

(Recap) Transformer

Wei Xu

(many slides from Greg Durrett)

Math Review

Dot Product

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

Bilinear Product

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

$$\begin{aligned} \mathbf{a}^\top W \mathbf{b} &= a_1(w_{11}b_1 + w_{12}b_2 + w_{13}b_3) \\ &\quad + a_2(w_{21}b_1 + w_{22}b_2 + w_{23}b_3) \\ &\quad + a_3(w_{31}b_1 + w_{32}b_2 + w_{33}b_3) \end{aligned}$$

Readings

- ▶ “The Annotated Transformer” by Sasha Rush
<https://nlp.seas.harvard.edu/2018/04/03/attention.html>
- ▶ “The Illustrated Transformer” by Jay Lamar
<http://jalammar.github.io/illustrated-transformer/>
- ▶ Jurafsky+Martin Chapter 7

Jurafsky+Martin NLP Textbook

Volume I: Large Language Models

Chapter

- 1: [Introduction](#)
- 2: [Words and Tokens](#)
- 3: [N-gram Language Models](#)
- 4: [Logistic Regression and Text Classification](#)
- 5: [Embeddings](#)
- 6: [Neural Networks](#)
- 7: [Transformers and Pretraining](#)
- 8: [Post-training](#)

Volume II: Advanced LLM Topics and Tools

Chapter

- 9: [Masked Language Models](#)
- 10: [Interpretability](#)
- 11: [Information Retrieval and RAG](#)
- 12: Agents [not written yet]
- 13: [Machine Translation](#)
- 14: [RNNs and LSTMs](#)
- 15: [Phonetics and Speech Feature Extraction](#)
- 16: [Automatic Speech Recognition](#)
- 17: [Text-to-Speech](#)

Volume III: Annotating Linguistic Structure

Chapter

- 18: [Sequence Labeling for Parts of Speech and Named Entities](#)
- 19: [Context-Free Grammars and Constituency Parsing](#)
- 20: [Dependency Parsing](#)
- 21: [Information Extraction: Relations, Events, and Time](#)
- 22: [Semantic Role Labeling and Argument Structure](#)
- 23: [Lexicons for Sentiment, Affect, and Connotation](#)
- 24: [Coreference Resolution and Entity Linking](#)
- 25: [Discourse Coherence](#)
- 26: [Conversation and its Structure](#)

Appendix (will be just on the web)

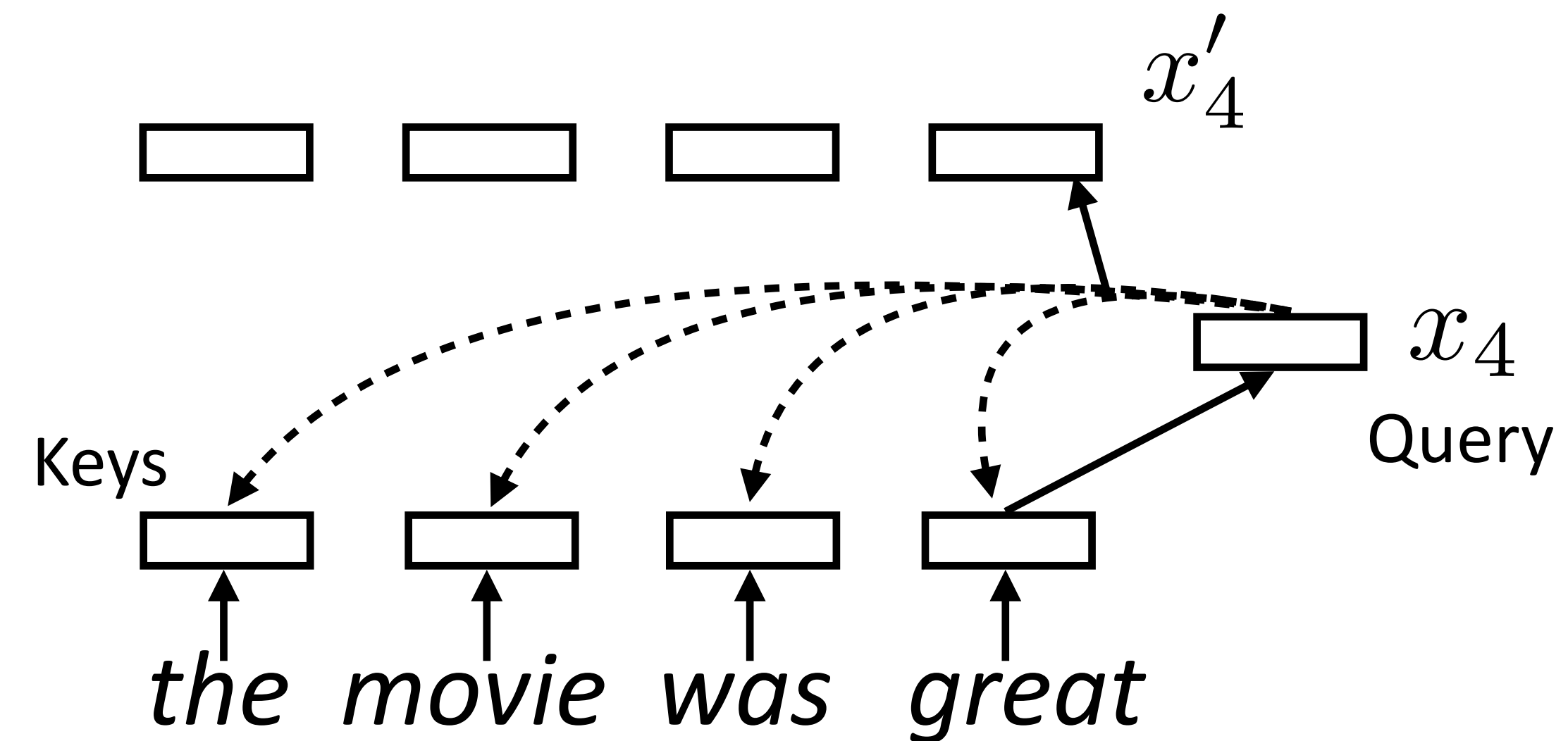
- A: [Hidden Markov Models](#)
- B: [Naive Bayes Classification](#)

Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$

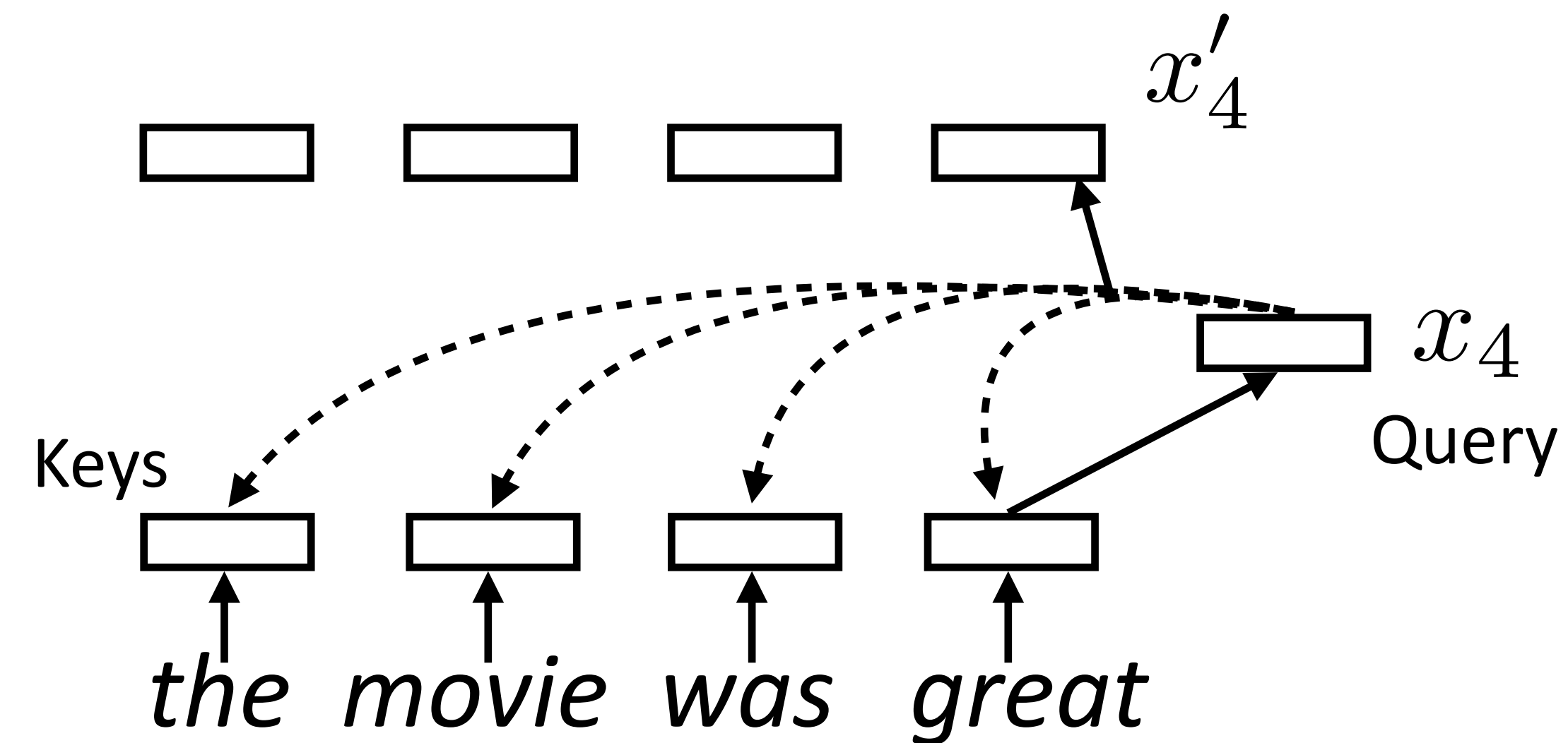


Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$



Queries
(tokens attending
to other tokens)

