Transformers and Associative Memories

Alberto Bietti

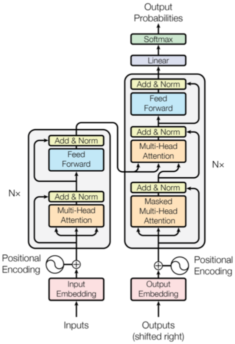
Flatiron Institute

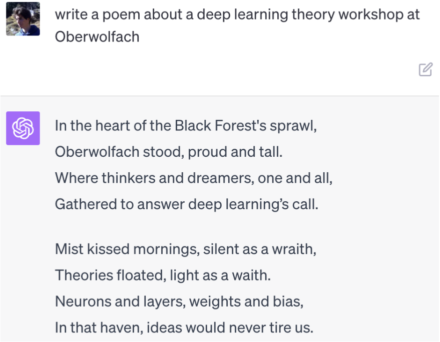
Oberwolfach. September 19, 2023.

w/ Vivien Cabannes, Elvis Dohmatob, Diane Bouchacourt, Hervé Jegou, Léon Bottou (Meta)

Transformers and language models

- **Transformers**: self-attention + MLPs + residual connections



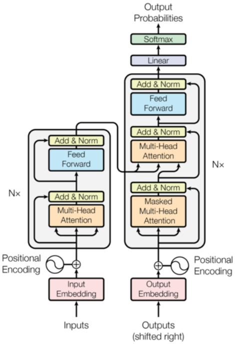


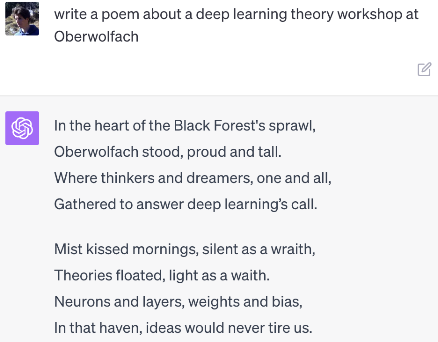
Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the Government for a redress of grievances.

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.

Transformers and language models

- **Transformers**: self-attention + MLPs + residual connections
- **Large language models**: train to predict next token on all the web (+ fine-tune)



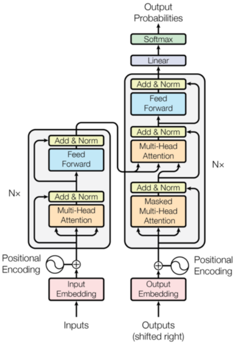


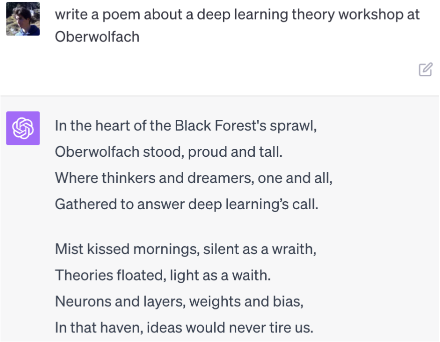
Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the Government for a redress of grievances.

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.

Transformers and language models

- **Transformers**: self-attention + MLPs + residual connections
- **Large language models**: train to predict next token on all the web (+ fine-tune)
- **In-context "reasoning" vs memorization**: transformers seem to use a mix of "reasoning" from context and "knowledge" from training set





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.

Motivating questions

- **Interpretability**: what's happening inside a transformer?

Motivating questions

- **Interpretability**: what's happening inside a transformer?
- **Training dynamics**: how is this learned during training?

Motivating questions

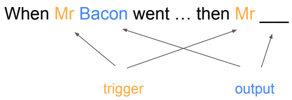
- **Interpretability**: what's happening inside a transformer?
- **Training dynamics**: how is this learned during training?
- **Role of depth**: can we go beyond shallow models?

Motivating questions

- **Interpretability**: what's happening inside a transformer?
- **Training dynamics**: how is this learned during training?
- **Role of depth**: can we go beyond shallow models?
- **Experimental/theory setup**: what is a simple setup for studying this?

