

9.520/6.7910: Statistical Learning Theory and Applications

- Class: **Tue, Thu 11:00 am - 12:30 pm**, 46-3002
- Office Hours: **Tue, Thu 1:30pm - 3:00pm, 46-5155D**
- Web: <https://poggio-lab.mit.edu/9-520/>
- Contact: 9.520@mit.edu, pierb@mit.edu
- 9.520/6.7910 will use Canvas: <https://canvas.mit.edu/courses/34219>
- Also check Canvas announcements for updates
- This year's course will be in-person

Outline

- Course description/logistic
- Why we need a science of intelligence
- Why we need theory in ML
- From approximation and generalization to optimization

Faces

- Instructors:

Faces

- Instructors:
 - Lorenzo Rosasco



Faces

- Instructors:
 - Lorenzo Rosasco
 - Tomaso Poggio



Faces

- Instructors:
 - Lorenzo Rosasco
 - Tomaso Poggio
 - Pierfrancesco Beneventano



Faces

- Instructors:
 - Lorenzo Rosasco
 - Tomaso Poggio
 - Pierfrancesco Beneventano
 - Ziyin Liu



Faces

- Instructors:
 - Lorenzo Rosasco
 - Tomaso Poggio
 - Pierfrancesco Beneventano
 - Ziyin Liu
 - Qianli Liao



Rules of the game

9.520: the 3 parts of the course

The first part is about:

- classical regularization and regularized least squares + kernel machines, SVM
- logistic regression, square and exponential loss
- large margin (minimum norm)
- stochastic gradient methods,
- overparametrization, implicit regularization
- approximation/estimation errors.

The second part is about deep networks and, particular, about:

- approximation theory -- why deep networks avoid the curse of dimensionality and are a universal parametric approximators
- optimization theory -- how weights and activation evolve in time and across layers during training with SGD
- learning theory -- how generalization in deep networks can be explained in terms of the complexity control implicit in regularized (or unregularized) SGD and in terms of the details of the sparse compositional architecture itself.

Syllabus at a glance

Class	Day	Date	Title	Instructor
Class 1	Th	9/4/2025	Course Outline --- SLT framework	TP + PB
Class 2	Tue	9/9/2025	Statistical Learning Theory	LR
Class 3	Th	9/11/2025	Least Squares and Overparametrization	LR
Class 4	Tue	9/16/2025	Logistic Regression and SGD	LR
Class 5	Th	9/18/2025	Implicit Regularization	LR
Class 6	Tue	9/23/2025	Neural Networks and Deep Learning	LR
Class 7	Th	9/25/2025	Random Features, NTK & RKHS	LR
Class 8	Tue	9/30/2025	Infinite Width Neural Networks & RKBS	LR
Class 9	Th	10/2/2025	Statistical Learning Theory (Part 1)	LR
Class 10	Tue	10/7/2025	Statistical Learning Theory (Part 2)	LR
Class 11	Th	10/9/2025	Sequential Prediction and Learning Dynamical Systems	LR
Class 12	Tue	10/14/2025	Classic to Modern --- What Changes	PB/TP
Class 13	Th	10/16/2025	Universal Approximation	TP
Class 14	Tue	10/21/2025	Compositional Sparsity	TP
Class 15	Th	10/23/2025	On Generalization Theory --- Rademacher, Convolutions, Sparsity	TP
Class 16	Tue	10/28/2025	Opt 1 --- The algorithms and how they behave on quadratics (speed, stability, Robbins-Monro)	PB
Class 17	Th	10/30/2025	Opt 2 --- What we know on the landscape, Edge of Stability	PB
Class 18	Tue	11/4/2025	Opt 3 --- The Effect of Mini-Batching, Implicit vs Explicit Regularization	PB
Class 19	Th	11/6/2025	Opt 4 --- Connecting Optimization and the Features Learned	PB
	Tue	11/11/2025		
Class 20	Th	11/13/2025		
Class 21	Tue	11/18/2025	Transformers	?
Class 22	Th	11/20/2025	Diffusions	PH
Class 23	Tue	11/25/2025		
	Th	11/27/2025		
Class 24	Tue	12/2/2025	Presentations	
Class 25	Th	12/4/2025	Presentations	
Class 26	Tue	12/9/2025	Panel: Directions for Deep Learning Theory	

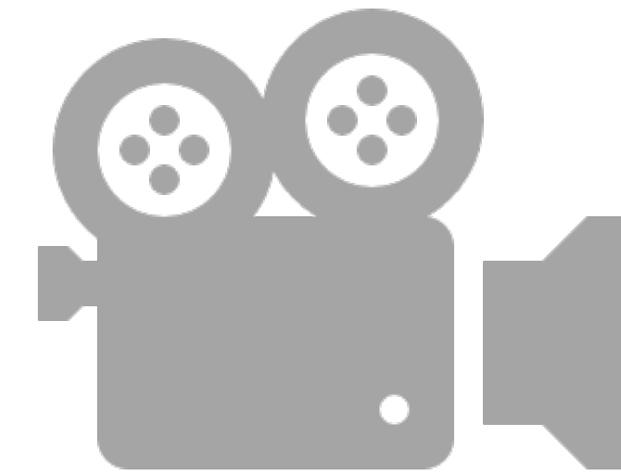
9.520/6.7910: Statistical Learning Theory and Applications

- Class: **Tue, Thu 11:00 am - 12:30 pm**, 46-3002
- Office Hours: **Tue, Thu 1:30pm - 3:00pm, 46-5155D**
- Web: <https://poggio-lab.mit.edu/9-520/>
- Contact: 9.520@mit.edu, pierb@mit.edu
- 9.520/6.7910 will use Canvas: <https://canvas.mit.edu/courses/34219>
- Also check Canvas announcements for updates
- This year's course will be in-person

Material



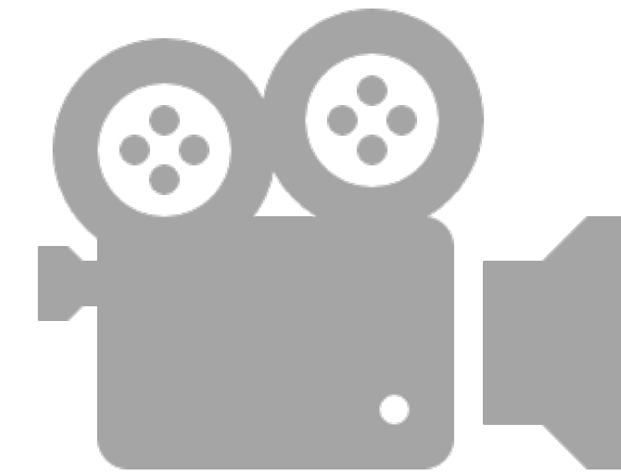
Slides— will be posted (for most lectures) on the website and Canvas



Videos— Recordings of lectures will be made available on Canvas

Video Recordings

- Most lectures will be recorded
- Some will be uploaded online publicly (e.g., the last one will be a panel with leading researchers on directions for the field).
- If you do not want to appear or be recorded let us know in advance and seat in the back.



