TRANSFORMERS

Mariano Rivera

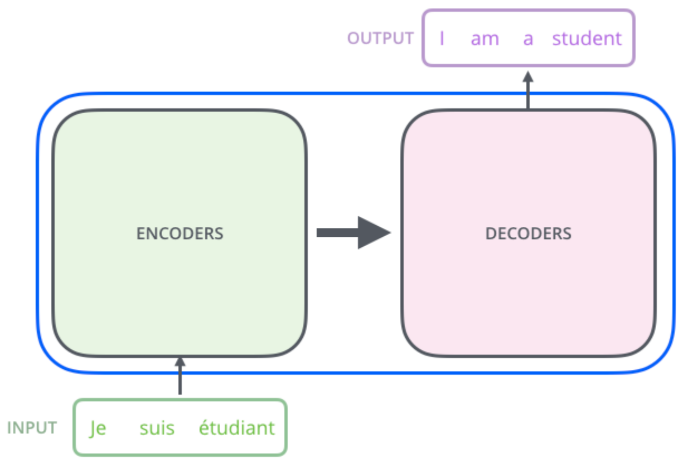
Useful tutorials

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- Alexander M. Rush. The annotated transformer ACL NLP-OSS 2018, http://nlp.seas.harvard.edu/2018/04/03/attention.html)
- **Michael Phi, Illustrated Guide to Transformers: Step by Step Explanation,** https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0
- Jay Alammar (2018) The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Consider the language translation problem

Fig. Jay Alammar (2018)

Encoder-Decoder solution, seq2seq models



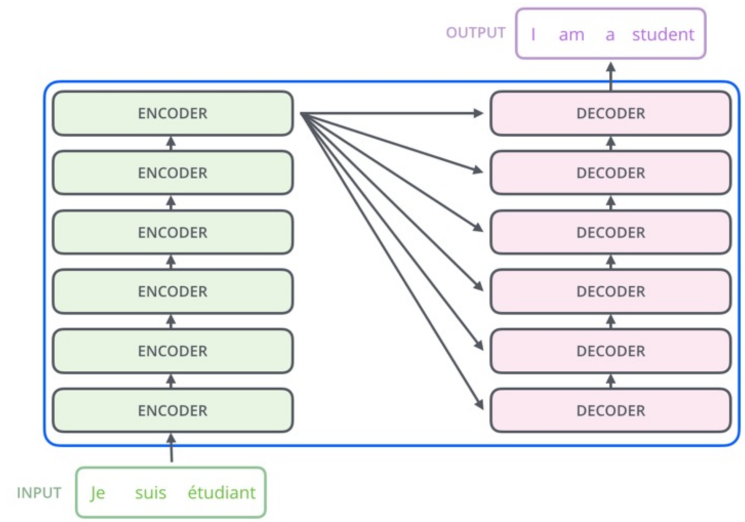
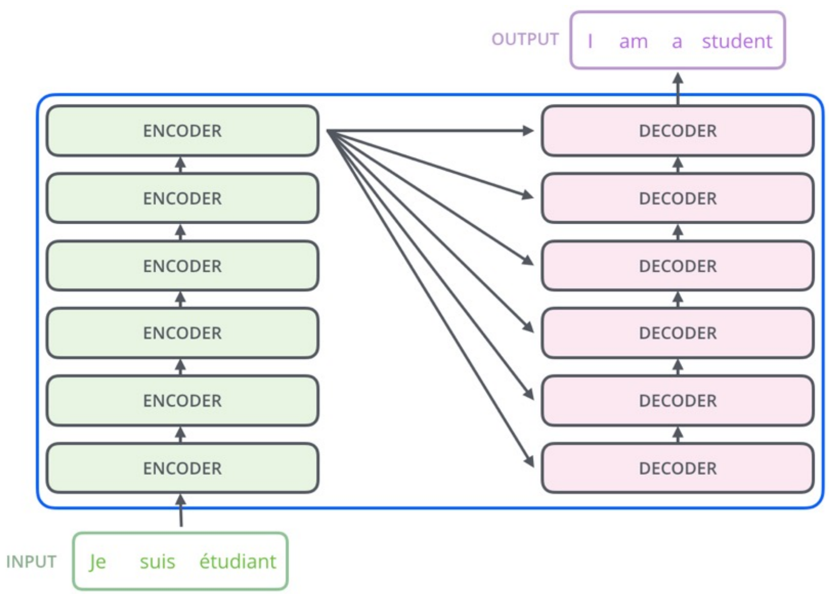


Fig. Jay Alammar (2018)

Encoder-Decoder with Attention



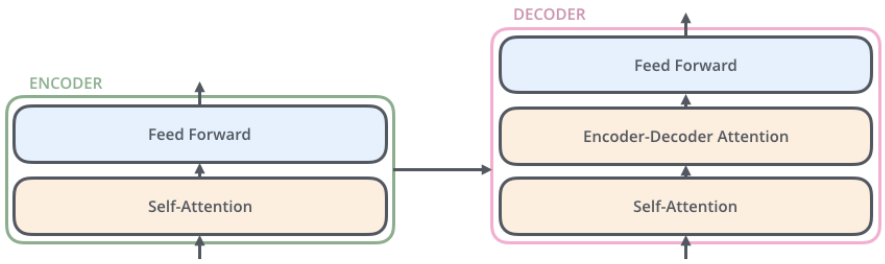


Fig. Jay Alammar (2018)

