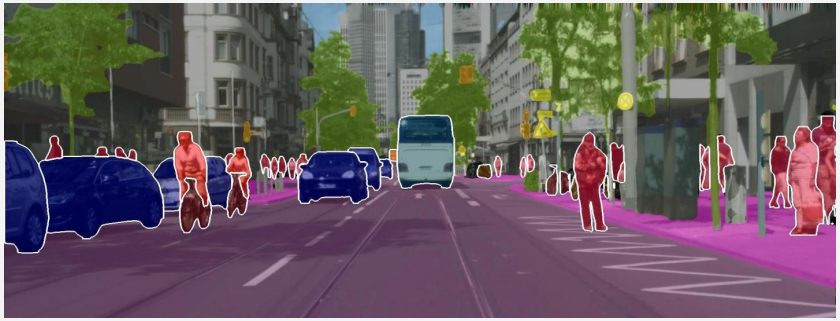
ROB 537
Learning-Based Control

Week **5**, Lecture **1**

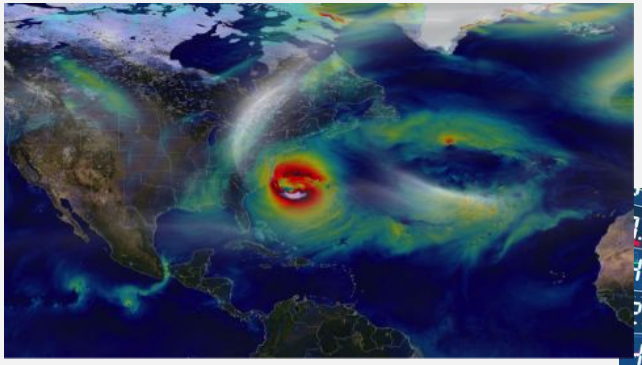
Attention and Transformers

Oregon State University



eurekalert

AWS - BERT



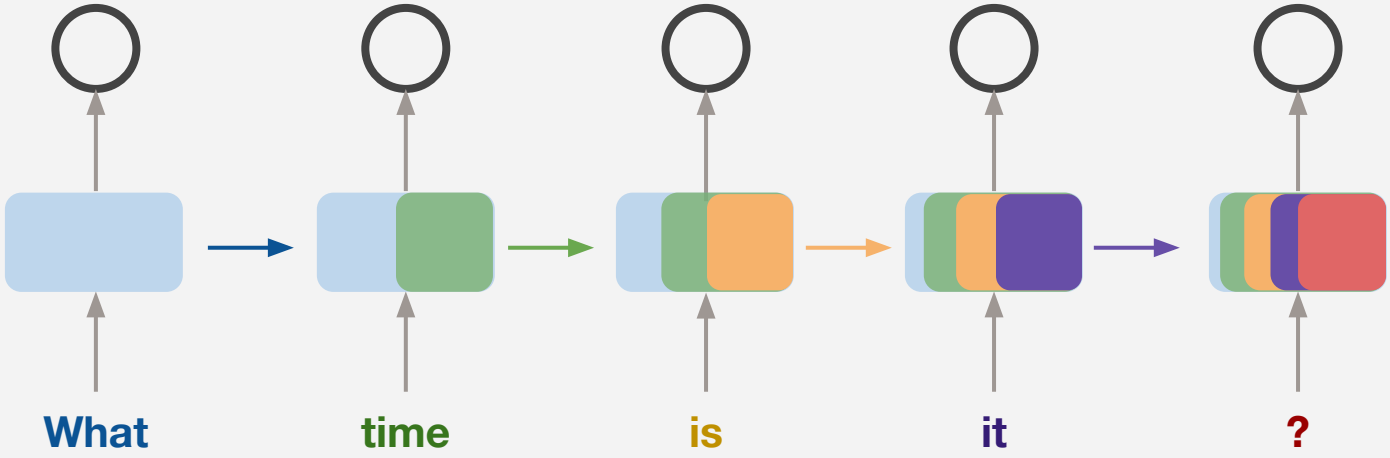
NASA

Adobe

Adobe

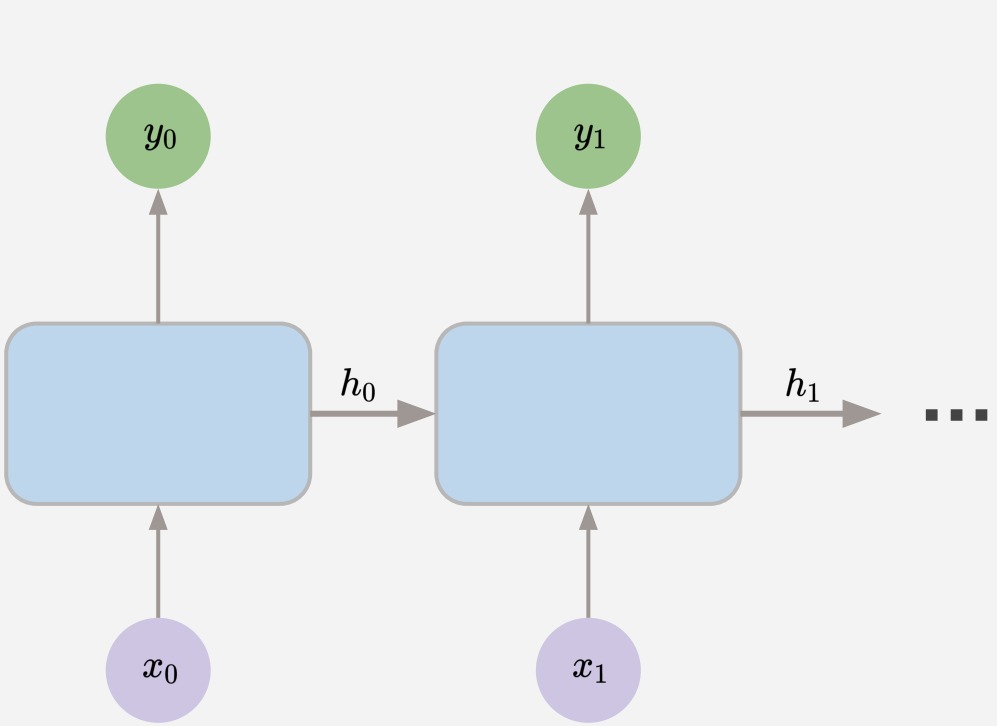
Sequences in the Wild

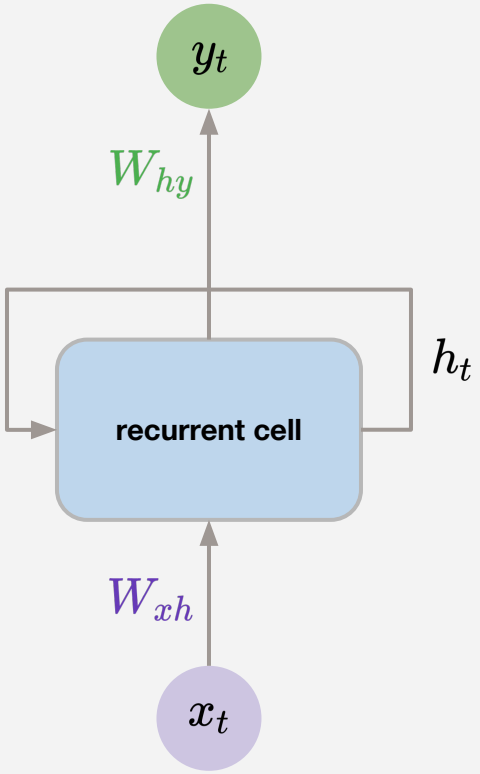
RNNs: Feeding Sequential Data Example



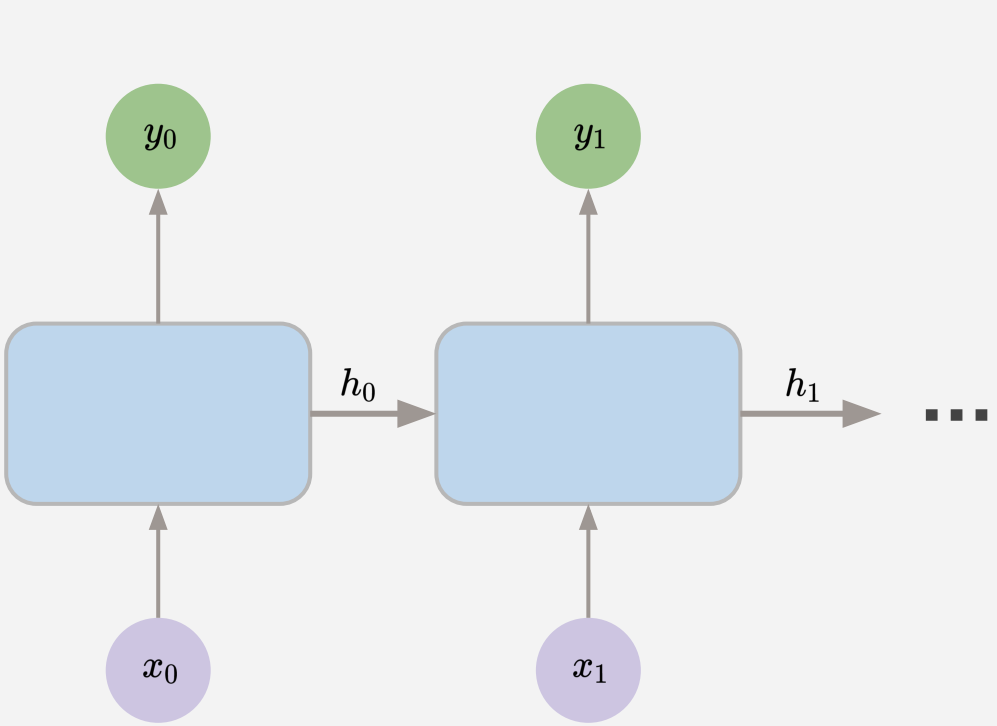
What time is it ?

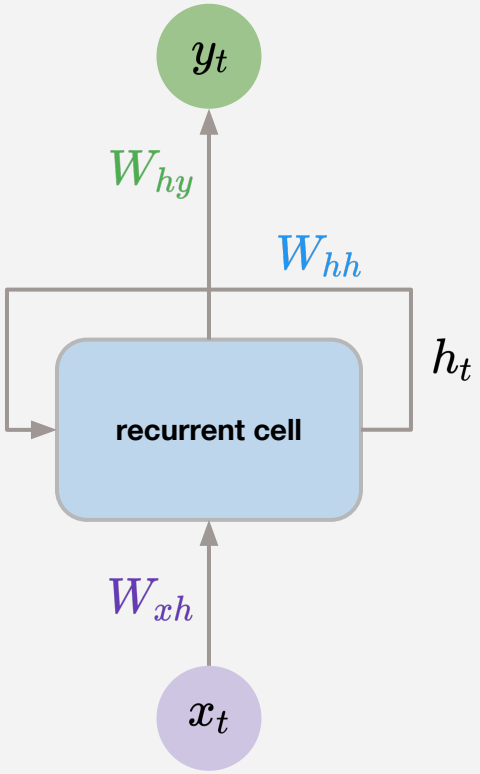
RNNs



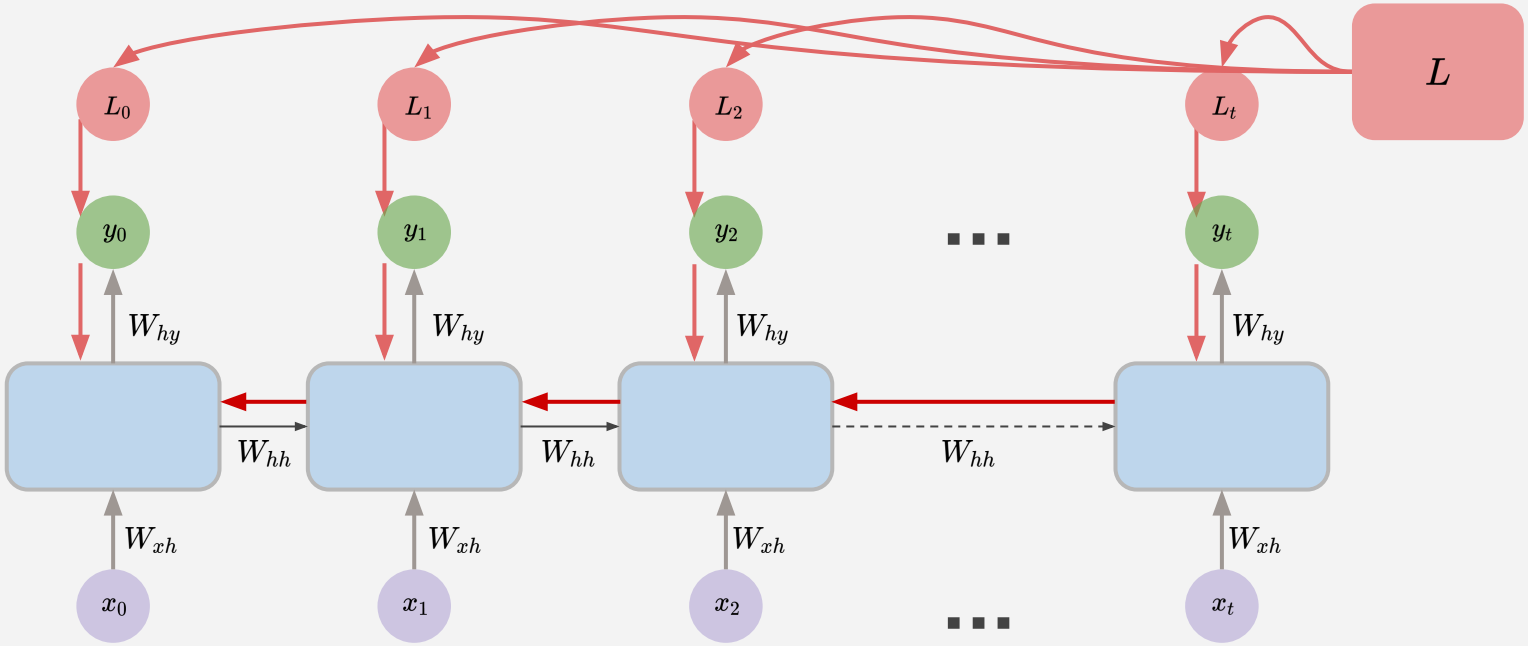


RNNs





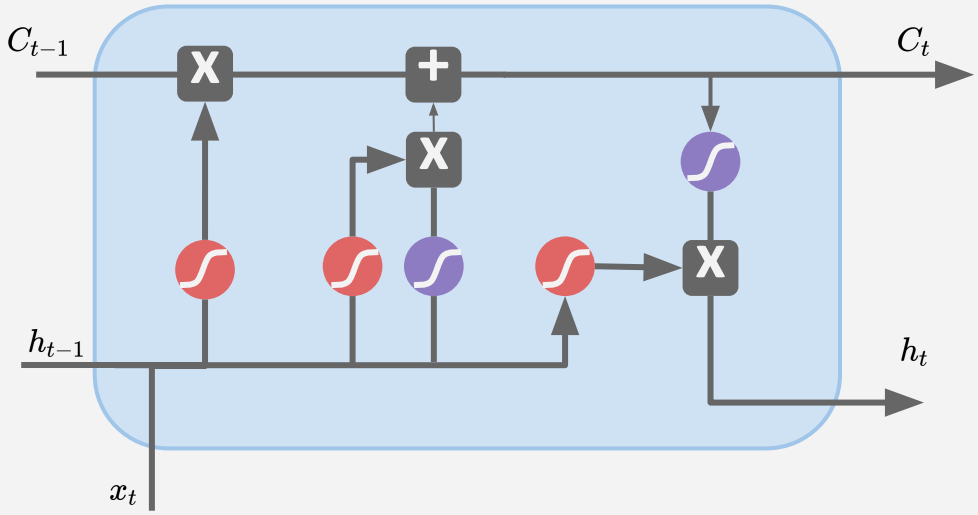
BPTT: Vanishing + Exploding Gradients



$$\frac{\partial L}{\partial W_{xh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{xh}}$$

The product of several weights smaller than 1.0 leads to vanishing gradients that limit the size of the network's reference window (since weights further back get closer to zero)

Review: LSTMs



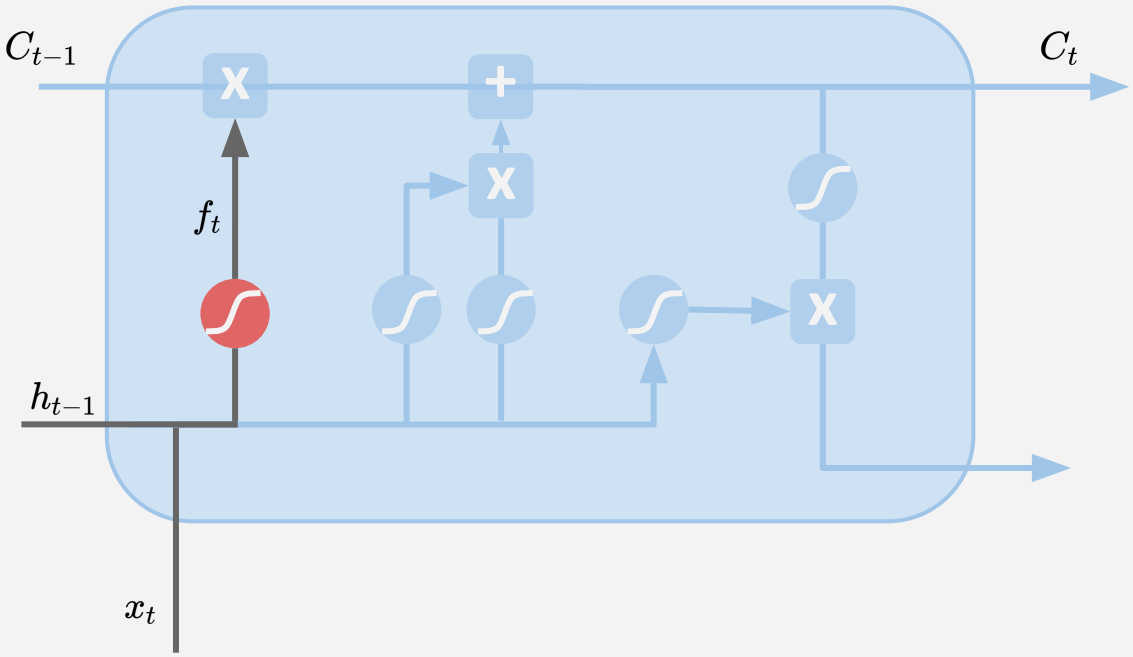
1. Maintains a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information from the cell state
   b. **Input gate** to selectively use the current input and hidden state information
   c. **Output gate** to propagate the modified cell state to the next cell

3. Gates are parameterized

   Backpropagation through time will update gradients to **control gates** for **long-term persistence**

LSTM Cell: Forget Gate

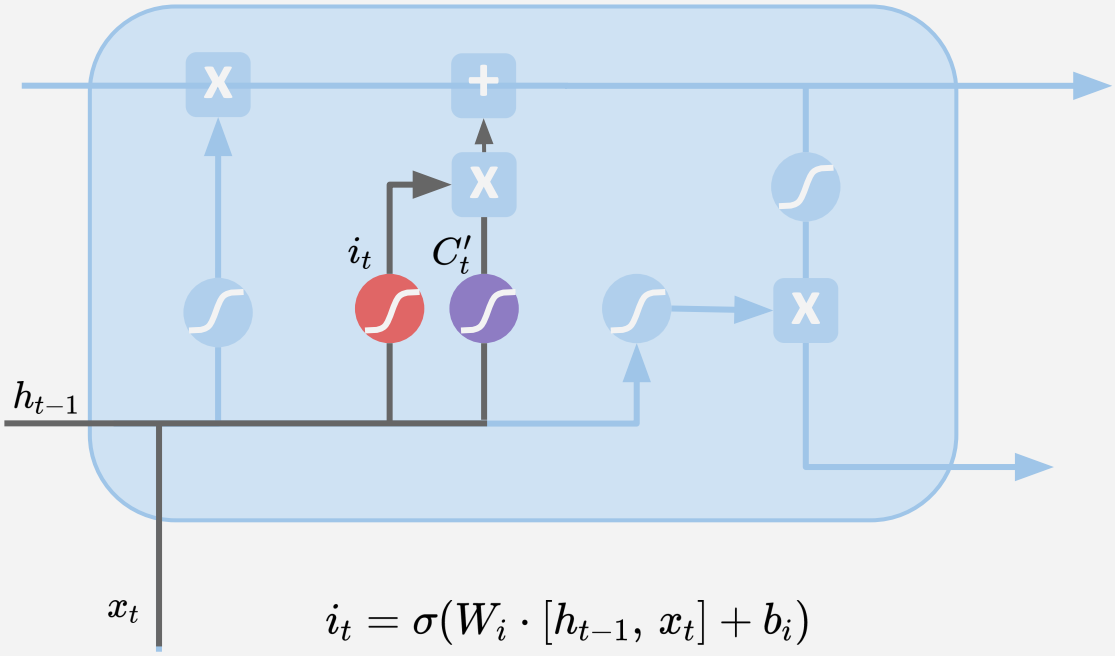


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Forget Gate: Conditioned on the current input and the last hidden state, **decide how much information to throw away** from the cell state.

A one through the sigmoid activation means “keep all of f_t in the cell state”, whereas zero means “forget everything from the cell state”.

LSTM Cell: Input Gate



$$i_t = \sigma(W_i \cdot [h_{t-1},\, x_t] + b_i)$$

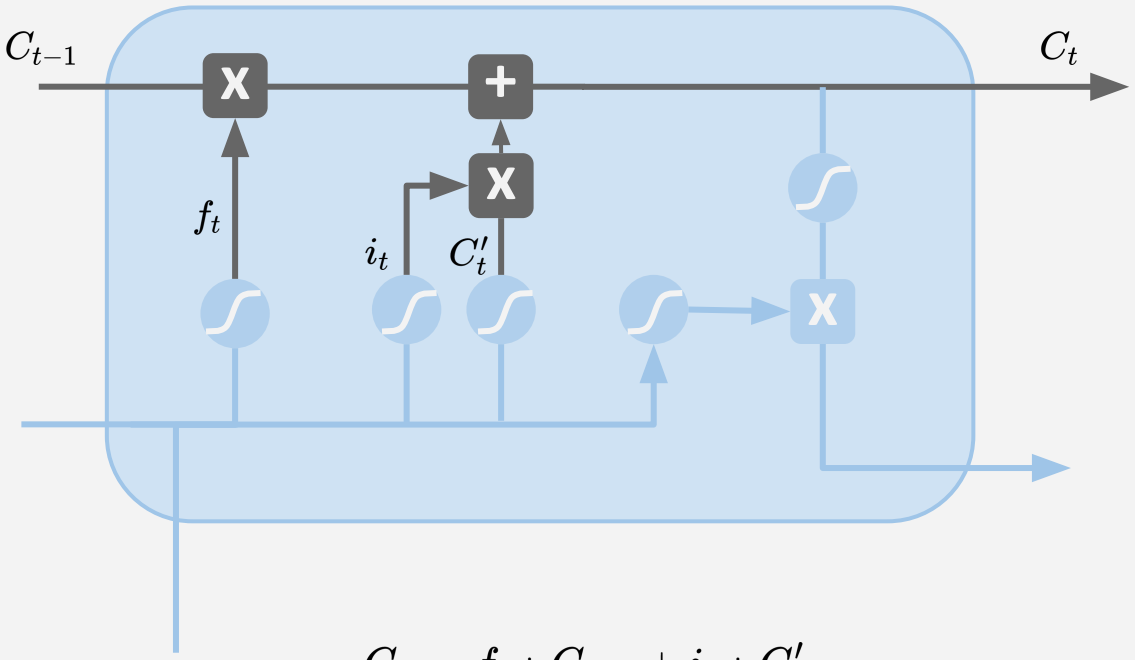
$$C'_t = \tanh\left(W_C \cdot [h_{t-1},\, x_t] + b_C\right)$$

Input Gate: Decide what **new information to store** in the cell state.

C'_t dictates the new information to accumulate from the previous hidden state and input.

The input gate layer i_t dictates how much percentage (notice the sigmoid activation) of C'_t to propagate.

LSTM Cell: Cell State Update

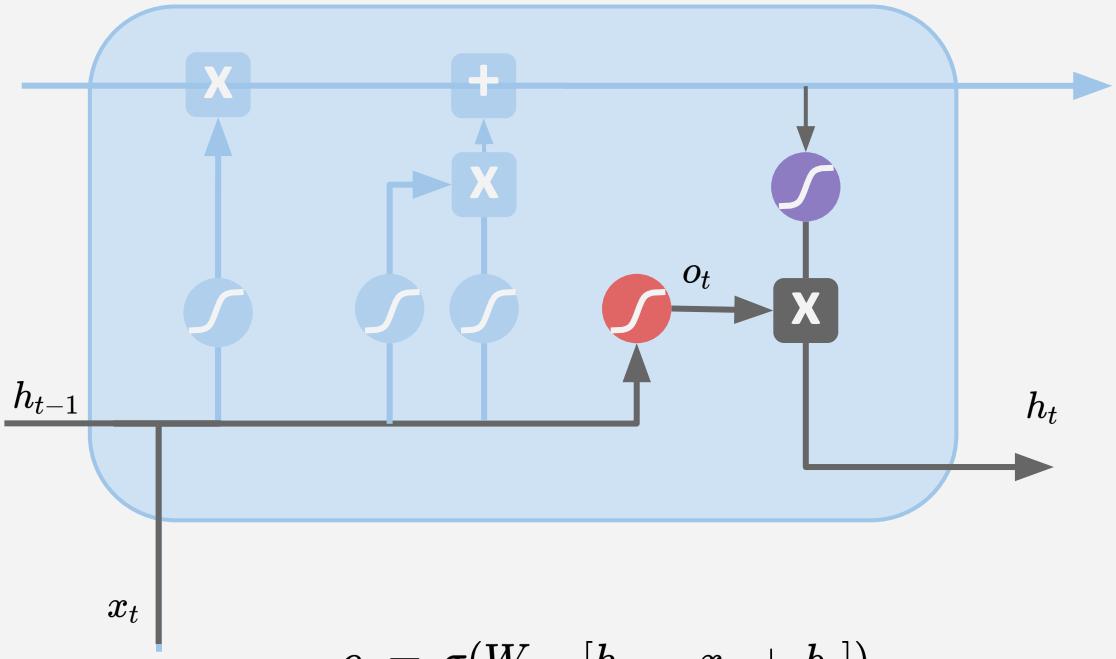


$$C_t = f_t * C_{t-1} + i_t * C'_t$$

Cell State Update: Incorporate information that has passed through the forget and the input gate to create new cell state C_t

The updated cell state C_t, is a filtered version of the previous cell state, that has potentially incorporated new information relevant to the current input and previous hidden state

LSTM Cell: Output Gate



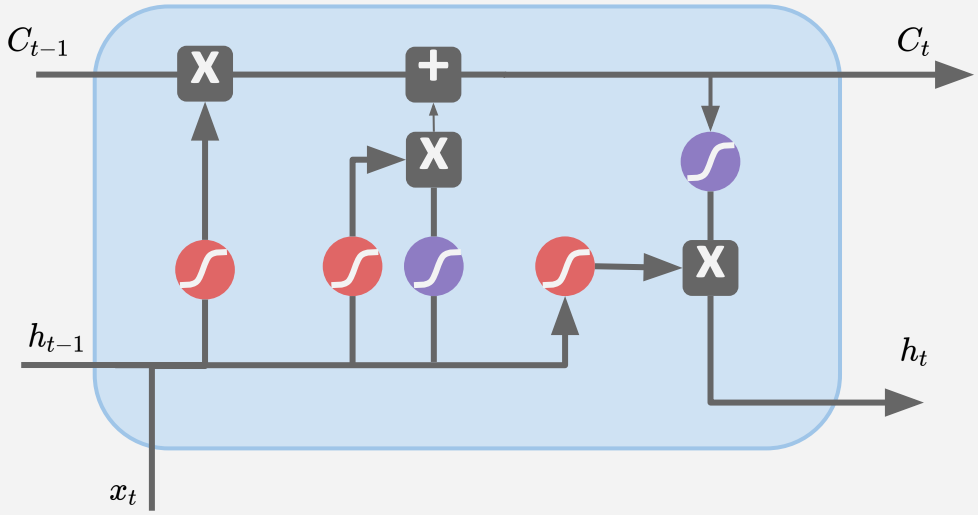
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t + b_o])$$

$$h_t = o_t * \tanh(C_t)$$

Output Gate: The output is a filtered version of the updated cell state C_t.

The output gate propagates a percentage (notice the sigmoid activation for o_t) of the current input and previous hidden state, along with the updated cell state C_t.

Review: LSTMs



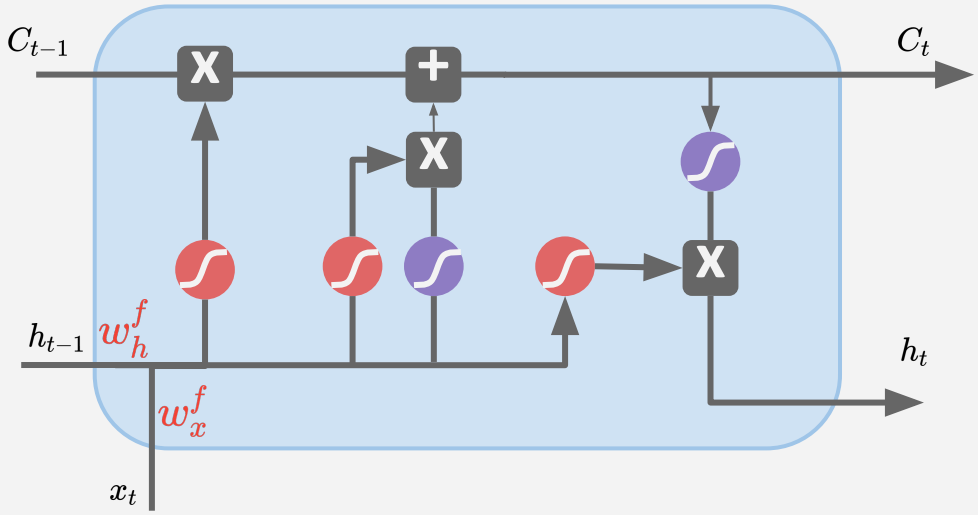
1. Maintains a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information from the cell state
   b. **Input gate** to selectively use the current input and hidden state information
   c. **Output gate** to propagate the modified cell state to the next cell

3. Gates are parameterized

Backpropagation through time will update gradients to **control gates** for **long-term persistence**

Review: LSTMs



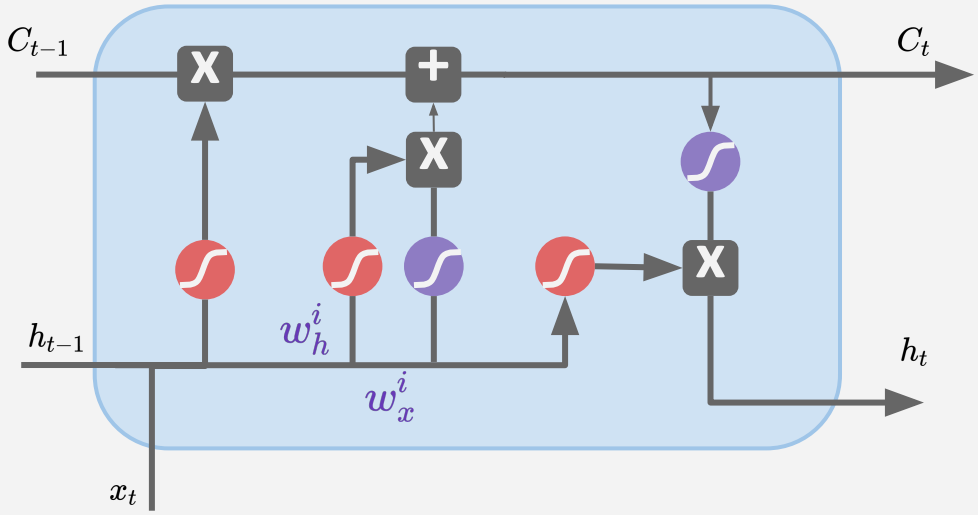
1. Maintains a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information from the cell state
   b. **Input gate** to selectively use the current input and hidden state information
   c. **Output gate** to propagate the modified cell state to the next cell

3. Gates are parameterized

Backpropagation through time will update gradients to **control gates** for **long-term persistence**

Review: LSTMs



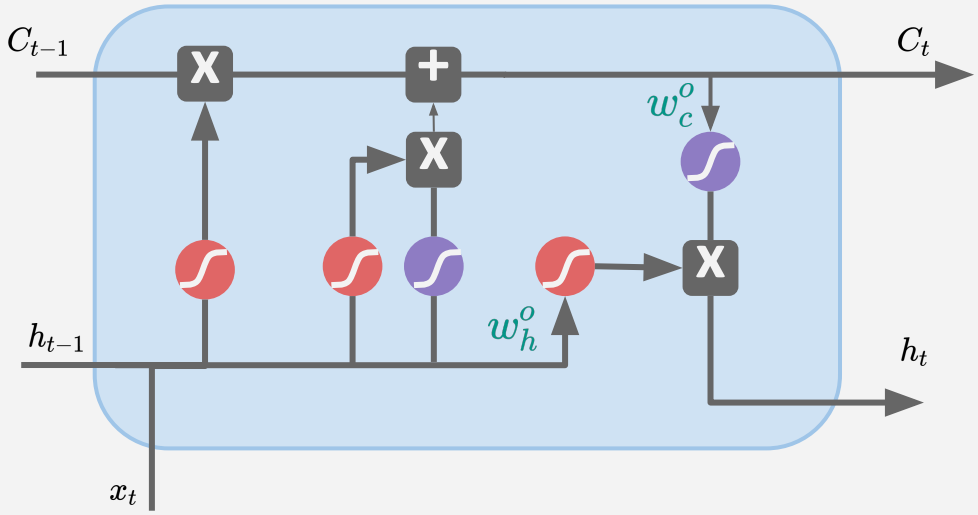
1. Maintains a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information from the cell state
   b. **Input gate** to selectively use the current input and hidden state information
   c. **Output gate** to propagate the modified cell state to the next cell

3. Gates are parameterized

   Backpropagation through time will update gradients to **control gates** for **long-term persistence**

Review: LSTMs



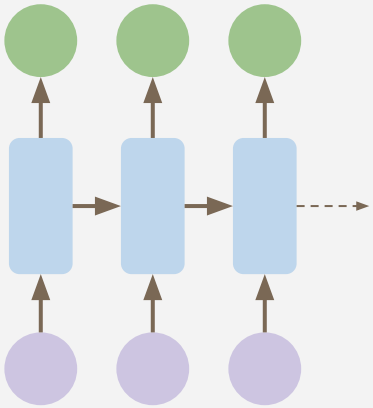
1. Maintains a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information from the cell state
   b. **Input gate** to selectively use the current input and hidden state information
   c. **Output gate** to propagate the modified cell state to the next cell

3. Gates are parameterized

Backpropagation through time will update gradients to **control gates** for **long-term persistence**

Limitations of Recurrent Models

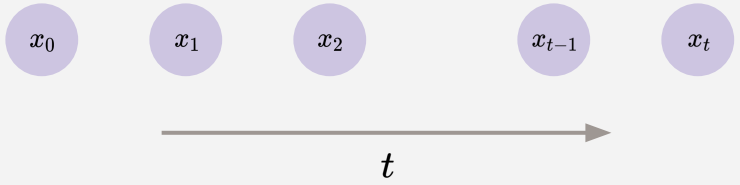


1. Encoding bottleneck
   - Temporal separation required
   - Storing and maintaining large sequences
2. Short Memory - Finite reference window
   - Difficulty in accessing information from many time steps ago
   - Vanishing gradients
3. Slow computation
   - Cannot be parallelized - Inputs are fed one token at a time, serially.

