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

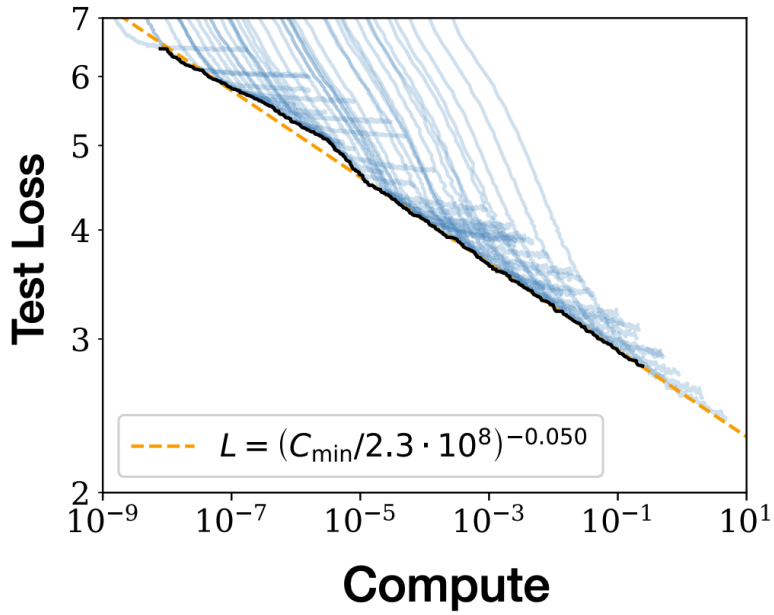
April 3, 2025

Neural Scaling Laws & Emergence

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

Functional form: $\mathcal{L} \propto D^{-\alpha} + N^{-\beta} + C$.

- D – number of data points.
- N – number of parameters.

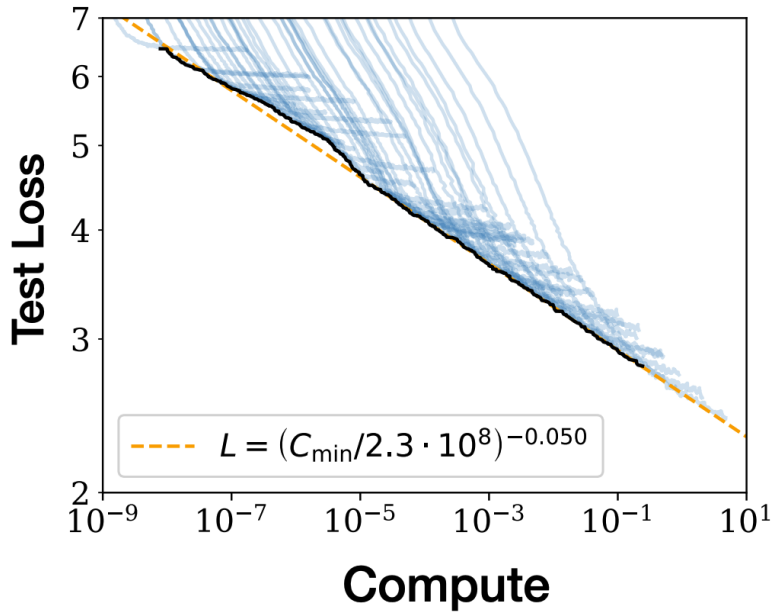


Neural Scaling Laws & Emergence

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

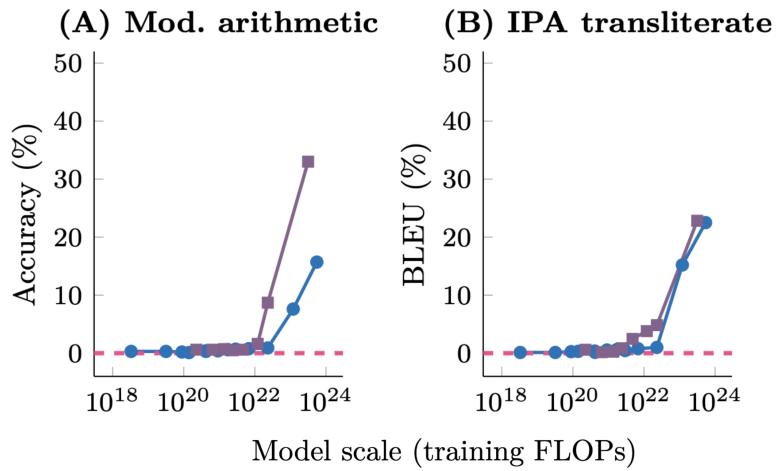
Functional form: $\mathcal{L} \propto D^{-\alpha} + N^{-\beta} + C$.

- 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

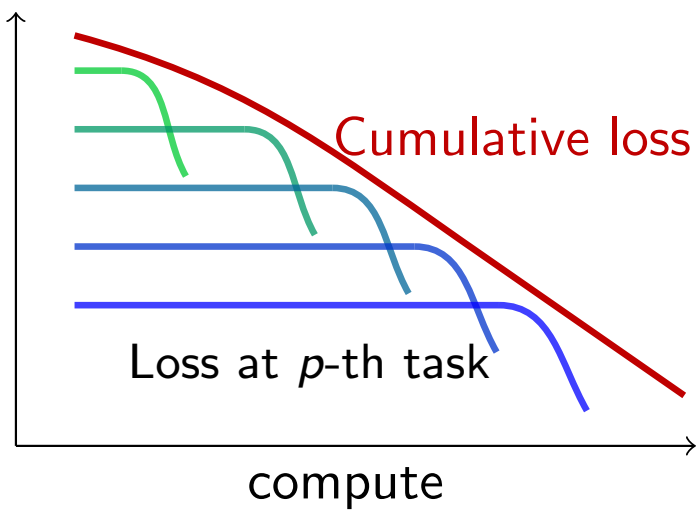
Question: How do we reconcile the *emergent behavior* in skill acquisition and the *smooth power-law decay* in the cumulative loss?

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 different timescales results in a predictable *power-law* rate in the loss.

