Tranformers for Computer Vision

Alberto Albiol Colomer

UNIVERSIDAD
POLITECNICA
DE VALENCIA

Contents

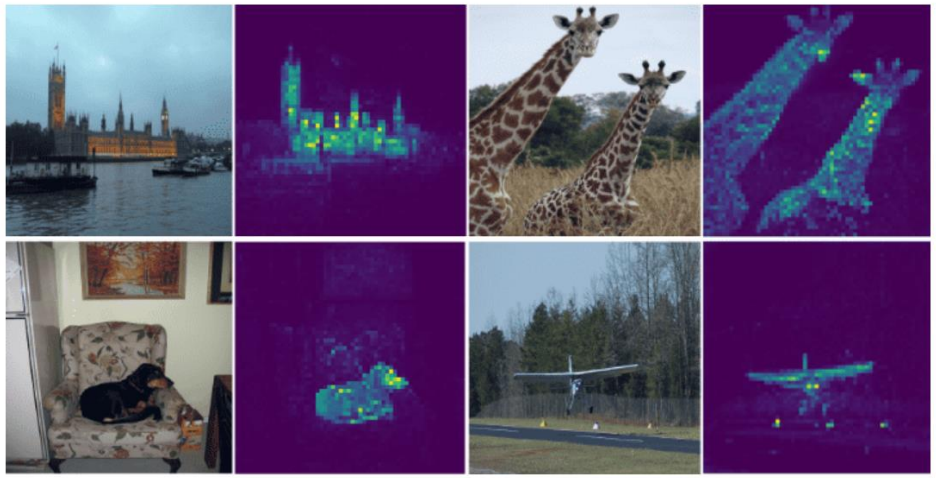
- Fundamentals of the attention mechanism
- Self-attention
- Multihead Self-attention
- Positional encoding
- Masked self-attention
- Transformer layer

(1967) All you need is **love**!

(2017) All you need is....
ATTENTION!!

(Today)
All I need
is **YOUR** attention

Fundamentals of attention



DINO head attention visualization. Source DINO [6]

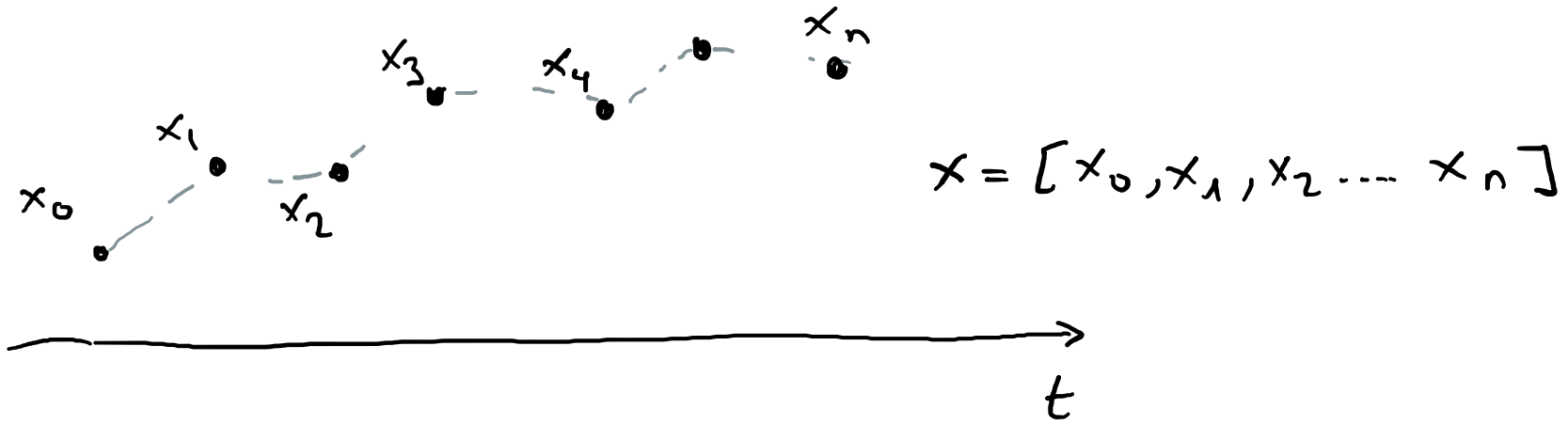
- Attention compares to how humans focus on specific parts of information when processing complex tasks.
- Attention helps to introduce context to improve data representation and meaning

Importance of context

"The detective found the missing key in the suspect's pocket."

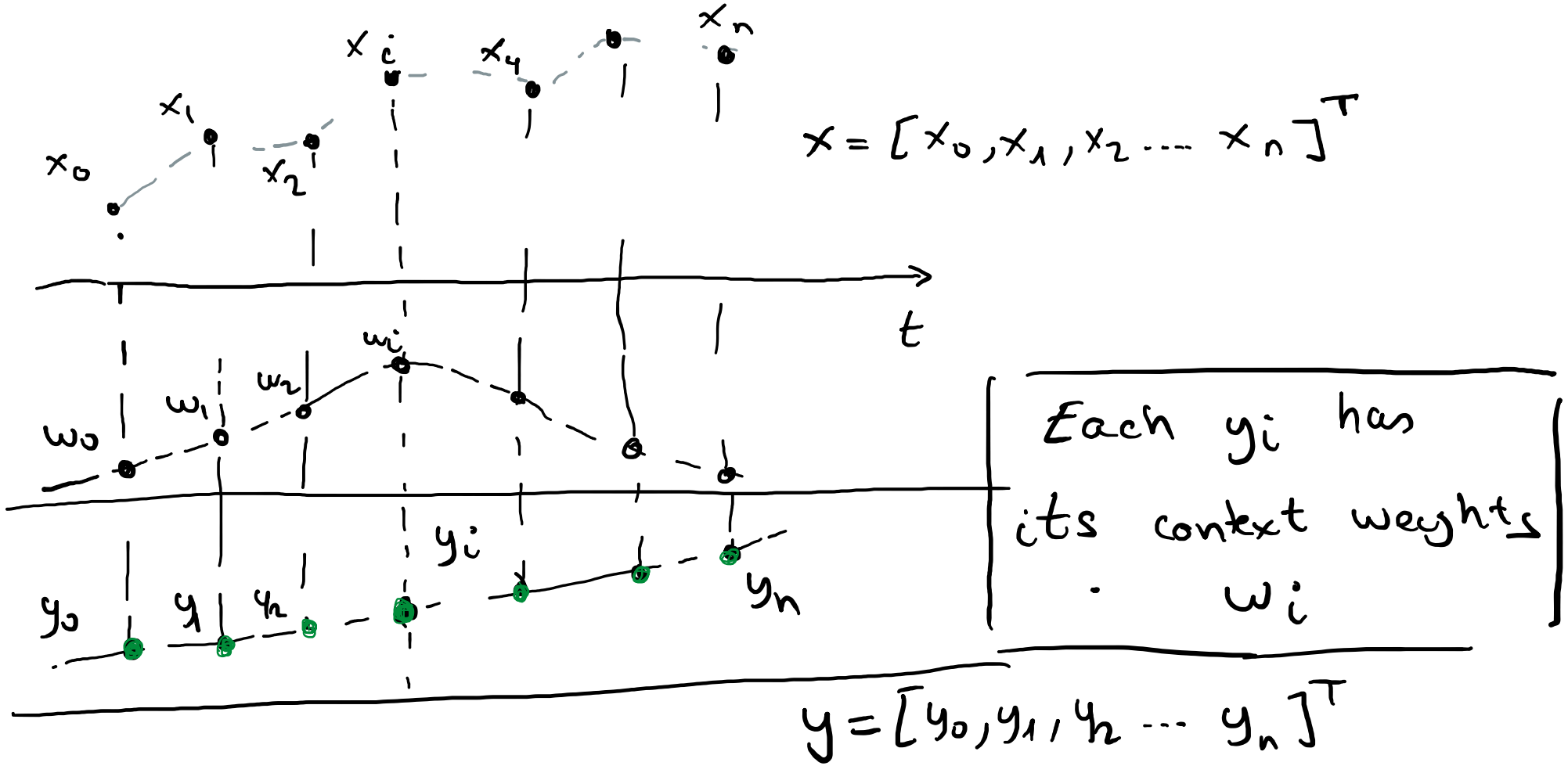
- What does key mean in this sentence?
  - It can be a key that opens a door
  - It can be something that incriminates the suspect

Attention in time series data



- Context (nearby points) can be used to reduce noise -> i.e. have more signal

Using context to enhance signal

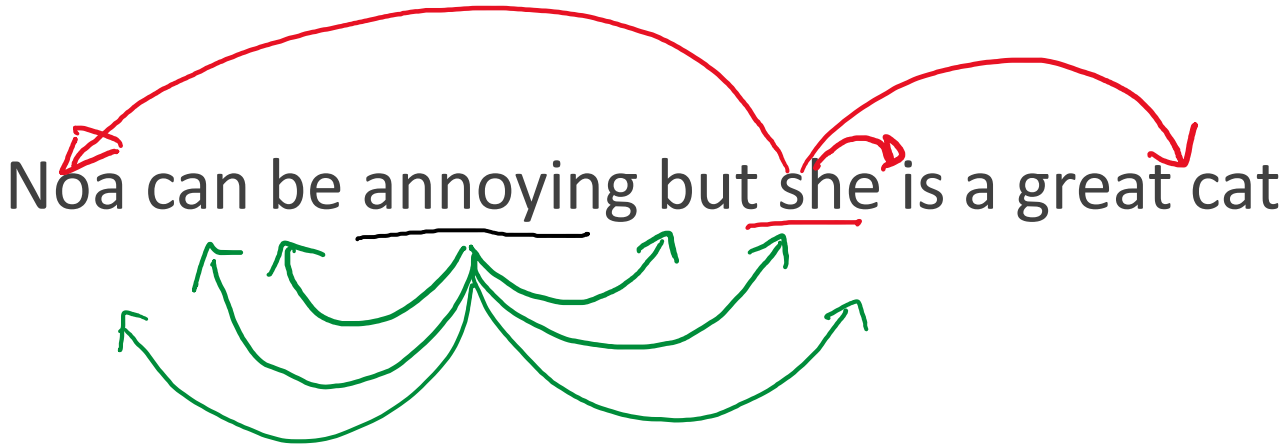


Using context to enhance signal

$$x = [x_0, x_1, x_2 \cdots x_n]^T \Rightarrow y = [y_0, y_1, y_2 \cdots y_n]^T$$

