Associative Memories as a Building Block in Transformers

Alberto Bietti

Flatiron Institute, Simons Foundation

Cargèse, August 2025

Associative Memories as a Building Block in Transformers

Alberto Bietti

Flatiron Institute, Simons Foundation

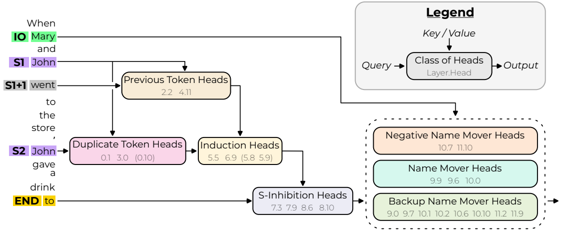
Cargèse, August 2025

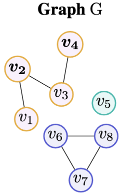
w/ V. Cabannes, E. Dohmatob, D. Bouchacourt, H. Jégou, L. Bottou (Meta), E. Nichani, J. Lee (Princeton), M. Vural (U Toronto), D. Wu (NYU/Flatiron)

What are Transformer LLMs doing?

Reasoning over context

- Circuits of attention heads (Elhage et al., 2021; Olsson et al., 2022; Wang et al., 2022)
- Many results on expressivity (e.g., circuits, formal languages, graph connectivity)
  - e.g., (Merrill et al., 2022; Liu et al., 2023; Sanford et al., 2023)





Task: Are v_2 and v_4 connected?

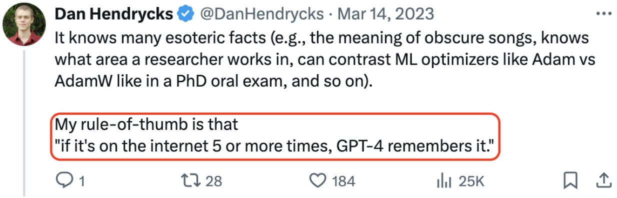
What are Transformer LLMs doing?

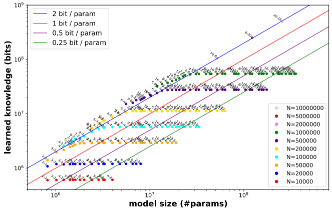
Reasoning over context

- Circuits of attention heads (Elhage et al., 2021; Olsson et al., 2022; Wang et al., 2022)
- Many results on expressivity (e.g., circuits, formal languages, graph connectivity)
  - e.g., (Merrill et al., 2022; Liu et al., 2023; Sanford et al., 2023)

Knowledge storage

- Memorization, factual recall, parameter scaling
  - e.g., (Geva et al., 2020; Allen-Zhu and Li, 2024)
- Allows higher-level reasoning





What are Transformer LLMs doing?

Reasoning over context

- Circuits of attention heads (Elhage et al., 2021; Olsson et al., 2022; Wang et al., 2022)
- Many results on expressivity (e.g., circuits, formal languages, graph connectivity)
  - e.g., (Merrill et al., 2022; Liu et al., 2023; Sanford et al., 2023)

Knowledge storage

- Memorization, factual recall, parameter scaling
  - e.g., (Geva et al., 2020; Allen-Zhu and Li, 2024)
- Allows higher-level reasoning

Goal: tractable model for both + training dynamics?

Transformer language models

Input: sequence of discrete tokens $(z_1, \ldots, z_T) \in [N]^T$

Transformer language models

Input: sequence of discrete tokens $(z_1, \ldots, z_T) \in [N]^T$

Embeddings

- input e_z, positional p_t, output u_y, in $\mathbb{R}^d$
- this talk: **fixed** to **random** init $\mathcal{N}(0, 1/d)$

Transformer language models

Input: sequence of discrete tokens $(z_1, \dots, z_T) \in [N]^T$

Embeddings

- input e_z, positional p_t, output u_y, in $\mathbb{R}^d$
- this talk: **fixed** to **random** init $\mathcal{N}(0, 1/d)$

Residual streams (Elhage et al., 2021)

- embed each token $z_t \in [N]$ as $x_t := e_{z_t} + p_t$

Transformer language models

Input: sequence of discrete tokens $(z_1, \dots, z_T) \in [N]^T$

Embeddings

- input e_z, positional p_t, output u_y, in $\mathbb{R}^d$
- this talk: **fixed** to **random** init $\mathcal{N}(0, 1/d)$

Residual streams (Elhage et al., 2021)

- embed each token $z_t \in [N]$ as $x_t := e_{z_t} + p_t$
- (causal) self-attention $x_t := x_t + \mathrm{MHSA}(x_t, x_{1:t})$

$$\mathrm{MHSA}(x_t, x_{1:t}) = \sum_{h=1}^{H} \sum_{s=1}^{t} \beta_s^h W_O^{h\top} W_V^h x_s, \quad \text{with } \beta_s^h = \frac{\exp({x_s}^\top W_K^{h\top} W_Q^h x_t)}{\sum_{s=1}^t \exp({x_s}^\top W_K^{h\top} W_Q^h x_t)}$$

where $W_K, W_Q, W_V, W_O \in \mathbb{R}^{d_h \times d}$ (key/query/value/output matrices)

Transformer language models

Input: sequence of discrete tokens $(z_1, \dots, z_T) \in [N]^T$

Embeddings

- input e_z, positional p_t, output u_y, in $\mathbb{R}^d$
- this talk: **fixed** to **random** init $\mathcal{N}(0, 1/d)$

Residual streams (Elhage et al., 2021)

- embed each token $z_t \in [N]$ as $x_t := e_{z_t} + p_t$
- (causal) self-attention $x_t := x_t + \mathrm{MHSA}(x_t, x_{1:t})$
- feed-forward $x_t := x_t + \mathrm{MLP}(x_t)$

$$\mathrm{MLP}(x_t) = V^\top \sigma(U x_t)$$

where $U, V \in \mathbb{R}^{m \times d}$, often $m = 4d$

Transformer language models

Input: sequence of discrete tokens $(z_1, \dots, z_T) \in [N]^T$

Embeddings

- input e_z, positional p_t, output u_y, in $\mathbb{R}^d$
- this talk: **fixed** to **random** init $\mathcal{N}(0, 1/d)$

Residual streams (Elhage et al., 2021)

- embed each token $z_t \in [N]$ as $x_t := e_{z_t} + p_t$
- (causal) self-attention $x_t := x_t + \mathrm{MHSA}(x_t, x_{1:t})$
- feed-forward $x_t := x_t + \mathrm{MLP}(x_t)$
- residual stream x_t is a sum of embeddings/"features"

Transformer language models

Input: sequence of discrete tokens $(z_1, \dots, z_T) \in [N]^T$

Embeddings

- input e_z, positional p_t, output u_y, in $\mathbb{R}^d$
- this talk: **fixed** to **random** init $\mathcal{N}(0, 1/d)$

Residual streams (Elhage et al., 2021)

- embed each token $z_t \in [N]$ as $x_t := e_{z_t} + p_t$
- (causal) self-attention $x_t := x_t + \mathrm{MHSA}(x_t, x_{1:t})$
- feed-forward $x_t := x_t + \mathrm{MLP}(x_t)$
- residual stream x_t is a sum of embeddings/"features"

Next-token prediction

- cross-entropy loss

$$\sum_{t<T} \ell(z_{t+1}; (u_j^\top x_t)_j)$$

Outline

1. Associative memories
2. Application to Transformers I: factual recall (Nichani et al., 2024)
3. Application to Transformers II: reasoning / retrieval (B. et al., 2023; Vural et al., 2025+)

Associative memories: background

Hopfield nets (Hopfield, 1982)

- Store N patterns $\xi_i \in \{\pm 1\}^d$ using Hebb's rule:

$$W = \sum_{i=1}^{N} \xi_i \xi_i^\top \in \mathbb{R}^{d \times d}$$

- Recover pattern from corrupted version by iterating: $x' = \mathrm{sign}(Wx)$

Associative memories: background

Hopfield nets (Hopfield, 1982)

- Store N patterns $\xi_i \in \{\pm 1\}^d$ using Hebb's rule:

$$W = \sum_{i=1}^{N} \xi_i \xi_i^\top \in \mathbb{R}^{d \times d}$$

- Recover pattern from corrupted version by iterating: $x' = \mathrm{sign}(Wx)$
- Iterates descend on a quadratic energy landscape $E(x) = -x^\top W x$
- Can store $N = \tilde{\Theta}(d)$ patterns (Amit et al., 1985; McEliece et al., 1988)

Associative memories: background

Hopfield nets (Hopfield, 1982)

