ROB 537
Learning-Based Control

Week 7, Lecture 1

Transformers and Attention Mechanism

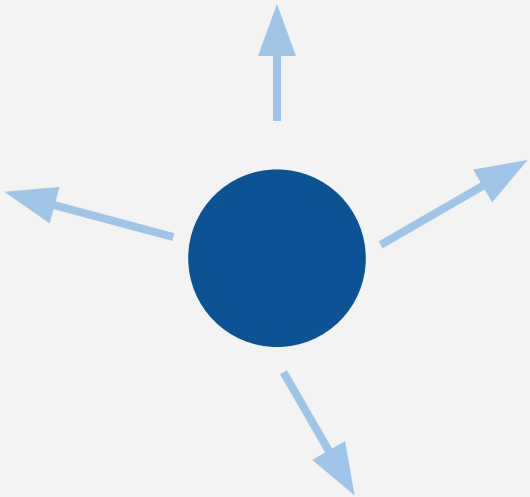
HW **m** due on **n/o 11:59 PM**

Robotics Seminars: 10AM Fridays

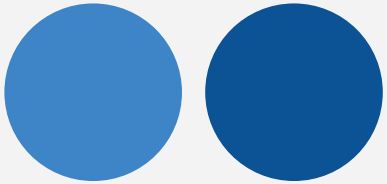
Oregon State University

Given this image of a ball, can you predict where it goes next?

Oregon State University



Given this image of a ball, can you predict where it goes next?



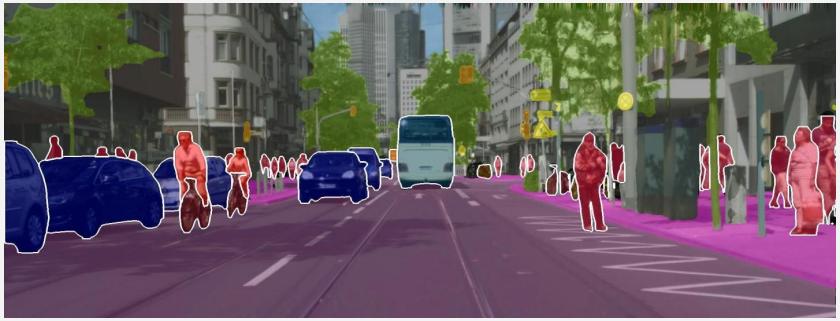
Given this image of a ball, can you predict where it goes next?

Oregon State University

Given this image of a ball, can you predict where it goes next?

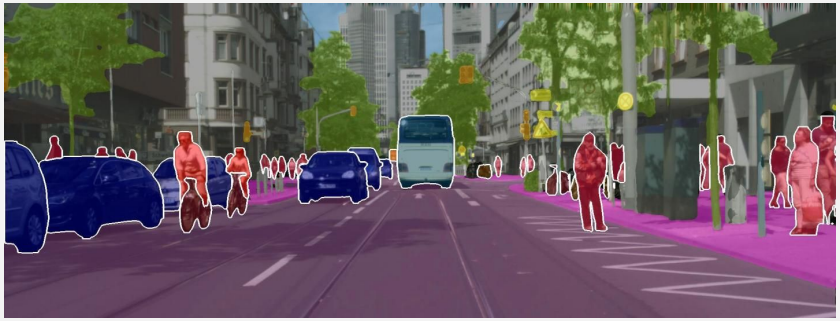
Given this image of a ball, can you predict where it goes next?

Given this image of a ball, can you predict where it goes next?

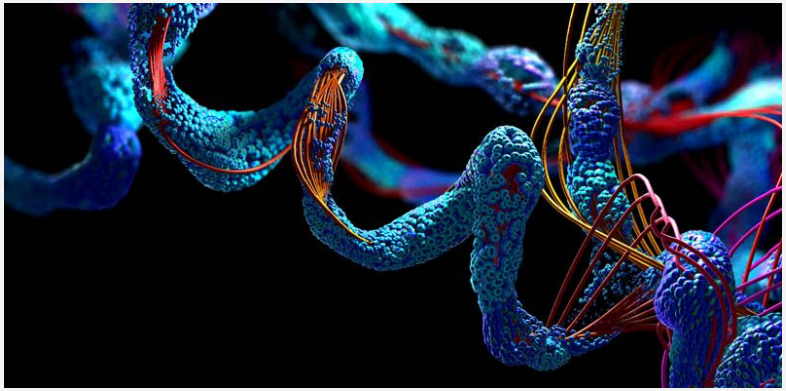


eurekalert

Sequences in the Wild



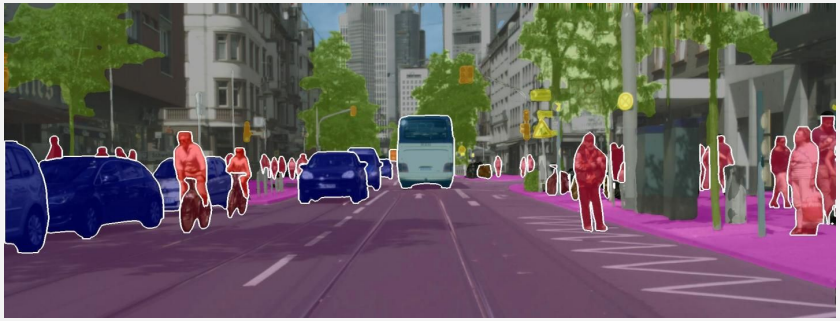
eurekalert



AWS - BERT

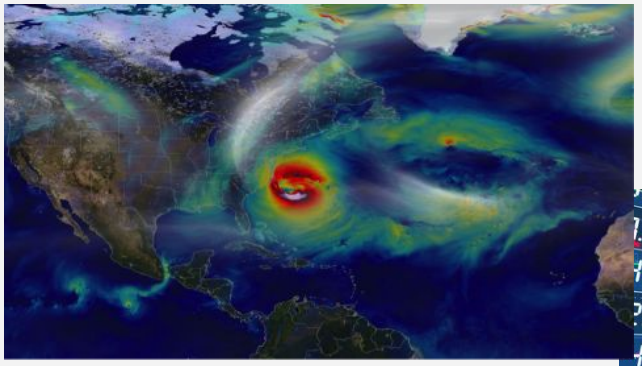
Sequences in the Wild

Oregon State University



eurekalert

AWS - BERT



NASA

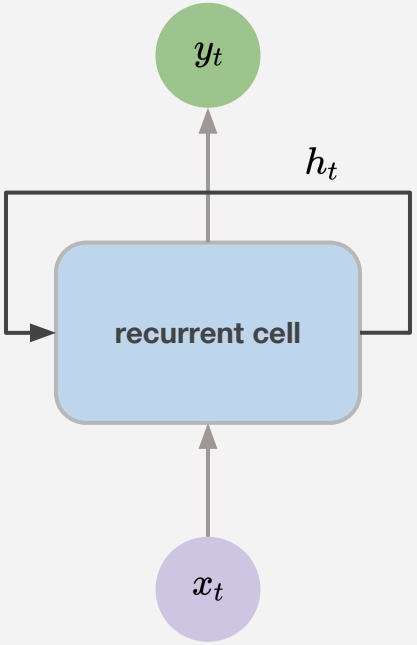
Adobe

Adobe

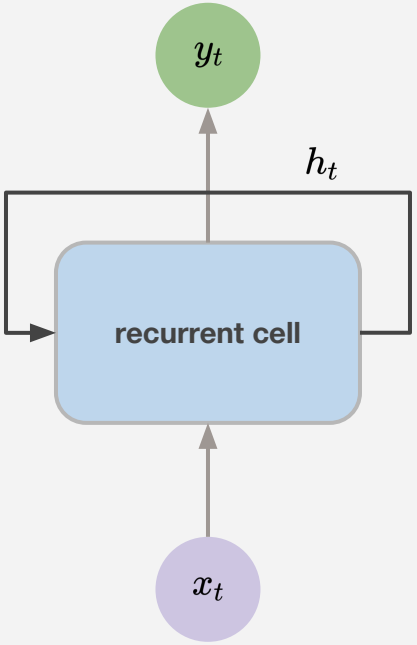
Sequences in the Wild

Oregon State University

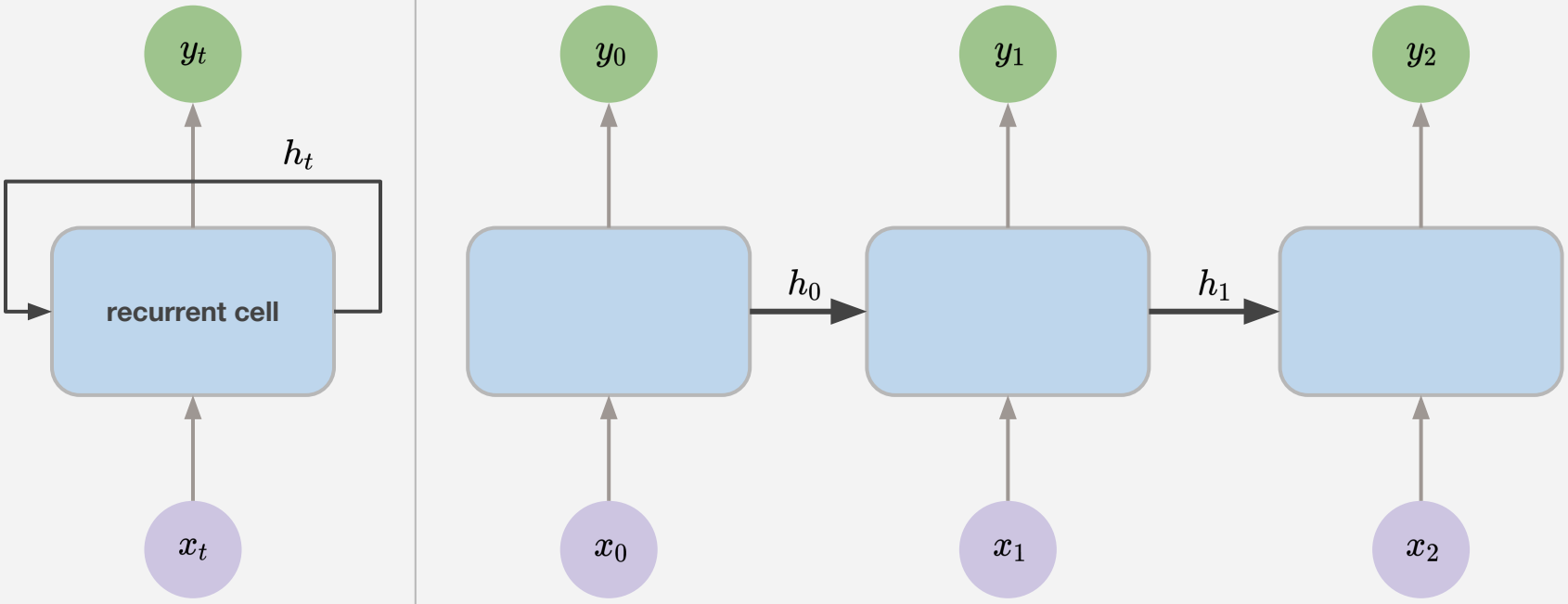
Recurrence



Recurrence



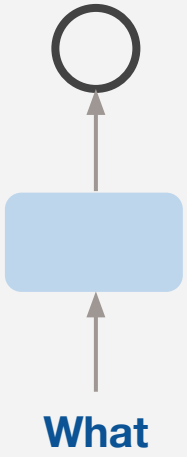
Recurrence



RNNs: Feeding Sequential Data Example

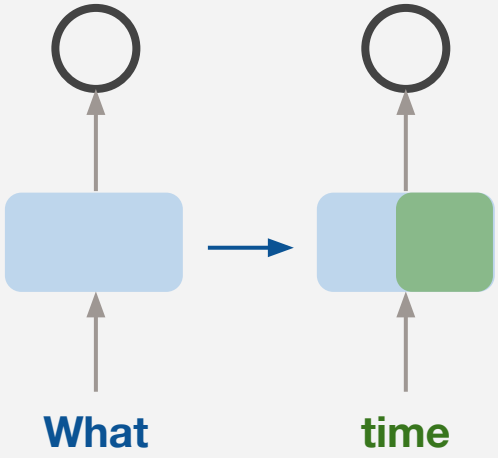
What time is it ?

RNNs: Feeding Sequential Data Example



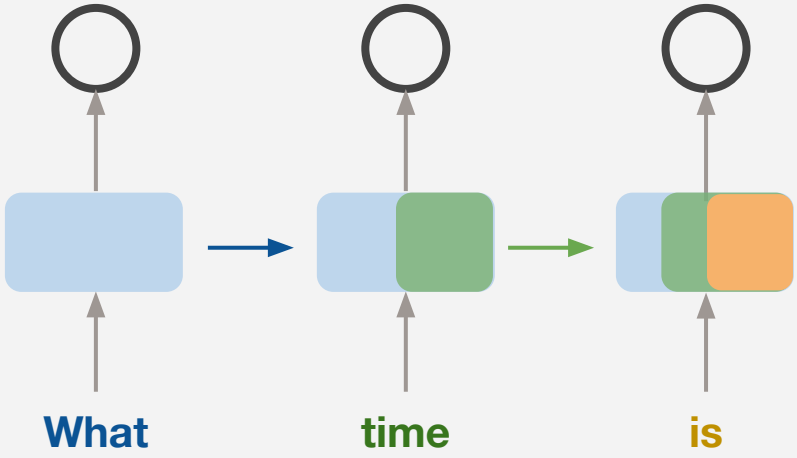
What time is it ?

RNNs: Feeding Sequential Data Example



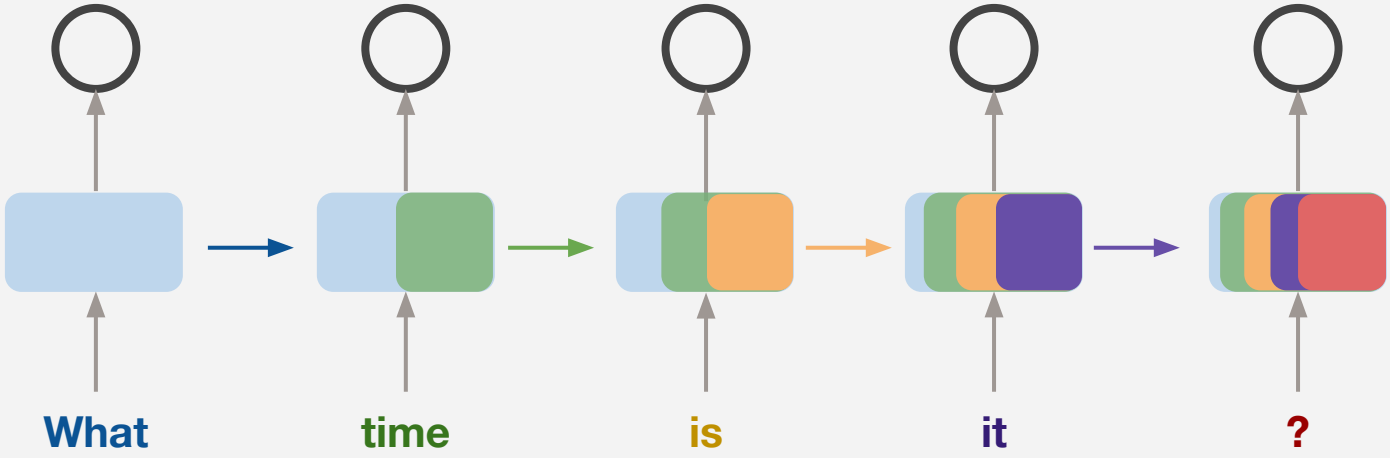
What time is it ?

RNNs: Feeding Sequential Data Example



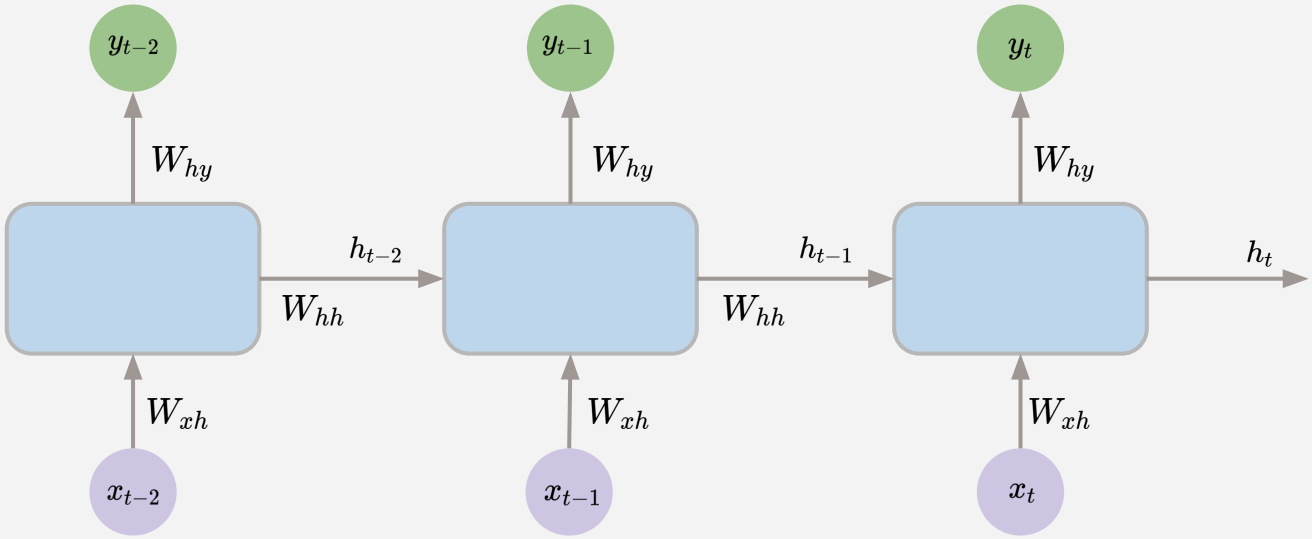
What time is it ?

RNNs: Feeding Sequential Data Example

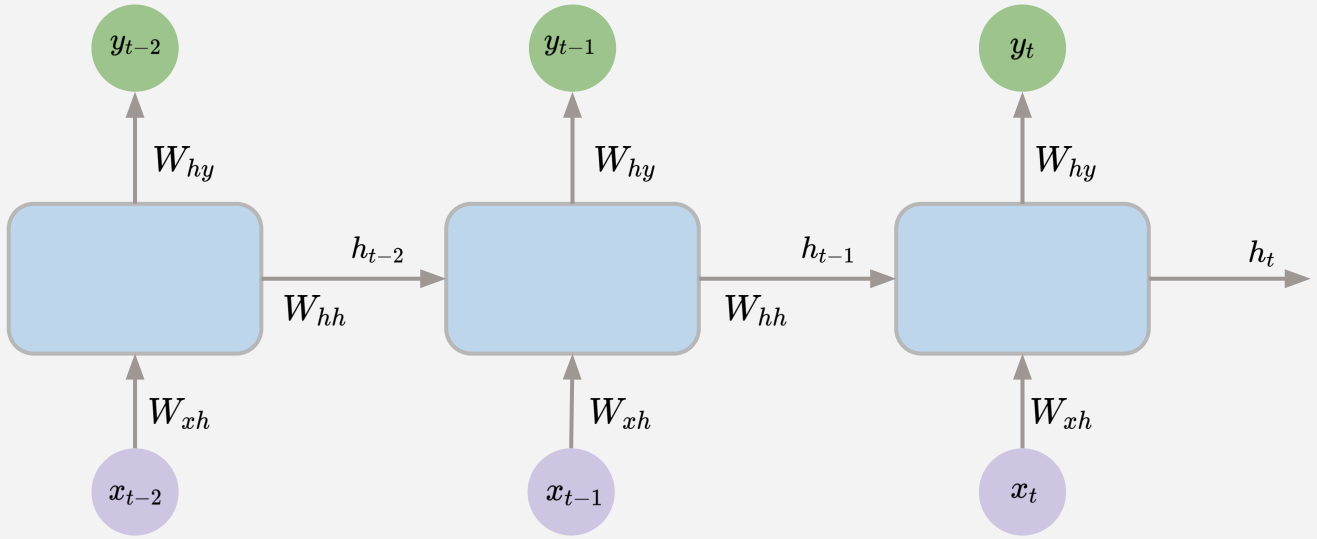


What time is it ?

RNNs: Forward Pass

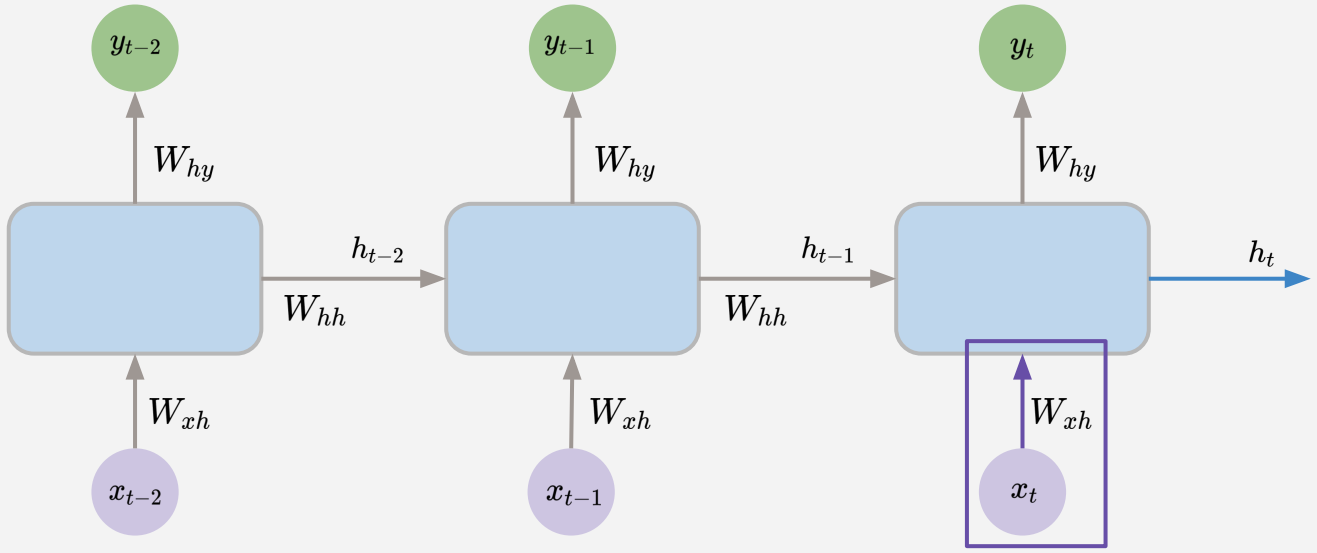


RNNs: Forward Pass



The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

RNNs: Forward Pass

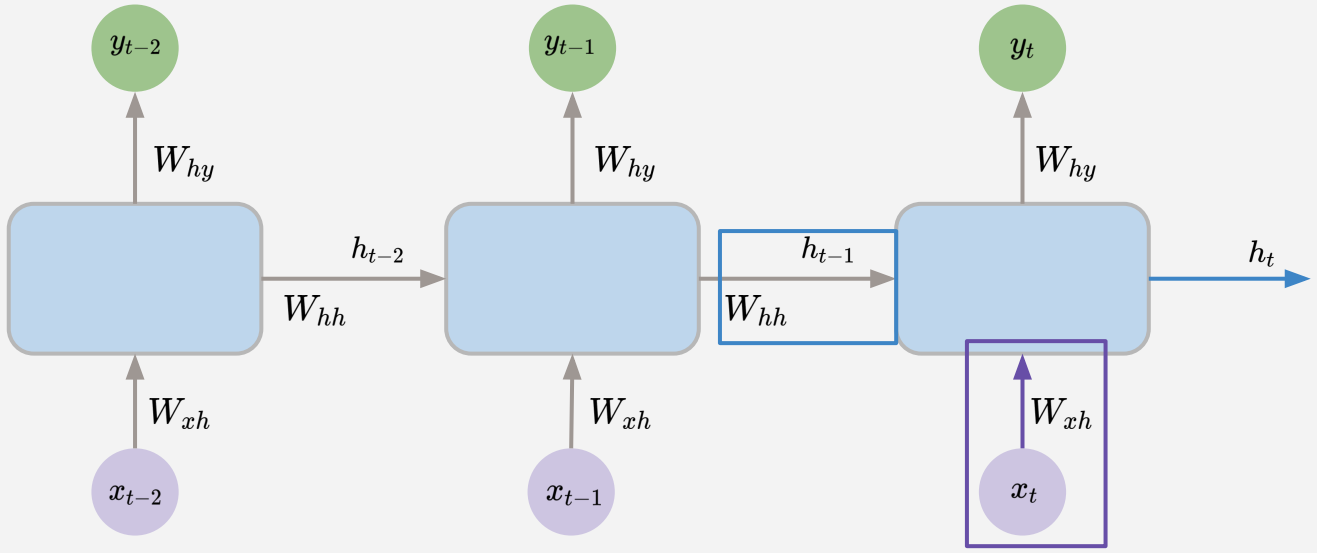


The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

$$h_t = \tanh\left(\boxed{W_{xh} \cdot x_t} + W_{hh} \cdot h_{t-1} + b_t \right)$$

weighted input

RNNs: Forward Pass



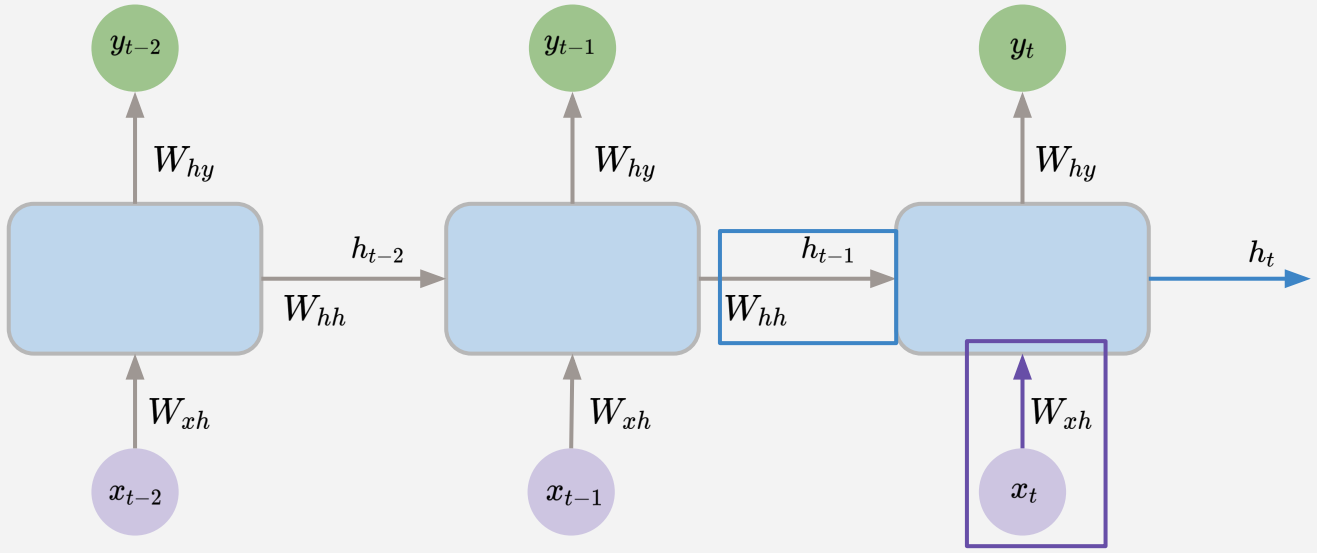
The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

$$h_t = \tanh\left(\boxed{W_{xh} \cdot x_t} + \boxed{W_{hh} \cdot h_{t-1}} + b_t \right)$$

weighted input

weighted previous hidden state

RNNs: Forward Pass

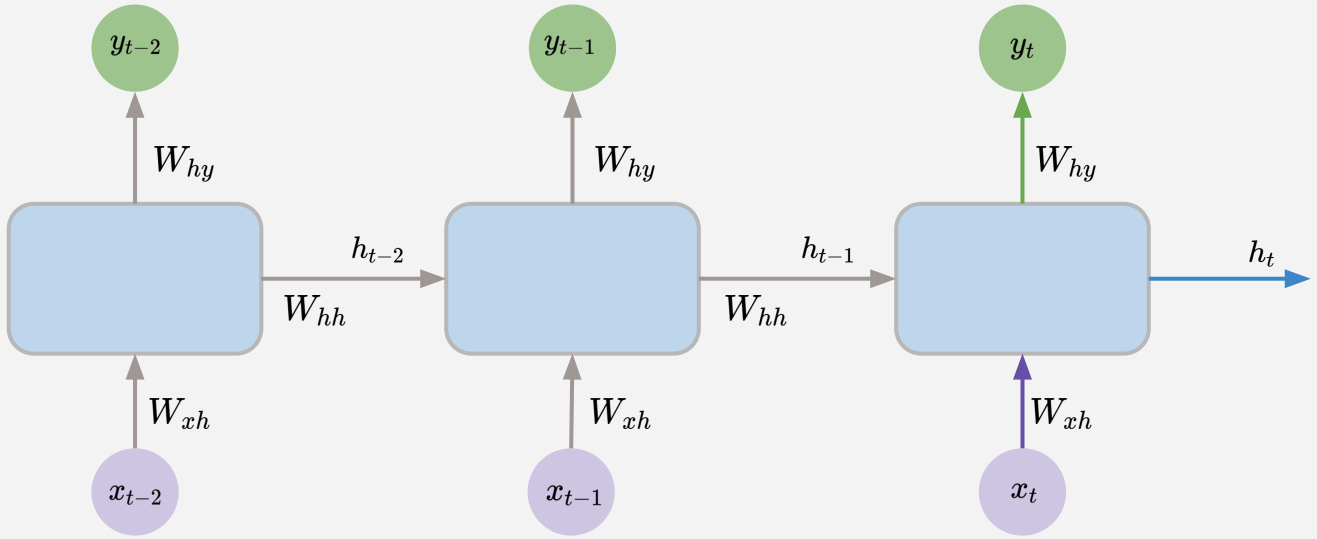


The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

$$h_t = \tanh\left(\boxed{W_{xh} \cdot x_t} + \boxed{W_{hh} \cdot h_{t-1}} + \boxed{b_t} \right)$$

weighted input — **weighted previous hidden state** — **bias**

RNNs: Forward Pass



The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

$$h_t = \tanh\left(\boxed{W_{xh} \cdot x_t} + \boxed{W_{hh} \cdot h_{t-1}} + \boxed{b_t} \right)$$

