Copy/Pointer + Transformer

Wei Xu

(many slides from Greg Durrett)

Administrivia

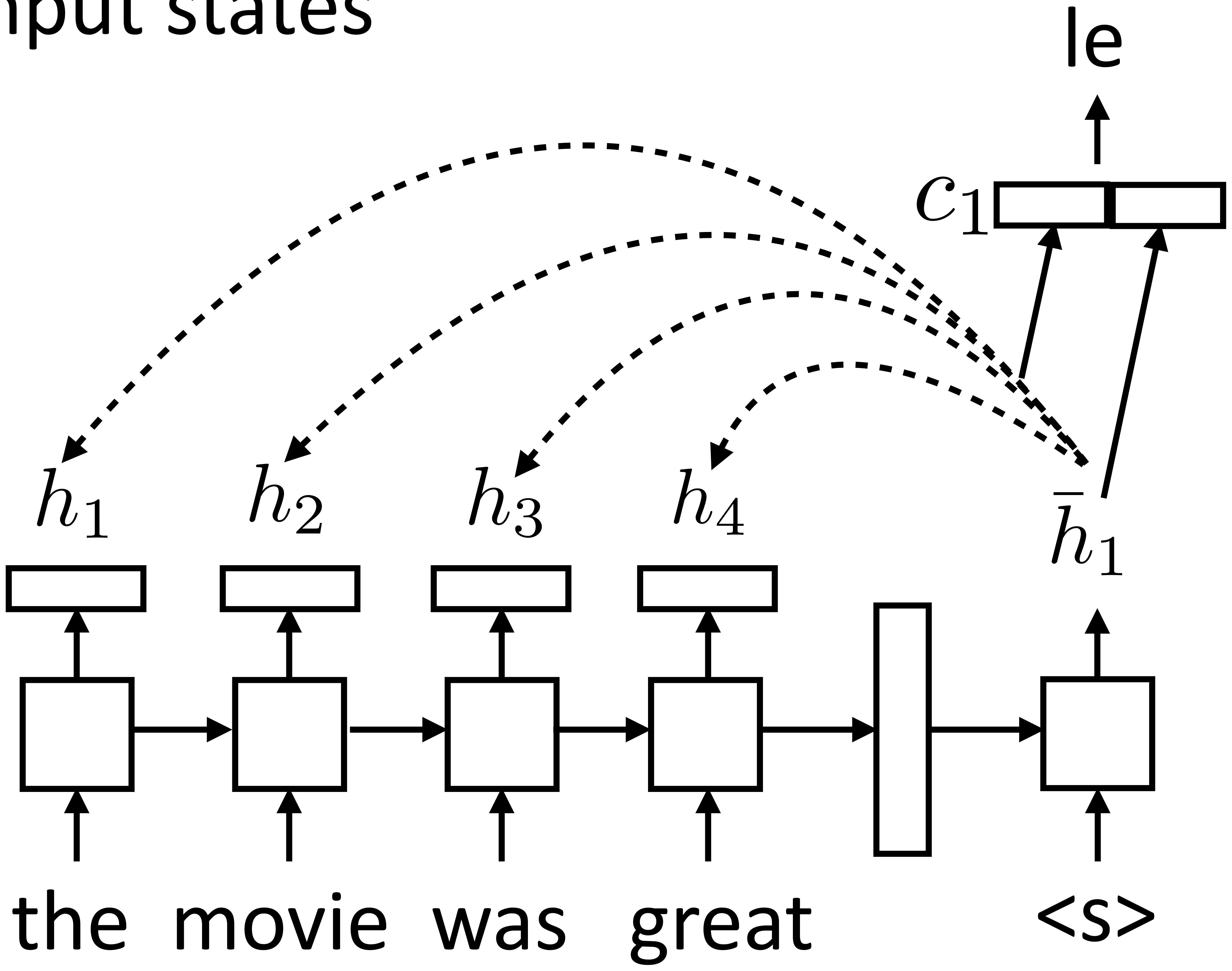
- Midterm is released (due 3/18)
- Final course project — plan to discuss more next class.

This and Next Lecture

- Copy mechanisms /Pointer networks for copying words to the output
- Transformer architecture
- Frontiers in MT Research
- Applications of Seq2Seq (beyond MT)
- Decoding in seq2seq models

Recap: Attention

- For each decoder state, compute weighted sum of input states



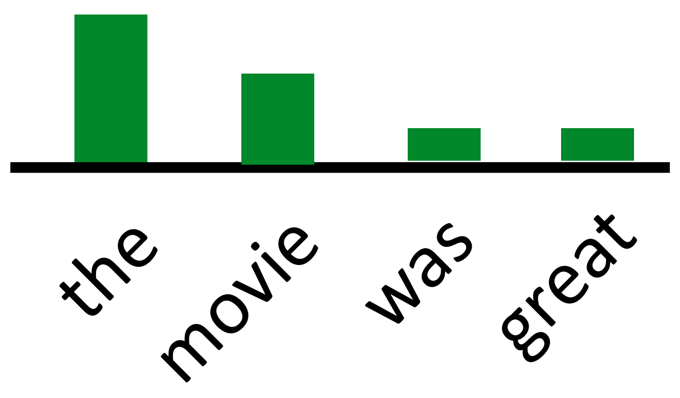
- No attn: $P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h}_i)$

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W[c_i; \bar{h}_i])$$

$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

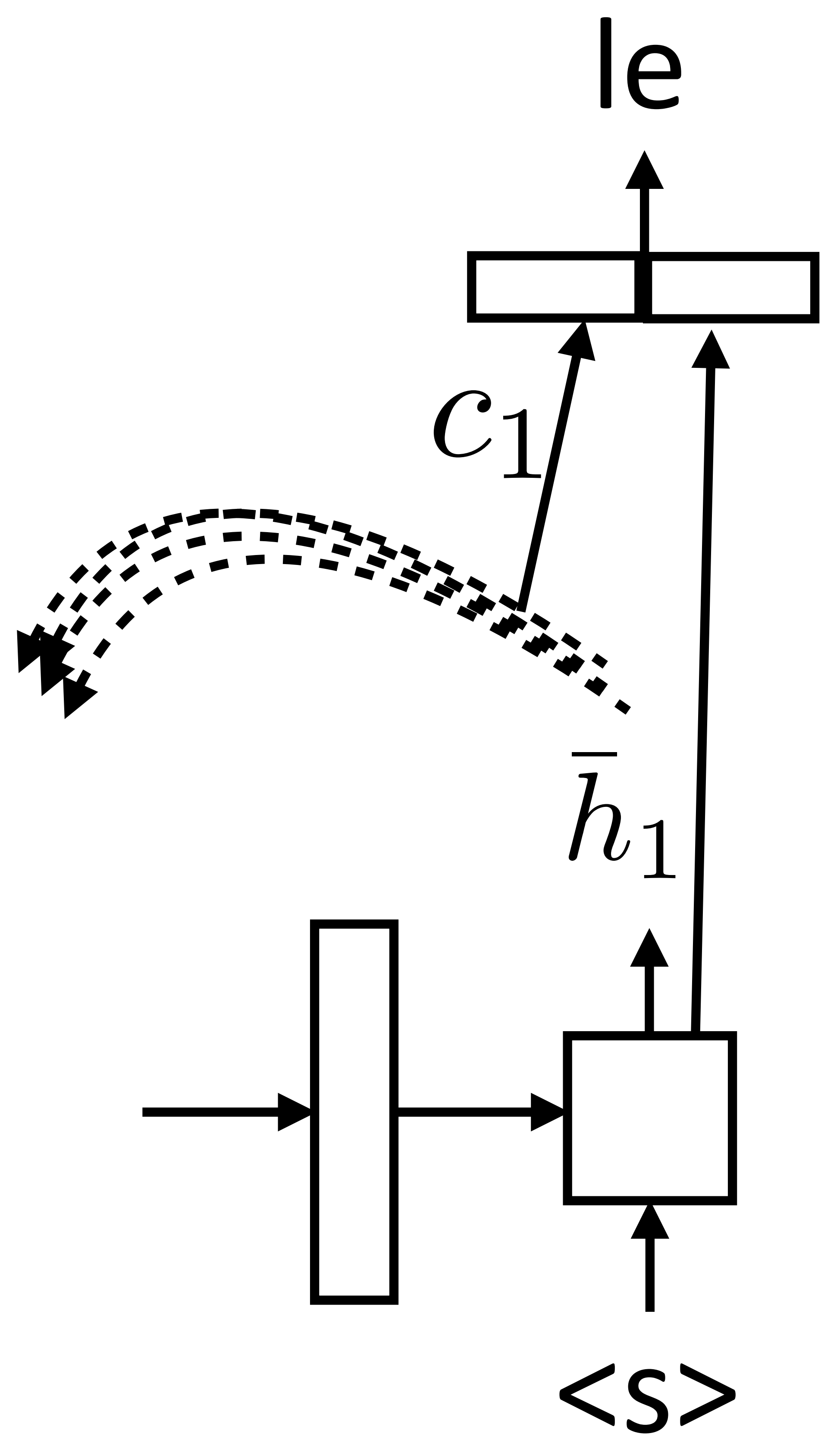
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$



$$e_{ij} = f(\bar{h}_i, h_j)$$

- Some function f (next slide)

Recap: Attention



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

- Bahdanau+ (2014): additive

$$f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j$$

- Luong+ (2015): dot product

$$f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j$$

- Luong+ (2015): bilinear

- Note that this all uses outputs of hidden layers

Copy / Pointer Networks

Unknown Words

en: The *ecotax* portico in *Pont-de-Buis* , . . . [truncated] . . . , was taken down on Thursday morning

fr: Le *portique* *écotaxe* de *Pont-de-Buis* , . . . [truncated] . . . , a été *démonté* jeudi matin

nn: Le unk de unk à unk , . . . [truncated] . . . , a été pris le jeudi matin

- Attention mechanism:

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W[c_i; \bar{h}_i])$$

from attention

from RNN hidden state

- Problems: want to be able to copy named entities like Pont-de-Buis, but target word has to be in the vocabulary, attention + RNN need to generate good embedding to pick it.

Jean et al. (2015), Luong et al. (2015)

Copying

en: The *ecotax* portico in *Pont-de-Buis* , . . . [truncated] . .

fr: Le *portique* *écotaxe* de *Pont-de-Buis* , . . . [truncated]

nn: Le unk de unk à unk , . . . [truncated] . . . , a été pris

Le
de
…
matin

Pont-de-Buis
ecotax

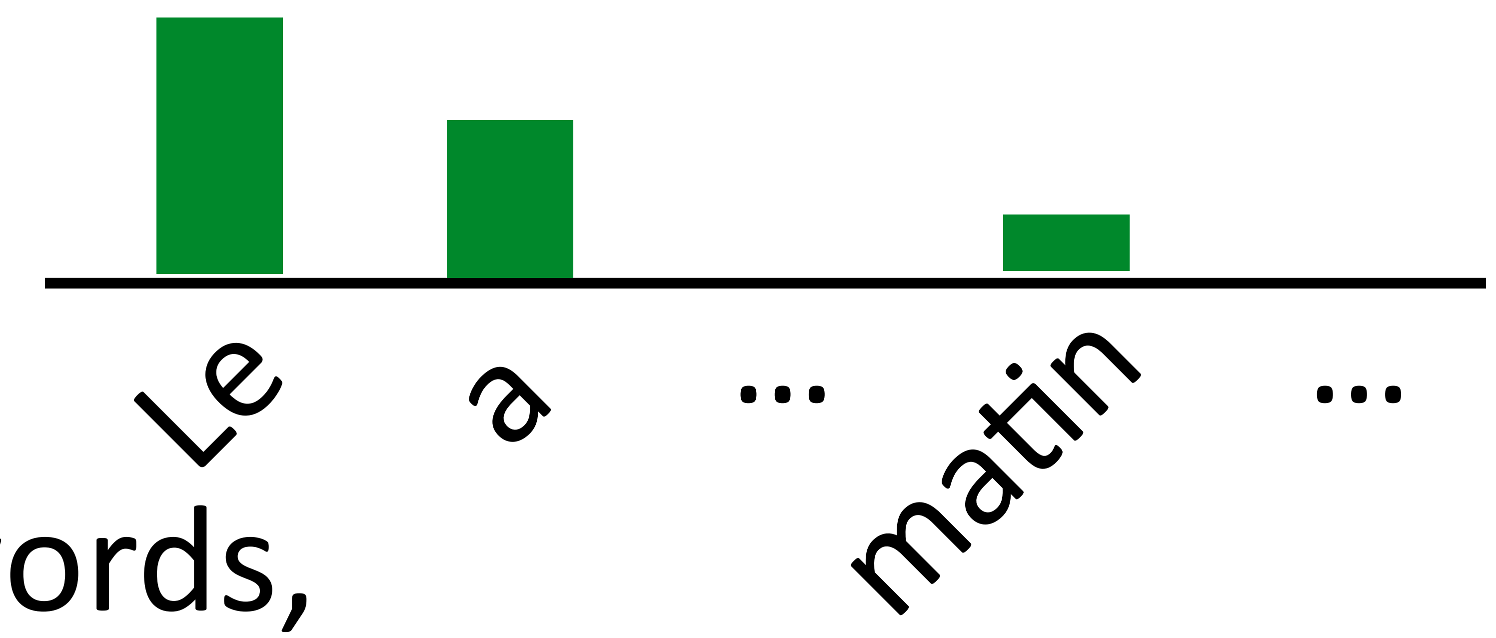
- Some words we want to copy may not be in the fixed output vocab (*Pont-de-Buis*)

- Solution: Vocabulary contains “normal” vocab as well as words in input.

Pointer Networks

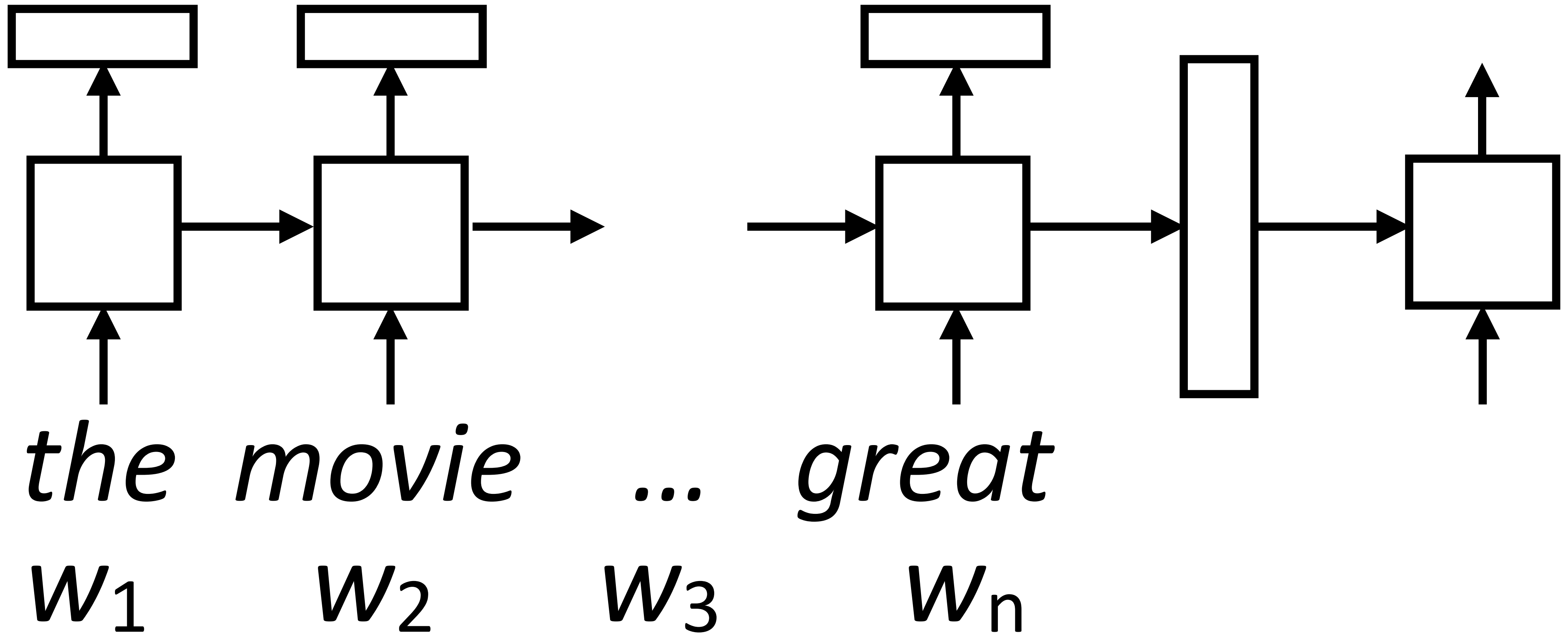
- Standard decoder (P_{vocab}): softmax over vocabulary