- Store N patterns $\xi_i \in \{\pm 1\}^d$ using Hebb's rule:

$$W = \sum_{i=1}^{N} \xi_i \xi_i^\top \in \mathbb{R}^{d \times d}$$

- Recover pattern from corrupted version by iterating: $x' = \mathrm{sign}(Wx)$
- Iterates descend on a quadratic energy landscape $E(x) = -x^\top W x$
- Can store $N = \tilde{\Theta}(d)$ patterns (Amit et al., 1985; McEliece et al., 1988)

Modern Hopfield nets (a.k.a dense associative memories)

- Improve capacity through higher-order energy function
  - (Krotov and Hopfield, 2016; Demircigil et al., 2017; Lucibello and Mézard, 2024)
- e.g., capacity d^{k-1} when using energy $E(x) = -\sum_{i=1}^{N} (\xi_i^\top x)^k$

Associative memories: background

Hopfield nets (Hopfield, 1982)

- Store N patterns $\xi_i \in \{\pm 1\}^d$ using Hebb's rule:

$$W = \sum_{i=1}^{N} \xi_i \xi_i^\top \in \mathbb{R}^{d \times d}$$

- Recover pattern from corrupted version by iterating: $x' = \operatorname{sign}(Wx)$
- Iterates descend on a quadratic energy landscape $E(x) = -x^\top W x$
- Can store $N = \tilde{\Theta}(d)$ patterns (Amit et al., 1985; McEliece et al., 1988)

Modern Hopfield nets (a.k.a dense associative memories)

- Improve capacity through higher-order energy function
  - (Krotov and Hopfield, 2016; Demircigil et al., 2017; Lucibello and Mézard, 2024)
- e.g., capacity d^{k-1} when using energy $E(x) = -\sum_{i=1}^{N} (\xi_i^\top x)^k$

Attention as associative memory

- Softmax attention as one step retrieval in dense associative memory over *context*
  - Ramsauer et al. (2020); Smart, B., and Sengupta (2025): emerges from in-context denoising

Transformer weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{e_z\}_{z \in \mathcal{Z}}$ and $\{u_y\}_{y \in \mathcal{Y}}$:

$$\|e_z\| \approx 1 \quad \text{and} \quad e_z^\top e_{z'} \approx 0$$
$$\|u_y\| \approx 1 \quad \text{and} \quad u_y^\top u_{y'} \approx 0$$

Transformer weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{e_z\}_{z \in \mathcal{Z}}$ and $\{u_y\}_{y \in \mathcal{Y}}$:

$$\|e_z\| \approx 1 \quad \text{and} \quad {e_z}^\top e_{z'} \approx 0$$
$$\|u_y\| \approx 1 \quad \text{and} \quad {u_y}^\top u_{y'} \approx 0$$

- Consider **pairwise associations** $(z, y) \in \mathcal{M}$ with **weights** α_{zy} and define:

$$W = \sum_{(z,y) \in \mathcal{M}} \alpha_{zy} u_y {e_z}^\top$$

Transformer weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{e_z\}_{z \in \mathcal{Z}}$ and $\{u_y\}_{y \in \mathcal{Y}}$:

$$\|e_z\| \approx 1 \quad \text{and} \quad {e_z}^\top e_{z'} \approx 0$$
$$\|u_y\| \approx 1 \quad \text{and} \quad {u_y}^\top u_{y'} \approx 0$$

- Consider **pairwise associations** $(z, y) \in \mathcal{M}$ with **weights** α_{zy} and define:

$$W = \sum_{(z,y) \in \mathcal{M}} \alpha_{zy} u_y {e_z}^\top \quad \Longrightarrow \quad {u_y}^\top W e_z \approx \alpha_{zy}$$

Transformer weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{e_z\}_{z\in\mathcal{Z}}$ and $\{u_y\}_{y\in\mathcal{Y}}$:

$$\|e_z\| \approx 1 \quad \text{and} \quad {e_z}^\top e_{z'} \approx 0$$
$$\|u_y\| \approx 1 \quad \text{and} \quad {u_y}^\top u_{y'} \approx 0$$

- Consider **pairwise associations** $(z, y) \in \mathcal{M}$ with **weights** α_{zy} and define:

$$W = \sum_{(z,y)\in\mathcal{M}} \alpha_{zy} u_y {e_z}^\top \quad \Longrightarrow \quad {u_y}^\top W e_z \approx \alpha_{zy}$$

Examples in Transformers

- e_z, u_y are input/output/positional embeddings, or intermediate representations
- Logits in attention heads: $x_k^\top W_{KQ} x_q$
- Logits in next-token prediction: $u_y^\top U\sigma(Vx_t)$ or $u_y^\top W_{OV} x_k$

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = u_k{}^\top W e_z,$$

with ℓ the ***cross-entropy loss*** *and e_z, u_k input/output embeddings.*

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = {u_k}^\top W e_z,$$

with ℓ the ***cross-entropy loss*** *and e_z, u_k input/output embeddings. Then,*

$$\nabla L(W) = \sum_{k=1}^{K} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) u_k {e_z}^\top]$$

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = {u_k}^\top W e_z,$$

*with ℓ the **cross-entropy loss** and e_z, u_k input/output embeddings. Then,*

$$\nabla L(W) = \sum_{k=1}^{K} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) u_k {e_z}^\top]$$

- **Example**: $z \sim \mathrm{Unif}([N])$, $y = f_*(z)$

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = {u_k}^\top W e_z,$$

with ℓ the ***cross-entropy loss*** *and e_z, u_k input/output embeddings. Then,*

$$\nabla L(W) = \sum_{k=1}^{K} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) u_k {e_z}^\top]$$

- **Example**: $z \sim \mathrm{Unif}([N])$, $y = f_*(z)$
  - After **one gradient step** on the population loss, assuming near-orthonormal embeddings

$$W_1 = \frac{\eta}{N} \sum_{z,k} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right) u_k {e_z}^\top \quad \implies \quad {u_k}^\top W_1 e_z \approx \frac{\eta}{N} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right)$$

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = u_k{}^\top W e_z,$$

*with ℓ the **cross-entropy loss** and e_z, u_k input/output embeddings. Then,*

$$\nabla L(W) = \sum_{k=1}^{K} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) u_k e_z{}^\top]$$

- **Example**: $z \sim \mathrm{Unif}([N])$, $y = f_*(z)$
  - After **one gradient step** on the population loss, assuming near-orthonormal embeddings

$$W_1 = \frac{\eta}{N} \sum_{z,k} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right) u_k e_z{}^\top \quad \Longrightarrow \quad u_k{}^\top W_1 e_z \approx \frac{\eta}{N} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right)$$

  - **Corollary**: $\hat{f}(z) = \arg\max_k u_k{}^\top W_1 e_z$ has near-perfect accuracy

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = {u_k}^\top W e_z,$$

*with ℓ the **cross-entropy loss** and e_z, u_k input/output embeddings. Then,*

$$\nabla L(W) = \sum_{k=1}^{K} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) u_k {e_z}^\top]$$

- **Example**: $z \sim \mathrm{Unif}([N])$, $y = f_*(z)$
  - After **one gradient step** on the population loss, assuming near-orthonormal embeddings

$$W_1 = \frac{\eta}{N} \sum_{z,k} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right) u_k {e_z}^\top \quad \implies \quad {u_k}^\top W_1 e_z \approx \frac{\eta}{N} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right)$$

  - **Corollary**: $\hat{f}(z) = \arg\max_k {u_k}^\top W_1 e_z$ has near-perfect accuracy
- More generally, replace u_k by “backward” vector

Gradient associative memories

Lemma (Gradients as memories, B. et al., 2023)

Let p be a data distribution over $(z, y) \in [N]^2$, and consider the loss

$$L(W) = \mathbb{E}_{(z,y)\sim p}[\ell(y, F_W(z))], \quad F_W(z)_k = u_k^\top W e_z,$$

*with ℓ the **cross-entropy loss** and e_z, u_k input/output embeddings. Then,*

$$\nabla L(W) = \sum_{k=1}^{K} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) u_k e_z^\top]$$

- **Example**: $z \sim \mathrm{Unif}([N])$, $y = f_*(z)$
  - After **one gradient step** on the population loss, assuming near-orthonormal embeddings

$$W_1 = \frac{\eta}{N} \sum_{z,k} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right) u_k e_z^\top \quad \implies \quad u_k^\top W_1 e_z \approx \frac{\eta}{N} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right)$$

  - **Corollary**: $\hat{f}(z) = \arg\max_k u_k^\top W_1 e_z$ has near-perfect accuracy
- More generally, replace u_k by "backward" vector

Note: related to (Ba et al., 2022; Damian et al., 2022; Dandi et al., 2023; Yang and Hu, 2021)

