Emergence, Scaling Laws, and Compositional Reasoning: Insights via the SGD Dynamics

Eshaan Nichani
Princeton University
Cargese 2025

Joint work with: Zixuan Wang, Yunwei Ren, Alex Damian, Jason D. Lee (Princeton), Alberto Bietti (Flatiron Institute), Denny Wu (NYU/Flatiron), Daniel Hsu (Columbia)

The LLM Revolution

Large Language Models (LLMs) have demonstrated impressive empirical successes on a range of complex tasks.

ChatGPT

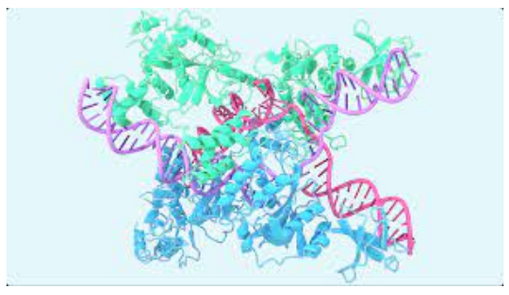
GPT 4o

GitHub Copilot

Coding/Mathematics

Vision

Reinforcement Learning

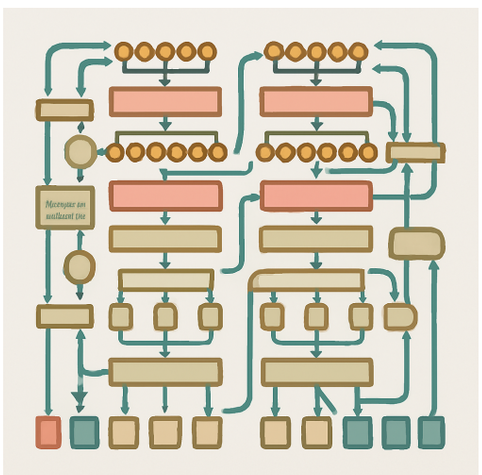


AI for Science

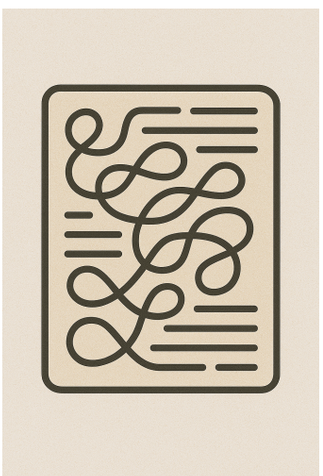
Q: When (and how) do LLMs learn to perform such tasks during pretraining?

LLM Pretraining

Vast amounts of diverse training data



Sophisticated model architectures



Elaborate training recipes

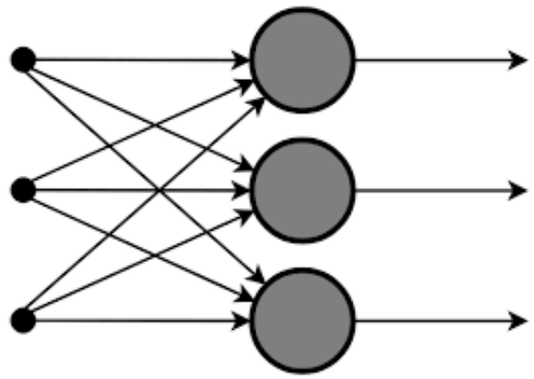
Intractable to analyze optimization dynamics theoretically!

Existing theory thus largely restricted to expressive capabilities of transformers

- Provides limited insight on when (and how) capabilities are learned during training, what solutions are learned, etc.

Sandbox Tasks

Approach: Define minimal, sandbox tasks which formalize core LLM phenomena as statistical problems



$$w_{t+1} = w_t - \eta g_t$$

Well-specified data generating process	Simple (shallow) model architectures	Simplified versions of gradient descent

Can prove guarantees about the optimization dynamics!

Goal: Analysis of optimization dynamics reveals insights about original phenomenon

Outline

Part I: How do transformers learn to perform **multi-step reasoning?**

Part II: How do **emergence & scaling laws** arise via SGD?

Part I: Learning Compositional Functions with Transformers from Easy-to-Hard Data

Zixuan Wang*, **EN***, Alberto Bietti, Alex Damian, Daniel Hsu, Jason D. Lee, Denny Wu

COLT 2025

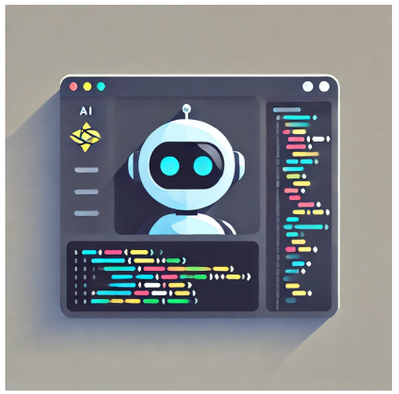
Motivation: Multi-step reasoning

LLMs solve complex tasks which require combining multiple reasoning steps.

Multi-hop QA in natural language

Prompt: "John plays quarterback.... The football game is on Sunday.... On what day does John play?"

Coding/Mathematics



Practical reasoning tasks often require composing **contextual information** with **parametric knowledge** (i.e global facts).

Contextual Knowledge	(John → quarterback), (football→Sunday)	Definitions, variable assignments
Parametric Knowledge	(quarterback → football)	Which theorem to use / function to call

Compositional Reasoning

Motivating Questions:

- What is a **model** for compositional reasoning tasks with both contextual and parametric knowledge?
- How do transformers efficiently **express** such tasks?
- Can transformers **learn** such reasoning tasks via **gradient-based training**?

Our contributions:

- We introduce the **k-fold composition task**, and prove that there exists a $O(\log k)$-depth transformer which solves the task.
- We prove that **curriculum learning** is both **necessary and sufficient** to learn the k-fold composition via gradient-based training.

Task: k-fold composition

Prompt: "John plays quarterback. Sarah plays shortstop. The football game is on Sunday. The softball game is on Thursday. On what day does John play?"

Multi-step reasoning tasks can be viewed as function composition:

Input (Contextual knowledge):

John → quarterback, Sarah → shortstop; football→Sunday, softball→Thursday; John

σ_1: Names → Positions

σ_2: Sports → Days

x = John

Parametric Knowledge: (quarterback→football; shortstop→softball)

π: Positions → Sports

Target function: $f(\sigma_1, \sigma_2, x) = (\sigma_2 \circ \pi \circ \sigma_1)(x)$

Definition (k-fold composition):

Parametric knowledge: $\pi := (\pi_1, \ldots, \pi_k) \in (S_N)^k$ is tuple of k hidden permutations.

Input: $\sigma := (\sigma_1, \ldots, \sigma_k) \in (S_N)^k$ is tuple of k input permutations, $x \in [N]$ is starting index. (Input space $(\sigma, x) \in \mathcal{X} := (S_N)^k \times [N]$)

Output: $f_\pi : \mathcal{X} \to [N]$ is defined as $f_\pi(\sigma, x) := (\sigma_k \circ \pi_k \circ \sigma_{k-1} \circ \cdots \circ \sigma_1 \circ \pi_1)(x)$