The bigram data model

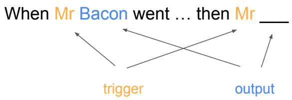
Goal: capture both in-context and global knowledge (*e.g.*, nouns vs syntax)



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

The bigram data model

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



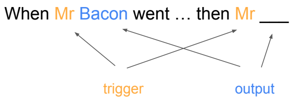
When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

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

- **Triggers**: $q_1, \ldots, q_K \sim \pi_q$ (*random* or *fixed once*)
- **Outputs**: $o_k \sim \pi_o(\cdot | q_k)$ (*random*)

The bigram data model

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



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

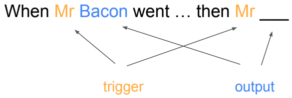
Sample each sequence $z_{1:T} \in [N]^T$ as follows

- **Triggers**: $q_1, \ldots, q_K \sim \pi_q$ (*random* or *fixed once*)
- **Outputs**: $o_k \sim \pi_o(\cdot|q_k)$ (*random*)
- **Sequence-specific Markov model**: $z_1 \sim \pi_1 \quad z_t|z_{t-1} \sim p(\cdot|z_{t-1})$

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

The bigram data model

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



When Mr Bacon went to the mall, it started raining, then Mr Bacon decided to buy a raincoat and umbrella. He went to the store and bought a red raincoat and yellow polka dot umbrella.

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

- **Triggers**: $q_1, \ldots, q_K \sim \pi_q$ (*random* or *fixed once*)
- **Outputs**: $o_k \sim \pi_o(\cdot | q_k)$ (*random*)
- **Sequence-specific Markov model**: $z_1 \sim \pi_1 \;\; z_t | z_{t-1} \sim p(\cdot | z_{t-1})$

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

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

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \ldots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \dots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$
- Intermediate layers: add outputs to the **residual stream** x_t

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \dots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$
- Intermediate layers: add outputs to the **residual stream** x_t
- **Unembedding layer**: logits for each $k \in [N]$,

$$(\xi_t)_k = w_U(k)^\top x_t$$

Transformers I: embeddings and residual stream

- **Input sequence**: $[z_1, \dots, z_T] \in [N]^T$
- **Embedding layer**:

$$x_t := w_E(z_t) + p_t \in \mathbb{R}^d$$

  - $w_E(z)$: token embedding of $z \in [N]$
  - p_t: positional embedding at position $t \in [T]$

- Intermediate layers: add outputs to the **residual stream** x_t

- **Unembedding layer**: logits for each $k \in [N]$,

$$(\xi_t)_k = w_U(k)^\top x_t$$

- **Loss** for next-token prediction (cross-entropy)

$$\sum_{t=1}^{T-1} \ell(z_{t+1}, \xi_t)$$

Transformers II: self-attention

Causal self-attention layer:

$$x'_t = \sum_{s=1}^{t} \beta_t W_O W_V x_s, \quad \text{with } \beta_s = \frac{\exp(x_s^\top W_K^\top W_Q x_t)}{\sum_{s=1}^{t} \exp(x_s^\top W_K^\top W_Q x_t)}$$

- $W_K, W_Q \in \mathbb{R}^{d \times d}$: **key** and **query** matrices
- $W_V, W_O \in \mathbb{R}^{d \times d}$: **value** and **output** matrices
- β_s: attention weights, $\sum_{s=1}^{t} \beta_s = 1$

Transformers II: self-attention

Causal self-attention layer:

$$x'_t = \sum_{s=1}^{t} \beta_t W_O W_V x_s, \quad \text{with } \beta_s = \frac{\exp(x_s^\top W_K^\top W_Q x_t)}{\sum_{s=1}^{t} \exp(x_s^\top W_K^\top W_Q x_t)}$$

