# CS378: Natural Language Processing
# Lecture 16: Transformers

Greg Durrett

TEXAS
The University of Texas at Austin

# Multi-Head Self-Attention

# Multi-Head Self Attention

- Multiple "heads" analogous to different convolutional filters
- Let $E$ = [sent len, embedding dim] be the input sentence. This will be passed through three different linear layers to produce three mats:
  - Query $Q = EW^Q$: these are like the **decoder hidden state** in attention
  - Keys $K = EW^K$: these control what gets attended to, along with the query
  - Values $V = EW^V$: these vectors get summed up to form the output

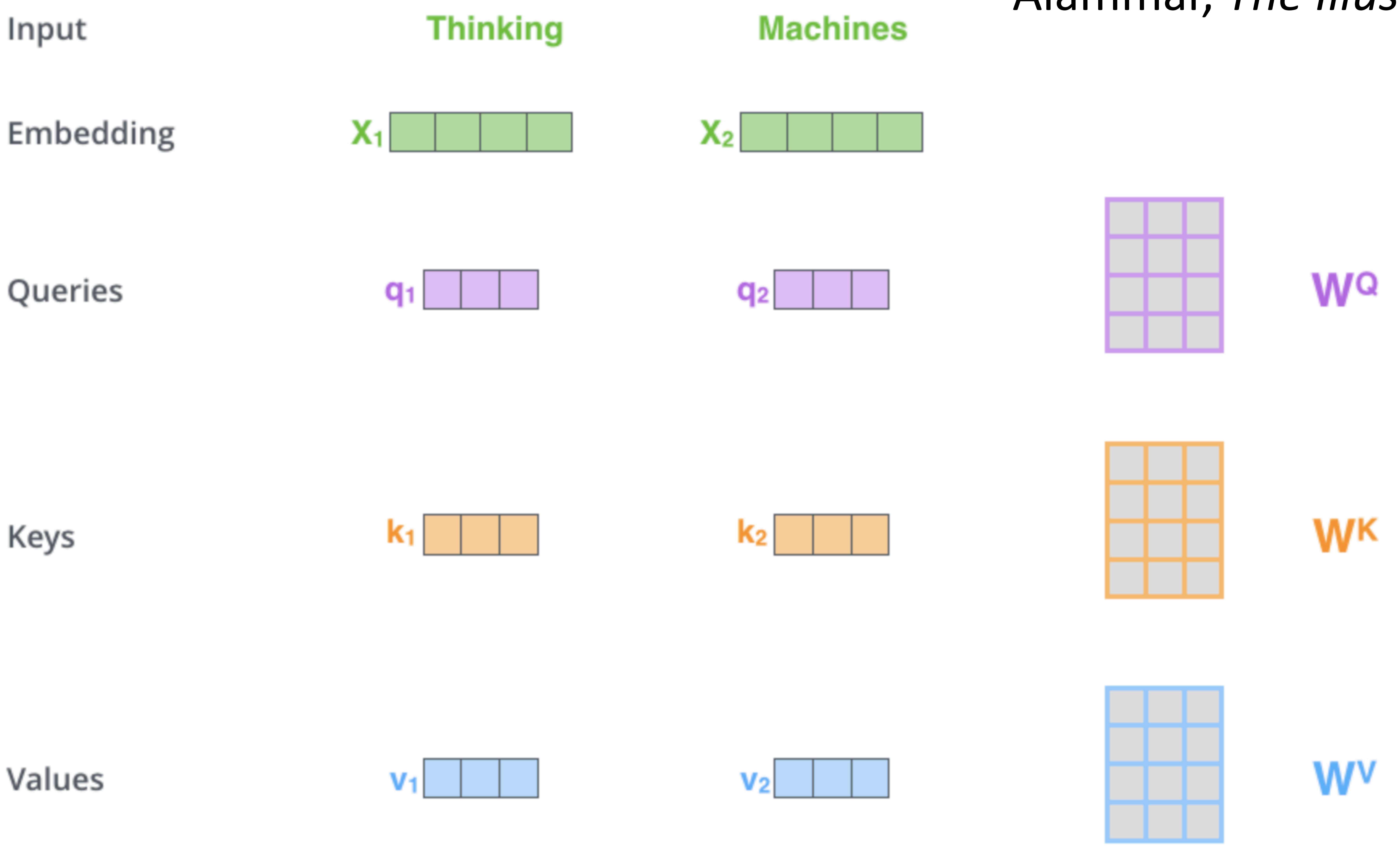
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

dim of keys

Vaswani et al. (2017)