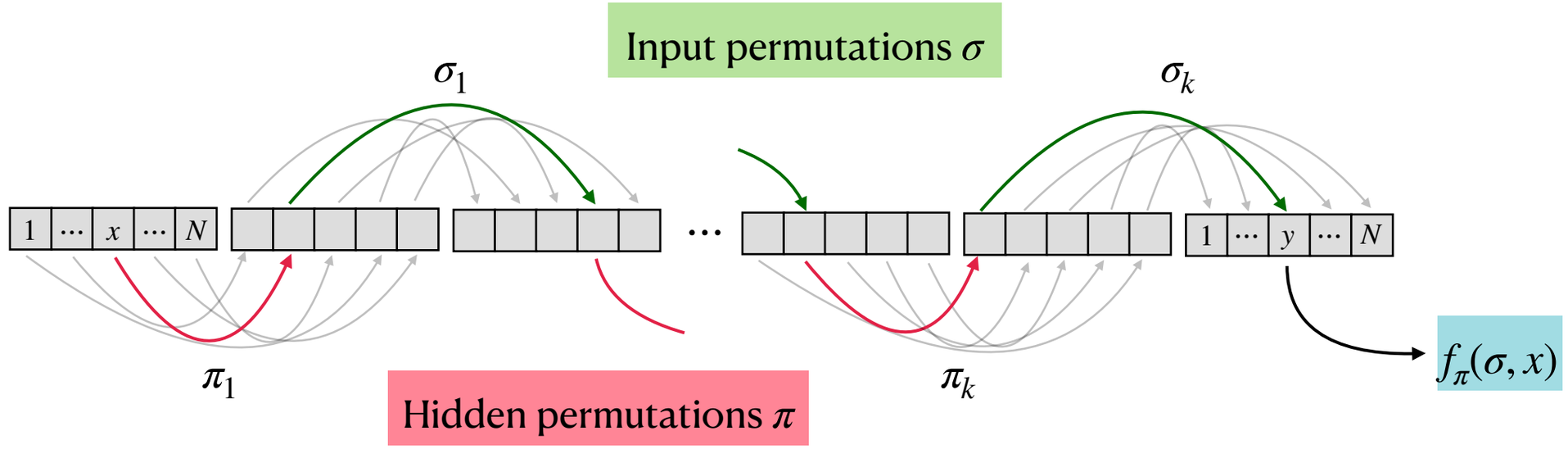
Task: k-fold composition

Definition (k-fold composition):

Parametric knowledge: $\pi := (\pi_1, \ldots, \pi_k) \in (S_N)^k$ is tuple of k hidden permutations.

Input: $\sigma := (\sigma_1, \ldots, \sigma_k) \in (S_N)^k$ is tuple of k input permutations, $x \in [N]$ starting index

Output: $f_\pi : \mathcal{X} \to [N]$ is defined as $f_\pi(\sigma, x) := (\sigma_k \circ \pi_k \circ \sigma_{k-1} \circ \cdots \circ \sigma_1 \circ \pi_1)(x)$



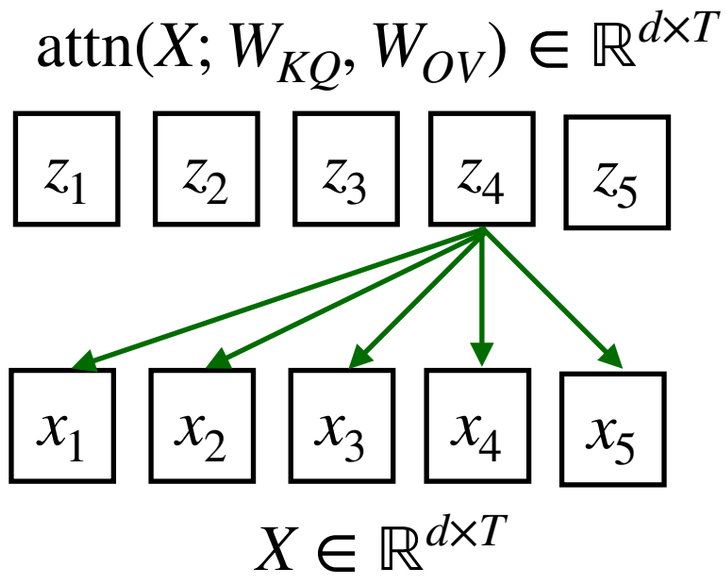
- Hidden π defines a *function class* $\mathcal{F} := \{f_\pi : \pi \in (S_N)^k\}$.
- Each permutation σ_i represented using N tokens. Sequence length is $T := kN$.
- Connection to pointer chasing (Papadimitriou & Sipser, 1984). Related tasks include automata simulation (Liu et al., 2023), k-hop induction head (Sanford et al., 2024), path-star task (Bachmann & Nagarajan, 2024), cycle task (Abbe et al., 2024).

Model: Multi-layer Transformer

Self-attention head $\text{attn}(\,\cdot\,; W_{KQ}, W_{OV}) : \mathbb{R}^{d\times T} \to \mathbb{R}^{d\times T}$**:**

- Parametrized by key-query and value matrices $W_{KQ}, W_{OV} \in \mathbb{R}^{d\times d}$.
- Each output is convex combination of values $W_{OV}x_j$:

$$\text{attn}(X; W_{KQ}, W_{OV})_i = \sum_{j=1}^{T} \frac{\exp(x_j^T W_{KQ} x_i)}{\sum_{j'=1}^{T} \exp(x_{j'}^T W_{KQ} x_i)} W_{OV} x_j$$



Multi-layer transformer composes multiple self-attention heads in series

- Embedding layer ϕ maps (token, position) to $\mathbb{R}^d$.
- Output of self-attention added to previous layer:
- Readout layer Ψ maps $X^{(L)}$ to N-dimensional output at each position.

$$X_i^{(0)} = \phi(s_i, i)$$

$$X^{(\ell)} = X^{(\ell-1)} + \text{attn}(X^{(\ell-1)}; W_{KQ}^{(\ell)}, W_{OV}^{(\ell)})$$

$$\text{TF}(s_{1:T}) = \Psi X^{(L)}$$

Expressivity via Parallelism

Q: Can transformers efficiently compute f_π?

A: Yes! $O(\log k)$ layers suffice.

Theorem: Let k be a power of 2. There exists an embedding function ϕ with $d = \tilde{O}(kN)$ such that, for any $\pi \in (S_N)^k$, there exists an $L = \log_2 k + 1$ depth transformer which exactly expresses the k-fold composition task f_π.

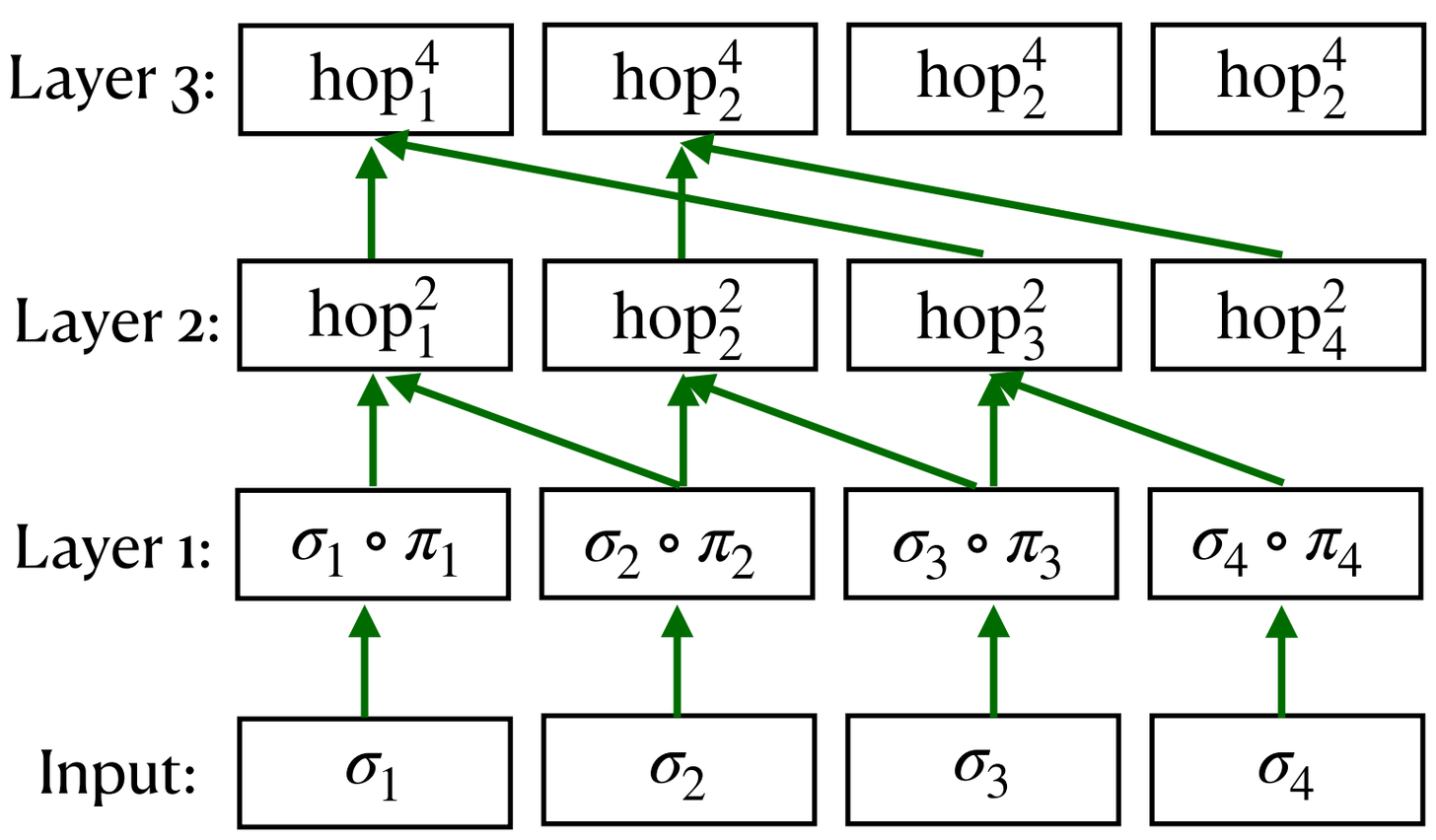
Key Idea: **Parallelism** (i.e build a binary tree)

