Transformer + Course Projects

Wei Xu

(many slides from Greg Durrett)

Administrivia

- 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/

Transformers

Recap: Self-Attention

- Assume we're using GloVe — what do we want our neural network to do?

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

- What words need to be contextualized here?
  - Pronouns need to look at antecedents
  - Ambiguous words should look at context
  - Words should look at syntactic parents/children
- Problem: LSTMs and CNNs don't do this

Vaswani et al. (2017)

Recap: Self-Attention

- Want:

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

- LSTMs/CNNs: tend to look at local context

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

- To appropriately contextualize embeddings, we need to pass information over long distances dynamically for each word

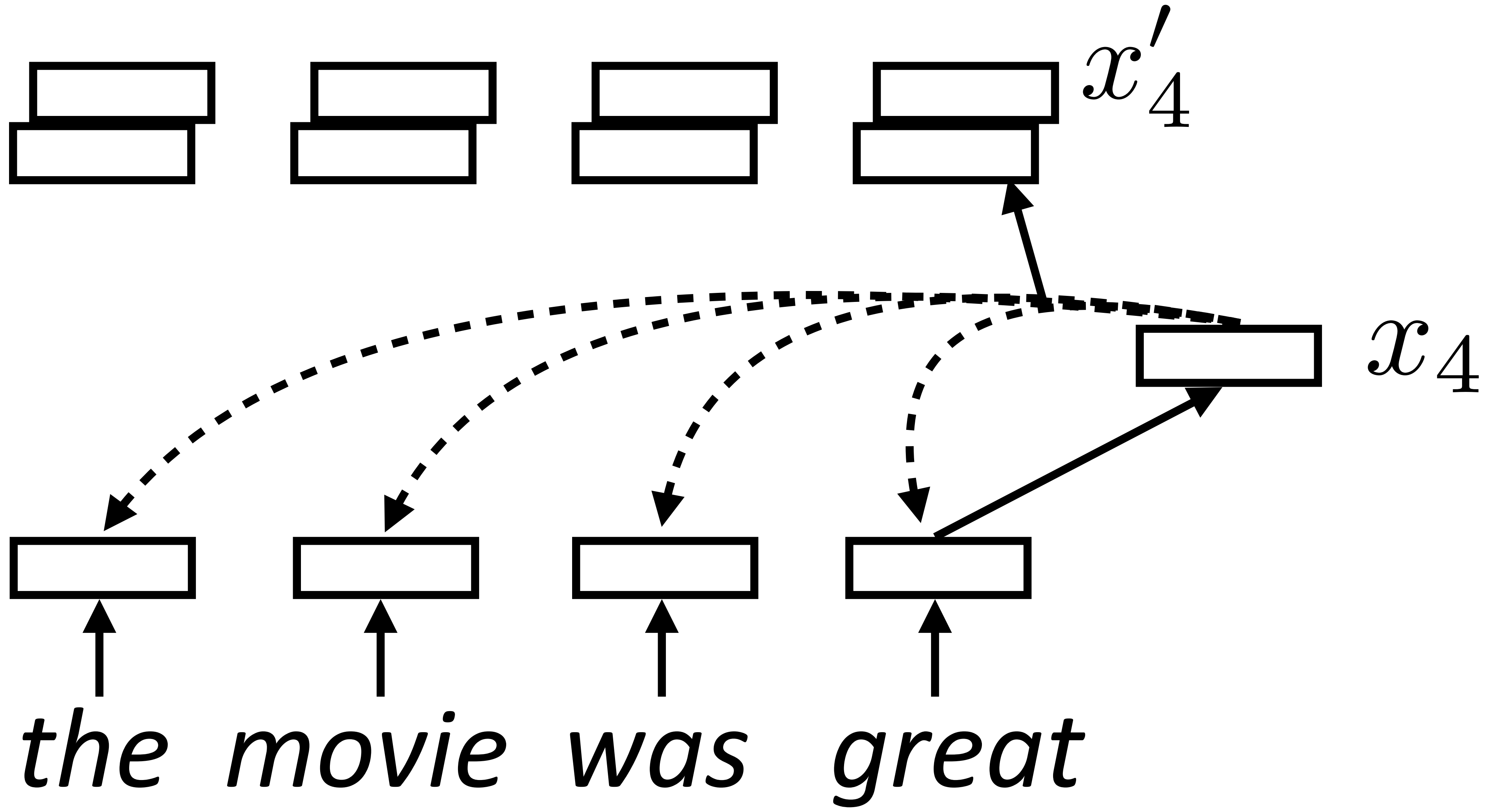
Vaswani et al. (2017)

Recap: Self-Attention

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

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

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



- Multiple “heads” analogous to different convolutional filters. Use parameters W_k and V_k to get different attention values + transform vectors

$$\alpha_{k,i,j} = \text{softmax}(x_i^\top W_k x_j) \quad x'_{k,i} = \sum_{j=1}^{n} \alpha_{k,i,j} V_k x_j$$

Vaswani et al. (2017)

Recap: What can self-attention do?

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

0	0.5	0	0	0.1	0.1	0	0.1	0.2	0	0	0

0	0.1	0	0	0	0	0	0	0.5	0	0.4	0

- Attend nearby + to semantically related terms
- Why multiple heads? Softmaxes end up being peaked, single distribution cannot easily put weight on multiple things

Vaswani et al. (2017)

Multi-Head Self Attention

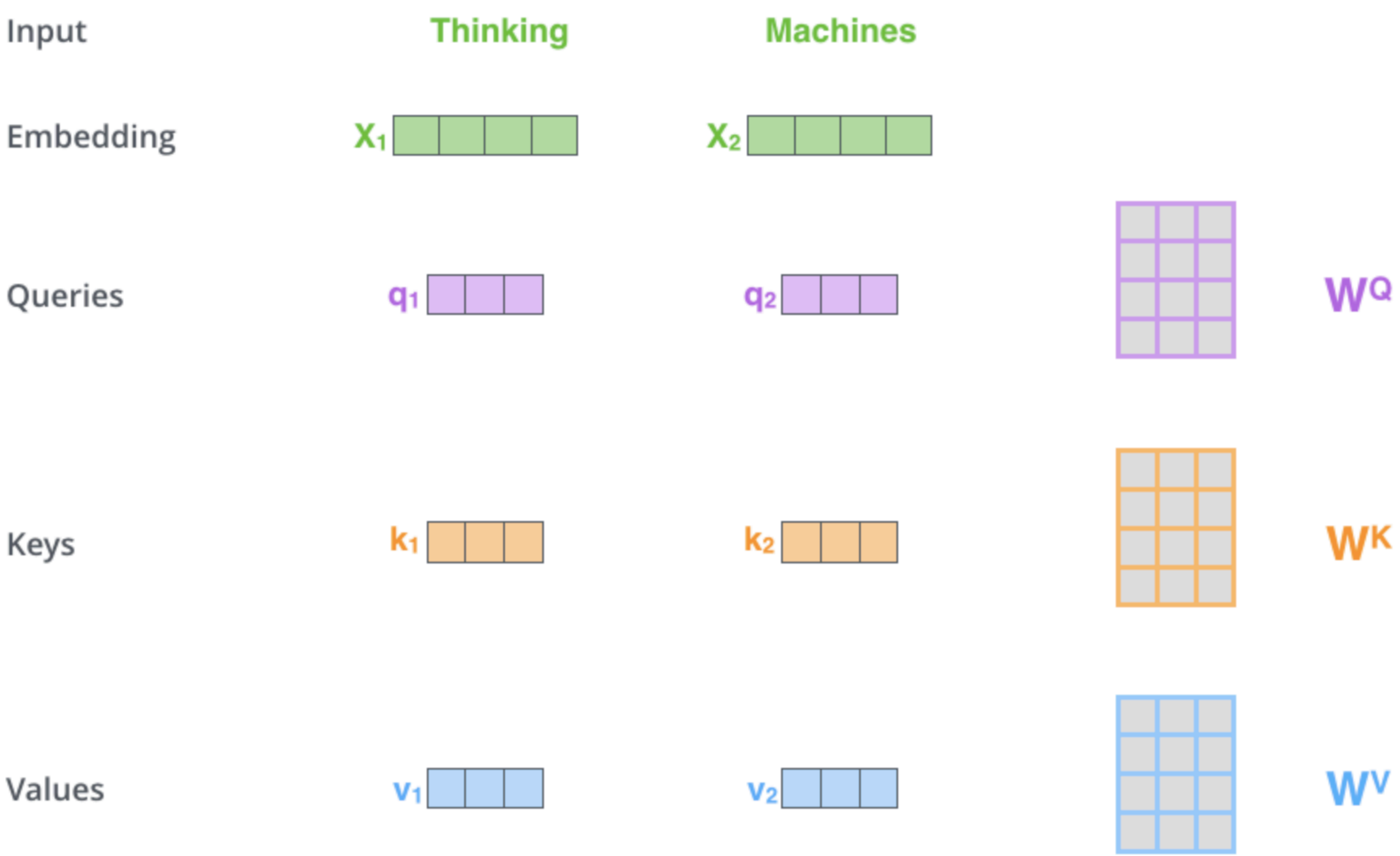
- Multiple "heads" analogous to different convolutional filters
- Let X = [sent len, embedding dim] be the input sentence
- Query $Q = W^Q X$: these are like the **decoder hidden state** in attention
- Keys $K = W^K X$: these control what gets attended to, along with the query
- Values $V = W^V X$: these 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

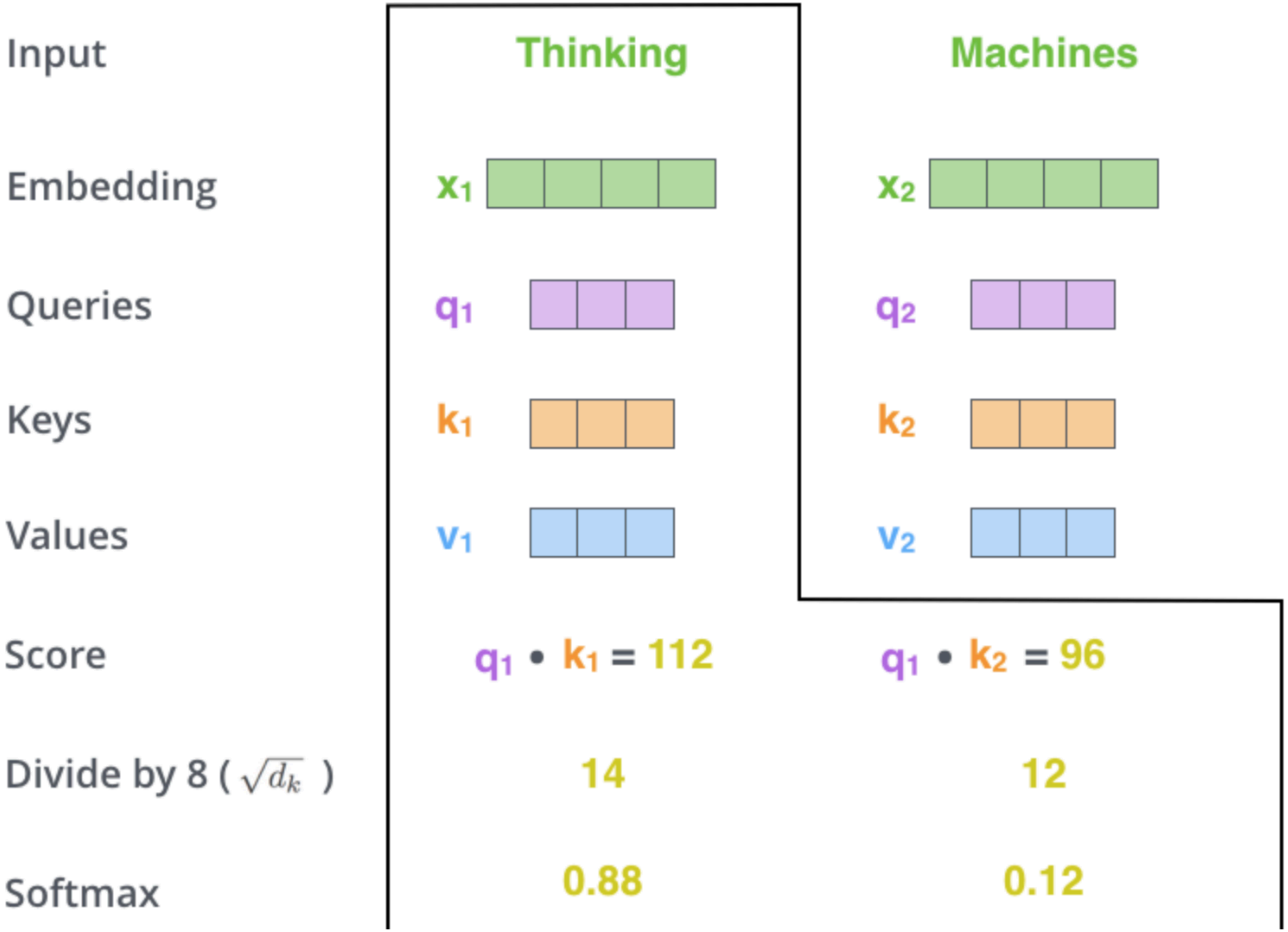
Vaswani et al. (2017)

