SAND2019-5079PE

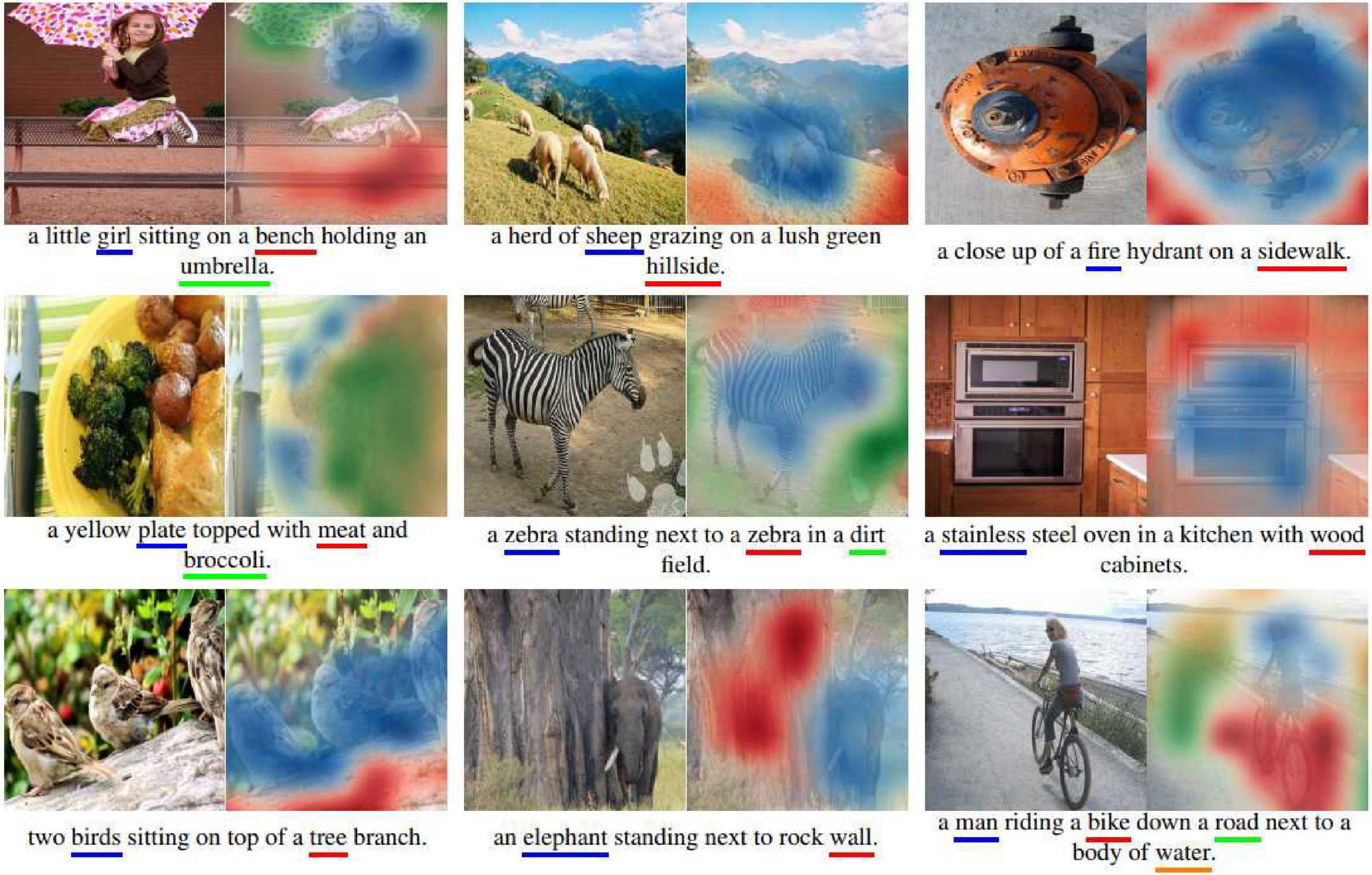
Transformers for Neural Machine Translation and Beyond

PRESENTED BY

David Kavaler

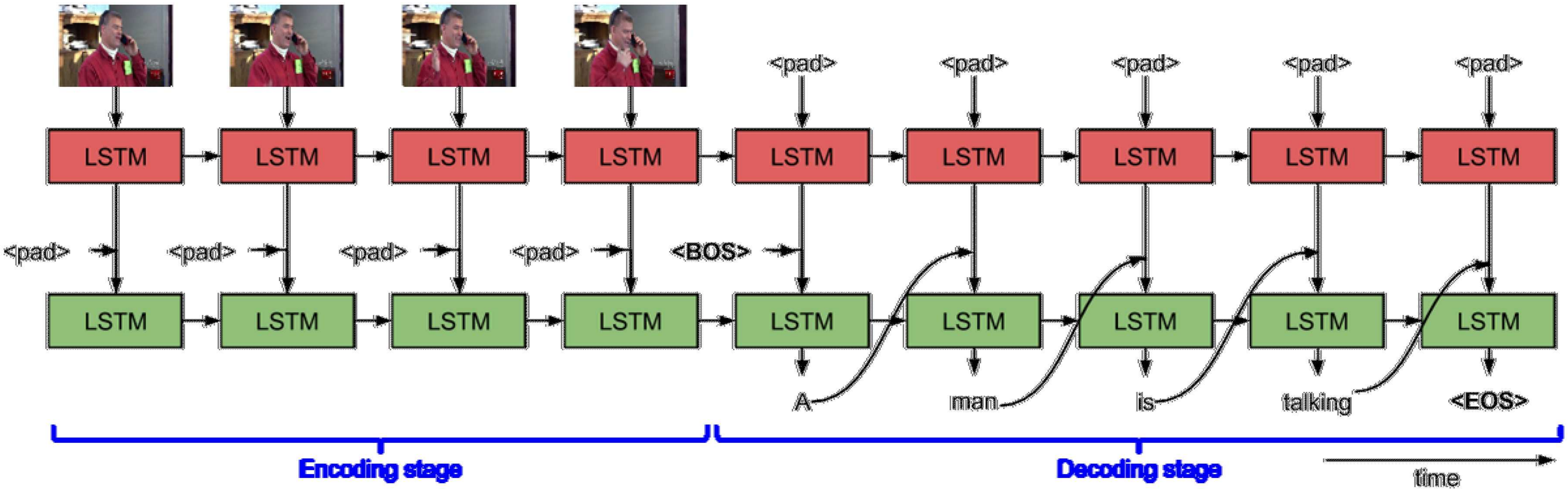
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sequence to Sequence (Encoder-Decoder) Models



Lu et al. (2017)

Sequence to Sequence (Encoder-Decoder) Models



https://vsubhashini.github.io/s2vt.html

Sequence to Sequence (Encoder-Decoder) Models

PROMPT:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

https://openai.com/blog/better-language-models/#sample1

Sequence to Sequence (Encoder-Decoder) Models

PROMPT:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION:

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

https://openai.com/blog/better-language-models/#sample1

Sequence to Sequence (Encoder-Decoder) Models

PROMPT:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION:

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

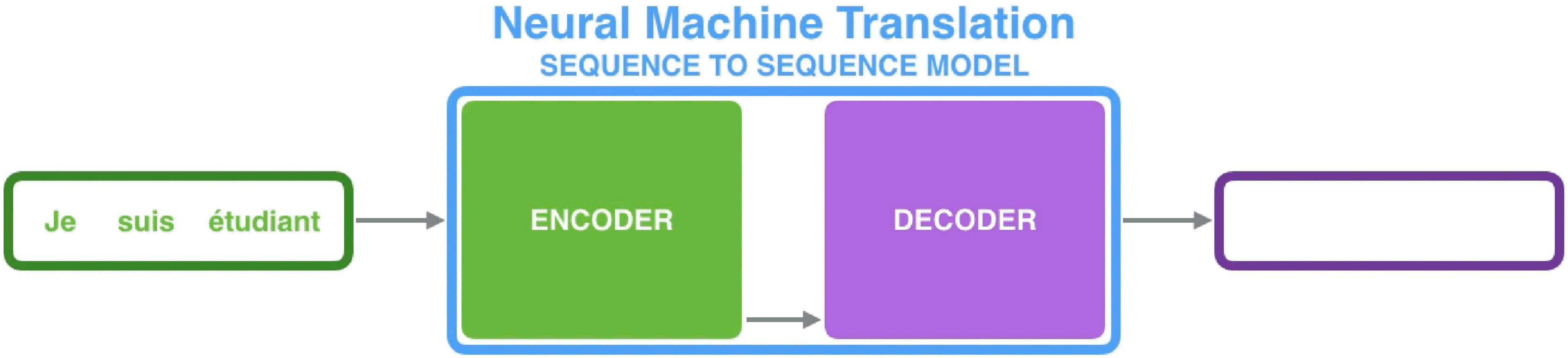
…

https://openai.com/blog/better-language-models/#sample1

Sequence to Sequence (Encoder-Decoder) Models

Deep learning models that take a sequence of items and outputs another sequence of items

- Generic input-output (sequence-sequence) format
- Successful across disciplines (machine translation, image captioning, etc)

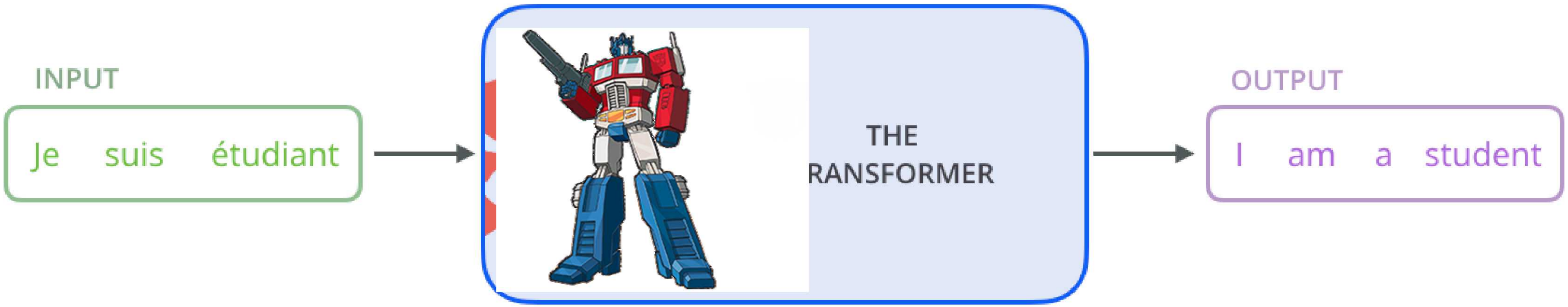


https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Sequence to Sequence (Encoder-Decoder) Models

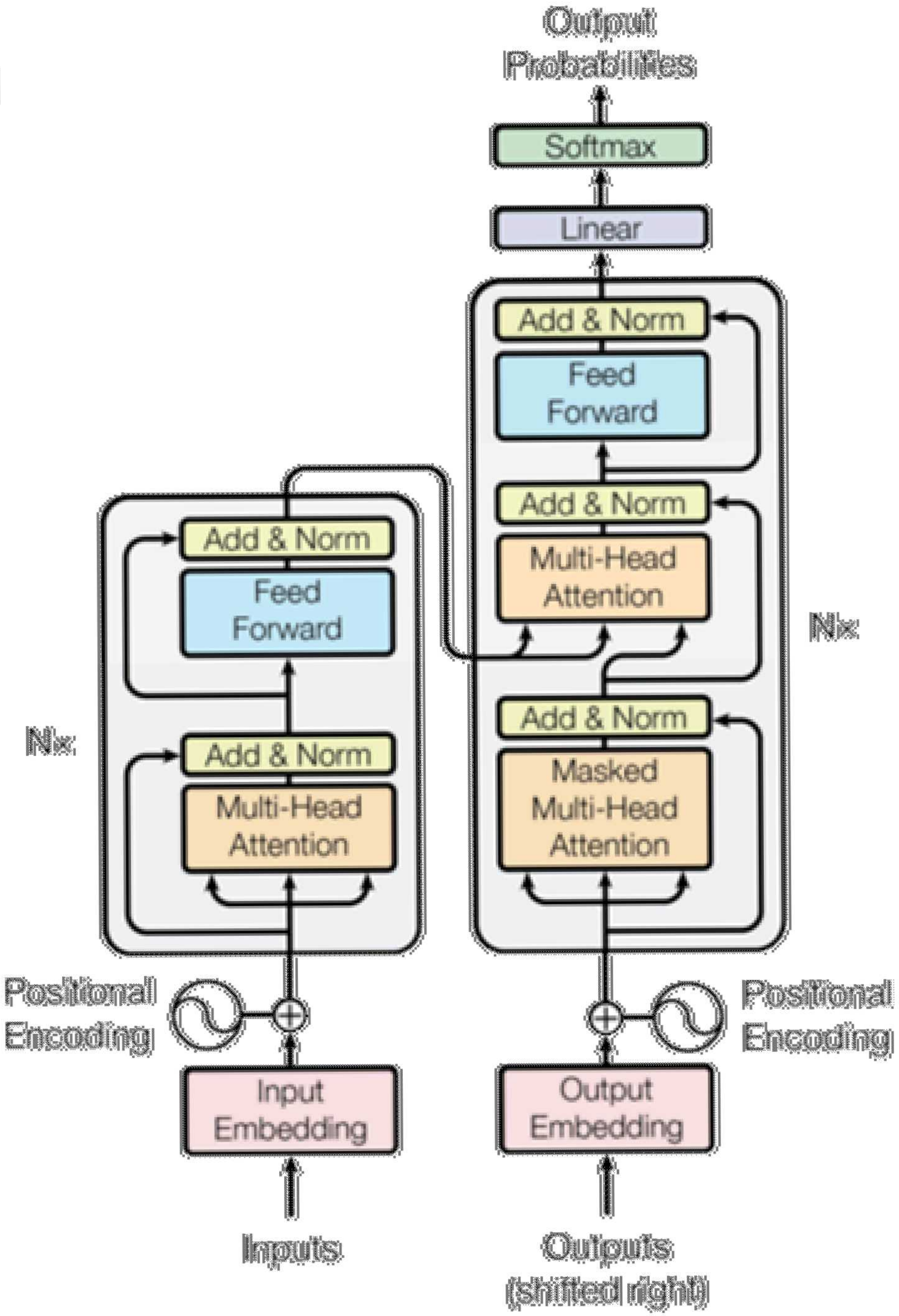
Deep learning models that take a sequence of items and outputs another sequence of items

- Generic input-output (sequence-sequence) format
- Successful across disciplines (machine translation, image captioning, etc)