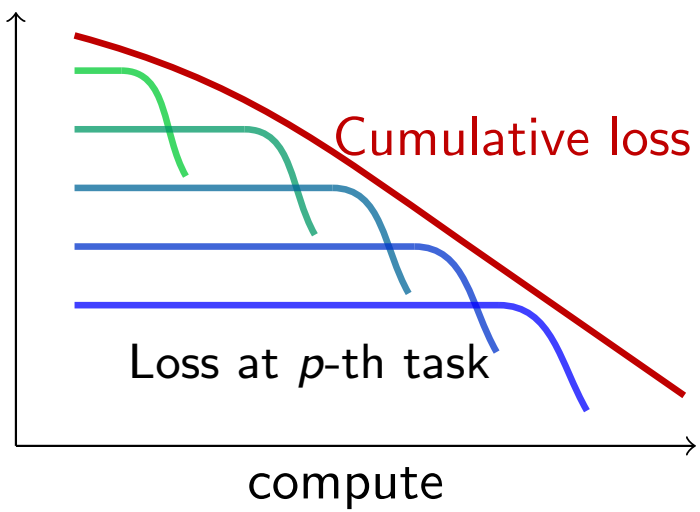


Neural Scaling Laws & Emergence

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 different 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: $f_*(\boldsymbol{x}) = \sigma_*(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle), \ \boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

- Requires learning the direction $\boldsymbol{\theta} \in \mathbb{R}^d$ and link function $\sigma_* : \mathbb{R} \to \mathbb{R}$.
  - Learning algorithm should adapt to **low-dimensional structure**.

Emergence in Gradient-based Feature Learning

Gaussian single-index model: $f_*(\boldsymbol{x}) = \sigma_*(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle), \ \boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

- Requires learning the direction $\boldsymbol{\theta} \in \mathbb{R}^d$ and link function $\sigma_* : \mathbb{R} \to \mathbb{R}$.
  - Learning algorithm should adapt to **low-dimensional structure**.

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

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

The information exponent of σ_* is $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

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

Emergence in Gradient-based Feature Learning

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

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

The information exponent of σ_* is $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

$$
\begin{aligned}
-\mathbb{E}[\nabla_{\boldsymbol{w}} \mathcal{L}(f_{\mathsf{NN}})] &\approx \mathbb{E}[\nabla_{\boldsymbol{w}}(f_{\mathsf{NN}}(\boldsymbol{x}) f_*(\boldsymbol{x}))] \\
&= \boldsymbol{\theta} \cdot \mathbb{E}[\sigma_*'(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)\sigma'(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)] + \boldsymbol{w} \cdot \mathbb{E}[\ldots] \quad \textit{Stein's lemma}
\end{aligned}
$$

Emergence in Gradient-based Feature Learning

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

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

The information exponent of σ_* is $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

$$
\begin{aligned}
-\mathbb{E}[\nabla_{\boldsymbol{w}} \mathcal{L}(f_{\mathrm{NN}})] &\approx \mathbb{E}[\nabla_{\boldsymbol{w}}(f_{\mathrm{NN}}(\boldsymbol{x}) f_*(\boldsymbol{x}))] \\
&= \boldsymbol{\theta} \cdot \mathbb{E}[\sigma_*'(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)\sigma'(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)] + \boldsymbol{w} \cdot \mathbb{E}[\ldots] \quad \textit{Stein's lemma} \\
&= \boldsymbol{\theta} \cdot \sum_{j=0}^{\infty} (j+1)^2 \alpha_{j+1}^* \beta_{j+1} \underbrace{\langle \boldsymbol{w}, \boldsymbol{\theta} \rangle^j}_{d^{-j/2} \text{ at initialization}} + \ldots \quad \textit{Hermite expansion}
\end{aligned}
$$

Emergence in Gradient-based Feature Learning

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

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

The information exponent of σ_* is $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

$$
\begin{aligned}
-\mathbb{E}[\nabla_{\boldsymbol{w}} \mathcal{L}(f_{\mathrm{NN}})] &\approx \mathbb{E}[\nabla_{\boldsymbol{w}}(f_{\mathrm{NN}}(\boldsymbol{x}) f_*(\boldsymbol{x}))] \\
&= \boldsymbol{\theta} \cdot \mathbb{E}[\sigma_*'(\langle \boldsymbol{x}, \boldsymbol{\theta} \rangle)\sigma'(\langle \boldsymbol{x}, \boldsymbol{w} \rangle)] + \boldsymbol{w} \cdot \mathbb{E}[\ldots] \quad \textit{Stein's lemma} \\
&= \boldsymbol{\theta} \cdot \sum_{j=0}^{\infty} (j+1)^2 \alpha_{j+1}^* \beta_{j+1} \underbrace{\langle \boldsymbol{w}, \boldsymbol{\theta} \rangle^j}_{d^{-j/2} \text{ at initialization}} + \ldots \quad \textit{Hermite expansion}
\end{aligned}
$$

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 $n \simeq T \gtrsim d^{\Theta(k)}$ samples and SGD steps.

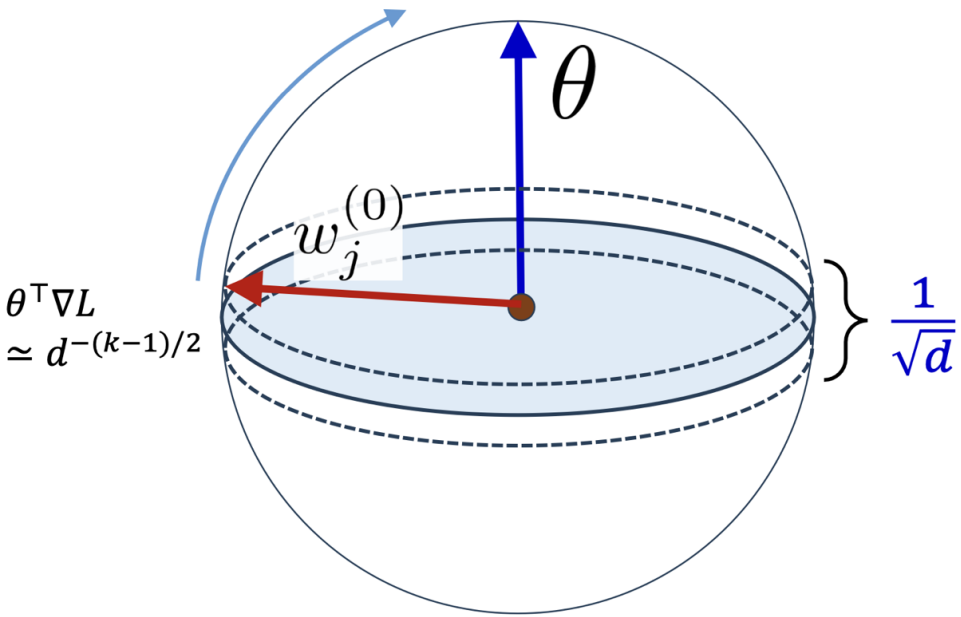
Emergence in Gradient-based Feature Learning

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

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

The information exponent of σ_* is $k = \mathrm{IE}(\sigma_*) = \min\{k \in \mathbb{N}_+ : \alpha_k^* \neq 0\}$.

