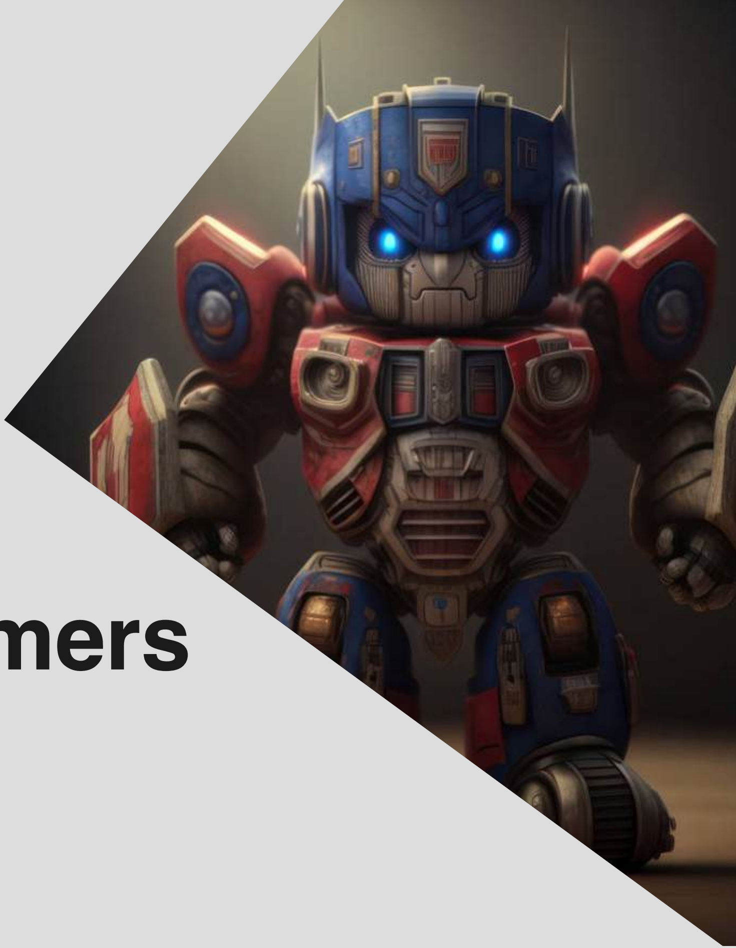
The Alan Turing Institute

# A perspective on the fundamentals of transformers

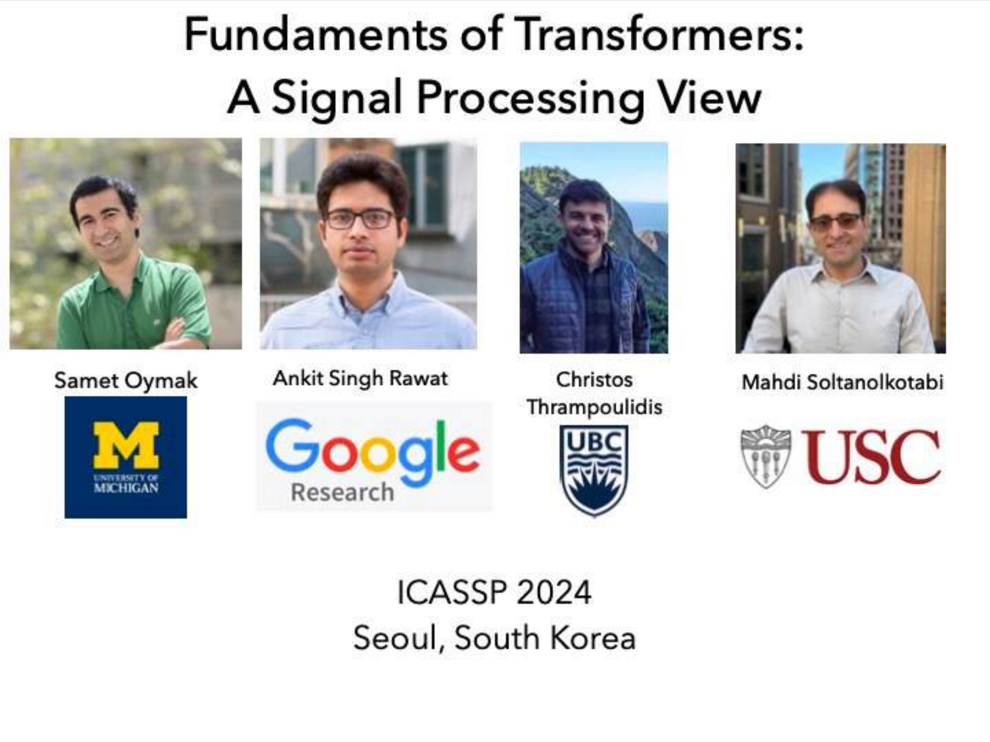


Edward Gunn, DARe

# Overview

- Transformers primer
- Optimisation
- Approximation
- Memorisation
- In-context learning

# Acknowledgements

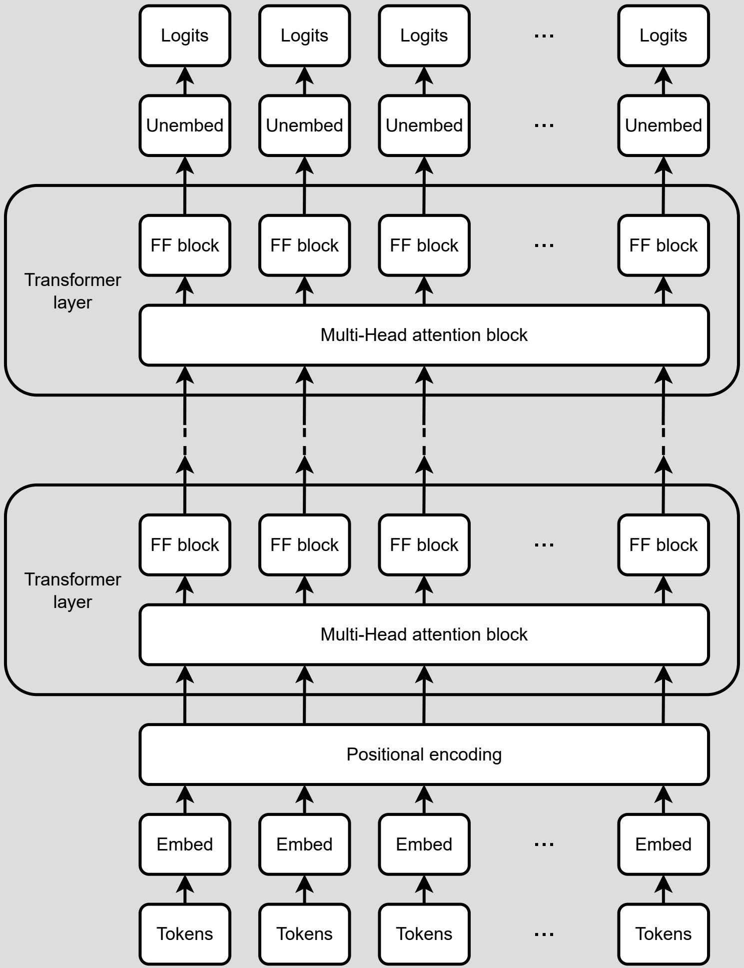



[Link]

# Overview

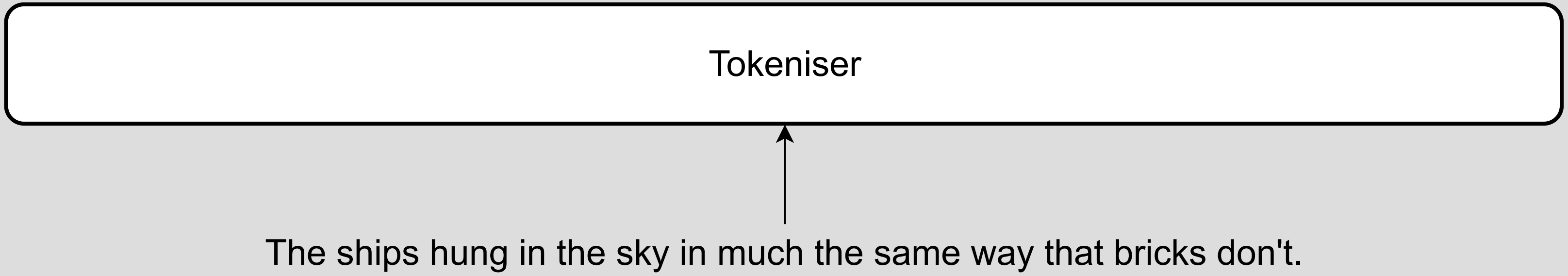
- **Transformers primer**
- Optimisation
- Approximation
- Memorisation
- In-context learning

# Transformer

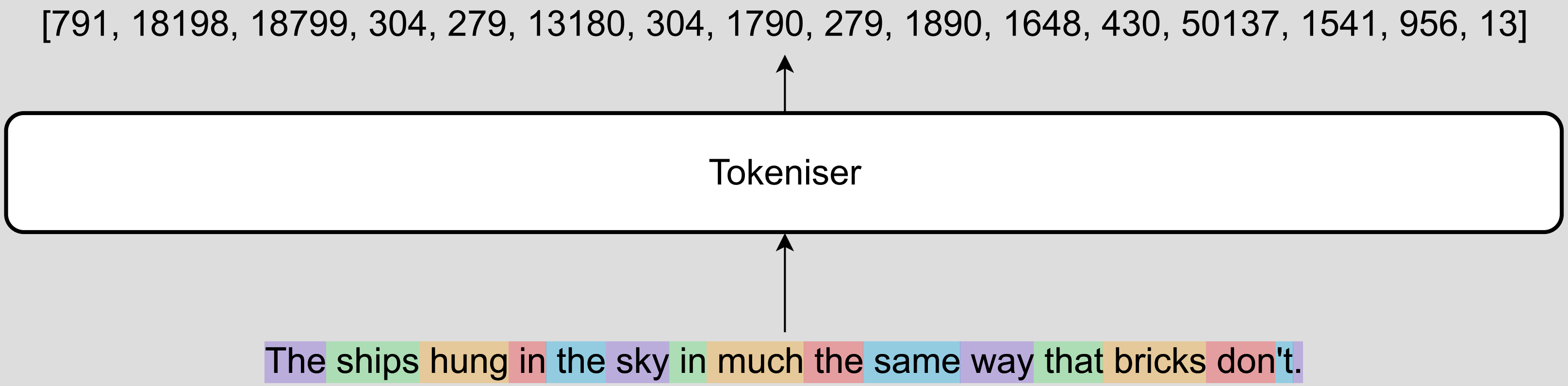



[Vaswani et al.'17]