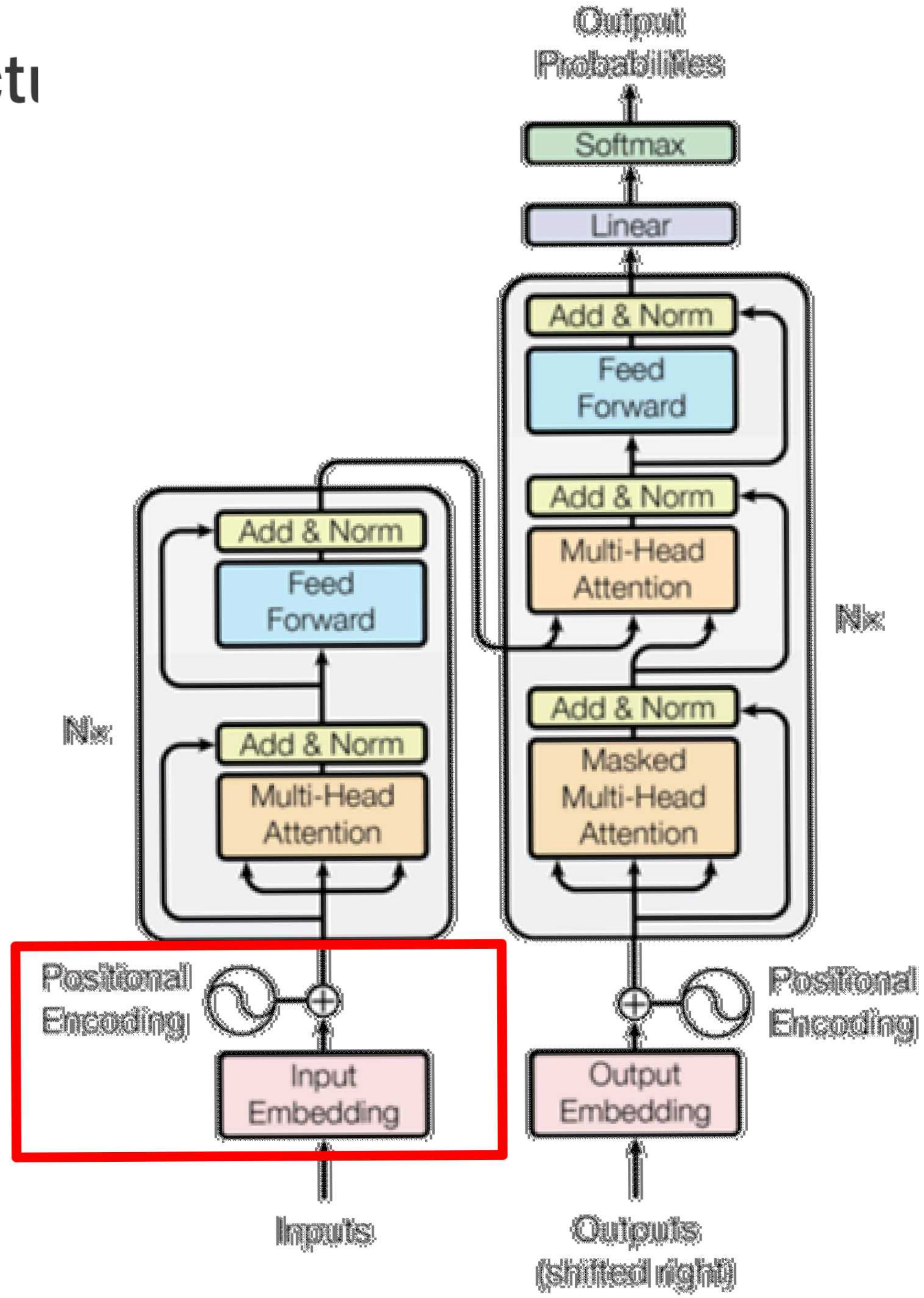


https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

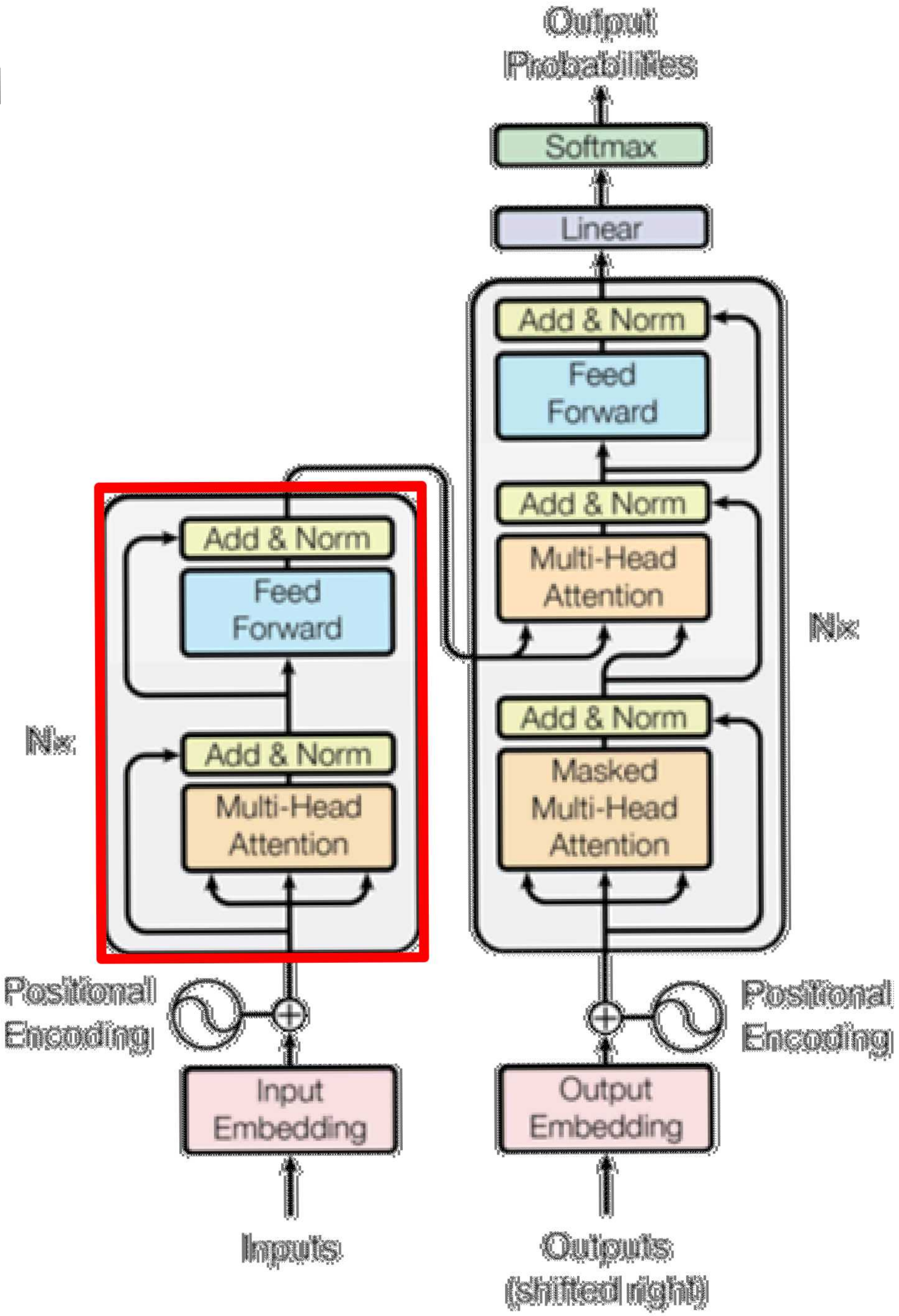
Transformer Architectu



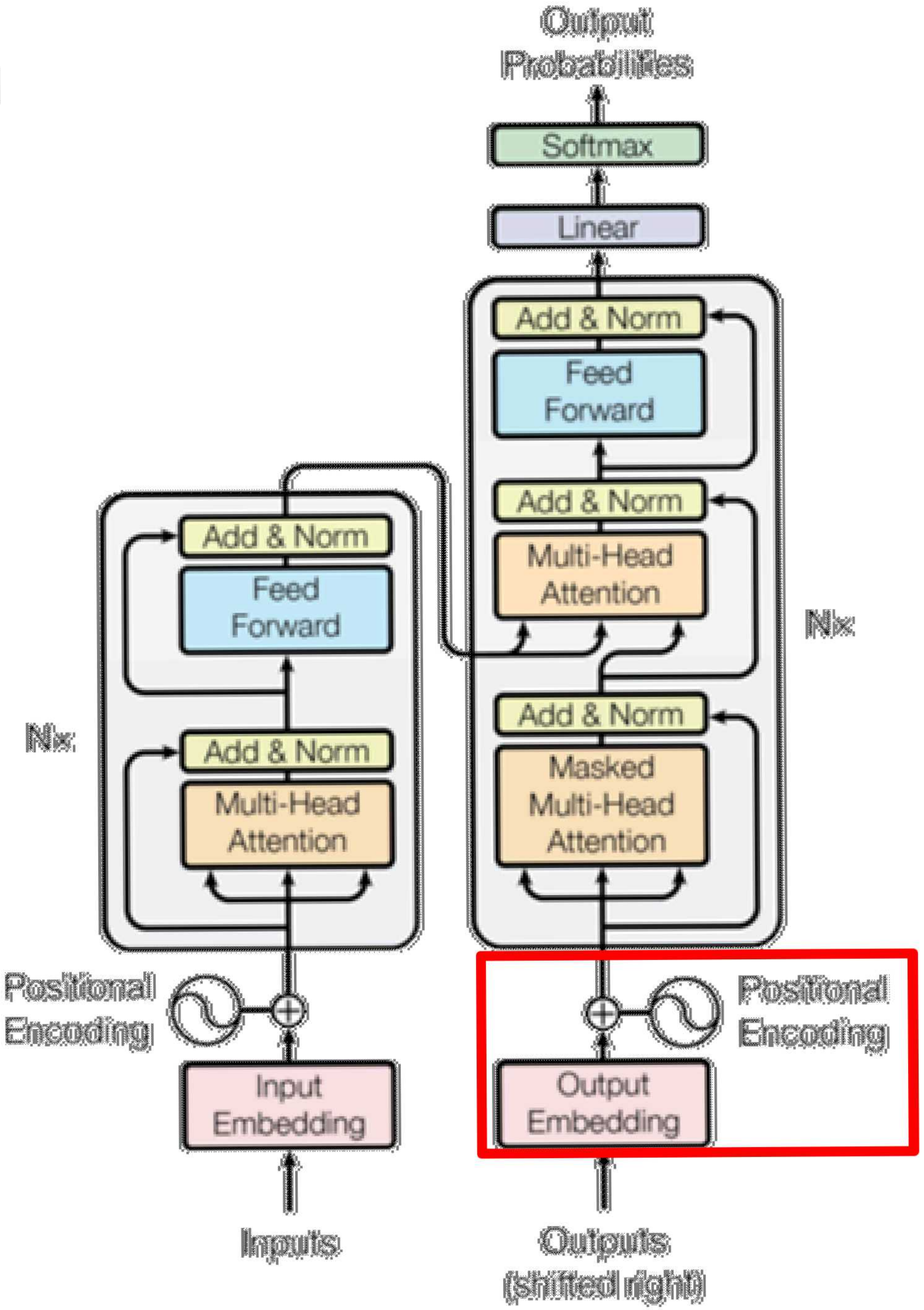
Transformer Architectu



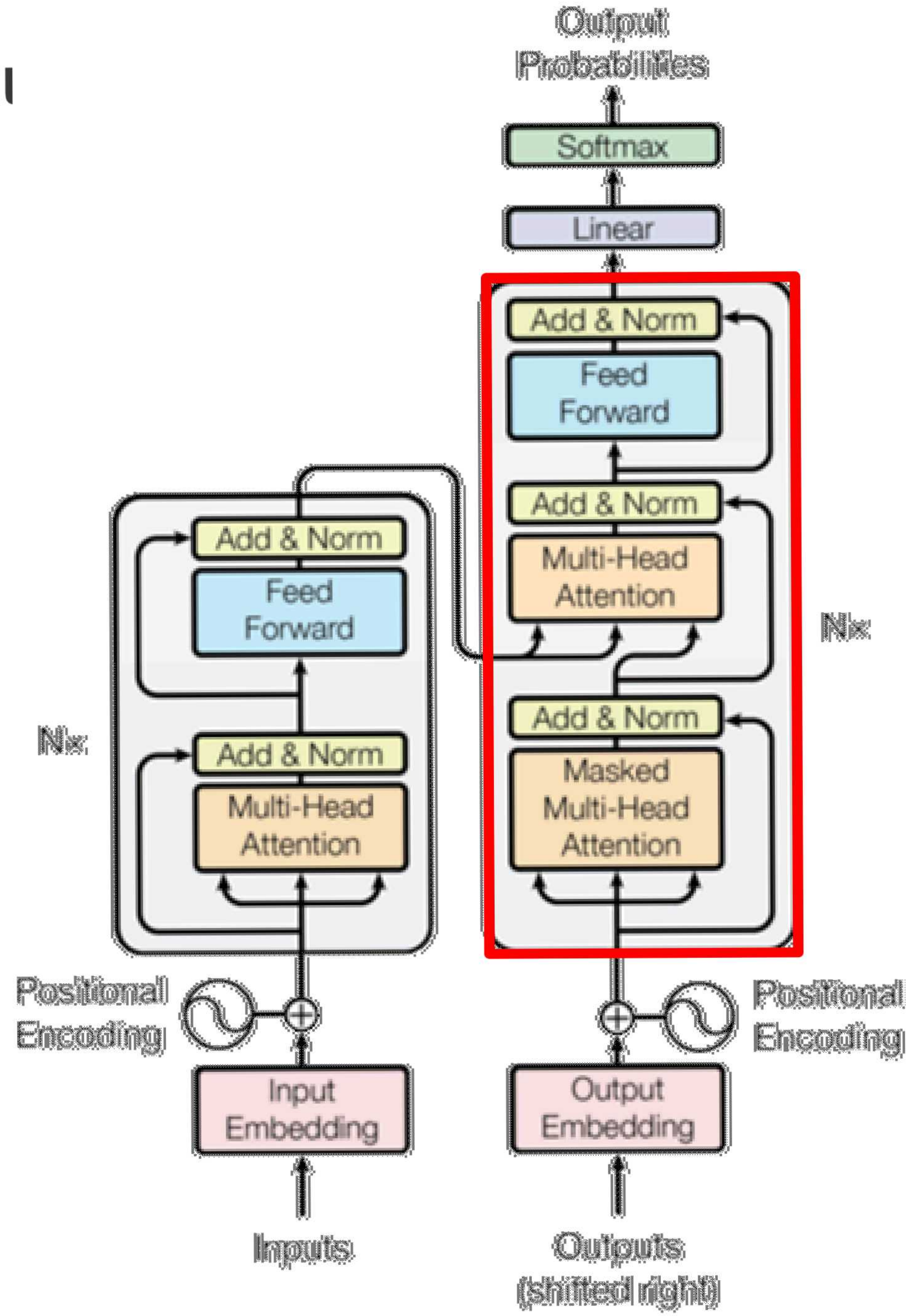
Transformer Architecti



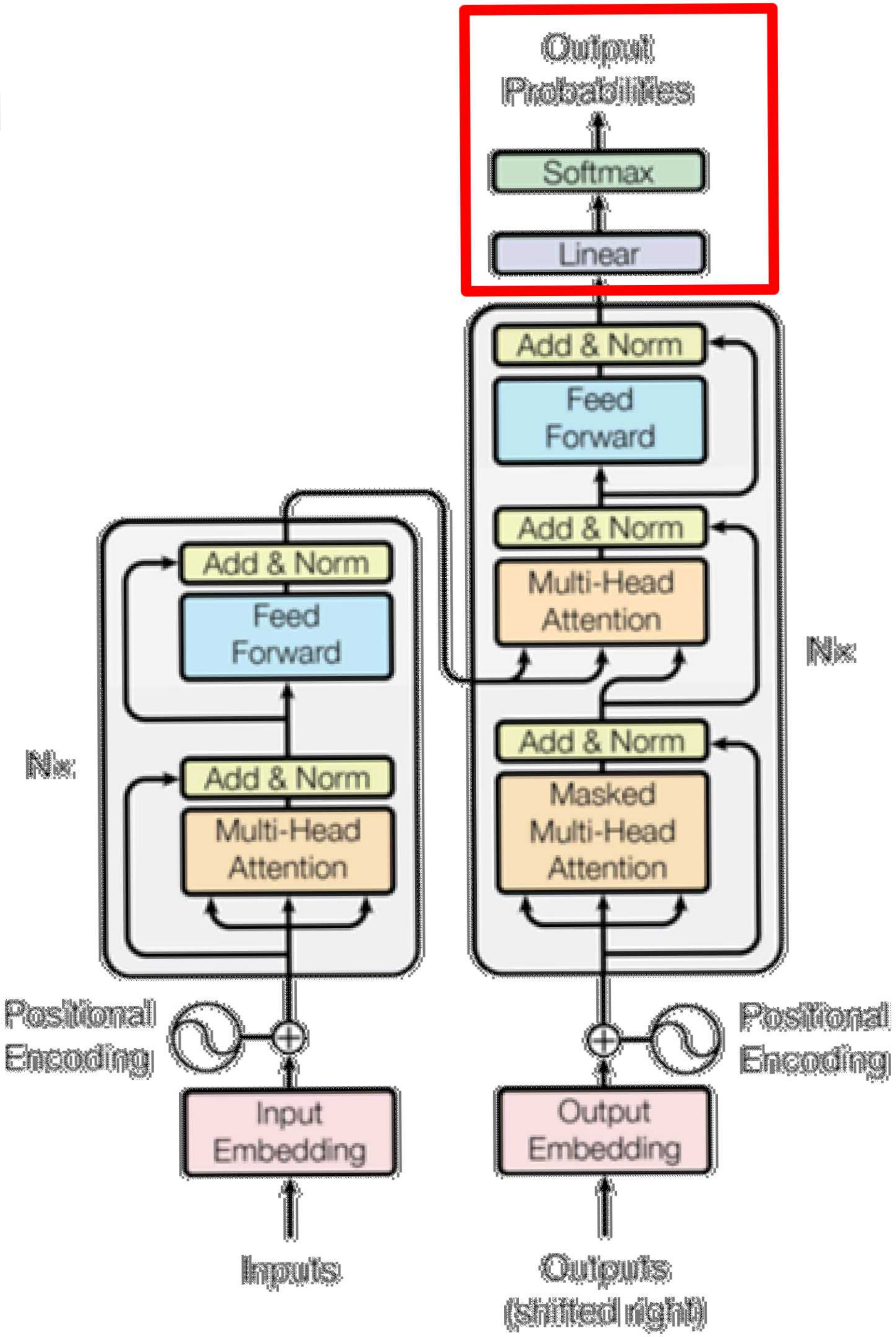
Transformer Architect



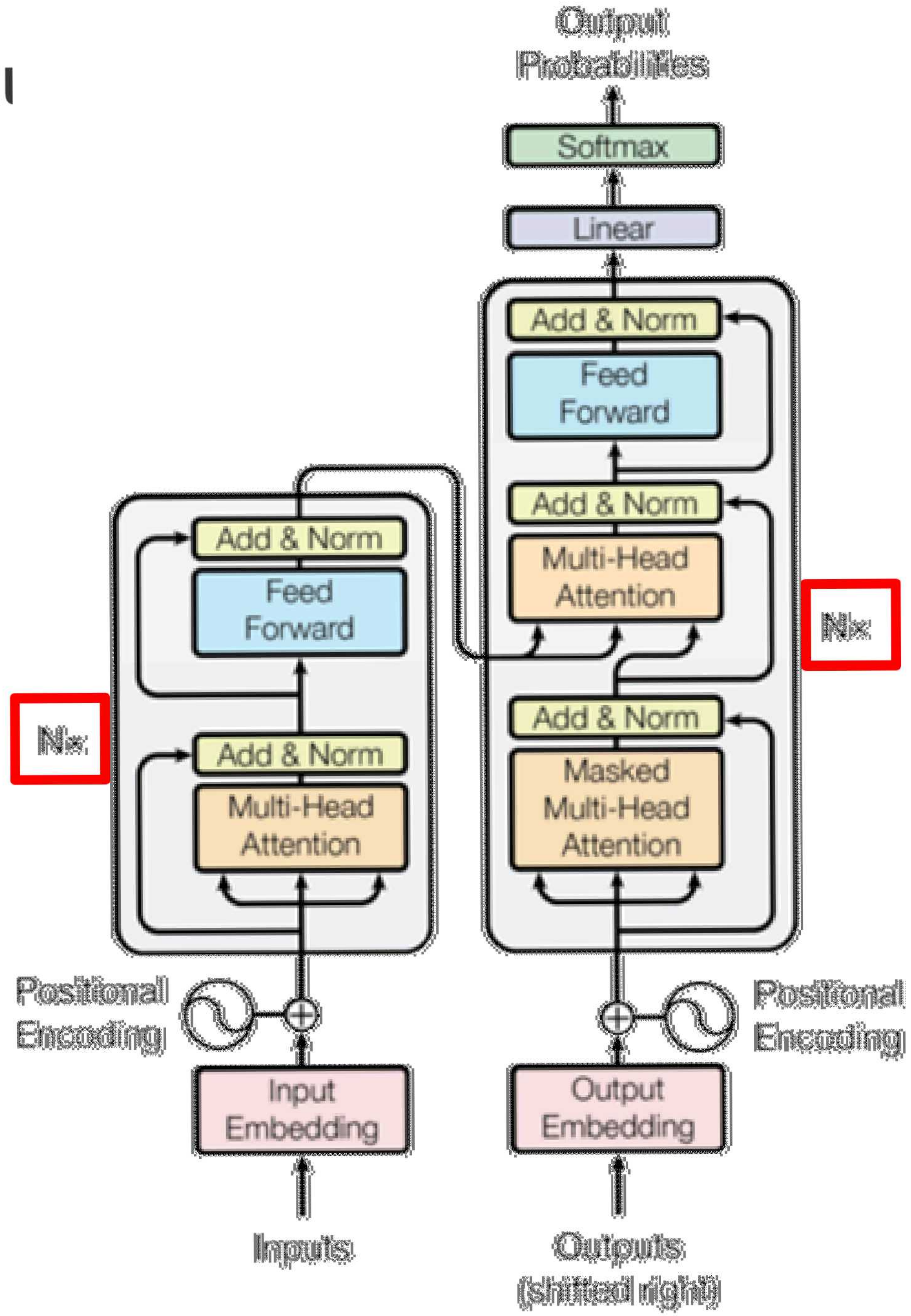
Transformer Architectu



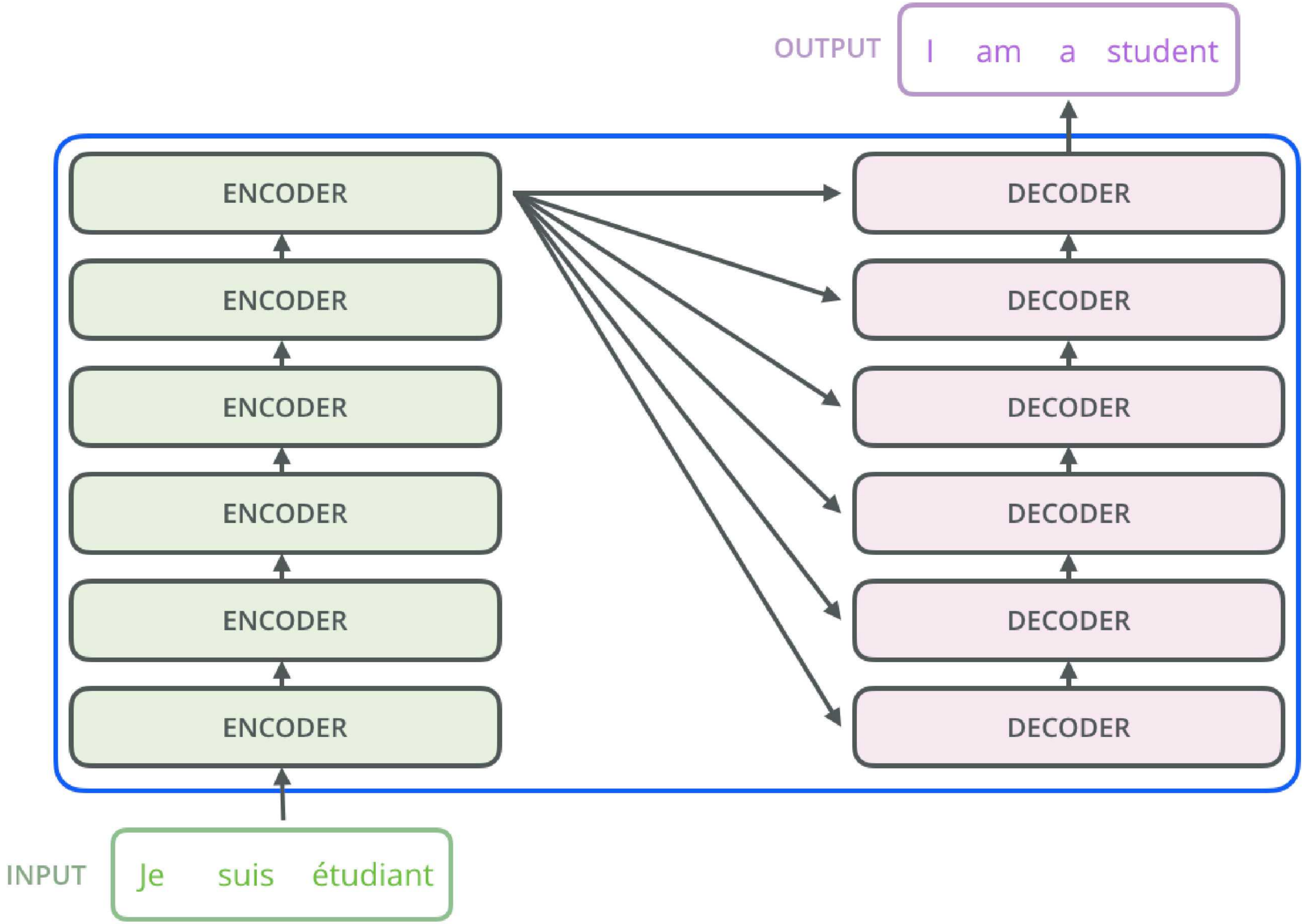
Transformer Architectu



Transformer Architectu



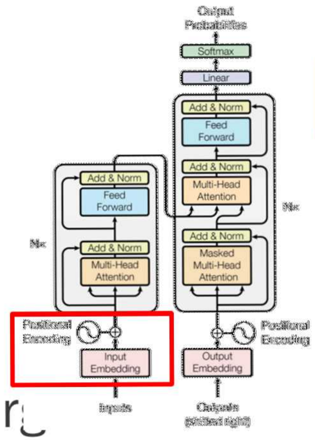
Transformer Architecture Overview



Transformer Complexity

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

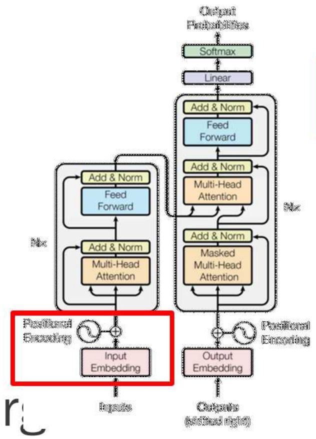
Embedding and Positional Encoding



Standard embedding lookup (weight matrix vocabulary index)

- Embedding weights shared between encoder input, decoder teacher forcing, and targ sequence

Embedding and Positional Encoding



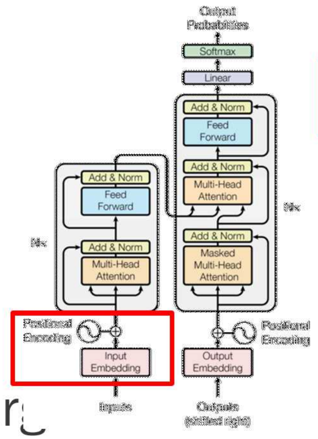
Standard embedding lookup (weight matrix vocabulary index)

- Embedding weights shared between encoder input, decoder teacher forcing, and targ sequence

Positional encoding

- RNNs have position encoded in architecture – forward feeding hidden state
- Want to encode information about word position in input sequence
- Want to handle variable length inputs, but be consistent across all sequences