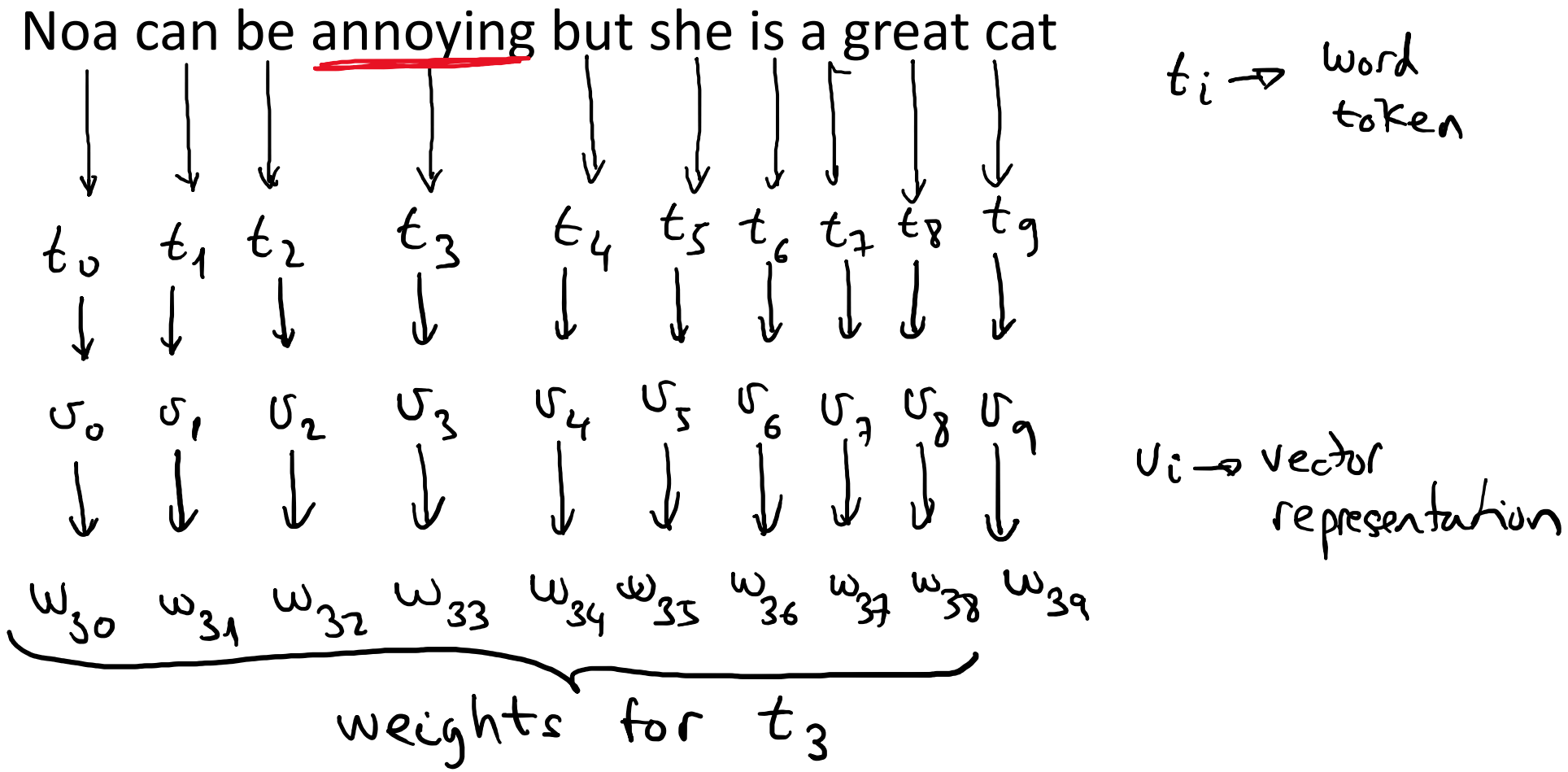
- Introducing the information from context we **TRANSFORM** the original signal into another one with less noise.
- The number of elements of the input and output signals is the same

Attention in text data



- **Proximity** here is not the best option to apply attention (green arrows)
- Attention should be based on **meaning, grammar, linguistics**.... (red arrows)

How to compute weights for attention



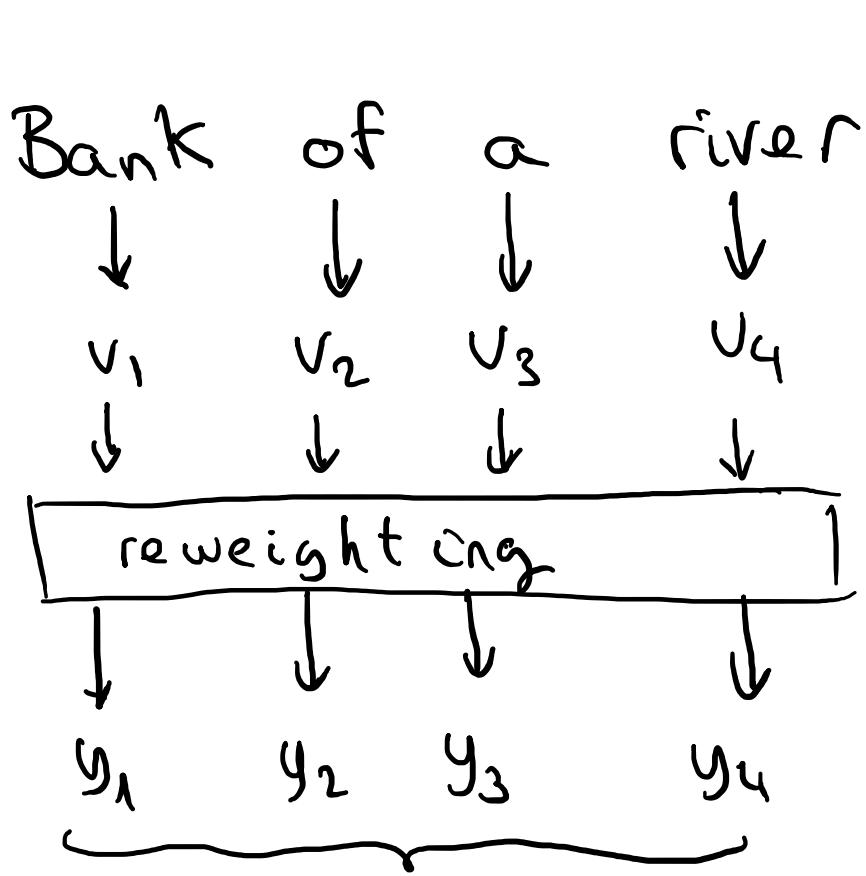
Calculating similarity weights

- Similar words tend to have vectors that point in similar directions. For example, the vectors for "king" and "queen" might be in similar directions because they share a semantic relationship.
- But similarity can be for many different reasons:
  - King, Queen share: Power, country, family, wealth, gender....
  - Examples of other similar words:
    - Army, son, poor, ….
  - Examples of not similar words:
    - Dog, school, sun,

IMPORTANT: Similarity can be restricted to only a few aspects

Calculating weights

- A common measure for vector alignment is dot product



$w_{11} = v_1 \cdot v_1$
$w_{12} = v_1 \cdot v_2$
$w_{13} = v_1 \cdot v_3$
$w_{14} = v_1 \cdot v_4$

norm softmax

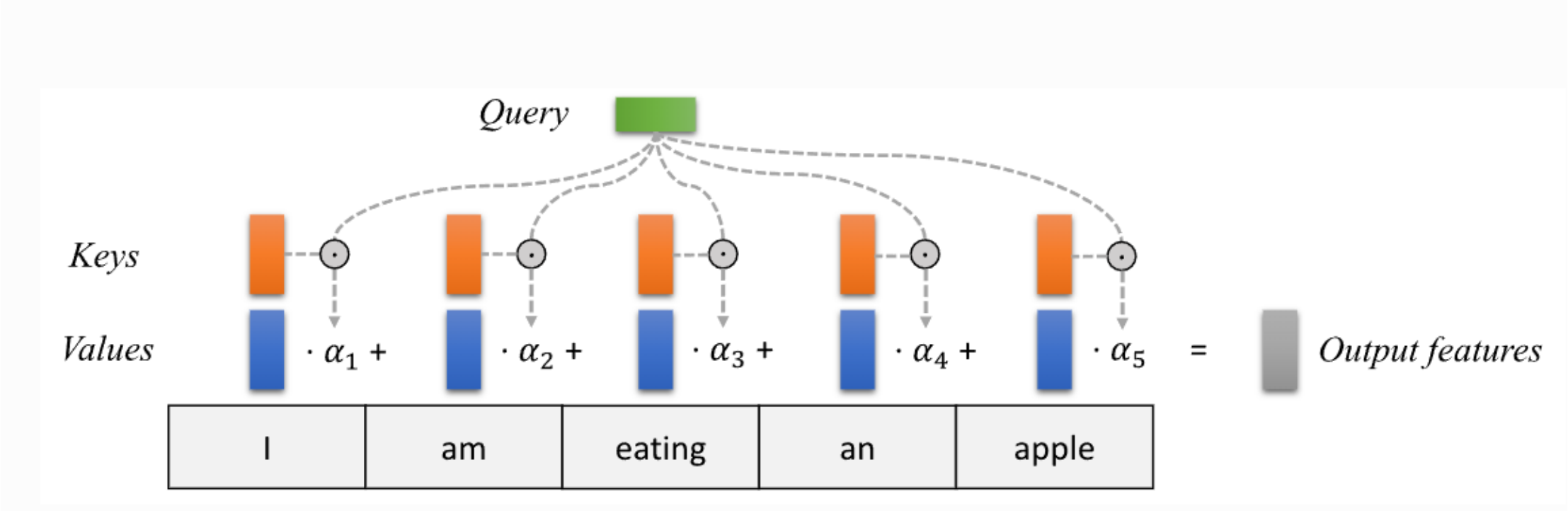
w_{11} w_{12} w_{13} w_{14}

$y_1 = w_{11} v_1 + w_{12} v_2 + w_{13} v_3 + w_{14} v_4$

$y_4 = w_{41} v_1 + w_{42} v_2 + w_{43} v_3 + w_{44} v_4$

have more context !! ⇒ river ⇒ bank

Self attention



Important remarks about attention so far

- Proximity was not important (although it can be)
- Order does not matter (although it really does)
- Weights were not trained (no much flexibility)
- Similarity between words is global, i.e. not restricted to a few aspects of meaning

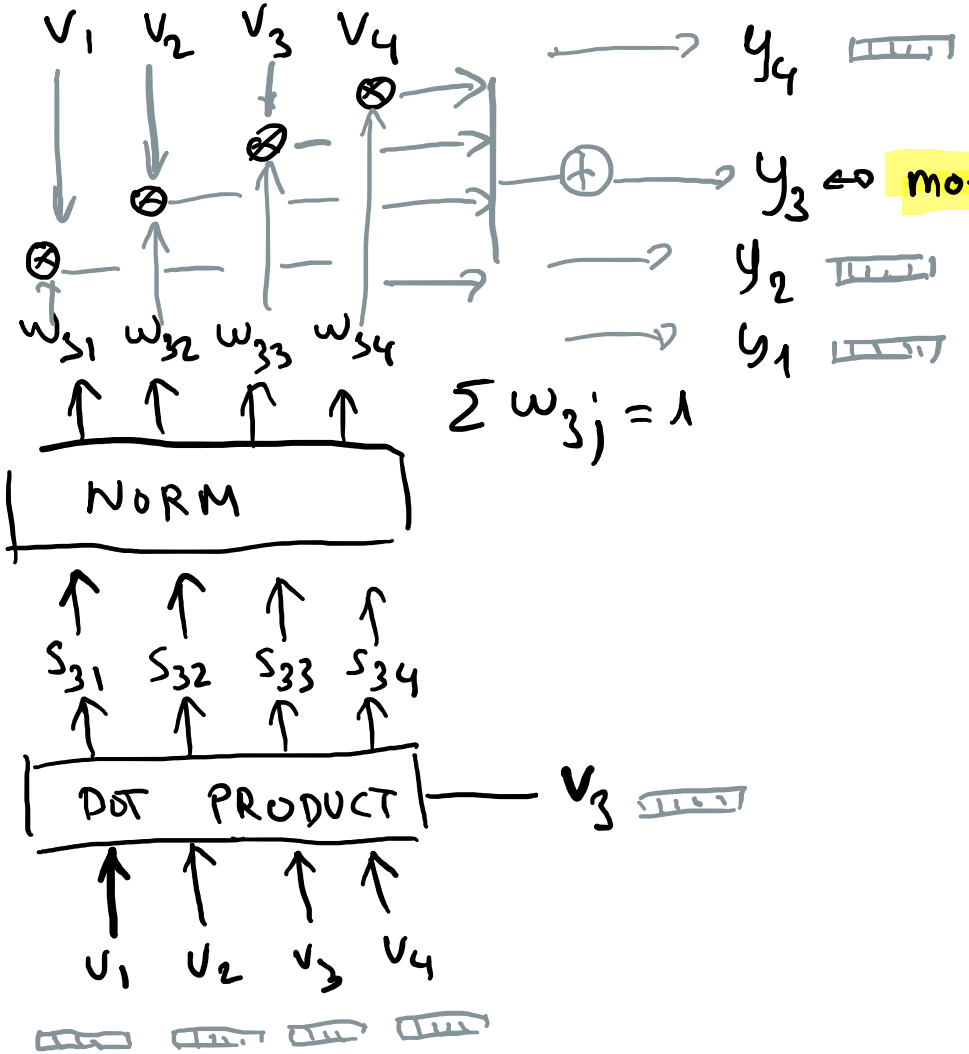
But....