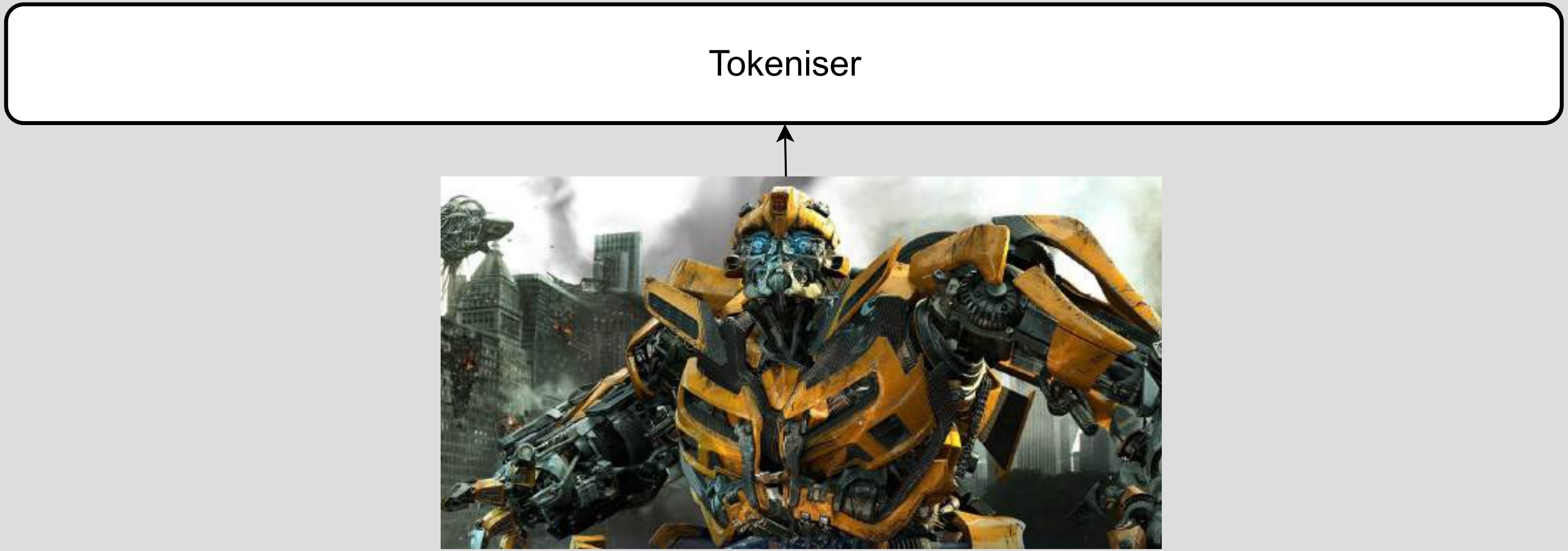
# Tokenization



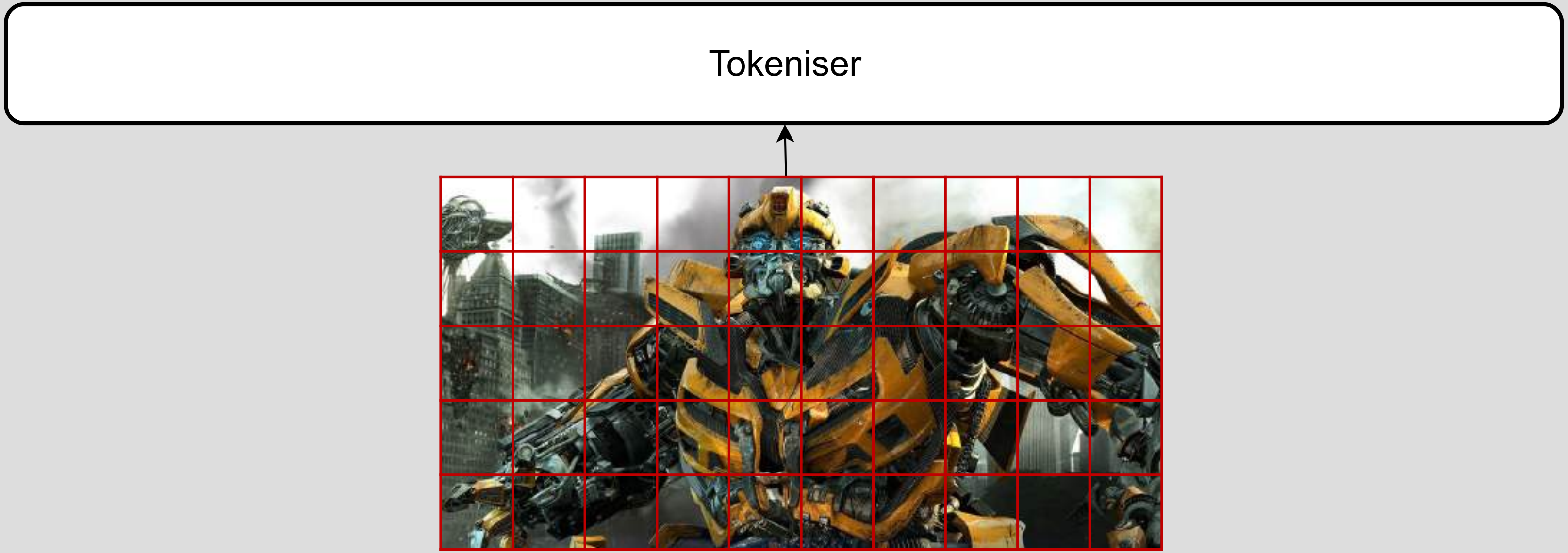
# Tokenization



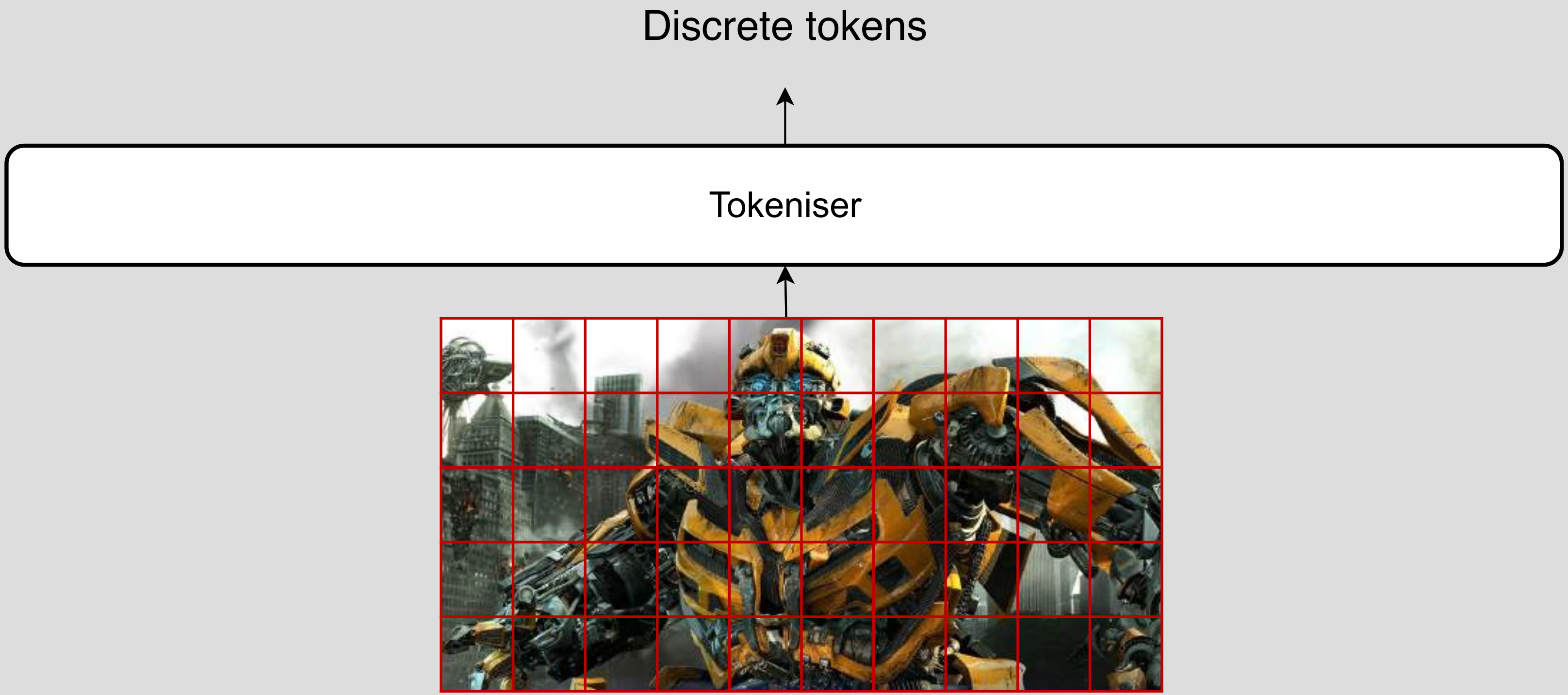
# Tokenization



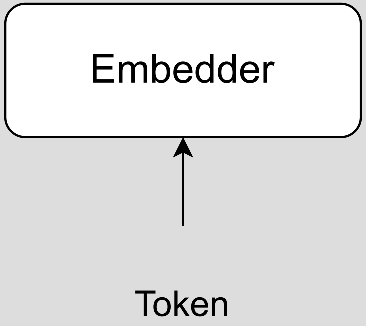
# Tokenization



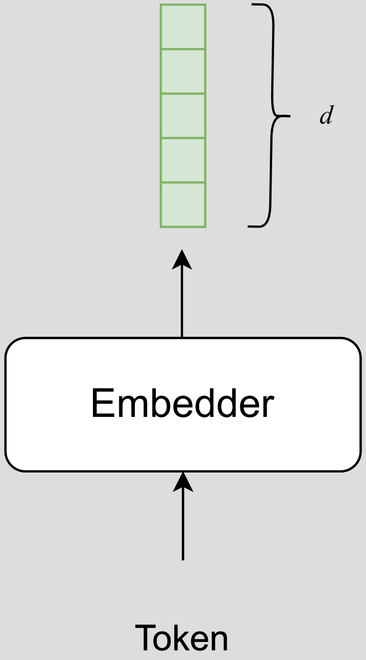
# Tokenization



# Embeddings

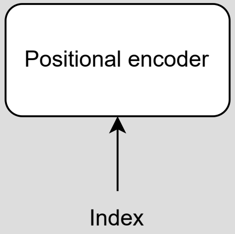


# Embeddings

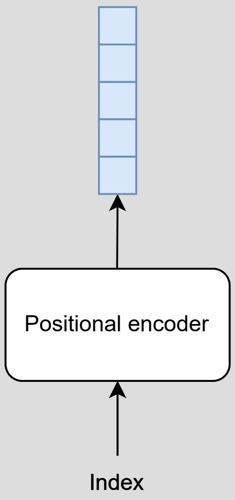


# Positional encodings

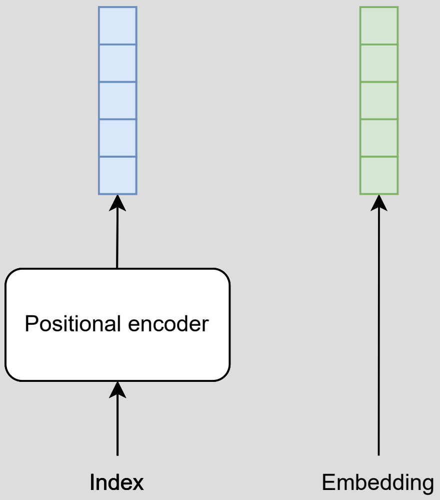
# Positional encodings



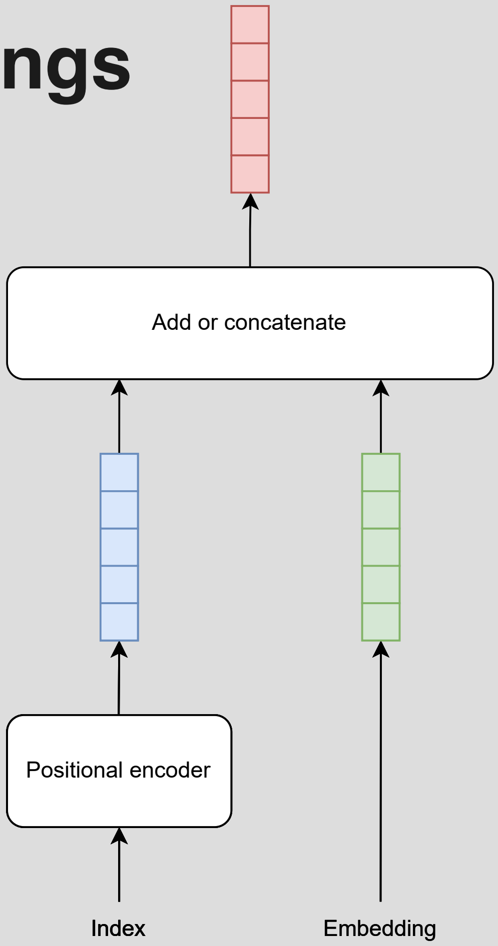
# Positional encodings



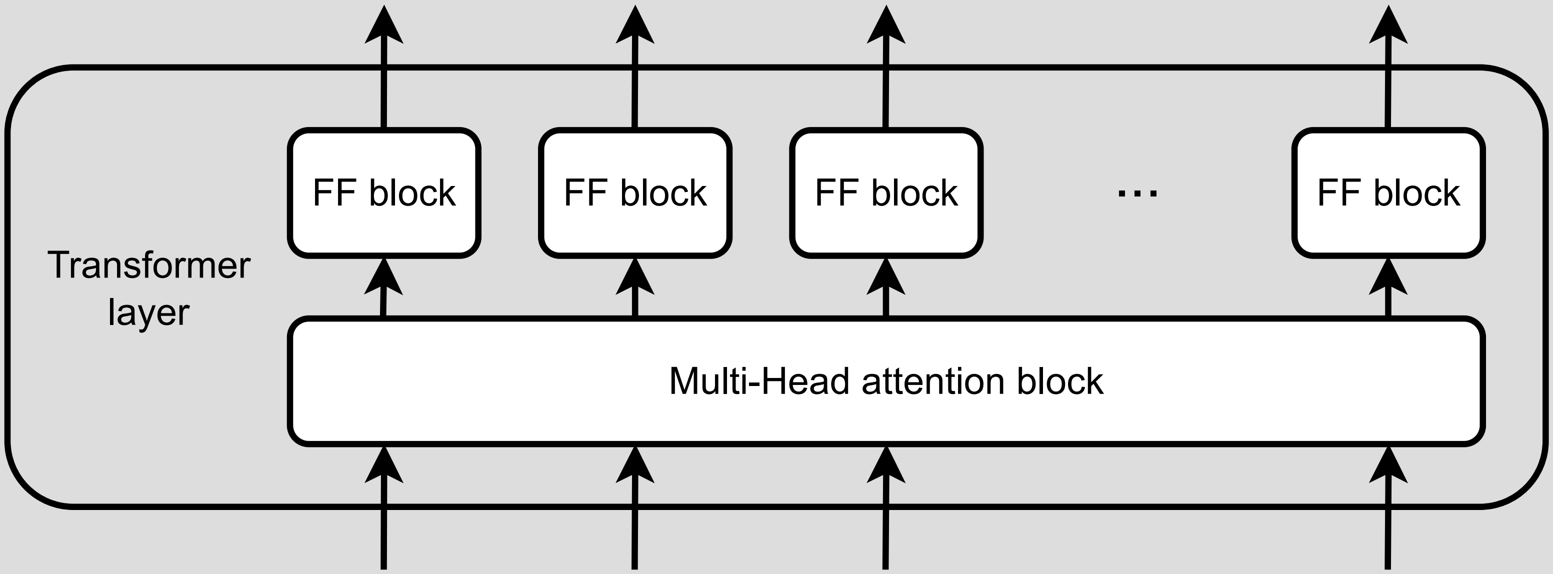
# Positional encodings



# Positional encodings

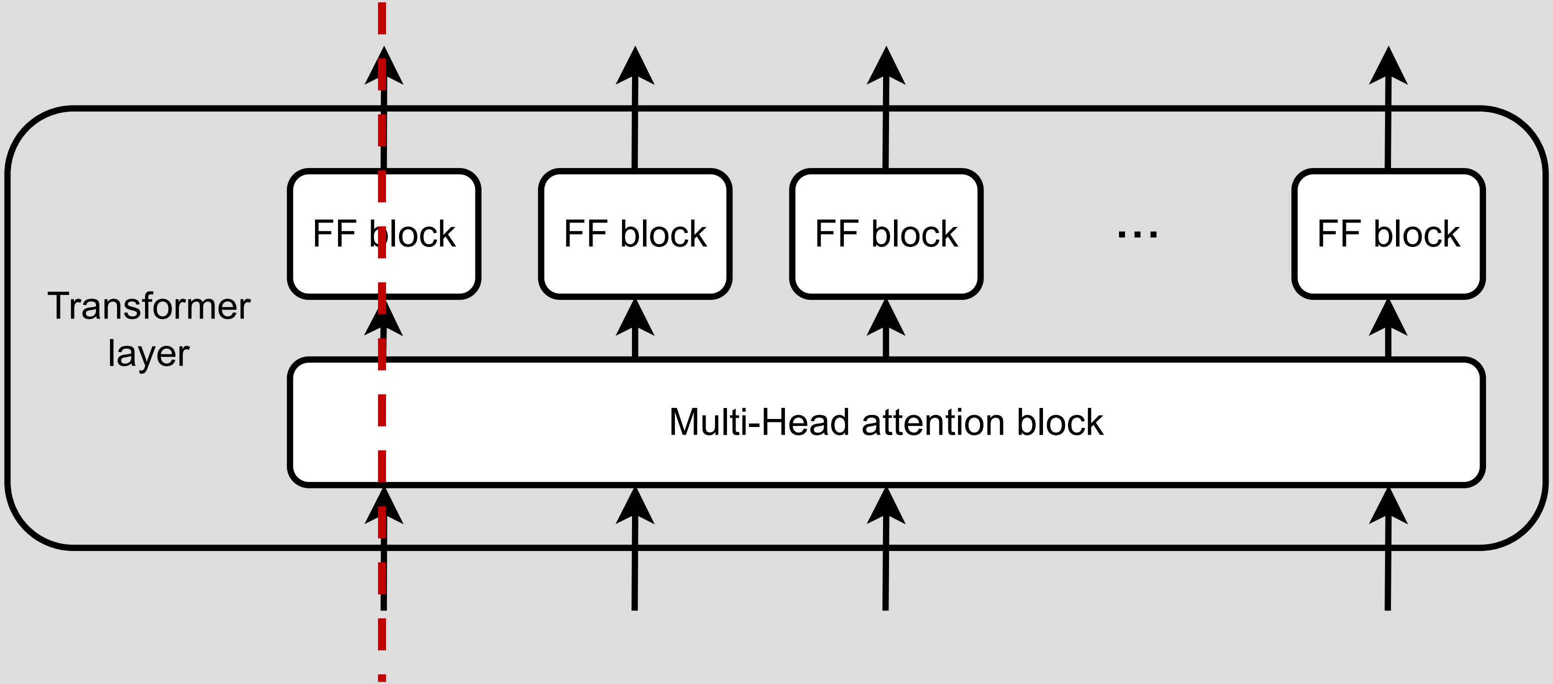


# Transformer layer



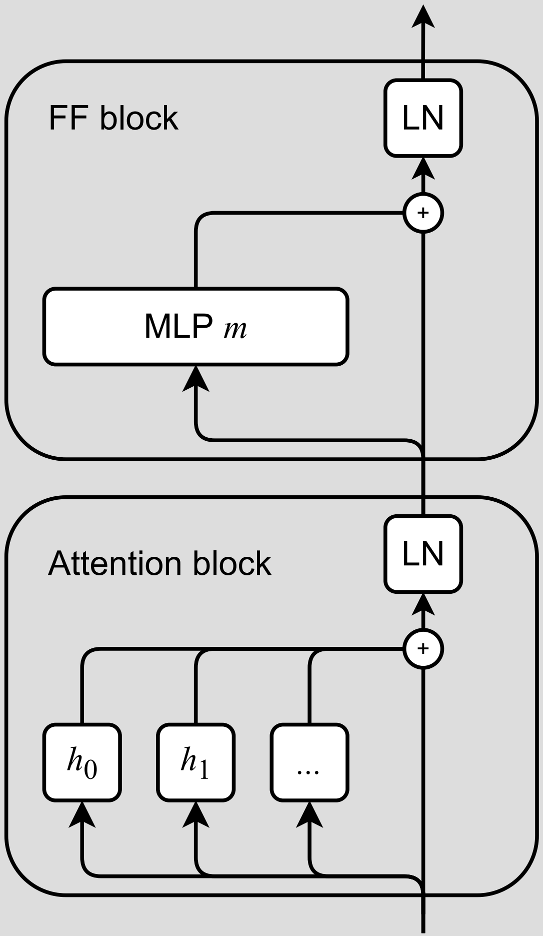

[Vaswani et al.'17]

# Transformer layer



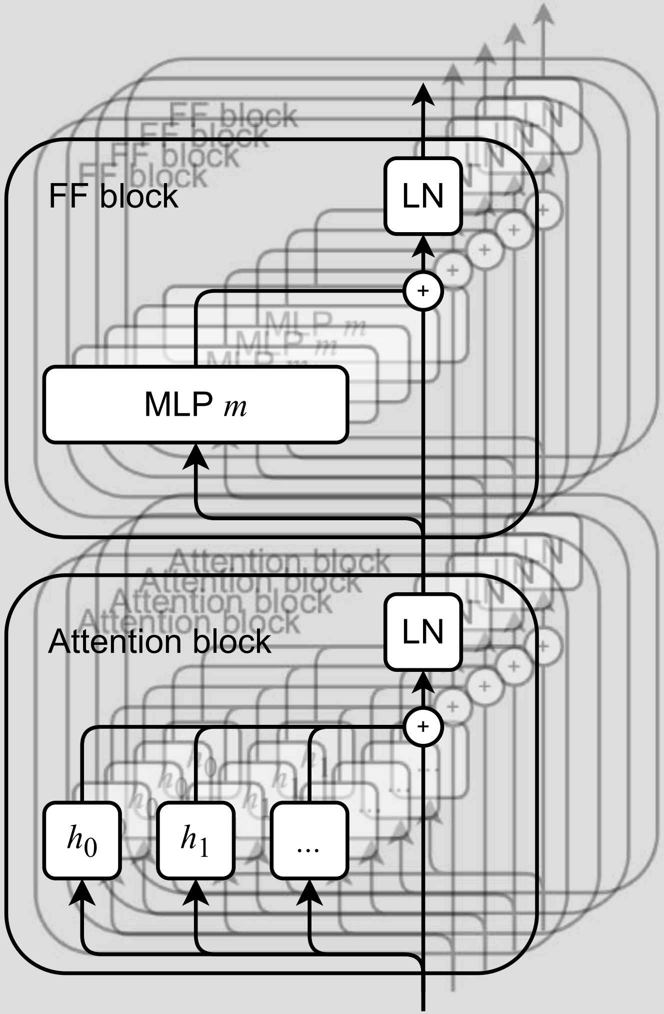

[Vaswani et al.'17]

# Transformer layer



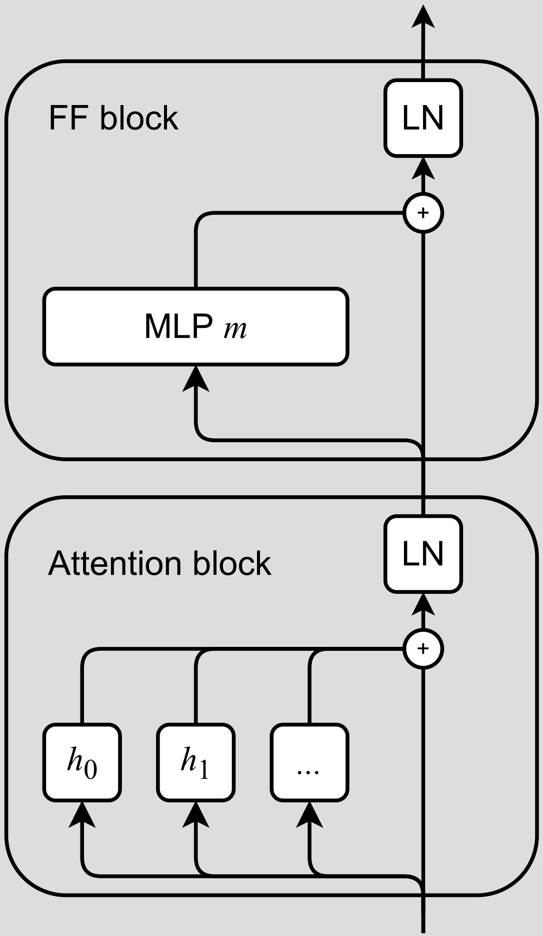

[Elhage et al.'21]

# Transformer layer



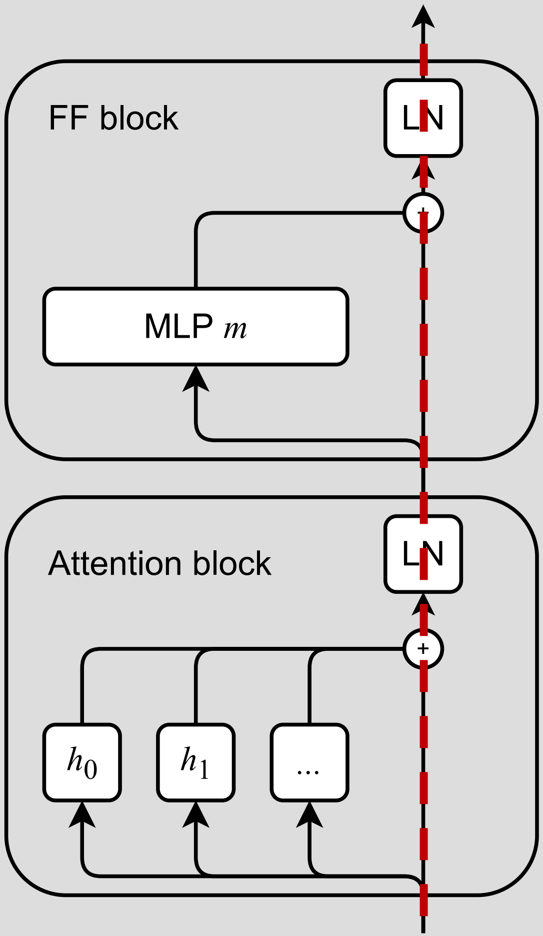

[Elhage et al.'21]

# Transformer layer



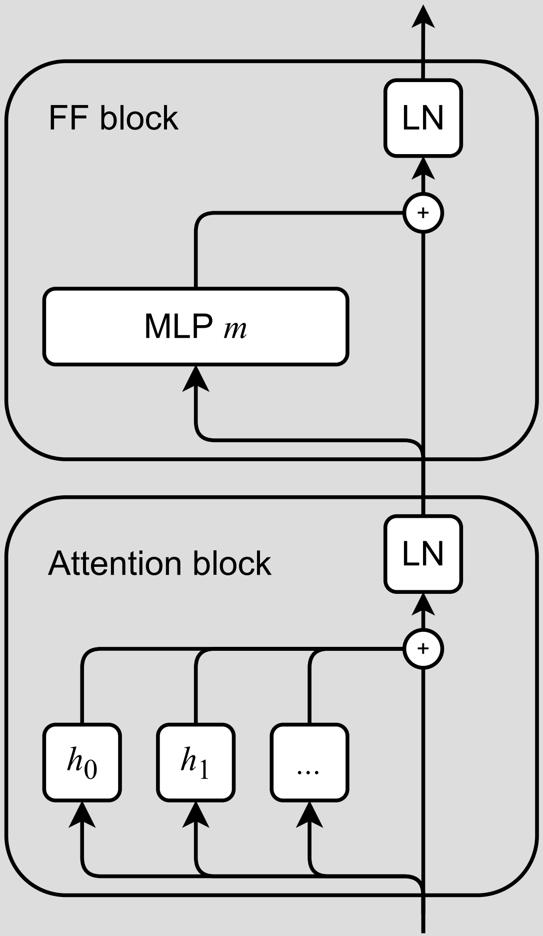

[Elhage et al.'21]

# Transformer layer




[Elhage et al.'21]

# Transformer layer

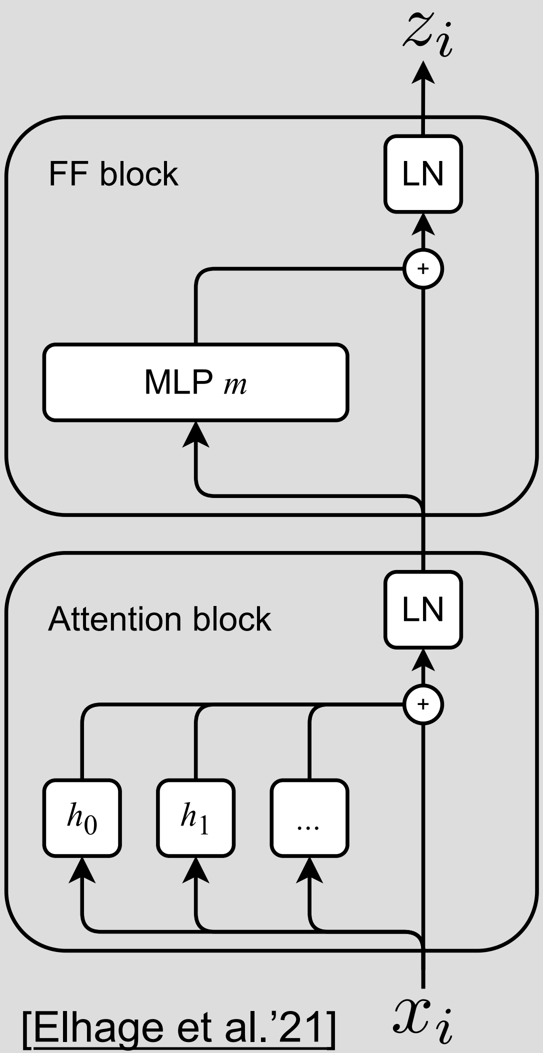



[Elhage et al.'21]

# Transformer layer

$$\hat{x}_i = \text{Attention}(X)$$
$$z_i = \text{FeedForward}(\hat{x}_i)$$

$$X = [x_1, x_2, \ldots, x_T]^\top$$



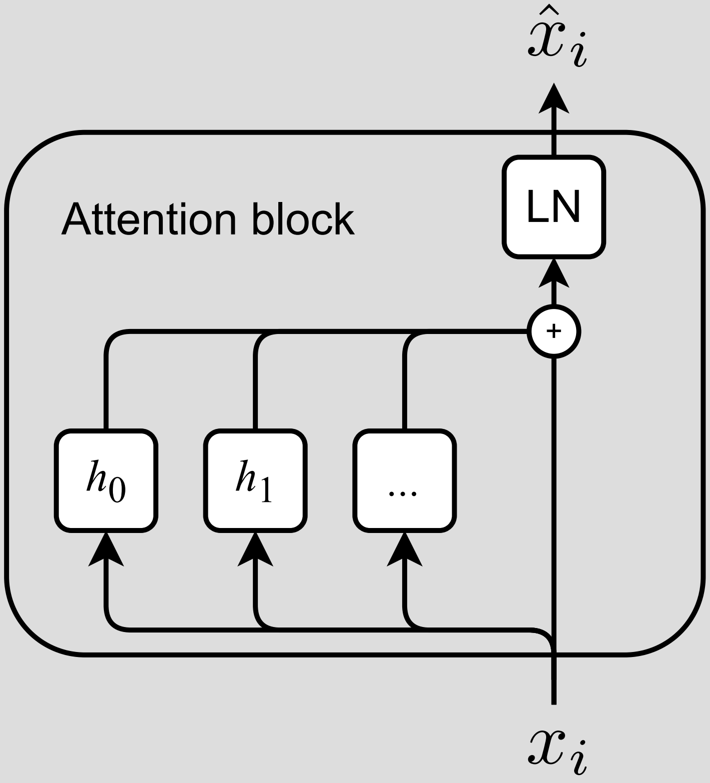

[Elhage et al.'21]

