

Deep Learning

Lecture 8: Attention and transformers

Prof. Gilles Louppe
g.louppe@uliege.be

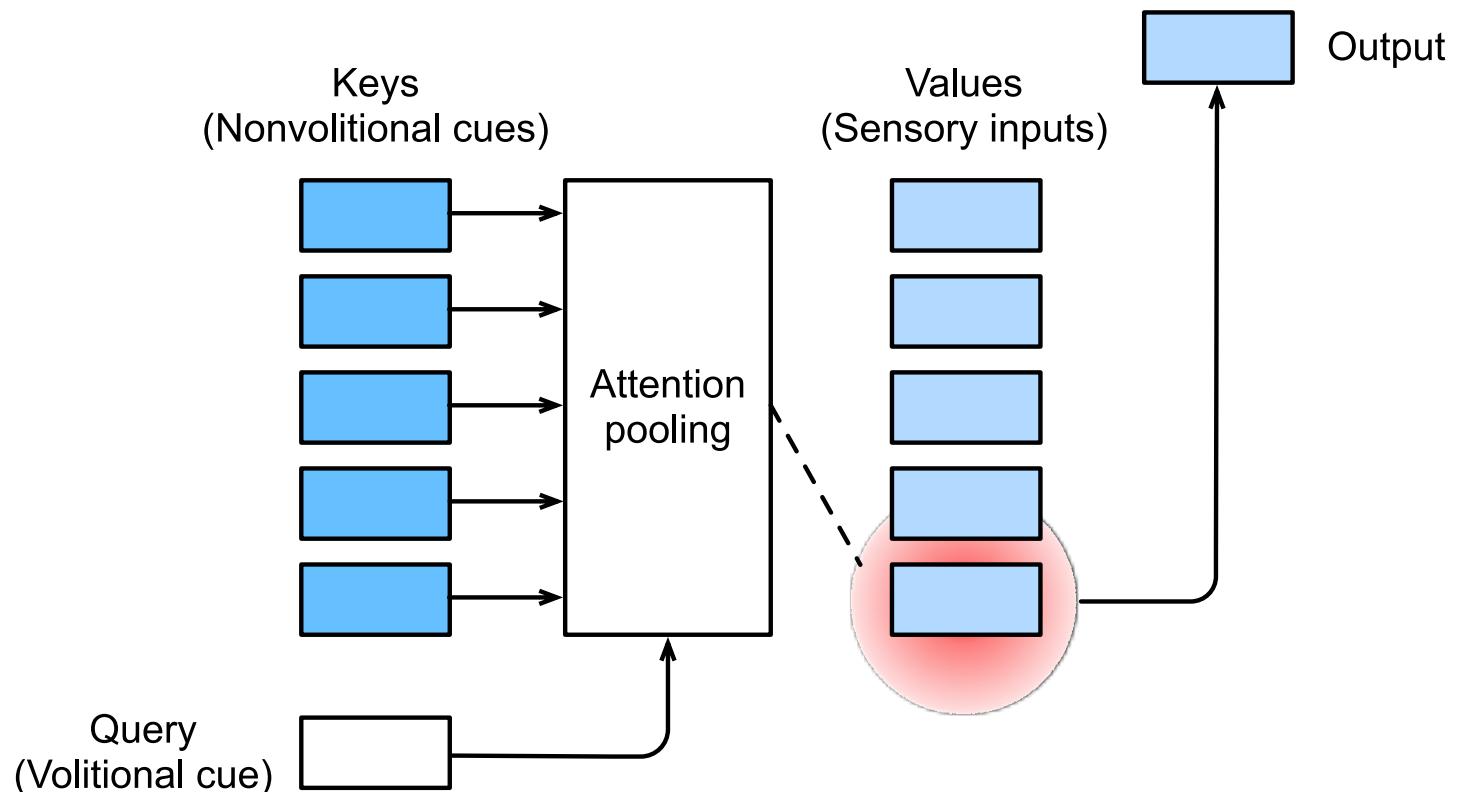


Today

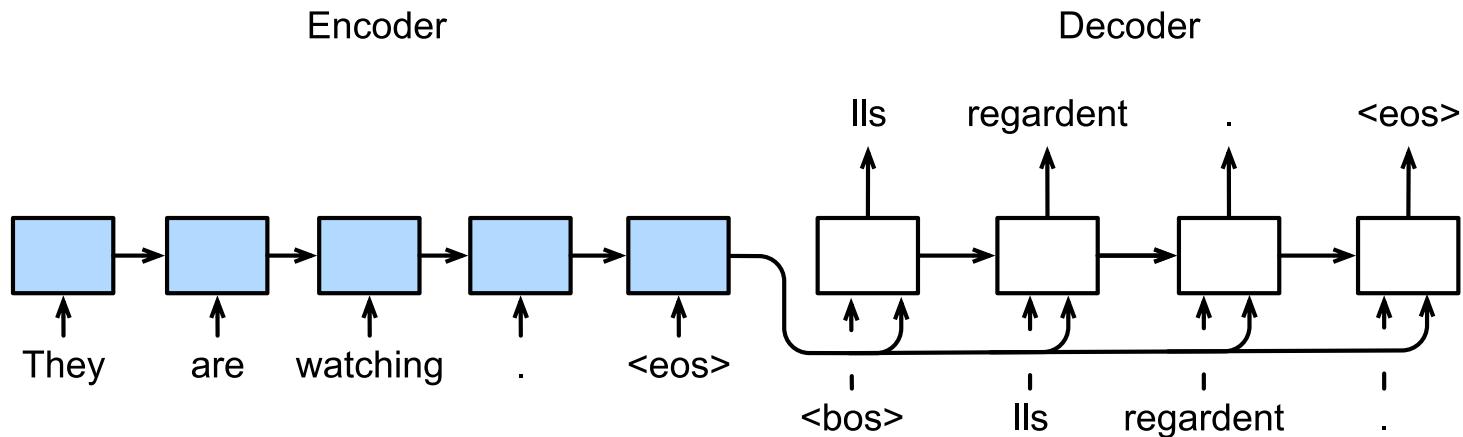
Attention is all you need!

- Bahdanau attention
- Attention layers
- Transformers
- Applications





Bahdanau attention

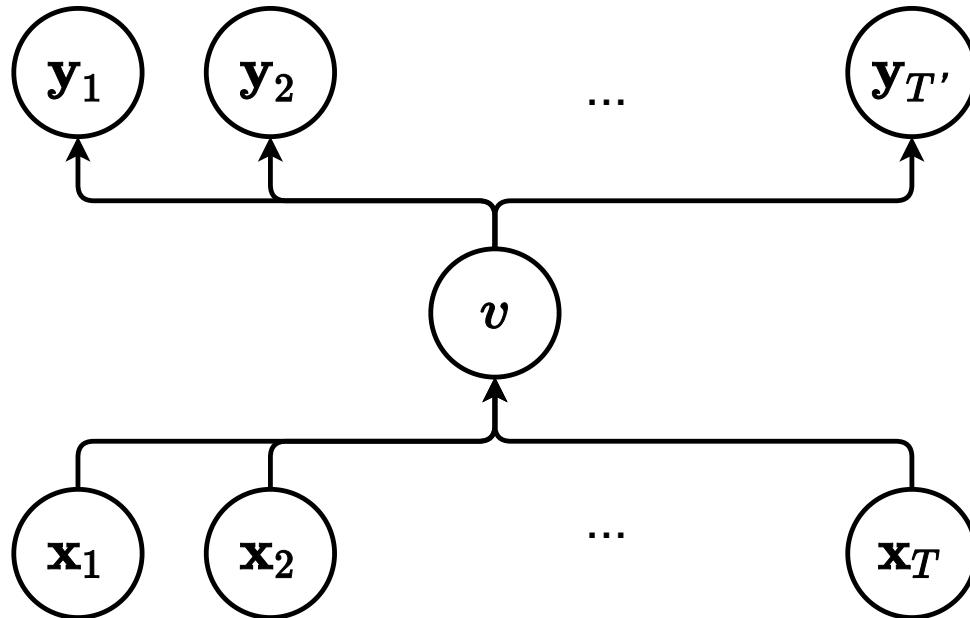


Standard RNN-based sequence-to-sequence models compress an input sequence $\mathbf{x}_{1:T}$ into a single thought vector v , and then produce an output sequence $\mathbf{y}_{1:T'}$ from an autoregressive generative model

$$\mathbf{h}_t = \phi(\mathbf{x}_t, \mathbf{h}_{t-1})$$

$$v = \mathbf{h}_T$$

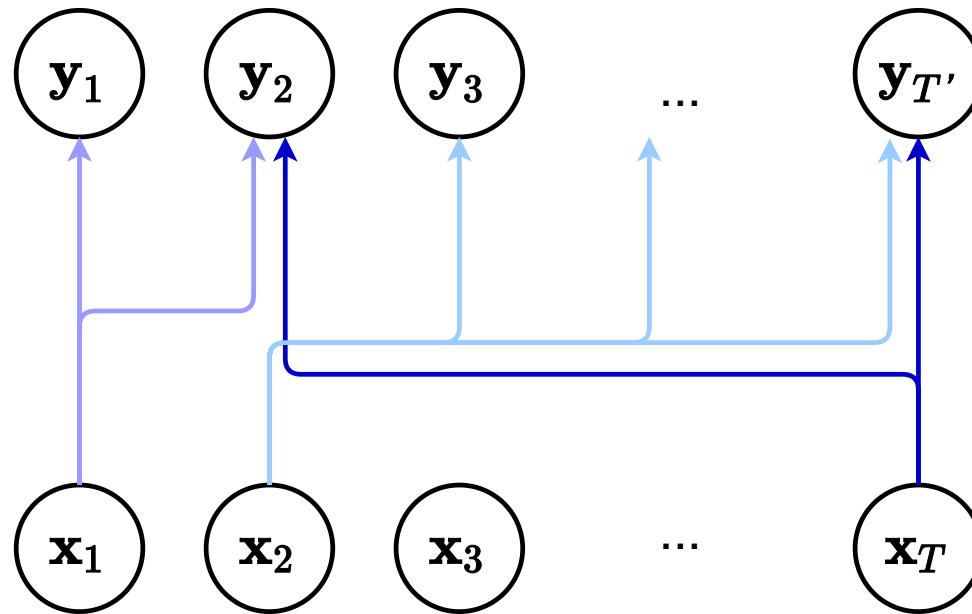
$$\mathbf{y}_i \sim p(\cdot | \mathbf{y}_{1:i-1}, v).$$



