

Deep learning

Transformers

2024/12/16

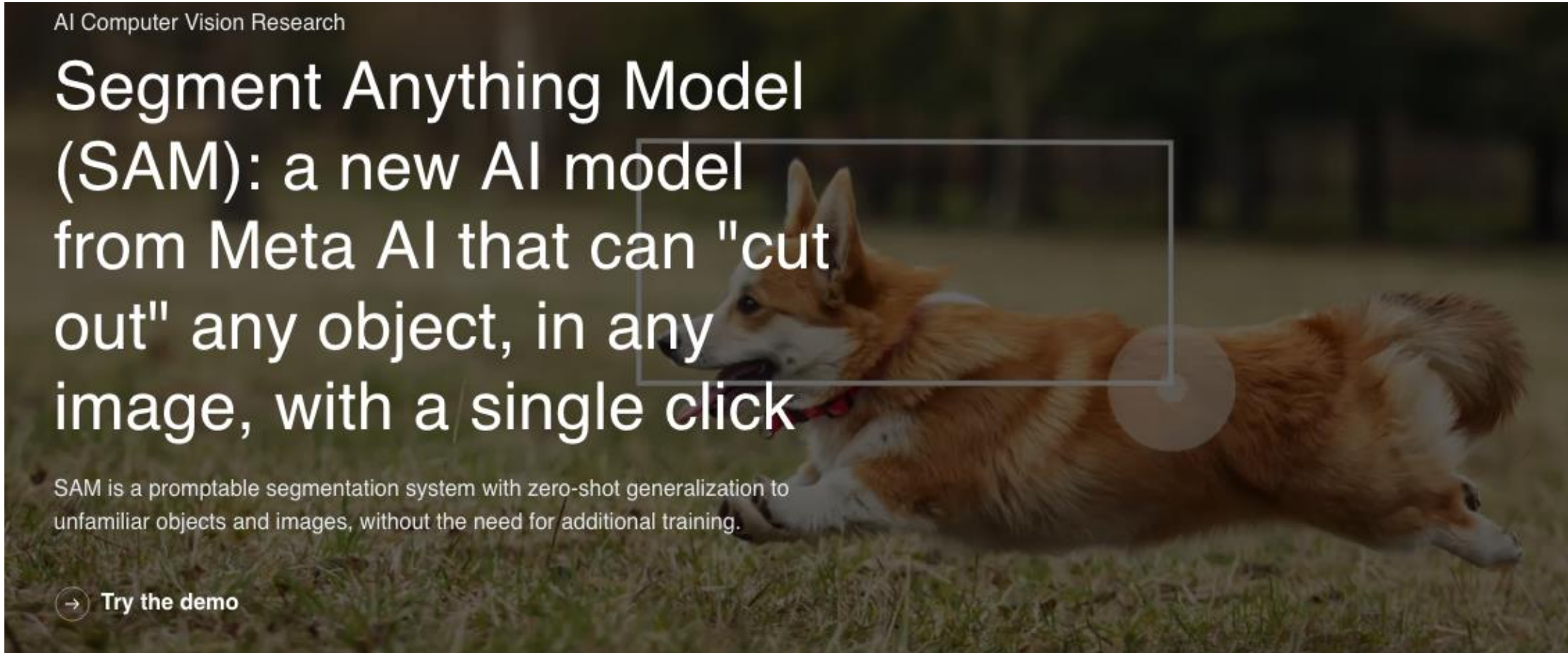
Jon Sparring,
Department of Computer Science

UNIVERSITY OF COPENHAGEN



Segment Anything

<https://segment-anything.com/>



AI Computer Vision Research

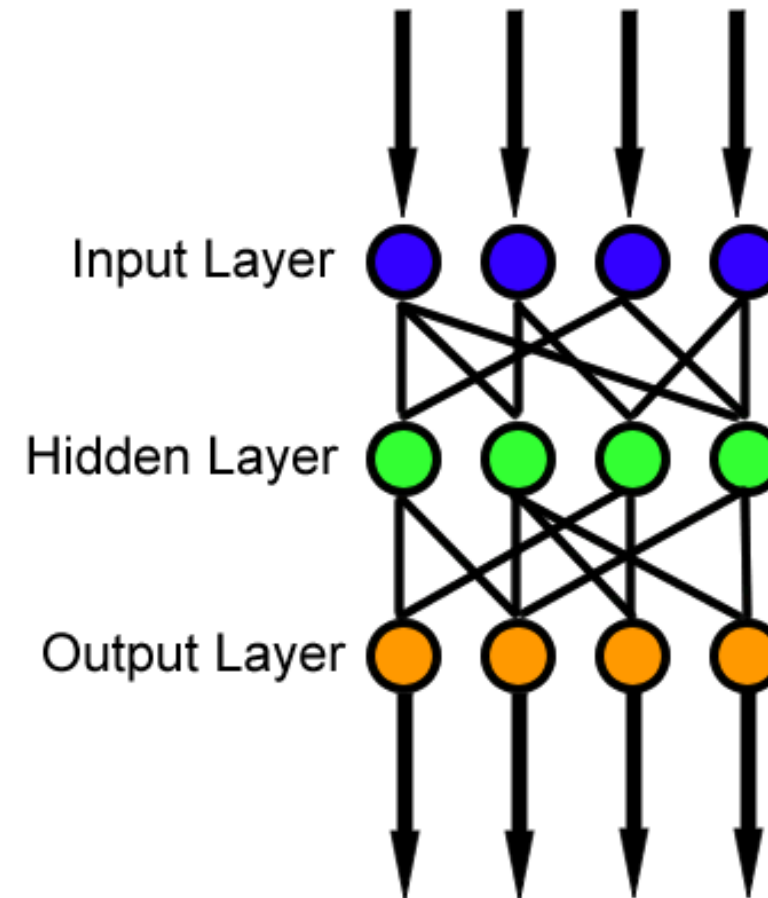
Segment Anything Model (SAM): a new AI model from Meta AI that can "cut out" any object, in any image, with a single click

SAM is a promptable segmentation system with zero-shot generalization to unfamiliar objects and images, without the need for additional training.

