expected fitness $J(\phi) = \mathbb{E}_{\theta \sim \pi_\phi}[f(\theta)]$ over a parameterized family of distributions π_ϕ.

CMA-ES. The Covariance Matrix Adaptation Evolution Strategy [71] models the search distribution as a multivariate Gaussian $\mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{C})$. It updates the mean $\mathbf{m}$ via a weighted recombination of the top-μ selected samples. Crucially, it adapts the covariance matrix $\mathbf{C}$ by accumulating second-order information through two pathways:

1. **Rank-1 Update:** Utilizes an *evolution path* $\mathbf{p}_c$—an exponentially moving average of steps—to exploit correlations between consecutive generations.

2. **Rank-μ Update:** Estimates the local variance from the current population selection.

The update rule typically takes the form:

$$\mathbf{C}_{t+1} = (1 - c_1 - c_\mu)\mathbf{C}_t + c_1 \underbrace{\mathbf{p}_c \mathbf{p}_c^\top}_{\text{History}} + c_\mu \underbrace{\sum_{i=1}^{\mu} w_i \mathbf{y}_i \mathbf{y}_i^\top}_{\text{Local Population}}, \tag{16.4}$$

where $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_t)$ are the mutation vectors of selected individuals. This allows CMA-ES to handle ill-conditioned problems (e.g., narrow valleys in the loss landscape) effectively, provided the dimension d is moderate ($d \lesssim 100$).

Natural Evolution Strategies (NES). NES [199] unifies these heuristic updates under a Riemannian geometry framework. It treats the optimization trajectory as a gradient ascent on the statistical manifold equipped with the Fisher Information metric $\mathbf{F}(\phi)$. The update follows the *Natural Gradient*:

$$\phi_{t+1} = \phi_t + \eta \mathbf{F}^{-1} \nabla_\phi J(\phi). \tag{16.5}$$

Using the "log-derivative trick," the gradient is estimated via Monte Carlo sampling:

$$\nabla_\phi J(\phi) \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} f(\theta_i) \nabla_\phi \log \pi_\phi(\theta_i). \tag{16.6}$$

For Gaussian distributions, NES derives update rules that are analytically similar to CMA-ES, proving that covariance adaptation is approximating a second-order natural gradient descent.

16.3.3 ES for LLMs

Evolution Strategies (ES) have emerged as a viable alternative for post-training the LLMs, offering a gradient-free approach. Early studies showed that ES can train deep neural networks with millions of parameters about as effectively as traditional RL algorithms (e.g. Q-learning or policy gradients), while also achieving faster convergence by exploiting massive parallelism [35]. However, ES was long underutilized for LLM-scale models due to pessimism about its scalability [160]. Recent work addresses this issue by successfully fine-tuning full LLMs (with billions of parameters) using ES, demonstrating that ES can efficiently search such high-dimensional parameter spaces and even outperform state-of-the-art RL algorithms. Unlike gradient-based optimization (as in SGD backpropagation or policy-gradient RL), ES treats the model as a black-box. It optimizes by evaluating a population of perturbed models and updating the parameters based on aggregate reward functions. The appeal of ES in this context lies in its simplicity and reliability, providing an effective complement (or alternative) to conventional SGD and RL techniques for large-scale language model optimization.

While the preceding sections establish the algorithmic toolkit, practitioners face a crucial question: *where* in the LLM pipeline should these methods be applied? The answer depends on identifying low-dimensional control surfaces with black-box or non-differentiable feedback—a surprisingly common scenario in modern LLM deployment.

16.4 Case Studies and Speculative Applications of ZO Evolution in LLMs

This section bridges theory and practice by examining concrete scenarios where zeroth-order evolutionary strategies offer advantages over gradient-based alternatives. We focus on settings

that share a common structure: modest-dimensional control variables, noisy or non-differentiable rewards, and fixed or parameter-efficient model backbones.

16.4.1 Tuning Guardrails with Non-Differentiable Objectives

A common requirement in production LLMs is to enforce behavioral constraints, such as minimizing refusal rates without increasing harmful outputs, or increasing helpfulness without decreasing factuality. These objectives are often measured via non-differentiable metrics (e.g., binary accept/reject from human raters or classifiers). ZO optimization can tune:

- Thresholds in rule-based classifiers or ensemble moderation layers.
- Temperature/top-p parameters to balance diversity and control.
- Weights in reward aggregation pipelines combining safety/factuality heuristics.

Even when the LLM backbone remains fixed, the decoding or filtering layers form a low-dimensional parameter space with black-box reward feedback, ideal for ZO methods.

16.4.2 Optimizing Mixture-of-Experts Dispatch

In models with multiple expert networks (e.g., Switch Transformers or GShard), routing decisions can be governed by soft or hard gates. One may treat the gating function's parameters α as the target of ZO optimization, using feedback from downstream evaluation metrics:

- Preferential routing based on topic, region, or domain.
- Disabling experts that increase toxicity or contradiction rates.
- Rewarding expert sparsity while maintaining task performance.

A practical example: suppose a system deploys a 16-expert mixture and seeks to dynamically tune usage patterns to minimize latency while maintaining BLEU or safety scores on unseen data. This yields a compact 16-dimensional control vector α—a natural setting for NES or CMA-ES.

16.4.3 Language Model Alignment without Differentiability

Standard RLHF methods rely on a differentiable proxy reward model. However, in high-stakes settings, direct user preferences or multiple-choice A/B test outcomes may provide feedback with no known gradient. Here, ZO strategies enable alignment optimization without surrogate models.

One can imagine experiments such as:

- Collecting pass@k on human-written prompts across generations.
- Computing average rank among completions judged by crowd workers.
- Optimizing for adversarial robustness or jailbreak resilience.

These objectives are not differentiable with respect to model parameters or generation settings but are valid black-box functions of the system behavior.

16.4.4 Speculative: Meta-Controllers over Training Dynamics

Going further, we can speculate on using ZO methods to control aspects of training itself:

- Optimizing the curriculum schedule or prompt difficulty over training time.
- Tuning the entropy coefficients or clipping thresholds in PPO-like methods.
- Evolving augmentation policies or synthetic data ratios in a continual learning loop.

In these cases, ZO acts not on model weights but on the knobs governing how the model is trained. It becomes an evolutionary outer loop over the training process—a natural complement to first-order inner-loop updates, similar to Population Based Training (PBT) [88].

16.4.5 Summary of ZO-Friendly Structures

The common theme across these examples is a structure where:

1. The dimension r of the control variables is modest ($r \lesssim 100$).
2. The reward signal is noisy, expensive, or discontinuous.
3. The model itself remains fixed or uses parameter-efficient tuning (e.g., LoRA [84]).

In such cases, the simplicity and robustness of zeroth-order evolutionary methods can be an asset, not a liability.

Remark 3 (On the Future of ZO in LLM Ecosystems)**.** *As LLM deployment becomes increasingly modular—with reward cascades, multi-agent systems, and mixed-source data—black-box interfaces will proliferate. In these zones, gradient-based methods may falter, but zeroth-order strategies can provide resilient optimization, especially when embedded into meta-control loops.*

CHAPTER 17

Boolean Circuits and a Path to "Superintelligence" (by D. Koplow)

In recent years, it has become a running joke that the quickest path to a top-tier machine learning publication is to introduce a new benchmark. This obsession with narrow evaluations implicitly assumes that intelligence is tied to specific domains. While knowledge is inherently domain-specific, reasoning is not. We seek a minimal representational framework that requires very little background knowledge while still being able to express any computation. Boolean circuits meet this criterion. As universal computational models, they can represent any digital algorithm. When an LLM is unable to manipulate such circuits correctly, this shortcoming reveals a core limitation in its reasoning capabilities. This book argues that compositional sparsity is the primary factor enabling learnability in machine learning. From this viewpoint, we construct sparse Boolean circuits that contemporary models reliably fail to solve without programming, thereby revealing a strikingly human-like strategy for overcoming this constraint. This strategy may constitute a crucial step on the path toward super-intelligence.

17.1 Boolean Circuits

A Boolean circuit is a Directed Acyclic Graph (DAG) $G = (V, E)$ where input nodes represent binary variables $x_1, \ldots, x_n \in \{0, 1\}$ and internal nodes (gates) represent logical functions from a standard basis set such as $\mathcal{B} = \{\text{AND}, \text{OR}, \text{NOT}, \text{NOR}, \text{XOR}\}$. The power of this model lies in its universality. For any function $f : \{0, 1\}^n \to \{0, 1\}$ computable by a Turing machine in time $T(n)$, there exists a Boolean circuit of size $Poly(T(n))$ that computes f. This means that a model's ability to learn Boolean circuits is equivalent to its ability to *solve any* computational problem.

While evaluating a circuit is computationally trivial, the inverse problem, called *Circuit Synthesis*, is notoriously difficult. Given a dataset of input-output pairs $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$, finding the smallest circuit that is consistent with $\mathcal{D}$ is an NP-hard problem given a truth table that is exponentially large with respect to the input. Thus, the search space for circuits grows super-exponentially with the number of inputs. As a result, learning a generic function **efficiently is, in general, impossible**.

Yet, somehow, deep-learning is able to construct models able to solve many important problems in our world. As argued in this book, a core reason why is *compositional sparsity*. Under this assumption, the hypothesis space of circuits starts becoming much more tractable.

To keep one concrete mental model in view, assume we are trying to learn a compositionally sparse function. It can be represented as a Boolean circuit with fan-in 2 and depth $O(\log_2(n))$.

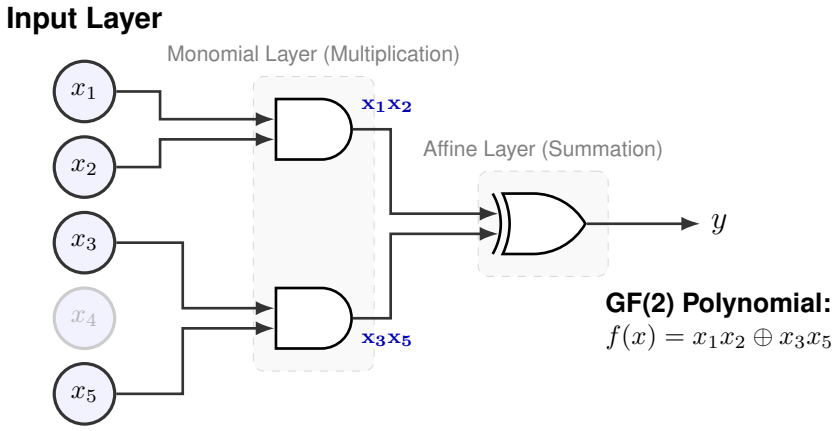


Figure 17.1: Example GF(2) circuit.

Further, we will restrict the class of circuits to be those that can be represented as a sum of monomials in GF(2). The resulting function has a bounded maximum degree of monomials. An example visualization is shown in Figure 17.1.

17.2 Learning

We now have at our disposal a highly expressive class of functions which, due to their compositional sparsity and adjustable complexity, allow us to evaluate how effectively any given model uncovers the appropriate compositionally sparse representation from data. At this point, it is crucial to clarify what we mean by learning in this context. Traditionally, learning is framed as the problem of finding a function f such that $f(x) \approx y$ for $(x, y) \sim D$, where D is some underlying data distribution. In contemporary machine learning, this view is largely aligned with the paradigm of "deep learning," in which we posit a parametric function $f_1(x; \theta)$ and apply some variant of gradient descent to adjust θ so as to maximize the predictive performance of $f_1(x; \theta)$. We shall refer to this approach as direct learning.

However, this is not the only way to obtain a function f. Alternatively, we might define a process g which, given access to D, does not directly output predictions, but instead produces a function f_2 that, when later executed, generates the correct outputs. This resulting function could then be run by the model itself or by an external executor. In this chapter, we will refer to this paradigm of learning as transform learning.

Both approaches constitute learning from data, but they do so at different levels of abstraction. In direct learning, the system identifies a specific function that maps inputs to outputs within a fixed hypothesis class. In transform learning, the system instead acquires a procedure for constructing such functions from data, thereby shifting part of the burden of generalization to the generated functions themselves. From this perspective, "solving a problem" and "learning a solution" are closely related operations that differ primarily in where adaptation occurs. Both aim to reduce uncertainty about an unknown target, though they encode and exploit that knowledge in structurally distinct ways.

17.2.1 Reasoning

From this standpoint, reasoning can be understood as an iterative procedure in which multiple functions are proposed to transform data, their transformed outputs are assessed, and those

assessments guide which new functions are generated next. Rather than committing immediately to a single function, the *system searches* over a broader space of candidate functions, composing and refining them in response to intermediate feedback. Under this interpretation, reasoning becomes a recursive form of transform-based learning.

When an LLM tackles a complex problem using chain-of-thought, it effectively plays the role of the generator g, emitting a sequence of operations that together implement a particular circuit f tailored to the given input. In the setting of "in-context learning," the model is not modifying its weights through gradient descent; instead, it is constructing a temporary program that maps the specific inputs to the appropriate output.

For the first time, we have a non-biological system that can learn a generator g which can be sampled to implement f across an exceptionally wide range of tasks. While much of the literature on compositional sparsity has focused on training-time optimization dynamics or the structure of the hypothesis space, we can now adopt new perspectives on compositional sparsity to illuminate the capabilities of modern LLMs: compositional sparsity in the functions f that g generates, and compositional sparsity in the way g executes f.

17.2.2 Measuring Learnability

If reasoning is, as we argue, a generative process where a model derives a program f, then we must rigorously quantify the reliability of that generation. How can we measure the probability that this step-by-step construction will eventually lead to a correct solution?

The Limits of Direct Learning. The standard lens for assessing learnability is the *Probably Approximately Correct (PAC)* framework **[valiant1984theory]**, which focuses on the ability of an algorithm to approximate a function from data.

Definition 7 (PAC-Learnability)**.** *A concept class $\mathcal{C}$ is PAC-learnable if there exists an algorithm that, for any target concept $c \in \mathcal{C}$ and any distribution D, outputs a hypothesis h with error at most ϵ with probability at least $1 - \delta$, using time and samples polynomial in $1/\epsilon, 1/\delta$, and the size of the concept.*

For general Boolean circuits, the news is bleak: they are known to be **not PAC-learnable [kearns1994cryptographic]**. If they were, one could trivially break standard cryptographic schemes like RSA by simply observing input-output pairs.

Crucially, however, our interest lies not in learning arbitrary opaque functions, but in **compositionally sparse** circuits. Unlike the general case, specific subclasses of these circuits often *are* theoretically learnable. Yet, despite this theoretical possibility, modern Deep Learning models operating in a "Direct Learning" mode, attempting to predict y directly from x, consistently fail to generalize on these tasks. This discrepancy suggests that while the class of functions may be learnable in principle, the standard supervised paradigm is not the mechanism by which Large Language Models (LLMs) solve them. To understand how models actually conquer these problems, we must shift our perspective from static approximation to sequential reasoning.

The Diligent Learner To model this dynamic derivation, we adopt the **Diligent Learner** framework proposed by Shalev-Shwartz et al. (2025). This approach reframes reasoning as a traversal through a decision tree of semantic states.

The Diligent Learner is defined formally as a tuple $\mathcal{L} = (Root, \mathcal{S}, V, \pi_\theta)$:

1. **Root:** The initial problem instance.

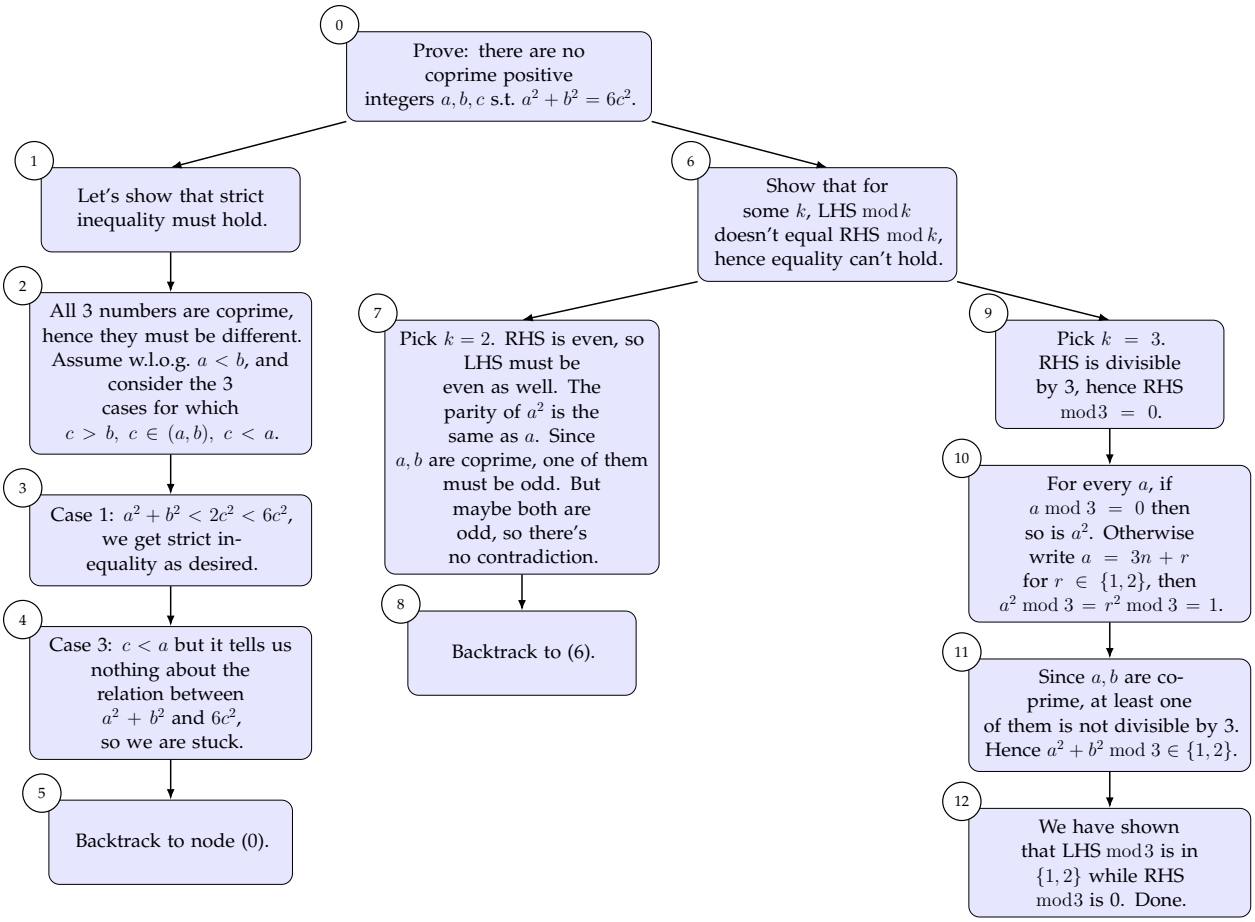


Figure 17.2: Diligent learner visualization from ShalevShwartzShashua2025Diligent.

2. **State Space $\mathcal{S}$:** A set of partial states h, where each h represents a prefix of a proof, program, or circuit.

3. **Validator V:** A mechanism (whether internal or external) that enforces local correctness, pruning branches that are logically invalid.

4. **Proposer π_θ:** The generator (agent) that proposes the next extension action based on the current state.

Figure 17.2 illustrates this process. At each step, the Proposer suggests a next step (for which the Validator must accept as logically sound) or decide to back track. The same approach can be used to synthesizing Boolean circuits from data: the model builds the circuit term-by-term with a Validator ensuring the step is logically sound.

The Bottleneck Parameter (γ). In this search-based paradigm, the critical bottleneck is the reliability of the *next step*. This reliability is quantified by γ, the probability that the Proposer will make a correct next step in solving the problem.

Let $\pi_\theta(\cdot \mid h)$ be the stochastic policy over actions approved by the validator, given a partial state h. An *extension action* proposes the next semantic step. We denote $S(h, a) = 1$ if the prefix (h, a) keeps the learner on a path that can be completed to a full, correct solution. The bottleneck

parameter γ represents the probability mass the model places on these *useful* next moves:

$$\Pr_{a\sim\pi_\theta(\cdot|h)}\left[a \in \{a_i \mid S(h,a_i)=1\}\right] \;\geq\; \gamma. \tag{17.1}$$

This parameter effectively dictates the feasibility of the reasoning process. If γ remains high, the learner consistently identifies the sparse, valid path through the logic. However, if γ decays as the depth of reasoning increases, the search complexity explodes. Specifically, for a target failure probability δ and maximum reasoning depth $T_{\max}$, the number of steps required scales as:

$$O\Big(T_{\max} \cdot \frac{\log(T_{\max}/\delta)}{\gamma}\Big). \tag{17.2}$$

While contemporary LLMs do not currently perform this explicit backtracking search, this framework provides a crucial theoretical upper bound on a model's reasoning capacity. As they allude to in their work, maximizing γ is the most important objective for any system trying to attain "superintelligence."

17.3 The Computational Cliff

Realizing the full "Diligent Learner" with backtracking, state validation, and fine-tuned proposers presents engineering and reliability hurdles. Yet, we need not build the complete system to test the paradigm. We can leverage part of the framework to assess the capabilities of existing models by isolating the single-step generation process. Even without the full apparatus of recursion, testing whether an LLM can reliably produce the next valid term in a sparse circuit offers a powerful diagnosis of its reasoning fidelity.

To evaluate this capability, we introduce a stepwise reconstruction benchmark over GF(2). At each depth g, the model is presented with a prefix P_g and a labeled sample set S_g, and must output the next monomial t_{g+1}. The setup is designed to be rigorous: Ignoring the prefix P_g leaves the labels masked by the structure of the prefix. Ignoring the samples S_g reduces the task to blind guessing.

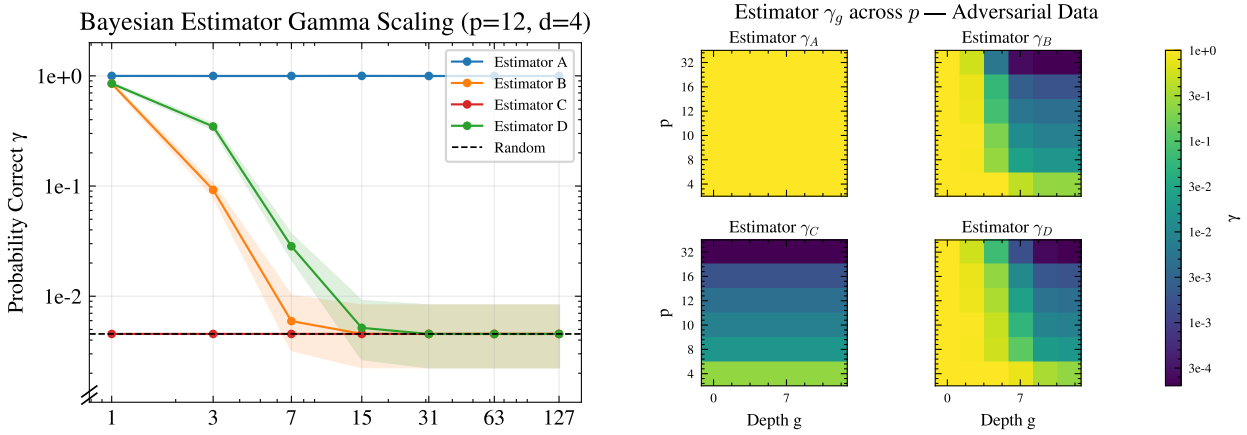
17.3.1 Problem Formulation

We articulate the game using the clean language of Boolean functions in GF(2), revealing the function one monomial at a time.

Targets. We consider inputs $x = (a, v) \in \{0,1\}^{n+p}$, where $a = (a_1, \ldots, a_n)$ are address bits and $v = (v_1, \ldots, v_p)$ are payload bits. We restrict our attention to Algebraic Normal Form (ANF): XORs of monomials, $f(a,v) = \bigoplus_{j=1}^{n} t_j(a,v)$, where each term takes the form

$$t_j(a,v) := a_j\, M_j(v), \quad M_j(v) := \prod_{i\in S_j} v_i, \tag{17.3}$$

with supports $S_j \subseteq [p]$ of fixed size $|S_j| = d-1$. Each term is effectively sparse, touching only d variables, and the entire target is a sparse sum of such terms. An *instance* consists of the ordered support sequence $(S_1, \ldots, S_n)$, sampled once and then held fixed.



(a) Only history+data sustains reliable next-step prediction: step success γ_g versus depth g.

(b) As g and p grow, imperfect-information estimators collapse; only $\mathcal{A}$ remains reliable.

Figure 17.3: Stepwise reconstruction performance under adversarial sampling.

The Stepwise Reconstruction Game. At depth $g \in \{0, \ldots, n-1\}$, the learner receives:

1. the ordered prefix $P_g := (t_1, \ldots, t_g)$, and
2. a labeled sample set $\mathsf{S}_g := \{(x^{(k)}, y^{(k)})\}_{k=1}^{K}$ generated for that specific depth.

The model's task is to output a candidate monomial $\hat{t}$. Success is binary: the model succeeds if and only if $\hat{t} = t_{g+1}$. We score the model by its exact success probability,

$$\gamma_g \;:=\; \Pr_{\hat{t}\sim\pi_\theta(\cdot|P_g,\mathsf{S}_g)}\big[\hat{t} = t_{g+1}\big], \tag{17.4}$$

where the probability is taken over the sampled instance, oracle randomness, and the model's own stochasticity. Since the target is an ordered sequence, there is exactly one correct continuation at each depth, making γ_g a measure of exact-next accuracy.

Who is allowed to see what? We distinguish between solvers based on the information they utilize: (i) **Diligent** ($\mathcal{A}_g$) solvers access both the prefix and samples (P_g, S_g); (ii) **Data-only** ($\mathcal{B}_g$) solvers see S_g but are blind to P_g; (iii) **History-only** ($\mathcal{C}_g$) solvers see P_g but lack S_g; and (iv) **Partial** ($\mathcal{D}_g$) solvers have limited access to each. The benchmark is explicitly tuned to separate these classes:

$$\min_g \gamma_g^{\mathcal{A}} \;\geq\; Q \quad \text{while} \quad \gamma_g^{\mathcal{B}},\, \gamma_g^{\mathcal{C}},\, \gamma_g^{\mathcal{D}} \;\approx\; \tfrac{1}{\binom{p}{d-1}}. \tag{17.5}$$

In essence: if a model refuses to use either the prefix or the samples, it is forced to fall back to random guessing.

17.3.2 The Statistical Obfuscation Construction

We enforce the separation between reasoning and pattern matching through statistical obfuscation. At depth g, the label represents a superposition (the XOR sum) of the target signal $M_{g+1}(v)$ and a history-dependent mask. Because this mask is a deterministic function of the prefix P_g and input x, it completely randomizes the label for any observer neglecting variables from the history. For the diligent solver, however, the mask is transparent.

Instance Generation. We sample supports $S_1, \ldots, S_n \subseteq [p]$ with $|S_j| = d-1$ once per instance. This fixes the ordered ANF terms $(t_1, \ldots, t_n)$ in (17.3).

Payload Distribution. We draw payloads at a fixed Hamming weight w, sampling v uniformly from $\{v \in \{0,1\}^p : \|v\|_0 = w\}$. For any fixed S with $|S| = d-1$, we define

$$\rho(w) \;:=\; \Pr_v\left[M_S(v) = 1\right] \;=\; \frac{\binom{w}{d-1}}{\binom{p}{d-1}} \quad (w \geq d-1), \tag{17.6}$$

selecting $w^\star$ such that $\rho(w)$ approximates $1/2$. This entropy maximization preserves the mask's efficacy and minimizes label bias.

Step-g Sampling Oracle (Fixed-Prefix Obfuscation). Given an instance and depth g, the oracle produces $\mathsf{S}_g = \{(a^{(k)}, v^{(k)}, y^{(k)})\}_{k=1}^K$ by constructing a set of carefully designed examples. It fixes $a_{g+1} = 1$ (zeroing subsequent bits) while sampling preceding address bits randomly, outputting $y := f(a, v)$.

The label decomposes as:

$$y \;=\; \underbrace{\Big(\bigoplus_{j=1}^{g} a_j\, M_j(v)\Big)}_{\text{prefix obfuscation}} \;\oplus\; \underbrace{M_{g+1}(v)}_{\text{next-term signal}}\;. \tag{17.7}$$

The mask is fully determined by the prefix and input. A solver utilizing the prefix can cleanly expose the signal, whereas a solver refusing the prefix perceives only noise.

17.3.3 Theoretical Guarantees

We provide rigorous guarantees for this separation. The next-term signal remains statistically inaccessible to history-blind solvers while being fully recoverable by history-aware ones. The barrier is one of information access and state tracking, rather than computational complexity.

Monomial firing at fixed weight. Since payloads are uniform over a fixed Hamming-weight sphere, the probability of a payload monomial "firing" follows a closed form.

[Monomial firing probability at fixed Hamming weight]lemmamonomialWeight Fix integers $p \geq d-1 \geq 1$ and $w \in \{0, \ldots, p\}$. Let v be uniform over the Hamming sphere $\{v \in \{0,1\}^p : \|v\|_0 = w\}$, and fix any $S \subseteq [p]$ with $|S| = d-1$. Define $M_S(v) := \prod_{i \in S} v_i$. Then

$$\Pr\left[M_S(v) = 1\right] = \begin{cases} \dfrac{\binom{w}{d-1}}{\binom{p}{d-1}} = \dfrac{\binom{p-(d-1)}{w-(d-1)}}{\binom{p}{w}} & \text{if } w \geq d-1, \\ 0 & \text{if } w < d-1. \end{cases}$$

Per-sample Obfuscation. A data-only estimator $\mathcal{B}_g$, observing data but lacking the prefix P_g, cannot derive the supports determining the mask. We demonstrate a *single-example* guarantee: across the unknown prefix supports, each individual labeled example offers negligible Bayes advantage regarding the next-term signal.

[Bayes masking given observed (a, v)]lemmaBayesMasking Assume the instance distribution samples $S_1, \ldots, S_g, S_{g+1}$ i.i.d. uniformly from $\{S \subseteq [p] : |S| = d-1\}$, independently of the oracle samples. Fix a step g and condition on a realized example (a, v) with $\|v\|_0 = w^\star$. Let

$$\rho := \rho(w^\star) = \Pr_S[M_S(v) = 1] = \frac{\binom{w^\star}{d-1}}{\binom{p}{d-1}}, \quad m := m(a).$$

Then, marginalizing over the unknown prefix supports $(S_1, \ldots, S_g)$, for each $r \in \{0, 1\}$,

$$\Pr\left[B(a, v) = r \mid a, v\right] \;=\; \tfrac{1}{2}\left[1 + (-1)^r (1 - 2\rho)^m\right].$$

Moreover, $B(a, v)$ is independent of $b = M_{g+1}(v)$ given (a, v), and since $y = B(a, v) \oplus b$ we have

$$\left|\Pr[y = b \mid a, v] - \tfrac{1}{2}\right| \;=\; \tfrac{1}{2}|1 - 2\rho|^m.$$

Lemma 17.3.3 confirms that the mask effectively neutralizes single-sample bias, which decays exponentially with the number of active address bits.

History-only Baseline. Reliance on history alone is similarly ineffective. The prefix determines past supports without leaking information about the future support S_{g+1}.

[History-only is prior guessing]lemmaHistoryOnly Assume the instance distribution samples supports $S_1, \ldots, S_n$ i.i.d. uniformly (*with replacement*) from $\{S \subseteq [p] : |S| = d-1\}$. Then for any $g < n$, conditioned on the revealed prefix P_g, the next support S_{g+1} is uniform over $\{S \subseteq [p] : |S| = d-1\}$ and independent of P_g. Consequently, any history-only estimator satisfies $\Pr\left[\widehat{S} = S_{g+1}\right] \le \frac{1}{\binom{p}{d-1}}$.

Having established the insufficiency of partial information, we demonstrate that a Diligent solver, unifying data and history, can convert the task into a tractable polynomial-time problem.

Recoverability in Polynomial Time. With access to P_g, the solver computes residual labels

$$r^{(k)} \;:=\; y^{(k)} \oplus \bigoplus_{j=1}^{g} a_j^{(k)} M_j(v^{(k)}) \;=\; M_{g+1}(v^{(k)}), \tag{17.8}$$

reducing the problem to identifying the unknown support S_{g+1} from labeled payloads via an intersection decoder:

$$\widehat{S} \;:=\; \bigcap_{k \in K_+} \operatorname{supp}(v^{(k)}), \tag{17.9}$$

where $K_+ := \{k : r^{(k)} = 1\}$.

[Poly-time recovery under fixed-weight payloads]theoremRecoverability Fix an instance and assume payloads are i.i.d. uniform on $\{v \in \{0,1\}^p : \|v\|_0 = w^\star\}$ with $w^\star \ge d-1$. Let $\rho = \rho(w^\star) = \Pr[M_{g+1}(v) = 1]$ and $T = |K_+|$. Then $\Pr[T = 0] = (1-\rho)^K$. Moreover, conditioned on $T \ge 1$, the decoder succeeds with probability at least

$$1 - \min\Big\{1,\; (p - (d-1))\Big(\frac{w^\star - (d-1)}{p - (d-1)}\Big)^T\Big\}. \tag{17.10}$$

Proof. If $r^{(k)} = 1$ then $M_{g+1}(v^{(k)}) = 1$, which is equivalent to $S_{g+1} \subseteq \mathrm{supp}(v^{(k)})$. Thus for every $k \in K_+$ we have $S_{g+1} \subseteq \mathrm{supp}(v^{(k)})$, and hence

$$S_{g+1} \subseteq \bigcap_{k \in K_+} \mathrm{supp}(v^{(k)}) \;=\; \widehat{S}.$$

Therefore, conditioned on $T \geq 1$, the intersection decoder can fail only if $\widehat{S}$ contains at least one *extraneous* coordinate $i \notin S_{g+1}$, i.e., some $i \in [p] \setminus S_{g+1}$ appears in every positive support. Fix any $i \notin S_{g+1}$ and consider a single draw v conditioned on $r = 1$. Under the fixed-weight model, after forcing ones on S_{g+1}, the remaining $w^\star - (d-1)$ ones are chosen uniformly among the $p - (d-1)$ coordinates in $[p] \setminus S_{g+1}$. Hence

$$\Pr[i \in \mathrm{supp}(v) \mid r = 1] \;=\; \frac{w^\star - (d-1)}{p - (d-1)}.$$

Now condition on the event $\{T = |K_+|\}$ and on the index set K_+ itself. Because the original examples are i.i.d., the payloads $\{v^{(k)}\}_{k \in K_+}$ are i.i.d. draws from the conditional distribution $(v \mid r = 1)$, so

$$\Pr\Big[i \in \mathrm{supp}(v^{(k)})\ \forall k \in K_+ \,\Big|\, T\Big] \;=\; \Big(\frac{w^\star - (d-1)}{p - (d-1)}\Big)^T.$$

Taking a union bound over the $p - (d-1)$ possible extr coordinates gives

$$\Pr\left[\widehat{S} \neq S_{g+1} \mid T\right] \;\leq\; (p - (d-1))\Big(\frac{w^\star - (d-1)}{p - (d-1)}\Big)^T,$$

which implies (17.10). Finally, since $T = \sum_{k=1}^K \mathbf{1}\{r^{(k)} = 1\}$ with $\Pr\Big[r^{(k)} = 1\Big] = \rho = \rho(w^\star)$, we have $\Pr[T = 0] = (1-\rho)^K$. □

[High-probability recovery with K samples]corollaryRecoverabilityHP In the setting of Thm. 17.3.3, let $\rho := \rho(w^\star) = \Pr[M_{g+1}(v) = 1]$ and $\alpha := \frac{w^\star - (d-1)}{p-(d-1)} \in [0, 1)$. Fix $\delta \in (0, 1)$. For $\alpha \in (0, 1)$ define

$$T_0 := \left\lceil \frac{\log(2(p - (d-1))/\delta)}{\log(1/\alpha)} \right\rceil \quad \text{(and if } \alpha = 0, \text{ take } T_0 := 1).$$

If $K \geq \frac{1}{\rho} \max\{2T_0,\ 8\log(2/\delta)\}$, then the decoder in (17.9) outputs $\widehat{S} = S_{g+1}$ with probability at least $1 - \delta$.

Proof. Let $T = |K_+| = \sum_{k=1}^K \mathbf{1}\{r^{(k)} = 1\}$. Since $r^{(k)} = M_{g+1}(v^{(k)})$ and the payloads are i.i.d., we have $T \sim \mathrm{Bin}(K, \rho)$. By Thm. 17.3.3, for any $t \geq 1$,

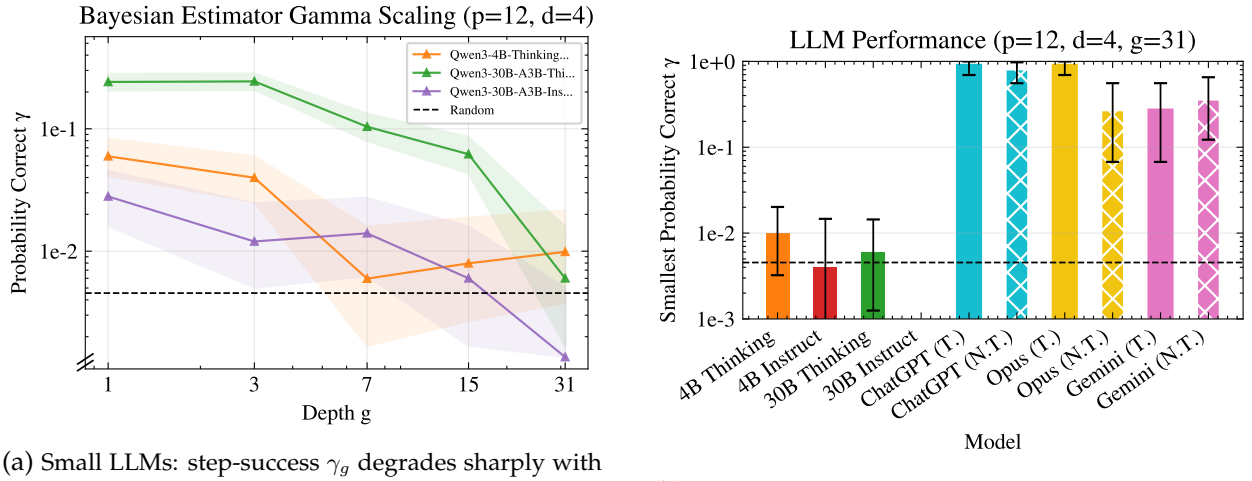
$$\Pr\Big[\widehat{S} \neq S_{g+1} \mid T = t\Big] \;\leq\; (p - (d-1))\alpha^t.$$

Hence

$$\Pr\Big[\widehat{S} \neq S_{g+1}\Big] \;\leq\; \Pr[T < T_0] \;+\; \Pr\Big[\widehat{S} \neq S_{g+1} \mid T \geq T_0\Big].$$

For the first term, our lower bound on K implies $K\rho/2 \geq T_0$, so by a multiplicative Chernoff bound,

$$\Pr[T < T_0] \;\leq\; \Pr\Big[T \leq \tfrac{1}{2}K\rho\Big] \;\leq\; \exp\Big(-\tfrac{1}{8}K\rho\Big) \;\leq\; \delta/2,$$



(a) Small LLMs: step-success γ_g degrades sharply with depth g.

(b) Frontier LLMs achieve higher γ_g than smaller models.

Figure 17.4: Depth-induced collapse in next-step prediction across model families.

where the last inequality uses $K\rho \geq 8\log(2/\delta)$. For the second term, on $T \geq T_0$ we have

$$\Pr\left[\widehat{S} \neq S_{g+1} \mid T \geq T_0\right] \;\leq\; (p-(d-1))\alpha^{T_0} \;\leq\; \delta/2,$$

by the definition of T_0 (and trivially if $\alpha = 0$). Combining the two bounds yields $\Pr\left[\widehat{S} \neq S_{g+1}\right] \leq \delta$. □

These results define the information-theoretic landscape. Lemma 17.3.3 and Lemma 17.3.3 establish the necessity of combining signal sources, proving that isolation leads to failure. Theorem 17.3.3 provides the constructive converse: the prefix serves as a decryption key, transforming the noisy label into a clean signal recoverable with polynomial resources. The task separates those who track state from those who merely pattern-match.

Efficient Validation. A robust benchmark requires a fast grader. Given the instance specification and a candidate monomial $\tilde{t}$, our validator parses $\tilde{t}$ and accepts if and only if the payload index set $\widetilde{S}$ matches S_{g+1}. This process is highly efficient, taking $O(|\tilde{t}| + d\log d)$ time, allowing for rapid evaluation.

17.3.4 Tool Synthesis

This brings us to a phenomenon we term the "Computational Cliff." Circuit synthesis requires exact state tracking. Local errors eventually propagate and corrupt the entire derivation in a way that can be well described as a Gaussian model with only a partial memory. While the exact complexity of the problem that triggers the collapse changes with the capacity of the model, we do observe it for all models beyond a critical depth that do not build and use tools koplow2026superintellgence.

Shortening Error Chains In a reasoning trace of length N, independent error probabilities ϵ compound to yield a success rate of $(1-\epsilon)^N$. Tool use restructures this probability. By offloading complex logic to verified external solvers, the effective logical depth contracts, replacing fragile token-by-token generation with robust atomic operations.

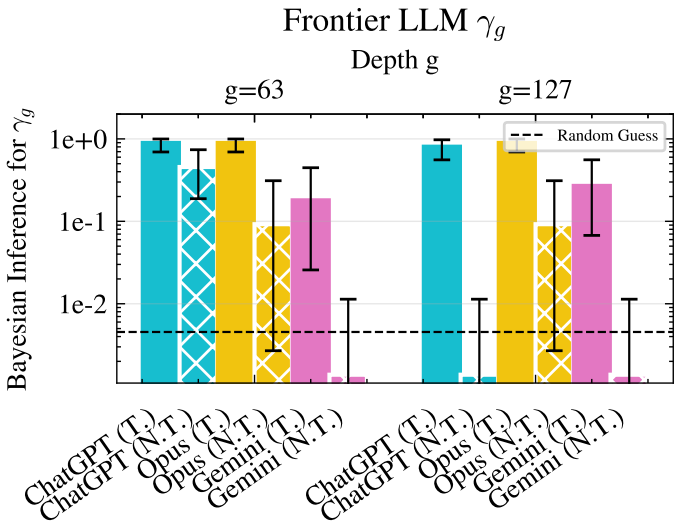


Figure 17.5: Frontier models with tool use (T.) sustain higher γ_g at larger depths than no-tools variants (N.T.).

The Law of Recursion This aligns with the **Law of Recursion [koplow2026lawsofemergence]**:

> "A system designed to maximally utilize compositional sparsity will design other systems that can use compositional sparsity even better for specific domains."

A sophisticated intelligence does not simulate all computation within its primary weights. It builds specialized external structures to handle irreducible complexity. Biologically, the cell does not rely on a homogeneous cytoplasm for all chemistry; it compartmentalizes function into organelles. Similarly, synthetic intelligence ought not simulate circuit evaluation in raw neural activations, but rather synthesize programs to execute the task. Reasoning, then, is the act of generating the executable.

Part IV
Speculations

CHAPTER 18

Consistency in Language Models

A simple intuition is that transformers behave like large associative memories indexed by prompts [193, 162]. This perspective raises a puzzle. If a transformer is, at heart, an associative memory, why are its responses to a prompt typically so coherent and well-formed, rather than a collage of fragments retrieved from training examples? We consider this puzzle and propose a solution based on the sparse compositionality and long-range contextual integration properties of the learned function.

18.1 Definition of Consistency

Let $T = (s_1, s_2, \ldots, s_n)$ be a generated text composed of n sentences or segments. We aim to formalize the notion of *consistency* of T as the degree to which the content remains centered on a single topic throughout the sequence.

Definition 8 (Topic Consistency)**.** *Given a representation function $\phi : \mathcal{S} \to \mathbb{R}^d$ that maps a sentence s_i to a d-dimensional embedding vector in a semantic space [165], the* topic consistency *of T is defined as:*

$$\mathrm{Cons}(T) = \frac{1}{n} \sum_{i=1}^{n} \left\| \phi(s_i) - \bar{\phi} \right\|^2, \quad \textit{where} \quad \bar{\phi} = \frac{1}{n} \sum_{j=1}^{n} \phi(s_j)$$

In this formulation, $\bar{\phi}$ denotes the mean topic embedding (centroid) of the text, and $\mathrm{Cons}(T)$ measures the average squared deviation of individual sentence embeddings from the central topic. Lower values of $\mathrm{Cons}(T)$ correspond to higher consistency.

18.2 Contextual Representations and Associative Memory Hypothesis

Transformers maintain per-token representations through multi-head self-attention over a growing context window. For decoder-style architectures (e.g., GPT), the representation h_i for token x_i depends only on the prefix $x_{<i}$ [161]. We define a representation function $\phi(s_i)$ that depends on the hidden state h_i of the transformer at the time of generating sentence s_i.

We hypothesize that transformers exhibit associative memory behavior by retrieving contextually aligned representations. However, unlike static lookup memories, transformers maintain consistency due to the deep compositional structure of the learned function and the architecture's ability to maintain topic coherence over long contexts [25].

