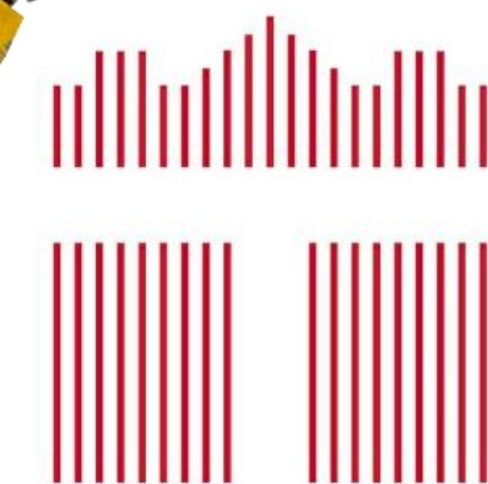
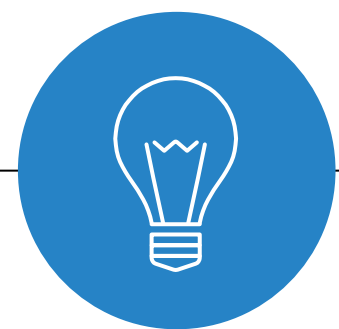


# *Applied Deep Learning*



# Transformer

March 29th, 2021 <http://adl.miulab.tw>



**National  
Taiwan  
University**  
國立臺灣大學

2

# Sequence Encoding

## Basic Attention

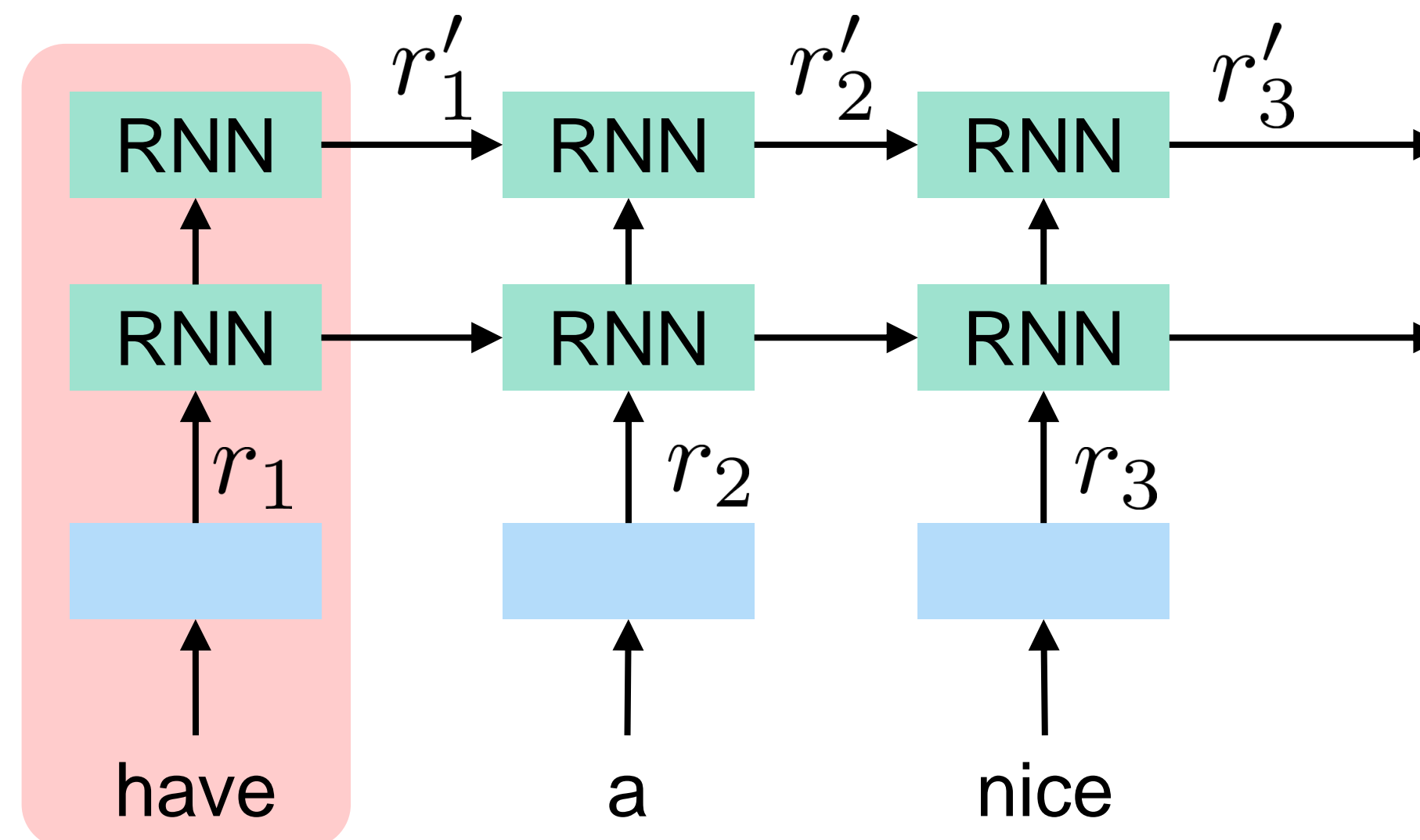
# Representations of Variable Length Data

- Input: word sequence, image pixels, audio signal, click logs
- Property: continuity, temporal, importance distribution
- Example
  - ✓ Basic combination: average, sum
  - ✓ Neural combination: network architectures should consider input domain properties
    - CNN (convolutional neural network)
    - RNN (recurrent neural network): temporal information

Network architectures should consider the input domain properties

# 4 Recurrent Neural Networks

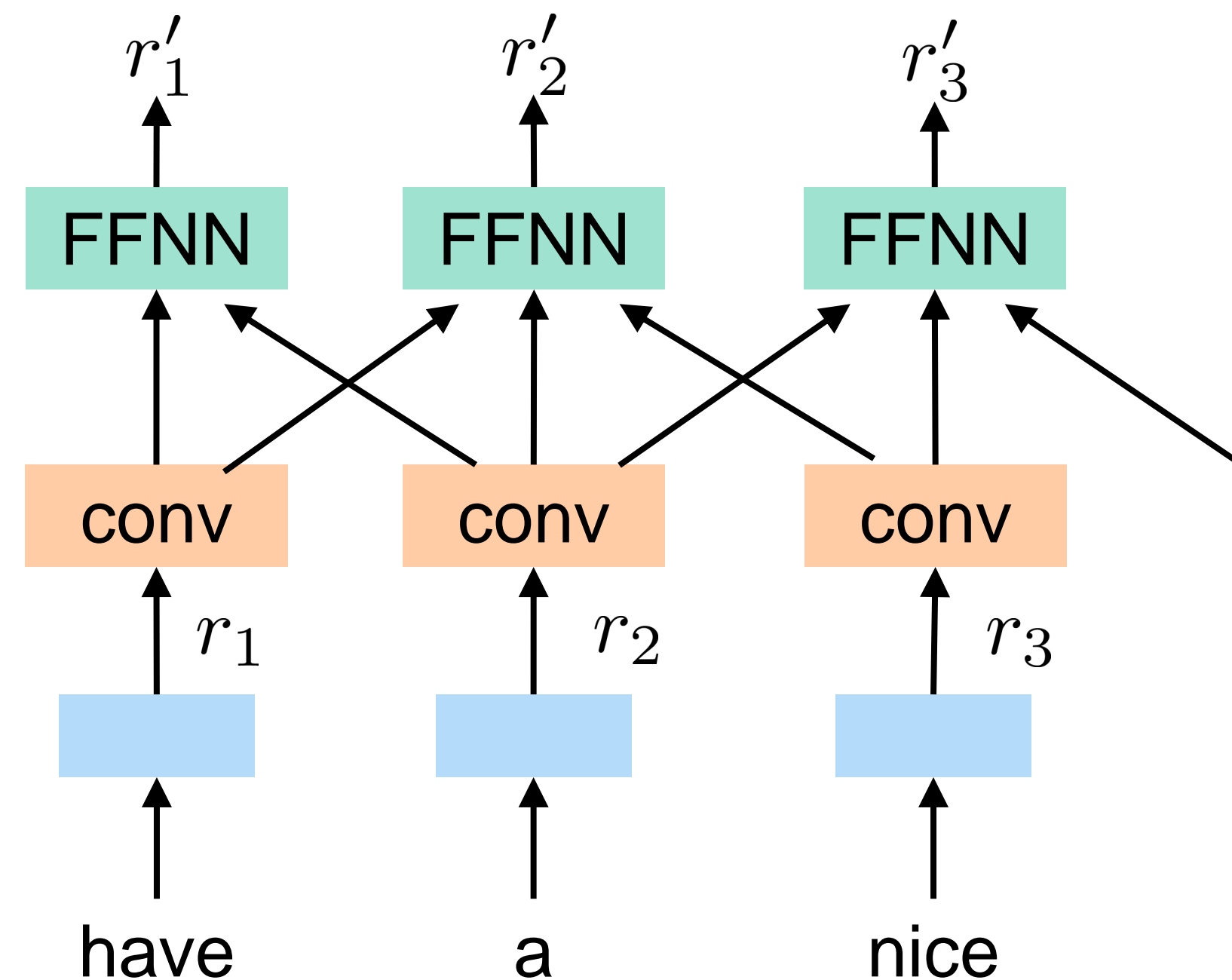
- Learning variable-length representations
  - ✓ Fit for sentences and sequences of values
- Sequential computation makes parallelization difficult
- No explicit modeling of long and short range dependencies



5

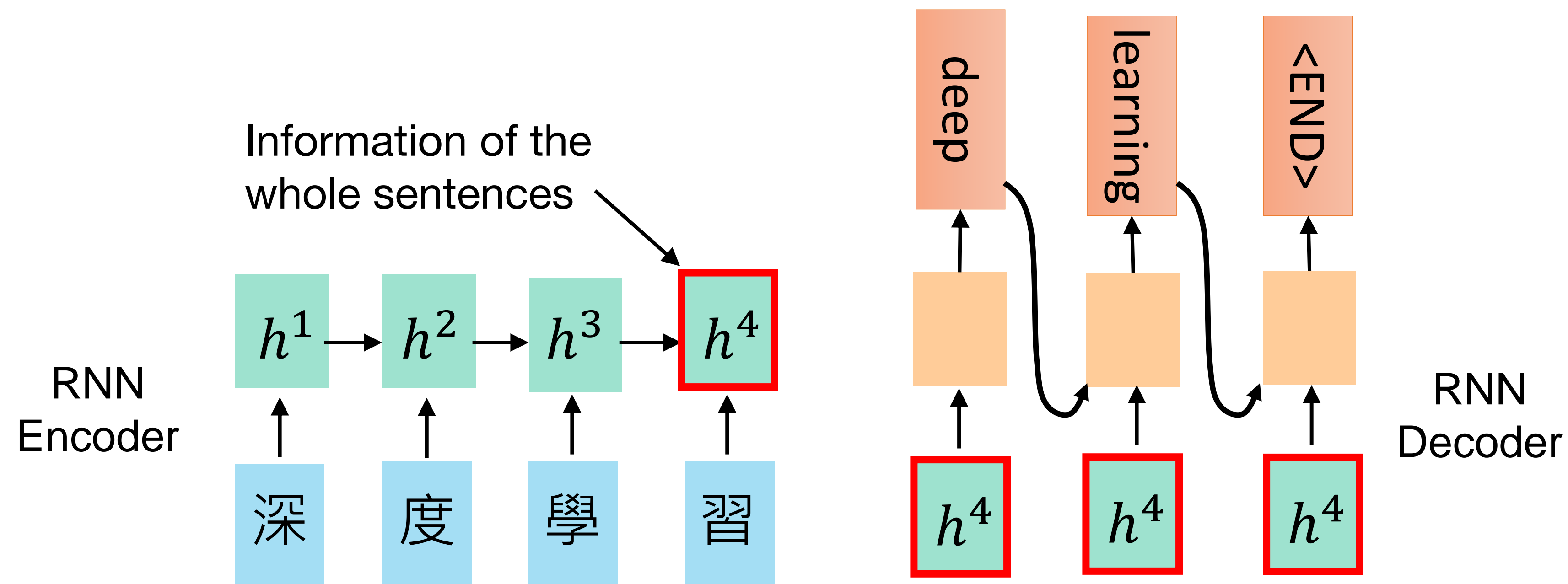
# Convolutional Neural Networks

- Easy to parallelize
- Exploit local dependencies
  - ✓ **Long-distance** dependencies require many layers

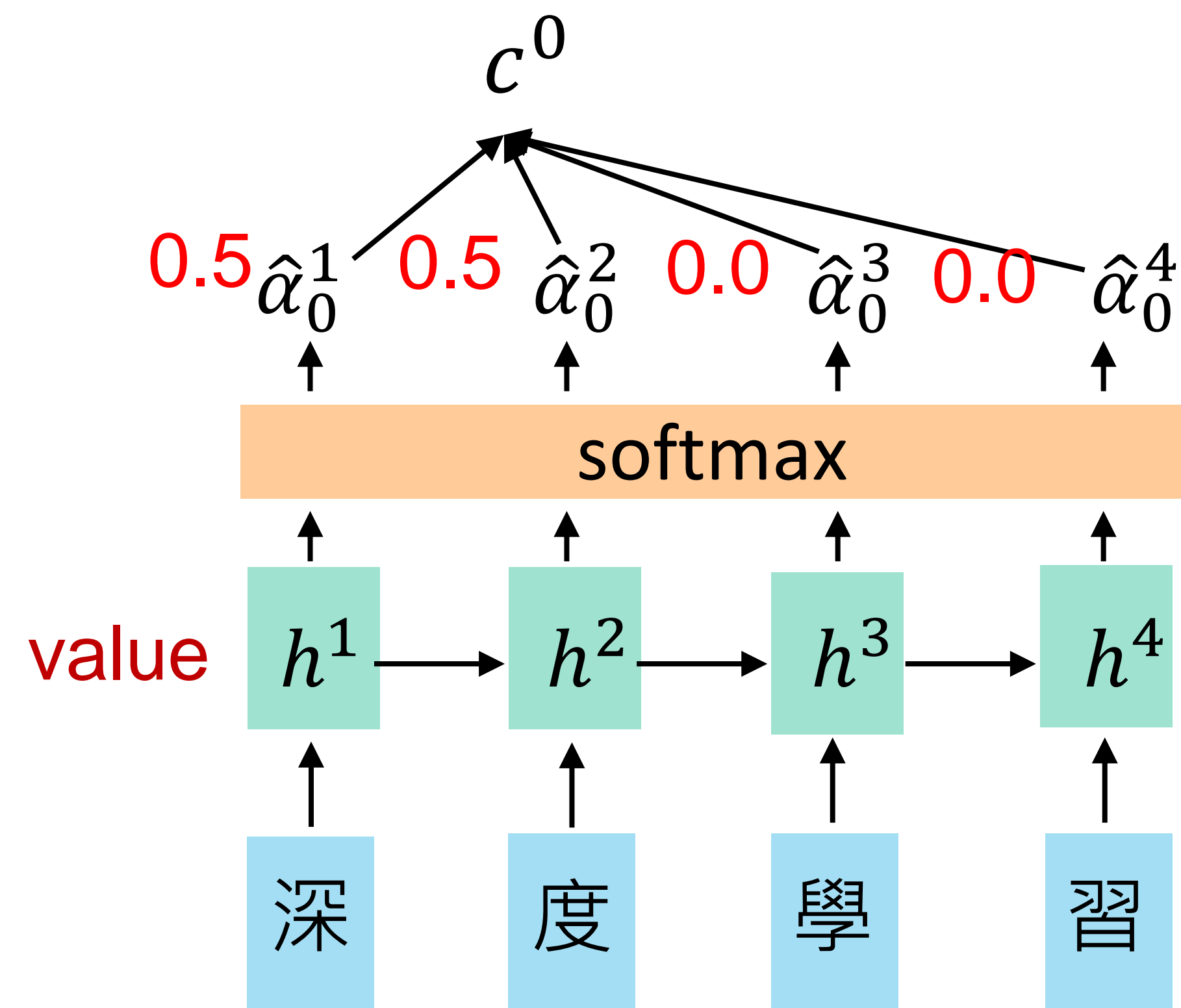
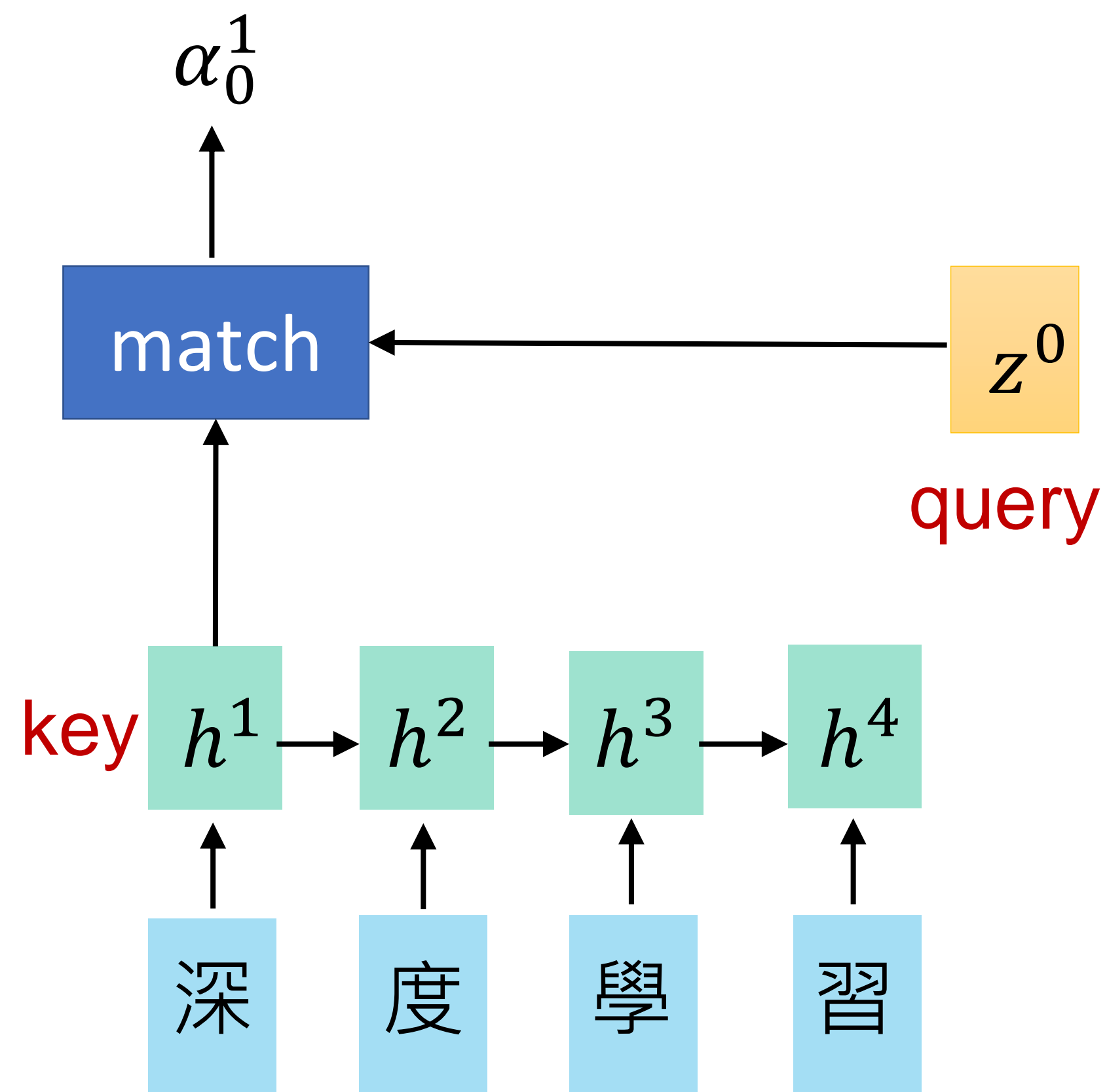


# Attention

- Encoder-decoder model is important in NMT
- RNNs need **attention mechanism** to handle long dependencies
- Attention allows us to access any state



# Basic Attention



# Dot-Product Attention

- Input: a query  $q$  and a set of key-value ( $k$ - $v$ ) pairs to an output
- Output: weighted sum of values