# Attention

$$\tilde{x}_i = x_i + \sum_{j=0}^{K} h_j(X)$$

$$\hat{x}_i = \mathrm{LN}(\tilde{x}_i)$$

$$X = [x_1, x_2, \ldots, x_T]^\top$$

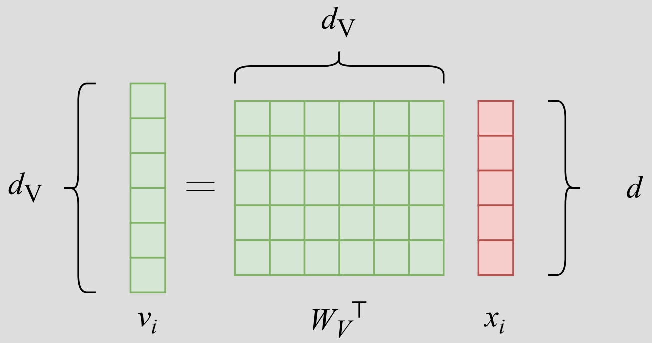


# Attention heads $h$

# Attention heads $h$

1. Compute the **value vector** for each token $x_i$

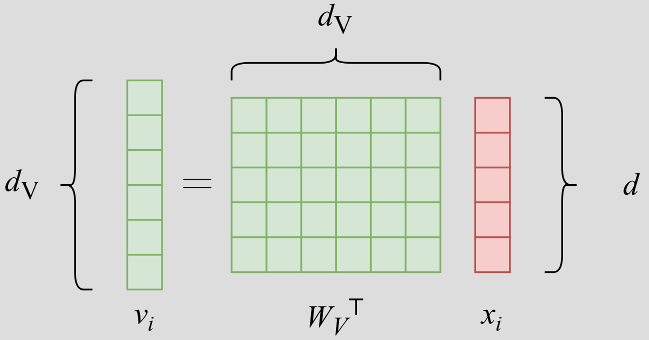
$$v_i = W_V^\top x_i$$



# Attention heads $h$

1. Compute the **value vector** for each token $x_i$

$$v_i = W_V^\top x_i$$



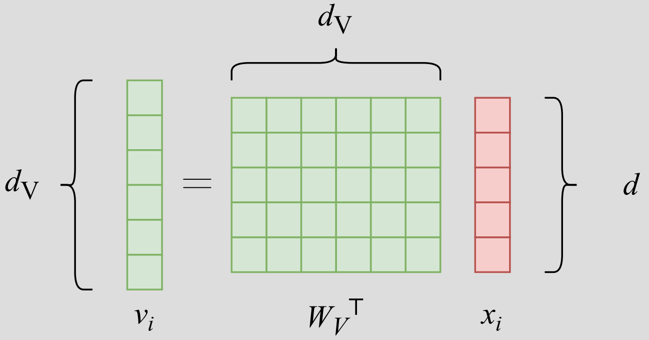

2. Compute the “result vector” by **linearly** combining value vectors according to the attention pattern

$$r_i = \sum_{j=1}^{T} A_{i,j} v_j$$

# Attention heads $h$

1. Compute the **value vector** for each token $x_i$

$$v_i = W_V^\top x_i$$




2. Compute the "result vector" by **linearly** combining value vectors according to the attention pattern

$$r_i = \sum_{j=1}^{T} A_{i,j} v_j$$

3. Compute the output vector of the head for each token

$$h(X) = W_O r_i$$

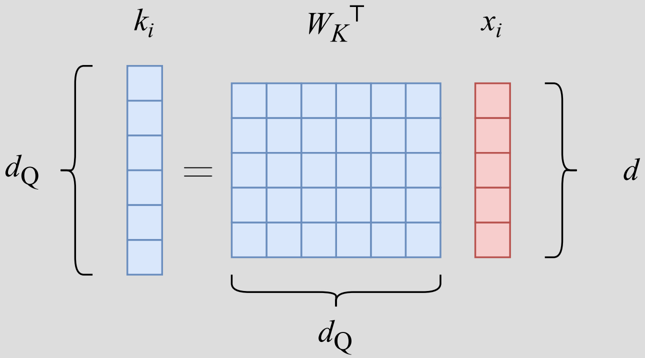
$$X = [x_1, x_2, \ldots, x_T]^\top$$

# Attention patterns

# Attention patterns

1. Compute the **key** vector for each token $x_i$

$$k_i = W_K^\top x_i$$



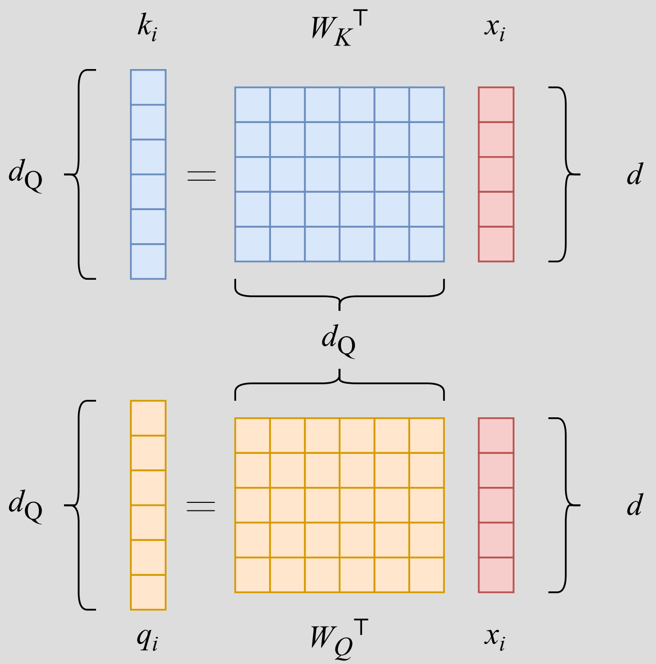
# Attention patterns

1. Compute the **key** vector for each token $x_i$

$$k_i = W_K^\top x_i$$

2. Compute the **query** vector for each token $x_i$

$$q_i = W_Q^\top x_i$$



$\mathbb{S} = \mathrm{Softmax}$

# Attention patterns

1. Compute the **key** vector for each token $x_i$

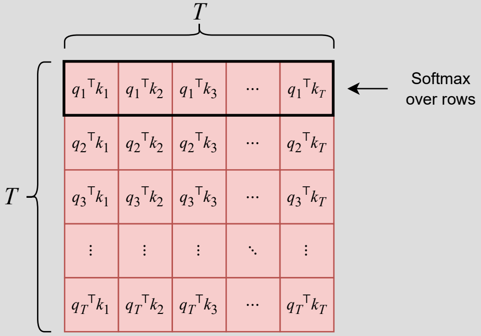
$$k_i = W_K^\top x_i$$

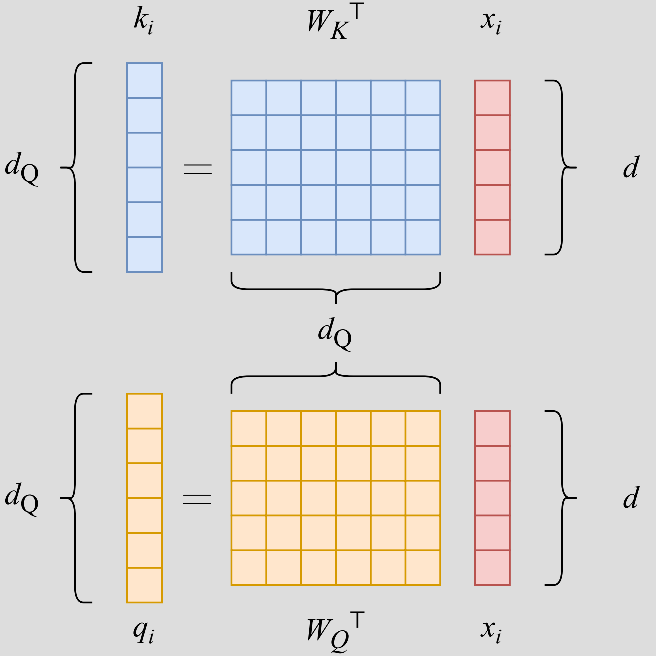
2. Compute the **query** vector for each token $x_i$

$$q_i = W_Q^\top x_i$$

3. Take the **softmax**

$$A = \mathbb{S}(QK^\top)$$




$$X = [x_1, x_2, \ldots, x_T]^\top$$

$$K = XW_K = [k_1, k_2, \ldots, k_T]^\top$$

$$Q = XW_Q = [q_1, q_2, \ldots, q_T]^\top$$

$\mathbb{S} = \text{Softmax}$

# Attention patterns

1. Compute the **key** vector for each token $x_i$

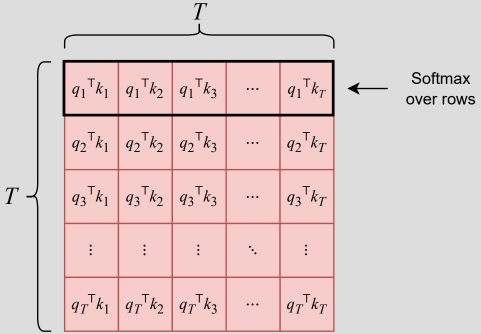
$$k_i = W_K^\top x_i$$

2. Compute the **query** vector for each token $x_i$

$$q_i = W_Q^\top x_i$$

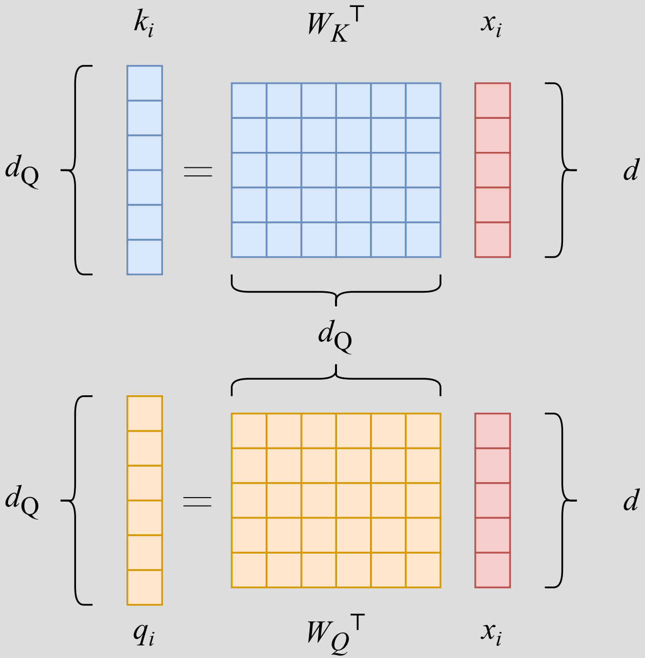
3. Take the **softmax**

$$A = \mathbb{S}(QK^\top)$$




4. We can alternatively do it in one step

$$A = \mathbb{S}(XW_Q W_K^\top X^\top)$$




$$X = [x_1, x_2, \ldots, x_T]^\top$$
$$K = XW_K = [k_1, k_2, \ldots, k_T]^\top$$
$$Q = XW_Q = [q_1, q_2, \ldots, q_T]^\top$$

$\mathbb{S} = \mathrm{Softmax}$

# Attention patterns

1. Compute the **key** vector for each token $x_i$

$$k_i = W_K^\top x_i$$

2. Compute the **query** vector for each token $x_i$

$$q_i = W_Q^\top x_i$$

3. Take the **softmax**

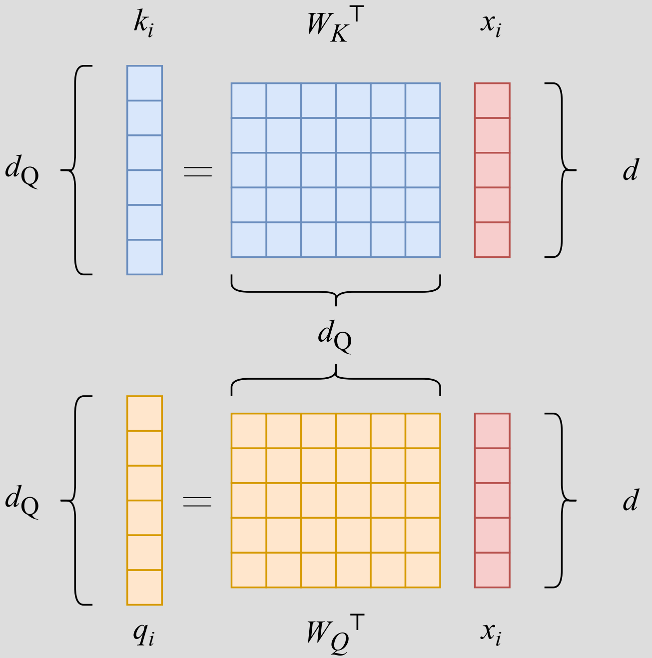
$$A = \mathbb{S}(QK^\top)$$

$$T\left\{\overbrace{\begin{bmatrix} q_1{}^\top k_1 & q_1{}^\top k_2 & q_1{}^\top k_3 & \cdots & q_1{}^\top k_T \\ q_2{}^\top k_1 & q_2{}^\top k_2 & q_2{}^\top k_3 & \cdots & q_2{}^\top k_T \\ q_3{}^\top k_1 & q_3{}^\top k_2 & q_3{}^\top k_3 & \cdots & q_3{}^\top k_T \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_T{}^\top k_1 & q_T{}^\top k_2 & q_T{}^\top k_3 & \cdots & q_T{}^\top k_T \end{bmatrix}}^{T}\right.$$

← Softmax over rows

4. We can alternatively do it in one step

$$A = \mathbb{S}(X\boxed{W_Q W_K^\top}X^\top)$$



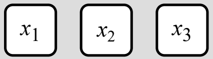

$$X = [x_1, x_2, \ldots, x_T]^\top$$

$$K = XW_K = [k_1, k_2, \ldots, k_T]^\top$$

$$Q = XW_Q = [q_1, q_2, \ldots, q_T]^\top$$

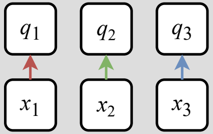
# Self attention

# Self attention



# Self attention

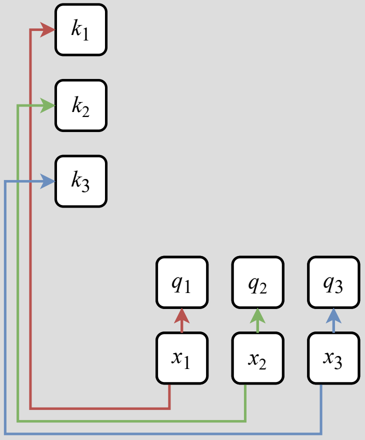
$$q_i = W_Q^\top x_i$$



# Self attention

$$q_i = W_Q^\top x_i$$

$$k_i = W_K^\top x_i$$

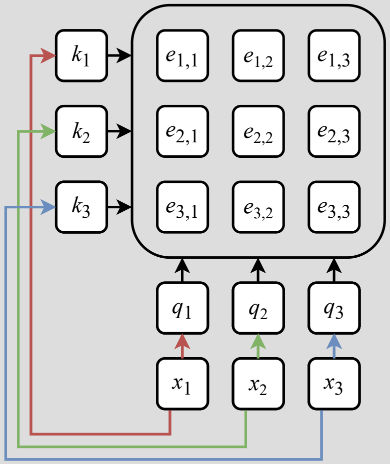


# Self attention

$$q_i = W_Q^\top x_i$$

$$k_i = W_K^\top x_i$$

$$e_{i,j} = q_j^\top k_i$$



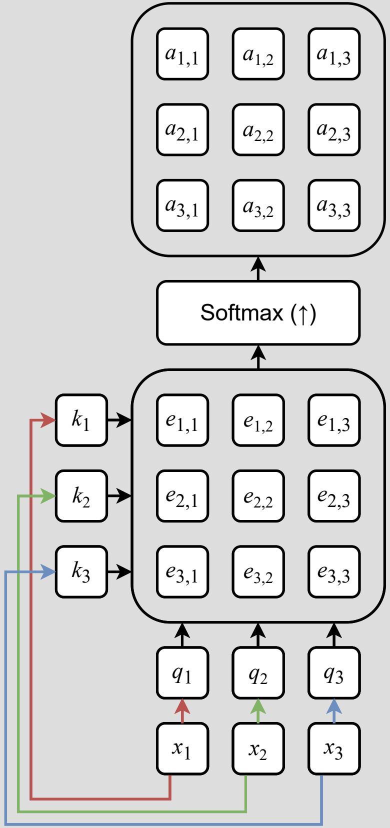
# Self attention

$$q_i = W_Q^\top x_i$$

$$k_i = W_K^\top x_i$$

$$e_{i,j} = q_j^\top k_i$$

$$[a_{1,j}, \ldots, a_{T,j}] = \mathbb{S}([e_{1,j}, \ldots, e_{T,j}])$$




$\mathbb{S} = \mathrm{Softmax}$

# Self attention

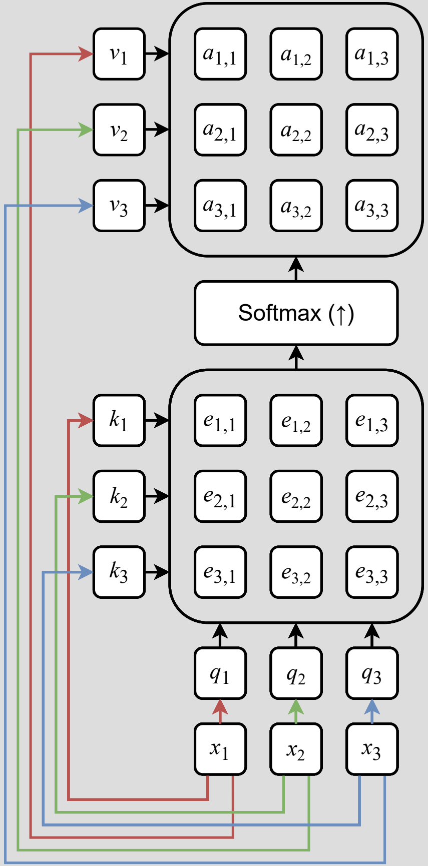
$$q_i = W_Q^\top x_i$$

$$k_i = W_K^\top x_i$$

$$e_{i,j} = q_j^\top k_i$$

$$[a_{1,j}, \ldots, a_{T,j}] = \mathbb{S}([e_{1,j}, \ldots, e_{T,j}])$$

$$v_i = W_V^\top x_i$$




$\mathbb{S} = \text{Softmax}$

# Self attention

$$q_i = W_Q^\top x_i$$

$$k_i = W_K^\top x_i$$

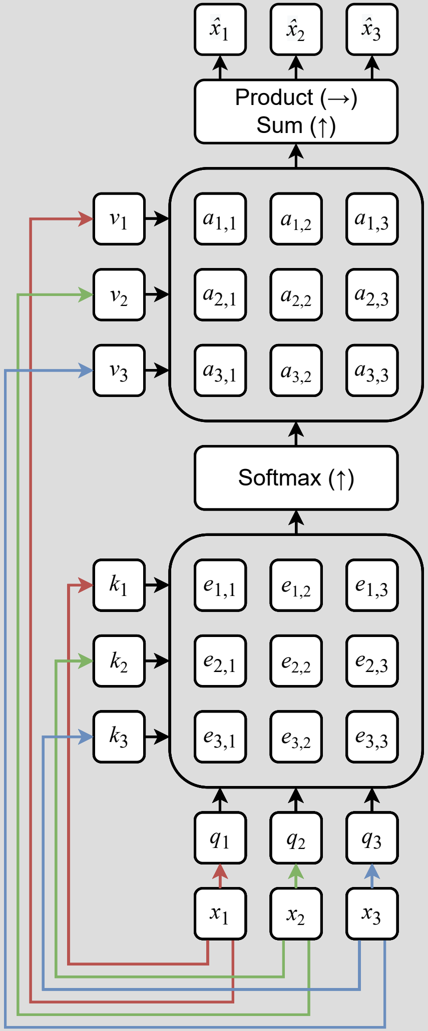
$$e_{i,j} = q_j^\top k_i$$

$$[a_{1,j}, \ldots, a_{T,j}] = \mathbb{S}([e_{1,j}, \ldots, e_{T,j}])$$

$$v_i = W_V^\top x_i$$

$$\hat{x}_i = \sum_{j=1}^{T} a_{i,j} v_j$$

$\mathbb{S} = \text{Softmax}$



# Self attention

# Self attention

- Putting it all together:

$$\hat{X} = \mathbb{S}(QK^\top)V$$

$$\mathbb{S} = \mathrm{Softmax}$$

# Self attention

- Putting it all together:

$$\hat{X} = \mathbb{S}(QK^\top)V$$

$$\hat{X} = \mathbb{S}(XW_Q W_K^\top X^\top)XW_V$$

$$\mathbb{S} = \mathrm{Softmax}$$

$$X = [x_1, x_2, \ldots, x_T]^\top$$

# Self attention

- Putting it all together:

$$\hat{X} = \mathbb{S}(QK^\top)V$$

$$\hat{X} = \mathbb{S}(XW_Q W_K^\top X^\top)XW_V$$

- In practice scale by dimension to avoid small gradients:

$$\hat{X} = \mathbb{S}(\frac{QK^\top}{\sqrt{d_k}})V$$

$$\mathbb{S} = \mathrm{Softmax}$$

$$X = [x_1, x_2, \ldots, x_T]^\top$$

# Overview

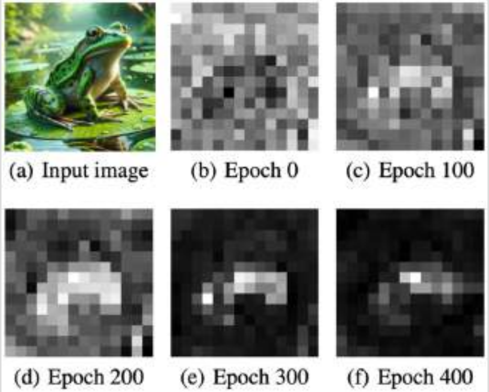
- Transformers primer
- **Optimisation**
- Approximation
- Memorisation
- In-context learning

Optimiz prime

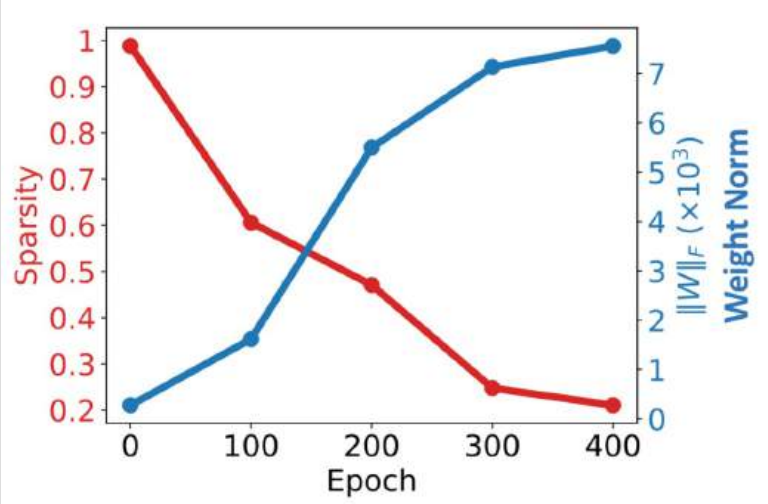
# Empirical motivations

1. Attention map is sparse

2. Attention map gets sparser as training evolves









3. Attention weights increase in norm

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$
    1. Read the token's last output: $x^{\mathrm{att}} = X^\top \mathbb{S}(XWz)$ (last token $z$)

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$
    1. Read the token's last output: $x^{\text{att}} = \boxed{X^\top \mathbb{S}(XWz)}$ (last token $z$)

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$
    1. Read the token's last output: $x^{\text{att}} = X^\top \mathbb{S}(XWz)$ (last token $z$)

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$

    1. Read the token's last output: $x^{\text{att}} = X^\top \mathbb{S}(XWz)$ (last token $z$)
    2. Linear decoder $v \in \mathbb{R}^d$

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$

    1. Read the token's last output: $x^{\mathrm{att}} = X^\top \mathbb{S}(XWz)$ (last token $z$)
    2. Linear decoder $v \in \mathbb{R}^d$

2. Prediction: $\hat{y}(X) = v^\top x^{\mathrm{att}}$

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$
   1. Read the token's last output: $x^{\text{att}} = X^\top \mathbb{S}(XWz)$ (last token $z$)
   2. Linear decoder $v \in \mathbb{R}^d$
2. Prediction: $\hat{y}(X) = v^\top x^{\text{att}}$
3. Loss: $\ell$ strictly decreasing smooth loss (e.g. logistic)

[Tarzanagh et al.'23b]

# Toy classification model

1. Classification: Map input sequence $X \in \mathbb{R}^{T \times d}$ to label $y \in \{-1, 1\}$
    1. Read the token's last output: $x^{\text{att}} = X^\top \mathbb{S}(XWz)$ (last token $z$)
    2. Linear decoder $v \in \mathbb{R}^d$
2. Prediction: $\hat{y}(X) = v^\top x^{\text{att}}$
3. Loss: $\ell$ strictly decreasing smooth loss (e.g. logistic)
4. Training objective:

$$\min_W \left\{ \mathcal{L}(W) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i \cdot v^\top x_i^{\text{att}}) \text{ where } x_i^{\text{att}} = X_i^\top \mathbb{S}(X_i W z_i) \right\}$$

[Tarzanagh et al.'23b]

# Gradient descent vs regularization paths

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top X_i^\top \mathbb{S}(X_i W z_i)\right)$$

[Ji et al.'20]

# Gradient descent vs regularization paths

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i W z_i)}_{x_i^{\text{att}}}\right)$$

[Ji et al.'20]

# Gradient descent vs regularization paths

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i W z_i)}_{x_i^{\text{att}}}\right)$$

What attention weights does GD find?