General view

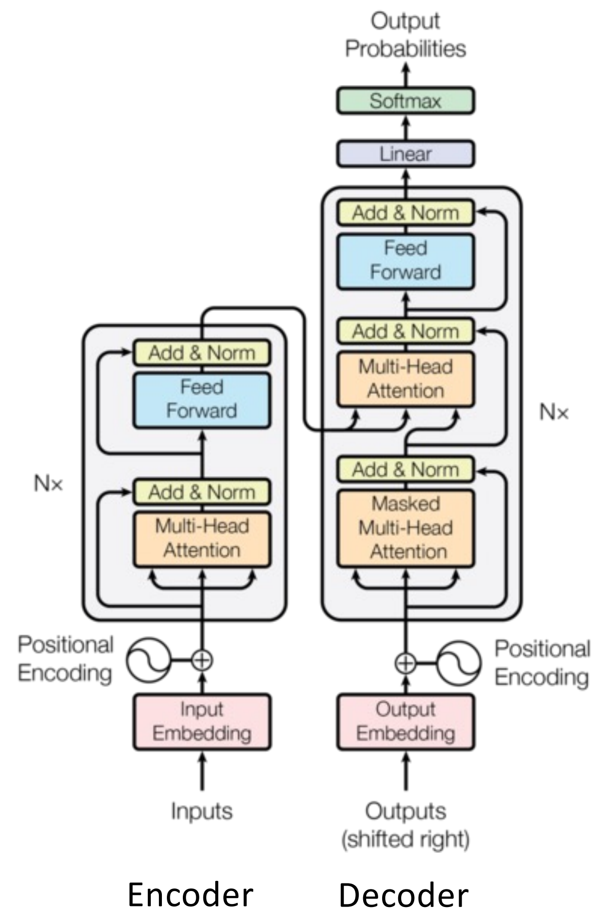
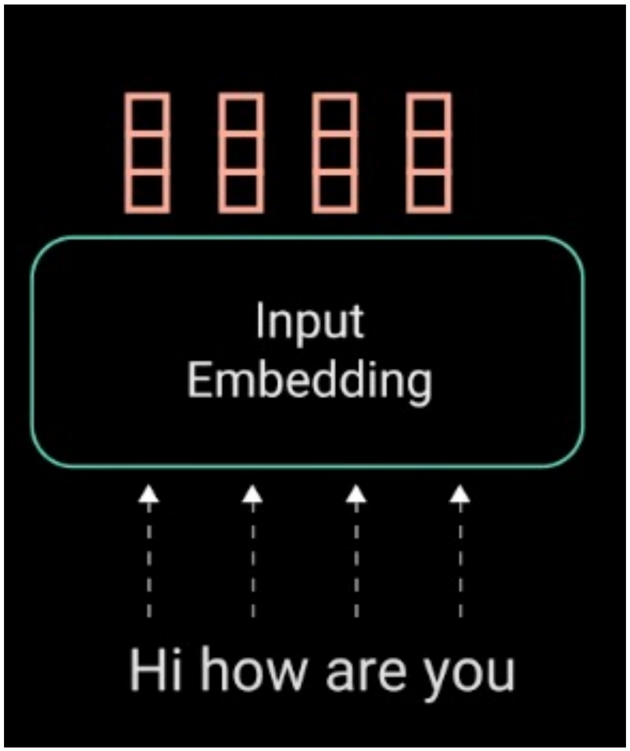


Fig. Rush (2018)

ENCODER



Input Embeddings

Word embedding layer can be thought of as a lookup table to grab a learned vector representation of each word.

The encoded list is a long as the longest sentence

Positional Encoding

The transformer encoder has no recurrence like recurrent neural networks, we must add information about the positions into the input embeddings.

This is done using positional encoding.

The authors use clever trick using sine and cosine functions.

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

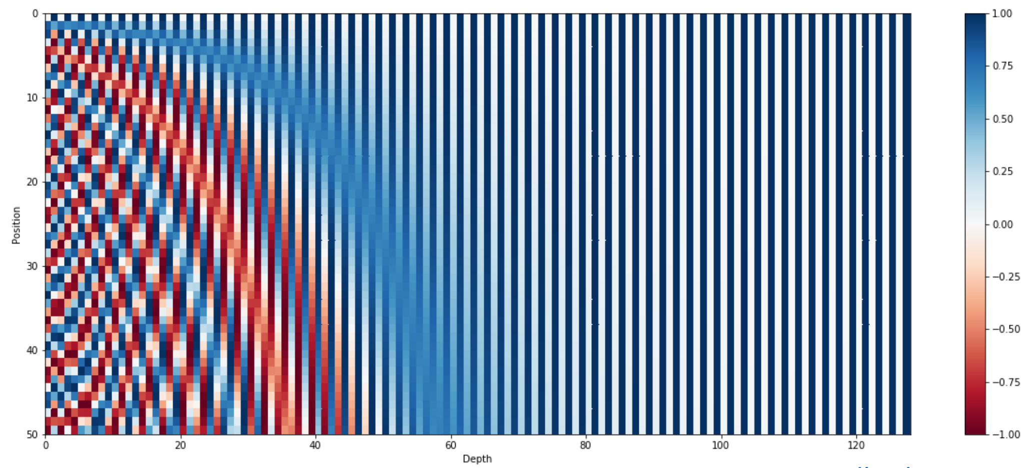
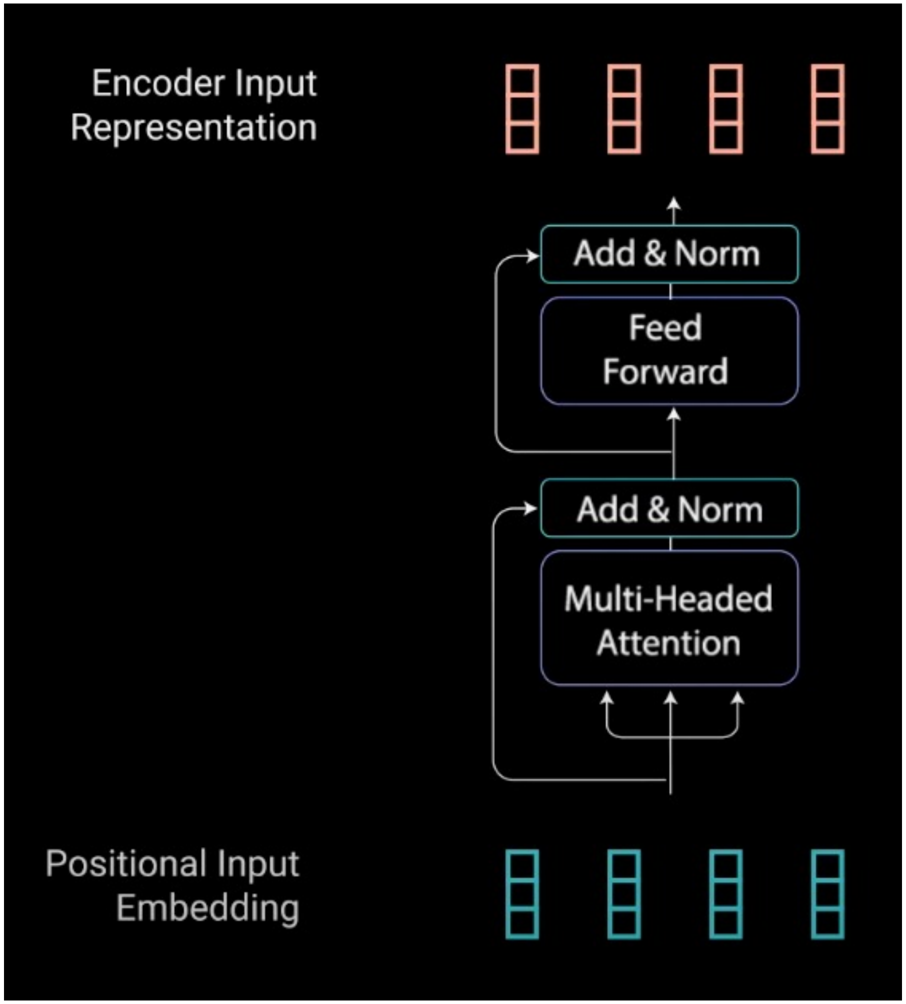