- Layer 1 encodes hidden π_i's, and computes $\sigma_i \circ \pi_i$.
- Layer 2 composes $\sigma_{i+1} \circ \pi_{i+1}$ and $\sigma_i \circ \pi_i$ to get

$$\text{hop}_i^2 := \sigma_{i+1} \circ \pi_{i+1} \circ \sigma_i \circ \pi_i$$

- Layer 3 composes hop_i^2 and hop_{i+2}^2 to get hop_i^4

... and so on.



Hardness of Learning via SGD

Q: Can the k-fold composition f_π be learned from *samples* via gradient descent?

Data Distribution: (σ, x) sampled uniformly from $\mathcal{X}$.

A: No! *Statistical Query* lower bound.

Statistical Query (SQ) Model:

- Learner specifies a query $q : \mathcal{X} \times [N] \to \mathbb{R}$ and tolerance τ.
- SQ Oracle responds with $\hat{q}$, noisy estimate to q, satisfying

$$|\hat{q} - \mathbb{E}_{\sigma,x}[q(\sigma, x, f_\pi(\sigma, x))]| \leq \tau.$$

Theorem (SQ Lower Bound): Any SQ learner must either make $\geq N^{\Omega(k)}$ queries or use tolerance $\tau \leq N^{-\Omega(k)}$

- Gradient descent is an SQ algorithm.
- Concentration heuristic $\tau \asymp n^{-1/2} \implies$ SGD requires exponential compute or samples.
- Proof Idea: Construct an exponentially large subset of $\mathcal{F}$ which is *nearly orthogonal* under inner product $\langle f_\pi, f_\rho \rangle := \mathbb{E}_{\sigma,x}[f_\pi(\sigma, x) = f_\rho(\sigma, x)] - 1/N$.

Curriculum Learning

Curriculum learning — training on *easier* examples of k'-fold data for $k' \leq k$ — can lead to efficient learning.

Parallel construction motivates natural easy-to-hard curriculum.

- For $i \in [k], j \in [N]$, define $\mathrm{hop}_i^r(\sigma) := \sigma_{i+r-1} \circ \pi_{i+r-1} \circ \cdots \circ \sigma_i \circ \pi_i$.

Algorithm 1 (Curriculum Learning):

- Initialize all W_{OV} matrices to ground truth, and all W_{KQ} matrices to $\mathbf{0}_{d\times d}$.
- In stage ℓ: Consider loss for predicting labels $\mathrm{hop}_i^{2^{\ell-1}}(\sigma)$, and take 1 gradient step on all the W_{KQ} matrices.

Theorem (GD upper bound with curriculum):

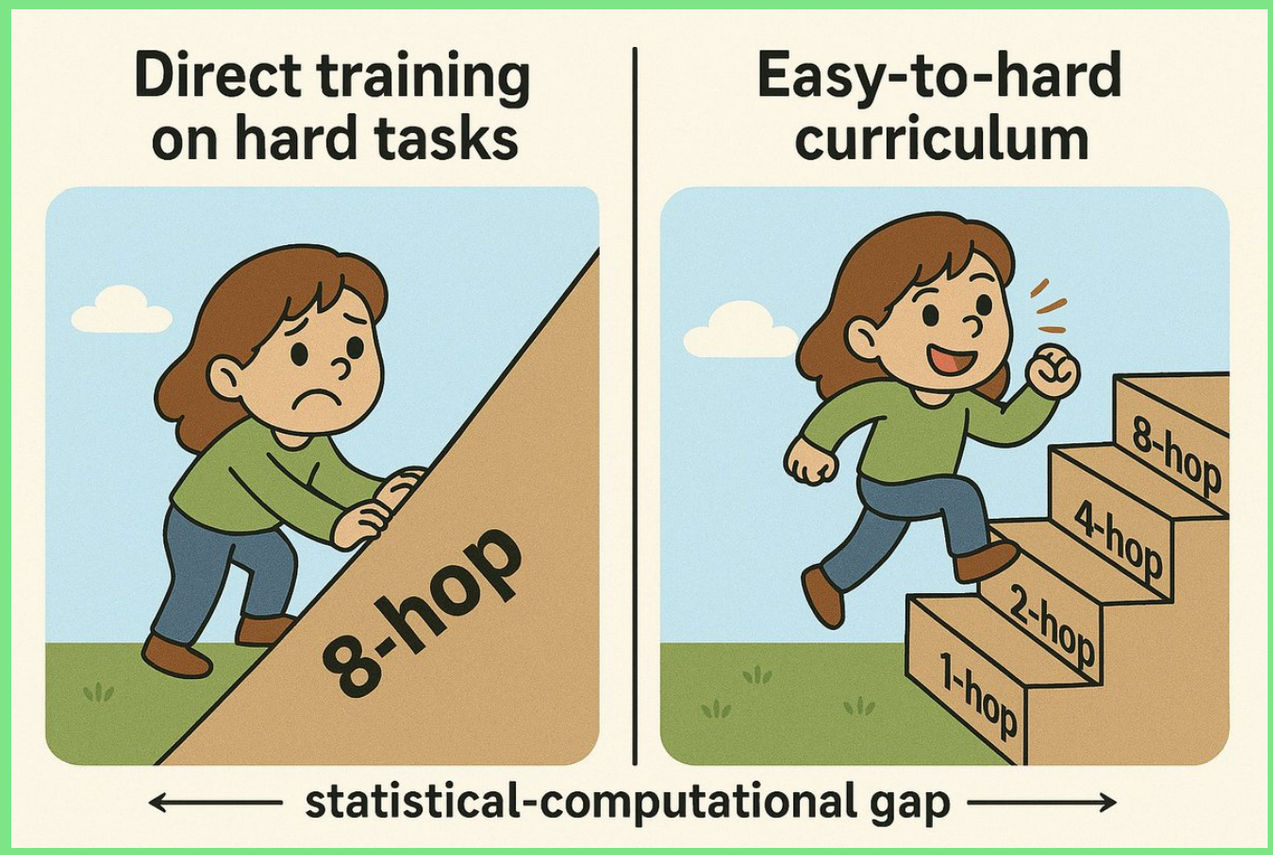
Assume $k = 2^{L-1}$, if the sample size $n \geq \tilde{\Omega}(k^4 N^6)$, the output of gradient descent with curriculum learning learns the k-fold composition task.

Proof Idea: $W_{KQ}^{(\ell)}$ goes from $\mathbf{0}$ to ground truth in the ℓth stage.

Naive: $n \asymp N^{\Omega(k)}$ **(Exponential)** $\Longrightarrow$ Curriculum: $n \asymp \mathrm{poly}(N, k)$ **(Polynomial)**

Curriculum Learning

Easy examples / intermediate supervision is essential for complex tasks!



