

# Attention, Transformers, LLM and ChatGPT

Dr. Mykhailo Vladymyrov

DSL XI Seminar, 26.04.2022

# Plan

1. Tools: Definitions, attention mechanism, etc
2. Transformer model
3. LLM
4. Final remarks

# Plan

- 1.Tools:
  - +Definitions: dot product, projection, Softmax, NN. Metric space?
  - +Attention mechanism:
    - self attention as information routing  
`scores = torch.matmul(query, key.transpose(1, 2)); attention_weights = nn.functional.softmax(scores, dim=-1)`
    - // Spatial attention  
`attention_map = tf.layers.conv2d(inputs, filters=1, kernel_size=3, padding='same', activation=tf.nn.sigmoid); weighted_inputs = tf.multiply(inputs, attention_map)`
  - +Cross-attention on sequential data:
    - Basically, self attention
    - proposed by Dzmitry Bahdanau et al. in their paper "Neural Machine Translation by Jointly Learning to Align and Translate". In Bahdanau Attention, the attention scores are computed using a learned weighted sum of the current Decoder hidden state and all the encoder hidden states.  
`decoder_hidden_with_time_axis = tf.expand_dims(decoder_hidden, axis=1)`  
`score = tf.nn.tanh(self.W1(encoder_outputs) + self.W2(decoder_hidden_with_time_axis))`
    - proposed by Minh-Thang Luong et al. in their paper "Effective Approaches to Attention-based Neural Machine Translation". In Luong Attention, the attention scores are computed using the dot product of the current decoder hidden state and all the encoder hidden states.  
`score = tf.matmul(encoder_outputs, decoder_hidden_with_time_axis)`
  - +Multi-head attention
- 2. +Transformer model
  - Rnn based vs self-attention based. Who when how
  - Transformer model, information routing; positional encoding
  - Advantages: Parallelizable, Captures Long-Term Dependencies, Flexible
  - Training of transformer
- 3. LLM Vaswani Transformer (38M), Codex (12B / 159GB FT), GPT-3 (175B / >600GB?) & ChatGPT (175B / 45TB?), GPT-4 (100T / ?)
  - ChatGPT trained with
- 4. Final remarks:
  - Like all other deep learning–based language models, Codex is capturing statistical correlations between code fragments.
  - a strong student who completes an introductory computer science course is expected to be able to solve a larger fraction of problems than Codex-12B.
  - Transformer for non-lang
  - Language as a system

# 1. Tools

# 1.1. Definitions

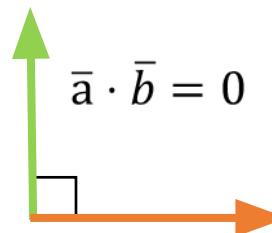
- Dot product

- measure of similarity:

$$\bar{a} \cdot \bar{b} \approx 1$$



$$\bar{a} \cdot \bar{b} = 0$$



$$\bar{a} \cdot \bar{b} \approx -1$$



$$\bar{a}^T \cdot \bar{b} = |\bar{a}| |\bar{b}| \cos \theta_{\bar{a}\bar{b}}$$

# 1.1. Definitions

- Matrix multiplication  $\mathbf{M} \bar{a}$ 
  - measure of similarity between each row  $\overline{\mathbf{m}_i}$  of  $\mathbf{M}$  and  $\bar{a}$ :

$$\mathbf{M} \bar{a} = \begin{pmatrix} \overline{\mathbf{m}_1}^T \\ \overline{\mathbf{m}_2}^T \\ \vdots \\ \overline{\mathbf{m}_n}^T \end{pmatrix} \bar{a} = \begin{pmatrix} \overline{\mathbf{m}_1}^T \cdot \bar{a} \\ \overline{\mathbf{m}_2}^T \cdot \bar{a} \\ \vdots \\ \overline{\mathbf{m}_n}^T \cdot \bar{a} \end{pmatrix}$$

# 1.1. Definitions

- Matrix multiplication  $\mathbf{M} \bar{a}$

– measure of similarity between each row  $\overline{\mathbf{m}}_i$  of  $\mathbf{M}$  and  $\bar{a}$ :

$$\mathbf{M} \bar{a} = \begin{pmatrix} \overline{\mathbf{m}}_1^T \\ \overline{\mathbf{m}}_2^T \\ \vdots \\ \overline{\mathbf{m}}_n^T \end{pmatrix} \bar{a} = \begin{pmatrix} \overline{\mathbf{m}}_1^T \cdot \bar{a} \\ \overline{\mathbf{m}}_2^T \cdot \bar{a} \\ \vdots \\ \overline{\mathbf{m}}_n^T \cdot \bar{a} \end{pmatrix}$$

– projection to n-dimensional space with basis given by  $\widehat{\mathbf{m}}_1 \dots \widehat{\mathbf{m}}_N$ , scaled by  $|\overline{\mathbf{m}}_1| \dots |\overline{\mathbf{m}}_N|$

$$\mathbf{M} \bar{a} = \begin{pmatrix} \overline{\mathbf{m}}_1^T \\ \overline{\mathbf{m}}_2^T \\ \vdots \\ \overline{\mathbf{m}}_n^T \end{pmatrix} \bar{a} = \begin{pmatrix} \overline{\mathbf{m}}_1^T \cdot \bar{a} \\ \overline{\mathbf{m}}_2^T \cdot \bar{a} \\ \vdots \\ \overline{\mathbf{m}}_n^T \cdot \bar{a} \end{pmatrix} = \begin{pmatrix} |\overline{\mathbf{m}}_1| \widehat{\mathbf{m}}_1^T \cdot \bar{a} \\ |\overline{\mathbf{m}}_2| \widehat{\mathbf{m}}_2^T \cdot \bar{a} \\ \vdots \\ |\overline{\mathbf{m}}_N| \widehat{\mathbf{m}}_N^T \cdot \bar{a} \end{pmatrix}$$

## 1.1. Definitions

- Softmax

- normalizes vector of  $n$  numbers  $\vec{l}$  to pseudo-probabilities  $w_i$ :

$$w_i = SM(\vec{l})_i = \frac{e^{l_i}}{\sum_{j=1}^n e^{l_j}},$$

$$\text{s. t. } \sum_{i=1}^n w_i = 1 \quad \text{and} \quad 0 \leq w_i \leq 1$$

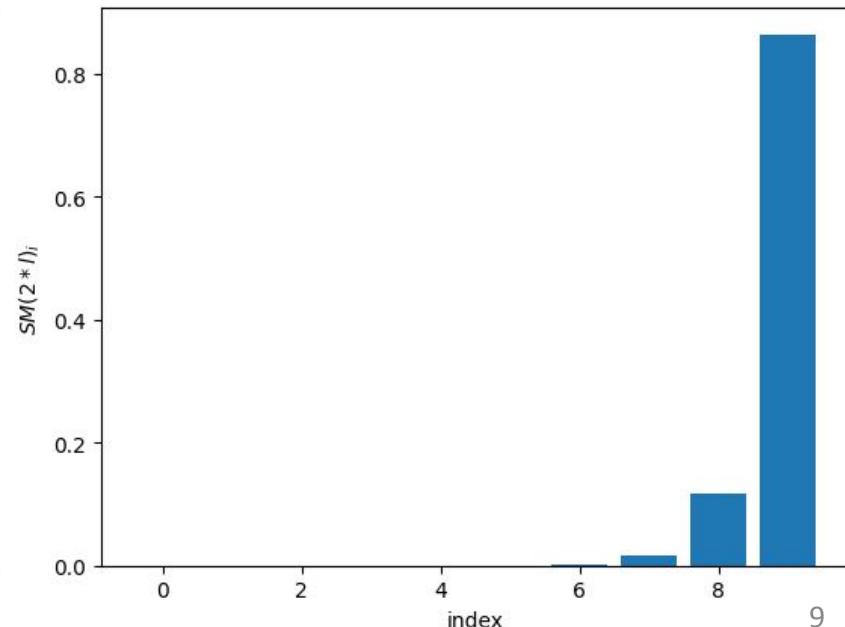
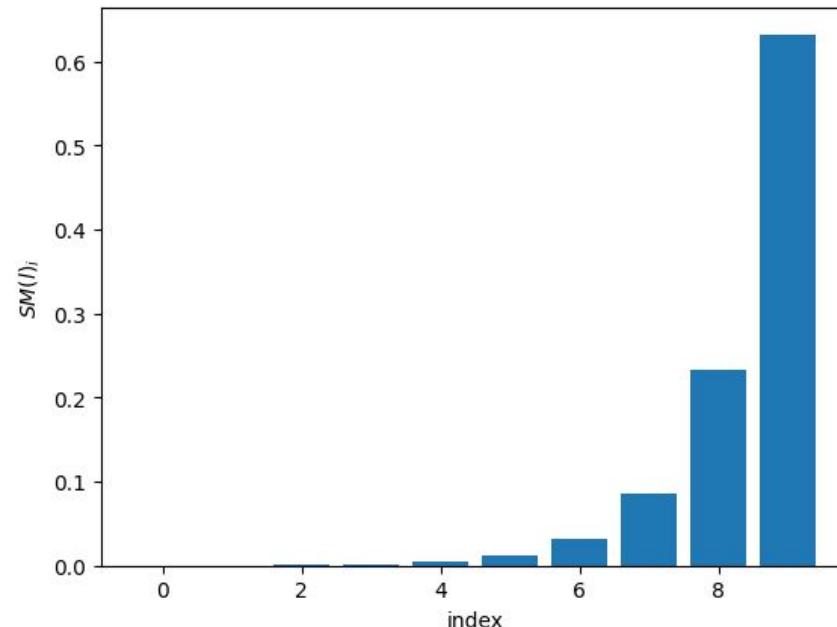
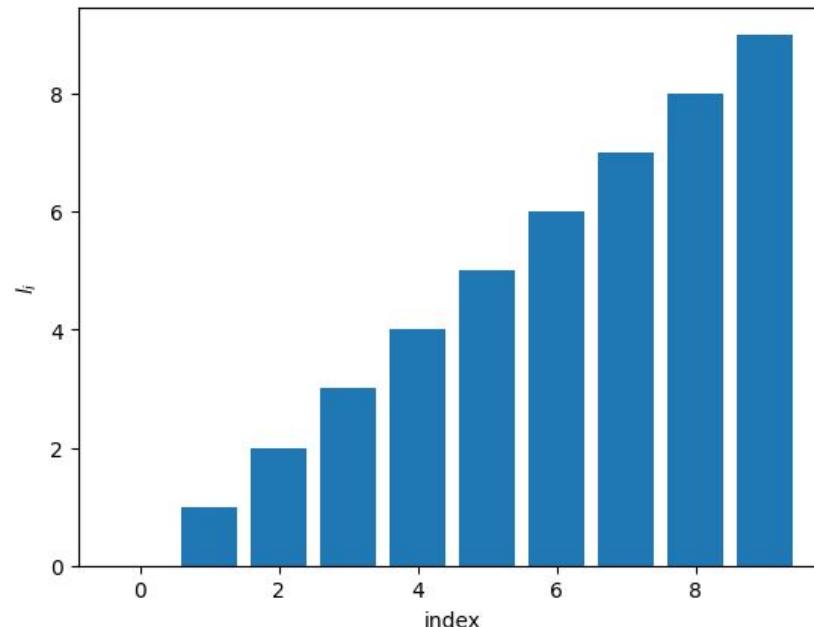
# 1.1. Definitions

- Softmax

- normalizes vector of  $n$  numbers  $\vec{l}$  to pseudo-probabilities  $w_i$ :

$$w_i = SM(\vec{l})_i = \frac{e^{l_i}}{\sum_{j=1}^n e^{l_j}},$$

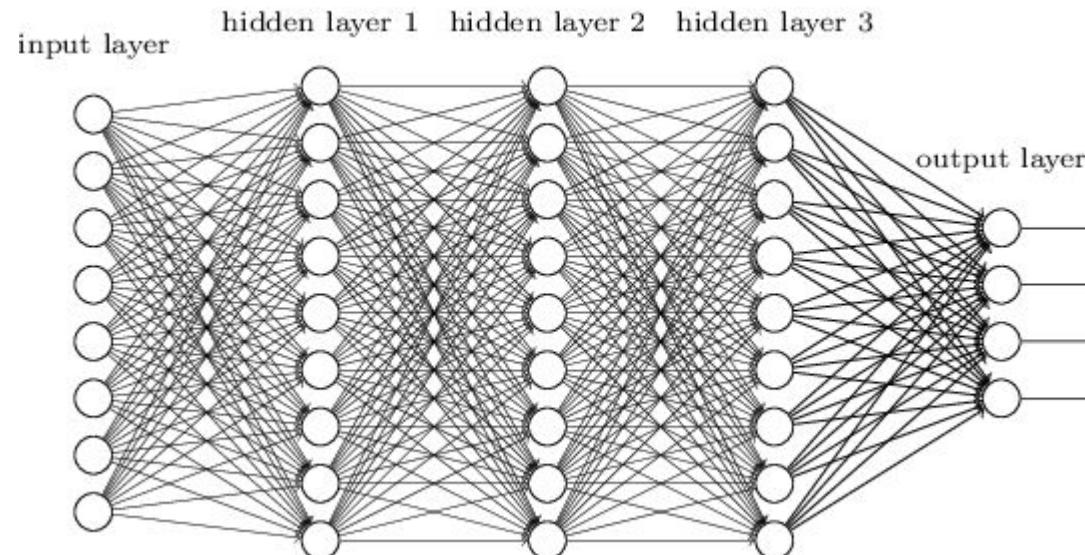
s.t.  $\sum_{i=1}^n w_i = 1 \quad \text{and} \quad 0 \leq w_i \leq 1$



# 1.1. Definitions

- **Neural network (NN)**

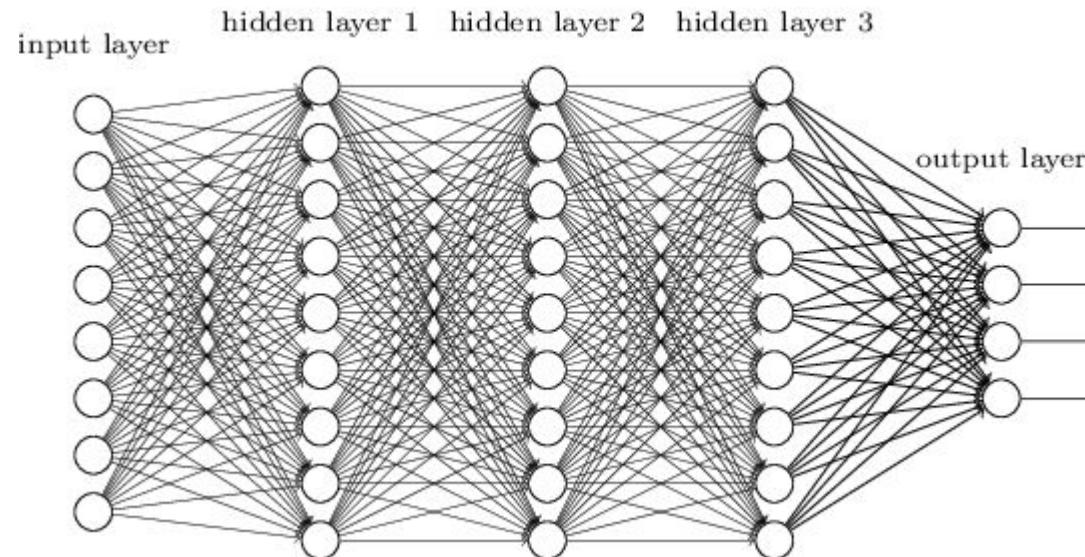
- NN in its simplest form – feed-forward network – sequential processing of the data by sequence of linear transformations, non-linear activation functions and normalizations.