- $W_K, W_Q \in \mathbb{R}^{d \times d}$: **key** and **query** matrices
- $W_V, W_O \in \mathbb{R}^{d \times d}$: **value** and **output** matrices
- β_s: attention weights, $\sum_{s=1}^{t} \beta_s = 1$
- **Single-head** attention (in practice, multi-head with multiple such matrices, $d_h \times d$)

Transformers II: self-attention

Causal self-attention layer:

$$x'_t = \sum_{s=1}^{t} \beta_t W_O W_V x_s, \quad \text{with } \beta_s = \frac{\exp(x_s^\top W_K^\top W_Q x_t)}{\sum_{s=1}^t \exp(x_s^\top W_K^\top W_Q x_t)}$$

- $W_K, W_Q \in \mathbb{R}^{d\times d}$: **key** and **query** matrices
- $W_V, W_O \in \mathbb{R}^{d\times d}$: **value** and **output** matrices
- β_s: attention weights, $\sum_{s=1}^t \beta_s = 1$
- **Single-head** attention (in practice, multi-head with multiple such matrices, $d_h \times d$)
- Each x'_t is then added to the corresponding residual stream

$$x_t := x_t + x'_t$$

Transformers III: feed-forward

Feed-forward layer: apply simple transformation to each token representation

- MLP (practice):

$$x_t' = W_2 \sigma(W_1 x_t), \qquad W_2 \in \mathbb{R}^{d \times D}, W_1 \in \mathbb{R}^{D \times d}$$

- Linear (in this work):

$$x_t' = W_F x_t, \qquad W_F \in \mathbb{R}^{d \times d}$$

Transformers III: feed-forward

Feed-forward layer: apply simple transformation to each token representation

- MLP (practice):

$$x_t' = W_2 \sigma(W_1 x_t), \qquad W_2 \in \mathbb{R}^{d \times D}, W_1 \in \mathbb{R}^{D \times d}$$

- Linear (in this work):

$$x_t' = W_F x_t, \qquad W_F \in \mathbb{R}^{d \times d}$$

- Added to the residual stream: $x_t := x_t + x_t'$

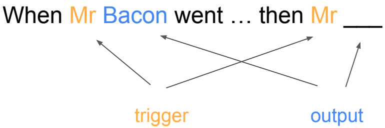
Transformers III: feed-forward

Feed-forward layer: apply simple transformation to each token representation

- MLP (practice):
$$x'_t = W_2\sigma(W_1 x_t), \qquad W_2 \in \mathbb{R}^{d\times D}, W_1 \in \mathbb{R}^{D\times d}$$
- Linear (in this work):
$$x'_t = W_F x_t, \qquad W_F \in \mathbb{R}^{d\times d}$$

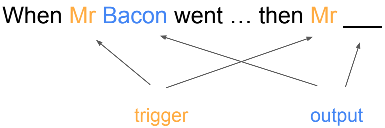
- Added to the residual stream: $x_t := x_t + x'_t$
- Some evidence that feed-forward layers store "global knowledge" (Geva et al., 2020; Meng et al., 2022)

Transformers on the bigram task

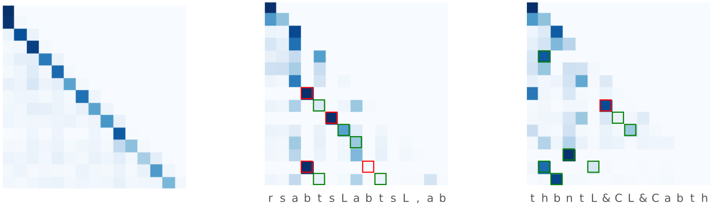


- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions
- **2-layer transformer succeeds**: $\sim 99\%$ accuracy

