RoFormer: Enhanced Transformer with Rotary Position Embedding

fzeng

05/03/24

Outline

- Review of attention and why positional encodings are needed
- Sinusoidal positional encodings intuition and limitations
- Positional encoding desiderata
- Derivation of rotary positional embeddings

Attention

Attention

- Not all parts of the input equally important for task at hand
- E.g. image classification: background does not matter, helps to ignore spurious features
- Idea: provide more weight for more relevant features, fade out less relevant features
- Features now context-aware

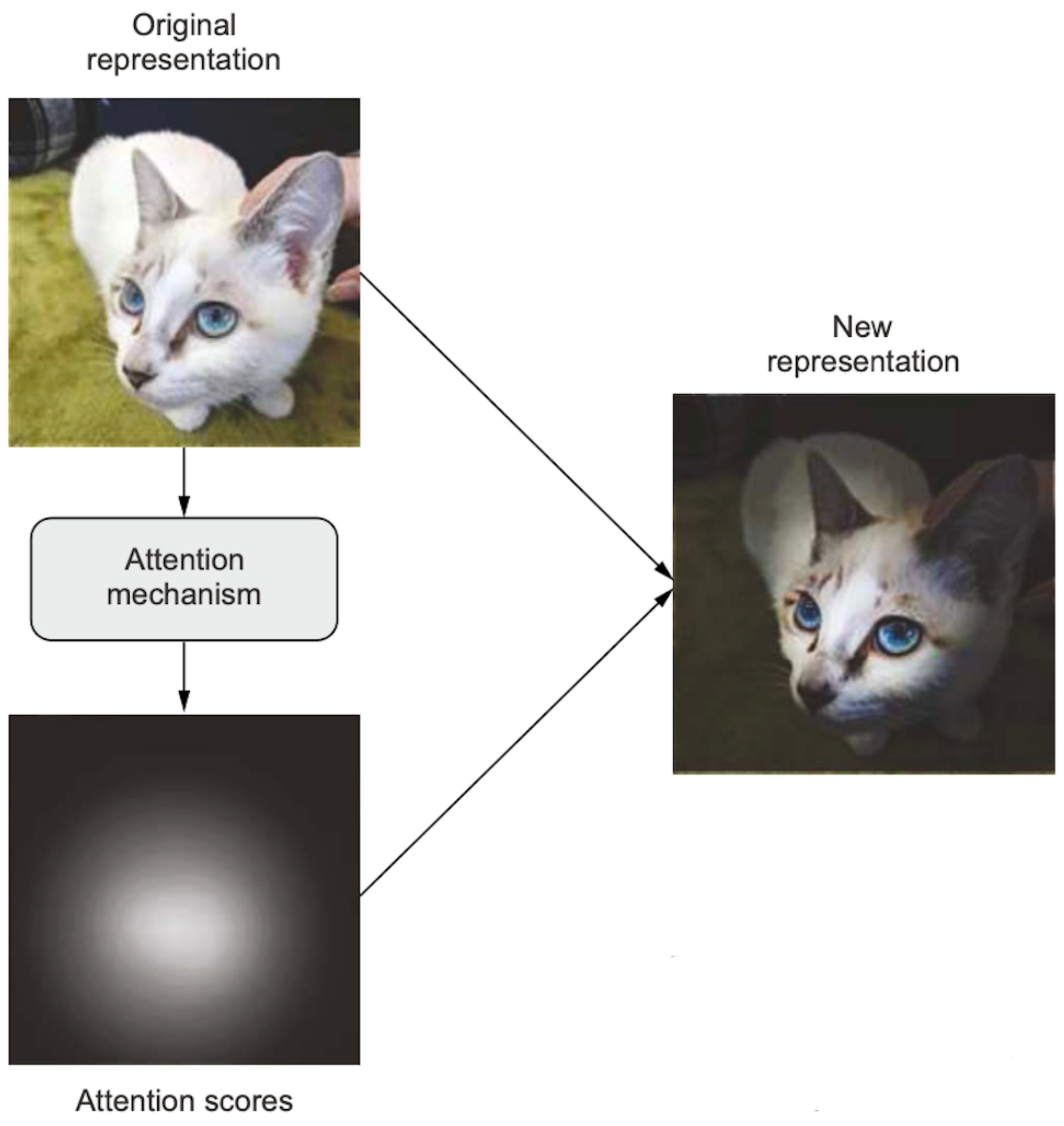


Image from Deep Learning with Python

Attention

- 3 components: query, key, values
- Terminology inspired by search engines
- Suppose you have a dataset of key-value pairs: (image tags, images)
- For a given query, how would you weigh your values to return the values blended by how important they are?
- Need some notion of similarity between the query and each of the keys!

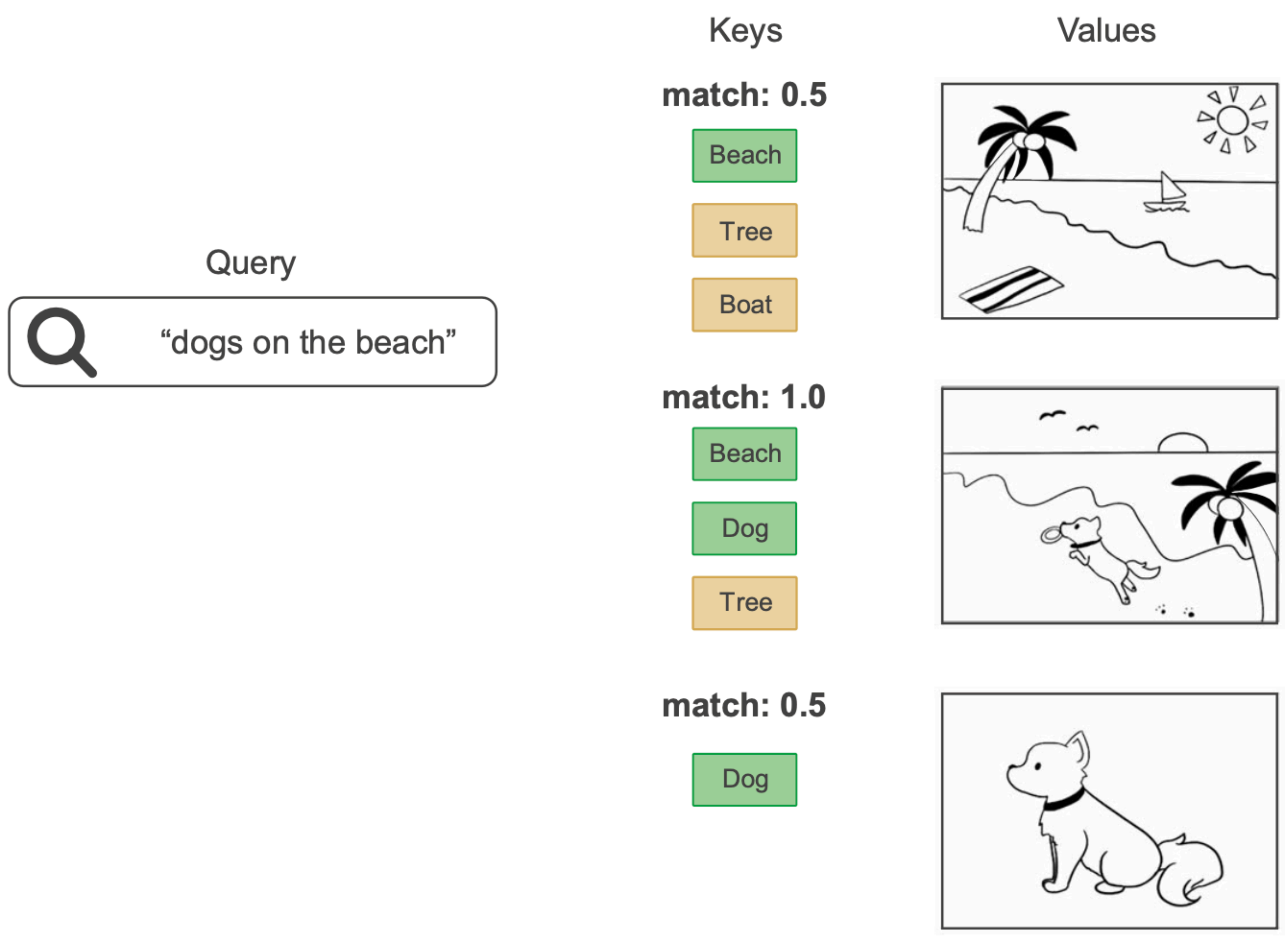


Image from Deep Learning with Python

Please Pay Attention

- We will derive the most famous equation in machine learning (Eq 1 in Attention Is All You Need):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