→ Try the demo

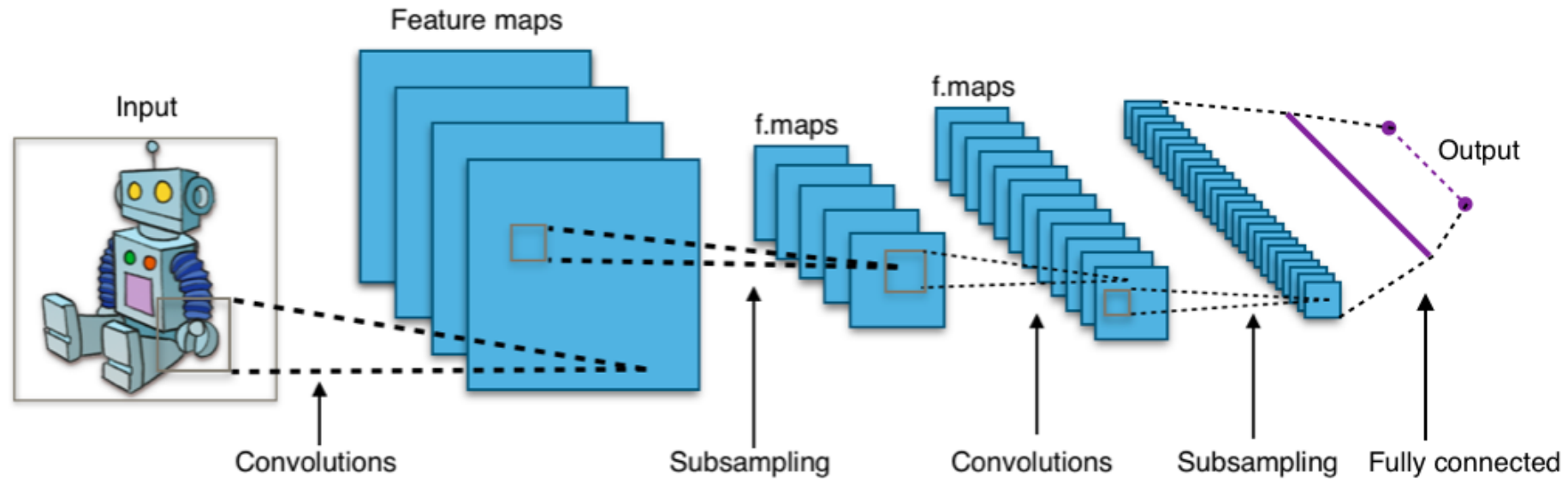
Feed forward neural networks

- Gauss, Legendre (around 1800):
Single layer, least squares
- McCulloch & Pitts (1940's):
Artificial neuron
- Rosenblatt, Joseph (around 1960):
Multilayer perceptron
- Linnainmaa (1970's):
Backpropagation
- Bengio et al (early 2000's): Deep
learning



- By Paskari at the English-language Wikipedia, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=146663611>

Convolutional neural networks

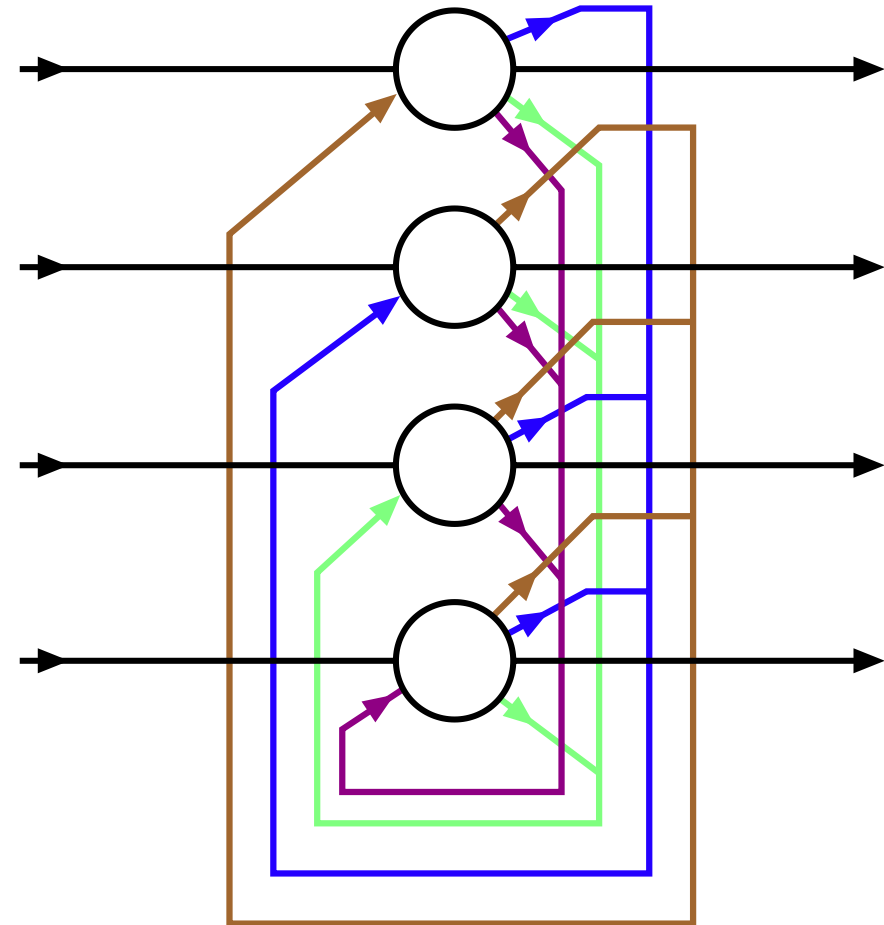


- Hubel & Wiesel (1950's): visual cortex studies
- Fukushima (1969): CNN + ReLU, hand-designed kernels
- Fukushima (1979), LeCun (1987): training kernels
- Oh & Jung (2004): GPU implementation