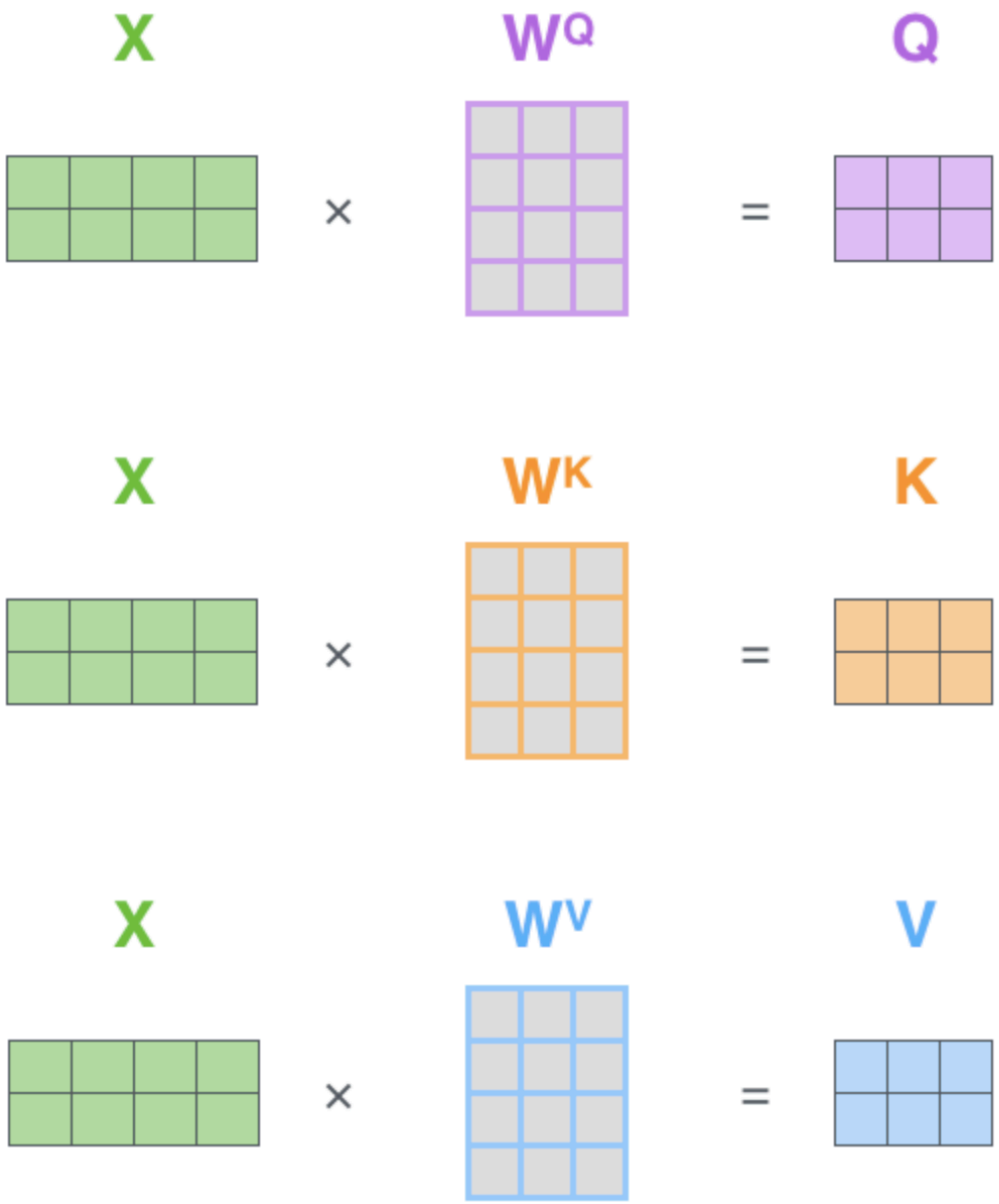
# Self-Attention

Alammar, *The Illustrated Transformer*

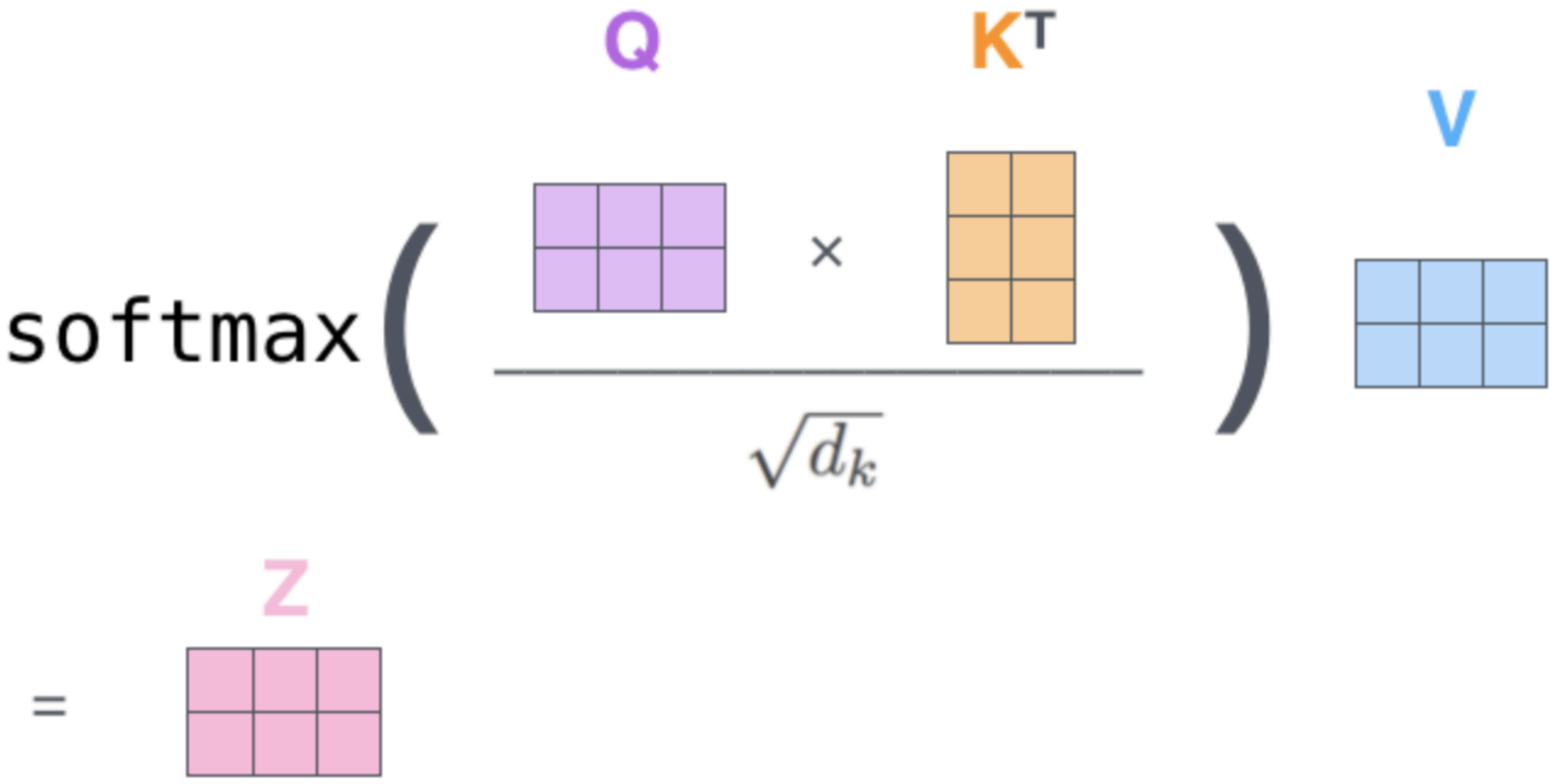


# Self-Attention

Alammar, *The Illustrated Transformer*




sent len x sent len (attn for each word to each other)