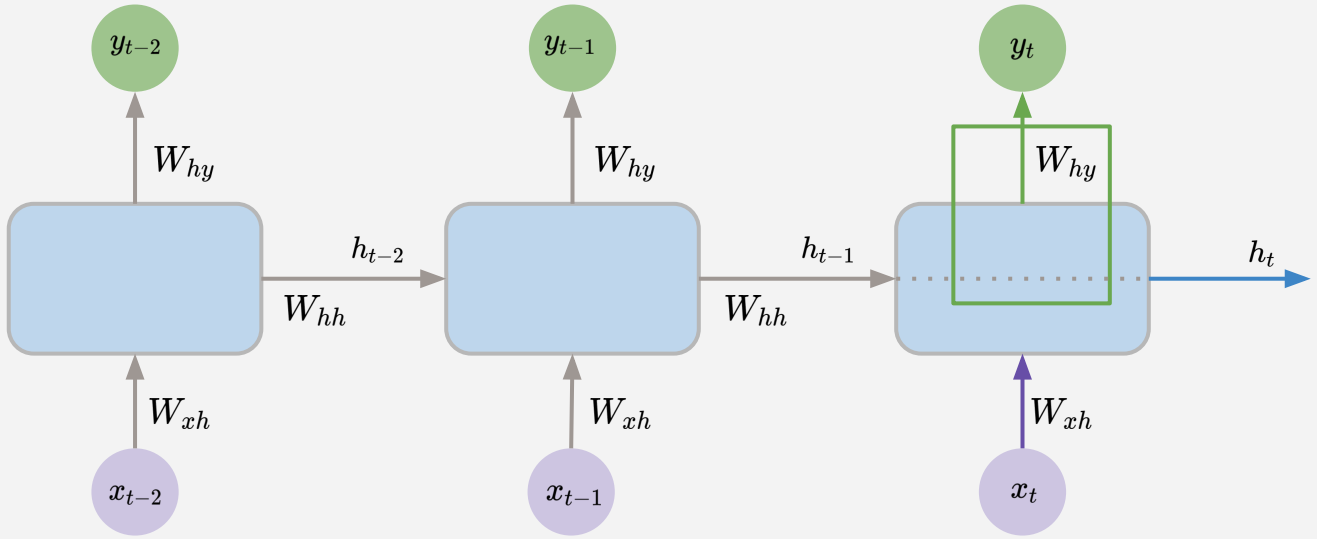
weighted input

weighted previous hidden state

bias

$$y_t = \sigma\left(W_{hy} \cdot h_t + b_y \right)$$

RNNs: Forward Pass



The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

$$h_t = \tanh\left(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_t \right)$$

weighted input

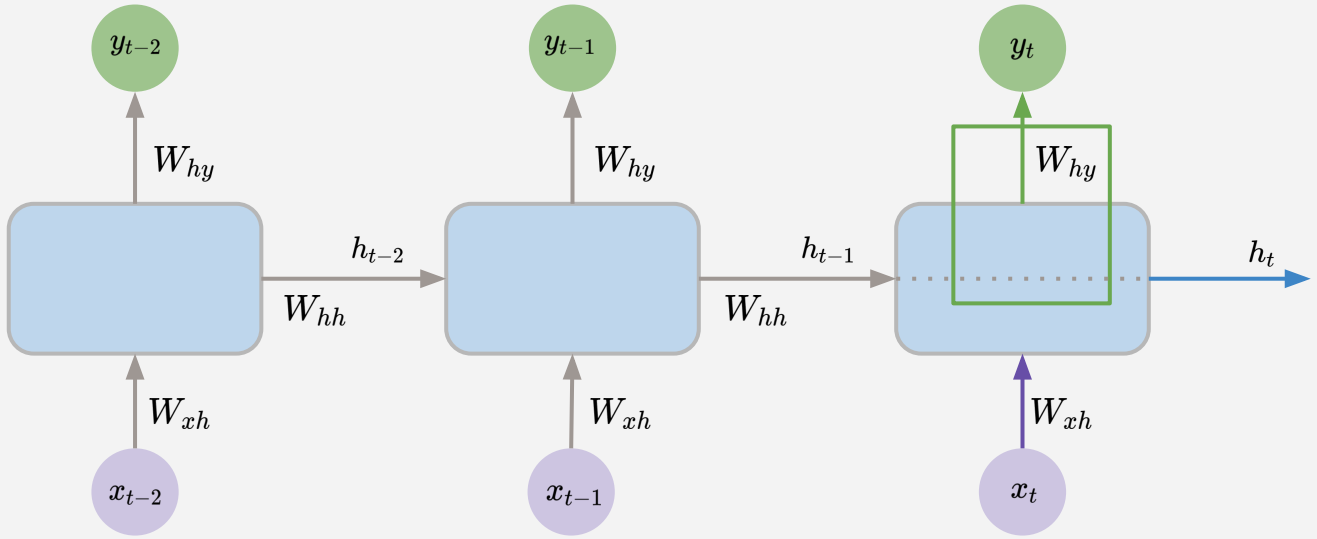
weighted previous hidden state

bias

$$y_t = \sigma\left(W_{hy} \cdot h_t + b_y \right)$$

weighted hidden state

RNNs: Forward Pass



The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

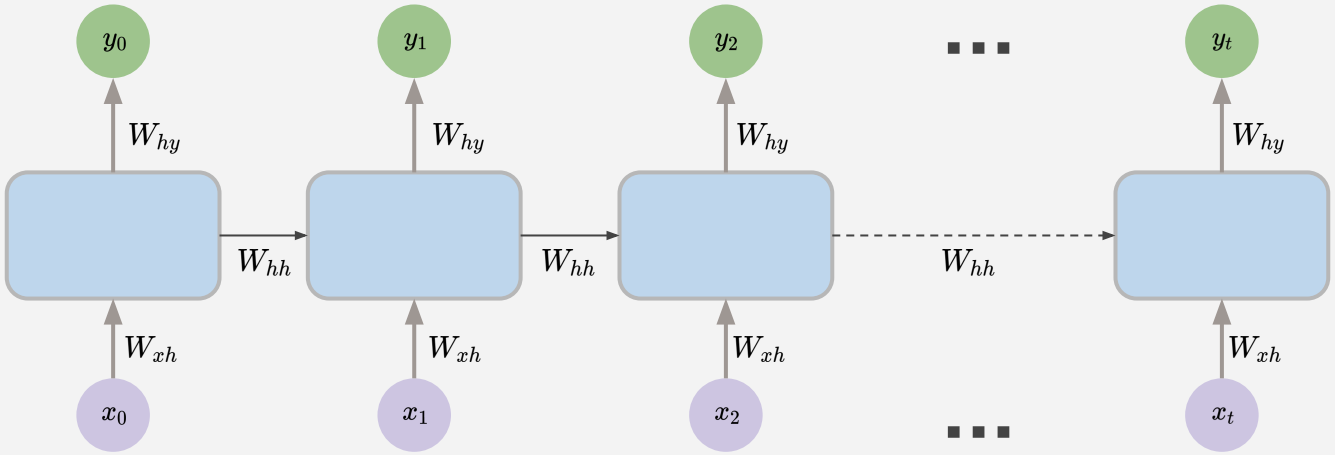
$$h_t = \tanh\left(\boxed{W_{xh} \cdot x_t} + \boxed{W_{hh} \cdot h_{t-1}} + \boxed{b_t}\right)$$

weighted input — weighted previous hidden state — bias

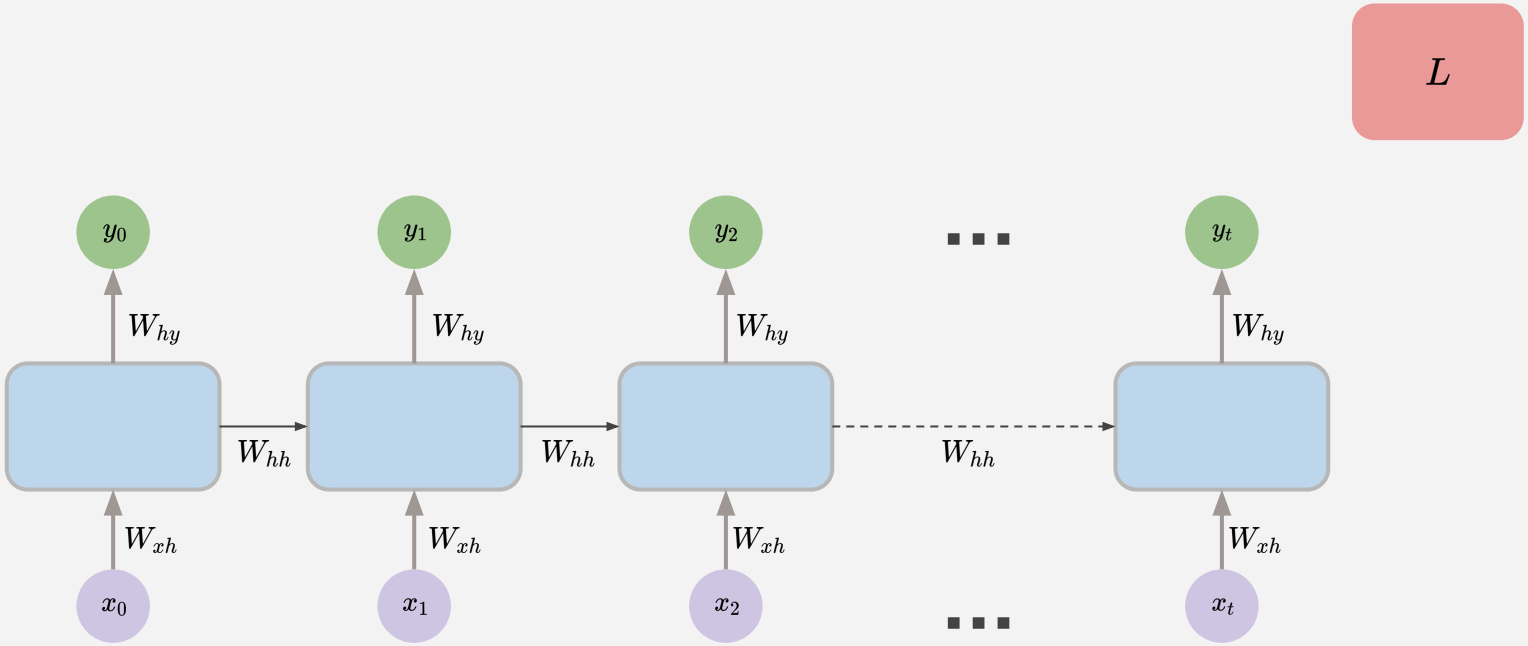
$$y_t = \sigma\left(\boxed{W_{hy} \cdot h_t} + \boxed{b_y}\right)$$

weighted hidden state — bias

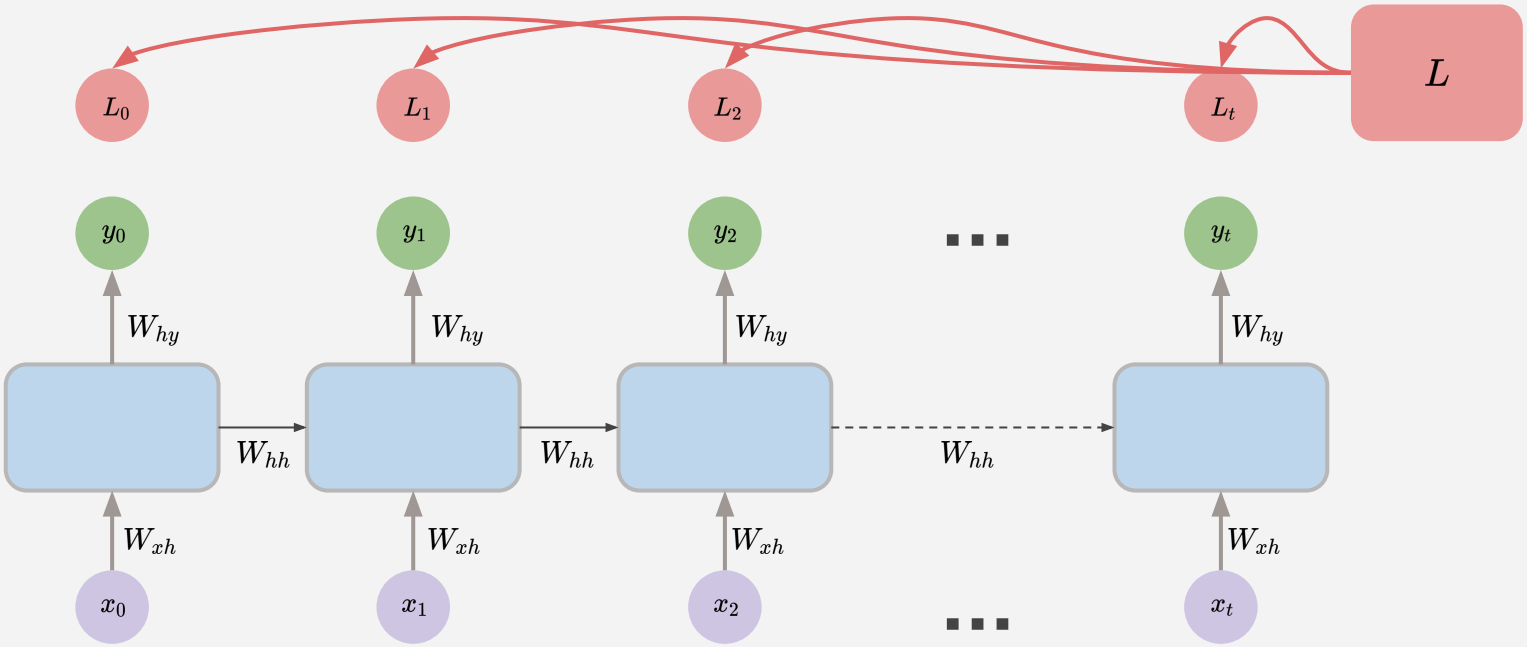
RNNs: Backpropagation Through Time (BPTT)



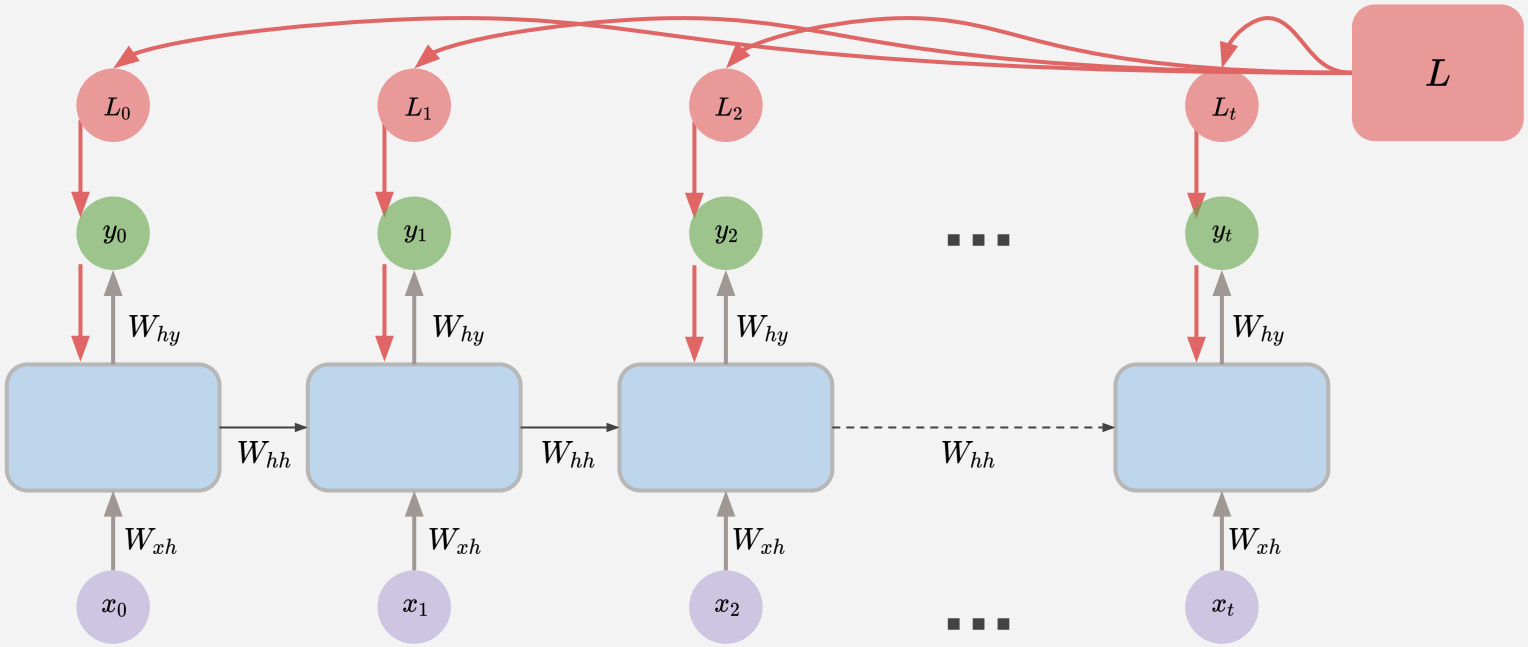
RNNs: Backpropagation Through Time (BPTT)



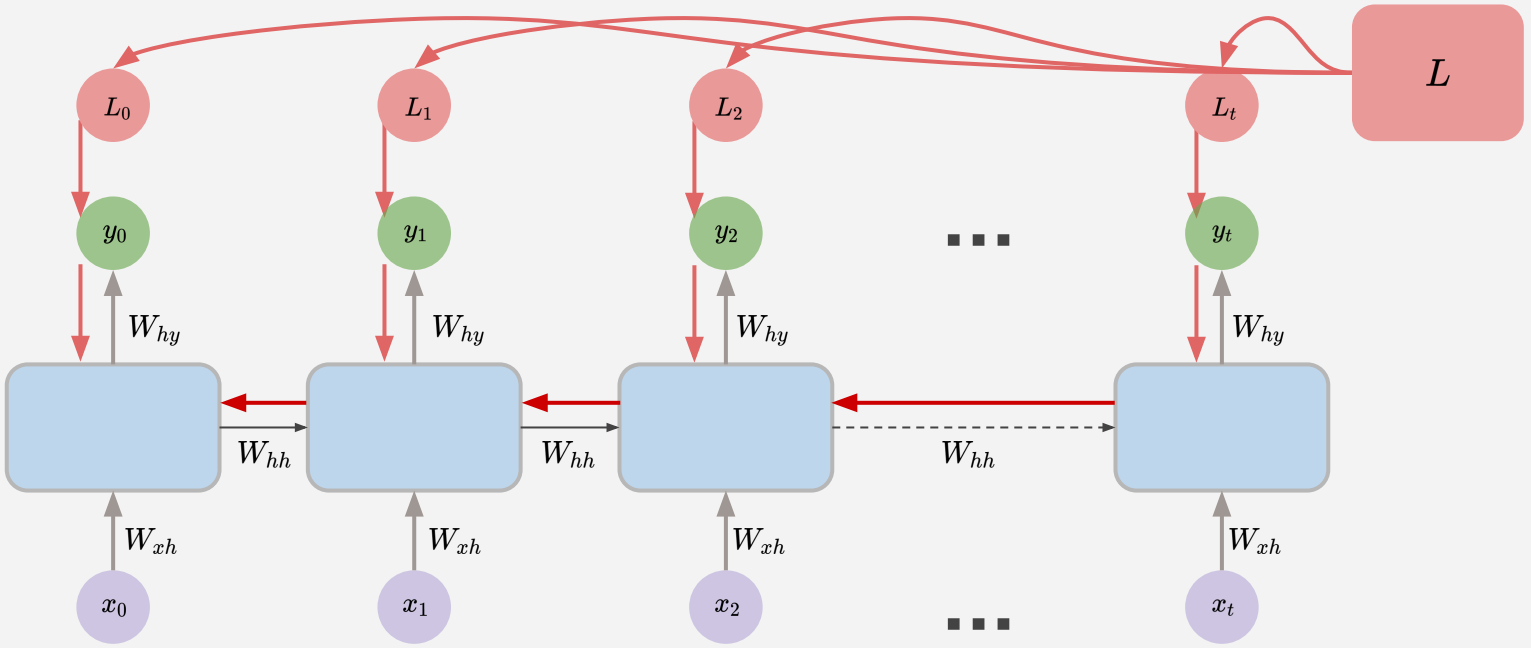
RNNs: Backpropagation Through Time (BPTT)



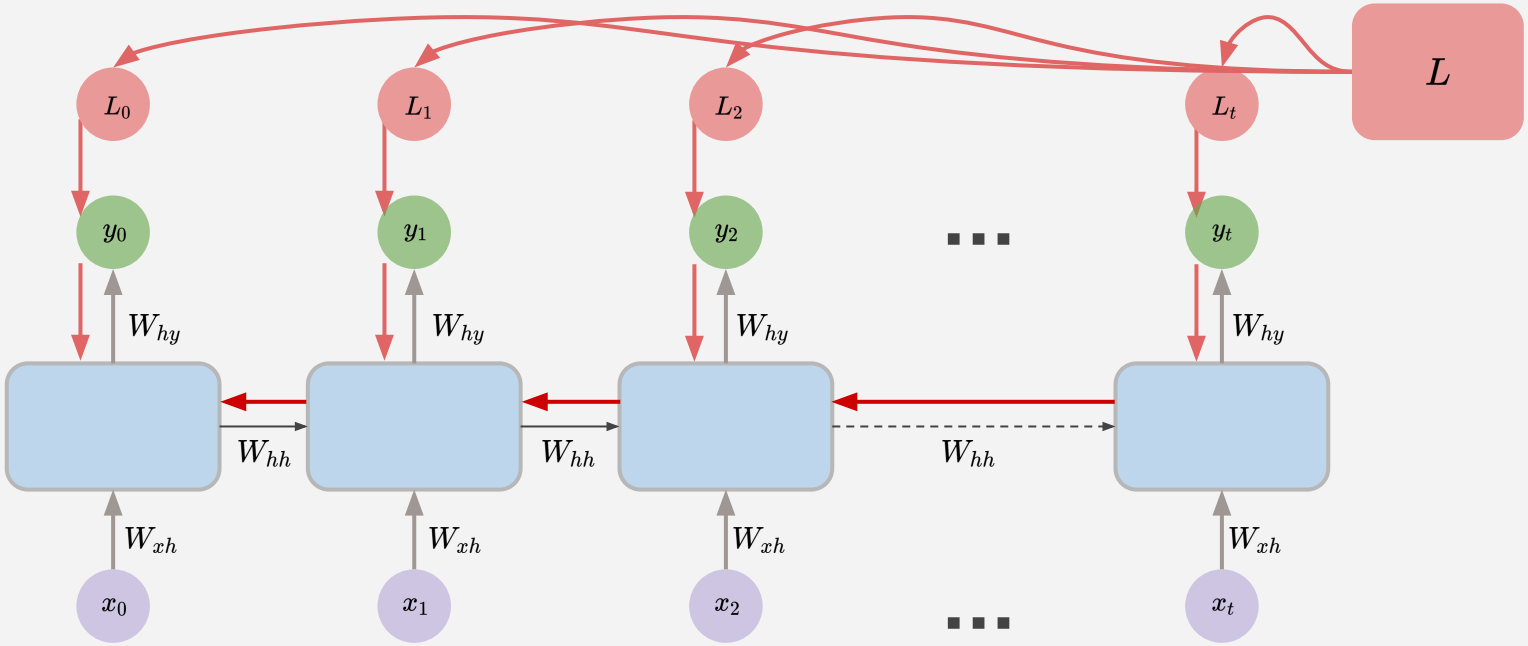
RNNs: Backpropagation Through Time (BPTT)



RNNs: Backpropagation Through Time (BPTT)

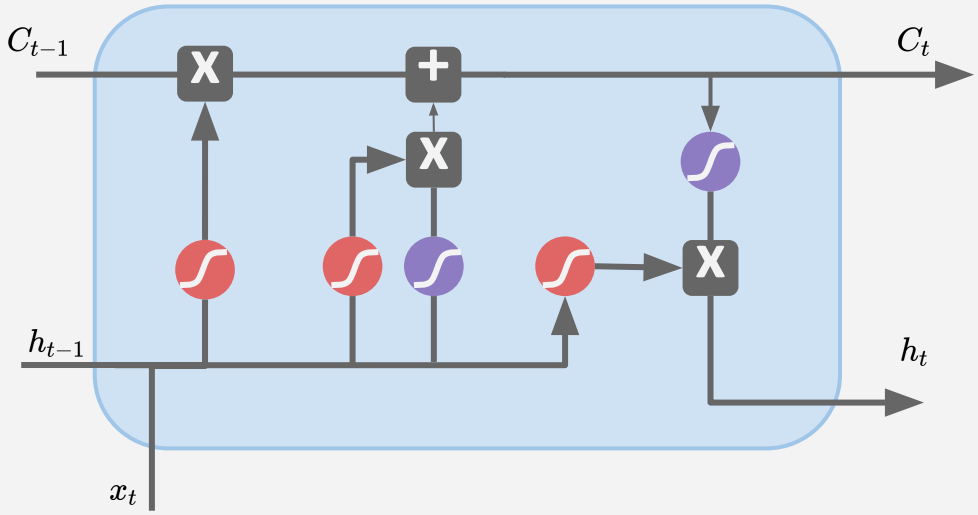


BPTT: Vanishing + Exploding Gradients



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

Review: LSTMs



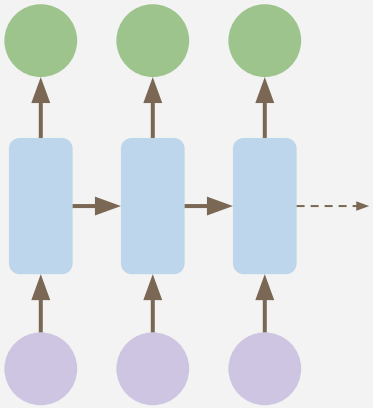
1. Maintains a persistent cell state - robust to gradient updates

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

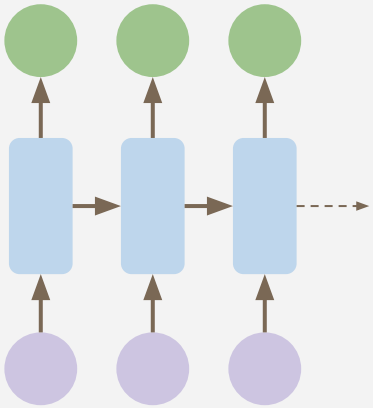
3. Gates are parameterized

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

Limitations of Recurrent Models



Limitations of Recurrent Models

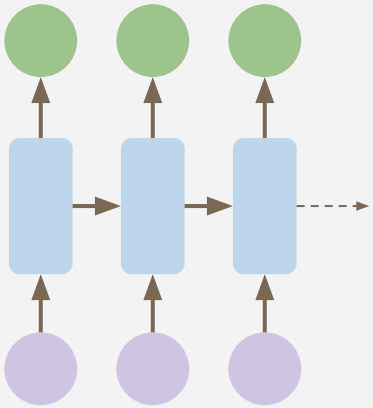


1. Encoding bottleneck

    Temporal separation required

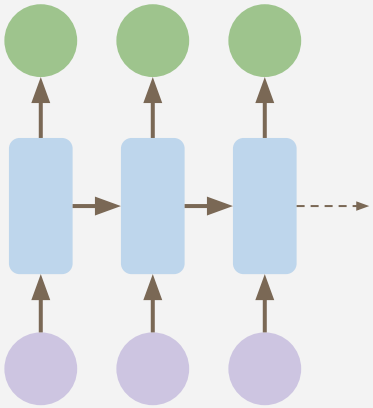
    Storing and maintaining large sequences

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

Limitations of Recurrent Models

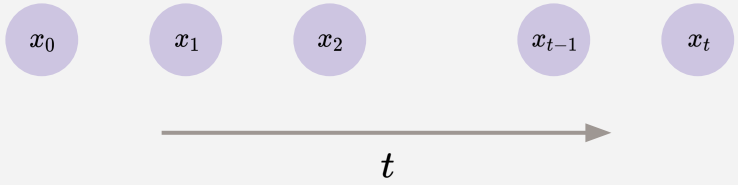


1. Encoding bottleneck
   - Temporal separation required
   - Storing and maintaining large sequences
2. Short Memory - Finite reference window
   - Difficulty in accessing information from many time steps ago
   - Vanishing gradients
3. Slow computation
   - Cannot be parallelized

Goals of Sequence Modeling

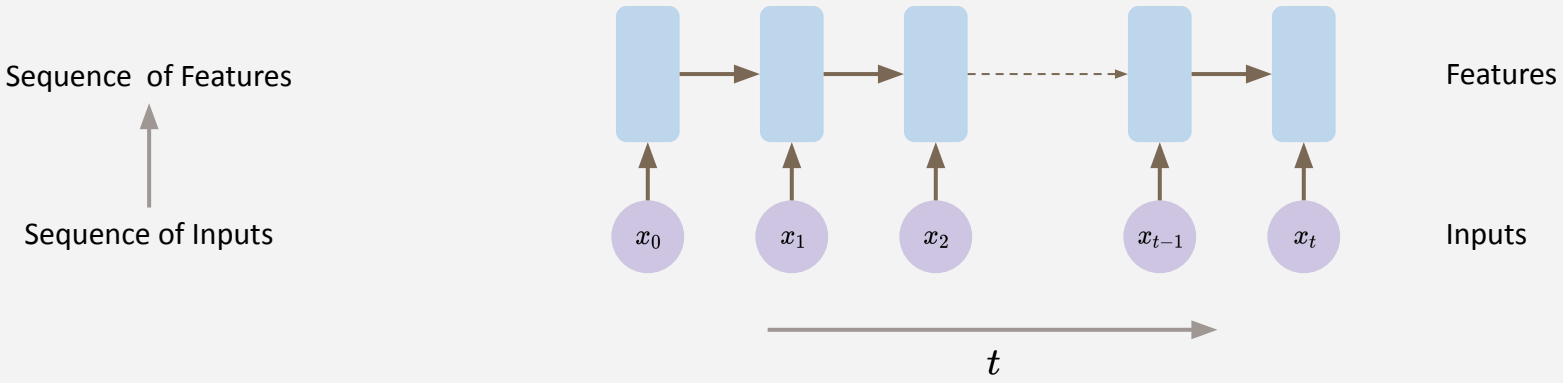
Goals of Sequence Modeling

Sequence of Inputs

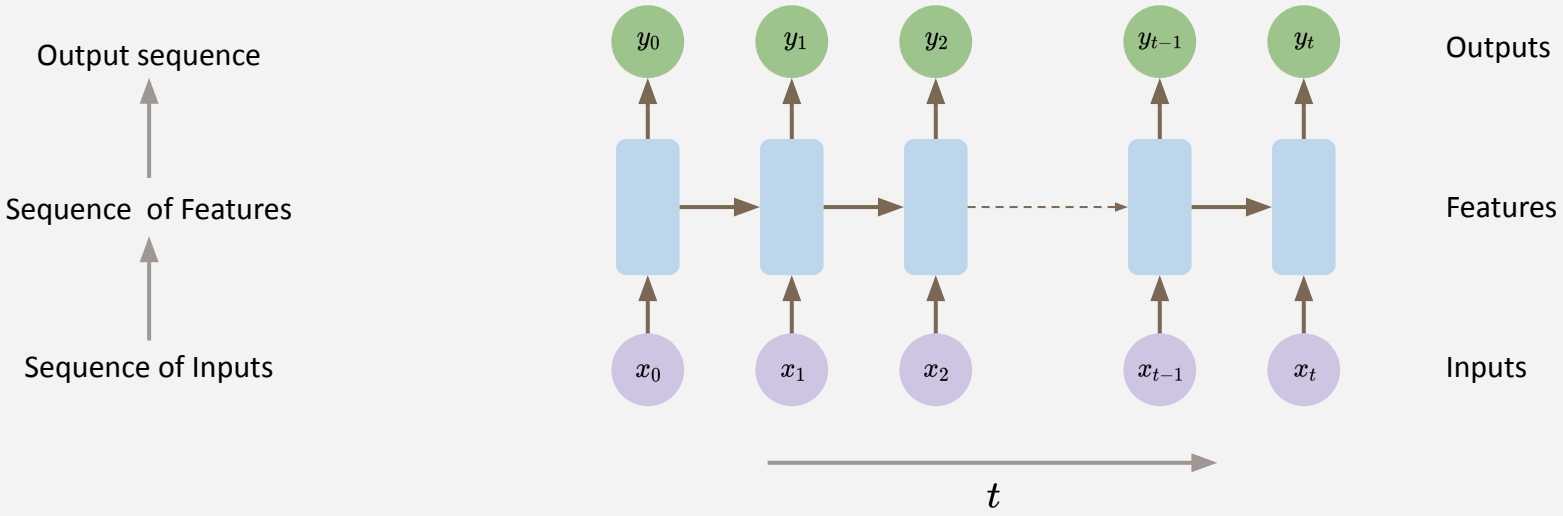


Inputs

Goals of Sequence Modeling

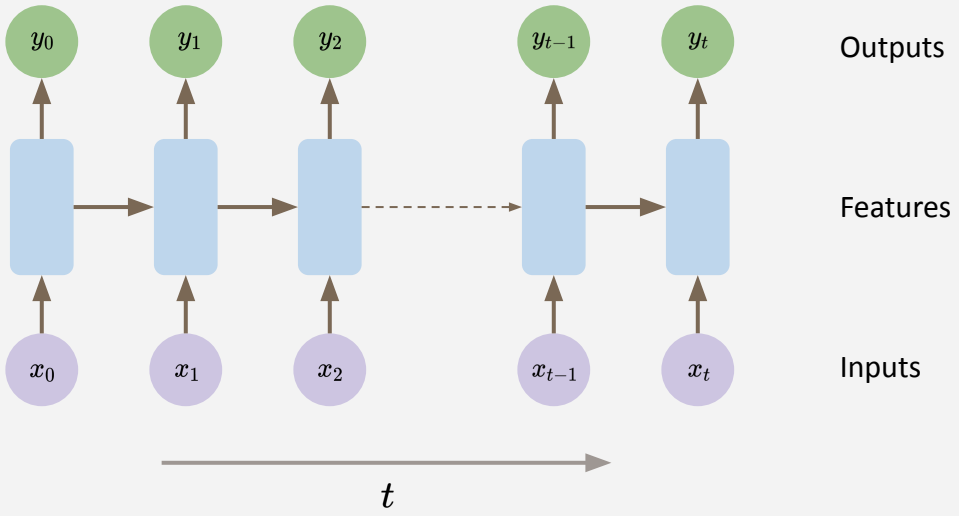


Goals of Sequence Modeling



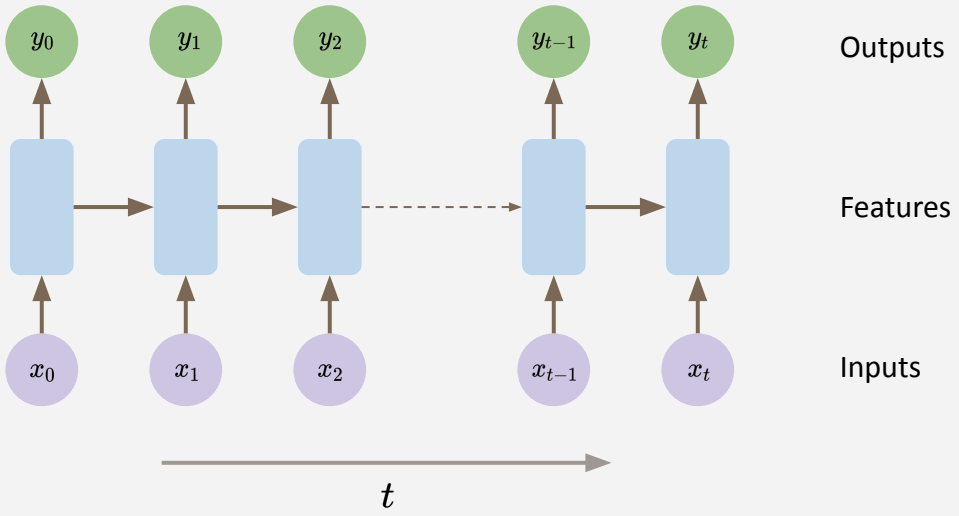
Goals of Sequence Modeling

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



Goals of Sequence Modeling

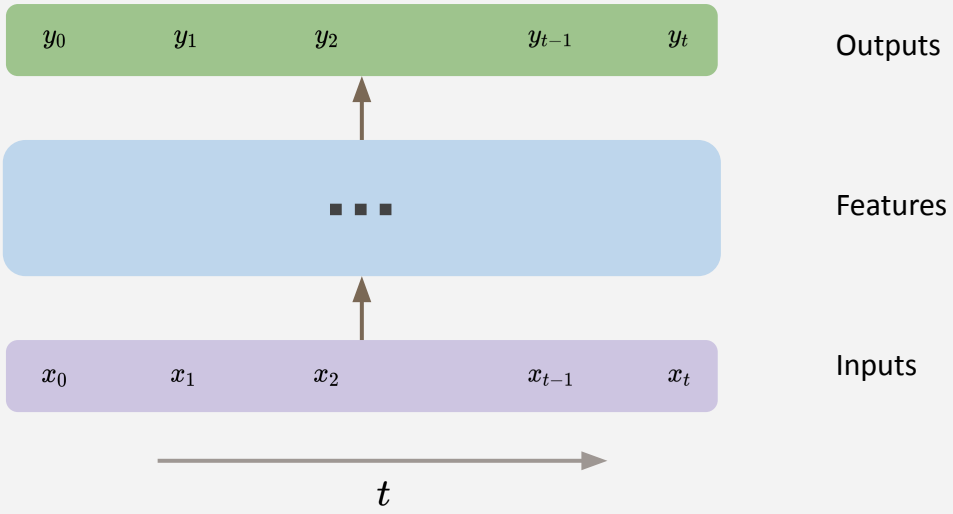
1. Multimodal robust encoding
2. Longer Memory
   Infinite reference window
3. Parallelization
   Efficient compute



What if we eliminate recurrence and accumulate sequences?