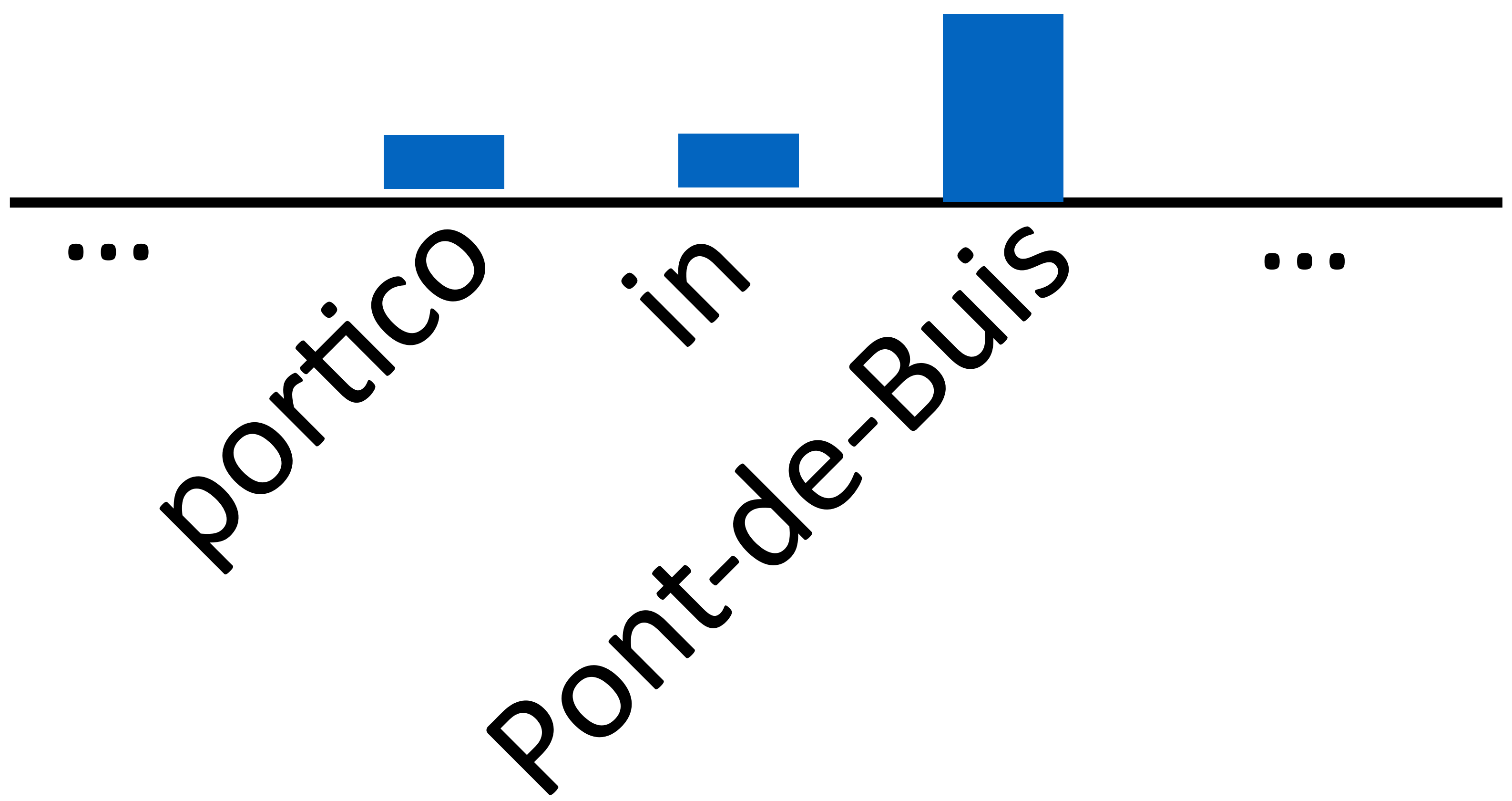
$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \text{softmax}(W[c_i; \bar{h}_i])$$



- Pointer network (P_{pointer}): predict from *source* words, instead of target vocabulary

$$P_{\text{pointer}}(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) \propto \begin{cases} \exp(h_j^\top V \bar{h}_i) & \text{if } y_i = w_j \\ 0 \text{ otherwise} & \end{cases}$$





Pointer Generator Mixture Models

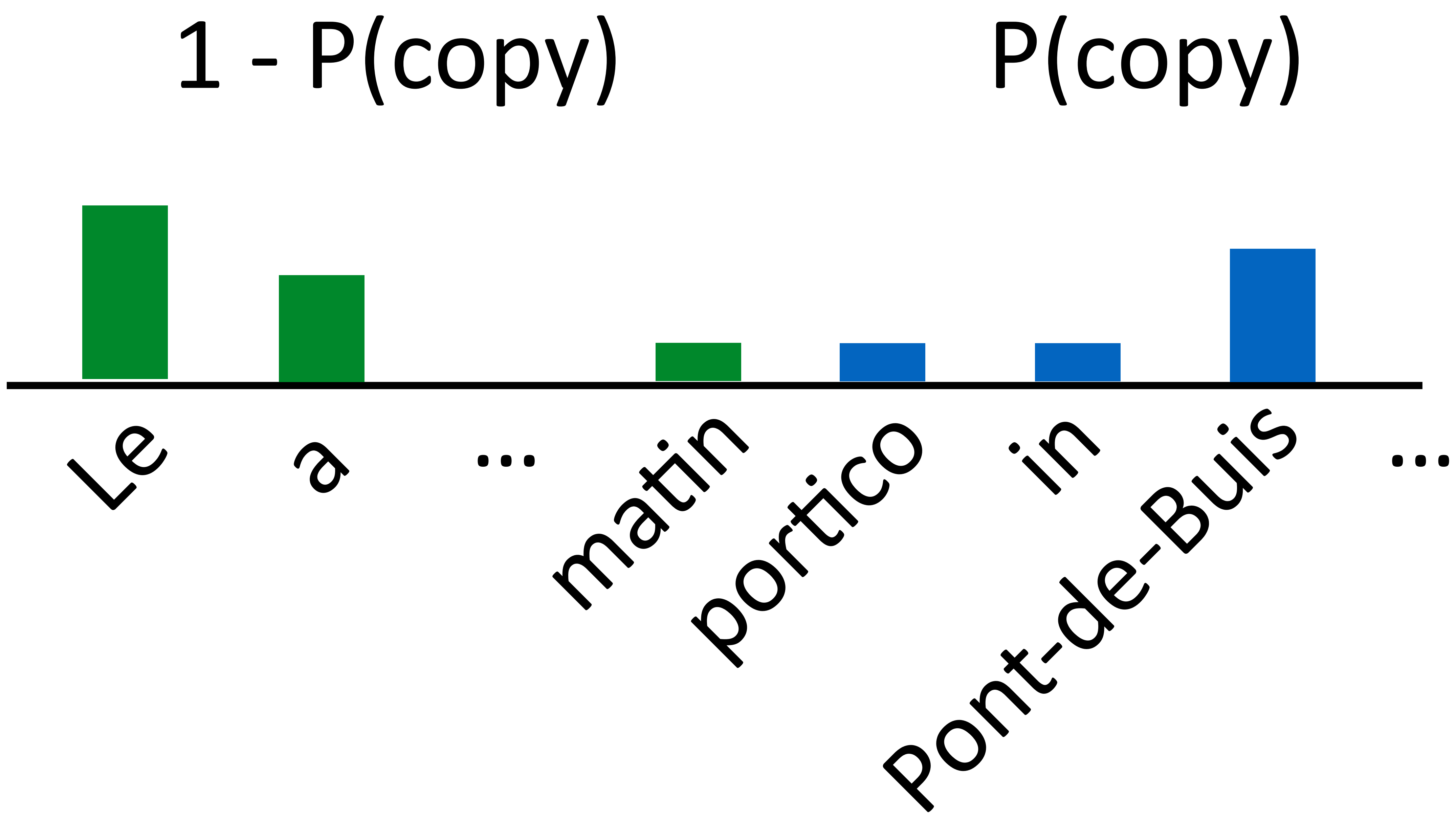
- Define the decoder model as a mixture model of P_{vocab} and P_{pointer}

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = P(\text{copy})P_{\text{pointer}} + (1 - P(\text{copy}))P_{\text{vocab}}$$

- Predict P(copy) based on decoder state, input, etc.

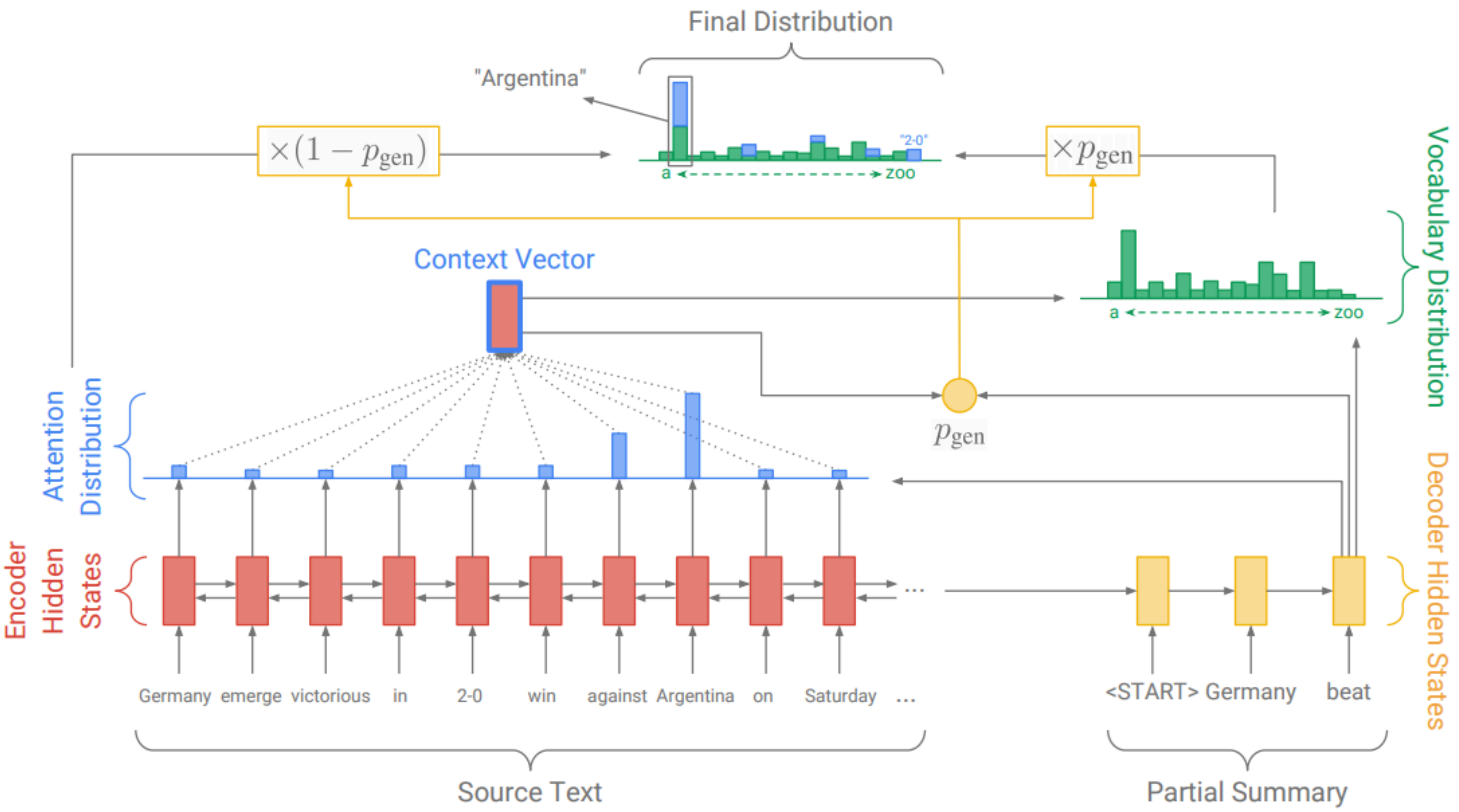
- Marginalize over copy variable during training and inference

- Model will be able to both generate and copy, flexibly adapt between the two



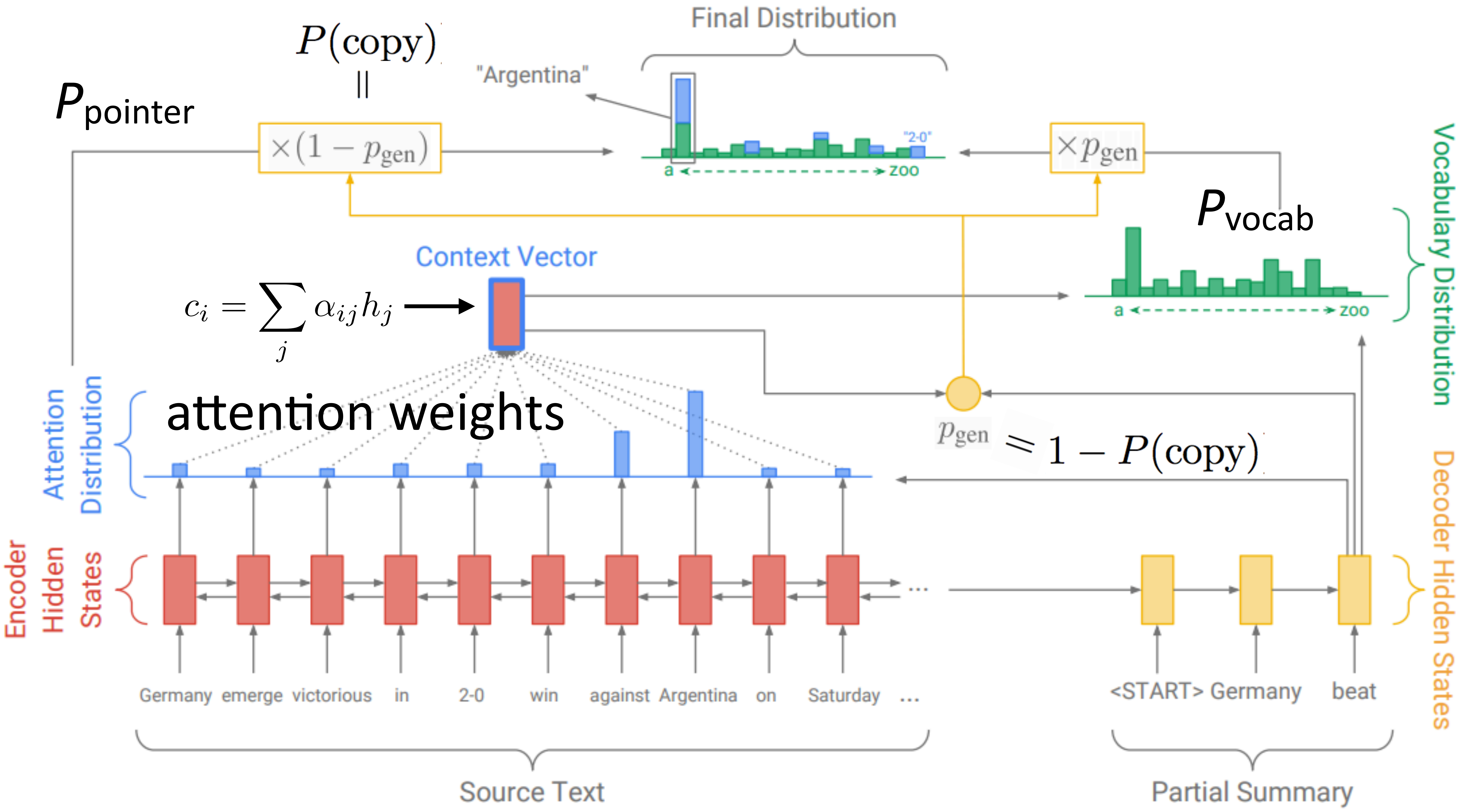
Gulcehre et al. (2016), Gu et al. (2016)

Copying in Summarization



See et al. (2017)

Copying in Summarization



See et al. (2017)

Copying in Summarization

	ROUGE			METEOR	
	1	2	L	exact match	+ stem/syn/para
abstractive model (Nallapati et al., 2016)*	35.46	13.30	32.65	-	-
seq-to-seq + attn baseline (150k vocab)	30.49	11.17	28.08	11.65	12.86
seq-to-seq + attn baseline (50k vocab)	31.33	11.81	28.83	12.03	13.20
pointer-generator	36.44	15.66	33.42	15.35	16.65
pointer-generator + coverage	**39.53**	**17.28**	**36.38**	17.32	18.72
lead-3 baseline (ours)	40.34	17.70	36.57	20.48	22.21
lead-3 baseline (Nallapati et al., 2017)*	39.2	15.7	35.5	-	-
extractive model (Nallapati et al., 2017)*	39.6	16.2	35.3	-	-

- maintain a coverage vector, which is the sum of attention distributions over all previous decoder timesteps

See et al. (2017)

Copying in Summarization

Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, ***muhammadu buhari*** told cnn's christiane amanpour that **he plans to aggressively fight corruption that has long plagued nigeria** and go after the root of the nation's unrest. *buhari* said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, **he said his administration is confident it will be able to thwart criminals** and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. *buhari* defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. **the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.**

Baseline Seq2Seq + Attention: **UNK UNK** says his administration is confident it will be able to **destabilize nigeria's economy**. **UNK** says his administration is confident it will be able to thwart criminals and other **nigerians**. **he says the country has long nigeria and nigeria's economy.**

Pointer-Gen: *muhammadu buhari* says he plans to aggressively fight corruption **in the northeast part of nigeria**. he says he'll "rapidly give attention" to curbing violence **in the northeast part of nigeria**. he says his administration is confident it will be able to thwart criminals.