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d} I)$

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$, consider "one gradient step" solution

$$W = \sum_{z=1}^{N} u_{f^*(z)} e_z{}^\top \in \mathbb{R}^{d \times d}$$

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$, consider "one gradient step" solution

$$W = \sum_{z=1}^{N} u_{f^*(z)} e_z{}^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $\arg\max_y \gamma_{z,y} = f^*(z)$ for all z?

$$\gamma_{z,y} := u_y{}^\top W e_z = \sum_{z'} u_y^\top u_{f^*(z')} e_z^\top e_{z'}$$

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$, consider "one gradient step" solution

$$W = \sum_{z=1}^{N} u_{f^*(z)} e_z^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $\arg\max_y \gamma_{z,y} = f^*(z)$ for all z?

$$\gamma_{z,y} := u_y^\top W e_z = \sum_{z'} u_y^\top u_{f^*(z')} e_z^\top e_{z'}$$

$$\mathbb{E}[\gamma_{z,y}] = \begin{cases} 1, & \text{if } y = f^*(z) \\ 0, & \text{otherwise.} \end{cases} \qquad \mathrm{Var}[\gamma_{z,y}] \lesssim \frac{|\{f^*(z') = y\}|}{d} + \frac{|\{f^*(z') \neq y\}|}{d^2} \overset{?}{\lesssim} 1$$

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d}I)$
- For some $f^* : [N] \to [M]$, consider "one gradient step" solution

$$W = \sum_{z=1}^{N} u_{f^*(z)} e_z{}^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $\arg\max_y \gamma_{z,y} = f^*(z)$ for all z?

$$\gamma_{z,y} := u_y{}^\top W e_z = \sum_{z'} u_y^\top u_{f^*(z')} e_z^\top e_{z'}$$

$$\mathbb{E}[\gamma_{z,y}] = \begin{cases} 1, & \text{if } y = f^*(z) \\ 0, & \text{otherwise.} \end{cases} \qquad \mathrm{Var}[\gamma_{z,y}] \lesssim \frac{|\{f^*(z') = y\}|}{d} + \frac{|\{f^*(z') \neq y\}|}{d^2} \overset{?}{\lesssim} 1$$

- **Examples**: (Cabannes, Dohmatob, and B., 2024a; Nichani, Lee, and B., 2024)
  - f^* injective: can store up to $N \approx d^2$ associations (much better than one hot!)

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$, consider "one gradient step" solution

$$W = \sum_{z=1}^{N} u_{f^*(z)} e_z{}^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $\arg\max_y \gamma_{z,y} = f^*(z)$ for all z?

$$\gamma_{z,y} := u_y{}^\top W e_z = \sum_{z'} u_y^\top u_{f^*(z')} e_z^\top e_{z'}$$

$$\mathbb{E}[\gamma_{z,y}] = \begin{cases} 1, & \text{if } y = f^*(z) \\ 0, & \text{otherwise.} \end{cases} \qquad \mathrm{Var}[\gamma_{z,y}] \lesssim \frac{|\{f^*(z') = y\}|}{d} + \frac{|\{f^*(z') \neq y\}|}{d^2} \overset{?}{\lesssim} 1$$

- **Examples**: (Cabannes, Dohmatob, and B., 2024a; Nichani, Lee, and B., 2024)
  - f^* injective: can store up to $N \approx d^2$ associations (much better than one hot!)
  - $f^*(z) \in \{0, 1\}$: can store up to $N \approx d$ associations

Capacity: Intuition

- Random embeddings $e_z, u_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$, consider "one gradient step" solution

$$W = \sum_{z=1}^{N} u_{f^*(z)} e_z^\top \in \mathbb{R}^{d \times d}$$

- When can we recover $\arg\max_y \gamma_{z,y} = f^*(z)$ for all z?

$$\gamma_{z,y} := u_y^\top W e_z = \sum_{z'} u_y^\top u_{f^*(z')} e_z^\top e_{z'}$$

$$\mathbb{E}[\gamma_{z,y}] = \begin{cases} 1, & \text{if } y = f^*(z) \\ 0, & \text{otherwise.} \end{cases} \qquad \mathrm{Var}[\gamma_{z,y}] \lesssim \frac{|\{f^*(z') = y\}|}{d} + \frac{|\{f^*(z') \neq y\}|}{d^2} \overset{?}{\lesssim} 1$$

- **Examples**: (Cabannes, Dohmatob, and B., 2024a; Nichani, Lee, and B., 2024)
  - f^* injective: can store up to $N \approx d^2$ associations (much better than one hot!)
  - $f^*(z) \in \{0, 1\}$: can store up to $N \approx d$ associations
  - Scaling laws: store the most frequent tokens with under-parameterized model

Capacity $\approx$ number of parameters

Low-rank

- $W = W_1^\top W_2$, with $W_1, W_2 \in \mathbb{R}^{m \times d}$ (e.g., key-query or output-value matrices)
- can store $N \approx md$ associations when $m \leq d$
- construction: random W_1, one step on W_2

(Nichani, Lee, and B., 2024), related to Krotov and Hopfield (2016); Demircigil et al. (2017)

Capacity $\approx$ number of parameters

Low-rank

- $W = W_1^\top W_2$, with $W_1, W_2 \in \mathbb{R}^{m \times d}$ (e.g., key-query or output-value matrices)
- can store $N \approx md$ associations when $m \leq d$
- construction: random W_1, one step on W_2

Non-linear MLP

- $\hat{f}(z) = \arg\max_y u_y{}^\top W_1 \sigma(W_2^\top e_z)$, $W_1, W_2 \in \mathbb{R}^{d \times m}$
- can store $N \approx md$ associations for any width m
- construction: using Hermite polynomials of degree $\approx \log N / \log d$ in kernel regime

(Nichani, Lee, and B., 2024), related to Krotov and Hopfield (2016); Demircigil et al. (2017)

Capacity $\approx$ number of parameters

Low-rank

- $W = W_1^\top W_2$, with $W_1, W_2 \in \mathbb{R}^{m \times d}$ (e.g., key-query or output-value matrices)
- can store $N \approx md$ associations when $m \leq d$
- construction: random W_1, one step on W_2

Non-linear MLP

- $\hat{f}(z) = \arg\max_y u_y{}^\top W_1 \sigma(W_2^\top e_z)$, $W_1, W_2 \in \mathbb{R}^{d \times m}$
- can store $N \approx md$ associations for any width m
- construction: using Hermite polynomials of degree $\approx \log N / \log d$ in kernel regime

Multi-input

- $\hat{f}(z_1, z_2) = \arg\max_y u_y{}^\top W_1 \sigma(W_2^\top (e_{z_1} + \tilde{e}_{z_2}))$
- also $N \approx md$ capacity

(Nichani, Lee, and B., 2024), related to Krotov and Hopfield (2016); Demircigil et al. (2017)

Capacity $\approx$ number of parameters

Low-rank

- $W = W_1^\top W_2$, with $W_1, W_2 \in \mathbb{R}^{m \times d}$ (e.g., key-query or output-value matrices)
- can store $N \approx md$ associations when $m \leq d$
- construction: random W_1, one step on W_2

Non-linear MLP

- $\hat{f}(z) = \arg\max_y u_y{}^\top W_1 \sigma(W_2^\top e_z)$, $W_1, W_2 \in \mathbb{R}^{d \times m}$
- can store $N \approx md$ associations for any width m
- construction: using Hermite polynomials of degree $\approx \log N / \log d$ in kernel regime

Multi-input

- $\hat{f}(z_1, z_2) = \arg\max_y u_y{}^\top W_1 \sigma(W_2^\top (e_{z_1} + \tilde{e}_{z_2}))$
- also $N \approx md$ capacity

Note: matches information-theoretic lower bounds

(Nichani, Lee, and B., 2024), related to Krotov and Hopfield (2016); Demircigil et al. (2017)

Outline

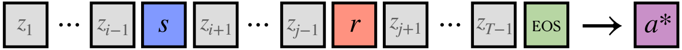
Toy model of factual recall

$$z_1 \ \cdots \ z_{i-1} \ \ s \ \ z_{i+1} \ \cdots \ z_{j-1} \ \ r \ \ z_{j+1} \ \cdots \ z_{T-1} \ \ \text{EOS} \ \longrightarrow \ a^*$$

The capital of France is Paris

- $s \in \mathcal{S}$: subject token
- $r \in \mathcal{R}$: relation token
- $a^*(s, r) \in \mathcal{A}_r$: attribute/fact to be stored
- $z_i \in \mathcal{N}$: noise tokens

Toy model of factual recall



The capital of France is Paris

- $s \in \mathcal{S}$: subject token
- $r \in \mathcal{R}$: relation token
- $a^*(s, r) \in \mathcal{A}_r$: attribute/fact to be stored
- $z_i \in \mathcal{N}$: noise tokens

Q: How many parameters do Transformers need to solve this?