Attention

- Concretely: suppose we have m keys and values $\{(\mathbf{k}_1, \mathbf{v}_1), \ldots (\mathbf{k}_m, \mathbf{v}_m)\}$
- Define attention on a query as:

$$\text{Attention}(\mathbf{q}) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$$

  for some weighing function $\alpha(\mathbf{q}, \mathbf{k}_i)$

- Idea: assigns different importance to each $\mathbf{v}_i$ depending on how similar $\mathbf{q}$ and $\mathbf{k}_i$ are!

Attention

- Attention$(\mathbf{q}) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$
- What is a good choice for $\alpha(\mathbf{q}, \mathbf{k}_i)$?
- Want non-negativity: $\alpha(\mathbf{q}, \mathbf{k}_i) > 0$
- Want normalization to 1: $\sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i) = 1$

Attention

- Suppose we have an arbitrary similarity function $a(\mathbf{q}, \mathbf{k}_i)$
- We can use it to construct α:
- Non-negativity: take exponentials, $\exp(a(\mathbf{q}, \mathbf{k}_i)) > 0$
- Normalization to 1: divide by sum of all values,

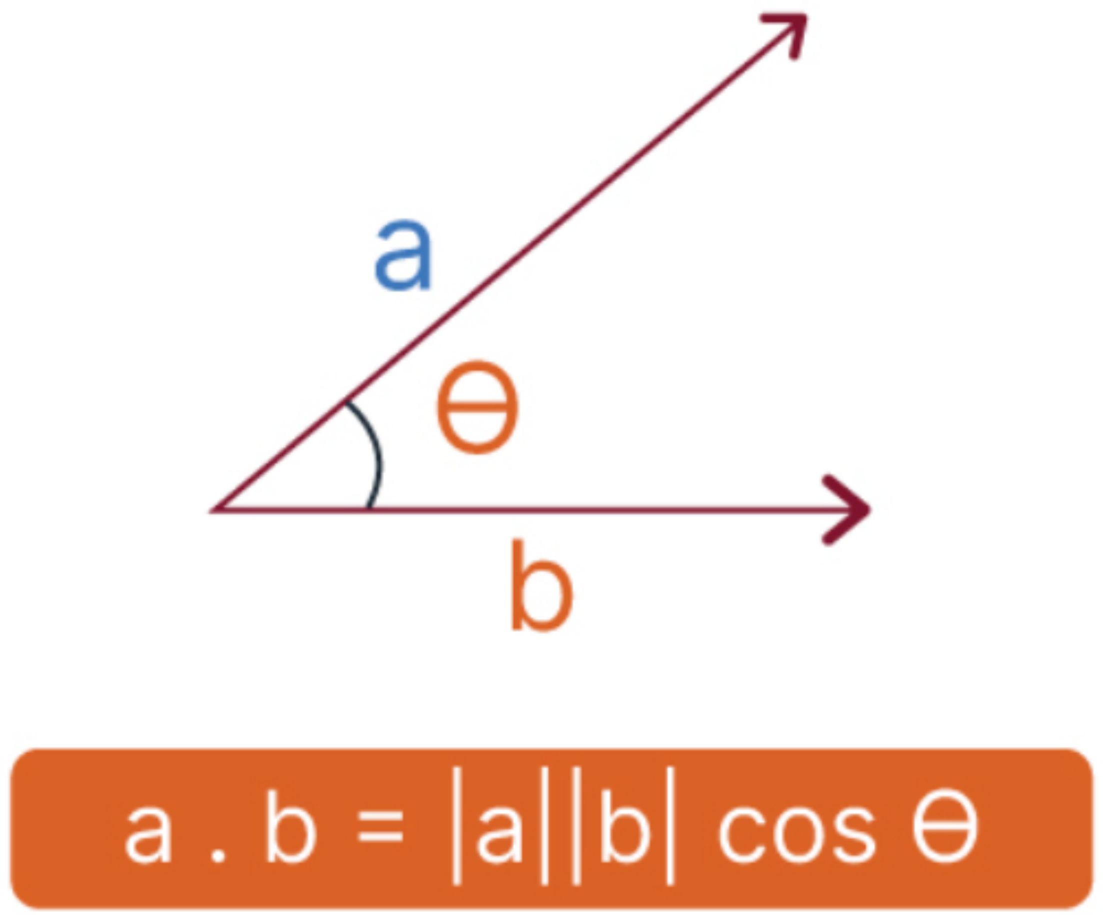
$$\alpha(\mathbf{q}, \mathbf{k}_i) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_j \exp(a(\mathbf{q}, \mathbf{k}_j))} .$$

- Actually the above is just the softmax function:

$$\text{softmax}(\mathbf{x_i}) = \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x}_j)} .$$

Attention

- $\text{Attention}(\mathbf{q}) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$
- What is a good choice for $a(\mathbf{q}, \mathbf{k}_i)$?
- Dot product: distance metric that extends to arbitrary dimensions, measures “angle” between two vectors as notion of similarity
- So now we have dot product $\mathbf{q}^\top \mathbf{k}_i$



Attention

- Suppose $\mathbf{q}, \mathbf{k}_i$ are d-dimensional and drawn independently from standard normal distribution
- Dot product $\mathbf{q}^\top \mathbf{k}_i$ is now the sum of d products of two independent standard Gaussians
- If $X_i, Y_i \sim \mathcal{N}(0,1)$ i.i.d, then $E[X_i Y_i] = 0,\ Var(X_i Y_i) = 1$
- By linearity of expectations, $E\left[\sum_{i=1}^{d} X_i Y_i\right] = 0$
- By linearity of variance, $Var\left(\sum_{i=1}^{d} X_i Y_i\right) = d$
- High variance leads to instability especially since we have exponentials 😔