Fig. Kazemnejad's Blog, 2019

For every odd index on the input vector, create a vector using the cos function.

For every even index, create a vector using the sin function.

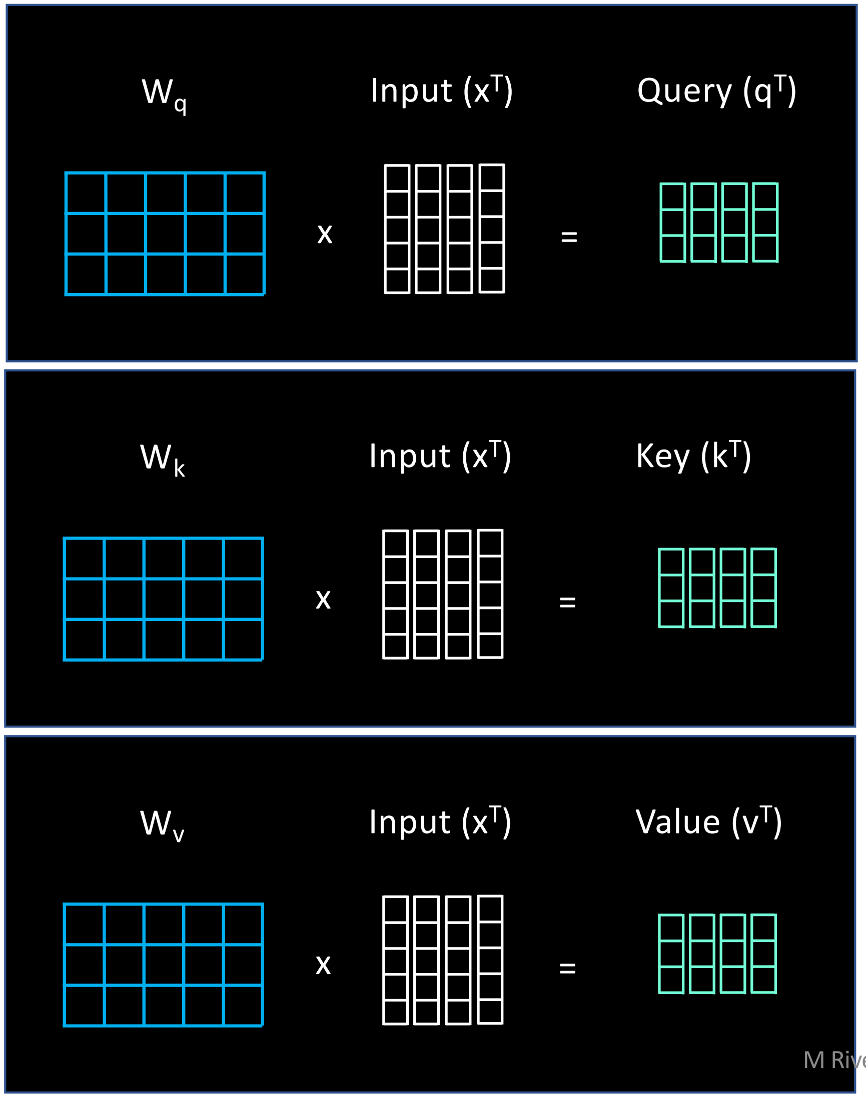


Encoder Layer

It map all input sequences into an abstract continuous representation that holds the learned information for that entire sequence.

Modules:

- multi-headed attention
- fully connected network
- residual connections around each of the two sublayers followed by a layer normalization.



Query, Key, and Value Vectors

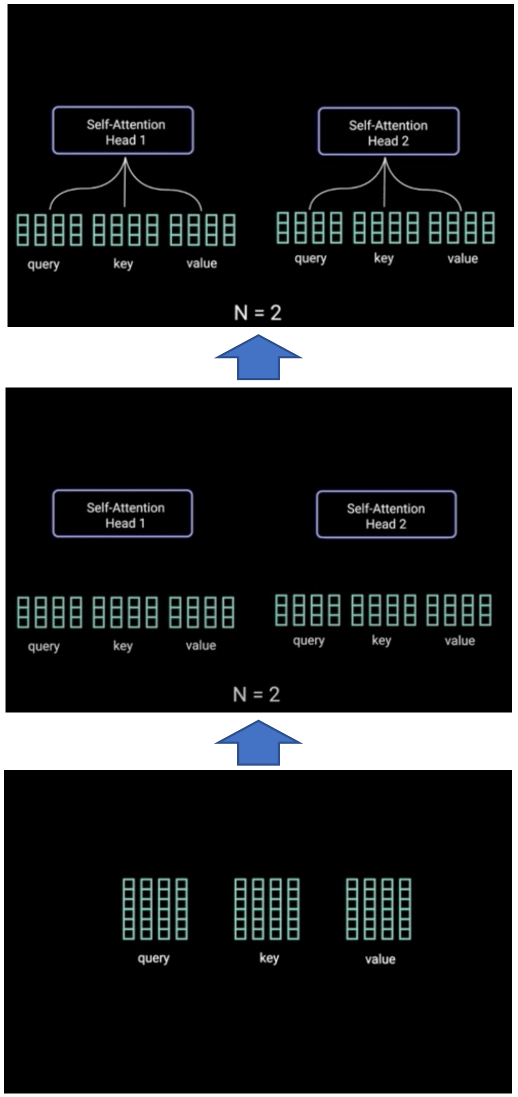
The input is fed into 3 distinct fully connected layers to create the query, key, and value vectors

$$Q = W_q x$$

$$K = W_k x$$

$$V = W_v x$$

"The query key and value concept come from retrieval systems. For example, when you type a query to search for some video on Youtube, the search engine will map your **query** against a set of **keys** (video title, description etc.) associated with candidate videos in the database, then present you the best matched videos (**values**)."