What we have seen is the basis of Self Attention

Other self-attencion score functions

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Attention so far

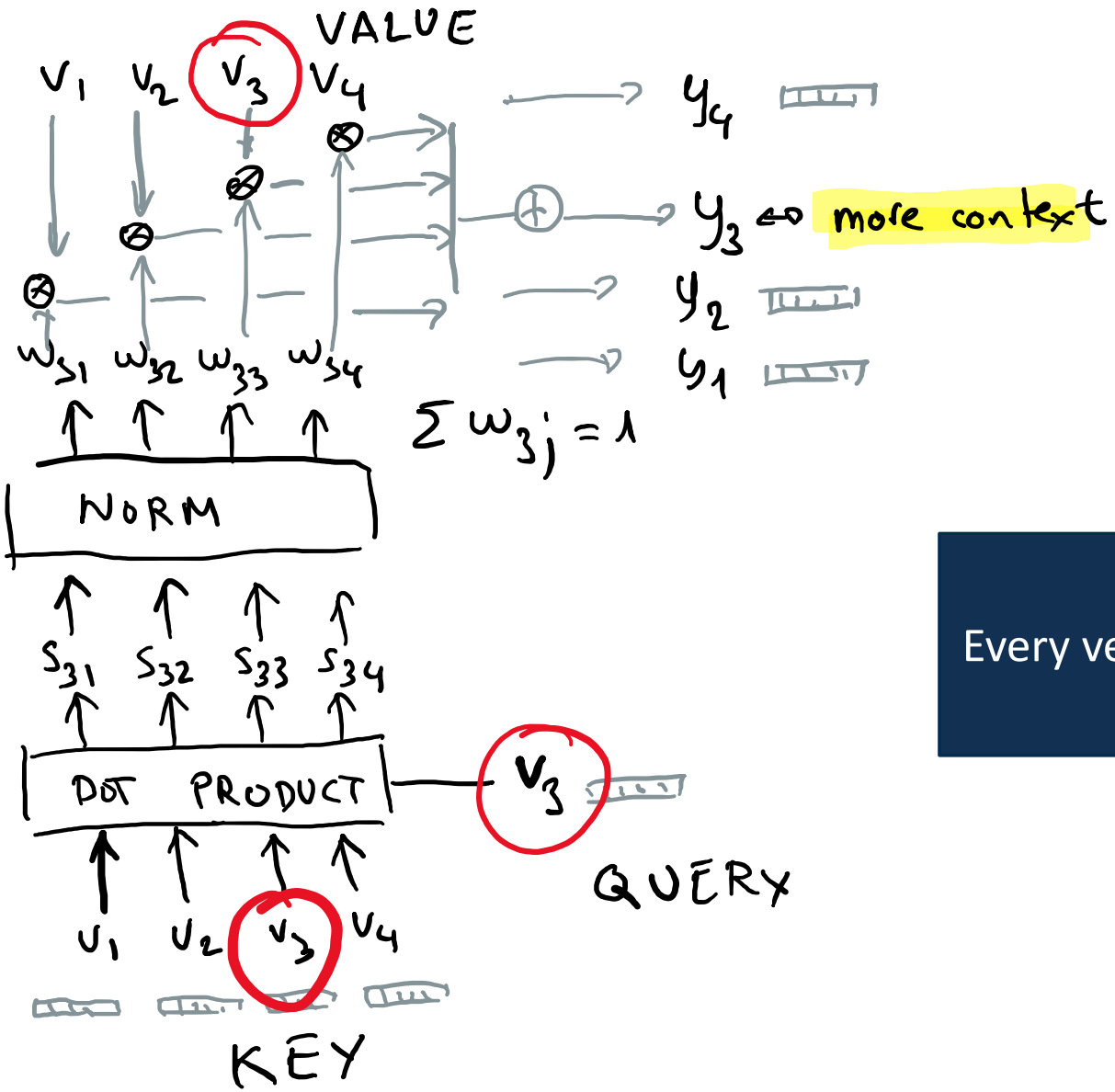


$$y_3 = w_{31} v_1 + w_{32} v_2 + w_{33} v_3 + w_{34} v_4$$

Self-attention in transformers

- We want to query/extract specific context for each word
- The kind of context that we want must be *trainable/learnable*
- The new representation after extracting context should also be *trainable/learnable*

Attention so far

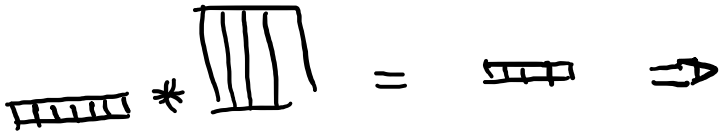


Every vector appears in three places!!