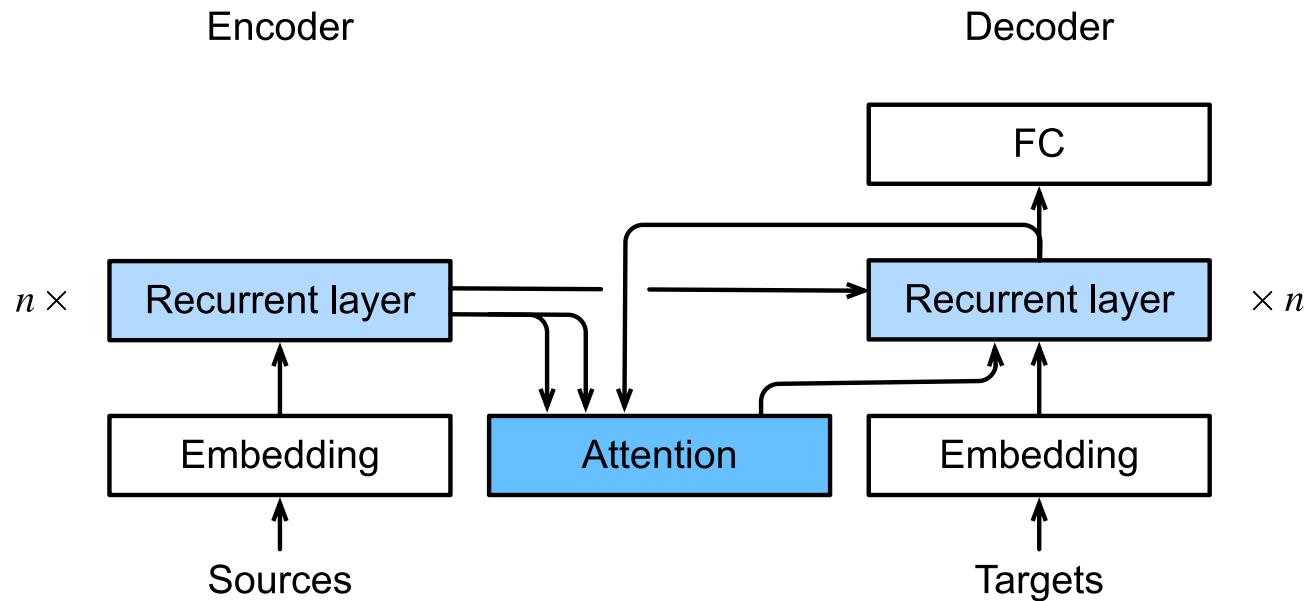
This architecture assumes that the sole vector v carries enough information to generate entire output sequences. This is often **challenging** for long sequences.

Instead, attention mechanisms can transport information from parts of the input signal to parts of the output **specified dynamically**.

Under the assumption that each output token comes from one or a handful of input tokens, the decoder should attend to only those tokens that are relevant for producing the next output token.



Attention-based machine translation



Following Bahdanau et al. (2014), the encoder is specified as a bidirectional RNN that computes an annotation vector for each input token,

$$\mathbf{h}_i = (\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i)$$

for $i = 1, \dots, T$, where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ respectively denote the forward and backward hidden recurrent states of the bidirectional RNN.

From this, they compute a new process $\mathbf{s}_i, i = 1, \dots, T$, which looks at weighted averages of the $\mathbf{h}_j, j = 1, \dots, T$, where the **weights are functions of the signal**.

Given $\mathbf{s}_1, \dots, \mathbf{s}_{i-1}$, first compute an attention vector

$$\alpha_{i,j} = \text{softmax}_j(\mathbf{e}_{i,j})$$

for $j = 1, \dots, T$, where

$$\mathbf{e}_{i,j} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$$

and a is an [attention scoring function](#), here specified as a one hidden layer [tanh](#) MLP.

Then, compute the context vector from the weighted \mathbf{h}_j 's,

$$\mathbf{c}_i = \sum_{j=1}^T \alpha_{i,j} \mathbf{h}_j.$$

The model can now make the prediction \mathbf{y}_i as

$$\begin{aligned}\mathbf{s}_i &= f(\mathbf{s}_{i-1}, y_{i-1}, c_i) \\ \mathbf{y}_i &\sim g(\mathbf{y}_{i-1}, \mathbf{s}_i, \mathbf{c}_i),\end{aligned}$$

where f is a GRU.

This is **context attention**, where \mathbf{s}_{i-1} modulates what to look in $\mathbf{h}_1, \dots, \mathbf{h}_T$ to compute \mathbf{s}_i and sample \mathbf{y}_i .

(whiteboard example)

L' accord sur la zone économique européenne a été signé en août 1992.

The agreement on the European Economic Area was signed in August 1992.

. <end>

(a)

Il convient de noter que l'environnement marin est le moins connu de l'environnement.

It should be noted that the marine environment is the least known of environments.

. <end>

(b)

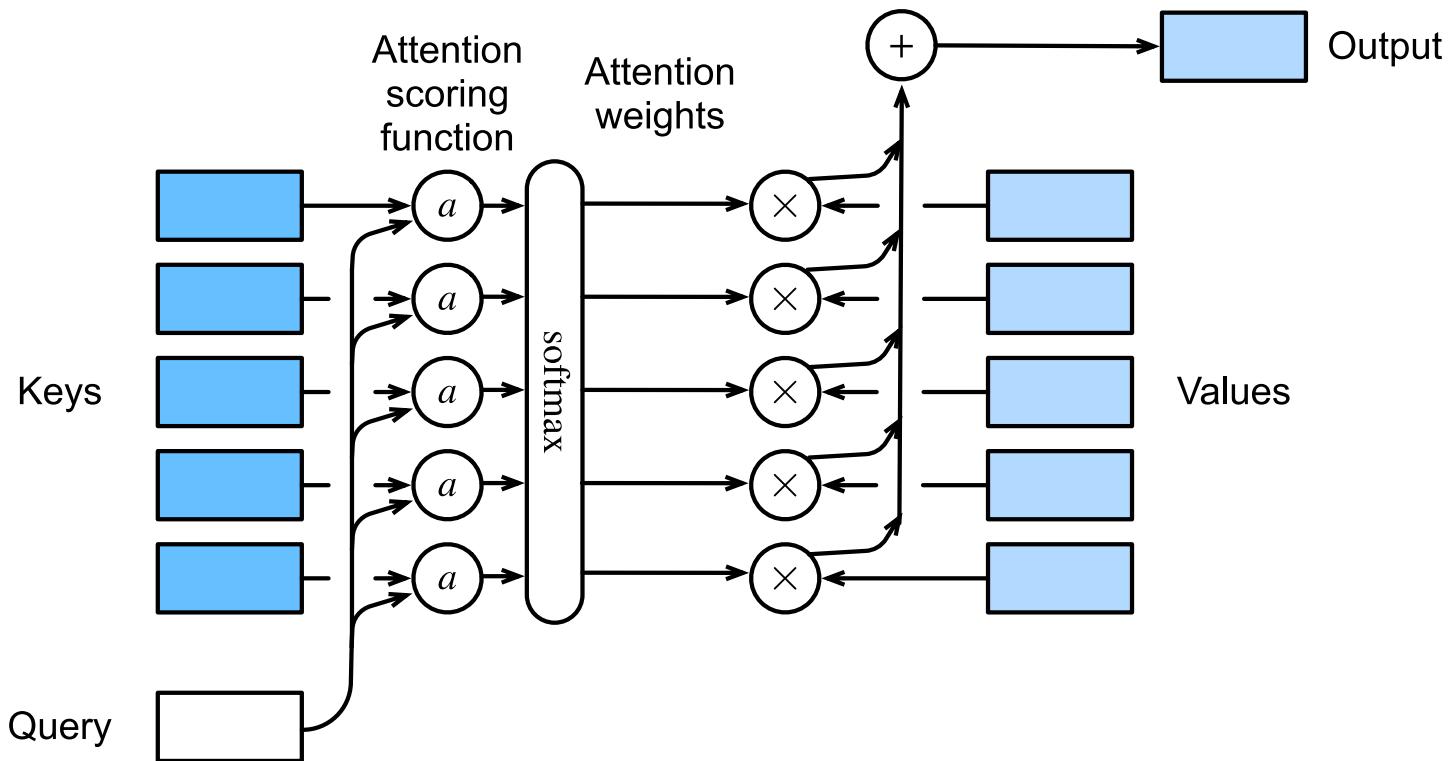
Attention layers

The attention mechanisms can be defined generically as follows.