Goals of Sequence Modeling

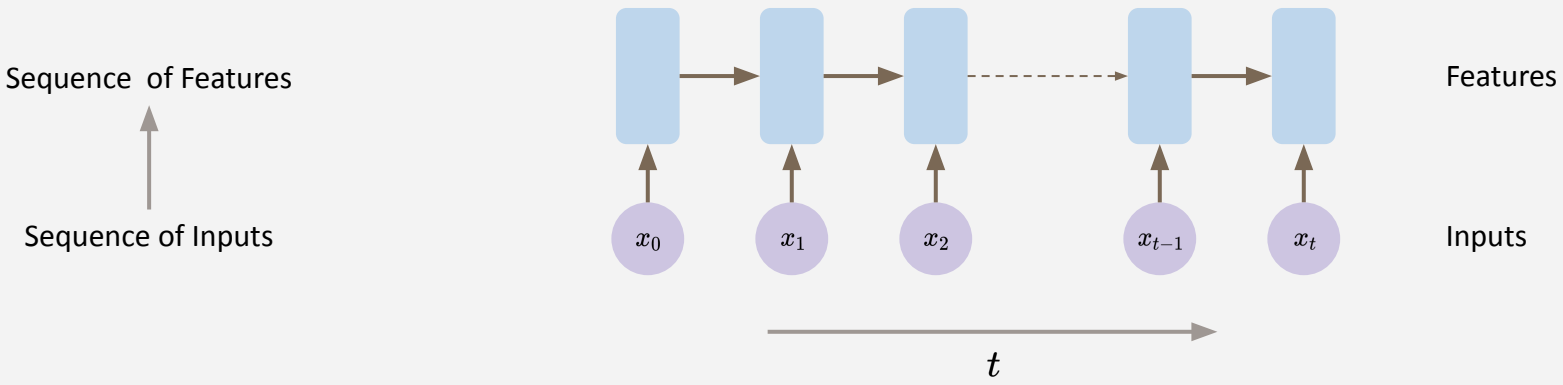
Goals of Sequence Modeling

Sequence of Inputs

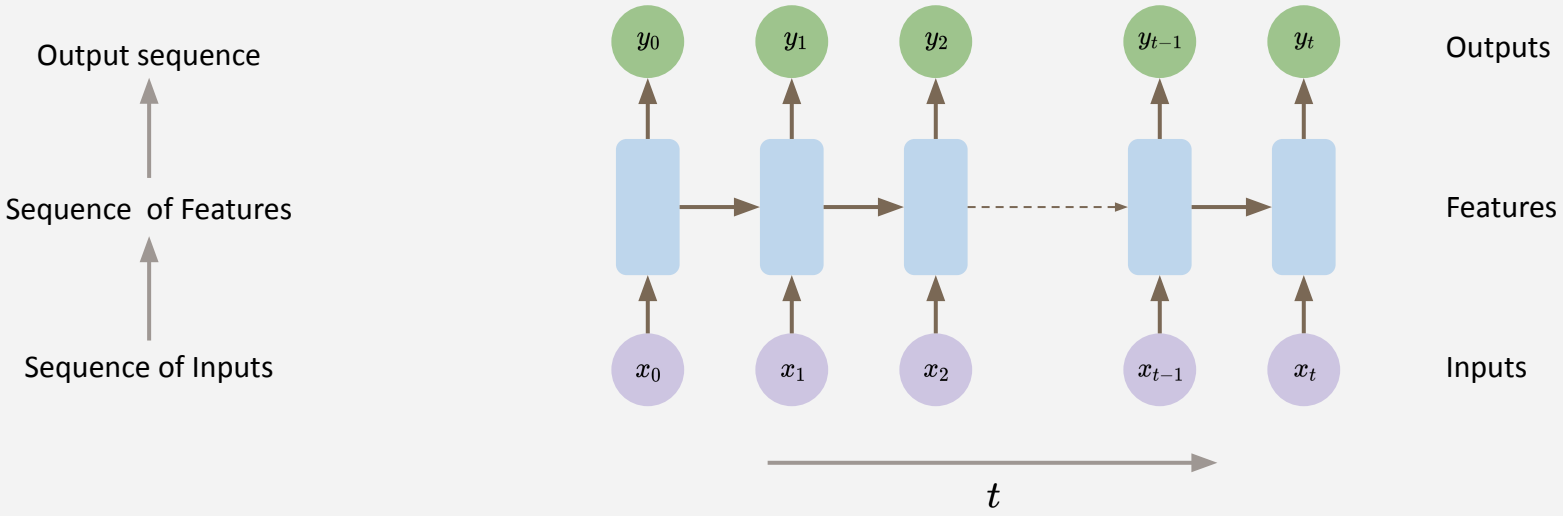


Inputs

Goals of Sequence Modeling



Goals of Sequence Modeling



Goals of Sequence Modeling

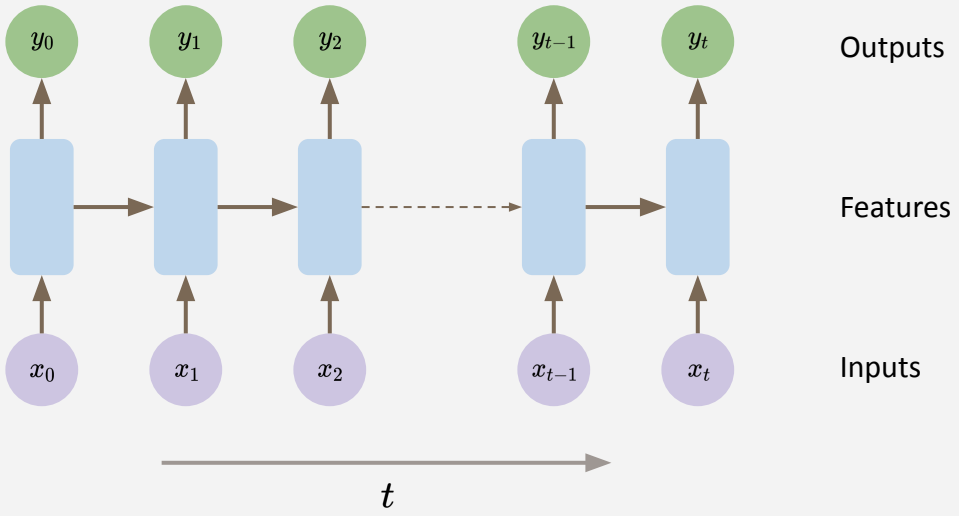
1. Multimodal robust encoding

2. Longer Memory

   Infinite reference window

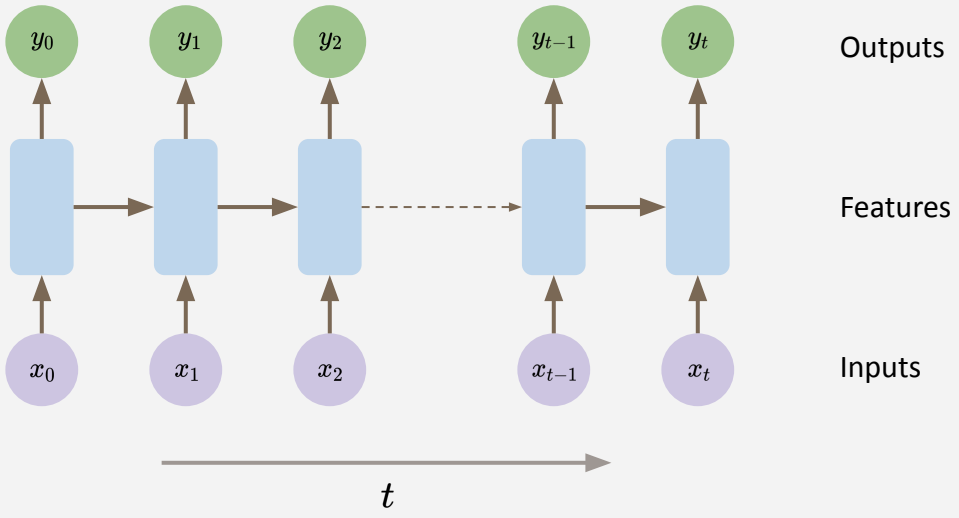
3. Parallelization

   Efficient compute



Goals of Sequence Modeling

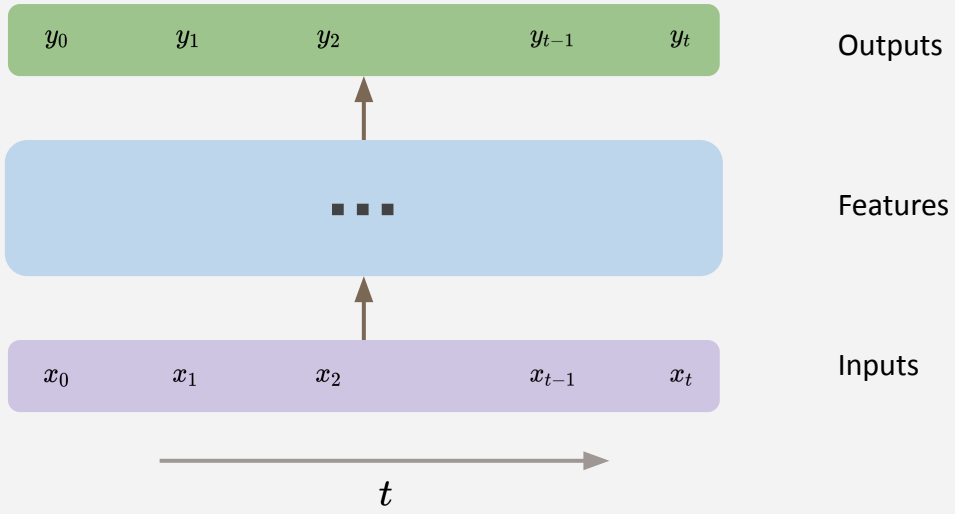
1. Multimodal robust encoding
2. Longer Memory
   Infinite reference window
3. Parallelization
   Efficient compute



What if we eliminate recurrence and accumulate sequences?

Goals of Sequence Modeling

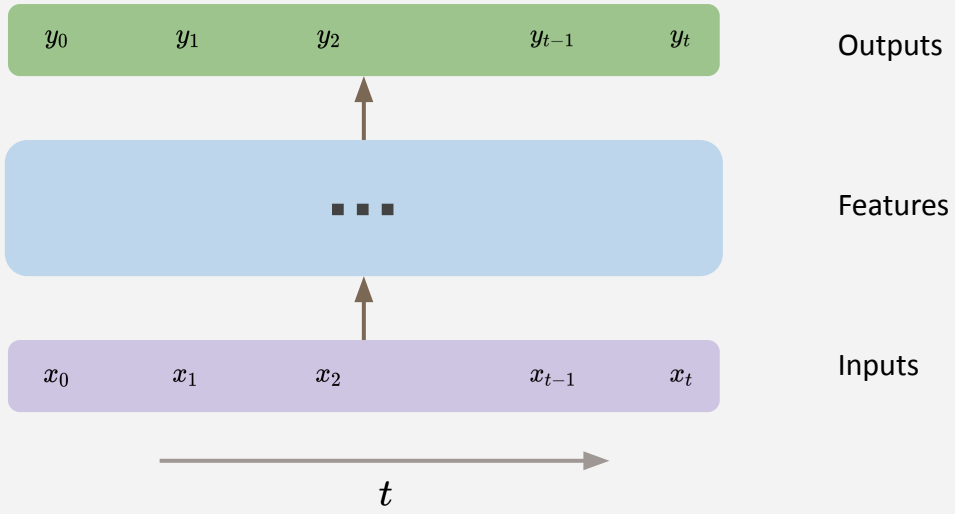
Idea: Feedforward Network



Goals of Sequence Modeling

Idea: Feedforward Network

1. **No Recurrence**
   + parallelization

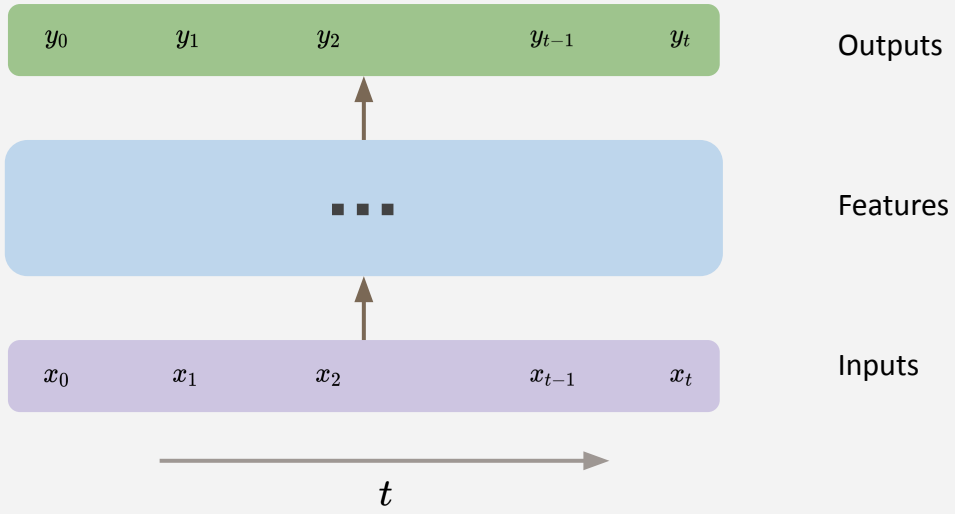


Goals of Sequence Modeling

Idea: Feedforward Network

1. **No Recurrence**
   + parallelization

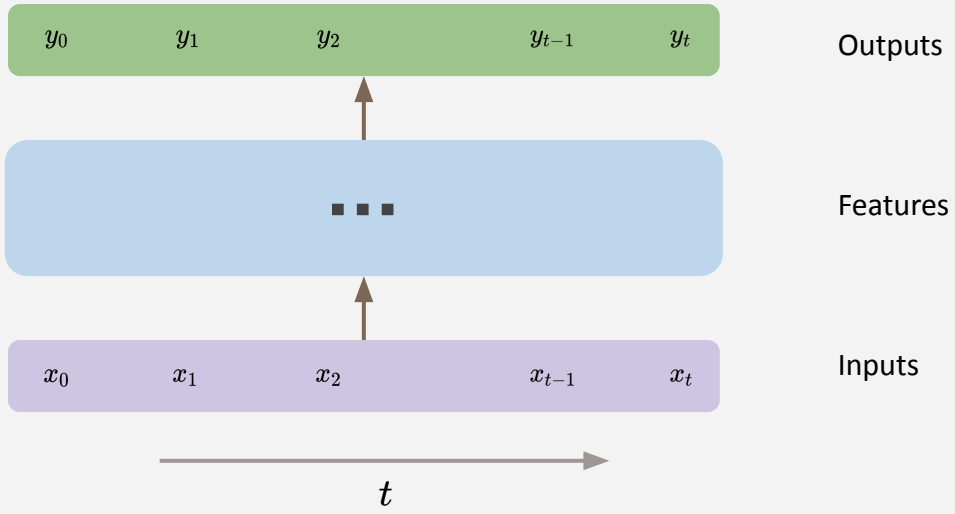
2. **No Memory**



Goals of Sequence Modeling

Idea: Feedforward Network

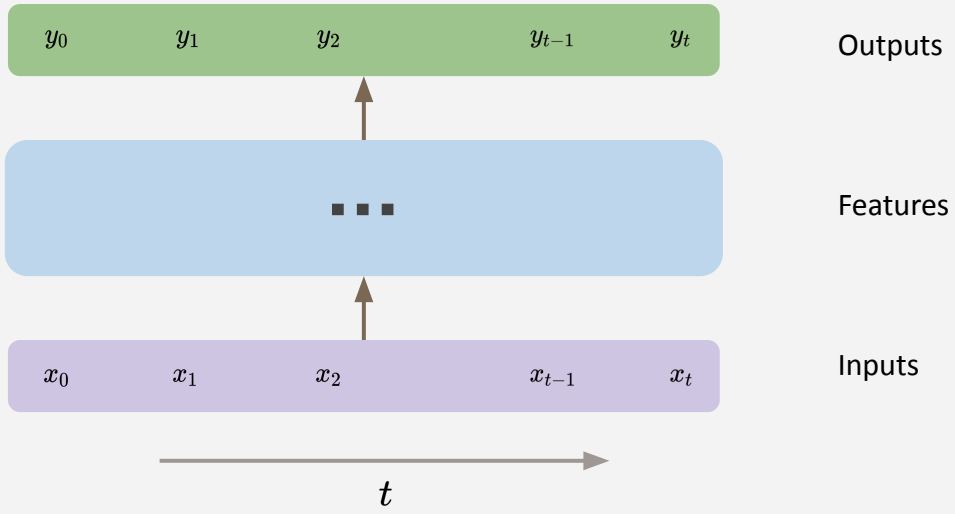
1. **No Recurrence**
   + parallelization
2. **No Memory**
3. **Not Scalable**



Goals of Sequence Modeling

Idea: Feedforward Network

1. **No Recurrence**
   + parallelization
2. **No Memory**
3. **Not Scalable**

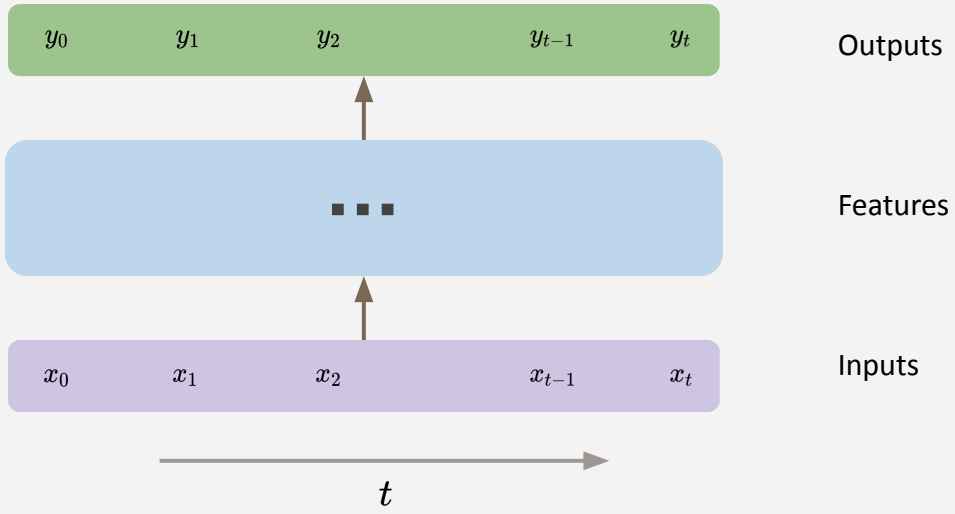


Insight Residual Connections

Goals of Sequence Modeling

Idea: Feedforward Network

1. **No Recurrence**
   + parallelization
2. **No Memory**
3. **Not Scalable**

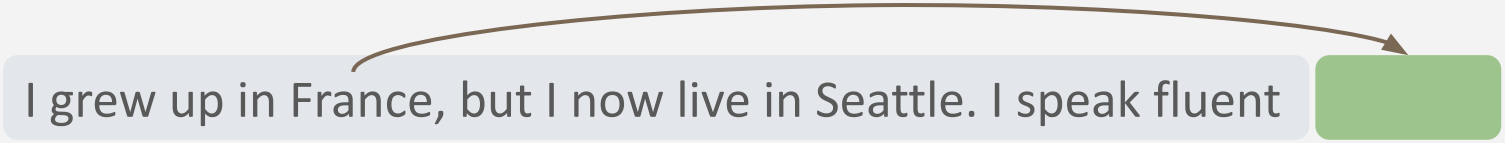


Insight Residual Connections + Attend to What is Important

Model Long-Term Dependencies

I grew up in France, but I now live in Seattle. I speak fluent 

Model Long-Term Dependencies



Attention Is All You Need

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Attention Intuition

Attend to the most important parts of an input

Attention Intuition

Attend to the most important parts of an input

1. Identify which parts to attend

2. Extract parts with high attention

3. Condition output on extracted parts

Attention Intuition

Attend to the most important parts of an input

1. Identify which parts to attend } **Search**

2. Extract parts with high attention

3. Condition output on extracted parts

Attention Intuition

Attend to the most important parts of an input

1. Identify which parts to attend } **Search**

2. Extract parts with high attention } **Feature Extraction**

3. Condition output on extracted parts

Attention Intuition

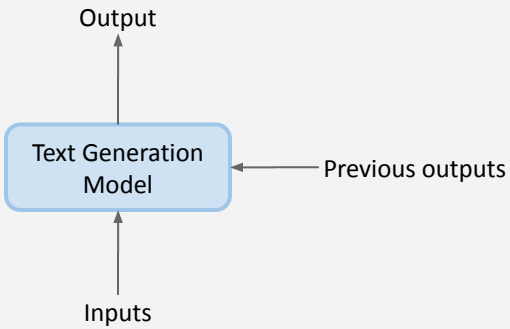
Attend to the most important parts of an input

1. Identify which parts to attend } **Search**

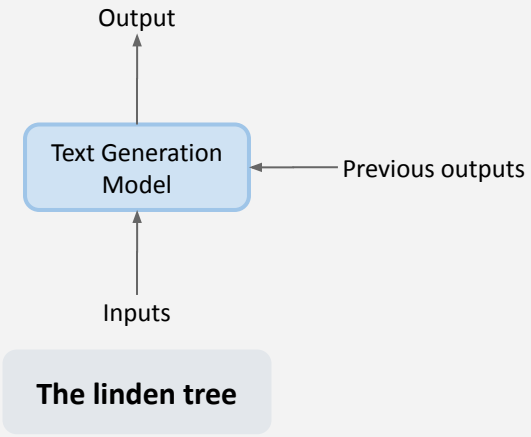
2. Extract parts with high attention } **Feature Extraction**

3. Condition output on extracted parts } **Linear Feedforward**

Attention: Text Generation



Attention: Text Generation

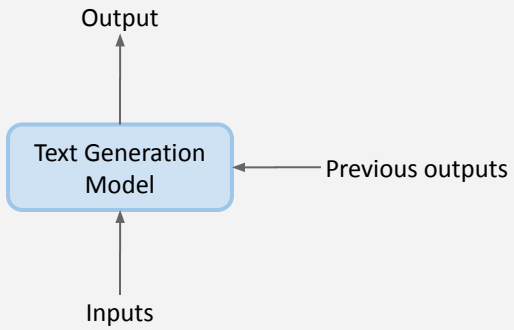


Transformer Network input

Attention: Text Generation

Transformer Network output

..seems to call to him as he passes by in the depth of night. But he turns away, into the cold wind.



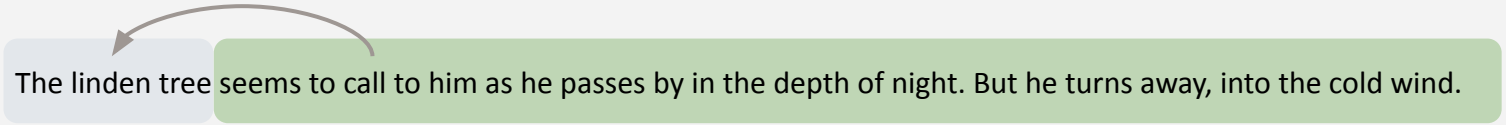
Transformer Network input

The linden tree

Attention: Text Generation

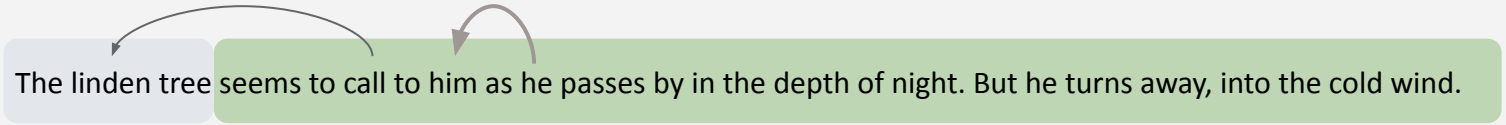
The linden tree seems to call to him as he passes by in the depth of night. But he turns away, into the cold wind.

Attention: Text Generation



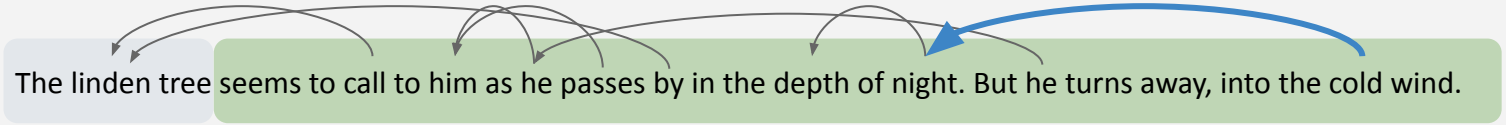
Words in the generated output sequence learn to reference, or attend to, previously generated words

Attention: Text Generation



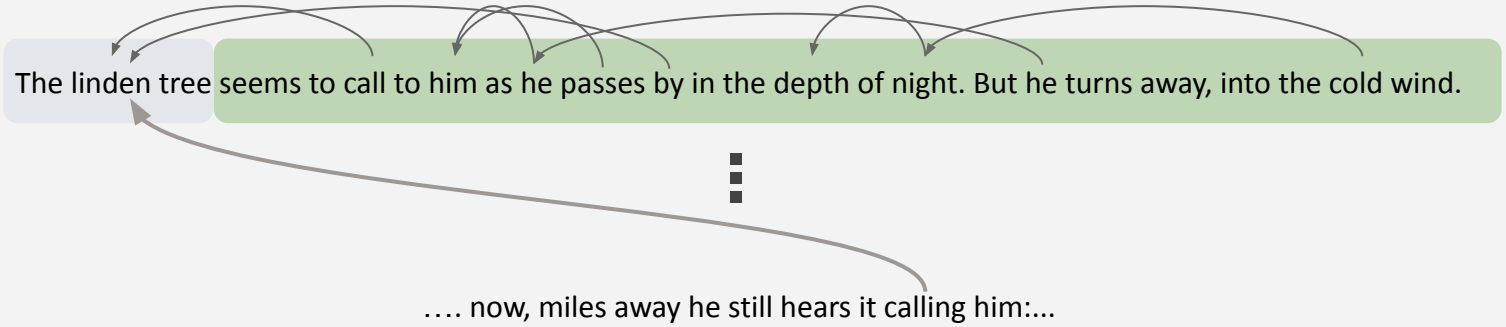
Words in the generated output sequence learn to reference, or attend to, previously generated words

Attention: Text Generation



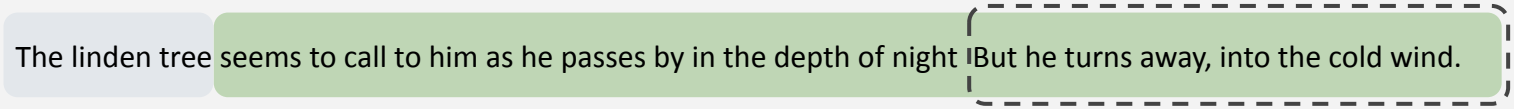
Words in the generated output sequence learn to reference, or attend to, previously generated words

Attention: Text Generation



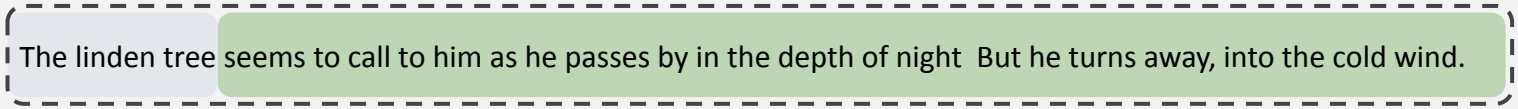
Words in the generated output sequence learn to reference, or attend to, previously generated words

Attention: Text Generation



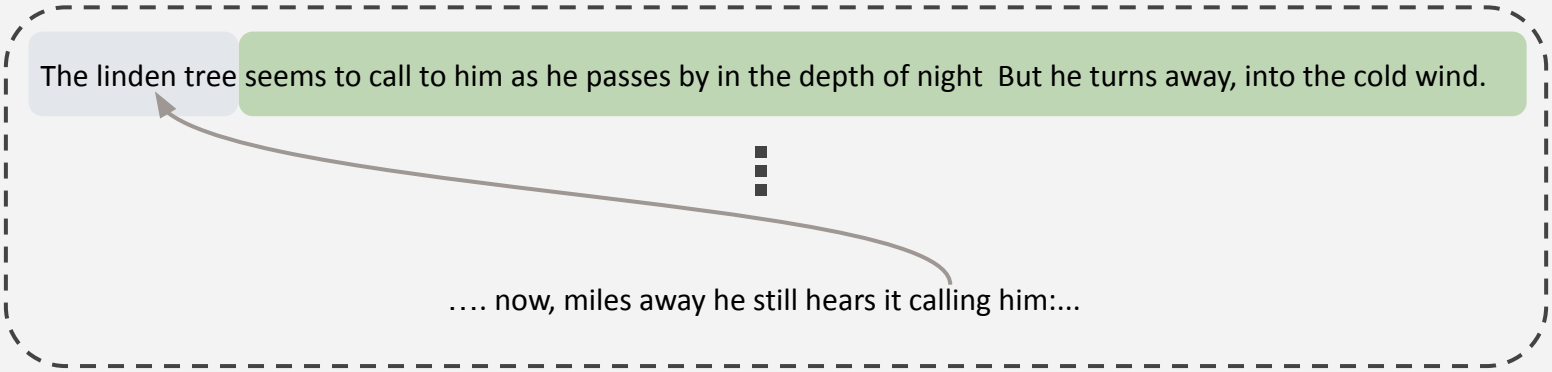
Recurrent Neural Networks have finite (and small) persistence

Attention: Text Generation



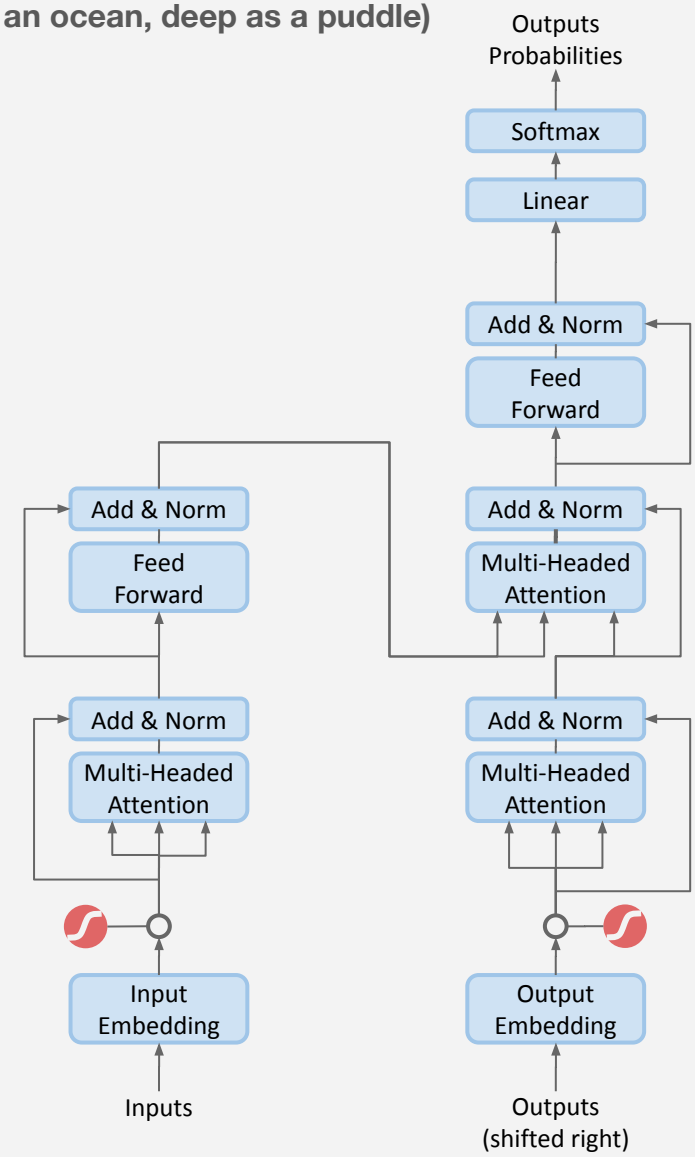
LSTMs have longer persistence

Attention: Text Generation



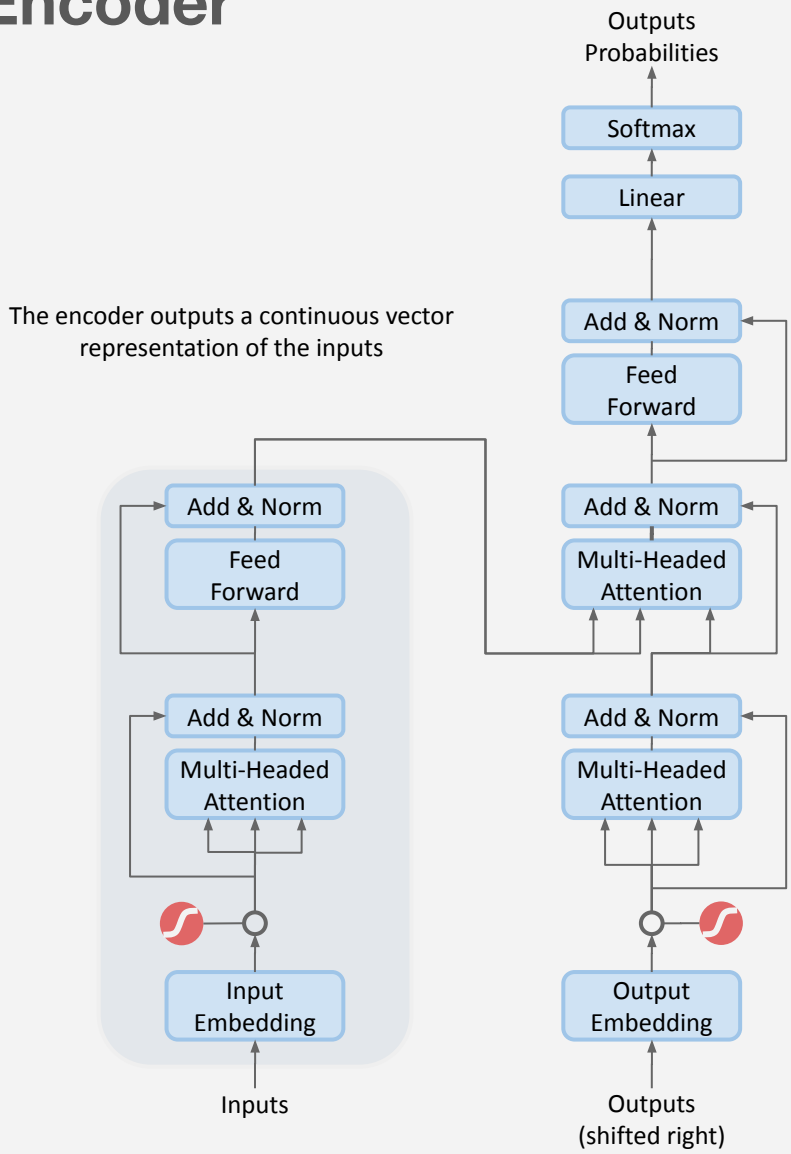
Attention mechanisms can provide infinite persistence (in theory)

Transformer (Wide as an ocean, deep as a puddle)



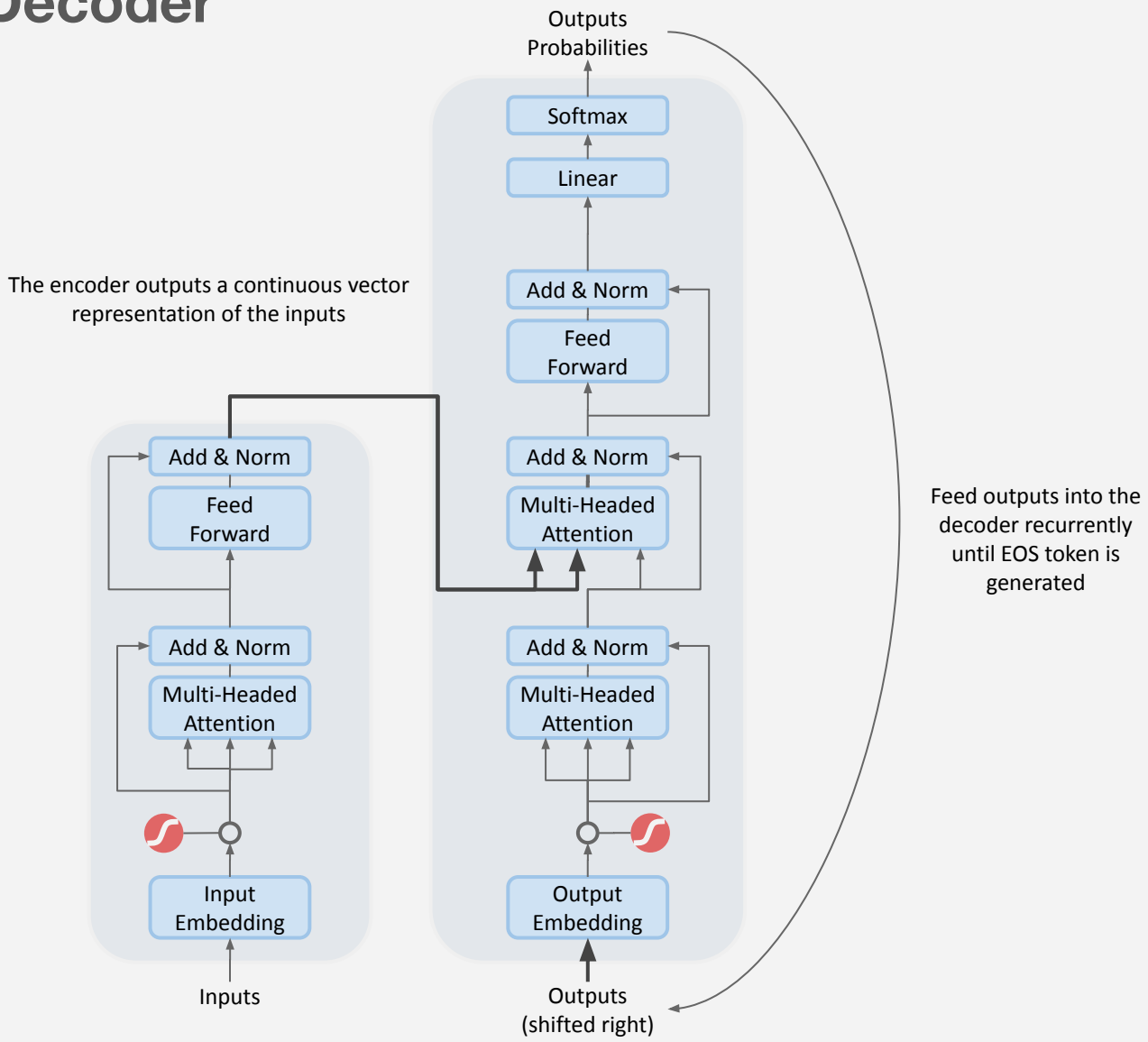
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer: Encoder



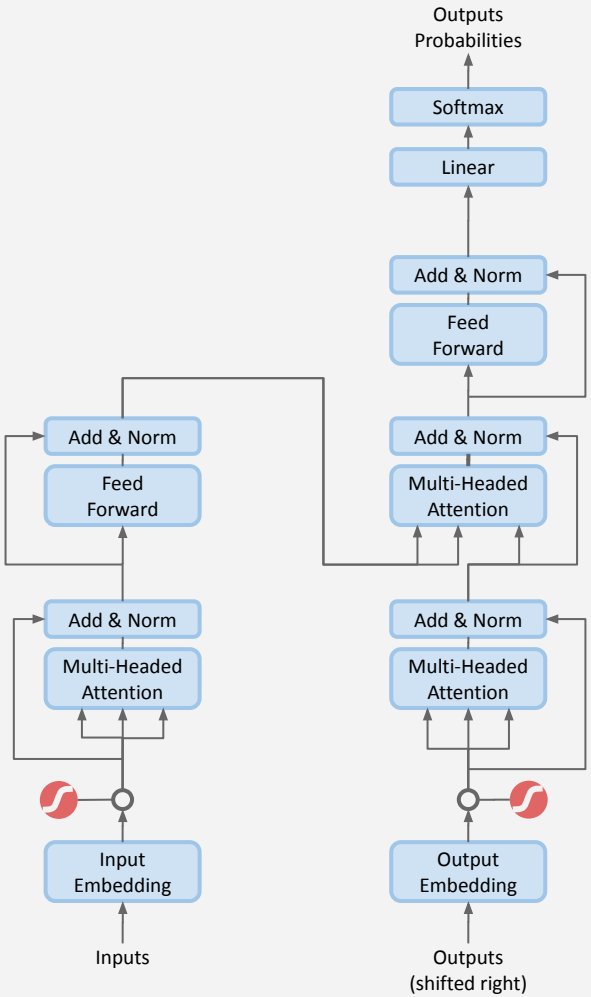
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer: Decoder

