Part I: Emergence and scaling laws for SGD learning

Yunwei Ren*[1], Eshaan Nichani*[1], Denny Wu[23], Jason D. Lee[1]

[1]Princeton University [2]NYU [3]Flatiron Institute

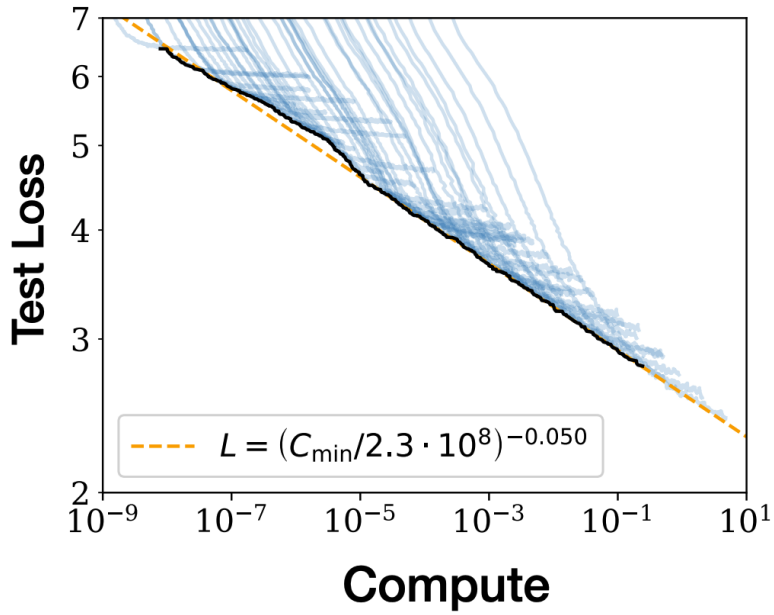
April 3, 2025

Neural Scaling Laws & Emergence

Neural Scaling Laws [Kaplan et al. 20] [Ho mann et al. 22].
Increasing compute and data leads to *power-law decay* in the loss.

Functional form: $L \;/\; D \quad + N \quad + C$.

D { number of data points.
N { number of parameters.

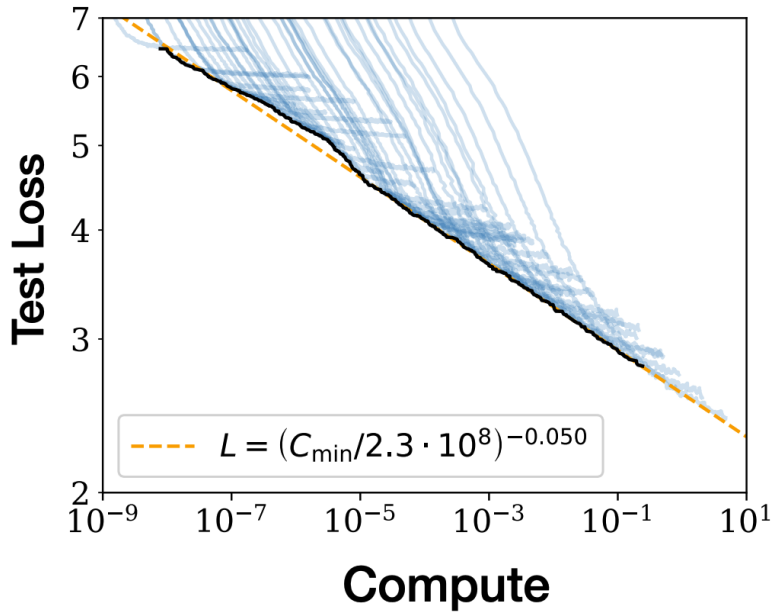


Neural Scaling Laws & Emergence

Neural Scaling Laws [Kaplan et al. 20] [Ho mann et al. 22].
Increasing compute and data leads to *power-law decay* in the loss.

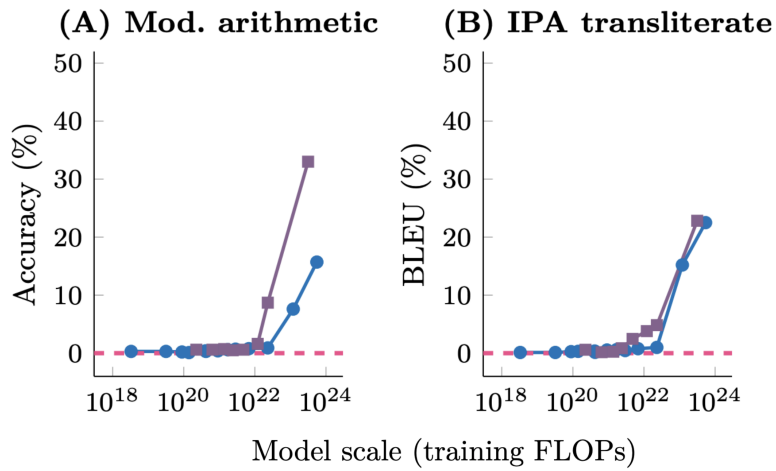
Functional form: $L \;/\; D \quad + N \quad + C$.

D { number of data points.
N { number of parameters.



Emergent Capabilities [Wei et al. 22] [Ganguli et al. 22].
Learning of individual tasks (skills) exhibits *sharp transition* with scale.

/ Unpredictable skill acquisition time.



Neural Scaling Laws & Emergence

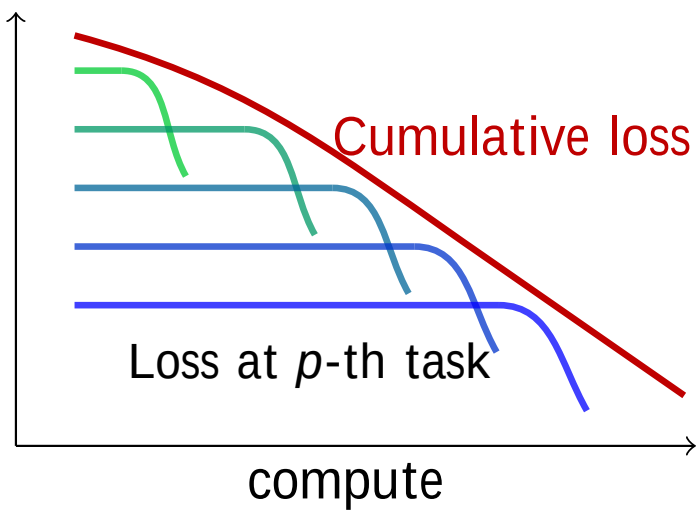
Question: How do we reconcile the *emergent behavior* in skill acquisition and the *smooth power-law decay* in the cumulative loss?

Neural Scaling Laws & Emergence

Question: How do we reconcile the *emergent behavior* in skill acquisition and the *smooth power-law decay* in the cumulative loss?

Hypothesis: Additive Model [Michaud et al. 24] [Nam et al. 24]

- Cumulative objective can be decomposed into a large number of distinct \skills", the learning of each exhibits *abrupt emergence*.
- Juxtaposition of numerous emergent learning curves at di erent timescales results in a predictable *power-law* rate in the loss.

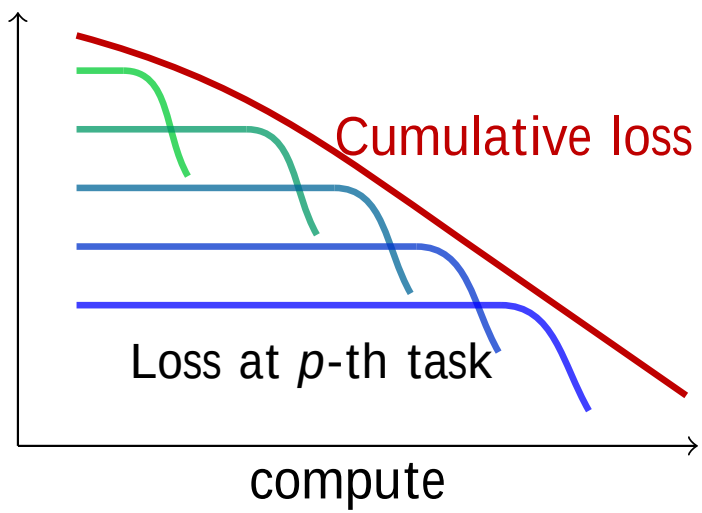


Neural Scaling Laws & Emergence

Question: How do we reconcile the *emergent behavior* in skill acquisition and the *smooth power-law decay* in the cumulative loss?