Attention

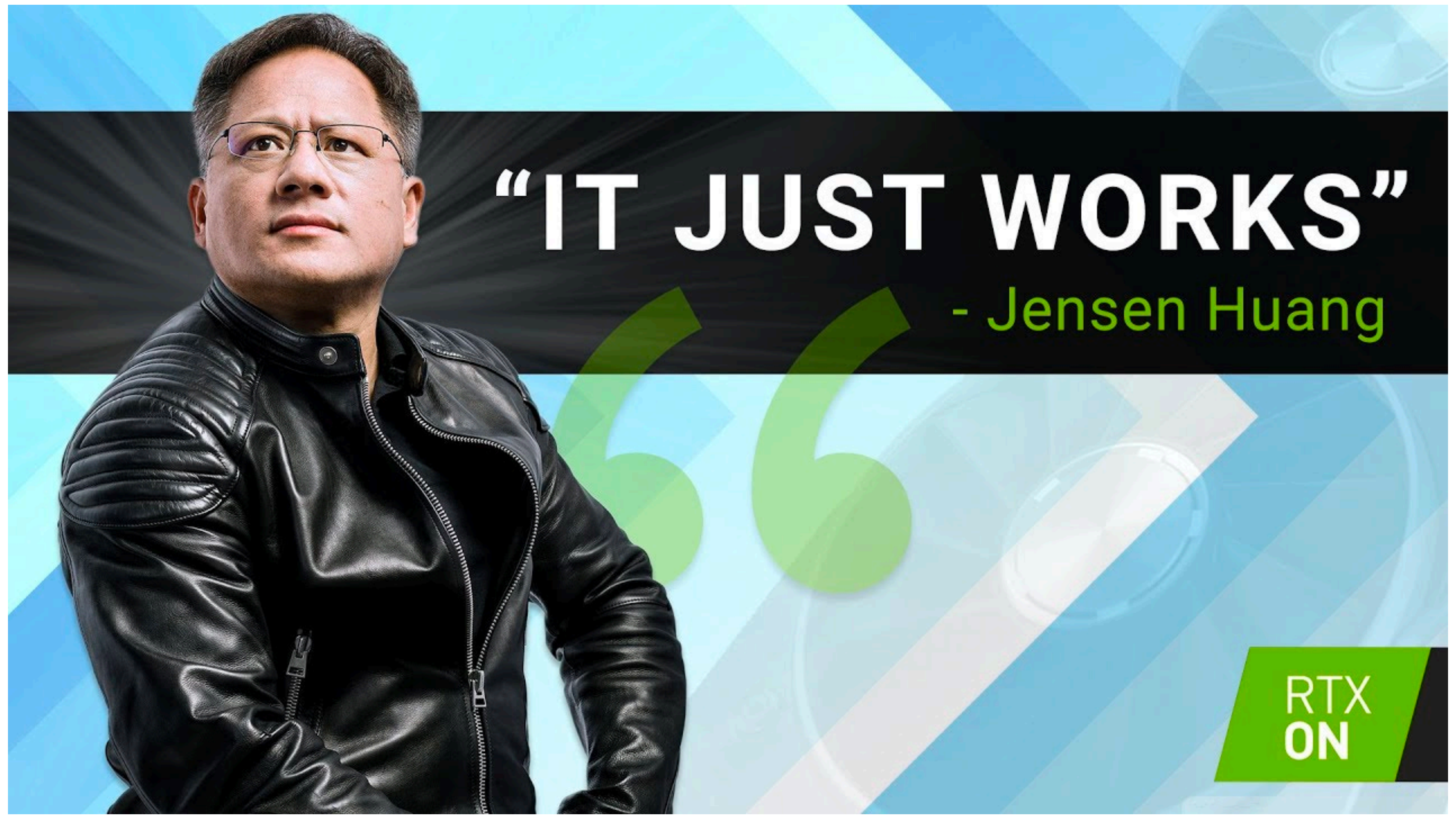
- Solution: scale by $1/\sqrt{d}$ to result in unit variance, since $Var(cX) = c^2 Var(X)$
- Putting everything together, we have scaled dot-product attention:

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \frac{\exp(\mathbf{q}^\top \mathbf{k}_i / \sqrt{d})}{\sum_j \exp(\mathbf{q}^\top \mathbf{k}_j / \sqrt{d})} = \mathrm{softmax}\left(\frac{\mathbf{q}^\top \mathbf{k}_i}{\sqrt{d}}\right)$$

- We are getting close 😊

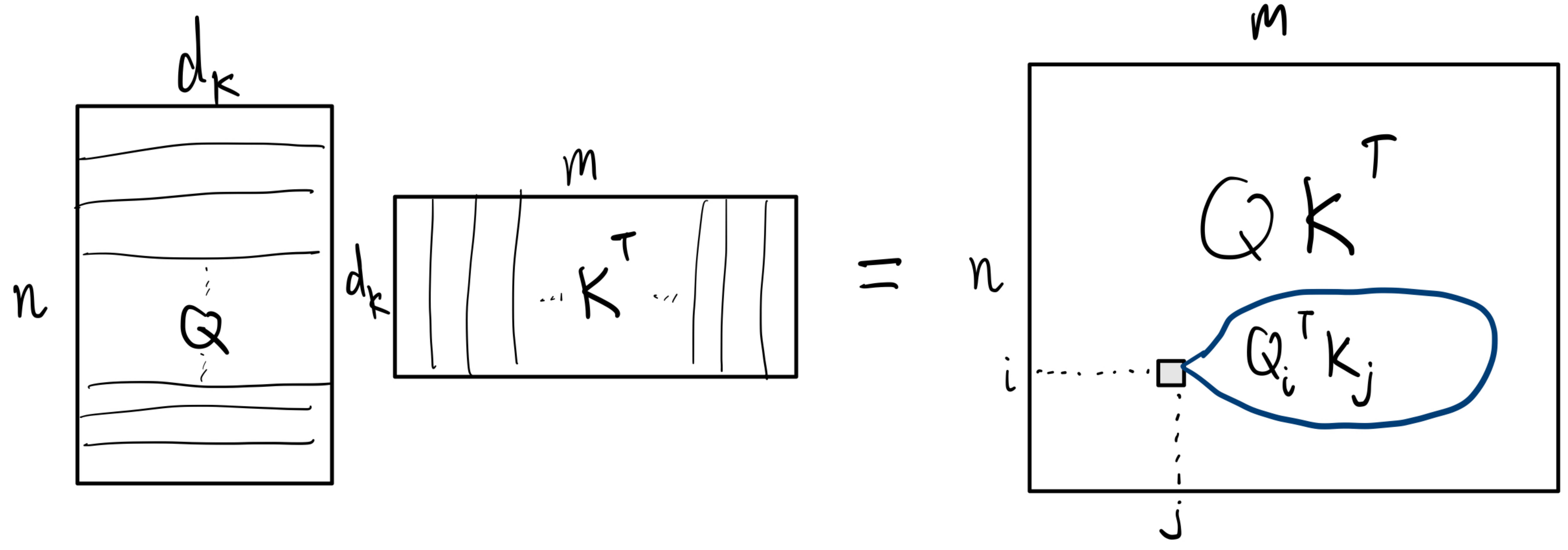
Batching

- Jensen Huang has blessed us with GPUs optimized for multiplying large matrices
- Instead of processing just an individual sample at a time, more efficient throughput-wise to *batch* multiple samples together



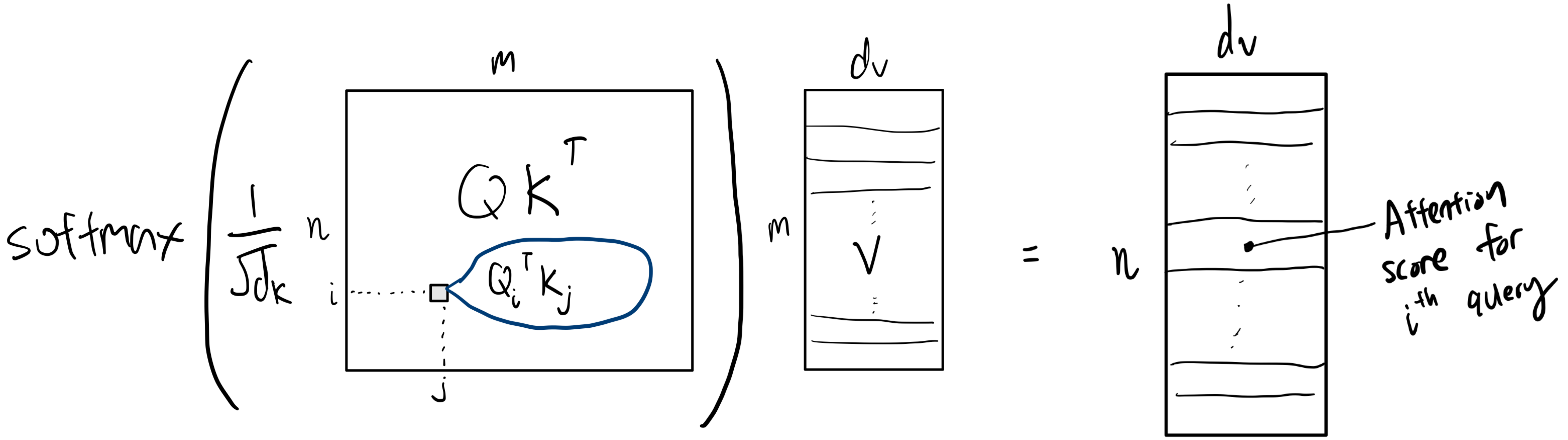
Batched Attention

- Suppose you have n queries and m keys and m values stacked together as matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ respectively
- Each key, query must have same dimension for dot product: d_k
- Each value has dimension d_v
- First compute $\mathbf{QK}^\top$