Naive: $n \asymp N^{\Omega(k)}$ **(Exponential)** $\Longrightarrow$ Curriculum: $n \asymp \mathrm{poly}(N, k)$ **(Polynomial)**

Mixed Training

In practice, designing an explicit curriculum is challenging.

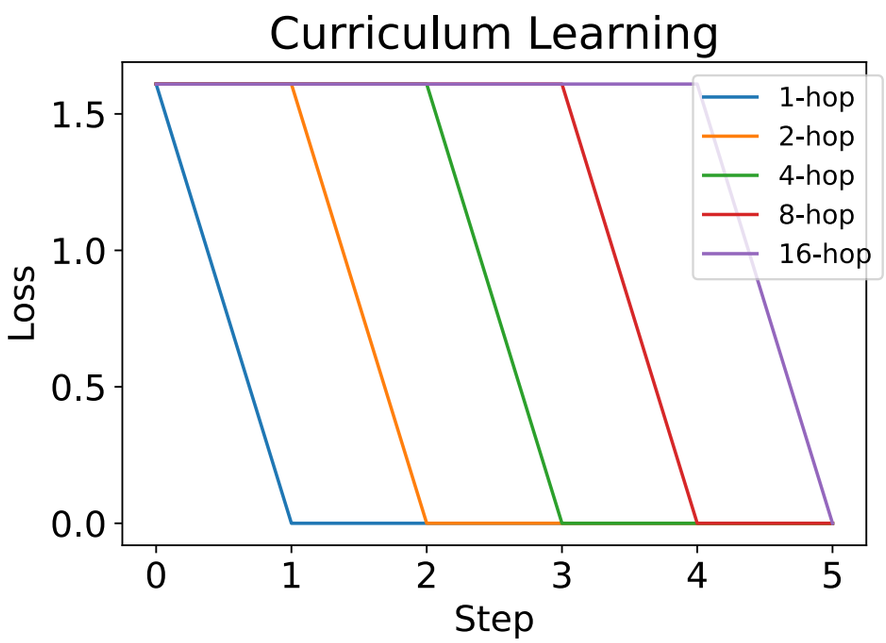
- Hard to ensure increasing difficulty, facing catastrophic forgetting
- Instead, practitioners use train on a **mixture** of examples with different difficulties.

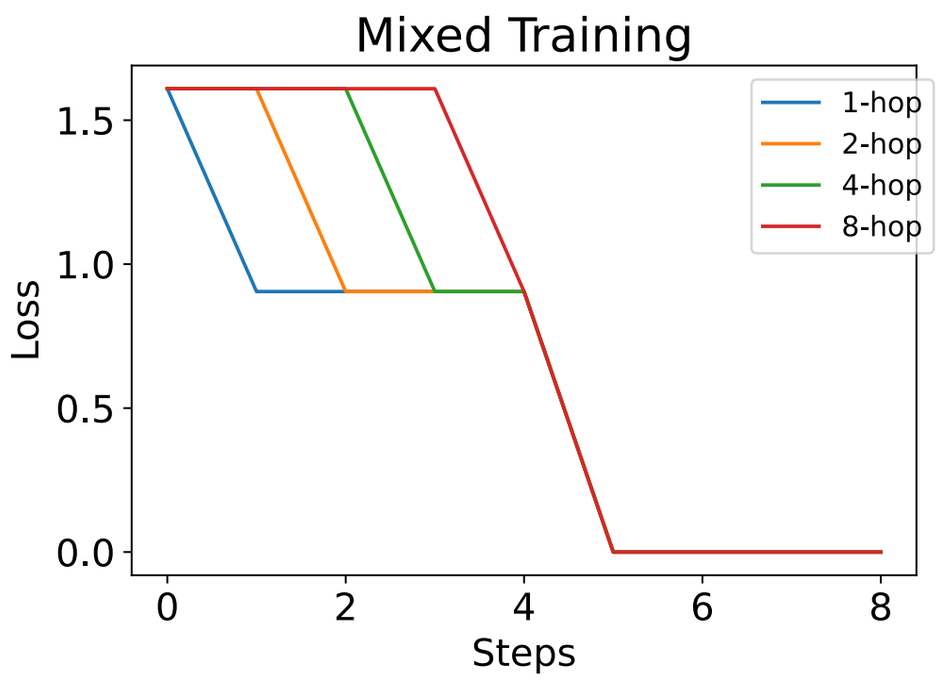
Theorem (informal): Training on a mixture of $2^{\ell-1}$-hop data for $\ell \in [L]$ learns the k-fold composition in $n = \text{poly}(N, k)$ samples.

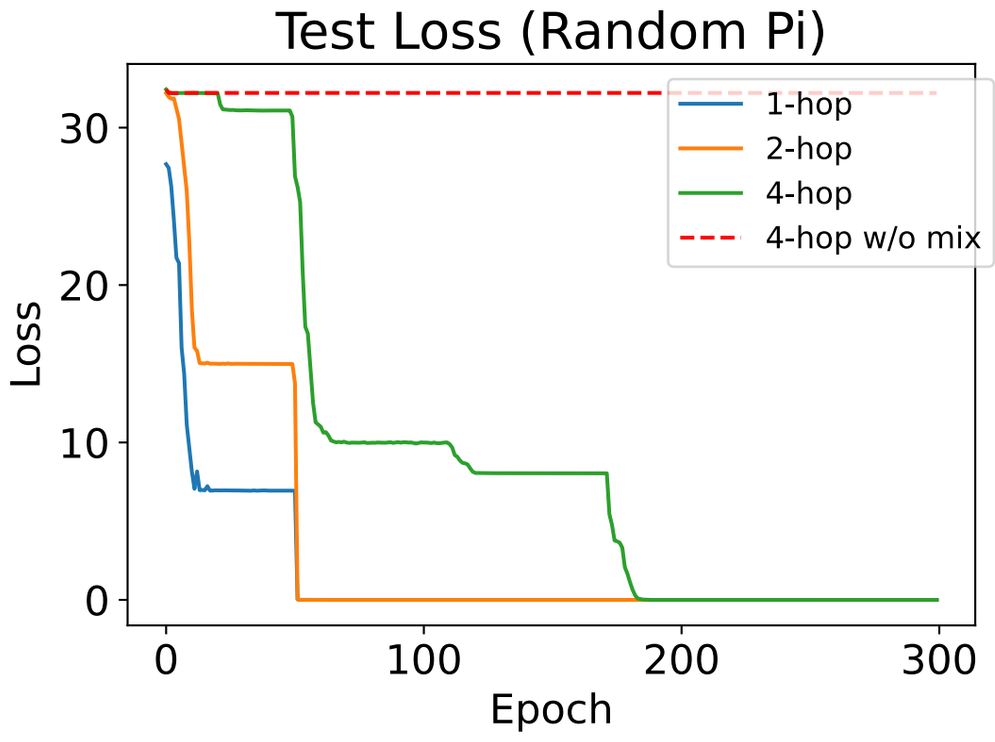
Intuition: If layer ℓ (the $2^{\ell-1}$-fold function) is not learned, no signal to layers $\ell' > \ell$.

- Data mixture **implicitly induces curriculum learning**

Simulations







Remarks

- k-fold composition exhibits a **statistical-computational gap**.
  - Information theoretic sample complexity is $n \asymp kN \log N$,
  - Computationally efficient SQ learner requires $n \gtrsim N^{\Omega(k)}$.
- Mirrors k-**sparse parity** in d dimensions, which requires d^k samples/compute.
- While k-parity is expressible by 1-layer transformer, k-fold composition requires larger depth $\Longrightarrow$ fundamentally a compositional task!

Embedding Dimension:

- Other parallel constructions (Liu et al., 2023; Sanford et al., 2024) require $d = \tilde{O}(1)$ embedding dimension.
- Since $\log |\mathscr{F}| = \tilde{\Theta}(kN)$, with fixed embeddings we need $d = \text{poly}(Nk)$.
- If embedding is trainable, construction with $d = \tilde{O}(1)$ exists...but challenging to show learnability via GD

Takeaways

Tasks expressible by transformers are not necessarily efficiently learnable via GD.