	Keys			
	The	movie	was	great
The	0.25	0.55	0.13	0.07
movie	0.10	0.30	0.12	0.48
was	0.08	0.42	0.35	0.15
great	0.05	0.60	0.10	0.25

row-wise softmax

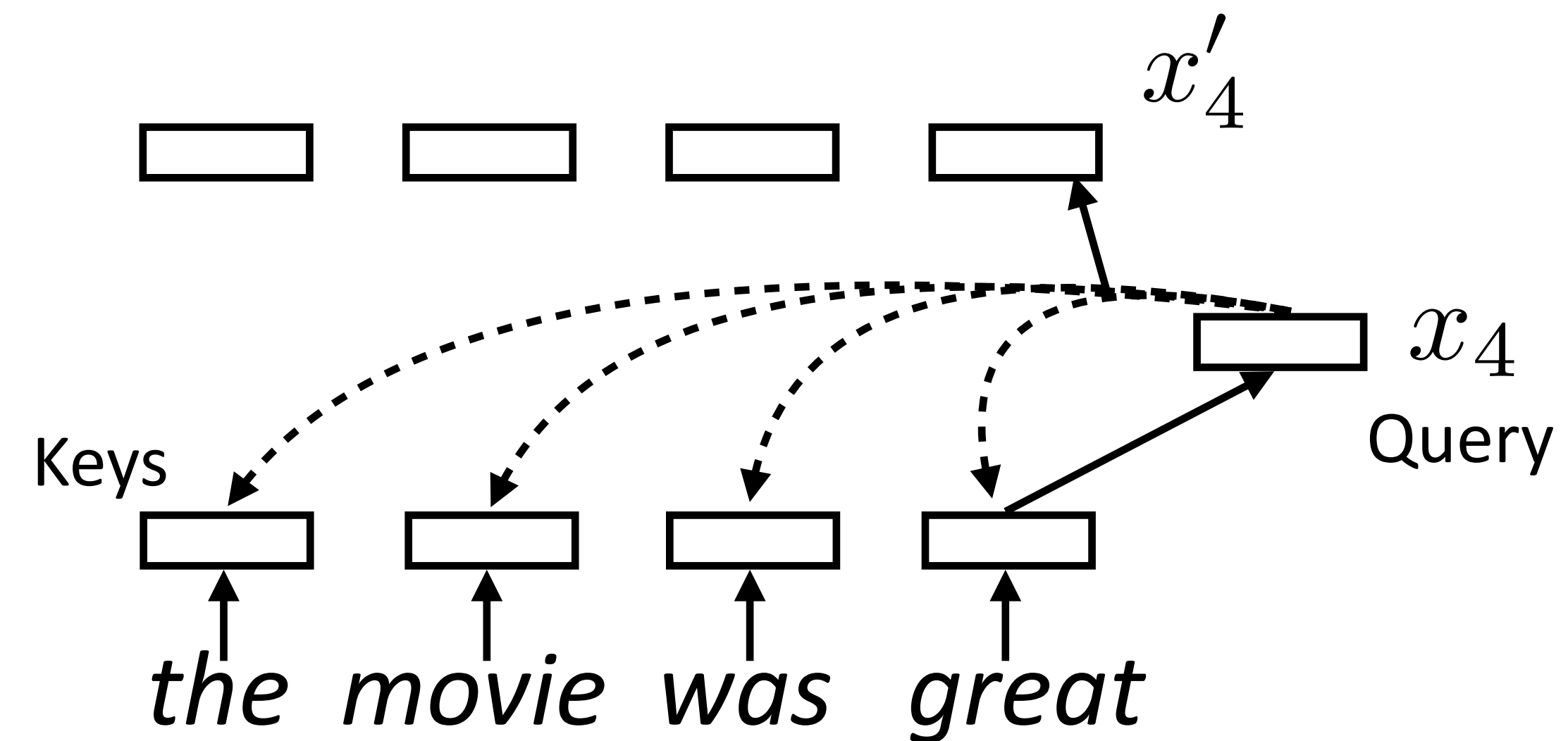
$$\begin{aligned}
 x'_{\text{great}} &= \alpha_{41}x_{\text{the}} + \alpha_{42}x_{\text{movie}} + \alpha_{43}x_{\text{was}} + \alpha_{44}x_{\text{great}} \\
 &= 0.05x_{\text{the}} + 0.60x_{\text{movie}} + 0.10x_{\text{was}} + 0.25x_{\text{great}}
 \end{aligned}$$

Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

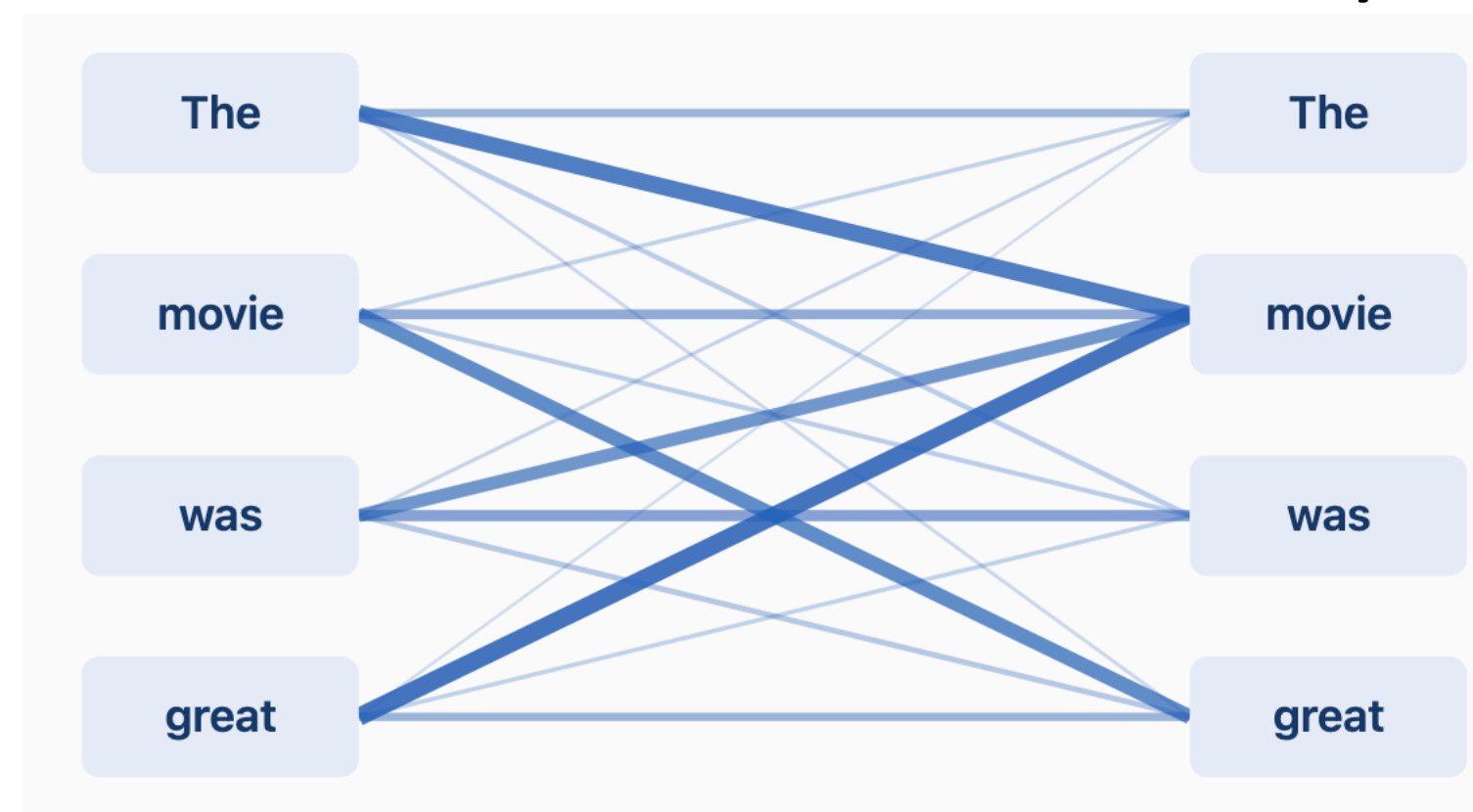
$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$



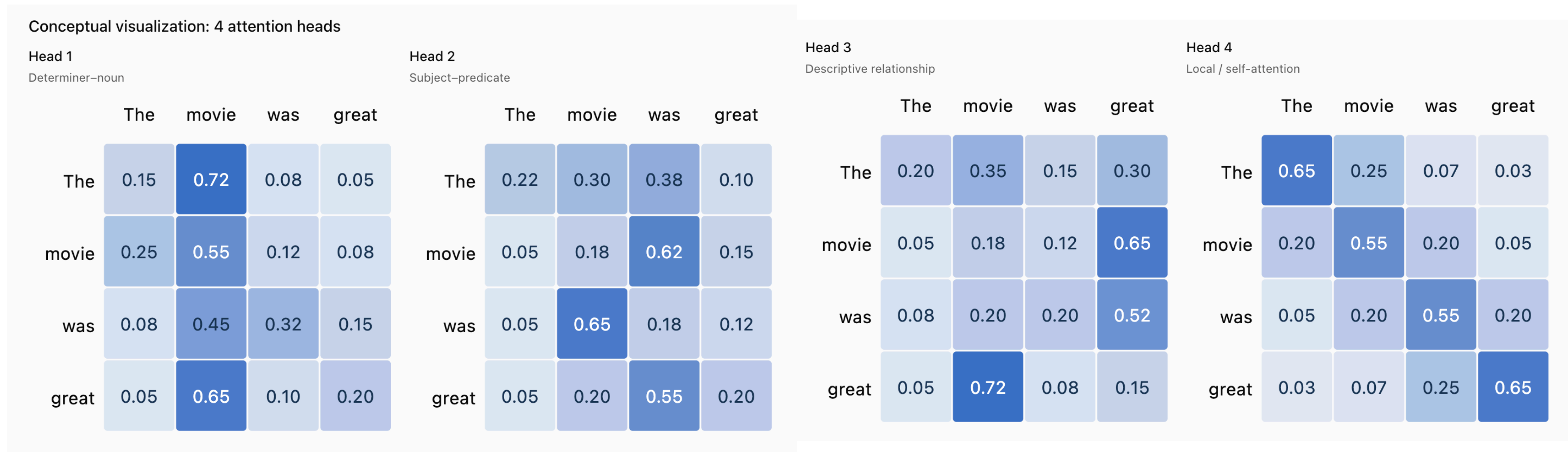
Queries
(tokens attending
to other tokens)

