Foundations of Deep Learning: Compositional Sparsity

Tomaso Poggio,
McGovern, CSAIL
MIT

CENTER FOR
Brains
Minds+
Machines

DRAFT

Thirty Brief Lectures on Foundations of Deep Learning

Tomaso Poggio *with Gemini (and ChatGPT)*

based on work with Pierfrancesco Beneventano, Yulu Gan, David Koplow, Qianli Liao, Daniel Mitropolsky, Liu Ziyin

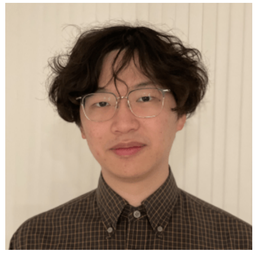
Intelligence and its Fundamental Principles

https://poggio-lab.mit.edu/

First version: January 2, 2026

Current version: February 25, 2026

A New “NeoLab”: Theory for AI





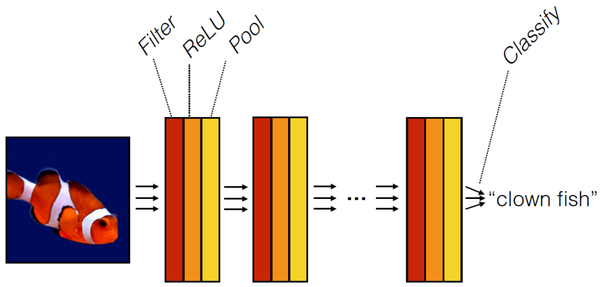




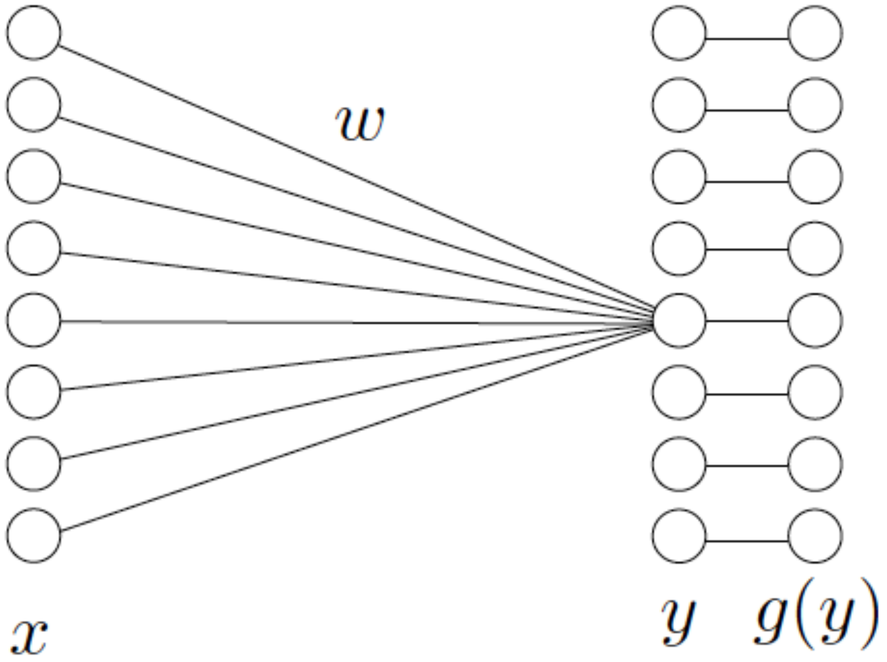



Computation in a deep neural net

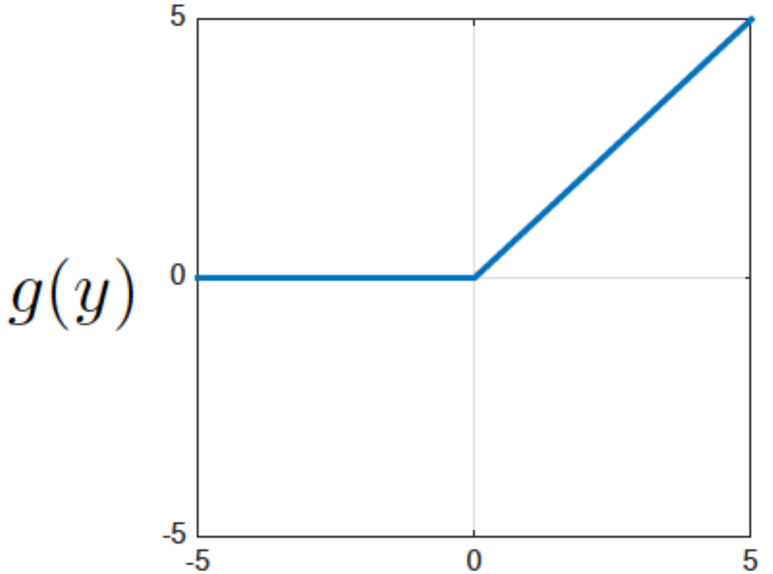
Computation in a neural net



$$f(\mathbf{x}) = f_L(\dots f_2(f_1(\mathbf{x})))$$



Rectified linear unit (ReLU)