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 $T \asymp d^{k-1}$ steps.
- **Descent Phase.** Loss sharply decreases in $T = \tilde{\Theta}(1)$ steps.

Learning Extensive-width Neural Network

Target Function: Width-P two-layer neural network

$$f_*(\boldsymbol{x}) = \sum_{m=1}^{P} a_p \cdot \sigma(\langle \boldsymbol{x}, \boldsymbol{\theta}_p \rangle), \quad \sum a_p^2 = 1, \ \{\boldsymbol{\theta}_p\}_{p=1}^{P} \text{ orthonormal.}$$

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

- σ with IE $k > 2$ $\Rightarrow$ *emergent* learning curve for each direction.
- Varying second-layer $\{a_p\}_{p=1}^{P}$ $\Rightarrow$ separation in length of search phase.

Learning Extensive-width Neural Network

Target Function: Width-P two-layer neural network

$$f_*(\boldsymbol{x}) = \sum_{m=1}^{P} a_p \cdot \sigma(\langle \boldsymbol{x}, \boldsymbol{\theta}_p \rangle), \quad \sum a_p^2 = 1, \ \{\boldsymbol{\theta}_p\}_{p=1}^{P} \text{ orthonormal.}$$

- **Extensive width.** $P \asymp d^{\gamma}$ for $\gamma \in [0, c], c > 0$.
  - Large number of tasks $\Rightarrow$ infinite-dimensional *effective dynamics*.
- **Large condition number.** $a_{\text{max}}/a_{\text{min}} = \Theta(\text{Poly}(P))$.
  - Covers **power-law decay** in second-layer $a_p \asymp p^{-\beta}, \beta \in [0, \infty)$.
- **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

$$f_*(\boldsymbol{x}) = \sum_{m=1}^{P} a_p \cdot \sigma(\langle \boldsymbol{x}, \boldsymbol{\theta}_p \rangle), \quad \sum a_p^2 = 1, \ \{\boldsymbol{\theta}_p\}_{p=1}^{P} \text{ orthonormal.}$$

- **Extensive width.** $P \asymp d^{\gamma}$ for $\gamma \in [0, c], c > 0$.
  - Large number of tasks $\Rightarrow$ infinite-dimensional *effective dynamics*.
- **Large condition number.** $a_{\text{max}}/a_{\text{min}} = \Theta(\text{Poly}(P))$.
  - Covers **power-law decay** in second-layer $a_p \asymp p^{-\beta}, \beta \in [0, \infty)$.
- **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 $\boldsymbol{\theta}_p$,
- smooth scaling law for MSE.

Main Theorem: Complexity of SGD Learning

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

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

Main Theorem: Complexity of SGD Learning

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

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

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

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \eta \nabla_{\boldsymbol{w}_i} (f_*(\boldsymbol{x}_t) - f(\boldsymbol{x}_t))^2, \quad \boldsymbol{x}_t \sim \mathcal{N}(0, \boldsymbol{I}_d).$$

Main Theorem: Complexity of SGD Learning

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

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

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

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \eta \nabla_{\boldsymbol{w}_i} (f_*(\boldsymbol{x}_t) - f(\boldsymbol{x}_t))^2, \quad \boldsymbol{x}_t \sim \mathcal{N}(0, \boldsymbol{I}_d).$$

Theorem (Complexity of SGD learning)

Assume $P \ll d^{0.1}$, $\bar{p} < P$. If we train a student NN with $m = \tilde{\Theta}(\bar{p})$ neurons using online SGD with $\eta \asymp \frac{a_{\bar{p}}}{d^{k/2}\mathrm{poly}(P)}$, then w.h.p.,

- *If $p < \bar{p}$, alignment with $\boldsymbol{\theta}_p$ emerges at* $T_p \asymp a_p^{-1} \cdot \eta^{-1} d^{k/2-1}$.

Main Theorem: Complexity of SGD Learning

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

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

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

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) - \eta \nabla_{\boldsymbol{w}_i} (f_*(\boldsymbol{x}_t) - f(\boldsymbol{x}_t))^2, \quad \boldsymbol{x}_t \sim \mathcal{N}(0, \boldsymbol{I}_d).$$

Theorem (Complexity of SGD learning)

Assume $P \ll d^{0.1}$, $\bar{p} < P$. If we train a student NN with $m = \tilde{\Theta}(\bar{p})$ neurons using online SGD with $\eta \asymp \frac{a_{\bar{p}}}{d^{k/2}\mathrm{poly}(P)}$, then w.h.p.,

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

Main Theorem: Complexity of SGD Learning

Theorem (Complexity of SGD learning)

Assume $P \ll d^{0.1}$, $\bar{p} < P$. *If we train a student NN with* $m = \tilde{\Theta}(\bar{p})$ *neurons using online SGD with* $\eta \asymp \frac{a_{\bar{p}}}{d^{k/2}\mathrm{poly}(P)}$, *then w.h.p.,*

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

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

Main Theorem: Complexity of SGD Learning

Theorem (Complexity of SGD learning)

Assume $P \ll d^{0.1}$, $\bar{p} < P$. *If we train a student NN with* $m = \tilde{\Theta}(\bar{p})$ *neurons using online SGD with* $\eta \asymp \frac{a_{\bar{p}}}{d^{k/2}\mathrm{poly}(P)}$, *then w.h.p.,*

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

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

- Prior works required sample size or compute *exponentially* large in the condition number: $n, P \asymp \exp\left(\frac{a_{\max}}{a_{\min}}\right)$ [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].