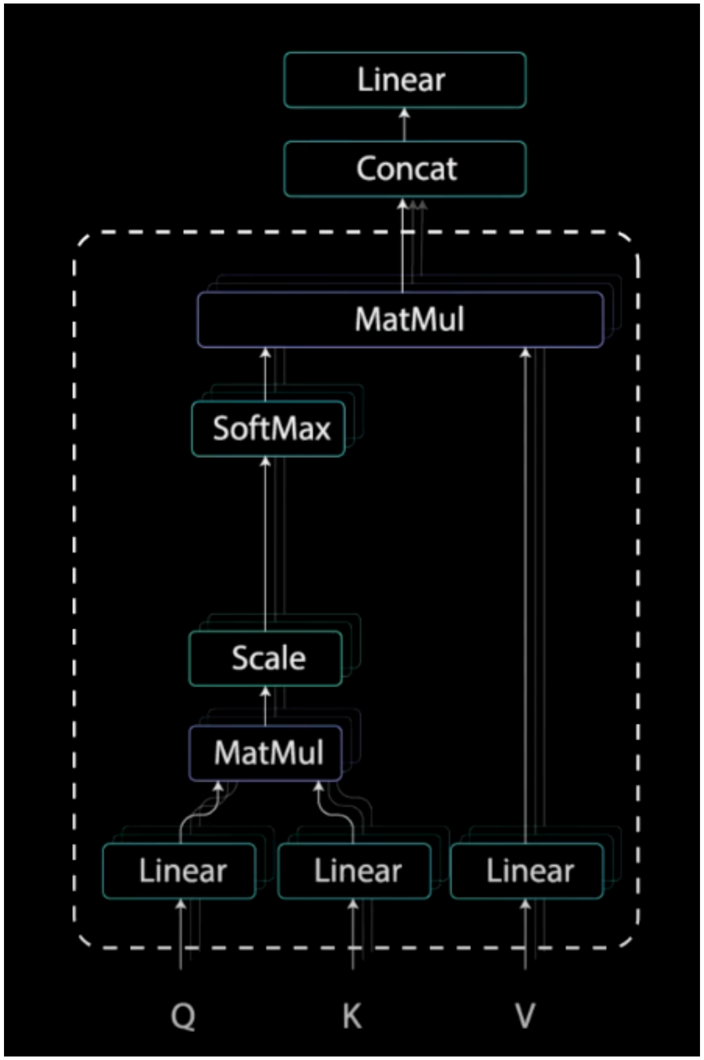
Computing Multi-headed Attention

Pass the query, key, and value into N vectors to applying self-attention.

Each self-attention process is called a head.

Each head produces an output vector that gets concatenated into a single vector before going through the final linear layer.

In theory, each head would learn something different therefore giving the encoder model more representation power.

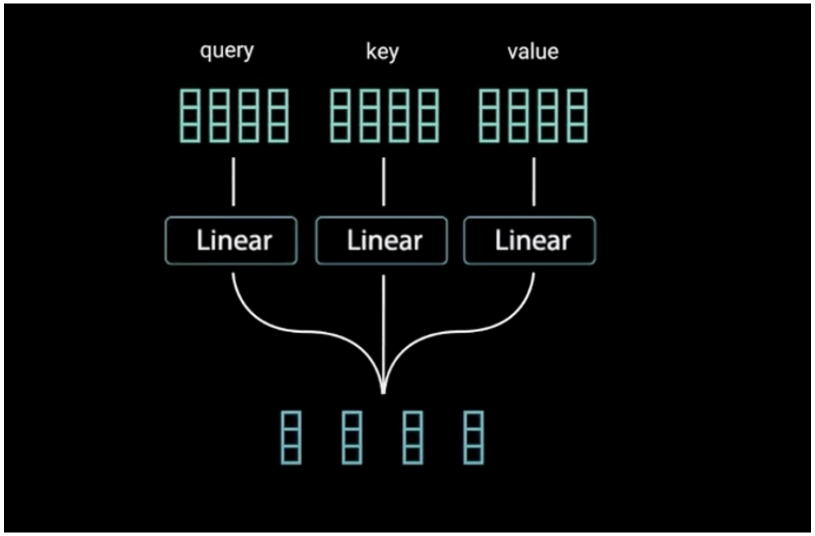


Multi-Headed Attention

Applies a specific attention mechanism called self-attention.

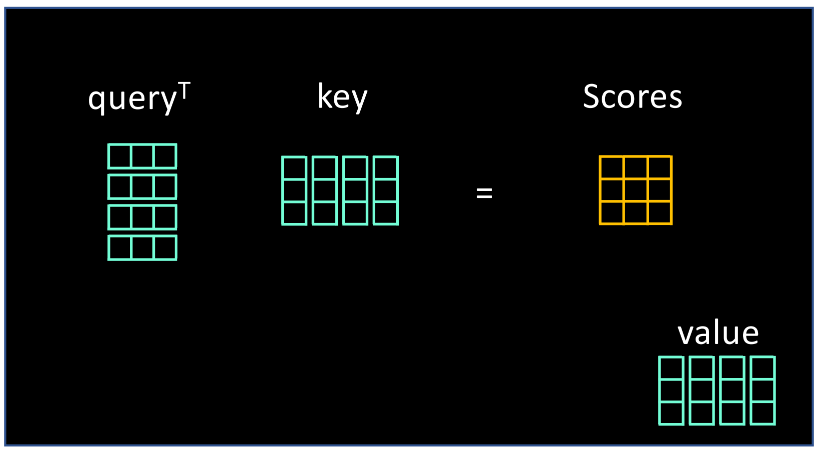
It associates each word in the input to other words.