# 1.1. Definitions

- **Neural network (NN)**

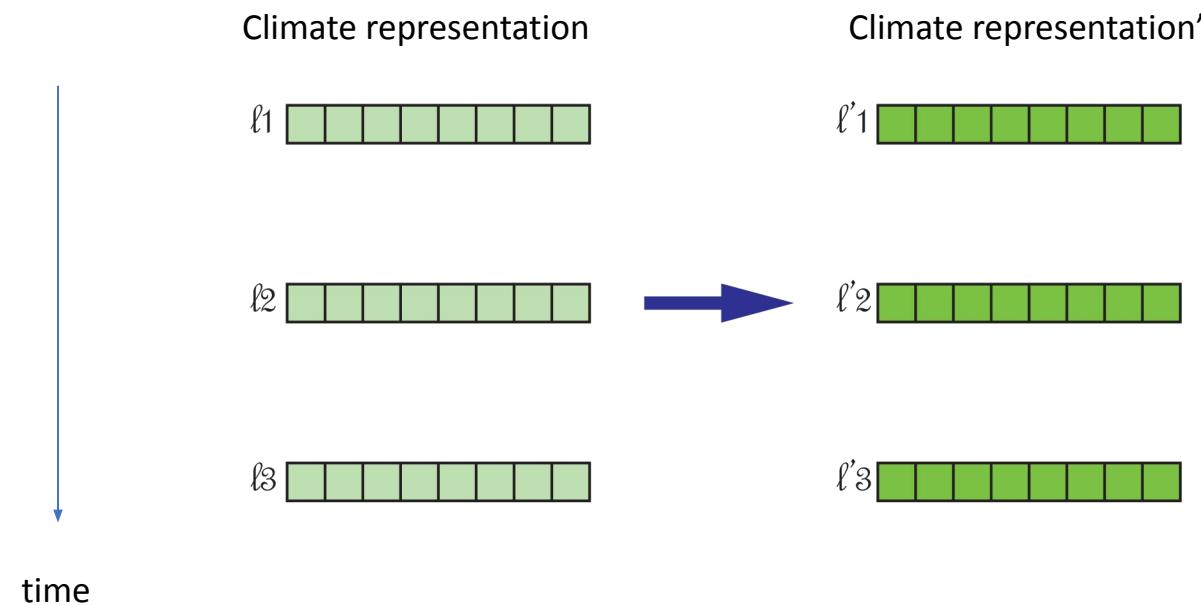
- NN in its simplest form – feed-forward network – sequential processing of the data by sequence of linear transformations, non-linear activation functions and normalizations.



- NN is a general fitter: any function can be approximated with a NN
- FFN operates on level of independent scalars.

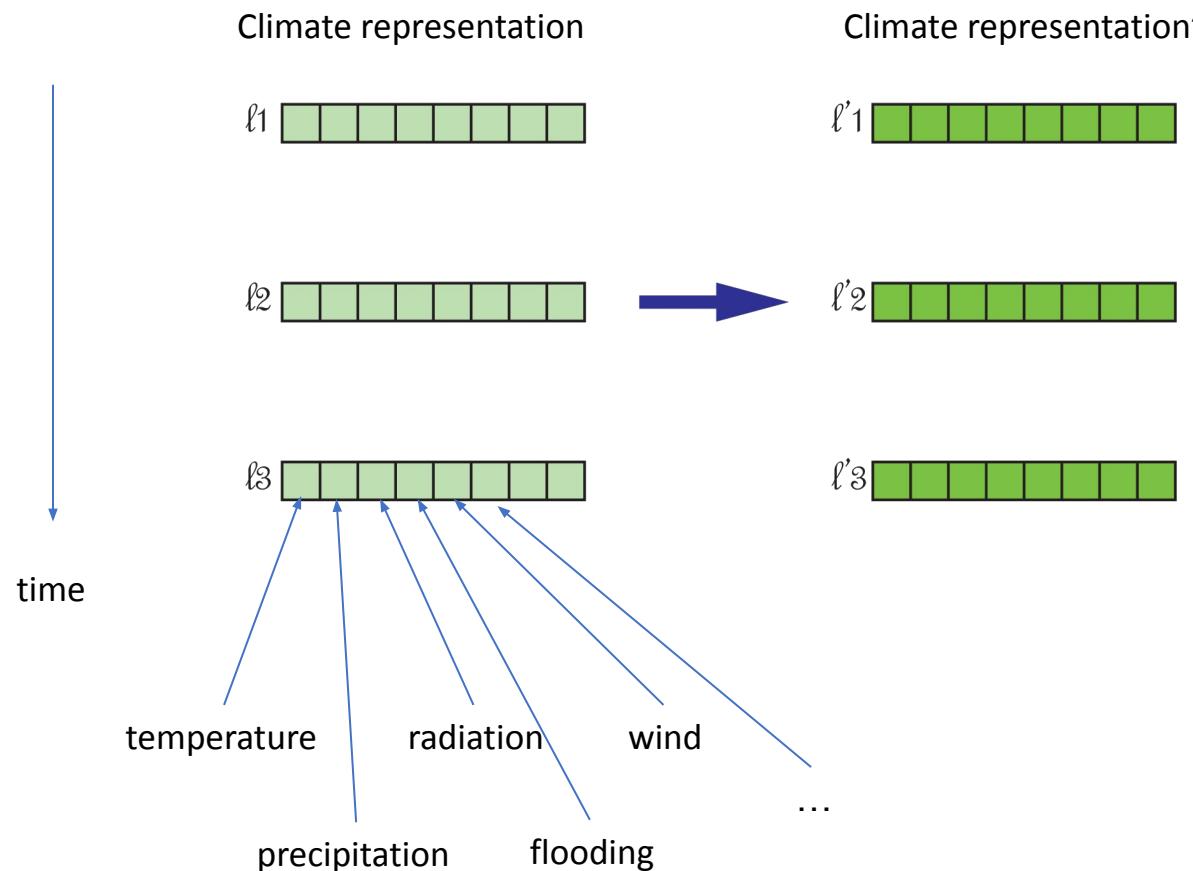
# 1.2. Attention mechanism

Self attention is routing mechanism of tensor information.



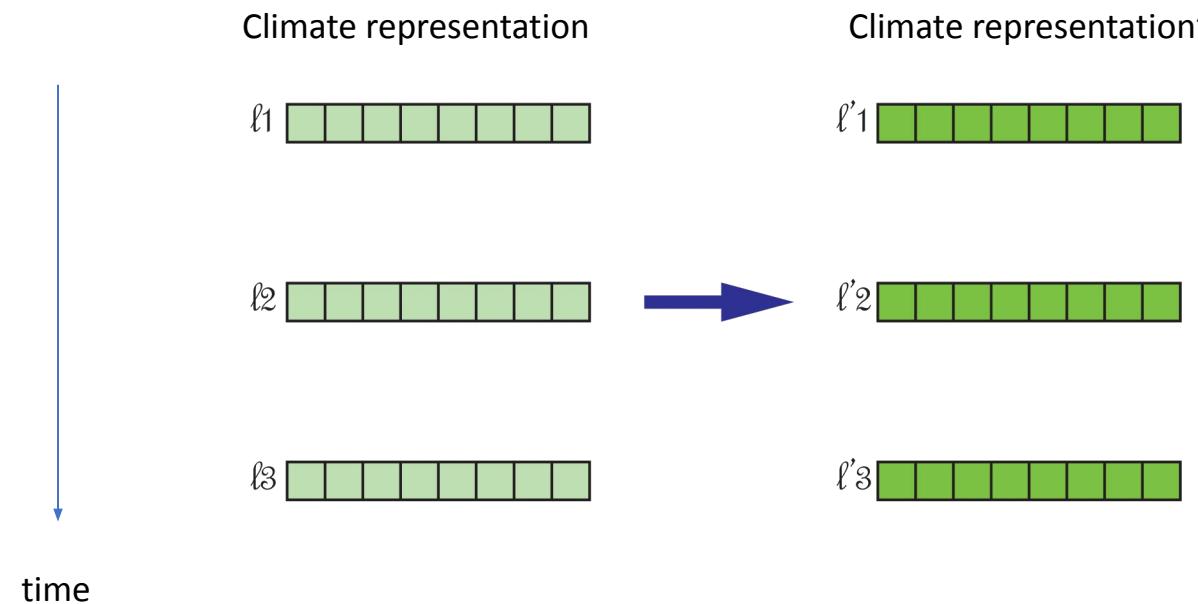
# 1.2. Attention mechanism

Self attention is routing mechanism of tensor information.



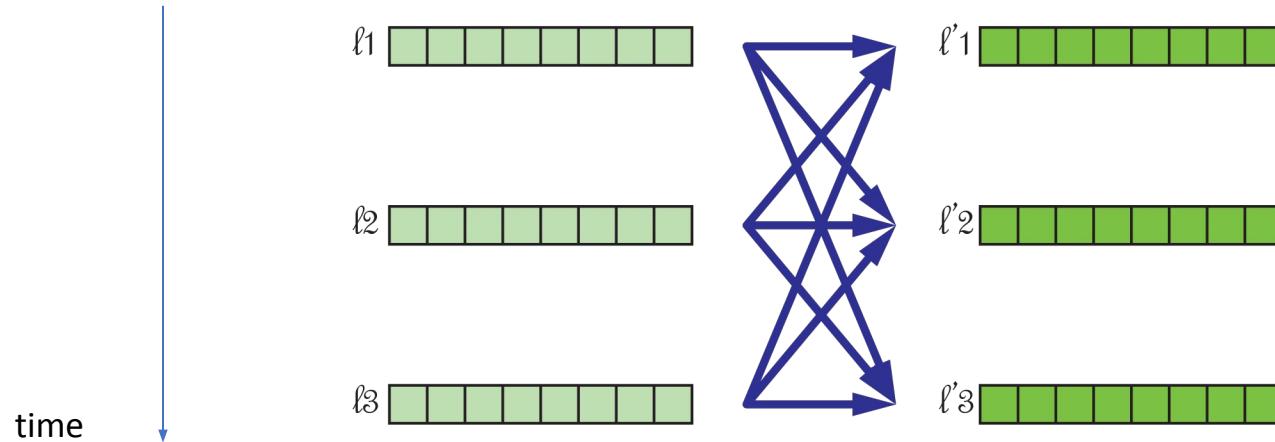
# 1.2. Attention mechanism

Self attention is routing mechanism of tensor information.



# 1.2. Attention mechanism

Self attention is routing mechanism of tensor information.



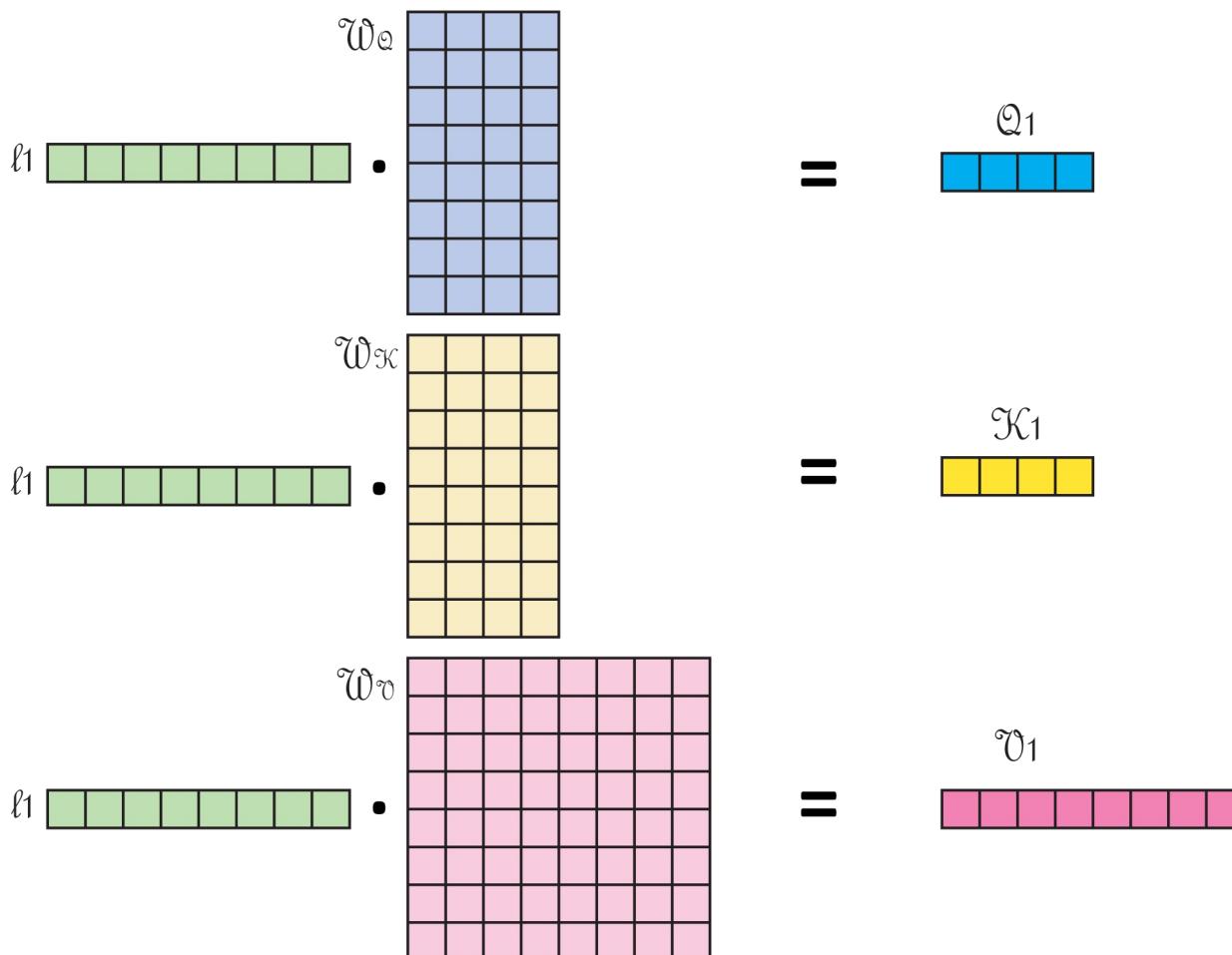
Given sequence of  $n$  vectors of dimension  $d$ :

- FFN:  $(n*d)^2$  parameters; any position
- CNN:  $d^2 * k$ ;  $k$  nearest positions
- RNN:  $d^2 * c$ ; memory of “nearby” positions