Videos— Recordings of lectures will be made available on Canvas

Grading policies

Problem sets (0.45)

- 3 problem sets (0.15 each)
- 3 - 4 questions (exercises and/or MATLAB)
- Due in 10 days (out on a Thursday, and due on Sunday)
- Late policy on next slide
- typeset in LaTeX (template will be provided)
- Online submission on Canvas by due date

Project (0.45)

No problem sets

Required!

- Discussions during class - ask questions!
- Discussions on Canvas

Change from the Past

- This past July, a UCLA-led team (Huang & Yang) demonstrated that the publicly available Gemini 2.5 Pro could attain a gold-level score at IMO with structured prompting and verification—it solved 5 of the 6 problems.
- We believe LLMs are great tools for students and researchers:
We expect and encourage you to make use of them.
- Thus, we will devote more time and grade to the project, and
- We will grade a few initial deliverables of the projects in October.

Change from the Past

- This class is aimed at PhD students. We understand your goal is to learn how to produce research as fast as possible and you do know already *how to study* material when you want to.
- Our role will be to **provide resources to** and **actively help students to**:
 - learn this subject
 - produce research in this field.
- We want to guide you towards a research project that you'll be able to submit at ICML or COLT at the end of the winter break. In your interest.
- We put an effort to find projects which are *significant* and *well-defined* to help you do so — *out in two weeks*.

Grading policies

Project (90%)

- Literature review + Implications for 3 problems (15%)
- First Checkpoint (10%)
- Presentation (25%)
- Final Report (Paper) (40%)

Class Participation (10%)

- *Attending class lectures is required!*
- Discussions during class - ask questions!
- Show up to OH in November to discuss your project.

Deadlines & Penalties

Deadlines

- Deadlines are on Friday at 11:59pm Local time.
- You are allowed to 1 delay of max 3 days (end of Monday) on the deliverables/presentations which are *not* in December (not the presentation or the final report).

Penalties

- Any other (or longer) delay before December: -3 points.
No delays allowed in Dec.
- -15 points, if the project overlaps with current or past research or coursework of a member.
Do come discuss with us about it.

Project: Timeline & Tasks

Timeline

- **Sept 26:** Fill a form indicating 3 perspective projects and your group (max 2 people).
- **Oct 10:** 2-3 pages of Literature Review, 1 page of implications for each of the 3 projects.
- **Oct 17:** Choose the project of the 3 that you will work on.
- **Oct 31:** Hand in initial results.
- **Nov 1-14:** Come to an office hour.
- **Dec 2-4:** Presentations.
- **Dec 10:** Hand in final report.

Tasks in detail

Timeline and Grading

- **Sept 26:** Fill a form indicating 3 perspective projects and your group (max 2 people).
 - You will just have to fill a google form.
 - Pick 3 of our projects or 2 projects from ours + propose one.
 - If you propose one upload a pdf of 0.5-1 page describing it.

Tasks in detail

Timeline and Grading

- **Oct 10:** Literature Review and Implications: First of 2 interim milestone.
 - Hand in 3-4 pages for each of the projects:
 - 2-3 pages for a literature review: they should contain (but not limited to, if applicable) motivations, previous attempts, related empirical and mathematical work, why it is difficult, etc.
 - 1 page of implications: at least 3 detailed consequences for practice and theory.
 - We advise to start with some deep researches and to move on from there reading the articles suggested.
 - The goal is to make sure you have an understanding of what projects you are getting into.
- 15 points (5 each).

Tasks in detail

Timeline and Grading

- **Oct 17:** Choose what project you'll work on.
 - You need to discuss with us if you want to pick the one you proposed: We want to make sure it does not overlap extensively with other previous or current projects and that it has significance to an SLT for Neural Networks.
 - No grade of course, just filling a form.

Tasks in detail

Timeline and Grading

- **Oct 31:** Hand in initial results.
 - The second out of 2 interim milestone.
 - This is for you. We just want to make sure you started, you have a direction, and you are on track.
 - We want to see 1-2 pages with initial plots or results with a proposal of sketch of the proof.
 - First draft of the abstract.
- 10 points: we will give them if we think you fruitfully worked 9h.

Tasks in detail

Timeline and Grading

- **Nov 1-14:** Show up to office hours.
 - We want to make sure you are on track.
 - Discuss the first checkpoint.
 - Help you overcome major difficulties.
 - Help you direct the efforts.

Tasks in detail

Timeline and Grading

- **Dec 2-4:** Presentations.
 - Short presentation of the project, questions are also on the course content.
- 25 points, -5 overtime penalty.
 - More details to come when we will know how many projects we have.

Tasks in detail

Timeline and Grading

- **Dec 10:** Final report.
 - min 5, max 9 pages.
 - In the form of a Neurips article.
 - Link to public repository (Github) with the code.
 - Unlimited appendix.
- 40 points:
 - Execution of the idea (10).
 - Positioning and Related Work (10).
 - Clarity (5).
 - Novelty (5).
 - Limitations (5).
 - Implications and Significance (5).

Outline

- Course description/logistic
- Why we need a science of intelligence
- Why we need theory in ML
- From approximation and generalization to optimization

What I am going to tell you

I believe that we need a *science of intelligence*, and we need a *theory* of the machines we build and eventually of our own intelligence.

My next two points are 1) why a science and 2) why theory. I will then make the point that the theory should spell out *fundamental principles of intelligence*.

To make this concrete and to connect it to what we will do in the class, I will then take a quick overview of the three parts of ML theory: approximation, generalization and optimization. In approximation and generalization, we may know one such fundamental principle: compositional sparsity. Optimization is still work in progress: half of the class will look at ongoing work to establish whether sparsity and what else can be the foundations on which to build a theory.

Accelerated Article Preview

Highly accurate protein structure prediction with AlphaFold

Received: 11 May 2021

Accepted: 12 July 2021

Accelerated Article Preview Published
online 15 July 2021

Cite this article as: Jumper, J. et al. Highly
accurate protein structure prediction with
AlphaFold. *Nature* <https://doi.org/10.1038/s41586-021-03819-2>

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Židek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli & Demis Hassabis

PaLM: Scaling Language Modeling with Pathways

Attention Is All You Need

Emergent Abilities of Large Language Models

Jason Wei¹ Yi Tay¹ Rishi Bommasani² Colin Raffel³
Barret Zoph¹ Sebastian Borgeaud⁴ Dani Yogatama⁴ Maarten Bosma¹
Denny Zhou¹ Donald Metzler¹ Ed H. Chi¹ Tatsunori Hashimoto²
Oriol Vinyals⁴ Percy Liang² Jeff Dean¹ William Fedus¹

Gemini 2.5 Pro Capable of Winning Gold at IMO 2025*

Yichen Huang (黄溢辰)[†] Lin F. Yang (杨林)[‡]

July 28, 2025

Amazing progress in the engineering of Intelligence!

Ironically we do not understand well why systems such as LLMs work as well as they do...

We should invest not only in the engineering of intelligence (A+M+M+...do that) but also in the *science of intelligence*.