Inner product of  
query and corresponding key

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} v_i$$

- ✓ Query  $q$  is a  $d_k$ -dim vector
- ✓ Key  $k$  is a  $d_k$ -dim vector
- ✓ Value  $v$  is a  $d_v$ -dim vector



# Dot-Product Attention in Matrix

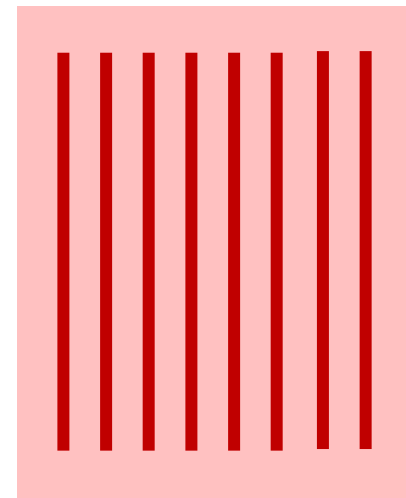
- Input: *multiple* queries  $q$  and a set of key-value ( $k$ - $v$ ) pairs to an output
- Output: a set of weighted sum of values

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} v_i$$

$$A(Q, K, V) = \text{softmax}(QK^T)V$$

$$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$$

softmax  
row-wise



$$= [|Q| \times d_v]$$

10

# Sequence Encoding

## Self-Attention

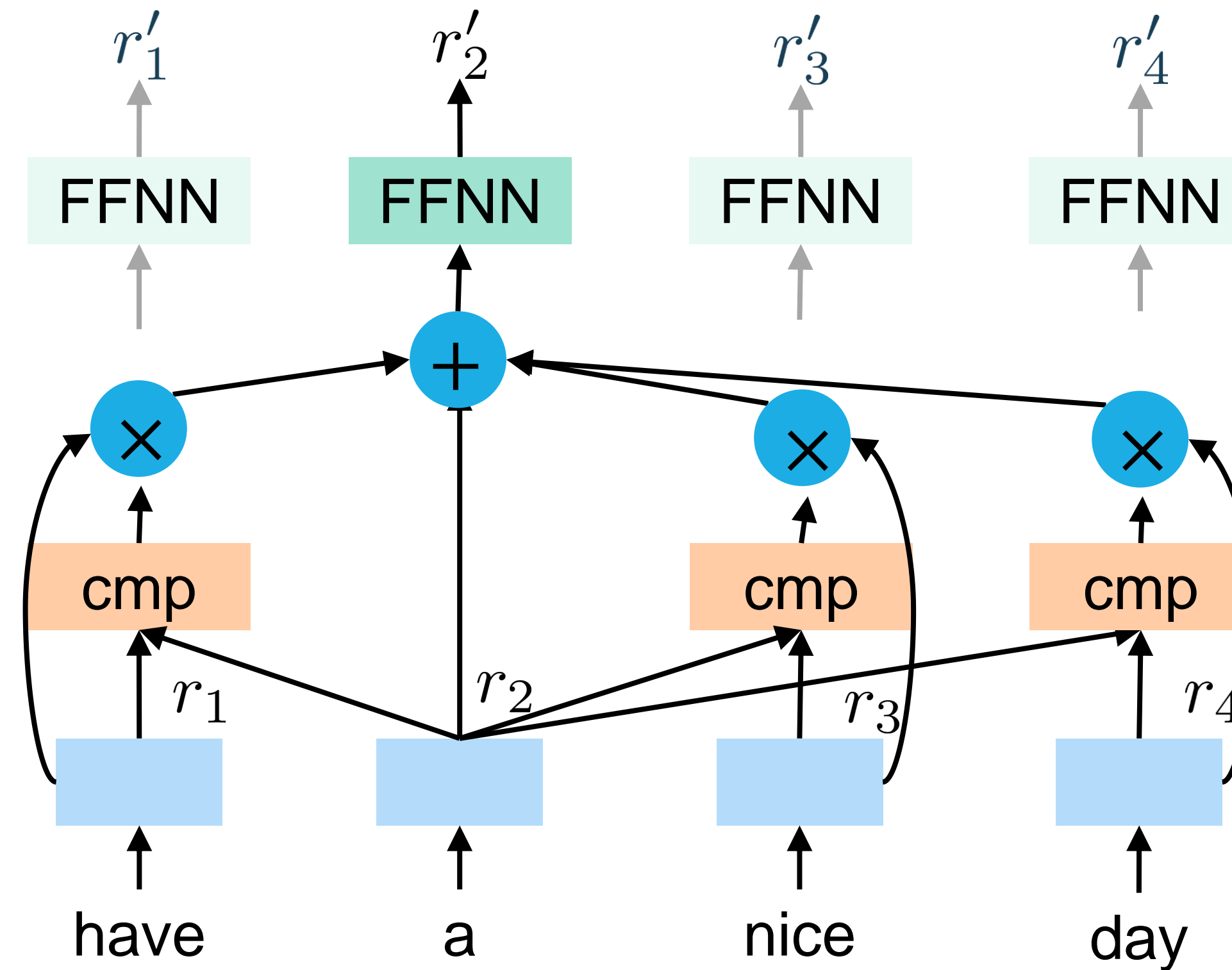
# Attention

- ⦿ Encoder-decoder model is important in NMT
- ⦿ RNNs need **attention mechanism** to handle long dependencies
- ⦿ Attention allows us to access any state

Using attention to replace recurrence architectures

# 12 Self-Attention

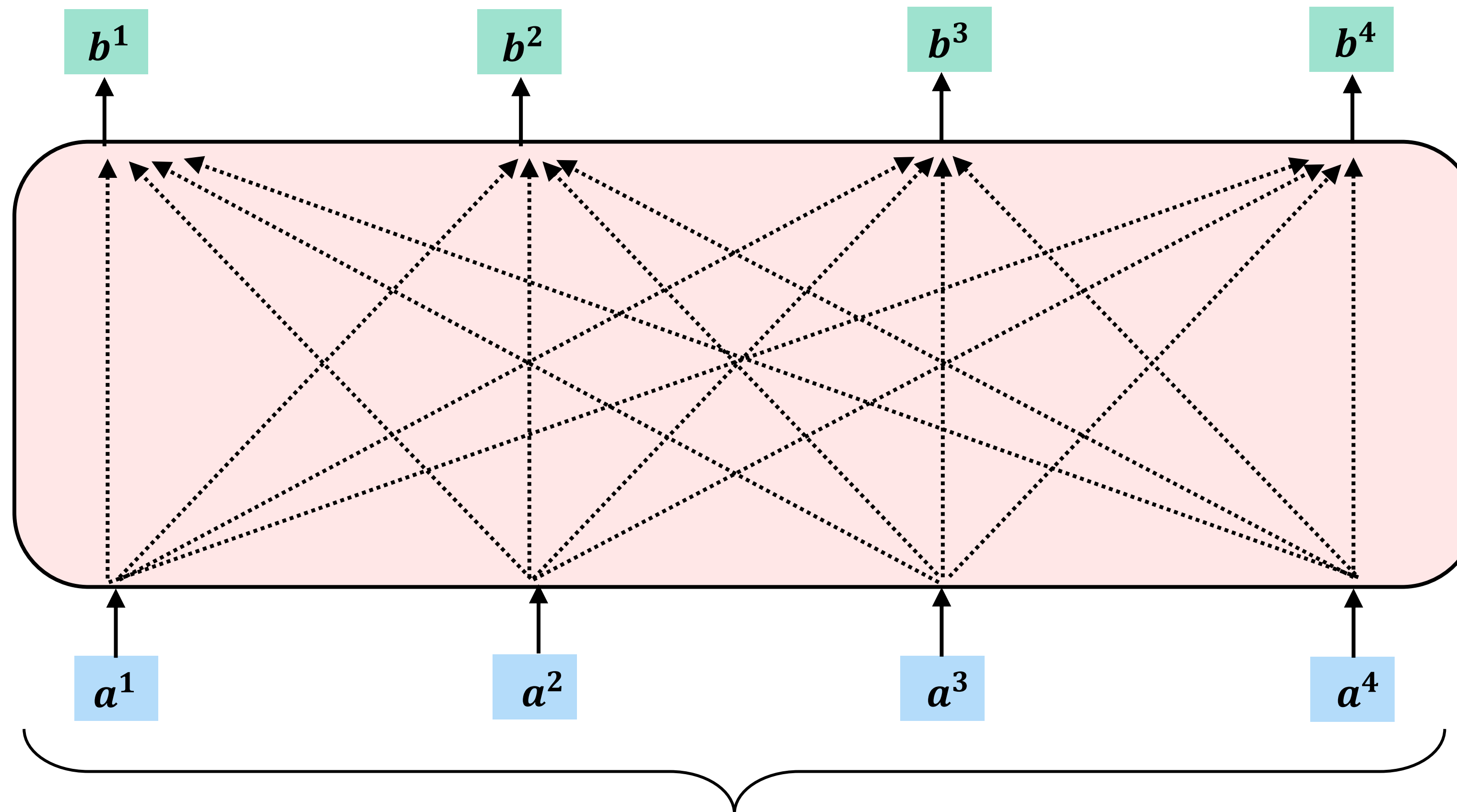
- Constant “path length” between two positions
- Easy to parallelize