Batched Attention

- Next scale matrix entries, take softmax over each *row* in the matrix
- Multiply by $\mathbf{V}$, get batched attention:

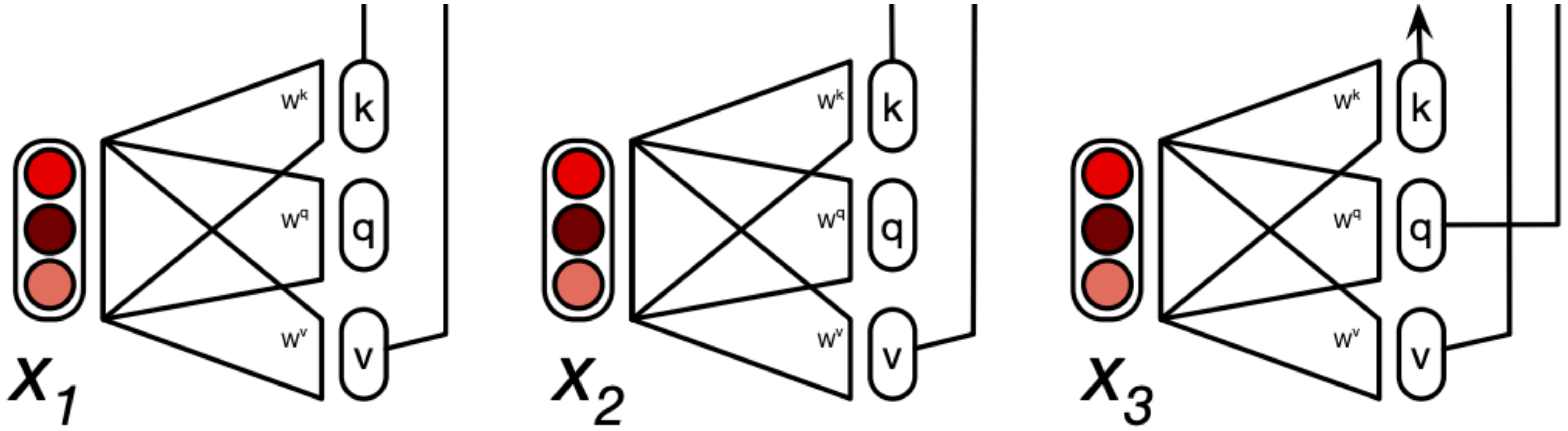


Overall:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d}}\right)\mathbf{V}$$

Self-Attention

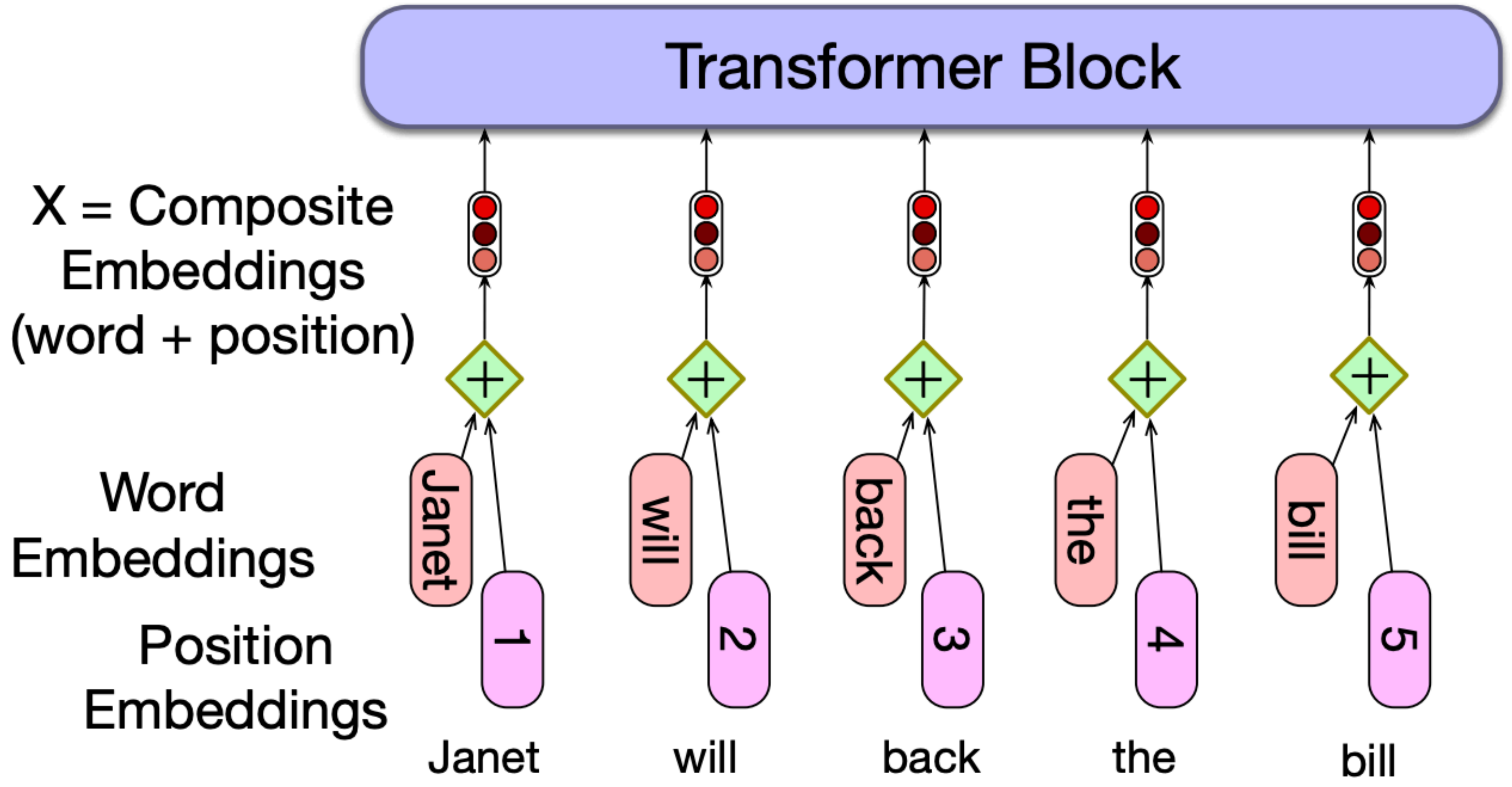
- Given input vector x_i (corresponding to some token)
- Learn weight matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$
- Project x_i by respective matrices for query, key, and value: $q_i = x_i\mathbf{W}^Q, k_i = x_i\mathbf{W}^K, v_i = x_i\mathbf{W}^V$
- Parallelizing this computation with input matrix $\mathbf{X}$ instead, we recover $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \mathbf{K} = \mathbf{X}\mathbf{W}^K, \mathbf{V} = \mathbf{X}\mathbf{W}^V$



Positional Encodings

Positional Encoding

- Self-attention by itself is order-agnostic
- But ordering information is important in language!
- Idea: for each input token, add (not concatenate!) a vector denoting positional information to it
- Drawback to naive approach: short sequences much more common than long ones during training, so later embeddings may be poorly trained and fail to generalize