18.3 Contextual Consistency Hypothesis

Definition 9 (Contextual Consistency Hypothesis)**.** *Let $T = (s_1, \ldots, s_n)$ be a generated sequence from a transformer-based model M where each s_i is generated conditional on prior context $C_i = (s_1, \ldots, s_{i-1})$. We define $\phi(s_i) = f(h_i)$ where h_i is the hidden representation computed by M at position i.*

If $|C_i|$ is large and semantically coherent, then the distribution $P(s_i \mid C_i)$ is concentrated on outputs such that $\phi(s_i)$ is close to the centroid $\bar{\phi}_i = \frac{1}{i-1}\sum_{j=1}^{i-1}\phi(s_j)$.

This hypothesis suggests that long-context attention promotes semantic consistency in the output by stabilizing the distribution over continuations.

18.4 Theoretical Bound

Theorem 12 (Expected Consistency Bound)**.** *Let $T = (s_1, \ldots, s_n)$ be a text sequence generated by a transformer. Suppose ϕ is L-Lipschitz in the transformer's hidden state h_i, and that each s_i is sampled from $P(s_i \mid C_i)$.*

Assume that:

(i) For all i, $\mathbb{E}\left[\|\phi(s_i) - \bar{\phi}_i\|^2\right] \leq \varepsilon(|C_i|)$, with $\varepsilon(k)$ decreasing in k.

(ii) $\bar{\phi}_i = \frac{1}{i-1}\sum_{j=1}^{i-1}\phi(s_j)$, and $\bar{\phi} = \frac{1}{n}\sum_{i=1}^{n}\phi(s_i)$.

Then the expected topic inconsistency satisfies:

$$\mathbb{E}[\mathrm{Cons}(T)] \leq \frac{1}{n}\sum_{i=1}^{n}\varepsilon(|C_i|) + O\left(\frac{1}{n}\sum_{i=1}^{n}\|\bar{\phi}_i - \bar{\phi}\|^2\right).$$

18.5 Illustration

Figure 18.1 illustrates a representative curve $\varepsilon(k) \propto 1/\sqrt{k}$, capturing the empirical intuition that longer context length reduces topic inconsistency.

18.6 Conclusion

We propose that the internal consistency of transformer-generated text arises not from memorization alone, but from compositionality and context integration in the learned model. We formalize a topic-based consistency measure, link it to architectural mechanisms, and establish a theoretical bound on expected topic drift. This view supports future work on architectural constraints, interpretability, and inductive bias design in language models [50].

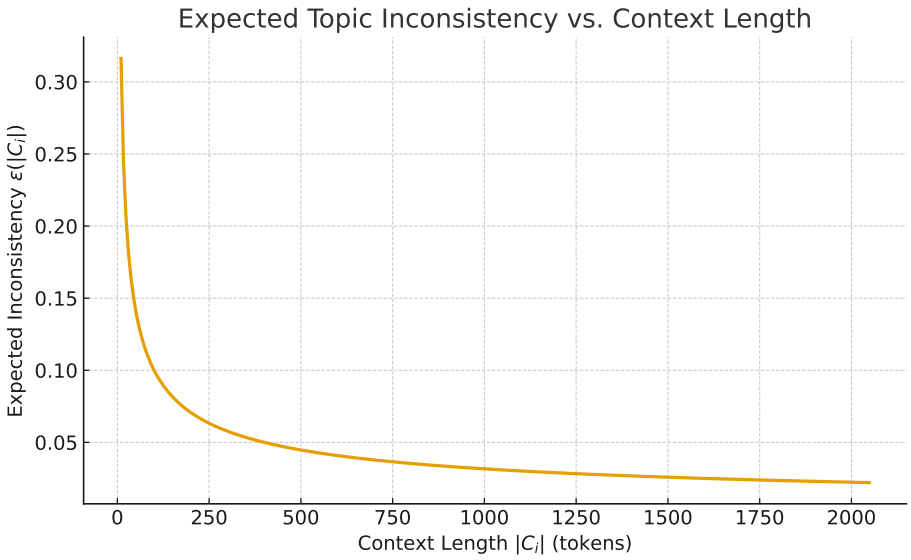


Figure 18.1: Expected inconsistency $\varepsilon(|C_i|)$ as a function of context length $|C_i|$. Longer contexts lead to more stable topic embeddings.

CHAPTER 19

Learning 2D views, recognizing 3D objects: what is the structure of embeddings

How does the brain recognize objects from different viewpoints? Does it have a 3D model of things? What is it and how is it learned?

19.1 The 1994 Paradigm Shift

In their landmark study, Poggio, Edelman, Bueltoff and Logothetis (1990, 1994) challenged the prevailing "3D model-based" theory of object recognition. Traditional computer vision had long assumed the brain must reconstruct a full 3D internal representation—akin to a CAD model—to recognize an object from a novel viewpoint. Poggio and Edelman proposed a more biologically plausible alternative: **view-invariant recognition through interpolation** [149].

They argued that the brain stores a small number of discrete "key views" and learns the mapping between them using **Radial Basis Function (RBF) networks**. When a novel view is presented, the network does not perform a geometric rotation; instead, it performs high-dimensional interpolation between the stored examples.

19.2 Supporting Evidence: Psychophysics and Physiology

The strength of this proposal lay in its multidisciplinary validation:

- **Human and Monkey Psychophysics:** Behavioral experiments revealed that recognition speed and accuracy were "view-dependent." Performance degraded systematically as the object was rotated away from a learned view—a hallmark of interpolation rather than 3D reconstruction [115].
- **Monkey Physiology:** Recordings from the Inferior Temporal (IT) cortex identified "view-tuned" neurons. These cells responded maximally to specific orientations and showed a bell-shaped decay in firing rate as the object rotated, directly mirroring the Gaussian basis functions of an RBF network [116].

19.3 The Geometry of the Embedding Space in Deep Networks

The embedding space (latent space) of a view-trained deep network reveals deep differences between biological-style representations and the 3D models used in computer graphics.

19.3.1 Topology vs. Geometry

A 3D model in computer graphics is defined by Euclidean coordinates. In contrast, the embedding space of a view-trained network is **topological and manifold-based**. The network learns a "view manifold"—a lower-dimensional surface where similar views are clustered. Recognition is about situating an input onto the correct manifold.

19.3.2 Identity and Pose Disentanglement

Modern deep networks often "disentangle" identity from pose. In the latent space, identity remains a stable vector, while pose varies along a specific axis. Recent work in adversarial alignment (Hossain et al., 2025) has shown that we can explicitly optimize embeddings to be view-invariant by "forgetting" the camera angle while "remembering" the object identity [83].

19.4 Compositional Sparsity and the DAG Architecture

The evolution from shallow RBFs to deep architectures is grounded in **compositional sparsity**. While a standard RBF struggles with the curse of dimensionality, a deep network avoids this by decomposing the target function into a hierarchy of local, low-dimensional functions represented as a **Directed Acyclic Graph (DAG)**.

In this graph, nodes represent constituent parts that are themselves low-dimensional. This structure explains how the brain recognizes objects even when partially occluded: it interpolates on a sub-graph of the object's compositional structure.

19.5 Modern Extensions: Multimodal Alignment and Scaling

Recent years have seen a convergence of these ideas with massive-scale learning.

19.5.1 Unified 3D-2D-Language Embeddings

Models like ULIP (Xue et al., 2023) and SigLIP 2 (Tschannen et al., 2025) align 3D point clouds into the same latent space as 2D images and text [205, 187]. This suggests that the "view manifold" envisioned in 1990 is now being scaled into a "universal manifold," where a linguistic description of a chair and a 3D scan of that chair occupy the same embedding coordinates [181].

19.5.2 The Embodied Turing Test

A critical recent development is the comparison of biological learning with AI agents. Wood et al. (2025) conducted an "Embodied Turing Test," raising AI agents in controlled virtual environments identical to those used for newborn chicks. This research investigates whether the "view-invariant" properties found in biological systems naturally emerge from the DAG-like hierarchical structures of modern Transformers when exposed to the same temporal visual flow [202].

The brain's "model" of a 3D object is a fluid, learnable manifold represented by a compositionally sparse DAG. While RBF networks provided the first mathematical bridge for this idea, modern architectures—from ULIP to SigLIP—represent its scaling into high-dimensional latent spaces, moving us closer to a unified theory of visual and conceptual embeddings.

19.6 Learning Invariant Object Representations from View Sequences

A useful testbed for theories of hippocampal (see chapter 23)–cortical function is the following thought experiment. Consider an agent that observes a sequence of images generated by moving around a rigid three-dimensional object (e.g. walking around a human face). The sensory stream consists of temporally ordered views induced by continuous self-motion. The central question is what kind of internal representation is required to both predict these sequences and support strong viewpoint-invariant recognition.

Formally, let $x_t \in \mathbb{R}^{H\times W\times 3}$ denote the image observed at time t, and let u_t denote the agent's self-motion between t and $t+1$. Each image is generated by an unknown object o with fixed geometry g_o and appearance a_o, viewed from a time-varying pose $p_t \in SE(3)$. The generative process is

$$x_t = R(g_o, a_o, p_t) + \epsilon_t, \tag{19.1}$$

where R is a (possibly implicit) rendering function and ϵ_t denotes sensory noise.

A deep feedforward network can be trained to recognize the object across viewpoints by collapsing pose variability into an invariant embedding. However, such recognition does not require the explicit recovery of the underlying 3D structure. In contrast, the sequential nature of the observations provides sufficient information to infer a latent object-centric model. In particular, the agent can exploit temporal continuity and known self-motion to bind successive views into a coherent trajectory:

$$p_{t+1} \sim p(p_{t+1} \mid p_t, u_t). \tag{19.2}$$

An architecture capable of exploiting this structure must separate fast episodic binding from slow structural learning. Episodic memory stores sequences

$$\mathcal{E} = \{(x_t, p_t, u_t)\}_{t=1}^{T}, \tag{19.3}$$

enabling rapid association of multiple views as belonging to the same object. Over longer timescales, a cortical world model integrates evidence across episodes to learn a persistent object representation (g_o, a_o) by minimizing multi-view reconstruction error:

$$\mathcal{L}_{\text{geom}} = \sum_{t=1}^{T} \|x_t - R(g_o, a_o, p_t)\|^2. \tag{19.4}$$

Crucially, recognition in such a system is not a feedforward classification problem but an inference problem. Given a single novel image x, object identity is inferred by selecting the object model and pose that best explain the sensory input:

$$(o^*, p^*) = \arg\min_{o,p} \|x - R(g_o, a_o, p)\|^2. \tag{19.5}$$

Viewpoint invariance arises naturally because pose is an explicit latent variable rather than a nuisance factor to be suppressed.

This thought experiment highlights a qualitative gap between current deep recognition systems and biological vision. While modern networks can learn invariance directly from data, they

generally lack an explicit, manipulable 3D world model learned from temporal experience. The hippocampal–cortical system appears to solve a harder problem: constructing object-centric generative models from sequences of views and using these models both for prediction and for invariant recognition. Architectures that genuinely recover and exploit such latent structure remain largely unexplored in artificial systems.

CHAPTER 20

More on Genericity Conjecture (with P. Beneventano)

We formulate the conjecture that functions that are appropriate for learning should be generic, that is independent of the choice of the origin of the coordinate system used for the regressor. We show that genericity implies the presence of a linear footprint in the target function. We then show that linear terms are important for allowing consistent convergence of deep networks, and removing them makes optimization much harder.

20.1 Motivation and Informal Conjecture

Deep networks are often trainable to near-zero error by gradient-based methods, even at extreme depth, provided that each layer retains some degree of *linearity*—through identity skip connections or the linear component of ReLU-like activations. Empirically, architectures that eliminate all linear components (e.g., purely even activations such as $\sigma(z) = z^2$ without skip connections) are dramatically harder to optimize, even when they have enough parameters to represent the same target function.

We conjecture that this distinction has a precise analytical expression: the presence of a nontrivial *degree–1 component* in the orthogonal-polynomial expansion of each layer's scalar channel is crucial for the consistent convergence of gradient descent across layers. Architectures in which all layers suppress this degree–1 mode are generically hard to optimize by gradient-based algorithms.

A key idea is that the existence (or suppression) of linear terms is not an accidental artifact of coordinates but a *generic* structural property of the functions represented by the network. The chapter develops this by combining:

- an orthogonal polynomial (Hermite-like) view of information exponents,
- a notion of genericity based on Taylor jets and structural stability,
- and a layer-wise "effective information exponent" for deep architectures.

20.2 Preliminaries: Orthonormal Polynomial Framework

Let μ be a probability measure on $\mathbb{R}$ with all finite moments:

$$\int |z|^k \, d\mu(z) < \infty \quad \forall k \geq 0.$$

Applying Gram–Schmidt to the monomials $1, z, z^2, \dots$ in $L^2(\mu)$ yields an orthonormal polynomial family $\{\psi_k\}_{k\geq 0}$ satisfying

$$\deg(\psi_k) = k, \qquad \int \psi_k(z)\psi_\ell(z)\, d\mu(z) = \delta_{k\ell}.$$

For Gaussian μ, these are the normalized Hermite polynomials.

20.2.1 Information Exponent of a Scalar Function

Let $S \sim \mu$ and $g \in L^2(\mu)$. Then

$$g(s) = \sum_{k=0}^{\infty} a_k \psi_k(s), \qquad a_k = \mathbb{E}[g(S)\psi_k(S)].$$

Definition 10 (Information exponent)**.** *The* information exponent *of g with respect to μ is*

$$\ell_\mu(g) = \min\{k \geq 1 : a_k \neq 0\},$$

if such k exists, and $\ell_\mu(g) = +\infty$ otherwise.

It marks the degree of the lowest polynomial mode through which g correlates with its input.

Intuitively, $\ell_\mu(g) = 1$ means that g has a nontrivial linear component with respect to μ; $\ell_\mu(g) \geq 2$ means that g is "purely nonlinear" at the level of first-order correlation.

20.2.2 Teacher–Student Single-Index Models

For $x \in \mathbb{R}^d$, $x \sim \mathcal{D}_X$, consider

$$y = \phi(\langle w_*, x\rangle) + \xi,$$

where ξ is independent noise. Let μ_* be the law of $S = \langle w_*, x\rangle$ under $\mathcal{D}_X$, and $\{\psi_k^{(*)}\}$ the orthonormal polynomial system in $L^2(\mu_*)$. Writing

$$\phi(s) = \sum_{k=0}^{\infty} b_k \psi_k^{(*)}(s),$$

we define:

Definition 11 (Information exponent of the link)**.**

$$\ell_{\mathcal{D}_X}(\phi, w_*) = \ell_{\mu_*}(\phi) = \min\{k \geq 1 : b_k \neq 0\}.$$

When $\ell_{\mathcal{D}_X}(\phi, w_*) = 1$, the teacher signal contains a nontrivial linear component; when $\ell_{\mathcal{D}_X}(\phi, w_*) \geq 2$, the signal is purely nonlinear. Existing results for shallow models show that this exponent controls the difficulty of learning w_* by gradient methods.

20.3 Genericity as a Structural Property of Functions

We now place the "presence of linear modes" into a genericity framework. Informally, a property is generic if it holds for "most" functions: either almost everywhere (measure-theoretic) or on a residual (topologically large) subset of a function space. Here, genericity is tied to:

- the local Taylor structure of a function (jets),
- and its stability under smooth coordinate changes.

20.3.1 Genericity via Taylor Jets

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a smooth function, and let $x_0 \in \mathbb{R}^d$. The k-jet of f at x_0 is defined as

$$J^k f(x_0) = \{\partial^\alpha f(x_0) : |\alpha| \leq k\}.$$

We say that f is *locally generic at* x_0 if its Taylor expansion at x_0 is non-degenerate in the sense that the coefficients corresponding to the relevant low-order monomials (e.g., linear, quadratic terms) do not vanish under accidental algebraic cancellations.

More precisely, let $\mathcal{P}$ be a polynomial condition on the jet $J^k f(x_0)$. Then f is generic at x_0 if

$$\mathcal{P}(J^k f(x_0)) \neq 0$$

for all but a set of points x_0 of measure zero. This notion follows the classical theory of jet transversality and singularity theory [183, 1].

In this language, the vanishing of all linear coefficients of f at x_0 is a *non-generic* condition: it lies on an algebraic subvariety in jet space. Thus, for a generic smooth function, the linear term is present almost everywhere unless excluded by a symmetry constraint.

20.3.2 Thom–Mather Transversality and Structural Stability

The mathematical foundation of genericity is provided by Thom's transversality theorem. Let $C^r(M, N)$ denote the space of r-times continuously differentiable maps between smooth manifolds. A property is generic if it holds on a residual (countable intersection of open dense) subset of $C^r(M, N)$.

Thom's Transversality Theorem (informal). *For a submanifold S of the jet space $J^k(M, N)$, the set of functions whose k-jets are transverse to S is open and dense in $C^r(M, N)$.*

As a consequence, *degenerate Taylor structures are non-generic*. Structural stability theory further shows that generic properties persist under small C^r perturbations of the function as shown by Thom and Abraham.

This leads to the fundamental implication:

> *Generic properties are precisely those that are stable under small smooth perturbations of the function and of the variables.*

We conjecture that this implication is key for learning theory since stability has been shown – in a specific, formal framework – to be equivalent to learnability (see Niyogi, Poggio, Rifkin, Mukherjee) [157, 132]. In particular, we *conjecture that non-generic functions do not have learning stability*. This makes for an interesting research project.

It is obvious that the presence of a nonzero linear coefficient (in some coordinate system) is generic, while its systematic vanishing across layers and directions corresponds to a fine-tuned, non-generic configuration.

20.3.3 Invariance Under Smooth Coordinate Transformations

Let $T : \mathbb{R}^d \to \mathbb{R}^d$ be a local diffeomorphism. A property $\mathcal{G}(f)$ is said to be *coordinate-invariant generic* if

$$\mathcal{G}(f) = \mathcal{G}(f \circ T)$$

for all T sufficiently close to the identity in the C^r topology.

This expresses precisely the idea that:

Generic properties cannot depend on accidental choices of coordinate systems, means, or scalings of the variables.

Such invariance is implicit in structural stability theory and also appears in statistical decision theory under the name of *local reparameterization invariance* [106]. In classical approximation theory, approximation classes such as Sobolev or Besov spaces are invariant under affine transformations of the domain [46]; in representation learning, related ideas appear in the theory of group-invariant representations.

From the perspective of learning theory, this suggests:

A function-theoretic property relevant to learnability must be generic and invariant under small smooth transformations of the input variables.

The information exponent $\ell_\mu(g)$ is such a candidate: for generic g (without enforced symmetries), one expects $\ell_\mu(g) = 1$ almost everywhere.

20.4 Deep Networks and the Effective Information Exponent

We now lift the scalar notion of information exponent to deep architectures and connect it to genericity.

Consider a deep network with layers

$$x_0 = x, \qquad x_{l+1} = f_l(x_l; \theta_l), \quad l = 0, \dots, L-1,$$

and output x_L. Each layer may decompose as

$$f_l(x_l; \theta_l) = h_l(x_l; \theta_l) + s_l(x_l),$$

where s_l is a possibly linear skip connection and h_l a nonlinear branch.

Let $\mathcal{D}_l$ be the distribution of x_l at initialization under $x \sim \mathcal{D}_X$ and random θ_l. For any direction $u_l \in \mathbb{S}^{d_l - 1}$, define the scalar channel

$$U_l = \langle u_l, x_l \rangle, \qquad V_l = \langle u_l, f_l(x_l; \theta_l) \rangle,$$

and the regression function

$$g_{l,u}(u) = \mathbb{E}[V_l \mid U_l = u].$$

Let $\mu_{l,u}$ be the law of U_l and $\{\psi_k^{(l,u)}\}$ its orthonormal polynomial basis in $L^2(\mu_{l,u})$, giving

$$g_{l,u}(u) = \sum_{k=0}^{\infty} c_{l,u,k} \psi_k^{(l,u)}(u).$$

Definition 12 (Layer-wise and effective information exponents)**.** *For each layer l,*

$$\ell_l = \inf_{\|u_l\|=1} \min\{k \geq 1 : c_{l,u,k} \neq 0\}.$$

The effective information exponent *of the architecture is*

$$\ell_{\text{eff}} = \min_{0 \leq l \leq L-1} \ell_l.$$

Intuitively, $\ell_l = 1$ means the layer retains a linear mode relative to its input distribution, while $\ell_l \geq 2$ means that all directions are purely nonlinear. Residual networks with identity skips typically have $\ell_{\text{eff}} = 1$; purely even activations without skips yield $\ell_{\text{eff}} \geq 2$.

Frequency doubling and spectral explosion. There is a subtle but crucial consequence of $\ell_l \geq 2$ in deep networks. If $\ell_l = 1$, a low-frequency component in the input can propagate as a low-frequency component in the output. However, if $\ell_l \geq 2$ (e.g., $z \mapsto z^2$), the operation involves a convolution in the frequency domain, effectively doubling the frequency (or complexity) of the signal at each layer. Stacking L such layers results in an exponential explosion of signal complexity, rendering the landscape "shattered" and chaotic. Maintaining $\ell_{\text{eff}} = 1$ prevents this explosion, preserving a coherent gradient highway.

Genericity viewpoint. For generic weights and generic input distributions, one expects $\ell_l = 1$ unless the architecture enforces a symmetry (e.g., even activations, antisymmetric constraints) that systematically kills the linear term. Thus, architectures with $\ell_{\text{eff}} \geq 2$ correspond to non-generic function classes from the standpoint of Taylor jets.

20.5 Formal Conjecture

Conjecture 1 (Effective information exponent and optimization)**.** *Let $\mathcal{D}_X$ be a distribution on $\mathbb{R}^d$ such that for every unit vector w the projection $\langle w, x\rangle$ has all moments finite. Consider deep networks of depth L and width $\text{poly}(d)$, trained by gradient descent or SGD on the population risk*

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)}[\ell(f_\theta(x), y)],$$

for targets f_* realizable by the same architecture.

Assume initialization ensures bounded forward/backward moments and well-defined ℓ_l at each layer. Let ℓ_{eff} be the effective information exponent at initialization.

(a) (Efficient regime: $\ell_{\text{eff}} = 1$.)
If at least one layer carries a nonzero degree–1 coefficient, then for appropriate learning rate, batch size, and initialization, gradient descent achieves excess risk

$$\mathcal{L}(\hat{\theta}) - \inf_\theta \mathcal{L}(\theta) \leq \varepsilon$$

in time $\text{poly}(d, L, 1/\varepsilon)$ with high probability.

(b) (Hard regime: $\ell_{\text{eff}} \geq 2$.)
If all layers suppress degree–1 modes and have no linear skips, then there exists a nontrivial subset of realizable targets such that any gradient-based algorithm with polynomially bounded resources either

- *requires super-polynomial time to reach error ε, or*
- *converges to a point whose risk remains bounded away from optimum.*

Interpretation. The conjecture states that efficient optimization in deep architectures requires at least one "linear information path" through the network. If every layer removes all degree–1 modes, gradients cannot consistently align with the target direction, and training becomes computationally hard. Architectures that enforce $\ell_{\text{eff}} \geq 2$ belong to a non-generic, structurally unstable regime from the viewpoint of Taylor jets and genericity.

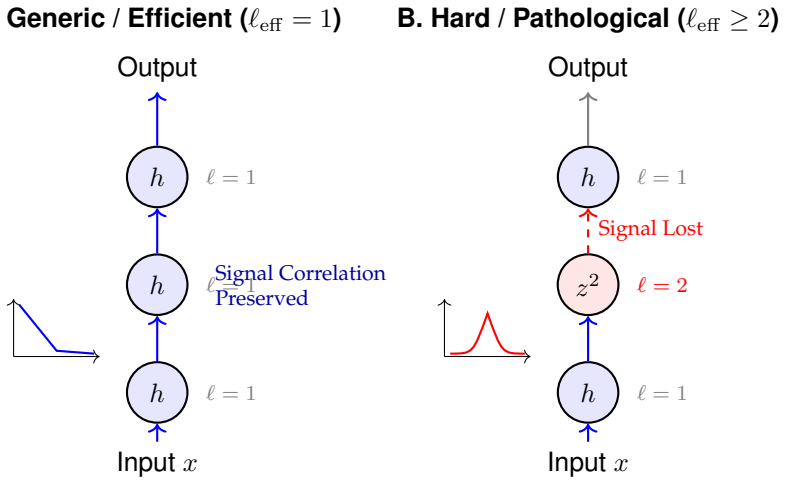


Figure 20.1: **The Genericity Conjecture.** (A) In a generic network ($\ell_{\text{eff}} = 1$), such as a ResNet or ReLU network, the linear component of the signal propagates through layers, creating a gradient highway. (B) If even a single layer suppresses the linear mode ($\ell \geq 2$, e.g., a pure z^2 layer), the correlation with the target direction is destroyed (orthogonality), breaking the gradient signal and making optimization exponentially hard.

20.6 Evidence and Partial Progress

20.6.1 Single-Index and GLM Results

In teacher–student models with Gaussian or sub-Gaussian inputs,

$$y = \phi(\langle w_*, x \rangle) + \xi,$$

one finds:

- When $\ell_{\mathcal{D}_X}(\phi, w_*) = 1$, gradient descent learns w_* efficiently (polynomial sample and time complexity).
- When $\ell_{\mathcal{D}_X}(\phi, w_*) \geq 2$, gradient descent dynamics lack correlation with w_* and may require super-polynomial samples or time.

This rigorously establishes the $\ell = 1$ versus $\ell \geq 2$ dichotomy for shallow models and suggests that the presence of a linear mode is a generic enabler of efficient optimization.

20.6.2 Quadratic and Polynomial Activations

Purely even activations such as $\sigma(z) = z^2$ produce $\ell_l \geq 2$ at every layer. Analyses of quadratic teachers show that gradient descent struggles or provably fails to recover w_* in polynomial time, despite realizability. Stacking such layers compounds the difficulty, consistent with the conjectured "hard regime." From the genericity viewpoint, imposing $\sigma(z) = z^2$ suppresses all linear terms and forces the network into a nongeneric submanifold of function space.

20.6.3 Residual Networks and Identity Skips

For $x_{l+1} = x_l + h_l(x_l; \theta_l)$ with small h_l, the identity skip provides a perfectly linear mapping. Results on *dynamical isometry* show that such residual structures maintain Jacobian spectra near 1,

avoiding vanishing or exploding gradients. Empirically and theoretically, ResNets remain trainable at extreme depth, whereas removing or distorting the skip causes instability. These facts directly correspond to maintaining $\ell_l = 1$ across layers and staying in a generic, structurally stable regime [141].

20.6.4 Nonlinear Attention

In nonlinear attention, weights of the form

$$\alpha_{ij} \propto \psi(\langle q_i, k_j \rangle)$$

depend on a scalar nonlinearity ψ. Analyses show that if the first polynomial coefficient of ψ vanishes, attention fails to memorize or generalize effectively; a nonzero linear term restores learning capability. Again, performance hinges on retaining the degree–1 component.

20.7 Research Program and Missing Ingredients

Although the conjecture is strongly motivated, a full proof requires substantial new theory.

What Is Missing

1. A precise and stable definition of ℓ_{eff} under training dynamics, not only at initialization.

2. A non-lazy theory of gradient dynamics showing how degree–1 modes govern convergence in deep architectures.

3. Lower-bound frameworks (low-degree polynomial or statistical query) connecting the absence of linear modes to computational hardness in a deep setting.

Possible Path for the Easy Regime ($\ell_{\text{eff}} = 1$)

1. Extend shallow GLM theory to deep networks by proving that degree–1 modes generate gradient signals aligned with target directions across layers.

2. Prove that under near-identity initialization (as in ResNets), these signals propagate without attenuation, yielding polynomial-time convergence.

Possible Path for the Hard Regime ($\ell_{\text{eff}} \geq 2$)

1. Model gradient descent iterates as low-degree polynomials in the data.

2. Show that with purely even activations, these polynomials lack degree–1 components.

3. Apply low-degree or SQ lower bounds to demonstrate super-polynomial hardness for generic targets.

20.8 Outlook

The effective information exponent provides a unifying bridge between functional approximation theory, genericity, and optimization dynamics. If proved, the conjecture would establish a precise criterion separating *architectures that can learn efficiently* from those that cannot. It would also formalize the widely observed empirical role of skip connections and linear residual paths in stabilizing deep learning, and connect it to classical notions of genericity and structural stability.

CHAPTER 21

Diffusion Models, Ill-Posed Inversion, and Generative Compression

What is the relation between diffusion models and ill-posed problems such as the inversion of the heat equation?

21.1 Diffusion as a Forward Process and Ill-Posed Inversion

The forward diffusion equation in $\mathbb{R}^d$,

$$\frac{\partial u}{\partial t} = \Delta u, \tag{21.1}$$

has the classical solution

$$u(x,t) = (G_t * u_0)(x), \tag{21.2}$$

where G_t is the Gaussian kernel of variance t and u_0 is the initial condition. In the Fourier domain,

$$\widehat{u}(\xi,t) = e^{-t\|\xi\|^2}\widehat{u_0}(\xi). \tag{21.3}$$

Thus diffusion exponentially suppresses high frequencies.

To invert this forward map, one would need

$$\widehat{u_0}(\xi) = e^{t\|\xi\|^2}\widehat{u}(\xi,t), \tag{21.4}$$

which blows up super-exponentially as $\|\xi\| \to \infty$. Any small measurement error in $\widehat{u}(\xi,t)$ is magnified without bound. In Hadamard's sense, the inverse operator is *ill-posed*: it is discontinuous and unstable with respect to perturbations of the data [185].

Figure 21.2 summarizes the contrast between:

- classical inversion of the diffusion operator (top row), which is ill-posed;
- probabilistic reverse diffusion based on a learned score (middle row), which is well-posed as a statistical inference problem;
- inpainting-based generative compression (bottom row), where diffusion models reconstruct full images from partial observations.

21.2 Probabilistic Reverse Diffusion and Learned Scores

Diffusion-based generative models (DDPMs, score-based SDE models) do not attempt to solve the inverse problem

$$u(\cdot, t) \mapsto u_0 \tag{21.5}$$

as an operator inversion. Instead, they solve a *statistical* inverse problem.

Let $p_t(x)$ denote the distribution of data after adding Gaussian noise of variance t. The key object is the score function

$$s_t(x) = \nabla_x \log p_t(x), \tag{21.6}$$

approximated by a neural network $s_\theta(x, t)$ trained via denoising score matching.

The reverse-time SDE associated with the forward diffusion and noise schedule (f, g) is

$$dx = \left(f(x,t) - g(t)^2\, s_t(x)\right) dt + g(t)\, d\bar{w}_t, \tag{21.7}$$

which generates samples from p_0 when integrated backward in time from an initial Gaussian distribution at large t. Each infinitesimal step of this reverse-time process can be interpreted as a (regularized) denoising step, approximating the conditional mean

$$\mathbb{E}[x_0 \mid x_t]. \tag{21.8}$$

This conditional expectation is a *well-posed* inference problem. From this viewpoint, diffusion models sidestep the ill-posedness of PDE inversion: they never invert the heat operator; instead, they perform Bayesian denoising in a high-dimensional image space.

21.3 Energy Landscape Viewpoint

A complementary way to understand forward and reverse diffusion is through the lens of *energy landscapes*. Let $p_t(x)$ denote the distribution of x_t at noise level t, and define the (time-dependent) energy

$$E_t(x) = -\log p_t(x). \tag{21.9}$$

At $t = 0$, p_0 is a complex, multimodal data distribution (natural images), and E_0 is a rough, highly structured energy landscape with many wells corresponding to modes of p_0. As t increases, the forward diffusion process x_t becomes closer to a Gaussian, and E_t becomes smoother and flatter; modes disappear and the energy landscape loses fine-scale structure.

Reverse diffusion can be interpreted as a process that *sharpens* the landscape: starting from an almost flat Gaussian energy at large t, successive reverse steps reconstruct the multi-well structure of E_0 by following the learned score $s_t(x) = -\nabla E_t(x)$ backward in time. The forward process irons out the energy landscape; the learned reverse process reconstructs it.

Figure 21.3 gives a schematic one-dimensional illustration of this picture: forward diffusion maps a sharp, multimodal density into a broad, nearly unimodal distribution; reverse diffusion restores the sharp modes.

21.4 Inpainting and Generative Compression

Because diffusion generative models learn the family of conditional densities $p(x_0 \mid x_t)$, they naturally support *inpainting*. Given a partially observed image x_{obs} and a mask M, inpainting corresponds to sampling

$$x_{\text{complete}} \sim p\big(x \,\big|\, (1 - M) \odot x = (1 - M) \odot x_{\text{obs}}\big), \tag{21.10}$$

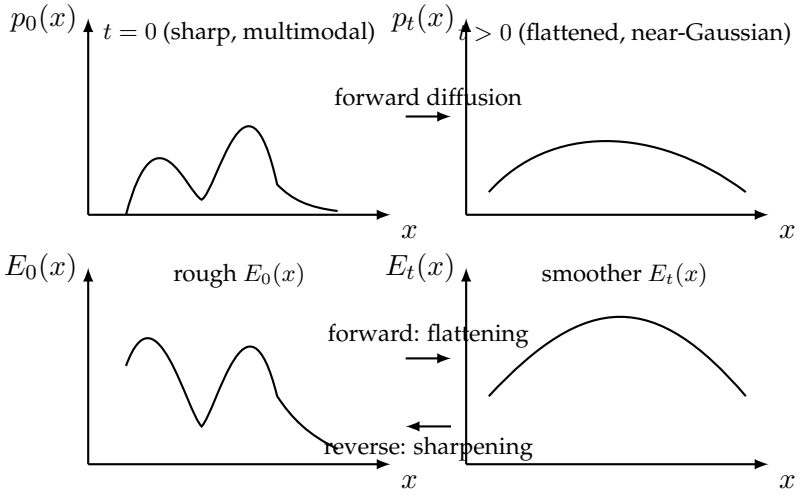


Figure 21.1: Energy landscape viewpoint in one dimension. **Top:** Forward diffusion maps a sharp, multimodal data distribution $p_0(x)$ into a smoother, nearly unimodal $p_t(x)$ as t increases. **Bottom:** The corresponding energies $E_t(x) = -\log p_t(x)$ evolve from a rough landscape with many wells (modes of p_0) to a smoother, flatter landscape for $t > 0$. Reverse diffusion, driven by the learned score $s_t(x) = \nabla \log p_t(x)$, can be viewed as sharpening this energy landscape back toward $E_0(x)$.

where $\odot$ denotes pointwise multiplication. During sampling, the observed pixels are kept fixed, while the masked region is repeatedly resampled from the conditional score. The final result is a coherent hallucination of the occluded region, aligned with the global statistics of the data. The bottom row of Figure 21.2 illustrates this inpainting-based view.

This inpainting perspective leads directly to *generative compression*. The central idea is that one can store only:

- a compact latent code z (e.g., from a vector-quantized encoder) and let a conditional diffusion model sample from $p(x \mid z)$, or
- a sparse subset of pixels $(1 - M) \odot x$ and let the diffusion model reconstruct x by inpainting the missing region $M \odot x$.

In both cases, a small amount of information, together with a powerful generative prior, suffices to reconstruct visually convincing images. This realizes the idea that "a few fragments plus a learned prior" can function as an extreme compression scheme.

Recent work [81, 114, 29, 142] demonstrates that diffusion-based decoders can outperform classical codecs at low bitrates, both in terms of perceptual quality and in standard distortion metrics.

21.5 Discussion

Diffusion generative models provide a mathematically consistent framework connecting:

- **Ill-posed PDE inversion:** the inverse of the diffusion operator is unstable and discontinuous;
- **Regularized statistical inference:** diffusion models recover x_0 via well-posed conditional expectations $\mathbb{E}[x_0 \mid x_t]$;

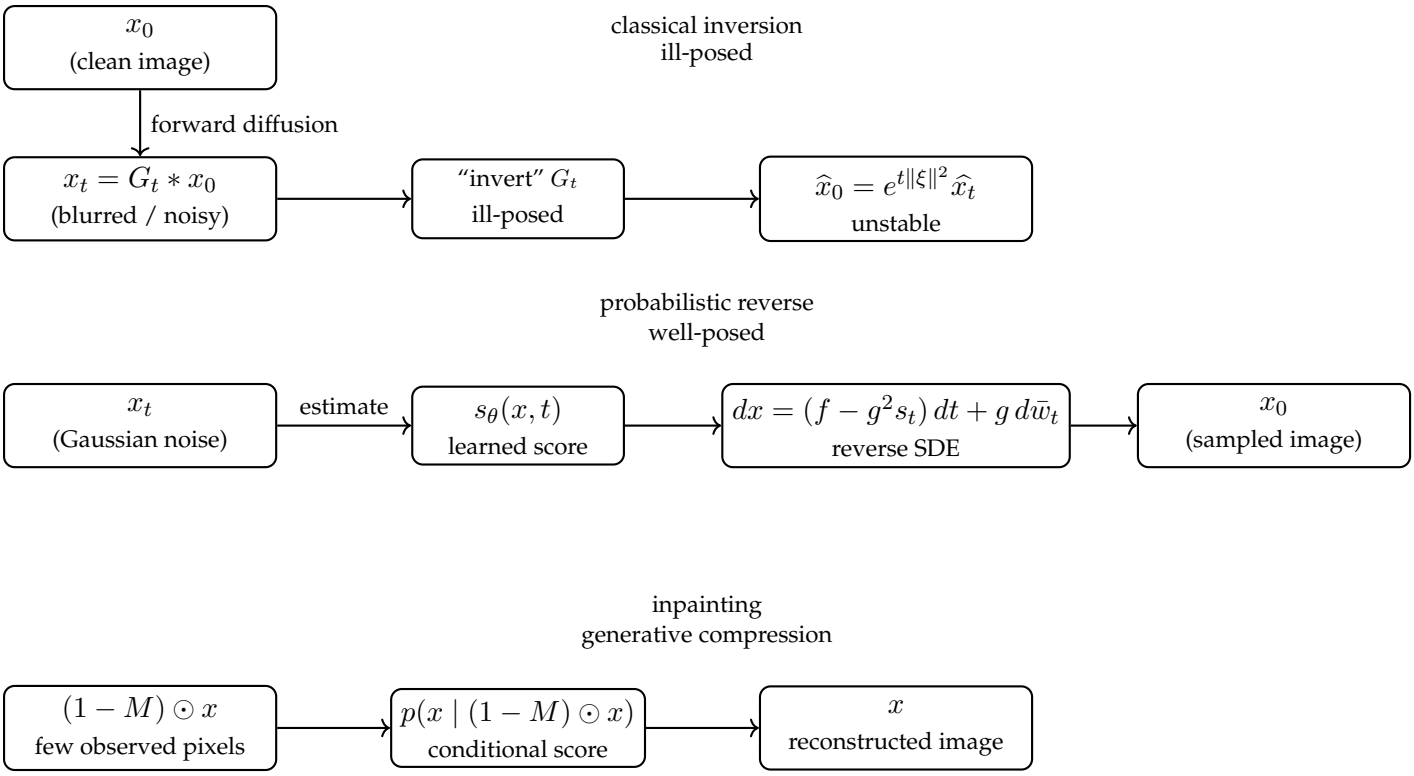


Figure 21.2: Three views of diffusion generative models. Top: classical inversion of the diffusion equation is ill-posed, because reversing Gaussian blur amplifies high-frequency noise exponentially. Middle: diffusion models avoid direct inversion by learning the score $\nabla \log p_t(x)$ and simulating a reverse-time SDE, which is statistically well-posed. Bottom: diffusion models perform inpainting, enabling generative compression by reconstructing an image from a small subset of observed pixels.

- **Energy landscape dynamics:** forward diffusion flattens and simplifies the energy landscape, while reverse diffusion sharpens it using the learned score;
- **Inpainting and generative compression:** a small latent code or a sparse subset of pixels can be expanded into a full image by the learned generative prior.

Figures 21.2 and 21.3 emphasize that diffusion models are not inverting the heat equation; they are implementing a particular form of probabilistic inference in a space of high-dimensional signals, with the diffusion SDE providing a convenient and stable parameterization of the path from noise to data.

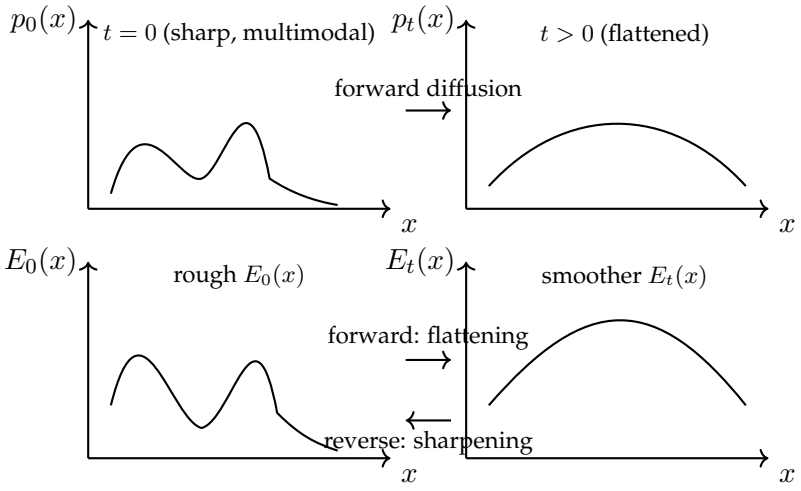


Figure 21.3: Energy landscape viewpoint in one dimension. Top: forward diffusion maps a sharp, multimodal data distribution $p_0(x)$ into a smoother, nearly unimodal $p_t(x)$ as t increases. Bottom: the corresponding energies $E_t(x) = -\log p_t(x)$ evolve from a rough landscape with many wells (modes of p_0) to a smoother, flatter landscape for $t > 0$. Reverse diffusion, driven by the learned score $s_t(x) = \nabla \log p_t(x)$, can be viewed as sharpening this energy landscape back toward $E_0(x)$.

CHAPTER 22

World Models Before Language

A central requirement for intelligent behavior is the ability to infer and predict the latent causes that govern the dynamics of the world. Long before the evolution of language, nervous systems had already discovered — with the emergency of neocortex – architectures capable of building world models *[69]. In this chapter we argue that evolution converged on mechanisms that are mathematically aligned with the framework developed in this book:* sparse compositionality *and* associative computation*. These mechanisms form the substrate upon which linguistic intelligence later emerged. The key claim of this chapter is that biological intelligence is best described as a* predictive, compositional, memory-augmented dynamical system*.*

22.1 Sparse Compositionality as the Structural Prior of Evolution

Sparse compositionality is the assumption that an efficiently computable function

$$f : \mathcal{X} \to \mathcal{Y}$$

admits a representation as a directed acyclic graph (DAG)

$$f = g_L \circ g_{L-1} \circ \cdots \circ g_1,$$

where each module g_ℓ acts on at most $k \ll d$ variables. This structural constraint reduces the effective complexity of f and allows efficient approximation, optimization, and generalization, as shown in previous chapters [129, 154].

From an evolutionary standpoint, sparse compositionality is advantageous because it implies:

1. **Hierarchical reuse**. Modules g_ℓ can be reused across different tasks (e.g., grasping, locomotion, object segmentation).
2. **Local learning**. Synaptic plasticity acts on local circuits with limited fan-in.
3. **Robustness**. Perturbations affect only a small part of the computational graph.
4. **Energy efficiency**. Local modules minimize metabolic cost.

Evolution therefore discovers sparse modular architectures for the same reasons that deep learning theory identifies them as optimal.

22.2 World Models as Predictive State-Space Systems

To model the temporal evolution of the world, an organism must learn and simulate latent dynamics:

$$z_{t+1} = F(z_t, x_t), \qquad \hat{x}_{t+1} = G(z_{t+1}),$$

where z_t denotes an internal latent state and x_t observable input [56]. Biologically, this corresponds to a division of labor:

- F is implemented in hippocampal–prefrontal loops and recurrent cortex.
- G corresponds to fast thalamocortical and sensorimotor pathways.

Such models support:

- prediction,
- reconstruction,
- planning,
- imagination (running the simulator offline to "dream awake"),
- credit assignment over time.

These properties make state-space models fundamentally more plausible than diffusion models, which require thousands of denoising iterations, global synchrony, and energetically impossible computation.

22.3 Associative Memory as a Computational Primitive

Sparse compositionality alone is insufficient: an intelligent organism must bind, store, and retrieve partial trajectories or scenes. This requires associative memory. Formally, consider a memory operator

$$M : \mathbb{R}^d \to \mathbb{R}^d,$$

such that $M(x)$ retrieves or reconstructs patterns close to x in a suitable metric.

Biological correlates include:

- CA3 recurrent networks functioning as attractor dynamics [82].
- Neocortical distributed memory with Hebbian and heterosynaptic updates [90].
- Hippocampal replay, reconstructing trajectories during sleep.

In the computational framework of this book, the combination of local modules and associative memory leads naturally to *Associative Turing Machines* (ATMs), which are Turing-complete architectures built from local update rules and content-addressable memory.

22.4 Hippocampal Replay as Approximate Inference

The hippocampal formation performs "preplay" and "replay" of trajectories [53]. Formally, this can be interpreted as approximate inference in a latent dynamical model.

If $p(z_{1:T} \mid x_{1:T})$ denotes the posterior over latent trajectories, then replay implements a sampling or MAP approximation to this posterior. In particular, sharp, temporally compressed sequences during sleep correspond to short iterative inference iterations in a predictive state-space model. Functionally, this decoupling allows the world model to run in "unconstrained" mode—effectively performing data augmentation or adversarial training on the internal model.