$$g(y) = \max(0, y)$$

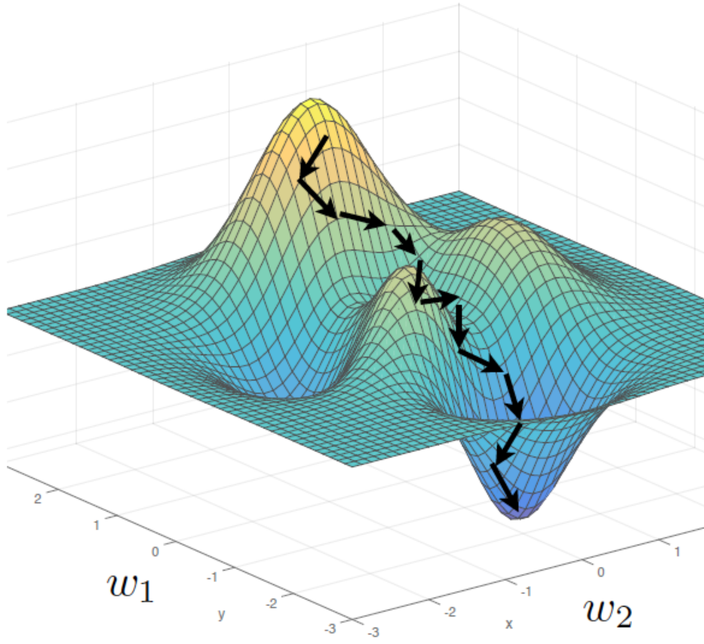
Gradient descent and SGD

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_i \ell(\mathbf{z}_i, f(\mathbf{x}_i; \mathbf{w})) = L(\mathbf{w})$$

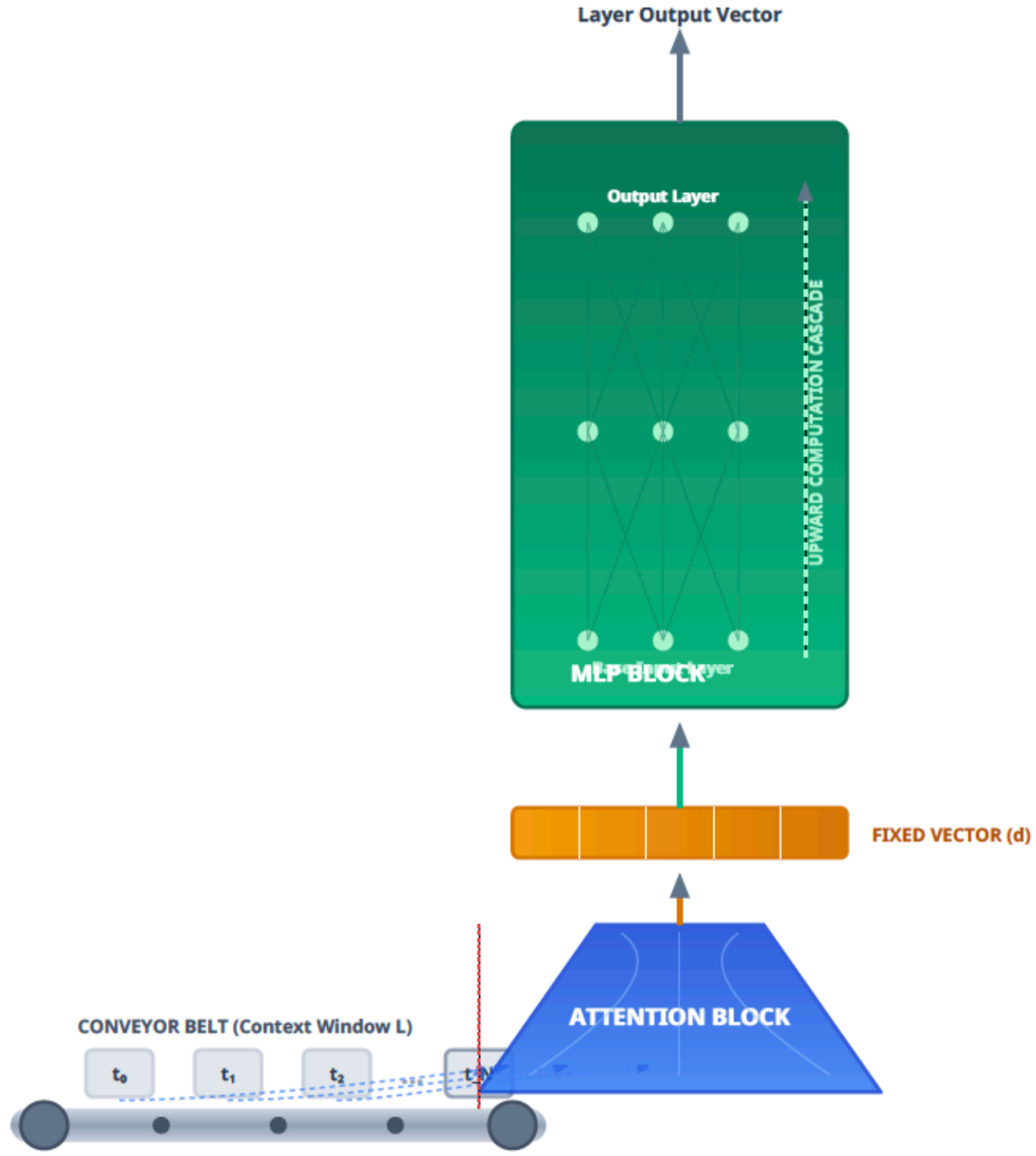
One iteration of gradient descent:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \frac{\partial L(\mathbf{w}^t)}{\partial \mathbf{w}}$$

