Associative Memories as a Tractable Building Block in Transformers

Alberto Bietti

Flatiron Institute, Simons Foundation

FLAIR Seminar, EPFL, December 2024.

SIMONS FOUNDATION FLATIRON INSTITUTE

Associative Memories as a Tractable Building Block in Transformers

Alberto Bietti

Flatiron Institute, Simons Foundation

FLAIR Seminar, EPFL, December 2024.

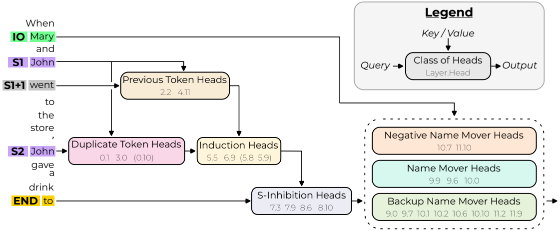
w/ V. Cabannes, E. Dohmatob, D. Bouchacourt, H. Jégou, L. Bottou (Meta AI),
E. Nichani, J. Lee (Princeton), B. Simsek, L. Chen, J. Bruna (NYU)

SIMONS FOUNDATION FLATIRON INSTITUTE

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/logic/MPC)



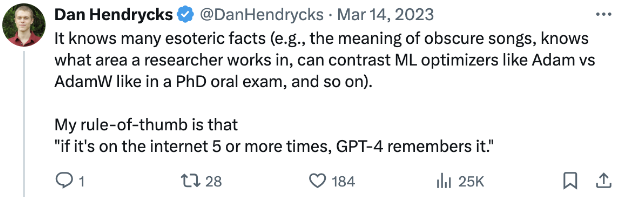
What are Transformer LLMs doing?

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/logic/MPC)

Knowledge storage

- Factual recall, memorization, scaling parameters
  - (Geva et al., 2020; Meng et al., 2022; Allen-Zhu and Li, 2024)
- Allows higher-level reasoning



Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the Government for a redress of grievances.

This is known as the First Amendment to the United States Constitution and it is designed to protect the fundamental rights of citizens of the United States. It guarantees citizens the right to practice any religion of their choosing, the freedom of speech and of the press, and the right to peacefully assemble and to petition the government.

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/logic/MPC)

Knowledge storage

- Factual recall, memorization, scaling parameters
  - (Geva et al., 2020; Meng et al., 2022; Allen-Zhu and Li, 2024)
- Allows higher-level reasoning

Q: What is a good model for these + training dynamics?

Transformer setup

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 setup

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 setup

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 setup

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 $x_t := x_t + \mathrm{MLP}(x_t)$

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

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

Transformer setup

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 $x_t := x_t + \text{MHSA}(x_t, x_{1:t})$
- feed-forward $x_t := x_t + \text{MLP}(x_t)$
- residual stream x_t is a sum of many embeddings/"features"

Transformer setup

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 many embeddings/"features"

Next-token prediction

- cross-entropy loss

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

Outline

Weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

Weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

Weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad v_i^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j u_i^\top \quad \Longrightarrow \quad v_j^\top W u_i \approx \alpha_{ij}$$

Weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top \quad \implies \quad {v_j}^\top W u_i \approx \alpha_{ij}$$

- Examples in Transformers:
  - Logits in attention heads: $x_k^\top W_{KQ} x_q$
  - Logits in next-token prediction: $v_y^\top U \sigma(V x_t)$ or $v_y^\top W_{OV} x_k$

Weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top \quad \implies \quad {v_j}^\top W u_i \approx \alpha_{ij}$$

- Examples in Transformers:
  - Logits in attention heads: $x_k^\top W_{KQ} x_q$
  - Logits in next-token prediction: $v_y^\top U \sigma(V x_t)$ or $v_y^\top W_{OV} x_k$
- Related to Hopfield (1982); Kohonen (1972); Willshaw et al. (1969); Iscen et al. (2017)

Weights as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top \quad \Longrightarrow \quad {v_j}^\top W u_i \approx \alpha_{ij}$$

- Examples in Transformers:
  - Logits in attention heads: $x_k^\top W_{KQ} x_q$
  - Logits in next-token prediction: $v_y^\top U \sigma(V x_t)$ or $v_y^\top W_{OV} x_k$
- Related to Hopfield (1982); Kohonen (1972); Willshaw et al. (1969); Iscen et al. (2017)
- Note: attention itself is also related to AM (Ramsauer et al., 2020; Schlag et al., 2021)

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

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

with ℓ the cross-entropy loss and u_z, v_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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

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

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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

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

- **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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

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

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

$$ {v_k}^\top W_1 u_z \approx \frac{\eta}{N}\left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N}\right)$$

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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

