# Deep Learning

Lecture 8: Attention and transformers

Prof. Gilles Louppe
g.louppe@uliege.be

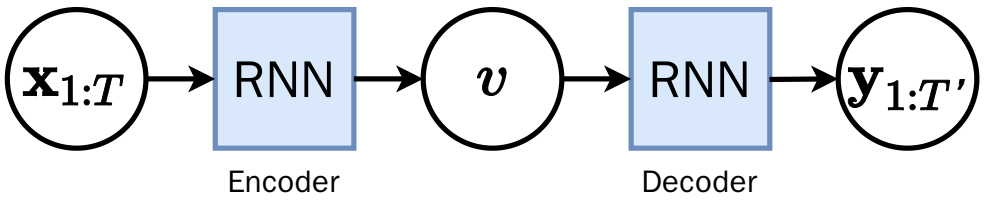
LIÈGE université

# Today

Attention is all you need!

- Context attention
- Key-value store
- Transformers

# Context attention



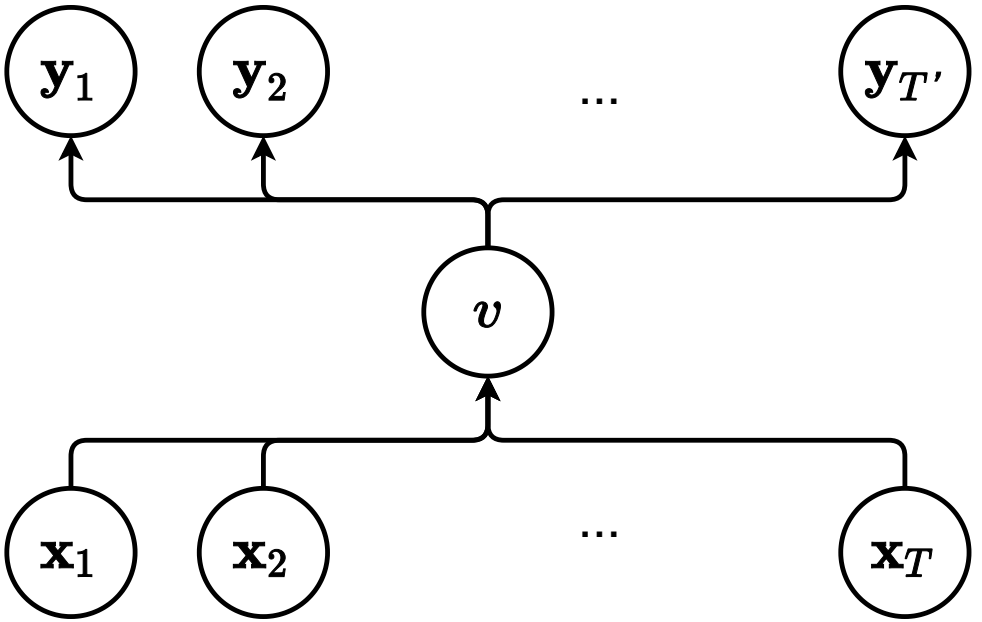

Standard RNN-based sequence-to-sequence models compress an input sequence $\mathbf{x}_{1:T}$ into a thought vector $v$ corresponding to the final recurrent state:

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

Then, they produce an output sequence $\mathbf{y}_{1:T'}$ from an autoregressive generative model

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

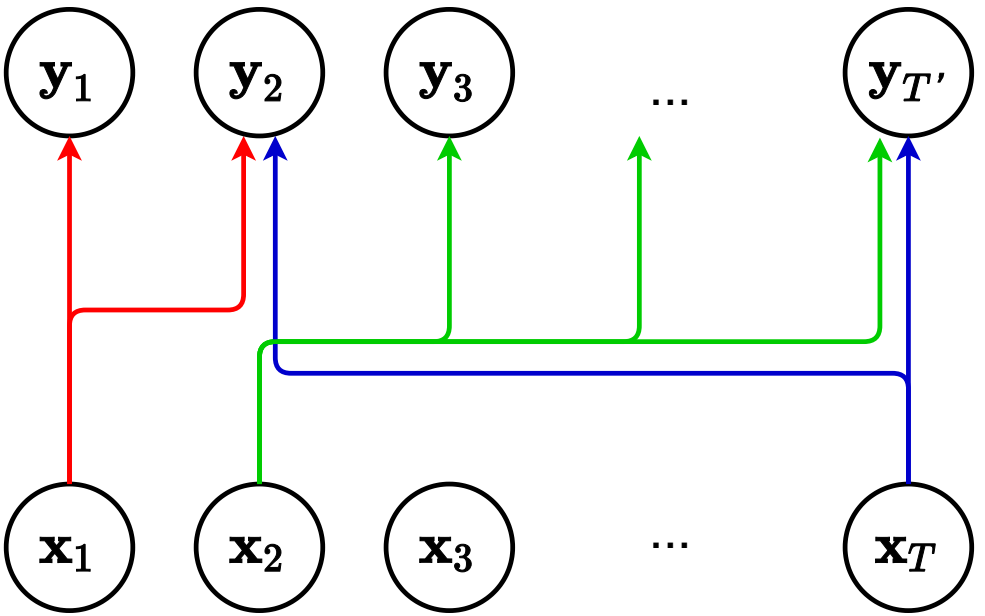
where $p(\cdot|\mathbf{y}_{1:i-1}, v)$ is itself an RNN.




This architecture assumes that the sole thought vector $v$ carries out enough information in itself 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 $i = 1, \ldots, T$, 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 $\mathbf{s}_i, i = 1, \ldots, T$, which looks at weighted averages of the $\mathbf{h}_j, j = 1, \ldots, T$, where the **weights are functions of the signal**.

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

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

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

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

and $a$ is an attention function, here specified as a one hidden layer $\tanh$ MLP.

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

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

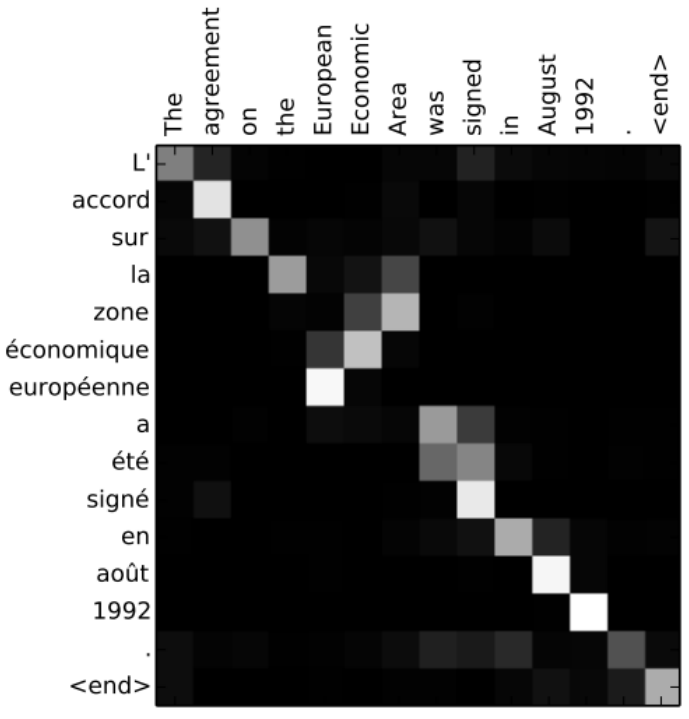
The model can now make the prediction $\mathbf{y}_i$:

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

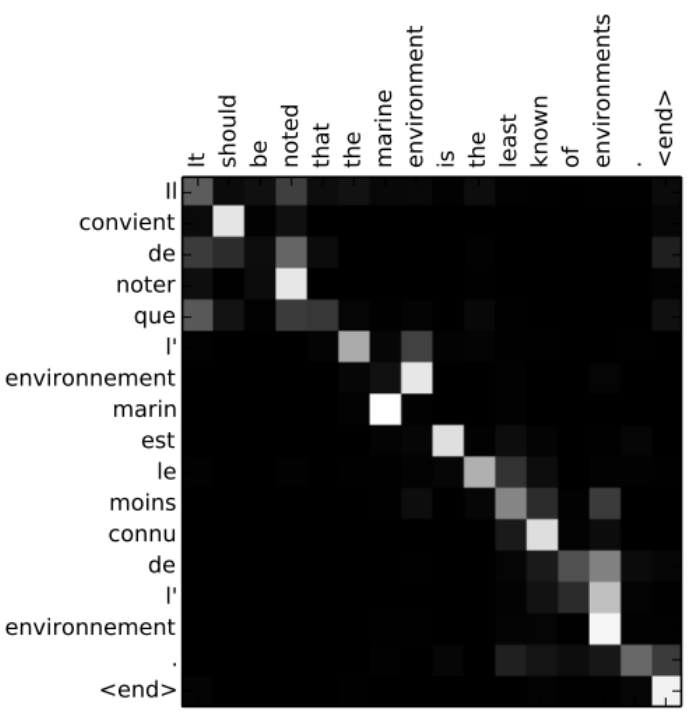
where $f$ is a GRU.

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

(whiteboard example)


The agreement on the European Economic Area was signed in August 1992 . <end>
L' accord sur la zone économique européenne a été signé en août 1992 . <end>

(a)


It should be noted that the marine environment is the least known of environments . <end>
Il convient de noter que l' environnement marin est le moins connu de l' environnement . <end>

(b)

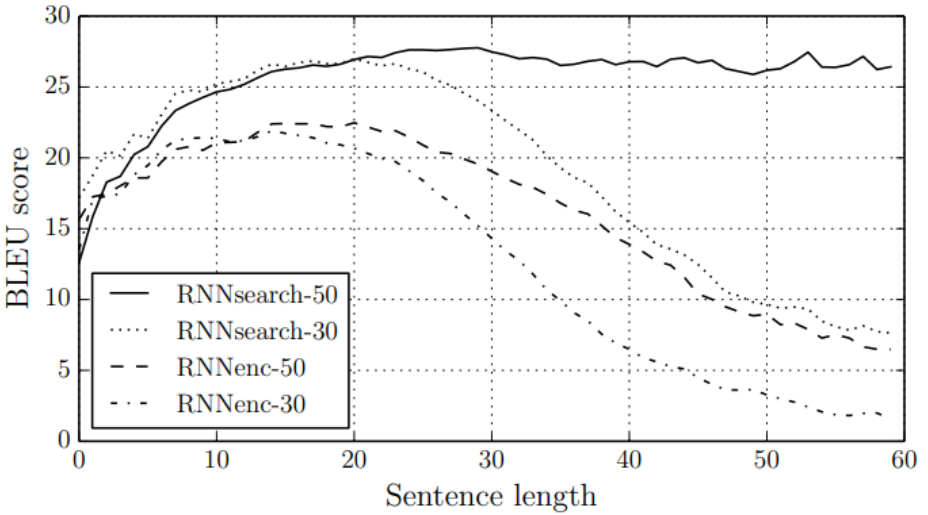
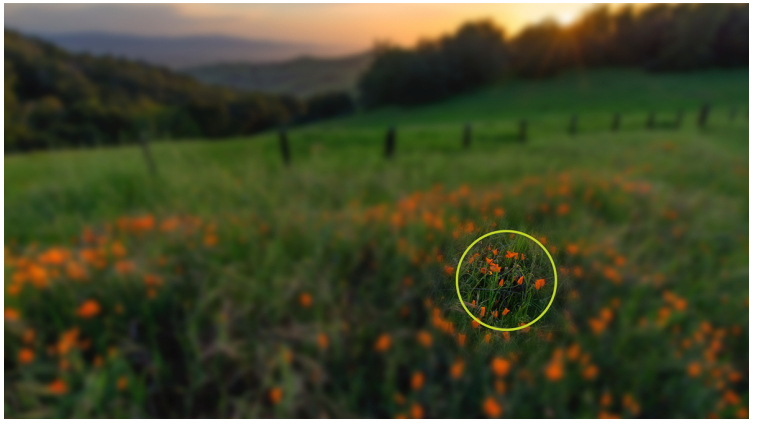