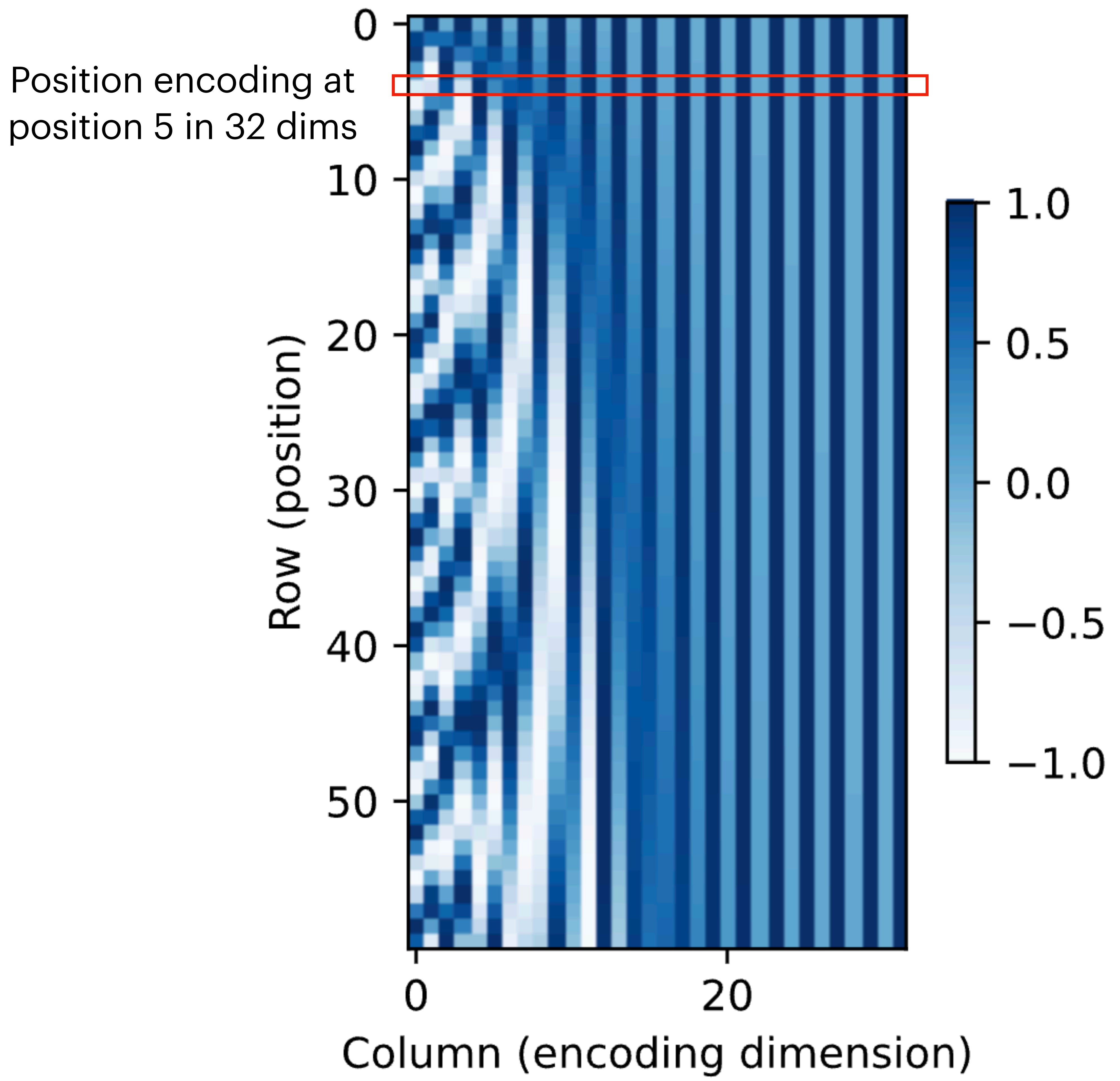
Naive approach of just using position itself as the position embedding

Positional Encoding

- In Attention Is All You Need, authors used sinusoidal positional encoding
- Positional encoding for ith row and $2j$ or $2j+1$ th column in d dimensions:

$$p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right),$$

- $$p_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right).$$



Attention Is All You Need (Vaswani et al. 2017)

Why Sinusoidal Positional Encodings?

- Can transform from one index to an offset using only linear operations
- To get from $\boldsymbol{p}_{i,2t}, \boldsymbol{p}_{i,2t+1}$ to $\boldsymbol{p}_{i+k,2t} \boldsymbol{p}_{i+k,2t+1}$:
- Write $a = \dfrac{i}{10000^{2t/d}}, \qquad b = \dfrac{k}{10000^{2t/d}}$.
- Then

$$\begin{aligned} \boldsymbol{p}_{i,2t} &= \sin(a) \\ \boldsymbol{p}_{i,2t+1} &= \cos(a) \end{aligned}$$

- $$\begin{aligned} \boldsymbol{p}_{i+k,2t} &= \sin(a+b) \\ \boldsymbol{p}_{i+k,2t+1} &= \cos(a+b) . \end{aligned}$$

$$\begin{aligned} p_{i,2j} &= \sin\left(\frac{i}{10000^{2j/d}}\right), \\ p_{i,2j+1} &= \cos\left(\frac{i}{10000^{2j/d}}\right). \end{aligned}$$

Attention Is All You Need (Vaswani et al. 2017)

Why Sinusoidal Positional Encodings?

- Recall

$$\sin(a+b) = \sin(a)\cos(b) + \cos(a)\sin(b)$$
$$\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$$

- Then the following rotation matrix gives the desired transformation:

$$\begin{bmatrix} \cos(b) & \sin(b) \\ -\sin(b) & \cos(b) \end{bmatrix} \begin{bmatrix} \sin(a) \\ \cos(a) \end{bmatrix} = \begin{bmatrix} \sin(a+b) \\ \cos(a+b) \end{bmatrix}.$$

- Hope: $\mathbf{W}$ learns how to perform this rotation by offsets during training

Attention Is All You Need (Vaswani et al. 2017)