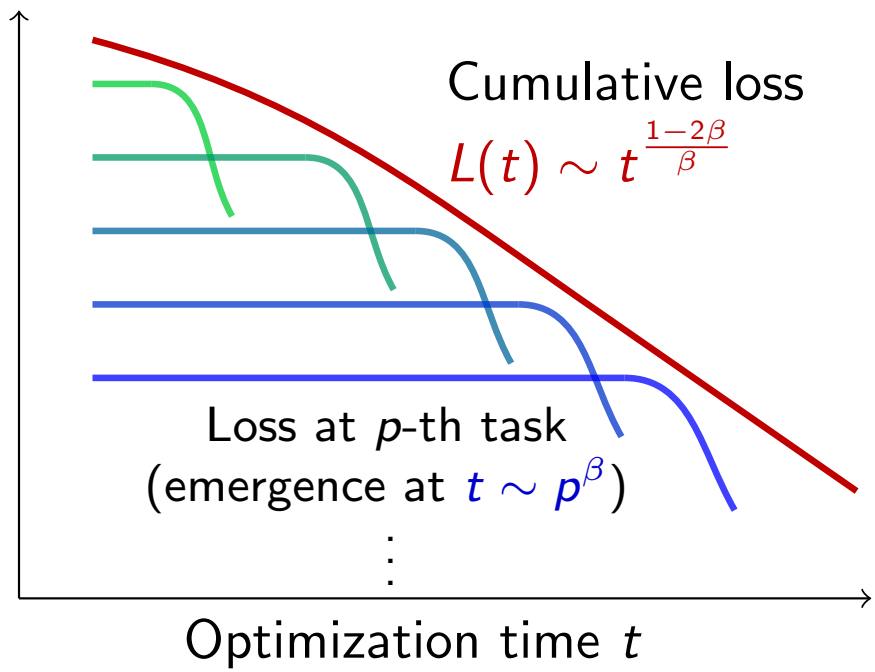
Neural Scaling Laws for Feature Learning

Corollary (Emergence & Scaling Laws)

Assume $a_p \asymp p^{-\beta}$ *for* $\beta > 1/2$ *and fixed learning rate* η*, then*

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

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



Neural Scaling Laws for Feature Learning

Corollary (Emergence & Scaling Laws)

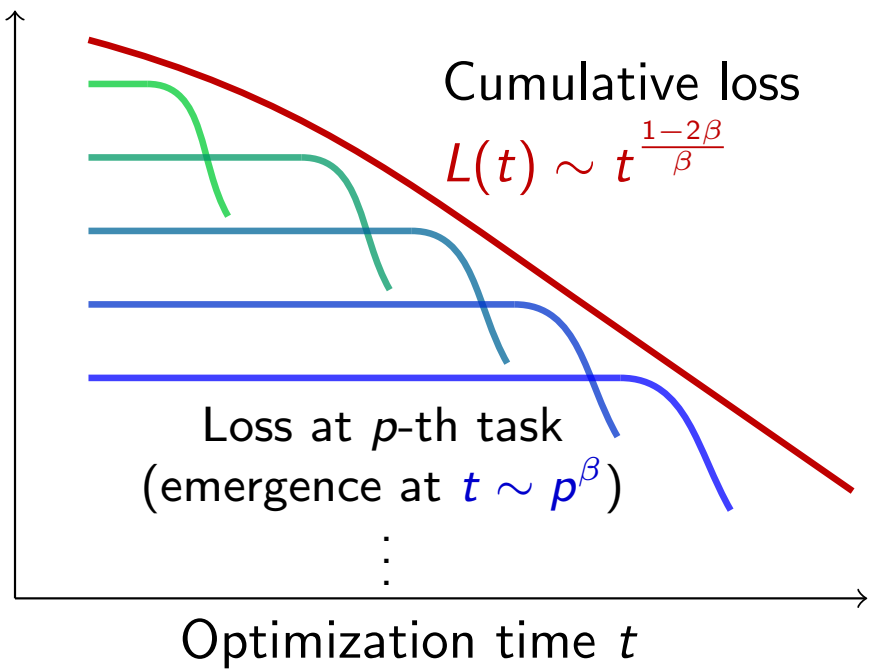
Assume $a_p \asymp p^{-\beta}$ *for* $\beta > 1/2$ *and fixed learning rate* η*, then*

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

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

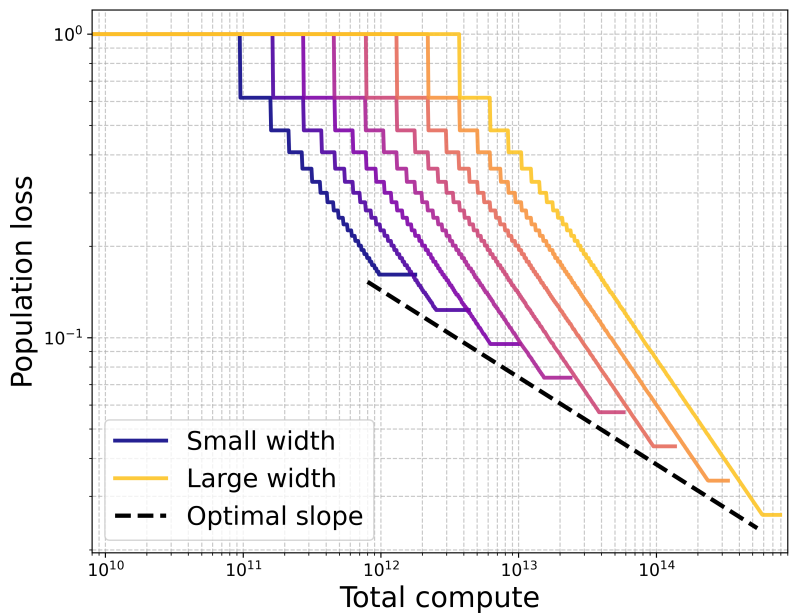
Intuition: Assume *decoupled* learning of different teacher components,

- Direction $\boldsymbol{\theta}_p$ learned at $T_p \propto p^{\beta} \eta^{-1} d^{k/2-1}$.
- $\mathcal{L}(t) \approx \sum_{p=1}^{P} a_p^2 \mathbb{I}\{t < T_p\} \Rightarrow \mathcal{L}(T_p) \approx \int_p^{\infty} s^{-2\beta} \mathrm{d}s \sim p^{1-2\beta}$.
- $m < P$ student neurons reaches $\sum_{s>m} a_s^2 \asymp m^{1-2\beta}$ error.