- Want to be able to predict on sequences with length longer than anything seen in training

Embedding and Positional Encoding



Standard embedding lookup (weight matrix vocabulary index)

- Embedding weights shared between encoder input, decoder teacher forcing, and tar sequence

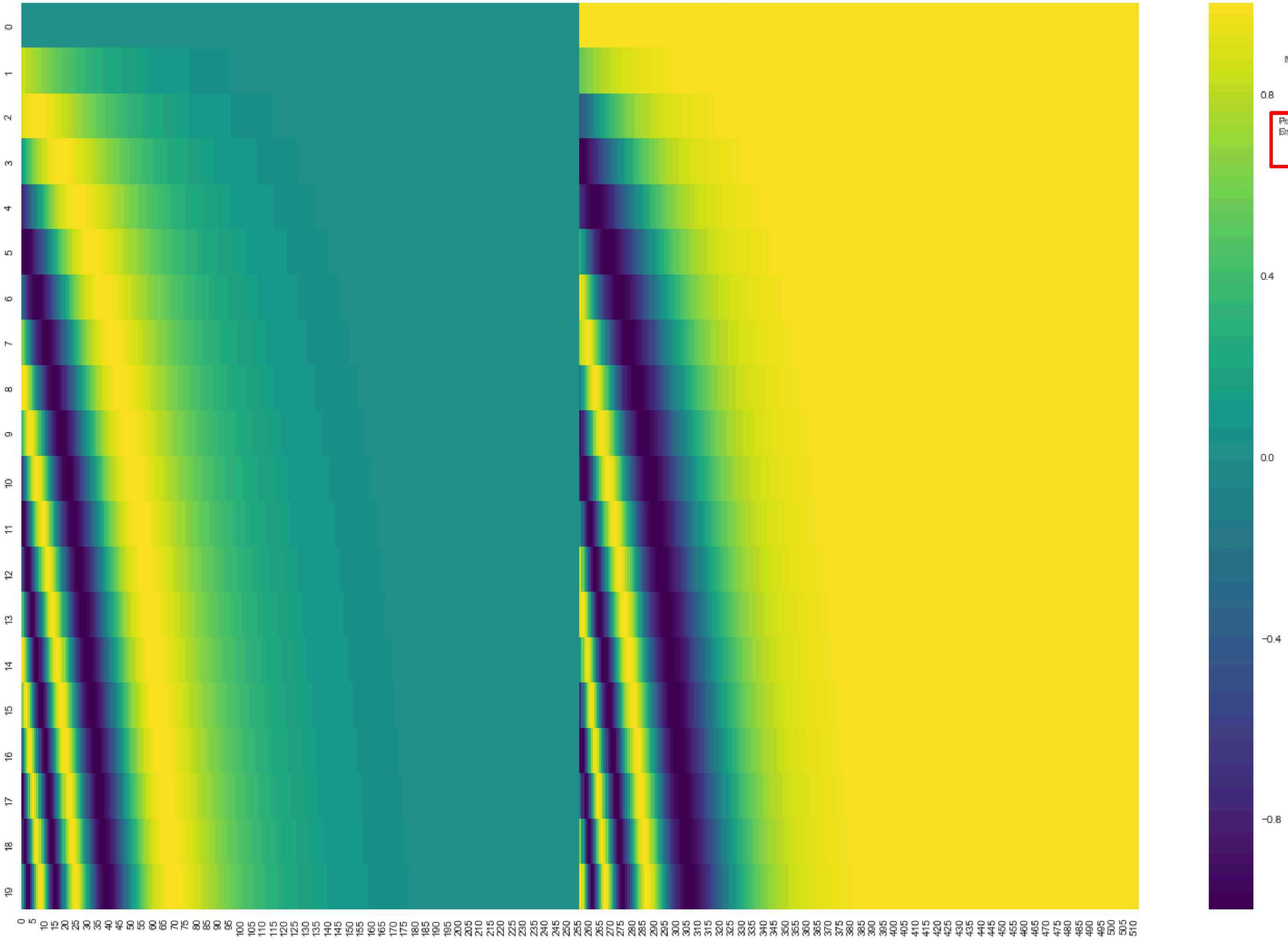
Positional encoding

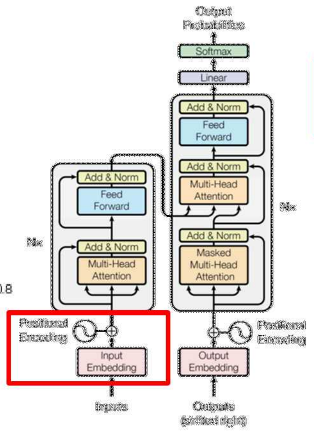
- RNNs have position encoded in architecture – forward feeding hidden state
- Want to encode information about word position in input sequence
- Want to handle variable length inputs, but be consistent across all sequences
- Want to be able to predict on sequences with length longer than anything seen in training

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

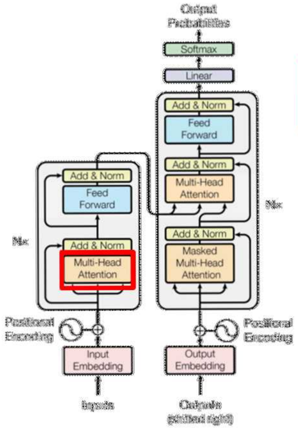
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Embedding and Positional Encoding





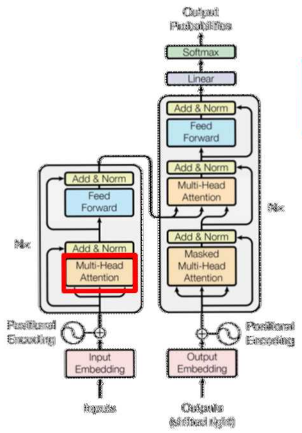
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

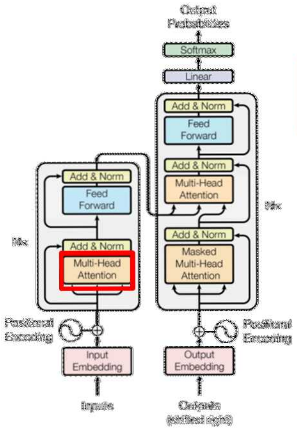
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

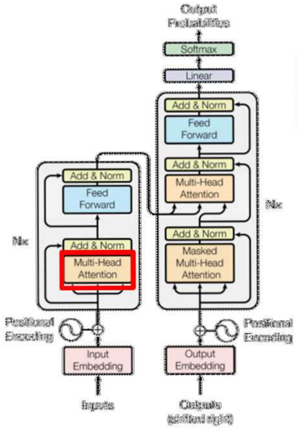
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

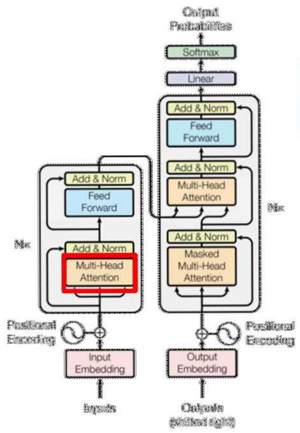
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

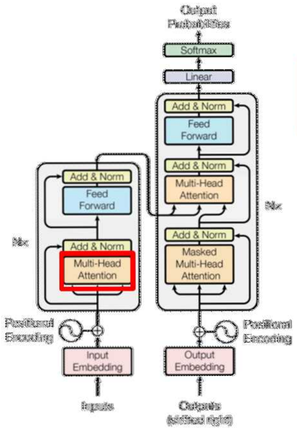
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

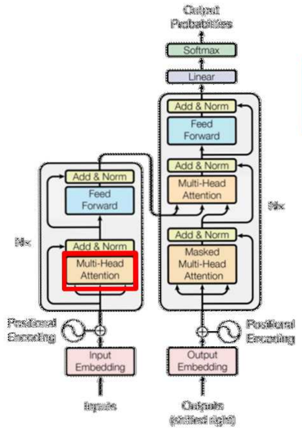
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