learning rate



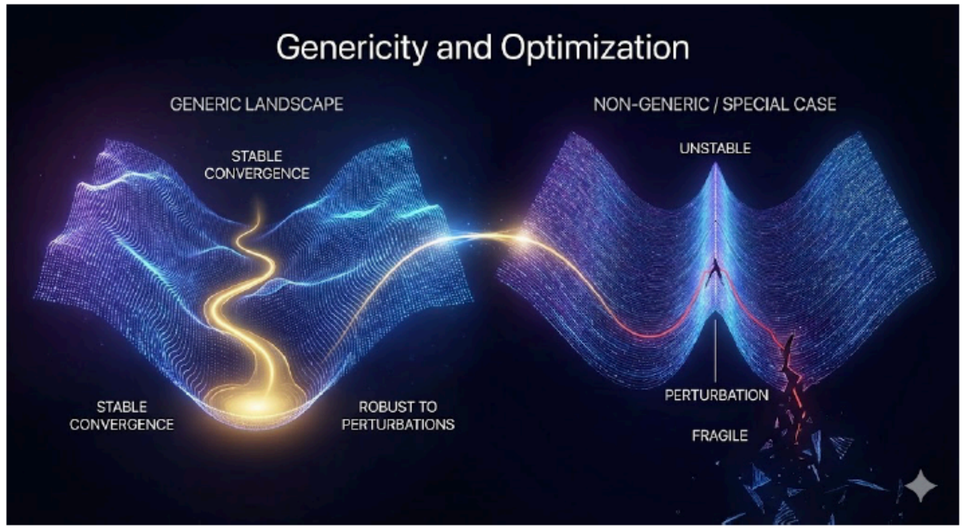
Transformer



Plan

- Why Theory?
- Genericity implies Optimization
- Efficient Turing computability implies Compositionality which implies Deep Networks
- Foundational Principles: Compositionality, Genericity…

Genericity and Optimization



Why does local gradient descent succeed in optimizing deep networks, despite the non-convex nature of the loss landscape? The proposal is that the answer lies in a property of learnable functions we call *Genericity:*

- it makes sense to only learn *generic* functions — functions with structure invariant to basic transformations like shifts in the coordinate origin; non-generic functions have measure zero.
- generic functions always have "linear footprints"—low degree correlations—that allow gradient descent to work.

Plan

- Why Theory?
- Genericity implies Optimization
- Efficient Turing computability implies Compositionality which implies Deep Networks
- Foundational Principles: Compositionality, Genericity…

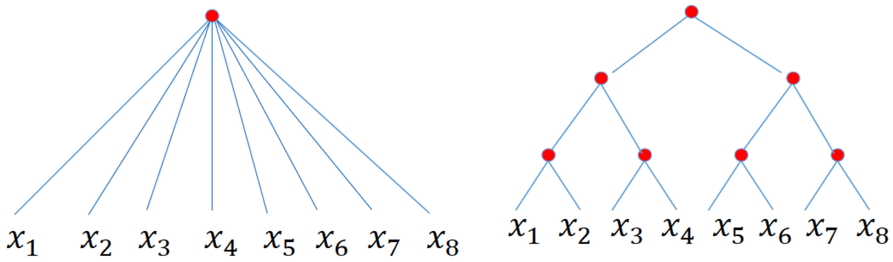
Caveat

Genericity and Compositional Sparsity
are more about the functions to be learned
than about networks

Several puzzles in AI are solved by Compositional Sparsity

What is the curse of dimensionality?

$$y = f(x_1, x_2, \cdots)$$



Theory says that both shallow and deep network can approximate a function of d variables equally well. It says that both should suffer from *the curse of dimensionality*: number of parameters depends exponentially on d: $N = O(\epsilon^{-\frac{d}{m}})$

As an example for a CIFAR image $N \approx (0.1^{-\frac{1000}{1}}) \approx 10^{1000}$

Mhaskar, Poggio, Liao, 2016

Compositional Sparsity answers several of the puzzles of Deep Learning

Why do kernel machines suffer the curse of dimensionality?

Can deep nets represent/approximate high-dimensional functions?

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

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?