Hypothesis: Additive Model [Michaud et al. 24] [Nam et al. 24]

- Cumulative objective can be decomposed into a large number of distinct \skills", the learning of each exhibits *abrupt emergence*.
- Juxtaposition of numerous emergent learning curves at di erent timescales results in a predictable *power-law* rate in the loss.



This work: theoretical justification for the *additive model* hypothesis in SGD learning of shallow NN.

Emergence in Gradient-based Feature Learning

Gaussian single-index model: $f(x) = \quad (\langle x; \quad i \rangle; x \quad N(0; I_d)$.

- Requires learning the direction $\quad 2 \mathbb{R}^d$ and link function $\quad : \mathbb{R} \; ! \; \mathbb{R}$.

Learning algorithm should adapt to **low-dimensional structure**.

Emergence in Gradient-based Feature Learning

Gaussian single-index model: $f(x) = \quad (hx; \quad i);\ x \quad N(0; I_d)$.

- Requires learning the direction $\quad 2\ \mathbb{R}^d$ and link function $\quad : \mathbb{R} \ ! \ \mathbb{R}$.

Learning algorithm should adapt to **low-dimensional structure**.

Hermite expansion: $\quad (z) = \sum_{j=0}^{1} \quad _j \mathrm{He}_j(z), \quad _j = \mathbb{E}[\quad (z)\mathrm{He}_j(z)]$.

Information exponent [Ben Arous et al. 2021] [Dudeja & Hsu 2018]

The information exponent of $\quad$ is $k = \mathrm{IE}(\quad) = \min\{k \ 2\ \mathbb{N}_+ : \quad _k \ 6= 0\}$.

Intuition: amount of information in the gradient at *random initialization*.

Emergence in Gradient-based Feature Learning

Hermite expansion: $(z) = \sum_{j=0}^{1} \; _j \mathrm{He}_j(z), \quad _j = \mathrm{E}[\quad (z)\mathrm{He}_j(z)].$

Information exponent [Ben Arous et al. 2021] [Dudeja & Hsu 2018]

The information exponent of is $k = \mathrm{IE}(\quad) = \min\{k \; 2 \; \mathrm{N}_+ : \; _k \; 6= 0\}$.

$$\mathrm{E}[\, r \; _w L(f_{\mathrm{NN}})] \quad \mathrm{E}[\, r \; _w (f_{\mathrm{NN}}(x) f \; (x))]$$
$$= \quad \mathrm{E}[\; ^0(hx; \quad i) \; ^0(hx; w\, i)] + w \quad \mathrm{E}[:::] \quad \textit{Stein's lemma}$$

Emergence in Gradient-based Feature Learning

Hermite expansion: $(z) = \sum_{j=0}^{1} \; _j \mathrm{He}_j(z), \quad _j = \mathrm{E}[\quad (z)\mathrm{He}_j(z)].$

Information exponent [Ben Arous et al. 2021] [Dudeja & Hsu 2018]

The information exponent of is $k = \mathrm{IE}(\quad) = \min\{k \; 2 \; \mathrm{N}_+ : \; _k \not\approx 0\}$.

$$
\begin{aligned}
\mathrm{E}[\, r_{\,w} L(f_{\mathrm{NN}})] \quad & \mathrm{E}[\, r_{\,w} (f_{\mathrm{NN}}(x) f \; (x))] \\
= \quad & \mathrm{E}[\; ^{0}(hx; \quad i) \; ^{0}(hx; w\, i)] + w \quad \mathrm{E}[:::] \quad \textit{Stein's lemma} \\
= \quad & \sum_{j=0}^{\not\infty} (j+1)^2 \; _{j+1} \; _{j+1} \; \underbrace{hw; \; i^{j}}_{\{z\}} \quad +::: \quad \textit{Hermite expansion}
\end{aligned}
$$

$d^{\,j=2}$ at initialization

Emergence in Gradient-based Feature Learning

Hermite expansion: $(z) = \sum_{j=0}^{1} \; _j \mathrm{He}_j(z), \quad _j = \mathrm{E}[\quad (z)\mathrm{He}_j(z)].$

Information exponent [Ben Arous et al. 2021] [Dudeja & Hsu 2018]

The information exponent of is $k = \mathrm{IE}(\quad) = \min\{k \in \mathbb{N}_+ : \; _k \neq 0\}$.

$$\begin{aligned}
\mathrm{E}[\, r_w L(f_{\mathrm{NN}})] \quad & \mathrm{E}[\, r_w (f_{\mathrm{NN}}(x) f \; (x))] \\
= \quad & \mathrm{E}[\; ^0(\langle x; \; i\rangle) \; ^0(\langle x; w\, i\rangle)] + w \quad \mathrm{E}[:::] \quad \textit{Stein's lemma} \\
= \quad & \sum_{j=0}^{\infty} (j+1)^2 \; _{j+1} \; _{j+1} \; \langle w; \; i\rangle^j \quad +::: \quad \textit{Hermite expansion}
\end{aligned}$$

$d^{\,j=2}$ at initialization

Theorem ([Ben Arous et al. 21], [Bietti et al. 22], [Damian et al. 23]...)

A two-layer neural network can learn single-index f with information exponent k using $n \; ' \; T \; \& \; d^{\Theta(k)}$ samples and SGD steps.

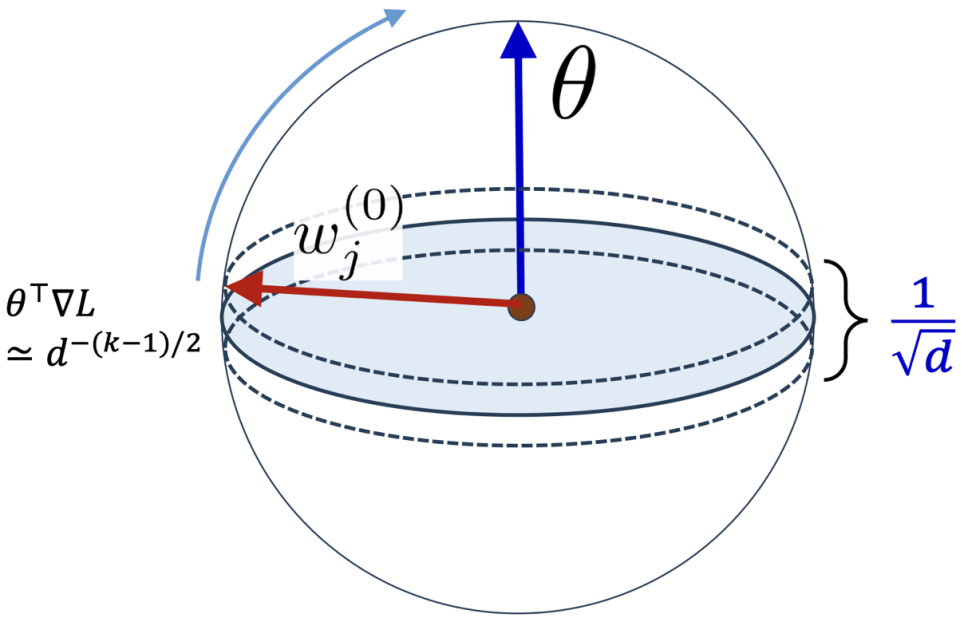
Emergence in Gradient-based Feature Learning

Hermite expansion: $(z) = \sum_{j=0}^{1} \; _j \mathrm{He}_j(z), \quad _j = \mathrm{E}[\quad (z)\mathrm{He}_j(z)].$

Information exponent [Ben Arous et al. 2021] [Dudeja & Hsu 2018]

The information exponent of is $k = \mathrm{IE}(\quad) = \min\{k \in \mathbb{N}_+ : \; _k \neq 0\}$.