	Keys			
	The	movie	was	great
The	0.25	0.55	0.13	0.07
movie	0.10	0.30	0.12	0.48
was	0.08	0.42	0.35	0.15
great	0.05	0.60	0.10	0.25

Queries



Multi-head Self-Attention



- Multiple “heads” use different sets of parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{h,i,j} = \text{softmax} (x_i^\top W_h x_j)$$

↖
index of attention heads

$$x'_{h,i} = \sum_{j=1}^n \alpha_{h,i,j} V_h x_j$$

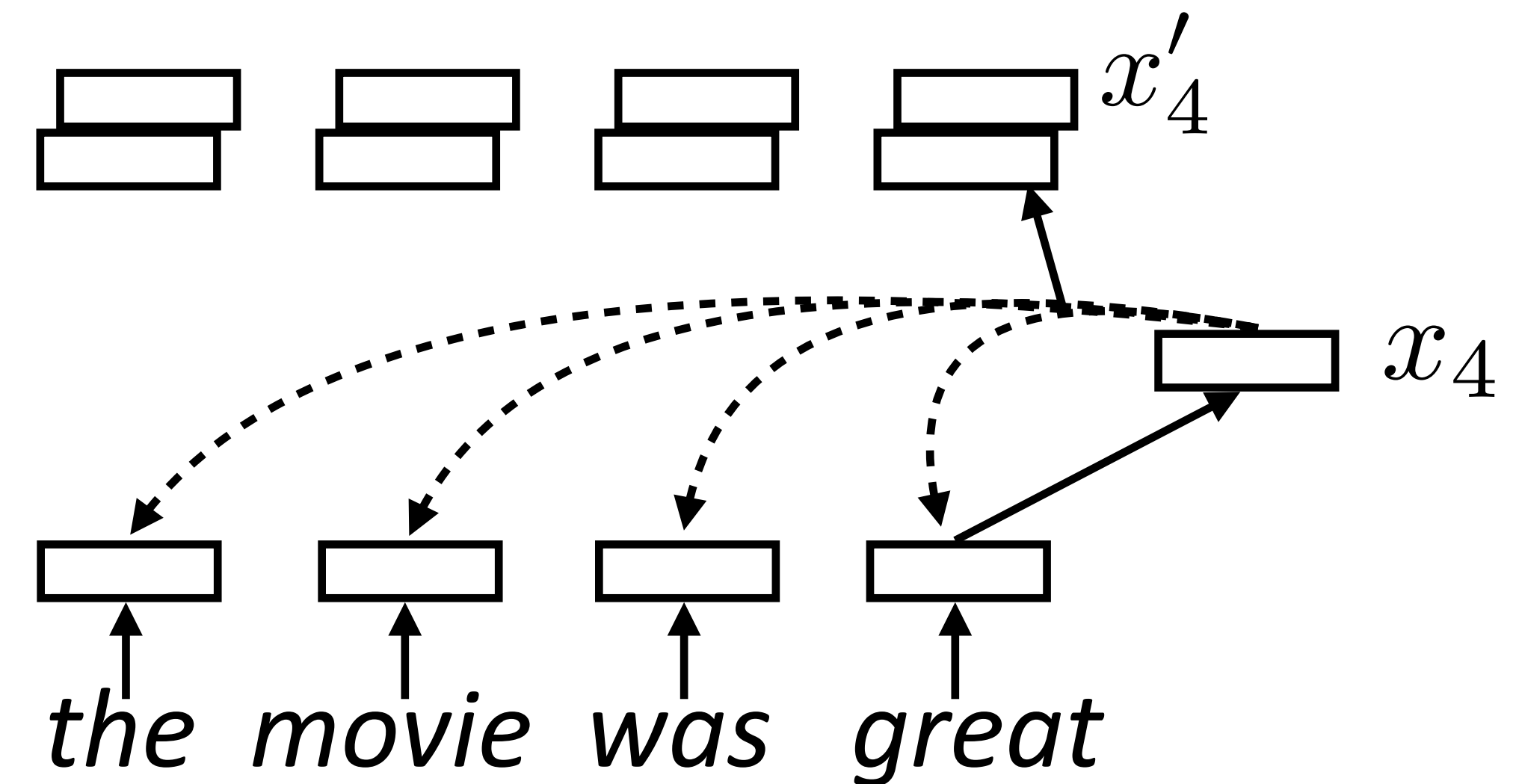
Vaswani et al. (2017)

Multi-head Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$



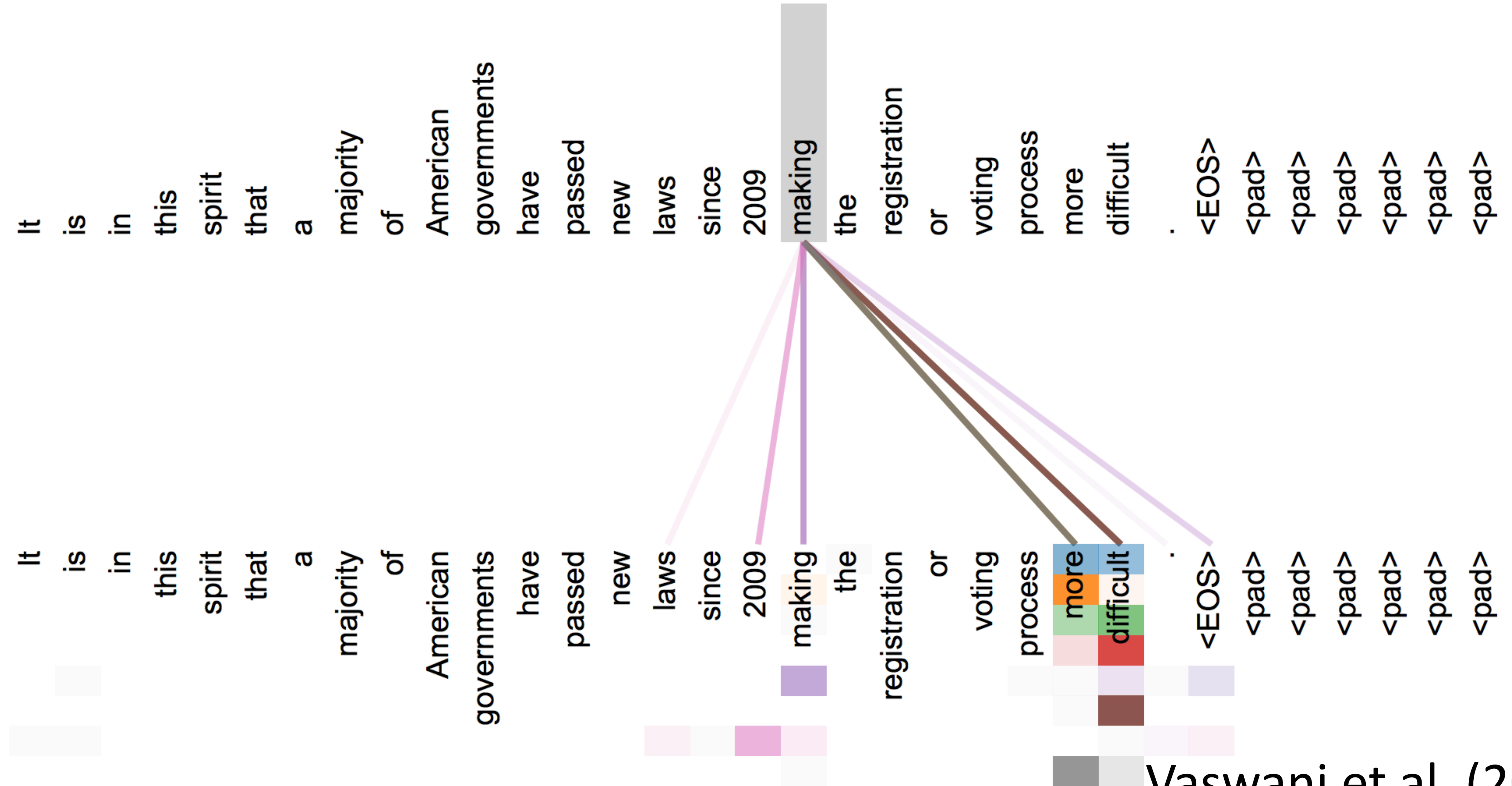
- Multiple “heads” use different sets of parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{h,i,j} = \text{softmax}(x_i^\top W_h x_j)$$