[Ji et al.'20]

# Gradient descent vs regularization paths

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i W z_i)}_{x_i^{\text{att}}}\right)$$

What attention weights does GD find?

Gradient descent trajectory

$$\text{Given } W_0 \in \mathbb{R}^{d\times d}, \eta > 0, \text{ do:}$$
$$W_{k+1} = W_k - \eta \nabla \mathcal{L}(W_k)$$

[Ji et al.'20]

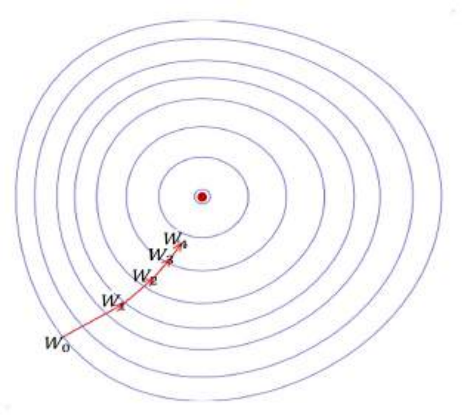
# Gradient descent vs regularization paths

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i {\color{red}W} z_i)}_{x_i^{\text{att}}}\right)$$

What attention weights does GD find?

Gradient descent trajectory

$$\text{Given } W_0 \in \mathbb{R}^{d\times d}, \eta > 0, \text{ do:}$$
$$W_{k+1} = W_k - \eta \nabla \mathcal{L}(W_k)$$




[Ji et al.'20]

# Gradient descent vs regularization paths

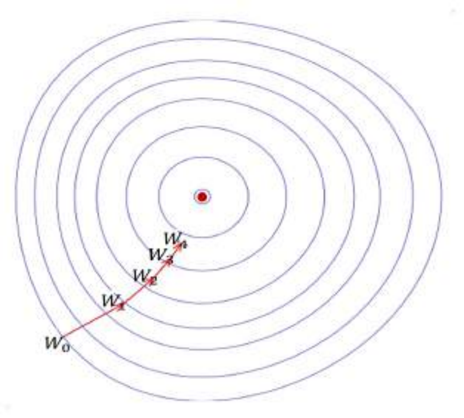
$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i {\color{red}W} z_i)}_{x_i^{\mathrm{att}}}\right)$$

What attention weights does GD find?

Gradient descent trajectory

$$\text{Given } W_0 \in \mathbb{R}^{d\times d}, \eta > 0, \text{ do:}$$
$$W_{k+1} = W_k - \eta \nabla \mathcal{L}(W_k)$$

$$\lim_{k\to\infty} W_k =$$




[Ji et al.'20]

# Gradient descent vs regularization paths

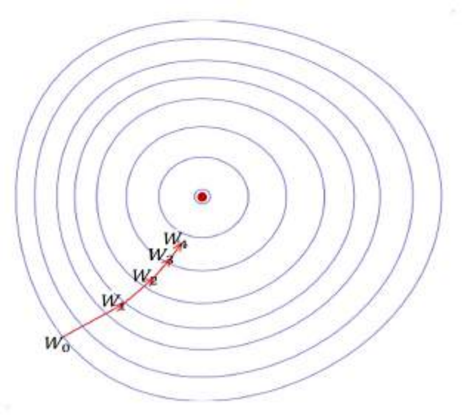
$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i {\color{red}W} z_i)}_{x_i^{\text{att}}}\right)$$

What attention weights does GD find?

Gradient descent trajectory

$$\text{Given } W_0 \in \mathbb{R}^{d\times d}, \eta > 0, \text{ do:}$$
$$W_{k+1} = W_k - \eta \nabla \mathcal{L}(W_k)$$

$$\lim_{k\to\infty} W_k = {\color{red}???}$$



[Ji et al.'20]

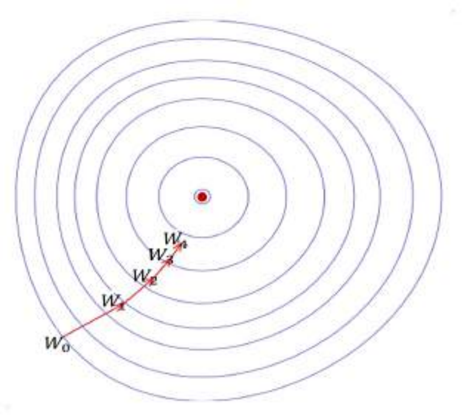
# Gradient descent vs regularization paths

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i \cdot v^\top \underbrace{X_i^\top \mathbb{S}(X_i W z_i)}_{x_i^{\text{att}}}\right)$$

What attention weights does GD find?

Gradient descent trajectory

$$\text{Given } W_0 \in \mathbb{R}^{d\times d}, \eta > 0, \text{ do:}$$
$$W_{k+1} = W_k - \eta \nabla \mathcal{L}(W_k)$$




$$\lim_{k\to\infty} W_k = ???$$
$$\approx$$
$$\lim_{R\to\infty} W_R = ???$$

Regularization path proxy for GD

$$\text{Given } R > 0, \text{ find } d \times d \text{ matrix:}$$
$$\bar{W}_R = \arg \min_{\|W\|_F \le R}$$

[Ji et al.'20]

# Softmax intuition

# Softmax intuition

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^{T} e^{l_\tau}} = \frac{1}{1 + \sum_{\tau \neq t} e^{-(l_t - l_\tau)}}$$

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^{T} e^{l_\tau}} = \frac{1}{1 + \sum_{\tau \neq t} e^{-(l_t - l_\tau)}}$$

$$\sum_{t=1}^{T} \mathbb{S}(l)_t = 1 \text{ and } 1 > \mathbb{S}(l)_t > 0$$

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^{T} e^{l_\tau}} = \frac{1}{1 + \sum_{\tau \neq t} e^{-(l_t - l_\tau)}}$$

$$\sum_{t=1}^{T} \mathbb{S}(l)_t = 1 \text{ and } 1 > \mathbb{S}(l)_t > 0$$

- The only way to attain $\mathbb{S}(l)_t = 1$ is:
    i. $\|l\| \to \infty$
    ii. $l_t > l_\tau$ for all $\tau \neq t$

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^{T} e^{l_\tau}} = \frac{1}{1 + \boxed{\sum_{\tau \neq t} e^{-(l_t - l_\tau)}}}$$

$$\sum_{t=1}^{T} \mathbb{S}(l)_t = 1 \text{ and } 1 > \mathbb{S}(l)_t > 0$$

- The only way to attain $\mathbb{S}(l)_t = 1$ is:
    i. $\|l\| \to \infty$
    ii. $l_t > l_\tau \text{ for all } \tau \neq t$

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^{T} e^{l_\tau}} = \frac{1}{1 + \sum_{\tau \neq t} e^{-(l_t - l_\tau)}}$$

$$\sum_{t=1}^{T} \mathbb{S}(l)_t = 1 \text{ and } 1 > \mathbb{S}(l)_t > 0$$

- The only way to attain $\mathbb{S}(l)_t = 1$ is:
  i. $\|l\| \to \infty$
  ii. $l_t > l_\tau \text{ for all } \tau \neq t$

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^{T} e^{l_\tau}} = \frac{1}{1 + \sum_{\tau \neq t} e^{-(l_t - l_\tau)}}$$

$$\sum_{t=1}^{T} \mathbb{S}(l)_t = 1 \text{ and } 1 > \mathbb{S}(l)_t > 0$$

- The only way to attain $\mathbb{S}(l)_t = 1$ is:
  i. $\|l\| \to \infty$
  ii. $l_t > l_\tau$ for all $\tau \neq t$

- When $l = X^\top W z$ we require $\|l\| = \|X^\top W z\| \leq \|X\|\|W\|\|z\| \to \infty$

# Softmax intuition

- Softmax maps $l = [l_1, \ldots, l_T]$ into a probability distribution

$$\mathbb{S}(l)_t = \frac{e^{l_t}}{\sum_{\tau=1}^T e^{l_\tau}} = \frac{1}{1 + \sum_{\tau \neq t} e^{-(l_t - l_\tau)}}$$

$$\sum_{t=1}^T \mathbb{S}(l)_t = 1 \text{ and } 1 > \mathbb{S}(l)_t > 0$$

- The only way to attain $\mathbb{S}(l)_t = 1$ is:
  i. $\|l\| \to \infty$
  ii. $l_t > l_\tau \text{ for all } \tau \neq t$

- When $l = X^\top W z$ we require $\|l\| = \|X^\top W z\| \leq \|X\| \|W\| \|z\| \to \infty$
- So **necessarily** $\|W\| \to \infty$

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \text{ where } s_t = \mathbb{S}(XWz)_t$$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \text{ where } s_t = \mathbb{S}(XWz)_t$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot (yv^\top x_t)\right)$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \text{ where } s_t = \mathbb{S}(XWz)_t$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \text{ where } \boxed{s_t = \mathbb{S}(XWz)_t}$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \ \text{ where } \boxed{s_t = \mathbb{S}(X\boxed{W}z)_t}$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \ \text{ where } \boxed{s_t = \mathbb{S}(X\boxed{W}z)_t}$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

- For decreasing loss $\ell$, maximize inner sums over probabilities $s_t$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \text{ where } \boxed{s_t = \mathbb{S}(X\boxed{W}z)_t}$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

- For decreasing loss $\ell$, maximize inner sums over probabilities $s_t$

$$s_{\text{opt}} \leftarrow 1$$
$$s_{t \neq \text{opt}} \leftarrow 0$$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \text{ where } \boxed{s_t = \mathbb{S}(X\boxed{W}z)_t}$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

- For decreasing loss $\ell$, maximize inner sums over probabilities $s_t$

$$s_{\text{opt}} \leftarrow 1$$
$$s_{t \neq \text{opt}} \leftarrow 0$$

i. $\|W\| \to \infty$

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z$ for all $\tau \neq \text{opt}$

[Tarzanagh et al.'23b]

# Token selection

- Attention outputs softmax combinations of tokens

$$x^{\text{att}} = \sum_{t=1}^{T} s_t \cdot x_t \ \text{where} \ \boxed{s_t = \mathbb{S}(X\boxed{W}z)_t}$$

- Loss minimizes $\ell(y \cdot v^\top x^{\text{att}}) = \ell\left(\sum_{t=1}^{T} s_t \cdot \underbrace{(yv^\top x_t)}_{\text{scores of tokens}}\right)$

- For decreasing loss $\ell$, maximize inner sums over probabilities $s_t$

$$s_{\text{opt}} \leftarrow 1$$
$$s_{t \neq \text{opt}} \leftarrow 0$$

The optimal softmax choice is to select token with the largest score!

i. $\|W\| \to \infty$
ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z$ for all $\tau \neq \text{opt}$

[Tarzanagh et al.'23b]

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z$ for all $\tau \neq \text{opt}$

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\mathrm{opt}}^\top W z > x_\tau^\top W z$ for all $\tau \neq \mathrm{opt}$

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\mathrm{opt}}^\top W z > x_\tau^\top W z$ for all $\tau \neq \mathrm{opt}$ ➡ Token separation

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z \text{ for all } \tau \neq \text{opt}$ ➡ Token separation

- Potentially, infinitely many directions $\bar{W} (\|\bar{W}\| = 1)$ satisfy

$$x_{\text{opt}}^\top \bar{W} z > x_i^\top \bar{W} z$$

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z \text{ for all } \tau \neq \text{opt}$ ➡ Token separation

- Potentially, infinitely many directions $\bar{W}(\|\bar{W}\| = 1)$ satisfy

$$x_{\text{opt}}^\top \bar{W} z > x_i^\top \bar{W} z$$

But which one do GD and RP select?

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z$ for all $\tau \neq \text{opt}$ ➡ Token separation

- Potentially, infinitely many directions $\bar{W}(\|\bar{W}\| = 1)$ satisfy

$$x_{\text{opt}}^\top \bar{W} z > x_i^\top \bar{W} z$$

$$W^{\text{mm}} = \arg\min_W \|W\|_F \text{ subj. to } (x_{i,\text{opt}_i} - x_{i,t})^\top W z_i \geq 1 \text{ for all } t \neq \text{opt}_i, i = 1, \ldots, n$$

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z \text{ for all } \tau \neq \text{opt}$ ➡ Token separation

- Potentially, infinitely many directions $\bar{W}(\|\bar{W}\| = 1)$ satisfy

$$x_{\text{opt}}^\top \bar{W} z > x_i^\top \bar{W} z$$

$$W^{\text{mm}} = \arg\min_W \|W\|_F \text{ subj. to } (x_{i,\text{opt}_i} - x_{i,t})^\top W z_i \geq 1 \text{ for all } t \neq \text{opt}_i, i = 1, \ldots, n$$

**Theorem 2 (TLTO'23 Regularization Path → Att-SVM)** Suppose optimal indices $(\text{opt}_i)_{i=1}^n$ are unique and (Att-SVM) is feasible. Let $W^{\text{mm}}$ be the unique solution of (Att-SVM) with Frobenius norm. Then

$$\lim_{R \to \infty} \frac{\bar{W}_R}{R} = \frac{W^{\text{mm}}}{\|W^{\text{mm}}\|_F}$$

Regularization path

$$\bar{W}_R = \arg\min_{\|W\|_F \leq R} \mathcal{L}(W)$$

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z \text{ for all } \tau \neq \text{opt}$ ➡ Token separation

- Potentially, infinitely many directions $\bar{W} (\|\bar{W}\| = 1)$ satisfy

$$x_{\text{opt}}^\top \bar{W} z > x_i^\top \bar{W} z$$

Max margin SVM solution

$$W^{\text{mm}} = \arg\min_W \|W\|_F \text{ subj. to } (x_{i,\text{opt}_i} - x_{i,t})^\top W z_i \geq 1 \text{ for all } t \neq \text{opt}_i, i = 1, \ldots, n$$

**Theorem 2 (TLTO'23 Regularization Path → Att-SVM)** Suppose optimal indices $(\text{opt}_i)_{i=1}^n$ are unique and (Att-SVM) is feasible. Let $W^{\text{mm}}$ be the unique solution of (Att-SVM) with Frobenius norm. Then

$$\lim_{R \to \infty} \frac{\bar{W}_R}{R} = \frac{W^{\text{mm}}}{\|W^{\text{mm}}\|_F}$$

Regularization path

$$\bar{W}_R = \arg \min_{\|W\|_F \leq R} \mathcal{L}(W)$$

[Tarzanagh et al.'23b]

# Implicit bias

Optimal loss $\mathcal{L}_*$ can be achieved (asymptotically) if and only if

i. $\|W\| \to \infty$ ➡ Norm divergence

ii. $x_{\text{opt}}^\top W z > x_\tau^\top W z \text{ for all } \tau \neq \text{opt}$ ➡ Token separation

- Potentially, infinitely many directions $\bar{W}(\|\bar{W}\| = 1)$ satisfy

$$x_{\text{opt}}^\top \bar{W} z > x_i^\top \bar{W} z$$

Max margin SVM solution

$$W^{\text{mm}} = \arg\min_{W} \|W\|_F \text{ subj. to } (x_{i,\text{opt}_i} - x_{i,t})^\top W z_i \geq 1 \text{ for all } t \neq \text{opt}_i, i = 1, \ldots, n$$

**Theorem 2 (TLTO'23 Regularization Path → Att-SVM)** Suppose optimal indices $(\text{opt}_i)_{i=1}^n$ are unique and (Att-SVM) is feasible. Let $W^{\text{mm}}$ be the unique solution of (Att-SVM) with Frobenius norm. Then

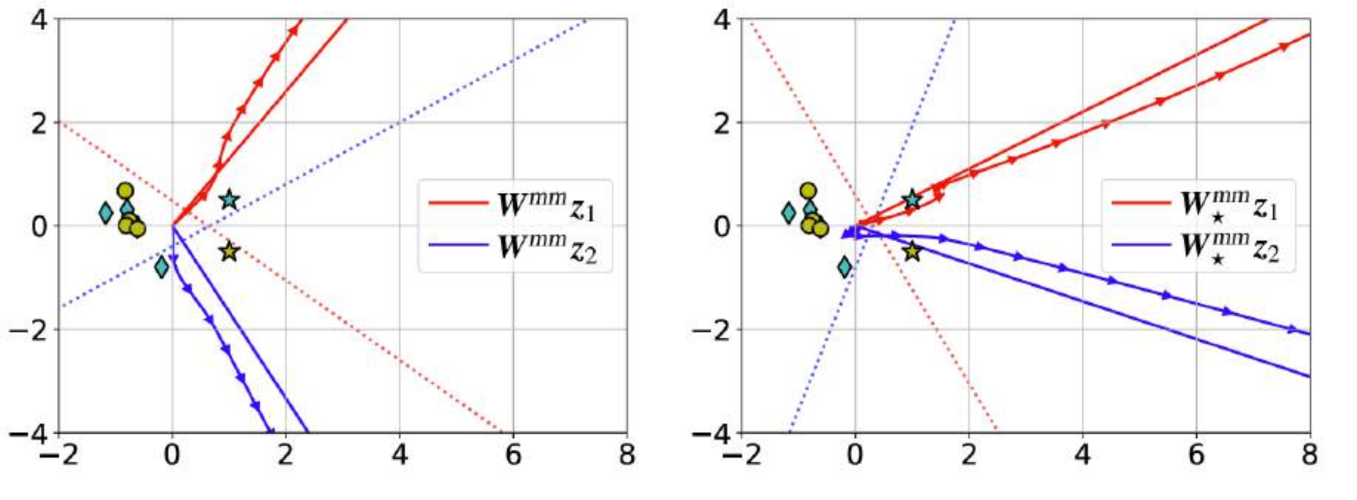
$$\lim_{R \to \infty} \frac{\bar{W}_R}{R} = \frac{W^{\text{mm}}}{\|W^{\text{mm}}\|_F}$$

Weights go to ∞, but the direction converges to SVM solution!

Regularization path

$$\bar{W}_R = \arg\min_{\|W\|_F \leq R} \mathcal{L}(W)$$

# What about GD

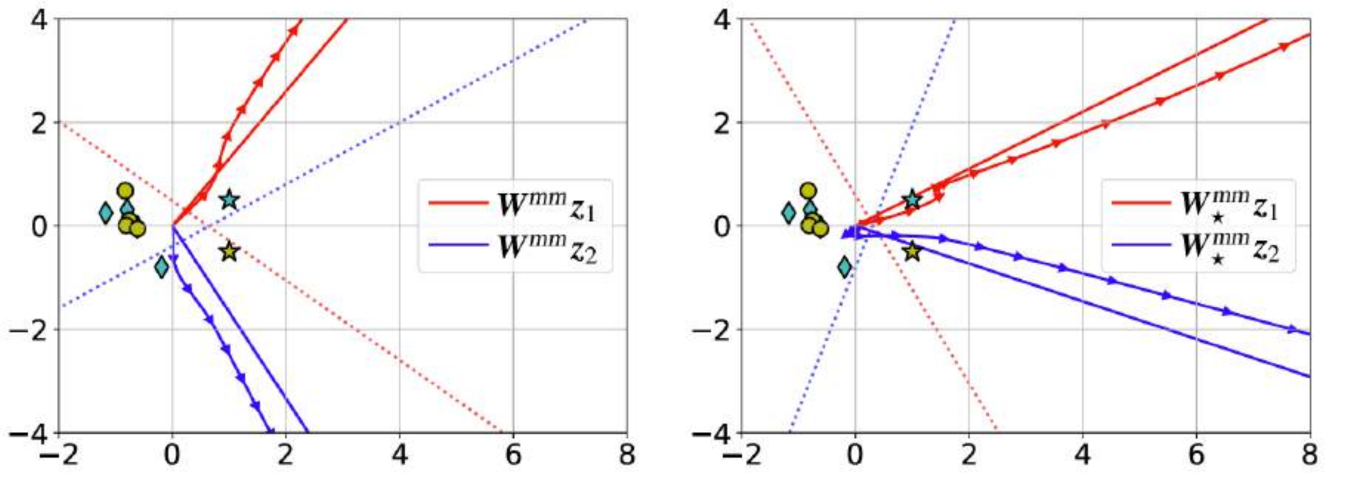



(a) $\boldsymbol{W}$-parameterization (b) $(\boldsymbol{K}, \boldsymbol{Q})$-parameterization

Arrows represent GD trajectory
Solid lines are SVM solution
Dotted lines represent separating hyperplanes

[Tarzanagh et al.’23b]

# What about GD



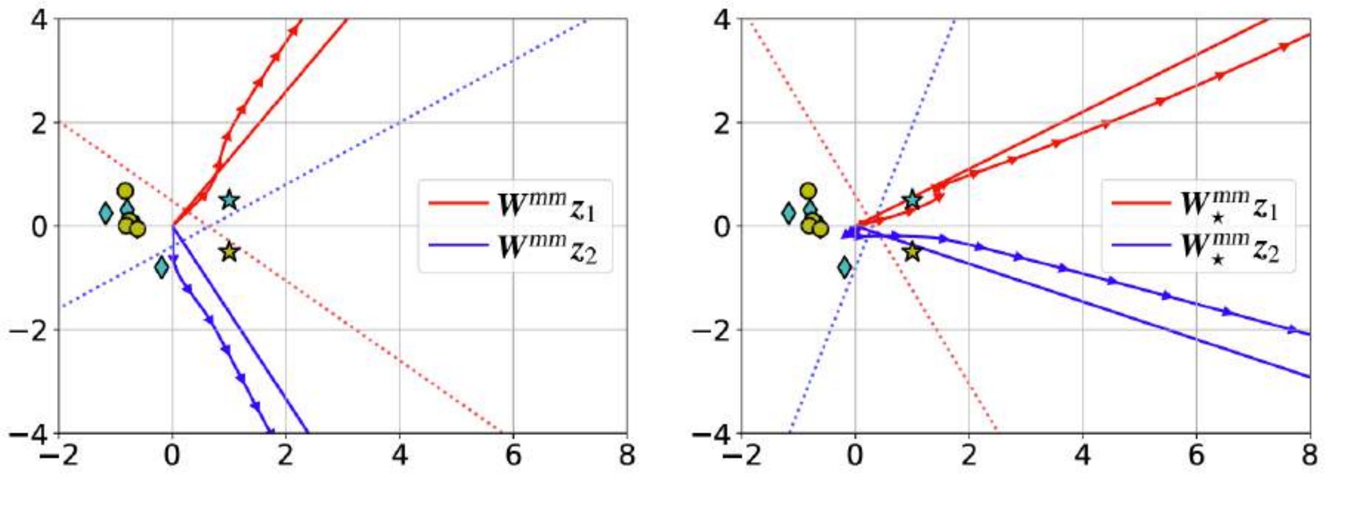

(a) $\boldsymbol{W}$-parameterization (b) $(\boldsymbol{K}, \boldsymbol{Q})$-parameterization

Weights diverge

Arrows represent GD trajectory
Solid lines are SVM solution
Dotted lines represent separating hyperplanes

[Tarzanagh et al.'23b]

# What about GD




(a) $W$-parameterization (b) $(K, Q)$-parameterization

Weights diverge
Converge to max-margin solution

Arrows represent GD trajectory
Solid lines are SVM solution
Dotted lines represent separating hyperplanes

[Tarzanagh et al.'23b]

# Initialization dependence

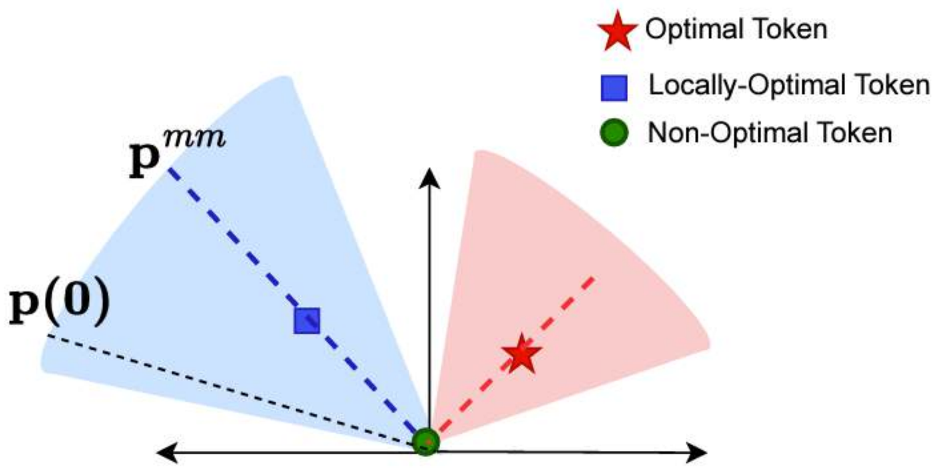



Figure 2: Gradient descent initialization $\boldsymbol{p}(0)$ inside the cone containing the locally-optimal solution $\boldsymbol{p}^{mm}$.