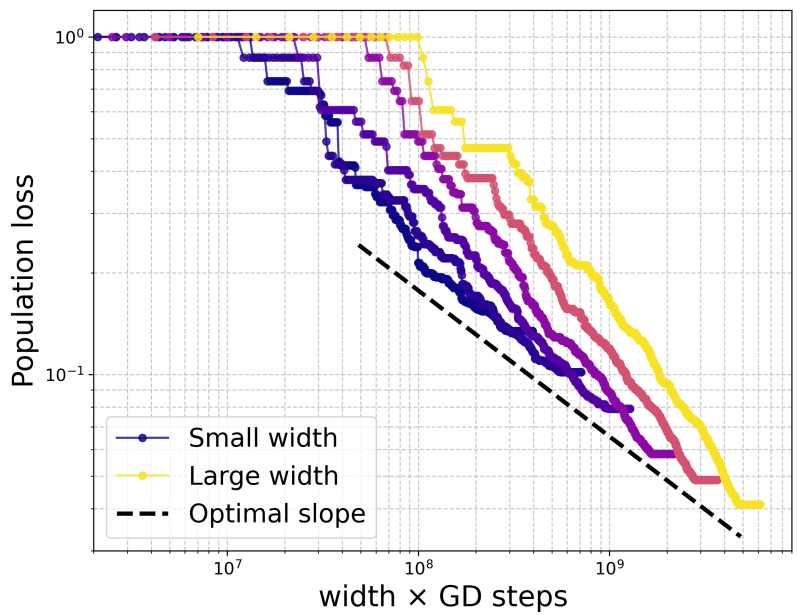


Neural Scaling Laws for Feature Learning

- $m^{1-2\beta}$ – approximation barrier.
  - Determined by the student network width.
- $\left(\eta t d^{1-k/2}\right)^{\frac{1-2\beta}{\beta}}$ – 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-η) 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: $\eta \asymp \frac{a_p}{d^{k/2}\mathrm{poly}(P)}$, $a_p \asymp p^{-\beta}$.
- Direction $\boldsymbol{\theta}_p$ now learned at $n = T_p \sim p^{\beta}\eta^{-1}d^{k/2-1} = p^{2\beta}d^{k-1}\mathrm{poly}(P)$.

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

$$\mathcal{L}(n, m) \sim \left(\frac{n}{d^{k-1}\mathrm{poly}(P)}\right)^{\frac{1-2\beta}{2\beta}} + m^{1-2\beta}$$

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

Proof Sketch

- **Decoupled gradient dynamics.**
  Let $\boldsymbol{w}_{\iota(p)}$ denote the neuron that eventually converged to direction $\boldsymbol{\theta}_p$, and $m_{p,q}(t) = \langle \boldsymbol{w}_p(t)/\|\boldsymbol{w}_p(t)\|, \boldsymbol{\theta}_q \rangle$ measures the overlap at time t.
  - **Claim 1 - decoupling**. When all $m^2_{\iota(p),q}$ $(q \neq p)$ are small, the learning of different directions can be approximately decoupled.
  - **Claim 2 - sharp transitions**. Since the norm of $\boldsymbol{w}_p$ grows rapidly after alignment is achieved, all $m^2_{\iota(p),q}$ $(q \neq p)$ remain small when the q-th direction is recovered by $\boldsymbol{w}_{\iota(q)}$ – after which $\boldsymbol{\theta}_q$ no longer affects the learning dynamics ("automatic deflation").

Proof Sketch

- **Decoupled gradient dynamics.**
  Let $\boldsymbol{w}_{\iota(p)}$ denote the neuron that eventually converged to direction $\boldsymbol{\theta}_p$, and $m_{p,q}(t) = \langle \boldsymbol{w}_p(t)/\|\boldsymbol{w}_p(t)\|, \boldsymbol{\theta}_q \rangle$ measures the overlap at time t.
  - **Claim 1 - decoupling**. When all $m^2_{\iota(p),q}$ $(q \neq p)$ are small, the learning of different directions can be approximately decoupled.
  - **Claim 2 - sharp transitions**. Since the norm of $\boldsymbol{w}_p$ grows rapidly after alignment is achieved, all $m^2_{\iota(p),q}$ $(q \neq p)$ remain small when the q-th direction is recovered by $\boldsymbol{w}_{\iota(q)}$ – after which $\boldsymbol{\theta}_q$ no longer affects the learning dynamics ("automatic deflation").

- **From gradient flow to online SGD.**
  - Martingale decomposition in [Ben Arous et al. 21] + a refined 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 σ).
  - ☹ Dynamics of different directions *cannot be decoupled*.
- *Companion work:* introduce algorithmic modification (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 σ).
  - ☹ Dynamics of different directions *cannot be decoupled*.
- *Companion work:* introduce algorithmic modification (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: $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \sum_{i=1}^{d} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^\top$, $\lambda_i \asymp i^{-\alpha}$.
  - Two-parameter scaling law? (source & capacity conditions)
- Beyond additive structure $\Rightarrow$ 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

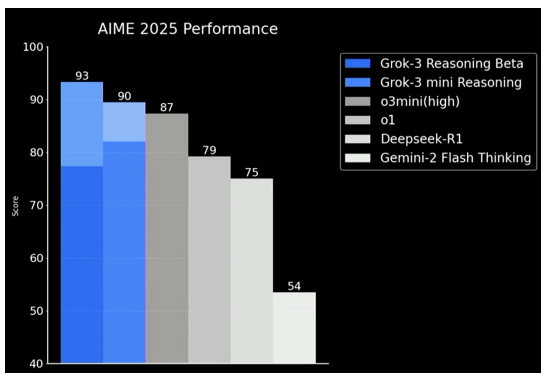
April 3, 2025

Motivation: multi-step reasoning tasks

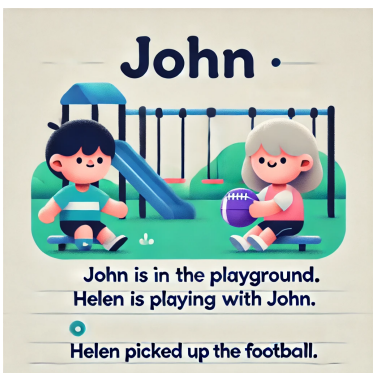
Large language models exhibits remarkable reasoning capabilities.