How many parameters do we need?

- One-layer Transformer, with or without MLP, random embeddings
- Embedding dimension d, head dimension d_h, MLP width m, H heads

Theorem (Nichani et al., 2024, informal)

- *Attention + MLP:* $Hd_h \gtrsim S + R$ *and* $md \gtrsim SR$ *succeeds*
- *Attention-only:* $d \gtrsim R + A_{\max}$ *and* $Hd_h \gtrsim S$ *succeeds* $(A_{\max} := \max_r |\mathcal{A}_r|)$

How many parameters do we need?

- One-layer Transformer, with or without MLP, random embeddings
- Embedding dimension d, head dimension d_h, MLP width m, H heads

Theorem (Nichani et al., 2024, informal)

- *Attention + MLP:* $Hd_h \gtrsim S + R$ *and* $md \gtrsim SR$ *succeeds*
- *Attention-only:* $d \gtrsim R + A_{\max}$ *and* $Hd_h \gtrsim S$ *succeeds* ($A_{\max} := \max_r |\mathcal{A}_r|$)

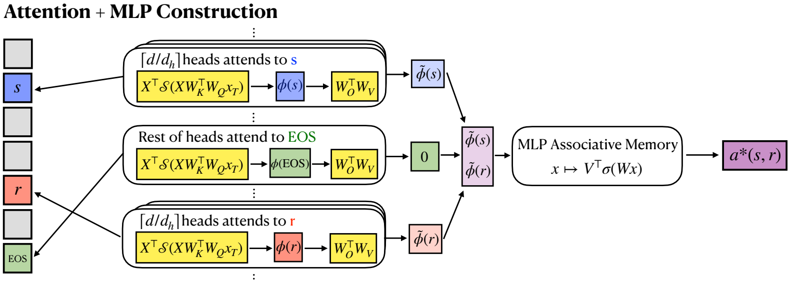
- Total parameters scale with number of facts SR (up to $A_{\max}$)
- Constructions are based on associative memories
- Attention-only needs large enough d
- Noise is negligible (log factors)

How many parameters do we need?

- One-layer Transformer, with or without MLP, random embeddings
- Embedding dimension d, head dimension d_h, MLP width m, H heads

Theorem (Nichani et al., 2024, informal)

- *Attention + MLP:* $Hd_h \gtrsim S + R$ *and* $md \gtrsim SR$ *succeeds*
- *Attention-only:* $d \gtrsim R + A_{\max}$ *and* $Hd_h \gtrsim S$ *succeeds* ($A_{\max} := \max_r |\mathcal{A}_r|$)

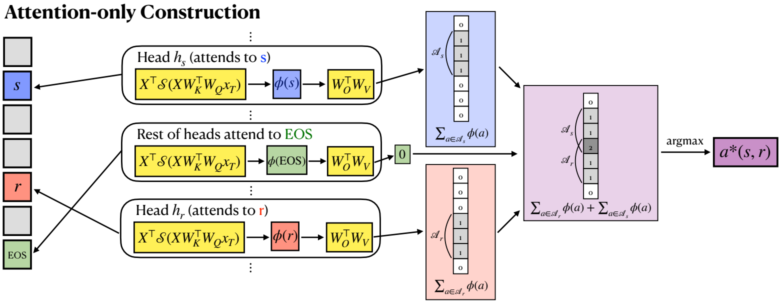


How many parameters do we need?

- One-layer Transformer, with or without MLP, random embeddings
- Embedding dimension d, head dimension d_h, MLP width m, H heads

Theorem (Nichani et al., 2024, informal)

- *Attention + MLP:* $Hd_h \gtrsim S + R$ *and* $md \gtrsim SR$ *succeeds*
- *Attention-only:* $d \gtrsim R + A_{\max}$ *and* $Hd_h \gtrsim S$ *succeeds* ($A_{\max} := \max_r |\mathcal{A}_r|$)

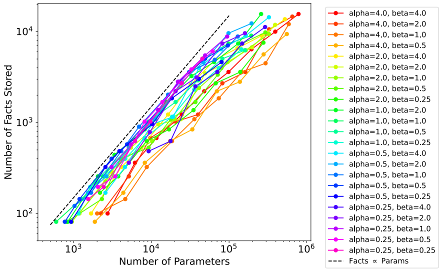


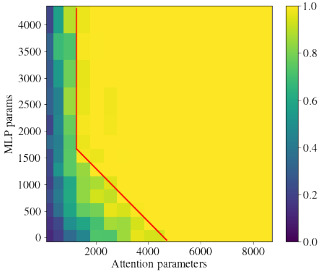
How many parameters do we need?

- One-layer Transformer, with or without MLP, random embeddings
- Embedding dimension d, head dimension d_h, MLP width m, H heads

Theorem (Nichani et al., 2024, informal)

- *Attention + MLP:* $Hd_h \gtrsim S + R$ *and* $md \gtrsim SR$ *succeeds*
- *Attention-only:* $d \gtrsim R + A_{\max}$ *and* $Hd_h \gtrsim S$ *succeeds* ($A_{\max} := \max_r |\mathcal{A}_r|$)





Training dynamics

- One-layer Transformer with **linear attention** and **one-hot** embeddings
- Gradient flow with initialization $W_{OV}(a,z), w_{KQ}(z) \approx \alpha > 0$

Theorem (Nichani et al., 2024, informal)

- *We have global convergence to zero loss*
- *There is an intermediate phase where the model predicts with $p(a|r)$ instead of $p(a|s,r)$*

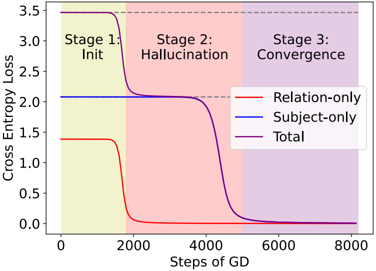
Training dynamics

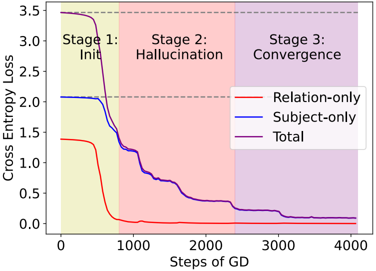
- One-layer Transformer with **linear attention** and **one-hot** embeddings
- Gradient flow with initialization $W_{OV}(a, z), w_{KQ}(z) \approx \alpha > 0$

Theorem (Nichani et al., 2024, informal)

- *We have global convergence to zero loss*
- *There is an intermediate phase where the model predicts with $p(a|r)$ instead of $p(a|s, r)$*

- Intermediate phase corresponds to **hallucination** (over $\mathcal{A}_r$, ignoring s)



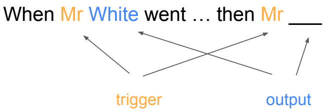


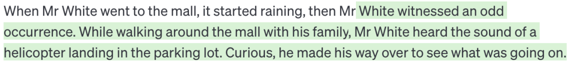
Outline

1. Associative memories
2. Application to Transformers I: factual recall (Nichani et al., 2024)
3. Application to Transformers II: reasoning / retrieval (B. et al., 2023; Vural et al., 2025+)

The bigram data model for in-context reasoning

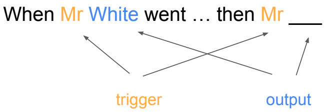
Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)





The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)

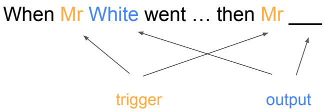


When Mr White went to the mall, it started raining, then Mr White witnessed an odd occurrence. While walking around the mall with his family, Mr White heard the sound of a helicopter landing in the parking lot. Curious, he made his way over to see what was going on.

Fix **trigger tokens**: $q_1, \ldots, q_K$

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr White went to the mall, it started raining, then Mr White witnessed an odd occurrence. While walking around the mall with his family, Mr White heard the sound of a helicopter landing in the parking lot. Curious, he made his way over to see what was going on.

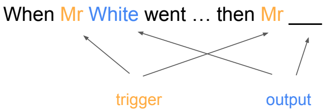
Fix **trigger tokens**: $q_1, \ldots, q_K$

Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Output tokens**: $o_k \sim \pi_o(\cdot | q_k)$ (*random*)

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr White went to the mall, it started raining, then Mr White witnessed an odd occurrence. While walking around the mall with his family, Mr White heard the sound of a helicopter landing in the parking lot. Curious, he made his way over to see what was going on.