[Tarzanagh et al.'23a]

# Better models

What if the decoder is nonlinear? Or there is more than two classes?

[Tarzanagh et al.’23b]

# Better models

What if the decoder is nonlinear? Or there is more than two classes?

**No unique** optimal token per sequence

- $\mathcal{R}_i$ the set of optimal tokens per sequence $i$

[Tarzanagh et al.'23b]

# Better models

What if the decoder is nonlinear? Or there is more than two classes?

**No unique** optimal token per sequence

- $\mathcal{R}_i$ the set of optimal tokens per sequence $i$

$$\boxed{W^{\mathrm{mm}} = \arg\min_{W} \|W\| \text{ subj. to } \begin{cases} (x_{i,t} - x_{i,\tau})^\top W z_i \geq 1 & t \in \mathcal{R}_i, \tau \notin \mathcal{R}_i, i = 1, \ldots, n \\ (x_{i,t} - x_{i,t'})^\top W z_i = 0 & t, t' \in \mathcal{R}_i, i = 1, \ldots, n \end{cases}}$$

[Tarzanagh et al.'23b]

# Better models

What if the decoder is nonlinear? Or there is more than two classes?

**No unique** optimal token per sequence

- $\mathcal{R}_i$ the set of optimal tokens per sequence $i$

$$W^{\mathrm{mm}} = \arg\min_{W} \|W\| \text{ subj. to } \begin{cases} (x_{i,t} - x_{i,\tau})^\top W z_i \geq 1 & t \in \mathcal{R}_i, \tau \notin \mathcal{R}_i, i = 1, \ldots, n \\ (x_{i,t} - x_{i,t'})^\top W z_i = 0 & t, t' \in \mathcal{R}_i, i = 1, \ldots, n \end{cases}$$

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$: Finite component weights relevant tokens

[Tarzanagh et al.'23b]

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$ : Finite component weights relevant tokens

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$ : Finite component weights relevant tokens

$W^{\mathrm{fin}}$ assigns weights to **non-zero** tokens

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$: Finite component weights relevant tokens

$W^{\mathrm{fin}}$ assigns weights to **non-zero** tokens

$$\frac{e^{x_{i,t}^\top W^{\mathrm{fin}} z_i}}{e^{x_{i,\tau}^\top W^{\mathrm{fin}} z_i}} = e^{(x_{i,t}-x_{i,\tau})^\top W^{\mathrm{fin}} z_i} = \frac{s_{i,t}^*}{s_{i,\tau}^*}$$

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$ : Finite component weights relevant tokens

$W^{\mathrm{fin}}$ assigns weights to **non-zero** tokens

$$\frac{e^{x_{i,t}^\top W^{\mathrm{fin}} z_i}}{e^{x_{i,\tau}^\top W^{\mathrm{fin}} z_i}} = e^{(x_{i,t} - x_{i,\tau})^\top W^{\mathrm{fin}} z_i} = \frac{s^*_{i,t}}{s^*_{i,\tau}}$$

$$(x_{i,t} - x_{i,\tau})^\top W^{\mathrm{fin}} z_i = \log\left(\frac{s^*_{i,t}}{s^*_{i,\tau}}\right) \text{ for all } t, \tau \in \mathcal{O}_i$$

$\mathcal{O}_i$ is the set of tokens selected by the attention mechanism

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$ : Finite component weights relevant tokens

$W^{\mathrm{fin}}$ assigns weights to **non-zero** tokens

$$\frac{e^{x_{i,t}^\top W^{\mathrm{fin}} z_i}}{e^{x_{i,\tau}^\top W^{\mathrm{fin}} z_i}} = e^{(x_{i,t}-x_{i,\tau})^\top W^{\mathrm{fin}} z_i} = \frac{s^*_{i,t}}{s^*_{i,\tau}}$$

$$(x_{i,t} - x_{i,\tau})^\top W^{\mathrm{fin}} z_i = \log\left(\frac{s^*_{i,t}}{s^*_{i,\tau}}\right) \quad \text{for all } t, \tau \in \mathcal{O}_i$$

$W^{\mathrm{mm}}$ **selects** and **suppresses** tokens

$\mathcal{O}_i$ is the set of tokens selected by the attention mechanism

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$ : Finite component weights relevant tokens

$W^{\mathrm{fin}}$ assigns weights to **non-zero** tokens

$$\frac{e^{x_{i,t}^\top W^{\mathrm{fin}} z_i}}{e^{x_{i,\tau}^\top W^{\mathrm{fin}} z_i}} = e^{(x_{i,t} - x_{i,\tau})^\top W^{\mathrm{fin}} z_i} = \frac{s^*_{i,t}}{s^*_{i,\tau}}$$

$$(x_{i,t} - x_{i,\tau})^\top W^{\mathrm{fin}} z_i = \log\left(\frac{s^*_{i,t}}{s^*_{i,\tau}}\right) \text{ for all } t, \tau \in \mathcal{O}_i$$

$W^{\mathrm{mm}}$ **selects** and **suppresses** tokens

$$(x_{i,t} - x_{i,\tau})^\top W z_i = 0 \text{ for all } t, \tau \in \mathcal{O}_i$$

$\mathcal{O}_i$ is the set of tokens selected by the attention mechanism

[Tarzanagh et al.'23b]

# Better models

Conjecture: $W_k \approx W^{fin} + C_k W^{\mathrm{mm}}$ with $C_k \to \infty$

- $W^{\mathrm{mm}}$ : Directional component that diverges to infinity
- $W^{\mathrm{fin}}$ : Finite component weights relevant tokens

$W^{\mathrm{fin}}$ assigns weights to **non-zero** tokens

$$\frac{e^{x_{i,t}^\top W^{\mathrm{fin}} z_i}}{e^{x_{i,\tau}^\top W^{\mathrm{fin}} z_i}} = e^{(x_{i,t}-x_{i,\tau})^\top W^{\mathrm{fin}} z_i} = \frac{s^*_{i,t}}{s^*_{i,\tau}}$$

$$(x_{i,t} - x_{i,\tau})^\top W^{\mathrm{fin}} z_i = \log\left(\frac{s^*_{i,t}}{s^*_{i,\tau}}\right) \text{ for all } t, \tau \in \mathcal{O}_i$$

$W^{\mathrm{mm}}$ **selects** and **suppresses** tokens

$$(x_{i,t} - x_{i,\tau})^\top W z_i = 0 \text{ for all } t, \tau \in \mathcal{O}_i$$

$$(x_{i,t} - x_{i,\tau})^\top W z_i \geq 1 \text{ for all } t \in \mathcal{O}_i, \tau \notin \mathcal{O}_i$$

$\mathcal{O}_i$ is the set of tokens selected by the attention mechanism

# Faster optimizers

[Vasudeva et al.'24]

# Faster optimizers

Theorem (VDT24) (Informal) Assume nearly-orthogonal tokens and sub-optimal tokens. $x_{i,t}$ have equal scores $v^\top x_{i,t} = v^\top x_{i,t'}, \forall t, t' \neq \mathrm{opt}_i$. Then, NGD and Polyak-step both converge globally to Att-SVM with fast rate:

$$\left\langle \frac{W_k}{\|W_k\|}, \frac{W^{\mathrm{mm}}}{\|W^{\mathrm{mm}}\|} \right\rangle \geq 1 - C\frac{\log^2(k)}{k}$$

[Vasudeva et al.'24]

# Faster optimizers

Theorem (VDT24) (Informal) Assume nearly-orthogonal tokens and sub-optimal tokens. $x_{i,t}$ have equal scores $v^\top x_{i,t} = v^\top x_{i,t'}, \forall t, t' \neq \mathrm{opt}_i$. Then, NGD and Polyak-step both converge globally to Att-SVM with fast rate:

$$\left\langle \frac{W_k}{\|W_k\|}, \frac{W^{\mathrm{mm}}}{\|W^{\mathrm{mm}}\|} \right\rangle \geq 1 - C\frac{\log^2(k)}{k}$$

- Compare to $\mathcal{O}\left(\frac{1}{\log(k)}\right)$ of GD

[Vasudeva et al.'24]

# Optimization summary

# Optimization summary

1. Attention maps are **sparse** as it is often optimal to select only few tokens

# Optimization summary

1. Attention maps are **sparse** as it is often optimal to select only few tokens
2. When few tokens are selected attention becomes **equivalent to an SVM**

# Optimization summary

1. Attention maps are **sparse** as it is often optimal to select only few tokens
2. When few tokens are selected attention becomes **equivalent to an SVM**
3. Gradient descent converges to the **max-margin** solution of this SVM

# Optimization summary

1. Attention maps are **sparse** as it is often optimal to select only few tokens
2. When few tokens are selected attention becomes **equivalent to an SVM**
3. Gradient descent converges to the **max-margin** solution of this SVM
4. Initialization can cause us to get stuck in **local optima**

# Optimization summary

1. Attention maps are **sparse** as it is often optimal to select only few tokens
2. When few tokens are selected attention becomes **equivalent to an SVM**
3. Gradient descent converges to the **max-margin** solution of this SVM
4. Initialization can cause us to get stuck in **local optima**
5. Closer approximations to true attention are conjectured to converge to similar SVM

# Optimization summary

1. Attention maps are **sparse** as it is often optimal to select only few tokens
2. When few tokens are selected attention becomes **equivalent to an SVM**
3. Gradient descent converges to the **max-margin** solution of this SVM
4. Initialization can cause us to get stuck in **local optima**
5. Closer approximations to true attention are conjectured to converge to similar SVM
6. Practical optimizers converge to the same solution at a **faster rate**

# Overview

- Transformers primer
- Optimisation
- **Approximation**
- Memorisation
- In-context learning

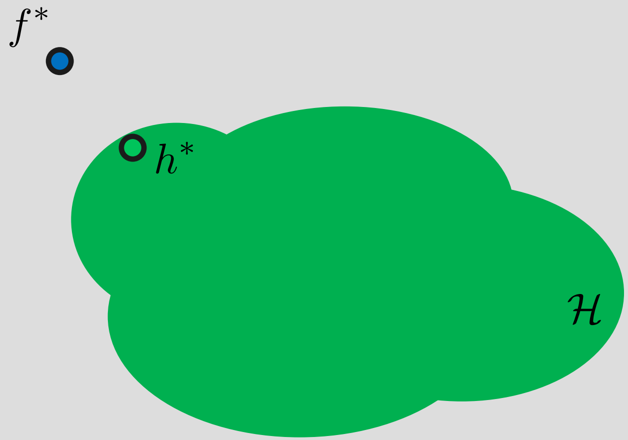
Approximus prime?

# Goal of approximation

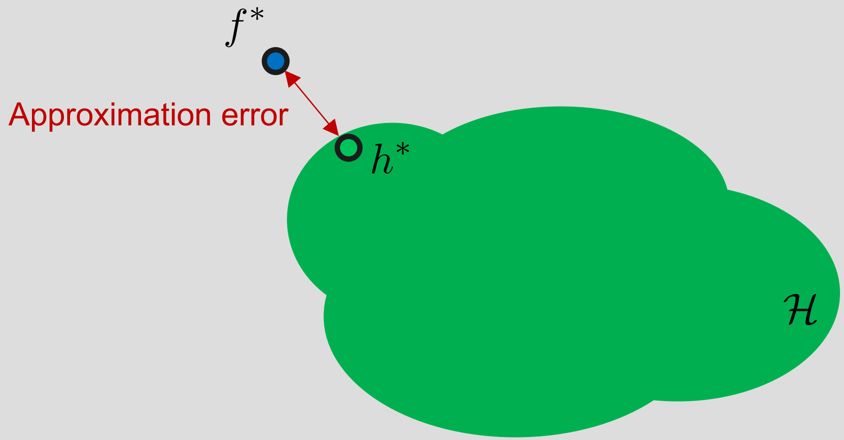
# Goal of approximation

# Goal of approximation

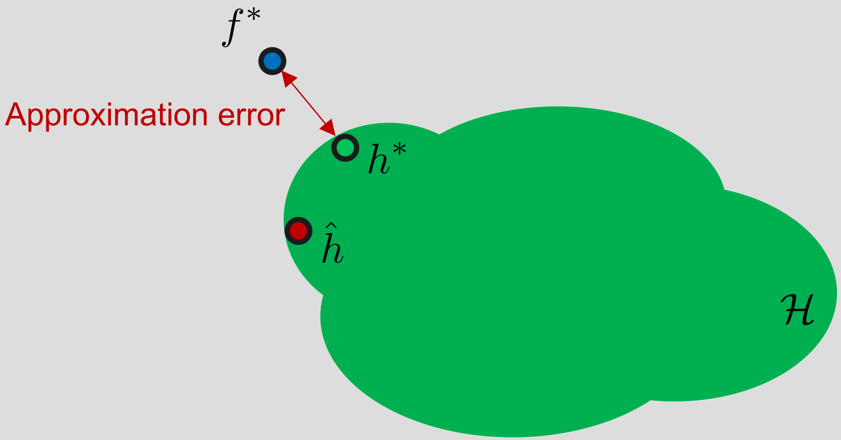
# Goal of approximation



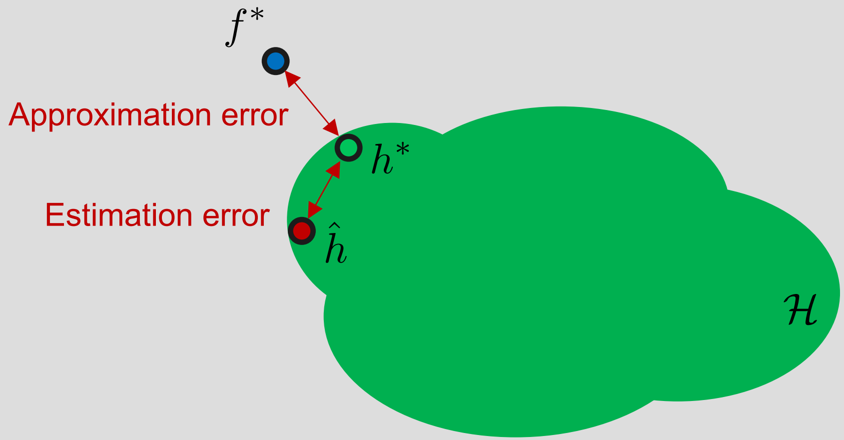
# Goal of approximation



# Goal of approximation

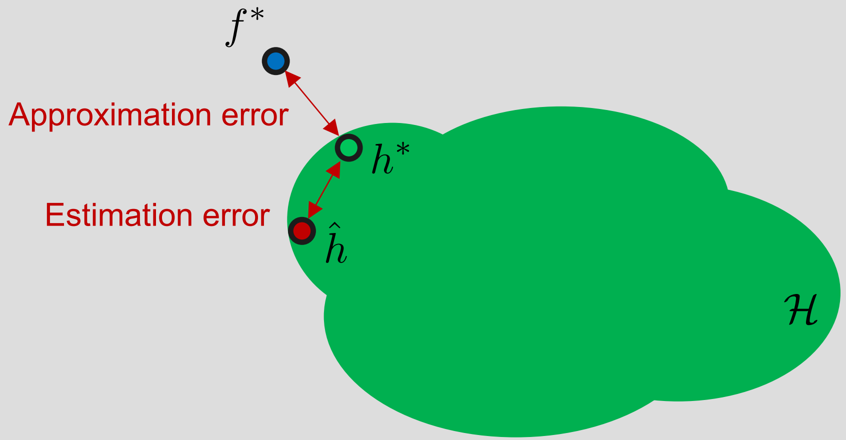


# Goal of approximation



# Goal of approximation

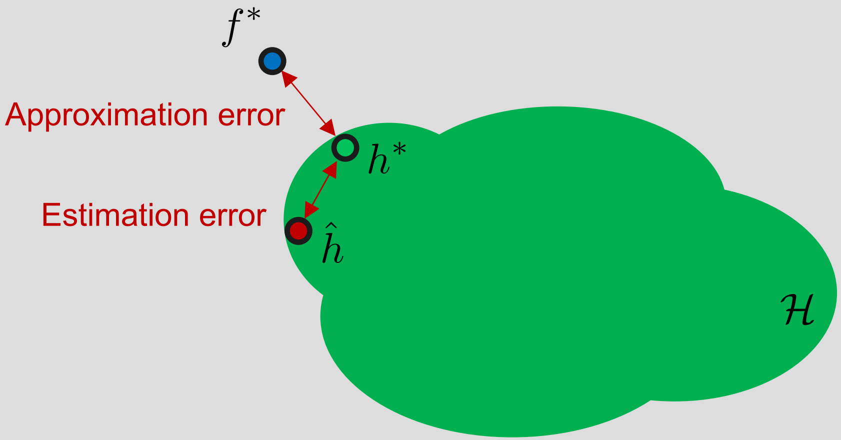
- For an unknown function $f^* \in \mathcal{C} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ that captures the process of interest



# Goal of approximation

- For an unknown function $f^* \in \mathcal{C} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ that captures the process of interest

- Goal: recover $f^*$ from finite observations $\mathcal{S}_n = \{(x_i, f^*(x_i) = y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$

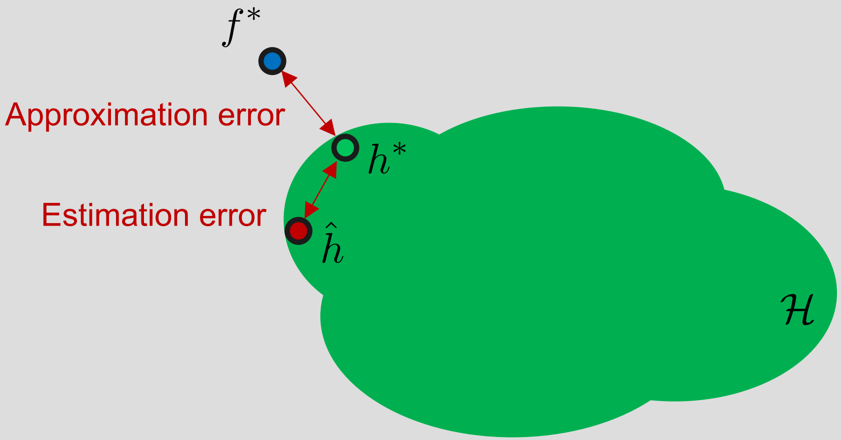


# Goal of approximation

- For an unknown function $f^* \in \mathcal{C} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ that captures the process of interest

- Goal: recover $f^*$ from finite observations $\mathcal{S}_n = \{(x_i, f^*(x_i) = y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$

- Solution: Approximate $f^*$ by searching for a "close" function in $\mathcal{H}$ based on finite observations

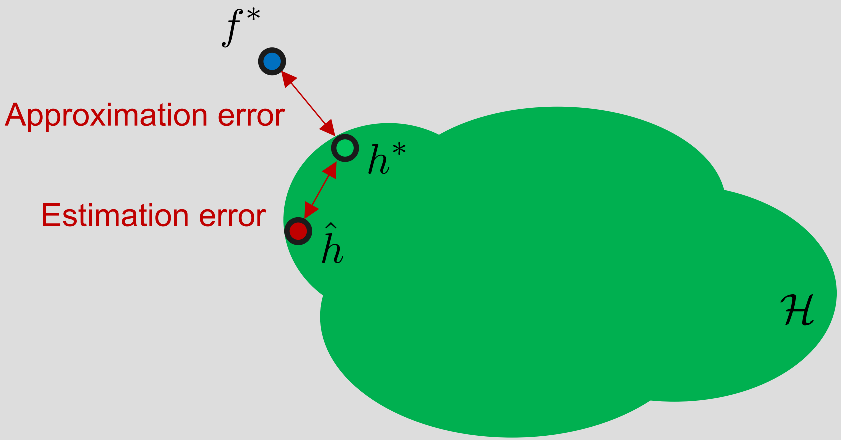


# Goal of approximation

- For an unknown function $f^* \in \mathcal{C} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ that captures the process of interest
- Goal: recover $f^*$ from finite observations $\mathcal{S}_n = \{(x_i, f^*(x_i) = y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$
- Solution: Approximate $f^*$ by searching for a "close" function in $\mathcal{H}$ based on finite observations

Universal approximation

Given $\epsilon > 0$ and any $f^* \in \mathcal{C}$, there exists $h \in \mathcal{H}$ such that the approximation error $D(h, f^*) \leq \epsilon$



# Transformers are universal approximators

When $\mathcal{C}$ represents all continuous and permutation equivariant seq-to-seq functions with compact support and $\mathcal{H}$ represents all Transformer networks

# Transformers are universal approximators

When $\mathcal{C}$ represents all continuous and permutation equivariant seq-to-seq functions with compact support and $\mathcal{H}$ represents all Transformer networks