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



Emergent learning curve

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

Learning Extensive-width Neural Network

Target Function: Width-P two-layer neural network

$$f(x) = \sum_{m=1}^{P} a_p \quad (\langle x; \ _p i \rangle); \qquad \sum_{p}^{P} a_p^2 = 1; \ f \ _p g_{p=1}^{P} \text{ orthonormal:}$$

Motivation: learning sum of (orthogonal) single-index tasks.

with IE $k > 2$) *emergent* learning curve for each direction.

Varying second-layer $f a_p g_{p=1}^{P}$) separation in length of search phase.

Learning Extensive-width Neural Network

Target Function: Width-P two-layer neural network

$$f(x) = \sum_{m=1}^{P} a_p \quad (hx; \ _p i); \qquad \sum_{p}^{P} a_p^2 = 1; \ f \ _p g_{p=1}^{P} \text{ orthonormal:}$$

- **Extensive width.** $P \quad d$ for 2 $[0; c]; c > 0$.
  Large number of tasks) in nite-dimensional e ective dynamics.
- **Large condition number.** $a_{\max}=a_{\min} = \quad (\text{Poly}(P))$.
  Covers **power-law decay** in second-layer $a_p \quad p \quad ; \quad 2$ $[0; 1)$.
- **Single-phase SGD learning** with MSE loss.
  Avoids layer-wise training which alters scaling law.

Learning Extensive-width Neural Network

Target Function: Width-P two-layer neural network

$$f(x) = \sum_{m=1}^{P} a_p \quad (\langle x; \ _p i); \qquad \sum_{p}^{P} a_p^2 = 1; \ f \ _p g_{p=1}^{P} \text{ orthonormal:}$$

- **Extensive width.** $P \quad d$ for 2 $[0; c]; c > 0$.
  Large number of tasks) in nite-dimensional *e ective dynamics*.
- **Large condition number.** $a_{\max}/a_{\min} = \quad (\mathrm{Poly}(P))$.
  Covers **power-law decay** in second-layer $a_p \quad p \quad ; \quad 2$ $[0; 1)$.
- **Single-phase SGD learning** with MSE loss.
  Avoids layer-wise training which alters scaling law.

This work: optimization and sample complexity for SGD training: sharp emergence time for each $_p$, smooth scaling law for MSE.

Main Theorem: Complexity of SGD Learning

- **Student model:** *2-homogeneous* two-layer NN (matching)

$$f(x) = \sum_{i=1}^{m} kw_i k_2^2 \quad (hx; w_i = kw_i ki):$$

Main Theorem: Complexity of SGD Learning

- **Student model:** *2-homogeneous* two-layer NN (matching)

$$f(x) = \sum_{i=1}^{m} \|w_i\|_2^2 \quad (\langle x; w_i \rangle = \|w_i\| i):$$

- **Online SGD training:** at time t, compute MSE gradient update

$$w_i(t+1) = w_i(t) \quad r_{w_i}(f(x_t) \quad f(x_t))^2; \quad x_t \quad N(0; I_d):$$

Main Theorem: Complexity of SGD Learning

- **Student model:** *2-homogeneous* two-layer NN (matching)

$$f(x) = \sum_{i=1}^{m} \|w_i\|_2^2 \quad (\langle x; w_i/\|w_i\|\rangle):$$

- **Online SGD training:** at time t, compute MSE gradient update

$$w_i(t+1) = w_i(t) \quad r_{w_i}(f(x_t) \quad f(x_t))^2; \quad x_t \quad N(0; I_d):$$

Theorem (Complexity of SGD learning)

Assume $P \quad d^{0:1}$, $p < P$. *If we train a student NN with* $m = \tilde{}(p)$ *neurons using online SGD with* $\frac{a_{\bar{p}}}{d^{k=2}\mathrm{poly}(P)}$, *then w.h.p.,*

- *If* $p < p$, *alignment with* $_p$ *emerges at* $T_p \quad a_p{}^{1} \quad {}^{1}d^{k=2\ 1}$.

Main Theorem: Complexity of SGD Learning

- **Student model:** *2-homogeneous* two-layer NN (matching)

$$f(x) = \sum_{i=1}^{m} \|w_i\|_2^2 \quad (\langle x; w_i/\|w_i\| \rangle):$$

- **Online SGD training:** at time t, compute MSE gradient update

$$w_i(t+1) = w_i(t) \quad r_{w_i}(f(x_t) \quad f(x_t))^2; \quad x_t \quad N(0; I_d):$$

Theorem (Complexity of SGD learning)

Assume $P \quad d^{0:1}$, $p < P$. *If we train a student NN with* $m = \tilde{}(p)$ *neurons using online SGD with* $\frac{a_{\bar{p}}}{d^{k=2}\mathrm{poly}(P)}$, *then w.h.p.,*

- *If* $p < p$, *alignment with* $_p$ *emerges at* $T_p \quad a_p{}^1 \quad {}^1 d^{k=2} \ {}^1$.
- *All directions up to* p *are learned at* $n \quad T \quad a_{\bar{p}}{}^2 d^k \ {}^1\mathrm{poly}(P)$.

Main Theorem: Complexity of SGD Learning

Theorem (Complexity of SGD learning)

Assume $P \quad d^{0:1}$, $p < P$. *If we train a student NN with* $m = \tilde{}(p)$ *neurons using online SGD with* $\frac{a_{\bar{p}}}{d^{k=2}\mathrm{poly}(P)}$, *then w.h.p.,*

- p *If* $p < p$, *alignment with* $_p$ *emerges at* $T_p \quad a_p{}^1 \quad {}^1 d^{k=2\ 1}$.
- p *All directions up to* p *are learned at* $n \quad T \quad a_{\bar{p}}{}^2 d^k \ {}^1\mathrm{poly}(P)$.

Corollary: ignoring logarithmic factors, achieving small population loss (i.e., learning *all* tasks) requires $m \quad P$ neurons and $n \quad T \quad a_{\min}^2 d^k \ {}^1\mathrm{poly}(P)$.

Main Theorem: Complexity of SGD Learning

Theorem (Complexity of SGD learning)

Assume $P \quad d^{0:1}$, $p < P$. *If we train a student NN with* $m = \tilde{}(p)$ *neurons using online SGD with* $\frac{a_{\bar{p}}}{d^{k=2}\mathrm{poly}(P)}$, *then w.h.p.,*

- p *If* $p < p$, *alignment with* $_p$ *emerges at* $T_p \quad a_p{}^{1} \quad {}^{1}d^{k=2\ 1}$.
- p *All directions up to* p *are learned at* $n \quad T \quad a_{\bar{p}}{}^{2}d^{k\ 1}\mathrm{poly}(P)$.

Corollary: ignoring logarithmic factors, achieving small population loss (i.e., learning *all* tasks) requires $m \quad P$ neurons and $n \quad T \quad a_{\min}^{2}d^{k\ 1}\mathrm{poly}(P)$.

Prior works required sample size or compute *exponentially* large in the condition number: $n; P \quad \exp \frac{a_{\max}}{a_{\min}}$ [Li et al. 22] [Oko et al. 24].

Our learning procedure does not involve reinitialization [Ge et al. 21] or Stiefel constraint [Ben Arous et al. 24].