*Understanding human intelligence seemed the best way
to develop AI*

Should we give up?

*My answer is no: first, even if we develop AI we still need
to understand the brain*

Why science of intelligence

- The first is obvious for us researchers: understanding how our brain creates the human mind — [how the brain computes](#) — is the greatest problem in science.
- Despite its recent successes, the engineering-only approach may hit a wall soon...brain science may then provide the way out. In particular, I believe it is virtually impossible to build an AI with human-like intelligence without understanding the human brain/mind (because there are infinite forms of intelligence).

Why science of intelligence

- To develop powerful brain-machines interfaces we need to know *how human intelligence works and how the brain is wired up*, where to plug the connectors and what are the communication protocols. This is, right now, the only plausible way to achieve many transformative goals, including, eventually
 - make ourselves much smarter*
 - live forever*
 - travel to the stars*

Outline

- Course description/logistic
- Why we need a science of intelligence
- Why we need theory in ML
- From approximation and generalization to optimization

Alessandro Volta and Electricity

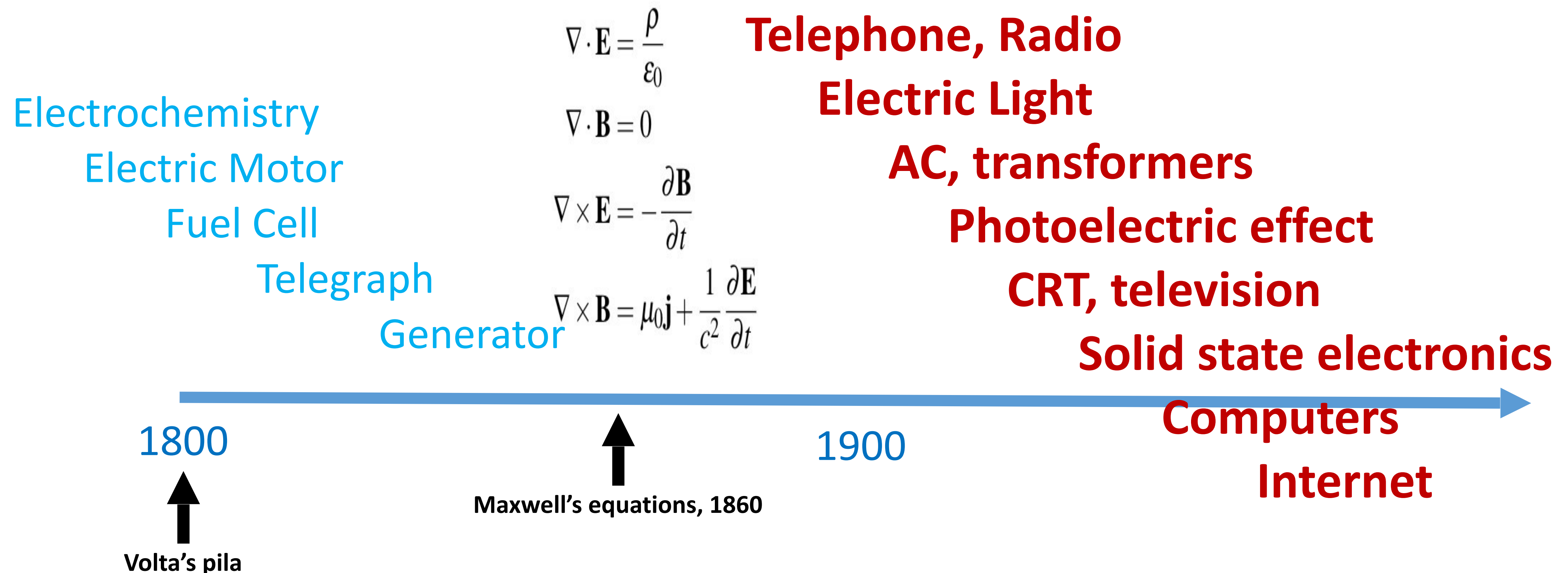
The paper on the *pila* was published in 1800 in the Proceedings of the Royal Society, London.

Volta was made a Count by Napoleon.



No theory and little understanding of what electricity is.

cf: Electricity from Volta and **after Maxwell**



We need a theory of deep learning

- because we need better systems
- because a theory provides deep explainability + control (for alignment etc.)
- because it may help find and understand basic principles of intelligence, including of our own intelligence

We need a theory of deep learning

As we have discussed, current attempts to understand, predict and control behaviors of machine learning models are not based on fundamental principles and are unlikely to be successful. On the other hand, for the reasons given above, there is a real hope that such fundamental principles may exist and can be discovered. This is evidenced by recent progress in the theory of deep learning... These considerations call for a major expansion in the effort to understand fundamental principles of deep learning. **Understanding the fundamental theory of deep learning and applying its analyses to real systems is a compelling and urgent need.**

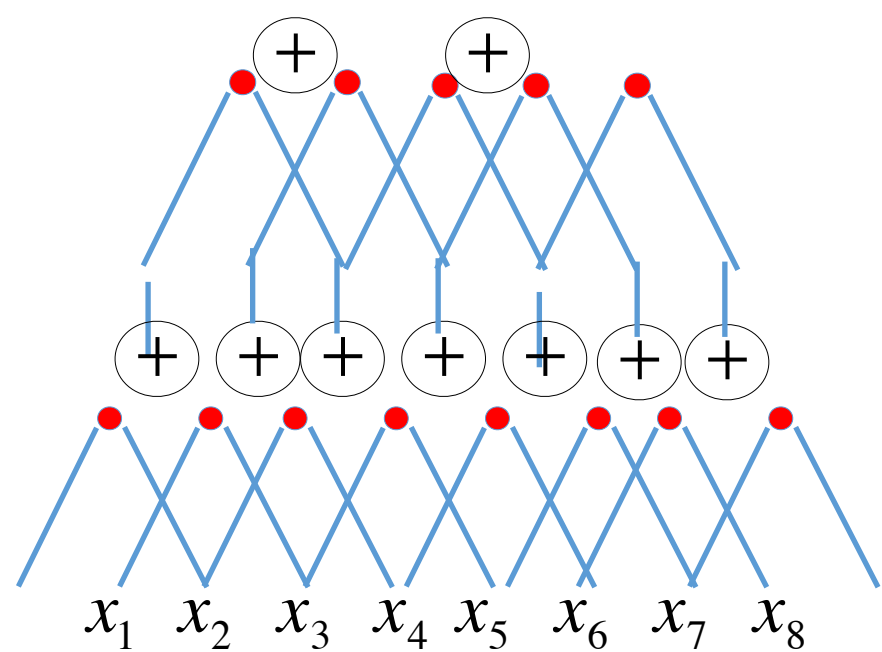
The theory we need
should be looking for

