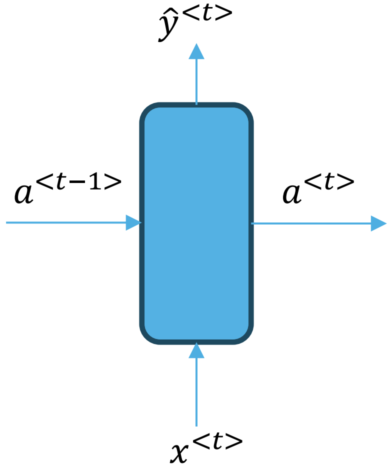
# Transformers

Antoni Kwiatek
Gradient Science Club 2025

# RNN & LSTM

# RNN



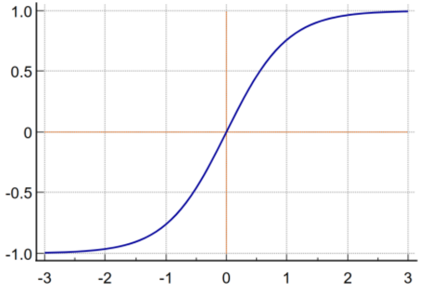
# RNN

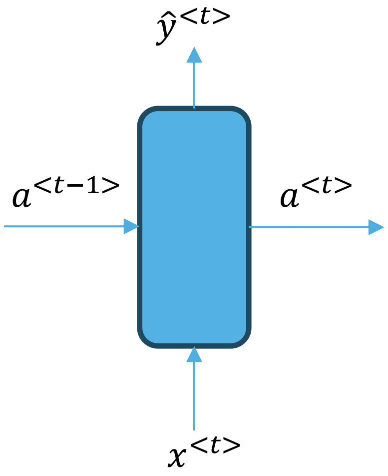
$$a^{<t>} = g\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(W_{ya}a^{<t>} + b_y\right)$$