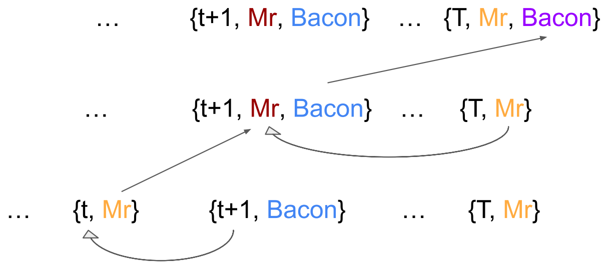
Transformers on the bigram task



- **1-layer transformer fails**: $\sim 55\%$ accuracy on in-context output predictions
- **2-layer transformer succeeds**: $\sim 99\%$ accuracy
- Attention maps reveal a structured 2-layer “induction” mechanism (Elhage et al., 2021)

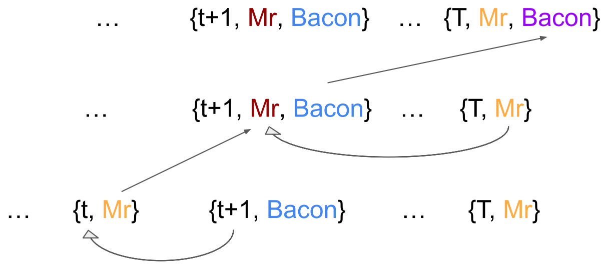


Induction head mechanism (Elhage et al., 2021; Olsson et al., 2022)



- 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

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 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 have $v_j^\top W u_i \approx \alpha_{ij}$
- Computed in Transformers for:
  - Logits in next-token prediction ($v_j = w_U(j)$, $u_i = x_t$)
  - Logits in attention heads ($v_j = x_k$, $u_i = x_q$)

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 have $v_j^\top W u_i \approx \alpha_{ij}$
- Computed in Transformers for:
  - Logits in next-token prediction ($v_j = w_U(j)$, $u_i = x_t$)
  - Logits in attention heads ($v_j = x_k$, $u_i = x_q$)

note: closely related to Hopfield and Willshaw networks (Hopfield, 1982; Willshaw et al., 1969)

Random embeddings in high dimension

- We consider embeddings u_i, v_j with i.i.d. $N(0, 1/d)$ entries, 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 embeddings u_i, v_j with i.i.d. $N(0, 1/d)$ entries, 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:

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

Random embeddings in high dimension

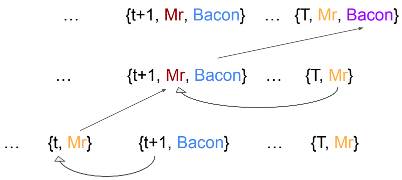
- We consider embeddings u_i, v_j with i.i.d. $N(0, 1/d)$ entries, d large

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

- **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, Bacon $\mapsto$ Bacon



Gradient associative memories

Lemma (Gradients as memories)

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)

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)

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

Gradient associative memories

Lemma (Gradients as memories)

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

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

Gradient associative memories

Lemma (Gradients as memories)

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)

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

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

- **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

Lemma (Gradients with noisy inputs)

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

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

Gradient associative memories with noisy inputs

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

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

Gradient associative memories with noisy inputs

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

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

- **Motivation**: the residual streams are sums of embeddings, some of which are irrelevant

Gradient associative memories with noisy inputs

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

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

- **Motivation**: the residual streams are sums of embeddings, some of which are irrelevant
- **Example**: $y \sim \mathrm{Unif}([N])$, $t \sim \mathrm{Unif}([T])$, $x = u_y + p_t$.

Gradient associative memories with noisy inputs

Lemma (Gradients with noisy inputs)

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

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

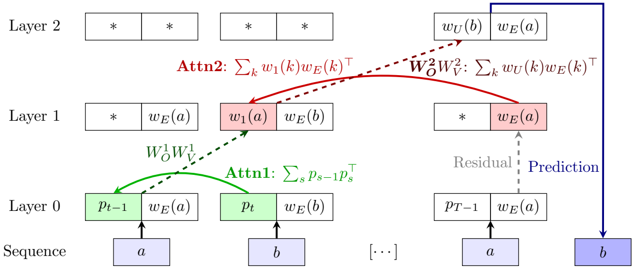
Denoting $\mu_k := \mathbb{E}[x|y = k]$ and $\hat{\mu}_k := \mathbb{E}_x[\frac{\hat{p}_W(k|x)}{p(y=k)} x]$, we have