fundamental principles of intelligence,

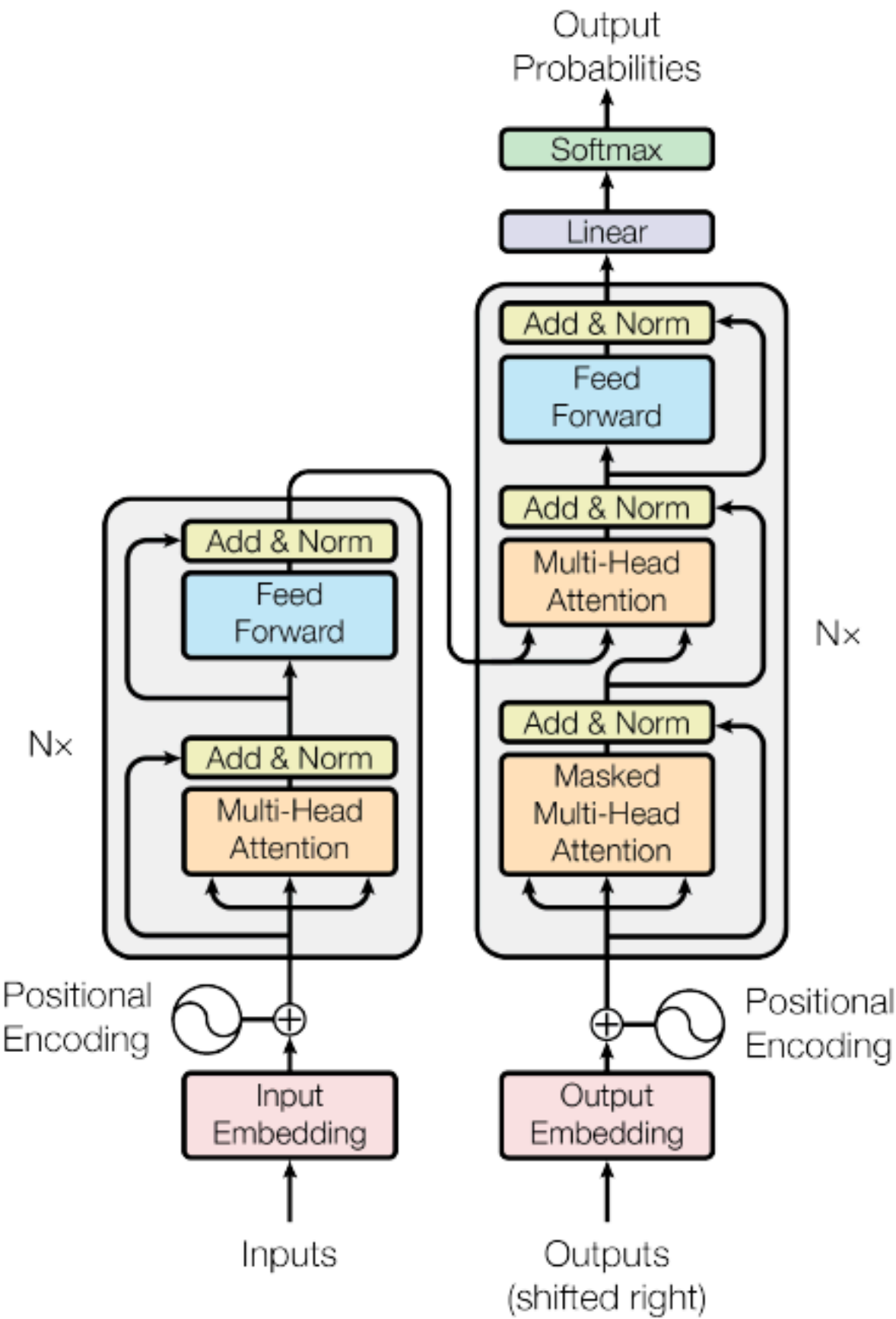
not just theorems for the sake of math

The stunning success of deep learning (DL) over the past decade is a puzzling natural phenomenon of great scientific interest. This success has been amply demonstrated through a plethora of convincing experiments, and yet it is crying out for a thorough scientific understanding through foundational research (C.P.)

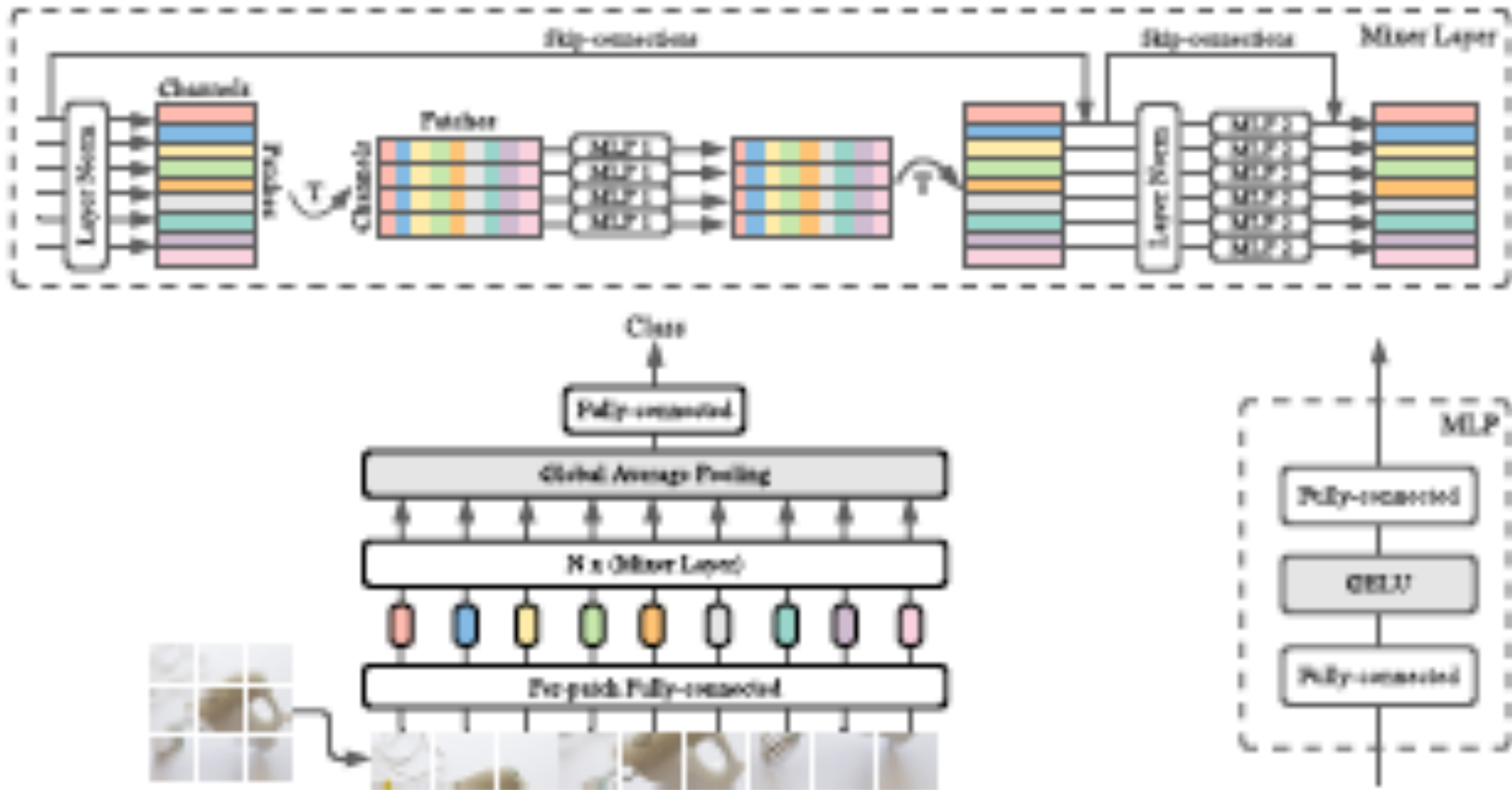
For the first time in human history we have more than just human intelligence to study. Are there common fundamental principles?