↖
index of attention heads

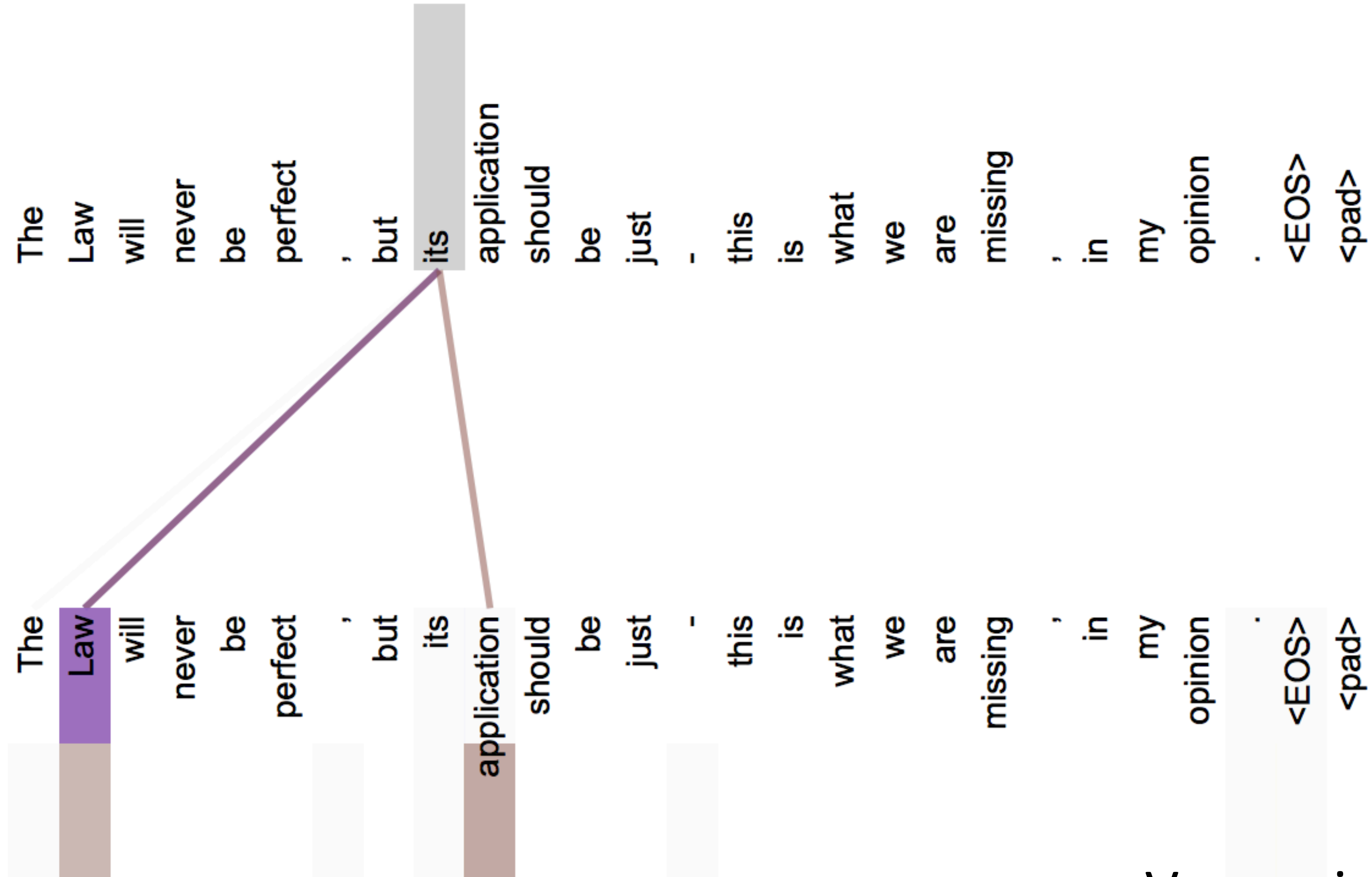
$$x'_{h,i} = \sum_{j=1}^n \alpha_{h,i,j} V_h x_j$$

Visualization



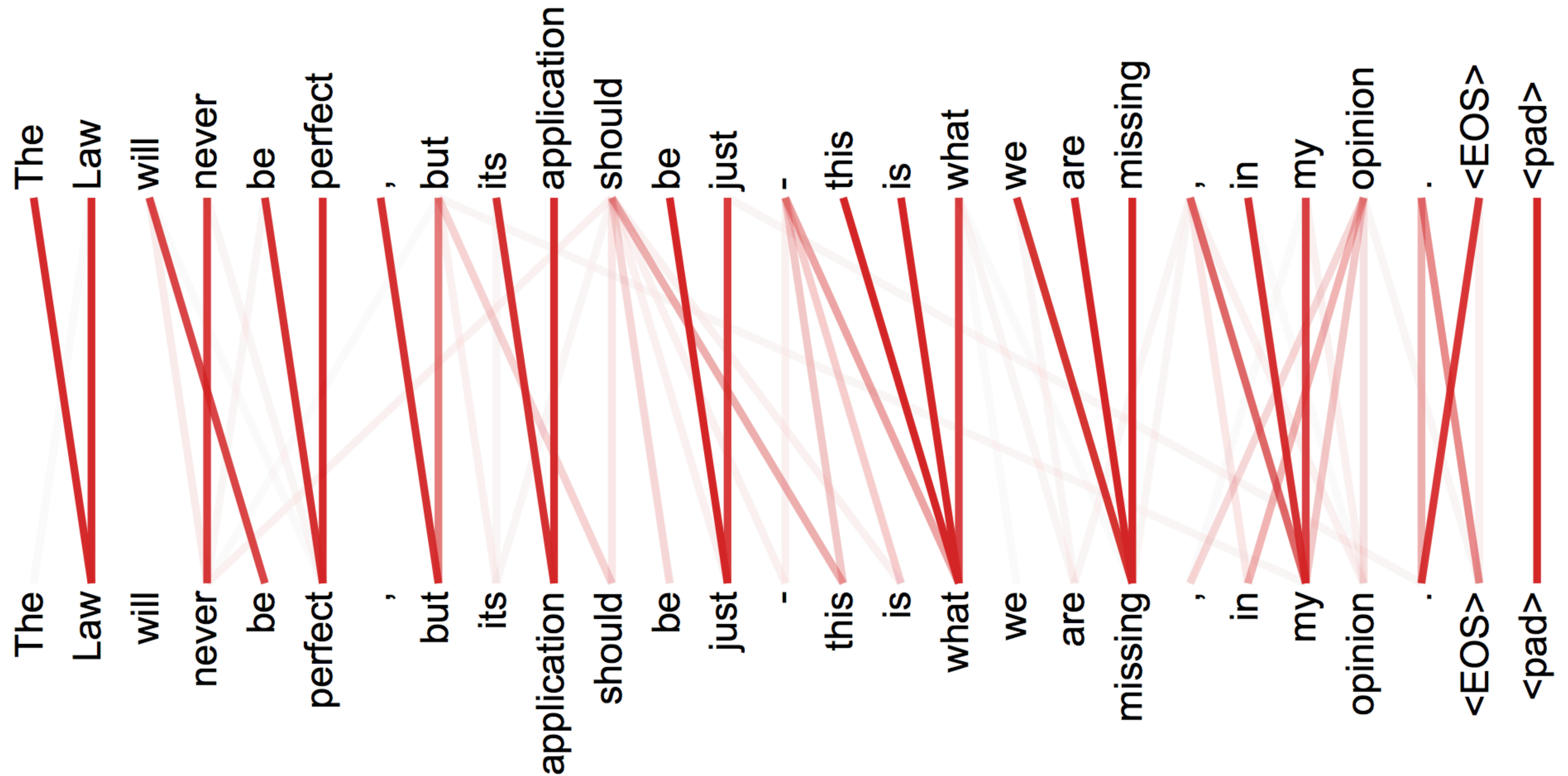
Vaswani et al. (2017)

Visualization



Vaswani et al. (2017)

Visualization



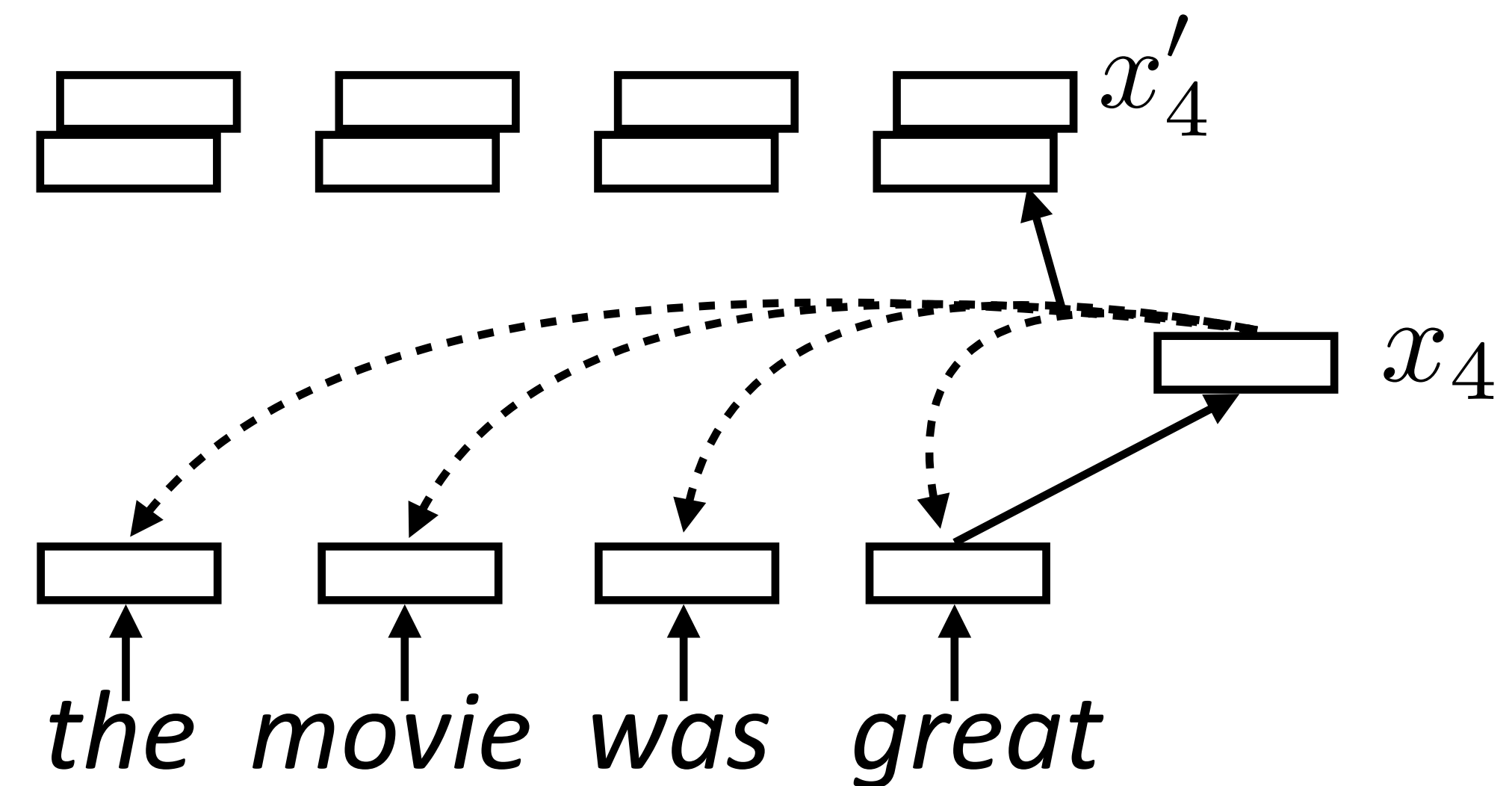
Vaswani et al. (2017)