Pointer-Gen + Coverage: *muhammadu buhari* says he plans to aggressively fight corruption that has long plagued nigeria. he says his administration is confident it will be able to thwart criminals. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

Figure 1: Comparison of output of 3 abstractive summarization models on a news article. The baseline model makes **factual errors**, a **nonsensical sentence** and struggles with OOV words *muhammadu buhari*. The pointer-generator model is accurate but **repeats itself**. Coverage eliminates repetition. The final summary is composed from **several fragments**.

See et al. (2017)

Transformers

Attention is All You Need

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

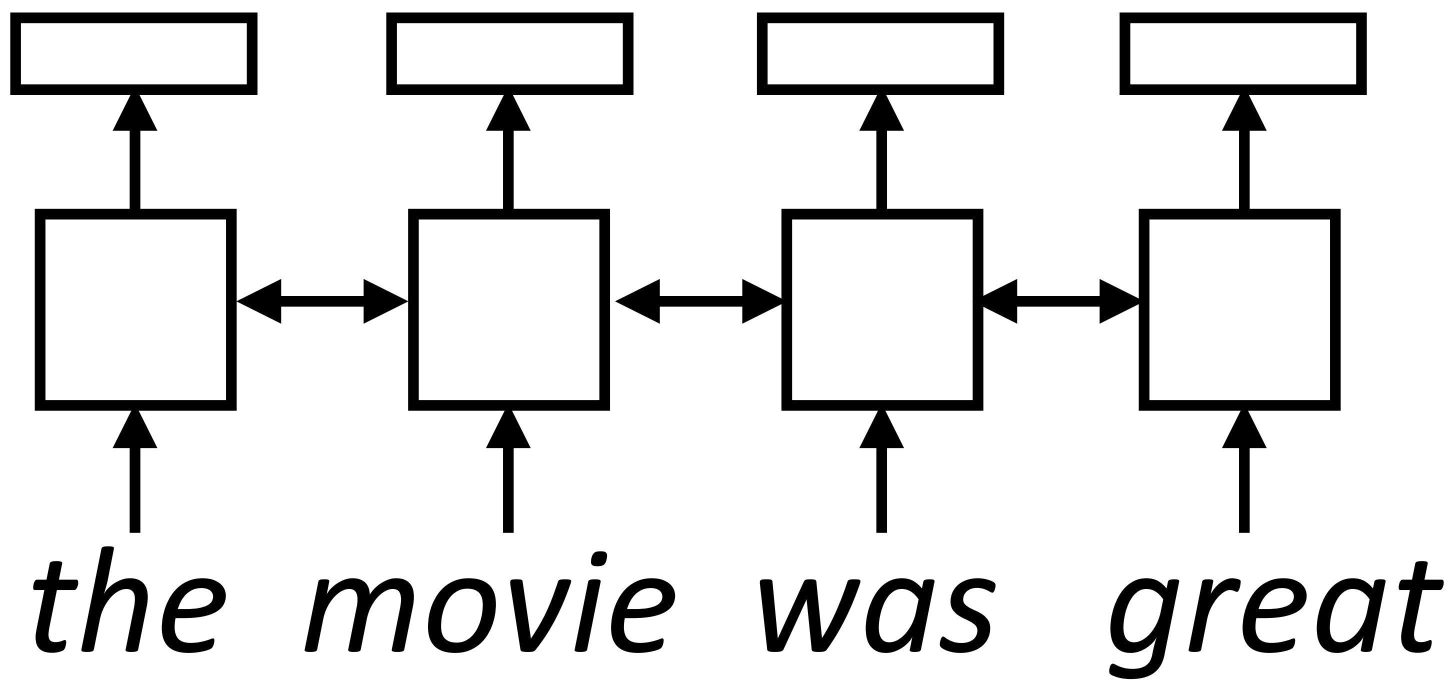
Vaswani et al. (2017)

Readings

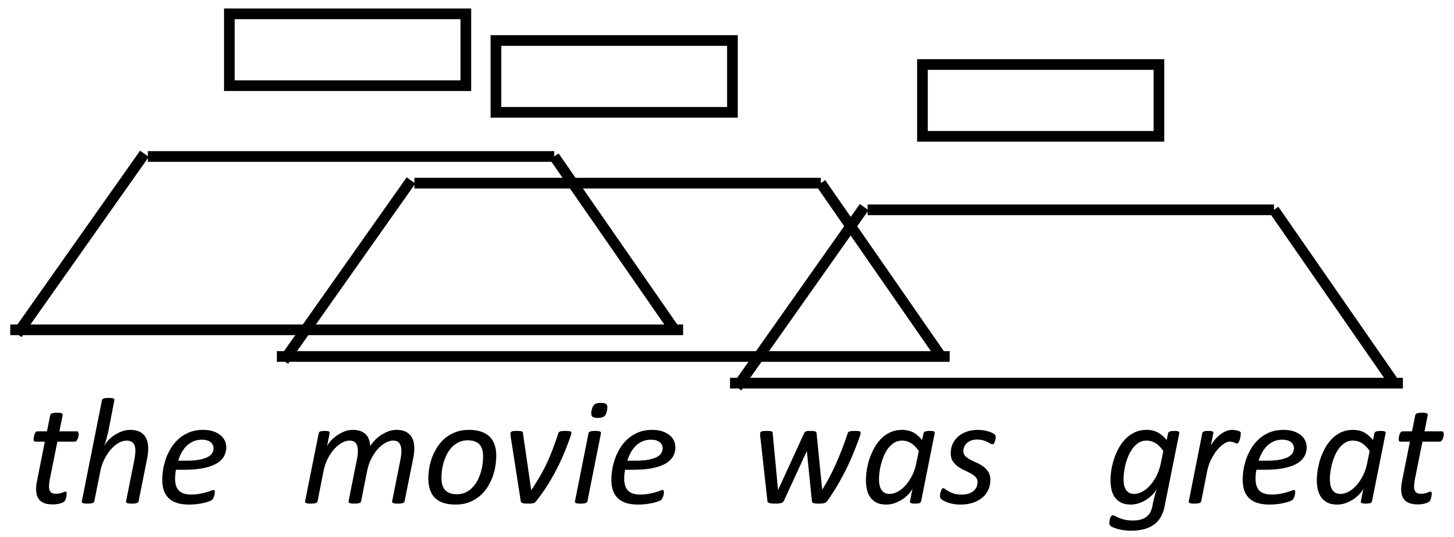
- "The Annotated Transformer" by Sasha Rush
  https://nlp.seas.harvard.edu/2018/04/03/attention.html
- "The Illustrated Transformer" by Jay Lamar
  http://jalammar.github.io/illustrated-transformer/

Sentence Encoders

- LSTM abstraction: maps each vector in a sentence to a new, context-aware vector



- CNNs do something similar with filters



- Attention can give us a third way to do this

Vaswani et al. (2017)

Self-Attention

- Assume we're using GloVe/word2vec embeddings — what do we want our neural network to do?

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

- Q: What words need to be contextualized here?

Vaswani et al. (2017)

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)

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

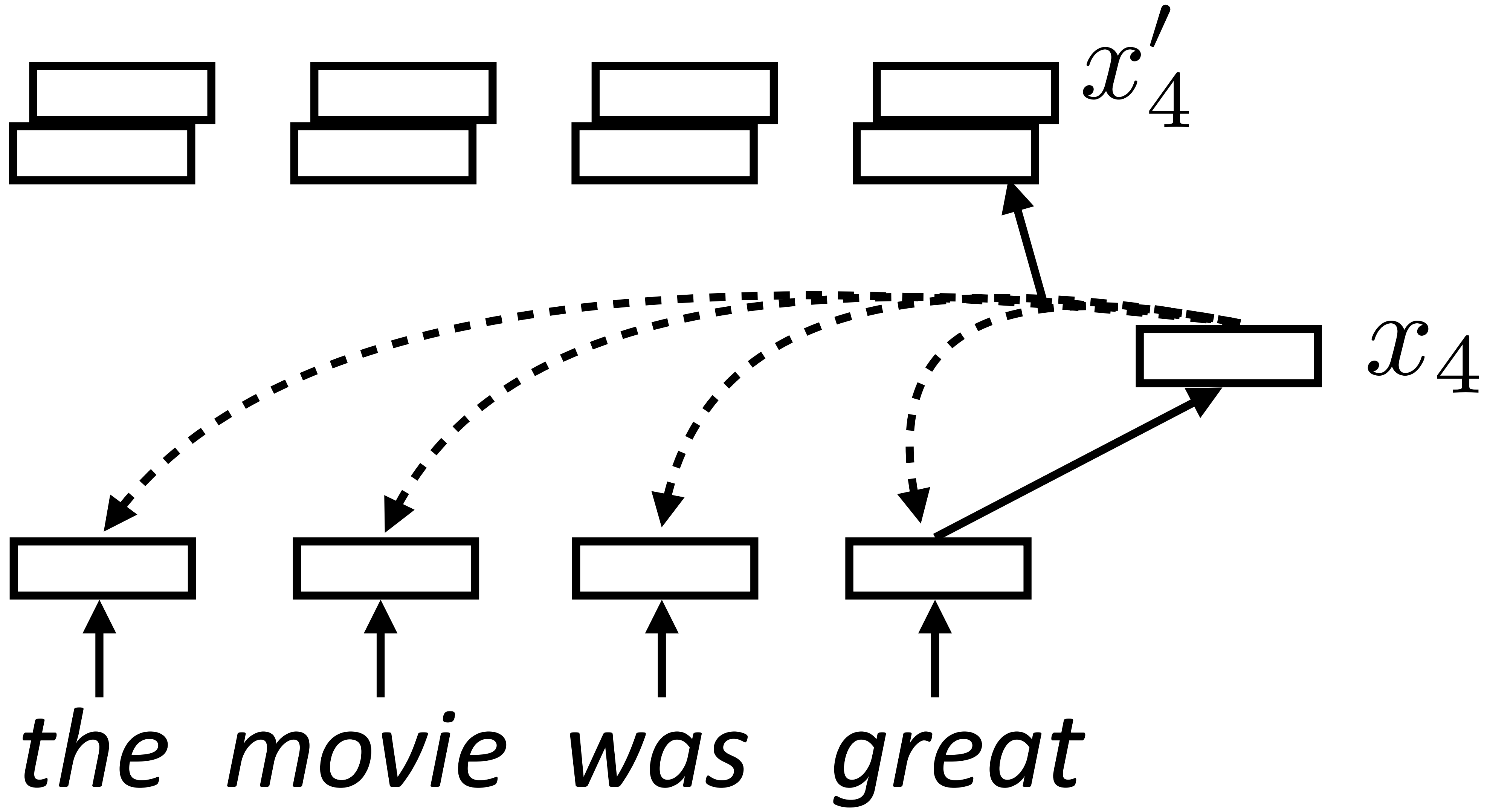
Vaswani et al. (2017)

Self-Attention

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

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

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

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

Vaswani et al. (2017)

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
- This is a demonstration, we will revisit what these models actually learn when we discuss BERT
- 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

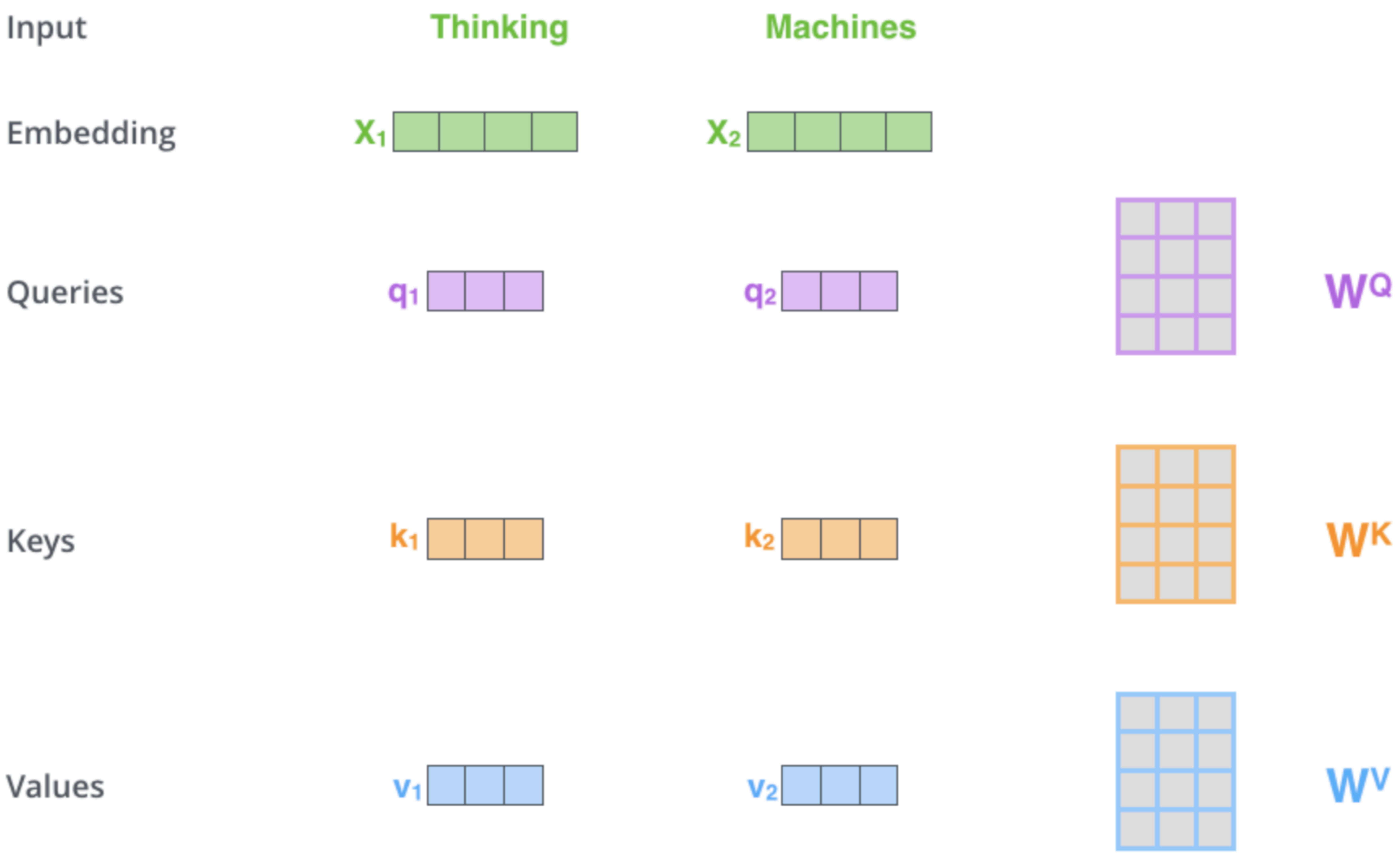
- Multiple “heads” analogous to different convolutional filters
- Let X = [sent len, embedding dim] be the input sentence
- Query $Q = W^Q X$: these are like the **decoder hidden state** in attention
- Keys $K = W^K X$: these control what gets attended to, along with the query
- Values $V = W^V X$: these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