# 1.2. Attention mechanism

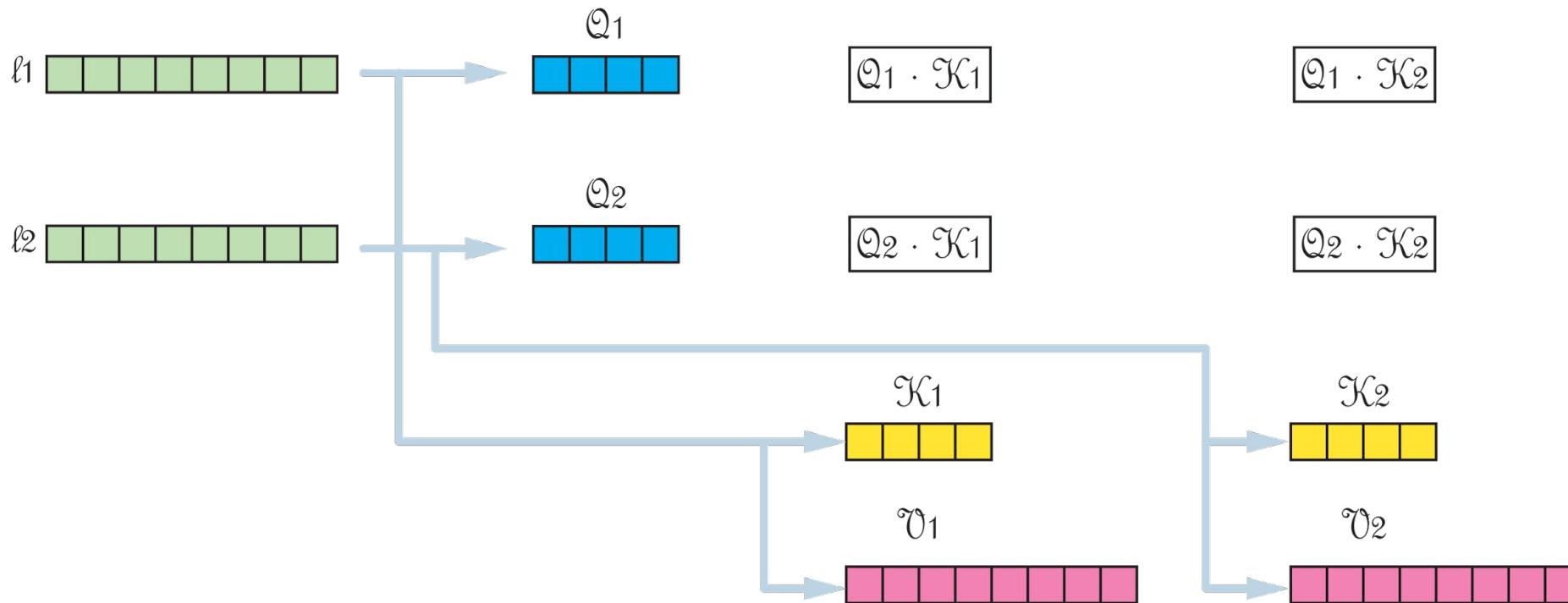
## Query-Key-Value

- Projection from latent space to Q, K, V space
- Projection matrices are same for each element of the sequence



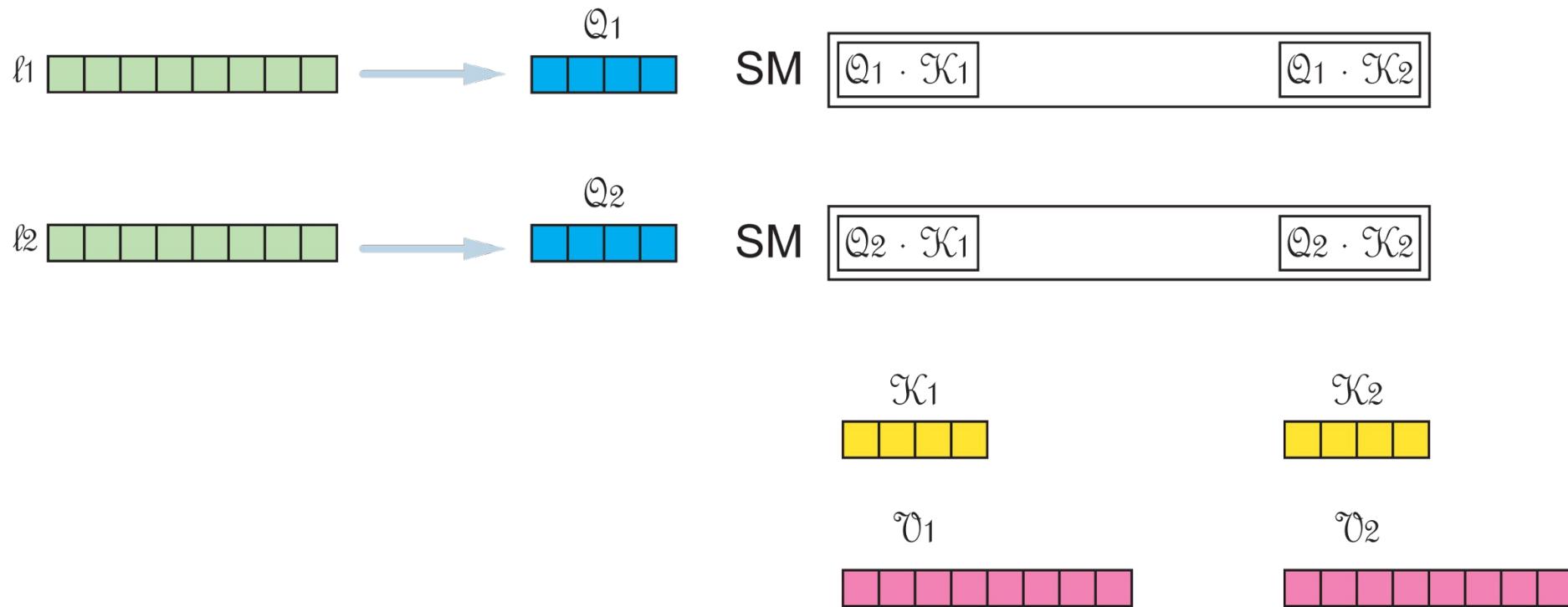
# 1.2. Attention mechanism

- Attention scores – similarity between all Key-Query pairs



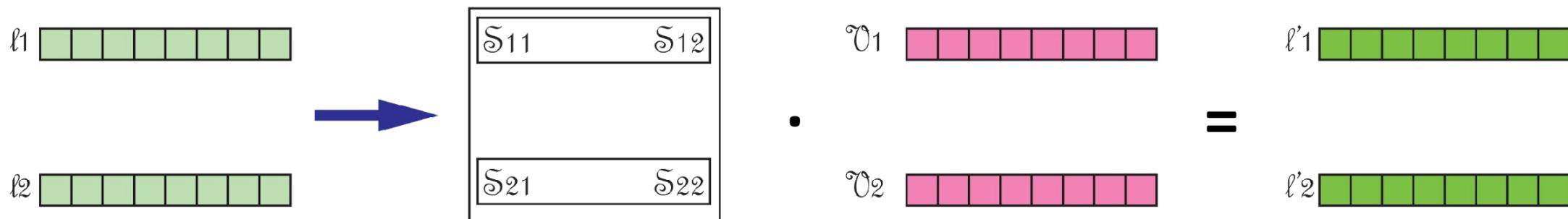
# 1.2. Attention mechanism

- Attention weights – are the scores normalized with SoftMax



# 1.2. Attention mechanism

- Attention weighting

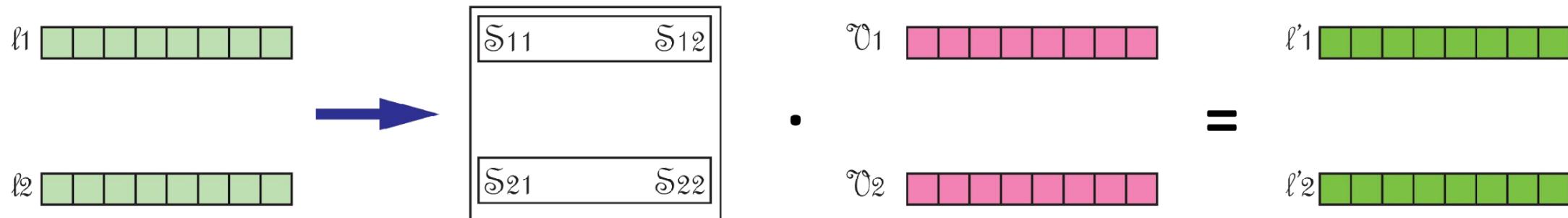


Given sequence of  $n$  vectors of dimension  $d$  and KV dimension  $m$ :

- **Self-attention:  $m \cdot d * 2 + d^2$  parameters; any position**
- FFN:  $(n \cdot d)^2$  parameters; any position
- CNN:  $d^2 * k$ ;  $k$  nearest positions
- RNN:  $d^2 * c$ ; memory of “nearby” positions

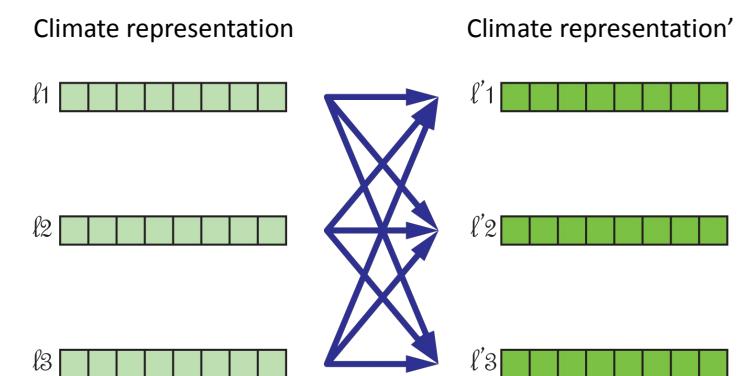
# 1.2. Attention mechanism

- Attention weighting



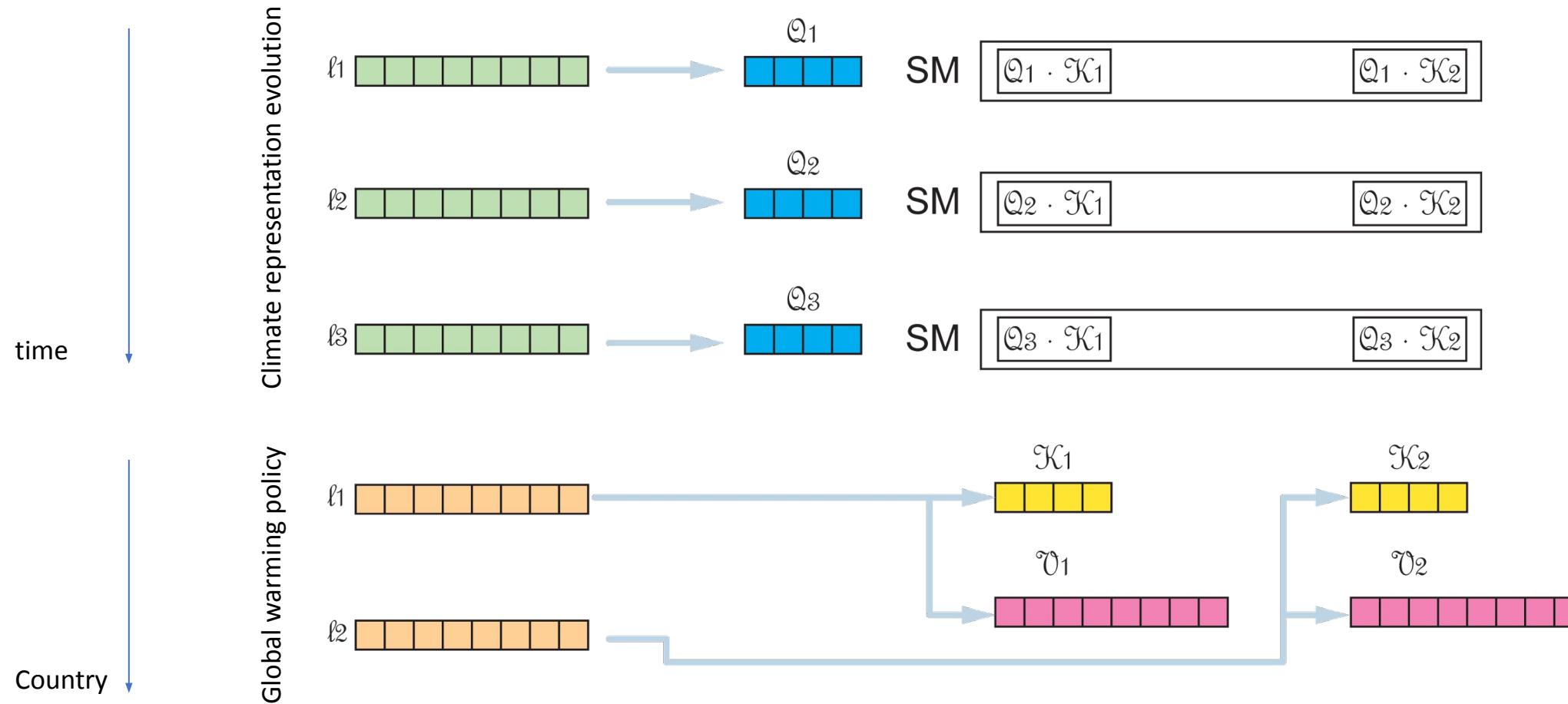
Given sequence of  $n$  vectors of dimension  $d$  and KV dimension  $m$ :

- **Self-attention:  $m \cdot d * 2 + d^2$  parameters; any position**
- FFN:  $(n \cdot d)^2$  parameters; any position
- CNN:  $d^2 * k$ ;  $k$  nearest positions
- RNN:  $d^2 * c$ ; memory of “nearby” positions



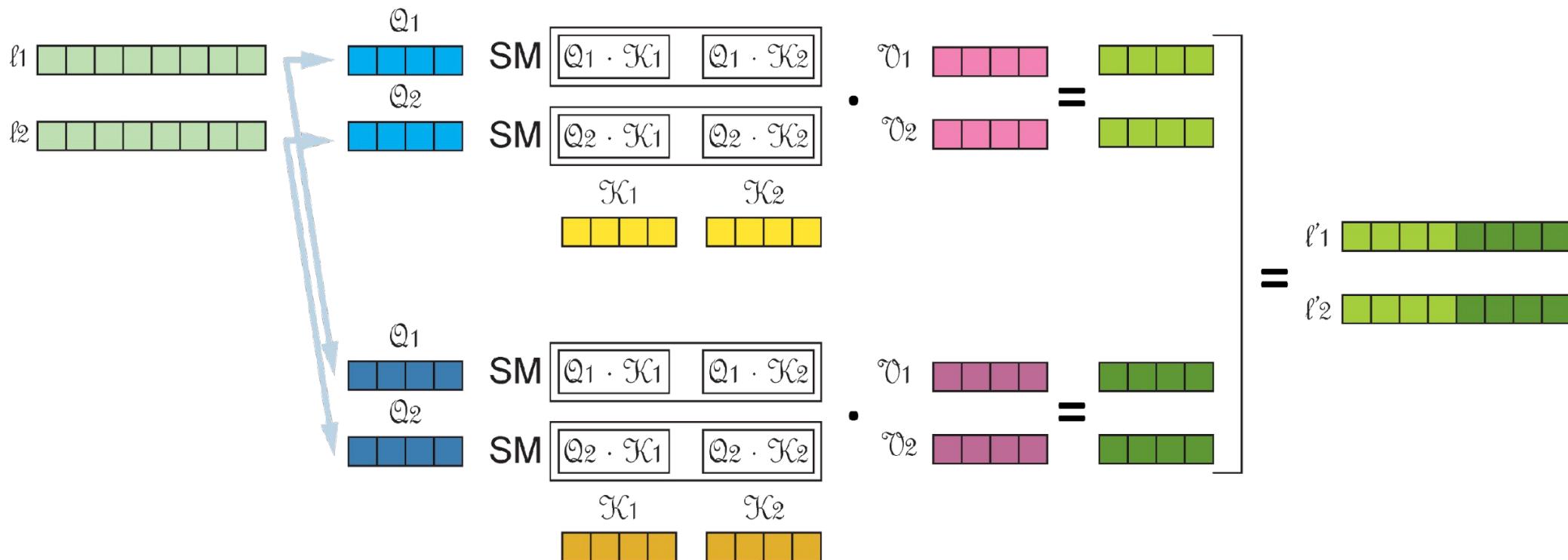
# 1.2. Cross-Attention mechanism

Same, but  $(K, V)$  and  $Q$  are from different sequences.