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

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

$$ {v_k}^\top W_1 u_z \approx \frac{\eta}{N}\left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N}\right)$$

  - **Corollary**: $\hat{f}(z) = \arg\max_k {v_k}^\top W_1 u_z$ 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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

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

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

$$ {v_k}^\top W_1 u_z \approx \frac{\eta}{N} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right)$$

  - **Corollary**: $\hat{f}(z) = \arg\max_k {v_k}^\top W_1 u_z$ has near-perfect accuracy
- More generally, replace v_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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

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

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

$$ {v_k}^\top W_1 u_z \approx \frac{\eta}{N} \left(\mathbb{1}\{f_*(z) = k\} - \frac{1}{N} \right)$$

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

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

Capacity: Intuition

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

Capacity: Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$

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

Capacity: Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$

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

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

$$\gamma_{z,y} := v_y{}^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

Capacity: Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$

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

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

$$\gamma_{z,y} := v_y{}^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

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

Capacity: Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$

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

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

$$\gamma_{z,y} := v_y{}^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

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

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

Capacity: Intuition

- Random embeddings $u_z, v_y \sim \mathcal{N}(0, \frac{1}{d} I)$
- For some $f^* : [N] \to [M]$

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

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

$$\gamma_{z,y} := v_y^\top W u_z = \sum_{z'} v_y^\top v_{f^*(z')} u_z^\top u_{z'}$$

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

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

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

Capacity ≈ 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

- $\hat{f}(z) = \arg\max_y {v_y}^\top W_1 \sigma(W_2^\top u_z)$, $W_1, W_2 \in \mathbb{R}^{d \times m}$
- 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 Bietti, 2024), related to Krotov and Hopfield (2016); Demircigil et al. (2017)

Capacity ≈ 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

- $\hat{f}(z) = \arg\max_y v_y{}^\top W_1 \sigma(W_2^\top u_z)$, $W_1, W_2 \in \mathbb{R}^{d \times m}$
- 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

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

(Nichani, Lee, and Bietti, 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

- $\hat{f}(z) = \arg\max_y {v_y}^\top W_1 \sigma(W_2^\top u_z)$, $W_1, W_2 \in \mathbb{R}^{d \times m}$
- 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

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

Note: matches information-theoretic lower bounds

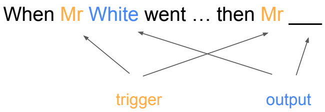
(Nichani, Lee, and Bietti, 2024), related to Krotov and Hopfield (2016); Demircigil et al. (2017)

Outline

1. Associative memories
2. Application to Transformers I: induction heads (B. et al., 2023)
3. Application to Transformers II: factual recall (Nichani et al., 2024)
4. Scaling laws and optimization (Cabannes et al., 2024a,b)

The bigram data model for in-context reasoning

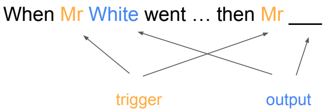
Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



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.

The bigram data model for in-context reasoning

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

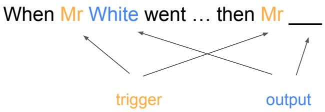


When Mr White went to the mall, it started raining, then Mr White witnessed an odd occurrence. While walking around the mall with his family, Mr White heard the sound of a helicopter landing in the parking lot. Curious, he made his way over to see what was going on.

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.

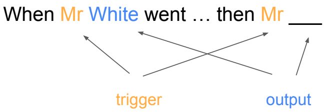
Fix **trigger tokens**: $q_1, \dots, q_K$

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

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

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**: $q_1, \dots, q_K$

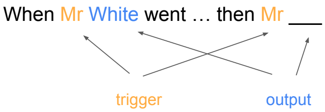
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **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**: $q_1, \ldots, q_K$

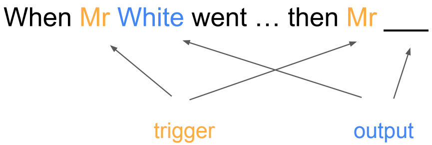
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **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

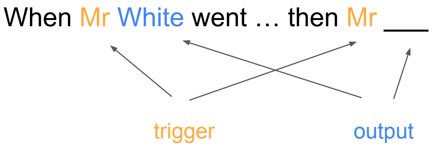
$$p(j|i) = \begin{cases} \mathbb{1}\{j = o_k\}, & \text{if } i = q_k, \quad k = 1, \ldots, K \\ \pi_b(j|i), & \text{o/w.} \end{cases}$$

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

Transformers on the bigram task

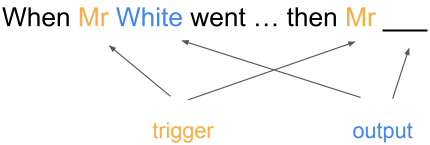


Transformers on the bigram task



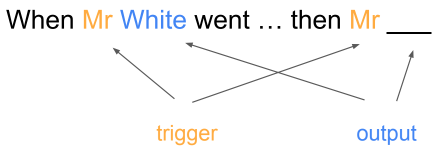
- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions