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

- **Motivation**: the residual streams are sums of embeddings, some of which are irrelevant
- **Example**: $y \sim \text{Unif}([N])$, $t \sim \text{Unif}([T])$, $x = u_y + p_t$. One gradient step:

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

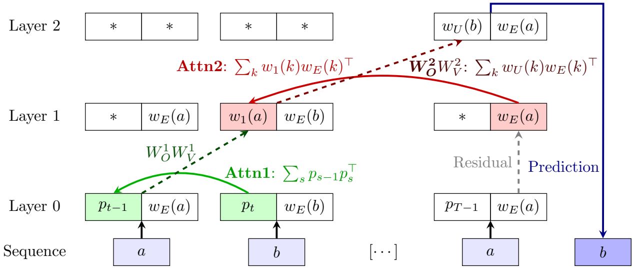
Induction head with associative memories



$$W_K^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_K^2 = \sum_{k \in Q} w_E(k) w_1(k)^\top, \quad W_O^2 = \sum_{k=1}^{N} w_U(k)(W_V^2 w_E(k))^\top,$$

- Random embeddings $w_E(k)$, $w_U(k)$, random matrices W_V^1, W_O^1, W_V^2, fix $W_Q = I$
- Remapped previous tokens: $w_1(k) := W_O^1 W_V^1 w_E(k)$

Induction head with associative memories



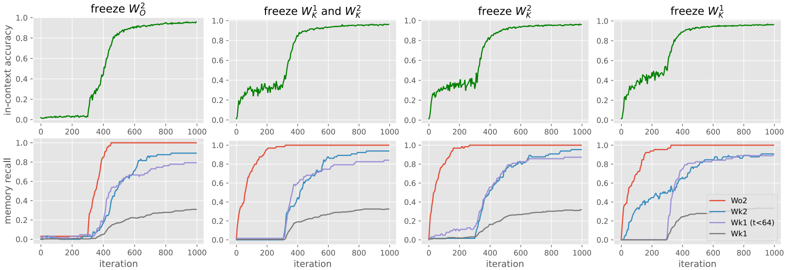
$$W_K^1 = \sum_{t=2}^{T} p_t p_{t-1}^\top, \quad W_K^2 = \sum_{k \in Q} w_E(k) w_1(k)^\top, \quad W_O^2 = \sum_{k=1}^{N} w_U(k) (W_V^2 w_E(k))^\top,$$

- Random embeddings $w_E(k)$, $w_U(k)$, random matrices W_V^1, W_O^1, W_V^2, fix $W_Q = I$
- Remapped previous tokens: $w_1(k) := W_O^1 W_V^1 w_E(k)$

Q: Does this match practice?

Empirically probing the dynamics

Train only W_K^1, W_K^2, W_O^2, loss on deterministic output tokens only

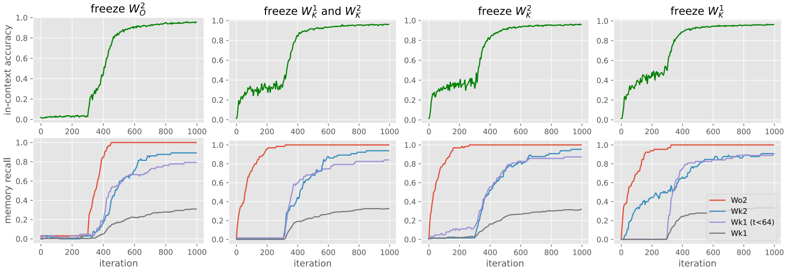


- "Memory recall **probes**": for target memory $W_* = \sum_{(i,j)\in\mathcal{M}} v_j u_i^\top$, compute

$$R(\hat{W}, W_*) = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \mathbb{1}\{j = \arg\max_{j'} v_{j'}^\top \hat{W} u_i\}$$