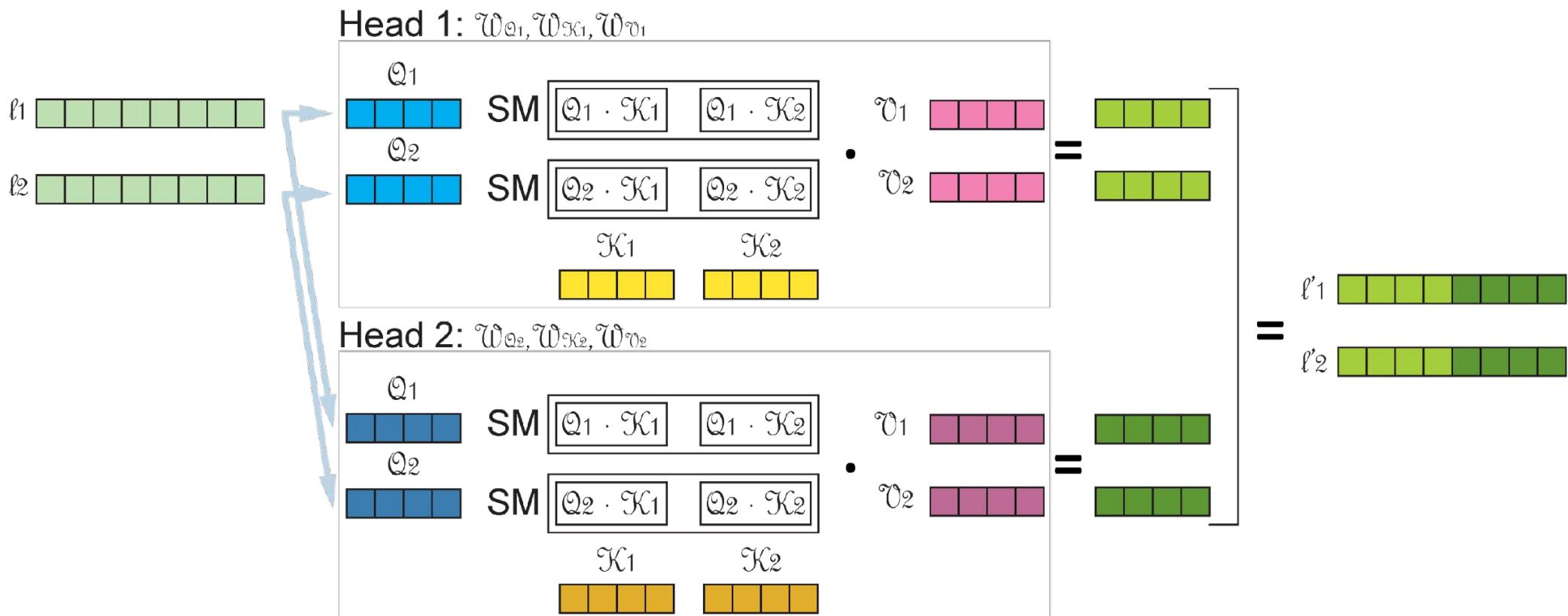
# 1.2. Multi-head Attention mechanism

Same, but now multiple variant of (K, V, Q) are generated. This corresponds to building distinct representations.



# 1.2. Multi-head Attention mechanism

Same, but now multiple variant of (K, V, Q) are generated. This corresponds to building distinct representations.



## 2. Transformer

## 2.1. RNN vs Transformer

- Both RNN and Transformer are designed to handle sequential data.
- Transformer processes the sequence in parallel, while RNN – sequentially
- Maximum sequence length:
  - LSTM: ~100 elements
  - Transformer: ~ 1000 elements

## 2.2. Transformer architecture

Introduced in 2017 for the Machine Translation task.

Relies solely on the attention mechanism

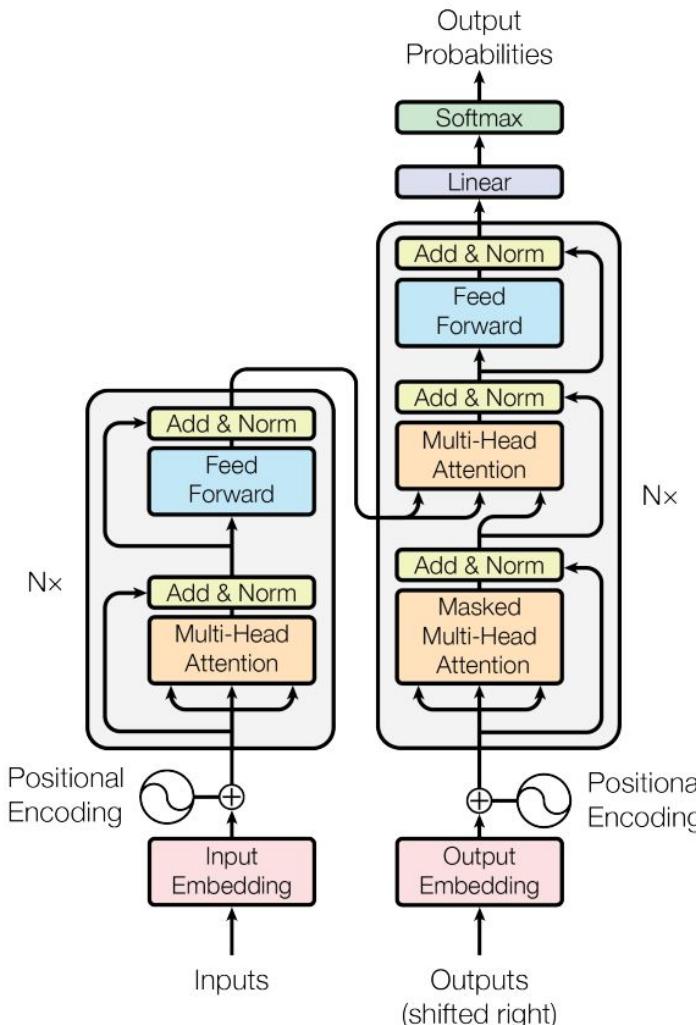
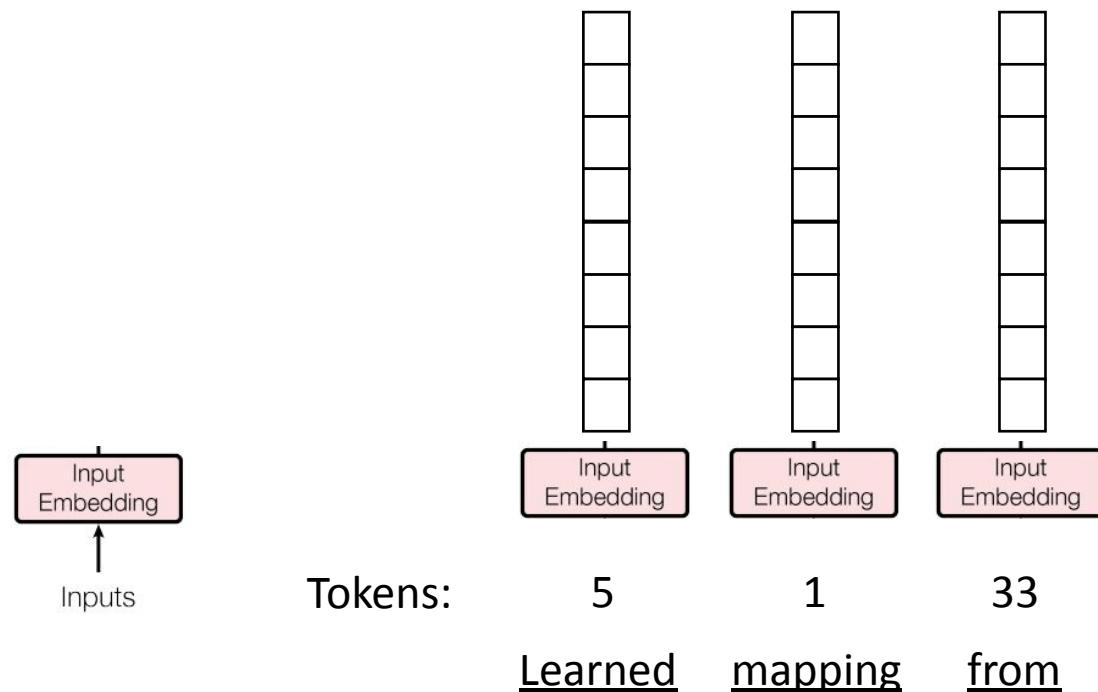


Figure 1: The Transformer - model architecture.

Vaswani et al, 2017, Attention Is All You Need

## 2.2. Input embedding

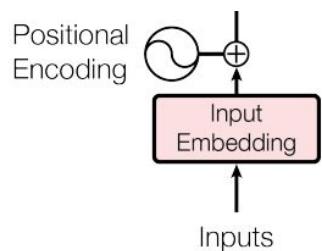
Learned mapping from word ID-s / tokens to model representation space (in practice 512D+, here 8D for demonstration)



## 2.2. Positional encoding

Positional encoding injects information about absolute and relative position of the elements in the sequence.

- Parametrized by same number of params as embedding dimension
- Cyclic in position with range of frequencies

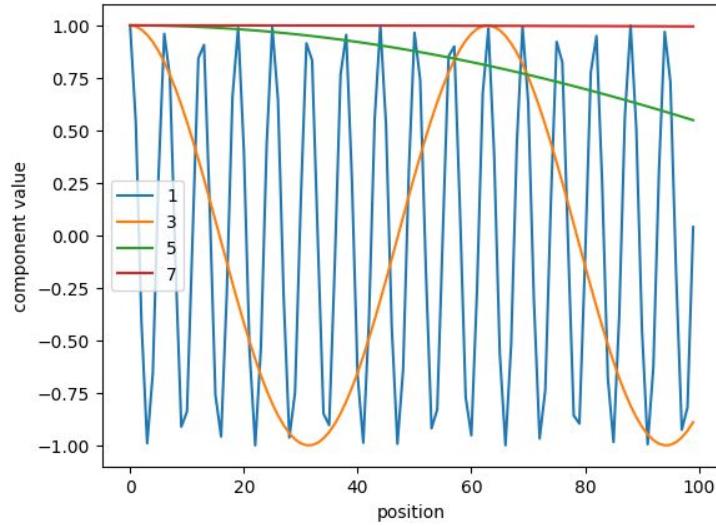
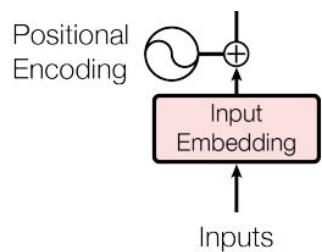
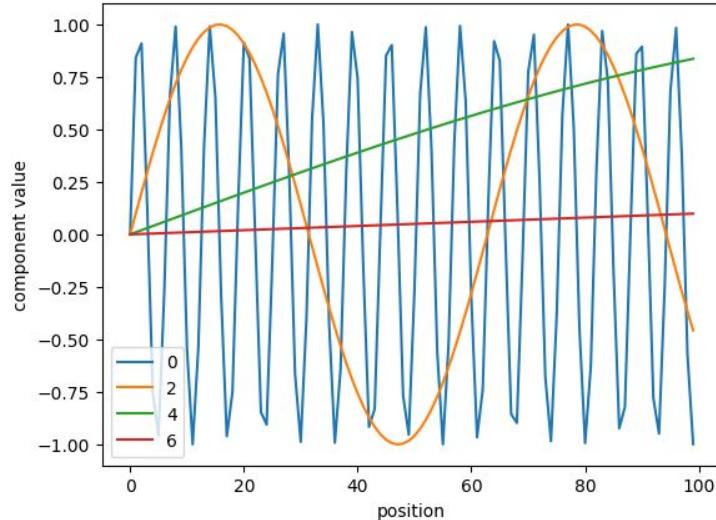


## 2.2. Positional encoding

Positional encoding injects information about absolute and relative position of the elements in the sequence.

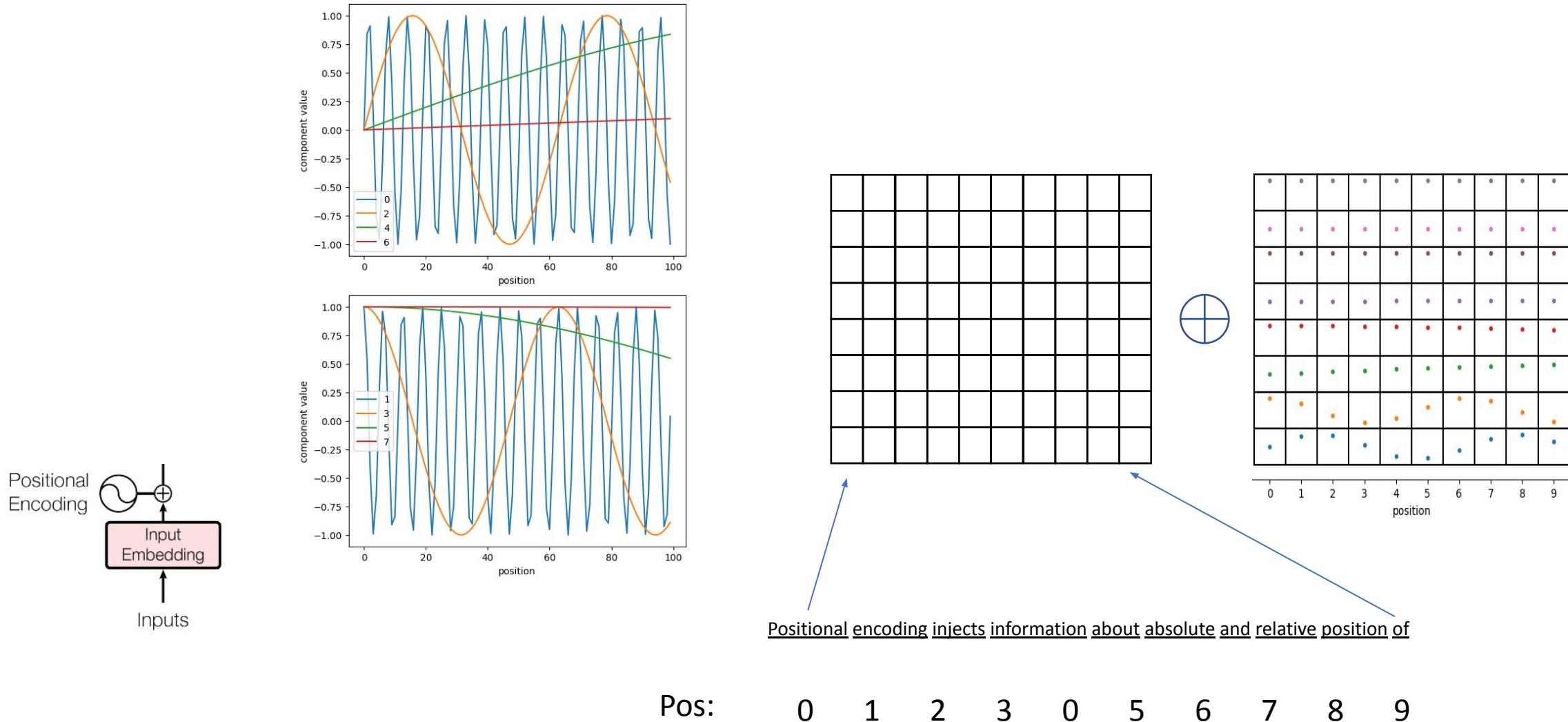
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