13

# Self-Attention

- Constant “path length” between two positions
- Easy to parallelize

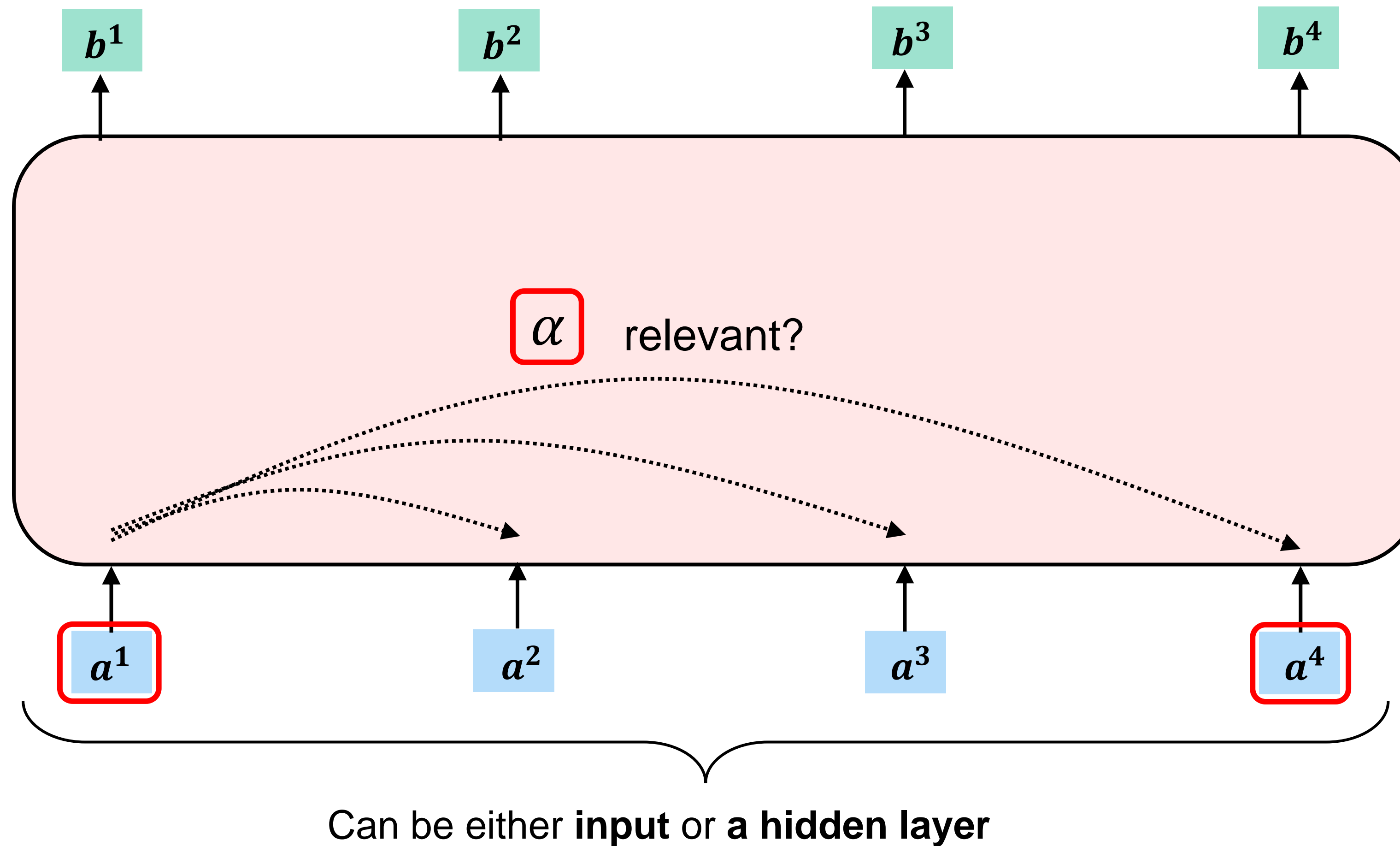


Can be either **input** or a **hidden layer**

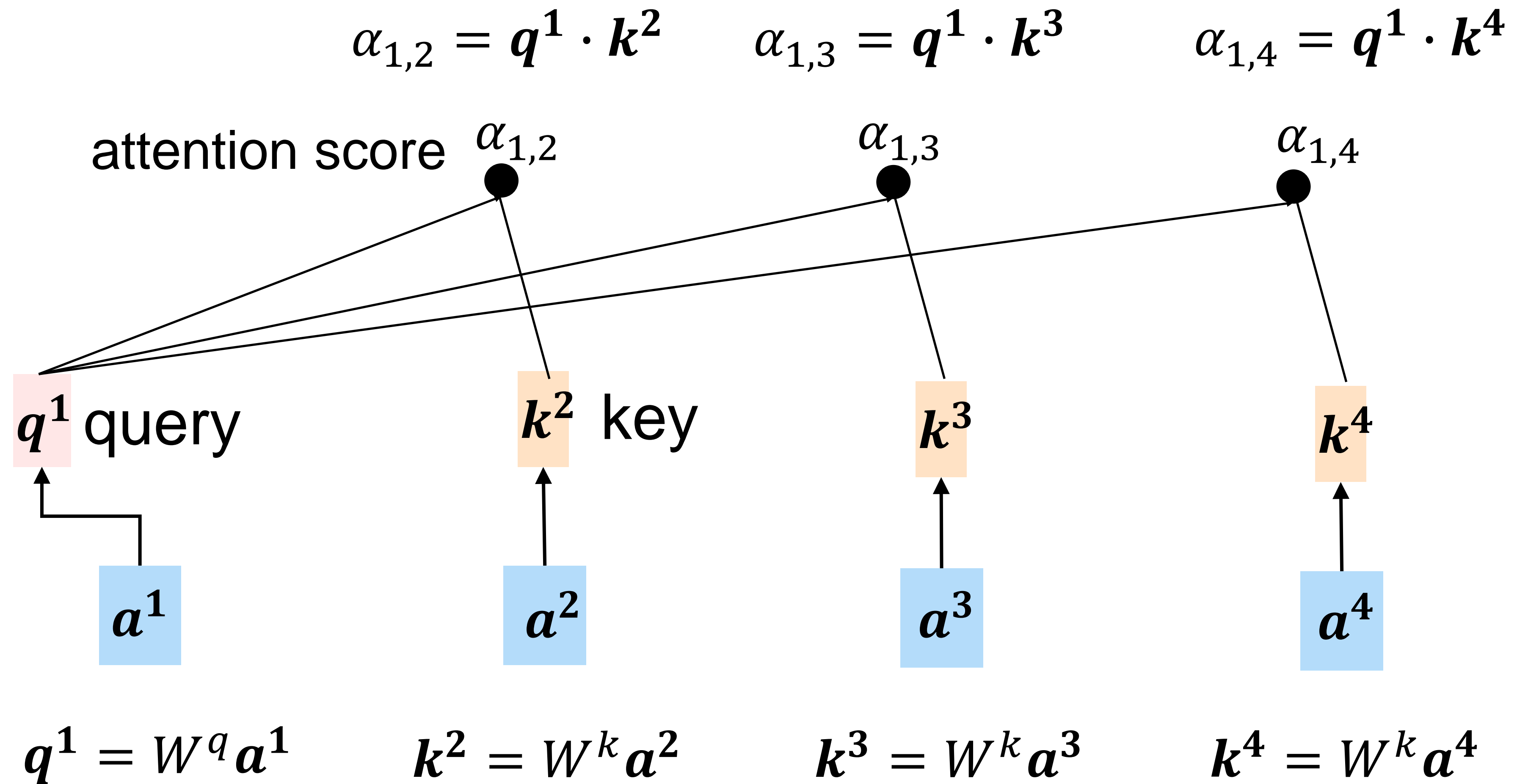
14

# Self-Attention

- Constant “path length” between two positions
- Easy to parallelize

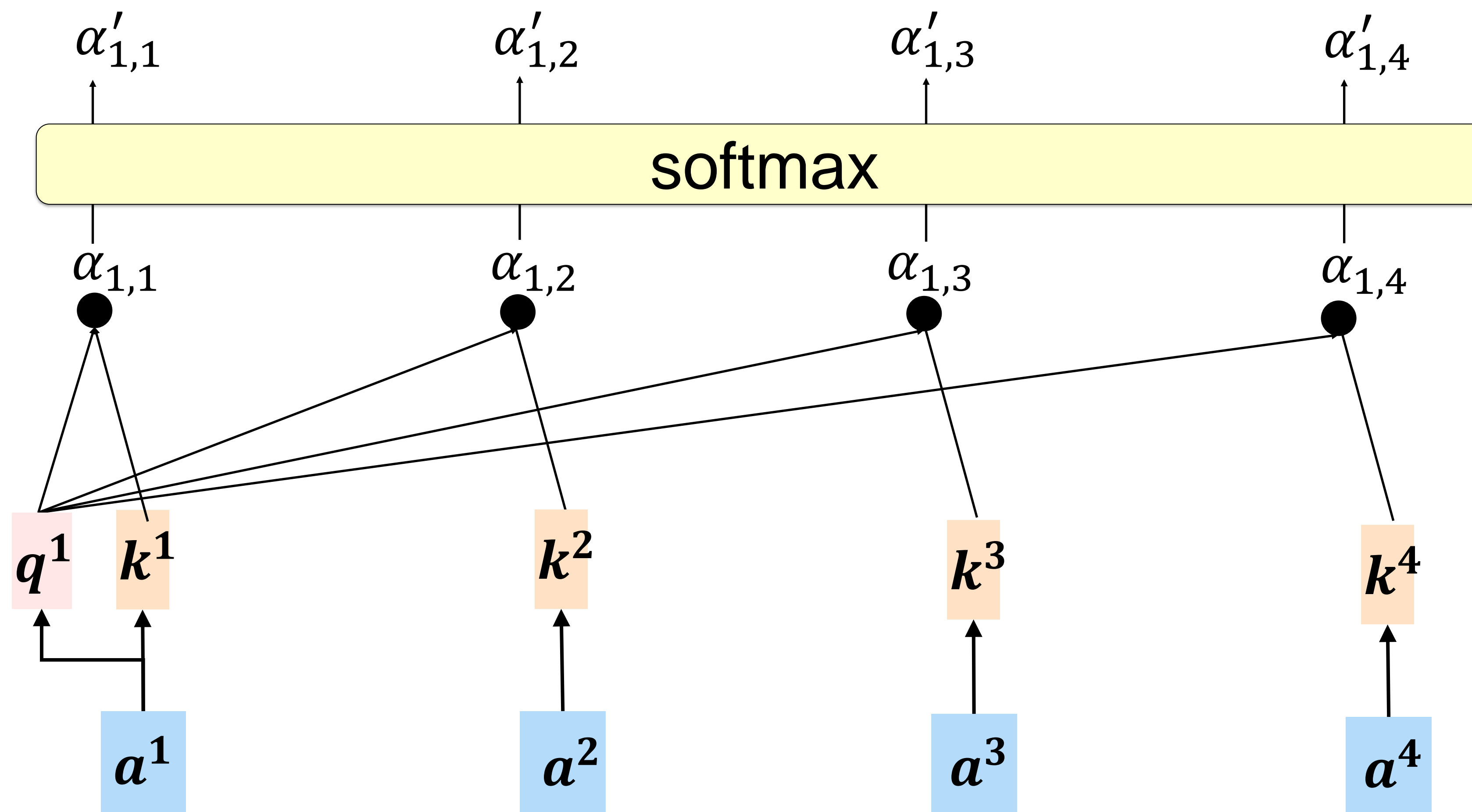


# Self-Attention



# Self-Attention

$$\alpha'_{1,i} = e^{\alpha_{1,i}} / \sum_j e^{\alpha_{1,j}}$$



$$q^1 = W^q a^1$$

$$k^1 = W^k a^1$$

$$k^2 = W^k a^2$$

$$k^3 = W^k a^3$$

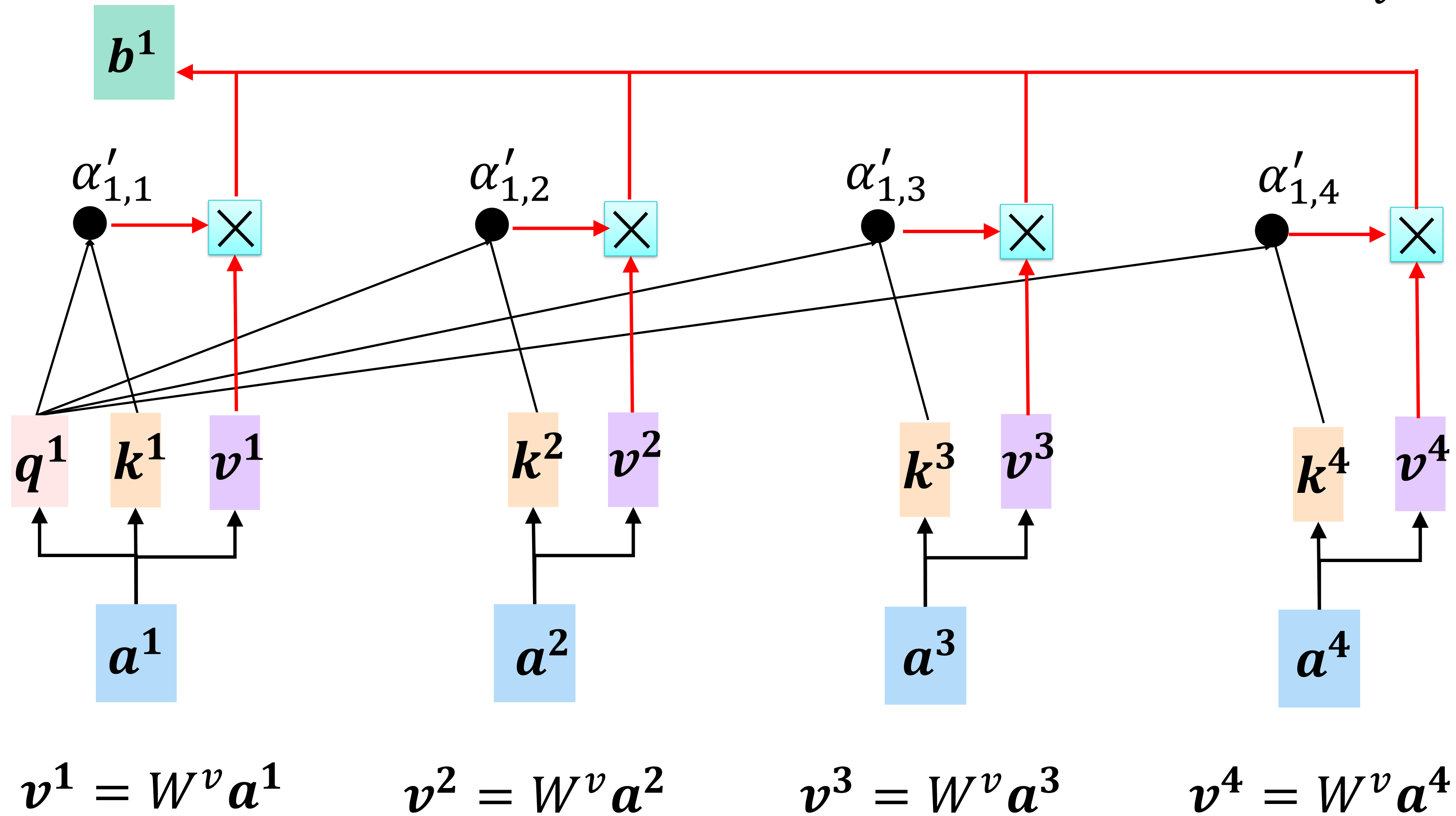
$$k^4 = W^k a^4$$



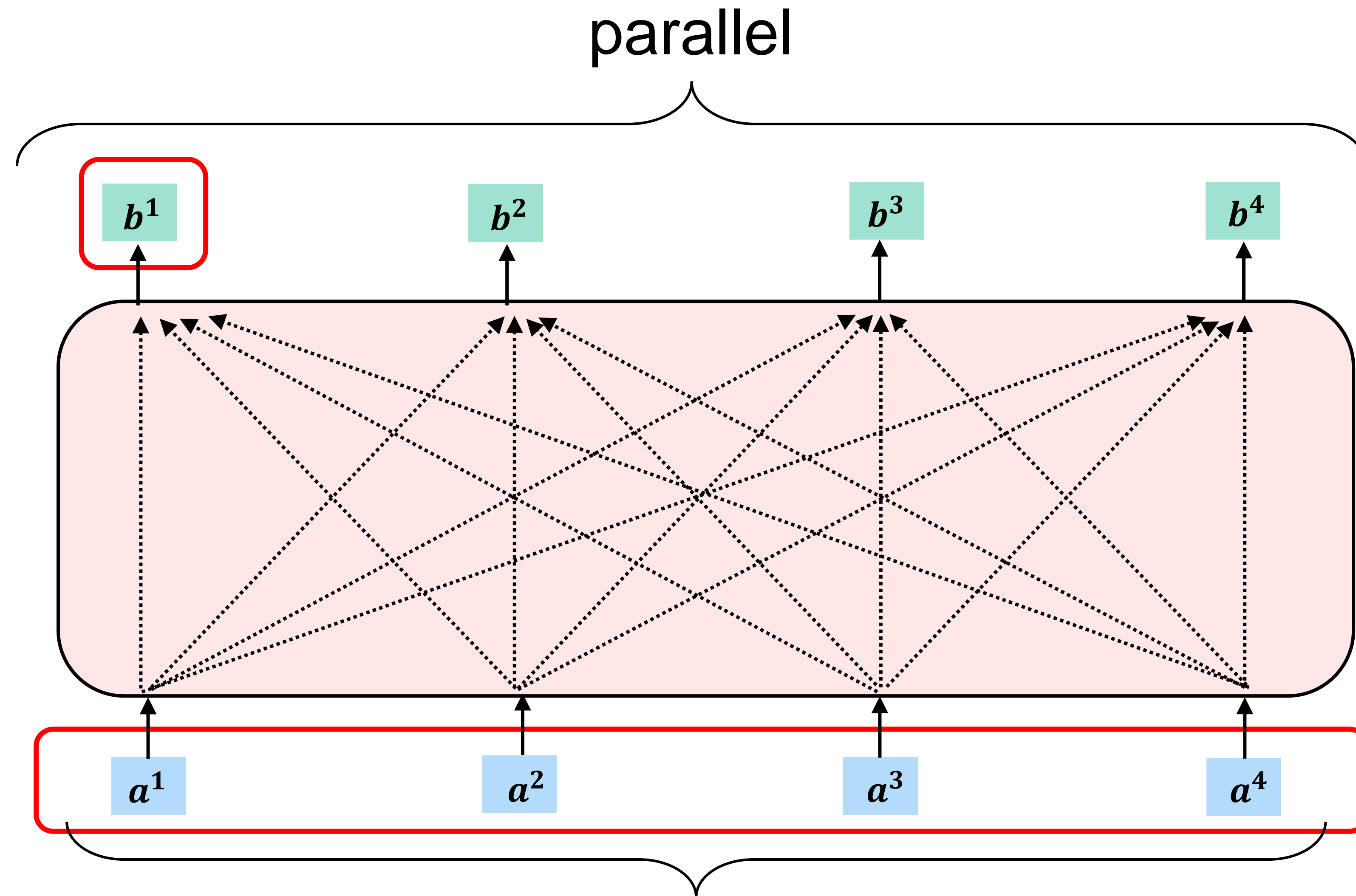
# Self-Attention

extract information based  
on attention scores

$$b^1 = \sum_i \alpha'_{1,i} v^i$$



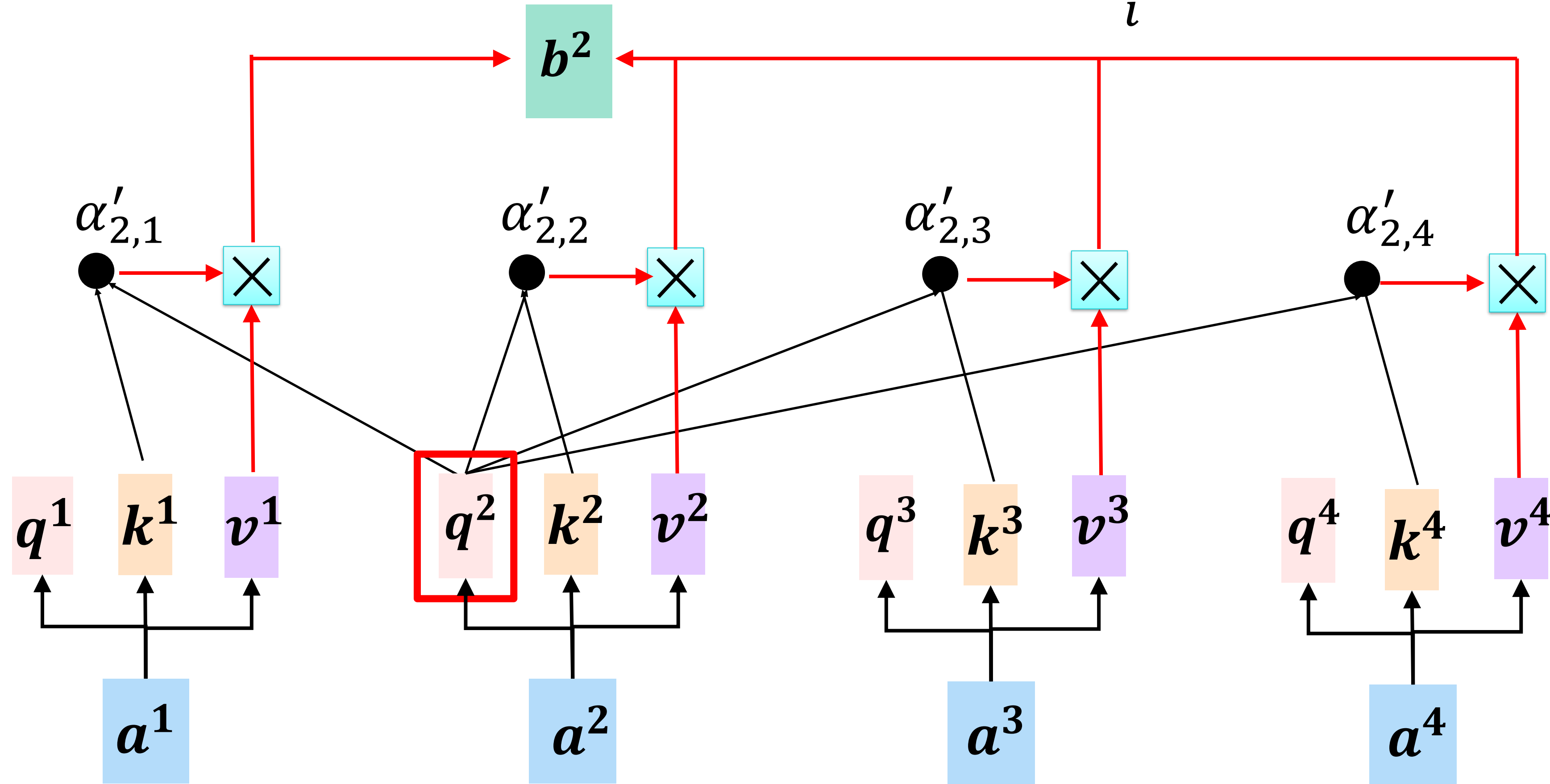
# Self-Attention



Can be either **input** or a **hidden layer**

# Self-Attention

$$b^2 = \sum_i \alpha'_{2,i} v^i$$



# Self-Attention

$$q^i = W^q a^i$$

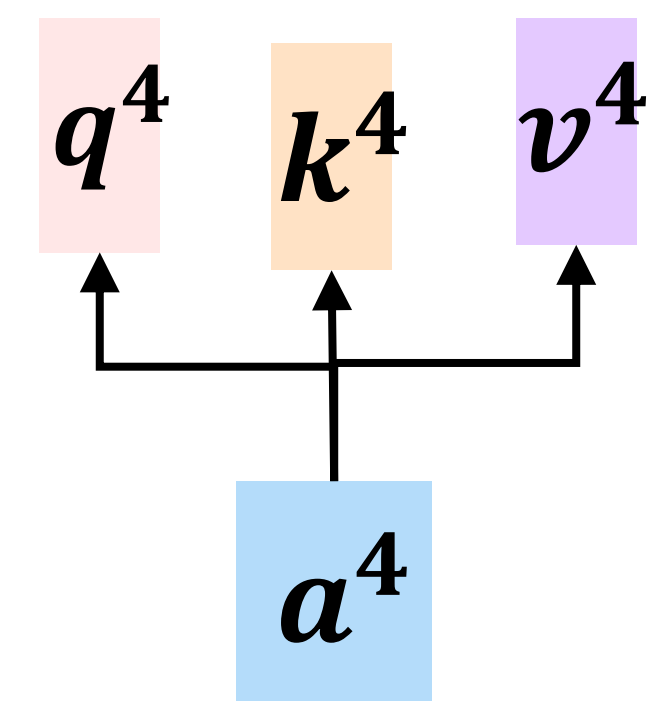
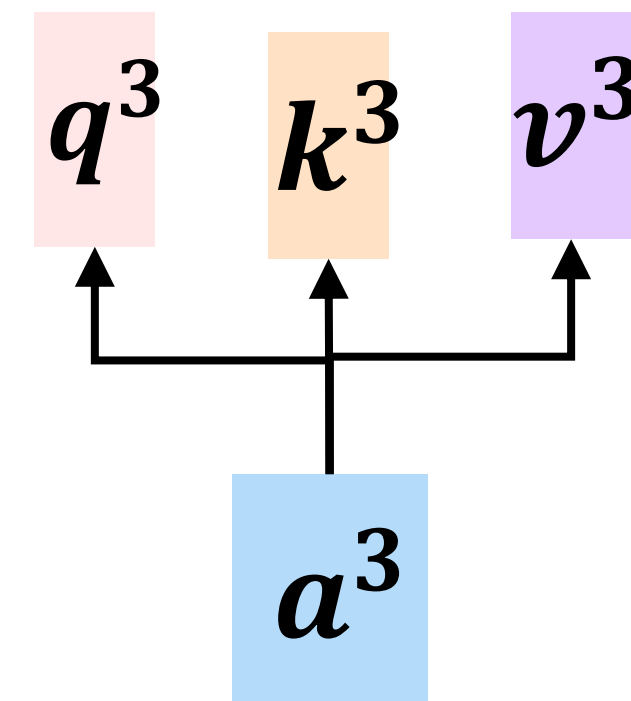
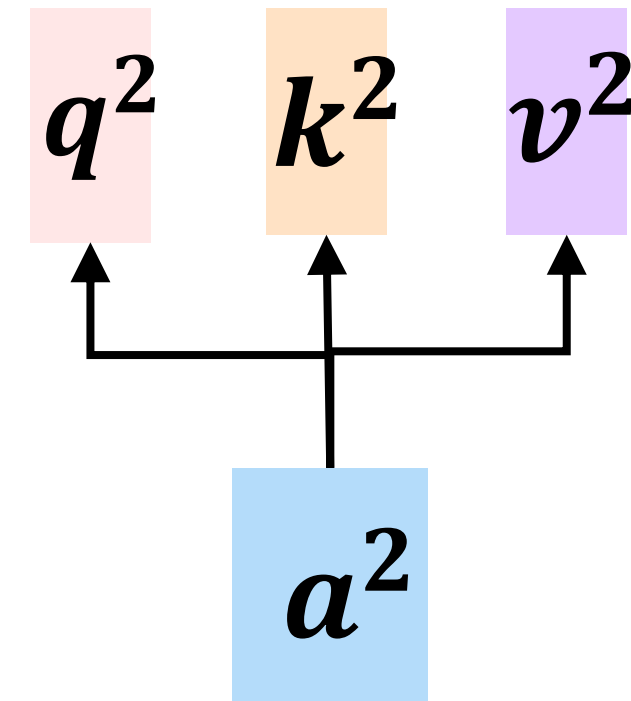
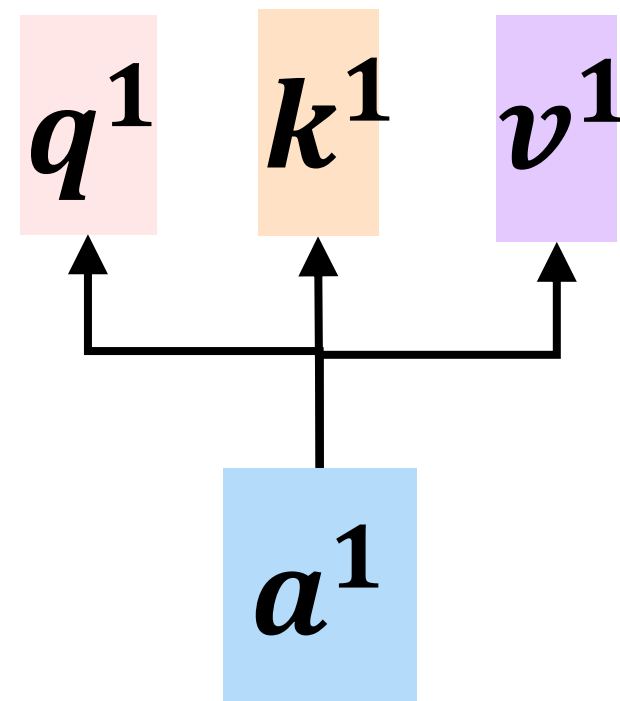
$$\begin{matrix} q^1 & q^2 & q^3 & q^4 \\ \hline W^q & a^1 & a^2 & a^3 & a^4 \\ \hline Q & & & & I \end{matrix}$$

$$k^i = W^k a^i$$

$$\begin{matrix} k^1 & k^2 & k^3 & k^4 \\ \hline W^k & a^1 & a^2 & a^3 & a^4 \\ \hline K & & & & I \end{matrix}$$

$$v^i = W^v a^i$$

$$\begin{matrix} v^1 & v^2 & v^3 & v^4 \\ \hline W^v & a^1 & a^2 & a^3 & a^4 \\ \hline V & & & & I \end{matrix}$$

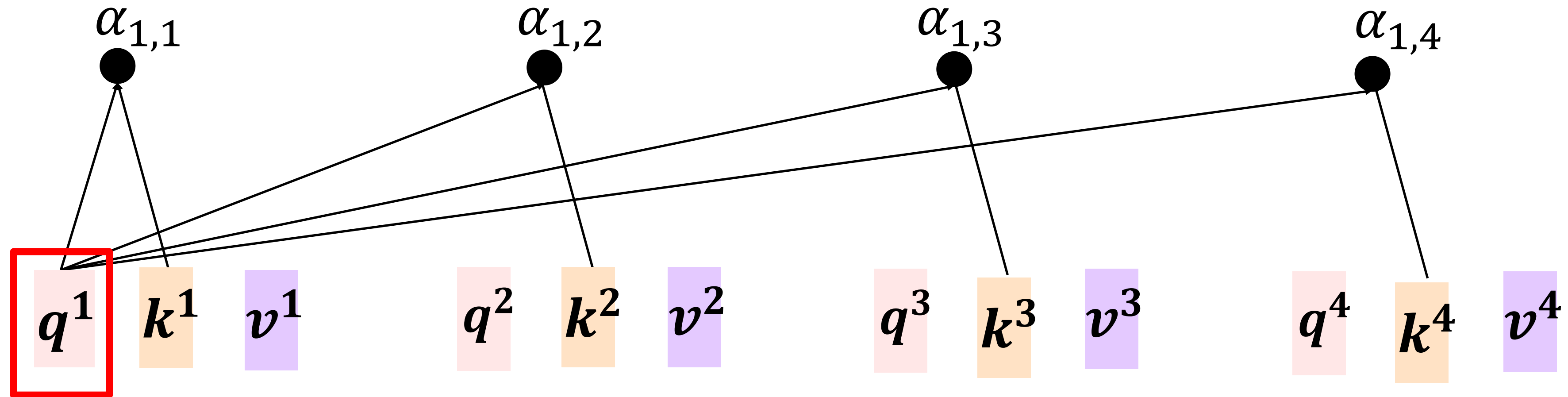


# Self-Attention

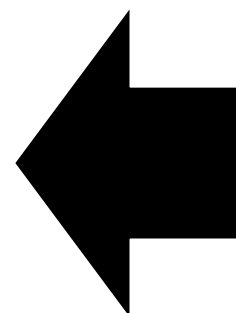
$$\alpha_{1,1} = k^1 q^1 \quad \alpha_{1,2} = k^2 q^1$$

$$\alpha_{1,3} = k^3 q^1 \quad \alpha_{1,4} = k^4 q^1$$

$$\begin{bmatrix} \alpha_{1,1} \\ \alpha_{1,2} \\ \alpha_{1,3} \\ \alpha_{1,4} \end{bmatrix} = \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} q^1$$