$$Loss\left(\hat{y}^{<T_y>},\ y^{<T_y>}\right)$$

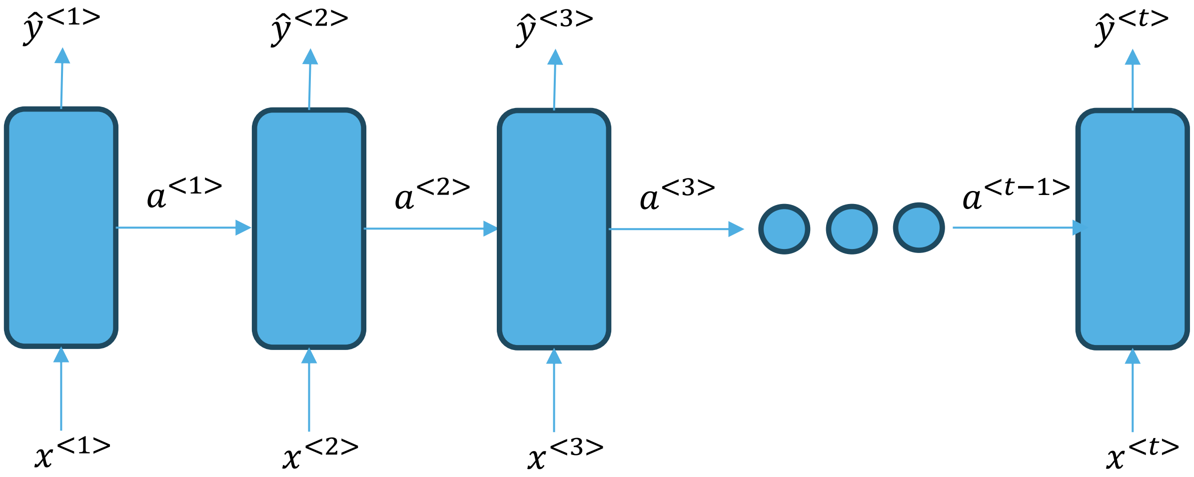
$$g(x) = \tanh(x)$$



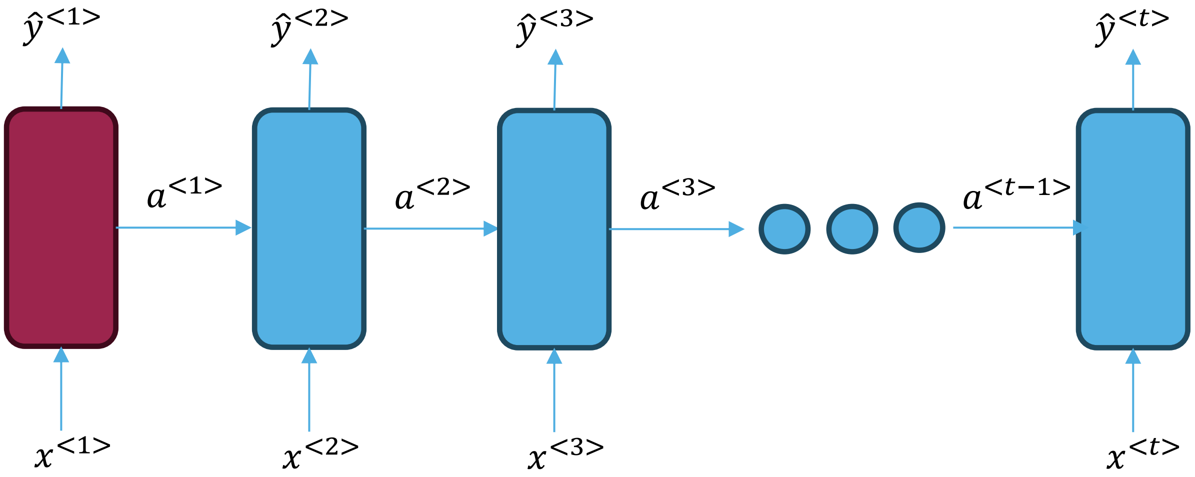




$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

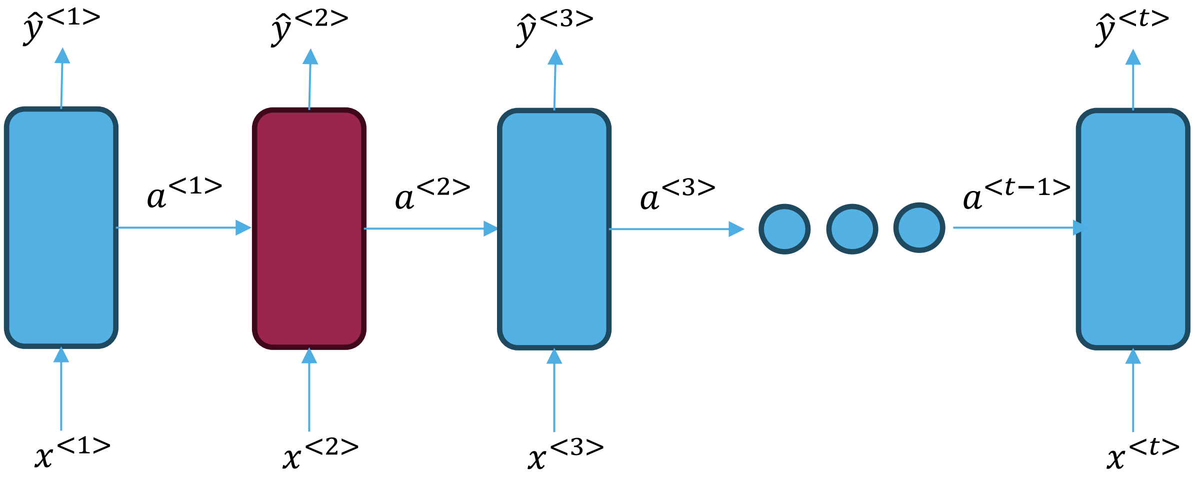
# RNN



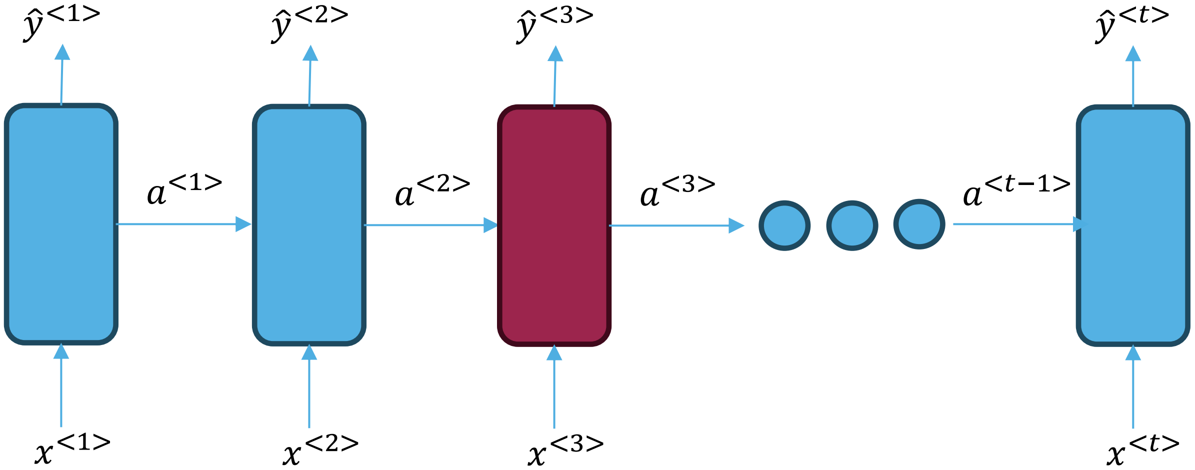
# RNN



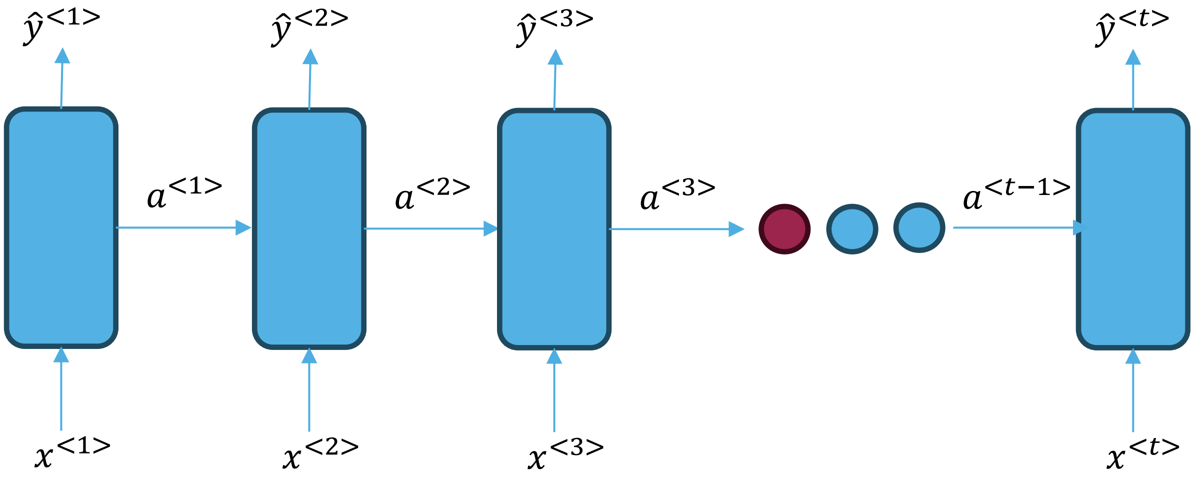
# RNN



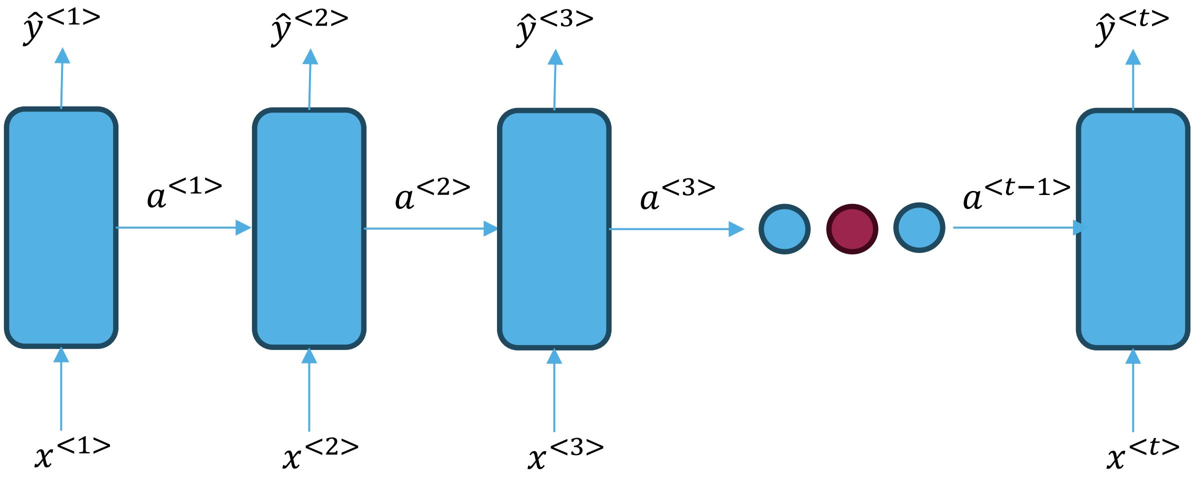
# RNN



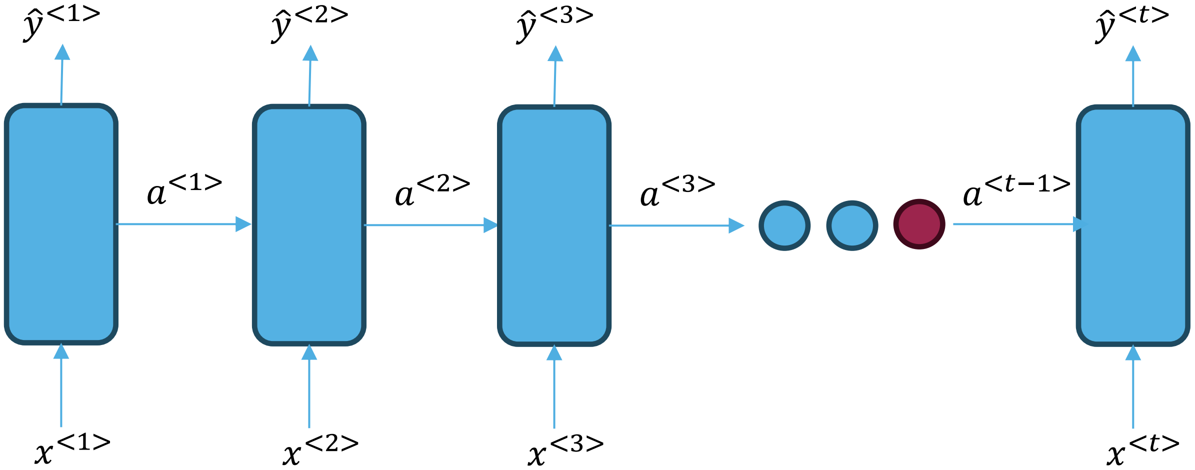
# RNN



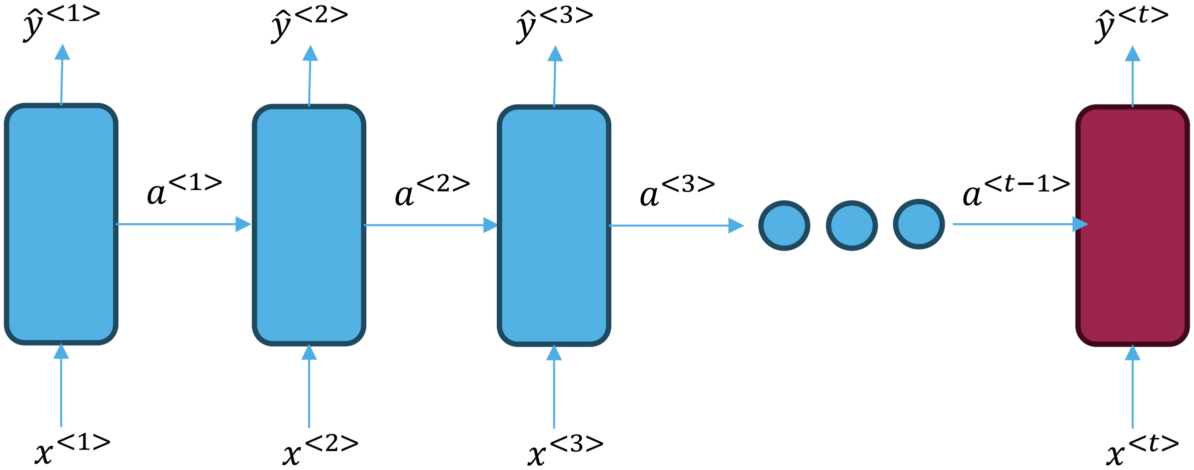
# RNN



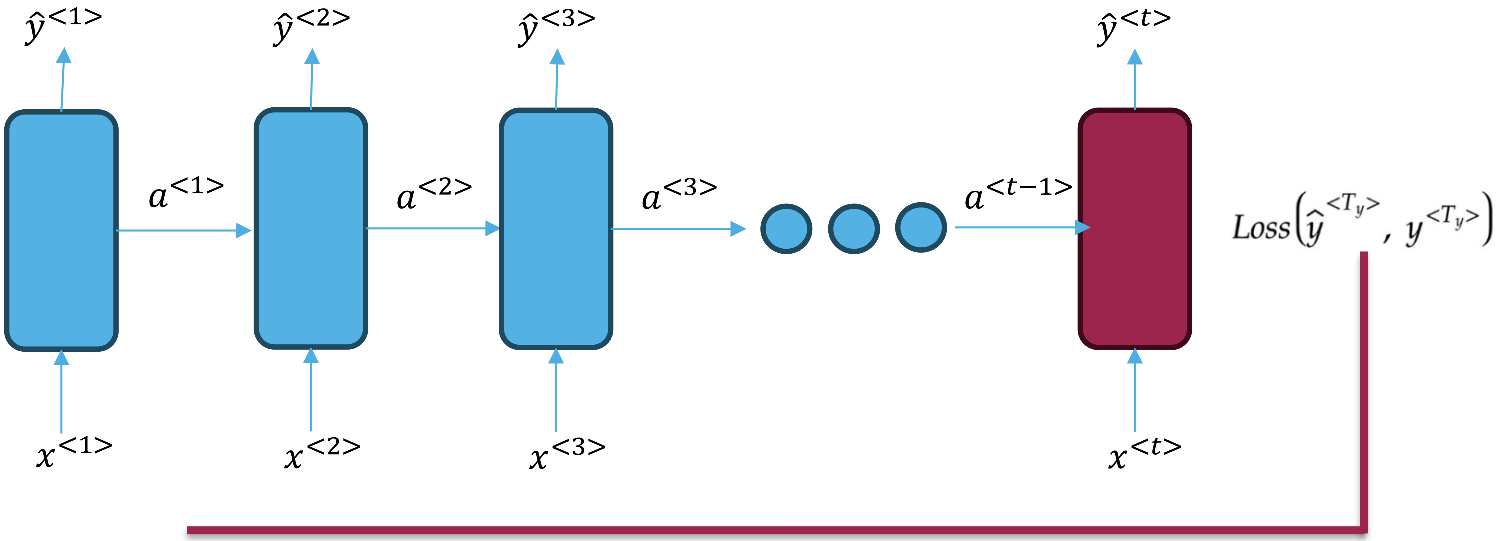
# RNN



# RNN

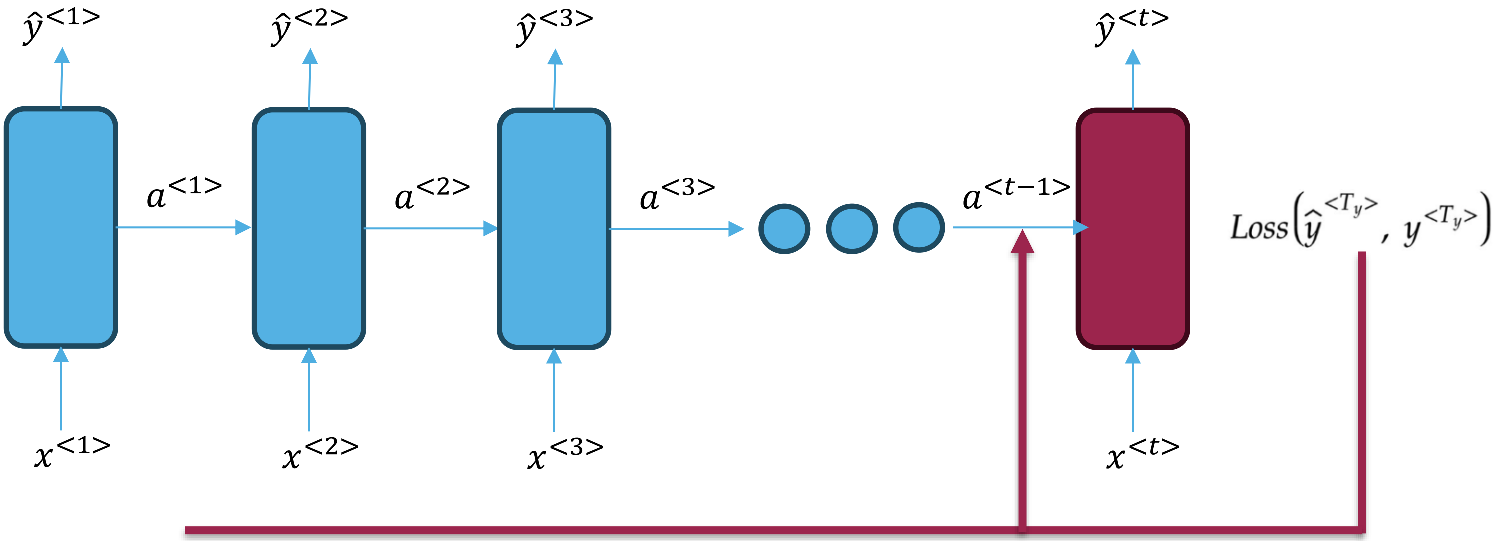


# RNN



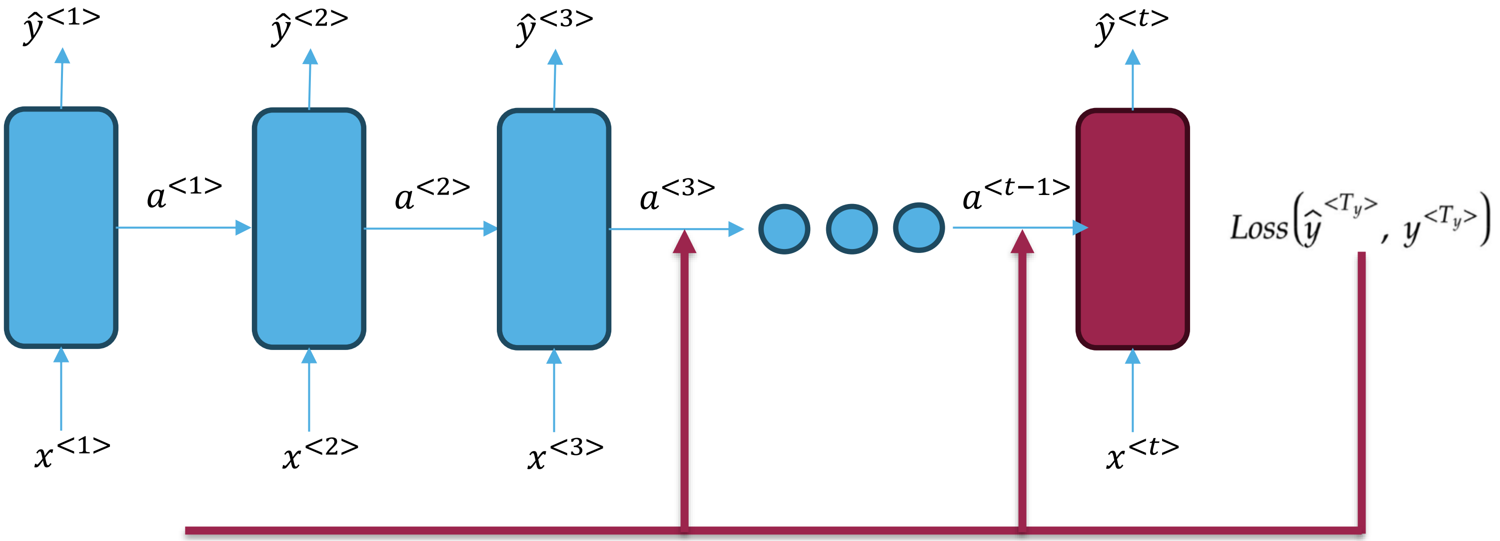

$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdot \dots \cdot \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

# RNN



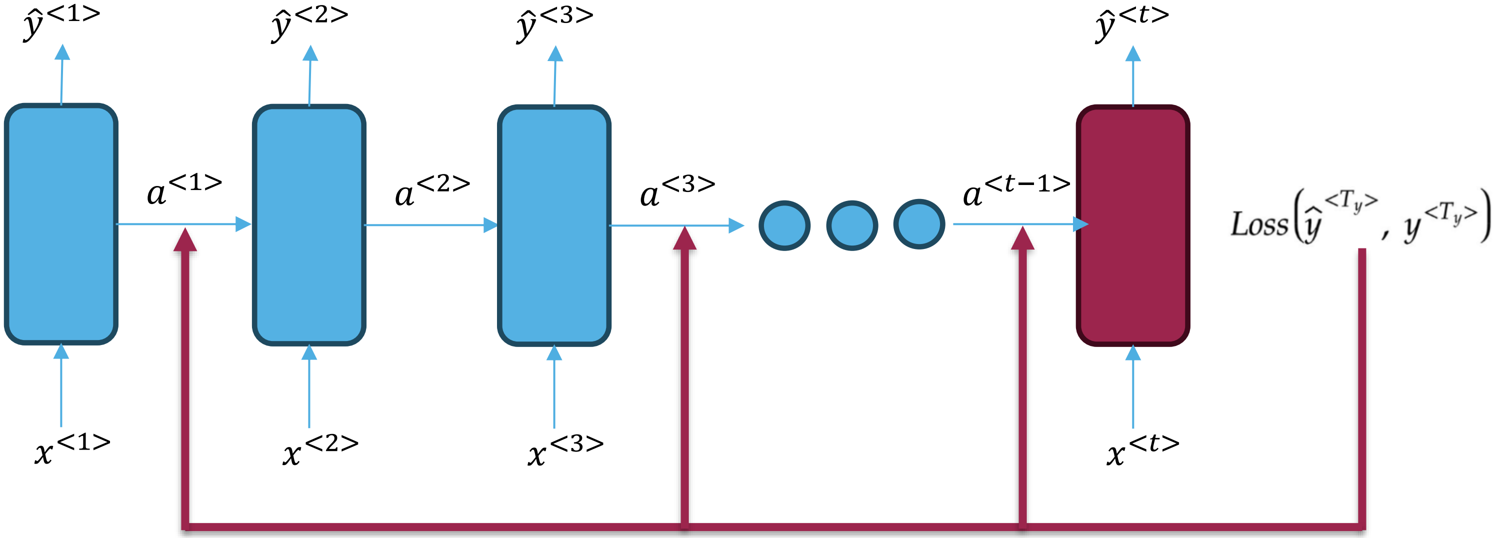

$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdot \cdots \cdot \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

# RNN




$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdot \dots \cdot \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

# RNN




$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdots\cdots \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

# RNN Problems

$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdot \cdots \cdot \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

# RNN Problems

$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdot \ldots \cdot \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

$$\frac{1}{5} * \frac{1}{5} * \frac{1}{5} * \frac{1}{5} * \frac{1}{5} * \frac{1}{5} * \cdots * \frac{1}{5} = 0.000\ldots$$