**Theorem 2 [Yun et al.'19]:** Transformer networks with constant width and large enough depth can universally approximate $\mathcal{C}$. In particular, for any given $\epsilon > 0$ and $f^* \in \mathcal{C}$, there exists an $h \in \mathcal{H}$ satisfying

$$D_p(h, f^*) := \left( \int \|h(X) - f^*(X)\|_p^p \, dX \right)^{1/p} \leq \epsilon$$

# Transformers are universal approximators

When $\mathcal{C}$ represents all continuous and ~~permutation equivariant~~ seq-to-seq functions with compact support and $\mathcal{H}$ represents all Transformer networks with positional encodings

**Theorem 3 [Yun et al.'19]:** Transformer networks with constant width and large enough depth can universally approximate $\mathcal{C}$. In particular, for any given $\epsilon > 0$ and $f^* \in \mathcal{C}$, there exists an $h \in \mathcal{H}$ satisfying

$$D_p(h, f^*) := \left( \int \|h(X) - f^*(X)\|_p^p \, dX \right)^{1/p} \leq \epsilon$$

# Transformers are universal approximators

When $\mathcal{C}$ represents all continuous and permutation equivariant seq-to-seq functions with compact support and $\mathcal{H}$ represents Transformer networks with the following form:

$$\mathcal{H} = \{\mathrm{FF}_2 \circ \mathrm{Attn} \circ \mathrm{FF}_1 \in \mathbb{R}^{d \times T} \to \mathbb{R}^{d \times T}\}$$

# Transformers are universal approximators

When $\mathcal{C}$ represents all continuous and permutation equivariant seq-to-seq functions with compact support and $\mathcal{H}$ represents Transformer networks with the following form:

$$\mathcal{H} = \{\mathrm{FF}_2 \circ \mathrm{Attn} \circ \mathrm{FF}_1 \in \mathbb{R}^{d\times T} \to \mathbb{R}^{d\times T}\}$$

**Theorem [Kajitsuka & Sato'23]:** For any given $\epsilon > 0$ and $f^* \in \mathcal{C}$, there exists an $h \in \mathcal{H}$ With **one layer** and **single-head attention** such that the following holds:

$$D_p(h, f^*) := \left(\int \|h(X) - f^*(X)\|_p^p \, dX\right)^{1/p} \leq \epsilon$$

# Sparse transformers

# Sparse transformers

- Vanilla self-attention has quadratic computational cost in input sequence length $T$
  - Each attention head implements $\mathcal{O}(T^2)$ pairwise interactions

# Sparse transformers

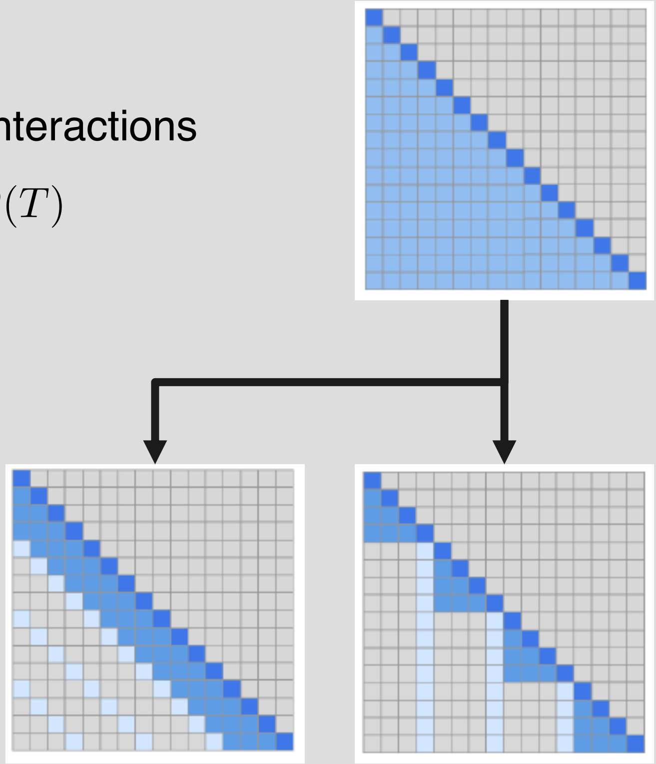
- Vanilla self-attention has quadratic computational cost in input sequence length $T$
  - Each attention head implements $\mathcal{O}(T^2)$ pairwise interactions
- **Sparse transformers** reduce this to $\mathcal{O}(T^{1.5})$ or even $\mathcal{O}(T)$

[Child et al., 2019]

# Sparse transformers

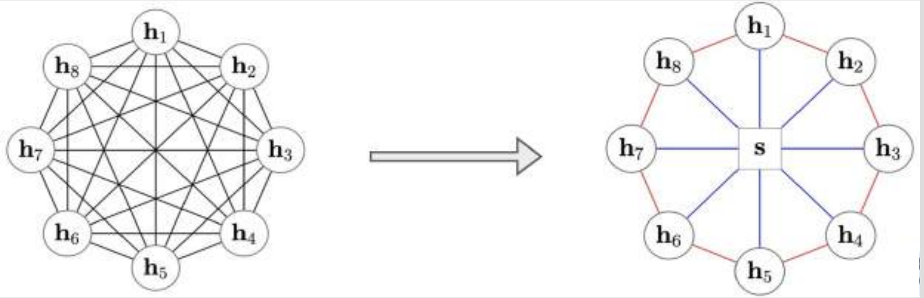
- Vanilla self-attention has quadratic computational cost in input sequence length $T$
  - Each attention head implements $\mathcal{O}(T^2)$ pairwise interactions
- **Sparse transformers** reduce this to $\mathcal{O}(T^{1.5})$ or even $\mathcal{O}(T)$

Transformer



Strided

Fixed




Star-Transformer [Guo et al.’19]

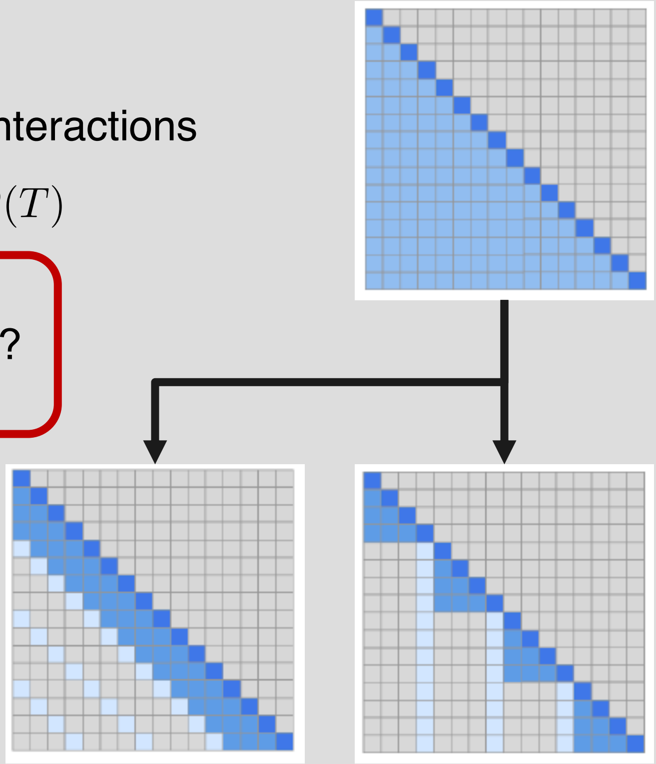
[Child et al., 2019]

# Sparse transformers

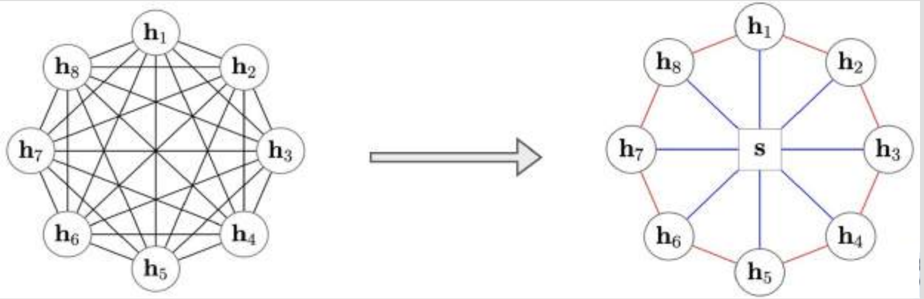
- Vanilla self-attention has quadratic computational cost in input sequence length $T$
  - Each attention head implements $\mathcal{O}(T^2)$ pairwise interactions
- **Sparse transformers** reduce this to $\mathcal{O}(T^{1.5})$ or even $\mathcal{O}(T)$

Can sparse transformers be universal approximators?
- If so, what are the requirements on the sparsity pattern?
- How sparse can attention be?

Transformer



Strided    Fixed




Star-Transformer [Guo et al.'19]

# Sparse transformers

# Sparse transformers

- Conditions on sparsity patterns

# Sparse transformers

- **Conditions on sparsity patterns**
    - Every token always attends to itself

# Sparse transformers

- **Conditions on sparsity patterns**
    - Every token always attends to itself
    - There exists a chain of direct connections that covers all tokens

# Sparse transformers

- **Conditions on sparsity patterns**
    - Every token always attends to itself
    - There exists a chain of direct connections that covers all tokens
    - With enough layers every token can attend to all other tokens

# Sparse transformers

- **Conditions on sparsity patterns**
  - Every token always attends to itself
  - There exists a chain of direct connections that covers all tokens
  - With enough layers every token can attend to all other tokens

**Theorem (informal) (Yun et al.'20).** Given unbounded depth, any sparse Transformer satisfying the above conditions is a universal approximator of sequence-to-sequence functions

# Sparse transformers

- **Conditions on sparsity patterns**
  - Every token always attends to itself
  - There exists a chain of direct connections that covers all tokens
  - With enough layers every token can attend to all other tokens

**Theorem (informal) (Yun et al.'20).** Given unbounded depth, any sparse Transformer satisfying the above conditions is a universal approximator of sequence-to-sequence functions

**Corollary.** There are sparse Transformers with $\mathcal{O}(T)$ connections per attention layer that are universal approximators ([Guo et al.'19, Beltagy et al.'20, Zaheer et al.'20])

# Approximation summary

# Approximation summary

1. Transformers are **universal approximators**

# Approximation summary

1. Transformers are **universal approximators**

2. **Sparse transformers** can also be universal approximators

# Overview

- Transformers primer
- Optimisation
- Approximation
- **Memorisation**
- In-context learning

# Motivation

# Motivation

- Recently, increasing model size has been the driving factor to realize performance gains
  - GPT-3 with 175B parameters [Brown et al.'20 ], PALM with 540B parameters [Chowdhery et al.’22].
  - Even trillion parameter models (e.g., [Fedus et al.’22 ], [Du et al.’22 ])

# Motivation

- Recently, increasing model size has been the driving factor to realize performance gains
  - GPT-3 with 175B parameters [Brown et al.'20 ], PALM with 540B parameters [Chowdhery et al.’22].
  - Even trillion parameter models (e.g., [Fedus et al.’22 ], [Du et al.’22 ])
- Very likely that these giant models utilize their parameters to memorize various patterns encountered during training
  - Closed-book QA: language models as knowledge bases [Petroni et al.'19, Roberts et al.’20]

# Motivation

- Recently, increasing model size has been the driving factor to realize performance gains
  - GPT-3 with 175B parameters [Brown et al.'20 ], PALM with 540B parameters [Chowdhery et al.'22].
  - Even trillion parameter models (e.g., [Fedus et al.'22 ], [Du et al.'22 ])
- Very likely that these giant models utilize their parameters to memorize various patterns encountered during training
  - Closed-book QA: language models as knowledge bases [Petroni et al.'19, Roberts et al.'20]
- Critical to demystify the memorization mechanism in giant models
  - Helpful to design new efficient architectures, learning strategies, and inference methods.
  - E.g., memory-augmented NLP models

# Motivation

- Recently, increasing model size has been the driving factor to realize performance gains
  - GPT-3 with 175B parameters [Brown et al.'20 ], PALM with 540B parameters [Chowdhery et al.’22].
  - Even trillion parameter models (e.g., [Fedus et al.’22 ], [Du et al.’22 ])
- Very likely that these giant models utilize their parameters to memorize various patterns encountered during training
  - Closed-book QA: language models as knowledge bases [Petroni et al.'19, Roberts et al.’20]
- Critical to demystify the memorization mechanism in giant models
  - Helpful to design new efficient architectures, learning strategies, and inference methods.
  - E.g., memory-augmented NLP models
- **Hopfield networks:** A model for memorization via neural networks
  - Can shed light on memorization mechanism in many existing architectures, e.g., Transformers
  - Can provide a framework to design/analyze iterative model architectures

# Hopfield networks

[Blog]
[Hopfield ’82]

[Blog]
[Hopfield '82]

# Hopfield networks

- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$

[Blog]
[Hopfield '82]

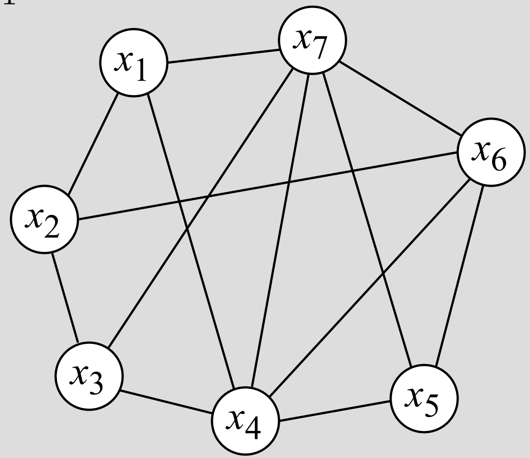
# Hopfield networks

- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$
- In classical Hopfield networks these are polar $x_i \in \{-1, 1\}^d$

[Blog]
[Hopfield '82]

# Hopfield networks

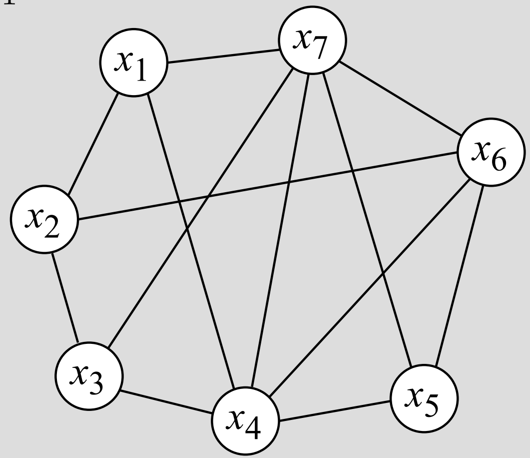
- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$
- In classical Hopfield networks these are polar $x_i \in \{-1, 1\}^d$
- Build a graph with weights matrix $W = \sum_{i=1}^{N} x_i x_i^\top$



[Blog]
[Hopfield '82]

# Hopfield networks

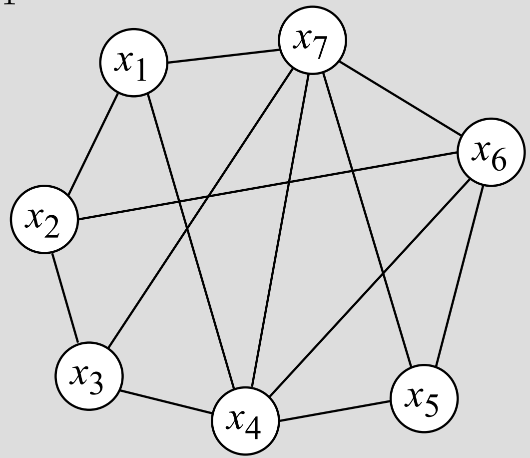
- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$
- In classical Hopfield networks these are polar $x_i \in \{-1, 1\}^d$
- Build a graph with weights matrix $W = \sum_{i=1}^{N} x_i x_i^\top$
- The **asynchronous update rule** with state pattern $\xi$ is



[Blog]
[Hopfield '82]

# Hopfield networks

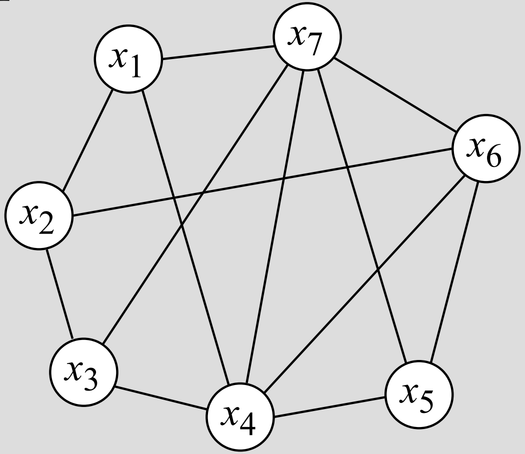
- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$
- In classical Hopfield networks these are polar $x_i \in \{-1, 1\}^d$
- Build a graph with weights matrix $W = \sum_{i=1}^{N} x_i x_i^\top$
- The **asynchronous update rule** with state pattern $\xi$ is
    - For each element one at a time $\xi_i^{t+1} = \mathrm{sgn}(W\xi^t - b)_i$



[Blog]
[Hopfield '82]

# Hopfield networks

- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$
- In classical Hopfield networks these are polar $x_i \in \{-1, 1\}^d$
- Build a graph with weights matrix $W = \sum_{i=1}^{N} x_i x_i^\top$
- The **asynchronous update rule** with state pattern $\xi$ is
  - For each element one at a time $\xi_i^{t+1} = \mathrm{sgn}(W\xi^t - b)_i$
  - Until convergence, all stored patters are fixed points $x_i = \mathrm{sgn}(Wx - b)_i$

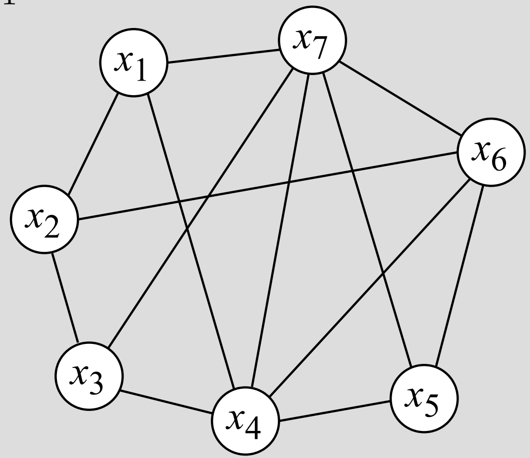


[Blog]
[Hopfield '82]

# Hopfield networks

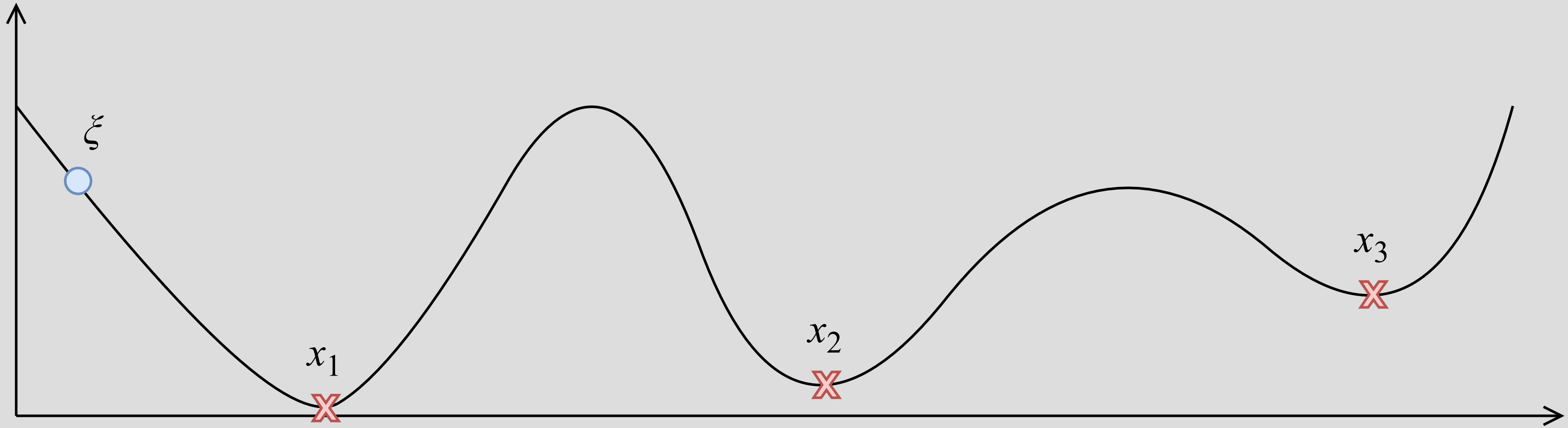
- We want to be able to store and reproduce N patterns $\{x_i\}_{i=1}^N$
- In classical Hopfield networks these are polar $x_i \in \{-1, 1\}^d$
- Build a graph with weights matrix $W = \sum_{i=1}^{N} x_i x_i^\top$
- The **asynchronous update rule** with state pattern $\xi$ is
    - For each element one at a time $\xi_i^{t+1} = \mathrm{sgn}(W\xi^t - b)_i$
    - Until convergence, all stored patters are fixed points $x_i = \mathrm{sgn}(Wx - b)_i$
- We are minimizing an energy function

$$E(\xi) = -\frac{1}{2}\xi^\top W \xi + \xi^\top b = -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\xi_i\xi_j + \sum_{i=1}^{d} b_i \xi_i$$



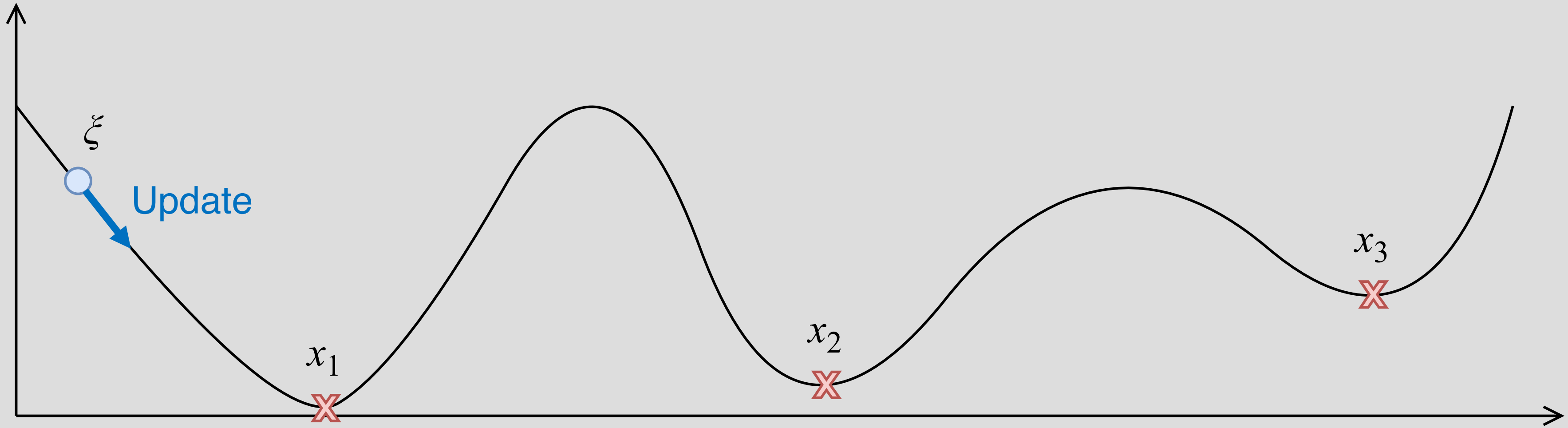
# Energy function

$$E(\xi) = -\frac{1}{2}\xi^\top W \xi + \xi^\top b = -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\xi_i\xi_j + \sum_{i=1}^{d} b_i\xi_i$$