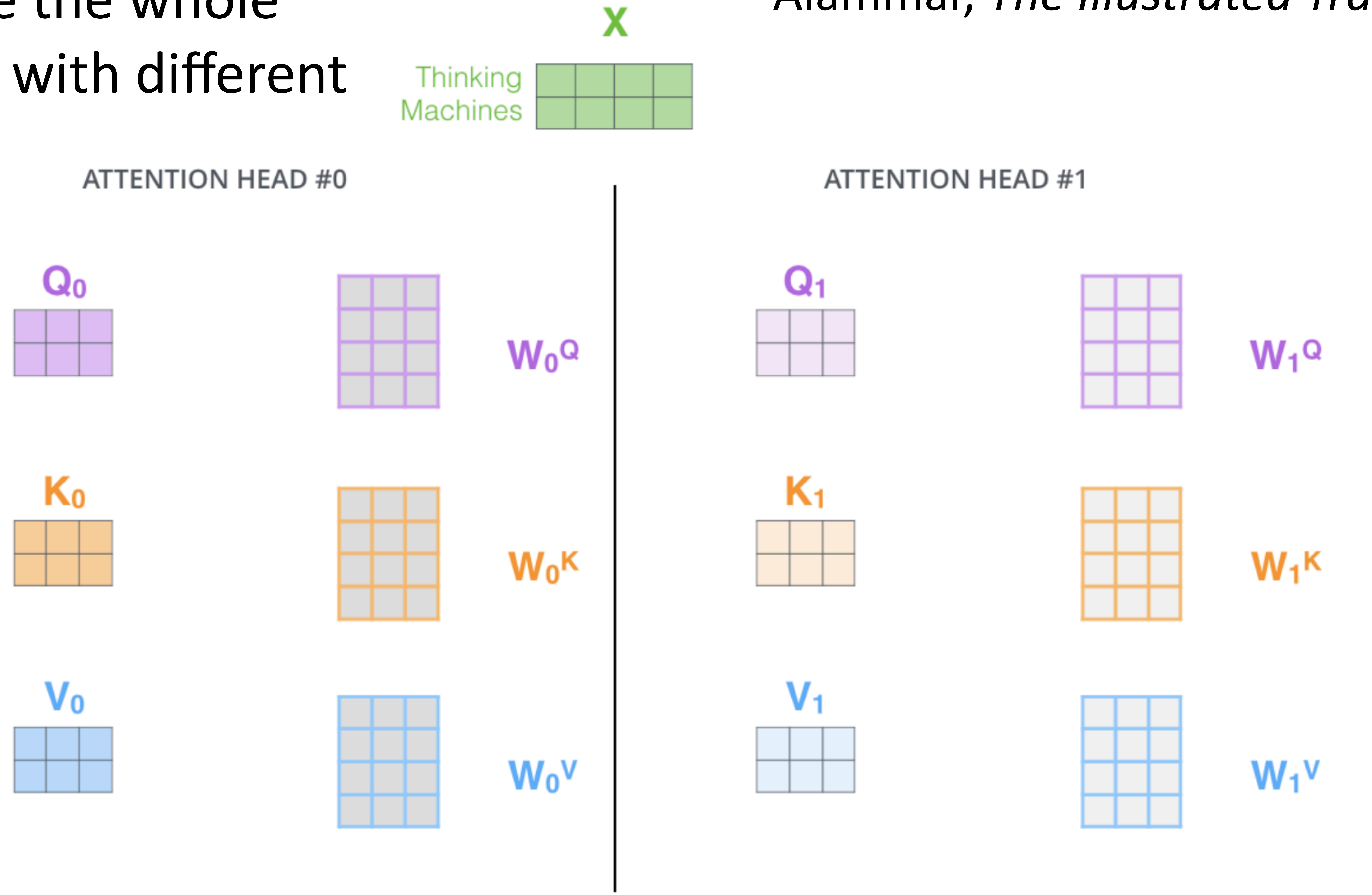

sent len x hidden dim

Z is a weighted combination of V rows

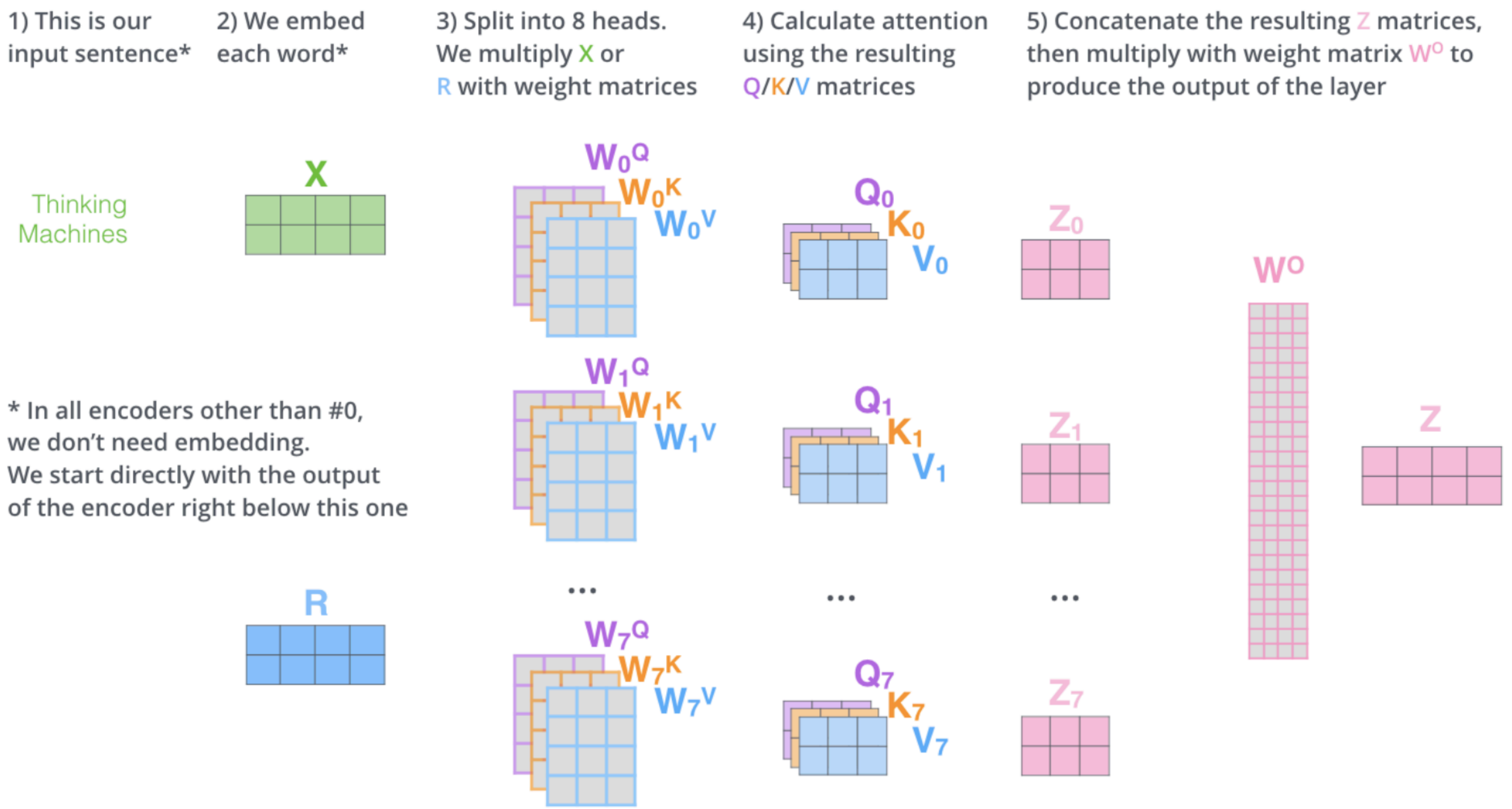
# Multi-head Self-Attention

Just duplicate the whole computation with different weights:

Alammar, *The Illustrated Transformer*



# Multi-head Self-Attention



# 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

Vaswani et al. (2017)