# RNN Problems

$$\frac{\partial L_T}{\partial h_1} = \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_3}{\partial h_2} \cdot \dots \cdot \frac{\partial h_T}{\partial h_{T-1}} \cdot \frac{\partial L_T}{\partial h_T}$$

Vanishing Gradient

Exploding Gradient

# LSTM

LSTM with solution for Vanishing and Exploding Gradient!

# LSTM

LSTM with solution for Vanishing and Exploding Gradient!

Still not enough…

# LSTM

LSTM with solution for Vanishing and Exploding Gradient!

## Still not enough…

Time complexity with backpropagation sucks

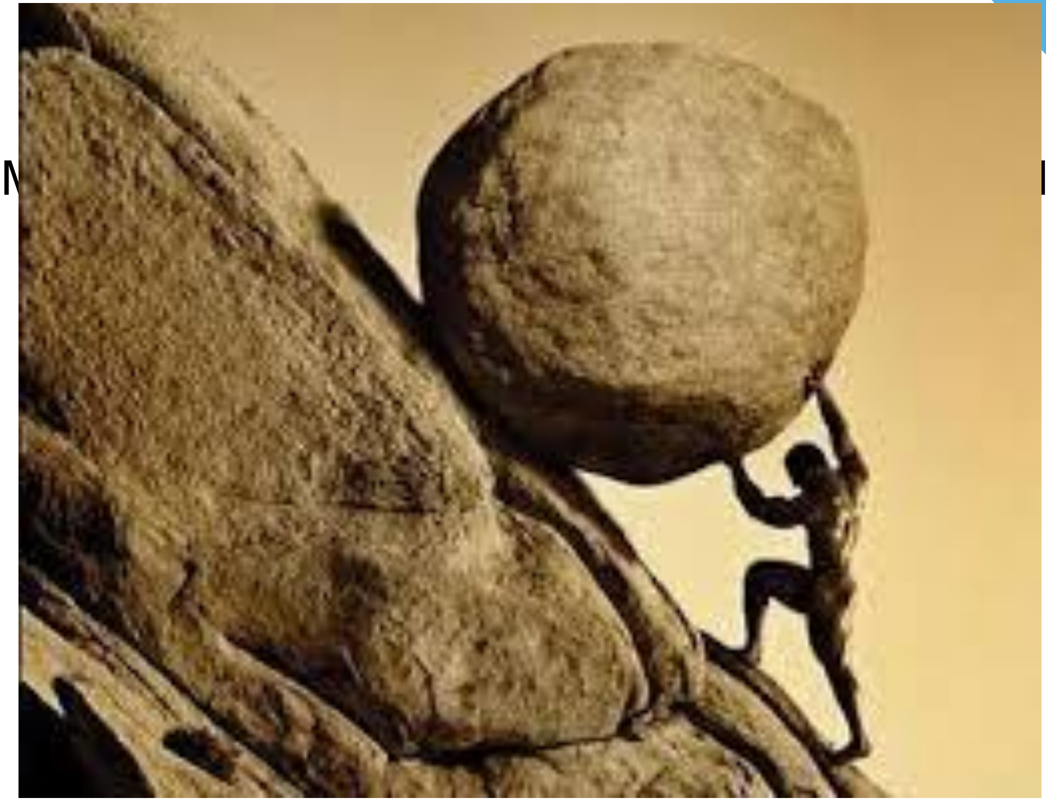
Sequential processing

Ineffective Transfer Learning

Difficult to train

# LSTM

LSTM lient!



# Attention Is All You Need

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

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

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

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

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

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

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

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



## Abstract

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

# Plan for Today

- RNN & LSTM roast
- *Attention is all you need*
- Deep dive into architecture
- Example workflow
- Live coding: Visualizing Attention (BERT example)
- Kahoot

# Deep dive into architecture

# Transformers architecture

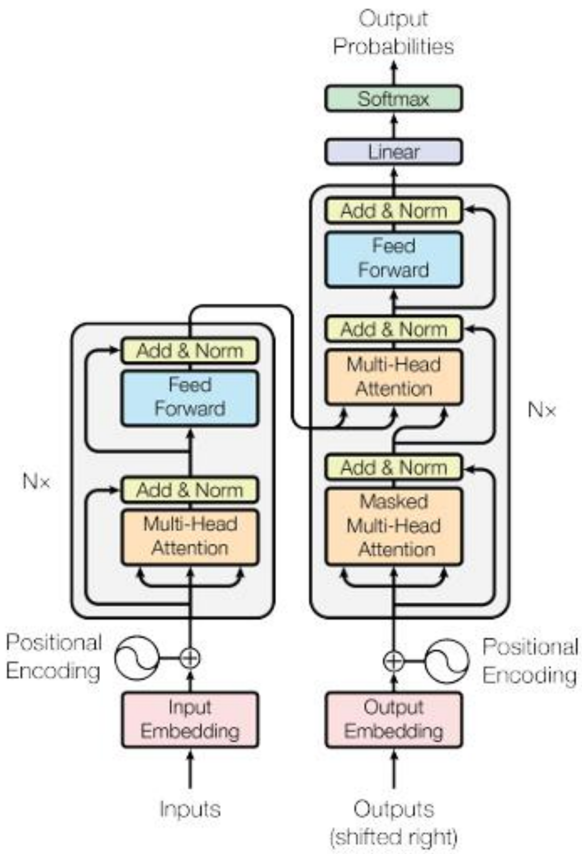



Figure 1: The Transformer - model architecture.