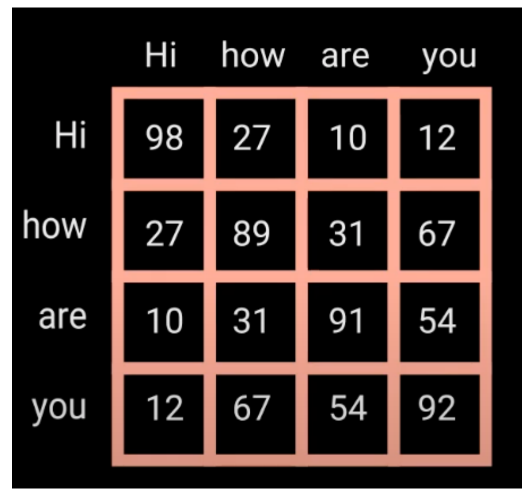
So in our example, it can learn to associate the word “you”, with “how” and “are”. It’s also possible that the model learns that words structured in this pattern are typically a question so respond appropriately.

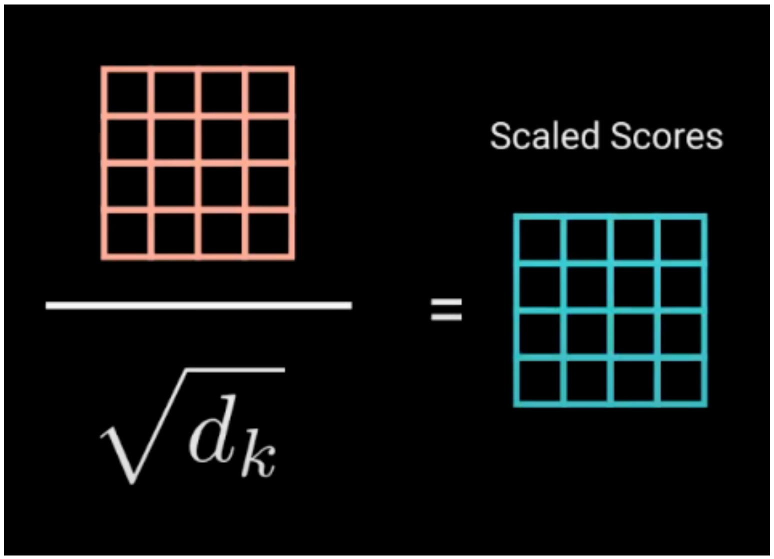


The query, key, and value vector through a linear layer, the queries and keys undergo a dot product matrix multiplication to produce a **Score Matrix**.

Score Matrix determines how much focus should a word be put on other words: each word has a score that corresponds to other words in the time-step. The higher the score the more focus.







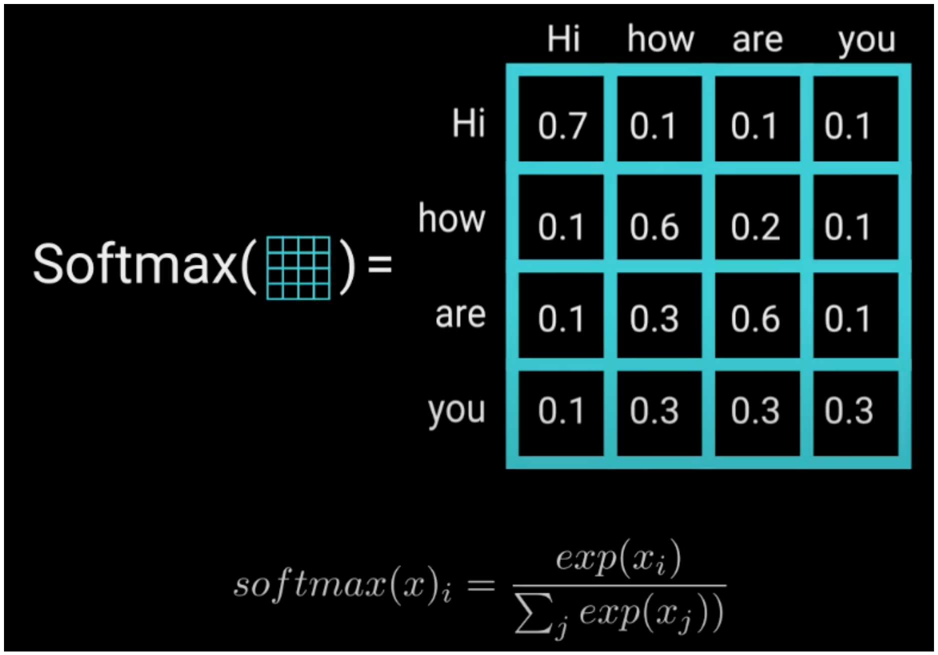
Scaling Down the Attention Scores

The scores get scaled down by getting divided by the square root of the dimension of query and key.

It allows more stable gradients, as multiplying values can have exploding effects.

d_k is the dimension of the key vectors.