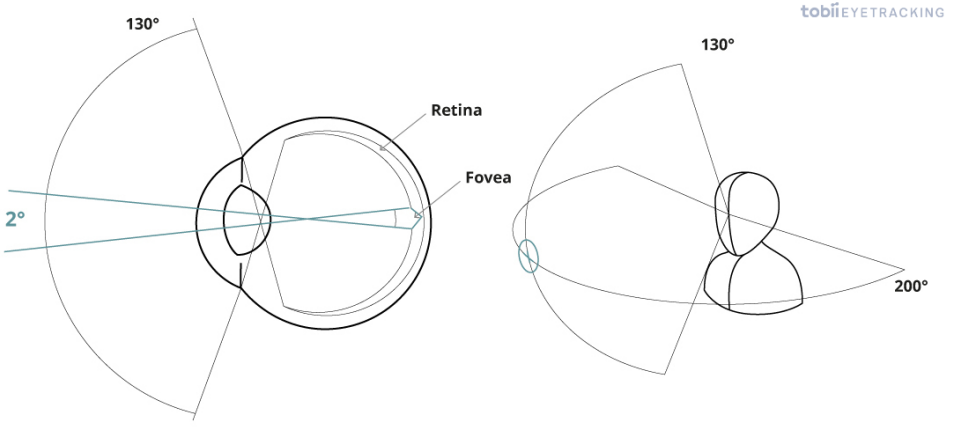

Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

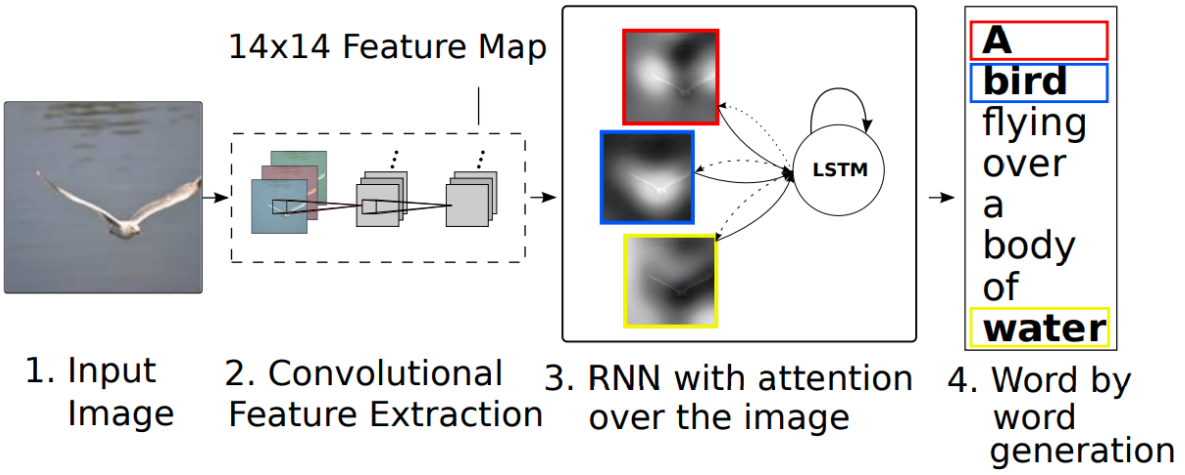
# Attention-based caption generation

The human eye cannot process a whole visual scene at once. The fovea enables high-acuity vision in only a tiny region of our field of view.

Instead, we must integrate information from a series of glimpses.





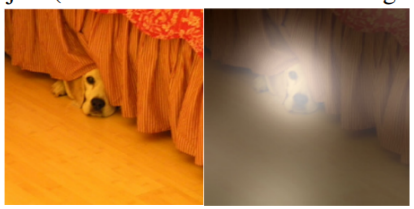



Following Xu et al. (2015), the context attention mechanism can be adapted to caption generation:

- Encoder: a CNN that extracts a feature map over the input image.
- Decoder: an attention-based RNN that computes at each step an attention map over the entire feature map, effectively deciding which regions to focus on.

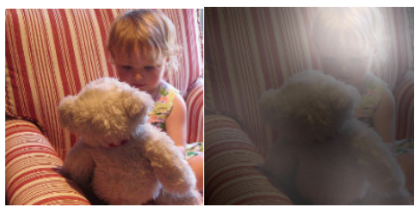
*Figure 3.* Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

A woman is throwing a frisbee in a park.

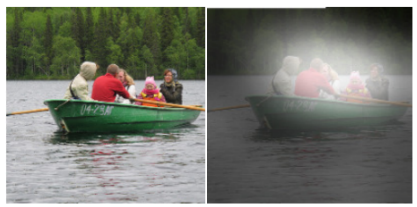


A dog is standing on a hardwood floor.


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



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

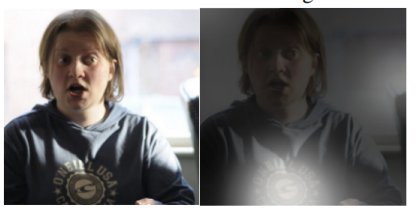


A group of people sitting on a boat in the water.

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

*Figure 5.* Examples of mistakes where we can use attention to gain intuition into what the model saw.

A large white bird standing in a forest.



A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

# Key-value store

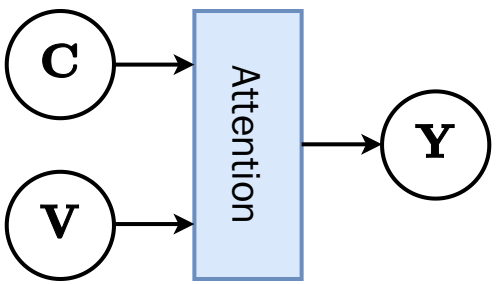
## Context attention

The previous context attention mechanism can be generically defined as follows.

Given a context tensor $\mathbf{C} \in \mathbb{R}^{T \times C}$ and a value tensor $\mathbf{V} \in \mathbb{R}^{S \times D}$, context attention computes an output tensor $\mathbf{Y} \in \mathbb{R}^{T \times D}$ with

$$\mathbf{Y}_j = \sum_{i=1}^{S} \text{softmax}_i(a(\mathbf{C}_j, \mathbf{V}_i; \theta))\mathbf{V}_i,$$

where $a : \mathbb{R}^C \times \mathbb{R}^D \to \mathbb{R}$ is a scalar attention function.

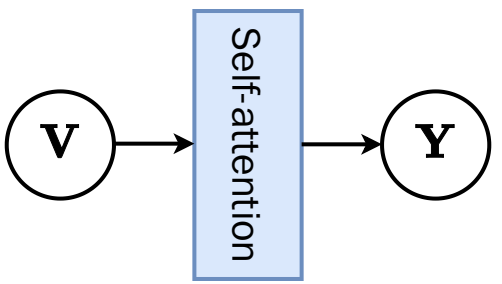


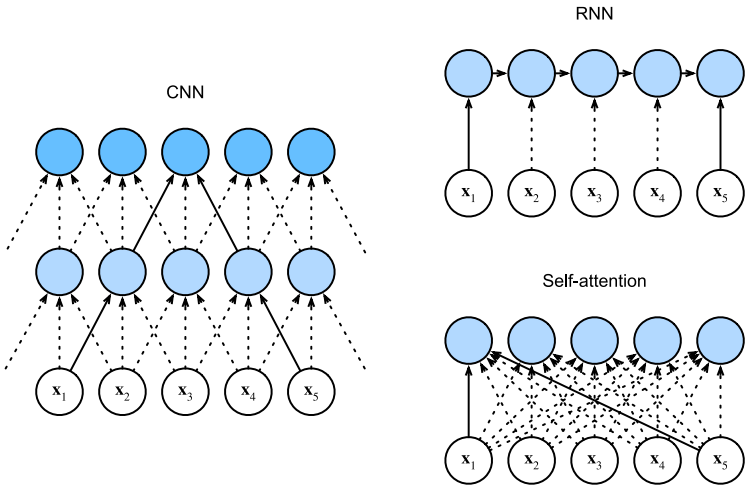
## Self-attention

When $\mathbf{C} = \mathbf{V}$, context attention is becomes a self-attention layer.

Given a value tensor $\mathbf{V} \in \mathbb{R}^{S \times D}$, self-attention computes an output tensor $\mathbf{Y} \in \mathbb{R}^{S \times D}$ with

$$\mathbf{Y}_j = \sum_{i=1}^{S} \text{softmax}_i(a(\mathbf{V}_j, \mathbf{V}_i; \theta))\mathbf{V}_i.$$






## Complexity

| 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.

—

Credits: Dive Into Deep Learning, 10.6.2.

# Key-value store

Following the terminology of Graves et al. (2014) et Vaswani et al. (2017), attention can be generalized to an averaging of values associated to keys matching a query.