Transformers on the bigram task



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

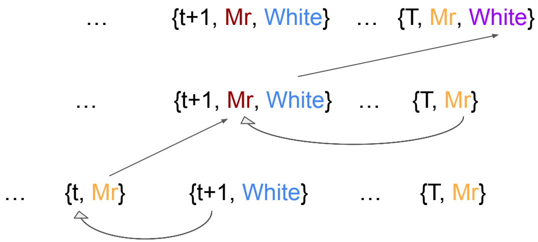
Transformers on the bigram task



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

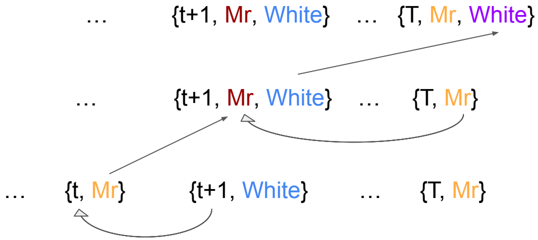
See (Sanford, Hsu, and Telgarsky, 2023, 2024) for representational lower bounds

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



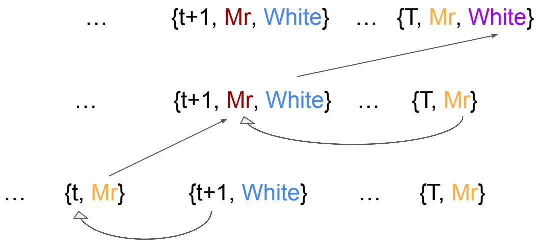
- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream

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

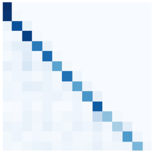


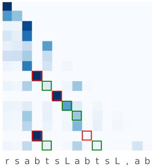
- 1st layer: **previous-token head**
  - attends to previous token and copies it to residual stream
- 2nd layer: **induction head**
  - attends to output of previous token head, copies attended token

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

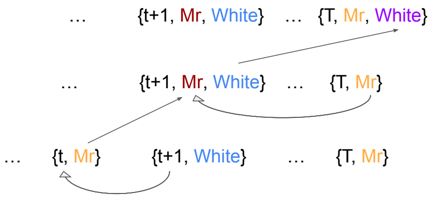
- We consider **random** embeddings u_i with i.i.d. $\mathcal{N}(0, 1/d)$ entries and d large

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

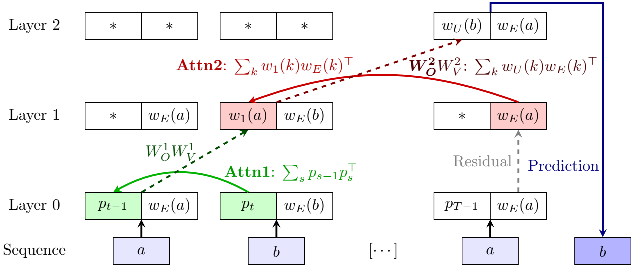
- **Remapping**: multiply by random matrix W with $\mathcal{N}(0, 1/d)$ entries:

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

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



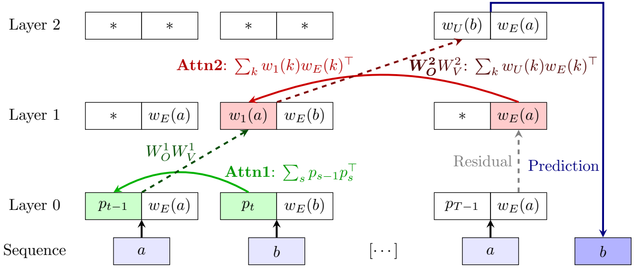
Induction head with associative memories



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

- 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



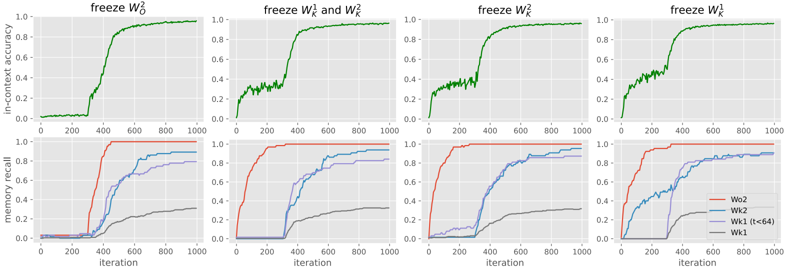
$$W_{KQ}^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_{KQ}^2 = \sum_{k \in Q} e_k \tilde{e}_k^\top, \quad W_{OV}^2 = \sum_{k=1}^{N} u_k e_k^\top,$$