Attention weights

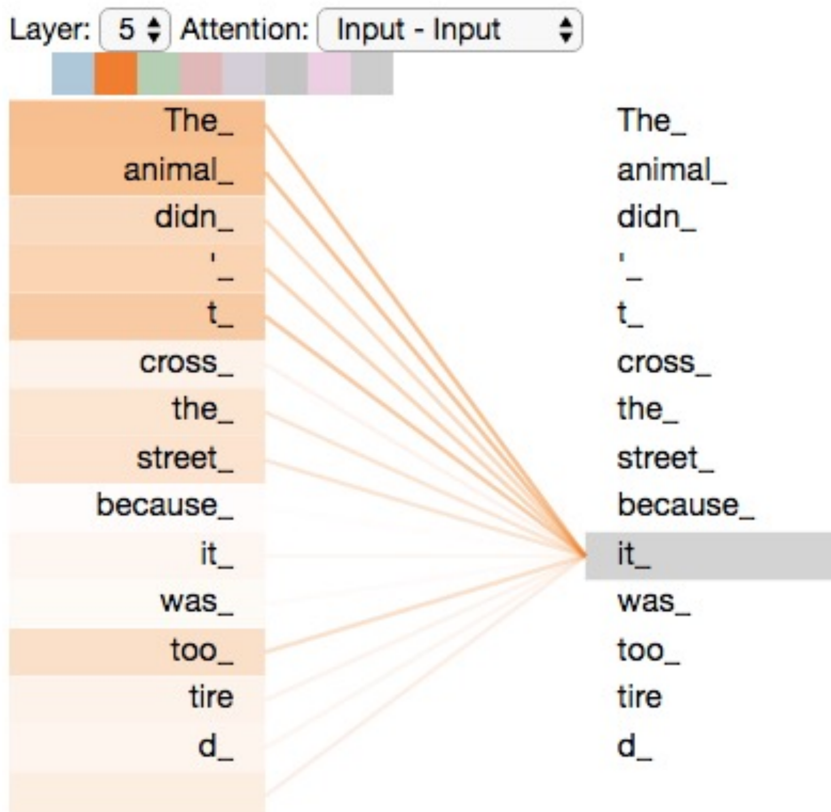


Softmax of the Scaled Scores

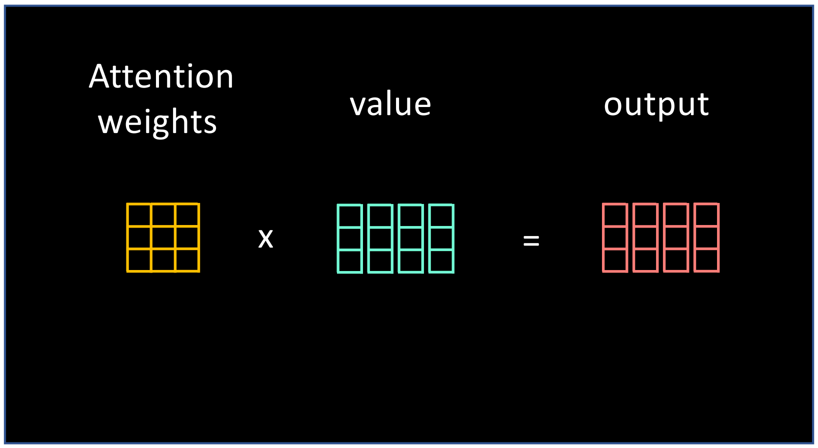
Softmax of the scaled score to get the attention weights: (probability values between 0 and 1).

Higher scores get heighten, and lower scores are depressed.

This allows the model to be more confident about which words to attend too.



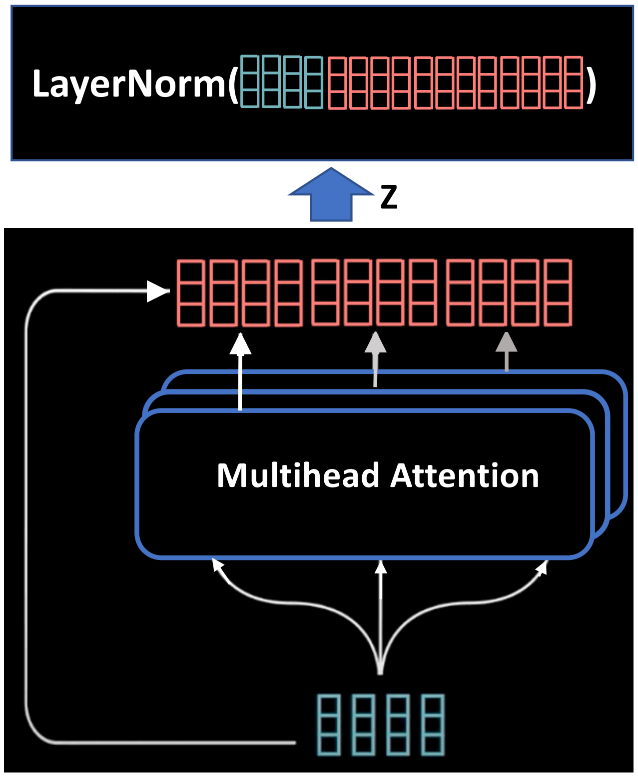
https://github.com/jessevig/bertviz



Multiply Softmax Output with Value vector

Attention weights are multiplied the value vector to get an output vector.

- The higher softmax scores will keep the value of words the model learns is more important.
- The lower scores will drown out the irrelevant words. Then you feed the output of that into a linear layer to process

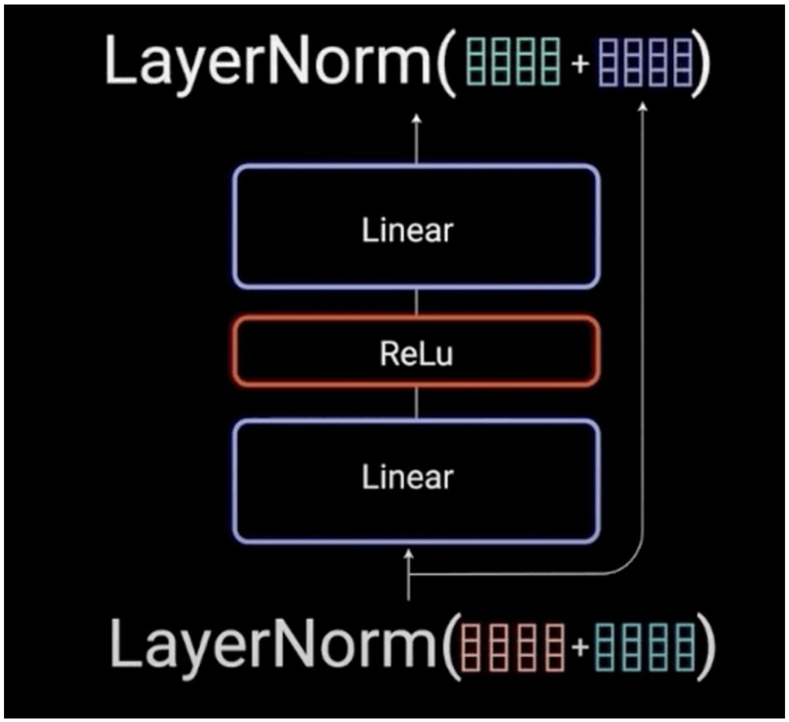


The Residual Connections, Layer Normalization, and Feed Forward Network

The multi-headed attention output vector is added to the original positional input embedding.

This is called a residual connection.

The output of the residual connection goes through a layer normalization



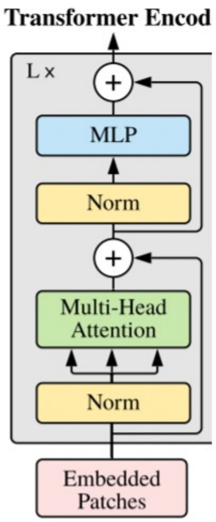
The normalized residual output gets projected through a pointwise feed-forward network for further processing.

The pointwise feed-forward network is a couple of linear layers with a ReLU activation in between.

The output of that is then again added to the input of the pointwise feed-forward network and further normalized.

- Residual connections help the network train.
- Layer normalizations stabilize the network reducing the training time.
- Pointwise feedforward layer project the attention outputs to a richer representation.