Improved attention

- Add parameters -> increase expression of the model

FOR KEYS → $v_i M_K = v_{Ki}$



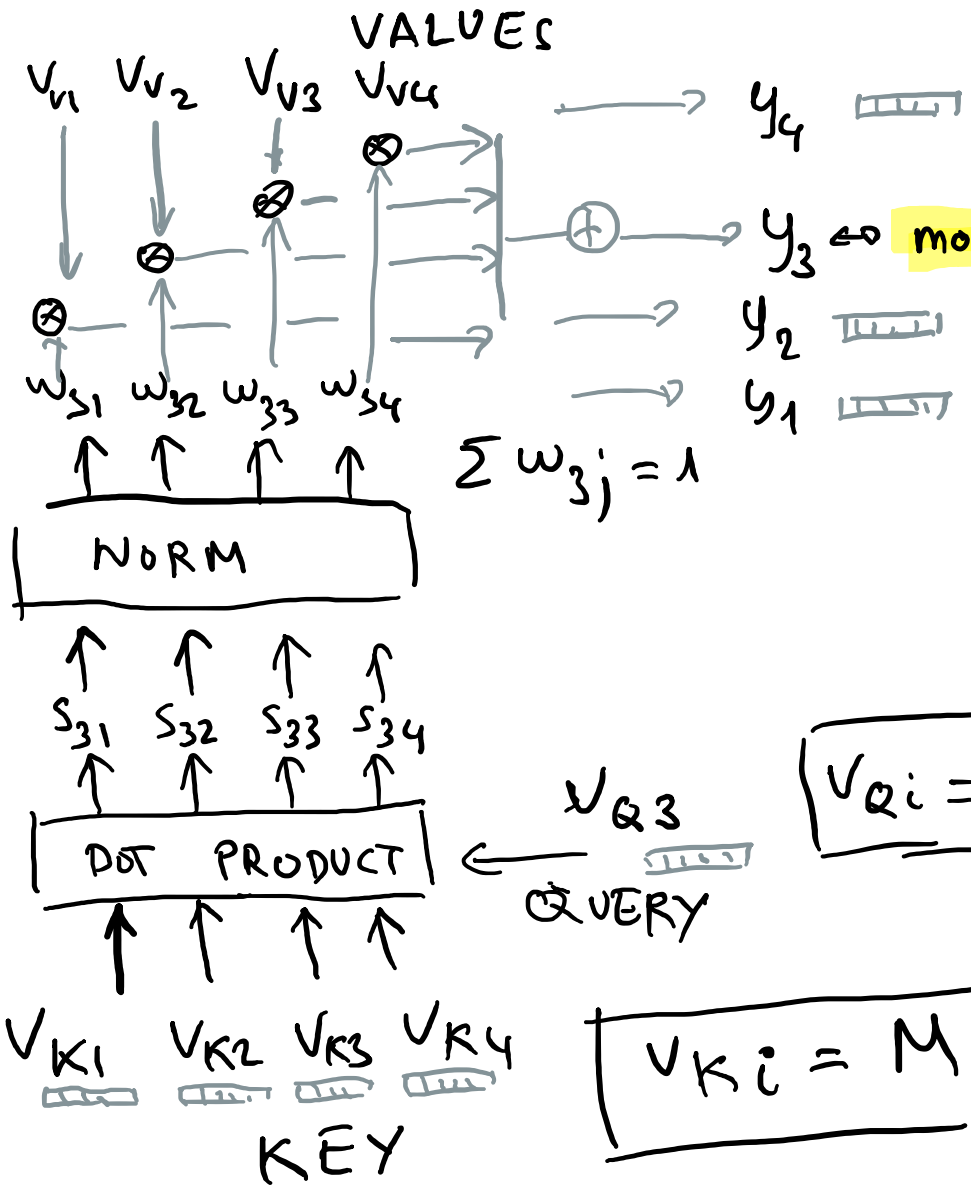
⇒ can change dimension

1xM MxN 1xN

FOR QUERIES → $v_i M_Q = v_{Qi}$

FOR VALUES → $v_i M_V = v_{Vi}$

Self-attention in transformers



$$v_{vi} = M_v v_i$$ VALUES

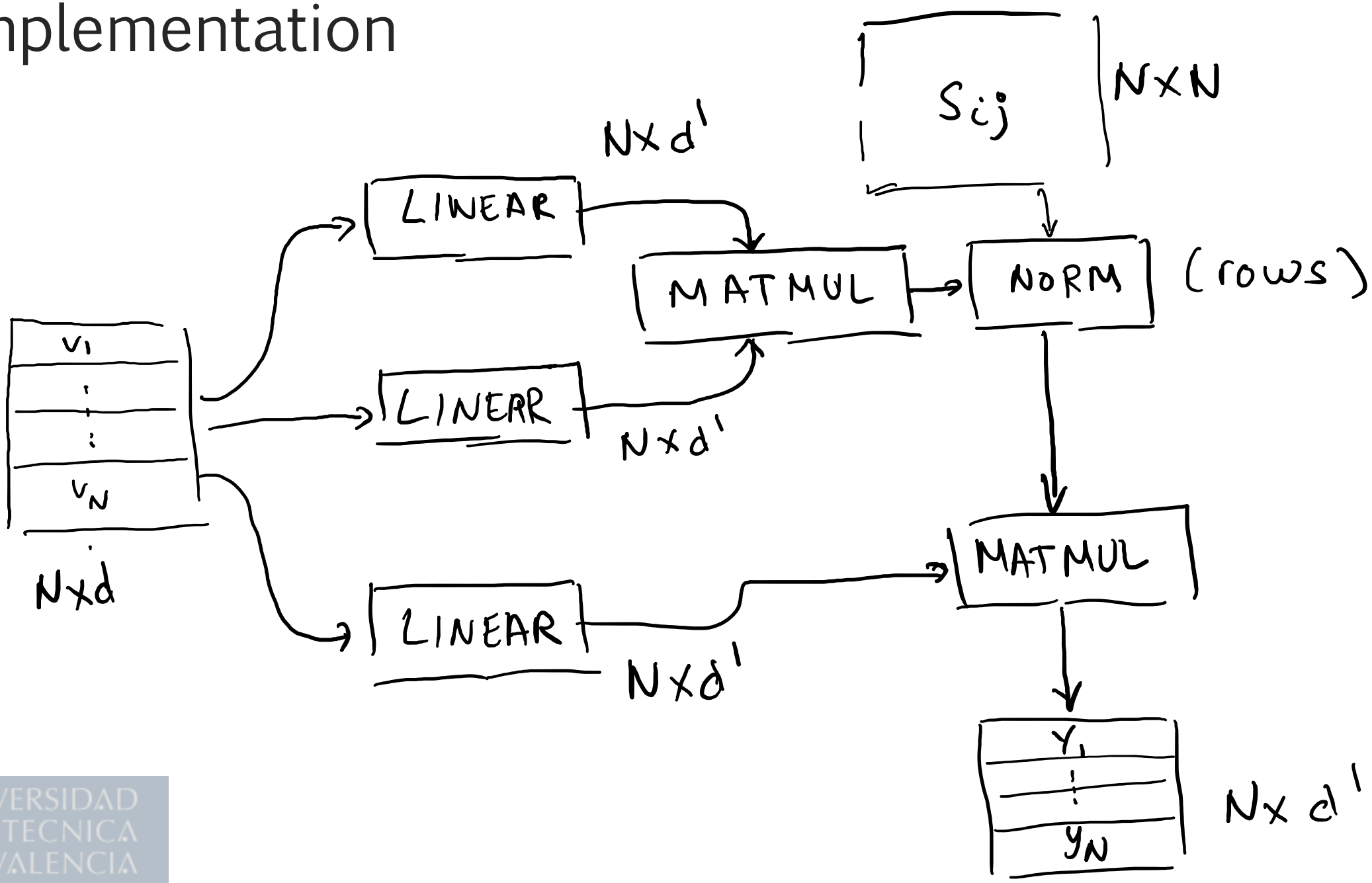
PARAMETERS

M_Q, M_K, M_V

$$v_{Qi} = M_Q v_i$$ QUERIES

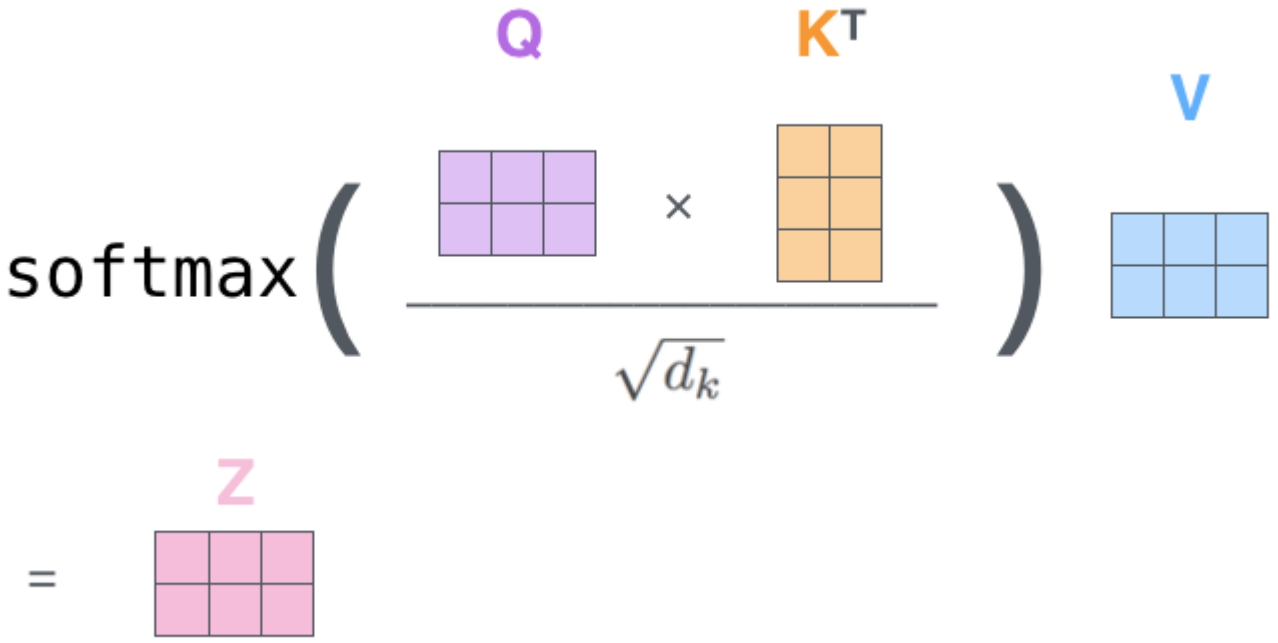
$$v_{Ki} = M_K \cdot v_i$$ KEYS

Implementation

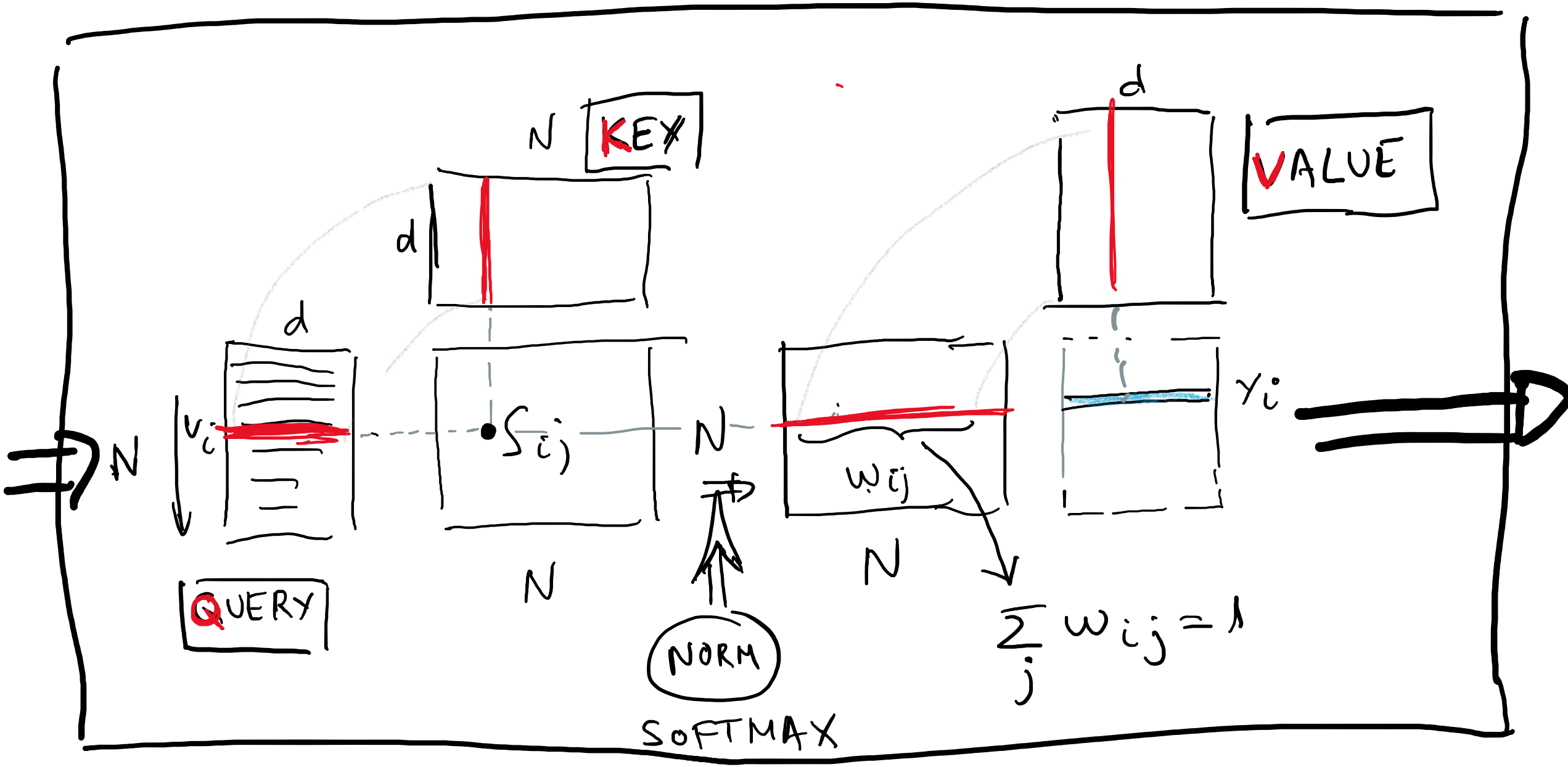


Self-Attention formula

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

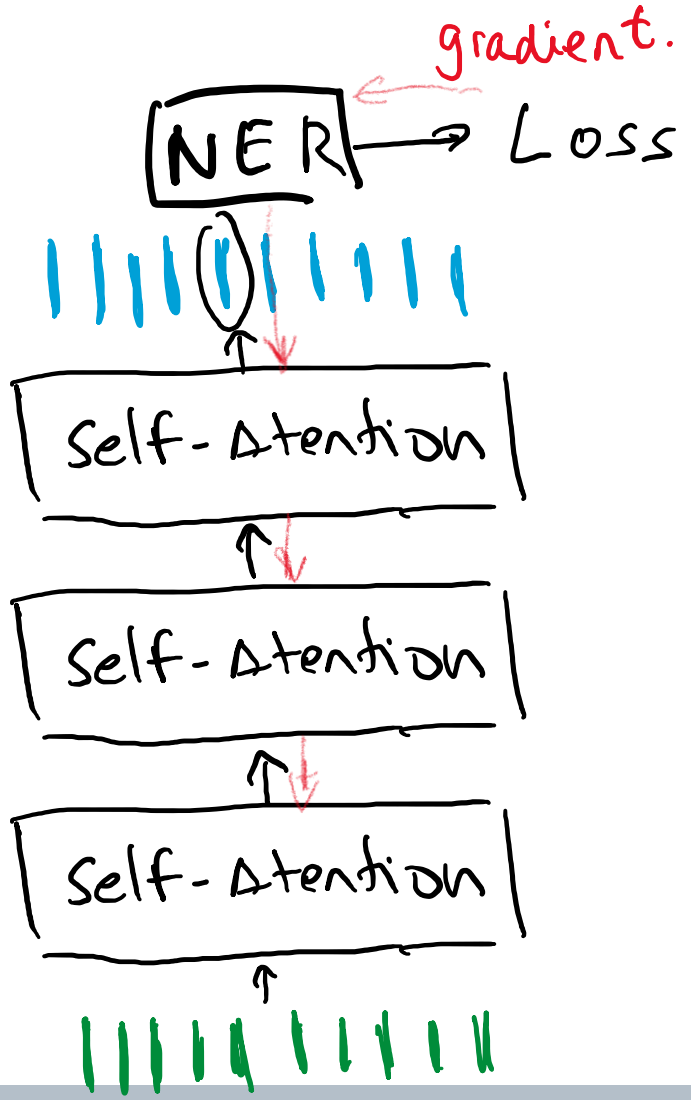


Self-Attention



Self-attention in neural networks

- Can be implemented easily:
  - **differenciable**
- Can be **stacked** in multiple layers:
  - very layer captures more and more context



Multi-head attention

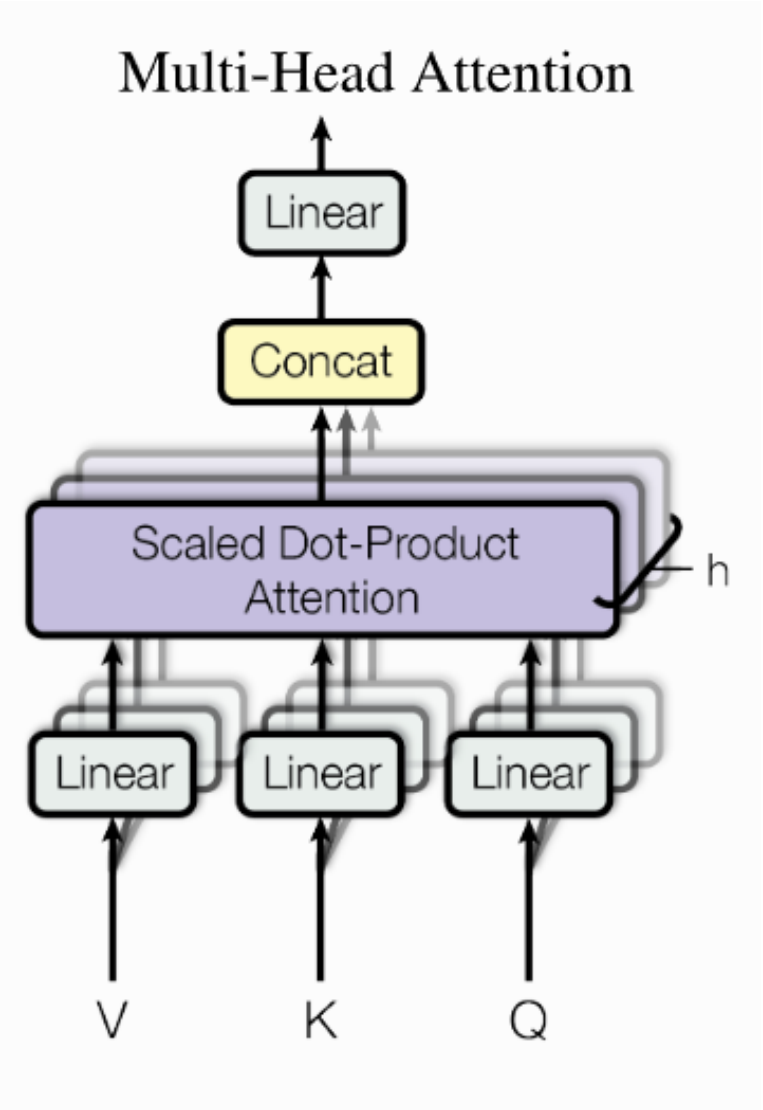
Language can have many little nuances

Multihead attention

- Introduce several self-attention mechanisms
- Multiple simpler self-attention can be more useful than a unique complex self-attention mechanism