Multi-Head Self Attention



Credit: Alammar, *The Illustrated Transformer*

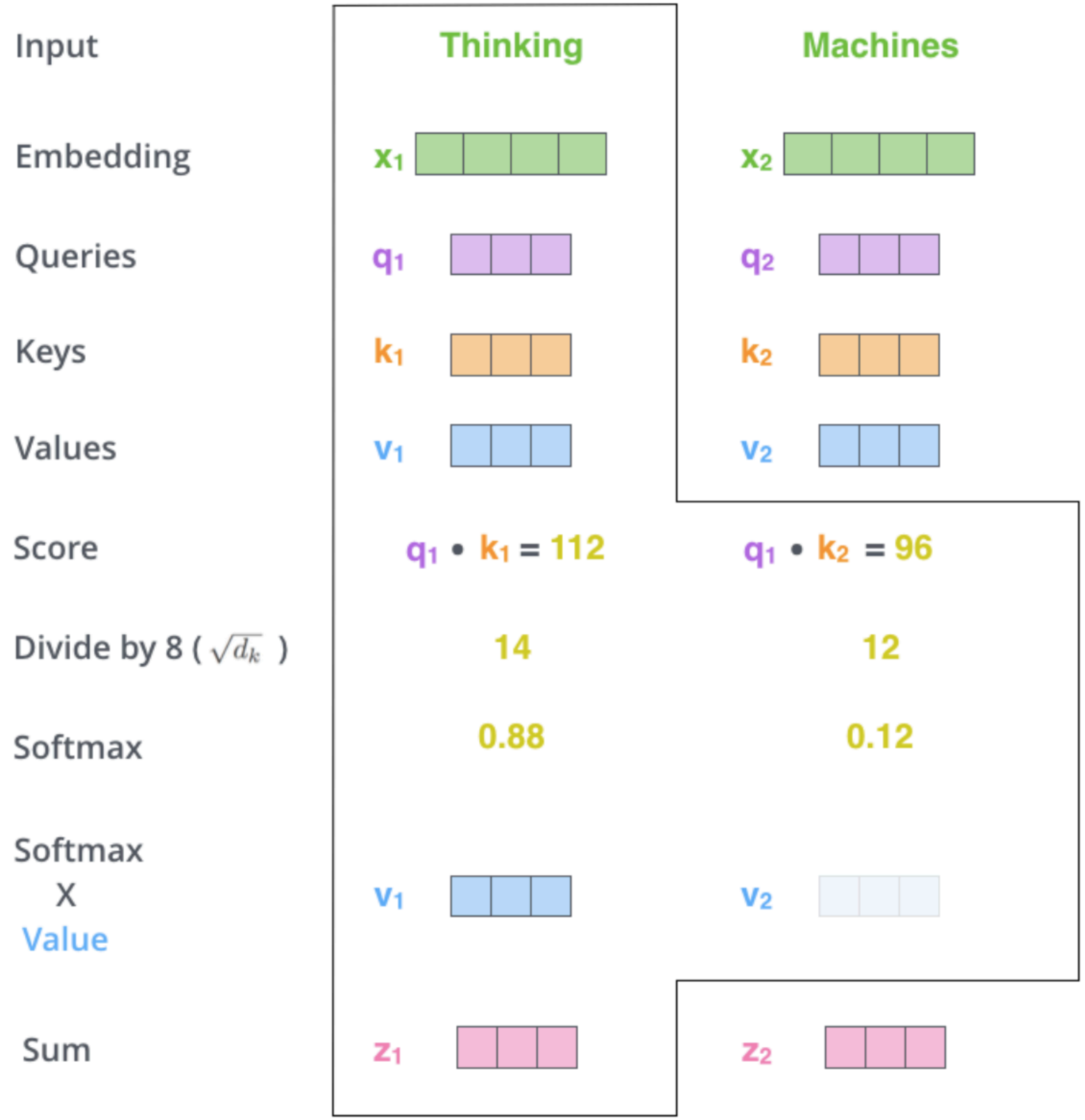
Multi-Head Self Attention



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

Credit: Alammar, *The Illustrated Transformer*

Multi-Head Self Attention

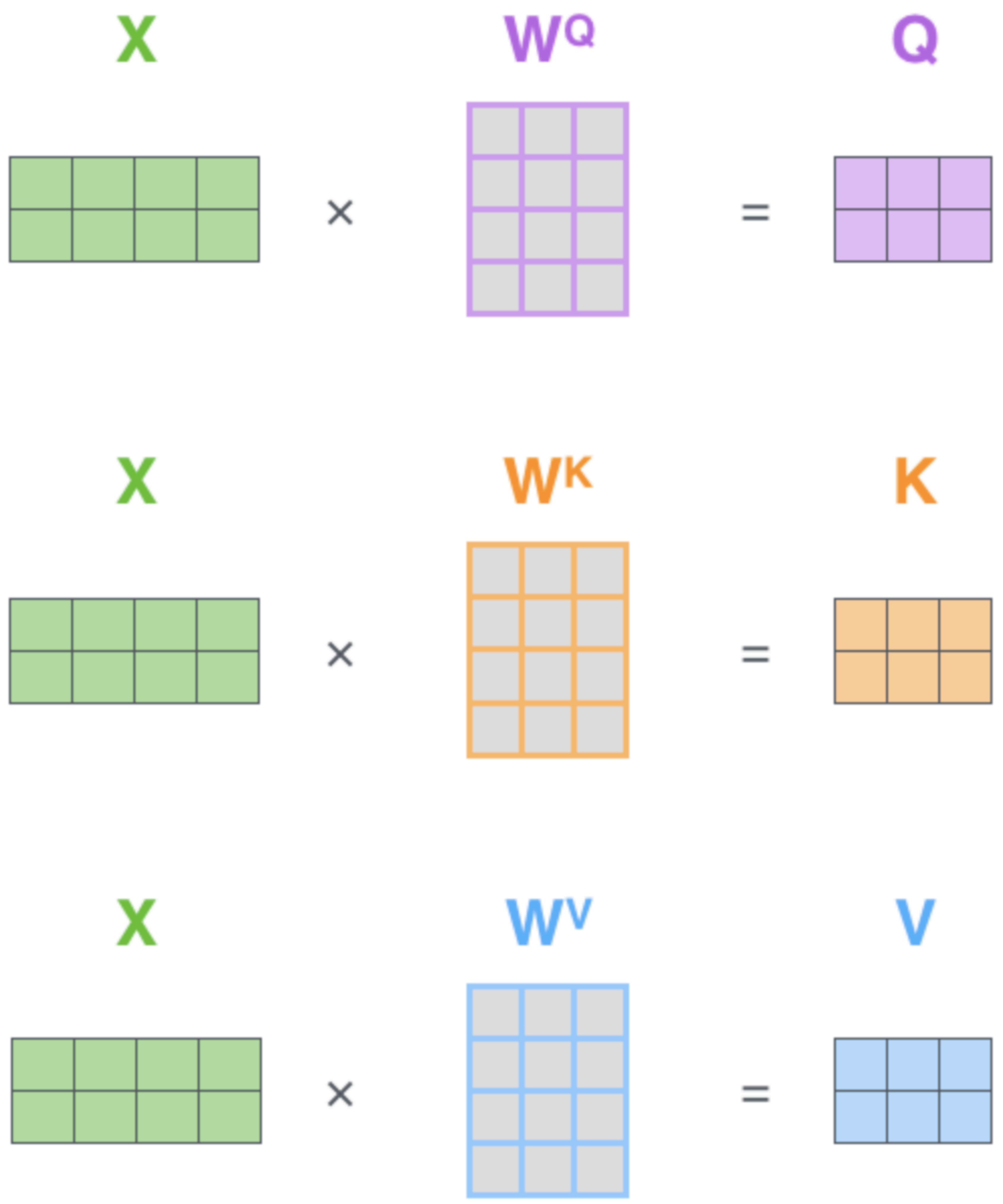


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

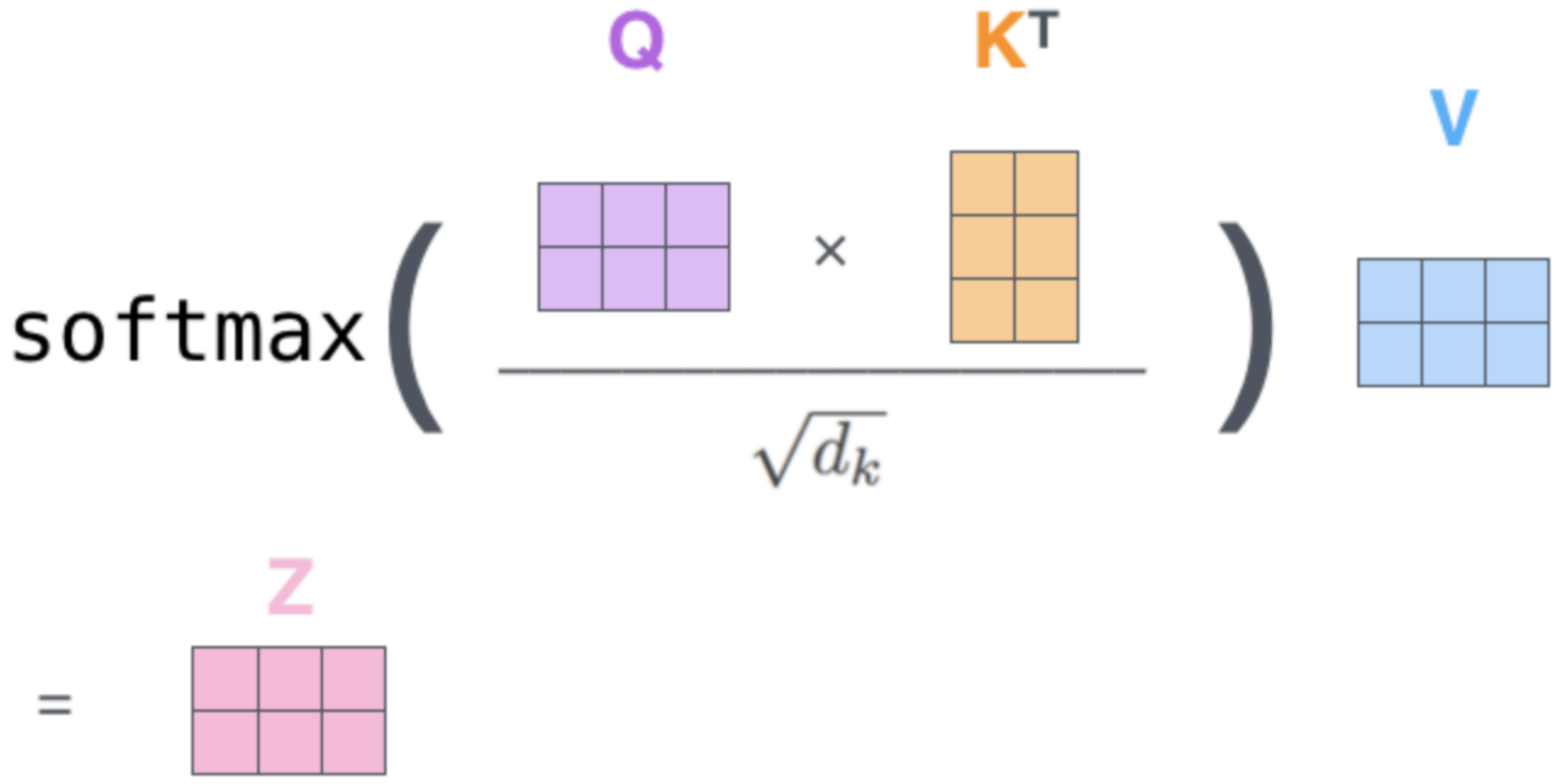
Credit: Alammar, *The Illustrated Transformer*