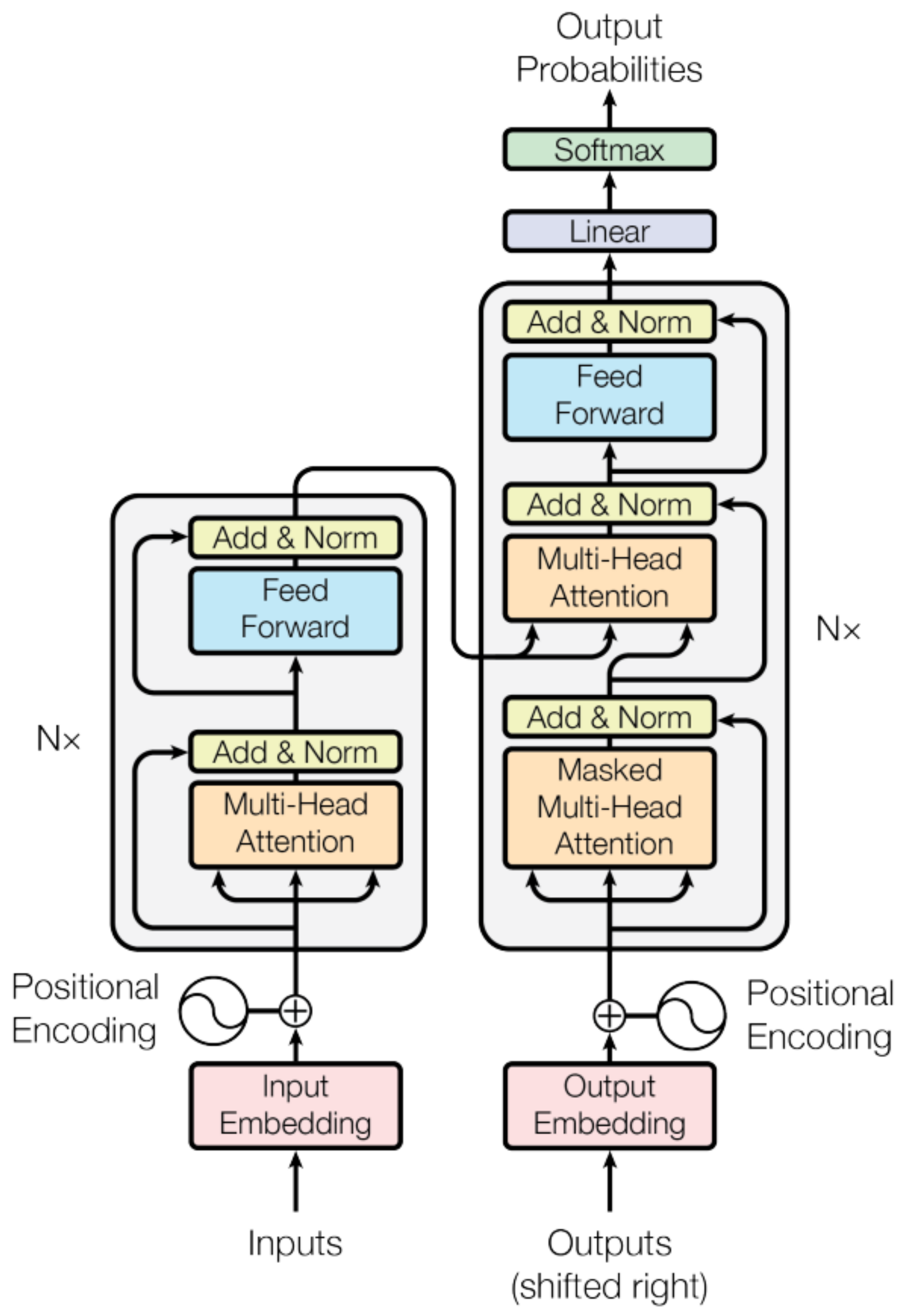
Positional Encoding

- For first layer, add positional encoding to embeddings:

$$\boldsymbol{q}_m = \mathbf{W}_q \left(\boldsymbol{x}_m + \boldsymbol{p}_m\right)$$

$$\boldsymbol{k}_n = \mathbf{W}_k \left(\boldsymbol{x}_n + \boldsymbol{p}_n\right)$$

- $$\boldsymbol{v}_n = \mathbf{W}_v \left(\boldsymbol{x}_n + \boldsymbol{p}_n\right)$$



Attention Is All You Need (Vaswani et al. 2017)

Flaws

- However, this is still poor for encoding relative positional information if you expand the query/key dot product:

$$q_m^\top k_n = \left(\mathbf{W}_q\left(x_m + p_m\right)\right)^\top \left(\mathbf{W}_k\left(x_n + p_n\right)\right)$$

- $$\begin{aligned} &= x_m^\top \mathbf{W}_q^\top \mathbf{W}_k x_n + x_m^\top \mathbf{W}_q^\top \mathbf{W}_k p_n \\ &+ p_m^\top \mathbf{W}_q^\top \mathbf{W}_k x_n + p_m^\top \mathbf{W}_q^\top \mathbf{W}_k p_n \end{aligned}$$

- Two terms contain only one of p_m or p_n, not possible to preserve relative offsets!