- 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

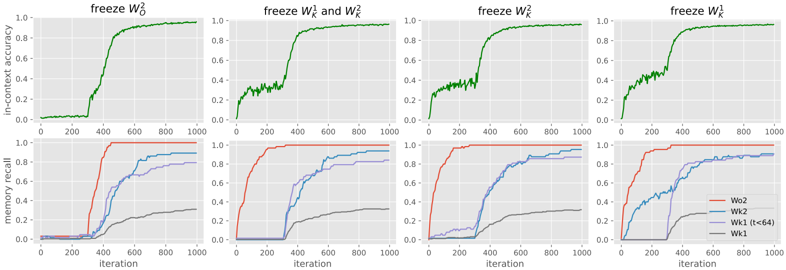


- "Memory recall **probes**": for target memory $W_* = \sum_{i=1}^M v_i u_i^\top$, compute

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

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 $W_* = \sum_{i=1}^M v_i u_i^\top$, compute

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

- 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 (Snell et al., 2021; Oymak et al., 2023)

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)

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

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

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)

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

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

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*

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

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)

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

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

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

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

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

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)

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

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

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

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

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

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

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)

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

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

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*

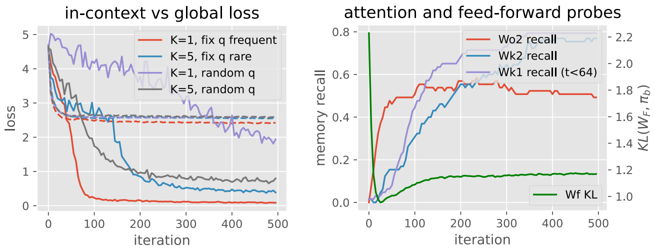
$$\nabla_W L(W) = \sum_{k=1}^{N} p(y = k) v_k (\hat{\mu}_k - \mu_k)^\top.$$

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

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

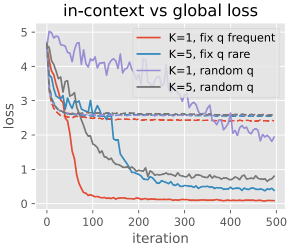
- Similar arguments for attention matrices

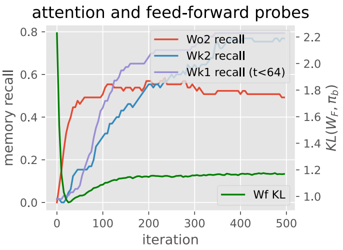
Global vs in-context associations

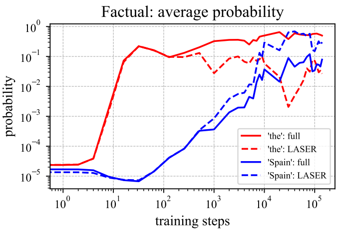


- Global bigrams are learned much faster than induction head, tend to be stored in MLPs

Global vs in-context associations





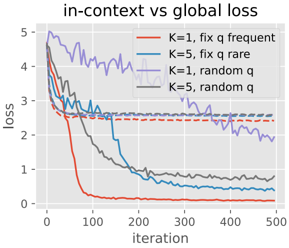


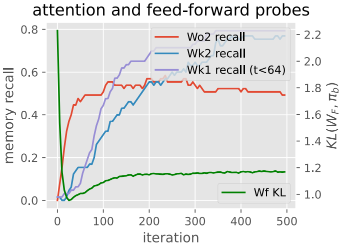
- Global bigrams are learned much faster than induction head, tend to be stored in MLPs

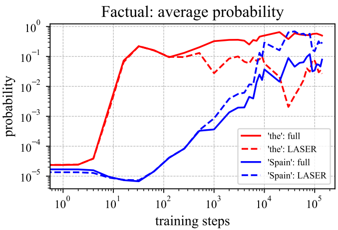
Trade-offs between global and in-context predictions (Chen, Bruna, and B., 2024)

- 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" $\to$ {the, Spain} on Pythia-1B
  - Ablating late-layer MLPs (Sharma et al., 2023) changes prediction from global to in-context

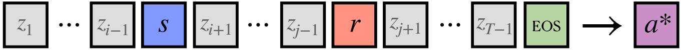
Theorem (Chen et al., 2024, informal)

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

Outline

1. Associative memories
2. Application to Transformers I: induction heads (B. et al., 2023)
3. Application to Transformers II: factual recall (Nichani et al., 2024)
4. Scaling laws and optimization (Cabannes et al., 2024a,b)