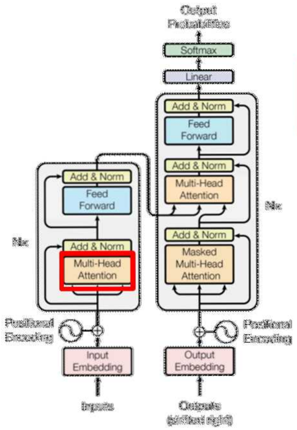
Self-Attention



Core concept: words in a sentence are related to each other in a complex manner

The animal didn't cross the street because it was too tired

Self-Attention

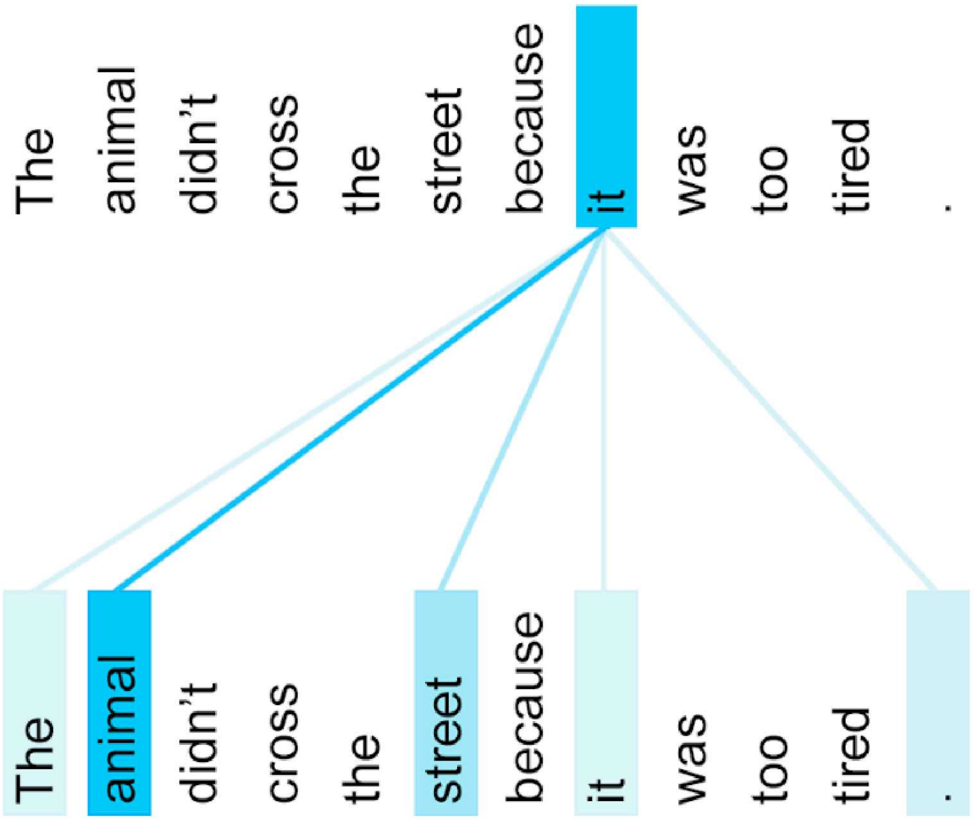


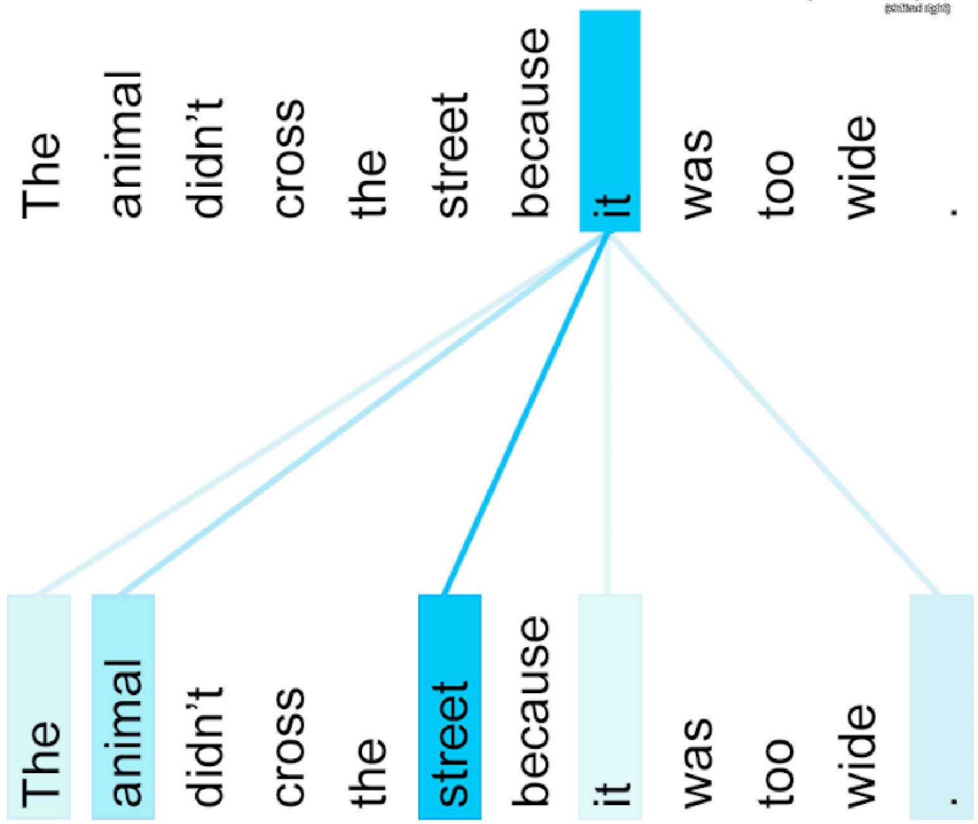
Core concept: words in a sentence are related to each other in a complex manner

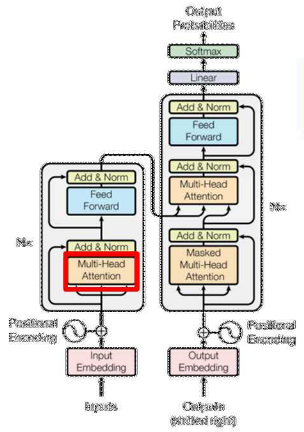
The animal didn't cross the street because it was too tired

Self-attention is used to represent these complex dependencies

Self-Attention







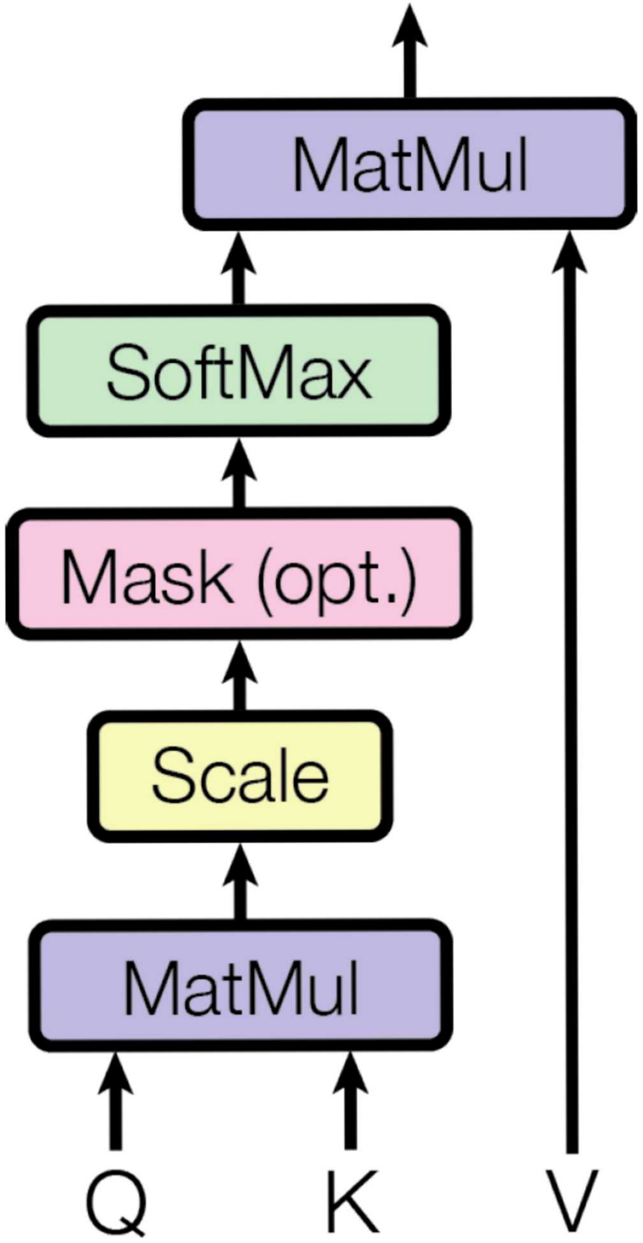
Self-Attention

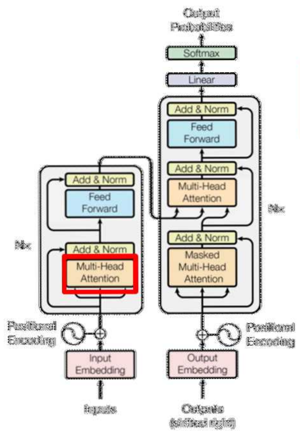
Intuition:

Query: current token

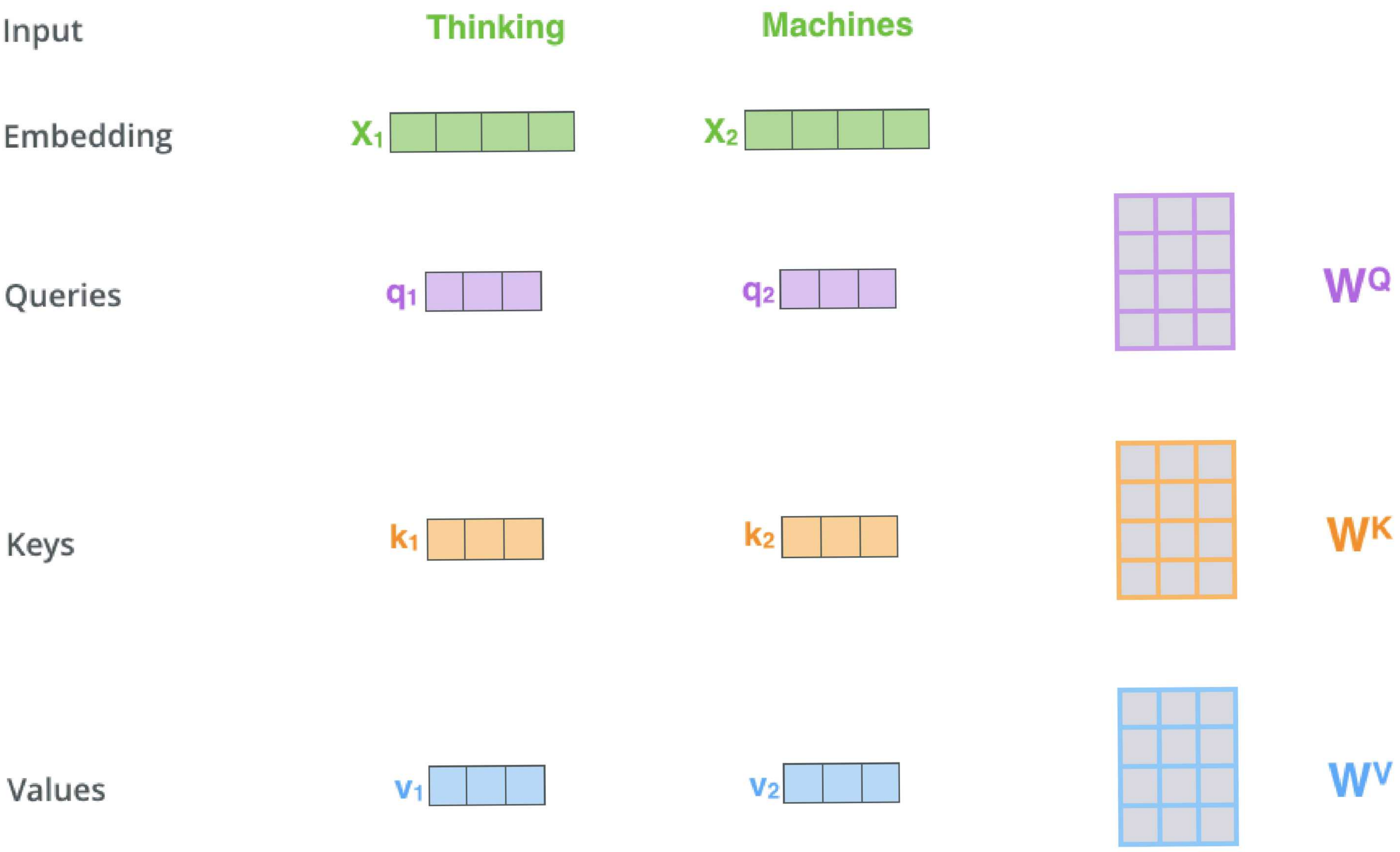
Key: tokens to compare with (all tokens in input sequence)

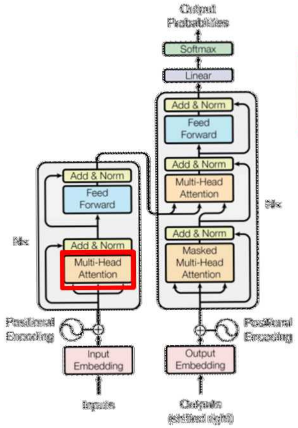
Value: output (to be scaled by softmax of Q, K operation)