# Transformers architecture

## Input layer

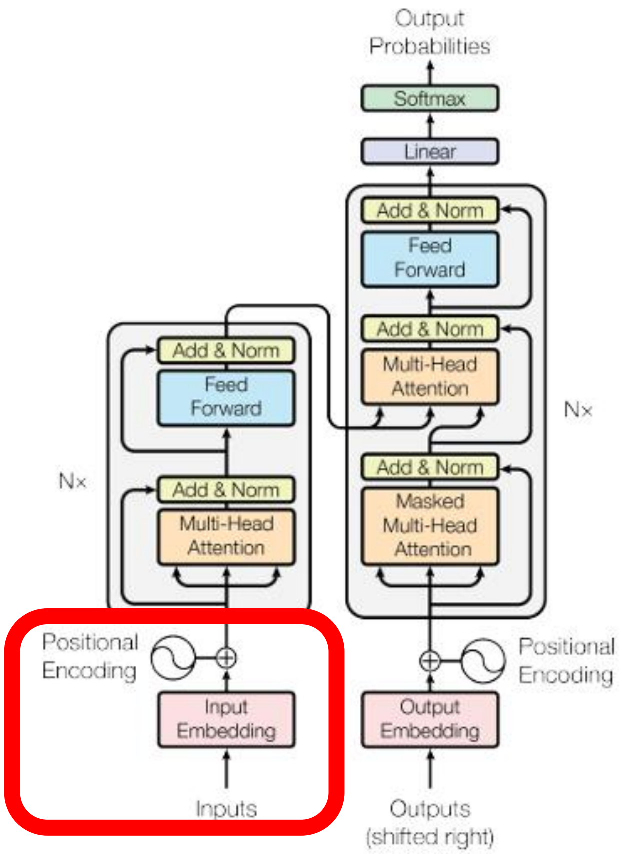


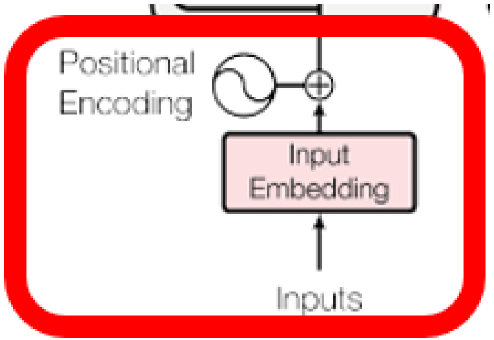

Figure 1: The Transformer - model architecture.

# Transformers architecture



Raw text

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



# Transformers architecture

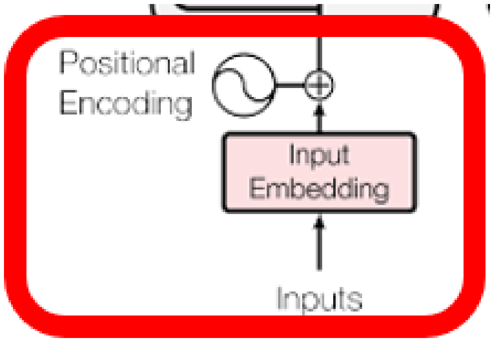
## Input layer

Raw text

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

Tokenization

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



# Transformers architecture

## Input layer

Raw text

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

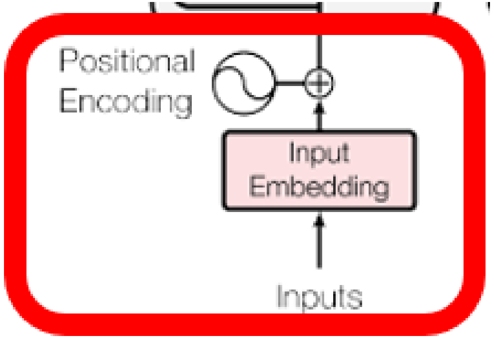
Tokenization

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

Tokens ID (position in vocabulary)

```
[976, 13983, 9289, 8088, 290, 12901, 2236, 480, 673, 3101, 25920, 13]
```

[Tokenizer - OpenAI API]

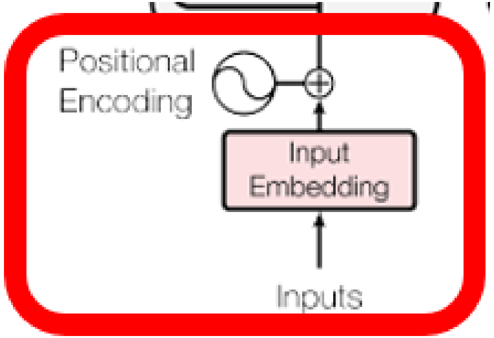


# Transformers architecture



Tokens ID (position in vocabulary)

```
[976, 13983, 9289, 8088, 290, 12901, 2236, 480, 673, 3101, 25920, 13]
```



# Transformers architecture

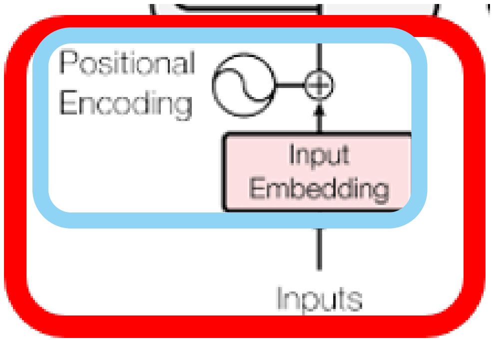
## Input layer

Tokens ID (position in vocabulary)

```
[976, 13983, 9289, 8088, 290, 12901, 2236, 480, 673, 3101, 25920, 13]
```

| | | | |
|---|---|---|---|
| 321.32 | 321.32 | 321.32 | 321.32 |
| 1235.5 | 1235.5 | 1235.5 | 1235.5 |
| 2356.6 | 2356.6 | 2356.6 | 2356.6 |
| 0.232 | 0.232 | 0.232 | 0.232 |
| ......... | ......... | ......... | ......... |
| 123.56 | 123.56 | 123.56 | 123.56 |
| 3456.0 | 3456.0 | 3456.0 | 3456.0 |
| 45.654 | 45.654 | 45.654 | 45.654 |
| 1239.0 | 1239.0 | 1239.0 | 1239.0 |
| 54.222 | 54.222 | 54.222 | 54.222 |

Embeddings (Vector of size $d_{model} = 512$)



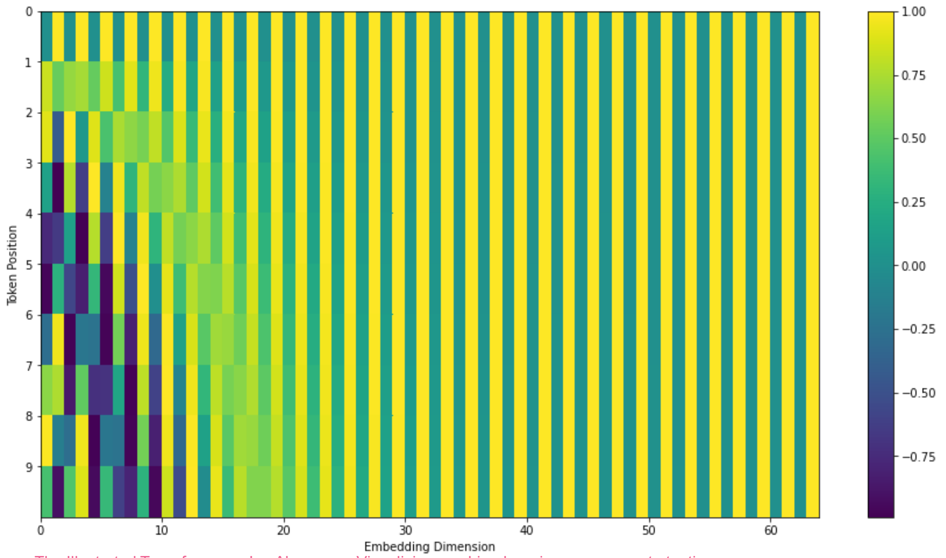
# Transformers architecture

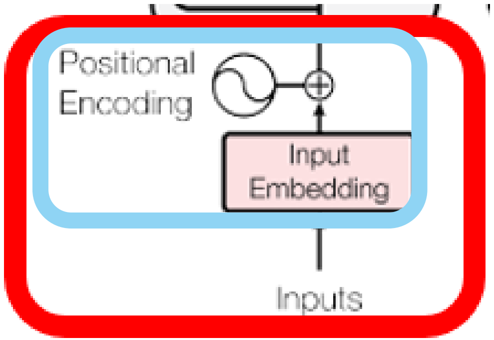


Positional Encoding – how the model knows where each word is

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

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

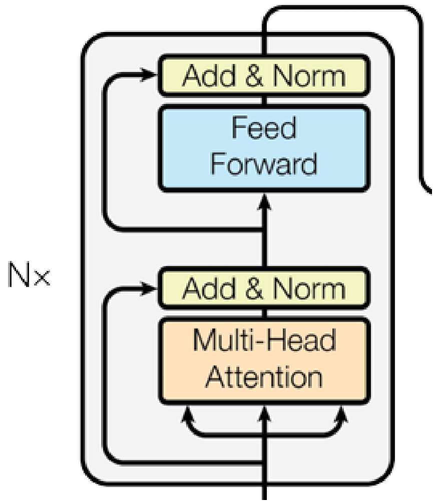



The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time.

# Transformers architecture

## Encoder Layer



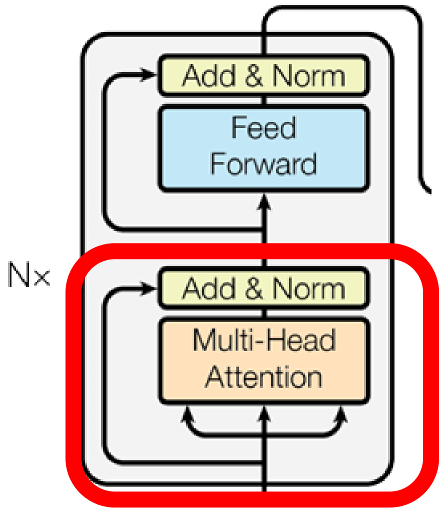
# Transformers architecture

Encoder Layer

Multi-Head Attention

Before Multi-Head Attention, there is *Self-attention*

Self-attention allows the model to relate words to each other.