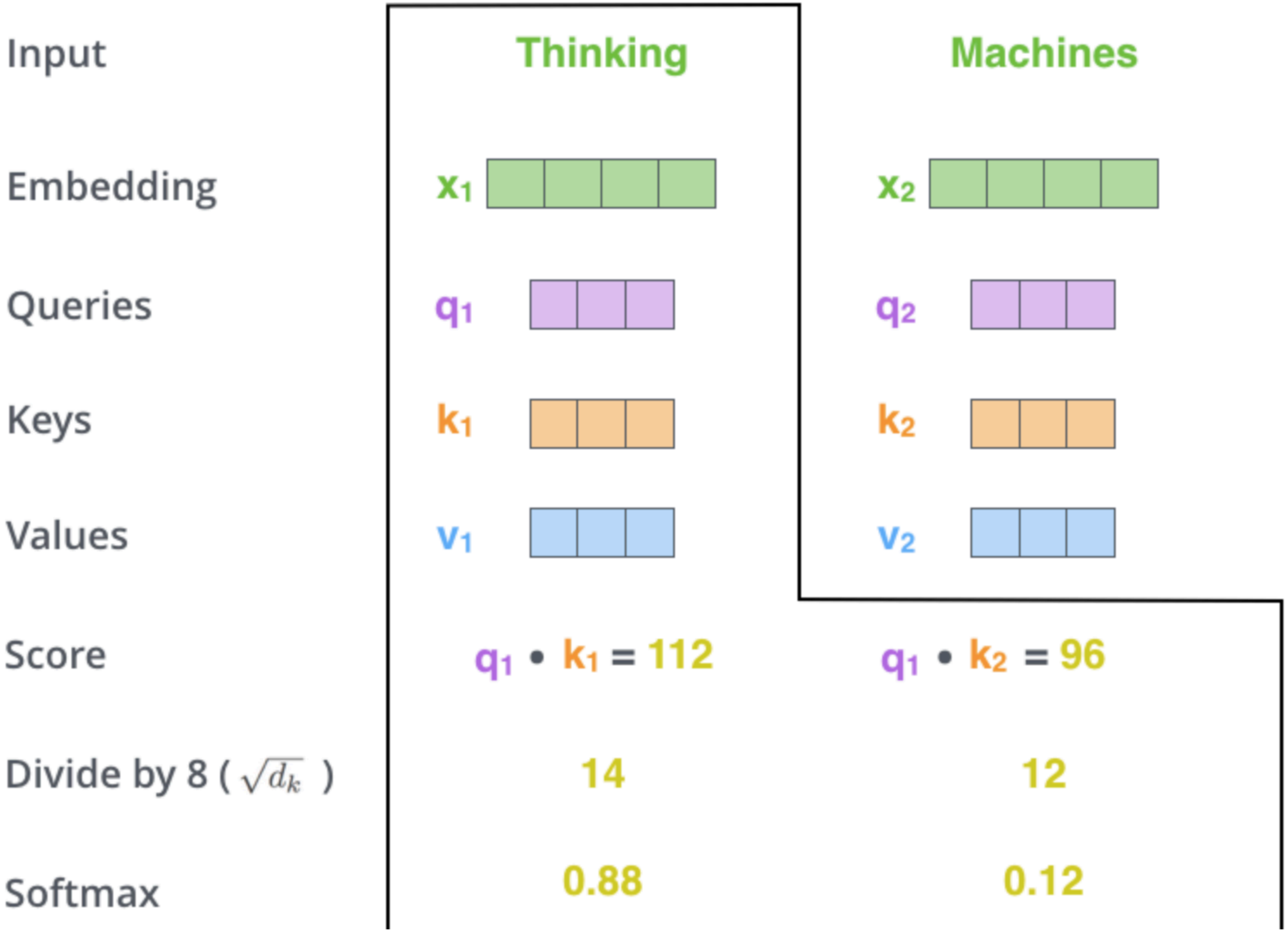
Vaswani et al. (2017)

Multi-Head Self Attention



Credit: Alammar, *The Illustrated Transformer*

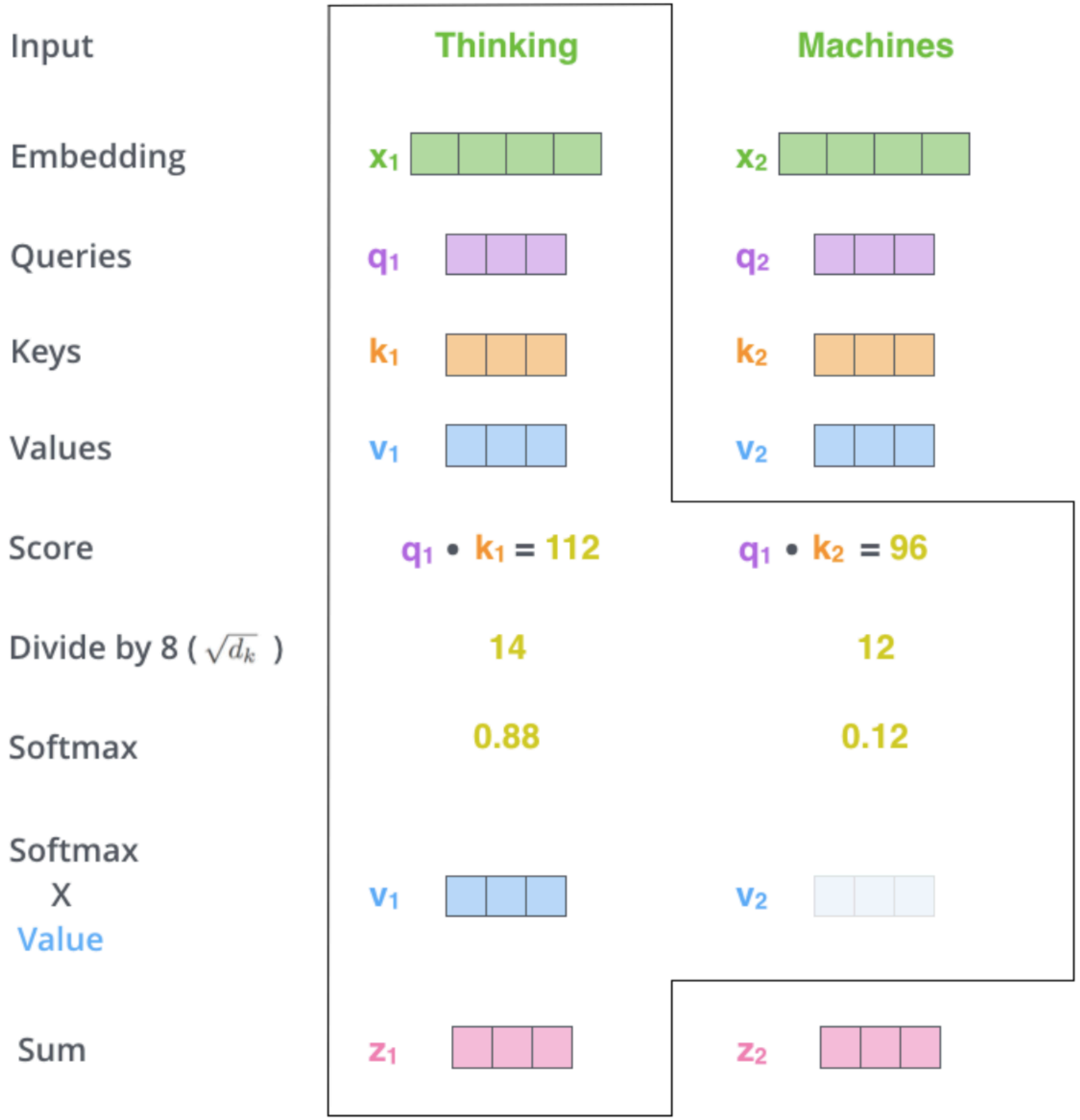
Multi-Head Self Attention



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

Credit: Alammar, *The Illustrated Transformer*

Multi-Head Self Attention

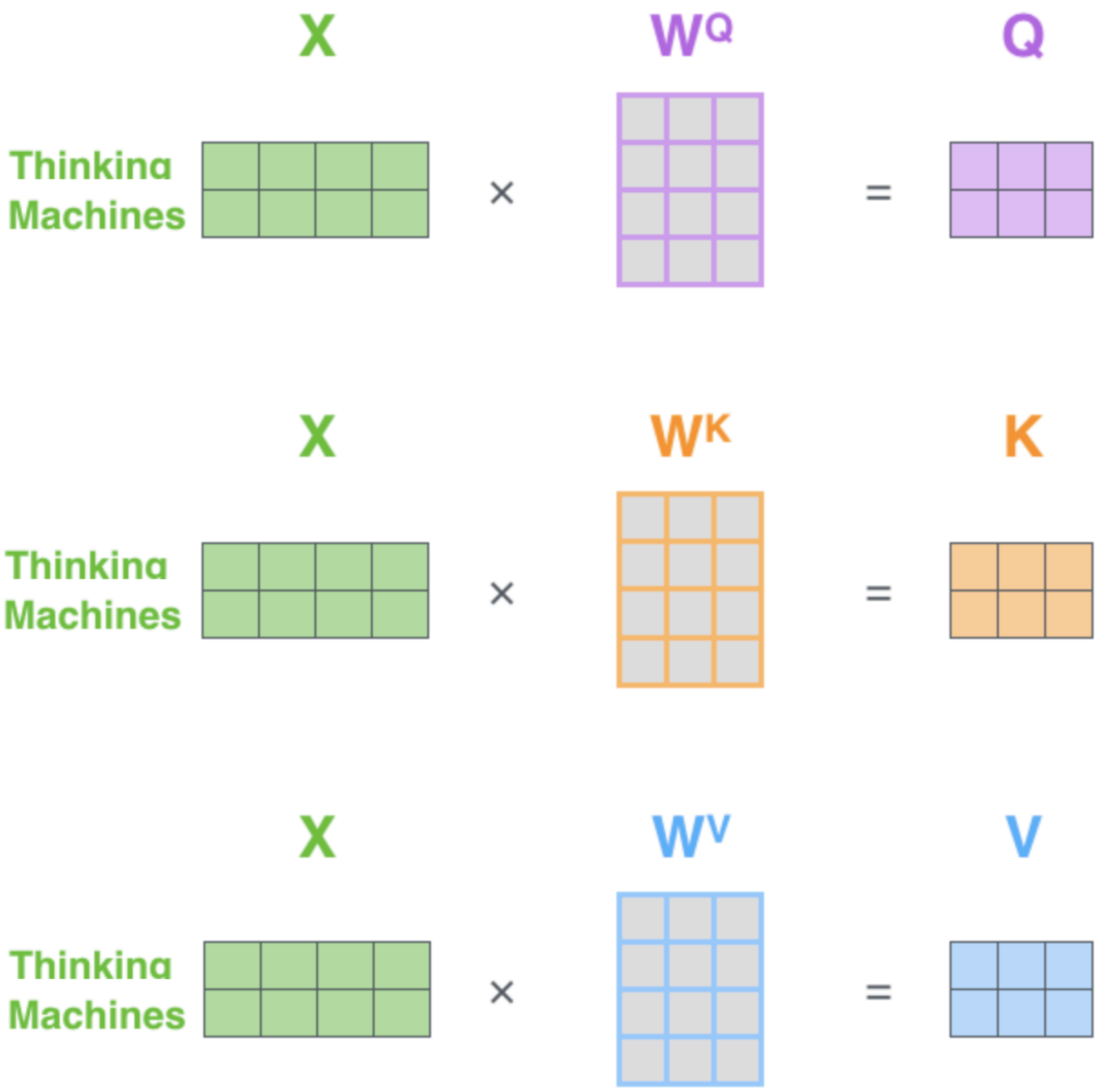


$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

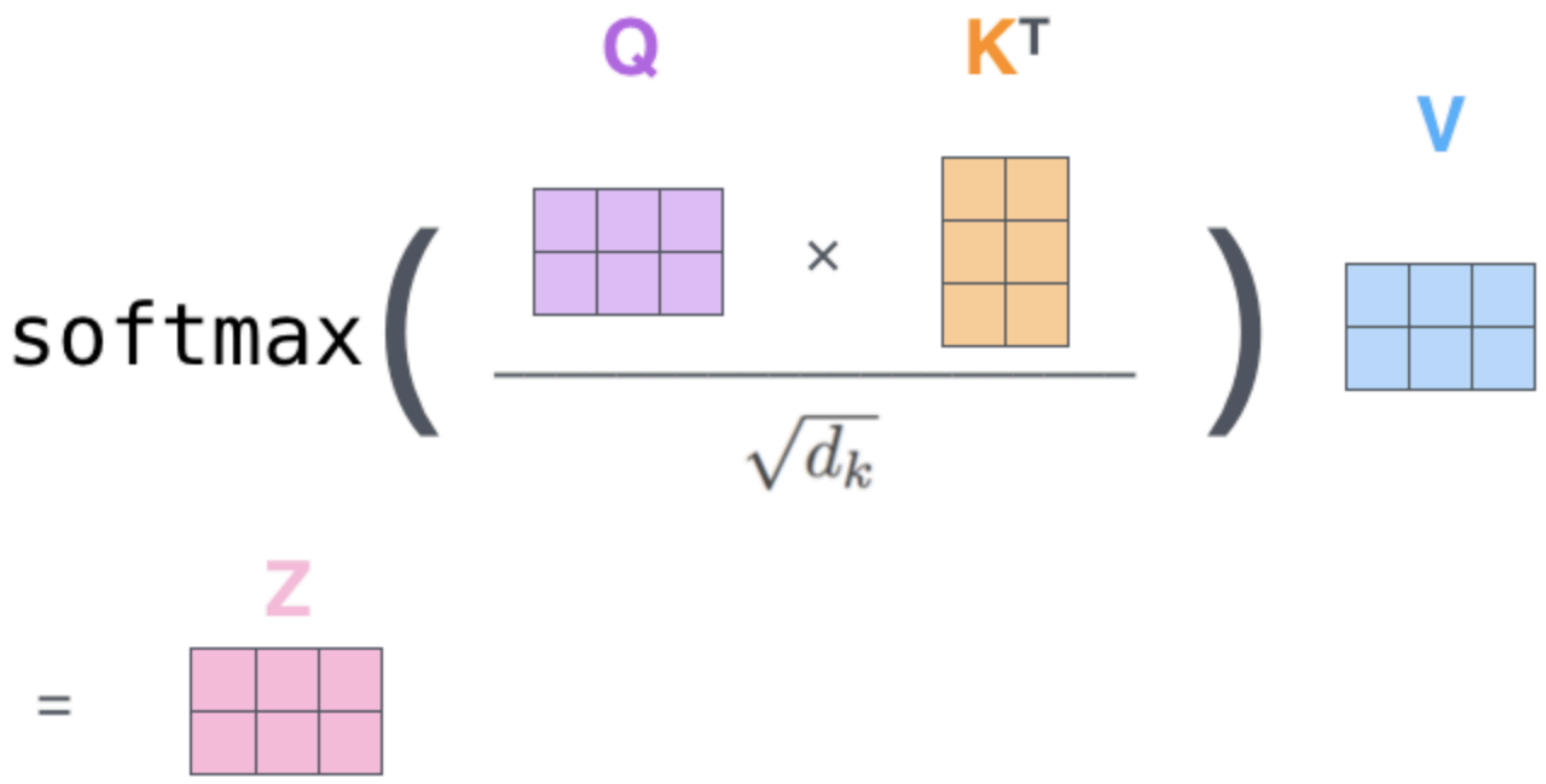
Credit: Alammar, *The Illustrated Transformer*

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)