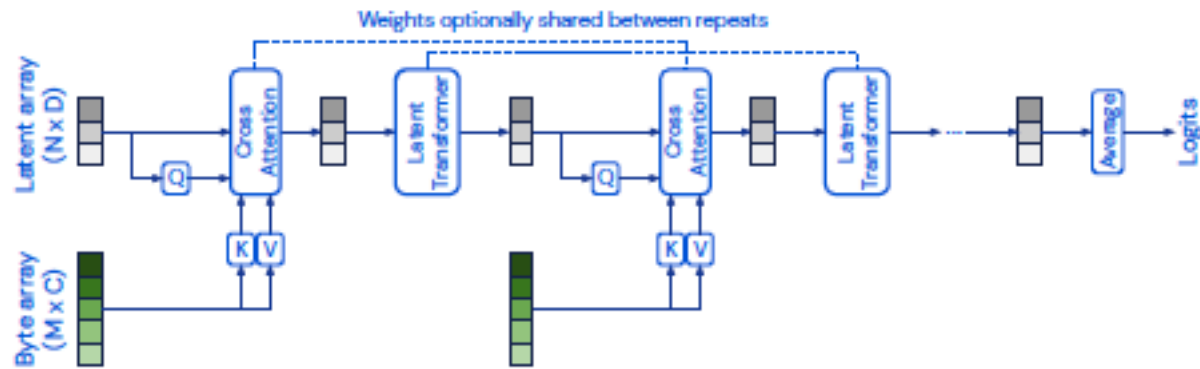
CNN with or without weight sharing



The Transformer - model architecture.



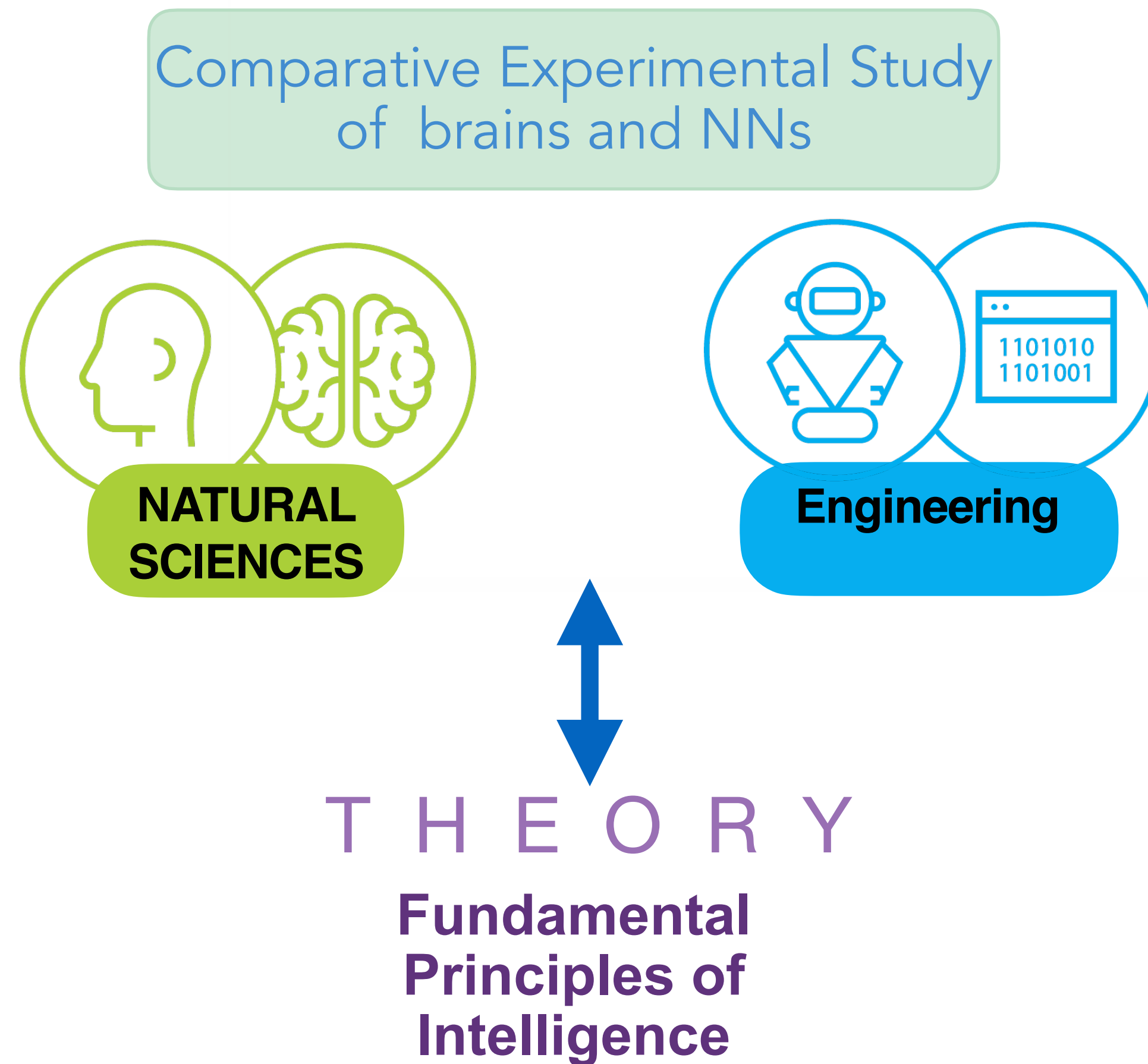
MLP mixer



The Perceiver



A comparative study of intelligence



- Comparative study of intelligent behavior: what are similarities and differences? What are the roots of the differences?
- Recording from inside the brain and the LLMs, looking for circuits, again similarities and differences.
- Theory looking for fundamental common principles

Why theory: Principles of Intelligence

For the first time in human history, we can compare different kinds of minds. Understanding them is no longer just an intellectual pursuit — it is an urgent scientific and societal necessity, and it may be the key to understanding ourselves. By examining how intelligence manifests across brains and machines, we may uncover fundamental principles that govern cognitive systems.