# Transformers

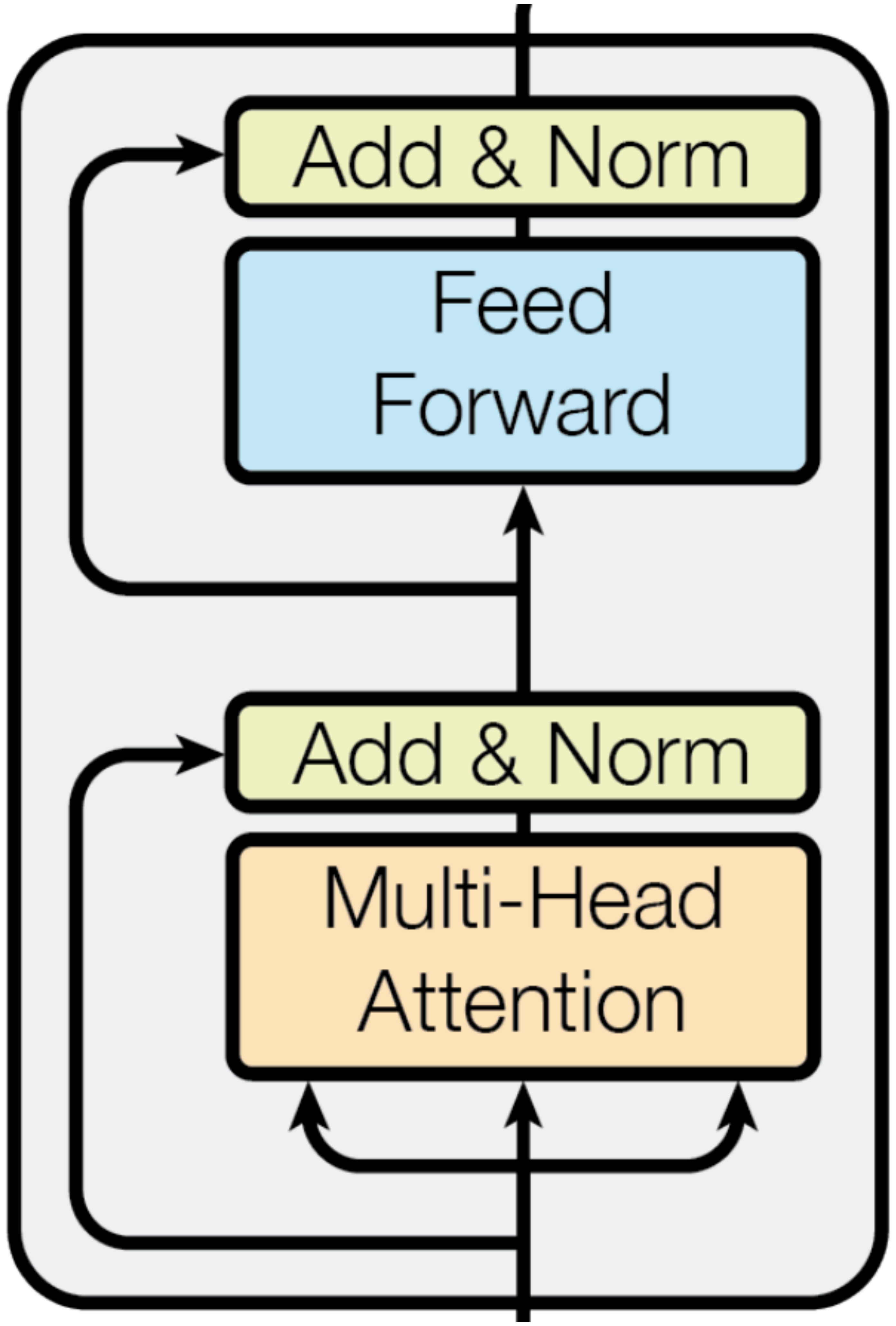
# Architecture

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

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

  - These feedforward layers are where most of the parameters are

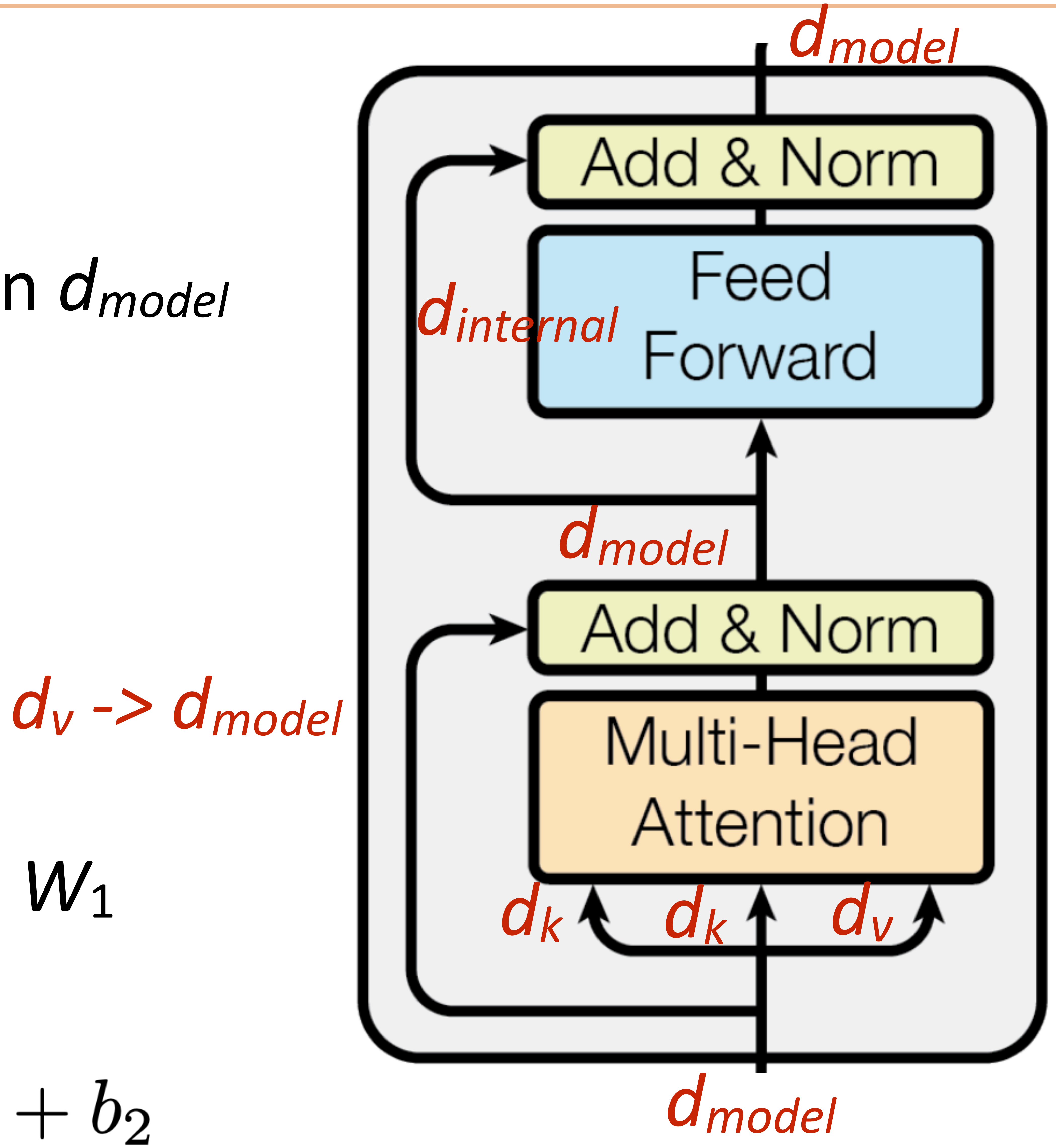
- Residual connections in the model: input of a layer is added to its output
- Layer normalization: controls the scale of different layers in very deep networks (not needed in A4)



# Dimensions

- Vectors: $d_{model}$
- Queries/keys: $d_k$, always smaller than $d_{model}$
- Values: separate dimension $d_v$, output is multiplied by $W^O$ which is $d_v$ x $d_{model}$ so we can get back to $d_{model}$ before the residual
- FFN can explode the dimension with $W_1$ and collapse it back with $W_2$

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




Vaswani et al. (2017)