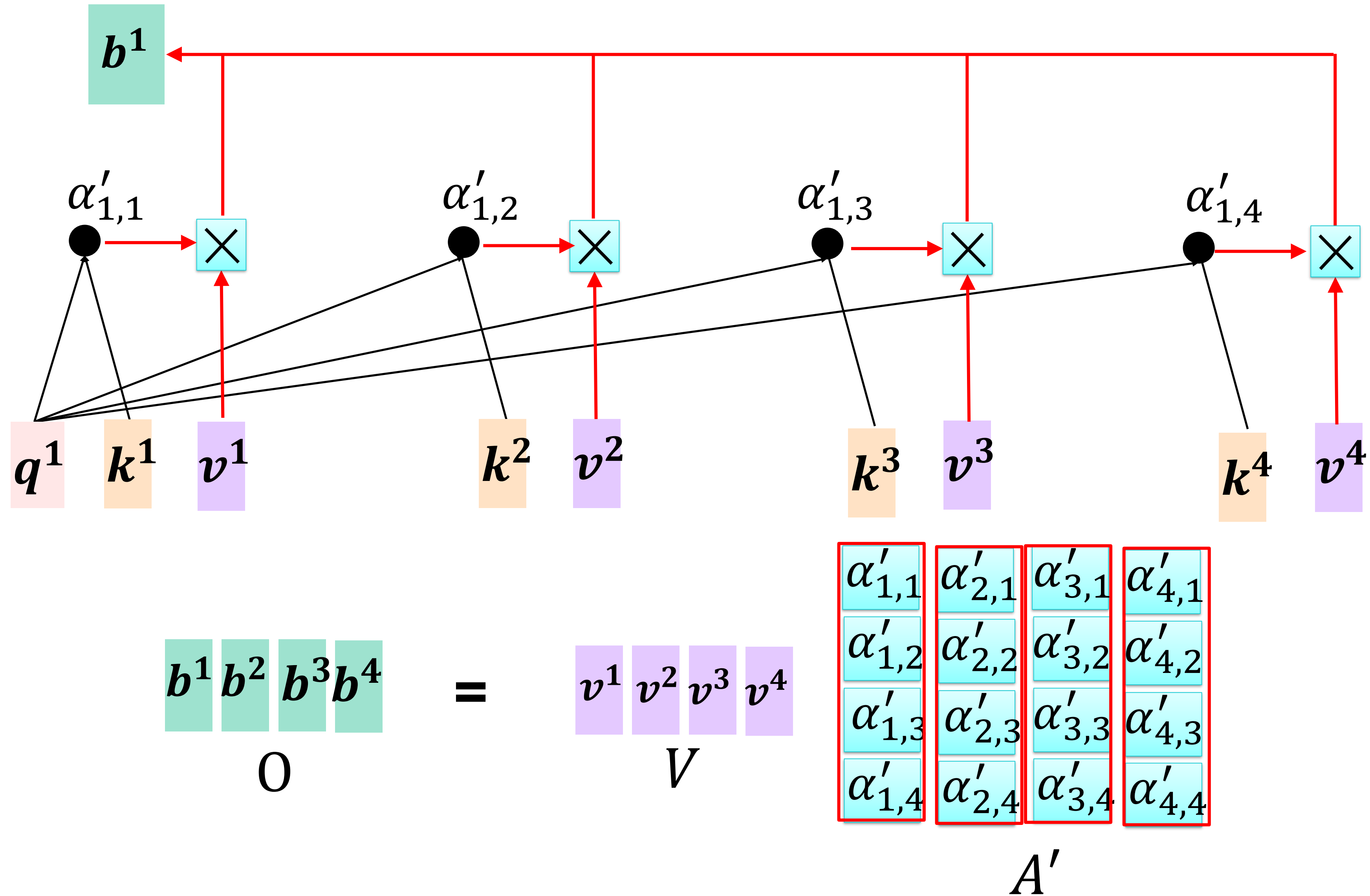
$$A' = \begin{bmatrix} \alpha'_{1,1} & \alpha'_{2,1} & \alpha'_{3,1} & \alpha'_{4,1} \\ \alpha'_{1,2} & \alpha'_{2,2} & \alpha'_{3,2} & \alpha'_{4,2} \\ \alpha'_{1,3} & \alpha'_{2,3} & \alpha'_{3,3} & \alpha'_{4,3} \\ \alpha'_{1,4} & \alpha'_{2,4} & \alpha'_{3,4} & \alpha'_{4,4} \end{bmatrix}$$



$$A = \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & \alpha_{4,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & \alpha_{4,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & \alpha_{4,4} \end{bmatrix}$$

$$= \begin{bmatrix} k^1 \\ k^2 \\ k^3 \\ k^4 \end{bmatrix} \begin{bmatrix} q^1 & q^2 & q^3 & q^4 \end{bmatrix} = K^T Q$$

# Self-Attention



# Self-Attention

$$\begin{aligned} Q &= W^q I \\ K &= W^k I \\ V &= W^v I \end{aligned}$$

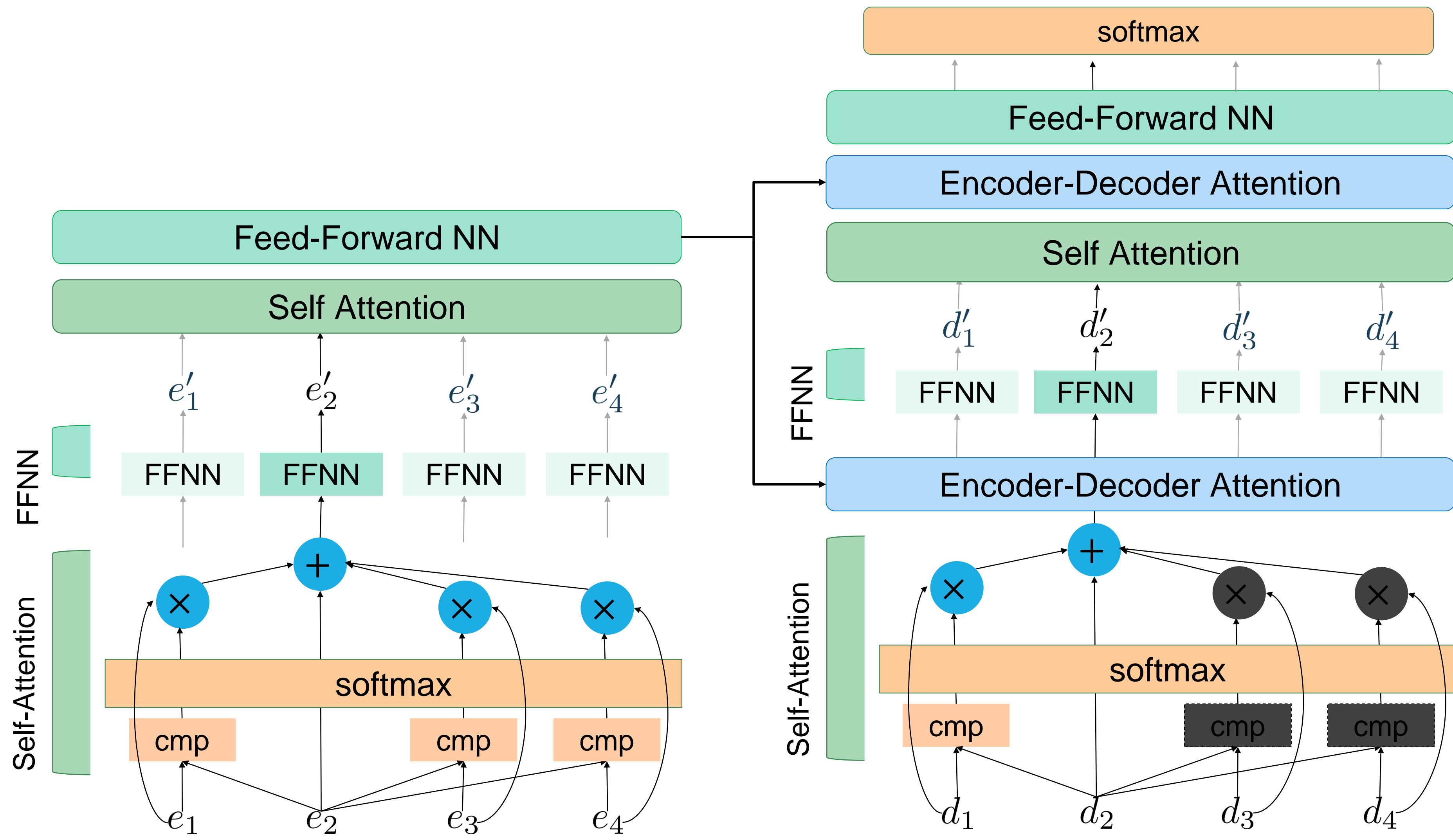
Parameters to be learned

$$A' \leftarrow A = K^T Q$$

Attention Matrix

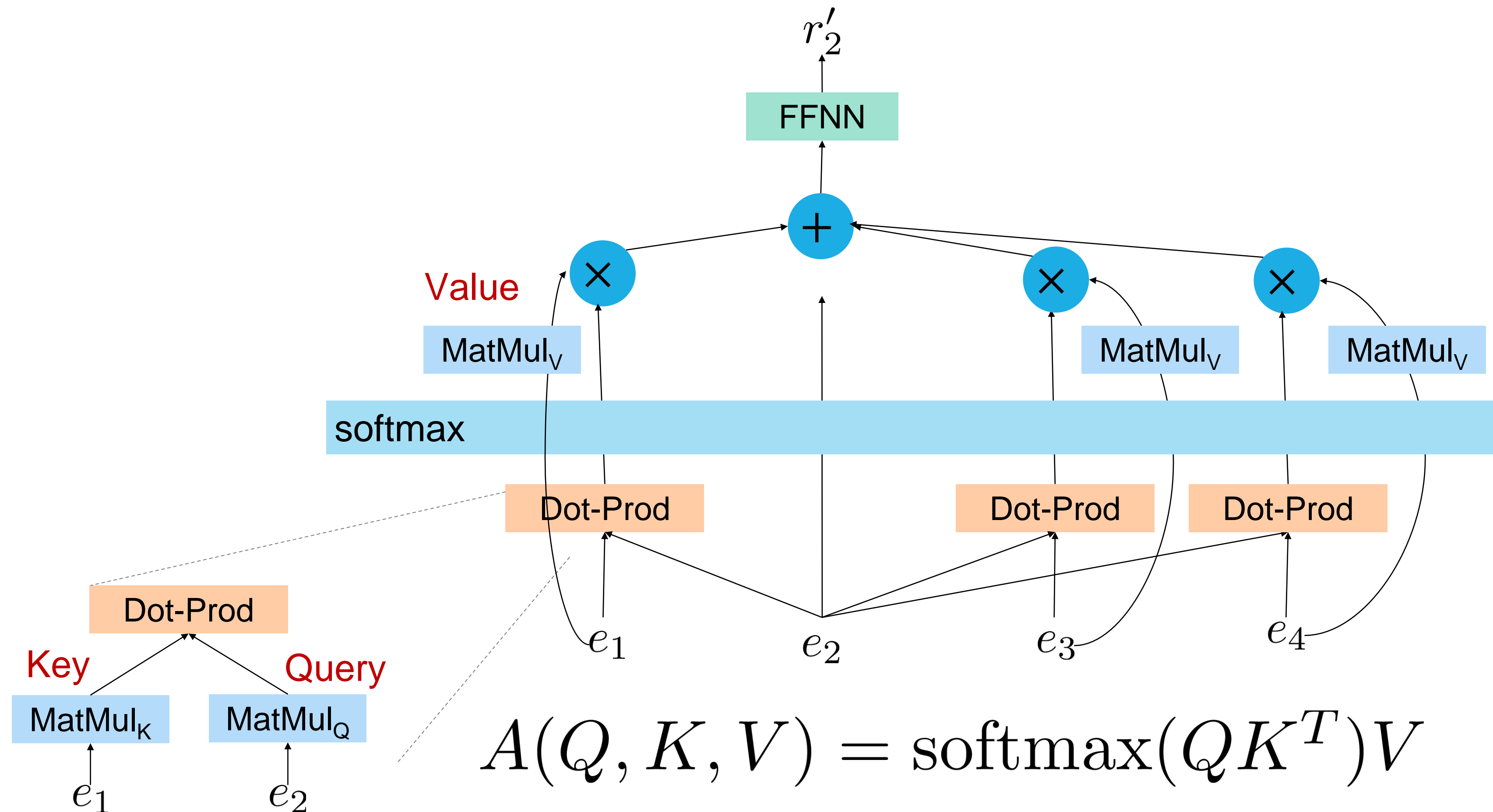
$$O = V A'$$

# Transformer Idea

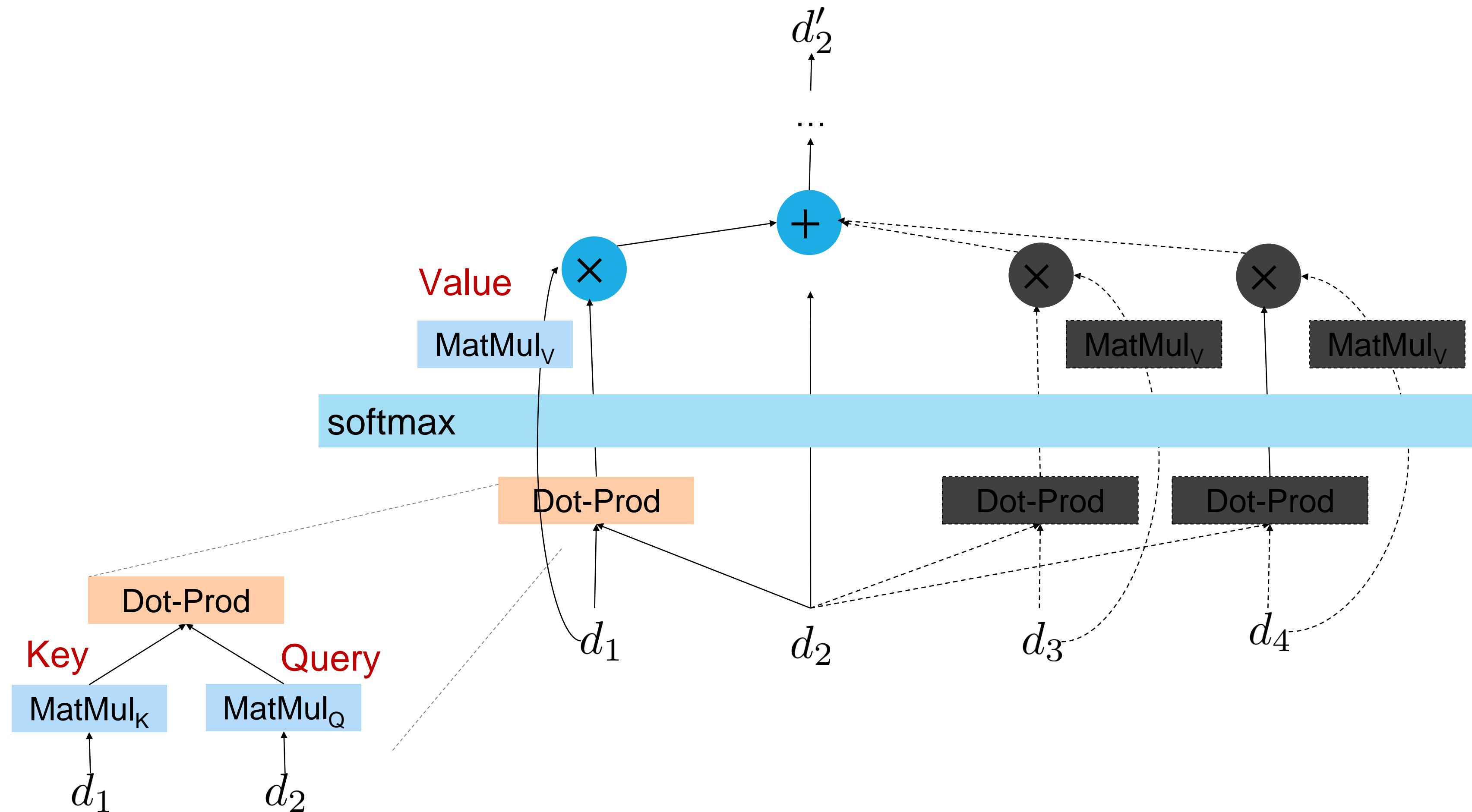




# Encoder Self-Attention (Vaswani+, 2017)



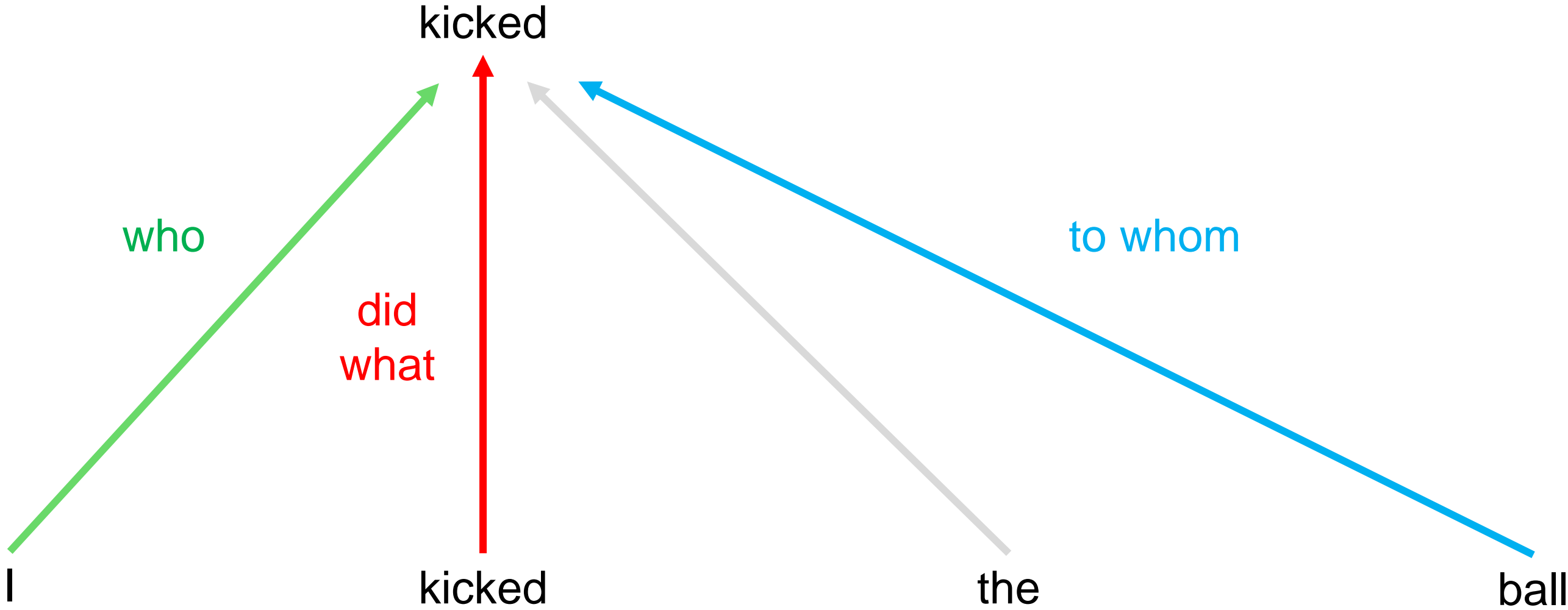
# Decoder Self-Attention (Vaswani+, 2017)