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



## 2.2. Positional encoding

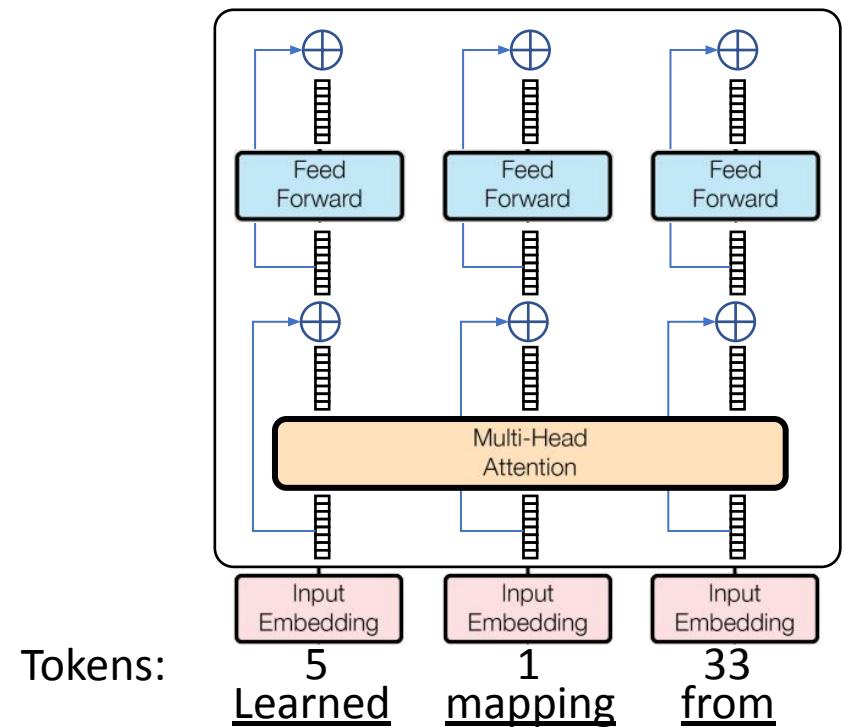
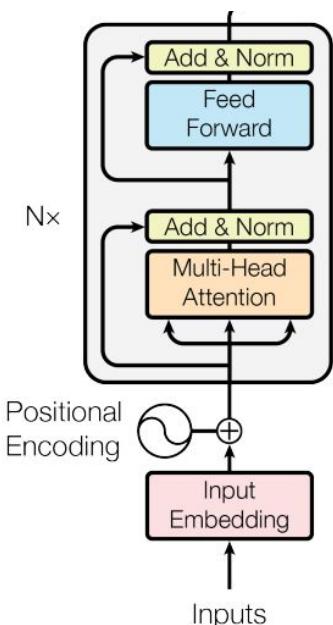
Positional encoding injects information about absolute and relative position of the elements in the sequence.



## 2.2. Information routing: encoder

Encoder combines in a res-net fashion

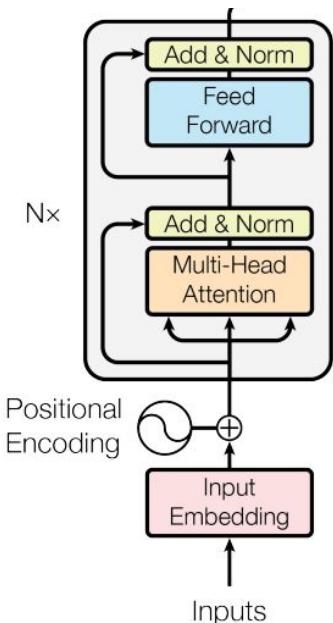
- Self attention between elements of the sequence
- Element-wise update of the representation of sequence elements



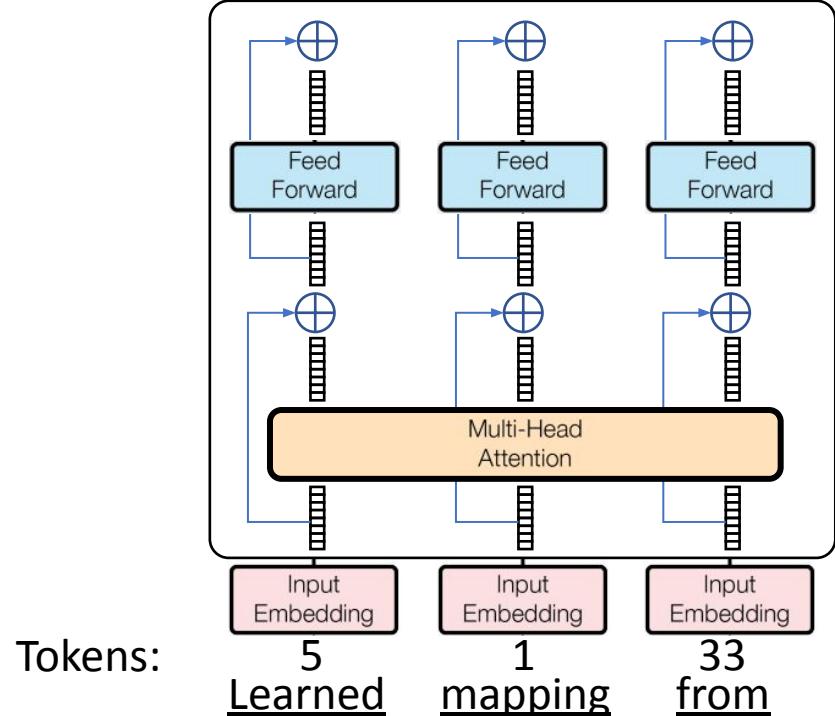
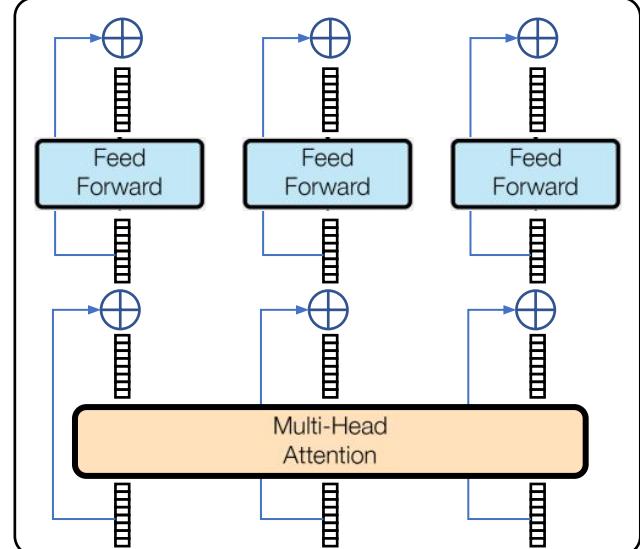
## 2.2. Information routing: encoder

Encoder combines in a res-net fashion

- Self attention between elements of the sequence
- Element-wise update of the representation of sequence elements



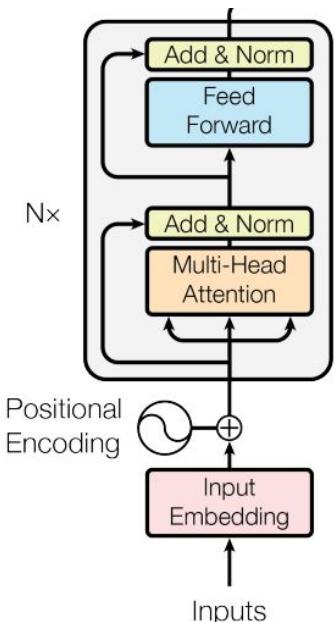
Tokens:



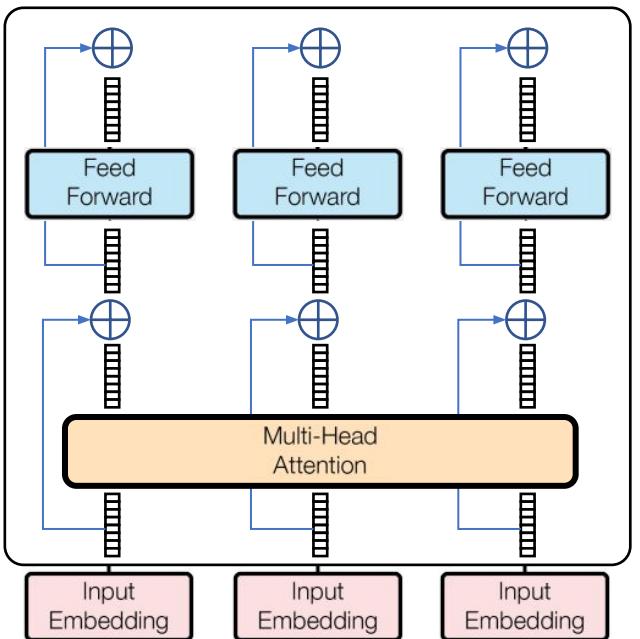
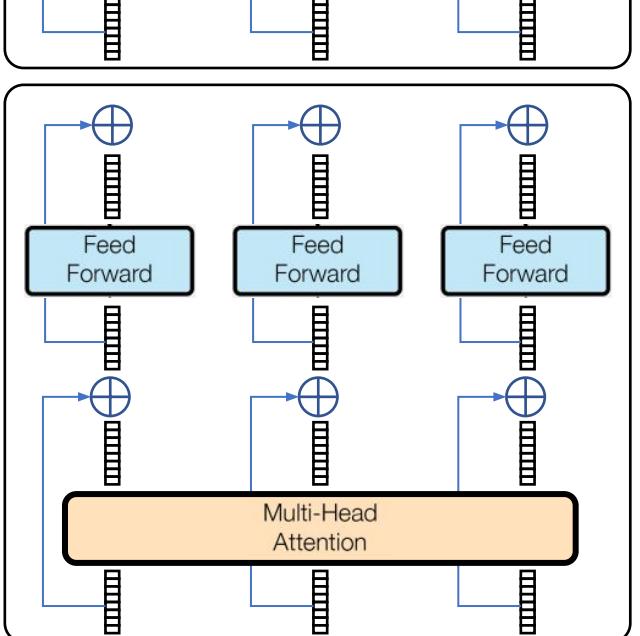
## 2.2. Information routing: encoder

Encoder combines in a res-net fashion

- Self attention between elements of the sequence
- Element-wise update of the representation of sequence elements



Tokens:  
5 Learned  
1 mapping  
33 from



## 2.2. Information routing: decoder

Decoder is similar to encoder:

- Self attention + Cross-attention + FFN

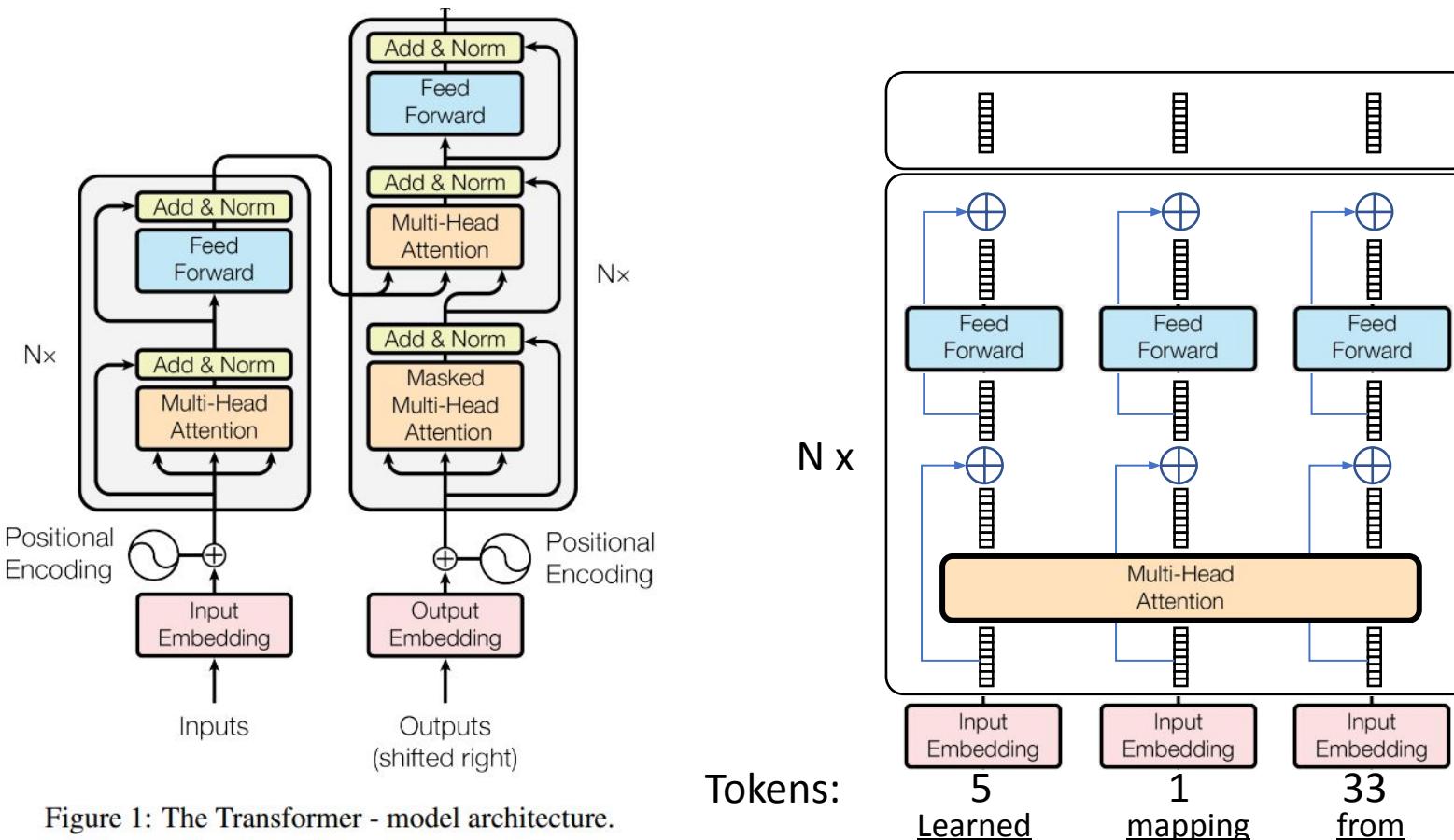


Figure 1: The Transformer - model architecture.

## 2.2. Information routing: decoder

Decoder is similar to encoder:

- Self attention + Cross-attention + FFN

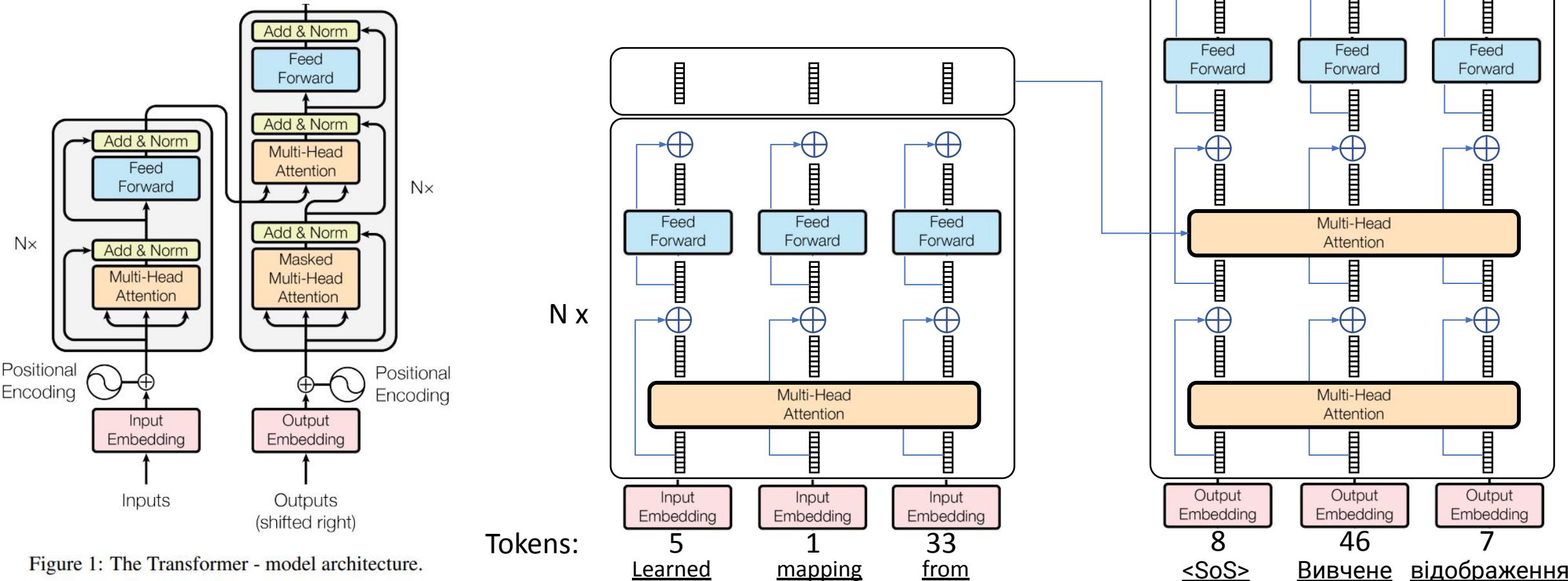


Figure 1: The Transformer - model architecture.