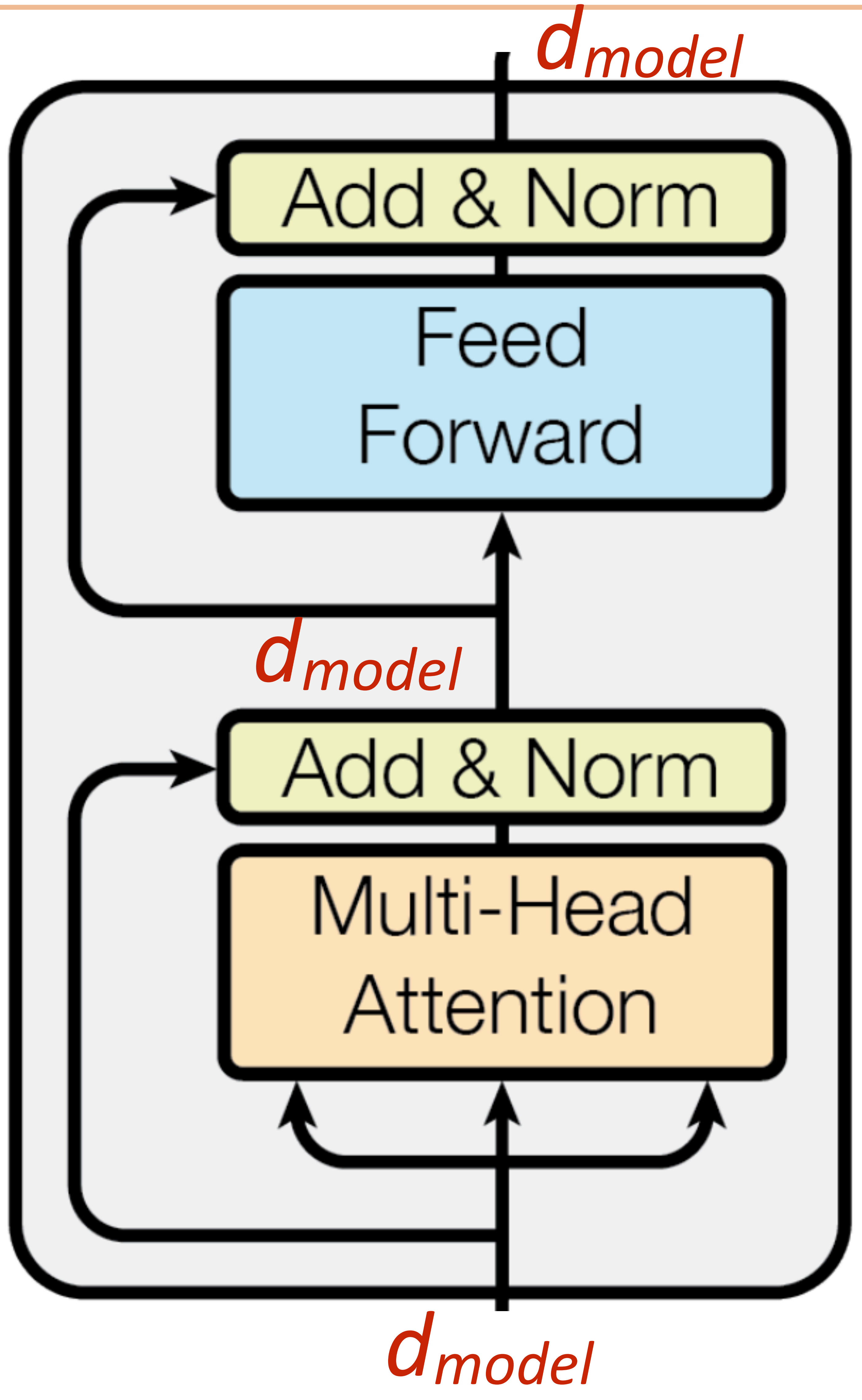
# Transformer Architecture

| | $N$ | $d_{\text{model}}$ | $d_{\text{ff}}$ | $h$ | $d_k$ | $d_v$ |
|---|---|---|---|---|---|---|
| base | 6 | 512 | 2048 | 8 | 64 | 64 |

- From Vaswani et al.

| Model Name | $n_{\text{params}}$ | $n_{\text{layers}}$ | $d_{\text{model}}$ | $n_{\text{heads}}$ | $d_{\text{head}}$ |
|---|---|---|---|---|---|
| GPT-3 Small | 125M | 12 | 768 | 12 | 64 |
| GPT-3 Medium | 350M | 24 | 1024 | 16 | 64 |
| GPT-3 Large | 760M | 24 | 1536 | 16 | 96 |
| GPT-3 XL | 1.3B | 24 | 2048 | 24 | 128 |
| GPT-3 2.7B | 2.7B | 32 | 2560 | 32 | 80 |
| GPT-3 6.7B | 6.7B | 32 | 4096 | 32 | 128 |
| GPT-3 13B | 13.0B | 40 | 5140 | 40 | 128 |
| GPT-3 175B or "GPT-3" | 175.0B | 96 | 12288 | 96 | 128 |

- From GPT-3; $d_{head}$ is our $d_k$



# Transformer Architecture

| 1 | description | FLOPs / update | % FLOPS MHA | % FLOPS FFN | % FLOPS attn | % FLOPS logit |
|---|---|---|---|---|---|---|
| 8 | OPT setups | | | | | |
| 9 | 760M | 4.3E+15 | 35% | 44% | 14.8% | 5.8% |
| 10 | 1.3B | 1.3E+16 | 32% | 51% | 12.7% | 5.0% |
| 11 | 2.7B | 2.5E+16 | 29% | 56% | 11.2% | 3.3% |
| 12 | 6.7B | 1.1E+17 | 24% | 65% | 8.1% | 2.4% |
| 13 | 13B | 4.1E+17 | 22% | 69% | 6.9% | 1.6% |
| 14 | 30B | 9.0E+17 | 20% | 74% | 5.3% | 1.0% |
| 15 | 66B | 9.5E+17 | 18% | 77% | 4.3% | 0.6% |
| 16 | 175B | 2.4E+18 | 17% | 80% | 3.3% | 0.3% |