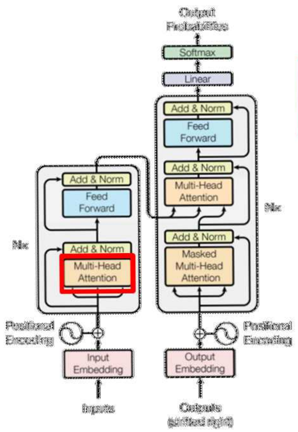


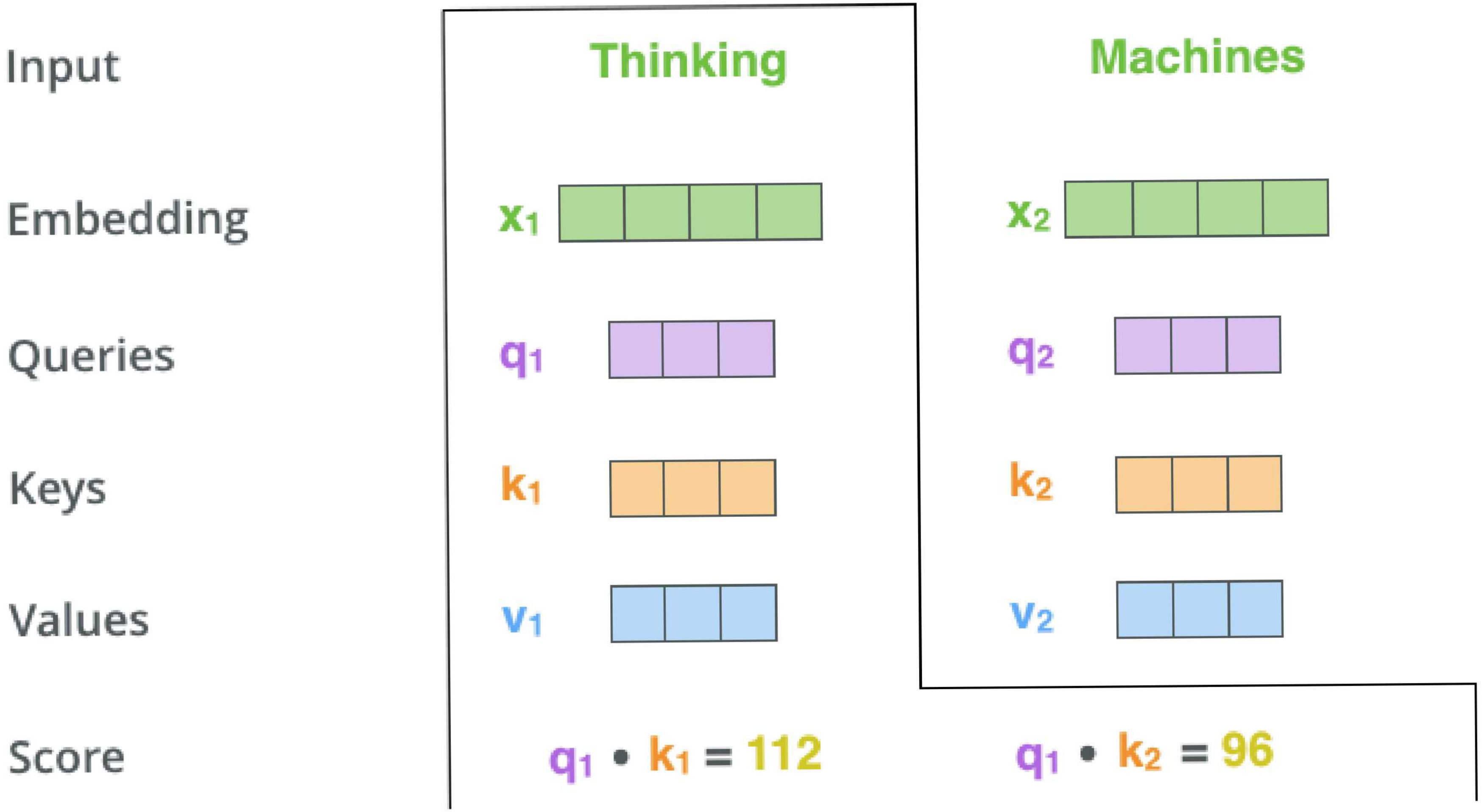
Self-Attention: Intuition



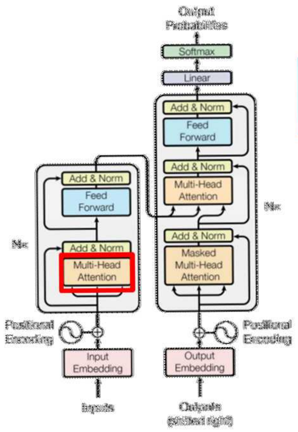


Self-Attention: Intuition



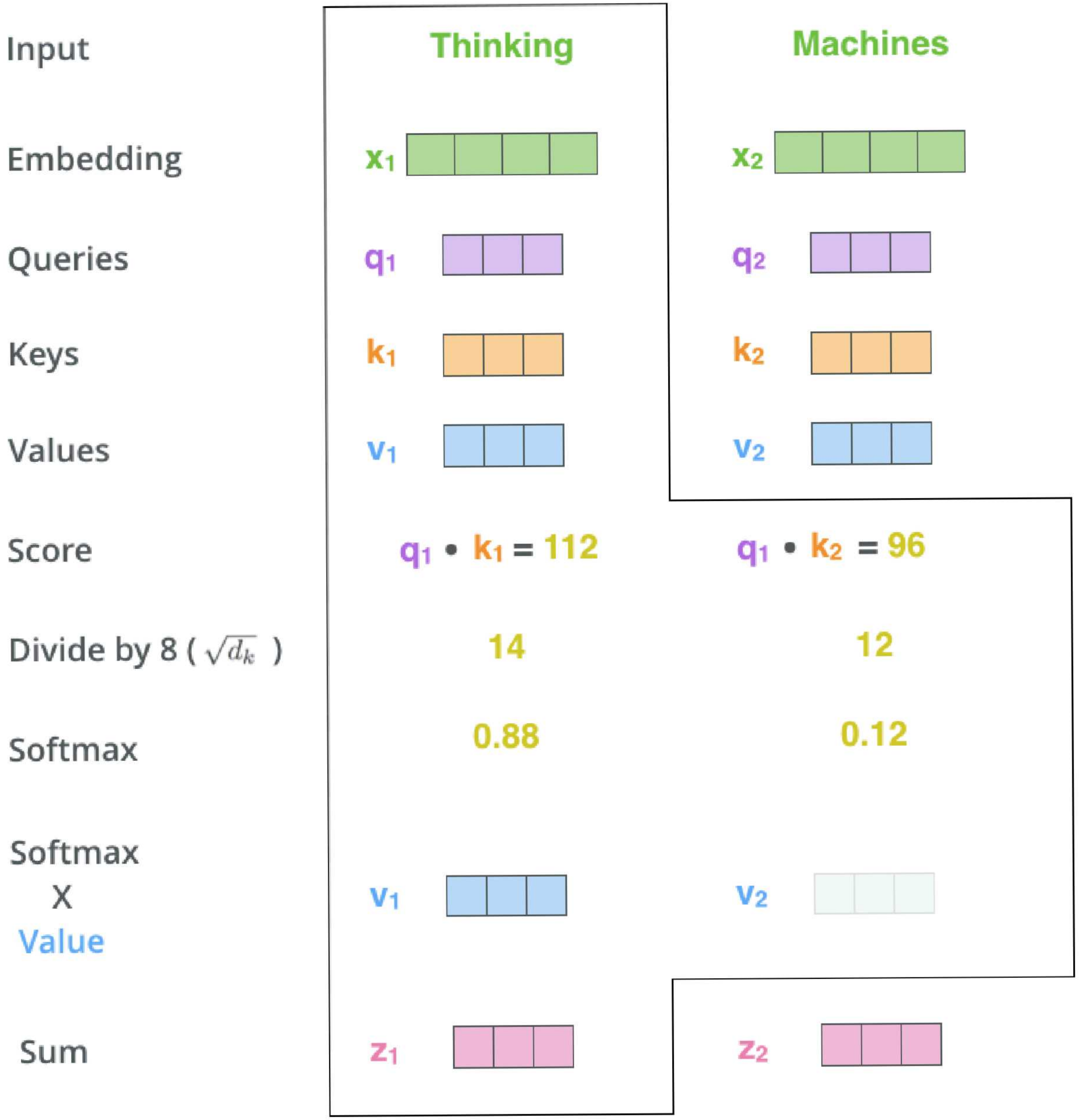


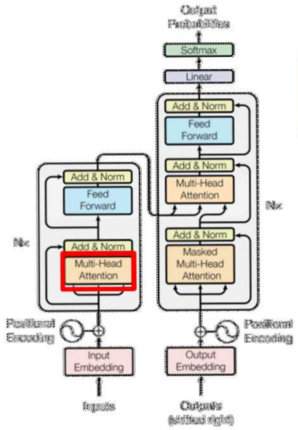
Self-Attention: Intuition



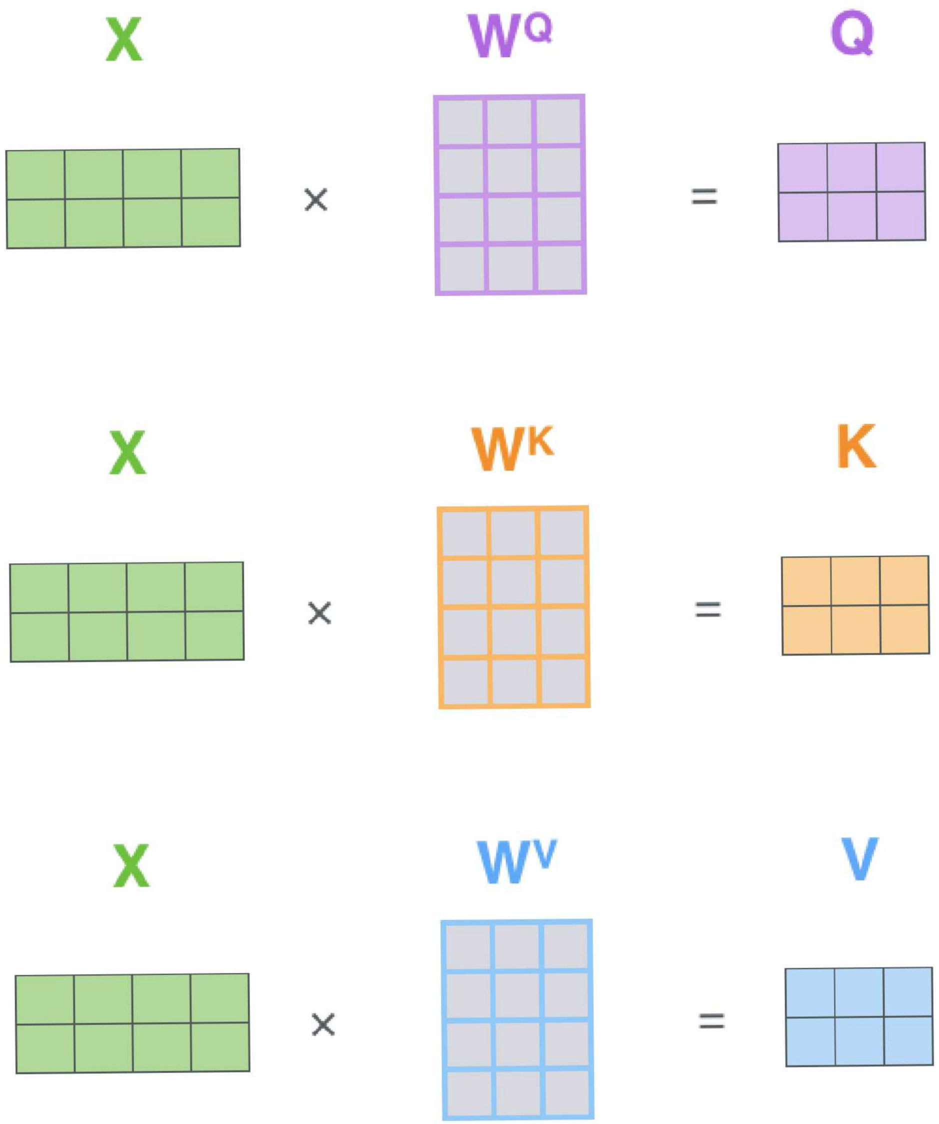
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

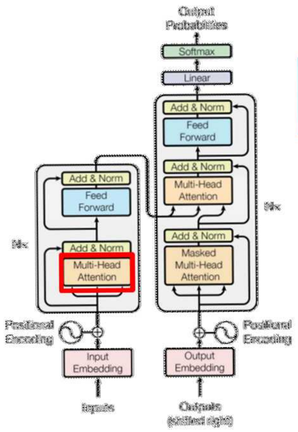
Self-Attention: Intuition



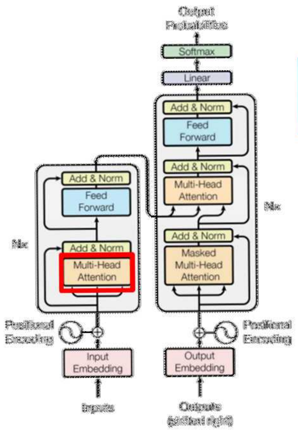


Self-Attention: Matrix Form

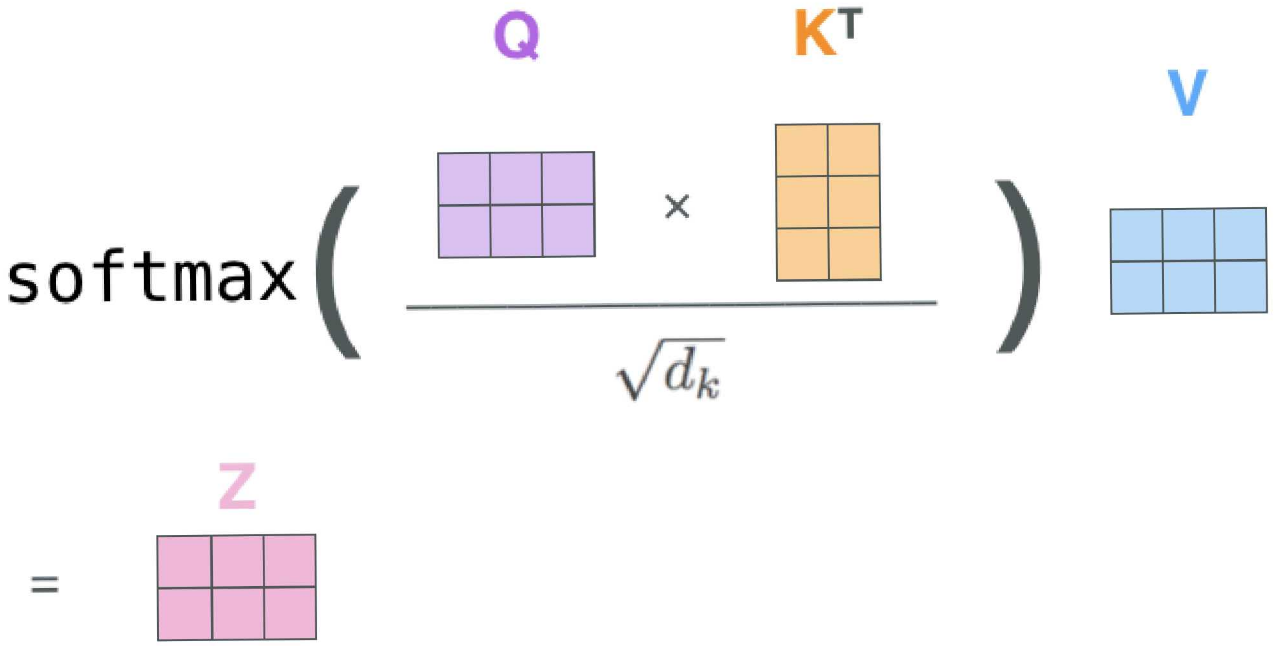




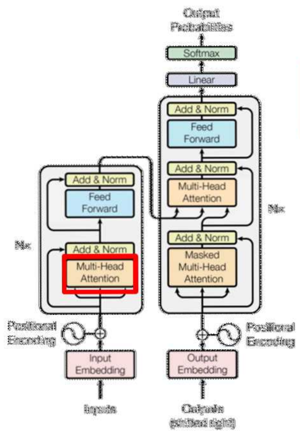
Self-Attention: Matrix Form



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



Multi-Headed Attention

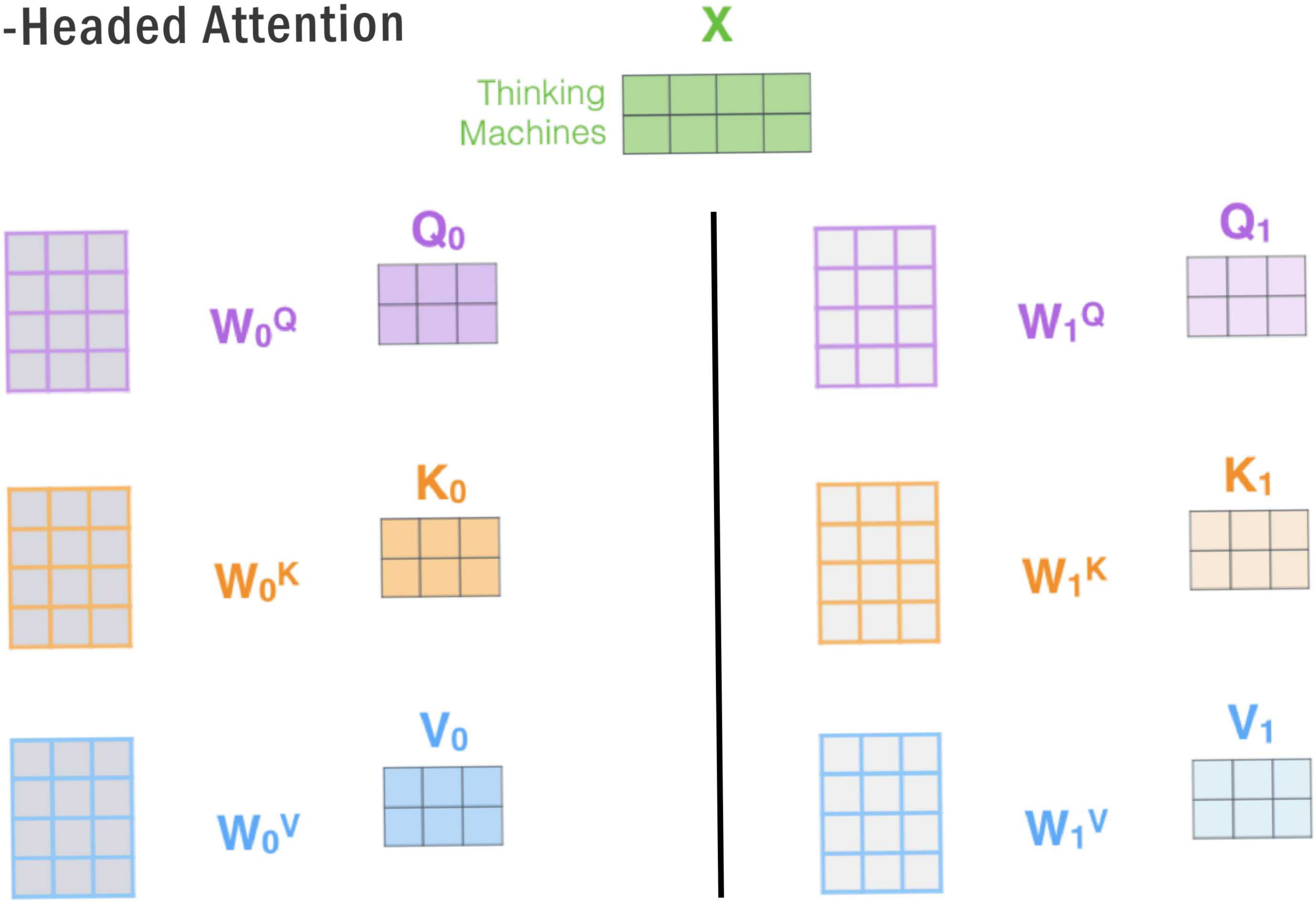


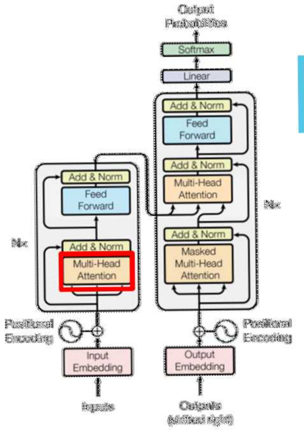
Single attention layer doesn't allow for separate representations of word relationships

- E.g., want to represent word context, part of speech, constituency parse, etc.