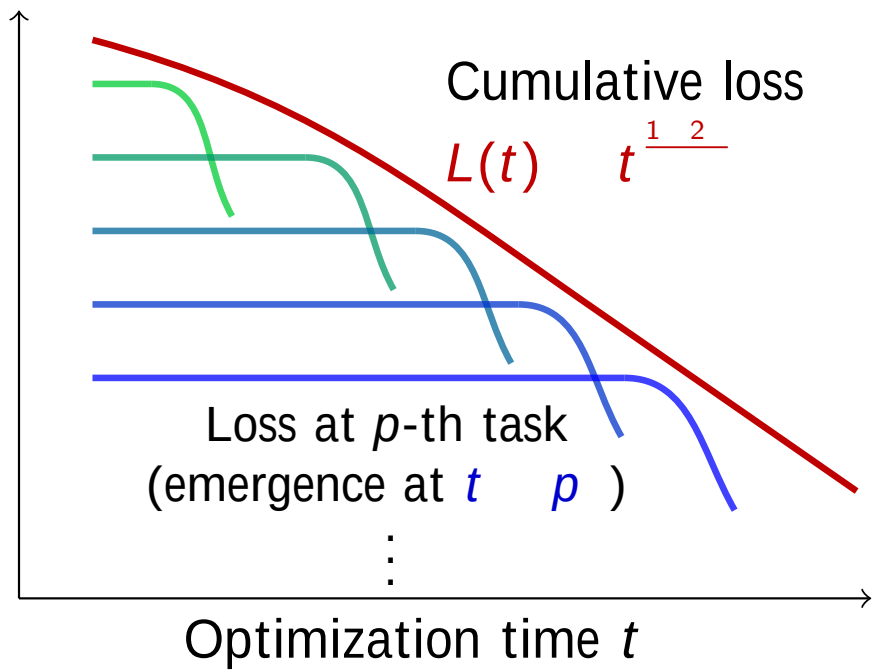
Neural Scaling Laws for Feature Learning

Corollary (Emergence & Scaling Laws)

Assume $a_p \quad p \quad$ *for* $\quad > 1/2$ *and* *fixed learning rate* $\quad$, *then*

1. **Emergence:** *the p-th task is learned at time* $t \quad d^{k/2 \quad 1} p$.
2. **Scaling law:** *population MSE exhibits power-law decay*

$$L(t) \quad m^{1 \quad 2} \_ \quad t d^{1 \quad k/2} \quad .$$



Neural Scaling Laws for Feature Learning

Corollary (Emergence & Scaling Laws)

Assume $a_p \quad p \quad$ *for* $\quad > 1=2$ *and* $\quad$ *xed learning rate* $\quad$, *then*

1. **Emergence:** *the p-th task is learned at time* $\quad t \quad d^{k=2 \quad 1}p \quad .$
2. **Scaling law:** *population MSE exhibits power-law decay*

$$L(t) \quad m^{1 \quad 2} \_ \quad td^{1 \quad k=2}{}^{\frac{1 \quad 2}{}}.$$

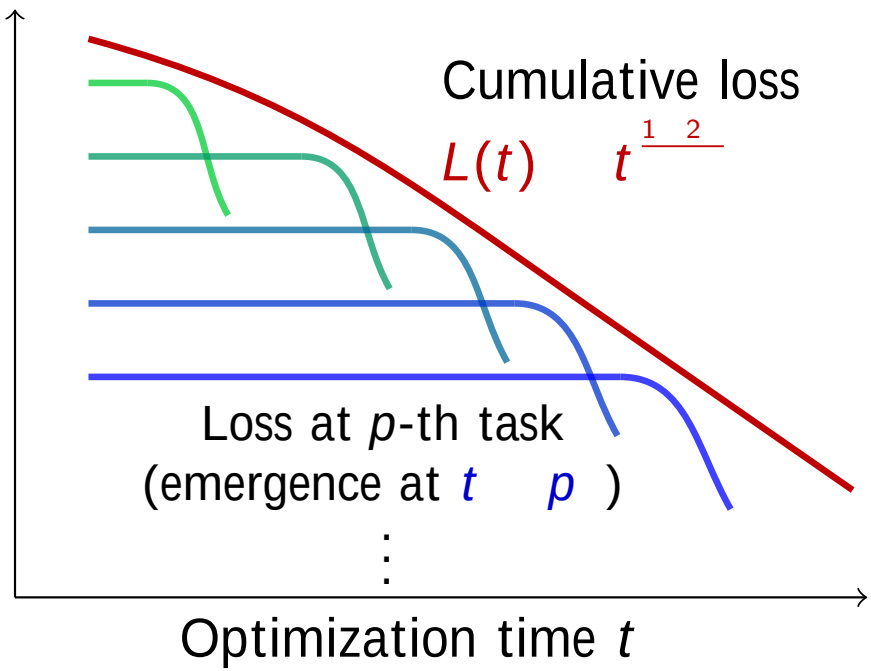
Intuition: Assume *decoupled* learning of di erent teacher components,

Direction $\quad_p$ learned at $T_p \;/\; p \quad {}^1 d^{k=2 \quad 1}$.

$$L(t) \quad \sum_{p=1}^{P} a_p^2 \mathbb{I}\{t < T_p\} \;)$$

$$L(T_p) \quad \int_p^{\infty} s^{-2}\, ds \quad p^{1 \quad 2} .$$

$m_B < P$ student neurons reaches $\sum_{s>m} a_s^2 \quad m^{1 \quad 2}$ error.



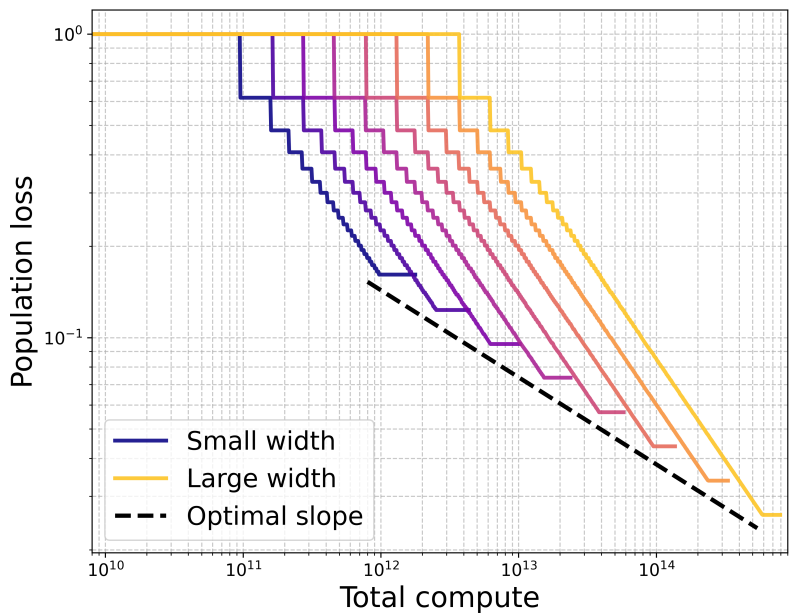
Neural Scaling Laws for Feature Learning

p $m^{1\ 2}$ { approximation barrier.