$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} + \frac{1}{c^2} \frac{\partial \mathbf{E}}{\partial t}$$

What I told you

- We need a *science of intelligence*
- We need a *theory* of intelligence
- The theory should be like a theory in physics, spelling out *fundamental principles of intelligence*

Outline

- Course description/logistic
- Why we need a science of intelligence
- Why we need theory in ML
- Approximation, generalization and optimization
- Reasoning

A Conceptual Framework for Machine Learning

The goal of learning theory might be said to be: “find” a parametric f that approximates f_μ — the unknown conditional probability given x — as well as possible. We need a space $\mathbb{F}$ of *parametric functions* with non-exponential number of parameters + powerful enough to approximate a broad range of possible target functions.

The first step in this conceptual framework, is

- choose $\mathbb{F}$ (*approximation theory*);
- find the values of the parametric approximator by minimizing a loss function on a training set (*optimization theory*);
- study generalization properties of the trained representation (*generalization theory*);

A Conceptual Framework for Machine Learning

A supervised learning problem is defined by an unknown density μ on the product space $X \times Y$ from which the training data $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$, -- supposed to be random -- is drawn.

The measure μ defines a function

$$f_\mu : X \rightarrow Y$$

The goal of learning theory might be said to be: "find" f that minimizes the generalization error, that is $\int_X (f - f_\mu)^2 d\mu$; in other words find a parametric f that approximates f_μ as well as possible.

Conceptual Framework of Machine Learning

1. choose $\mathbb{H}_\lambda$, a space $\mathbb{H}_\lambda$ of functions with parameters λ - in which to search for f .
The functions in $\mathbb{H}_\lambda$ must have a *non-exponential* number of parameters (in the effective dimensionality) AND be powerful enough to approximate a broad range of target functions (realm of approximation theory);
2. optimize the values of the parametric approximator by minimizing a loss function on a training set (realm of optimization theory);
3. study generalization properties of the trained representation (domain of learning theory).

What you will see in the class

We will see this framework in classical machine learning, basically shallow networks, during the first part of the class.

We will see the same framework in the case of deep networks; in this case one of the fundamental principles for approximation and generalization: *compositional sparsity*.

Optimization is still work in progress: a good chunk of 9.520 this year will look at ongoing work on it.

Connecting sparsity to transformers and reasoning

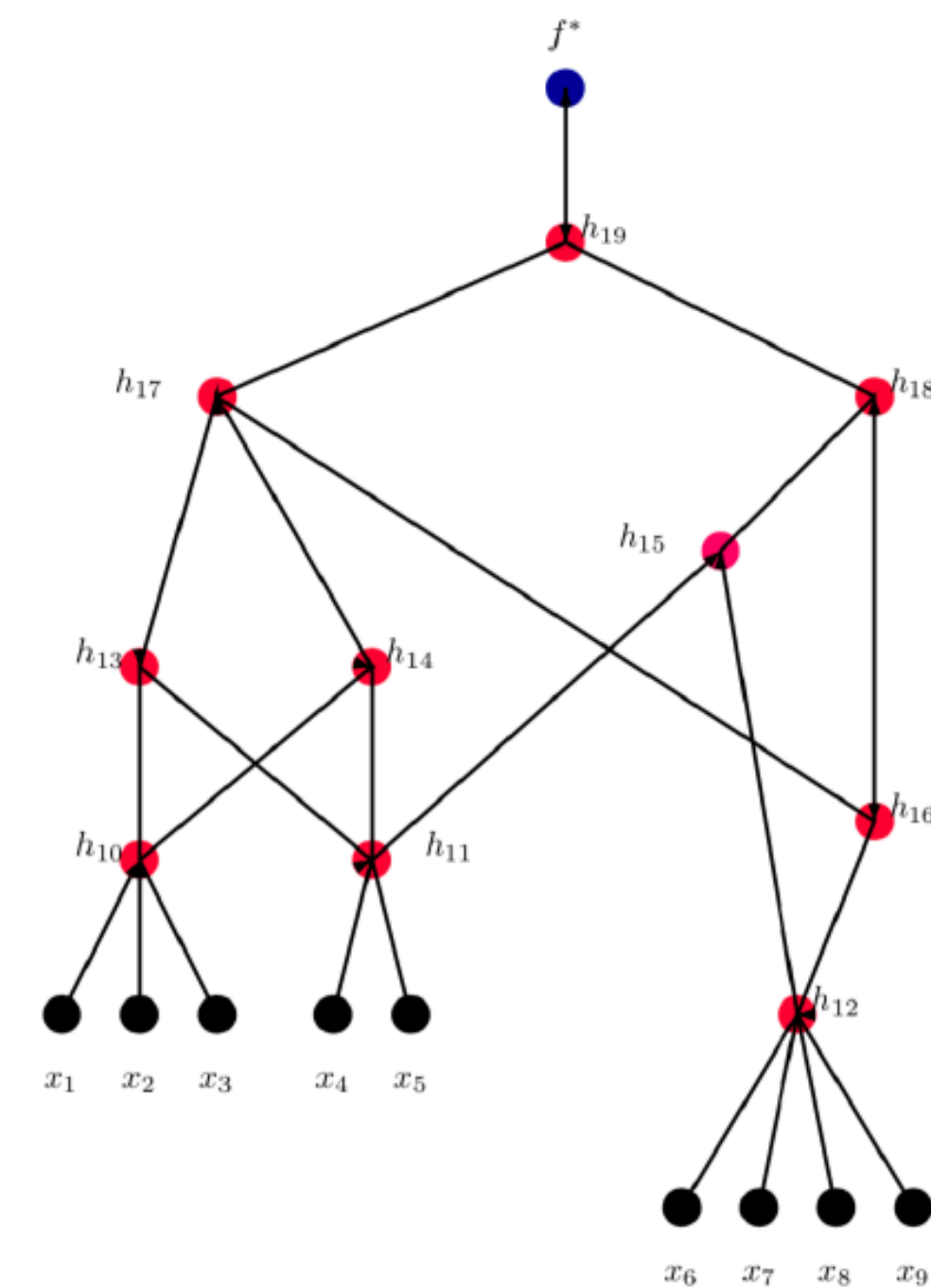
The next few slides will give a flavor of the topics of sparsity and compositionality and how they connect to ongoing research on reasoning in LLMs.

If you know about ML this will be fine...otherwise don't worry. We will look at this again.

Definition: Compositional Sparsity

A function $f : \mathcal{X}^d \rightarrow \mathcal{X}$ is *compositionally sparse* if it can be represented as the composition of at most $O(\text{poly}(d))$ constituent functions each of which is *sparse*, i.e., depends on at most a constant (small) number of variables (so that it does not suffer the curse of dimensionality).

A compositionally sparse function can be represented in terms of a DAG (Directed Acyclic Graph), in which the leaves represent the inputs, the root denotes the output function, and the internal nodes represent the constituent functions.



Definition: Efficient Turing computability

If f is *efficiently computable* then it can be approximated to arbitrary precision by a Boolean function in class **P** operating on bitstring representations of its inputs.