# Transformers architecture

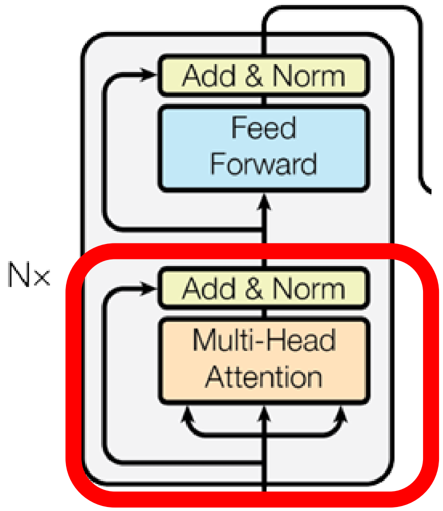
## Encoder Layer

Multi-Head Attention

Before Multi-Head Attention, there is Self-attention

Self-attention allows the model to relate words to each other.

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



# Transformers architecture

## Encoder Layer

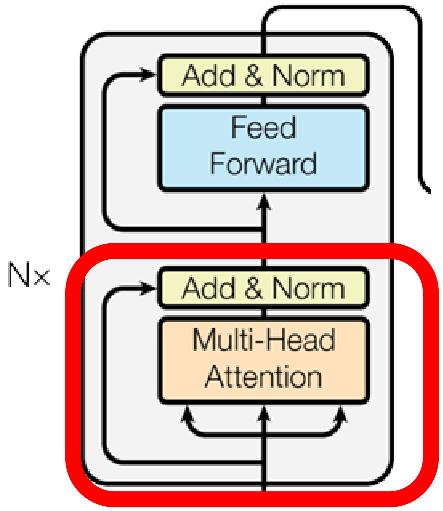
### Multi-Head Attention

Before Multi-Head Attention, there is Self-attention

Self-attention allows the model to relate words to each other.

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

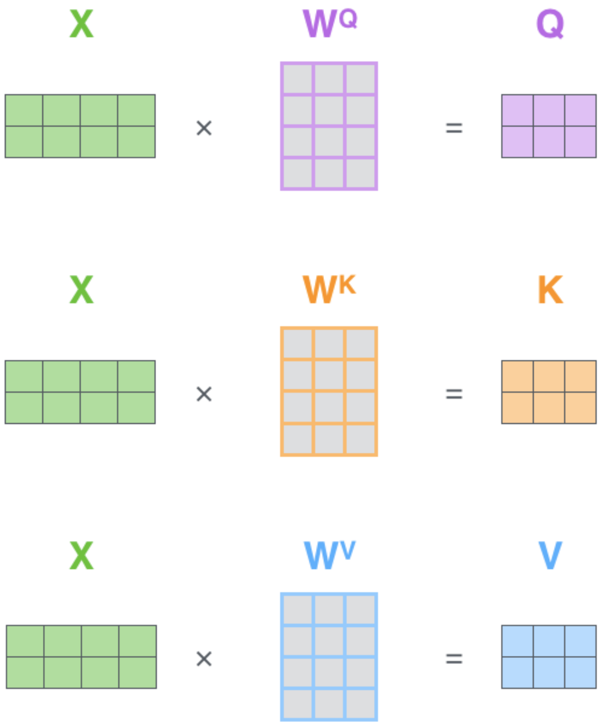
*Q seeks a match in K to retrieve content from V*



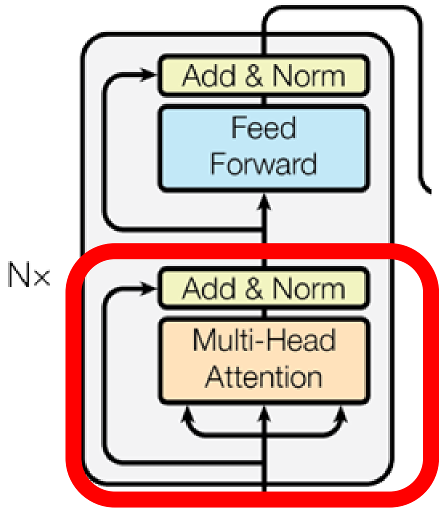
# Transformers architecture

## Encoder Layer

### Multi-Head Attention




The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time.



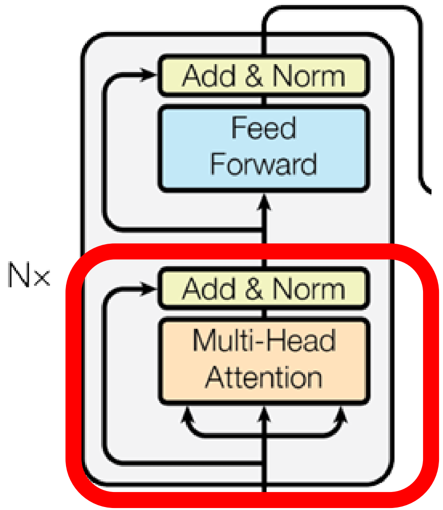
# Transformers architecture

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

## Encoder Layer

### Multi-Head Attention

$$\text{Softmax}\left(\frac{Q \times K^T}{\sqrt{d_{model}}}\right)$$



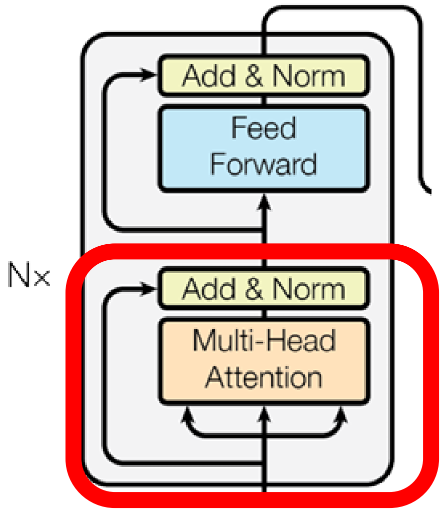
# Transformers architecture

## Encoder Layer

### Multi-Head Attention

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

| | The | animal | didn't | cross | the | street | because | it | was | too | tired | . |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| The | 0.07 | 0.12 | 0.10 | 0.09 | 0.06 | 0.06 | 0.05 | 0.11 | 0.09 | 0.10 | 0.05 | 0.12 |
| animal | 0.13 | 0.07 | 0.07 | 0.07 | 0.08 | 0.10 | 0.09 | 0.08 | 0.10 | 0.07 | 0.08 | 0.08 |
| didn't | 0.07 | 0.10 | 0.06 | 0.08 | 0.09 | 0.05 | 0.09 | 0.06 | 0.05 | 0.12 | 0.12 | 0.11 |
| cross | 0.07 | 0.06 | 0.11 | 0.08 | 0.06 | 0.09 | 0.06 | 0.13 | 0.07 | 0.10 | 0.07 | 0.09 |
| the | 0.08 | 0.06 | 0.12 | 0.10 | 0.12 | 0.11 | 0.08 | 0.12 | 0.05 | 0.06 | 0.05 | 0.06 |
| street | 0.07 | 0.07 | 0.11 | 0.07 | 0.07 | 0.09 | 0.06 | 0.11 | 0.05 | 0.13 | 0.11 | 0.06 |
| because | 0.05 | 0.11 | 0.10 | 0.10 | 0.11 | 0.05 | 0.07 | 0.06 | 0.12 | 0.09 | 0.07 | 0.05 |
| it | 0.03 | 0.62 | 0.04 | 0.04 | 0.05 | 0.03 | 0.02 | 0.04 | 0.04 | 0.03 | 0.04 | 0.03 |
| was | 0.09 | 0.08 | 0.05 | 0.06 | 0.06 | 0.10 | 0.07 | 0.09 | 0.13 | 0.07 | 0.08 | 0.11 |
| too | 0.06 | 0.05 | 0.06 | 0.05 | 0.12 | 0.10 | 0.09 | 0.11 | 0.10 | 0.06 | 0.11 | 0.08 |
| tired | 0.11 | 0.12 | 0.07 | 0.06 | 0.06 | 0.08 | 0.11 | 0.12 | 0.05 | 0.08 | 0.08 | 0.06 |
| . | 0.05 | 0.07 | 0.12 | 0.07 | 0.08 | 0.10 | 0.07 | 0.12 | 0.12 | 0.06 | 0.08 | 0.06 |



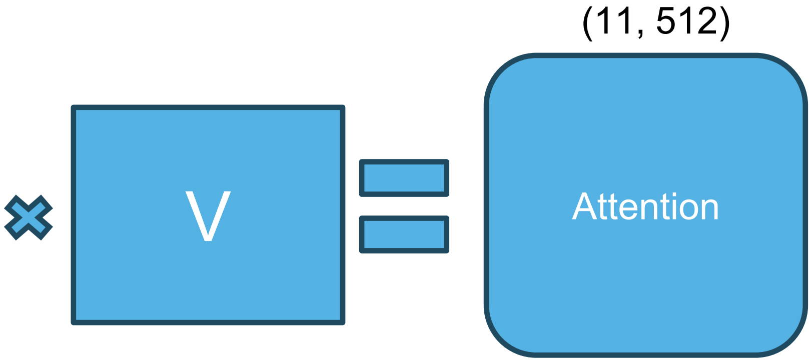
# Transformers architecture

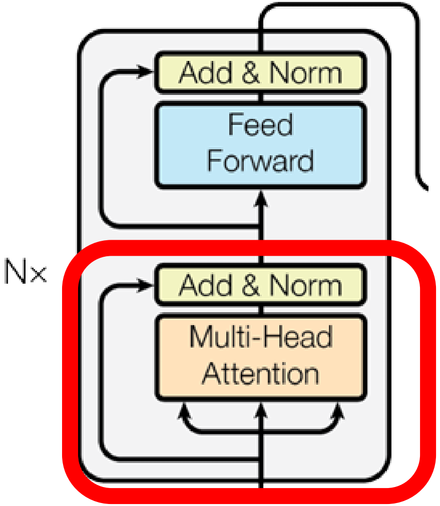
## Encoder Layer

### Multi-Head Attention

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

| | The | animal | didn't | cross | the | street | because | it | was | too | tired | . |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| The | 0.07 | 0.12 | 0.10 | 0.09 | 0.06 | 0.06 | 0.05 | 0.11 | 0.09 | 0.10 | 0.05 | 0.12 |
| animal | 0.13 | 0.07 | 0.07 | 0.07 | 0.08 | 0.10 | 0.09 | 0.08 | 0.10 | 0.07 | 0.08 | 0.08 |
| didn't | 0.07 | 0.10 | 0.06 | 0.08 | 0.09 | 0.05 | 0.09 | 0.06 | 0.05 | 0.12 | 0.12 | 0.11 |
| cross | 0.07 | 0.06 | 0.11 | 0.08 | 0.06 | 0.09 | 0.06 | 0.13 | 0.07 | 0.10 | 0.07 | 0.09 |
| the | 0.08 | 0.06 | 0.12 | 0.10 | 0.12 | 0.11 | 0.08 | 0.12 | 0.05 | 0.06 | 0.05 | 0.06 |
| street | 0.07 | 0.07 | 0.11 | 0.07 | 0.07 | 0.09 | 0.06 | 0.11 | 0.05 | 0.13 | 0.11 | 0.06 |
| because | 0.05 | 0.11 | 0.10 | 0.10 | 0.11 | 0.05 | 0.07 | 0.06 | 0.12 | 0.09 | 0.07 | 0.05 |
| it | 0.03 | 0.62 | 0.04 | 0.04 | 0.05 | 0.03 | 0.02 | 0.04 | 0.04 | 0.03 | 0.04 | 0.03 |
| was | 0.09 | 0.08 | 0.05 | 0.06 | 0.06 | 0.10 | 0.07 | 0.09 | 0.13 | 0.07 | 0.08 | 0.11 |
| too | 0.06 | 0.05 | 0.06 | 0.05 | 0.12 | 0.10 | 0.09 | 0.11 | 0.10 | 0.06 | 0.11 | 0.08 |
| tired | 0.11 | 0.12 | 0.07 | 0.06 | 0.06 | 0.08 | 0.11 | 0.12 | 0.05 | 0.08 | 0.08 | 0.06 |
| . | 0.05 | 0.07 | 0.12 | 0.07 | 0.08 | 0.10 | 0.07 | 0.12 | 0.12 | 0.06 | 0.08 | 0.06 |