Multi-head Self-Attention

- Each word forms a “query” which then computes attention over each word

$$\alpha_{i,j} = \text{softmax}(x_i^\top x_j) \quad \text{scalar}$$

$$x'_i = \sum_{j=1}^n \alpha_{i,j} x_j \quad \text{vector} = \text{sum of scalar} * \text{vector}$$



- Multiple “heads” use different sets of parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{h,i,j} = \text{softmax}(x_i^\top W_h x_j)$$

↖
index of attention heads

$$x'_{h,i} = \sum_{j=1}^n \alpha_{h,i,j} V_h x_j$$

Multi-Head Self Attention

- ▶ Multiple “heads” - actual implementation that uses matrix operations
- ▶ Let $X = [\text{sent len}, \text{embedding dim}]$ be the input sentence
- ▶ Query $Q = XW^Q$: these are like the **decoder hidden state** in attention
- ▶ Keys $K = XW^K$: these control what gets attended to, along with the query
- ▶ Values $V = XW^V$: these vectors get summed up to form the output

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

← dim of keys

Multi-Head Self Attention

- Multiple “heads” - actual implementation that uses matrix operations

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

← dim of keys

- Each attention head has its own set of learned parameters W^Q, W^K, W^V
- Query vector of token i: $q_i = x_i W^Q$; Key vector of token j: $k_j = x_j W^K$

$$s_{ij} = \frac{q_i k_j^\top}{\sqrt{d_k}} = \frac{(x_i W^Q)(x_j W^K)^\top}{\sqrt{d_k}} = x_i \underbrace{\left(\frac{W^Q (W^K)^\top}{\sqrt{d_k}} \right)}_{W_{\text{bilinear}} \in \mathbb{R}^{d_x \times d_x}} x_j^\top = x_i W_{\text{bilinear}} x_j^\top$$

Multi-Head Self Attention

- Multiple “heads” - actual implementation that uses matrix operations

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

← dim of keys

- Each attention head has its own set of learned parameters W^Q, W^K, W^V

- X as input sentence: $Q = XW^Q, \quad K = \underbrace{XW^K}_{d_x \times d_k}, \quad V = XW^V$

$$\frac{QK^T}{\sqrt{d_k}} = \frac{(XW^Q)(XW^K)^T}{\sqrt{d_k}} = X \underbrace{\left(\frac{W^Q(W^K)^T}{\sqrt{d_k}} \right)}_{W_{\text{bilinear}} \in \mathbb{R}^{d_x \times d_x}} X^T = XW_{\text{bilinear}}X^T$$

Vaswani et al. (2017)

Multi-Head Self Attention

- ▶ Why scaled by $\sqrt{d_k}$?

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

← dim of keys & queries

- ▶ Query vector of token i: $q_i = x_i W^Q$, Key vector of token j: $k_j = x_j W^K$
- ▶ As d_k increases, variance/std. of dot product also grows, causing softmax to become overly peaked

$$s_{ij} = q_i k_j^\top = \sum_{l=1}^{d_k} q_{il} k_{jl}$$

$$\mathbb{E}[s_{ij}] = 0$$

$$\text{Var}(s_{ij}) = d_k$$

$$\text{Std}(s_{ij}) = \sqrt{d_k}$$

Vaswani et al. (2017)

Multi-Head Self Attention

- ▶ Why scaled by $\sqrt{d_k}$?

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

← dim of keys & queries

- ▶ Query vector of token i: $q_i = x_i W^Q$, Key vector of token j: $k_j = x_j W^K$
- ▶ As d_k increases, variance/std. of dot product also grows, causing softmax to become overly peaked

$$s_{ij} = q_i k_j^\top = \sum_{l=1}^{d_k} q_{il} k_{jl}$$

$$\mathbb{E}[s_{ij}] = 0$$

$$\text{Var}(s_{ij}) = d_k$$

$$\text{Std}(s_{ij}) = \sqrt{d_k}$$

Vaswani et al. (2017)

Multi-Head Self Attention

- ▶ Each attention head has its own set of learned parameters W^Q, W^K, W^V
- ▶ H attention heads: $1, 2, \dots, H$

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

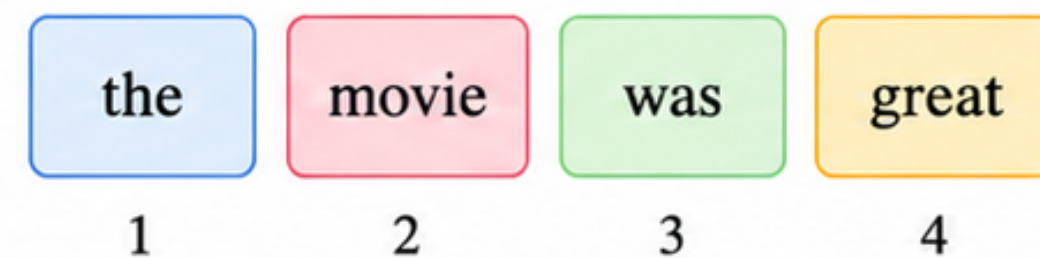
$$\text{head}_h = \text{softmax} \left(\frac{(XW_h^Q)(XW_h^K)^\top}{\sqrt{d_k}} \right) XW_h^V, \quad h = 1, \dots, H$$

Multi-Head Self Attention

- ▶ No notion of position of words/tokens in the attention or softmax
- ▶ Reordering tokens simply reorders attention weights, leaving each token's weighted sum unchanged.

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

1. Original order



2. Attention scores (query = "great")

j	word	score $s_{4j} = q_4 k_j^T / \sqrt{d_k}$
1	the	0.00
2	movie	2.48
3	was	0.69
4	great	1.61

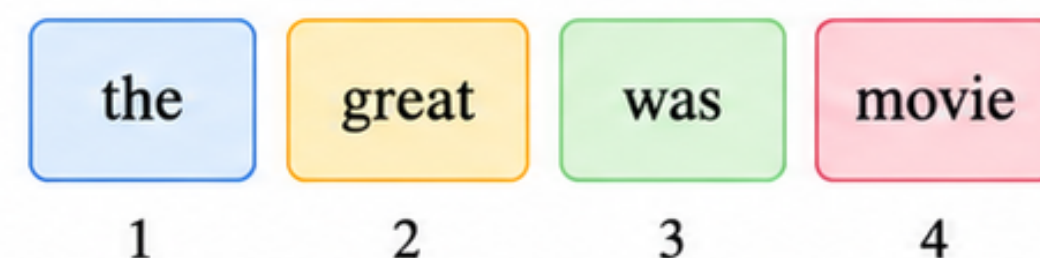
3. Softmax (attention weights)

word	α_{4j} (softmax)
the	0.05
movie	0.60
was	0.10
great	0.25

4. Output for "great"

$$x'_4 = \sum_{j=1}^4 \alpha_{4j} x_j$$
$$= 0.05x_{\text{the}} + 0.60x_{\text{movie}} + 0.10x_{\text{was}} + 0.25x_{\text{great}}$$

2. Reordered order



2. Attention scores (query = "great")

j	word	score $s_{4j} = q_4 k_j^T / \sqrt{d_k}$
1	the	0.00
2	great	1.61
3	was	0.69
4	movie	2.48

3. Softmax (attention weights)

word	α_{4j} (softmax)
the	0.05
great	0.25
was	0.10
movie	0.60

4. Output for "great"

$$x'_4 = \sum_{j=1}^4 \alpha_{4j} x_j$$
$$= 0.05x_{\text{the}} + 0.60x_{\text{movie}} + 0.10x_{\text{was}} + 0.25x_{\text{great}}$$