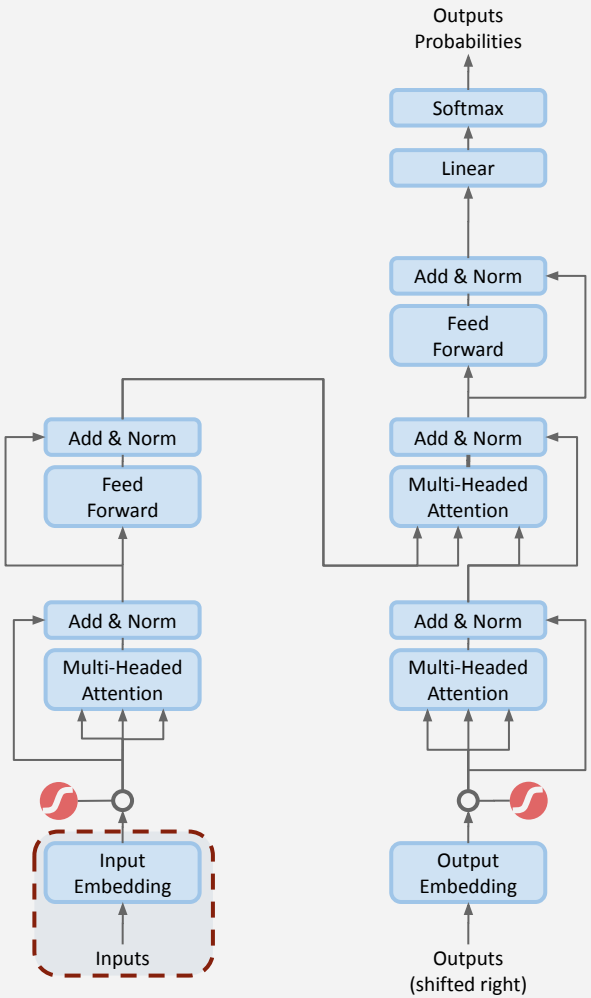


Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer

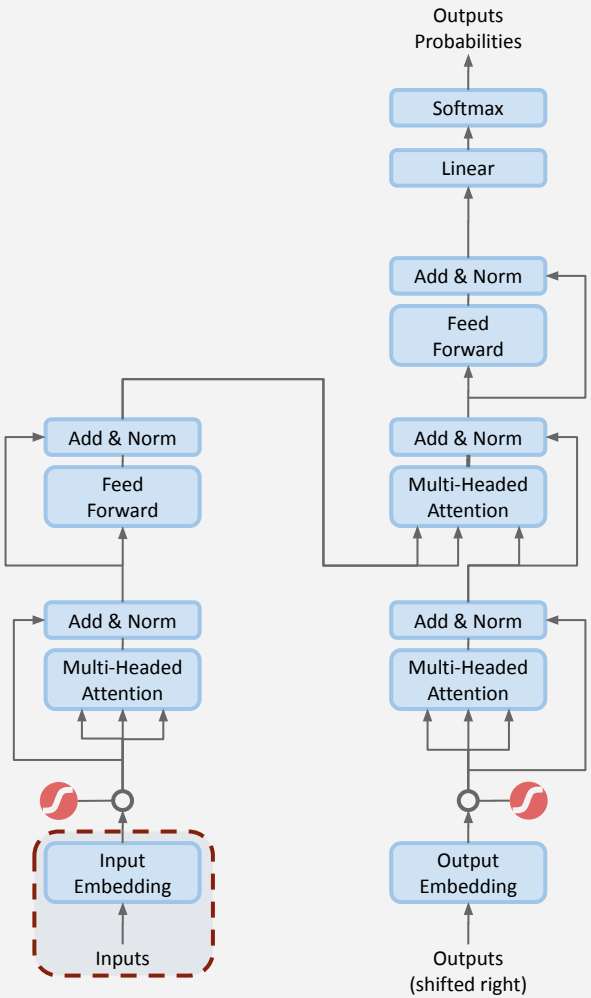


Input Embedding



Input Embedding

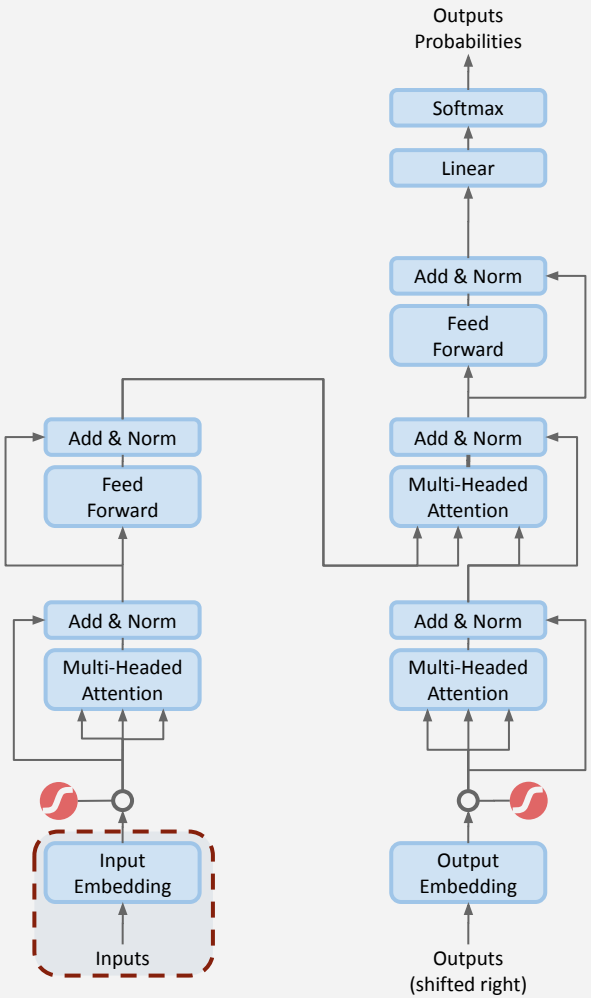
Input Embedding

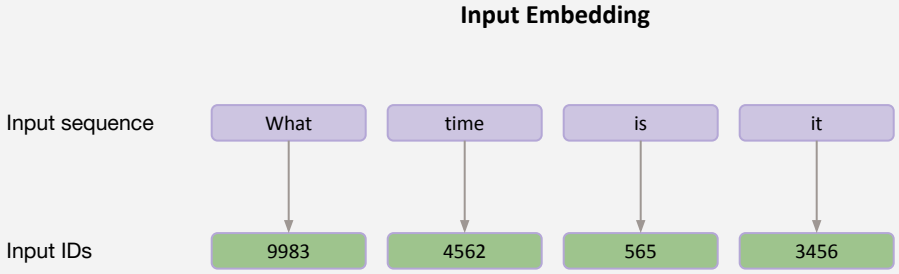


Input Embedding

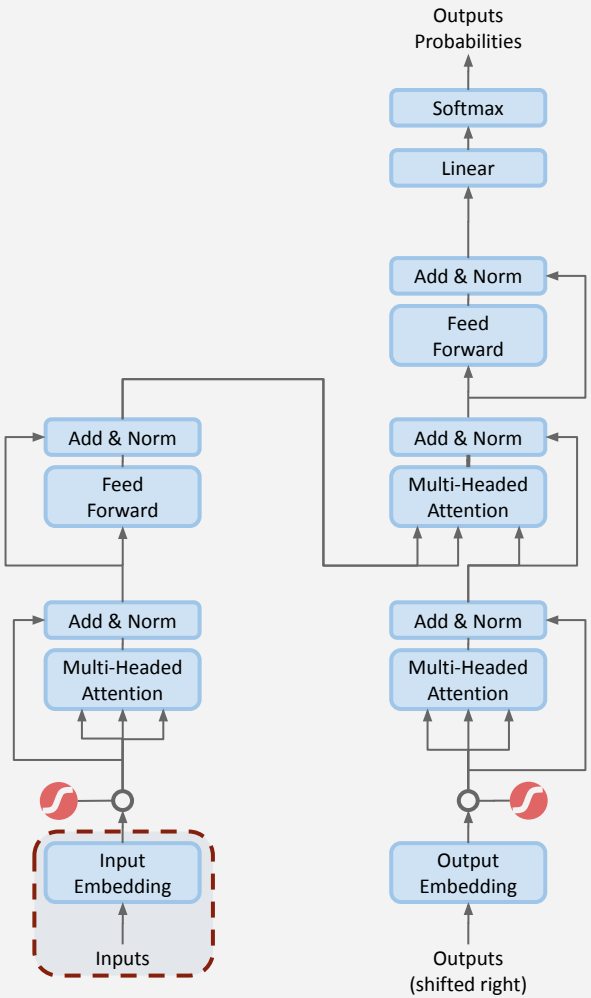
Input sequence | What | time | is | it

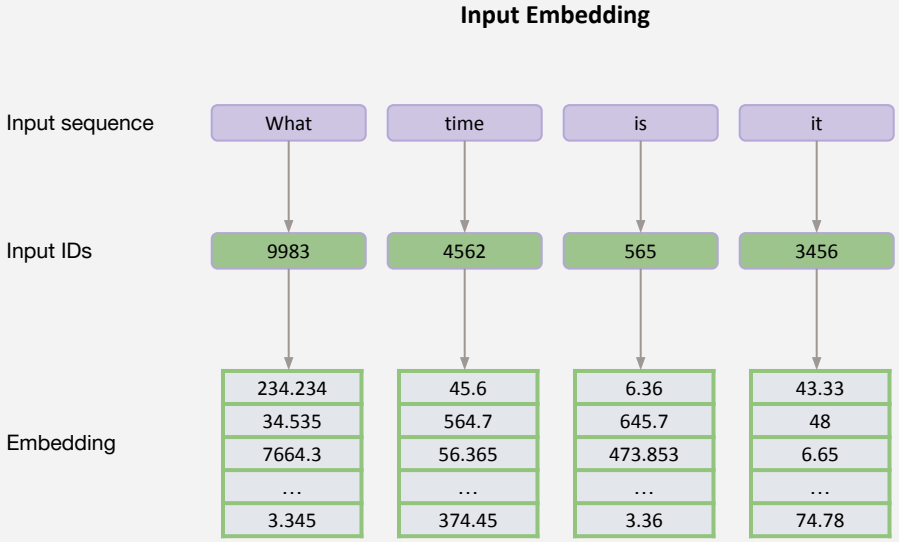
Input Embedding





Input Embedding





The input embedding is a vector of continuous values that maps input IDs (ids from a pool of vocabulary) to an embedding space with richer representation.

The values will be updated according to the loss function:
words that are semantically related will be closer in the embedding space

Positional Encoding

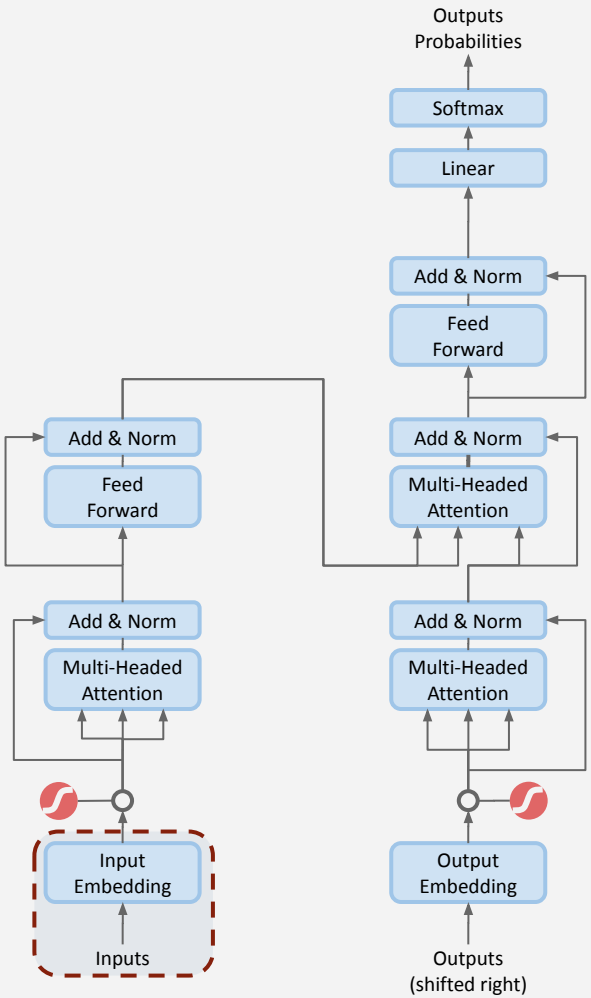
1. Encoding should retain information about positional dependencies

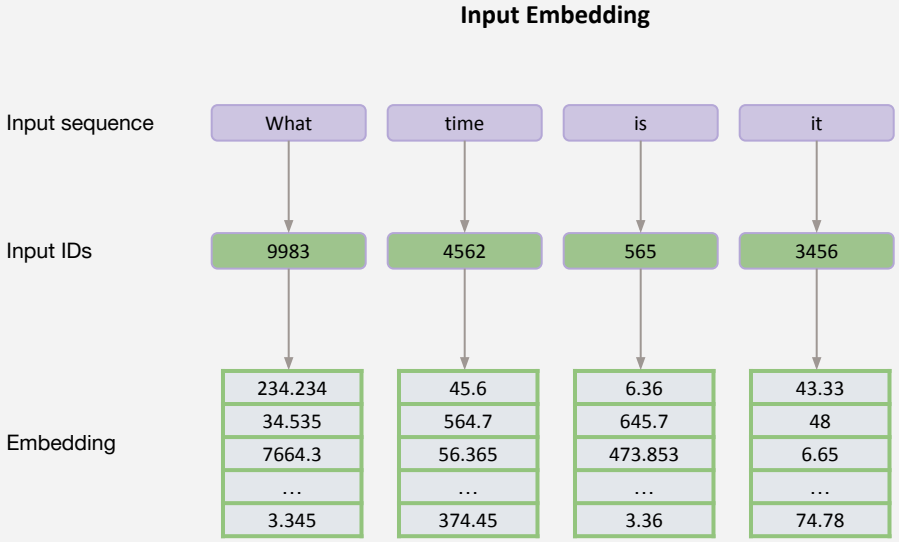
2. **Words must enforce semantic (contextual) relationships**

   Relative positions in sentences

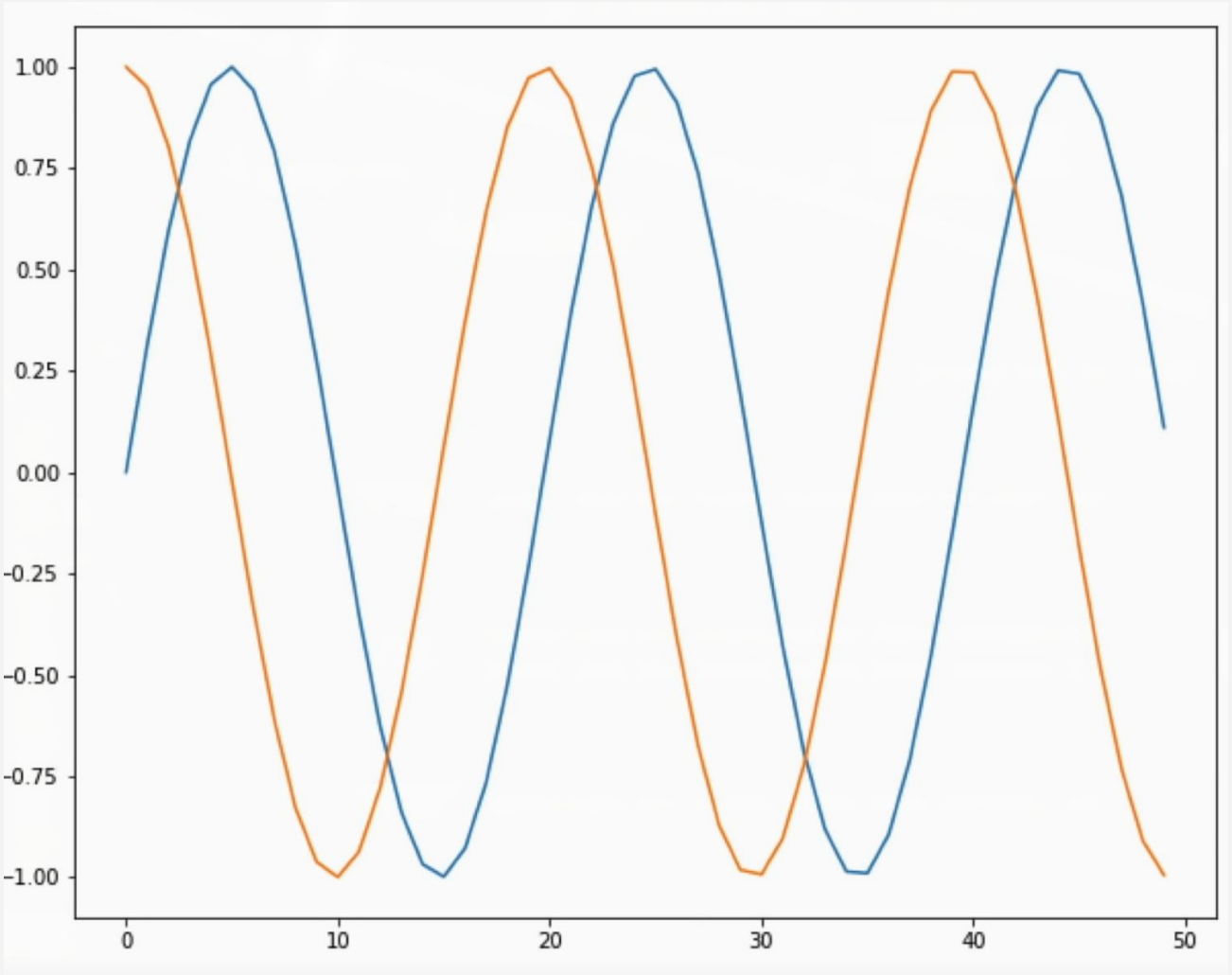
3. We want our model to learn this positional information