RoFormer: Enhanced Transformer with Rotary Position Embedding

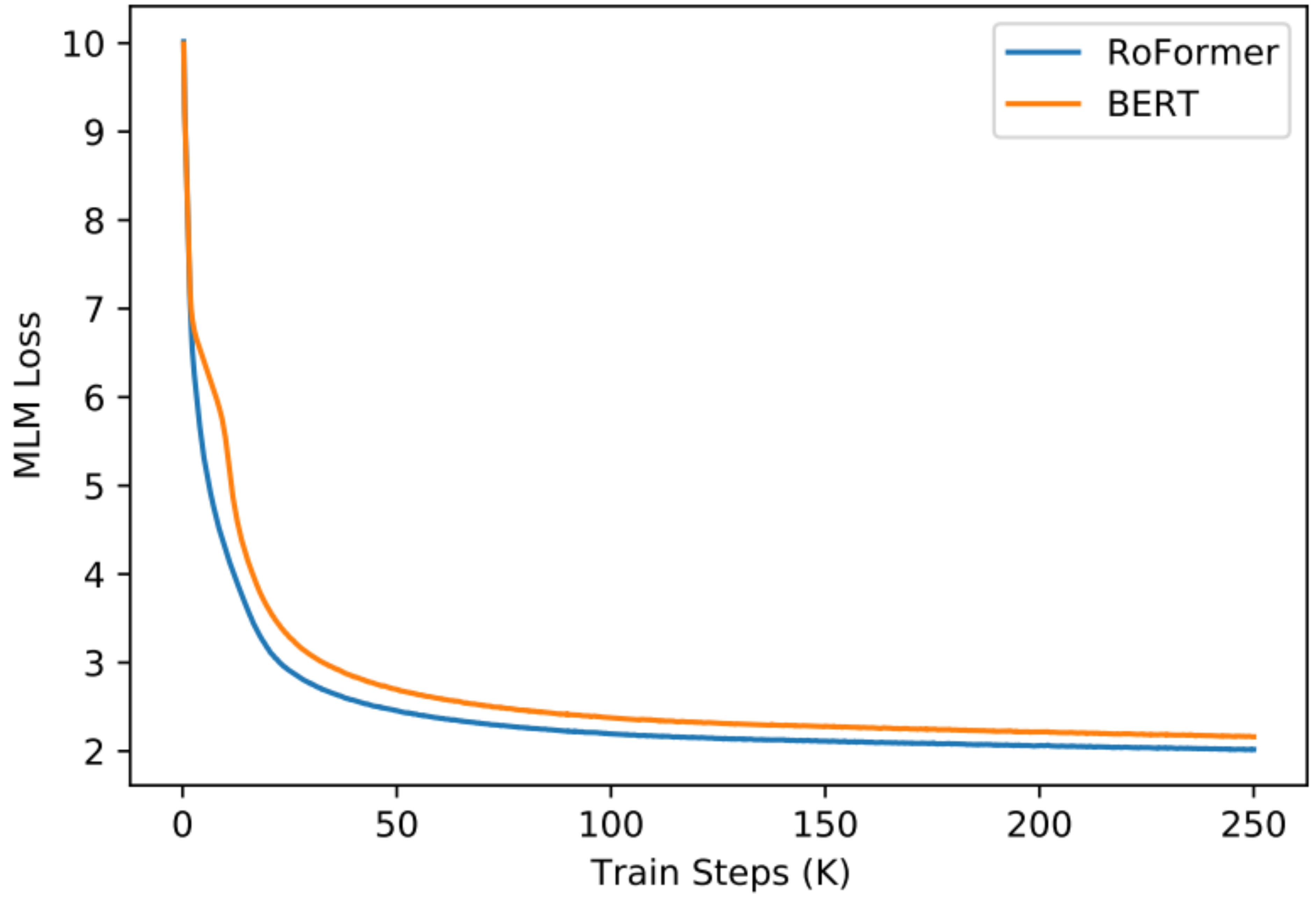
RoPE

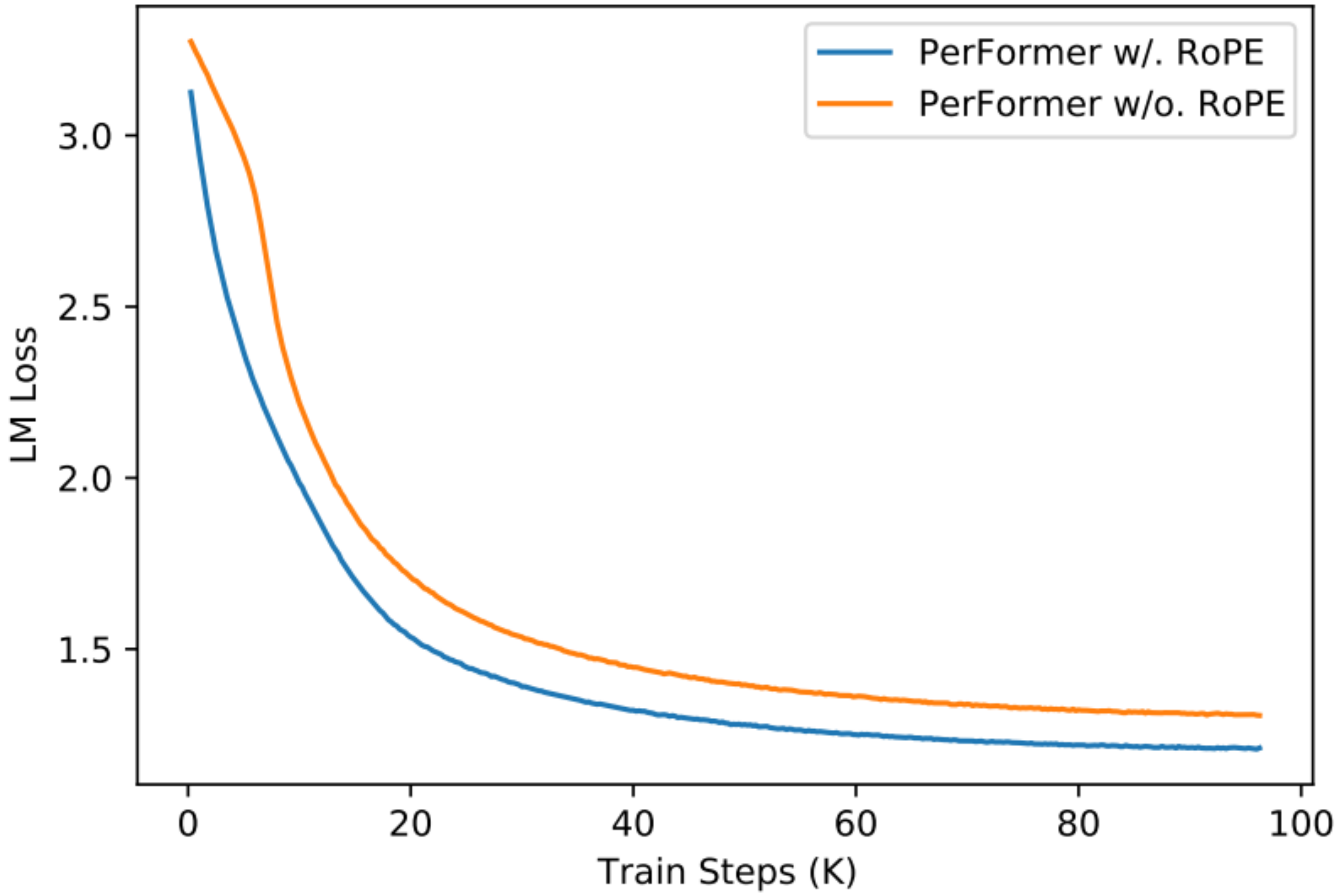
- Used by most open source models today

	OLMo-7B	LLaMA2-7B	OpenLM-7B	Falcon-7B	PaLM-8B
Dimension	4096	4096	4096	4544	4096
Num heads	32	32	32	71	16
Num layers	32	32	32	32	32
MLP ratio	~8/3	~8/3	~8/3	4	4
Layer norm type	non-parametric	RMSNorm	parametric	parametric	parametric
Positional embeddings	RoPE	RoPE	RoPE	RoPE	RoPE
Attention variant	full	GQA	full	MQA	MQA
Biases	none	none	in LN only	in LN only	none
Block type	sequential	sequential	sequential	parallel	parallel
Activation	SwiGLU	SwiGLU	SwiGLU	GeLU	SwiGLU
Sequence length	2048	4096	2048	2048	2048
Batch size (instances)	2160	1024	2048	2304	512
Batch size (tokens)	~4M	~4M	~4M	~4M	~1M
Weight tying	no	no	no	no	yes

RoPE

- Faster convergence than with sinusoidal position encoding





RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE

- Used by most open source models today

	OLMo-7B	LLaMA2-7B	OpenLM-7B	Falcon-7B	PaLM-8B
Dimension	4096	4096	4096	4544	4096
Num heads	32	32	32	71	16
Num layers	32	32	32	32	32
MLP ratio	~8/3	~8/3	~8/3	4	4
Layer norm type	non-parametric	RMSNorm	parametric	parametric	parametric
Positional embeddings	RoPE	RoPE	RoPE	RoPE	RoPE
Attention variant	full	GQA	full	MQA	MQA
Biases	none	none	in LN only	in LN only	none
Block type	sequential	sequential	sequential	parallel	parallel
Activation	SwiGLU	SwiGLU	SwiGLU	GeLU	SwiGLU
Sequence length	2048	4096	2048	2048	2048
Batch size (instances)	2160	1024	2048	2304	512
Batch size (tokens)	~4M	~4M	~4M	~4M	~1M
Weight tying	no	no	no	no	yes

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

Goals

- Can we come up with a positional encoding scheme that:
  - Models relative positional information directly
  - Doesn't introduce terms that depend on absolute position indices
- Perhaps something like

$$q_m^T k_n = x_m^T \mathbf{W}_q \mathbf{R}_{n-m} \mathbf{W}_k \mathbf{x_n}$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Overview

- Rotate the pre-activations instead of adding:

  $\boldsymbol{q}_m = \mathbf{R}^d_{\Theta,m} \mathbf{W}_q \boldsymbol{x}_m$

- $\boldsymbol{k}_n = \mathbf{R}^d_{\Theta,n} \mathbf{W}_k \boldsymbol{x}_n$

- $$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

Deriving RoPE

Goal

- Consider $d = 2$ case (can easily generalize from here to even dimensions)
- Want only dot-product attention to only depend on relative positions:

$$\boldsymbol{q}_m^T \boldsymbol{k}_n = \left\langle f_q\left(\boldsymbol{x}_m, m\right), f_k\left(\boldsymbol{x}_n, n\right) \right\rangle = g\left(\boldsymbol{x}_m, \boldsymbol{x}_n, m - n\right)$$

- Goal is to learn a suitable f_q, f_k, g
- Notation for initial conditions (we can choose $\boldsymbol{q}, \boldsymbol{k}$ afterwards)

$$\boldsymbol{q} = f_q\left(\boldsymbol{x}_q, 0\right)$$

$$\boldsymbol{k} = f_k\left(\boldsymbol{x}_k, 0\right)$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

Re-interpretation in Complex Form

- Since $d = 2$, can re-interpret f_q, f_k, g in complex polar coordinates:

$$f_q\left(\boldsymbol{x}_q, m\right) = R_q\left(\boldsymbol{x}_q, m\right) e^{i\Theta_q\left(\boldsymbol{x}_q, m\right)}$$

$$f_k\left(\boldsymbol{x}_k, n\right) = R_k\left(\boldsymbol{x}_k, n\right) e^{i\Theta_k\left(\boldsymbol{x}_k, n\right)}$$

$$g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right) = R_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right) e^{i\Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right)}$$

- $R_{\{q,k,v\}}$: magnitude
- $\Theta_{\{q,k,g\}}$: angle

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

Re-interpretation in Complex Form

- Do the same for initial conditions $\boldsymbol{q}, \boldsymbol{k}$:

$$\boldsymbol{q} = \|\boldsymbol{q}\| e^{i\theta_q} = R_q\left(\boldsymbol{x}_q, 0\right) e^{i\Theta_q\left(\boldsymbol{x}_q, 0\right)}$$

$$\boldsymbol{k} = \|\boldsymbol{k}\| e^{i\theta_k} = R_k\left(\boldsymbol{x}_k, 0\right) e^{i\Theta_k\left(\boldsymbol{x}_k, 0\right)}$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