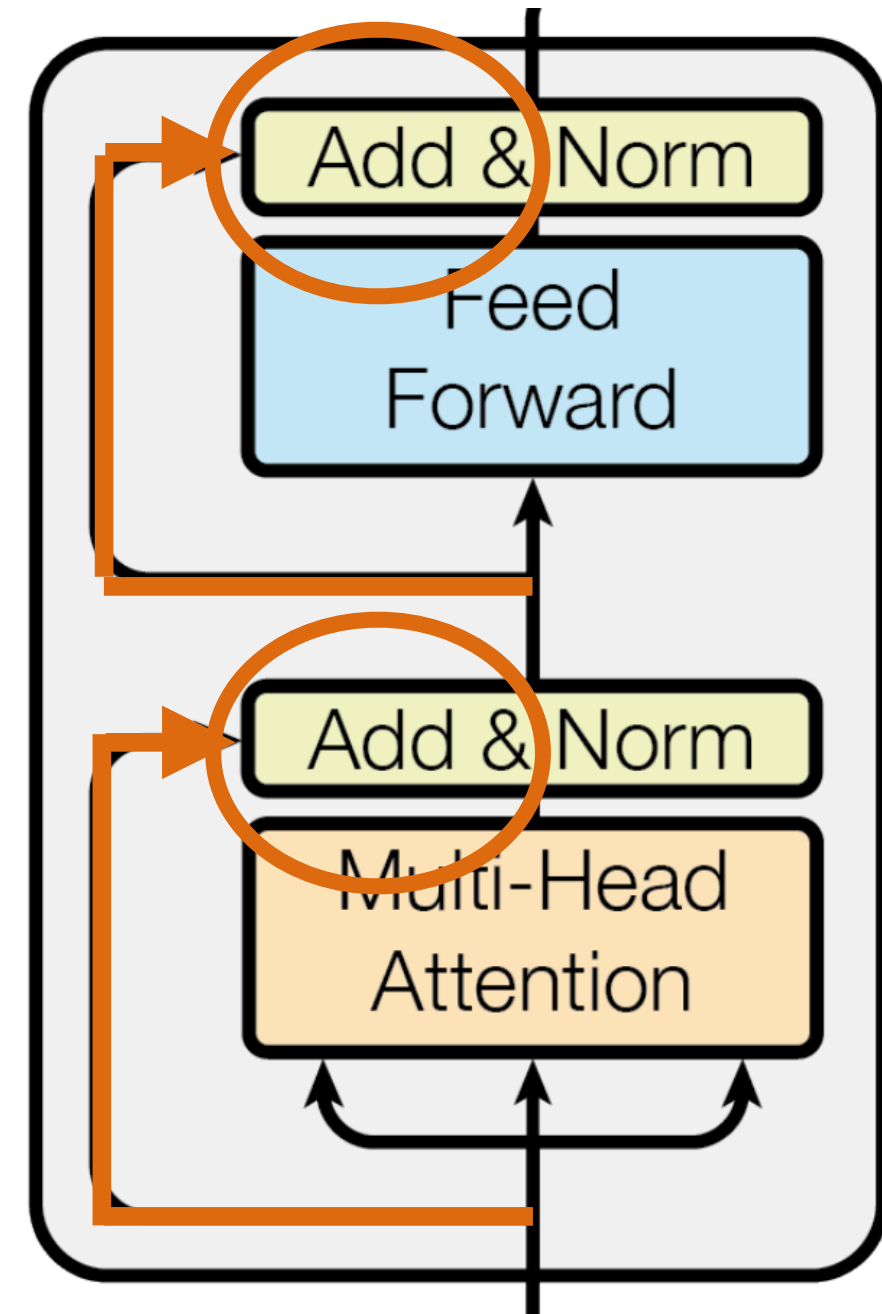
Properties of Self-Attention

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- ▶ n = sentence length, d = hidden dim, k = kernel size, r = restricted neighborhood size
- ▶ **Quadratic complexity**, but $O(1)$ sequential operations (not linear like in RNNs) and $O(1)$ “path” for words to inform each other

Transformer

Transformers



- ▶ Alternate multi-head self-attention layers and feedforward layers
- ▶ **Residual** connections let the model “skip” each layer — these are particularly useful for training deep networks

Encoder Layer 6

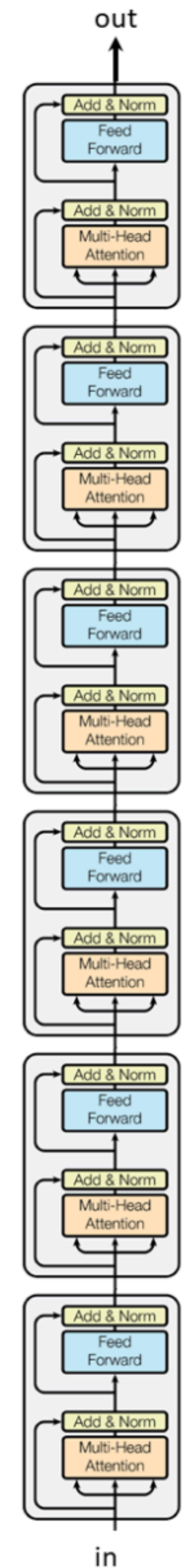
Encoder Layer 5

Encoder Layer 4

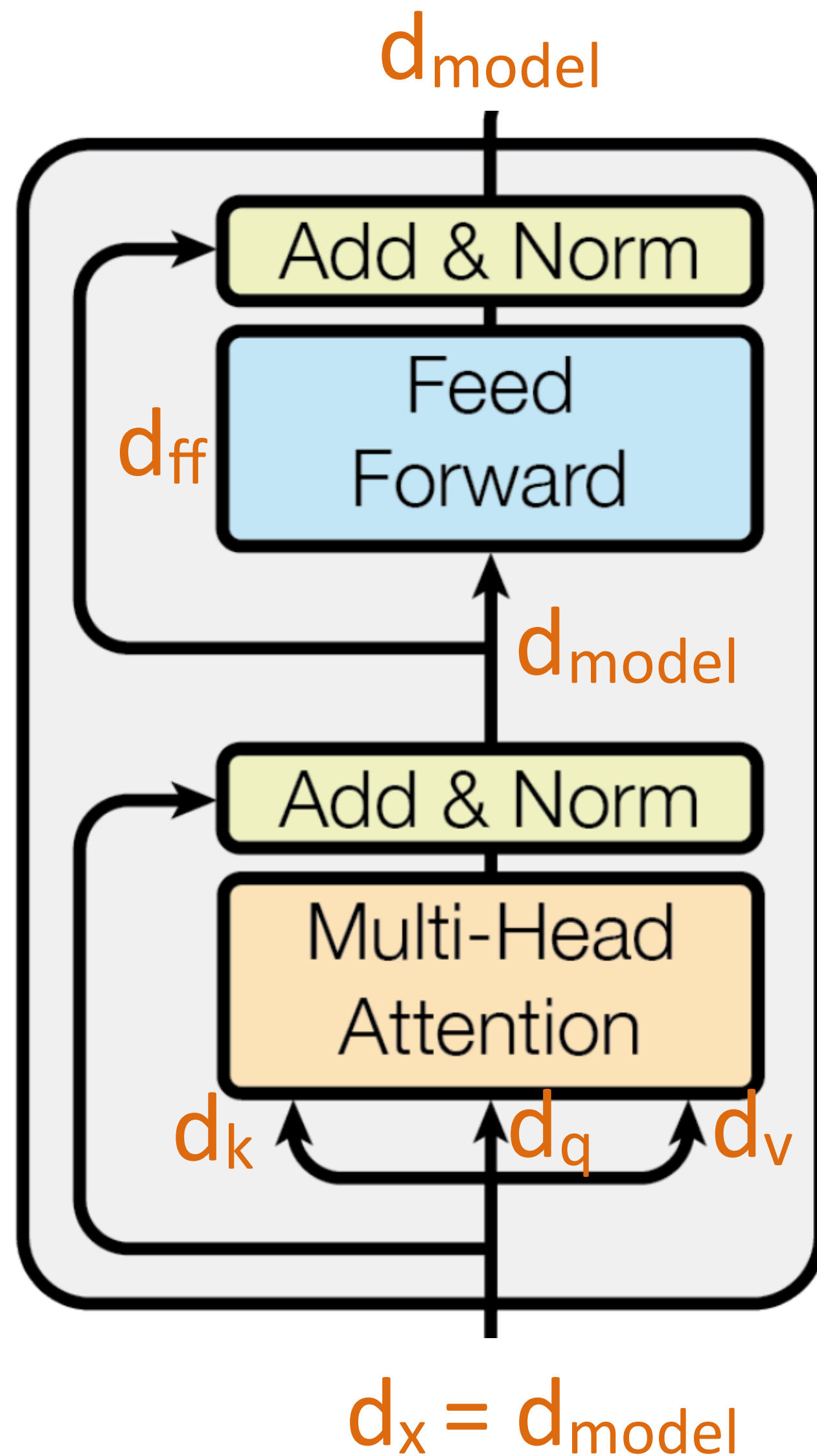
Encoder Layer 3

Encoder Layer 2

Encoder Layer 1



Transformers



$$X' = \text{LayerNorm}(Z + \text{FFN}(Z))$$

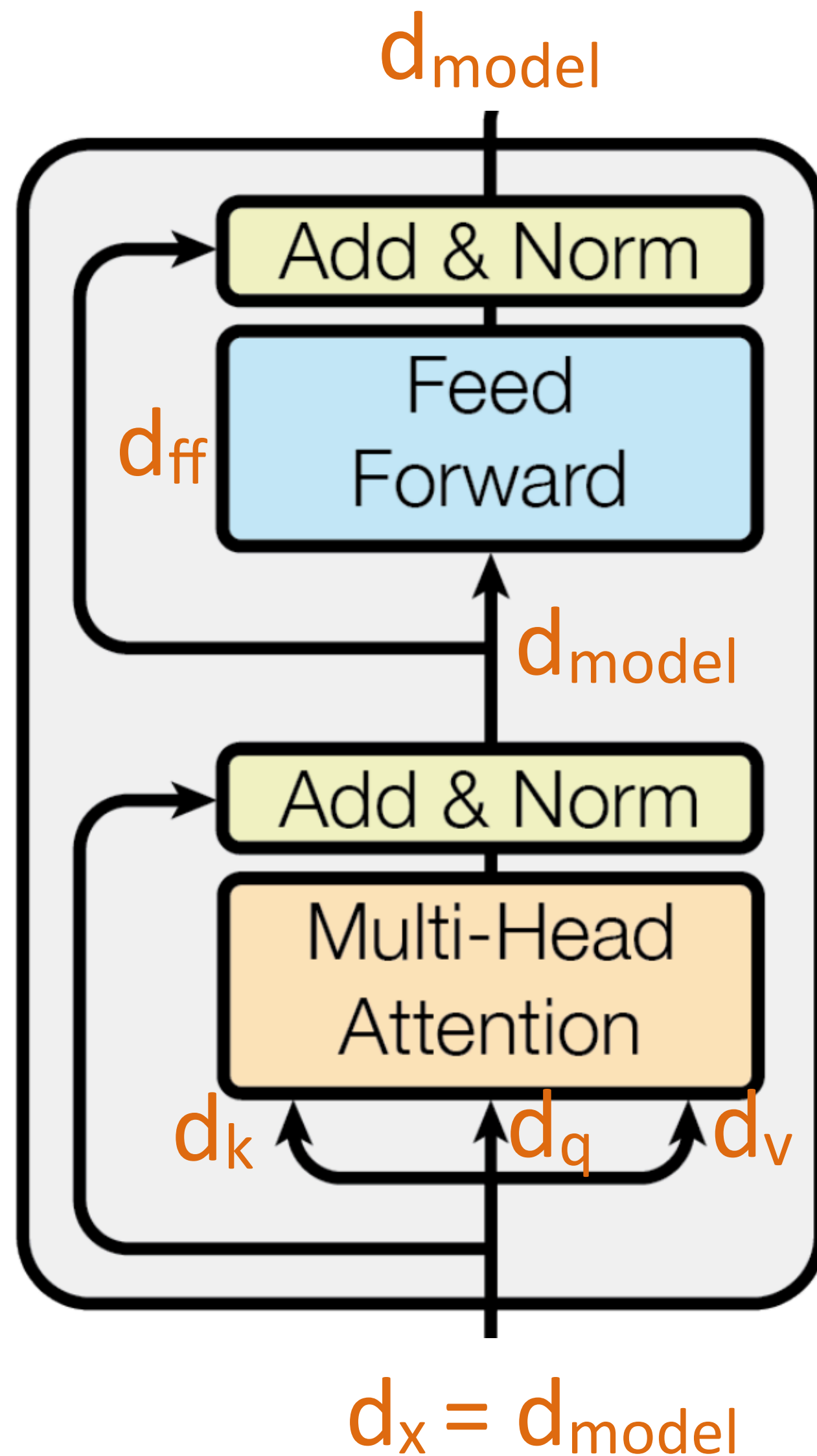
$$Z = \text{LayerNorm}(X + \text{MultiHead}(X))$$