Input Embedding

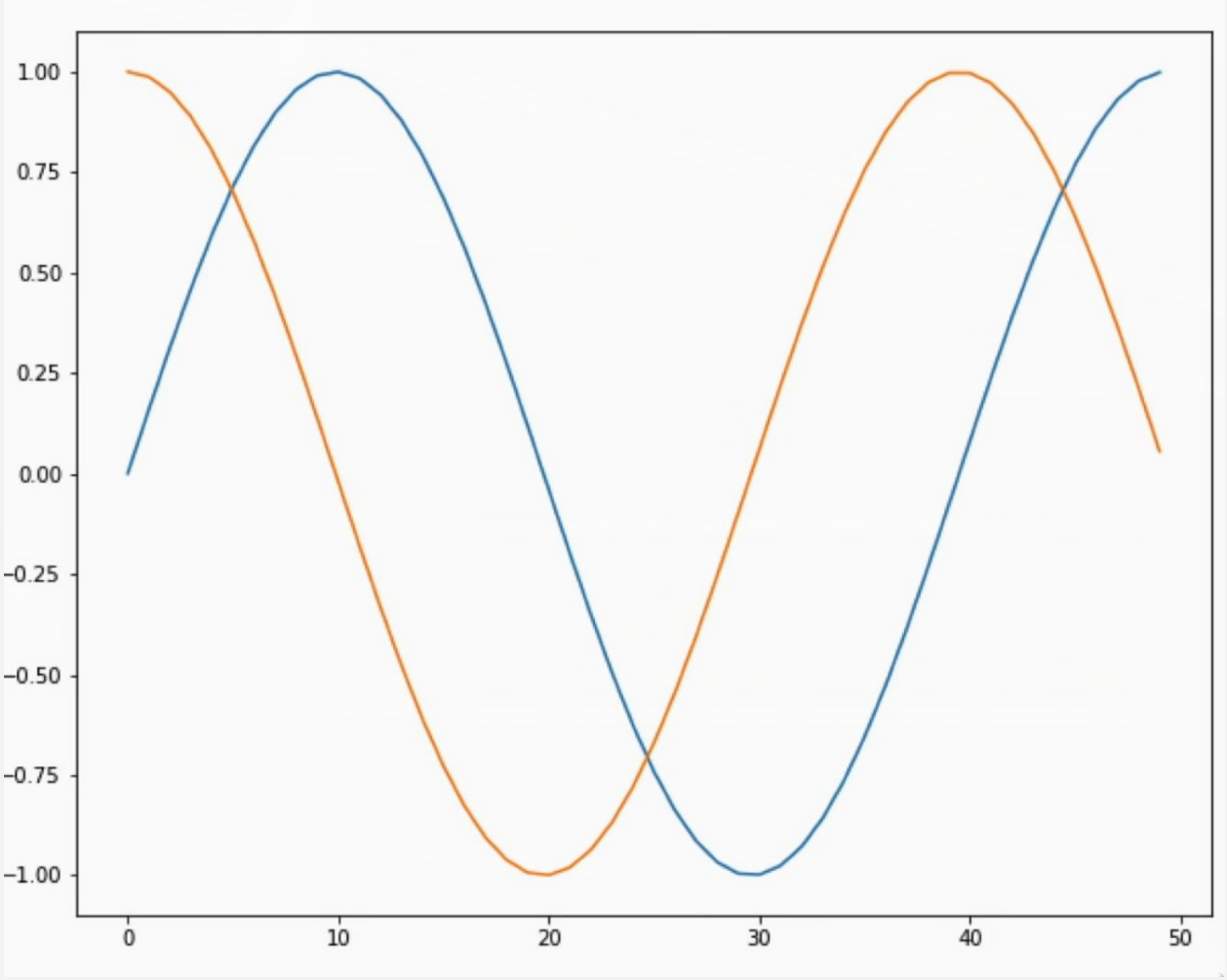




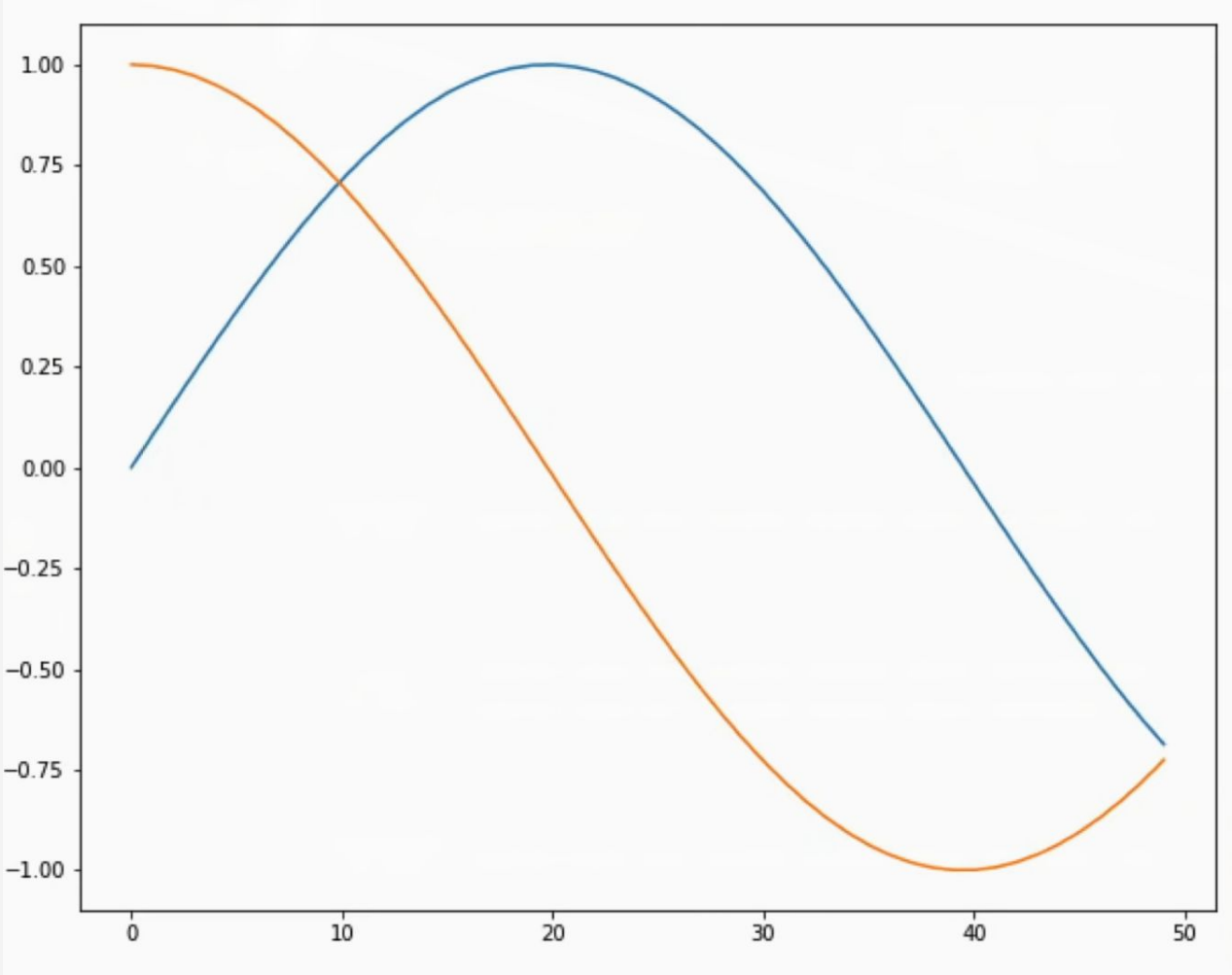
Positional Embedding



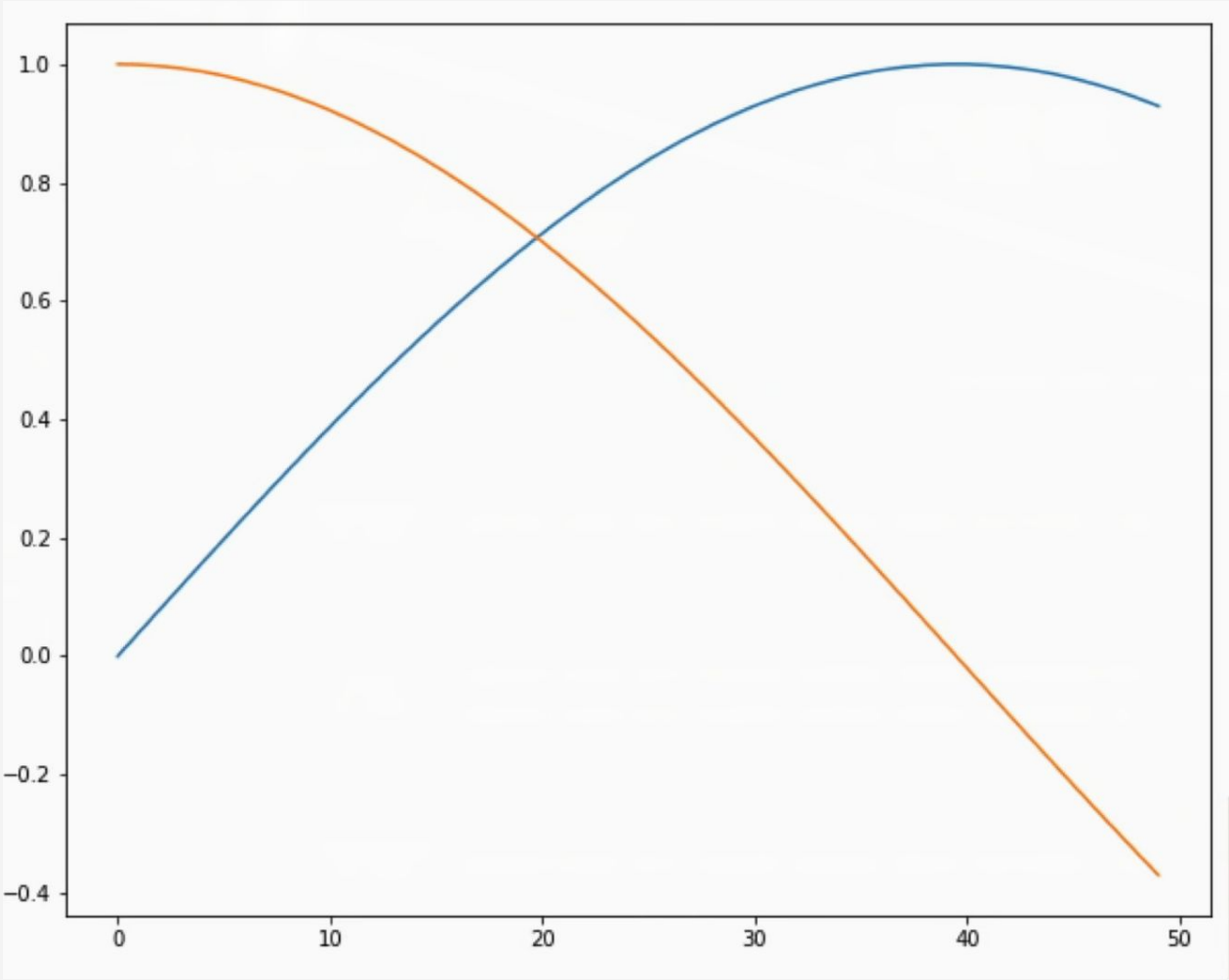
Positional Embedding



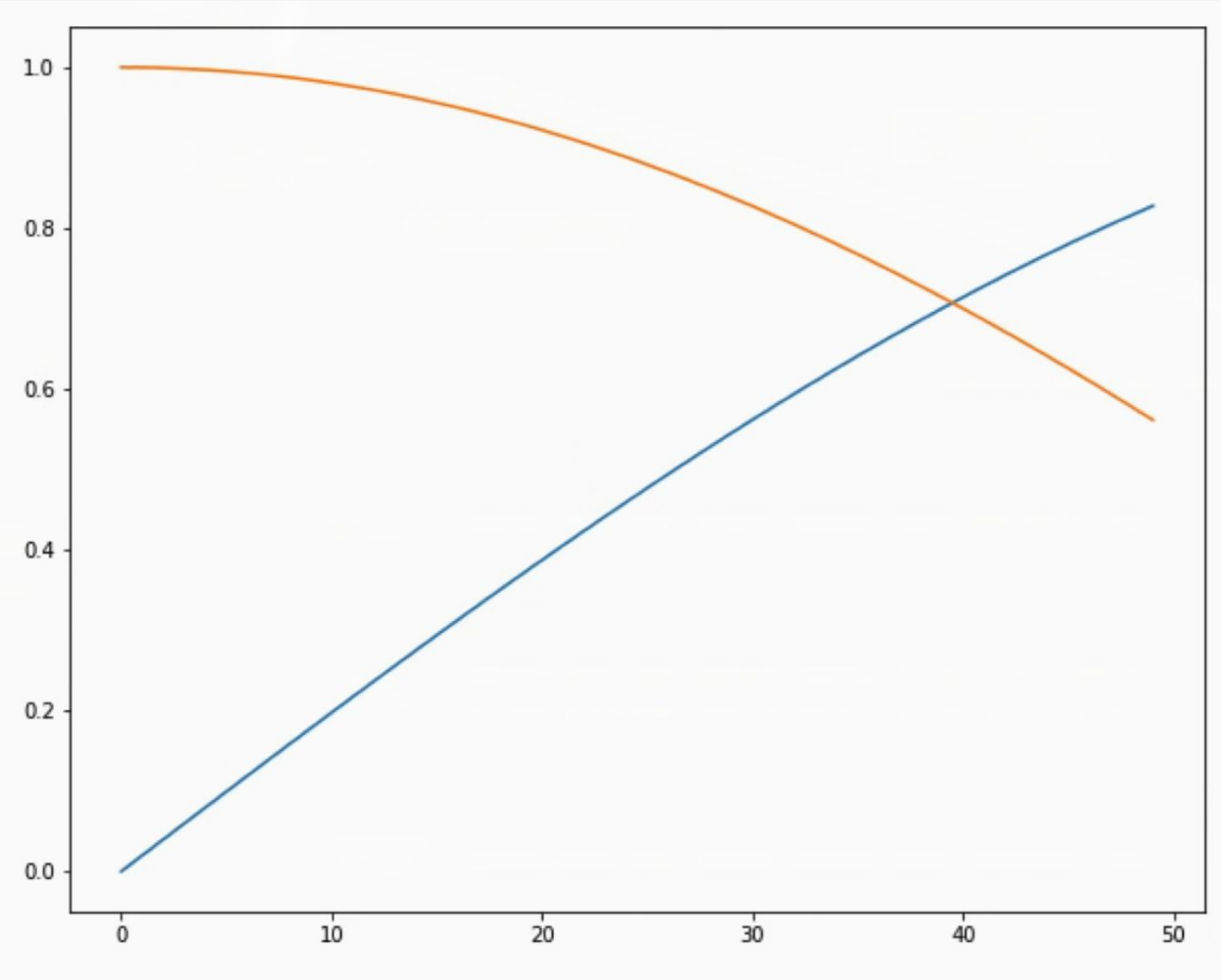
Positional Embedding



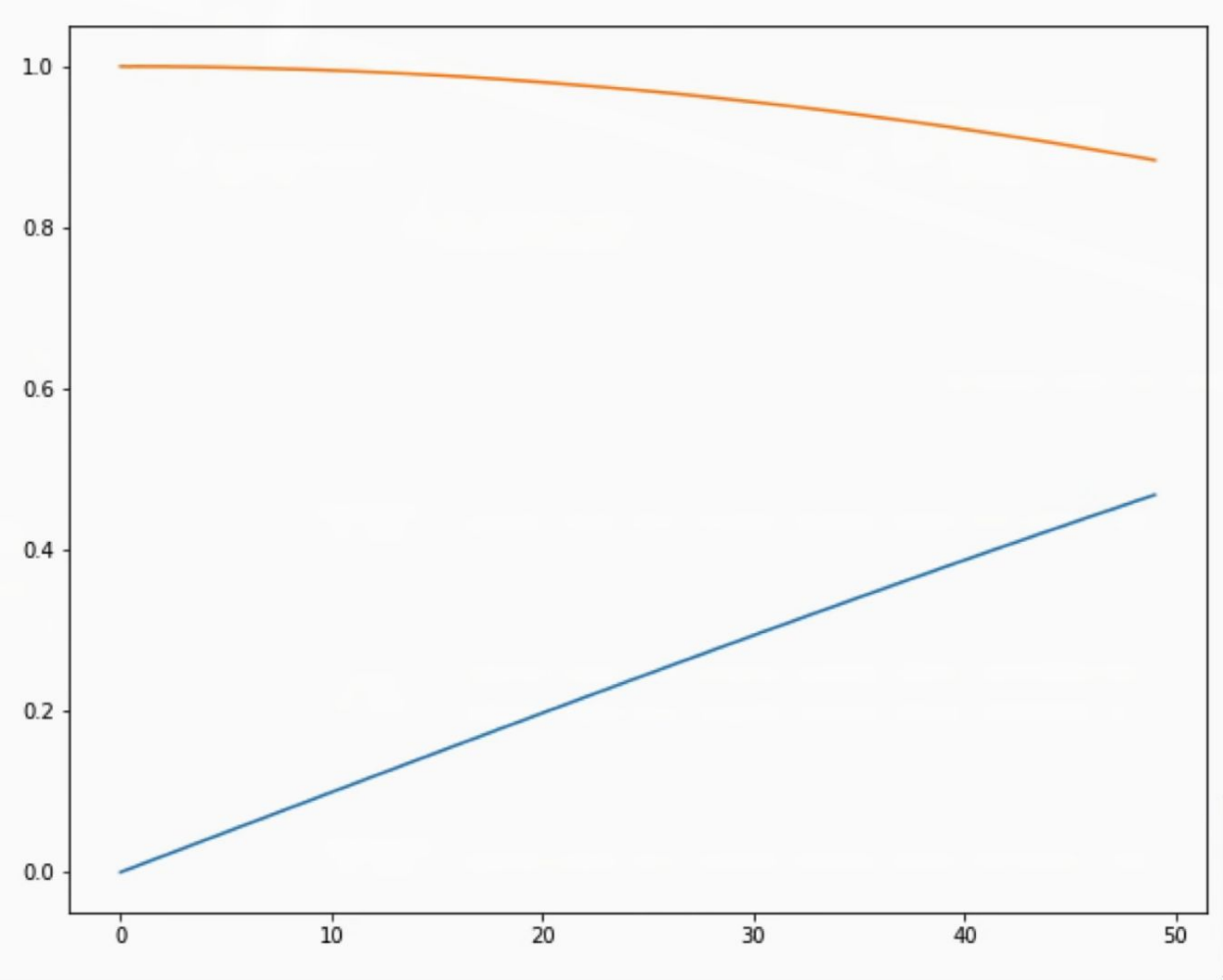
Positional Embedding



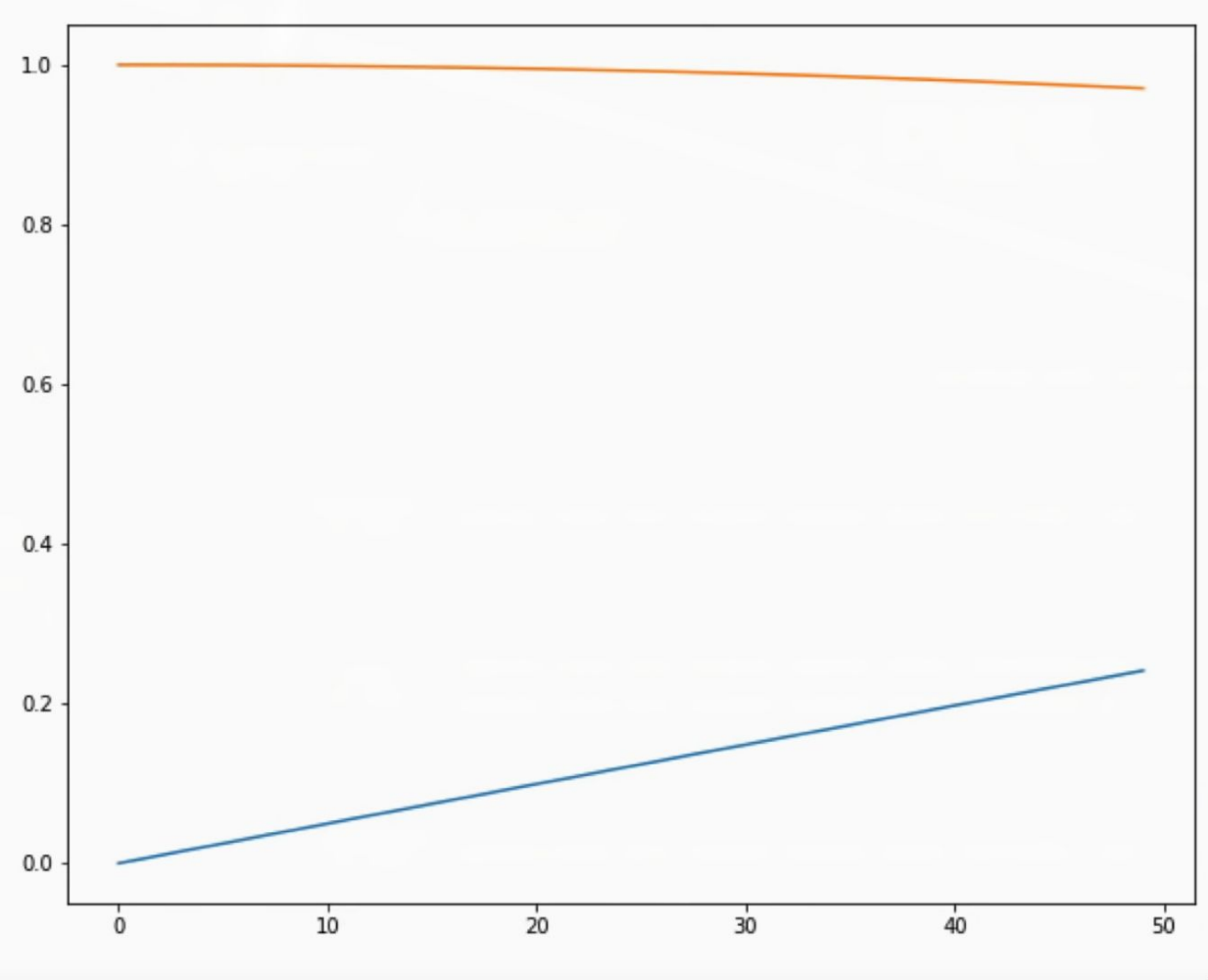
Positional Embedding



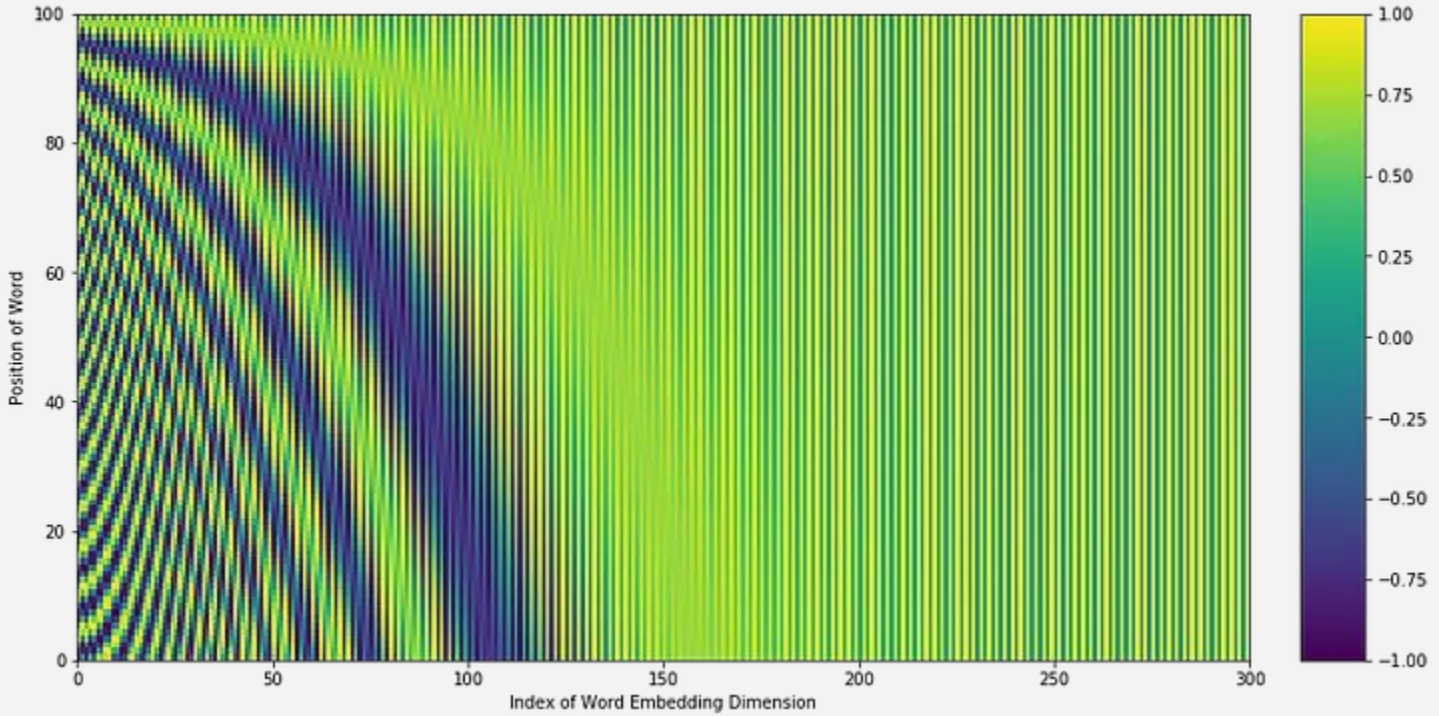
Positional Embedding



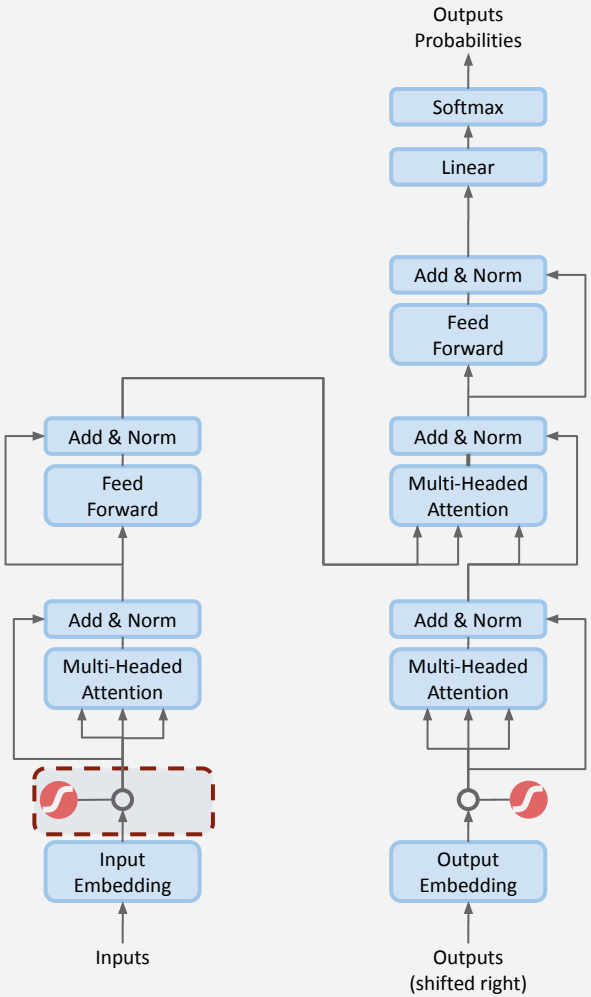
Positional Embedding

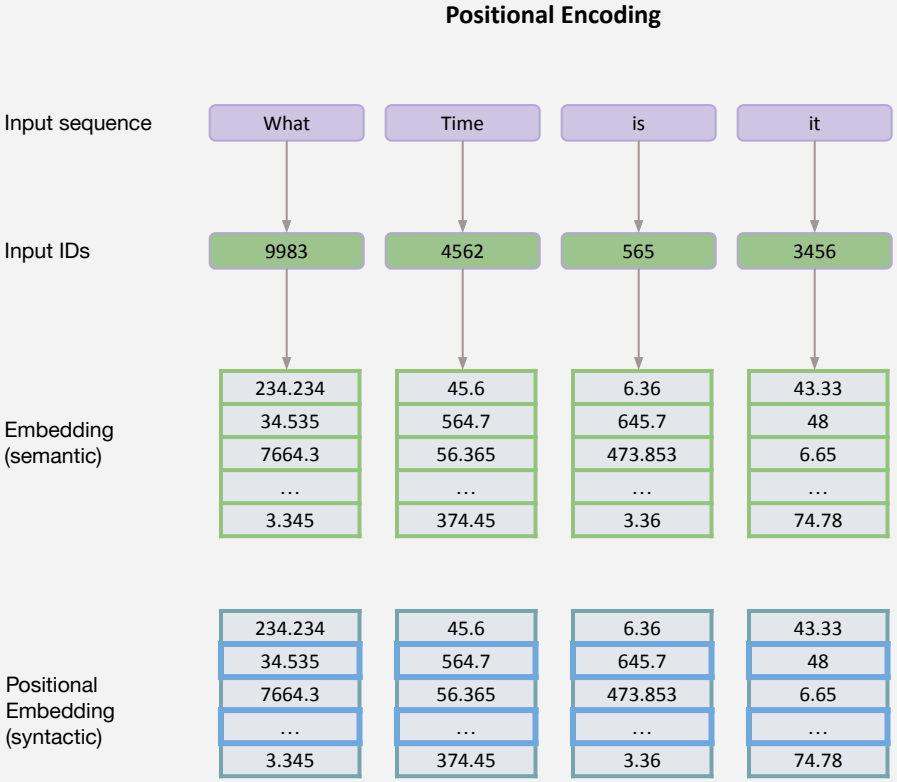


Positional Embedding

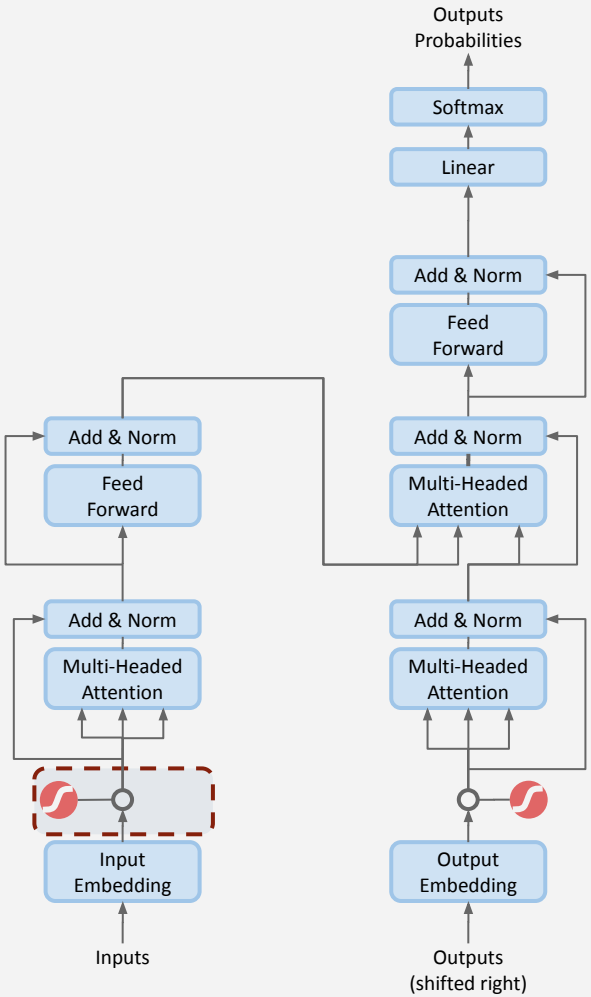


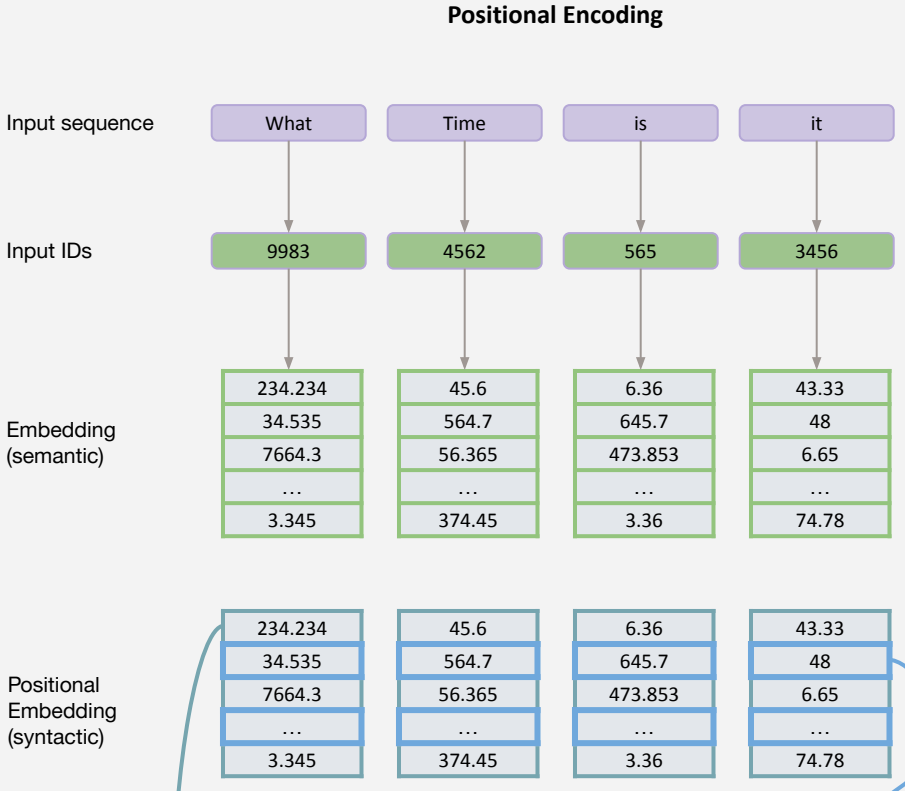
Positional Encoding





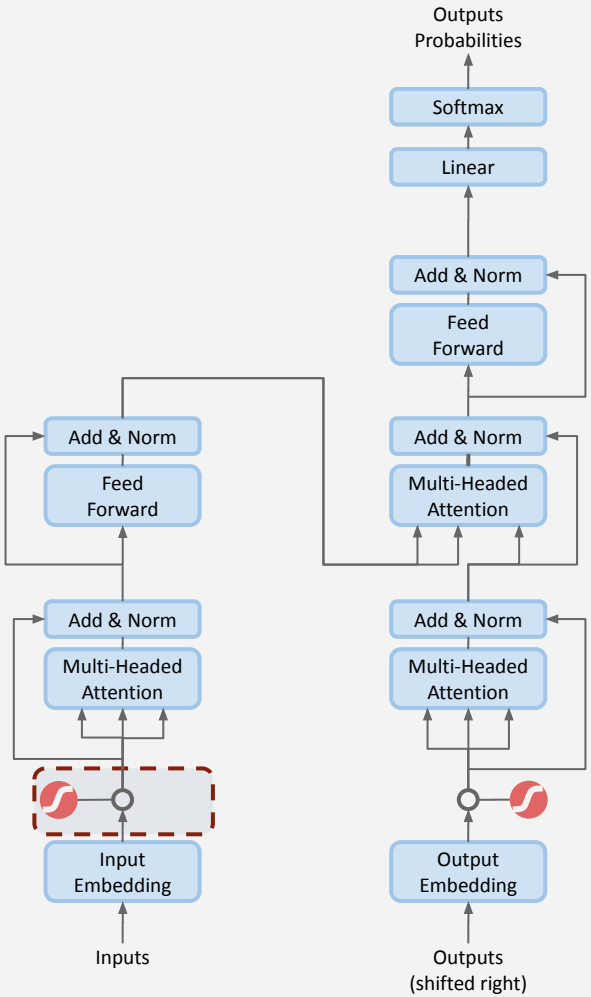
Positional Encoding

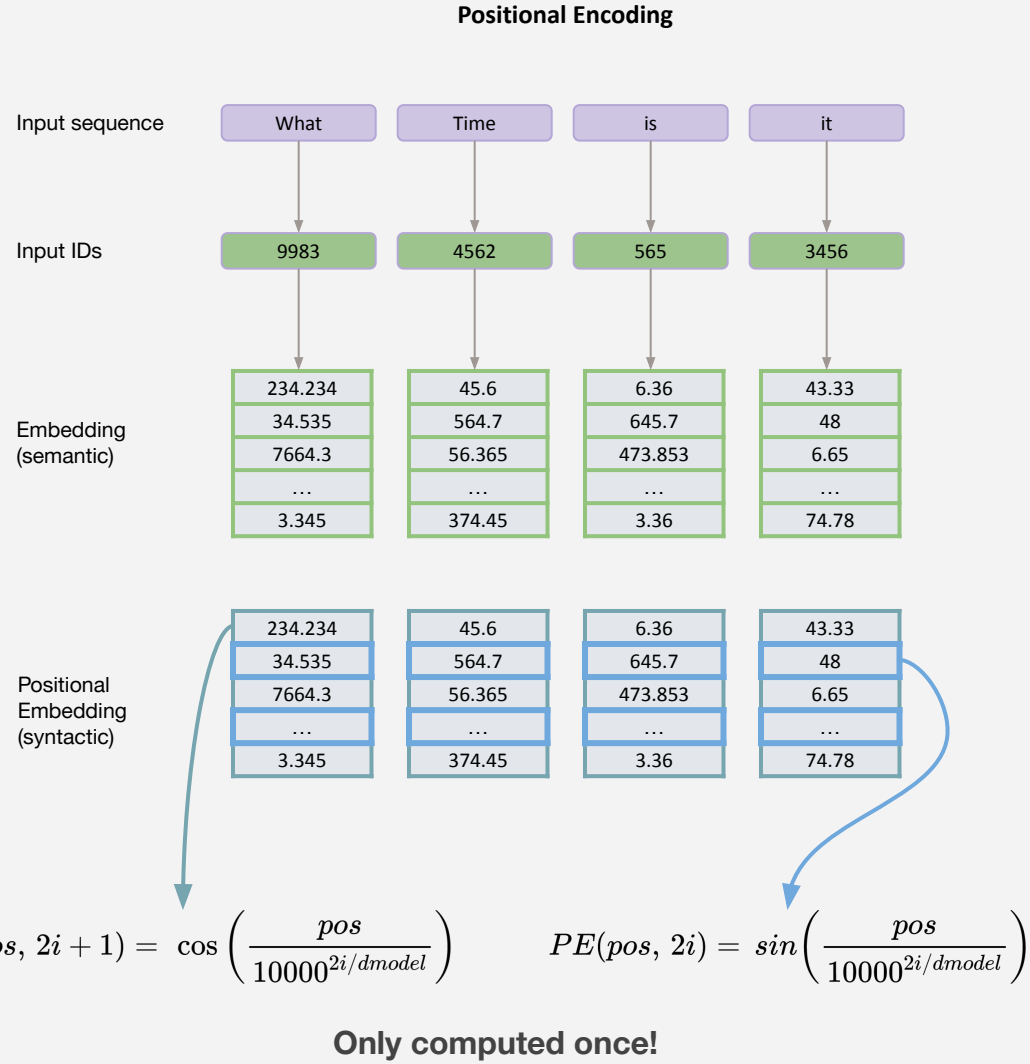




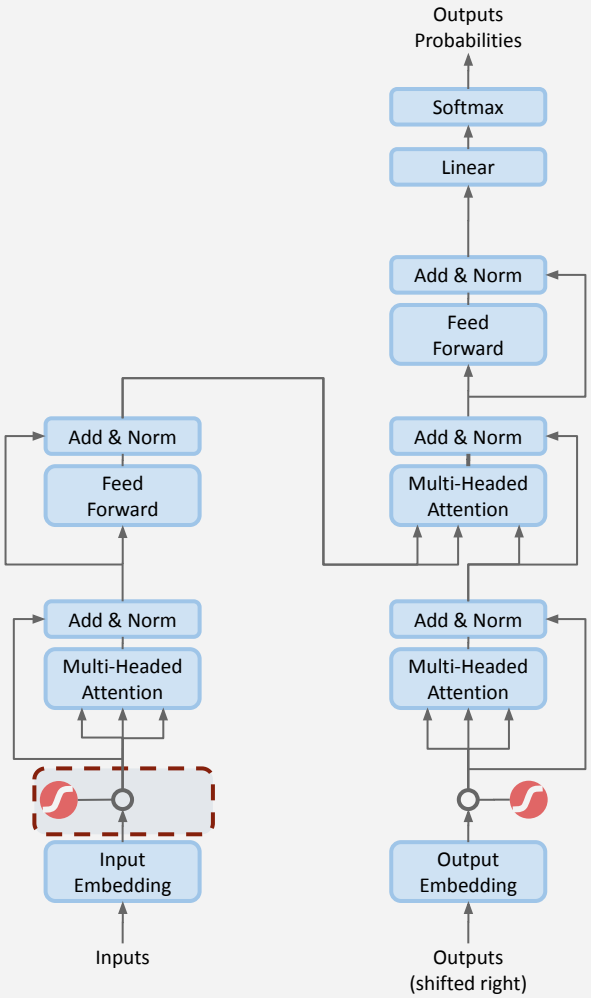
$$PE(pos,\ 2i+1) = \cos\left(\frac{pos}{10000^{2i/dmodel}}\right) \qquad PE(pos,\ 2i) = sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

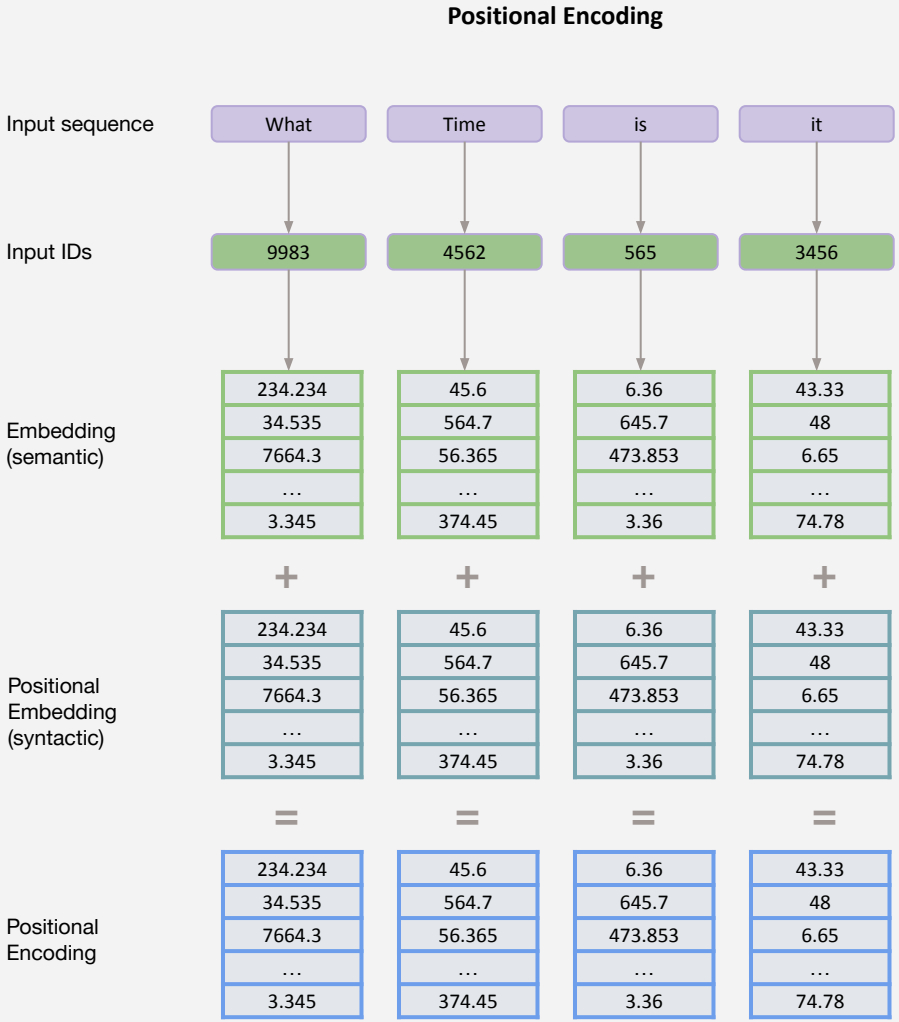
Positional Encoding



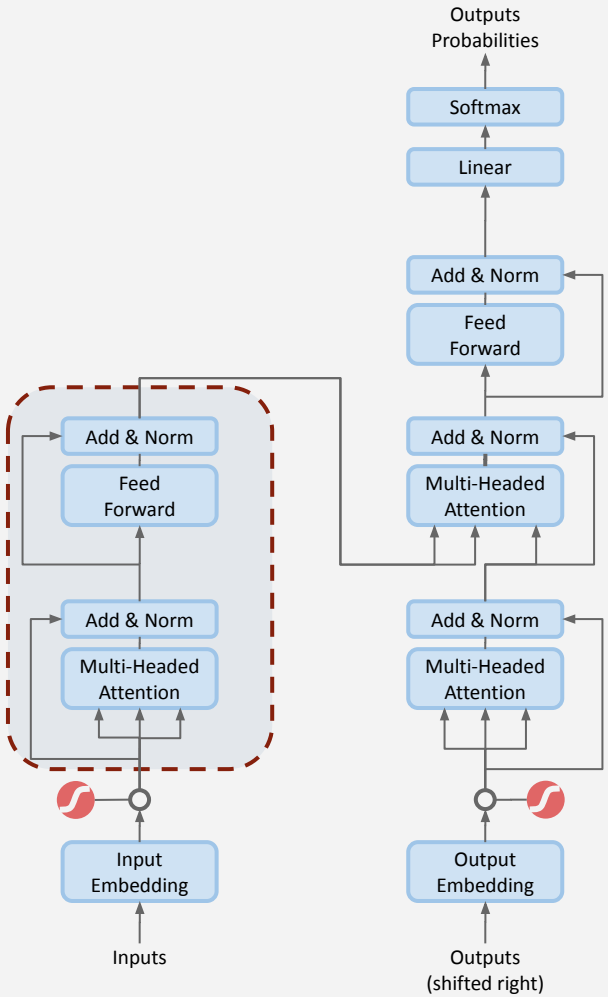


Positional Encoding

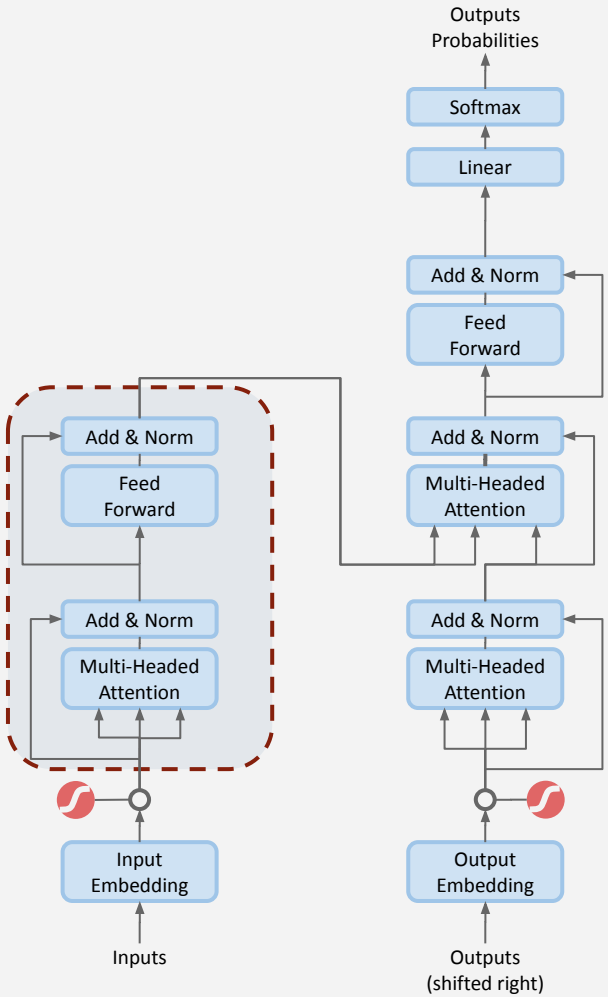


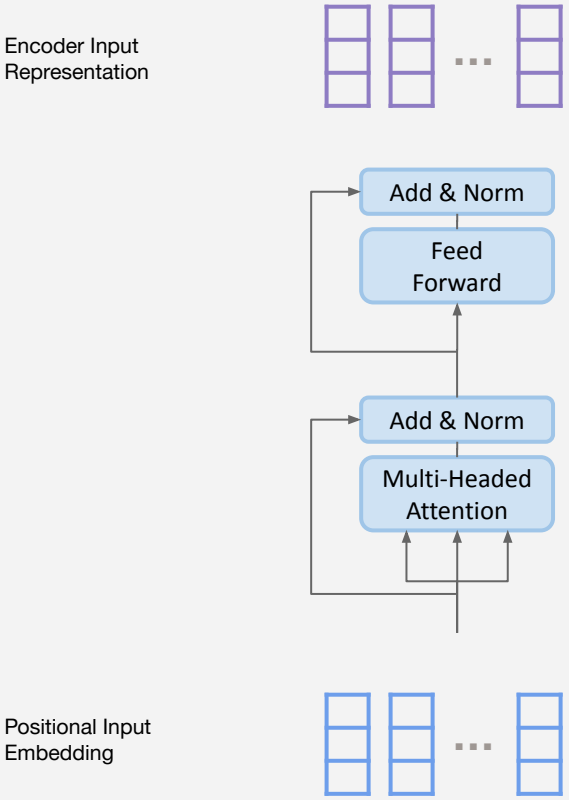


Encoder Layer

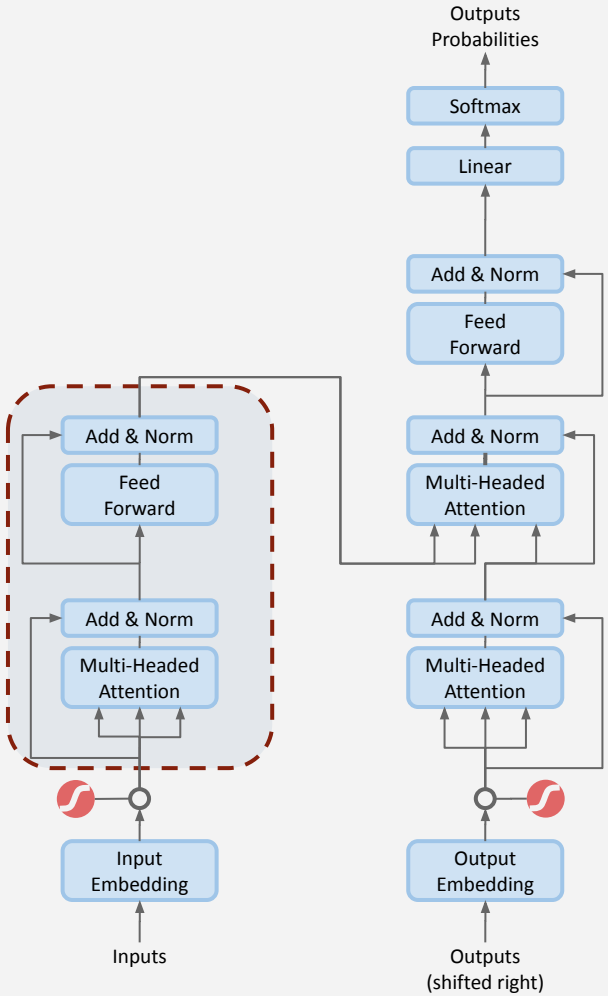


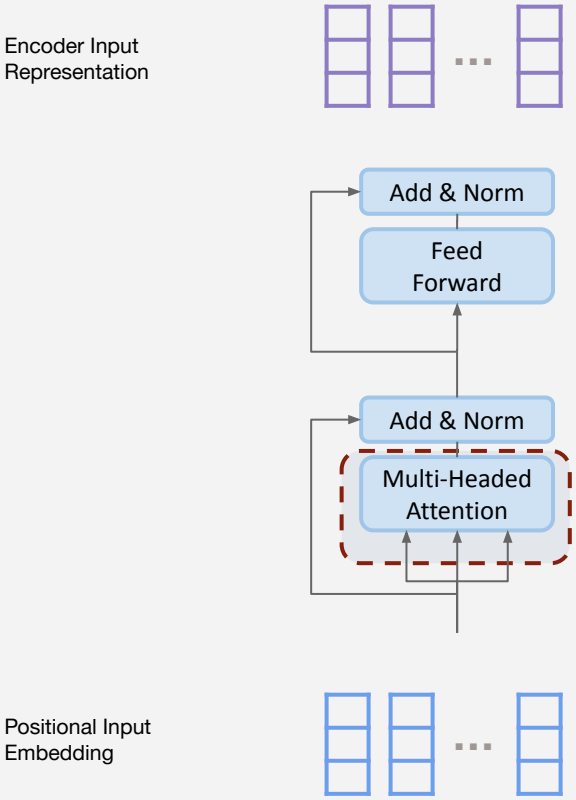
Encoder Layer



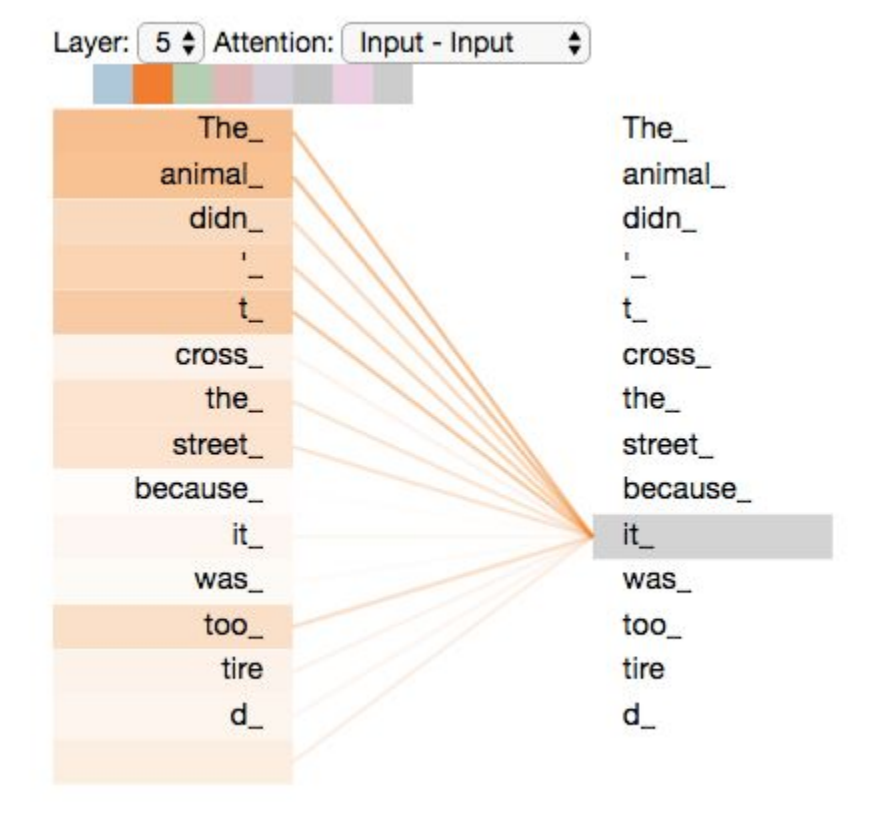


Encoder Layer



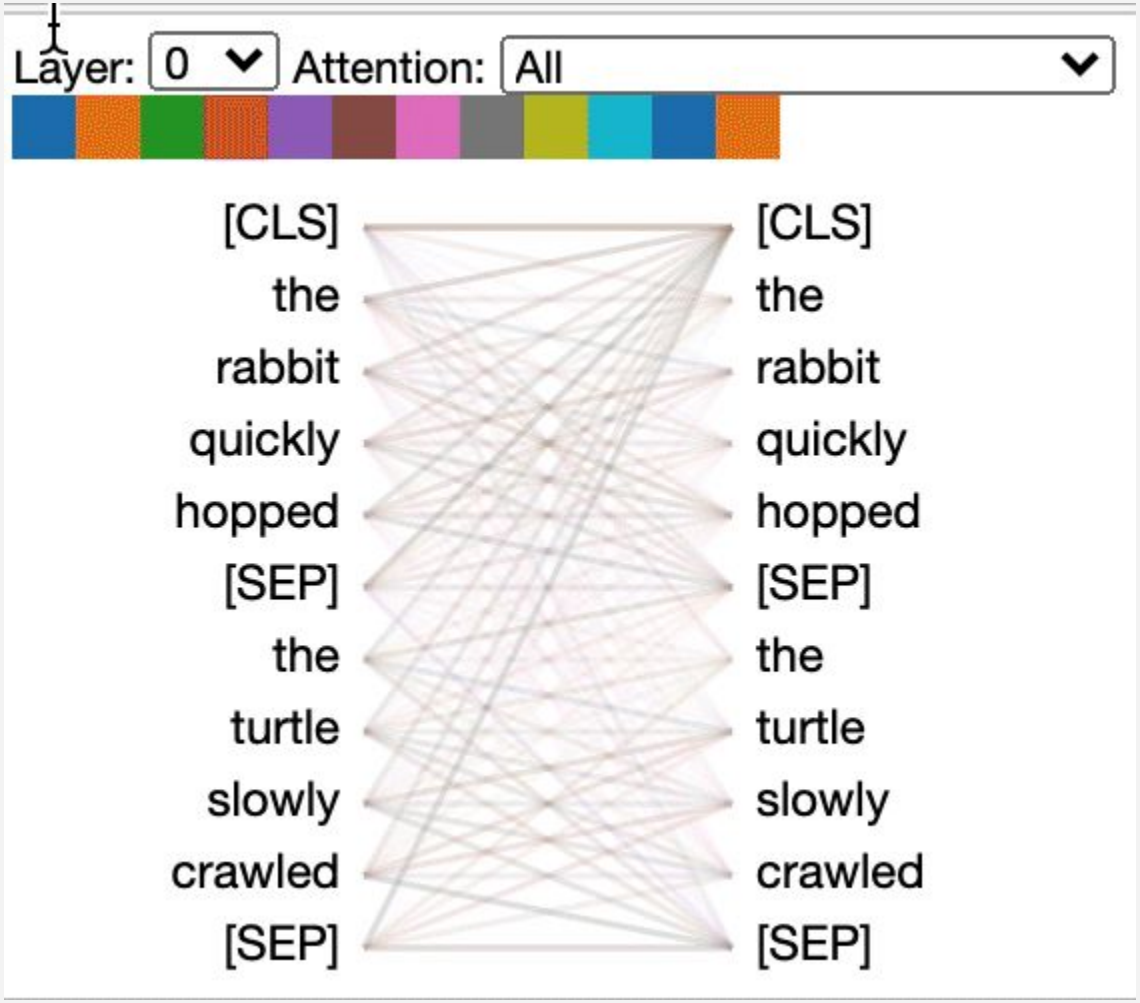


Self Attention (BERT)



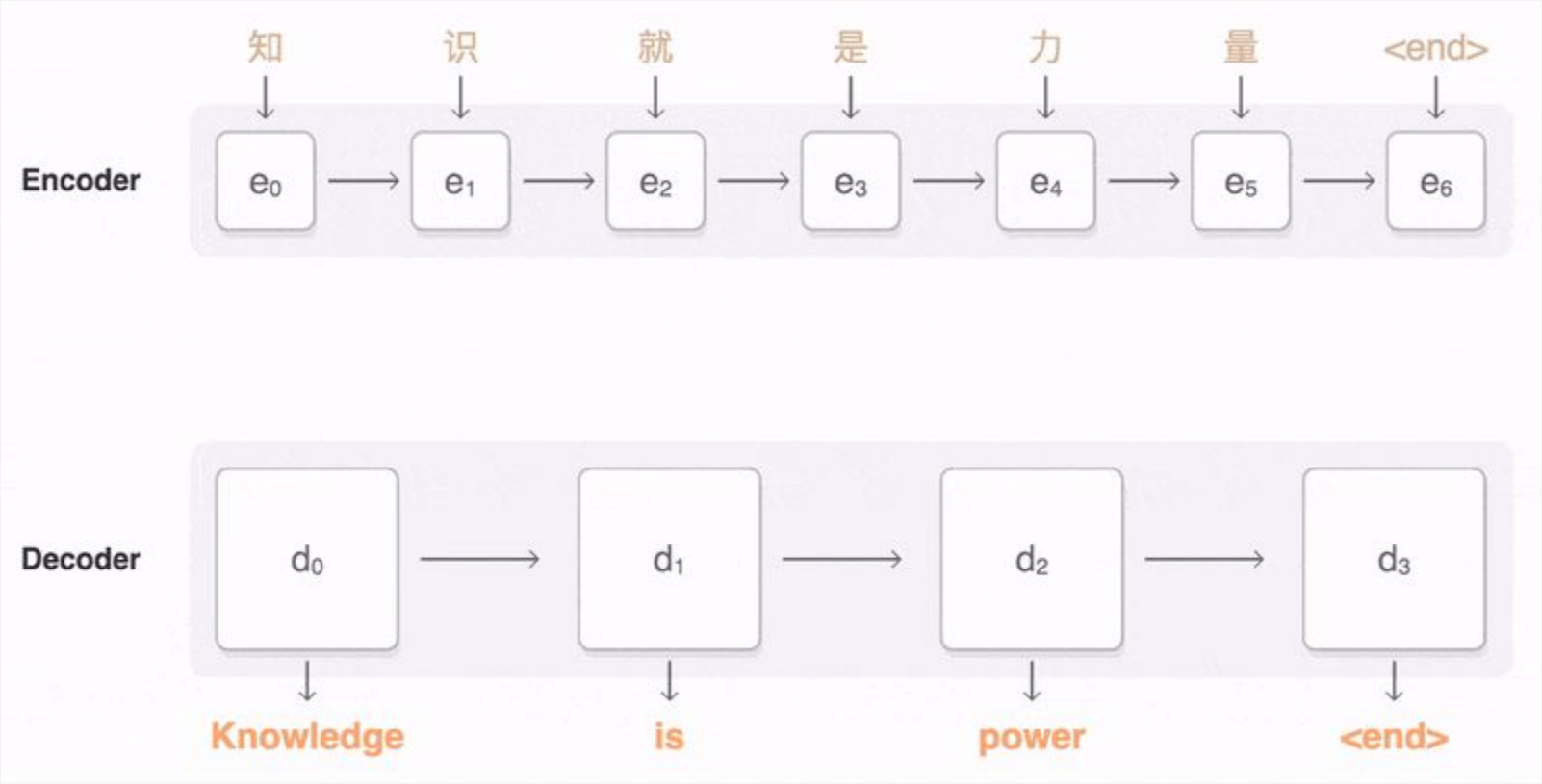
BertViz: https://github.com/jessevig/bertviz