Compositionality

Compositionality in Computer Science and Language

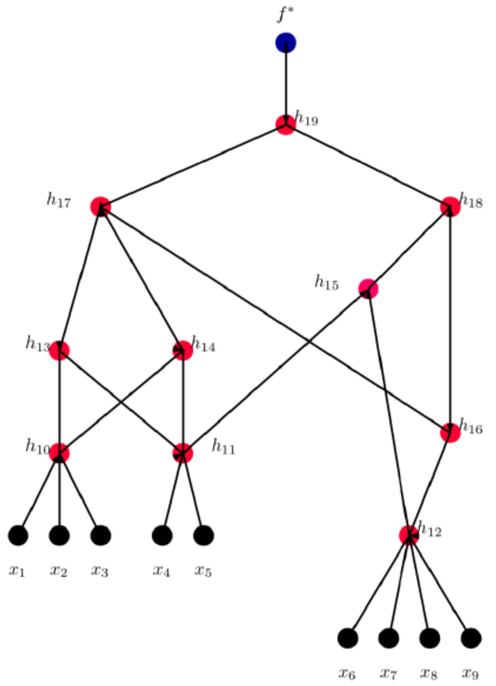
Compositionality is a foundational principle in both computer science and natural language that enables scalability, generalization, and interpretability. At its core, *compositionality means building complex systems or meanings from simpler parts* in a structured way.

Definitions

Definition: Compositional Sparsity

A function $f : \mathcal{X}^d \to \mathcal{X}$ is *compositionally sparse* if it can be represented as the composition of at most $O(poly(d))$ *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.



Note on Sparse Compositionality

I use the term Sparse Compositionality to emphasize simplicity of constituent functions (compositionality is always trivially true because $f = I \circ f$). Notice that sparse compositionality is in general not unique: if $f = g_L \circ g_{L-1} \circ \ldots \circ g_1 = g_L \circ \overline{g}$ with $\overline{g} = g_{L-1} \circ \ldots \circ g_1$.

Definition: Efficient Turing computability

If f *is efficiently computable then it is approximable to arbitrary precision by a Boolean function in class* $\mathbf{P}$ *operating on bitstring representations of its inputs. This also means that efficiently computable functions can be computed in non-exponential time.*

Theorems

Theorem: efficient computability implies compositional sparsity and the latter implies existence of a deep network representation

---

If a functions 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

Poggio, Fraser, Bulletin of the AMS, 2024

Proof

Theorem 1 (TM $\Rightarrow$ LTF circuits $\Rightarrow$ bounded-fan-in Boolean DAGs). *Let $f : \{0,1\}^n \to \{0,1\}$ be computable by a deterministic Turing machine M in time $T(n)$. Then for each input length n there exists:*

1. *a Boolean circuit C_n^{Bool} of size and depth $\mathrm{poly}(T(n))$ computing f on $\{0,1\}^n$;*
2. *a (linear) threshold circuit C_n^{LTF} of size $\mathrm{poly}(T(n))$ computing f; and*
3. *a bounded-fan-in Boolean circuit C_n^{bfi} of size and depth $\mathrm{poly}(T(n))$ computing f,*

yielding a compositionally sparse DAG representation for f at input length n.

Proof. Step 1: Turing machine to Boolean circuits. By standard simulation (configuration graph unrolling), a deterministic TM running in time $T(n)$ on inputs of length n is simulated by a Boolean circuit C_n^{Bool} of size and depth $\mathrm{poly}(T(n))$. Each time step of M contributes a layer whose gates enforce the local update rule on tape cells and finite control [1, 7].

Step 2: Boolean circuits to linear threshold circuits. Each Boolean gate (AND, OR, NOT) is a special case of a linear threshold gate (LTF):

$$\mathrm{AND}(x,y) = 1\{x+y \geq 2\}, \quad \mathrm{OR}(x,y) = 1\{x+y \geq 1\}, \quad \mathrm{NOT}(x) = 1\{-x \geq 1\}.$$

Replacing each Boolean gate in C_n^{Bool} with the corresponding LTF yields C_n^{LTF}, preserving size up to a constant factor.

Step 3: Bounding fan-in. An LTF gate on r inputs computes $1\{\sum_{i=1}^{r} w_i x_i \geq \theta\}$ for integers w_i, θ. This is implemented by a Boolean adder network computing the binary representation of $S = \sum_i w_i x_i$ and a comparator circuit checking $S \geq \theta$. Both components have bounded fan-in (e.g., 2) and size $\tilde{O}(r)$ with depth $O(\log r)$ [1]. Since $r_{\max} \leq \mathrm{size}(C_n^{\mathrm{LTF}}) = \mathrm{poly}(T(n))$, the resulting circuit C_n^{bfi} remains polynomial in size and depth.