Given a context or query vector $\mathbf{q} \in \mathbb{R}^q$, a key tensor $\mathbf{K} \in \mathbb{R}^{m \times k}$, and a value tensor $\mathbf{V} \in \mathbb{R}^{m \times v}$, an attention layer computes an output vector $\mathbf{y} \in \mathbb{R}^v$ with

$$\mathbf{y} = \sum_{i=1}^m \text{softmax}_i(a(\mathbf{q}, \mathbf{K}_i; \theta)) \mathbf{V}_i,$$

where $a : \mathbb{R}^q \times \mathbb{R}^k \rightarrow \mathbb{R}$ is a scalar attention scoring function.



Additive attention

When queries and keys are vectors of different lengths, we can use an additive attention as the scoring function.

Given $\mathbf{q} \in \mathbb{R}^q$ and $\mathbf{k} \in \mathbb{R}^k$, the **additive attention** scoring function is

$$a(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^T \tanh(\mathbf{W}_q^T \mathbf{q} + \mathbf{W}_k^T \mathbf{k})$$

where $\mathbf{w}_v \in \mathbb{R}^h$, $\mathbf{W}_q \in \mathbb{R}^{q \times h}$ and $\mathbf{W}_k \in \mathbb{R}^{k \times h}$ are learnable parameters.

Scaled dot-product attention

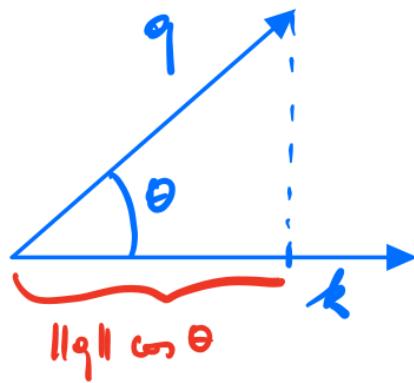
When queries and keys are vectors of the same length d , we can use a scaled dot-product attention as the scoring function.

Given $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{k} \in \mathbb{R}^d$, the **scaled dot-product attention** scoring function is

$$a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T \mathbf{k}}{\sqrt{d}}.$$

For n queries $\mathbf{Q} \in \mathbb{R}^{n \times d}$, keys $\mathbf{K} \in \mathbb{R}^{m \times d}$ and values $\mathbf{V} \in \mathbb{R}^{m \times v}$, the scaled dot-product attention layer computes an output tensor

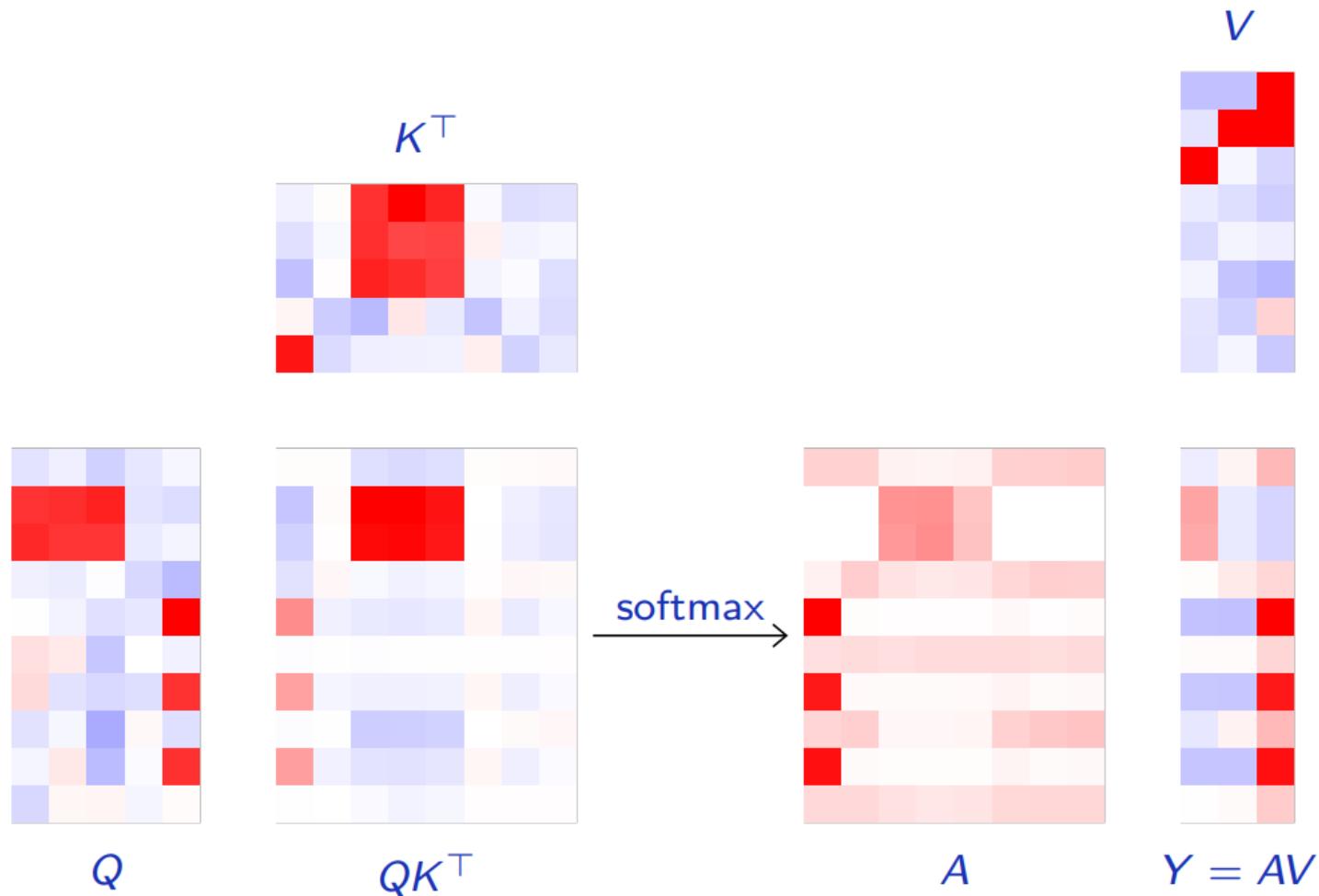
$$\mathbf{Y} = \underbrace{\text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} \right)}_{\text{attention matrix } \mathbf{A}} \mathbf{V} \in \mathbb{R}^{n \times v}.$$



$$q^T k = \|q\| \|k\| \cos \theta$$

Recall that the dot product is simply a un-normalised cosine similarity, which tells us about the alignment of two vectors.

Therefore, the \mathbf{QK}^T matrix is a **similarity matrix** between queries and keys.



In the currently standard models for sequences, the queries, keys and values are linear functions of the inputs.

Given the learnable matrices $\mathbf{W}_q \in \mathbb{R}^{d \times x}$, $\mathbf{W}_k \in \mathbb{R}^{d \times x'}$, and $\mathbf{W}_v \in \mathbb{R}^{v \times x'}$, and two input sequences $\mathbf{X} \in \mathbb{R}^{n \times x}$ and $\mathbf{X}' \in \mathbb{R}^{m \times x'}$, we have

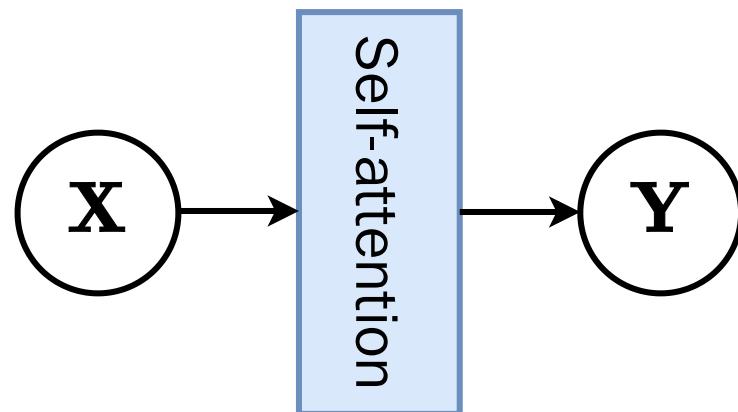
$$\begin{aligned}\mathbf{Q} &= \mathbf{X} \mathbf{W}_q^T \in \mathbb{R}^{n \times d} \\ \mathbf{K} &= \mathbf{X}' \mathbf{W}_k^T \in \mathbb{R}^{m \times d} \\ \mathbf{V} &= \mathbf{X}' \mathbf{W}_v^T \in \mathbb{R}^{m \times v}.\end{aligned}$$