Multi-Head Self Attention

every row in X is a word in input sent



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

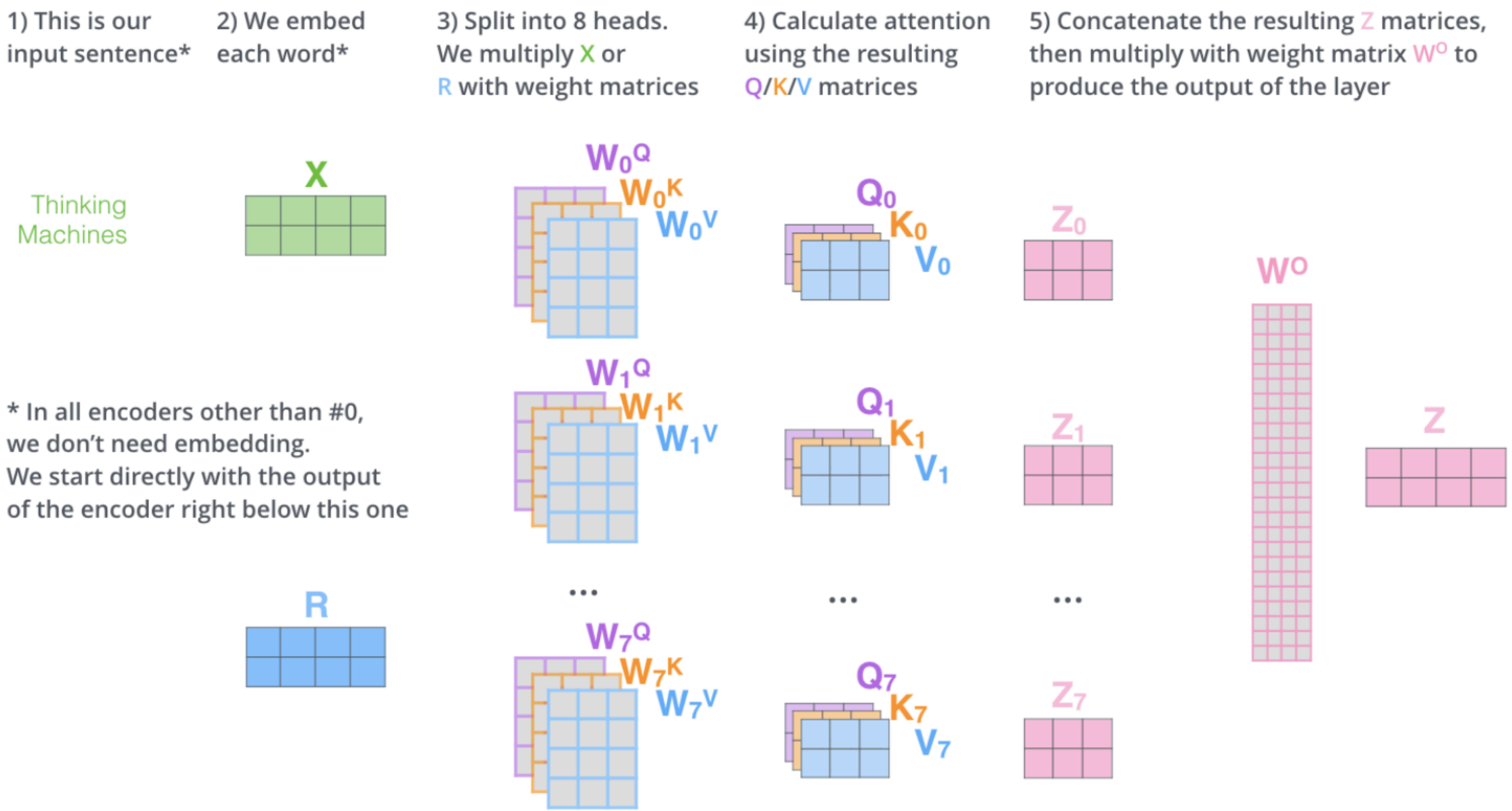


sent len x hidden dim

Z is a weighted combination of V rows

Credit: Alammar, *The Illustrated Transformer*

Multi-Head Self Attention



Credit: Alammar, *The Illustrated Transformer*

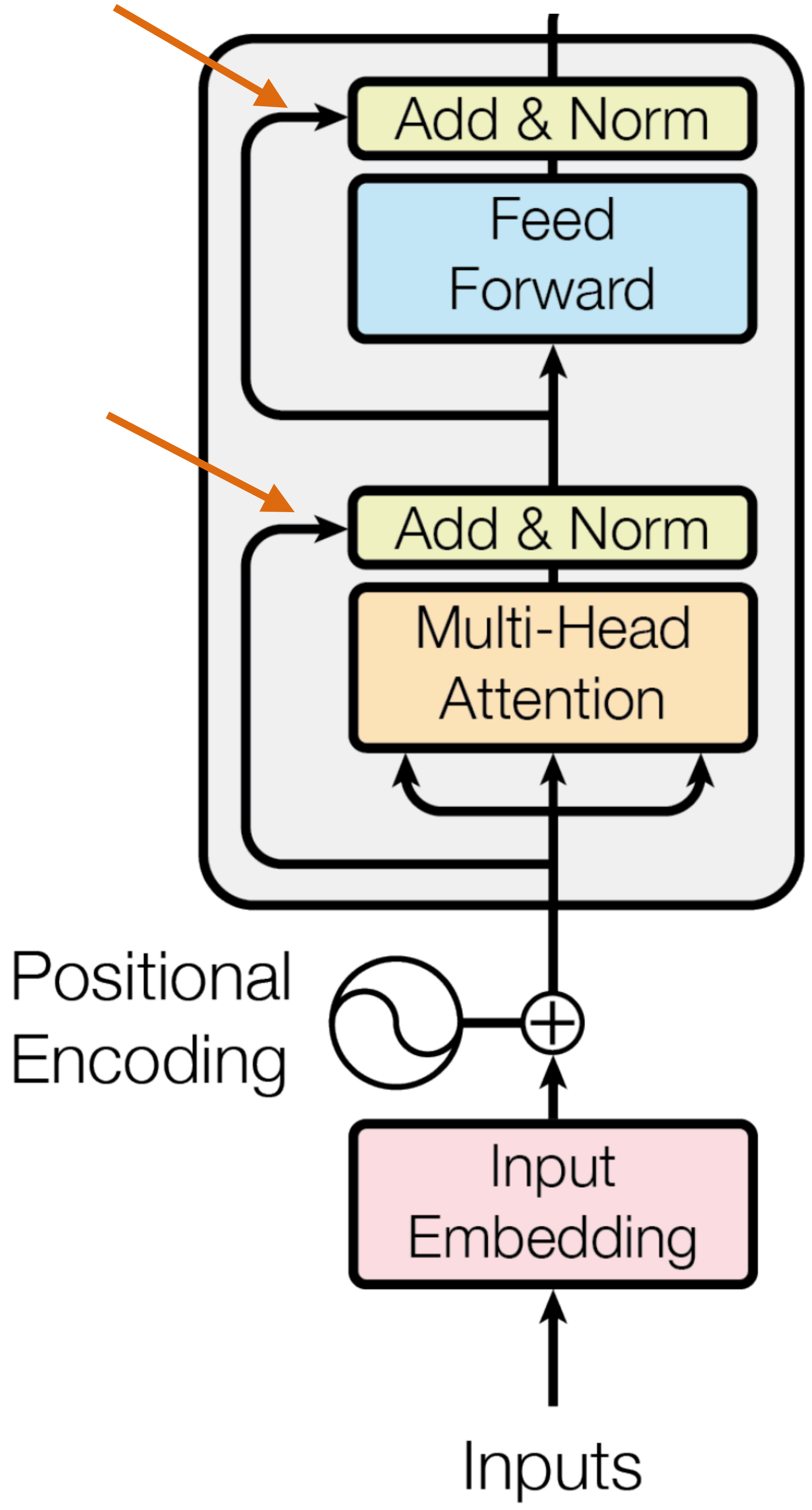
Properties of Self-Attention

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

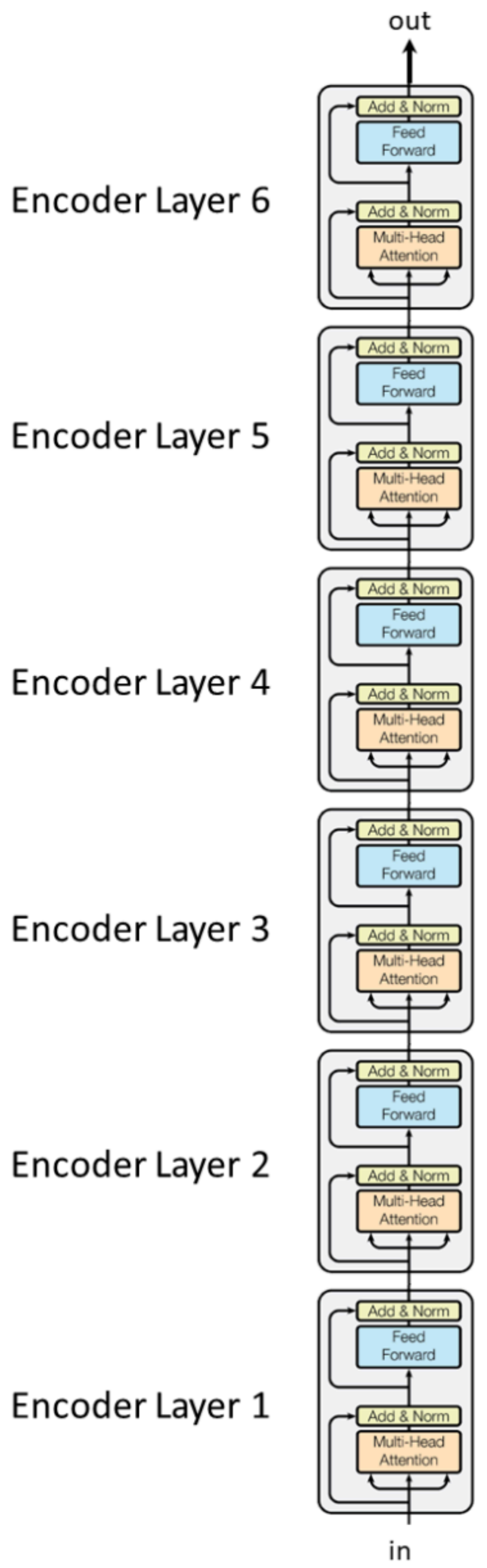
Vaswani et al. (2017)