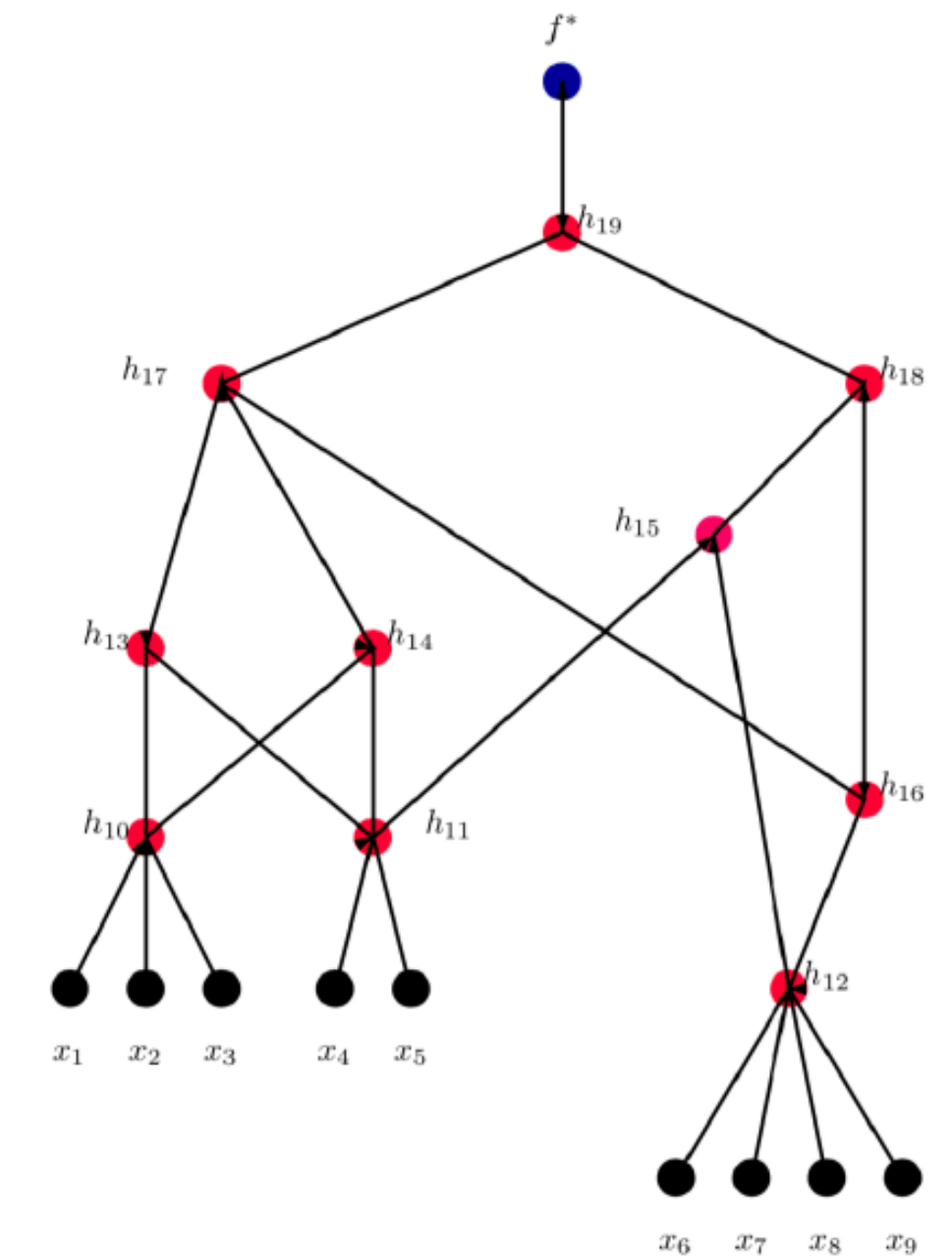
Theorem: efficient computability implies compositional sparsity and the latter implies existence of learnable representation

If a function is *efficiently Turing computable* then

1. it is *compositionally sparse*, in the sense of a sparse DAG;
2. there exists a *deep network that approximates* it arbitrarily well without curse of dimensionality
- 3.

Repeating in different words

Theorem *All efficiently Turing computable functions are compositions of sparse functions; for each such function there exist a sparsely connected deep network that approximates it without curse of dimensionality.*



This result solves a few puzzles

- Why do deep neural nets apparently escape the curse of dimensionality?
- The puzzle has a long history in mathematics starting with Hilbert's thirteen problem
- It affects approximation (and optimization)

It started with some puzzles of Deep Learning

How can deep nets escape the curse of dimensionality (in approximation)?

Why do kernel machines suffer the curse of dimensionality?

Why deep networks? Why is depth important?

Why are CNNs so much better than dense NN?

What controls generalization? Norm-like complexity, sparsity, rank?

Are all functions that can be computed by physical systems sparse?

Sparsity in the weight matrices

- Sparsity in the weight matrices of a deep network implies generalization bounds that are order of magnitude better than for deep networks with dense matrices (Galanti, Xu, Galanti, Poggio, Neurolps 2023)
- Thus sparsity seems a key principle for *approximation* and *for generalization*

The theorem “explains” a magic of transformers

From our theorems it follows as a corollary that every Turing computable function is learnable if training examples for each of the constituent functions are available. In other words *our results prove*, as a special case, the following (see also Malach, 2023)

Theorem (informal)

Any computer program or intelligent agent that can be simulated by a computer, can be learned, given the right training set, by a simple next-token (auto-regressive) predictor.

In other words: optimization (learnability) of Boolean functions

A compositionally sparse Boolean function is in general difficult to learn (it is P) because learnability of P would violate cryptographic assumptions; however the constituent functions — since they are sparse — are usually easily learnable.

Connection with Reasoning in SoA LLMs

General “intelligence” — as in pretrained transformers — is not consistently compositional (from a number of empirical data).

Thus SoA systems try to exploit CoT ideas investing test-time compute during inference to improve the ability to solve novel complex tasks via a chain of sequential reasoning steps, searching over possible constituent functions.

Connection with Reasoning

The main premise of PAC-learning is that if the number of examples seen during training is sufficiently large with respect to the VC-dimension of H , then we can “probably approximately” find the hypothesis g . In particular, for finite H , the required number of examples (sample complexity) grows logarithmically with $|H|$.

This is a very powerful guarantee as we can choose H to be the set of all functions we can implement in Python using d characters and learn this class by order of d examples.

Connection with Reasoning

The size $|H|$ of Boolean functions in d variables is 2^{2^d} . So there is no sample complexity but huge computational complexity.

However, let us assume the obvious restriction to consider functions in H that are also in P . In particular, we assume “sparse” Boolean functions of at most k variables as primitives. That is, our building blocks are functions

$$g : \{-1, 1\}^k \rightarrow \{-1, 1\}, \quad k \ll d$$

There are 2^{2^k} such primitives. Now consider functions built as compositions of such primitives, arranged in a DAG of depth L .

Formally:

$$f(x_1, \dots, x_d) = g^{(L)} \left(g^{(L-1)} \left(\dots g^{(1)} (x_{i_1}, \dots, x_{i_k}) \dots \right) \right).$$

Connection with Reasoning

This is the Boolean-circuit (or formula) model, restricted to nodes of fan-in k . The number of distinct depth- L circuits of size s (number of nodes in DAG) using such primitives is at most $\left(2^{2^k}\right)^s \cdot d^{ks}$. This is much, much smaller than 2^{2^d} when $k \ll d$ and s, L are moderate. So sparse compositionality implies an *exponentially smaller hypothesis class compared to all Boolean functions which could be generated more easily*.