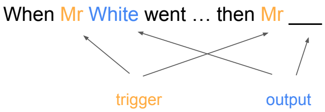
Fix **trigger tokens**: $q_1, \dots, q_K$
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Output tokens**: $o_k \sim \pi_o(\cdot|q_k)$ (*random*)
- **Sequence-specific Markov model**: $z_1 \sim \pi_1$, $z_t|z_{t-1} \sim p(\cdot|z_{t-1})$ with

$$p(j|i) = \begin{cases} \mathbb{1}\{j = o_k\}, & \text{if } i = q_k, \quad k = 1, \dots, K \\ \pi_b(j|i), & \text{o/w.} \end{cases}$$

The bigram data model for in-context reasoning

Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr White went to the mall, it started raining, then Mr White witnessed an odd occurrence. While walking around the mall with his family, Mr White heard the sound of a helicopter landing in the parking lot. Curious, he made his way over to see what was going on.

Fix **trigger tokens**: $q_1, \ldots, q_K$
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Output tokens**: $o_k \sim \pi_o(\cdot|q_k)$ (*random*)
- **Sequence-specific Markov model**: $z_1 \sim \pi_1$, $z_t|z_{t-1} \sim p(\cdot|z_{t-1})$ with

$$p(j|i) = \begin{cases} \mathbb{1}\{j = o_k\}, & \text{if } i = q_k, \quad k = 1, \ldots, K \\ \pi_b(j|i), & \text{o/w.} \end{cases}$$

π_b: **global bigrams** model (estimated from Karpathy's character-level Shakespeare)

Transformers on the bigram task

When Mr White went … then Mr ___

trigger output

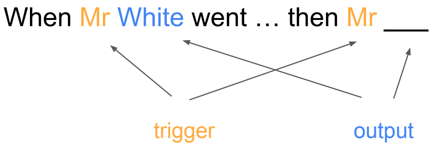
Transformers on the bigram task

When Mr White went … then Mr ___

trigger output

- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions

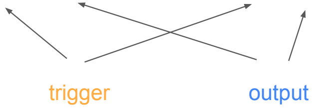
Transformers on the bigram task



- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions
- **2-layer transformer succeeds**: $\sim 99\%$ accuracy

Transformers on the bigram task

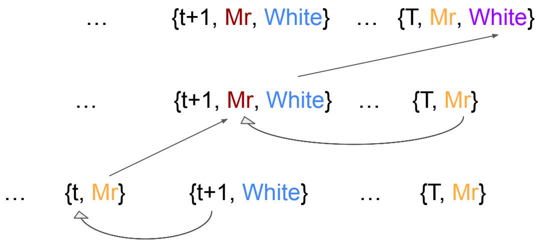
When Mr White went … then Mr ___



- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions
- **2-layer transformer succeeds**: $\sim 99\%$ accuracy

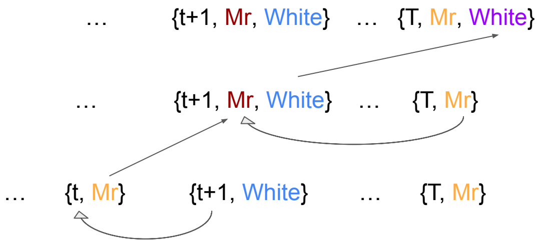
See (Sanford, Hsu, and Telgarsky, 2023, 2024) for representational lower bounds

(1-hop) Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)



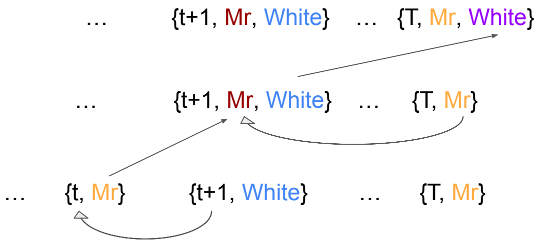
- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream

(1-hop) Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)

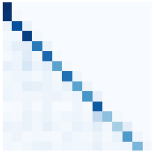


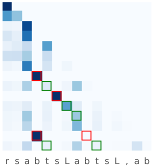
- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream
- 2nd layer: **induction head**
  - attends to output of previous token head, copies attended token

(1-hop) Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)



- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream
- 2nd layer: **induction head**
  - attends to output of previous token head, copies attended token
- Matches observed attention scores:





Random embeddings in high dimension

- We consider **random** embeddings u_i with i.i.d. $\mathcal{N}(0, 1/d)$ entries and d large

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j = O(1/\sqrt{d})$$

Random embeddings in high dimension

- We consider **random** embeddings u_i with i.i.d. $\mathcal{N}(0, 1/d)$ entries and d large

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j = O(1/\sqrt{d})$$

- **Remapping**: multiply by random matrix W with $\mathcal{N}(0, 1/d)$ entries:

$$\|W u_i\| \approx 1 \quad \text{and} \quad u_i^\top W u_i = O(1/\sqrt{d})$$

Random embeddings in high dimension

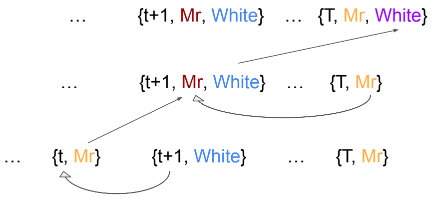
- We consider **random** embeddings u_i with i.i.d. $\mathcal{N}(0, 1/d)$ entries and d large

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j = O(1/\sqrt{d})$$

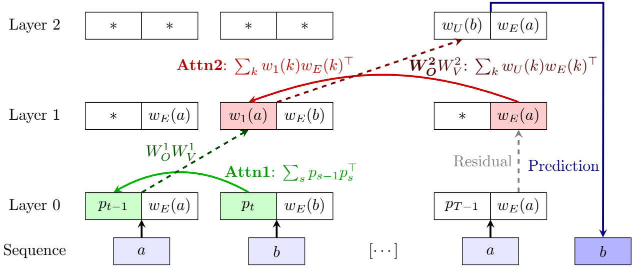
- **Remapping**: multiply by random matrix W with $\mathcal{N}(0, 1/d)$ entries:

$$\|W u_i\| \approx 1 \quad \text{and} \quad u_i^\top W u_i = O(1/\sqrt{d})$$

- Value/Output matrices help with token **remapping**: Mr $\mapsto$ Mr, White $\mapsto$ White



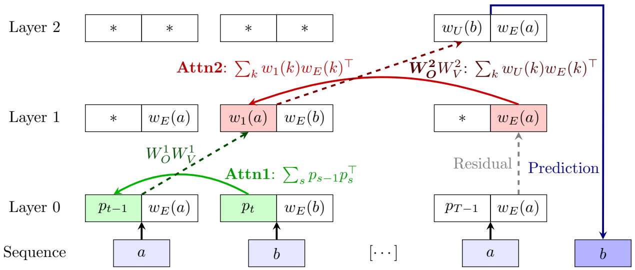
Induction head with associative memories



$$W_{KQ}^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_{KQ}^2 = \sum_{k \in Q} e_k \tilde{e}_k^\top, \quad W_{OV}^2 = \sum_{k=1}^{N} u_k e_k^\top,$$

- Random embeddings e_k, u_k, random matrix W_{OV}^1 (frozen at init)
- **Remapped** previous tokens: $\tilde{e}_k := W_{OV}^1 e_k$

Induction head with associative memories



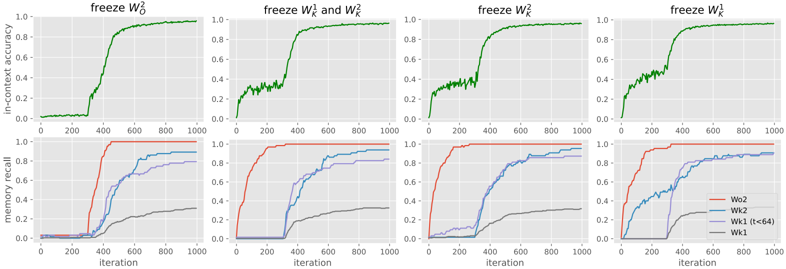
$$W_{KQ}^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_{KQ}^2 = \sum_{k \in Q} e_k \tilde{e}_k^\top, \quad W_{OV}^2 = \sum_{k=1}^{N} u_k e_k^\top,$$

- Random embeddings e_k, u_k, random matrix W_{OV}^1 (frozen at init)
- **Remapped** previous tokens: $\tilde{e}_k := W_{OV}^1 e_k$

Q: Does this match practice?