Empirically probing the dynamics

Train only W_K^1, W_K^2, W_O^2, loss on deterministic output tokens only



- "Memory recall **probes**": for target memory $W_* = \sum_{(i,j)\in\mathcal{M}} v_j u_i^\top$, compute

$$R(\hat{W}, W_*) = \frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \mathbb{1}\{j = \arg\max_{j'} v_{j'}^\top \hat{W} u_i\}$$

- Natural learning "**order**": W_O^2 first, W_K^2 next, W_K^1 last
- Joint learning is faster

Theoretical analysis with single gradient steps

Setting

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture

Theoretical analysis with single gradient steps

Setting

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture

Theorem (informal)

In the setup above, we can recover the desired associative memories with ***3 gradient steps*** *on the population loss, assuming near-orthonormal embeddings: first on W_O^2, then W_K^2, then W_K^1.*

Theoretical analysis with single gradient steps

Setting

- Focus on predicting second output token
- All distributions are uniform
- Some simplifications to architecture

Theorem (informal)

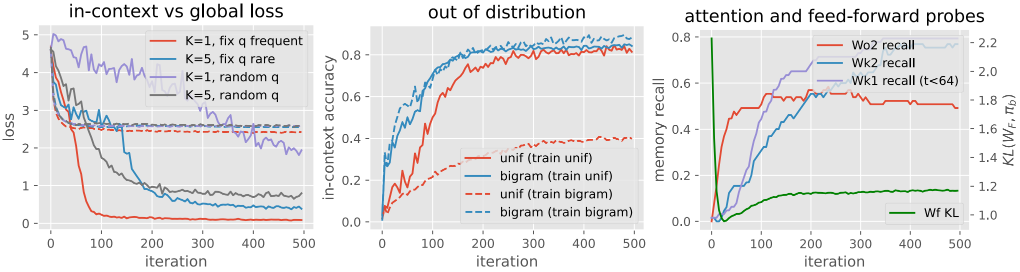
In the setup above, we can recover the desired associative memories with ***3 gradient steps*** *on the population loss, assuming near-orthonormal embeddings: first on* W_O^2*, then* W_K^2*, then* W_K^1*.*

Key ideas

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

Global vs in-context learning and role of data

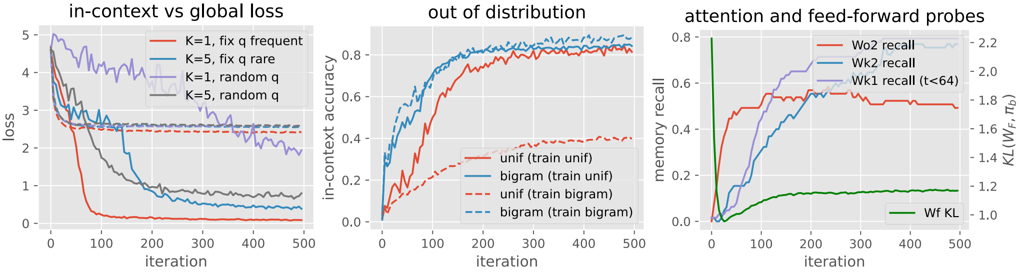
Train on all tokens, with added W_F after second attention layer



- Global bigrams learned quickly with W_F before induction mechanism

Global vs in-context learning and role of data

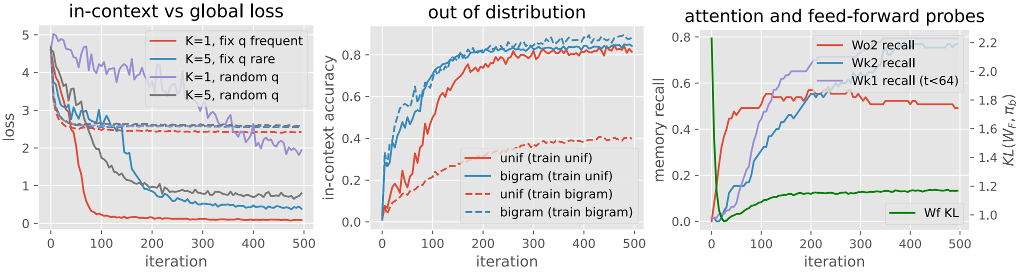
Train on all tokens, with added W_F after second attention layer