By Aphex34 - Own work, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=45679374>

Recurrent neural networks

- Cajal (early 1900's): Recurrence in cerebellar cortex
- McCulloch & Pitts: Recurrent neuron
- Rosenblatt (1960's), Hopfield (1982): Hebbian learning
- Hochreiter & Schmidhuber (1995): Long short-term memory (LSTM)
- Schuster et al (1997): Bidirectional recurrent neural networks (BRNN)



By Zawersh, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=37811881>

Recurrence

Consider: $x_t, \alpha \in \mathbb{R}$

Running average: $y_0 = x_0, y_t = \alpha y_{t-1} + (1 - \alpha)x_t$

Exampel:

$$x = [7 \quad 0 \quad 3 \quad 0 \quad 1 \quad 8 \quad 7 \quad 3 \quad 10 \quad 0]$$

$$y_{\alpha=0.3} = [7 \quad 2 \quad 3 \quad 1 \quad 1 \quad 6 \quad 7 \quad 4 \quad 8 \quad 2]$$

$$y_{\alpha=0.7} = [7 \quad 5 \quad 5 \quad 4 \quad 3 \quad 4 \quad 5 \quad 5 \quad 7 \quad 5]$$

$$y_0 = x_0$$

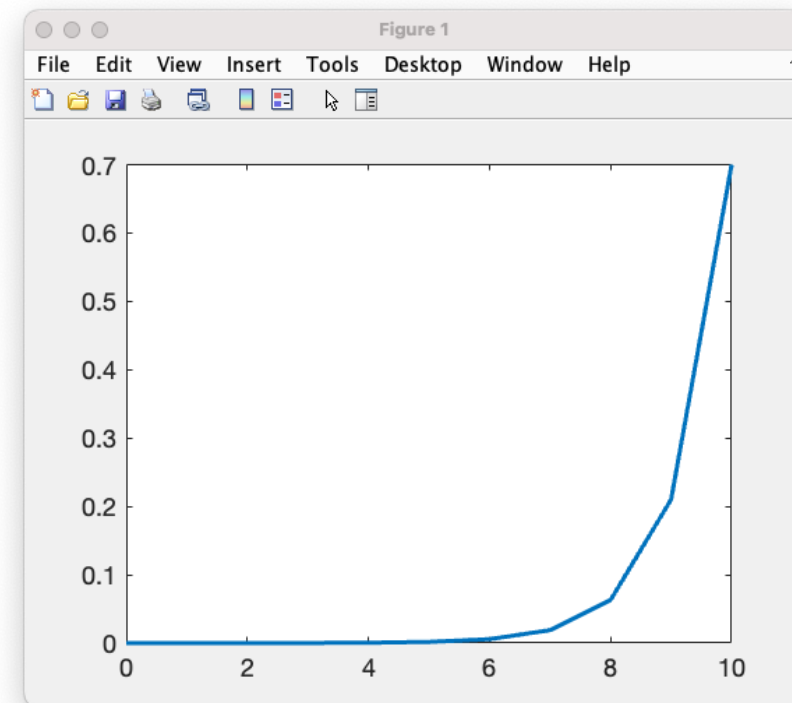
$$y_1 = \alpha y_0 + (1 - \alpha)x_1 = \alpha x_0 + (1 - \alpha)x_1$$

$$y_2 = \alpha y_1 + (1 - \alpha)x_2 = \alpha^2 x_0 + \alpha(1 - \alpha)x_1 + (1 - \alpha)x_2$$

$$y_3 = \alpha y_2 + (1 - \alpha)x_3 = \alpha^3 x_0 + \alpha^2(1 - \alpha)x_1 + \alpha(1 - \alpha)x_2 + (1 - \alpha)x_3$$

...

$$y_t = (1 - \alpha) \sum_{i=0}^t \alpha^{t-i} x_i$$



Long short-term memory (LSTM)

https://en.wikipedia.org/wiki/Long_short-term_memory

$$x = [7 \ 0 \ 3 \ 0 \ 1 \ 8 \ 7 \ 3 \ 10 \ 0]$$

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

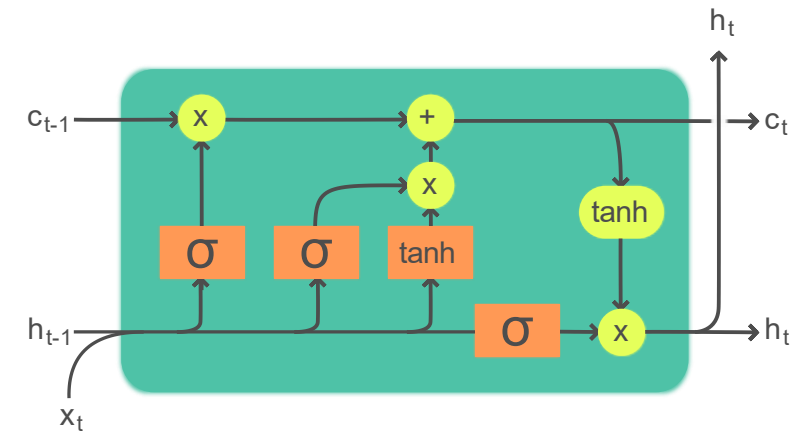
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

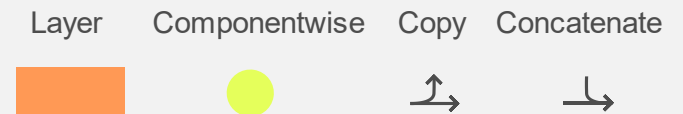
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \sigma_h(c_t)$$

- $x_t \in \mathbb{R}^d$: input vector to the LSTM unit
- $f_t \in (0, 1)^h$: forget gate's activation vector
- $i_t \in (0, 1)^h$: input/update gate's activation vector
- $o_t \in (0, 1)^h$: output gate's activation vector
- $h_t \in (-1, 1)^h$: hidden state vector also known as output vector of the LSTM unit
- $\tilde{c}_t \in (-1, 1)^h$: cell input activation vector
- $c_t \in \mathbb{R}^h$: cell state vector
- $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$: weight matrices and bias vector parameters which need to be learned during training



Legend:



- By Guillaume Chevalier - File:The_LSTM_Cell.svg, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=109362147>

Transformer networks

- Cho ea, Sutskever ea (2014): seq2seq
- Vaswani ea (2017): Attention is all you need
- Dosovitskiy ea (2020): An Image is Worth 16x16 Words ...