With $\mathbf{Q}$ the tensor of row $T$ queries, $\mathbf{K}$ the tensor of $T'$ row keys, and $\mathbf{V}$ the tensor of $T'$ row values,

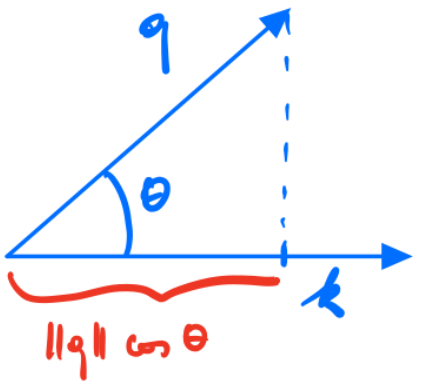
$$\mathbf{Q} \in \mathbb{R}^{T \times D}, \mathbf{K} \in \mathbb{R}^{T' \times D}, \mathbf{V} \in \mathbb{R}^{T' \times D'},$$

and using a dot-product for attention function, an attention operation yields

$$\mathbf{Y}_j = \sum_{i=1}^{T'} \frac{\exp(\mathbf{Q}_j \mathbf{K}_i^T)}{\sum_{r=1}^{T'} \exp(\mathbf{Q}_j \mathbf{K}_r^T)} \mathbf{V}_i$$

or

$$\mathbf{Y} = \underbrace{\mathrm{softmax}(\mathbf{Q}\mathbf{K}^T)}_{\text{attention matrix } \mathbf{A}} \mathbf{V}.$$




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

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

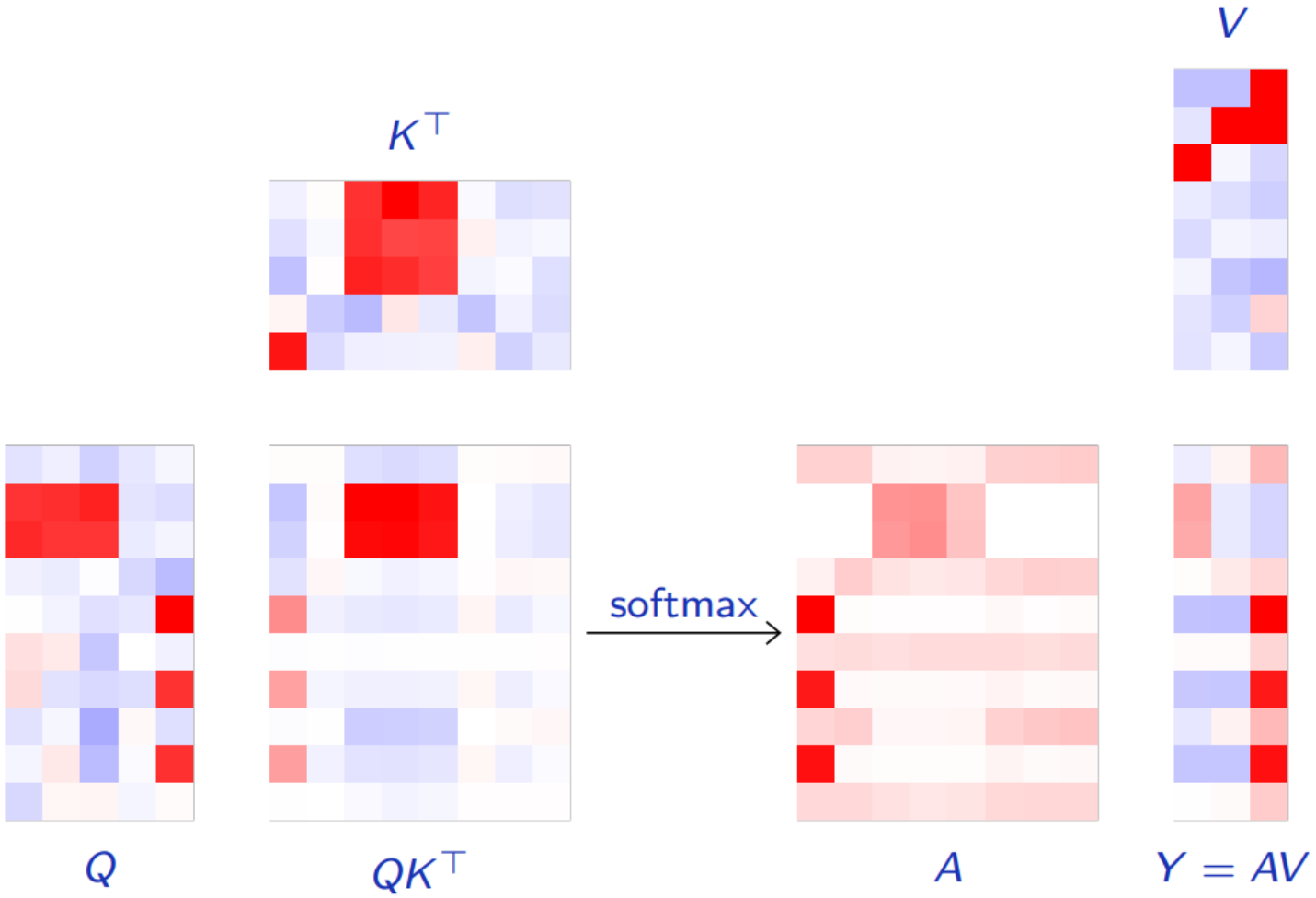
Therefore, the $\mathbf{QK}^T$ matrix is a similarity matrix between queries and keys.

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

Given the (learnable) matrices $\mathbf{W}_Q \in \mathbb{R}^{D \times C}$, $\mathbf{W}_K \in \mathbb{R}^{D \times C'}$, and $\mathbf{W}_V \in \mathbb{R}^{D' \times C'}$, and two input sequences $\mathbf{X} \in \mathbb{R}^{T \times C}$ and $\mathbf{X'} \in \mathbb{R}^{T' \times C'}$, we have

$$\begin{aligned}\mathbf{Q} &= \mathbf{X}\mathbf{W}_Q^T \in \mathbb{R}^{T \times D} \\ \mathbf{K} &= \mathbf{X'}\mathbf{W}_K^T \in \mathbb{R}^{T' \times D} \\ \mathbf{V} &= \mathbf{X'}\mathbf{W}_V^T \in \mathbb{R}^{T' \times D'}.\end{aligned}$$

As for context attention, we obtain self-attention when $\mathbf{X} = \mathbf{X}'$.


$V$
$K^\top$
softmax
$Q$
$QK^\top$
$A$
$Y = AV$

—

(demo)

# Transformers

# 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.

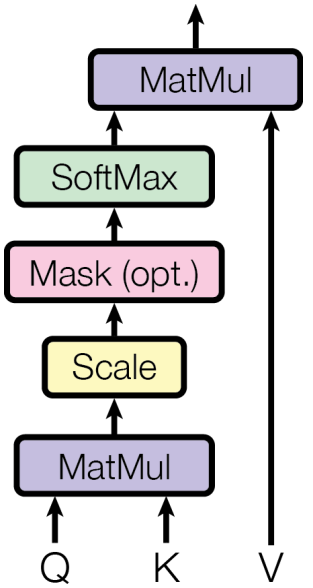
The transformer was designed for a sequence-to-sequence translation task, but it is currently key to state-of-the-art approaches across NLP tasks.