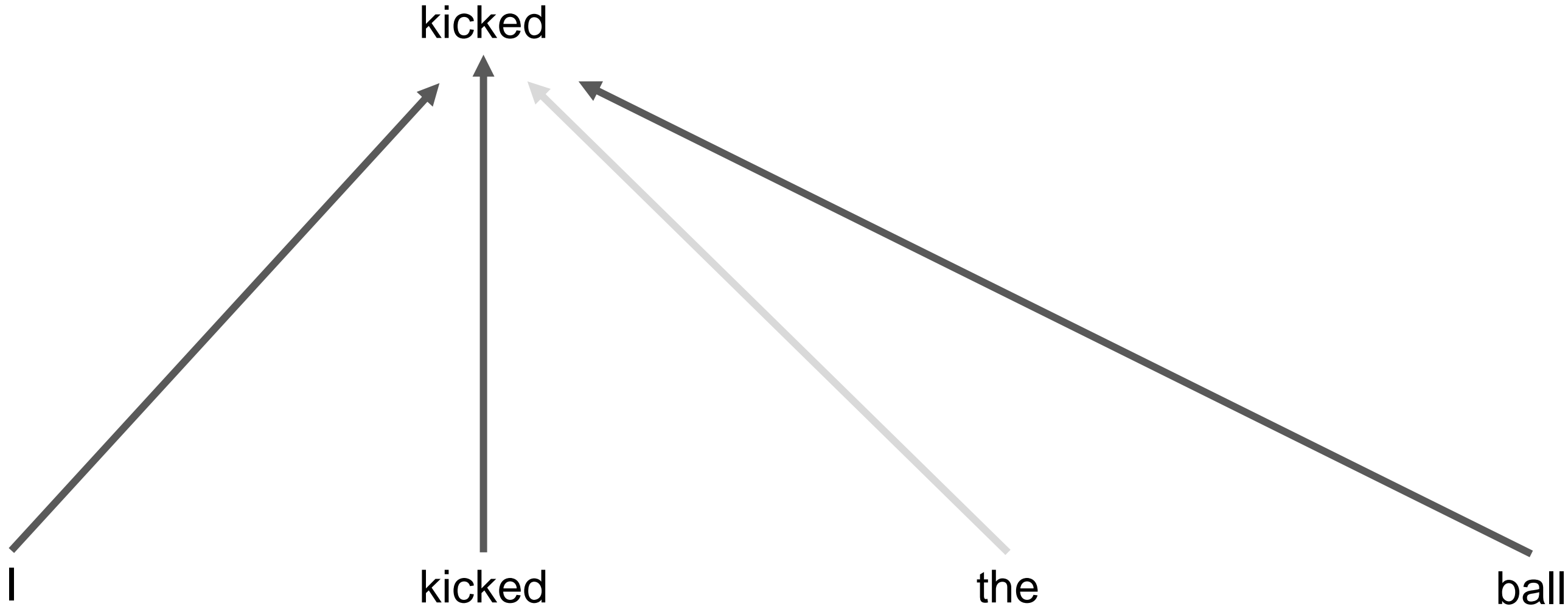
27

# Sequence Encoding Multi-Head Attention

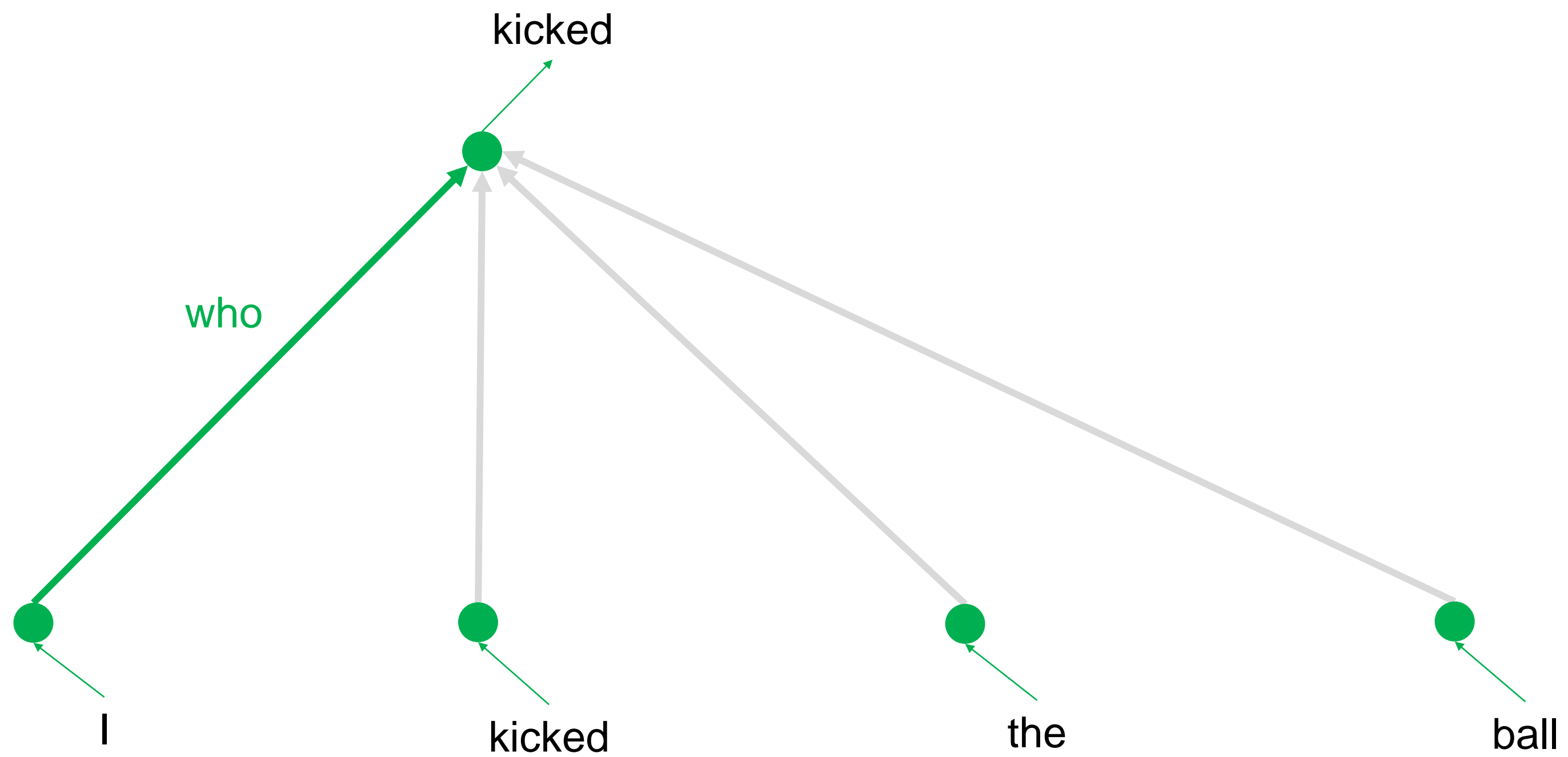
# Convolutions



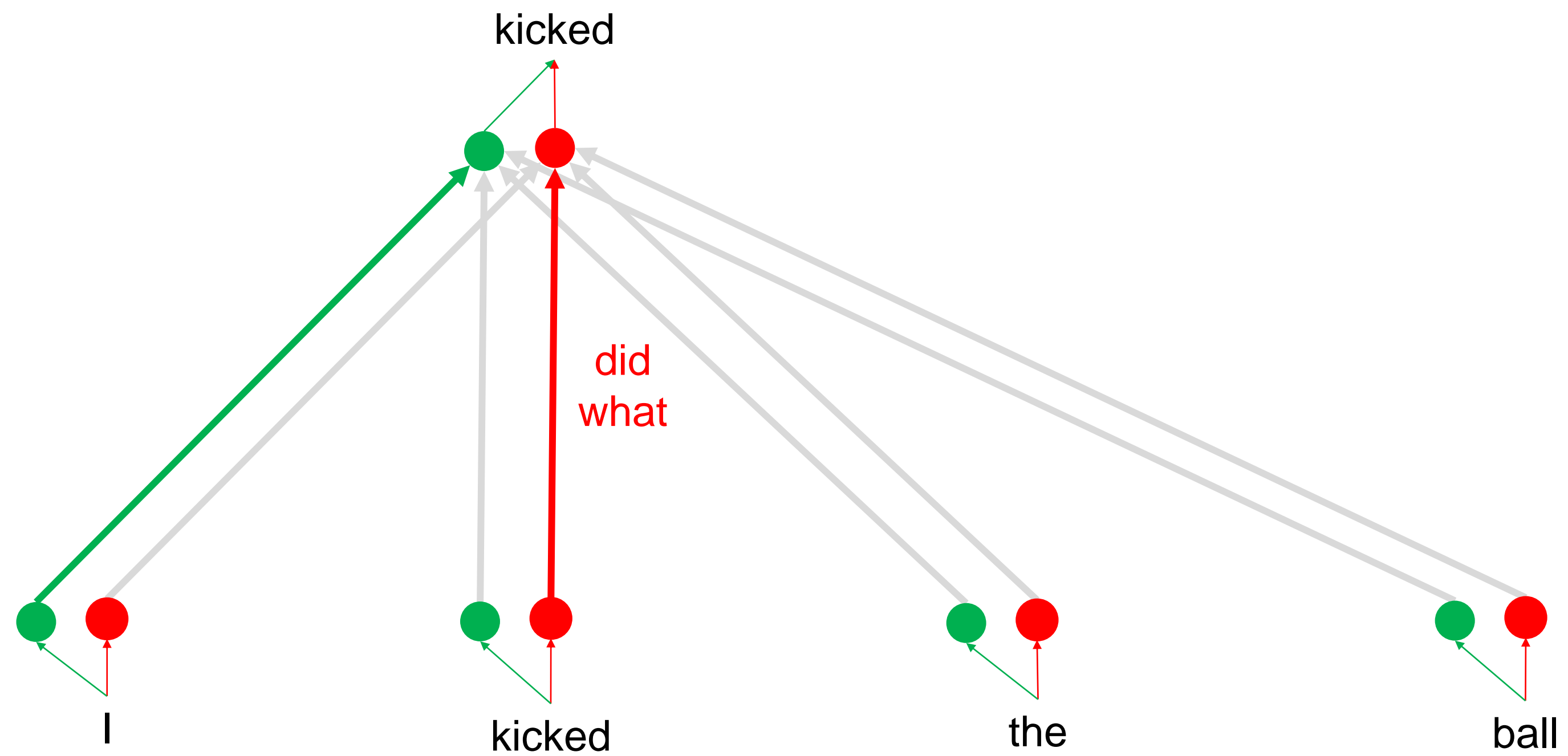
# Self-Attention



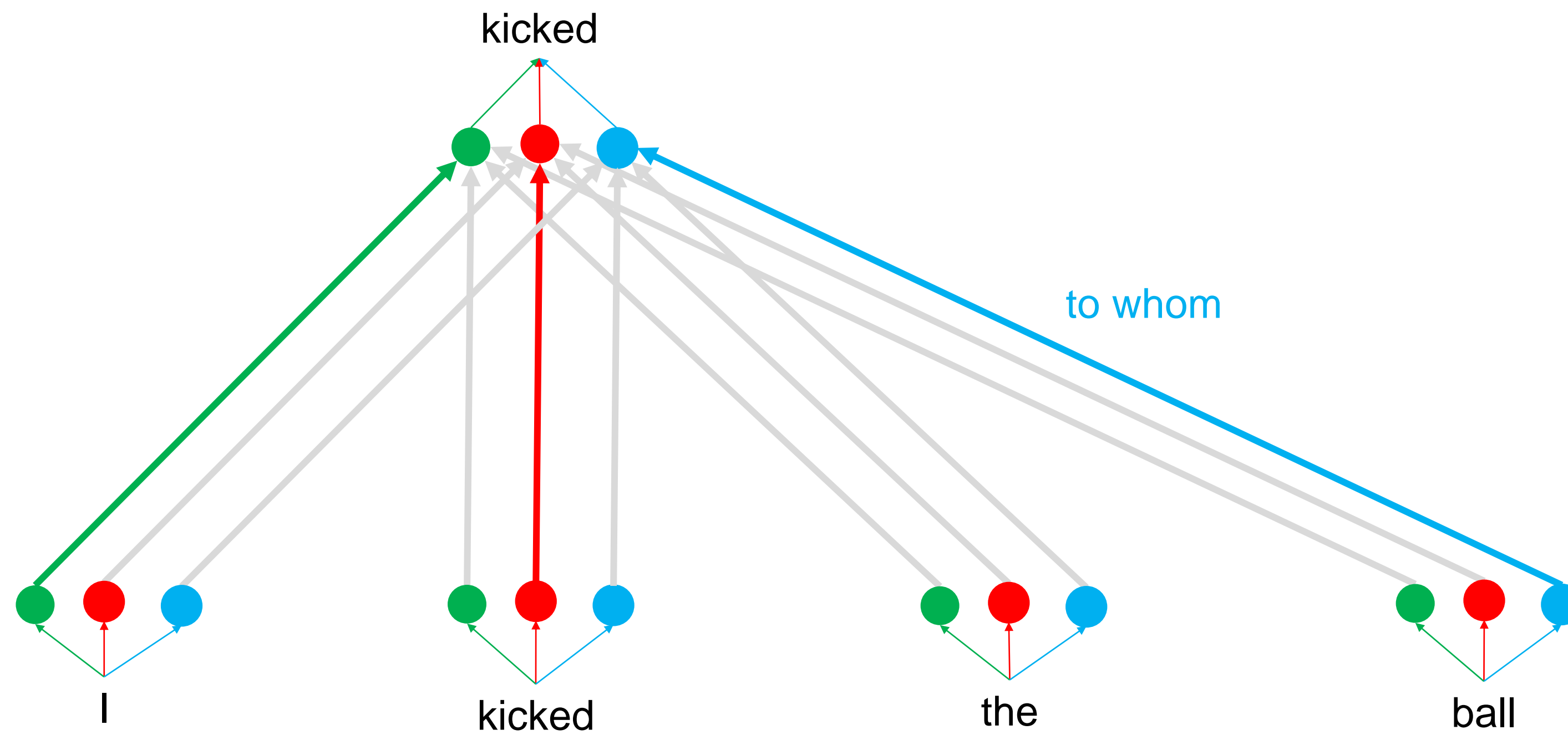
# Attention Head: who



# Attention Head: did what

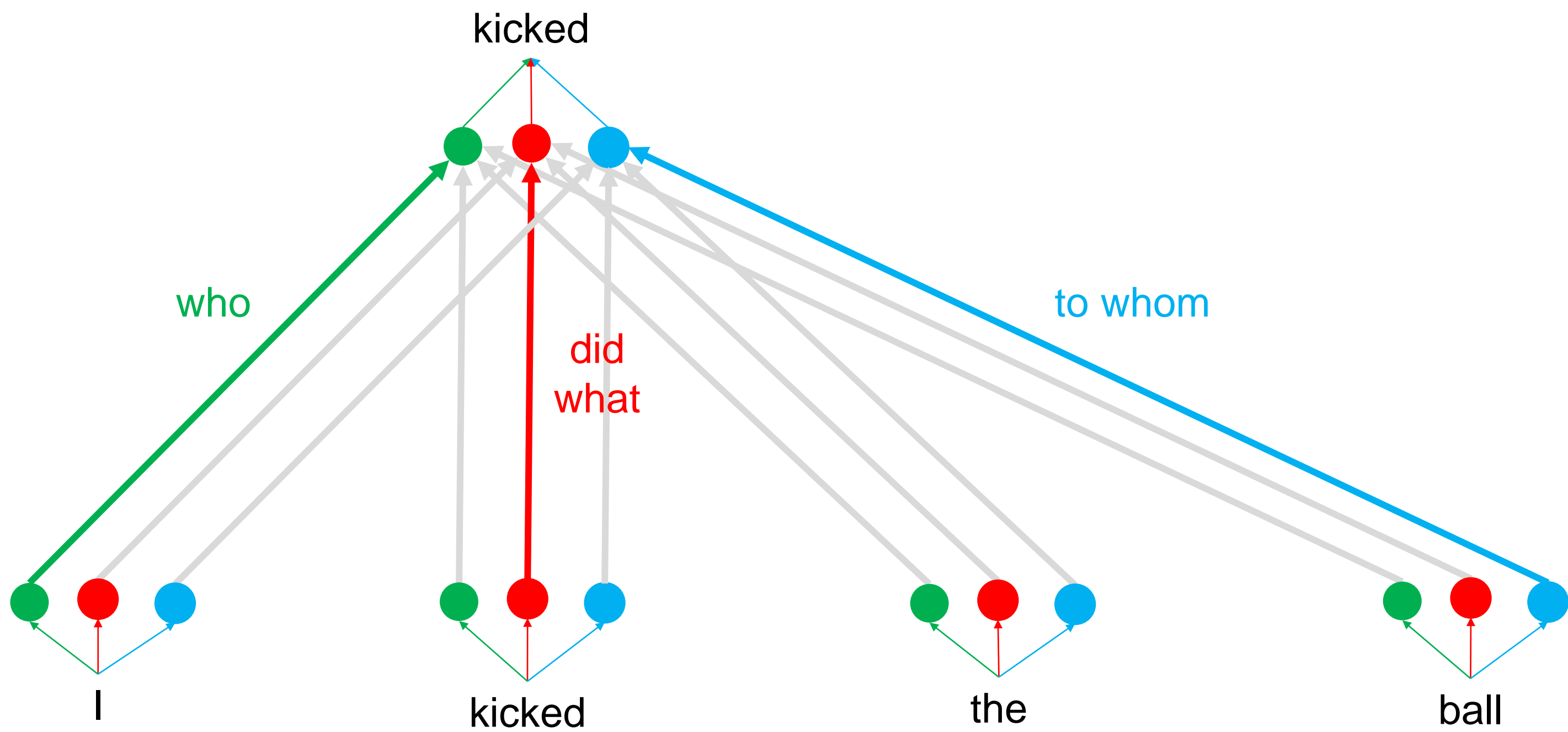


# Attention Head: to whom



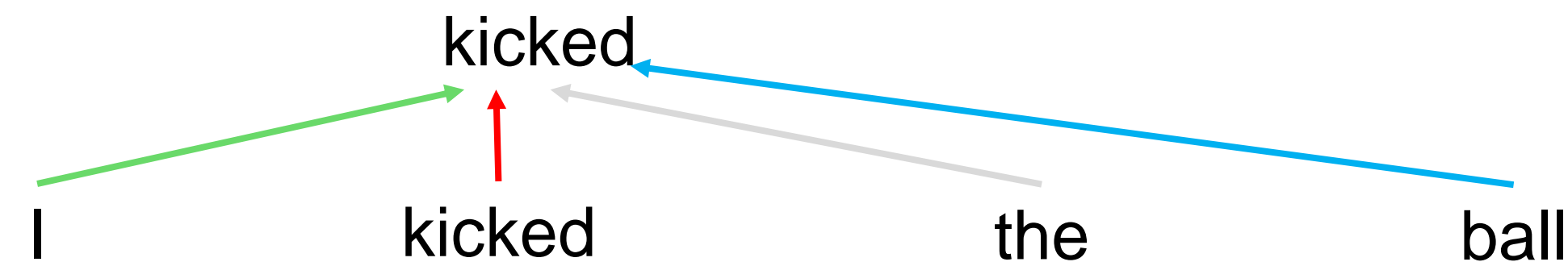


# Multi-Head Attention



# Comparison

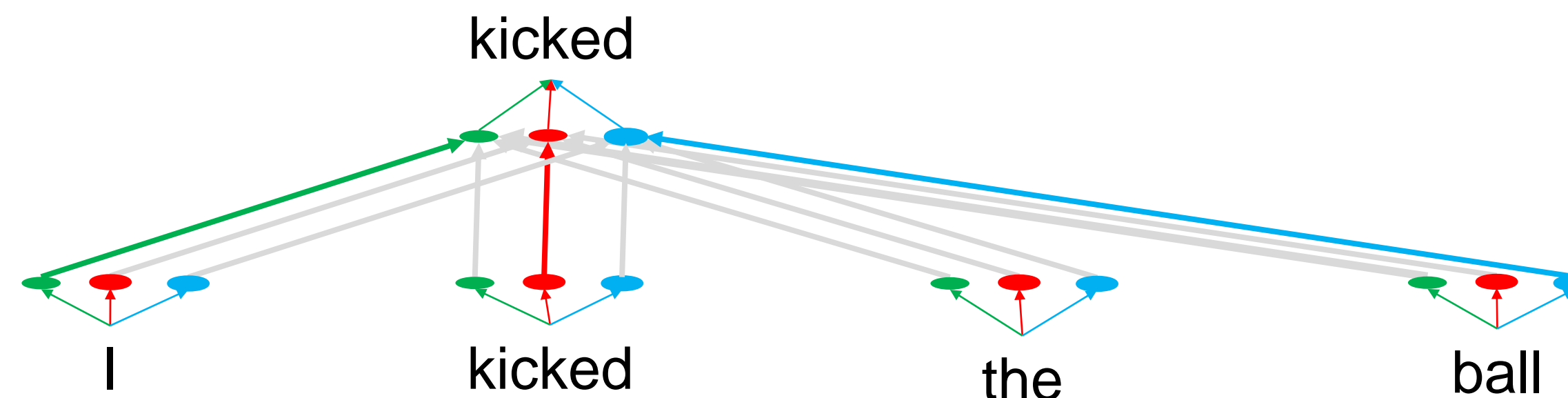
- Convolution: different linear transformations by relative positions