Summary



THEN, GIVEN THE SEQUENCE

$$t = [\,\text{'I'},\ \text{'am'},\ \text{'student'}\,]$$

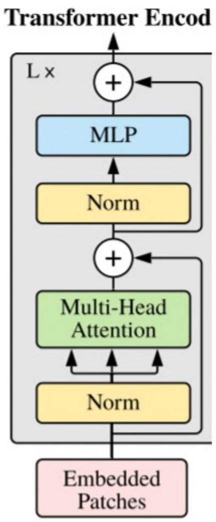
it is transformed to indices in a lexicon

$$I = [\,40,\ 100,\ 1521\,]$$

And embedded in vector of dimension d_K

$$x = \big[\,[\ ],\ [\ ],\ [\ ]\,\big] \quad d_K$$

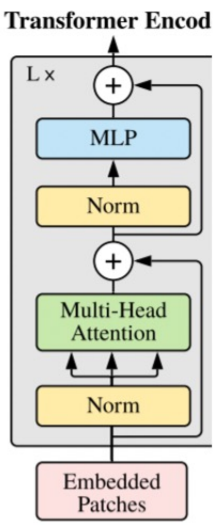
d_K is the dimension of all the vectors in the transformer



it is important to include positional information into the embedded tokens

Let p_t the vector of dimension d_K that codifies the t-th position, then

$$x_t \leftarrow x_t + p_t$$



Next, it is pased to the Attention Head

for de ith head we compute

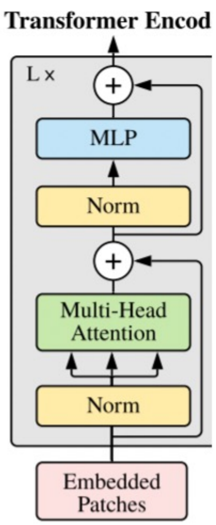
$$q_i = W_q^i x \qquad \text{(query)}$$

$$k_i = W_k^i x \qquad \text{(key)}$$

$$v_i = W_v^i x \qquad \text{(value)}$$

Then, we compute the score matrix:

$$S_i = q_i^T k = x^T W_q^{iT} W_k^i x$$



The score matrix can be seen as a crosscorrelation matrix:

$$S^i = \{X^T, X\}_{W_q^T W_K}$$

Then, it is normalized (scaled):

$$S^i \leftarrow \frac{S^i}{\sqrt{d_k}}$$

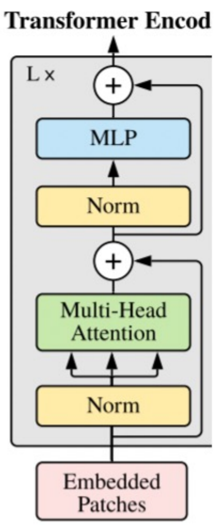
the internal vector's dimension

and

$$A^i = \mathrm{softmax}(S^i)$$

por renglones

the ith self attention matrix



Since we have H heads, we concatenate their outputs

original embedded data

$$z = [x, z_1, z_2, \dots z_H]$$

where

$$z_i = A^i V_i$$

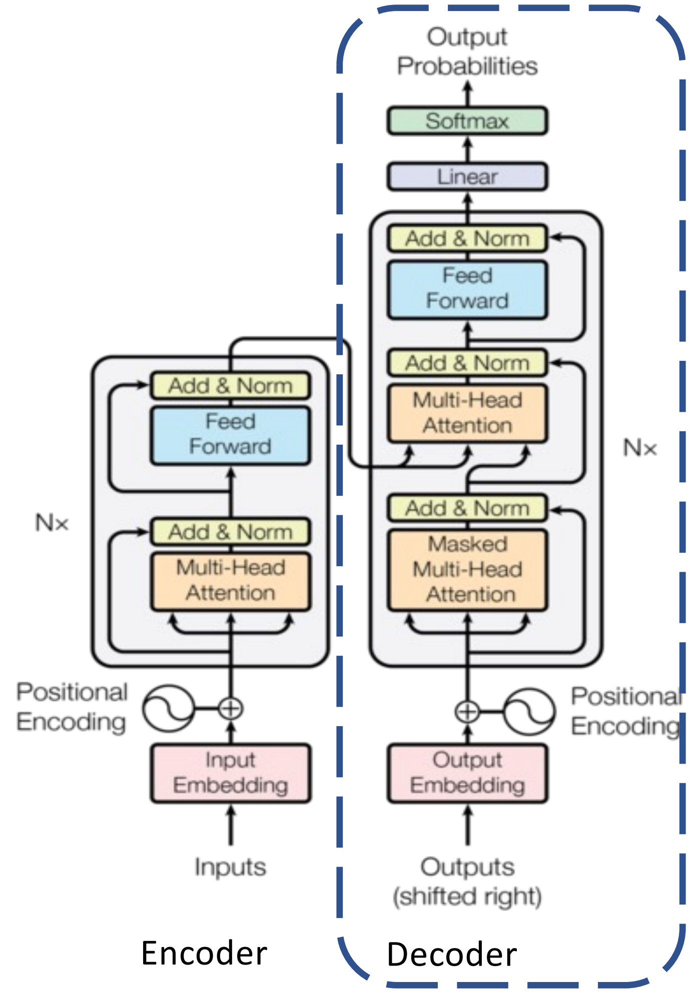
$$= \text{softmax}(X^T W_q^{i\,T} W_k^i X) W_v^i x$$

$$\underbrace{s\left(\frac{Q^T K}{\sqrt{d}}\right) V}$$

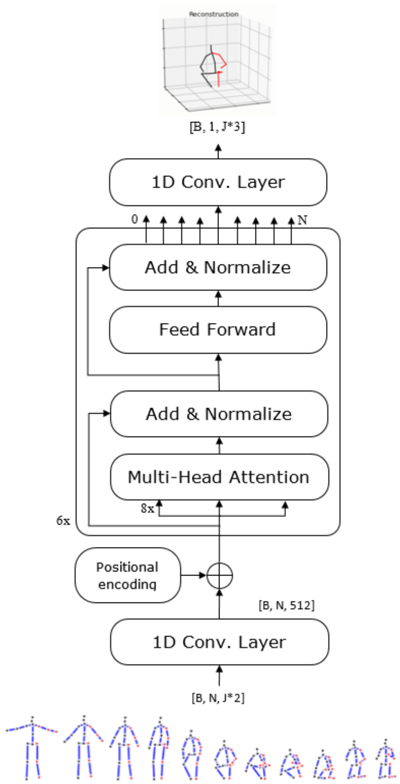
Note the dependency on x of A

DECODER

(Beyond the scope and interest of this talk)