## Scaled dot-product attention

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

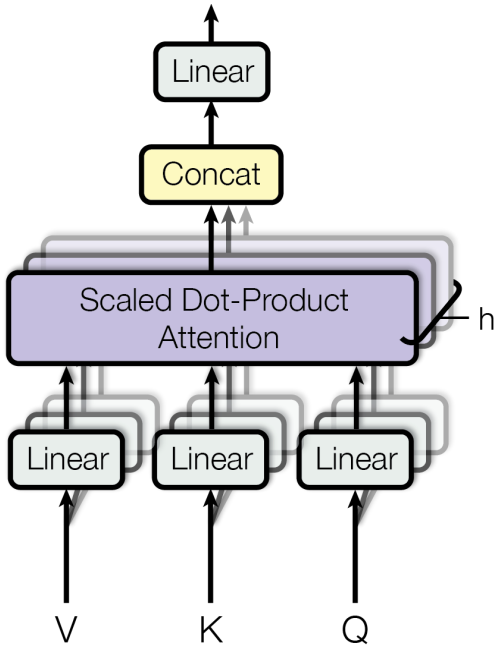
$$\text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

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



## Multi-head attention

Instead of performing a single attention function with $d_{\text{model}}$-dimensional keys, values and queries, the transformer architecture project the queries, keys and values $h = 8$ times with different, learned linear projections to $d_k = 64$, $d_k = 64$ and $d_v = 64$ dimensions respectively.




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

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

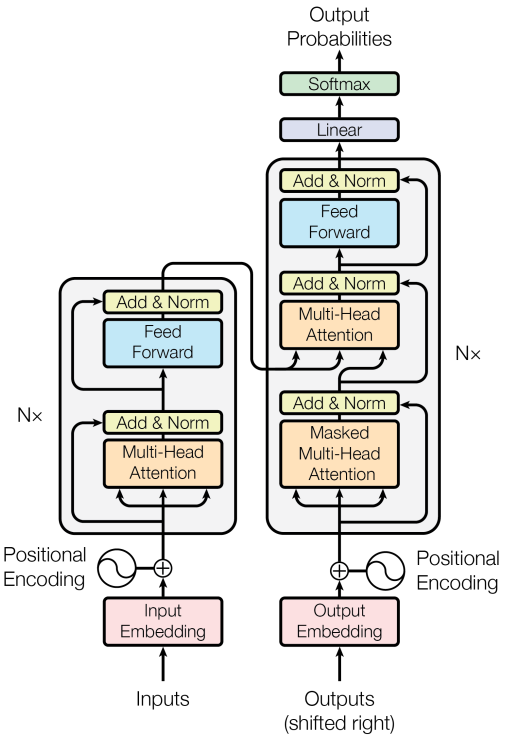
with

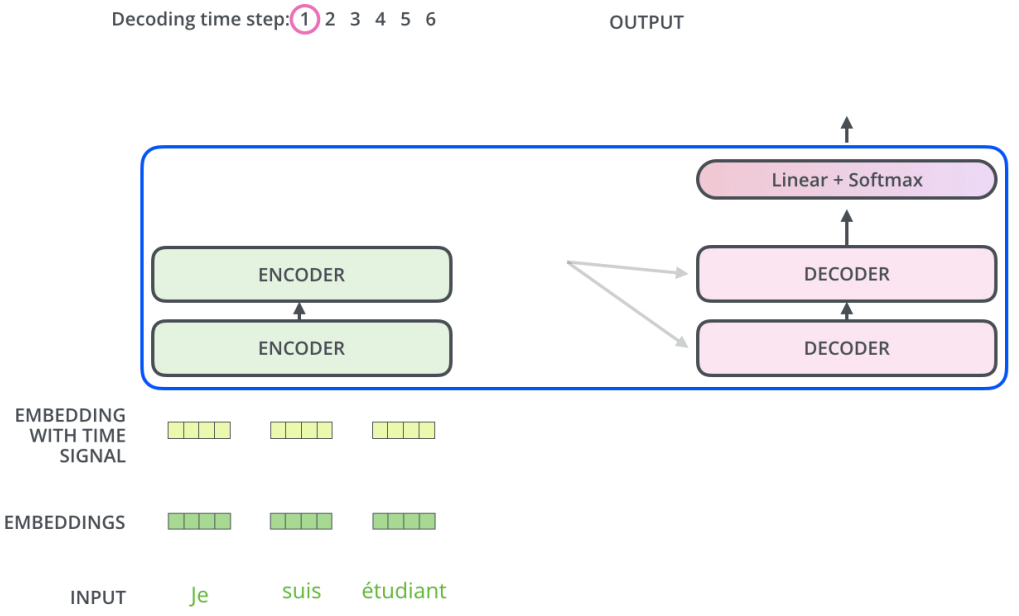
$$\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, \mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}, \mathbf{W}_i^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

## Encoder-decoder architecture

Their complete transformer model is composed of:

- An encoder that combines $N = 6$ modules, each composed of a multi-head attention sub-module, 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.

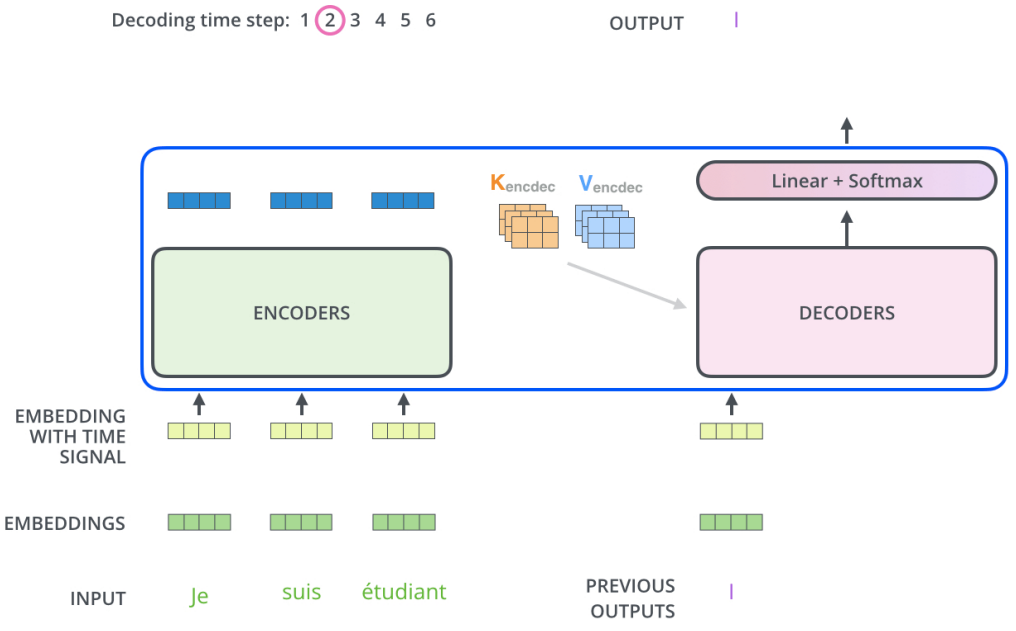





The encoders start by processing the input sequence. The output of the top encoder is then transformed into a set of attention vectors $\mathbf{K}$ and $\mathbf{V}$ that will help the decoders focus on appropriate places in the input sequence.

—

Credits: Jay Alammar, The Illustrated Transformer.




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.

—

Credits: Jay Alammar, The Illustrated Transformer.

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.

—

Credits: Jay Alammar, The Illustrated Transformer.