Self-attention

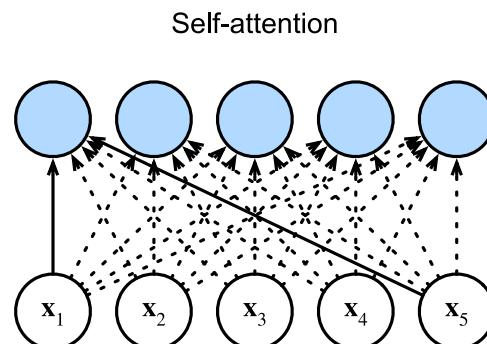
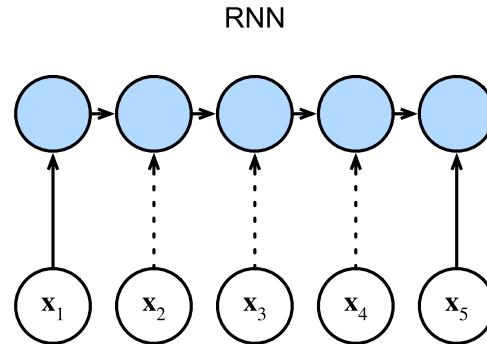
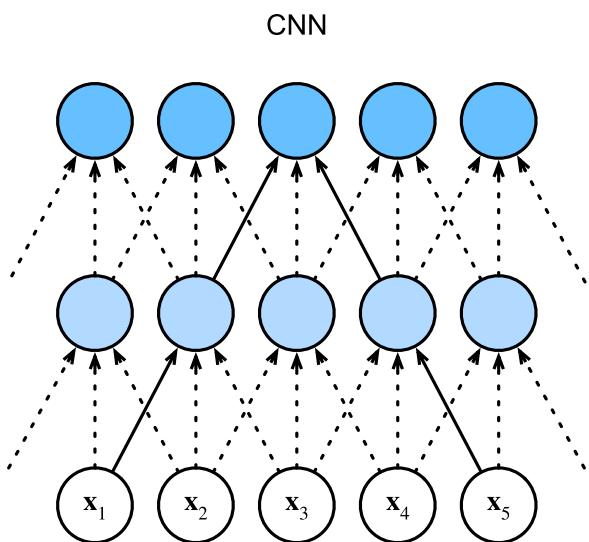
When the queries, keys and values are derived from the same inputs, the attention mechanism is called **self-attention**.

For the scaled dot-product attention, the self-attention layer is obtained when $\mathbf{X} = \mathbf{X}'$.

Therefore, self-attention can be used as a regular feedforward-kind of layer, similarly to fully-connected or convolutional layers.



CNNs vs. RNNs vs. self-attention

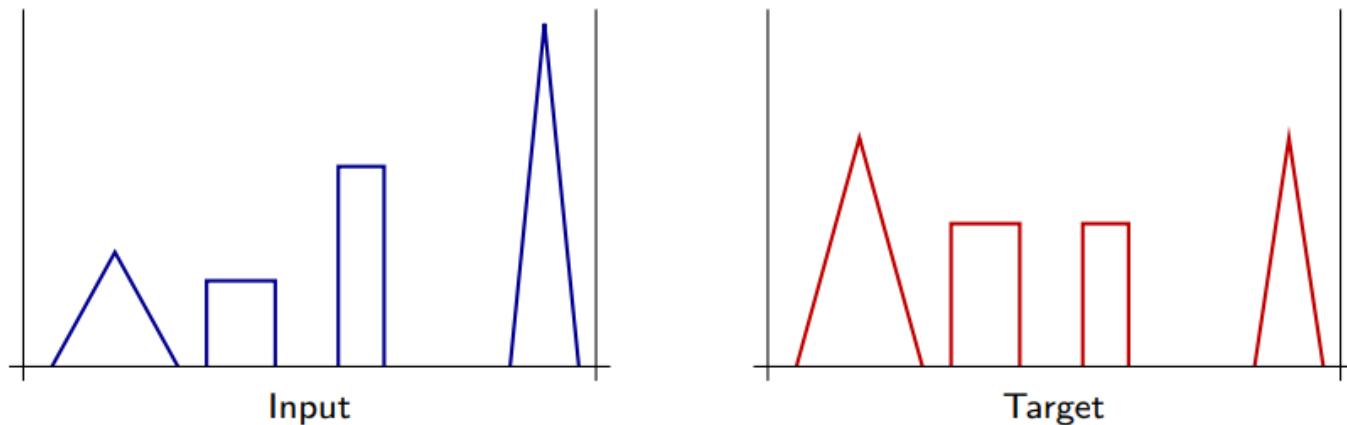


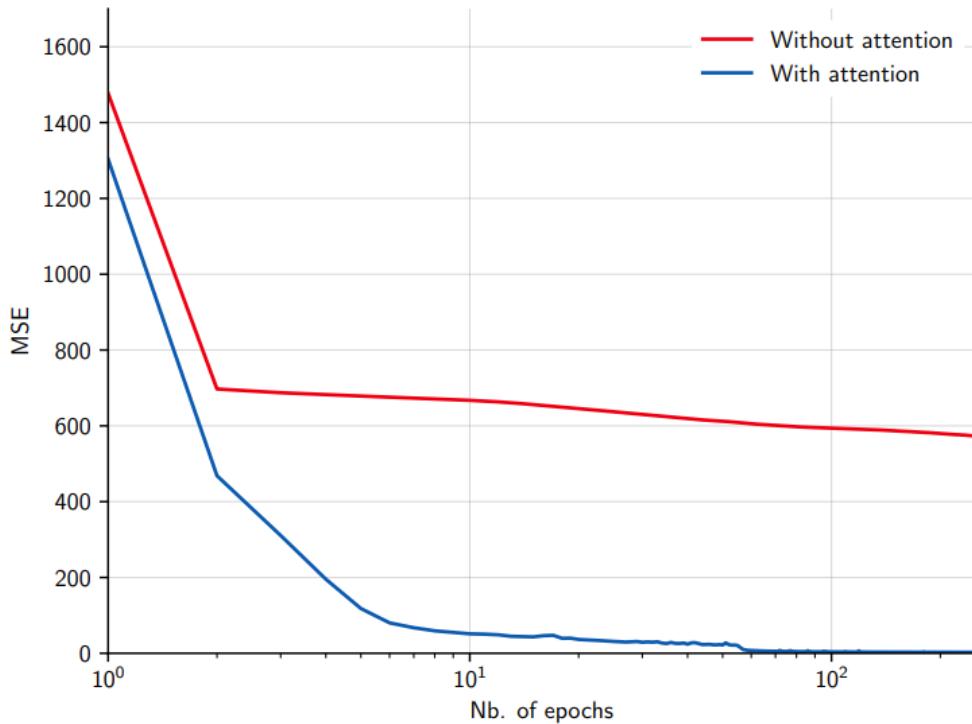
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

where n is the sequence length, d is the embedding dimension, and k is the kernel size of convolutions.

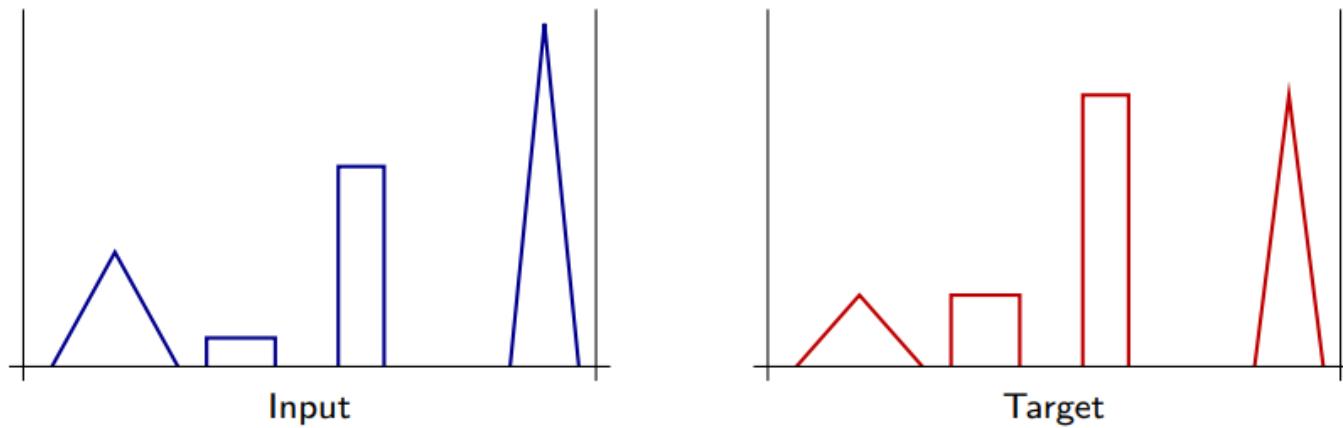
A toy example

To illustrate the behavior of the attention mechanism, we consider a toy problem with 1D sequences composed of two triangular and two rectangular patterns. The target sequence averages the heights in each pair of shapes.