- 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."
- $f_{\text{location}}(\textbf{John}) = \text{playground}$, $f_{\text{with}}(\text{Helen}) = \textbf{John}$, $f_{\text{hold}}(\text{football}) = \text{Helen}$
- The model needs to compute $f_{\text{location}} \circ f_{\text{with}} \circ f_{\text{hold}}(\text{football})$ 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."
- $f_{\text{location}}(\textbf{John}) = \text{playground}$, $f_{\text{with}}(\text{Helen}) = \textbf{John}$, $f_{\text{hold}}(\text{football}) = \text{Helen}$
- The model needs to compute $f_{\text{location}} \circ f_{\text{with}} \circ f_{\text{hold}}(\text{football})$ and get playground.
- Note: (location, with, hold) are also function inputs.

Abstractly, the model can do the following function composition:

- Inputs: $(\sigma_1, \sigma_2, ..., \sigma_k, x)$
- Outputs: $f^{(k)}_{\sigma_k} \circ f^{(k-1)}_{\sigma_{k-1}} \cdots f^{(1)}_{\sigma_1}(x)$ 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.
- Inefficient 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.
- Inefficient 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.
- Inefficient 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 efficiently via gradient-based training?

Transformers and self-attention

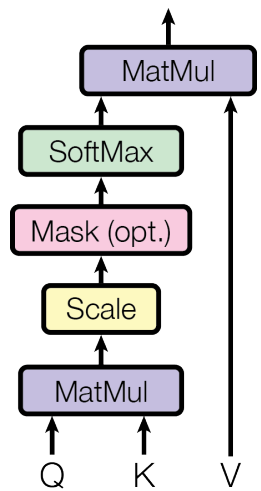
Definition (Self-attention)

The self-attention head $\mathrm{attn}(\cdot; W_{KQ}, W_{OV}) : \mathbb{R}^{d\times T} \to \mathbb{R}^{d\times T}$ is, with $W_{KQ}, W_{OV} \in \mathbb{R}^{d\times d}$

$$\mathrm{attn}(X; W_{KQ}, W_{OV}) = X + W_{OV} X \mathcal{S}(X^\top W_{KQ} X),$$

where the softmax function $\mathcal{S}$ is applied column-wise.

- Convex combination of all values $W_{OV}X$ with coefficient $\mathcal{S}(X^\top W_{KQ} X)_{(i,j)}$, 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

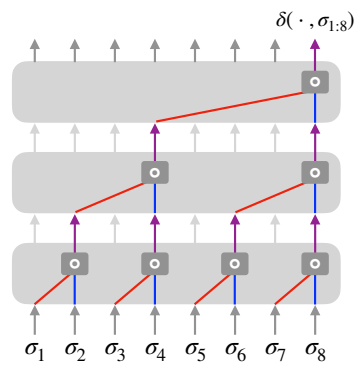
Can transformers compose (some) functions internally?

- 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)



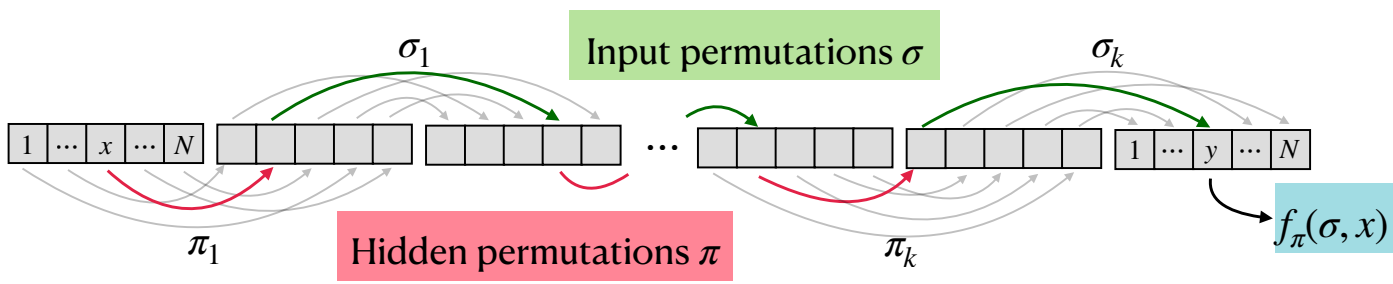
(Finite state automata)

Our task: k-fold composition

Definition. (k-fold composition)

- $\pi := (\pi_1, \ldots, \pi_k) \in (S_N)^k$ is a tuple of k hidden permutations.
- Input: $(\sigma, x) \in \mathcal{X} := (S_N)^k \times [N]$, where $\sigma = (\sigma_1, \ldots, \sigma_k)$, x is an index in $[N]$.
- Output $f_\pi : \mathcal{X} \to [N]$ is defined as

$$f_\pi(\sigma, x) := (\sigma_k \circ \pi_k \circ \sigma_{k-1} \circ \pi_{k-1} \circ \cdots \circ \sigma_1 \circ \pi_1)(x).$$



Remark: $\pi := (\pi_1, \ldots, \pi_k) \in (S_N)^k$ induces a function class $\mathcal{F}$.

Construction for k-fold composition

Theorem (Expressivity of Transformers)

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

Construction for k-fold composition

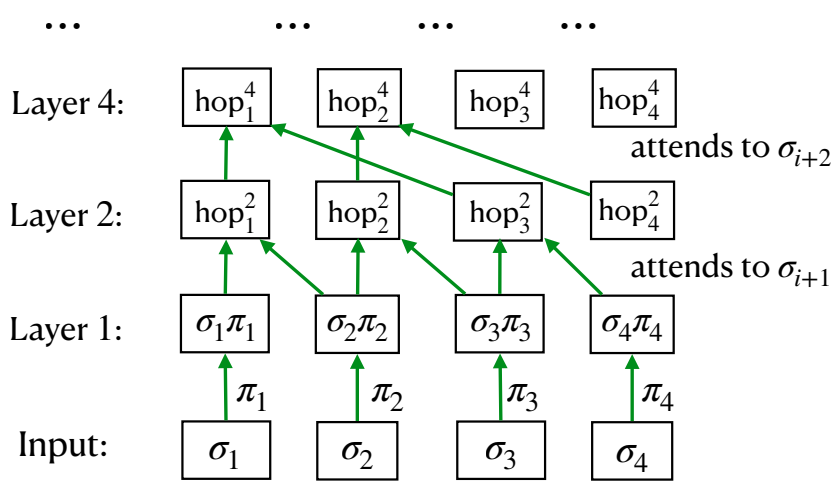
Theorem (Expressivity of Transformers)