# Transformers architecture

## Encoder Layer

### Multi-Head Attention

1) This is our input sentence*

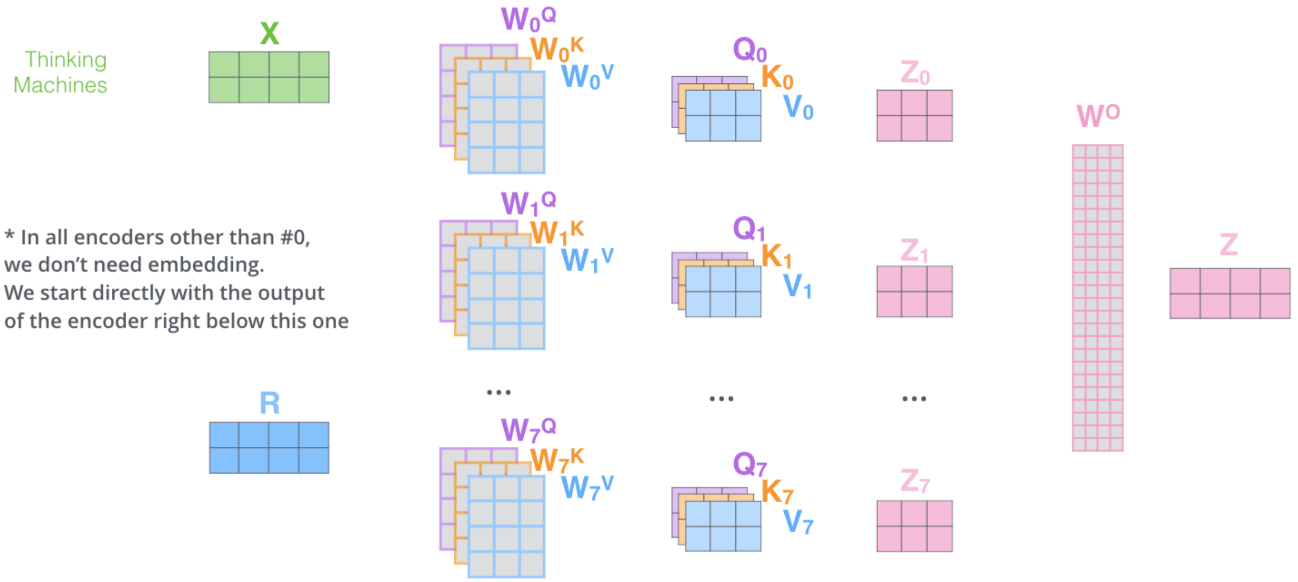
2) We embed each word*

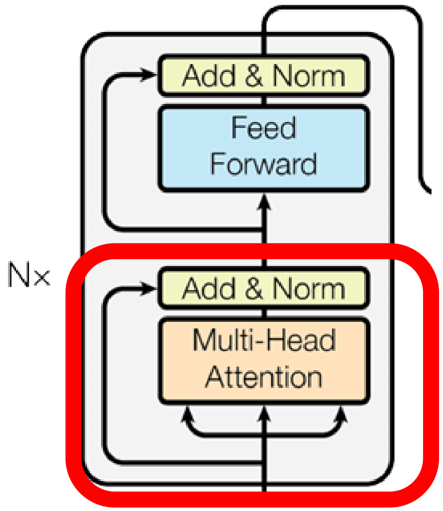
3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix $W^O$ to produce the output of the layer

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



# Transformers architecture

## Encoder Layer

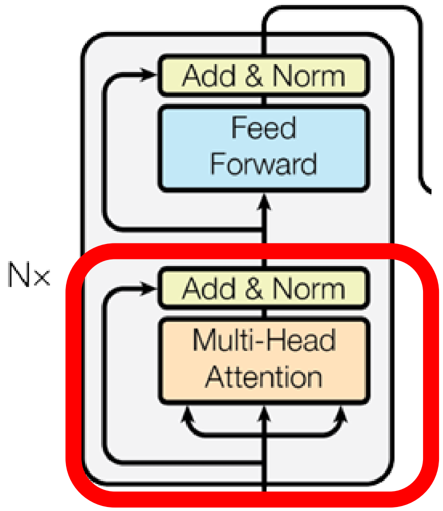
Multi-Head Attention

Add & Norm

Residual Connection

Layer Normalization

$$Output = Processed_{data} + Input$$



# Transformers architecture

## Encoder Layer

Multi-Head Attention

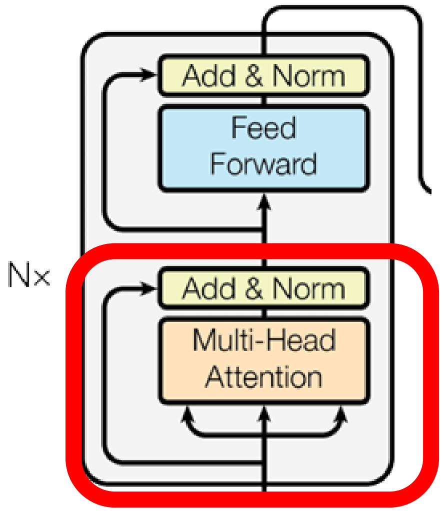
## Add & Norm

Residual Connection

Layer Normalization

$$z = x + F(x)$$

$$y = LayerNorm(z)$$



# Transformers architecture

## Encoder Layer

Multi-Head Attention

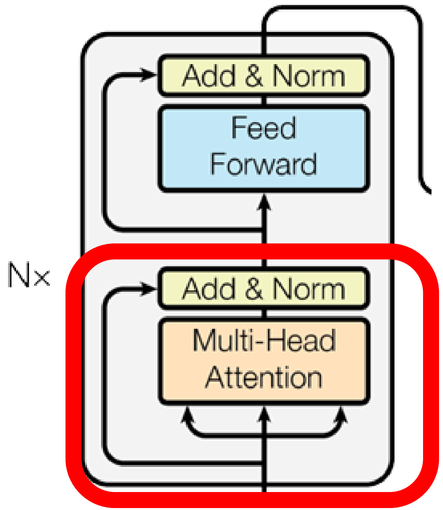
### Add & Norm

Residual Connection

Layer Normalization

$$z = x + F(x)$$

$$y = LayerNorm(z)$$



# Transformers architecture

## Encoder Layer

Multi-Head Attention

## Add & Norm

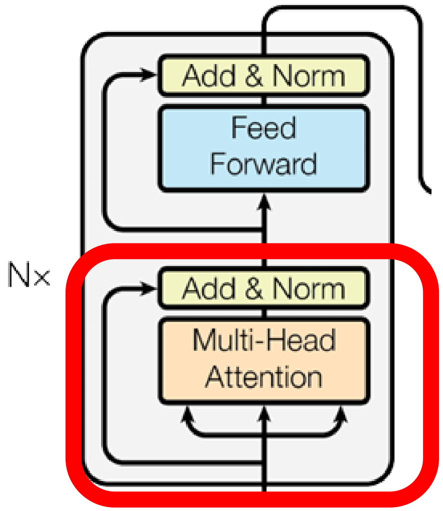
Residual Connection

Layer Normalization

$$z = x + F(x)$$

$$y = x + F(LayerNorm(x))$$

modern models



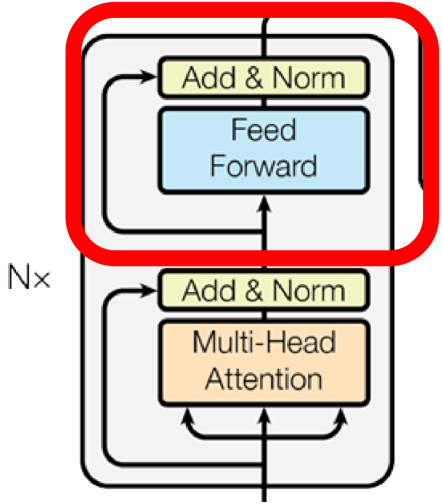
# Transformers architecture

## Encoder Layer

Feed-Forward

## Feed-Forward

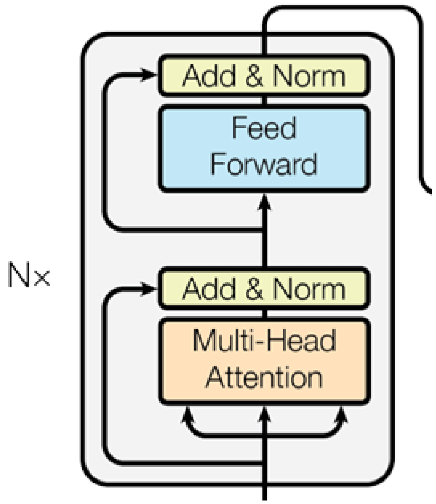
$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