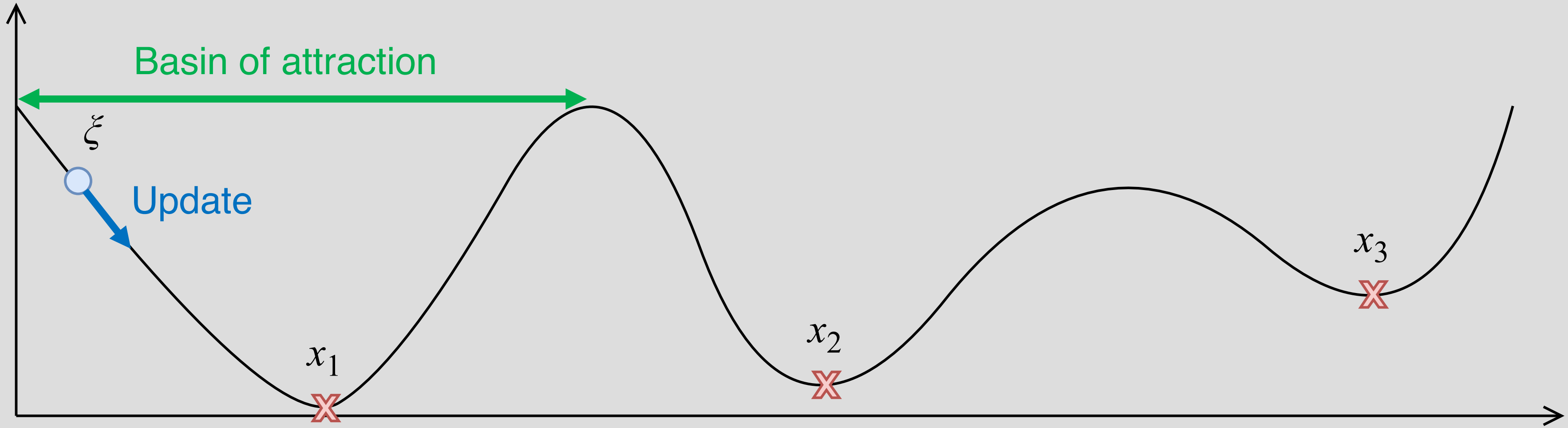
# Energy function

$$E(\xi) = -\frac{1}{2}\xi^\top W \xi + \xi^\top b = -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\xi_i\xi_j + \sum_{i=1}^{d} b_i\xi_i$$



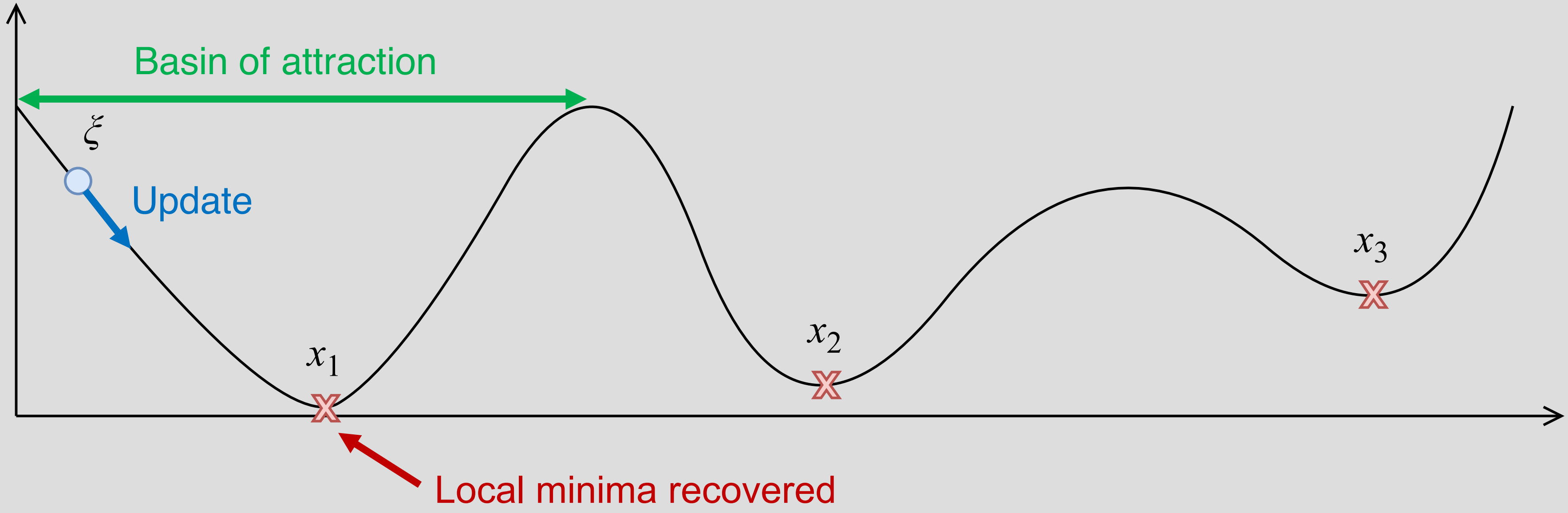
# Energy function

$$E(\xi) = -\frac{1}{2}\xi^\top W \xi + \xi^\top b = -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\xi_i\xi_j + \sum_{i=1}^{d} b_i\xi_i$$



# Energy function

$$E(\xi) = -\frac{1}{2}\xi^\top W \xi + \xi^\top b = -\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\xi_i\xi_j + \sum_{i=1}^{d} b_i\xi_i$$



# Example

[Blog]

# Example

$x_{\text{Homer}} \in \{-1, 1\}^{64}$

[Blog]

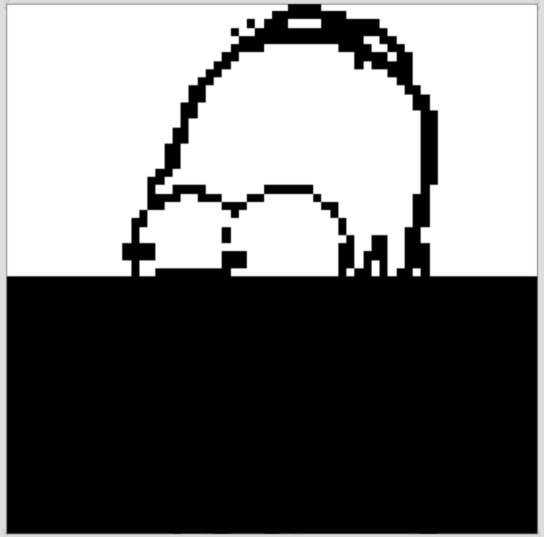
# Example

$$x_{\text{Homer}} \in \{-1, 1\}^{64}$$

$$W = x_{\text{Homer}} x_{\text{Homer}}^{\top}$$

[Blog]

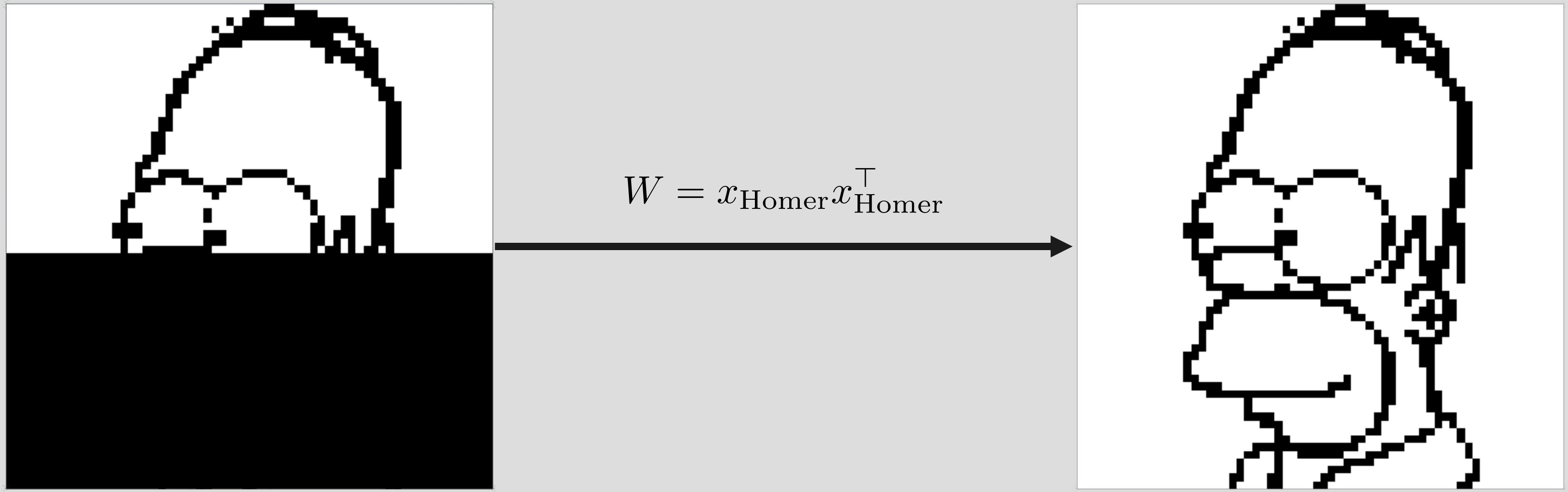
# Example



$x_{\text{Homer}} \in \{-1, 1\}^{64}$

$$W = x_{\text{Homer}} x_{\text{Homer}}^\top$$

# Example

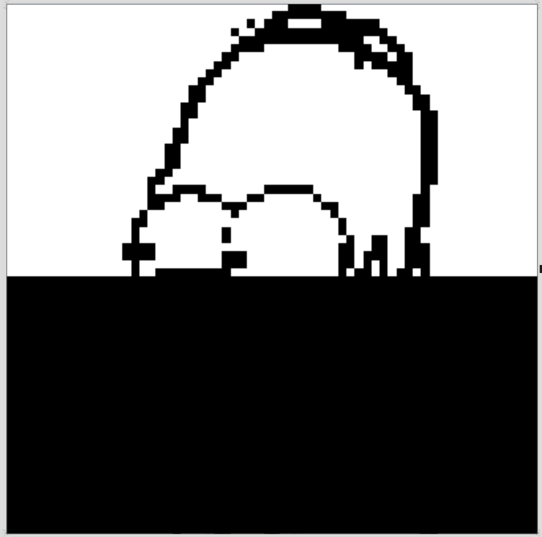


$$W = x_{\text{Homer}} x_{\text{Homer}}^\top$$

$$x_{\text{Homer}} \in \{-1, 1\}^{64}$$

[Blog]

# Example



$x_{\text{Homer}} \in \{-1, 1\}^{64}$

$W = x_{\text{Homer}} x_{\text{Homer}}^{\top}$

$$C \approx \frac{d}{2\log(d)} \approx 0.14d$$

# Modern Hopfield networks

# Modern Hopfield networks

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$

# Modern Hopfield networks

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

# Modern Hopfield networks

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

# Modern Hopfield networks

- New energy function [Krotov, Hopfield’16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

# Modern Hopfield networks

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} (x_i^\top \xi)^a$$

# Modern Hopfield networks

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} (x_i^\top \xi)^a$$

**Capacity:**

$$C \approx= \frac{1}{2(2a-3)!!} \frac{d^{a-1}}{\log(d)}$$

# Modern Hopfield networks

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} (x_i^\top \xi)^a$$

**Capacity:**

$$C \approx= \frac{1}{2(2a-3)!!} \frac{d^{a-1}}{\log(d)}$$

**Interaction function:**

$$F(z) = \exp(z)$$

# Modern Hopfield networks

$$\text{lse}(\beta, z) = \beta^{-1} \log \left( \sum_{l=1}^{N} \exp(\beta z_l) \right)$$

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \text{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} (x_i^\top \xi)^a$$

**Capacity:**

$$C \approx= \frac{1}{2(2a-3)!!} \frac{d^{a-1}}{\log(d)}$$

**Interaction function:**

$$F(z) = \exp(z)$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} \exp(x_i^\top \xi) = -\exp(\text{lse}(1, X^\top \xi))$$

# Modern Hopfield networks

$$\mathrm{lse}(\beta, z) = \beta^{-1} \log \left( \sum_{l=1}^{N} \exp(\beta z_l) \right)$$

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \mathrm{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} (x_i^\top \xi)^a$$

**Capacity:**

$$C \approx= \frac{1}{2(2a-3)!!} \frac{d^{a-1}}{\log(d)}$$

**Interaction function:**

$$F(z) = \exp(z)$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} \exp(x_i^\top \xi) = -\exp(\mathrm{lse}(1, X^\top \xi))$$

**Capacity:**

$$C \approx 2^{d/2}$$

# Modern Hopfield networks

$$\mathrm{lse}(\boxed{\beta}, z) = \beta^{-1} \log \left( \sum_{l=1}^{N} \exp(\beta z_l) \right)$$

- New energy function [Krotov, Hopfield'16] $E(\xi) = -\sum_{i=1}^{N} F(x_i^\top \xi)$
- Update rule $\xi_l^{\text{new}} = \mathrm{sgn}[-E(\xi^{(l^+)}) + E(\xi^{(l^-)})]$

Difference in energy with component $l$ flipped. Update to decrease energy

**Interaction function:**

$$F(z) = z^a$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} (x_i^\top \xi)^a$$

**Capacity:**

$$C \approx= \frac{1}{2(2a-3)!!} \frac{d^{a-1}}{\log(d)}$$

**Interaction function:**

$$F(z) = \exp(z)$$

**Energy function:**

$$E(\xi) = -\sum_{i=1}^{N} \exp(x_i^\top \xi) = -\exp(\mathrm{lse}(1, X^\top \xi))$$

**Capacity:**

$$C \approx 2^{d/2}$$

# Example

[Blog]

# Example

[Blog]

# Example


# Continuous Hopfield networks

# Continuous Hopfield networks

- Generalize energy function to continuous valued patterns [Krotov, Hopfield.'20]

$$E(\xi) = -\mathrm{lse}(\beta, X^\top \xi) + \frac{1}{2}\xi^\top \xi + \beta^{-1}\log N + \frac{1}{2}M^2$$

$$\mathrm{lse}(\beta, z) = \beta^{-1}\log\left(\sum_{l=1}^{N}\exp(\beta z_l)\right)$$

# Continuous Hopfield networks

- Generalize energy function to continuous valued patterns [Krotov, Hopfield.'20]

$$E(\xi) = -\mathrm{lse}(\beta, X^\top \xi) + \frac{1}{2}\xi^\top \xi + \beta^{-1}\log N + \frac{1}{2}M^2$$

- Update rule $\xi^{\mathrm{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\mathrm{lse}(\beta, z) = \beta^{-1}\log\left(\sum_{l=1}^{N}\exp(\beta z_l)\right)$$

# Continuous Hopfield networks

- Generalize energy function to continuous valued patterns [Krotov, Hopfield.'20]

$$E(\xi) = -\mathrm{lse}(\beta, X^\top \xi) + \frac{1}{2}\xi^\top \xi + \beta^{-1} \log N + \frac{1}{2} M^2$$

- Update rule $\xi^{\text{new}} = X \mathbb{S}(\beta X^\top \xi)$

Properties

$$\mathrm{lse}(\beta, z) = \beta^{-1} \log \left( \sum_{l=1}^{N} \exp(\beta z_l) \right)$$

# Continuous Hopfield networks

- Generalize energy function to continuous valued patterns [Krotov, Hopfield.'20]

$$E(\xi) = -\mathrm{lse}(\beta, X^\top \xi) + \frac{1}{2}\xi^\top \xi + \beta^{-1}\log N + \frac{1}{2}M^2$$

- Update rule $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

Properties

- Global convergence to local minimum

$$\mathrm{lse}(\beta, z) = \beta^{-1}\log\left(\sum_{l=1}^{N}\exp(\beta z_l)\right)$$

# Continuous Hopfield networks

- Generalize energy function to continuous valued patterns [Krotov, Hopfield.'20]

$$E(\xi) = -\mathrm{lse}(\beta, X^\top \xi) + \frac{1}{2}\xi^\top \xi + \beta^{-1} \log N + \frac{1}{2} M^2$$

- Update rule $\xi^{\mathrm{new}} = X \mathbb{S}(\beta X^\top \xi)$

Properties

- Global convergence to local minimum
- Exponential storage capacity

$$\mathrm{lse}(\beta, z) = \beta^{-1} \log \left( \sum_{l=1}^{N} \exp(\beta z_l) \right)$$

# Continuous Hopfield networks

- Generalize energy function to continuous valued patterns [Krotov, Hopfield.'20]

$$E(\xi) = -\mathrm{lse}(\beta, X^\top \xi) + \frac{1}{2}\xi^\top \xi + \beta^{-1} \log N + \frac{1}{2}M^2$$

- Update rule $\xi^{\mathrm{new}} = X \mathbb{S}(\beta X^\top \xi)$

Properties

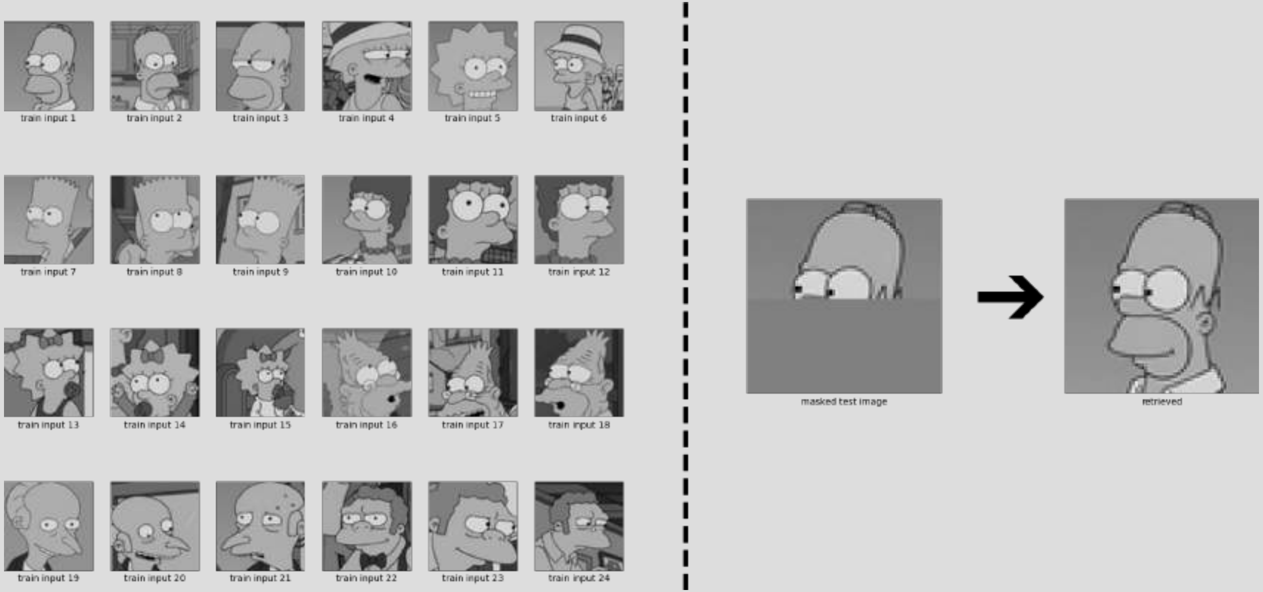
- Global convergence to local minimum
- Exponential storage capacity
- Convergence after one update step

$$\mathrm{lse}(\beta, z) = \beta^{-1} \log \left( \sum_{l=1}^{N} \exp(\beta z_l) \right)$$

# Example

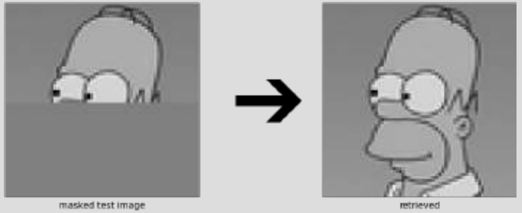
[Blog]

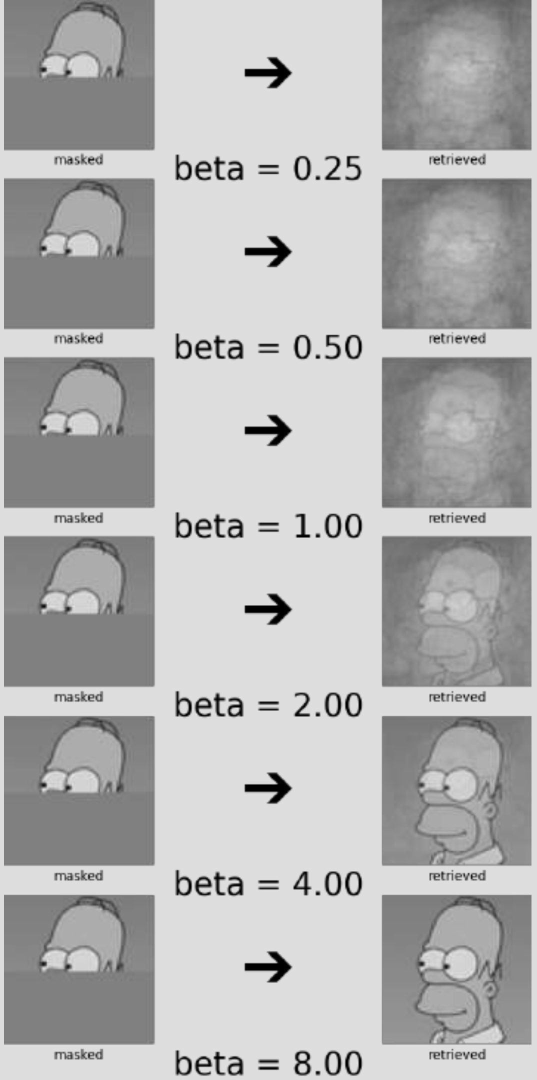
# Example




[Blog]

# Example


[Blog]