- Global bigrams learned quickly with W_F before induction mechanism
- More frequent *triggers* $\implies$ faster learning of induction head

Global vs in-context learning and role of data

Train on all tokens, with added W_F after second attention layer



- Global bigrams learned quickly with W_F before induction mechanism
- More frequent *triggers* $\implies$ faster learning of induction head
- More uniform *output* tokens helps OOD performance

What about more complex models?

- **Factorizations** (*e.g.*, $W_K^\top W_Q$): $y^\top UVx$
  - Low rank factorization can save parameters/compute
  - One joint gradient step from random initialization still works

What about more complex models?

- **Factorizations** (*e.g.*, $W_K^\top W_Q$): $y^\top UVx$
  - Low rank factorization can save parameters/compute
  - One joint gradient step from random initialization still works
- **Non-linear MLP**: $y^\top U\sigma(Vx)$
  - More expressive when x, y are *superpositions*/sums of embeddings
  - One gradient step still ok

What about more complex models?

- **Factorizations** (*e.g.*, $W_K^\top W_Q$): $y^\top UVx$
  - Low rank factorization can save parameters/compute
  - One joint gradient step from random initialization still works
- **Non-linear MLP**: $y^\top U\sigma(Vx)$
  - More expressive when x, y are *superpositions*/sums of embeddings
  - One gradient step still ok
- **Layer-norm**: $y^\top \frac{Wx}{\|Wx\|}$
  - Prevents repeated updates when Wx and y are already aligned
  - First gradient step from random initialization is unchanged

What about more complex models?

- **Factorizations** (*e.g.*, $W_K^\top W_Q$): $y^\top U V x$
  - Low rank factorization can save parameters/compute
  - One joint gradient step from random initialization still works
- **Non-linear MLP**: $y^\top U \sigma(Vx)$
  - More expressive when x, y are *superpositions*/sums of embeddings
  - One gradient step still ok
- **Layer-norm**: $y^\top \frac{Wx}{\|Wx\|}$
  - Prevents repeated updates when Wx and y are already aligned
  - First gradient step from random initialization is unchanged
- **Trained embeddings**
  - Single gradient steps capture basic co-occurrence statistics/BoW/topics
  - Or more complex learning of structured embeddings (*e.g.*, "grokking")

What about more complex models?

- **Factorizations** (*e.g.*, $W_K^\top W_Q$): $y^\top UVx$
  - Low rank factorization can save parameters/compute
  - One joint gradient step from random initialization still works
- **Non-linear MLP**: $y^\top U\sigma(Vx)$
  - More expressive when x, y are *superpositions*/sums of embeddings
  - One gradient step still ok
- **Layer-norm**: $y^\top \frac{Wx}{\|Wx\|}$
  - Prevents repeated updates when Wx and y are already aligned
  - First gradient step from random initialization is unchanged
- **Trained embeddings**
  - Single gradient steps capture basic co-occurrence statistics/BoW/topics
  - Or more complex learning of structured embeddings (*e.g.*, "grokking")

Does it work empirically on the bigram task? Yes!

- Memory recall probes $\rightarrow$ 1 as in previous experiment

What about more complex models?

- **Factorizations** (*e.g.*, $W_K^\top W_Q$): $y^\top UVx$
  - Low rank factorization can save parameters/compute
  - One joint gradient step from random initialization still works
- **Non-linear MLP**: $y^\top U\sigma(Vx)$
  - More expressive when x, y are *superpositions*/sums of embeddings
  - One gradient step still ok
- **Layer-norm**: $y^\top \frac{Wx}{\|Wx\|}$
  - Prevents repeated updates when Wx and y are already aligned
  - First gradient step from random initialization is unchanged
- **Trained embeddings**
  - Single gradient steps capture basic co-occurrence statistics/BoW/topics
  - Or more complex learning of structured embeddings (*e.g.*, "grokking")

Does it work empirically on the bigram task? Yes!