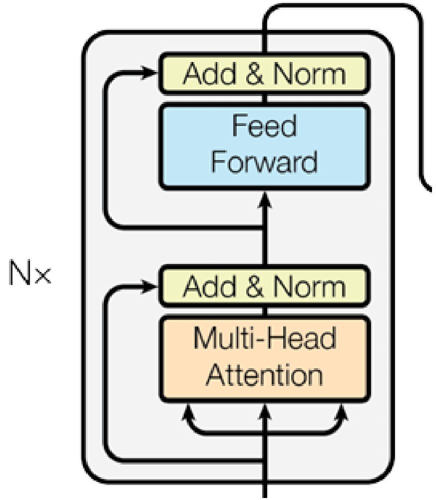
# Transformers architecture

## Encoder Layer

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

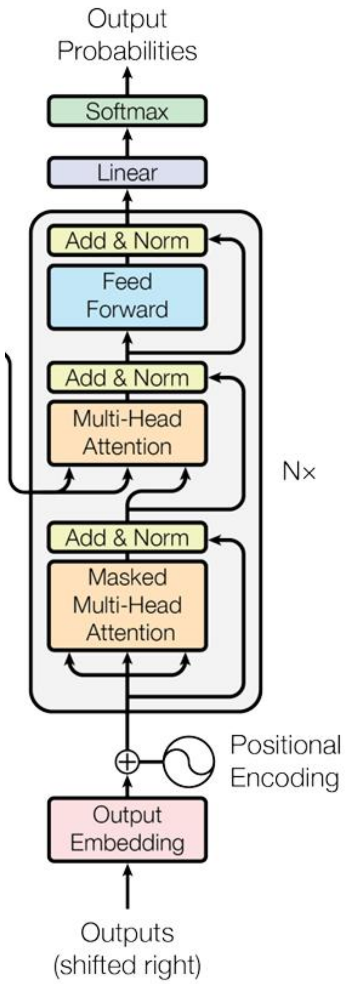


# Encoder layer - where does it go?



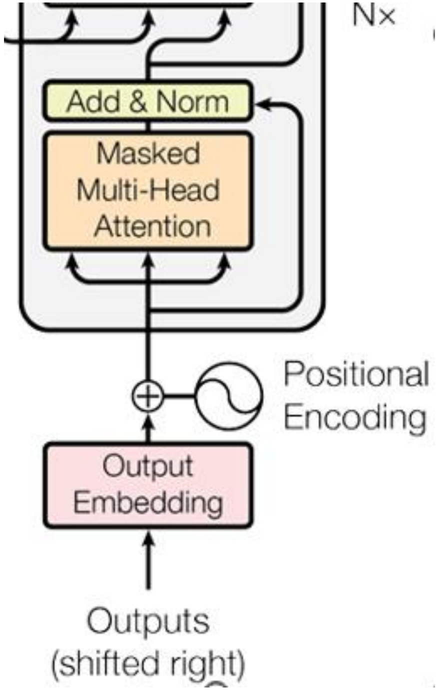
# Transformers architecture

## Decoder Layer



# Transformers architecture

## Decoder Layer

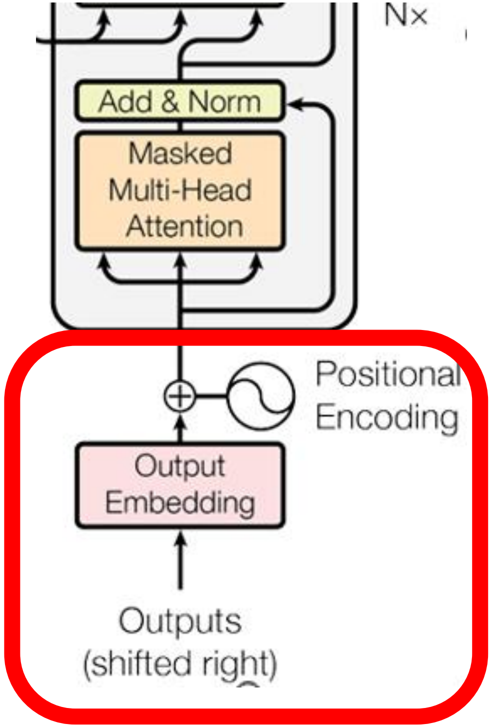


# Transformers architecture

## Decoder Layer

Shifted right – what does it mean?

Another input and same process?



# Transformers architecture

## Decoder Layer

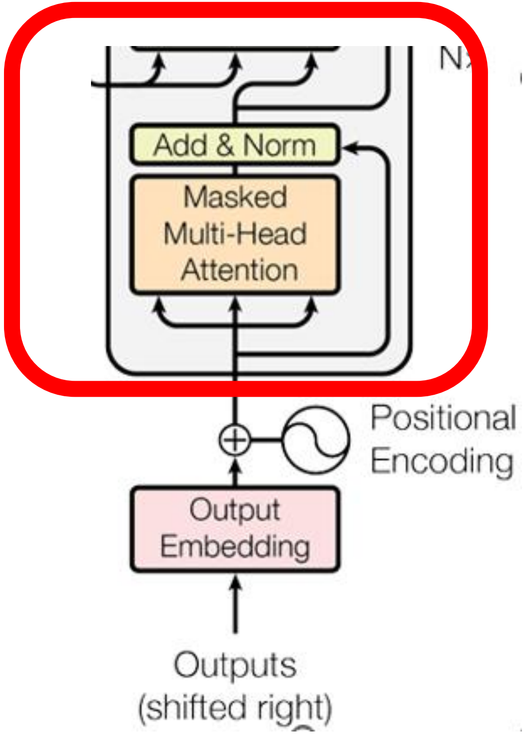
Masked-Attention

Masked Multi-Head Attention

Self-attention

Masked

Multi-head



# Transformers architecture

## Decoder Layer

Masked-Attention

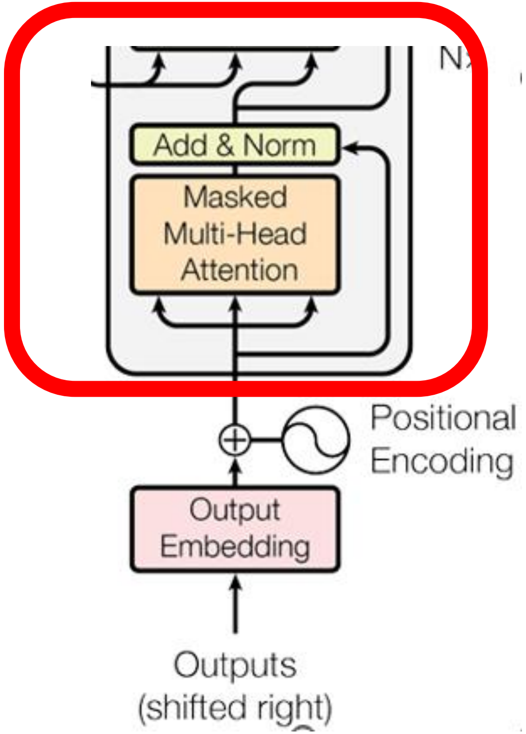
Masked Multi-Head Attention

Self-attention

Masked

Multi-head

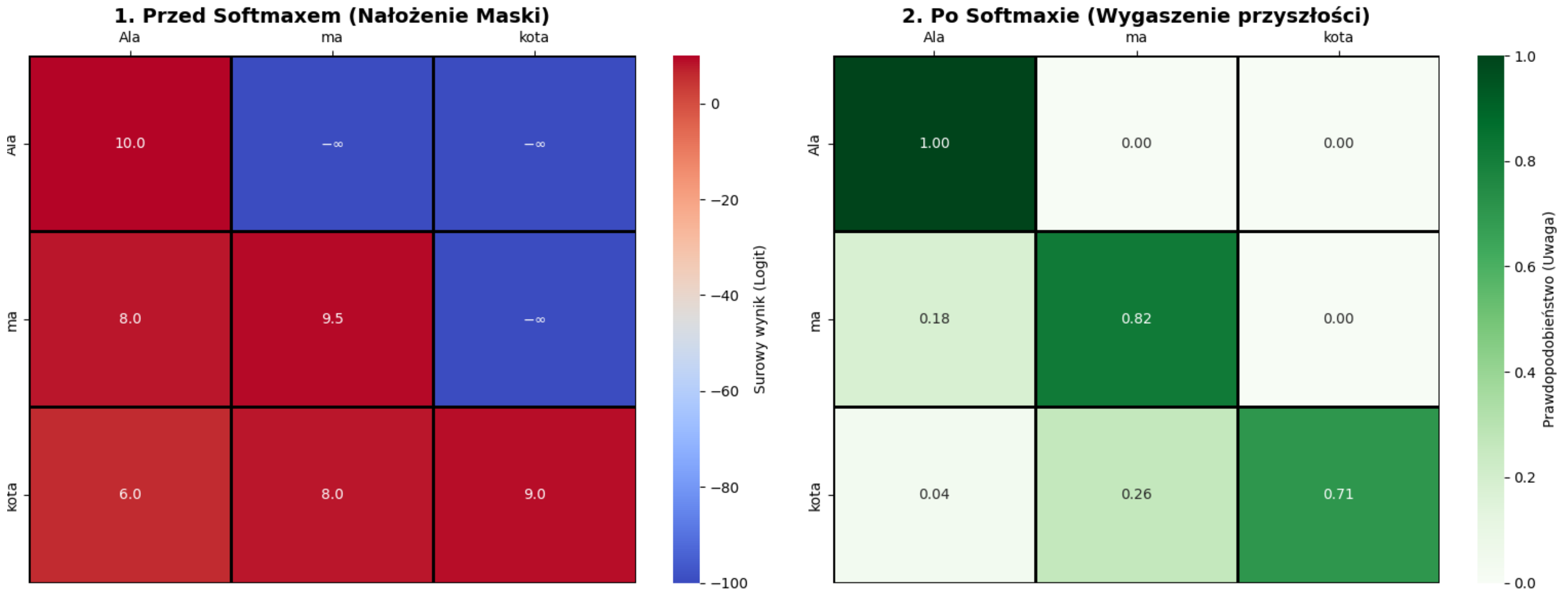
Output at a certain position can only depend on the words on the previous positions



# Transformers architecture

## Decoder Layer

### Masked-Attention



# Transformers architecture

## Decoder Layer

Encoder layer - where does it go?

V K

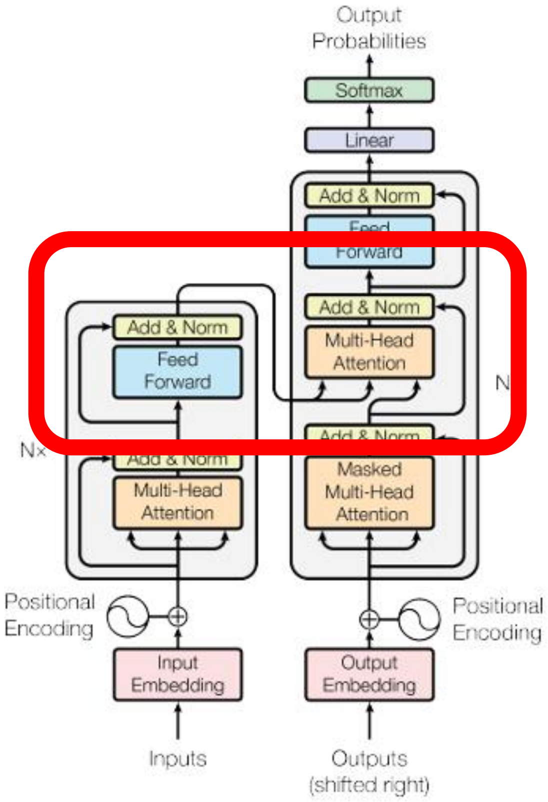



Figure 1: The Transformer - model architecture.

# Transformers architecture

## Decoder Layer

Encoder layer - where does it go?