Empirically probing the dynamics

Train only W_{KQ}^1, W_{KQ}^2, W_{OV}^2, loss on deterministic output tokens only

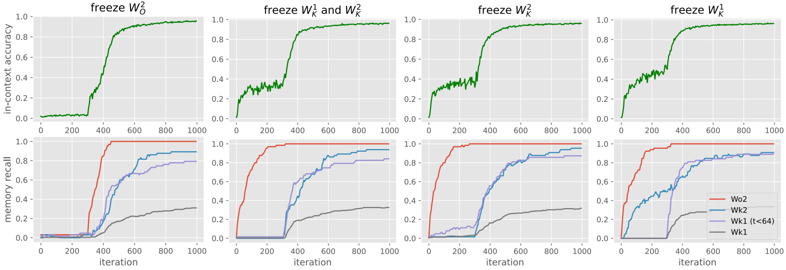


- "Memory recall **probes**": for target memory $W_* = \sum_{i=1}^{M} u_i e_i^\top$, compute

$$R(\hat{W}, W_*) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}\{i = \arg\max_j u_j^\top \hat{W} e_i\}$$

Empirically probing the dynamics

Train only W_{KQ}^1, W_{KQ}^2, W_{OV}^2, loss on deterministic output tokens only



- "Memory recall **probes**": for target memory $W_* = \sum_{i=1}^M u_i e_i^\top$, compute

$$R(\hat{W}, W_*) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}\{i = \arg\max_j u_j^\top \hat{W} e_i\}$$

- Natural learning "**order**": W_{OV}^2 first, W_{KQ}^2 next, W_{KQ}^1 last
- Joint learning is faster

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Theorem (B. et al., 2023, informal)

In the setup above, we can recover the desired associative memories with ***3 gradient steps*** *on the population loss: first on* W_{OV}^2*, then* W_{KQ}^2*, then* W_{KQ}^1.

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Theorem (B. et al., 2023, informal)

In the setup above, we can recover the desired associative memories with ***3 gradient steps*** *on the population loss: first on W_{OV}^2, then W_{KQ}^2, then W_{KQ}^1.*

Key ideas

- Attention is uniform at initialization $\implies$ inputs are sums of embeddings
- W_{OV}^2: correct output appears w.p. 1, while other tokens are noisy and cond. indep. of z_T
- $W_{KQ}^{1/2}$: correct associations lead to more focused attention

Gradient steps for the bigram task

Setting: transformer on the bigram task

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture
- Infinite width, infinite data, $N \gg T$

Theorem (B. et al., 2023, informal)

*In the setup above, we can recover the desired associative memories with **3 gradient steps** on the population loss: first on W_{OV}^2, then W_{KQ}^2, then W_{KQ}^1.*

Key ideas

- Attention is uniform at initialization $\implies$ inputs are sums of embeddings
- W_{OV}^2: correct output appears w.p. 1, while other tokens are noisy and cond. indep. of z_T
- $W_{KQ}^{1/2}$: correct associations lead to more focused attention
- See also Eshaan's talk for k-hop with finite samples

Key idea: gradient associative memories with noisy inputs

Insight: residual streams, attention output at init, are noisy sums of embeddings

Lemma (Gradients with noisy inputs, B. et al., 2023)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, and consider the loss

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, F_W(x))], \quad F_W(z)_k = u_k{}^\top W x.$$

Key idea: gradient associative memories with noisy inputs

Insight: residual streams, attention output at init, are noisy sums of embeddings

Lemma (Gradients with noisy inputs, B. et al., 2023)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, *and consider the loss*

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, F_W(x))], \quad F_W(z)_k = {u_k}^\top W x.$$

Denoting $\mu_k := \mathbb{E}[x|y = k]$ *and* $\hat{\mu}_k := \mathbb{E}_x[\frac{\hat{p}_W(k|x)}{p(y=k)} x]$, *we have*

$$\nabla_W L(W) = \sum_{k=1}^{N} p(y = k) u_k (\hat{\mu}_k - \mu_k)^\top.$$

Key idea: gradient associative memories with noisy inputs

Insight: residual streams, attention output at init, are noisy sums of embeddings

Lemma (Gradients with noisy inputs, B. et al., 2023)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, and consider the loss

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, F_W(x))], \quad F_W(z)_k = u_k{}^\top W x.$$

Denoting $\mu_k := \mathbb{E}[x|y = k]$ and $\hat{\mu}_k := \mathbb{E}_x[\frac{\hat{p}_W(k|x)}{p(y=k)} x]$, we have

$$\nabla_W L(W) = \sum_{k=1}^{N} p(y = k) u_k (\hat{\mu}_k - \mu_k)^\top.$$

- **Example**: $y \sim \mathrm{Unif}([N])$, $t \sim \mathrm{Unif}([T])$, $x = e_y + p_t$.

Key idea: gradient associative memories with noisy inputs

Insight: residual streams, attention output at init, are noisy sums of embeddings

Lemma (Gradients with noisy inputs, B. et al., 2023)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, and consider the loss

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, F_W(x))], \quad F_W(z)_k = {u_k}^\top W x.$$

Denoting $\mu_k := \mathbb{E}[x|y = k]$ *and* $\hat{\mu}_k := \mathbb{E}_x[\frac{\hat{p}_W(k|x)}{p(y=k)} x]$, *we have*

$$\nabla_W L(W) = \sum_{k=1}^{N} p(y = k) u_k (\hat{\mu}_k - \mu_k)^\top.$$

- **Example**: $y \sim \mathrm{Unif}([N])$, $t \sim \mathrm{Unif}([T])$, $x = e_y + p_t$. One gradient step:

$$u_k^\top W_1(e_y + p_t) \approx \frac{\eta}{N} \mathbb{1}\{y = k\} + O\left(\frac{1}{N^2}\right)$$

Key idea: gradient associative memories with noisy inputs

Insight: residual streams, attention output at init, are noisy sums of embeddings

Lemma (Gradients with noisy inputs, B. et al., 2023)

Let p be a data distribution over $(x, y) \in \mathbb{R}^d \times [N]$, and consider the loss

$$L(W) = \mathbb{E}_{(x,y)\sim p}[\ell(y, F_W(x))], \quad F_W(z)_k = {u_k}^\top W x.$$

Denoting $\mu_k := \mathbb{E}[x|y=k]$ and $\hat{\mu}_k := \mathbb{E}_x[\frac{\hat{p}_W(k|x)}{p(y=k)} x]$, we have

$$\nabla_W L(W) = \sum_{k=1}^{N} p(y=k) u_k (\hat{\mu}_k - \mu_k)^\top.$$

- **Example**: $y \sim \mathrm{Unif}([N])$, $t \sim \mathrm{Unif}([T])$, $x = e_y + p_t$. One gradient step:

$$u_k^\top W_1(e_y + p_t) \approx \frac{\eta}{N} \mathbb{1}\{y = k\} + O\left(\frac{1}{N^2}\right)$$

- Similar arguments for attention matrices

Finite width and finite samples (Vural et al., 2025+)

- Data model: $y = f^*(z_{t^*})$, with **one planted relevant token** z_{t^*} in a context of length T

Finite width and finite samples (Vural et al., 2025+)

- Data model: $y = f^*(z_{t^*})$, with **one planted relevant token** z_{t^*} in a context of length T
- Assume first layer is already learned (trigger appears along with relevant token)

Finite width and finite samples (Vural et al., 2025+)

- Data model: $y = f^*(z_{t^*})$, with **one planted relevant token** z_{t^*} in a context of length T
- Assume first layer is already learned (trigger appears along with relevant token)
- Three steps: first W_{OV}, then W_{KQ}, then W_{OV} again (see also Oymak et al., 2023)

Finite width and finite samples (Vural et al., 2025+)

- Data model: $y = f^*(z_{t^*})$, with **one planted relevant token** z_{t^*} in a context of length T
- Assume first layer is already learned (trigger appears along with relevant token)
- Three steps: first W_{OV}, then W_{KQ}, then W_{OV} again (see also Oymak et al., 2023)
- d: embed dimension, N: vocab size, T: context length, n: sample size

Finite width and finite samples (Vural et al., 2025+)

- Data model: $y = f^*(z_{t^*})$, with **one planted relevant token** z_{t^*} in a context of length T
- Assume first layer is already learned (trigger appears along with relevant token)
- Three steps: first W_{OV}, then W_{KQ}, then W_{OV} again (see also Oymak et al., 2023)
- d: embed dimension, N: vocab size, T: context length, n: sample size

Theorem (Learnability of induction head, Vural, Wu, and B., 2025+, informal)

Assume $d \gg \sqrt{N} \gg 1$ and $n \gg N$. The Transformer learns the desired mapping iff

$$d \gg \frac{T^2}{N} \min\left\{1, \left(\frac{N^2}{Tn}\right)^{2/3}\right\}$$

Finite width and finite samples (Vural et al., 2025+)

- Data model: $y = f^*(z_{t^*})$, with **one planted relevant token** z_{t^*} in a context of length T
- Assume first layer is already learned (trigger appears along with relevant token)
- Three steps: first W_{OV}, then W_{KQ}, then W_{OV} again (see also Oymak et al., 2023)
- d: embed dimension, N: vocab size, T: context length, n: sample size