- Depends on underlying data distribution (easy-to-hard data), loss function (intermediate supervision), etc.

Future Directions:

- Learning guarantees (upper bounds) or characterizing complexity of learning via GD (lower bounds) for more general compositional / algorithmic tasks.
- What are other ways (beyond curriculum/intermediate supervision) where the data distribution effects complexity of SGD learning?

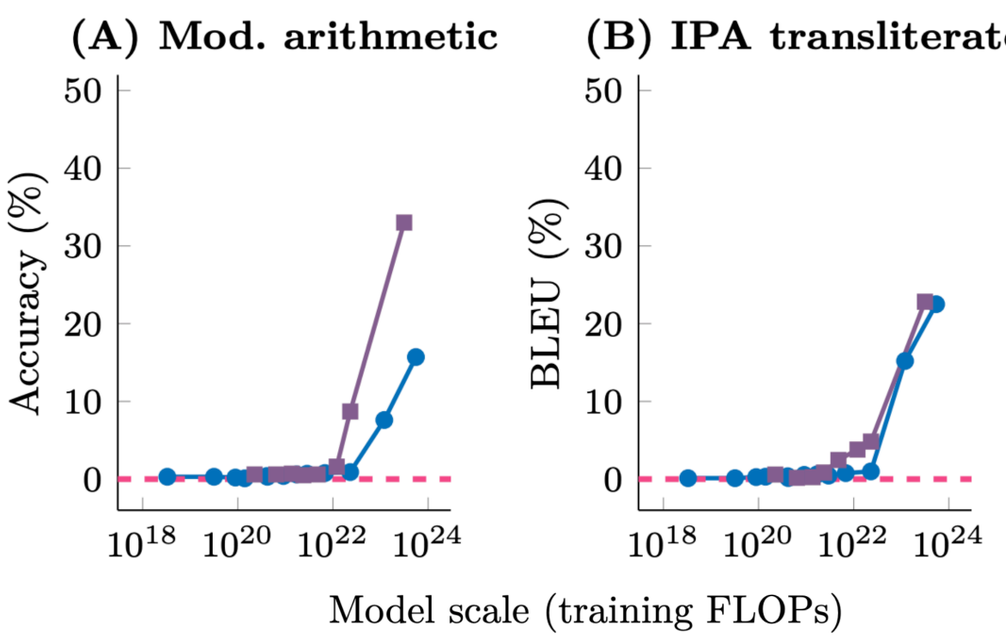
Part II: Emergence and scaling laws for SGD learning of shallow neural networks

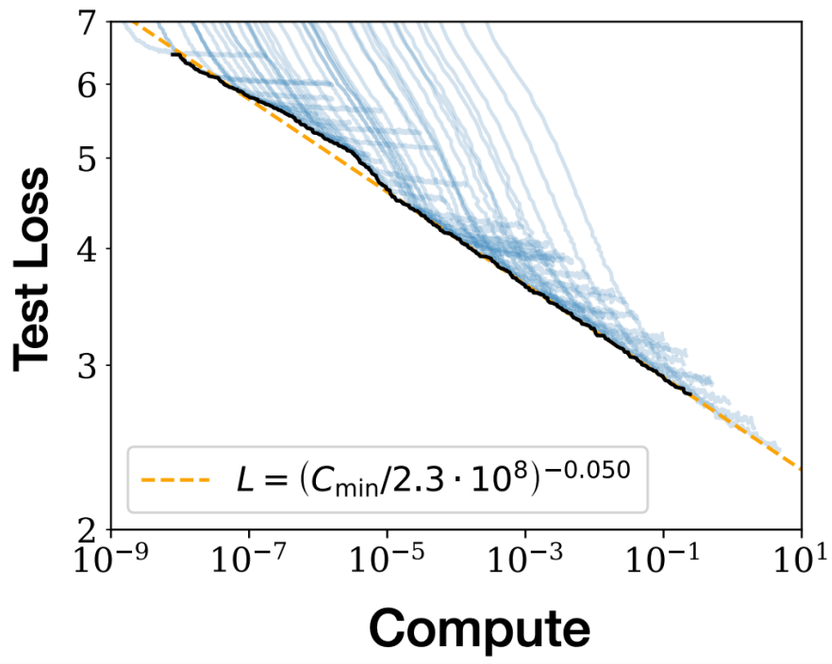
Yunwei Ren*, **EN***, Denny Wu, Jason D. Lee

Under Review, 2025

Emergence & Scaling Laws

LLMs are pretrained on vast amounts of data, with examples requiring skills/capabilities beyond simply multi-step reasoning.





Emergent Capabilities

(Wei et al., 2022):

Learning of a single task (skill) exhibits sharp transition as model scale increases

Neural Scaling Laws

(Kaplan et al., 2020, Hoffman et al., 2022)

Increasing compute & data leads to predictable, power-law decay in pretraining loss

Skill acquisition time is unpredictable

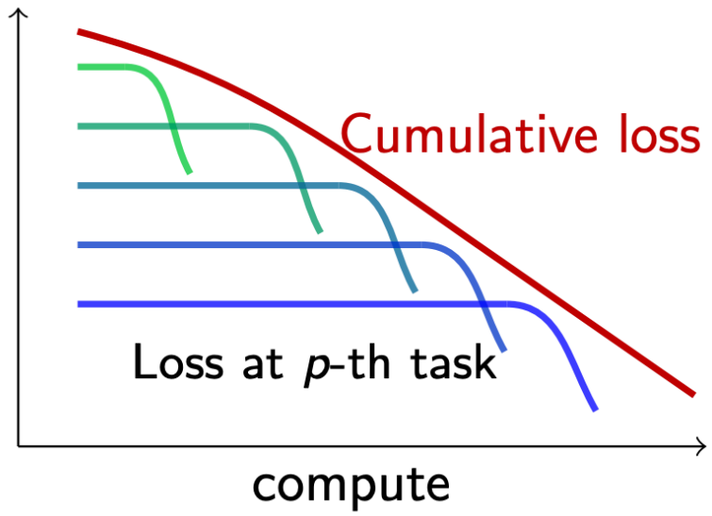
$$\text{Loss} \propto (\#\ \text{params})^{-\alpha} + (\#\ \text{data points})^{-\beta}$$

Emergence & Scaling Laws

Q: How to reconcile the **emergent behavior** in skill acquisition with the **smooth power-law decay** in the cumulative loss?

Additive Model Hypothesis (Michaud et al., 2024; Nam et al., 2024):

- Cumulative loss can be decomposed into many distinct *skills*.
- Learning of each individual skill exhibits *emergence*.
- *Juxtaposition* of many emergent learning curves at *varying timescales* leads to predictable power law decay in the loss.



Our contributions:

- Theoretical justification for additive model hypothesis in **gradient-based feature learning** of shallow neural networks.
- Novel analysis for learning extensive width neural network.

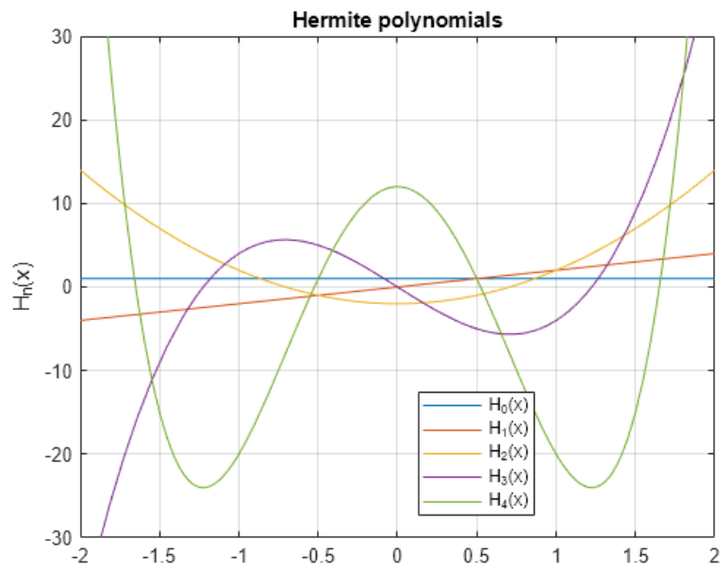
Background: Single-index model

Target: $f^*(\mathbf{x}) = \sigma_*(\langle \mathbf{x}, \theta \rangle)$, where $\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)$.