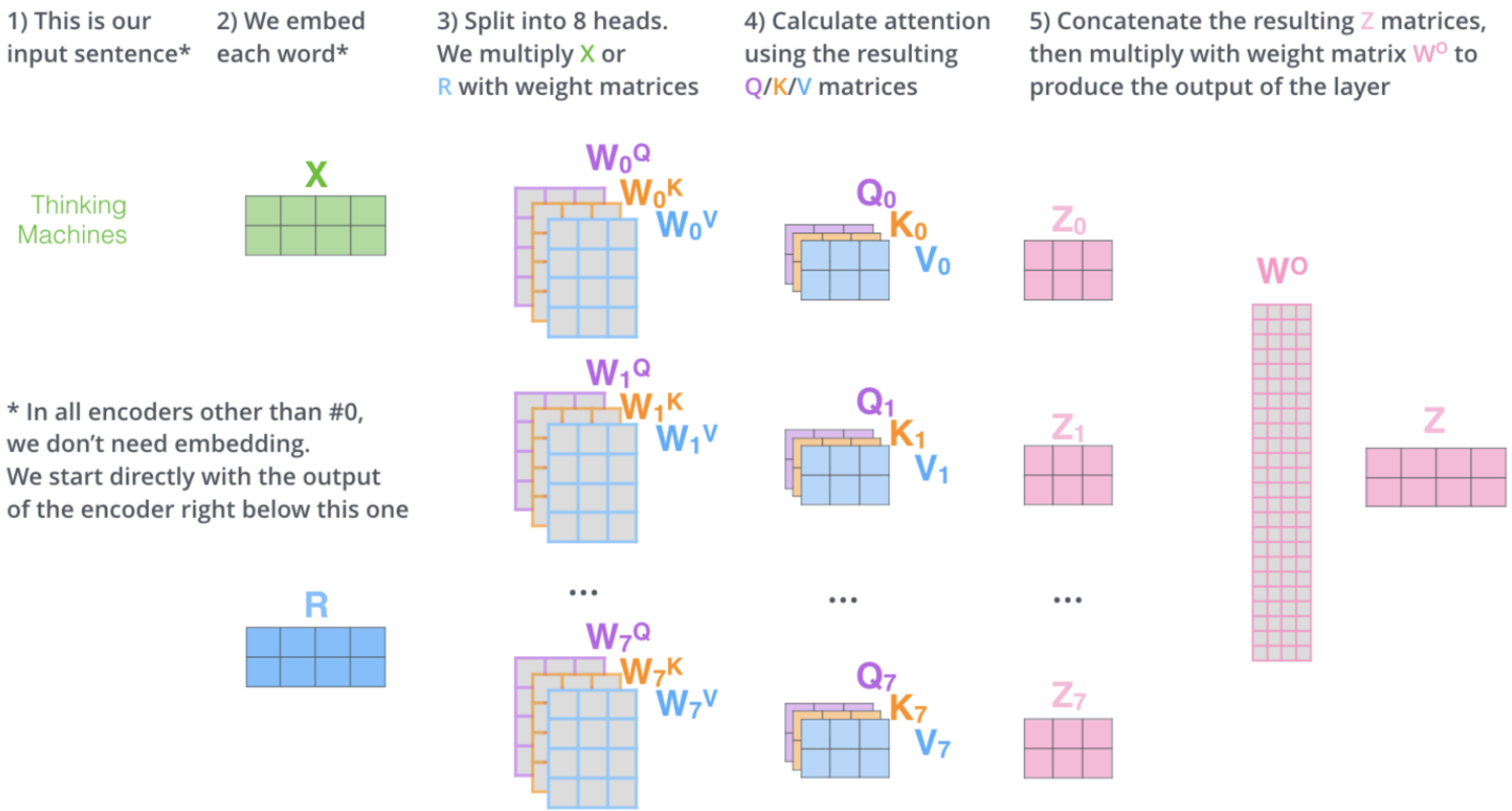


sent len x hidden dim

Z is a weighted combination of V rows

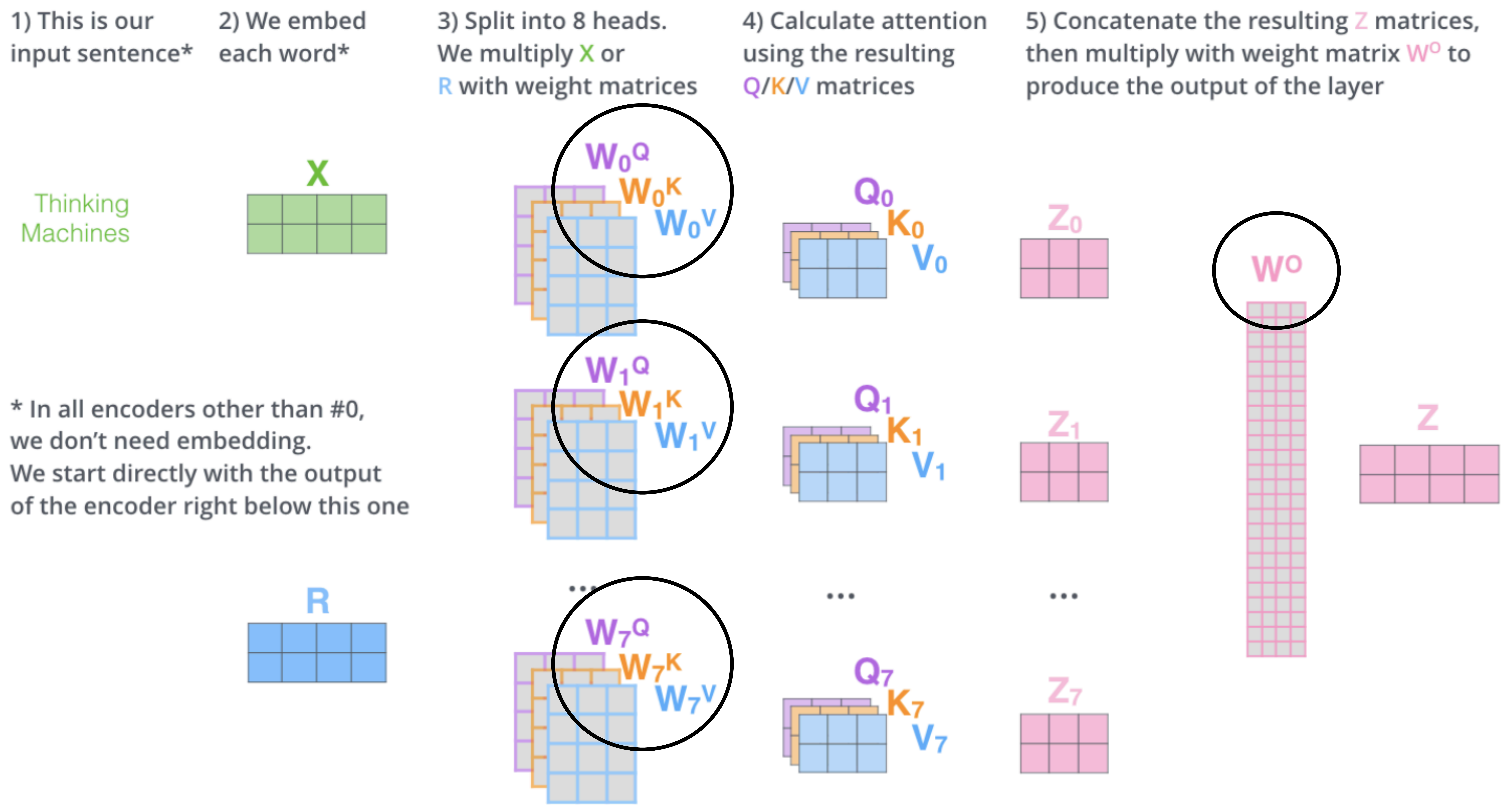
Credit: Alammar, *The Illustrated Transformer*

Multi-Head Self Attention



Credit: Alammar, *The Illustrated Transformer*

Multi-Head Self Attention



Credit: Alammar, *The Illustrated Transformer*

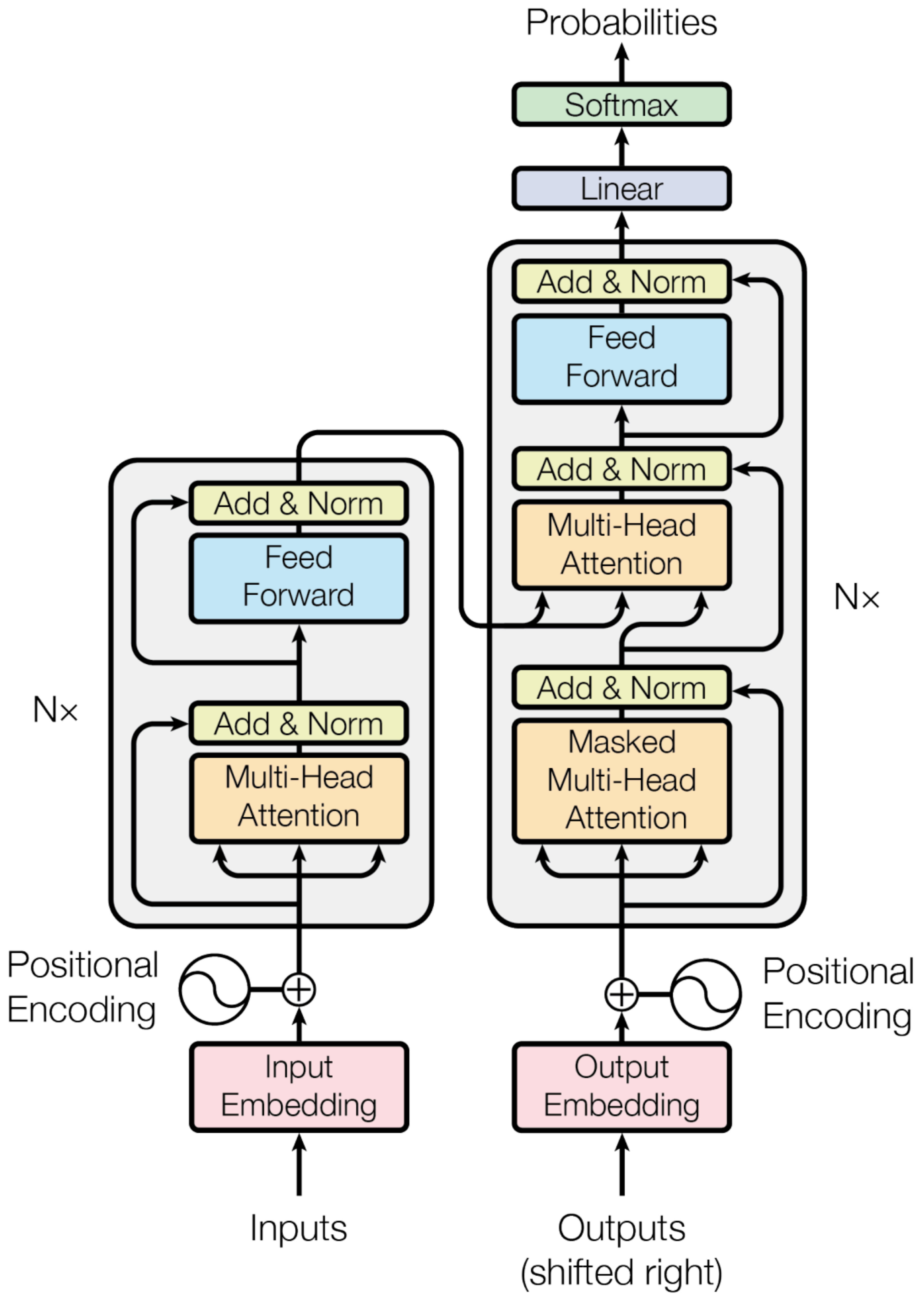
Properties of Self-Attention

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)

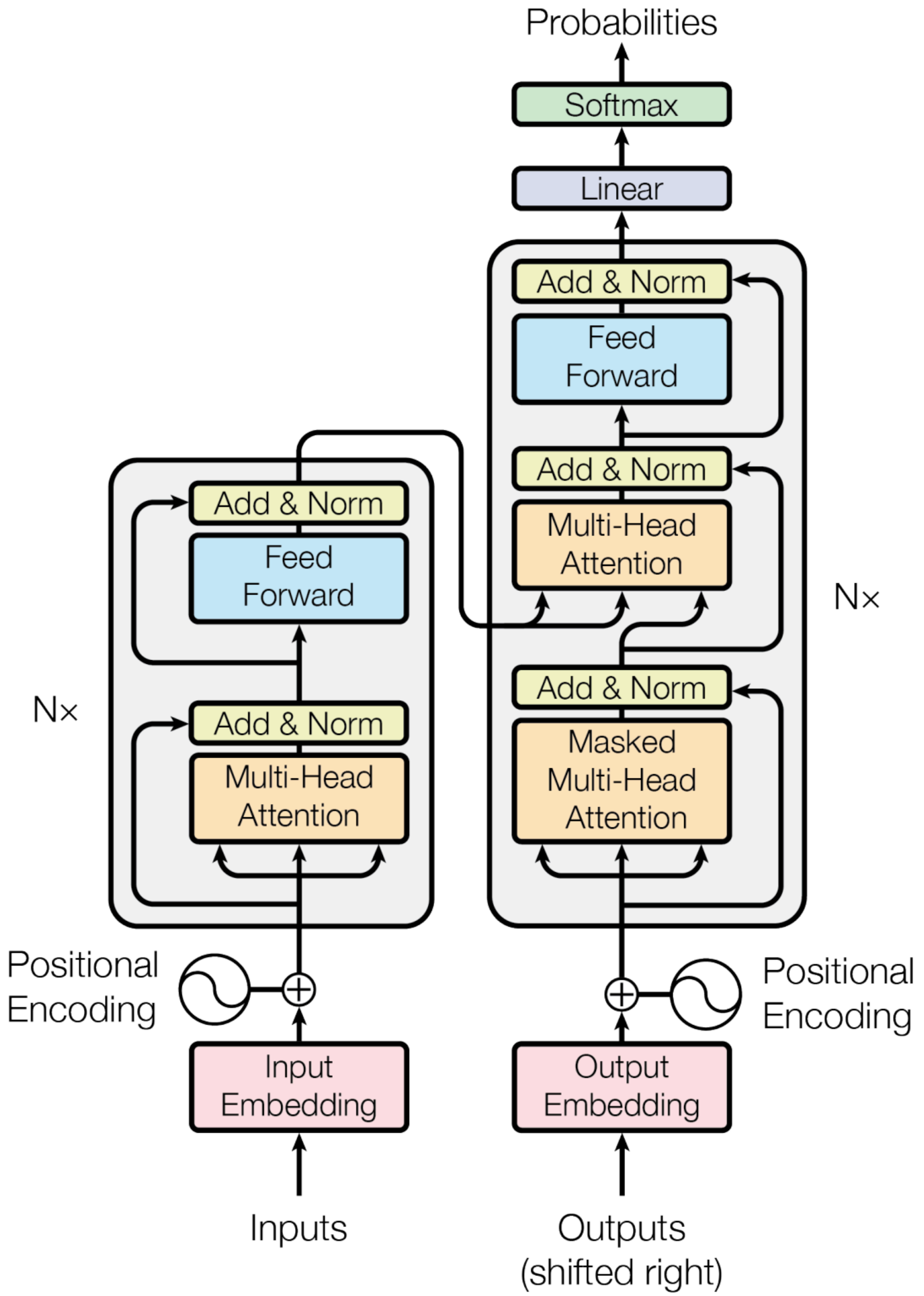
Transformers for MT: Complete Model



- 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)

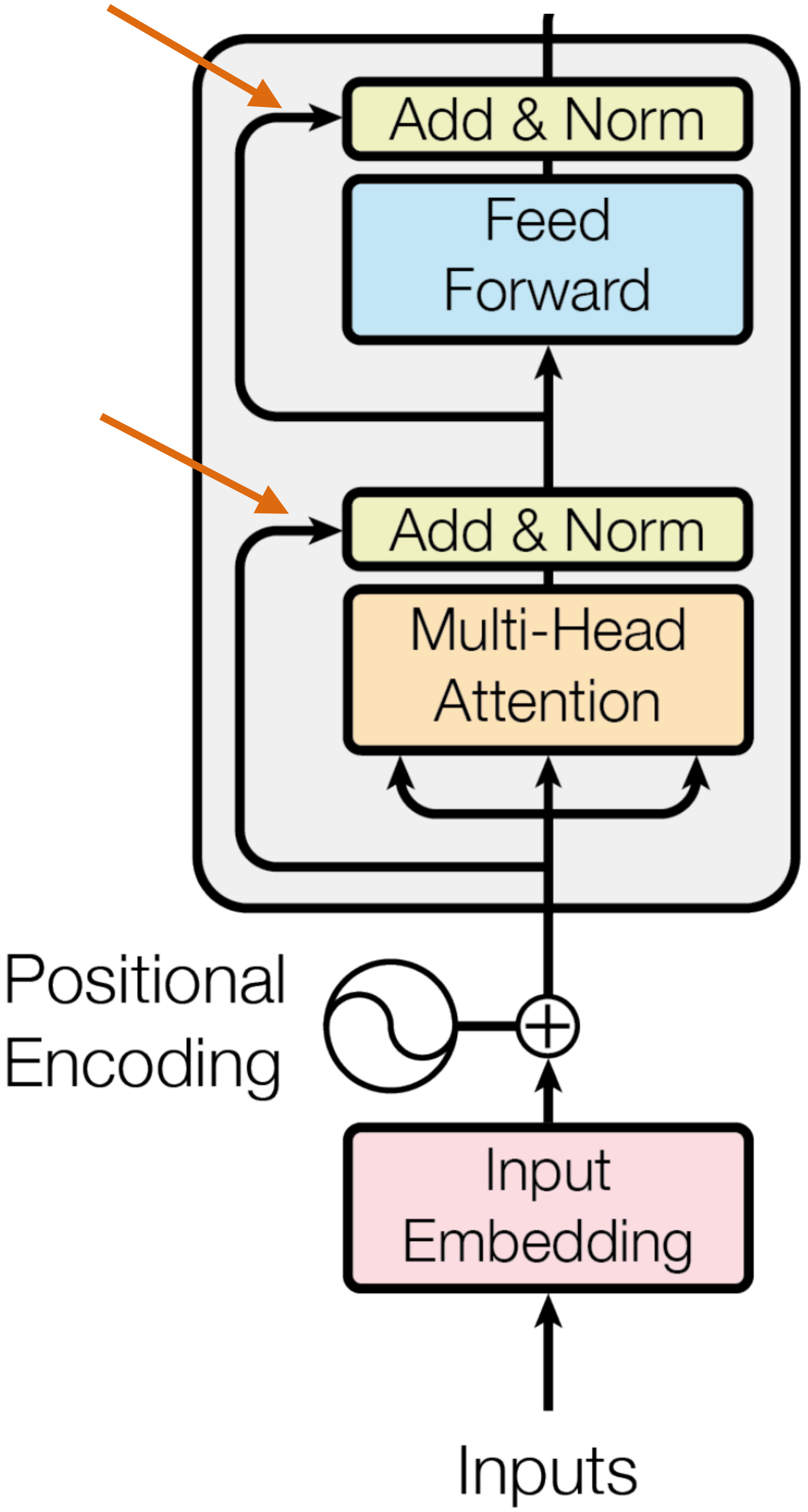
Transformers for MT: Complete Model



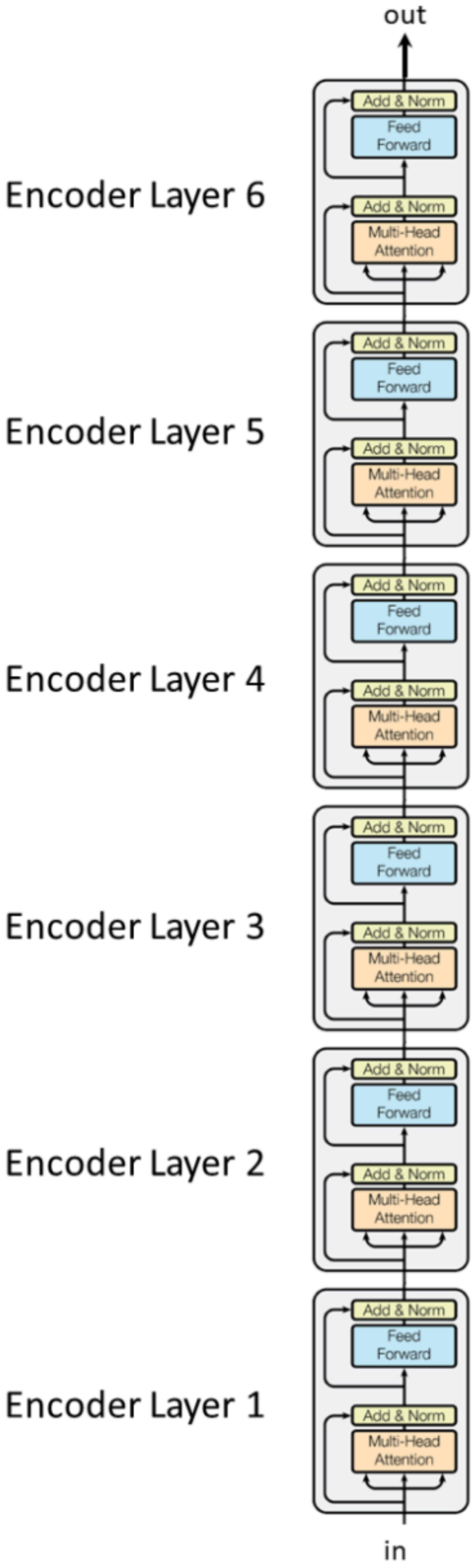
- Many other details to get it to work: residual connections, layer normalization, positional encoding, optimizer with learning rate schedule, label smoothing ….