Multihead attention

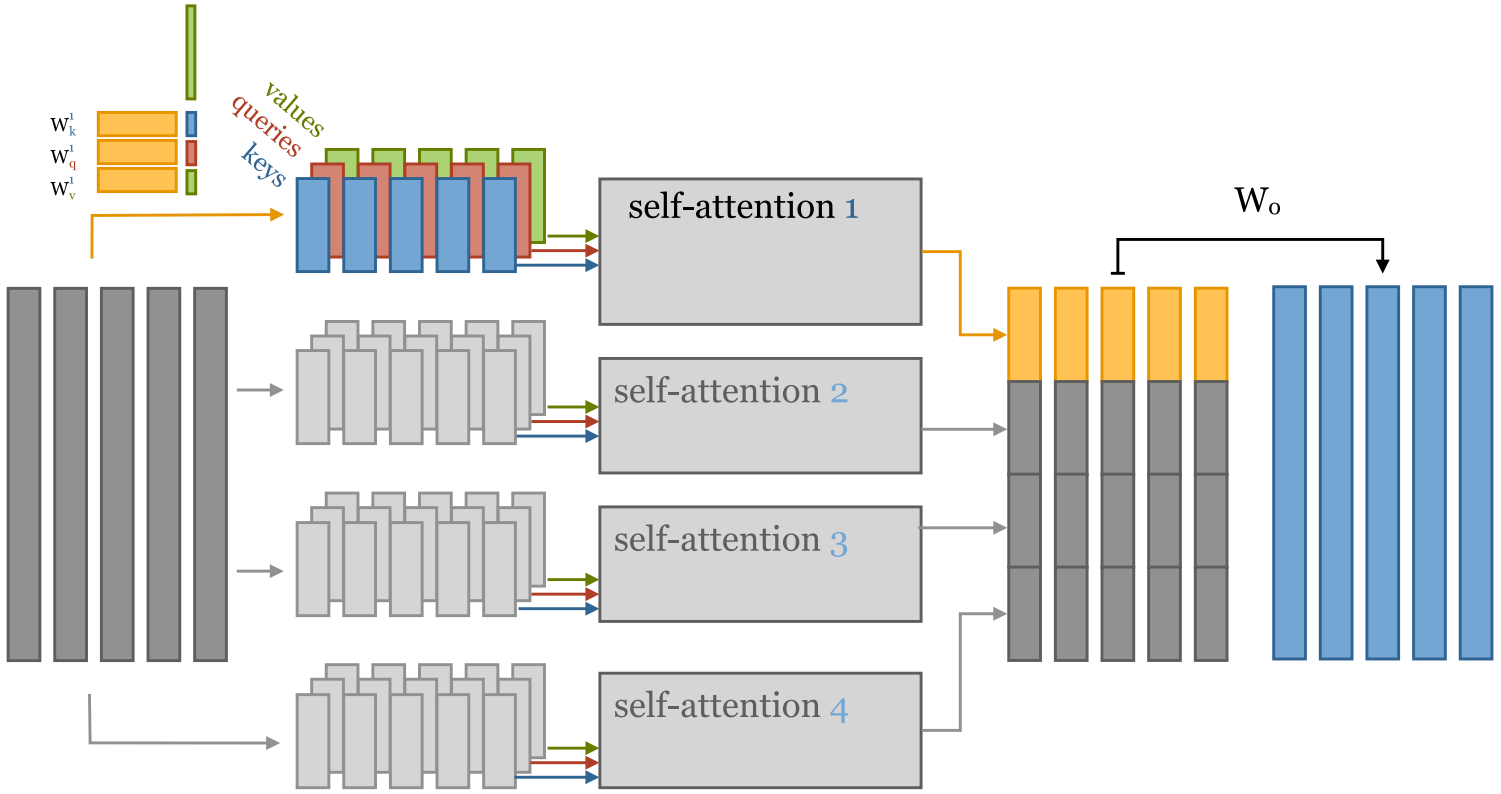
- Repeats attention h times with different weights
- Each block is called head



Multihead attention

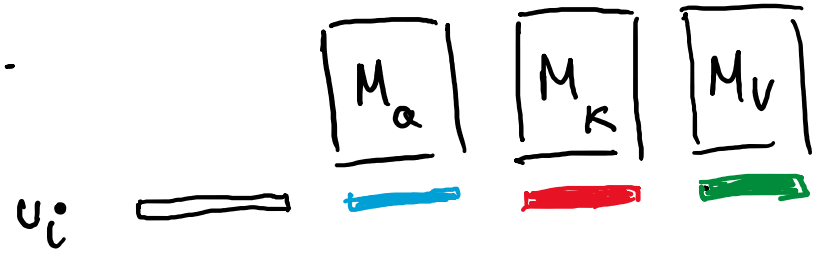
- Remember: The dimensionality of a word out of an attention layer can change

Implementation tricks

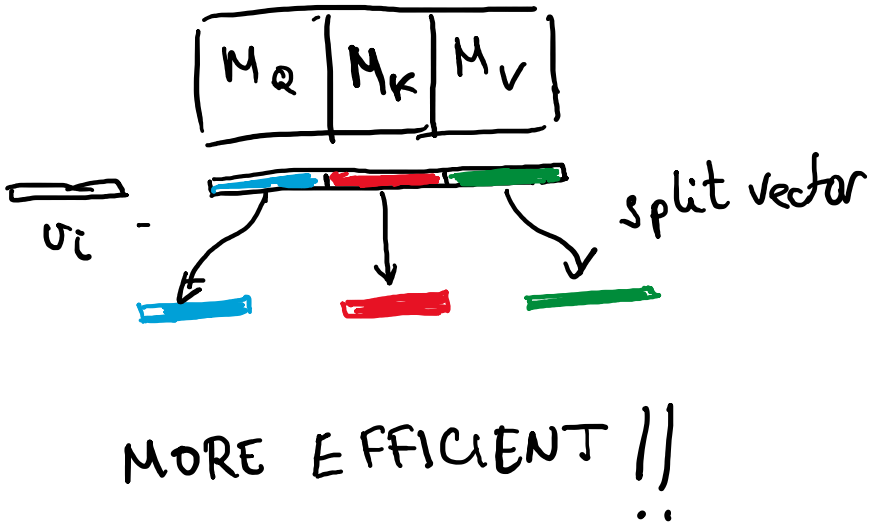


Implementation tricks

To extract queries-keys-values each vector is multiplied with a different matrix

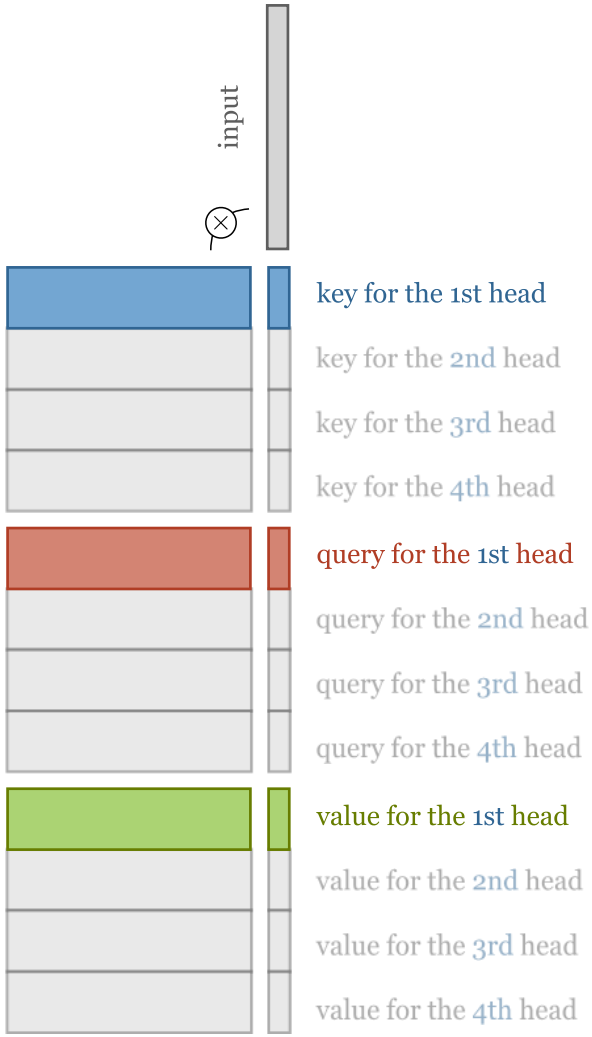


We can do the same with only one matrix multiplication



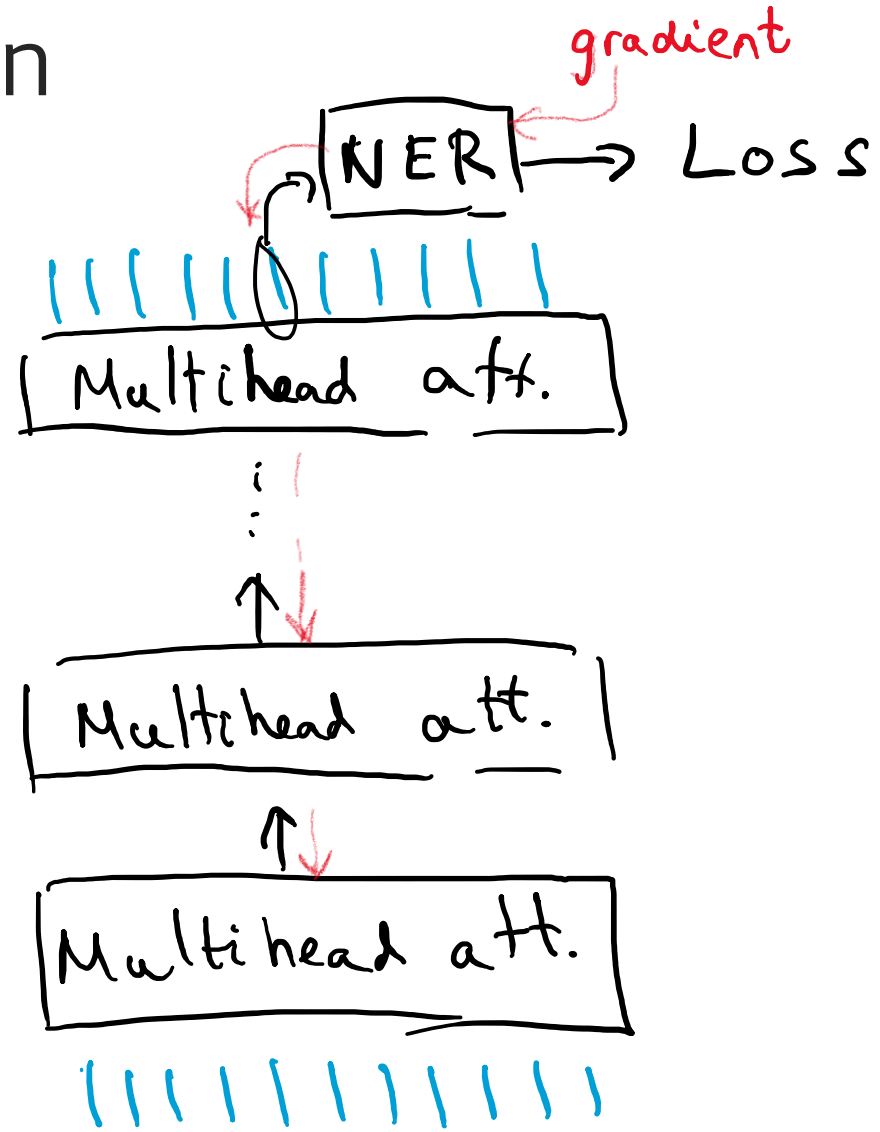
Implementation tricks

- We can do the same for multihead attention
- Only one matrix multiplication
- Usually d' = d/nheads