Transformers



- Alternate multi-head self-attention layers and feedforward layers

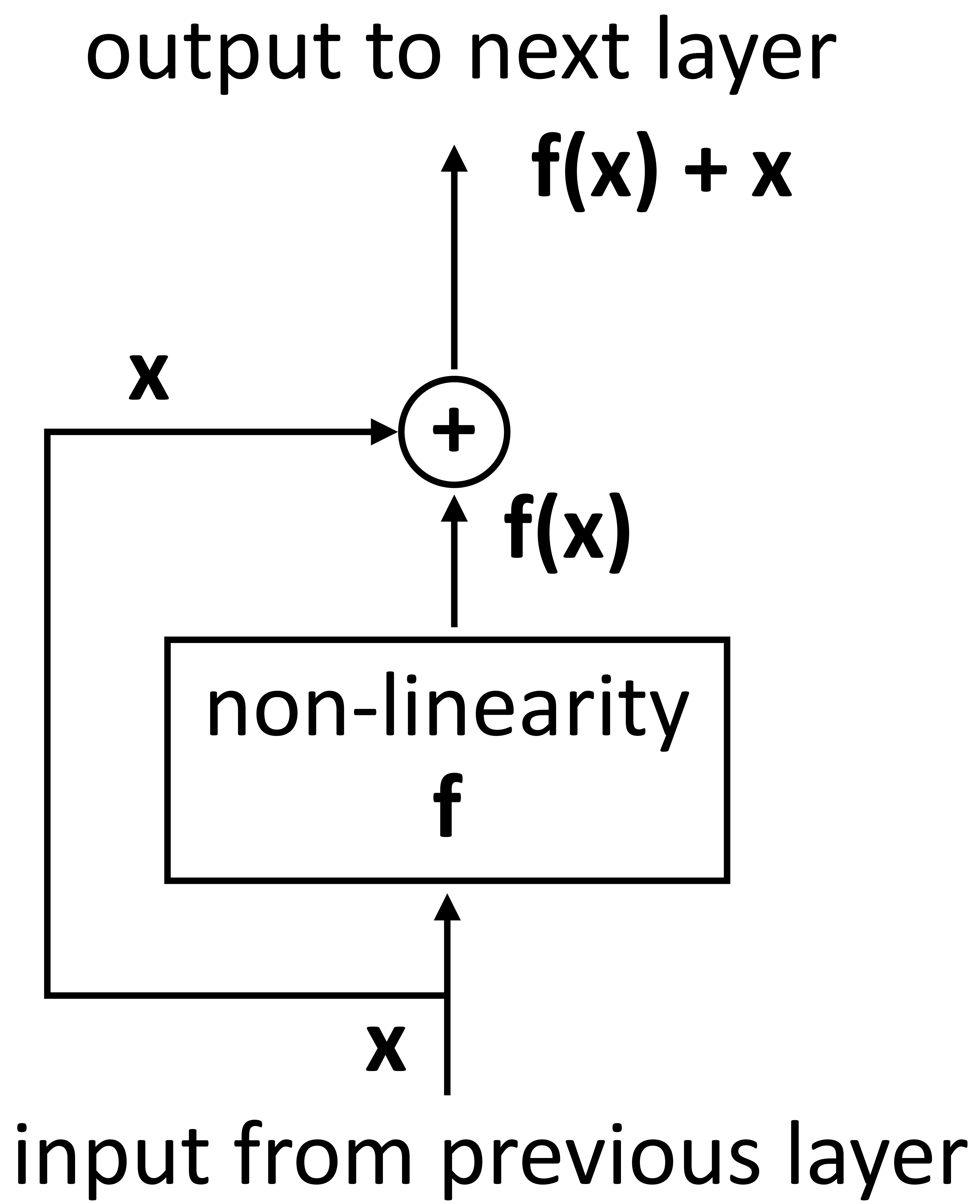
- Residual connections let the model "skip" each layer — these are particularly useful for training deep networks



Vaswani et al. (2017)

Residual Connections

- allow gradients to flow through a network directly, without passing through non-linear activation functions



He et al. (2015)

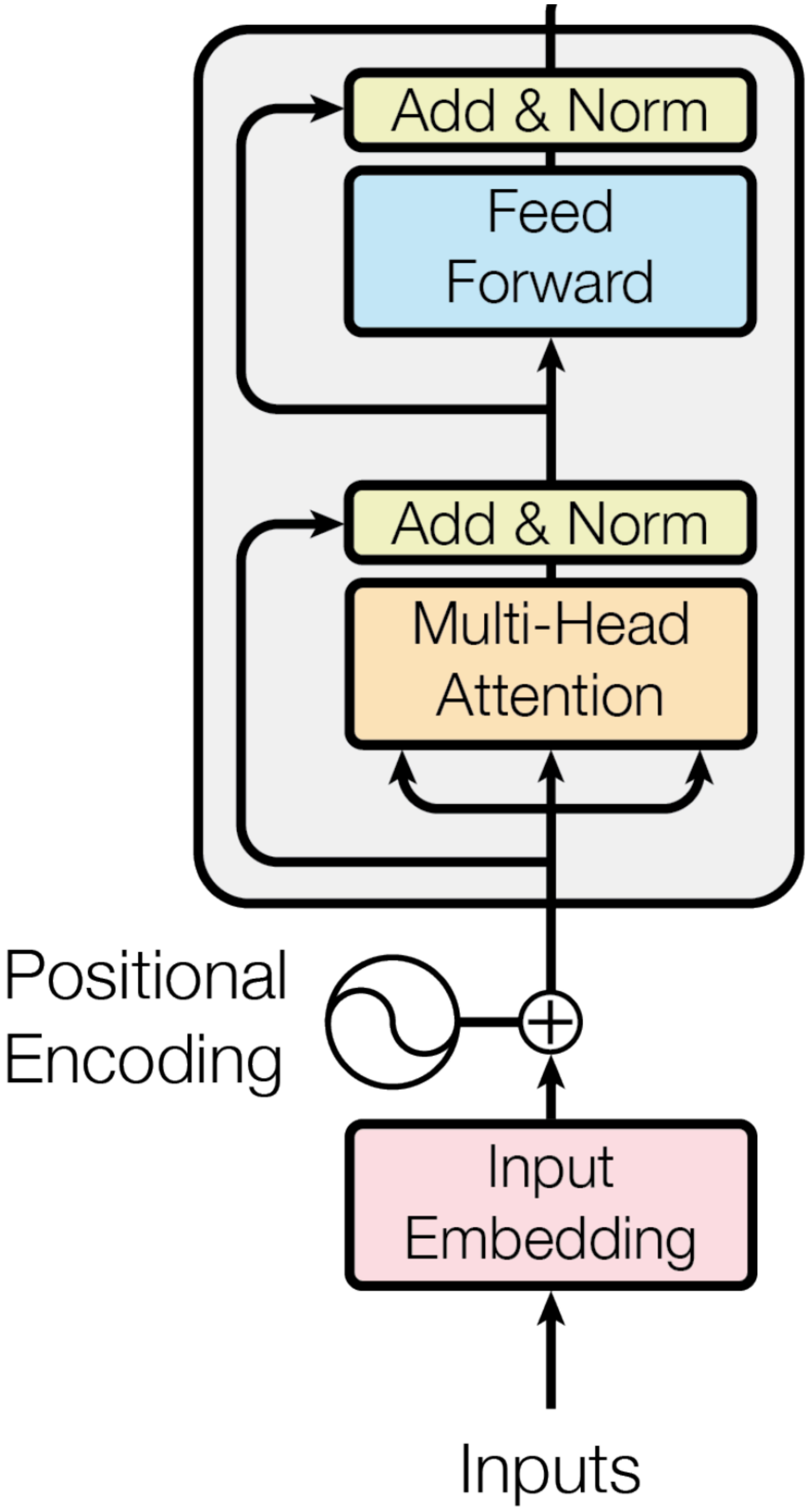
Transformers: Position Sensitivity

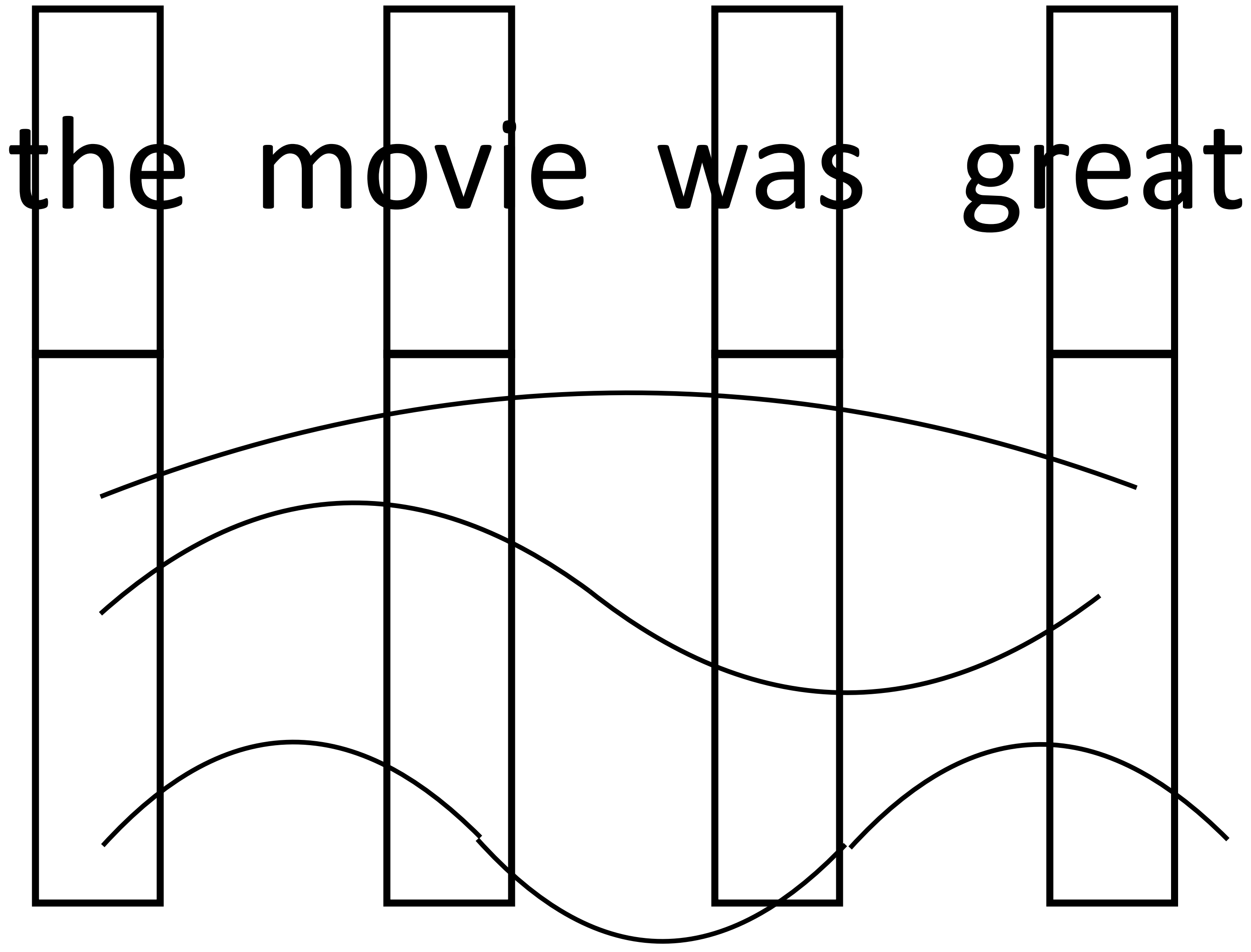
The ballerina is very excited that she will dance in the show.