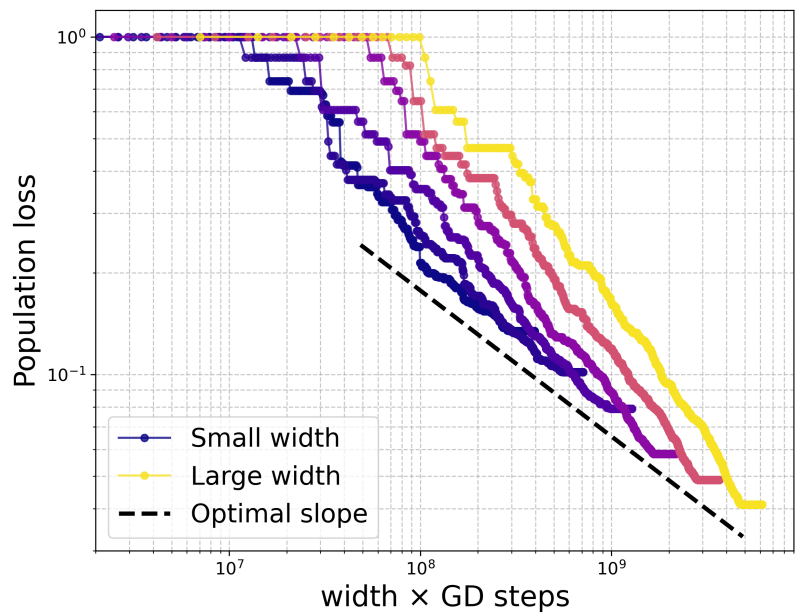
Determined by the student network width.

p $td^{1\ k=2}$ $\frac{1\ 2}{}$ { optimization error.

Determined by the number of online SGD steps.



(a) Theoretical scaling law.



(b) Empirical scaling law.

Remark on Discretization

Note: the continuous-time (or constant-) rate can be misleading!

/ Exponent does not match known rates for weak ℓ_p ball [Johnstone 17].

Adaptive learning rate for p-th task. Consider only learning the top-p neurons.

\Optimal" step size for the p-th task: $\frac{a_p}{d^{k=2}\mathrm{poly}(P)}$, a_p p .

Direction p now learned at
$n = T_p$ p $^1 d^{k=2}$ $^1 = p^2$ d^k $^1\mathrm{poly}(P)$.

Sample complexity: under this learning rate for the rst p neurons,

$$L(n;m) \qquad \frac{n}{d^k \ ^1\mathrm{poly}(P)} \quad \frac{1 \quad 2}{2} \quad + m^{1 \quad 2}$$

, This now matches the known rates for weak ℓ_p ball.

Proof Sketch

Decoupled gradient dynamics.

Let $w_{(p)}$ denote the neuron that eventually converged to direction $_p$, and $m_{p;q}(t) = \hbar w_p(t)=kw_p(t)k;\ _q i$ measures the overlap at time t.

Claim 1 - decoupling. When all $m^2_{(p);q}$ ($q \neq p$) are small, the learning of di erent directions can be approximately decoupled.

Claim 2 - sharp transitions. Since the norm of w_p grows rapidly after alignment is achieved, all $m^2_{(p);q}$ ($q \neq p$) remain small when the q-th direction is recovered by $w_{(q)}$ { after which $_q$ no longer a ects the learning dynamics (\automatic de ation").

Proof Sketch

Decoupled gradient dynamics.

Let $w_{(p)}$ denote the neuron that eventually converged to direction $_p$, and $m_{p;q}(t) = \hbar w_p(t)/kw_p(t)k;\ _q i$ measures the overlap at time t.

Claim 1 - decoupling. When all $m^2_{(p);q}$ $(q \neq p)$ are small, the learning of di erent directions can be approximately decoupled.

Claim 2 - sharp transitions. Since the norm of w_p grows rapidly after alignment is achieved, all $m^2_{(p);q}$ $(q \neq p)$ remain small when the q-th direction is recovered by $w_{(q)}$ { after which $_q$ no longer a ects the learning dynamics (\automatic de ation").

From gradient flow to online SGD.

Martingale decomposition in [Ben Arous et al. 21] + a re ned stochastic induction argument from [Ren & Lee 24].

\Unstable" discretization that couples the online SGD dynamics for learning the top-p teacher neurons.

Remarks and Future Directions

- **Edge Case:** information exponent $k = 2$ (e.g., quadratic).
  - / Dynamics of di erent directions *cannot be decoupled*.
- *Companion work:* introduce algorithmic modi cation (Stiefel SGD) and handle the special case of quadratic activation.

  Gerard Ben Arous, Murat A. Erdogdu, N. Mert Vural, Denny Wu. *\Scaling laws for learning quadratic two-layer neural networks with SGD in high dimensions".*

Remarks and Future Directions

- **Edge Case:** information exponent $k = 2$ (e.g., quadratic).
  - / Dynamics of di erent directions *cannot be decoupled*.
- *Companion work:* introduce algorithmic modi cation (Stiefel SGD) and handle the special case of quadratic activation.

  Gerard Ben Arous, Murat A. Erdogdu, N. Mert Vural, Denny Wu. *\Scaling laws for learning quadratic two-layer neural networks with SGD in high dimensions".*

Future Directions

- Anisotropic input: x $N(0; \mathbf{\Sigma})$, where $\mathbf{\Sigma} = \sum_{i=1}^{d}$ $_i v_i v_i^{>}$; $_i$ i .

  Two-parameter scaling law? (source & capacity conditions)
- Beyond additive structure) scaling laws for compositional generalization? (More on this in Part II)

Part II: Learning Compositional Functions with Transformers

Zixuan Wang*[1], Eshaan Nichani*[1], Alberto Bietti[3], Alex Damian[1], Daniel Hsu[4], Jason D. Lee[1], Denny Wu[23]

[1]Princeton University [2]NYU [3]Flatiron Institute [4]Columbia University

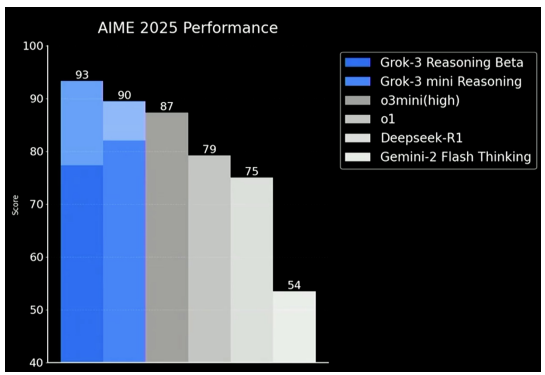
April 3, 2025

Motivation: multi-step reasoning tasks

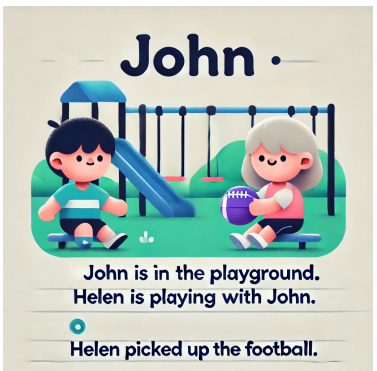
Large language models exhibits remarkable reasoning capabilities.

They can solve complex tasks that require combining multiple reasoning steps. [Wei et al., 2022]

Mathematics



Multi-hop QA



Coding

Examples (Multi-hop QA)

Prompt: **John** is in the playground. [...] Helen is playing with **John**. [...] Helen picked up the football. [...] Where is the football?"
Answer: \In the playground."

Function composition

Multi-step reasoning tasks can be viewed as **function composition**.

Prompt: **John** is in the playground. [...] Helen is playing with **John**. [...] Helen picked up the football. [...] Where is the football?" **Answer:** \In the playground."

f_{location}(**John**) = playground, f_{with}(Helen) = **John**,
f_{hold}(football) = Helen

The model needs to compute f_{location} f_{with} f_{hold}(football) and get playground.

Note: (location, with, hold) are also function inputs.

Function composition

Multi-step reasoning tasks can be viewed as **function composition**.

Prompt: **John** is in the playground. [...] Helen is playing with **John**. [...] Helen picked up the football. [...] Where is the football?" **Answer:** \In the playground."

$f_{\text{location}}(\textbf{John})$ = playground, $f_{\text{with}}(\text{Helen}) = \textbf{John}$,
$f_{\text{hold}}(\text{football}) = \text{Helen}$

The model needs to compute $f_{\text{location}} \quad f_{\text{with}} \quad f_{\text{hold}}(\text{football})$ and get playground.

Note: (location, with, hold) are also function inputs.