Vaswani et al. (2017)

Transformers



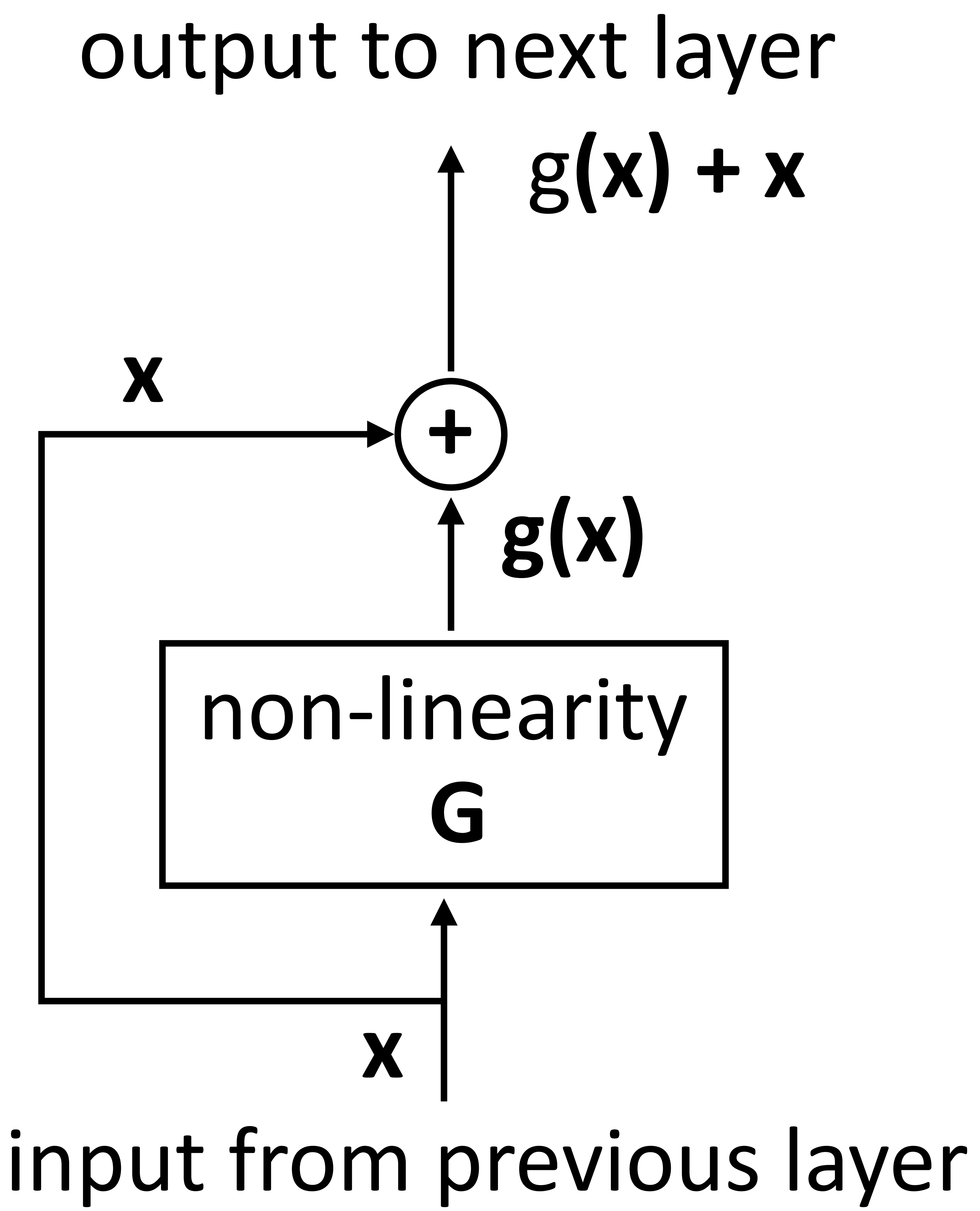
- Alternate multi-head self-attention layers and feedforward layers
- Residual connections let the model "skip" each layer — these are particularly useful for training deep networks



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)

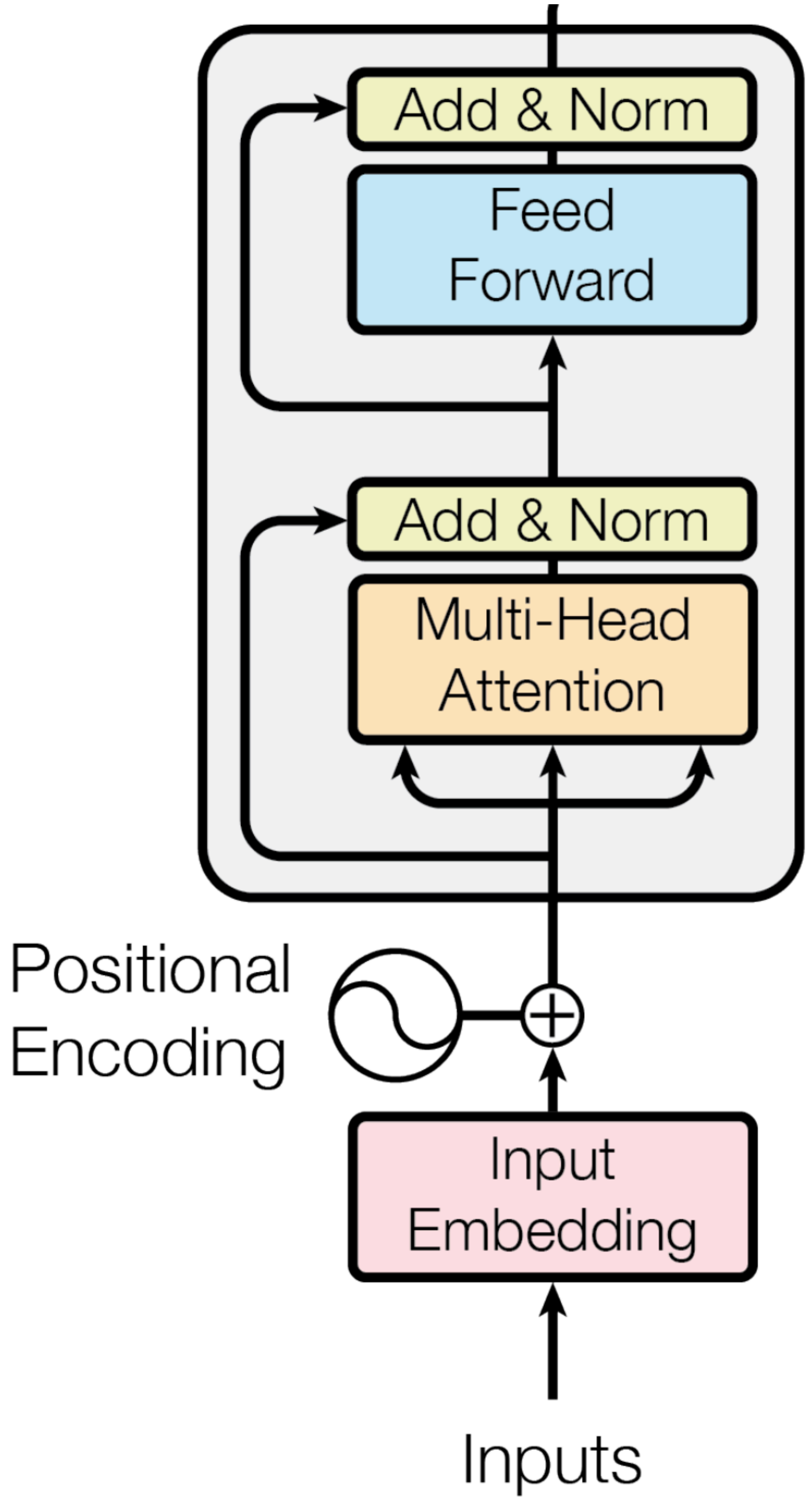
Transformers: Position Sensitivity

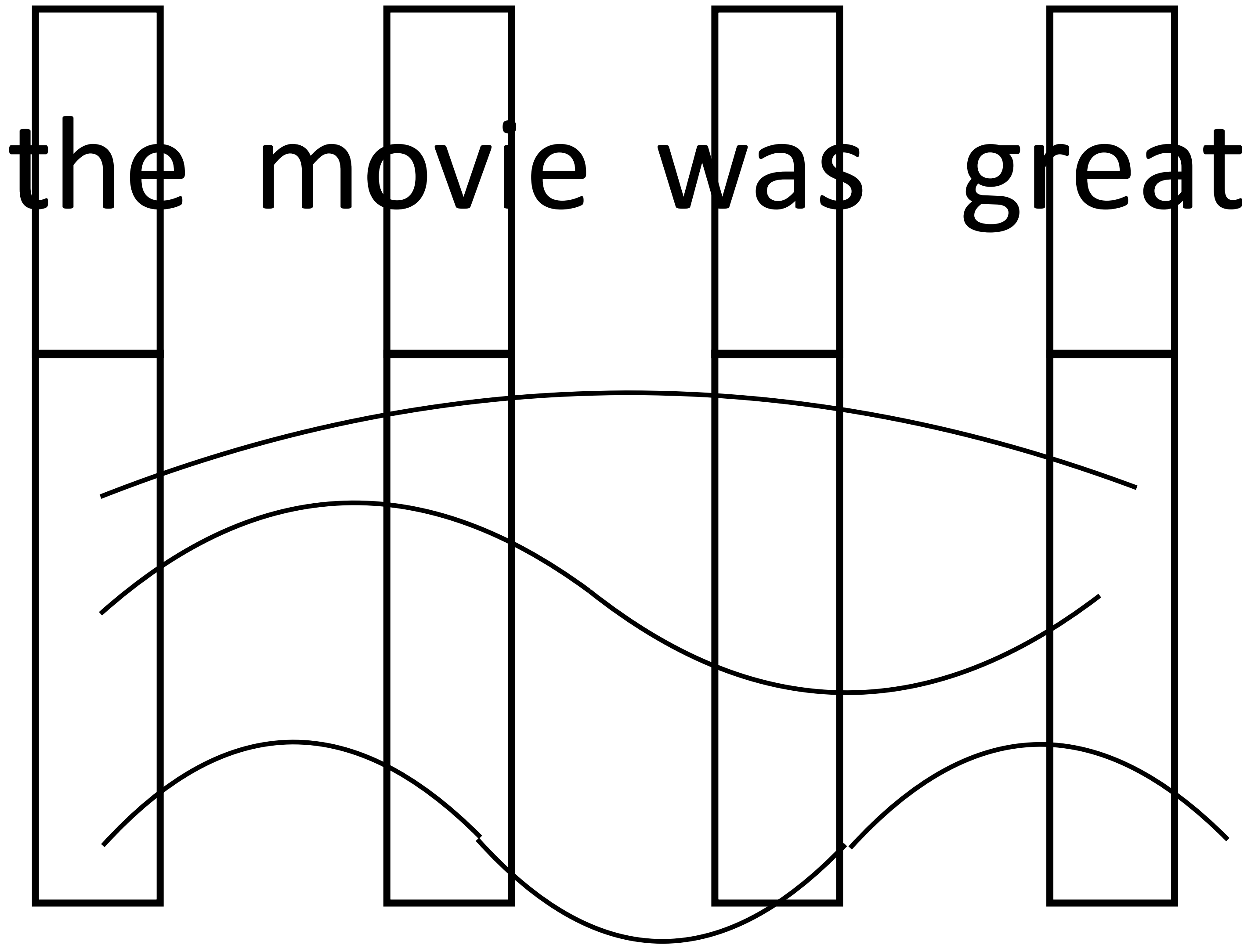
The ballerina is very excited that she will dance in the show.

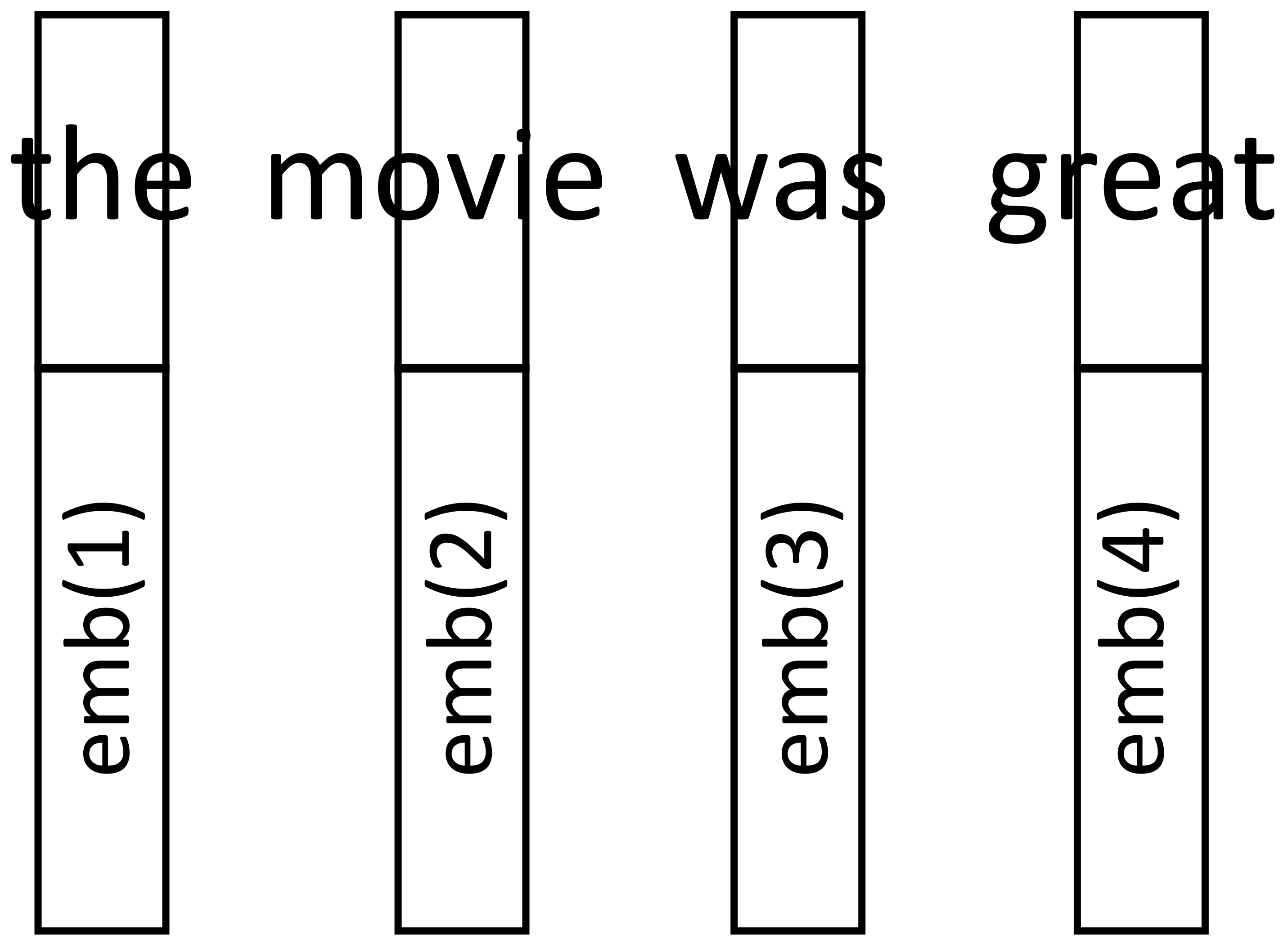
- If this is in a longer context, we want words to attend *locally*
- But transformers have *no notion of position* by default