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

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

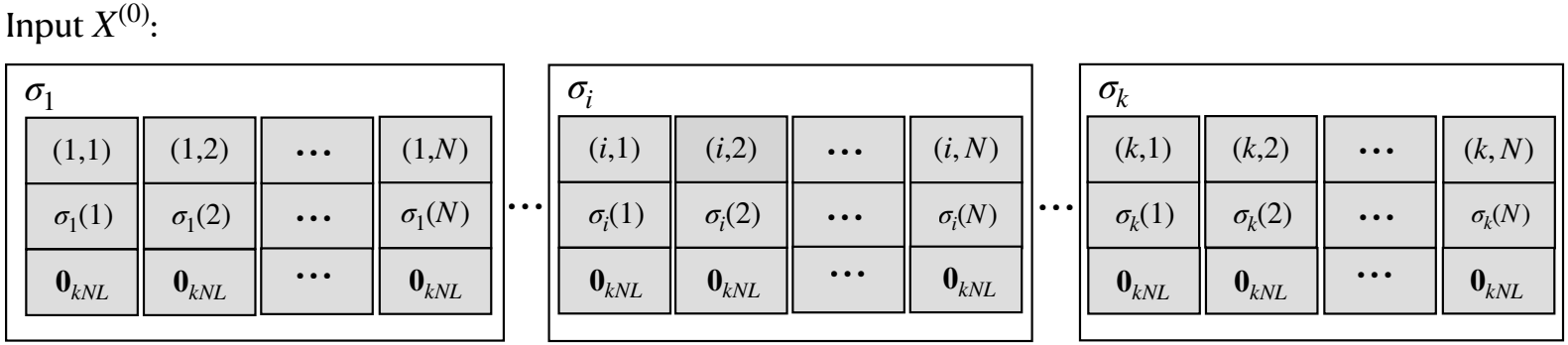
$$\text{hop}_i^2 := \sigma_{i+1} \circ \pi_{i+1} \circ \sigma_i \circ \pi_i$$

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

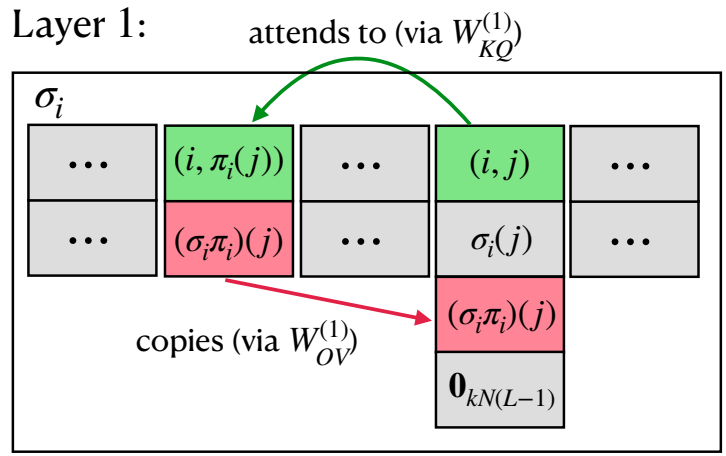


The parallel solution (Layer 1)

- Embedding for each $(i,j) \in [N] \times [k]$ stores one-hot embedding of (i,j) and $(i, \sigma_i(j))$



- The key-query matrix makes position (i,j) attend to position $(i, \pi(j))$.
- The value matrix copies $\text{hop}_i^1(j) := (\sigma_i \pi_i)(j)$ into the residual stream of (i,j)
- The first layer computes 1-hop!

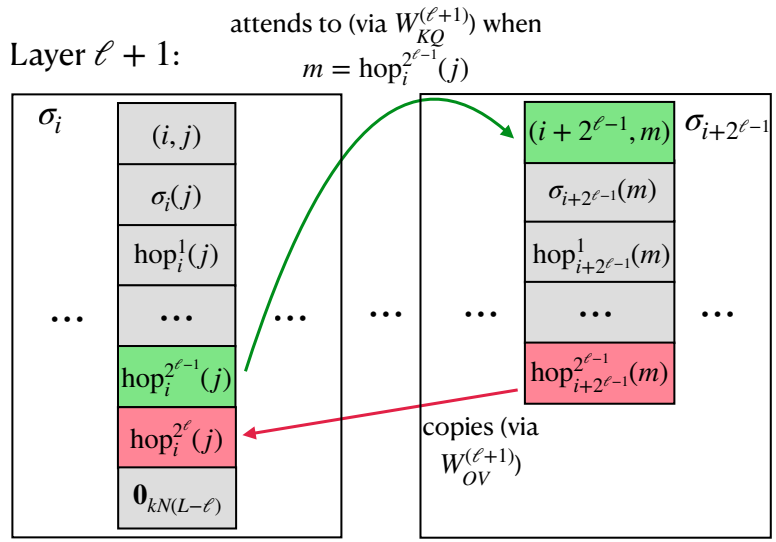


The parallel solution (Layer $\ell + 1$)

- Assume that the previous layer ℓ computes $\mathrm{hop}_i^{2^{\ell-1}}$ (i.e residual stream of (i, j) contains $\mathrm{hop}_i^{2^{\ell-1}}(j)$).
- The key-query matrix makes position (i, j) attend to position $(i + 2^{\ell-1}, \mathrm{hop}_{i^{2^{\ell-1}}}(j))$.
- Value matrix copies $\mathrm{hop}_{i+2^{\ell-1}}^{2^{\ell-1}}(\mathrm{hop}_i^{2^{\ell-1}}(j))$ into residual stream of (i, j).
- Thus the $(\ell + 1)$th layer computes

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

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



Computational hardness of k-fold composition

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

Computational hardness of k-fold composition

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

- **Negative result:** Statistical Query lower bound.
- **SQ framework:** Learner specifies query $q : \mathcal{X} \times [N] \to \mathbb{R}$ and tolerance τ. SQ oracle responds with $\hat{q}$, a noisy estimate of q

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

Computational hardness of k-fold composition

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

- **Negative result:** Statistical Query lower bound.
- **SQ framework:** Learner specifies query $q : \mathcal{X} \times [N] \to \mathbb{R}$ and tolerance τ. SQ oracle responds with $\hat{q}$, a noisy estimate of q

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

Theorem (SQ lower bound for k-fold composition)

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

- $\tau \approx n^{-1/2}$ by i.i.d. concentration heuristic, where n is the sample size.
  $\Rightarrow$ Either runtime ($\geq$ 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 efficiently!