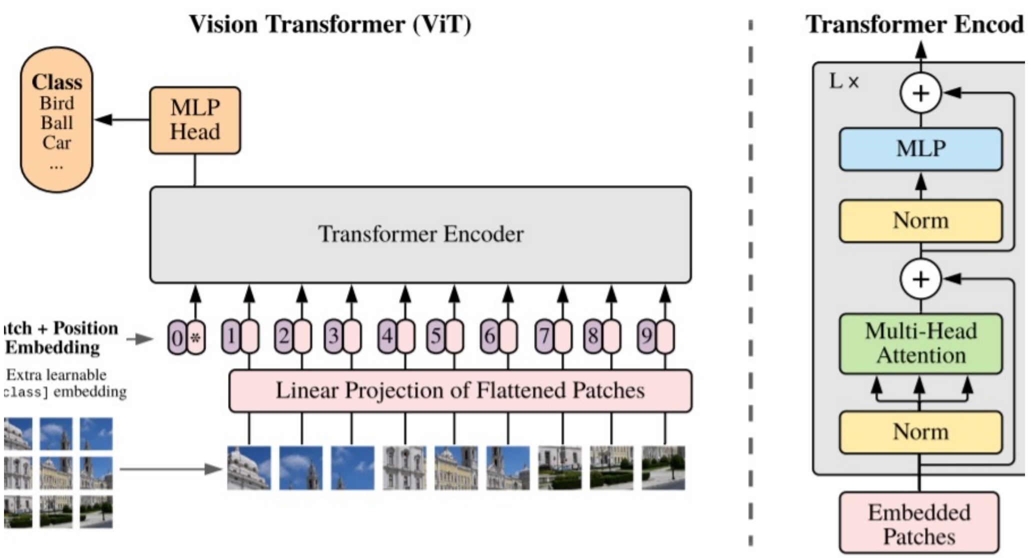


3D Human pose estimation



Adrian Llopart, LiftFormer: 3D Human Pose Estimation using attention models. CoRR, abs/2009.00348 (2020)

Visual Transformer: image classification



- Alexey Dosovitskiy et al. An image is worth 16x16 words: transformers for image recognition at scale. **ICLR 2021.** https://arxiv.org/pdf/2010.11929v1.p

- https://paperswithcode.com/paper/an-image-is-worth-16x16-words-transformers-1

TRANSFORMERS

Conclusion

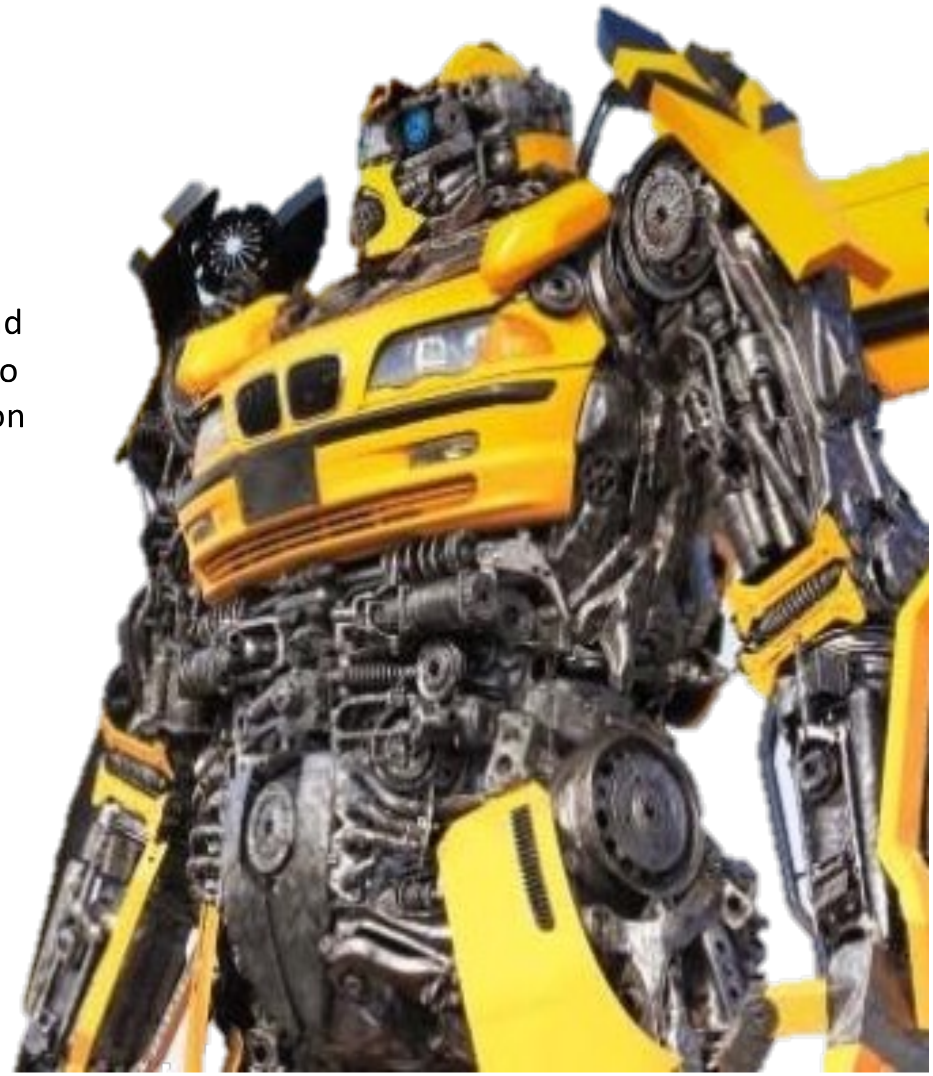
Growing interest in applying Transformer-based models to vision problems:

> Image classification, object detection, action recognition and segmentation, generative modeling, multimodal tasks, video processing (recognition and forecasting), and low-level vision (image super-resolution, image enhancement, and colorization).

An effective use in vision problems is Segment Anything Model (SAM).

Vision Transformers would be the base of “fundation models” for vison tasks

Transformer suffer from the need for large training databases, numerical instabilities, and a large number of parameters.



Mariano Rivera