$$X \in \mathbb{R}^{n \times d_{\text{model}}}$$

- ▶ Alternate multi-head self-attention layers and feedforward layers that operate over each word individually

← post-LN (original Transformer)

Transformers



$$X' = \text{LayerNorm}(Z + \text{FFN}(Z))$$

► or $X' = Z + \text{LayerNorm}(\text{FFN}(Z))$

- Alternate multi-head self-attention layers and feedforward layers that operate over each word individually

$$Z = \text{LayerNorm}(X + \text{MultiHead}(X))$$

► or $Z = X + \text{LayerNorm}(\text{MultiHead}(X))$

← post-LN (original Transformer)

← pre-LN (modern approach)

$$X \in \mathbb{R}^{n \times d_{\text{model}}}$$

Transformers

- ▶ Alternate multi-head self-attention layers and feedforward layers (w/ ReLU activation) that operate over each word individually

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

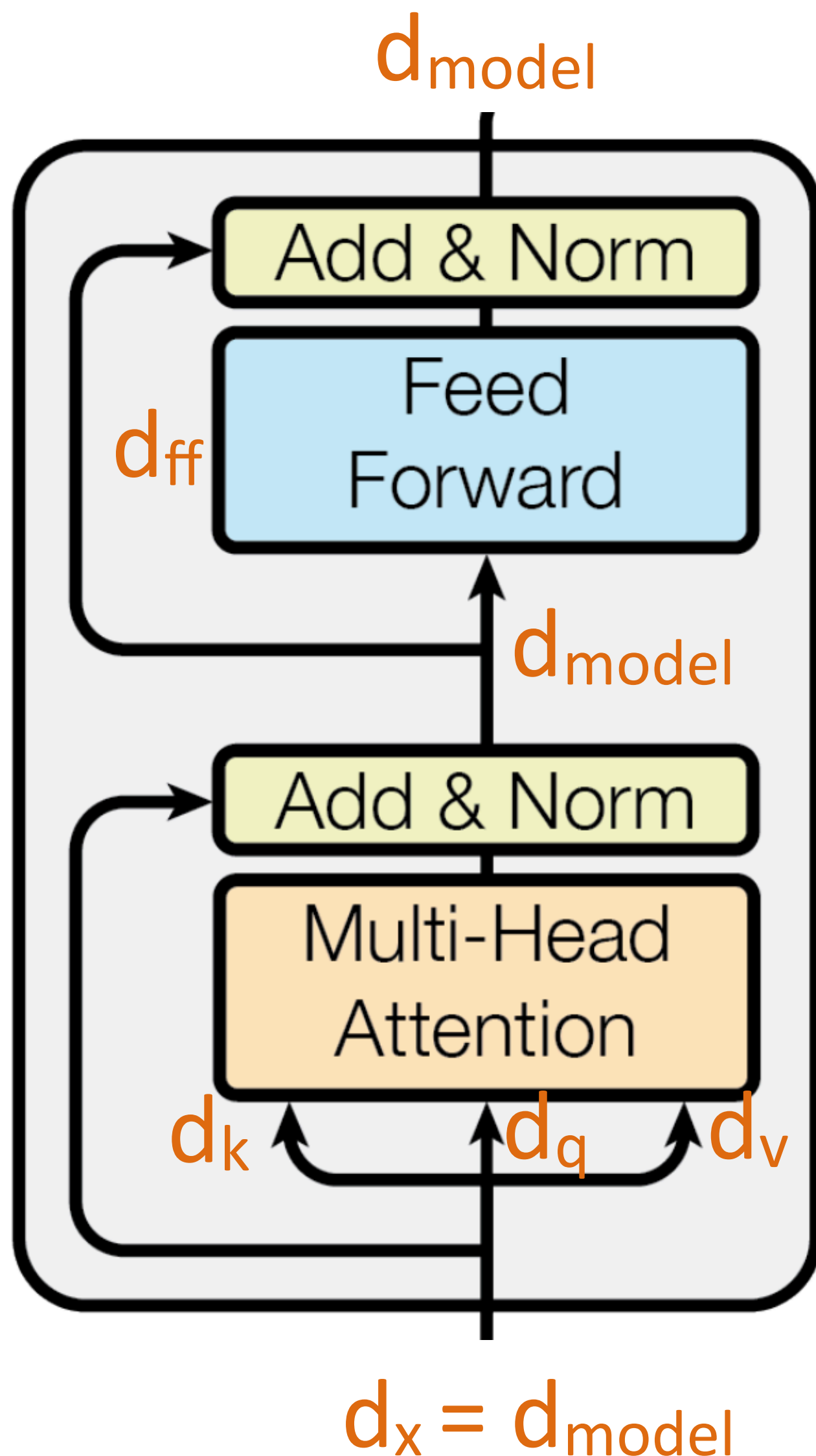
$$h = 8$$

$$d_x = 512$$

$$d_{\text{ff}} = 2048$$

$$d_k = d_q = d_{\text{model}} / h = 64$$

$$d_v = d_{\text{model}} / h = 64$$



$$N = 6$$

Encoder Layer 6

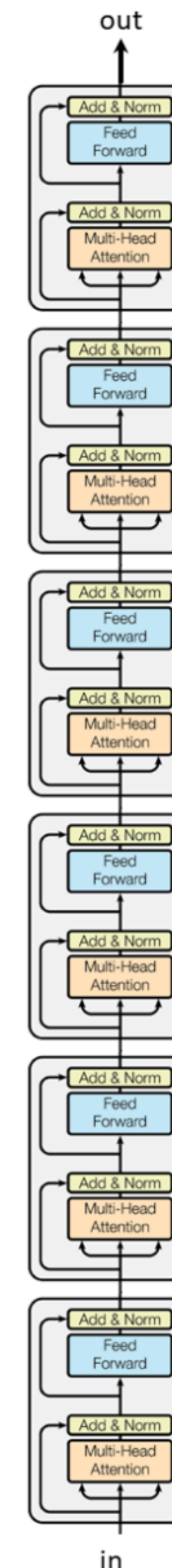
Encoder Layer 5

Encoder Layer 4

Encoder Layer 3

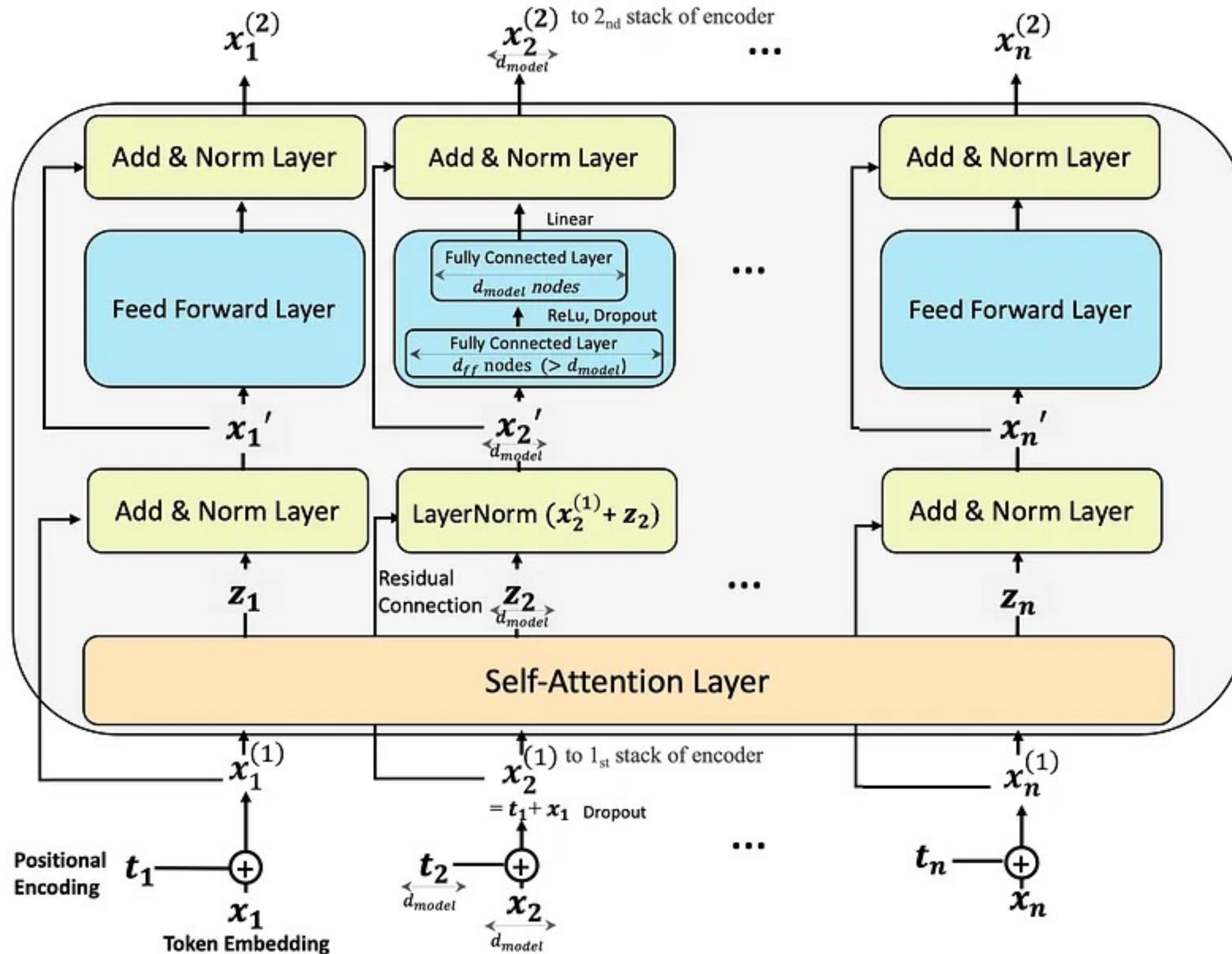
Encoder Layer 2

Encoder Layer 1



Vaswani et al. (2017)