- Requires learning direction $\theta \in \mathbb{R}^d$ and link function $\sigma_* : \mathbb{R} \to \mathbb{R}$.
- Efficient learner must adapt to low-dimensional structure.
- Canonical model for feature learning in two-layer neural networks

Complexity of learning is governed by the **information exponent:**

- **Hermite polynomials** $\{He_j(z)\}_{j \geq 0}$ form an orthonormal basis of $L^2(\mathbb{R})$ wrt Gaussian measure.
- **Hermite expansion:** $\sigma_*(z) = \sum_{j \geq 0} \alpha_j^* He_j(z)$, where $\alpha_j^* = \mathbb{E}_{z \sim \mathcal{N}(0,1)}[\sigma_*(z) He_j(z)]$.



Definition: The information exponent of σ_* is $k = \mathrm{IE}(\sigma_*) = \min\{k \geq 0 : a_k^* \neq 0\}$

Background: Single-index model

IE captures *signal* in gradient at *random initialization*.

$$f^*(\mathbf{x}) = \sigma_*(\langle \mathbf{x}, \theta \rangle), \mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)$$

$$\sigma_*(z) = \sum_{j \geq 0} \alpha_j^* He_j(z)$$

$$k = \min\{k \geq 0 : a_k^* \neq 0\}$$

Example: Learner $f(\mathbf{x}) = \sigma_*(\langle \mathbf{x}, \mathbf{w} \rangle)$, population correlation loss $\mathscr{L}(\mathbf{w}) = -\mathbb{E}_{\mathbf{x}}[\sigma_*(\langle \mathbf{x}, \mathbf{w} \rangle)\sigma_*(\langle \mathbf{x}, \theta \rangle)]$

$$\begin{aligned}
-\nabla_w \mathscr{L}(\mathbf{w}) &= \nabla_w \mathbb{E}[\sigma_*(\langle \mathbf{x}, \mathbf{w} \rangle)\sigma_*(\langle \mathbf{x}, \theta \rangle)] \\
&= \nabla_w \sum_{j \geq k} \alpha_j^2 \langle \mathbf{w}, \theta \rangle^j && \text{Hermite expansion formula (when } \mathbf{w}, \theta \text{ unit norm)} \\
&= \theta \sum_{j \geq k} j\alpha_j^2 \langle \mathbf{w}, \theta \rangle^{j-1} && d^{-(j-1)/2} \text{ at random initialization} \\
&\asymp d^{-(k-1)/2}\theta
\end{aligned}$$

Theorem (Ben Arous et al., 2021; Bietti et al., 2022; Damian et al., 2023, etc.):

A two-layer neural network can learn single index f_* with information exponent k using $n \simeq T \gtrsim d^{\Theta(k)}$ samples/steps of SGD

Emergence in Gradient-Based Feature Learning

IE captures *signal* in gradient at *random initialization*:

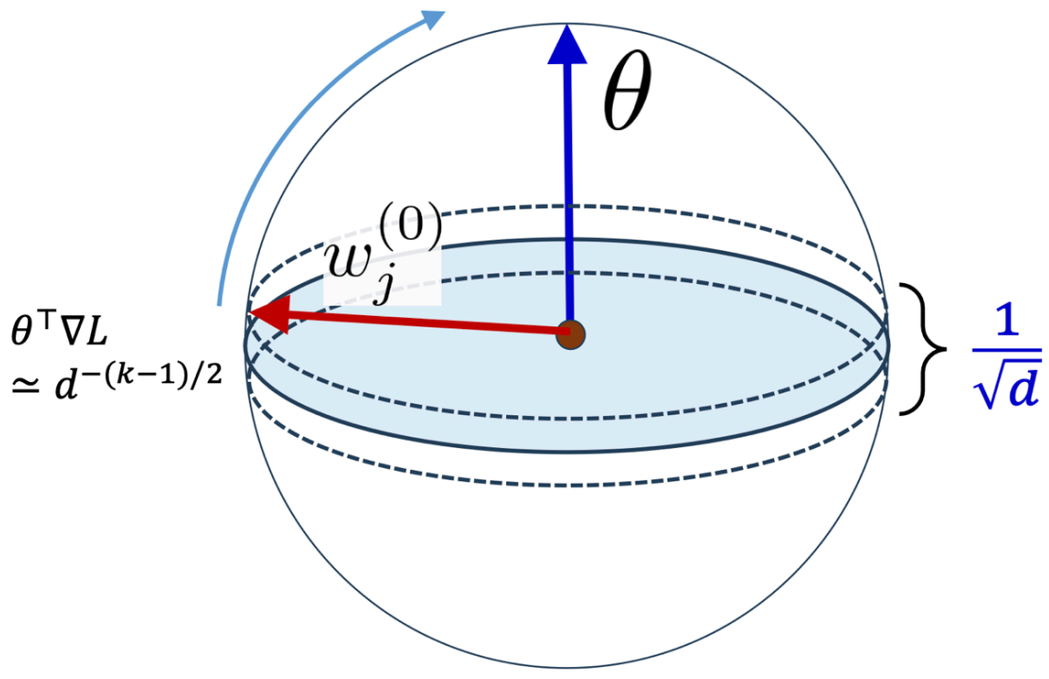
$$-\nabla_w \mathscr{L}(\mathbf{w}) \asymp d^{-(k-1)/2}\theta$$

$$f^*(\mathbf{x}) = \sigma_*(\langle \mathbf{x}, \theta \rangle), \mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)$$

$$\sigma_*(z) = \sum_{j \geq 0} a_j^* He_j(z)$$

$$k = \min\{k \geq 0 : a_k^* \neq 0\}$$

Most samples in SGD are used to escape the high-entropy "equator."



- **Search Phase:** Online SGD exhibits a loss plateau up to $T \asymp d^{k-1}$ steps
- **Descent Phase:** Loss sharply decreases in $T = \tilde{\Theta}(1)$ steps

Learning of a single-index model with online SGD exhibits an **emergent learning curve!**

Setting: Extensive Width Neural Network

Target Function: Width P two-layer neural network

$$f_*(\mathbf{x}) = \sum_{p=1}^{P} a_p \cdot \sigma(\langle \mathbf{x}, \mathbf{v}_p^* \rangle), \quad \sum_{p=1}^{P} a_p^2 = 1, \{\mathbf{v}_p^*\}_{p \in [P]} \text{ orthonormal}$$

- Each single-index task corresponds to a distinct *skill*.
- σ is even, with IE $k > 2$ $\implies$ Emergent learning curve for each individual skill
- Varying second layer $a_1 \geq a_2 \geq \cdots \geq a_P$ $\implies$ separation in length of search phase
- Example of (well-specified) multi-index model.

Remark:

- Prior works on scaling laws are either restricted to linear regression (Bordelon et al., 2024; Paquette et al., 2024; Lin et al., 2024), or explicitly decompose cumulative loss into sum of losses per skill (Michaud et al., 2024; Nam et al., 2024)
- We model more challenging *feature learning* setting where tasks are not decoupled.

Setting: Extensive Width Neural Network

Target Function: Width P two-layer neural network

$$f_*(\mathbf{x}) = \sum_{p=1}^{P} a_p \cdot \sigma(\langle \mathbf{x}, \mathbf{v}_p^* \rangle), \quad \sum_{p=1}^{P} a_p^2 = 1, \{\mathbf{v}_p^*\}_{p \in [P]} \text{ orthonormal}$$

Technical Challenges:

- **Extensive width:** $P \propto d^\gamma$ for $\gamma = \Theta(1)$ (i.e large number of tasks)
  - $P \to \infty$ necessary to obtain smooth scaling law
  - Most prior works for *multi-index models* assume $P = \Theta(1)$.
- **Large condition number:** $\kappa = a_{\max}/a_{\min} = \text{poly}(P)$.
  - Covers power law decay in second layer ($a_p \propto p^{-\beta}, \beta > 0$)
- **Single-phase SGD learning** with MSE loss
  - Layer-wise training with correlation loss alters scaling law.