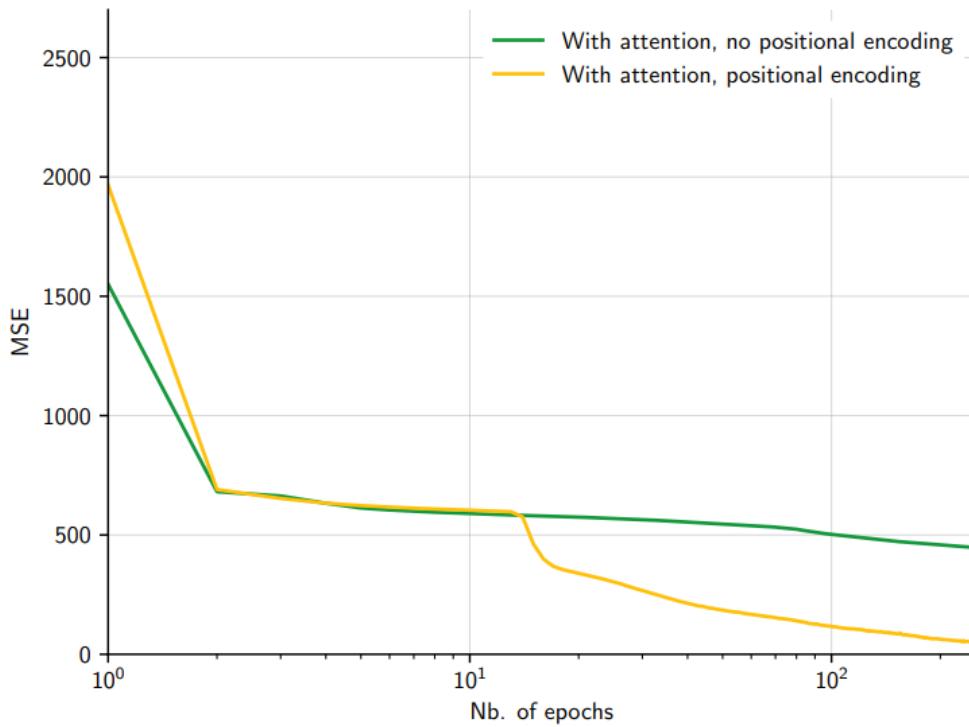
We can modify the toy problem to consider targets where the pairs to average are the two right and leftmost shapes.



The performance is expected to be poor given the inability of the self-attention layer to take into account absolute or relative positions. Indeed, self-attention is permutation-invariant:

$$\begin{aligned}\mathbf{y} &= \sum_{i=1}^m \text{softmax}_i \left(\frac{\mathbf{q}^T \mathbf{K}_i^T}{\sqrt{d}} \right) \mathbf{V}_i \\ &= \sum_{i=1}^m \text{softmax}_{\sigma(i)} \left(\frac{\mathbf{q}^T \mathbf{K}_{\sigma(i)}^T}{\sqrt{d}} \right) \mathbf{V}_{\sigma(i)}\end{aligned}$$

for any permutation σ of the key-value pairs.



However, this problem can be fixed by providing positional encodings explicitly to the attention layer.

Transformers

Vaswani et al. (2017) proposed to go one step further: instead of using attention mechanisms as a supplement to standard convolutional and recurrent layers, they designed a model, the **transformer**, combining only attention layers.

The transformer was designed for a sequence-to-sequence translation task, but it is currently key to state-of-the-art approaches across NLP tasks.

Scaled dot-product attention

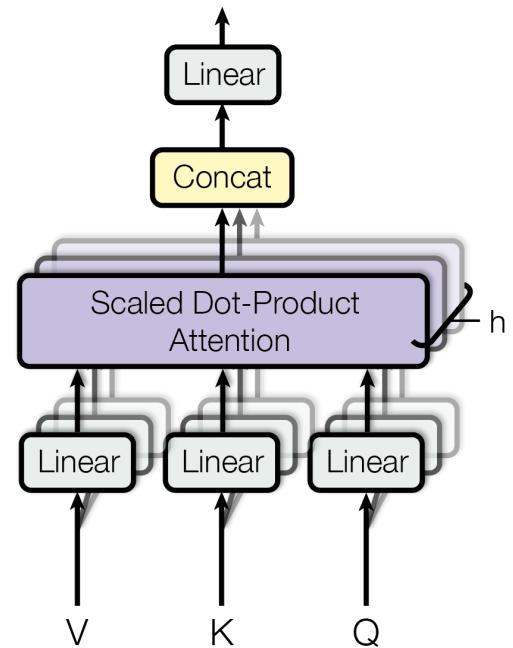
The first building block of the transformer architecture is a scaled dot-production attention module

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

where the $1/\sqrt{d_k}$ scaling is used to keep the (softmax's) temperature constant across different choices of the query/key dimension d_k .

Multi-head attention

The transformer architecture projects the queries, keys and values $h = 8$ times with distinct, learned linear projections to $d_k = 64$, $d_k = 64$ and $d_v = 64$ dimensions respectively.



$$\text{multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat} (\mathbf{H}_1, \dots, \mathbf{H}_h) \mathbf{W}^O$$

$$\mathbf{H}_i = \text{attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

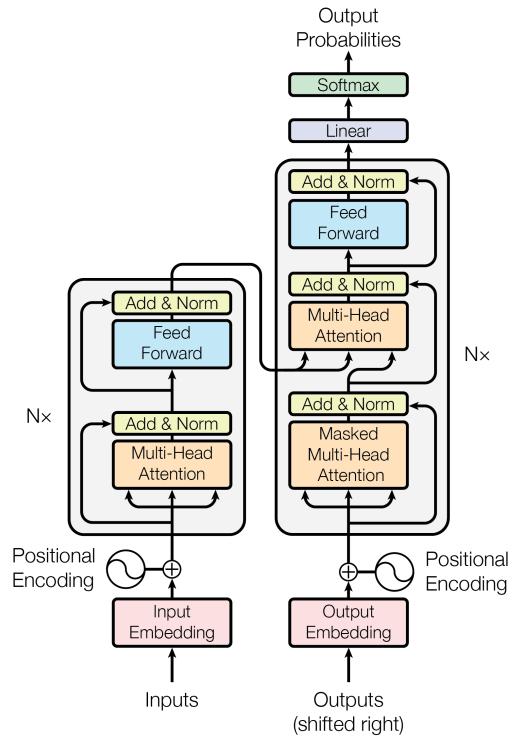
with

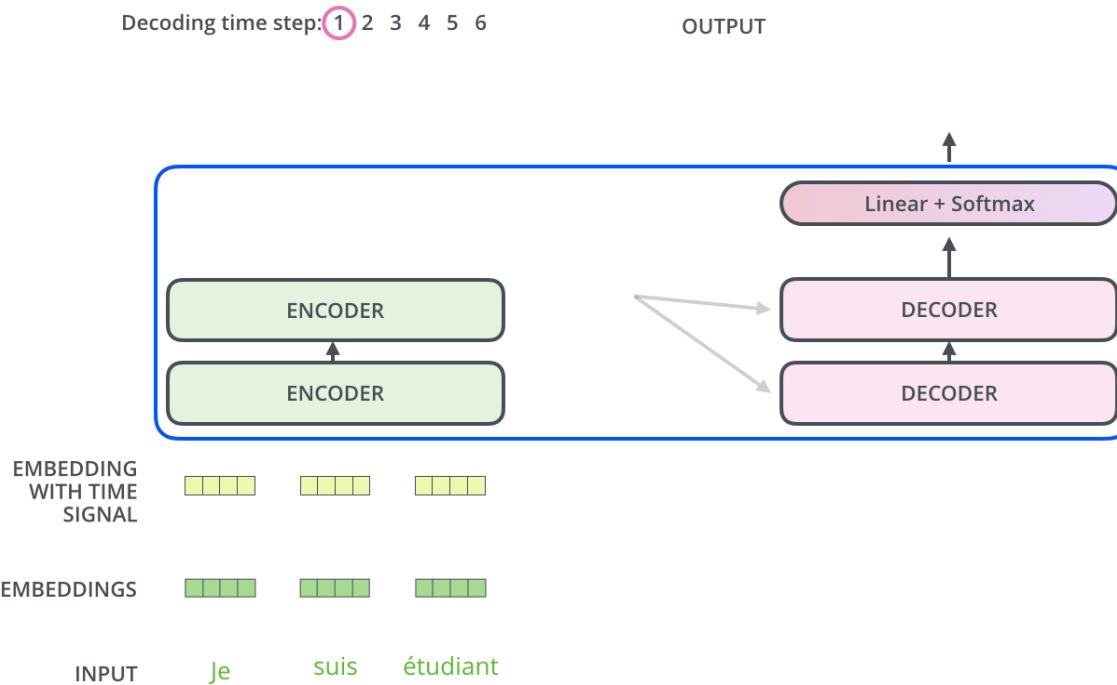
$$\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, \mathbf{W}_i^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