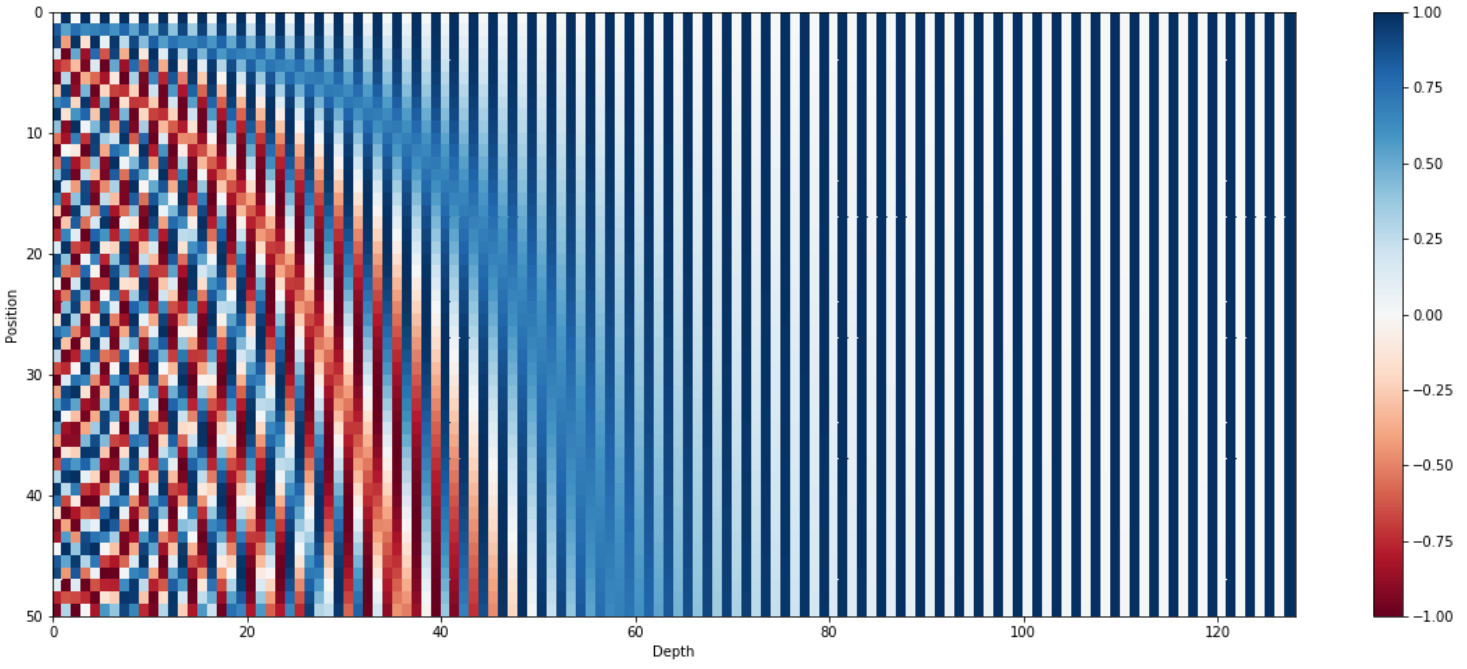
## Positional encoding

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 $d_{\text{model}}$ as the internal representation and is of the form

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

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 $d_{\text{model}}$-dimensional space.




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

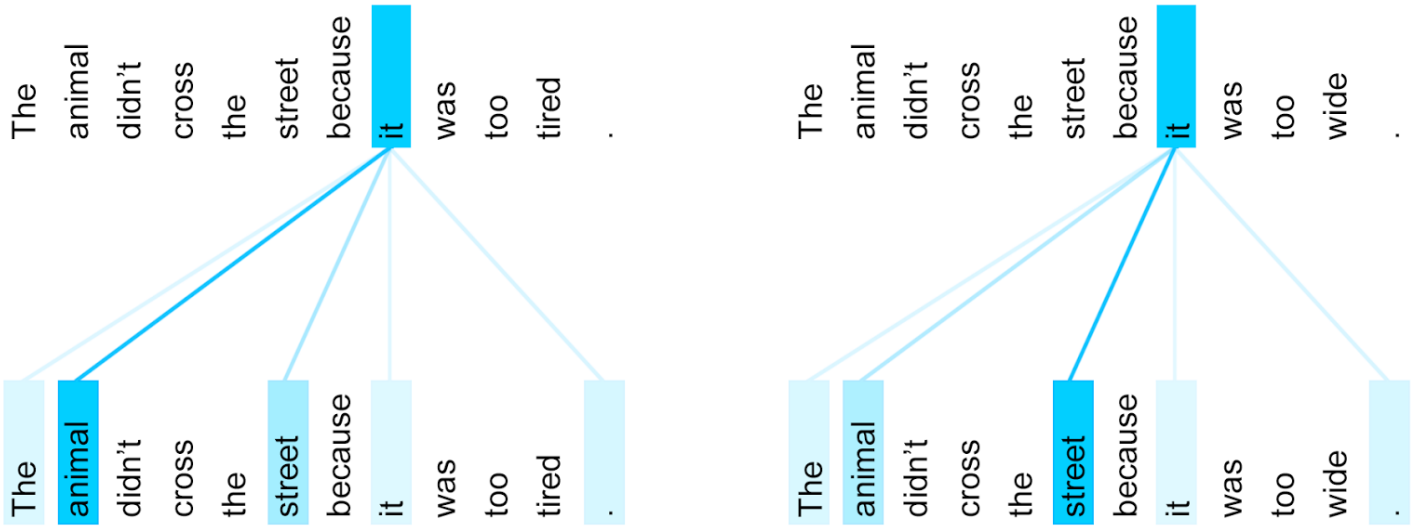
## Machine translation

The 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.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