Complexity of SGD Learning

Student Model: *2-homogeneous* two-layer neural network (well specified)

$$f(\mathbf{x}) = \sum_{i=1}^{m} \|\mathbf{v}_i\|_2^2 \sigma(\langle \mathbf{x}, \mathbf{v}_i/\|\mathbf{v}_i\| \rangle).$$

(2-homogeneity assumption motivated by ReLU)

Online SGD training: At time t, sample $x_t \sim \mathcal{N}(0, \mathbf{I}_d)$, and compute update

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) - \eta \nabla_{\mathbf{v}_i} (f_*(\mathbf{x}_t) - f(\mathbf{x}_t))^2.$$

Theorem: Assume $P \ll d^{0.1}$, $P_* \leq P$. If we train student NN with $m = \tilde{\Theta}(P_*)$ neurons using online SGD with learning rate $\eta \asymp \frac{a_{P_*}}{d^{k/2}\mathrm{poly}(P)}$, then whp:

- If $p \leq P_*$, a student neuron aligns with $\mathbf{v}_p^*$ at time $T_p \asymp a_p^{-1} \eta^{-1} d^{k/2-1}$
- All directions up to P_* are learned at time $n \asymp T \asymp a_{P_*}^{-2} d^{k-1} \mathrm{poly}(P)$

Complexity of SGD Learning

Theorem: Assume $P \ll d^{0.1}$, $P_* \leq P$. If we train student NN with $m = \tilde{\Theta}(P_*)$ neurons using online SGD with learning rate $\eta \asymp \frac{a_{P_*}}{d^{k/2}\mathrm{poly}(P)}$, then whp:

- If $p \leq P_*$, a student neuron aligns with $\mathbf{v}_p^*$ at time $T_p \asymp a_p^{-1}\eta^{-1}d^{k/2-1}$
- All directions up to P_* are learned at time $n \asymp T \asymp a_{P_*}^{-2}d^{k-1}\mathrm{poly}(P)$

Remarks:

- d-dependence matches complexity of online SGD from (Ben Arous et al., 2021)
- $m \asymp P$ neurons and $n \asymp T \asymp a_{\min}^{-2}d^{k-1}\mathrm{poly}(P)$ samples needed to obtain small population loss (i.e learn all tasks)
  - Prior works (Li et al., 2021, Oko et al., 2024) require sample size/compute to be *exponential* in condition number $n, m \asymp \exp(\kappa)$!
  - Our learning procedure is *single-stage*. Does not require *reinitialization* (Ge et al., 2021) or *Stiefel constraint* (Ben Arous et al., 2024)

A Neural Scaling Law for Feature Learning

Corollary (Emergence & Scaling Laws): Assume $a_p \propto p^{-\beta}$ for $\beta > 1/2$. Then:

- **Emergence:** pth task is learned at time $T_p \sim \eta^{-1} d^{k/2-1} p^{\beta}$
- **Scaling Law:** population MSE exhibits power law decay:

$$\mathscr{L}(t) \sim m^{1-2\beta} \vee \left(\frac{\eta t}{d^{k/2-1}} \right)^{\frac{1-2\beta}{\beta}}.$$

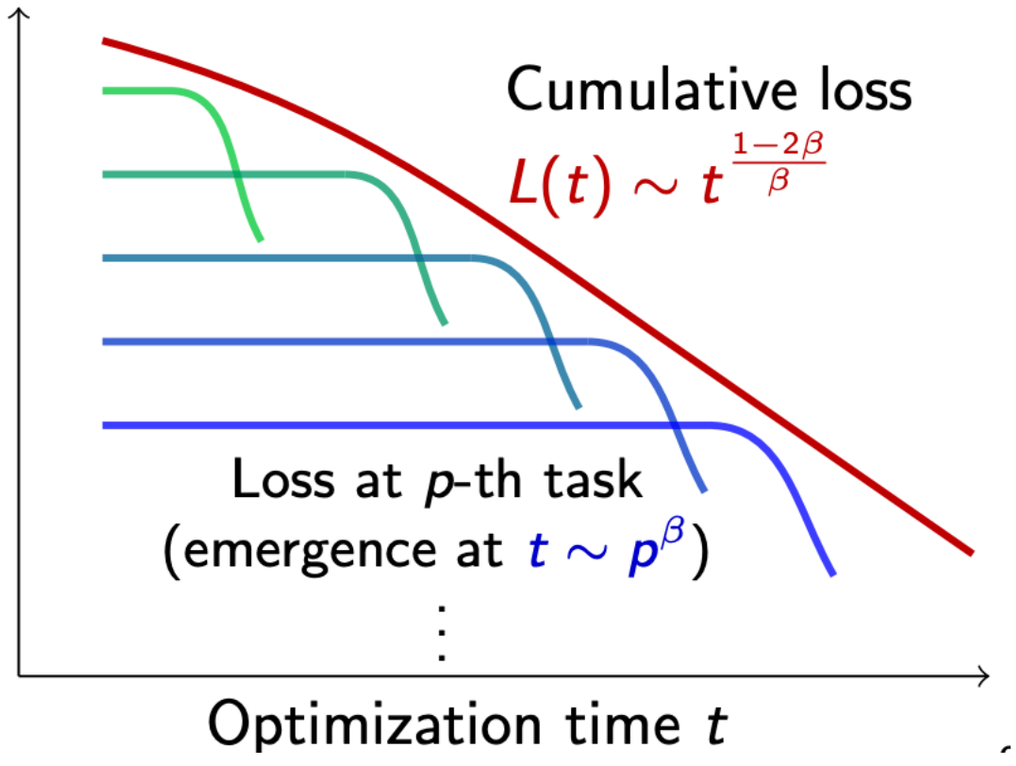
Intuition:

- Loss decomposes as $\mathscr{L}(t) \approx \sum_{p=1}^{P} a_p^2 \mathbf{1}(t \le T_p)$

  $\Longrightarrow \mathscr{L}(T_p) \approx \sum_{s=p}^{P} a_s^2 \approx \int_p^{\infty} s^{-2\beta} ds \asymp p^{1-2\beta}$