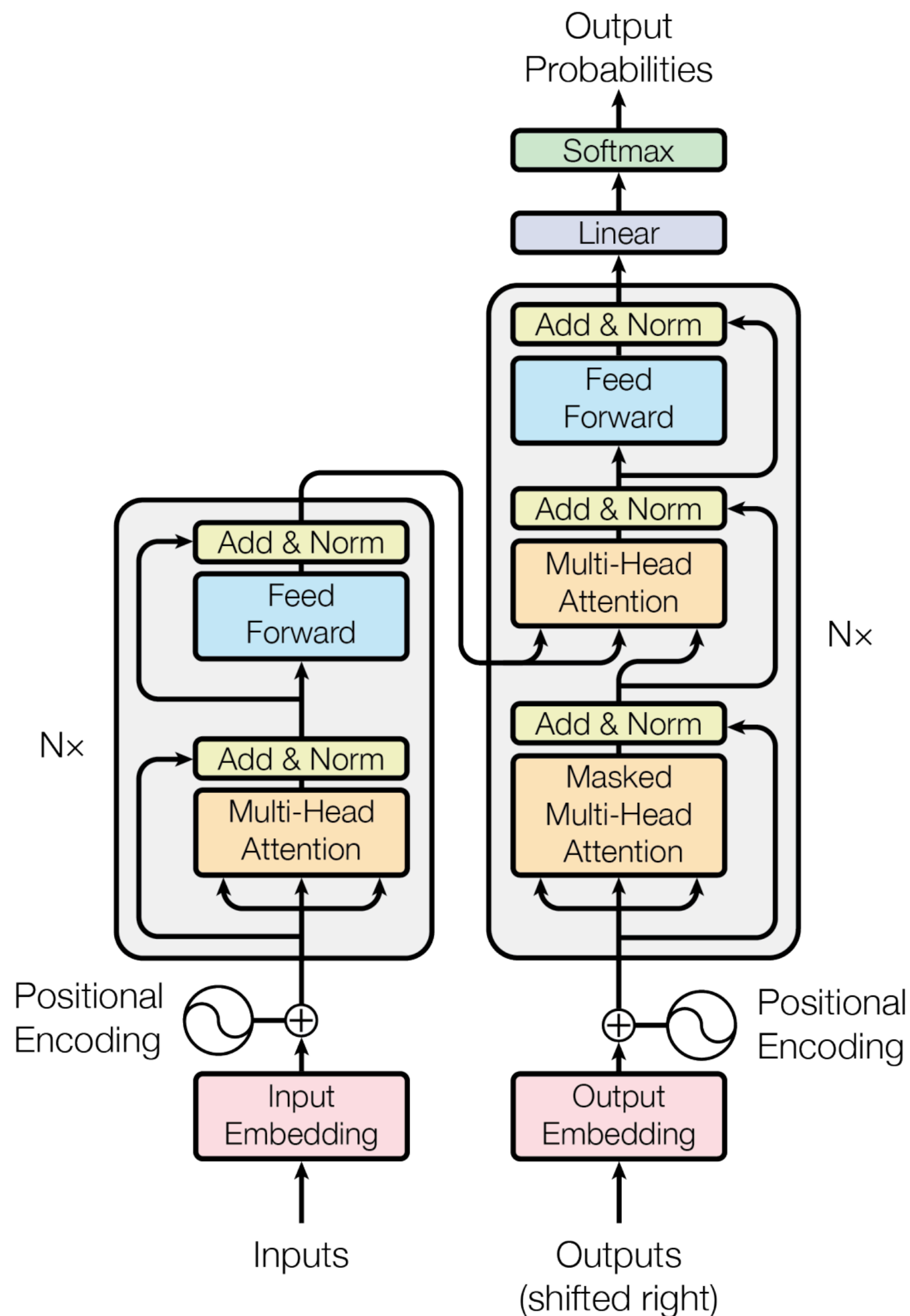
Transformers



- ▶ Alternate multi-head self-attention layers and feedforward layers that operate over each word individually

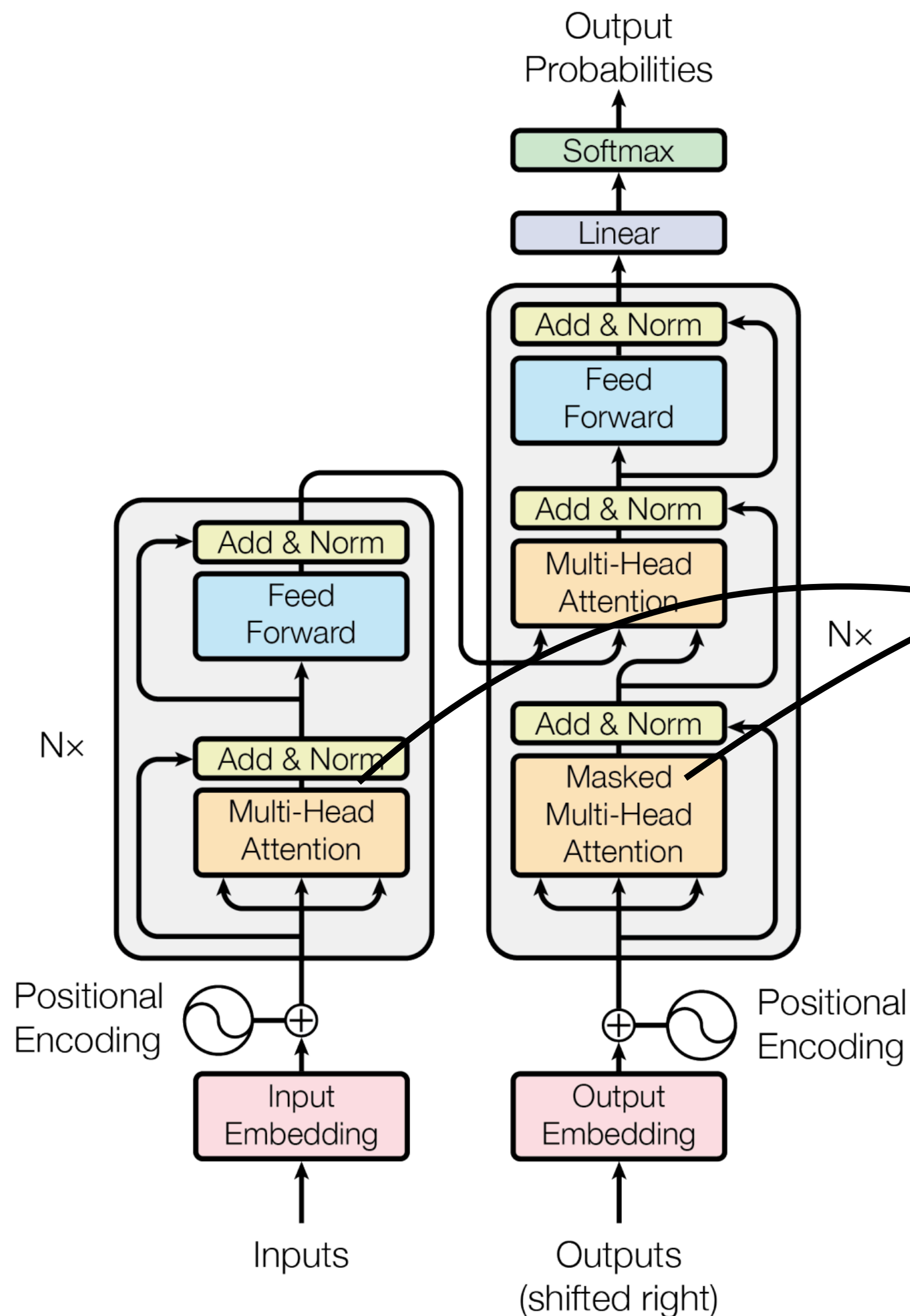
Transformer encoder-decoder

- ▶ The original Transformer used an encoder-decoder architecture for machine translation.
- ▶ Transformers can be used as encoders, decoders, or both, similar to LSTMs and CNNs.



Transformer encoder-decoder

- ▶ The original Transformer used an encoder-decoder architecture for machine translation.
- ▶ Transformers can be used as encoders, decoders, or both, similar to LSTMs and CNNs.



Keys →

	The	movie	was	great
The	0.25	0.55	0.13	0.07
movie	0.10	0.30	0.12	0.48
was	0.08	0.42	0.35	0.15
great	0.05	0.60	0.10	0.25

Queries →

self-attention
(encoder)

Keys →

	Le	film	était	bon
Le	1.00	0.00	0.00	0.00
film	0.58	0.42	0.00	0.00
était	0.16	0.49	0.35	0.00
bon	0.08	0.54	0.14	0.24

Queries →

Masked: 0 attention (light blue) Allowed (dark blue)

masked self-attention
(decoder)