Goals of Sequence Modeling

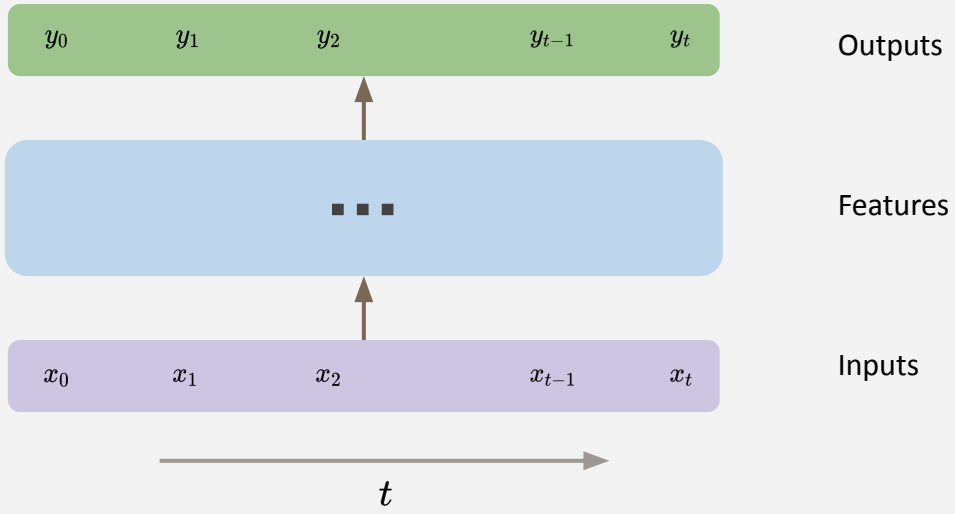
Idea: Feedforward Network



Goals of Sequence Modeling

Idea: Feedforward Network

1. **No Recurrence**
   - \+ parallelization

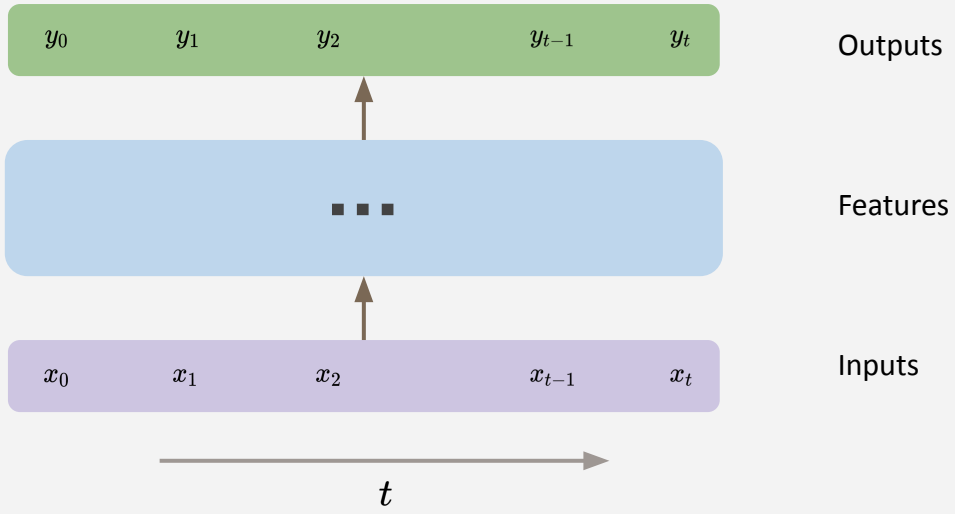


Goals of Sequence Modeling

Idea: Feedforward Network

1. **No Recurrence**
   + parallelization

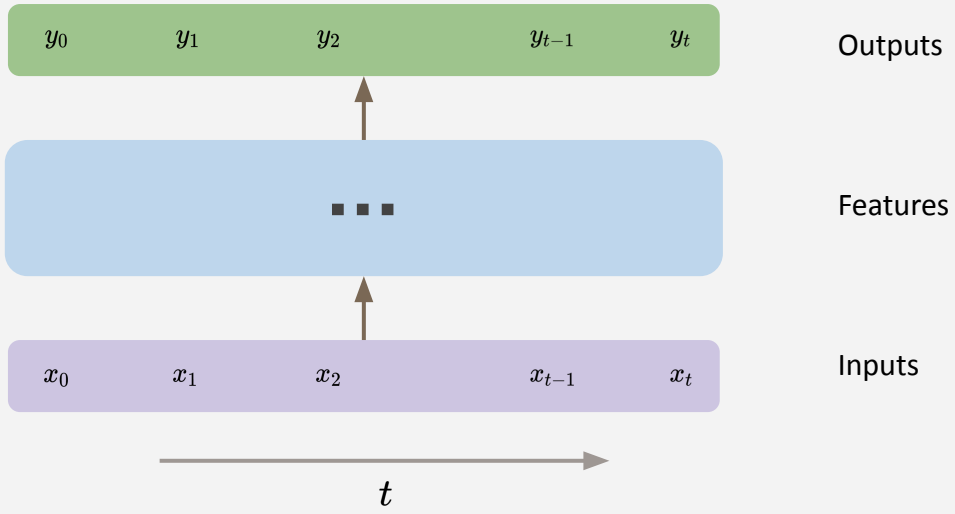
2. **No Memory**



Goals of Sequence Modeling

Idea: Feedforward Network

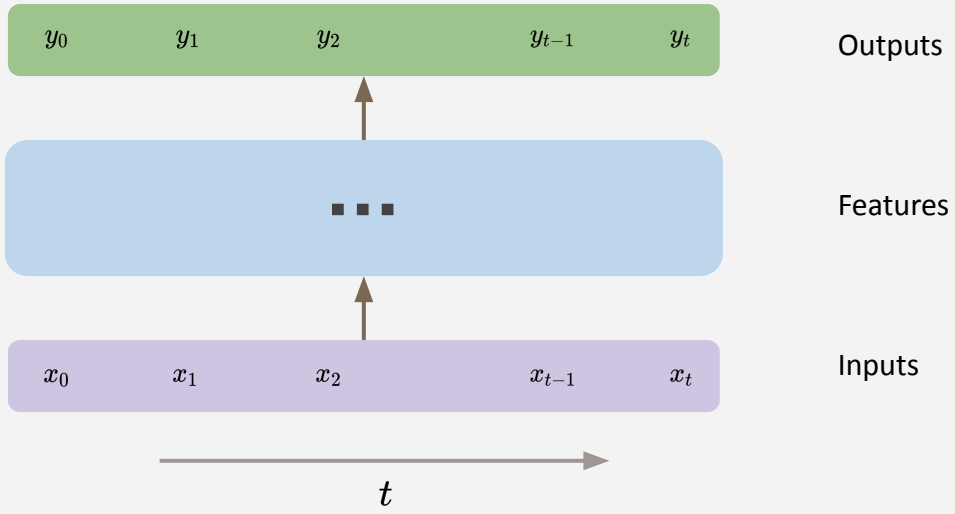
1. **No Recurrence**
   + parallelization
2. **No Memory**
3. **Not Scalable**



Goals of Sequence Modeling

Idea: Feedforward Network

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



Insight: Residual + Attend to What is Important

Attention Is All You Need

Oregon State University

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

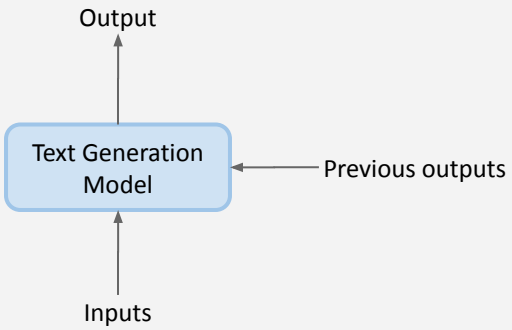
Attend to the most important parts of an input

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

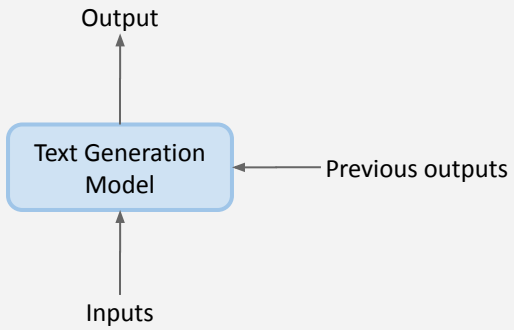
2. Extract parts with high attention } **Feature Extraction**

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

Attention: Text Generation



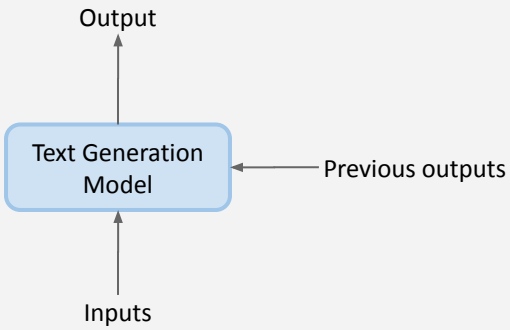
Attention: Text Generation



The linden tree

Attention: Text Generation

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



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.

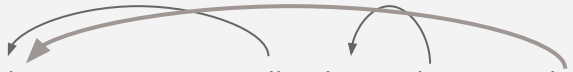
Attention: Text Generation

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

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.

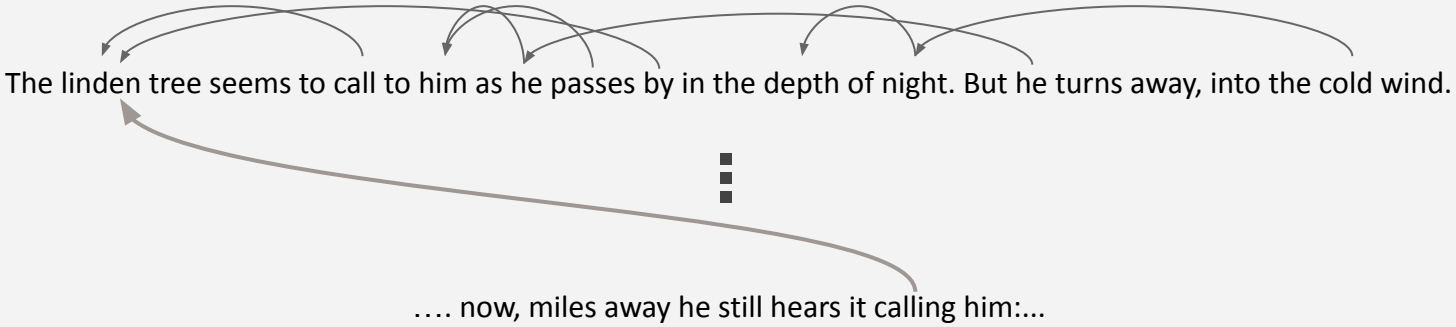
Attention: Text Generation



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

Attention: Text Generation



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.

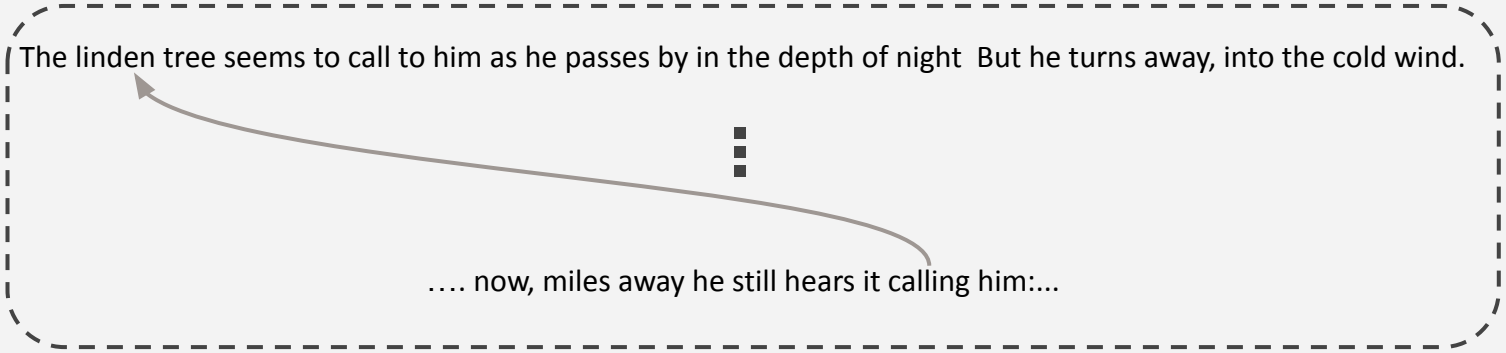
Recurrent Neural Networks have finite (and small) persistence

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.

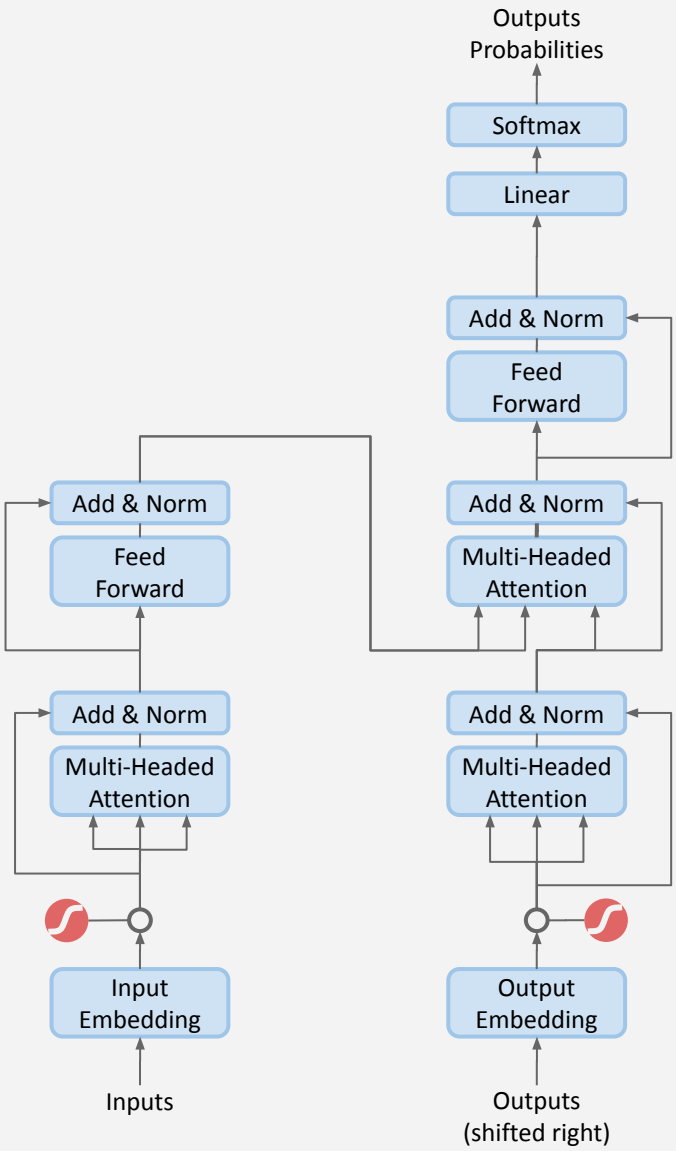
LSTMs have longer persistence

Attention: Text Generation



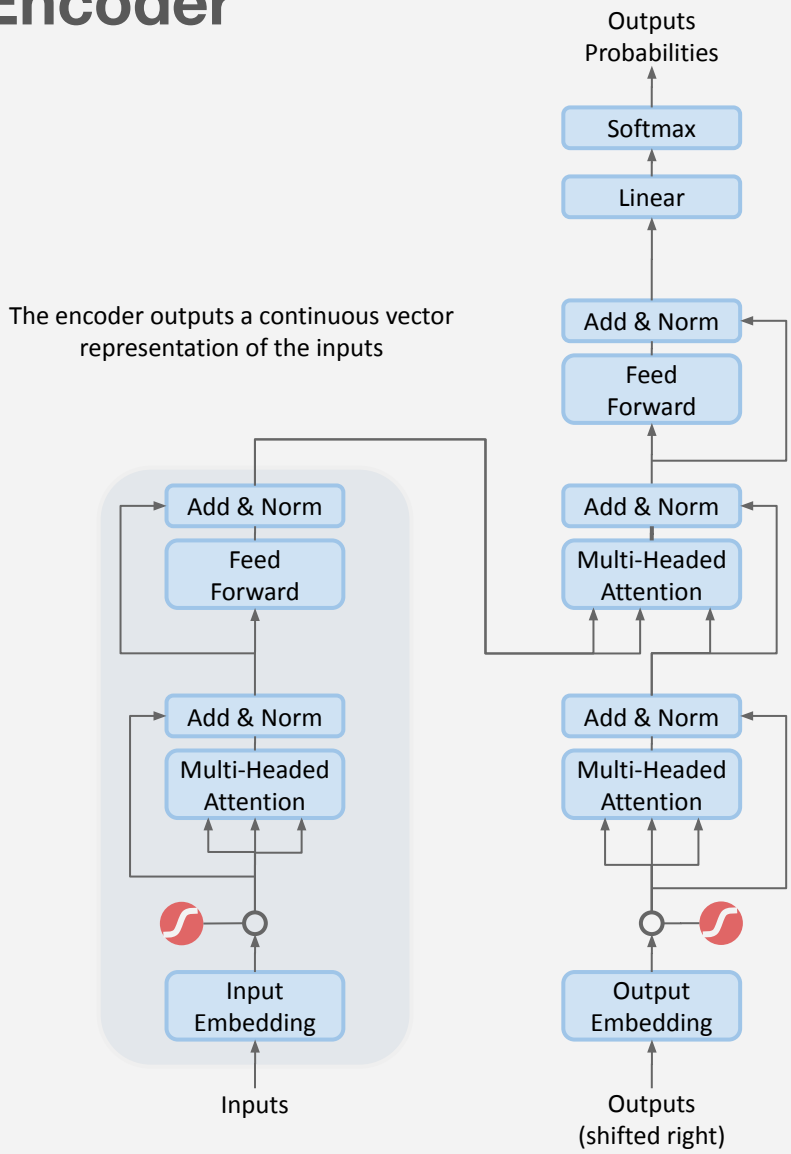
Attention mechanisms can provide infinite persistence (in theory)

Transformer



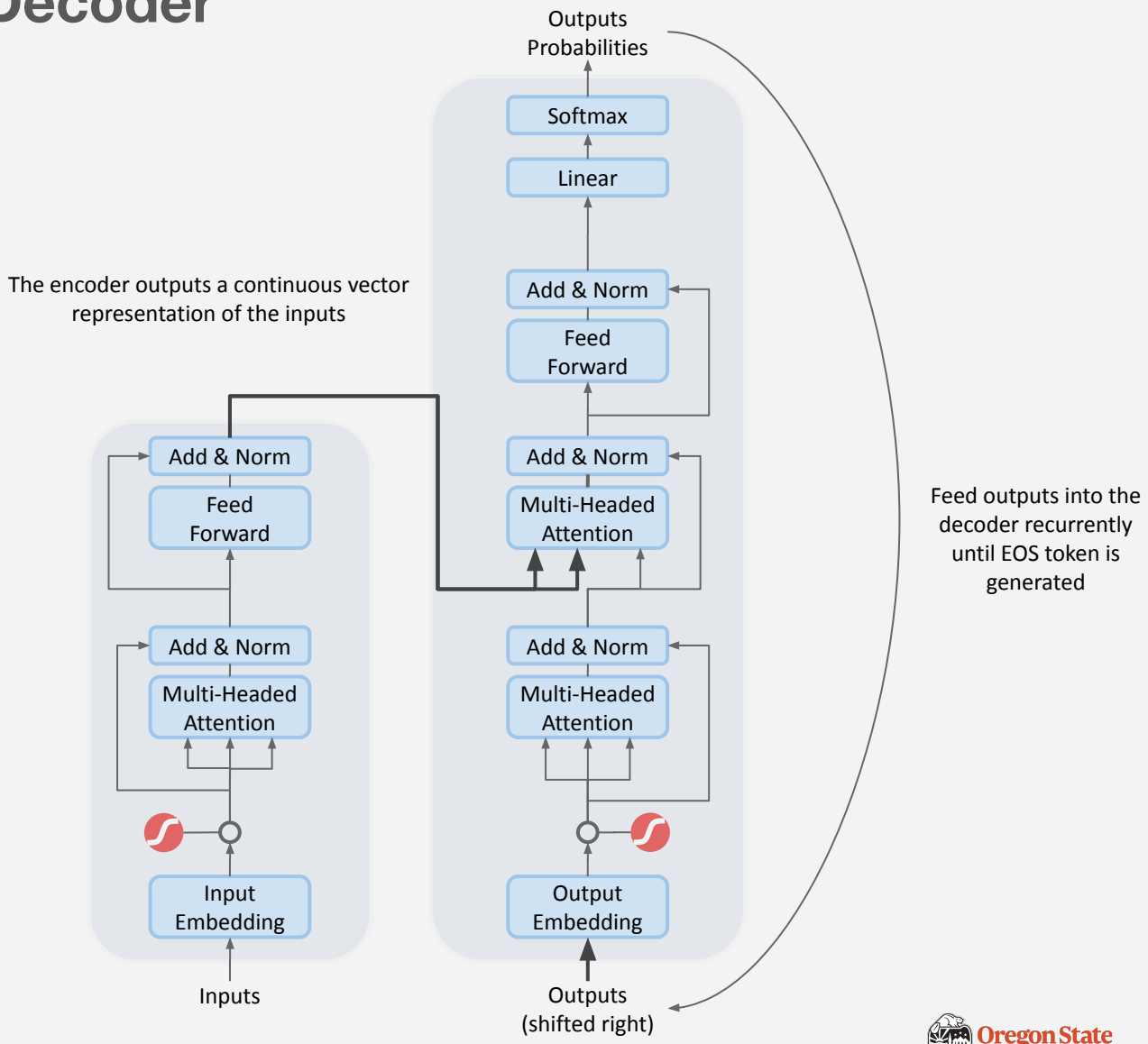
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer: Encoder



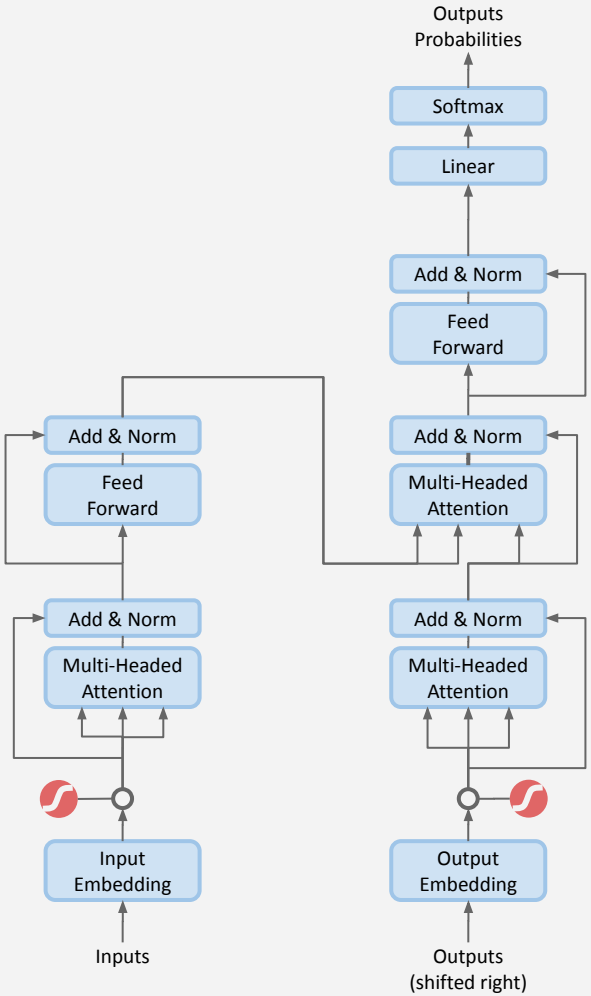
Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer: Decoder

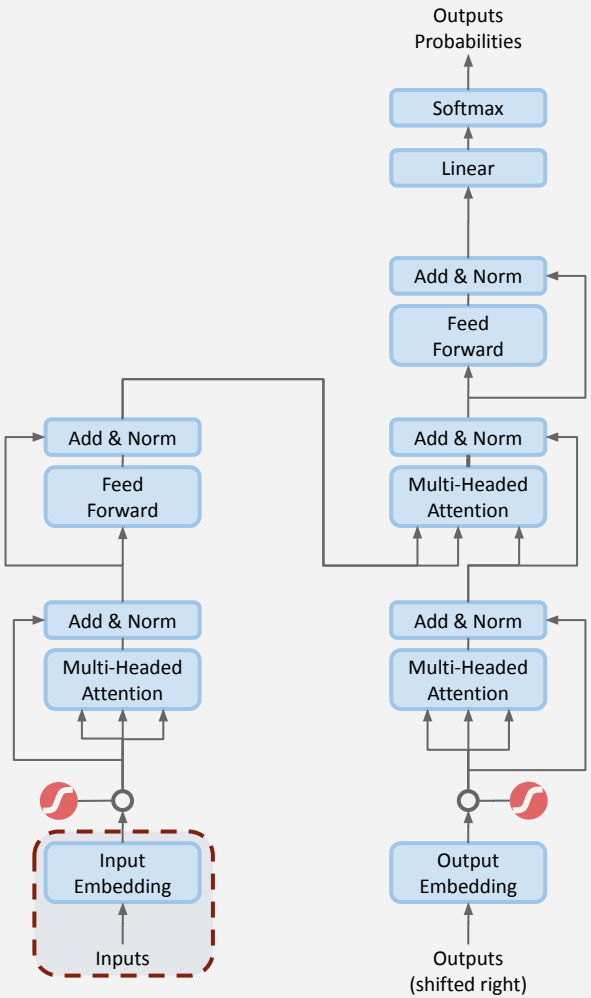


Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Transformer

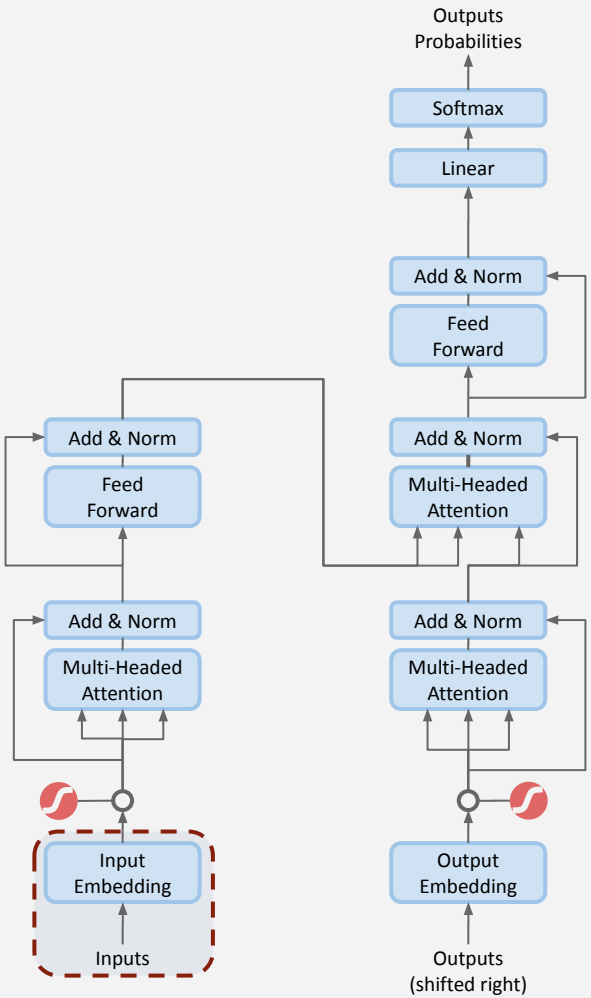


Input Embedding



Input Embedding

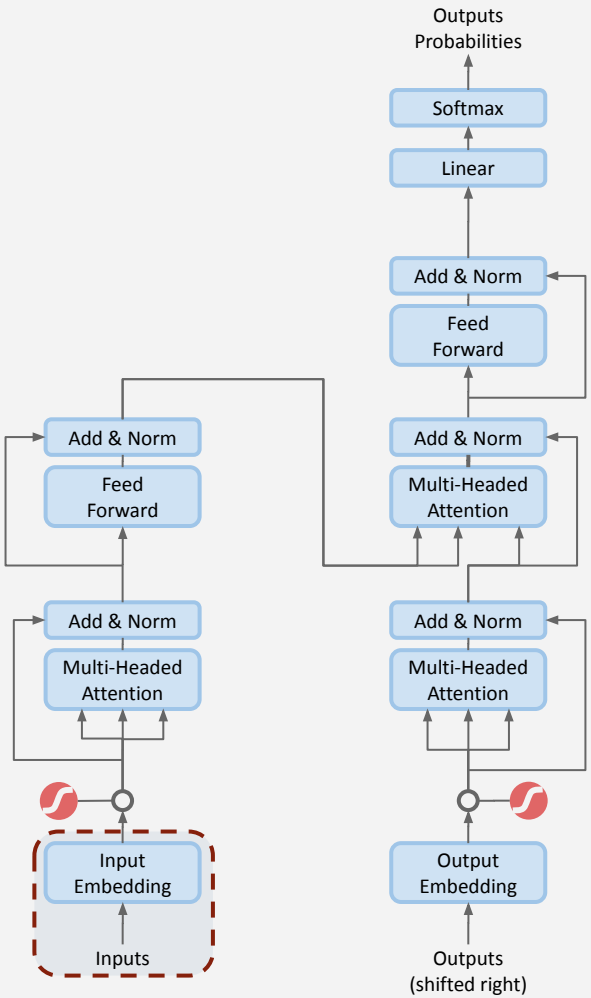
Input Embedding

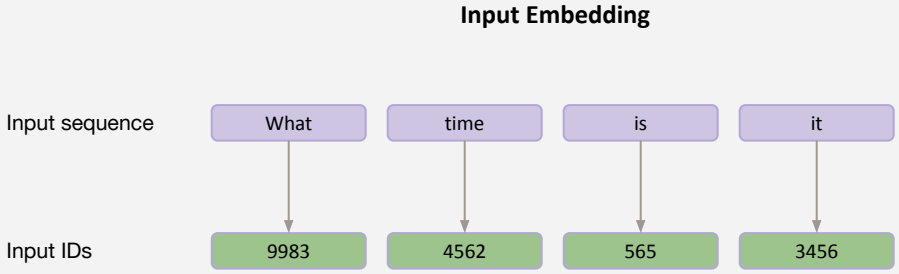


Input Embedding

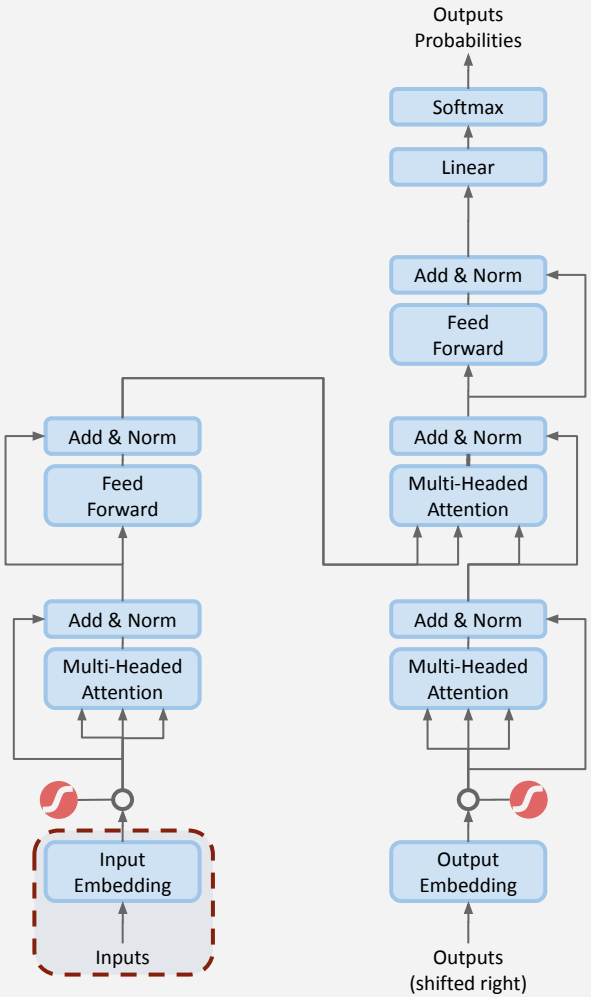
Input sequence | What | time | is | it

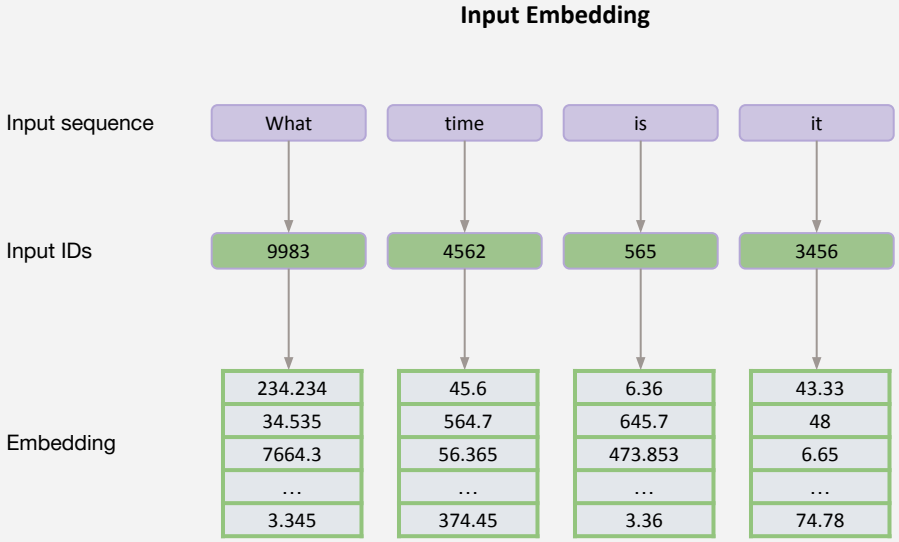
Input Embedding





Input Embedding





Positional Encoding

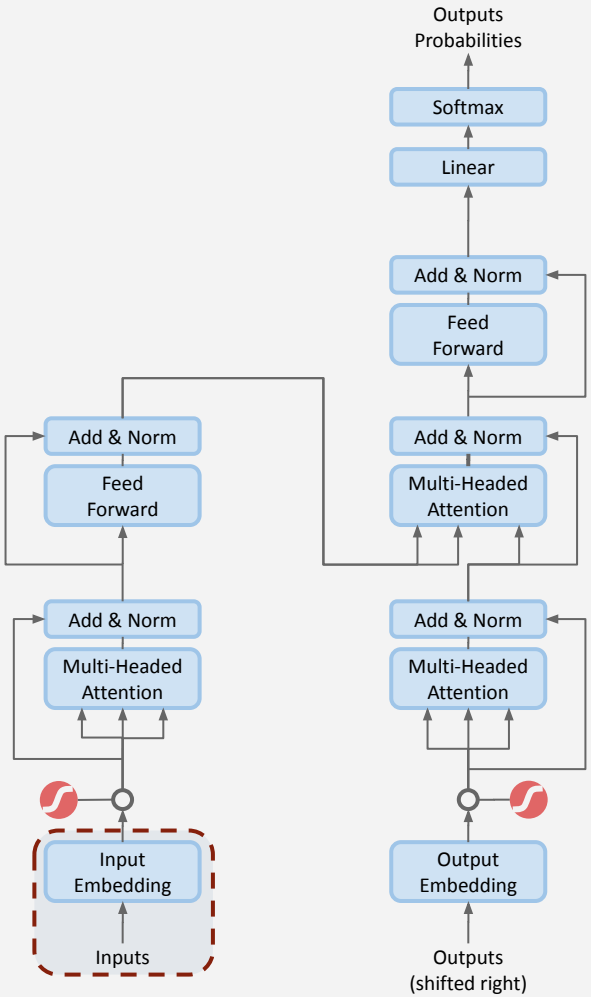
1. Encoding should retain information about positional dependencies