Self Attention (BERT)



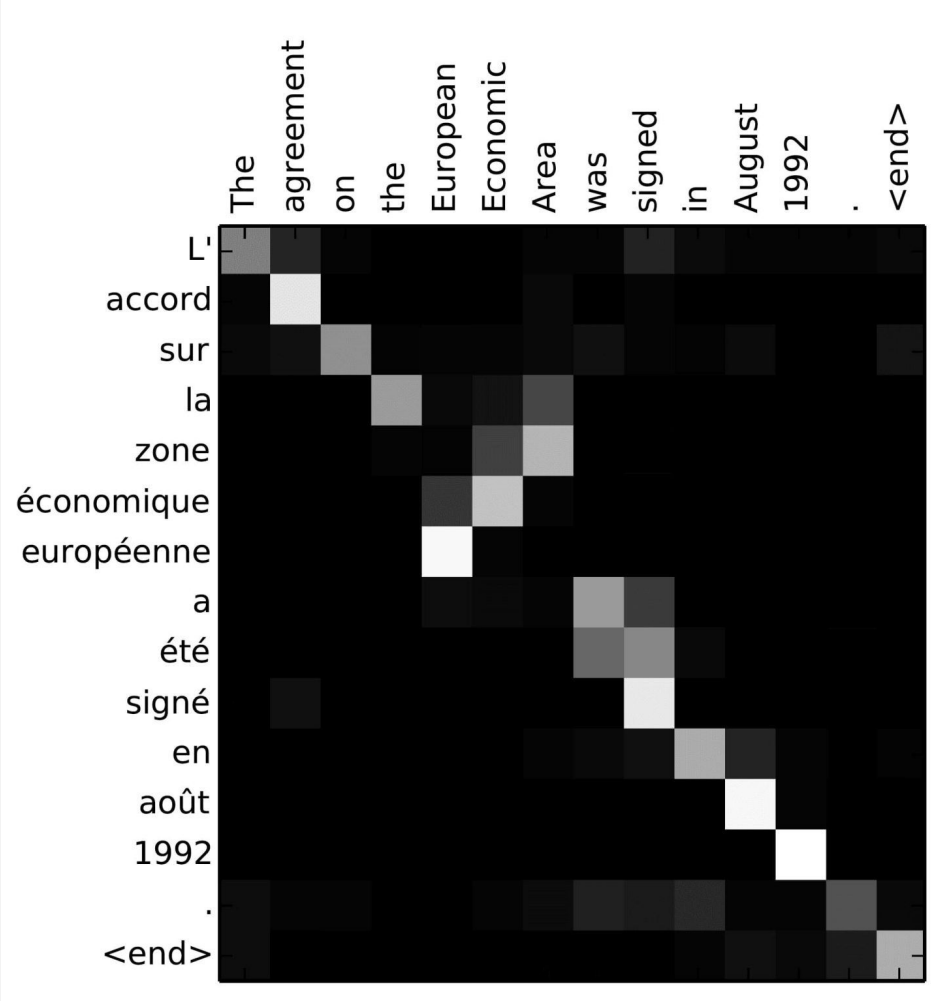
BertViz: https://github.com/jessevig/bertviz

Transformer: Machine Translation



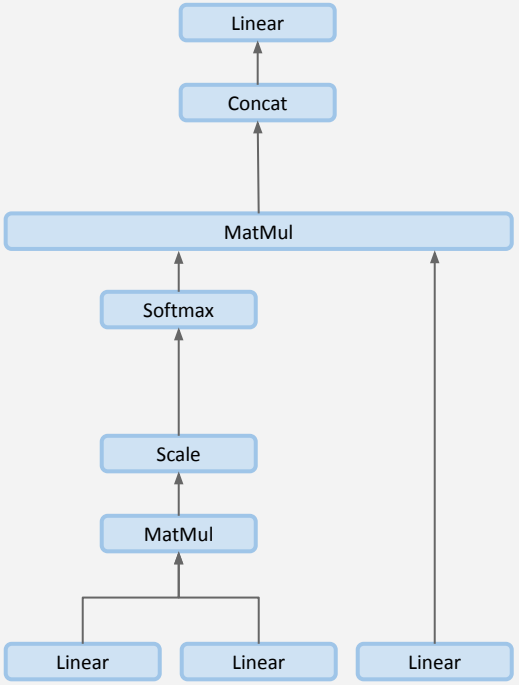
Google seq2seq: https://google.github.io/seq2seq/

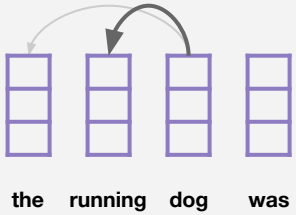
Transformer: Machine Translation



Bahdanau et al., 2015: https://arxiv.org/abs/1409.0473

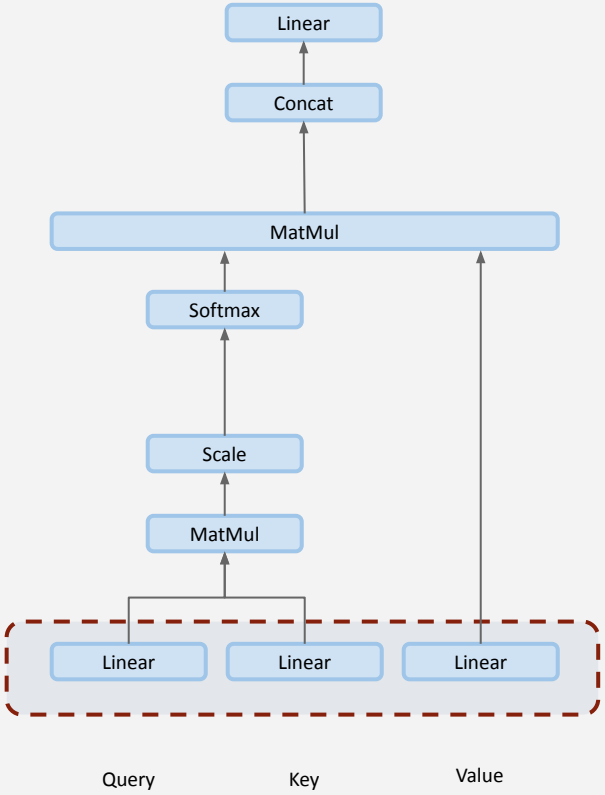
Self-Attention



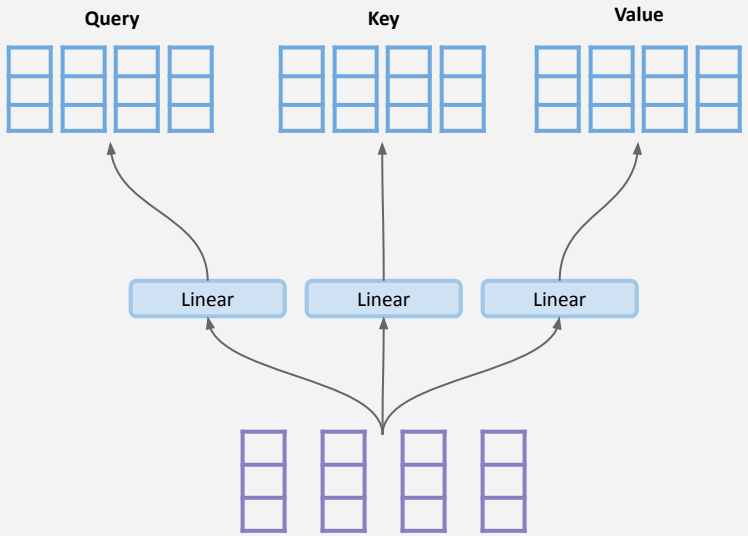


The self-attention mechanism allows the network to learn how each word in the input associates with other word in the input (or **how words attend to other words**).

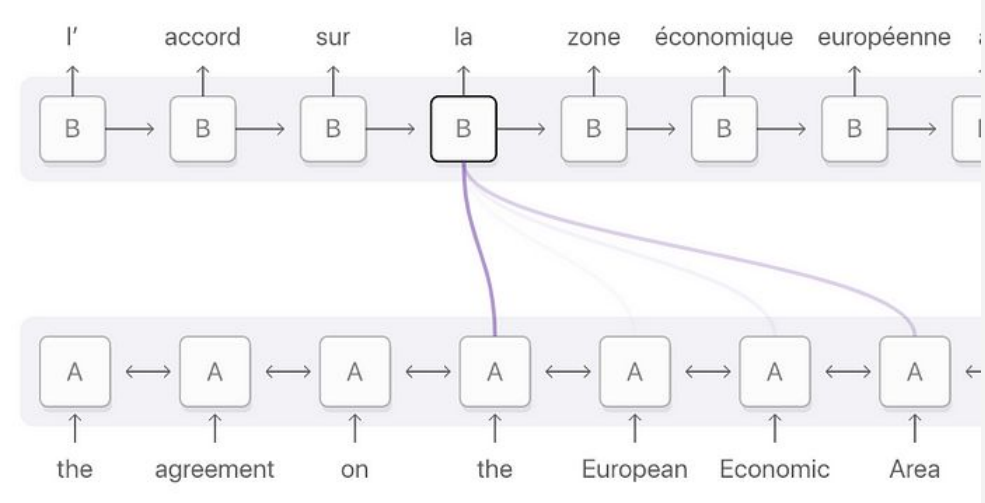
Self-Attention



The three linear layers produce **query**, **key** and **value** vectors.



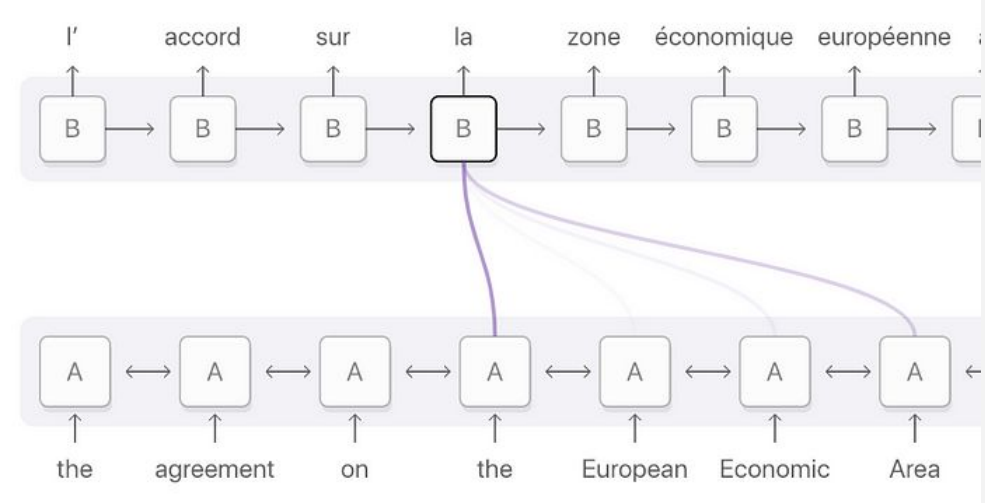
Transformer: Machine Translation



query [i] “la”

Bahdanau et al., 2015: https://arxiv.org/abs/1409.0473

Transformer: Machine Translation

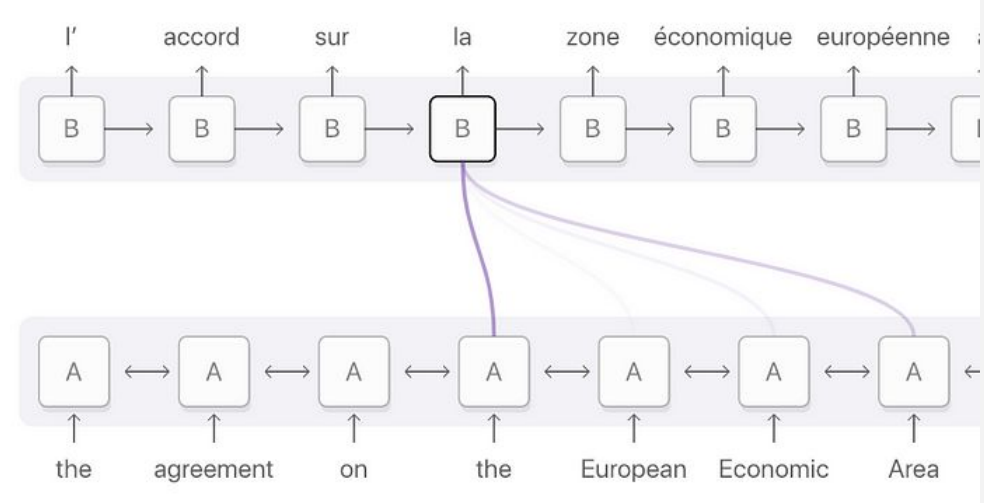


query [i] "la"

key [j] "the"

Bahdanau et al., 2015: https://arxiv.org/abs/1409.0473

Transformer: Machine Translation



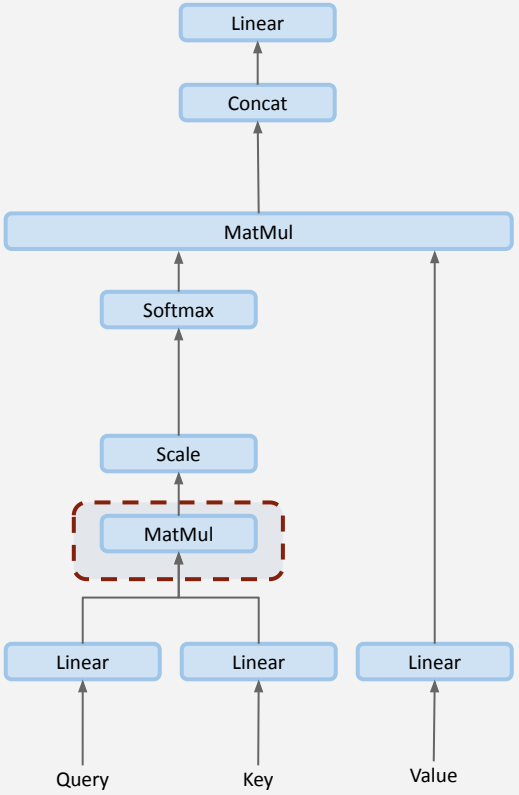
query [i] "la"

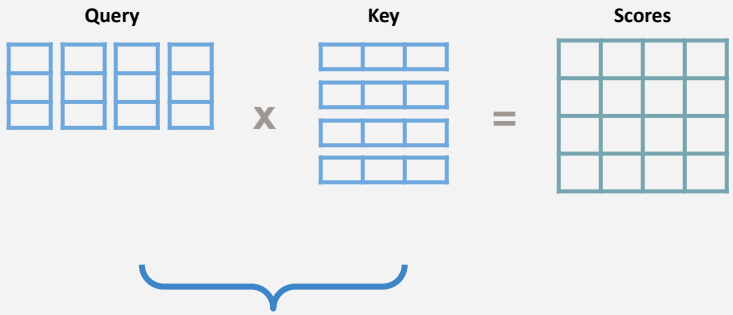
key [j] "the"

key [k] "Area"

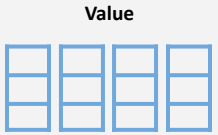
Bahdanau et al., 2015: https://arxiv.org/abs/1409.0473

Self-Attention

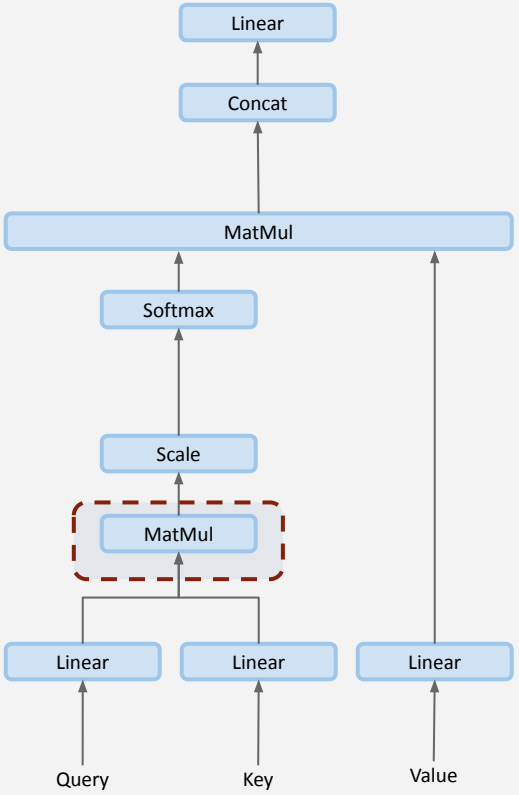


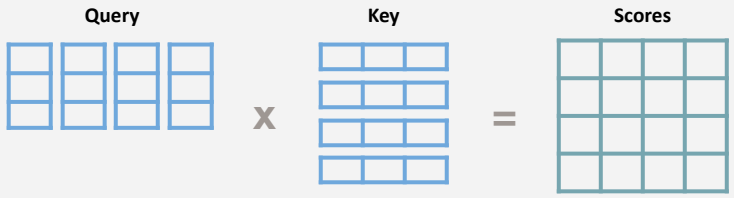


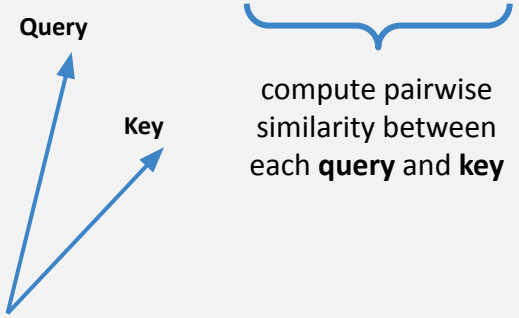
compute pairwise similarity between each **query** and **key**



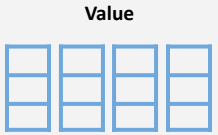
Self-Attention: Cosine Similarity



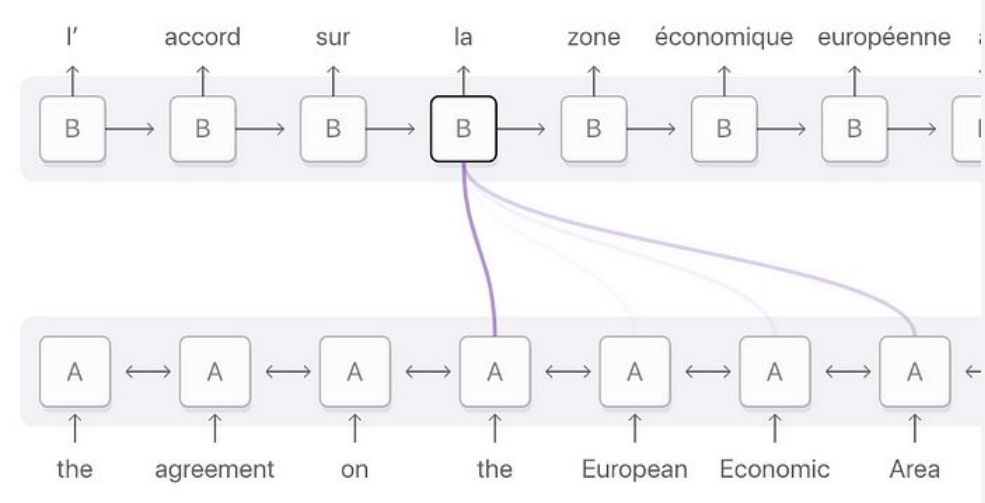




compute pairwise similarity between each **query** and **key**



Transformer: Machine Translation



query [i] "la"

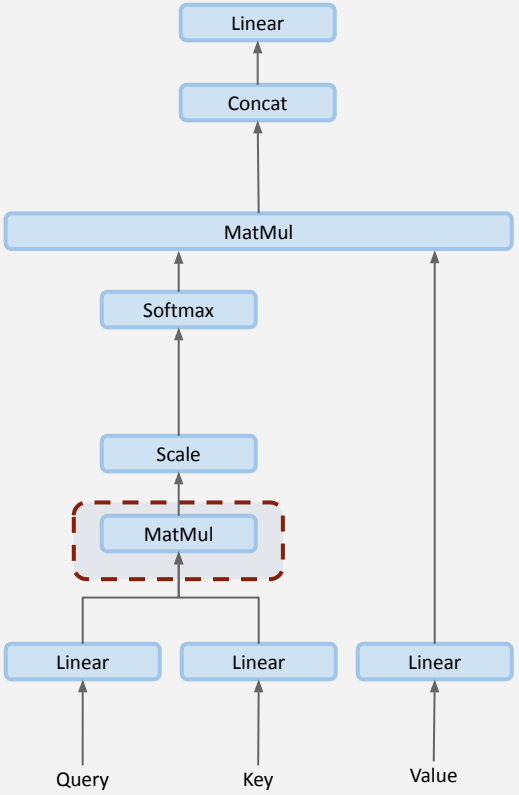
key [j] "the"

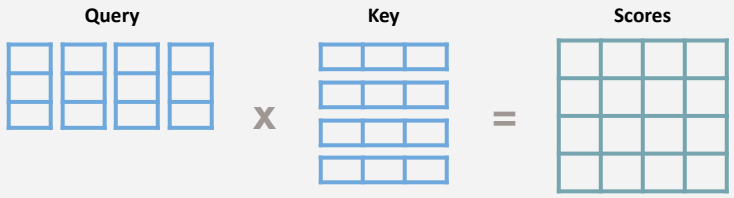
key [k] "Area"

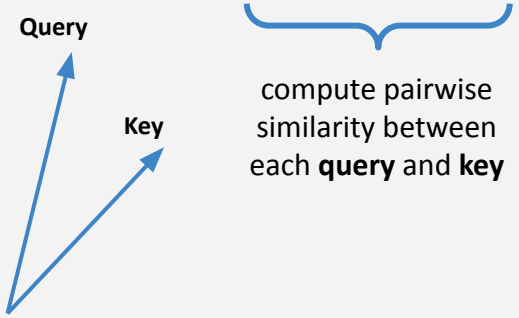
Attention Score [i, j] = query [i] * key [j]

Bahdanau et al., 2015: https://arxiv.org/abs/1409.0473

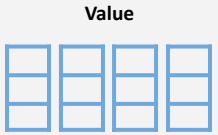
Self-Attention: Cosine Similarity



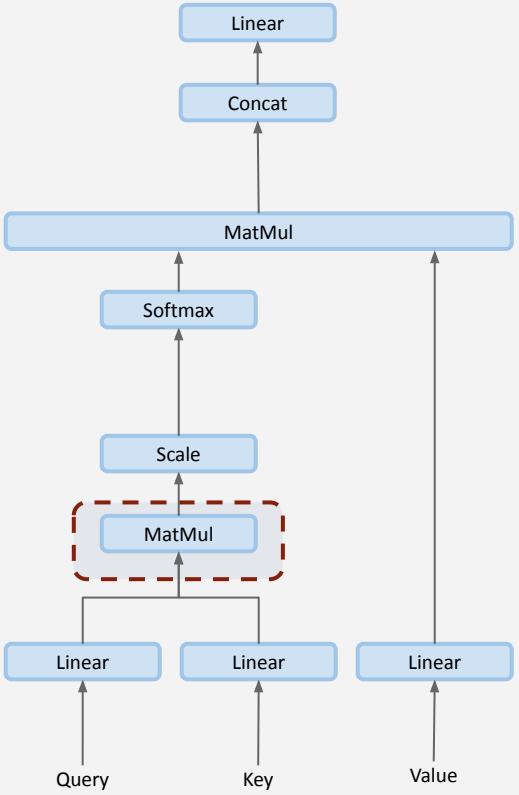


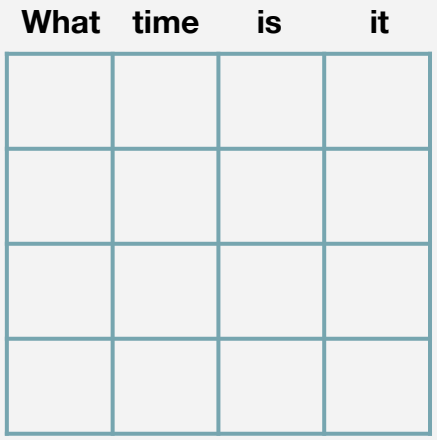


compute pairwise similarity between each **query** and **key**



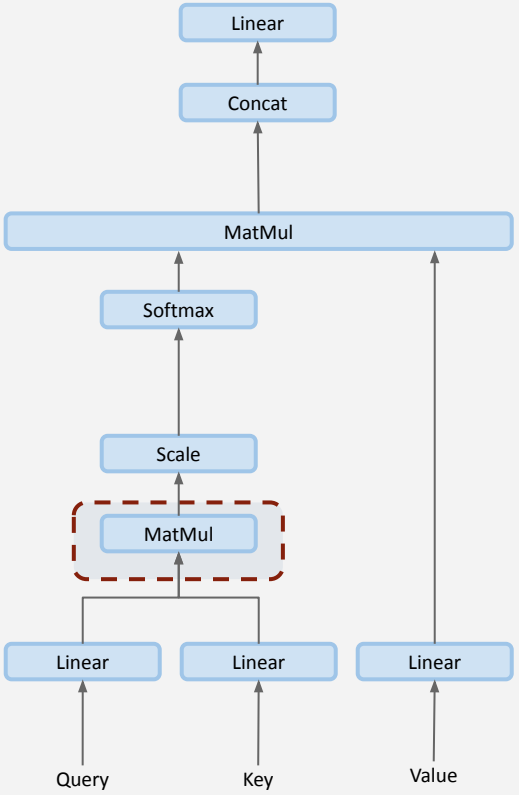
Self-Attention





Attention Scores: how much each word should attend to every other word.

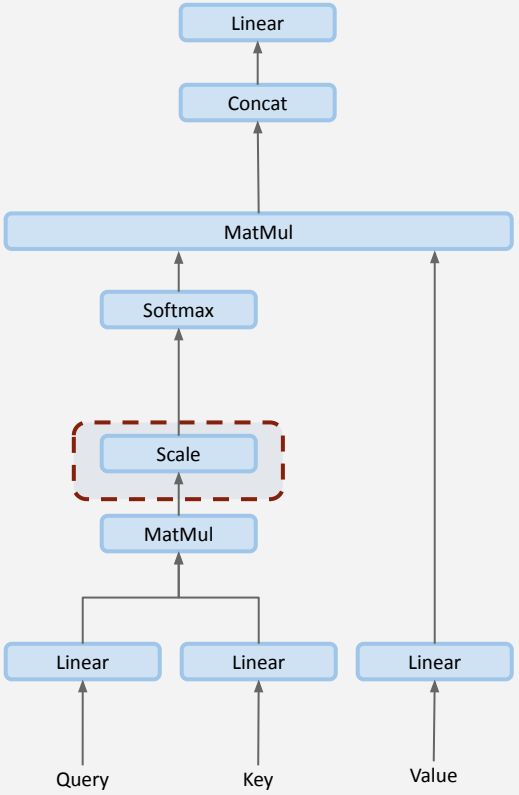
Self-Attention

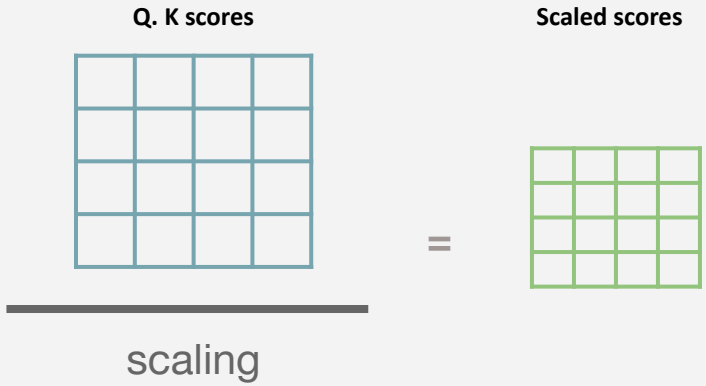


	What	time	is	it
Wie	86	23	26	66
spät	55	95	54	72
ist	95	19	98	63
es	36	44	30	73

Attention Scores: how much each word should attend to every other word.

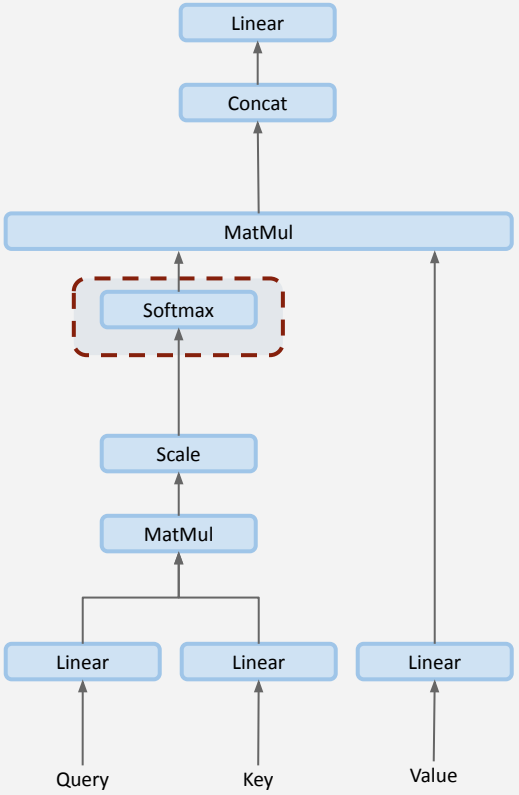
Self-Attention





Scaled Scores: attention scores are scaled down to avoid vanishing/exploding effects during backpropagation.

Self-Attention



softmax ( ) =

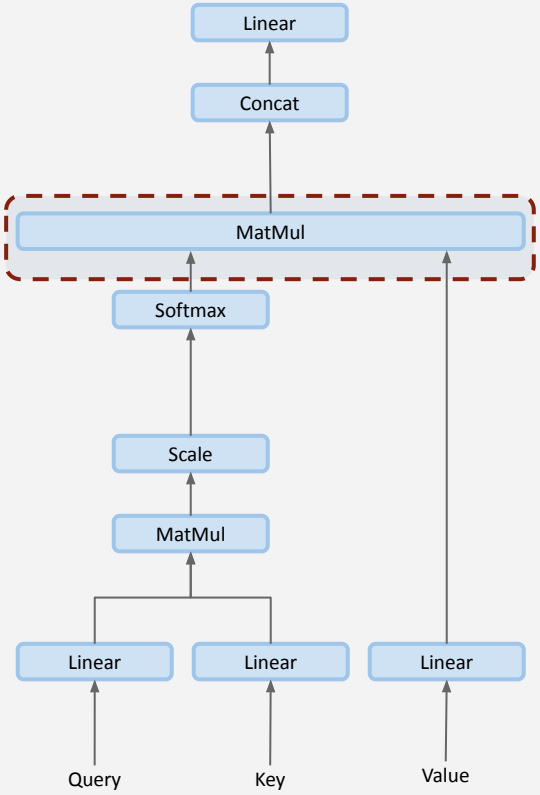
Attention weights

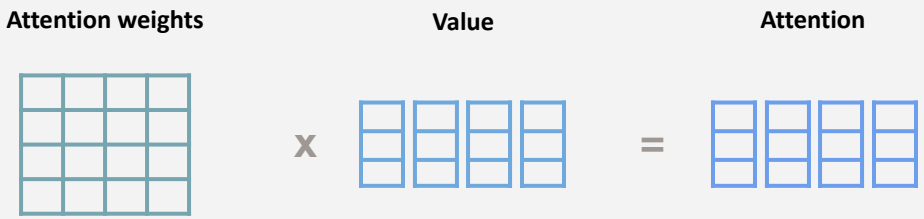
	What	time	is	it
What	0.6	0.1	0.2	0.1
time	0.1	0.7	0.1	0.1
is	0.1	0.2	0.6	0.1
it	0.1	0.2	0.2	0.5

$$soft\max(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Attention Weights: The softmax transforms the scaled scores to probabilities. The softmax heightens higher scores and suppresses lower scores.

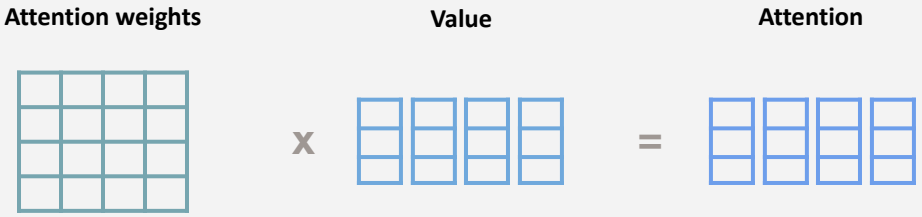
Self-Attention



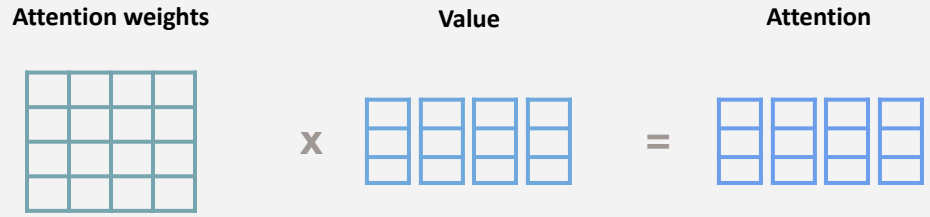


The attention weights (softmax scores) will allow the network to learn which words are important in the current context (**value** vector).

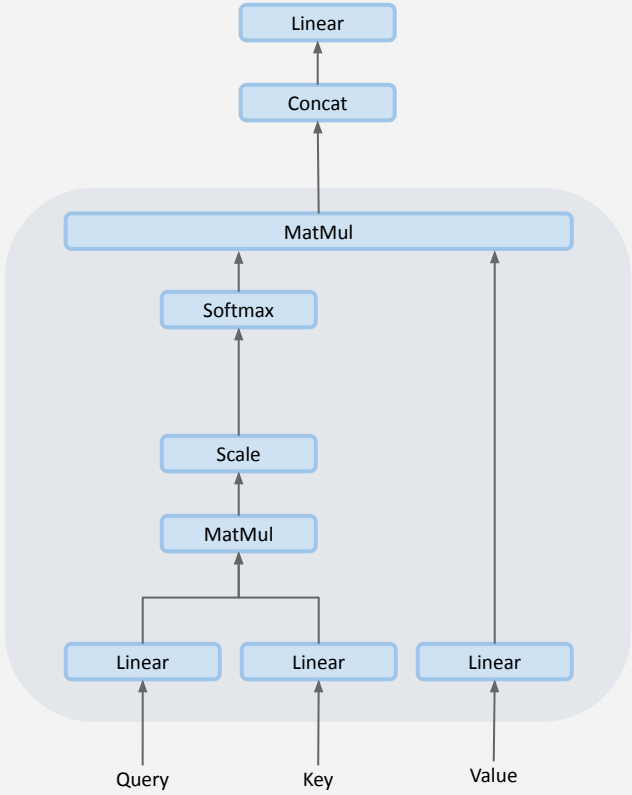
Self-Attention: Visual Intuition

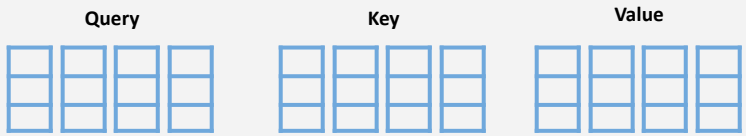


Self-Attention: Visual Intuition



Self-Attention

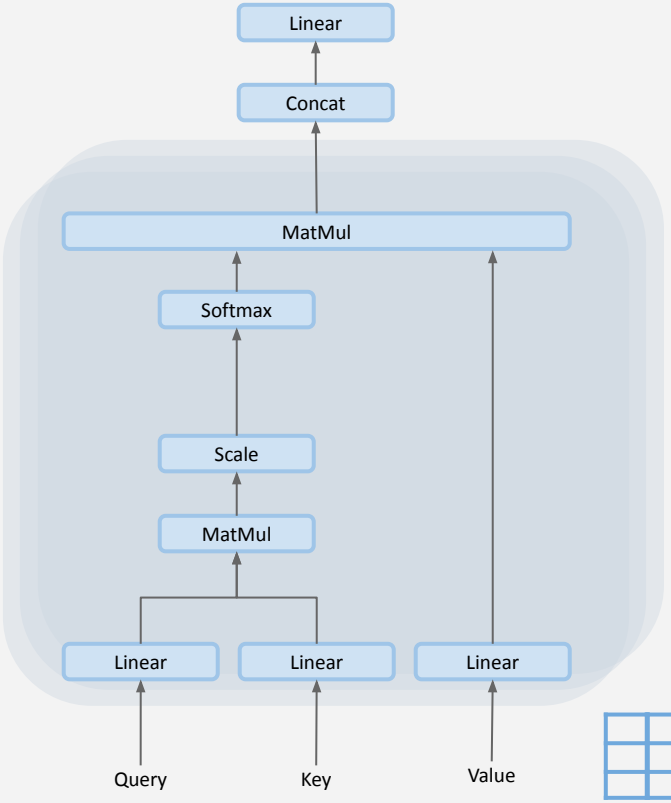


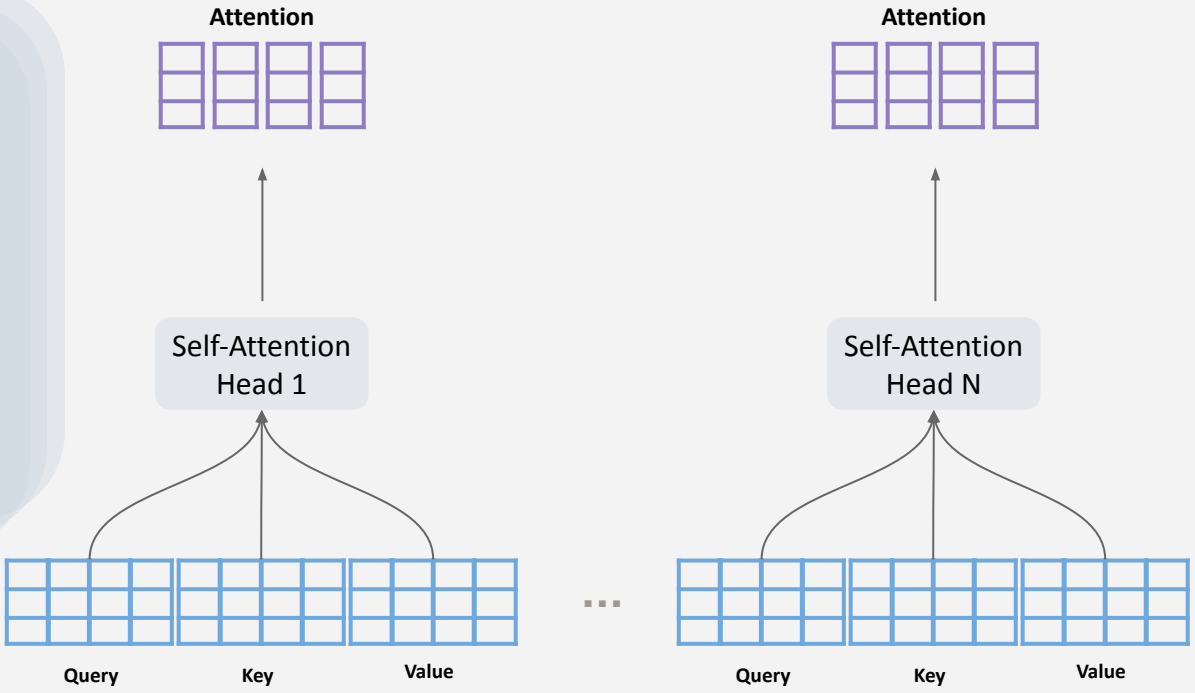


Multi-headed Attention

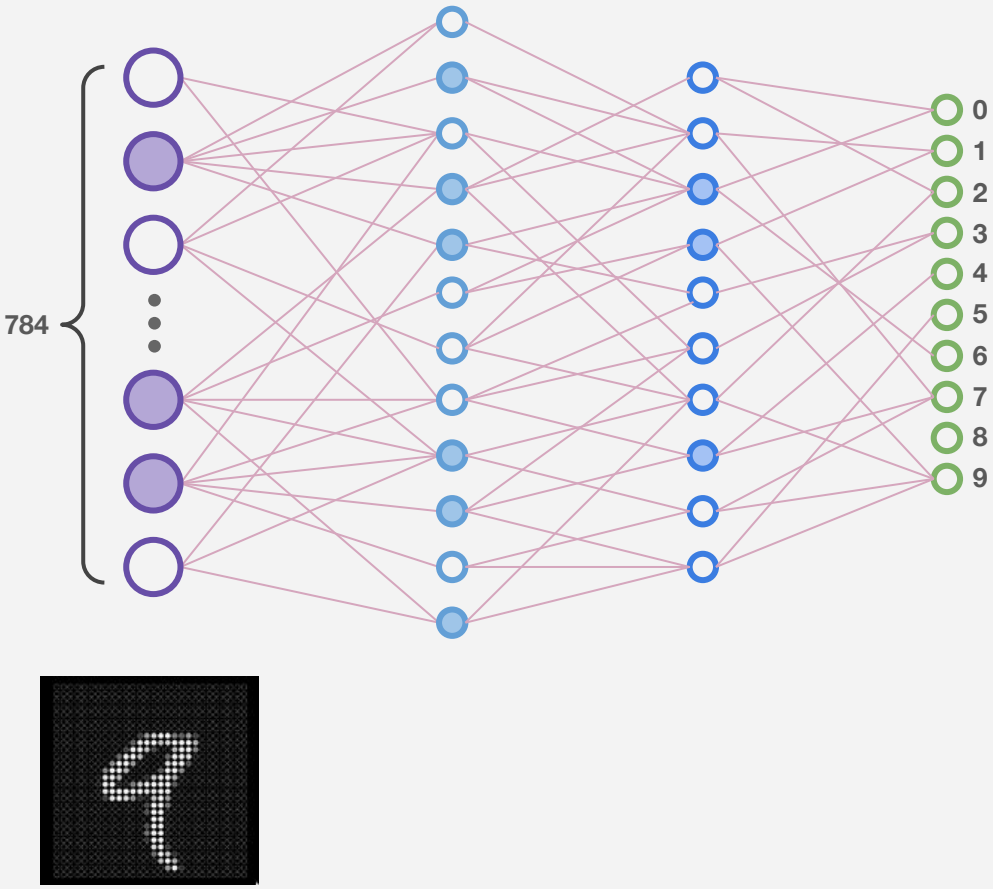
The query, key and value vectors are split into ***N*** vectors that go through the self-attention mechanism individually.
Each self-attention process is called a head.