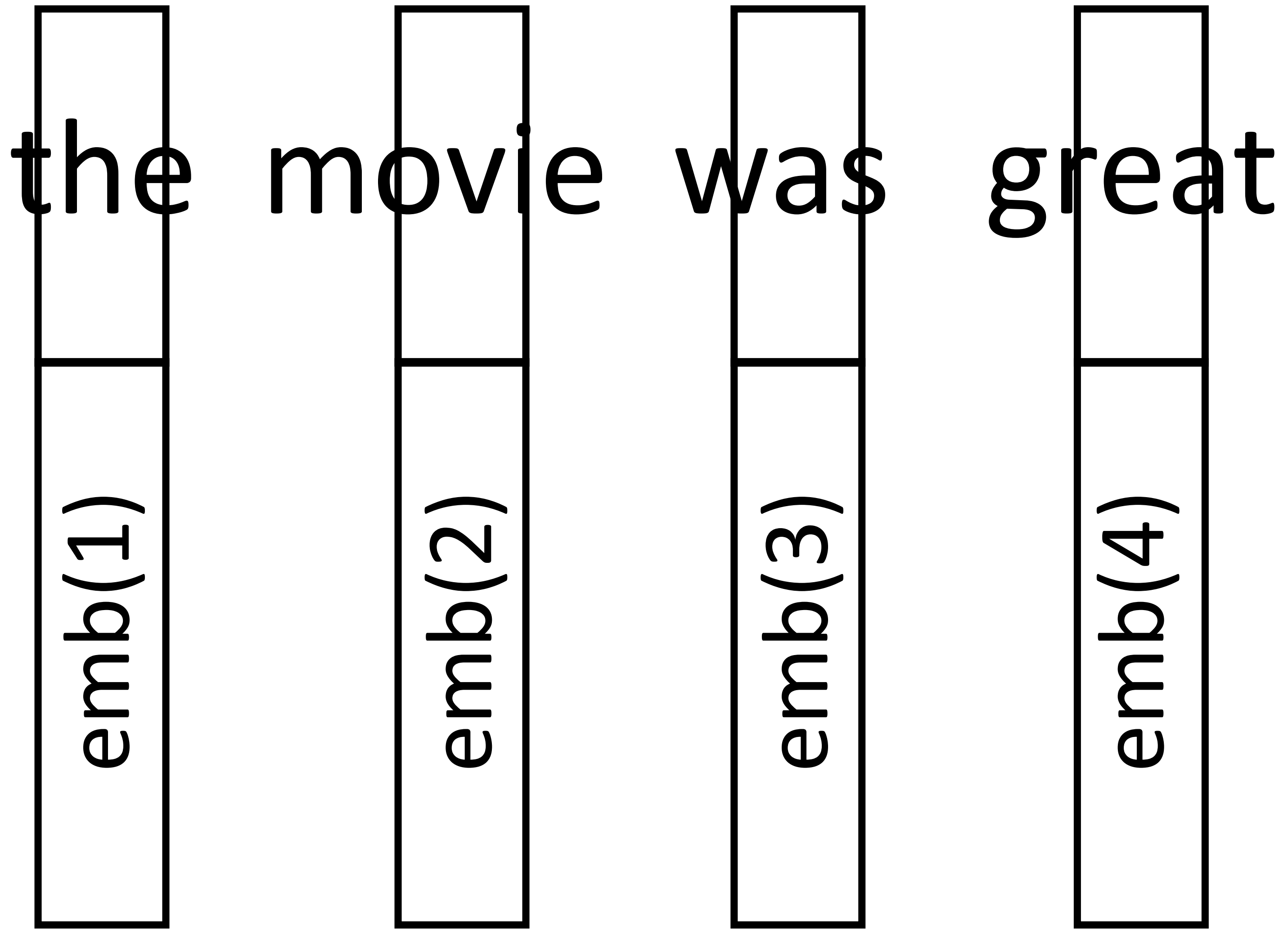
- If this is in a longer context, we want words to attend *locally*
- But transformers have *no notion of position* by default

Vaswani et al. (2017)

Transformers





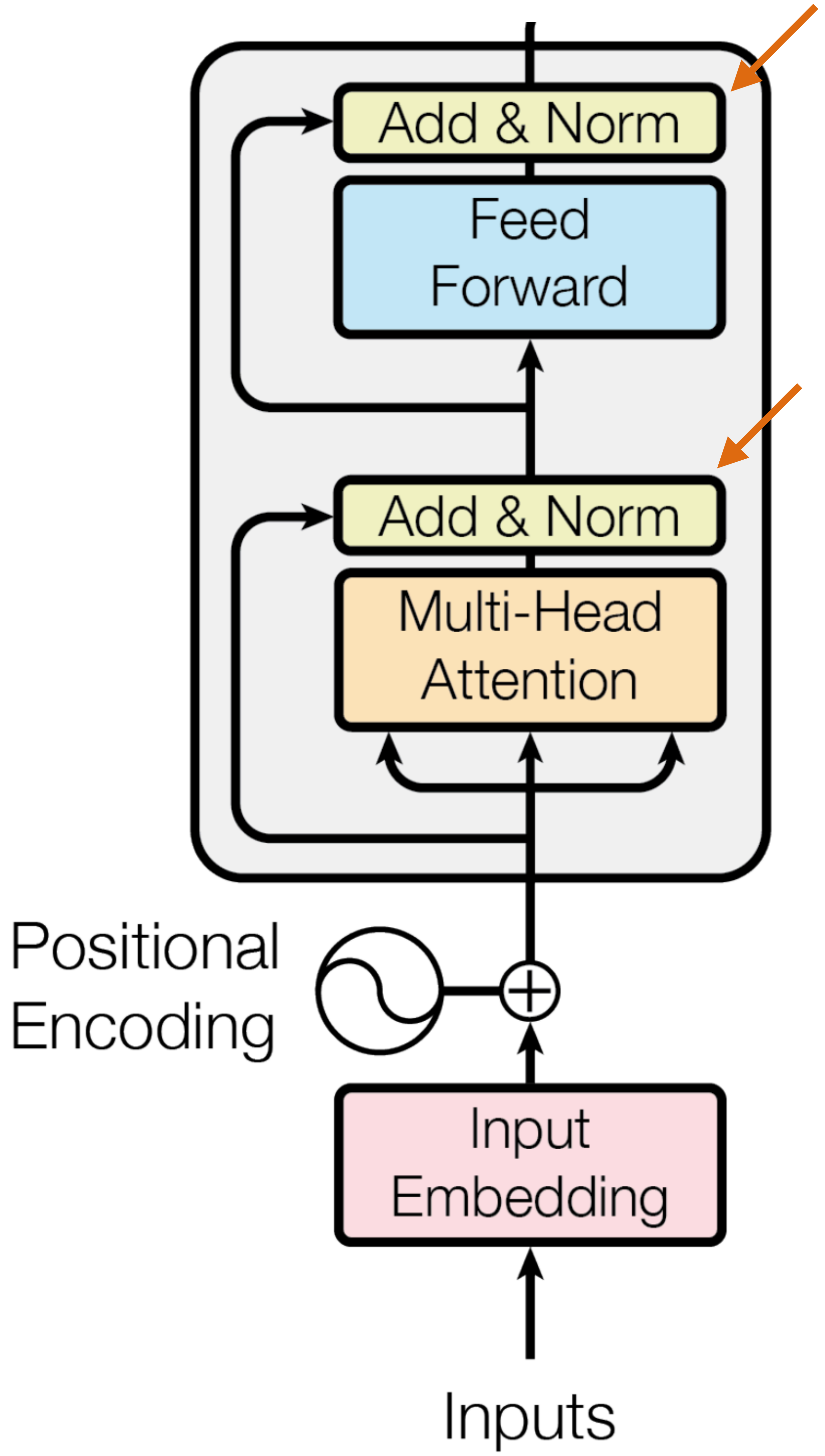


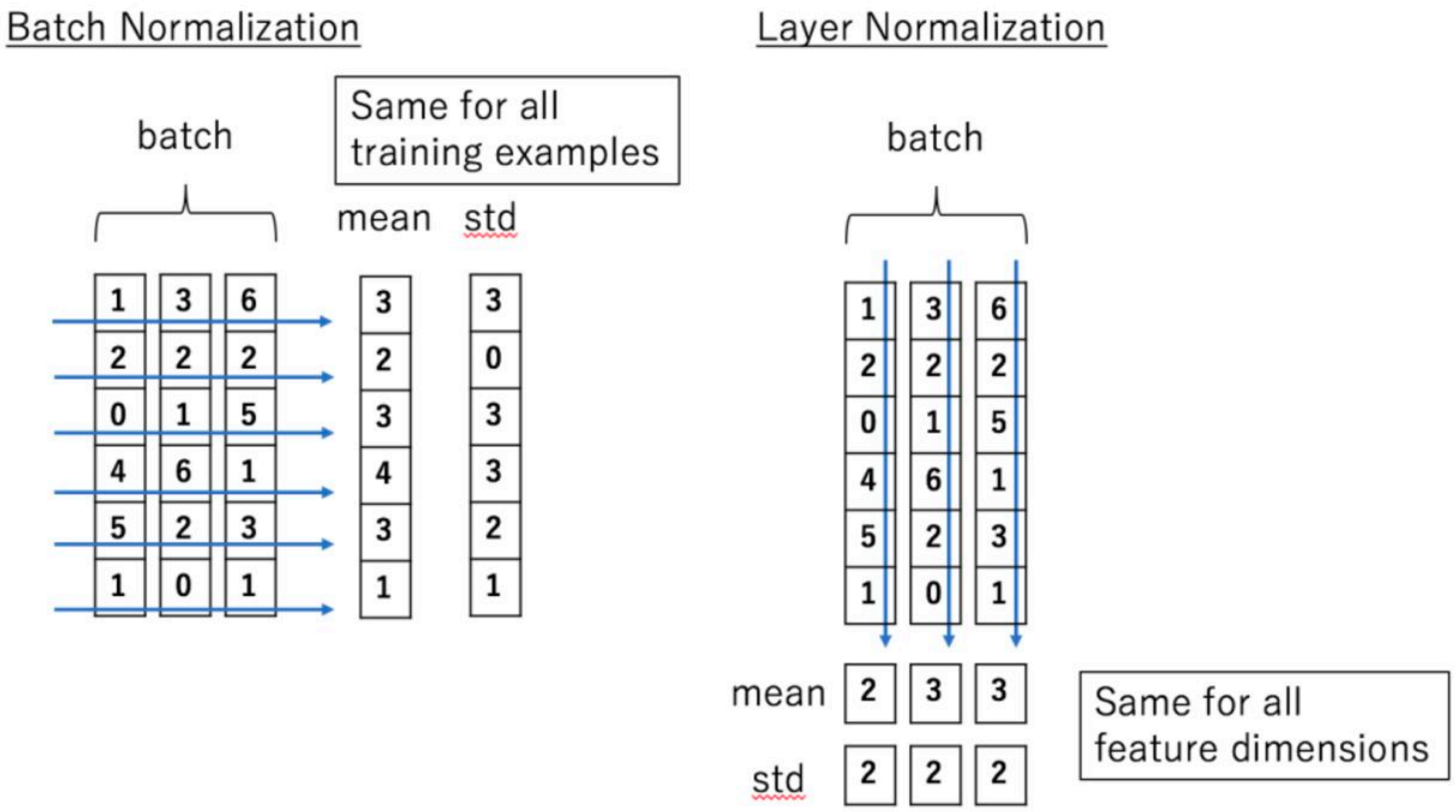
- Augment word embedding with position embeddings, each dim is a sine/cosine wave of a different frequency. Closer points = higher dot products
- Works essentially as well as just encoding position as a one-hot vector

Vaswani et al. (2017)