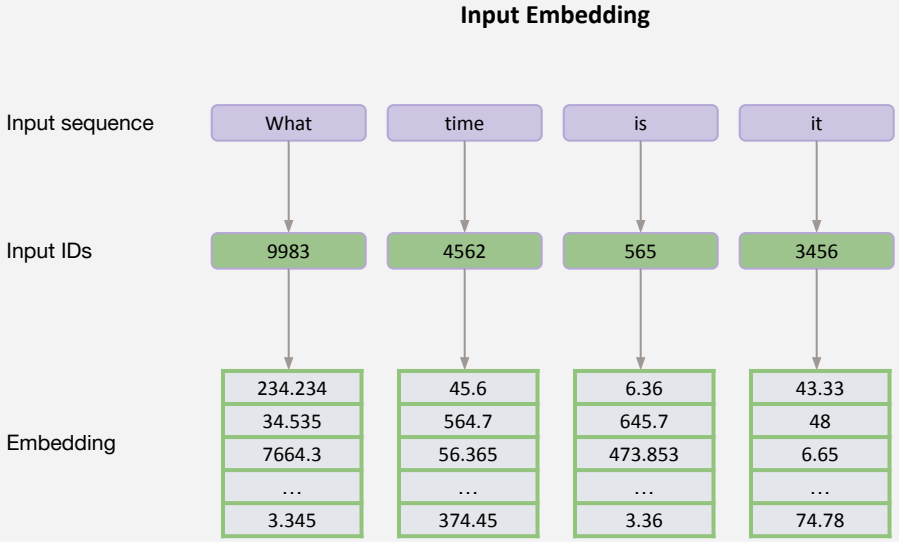
2. **Words must enforce semantic (contextual) relationships**

   Relative positions in sentences

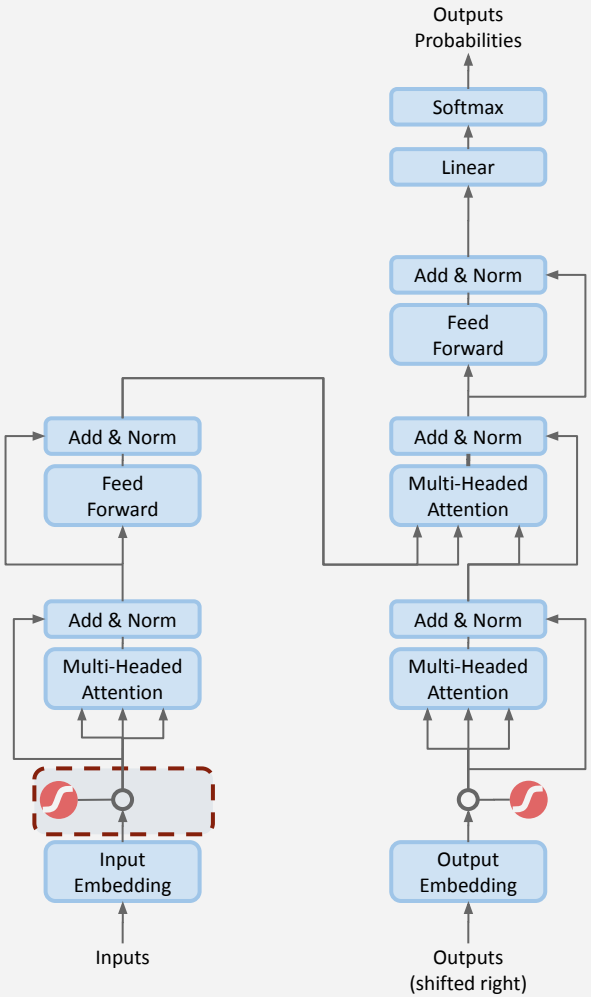
3. We want our model to learn this positional information

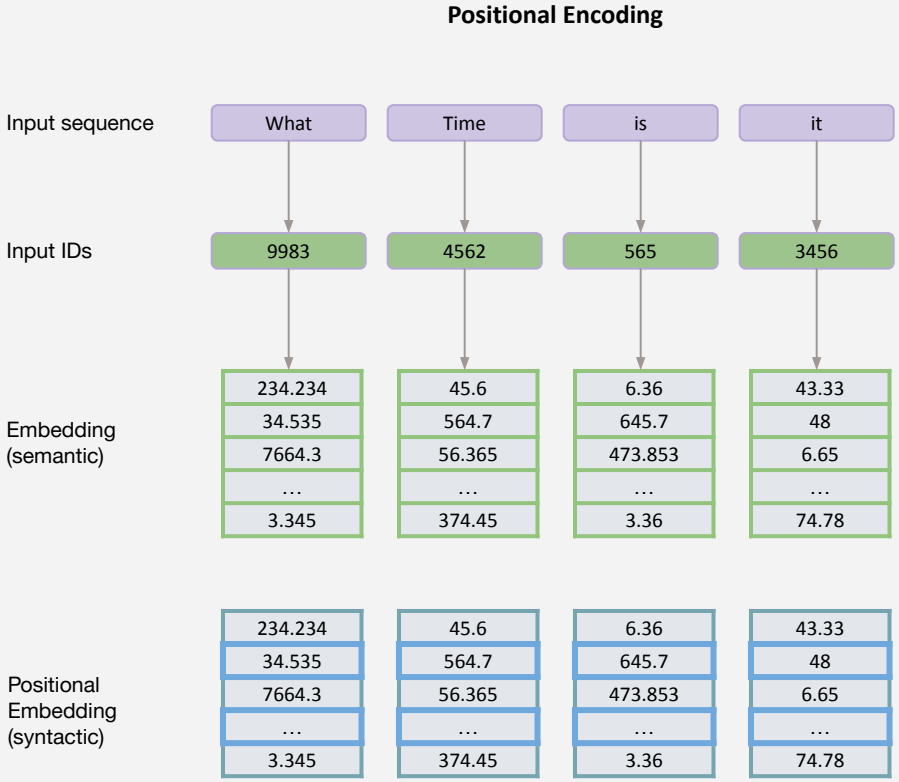
Input Embedding



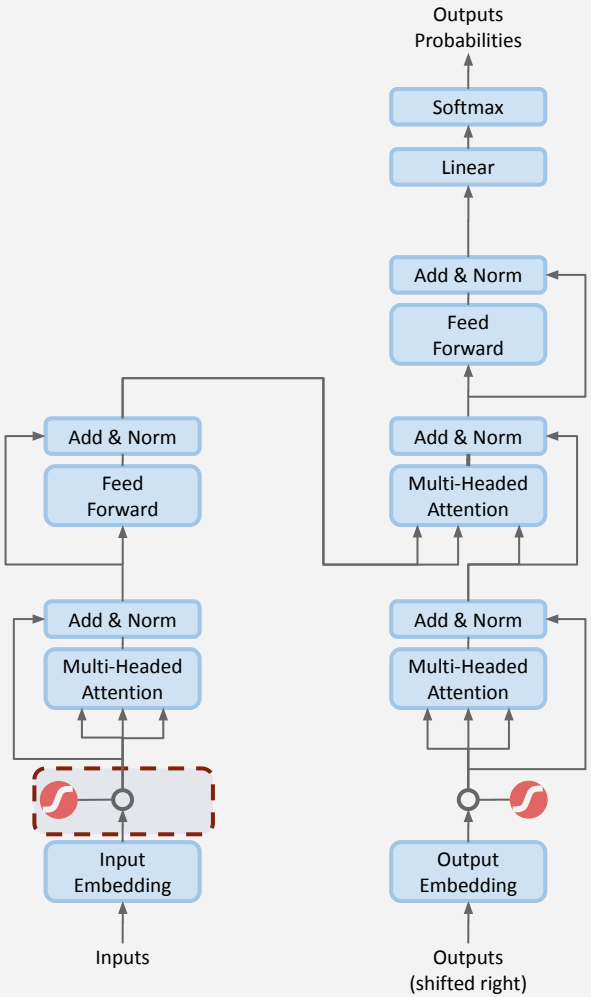


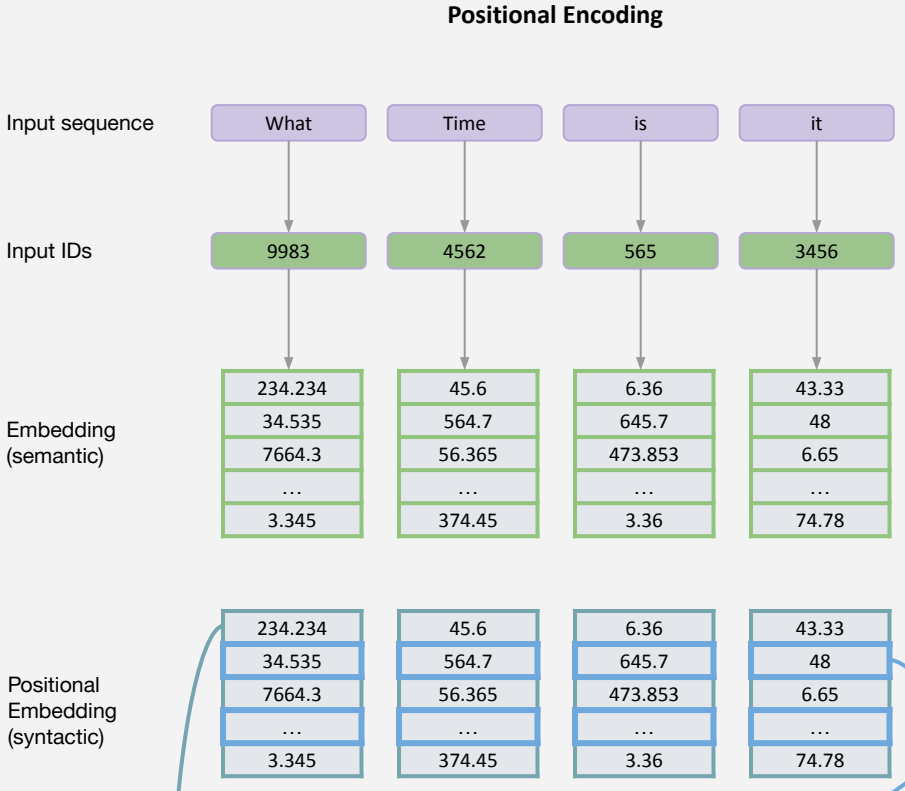
Positional Encoding





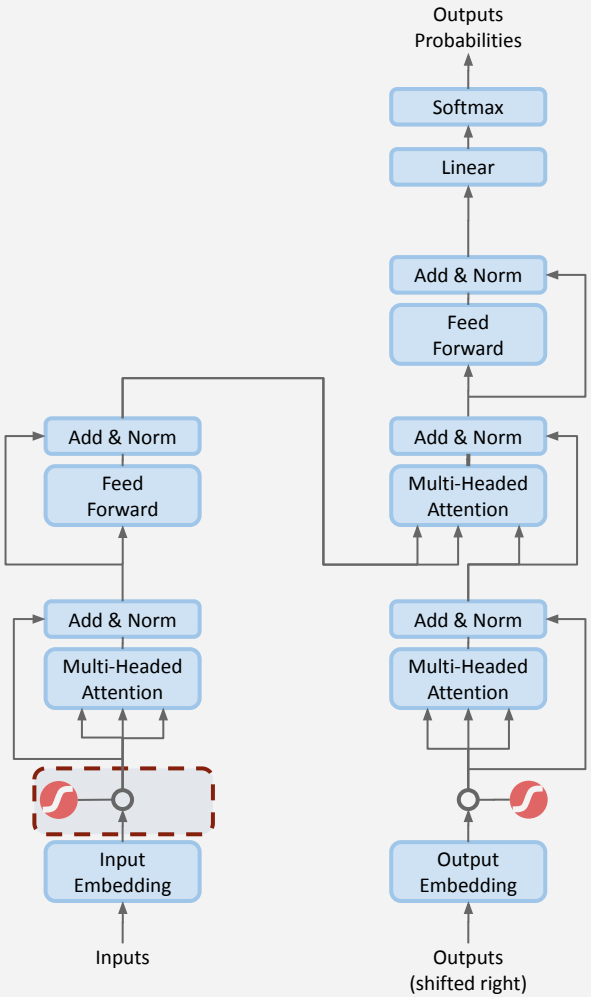
Positional Encoding

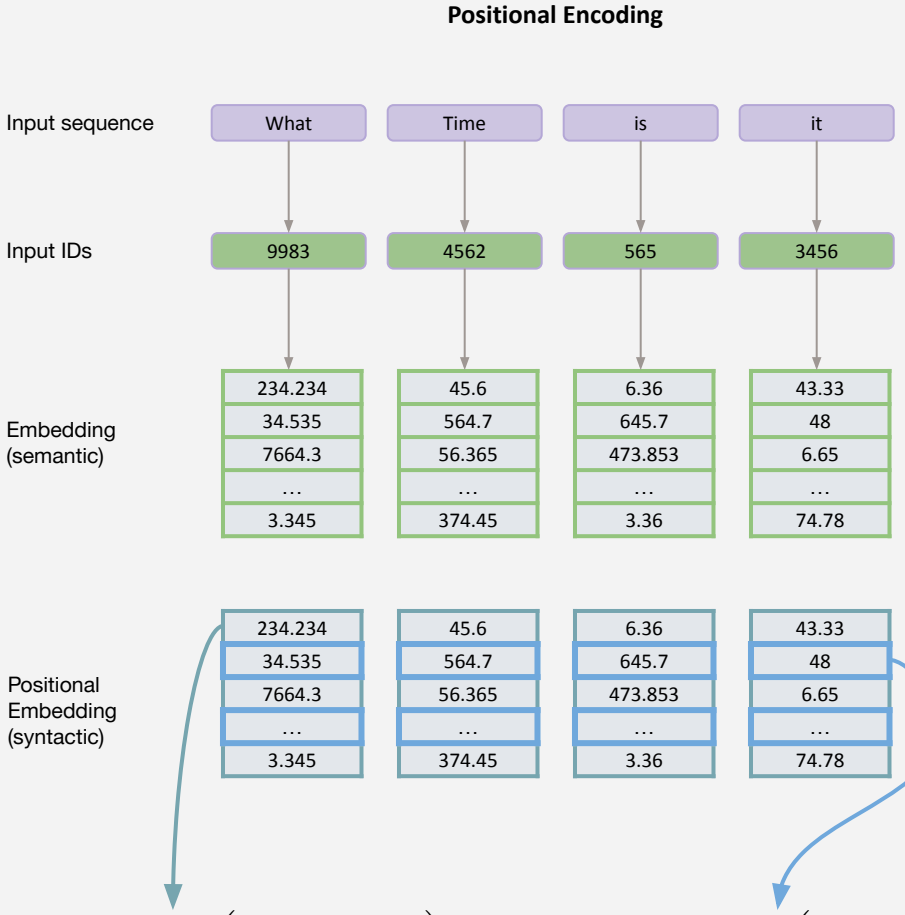




$$PE(pos,\ 2i+1) = \cos\left(\frac{pos}{10000^{2i/dmodel}}\right) \qquad PE(pos,\ 2i) = sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

Positional Encoding



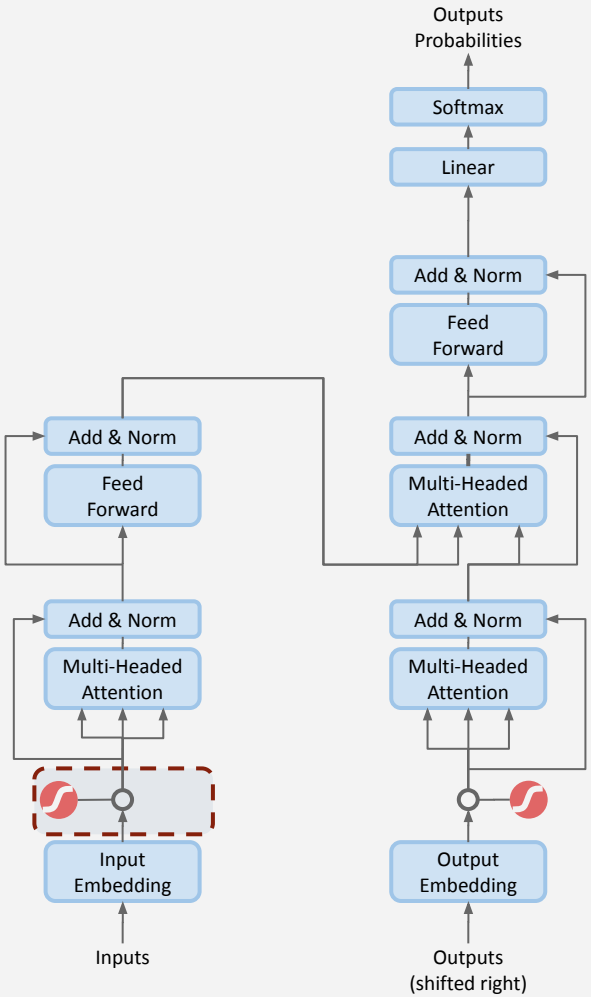


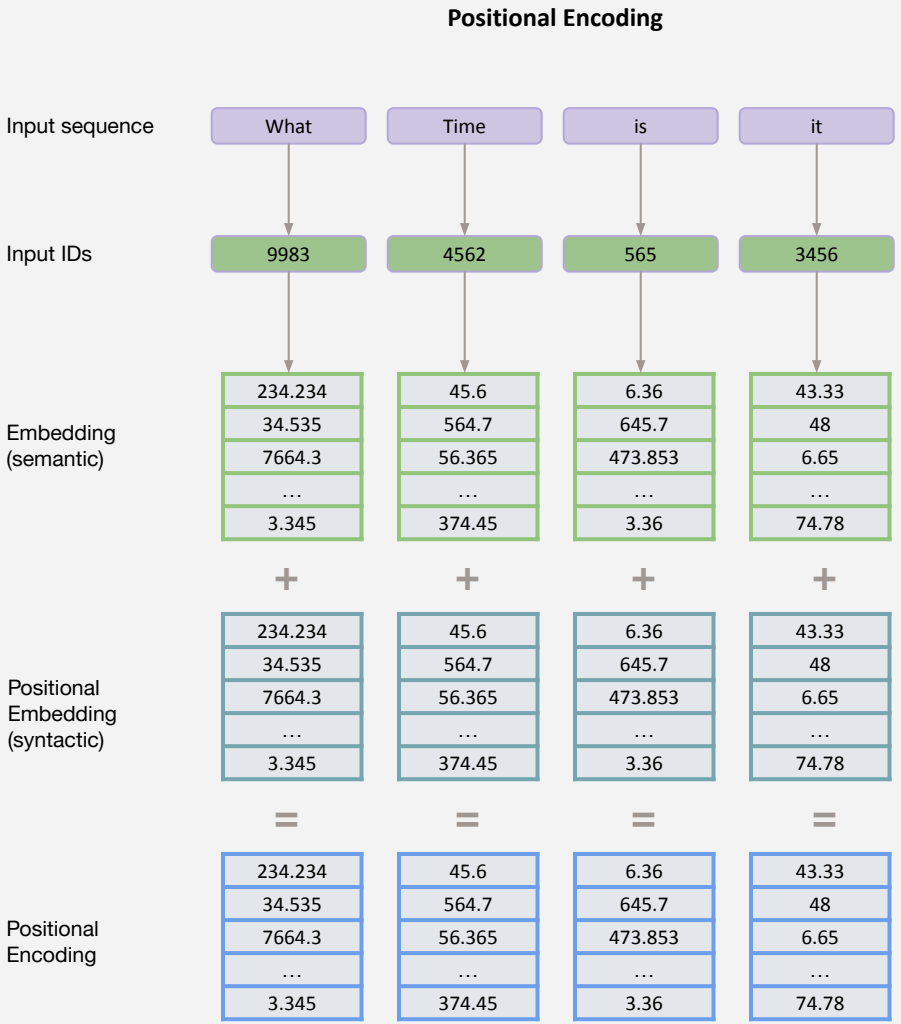
$$PE(pos,\ 2i+1) = \cos\left(\frac{pos}{10000^{2i/dmodel}}\right) \qquad PE(pos,\ 2i) = sin\left(\frac{pos}{10000^{2i/dmodel}}\right)$$

Only computed once!