Abstractly, the model can do the following function composition:

Inputs: $(\ _1; \ _2; \ldots; \ _k; x)$

Outputs: $f^{(k)}_{k} \quad f^{(k \ 1)}_{k \ 1} \quad f^{(1)}_{1}(x)$ where the hidden function is position dependent.

How does transformer compose functions?

With Chain-of-Thought (CoT):

- Expressivity [Li et al., 2024]
- Require explicit k inference steps to compose k functions.
- Ine cient for simple tasks?

How does transformer compose functions?

With Chain-of-Thought (CoT):

- Expressivity [Li et al., 2024]
- Require explicit k inference steps to compose k functions.
- Ine cient for simple tasks?

Without Chain-of-Thought:

- Limited expressivity [Merrill and Sabharwal, 2023]
- Only a single inference step.
- Potentially faster?

How does transformer compose functions?

With Chain-of-Thought (CoT):

- Expressivity [Li et al., 2024]
- Require explicit k inference steps to compose k functions.
- Ine cient for simple tasks?

Without Chain-of-Thought:

- Limited expressivity [Merrill and Sabharwal, 2023]
- Only a single inference step.
- Potentially faster?

Focus of this talk:

- Can transformers compose (some) functions internally?
- Can they learn composition e ciently via gradient-based training?

Transformers and self-attention

Definition (Self-attention)

The self-attention head $\mathrm{attn}(\,;W_{KQ};W_{OV}) : \mathrm{R}^{d\ \ T} \ ! \ \mathrm{R}^{d\ \ T}$ is, with $W_{KQ};W_{OV}\ 2\ \mathrm{R}^{d\ \ d}$

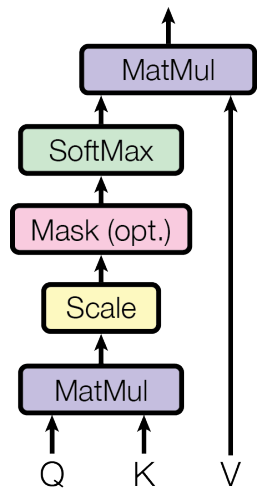
$$\mathrm{attn}(X;W_{KQ};W_{OV}) = X + W_{OV}XS(X^{>}W_{KQ}X);$$

where the softmax function S is applied column-wise.

Convex combination of all values $W_{OV}X$ with coe cient $S(X^{>}W_{KQ}X)_{(i;j)}$, i.e. attention score.

A multi-layer transformer composes multiple self-attention layers.

We focus on attention-only transformers now.



Transformers parallelize composition

Can transformers compose (some) functions internally?

Yes! Enabled by parallelism.

Composing k functions only requires $O(\log k)$ layers.

Transformers parallelize composition

Can transformers compose (some) functions internally?

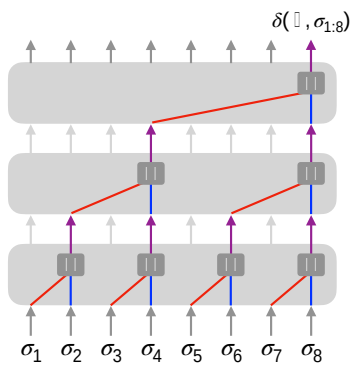
Yes! Enabled by parallelism.

Composing k functions only requires $O(\log k)$ layers.

Examples (Efficiently parallelizable tasks)

k-hop induction head [Sanford et al., 2024]; Finite state automata [Liu et al., 2023].

(2-hop Induction head)



(Finite state automata)

Our task: k-fold composition

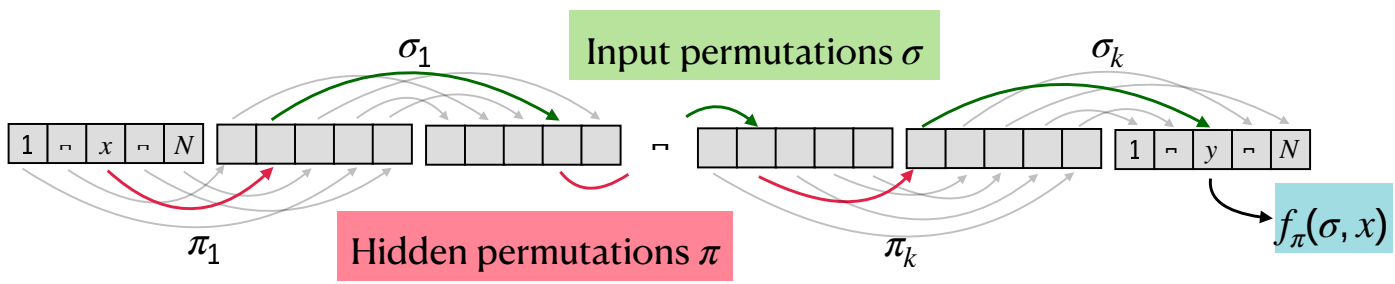
Definition. (k-fold composition)

$:= (\ _1; : : : ;\ _k)\ 2\ (S_N)^k$ is a tuple of k hidden permutations.

Input: $(\ ; x)\ 2\ X := (S_N)^k\quad [N]$, where $\quad = (\ _1; : : : ;\ _k)$, x is an index in $[N]$.

Output $f\ : X\ !\ [N]$ is de ned as

$$f\ (\ ; x) := (\ _k\quad _k\quad _k\ 1\quad _k\ 1\quad\quad\quad _1\quad _1)(x):$$



Remark: $:= (\ _1; : : : ;\ _k)\ 2\ (S_N)^k$ induces a function class F.

Construction for k-fold composition

Theorem (Expressivity of Transformers)

Assume k is a power of 2. For any $2 (S_N)^k$, there exists an $L = \log_2 k + 1$ layer transformer with embedding dimension $\mathcal{O}(kN)$ that expresses k-fold composition.

Construction for k-fold composition

Theorem (Expressivity of Transformers)

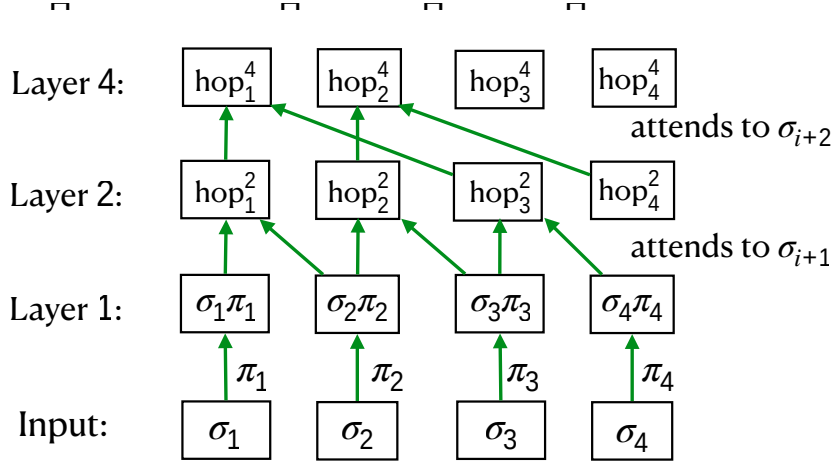
Assume k is a power of 2. For any $\quad 2\ (S_N)^k$, there exists an $L = \log_2 k + 1$ layer transformer with embedding dimension $\mathcal{O}(kN)$ that expresses k-fold composition.

Layer 1 encodes hidden $_i$'s.