## 2.2. Information routing: decoder

Decoder is similar to encoder:

- Self attention + Cross-attention + FFN

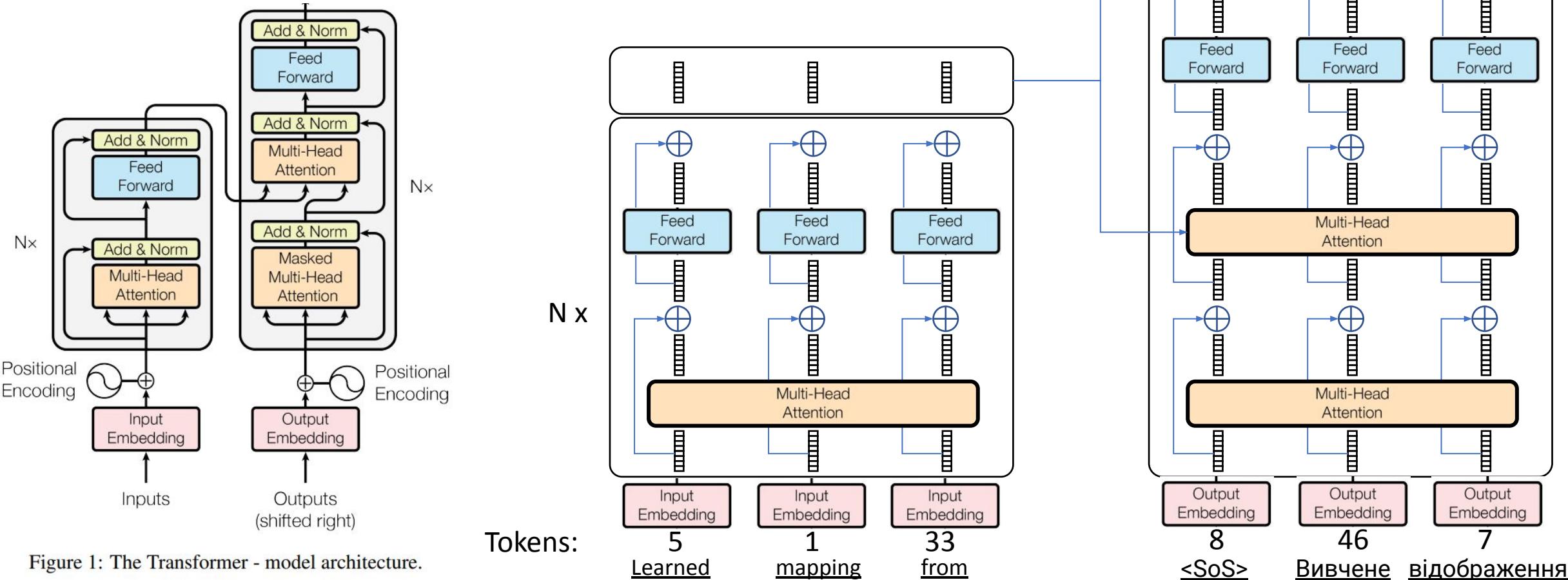


Figure 1: The Transformer - model architecture.

# 2.3. Prediction

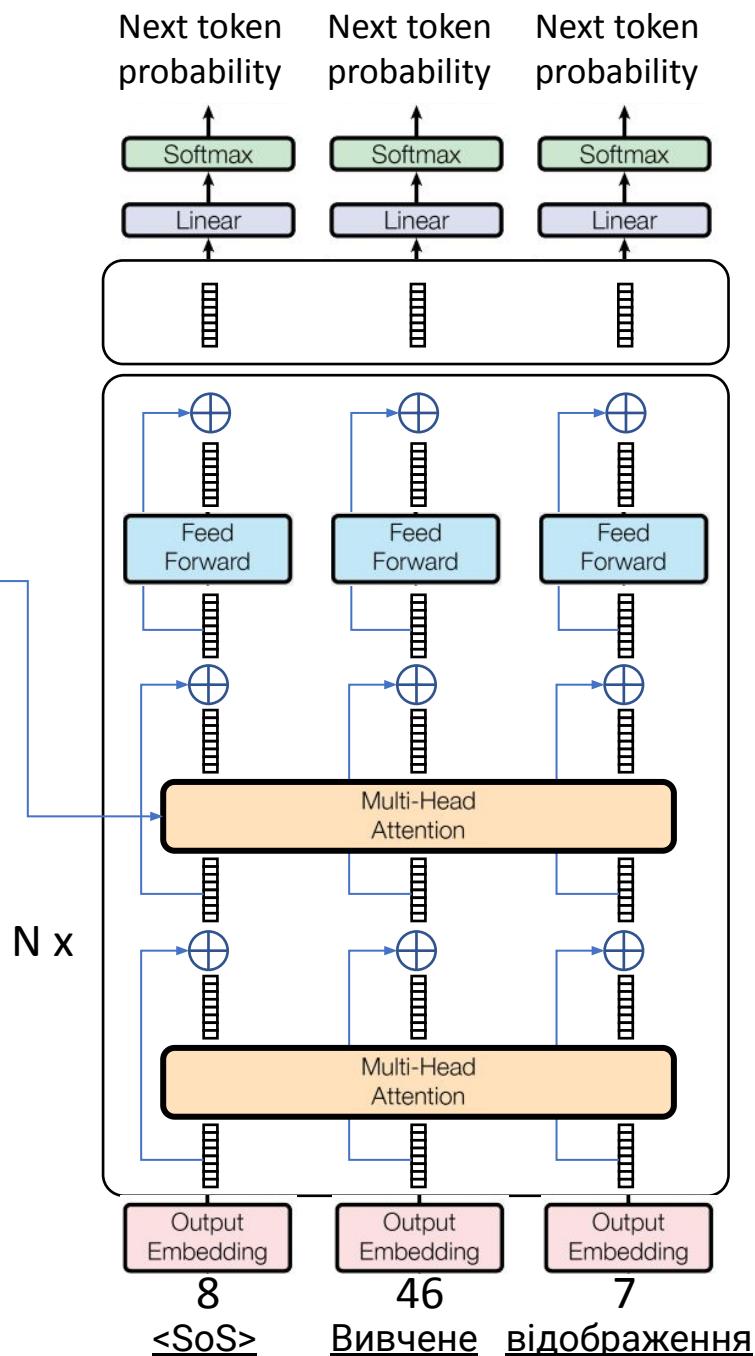
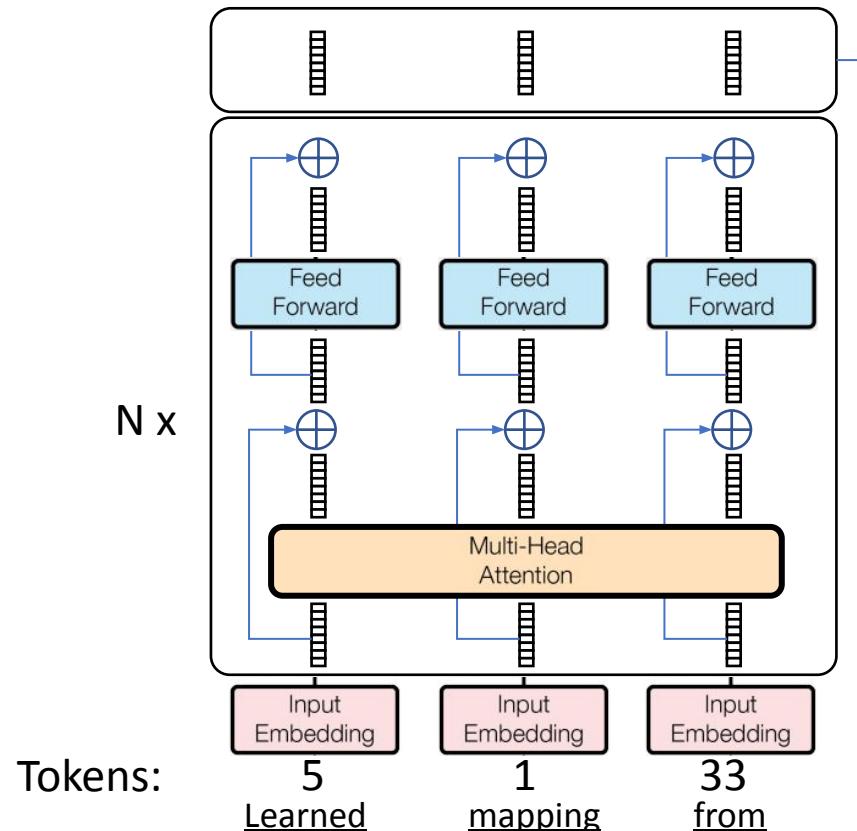
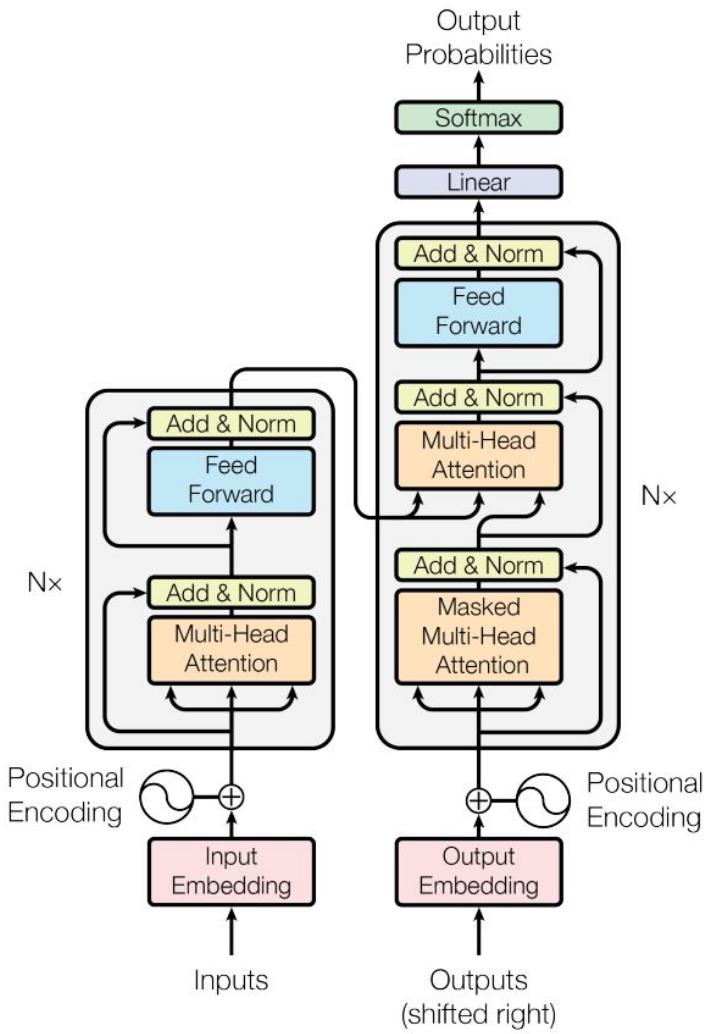


Figure 1: The Transformer - model architecture.

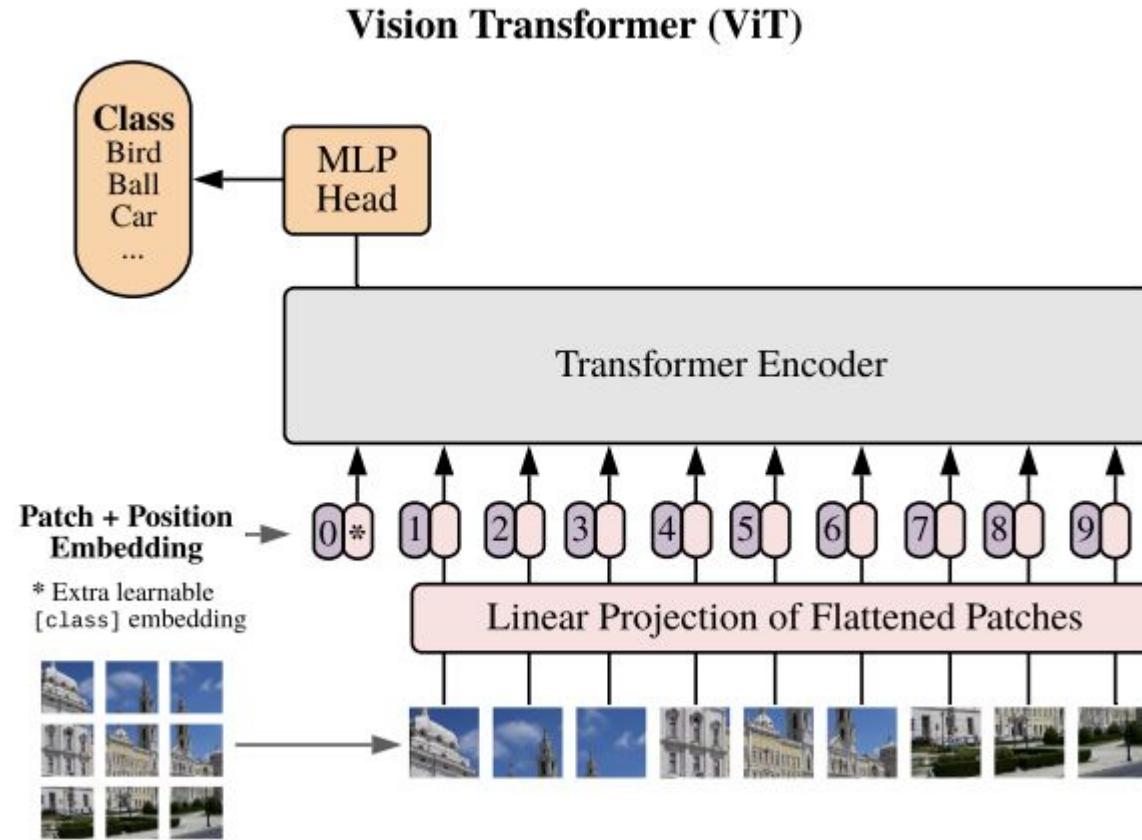
## 2.4. Transformer training

- 37k tokens
- 4.5M sentence pairs
- 8 P100 GPUs:
  - Base (65M) : 100k iterations, 12h
  - Big (213M): 300k iterations, 3.5d

Way faster than previous models (RNNs)

# Transformer for non-language tasks

Same approach was found to be useful to non-language tasks, e.g. visual transformer:



Dosovitskiy et al, 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

### 3. LLM

# Overview of LLMs

- Vaswani Transformer (65M)
- Codex (12B)
- GPT-3 (175B) & ChatGPT (175B)
- GPT-4 (1T)

# Overview of LLMs

[GPT-J](#) (6B) (EleutherAI)

[GPT-Neo](#) (1.3B, 2.7B, 20B) (EleutherAI)

[Pythia](#) (1B, 1.4B, 2.8B, 6.9B, 12B)

[Polyglot](#) (1.3B, 3.8B, 5.8B)

J1 (7.5B, 17B, 178B) (AI21)

[LLaMa](#) (7B, 13B, 33B, 65B) (Meta)

[OPT](#) (1.3B, 2.7B, 13B, 30B, 66B, 175B) (Meta)

[Fairseq](#) (1.3B, 2.7B, 6.7B, 13B) (Meta)

[Cerebras-GPT](#) (1.3B, 2.7B, 6.7B, 13B) (Cerebras)

[GLM-130B](#)

[YaLM](#) (100B) (Yandex)

[UL2 20B](#) (Google)

[PanGu- \$\alpha\$](#)  (200B) (Huawei)

Cohere (Medium, XLarge)

Claude (instant-v1.0, v1.2) (Anthropic)

[CodeGen](#) (2B, 6B, 16B) (Salesforce)

[NeMo](#) (1.3B, 5B, 20B) (NVIDIA)

RWKV (14B)

BLOOM (1B, 3B, 7B)

GPT-4 (OpenAI)

GPT-3.5 (OpenAI)

GPT-3 (ada, babbage, curie, davinci) (OpenAI)

Codex (cushman, davinci) (OpenAI)

T5 (11B) (Google)

[CPM-Bee](#) (10B)

Fine-tuned models

[Alpaca](#) (7B)

[Convo](#) (6B)

J1-Grande-Instruct (17B) (AI21)

InstructGPT (175B)

BLOOMZ (176B)

Flan-UL2 (20B)

Flan-T5 (11B)

T0 (11B)

Galactica (120B) (Meta)

# Overview of LLMs

GPT-J (6B) (EleutherAI)

GPT-Neo (1.3B, 2.