Credit: Stephen Roller on Twitter

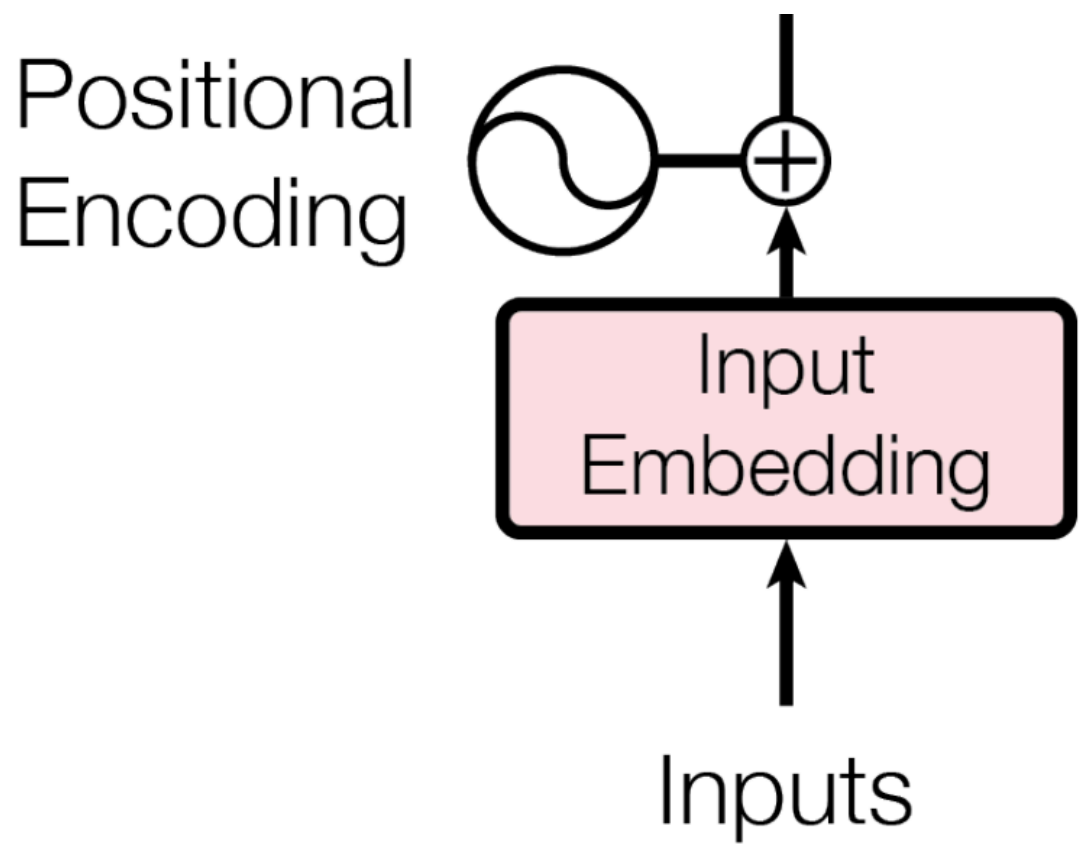
# Transformers: Position Sensitivity

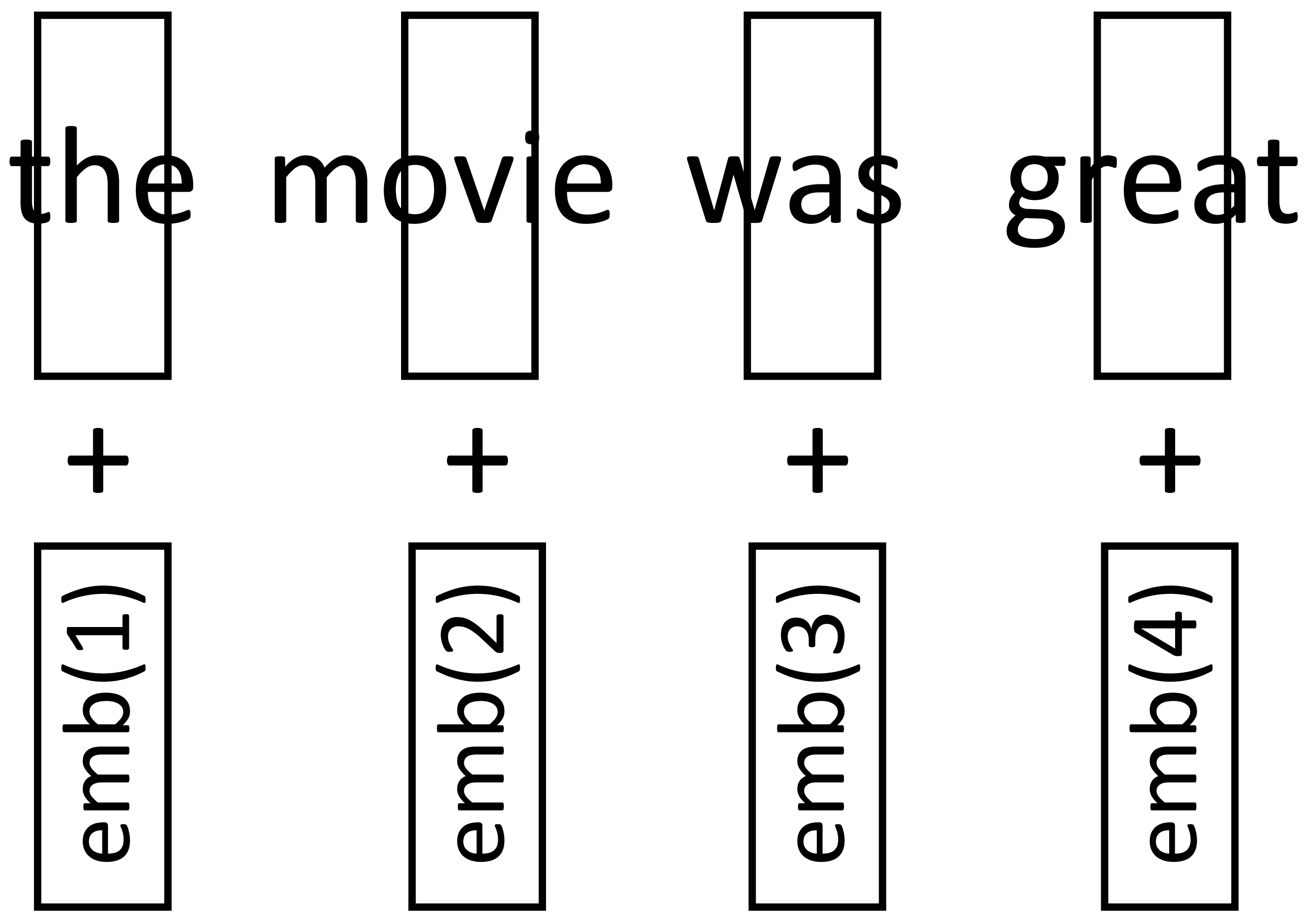
*The ballerina is very excited that she will dance in the show.*

- If this is in a longer context, we want words to attend *locally*
- But transformers have *no notion of position* by default

Vaswani et al. (2017)