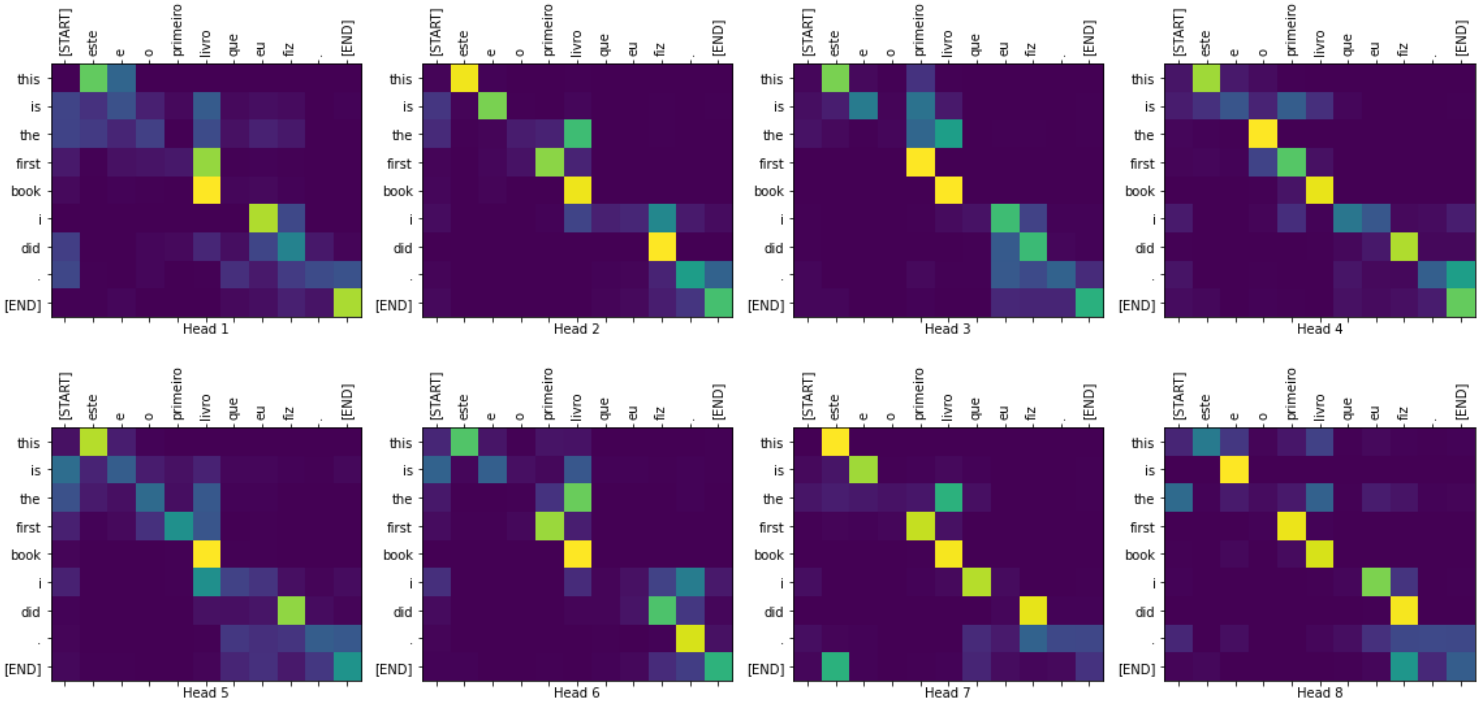
| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | **$3.3 \cdot 10^{18}$** | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |




Self-attention layers learnt "it" could refer to different entities in different contexts.

—

Credits: Transformer: A Novel Neural Network Architecture for Language Understanding, 2017.




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

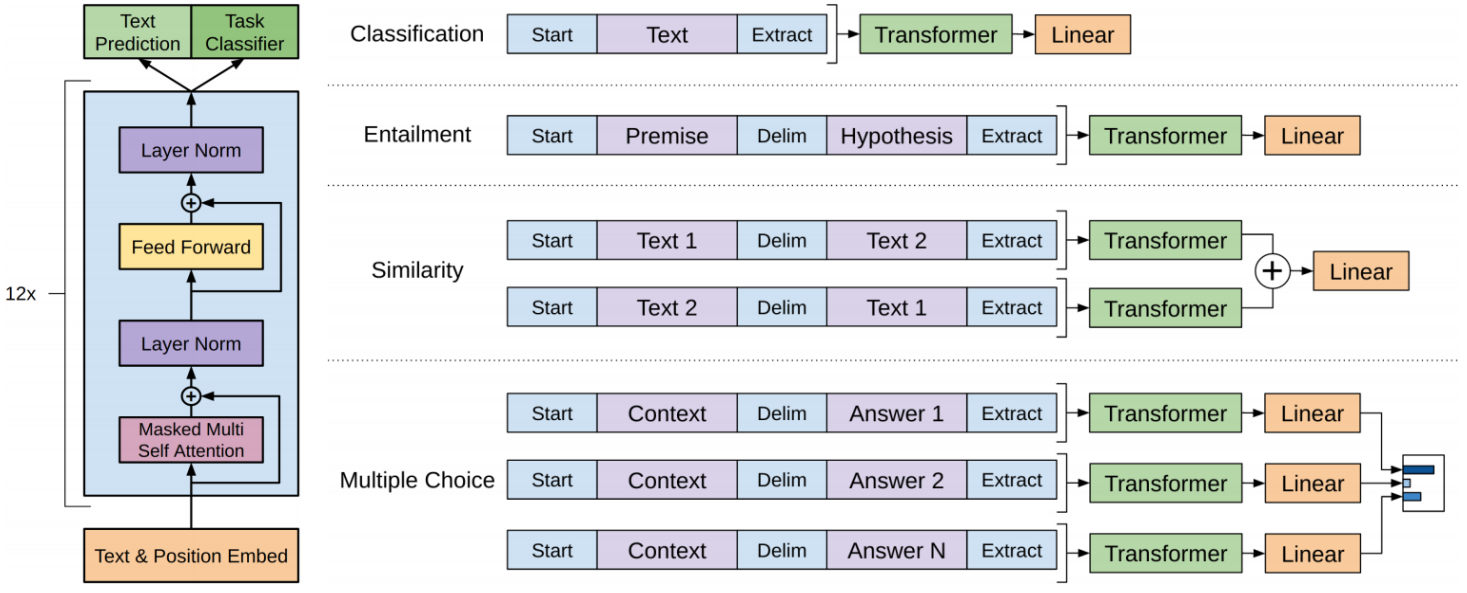
—

Credits: Transformer model for language understanding.

# 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.

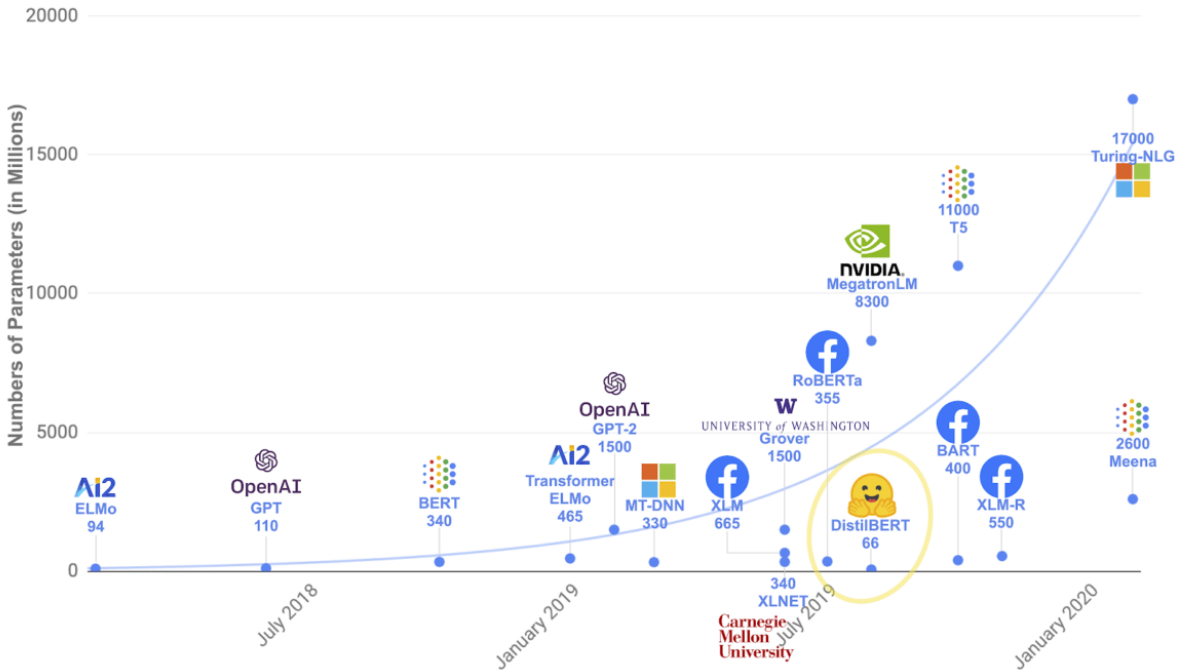



GPT, Radford et al. (2018)

—

Credits: Radford et al., Improving Language Understanding by Generative Pre-Training, 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)

# Applications

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.