Step 4: Compositional sparsity. Viewing C_n^{bfi} as a DAG with constant in-degree (fan-in ≤ 3) provides a compositionally sparse representation (Def. 2). $\square$

Lemma 1 (Bounded-fan-in circuits for $F_{n,m_{\text{out}}}$). *If f is efficiently computable (Def. 1), then for each (n, m_{out}) there exists a bounded-fan-in Boolean circuit $C_{n,m_{\text{out}}}$ of size and depth $\text{poly}(n + m_{\text{out}})$ computing $F_{n,m_{\text{out}}}$.*

Lemma 2 (Circuits are compositional DAGs). *Viewing each gate of $C_{n,m_{\text{out}}}$ as a node with fan-in ≤ 3 yields a DAG with local arity $k \leq 3$, size $s = \#\text{gates} = \text{poly}(n + m_{\text{out}})$, and depth $L = \text{poly}(n + m_{\text{out}})$. Thus $\{F_{n,m_{\text{out}}}\}$ is compositionally sparse.*

Lemma 3 (Gate emulation). *For standard activations, each Boolean gate $g : \{0,1\}^r \to \{0,1\}$ with $r \leq 3$ admits a constant-size neural subnetwork N_g that computes g exactly on $\{0,1\}^r$, or with error $\leq \varepsilon$ on a δ-neighborhood (size depending only on $\log(1/\varepsilon)$, $\log(1/\delta)$) [9]. Multi-bit wires are handled in parallel.*

Lemma 4 (Circuit $\Rightarrow$ neural network). *Replacing each gate in $C_{n,m_{\text{out}}}$ by N_g and wiring accordingly yields a network $\Phi_{n,m_{\text{out}}}$ of size/depth $\text{poly}(n + m_{\text{out}})$ that computes (or ε-approximates) $F_{n,m_{\text{out}}}$ on encoded inputs $Q_n(x)$. Choosing sub-gate accuracies so the accumulated error is $\leq 2^{-m_{\text{out}}}$ gives the desired precision.*

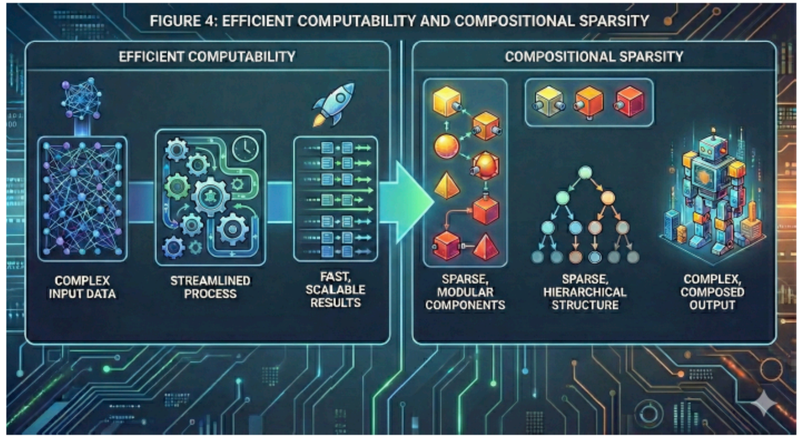
Theorem 2 (Efficient computability $\Rightarrow$ compositional sparsity and neural approximation). *If f is efficiently computable (Def. 1), then for every (n, m_{out}) there exist:*

1. *a compositionally sparse DAG for $F_{n,m_{\text{out}}}$ with $k \leq 3$, $s \leq \text{poly}(n + m_{\text{out}})$, $L \leq \text{poly}(n + m_{\text{out}})$ (Lemmas 1–2); and*
2. *a deep network $\Phi_{n,m_{\text{out}}}$ with size/depth $\text{poly}(n + m_{\text{out}})$ such that*

$$\left\| \Phi_{n,m_{\text{out}}}(Q_n(x)) - f(x) \right\|_\infty \leq 2^{-m_{\text{out}}}, \qquad \forall x \in [0,1]^d.$$

Equivalently, for $\varepsilon = 2^{-m_{\text{out}}}$ and $M = n + \lceil \log_2(1/\varepsilon) \rceil$, there is Φ_ε with size/depth $\text{poly}(M)$ achieving $\sup_x \|\Phi_\varepsilon(Q_n(x)) - f(x)\|_\infty \leq \varepsilon$.

Efficient Computability and Compositional Sparsity of Functions



Theorem

- *Efficient computability* $\Longrightarrow$ *compositional sparsity,*

that is, polynomial-time Turing computation $\Longrightarrow$ *bounded-fan-in* computation DAGs

Theorem

Compositional sparsity of the target function $\Longrightarrow$

- no *curse of dimensionality* when the approximation is via a deep network
- *small* bounded-fan-in circuits
- *deep nets are universal parametric approximants* with non-exponential # parameters

Implications of Sparse Compositionality

1. Autoregressive learning is powerful

The power of next word prediction (that is, of autoregressive learning)

From our theorems it follows *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 predictor.

.