Oregon State University

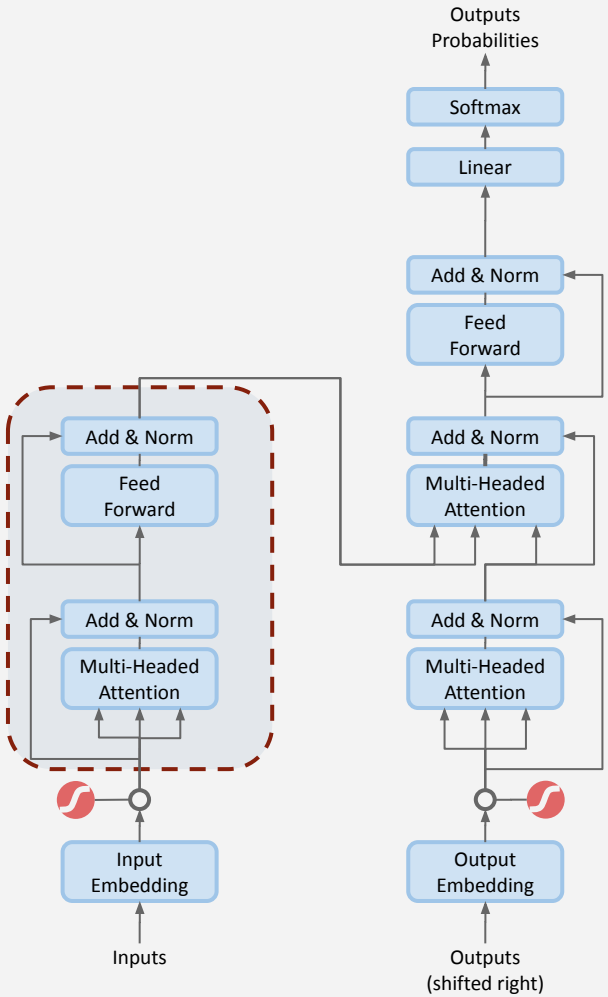
Positional Encoding



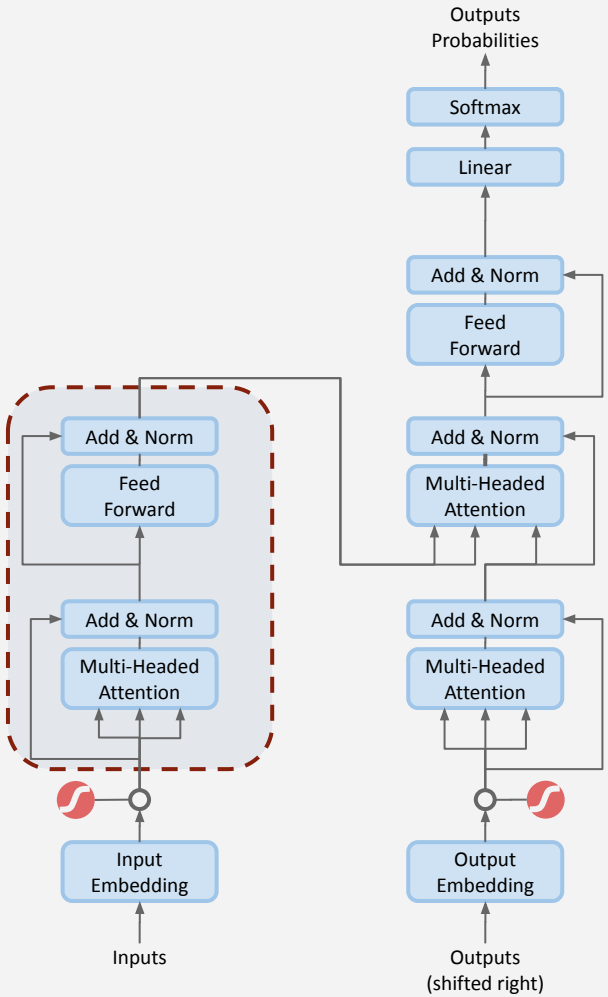


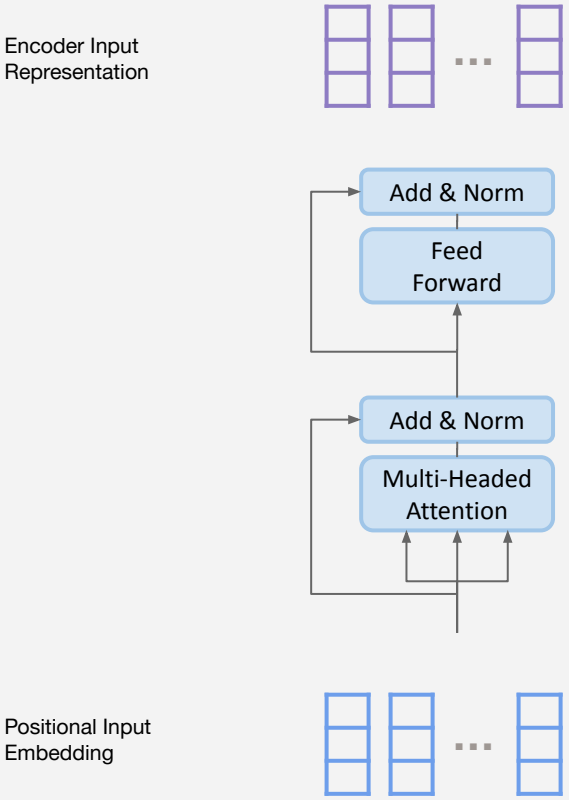
Oregon State University

Encoder Layer

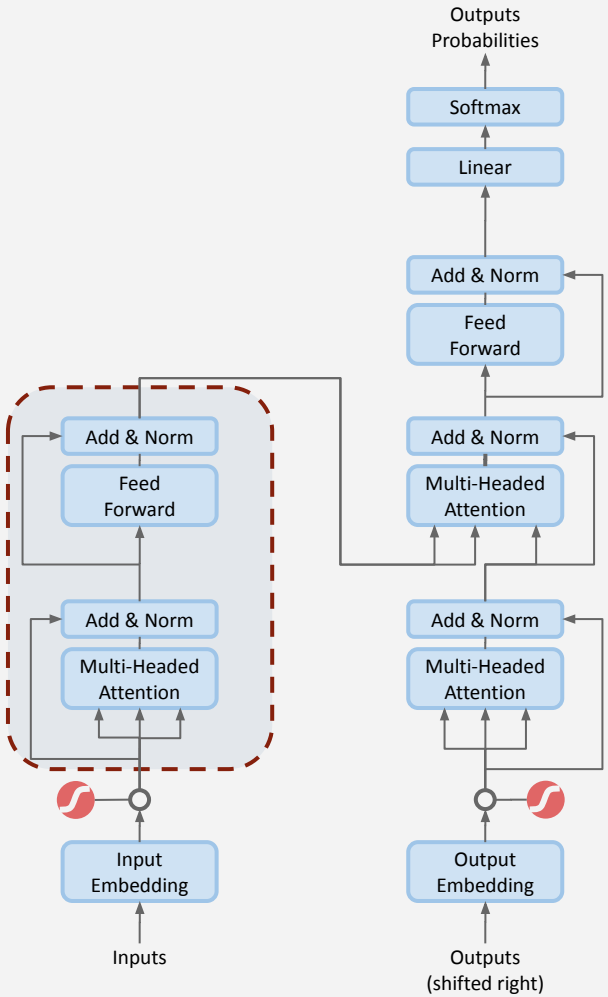


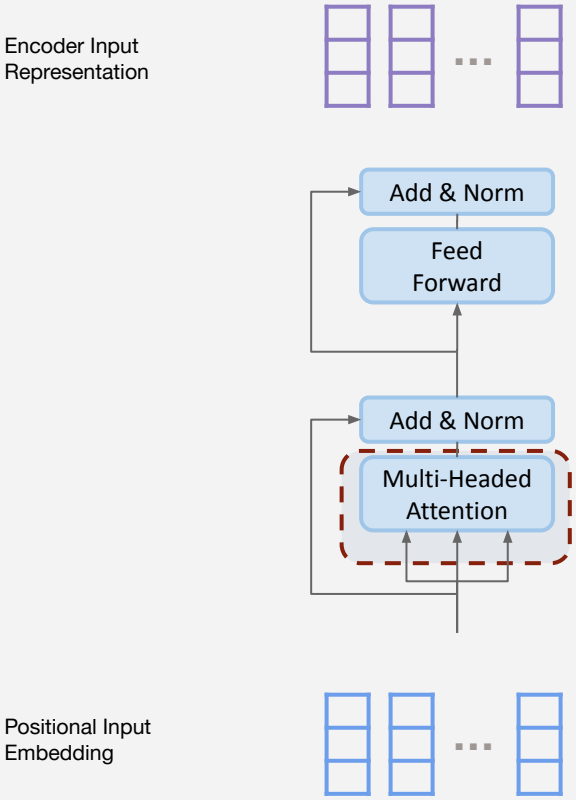
Encoder Layer



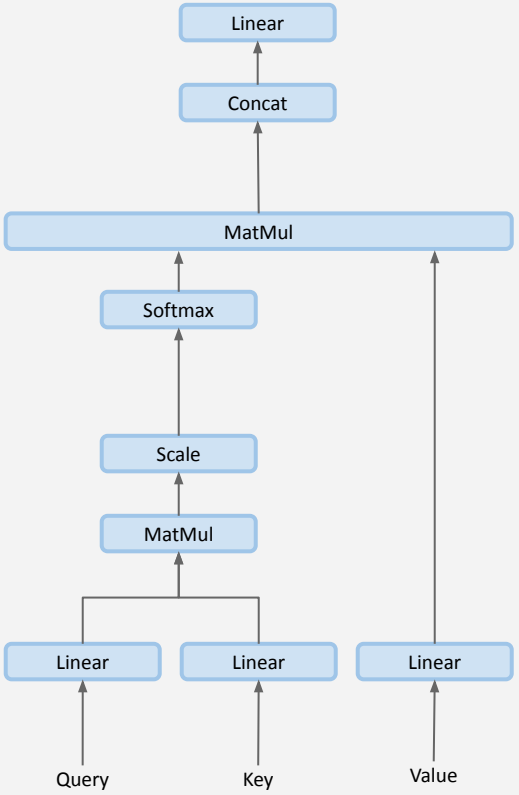


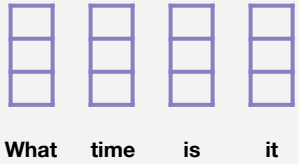
Encoder Layer



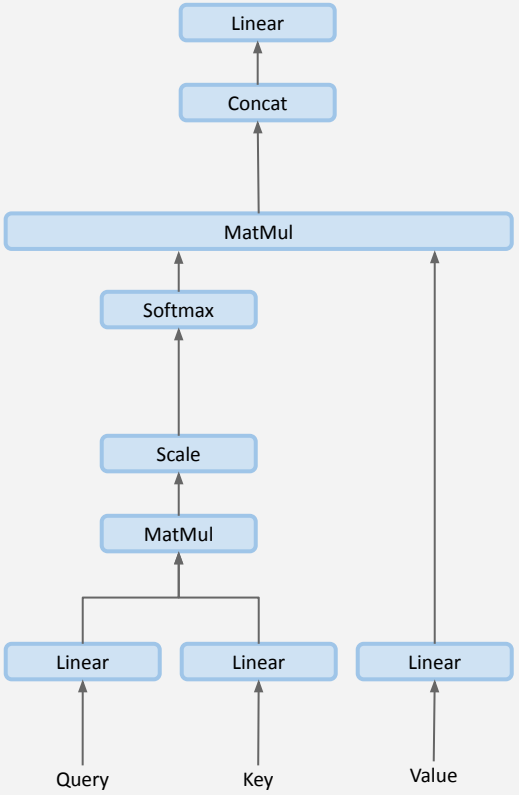


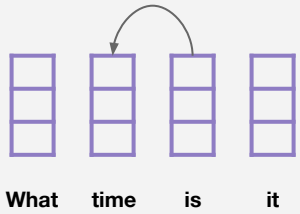
Self-Attention



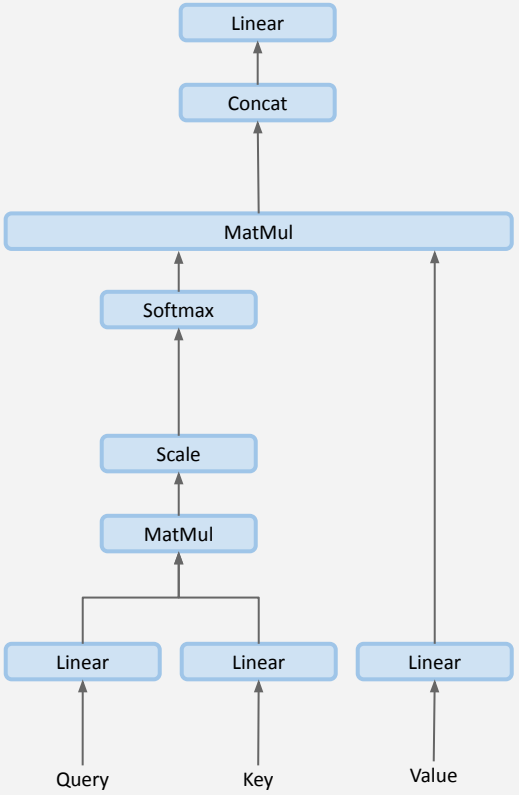


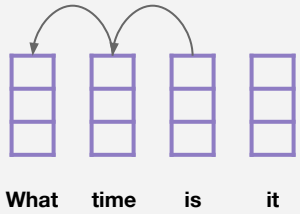
Self-Attention



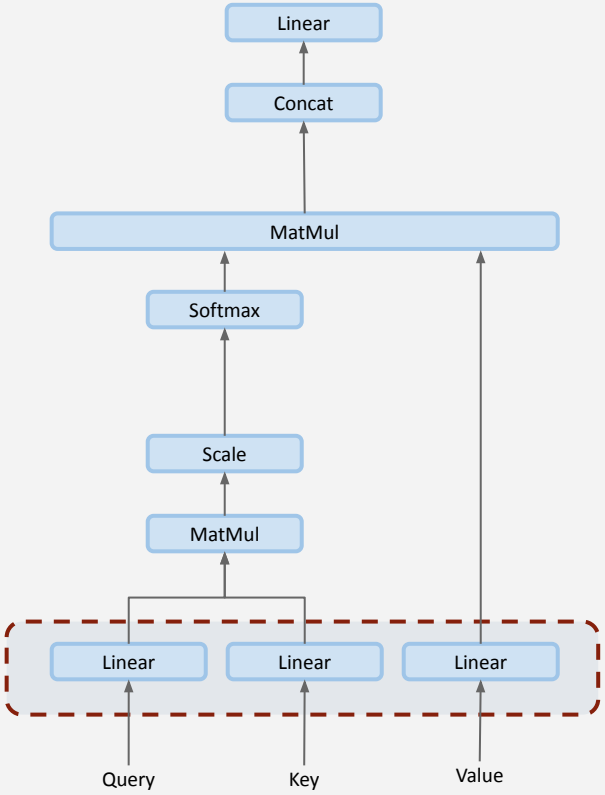


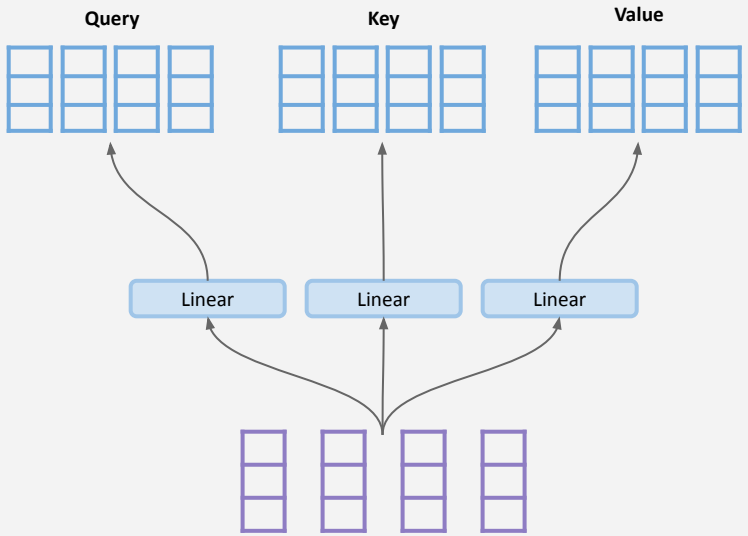
Self-Attention



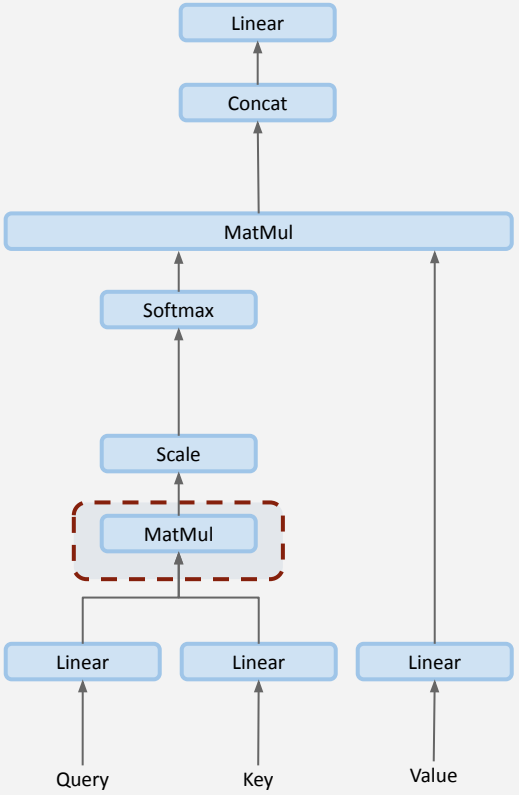


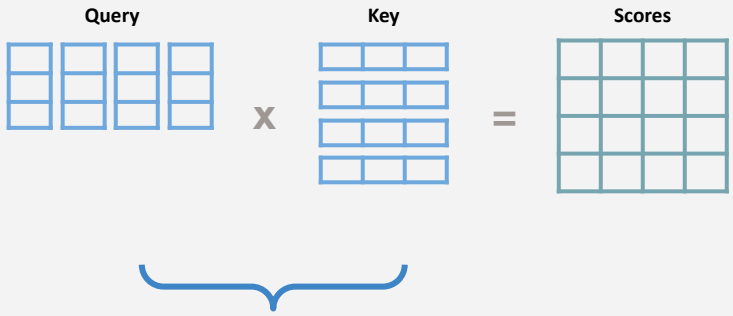
Self-Attention



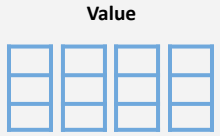


Self-Attention

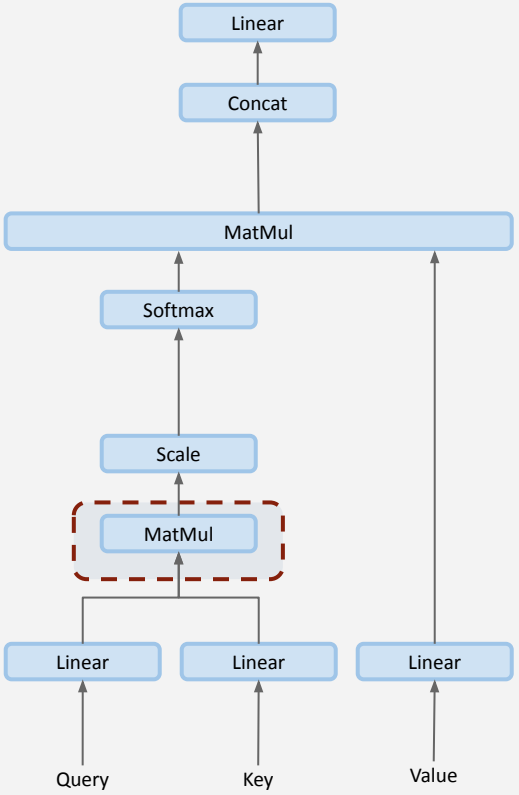


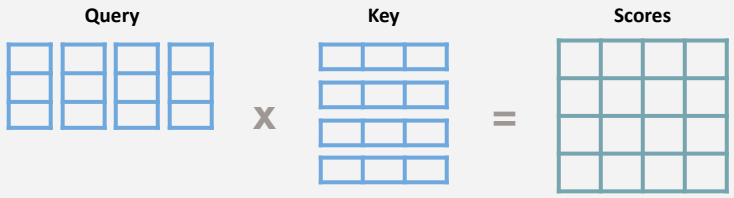


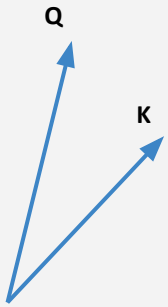
compute pairwise similarity between each **query** and **key**



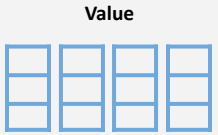
Self-Attention





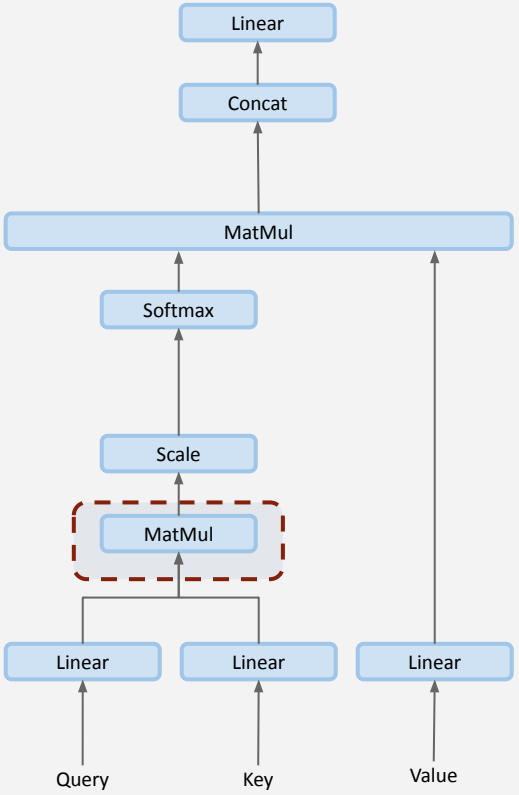


compute pairwise similarity between each **query** and **key**



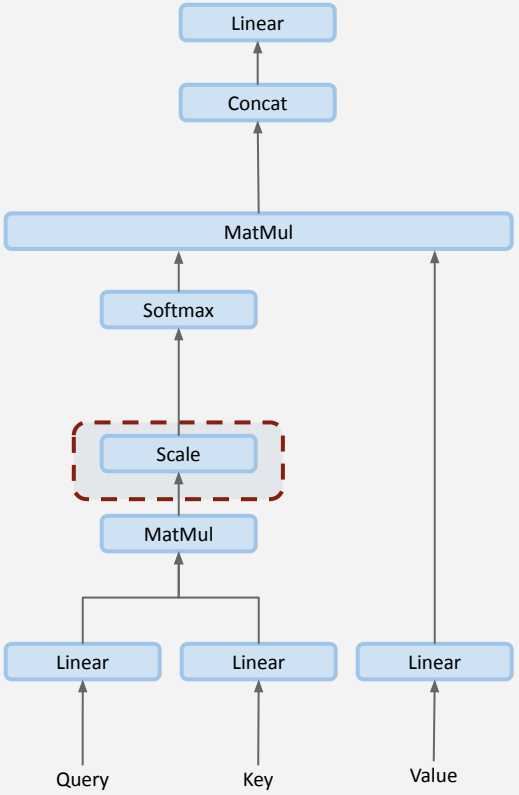
Oregon State University

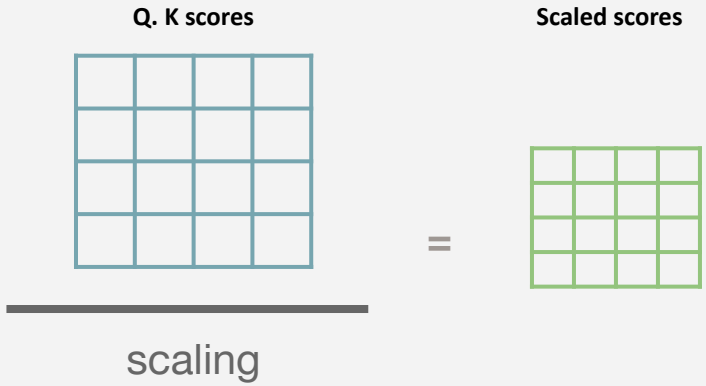
Self-Attention



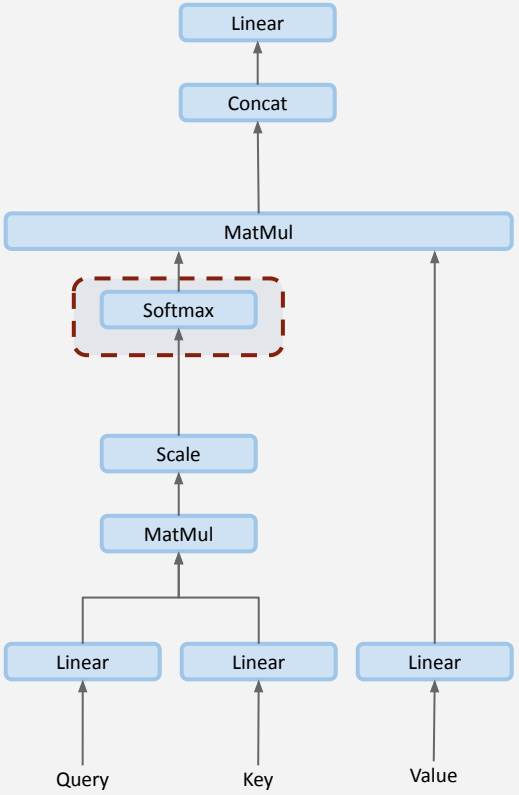
	What	time	is	it
What	86	23	26	66
time	55	95	54	72
is	95	19	98	63
it	36	44	30	73

Self-Attention

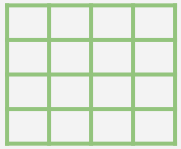




Self-Attention



softmax ( ) =

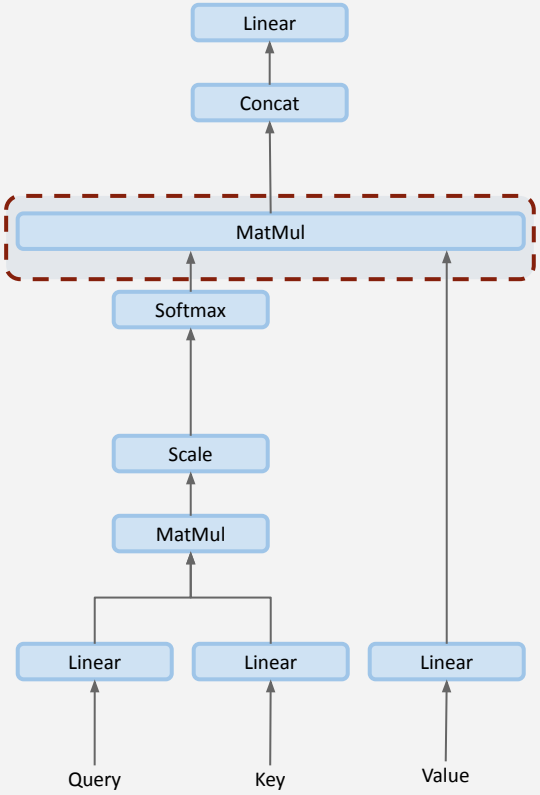


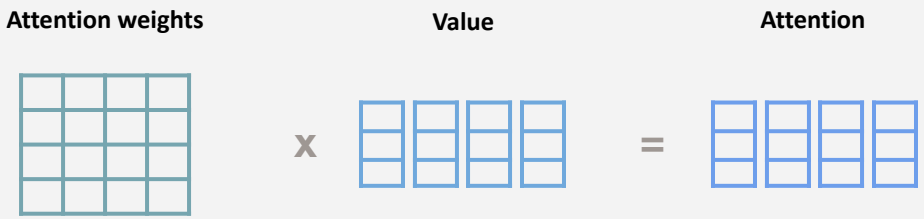
$$soft\max(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Attention weights

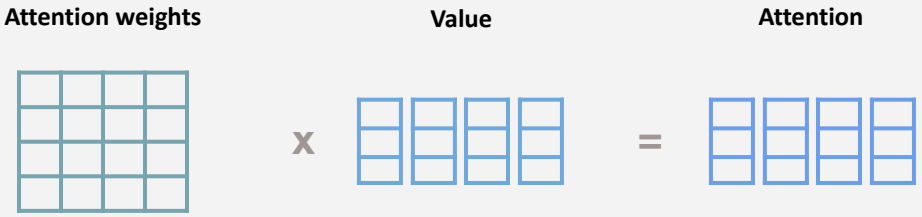
	What	time	is	it
What	0.6	0.1	0.2	0.1
time	0.1	0.7	0.1	0.1
is	0.1	0.2	0.6	0.1
it	0.1	0.2	0.2	0.5

Self-Attention

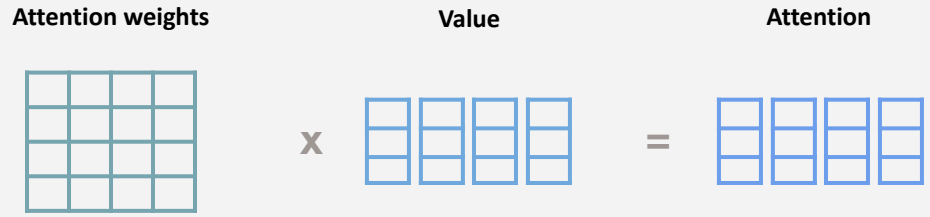




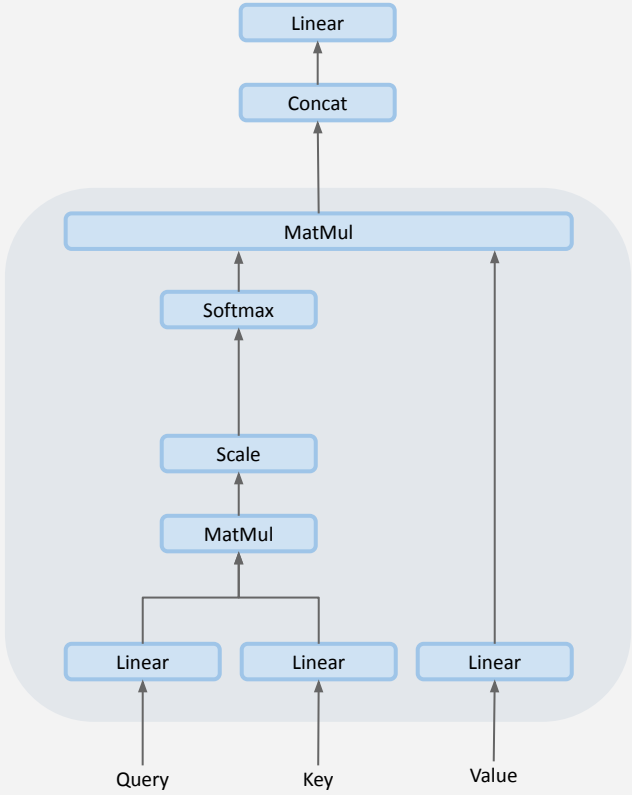
Self-Attention: Intuition

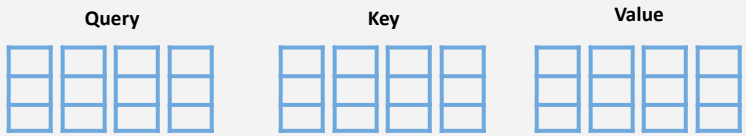


Self-Attention: Intuition

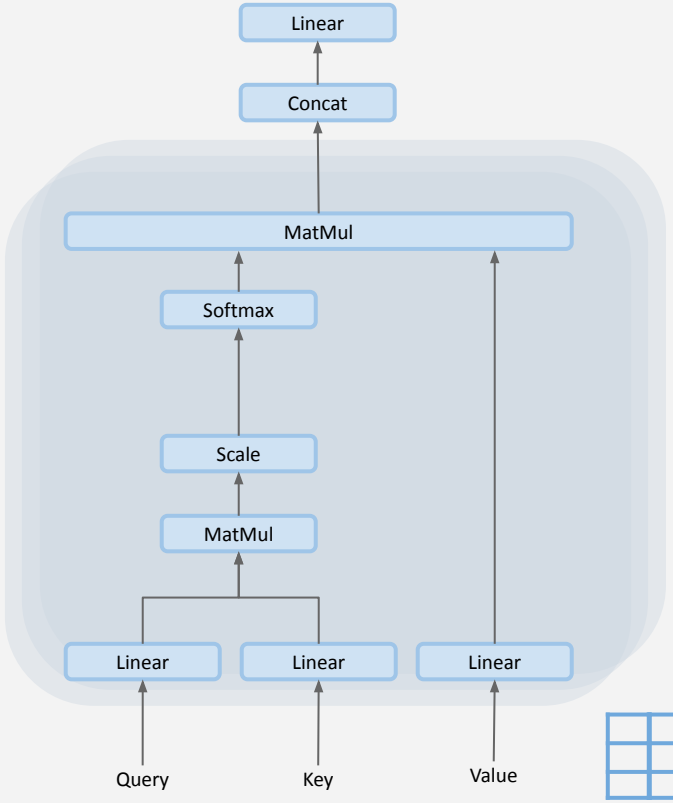


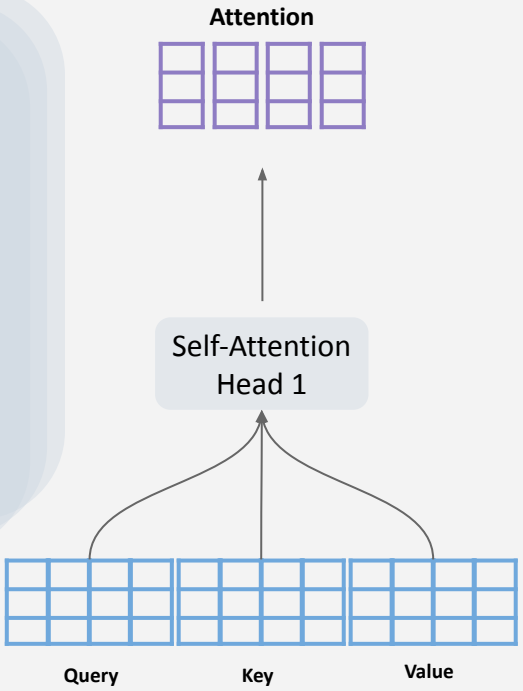
Self-Attention



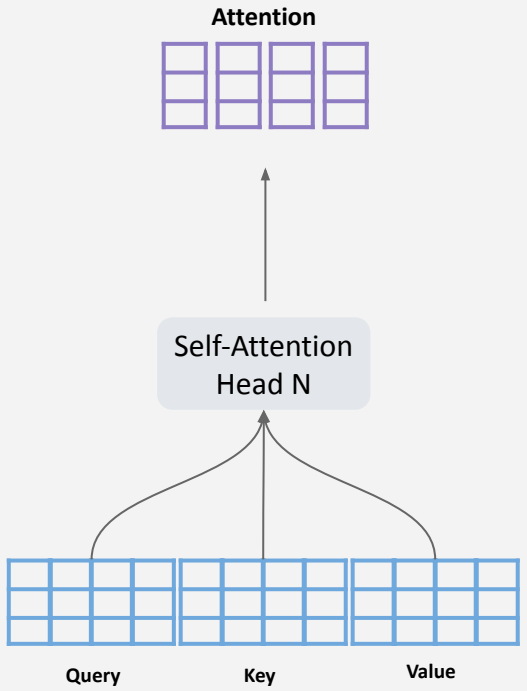


Multi-headed Attention



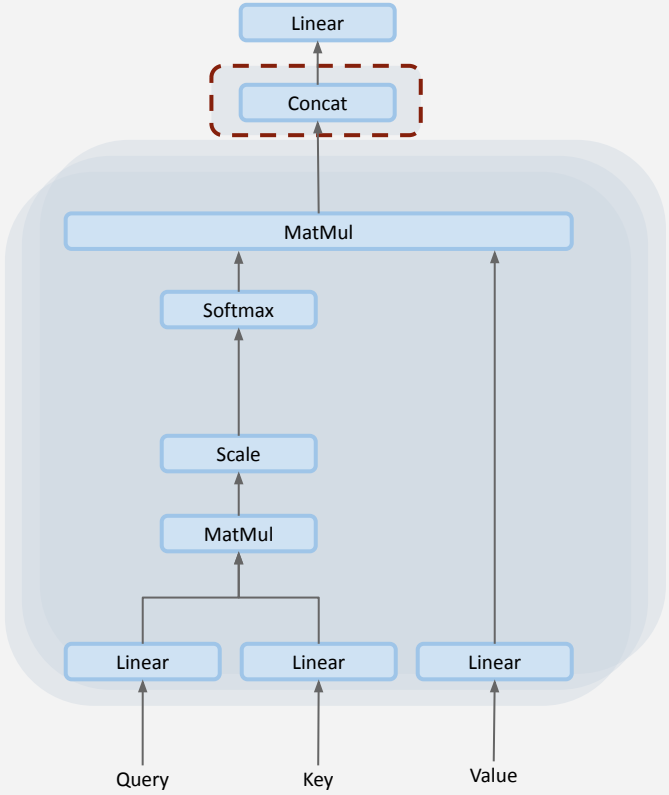


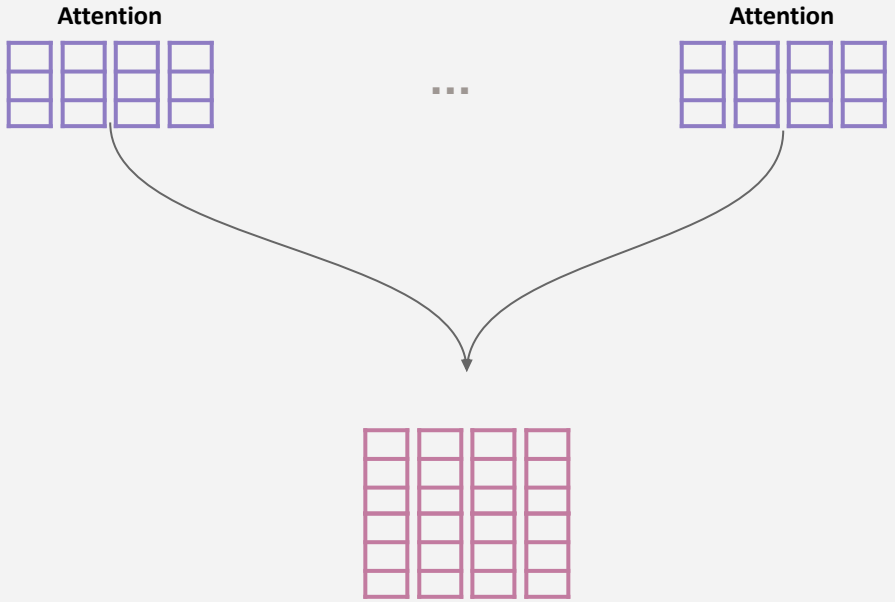
...



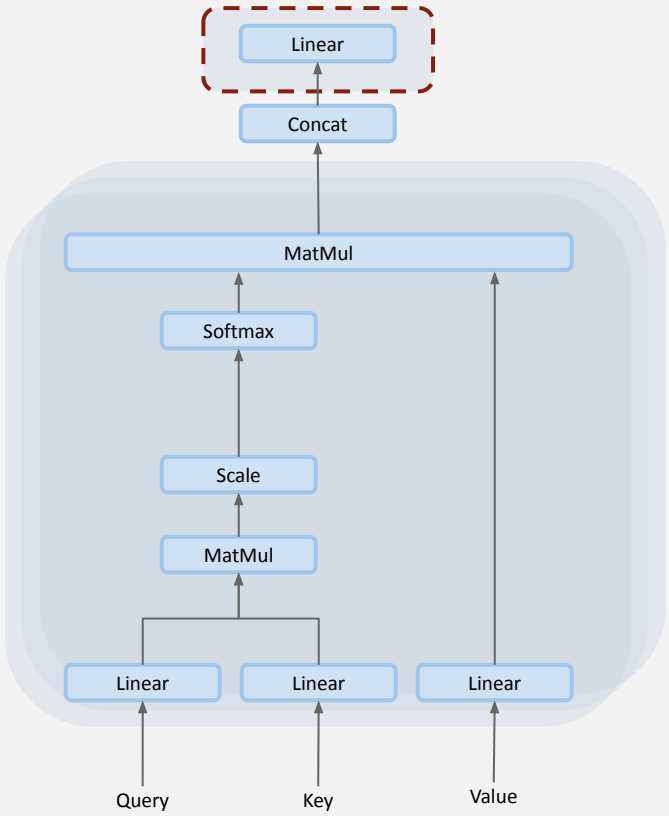
N

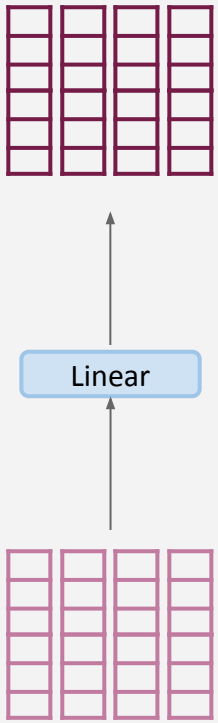
Multi-headed Attention



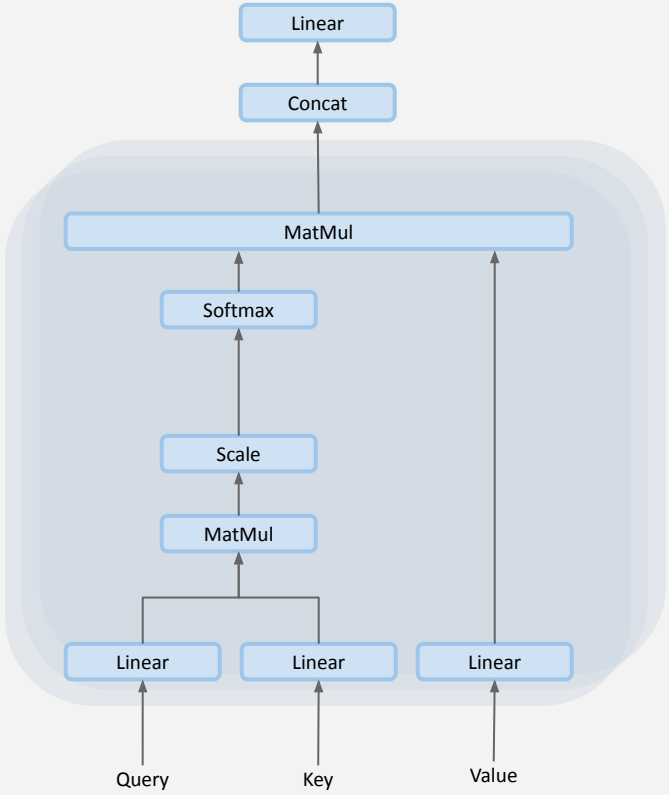


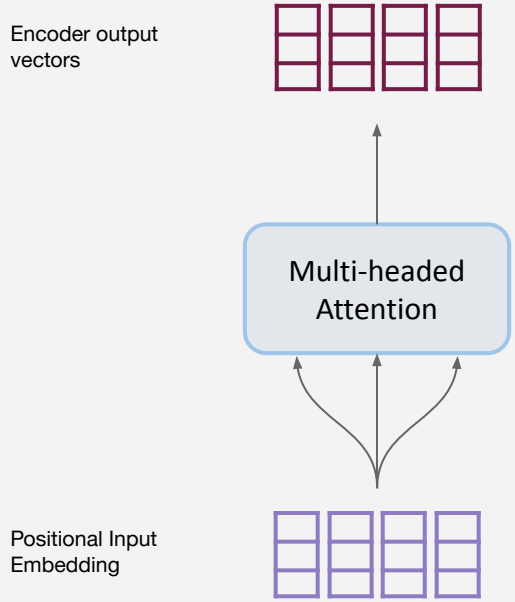
Multi-headed Attention





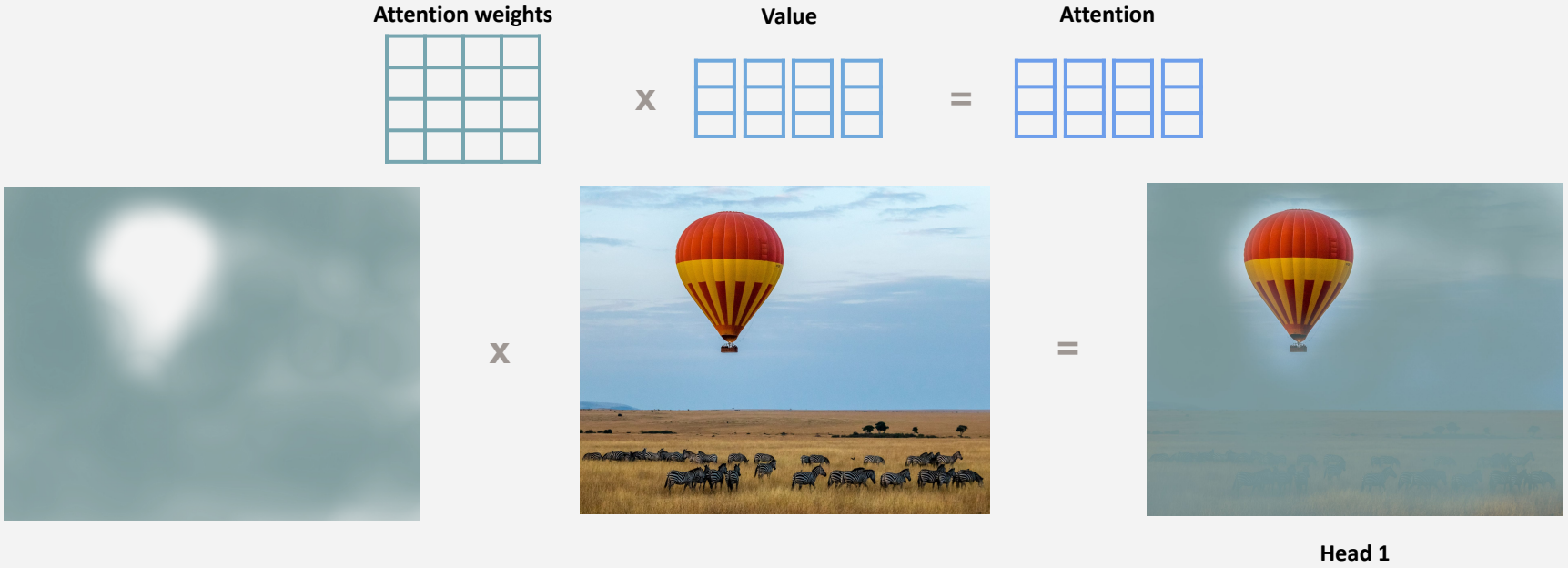
Multi-headed Attention



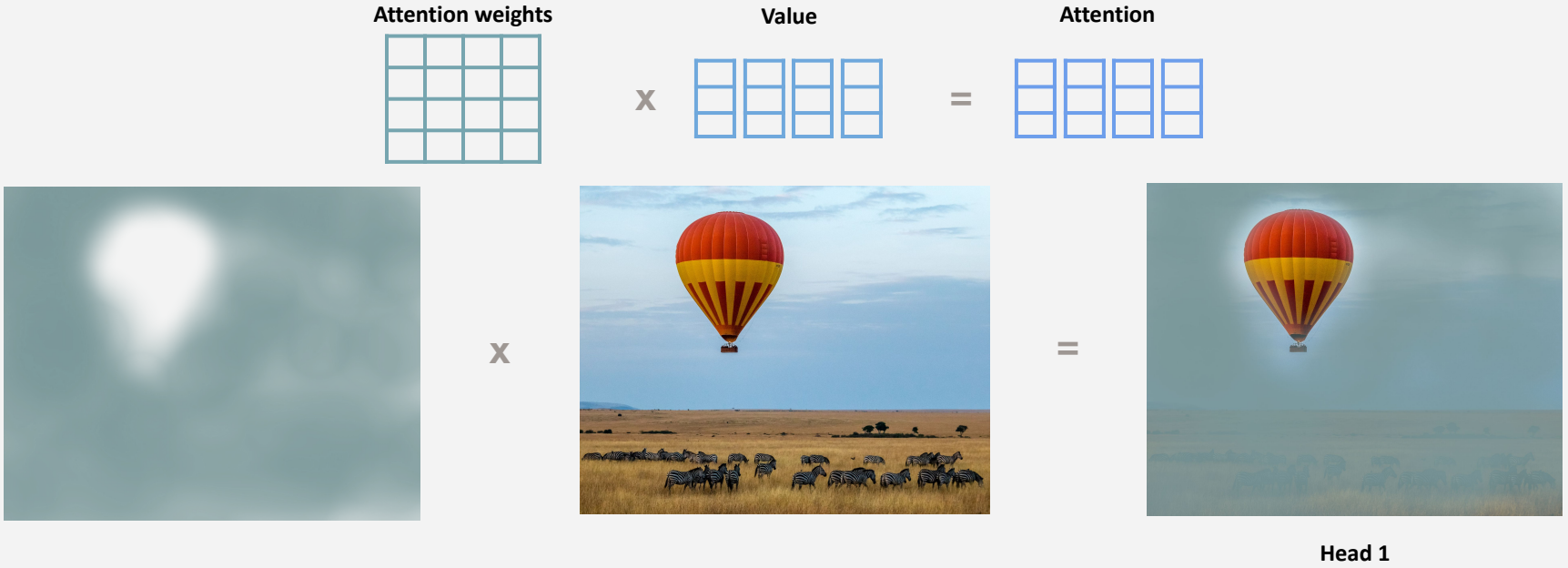


Multi-headed Attention is a module that computes the attention weights for input and produces an output that encodes information on how each word should attend to every other word