Connection with Reasoning

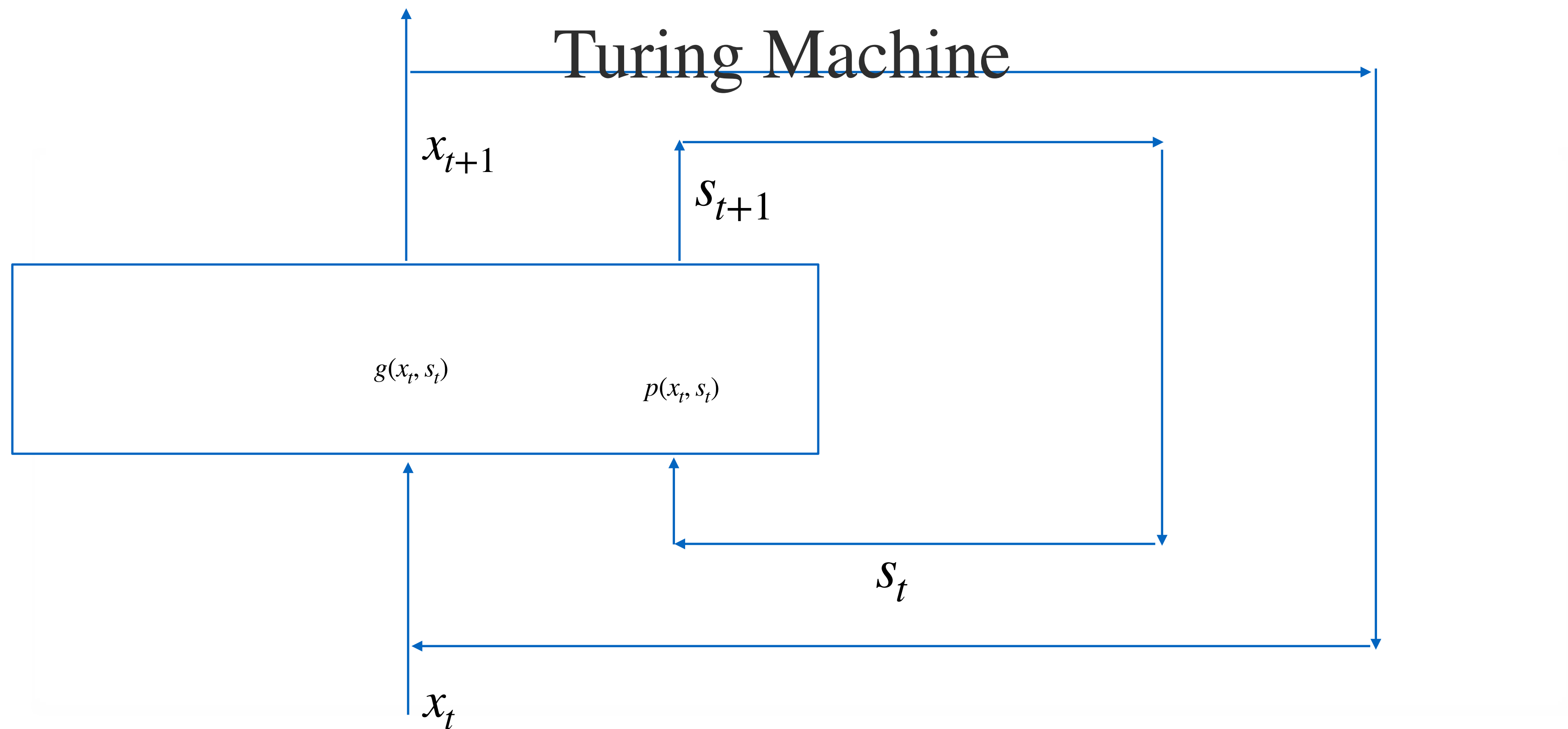
Every Boolean function in **P** must be “simple” in some representation. In particular, it must be either **sparse** (few significant Fourier monomials, like parity), or **compositionally sparse** (efficient circuit built from low-arity functions, like majority or AND/OR). Arbitrary Boolean functions (the vast majority) lack such structure and hence require exponential-size circuits, putting them outside **P**. Thus being in **P** forces Boolean functions into the tiny corner of the function universe that is either **sparse** or **compositionally sparse**. The exact kind of sparsity depends on the computational model, but some form of it is necessary. We describe it as a DAG with s nodes of depth L with at most k inputs for each node.

Final Remarks

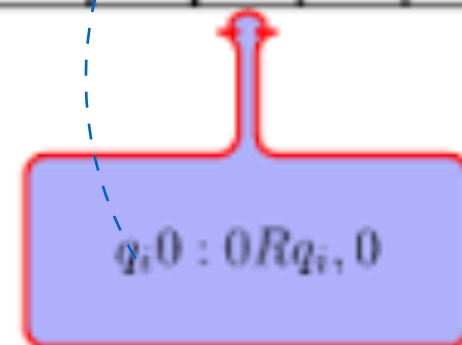
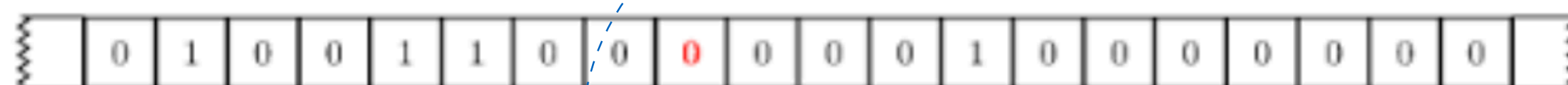
Compositionality enables:

- Efficient learning
- Transfer
- Robust generalization

Remarkably, sparse compositionality is very common: all efficiently computable functions are compositionally sparse (but their decomposition is not unique)



$$x_{t+1} = g(x_t, s_t) \quad s_{t+1} = p(s_t, x_t)$$



Turing Machine

A Turing machine as a deterministic dynamical system

At each discrete time step, the machine updates its configuration consisting of a finite control state and the contents of the tape (including the head position).

At time step t , the configuration of the machine is given by

$$C_t = (s_t, x_t),$$

where $s_t \in S$ is the finite internal state, x_t encodes the entire tape contents together with the current head position.

The transition function of the Turing machine induces a global deterministic update:

$$C_{t+1} = \Phi(C_t).$$

Thus, the evolution of the machine can be described as an iterated map on the space of configurations.

Final Remarks

What about PDEs and sparsity?

The term *compositional sparsity* come from W. Dahmen writing about PDEs

ML is really interested in learning *computable functions*. Thus the necessity of computability for learnability is OK because it follows from computability.

For the PDEs community the question is a bit different because it wants to simulate physical phenomena. The key question then is a version of the physical Church-Turing thesis:

Can all (classical) physical phenomena be efficiently simulated by a Turing Machine?