Auto-Regressive Next-Token Predictors are Universal Learners, Malach, 2023
On efficiently computable functions, deep networks and sparse compositionality, Poggio, 2025

Autoregressive Universality

Theorem 4 (Autoregressive learnability via compositional sparsity). *Let f be efficiently Turing computable and let $(F_{n,m_{\text{out}}})$ be its finite-precision family. For each (n, m_{out}) there exists a dataset $\mathcal{D}_{n,m_{\text{out}}}$ over sequences encoding the computation DAG of $F_{n,m_{\text{out}}}$ such that training a linear (or shallow) autoregressive next-token predictor on $\mathcal{D}_{n,m_{\text{out}}}$ yields a predictor $\hat{F}_{n,m_{\text{out}}}$ satisfying*

$$\Pr\left[\hat{F}_{n,m_{\text{out}}}(x) = F_{n,m_{\text{out}}}(x)\right] \geq 1 - \delta$$

with sample complexity and training time polynomial in $s(n, m_{\text{out}})$, $\log(1/\delta)$, and m_{out}. Consequently, composing these predictors yields a neural approximant $\Phi_{n,m_{\text{out}}}$ with accuracy $\varepsilon = 2^{-m_{\text{out}}}$ and size/depth $\text{poly}(n + m_{\text{out}})$. Cf. the universality construction of [4].

Proof idea. Encode each gate evaluation in the computation DAG of $F_{n,m_{\text{out}}}$ as a token and factor the joint distribution along DAG edges. Under this factorization, local predictors learn gate-wise conditionals from polynomial data (since fan-in is bounded), and chaining them recovers $F_{n,m_{\text{out}}}$ with high probability. The construction mirrors [4], instantiated on the bounded-fan-in DAG furnished by Lemmas 1 and 2. □

Thus step–by–step supervision of partial states is sufficient to emulate any efficient Turing computation

The reason is that each of the constituent functions is sparse and sparse Boolean functions are easy to learn

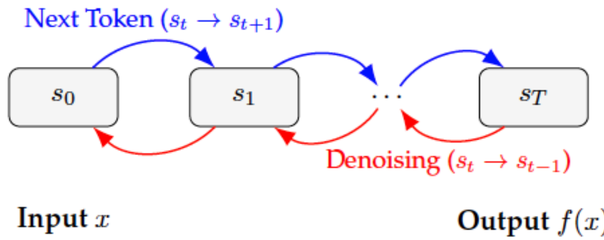
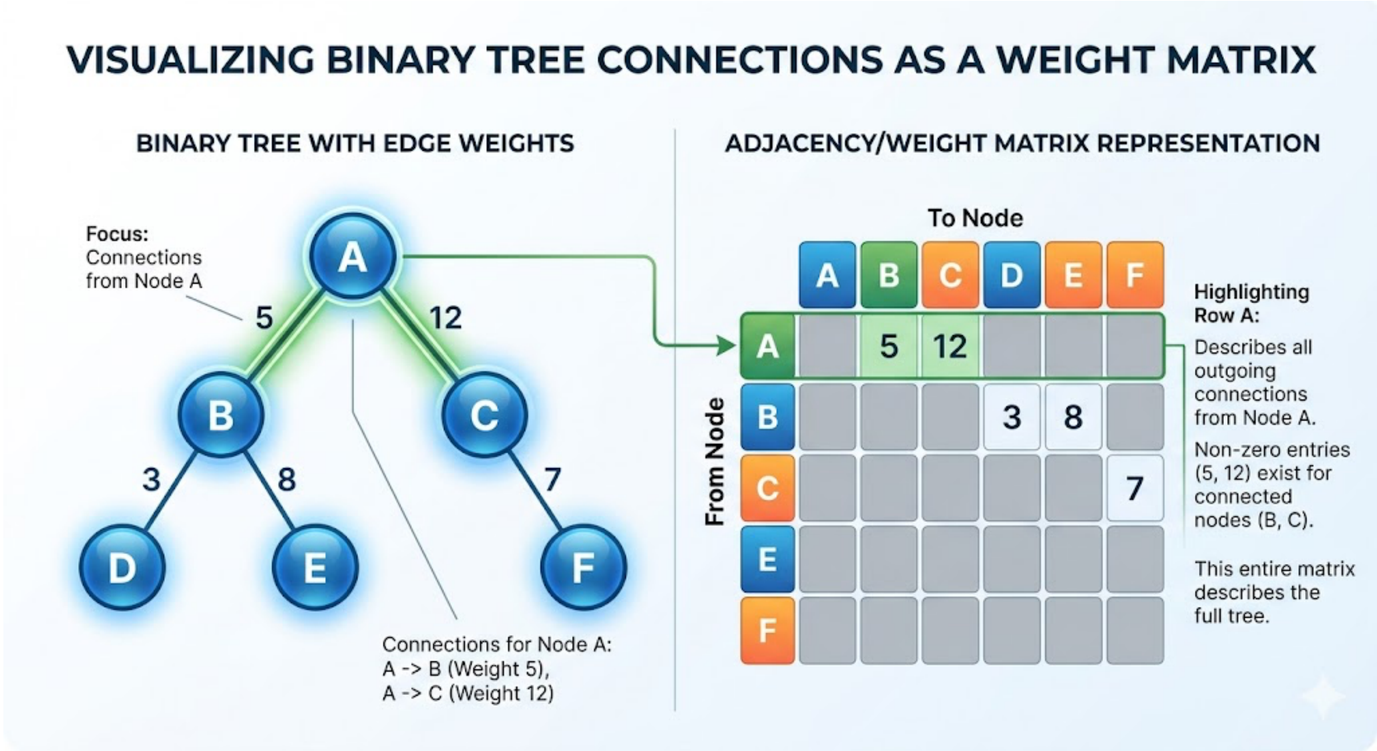