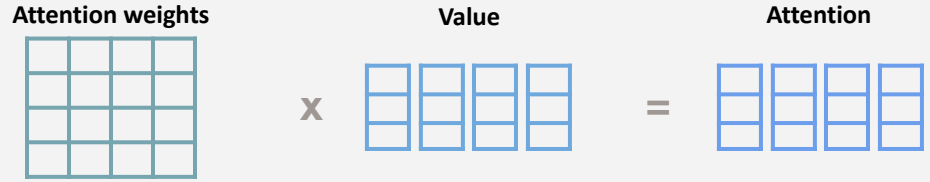
Multi-headed Attention: Intuition



Multi-headed Attention: Intuition



Multi-headed Attention: Intuition

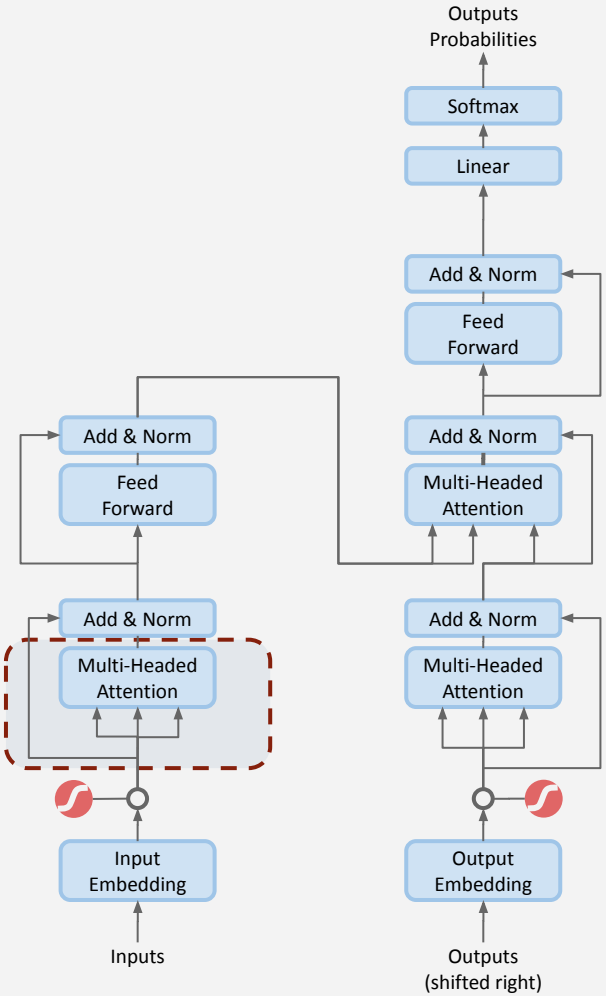


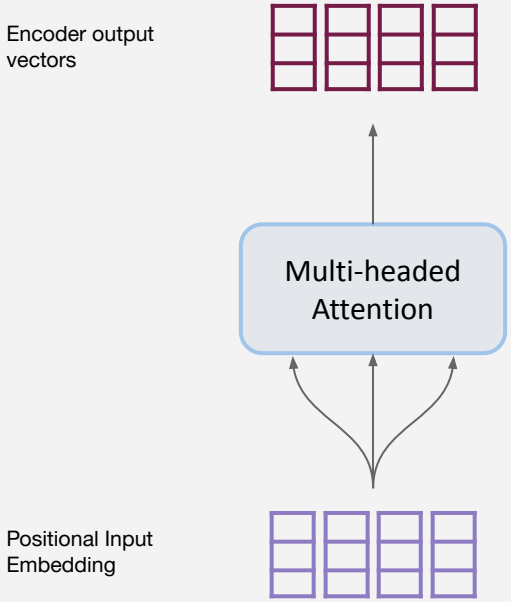
Head 1

Head 2

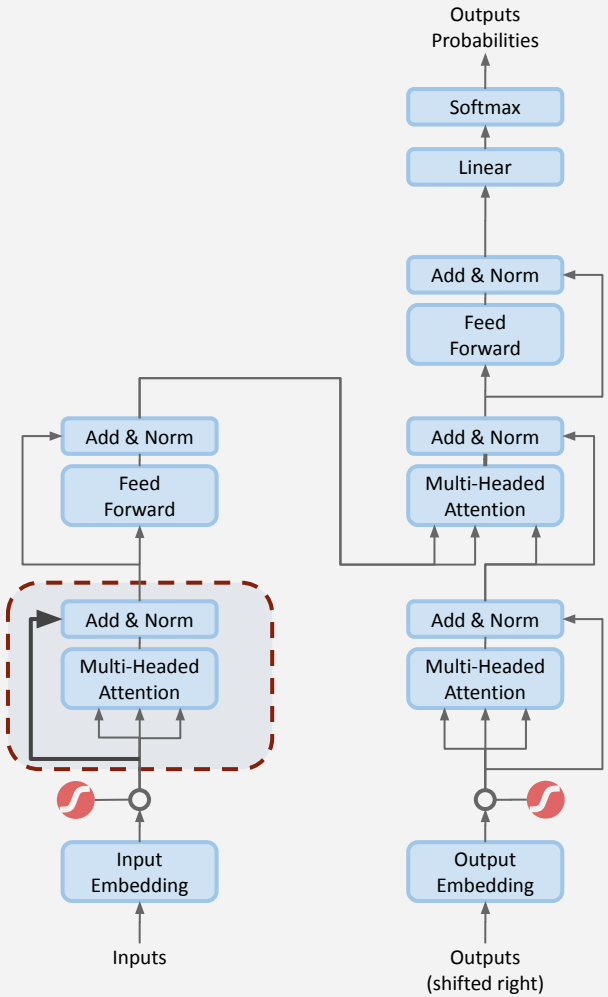
Head n

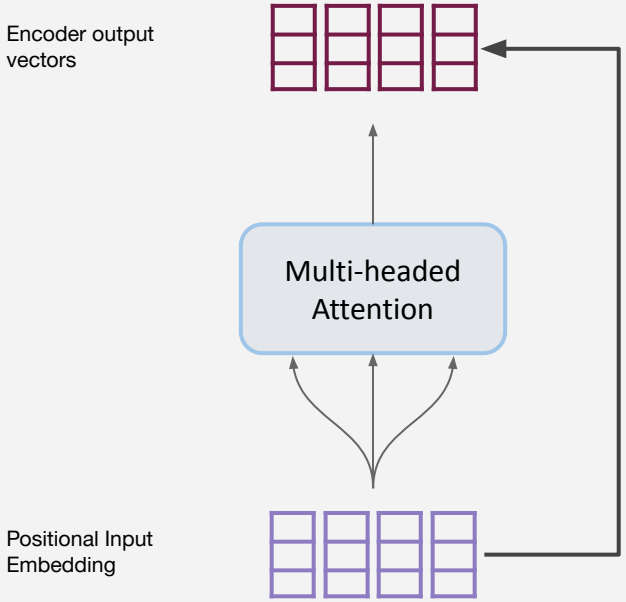
Encoder



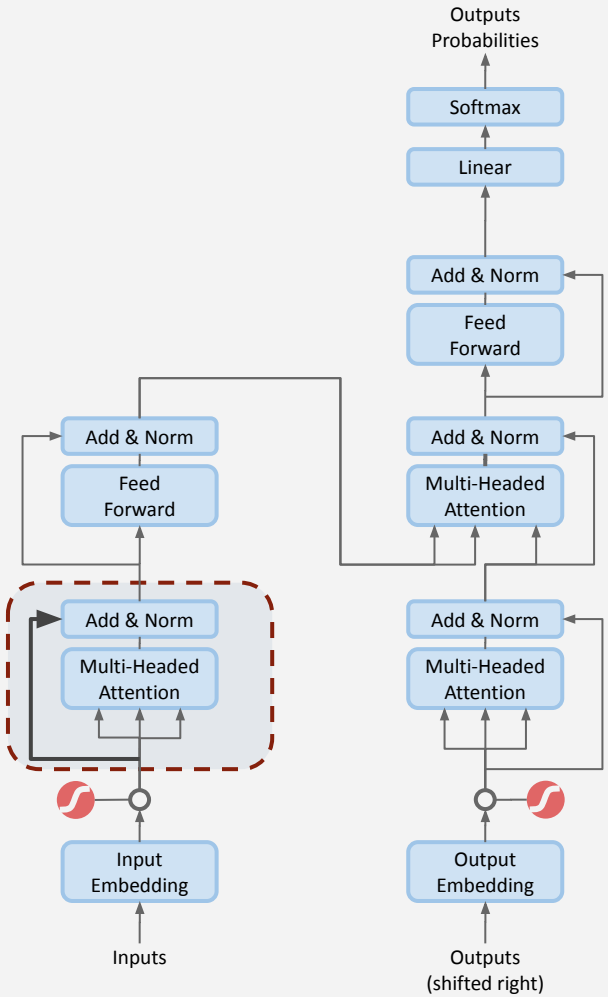


Encoder

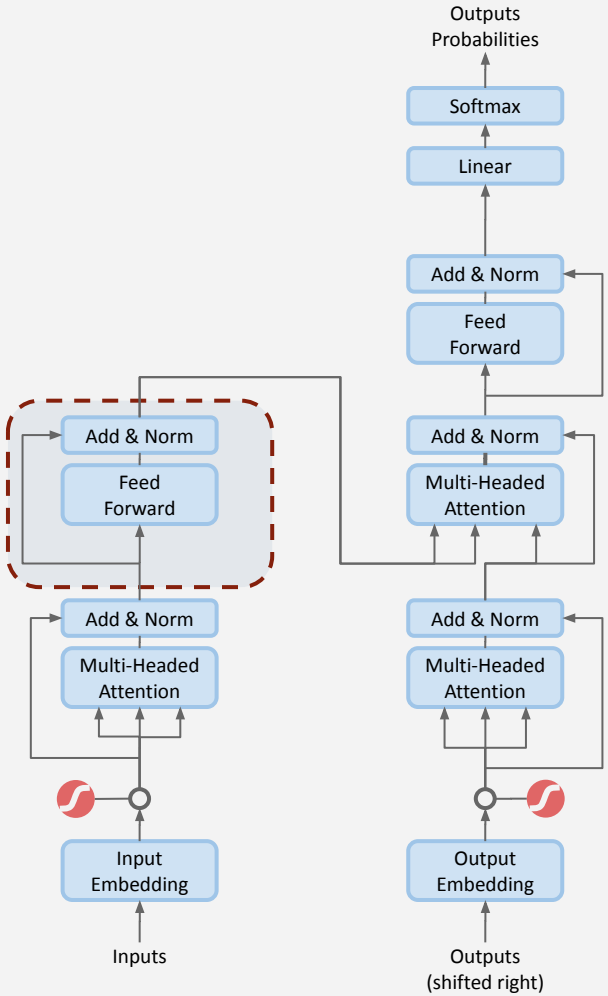


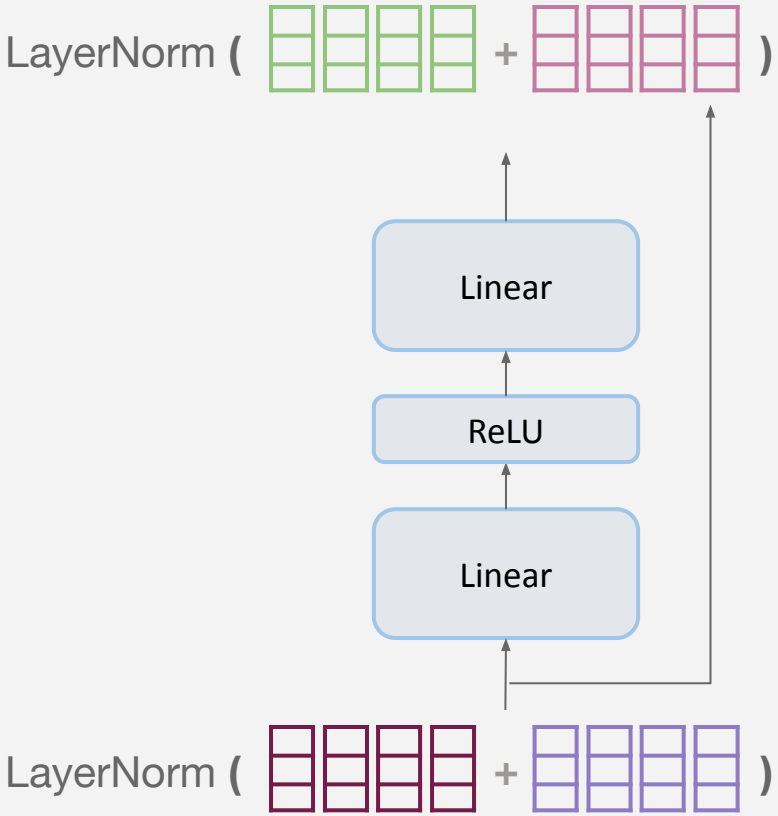


Encoder



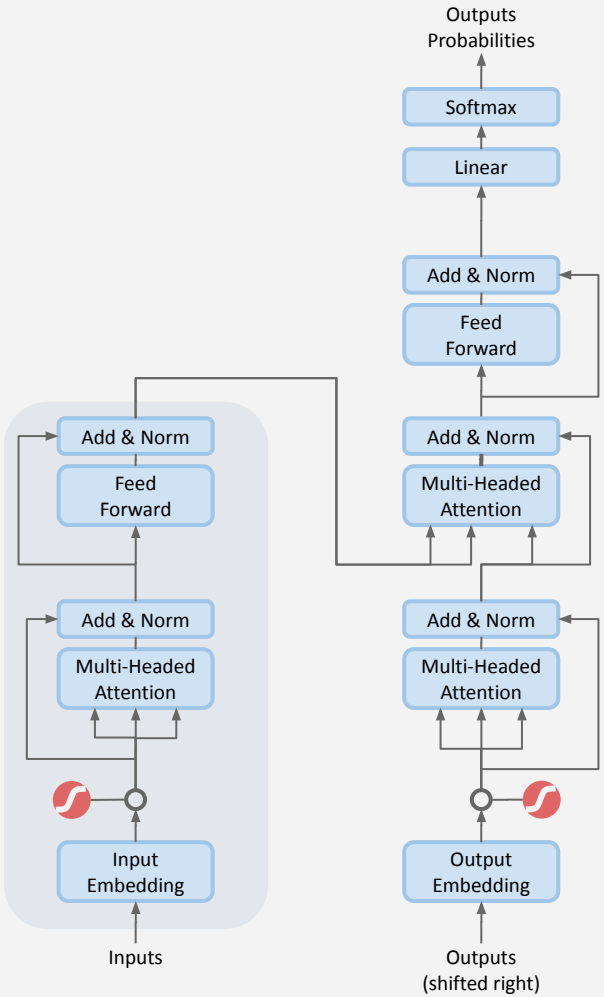
Encoder





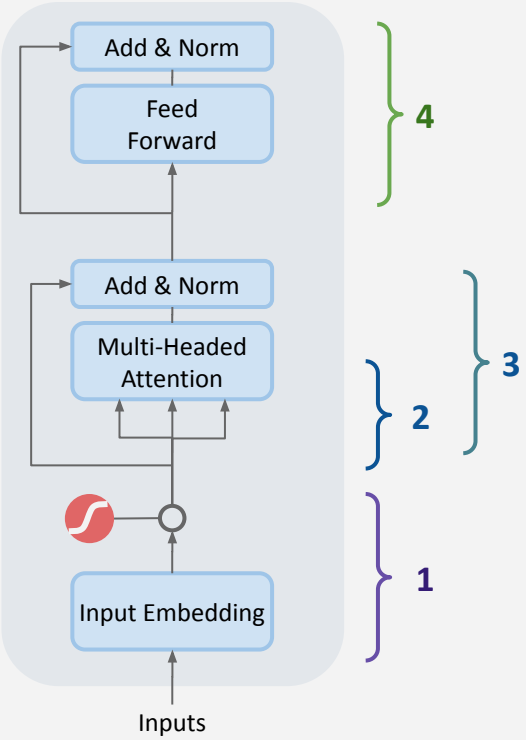
Oregon State University

Encoder



Encoder

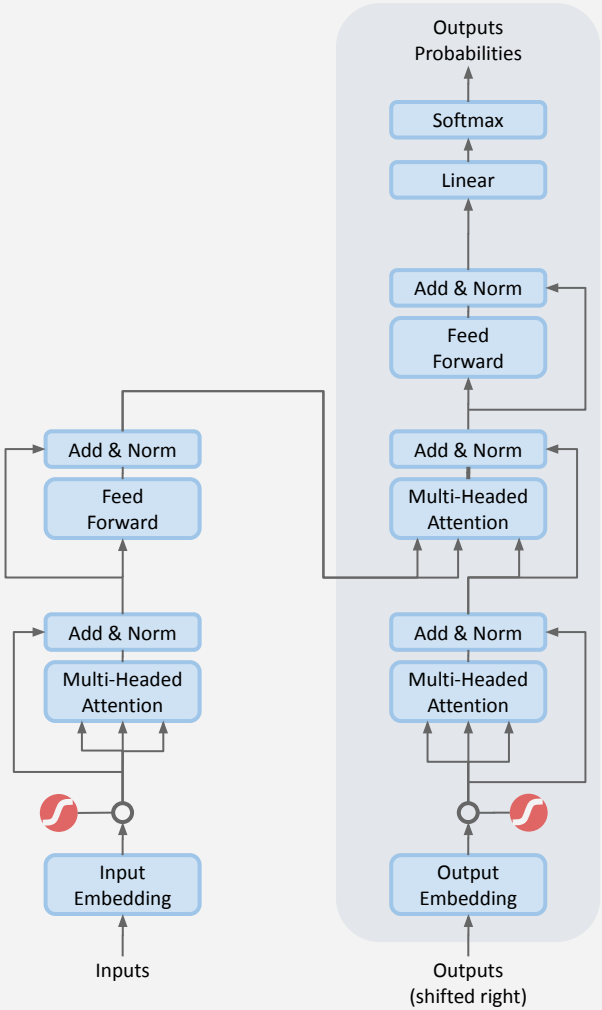
1. **Encode** input + **position** information
2. Learn **query**, **key** and **value** for search
3. Compute **attention weights** for all heads
4. Attend to **features** (=extract) with **high attention**



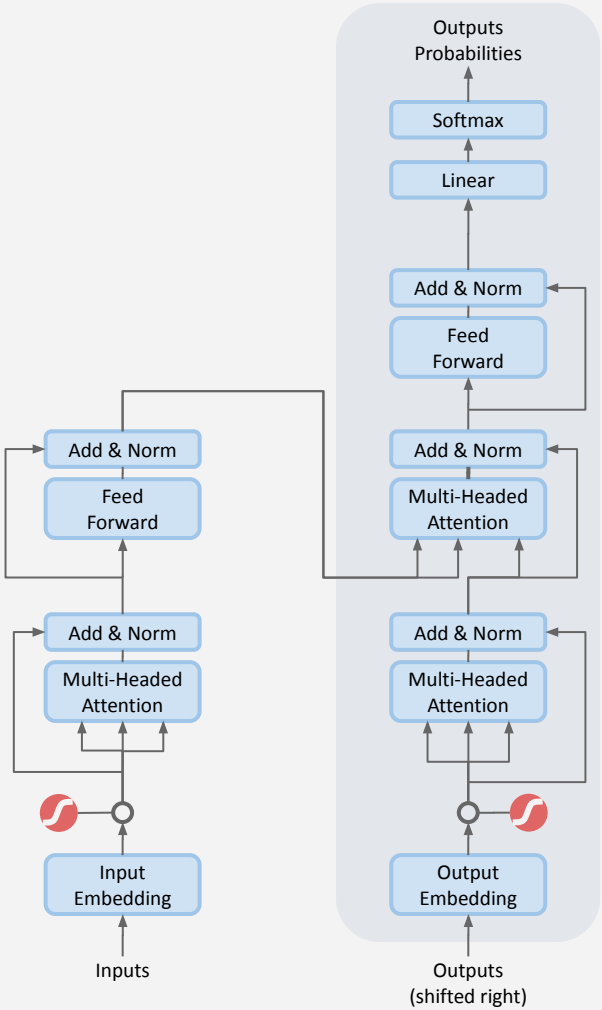
Each head (ideally) **attends to different parts of the input**

The encoder captures important semantic relationships

Decoder



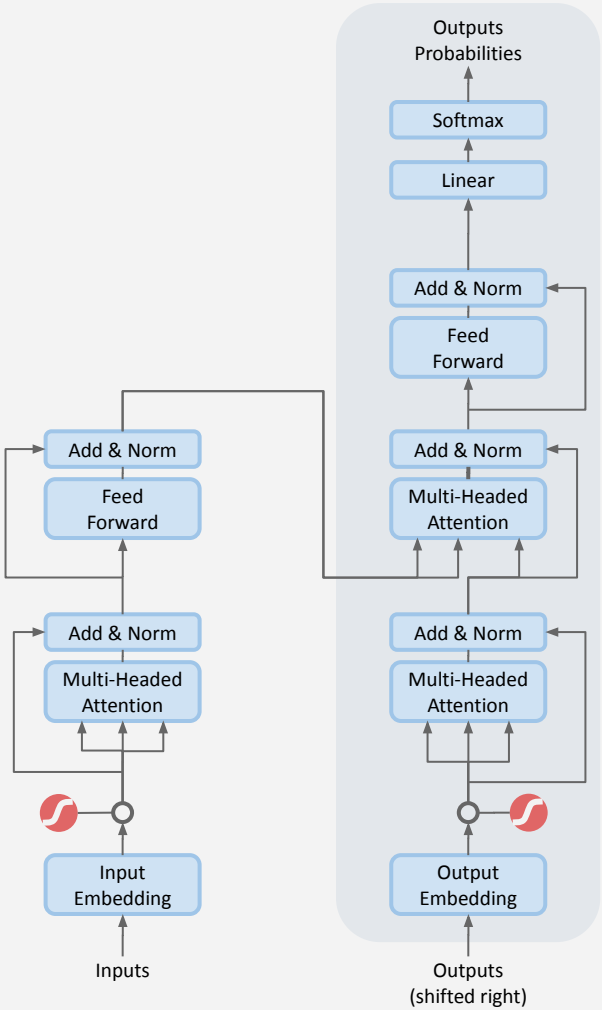
Decoder



What time is it?

Decoder

Decoder

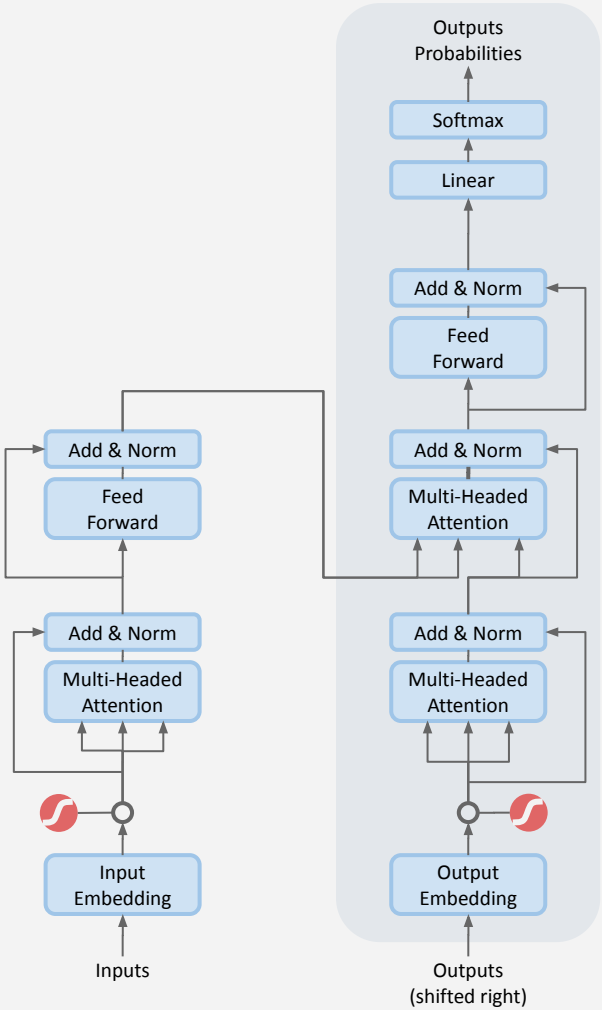


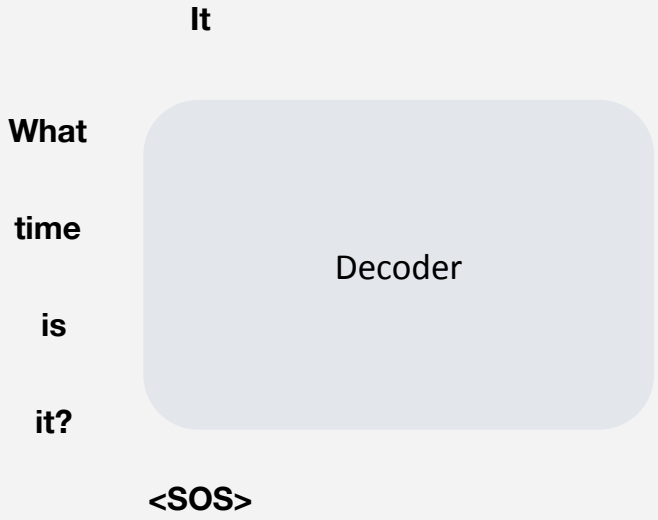
What time is it?