Layer 2 composes $_{i+1}\quad _{i+1}$ and $_i\quad _i$ to get

$$\text{hop}_i^2 := \quad _{i+1}\quad _{i+1}\quad _i\quad _i$$

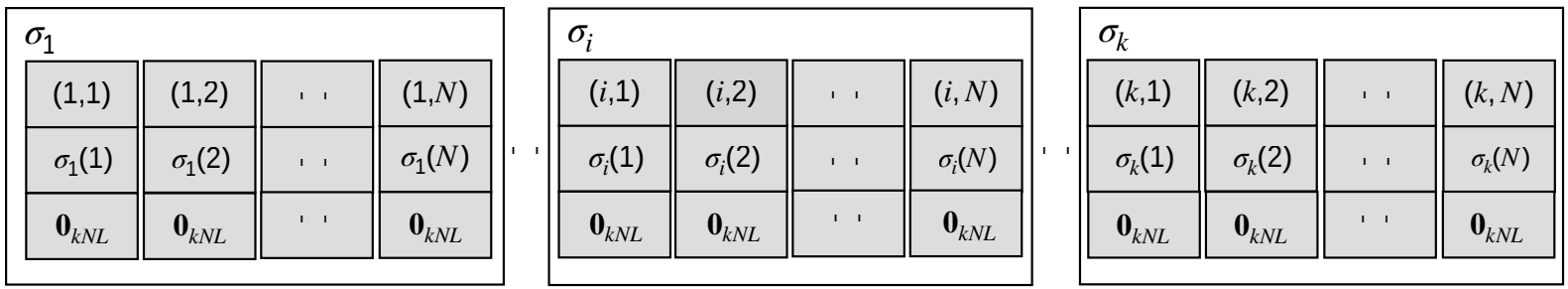
Layer $\grave{}$ composes $\text{hop}_i^{2^l\ 2}$ and $\text{hop}_{i+2^l\ 2}^{2^l\ 2}$ and get $\text{hop}_i^{2^l\ 1}$.



The parallel solution (Layer 1)

Embedding for each $(i;j)$ 2 $[N]$ $[k]$ stores one-hot embedding of $(i;j)$ and $(i;\ _i(j))$

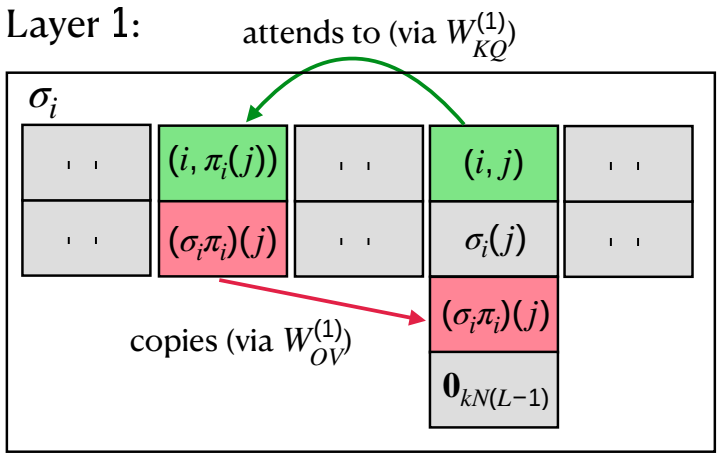
Input $X^{(0)}$:



The key-query matrix makes position $(i;j)$ attend to position $(i;\ (j))$.

The value matrix copies $\text{hop}_i^1(j) := (\ _i\ _i)(j)$ into the residual stream of $(i;j)$

The rst layer computes 1-hop!



The parallel solution (Layer $\ell + 1$)

Assume that the previous layer ℓ computes $\text{hop}_i^{2^{\ell-1}}$ (i.e residual stream of $(i;j)$ contains $\text{hop}_i^{2^{\ell-1}}(j)$).

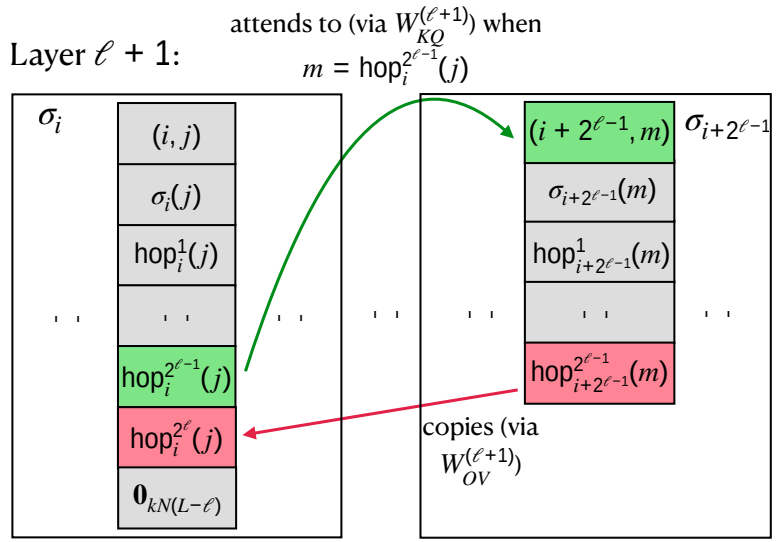
The key-query matrix makes position $(i;j)$ attend to position $(i + 2^{\ell-1}; \text{hop}_{i^{2^{\ell-1}}}(j))$.

Value matrix copies $\text{hop}_{i+2^{\ell-1}}^{2^{\ell-1}}(\text{hop}_i^{2^{\ell-1}}(j))$ into residual stream of $(i;j)$.

Thus the $(\ell + 1)$th layer computes

$$\text{hop}_i^{2^\ell} = \text{hop}_{i+2^{\ell-1}}^{2^{\ell-1}} \circ \text{hop}_i^{2^{\ell-1}};$$

Finally, the L-th layer outputs k-hop.



Computational hardness of k-fold composition

Is this parallel construction for the k-fold composition e ciently learnable by GD?

Computational hardness of k-fold composition

Is this parallel construction for the k-fold composition e ciently learnable by GD?

Negative result: Statistical Query lower bound.

SQ framework: Learner speci es query $q : X \quad [N] \; ! \; \mathrm{R}$ and tolerance . SQ oracle responds with $\hat{q}$, a noisy estimate of q

$$j\hat{q} \quad \mathrm{E}_{;x}[q(\; ; x; f \; (\; ; x))]j$$

Computational hardness of k-fold composition

Is this parallel construction for the k-fold composition e ciently learnable by GD?

Negative result: Statistical Query lower bound.

SQ framework: Learner speci es query $q : X \quad [N] \; ! \; \mathrm{R}$ and tolerance . SQ oracle responds with $\hat{q}$, a noisy estimate of q

$$j\hat{q} \quad \mathrm{E}_{;x}[q(\; ; x; f \; (\; ; x))]j$$

Theorem (SQ lower bound for k-fold composition)

Any SQ learner for the k-fold composition function class F must either make $q \quad N^{\Omega(k)}$ queries or use a tolerance $\quad N^{\Omega(k)}$ to output a predictor $\hat{f}$ with loss $L(\hat{f}) \; . \; 1$.

$n^{1=2}$ by i.i.d. concentration heuristic, where n is the sample size.

) Either runtime (number of queries) or sample size must be $N^{\Omega(k)}$.

Easy-to-hard: efficiency of curriculum learning

However, a curriculum can guide the transformer to learn the solution e ciently!

Stage 1: GD on the data with label $\text{hop}_i^1 = {}_i\ {}_i$ for $i\ 2\ [k]$.

Stage `: GD on the data with label $\text{hop}_i^{2^{`\ 1}}$.

We can learn the k-fold function with poly$(N; k)$ samples!

Easy-to-hard: efficiency of curriculum learning

However, a curriculum can guide the transformer to learn the solution e ciently!

Stage 1: GD on the data with label $\mathrm{hop}_i^1 =$ $_i$ $_i$ for i 2 $[k]$.

Stage `: GD on the data with label $\mathrm{hop}_i^{2^{`}}$ ^{1}.

We can learn the k-fold function with poly$(N; k)$ samples!

Theorem (GD upper bound with curriculum)

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

Easy-to-hard: efficiency of curriculum learning

However, a curriculum can guide the transformer to learn the solution e ciently!

Stage 1: GD on the data with label $\text{hop}_i^1 = {}_i \ {}_i$ for $i \ 2 \ [k]$.