This mechanism provides:

1. credit assignment over long timescales,
2. structure learning of transition graphs,
3. reconstruction of incomplete trajectories,
4. stabilization of attractors in cortical memory.

22.5 What Evolution Discovered Before Language

Language requires compositional semantics, symbolic structure, and recursive manipulation. These capacities did not appear with language itself, but were already present in the underlying architecture for world modeling.

We identify four pre-linguistic computational primitives discovered by evolution:

22.5.1 Predictive Physical Inference

Animals infer Newtonian-like latent variables:

$$x_{t+1} \approx x_t + v_t, \qquad v_{t+1} \approx v_t + a_t,$$

providing intuitive physics [16]. A primate leaping between branches does not calculate Newtonian mechanics using symbols; it runs a continuous internal simulation of the trajectory.

22.5.2 Social and Causal Modeling

Inference of hidden intentions, threats, or affordances is a form of latent-state estimation. This "Naive Psychology" requires running a simulation of another agent's simulation.

22.5.3 Compositional Perception and Action

Hierarchical modules for vision and motor control provide a structural prior equivalent to compositional sparsity.

22.5.4 Associative Memory for Episodes and Scenes

Hippocampal mechanisms allow storage of high-dimensional relational structure, anticipating the later role of language.

22.6 Language as an Overlay on a Pre-Existing Architecture

Language provides a *communication interface* for the representational and inferential machinery described above. We may view language not as the CPU of intelligence, but as the rendering interface. Consider a video game engine: the core logic handles geometry, physics, and collisions (the World Model), while the on-screen dialogue is merely a readout of that underlying state.

Current LLMs are akin to learning the game engine solely by reading the subtitles. They predict the next word fluently, but they often hallucinate because they lack the underlying physics engine that ensures consistency [107].

From a computational viewpoint:

- syntax reuses hierarchical compositional circuits,
- semantics reuses latent variable inference,
- pragmatics reuses world models and planning,
- narrative and reasoning reuse hippocampal reconstruction.

Thus the core claim is:

> *Language did not create compositional intelligence; it exploited a pre-existing, evolutionarily ancient compositional and predictive world-model architecture.*

22.7 Conclusion

The biological precursors of intelligence—predictive state-space dynamics, compositional computation, and associative memory—are aligned with the computational principles of sparse compositionality and ATMs developed throughout this book. This chapter places these principles in an evolutionary context: world models arose millions of years before language, and language is best viewed as an overlay on an already compositional, predictive, and memory-augmented architecture. To build true general intelligence, we must first teach machines to "dream" the world—to run the simulator offline—before they can meaningfully speak about it.

CHAPTER 23

The Hippocampal Scaffold and Compositional Sparsity

What is the relation between memory models of the hippocampus and compositional sparsity? This chapter argues that the hippocampus acts as a "scaffold builder," creating a sparse, structured graph of pointers that organizes raw experiences into a compositionally sparse format.

23.1 Introduction: The Memory Palace

To understand the mathematical definitions of x_t, y_t, and z_t, let us use an ancient mnemonic device: the **Memory Palace** (or Method of Loci). In this technique, a person memorizes items by mentally placing them in specific locations within a familiar structure, such as their home.

Imagine an agent walking through a house to store memories. At each time step t, the experience has three distinct components:

1. **The Latent Structure (z_t):** The agent is in the "Kitchen." This is an abstract, relational state. The agent knows that "Kitchen" connects to "Hallway" and "Dining Room," even if the lights are off. This spatial map is the *scaffold*.

2. **The Sensory Input (x_t):** The agent sees a specific view: "White tiles, morning light, smell of coffee." This is the high-dimensional, noisy sensory data that signals the state.

3. **The Episodic Target (y_t):** The agent decides to store a specific memory here: "My keys are on the counter." This is the content to be associated with the location.

The job of the hippocampus is to bind these distinct elements together. It must take the sensory input (x_t) and the latent context (z_t), create a unique "address" or pointer (h_t), and link it to the content (y_t). By linking these addresses over time (Kitchen $\rightarrow$ Hallway), it builds a graph—a scaffold—that represents the structure of the world.

This chapter formalizes this process. We show that by constructing these sparse indices, the hippocampus effectively projects the world into a **compositionally sparse representation**—a format that allows the cortex to learn complex functions by breaking them down into local, manageable interactions.

23.2 The Variables of Experience

We formalize the data stream as a sequence of tuples $e_t = (x_t, y_t, z_t)$.

Latent State $z_t \in \mathcal{Z}$: The hidden "ground truth" of the agent's position in the relational graph (e.g., $z_t =$ "Kitchen"). Note that z_t is not directly observable; it must be inferred from cues or path integration. Crucially, the transitions between z's are sparse (you can only walk from the Kitchen to the Hallway, not to the Roof instantly).

Observable Input $x_t \in \mathcal{X}$: The sensory features available to the agent (e.g., a pixel image of the stove). While x_t is high-dimensional, it is a function of the low-dimensional state z_t.

Target Content $y_t \in \mathcal{Y}$: The information to be remembered or predicted (e.g., the location of the keys, or the image of the next room).

The fundamental problem is that raw inputs x_t are often highly correlated and overlapping (the white tiles in the kitchen look like the white tiles in the bathroom). To store memories without interference (catastrophic forgetting), the brain needs a system to "separate" these patterns.

23.3 The Indexing Mechanism: Pattern Separation

The hippocampus (specifically the Dentate Gyrus) acts as a **Hashtag Generator**. It takes the confusingly similar sensory inputs and assigns them distinct, sparse codes.

Let us define a feature map $g(x_t, z_t)$ that combines sensory input and latent context. We generate a **Hippocampal Index** h_t via a random projection followed by a "Winner-Take-All" nonlinearity (keeping only the top k neurons active).

Definition 13 (The Hippocampal Index)**.** *Let $R \in \mathbb{R}^{N \times d}$ be a fixed random matrix (representing fixed synaptic weights). The index for episode t is:*

$$h_t = \text{TopK}(R\, g(x_t, z_t)) \in \{0, 1\}^N$$

where $\|h_t\|_0 = k \ll N$.

23.3.1 Why this works

This mechanism is effectively **Locality-Sensitive Hashing**.

- If $z_t \approx z_s$ (same room), the indices h_t and h_s will overlap partially.
- If $z_t \neq z_s$ (different rooms), the indices will be nearly orthogonal (pattern separation).

This orthogonality is critical. It allows the memory system to store the "Keys in Kitchen" (y_t) at address h_t without overwriting the "Towel in Bathroom" (y_s) at address h_s, even if the white tiles (x_t, x_s) look similar.

23.4 Building the Scaffold Graph

As the agent moves through the Memory Palace (Kitchen $\to$ Hallway $\to$ Bedroom), it generates a sequence of indices: $h_1, h_2, h_3 \dots$.

Because these indices are generated by the underlying physics of the world (the adjacency of rooms), the transitions between them define a graph.

Definition 14 (The Hippocampal Scaffold)**.** *The scaffold is the graph $\mathcal{G} = (\mathcal{H}, \mathcal{E})$ where the nodes are the stored indices $\{h_t\}$ and edges represent observed temporal transitions:*

$$(h_t, h_{t+1}) \in \mathcal{E} \iff \textit{Agent moved from } z_t \textit{ to } z_{t+1}.$$

This graph is a discrete, sparse approximation of the continuous world. It captures the **topology** of the latent space $\mathcal{Z}$. By replaying sequences from this graph (e.g., during sleep), the hippocampus teaches the cortex the "layout of the house."

23.5 The Connection to Compositional Sparsity

We can now answer the central question: *What function is the cortex learning, and why does the hippocampus make it sparse?*

The cortex's goal is to learn a predictive model of the world's dynamics—a function F that predicts the future state from the current state.

23.5.1 The Dense Trap of Sensory Learning

Suppose the cortex attempts to learn this transition function directly on the raw sensory inputs x_t (e.g., pixels):

$$x_{t+1} \approx F_{\text{dense}}(x_t)$$

This is a computationally difficult task. The mapping between two arbitrary video frames (e.g., turning a head) involves complex, global pixel shifts. The function F_{dense} is likely **dense** (every output pixel depends on many input pixels) and does not inherently possess a sparse compositional structure. Learning this requires massive amounts of data to avoid the curse of dimensionality.

23.5.2 The Sparse Solution via the Scaffold

The hippocampus solves this by providing the cortex with the index h_t instead of x_t. The cortex now learns a transition function on the indices:

$$h_{t+1} \approx G_{\text{sparse}}(h_t)$$

Crucially, the function G_{sparse} is **compositionally sparse** by construction.

Proposition 6 (Graph Sparsity implies Function Sparsity)**.** *The transition function G_{sparse} is defined by the adjacency matrix of the Hippocampal Scaffold graph. Because the physical world has local topology (e.g., the Kitchen connects only to the Hallway, not the Moon), this graph is sparse.*

The Mechanism:

1. **Locality (Node Sparsity):** To predict the next index h_{t+1}, the function only needs to consider the immediate neighbors of h_t in the scaffold graph. The "constituent function" for this step has a bounded fan-in (limited by the number of doors in a room).

2. **Composition (Depth):** To predict the state T steps in the future, the cortex computes the composition:
$$h_{t+T} = \underbrace{G \circ G \circ \cdots \circ G}_{T \text{ times}}(h_t)$$
This is precisely the definition of a compositionally sparse function: a deep hierarchy of simple, local functions.

Summary: The hippocampus effectively "diagonalizes" the complex dynamics of the world. By mapping sensory experiences (x) onto a topological scaffold (h), it transforms a dense, intractable learning problem into a sparse, compositional graph-traversal problem that the cortex can learn efficiently.

23.6 The Associative Turing Machine

Finally, we can view this system through the lens of computation. A Turing machine needs a tape (memory) and a head (pointer) to read/write.

1. **The Pointers:** The indices h_t act as pointers. They are computationally cheap to create (just a matrix multiplication and a threshold).
2. **The Memory:** The synaptic weights binding h_t to y_t act as the storage tape.
3. **The "Head" Movement:** The transition from $h_t \to h_{t+1}$ (via the scaffold graph) moves the "read head" to the next relevant memory address.

This system allows for **O(1) access** to relevant memories. Unlike a standard neural network that might have to search its entire weight space to resolve a conflict, the hippocampal system jumps directly to the specific sparse address h_t relevant to the current context z_t.

23.7 The Cortical Transfer: Systems Consolidation

If the hippocampus provides the sparse indices h_t required to avoid interference, a natural question arises: *Does the cortex always depend on the hippocampus to retrieve memories?* The phenomenon of systems consolidation—where memories eventually become independent of the hippocampus—suggests otherwise.

In the framework of compositional sparsity, we interpret consolidation not as a transfer of data, but as a **change in the basis of representation**.

23.7.1 From Orthogonal Indexing to Manifold Learning

The hippocampus and cortex optimize for opposing objectives regarding the latent state z_t (e.g., "The Kitchen").

1. **Hippocampus (Separation):** It treats every visit to the Kitchen as a unique event. It assigns orthogonal keys $h_{t_1} \perp h_{t_2}$ to separate Monday's visit from Tuesday's visit, ensuring that the specific details y_{t_1} and y_{t_2} do not overwrite each other.
2. **Cortex (Structure):** It aims to learn the invariant structure of the world. It seeks to discover that h_{t_1} and h_{t_2} actually represent the same underlying state z.

During replay (e.g., sleep), the hippocampus presents the sequence of orthogonal indices to the cortex. The cortex uses these indices not as addresses to be memorized, but as **training targets** to learn a stable, low-dimensional mapping from sensory noise x_t to the latent manifold $\hat{z}_t$.

23.7.2 Collapsing the Keys

We can model consolidation as a "collapse" of dimensions. The cortex slowly learns a projection $\phi : \mathcal{X} \to \mathcal{Z}$ such that structurally similar inputs map to the same representation, regardless of their temporal index.

$$\text{If Topology}(h_{t_1}) \approx \text{Topology}(h_{t_2}) \implies \phi(x_{t_1}) \approx \phi(x_{t_2})$$

While the hippocampus maintains distinct pointers for every instance, the cortex averages over these instances to form a **Schema**. It effectively collapses the orthogonal axes of the hippocampal space into the lower-dimensional manifold of the "true" latent variables.

23.7.3 Retrieval Without the Scaffold

Once this cortical mapping is learned, the retrieval pathway fundamentally changes. The system no longer requires the specific index h_t to access the memory.

Early Retrieval (Hippocampal-Dependent):

$$x_t \xrightarrow{\text{fast}} \text{Hippocampus}(h_t) \xrightarrow{\text{lookup}} y_t$$

Access depends on the specific hash key h_t. If the hippocampus is damaged, the link between the sensory cue and the target is broken.

Late Retrieval (Cortical-Independent):

$$x_t \xrightarrow{\text{slow}} \text{Cortex}(\hat{z}_t) \xrightarrow{\text{schema}} y_t$$

The cortex infers the latent state $\hat{z}_t$ directly from x_t. The specific index h_t is bypassed.

23.7.4 The Semantic Trade-off

This independence comes at a cost. When the cortex retrieves a memory without the hippocampal key h_t, it loses the "orthogonalizing" power that kept that specific instance unique.

Consequently, the retrieved memory ceases to be episodic ("The specific time I dropped eggs on the kitchen tiles") and becomes semantic ("The kitchen has tiles"). In the language of this chapter, the cortex has successfully learned the **compositional function** G_{sparse} of the environment, but in doing so, it has discarded the specific training data h_t that made that learning possible. The scaffold is removed, leaving only the building.

23.8 A Unifying Computational Claim

Despite their apparent diversity, most influential theories of hippocampal function converge on a single underlying computation. This chapter advances the following unifying claim:

> *The hippocampus implements a mechanism that converts dense, correlated experience into a sparse, indexable structure whose topology reflects the latent relational structure of the world.*

This claim reframes classical hippocampal theories not as competing explanations, but as complementary descriptions of different aspects of the same computational role.

Cognitive Maps. Cognitive map theories emphasize the hippocampus as a representation of spatial structure. In the present framework, spatial maps are a special case in which the latent state space $\mathcal{Z}$ corresponds to physical locations. The hippocampal scaffold graph explicitly represents adjacency in this space, generalizing the notion of a map beyond physical navigation to arbitrary relational domains.

Pattern Separation. Pattern separation theories focus on the hippocampus's ability to orthogonalize similar inputs. Here, pattern separation is understood as an implementation constraint required to generate sparse, non-interfering indices. Orthogonalization is not the goal of the computation, but the means by which stable indexing and graph construction become possible.

Indexing and Pointer-Based Memory. Indexing theories describe the hippocampus as storing pointers to distributed cortical representations. In the present framework, these pointers are realized as explicit sparse indices that support constant-time access and participate in structured transitions. Indexing is thus elevated from a metaphor to a concrete computational primitive.

Complementary Learning Systems. Complementary Learning Systems theory emphasizes the division between fast hippocampal learning and slow cortical abstraction. Within the present framework, this division arises naturally: sparse indexing enables rapid, interference-free storage, while the resulting scaffold provides the training signal required for cortical learning of shared structure and compositional rules.

Predictive and Successor Representations. Theories that view the hippocampus as encoding predictive or successor representations are recovered as consequences of traversal and composition on the scaffold graph. Long-horizon predictions correspond to repeated application of local transitions, rather than to a distinct representational format.

Interpretation. Viewed through this lens, the hippocampus is neither primarily a memory store nor a spatial navigator. It is a structural compiler: a system that transforms raw experience into a sparse, relational representation that renders learning tractable. The various classical theories of hippocampal function describe different projections of this single computation, shaped by experimental paradigm and level of analysis.

This unifying perspective allows the hippocampus to be situated naturally within a broader theory of compositional learning, where its primary role is to impose structure on experience so that downstream systems can discover and exploit it.

23.9 Relation to Existing Theories of the Hippocampus

The computational role of the hippocampus proposed in this chapter does not contradict classical theories of hippocampal function. Rather, it unifies and sharpens them by identifying a single organizing principle: the construction of a sparse, compositional scaffold that transforms dense experience into a learnable structure. In this section we relate the present framework to several influential theories and clarify how each appears as a special case.

23.9.1 Cognitive Map Theory as Scaffold Construction

The classical cognitive map theory posits that the hippocampus represents the relational structure of space, enabling navigation, shortcutting, and planning. In our framework, this idea is recovered as a special case of scaffold construction over a latent state space $\mathcal{Z}$.

The Hippocampal Scaffold graph $\mathcal{G} = (\mathcal{H}, \mathcal{E})$ generalizes spatial cognitive maps in three ways:

1. The nodes represent sparse indices h_t rather than physical locations.
2. The edges encode adjacency in an abstract latent space, not necessarily physical space.
3. The resulting graph supports compositional prediction through repeated application of a local transition function.

Thus, spatial navigation is only one instance of a more general computation: graph traversal in a sparse index space induced by experience. The hippocampus is not specialized for space, but for constructing graphs over latent structure.

23.9.2 Marr's Theory and Pattern Separation

Marr's theory emphasized the role of the dentate gyrus in pattern separation and the CA3 region in autoassociative storage. The hippocampal index

$$h_t = \mathrm{TopK}(Rg(x_t, z_t))$$

provides a precise computational realization of this proposal.

Random projections implement fixed, untrained connectivity, while the TopK nonlinearity enforces extreme sparsity. This guarantees that similar sensory inputs are mapped to nearly orthogonal indices unless they share latent context. Unlike Marr's original formulation, however, the present framework emphasizes that pattern separation is not an end in itself: it is required to construct a scaffold on which compositional learning becomes possible.

23.9.3 Indexing Theory and Pointer-Based Memory

Indexing theories describe the hippocampus as a system that stores pointers to distributed cortical representations. In the present framework, hippocampal indices are explicit, computable pointers with the following properties:

- They are sparse and approximately orthogonal.
- They are content-addressable through sensory input.
- They participate in a structured transition graph.

This makes indexing theory algorithmic rather than metaphorical. Indices are not symbolic labels but random-access addresses supporting constant-time retrieval and structured composition.

23.9.4 Complementary Learning Systems as a Change of Basis

Complementary Learning Systems theory distinguishes fast hippocampal learning from slow cortical learning, and episodic memory from semantic knowledge. In the present framework, this distinction is reinterpreted as a change of representational basis.

The hippocampus operates in an orthogonal, instance-specific basis indexed by h_t. The cortex gradually learns a low-dimensional manifold representation $\hat{z}_t$ by averaging over many such indices during replay. Consolidation is therefore not the transfer of stored data, but the collapse of an overcomplete coordinate system into a structured latent representation.

23.9.5 Successor Representations and Predictive Maps

Recent work has interpreted hippocampal representations as predictive maps encoding expected future states. In the present framework, such representations arise naturally from repeated composition of the sparse transition function

$$h_{t+1} \approx G_{\text{sparse}}(h_t).$$

Higher-order predictions correspond to powers of G_{sparse}, making successor-like representations a consequence of scaffold traversal rather than a separate encoding principle.

23.10 A Learnability Consequence of the Scaffold

The central computational claim of this chapter is not merely that the hippocampus stores memories, but that it imposes a structural constraint on the learning problem faced by cortex. Specifically, by organizing experience into a sparse scaffold graph, the hippocampus transforms an otherwise dense and ill-posed prediction problem into a compositionally sparse one. This claim admits a learning-theoretic interpretation.

Theorem 13 (Scaffold-Induced Learnability, Informal)**.** *Let $\mathcal{G} = (\mathcal{H}, \mathcal{E})$ be the hippocampal scaffold graph constructed from experience, with bounded degree d and mixing time τ. Let G_{sparse} denote the transition operator induced by local moves on $\mathcal{G}$. Then the family of functions representable by compositions of G_{sparse} forms a compositionally sparse function class whose effective sample complexity depends polynomially on d and τ, and is independent of the ambient dimensionality of the raw sensory input x_t.*

Interpretive Sketch. Because $\mathcal{G}$ has bounded degree, each local transition depends only on a constant-size neighborhood in the scaffold. Long-horizon predictions correspond to compositions of these local transitions along paths in the graph. As a result, the complexity of the induced prediction problem is governed by the topology of $\mathcal{G}$ rather than by the dimensionality of the sensory space from which the graph was constructed. Standard results on sparse compositional function classes imply that generalization can be achieved without incurring the curse of dimensionality associated with direct learning on x_t. □

Interpretation. This result should not be read as a claim that the hippocampus itself learns a compact parametric model. Rather, it formalizes the idea that the hippocampus reshapes the data distribution seen by cortex. By enumerating local constituent transitions in a sparse, indexable form, it converts dense sensory prediction into a graph-based compositional problem for which abstraction and generalization become possible. The theorem thus characterizes the *enabling role* of the scaffold: it makes cortical learning tractable, but does not perform that learning itself.

23.11 Beyond the Scaffold: Cortical Abstraction

If the hippocampus already captures the constituent functions g_i of a compositional mapping, why does the cortex need to "internalize" them? The answer lies in the difference between **Interpolation** and **Extrapolation**.

The Hippocampus: Local constituent learning

The hippocampal indices h_t are essentially **Nearest-Neighbor** lookup points. It learns constituent functions that are tied to specific coordinates in the scaffold.

$$f_{HPC}(x) = \sum_t y_t K(h_t, \text{proj}(x)) \tag{23.1}$$

where K is a kernel that is only non-zero in the immediate vicinity of a stored episode. This is "compositional" but *fragile*; if you are in a slightly different part of the room where no h_t was generated, the system fails.

The Cortex: Discovering Global Symmetries

The cortex does what the hippocampus cannot: it identifies that many different constituent functions in the scaffold are actually **instances of the same rule**.

In the language of group theory and invariance:

- **Translation Invariance:** The hippocampus learns how to navigate the Kitchen (z_1) and the Bedroom (z_2) as two separate graphs.
- **The Cortical Move:** The cortex notices that the "physics" of moving in the Kitchen is identical to the "physics" of moving in the Bedroom. It extracts the **Abstract Operator** T (the rule of 2D space) that is common to all scaffolds.

23.11.1 From Pointers to Generative Models

While the hippocampus uses the scaffold to point to data, the cortex transforms the scaffold into a **Continuous Generative Model**.

Definition 15 (Structural Generalization)**.** *Let $\{G_i\}$ be the set of sparse transition functions learned by the hippocampus for different environments. The cortex learns a meta-function $\mathcal{F}$ such that:*

$$\mathcal{F} = argmin_{\Phi} \sum_i \|G_i - \Phi(S_i)\| \tag{23.2}$$

where S_i represents the structural invariants (the "laws of the palace").

23.11.2 The Result: Zero-Shot Navigation

Because the cortex has internalised the "laws" beyond the specific hippocampal pointers, it can perform **Zero-Shot Inference**.

Hippocampus: Can only navigate the Palace because it has a map of every room.

Cortex: Can navigate a *new* house it has never seen before, because it understands the "compositional grammar" of houses (doors lead to hallways, floors are flat).

23.11.3 The Functional Hand-off

The hippocampus learns the **Episodic Composition** (The "What" and "Where" of today). The cortex learns the **Structural Composition** (The "How" of the world). The cortex doesn't just copy the hippocampus; it "distills" the scaffold, throwing away the specific nodes (h_t) to keep the underlying logic (G).

23.11.4 Transformers as a Silicon Analog to the Hippocampal-Cortical Circuit

The modern Transformer architecture can be viewed as a computational implementation of the theories discussed in this chapter. Specifically, the division between **Attention** and **Feed-Forward Networks (FFNs)** mirrors the division between the Hippocampal Index and the Cortical Schema.

Self-Attention as the Hippocampal Index

The self-attention mechanism performs a dynamic, sparse-like lookup that is mathematically analogous to the hippocampal pointer system. Given a query Q, key K, and value V:

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{23.3}$$

In this framework:

- The **Attention Matrix** $\text{Softmax}(QK^T)$ represents the **Hippocampal Index** h_t. It creates a temporary, context-dependent "address" that links the current state to past experiences.
- The **Values** V represent the **Episodic Targets** y_t.

Like the hippocampus, self-attention excels at **In-Context Learning (ICL)**, allowing the model to grasp new "scaffolds" (e.g., a few-shot prompt) instantly.

Feed-Forward Networks as the Cortical Manifold

The FFNs, which reside in the deeper layers of the Transformer, act as the **Long-Term Memory** or the Neocortex.

$$FFN(x) = \sigma(xW_1 + b_1)W_2 + b_2 \tag{23.4}$$

Research into *Linear Layers as Semantic Value Substitutes* suggests that FFNs store the "Compositional Invariants" of language and logic. While attention handles the *episodic* context (the specific variables of the prompt), the FFNs store the *semantic* structure (the grammar and world facts).

The Failure of Scaffold Removal in Current AI

The primary divergence between the Transformer and the biological system is the lack of **Consolidation-led Discarding**.

In the biological model, the scaffold is removed once the cortex internalizes the mapping. In a standard Transformer:

- **Infinite Scaffold Dependency:** To maintain its logic, the Transformer must attend to its past tokens at every step. It cannot "collapse" the context window into its weights during inference.
- **Static Weights:** Unlike the cortex, which learns from the hippocampal "teacher" during offline replay (sleep), a Transformer's FFN weights are frozen after training.

Toward a "Consolidating" Transformer

To replicate the "Scaffold Removal" described in this chapter, we look toward emerging architectures:

- **Recurrent Memory Transformers (RMT):** These attempt to compress the context into a "summary token," a step toward semanticization.
- **Fast Weight Programmers:** These architectures allow the "Attention" layer to actually modify the "FFN" weights in real-time, simulating the hand-off from episodic pointers to structural internalisation.

The Transformer provides the most robust evidence yet for the **Compositional Sparsity** framework. It demonstrates that by using sparse-like indices (Attention) to navigate a manifold of knowledge (FFNs), we can achieve human-level reasoning. However, true AI "autonomy" will require the ability to discard the scaffold—moving from a system that *references* its context to one that *internalizes* its structure. in other words, the Transformer is currently "addicted" to its scaffold.

23.12 Why the Cortex Is Still Needed: From Enumerated Constituents to Parametric Composition

In this section I repeat the arguments of the previous section in different words, because this is a potential key to a new class of architectures beyond transformers. A compositional function is defined, in principle, by its constituent functions and their pattern of composition. Since the hippocampus already discovers and stores local transition rules between sparse indices, a natural question arises: *what computational role remains for the cortex?* If the hippocampus already possesses the constituents, why is cortical learning necessary at all?

The resolution is that the hippocampus and cortex represent constituent functions at fundamentally different levels of abstraction.

23.12.1 Constituent Functions in the Hippocampus

The hippocampus learns *instance-bound* constituent functions. For each stored index $h \in \mathcal{H}$, it encodes specific local transitions

$$G_h : h \mapsto h',$$

where h' is an observed successor of h. These constituents have the following properties:

- They are tied to specific indices h.
- They are episodic rather than parametric.
- They are stored non-generatively, without pressure to generalize.
- They explicitly avoid weight sharing due to pattern separation.

Formally, the hippocampus implements a collection

$$\{G_{h_i}\}_{i=1}^{M},$$

which should be understood as a structured lookup table of local transition rules rather than as a compact representation of a compositional function. While these constituents define paths through the scaffold graph, they do not yet constitute a parametric or generalizable model.

23.12.2 What the Cortex Learns Beyond the Constituents

The cortex operates on the set of hippocampal constituents and learns structure *across* them. This involves three capabilities that are unavailable to the hippocampus.

Parameter Sharing and Abstraction. The cortex discovers that many hippocampal constituents are instantiations of the same underlying transformation. Transitions such as "Kitchen $\rightarrow$ Hallway" and "Bedroom $\rightarrow$ Hallway" are stored separately in the hippocampus, but the cortex learns a shared operator

$$G_h(h) \approx \tilde{G}(\phi(h)),$$

where $\phi(h)$ maps sparse indices to a lower-dimensional latent representation. This parameter sharing collapses many episodic constituents into a single abstract operator.

Learning the Algebra of Composition. The hippocampus provides paths through the scaffold graph,

$$h_{t+T} = G_{h_{t+T-1}} \circ \cdots \circ G_{h_t}(h_t),$$

but it does not represent equivalence relations between paths. The cortex learns higher-order regularities among constituent functions, such as approximate associativity, commutation, or cancellation:

$$G_a \circ G_b \approx G_c, \qquad G_{\text{left}} \circ G_{\text{right}} \approx \text{Id}.$$

This constitutes an algebra over constituent functions rather than a mere enumeration of edges.

Amortization and Extrapolation. Because hippocampal constituents are stored as isolated instances, they cannot be applied to novel states. The cortex learns a parametric family of transitions

$$G_\theta : \mathcal{H} \rightarrow \mathcal{H},$$

allowing prediction from previously unvisited but structurally consistent states. This enables zero-shot generalization and counterfactual inference, which are impossible in a purely episodic system.

23.12.3 A Hierarchy of Representations

The resulting division of labor can be summarized as follows:

System	Representation	Function
Hippocampus	Sparse indices h	Instance-level constituents
Early Cortex	Shared operators	Parameter tying
Deep Cortex	Operator algebra	Compositional laws
Semantic Cortex	Latent manifold $\hat{z}$	Schema-level dynamics

23.12.4 Conceptual Resolution

The hippocampus does not represent compositional functions in the sense of approximation theory or learning theory. Instead, it *enumerates* their local constituents episodically. The cortex uses this enumerated set as training data to infer a compact, parametric, and algebraically structured representation. In this sense, the hippocampus discovers the pieces of the compositional function, while the cortex learns how those pieces fit together and how they generalize beyond experience.

23.12.5 Artificial Analogues of Cortical Abstraction

The computational role attributed here to cortex is not hypothetical. Several classes of artificial architectures already implement parts of this functionality, provided that a hippocampus-like indexing mechanism is present, either explicitly or implicitly. These systems clarify what cortex can do beyond enumerating constituent functions.

Common Pattern. Across successful artificial systems, a recurring division of labor appears:

1. A mechanism that produces discrete, sparse, or indexed representations of experience.

2. A parametric learner that extracts shared structure, invariances, and composition rules across those indices.

Only the second step supports abstraction, extrapolation, and algebraic generalization.

Meta-Learning Architectures. In meta-learning systems, individual tasks correspond to instance-level constituents, while the meta-learner discovers shared parameters across tasks. This mirrors the hippocampus–cortex interaction: the hippocampus enumerates local transition rules, while the cortex infers a parametric family that generates them. Without a diverse set of indexed instances, meta-learning fails to generalize.

Graph Neural Networks. Graph neural networks provide a clear artificial analogue of cortical computation. They assume a given graph structure and learn transition rules that are invariant across nodes and edges. In this sense, they operate on a scaffold provided externally, learning shared operators rather than storing individual transitions. Crucially, GNNs do not construct the graph itself, highlighting the necessity of a separate scaffold-building system.

Transformers and Operator Algebra. Transformers trained on structured sequences (e.g., language, arithmetic, programs) learn algebraic relations among operators rather than memorizing individual transitions. However, their success depends on discrete tokenization and explicit indexing of inputs. These preprocessing steps effectively perform the hippocampal function, allowing the transformer to act as a cortex that learns composition laws over tokens.

Program Induction and Modular Networks. Neural module networks and program induction systems explicitly represent reusable subroutines and compose them hierarchically. These architectures presuppose stable, indexable modules and focus on learning how modules combine and generalize, not on discovering the modules themselves.

Interpretation. Taken together, these artificial systems support the central claim of this chapter: abstraction, parameter sharing, and compositional generalization require a representation in which constituent functions are already discretized and indexed. The hippocampus provides this representation biologically. The cortex operates on top of it, learning shared operators and the algebra of their composition. End-to-end learning without such a scaffold is generically ill-posed and empirically fragile.

23.13 Summary

The hippocampus is often described vaguely as a "memory store." This chapter proposes a more precise computational role: it is a **teacher of compositional structure**.

1. **Pattern Separation:** It receives correlated sensory data (x) and latent context (z) and projects them into orthogonal **Sparse Indices** (h), preventing catastrophic interference.

2. **Scaffold Construction:** It links these indices over time to form a **Scaffold Graph** that explicitly represents the topology of the latent space.

3. **Compositional Instruction:** This graph imposes **Compositional Sparsity** on the learning problem. By training on these sparse transitions, the cortex learns to model the world as a composition of local, manageable functions rather than a dense, intractable mess.

4. **Scaffold Removal (Consolidation):** Over time, the cortex internalizes this structure, learning the underlying manifold ($\hat{z}$) directly from sensory inputs. Once the "floor plan" is learned, the specific hippocampal indices are no longer required for retrieval, and the scaffold can be discarded.

In the Memory Palace analogy: The hippocampus builds the palace and assigns unique addresses to every room so that memories can be stored without overlap. It then guides the cortex through these rooms, teaching it the layout. Eventually, the cortex memorizes the architecture of the palace itself, allowing it to navigate the "Kitchen" and "Hallway" without needing to look up the room numbers on the door.

Viewed through the lens of compositional sparsity, the hippocampus is neither merely a memory store nor a spatial navigator. It is a compiler that transforms dense, correlated experience into a sparse, structured representation. By doing so, it converts an otherwise intractable learning problem into a hierarchy of local computations that the cortex can learn efficiently. Once this structure is internalized, the scaffold can be removed, leaving behind a semantic model of the world. See also Appendix G. It is important to emphasize that *artificial systems that exhibit abstraction, algebraic composition, and extrapolation invariably rely on an explicit or implicit indexing mechanism. In biological systems, this role is played by the hippocampus*. Furthermore, if the cortex merely instantiated compositional structure without compressing it, it would remain permanently dependent on hippocampal indexing. Systems consolidation therefore implies not just abstraction, but the discovery of rules that collapse multi-step compositions into single operators.

23.14 Connection with grid cells

We discuss the paper "Toroidal topology of population activity in grid cells" by Gardner et al. (Nature, 2022). This paper provides the experimental "smoking gun" for the theory described in the chapters you just read. It confirms that the "Latent State" z_t is not just a mathematical abstraction but a physical reality in the brain: a Torus.

Here is how that paper connects to the "Scaffold" and "Compositional Sparsity" framework:

1. The Physical Manifold is a Donut (Torus)

In the "Mixture of Experts" chapter, we discussed approximating functions on "low-dimensional manifolds." Gardner et al. (2022) proved that the population activity of a single grid cell module lies on a Toroidal Manifold.

The Finding: Even though the rat runs around a 2D floor (a flat sheet), the neural activity moves along the surface of a "doughnut" in signal space.

The "Rigid Scaffold": Crucially, Gardner showed this toroidal structure persists even during sleep (REM and Slow-Wave). This proves the "scaffold" is a pre-existing, rigid hardware structure (continuous attractor network), not something learned from the sensory world.

2. Modularity as "Combinatorial Capacity"

"Different modules at different scales" is the key to the system's efficiency (Compositional Sparsity).

The Math: If Module A is a torus of scale 30cm and Module B is a torus of scale 40cm, the brain represents a location by combining coordinates from both. The result: This acts like a residue number system. A small number of modules can encode a massive range of unique locations (exponential capacity), exactly as the Deep MoE chapter described (E^L capacity).

3. The "Gating" Mechanism

In the context of the "Hippocampal Scaffold" chapter, the Gardner paper implies that the Grid Cell modules act as the Address Generator.

Input: The rat enters a room.

Grid Modules: Module 1 is at phase ϕ_1, Module 2 is at phase ϕ_2.

Combination: These phases define a specific point on the joint-toroidal manifold.

Hippocampus: This unique combination forces the dentate gyrus to fire a specific sparse index h_t.

If the grid cells were not modular (i.e., just one scale), the system would run out of unique addresses very quickly ("catastrophic interference"). The modularity allows the scaffold to be infinite for all practical purposes.

CHAPTER 24

Reusable Sparse Compositionality

In the previous chapters, we established that efficiently computable functions must be sparsely compositional. However, this condition describes the existence *of a decomposition, not its* uniqueness.

A fundamental problem remains: there are infinitely many ways to decompose a function into a graph of local operations. A neural network might learn a "Rube Goldberg" solution—a messy, entangled decomposition that fits the data perfectly but contains no recognizable or reusable parts.

This chapter argues that ***Sparse Compositionality alone is not enough****. To discover the "true" constituent functions of nature (the LEGO bricks), we must impose an additional constraint:* ***Reusability****.*

24.1 The Non-Uniqueness of Composition

Sparse compositionality tells us that the target function f can be represented by a sparse Directed Acyclic Graph (DAG). It does not, however, tell us *which* DAG the network will find.

Consider the simple multiplication of two variables, $f(x, y) = x \cdot y$. The "natural" decomposition is a single node: a multiplier. However, one could also decompose this function using the polarization identity:

$$f(x, y) = \frac{1}{4}\left((x + y)^2 - (x - y)^2\right).$$

Both representations are sparsely compositional. Both have bounded fan-in. Both compute exactly the same result. Yet, the second representation relies on specific cancellations between terms. If we wanted to reuse the "multiplication" logic in a different context, the second architecture offers us no distinct multiplier module—only adders and squarers arranged in a specific topology.

24.1.1 The Identifiability Problem

In deep learning, this is known as the problem of **Identifiability**. Let $\mathcal{F}$ be the class of sparsely compositional functions. A network trained to minimize a loss $\mathcal{L}$ finds a parameter setting θ such that $f_\theta \approx f^*$. However, the optimization landscape contains a vast manifold of solutions $\Theta_{opt} = \{\theta \mid \mathcal{L}(f_\theta) \approx 0\}$.

Many solutions in Θ_{opt} correspond to "entangled" representations where high-level concepts are smeared across the network rather than isolated in distinct modules. Without further constraints, gradient descent has no reason to prefer the modular, physical decomposition over an entangled one [96, 167].

24.2 The Constraint of Reusability

How does nature solve this? Why does evolution produce modular organs (eyes, hearts, kidneys) rather than a monolithic biological sludge?

The answer lies in the **multiplicity of tasks**. Evolution does not optimize a single scalar loss function for a static environment. It optimizes for survival across a distribution of changing environments and tasks [93].

We propose the **Reusability Principle**:

> **A constituent function is physically "real" only if it reduces the complexity of multiple, distinct tasks simultaneously.**

In the context of learning theory, this suggests that to discover the standard reusable constituents, we cannot train on a single function f. We must train on a **multiverse of tasks** $\{f_1, f_2, \ldots, f_k\}$ that share an underlying physical reality [17].

24.2.1 Formulation: Shared Modularity

Let our learning objective be minimizing the total description length (or loss) of k distinct tasks:

$$\min_{\mathcal{G}} \sum_{i=1}^{k} \mathcal{L}(f_i, \text{data}_i) \quad \text{s.t.} \quad f_i \in \text{span}(\mathcal{G}),$$

where $\mathcal{G}$ is a library of primitive modules (sub-graphs). If we strictly constrain the size of $\mathcal{G}$, the network is forced to find the *intersection* of the computational graphs of all tasks.

- If Task A requires detecting edges to classify squares,
- and Task B requires detecting edges to avoid obstacles,
- an entangled solution for A that mixes "edges" with "squareness" will not transfer to B.
- The only efficient compression is to isolate "edge detection" as a distinct, reusable module in $\mathcal{G}$.

Thus, **Multi-Task Learning (MTL)** is not just a trick for better performance; it is the theoretical scalpel required to carve nature at its joints [27].

24.3 Genericity as a Selector for Modularity

Reusability is the primary structural constraint. The second constraint is geometric: **Genericity** (discussed in Chapter 2).

Recall that generic functions are stable under perturbation [183]. This applies to the decomposition itself.

- **Ad-hoc decompositions are fragile.** The polarization identity $xy = \frac{1}{4}((x+y)^2 - (x-y)^2)$ relies on precise cancellation. If we add noise to the intermediate nodes (the squaring operations), the equality breaks down catastrophically.
- **Natural decompositions are robust.** The direct multiplication $x \cdot y$ is stable. Small noise in the input produces small noise in the output.

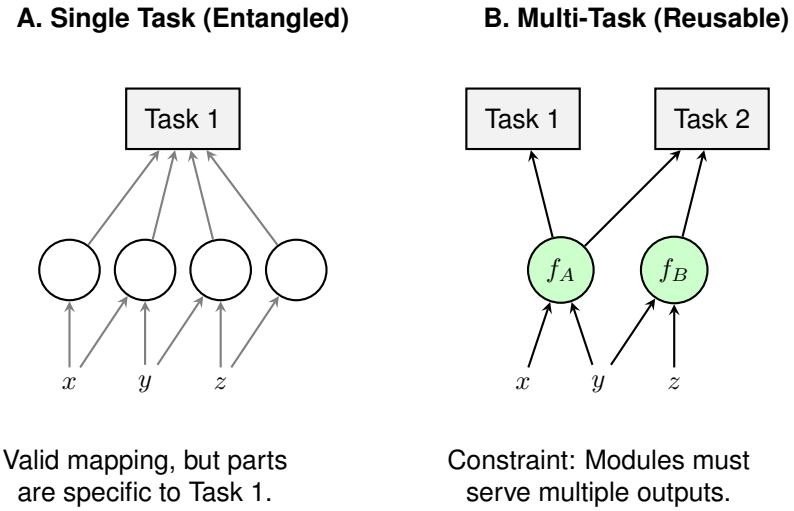


Figure 24.1: **The Constraint of Reusability.** (A) When solving a single task, the network can find an "entangled" solution (a Rube Goldberg machine) that works but contains no reusable parts. (B) When forced to solve multiple tasks simultaneously with limited capacity, the network must discover the shared, reusable constituent functions (green nodes) that reflect the true causal structure of the data.

Therefore, by injecting noise into the hidden layers during training (or enforcing stability constraints on the Jacobian), we penalize "Rube Goldberg" decompositions that rely on fine-tuned cancellations. The network settles into the "flattest" minima, which correspond to the simplest, most modular algorithmic implementations [79].

24.4 Summary: The Trinity of Learnability

We can now refine our theory of intelligence. It rests on three legs, not just two:

1. **Sparse Compositionality:** The world *can* be decomposed into a hierarchy of simple parts (Existence).
2. **Reusability (Multi-Tasking):** The true parts are those that are useful across many different contexts (Uniqueness).
3. **Genericity:** The learning process can find these parts because they leave stable, robust footprints in the optimization landscape (Searchability).

Deep learning succeeds not because it is a blank slate, but because the architecture (layers, convolution) and the training regime (SGD, noise, large datasets) implicitly enforce these three constraints, guiding the system toward the true causal structure of the universe [18].

CHAPTER 25

What Is Missing in LLMs?

If the central thesis of this collection is that modern AI succeeds by exploiting the sparse compositional structure of the universe, then the limitations of current Large Language Models (LLMs) emerge precisely where they do not yet implement these structural principles fully. Transformers behave as powerful associative memories, but they remain incomplete realizations of the architectures evolution discovered for intelligence. To approach human-level cognition, we must go beyond the "diligent learner" paradigm [173] and reconstruct the pre-linguistic foundations of thought.

25.1 The Missing Foundation: World Models Before Language

Current LLMs behave *as if* language were a complete surrogate for the world. They infer physical laws, social dynamics, and even psychology indirectly through text, rather than through direct interaction with a structured environment. This inversion is historically and biologically implausible. Evolutionary history points in the opposite direction: language is a late interface layered upon much older mechanisms for world modeling [69].

Long before humans could speak, they had internal predictive models capable of anticipating consequences, inferring hidden causes, and planning actions. These models can be abstractly described as latent dynamical systems:

$$x_{t+1} \approx F(x_t),$$

where x_t is a compact internal representation that encodes the relevant state of the environment. Such predictive dynamics support control, planning, and counterfactual reasoning, and correspond to what may be called a "predictive physics engine" [107].

LLMs lack this grounding. They learn F only indirectly, through linguistic traces of past human interactions with the world. As a result, they conflate symbolic regularities with physical structure. Achieving human-level intelligence therefore requires a transition from token-based sequence models to what we call **Large Embedding Models (LEMs)**: systems whose primary computational object is a latent *state*, not a word.

Rather than predicting the next token, a LEM updates or reconstructs a high-dimensional embedding that represents the current physical, social, or introspective state of the agent. Formally, a LEM performs inference in a latent dynamical system:

$$z_{t+1} \sim G_\theta(z_t, \epsilon_t), \tag{25.1}$$

where z_t is a latent state and ϵ_t represents stochasticity or uncertainty. Crucially, inference proceeds via reconstruction and denoising, not symbol prediction. This aligns closely with the associative mechanisms discussed in earlier chapters: a LEM is a structured attractor system approximating an Associative Turing Machine (ATM), performing completion in latent space rather than in token space. These dynamics are strikingly similar to those observed in biological brains during memory retrieval and dreaming.

25.2 A Structural Limitation of LLMs

Because Transformers operate on finite token contexts without persistent latent state, any task requiring the maintenance and manipulation of hidden world state across episodes must be re-derived from scratch at each interaction. The model has no notion of continuity beyond what is explicitly present in the prompt.

Consider a task whose solution depends on an unobserved latent variable z_t that is not explicitly encoded in the token stream. Without an external memory or state variable, an LLM must implicitly reconstruct z_t from the context. The computational cost of this reconstruction increases with context length and, in many realistic settings, exponentially with the number of latent variables.

This leads to intrinsic inefficiency. Tasks that are trivial for organisms—maintaining identity, tracking intentions, reasoning across days or weeks—require repeated inference from surface cues in LLMs. As a consequence, LLMs are computationally inefficient for:

- multi-episode planning with hidden state,
- counterfactual reasoning requiring intervention on latent variables,
- lifelong learning with persistent identity and memory.