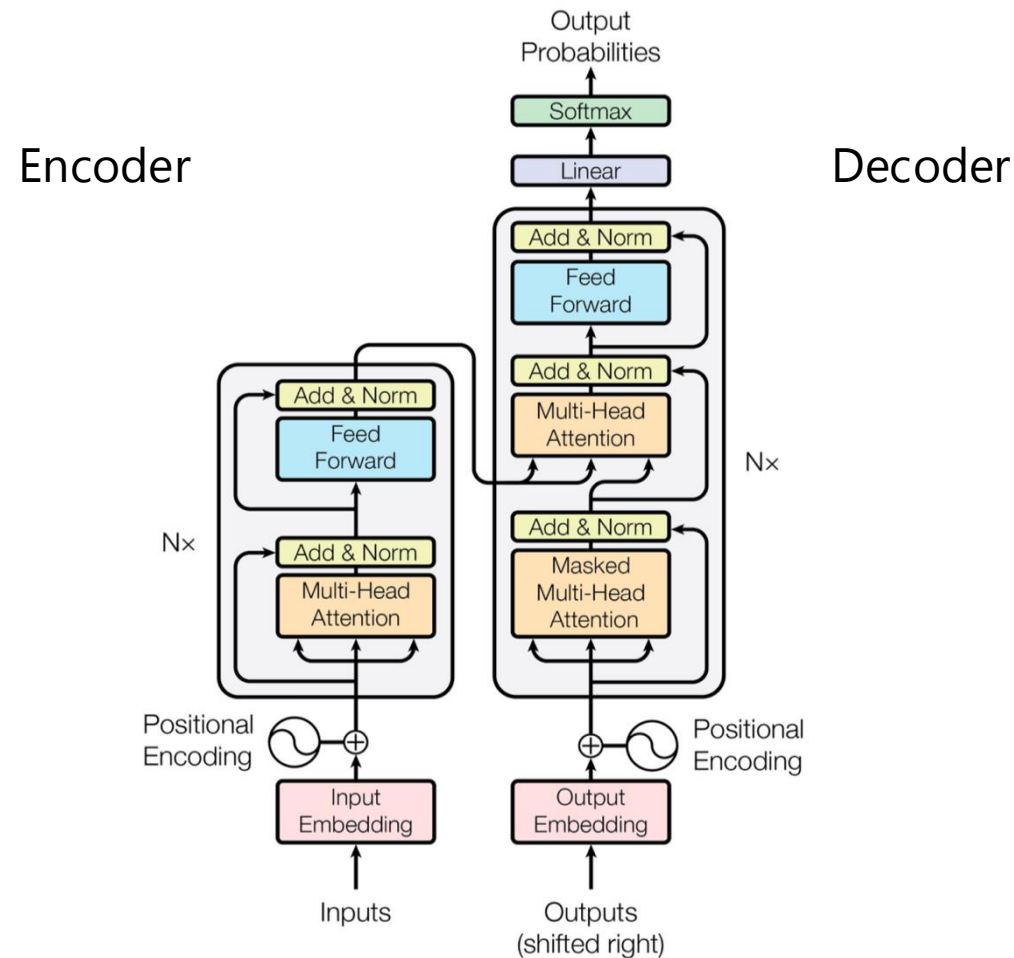
Vision Transformers

Transformers | Davide Coccoimini | 2021

By Davide Coccoimini - Own work, CC BY-SA 4.0,
<https://commons.wikimedia.org/w/index.php?curid=110678226>