Stage `: GD on the data with label $\text{hop}_i^{2^{`-1}}$.

We can learn the k-fold function with poly$(N; k)$ samples!

Theorem (GD upper bound with curriculum)

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

Takeaway: $n \ N^{\Omega(k)}$ (**Exponential**)) n poly$(N; k)$ (**Polynomial**).

, Easy examples/intermediate supervision is essential for complex tasks!

Learning with curriculum

Stage 1:

In the construction, the rst layer makes the $(i; j)$ position attend to $(i; \ _i(j))$ position.

With 1-hop data, learning this rst layer is easy!

Gradient descent can correctly learn the constructed attention pattern, i.e. recover the hidden permutations $\ _i$'s.

Learning with curriculum

Stage 1:

In the construction, the rst layer makes the $(i; j)$ position attend to $(i; \ _i(j))$ position.

With 1-hop data, learning this rst layer is easy!

Gradient descent can correctly learn the constructed attention pattern, i.e. recover the hidden permutations $\ _i$'s.

Stage ` (` 2):

`th layer learns to compose $\text{hop}_i^{2^{`-1}}$ and $\text{hop}_{i+2^{`-1}}^{2^{`-1}}$ computed in the previous layer.

With $2^{`-1}$-hop data, GD correctly makes $(i; j)$ position attend to $(i + 2^{`-1}; \text{hop}_i^{2^{`-1}}(j))$ position, thus matching the construction.

Learning with data mixture

In practice, designing explicit curriculum is challenging:

- Hard to ensure increasing difficulty; facing catastrophic forgetting
- Instead, practitioners use **data mixture** with different difficulties

Learning with data mixture

In practice, designing explicit curriculum is challenging:

- Hard to ensure increasing di culty; facing catastrophic forgetting
- Instead, practitioners use **data mixture** with di erent di culties

Training on data mixture also works!

Theorem (Upper bound on mixed data, informal)

Assume $k = 2^{L-1}$, if the sample size $n \geq \tilde{}(k^4 N^6)$, the output of gradient descent training on the mixture of f1,2,4,...,kg-fold data learns the k-fold composition function.

Learning with data mixture

In practice, designing explicit curriculum is challenging:

- Hard to ensure increasing difficulty; facing catastrophic forgetting
- Instead, practitioners use **data mixture** with different difficulties

Training on data mixture also works!

Theorem (Upper bound on mixed data, informal)

Assume $k = 2^{L-1}$, if the sample size $n \gtrsim \tilde{}(k^4 N^6)$, the output of gradient descent training on the mixture of f1,2,4,...,kg-fold data learns the k-fold composition function.

Intuition: if layer ℓ (the $2^{\ell-1}$-fold function) is not learned, GD cannot learn later layers with $\ell' > \ell$ | It is essential to learn from easy to hard!

Data mixture **implicitly induces curriculum learning**.

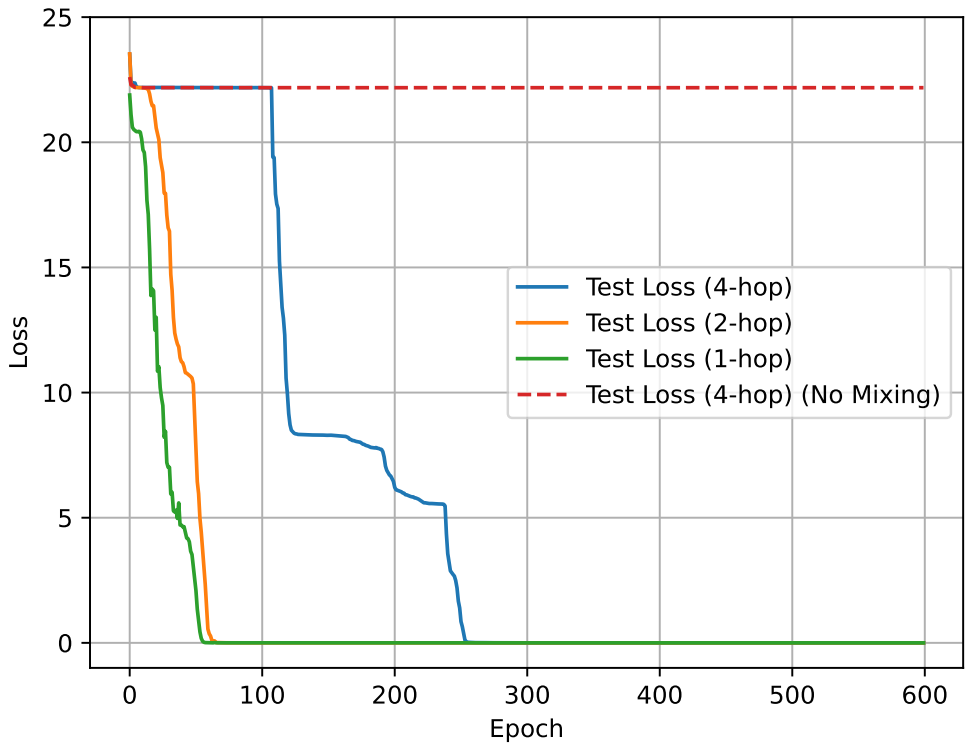
Experiments

Simulation on normal transformers with MLP:

Without data mixture: Transformer cannot learn the 4-hop task.

With mixture: Transformer learns **from easy to hard**!

Learning sequentially from easy (1-hop) to hard tasks (4-hop).



Remarks

Connection to k-sparse parity

k-fold composition exhibits a *statistical-computational gap*.
Information theoretic sample complexity is $n \quad kN \log N$, while a computationally e cient SQ learner requires $n \quad N^{\Omega(k)}$.

Mirrors k-parity in d dimensions (requires d^k samples/compute).

Remarks

Connection to k-sparse parity

k-fold composition exhibits a *statistical-computational gap*. Information theoretic sample complexity is $n \quad kN \log N$, while a computationally e cient SQ learner requires $n \quad N^{\Omega(k)}$.

Mirrors k-parity in d dimensions (requires d^k samples/compute).

In both settings, e cient gradient-based learning requires **curriculum/intermediate supervision**
[Abbe et al., 2023, Wen et al., 2024, Kim and Suzuki, 2024].

Unlike k-parity, k-fold composition *cannot be expressed by a shallow neural network*.

Remarks

Connection to k-sparse parity

k-fold composition exhibits a *statistical-computational gap*. Information theoretic sample complexity is $n \quad kN \log N$, while a computationally e cient SQ learner requires $n \quad N^{\Omega(k)}$.

Mirrors k-parity in d dimensions (requires d^k samples/compute).

In both settings, e cient gradient-based learning requires **curriculum/intermediate supervision** [Abbe et al., 2023, Wen et al., 2024, Kim and Suzuki, 2024].

Unlike k-parity, k-fold composition *cannot be expressed by a shallow neural network*.

Future directions

How to learn general compositions?

Learning a more e cient embedding.

References


Abbe, E., Cornacchia, E., and Lot , A. (2023).
Provable advantage of curriculum learning on parity targets with mixed inputs.
Advances in Neural Information Processing Systems, 36:24291{24321.


Kim, J. and Suzuki, T. (2024).
Transformers provably solve parity e ciently with chain of thought.
arXiv preprint arXiv:2410.08633.


Li, Z., Liu, H., Zhou, D., and Ma, T. (2024).
Chain of thought empowers transformers to solve inherently serial problems.
arXiv preprint arXiv:2402.12875.


Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. (2023).
Transformers learn shortcuts to automata.
In *The Eleventh International Conference on Learning Representations*.


Merrill, W. and Sabharwal, A. (2023).
The parallelism tradeo : Limitations of log-precision transformers.
Transactions of the Association for Computational Linguistics, 11:531{545.


Sanford, C., Hsu, D., and Telgarsky, M. (2024).