V

K

Q $= Vector\ from\ Masked\ Multi - Head\ Attention$

*Current state of output sentence

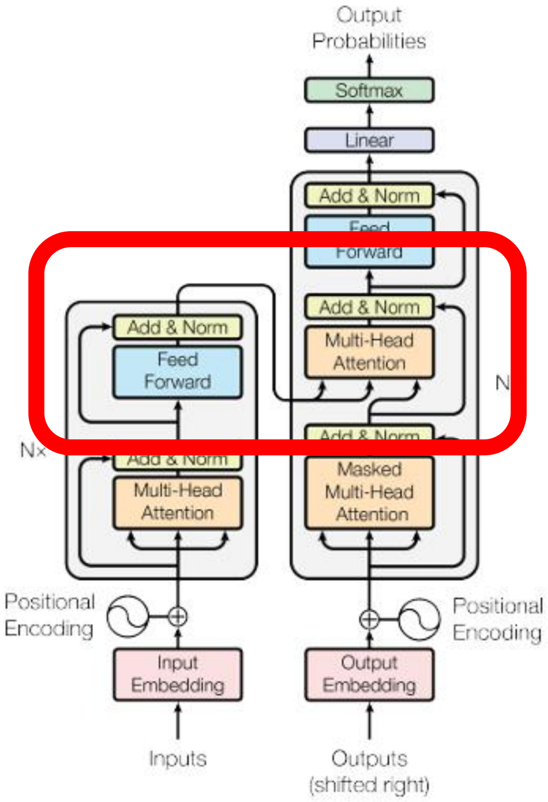



Figure 1: The Transformer - model architecture.

# Transformers architecture

## Decoder Layer

Encoder layer - where does it go?

V

K

Q

$= Vector\ from\ Masked\ Multi-Head\ Attention$

*Current state of output sentence

*Q seeks a match in K to retrieve content from V*

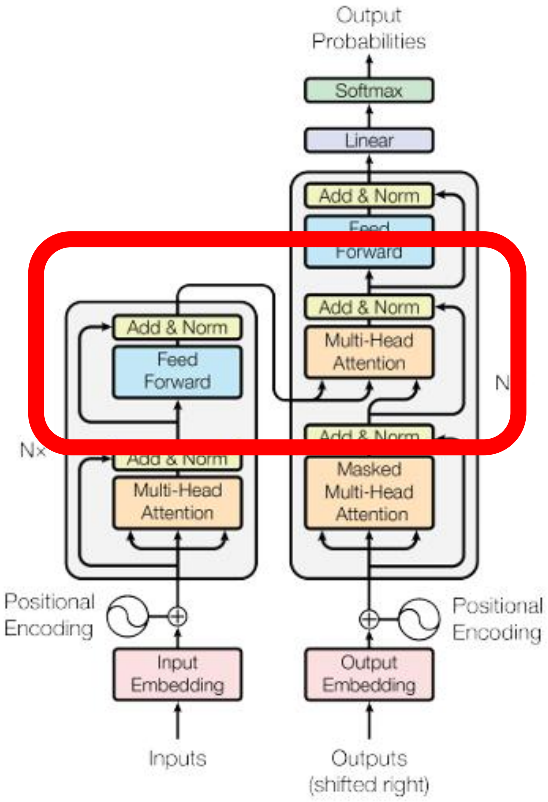



Figure 1: The Transformer - model architecture.

# Transformers architecture

## Decoder Layer

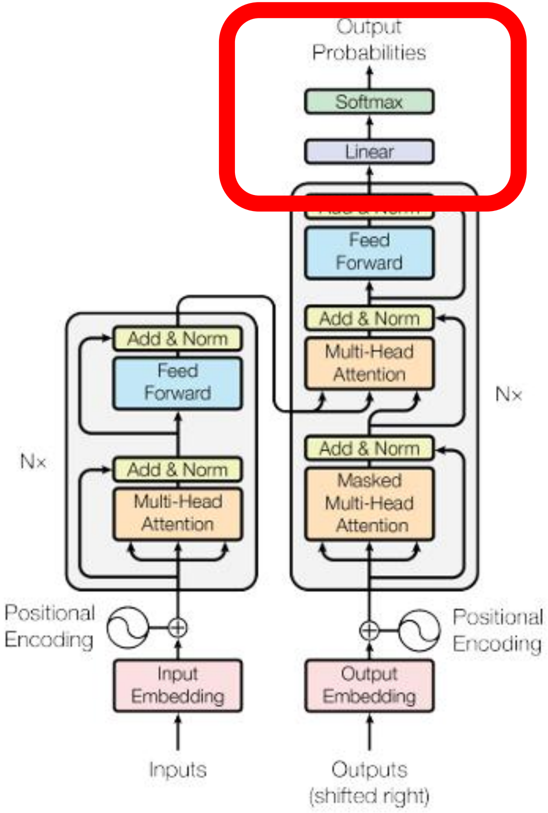
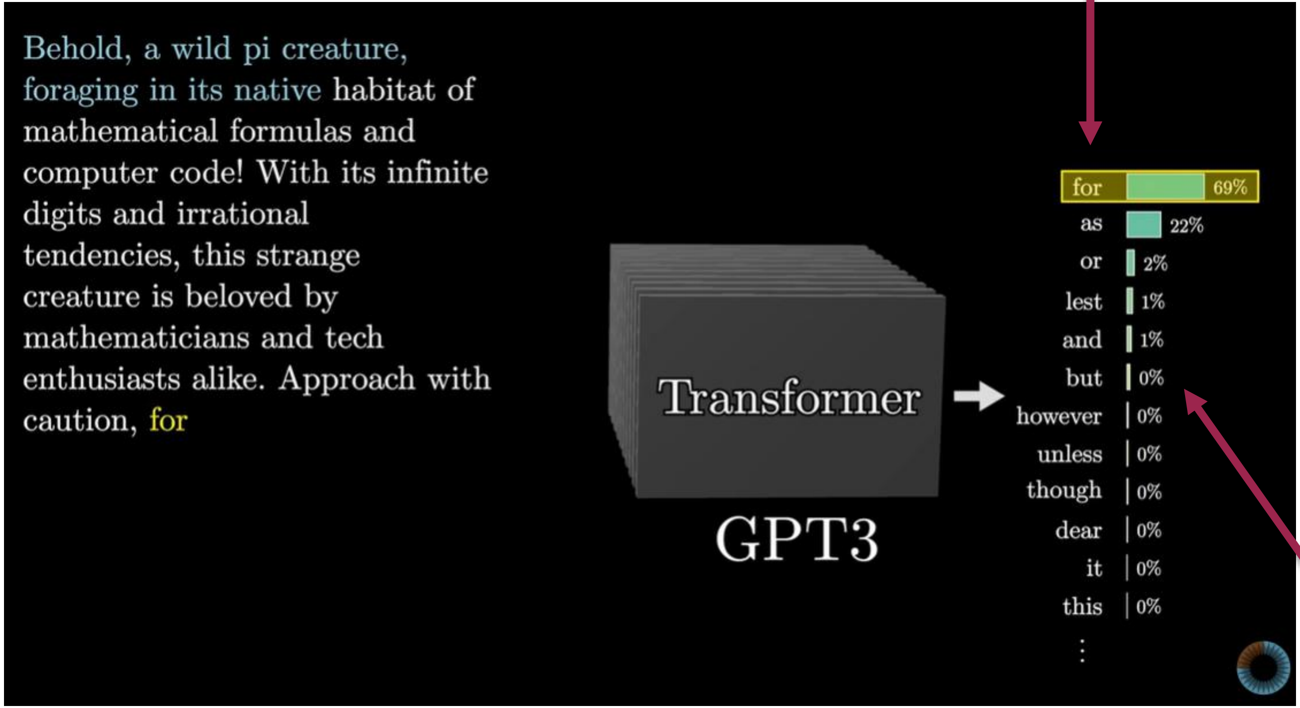



Figure 1: The Transformer - model architecture.

# Transformers architecture

Linear (dictionary)

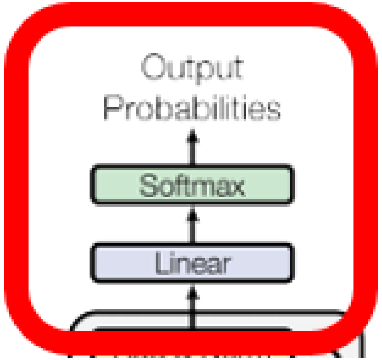



Softmax

Transformers, the tech behind LLMs | Deep Learning Chapter 5

# Decoder Layer

Output Layer



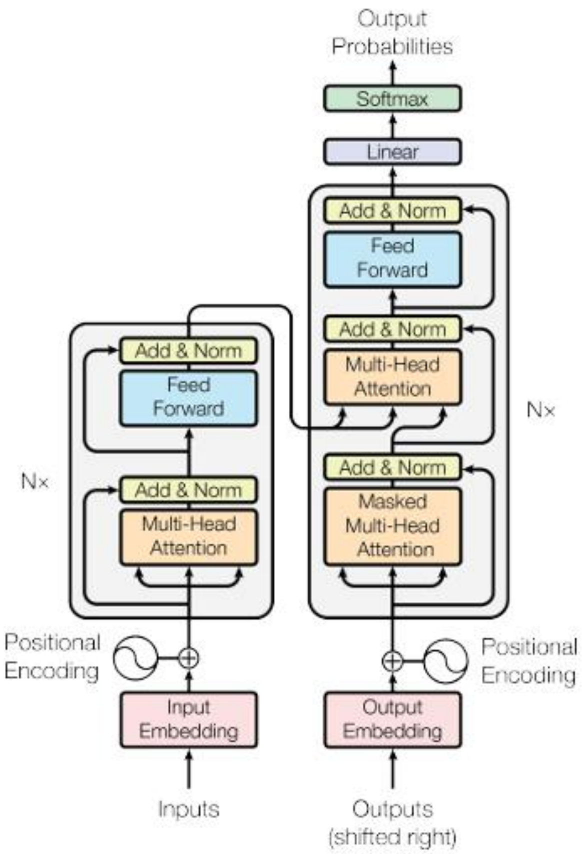



Figure 1: The Transformer - model architecture.

# 2017-present

*„Attention Is All You Need"*

2017

GPT-1 (OpenAI)

BERT (Google)

2018

GPT-2 (OpenAI)

T5 (Google)

2019

GPT-3 (OpenAI)

2020

Fine-tuning

2021

RLHF

ChatGPT (OpenAI)

2022

GPT-4 (OpenAI)

Gemini (Google)

Claude (Anthropic)

LLaMa (Meta)

2023

**Gradient Chatbot**

2024 & 2025

xLSTM
(Extended LSTM)

GPT-4
(OpenAI)

Gemini
(Google)

Claude
(Anthropic)

LLaMa
(Meta)

GPT-1
(OpenAI)

BERT
(Google)

GPT-2
(OpenAI)

T5
(Google)

RLHF

ChatGPT
(OpenAI)

GPT-3
(OpenAI)

Fine-tuning

*„Attention Is All You Need"*

**Gradient Chatbot**

2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 & 2025

# Live coding

# Questions
# &
# Discussion

Thank you!
See you next week.