Vaswani: Attention Is All You Need

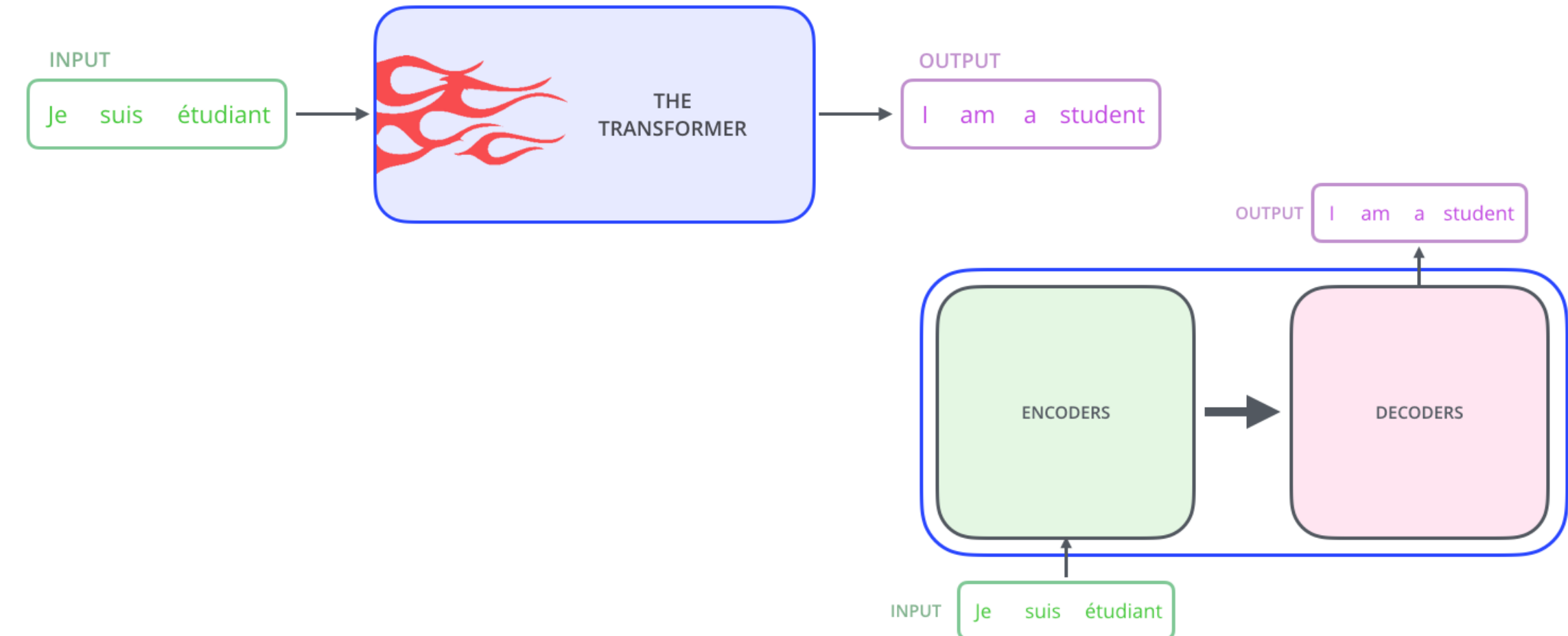
Dosovitskiy et al (2020): An Image is Worth 16x16 Words...