Decoder

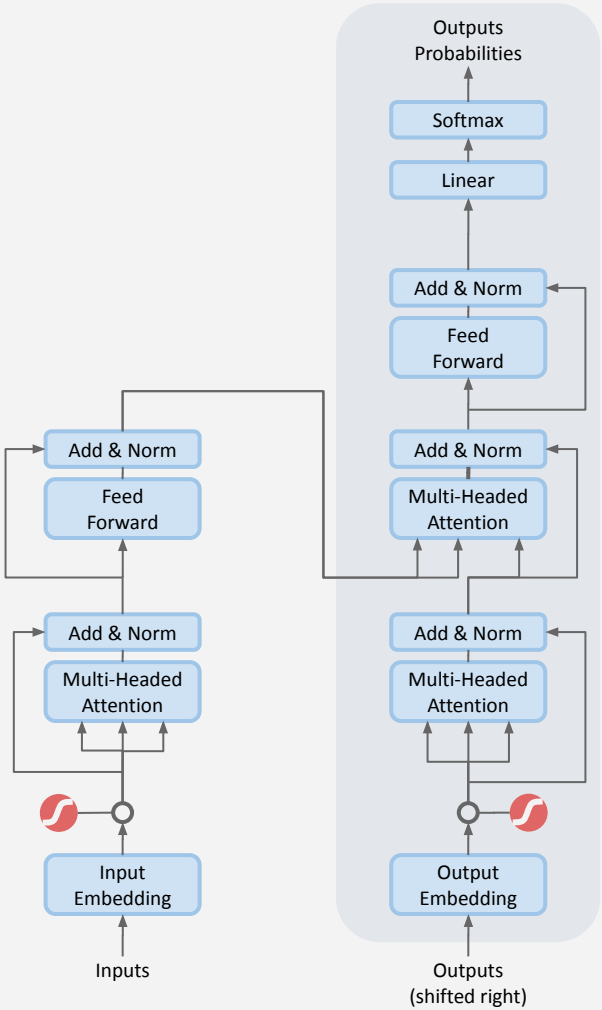
<SOS>

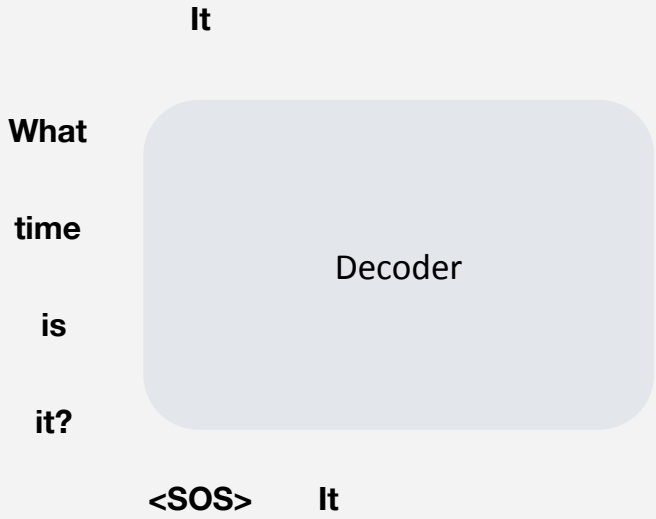
Decoder



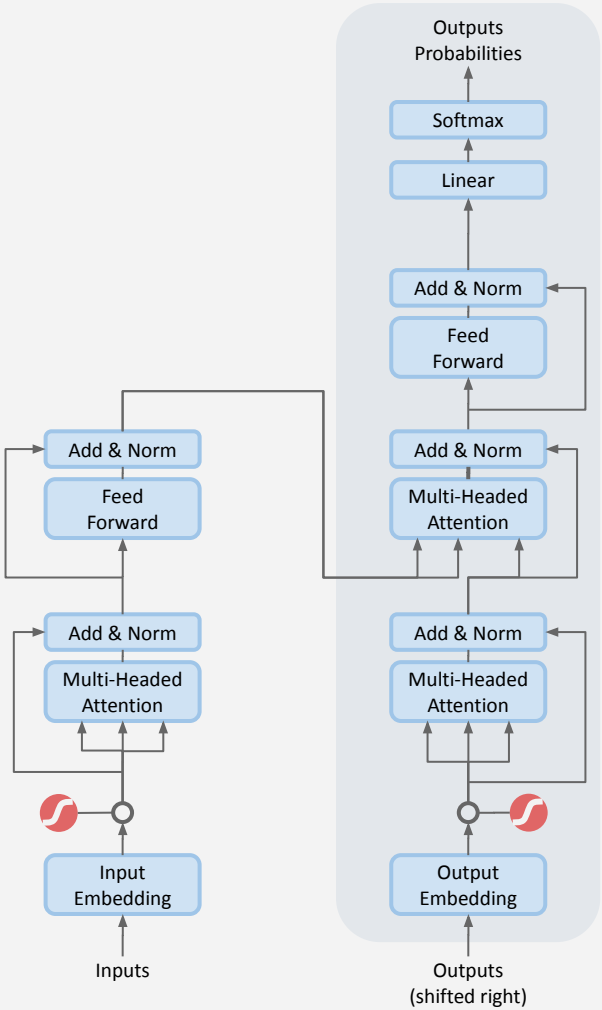


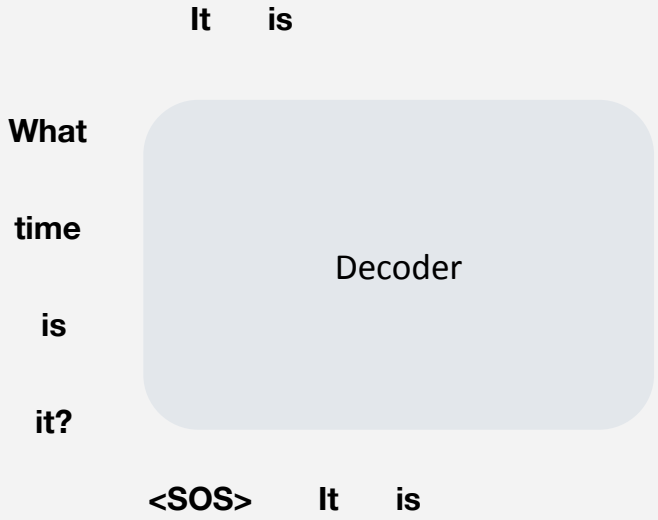
Decoder



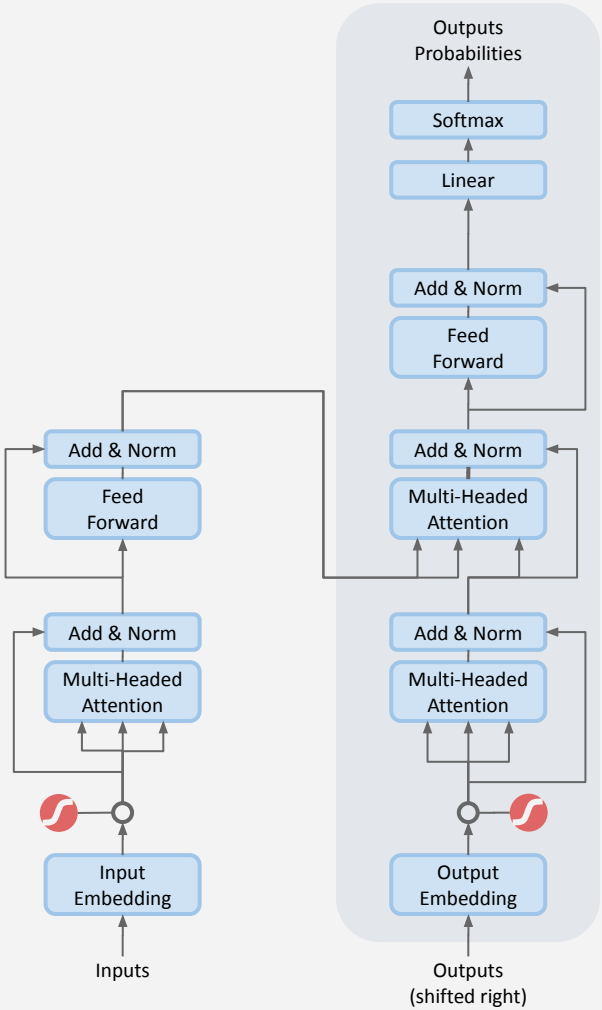


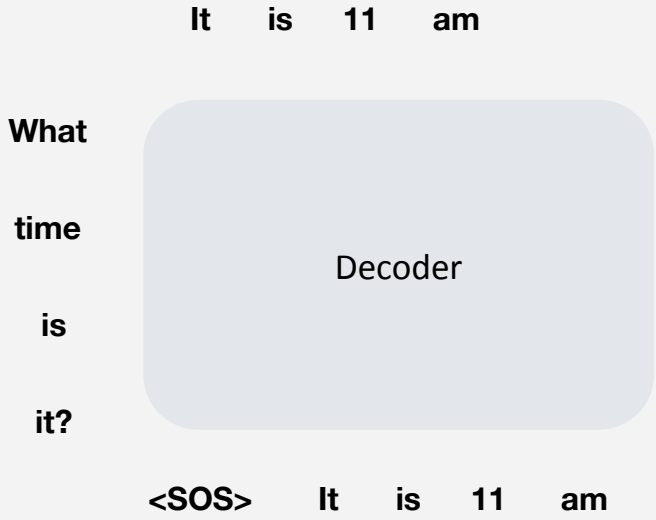
Decoder



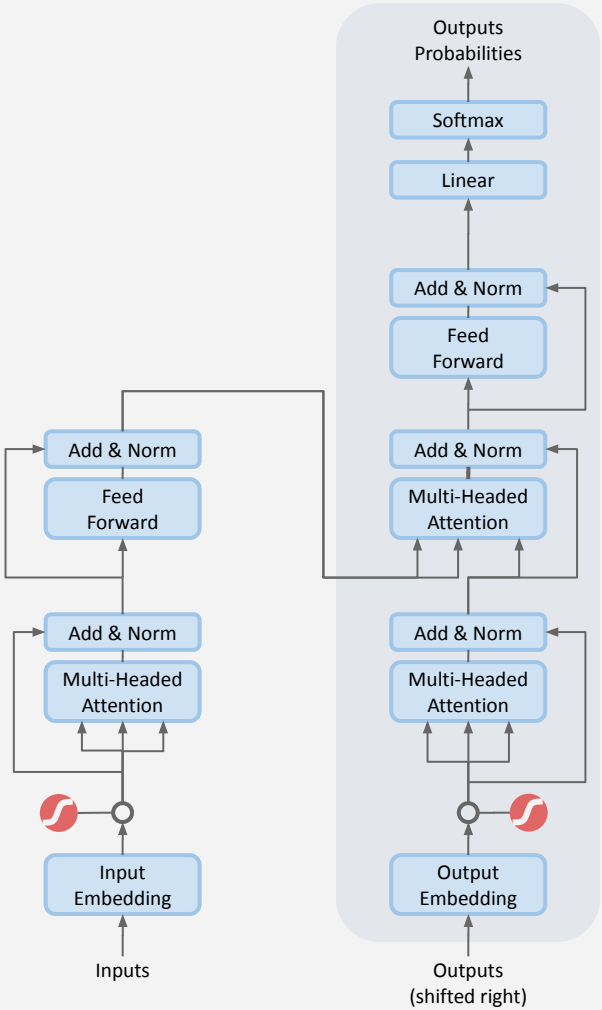


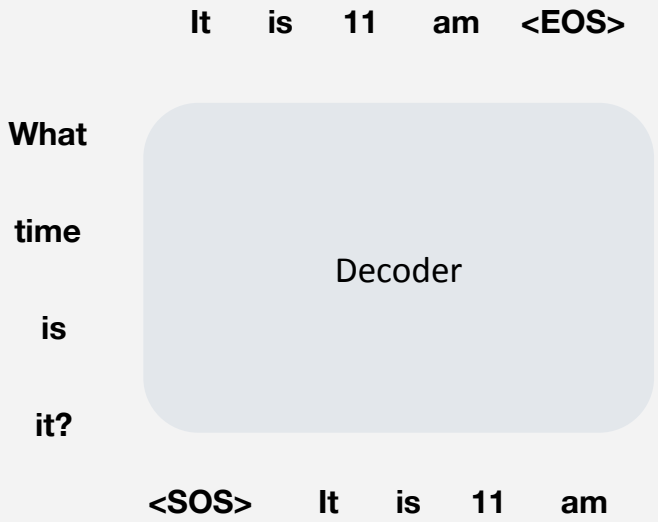
Decoder



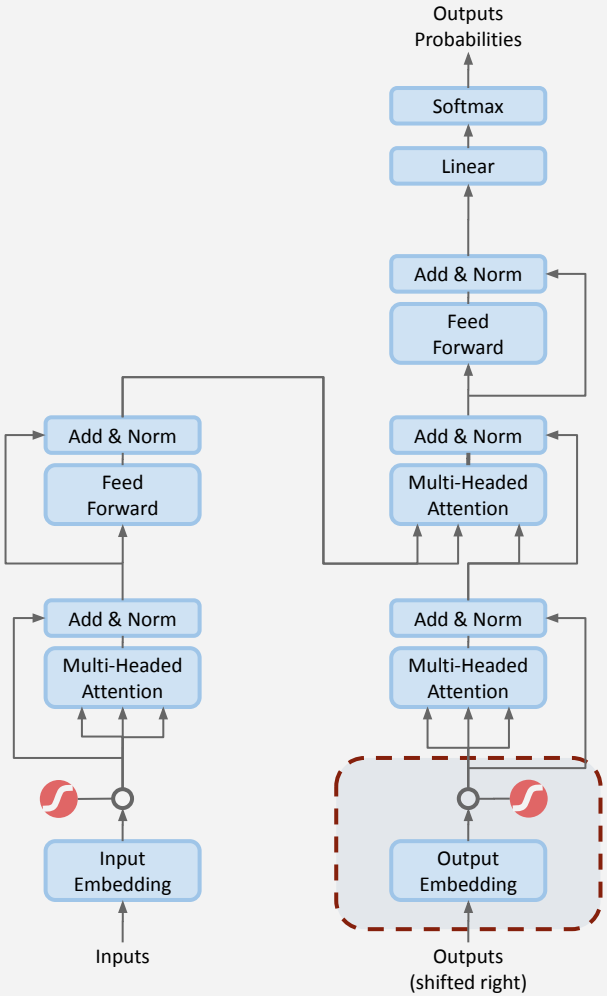


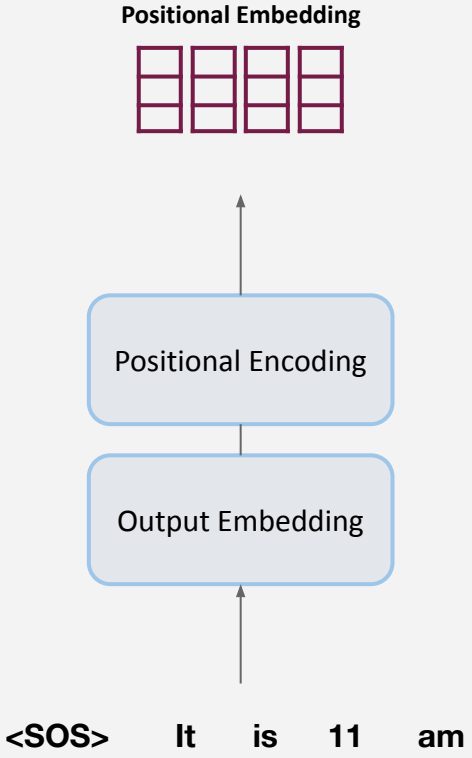
Decoder



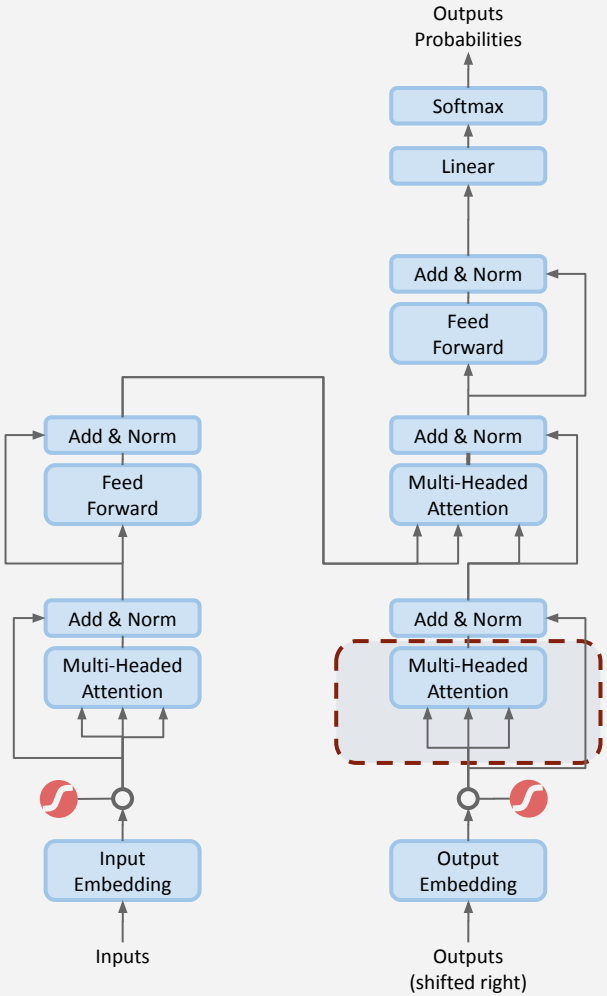


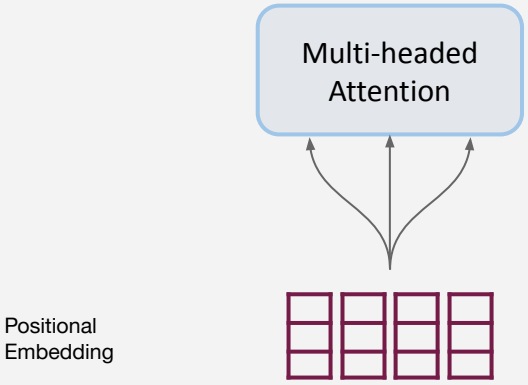
Decoder



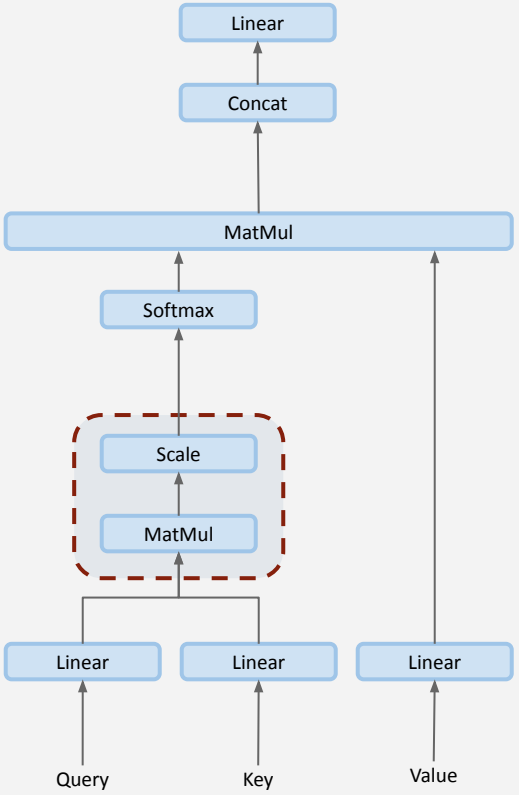


Decoder





Self-Attention

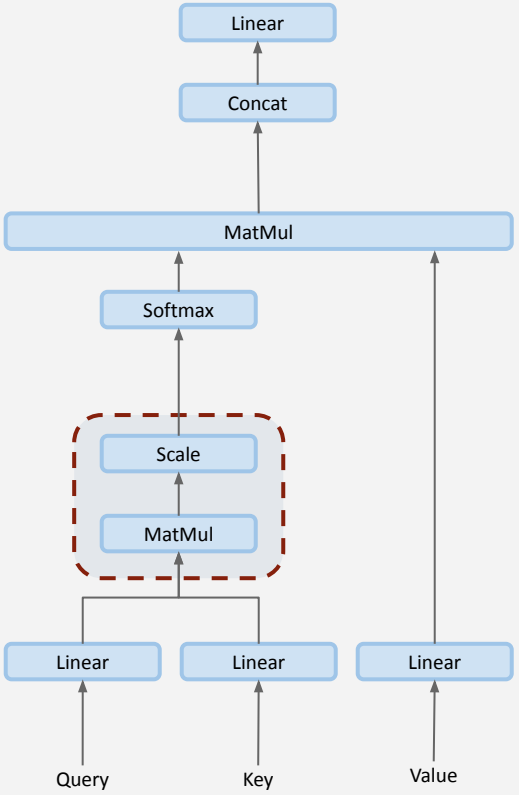


Scaled Scores

	<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

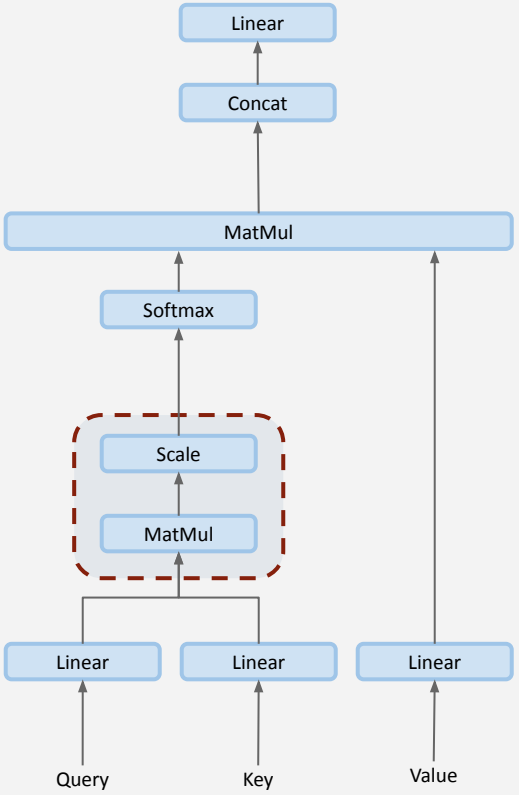


Scaled Scores

	<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

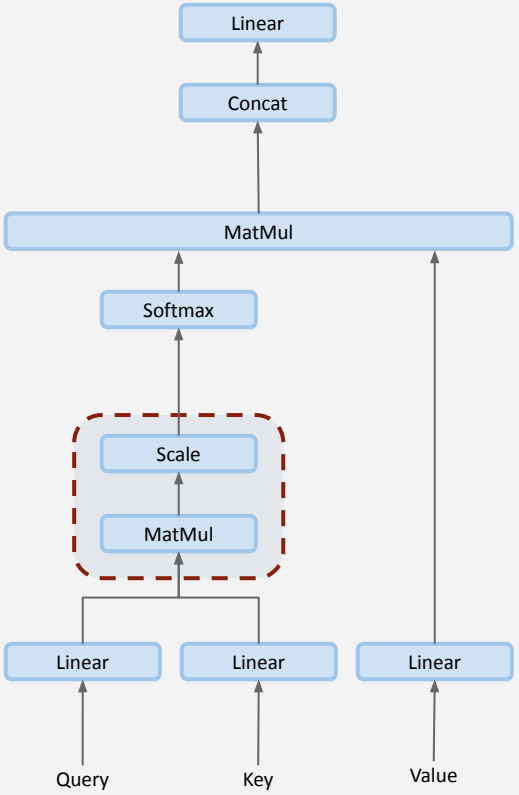


Scaled Scores

	<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

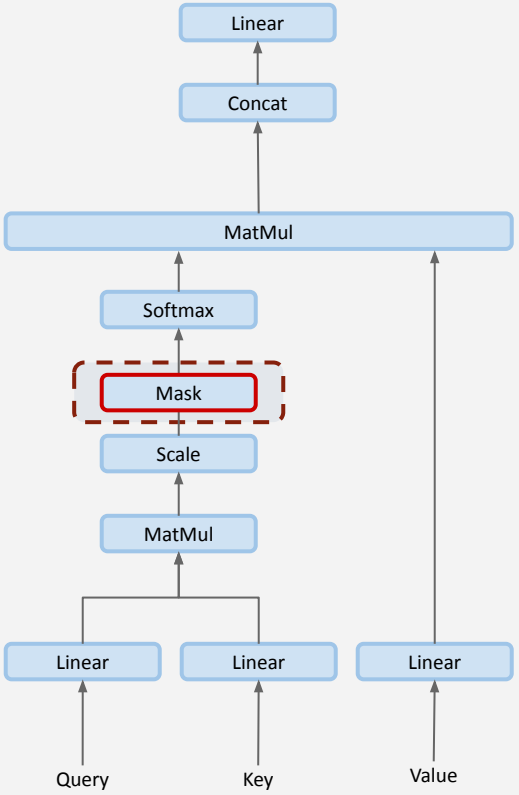


Scaled Scores

	<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: Masking

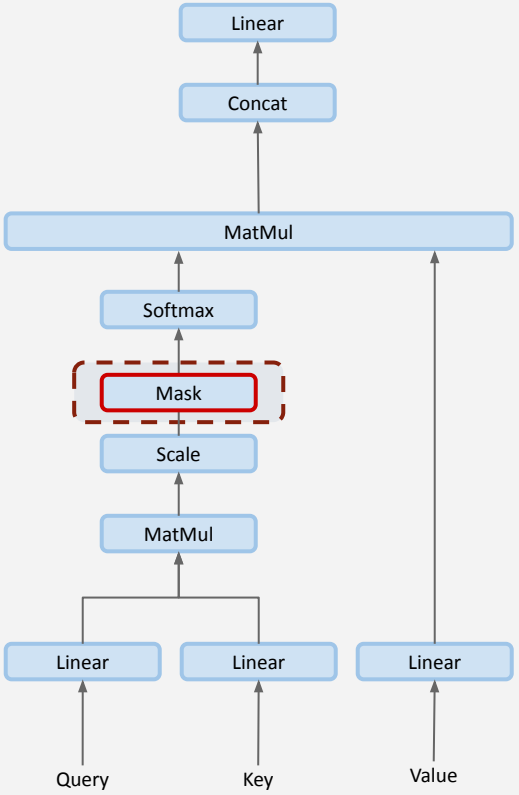


Scaled Scores

	<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

=

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

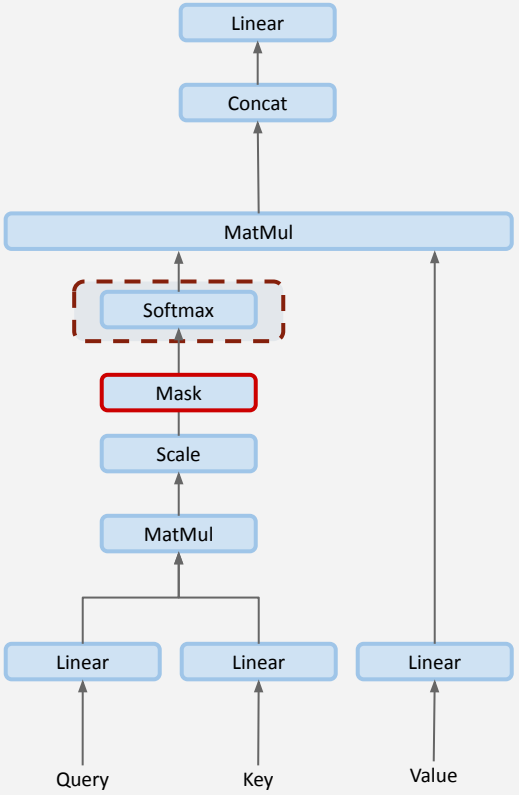
=

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

) =

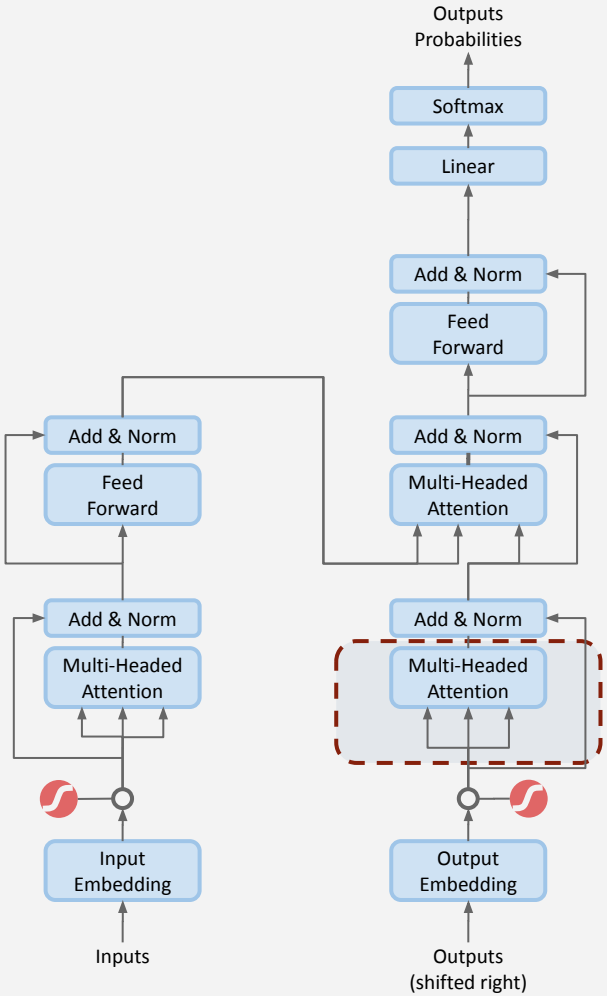
Attention Weights

	<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

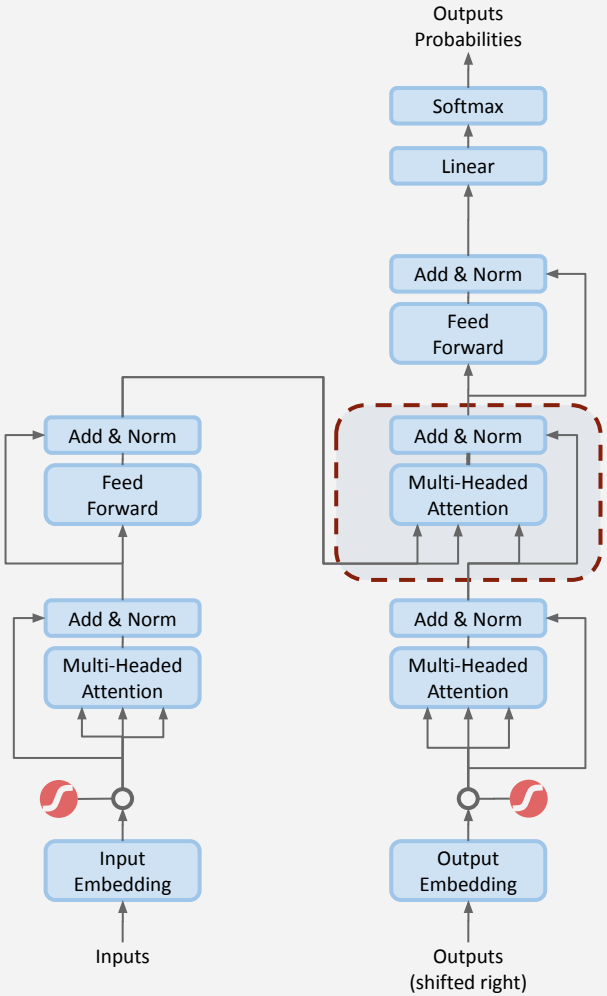
$$soft\max(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

=

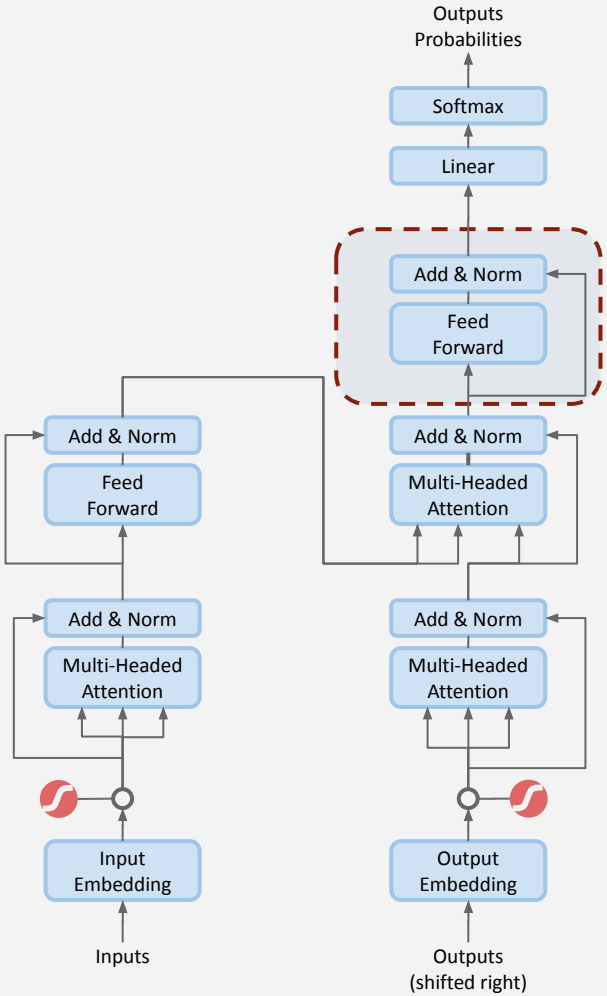
Decoder



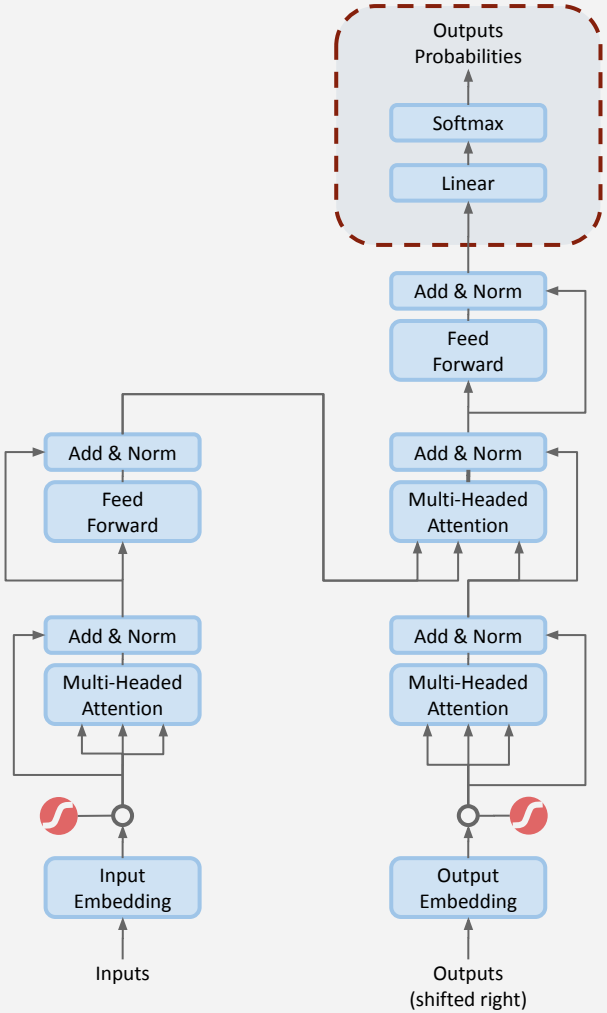
Decoder

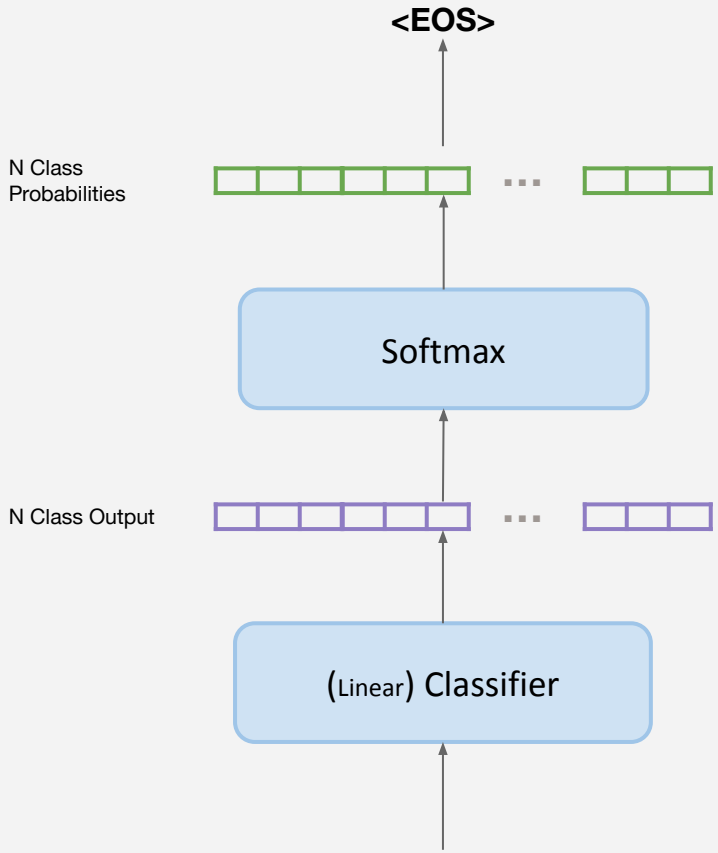


Decoder

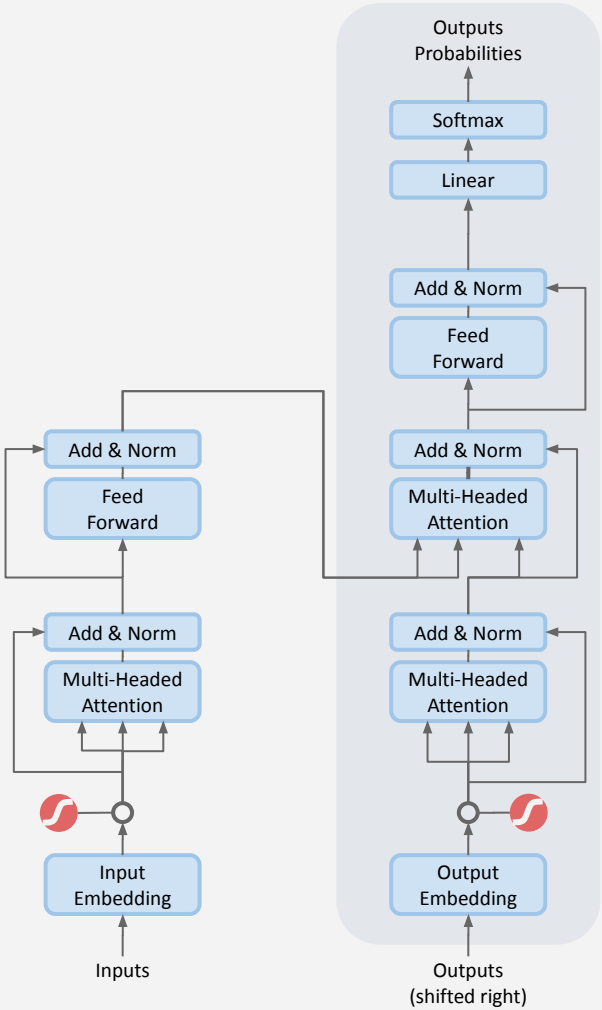


Decoder: Linear Classifier



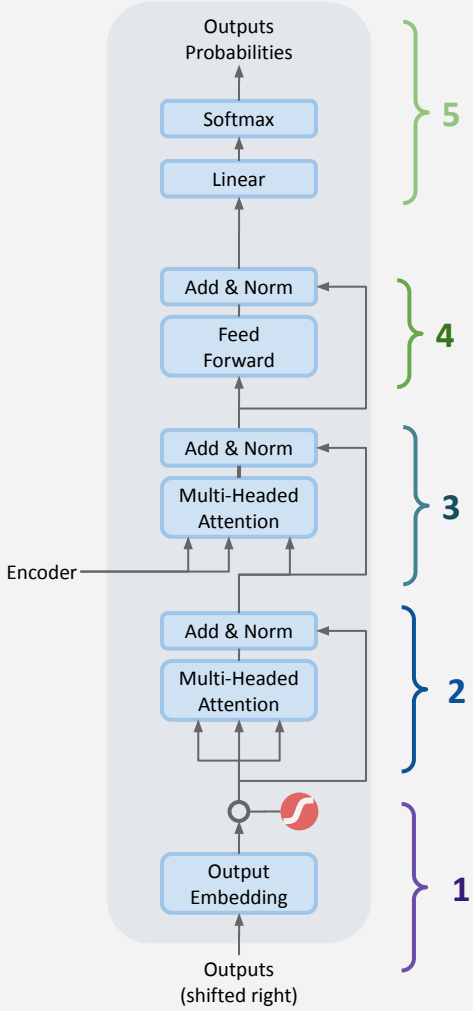


Decoder: Linear Classifier



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

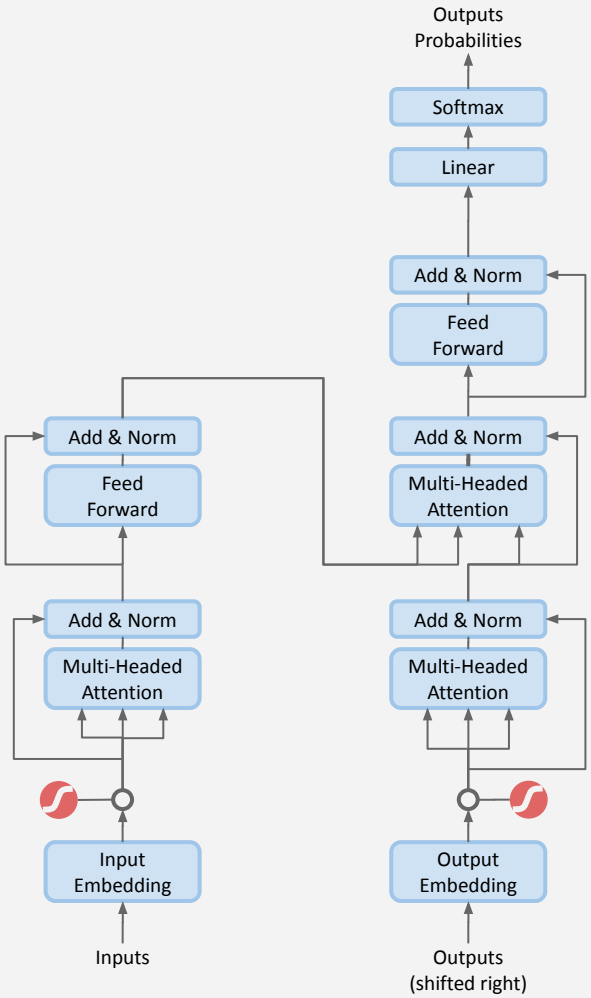


The decoder attends to its past output (it is autoregressive**)**
and to the encoder output (transformer input**)**

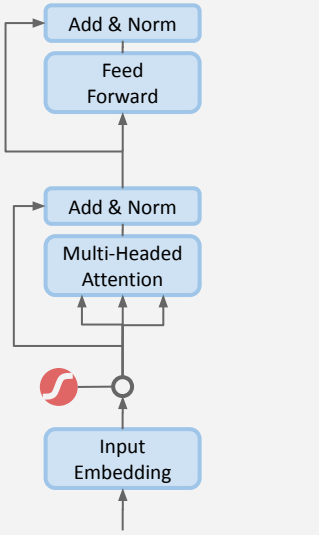
Transformer: Training

Tea and water please → Tee und wasser bitte

Transformer: Training



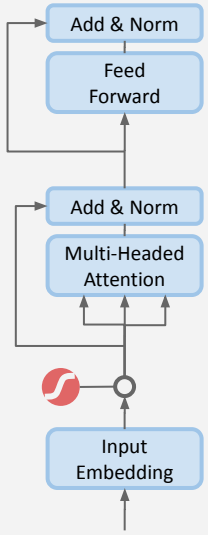
Transformer: Training



<SOS> Tea and water please <EOS>

Transformer: Training

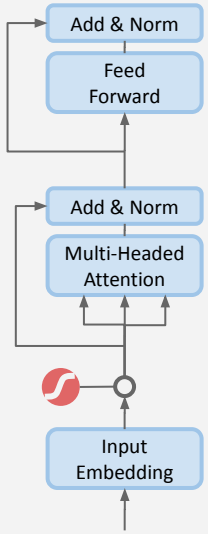
The encoder outputs a vector representation of the input that captures the **positions** and the **semantic relationships** using the multi-head attention.