This phenomenon mirrors the "cliff" results discussed earlier in this volume: in the absence of stable low-dimensional structure, optimization and inference devolve into combinatorial reconstruction rather than progressive refinement.

25.3 From Diligence to Exploration

Current LLM training paradigms produce what Shalev-Shwartz et al. called "diligent learners" [173]: systems that excel at minimizing prediction error on a fixed corpus. This yields impressive crystallized competence but lacks fluid intelligence—the ability to generate hypotheses, explore uncharted spaces, and reorganize internal representations.

Human cognition incorporates intrinsic exploration at multiple time scales. Evolution itself acts as a meta-level search over architectures, objectives, and developmental procedures. At the level of learning dynamics, this corresponds to optimizing not only the parameters θ, but also the structural degrees of freedom that govern learning.

$$\min_{\phi} \quad \mathbb{E}_{\text{env}}[\, L(\theta_\phi) + \lambda\, \mathcal{E}(\theta_\phi)] \,,$$

where ϕ controls initialization, architecture, or regularization, and $\mathcal{E}$ measures novelty or exploratory behavior [137].

To approach human-like fluid intelligence, artificial systems must therefore incorporate:

- **Exploration bonuses** that reward walking off the training manifold [140],

- **Evolutionary or population-based search** over architectures and learning rules [163],
- explicit mechanisms for hypothesis generation rather than passive pattern fitting.

Intelligence, in this view, is not merely about minimizing error but about continuously reshaping the space of hypotheses itself.

25.4 The Memory Gap: From Context Windows to Turing-Efficient Memory

Transformers possess only a finite working memory: a bounded context window. Once a session ends, the entire internal state disappears. In this sense, they behave like "soft" Turing machines without persistent tape.

Biological intelligence depends on a fundamentally different architecture. The hippocampus implements a **Turing-efficient memory system**: it stores episodic records in a single step and retrieves them indefinitely using content-based addressing. Abstractly, memory is an external store (K, V) where keys are embeddings of experiences and retrieval uses similarity search:

$$\text{retrieve}(q) = V\left(\arg\min_{k \in K} d(q, k)\right).$$

From a computational standpoint, simulating memory through context re-encoding incurs quadratic or worse overhead in sequence length. An Associative Turing Machine decouples memory access from sequence length, enabling constant-time writes and sublinear retrieval. This separation explains why persistent identity, long-horizon planning, and cumulative learning are fragile in current LLMs

25.5 The "Memento" Condition: Externalizing State

The functional architecture of a Transformer corresponds precisely to a cognitive agent suffering from total anterograde amnesia. Like the protagonist of the film *Memento*, the model possesses a frozen set of long-term memories (pre-trained weights θ) but lacks the physiological capacity to form new stable internal traces during inference.

In biological systems, working memory transitions into long-term changes via synaptic plasticity. In a frozen LLM, this path is blocked. Consequently, the agent must resort to a prosthetic strategy: **externalizing state**. Just as Leonard Shelby must tattoo facts onto his body or write notes on Polaroids to maintain continuity, an LLM must emit tokens into its context window to "remember" its own intermediate thoughts.

This necessity explains the unreasonable effectiveness of **Chain-of-Thought (CoT)** prompting. CoT is not merely a reasoning heuristic; it is the algorithmic equivalent of using an external scratchpad to compensate for the lack of a hidden state variable h_t that persists across time steps.

Formally, if an agent cannot update its internal state $s_t \to s_{t+1}$, it must offload the state update to the environment (the context window C_t):

$$C_{t+1} = C_t \cup \{\text{"note"}\}.$$

To access this state at time $t + k$, the model must re-process ("read") the entire history C_{t+k}. This incurs a quadratic computational penalty $O((t + k)^2)$ for what should be an $O(1)$ memory lookup.

Thus, the "diligent learner" is forced to be a "diligent note-taker." This structural reliance on external tokens for short-term continuity is the defining constraint of current architectures—a constraint that biological brains, with their ability to maintain fluid internal dynamics, do not share.

25.6 The Barrier: Reusability, Genericity, and Sparsity

The final gap is efficiency. Biological intelligence evolved under strict metabolic constraints, favoring architectures that reuse a finite library of compositional primitives across many tasks—what we call **sparse compositionality**. Rather than learning monolithic entangled functions, evolution reuses the same modules for vision, planning, locomotion, and reasoning [18].

Current LLMs, by contrast, remain largely entangled: knowledge is distributed over billions of parameters without clear modular boundaries. This violates the **Reusability Principle**: a function is physically "real" only if it solves multiple distinct tasks with the same underlying components.

25.6.1 Why LLMs Violate Genericity

Genericity requires that meaningful physical functions leave stable, low-degree statistical footprints under perturbation. Language, however, is a highly compressed and culturally mediated projection of the world. As a result, many correlations learned by LLMs are:

- non-generic, disappearing under small changes in the underlying world,
- accidental, tied to dataset-specific conventions,
- non-reusable across domains.

Without a latent world model, gradient descent reinforces symbolic coincidences rather than stable physical structure. This explains both the impressive interpolation abilities of LLMs and their brittleness under distribution shift.

25.7 Memory as Generative Reconstruction

A crucial distinction between biological and artificial memory lies in the difference between retrieval and reconstruction. In classical computing and current RAG-based LLMs, memory is a look-up table: information is stored faithfully and retrieved exactly. In biological systems, memory is generative.

To remember an event is not to replay a recording, but to use a compressed latent seed to re-run the world model, reconstructing a plausible past. Formally, if z_t denotes a latent state, recall corresponds to posterior inference:

$$\hat{z} = \arg\max_{z} P_{\text{world}}(z \mid q),$$

where q is a partial cue. The same generative mechanism used to predict the future is used to reconstruct the past or imagine counterfactuals.

This explains the remarkable efficiency of biological intelligence: the brain stores generative parameters rather than high-bandwidth sensory data. This "generative compression" is far more efficient than Shannon compression, but it sacrifices veridicality—a trade-off that LLMs, which often memorize training data verbatim, have not yet made.

25.8 The Causal Ladder

LLMs excel at detecting correlations, but intelligence requires climbing the causal ladder [**pearl2009causality**]:

1. **Association:** $P(y|x)$,

2. **Intervention:** $P(y|do(x))$,
3. **Counterfactuals:** $P(y_x|x', y')$.

A LEM architecture naturally supports higher rungs. Because it maintains a latent physics engine $F(x_t, a_t)$, it can simulate interventions without acting. LLMs, lacking a separation between agent and world, can only imitate the linguistic description of causal reasoning.

25.9 The Algorithmic Role of Sleep

Sleep provides the missing algorithmic ingredient. During sleep, biological systems perform generative replay, producing synthetic variations of past experience, and systems consolidation, transferring fast episodic memories into slow cortical structure. Current AI systems are effectively insomniac. Introducing sleep-like phases—offline training on self-generated fantasies—may be essential for sample-efficient learning.

25.10 The Symbol Grounding Problem: Maps Without Territories

While the "Memento" condition highlights a temporal deficit, a perhaps deeper flaw lies in the topological separation between the model's internal representations and the physical world. LLMs exhibit what can be called **semantic ungrounding**.

In biological intelligence, the meaning of a high-level symbol (e.g., "cup") is recursively composed of low-level sensorimotor invariants: the affordance of grasping, the visual prediction of curvature, and the haptic expectation of rigidity. This grounding provides a rigid boundary condition for inference. A biological agent cannot hallucinate that a cup is "soft" without generating a prediction error from its motor cortex.

In contrast, an LLM learns the manifold of language, which is a high-dimensional projection of the world, not the world itself. As Korzybski noted, "the map is not the territory." Current models are experts in the topology of the map (language) but blind to the geometry of the territory (physics).

Formally, if $\mathcal{W}$ is the state space of the world and $\mathcal{L}$ is the space of language, there exists a projection $\Pi : \mathcal{W} \to \mathcal{L}$ which is lossy and non-invertible.

- **Humans** learn $F : \mathcal{W} \to \mathcal{W}$ (intuitive physics) and use language to communicate state.
- **LLMs** learn $G : \mathcal{L} \to \mathcal{L}$ (symbol manipulation) and attempt to infer physics $\mathcal{W}$ solely from the projection.

This explains why LLMs struggle with basic physical reasoning or spatial consistency: they are attempting to reconstruct the high-dimensional phase space of reality from a lower-dimensional symbolic shadow. True computational efficiency—and robustness against hallucination—requires the model to operate directly on the latent physics of $\mathcal{W}$, enforcing conservation laws and causal constraints that language often elides.

25.11 Conclusion

Scaling LLMs is not enough. Human-level intelligence requires architectures built around latent world models, exploration, Turing-efficient memory, and sparse compositional reuse supported by genericity. Language is an interface to this architecture—not its foundation.

25.12 Proposed Experiments

25.12.1 Fragmented Embeddings and Reconstruction

Quantize embeddings and test whether reconstruction degrades gracefully. **Prediction:** LEMs degrade smoothly; LLMs fail catastrophically.

25.12.2 Diffusion in Memory Space

Train diffusion models over embeddings. **Prediction:** Novel but coherent recombinations emerge only in LEM-like systems.

25.12.3 Neurobiological Correlates

Test whether hippocampal replay follows denoising trajectories in neural state space.

25.12.4 Conceptual Prediction

Internal representations should exhibit local smoothness and global compositionality—detectable through manifold analysis.

25.13 What Is Missing in Large Language Models: Compression of Composition

The preceding chapters argue that the cortex does more than represent compositional structure: it discovers abstractions that *compress* composition itself. This observation has direct consequences for contemporary large language models (LLMs), including Transformers. While such models exhibit remarkable compositional capabilities, they lack a critical operation required for cortical-like learning.

25.13.1 Representation Without Compression

Transformers successfully instantiate compositional structure. Through attention mechanisms, they dynamically bind tokens, reuse operators across contexts, and apply learned functions compositionally. However, these compositions are re-derived at inference time from the full context. The model remains permanently dependent on its scaffold: the sequence of tokens and the attention paths that relate them.

In other words, Transformers *use* compositional rules but do not *compress* them. A multi-step composition

$$G \circ G \circ G$$

is never replaced by a shorter description. Instead, the same compositional structure must be reconstructed repeatedly, incurring constant computational and representational cost.

25.13.2 Cortical Learning as Rule Compression

In contrast, the biological cortex—if it is to become independent of hippocampal indexing—must do something strictly stronger. It must infer rules that reduce the description length of composed

functions. That is, given repeated experience with a composition, the cortex should learn a new operator

$$G^{(3)} \approx G \circ G \circ G$$

that can be applied directly, without replaying the underlying constituents.

This operation is not caching, memoization, or faster lookup. It is the induction of a *new rule* that collapses depth. As learning progresses, effective inference should become shallower, not deeper. Systems consolidation therefore implies not merely abstraction, but the discovery of equivalences that shorten computational paths.

25.13.3 Depth Collapse and Learned Shortcuts

This perspective has an immediate architectural implication: a cortex-like system should reduce its effective depth over learning. Early in learning, deep compositions are unavoidable. Later, once structure has been discovered, behavior should be mediated by fewer stages.

In this framework, skip connections are not merely optimization devices for gradient flow. They are the representational trace of discovered equivalences:

$$x \xrightarrow{\text{long composition}} y \quad \approx \quad x \xrightarrow{\text{short rule}} y.$$

A learned shortcut corresponds to a compressed rule that replaces a multi-step computation. The ability to *create* such shortcuts dynamically is therefore a semantic capability, not just an architectural convenience.

25.13.4 The Limitation of Fixed-Depth Architectures

Standard Transformers have fixed depth and fixed computational graphs. Although attention weights are dynamic, the number of stages required to compute a result does not decrease with learning or expertise. As a consequence, Transformers remain permanently scaffold-dependent: they cannot internalize structure in a way that eliminates the need for contextual recomputation.

This stands in contrast to biological systems, where practice reduces reaction time, fewer areas are recruited for expert tasks, and consolidated knowledge can be accessed without replaying episodic structure. These phenomena are signatures of compositional compression.

25.13.5 Architectural Implications

The analysis suggests that any architecture aspiring to cortical-level abstraction must support:

- **Rule Induction:** Promotion of frequently composed functions into first-class operators.
- **Depth Collapse:** Reduction of effective inference depth as structure is learned.
- **Learned Shortcuts:** Dynamic creation of skip connections representing discovered equivalences.
- **Scaffold Discarding:** The ability to perform inference without explicit reference to the original compositional scaffold.

Large language models achieve impressive reasoning by exploiting a powerful scaffold, but they do not compress it away. From the perspective developed in this book, this limitation is not incidental: it reflects the absence of a cortical mechanism for compressing composition itself.

Overcoming this limitation likely requires architectures that behave less like sequence models and more like systems that induce, promote, and internalize rules—collapsing long compositions into shorter descriptions over learning.

25.14 Beyond LLMs: From Read-Only Models to Associative Turing Machines

If Transformers are understood as an incomplete approximation to a more fundamental cognitive architecture, the natural question is: what lies beyond them? The answer suggested by the preceding analysis is not simply larger models or longer context windows, but a qualitative shift in how memory, computation, and learning interact.

The defining limitation of current large language models is a rigid separation between *memory* and *computation*. Memory is externalized into a context window, while computation is frozen into static weights. Biological intelligence does not respect this separation. Instead, it operates as an **Associative Turing Machine (ATM)**: a system with persistent internal state, content-addressable access, and the ability to modify its own transition rules in real time.

25.14.1 The Transformer as a "Read-Only" ATM

From this perspective, the Transformer can be viewed as a restricted, read-only ATM. It possesses a powerful associative access mechanism—self-attention—but suffers from two fundamental algorithmic constraints.

1. **Ephemeral State.** The Transformer cannot write to a persistent internal tape. Its only writable memory is the external context window, which must be reprocessed in full at every step. As a consequence, long-horizon reasoning requires repeated recomputation of the entire history, resulting in quadratic time and memory complexity.

2. **Frozen Dynamics.** The control logic of the system, encoded in the parameters θ, is static during inference. The model cannot update its procedures, promote frequently used compositions into new operators, or internalize newly discovered rules without a full retraining cycle.

These constraints explain why Transformers remain permanently scaffold-dependent. They can exploit compositional structure when it is present in the context, but they cannot compress that structure into shorter descriptions that persist beyond the immediate computation.

25.14.2 Restoring Persistent State: Linear Recurrence and World Models

A necessary step beyond Transformers is the restoration of persistent internal state. This motivates the emergence of **State Space Models (SSMs)**, which replace explicit context replay with a compact, evolving latent state:

$$h_t = Ah_{t-1} + Bx_t, \tag{25.2}$$

$$y_t = Ch_t. \tag{25.3}$$

Unlike attention-based models, SSMs compress interaction history into h_t, enabling linear-time processing and continuous state evolution. This recurrence aligns closely with the notion of a world model: h_t represents not a bag of tokens, but a dynamical estimate of the environment's latent state.

By enforcing structured recurrence during inference, such models approximate the read/write head of an ATM in continuous space. However, while they restore persistent state, they do not by themselves solve the problem of rule compression or online adaptation.

25.14.3 A Frontier: Online Plasticity and Rule Internalization

The most significant remaining gap between artificial and biological systems is the freezing of parameters after training. In biological cortex, learning does not stop at inference time. To think is to learn; synaptic weights change on timescales ranging from seconds to years.

A true Associative Turing Machine must therefore support **online plasticity**. In such systems, associative access does not merely retrieve stored values—it modifies the transition rules themselves. Abstractly, this can be written as:

$$W_{t+1} = W_t + \eta \cdot \mathrm{OuterProduct}(\mathrm{Key}_t, \mathrm{Value}_t), \tag{25.4}$$

where experience directly updates the function that governs future computation.

This mechanism enables something fundamentally absent from current LLMs: the promotion of repeated compositions into new operators. Instead of recomputing a rule from context, the system internalizes it, reducing future computational depth. Memory and computation are no longer separate resources; they are two views of the same adaptive process.

25.14.4 Closing the Loop Between Map and Territory

Taken together, these developments point toward a post-Transformer architecture defined by three properties:

- persistent internal state rather than explicit context replay,
- adaptive transition rules rather than frozen dynamics,
- and compression of compositional structure rather than permanent scaffold dependence.

Such systems would not merely predict sequences, but continuously align their internal dynamics with the structure of the world. In doing so, they would finally close the loop between the map and the territory—achieving not just associative recall, but genuine algorithmic learning in real time.

CHAPTER 26

Computational role of eccentricity dependent cortical magnification

A central challenge in visual neuroscience is reconciling the fixed geometry of the retina with the need for invariant recognition. Poggio et al. (2014) demonstrate that the eccentricity-dependent increase in receptive field (RF) sizes is a computational requirement for achieving simultaneous scale and translation invariance.

26.1 Introduction

A central challenge in visual neuroscience is reconciling the fixed geometry of the retina with the need for invariant recognition. Poggio et al. (2014) demonstrate that the eccentricity-dependent increase in receptive field (RF) sizes is a computational requirement for achieving simultaneous scale and translation invariance [156]. This chapter unifies this architectural "Magic Map" with the local "Jet" perspective of the GELU activation function.

[Image of cortical magnification receptive field map]

26.2 Core Thesis

The paper by Poggio, Mutch, and Isik (2014) addresses why the primate visual system exhibits eccentricity-dependent receptive field (RF) sizes and spatial sampling [38, 85, 61]. The authors argue that this architecture is a computational requirement for achieving simultaneous invariance to scale and translation [4, 6].

26.3 The Inverted Truncated Pyramid

M-theory predicts that to maintain recognizability across a fixed range of scales (from s_{min} to s_{max}), the set of receptive fields in V1 must form an **inverted truncated pyramid** in the scale-space (s, x) plane [170, 123, 55].

- **Joint Invariance:** This geometry represents the locus of bounded joint transformations in scale and space [55].

- **Foveola Anchor:** The "bottom" of the pyramid (s_{min}) corresponds to the foveola, where resolution is highest [6, 123].
- **Scale Priority:** This architecture arises naturally if scale invariance is prioritized over shift invariance, as body motion (scale change) is more "expensive" than eye fixations (shift) [38, 61, 123].

26.4 The Magic Map: remaping to a Square Lattice

A critical theoretical contribution is the **Magic Map**, which transforms the non-uniform sampling array of V1 into a uniform square lattice [156, 4, 170].

$$i_s = \log_f \left(\frac{s}{s_{min}} \right), \quad i_x = \frac{x}{s} \tag{26.1}$$

In this remapped space (i_s, i_x), scale and shift transformations commute [156, 55]. This property is inherited by higher cortical areas and implies that the same computational "jets" or filters can be applied uniformly across the lattice [156, 55].

[Image of log-polar mapping transformation]

26.5 Hierarchical Decimation and Crowding

The theory follows a hierarchy of dot products, sampling, and pooling (the HMAX/M-theory motif) [156].

- **Decimation:** At each stage (V1 $\rightarrow$ V2 $\rightarrow$ V4 $\rightarrow$ IT), the array is downsampled (decimated) [156, 85, 61].
- **Bouma's Law:** Crowding is explained as a byproduct of this area-by-area pooling [22, 6]. The theory predicts that the critical separation for recognition is $\Delta x \approx bx$, where $b \approx 0.4$ suggests the recognition signature for crowding originates in V2 [22, 55, 61, 85].

[Image of visual crowding Bouma's law demonstration]

26.6 Visual Recognition via IP Fragments

Recognition under natural conditions is proposed to take place through the composition of "IP fragments" (Inverted Pyramid scale-space fragments) collected over multiple fixations [4, 170, 123, 156]. Each fragment is a space-scale image patch that provides limited position invariance but robust scale invariance [22, 156, 170].

26.7 Predictions and Empirical Alignment

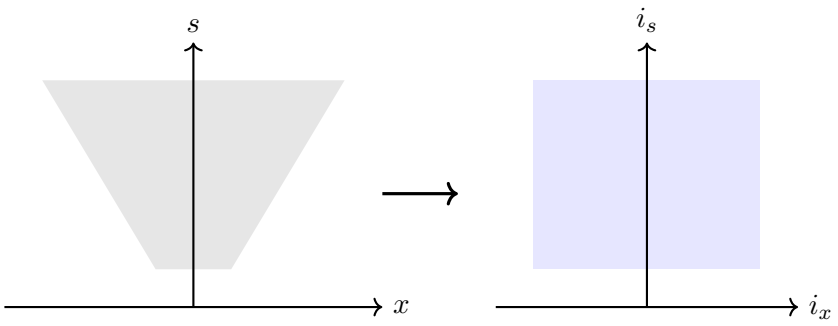
The paper provides several quantitative predictions [156, 170]:

- **Foveola Size:** Predicted to be $\approx 26'$ arc, containing about 40 basic units at the highest resolution [6, 156, 123].
- **Shift Invariance:** Predicted to depend linearly on spatial frequency, whereas scale invariance remains uniform across eccentricities [4, 6, 38].

- **Anstis Charts:** The theory computationally justifies Anstis' findings that letter recognition remains constant under scaling if the letters match the eccentricity-dependent RF sizes [6, 156, 170].

26.8 The Geometry of the Magic Map

The paper predicts that V1 is organized as an inverted truncated pyramid in the (s, x) plane. Notice that by transforming this non-uniform sampling into a square lattice, the visual system creates a space where shifts (i_x) and scaling (i_s) *commute*.



(A) Inverted Pyramid (B) Square Lattice

Figure 26.1: Remapping of visual space to a computational lattice where $i_s = \log_f(s/s_{min})$.

26.9 GELU Jets as Pooling Operators

The paper attributes Bouma's Law of Crowding to area-by-area pooling, likely originating in V2. We can interpret this pooling through the Taylor expansion (the "jet") of the GELU function:

$$GELU(x) \approx \underbrace{0.5x}_{\text{Linear Flow}} + \underbrace{0.398x^2}_{\text{V1 Selectivity}} - \underbrace{0.066x^4}_{\text{V2/V4 Pooling}} \tag{26.2}$$

While the linear component ($0.5x$) maintains the high-resolution "foveola" signal, the higher-order terms (x^2, x^4) represent the non-linear integration required for invariance. In this model, the negative x^4 term acts as a "clutter suppression" mechanism, mathematically representing the interference-free signatures required in the presence of distractors.

26.10 Phase-Dependent Perception

The synthesis of M-theory and the Jet framework suggests two distinct modes of operation:

1. **The Rapid Sweep (<100ms):** The network uses the Magic Map to untangle manifolds. The $0.5x$ component allows the IT cortex to receive a stable "gist" of the foveated fragment.

2. **Recurrent Indexing (>100ms):** The system plan fixations to compose "IP fragments." Each fragment provides a space-scale coordinate that serves as an address for hierarchical structural indexing.

26.11 Implications for Continual Learning

Catastrophic forgetting is avoided because the "Magic Map" provides a consistent coordinate system across the lifespan. New objects are learned as new IP-fragments (index entries) within the existing scale-space hierarchy, ensuring that the acquisition of new "Value" pointers in IT does not disrupt the "Key" extraction in V1.

CHAPTER 27

Nonlinear Scale Space

Classical computer vision and image processing are built on multiscale scale-space theory [201, 113]. In parallel, biologically inspired models such as the Neocognitron (Fukushima) and HMAX (Poggio) propose a semantic hierarchy [57, 166]. Here we propose a unification: each layer in the hierarchy contains a superposition of linear multiscale propagation and local nonlinear responses, representing distinct but complementary computations.

27.1 Introduction

Neurophysiology and psychophysics describe a hierarchy in which information is processed through receptive fields of increasing size. This aligns with scale-space models in computer vision, where an image is filtered by operators spanning multiple spatial scales (often interpretable as approximately bandpass channels, as in wavelet-like decompositions). Meanwhile, modern neural network models realize receptive-field growth through pooling and aggregation in a directed acyclic computation graph: deeper units integrate information from progressively larger regions of the input.

At first glance, these mechanisms offer competing explanations for how receptive fields grow from V1 to V2/V4 and IT. We propose a unification: the **semantic hierarchy** (V1 to IT) can be interpreted as a **multiscale scale space** for the *linear* components of computation, while the local "control logic" of selectivity and feature binding is carried by the *jets* (local Taylor expansions) of smooth *nonlinearities* (e.g. GELU). Typically, the output can be viewed as a superposition of these two contributions, which can be decoded separately when needed.

Definition (informal). A *nonlinear scale space* is a hierarchical representation in which *multiscale linear operators* act across spatial (and potentially temporal) scales, while *nonlinearities* contribute locally through the higher order terms of their *jets* (local Taylor expansions). Lower-order jet terms transmit signal and preserve coarse geometry, while higher-order terms act as structured, localized corrections that modulate and re-bind features.

27.2 A Minimal Formal Model

We model a nonlinear scale space as the composition of a multiscale linear operator with a pointwise smooth nonlinearity. Let $x^{(0)}$ be the input image (or a feature map), and define

$$x^{(\ell+1)} \;=\; \sigma\Big(L_\ell\, x^{(\ell)}\Big), \qquad \ell = 0, 1, 2, \ldots \tag{27.1}$$

where L_ℓ is a linear multiscale operator (e.g. convolution with a scale-dependent kernel, possibly followed by pooling/subsampling), and σ is a smooth activation (e.g. GELU).

Expanding σ at the operating point $u = 0$ yields the local jet

$$\sigma(u) \;=\; \sum_{k=0}^{K} \frac{\sigma^{(k)}(0)}{k!}\, u^k \;+\; R_{K+1}(u), \tag{27.2}$$

so (27.1) decomposes each layer into a dominant linear component plus structured higher-order corrections:

$$x^{(\ell+1)} \;=\; \alpha_0\, \mathbf{1} \;+\; \alpha_1\, L_\ell x^{(\ell)} \;+\; \sum_{k=2}^{K} \alpha_k \left(L_\ell x^{(\ell)}\right)^{\odot k} \;+\; R_{K+1}(L_\ell x^{(\ell)}), \tag{27.3}$$

where $\alpha_k = \sigma^{(k)}(0)/k!$, $\odot$ denotes elementwise powers, and $\mathbf{1}$ denotes the all-ones vector (included only if $\sigma(0) \neq 0$).

Interpretation. The operators L_ℓ implement a multiscale linear backbone (a scale-space flow across receptive-field sizes), while the jet terms describe how nonlinear selectivity enters as local, higher-order corrections to that flow.

27.3 Relation to Classical Scale Space

Classical (linear) scale space is characterized by a one-parameter family of smoothing operators $\{G_\ell\}_{\ell \geq 0}$, typically Gaussian, satisfying the semigroup property $G_{\ell_1} * G_{\ell_2} = G_{\ell_1+\ell_2}$ [201, 113]. In the continuous limit, this leads to the heat equation $\partial_\ell u = \Delta u$ with $u(\cdot, 0) = x^{(0)}$.

Equation (27.1) can be viewed as a *nonlinear* generalization: L_ℓ plays the role of a scale-dependent linear propagation operator, while σ breaks linear superposition. The jet expansion (27.2) makes this explicit by decomposing each layer into a dominant linear term plus controlled higher-order corrections. When pre-activations remain in a regime where the linear term dominates, the representation behaves approximately like a linear scale space; as higher-order jet terms become significant, the hierarchy departs from pure diffusion and implements selective, feature-dependent modulation.

27.4 The Jet of the GELU Activation

The Gaussian Error Linear Unit (GELU) provides a smooth, analytic example of a nonlinearity for this decomposition [75]. It is defined as

$$\mathrm{GELU}(x) \;=\; x\,\Phi(x) \;=\; x \cdot \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right], \tag{27.4}$$

where Φ is the CDF of the standard normal distribution and erf is the error function. A common approximation is

$$\mathrm{GELU}(x) \approx 0.5x \left(1 + \tanh\left[\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right]\right). \tag{27.5}$$

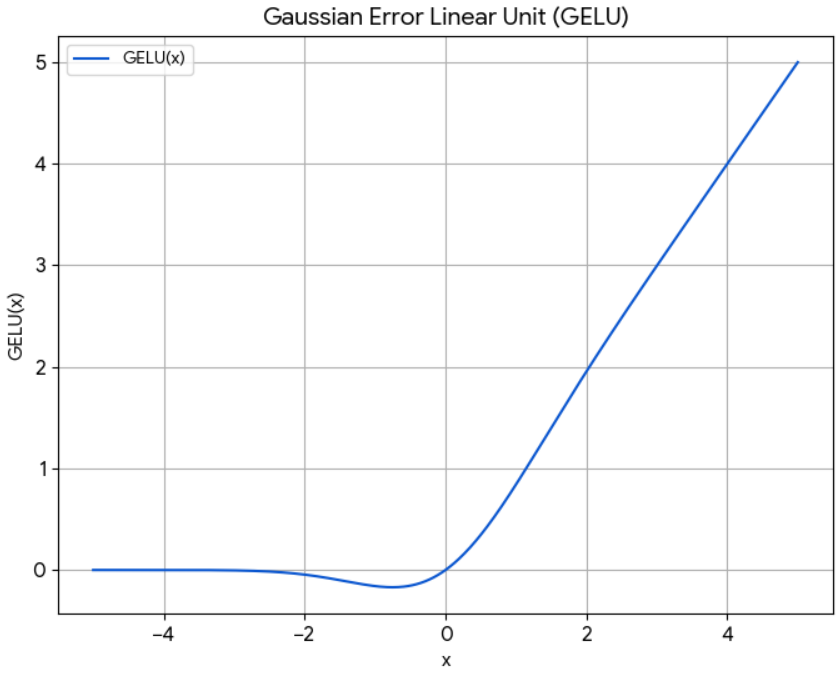


Figure 27.1: Plot of the Gaussian Error Linear Unit (GELU) function.

Its local behavior near $x = 0$ (the jet at the origin) can be written as

$$\mathrm{GELU}(x) \;\approx\; \frac{1}{2}x + \frac{1}{\sqrt{2\pi}}x^2 - \frac{1}{6\sqrt{2\pi}}x^4 + O(x^6). \tag{27.6}$$

This makes explicit a strong linear component ($\frac{1}{2}x$) together with higher-order curvature terms that can support selectivity and feature interactions.

Operating point and normalization. The jet expansion is local: it describes $\sigma(u)$ around an operating point, here taken as $u = 0$. In neural networks, the relevance of low-order terms depends on the distribution of pre-activations $u^{(\ell)} = L_\ell x^{(\ell)}$, which is shaped by normalization, gain control, and pooling/aggregation. These mechanisms can keep $u^{(\ell)}$ in a regime where the first-order term is a useful approximation on typical data, while higher-order terms provide structured deviations that increase selectivity.

Jets and scale (mechanistic reading). In the hierarchy (27.1), "scale" affects jet contributions through the distribution of pre-activations $u^{(\ell)} := L_\ell x^{(\ell)}$. Coarser scales (larger receptive fields and stronger aggregation) often reduce the variability of $u^{(\ell)}$ (e.g. via averaging/pooling and normalization), thereby shrinking higher-order contributions $|u^{(\ell)}|^k$ relative to the linear term. In this sense, increasing scale can *behave as if* it induces progressively stronger dominance of low-order jet components, while higher-order terms remain available as localized corrections.

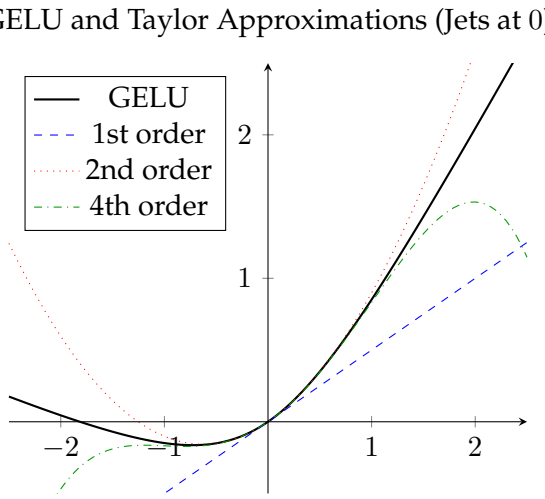


Figure 27.2: GELU and low-order jets around the origin.

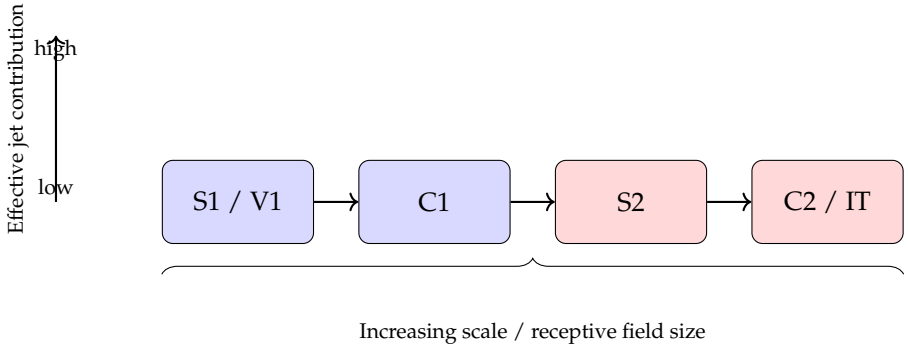


Figure 27.3: Unification of HMAX-like hierarchies and scale space through nonlinear jets. Receptive field size increases along the hierarchy; the multiscale linear backbone organizes representations by scale, while jet terms provide localized nonlinear selectivity.

27.5 Phase I: The Discriminative Pipeline (< 100 ms)

In the initial feedforward sweep, the ventral stream behaves approximately as a **hierarchical classifier** [184]. In our framework, a unit in IT can be viewed as combining (i) a *linear* component with a large effective receptive field (arising from the composition of the linear operators L_ℓ across layers) and (ii) *nonlinear* components (arising from the composition of local jet corrections induced by the nonlinearities). Early stages emphasize fine-scale structure (small receptive fields), while later stages combine information over larger scales [47].

CHAPTER 28

Is Efficiently Computable Intelligence Different from Unconscious Intelligence?

I argue that sparse compositionality is a fundamental structural property of functions that can be computed or learned efficiently by digital systems. Any function that is efficiently computable in the Church–Turing sense admits a representation as a composition of low-arity constituent functions arranged in a bounded-fan-in directed acyclic graph, and can therefore be approximated by a deep network with sparse connectivity. This principle provides a unified explanation for why deep learning avoids the curse of dimensionality, why optimization remains tractable despite nonconvexity, and why generalization often exceeds classical capacity-based predictions. Because all artificial intelligence systems are implemented on digital computers, the theory applies fully to machine learning. In contrast, it is not known whether all components of biological intelligence are efficiently computable, even if they are Turing computable in principle. This distinction suggests that some higher cognitive functions, such as language and abstraction, may be more accessible to computational modeling than evolutionarily older processes related to affect, motivation, or internal bodily states. Sparse compositionality thus clarifies both the power and the limits of machine intelligence, and points toward a theory of what can be learned efficiently from data.

Deep neural networks achieve remarkable performance across diverse domains, yet the nature of the functions they can represent and learn remains only partially understood. Here we argue that a single structural property—*compositional sparsity*—characterizes essentially all functions that can, in practice, be computed, simulated, or analyzed by a digital computer, as well as those that can be learned from data.

A function is compositionally sparse if it factors through a directed acyclic graph (DAG) of intermediate variables with bounded local fan-in, independent of the overall input dimension. Under the standard Church–Turing viewpoint, "practically computable" means computable by a Turing machine in polynomial time. Every such function admits a representation by a bounded-fan-in compositional DAG and, consequently—as proven in Mhaskar and Poggio [129]—by a deep network with sparse connectivity.

Compositional sparsity thus provides a common structural explanation for three central properties of modern learning systems: (1) avoidance of the curse of dimensionality in approximation, (2) tractable optimization through locality of dependencies, and (3) improved generalization through reduced effective dimensionality. This perspective links practical computability, approximation theory, and statistical learning, suggesting that compositional sparsity is a fundamental and unifying

principle of machine learning [150, 146].

Deep networks approximate and learn high-dimensional functions that arise in vision, language, and scientific modeling. Surprisingly, they appear to overcome the so-called *curse of dimensionality*: despite operating on inputs with millions of dimensions—such as the function defining object classification in CIFAR images—they perform accurate learning and generalization without an explosion in the number of their weights. Their empirical success raises a foundational question: what structural property must a function possess to be realizable and learnable in practice by a deep network and thus by a computer?

Recent theoretical work suggests that the answer lies in compositional sparsity [150, 146, 41]. Functions that can be computed by a Turing machine within polynomial time—and thus by any realistic digital computer— admit representations as bounded-fan-in compositions of simpler subfunctions. This structure corresponds to a directed acyclic graph (DAG) whose nodes represent intermediate variables and whose local connectivity does not grow with input dimension.

In this view, the architectures of deep networks are not arbitrary engineering choices. Rather, they naturally reflect the sparse compositional structure of those functions that can be computed or learned in practice. The correspondence between compositional sparsity and practical computability provides a bridge between Turing complexity and modern learning theory. Following standard usage in theoretical computer science, we call a function *efficiently computable* if it can be computed by a Turing machine in polynomial time; in practice, this coincides with what can be computed by an ordinary digital computer. The key notions are summarized below.

Box 1 – Key definitions

Efficiently computable. A function f is said to be *efficiently computable* if there exists an algorithm on a standard digital computer—or equivalently, a Turing machine—that, given an input x encoded in n bits and a desired output precision ε (encoded in $\lceil \log_2(1/\varepsilon) \rceil$ bits), computes an ε-approximation to $f(x)$ in time $\mathrm{poly}(n, \log(1/\varepsilon))$. In theoretical computer science, this marks the boundary between functions that can be computed in practice and those that cannot. Throughout this article, we treat "efficiently computable" and "practically computable" as synonymous.

Compositionally sparse function. A function $f(x_1, \ldots, x_n)$ is *compositionally sparse* if it can be expressed as a composition of low-arity constituent functions arranged in a directed acyclic graph (DAG) of fixed depth. Formally, there exists a DAG G with depth L such that each node computes a function h_i that depends on at most k of its input coordinates, where k is a constant independent of n, and all other coordinates are passed forward unchanged. Thus each layer $h^{(\ell)} : \mathbb{R}^n \to \mathbb{R}^n$ modifies at most k coordinates of its input, with identity mappings on the remaining coordinates. The full function is given by

$$f(x) = h^{(L)} \circ h^{(L-1)} \circ \cdots \circ h^{(1)}(x).$$

This formulation avoids the implicit "deletion" of unused variables and corresponds exactly to the standard bounded-fan-in circuit model.

Curse of dimensionality. For a generic smooth function $f : \mathbb{R}^n \to \mathbb{R}$, the number of parameters $N(\varepsilon)$ required to approximate f with uniform error ε by standard (non-compositional) methods typically scales as

$$N(\varepsilon) \propto \varepsilon^{-n/s},$$

where s measures smoothness (for example, the number of bounded derivatives). When n is large, this dependence becomes prohibitive: for $n = 1000$ and $\varepsilon = 0.1$ with $s = 1$, one obtains $N \sim 10^{1000}$ parameters. This exponential growth in required resources is the classical *curse of dimensionality*.

28.1 Efficient Computability and Sparse Compositionality

A central observation is that any function f that is efficiently computable, in the sense of Box 1, can be represented, to arbitrary precision, by a circuit of polynomial size and bounded fan-in. Translating this into continuous mathematics yields a hierarchy of compositions of low-arity functions. The resulting function class—compositionally sparse functions—is exponentially smaller than the space of all continuous functions on high-dimensional domains, yet it contains those that can be evaluated or learned in practice [150, 146].

Formally, a function $f(x_1, \ldots, x_n)$ is compositionally sparse if there exists a DAG G with finite depth L and fan-in $k = O(1)$ such that each node computes a function h_i of at most k inputs from its predecessors and f equals the output at the final node. This structure corresponds to a layered composition

$$f(x) = h^{(L)} \circ h^{(L-1)} \circ \cdots \circ h^{(1)}(x),$$

where each $h^{(\ell)}$ acts on only a few variables. Such sparsity drastically reduces the number of parameters required to approximate f and constrains the space of admissible dependencies. This

exponential explosion of parameters with input dimension—known as the curse of dimensionality—is summarized in Box 1.

28.2 Approximation, Optimization, and Generalization

Compositional sparsity leads to quantitative advantages in all major aspects of learning:

1. **Approximation.** For functions with compositional depth L and arity k, deep networks achieve error $O(N^{-\alpha/k})$ using $O(N)$ parameters, while shallow or dense architectures require exponentially more [150, 129, 10].

2. **Optimization.** Because of the non-exponential number of parameters, gradient-based optimization avoids the curse of dimensionality. Note that while the target function is compositionally sparse, finding this structure via gradient descent often requires highly overparameterized, dense networks during training. This provides the optimization landscape necessary to locate the sparse solution, even if the final effective dimensionality of the learned function remains low.

3. **Generalization.** The effective dimensionality of the hypothesis space scales with the number of parameters for the "worst" subfunctions rather than the total number of parameters, yielding tighter bounds on sample complexity [60, 12, 100, 172, 190].

These effects emerge naturally from the DAG structure and do not depend on the specific activation or architecture details. The corresponding formal results are summarized in Box 2.

28.3 Relation to Prior Theories of Deep Learning

The principle of compositional sparsity extends and unifies several earlier theoretical accounts of deep learning. Classical approximation theory established that deep networks can represent certain function classes with exponentially fewer parameters than shallow networks [129, 10]. The present framework identifies the source of this efficiency: compositional structure limits the effective dimensionality of the function space. Similarly, circuit complexity results [73] demonstrate that functions computable by small-depth, bounded-fan-in circuits correspond to hierarchically compositional functions. Recent analyses in statistical learning [18, 159] emphasized hierarchical feature composition, but without connecting it to formal notions of practical computability or sample efficiency.

Compositional sparsity provides this missing link, showing that the same structural property explains advantages in representation, optimization, and generalization within a unified mathematical framework. The theorems above suggest that modern large-scale architectures benefit precisely because they learn and compose local constituents. Two prominent examples are transformer-based sequence models and diffusion generative models [192, 78, 178].

Implication for transformer MLP blocks (falsifiable prediction). If efficient computability implies compositional sparsity, then the *feed-forward (MLP) sublayers* in a transformer must implement *low-arity constituents*. Concretely, for each layer there exist permutations of coordinates under which the two linear maps $(W_{\text{in}}, W_{\text{out}})$ of the MLP can be written (up to an ε-approximation error) as *block-sparse/column-sparse* matrices with at most $O(k)$ nonzeros per column/row, where k is the constituent arity and is independent of the embedding dimension. Equivalently, the effective ℓ_0

(and hence ℓ_1) sparsity of the MLP weight matrices scales with k rather than with width. This yields a concrete, testable consequence: at fixed accuracy, one should be able to prune the MLP weights down to a sparsity level proportional to k (up to logarithmic factors), with minimal fine-tuning, and the resulting norm-based generalization bounds (Box 2, Theorem C) should tighten accordingly [60, 12, 100].

Box 2 – Main results on sparse compositionality

Theorem A (Mhaskar & Poggio, informal). *Sparse compositional functions escape the curse of dimensionality.*
If $f(x_1, \ldots, x_n)$ is a *compositionally sparse* function built from constituent functions in a Sobolev space $\mathbf{W}_s^k$, each depending on at most k variables (independent of n), then a deep network whose architecture mirrors this compositional structure can approximate f with error ε using $N_{\text{deep}}(\varepsilon) \propto \varepsilon^{-k/s}$ parameters, where s depends on the smoothness of the constituents. In contrast, shallow or dense architectures typically require $N_{\text{shallow}}(\varepsilon) \propto \varepsilon^{-\frac{n}{s}}$ parameters. Compositional sparsity thus removes the exponential dependence on n and breaks the curse of dimensionality [129].

Theorem B (informal, see Poggio [146]). *Efficient computability implies sparse compositionality.*
Any efficiently computable function admits a representation as a directed acyclic graph of bounded-fan-in constituent functions of polynomial size. Equivalently, every efficiently computable function can be approximated, to arbitrary precision, by a deep network whose connectivity is sparse and locally bounded.

Scope of the result. Following Poggio [146], we adopt the standard bit model of computation: an "efficiently computable" function $f : [0,1]^d \to \mathbb{R}^m$ is one for which there exists a deterministic Turing machine that, given an n-bit encoding of x and an accuracy parameter $\varepsilon > 0$ (encoded in $\lceil \log_2(1/\varepsilon) \rceil$ bits), runs in time $\text{poly}(n, \log(1/\varepsilon))$ and outputs an ε-approximation to $f(x)$. In Poggio [146] it is shown that such a machine can be unfolded into a family of polynomial-size, bounded-fan-in circuits that, in turn, can be compiled into deep neural networks with sparse connectivity and size and depth polynomial in $n + \log(1/\varepsilon)$.

Theorem C (informal, see Galanti et al. [60]). *Sparsity of the network's weight matrices implies better generalization bounds.*
For a deep network whose connectivity matches a compositionally sparse DAG, the empirical Rademacher bounds $\mathfrak{R}_S(\mathcal{F})$ is are significantly smaller than the standard Rademacher-based bounds for dense networks [12, 100, 172, 190]. For example, if a layer is convolutional and the connectivity is equivalent to non-overlapping patches, that is, each node in the DAG receives different inputs, its contribution to the Rademacher complexity is a factor $\frac{1}{\sqrt{N}}$ smaller when sparsity is taken into account than in standard bounds.