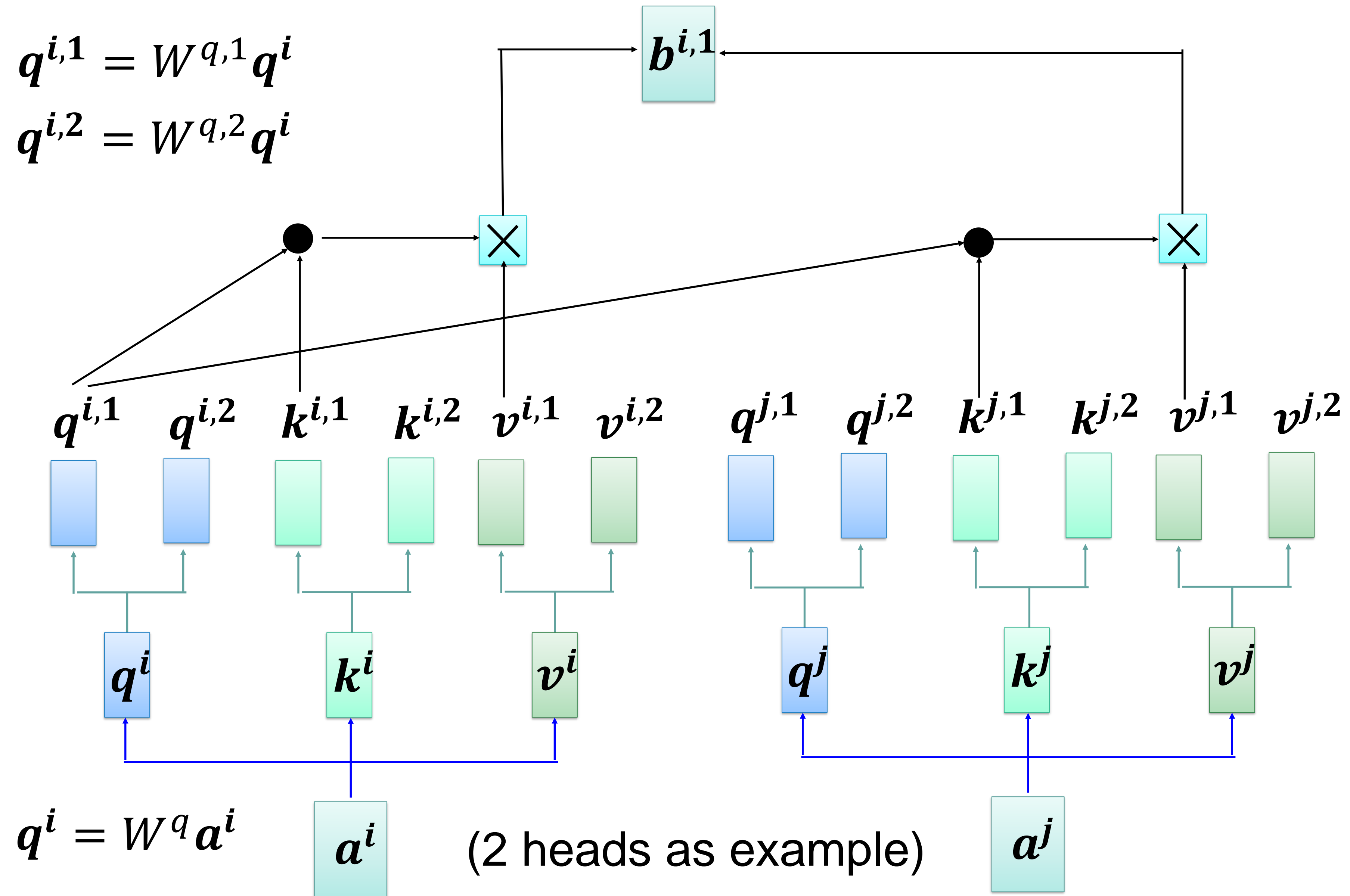
- Attention: a weighted average



- Multi-Head Attention: parallel attention layers with different linear transformations on input/output



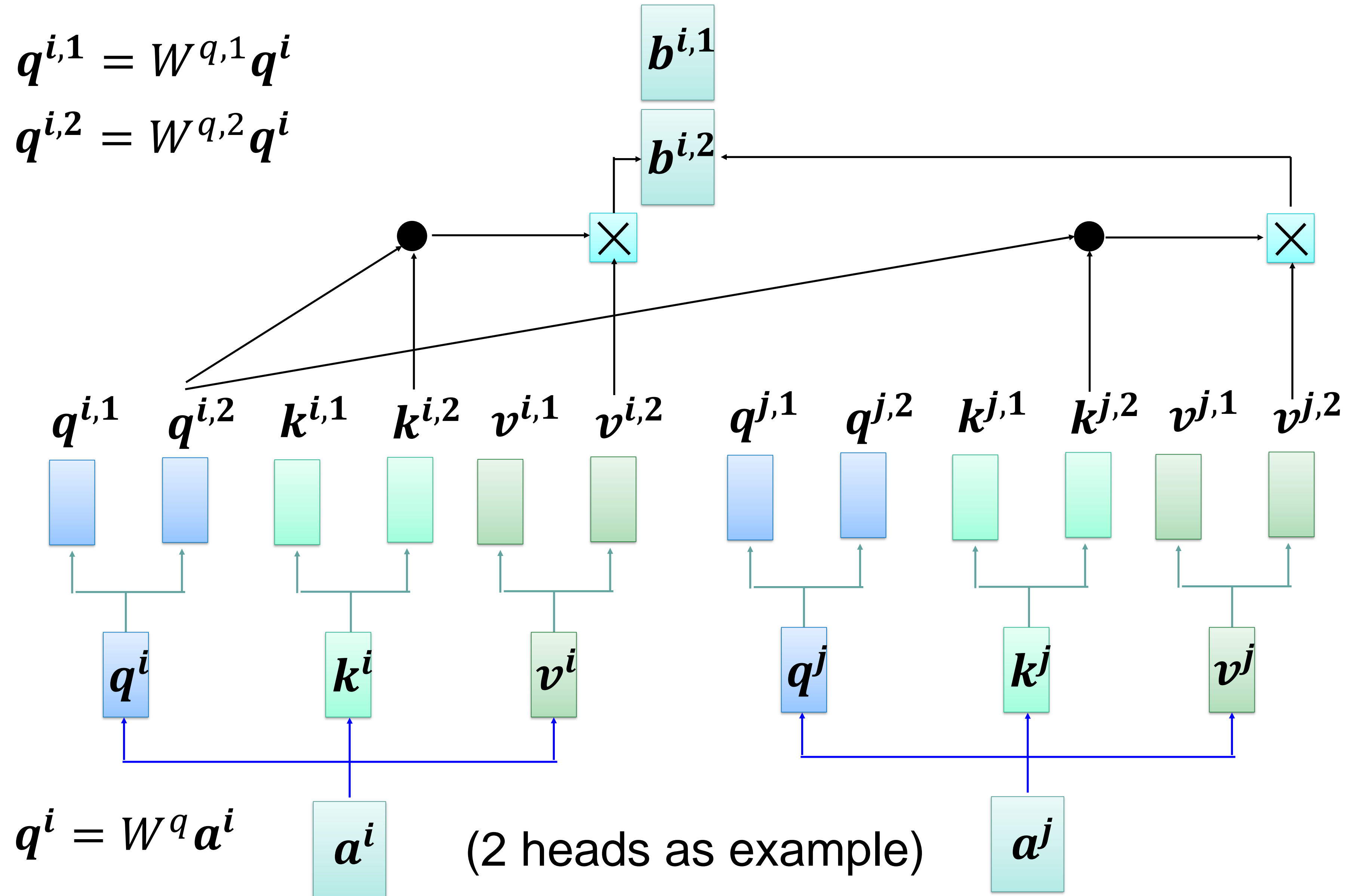
# Multi-Head Attention



# Multi-Head Attention

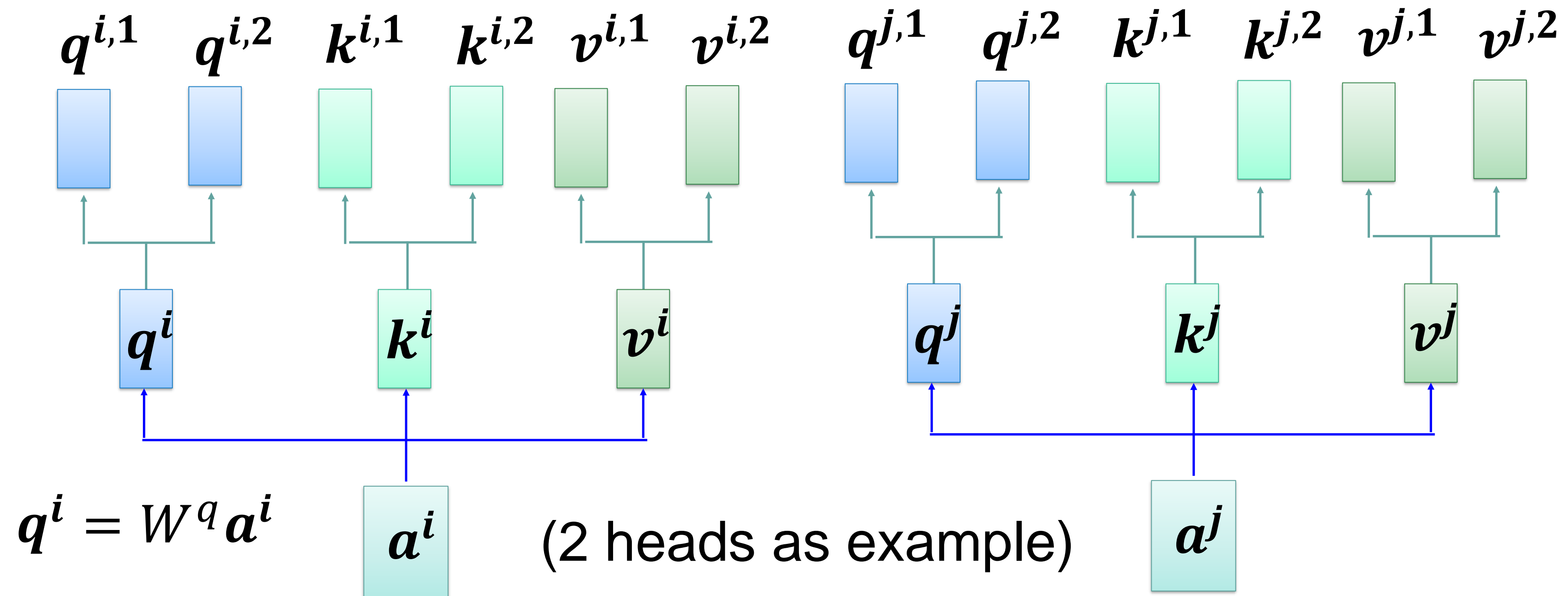
$$q^{i,1} = W^{q,1} q^i$$

$$q^{i,2} = W^{q,2} q^i$$



# Multi-Head Attention

$$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$$



38

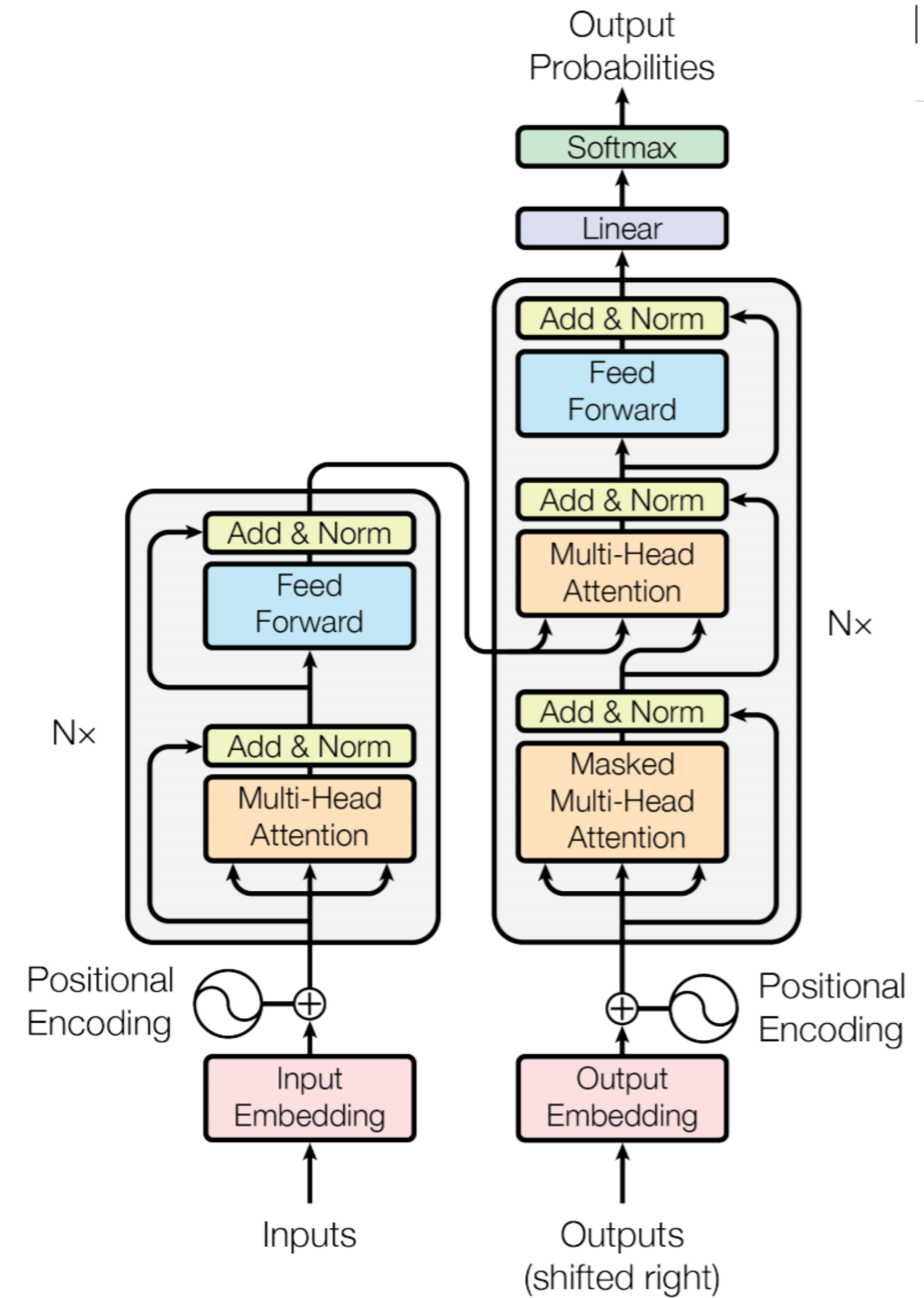
# Sequence Encoding Transformer



# Transformer Overview

- Non-recurrent encoder-decoder for MT
- PyTorch explanation by Sasha Rush

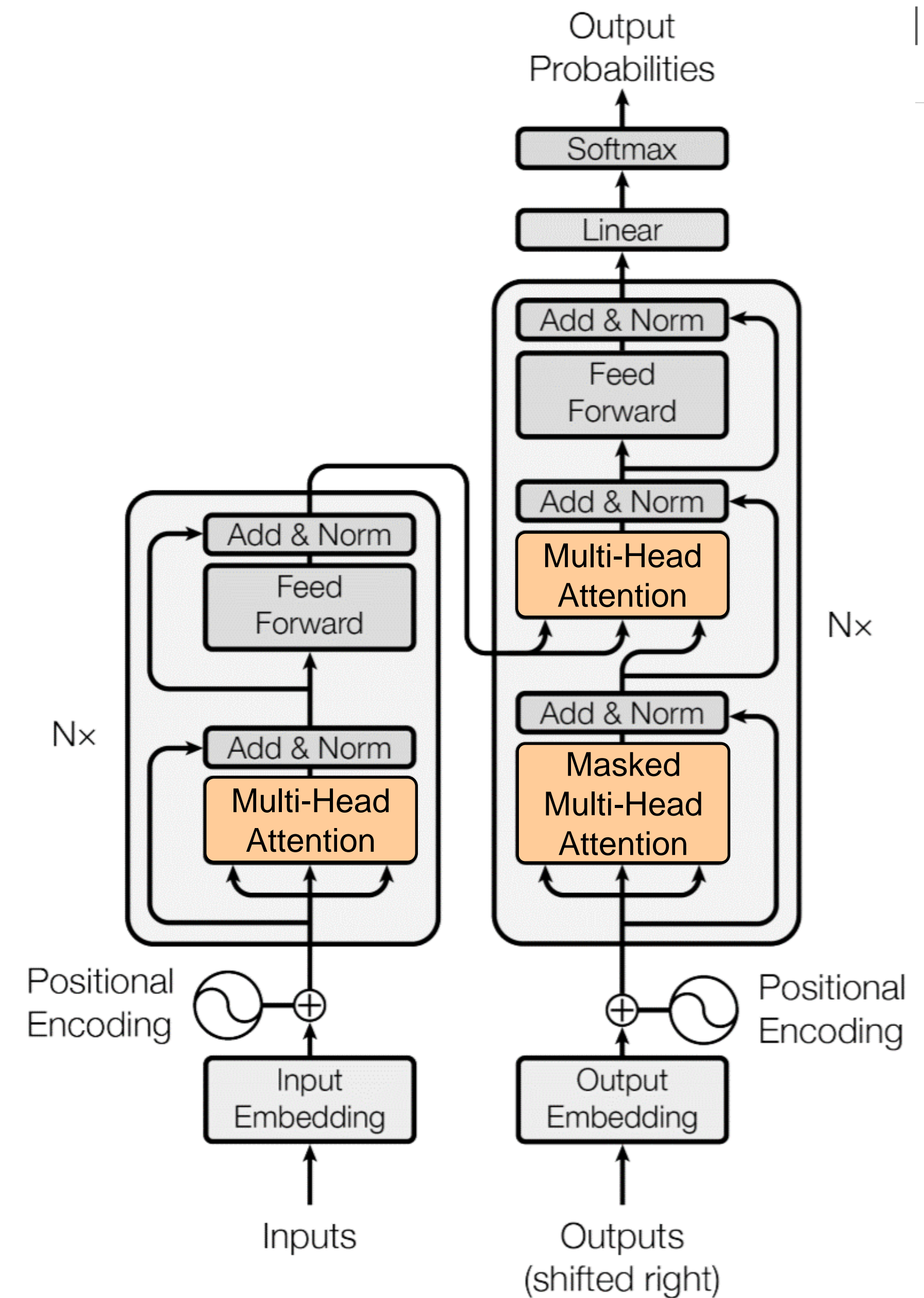
<http://nlp.seas.harvard.edu/2018/04/03/attention.html>