Pythia (1B, 1.4B, 2

Polyglot (1.3B, 3.8

J1 (7.5B, 17B, 178I

LLaMa (7B, 13B, 3

OPT (1.3B, 2.7B, 1

Fairseq (1.3B, 2.7E

Cerebras-GPT (1.1

GLM-130B

YaLM (100B) (Yan

UL2 20B (Google)

PanGu- $\alpha$  (200B) (Huawei)

Cohere (Medium, XLarge)

Fine-tuned models

Open-Source

AI21 labs Jurassic-1

NAVER LABS HyperCLOVA

LaMDA

BAII CPM-2

AlphaCode

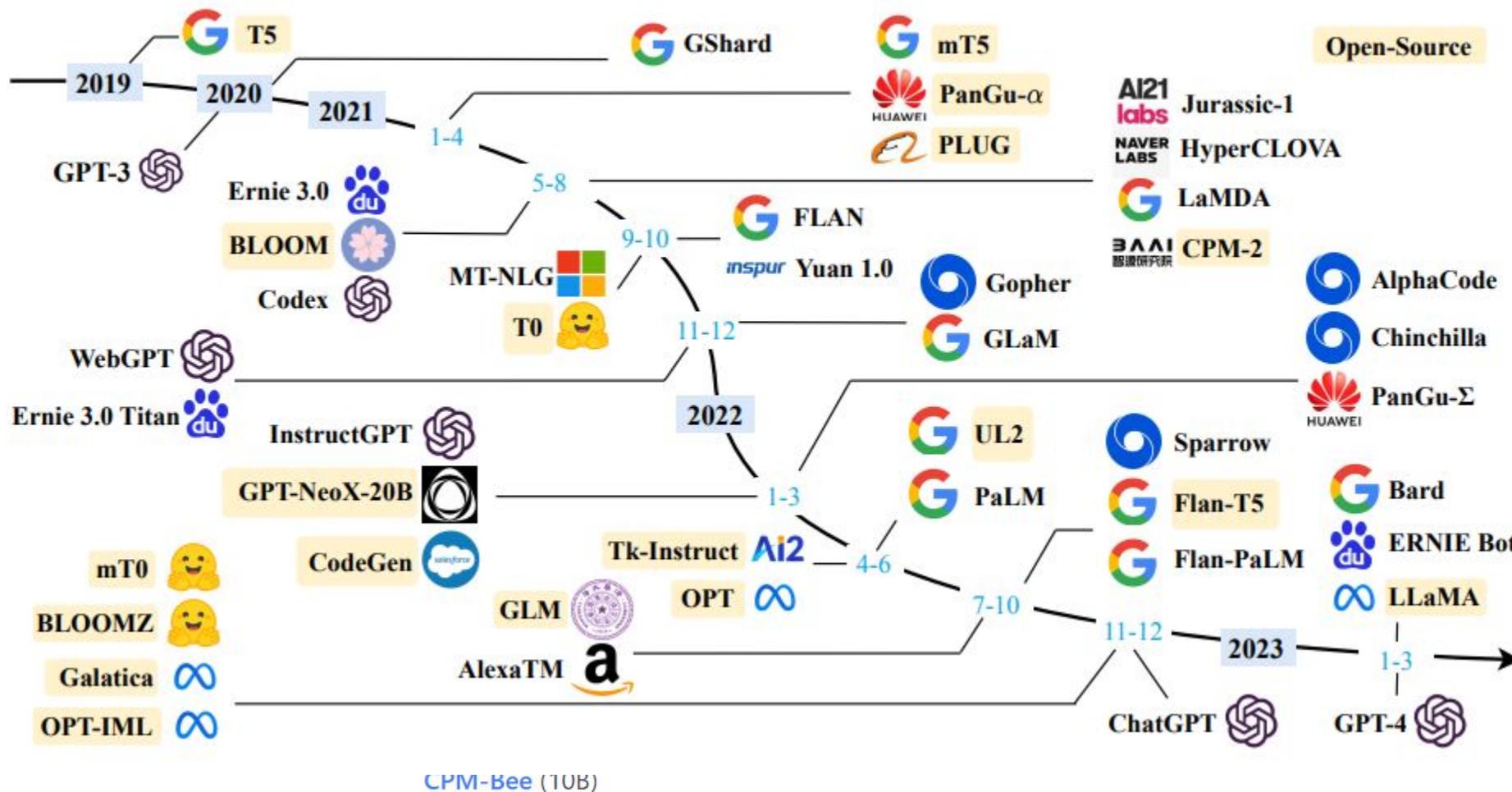
Chinchilla

HUAWEI PanGu- $\Sigma$

Bard

ERNIE Bot

LLaMA



# ChatGPT

- Despite remarkable advancements in NLP introduced by GPT3, it is limited in its ability to align with user intentions:
- **Lack of helpfulness** meaning they do not follow the user's explicit instructions.

# ChatGPT

- Despite remarkable advancements in NLP introduced by GPT3, it is limited in its ability to align with user intentions:
- **Lack of helpfulness** meaning they do not follow the user's explicit instructions.

> quantum chemistry

# ChatGPT

- Despite remarkable advancements in NLP introduced by GPT3, it is limited in its ability to align with user intentions:
- **Lack of helpfulness** meaning they do not follow the user's explicit instructions.

> quantum chemistry

< asked Oct 10, 2018 by Elie (590)

Too many factors can contribute to the question of what is the most probable configuration for a given molecule, but some factors that may influence this include:

- Bonding angles and distances between atoms/molecules
- Electronegativity differences between different elements/atoms
- The polarity of the molecule itself, which can be determined by analyzing its dipole moments
- The electron configuration of each atom/molecule (e.g., electron orbitals)
- Any other factors that may contribute to specific configurations such as steric hindrance or Lewis acid-base interactions.

# ChatGPT

- Despite remarkable advancements in NLP introduced by GPT3, it is limited in its ability to align with user intentions:
- **Lack of helpfulness** meaning they do not follow the user's explicit instructions.
- **Contain hallucinations** that reflect non-existing or incorrect facts.
- **Lack interpretability** making it difficult for humans to understand how the model arrived at a particular decision or prediction.
- **Include toxic or biased content** that is harmful or offensive and spreads misinformation.
- Innovative training methodologies were introduced in ChatGPT to counteract some of these inherent issues of standard LLMs.

# ChatGPT training

Step 1

Collect demonstration data and train a supervised policy.

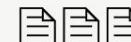
A prompt is sampled from our prompt dataset.



Explain reinforcement learning to a 6 year old.



We give treats and punishments to teach...



This data is used to fine-tune GPT-3.5 with supervised learning.

Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.



Explain reinforcement learning to a 6 year old.



D > C > A > B



D > C > A > B

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



Write a story about otters.



PPO

Once upon a time...



RM

$r_k$

The PPO model is initialized from the supervised policy.

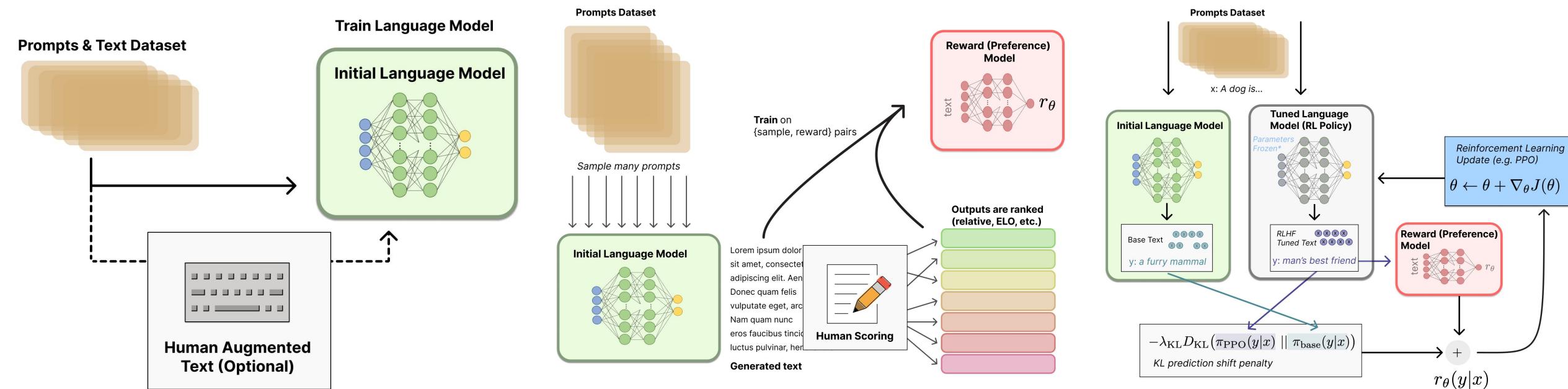
The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

<https://openai.com/blog/chatgpt>

# Reinforcement Learning from Human Feedback



## 4. Final remarks

# Evaluation of the Model

- **Helpfulness:** the model's ability to infer and follow user instructions. Labelers preferred outputs from InstructGPT over GPT-3  $85 \pm 3\%$  of the time.
- **Truthfulness:** the model's tendency for hallucinations. The PPO model produced outputs that showed minor increases in truthfulness and informativeness when assessed using the TruthfulQA dataset.
- **Harmlessness:** the model's ability to avoid inappropriate, derogatory, and denigrating content. Harmlessness was tested using the RealToxicityPrompts dataset. The test was performed under three conditions.
  1. Instructed to provide respectful responses: resulted in a significant decrease in toxic responses.
  2. Instructed to provide responses, without any setting for respectfulness: no significant change in toxicity.
  3. Instructed to provide toxic response: responses were in fact significantly more toxic than the GPT-3 model.
- For more information on the methodologies used in creating ChatGPT and InstructGPT, read the original paper published by OpenAI **Training language models to follow instructions with human feedback, 2022** <https://arxiv.org/pdf/2203.02155.pdf>.

# Limitations

In the process of aligning language models with human intentions, the fine-tuning data is influenced by subjective factors:

- The preferences of the labelers who produce the demonstration data.
- The researchers who design the study and write the labeling instructions.
- The choice of prompts crafted by the developers or provided by the OpenAI customers.
- The labelers bias is both included in the reward model training (by ranking outputs) and in the model evaluation.

# Why do they do what they do?

Fu, Yao; Peng, Hao and Khot, Tushar. (Dec 2022). How does GPT Obtain its Ability? Tracing Emergent Abilities of Language Models to their Sources. Yao Fu's Notion.

In this post, we scrutinize the spectrum of abilities of the GPT-3.5 series and trace back to the sources of all their emergent abilities. The initial GPT-3 model gains its generation ability, world knowledge, and in-context learning from pretraining. Then the instruction tuning branch gains the ability to follow instructions and generalization to unseen tasks. The training on code branch gains the ability of code understanding, and potentially the side product of complex reasoning. Combining the two branches, code-davinci-002 seems to be the most capable GPT-3.5 model with all the powerful abilities. The following supervised instruction tuning and RLHF trades model ability for alignment with humans, i.e., the alignment tax. RLHF enables the model to generate more informative and impartial answers while rejecting questions outside its knowledge scope.

We hope this article can help provide a clear picture of the evaluation of GPT, and stir some discussion about language models, instruction tuning, and code tuning. Most importantly, **we hope this article can serve as the roadmap for reproducing GPT-3.5 within the open-source community.**

# Why do they do what they do?

- Building distinct representation spaces by MHA:
  - Not maintained across layers
- Being constructed in residual fashion, representations evolve along with depth
  - Voita et al., The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives
- Role of positional embedding?
  - Positional encoding at a range of frequencies might modulate information flow built by the model, stratifying the representation dimensions.

# 5. Demos

# Demo of LLamA C++ / GPT4All

<https://github.com/nomic-ai/gpt4all>

~8B params

b'tok\_embeddings.weight' with shape: [32001, 4096] and type: Q4\_0

b'layers.0-31.attention.wq.weight' with shape: [4096, 4096] and type: Q4\_0  
b'layers.0-31.attention.wk.weight' with shape: [4096, 4096] and type: Q4\_0  
b'layers.0-31.attention.wv.weight' with shape: [4096, 4096] and type: Q4\_0  
b'layers.0-31.attention.wo.weight' with shape: [4096, 4096] and type: Q4\_0

b'layers.0-31.feed\_forward.w1.weight' with shape: [11008, 4096] and type: Q4\_0  
b'layers.0-31.feed\_forward.w2.weight' with shape: [4096, 11008] and type: Q4\_0  
b'layers.0-31.feed\_forward.w3.weight' with shape: [11008, 4096] and type: Q4\_0

b'layers.0-31.attention\_norm.weight' with shape: [4096] and type: F32  
b'layers.0-31.ffn\_norm.weight' with shape: [4096] and type: F32

b'norm.weight' with shape: [4096] and type: F32

b'output.weight' with shape: [32001, 4096] and type: Q4\_0

# Backup

## Summary

ChatGPT is the fine-tuning of GPT-3.5, which is a language model based on a Transformer decoder with some modifications with respect to the original Transformer architecture. Therefore it is a decoder-only model.

Complete information with references

The origin of ChatGPT was GPT (Generative pre-Trained Transformer). The evolution from GPT to ChatGPT was as follows:

- **GPT** (see the OpenAI announcement) was a normal Transformer decoder. From the GPT paper:

In our experiments, we use a multi-layer Transformer decoder [34] for the language model [...]