Vaswani et al. (2017)

Transformers



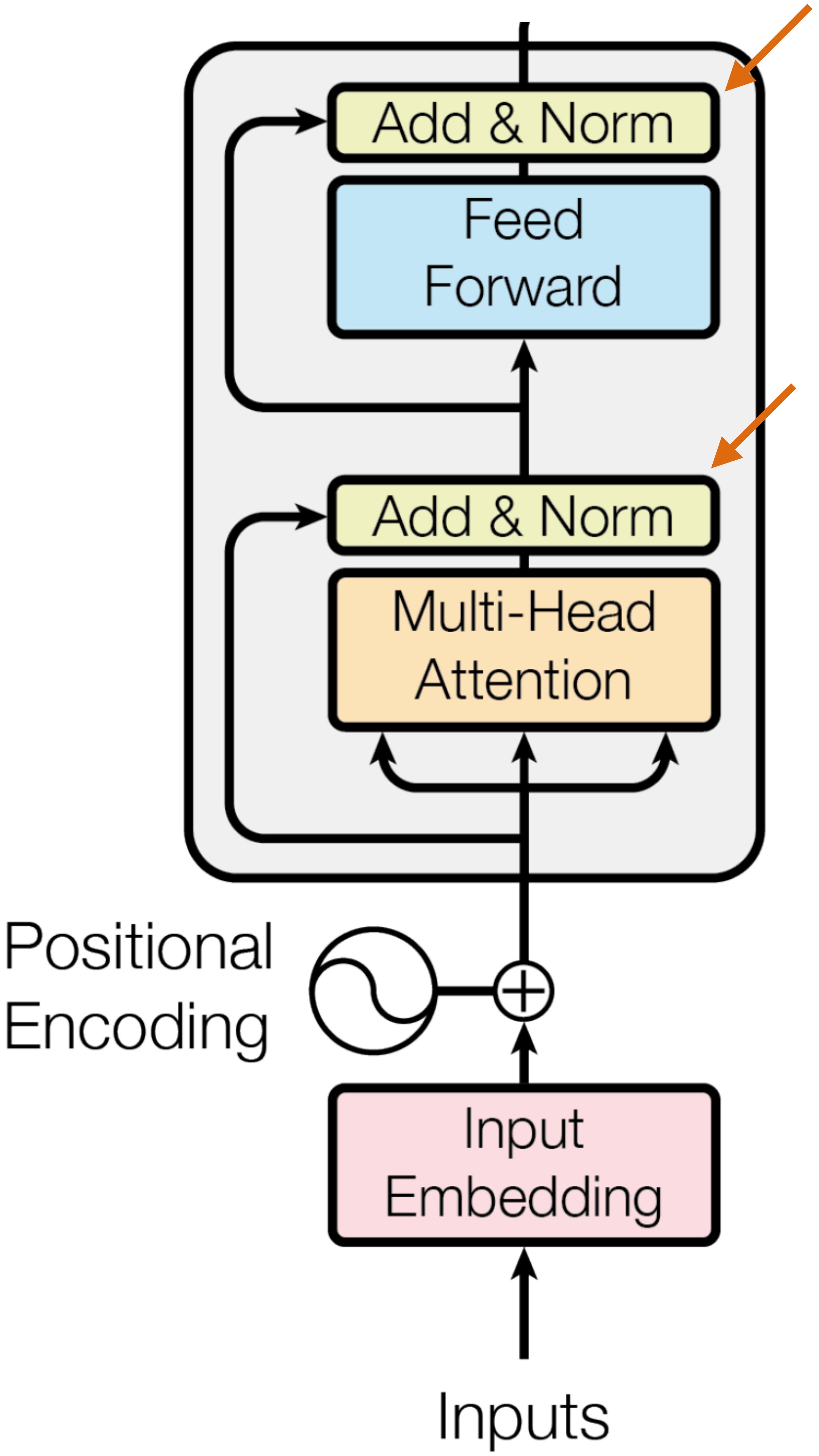




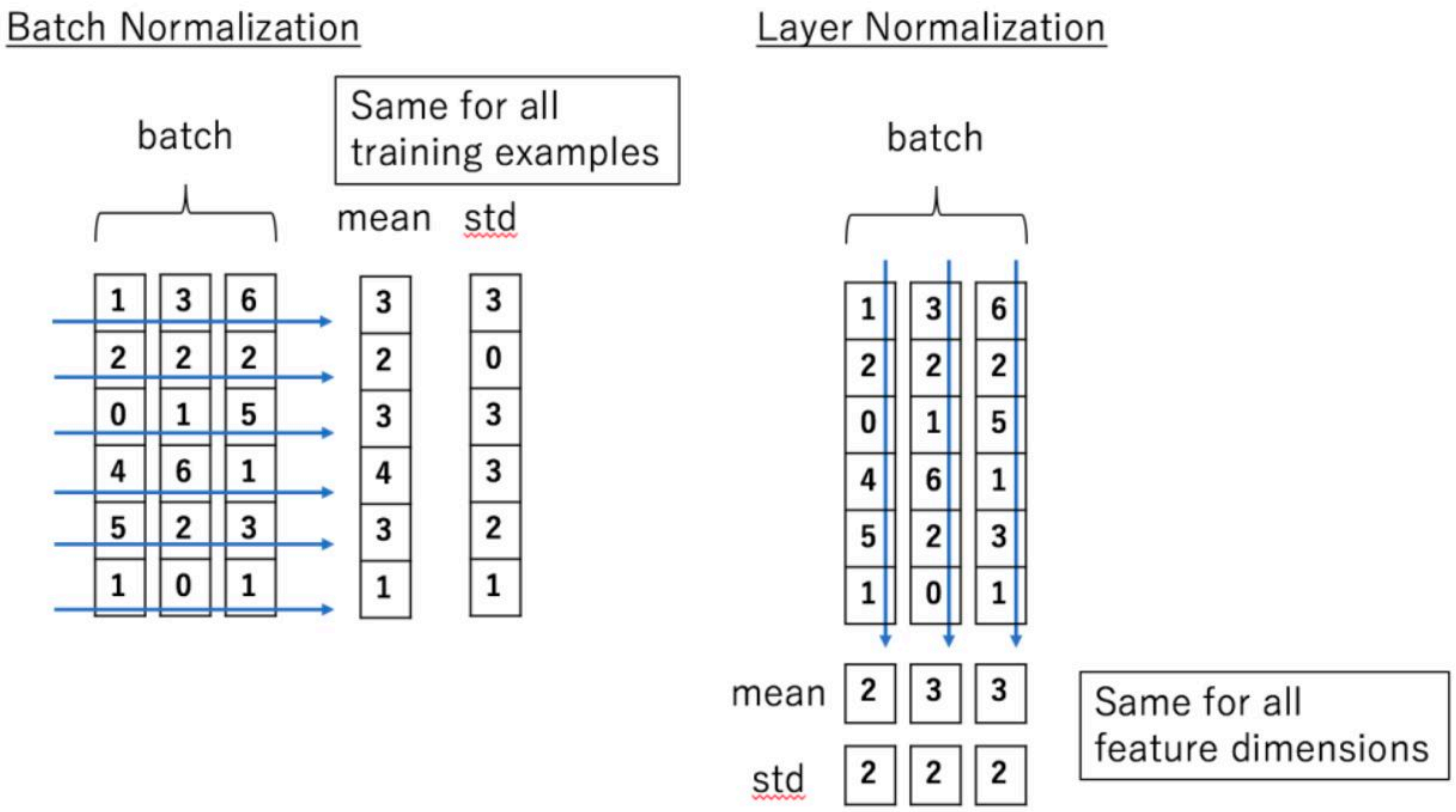
- Augment word embedding with position embeddings, each dim is a sine/cosine wave of a different frequency. Closer points = higher dot products
- Works essentially as well as just encoding position as a one-hot vector

Vaswani et al. (2017)

Layer Normalization



- subtract mean, divide by variance



Ba et al. (2016)

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

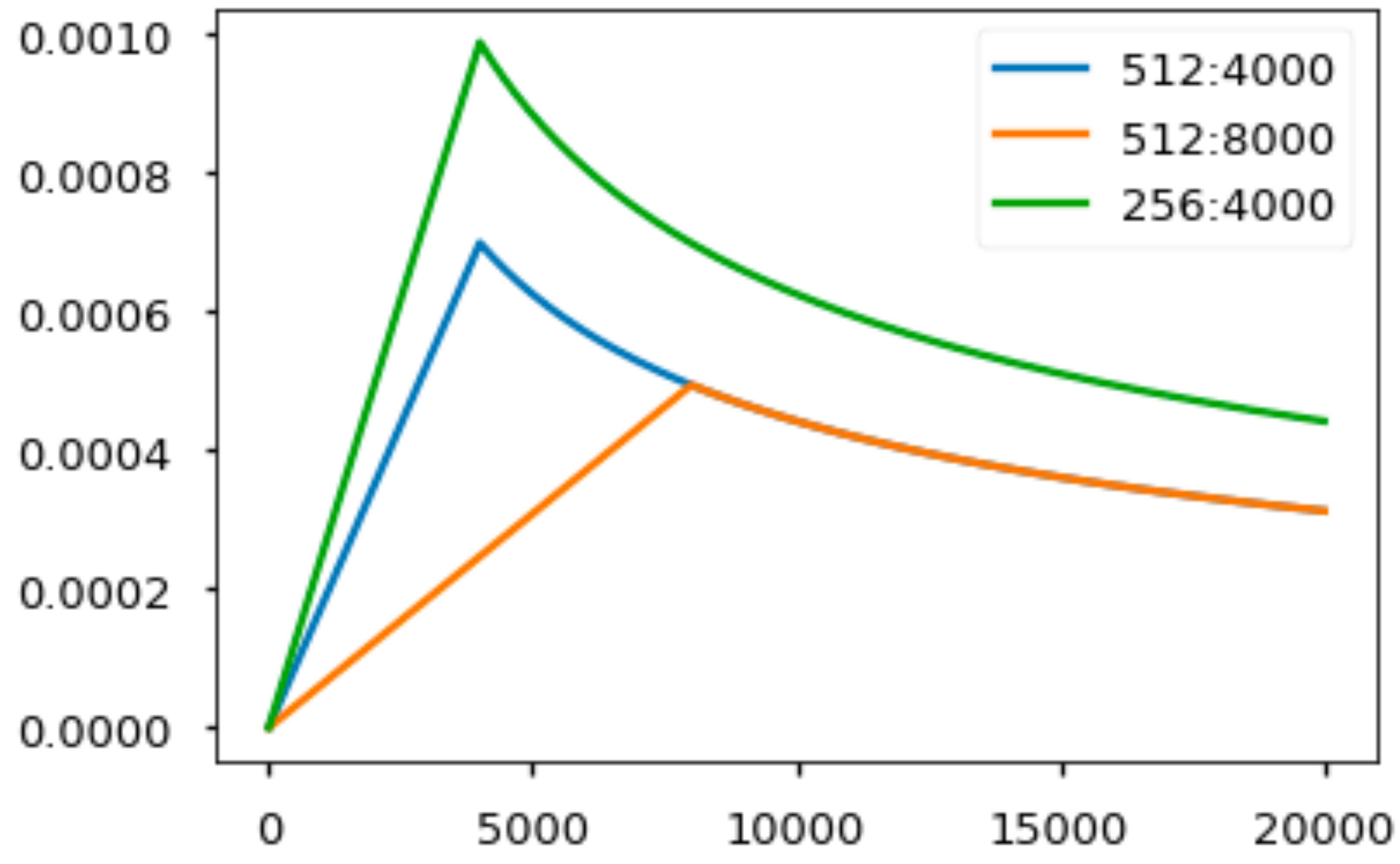
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

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)