Each head, in theory, will learn distinct features from the input, giving the encoder more representation power.

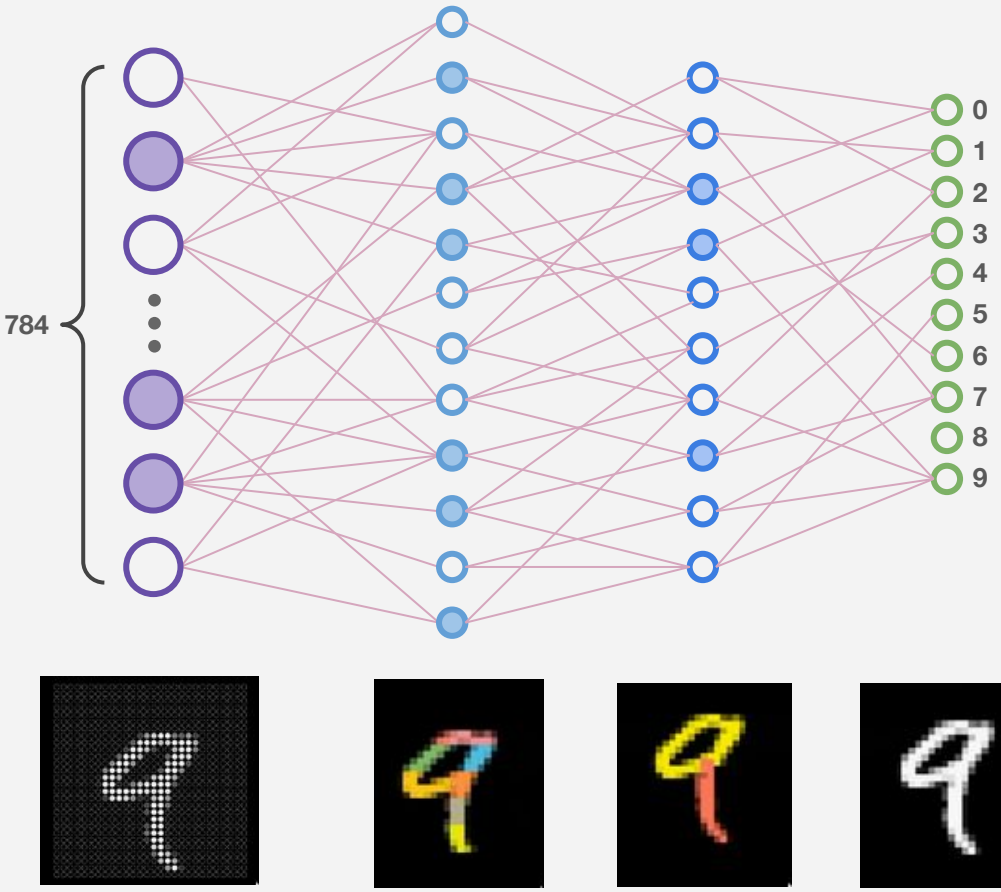




CNN: Representation



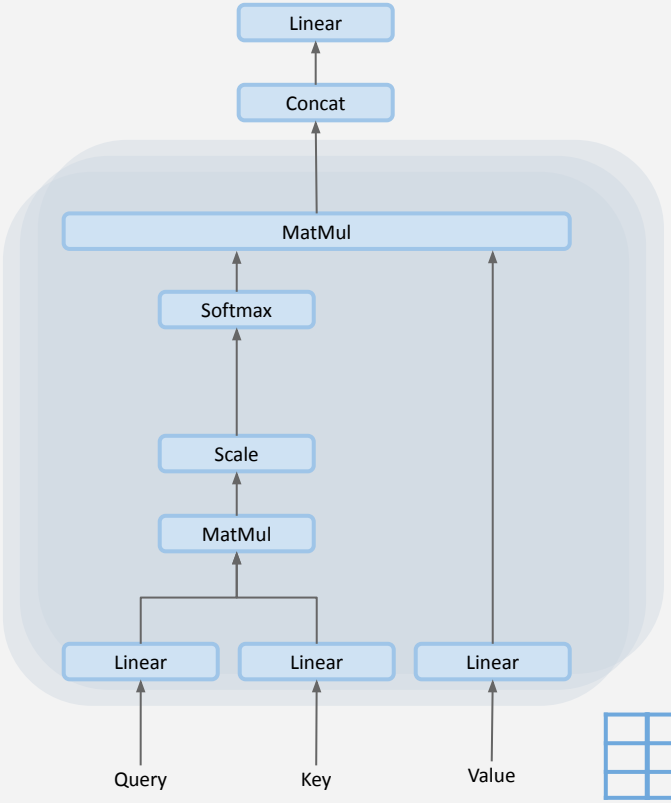
CNN: Representation

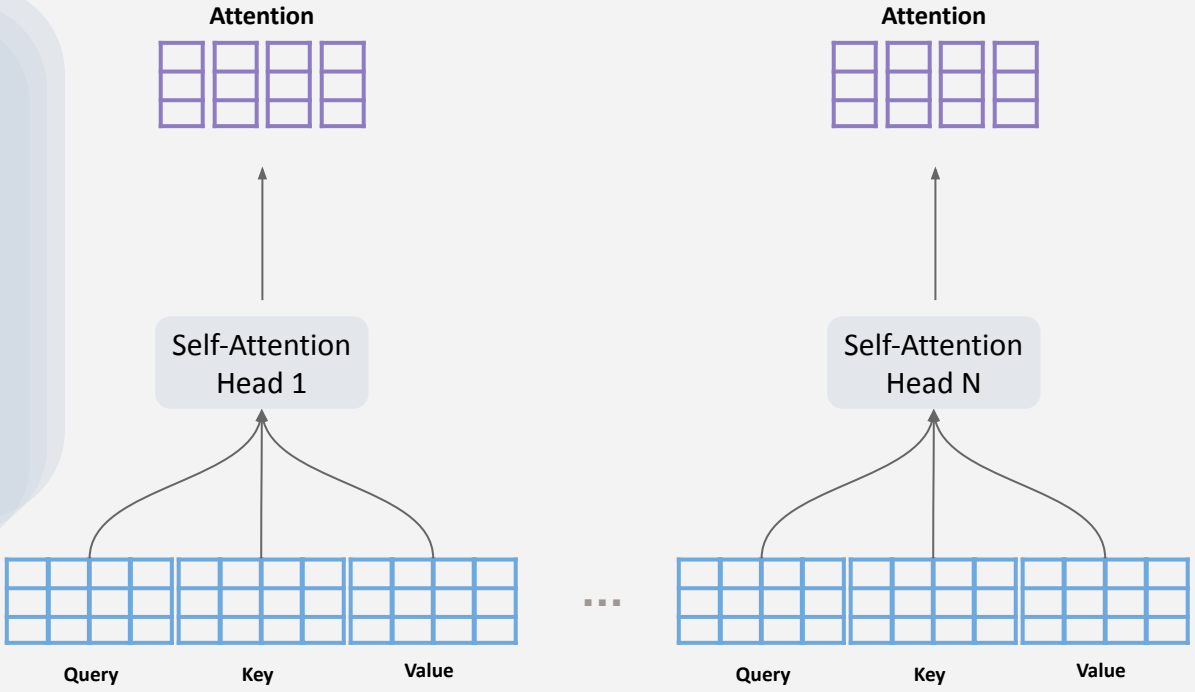


Multi-headed Attention

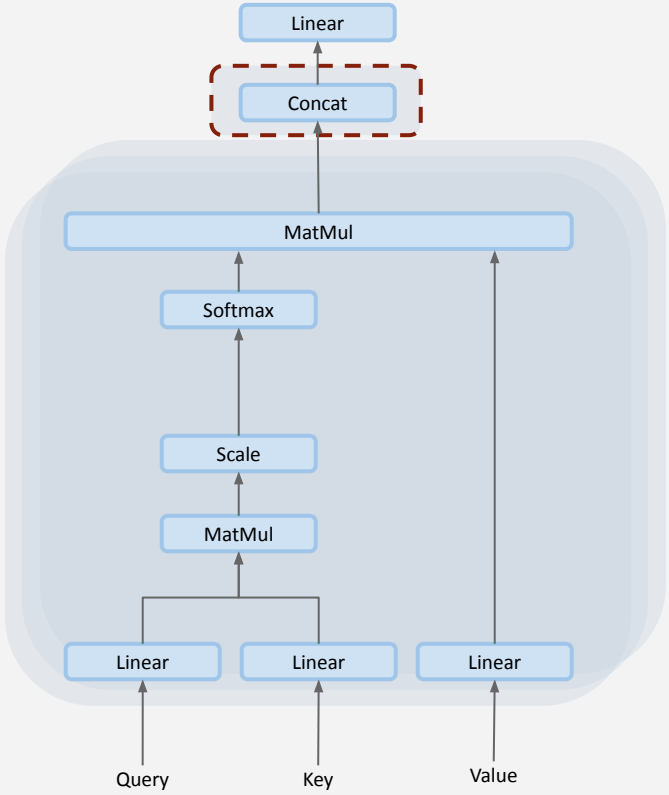
The query, key and value vectors are split into ***N*** vectors that go through the self-attention mechanism individually.
Each self-attention process is called a head.

Each head, in theory, will learn distinct features from the input, giving the encoder more representation power.





Multi-headed Attention

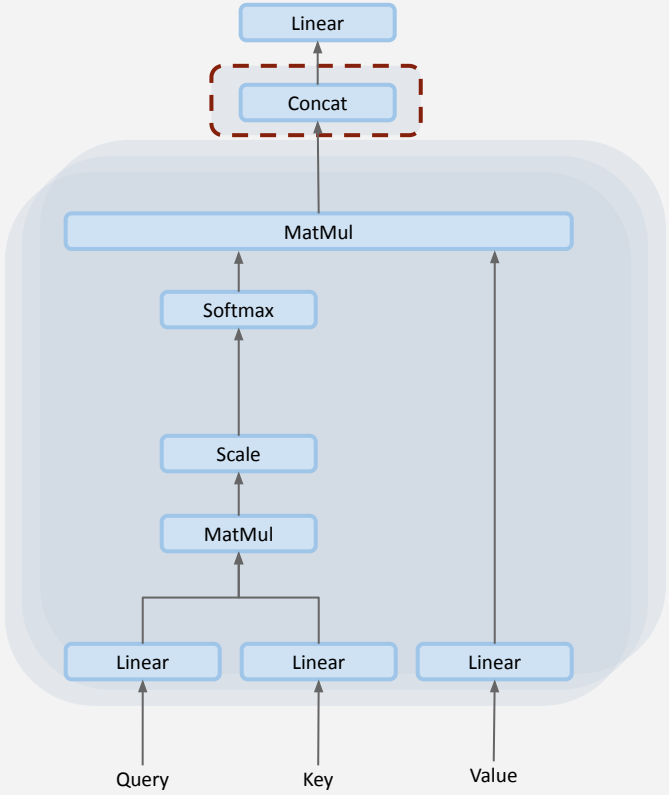


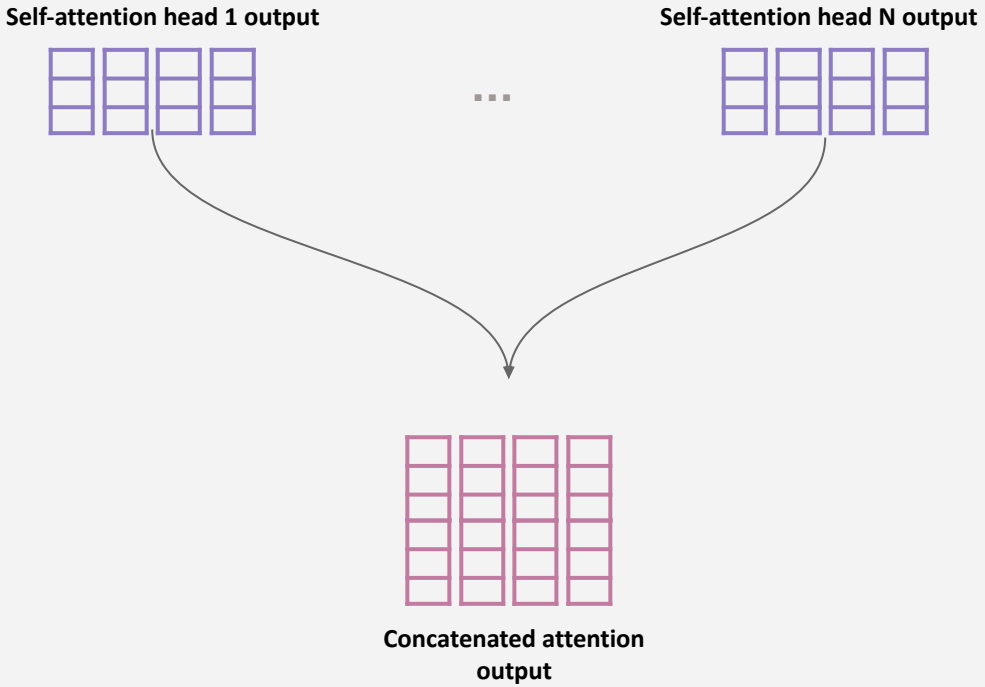
Self-attention head 1 output

...

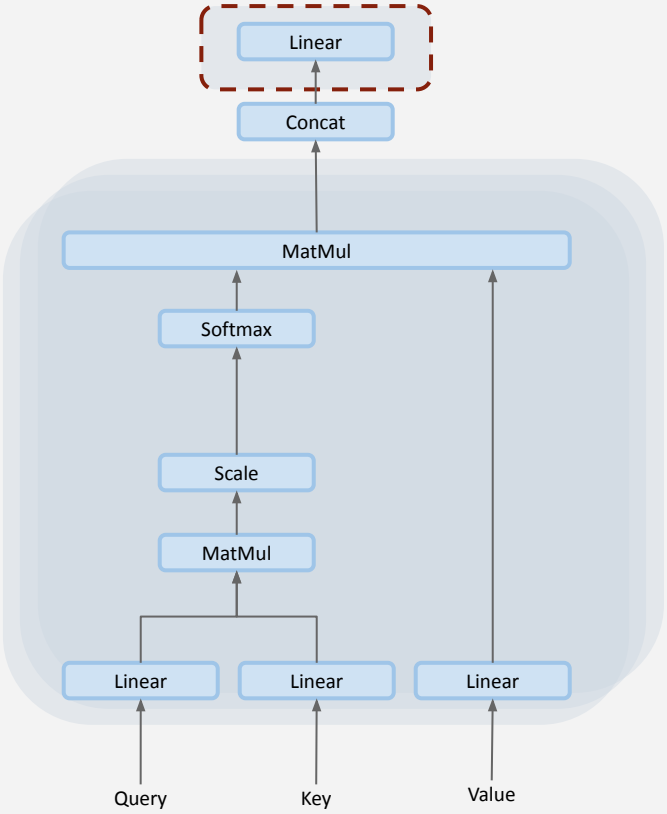
Self-attention head N output

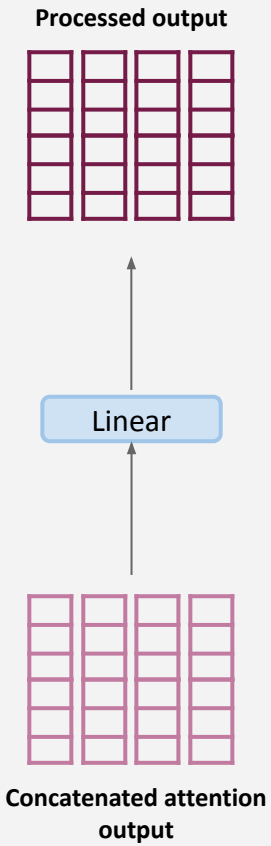
Multi-headed Attention



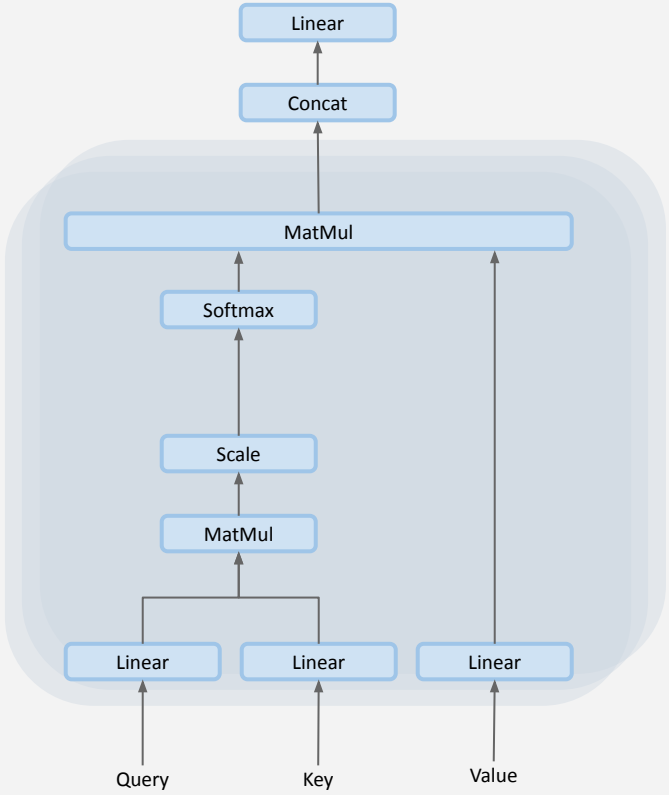


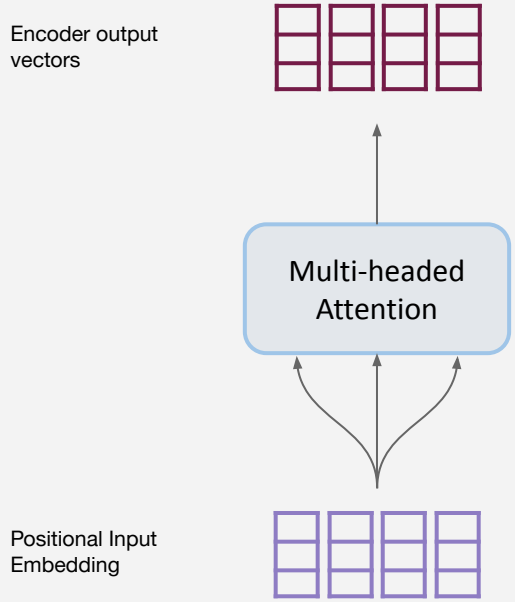
Multi-headed Attention





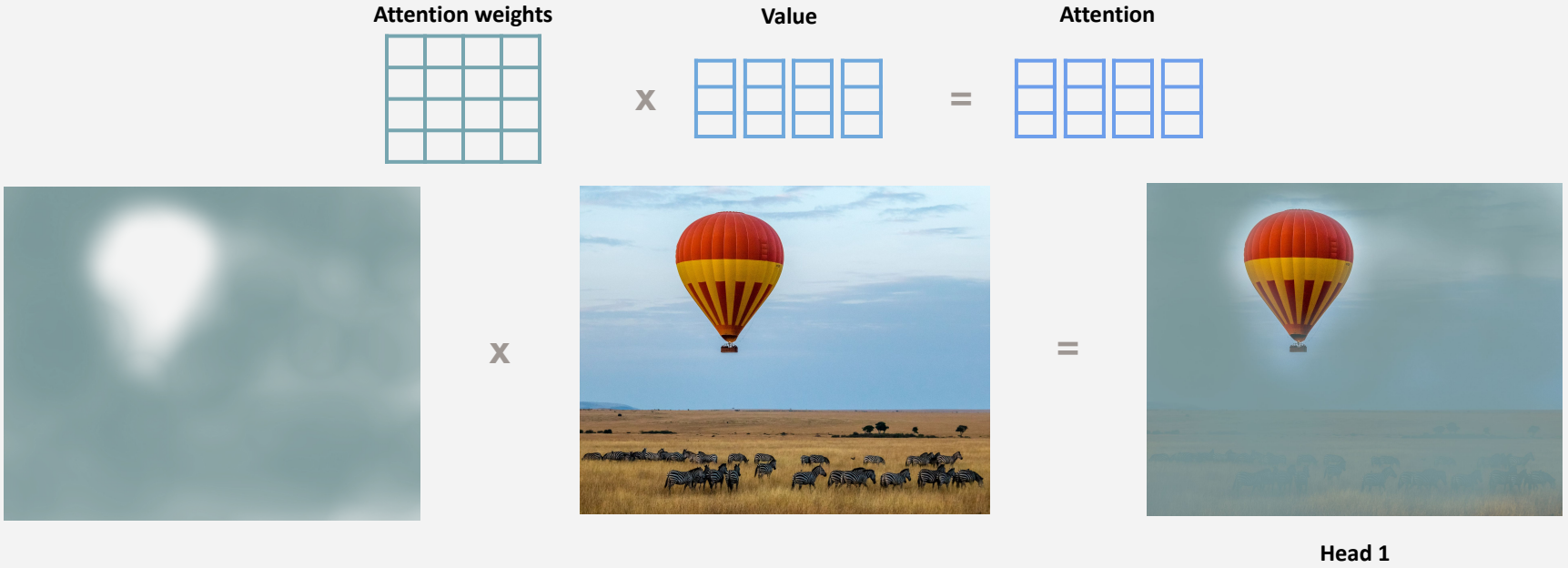
Multi-headed Attention



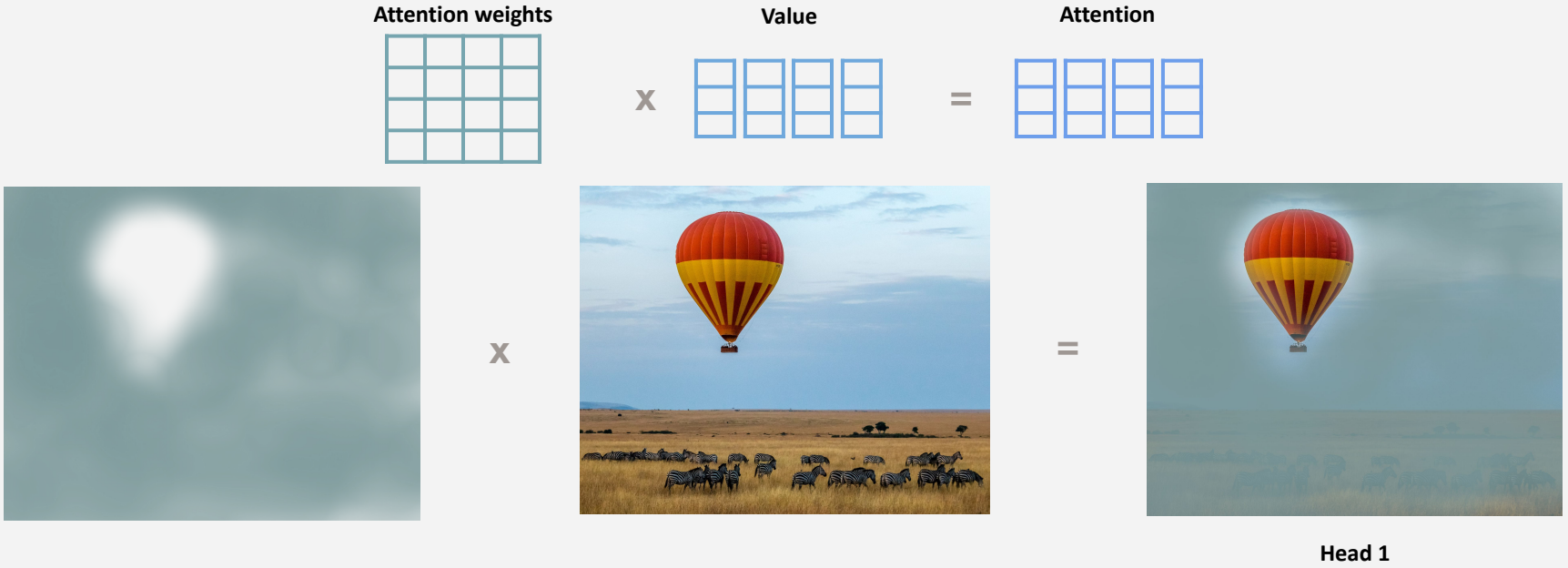


Multi-headed attention is a module that computes attention weights for the input
and produces an output that encodes information on how each word should attend to every other word

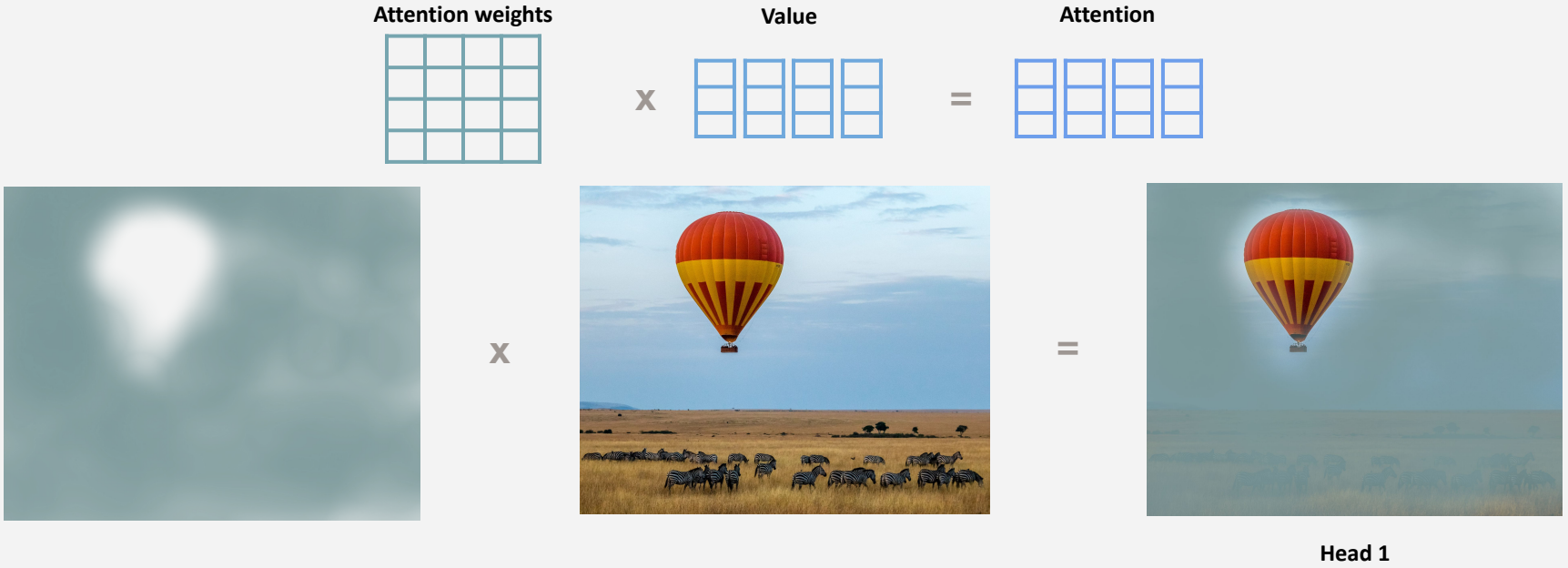
Multi-headed Attention: Visual Intuition



Multi-headed Attention: Visual Intuition



Multi-headed Attention: Visual Intuition



Show, Attend and Tell Neural Image Caption Generation with Visual Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

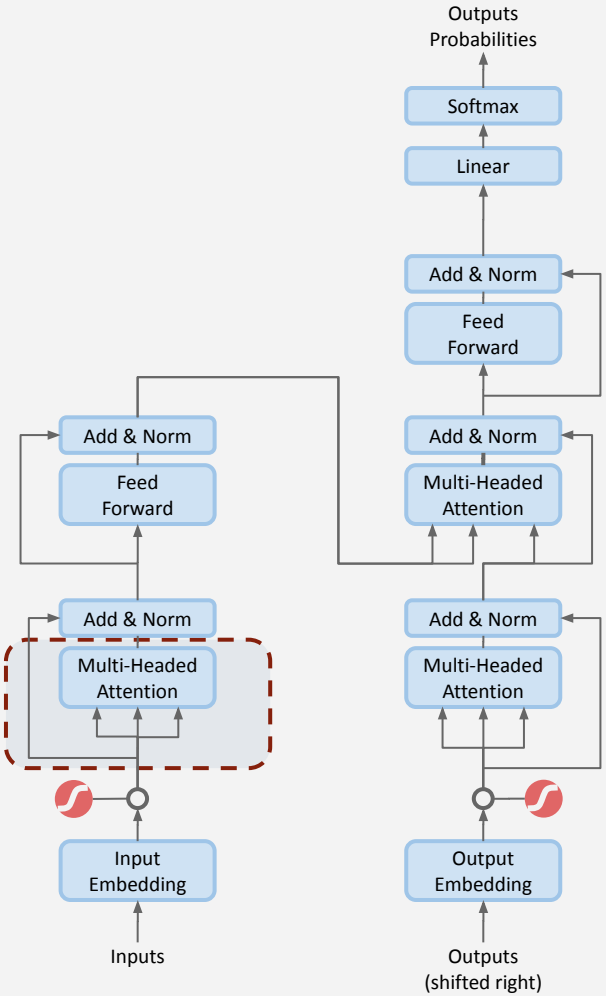
A little girl sitting on a bed with a teddy bear.

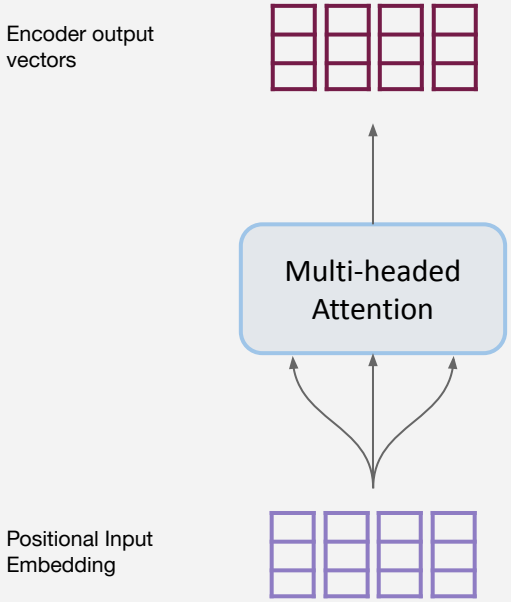
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

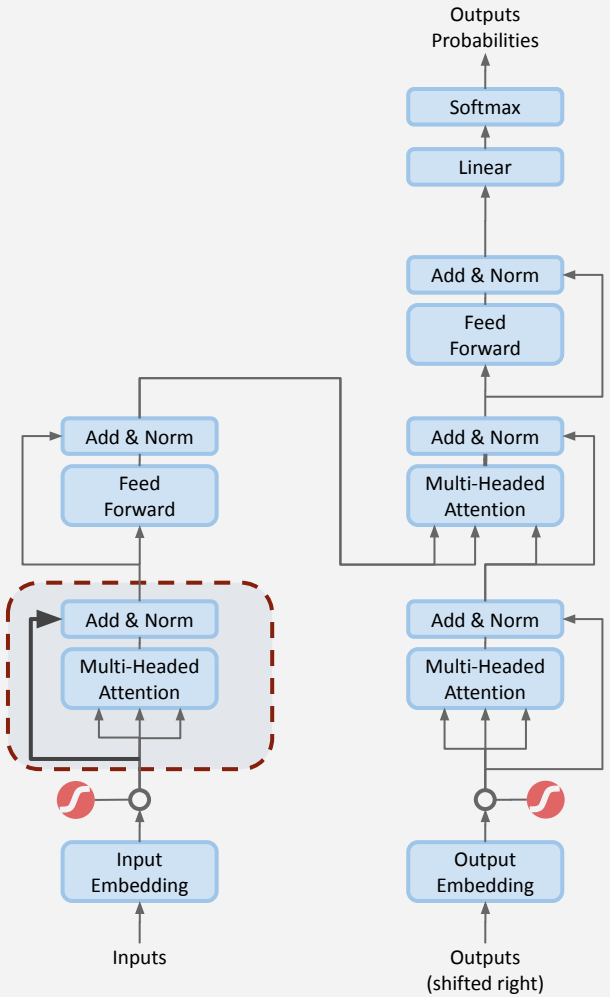
Xu et al., 2015: https://arxiv.org/abs/1502.03044

Encoder

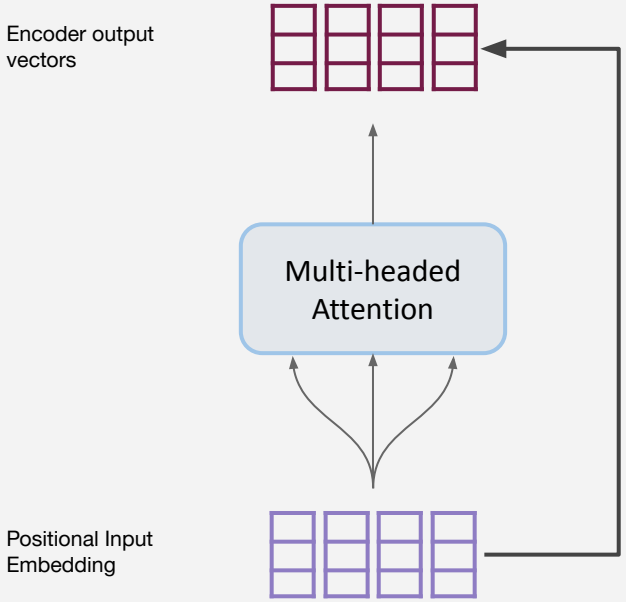




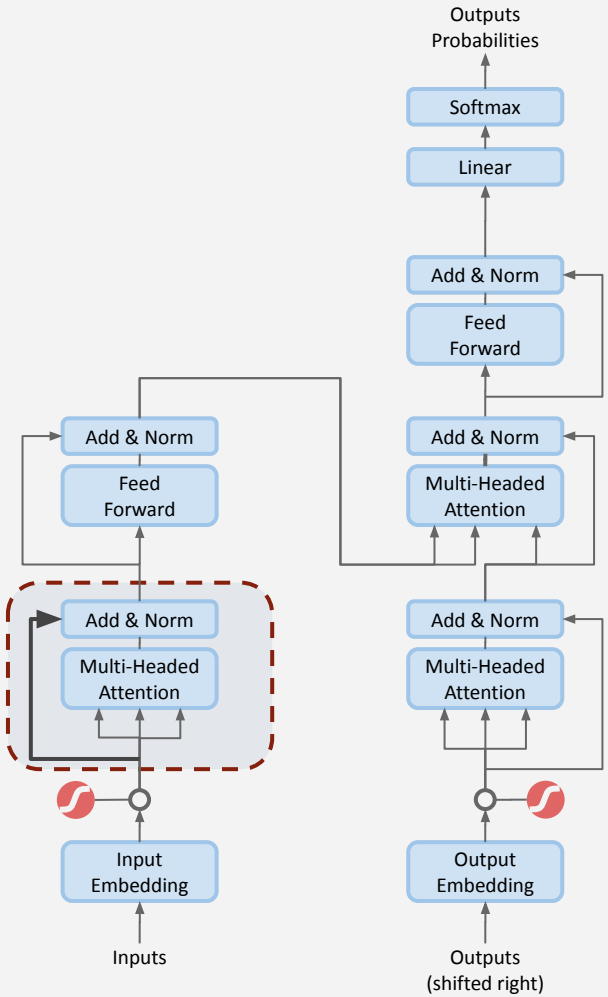
Encoder



The residual connection (also called a skip connection) allows direct flow of information.