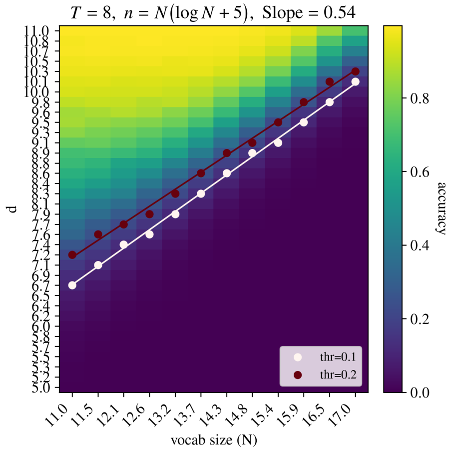
Theorem (Learnability of induction head, Vural, Wu, and B., 2025+, informal)

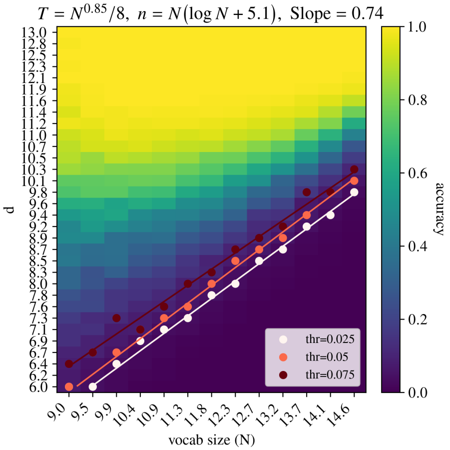
Assume $d \gg \sqrt{N} \gg 1$ and $n \gg N$. The Transformer learns the desired mapping iff

$$d \gg \frac{T^2}{N} \min\left\{1, \left(\frac{N^2}{Tn}\right)^{2/3}\right\}$$

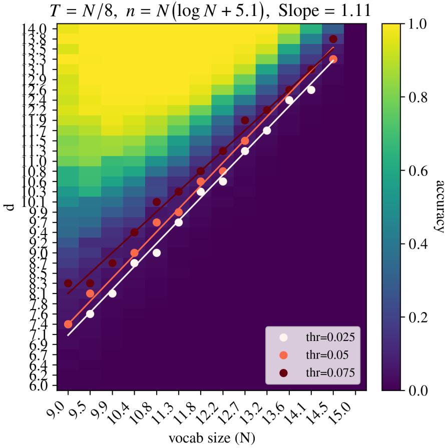
- If $T = \Theta(1)$, $d \gg \sqrt{N}$ is enough
- If $T \gg 1$, need $d \gg \sqrt{N}$ is **not** enough unless n is very large

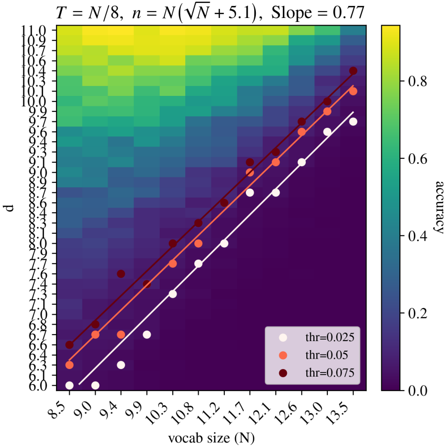
Finite width and finite samples (Vural et al., 2025+)





Finite width and finite samples (Vural et al., 2025+)





Concluding remarks

Transformer weights as associative memories

- Storage capacity and gradient-based learning
- Toy models of reasoning and factual recall

Concluding remarks

Transformer weights as associative memories

- Storage capacity and gradient-based learning
- Toy models of reasoning and factual recall

Future directions

- Analysis for more general tasks
- Fine-grained optimization
- Learning embeddings

Concluding remarks

Transformer weights as associative memories

- Storage capacity and gradient-based learning
- Toy models of reasoning and factual recall

Future directions

- Analysis for more general tasks
- Fine-grained optimization
- Learning embeddings

Thank you!

References I

Z. Allen-Zhu and Y. Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

D. J. Amit, H. Gutfreund, and H. Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530, 1985.

A. B., V. Cabannes, D. Bouchacourt, H. Jegou, and L. Bottou. Birth of a transformer: A memory viewpoint. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

J. Ba, M. A. Erdogdu, T. Suzuki, Z. Wang, D. Wu, and G. Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

V. Cabannes, E. Dohmatob, and A. B. Scaling laws for associative memories. In *International Conference on Learning Representations (ICLR)*, 2024a.

V. Cabannes, B. Simsek, and A. B. Learning associative memories with gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024b.

L. Chen, J. Bruna, and A. B. How truncating weights improves reasoning in language models. *arXiv preprint arXiv:2406.03068*, 2024.

A. Damian, J. Lee, and M. Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory (COLT)*, 2022.

References II

Y. Dandi, F. Krzakala, B. Loureiro, L. Pesce, and L. Stephan. Learning two-layer neural networks, one (giant) step at a time. *arXiv preprint arXiv:2305.18270*, 2023.

M. Demircigil, J. Heusel, M. Löwe, S. Upgang, and F. Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168:288–299, 2017.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

M. Geva, R. Schuster, J. Berant, and O. Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

M. Hutter. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

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

References III

C. Lucibello and M. Mézard. Exponential capacity of dense associative memories. *Physical Review Letters*, 132(7):077301, 2024.

R. McEliece, E. Posner, E. Rodemich, and S. Venkatesh. The capacity of the hopfield associative memory. *IEEE transactions on Information Theory*, 33(4):461–482, 1988.

W. Merrill, A. Sabharwal, and N. A. Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

E. Nichani, J. D. Lee, and A. B. Understanding factual recall in transformers via associative memories. *arXiv preprint arXiv:2412.06538*, 2024.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

S. Oymak, A. S. Rawat, M. Soltanolkotabi, and C. Thrampoulidis. On the role of attention in prompt-tuning. In *International Conference on Machine Learning*, 2023.

H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

References IV

C. Sanford, D. Hsu, and M. Telgarsky. Representational strengths and limitations of transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

C. Sanford, D. Hsu, and M. Telgarsky. One-layer transformers fail to solve the induction heads task. *arXiv preprint arXiv:2408.14332*, 2024.

P. Sharma, J. T. Ash, and D. Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*, 2023.

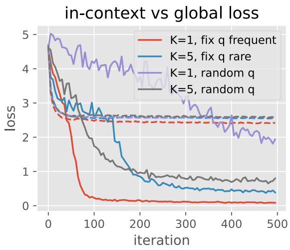
M. Smart, A. B., and A. M. Sengupta. In-context denoising with one-layer transformers: connections between attention and associative memory retrieval. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

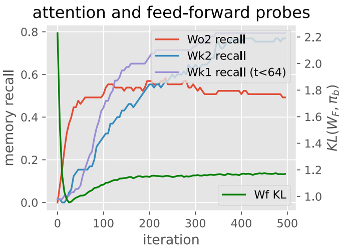
M. Vural, D. Wu, and A. B. Sample complexity of learning attention. *in prep*, 2025+.

K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

G. Yang and E. J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

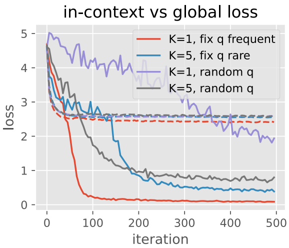
Global vs in-context associations

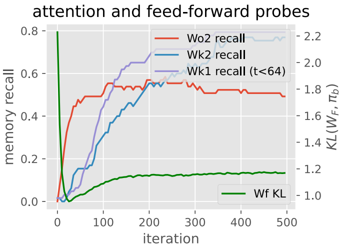


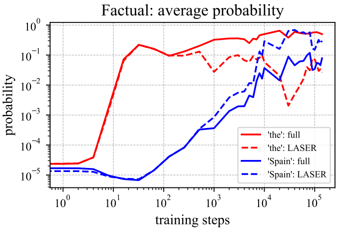


- Global bigrams are learned much faster than induction head, tend to be stored in MLPs

Global vs in-context associations





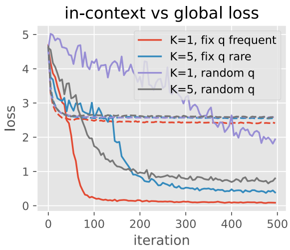


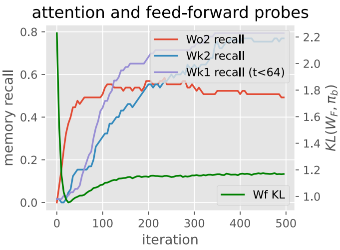
- Global bigrams are learned much faster than induction head, tend to be stored in MLPs

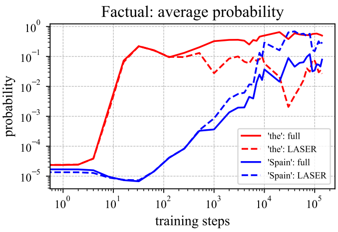
Trade-offs between global and in-context predictions (Chen, Bruna, and B., 2024)

- Trade-offs also appear in LLMs
  - "Madrid is located in" $\rightarrow$ {the, Spain} on Pythia-1B
  - Ablating late-layer MLPs (Sharma et al., 2023) changes prediction from global to in-context

Global vs in-context associations







- Global bigrams are learned much faster than induction head, tend to be stored in MLPs

Trade-offs between global and in-context predictions (Chen, Bruna, and B., 2024)

- Trade-offs also appear in LLMs
  - "Madrid is located in" $\rightarrow$ {the, Spain} on Pythia-1B
  - Ablating late-layer MLPs (Sharma et al., 2023) changes prediction from global to in-context

Theorem (Chen et al., 2024, informal)

In toy setting, feed-forward layer learns global bigram after $O(1)$ samples, attention after $O(N)$ samples due to noise.