# Continuous Hopfield networks

This update rule looks familiar $\xi^{\mathrm{new}} = X\mathbb{S}(\beta X^\top \xi)$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$\Xi = [\xi_1, ..., \xi_S]$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$Q = \Xi^\top = RW_Q$$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$Q = \Xi^\top = RW_Q$$

$$K = X^\top = YW_K$$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$Q = \Xi^\top = RW_Q$$

$$K = X^\top = YW_K$$

$$\beta = \frac{1}{\sqrt{d_k}}$$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi) \qquad (Q^{\text{new}})^\top = K^\top \mathbb{S}(\frac{1}{\sqrt{d_k}} KQ^\top)$$

Substitute

$$Q = \Xi^\top = RW_Q$$

$$K = X^\top = YW_K$$

$$\beta = \frac{1}{\sqrt{d_k}}$$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$(Q^{\text{new}})^\top = K^\top \mathbb{S}(\frac{1}{\sqrt{d_k}} KQ^\top)$$

Substitute

$$Q = \Xi^\top = RW_Q$$

$$Q^{\text{new}} = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)K$$

Transpose

$$K = X^\top = YW_K$$

$$\beta = \frac{1}{\sqrt{d_k}}$$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$Q = \Xi^\top = RW_Q$$

$$K = X^\top = YW_K$$

$$\beta = \frac{1}{\sqrt{d_k}}$$

$$(Q^{\text{new}})^\top = K^\top \mathbb{S}(\frac{1}{\sqrt{d_k}} KQ^\top)$$ Substitute

$$Q^{\text{new}} = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)K$$ Transpose

$$Z = Q^{\text{new}} W_V = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)KW_V$$ Right multiply by $W_V$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$Q = \Xi^\top = RW_Q$$

$$K = X^\top = YW_K$$

$$\beta = \frac{1}{\sqrt{d_k}}$$

$$(Q^{\text{new}})^\top = K^\top \mathbb{S}(\frac{1}{\sqrt{d_k}} KQ^\top)$$ Substitute

$$Q^{\text{new}} = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)K$$ Transpose

$$Z = Q^{\text{new}} W_V = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)KW_V$$ Right multiply by $W_V$

$$\boxed{Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V}$$ Rewrite with $V = KW_V = YW_KW_V$

[Ramsauer et al.'20]

# Continuous Hopfield networks

$$Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V$$

This update rule looks familiar $\xi^{\text{new}} = X\mathbb{S}(\beta X^\top \xi)$

$$\Xi = [\xi_1, ..., \xi_S]$$

$$\Xi^{\text{new}} = X\mathbb{S}(\beta X^\top \Xi)$$

$$Q = \Xi^\top = RW_Q$$

$$K = X^\top = YW_K$$

$$\beta = \frac{1}{\sqrt{d_k}}$$

$$(Q^{\text{new}})^\top = K^\top \mathbb{S}(\frac{1}{\sqrt{d_k}} KQ^\top)$$

Substitute

$$Q^{\text{new}} = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)K$$

Transpose

$$Z = Q^{\text{new}} W_V = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)KW_V$$

Right multiply by $W_V$

$$\boxed{Z = \mathbb{S}(\frac{1}{\sqrt{d_k}} QK^\top)V}$$

Rewrite with
$V = KW_V = YW_KW_V$

$$Z = \mathbb{S}(\beta RW_QW_K^\top Y^\top)YW_KW_V$$

[Ramsauer et al.'20]

# Memorisation summary

# Memorisation summary

1. Hopfield networks are a classical model with **associative memory**

# Memorisation summary

1. Hopfield networks are a classical model with **associative memory**
2. Modern Hopfield networks **increase capacity** and **prevent interference** from correlated data

# Memorisation summary

1. Hopfield networks are a classical model with **associative memory**

2. Modern Hopfield networks **increase capacity** and **prevent interference** from correlated data

3. Modern Hopfield networks can be **extended to a continuous domain**

# Memorisation summary

1. Hopfield networks are a classical model with **associative memory**
2. Modern Hopfield networks **increase capacity** and **prevent interference** from correlated data
3. Modern Hopfield networks can be **extended to a continuous domain**
4. The **attention mechanism** carries out the **update rule** for a continuous Hopfield network

# Overview

- Transformers primer
- Optimisation
- Approximation
- Memorisation
- **In-context learning**

# What is in-context learning?

# What is in-context learning?

**Example 1:**

Lemon -> Yellow
Carrot -> Orange
Cucumber ->

Cucumber -> Green

# What is in-context learning?

**Example 1:**

Lemon -> Yellow
Carrot -> Orange
Cucumber ->

Cucumber -> Green

**Example 2:**

2?5 = 7
13?7 = 20
4?9 =

4?9 = 13

# What is in-context learning?

[Oswald et al.'22]

# What is in-context learning?

- $D_{\mathcal{X}}$ : distribution over examples $x_i$

[Oswald et al.'22]

# What is in-context learning?

- $D_{\mathcal{X}}$ : distribution over examples $x_i$
- $D_{\mathcal{F}}$ : distribution over functions $f$

[Oswald et al.'22]

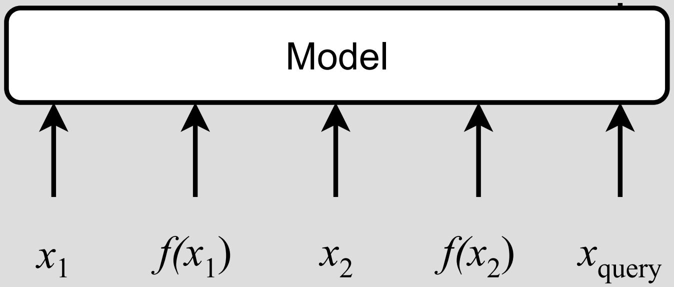
# What is in-context learning?

- $D_{\mathcal{X}}$ : distribution over examples $x_i$
- $D_{\mathcal{F}}$ : distribution over functions $f$
- Prompt: $P = (x_1, f(x_1), \ldots x_n, f(x_n), x_{\text{query}})$

[Oswald et al.'22]

# What is in-context learning?

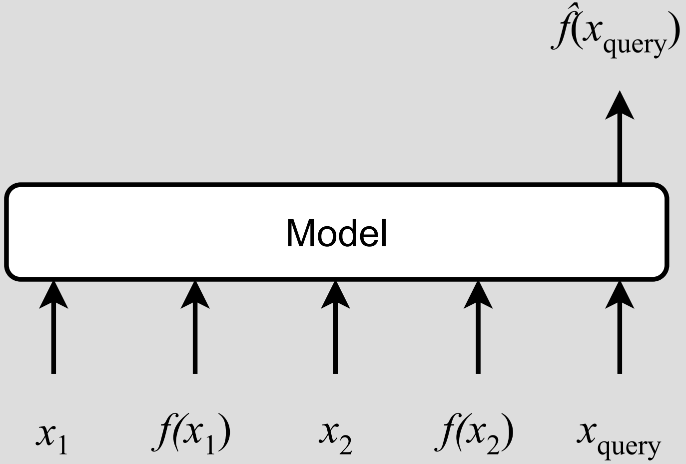
- $D_{\mathcal{X}}$ : distribution over examples $x_i$
- $D_{\mathcal{F}}$ : distribution over functions $f$
- Prompt: $P = (x_1, f(x_1), \ldots x_n, f(x_n), x_{\text{query}})$




[Oswald et al.'22]

# What is in-context learning?

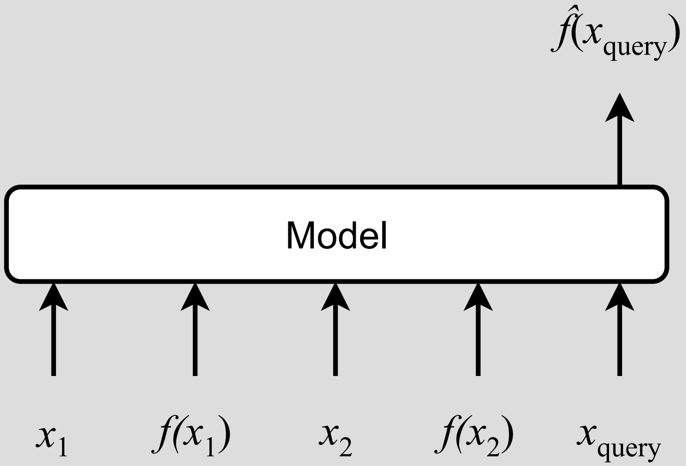
- $D_{\mathcal{X}}$ : distribution over examples $x_i$
- $D_{\mathcal{F}}$ : distribution over functions $f$
- Prompt: $P = (x_1, f(x_1), \ldots x_n, f(x_n), x_{\text{query}})$




[Oswald et al.'22]

# What is in-context learning?

- $D_{\mathcal{X}}$ : distribution over examples $x_i$
- $D_{\mathcal{F}}$ : distribution over functions $f$
- Prompt: $P = (x_1, f(x_1), \ldots x_n, f(x_n), x_{\text{query}})$




A model $M$ can ICL a function class $\mathcal{F}$ up to $\epsilon$ with respect to $(D_{\mathcal{X}}, D_{\mathcal{F}})$ for a loss function $\ell$ if:

$$\mathbb{E}_{(x,f)\sim(D_{\mathcal{X}},D_{\mathcal{F}})}\left[\ell(M(P), f(x_{query}))\right] \leq \epsilon$$

[Oswald et al.'22]

# Retrieval vs Learning

# Retrieval vs Learning

- **Hypothesis #1**: ICL is retrieval [Xie et al., 2022]
    - Not 'learning' of new skills, but 'locating' of skills the model already has (e.g., translation)

# Retrieval vs Learning

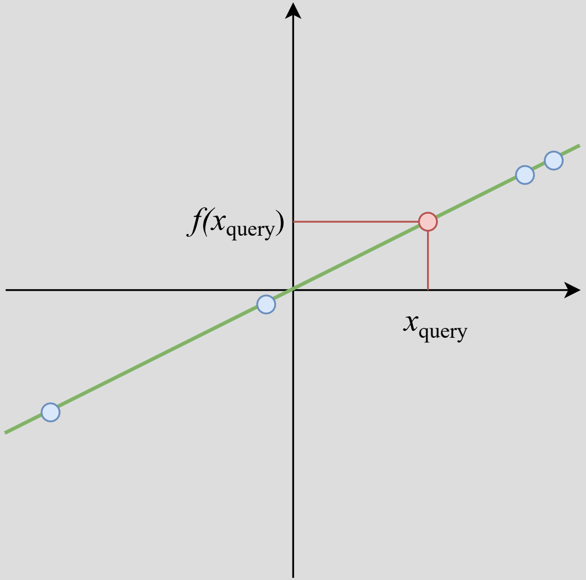
- **Hypothesis #1**: ICL is retrieval [Xie et al., 2022]
  - Not 'learning' of new skills, but 'locating' of skills the model already has (e.g., translation)

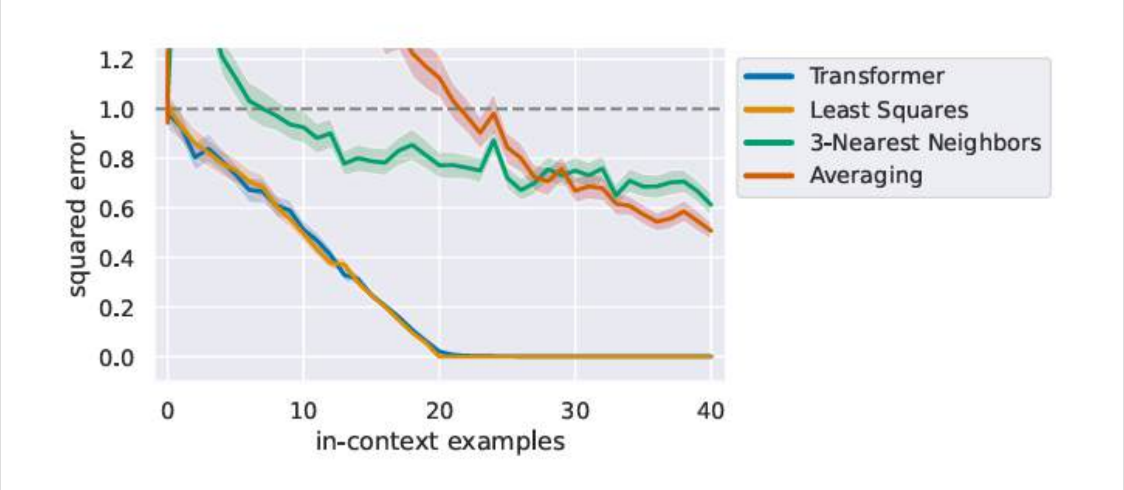
- **Hypothesis #2**: ICL is learning
  - It can identify novel functions from within a function class (e.g., puzzle-solving tasks)

# Linear regression

$$\mathcal{F} = \{f : f(x) = w^\top x\} \quad w \sim \mathcal{N}(0, I)$$

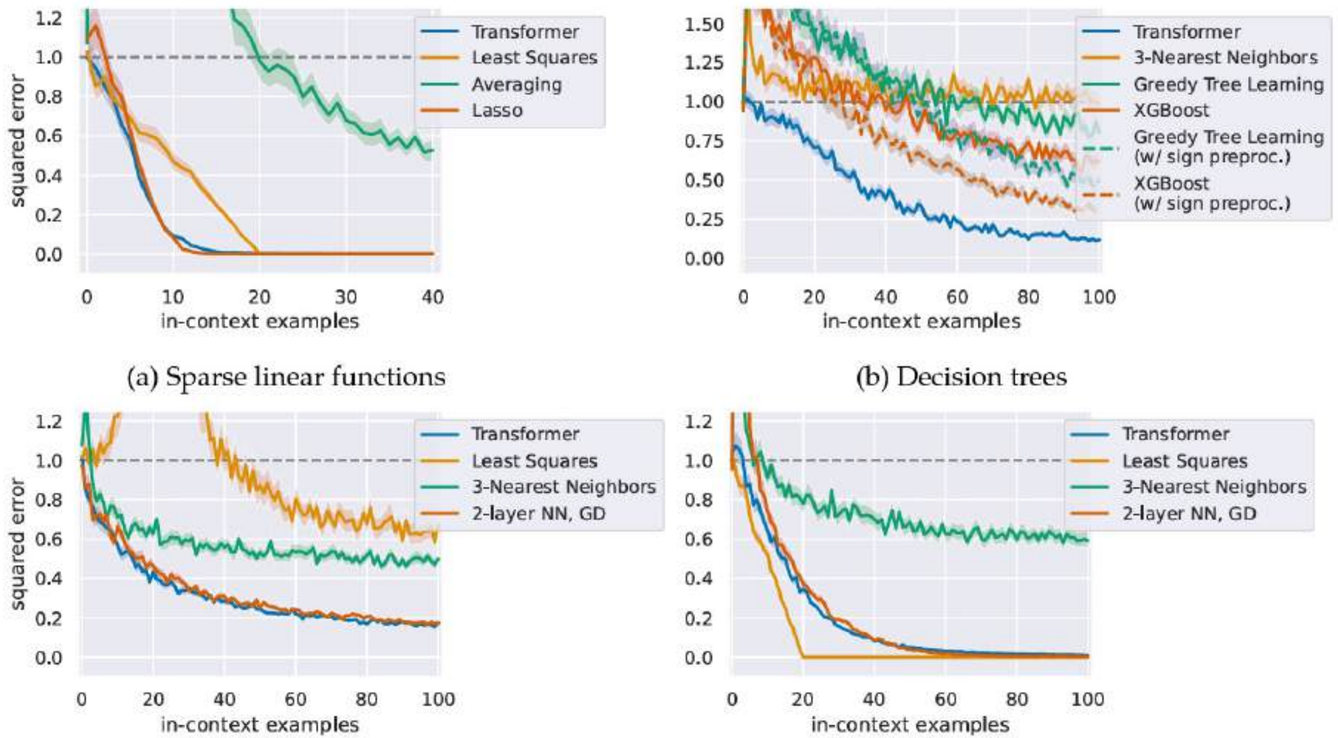
$$\mathcal{X} = \mathbb{R}^d \quad x \sim \mathcal{N}(0, I)$$




[Oswald et al.'22]

# Non-linear regression




(a) Sparse linear functions

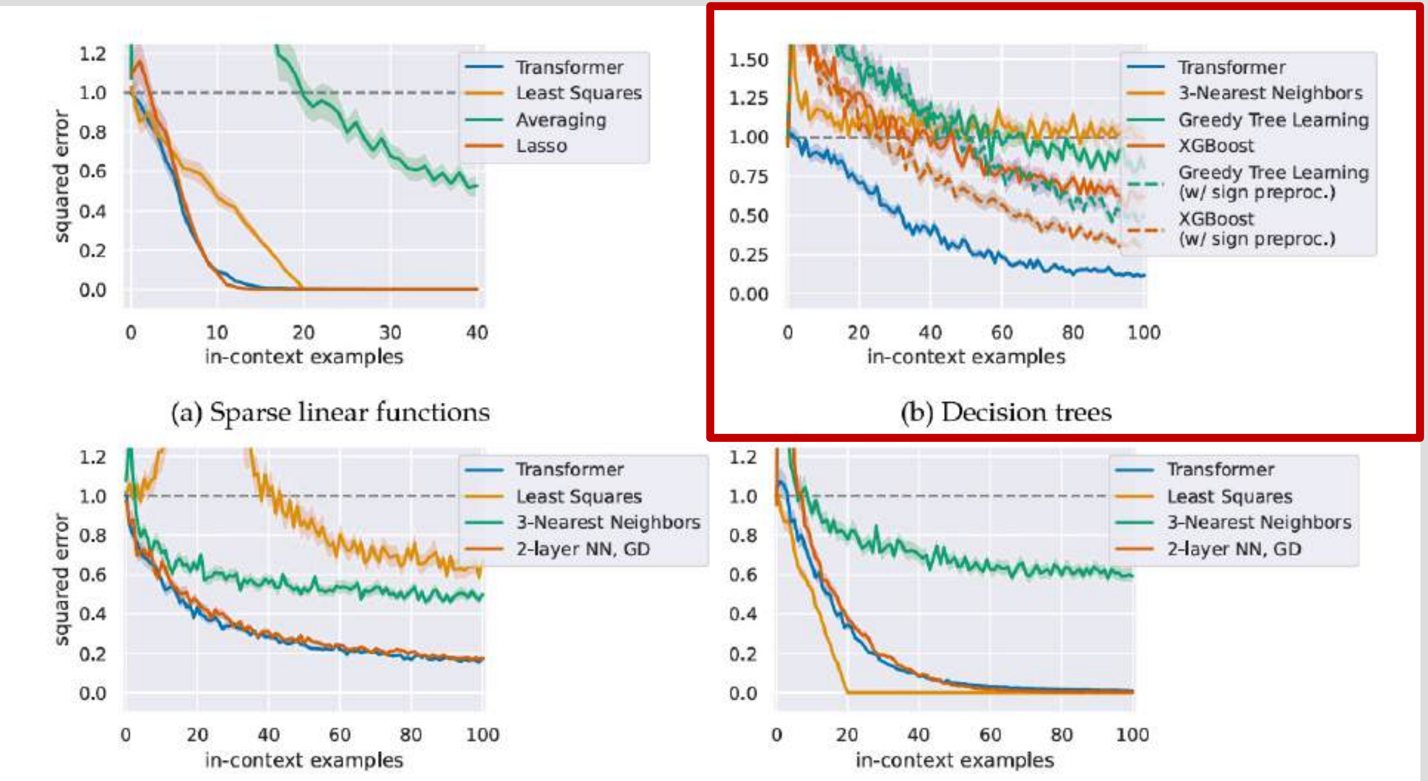
(b) Decision trees

(c) 2-layer NN

(d) 2-layer NN, eval on linear functions

[Oswald et al.'22]

# Non-linear regression




(a) Sparse linear functions

(b) Decision trees

(c) 2-layer NN

(d) 2-layer NN, eval on linear functions

# Gradient descent

# Gradient descent

$$\mathcal{L}(W + \Delta W) = \frac{1}{2N} \sum_{i=1}^{n} \|(W + \Delta W)x_i - y_i\|^2$$

# Gradient descent

$$\mathcal{L}(W + \Delta W) = \frac{1}{2N} \sum_{i=1}^{n} \|(W + \Delta W)x_i - y_i\|^2$$

$$\mathcal{L}(W + \Delta W) = \frac{1}{2N} \sum_{i=1}^{n} \|W x_i - (y_i - \Delta y_i)\|^2 \qquad \Delta y_i = \Delta W x_i$$

# Gradient descent

$$\mathcal{L}(W + \Delta W) = \frac{1}{2N}\sum_{i=1}^{n} \|(W + \Delta W)x_i - y_i\|^2$$

$$\mathcal{L}(W + \Delta W) = \frac{1}{2N}\sum_{i=1}^{n} \|Wx_i - (y_i - \Delta y_i)\|^2 \qquad \Delta y_i = \Delta W x_i$$

**Claim [Oswald et al.'22]:** We can construct a 1-head linear attention layer such that a Transformer step on every token $e_j$ is $e_j \leftarrow (x_j, y_j) + (0, -\Delta W x_j) = (x_j, y_j) + PVK^\top q_j$ where $e_j = (x_j, y_j - \Delta y_j)$

# In-context learning summary

# In-context learning summary

1. Transformers are **capable** of in context learning

# In-context learning summary

1. Transformers are **capable** of in context learning

2. They can learn more sample efficient algorithms than are known

# In-context learning summary

1. Transformers are **capable** of in context learning
2. They can learn more sample efficient algorithms than are known
3. 1-head Linear attention transformers **can and do** learn to do a single step of gradient descent

# Summary

# Summary

1. Detailed the internals of the **transformer**, particularly the **attention mechanism**

# Summary

1. Detailed the internals of the **transformer**, particularly the **attention mechanism**

2. Explained **sparseness** in attention maps

# Summary

1. Detailed the internals of the **transformer**, particularly the **attention mechanism**

2. Explained **sparseness** in attention maps

3. Showed gradient descent converges to a **max-margin** solution for an **equivalent SVM**

# Summary

1. Detailed the internals of the **transformer**, particularly the **attention mechanism**

2. Explained **sparseness** in attention maps

3. Showed gradient descent converges to a **max-margin** solution for an **equivalent SVM**

4. Showed transformers are **universal approximators**

# Summary

1. Detailed the internals of the **transformer**, particularly the **attention mechanism**
2. Explained **sparseness** in attention maps
3. Showed gradient descent converges to a **max-margin** solution for an **equivalent SVM**
4. Showed transformers are **universal approximators**
5. Showed equivalence between **continuous modern Hopfield networks** and dot product attention

# Summary

1. Detailed the internals of the **transformer**, particularly the **attention mechanism**
2. Explained **sparseness** in attention maps
3. Showed gradient descent converges to a **max-margin** solution for an **equivalent SVM**
4. Showed transformers are **universal approximators**
5. Showed equivalence between **continuous modern Hopfield networks** and dot product attention
6. Showed transformers implement a step of **gradient descent** when in-context learning

# Questions?