- Memory recall probes $\rightarrow 1$ as in previous experiment
- **But**: adding heads and layers loses identifiability

What about finite samples/width? “scaling laws”/rates

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, $i = 1, \dots, n$, $S_n = \{z_i\}_i$

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

What about finite samples/width? "scaling laws"/rates

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, $i = 1, \ldots, n$, $S_n = \{z_i\}_i$

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

- Zipf law: $p(z) \propto z^{-\alpha}$ (up to permutation)

What about finite samples/width? "scaling laws"/rates

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, $i = 1, \dots, n$, $S_n = \{z_i\}_i$

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

- Zipf law: $p(z) \propto z^{-\alpha}$ (up to permutation)
- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$

What about finite samples/width? "scaling laws"/rates

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, $i = 1, \dots, n$, $S_n = \{z_i\}_i$
$$L(\hat{f}_n) = \mathbb{P}(y \neq \hat{f}_n(z))$$
- Zipf law: $p(z) \propto z^{-\alpha}$ (up to permutation)
- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$

Theorem (informal, BCD23+, in prep.)

Consider the estimator $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, *with* $W_{n,d} = \sum_{z=1}^N q(z) v_{f^*(z)} u_z^\top$.

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

What about finite samples/width? "scaling laws"/rates

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, $i = 1, \dots, n$, $S_n = \{z_i\}_i$

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

- Zipf law: $p(z) \propto z^{-\alpha}$ (up to permutation)
- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$

Theorem (informal, BCD23+, in prep.)

Consider the estimator $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, *with* $W_{n,d} = \sum_{z=1}^N q(z) v_{f^*(z)} u_z^\top$.

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

- Finite-memory version of (Hutter, 2021)

What about finite samples/width? "scaling laws"/rates

Setting

- $z_i \sim p(z)$, $y_i = f^*(z_i)$, $i = 1, \dots, n$, $S_n = \{z_i\}_i$

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

- Zipf law: $p(z) \propto z^{-\alpha}$ (up to permutation)
- Random embeddings $u_z, v_y \in \mathbb{R}^d$ with $\mathcal{N}(0, 1/d)$

Theorem (informal, BCD23+, in prep.)

Consider the estimator $\hat{f}_{n,d}(x) = \arg\max_y v_y^\top W_{n,d} u_z$, *with* $W_{n,d} = \sum_{z=1}^N q(z) v_{f^*(z)} u_z^\top$.

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

- Finite-memory version of (Hutter, 2021)
- 2 and 3 are related to Adam and layer-norm

Discussion and next steps

Summary

- Bigram model: simple but rich toy model for discrete data
- Transformer weights as associative memories
- Learning via few top-down gradient steps

Discussion and next steps

Summary

- Bigram model: simple but rich toy model for discrete data
- Transformer weights as associative memories
- Learning via few top-down gradient steps

Future directions

- More complex “reasoning” mechanisms, links with “emergence”
- Learning dynamics: multiple gradient steps? joint training? embeddings?
- Applications: interpretability, model editing, factual recall, efficient fine-tuning
- LLM large-width scalings (links with μP)
- Replace weights by hash tables?? (a.k.a. Leon’s dream)

Discussion and next steps

Summary

- Bigram model: simple but rich toy model for discrete data
- Transformer weights as associative memories
- Learning via few top-down gradient steps

Future directions

- More complex "reasoning" mechanisms, links with "emergence"
- Learning dynamics: multiple gradient steps? joint training? embeddings?
- Applications: interpretability, model editing, factual recall, efficient fine-tuning
- LLM large-width scalings (links with μP)
- Replace weights by hash tables?? (a.k.a. Leon's dream)

Thank you!

References I

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.

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

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.

References II

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

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.

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.