Layer Normalization

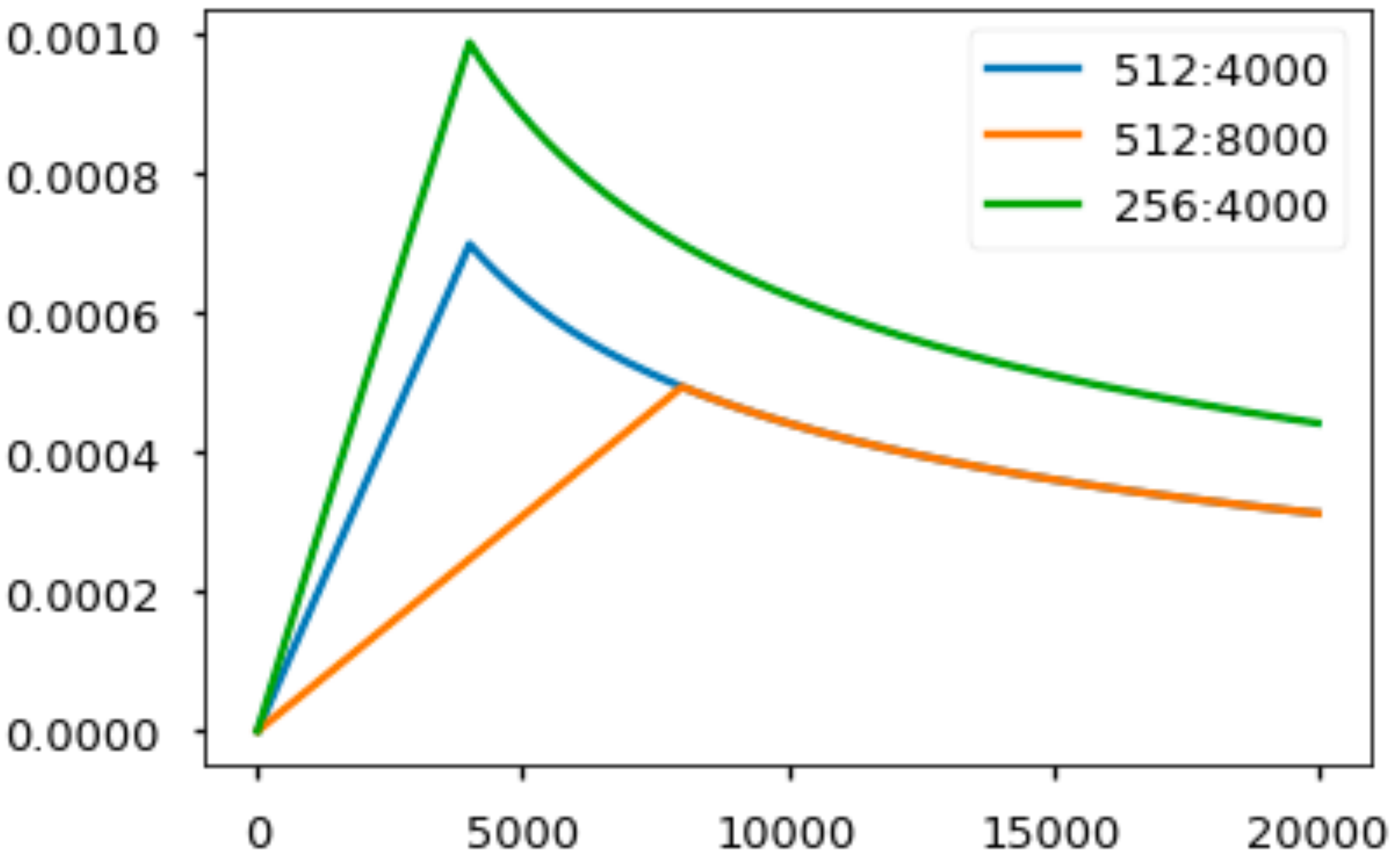
- subtract mean, divide by variance





Ba et al. (2016)

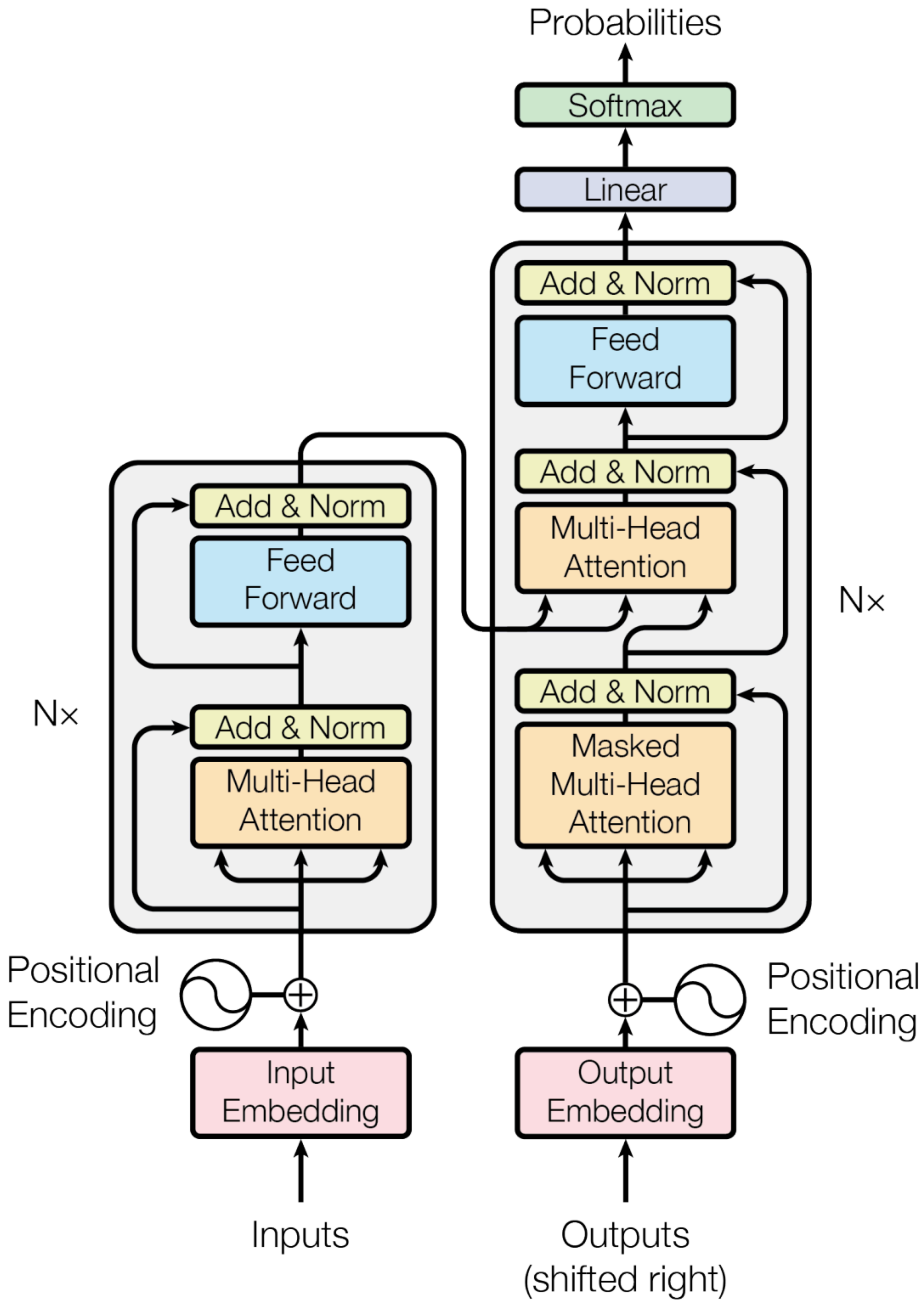
Transformers



- Adam optimizer with varied learning rate over the course of training
- Linearly increase for warmup, then decay proportionally to the inverse square root of the step number
- This part is very important!

Vaswani et al. (2017)

Transformers for MT: Complete Model



- Encoder and decoder are both transformers
- Decoder alternates attention over the output and attention over the input as well
- Decoder consumes the previous generated tokens but has *no recurrent state*

Vaswani et al. (2017)

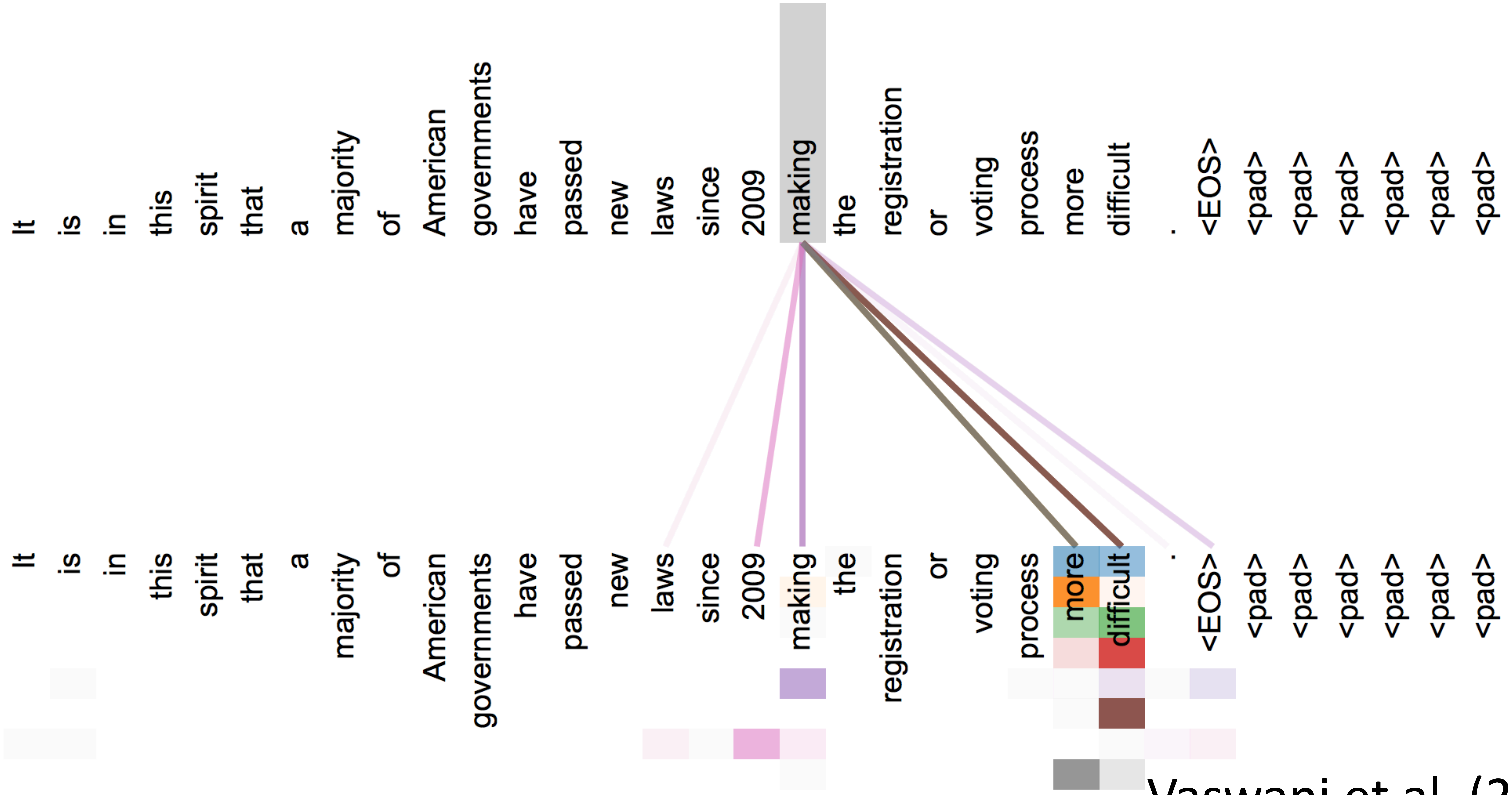
Transformers

Model	BLEU	
	EN-DE	EN-FR
ByteNet [18]	23.75	
Deep-Att + PosUnk [39]		39.2
GNMT + RL [38]	24.6	39.92
ConvS2S [9]	25.16	40.46
MoE [32]	26.03	40.56
Deep-Att + PosUnk Ensemble [39]		40.4
GNMT + RL Ensemble [38]	26.30	41.16
ConvS2S Ensemble [9]	26.36	**41.29**
Transformer (base model)	27.3	38.1
Transformer (big)	**28.4**	**41.8**

- Big = 6 layers, 1000 dim for each token, 16 heads, base = 6 layers + other params halved