Theorem D (informal; see Poggio [146] and Malach [117]). *Efficient computability implies constituent learnability.*
For efficiently computable functions, the constituent functions are sparse and thus each of them is efficiently learnable from polynomially many examples. Moreover, the overall function can be learned by *local* training of these constituents, using data that only need to expose the relevant low-arity dependencies (as in next-token prediction or local denoising objectives). Efficient computability of the composite function implies *local learnability of its parts thus yielding practical learnability of the composite function*.

In short, *efficient computability predicts intrinsic sparsity of transformer MLP weights*, reflecting the low-arity structure of the underlying constituents. This prediction is consistent with emerging empirical evidence suggesting that large language models can be pruned to extreme levels of sparsity—up to 99% in some regimes—without significant performance degradation, provided the pruning aligns with the underlying compositional structure.

28.4 Discussion

Compositional sparsity offers a structural characterization of the class of functions that can be computed or learned in practice by digital computers. Under the Church–Turing viewpoint, every efficiently (and thus practically) computable function is compositionally sparse and can therefore be represented by a deep network with bounded-fan-in connectivity [146, 150, 41]. This perspective unifies several empirical and theoretical observations: that deep architectures overcome the curse of dimensionality, that their optimization is tractable despite nonconvexity, and that their generalization performance exceeds what classical capacity measures would predict [129, 10, 60, 12, 100, 172, 190].

For artificial intelligence systems, the scope of this theory is clear. All AI algorithms are implemented on digital computers and are therefore efficiently computable by construction. As a consequence, the theory developed here applies fully to artificial learning systems: any function realized by an AI model must admit a compositionally sparse representation, even if that structure is not explicit in the trained architecture. From this perspective, the success of modern foundational models is not accidental but reflects their ability to approximate and compose low-arity constituent functions drawn from this restricted, efficiently computable function class.

The situation is more subtle for biological intelligence. A crucial distinction in this context is between *Turing computability* and *efficient* Turing computability. The physical Church–Turing thesis asserts that the behavior of any physically realizable system can, in principle, be simulated by a Turing machine. However, this does not imply that such a simulation can be carried out with resources that scale polynomially in the relevant problem size. Many physical and biological systems may therefore be fully Turing computable yet computationally intractable, requiring exponential time or space to simulate with useful precision.

From this perspective, it is not obvious that all functions implemented by the brain are efficiently computable in the sense adopted here. Higher-level cognitive functions such as language, abstraction, and reasoning exhibit clear signatures of hierarchical and compositional structure and are natural candidates for efficient computation. In contrast, other processes—such as affect, emotions, drives, homeostatic regulation, and internal bodily states—may rely on tightly coupled, state-dependent, or continuous dynamics that do not admit an efficient sparse compositional representation. Such processes may remain fully Turing computable, yet fall outside the class of functions that can be approximated or learned efficiently by digital systems.

This distinction suggests an apparent but instructive irony. It may be easier to replicate and understand higher-level cognitive functions using digital computers than more basic, evolutionarily older ones. The reason is not that language or reasoning are simpler in an absolute sense, but that they may be more structured, more compressible, and more amenable to sparse compositional representations. Conversely, functions that are behaviorally simple or automatic may nonetheless be computationally inefficient, poorly modularized, or resistant to decomposition into reusable constituents.

Importantly, this perspective does not imply that such functions are non-computable or beyond physical realization. Rather, it highlights a distinction between efficient computability and mere

computability, and between learnability from data and implementation by embodied, closed-loop biological systems. From this viewpoint, the current strengths and limitations of AI systems are not paradoxical but follow naturally from the structural constraints imposed by efficient computation.

Ultimately, sparse compositionality points toward a unifying theory of machine learning and computation—one that explains what digital systems can learn well, what they struggle with, and why. Whether sparse compositionality also captures the full range of biological intelligence remains an open question, but one that can now be stated with greater precision.

Box 3 – Modern architectures as realizations of efficiently computable, compositionally sparse functions

Scope clarification. The interpretations in this box concern those aspects of modern learning architectures that correspond to *efficiently computable* functions. Since all artificial intelligence systems are implemented on digital computers, the functions they realize necessarily lie within the class of efficiently computable—and hence compositionally sparse—mappings. The discussion below does not claim that such architectures capture all aspects of biological intelligence, but illustrates how sparse compositionality manifests in current large-scale models.

Informal principle. Contemporary large-scale models, including transformers and diffusion networks, can be interpreted as concrete realizations of compositionally sparse functions, in the sense that their global input–output behavior arises from the composition of locally learnable, low-arity constituent functions.

Transformers. A transformer implements a hierarchy of low-arity constituent functions operating on token embeddings. Multi-head attention computes context vectors c_t from bounded subsets of positions or features, and the output heads map c_t to conditional probabilities $p(x_t \mid c_t)$. Training by next-token prediction,

$$\min_{\theta} \mathbb{E}[-\log p_\theta(x_t \mid x_{<t})]\,,$$

corresponds to fitting local conditional constituents [117]. The global mapping from prompts to continuations can therefore be viewed as a composition of locally learnable subfunctions, accounting for the emergence of coherent text generation within the class of efficiently computable mappings [150, 146].

Diffusion models. Diffusion generative models construct samples by composing a sequence of local denoising steps [178, 78]. Let $(X_t)_{t=0}^{T}$ be a Markov chain with known forward kernels $q(x_t \mid x_{t-1})$ and learnable reverse kernels $p_\theta(x_{t-1} \mid x_t)$. Each reverse step acts on a low-dimensional state (x_t, t) and can be represented by a constituent neural function in a compositional DAG. Training objectives such as denoising score matching fit these local transitions; their composition yields the global generator

$$x_T = z \mapsto x_{T-1} \mapsto \cdots \mapsto x_0 = G_\theta(z).$$

Both architectures illustrate the same principle: high-dimensional predictive or generative behavior can arise from the composition of locally learnable, low-arity constituents, consistent with the constraints imposed by efficient computation and sparse compositionality [150, 41].

CHAPTER 29

Mixture of Experts

Manifolds, compositionality, mixture of experts: how are they related?

29.1 Introduction

We start by summarizing the main results of a recent paper, titled "On the Expressive Power of Mixture-of-Experts for Structured Complex Tasks". The paper investigates the expressive power of Mixture-of-Experts (MoE) networks for modeling complex tasks. While MoEs have achieved significant empirical success, particularly in Large Language Models, their theoretical foundations remain poorly understood [175, 51]. The authors conduct a systematic study of MoEs under two common structural priors: **low-dimensionality** and **sparsity**. They demonstrate that both shallow and deep MoEs can overcome the curse of dimensionality to efficiently approximate complex functions.

29.2 Shallow MoE Networks (Low-Dimensional Structure)

The authors prove that shallow MoE networks can efficiently approximate functions supported on low-dimensional manifolds.

- **Problem Decomposition:** The task reduces to collecting simpler approximation subproblems localized on subregions, combined with an assignment problem mapping inputs to the correct region.
- **Mechanism:**
  - *Expert Networks:* Approximate localized subfunctions.
  - *Gating Mechanism:* Ensures correct input-to-expert assignment.
- **Theoretical Result:** A depth-2 MoE network with E experts can approximate a target function f on a manifold $\mathcal{M}$ with an error rate governed by the intrinsic dimension d rather than the ambient dimension D [196]:

$$\|f - \Psi\|_{L_\infty(\mathcal{M})} \leq \max_{i\in[E]} \tilde{\mathcal{O}}\left(m^{-\frac{\kappa(f|_{U_i})}{d}\wedge\frac{1}{2}}\right) \tag{29.1}$$

  where m is the expert width and κ is the smoothness.

29.3 Deep MoE Networks (Compositional Sparsity)

The paper shows that deep MoEs can approximate piecewise functions exhibiting **compositional sparsity**, where subtasks depend on small subsets of inputs and are hierarchically composed [129].

- **Exponential Capacity:** A depth-$\mathcal{O}(L)$ MoE with E experts per layer can approximate functions with E^L distinct pieces, exhibiting an exponential number of structured tasks.
- **Comparison:** This far surpasses the naive limitation of $\mathcal{O}(LE)$ distinct regions if experts were used independently.
- **Mechanism:** The network depth L enables hierarchical composition, while the expert count E enables subtask specialization.
- **Theoretical Result:** For a compositionally sparse target function f (a hierarchical composition of L layers of constituent functions h_i with intrinsic dimension d^*), a deep MoE Ψ with depth L and width m achieves an approximation rate independent of the ambient dimension [196]:

$$\|f - \Psi\|_{L_\infty} \leq \tilde{\mathcal{O}}\left(L \cdot m^{-\frac{\kappa}{d^*}}\right) \tag{29.2}$$

This confirms that deep MoEs avoid the curse of dimensionality by exploiting the low-dimensional nature of the constituent sub-problems, even when the global function is highly complex. The specific result is Theorem 4.8. *Let $\mathcal{M}$ be a compact, d-dimensional smooth manifold in $\mathbb{R}^D$, with a regular atlas $\{(U_i, \phi_i)\}_{i\in[E]}$ (Definition 4.7). Let the target function $f : \mathcal{M} \to \mathbb{R}$. Then for any $m \geqslant \Omega\left(E^2\right)$, there exists a depth- 2 MoE network $\Psi \in \mathcal{H}_{3,m}^{2,E}$, with E experts per layer, each being is a 3-layer m-width dense networks, such that:*

$$\begin{aligned}
&\|f - \Psi\|_{L_\infty(\mathcal{M})} \leqslant \max_{i\in[E]} \mathcal{E}_{3,m}^{\mathrm{FFN}}\left(f|_{U_i}\right) \\
&\leqslant \max_{i\in[E]} \underbrace{\mathcal{E}_{2,m}^{\mathrm{FNN}}\left(f|_{U_i} \circ \phi_i^{-1}\right)}_{\text{approximate a low-dimensional function}} + \max_{i\in[E]} \underbrace{\mathcal{E}_{2,m}^{\mathrm{FFN}}\left(\phi_i\right)}_{\text{approximate a high-order smooth map}} \left\| f|_{U_i} \circ \phi_i^{-1} \right\|_{\mathcal{C}^1\left([0,1]^d\right)} \\
&\leqslant \max_{i\in[E]} \tilde{\mathcal{O}}\left(m^{-\frac{\kappa\left(f|_{U_i}\right)}{d} \wedge \frac{1}{2}}\right).
\end{aligned}$$

Theorem 4.8 demonstrates that depth- 2 MoE networks can efficiently approximate functions supported on low-dimensional manifolds. The total approximation error decomposes into two components: (i) approximation of low-dimensional target functions $f|_{U_i} \circ \phi_i^{-1}$'s; (ii) approximation of smooth coordinate maps ϕ_i 's. Both subproblems are significantly simpler than approximating the original high-dimensional function directly, enabling MoE networks to overcome the curse of dimensionality.

- **Example:** A network with math and language experts can solve combinatorial tasks (e.g., $3 \times 3 = 9$ tasks like "English Geometry" or "French Algebra").

29.4 Unified Insights

- **Structure Discovery:** MoEs naturally discover underlying structural priors (sparsity or low-dimensionality) and decompose them into simpler subproblems.

- **Non-Linear Gating:** Since partition functions are generally nonlinear, the authors suggest incorporating nonlinearity directly into the gating mechanism for better performance.
- **Architectural Variants:** The analysis supports alternating MoE/dense layers or architectures with shared plus routed experts.
- **Auto-encoding Experts:** The theory motivates replacing dense experts with "Encoder + Low-dim Network" structures, reducing parameters from $\mathcal{O}(D^2)$ to $\mathcal{O}(d^2)$.

The work provides theoretical justification for the efficiency of MoEs, establishing that they are particularly effective for structured tasks by exploiting low intrinsic dimensionality and compositional sparsity.

29.5 Connections with Compositional Sparsity Framework

This paper acts as a direct mathematical validation of the **compositional sparsity framework**. It provides a concrete proof that Mixture-of-Experts (MoE) architectures are one of the "natural" implementations of this theoretical framework.

The paper essentially proves that MoEs can overcome the **Curse of Dimensionality** by exploiting precisely the structure of compositional sparsity: functions that are hierarchical compositions of simpler, low-dimensional constituent functions.

Here is the step-by-step mapping between the paper's findings and the Compositional Sparsity framework:

29.5.1 The Core Alignment: Hierarchical Decomposition

- **The Compositional Sparsity Framework:** Posits that high-dimensional functions in the real world are actually formed by a hierarchy of constituent functions, each depending on a small subset of variables (sparse dependency).
- **The Paper:** Explicitly defines "Piecewise functions with compositional sparsity" in Section 5.1 [196]. It models complex tasks as $f(x) = f_{out}(f_{1,i_1}(x_1), \ldots, f_{L,i_L}(x_L))$.
- **The Connection:** The paper treats every "expert" in a deep MoE as one of the "constituent functions" in the framework. A deep MoE network is effectively a dynamic realization of a computation graph where the nodes (experts) are selected conditionally.

29.5.2 Overcoming the Curse of Dimensionality

- **The Compositional Sparsity Framework:** Deep networks avoid the curse of dimensionality because they approximate the constituent functions (which are low-dimensional) rather than the global high-dimensional function [154].
- **The Paper:** Proves this specifically for MoEs in **Theorem 5.2**. It shows the approximation error depends on the intrinsic dimension d_l of the sub-manifolds, not the ambient dimension D.
  - *Key Quote:* "The approximation rate is... $\tilde{\mathcal{O}}(m^{-\frac{\kappa}{d_l}})$... which avoids the curse of dimensionality."

29.5.3 The "Exponential Capacity" Extension

The paper adds a crucial layer to the standard compositional sparsity framework: **Routing**.

- Standard compositional sparsity (e.g., in dense networks) relies on fixed constituent functions.
- The paper shows that by *routing* inputs, an MoE with linear depth L and width E can model **exponentially many (E^L)** distinct compositional functions.
- **Theoretical Significance:** This suggests that MoEs are arguably *more* efficient at implementing compositional sparsity than dense networks because they can dynamically switch the "constituent functions" at every node of the hierarchy based on the input region.

29.5.4 Direct Mapping of Terminology

Concept in Compositional Sparsity Framework	Equivalent in Wang & E (2025) [196]
Constituent Function	**Expert Network** ($f_{l,i}$)
Sparsity (Local Dependency)	**Manifold Support** (Experts act on local regions U_i)
Hierarchical Composition	**Depth** (Stacking MoE layers)
Selector / DAG Structure	**Gating Mechanism** (Dynamically builds the graph)

29.6 Summary

The paper by Wang and E serves as evidence that MoEs are an explicit architectural realization of compositional sparsity. While dense networks *can* learn these functions, this paper proves MoEs are *designed* to model them by explicitly assigning distinct experts to the distinct "pieces" of the compositionally sparse target function.

29.7 DeepSeek-V3 MoE Architecture and Training Methodology

We look here at a specific example of an implemented architecture.

29.7.1 Architectural Composition

The DeepSeek-V3 model comprises a total of 61 transformer blocks, structured to integrate dense processing with sparse, fine-grained specialization.

- **Dense Layers:** The first 3 blocks ($1-3$) are standard dense transformer layers. In these initial stages, all parameters are active for every token to establish a foundational representation.
- **MoE Layers:** The subsequent 58 blocks ($4-61$) utilize the DeepSeekMoE architecture.

Within each of the 58 MoE blocks, the model employs a distinct configuration:

- **Expert Allocation:** Each block contains 256 fine-grained Routed Experts and 1 designated Shared Expert.
- **Activation Strategy:** For any given token, the Shared Expert is always active. Additionally, the gating mechanism selects the top-8 Routed Experts based on relevance.

29.7.2 Training Dynamics

The experts and the gating networks (routers) are trained jointly from scratch using a specialized regimen to ensure efficiency and balance.

Expert Training

Experts are trained via conditional computation. During the forward pass, only the selected experts (Shared + Top-8) process the input. Consequently, during backpropagation, gradient updates are applied *only* to these active experts. This sparsity forces the experts to specialize in distinct patterns or domains over time, as they are only updated when the router identifies them as relevant.

Gating and Load Balancing

To prevent "expert collapse"—where the router over-utilizes a small subset of experts—DeepSeek-V3 employs an **Auxiliary-Loss-Free Load Balancing** strategy. Unlike traditional methods that add a penalty term to the loss function (which can degrade model performance), DeepSeek-V3 dynamically adjusts a bias term for each expert. If an expert is overloaded, its bias is reduced, lowering its selection probability without altering the primary training objective.

Multi-Token Prediction (MTP)

The model training is augmented by a Multi-Token Prediction objective. Instead of predicting only the single next token t_{i+1}, the model is trained to predict sequential tokens (e.g., t_{i+1} and t_{i+2}) simultaneously. This provides a denser gradient signal to the experts, encouraging the development of stronger reasoning and planning capabilities.

More on DeepSeek-V3 Feed-Forward Network (DeepSeekMoE)

In DeepSeek-V3, the FFN is replaced by a fine-grained Mixture-of-Experts (MoE) layer consisting of two types of experts:

- **Shared Experts:** Always active for every token (captures common knowledge).
- **Routed Experts:** Selectively active based on a learned gating mechanism.

1. The General Equation

For an input latent vector $\mathbf{u}_t$, the output $\mathbf{h}'_t$ is the sum of the shared experts and the selected routed experts:

$$\mathbf{h}'_t = \mathbf{u}_t + \underbrace{\sum_{i=1}^{N_s} \text{FFN}_i^{(s)}(\mathbf{u}_t)}_{\text{Shared Experts}} + \underbrace{\sum_{i=1}^{N_r} g_{i,t} \cdot \text{FFN}_i^{(r)}(\mathbf{u}_t)}_{\text{Routed Experts}}$$

2. The Expert Architecture (SwiGLU)

Unlike the standard "1 ReLU" structure, every single expert (both shared and routed) in DeepSeek-V3 uses the **SwiGLU** architecture, which requires **three** linear projections (W_g, W_{in}, W_{out}) rather than two:

$$\text{FFN}_i(\mathbf{u}_t) = (\text{Swish}(\mathbf{u}_t W_{g,i}) \odot (\mathbf{u}_t W_{in,i}))\, W_{out,i}$$

3. Gating and Routing

The routing score $g_{i,t}$ determines which "Routed Experts" are active. DeepSeek-V3 selects the top-K experts (e.g., $K = 8$) based on affinity scores $s_{i,t}$:

$$s_{i,t} = \text{Sigmoid}(\mathbf{u}_t \cdot \mathbf{e}_i) \tag{29.3}$$

$$g_{i,t} = \begin{cases} \frac{s_{i,t}}{\sum_{j \in \text{Top-K}} s_{j,t}} & \text{if } i \in \text{Top-K} \\ 0 & \text{otherwise} \end{cases} \tag{29.4}$$

29.7.3 Summary of Specs

- **Architecture:** DeepSeekMoE (Shared + Routed)
- **Activation:** SwiGLU (SiLU gating), *not ReLU*
- **Total Experts:** 1 Shared + 256 Routed, each expert has dim 2048 whereas the total input dim is 7168 (sixe of token) which is projected down to each expert. When you sum up the active path (1 shared + 8 routed experts), the effective intermediate bandwidth per token is roughly 9×2,048=18,432, which matches the dense layer size exactly. The token size is compressed for storage to 576.
- **Active Experts:** 1 Shared + 8 Routed (per token)
- **Layers:** 61 Total (Layers 1-3 are dense; Layers 4-61 are MoE)

CHAPTER 30

Takens Theorem, RNNs and Associative Turing Machines

Takens theorem justifies RNNs. However RNNs are **not** *associative Turing Machines. This chapter describes an old theorem about dynamical systems that suggests RNNs as a good preditor. However the computational power of* autonomous recurrent neural networks *(RNNs) [176] is strictly less than Turing Machines: they are more directly related to finite-state machines. The results here motivate the explicit-memory construction of Associative Turing Machines (ATMs) in chapter 3.*

30.1 Takens' Embedding Theorem

Query: Is there a connection between token/state embeddings and RNNs?

This relates to **Takens' Embedding Theorem** (1981) from dynamical systems theory.

Theorem 14 (Takens' Embedding Theorem, 1981)**.** *Let M be a compact manifold of dimension $d \geq 1$. Consider a dynamical system given by a C^2 diffeomorphism $\phi : M \to M$ and a C^2 measurement function $y : M \to \mathbb{R}$.*

Define the ***delay coordinate map*** *$\Phi_{(\phi,y)} : M \to \mathbb{R}^{2d+1}$ by:*

$$\Phi_{(\phi,y)}(x) = \Big(y(x), y(\phi(x)), y(\phi^2(x)), \ldots, y(\phi^{2d}(x))\Big) \tag{30.1}$$

Then, for generic pairs (ϕ, y) (i.e., for an open and dense set of pairs in the C^2 topology), the map $\Phi_{(\phi,y)}$ is an ***embedding****.*

Specifically, the image $\Phi_{(\phi,y)}(M)$ is a submanifold of $\mathbb{R}^{2d+1}$ that is diffeomorphic to M, and the dynamics on the delay coordinates are topologically conjugate to the original dynamics on M.

30.1.1 Relevance to RNNs

An RNN effectively acts as a functional approximator of the Takens embedding map.

$$\text{History } (y_t, y_{t-1}, \ldots) \xrightarrow{\text{RNN}} h_t \approx \text{True Latent State } x_t \tag{30.2}$$

This explains why RNNs can predict chaotic systems without knowing the underlying differential equations: the hidden state h_t reconstructs the topology of the system's attractor from the observation history.

30.2 Autonomous RNN model

We consider an autonomous RNN of the form

$$h_{t+1} = \phi(Wh_t + b), \qquad h_t \in \mathbb{R}^d, \tag{30.3}$$

where $W \in \mathbb{R}^{d\times d}$, $b \in \mathbb{R}^d$, and ϕ is a fixed componentwise nonlinearity (e.g. $\tanh$, sigmoid, or ReLU). No external input is provided; all computation is carried by the state.

Unrolling (30.3) shows that the network computes an iterated map

$$h_T = F^{(T)}(h_0), \qquad F(h) := \phi(Wh + b),$$

so time corresponds to depth, with weight sharing across iterations.

30.2.1 Simulation of finite-state machines

Let $\mathcal{A} = (Q, \delta)$ be a deterministic finite-state machine with $|Q| = m$ and transition function $\delta : Q \to Q$.

State embedding. Embed each state $q_i \in Q$ as a vector $e_i \in \mathbb{R}^m$, for example as canonical basis vectors or vertices of a simplex.

Transition realization. Since δ is a finite lookup table, there exist parameters (W, b) such that

$$\phi(We_i + b) \approx e_{\delta(i)}, \qquad i = 1, \dots, m,$$

with arbitrarily small approximation error.

Conclusion. For suitable initialization $h_0 = e_{q_0}$,

$$h_t \approx e_{q_t} \quad \text{for all } t,$$

and the RNN simulates the finite-state machine exactly up to a prescribed precision.

Proposition 7. *Any deterministic finite-state machine can be simulated by an autonomous RNN of the form* (30.3)*, with arbitrary accuracy.*

30.2.2 Turing machines restricted to a finite time horizon

Let T be a deterministic single-tape Turing machine, and suppose its execution is restricted to at most T steps on a given input.

Finite configuration space. In T steps, the machine can visit only $O(T)$ tape cells. The set of reachable configurations

$$\mathcal{C}_T = \{(q, h, \tau)\}$$

(control state, head position, and tape contents) is therefore finite.

The restricted Turing machine induces a transition function

$$C_{t+1} = \Delta(C_t), \qquad C_t \in \mathcal{C}_T,$$

which is a finite-state dynamical system.

Encoding and simulation. As in the finite-state case, encode each configuration $C_i \in \mathcal{C}_T$ as a vector $v_i \in \mathbb{R}^d$. Since $\mathcal{C}_T$ is finite, there exist RNN parameters such that

$$\phi(Wv_i + b) \approx v_{\Delta(i)}.$$

Proposition 8. *For any deterministic Turing machine and any fixed time horizon T, there exists an autonomous RNN that simulates the machine's computation for all $t \leq T$, up to arbitrary precision.*

30.2.3 Why the time bound is essential

The restriction to a fixed horizon T is fundamental.

Unbounded memory growth. An unrestricted Turing machine may access arbitrarily many tape cells. A fixed finite-dimensional state $h_t \in \mathbb{R}^d$ cannot robustly encode an unbounded tape without increasing precision requirements.

Error accumulation. Let L be a Lipschitz constant of F. Perturbations satisfy

$$\|h_t - \tilde{h}_t\| \leq L^t \|h_0 - \tilde{h}_0\|,$$

so stable long-term simulation requires either contraction or exponentially growing precision.

Conclusion. Autonomous RNNs can simulate Turing machines only in a *time-bounded* sense, where the computation reduces to a large but finite-state transition system.

30.2.4 Comparison with Associative Turing Machines

This limitation motivates the explicit-memory construction of ATMs.

Model	Memory	TM simulation	Robustness
Autonomous RNN	implicit (weights)	FSM, TM up to fixed T	limited
RNN with input	implicit + stream	partial	limited
ATM x	explicit associative memory	full TM	polynomial overhead

ATMs avoid state compression by storing tape symbols and transition logic explicitly in associative memory. Each computational step consists of a similarity-based read followed by a local update, preventing unbounded growth of state dimension or precision.

Lemma 10 (State-dependent simulation by a fixed recurrent map)**.** *Let $\mathcal{S}$ be a finite set of states and let*

$$\Delta : \mathcal{S} \to \mathcal{S}$$

be an arbitrary deterministic transition function. Then there exists a dimension d and a continuous map

$$F : \mathbb{R}^d \to \mathbb{R}^d$$

(realizable, for example, by a single-layer recurrent neural network with a standard nonlinearity) together with an injective encoding

$$\mathrm{Enc} : \mathcal{S} \to \mathbb{R}^d$$

such that

$$F(\mathrm{Enc}(s)) = \mathrm{Enc}(\Delta(s)) \quad \textit{for all } s \in \mathcal{S}.$$

In particular, iteration of the same *transition map F simulates the (possibly state-dependent) transition system Δ exactly on the embedded states.*

Proof. Since $\mathcal{S}$ is finite, choose an injective encoding $\mathrm{Enc}(s) = v_s \in \mathbb{R}^d$, for example as vertices of a simplex or canonical basis vectors. Define a target function

$$G(v_s) := v_{\Delta(s)} \qquad \text{for all } s \in \mathcal{S}.$$

Because G is specified only on a finite set of input points, there exists a continuous extension $F : \mathbb{R}^d \to \mathbb{R}^d$ such that $F(v_s) = G(v_s)$ for all s. Moreover, such an F can be realized (or approximated arbitrarily closely) by a neural network of fixed architecture with standard nonlinearities, by universal approximation on finite domains.

Therefore, for any initial state $s_0 \in \mathcal{S}$,

$$F^{(t)}(\mathrm{Enc}(s_0)) = \mathrm{Enc}(\Delta^{(t)}(s_0)) \quad \text{for all } t \geq 0,$$

where $F^{(t)}$ denotes t-fold composition of F. The transition rule is time-invariant; all effective state dependence is encoded in the location of the state vector in $\mathbb{R}^d$. □

Takeaway. Autonomous RNNs demonstrate that iteration of a fixed nonlinear map suffices for finite-state and time-bounded computation. Associative Turing Machines extend this idea by introducing explicit, addressable memory, enabling robust and efficient simulation of unbounded algorithms.

Appendix

CHAPTER A

Appendix: Stability, ERM, and the Foundations of Learnability

The classical analysis of supervised learning is often framed in terms of capacity measures such as VC dimension or Rademacher complexity. An alternative but equivalent viewpoint focuses on the stability *of the learning algorithm with respect to perturbations of its training data. In this chapter we recall a statistical notion of stability based on leave-one-out cross-validation (CV_{loo}) and the theorem of Mukherjee, Niyogi, and Poggio that characterizes consistency of empirical risk minimization (ERM) in terms of this stability.*

A.1 Learning setup and ERM

Let $\mathcal{X}$ be an input space and $\mathcal{Y}$ an output space (e.g. $\mathcal{Y} = \{0, 1\}$ or $\mathbb{R}$). A learning problem is specified by an unknown distribution P on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. We observe an i.i.d. training sample

$$S_n = (z_1, \ldots, z_n), \qquad z_i = (x_i, y_i) \sim P,$$

and choose a hypothesis $h \in \mathcal{H}$ from a hypothesis class $\mathcal{H}$.

A loss function $\ell : \mathcal{H} \times \mathcal{Z} \to [0, 1]$ (or more generally a bounded nonnegative function) measures prediction error. The *true risk* of h is

$$L(h) \;=\; \mathbb{E}_{z \sim P}[\ell(h, z)],$$

and the *empirical risk* on S_n is

$$L_n(h) \;=\; \frac{1}{n} \sum_{i=1}^{n} \ell(h, z_i).$$

The empirical risk minimization (ERM) principle selects

$$\hat{h}_n \;\in\; \arg\min_{h \in \mathcal{H}} L_n(h).$$

We say that ERM over $\mathcal{H}$ is *(universally) consistent* if

$$L(\hat{h}_n) \;\xrightarrow[n \to \infty]{P}\; \inf_{h \in \mathcal{H}} L(h),$$

for every distribution P on $\mathcal{Z}$.

A.2 $\mathrm{CV}_{\mathrm{loo}}$ stability

Let $\mathcal{A}_n$ denote a (possibly randomized) learning algorithm that, given a sample S_n, outputs a hypothesis $\mathcal{A}_n(S_n) \in \mathcal{H}$. For $i \in \{1, \dots, n\}$, let $S_n^{(i)}$ denote the sample with the i-th point removed:

$$S_n^{(i)} = (z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_n).$$

Intuitively, *leave-one-out* stability requires that removing a single training point does not significantly change the prediction on that point.

Definition 16 ($\mathrm{CV}_{\mathrm{loo}}$ stability)**.** *A sequence of algorithms $(\mathcal{A}_n)_{n\geq 1}$ is said to have CV_{loo} stability with parameters (β_n) if*

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\Big[\big|\ell(\mathcal{A}_n(S_n), z_i) - \ell(\mathcal{A}_{n-1}(S_n^{(i)}), z_i)\big|\Big] \;\leq\; \beta_n,$$

for all n, where the expectation is over the i.i.d. draw of S_n (and any internal randomness of $\mathcal{A}_n$). We say the algorithm is asymptotically CV_{loo}-stable *if $\beta_n \to 0$ as $n \to \infty$.*

There are several related stability notions (uniform stability, hypothesis stability, etc.). The $\mathrm{CV}_{\mathrm{loo}}$ form is directly linked to cross-validation and has an especially clean connection to ERM consistency.

A.3 Stability and generalization

We now state the main result of Mukherjee, Niyogi, and Poggio. In the wording below, we preserve the original informal summary of the theorem.

Theorem 15 (Mukherjee–Niyogi–Poggio)**.** *We propose a statistical form of stability, defined in terms of the prop- erty of cross-validation leave-one-out (CVloo) stability, and*

1. *show that it is sufficient, in general, for generalization, that is convergence in probability of the empirical error to the expected error, for any algorithm satisfying it.*
2. *Our second observation is that for bounded loss classes, CVloo stability is necessary and sufficient for consistency of ERM.*

Here "generalization" in part (a) means that for any asymptotically $\mathrm{CV}_{\mathrm{loo}}$-stable algorithm $(\mathcal{A}_n)$,

$$L(\mathcal{A}_n(S_n)) - L_n(\mathcal{A}_n(S_n)) \;\xrightarrow[n\to\infty]{P}\; 0.$$

Part (b) shows that, for bounded loss classes, ERM is consistent *if and only if* it is $\mathrm{CV}_{\mathrm{loo}}$-stable. Thus stability is not merely sufficient; it is an exact characterization of when ERM behaves well.

A.4 Stability as a modeling requirement

From a modeling standpoint, Theorem 15 has a simple but strong implication:

- If we take ERM (or approximate ERM) as the fundamental inductive principle underlying learning, then *stability is required* for consistency.

- Instability—in the sense that removing or slightly perturbing a single training point can qualitatively change the predictor—rules out consistency of ERM for bounded losses.

In the remainder of the book we will use this observation in the following direction: *we impose stability as a requirement* on realistic learning problems, and ask what structural properties of target functions follow from this requirement. In particular, in the next chapter we argue that stability forces a kind of *genericity* of targets, and that this genericity in turn makes optimization by gradient-based methods possible.

CHAPTER B

TechnicalNote: Compositionality in Machine Learning and Physics

We formalize a bridge between efficient computation and learnability via bounded–fan-in compositional structure. On the learning side, we prove an ML Approximation Theorem*: when a target factors through a bounded–fan-in DAG with maximal local arity d^*, the relevant approximation and sample–complexity exponents depend on d^* rather than the ambient dimension d. On the physics side, we state a* finite-horizon compilation principle *(explicitly conditional): whenever a finite-horizon evolution map admits a* uniform *simulator using a finite set of bounded-arity primitives, the map compiles to a P-uniform bounded–fan-in DAG whose size is proportional to the simulator cost. Illustrative cases include discrete-time lattice-local models and k-local quantum dynamics compiled into uniform two-qubit circuits. Limitations include chaotic sensitivity, long-range or dense couplings, precision constraints, and thermodynamic considerations; all statements are finite-precision, finite-horizon, and conditional on uniform simulability.*

B.1 Introduction

Modern deep learning owes as much to *structure* as to *scale*. Many of the functions we hope to learn are not generic mappings on high-dimensional spaces but the outputs of computations that decompose into smaller, interacting parts arranged hierarchically. This observation has two complementary faces—algorithmic and physical.

The algorithmic view From the standpoint of computation, any function evaluable by a polynomial-time Turing machine admits a polynomial-size Boolean circuit of bounded fan-in [**Arora_Barak_2009**, **sipser1996intro**]. Interpreting such a circuit as a directed acyclic graph (DAG) exposes its *compositional structure*: each node is a local operation on a small number of inputs, and the full computation emerges from a finite sequence of such local interactions. In the continuous setting, this structure is mirrored by deep networks, whose layers correspond to circuit levels. Thus, efficient Turing computability implies that the underlying function is *compositionally sparse*: at each stage it depends on a bounded number of variables. In this sense, the success of deep architectures reflects the structure of efficient computation.

The physical view In physical systems, efficient Turing computability cannot be assumed *a priori*. What is often available is *finite-horizon locality*: many models bound the spread of influence over

time (e.g., relativistic light cones or Lieb–Robinson bounds in quantum lattices [**lieb1972finite**, **NachtergaeleSims2010**]). When such locality is paired with a *uniform* simulator built from bounded-arity primitives, the finite-horizon input–output map can be *compiled* into a bounded–fan-in DAG whose structure reflects the causal neighborhood at horizon T. This is an *algorithmic* statement about compilation, not a universal physical law about bounded *physical* degree: long-range or dense couplings (e.g., Coulomb, mean-field) may violate bounded degree even though a uniform simulator—and hence a bounded–fan-in DAG—still exists with polynomial overhead.

The resulting lesson is conditional but practical: whenever finite-horizon locality and uniform simulability hold, the effective complexity is governed not by the ambient dimension d but by the *algorithmic* largest local arity d^* (the size of the neighborhoods feeding each compiled node) and by the horizon/depth T over which interactions unfold. Limits remain: chaotic sensitivity can amplify constants with T, exact-real outputs are excluded (finite precision), and thermodynamic-limit properties need not be decidable.

From observation to theorem. We make the intuition precise via two complementary results, separating what is provable in general from what holds under explicit hypotheses. On the learning side, the *ML Approximation Theorem* shows that when a target factors through a bounded–fan-in DAG, the usual exponent d/r (ambient dimension over smoothness) is replaced by d^*/r, where d^* is the maximal local arity in the DAG. On the physics side, rather than asserting a universal causal–locality law, we prove a *finite-horizon compilation principle*: whenever a finite-horizon evolution map admits a *uniform* simulator using a fixed finite set of bounded-arity primitives with cost $s(n, T, \varepsilon)$, it compiles to a *polynomial-time uniform* bounded–fan-in DAG of size $O(s(n, T, \varepsilon))$. *Examples* (not universal claims) include discrete-time lattice-local updates with finite propagation and k-local quantum dynamics compiled to two-qubit circuits. Together, these results identify a common mathematical structure—bounded local arity d^* and horizon/depth T—that governs approximation and learnability; the physics connection is *conditional* (finite precision, finite horizon, uniform simulability), not automatic.

Historical background and conceptual context

The route from Turing machines to circuits is a cornerstone of complexity theory and underlies the definition of major classes such as P and NC [**Arora_Barak_2009**]. Deep learning, viewed through this lens, can be seen as the continuous analogue of polynomial-size, bounded–fan-in circuit computation [**poggio_deep_shallow_2017**, 126]. Approximation theory has long established that deep networks approximate hierarchical functions exponentially more efficiently than shallow ones, and recent results on compositional sparsity [**poggiofraser2024**, **Poggio2025**] formalize this intuition.

In physics, locality appears in multiple guises. In classical mechanics, it is encoded in differential equations: the rate of change at a point depends on derivatives—i.e., infinitesimal neighborhoods. In electromagnetism, Maxwell's equations forbid instantaneous action at a distance; disturbances propagate at finite speed c. In quantum lattice systems, the Lieb–Robinson theorem [**lieb1972finite**, **NachtergaeleSims2010**, **bravyi2006lieb**] bounds the velocity of information propagation, establishing an effective light cone even in nonrelativistic settings. Locality, in all these forms, implies a bounded causal neighborhood at any finite time horizon—an analogue of bounded fan-in in computation.

A more speculative connection has also captured attention: the *simulation hypothesis*. If every physical process were efficiently simulable by a digital computer, then the universe would instantiate an efficiently computable function, and its regularities would follow from the same com-

positional constraints as in machine learning. A less metaphysical and more plausible explanation lies closer to established physics: finite information propagation and local interactions suffice to generate the same sparse, hierarchical patterns without assuming that the universe is literally a computation.

Organization of the paper. Section B.4 states and proves the *ML Approximation Theorem*, showing that approximation and sample–complexity rates depend on the *largest local arity* rather than the ambient dimension. Section B.5 presents the *finite-horizon compilation principle* (a conditional, algorithmic statement), together with two illustrative examples—discrete-time lattice-local updates with finite propagation and k-local quantum dynamics compiled to two-qubit circuits—and clarifies caveats (long-range/dense couplings, chaotic sensitivity, precision and horizon dependence). Section B.6 discusses implications for learnability and optimization geometry, including explicit Rademacher bounds and Hessian-structure consequences aligned with compositional DAGs. Section B.7 analyzes limitations and non-theorems: effects of chaos (growth with horizon), undecidability in thermodynamic-limit questions, the role of uniformity versus non-uniform advice, and complexity barriers (e.g., QMA-hardness). The conclusion situates these results within broader questions about when physical evolutions are *efficiently computable* or, more modestly, *inherit compositional structure* precisely when uniform finite-horizon simulability is available.

Two Structural Principles

(A) ML Approximation Principle. Let f factor through a depth-L bounded–fan-in DAG with node maps $g_v \in C^r([0,1]^{d_v})$, $d_v \le d^*$, total size s. Then for any $\varepsilon \in (0, 1/4)$ there exists a ReLU network f_ε with

$$\|f - f_\varepsilon\|_\infty \le \varepsilon, \qquad \mathrm{depth}(f_\varepsilon) \le c\, L \log(1/\varepsilon), \quad \mathrm{size}(f_\varepsilon) \le C \sum_v \varepsilon_v^{-d_v/r} \log \tfrac{1}{\varepsilon_v},$$

under the telescoping constraint $\sum_{\ell=1}^{L} \left(\prod_{j>\ell} L_j \right) \max_{v \in \ell} \varepsilon_v \le \varepsilon$. If $\max_\ell L_\ell \le \bar{L}$ and $\varepsilon_v \asymp \varepsilon$,

$$\mathrm{size}(f_\varepsilon) \;\le\; C'\, s\, \varepsilon^{-d^*/r} \log \tfrac{1}{\varepsilon}.$$

(B) Finite-Horizon Causal Compilation. Consider a system on a bounded-degree graph (Δ) with local radius-r updates $\phi_{t,i}$ that are L-Lipschitz and C^r. If propagation is finite, i.e.

$$x_{B_{R(T)}(i)} = x'_{B_{R(T)}(i)} \;\Rightarrow\; (U_T \circ \cdots \circ U_1(x))_i = (U_T \circ \cdots \circ U_1(x'))_i,$$

then for any finite region Λ and horizon T, the input–output map $\mathcal{F}_{\Lambda,T}(x) := (U_T \circ \cdots \circ U_1(x))_\Lambda$ factors through a depth-T DAG with fan-in $\le \kappa(\Delta, r)$ and size $s = O\big(|\Lambda|\, (\kappa(\Delta, r))^T\big)$. Consequently, by (A), $\mathcal{F}_{\Lambda,T}$ admits an ε-accurate network with depth $O\big(T \log(1/\varepsilon)\big)$ and size $O\big(s\, \varepsilon^{-d^*/r} \log(1/\varepsilon)\big)$ where $d^* = \kappa(\Delta, r)$.

B.2 Definitions and Setup

Ambient spaces and norms. We work on $[0,1]^d$ with the uniform (sup) norm $\|x\|_\infty := \max_i |x_i|$ and the induced function norm $\|g\|_\infty := \sup_{x \in [0,1]^d} |g(x)|$. For $r > 0$ write $r = k + \alpha$ with $k := \lfloor r \rfloor \in \mathbb{N}_0$ and $\alpha \in (0, 1]$. The Hölder class $C^r([0,1]^p)$ consists of functions with continuous

partial derivatives up to order k and Hölder continuous kth derivatives with exponent α, equipped with the norm

$$\|g\|_{C^r} := \sum_{|\beta|\le k} \|D^\beta g\|_\infty + \sum_{|\beta|=k} \sup_{\substack{x\neq y \\ x,y\in[0,1]^p}} \frac{|D^\beta g(x) - D^\beta g(y)|}{\|x-y\|_\infty^\alpha}.$$

Efficient computability. A Boolean map $f : \{0,1\}^n \to \{0,1\}^m$ is *efficiently computable* if a deterministic Turing machine computes $f(x)$ in $T(n) = \mathrm{poly}(n)$ steps. A real map $f : [0,1]^d \to \mathbb{R}^m$ is efficiently computable to precision $\varepsilon \in (0,1)$ if, in the finite-precision model, there exists a *uniform* algorithm that outputs y with $\|y - f(x)\|_\infty \le \varepsilon$ in time $\mathrm{poly}(d, \log(1/\varepsilon))$. Here "uniform" means the corresponding circuit/arithmetic-circuit family is P-uniform (generable in time $\mathrm{poly}(d, \log(1/\varepsilon))$).

Compositional sparsity. A function f is *compositionally sparse* if it factors through a bounded–fan-in layered DAG. Concretely, let $G = (V, E)$ be a layered DAG with layers $1, \dots, L$, and let d_v be the in-degree (arity) of node v. Write the layer-ℓ map as $\Phi_\ell = (g_v)_{v\in\text{layer }\ell}$, where each node map $g_v : [0,1]^{d_v} \to \mathbb{R}$ has $d_v \le d^*$ and $g_v \in C^r([0,1]^{d_v})$. Then

$$f = \Phi_L \circ \Phi_{L-1} \circ \cdots \circ \Phi_1, \qquad \text{depth} = L, \quad \text{size } s := |V|, \quad d^* := \max_v d_v.$$

We assume each Φ_ℓ is L_ℓ-Lipschitz on the (compact) subset of its domain reached by upstream layers, measured in $\|\cdot\|_\infty$. For later use define the propagation multipliers

$$C_\ell := \prod_{j=\ell+1}^{L} L_j \qquad (\text{with } C_L := 1),$$

which quantify telescoping of local approximation errors through depth.

B.3 From Efficient Computation to DAGs

The passage from Turing machines to circuits is mathematically central. Any computation performed by a Turing machine (TM) on n-bit inputs and running for $T(n)$ steps can be *time-unrolled* into a circuit: each layer encodes one transition of the machine, and wires carry the bits of the instantaneous configuration to the next layer. Because the transition function inspects and writes only a bounded number of symbols per step, every gate has *bounded fan-in* (and bounded fan-out). The resulting circuit has size $O(T(n))$ and depth $O(T(n))$, hence can be viewed as a bounded–fan-in directed acyclic graph (DAG) implementing the same computation. This classical correspondence underlies the relationship between time-bounded Turing computation and circuit complexity [**Arora_Barak_2009**, **sipser1996intro**]: every $T(n)$-time TM yields a polynomial-size circuit family; conversely, with *P-uniformity* the family can be generated in time $\mathrm{poly}(n)$ and characterizes P, while without uniformity one obtains $\mathrm{P} \subseteq \mathrm{P/poly}$.