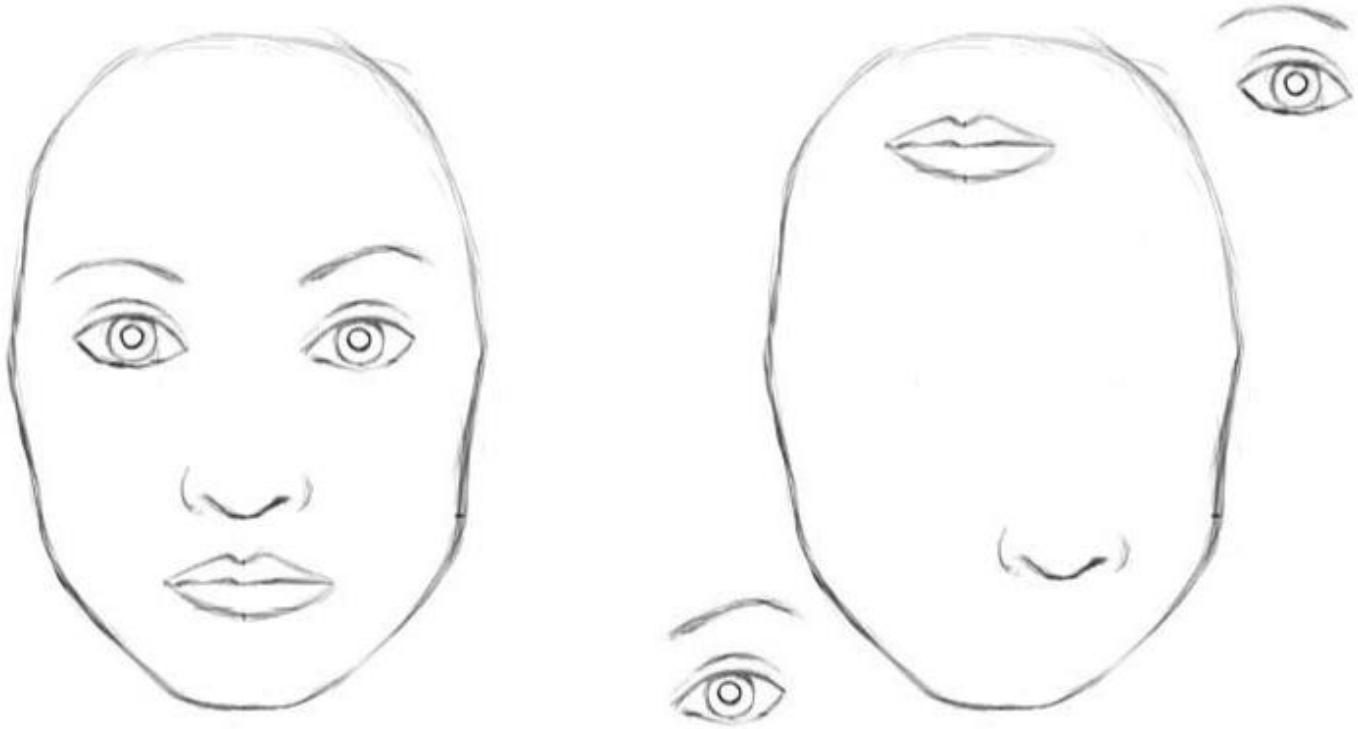


Stacked multi-head attention

- Multi-head attention can also be stacked
- And it is differenciable!!

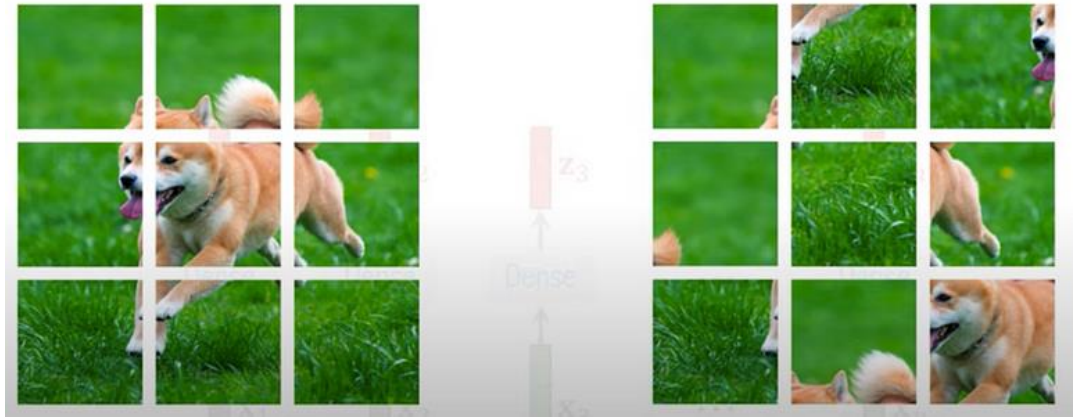


Importance of positional encoding



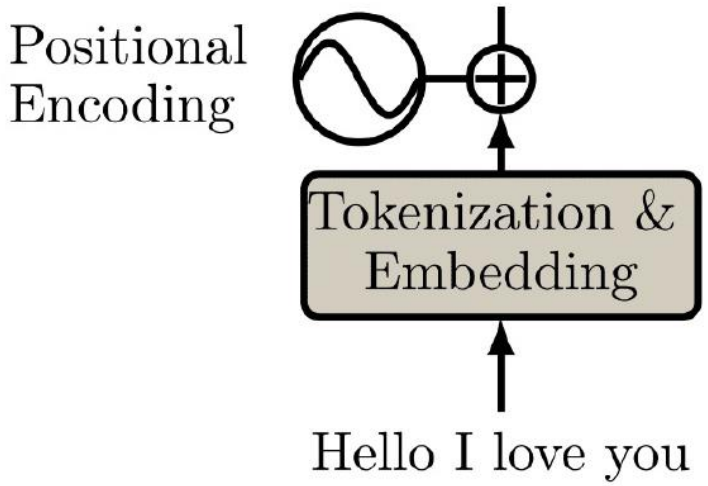
Positional encoding

Text	Tokenization
"Hello I love you" ⇨	"Hello", "I", "love", "you"
	"love", "Hello", "I", "you"
	"I", "love", "Hello", "you"



- If order is irrelevant then **we have a set** instead of a sequence
- But order is important-> we need positional encoding

Positional encoding



- **Positional encoding** is a set of small constants, which are added to the word embedding vector before the first self-attention layer.
- If the same word appears in a different position, the actual representation will be slightly different, **depending on where it appears in the input sentence.**

Good properties of positional encodings

1. Encoding of position should be unique
2. Distance between two consecutive time steps should be constant
3. Encoding scheme should generalize well to longer sentences
4. Encoding vectors should be deterministic

Types of positional encodings

- Fixed sinusoidal positional encoding:

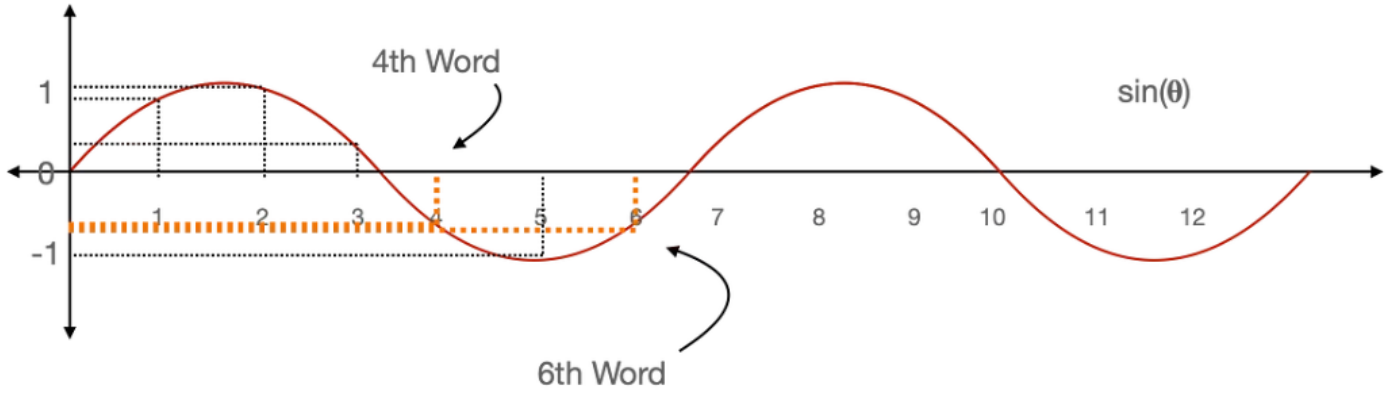
Mathematically:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/512}}\right)$$

$$PE_{(pos,2*i+1)} = \cos\left(\frac{pos}{10000^{2i/512}}\right)$$

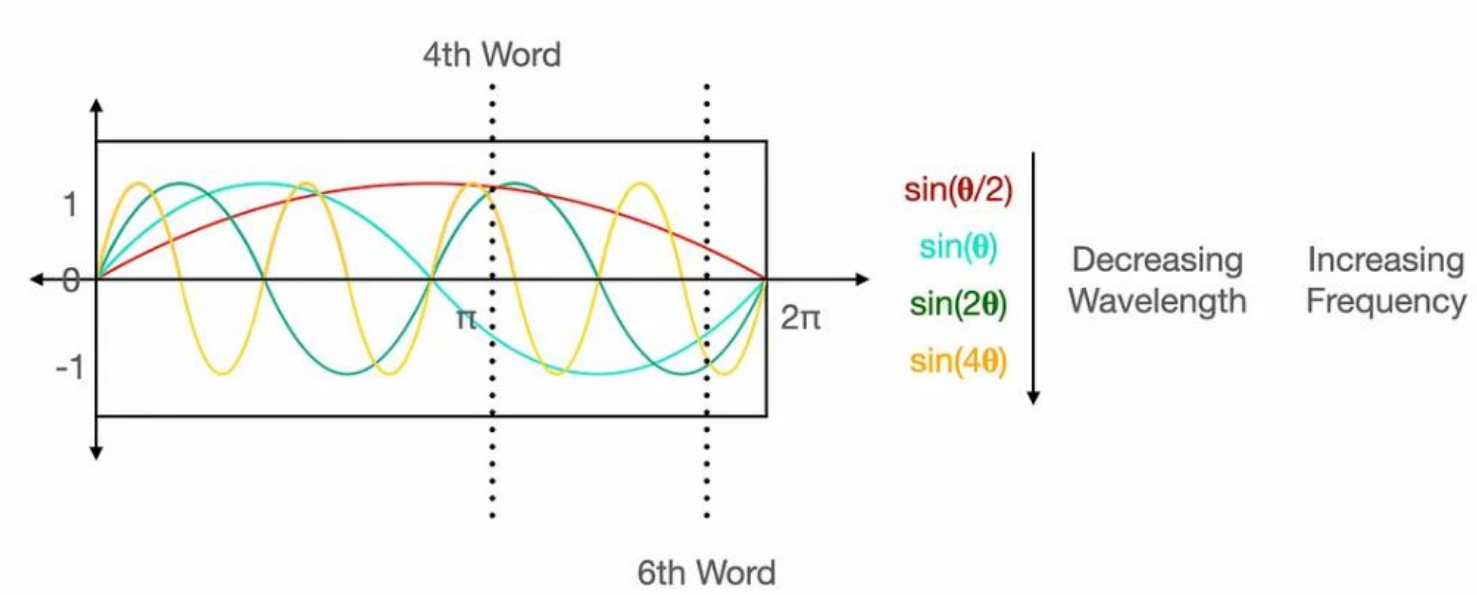
- Learned: Positional encodings are parameters which are learned during training