- Stage 1: GD on the data with label $\mathrm{hop}_i^1 = \sigma_i \pi_i$ for $i \in [k]$.
- Stage ℓ: GD on the data with label $\mathrm{hop}_i^{2^{\ell-1}}$.
- We can learn the k-fold function with $\mathrm{poly}(N, k)$ samples!

Easy-to-hard: efficiency of curriculum learning

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

- Stage 1: GD on the data with label $\mathrm{hop}_i^1 = \sigma_i \pi_i$ for $i \in [k]$.
- Stage ℓ: GD on the data with label $\mathrm{hop}_i^{2^{\ell-1}}$.
- We can learn the k-fold function with $\mathrm{poly}(N, k)$ samples!

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 function.

Easy-to-hard: efficiency of curriculum learning

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

- Stage 1: GD on the data with label $\mathrm{hop}_i^1 = \sigma_i \pi_i$ for $i \in [k]$.
- Stage ℓ: GD on the data with label $\mathrm{hop}_i^{2^{\ell-1}}$.
- We can learn the k-fold function with $\mathrm{poly}(N, k)$ samples!

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 function.

Takeaway: $n \asymp N^{\Omega(k)}$ (**Exponential**) $\Rightarrow$ $n \asymp \mathrm{poly}(N, k)$ (**Polynomial**).

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

Learning with curriculum

Stage 1:

- In the construction, the first layer makes the (i, j) position attend to $(i, \pi_i(j))$ position.
- With 1-hop data, learning this first layer is easy!
- Gradient descent can correctly learn the constructed attention pattern, i.e. recover the hidden permutations π_i's.

Learning with curriculum

Stage 1:

- In the construction, the first layer makes the (i, j) position attend to $(i, \pi_i(j))$ position.
- With 1-hop data, learning this first layer is easy!
- Gradient descent can correctly learn the constructed attention pattern, i.e. recover the hidden permutations π_i's.

Stage ℓ ($\ell \geq 2$):

- ℓth layer learns to compose $\mathrm{hop}_i^{2^{\ell-2}}$ and $\mathrm{hop}_{i+2^{\ell-2}}^{2^{\ell-2}}$ computed in the previous layer.
- With $2^{\ell-1}$-hop data, GD correctly makes (i, j) position attend to $(i + 2^{\ell-2}, \mathrm{hop}_i^{2^{\ell-2}}(j))$ 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 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 $n \geq \tilde{\Omega}(k^4 N^6)$, the output of gradient descent training on the mixture of $\{1,2,4,\ldots,k\}$-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)

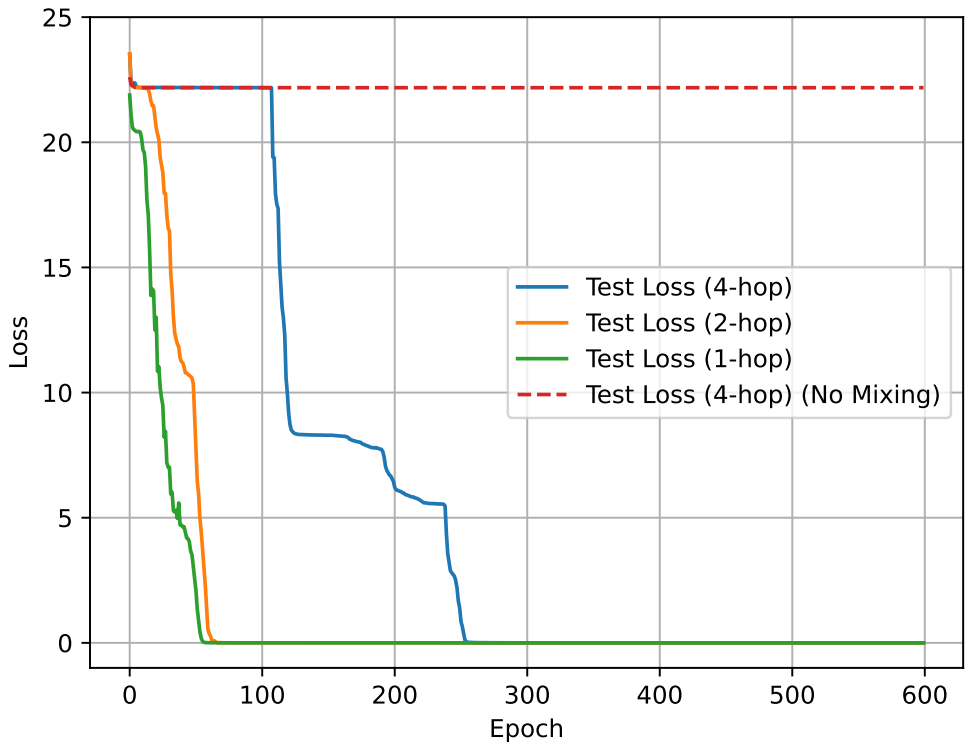
Assume $k = 2^{L-1}$, if the sample size $n \geq \tilde{\Omega}(k^4 N^6)$, the output of gradient descent training on the mixture of $\{1,2,4,\ldots,k\}$-fold data learns the k-fold composition function.

- 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 $n \asymp kN \log N$, while a computationally efficient SQ learner requires $n \geq N^{\Omega(k)}$.
- 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 $n \asymp kN \log N$, while a computationally efficient SQ learner requires $n \geq N^{\Omega(k)}$.
- Mirrors k-parity in d dimensions (requires d^k samples/compute).
- In both settings, efficient 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 $n \asymp kN \log N$, while a computationally efficient SQ learner requires $n \geq N^{\Omega(k)}$.
- Mirrors k-parity in d dimensions (requires d^k samples/compute).
- In both settings, efficient 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 efficient embedding.

References


Abbe, E., Cornacchia, E., and Lotfi, 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 efficiently 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 tradeoff: Limitations of log-precision transformers.
Transactions of the Association for Computational Linguistics, 11:531–545.


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