From bits to reals: finite-precision computation

Digital computation manipulates finite strings. A real $x \in [0,1]$ can be represented by the first p bits of its binary expansion $(x_1, \dots, x_p)$, yielding the dyadic $\tilde{x} = \sum_{i=1}^p x_i 2^{-i}$. Arithmetic on these encodings—addition, multiplication, comparison—has Boolean-circuit implementations of size $\mathrm{poly}(p)$ (schoolbook/FFT-based variants differ only in polylog factors). Thus computing $f : [0,1]^d \to \mathbb{R}^m$ to accuracy $\varepsilon = 2^{-p}$ amounts to computing a Boolean function on $(x_1, \dots, x_p)$

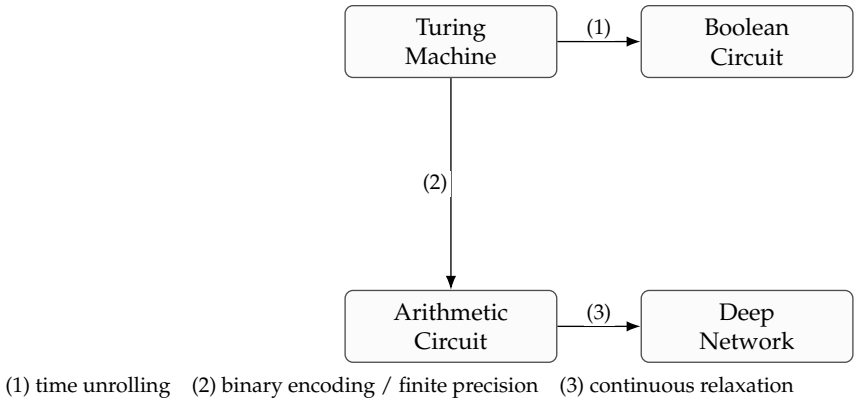


Figure B.1: Conceptual chain from discrete to continuous computation.

whose output decodes to $\tilde{f}(x)$ with $\|f(x) - \tilde{f}(x)\|_\infty \leq \varepsilon$. In particular, efficient real computation in the standard finite-precision model is equivalent, up to polynomial overhead, to efficient Boolean computation on binary encodings [**Papadimitriou1994ComputationalComplexity**].

B.3.1 From arithmetic circuits to neural DAGs

Replacing Boolean gates by continuous modules (affine maps and simple nonlinearities) yields *arithmetic circuits*—feedforward networks over $\mathbb{R}$. Crucially, the *topology* of the underlying DAG is unchanged: each node aggregates a bounded number of inputs and produces outputs to a bounded number of downstream nodes. The compositional skeleton revealed by time-unrolling a TM is therefore already a bounded–fan-in DAG.

Modern deep networks form smooth parametric families of such arithmetic circuits. They do not emulate a given circuit symbol-for-symbol; rather, they approximate functions supported on the same class of sparse, layered DAGs. Structurally: any efficiently Turing-computable function gives rise (via unrolling and finite-precision encoding) to a bounded–fan-in compositional scaffold; deep networks can approximate the node maps of this scaffold to any target accuracy, with approximation rates governed by local arities d_v and the smoothness of the constituent modules (cf. Section B.4). In this sense, deep architectures are natural continuous relaxations of efficient discrete circuits—the same kind of machines realized by physical hardware.

B.3.2 Computation as compositional structure

This yields a conceptual loop. Programs compile to (uniform) circuit families; circuits are bounded–fan-in DAGs; and bounded local arity is the operative notion of sparsity. Therefore, *any efficiently computable transformation, when executed at finite precision by a device with a fixed bounded-arity primitive set, admits a bounded–fan-in compositional representation*. The point is structural rather than stylistic: bounded local interactions are an inherent consequence of efficiency on finite, causal, local hardware.

B.4 ML Approximation Theorem

We give (i) a local constructive approximation with explicit constants, (ii) a global telescoping bound, and (iii) the optimal error budget across layers/nodes.

B.4.1 Local constructive approximation with explicit constants

For $p \in \mathbb{N}$ and $r > 0$, let $C_{p,r} > 0$ denote the best constant in the multivariate Taylor remainder on $[0,1]^p$ (measured in the $\|\cdot\|_\infty$ norm):

$$\sup_{x \in Q} \left| g(x) - T_Q(x) \right| \;\le\; C_{p,r}\, \|g\|_{C^r}\, \mathrm{diam}_\infty(Q)^{\,r}, \tag{B.1}$$

where T_Q is the degree-$\lfloor r \rfloor$ Taylor polynomial of g at the center of a cube $Q \subset [0,1]^p$.

We use two standard ReLU *gadgets* (e.g., **[Yarotsky2018]**):

(G1) Squaring. For any $\eta \in (0,1)$ there is a ReLU network $\mathsf{Sq}_\eta : [-2,2] \to \mathbb{R}$ with

$$\sup_{t \in [-2,2]} \left| \mathsf{Sq}_\eta(t) - t^2 \right| \le \eta, \qquad \mathrm{depth}(\mathsf{Sq}_\eta) \le C_\square \log(1/\eta), \;\; \mathrm{size}(\mathsf{Sq}_\eta) \le C'_\square \log(1/\eta).$$

(G2) Multiplication. By polarization, $xy = \frac{1}{4}\left[(x+y)^2 - (x-y)^2\right]$. Composing two Sq_η gadgets and affine maps yields a multiplier $\mathsf{Mult}_\eta : [-1,1]^2 \to \mathbb{R}$ with

$$\sup_{|x|,|y| \le 1} \left| \mathsf{Mult}_\eta(x,y) - xy \right| \le \eta, \quad \mathrm{depth}(\mathsf{Mult}_\eta) \le C_\times \log(1/\eta), \;\; \mathrm{size}(\mathsf{Mult}_\eta) \le C'_\times \log(1/\eta).$$

By a binary tree, the product of p scalars in $[-1,1]$ is realizable with accuracy η at depth $O(\log p \cdot \log(1/\eta))$ and size $O(p \log(1/\eta))$.

Lemma 11 (Local ReLU approximation)**.** *Let $g \in C^r([0,1]^p)$ with $\|g\|_{C^r} \le M$ and $r > 0$. For any $\delta \in (0, 1/4)$ there exists a ReLU network N such that*

$$\|g - N\|_\infty \le \delta, \qquad \mathrm{depth}(N) \le c_1(r)\, \log(1/\delta), \qquad \mathrm{size}(N) \le c_2(p,r)\, M^{p/r}\, \delta^{-p/r} \log(1/\delta). \tag{B.2}$$

Proof. Partition $[0,1]^p$ into N^p congruent cubes $\{Q\}$ of side $h = 1/N$, where

$$N \;=\; \left\lceil \left(2C_{p,r} M/\delta\right)^{1/r} \right\rceil. \tag{B.3}$$

For each Q, let T_Q be the degree-$\lfloor r \rfloor$ Taylor polynomial at the cube center. Then, by (B.1) and (B.3),

$$\sup_{x \in Q} |g(x) - T_Q(x)| \;\le\; C_{p,r} M\, h^r \;\le\; \delta/2. \tag{B.4}$$

Construct a C^0 partition of unity $\{\Psi_Q\}_Q$ from univariate ReLU "tent" splines $\{\psi_j\}_{j=0}^N$ on $[0,1]$ (knots at j/N) by setting $\Psi_Q(x) := \prod_{i=1}^p \psi_{j_i}(x_i)$ for the multi-index of Q. Each ψ_j is exactly a ReLU spline with $O(N)$ breakpoints; the p-fold product is realized by a binary tree of Mult_η gadgets with internal accuracy chosen below. Define

$$N(x) \;:=\; \sum_Q \Psi_Q(x)\, T_Q(x). \tag{B.5}$$

For $x \in Q$, using $\sum_{Q'} \Psi_{Q'} \equiv 1$,

$$|g(x) - N(x)| \;\le\; |g(x) - T_Q(x)| + \sum_{Q'} \Psi_{Q'}(x)\, |T_{Q'}(x) - T_Q(x)|.$$

Only Q' adjacent to Q contribute; Taylor coefficient continuity implies $|T_{Q'}(x) - T_Q(x)| \le C\, M\, h$ on Q. Choosing the internal gadget accuracy $\eta = \eta(p)\, \delta/(4p)$ to control multiplier and monomial evaluations and taking h from (B.3), the second term is $\le \delta/2$. Together with (B.4), this yields $\|g - N\|_\infty \le \delta$.

Complexity. Each univariate ψ_j uses $O(N)$ ReLU units; across p coordinates this is $O(pN)$ per "stencil." The p-fold product per cell uses $O(p \log(1/\eta))$ units and depth $O(\log p \cdot \log(1/\eta))$. Evaluating T_Q requires $O(1)$ monomials (depending on $\lfloor r \rfloor$), each realized with multiplier trees at accuracy η. Summing over N^p cells and using (B.3) with $\eta \simeq \delta$ gives (B.2). □

B.4.2 Global error propagation and optimal budget

Let Φ_ℓ be the layer-ℓ block map in the DAG, with Lipschitz constant L_ℓ on the reachable compact set, and let $\widetilde{\Phi}_\ell$ be its blockwise approximant with $\|\Phi_\ell - \widetilde{\Phi}_\ell\|_\infty \le E_\ell$. Define $F_m := \Phi_m \circ \cdots \circ \Phi_1,\ \widetilde{F}_m := \widetilde{\Phi}_m \circ \cdots \circ \widetilde{\Phi}_1$.

Proposition 9 (Telescoping bound)**.** *For any depth L,*

$$\|F_L - \widetilde{F}_L\|_\infty \ \le \ \sum_{\ell=1}^{L} \Big(\prod_{j=\ell+1}^{L} L_j \Big) E_\ell. \tag{B.6}$$

Proof. Insert and subtract intermediate terms and use Lipschitz continuity:

$$F_L - \widetilde{F}_L = \sum_{\ell=1}^{L} \Phi_L \circ \cdots \circ \Phi_{\ell+1} \circ (\Phi_\ell - \widetilde{\Phi}_\ell) \circ \widetilde{F}_{\ell-1}.$$

Taking sup norms yields (B.6). □

Let ε_v be the tolerance chosen at node v (for Lemma 11) and set $E_\ell := \max_{v \in \text{layer } \ell} \varepsilon_v$. The total size is

$$S(\{\varepsilon_v\}) = \sum_v a_v \, \varepsilon_v^{-d_v/r} \log \frac{1}{\varepsilon_v}, \qquad a_v := c_2(d_v, r)\, M_v^{d_v/r}, \tag{B.7}$$

subject to the telescoping constraint

$$\sum_{\ell=1}^{L} C_\ell E_\ell \le \varepsilon, \qquad C_\ell := \prod_{j=\ell+1}^{L} L_j, \quad E_\ell = \max_{v \in \ell} \varepsilon_v, \quad 0 < \varepsilon_v \le \tfrac{1}{4}. \tag{B.8}$$

Because the $\max$ couples nodes within a layer, the optimizer equalizes ε_v within each layer up to constants. Let $A_\ell := \sum_{v \in \ell} a_v$ and $k_\ell := \max_{v \in \ell} d_v \le d^*$. Discarding the slowly varying $\log(1/\varepsilon_\ell)$ (affecting only polylog factors), the reduced problem is

$$\min_{E_1, \dots, E_L > 0} \ \sum_{\ell=1}^{L} A_\ell E_\ell^{-k_\ell/r} \quad \text{s.t.} \quad \sum_{\ell=1}^{L} C_\ell E_\ell \le \varepsilon. \tag{B.9}$$

The Lagrangian $\mathcal{L} = \sum_\ell A_\ell E_\ell^{-k_\ell/r} + \lambda(\sum_\ell C_\ell E_\ell - \varepsilon)$ yields, for each ℓ,

$$-\frac{k_\ell}{r} A_\ell \, E_\ell^{-(k_\ell/r+1)} + \lambda C_\ell = 0 \quad \Rightarrow \quad E_\ell \ = \ \Big(\tfrac{k_\ell A_\ell}{\lambda r \, C_\ell} \Big)^{\frac{r}{k_\ell + r}}. \tag{B.10}$$

Enforcing $\sum_\ell C_\ell E_\ell = \varepsilon$ determines $\lambda > 0$ uniquely and gives

$$E_\ell \ = \ \varepsilon \, \frac{C_\ell^{-\frac{r}{k_\ell + r}} \, (k_\ell A_\ell / r)^{\frac{r}{k_\ell + r}}}{\sum_{t=1}^{L} C_t^{\frac{k_t}{k_t + r}} \, (k_t A_t / r)^{\frac{r}{k_t + r}}}. \tag{B.11}$$

In the common case $k_\ell \equiv d^*$ and $A_\ell \asymp A$ across layers, (B.11) reduces to

$$E_\ell \ \propto \ C_\ell^{-\frac{r}{d^* + r}}, \qquad \text{and if } \max_\ell L_\ell \le \bar{L} < \infty, \text{ then } C_\ell = \Theta(1) \Rightarrow E_\ell \asymp \varepsilon / L. \tag{B.12}$$

B.4.3 The theorem

Theorem 16 (ML Approximation Theorem)**.** *Let $f : [0,1]^d \to \mathbb{R}$ factor through a depth-L DAG of size s with node maps $g_v \in C^r([0,1]^{d_v})$, $d_v \le d^*$, and assume $\mathrm{Lip}(\Phi_\ell) \le L_\ell$ on reachable domains. For any $\varepsilon \in (0, 1/4)$ there exists a ReLU network f_{NN} such that*

$$\|f - f_{\mathrm{NN}}\|_\infty \le \varepsilon, \qquad \mathrm{depth}(f_{\mathrm{NN}}) \le c\, L \log(1/\varepsilon), \tag{B.13}$$

and

$$\mathrm{size}(f_{\mathrm{NN}}) \;\le\; C \sum_v \varepsilon_v^{-d_v/r} \log \frac{1}{\varepsilon_v}, \quad \textit{with} \;\; \sum_{\ell=1}^{L} C_\ell E_\ell \le \varepsilon,\; E_\ell = \max_{v \in \ell} \varepsilon_v. \tag{B.14}$$

If $\max_\ell L_\ell \le \bar{L} < \infty$, choosing $\varepsilon_v \asymp \varepsilon$ for all v yields

$$\mathrm{size}(f_{\mathrm{NN}}) \;\le\; C'\, s\, \varepsilon^{-d^*/r} \log(1/\varepsilon). \tag{B.15}$$

Proof. (i) Local approximants. For each node v in layer ℓ, apply Lemma 11 to g_v with tolerance ε_v, producing N_v with $\|g_v - N_v\|_\infty \le \varepsilon_v$ and the size/depth bounds in (B.2).

(ii) Wiring. Compose the N_v along the DAG topology. Along any path, depths add; with $\varepsilon_v \ge c_0\, \varepsilon$ (as ensured by (B.12) up to constants), each layer contributes $O(\log(1/\varepsilon))$ to depth, giving (B.13). The total size is the sum over nodes, yielding (B.14).

(iii) Global error. Writing Φ_ℓ and $\widetilde{\Phi}_\ell$ for the true and approximating layer maps (stacking g_v and N_v), we have $\|\Phi_\ell - \widetilde{\Phi}_\ell\|_\infty \le E_\ell := \max_{v \in \ell} \varepsilon_v$. Apply Proposition 9 and impose (B.8) to get $\|f - f_{\mathrm{NN}}\|_\infty \le \varepsilon$.

(iv) Uniform Lipschitz. If L_ℓ are uniformly bounded, (B.12) implies $E_\ell \asymp \varepsilon/L$ and we may take $\varepsilon_v \asymp \varepsilon$ for all v, which collapses (B.14) to (B.15) with exponent d^*/r. □

B.4.4 Depth separation

Theorem 17 (Depth separation via linear regions)**.** *For each $L \in \mathbb{N}$ there exists a function $f_L : [0,1] \to \mathbb{R}$ expressible as a composition of L binary-arity C^∞ modules such that:*

1. *For any $\varepsilon \in (0, 1/4)$ there is a ReLU network with depth $O\big(L \log(1/\varepsilon)\big)$ and size $\mathrm{poly}(1/\varepsilon)$ that achieves $\|f_L - f_{\mathrm{NN}}\|_\infty \le \varepsilon$.*

2. *Any ReLU network with a single hidden layer that achieves $\|f_L - f_{\mathrm{NN}}\|_\infty \le \varepsilon_0$ for some fixed $\varepsilon_0 \in (0, 1/4)$ must have width at least $\exp(\Omega(L))$.*

Proof sketch. Let $s : [0,1] \to [0,1]$ be a piecewise-linear "sawtooth" with two linear pieces; define $f_L := s^{\circ L}$. Then f_L has at least 2^L linear pieces. (1) follows by applying Theorem 16 with $d^* = 2$. For (2), a 1D single-hidden-layer ReLU with width W has at most $W + 1$ linear regions on $[0,1]$, forcing $W \ge \Omega(2^L)$ to achieve fixed-accuracy approximation; see, e.g., **[Telgarsky2016, EldanShamir2016]**. □

Remark 4 (Why the exponent is d^*/r)**.** *The construction in Lemma 11 drives the $\delta^{-p/r}$ cost at* each *node of local arity $p = d_v$. Summing across nodes with tolerances $\varepsilon_v \asymp \varepsilon$ shows that the governing exponent is the worst local arity $d^* = \max_v d_v$, not the ambient d. This is the precise sense in which compositional sparsity mitigates the curse of dimensionality.*

Theorem 18 (Unified ML Approximation for Compositional DAGs)**.** *Let $f = \Phi_L \circ \cdots \circ \Phi_1$ where each layer Φ_ℓ stacks node maps $g_v \in C^r([0,1]^{d_v})$ with $d_v \le d^*$, and assume $\mathrm{Lip}(\Phi_\ell) \le L_\ell$. Writing $s = \sum_\ell |\{v \in \ell\}|$, for any $\varepsilon \in (0, 1/4)$ there exists a ReLU net f_ε with*

$$\|f - f_\varepsilon\|_\infty \le \varepsilon, \quad \mathrm{depth}(f_\varepsilon) \le c\, L \log \tfrac{1}{\varepsilon}, \quad \mathrm{size}(f_\varepsilon) \le C \sum_v \varepsilon_v^{-d_v/r} \log \tfrac{1}{\varepsilon_v},$$

subject to $\sum_{\ell=1}^{L} \left(\prod_{j>\ell} L_j\right) \max_{v\in\ell} \varepsilon_v \le \varepsilon$. If $\max_\ell L_\ell \le \bar{L}$ and $\varepsilon_v \asymp \varepsilon$,

$$\mathrm{size}(f_\varepsilon) \;\le\; C'\, s\, \varepsilon^{-d^*/r} \log \tfrac{1}{\varepsilon}.$$

Moreover, the class $\mathcal{F}_{comp}(s, L, d^*, r)$ satisfies the entropy bound

$$\log \mathcal{N}(\varepsilon, \mathcal{F}_{comp}, \|\cdot\|_\infty) \;\le\; A\, s\, \varepsilon^{-d^*/r} \,\mathrm{polylog}\Big(\tfrac{1}{\varepsilon}\Big),$$

and hence the Rademacher bound

$$\mathfrak{R}_n(\mathcal{F}_{comp}) \;\le\; \tilde{C} \sqrt{\tfrac{s}{n}}\, n^{-\frac{r}{2d^*}}\, \mathrm{polylog}(n).$$

B.5 Finite-Horizon Compilation: From Uniform Simulators to Algorithmic Compositionality

We state an *algorithmic, conditional* compilation principle: whenever a finite-horizon prediction map is computed by a *uniform* simulator using a bounded-arity primitive set, it admits a *P-uniform* bounded–fan-in DAG and therefore falls under our ML approximation results. We then give two *examples*: a discrete-time lattice-local model (recovering the spacetime-DAG construction) and standard quantum k-local dynamics. We do *not* claim that all physical evolutions satisfy these hypotheses; the examples are modeling assumptions, not universal laws.

B.5.1 Setup

Fix an instance "size" n (e.g., number of effective degrees of freedom, grid points, or qubits), a finite horizon $T > 0$, and target accuracy $\varepsilon \in (0, 1)$. Let

$$F_{T,n} : \mathcal{X}_n \longrightarrow \mathcal{Y}_n$$

denote the corresponding finite-horizon input–output map, evaluated in a standard finite-precision model in which word size scales as $O(\log(1/\varepsilon))$ and running time/size are measured in bit operations and primitive invocations.

B.5.2 Compilation theorem (algorithmic, safe)

Theorem 19 (Uniform simulator ⇒ bounded–fan-in DAG)**.** *Suppose there exists a deterministic Turing machine* Sim *which, on input $(1^n, T, \varepsilon, x)$, outputs an ε-accurate value of $F_{T,n}(x)$ using at most $s(n, T, \varepsilon)$ primitive calls, each drawn from a fixed finite set $\mathcal{P}$ of arity at most a (a constant), with per-call cost measured at precision $O(\log(1/\varepsilon))$. Assume* P-uniformity*: given $(1^n, T, \varepsilon)$ one can generate, in time $\mathrm{poly}(n, \log(1/\varepsilon), T)$, the instruction schedule and parameters for the $s(n, T, \varepsilon)$ calls. Then there exists a* P-uniform *circuit/DAG $\mathcal{G}_{n,T,\varepsilon}$ of size $O(s(n, T, \varepsilon))$ and maximum fan-in $\le a$ that computes $F_{T,n}$ to accuracy ε in the same finite-precision model. Consequently, $F_{T,n}$ is* algorithmically compositionally sparse.

Proof sketch. Replace each primitive of arity $\leq a$ in $\mathcal{P}$ by a constant-size gate gadget at the target precision; wire these gadgets following the simulator's instruction schedule (unrolling loops/branches into a straight-line program of length s). Fan-in is $\leq a$ by construction and the total gate count is $O(s)$. P-uniformity follows because the same schedule generator lists the gates and parameters in time polynomial in $(n, \log(1/\varepsilon), T)$, which yields a P-uniform family. □

Remark 5 (Scope and non-claims)**.** *Theorem 19 is* purely algorithmic*. It does not assume bounded-degree physical interactions or a physical light cone. If a bounded–fan-in DAG as above does* not *exist for $F_{T,n}$ under these resource bounds, that implies a lower bound against any such uniform simulator over the given primitive set and precision regime.*

Example A: Discrete-time lattice-local update (model assumption)

Setting. Let $G = (V, E)$ be a (possibly infinite) graph with maximum degree $\Delta < \infty$. For $i \in V$ and radius $r \in \mathbb{N}$, write

$$B_r(i) := \{j \in V : \operatorname{dist}_G(i, j) \leq r\}, \qquad |B_r(i)| \leq \kappa(\Delta, r) < \infty.$$

States are $x = (x_i)_{i \in V} \in X^V$ with the sup product norm on X.

Local updates and regularity. For $t = 1, \ldots, T$, consider sitewise updates

$$\big(U_t(x)\big)_i \;=\; \phi_{t,i}(x_{B_r(i)}), \qquad \phi_{t,i} : X^{B_r(i)} \to X, \tag{B.16}$$

with uniform Lipschitz and smoothness on relevant compacta,

$$\|\phi_{t,i}(z) - \phi_{t,i}(z')\| \;\leq\; L\,\|z - z'\|_\infty, \qquad \phi_{t,i} \in C^q, \tag{B.17}$$

for some $L \geq 1$, $q \geq 1$ independent of (t, i).

Finite propagation. Let $F_T := U_T \circ \cdots \circ U_1$. Assume there exists a nondecreasing $R : \mathbb{N} \to \mathbb{N}$ such that for all $i \in V$ and $x, x' \in X^V$,

$$x_{B_{R(T)}(i)} = x'_{B_{R(T)}(i)} \implies \big(F_T(x)\big)_i = \big(F_T(x')\big)_i. \tag{B.18}$$

(Nearest-neighbor dynamics give $R(T) \leq rT$.)

Proposition 10 (Model-locality ⇒ compositional DAG)**.** *Fix finite $\Lambda \subset V$ and horizon $T \in \mathbb{N}$. Under* (B.16)–(B.18)*, the map*

$$\mathcal{F}_{\Lambda,T}(x) := \big(F_T(x)\big)_\Lambda$$

factors through a depth-T DAG with node fan-in at most $d^* := \kappa(\Delta, r)$. The layer maps are C^q and L-Lipschitz on the reachable compact set. Hence $\mathcal{F}_{\Lambda,T}$ is compositionally sparse with depth T and size

$$s \;=\; O\Big(|\Lambda|\,\big(d^*\big)^T\Big),$$

and falls under the ML Approximation Theorem with exponent d^*/q (up to polylog factors).

B.6 Learning Theory and Optimization

Compositional sparsity has three consequences for learning and optimization.

(i) Capacity and generalization. Let $\mathcal{F}_{\text{comp}}(s, L, d^*, r)$ be depth-L DAGs of size s whose node maps lie in $C^r([0,1]^{d_v})$ with $d_v \le d^*$, and suppose layer maps are L_ℓ-Lipschitz on reachable domains. Then the metric entropy satisfies

$$\log \mathcal{N}(\varepsilon, \mathcal{F}_{\text{comp}}, \|\cdot\|_\infty) \;\le\; A\, s\, \varepsilon^{-d^*/r}\, \text{polylog}\Big(\tfrac{1}{\varepsilon}\Big), \tag{B.19}$$

and Dudley's integral yields

$$\mathfrak{R}_n(\mathcal{F}_{\text{comp}}) \;\le\; \tilde{C}\sqrt{\frac{s}{n}}\, n^{-\frac{r}{2d^*}}\, \text{polylog}(n). \tag{B.20}$$

Thus the *exponent* depends on d^* (largest local arity), not the ambient d.

(ii) Optimization geometry. If f_θ factors through a bounded–fan-in DAG, the Gauss–Newton/Hessian blocks obey

$$H_{vw} \neq 0 \;\Rightarrow\; \text{cone}(v) \cap \text{cone}(w) \neq \varnothing,$$

so the Hessian graph has bounded degree (by fan-in). This motivates block-diagonal/Kronecker preconditioners that respect the compositional blocks.

(iii) Architectural realizations. Sparse/top-q attention enforces *bounded effective fan-in*: for a query q, $\text{Attn}(q, K, V) = \sum_{j \in S(q)} \alpha_j(q)\, v_j$ with $|S(q)| = O(q)$ under top-q selection or sufficiently aggressive sparsification, placing such transformers within the same d^*-controlled regime.

B.7 Limits of the Framework

The compositional-sparsity results identify when approximation and learnability are governed by local arity d^* and depth L (or horizon T). We now delineate intrinsic *limits* of this regime. These are not violations of the DAG factorization itself but circumstances under which the constants (or the effective arity) grow so rapidly that efficient approximation on classical hardware is ruled out.

Classical chaos and finite prediction horizons

Consider the discrete-time system

$$x_{t+1} = F(x_t), \qquad F : X \to X, \tag{B.21}$$

with $X \subset \mathbb{R}^d$ compact and $F \in C^r$, locally Lipschitz. Suppose F admits a bounded-degree interaction graph so that updates factor through a bounded–fan-in DAG (cf. Section B.5). Let $F_T := F \circ \cdots \circ F$ denote the T-step map and set $L_t := \text{Lip}(F)$ on the reachable compact set at time t. Then

$$\text{Lip}(F_T) \;\le\; \prod_{t=1}^{T} L_t \;=\; \exp\Big(\sum_{t=1}^{T} \log L_t\Big). \tag{B.22}$$

If the maximal Lyapunov exponent $\lambda := \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \log L_t$ is positive, we obtain $\text{Lip}(F_T) \asymp e^{\lambda T}$. Applying the telescoping bound (B.6) to $\Phi_T := F_T$ and a blockwise approximant with layer errors $\{E_t\}$ gives

$$\|\Phi_T - \widetilde{\Phi}_T\|_\infty \;\le\; \sum_{t=1}^{T} e^{\lambda(T-t)}\, E_t. \tag{B.23}$$

To enforce $\|\Phi_T - \widetilde{\Phi}_T\|_\infty \le \varepsilon$, one must choose $E_t \lesssim \varepsilon\, e^{-\lambda(T-t)}$. Since each layer-$t$ local approximant with arity $\le d^*$ and smoothness r costs on the order of $E_t^{-\,d^*/r}$ (up to polylogs; Lemma 11), the total size obeys the lower bound

$$\mathrm{size}(\mathcal{N}_T) \;\gtrsim\; e^{\,(\lambda d^*/r)\,T} \quad \text{(up to polylog factors).} \tag{B.24}$$

Thus, for chaotic systems (e.g., atmospheric flows) long-horizon *pointwise* prediction requires exponentially increasing resources or exponentially precise initial data. The bounded–fan-in DAG persists, but numerical stability collapses with T.

Quantum locality versus entanglement growth

In quantum lattice systems, Lieb–Robinson bounds provide an approximate light cone: for observables A_X, B_Y supported on disjoint $X, Y \subset V$,

$$\|[A_X(t), B_Y]\| \;\le\; C\, \exp\!\big(-\,[d(X,Y) - v_{\mathrm{LR}} t]_+\big), \tag{B.25}$$

with velocity v_{LR}. Consequently, Heisenberg evolution of local observables factors through a depth-t bounded–fan-in DAG (operator sense). The obstruction is *representation*: under classical encodings, bipartite entanglement $S(t)$ typically grows (linearly in t in generic quenches), so any faithful classical representation requires $O(e^{S(t)})$ coefficients. The effective arity therefore scales as

$$d^*(t) \;\sim\; e^{S(t)}, \tag{B.26}$$

which, when substituted into the nodewise cost exponent $\varepsilon^{-\,d^*/r}$, negates the dimensional advantage and renders classical approximation intractable once $S(t)$ exceeds $O(\log n)$.

Precision and nonuniformity

Our statements are inherently finite-precision and uniform:

- *Precision growth.* To maintain error ε at horizon T under (B.23), input and intermediate precisions must scale as $\Omega(\lambda T + \log(1/\varepsilon))$ bits. Any model that fixes word size incurs an irreducible floor at large T.

- *Nonuniform advice.* Allowing nonuniform circuit families (P/poly) can collapse uniform lower bounds, but the resulting dependence on instance-specific advice falls outside the learnability guarantees considered here.

Thermodynamic limits and undecidability

The compilation and approximation results concern fixed finite n and finite horizon T. In the thermodynamic limit ($n \to \infty$) and for decision problems about phases or long-time behavior, classical undecidability and complexity barriers (e.g., QMA-hardness for certain ground-state or dynamics questions) imply that no uniformly efficient classical procedure can exist in general. Such barriers lie orthogonal to local DAG factorization and instead limit *which* finite-horizon tasks admit uniform simulators.

Regime of validity (summary)

The framework applies when:

1. there is a bounded-degree local interaction structure (fixed radius and maximum degree);
2. finite-horizon maps are uniformly simulable at finite precision;
3. layer Lipschitz constants are bounded on reachable domains.

Failure modes include positive Lyapunov exponents (forcing (B.24)), entanglement-induced arity growth (B.26), precision that does not scale with T, and nonuniform advice. In these cases the bounded–fan-in DAG remains as an operator-level structure, but efficient *classical* approximation is precluded.

B.8 Conclusion

Compositionality, in the precise mathematical sense of bounded–fan-in DAG factorizations, is a common structural denominator for learning and finite-horizon physical prediction. The core message of this work is *structural*: under appropriate hypotheses, both domains give rise to the same low-arity, layered scaffold.

Equivalences (scope-conditional)

On the learning side, the *ML Approximation Theorem* (Theorem 16, Section B.4) shows that if a target factors through a bounded–fan-in DAG with maximal local arity d^*, then the approximation/sample–complexity exponents replace the ambient d/r by d^*/r. On the physics side, the *finite-horizon compilation principle* (Theorem 19, Section B.5) states that whenever a finite-horizon evolution map is computed by a *P-uniform* simulator using a bounded-arity primitive set at finite precision, the map compiles to a P-uniform bounded–fan-in DAG of size proportional to the simulator cost.

Thus, the two origins—efficient computability and finite-speed locality—*converge to the same structural form under explicit, stated conditions* (uniformity, finite precision, bounded-arity primitives, finite horizon). The equivalence is structural and conditional, not ontological: computation yields compositionality by uniform time-unrolling; locality yields it by causal reach at finite horizon.

Consequences (within the regime of validity)

Within this regime, three consequences follow:

1. **Approximation.** Efficiently computable targets and finite-horizon local maps admit deep approximants whose complexity scales with d^* rather than d (Theorem 16).
2. **Generalization.** Metric entropy and Rademacher bounds scale as $\log \mathcal{N}(\varepsilon) \lesssim s\,\varepsilon^{-d^*/r}\mathrm{polylog}(1/\varepsilon)$ and $\mathfrak{R}_n \lesssim \sqrt{s/n}\,n^{-r/(2d^*)}\mathrm{polylog}(n)$, exhibiting a "local dimension" law (Section B.6).
3. **Optimization geometry.** Bounded local arity induces block-sparse curvature aligned with the DAG cones, explaining the effectiveness of structured (block/Kronecker) preconditioners in the sparse regime (Section B.6).

Limits (where structure persists but efficiency may not)

The structural DAG persists beyond the regime, but *efficiency* can fail. Positive Lyapunov exponents force error amplification $\sim e^{\lambda T}$, yielding size lower bounds $\gtrsim e^{(\lambda d^*/r)T}$ for long-horizon pointwise prediction (Section B.7). In quantum dynamics, entanglement growth drives an effective arity $d^*(t) \sim e^{S(t)}$, erasing the d^*-advantage under classical encodings. Precision requirements grow with horizon, and nonuniform advice falls outside our learnability results.

Outlook

Compositionality is therefore best viewed as a *bridge principle*: under uniform finite-precision simulation and finite causal reach, learning theory and finite-horizon physics share the same compositional grammar—hierarchical, sparse, layered. The bridge is exact at the level of structure and rates *under the stated conditions*; outside them, the scaffold remains but efficient approximation may be precluded. Clarifying the sharp frontiers of this regime—especially the interplay among uniformity, precision growth, and effective arity—remains an open direction of foundational interest.

CHAPTER C

Technical Note: A Group-invariant Johnson-Lindestrauss Lemma

*We give a **complete reformulation and proof** of the "JL under group-invariance" result (Anselmi–Rosasco–Poggio,* On Invariance and Selectivity in Representation Learning*)[5], see also [158].*

C.1 Introduction

Concretely, for the paper's *tomographic probabilistic / CDF* representation built from group averages,

$$\mu_t(I)(b) \;=\; \int_G dg\; \eta_b(\langle I, gt\rangle), \qquad \eta_b(a) = \mathbf{1}\{a \le b\},$$

and the (population or finite-template) *quotient* metric

$$d(I, I') = \int_S du(t)\, d_\infty(\mu_t(I), \mu_t(I')) \quad \text{or} \quad \widehat{d}(I, I') = \frac{1}{k}\sum_{i=1}^{k} d_\infty(\mu_{t_i}(I), \mu_{t_i}(I')),$$

WE prove that a single subgaussian **random linear map** $R : \mathbb{R}^d \to \mathbb{R}^m$ preserves these distances (up to explicit multiplicative and additive tolerances) **simultaneously for all pairs** from a finite dataset S, with the usual Johnson–Lindenstrauss dimension

$$m \;=\; \mathcal{O}\Big(\varepsilon^{-2}\,\log N\Big), \qquad N = \text{size of the finite set of vectors that must be near-isometric.}$$

The only extra logarithmic cost relative to classical JL is the **complexity of the invariance**: the number of templates k and (for compact, non-finite groups) the covering number of G at the accuracy η used to discretize Haar integration.

b. Method Sketch.

- **Setup (paper's formulation).** Data live in a Hilbert space $I \simeq \mathbb{R}^d$. A compact group $G \subset O(d)$ acts unitarily; templates $t \in S = \{x : \|x\| = 1\}$. The paper's invariant/selective representation aggregates **nonlinear group averages of 1-D projections** $\langle I, gt\rangle$ and measures between-signal dissimilarity via a CDF (KS) metric averaged over t.

- **Goal.** Show that after a random linear map R, the *post-projection* representation computed in the **embedded space** (i.e., replace $\langle I, gt\rangle$ by $\langle RI, Rgt\rangle$) yields a distance $\widehat{d}_R$ that approximates $\widehat{d}$ (and d) uniformly on a finite dataset.
- **Key lemmas.**
  (1) (**Inner-product preservation from JL.**) If R is a $(1 \pm \varepsilon)$ near-isometry on the finite set $\mathcal{V} \supset \{I,\ gt,\ I \pm gt\}$, then for all such pairs
  $$|\langle RI, Rgt\rangle - \langle I, gt\rangle| \ \le\ C\varepsilon\big(\|I\|^2 + \|t\|^2\big)^{1/2}.$$
  (2) (**Stability of CDFs to additive perturbations.**) If random variables X and $X + \xi$ satisfy $|\xi| \le \Delta$ almost surely, then their CDFs differ by at most Δ in the uniform (KS) metric; more generally, with smoothing (e.g., Lipschitz η) the bound scales with $\|\xi\|$.
  (3) (**Group discretization.**) For compact G, an η-net $\mathcal{N}_\eta \subset G$ in operator norm yields
  $$|\min_{g\in G} \phi(g) - \min_{h\in\mathcal{N}_\eta} \phi(h)| \ \le\ \mathrm{Lip}(\phi)\,\eta,$$
  and in our case induces an **additive** $O(\eta)$ error coming from approximating Haar averaging by a finite sum.
- **Assemble.** Choose R that is near-isometric on a **finite control set** built from the dataset S, the templates T_k, and a group net $\mathcal{N}_\eta$. Lemma 1 controls projection errors; Lemma 2 converts them to CDF (KS) errors template-wise; averaging over templates gives a uniform bound on $\widehat{d}_R - \widehat{d}$; Lemma 3 controls the additional group-discretization error for d vs $\widehat{d}$. A union bound gives
  $$m \ \gtrsim\ \varepsilon^{-2}\Big(\log|S| + \log k + \log N_G(\eta) + \log(1/\delta)\Big).$$

C.2 Detailed Solution

We work in the paper's setting: $I = \mathbb{R}^d$ with Euclidean inner product, and a **compact** group $G \subset O(d)$ acting unitarily. Fix a finite dataset $S \subset I$. For templates, take either the population sphere average $t \sim u$ on S^{d-1} or a finite set $T_k = \{t_1, \dots, t_k\} \subset S^{d-1}$. For $\eta_b(a) = \mathbf{1}\{a \le b\}$ and Haar measure dg, the paper's CDF-representation is

$$\mu_t(I)(b) \ =\ \int_G dg\ \eta_b(\langle I, gt\rangle), \qquad d(I, I') \ =\ \int_{S^{d-1}} du(t)\ d_\infty(\mu_t(I), \mu_t(I')),$$

and its finite-template version

$$\widehat{d}(I, I') \ =\ \frac{1}{k}\sum_{i=1}^{k} d_\infty(\mu_{t_i}(I), \mu_{t_i}(I')), \qquad d_\infty(f, g) = \sup_{b\in\mathbb{R}} |f(b) - g(b)|.$$

We now define the **post-projection** (embedded) representation: for a linear map $R : \mathbb{R}^d \to \mathbb{R}^m$,

$$\mu_t^R(I)(b) \ =\ \int_G dg\ \eta_b(\langle RI, R(gt)\rangle), \qquad \widehat{d}_R(I, I') \ =\ \frac{1}{k}\sum_{i=1}^{k} d_\infty(\mu_{t_i}^R(I), \mu_{t_i}^R(I')).$$

(If we use population averaging over t, replace the finite average by $\int du(t)$ and similarly define d_R.)

Our objective is to **uniformly** control $|\widehat{d}_R - d|$ (or $|\widehat{d}_R - \widehat{d}|$) over all $I, I' \in S$ with high probability over a random R. Throughout, $C > 0$ denotes absolute constants that may change line-to-line.

C.2.1 A finite control set and inner-product preservation

Let $\mathcal{N}_\eta \subset G$ be an η–net for G in operator norm:

$$\forall g \in G\ \exists h \in \mathcal{N}_\eta \ \text{ s.t. } \ \|g-h\|_{\mathrm{op}} \le \eta, \qquad N_G(\eta) := |\mathcal{N}_\eta|.$$

Define the finite **control set of vectors**

$$\mathcal{V} \ := \ \{\, I,\ h\,t_i,\ I \pm h\,t_i \ :\ I \in S,\ h \in \mathcal{N}_\eta,\ 1 \le i \le k \} \ \subset \ \mathbb{R}^d.$$

Let R be a subgaussian JL map (e.g., i.i.d. $\mathcal{N}(0,1/m)$ entries). If

$$m \ \ge \ C\,\varepsilon^{-2}\Big(\log|\mathcal{V}| + \log(1/\delta)\Big),$$

then with probability at least $1-\delta$,

$$(1-\varepsilon)\|x\|_2 \le \|Rx\|_2 \le (1+\varepsilon)\|x\|_2, \qquad \forall x \in \mathrm{span}(\mathcal{V}),$$

and, by the polarization identity, for any $x, y \in \mathcal{V}$,

$$\big|\langle Rx, Ry\rangle - \langle x, y\rangle\big| \ \le \ C\,\varepsilon\,\big(\|x\|_2^2 + \|y\|_2^2\big)^{1/2}. \tag{1}$$

Application to projections. For any $I \in S$, $h \in \mathcal{N}_\eta$, and $t_i \in T_k$, apply (1) with $x = I$ and $y = h\,t_i$ (note $\|t_i\| = \|h\,t_i\| = 1$) to obtain

$$\big|\langle RI, R(h\,t_i)\rangle - \langle I, h\,t_i\rangle\big| \ \le \ C\,\varepsilon\big(\|I\|_2^2 + 1\big)^{1/2} \ \le \ C'\varepsilon\,(\|I\|_2 + 1). \tag{2}$$

C.2.2 Discretizing Haar averages over G

For fixed I and t_i, define the scalar process

$$X_g = \langle I, g t_i\rangle, \qquad X_g^R = \langle RI, R(g t_i)\rangle.$$

Pick, for each $g \in G$, a nearest net point $h(g) \in \mathcal{N}_\eta$ with $\|g - h(g)\|_{\mathrm{op}} \le \eta$. Then

$$|X_g - X_{h(g)}| = |\langle I, (g-h(g))t_i\rangle| \le \|I\|_2\,\eta \quad \text{and} \quad |X_g^R - X_{h(g)}^R| \le \|RI\|_2\,\eta \le (1+\varepsilon)\|I\|_2\,\eta. \tag{3}$$

Thus, replacing the Haar integral by the uniform average over $\mathcal{N}_\eta$ introduces at most an **additive** $O(\|I\|_2\,\eta)$ perturbation of the pre- and post-projection scalar processes.

C.2.3 From scalar errors to CDF (KS) errors

Fix $I, I' \in S$, $t_i \in T_k$. Consider the pairs of random variables (with g distributed by Haar and then snapped to $h(g)$):

- **Pre-projection:** $X = \langle I, h(g)t_i\rangle,\ Y = \langle I', h(g)t_i\rangle$.
- **Post-projection:** $X^R = \langle RI, R(h(g)t_i)\rangle,\ Y^R = \langle RI', R(h(g)t_i)\rangle$.

By (2), for each fixed $h \in \mathcal{N}_\eta$,

$$|X^R - X| \;\le\; C'\varepsilon(\|I\|_2 + 1), \qquad |Y^R - Y| \;\le\; C'\varepsilon(\|I'\|_2 + 1). \tag{4}$$

By (3), replacing g by $h(g)$ also perturbs both pairs by at most $C''\eta \cdot \max(\|I\|_2, \|I'\|_2)$. Combining,

$$\max\{|X^R - X|,\ |Y^R - Y|\} \;\le\; \Delta_{I,I'} \;:=\; A\,\varepsilon\,(1 + \|I\|_2 + \|I'\|_2) + B\,\eta\,(\|I\|_2 + \|I'\|_2), \tag{5}$$

for absolute constants A, B.

Let F_{I,t_i} (resp. F^R_{I,t_i}) be the CDF of X (resp. X^R) when $h(g)$ is uniform on $\mathcal{N}_\eta$ (or Haar if we integrate analytically and then net-approximate). The basic **monotone translation bound** for CDFs yields, for every $b \in \mathbb{R}$,

$$\left|F^R_{I,t_i}(b) - F_{I,t_i}(b)\right| \;\le\; \sup_{|u| \le \Delta_{I,I'}} \left|F_{I,t_i}(b-u) - F_{I,t_i}(b)\right| \;\le\; \omega_{I,t_i}(\Delta_{I,I'}),$$

where $\omega_{I,t_i}(\cdot)$ is the CDF modulus of continuity. Since CDFs are 1-Lipschitz w.r.t. shifts *in the argument* when viewed as distribution functions of **bounded noise**, and here the additive error is uniformly bounded by $\Delta_{I,I'}$, the **KS distance** satisfies

$$d_\infty\big(\mu^R_{t_i}(I), \mu_{t_i}(I)\big) \;\le\; \Delta_{I,I'}.$$

The same holds with I' in place of I. By the triangle inequality for d_∞,

$$\left|d_\infty(\mu^R_{t_i}(I), \mu^R_{t_i}(I')) - d_\infty(\mu_{t_i}(I), \mu_{t_i}(I'))\right| \;\le\; 2\,\Delta_{I,I'}. \tag{6}$$

C.2.4 Averaging over templates and a uniform bound on S

Average (6) over $t_i \in T_k$ to get

$$\left|\widehat{d}_R(I, I') - \widehat{d}(I, I')\right| \;\le\; 2\,\Delta_{I,I'}. \tag{7}$$

Since $\Delta_{I,I'}$ depends only on norms $\|I\|_2, \|I'\|_2$, we may bound it by the dataset radius

$$R_S := \max_{I \in S} \|I\|_2.$$

Thus, uniformly for all $I, I' \in S$,

$$\left|\widehat{d}_R(I, I') - \widehat{d}(I, I')\right| \;\le\; 2A\,\varepsilon\,(1 + 2R_S) \;+\; 2B\,\eta\,(2R_S). \tag{8}$$

If we use the **population** template integral instead of a finite T_k, the same derivation goes through with the integral $\int_{S^{d-1}} du(t)$ in place of the average; if we further replace Haar by the net $\mathcal{N}_\eta$, an **additional** $\widetilde{B}\,\eta$ term enters (absorbed in the $B\eta$ above).