Encoder-decoder architecture

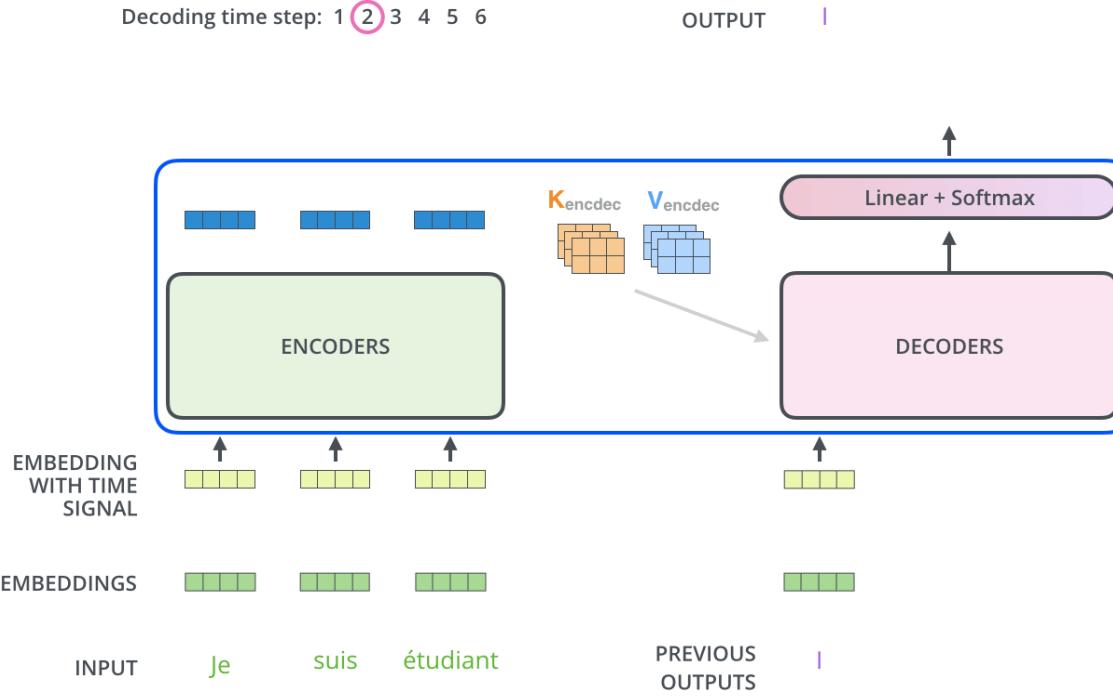
The transformer model is composed of:

- An encoder that combines $N = 6$ modules, each composed of a multi-head attention submodule, and a (per-component) one-hidden-layer MLP, with residual pass-through and layer normalization. All sub-modules and embedding layers produce outputs of dimension $d_{\text{model}} = 512$.
- A decoder that combines $N = 6$ modules similar to the encoder, but using masked self-attention to prevent positions from attending to subsequent positions. In addition, the decoder inserts a third sub-module which performs multi-head attention over the output of the encoder stack.





The encoders start by processing the input sequence. The output of the top encoder is then transformed into a set of attention vectors **K** and **V** that will help the decoders focus on appropriate places in the input sequence.



Each step in the decoding phase produces an output token, until a special symbol is reached indicating the transformer decoder has completed its output.

The output of each step is fed to the bottom decoder in the next time step, and the decoders bubble up their decoding results just like the encoders did.

In the decoder:

- The first masked self-attention sub-module is only allowed to attend to earlier positions in the output sequence. This is done by masking future positions.
- The second multi-head attention sub-module works just like multi-head self-attention, except it creates its query matrix from the layer below it, and takes the keys and values matrices from the output of the encoder stack.

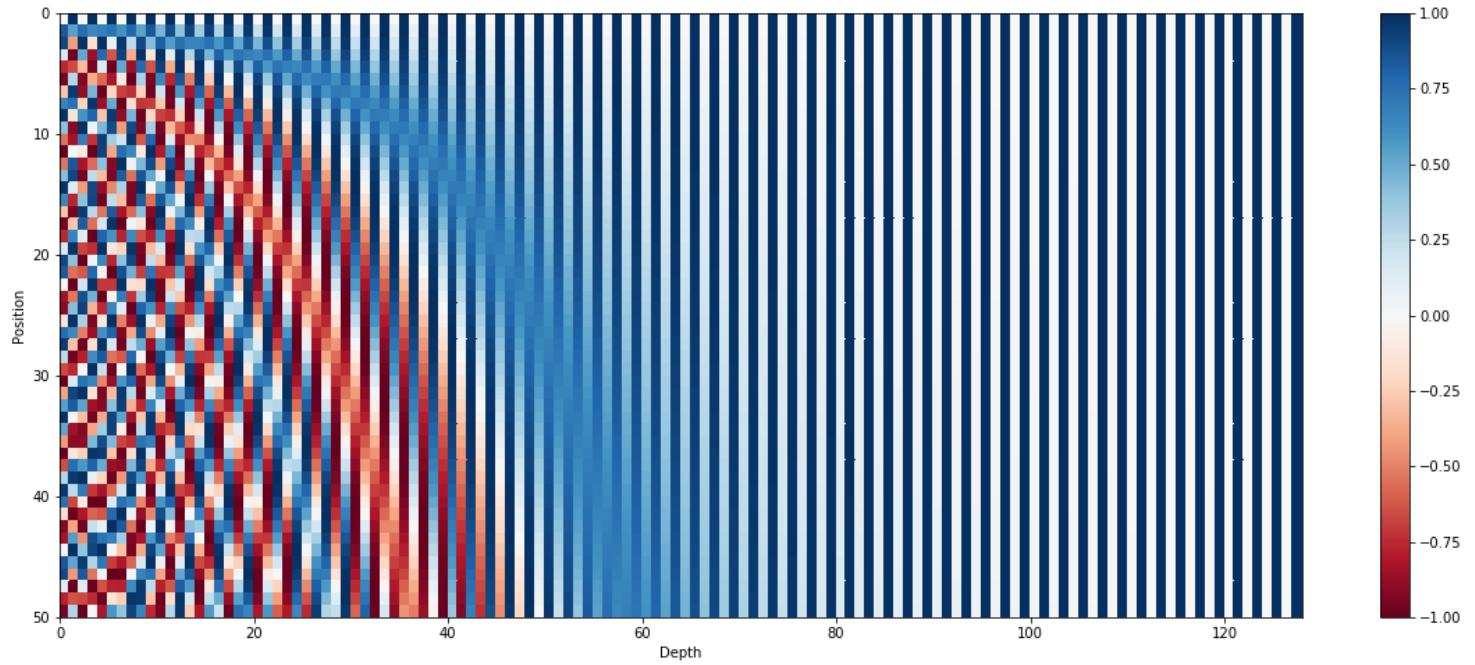
Positional encoding

As each word in a sentence *simultaneously* flows through the encoder/decoder stack, the model itself does not have any sense of position/order for each word.

Positional information is provided through an **additive** positional encoding of the same dimension d_{model} as the internal representation and is of the form

$$\begin{aligned} \text{PE}_{t,2i} &= \sin \left(\frac{t}{10000^{\frac{2i}{d_{\text{model}}}}} \right) \\ \text{PE}_{t,2i+1} &= \cos \left(\frac{t}{10000^{\frac{2i}{d_{\text{model}}}}} \right). \end{aligned}$$

After adding the positional encoding, words will be closer to each other based on the similarity of their meaning and their relative position in the sentence, in the d_{model} -dimensional space.

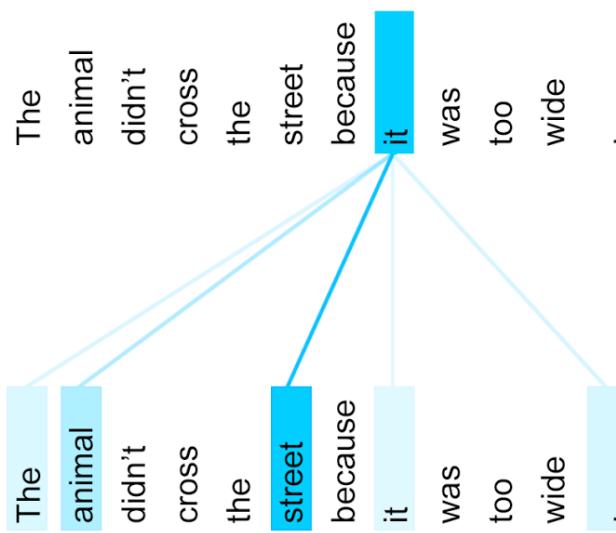
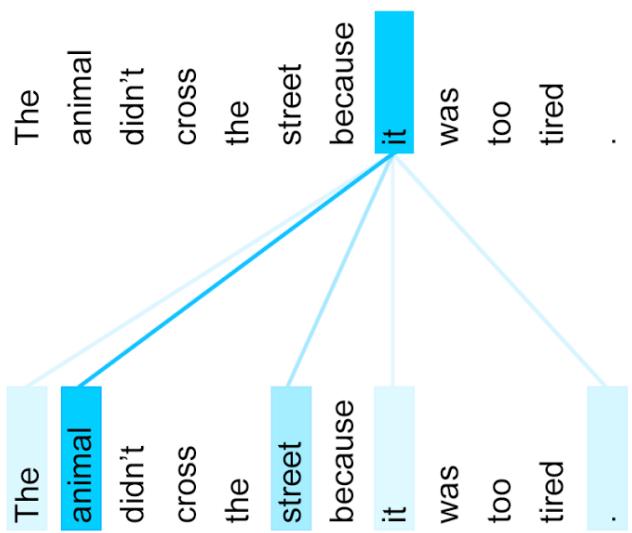


128-dimensional positonal encoding for a sentence with the maximum lenght of 50. Each row represents the embedding vector.