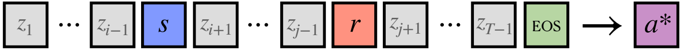
Toy model of factual recall



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

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 ($A_{\max} := \max_r |\mathcal{A}_r|$)*

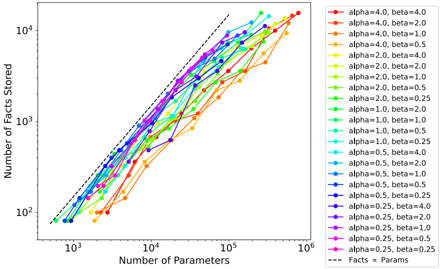
- Total parameters scale with number of facts SR (up to $A_{\max}$)
- Constructions are based on associative memories
- Attention-only needs large enough d
- Noise is negligible (log factors)

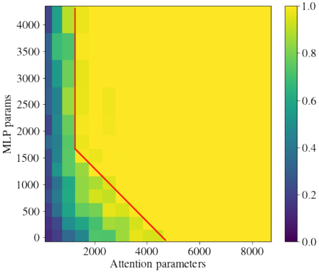
How many parameters do we need?

- 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* ($A_{\max} := \max_r |\mathcal{A}_r|$)





Training dynamics

- One-layer Transformer with **linear attention**, 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 $p(a|s, r)$*

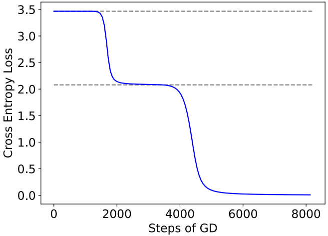
Training dynamics

- One-layer Transformer with **linear attention**, 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

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: $S_n = \{z_1, \ldots, z_n\}$, 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)

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

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

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

Scaling laws with finite capacity

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

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

- Single population gradient step: $q(z) \approx p(z)$

Scaling laws with finite capacity

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

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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 $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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 $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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 $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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

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

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

Scaling laws with optimization algorithms

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

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

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

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

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

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

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

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

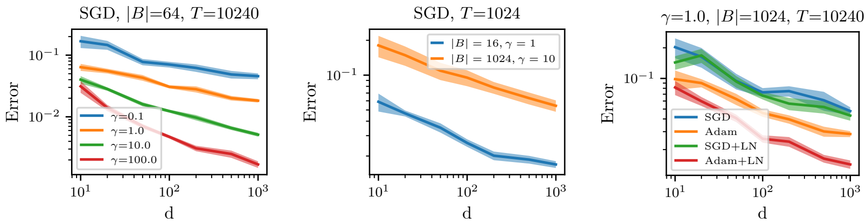
- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training
- Adam and layer-norm help with practical settings (large batch sizes + smaller step-size)

Scaling laws with optimization algorithms

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

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

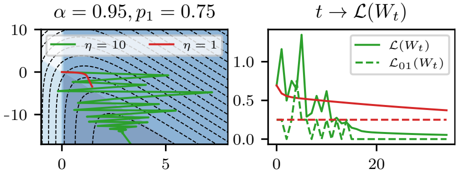
$$L(W) = \mathbb{E}_{z \sim p}[\ell(f^*(z), V W u_z)], \qquad \ell\text{: cross-entropy loss}$$

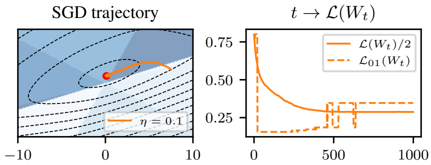
Optimization with imbalance and small capacity

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

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)





Concluding remarks

Transformer weights as associative memories

- Storage capacity and gradient-based learning
- Toy models of reasoning and factual recall
- More optimization can help with capacity

Concluding remarks

Transformer weights as associative memories

- Storage capacity and gradient-based learning
- Toy models of reasoning and factual recall
- More optimization can help with capacity

Future directions

- Finite sample results
- More complex reasoning problems
- 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
- More optimization can help with capacity

Future directions

- Finite sample results
- More complex reasoning problems
- 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.

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.

L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

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.

A. Iscen, T. Furon, V. Gripon, M. Rabbat, and H. Jégou. Memory vectors for similarity search in high-dimensional spaces. *IEEE transactions on big data*, 4(1):65–77, 2017.

T. Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, 1972.

References III

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

S. Mei, T. Misiakiewicz, and A. Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Conference on Learning Theory (COLT)*, 2019.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

E. Nichani, A. Damian, and J. D. Lee. Provable guarantees for nonlinear feature learning in three-layer neural networks. *arXiv preprint arXiv:2305.06986*, 2023.

E. Nichani, J. D. Lee, and A. Bietti. 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.

References IV

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.