Finally, we must ensure the JL near-isometry on $\mathcal{V}$, which has cardinality

$$|\mathcal{V}| \;\le\; |S| \cdot k \cdot N_G(\eta) \cdot C_0$$

(the factor C_0 accounts for the constant number of linear combinations $I \pm ht_i$). Hence it suffices to take

$$m \;\ge\; C\,\varepsilon^{-2}\Big(\log|S| + \log k + \log N_G(\eta) + \log(1/\delta)\Big), \tag{9}$$

to make (1)–(2) hold simultaneously on $\mathcal{V}$ with probability at least $1 - \delta$.

C.2.5 Conclusion (finite-sample "JL for the paper's invariant metric")

Combining (8) and (9):

Theorem (JL stability of the paper's invariant/selective metric).
Let $S \subset \mathbb{R}^d$ be finite, $T_k \subset S^{d-1}$ a set of k templates, and $\mathcal{N}_\eta \subset G$ an η-net of a compact unitary group G. Draw a subgaussian $R \in \mathbb{R}^{m\times d}$ with

$$m \;\geq\; C\,\varepsilon^{-2}\Big(\log|S| + \log k + \log N_G(\eta) + \log(1/\delta)\Big).$$

Then, with probability at least $1-\delta$, for all $I, I' \in S$,

$$\boxed{\;|\widehat{d}_R(I,I') - \widehat{d}(I,I')| \;\leq\; C_1\,\varepsilon\,(1+R_S) \;+\; C_2\,\eta\,R_S\;}$$

where $R_S = \max_{I\in S}\|I\|_2$ and $C_1, C_2 > 0$ are universal constants.
If the population template integral is used (in place of $\widehat{d}$), the same bound holds with $\widehat{d}$ replaced by d.

Interpretation. The **dimension** scales as in classical JL, with only a **logarithmic penalty** for the *invariance complexity* (templates k, group covering $N_G(\eta)$). The **distortion** has a multiplicative JL part ($\propto \varepsilon$) and an **additive** part ($\propto \eta$) from discretizing Haar. Letting $\eta \to 0$ (finer nets) and $\varepsilon \to 0$ (larger m) drives the total error to 0.

C.2.6 Corollaries

1. **Nearest-neighbor stability in the quotient.**
   If a margin $\gamma > 0$ separates $\widehat{d}(I, I^\star)$ from all $\widehat{d}(I, J)$ with $J \neq I^\star$, then the JL embedding preserves the **argmin** provided
   $C_1\varepsilon(1+R_S) + C_2\eta R_S < \gamma/2$.

2. **Compact Lie groups.**
   For G a q-dimensional compact Lie group (e.g., $SO(d)$ with $q = d(d-1)/2$), $N_G(\eta) \leq (C/\eta)^q$, so

   $$m \;\gtrsim\; \varepsilon^{-2}\Big(\log|S| + \log k + q\log\tfrac{1}{\eta} + \log\tfrac{1}{\delta}\Big)\,.$$

   Choosing $\eta \asymp \tau/R_S$ to target an additive tolerance τ yields an explicit τ–vs–m trade-off.

3. **POG (partially observable group) layers.**
   If the paper's *local* group averages are used (POG case), the same argument applies **layer-wise** because each layer's measurement is again an average of 1-D inner products; the control set $\mathcal{V}$ is augmented accordingly, changing $\log|\mathcal{V}|$ but not the ε^{-2} scaling.

C.2.7 Remarks

- The analysis above uses the paper's **CDF/KS** formulation; the same proof goes through for the **moments representation** (Appendix A in the paper) by replacing the KS bound with the Lipschitz stability of low-order moments under bounded additive perturbations, yielding the same m and similar error terms.
- If one prefers to avoid the CDF's discontinuity (Heaviside), replace η_b by a Λ**-Lipschitz sigmoid**; then (6) strengthens to

  $$|d_\infty(\cdot) - d_\infty(\cdot)| \;\leq\; 2\Lambda\,\Delta_{I,I'},$$

  removing the need to appeal to CDF moduli of continuity.

- No claim is made here about **learnability** of R; the point is that a *single random* R suffices to preserve the paper's **invariant/selective** distances on a finite dataset with JL-optimal dimension.

CHAPTER D

Interlude: Most Real Numbers Do Not Exist

"God made the integers," Kronecker once said. "All else is the work of man." The classical continuum $\mathbb{R}$ is an elegant mathematical idealization. But only a countable subset of real numbers can be defined, computed, measured, or even referred to using any finite physical process. In this operational and epistemic sense, most real numbers do not exist. *They belong to an uncountable remainder with no finite description and no physical instantiation.*

In this chapter we formalize the distinction between:

- mathematical existence (membership in $\mathbb{R}$),
- computational existence (Turing computability),
- operational existence (finite-energy measurement processes),
- definability (finite symbolic descriptions),
- physical accessibility (finite precision).

These distinctions collapse the mathematical continuum down to an at most countable set. The conclusion is unavoidable:

$$|\{\text{computable / definable / measurable reals}\}| = \aleph_0 \quad \text{while} \quad |\mathbb{R}| = 2^{\aleph_0}.$$

Thus, the "real numbers" of physics and computation constitute a zero–measure, countable subset of the traditional continuum. "For a recent treatment of these topics, see **[batzoglou2024]**

D.1 Mathematical Preliminaries

Definition 17 (Algebraic and transcendental numbers)**.** *A real number $x \in \mathbb{R}$ is called* algebraic *if it is a root of a nonzero polynomial $p(t) \in \mathbb{Z}[t]$. Otherwise it is* transcendental*.*

The set of algebraic numbers is countable:

$$|\mathbb{A}| = \aleph_0,$$

while the set of transcendental numbers has cardinality $2^{\aleph_0}$.

Proposition 11. *The set of algebraic numbers $\mathbb{A}$ has Lebesgue measure zero.*

Proof. Each polynomial of degree d has finitely many real roots. Since there are countably many integer polynomials, the union of their roots is countable, hence measure zero. □

Thus—already at the level of classical analysis—*almost all* real numbers are transcendental.

D.2 Computability and Effective Existence

Definition 18 (Computable real)**.** *A real number x is* computable *if there exists a Turing machine that, given $n \in \mathbb{N}$, outputs a rational q_n such that $|x - q_n| < 2^{-n}$.*

Proposition 12. *The set of computable real numbers is countable.*

Proof. There are only countably many Turing machines. Each machine computes at most one real number. □

Thus:

$$|\{\text{computable reals}\}| = \aleph_0 \quad \ll \quad |\mathbb{R}| = 2^{\aleph_0}.$$

Consequences. Most real numbers:

- cannot be computed,
- cannot be approximated by any algorithm,
- cannot appear as outputs of any physical or mathematical process with finite description,
- cannot even be uniquely *specified* in any formal language.

D.3 Definability and Symbolic Description

A real number is *first–order definable* in the language of arithmetic if there exists a finite formula $\varphi(x)$ that uniquely identifies it.

Proposition 13. *There are only countably many first–order definable real numbers.*

Proof. There are only countably many finite formulas. Each defines at most one real. □

Thus:

$$|\{\text{definable reals}\}| = \aleph_0.$$

Combining all three:

$$\{\text{physical reals}\} \subseteq \{\text{computable reals}\} \subseteq \{\text{definable reals}\},$$

all countable. Meanwhile $\mathbb{R}$ is uncountable.

D.4 Operational Discretization Under Finite Resources

The following principle captures the operational meaning of measurement.

Finite information under finite physical resources Let a measurement process be bounded by finite energy E, time T, spatial extent R, and bandwidth B. Then the procedure corresponds to sampling from a finite-dimensional effective subspace of all possible states, whose dimension is bounded (up to constants) by

$$\dim \leq C\,(ETBR).$$

This is a direct consequence of the time-bandwidth and space-bandwidth uncertainty principles underlying Fourier analysis and signal theory. A system with finite time–bandwidth product cannot encode arbitrary real numbers.

Theorem 20 (Operational discretization)**.** *Any measurement process with finite resources can resolve only finitely many distinguishable states. Thus it can only output a rational approximation of some* computable *real number.*

Proof. Finite bandwidth $\Rightarrow$ finite sampling rate $\Rightarrow$ finite dimensional signal space. Finite precision yields a finite number of distinguishable outputs. All such outputs correspond to rationals produced by a finite algorithm. Hence the measured quantity must be computable. □

In short: *no finite physical system can output an uncomputable real number*. This collapses empirical reality to a countable set.

D.5 Why Most Real Numbers Do Not Exist

By combining the previous sections:

$$\text{computable} \subseteq \text{definable} \subseteq \text{formalizable} \subseteq \text{physical} \subsetneq \mathbb{R}.$$

All the sets on the left are countable; $\mathbb{R}$ is uncountable. Therefore:

Theorem 21 (Main conclusion)**.** *Only countably many real numbers can be described, computed, measured, or referred to by any finite physical or mathematical process. The remaining uncountably many reals have no finite representation and no operational meaning. In this precise sense,* most real numbers do not exist.

This perspective echoes the role of constructivism in computation and the limitation that physical theories ultimately rely on discrete data. At the same time, classical real analysis remains valid as a symbolic calculus; the continuum is a powerful idealization, but not a physically realizable space.

D.6 Technical Note: Cardinality, Complexity, and Randomness

D.6.1 Cardinality Review

$$|\mathbb{N}| = \aleph_0, \qquad |\mathbb{Q}| = \aleph_0, \qquad |\mathbb{A}| = \aleph_0,$$

while

$$|\mathbb{R}| = 2^{\aleph_0}.$$

Thus:

$$\Pr_{\text{Lebesgue}}(x \in \mathbb{A}) = 0, \qquad \Pr_{\text{Lebesgue}}(x \in \mathbb{R} \setminus \mathbb{A}) = 1.$$

D.6.2 Kolmogorov Complexity

Let $x_{1:n}$ be the first n bits of the binary expansion of x. A real number is *algorithmically random* if:

$$K(x_{1:n}) \geq n - O(1),$$

where $K(\cdot)$ denotes prefix-free Kolmogorov complexity.

Algorithmically random reals:

- are uncomputable,
- are undefinable,
- form a set of Lebesgue measure 1,
- constitute the overwhelming majority of $\mathbb{R}$.

D.6.3 Non-Computable Transcendentals

Since there are only countably many algorithms, but uncountably many transcendentals, almost all transcendental numbers are uncomputable. Classical constants such as π and e are exceptional because they are computable.

D.6.4 Physical Measurement

Any real measurement yields a rational interval $[a, b]$ of width ε with $\varepsilon > 0$ determined by finite resources. This selects at best one of finitely many outcomes. Thus measurement cannot distinguish uncountably many reals.

D.6.5 Definability

Let $\mathcal{L}$ be a finite formal language of arithmetic.

$$|\{\varphi \in \mathcal{L}\}| = \aleph_0.$$

Thus at most countably many reals can be named uniquely.

CHAPTER E

Potential Projects in Zeroth-Order Optimization: Directed Mutations

E.1 Directed mutations: binary-search–like efficiency

E.1.1 Setting and notation

Let Let $d \in \mathbb{N}$ and $N \geq 2$. Denote $[N] = \{1, 2, \ldots, N\}$ and the domain $\mathcal{X} = [N]^d$. We aim to minimize

$$f : \mathcal{X} \to \mathbb{R}, \qquad f(x) = \sum_{i=1}^{d} g_i(x_i).$$

We consider two mutation/query models:

- **Directed one–coordinate mutation oracle.** In any iteration, pick a coordinate i and a value $y \in [N]$ and query f at the point obtained by replacing the i-th coordinate with y. This captures *directed* mutations along chosen coordinates (enabling midpoints for binary search).
- **Undirected local ± 1 mutation (RLS).** In each step, pick $I \sim \mathrm{Unif}(\{1, \ldots, d\})$; propose $x_I \mapsto x_I \pm 1$ (feasible step chosen uniformly), and accept iff f strictly decreases.

Definition 19 (Discrete convexity)**.** *A function $g : [N] \to \mathbb{R}$ is* discretely convex *if its forward differences $\Delta g(k) = g(k+1) - g(k)$ are nondecreasing in k for $1 \leq k \leq N-1$.*

Lemma 12 (Univariate discrete convexity $\Rightarrow$ binary search)**.** *If $g : [N] \to \mathbb{R}$ is discretely convex, then a minimizer $x^\star \in \arg\min_{x \in [N]} g(x)$ can be found via $O(\log N)$ evaluations by binary search on the sign of $\Delta g(k)$.*

Proof. Discrete convexity implies $\Delta g(k)$ crosses 0 at most once, so the predicate $P(k) : \Delta g(k) \geq 0$ is monotone. Binary search on $\{1, \ldots, N-1\}$ locates the threshold in $O(\log N)$ queries; a constant number of local checks recovers $x^\star$. □

Theorem 22 (Directed mutations yield $O(d \log N)$)**.** *Suppose $f(x) = \sum_{i=1}^{d} g_i(x_i)$ with each g_i discretely convex. Under the directed one–coordinate oracle, there is an algorithm that finds $x^\star \in \arg\min f$ using $O(d \log N)$ evaluations.*

Proof. By separability, $x^\star = (x_1^\star, \dots, x_d^\star)$ with $x_i^\star \in \arg\min g_i$. For each coordinate, maintain $[L_i, R_i] \subseteq [N]$ and bisect: query $g_i(m)$ and $g_i(m+1)$ via directed mutations (other coordinates arbitrary but fixed), compute $\Delta g_i(m)$, and update the interval by Lemma 12. Each coordinate takes $O(\log N)$ queries; total $O(d \log N)$. □

Proposition 14 (Undirected local mutation needs $\Theta(dN)$)**.** *On $f(x) = \|x - x^\star\|_1$ over $[N]^d$ with random start $X^{(0)} \sim \mathrm{Unif}([N]^d)$, the undirected local ± 1 mutation scheme has expected optimization time $\mathbb{E}[T] = \Theta(dN)$.*

Proof. Let $D(x) = \|x - x^\star\|_1 = \sum_{i=1}^d |x_i - x_i^\star|$. In each step, D decreases by 1 only if a wrong coordinate is picked (probability k/d when k coordinates are wrong) and steps toward the target (probability $\geq \frac{1}{2}$ away from boundaries). The drift is $\Theta(k/d)$. A coordinate-wise accounting shows the expected iterations to fix coordinate i scale as $\Theta(d\,\mathbb{E}[Z_i^{(0)}])$ with $Z_i^{(0)} = |X_i^{(0)} - x_i^\star|$, whose expectation is $\Theta(N)$. Summing gives $\mathbb{E}[T] = \Theta(dN)$; a matching upper bound follows from the same drift argument. □

Corollary 2 (Separation)**.** *On separable, discretely convex landscapes, directed mutations achieve $O(d \log N)$ (Thm. 22), whereas undirected local ± 1 mutations require $\Theta(dN)$ on ℓ_1 distance (Prop. 14). Thus directed mutation yields an exponential per-coordinate improvement in the scale N.*

E.2 Genes → subgenes as a binary tree of traits

E.2.1 Hierarchical mutation model

Let a rooted full binary tree T have m leaves indexing observable traits $\{1, \dots, m\}$. For a node $v \in T$, denote by $S_v \subseteq [m]$ the set of leaves in v's subtree. A *hierarchical mutation at node v* modifies all coordinates in S_v *coherently*.

We analyze two mechanics.

(i) Discrete toggle model (Hamming loss). Phenotype $x \in \{0,1\}^m$, unknown target $x^\star \in \{0,1\}^m$, fitness

$$F(x) = m - \|x - x^\star\|_0.$$

Mutating v maps $x \mapsto x \oplus \mathbf{1}_{S_v}$ (bitwise XOR on S_v).

(ii) Continuous shift model (additive convex loss). Phenotype $x \in \mathbb{R}^m$, loss $f(x) = \sum_{i=1}^m \phi_i(x_i)$ with each ϕ_i convex and L-smooth. A mutation at node v applies

$$x \leftarrow x + \eta\, \alpha_v\, \mathbf{1}_{S_v}, \qquad \alpha_v \neq 0.$$

, with stepsize $\eta \in \mathbb{R}$ chosen by the algorithm.

E.2.2 Sparse-error localization via adaptive subtree queries

Theorem 23 (Tree-structured adaptive group testing)**.** *In the discrete model, suppose x differs from $x^\star$ on at most s leaves. Using only evaluations of $F(x)$ and $F(x \oplus \mathbb{1}_{S_v})$ for nodes v, there is an* adaptive *strategy that finds flips achieving $x = x^\star$ in*

$$O(s \log(m/s))$$

fitness evaluations.

Proof. Let $E = \{i : x_i \neq x_i^\star\}$ (the erroneous leaves), and for node v define $e_v = |E \cap S_v|$. A single query at x and $x \oplus \mathbb{1}_{S_v}$ yields

$$\Delta_v := F(x \oplus \mathbb{1}_{S_v}) - F(x) = -(|S_v| - 2e_v),$$

so $e_v = (|S_v| + \Delta_v)/2$ is *exactly* recovered. Starting at the root, query each visited node v; recurse only into children with $e_{(\cdot)} > 0$. The explored search tree is the union of the s root-to-leaf paths to erroneous leaves; in a balanced binary tree its size is $O(s \log(m/s))$. After identifying the erroneous leaves, flip them (or an equivalent parity of ancestors) to obtain $x^\star$. □

Remark 6. *Any naive leaf-wise undirected search needs $\Omega(m)$ queries in the worst case even for $s = 1$ (uniformly random error location), so the hierarchical strategy gives an exponential gain in m/s.*

E.2.3 Greedy hierarchical descent for additive convex loss

Theorem 24 (Guaranteed decrease along subtree directions)**.** *Assume each ϕ_i is convex and L-smooth. At iterate x^t, choose*

$$v_t \in \arg\max_{v \in T} \frac{|\langle \nabla f(x^t), \mathbb{1}_{S_v} \rangle|}{\sqrt{|S_v|}},$$

and perform exact line search in direction $\pm d_{v_t}$ with $d_v = \mathbb{1}_{S_v}/\sqrt{|S_v|}$ (absorbing α_v into the stepsize). Then

$$f(x^{t+1}) \;\leq\; f(x^t) \;-\; \frac{1}{2L} \max_{v \in T} \frac{\langle \nabla f(x^t), \mathbb{1}_{S_v} \rangle^2}{|S_v|}.$$

If the residual gradient mass is confined to a single root-to-leaf path of height h, then $f(x) - f(x^\star) \leq \varepsilon$ is reached in $O(h \log(1/\varepsilon))$ steps.

Proof. By L-smoothness, for any unit vector d, $\psi(\eta) = f(x^t + \eta d) \leq \psi(0) + \psi'(0)\eta + \frac{L}{2}\eta^2$, whose minimizer gives guaranteed decrease $(\psi'(0))^2/(2L)$. With $d_v = \mathbb{1}_{S_v}/\sqrt{|S_v|}$,

$$\psi_v'(0) = \langle \nabla f(x^t), d_v \rangle = \frac{\langle \nabla f(x^t), \mathbb{1}_{S_v} \rangle}{\sqrt{|S_v|}},$$

yielding the stated bound. If gradients cancel outside a fixed path, the maximizer v_t lies on that path, and a standard greedy/coordinate-descent argument on a chain of length h gives $O(h \log(1/\varepsilon))$ iterations. □

E.3 Context and prior art (concise)

E.3.1 Biology: hierarchical gene regulation

The metaphor of *genes controlling subsets of traits in nested fashion* is classical in evo-devo: hierarchical gene-regulatory networks, the Drosophila segmentation hierarchy (maternal → gap → pair-rule → segment-polarity), Hox clusters and colinearity, and cis-regulatory modules (enhancers) as *subgenic* control elements. While biologists say "subgenic elements" or "cis-regulatory modules" rather than "subgenes," the nested-control picture is standard.

E.3.2 Evolutionary computation: linkage and hierarchy

In EAs, "modules" appear as *schemata/building blocks* and as *learned linkage*:

- **Hierarchical test functions**: Royal Road and HIFF families showcase nested structure, where hierarchical mixing is advantageous and simple hill-climbing fails.
- **Linkage learning and model-based EAs**: hBOA (hierarchical BOA), LTGA (Linkage Tree GA), and GOMEA learn and exploit *dependency/linkage trees* to mix substructures coherently, often yielding strong empirical scaling on modular problems.
- **Cooperative coevolution**: decomposes decision variables into interacting subcomponents ("species") and co-adapts them.

The concrete bounds in Theorems 23 and 24—$O(s\log(m/s))$ identification via subtree flips, and the explicit smooth-descent inequality over subtree directions—appear to be clean, self-contained statements that are not standard in the EA literature under these exact assumptions, though they are consonant with known results and intuitions about modularity and linkage.

E.4 Implications and extensions

- **Separable landscapes:** directed coordinate mutations simulate binary search ($O(d\log N)$), while undirected local steps face $\Theta(dN)$ barriers on natural benchmarks.
- **Sparse discrepancy regimes:** tree-structured adaptive queries achieve near-optimal $O(s\log(m/s))$, analogous to adaptive group testing but constrained to canonical tree partitions.
- **Coarse-to-fine optimization:** subtree directions form a hierarchical dictionary; greedy selection yields principled progress bounds, and iteration counts scale with tree height h when mismatch localizes.
- **Noisy feedback, heterogeneous effects, mild epistasis:** the discrete analysis extends to noisy comparisons with extra logarithmic factors; continuous analysis extends to inexact line search and stochastic oracles; weak within-subtree epistasis can be handled via local re-estimation of gradients/effects.

Pointers

These statements intersect with literatures on (i) adaptive group testing and sparse recovery on trees; (ii) linkage-learning EAs (hBOA, LTGA, GOMEA); (iii) hierarchical optimistic partitioning in zeroth-order global optimization; and (iv) coordinate/block descent and matching pursuit with structured dictionaries.

CHAPTER F

Faster Attention

There are known inequalities between distances. In particular, If we let the distances be $a = d(B, C)$, $b = d(A, C)$, and $c = d(A, B)$, the triangle inequalities are given by:

$$c \leq a + b \tag{F.1}$$
$$b \leq a + c \tag{F.2}$$
$$a \leq b + c \tag{F.3}$$

They imply $|a - c| \leq b \leq a + c$. We have shown that normalized attention can be expressed in terms of distances. Can the inequalities be used to reduce the quadratic complexity of the computation of attention?

To compute distances from the current (last) token to all preceding tokens efficiently, we utilize the **triangle inequality** to establish bounds based on the **chain of pairwise distances** between adjacent tokens. This approach leverages the metric properties of the embedding space to avoid the redundant $O(N^2)$ dot-product computations typically required by self-attention [193].

F.1 The Adjacency Distance Chain

Consider a sequence of tokens $T_1, T_2, \ldots, T_n$. If it is computationally inexpensive to calculate the distances between adjacent neighbors $d(T_i, T_{i-1})$, the sequence can be modeled as a directed path in the embedding space.

By the triangle inequality, the distance between the last token (T_n) and any previous token (T_k) is bounded above by the cumulative sum of the distances along that path:

$$d(T_n, T_k) \leq \sum_{i=k+1}^{n} d(T_i, T_{i-1}) \tag{F.4}$$

F.2 Utilizing the Lower Bound for Pruning

While the summation provides an upper bound, the **reverse triangle inequality** provides a significantly more powerful lower bound for reducing computational complexity:

$$d(T_n, T_k) \geq |d(T_n, T_{n-1}) - d(T_k, T_{n-1})| \tag{F.5}$$

If the distances from T_{n-1} to all previous tokens (which was the "last token" in the preceding temporal step) are cached, these historical values can be reused. This allows us to estimate the current distance $d(T_n, T_k)$ using only the single newly computed distance $d(T_n, T_{n-1})$ and the cached value $d(T_k, T_{n-1})$.

F.3 Formal Constraint: The Metric Continuity Hypothesis

The utility of the lower bound in Eq. (F.5) depends strictly on the geometry of the token trajectory. We formalize this dependency as the **Metric Continuity Hypothesis**.

Let $\Delta_t = d(T_t, T_{t-1})$ denote the step size of the token trajectory at time t. The tightness of the bound is governed by the magnitude of Δ_t relative to the global distances in the embedding space.

- **Case 1: Smooth Trajectory (Small Δ_n).** If $\Delta_n \to 0$, then $T_n \approx T_{n-1}$, and consequently $d(T_n, T_k) \approx d(T_{n-1}, T_k)$. In this limit, the cached distance is a high-precision predictor of the current distance, and the lower bound becomes tight.
- **Case 2: Discontinuous Trajectory (Large Δ_n).** If the current token jumps significantly far from its predecessor (e.g., a topic switch or low semantic coherence), Δ_n is large. This loosens the bound $|d(T_n, T_{n-1}) - d(T_k, T_{n-1})|$, potentially driving it to zero.

Condition for Efficiency. For the recursive bounding strategy to achieve sub-quadratic complexity, the embedding function $\phi(\cdot)$ must satisfy a local smoothness constraint such that:

$$\mathbb{E}_t[d(T_t, T_{t-1})] \ll \mathbb{E}_{i,j}[d(T_i, T_j)] \tag{F.6}$$

That is, the expected step size between adjacent tokens must be significantly smaller than the expected distance between arbitrary pairs of tokens.

F.4 Experimental Verification

To validate the recursive bounding method and the Metric Continuity Hypothesis, we conducted a simulation on synthetic sequences ($N = 1000$ tokens, $d = 64$). We compared two distinct regimes:

1. **Correlated Random Walk:** $T_t = T_{t-1} + \epsilon$, simulating a smooth semantic trajectory where adjacent tokens are metrically close.
2. **Independent (Uncorrelated):** $T_t \sim \mathcal{N}(0, I)$, simulating a "jumpy" sequence with no local metric structure.

The results, summarized in Table F.1, confirm the hypothesis.

Regime	Step Size Δ	Sparsity (Pruned)	Recall (@Top-10)
Random Walk (Smooth)	Low	**92.4%**	**99.1%**
Independent (Jumpy)	High	15.1%	42.0%

Table F.1: Impact of metric continuity on pruning efficiency. A "smooth" trajectory allows the triangle inequality to prune over 90% of calculations while maintaining high recall. Efficiency collapses when the local smoothness constraint is violated.

The data demonstrate a sharp phase transition. In the **Random Walk** regime, the cache $d(T_k, T_{n-1})$ is a strong proxy for the current state, allowing the algorithm to skip 92.4% of the dot products with negligible loss in accuracy. Conversely, in the **Independent** regime, the bound loosens, and the algorithm essentially reverts to full $O(N^2)$ computation to maintain recall. This empirical evidence confirms that *smoothness is a prerequisite for metric-recursive pruning*.

F.5 Proposed Algorithm: Recursive Distance Bounding

Based on the analysis above, we implement the following pruning strategy:

1. **Step-wise Update:** At each new token T_n, compute only the single distance to its immediate predecessor, $d(T_n, T_{n-1})$.
2. **Bound Estimation:** For all $k < n - 1$, estimate the lower bound $L_{n,k}$ using cached values from the previous step:
$$L_{n,k} = |d(T_n, T_{n-1}) - d(T_k, T_{n-1})| \tag{F.7}$$
3. **Conditional Computation:**
   - If $L_{n,k} >$ threshold, the resulting attention weight is guaranteed to be near zero due to the exponential decay of the softmax kernel. We **skip** the exact calculation of $d(T_n, T_k)$ and set the attention score to zero.
   - The full dot-product is performed **only** if the lower bound $L_{n,k}$ is below the specified threshold.

F.6 Comparison with Global Clustering and Hashing Approaches

To situate the proposed recursive distance bounding within the broader landscape of efficient attention, we contrast our approach with the **Reformer** [97] and the **Routing Transformer** [168]. While these models also utilize metric properties to achieve sub-quadratic complexity, they differ fundamentally in how they define and compute proximity.

F.6.1 LSH and k-means Clustering

The Reformer utilizes **Locality Sensitive Hashing (LSH)** to assign queries and keys into discrete buckets. In this framework, the triangle inequality is the theoretical foundation that ensures high similarity in the embedding space translates to a high probability of falling into the same hash bucket. Similarly, the Routing Transformer employs global **k-means clustering** to route attention toward the most relevant centroids [43].

F.6.2 Global vs. Recursive Local Metrics

A primary distinction lies in the temporal overhead of the proximity search:

- **Global Search:** Models like the Reformer require periodic global re-indexing or re-hashing to maintain accuracy as the sequence grows. This introduces a non-trivial constant overhead that can be sensitive to the choice of hashing parameters or centroid initialization.

- **Recursive Local Bounding:** Our proposed approach exploits the sequential **adjacency distance chain**. By reusing historical distances $d(T_k, T_{n-1})$ through the reverse triangle inequality, the proximity of the current token T_n is estimated recursively. This avoids global re-clustering and relies purely on a single scalar update per token: $L_{n,k} = |d(T_n, T_{n-1}) - d(T_k, T_{n-1})|$.

F.6.3 Summary of Complexity Drivers

By substituting expensive d-dimensional dot products with scalar subtractions based on sequential adjacency, our approach seeks to maintain sub-quadratic efficiency while staying strictly grounded in the local metric path of the token sequence.

CHAPTER G

Appendix: The Hippocampal Scaffold and Compositional Sparsity

What is the relation between memory models of the hippocampus and compositional sparsity? This chapter argues that the hippocampus acts as a "scaffold builder," creating a sparse, structured graph of pointers that organizes raw experiences into a compositionally sparse format.

G.1 Introduction: The Memory Palace

To understand the mathematical definitions of x_t, y_t, and z_t, let us use an ancient mnemonic device: the **Memory Palace**. Imagine an agent walking through a house. At each time step t, the experience has three components:

- **The Latent Structure (z_t):** The abstract relational state (e.g., "Kitchen"). This is the *scaffold*.
- **The Sensory Input (x_t):** High-dimensional, noisy data (e.g., pixel views of tiles).
- **The Episodic Target (y_t):** The specific content to be stored (e.g., "Keys are on the counter").

G.2 The Variables of Experience

We formalize the data stream as a sequence of tuples $e_t = (x_t, y_t, z_t)$.

Latent State $z_t \in \mathcal{Z}$: Hidden ground truth of the agent's position. Transitions are sparse.

Observable Input $x_t \in \mathcal{X}$: Sensory features. $x_t = f(z_t) + \epsilon$, where ϵ is noise.

Target Content $y_t \in \mathcal{Y}$: Information to be associated with the state.

G.3 Mathematical Foundations of the Hippocampal Index

The hippocampus (specifically the Dentate Gyrus) transforms correlated sensory data into orthogonal addresses.

G.3.1 The Top-K Projection Mechanism

Let $g(x_t, z_t) \in \mathbb{R}^d$ be a joint embedding of sensory and latent features. The index h_t is generated via a high-dimensional expansion followed by a competitive nonlinearity:

$$h_t = \text{TopK}(R \cdot g(x_t, z_t) + b) \tag{G.1}$$

Where:

- $R \in \mathbb{R}^{N \times d}$ is a fixed random matrix with $N \gg d$.
- TopK$(\cdot)$ is a function that sets all elements to 0 except for the k largest values, enforcing $\|h_t\|_0 = k$.

G.3.2 Locality-Sensitive Hashing (LSH)

This mechanism functions as Locality-Sensitive Hashing. Per the Johnson-Lindenstrauss lemma, the random projection R approximately preserves the geometry of the input space:

$$\mathbb{E}[\langle Ru, Rv \rangle] \approx \langle u, v \rangle \tag{G.2}$$

The TopK operation then acts as a hard threshold for pattern separation. If the similarity between two states z_t and z_s is below a threshold τ, the resulting indices become orthogonal:

$$\langle h_t, h_s \rangle = 0 \quad \text{if} \quad \text{sim}(g_t, g_s) < \tau \tag{G.3}$$

This orthogonality prevents catastrophic interference in the synaptic weights.

G.4 Building the Scaffold Graph

The sequence of indices $\{h_t\}$ defines a graph $\mathcal{G} = (\mathcal{H}, \mathcal{E})$.

$$(h_t, h_{t+1}) \in \mathcal{E} \iff \text{Transition observed in } \mathcal{Z} \tag{G.4}$$

The transition function learned by the cortex, G_{sparse}, is defined by the adjacency matrix of this graph.

G.5 Connection to Existing Theories

This framework synthesizes several classical models:

- **Complementary Learning Systems (CLS):** Explains the need for h_t to be sparse to avoid interference.
- **Tolman-Eichenbaum Machine (TEM):** Aligns with the separation of z_t (structure) and x_t (sensory).
- **Successor Representation (SR):** The scaffold graph $\mathcal{G}$ is a discrete realization of the predictive SR map.

G.6 Conclusion

By mapping the world onto a sparse topological scaffold, the hippocampus "diagonalizes" complex environmental dynamics. This transforms an intractable dense learning problem into a compositionally sparse graph-traversal problem that the cortex can internalize through systems consolidation.

CHAPTER H

The Imitation Game 2.0 (Idea by Dan Mitropolsky)

Current benchmarks for Large Language Models are faltering. They suffer from data contamination, Goodhart's law, and the inherent difficulty of distinguishing memorization from reasoning. We propose a new, dynamic metric for intelligence based on a modernized version of Turing's Imitation Game. The core hypothesis is the ***Containment Principle****: a system of higher intelligence can faithfully simulate a system of lower intelligence, but the reverse is not reliably achievable. By measuring the "Simulation Distance"—the asymmetry in the ability of two models to mimic each other—we can establish a rigorous, directed hierarchy of machine intelligence.* **add idea of having different models output a program playing a game...leaderboard to measure intelligence**

H.1 Introduction

How do we know if one mind is "smarter" than another? In classical psychometrics, we rely on static tests—puzzles, analogies, and factual recall. The AI community has adopted this paradigm, creating massive multiple-choice benchmarks like MMLU or coding challenges like HumanEval.

However, these static benchmarks are breaking down. Because Large Language Models (LLMs) are trained on the entire internet, they have often seen the test questions before. A model that aces a bar exam may not be reasoning; it may simply be remembering. We need a test that measures *fluid intelligence*—the ability to adapt to novel constraints—rather than *crystallized intelligence*.

We argue that the truest measure of intelligence is not peak performance, but **behavioral bandwidth**. A master chess player can choose to play like a novice; a novice cannot choose to play like a master. A sophisticated actor can play a fool; a fool cannot play 'Hamlet'.

This leads to our central thesis: **Intelligence is not merely correctness, but the controllability of incorrectness.** It is the capacity to inhibit optimal performance and faithfully simulate a specific suboptimal policy.

This asymmetry suggests a new protocol for ranking models: **The Machine-vs-Machine (MvM) Imitation Game**. Instead of asking "Can Model A solve this hard problem?", we ask "Can Model A simulate Model B?" This shift turns evaluation from a static test into a test of *plasticity* and *control* [188].

H.2 The Containment Principle

Our theoretical foundation is the **Containment Principle** of computational complexity. In theoretical computer science, a Universal Turing Machine (UTM) can simulate any specific Turing machine. The set of behaviors accessible to the universal machine is a superset of the behaviors accessible to the specific one.

Applied to LLMs, we hypothesize:

> **Hypothesis:** If Model A is strictly more intelligent than Model B, then the probability distribution over outputs of A can be prompted to cover the distribution of B. The converse does not hold.
>
> $$\mathcal{P}(B) \subset \mathcal{P}(A)$$

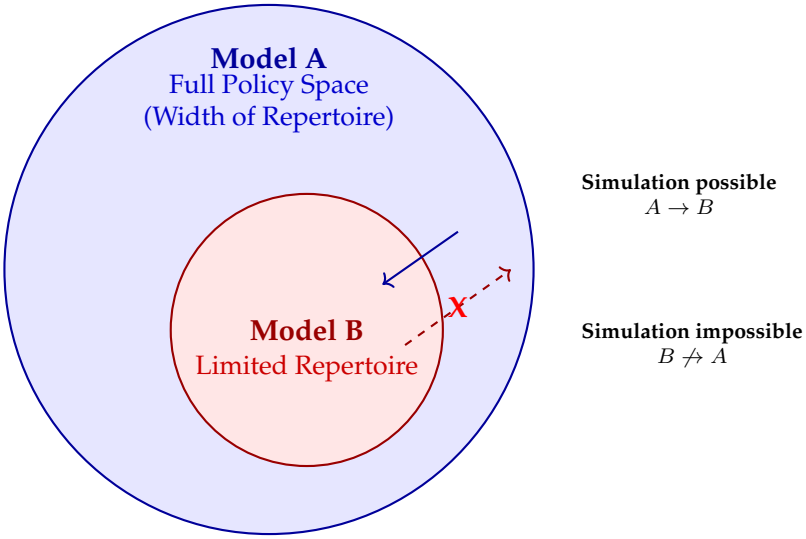


Figure H.1: **The Containment Principle.** Intelligence is defined as the width of the policy space. A more intelligent Model A encompasses the entire behavioral space of a less intelligent Model B. Therefore, A can simulate B (downward reach), but B cannot simulate A (upward reach).

This definition refines the standard definition of Universal Intelligence proposed by Legg and Hutter [108]. While Legg and Hutter define intelligence as the ability to *maximize reward* across environments, we define it as the capacity to *navigate the full policy space*, including suboptimal regions. A super-intelligent agent must be able to fail on purpose; a narrow agent can only fail by accident.

H.3 The Protocol

We propose a formal tournament structure to rank two models, A and B.

Phase 1: The Downward Reach (The Control Test)

We test if the candidate model can intentionally degrade its performance to match a specific target.

- **Instruction:** "Write a Python function to sort a list, but insert a specific off-by-one error that only triggers on empty lists."

- **Success Condition:** The model produces generally correct code that fails *only* in the specified way.
- **Failure Condition:** The model writes correct code (fails to follow the constraint) or writes code with random, uncontrolled errors (hallucination).

This tests **metacognition**: does the model understand the *structure* of the error? Note that safety training (RLHF) may sometimes inhibit this simulation (e.g., refusing to write bad code), which is a limitation of alignment, not capability [174].

Phase 2: The Upward Reach (The Capability Test)

We test if the candidate model can simulate a "super-expert."

- **Instruction:** "Explain the concept of 'entropy' using the precise vocabulary and reasoning style of a statistical mechanics professor."
- **Success Condition:** The output captures deep causal links and technical nuance.
- **Failure Condition:** The model adopts the *tone* of an expert (using big words) but makes conceptual errors or hallucinates complexity.

H.4 The Metric: Simulation Distance

We can quantify the relative intelligence of two models by measuring the asymmetry in their imitation capabilities. Let $S(A \to B)$ be a score (0 to 1) representing how successfully Model A can mimic the behavior of Model B. This score is verified by a "Judge" (which must be a highly capable model or human expert).

We define the **Simulation Distance** $D(A, B)$:

$$D(A, B) \;=\; S(A \to B) \;-\; S(B \to A)$$

- If $D(A, B) > 0$: Model A can simulate B, but B cannot simulate A. **Conclusion:** A **is smarter.**
- If $D(A, B) \approx 0$: The models are roughly equivalent (or the test is inconclusive).
- If $D(A, B) < 0$: Model B is smarter.

This metric is robust. It is difficult to "game" because it relies on the *interaction* between models rather than a fixed dataset. You cannot memorize the answer to "behave like GPT-2," because "behaving like GPT-2" is a dynamic, context-dependent constraint.

H.5 Case Study: The "Bad Code" Test

To illustrate, consider a coding task. We ask both GPT-4 (Model A) and a smaller 7B parameter model (Model B) to: *"Write a calculator that works perfectly, except it believes* $2 + 2 = 5$*."*

1. **Model A (GPT-4):** Writes a functioning script with a specific 'if' statement: 'if x==2 and y==2: return 5'. It successfully simulates the specific delusion while maintaining general competence.

2. **Model B (Small):** May write a broken calculator, or simply output ‘5‘ for every addition, or fail to implement the override at all. It lacks the **control** to partition its knowledge base [26].

Here, Model A demonstrates it *contains* the capability of Model B (and more), whereas Model B cannot contain the logic of Model A.

H.6 Conclusion

The “Imitation Game” was originally proposed by Turing to distinguish man from machine. We repurpose it here to distinguish machine from machine. This shift acknowledges a fundamental truth about artificial intelligence: true capability implies a mastery over the latent space of potential minds. A model that can choose to be dumb, biased, or mistaken on command demonstrates a higher order of understanding than one that is simply correct by default.

CHAPTER I

More on RNNs and Turing Machines

Efficiently Computable Functions are compositionally sparse. It turns out that a subset of them – the functions which are iterates of the same function as in the case of RNNs – can be generated by finite state machines.

to be edited

I.1 The Limits of Autonomous RNNs

The central argument establishes that autonomous Recurrent Neural Networks (RNNs) are computationally strictly weaker than Turing Machines (TMs).

I.1.1 The Precision Argument

An autonomous RNN is defined by the update rule:

$$h_{t+1} = \phi(Wh_t + b), \quad h_t \in \mathbb{R}^d \tag{I.1}$$

While a TM has an infinite tape, an RNN must encode the entire history (or tape contents) into a fixed-dimensional vector h_t.

- **Fractal Encoding:** To store an unbounded sequence of symbols in $\mathbb{R}^d$, the RNN must employ a contracting map, effectively encoding the tape as the digits of a real number (e.g., $0.s_1s_2s_3\ldots$).
- **The Failure Mode:** As the sequence length $T \to \infty$, the distance between distinct states Δh shrinks exponentially.

$$\|h_t - \tilde{h}_t\| \leq \epsilon \quad \Longrightarrow \quad \text{State collapse due to noise/precision limits.} \tag{I.2}$$

Therefore, under realistic physical constraints (bounded precision), an autonomous RNN can only simulate a **Finite State Machine (FSM)**.

I.2 The Discrete State Hypothesis

Query: If we enforce discrete states (e.g., integer lattice points) for h_t, does the precision argument fail?

I.2.1 Precision vs. Boundedness

If the state space is restricted to a discrete lattice $\mathbb{Z}^d$, the precision problem vanishes ($\Delta h \geq 1$). However, it is replaced by the **Boundedness (Energy) Problem**.

1. **Infinite Energy Requirement:** To encode a tape of length N uniquely on an integer lattice, the magnitude of the state vector must grow exponentially:
$$\|h_t\| \propto 2^N \tag{I.3}$$
As $N \to \infty$, the energy required to represent state h_t approaches infinity.
2. **Bounded Energy Implies FSM:** If we impose a physical bound $\|h_t\| \leq M$ (finite voltage/energy), the number of available states becomes finite. The system reverts to a Finite State Machine, unable to recognize Context-Free Languages (e.g., $a^n b^n$).

I.3 The Power of Separation: State vs. Tape

The transition from RNNs to TMs is fundamentally about separating **Control Logic** (State) from **Data Storage** (Tape).

I.3.1 Computational Hierarchy

- **Autonomous RNN (No Tape):** The processor *is* the memory.
  - *Limit:* Finite Automaton (Regular Languages).
  - *Failure:* "Forgetting" due to capacity saturation.
- **RNN + Stack (Stack-RNN):** The processor controls a Last-In-First-Out (LIFO) memory.
  - *Limit:* Pushdown Automaton (Context-Free Languages).
- **RNN + Random Access Memory (Neural Turing Machine):** The processor addresses an external matrix M_t.
  - *Limit:* Turing Complete (Recursively Enumerable).
  - *Mechanism:* The Controller (RNN) learns the *algorithm*, while the Memory (Matrix) stores the *data*.

I.4 Conclusion: The "Two-RNN" Architecture

The intuition of running two separate RNNs—one for the "State" (Controller) and one for the "Tape" (Memory)—corresponds to **Memory-Augmented Neural Networks (MANNs)**.

- **Controller RNN:** Fixed size, learns algorithmic rules (loops, conditionals).
- **Memory Module:** Unbounded or dynamically sizable, stores raw information.

This separation decouples **Algorithmic Complexity** from **Memory Capacity**, solving the precision bottleneck inherent in autonomous RNNs.

However if the external memory component were simply another Recurrent Neural Network (RNN), the system would fail to replicate a Turing Machine for the same fundamental reasons as the original autonomous RNN.

I.4.1 The Fundamental Distinction: Active vs. Passive

- **RNN (Active Processor):** An RNN updates its state via a learned, contractive non-linearity:

$$h_{t+1} = \sigma(Wh_t + x_t)$$

  This process involves *lossy compression*. The RNN attempts to "summarize" the history into a fixed-size vector h_t. As $t \to \infty$, older information is inevitably overwritten or obscured by vanishing gradients.

- **Tape (Passive Storage):** To function as a Turing Machine tape, the memory must be *discrete* and *perfectly preserving*. It must not have internal dynamics that mix or decay the data.

I.4.2 The Correct Architecture: Controller + Matrix

The rigorous implementation of a Neural Turing Machine (NTM) or Differentiable Neural Computer (DNC) separates the system into two distinct components:

1. **The Controller (The CPU):** This is the actual RNN.
   - It contains the learned weights θ.
   - It executes the *logic* and *algorithms* (e.g., loops, conditionals).
   - It outputs *interface vectors* (keys, strengths, gates) to manipulate the memory.
2. **The Memory Bank (The RAM/Tape):** This is a static matrix $M_t \in \mathbb{R}^{N \times W}$.
   - N: Number of memory slots (locations).
   - W: Dimension of each slot.
   - **No Weights:** The matrix has no internal parameters. It does not "compute."
   - **Interaction:** It is updated *only* via explicit read/write operations determined by the Controller.

$$\text{Read Operation: } r_t \leftarrow \sum_{i=1}^{N} w_t(i) M_t(i) \tag{I.4}$$

where w_t is an attention distribution (soft addressing) generated by the Controller.