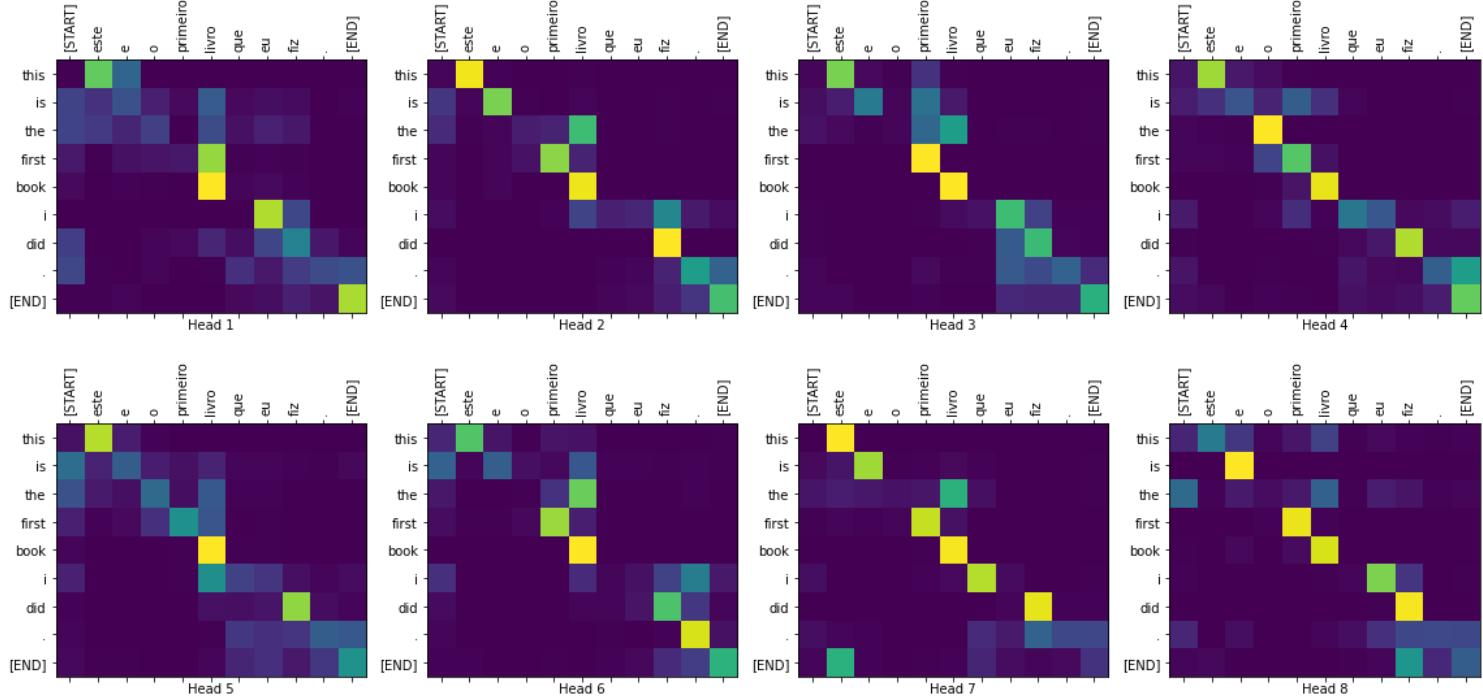
Machine translation

The transformer architecture is tested on English-to-German and English-to-French translation using WMT2014 datasets.

- English-to-German: 4.5M sentence pairs, 37k tokens vocabulary.
- English-to-French: 36M sentence pairs, 32k tokens vocabulary.
- 8 P100 GPUs (150 TFlops, FP16), 0.5 day for the small model, 3.5 days for the large one.



Self-attention layers learned that "it" could refer to different entities, in different contexts.

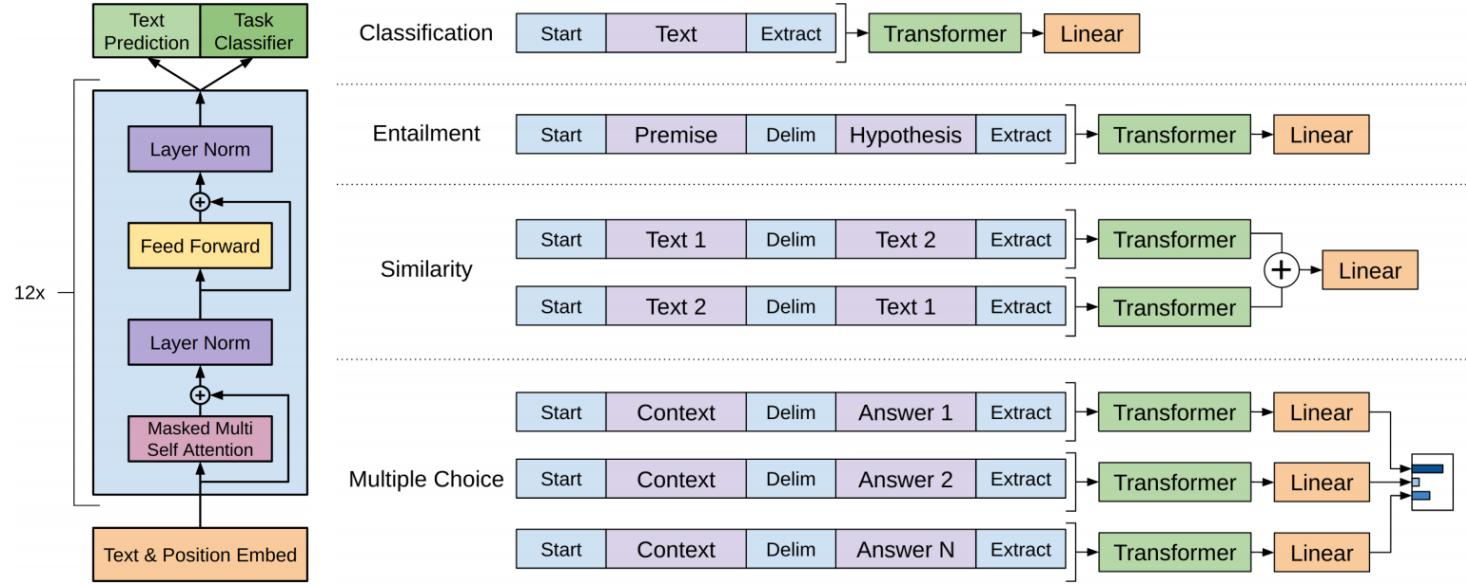


Attention maps extracted from the multi-head attention modules
show how input tokens relate to output tokens.

Language pre-training

Similar to pre-training computer vision models on ImageNet, language models can be pre-trained for tasks in natural language processing.

Notably, the models can be pre-trained in a **unsupervised manner** from very large datasets and then fine-tuned on supervised tasks with small data-sets.



GPT, Radford et al. (2018)

Increasing the training data and the model size leads to significant improvement of transformer language models. These models are now **the largest in deep learning.**



GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages. GPT-2 is trained with a simple objective: predict the next word, given all of the previous words within some text. The diversity of the dataset causes this simple goal to contain naturally occurring demonstrations of many tasks across diverse domains. GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data.

Radford et al. (2019)

The GPT-3 model has 175B parameters and was trained on 300B tokens from various sources.

Brown et al. (2020)

Vision transformers (ViTs)



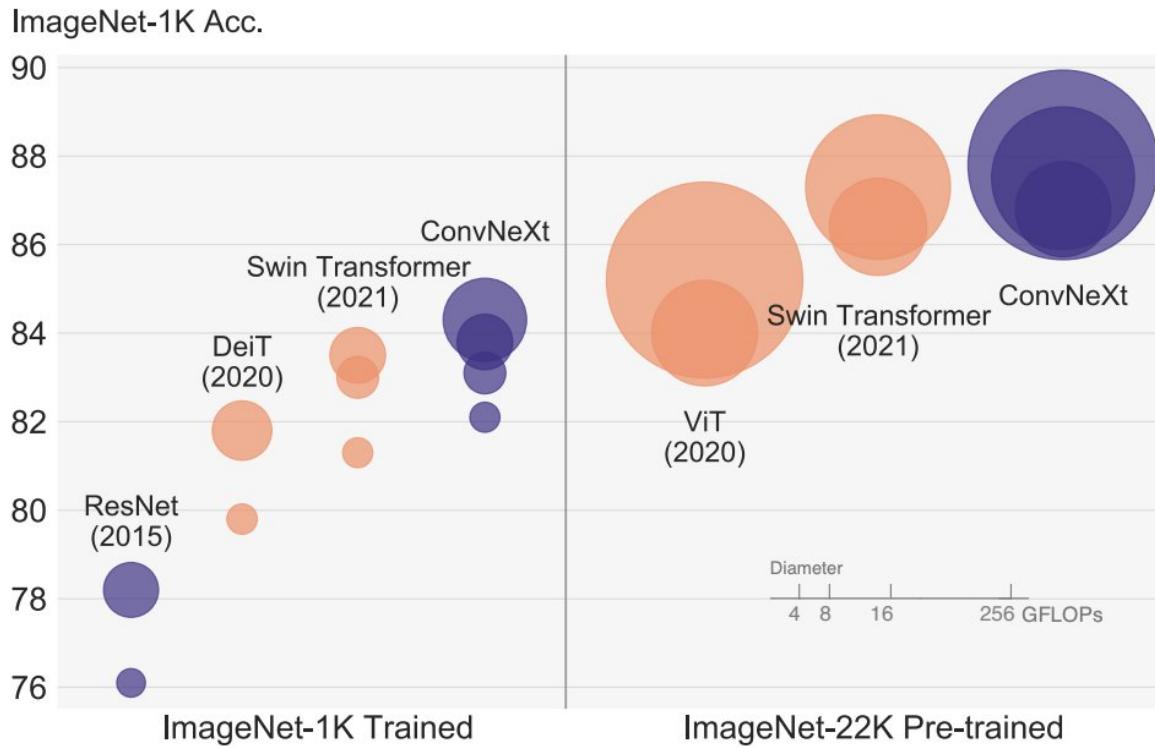


Figure 1. ImageNet-1K classification results for • ConvNets and ○ vision Transformers. Each bubble's area is proportional to FLOPs of a variant in a model family. ImageNet-1K/22K models here take $224^2/384^2$ images respectively. We demonstrate that a standard ConvNet model can achieve the same level of scalability as hierarchical vision Transformers while being much simpler in design.