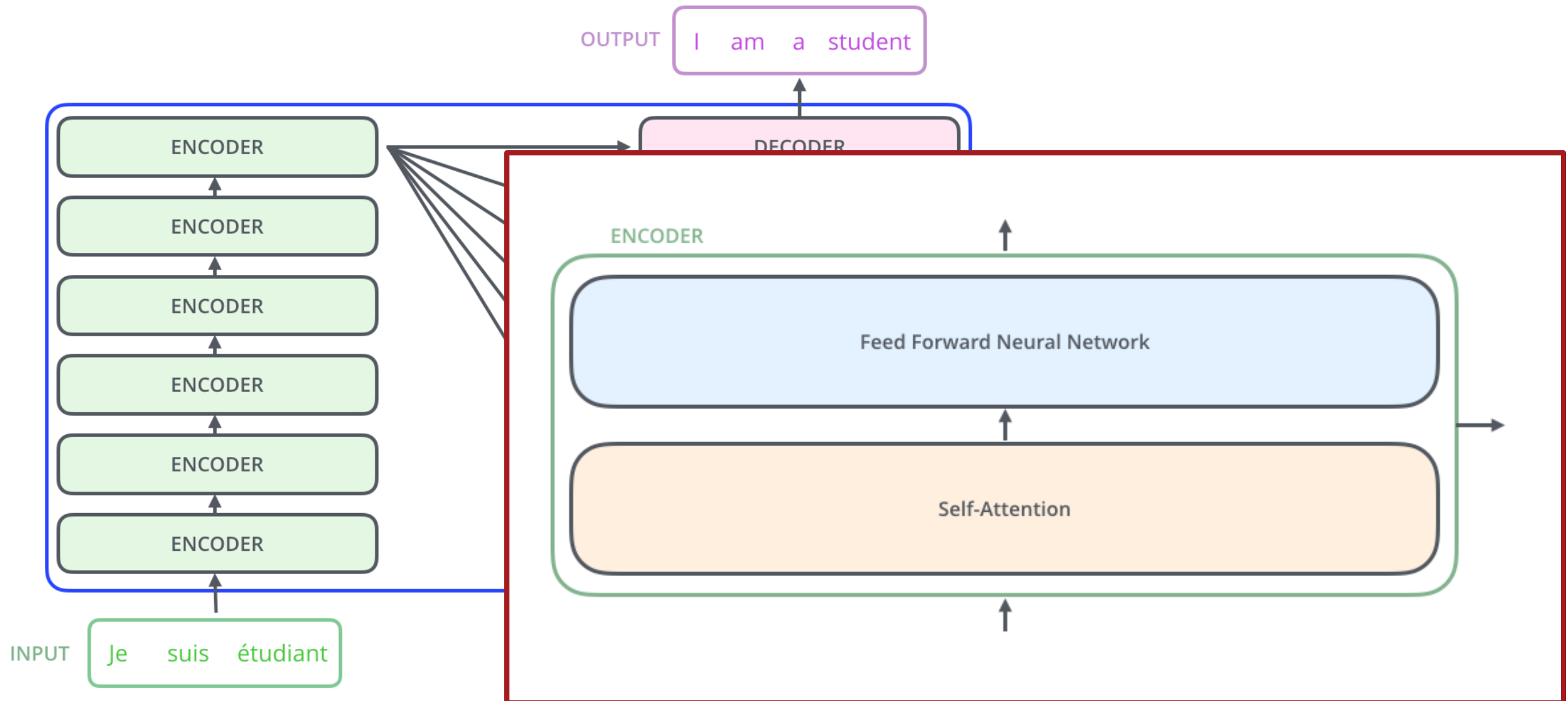
The Illustrated Transformer

<http://jalammar.github.io/illustrated-transformer/>

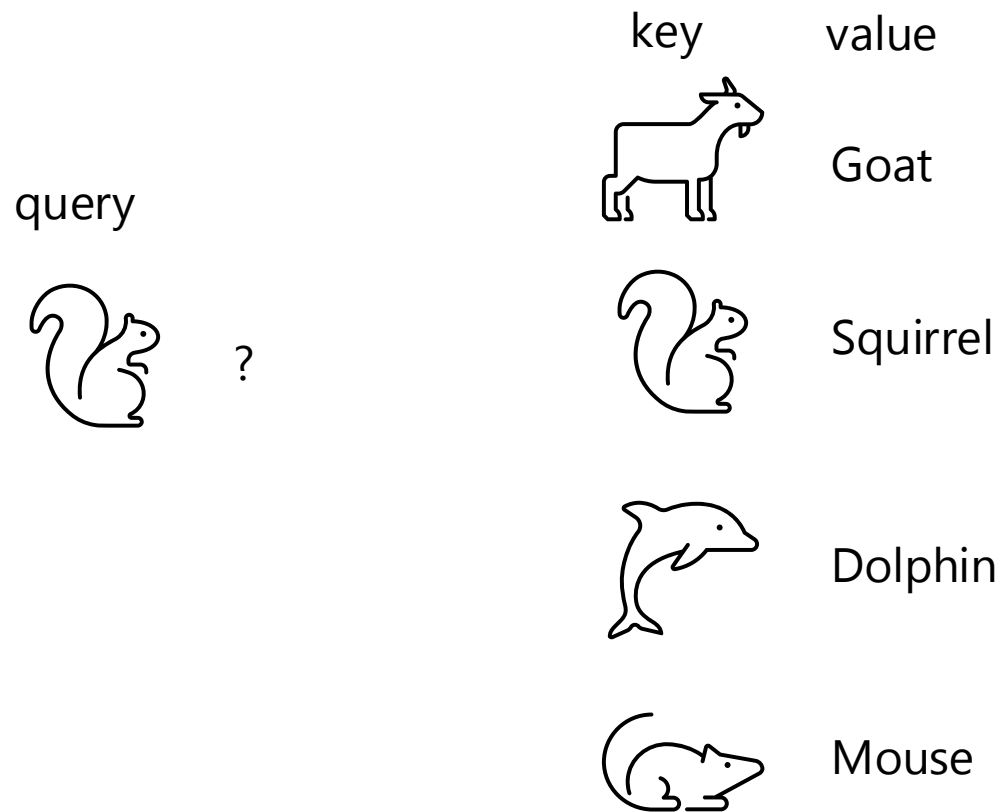
Transformers in the context of languages:



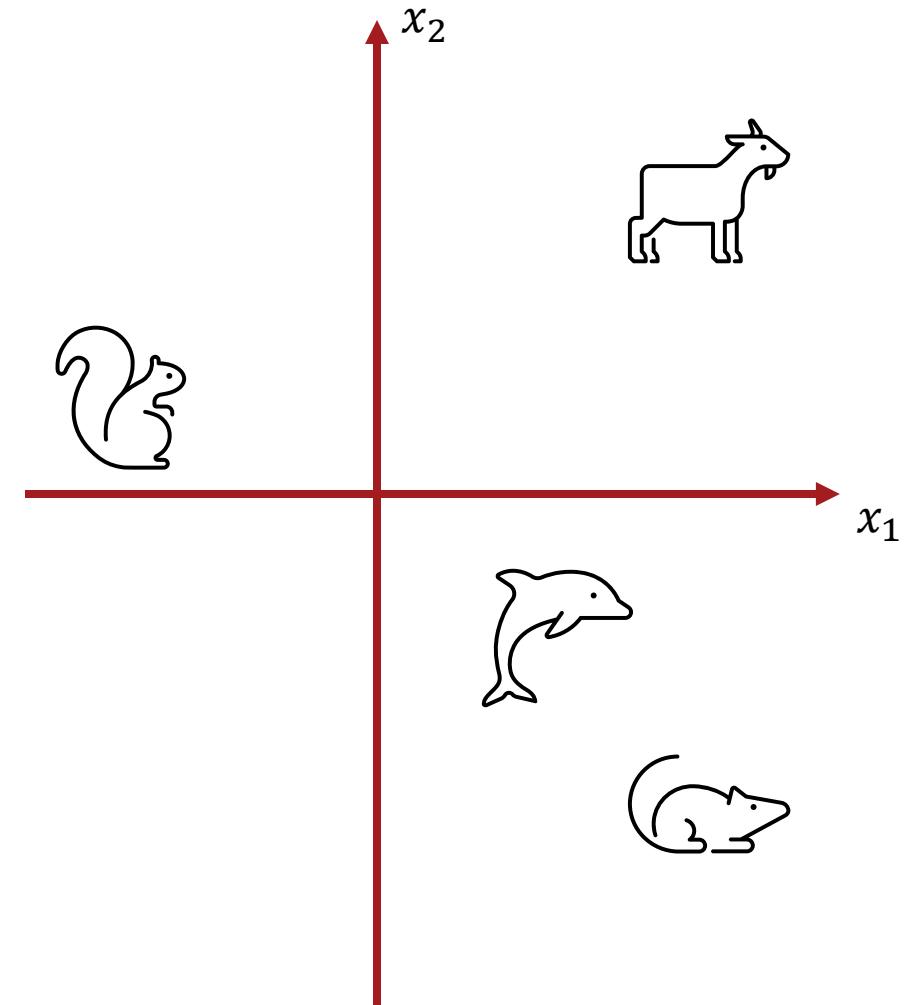
Encoder (and decoder) are stacks of identical units