- For $\left\langle f_q\left(\boldsymbol{x}_m, m\right), f_k\left(\boldsymbol{x}_n, n\right)\right\rangle = g\left(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n\right)$ to be true, this implies:

$$R_q\left(\boldsymbol{x}_q, m\right) R_k\left(\boldsymbol{x}_k, n\right) = R_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right)$$

$$\Theta_k\left(\boldsymbol{x}_k, n\right) - \Theta_q\left(\boldsymbol{x}_q, m\right) = \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right)$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

From previously

$$R_q\left(\boldsymbol{x}_q, m\right) R_k\left(\boldsymbol{x}_k, n\right) = R_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right)$$

$$\Theta_k\left(\boldsymbol{x}_k, n\right) - \Theta_q\left(\boldsymbol{x}_q, m\right) = \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right)$$

- Set $m = n$:

$$R_q\left(\boldsymbol{x}_q, m\right) R_k\left(\boldsymbol{x}_k, m\right) = R_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, 0\right) = R_q\left(\boldsymbol{x}_q, 0\right) R_k\left(\boldsymbol{x}_k, 0\right) = \|\boldsymbol{q}\|\|\boldsymbol{k}\|,$$

$$\Theta_k\left(\boldsymbol{x}_k, m\right) - \Theta_q\left(\boldsymbol{x}_q, m\right) = \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, 0\right) = \Theta_k\left(\boldsymbol{x}_k, 0\right) - \Theta_q\left(\boldsymbol{x}_q, 0\right) = \theta_k - \theta_q.$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

- $R_q\left(\boldsymbol{x}_q, m\right) R_k\left(\boldsymbol{x}_k, m\right) = \|\boldsymbol{q}\| \|\boldsymbol{k}\|$
- One possible solution for the magnitudes that doesn't depend on positional information at all:

$$R_q\left(\boldsymbol{x}_q, m\right) = R_q\left(\boldsymbol{x}_q, 0\right) = \|\boldsymbol{q}\|$$

$$R_k\left(\boldsymbol{x}_k, n\right) = R_k\left(\boldsymbol{x}_k, 0\right) = \|\boldsymbol{k}\|$$

- $R_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n - m\right) = R_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, 0\right) = \|\boldsymbol{q}\| \|\boldsymbol{k}\|$
- Now we just have to find $\Theta_{\{q,k,g\}}$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

From previously:

$$\Theta_k\left(\boldsymbol{x}_k, m\right) - \Theta_q\left(\boldsymbol{x}_q, m\right) = \theta_k - \theta_q$$

- Rearranging gives
  $\Theta_q(\boldsymbol{x}_q, m) - \theta_q = \Theta_k\left(\boldsymbol{x}_k, m\right) - \theta_k$
- Observation:
  - Symmetry means Θ_q, Θ_k can take on similar functional forms
  - $\Theta_{\{q,k\}}\left(\boldsymbol{x}_{\{q,k\}}, m\right) - \theta_{\{q,k\}}$ is a function of m
- Choose:
  $\Theta_{\{q,k\}}\left(\boldsymbol{x}_{\{q,k\}}, m\right) = \phi(m) + \theta_{\{q,k\}}$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

From previously:

$$\Theta_k\left(\boldsymbol{x}_k, n\right) - \Theta_q\left(\boldsymbol{x}_q, m\right) = \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m\right)$$

We chose

$$\Theta_{\{q,k\}}\left(\boldsymbol{x}_{\{q,k\}}, m\right) = \phi(m) + \theta_{\{q,k\}}$$

- Substitute $n = m + 1$:

$$\Theta_k\left(\boldsymbol{x}_k, m+1\right) - \Theta_q\left(\boldsymbol{x}_q, m\right)$$

$$= \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, 1\right)$$

$$= \phi(m+1) - \phi(m) + \theta_q - \theta_k$$

- Rearranging:

$$\phi(m+1) - \phi(m) = \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, 1\right) + \theta_q - \theta_k$$

- RHS is constant with respect to m!

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

- This induces an arithmetic progression, for some γ, θ of our choosing:
  $\phi(0) = \gamma$
- $\phi(m) = m\theta + \gamma$
- So overall, our angular component is

$$\Theta_{\{q,k\}}\left(\boldsymbol{x}_{\{q,k\}}, m\right) = m\theta + \gamma + \theta_{\{q,k\}}$$

From previously:

$$\phi(m+1) - \phi(m) = \Theta_g\left(\boldsymbol{x}_q, \boldsymbol{x}_k, 1\right) + \theta_q - \theta_k$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

- Putting it all together:

$$f_q\left(\boldsymbol{x}_q, m\right) = \|\boldsymbol{q}\| e^{i\theta_q + m\theta + \gamma} = \boldsymbol{q} e^{i(m\theta+\gamma)}$$

$$f_k\left(\boldsymbol{x}_k, n\right) = \|\boldsymbol{k}\| e^{i\theta_k + n\theta + \gamma} = \boldsymbol{k} e^{i(n\theta+\gamma)}$$

- Choose $\gamma = 0$, and set initial conditions to be similar to setup in Attention Is All You Need:

$$\boldsymbol{q} = \boldsymbol{W}_q \boldsymbol{x}_n, \boldsymbol{k} = \boldsymbol{W}_k \boldsymbol{x}_n$$

- This gives

$$f_q\left(\boldsymbol{x}_m, m\right) = \left(\boldsymbol{W}_q \boldsymbol{x}_m\right) e^{im\theta}$$

$$f_k\left(\boldsymbol{x}_n, n\right) = \left(\boldsymbol{W}_k \boldsymbol{x}_n\right) e^{in\theta}$$

From previously:

$$R_q\left(\boldsymbol{x}_q, m\right) = \|\boldsymbol{q}\|$$

$$R_k\left(\boldsymbol{x}_k, n\right) = \|\boldsymbol{k}\|$$

$$\Theta_{\{q,k\}}\left(\boldsymbol{x}_{\{q,k\}}, m\right) = m\theta + \gamma + \theta_{\{q,k\}}$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

From previously:

$$f_q\left(\boldsymbol{x}_m, m\right) = \left(\boldsymbol{W}_q \boldsymbol{x}_m\right) e^{im\theta}$$

$$f_k\left(\boldsymbol{x}_n, n\right) = \left(\boldsymbol{W}_k \boldsymbol{x}_n\right) e^{in\theta}$$

- Use rotation matrix to capture rotation:
- $\begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \mathbf{W}_q \boldsymbol{x}_m$
- To extend to d (even) dimensions, repeat this for each pair of coordinates with $\theta_i = 10000^{-2(i-1)/d}$ (similar to Attention is All You Need):

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

RoPE Derivation

- Finally, we get that the dot-product only gives us relative positional information:
- $\boldsymbol{q}_m^\top \boldsymbol{k}_n = \left(\boldsymbol{R}_{\Theta,m}^d \boldsymbol{W}_q \boldsymbol{x}_m\right)^\top \left(\boldsymbol{R}_{\Theta,n}^d \boldsymbol{W}_k \boldsymbol{x}_n\right) = \boldsymbol{x}^\top \boldsymbol{W}_q R_{\Theta,n-m}^d \boldsymbol{W}_k \boldsymbol{x}_n$
- In each coordinate i, rotating anti-clockwise by $n\theta_i$, then rotating clockwise by $m\theta_i$, for overall rotation of $(n-m)\theta_i$
- Shows that sinusoidal intuition from Attention Is All You Need is correct, but multiplying instead of adding gives a much cleaner formulation!

RoFormer: Enhanced Transformer with Rotary Position Embedding (Su et al., 2021)

Thank you!
Q&A