—

Credits: OpenAI Better Language Models and Their Implications.

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.

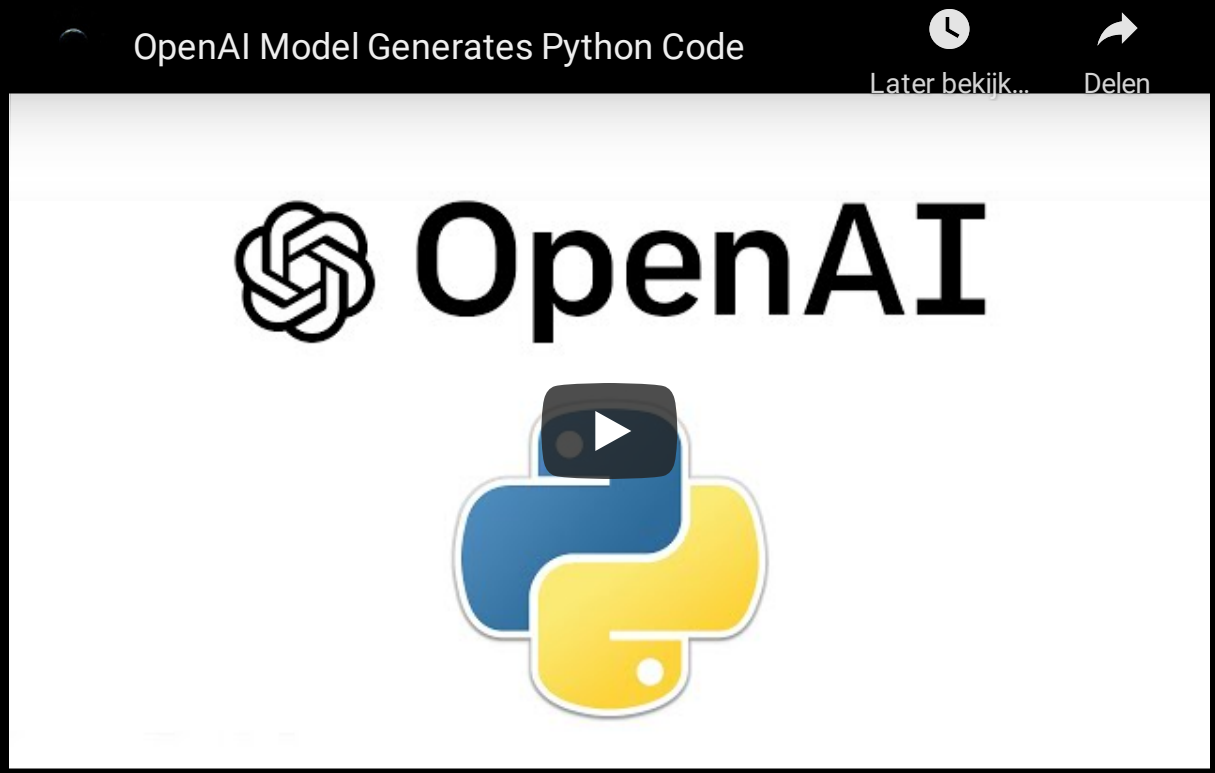
> The public at large will need to become more skeptical of text they find online, just as the "deep fakes" phenomenon calls for more skepticism about images.

—

Credits: OpenAI Better Language Models and Their Implications.

| | |
|---|---|
| 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.<br><br>"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!"<br><br>"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!"<br><br>"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."<br><br>"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 the model being primed arbitrary input. See OpenAI's blog post.

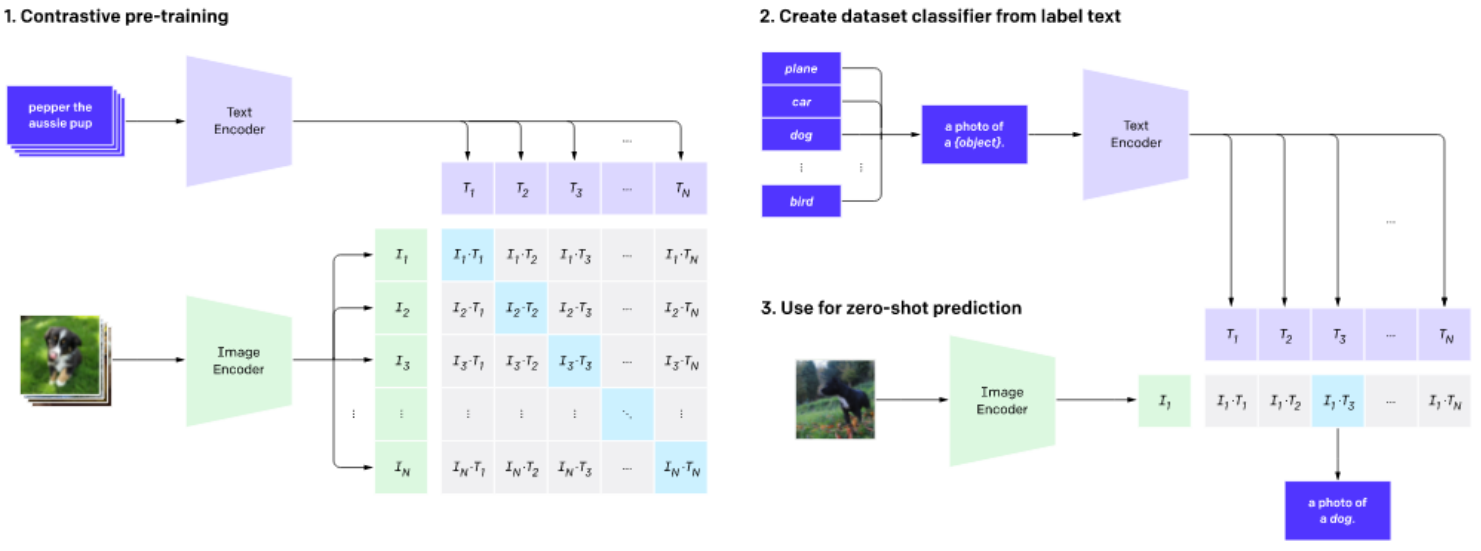



GPT-3 generates Python code

TEXT PROMPT an illustration of a baby panda in a suit watching tv

AI-GENERATED IMAGES

DALL·E: a 12-billion parameter version of GPT-3 trained to generate images from text descriptions. See OpenAI's blog post.




CLIP: connecting text and images for zero-shot classification. See demo.

The end.