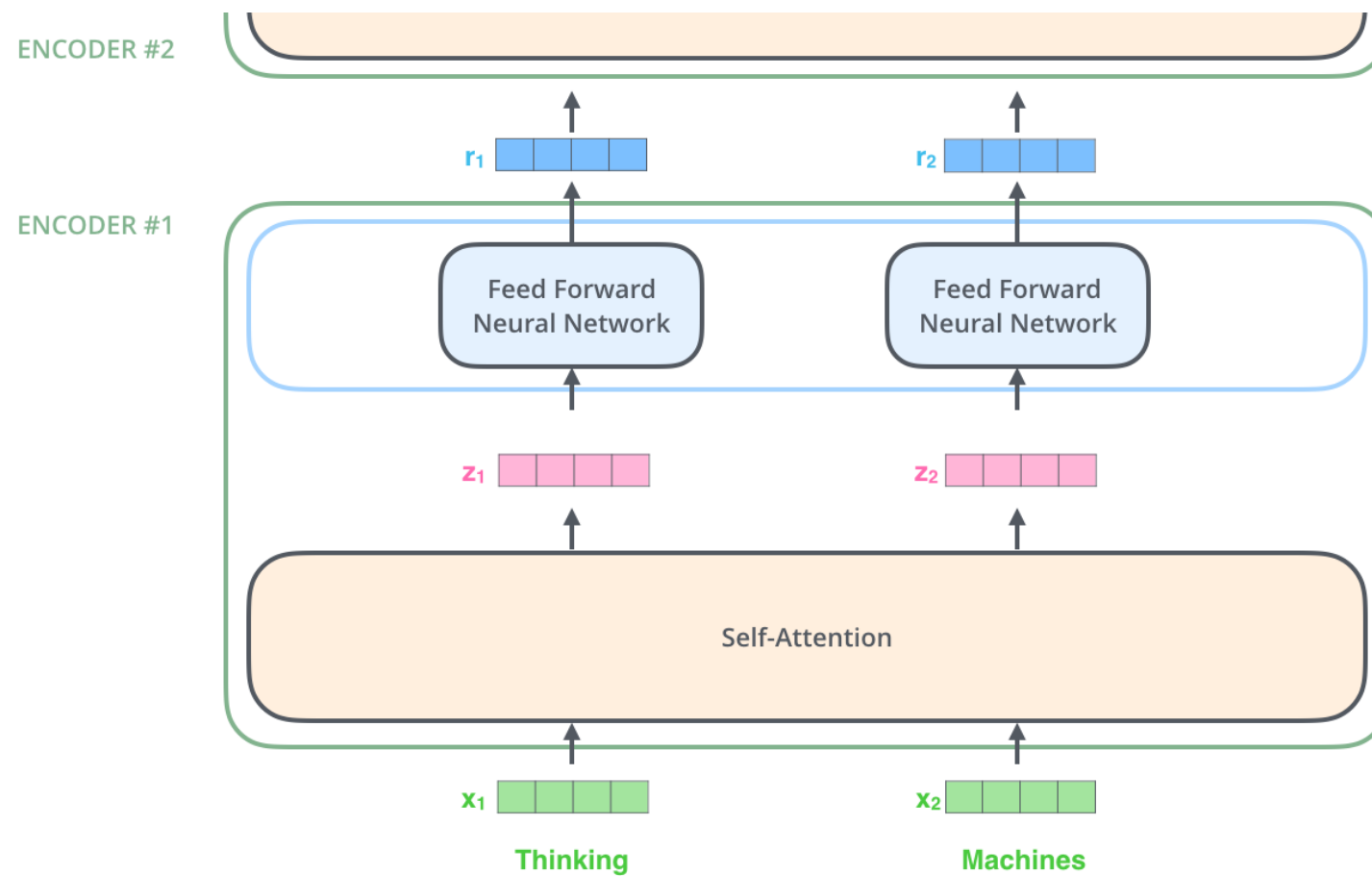


Classification by Query, Key, Value and Tokens

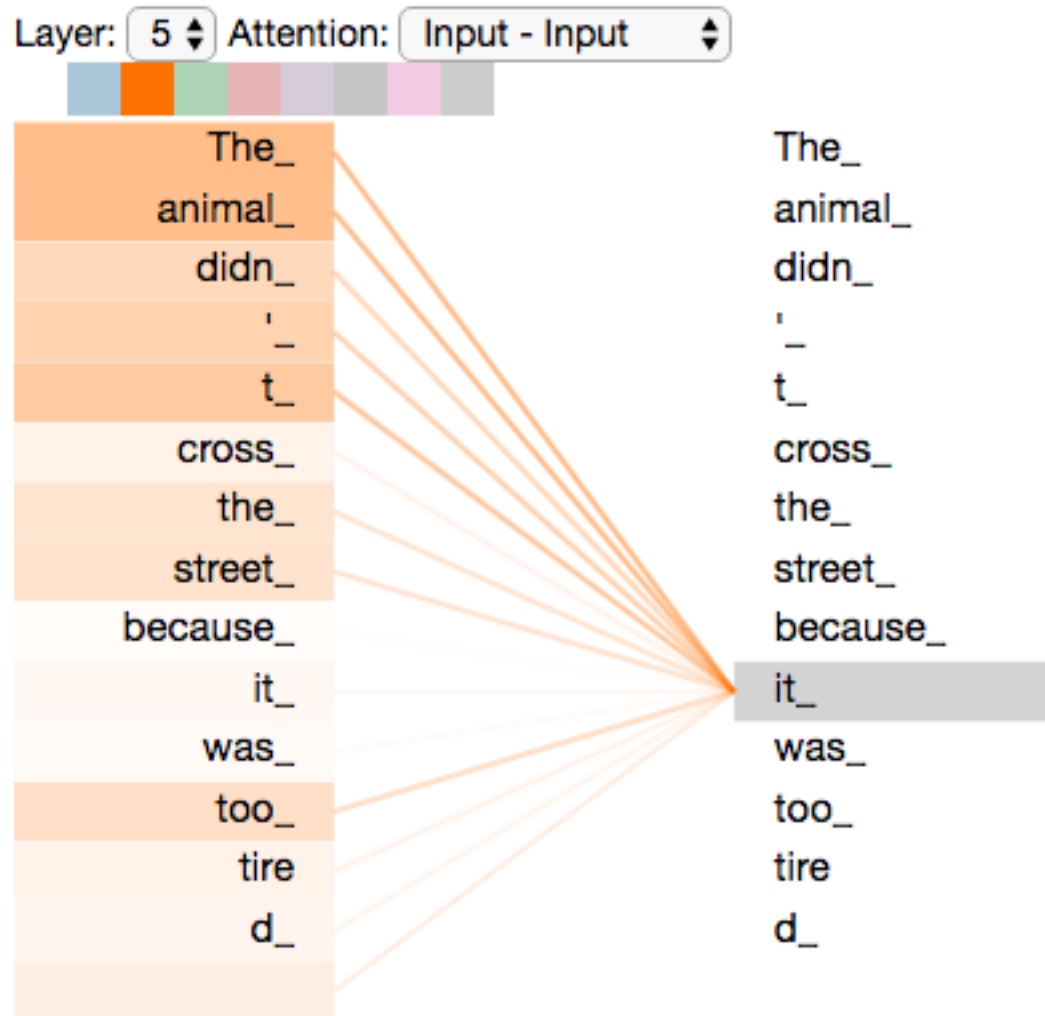


Tokens as embedding





Self attention: The animal didn't cross the street because it was too tired



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

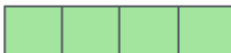
Matrix form

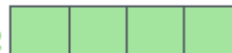
Input

Thinking

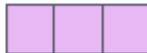
Machines

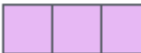
Embedding

x_1 

x_2 

Queries

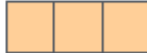
q_1 

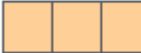
q_2 



W^Q

Keys

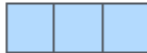
k_1 

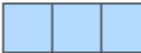
k_2 



W^K

Values

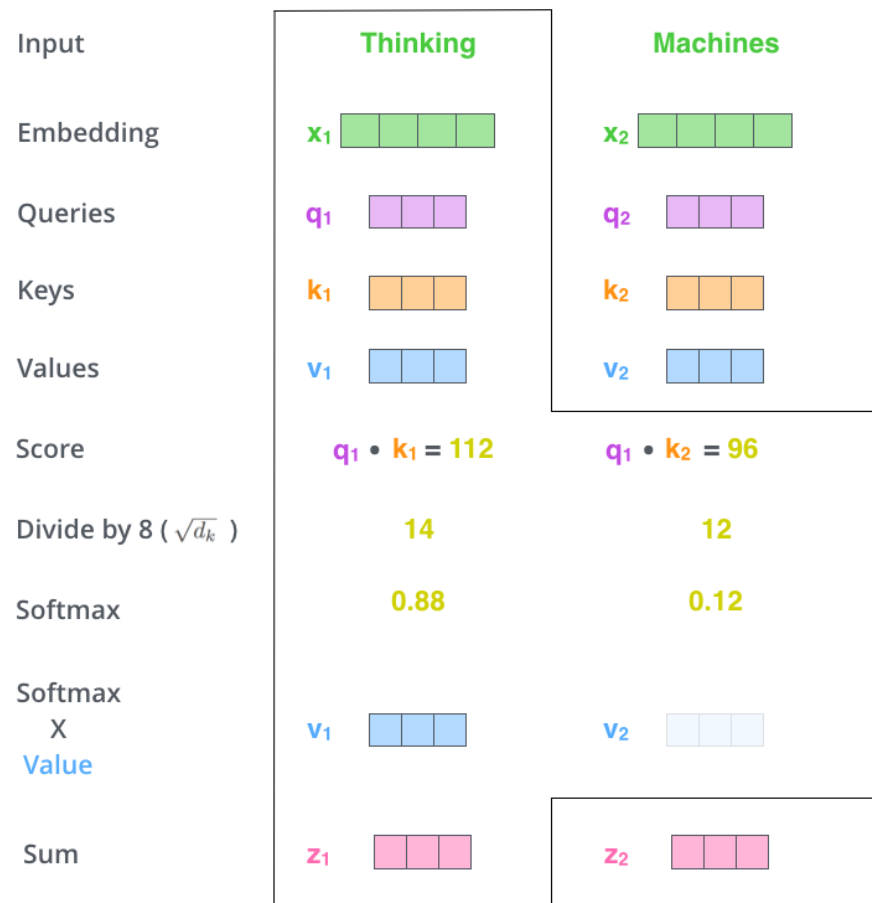
v_1 

v_2 



W^V

Self attention, single head



$$X \times W^Q = Q$$

$$X \times W^K = K$$

$$X \times W^V = V$$

$$\text{softmax} \left(\frac{Q \times K^T}{\sqrt{d_k}} \right) \times V = Z$$

Papers with code

<https://paperswithcode.com/method/transformer>

The screenshot shows the 'Transformer' method page on the 'paperswithcode.com' website. The page includes a search bar, navigation links for 'Browse State-of-the-Art', 'Datasets', 'Methods', and 'More'. The main title 'Transformer' is prominently displayed, followed by the text 'Introduced by Vaswani et al. in [Attention Is All You Need](#)'. A descriptive paragraph explains that the Transformer is a model architecture that eschews recurrence and relies on an attention mechanism. Below the text are two buttons: 'Read Paper' and 'See Code'. On the right side, there is a detailed diagram of the Transformer architecture. The diagram illustrates the flow from 'Inputs' through 'Input Embedding' and 'Positional Encoding' into an encoder stack (labeled 'Nx'). The encoder stack consists of 'Multi-Head Attention' and 'Feed Forward' sub-layers, each followed by an 'Add & Norm' residual connection. The output of the encoder is then passed to a decoder stack (also labeled 'Nx'), which includes 'Masked Multi-Head Attention', 'Multi-Head Attention' (attending to the encoder output), and 'Feed Forward' sub-layers, each followed by an 'Add & Norm' residual connection. The final output goes through a 'Linear' layer and a 'Softmax' layer to produce 'Output Probabilities'.

Source: [Attention Is All You Need](#)

[Read Paper](#) [See Code](#)

https://production-media.paperswithcode.com/methods/new_ModalNet-21.jpg