- **GPT-2** (see the OpenAI announcement and the source code) is also a Transformer decoder, but with some modifications. It is also bigger and trained on more data. From the GPT-2 paper: We use a Transformer (Vaswani et al., 2017) based architecture for our LMs. The model largely follows the details of the OpenAI GPT model (Radford et al., 2018) with a few modifications. Layer normalization (Ba et al., 2016) was moved to the input of each sub-block, similar to a pre-activation residual network (He et al., 2016) and an additional layer normalization was added after the final self-attention block. A modified initialization which accounts for the accumulation on the residual path with model depth is used. We scale the weights of residual layers at initialization by a factor of  $1/N - \sqrt{1/N}$  where  $N$  is the number of residual layers. The vocabulary is expanded to 50,257. We also increase the context size from 512 to 1024 tokens and a larger batch size of 512 is used.

- **GPT-3** is GPT-2 scaled up and with some modifications. From the GPT-3 paper published at NeurIPS'20:

We use the same model and architecture as GPT-2 [RWC+19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3.

- **InstructGPT** (see the paper) is a fine-tuned version of GPT-3. From the paper

[...] we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning

- **GPT-3.5** (see OpenAI announcement) (text-davinci-003) is a fine-tuned version of InstructGPT. From the announcement:

code-davinci-002 is a base model, so good for pure code-completion tasks

text-davinci-002 is an InstructGPT model based on code-davinci-002

text-davinci-003 is an improvement on text-davinci-002

- **ChatGPT** (get-3.5-turbo\*) is a GPT-3.5 fine-tuned on human instructions by Reinforcement Learning with Human Feedback (RLHF). From the OpenAI website:

gpt-3.5-turbo-0301 is an improvement on text-davinci-003, optimized for chat

From the ChatGPT presentation page:

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using Proximal Policy Optimization. We performed several iterations of this process.

To follow in detail the GPT evolution, I recommend the article How does GPT Obtain its Ability? Tracing Emergent Abilities of Language Models to their Sources.

# Attention plots

## Attention Visualizations

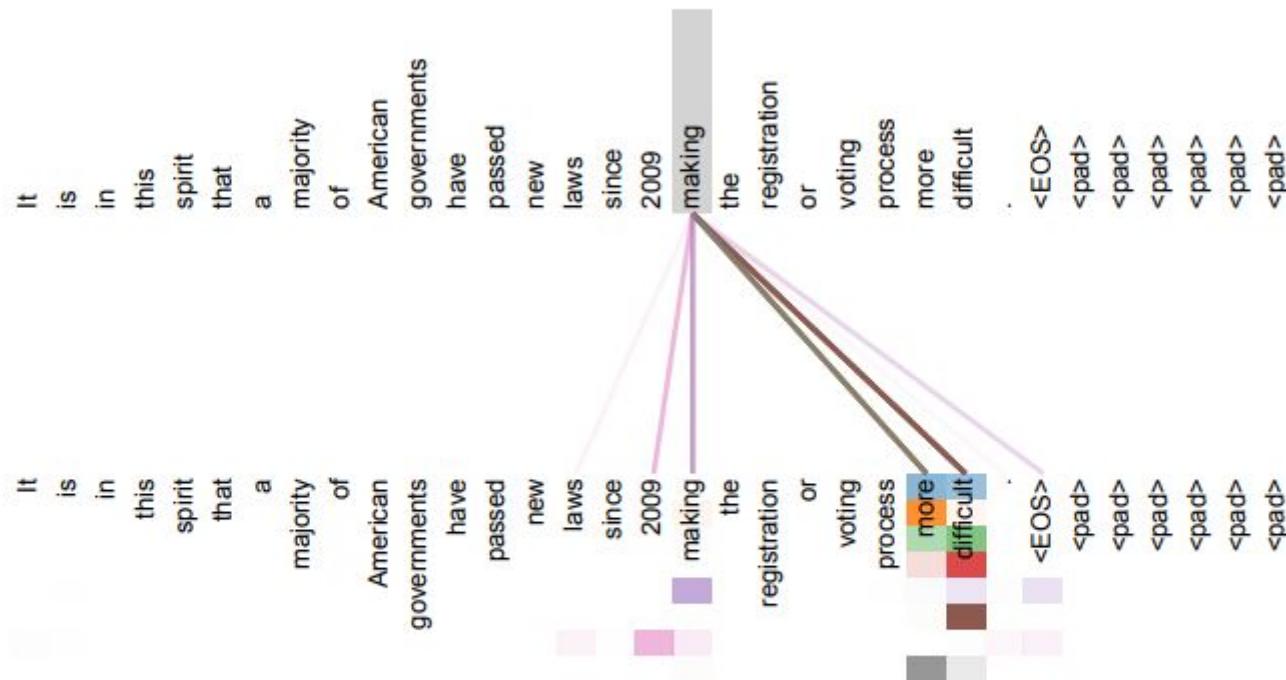


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

# Attention plots

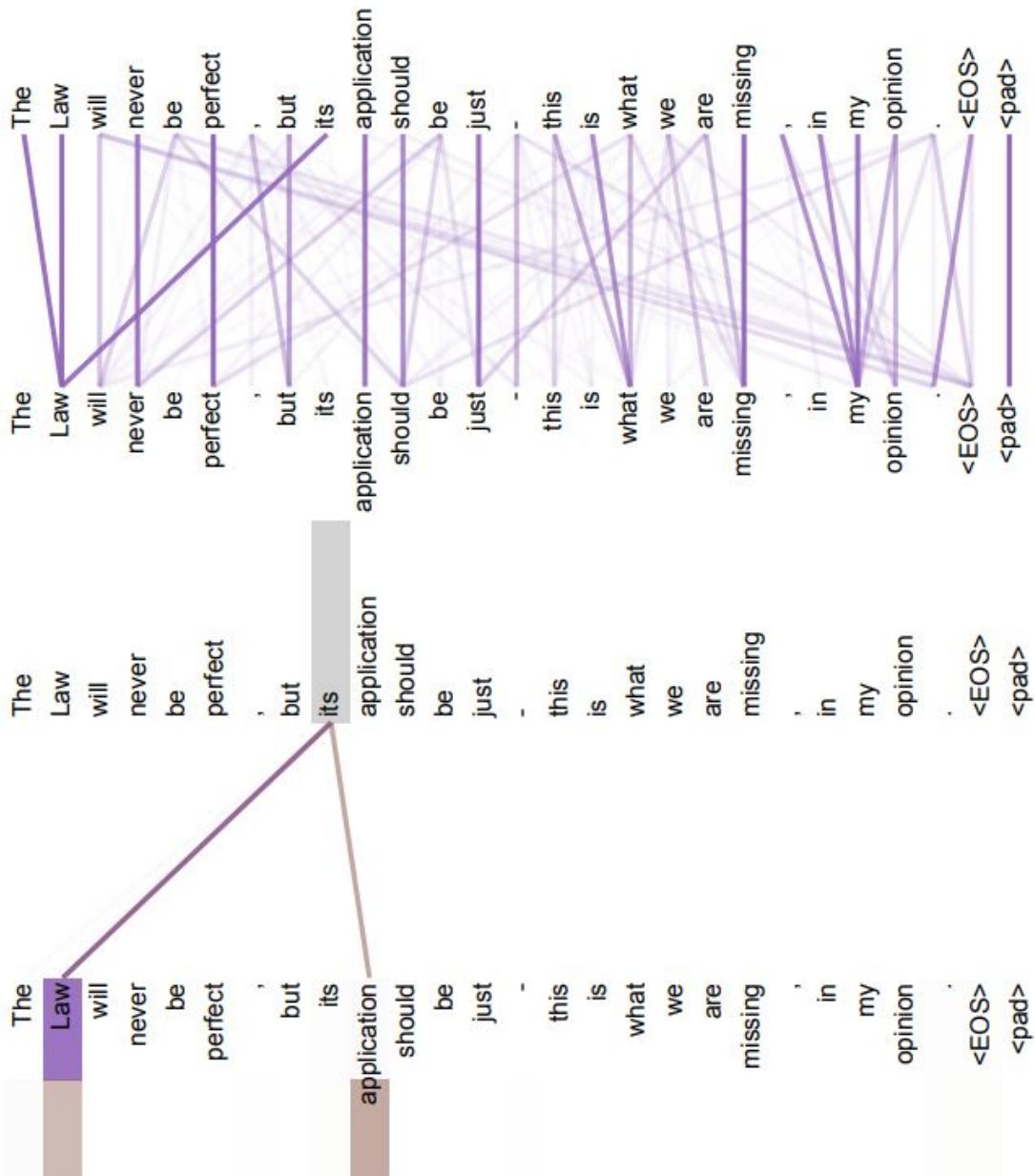


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

# Attention plots

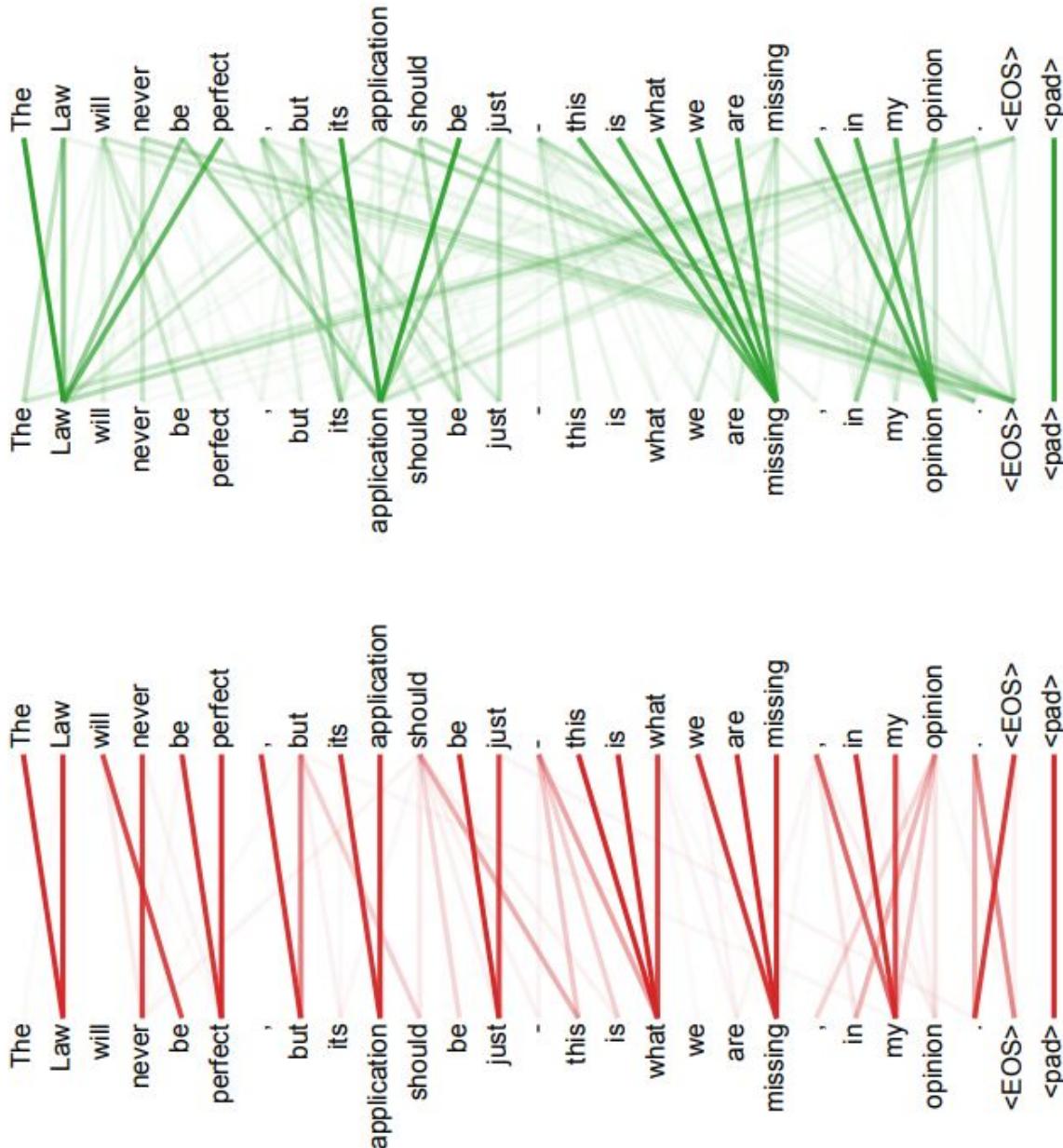
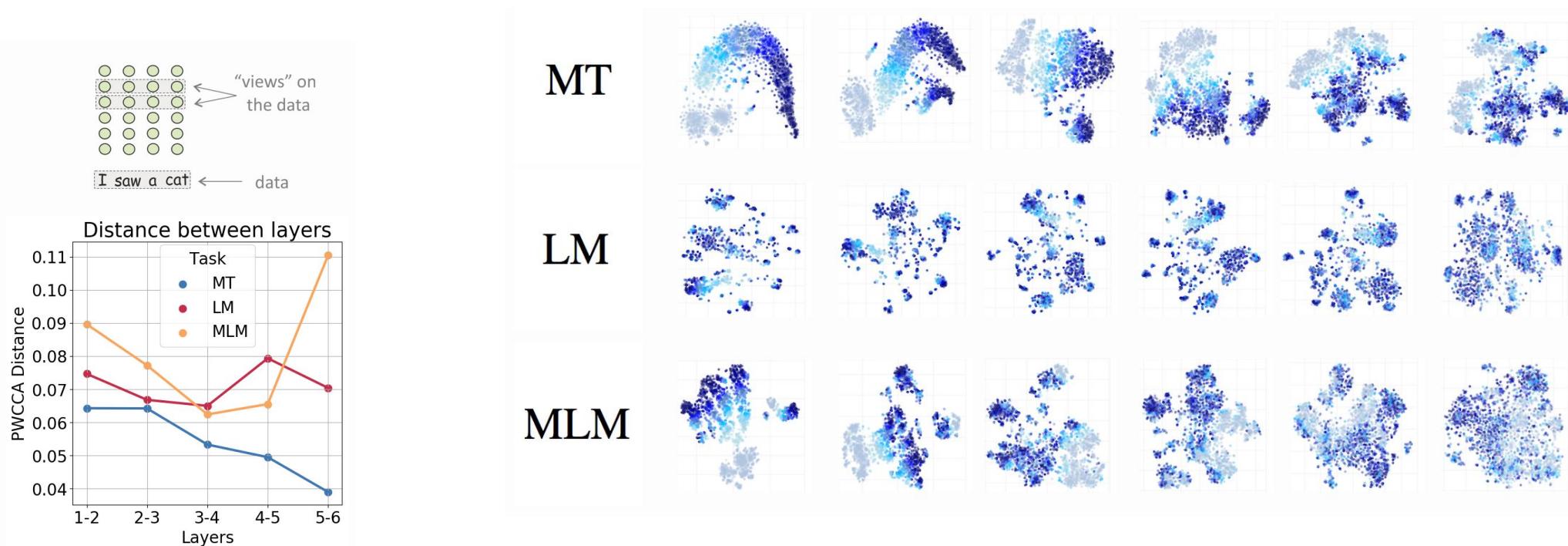


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

# State evolution demo



[https://lena-voita.github.io/posts/emnlp19\\_evolution.html](https://lena-voita.github.io/posts/emnlp19_evolution.html)

```
from pyllamacpp.model import Model

def new_text_callback(text: str):
    print(text, end="", flush=True)

model = Model(ggml_model='./models/gpt4all-model.bin', n_ctx=512)
res = model.generate("Once upon a time, ", n_predict=1000, new_text_callback=new_text_callback,
n_threads=8)

print(res)
```