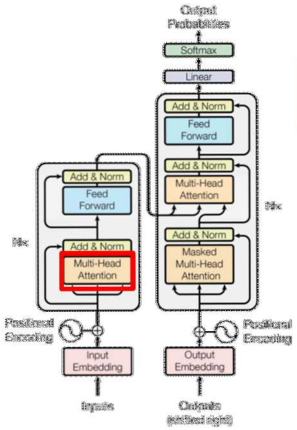
Solution: multiple attention "heads"

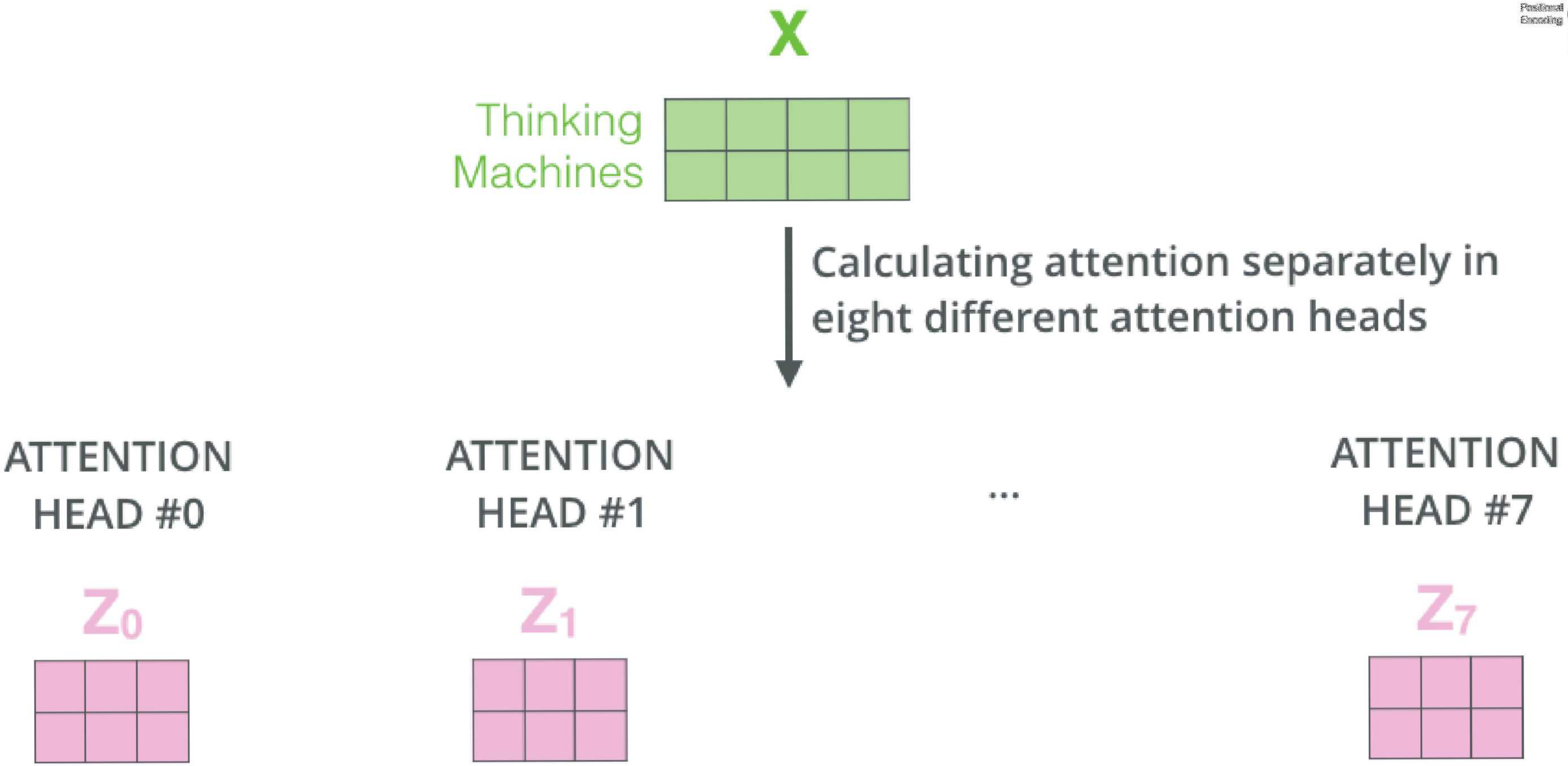
Multi-Headed Attention



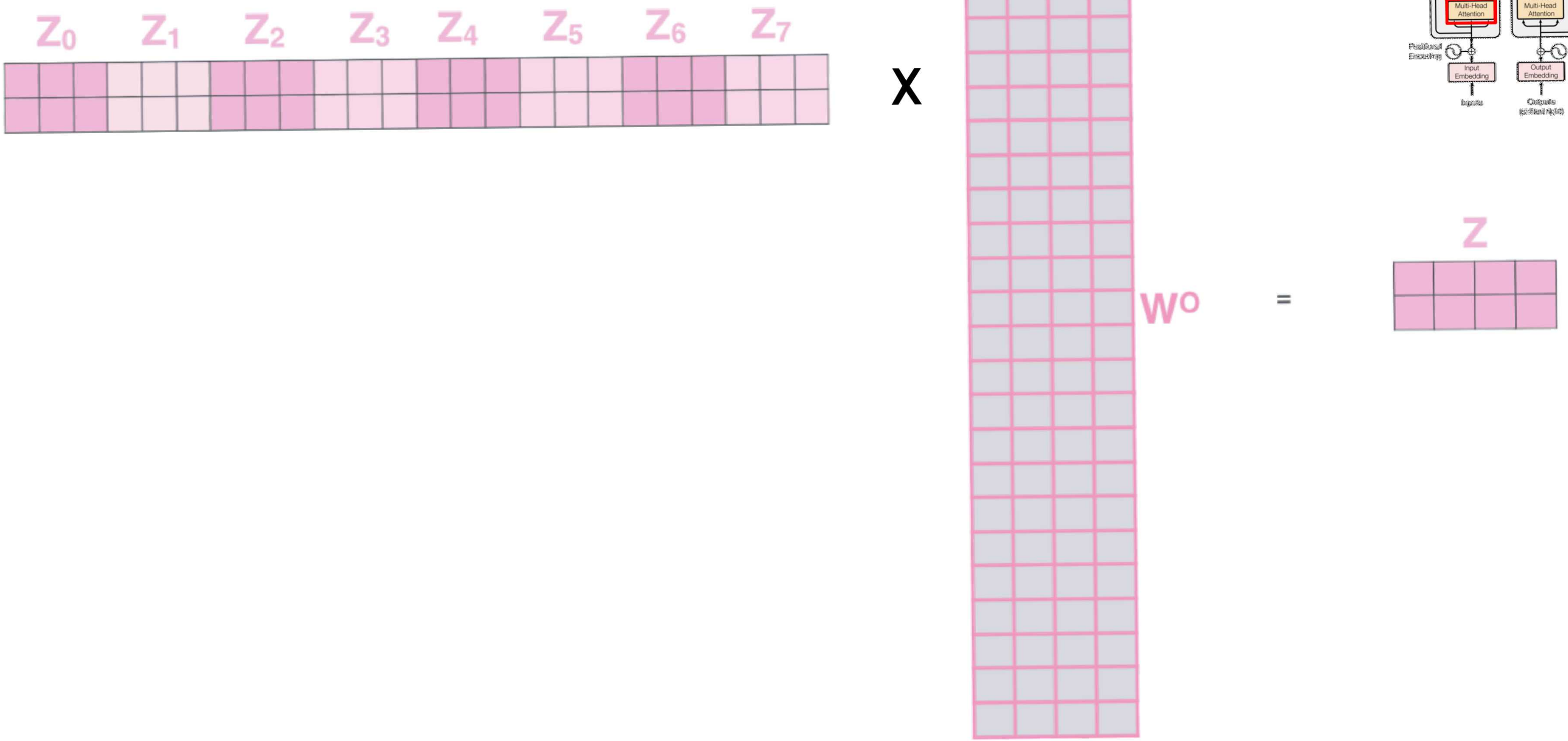


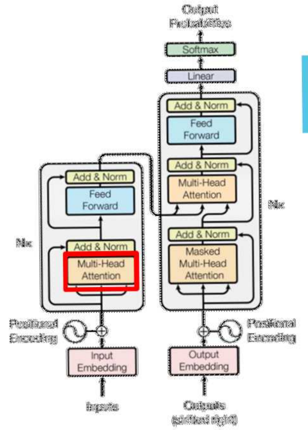
Multi-Headed Attention



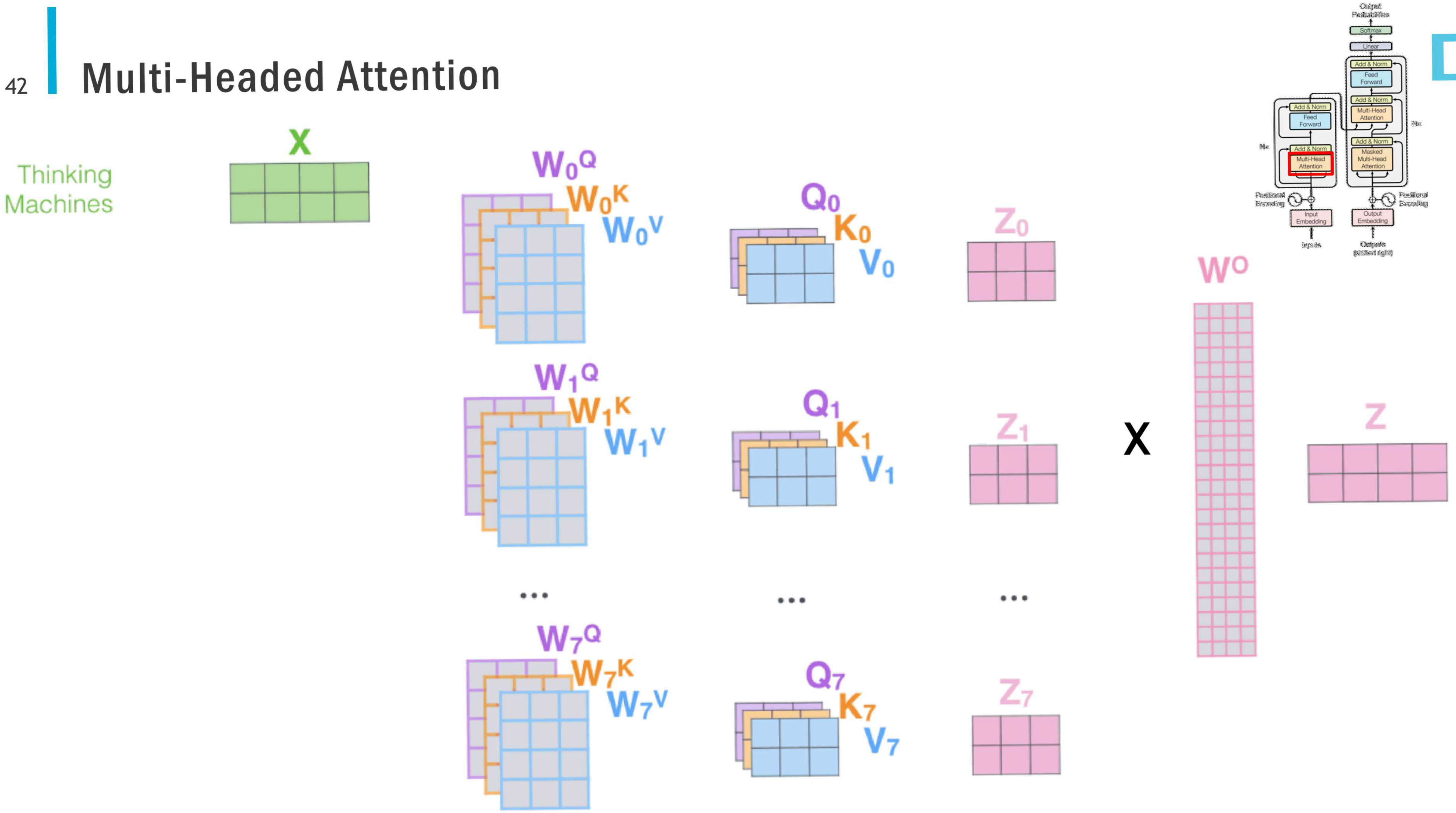


Multi-Headed Attention

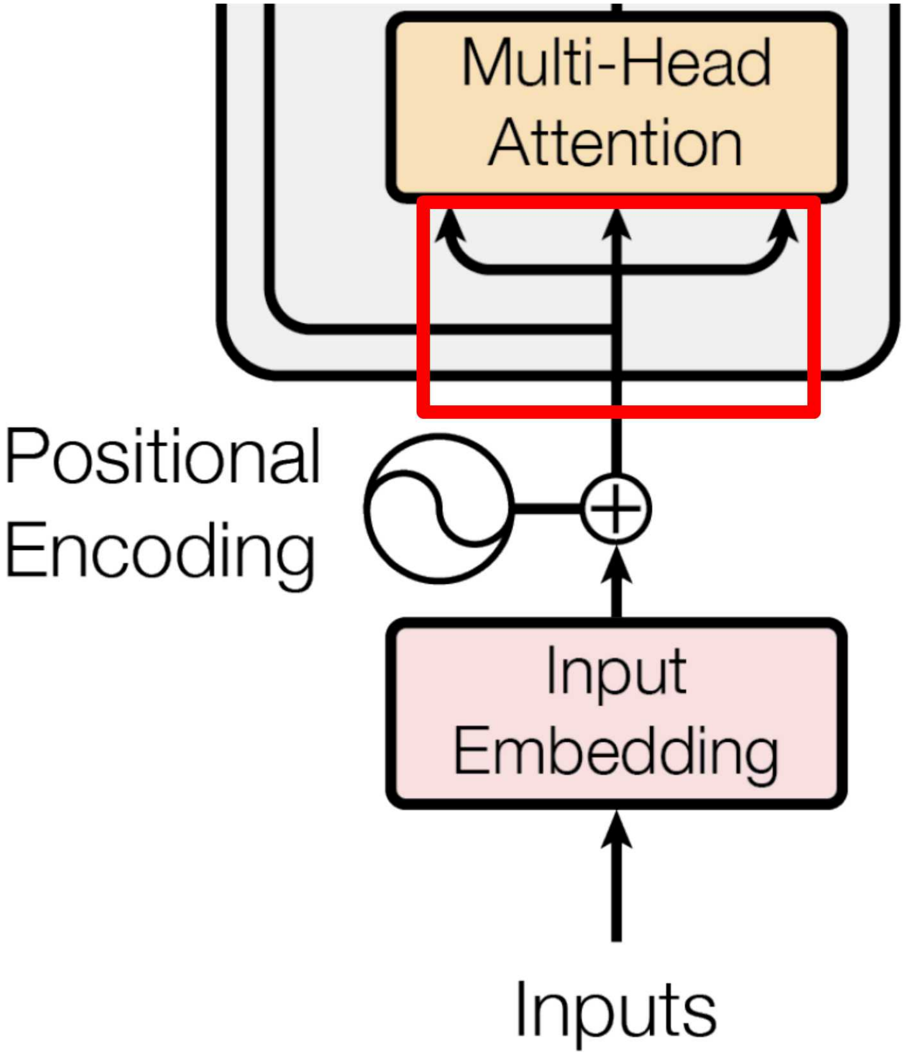


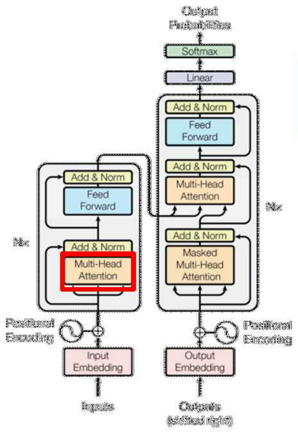


Multi-Headed Attention



Multi-Headed Attention: Implementation Details

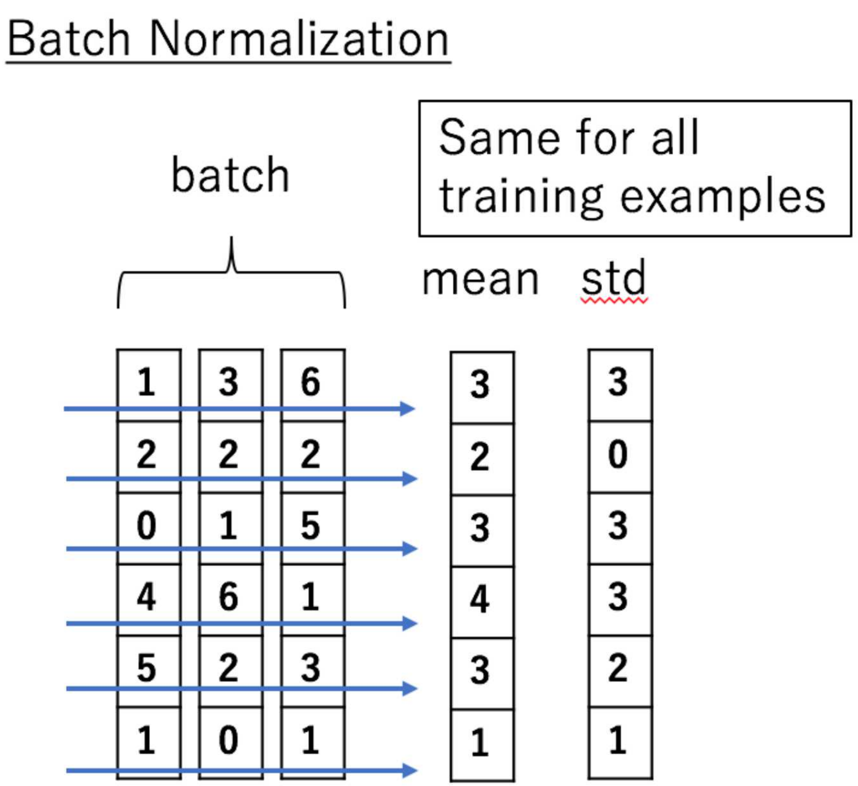


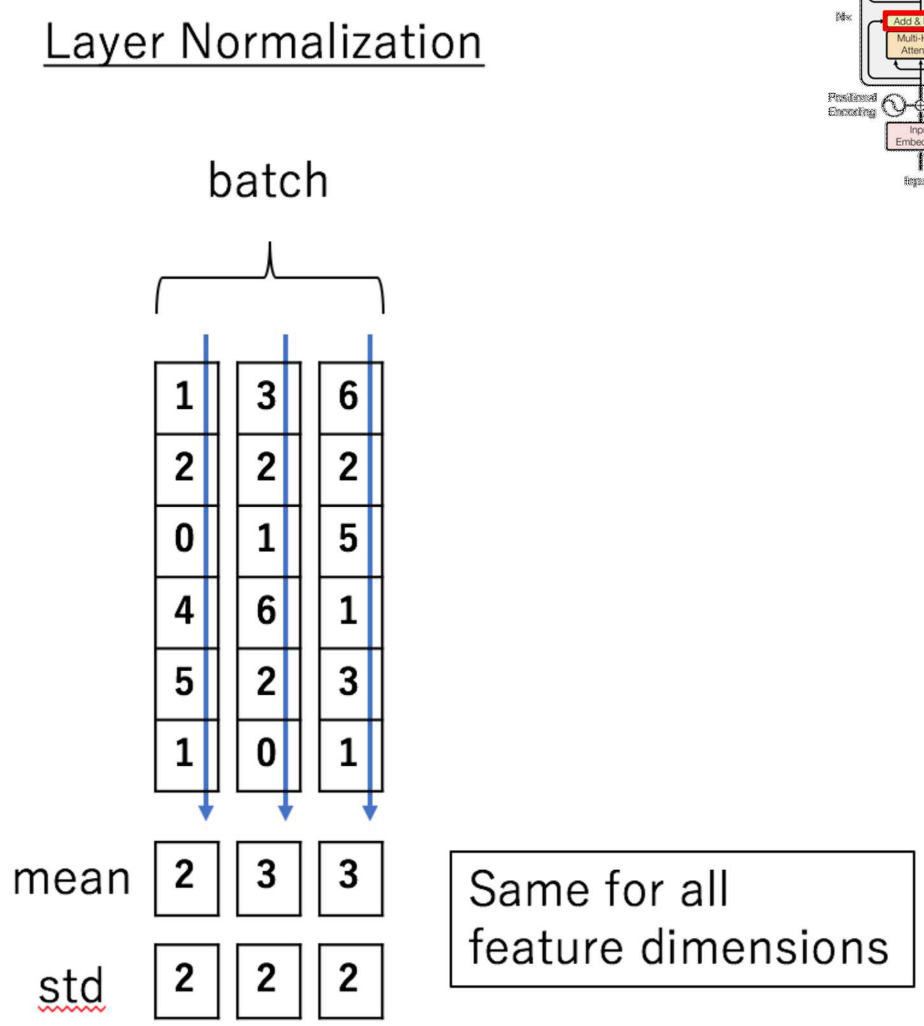


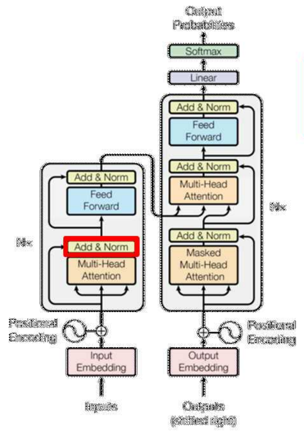
Encoder attention:

- Q, K, V matrices are all the same (**different weight** matrices). For multiple heads, reshape output from matrix multiplication
- I.e., if word embedding dimension is 512 and there are 8 heads, calculate XW^Q = Q (dimension 512), reshape into 8 chunks (result dimension 64 for each head)
- E.g., (seq_len, 512) => (seq_len, 64, 8)
- Thus, embedding dimension must be divisible by number of heads

Layer Normalization



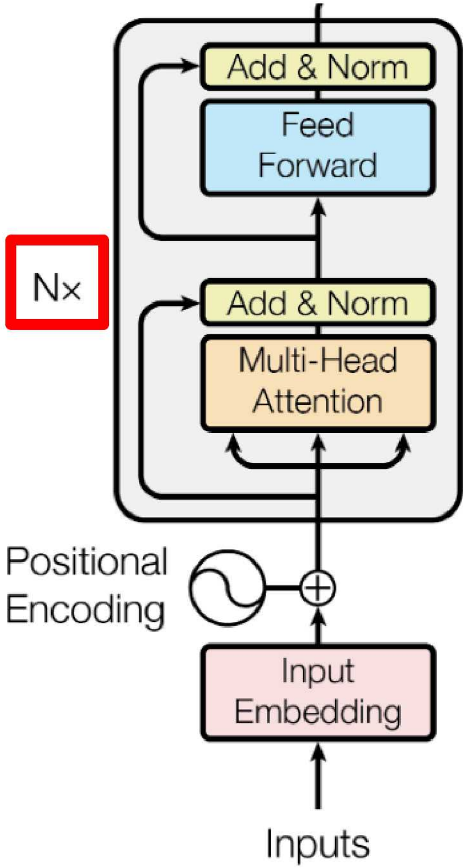




Feed Forward, Encoder Sublayers

Results of multi-headed attention + layer normalization are fed to fully connected feed forward network (2 layers)

- Input/output dimension 512, inner dimension 2048



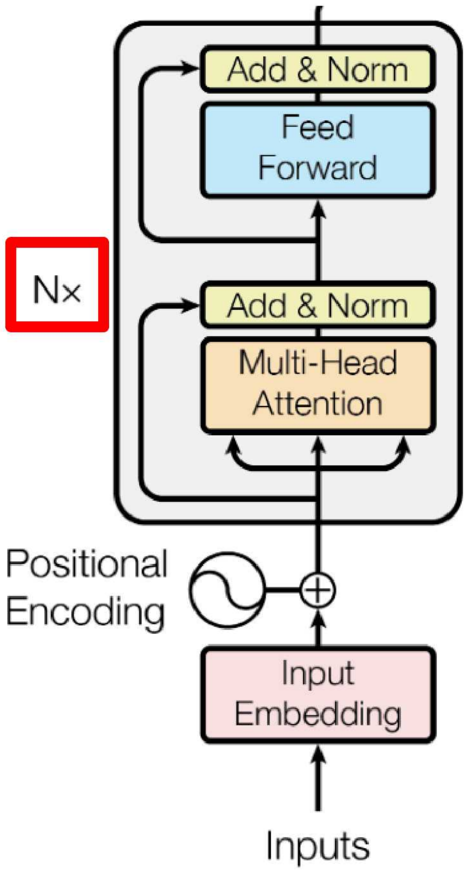
Feed Forward, Encoder Sublayers

Results of multi-headed attention + layer normalization are fed to fully connected feed forward network (2 layers)

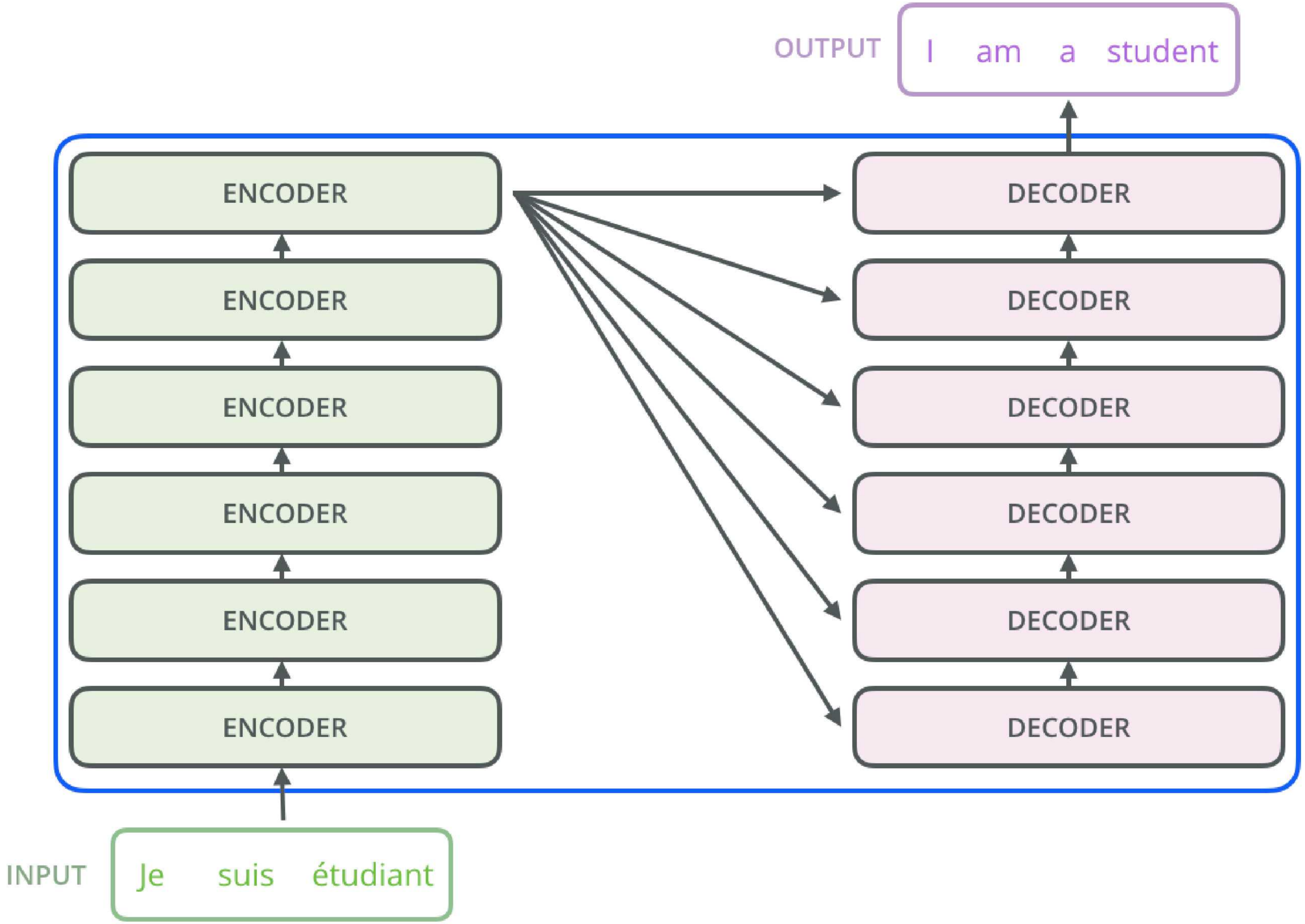
- Input/output dimension 512, inner dimension 2048

Encoder sublayers

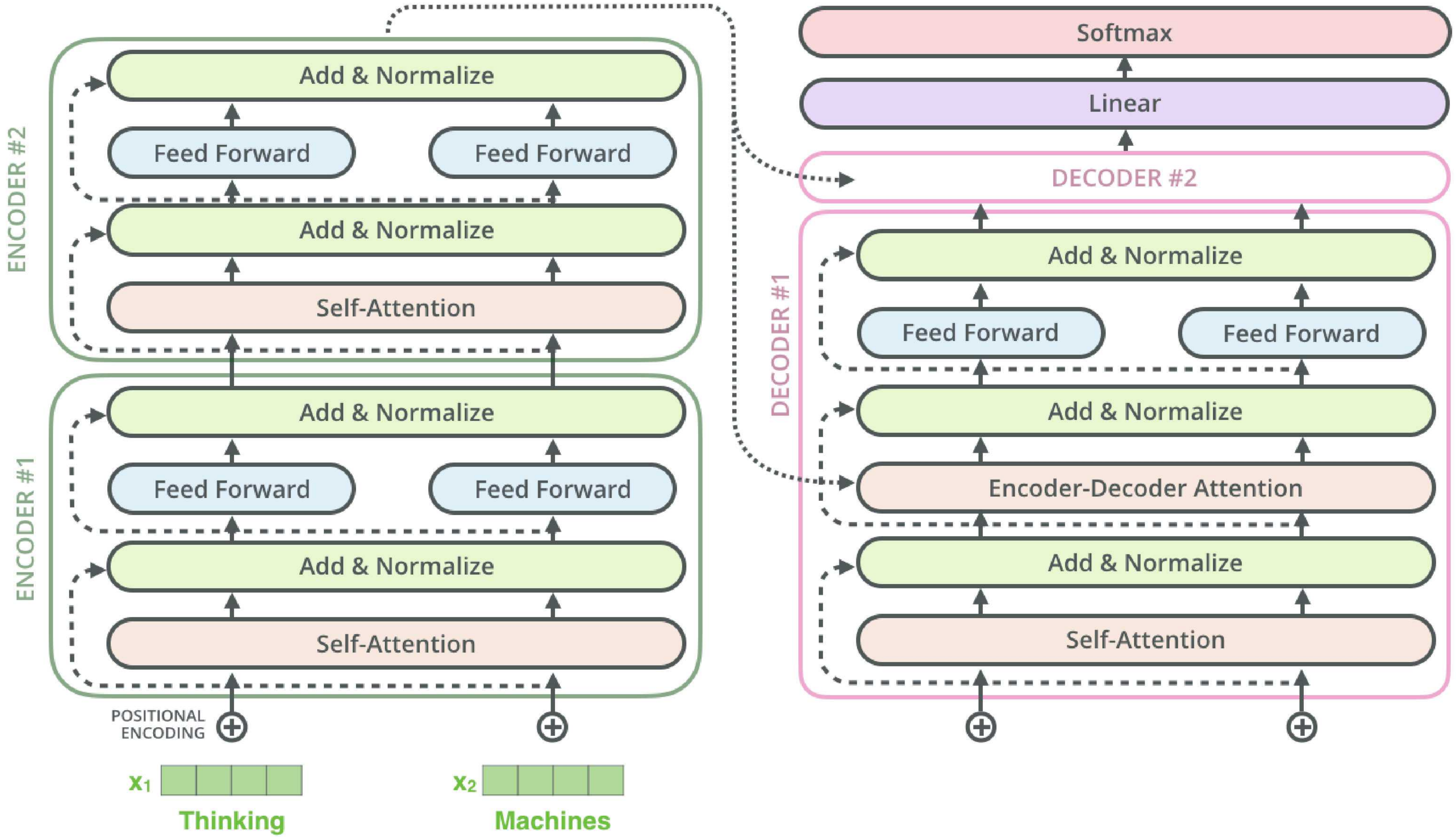
- 0^{th} sublayer, input is embedding of input sequence
- All other sublayers, input is output of feed forward network



Encoder and Decoder Sublayers



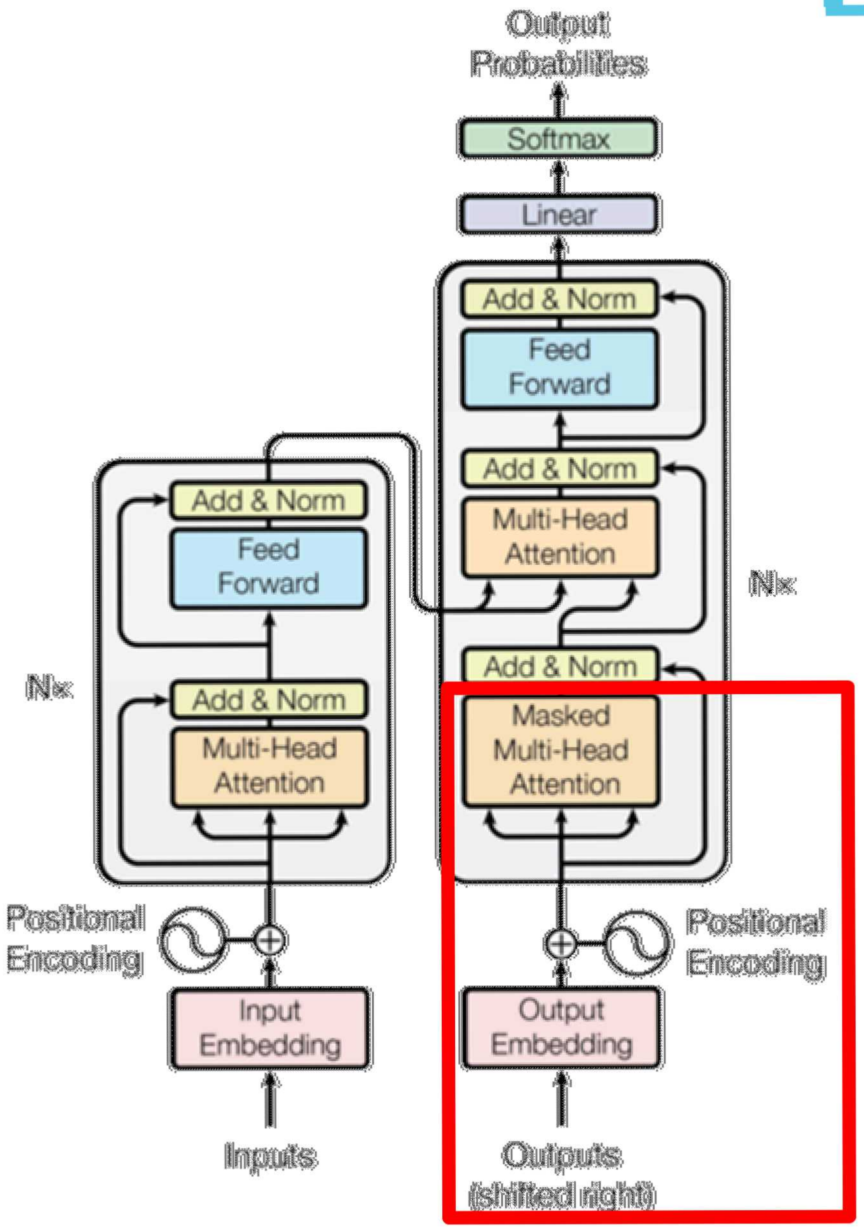
Encoder and Decoder Sublayers



Decoder Teacher Forcing and Masked Attention

Teacher forcing

- Want to be able to use previously seen words in the target sentence to inform future words
- During training, shift the target sentence right by one position (so we don’t have information of future words)