# Transformer Overview

- Non-recurrent encoder-decoder for MT
- PyTorch explanation by Sasha Rush

<http://nlp.seas.harvard.edu/2018/04/03/attention.html>





41

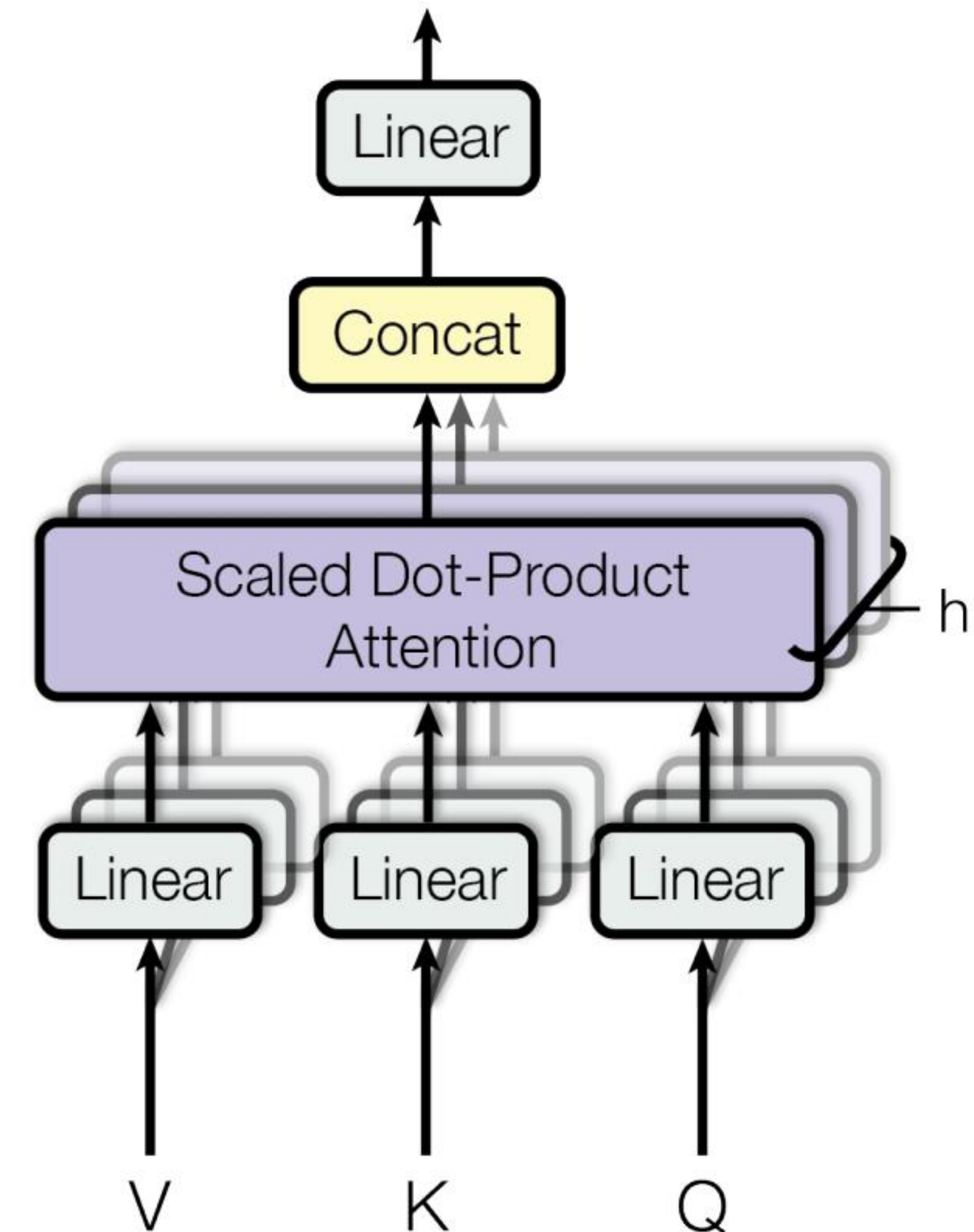
# Multi-Head Attention

- Idea: allow words to interact with one another
- Model
  - Map  $V$ ,  $K$ ,  $Q$  to lower dimensional spaces
  - Apply attention, concatenate outputs
  - Linear transformation

$$\text{MultiHead}(Q, K, V)$$

$$= \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

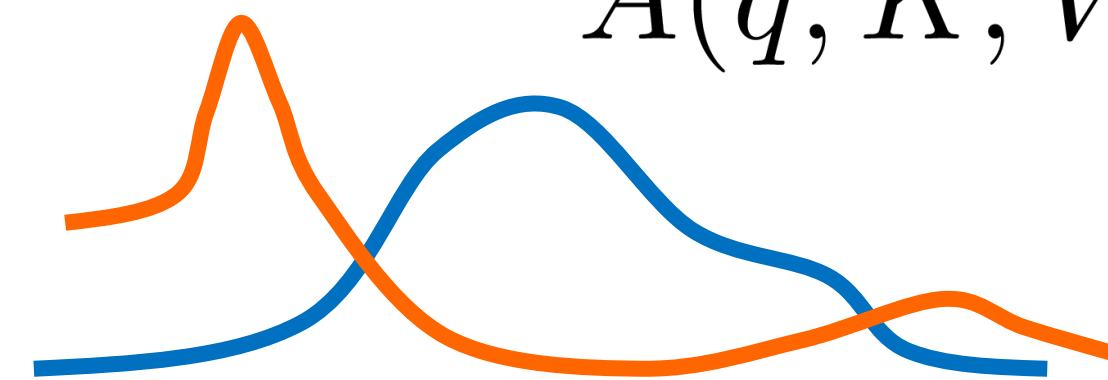
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



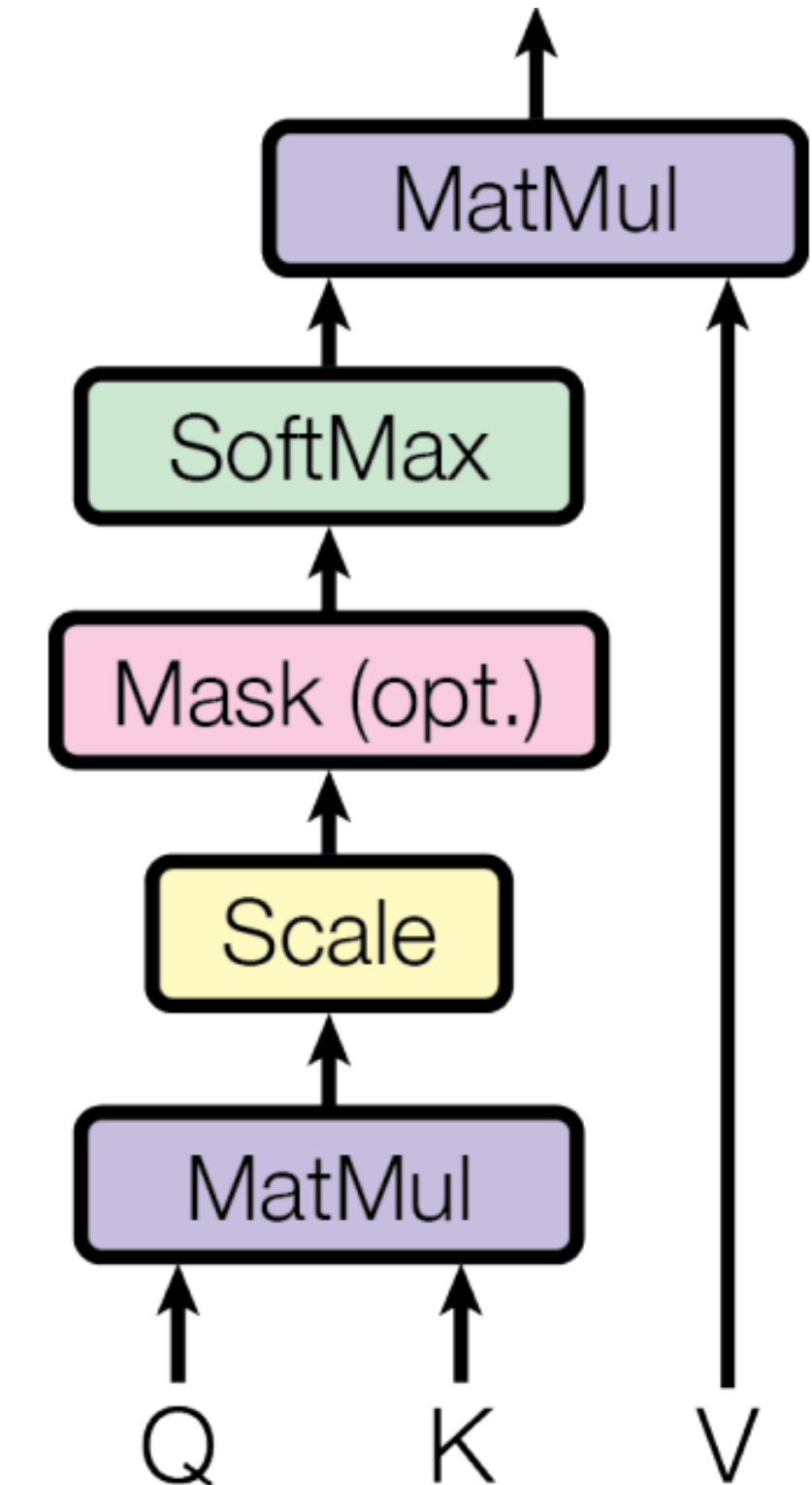
42

# Scaled Dot-Product Attention

- Problem: when  $d_k$  gets large, the variance of  $q^T k$  increases
  - some values inside softmax get large
  - the softmax gets very peaked
  - hence its gradient gets smaller
- Solution: scale by length of query/key vectors

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} v_i$$


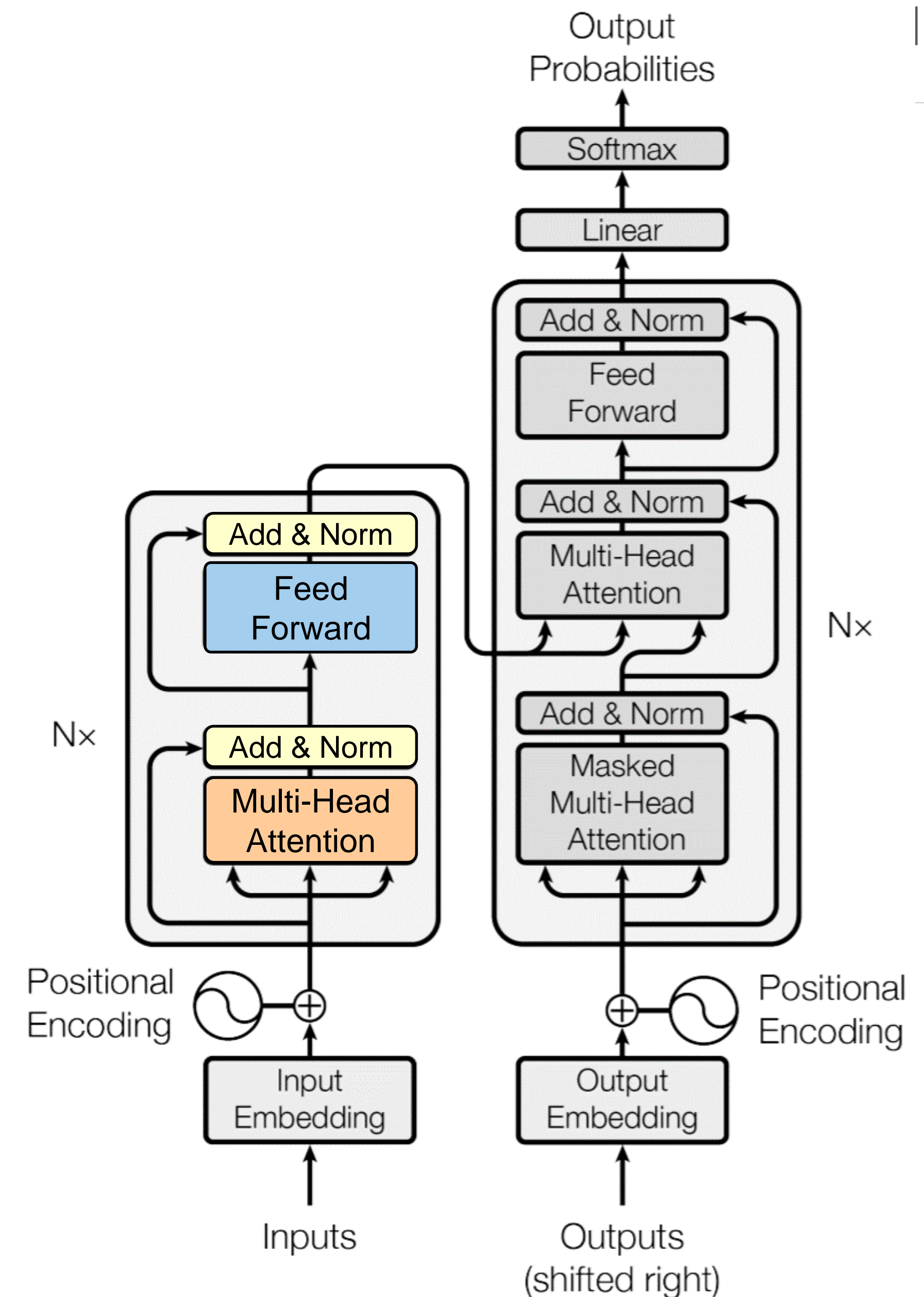
$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



# Transformer Overview

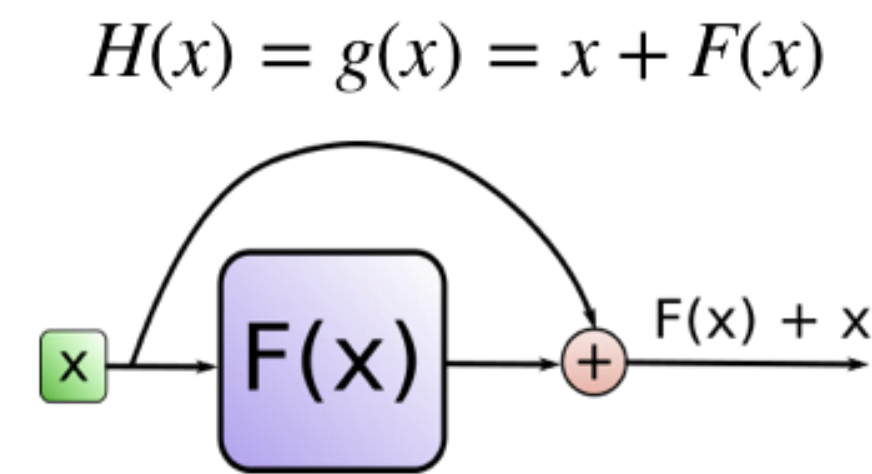
- Non-recurrent encoder-decoder for MT
- PyTorch explanation by Sasha Rush

<http://nlp.seas.harvard.edu/2018/04/03/attention.html>

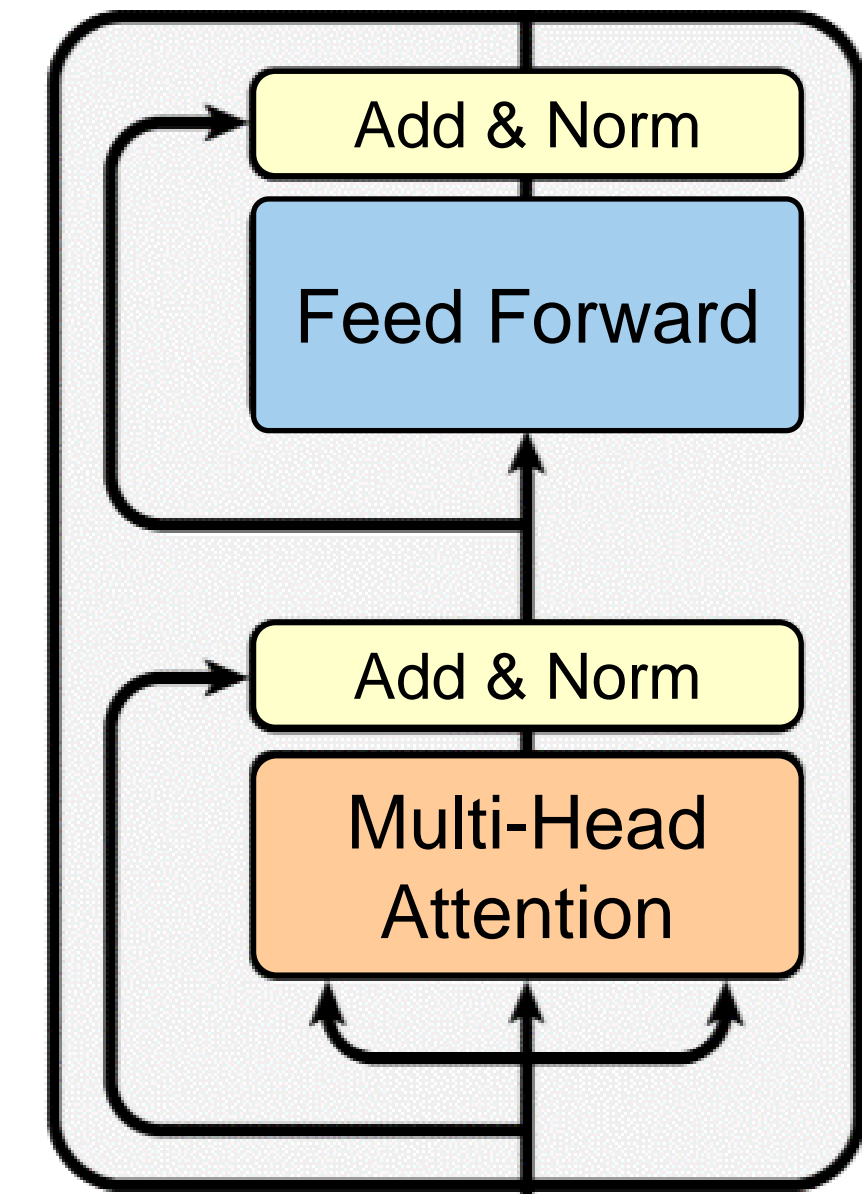


# Transformer Encoder Block

- Each block has
  - multi-head attention
  - 2-layer feed-forward NN (w/ ReLU)
- Both parts contain
  - Residual connection
  - Layer normalization (LayerNorm)



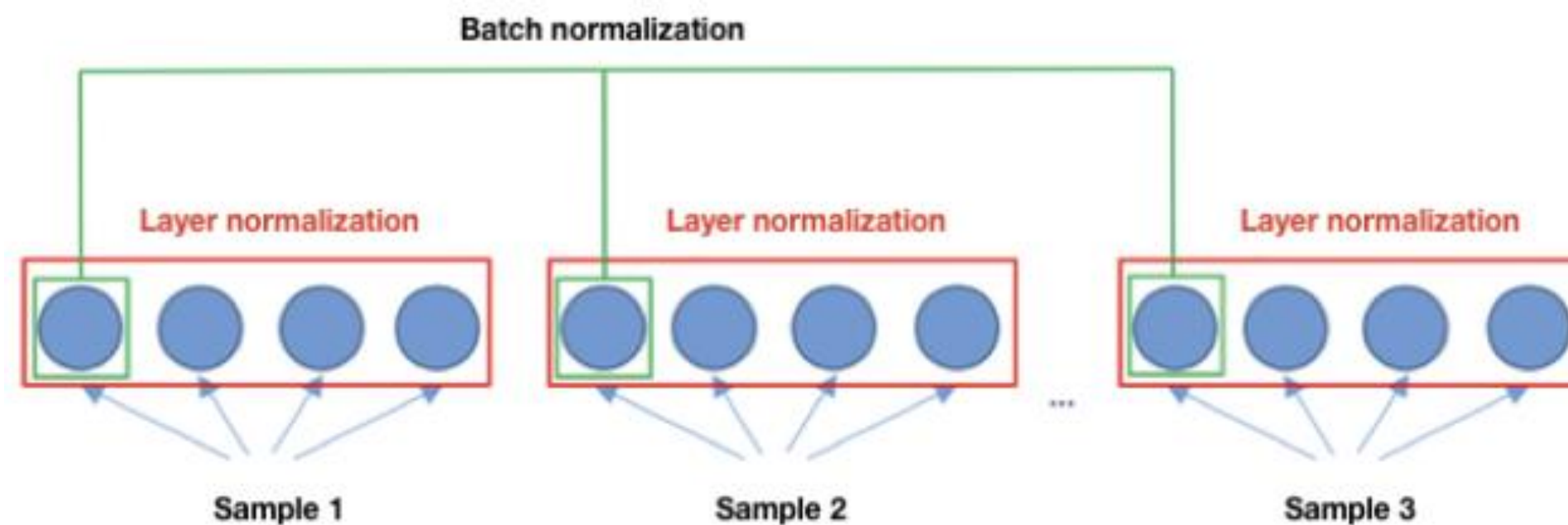
$$H(x) = g(x) = x + F(x)$$



Change input to have 0 mean and 1 variance per layer & per training point

→ LayerNorm(x + sublayer(x))

$$\mu^l = \frac{1}{H} \sum_{i=1}^H a_i^l \quad \sigma^l = \sqrt{\frac{1}{H} \sum_{i=1}^H (a_i^l - \mu^l)^2} \quad h_i = f\left(\frac{g_i}{\sigma_i} (a_i - \mu_i) + b_i\right)$$