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \ldots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \ldots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: Zipf law (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: Zipf law (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

- Hutter (2021): with infinite memory, we have

$$L(\hat{f}_n) \lesssim n^{-\frac{\alpha-1}{\alpha}}$$

Setup with heavy-tailed data

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, n samples: $S_n = \{z_1, \dots, z_n\}$, 0/1 loss:

$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$

- Heavy-tailed token frequencies: Zipf law (typical for language where N is very large)

$$p(z) \propto z^{-\alpha}$$

- Hutter (2021): with infinite memory, we have

$$L(\hat{f}_n) \lesssim n^{-\frac{\alpha-1}{\alpha}}$$

- **Q: What about finite capacity?**

Scaling laws with finite capacity

- Random embeddings $e_z, u_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y u_y^\top W_{n,d} e_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z^\top$$

Scaling laws with finite capacity

- Random embeddings $e_z, u_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y {u_y}^\top W_{n,d} e_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) u_{f^*(z)} {e_z}^\top$$

- Single population gradient step on cross-entropy loss: $q(z) \approx p(z)$

Scaling laws with finite capacity

- Random embeddings $e_z, u_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y u_y{}^\top W_{n,d} e_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z{}^\top$$

- Single population gradient step on cross-entropy loss: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, and B., 2024a, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$

Scaling laws with finite capacity

- Random embeddings $e_z, u_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y u_y{}^\top W_{n,d} e_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z{}^\top$$

- Single population gradient step on cross-entropy loss: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, and B., 2024a, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$, *and* $d \gg N$: $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k

Scaling laws with finite capacity

- Random embeddings $e_z, u_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y {u_y}^\top W_{n,d} e_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) u_{f^*(z)} {e_z}^\top$$

- Single population gradient step on cross-entropy loss: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, and B., 2024a, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$*, and* $d \gg N$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k
3. *For* $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

Scaling laws with finite capacity

- Random embeddings $e_z, u_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y {u_y}^\top W_{n,d} e_z$, with

$$W_{n,d} = \sum_{z=1}^{N} q(z) u_{f^*(z)} {e_z}^\top$$

- Single population gradient step on cross-entropy loss: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, and B., 2024a, informal)

1. *For* $q(z) = \sum_i \mathbb{1}\{z = z_i\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\frac{\alpha-1}{2\alpha}}$
2. *For* $q(z) = \mathbb{1}\{z \in S_n\}$*, and* $d \gg N$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-k}$ *for any* k
3. *For* $q(z) = \mathbb{1}\{z \text{ seen at least } s \text{ times in } S_n\}$*:* $L(\hat{f}_{n,d}) \lesssim n^{-\frac{\alpha-1}{\alpha}} + d^{-\alpha+1}$

- $n^{-\frac{\alpha-1}{\alpha}}$ is the same as (Hutter, 2021)
- $q = 1$ is best if we have enough capacity
- Can store at most d memories (approximation error: $d^{-\alpha+1}$)

Scaling laws with optimization algorithms

$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), UWe_z)] \qquad \rightarrow \qquad W_{n,d} \approx \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

Scaling laws with optimization algorithms

$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), UWe_z)] \qquad \to \qquad W_{n,d} \approx \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$

Scaling laws with optimization algorithms

$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), UWe_z)] \qquad \to \qquad W_{n,d} \approx \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$

Scaling laws with optimization algorithms

$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), UWe_z)] \qquad \to \qquad W_{n,d} \approx \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training

Scaling laws with optimization algorithms

$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), UWe_z)] \qquad \to \qquad W_{n,d} \approx \sum_{z=1}^{N} q(z) u_{f^*(z)} {e_z}^\top$$

Different algorithms lead to different memory schemes $q(z)$:

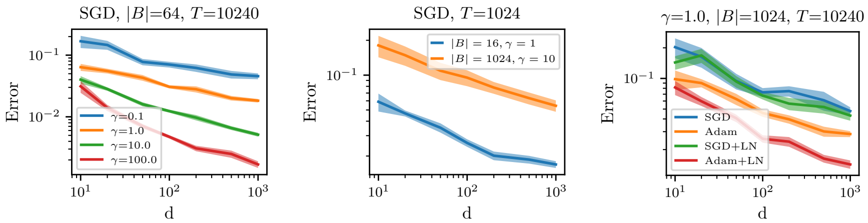
- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training
- Adam and layer-norm help with practical settings (large batch sizes + smaller step-size)

Scaling laws with optimization algorithms

$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), UWe_z)] \quad \rightarrow \quad W_{n,d} \approx \sum_{z=1}^{N} q(z) u_{f^*(z)} e_z^\top$$

Different algorithms lead to different memory schemes $q(z)$:

- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training
- Adam and layer-norm help with practical settings (large batch sizes + smaller step-size)



Optimization with imbalance and small capacity

$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), UWe_z)], \qquad \ell\text{: cross-entropy loss}$$

Benefits of large step-sizes + oscillations: (Cabannes, Simsek, and B., 2024b)

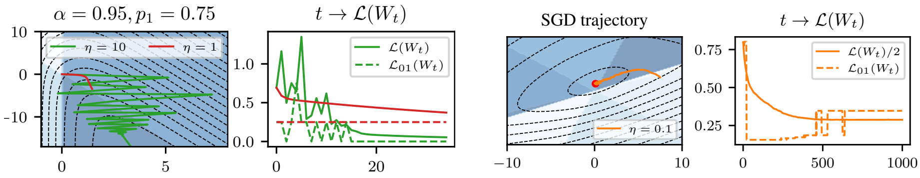
- Orthogonal embeddings $\implies$ logarithmic growth of margins for any step-size

Optimization with imbalance and small capacity

$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), UWe_z)], \qquad \ell\text{: cross-entropy loss}$$

Benefits of large step-sizes + oscillations: (Cabannes, Simsek, and B., 2024b)

- Orthogonal embeddings $\implies$ logarithmic growth of margins for any step-size
- Correlated embeddings + imbalance $\implies$ oscillatory regimes

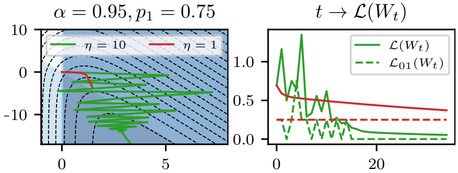


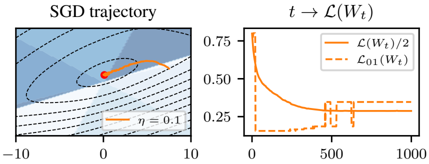
Optimization with imbalance and small capacity

$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), UWe_z)], \qquad \ell\text{: cross-entropy loss}$$

Benefits of large step-sizes + oscillations: (Cabannes, Simsek, and B., 2024b)

- Orthogonal embeddings $\implies$ logarithmic growth of margins for any step-size
- Correlated embeddings + imbalance $\implies$ oscillatory regimes
- Large step-sizes help reach perfect accuracy faster despite oscillations (empirically)





Optimization with imbalance and small capacity

$$L(W) = \mathbb{E}_{z\sim p}[\ell(f^*(z), UWe_z)], \qquad \ell\text{: cross-entropy loss}$$

Benefits of large step-sizes + oscillations: (Cabannes, Simsek, and B., 2024b)

- Orthogonal embeddings $\implies$ logarithmic growth of margins for any step-size
- Correlated embeddings + imbalance $\implies$ oscillatory regimes
- Large step-sizes help reach perfect accuracy faster despite oscillations (empirically)
- Over-optimization can hurt in under-parameterized settings (empirically)