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.

I. Schlag, K. Irie, and J. Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

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.

C. Snell, R. Zhong, D. Klein, and J. Steinhardt. Approximating how single head attention learns. *arXiv preprint arXiv:2103.07601*, 2021.

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.

D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222(5197):960–962, 1969.

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.

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: “visual words”, features, objects

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: "visual words", features, objects

Statistical learning setup:

- Data distribution $p(z, y)$ over pairs of **discrete tokens** $(z, y) \in [N] \times [M]$

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: "visual words", features, objects

Statistical learning setup:

- Data distribution $p(z, y)$ over pairs of **discrete tokens** $(z, y) \in [N] \times [M]$
- We want a predictor $\hat{f} : [N] \to [M]$ with **small 0-1 loss**:

$$L_{01}(\hat{f}) = \mathbb{P}(y \neq \hat{f}(z))$$

Learning associations

Motivation:

- DL theory often focuses on learning/approximation of **continuous** target functions
  - *e.g.*, smooth functions, sparse polynomials
- In practice, **discrete structure** and **memorization** are often crucial
  - language: words, syntactic rules, semantic concepts, facts
  - vision: "visual words", features, objects

Statistical learning setup:

- Data distribution $p(z, y)$ over pairs of **discrete tokens** $(z, y) \in [N] \times [M]$
- We want a predictor $\hat{f} : [N] \to [M]$ with **small 0-1 loss**:

$$L_{01}(\hat{f}) = \mathbb{P}(y \neq \hat{f}(z))$$

- Typically $\hat{f}(z) = \arg\max_y f_y(z)$ with $f_y : [N] \to \mathbb{R}$ for each $y \in [M]$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad u_i^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad v_i^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j u_i^\top$$

- We then have $v_j^\top W u_i \approx \alpha_{ij}$
- Computed in Transformers for logits in next-token prediction and self-attention

Matrices as associative memories

- Consider sets of **nearly orthonormal embeddings** $\{u_i\}_{i \in \mathcal{I}}$ and $\{v_j\}_{j \in \mathcal{J}}$:

$$\|u_i\| \approx 1 \quad \text{and} \quad {u_i}^\top u_j \approx 0$$
$$\|v_i\| \approx 1 \quad \text{and} \quad {v_i}^\top v_j \approx 0$$

- Consider **pairwise associations** $(i, j) \in \mathcal{M}$ with **weights** α_{ij} and define:

$$W = \sum_{(i,j) \in \mathcal{M}} \alpha_{ij} v_j {u_i}^\top$$

- We then have ${v_j}^\top W u_i \approx \alpha_{ij}$
- Computed in Transformers for logits in next-token prediction and self-attention

note: closely related to Hopfield (1982); Kohonen (1972); Willshaw et al. (1969)

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to ({v_k}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Lemma (Gradients as memories)

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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings.

Learning associative memories with gradients

- Simple **differentiable model** to learn such associative memories:

$$z \in [N] \to u_z \in \mathbb{R}^d \to W u_z \in \mathbb{R}^d \to (v_k{}^\top W u_z)_k \in \mathbb{R}^M$$

- u_z, v_y: nearly-orthogonal input/output embeddings, assume fixed
- **Cross-entropy loss** for logits $\xi \in \mathbb{R}^M$: $\ell(y, \xi) = -\xi_y + \log(\sum_k \exp \xi_k)$

Lemma (Gradients as memories)

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

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

with ℓ the cross-entropy loss and u_z, v_k input/output embeddings. Then,

$$\nabla L(W) = \sum_{k=1}^{M} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z{}^\top],$$

with $\hat{p}_W(y = k|z) = \exp(\xi_W(z)_k) / \sum_j \exp(\xi_W(z)_j)$.

Example: one gradient step

Data model: $z \sim \text{Unif}([N]), \quad y = f_*(z) \in [N]$

Example: one gradient step

Data model: $z \sim \text{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z{}^\top]$$

$$= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z{}^\top$$

$$= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z{}^\top$$

Example: one gradient step

Data model: $z \sim \mathrm{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$
\begin{aligned}
W_1 &= W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z^\top] \\
&= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z^\top \\
&= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z^\top
\end{aligned}
$$

- Then, for any (z, k) we have

$$
v_k^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)
$$

Example: one gradient step

Data model: $z \sim \mathrm{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z^\top]$$
$$= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z^\top$$
$$= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z^\top$$