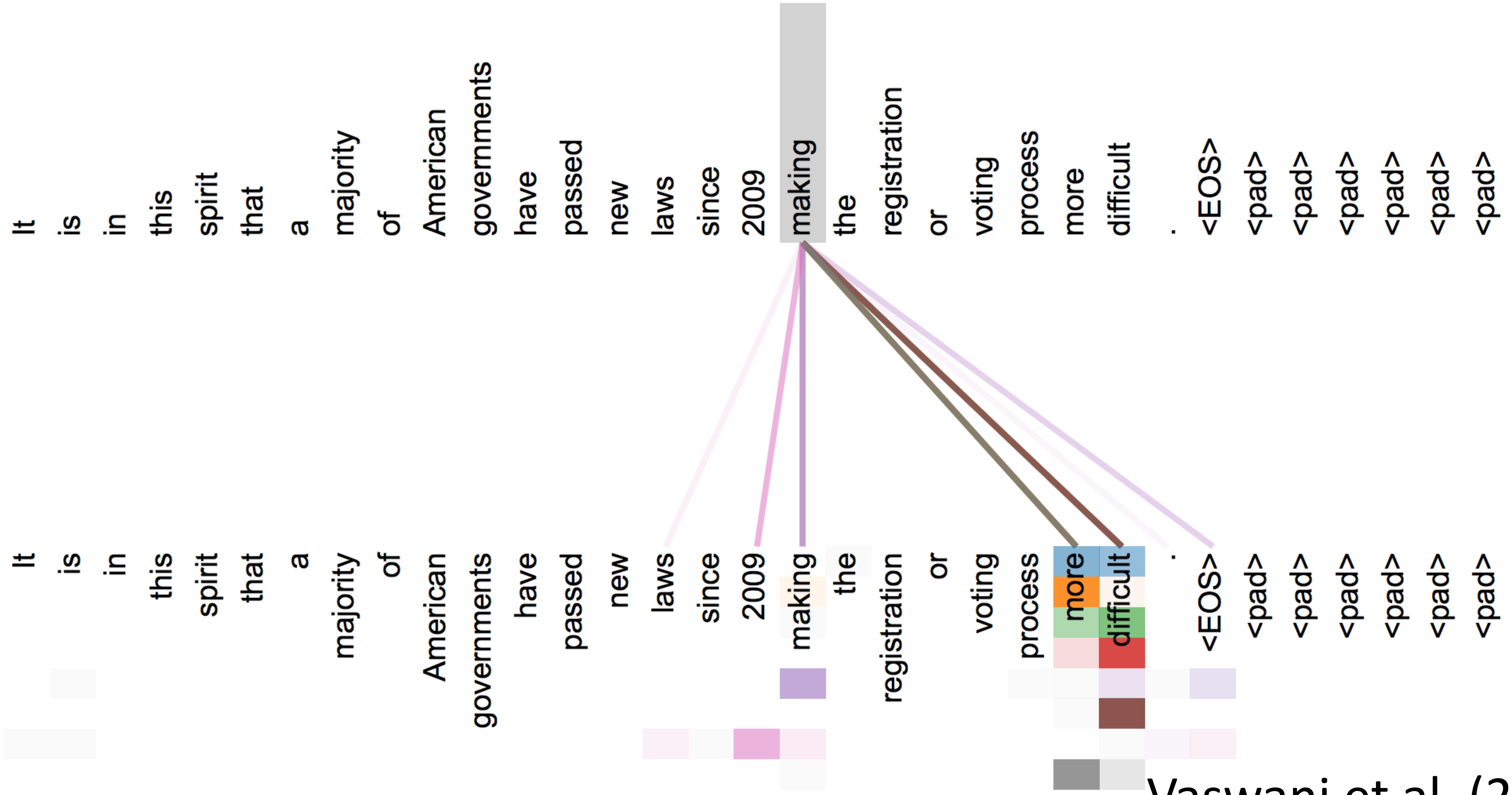
Transformers

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

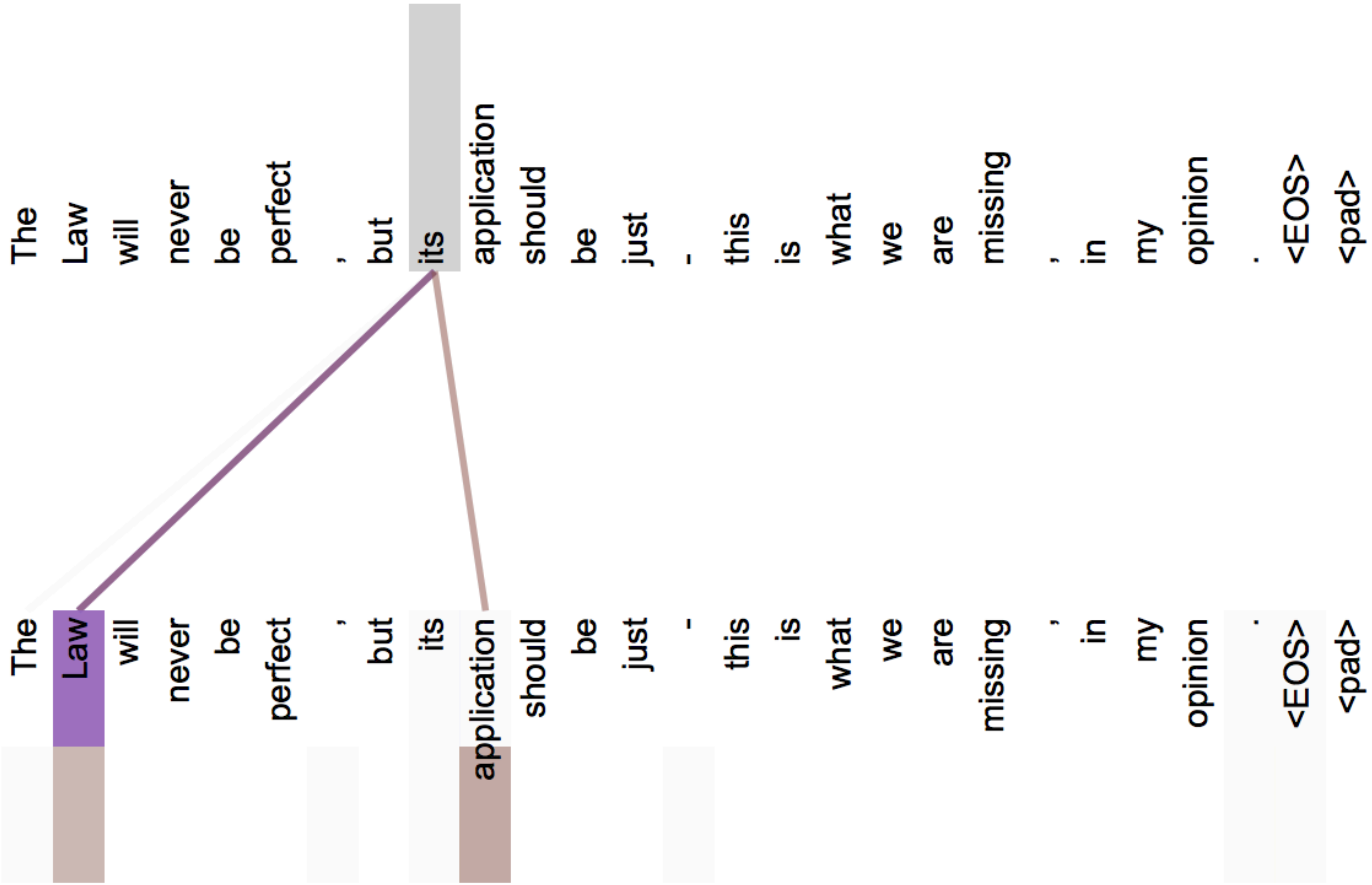
Vaswani et al. (2017)

Visualization



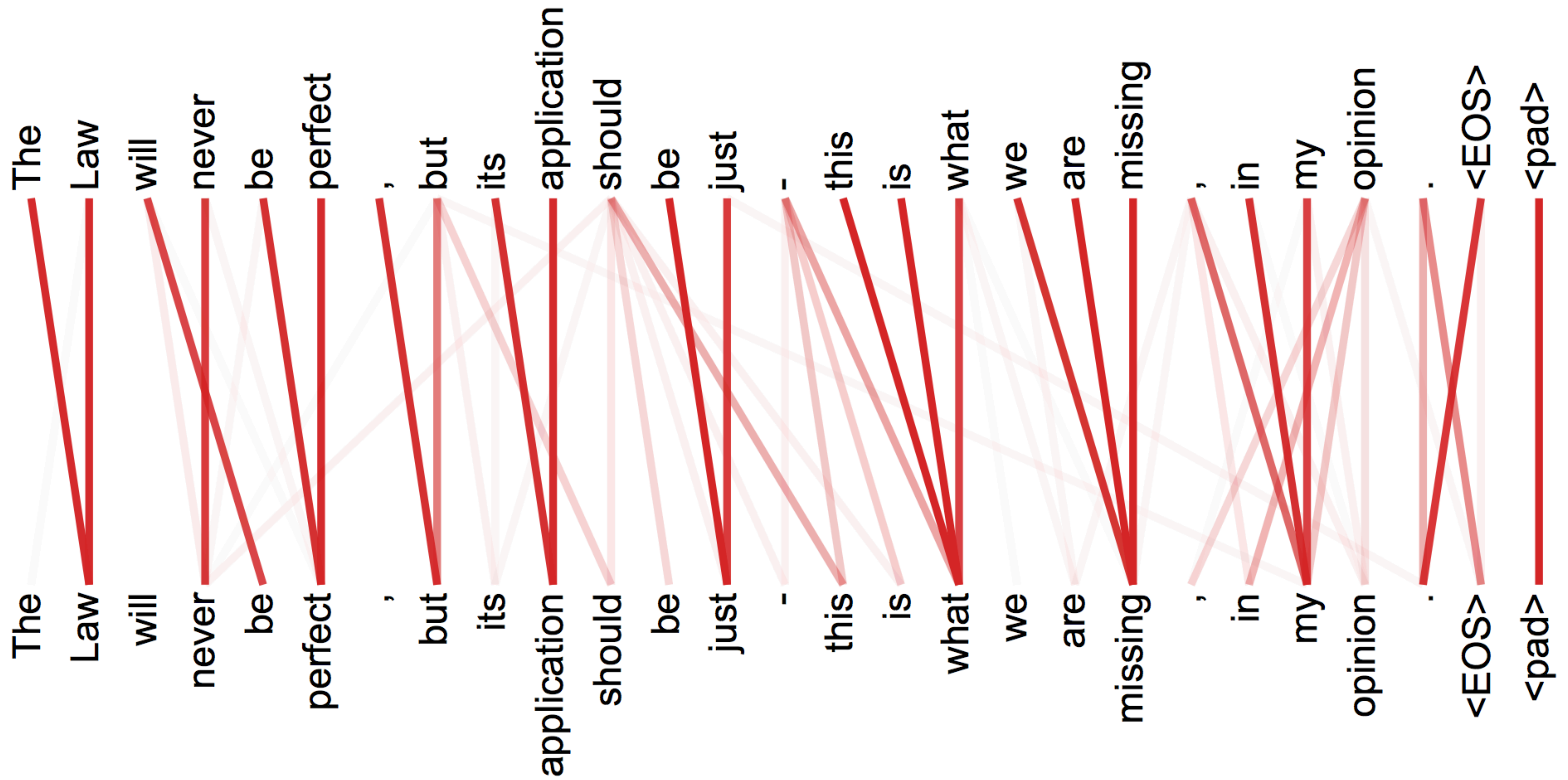
Vaswani et al. (2017)

Visualization



Vaswani et al. (2017)

Visualization



Vaswani et al. (2017)

Useful Resources

nn.Transformer:

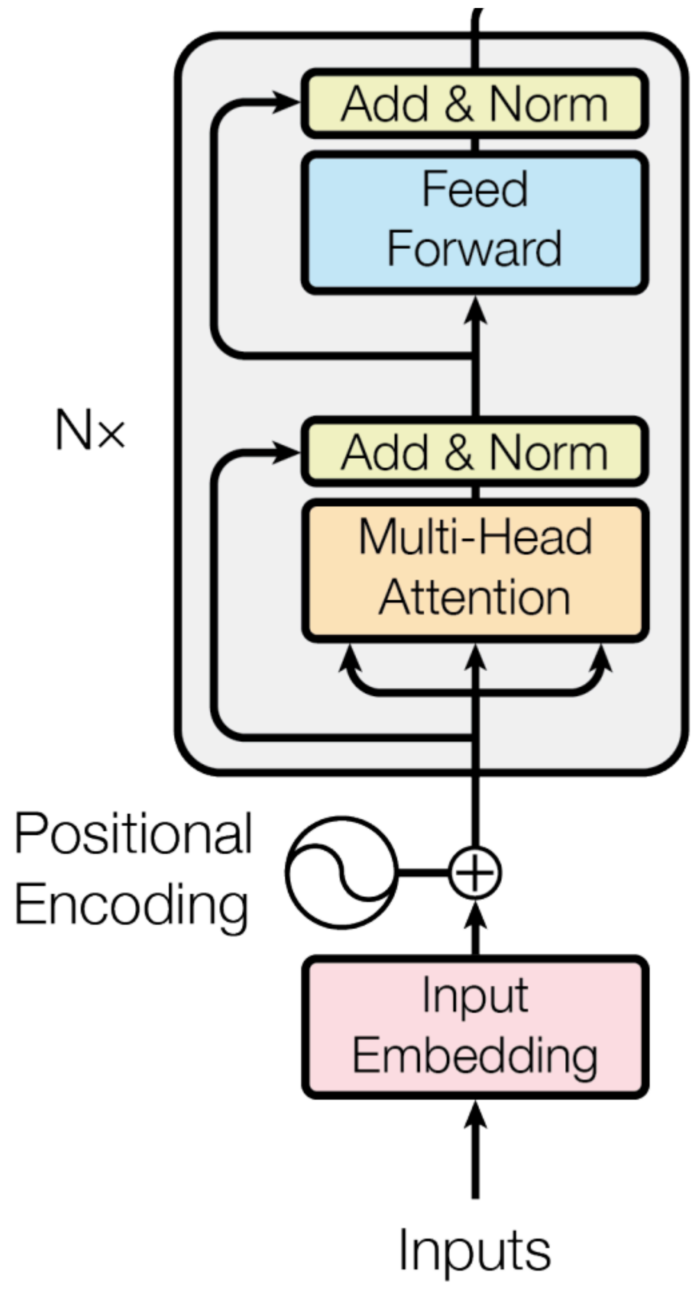
```
>>> transformer_model = nn.Transformer(nhead=16, num_encoder_layers=12)
>>> src = torch.rand((10, 32, 512))
>>> tgt = torch.rand((20, 32, 512))
>>> out = transformer_model(src, tgt)
```

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?



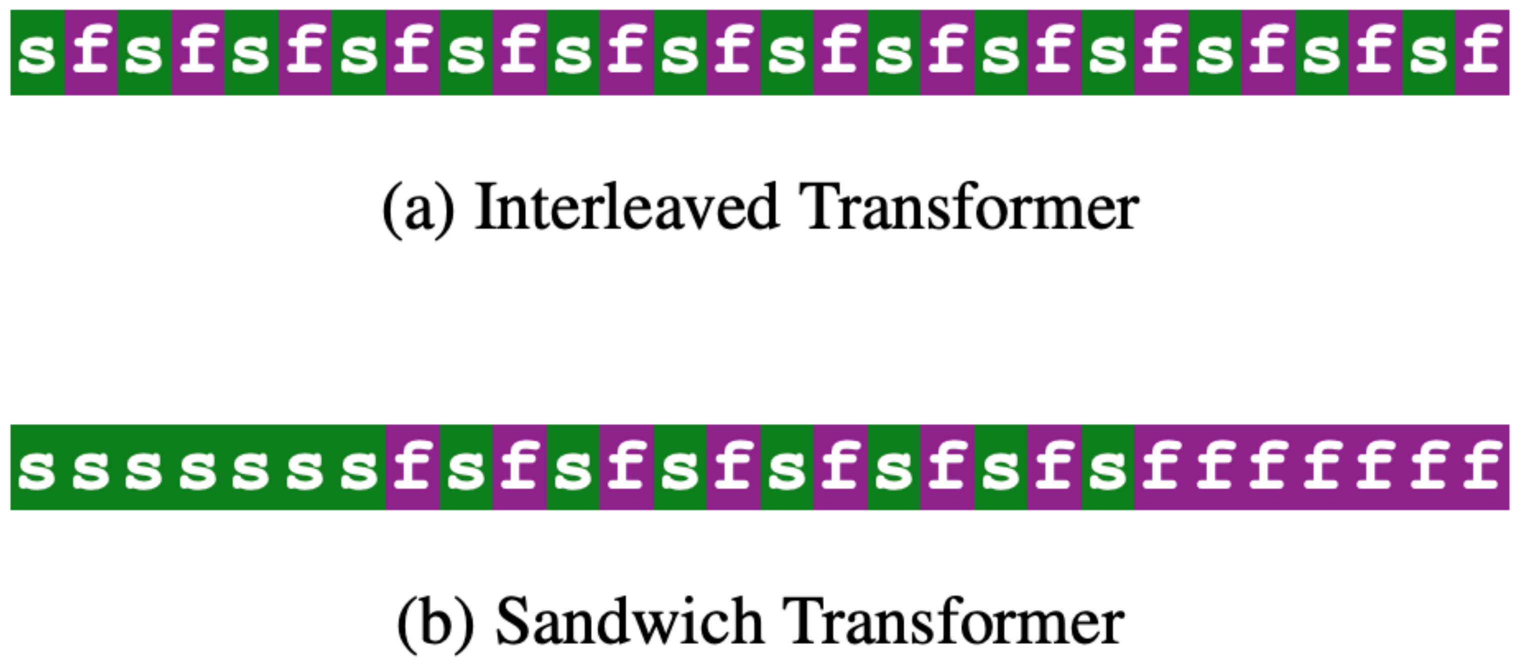
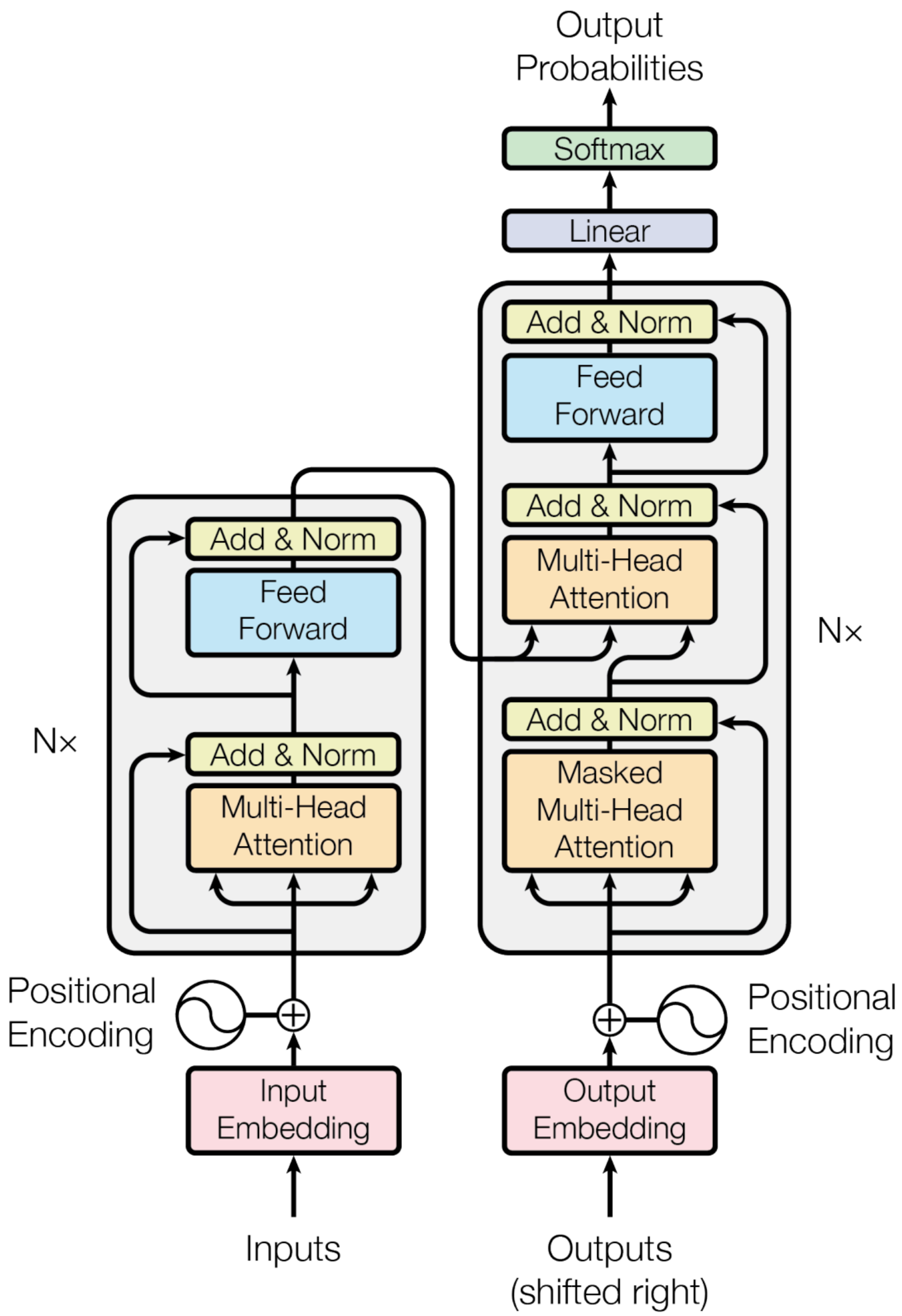


Figure 1: A transformer model (a) is composed of interleaved self-attention (green) and feedforward (purple) sublayers. Our sandwich transformer (b), a reordering of the transformer sublayers, performs better on language modeling. Input flows from left to right.

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)