Positional encodings

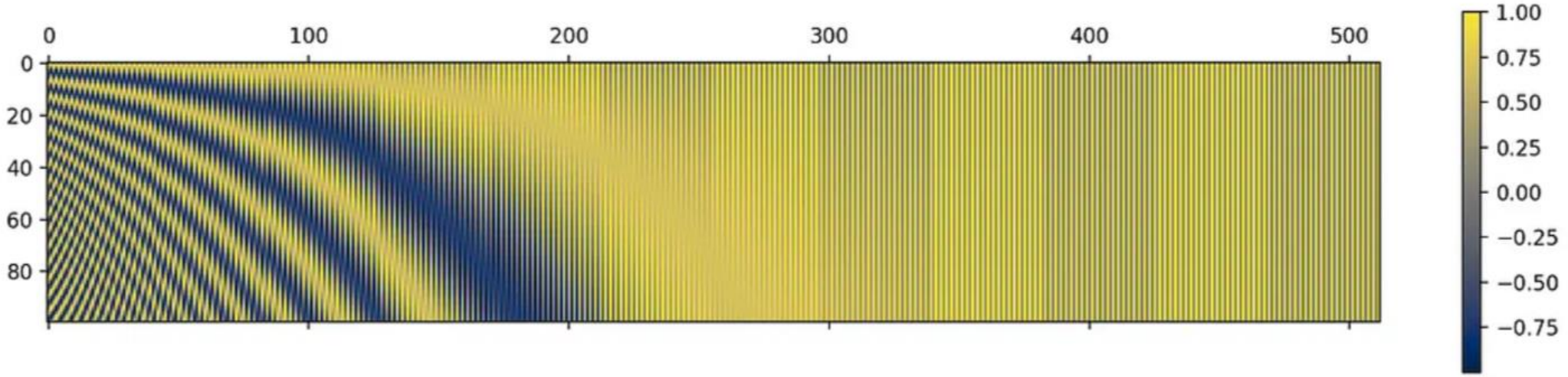


Cosine is used to distinguish between 4th and 6th words

But we need more frequencies to distinguish Between 4th and 11th words



Positional encodings



Relative positional encoding

We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

$$M.\begin{bmatrix}\sin(\omega_k.t)\\ \cos(\omega_k.t)\end{bmatrix}=\begin{bmatrix}\sin(\omega_k.(t+\phi))\\ \cos(\omega_k.(t+\phi))\end{bmatrix}$$

The advantage of using sin and cos functions is that a linear transformation can be used to “translate” the representation

Positional encoding

- Another property of sinusoidal position encoding is that the distance between neighboring time-steps are symmetrical and decays nicely with time.

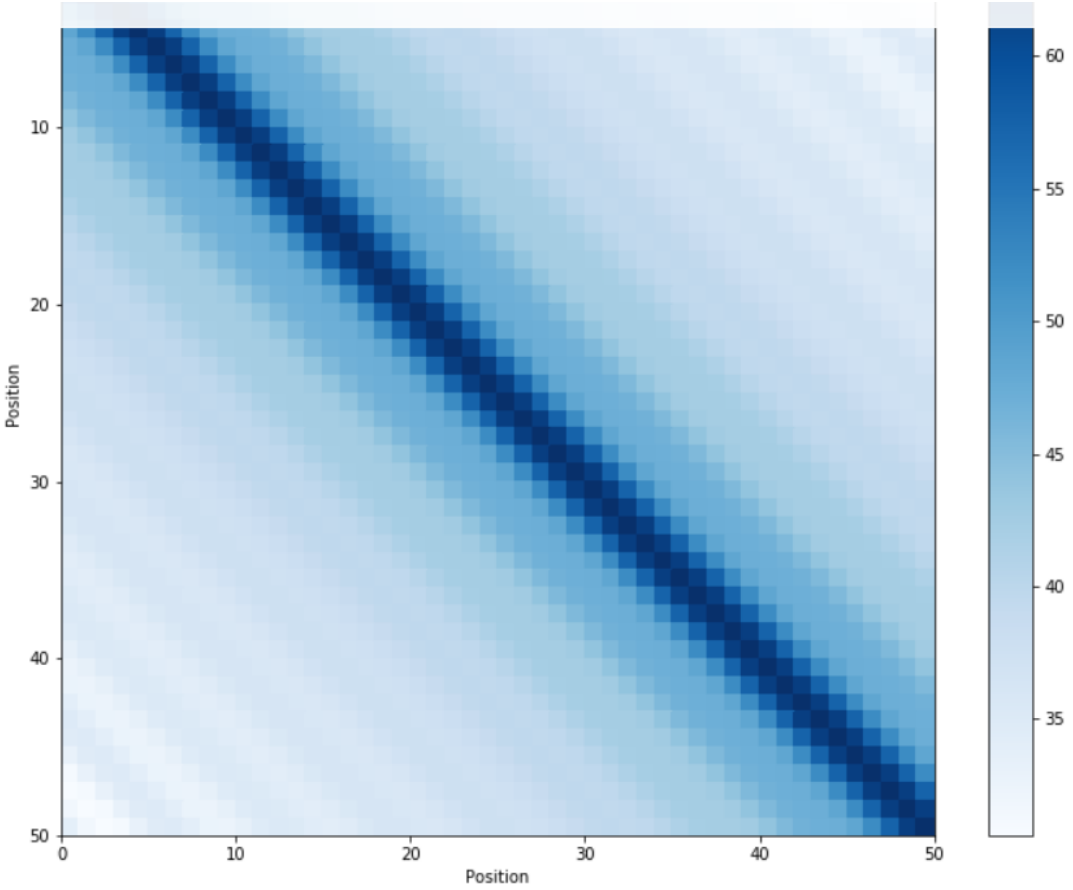


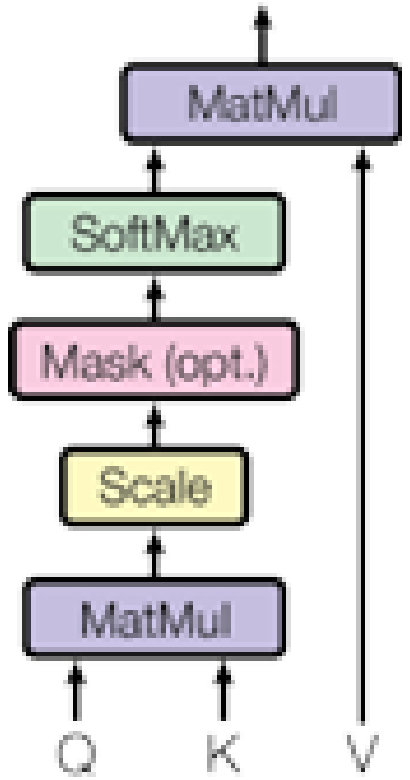
Figure 3 - Dot product of position embeddings for all time-steps

Comments on positional encodings

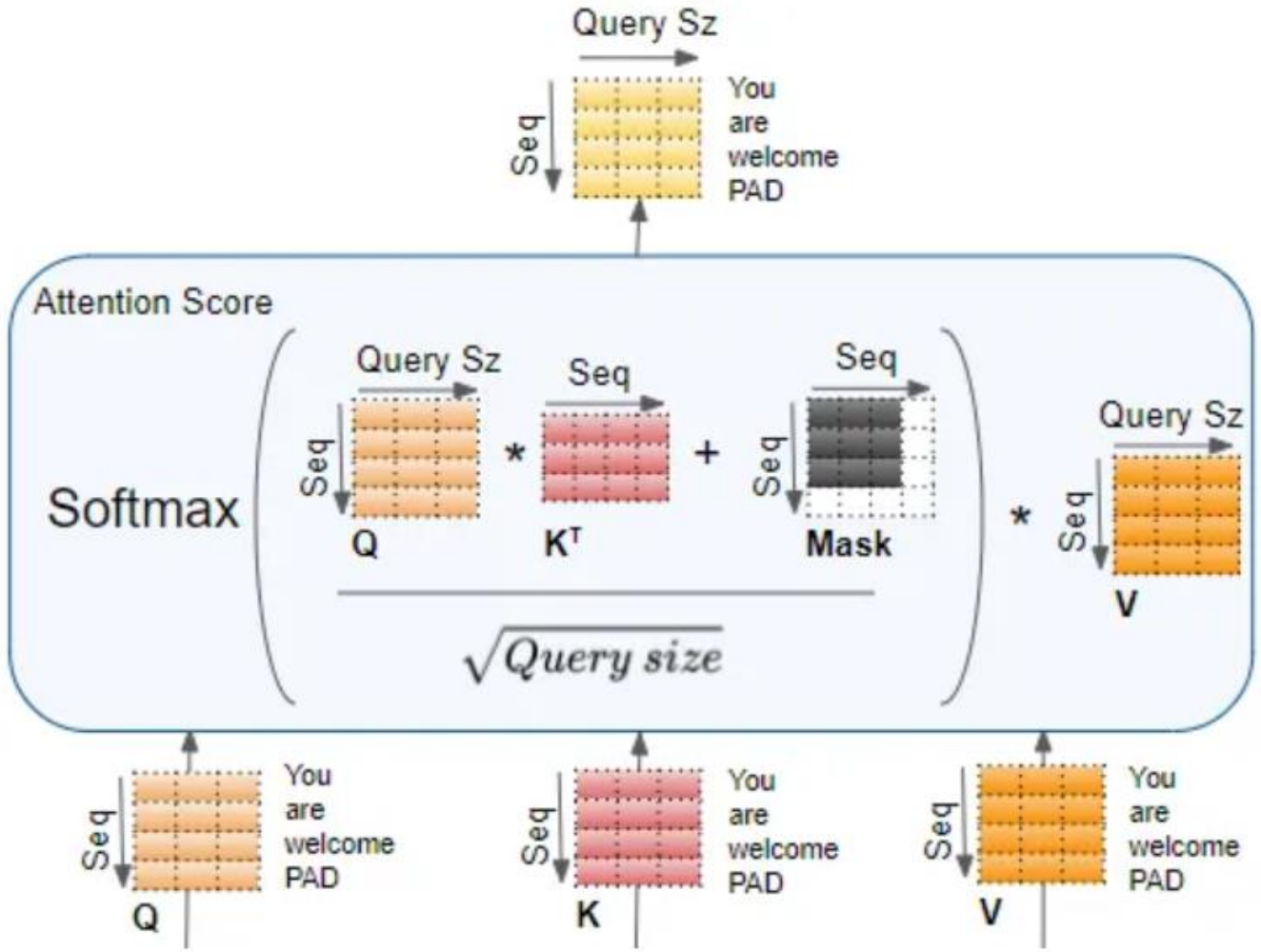
- Positional encodings are summed instead of concatenated to reduce the dimensionality
- In high dimensional spaces they can be in orthogonal subspaces of words representations