(demo)

Applications

SYSTEM PROMPT
(HUMAN-WRITTEN)

Legolas and Gimli advanced on the orcs, raising their weapons with a harrowing war cry.

MODEL COMPLETION
(MACHINE-WRITTEN,
FIRST TRY)

The orcs' response was a deafening onslaught of claws, claws, and claws; even Elrond was forced to retreat. "You are in good hands, dwarf," said Gimli, who had been among the first to charge at the orcs; it took only two words before their opponents were reduced to a blood-soaked quagmire, and the dwarf took his first kill of the night. The battle lasted for hours until two of the largest Orcs attempted to overwhelm Aragorn. When they finally stopped, they lay defeated and lifeless for miles and miles.

"I take nothing," said Aragorn. "But I give my word, at my peril and mine, that I will never forget this day of horror. None of us will forget. Ever!"

"I'll never forget it!" cried Gimli, who had been in the thick of the battle but hadn't taken part in it. One of the wounded orcs he had carried off, he was the only one of the survivors who remained uninjured. "We'll keep the memory of that day of evil, and the war with it, alive as long as we live, my friends!"

"Then we'll keep it alive as long as we live," added Legolas. "And we won't forget the first great battle of the night, even if we may have forgotten the final defeat."

"I agree," Gandalf said, "but we will all remember it as the last battle in Middle-earth, and the first great battle of the new day."

GPT-2 generates synthetic text samples in response to arbitrary inputs.



OpenAI Model Generates Python Code



Later bekijk...

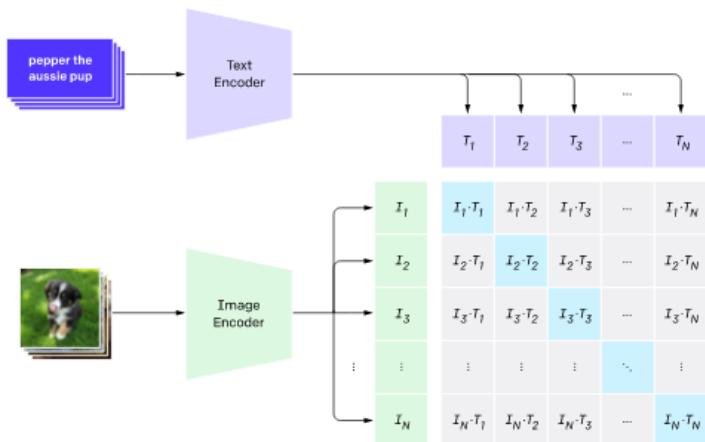


Delen

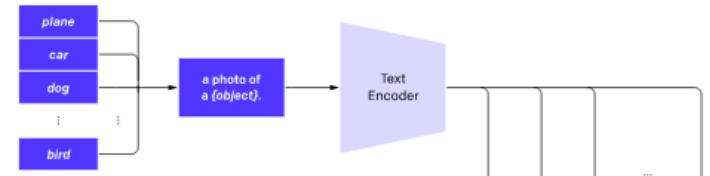


GPT-3 generates Python code.

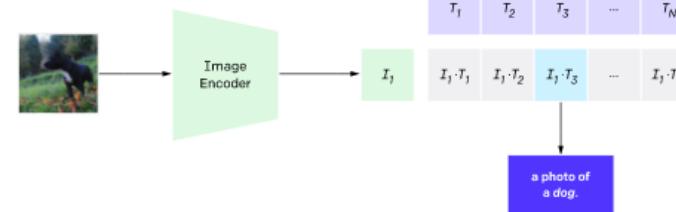
1. Contrastive pre-training



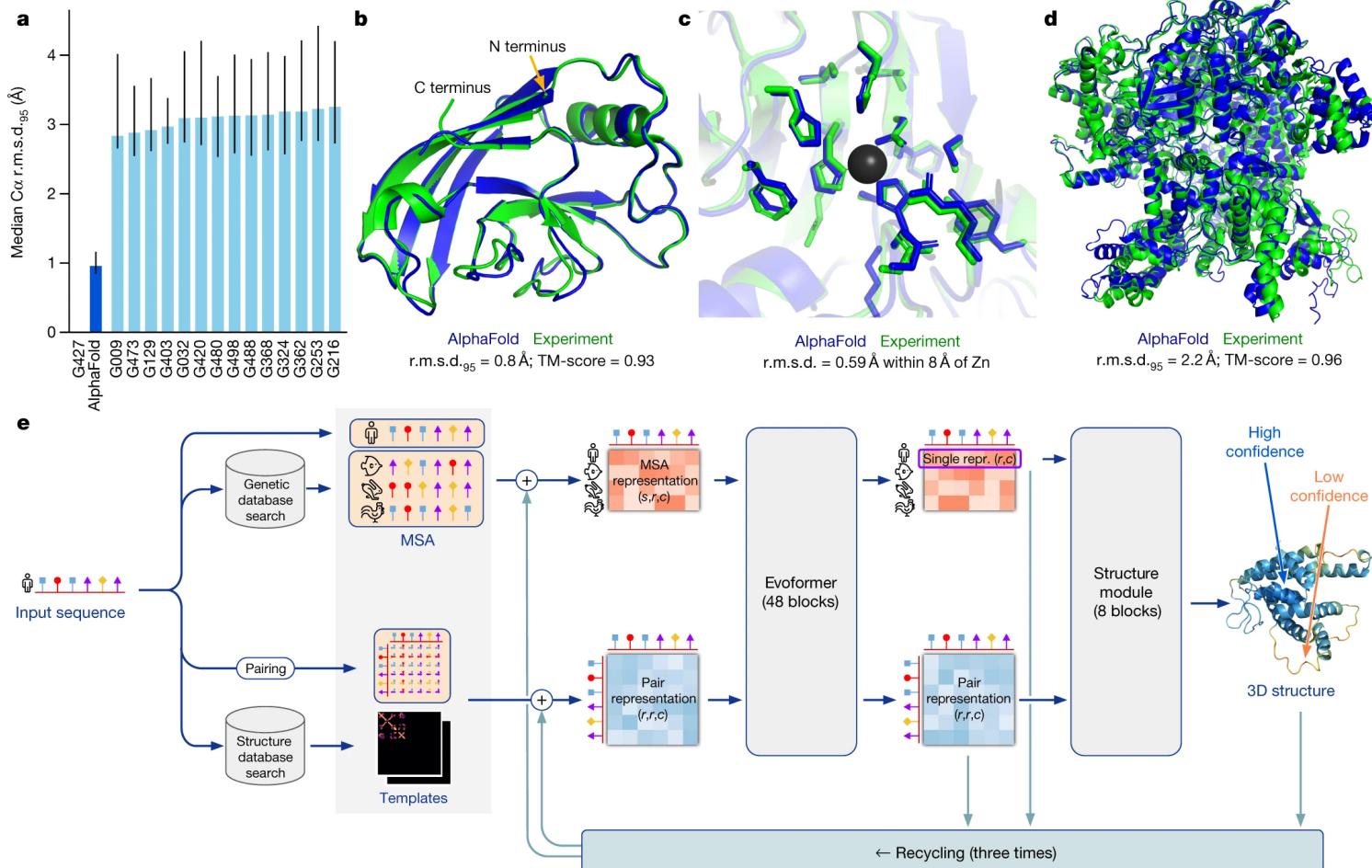
2. Create dataset classifier from label text



3. Use for zero-shot prediction



CLIP: connecting text and images for zero-shot classification (see [demo](#)).



AlphaFold: Highly accurate protein structure prediction.

• • • \hat{R}_{t-1}

Decision Transformer: Reinforcement Learning via Sequence Modeling.

Disclaimer

Large, general language models could have significant **societal impacts**, and also have many near-term applications, including:

- AI writing assistants
- more capable dialogue agents
- unsupervised translation between languages
- better speech recognition systems.

However, we can also imagine the application of these models for **malicious purposes**, including the following:

- generate misleading news articles
- impersonate others online
- automate the production of abusive or faked content to post on social media
- automate the production of spam/phising content.

The public at large will need to become more skeptical of text they find online, just as the "deep fakes" phenomenon calls for more skepticism about images.

The end.