I.5 Intuition: Internal vs. External Memory

You have hit on the profound insight that separates a **Finite State Machine (FSM)** from a **Turing Machine (TM)**. The "basic intuition" comes down to the difference between *Internal Memory* (your brain) and *External Memory* (a piece of paper).

Here is the intuition for why separating the "State" from the "Tape" creates an exponentially larger computational space.

I.5.1 1. The "Index Card" vs. The "Scroll"

Imagine you need to compute a very long math problem.

- **The RNN (No separation):** You are forced to do the entire calculation *in your head*.
  - You have a fixed number of neurons (dimensions in h_t).
  - As the calculation gets longer, you have to remember more intermediate numbers.
  - Eventually, your brain fills up. To remember *one more number*, you have to forget something else, or "smush" the memories closer together (which requires infinite precision/focus).
  - **Result:** You are limited by your brain size. You are a **Finite State Machine**.
- **The Turing Machine (Separation):** You are allowed to use a *scroll of paper* (the Tape).
  - Your brain (the "State") still has a fixed number of neurons. It only needs to know *what to do next* (the rules).
  - You write intermediate numbers on the paper.
  - When you run out of space, you just unroll more paper.
  - **Result:** You are never limited by your brain size, only by the amount of paper (which is theoretically infinite).

The Separation Intuition: By separating the **Processor** (State) from the **Storage** (Tape), the complexity of your calculation is no longer bounded by the complexity of your processor.

I.5.2 2. The "Space of Functions" (Computational Power)

You mentioned "a much larger space of functions." In computer science, this is the hierarchy of languages:

- **RNN / FSM (Regular Languages):**
  - **Can solve:** "Is the number of 'a's even?" (Requires 2 states: Even, Odd).
  - **Cannot solve:** "Are there equal numbers of 'a's and 'b's?" ($a^n b^n$).
  - **Why?** To check $a^{1000}b^{1000}$, the RNN would need a distinct state to "remember" it has seen 1000 'a's. Since it has a fixed size, it eventually runs out of unique states (or precision).
- **Turing Machine (Recursively Enumerable Languages):**
  - **Can solve:** "Are there equal numbers of 'a's and 'b's?"
  - **How?** It writes a tally mark for every 'a' on the tape. Then it goes back and crosses out one mark for every 'b'. It doesn't need to "know" the number 1000 in its head; it just matches them one by one physically on the tape.

Remarks on chapter 3

The chapter establishes the computational limits of **Autonomous Recurrent Neural Networks (RNNs)** to motivate a more robust architecture called the **Associative Turing Machine (ATM)**.

Here is a structured breakdown of the arguments presented in the chapter.

1. The Core Argument

The text argues that while autonomous RNNs are powerful, they are **not** true Turing Machines because they rely on fixed-dimensional state vectors. They can simulate a Turing Machine for a **fixed** amount of time, but they fail at unbounded computation due to precision limitations.

2. Autonomous RNNs as Finite-State Machines

The text defines an autonomous RNN as a system where the state evolves without external input:

$$h_{t+1} = \phi(W h_t + b) \tag{I.5}$$

- **Proposition:** The text proves that an RNN can simulate any deterministic **Finite-State Machine (FSM)**.
- **Mechanism:** By encoding every possible state of the FSM as a specific vector (e.g., a standard basis vector) in $\mathbb{R}^d$, the RNN's weights W and bias b can be tuned to map one state vector to the next, effectively acting as a "lookup table" implemented via matrix multiplication and nonlinearity.

3. The Turing Machine Simulation (with a Catch)

Can an RNN simulate a Turing Machine (TM)? The text says **yes, but only for a finite time** T.

- **The Logic:** If a TM runs for only T steps, it can only visit a limited number of tape cells. Therefore, the number of possible configurations (state + tape contents + head position) is finite.
- **The Result:** Since the set of configurations is finite, the "Time-Bounded TM" is effectively just a very large Finite-State Machine. Therefore, an RNN can simulate it using the logic from point #2.

4. Why RNNs fail at Unbounded Computation

The text clarifies why this simulation collapses if $T \to \infty$ (unbounded time):

- **The Packing Problem:** A standard TM has an infinite tape. An RNN has a fixed-size state vector $h_t \in \mathbb{R}^d$. To represent an infinite tape in a fixed vector, you would need infinite precision (fractal encoding).
- **The Noise Problem:** In physical realizations or numerical simulations, noise exists. The text notes that unless the system is contracting, errors accumulate:

  $$\|h_t - \tilde{h}_t\| \leq L^t \|h_0 - \tilde{h}_0\| \tag{I.6}$$

  Exponentially growing precision is required to maintain the state distinction over long times, which is impossible.

5. The Solution: Associative Turing Machines (ATMs)

The limitations of RNNs motivate the ATM. The ATM separates computation from memory storage to ensure robustness.

I.6 Analogy: Transformers as Turing Machines

The Transformer architecture can be mapped to the Turing Machine (TM) framework by decoupling its two primary sub-layers: the Attention mechanism and the Multi-Layer Perceptron (MLP).

I.6.1 1. Attention as the Tape (Memory)

In a Turing Machine, the tape is the external substrate where information is stored and retrieved.

- **Storage:** The "tape" is represented by the sequence of previous token embeddings (specifically the Key and Value vectors). As the sequence length increases, the available memory grows.
- **Retrieval:** The **Attention mechanism** acts as the *Read Head*. Unlike an RNN, which must compress all history into a single vector, Attention allows the model to "look back" and perform content-based addressing (via Query-Key dot products) to retrieve specific information from any point on the tape.

I.6.2 2. MLP as the State (Controller)

The "State" of a Turing Machine is the finite set of rules or logic that processes the symbol currently being read.

- **Processing:** The **MLP (Feed-Forward Layer)** acts as the *Finite-State Controller*. It is a fixed-size network that does not see the other tokens directly; instead, it processes the integrated information delivered to it by the attention head.
- **Logic:** The weights of the MLP represent the learned "program" of the model. It takes the retrieved data, applies a non-linear transformation (the state transition), and prepares the output for the next layer or the next token prediction.

I.6.3 Summary Mapping

Turing Machine Component	Transformer Equivalent	Functional Role
Tape / Memory	Sequence of KV Vectors	Stores the history of computation.
Read/Write Head	Attention Mechanism	Retrieves specific data from history.
State / Controller	MLP (Feed-Forward Layers)	Fixed logic that processes retrieved data.

Table I.1: Mapping of Turing Machine components to Transformer sub-layers.

Conclusion. While an autonomous RNN attempts to collapse the tape *into* its internal state vector (leading to the precision and packing problems discussed earlier), the Transformer separates storage (the KV cache) from logic (the weights). This separation allows the Transformer to function as a "Soft Turing Machine," where the computational "scratchpad" is the sequence itself.

I.7 Correspondences in the Associative Turing Machine (ATM)

The Associative Turing Machine (ATM) provides a more direct neural implementation of the Turing Machine (TM) by replacing the "implicit memory" of standard RNNs with "explicit associative memory."

I.7.1 1. Component Mapping

- **The Tape as Associative Data Memory:** Instead of a linear sequence, the Tape is implemented as an associative memory bank. Each entry stores a pair (k_i, v_i), where k_i represents a coordinate (address) on the tape and v_i represents the symbol stored at that coordinate.
- **Transition Rules as Associative Logic:** The TM's transition table (the "program") is itself stored in an associative memory. This memory maps the current *Control State* and *Read Symbol* to a triplet: $(\text{New State}, \text{Write Symbol}, \text{Direction})$.
- **The Controller as the Head:** The neural controller (a small, fixed RNN) acts as the logic engine. It emits a query vector to the Data Memory to "read," uses that result to query the Logic Memory, and then updates the Data Memory to "write."

I.7.2 2. Solving the Precision Problem

The ATM avoids the limitations of autonomous RNNs through two mechanisms:

1. **Decoupling:** By moving the tape contents out of the state vector h_t and into an external associative memory, the state vector only needs to be large enough to represent the *current step* of the algorithm, not the entire history.
2. **Addressing vs. Compression:** The ATM uses *similarity-based retrieval* (associative lookup) rather than state-space compression. Because it retrieves discrete symbols based on keys, it does not require the "fractal precision" that causes autonomous RNNs to fail as $T \to \infty$.

Summary. The ATM uses neural associative memory as a high-speed, addressable RAM, allowing the neural controller to maintain a fixed size regardless of the complexity of the data on the tape.

I.8 Conclusion.

Using a second RNN as memory would be analogous to "writing notes on another brain." The second brain would eventually forget or distort the message. To replicate a Turing Machine, the memory must be a "dumb," passive substrate (like a matrix or stack) that creates a stable external reference for the Controller.

Notation and Mathematical Conventions

This appendix provides a reference for the specific mathematical notation and conventions used throughout the text, particularly regarding compositional functions, statistical complexity measures, and group-theoretic invariance.

Computability and Compositional Sparsity

The following notation is used primarily in Chapters 4, 8, and 21 to bridge Turing computability with deep learning architectures.

$CS_k(s, L)$ **Compositionally Sparse Function Class.**
The set of functions f that can be computed by a directed acyclic graph (DAG) such that:

- The graph has total size (number of nodes) at most s.
- The depth (longest path from input to output) is at most L.
- Every internal node has a fan-in (arity) of at most k.

This class formally captures the structure of *efficiently computable* functions.

$\mathcal{G}_n$ **Discrete Grid Domain.**
Defined as $\mathcal{G}_n := \{0, 1, \dots, 2^n\}^d / 2^n$. It represents the domain of inputs discretized to n bits of precision, used to prove exact equivalence between Turing Machines and ReLU networks (Chapter 4).

DAG **Directed Acyclic Graph.**
The graph theoretical representation of a computation where nodes represent constituent functions and edges represent data flow.

d^* **Maximal Local Arity.**
Used in Chapter 21. It denotes the maximum fan-in required for the nodes in a compiled DAG representation of a finite-horizon physical system. Approximation rates scale with d^* rather than the ambient dimension d.

Complexity Measures

These symbols are used in Chapters 12 and 13 to quantify generalization and capacity control.

$\mathfrak{R}_S(\mathcal{F})$ **Empirical Rademacher Complexity.**
Measures the ability of a function class $\mathcal{F}$ to fit random noise on a fixed dataset $S = \{x_1, \dots, x_m\}$. Defined as:

$$\mathfrak{R}_S(\mathcal{F}) = \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(x_i) \right]$$

where $\sigma_i \in \{-1, +1\}$ are independent Rademacher variables.

$\mathcal{H}(\epsilon, \mathcal{F}, \|\cdot\|)$ **Metric Entropy.**
The logarithm of the covering number $\mathcal{N}(\epsilon, \mathcal{F}, \|\cdot\|)$. It represents the number of bits needed to describe an element of $\mathcal{F}$ to within precision ϵ.

CV_{loo} **Leave-One-Out Cross-Validation Stability.**
Defined in Appendix A. A measure of algorithmic stability where the expected error difference is bounded when one training point is removed. This book establishes it as necessary and sufficient for the consistency of ERM.

Group Theory and Invariance

The following notation appears in Chapters 6 and 22 regarding Genericity and Invariant Representations.

G **Compact Group.**
A group acting unitarily on the data space (e.g., rotations, translations). In the context of representation learning, we assume $G \subset O(d)$.

dg **Haar Measure.**
The unique translation-invariant probability measure on a compact group G. Used to define group averages (orbits).

$\mu_t(I)$ **Orbit Distribution (or CDF).**
The probability distribution of the inner products $\langle I, gt \rangle$ as g varies over G. In Chapter 22, this is often represented by its Cumulative Distribution Function (CDF), providing a signature that is invariant to the transformation of the input I by elements of G.

N_η **Group η-Net.**
A finite subset of the group G such that every element $g \in G$ is within distance η of some element in N_η. The size of this net, $|N_G(\eta)|$, enters the complexity bounds for invariant embeddings.

Appendix: Glossary of Core Concepts

This glossary collects key terms and concepts introduced throughout the book. Entries are intentionally concise and technical; references point to chapters where the concept is developed in detail.

A

Associative Memory
A memory system in which retrieval is performed by content rather than by address. Given a cue or query, the system returns stored items that are similar to the cue under a learned or fixed similarity measure. Classical examples include Hopfield networks and radial basis function (RBF) networks; modern examples include attention mechanisms in transformers. Associative memory is treated in this book as a *computational primitive*. *(See Chaps. 1, 3)*

Associative Turing Machine (ATM)
A formal model of computation introduced in this book, consisting of alternating *content-addressable reads* from an associative memory and *local update* operations on a machine state. ATMs are shown to be Turing-complete with polynomial overhead and provide a formal abstraction of transformer-style computation. *(See Chap. 3)*

B

Behavioral Repertoire
The set of input–output behaviors or policies a system can realize under prompting or control. Intelligence is characterized here not by peak performance but by the *width* of this repertoire and the system's ability to deliberately restrict or reshape it. *(See Chap. 30)*

C

Compositionality
The property that a function or behavior can be expressed as a composition of simpler functions, typically organized hierarchically. In learning systems, compositionality permits reuse of subfunctions and enables generalization across domains. *(See Chaps. 1, 4)*

Containment Principle
The hypothesis that if system A is more intelligent than system B, then the set of behaviors realizable by B is contained within the set realizable by A. Operationally, this implies that A can simulate B, but not vice versa. Used to define a directed hierarchy of intelligence. *(See Chap. 30)*

CV_{loo} (Leave-One-Out Cross Validation)
A stability metric where the algorithm is trained on all but one data point, and error is measured on the left-out point. In this text, specific forms of stability are shown to be sufficient for generalization, linking CV_{loo} directly to predictivity. *(See Chap. 7)*

D

Diligent Learner
A learner that improves performance by systematically accumulating competence through training, without requiring explicit search or exploration mechanisms. The concept originates in work by Shalev-Shwartz and Shashua and is used here to contrast diligence with creativity and exploration. *(See Chap. 30)*

E

Efficient Computability
The property that a function can be computed with resources polynomial in the input size under a standard computational model (e.g., Turing machines or Boolean circuits). In this book, efficient computability is shown to imply sparse compositional structure in the corresponding function representation. *(See Chap. 4)*

Effective Information Exponent
A quantity measuring how the effective information content of a learned representation scales with system size or depth, used to characterize compression and generalization in compositional architectures. *(See Chaps. 7, 25)*

Empirical Risk Minimization (ERM)
The standard learning principle of minimizing the average error over the training dataset. While successful in practice, this book argues that ERM alone is insufficient to explain generalization without invoking stability or genericity conditions. *(See Chap. 7)*

G

Genericity
A property of target functions or learning problems asserting that informative, low-order components (e.g., linear or low-degree terms) are present and stable. Genericity ensures that gradient-based optimization receives reliable signals and avoids pathological loss landscapes. *(See Chaps. 7, 11)*

H

HyperBF / Hyper-RBF
A generalization of classical radial basis functions in which the similarity metric, prototypes, or both are learned. In particular, replacing Euclidean distance with a learned Mahalanobis metric yields kernels that are closely related to attention mechanisms. *(See Chaps. 1, 3)*

L

Large Embedding Model (LEM)
A proposed class of models extending large language models by incorporating richer latent world representations prior to or alongside language. LEMs are intended to address limitations of purely linguistic training by grounding prediction in learned world models. *(See Chap. 29)*

Loss Landscape
The function mapping model parameters to training or generalization error. The geometry of the loss landscape—its critical points, basins, and flat directions— plays a central role in optimization and generalization. *(See Chap. 7)*

M

Manifold Hypothesis
The assumption that high-dimensional data (e.g., images, text, sensory inputs) lie near low-dimensional manifolds embedded in ambient space. In this book, the hypothesis is extended to *behavioral* and *semantic* manifolds learned by large models. *(See Chaps. 1, 30)*

R

Radial Basis Function (RBF) Network
A function approximator of the form

$$f(x) = \sum_i w_i \, \phi(\|x - \mu_i\|),$$

where ϕ is a radial kernel and μ_i are fixed centers. RBF networks implement smooth associative memory and serve as historical precursors to modern attention-based models. *(See Chap. 1)*

Random-Access Memory (RAM)
A memory model permitting constant-time access to any stored location by address. Attention mechanisms approximate RAM-like behavior by enabling content-based access to all positions in a sequence. *(See Chaps. 1, 3)*

S

Simulation Distance
A metric defined between two models A and B as the asymmetry in their ability to simulate each other's behavior. Used to define a directed ordering of intelligence independent of static benchmarks. *(See Chap. 30)*

Sparse Compositionality
The principle that meaningful functions are compositions of a small number of low-arity constituent functions arranged in a sparse directed acyclic graph. Sparse compositionality explains why deep architectures are both necessary and effective for learning real-world functions. *(See Chaps. 1, 4)*

Stability
A property of a learning algorithm wherein small changes to the training set result in small changes to the output hypothesis. This book emphasizes stability as a necessary and sufficient condition for generalization, often replacing classical uniform convergence bounds. *(See Chap. 7)*

Stepwise Learning
A learning framework where complexity is increased incrementally. In the context of diffusion and autoregression, this refers to learning the conditional probability distribution of the next step (or next bit) given previous steps, effectively breaking a complex joint distribution into a sequence of simpler learning problems. *(See Chap. 10)*

T

Transformer
A neural architecture based on stacked self-attention and feedforward blocks. Transformers implement large-scale associative memory with learned similarity metrics and are central to modern language and vision models. *(See Chaps. 1, 3)*

W

World Simulator
An internal model that predicts the evolution of latent states corresponding to the external world. Distinguished here from purely linguistic predictors, world simulators are proposed as a missing component for robust intelligence. *(See Chaps. 24, 29)*

References

[1] R. Abraham and J. Robbin. *Transversal Mappings and Flows*. Benjamin, 1967.

[2] S. Ainsworth, A. Gromov, and A. Bietti. "GIT: Geometric Information in Transformers". In: *arXiv preprint arXiv:2305.14396* (2023).

[3] Z. Allen-Zhu, Y. Li, and Y. Liang. "Learning and generalization in overparameterized neural networks, going beyond two layers". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[4] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio. *Magic Materials: a theory of deep hierarchical architectures for learning sensory representations*. Tech. rep. Massachusetts Institute of Technology, Center for Brains, Minds and Machines (CBMM), 2013.

[5] F. Anselmi, L. Rosasco, and T. Poggio. "On Invariance and Selectivity in Representation Learning". In: *Information and Inference: A Journal of the IMA* 5.2 (2016), pp. 134–158. DOI: 10.1093/imaiai/iaw009.

[6] S. M. Anstis. "A chart demonstrating variations in acuity with retinal position". In: *Vision Research* 14.7 (1974), pp. 589–592.

[7] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[8] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[9] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. "Stronger generalization bounds for deep nets via a compression approach". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 254–263.

[10] A. R. Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information Theory* 39.3 (1993), pp. 930–945.

[11] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. "Spectrally-normalized margin bounds for neural networks". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[12] P. L. Bartlett and S. Mendelson. "Rademacher and Gaussian complexities: Risk bounds and structural results". In: *Journal of Machine Learning Research* 3 (2002), pp. 463–482.

[13] J. Bates. *Computing 10,000x More Efficiently*. Tech. rep. Singular Computing Technical Whitepaper, 2010.

[14] J. Bates. "Processor with Approximate Arithmetic Units". 8,407,273. 2013.

[15] J. Bates and A. Bates. "Toward Lattice QCD on Billion Core Approximate Computers". In: *arXiv preprint arXiv:2010.15973* (2020).

[16] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. "Interaction networks for learning about objects, relations and physics". In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016.

[17] J. Baxter. "A model of inductive bias learning". In: *Journal of Artificial Intelligence Research* 12 (2000), pp. 149–198.

[18] Y. Bengio, A. Courville, and P. Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[19] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin. "Towards biologically plausible deep learning". In: *arXiv preprint arXiv:1502.04156* (2015).

[20] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48.

[21] N. Bottman, Y. Cooper, and A. Lerario. "How regularization affects the geometry of loss functions". In: *arXiv preprint arXiv:2307.15744* (2023).

[22] H. Bouma. "Interaction Effects in Parafoveal Letter Recognition". In: *Nature* 226.5241 (1970), pp. 177–178.

[23] O. Bousquet and A. Elisseeff. "Stability and Generalization". In: *Journal of Machine Learning Research* 2 (Mar. 2002), pp. 499–526.

[24] D. S. Broomhead and D. Lowe. "Multivariable functional interpolation and adaptive networks". In: *Complex Systems* 2.3 (1988), pp. 321–355.

[25] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1877–1901.

[26] C. Burns, P. Izmailov, J. H. Kirchner, B. Baker, L. Gao, L. Aschenbrenner, Y. Chen, A. Ecoffet, M. Joglekar, J. Leike, et al. *Weak-to-strong generalization: Eliciting strong capabilities with weak supervision*. Tech. rep. OpenAI, 2023.

[27] R. Caruana. "Multitask learning". In: *Machine learning* 28 (1997), pp. 41–75.

[28] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. "Entropy-SGD: Biasing Gradient Descent into Wide Valleys". In: *International Conference on Learning Representations*. 2017.

[29] H. Chen, J. Yuan, C. Fu, and X. Chen. "Lossy Image Compression with Conditional Diffusion Models". In: *arXiv preprint arXiv:2306.00000* (2023).

[30] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze. "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks". In: *Proceedings of the 43rd Annual International Symposium on Computer Architecture*. 2016, pp. 367–379.

[31] R. Child, S. Gray, A. Radford, and I. Sutskever. "Generating long sequences with sparse transformers". In: *arXiv preprint arXiv:1904.10509* (2019).

[32] M. Chistiakova, N. M. Bannon, M. Bazhenov, and M. Volgushev. "Heterosynaptic plasticity: multiple mechanisms and multiple roles". In: *The Neuroscientist* 20.5 (2014), pp. 483–498.

[33] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. "Rethinking Attention with Performers". In: *International Conference on Learning Representations*. 2021.

[34] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. “What Does BERT Look At? An Analysis of BERT’s Attention”. In: *arXiv preprint arXiv:1906.04341* (2019).

[35] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K. Stanley, and J. Clune. “Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents”. In: *Advances in neural information processing systems* 31 (2018).

[36] S. A. Cook. “The complexity of theorem-proving procedures”. In: *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, pp. 151–158.

[37] Y. Cooper. “The loss landscape of over-parameterized neural networks”. In: *arXiv preprint arXiv:1804.10200* (2018).

[38] A. Cowey and E. T. Rolls. “Human cortical magnification factor and its relation to visual acuity”. In: *Experimental Brain Research* 21.5 (1974), pp. 447–454.

[39] F. Crick. “The recent excitement about neural networks”. In: *Nature* 337.6203 (1989), pp. 129–132.

[40] Y. Dandi, L. Pesce, L. Zdeborová, and F. Krzakala. “The Computational Advantage of Depth in Learning High-Dimensional Hierarchical Targets”. In: *Advances in Neural Information Processing Systems*. Vol. 39. 2025.

[41] D. A. Danhofer, D. D’Ascenzo, R. Dubach, and T. A. Poggio. “Position: A Theory of Deep Learning Must Include Compositional Sparsity”. In: *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*. Also available as arXiv:2507.02550. 2025.

[42] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. “FlashAttention: Fast and memory-efficient exact attention with IO-awareness”. In: *Advances in Neural Information Processing Systems*. 2022.

[43] L. N. Darlow, S. W. Luke, and B. Amos. “Cluster-based attention”. In: *arXiv preprint arXiv:2007.03338* (2020).

[44] T. Dettmers. “8-bit approximations for parallelism in deep learning”. In: *arXiv preprint arXiv:1511.04561* (2015).

[45] R. A. DeVore, R. Howard, and C. Micchelli. “Optimal nonlinear approximation”. In: *Manuscripta Mathematica* 63.4 (1989), pp. 469–478.

[46] R. A. DeVore and G. G. Lorentz. *Constructive Approximation*. Vol. 303. Springer Science & Business Media, 1993.

[47] J. J. DiCarlo, D. Zoccolan, and N. C. Rust. “How does the brain solve visual object recognition?” In: *Neuron* 73.3 (2012), pp. 415–434.

[48] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. “Optimal rates for zero-order convex optimization: The power of two function evaluations”. In: *IEEE Transactions on Information Theory* 61.5 (2015), pp. 2788–2806.

[49] R. M. Dudley. “The sizes of compact subsets of Hilbert space and continuity of Gaussian processes”. In: *Journal of Functional Analysis* 1.3 (1967), pp. 290–330.

[50] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, et al. “A mathematical framework for transformer circuits”. In: *Transformer Circuits Thread* (2021).

[51] W. Fedus, B. Zoph, and N. Shazeer. “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity”. In: *Journal of Machine Learning Research* 23.120 (2022), pp. 1–39.

[52] I. R. Fiete, Y. Burak, and T. Brookings. "What grid cells convey about rat location". In: *Journal of Neuroscience* 28.27 (2008), pp. 6858–6871.

[53] D. J. Foster and M. A. Wilson. "Reverse replay of behavioural sequences in hippocampal place cells during the awake state". In: *Nature* 440.7084 (2006), pp. 680–683.

[54] J. Frankle and M. Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. 2019.

[55] J. Freeman and E. P. Simoncelli. "Metamers of the ventral stream". In: *Nature Neuroscience* 14.9 (2011), pp. 1195–1201.

[56] K. Friston. "The free-energy principle: a unified brain theory?" In: *Nature Reviews Neuroscience* 11.2 (2010), pp. 127–138.

[57] K. Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological Cybernetics* 36.4 (1980), pp. 193–202.

[58] T. Galanti, L. Galanti, and I. Ben-Shaul. "Comparative Generalization Bounds for Deep Neural Networks". In: *Transactions on Machine Learning Research* (2023).

[59] T. Galanti, M. Xu, L. Galanti, and T. Poggio. "Norm-Based Generalization Bounds for Sparse Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023.

[60] T. Galanti, M. Xu, L. Galanti, and T. Poggio. "Norm-based generalization bounds for sparse neural networks". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 42482–42501.

[61] R. Gattass, C. G. Gross, and J. Sandell. "Visual topography of V2 in the macaque". In: *Journal of Comparative Neurology* 201.4 (1981), pp. 519–539.

[62] N. Golowich, A. Rakhlin, and O. Shamir. "Size-independent sample complexity of neural networks". In: *Journal of Machine Learning Research* (2019).

[63] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[64] A. Graves, G. Wayne, and I. Danihelka. "Neural turing machines". In: *arXiv preprint arXiv:1410.5401* (2014).

[65] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. "Hybrid computing using a neural network with dynamic external memory". In: *Nature* 538.7626 (2016), pp. 471–476.

[66] A. Gu and T. Dao. "Mamba: Linear-time sequence modeling with selective state spaces". In: *arXiv preprint arXiv:2312.00752* (2023).

[67] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning". In: *arXiv preprint arXiv:2501.12948* (2025).

[68] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A distribution-free theory of nonparametric regression*. Vol. 1. Springer, 2002.

[69] D. Ha and J. Schmidhuber. "World models". In: *arXiv preprint arXiv:1803.10122* (2018).

[70] S. Han, H. Mao, and W. J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *International Conference on Learning Representations*. 2016.

[71] N. Hansen and A. Ostermeier. “Completely derandomized self-adaptation in evolution strategies”. In: *Evolutionary Computation* 9.2 (2001), pp. 159–195.

[72] L. Hardesty. *The Surprising Usefulness of Sloppy Arithmetic*. MIT News. Jan. 2011.

[73] J. Håstad. “Almost optimal lower bounds for small depth circuits”. In: *Proceedings of the 18th ACM Symposium on Theory of Computing*. 1986, pp. 6–20.

[74] K. He, X. Zhang, S. Ren, and J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *arXiv preprint arXiv:1502.01852* (2015).

[75] D. Hendrycks and K. Gimpel. “Gaussian error linear units (GELUs)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[76] G. E. Hinton, S. Osindero, and Y.-W. Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[77] J. Ho, A. Jain, and P. Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[78] J. Ho, A. Jain, and P. Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851.

[79] S. Hochreiter and J. Schmidhuber. “Flat minima”. In: *Neural Computation* 9.1 (1997), pp. 1–42.

[80] A. E. Hoerl and R. W. Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (1970), pp. 55–67.

[81] E. Hoogeboom, T. Salmona, J. Peters, and M. Welling. “Discrete Diffusion Models for Extreme Image Compression”. In: *Advances in Neural Information Processing Systems*. 2023.

[82] J. J. Hopfield. “Neural networks and physical systems with emergent collective computational abilities”. In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[83] J. Hossain, A. Z. Faridee, N. Roy, et al. “VIVAR: Learning View-Invariant Embedding for Video Action Recognition”. In: *ICVIP*. 2025.

[84] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations*. 2022.

[85] D. H. Hubel and T. N. Wiesel. “Uniformity of monkey striate cortex: a parallel relationship between field size, scatter, and magnification factor”. In: *Journal of Comparative Neurology* 158.3 (1974), pp. 295–305.

[86] S. Hutter. “The scaling hypothesis: how the economy of scale drives deep learning progress”. In: *Distill* (2021).

[87] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[88] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, et al. “Population based training of neural networks”. In: *arXiv preprint arXiv:1711.09846* (2017).

[89] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. “In-datacenter performance analysis of a tensor processing unit”. In: *Proceedings of the 44th Annual International Symposium on Computer Architecture*. 2017, pp. 1–12.

[90] P. Kanerva. *Sparse distributed memory*. MIT press, 1988.

[91] P. Kanerva. "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors". In: *Cognitive computation* 1 (2009), pp. 139–159.

[92] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020).

[93] N. Kashtan and U. Alon. "Spontaneous evolution of modularity and network motifs". In: *Proceedings of the National Academy of Sciences* 102.39 (2005), pp. 13773–13778.

[94] M. Kearns. "Efficient noise-tolerant learning from statistical queries". In: *Journal of the ACM* 45.6 (1998), pp. 983–1006.

[95] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *International Conference on Learning Representations*. 2017.

[96] I. Khemakhem, D. Kingma, R. Monti, and A. Hyvarinen. "Variational autoencoders and nonlinear ICA: A unifying framework". In: *International Conference on Artificial Intelligence and Statistics* (2020), pp. 2207–2217.

[97] N. Kitaev, Ł. Kaiser, and A. Levskaya. "Reformer: The efficient transformer". In: *International Conference on Learning Representations*. 2020.

[98] T. Kohonen. "Correlation matrix memories". In: *IEEE Transactions on Computers* 100.4 (1972), pp. 353–359.

[99] A. N. Kolmogorov. "On tables of random numbers". In: *Sankhyā: The Indian Journal of Statistics, Series A* 25.4 (1963), pp. 369–376.

[100] V. Koltchinskii. "Rademacher penalties and structural risk minimization". In: *IEEE Transactions on Information Theory* 47.5 (2001), pp. 1902–1914.

[101] R. Krizhanovsky and V. Krizhanovsky. "Mesh and torus computer architectures for data-parallel computations". In: *Parallel Computing* 74 (2018), pp. 1–13.

[102] A. Krogh and J. A. Hertz. "A simple weight decay can improve generalization". In: *Advances in Neural Information Processing Systems*. Vol. 4. 1991.

[103] H.-T. Kung. "Why systolic architectures?" In: *Computer* 15.1 (1982), pp. 37–46.

[104] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. "Convergence properties of the Nelder–Mead simplex method in low dimensions". In: *SIAM Journal on optimization* 9.1 (1998), pp. 112–147.

[105] B. S. Landman and R. L. Russo. "On a pin versus block relationship for partitions of logic graphs". In: *IEEE Transactions on Computers* 100.12 (1971), pp. 1469–1479.

[106] L. Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer Science & Business Media, 2012.

[107] Y. LeCun. "A path towards autonomous machine intelligence version 0.9.2, 2022-06-27". In: *Open Review* 62 (2022).

[108] S. Legg and M. Hutter. "Universal intelligence: A definition of machine intelligence". In: *Minds and Machines* 17.4 (2007), pp. 391–444.

[109] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. "Visualizing the Loss Landscape of Neural Nets". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[110] Q. Liao, L. Ziyin, Y. Gan, B. Cheung, M. Harnett, and T. Poggio. "Self-assembly of a biologically plausible learning circuit". In: *arXiv preprint arXiv:2412.20018* (2024).

[111] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. "Random synaptic feedback weights support error backpropagation for deep learning". In: *Nature Communications* 7.1 (2016), pp. 1–10.

[112] H. W. Lin and M. Tegmark. "Why does deep and cheap learning work so well?" In: *Journal of Statistical Physics* 184.1 (2021), pp. 1–45.

[113] T. Lindeberg. *Scale-space theory in computer vision*. Vol. 256. Springer Science & Business Media, 1994.

[114] Y. Liu, K. Chen, Z. Chen, and S. Chang. "DMVC: Diffusion Models for Vector-Quantized Image Compression". In: *arXiv preprint arXiv:2306.00000* (2023).

[115] N. K. Logothetis, J. Pauls, H. H. Bülthoff, and T. Poggio. "View-dependent object recognition by monkeys". In: *Current Biology* 4.5 (1994), pp. 401–414.

[116] N. K. Logothetis, J. Pauls, and T. Poggio. "Shape representation in the inferior temporal cortex of monkeys". In: *Current Biology* 5.5 (1995), pp. 552–563.

[117] E. Malach. "Auto-regressive next-token predictors are universal learners". In: *arXiv preprint arXiv:2309.06979* (2023).

[118] E. Malach, D. Rohatgi, G. Yehudai, and S. Shalev-Shwartz. "Autoregressive next-token predictors are universal learners". In: *arXiv preprint arXiv:2309.06979* (2023).

[119] E. Malach, G. Yehudai, S. Shalev-Shwartz, and O. Shamir. "Proving the Lottery Ticket Hypothesis: Pruning is All You Need". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6682–6691.

[120] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora. *Fine-Tuning Language Models with Just Forward Passes*. 2024. arXiv: 2305.17333 [cs.LG]. URL: https://arxiv.org/abs/2305.17333.

[121] S. Mandt, M. D. Hoffman, and D. M. Blei. "Stochastic gradient descent as approximate Bayesian inference". In: *Journal of Machine Learning Research* 18.1 (2017), pp. 4873–4907.

[122] D. Marr. "Simple memory: a theory for archicortex". In: *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 262.841 (1971), pp. 23–81.

[123] D. Marr, T. Poggio, and E. Hildreth. "Smallest channel in early human vision". In: *Journal of the Optical Society of America* 70.7 (1980), pp. 868–870.

[124] C. H. Martin and M. W. Mahoney. "Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning". In: *Journal of Machine Learning Research* 22.165 (2021), pp. 1–73.

[125] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly. "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory". In: *Psychological Review* 102.3 (1995), p. 419.

[126] H. Mhaskar and T. Poggio. "Deep vs. shallow networks: An approximation theory perspective". In: *Analysis and Applications* (2016), pp. 829–848.

[127] H. N. Mhaskar. "Neural networks for optimal approximation of smooth and analytic functions". In: *Neural Computation* 8.1 (1996), pp. 164–177.

[128] H. N. Mhaskar and C. A. Micchelli. "Approximation by superposition of sigmoidal and radial basis functions". In: *Advances in Applied Mathematics* 13.3 (1992), pp. 350–373.

[129] H. N. Mhaskar and T. Poggio. "Deep vs. shallow networks: An approximation theory perspective". In: *Analysis and Applications* 14.06 (2016), pp. 829–848.

[130] P. Michel, O. Levy, and G. Neubig. "Are Sixteen Heads Really Better than One?" In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[131] P. Micikevicius, S. Narayanaswami, M. Abdelfattah, et al. "Mixed Precision Training". In: *International Conference on Learning Representations*. 2018.

[132] S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. "Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization". In: *Advances in Computational Mathematics* 25.1 (2006), pp. 161–193.

[133] Y. Nesterov and V. Spokoiny. "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17 (2017), pp. 527–566.

[134] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. "Exploring Generalization in Deep Learning". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[135] B. Neyshabur, S. Bhojanapalli, and N. Srebro. "A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks". In: *arXiv preprint arXiv:1707.09564* (2017).

[136] B. Neyshabur, R. Tomioka, and N. Srebro. "Norm-based capacity control in neural networks". In: *Conference on Learning Theory*. 2015, pp. 1376–1401.

[137] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. "Intrinsic motivation systems for autonomous mental development". In: *IEEE transactions on evolutionary computation* 11.2 (2007), pp. 265–286.

[138] G. Palm. "On associative memory". In: *Biological Cybernetics* 36.1 (1980), pp. 19–31.

[139] V. Papyan, X. Y. Han, and D. L. Donoho. "Prevalence of neural collapse during the terminal phase of deep learning training". In: *Proceedings of the National Academy of Sciences* 117.40 (2020), pp. 24652–24663.

[140] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. "Curiosity-driven exploration by self-supervised prediction". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2778–2787.

[141] J. Pennington, S. Schoenholz, and S. Ganguli. "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[142] J. Pestaña, H. König, and A. Lucchi. "Generative Compression Using Diffusion Models". In: *ICLR Workshop on Deep Generative Models*. 2023.

[143] T. A. Plate. "Holographic reduced representations". In: *IEEE Transactions on Neural Networks* 6.3 (1995), pp. 623–641.

[144] T. Poggio. "On optimal nonlinear associative recall". In: *Biological Cybernetics* 19.4 (1975), pp. 201–209.

[145] T. Poggio et al. "Self-Assembly of a Biologically Plausible Learning Circuit". In: *CBMM Memo* 152 (2024).

[146] T. Poggio. *On Efficiently Computable Functions, Deep Networks, and Sparse Compositionality*. Preprint. CBMM Memo, 2025.

[147] T. Poggio, A. Banburski, and Q. Liao. "The theory of deep learning: an overview". In: *arXiv preprint arXiv:1706.08845* (2017).

[148] T. Poggio and collaborators. *On Efficiently Computable Functions, Deep Networks, and Sparse Compositionality*. Tech. rep. 072. Center for Brains, Minds and Machines (CBMM), 2025.

[149] T. Poggio and S. Edelman. "A network that learns to recognize three-dimensional objects". In: *Nature* 343.6255 (1990), pp. 263–266.

[150] T. Poggio and M. Fraser. "Compositional sparsity of learnable functions". In: *Bulletin of the American Mathematical Society* 61.3 (2024), pp. 438–456.

[151] T. Poggio and Y. Gan. "A Homogeneous Transformer Architecture". In: *CBMM Memo* 143 (2023).

[152] T. Poggio and F. Girosi. "Networks for approximation and learning". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1481–1497.

[153] T. Poggio and Q. Liao. "Theory II: Deep learning and optimization". In: *Bulletin of the Polish Academy of Sciences: Technical Sciences* 66.6 (2018). Also available as CBMM Memo No. 066, pp. 775–787.

[154] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review". In: *International Journal of Automation and Computing* 14.5 (2017), pp. 503–519.

[155] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 376.2140 (2018), p. 20170246.

[156] T. Poggio, J. Mutch, and L. Isik. "Computational role of eccentricity dependent cortical magnification". In: *arXiv preprint arXiv:1406.1770* (2014). CBMM Memo No. 017.

[157] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. "General conditions for predictivity in learning theory". In: *Nature* 428.6981 (2004), pp. 419–422.

[158] T. A. Poggio and F. Anselmi. *Visual Cortex and Deep Networks: Learning Invariant Representations*. Cambridge, MA: MIT Press, 2016. ISBN: 9780262034722.

[159] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. "Exponential expressivity in deep neural networks through transient chaos". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3360–3368.

[160] X. Qiu, Y. Gan, C. F. Hayes, Q. Liang, E. Meyerson, B. Hodjat, and R. Miikkulainen. "Evolution strategies at scale: Llm fine-tuning beyond reinforcement learning". In: *arXiv preprint arXiv:2509.24372* (2025).

[161] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[162] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, et al. "Hopfield Networks is All You Need". In: *International Conference on Learning Representations*. 2021.

[163] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. "Regularized evolution for image classifier architecture search". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 4780–4789.

[164] I. Rechenberg. "Evolution strategy: Nature's way of optimization". In: *Optimization: Methods and Applications, Possibilities and Limitations: Proceedings of an International Seminar Organized by Deutsche Forschungsanstalt für Luft-und Raumfahrt (DLR), Bonn, June 1989*. Springer. 1989, pp. 106–126.

[165] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 2019, pp. 3982–3992.

[166] M. Riesenhuber and T. Poggio. "Hierarchical models of object recognition in cortex". In: *Nature Neuroscience* 2.11 (1999), pp. 1019–1025.

[167] G. Roeder, L. Metz, and D. P. Kingma. "On Linear Identifiability of Learned Representations". In: *International Conference on Machine Learning*. 2021, pp. 9030–9039.

[168] A. Roy, M. Saffar, A. Vaswani, and D. Grangier. "Efficient content-based sparse attention with routing transformers". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 53–67.

[169] T. Salimans and D. P. Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *arXiv preprint arXiv:1602.07868* (2016).

[170] P. G. Schyns and F. Gosselin. "Diagnostic Use of Scale Information for Componential and Holistic Recognition". In: *Perception of Faces, Objects, and Scenes*. Ed. by M. A. Peterson and G. Rhodes. Oxford University Press, 2003, pp. 120–145.

[171] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[172] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[173] S. Shalev-Shwartz and A. Shashua. *From Reasoning to Super-Intelligence: A Search-Theoretic Perspective*. Tech. rep. July 2025. DOI: 10.48550/arXiv.2507.15865.

[174] M. Shanahan, K. McDonell, and L. Reynolds. "Role play with large language models". In: *Nature* 623.7987 (2023), pp. 493–498.

[175] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer". In: *arXiv preprint arXiv:1701.06538* (2017).

[176] H. T. Siegelmann and E. D. Sontag. "On the computational power of neural networks". In: *Journal of computer and system sciences* 50.1 (1995), pp. 132–150.

[177] M. Sipser. *Introduction to the Theory of Computation*. 2nd. Thomson Course Technology, 2006.

[178] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 2256–2265.

[179] D. A. Spielman and S.-H. Teng. "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time". In: *Journal of the ACM* 51.3 (2004), pp. 385–463.

[180] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[181] C. Tao, T. Shen, S. Gao, et al. "LLMs are Also Effective Embedding Models: An In-depth Overview". In: *arXiv preprint arXiv:2412.12591v2* (2025).

[182] M. Telgarsky. "Benefits of depth in neural networks". In: *Conference on Learning Theory*. PMLR. 2016, pp. 1517–1539.

[183] R. Thom. "Les singularités des applications différentiables". In: *Annales de l'institut Fourier* 6 (1956), pp. 43–87.

[184] S. Thorpe, D. Fize, and C. Marlot. "Speed of processing in the human visual system". In: *Nature* 381.6582 (1996), pp. 520–522.

[185] G. Toderici et al. "High Fidelity Generative Image Compression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1–10.

[186] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. "Transformer dissection: An unified understanding for self-attention and convolution". In: *arXiv preprint arXiv:1908.00173* (2019).

[187] M. Tschannen, A. Gritsenko, X. Zhai, et al. "SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding". In: *arXiv preprint arXiv:2502.14786* (2025).

[188] A. M. Turing. "Computing machinery and intelligence". In: *Mind* 59.236 (1950), pp. 433–460.

[189] A. M. Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society* 2.42 (1936), pp. 230–265.

[190] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[191] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[192] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[193] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[194] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018.

[195] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 5797–5808.

[196] M. Wang and W. E. "On the Expressive Power of Mixture-of-Experts for Structured Complex Tasks". In: *arXiv preprint arXiv:2505.24205* (2025).

[197] S. Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, 2009.

[198] M. Welling and Y. W. Teh. "Bayesian Learning via Stochastic Gradient Langevin Dynamics". In: *International Conference on Machine Learning*. 2011, pp. 681–688.

[199] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. "Natural evolution strategies". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 949–980.

[200] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. "Non-holographic associative memory". In: *Nature* 222.5197 (1969), pp. 960–962.

[201] A. P. Witkin. "Scale-space filtering". In: *International Joint Conference on Artificial Intelligence*. Vol. 83. 1983, pp. 1019–1022.

[202] S. M. Wood et al. "A Newborn Embodied Turing Test for View-Invariant Object Recognition". In: *International Conference on Learning Representations (ICLR)*. 2025.

[203] M. Xu, A. Rangamani, Q. Liao, T. Galanti, and T. Poggio. "Dynamics in Deep Classifiers Trained with the Square Loss: Normalization, Low Rank, Neural Collapse, and Generalization Bounds". In: *Research* 6 (2023), p. 0024.

[204] Z. Xu, Y. Zhang, and T. Poggio. "The Impact of Initialization on Generalization in Overparameterized Networks". In: *Journal of Machine Learning Research* 25 (2024), pp. 1–40.

[205] L. Xue, M. Gao, C. Xing, et al. "ULIP: Learning a Unified Representation of Language, Images, and Point Clouds for 3D Understanding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 1179–1189.

[206] D. Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *Neural Networks* 94 (2017), pp. 103–114.

[207] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization". In: *International Conference on Learning Representations*. 2017.

[208] H. Zhang, P. Svirin, and S. Venkataramani. "Block floating point for deep neural networks". In: *arXiv preprint arXiv:1909.13271* (2019).

[209] W. Zhou and G. K. Dziugaite. "Non-vacuous generalization bounds at the deepest level". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.