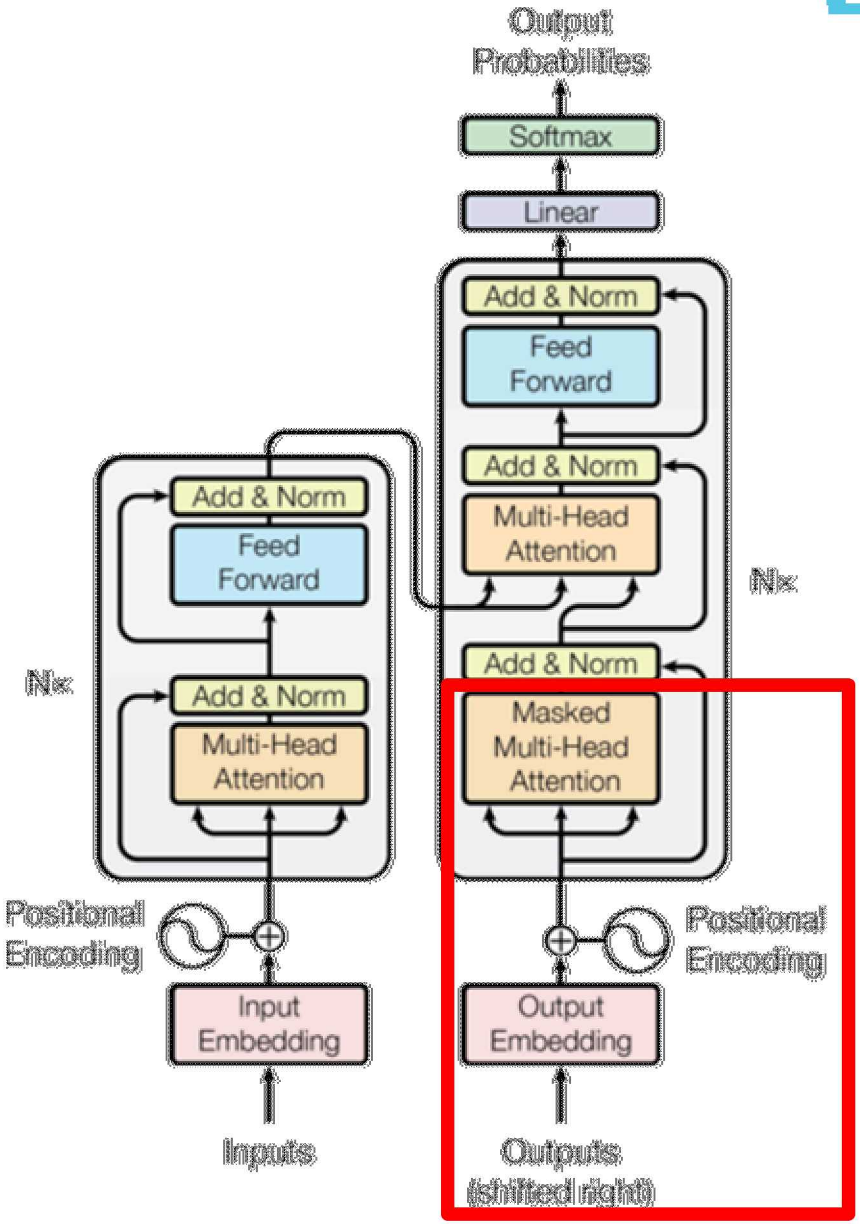
Decoder Teacher Forcing and Masked Attention

Teacher forcing

- Want to be able to use previously seen words in the target sentence to inform future words
- During training, shift the target sentence right by one position (so we don't have information of future words)

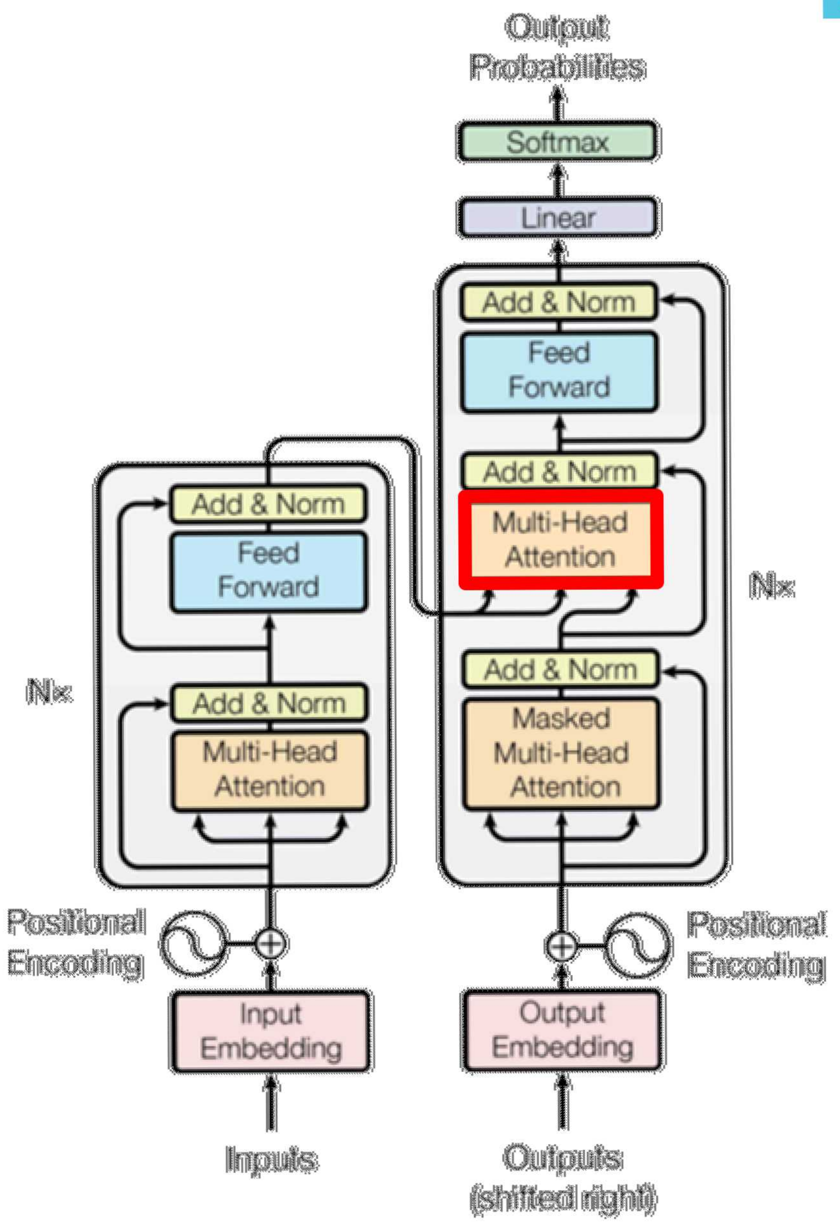
Masked attention

- Same process as encoder attention (Q, K, V based on teacher-forced target sentence), but masked so attention heads do not attend to future words



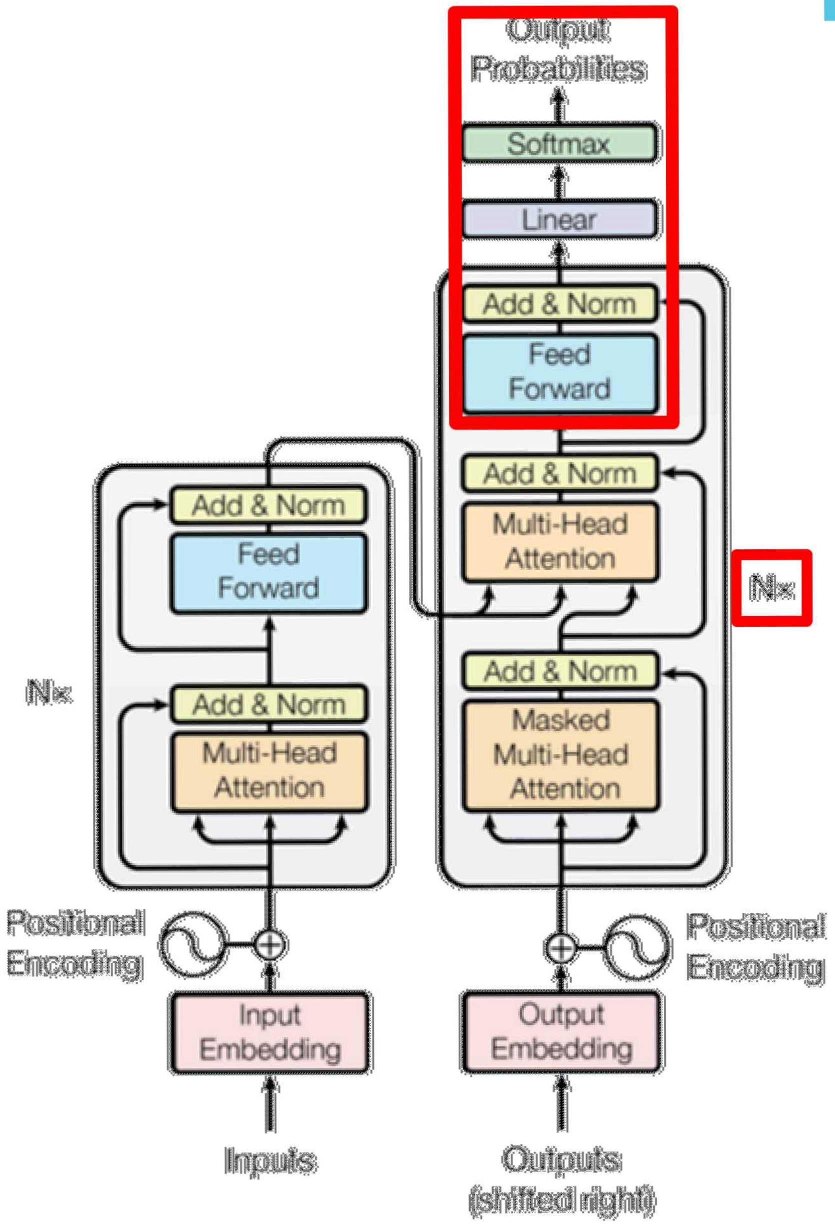
Encoder-Decoder Attention

- K, V are outputs from the final encoder sublayer
- Q is output from masked attention unit



Decoder Final Steps

- Encoder-decoder attention unit gets normalized, put into feed forward network (same dimensions as encoder FFN), repeated N times for each decoder sublayer
- Final output of decoder sublayer put through fully connected linear layer, into softmax over vocabulary



Model Training

Miscellaneous Training Details

Loss function

- Label-smoothed cross-entropy
- Rather than an output target of [1, 0, 0, …, 0] (for k size vocabulary), set the target to [1-e, e/(k-1), e/(k-1), …, e/(k-1)]
- Form of regularization

Miscellaneous Training Details

Loss function

- Label-smoothed cross-entropy
- Rather than an output target of [1, 0, 0, …, 0] (for k size vocabulary), set the target to [1-e, e/(k-1), e/(k-1), …, e/(k-1)]
- Form of regularization

Warmup and LR decay

- Adam optimizer
- $lr = {d_{model}}^{-0.5} * \min(stepnum^{-0.5}, stepnum * warmupsteps^{-1.5})$
  - Linearly increase LR for *warmupsteps*, decrease proportional to inverse square root of *stepnum*

Miscellaneous Training Details

Loss function

- Label-smoothed cross-entropy
- Rather than an output target of [1, 0, 0, …, 0] (for k size vocabulary), set the target to [1-e, e/(k-1), e/(k-1), …, e/(k-1)]
- Form of regularization

Warmup and LR decay

- Adam optimizer
- $lr = {d_{model}}^{-0.5} * \min(stepnum^{-0.5}, stepnum * warmupsteps^{-1.5})$
  - Linearly increase LR for *warmupsteps*, decrease proportional to inverse square root of *stepnum*

Shared embedding weights in input, target, and output

Miscellaneous Training Details

Loss function

- Label-smoothed cross-entropy
- Rather than an output target of [1, 0, 0, …, 0] (for k size vocabulary), set the target to [1-e, e/(k-1), e/(k-1), …, e/(k-1)]
- Form of regularization

Warmup and LR decay

- Adam optimizer
- $lr = {d_{model}}^{-0.5} * \min(stepnum^{-0.5}, stepnum * warmupsteps^{-1.5})$
  - Linearly increase LR for *warmupsteps*, decrease proportional to inverse square root of *stepnum*

Shared embedding weights in input, target, and output

Dropout at each sublayer output and embeddings

Model Prediction

Model Prediction Visualization

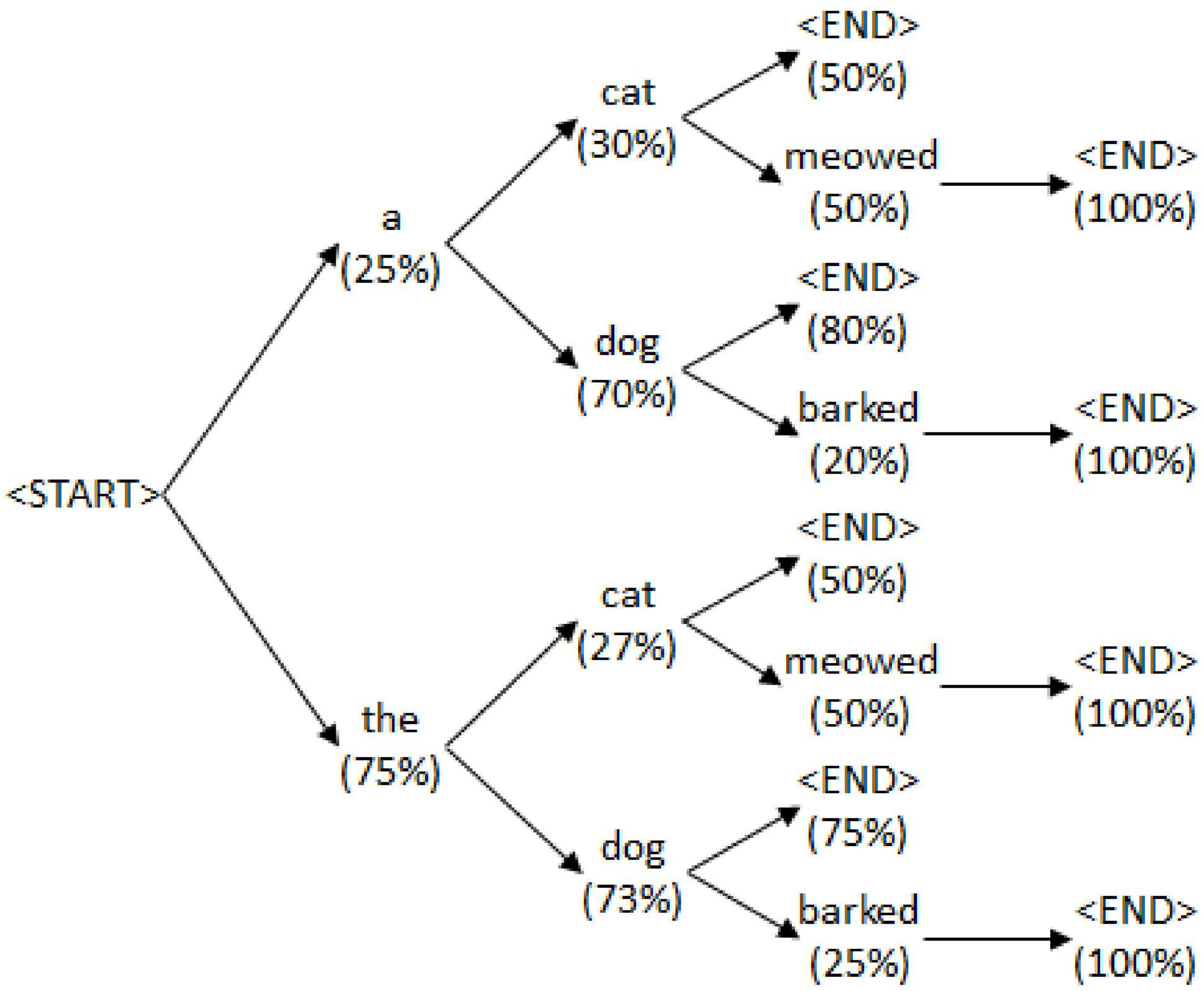
Determining Predicted Words

Greedy decoding

- Predict word that has maximum probability according to model

Beam search

- Expand all possible next word predictions, keep *k* most likely sequences generated. Continue until most probable sequence contains the <END> token.



Experimental Results

Translating C Source Code to English

Source to English Model

Data

- C functions and associated comments extracted using clang
- ~1.4M train, 338k validation, 271k test
- Duplicates removed *prior* to preprocessing

Source to English Model

Data

- C functions and associated comments extracted using clang
- ~1.4M train, 338k validation, 271k test
- Duplicates removed *prior* to preprocessing

Tokenization

- Source code: split on variable names, language keywords, operators, punctuators
- Comments: replace numbers with special token, remove punctuation, normalize whitespace

Source to English Model

Data

- C functions and associated comments extracted using clang
- ~1.4M train, 338k validation, 271k test
- Duplicates removed *prior* to preprocessing

Tokenization

- Source code: split on variable names, language keywords, operators, punctuators
- Comments: replace numbers with special token, remove punctuation, normalize whitespace

Models trained

- LSTM, transformer, fconv, dynconv, transformer with back-translation

Source to English Model

Data

- C functions and associated comments extracted using clang
- ~1.4M train, 338k validation, 271k test
- Duplicates removed *prior* to preprocessing

Tokenization

- Source code: split on variable names, language keywords, operators, punctuators
- Comments: replace numbers with special token, remove punctuation, normalize whitespace

Models trained

- LSTM, transformer, fconv, dynconv, transformer with back-translation

BLEU (bilingual evaluation underscore) as evaluation metric

- Best score: 18.26 using transformer

Source to English Model Results

```
ORIG: char getField ( struct board * target , int x    , int
y ) if ( x NUMBERTOKEN y NUMBERTOKEN x      target width y
target height ) return          FIELDOUTOFBOUNDS ; return *
calcFieldAddress (    target , x , y ) ;

TGT: Gets what is on a given field of the board returns
   FIELDXYZ constant

HYP: Returns the value of the field at the given
   coordinates
```

Source to English Model Results

ORIG: static void makedevice (char * path , int delete) const char * devicename ; int major , minor , type , len ; int mode NUMBERTOKEN ; uidt uid NUMBERTOKEN ; gidt gid NUMBERTOKEN ; char * devmajmin path strlen (path) ; *...rest of code omitted for slide brevity*

TGT: mknod in dev based on a path like sysblockhdahda1

HYP: mknod in dev based on a path like sysblockhdahdahdahdahdahdahdahdahdahdahdahdahda1 based on a dev based on a dev based on a path like sysblockhdahdahda1

Additional Resources

Open Source Implementations

https://github.com/tensorflow/tensor2tensor

https://github.com/pytorch/fairseq

https://github.com/OpenNMT/OpenNMT-py

https://github.com/huggingface/pytorch-pretrained-BERT

…

Additional Learning Resources

http://jalammar.github.io/illustrated-transformer/

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

https://towardsdatascience.com/how-to-code-the-transformer-in-pytorch-24db27c8f9ec