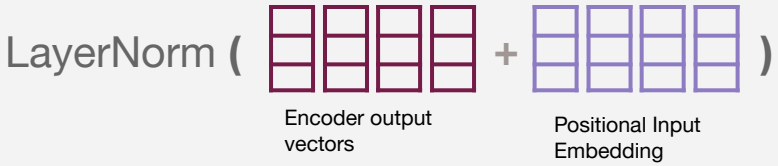


Encoder

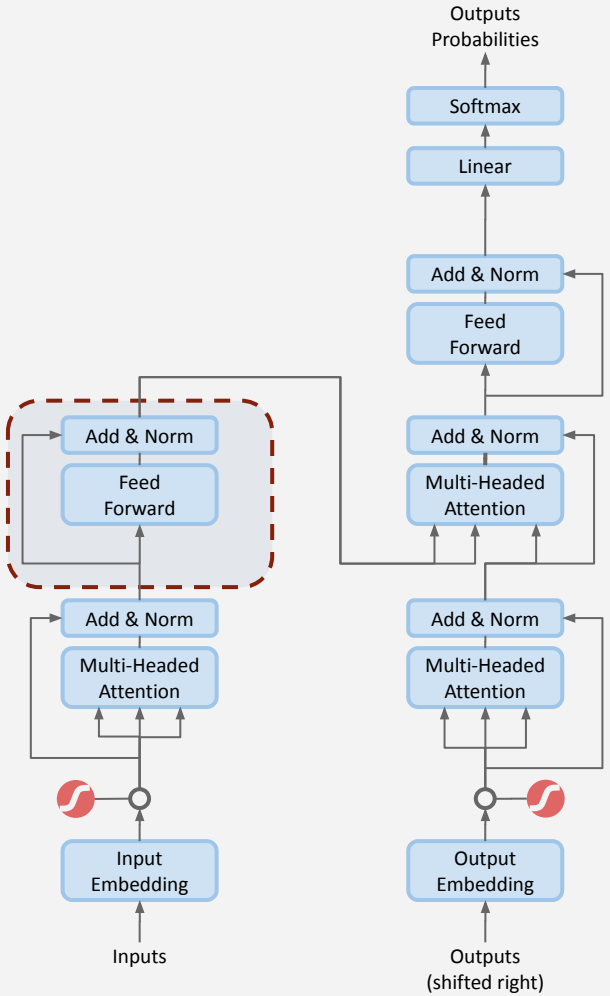


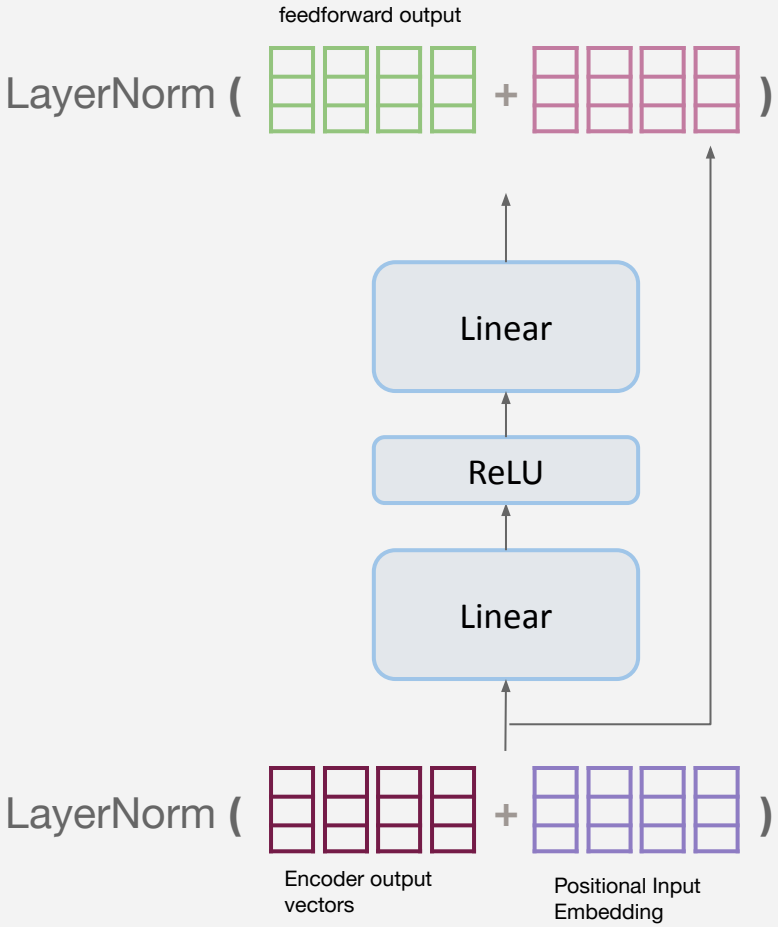
Layer normalization uses mean and variance across a layer (the input sequence, for instance) instead of computing them across a batch.

This allows for smoother gradient updates and better generalization accuracy.



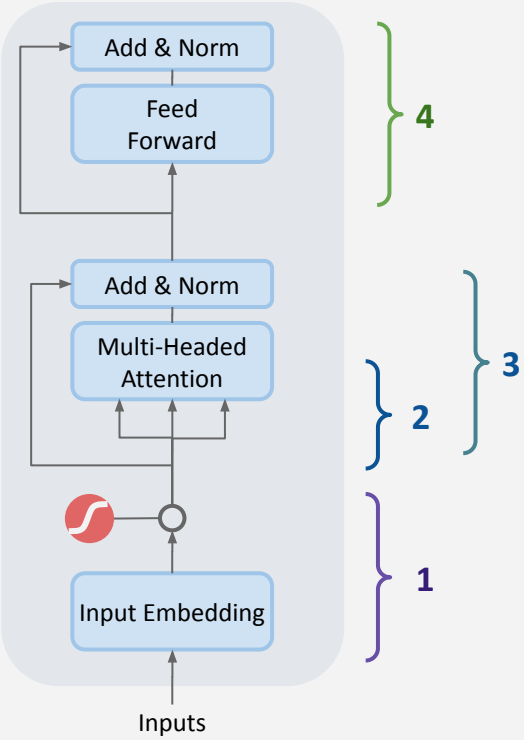
Encoder





Encoder

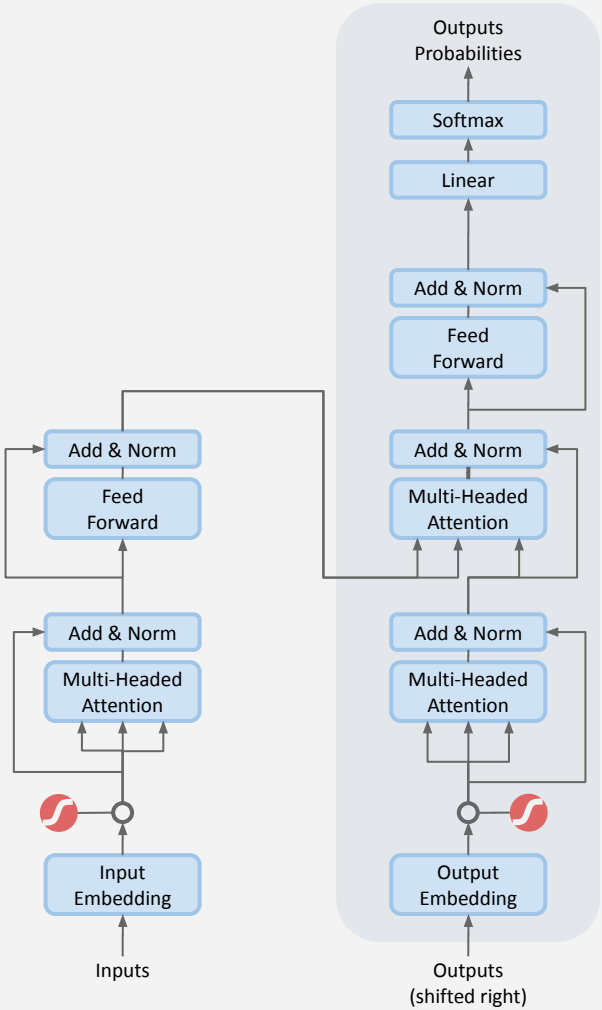
1. **Encode** input + **position** information
2. Learn **query**, **key** and **value** for search
3. Compute **attention weights** for all heads
4. Attend to **features** (=extract) with **high attention**



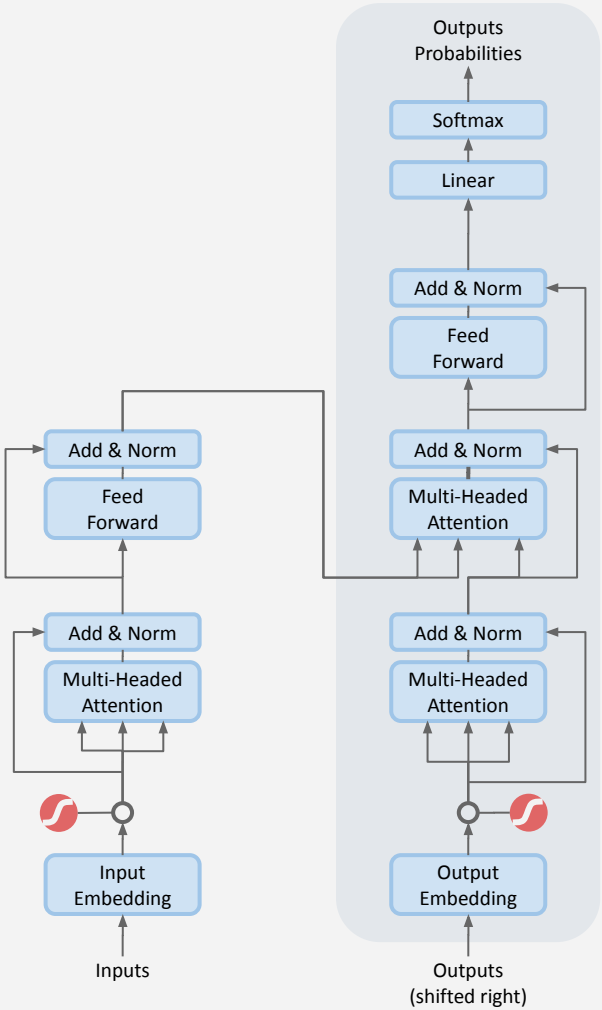
Each head (ideally) **attends to different parts of the input**

The encoder captures important semantic relationships

Decoder

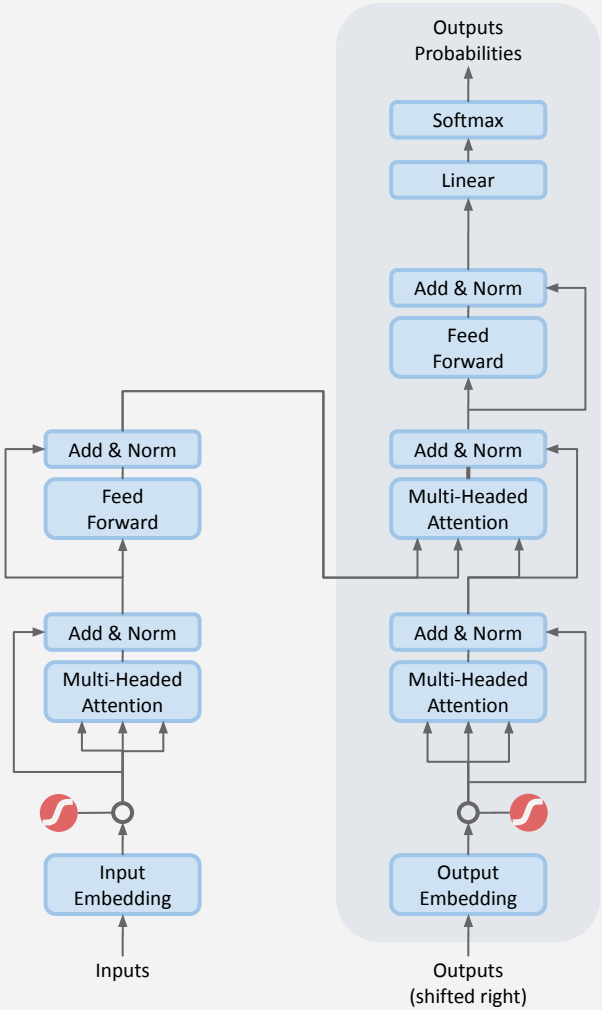


Decoder



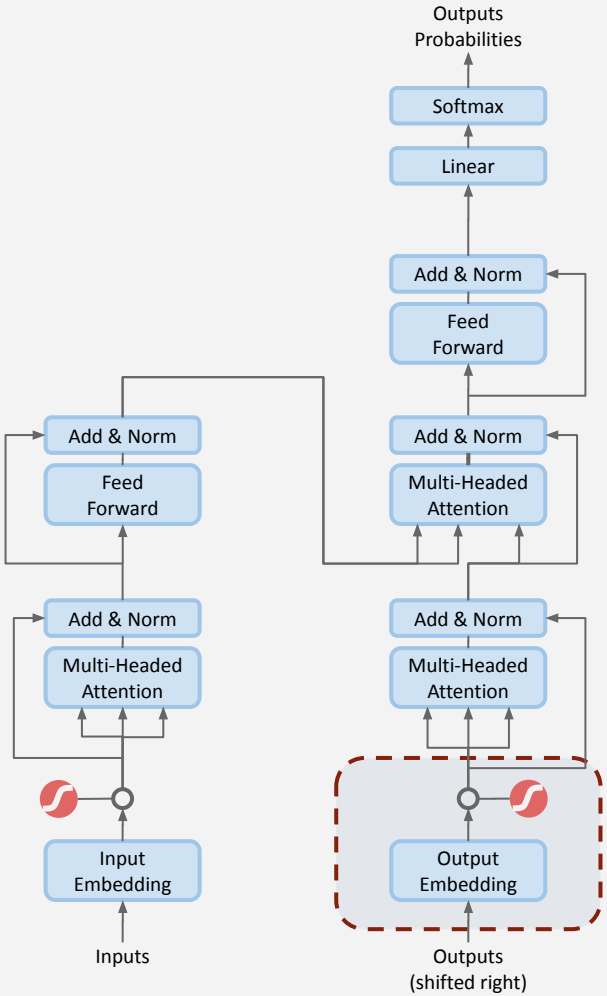
What time is it?

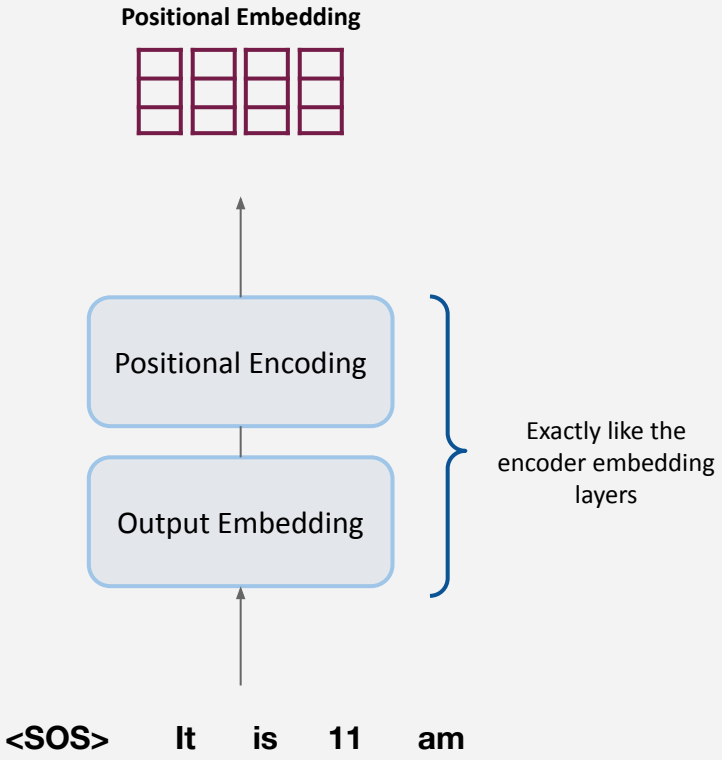
Decoder



What time is it? **It is 11 am**

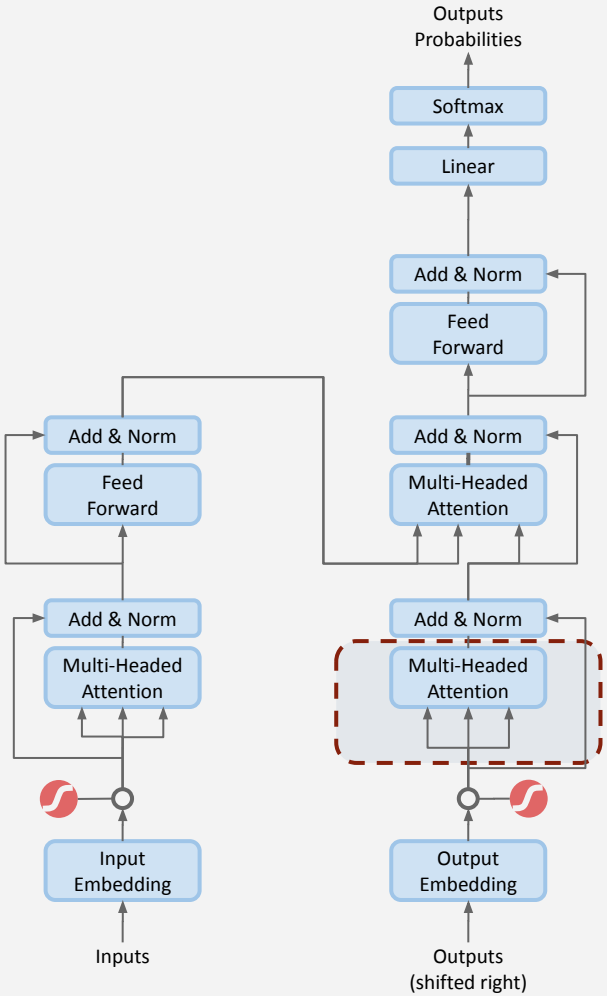
Decoder

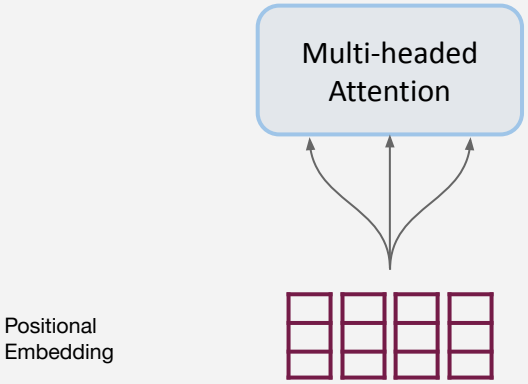




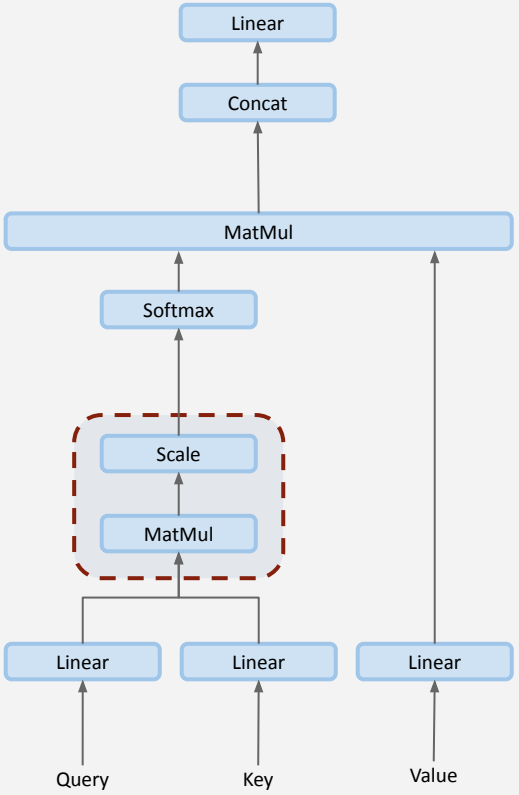
What time is it? **It is 11 am**

Decoder



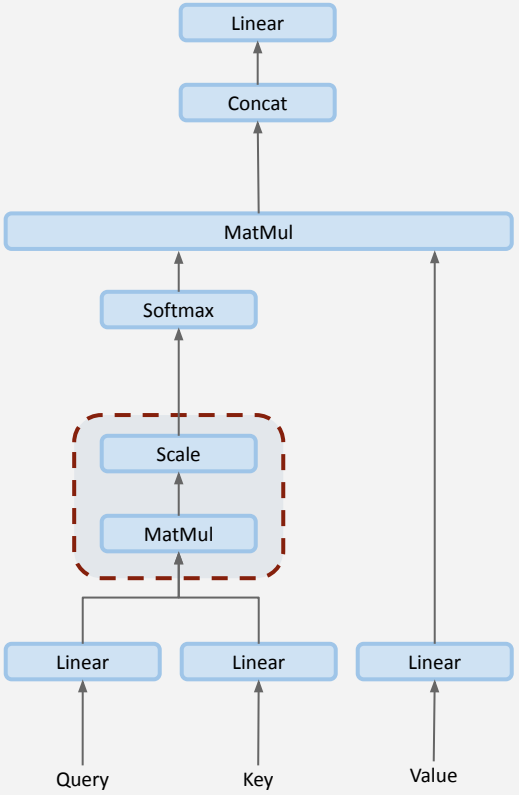


Self-Attention



It is 11 am. It is 7C and partly cloudy

Self-Attention

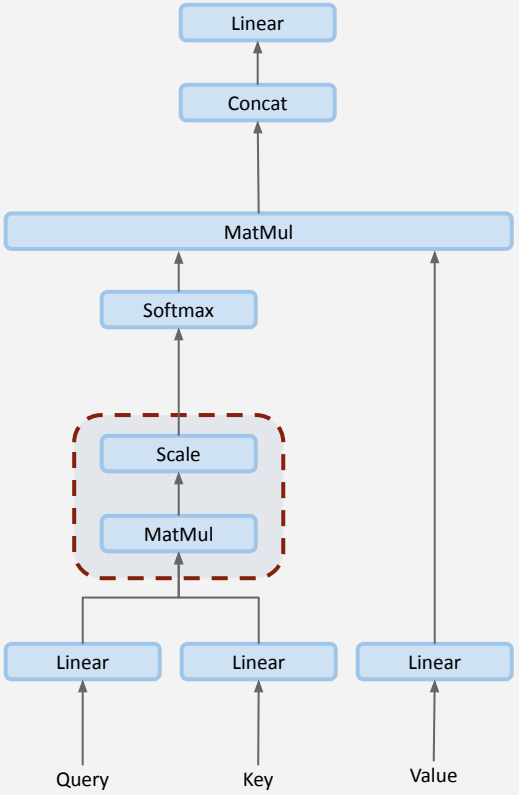


It is 11 am. It is 7C and partly cloudy

Scaled Scores

	<SOS>	it	is	11
<SOS>	0.6	0.1	0.2	0.1
it	0.1	0.7	0.1	0.1
is	0.1	0.2	0.6	0.1
11	0.1	0.2	0.2	0.5

Self-Attention

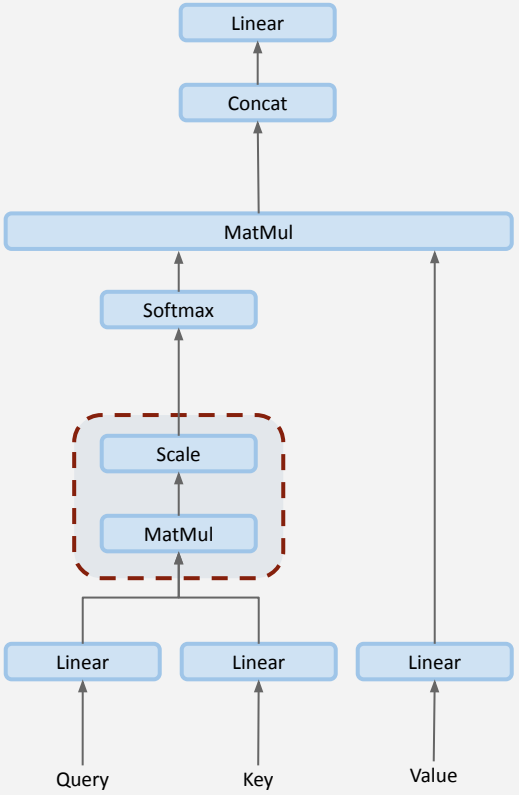


Words in the decoder should only attend to words that have been already generated.

Scaled Scores

	<SOS>	it	is	11
<SOS>	0.6	0.1	0.2	0.1
it	0.1	0.7	0.1	0.1
is	0.1	0.2	0.6	0.1
11	0.1	0.2	0.2	0.5

Self-Attention



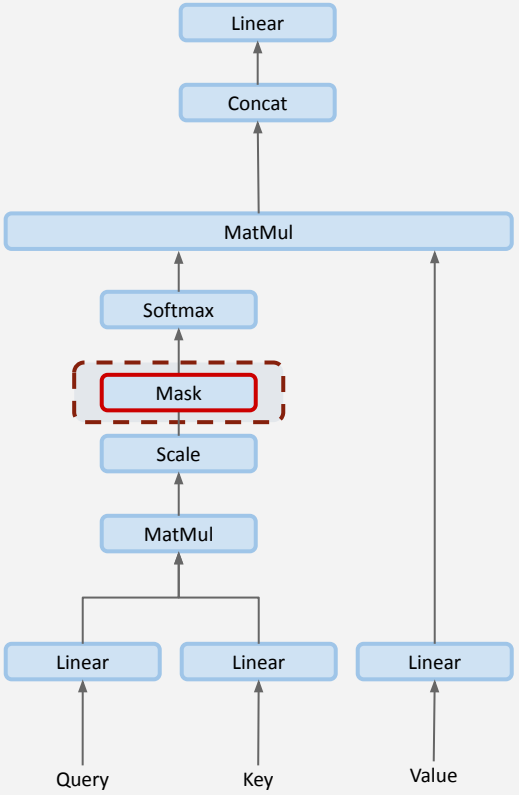
"it"can only attend to itself and the previously generated "<SOS>" (Start Of Sequence token).

Scaled Scores

	<SOS>	it	is	11
<SOS>	0.6	0.1	0.2	0.1
it	0.1	0.7	**0.1**	**0.1**
is	0.1	0.2	0.6	0.1
11	0.1	0.2	0.2	0.5

The decoder should not condition its outputs on future tokens ("**is**" and "**11**" here)

Self-Attention: Masking

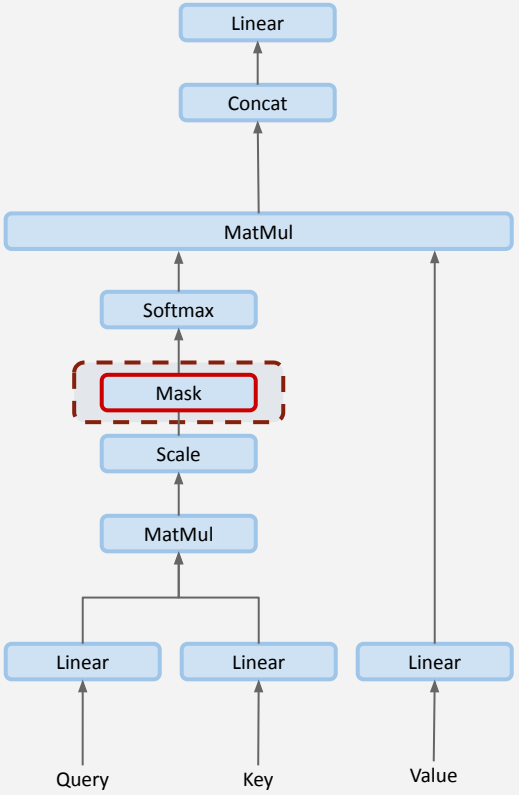


Scaled Scores

	<SOS>	it	is	11
<SOS>	0.6	**0.1**	**0.2**	**0.1**
it	0.1	0.7	**0.1**	**0.1**
is	0.1	0.2	0.6	**0.1**
11	0.1	0.2	0.2	0.5

A mask is used to zero-out the attention scores for future tokens

Self-Attention: Masking

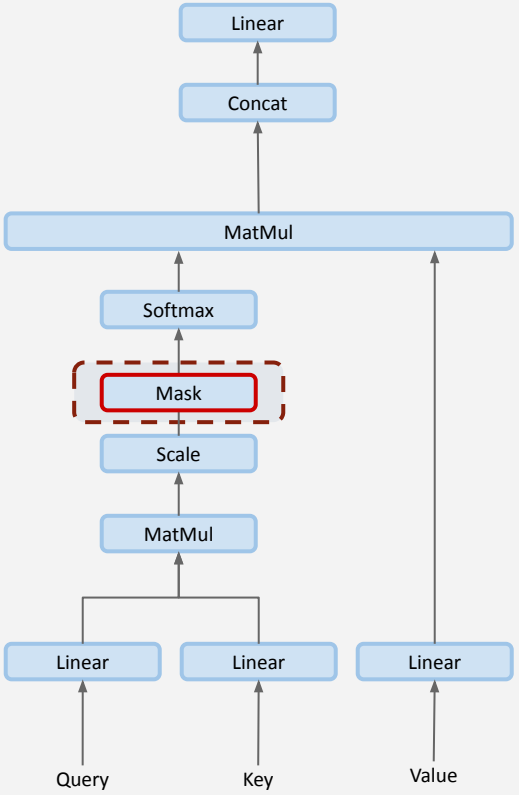


Scaled Scores

	<SOS>	it	is	11
<SOS>	0.6	-inf	-inf	-inf
it	0.1	0.7	-inf	-inf
is	0.1	0.2	0.6	-inf
11	0.1	0.2	0.2	0.5

A mask is used to zero-out the attention scores for future tokens

Self-Attention: Masking



Scaled Scores

0.6	0.1	0.2	0.1
0.1	0.7	0.1	0.1
0.1	0.2	0.6	0.1
0.1	0.2	0.2	0.5

+

Mask

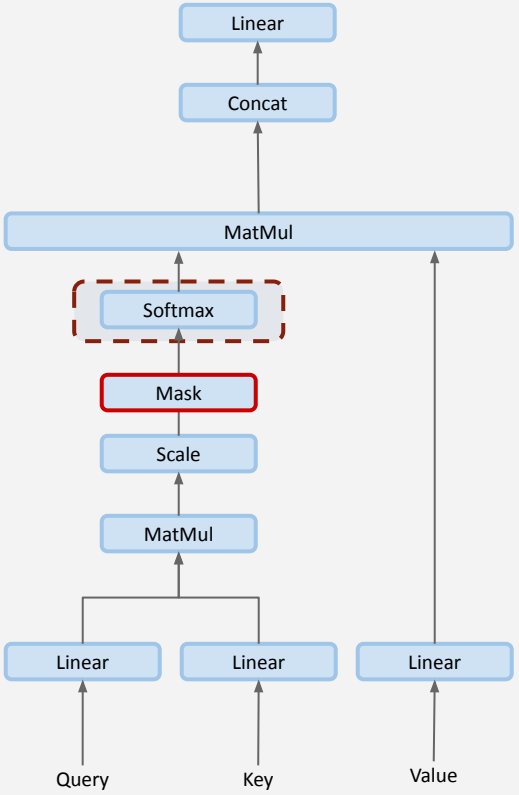
0	-inf	-inf	-inf
0	0	-inf	-inf
0	0	0	-inf
0	0	0	0

=

Masked Scores

0.6	-inf	-inf	-inf
0.1	0.7	-inf	-inf
0.1	0.2	0.6	-inf
0.1	0.2	0.2	0.5

Self-Attention: Masking



softmax (**Masked Scores**) =

0.6	-inf	-inf	-inf
0.1	0.7	-inf	-inf
0.1	0.2	0.6	-inf
0.1	0.2	0.2	0.5

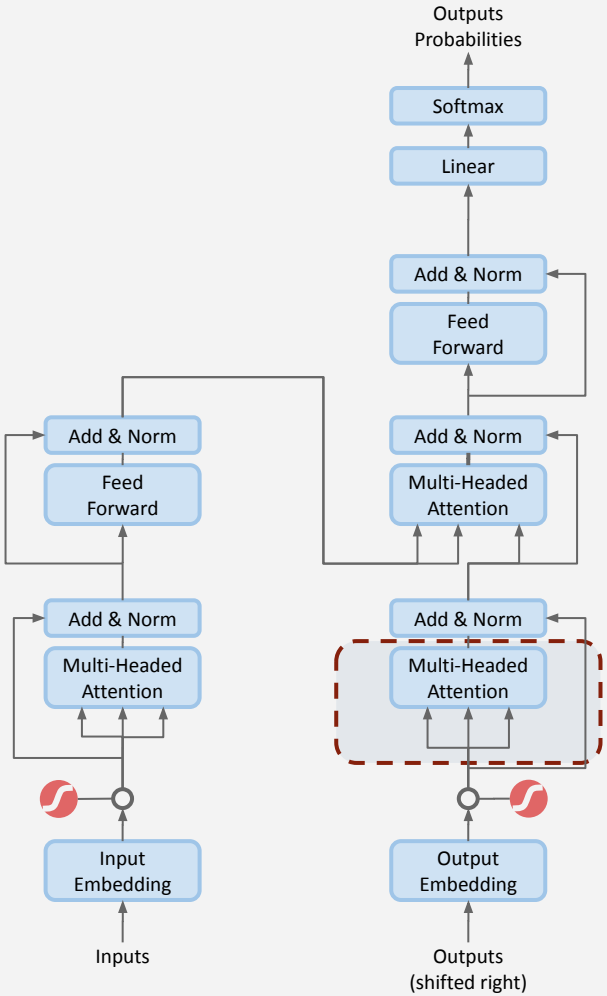
Attention Weights

	<SOS>	it	is	11
<SOS>	0.6	0	0	0
it	0.1	0.7	0	0
is	0.1	0.2	0.6	0
11	0.1	0.2	0.2	0.5

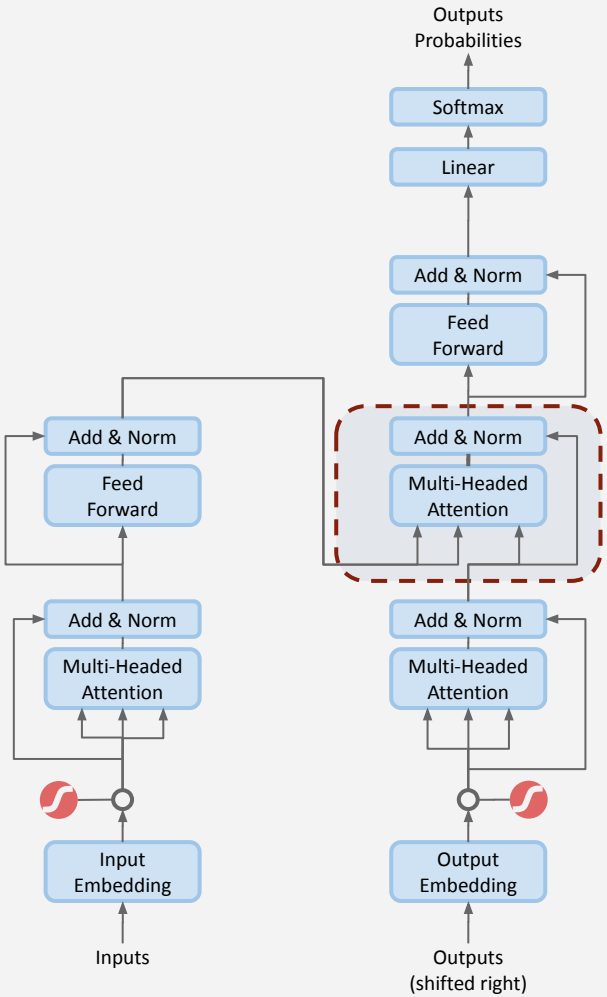
$$soft\max(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Words with zero attention weight are ignored (not attended to) by the decoder

Decoder



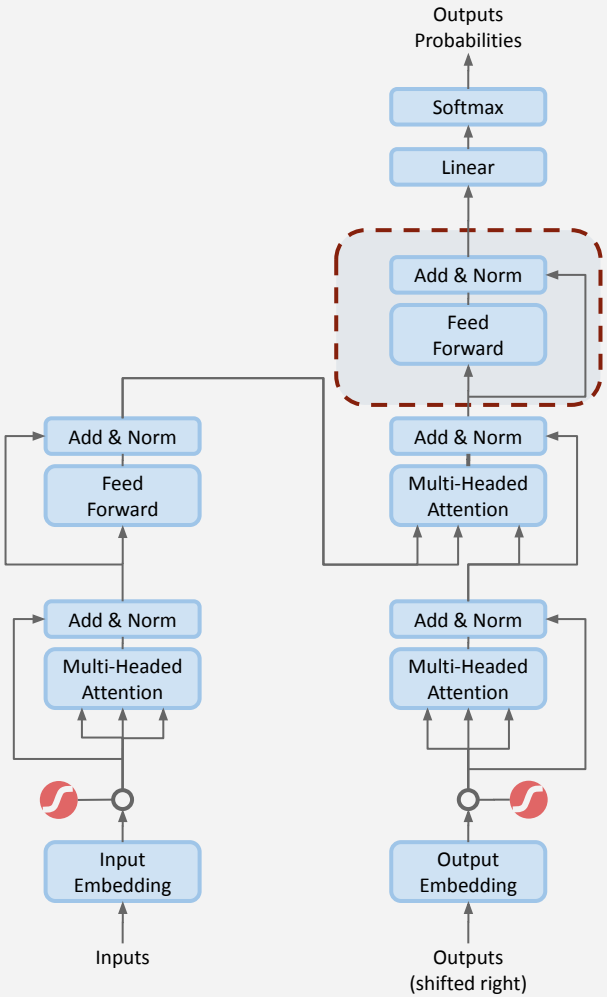
Decoder



The next multi-headed attention submodule also uses a mask like the previous layer.

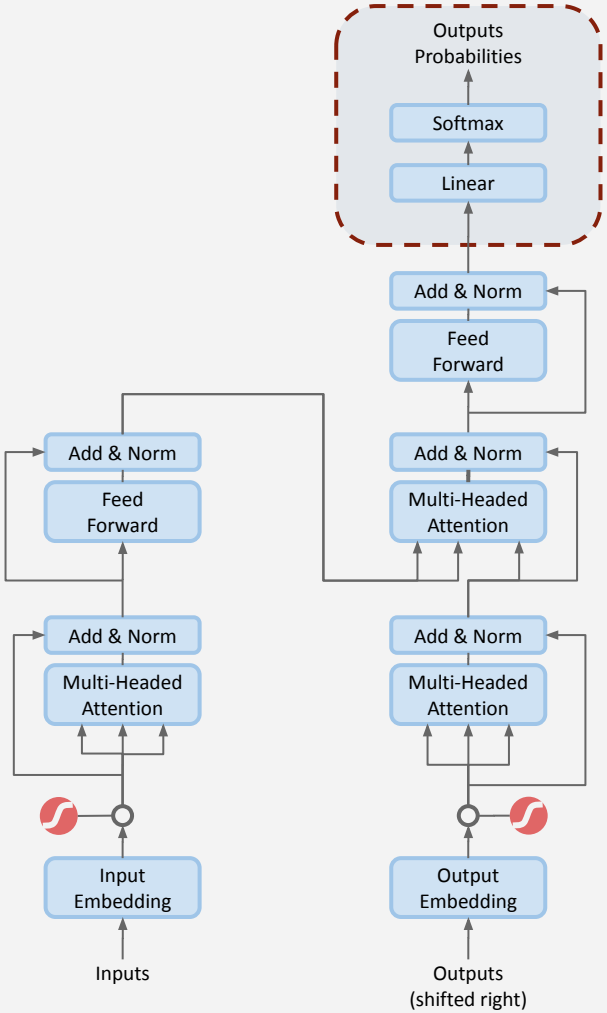
The **query** and **key** input vectors to this layer are the **encoder output**, which allows this layer to attend to words from the input sequence.