<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.

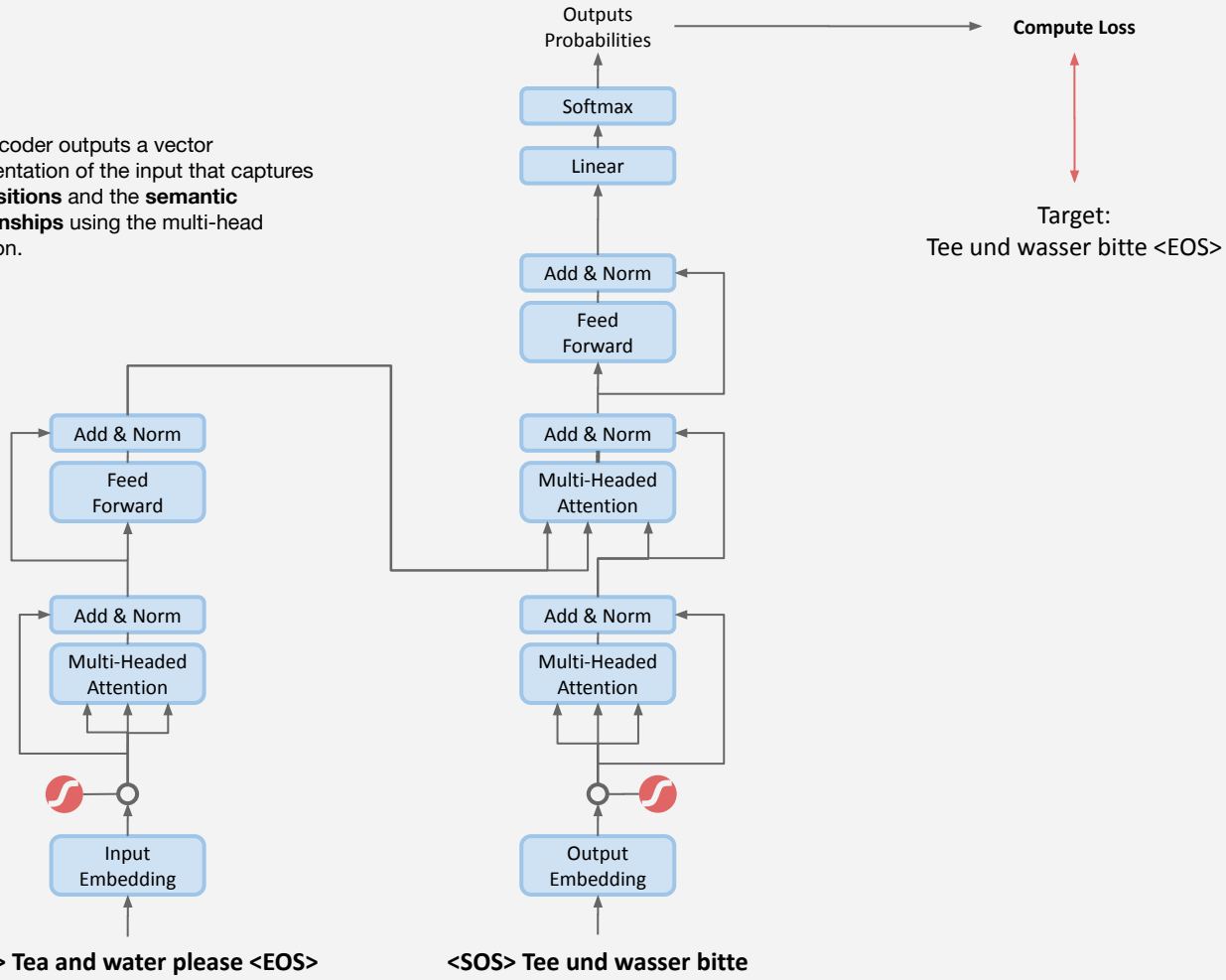


<SOS> Tea and water please <EOS>

<SOS> Tee und wasser bitte

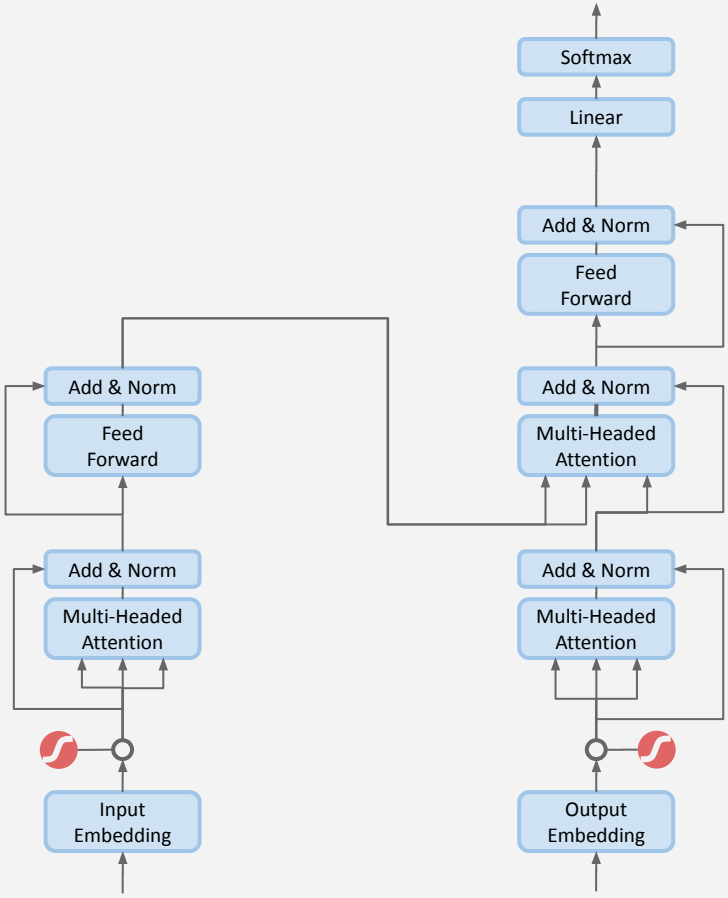
Transformer: Training

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

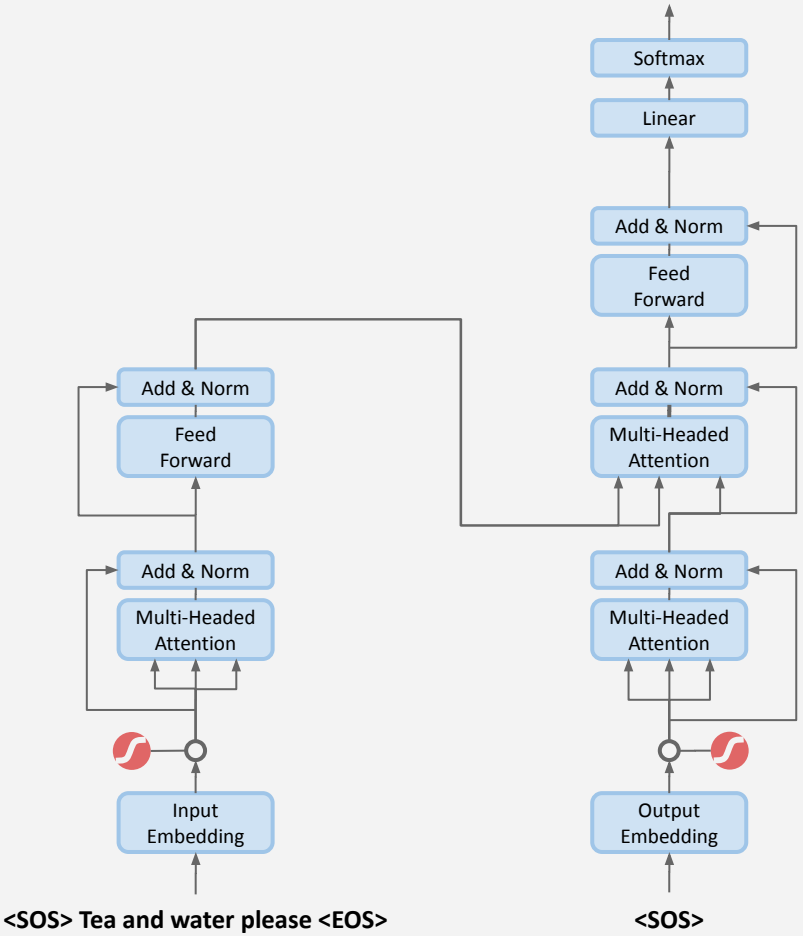


Transformer: Inference

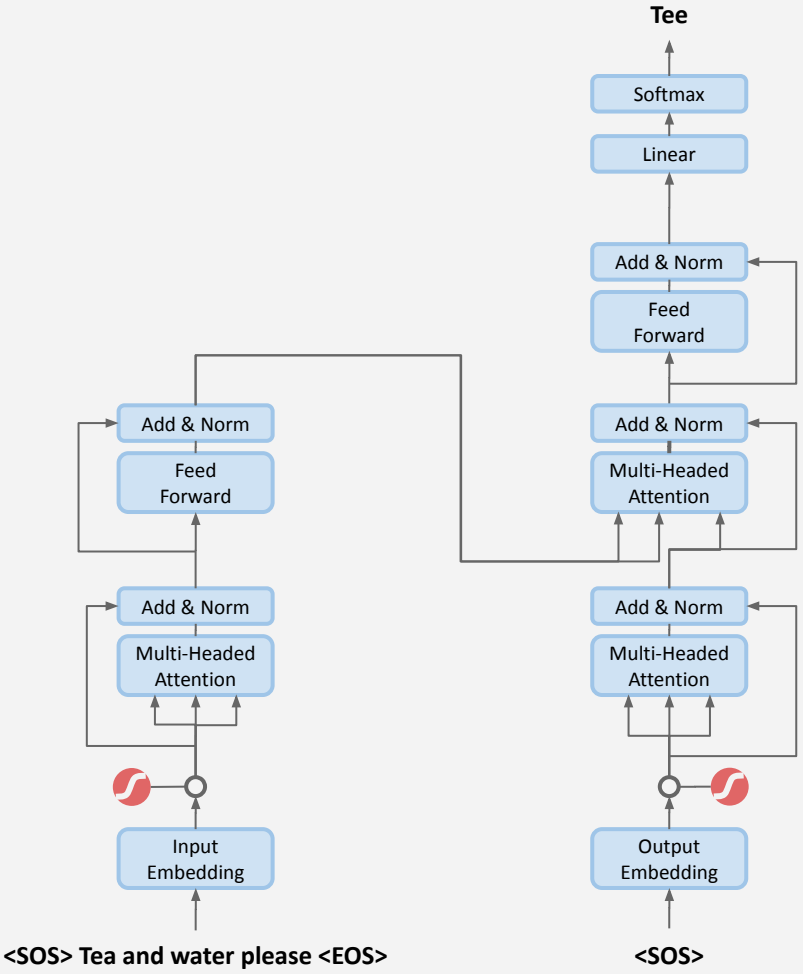
Transformer: Inference



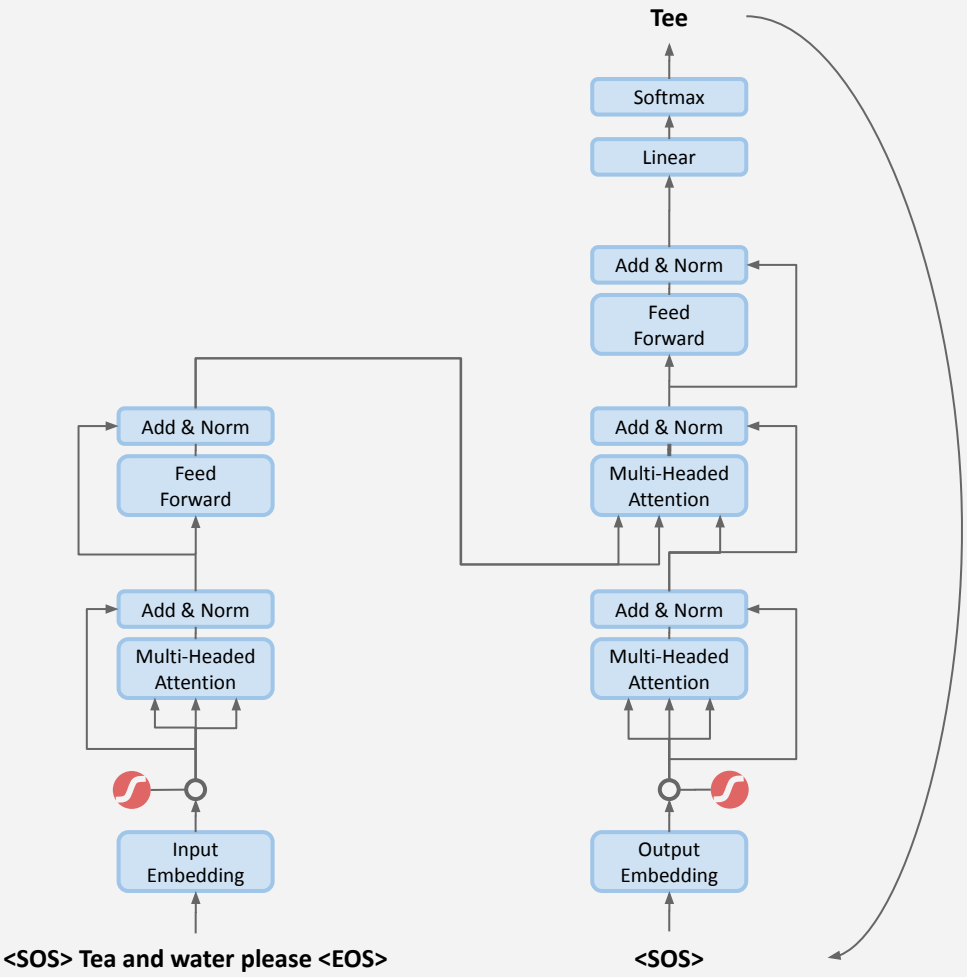
Transformer: Inference



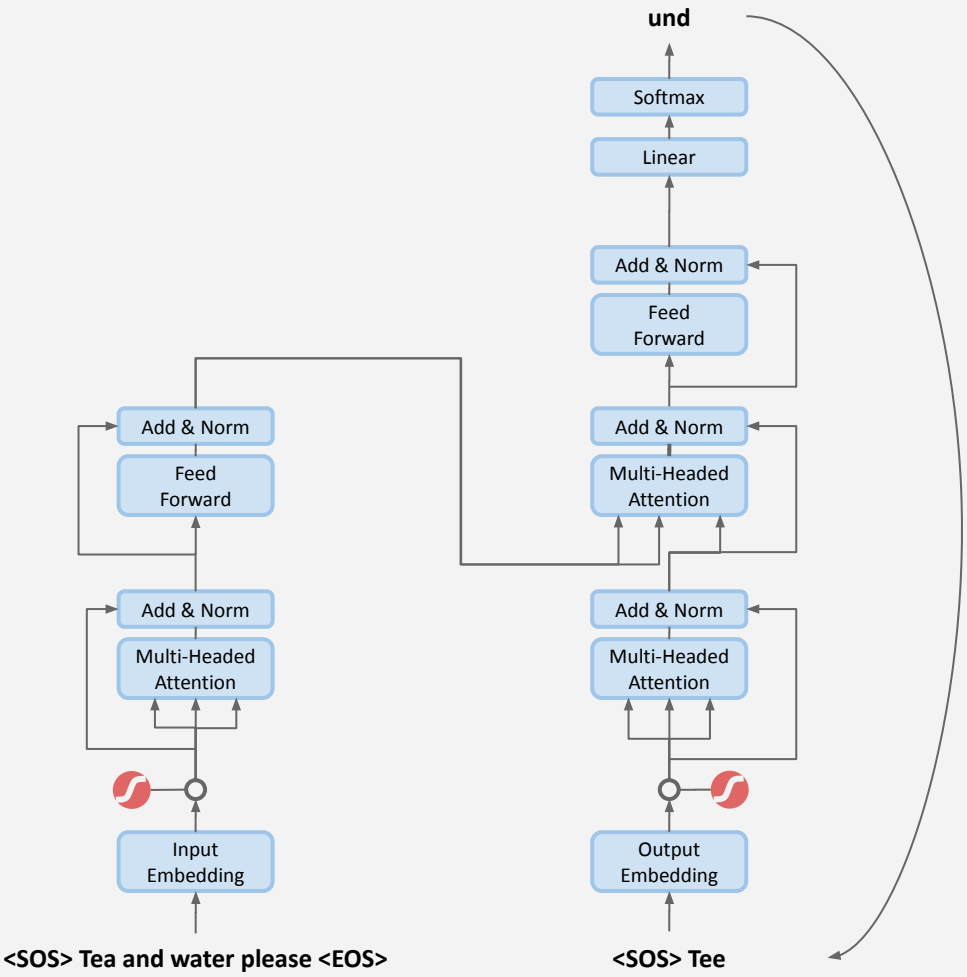
Transformer: Inference



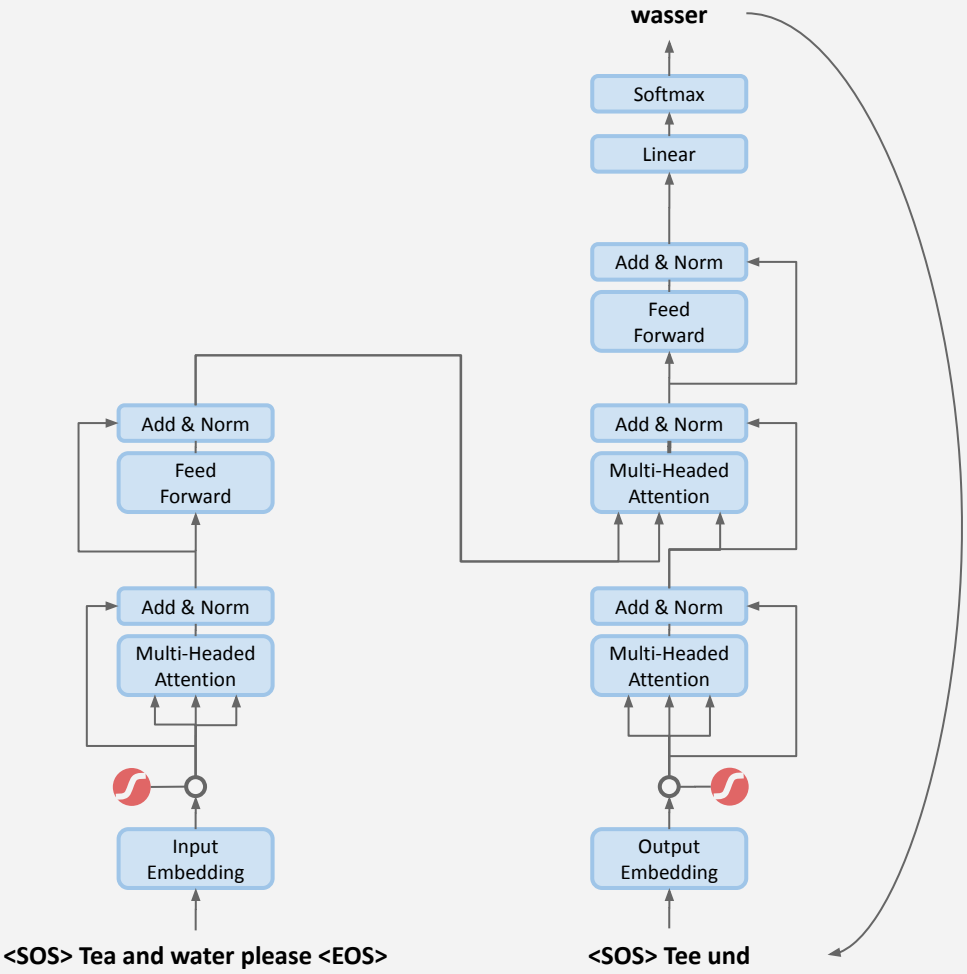
Transformer: Inference



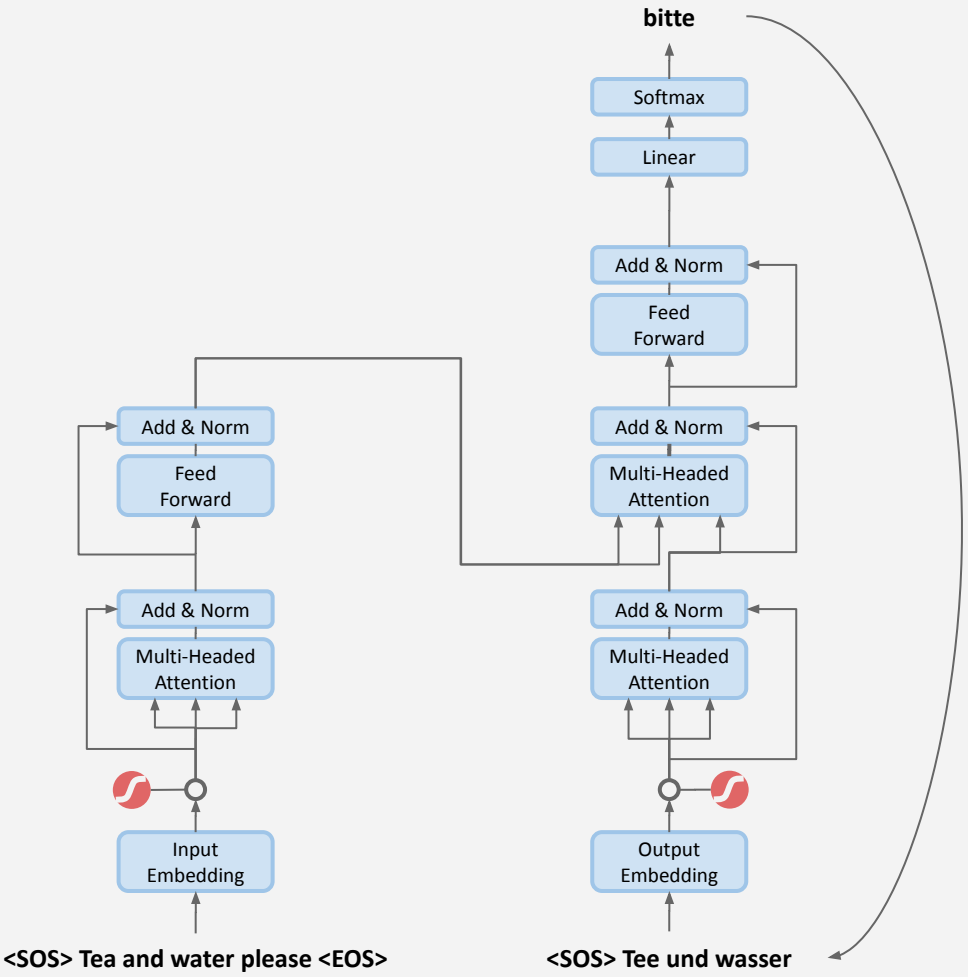
Transformer: Inference



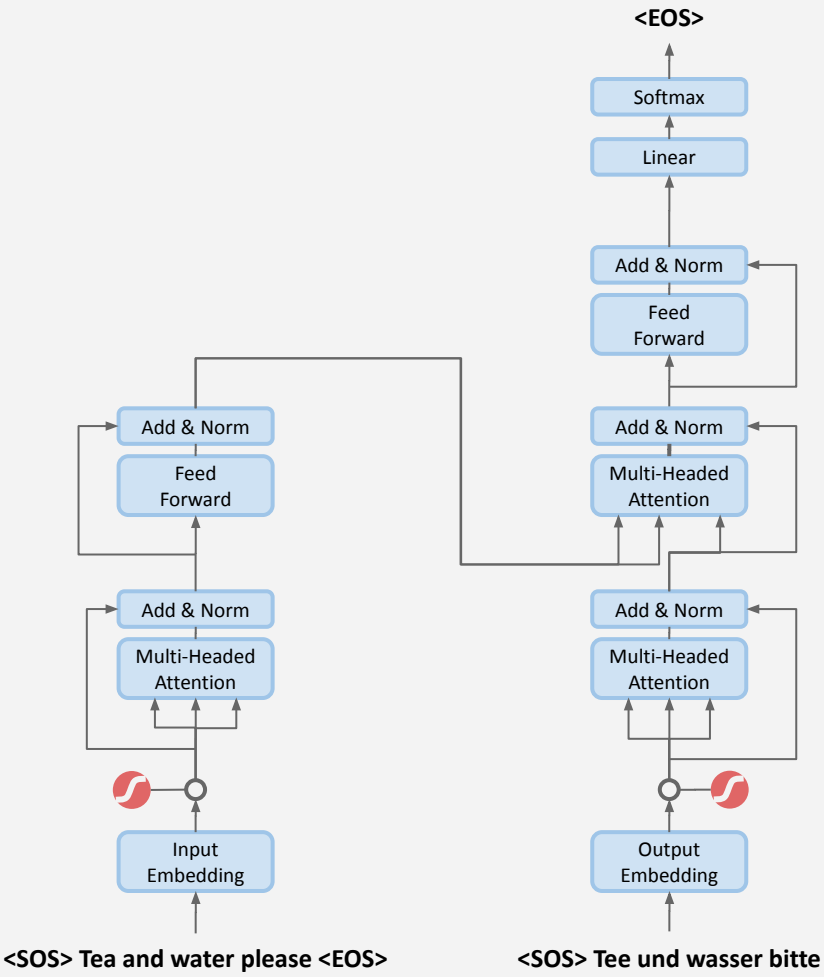
Transformer: Inference



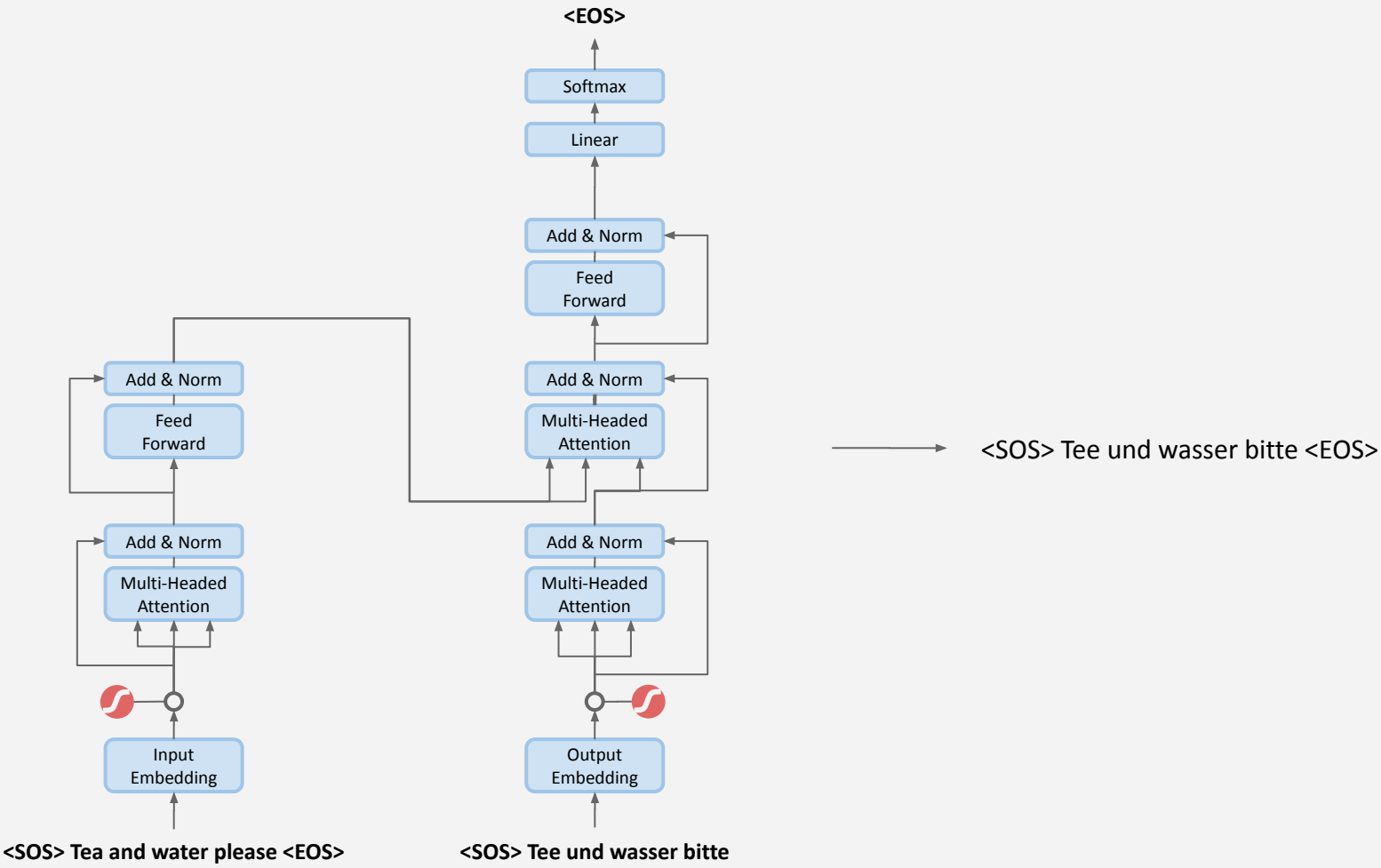
Transformer: Inference



Transformer: Inference



Transformer: Inference



Transformer: Inference Strategies

1. **Greedy**

   Select the token (word) with the highest probability

   + Easy to implement and efficient

2. **Beam Search**

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

   Maintains a tree of partially decoded sequences (**beam**)

   Select the sequence with the highest joint probability

   + Better exploration
   + High output quality

Attention and Transformers

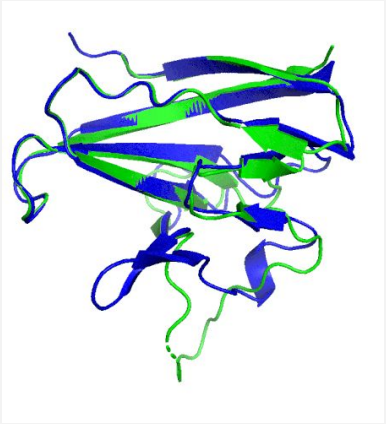
Language Models

An armchair in the shape of an avocado

GPT, LLaMA

Devlin et al., SAACL 2019

Protein Sequencing



AlphaFold2

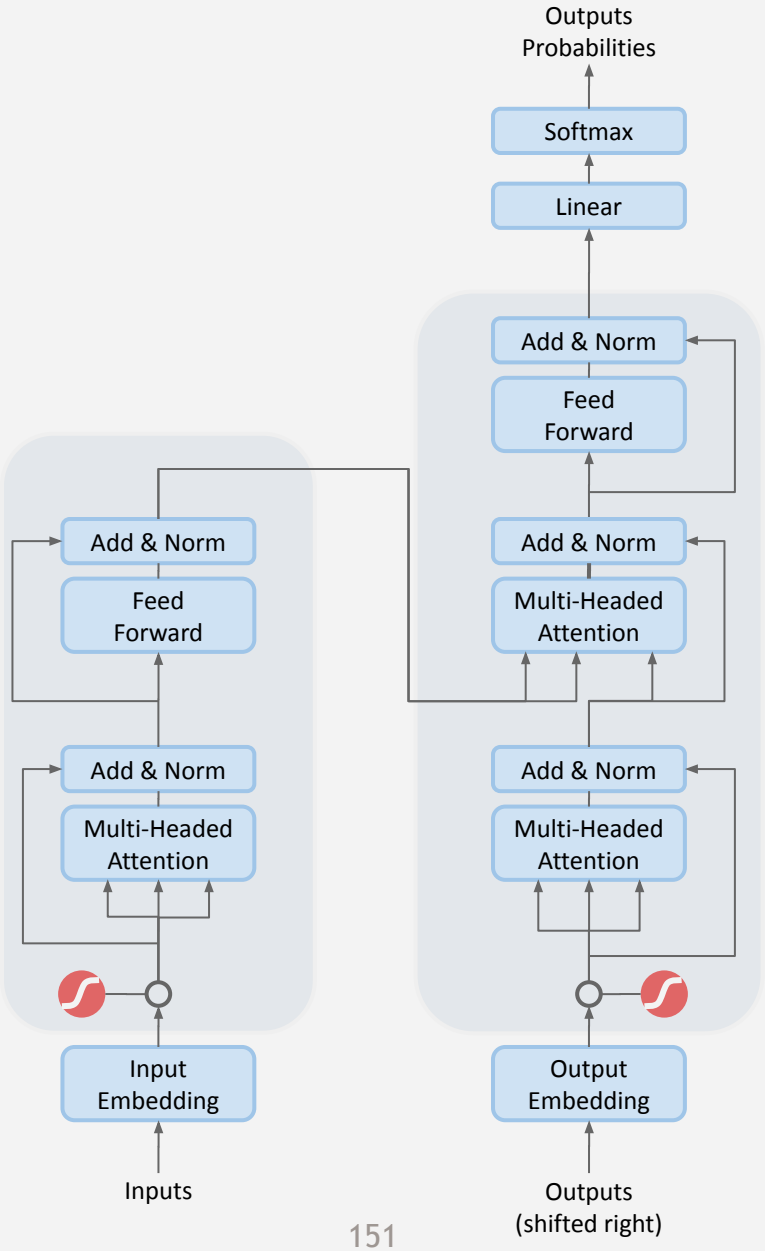
Jumper et al., Nature 2021

Computer Vision

Vision Transformer

Yang et al., NeurIPS 2021

Transformer



LSTM Cell