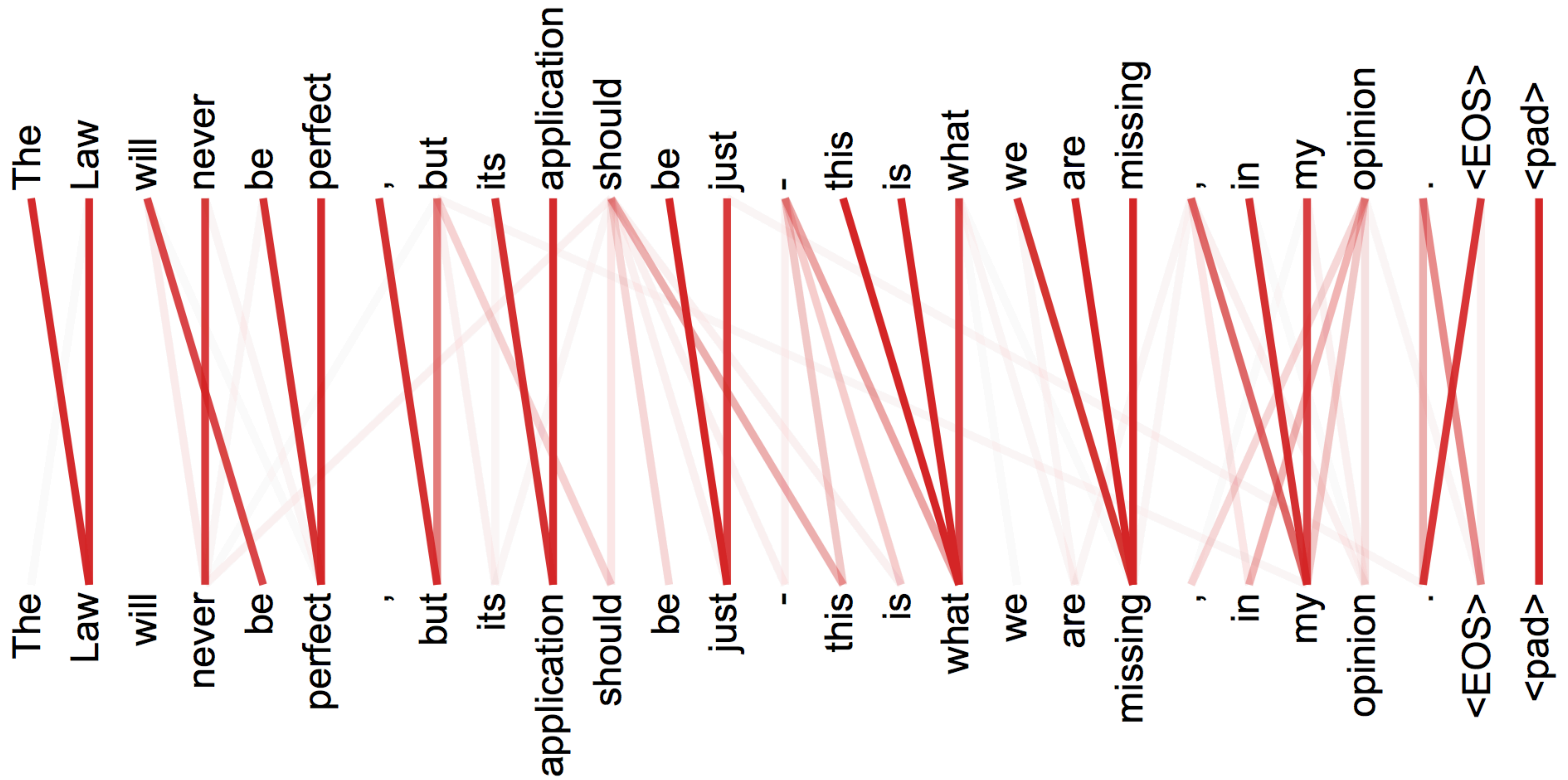
Vaswani et al. (2017)

Visualization



Vaswani et al. (2017)

Visualization



Vaswani et al. (2017)

Useful Resources

nn.Transformer:

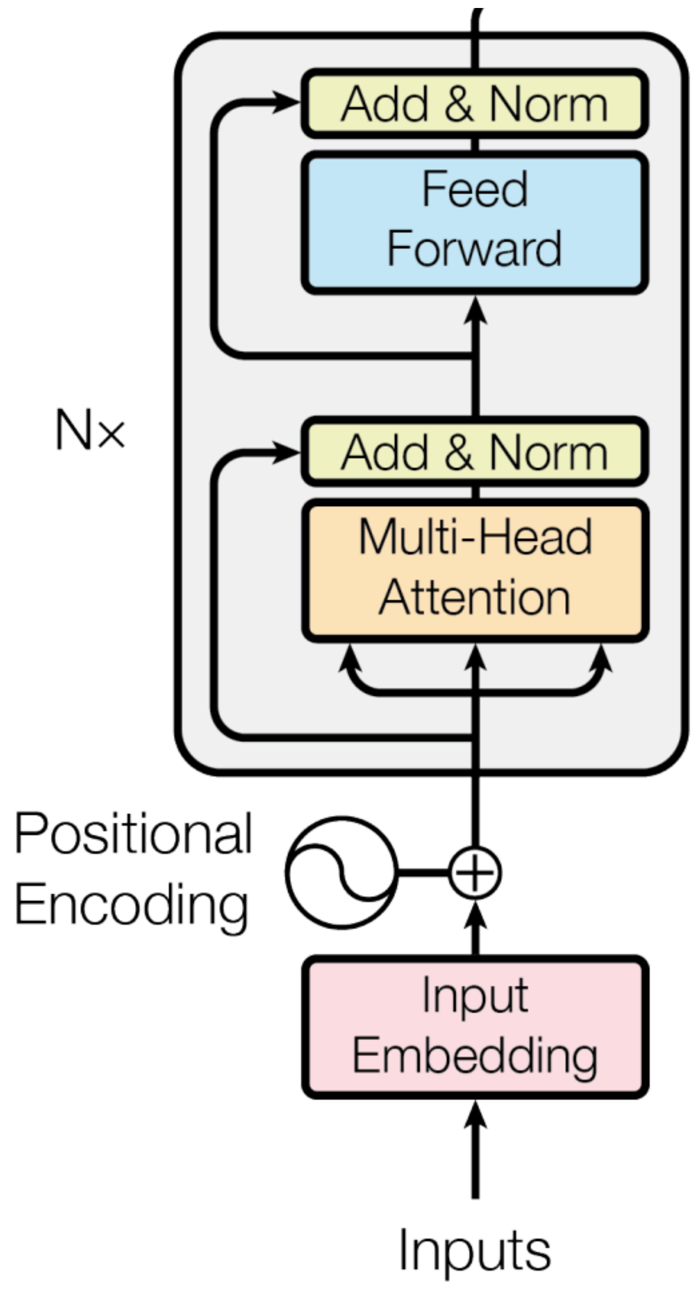
```
>>> transformer_model = nn.Transformer(nhead=16, num_encoder_layers=12)
>>> src = torch.rand((10, 32, 512))
>>> tgt = torch.rand((20, 32, 512))
>>> out = transformer_model(src, tgt)
```

nn.TransformerEncoder:

```
>>> encoder_layer = nn.TransformerEncoderLayer(d_model=512, nhead=8)
>>> transformer_encoder = nn.TransformerEncoder(encoder_layer, num_layers=6)
>>> src = torch.rand(10, 32, 512)
>>> out = transformer_encoder(src)
```

Other Transformer Variations

- Multilayer transformer networks consist of interleaved self-attention and feedforward sublayers.
- Could ordering the sublayers in a different pattern lead to better performance?



(a) Interleaved Transformer

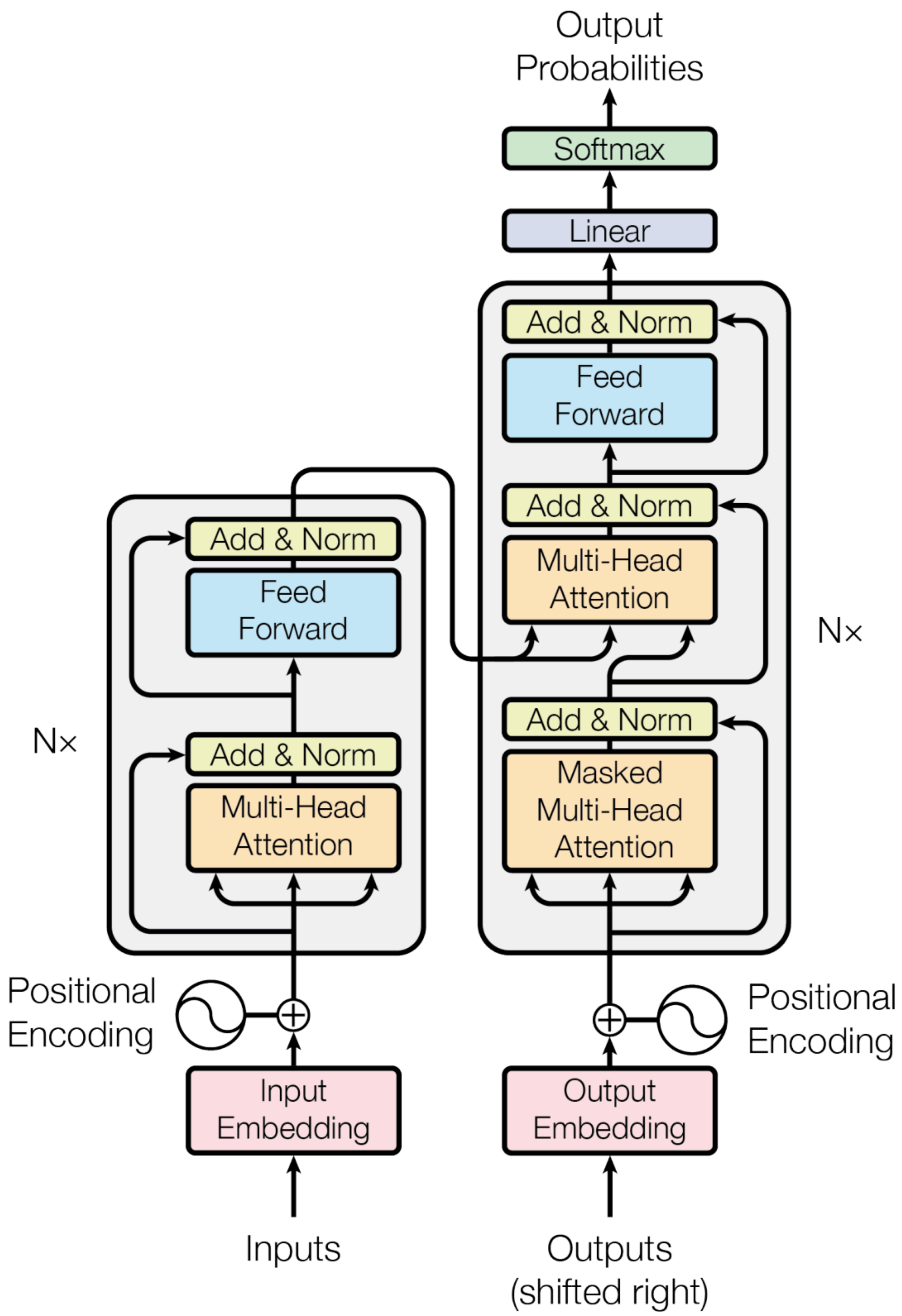
(b) Sandwich Transformer

Figure 1: A transformer model (a) is composed of interleaved self-attention (green) and feedforward (purple) sublayers. Our sandwich transformer (b), a reordering of the transformer sublayers, performs better on language modeling. Input flows from left to right.

Press et al. (2020)

Summary: Transformer Uses

- Supervised: transformer can replace LSTM as encoder, decoder, or both; such as in machine translation and natural language generation tasks.



- Encoder and decoder are both transformers
- Decoder consumes the previous generated token (and attends to input), but has *no recurrent state*
- Many other details to get it to work: residual connections, layer normalization, positional encoding, optimizer with learning rate schedule, label smoothing ….