Decoder

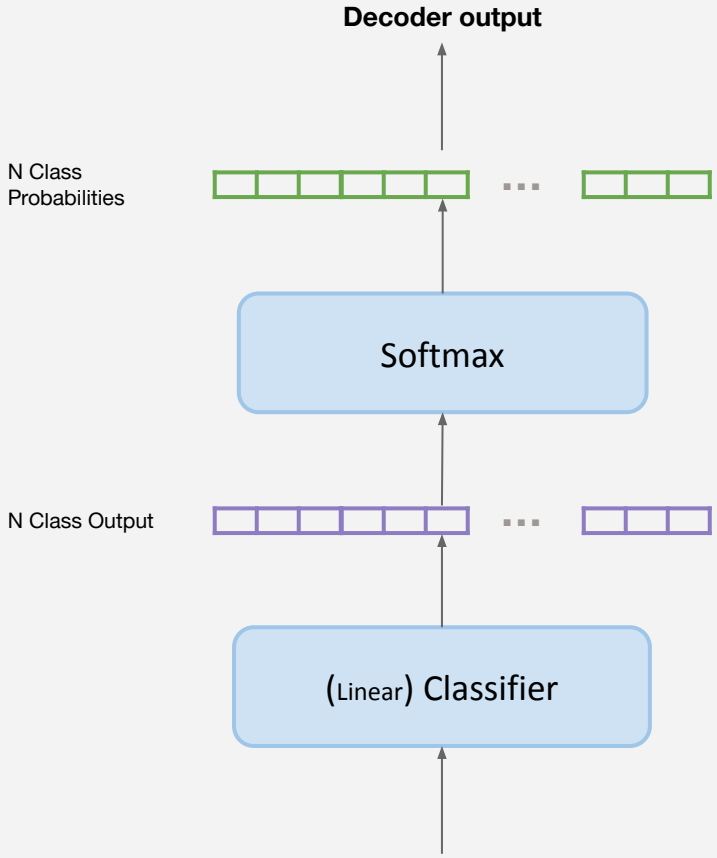


A traditional feedforward layer with a residual connection and layer normalization to process the output from the two preceding multi-headed attention layers

Decoder: Linear Classifier

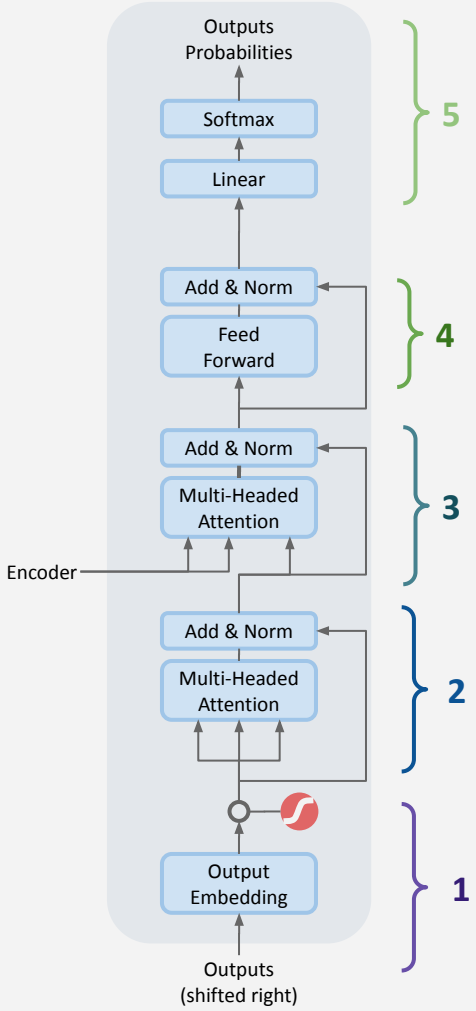


The classifier is trained to map the processed output from the preceding layers to continuous numbers for each token in the input vocabulary. The softmax layer converts the classifier output into probabilities.



Decoder

1. **Encode** output + **position** information
2. Attend to **decoder input**
3. Attend to **encoder input**
4. Process **features** with **high attention**
5. **Classify** output from previous layer

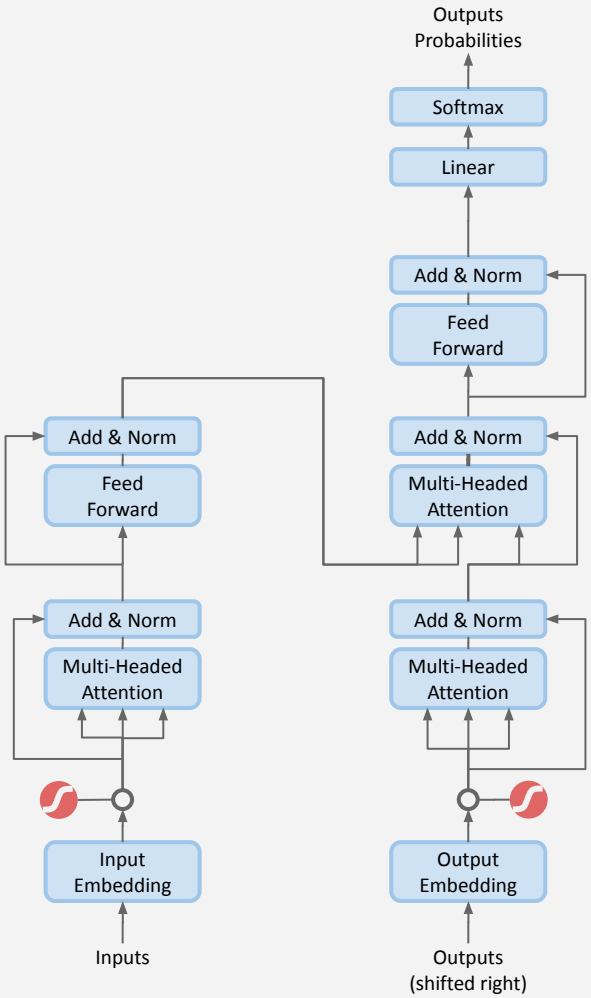


The decoder attends to its past output (it is autoregressive**)**
and to the encoder output (transformer input**)**

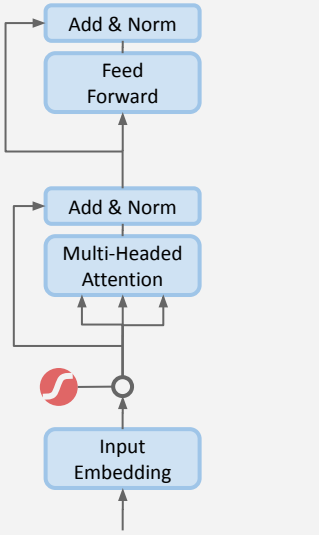
Transformer: Training

Tea and water please → Tee und wasser bitte

Transformer: Training



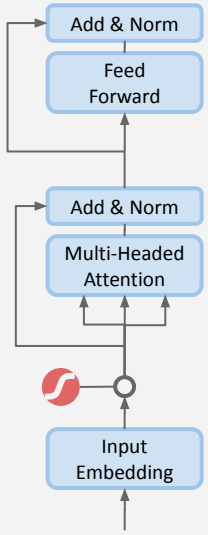
Transformer: Training



<SOS> Tea and water please <EOS>

Transformer: Training

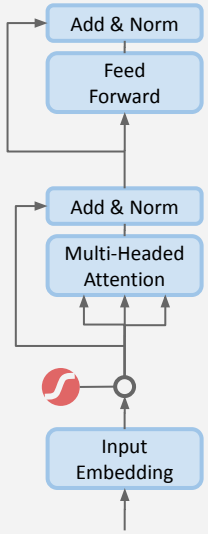
The encoder outputs a vector representation of the input that captures the **positions** and the **semantic relationships** using the multi-head attention.



<SOS> Tea and water please <EOS>

Transformer: Training

The encoder outputs a vector representation of the input that captures the **positions** and the **semantic relationships** using the multi-head attention.

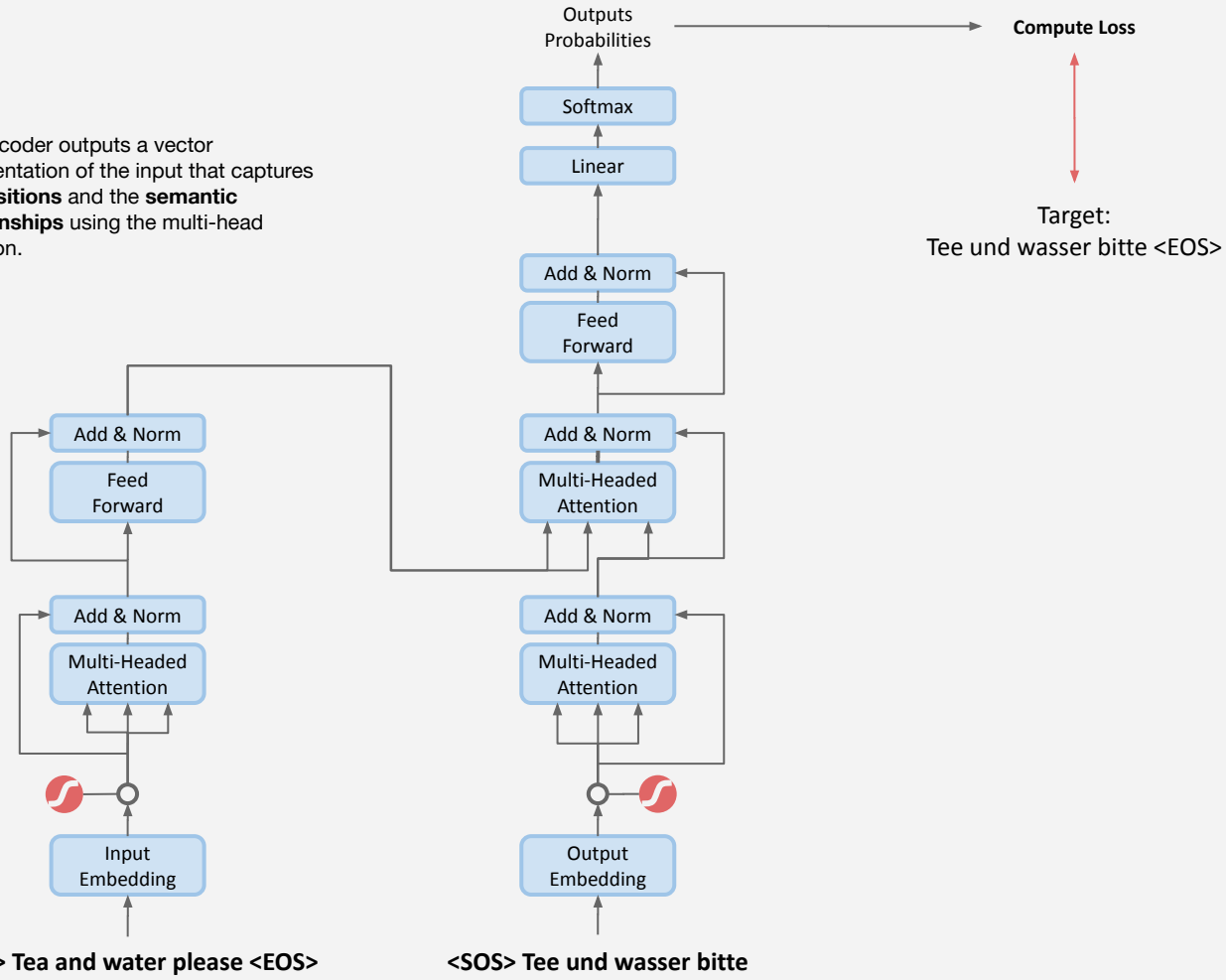


<SOS> Tea and water please <EOS>

<SOS> Tee und wasser bitte

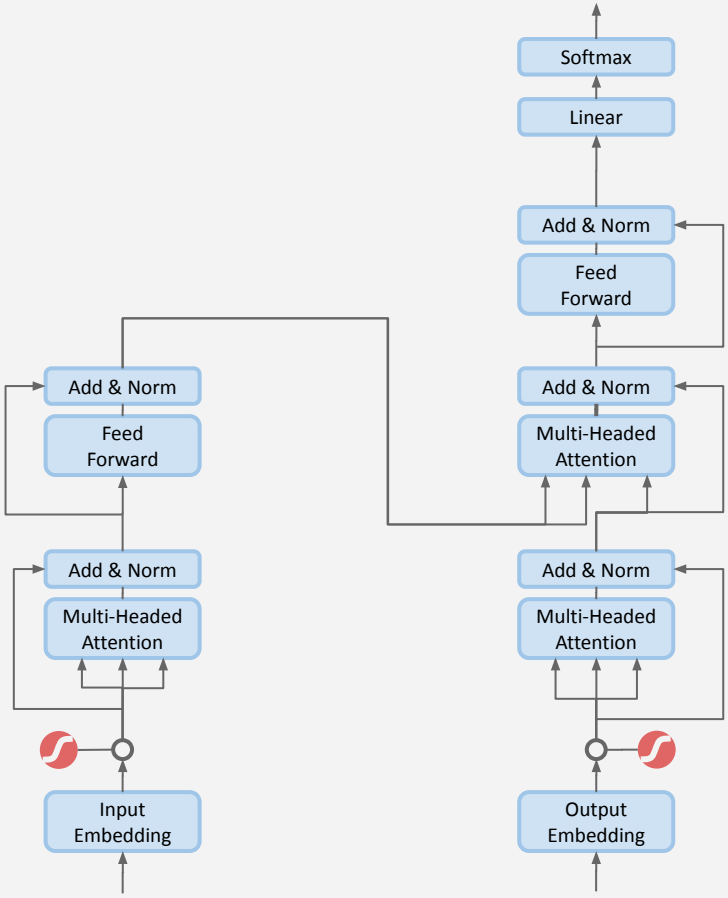
Transformer: Training

The encoder outputs a vector representation of the input that captures the **positions** and the **semantic relationships** using the multi-head attention.



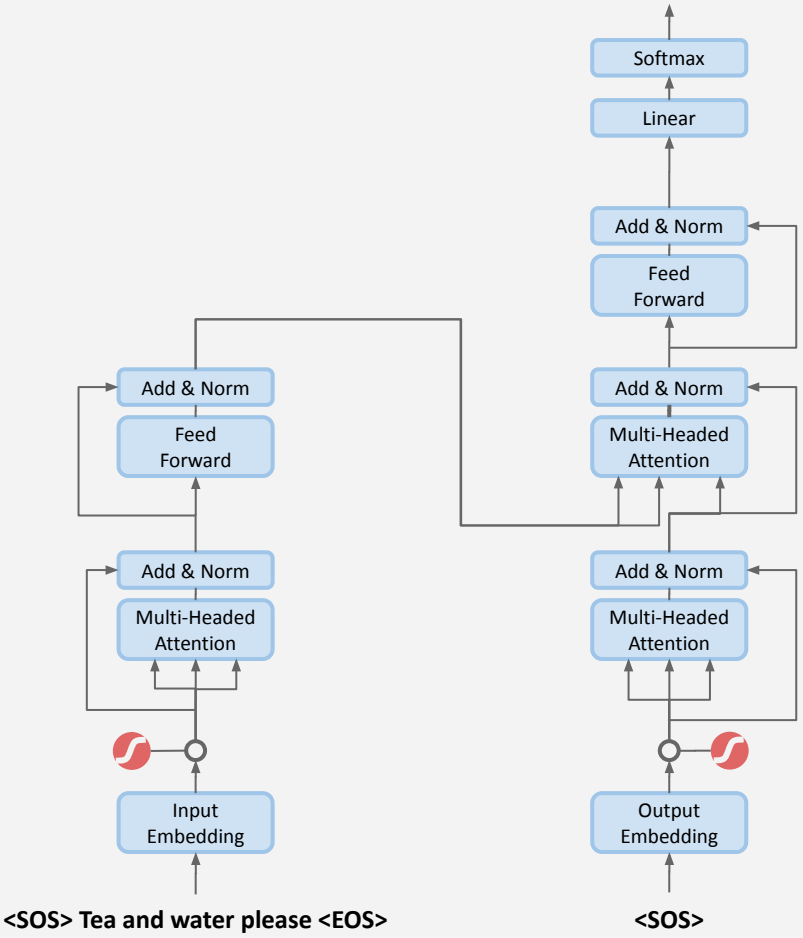
Transformer: Inference

Transformer: Inference

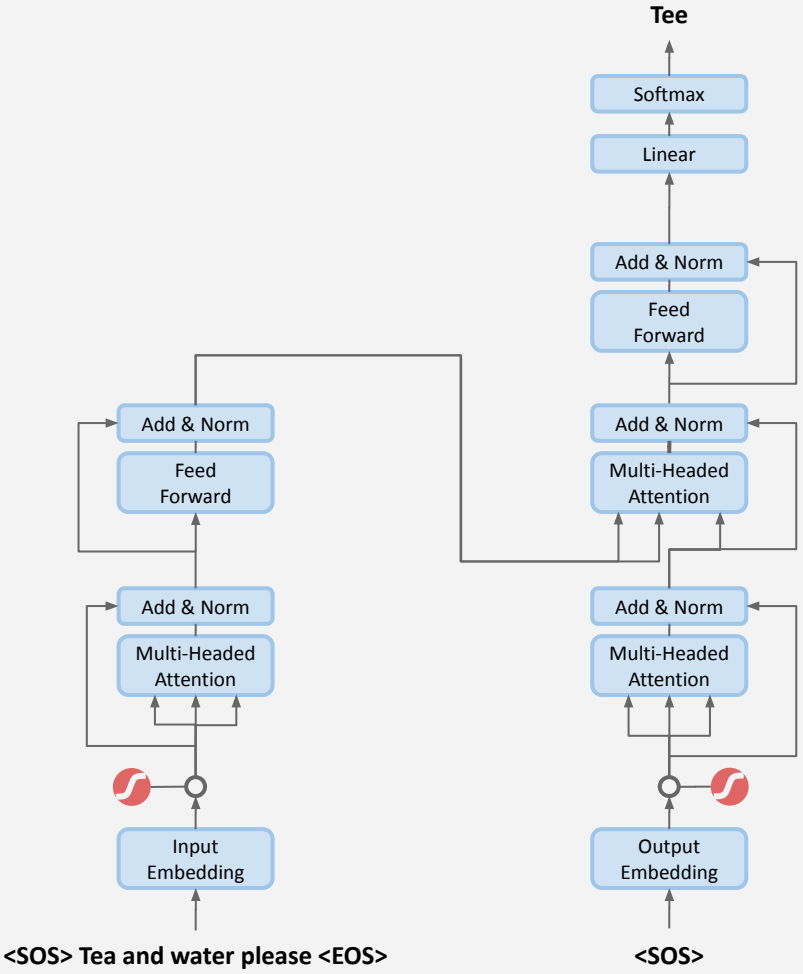


<SOS> Tea and water please <EOS>

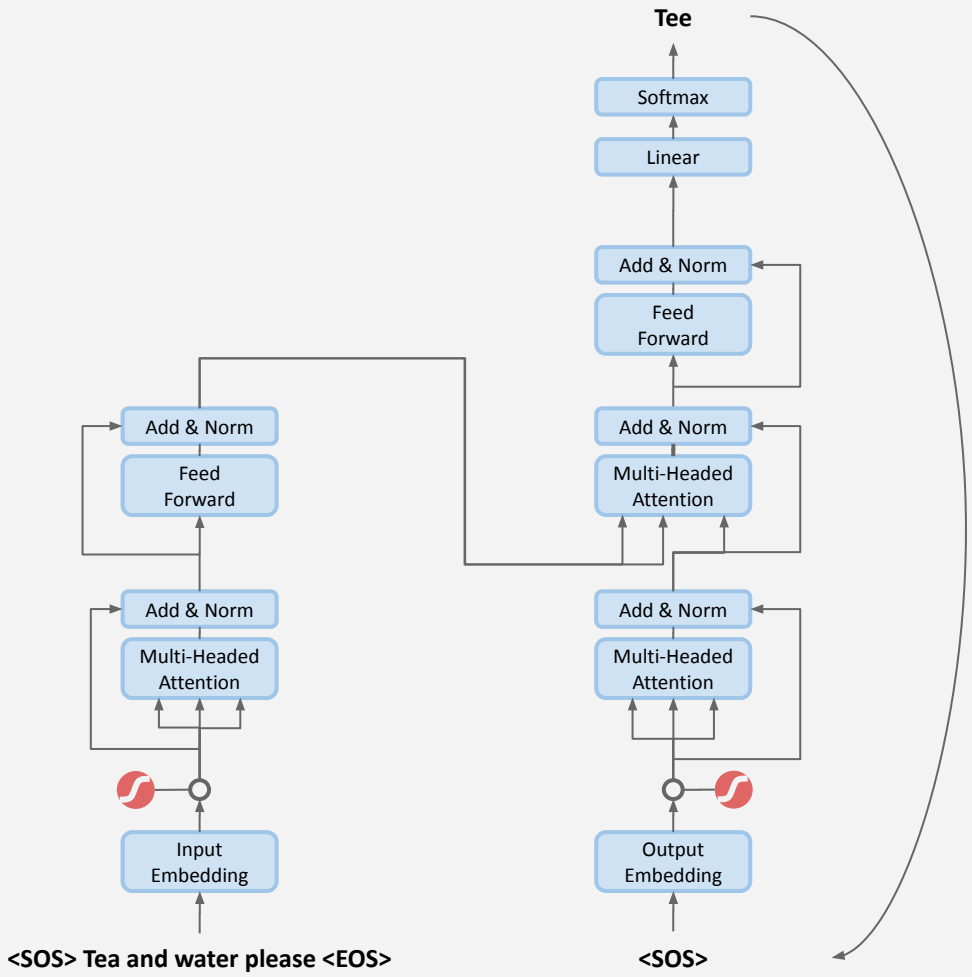
Transformer: Inference



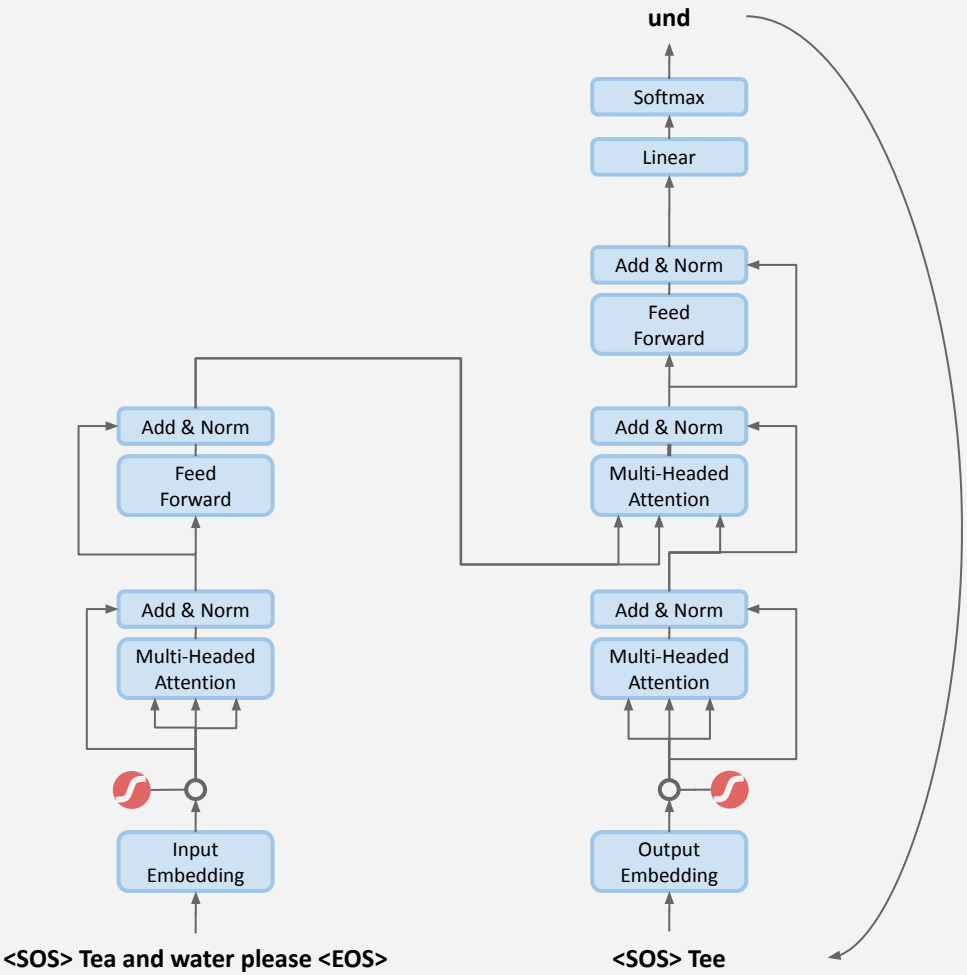
Transformer: Inference



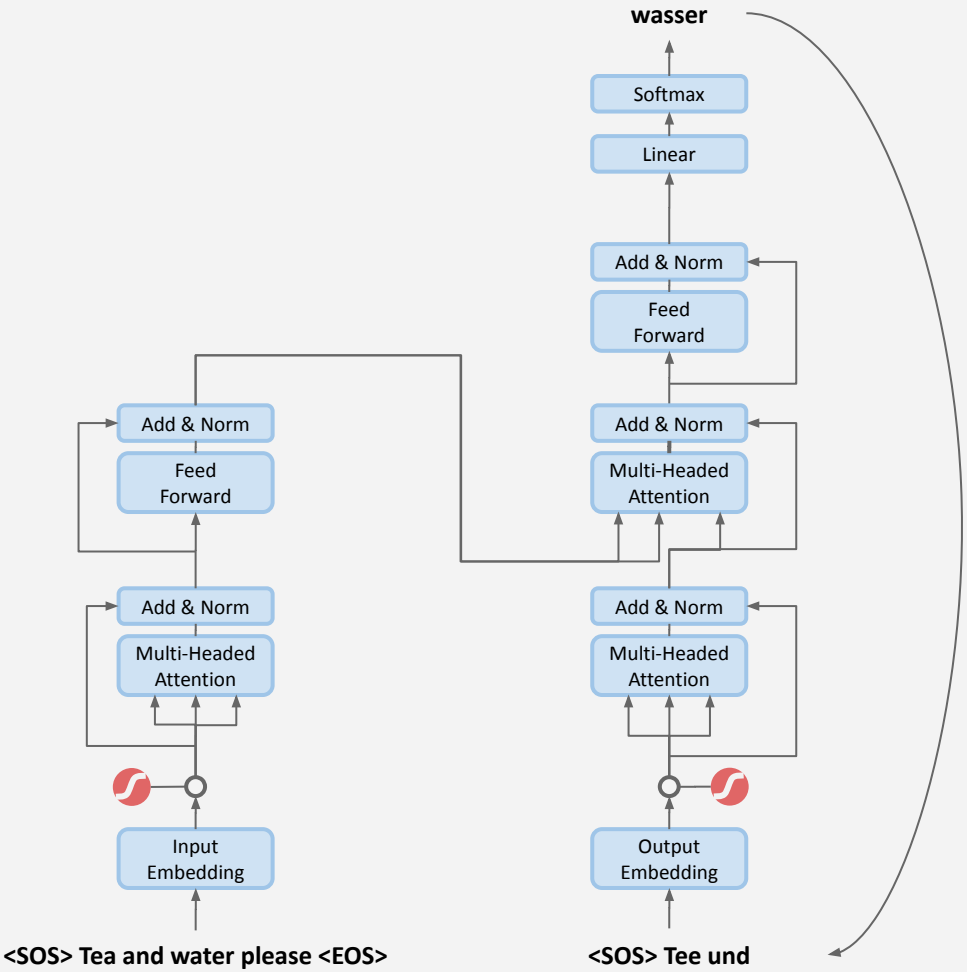
Transformer: Inference



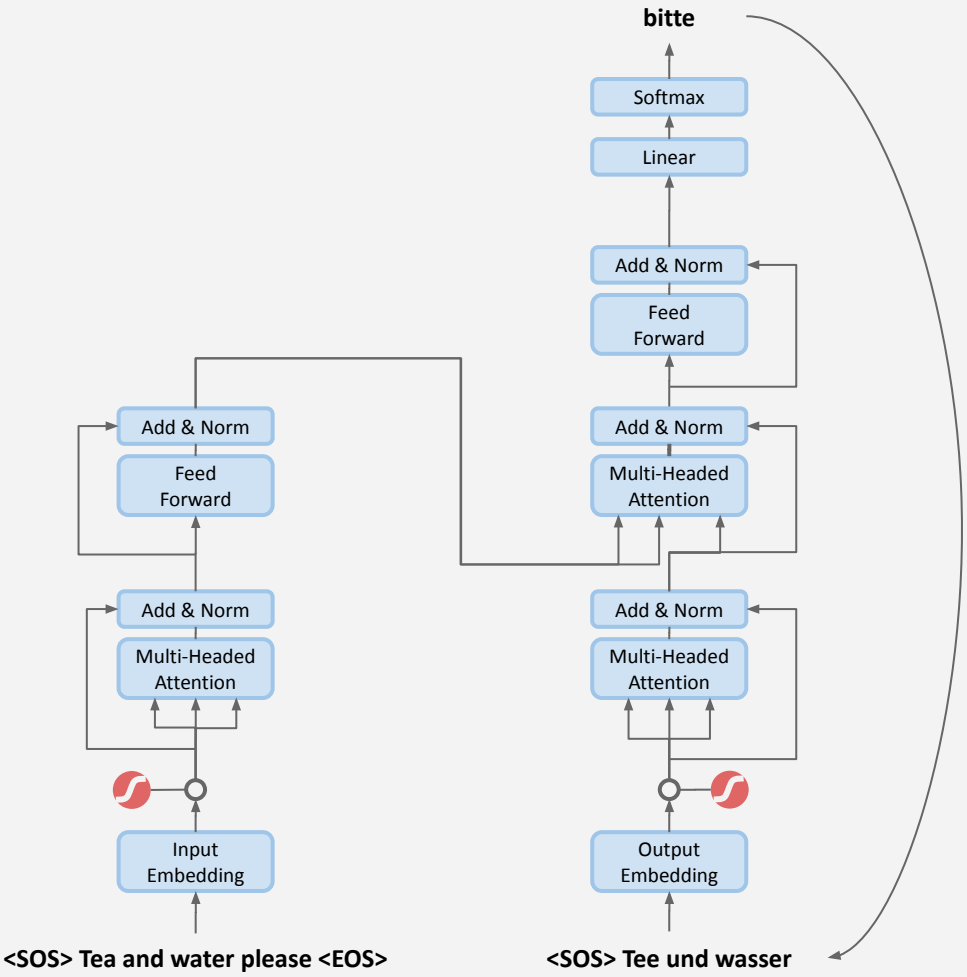
Transformer: Inference



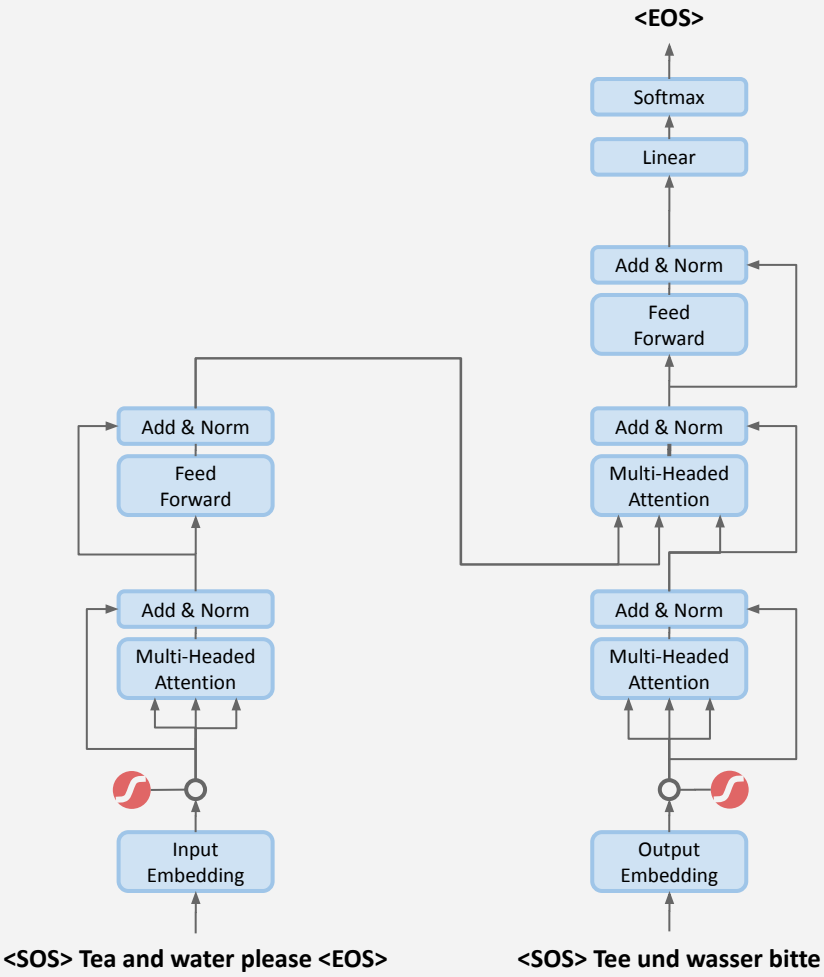
Transformer: Inference



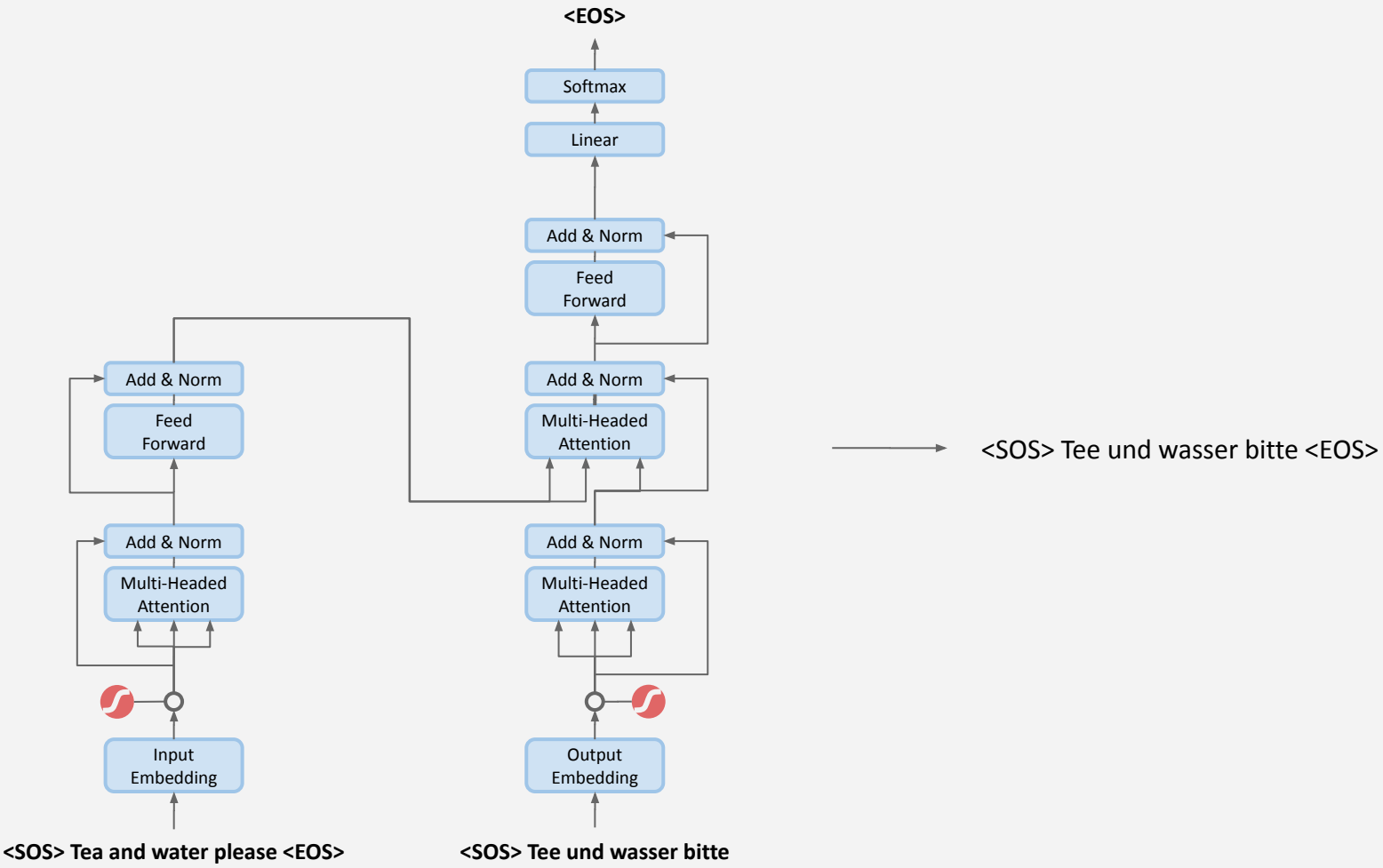
Transformer: Inference



Transformer: Inference



Transformer: Inference



Transformer: Inference Strategies

1. **Greedy**

   Select the token (word) with the highest probability in the final softmax layer

   + Easy to implement and efficient

2. **Beam Search**

   At each step, consider a fixed number (beam width) of likely candidates

   Maintain a tree of partially decoded sequences (beam)

   Select the sequence with the highest joint probability

   + Better exploration
   + High output quality

Attention and Transformers

Language Models

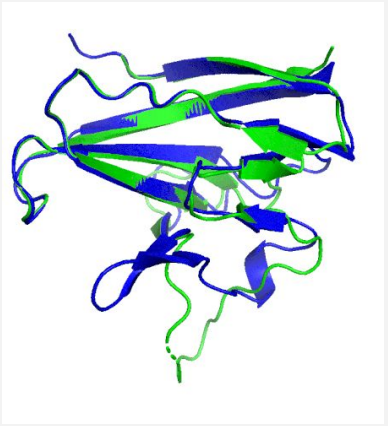
An armchair in the shape of an avocado

GPT, LLaMA

Devlin et al., SAACL 2019

Touvron et al., 2023

Protein Sequencing



AlphaFold2

Jumper et al., Nature 2021

Computer Vision

Vision Transformer

Yang et al., NeurIPS 2021