Figure 10.2: Two sides of the same coin. Autoregressive models (top) learn to predict the next computational state, simulating the Turing machine forward. Diffusion models (bottom) learn to recover the previous computational state, effectively reversing the computational trajectory (or denoising it).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

2.Compositional sparsity of the *function* implies sparsity in the weight matrices of the *network*



3. Generalization bounds from sparsity

Generalization bounds from sparsity

Theorem 5 (Rademacher Complexity of Sparse Compositions). *Let $\mathcal{H}_L$ be a class of depth-L ReLU networks. Suppose each layer ℓ has a weight matrix W_ℓ with Frobenius norm bounded by $M_F^{(\ell)}$. The empirical Rademacher complexity is bounded by:*

$$\mathfrak{R}_S(\mathcal{H}_L) \leq \frac{(\sqrt{2\log(2d)})^L \prod_{\ell=1}^{L} M_F^{(\ell)}}{\sqrt{N}}$$

*For a **compositionally sparse** network where each node has fan-in k, the matrices W_ℓ are k-sparse per row. If the non-zero weights are $O(1)$, then $\|W_\ell\|_F \leq \sqrt{s \cdot k}$, where s is the number of hidden units.*

Explaining the theorem

Theorem (informal)

Consider a deep RELU network with a $n \times n$ *dense weight matrix* W *at a certain layer. Its contribution to the Rademacher complexity of the network is* $\|Wx\| = \sqrt{\|Wx\|^2} = \sqrt{(\sum_k^n \|w_k\|^2)\|x\|^2} \leq \|W\|\|x\|$

Suppose now that W *is sparse, that is row* i *contains only* k_i *non-zeros (*$k_i < n$*) components. Then each row contributes* $\|w_i\|^2(\frac{k_i}{n}\|x\|^2)$ *to the sum in* $\sqrt{(\sum_k^n \|w_k\|^2)\|x\|^2}$ *implying that the contribution of* Wx *to the Rademacher complexity of the network is*

$$\sqrt{(\sum_i^n \|w_i\|^2)\frac{k_i}{n}\|x\|^2}$$

The case of non-overlapping convolution gives $\sqrt{(\|w\|^2)\|x\|^2} = \|w\|\|x\|$ instead of $\|W\|\|x\|$

Galanti, Xu, Poggio, NeurIps 2023

4.Mixture of experts

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.

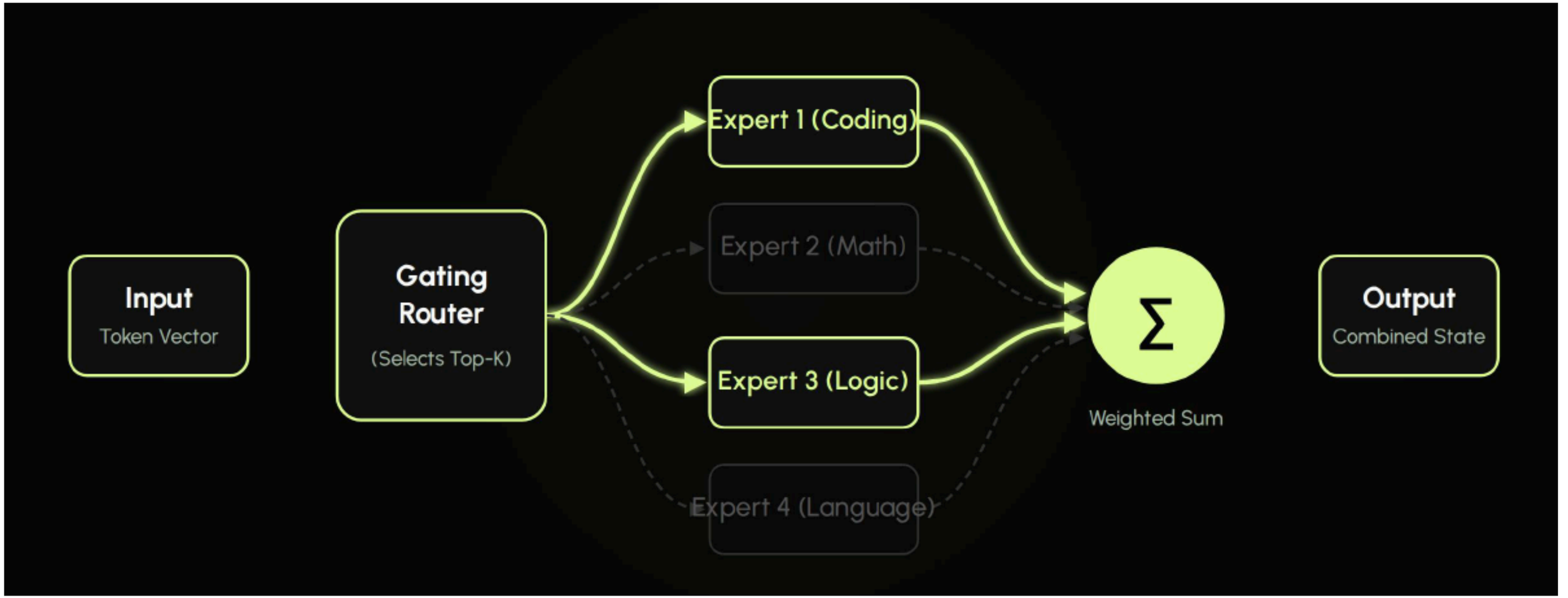
- **Dense Layers:** The first 3 blocks ($1-3$) are standard dense transformer layers. In these initial stages, all parameters are active for every token to establish a foundational representation.
- **MoE Layers:** The subsequent 58 blocks ($4-61$) utilize the DeepSeekMoE architecture.