- Then, for any (z, k) we have

$$v_k^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)$$

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

Example: one gradient step

Data model: $z \sim \text{Unif}([N]), \quad y = f_*(z) \in [N]$

- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^{N} \mathbb{E}_z[(\hat{p}_W(y = k|z) - p(y = k|z)) v_k u_z^\top]$$
$$= \eta \sum_{z,k} p(z)(p(y = k|z) - \hat{p}_W(y = k|z)) v_k u_z^\top$$
$$= \frac{\eta}{N} \sum_{z,k} (\mathbb{1}\{k = f^*(z)\} - \frac{1}{N}) v_k u_z^\top$$

- Then, for any (z, k) we have

$$v_k^\top W_1 u_z \approx \frac{\eta}{N} \mathbb{1}\{f_*(z) = k\} + O\left(\frac{\eta}{N^2}\right)$$

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

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

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \dots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \ldots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Lemma (Gradients with noisy inputs)

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

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

Gradient associative memories with noisy inputs

- In practice, inputs are often a collection of tokens / sum of embeddings

$$\mathbf{z} = \{z_1, \ldots, z_s\} \subset [N], \quad x = \sum_{j=1}^{s} u_{z_s} \in \mathbb{R}^d$$

  - *e.g.*, bag of words, output of attention operation, residual connections
- Some elements may be irrelevant for prediction

Lemma (Gradients with noisy inputs)

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

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

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

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

Example: filter out exogenous noise

- **Data model**: $y \sim \mathrm{Unif}([N]), \quad t \sim \mathrm{Unif}([T]), \quad x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings

Example: filter out exogenous noise

- **Data model**: $y \sim \text{Unif}([N]), \quad t \sim \text{Unif}([T]), \quad x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings
- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^N p(y=k) v_k (\hat{\mu}_k - \mu_k)^\top$$

$$= \frac{\eta}{N} \sum_{k=1}^N v_k (\mathbb{E}[u_y + n_t | y = k] - \mathbb{E}[u_y + n_t])^\top$$

$$= \frac{\eta}{N} \sum_{k=1}^N v_k u_k^\top - \frac{\eta}{N^2} \sum_{k,j} v_k u_j^\top$$

Example: filter out exogenous noise

- **Data model**: $y \sim \text{Unif}([N])$, $t \sim \text{Unif}([T])$, $x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings
- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$
\begin{aligned}
W_1 &= W_0 - \eta \sum_{k=1}^N p(y=k) v_k (\hat{\mu}_k - \mu_k)^\top \\
&= \frac{\eta}{N} \sum_{k=1}^N v_k (\mathbb{E}[u_y + n_t | y = k] - \mathbb{E}[u_y + n_t])^\top \\
&= \frac{\eta}{N} \sum_{k=1}^N v_k u_k^\top - \frac{\eta}{N^2} \sum_{k,j} v_k u_j^\top
\end{aligned}
$$

- Then, for any $k, y, t, x = u_y + n_t$, we have

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

Example: filter out exogenous noise

- **Data model**: $y \sim \text{Unif}([N]), \quad t \sim \text{Unif}([T]), \quad x = u_y + n_t \in \mathbb{R}^d$
  - where $\{n_t\}_{t=1}^T$ are another collection of embeddings, *e.g.*, positional embeddings
- After **one gradient step** on the population loss from $W_0 = 0$ with step η, we have

$$W_1 = W_0 - \eta \sum_{k=1}^N p(y = k) v_k (\hat{\mu}_k - \mu_k)^\top$$

$$= \frac{\eta}{N} \sum_{k=1}^N v_k (\mathbb{E}[u_y + n_t | y = k] - \mathbb{E}[u_y + n_t])^\top$$

$$= \frac{\eta}{N} \sum_{k=1}^N v_k u_k^\top - \frac{\eta}{N^2} \sum_{k,j} v_k u_j^\top$$

- Then, for any $k, y, t, x = u_y + n_t$, we have

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

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

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

- For any input embedding u_j, we have, thanks to near-orthonormality

$$\|W_0 u_j\| = \Theta_d(1) \quad \text{and} \quad \|\Delta W u_j\| = \Theta_d(1)$$

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

- For any input embedding u_j, we have, thanks to near-orthonormality

$$\|W_0 u_j\| = \Theta_d(1) \quad \text{and} \quad \|\Delta W u_j\| = \Theta_d(1)$$

- Contribution of updates is of similar order to initialization (not true for NTK!)
- Related to μP/mean-field (Chizat and Bach, 2018; Mei et al., 2019; Yang and Hu, 2021)

Link with feature learning

Maximal updates:

- First gradient update from standard initialization ($[W_0]_{ij} \sim \mathcal{N}(0, 1/d)$) take the form

$$W_1 = W_0 + \Delta W \in \mathbb{R}^{d \times d}, \quad \Delta W := \sum_j \alpha_j v_j u_j^\top, \quad \alpha_j = \Theta_d(1)$$