- $m < P$ student neurons reaches

  $\sum_{s>m} a_s^2 \asymp m^{1-2\beta}$ error

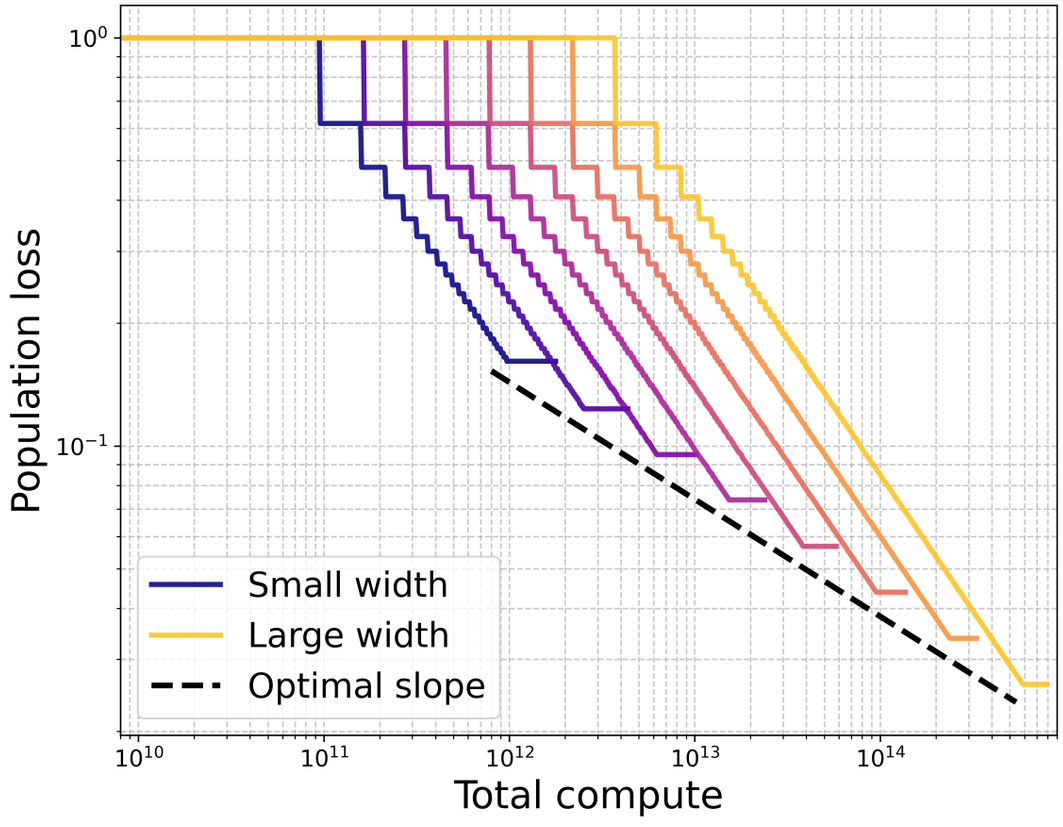


A Neural Scaling Law for Feature Learning

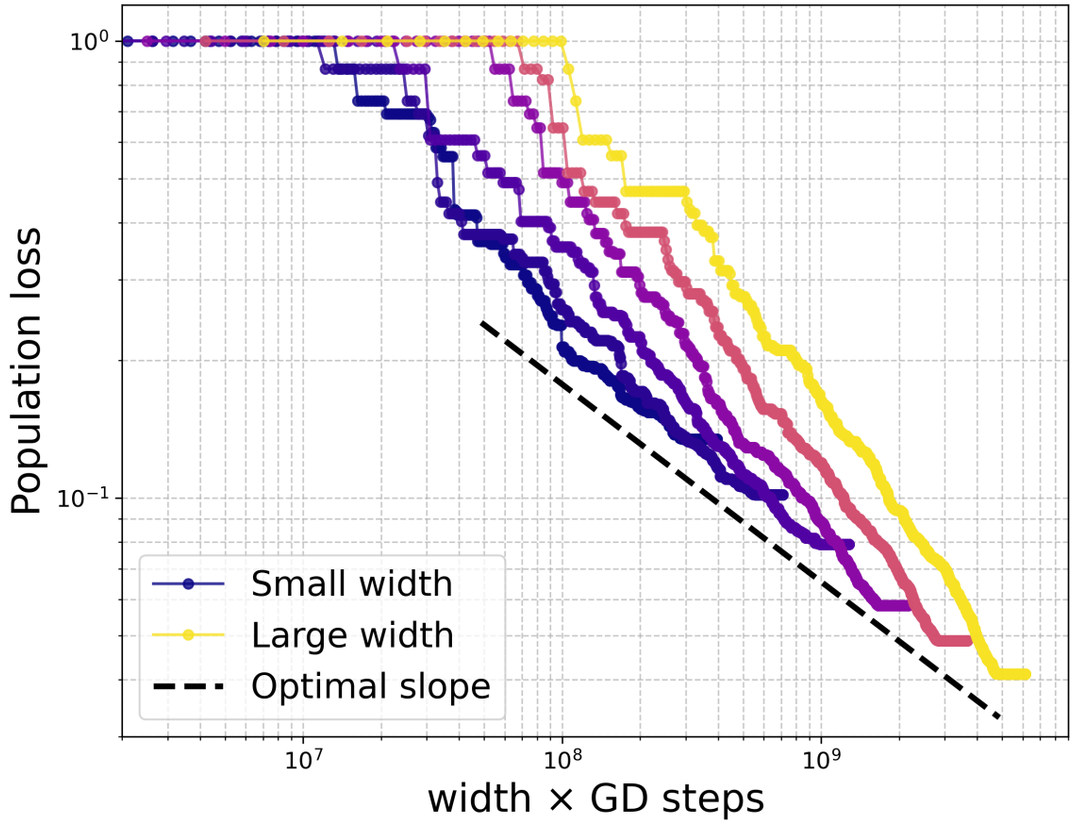
$$\mathscr{L}(t) \sim m^{1-2\beta} \vee \left(\frac{\eta t}{d^{k/2-1}}\right)^{\frac{1-2\beta}{\beta}}$$

Approximation barrier: $m^{1-2\beta}$ (Determined by student network width)

Optimization error: $(\eta t d^{1-k/2})^{\frac{1-2\beta}{\beta}}$ **(**Determined by number of online SGD steps)



Theoretical scaling law



Empirical scaling law

Remark on Discretization

Continuous time (constant η) rate can be misleading

$$\mathscr{L}(t) \sim m^{1-2\beta} \vee \left(\frac{\eta t}{d^{k/2-1}} \right)^{\frac{1-2\beta}{\beta}}$$

- η scaling with $a_{\min}$ is pessimistic.
- Does not match minimax rate for weak ℓ_p ball (Johnstone, 2017)

Adaptive learning rate for pth task:

- Say we only care about learning top p neurons
- "Optimal" learning rate: $\eta \asymp \frac{a_p}{d^{k/2}\mathrm{poly}(P)}$.
- Direction $\mathbf{v}_p^*$ now learned at time $T_p \asymp \eta^{-1} p^{\beta} d^{k/2-1} = p^{2\beta} d^{k-1}\mathrm{poly}(P)$.

Scaling law: under this learning rate for the first p neurons,

$$\mathscr{L}(n, m) \asymp m^{1-2\beta} \vee \left(\frac{n}{d^{k-1}\mathrm{poly}(P)} \right)^{\frac{1-2\beta}{2\beta}}$$

- Matches known rates!
- Is not anytime (# of SGD iterations n must be prespecified; η chosen in terms of n)

Proof Sketch

Naive analysis (correlation loss)

- Each student neuron $\mathbf{v}_p$ evolves independently
- Converges to teacher neuron maximizing correlation at initialization:

$$\pi(p) := \arg\max_q a_q \langle \mathbf{v}_p, \mathbf{v^*}_q \rangle^{k/2}.$$

- Since $a_1 \gg a_P$, for small direction $\mathbf{v}_P^*$ to be learned one needs $m \gtrsim P^{\Omega(a_{\min}^{-1})}$.

Proof Sketch

Simultaneous training with MSE:

- Let $\mathbf{v}_{\iota(p)}$ denote the neuron which eventually converges to $\mathbf{v}_p^*$, and let $m_{p,q} := \langle \frac{\mathbf{v}_p}{\|\mathbf{v}_p\|}, \mathbf{v}_q^* \rangle$ measure the overlap.
- When $m_{\iota(p),q}$ for $(q \neq p)$ is small, learning of different directions can be approximately decoupled.
- Norm growth of $\mathbf{v}_{\iota(p)}$ is coupled to directional convergence: once $m_{\iota(p),p} \approx 1$, $\|\mathbf{v}_{\iota(p)}\|^2$ rapidly converges to a_p.
  - $\mathbf{v}_p^*$ no longer affects the learning dynamics! "Automatic deflation" process (Ge et al., 2021)

From gradient flow to online SGD

- Martingale decomposition in (Ben Arous et al., 2021), refined stochastic induction argument (Ren & Lee, 2024)
- "Unstable" discretization for only learning top P_* neurons

Takeaways

Future Directions

- Anisotropic input: $\mathbf{x} \sim \mathcal{N}(0, \mathbf{\Sigma})$, where covariance of $\mathbf{\Sigma}$ has power law decay.
  - Two parameter scaling law with source & capacity conditions?
- Optimal learning rate schedule to obtain anytime rate?
- Beyond additive model structure:
  - Are there more realistic natural language / algorithmic settings where we observe emergence and scaling laws?
  - Real "skills" are not orthogonal, as learning one may aid in learning others. How can we model this, and what scaling laws do we observe?
  - Scaling laws for compositional reasoning?

Thanks for listening!

Learning compositional functions with transformers from easy-to-hard data

Emergence and scaling laws for SGD learning of shallow neural networks