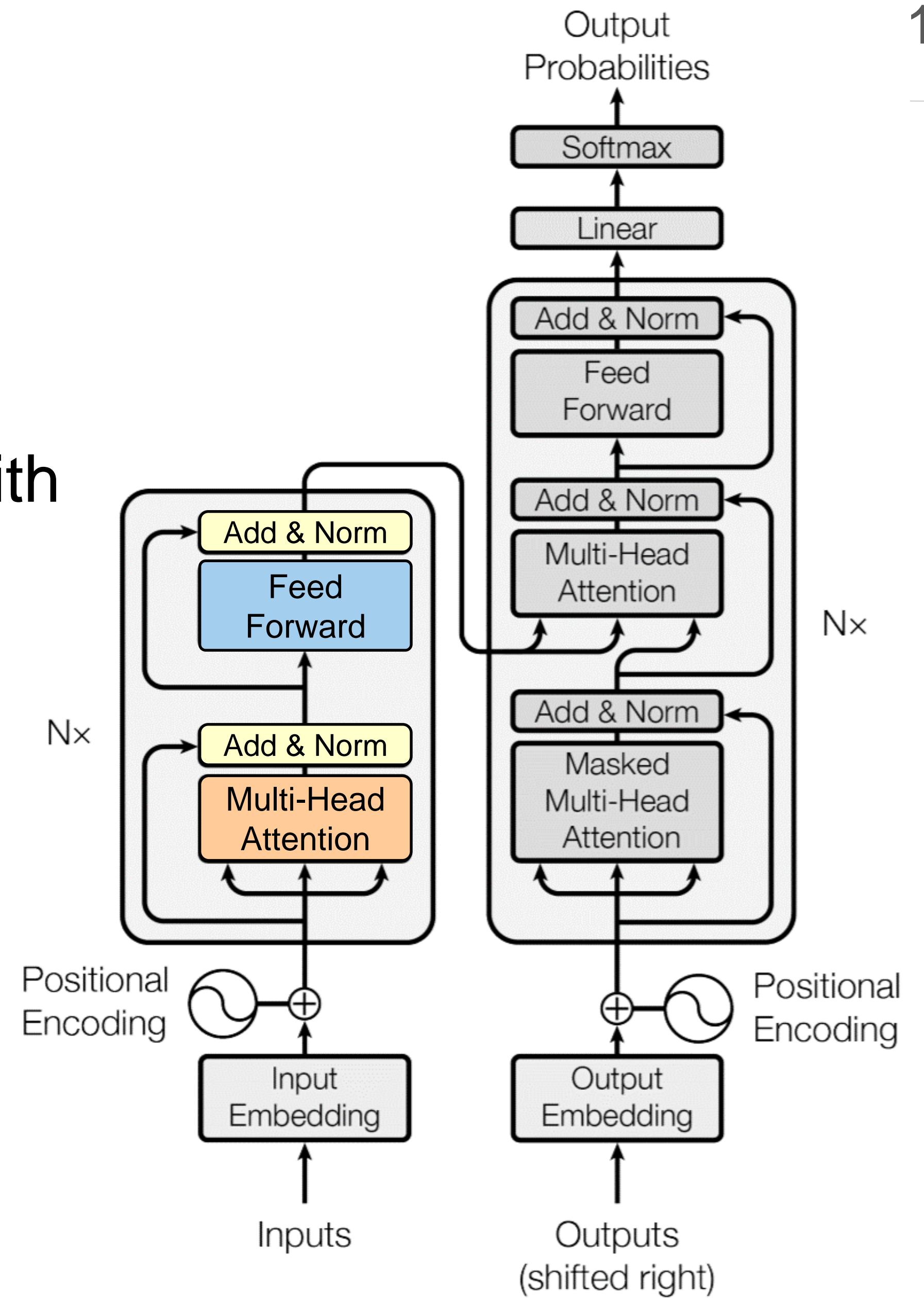
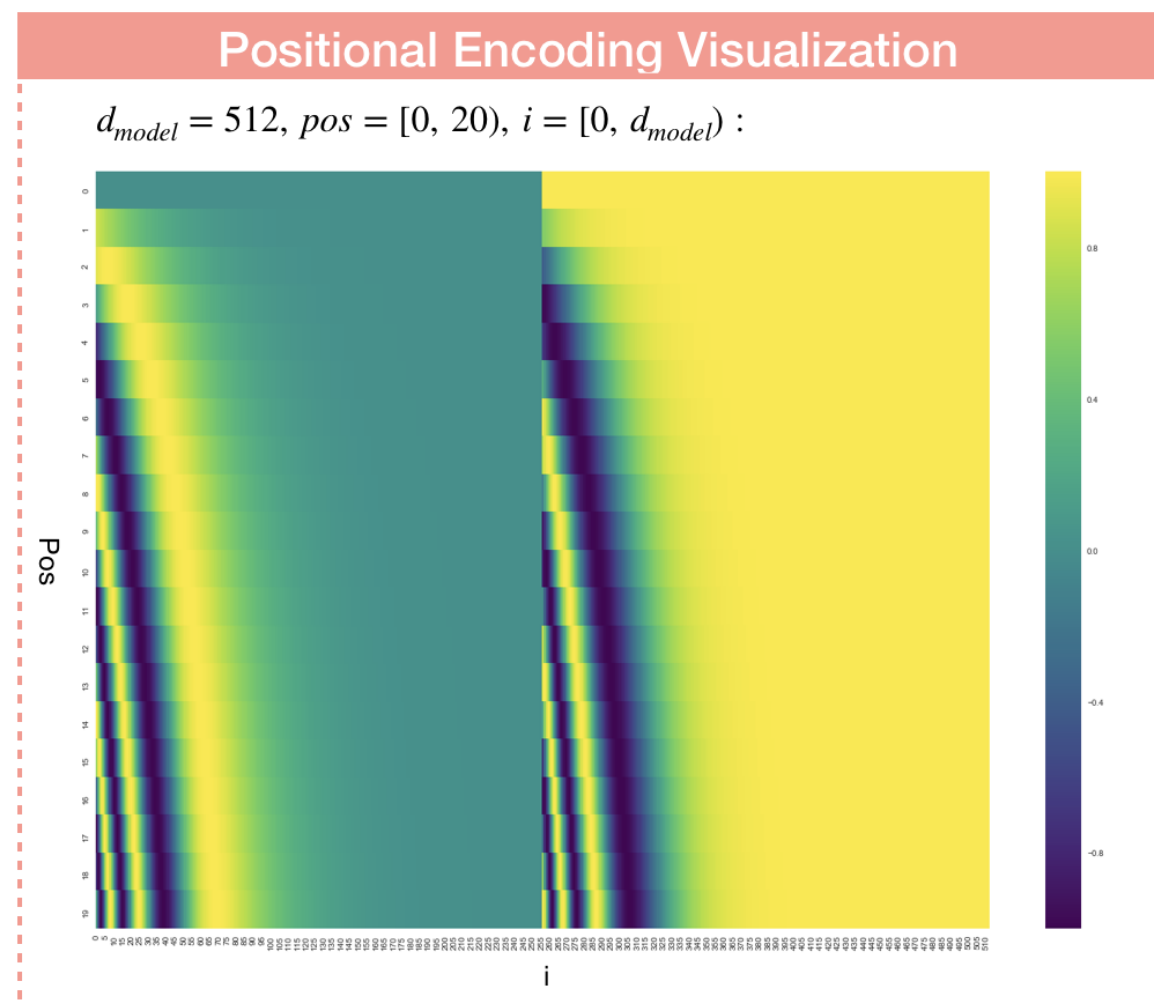


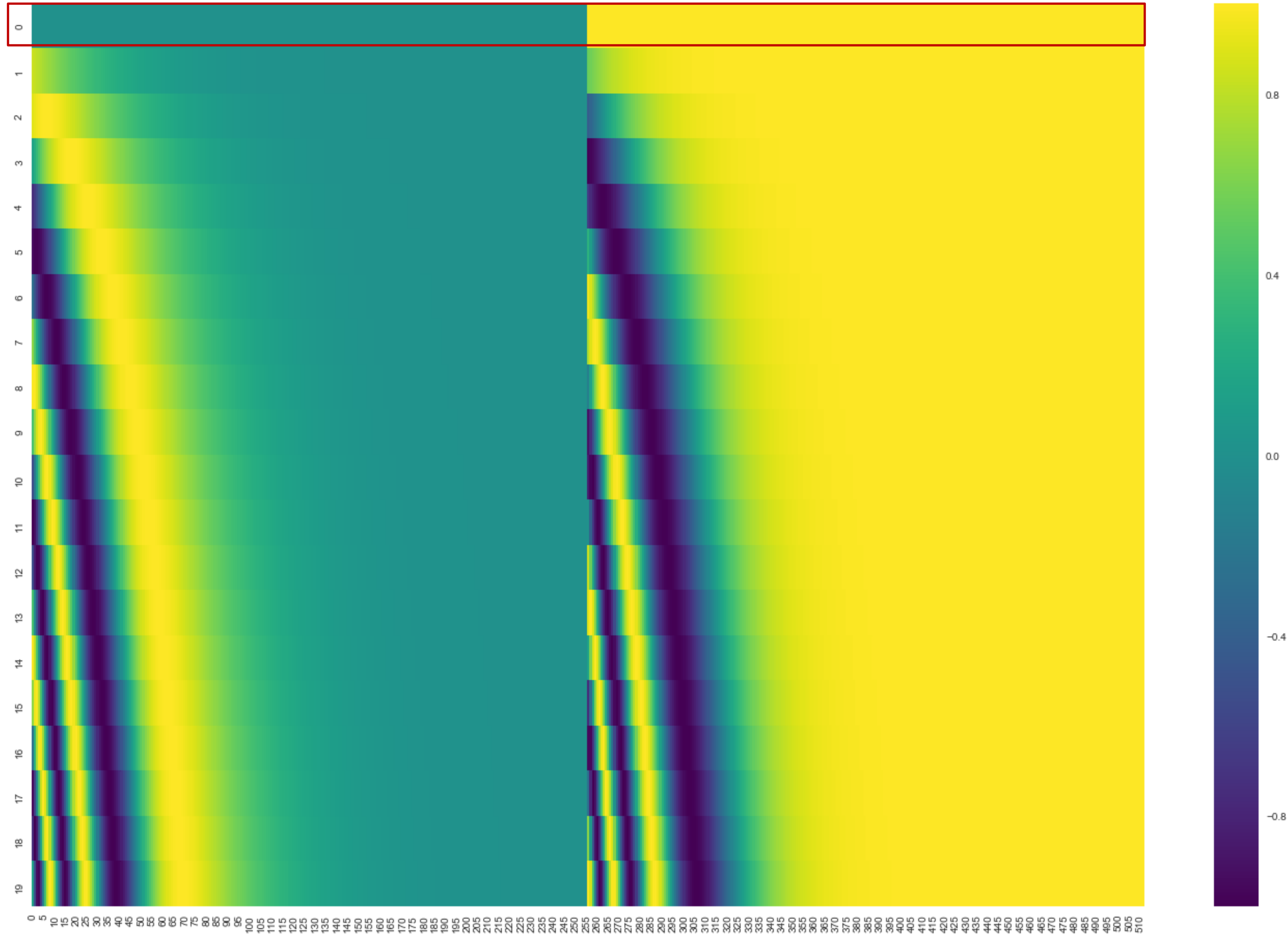
# Encoder Input

- Problem: temporal information is missing
- Solution: **positional encoding** allows words at different locations to have different embeddings with fixed dimensions

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$





# Multi-Head Attention Details

## encoder self attention

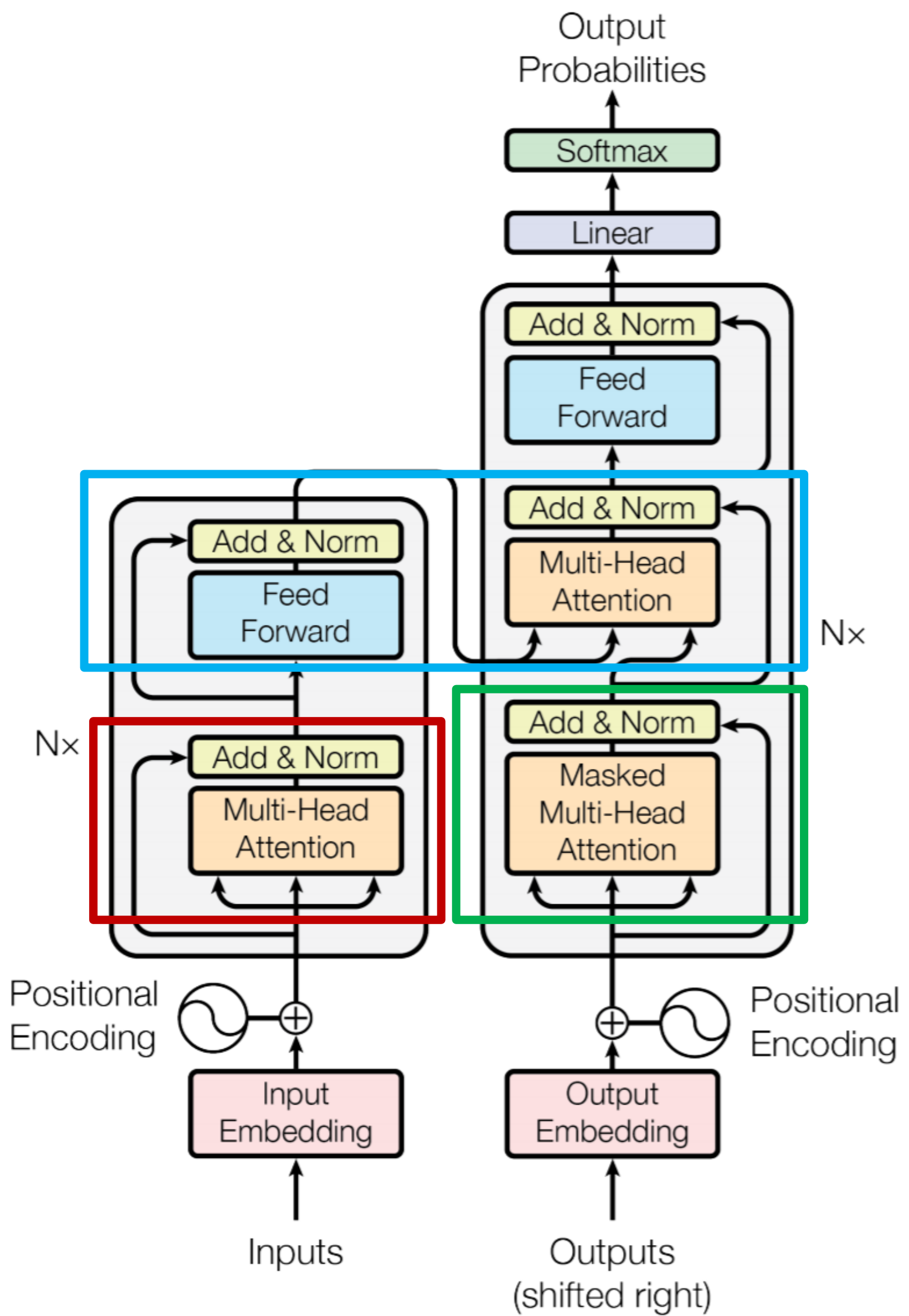
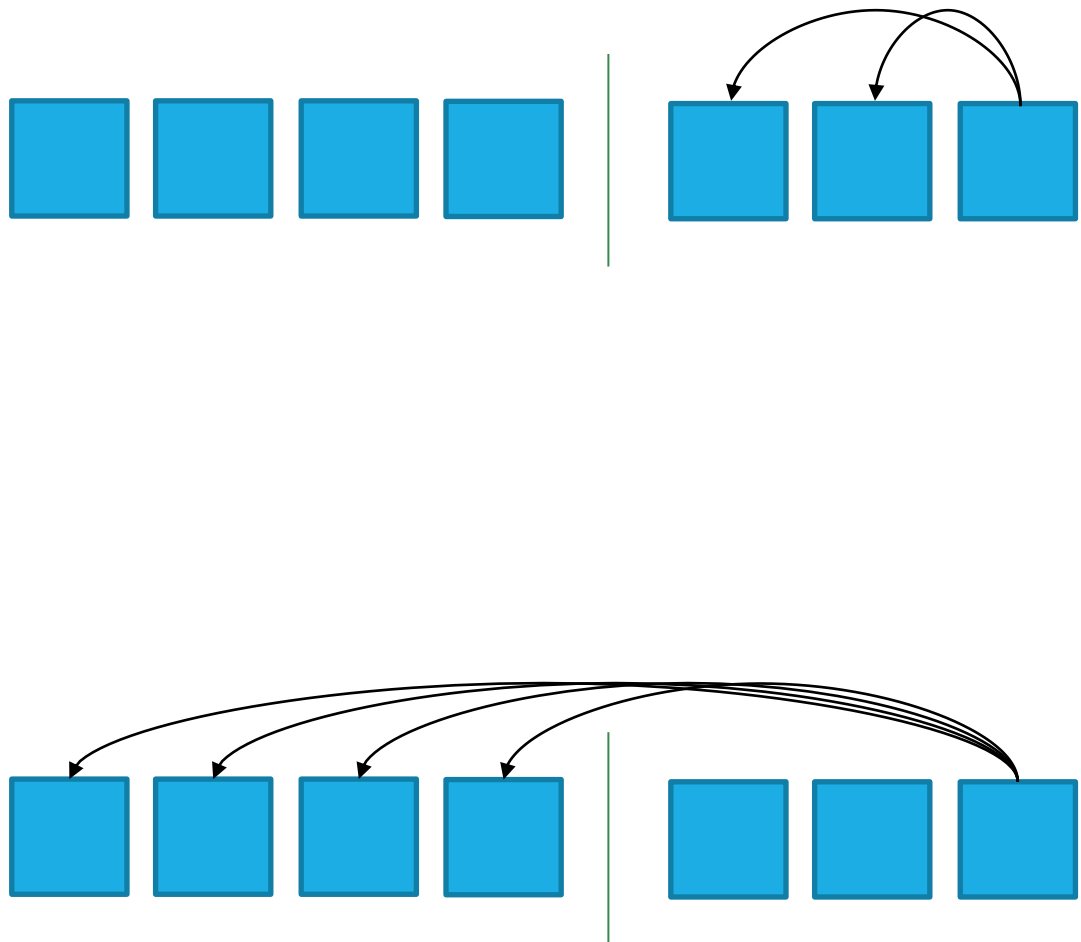
- 1. Multi-head Attention
- 2. **Q**uery=**K**ey=**V**alue

## decoder self attention

- 1. **M**asked Multi-head Attention
- 2. **Q**uery=**K**ey=**V**alue

## encoder-decoder attention

- 1. Multi-head Attention
- 2. Encoder Self attention=**K**ey=**V**alue
- 3. Decoder Self attention=**Q**uery



# Training Tips

- Byte-pair encodings (BPE)
- Checkpoint averaging
- ADAM optimizer with learning rate changes
- Dropout during training at every layer just before adding residual
- Label smoothing
- Auto-regressive decoding with beam search and length penalties



# MT Experiments

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.8</b>	$2.3 \cdot 10^{19}$	

# Parsing Experiments

Parser	Training	WSJ 23 F1
Vinyals & Kaiser et al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser et al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

51

# Concluding Remarks

- **Non-recurrence** model is easy to parallelize
- **Multi-head attention** captures different aspects by interacting between words
- **Positional encoding** captures location information
- Each transformer block can be applied to diverse tasks