# Transformers: Position Sensitivity



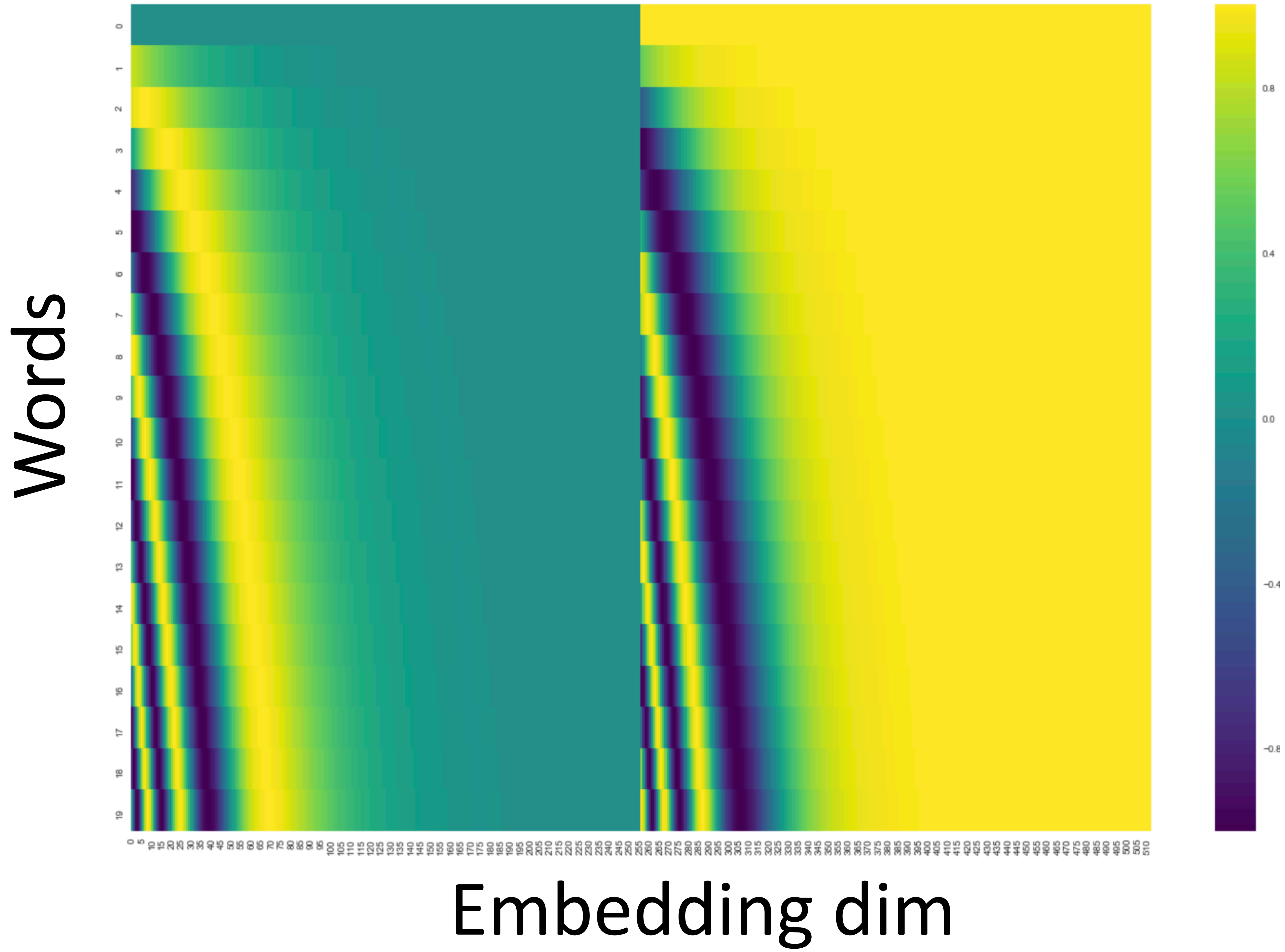




- Encode each sequence position as an integer, add it to the word embedding vector
- Why does this work?

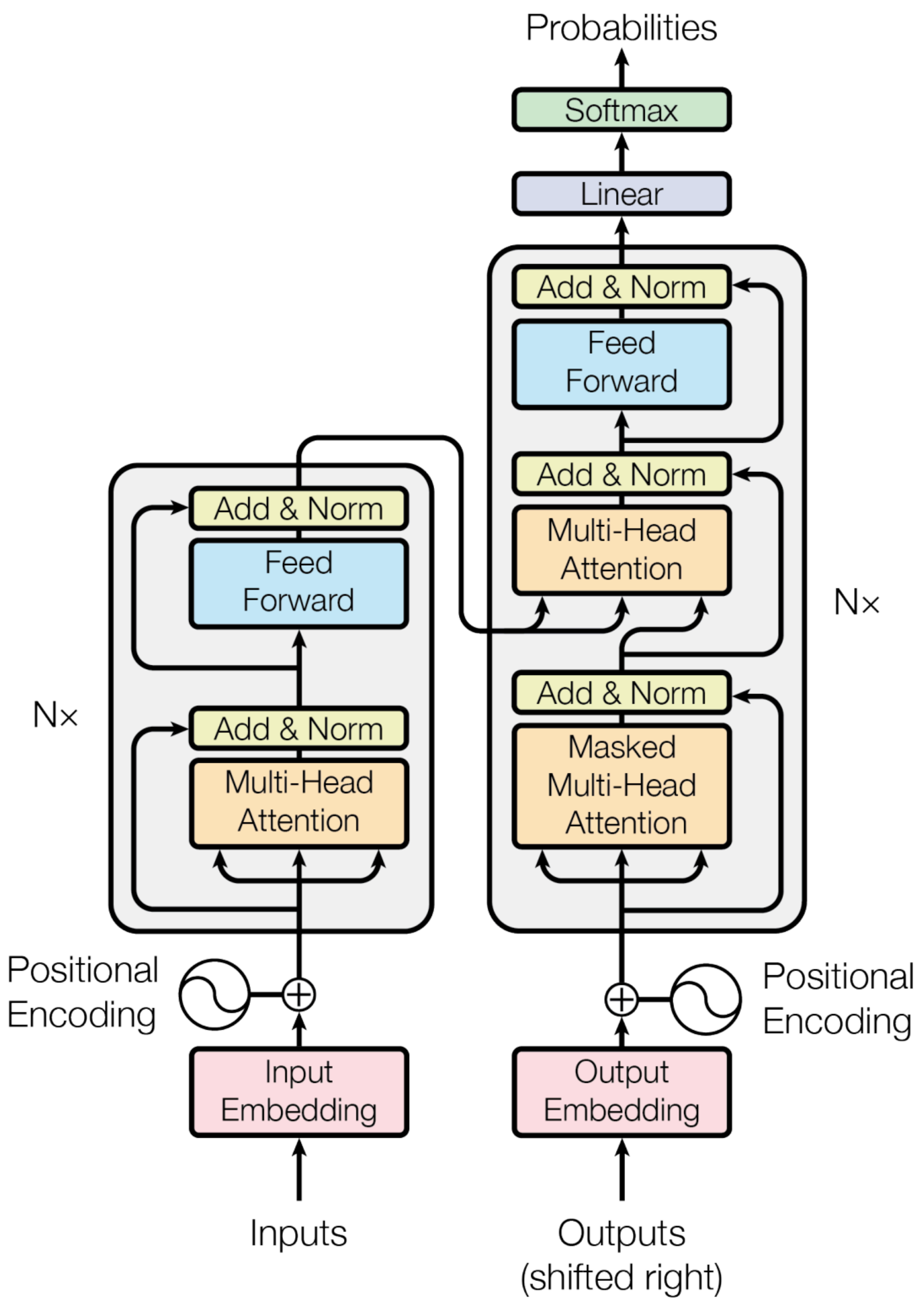
# Transformers

Alammar, *The Illustrated Transformer*

- Alternative from Vaswani et al.: sines/cosines of different frequencies (closer words get higher dot products by default)



# Transformers: Complete Model




- Original Transformer paper presents an **encoder-decoder** model
- Right now we don't need to think about both of these parts — will return in the context of MT
- Can turn the encoder into a decoder-only model through use of a triangular causal attention mask (only allow attention to previous tokens)

Vaswani et al. (2017)