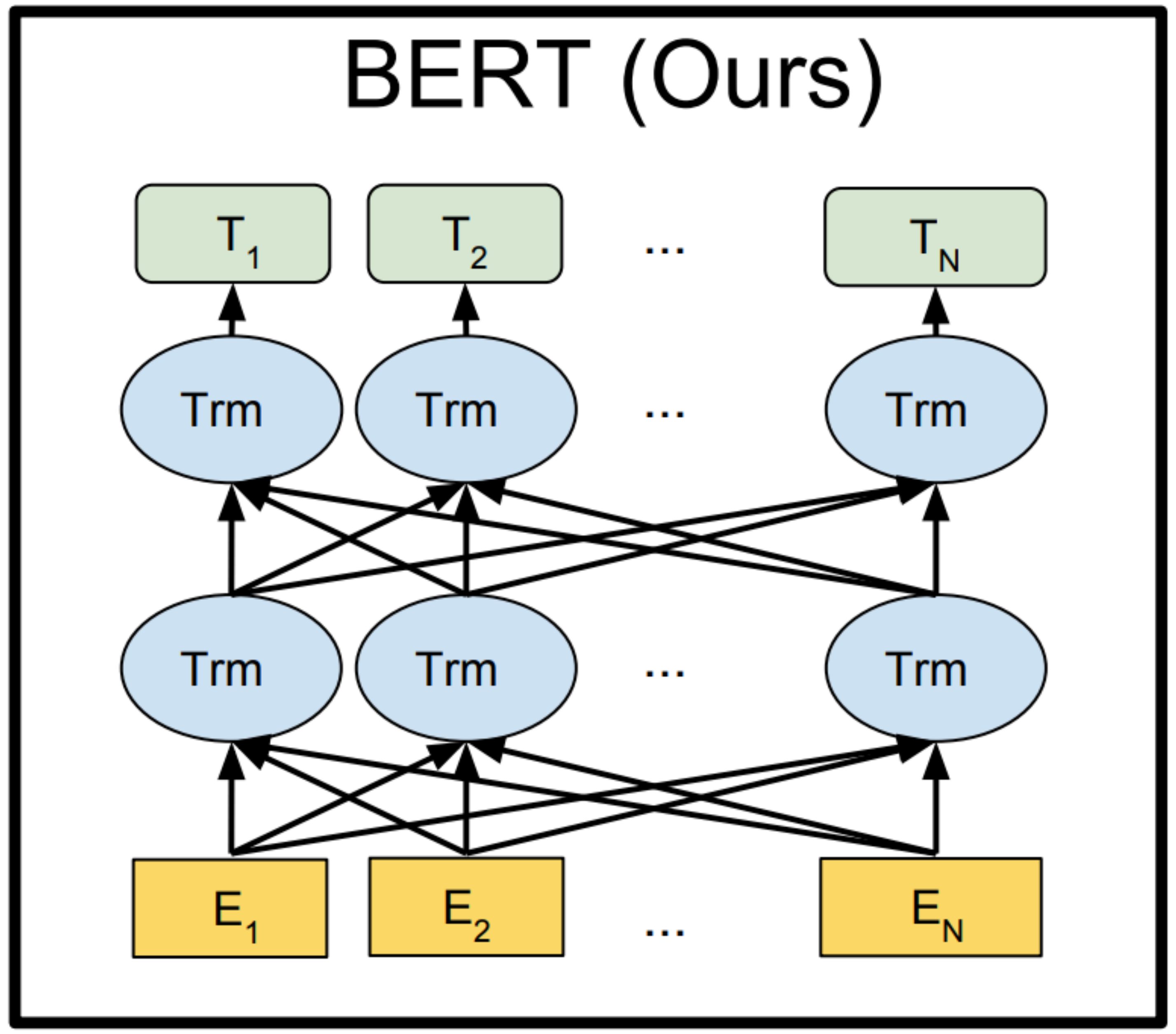
Vaswani et al. (2017)

Summary: Transformer Uses

- Unsupervised: transformers work better than LSTM for unsupervised pre-training of embeddings — predict word given context words

- BERT (Bidirectional Encoder Representations from Transformers): pretraining transformer language models similar to ELMo (based on LSTM)

- Stronger than similar methods, SOTA on ~11 tasks (including NER — 92.8 F1)



Course Project

Final Project

- **Groups Size:** 2-4 people; 1 is possible (email me for permission).
- **Submission:** 4-page report (required) + final project presentation (optional).
- **Prize:** We will give out 1-3 best project awards. 🏆

- **Shared project** with other classes is allowed
  - project is expected to be accordingly bigger/better
  - clearly declare at the beginning of your report that you are sharing project (with which class)
- **External collaborators** (non CS7650 students) are also allowed
  - clearly describe in the report which parts of the projects are your work

Your Two Choices

- Choice 1 — Custom Project
  - Topic of your own choice
    - choose something you are interested in
    - choose an easy or choose a challenging topic whichever suits you

- Choice 2 — Provided Research Ideas
  - #1: Studying PTSD using social media data
  - #2: Studying perceptions of disability on social media
  - #3: Analyzing text simplification corpus

Finding Research Topics

- Two basic starting points, for all of science:
  - **Nails** — start with a (domain) problem of interest and try to find good/better ways to address it than are currently known/used
  - **Hammers** — start with a technical method/approach of interest, and work out good ways to extend or improve it or new ways to apply it

Slide credit: Chris Manning

Typical Project Types

- This is not an exhaustive list —
- 1) Find an application/task of interest and explore how to approach/solve it effectively, often with an existing model
  - Could be task in the wild or some existing Kaggle competition or shared task (e.g.. WNUT or SemEval, etc.)
  - Or dialogue system (prepare for Amazon Alexa Challenges next year)
- 2) Analyze the behavior of models or existing datasets
  - how the model represents linguistic knowledge or what kinds of phenomena it can handle or errors that it makes.
  - what linguistic phenomena/errors exist in the dataset, how they affect model performance (see Idea #3 for an example).

Typical Project Types

- This is not an exhaustive list —
- 3) Create a new dataset, conduct some analysis, train a prediction model
  - for a new topic/task (see Idea #1 and #2 for an example), or for an existing task but better way to create higher quality dataset
  - may involve some manual annotation
  - conduct some quantitive and linguistic analyses
- 4) Implement a complex neural architecture and demonstrate its performance on some data, especially for non-English data
- 5) Come up with a new or variant neural network model and explore its empirical success (but this has become harder since 2020 —)

Place to start?

- Look at ACL Anthology for NLP papers:
  - https://aclanthology.org/
- Also look at the online proceedings of major ML/Web conferences
  - ICLR, NeurIPS , ICML
  - ICWSM (https://www.icwsm.org/2021/)
- Look at online preprint servers, especially:
  - https://arxiv.org/
- Look for an interesting problem in the world!
  - Psycholinguistics (e.g., Idea #1), computational social science, journalism, …

Finding Data

- Some people collect their own data for a project — **we like that!**
  - You may have a project that uses “unsupervised” data
  - You can annotate a small amount of data
  - You can find a website that effectively provides annotations, such as likes, starts, rating, responses, etc.
  - Look at research papers to see what data they use, how they get it
- Many others make use of existing datasets built by other researchers
  - Shared task at WNUT, SemEval, etc.
  - Kaggle competition
  - Datasets used in other papers (e.g. https://aclanthology.org/)

An Example

- Define Task

- Define Dataset
  - Provide basic data statistics
  - If your own data —
    - steps you take to collect/clean/annotate the data
    - provide some examples, quality control (this is important!)

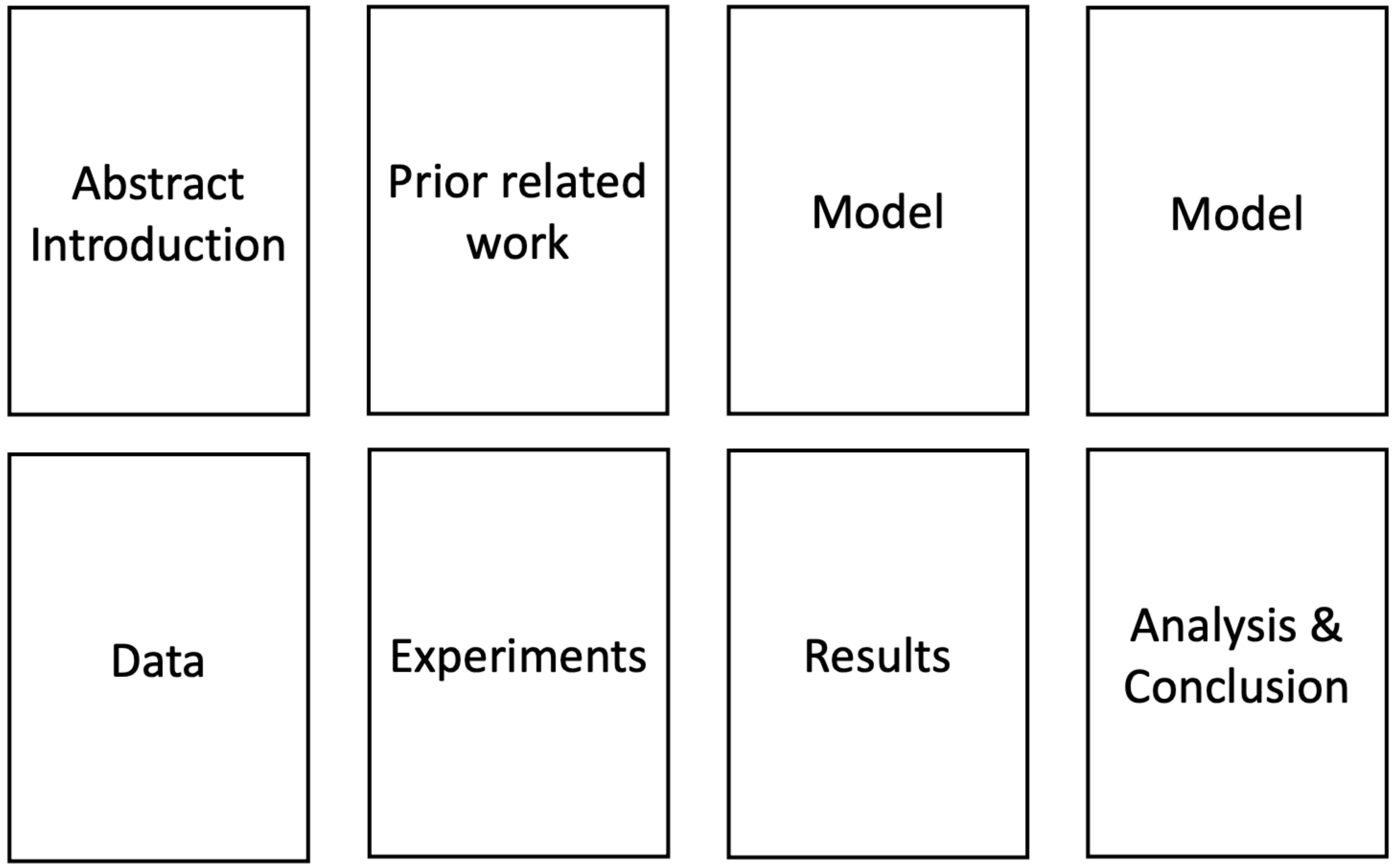
An Example

- Experiments
  - right from the beginning, separate off train/dev/test splits
  - search online for well-established metrics on this task
  - establish some baselines
  - Implement existing neural network model
    - compute metrics on train & dev, not test set
    - analyze outputs and errors

- Going beyond — try out different models, increasing quality/quantitive of your dataset, data argumentation, and other “researchy” ideas!

Final Project Writeup/Presentation

- **4-page writeup** due the day before final exam date (no late submission!)
- Use **LaTeX template** from ACL
- Include references; statement of each group members' contribution
- Writeup quality is important to your grade!
- X-minute **oral presentation (optional) at the final exam time** (X $\in$ [5, 10])



Credit: Stanford CS224n

Some example research ideas …

- Studying PTSD using social media data
- Studying perceptions of disability on social media
- Analyzing text simplification corpus

Details of above projects are in the written instructions posted on Piazza.

- and many more …

Have fun with your project!