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.

Theory paper On the Expressive Power of Mixture-of-Experts for Structured Complex Tasks
Mingze Wang, Weinan E

DeepSeek MoE



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.
- **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").

Wang and E
On the Expressive Power of Mixture-of-Experts for Structured Complex Tasks, 2026

Summary

The *“universal”property* of *Sparse Compositionality* of functions/tasks implies

- *Autoregressive Training (prediction of the next word) is powerful*
- *Avoiding curse of dimensionality* via depth
- *Sparse representations*
- *Better generalization*
- *Mixture of experts in LLMs are a good idea*
- *Transfer learning*: Modules trained on one task can be reused
- Justifies the Lottery Ticket Hypothesis (dense, randomly initialized neural networks contain small subnetworks (the "winning tickets") that can match the accuracy of the original network when trained in isolation from their original initializations)
- At fixed accuracy, transformer MLP weights should be prunable to sparsity levels determined by local arity
- ….

The answers to puzzles of Deep Learning

Why do kernel machines suffer the curse of dimensionality?

Can deep nets represent/approximate high-dimensional functions?

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

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?

References

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 61, Number 3, July 2024, Pages 438–456
https://doi.org/10.1090/bull/1820
Article electronically published on May 15, 2024

COMPOSITIONAL SPARSITY OF LEARNABLE FUNCTIONS

TOMASO POGGIO AND MAIA FRASER

ABSTRACT. Neural networks have demonstrated impressive success in various domains, raising the question of what fundamental principles underlie the effectiveness of the best AI systems and quite possibly of human intelligence.

BAMS 2024

On the Power of Decision Trees in Auto-Regressive Language Modeling

---

Yulu Gan
Massachusetts Institute of Technology
yulu@csail.mit.edu

Tomer Galanti
Texas A&M University
galanti@tamu.edu

Tomaso Poggio
Massachusetts Institute of Technology
tp@csail.mit.edu

Eran Malach
Harvard University
eran.malach@gmail.com

NeurIps 2024

Position: A Theory of Deep Learning Must Include Compositional Sparsity

David A. Danhofer · Davide DAscenzo · Rafael Dubach · Tomaso A Poggio

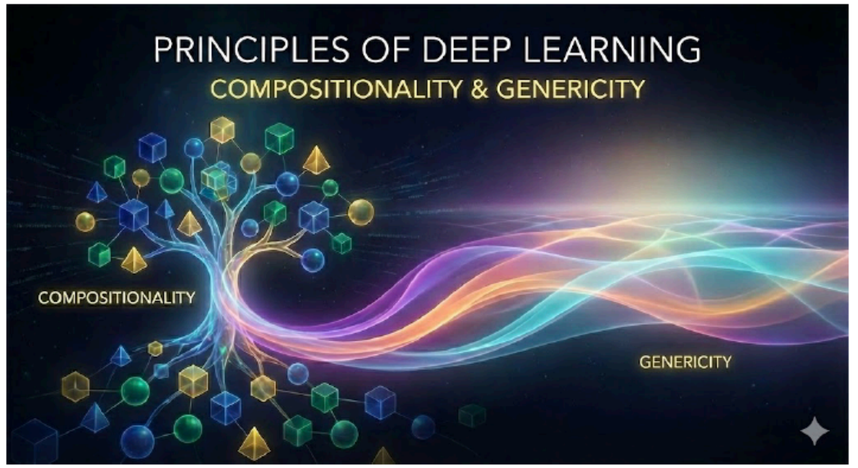
ICML 2025

On efficiently computable functions, deep networks and sparse compositionality

tomaso poggio and menjia xu

CBMM memo, 2025

Summary: (Some) Principles of Deep Learning



Principle I: **Sparse Compositionality**

The property of sparse compositionality in formally implied by the property of efficient Turing computabliity. Functions that are efficiently computable necessarily decompose into bounded-fan-in computational graphs (DAGs). 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 (ℓ) 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. Sparse compositionality implies *hierarchy, modularity and reuse of the modules, including transfer learning*

Principle II: **Genericity**

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. Once genericity holds, gradient-based optimization is feasible.

There are Additional Principles of Deep Learning!

…lot more to do!