- For any input embedding u_j, we have, thanks to near-orthonormality

$$\|W_0 u_j\| = \Theta_d(1) \quad \text{and} \quad \|\Delta W u_j\| = \Theta_d(1)$$

- Contribution of updates is of similar order to initialization (not true for NTK!)
- Related to μP/mean-field (Chizat and Bach, 2018; Mei et al., 2019; Yang and Hu, 2021)

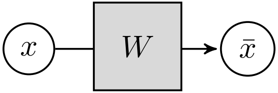
Large gradient steps on shallow networks:

- Useful for feature learning in **single-index** and **multi-index** models

$$y = f^*(x) + \text{noise}, \quad f^*(x) = g^*(Wx), \quad W \in \mathbb{R}^{r \times d}$$

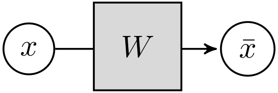
- Sufficient to break the curse of dimensionality when $r \ll d$
- (Ba et al., 2022; Damian et al., 2022; Dandi et al., 2023; Nichani et al., 2023)

Associative memories inside deep models



- Consider W that connects two nodes $x, \bar{x}$ in a feedforward computational graph

Associative memories inside deep models



- Consider W that connects two nodes $x, \bar{x}$ in a feedforward computational graph
- The loss gradient takes the form

$$\nabla_W L = \mathbb{E}[\nabla_{\bar{x}} \ell \cdot x^\top]$$

  where $\nabla_{\bar{x}} \ell$ is the **backward** vector (loss gradient w.r.t. $\bar{x}$)
- Often, this expectation may lead to associative memories as before
- A similar form can arise in attention matrices (see later!)

Questions

- **Finite capacity?** how much can we "store" with finite d?

Questions

- **Finite capacity?** how much can we “store” with finite d?
- **Finite samples?** how well can we learn with finite data?

Questions

- **Finite capacity?** how much can we "store" with finite d?
- **Finite samples?** how well can we learn with finite data?
- **Role of optimization algorithms?** multiple gradient steps? Adam?

Questions

- **Finite capacity?** how much can we "store" with finite d?
- **Finite samples?** how well can we learn with finite data?
- **Role of optimization algorithms?** multiple gradient steps? Adam?

$\Longrightarrow$ **study through scaling laws** (a.k.a. generalization bounds/statistical rates)

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: $S_n = \{z_1, \dots, z_n\}$, 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)

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

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

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

Scaling laws with finite capacity

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

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

- Single population gradient step: $q(z) \approx p(z)$

Scaling laws with finite capacity

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

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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 $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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 $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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 $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$ entries
- Estimator: $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, with

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

- Single population gradient step: $q(z) \approx p(z)$

Theorem (Cabannes, Dohmatob, B., 2023, 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

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

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

Scaling laws with optimization algorithms

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

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

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

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

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

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

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

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

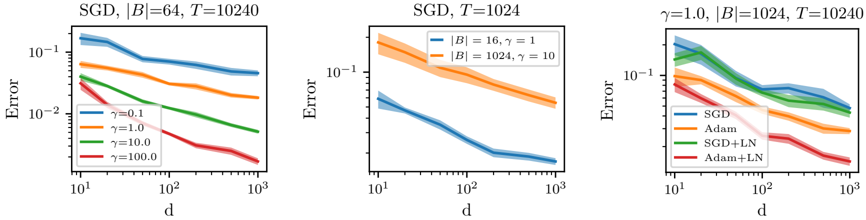
- One step of SGD with large batch: $q(z) \approx p(z)$
- SGD with batch size one + large step-size, $d \gg N$: $q(z) \approx 1$
- For $d \leq N$, smaller step-sizes can help later in training
- Adam and layer-norm help with practical settings (large batch sizes + smaller step-size)

Scaling laws with optimization algorithms

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

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)



Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$
- **Attention**: soft-max instead of hard-max to retrieve from context

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$
- **Attention**: soft-max instead of hard-max to retrieve from context
- **MLP**: $\hat{f}(z) = \arg\max_y v_y^\top \sum_{z'=1}^N v_{f^*(z')} \sigma(u_{z'}^\top u_z - b)$

Increasing capacity

Main idea: there are $\exp(d)$ near-orthogonal directions on the sphere

Strategies to increase memory capacity (from linear to exponential in d)

- **Nearest-neighbor** lookup: set $u_z = v_{f^*(z)}$ and take $\hat{f}(z) = \arg\max_y v_y^\top u_z$
- **Attention**: soft-max instead of hard-max to retrieve from context
- **MLP**: $\hat{f}(z) = \arg\max_y v_y^\top \sum_{z'=1}^N v_{f^*(z')} \sigma(u_{z'}^\top u_z - b)$

But: higher computational cost, more sensitive to noise, harder to learn