Masked self attention

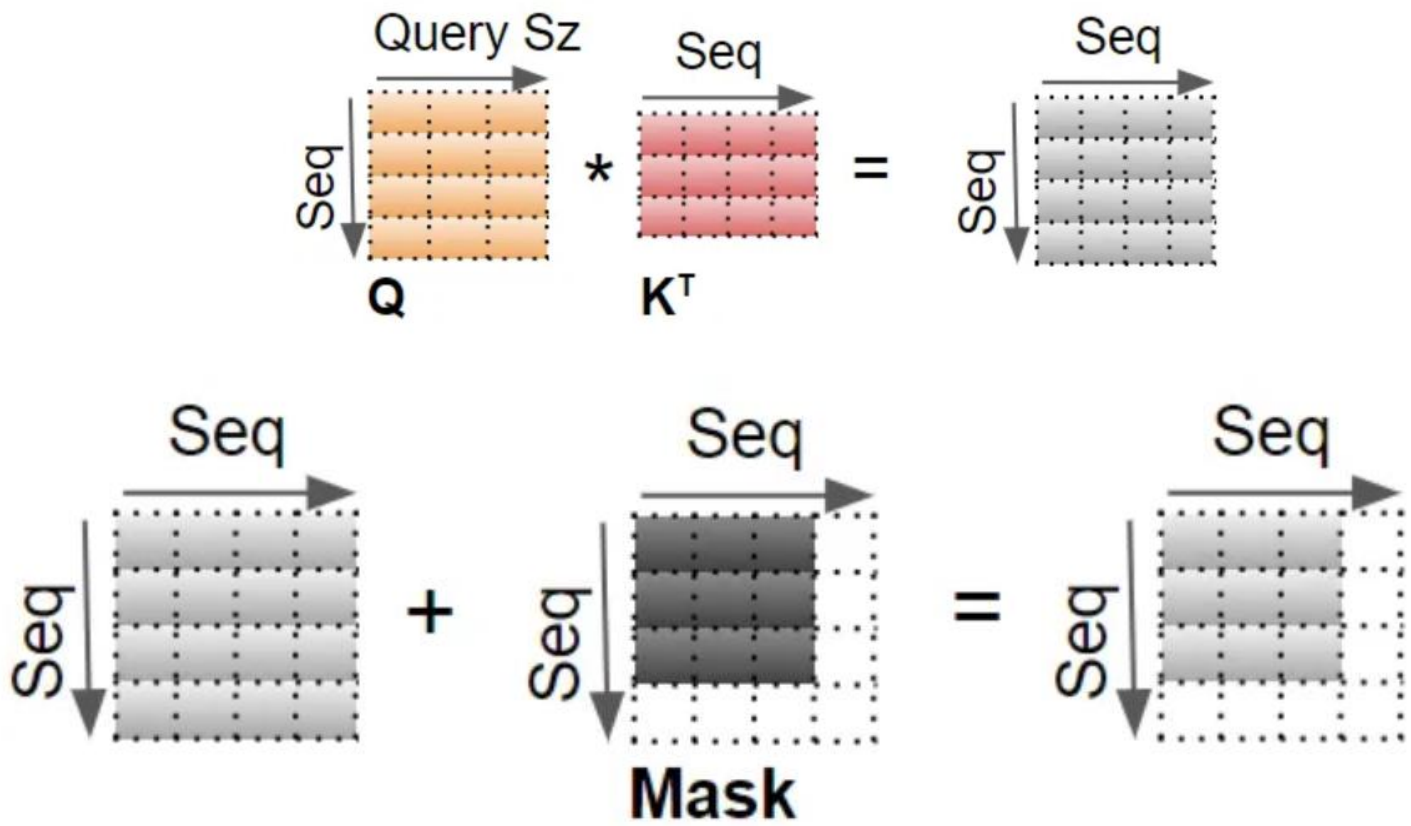
- Mask can be useful to restrict attention to less tokens
- It can be done by setting attention weights to zero
- In the log domain (before softmax) it is done by subtracking large value (typ -100)



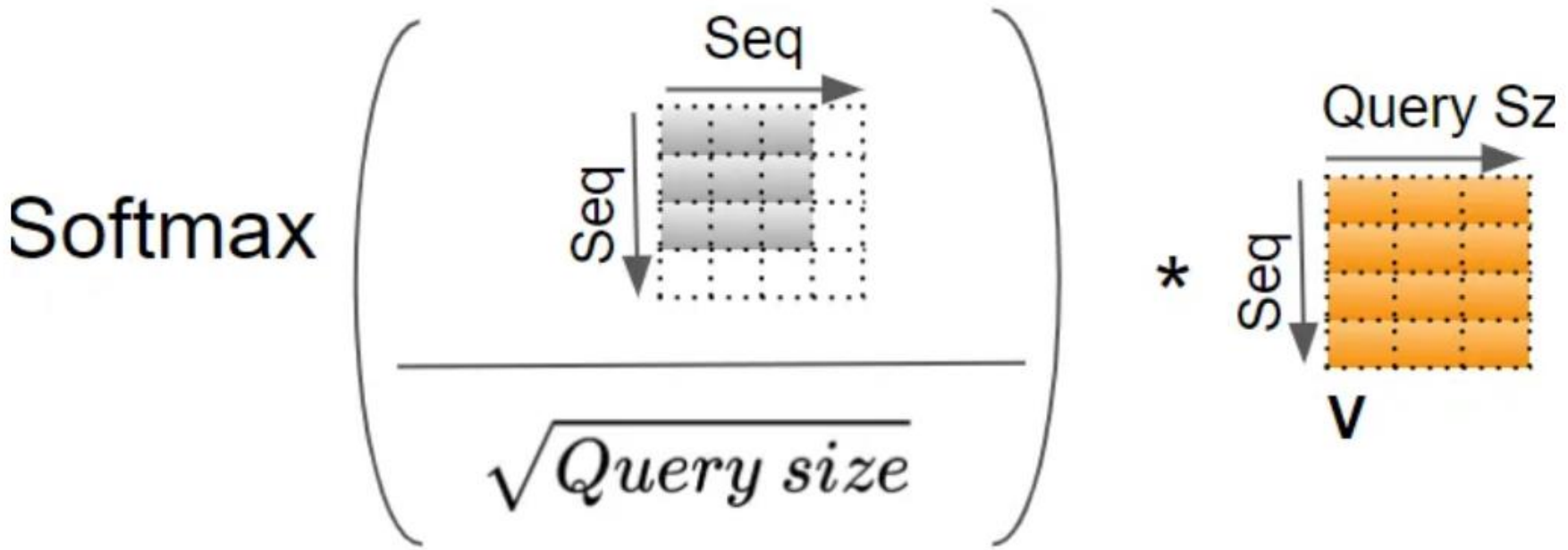
Masked self attention



Masked self-attention



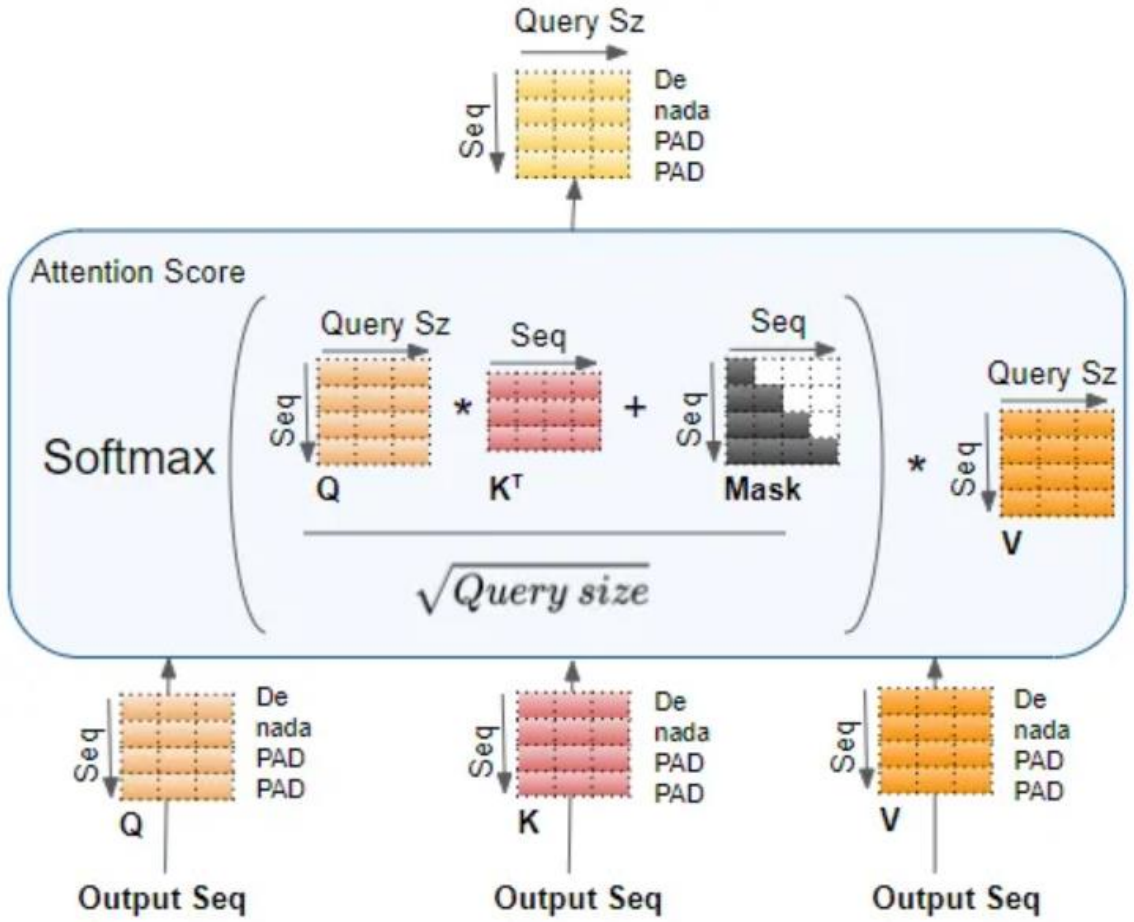
Masked self attention



Masked self-attention

- Helps to deal with sequences of different lengths
- Restrics attention to some parts (ej. Causal attention GPT models)
- Can restrict attention to closer tokens (visión Transformers)

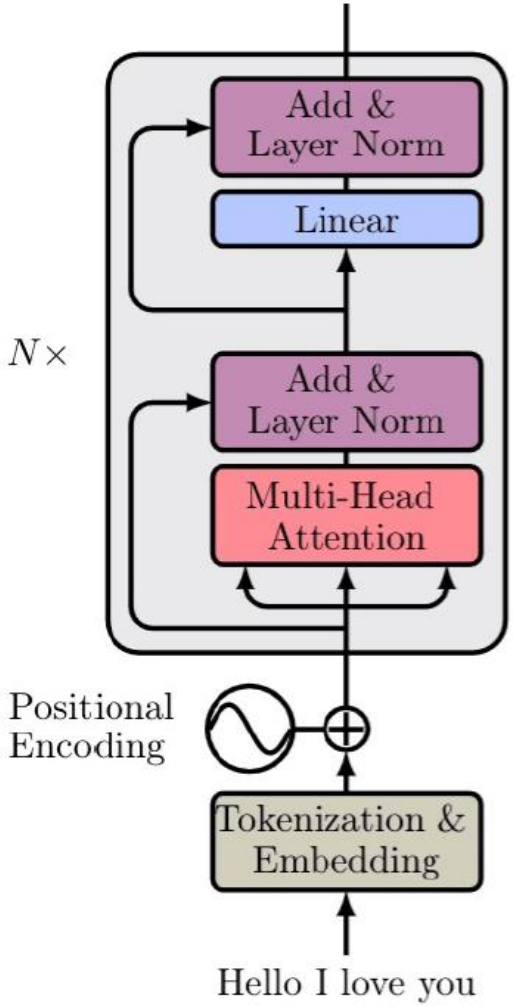
Masked self-attention (decoder model)



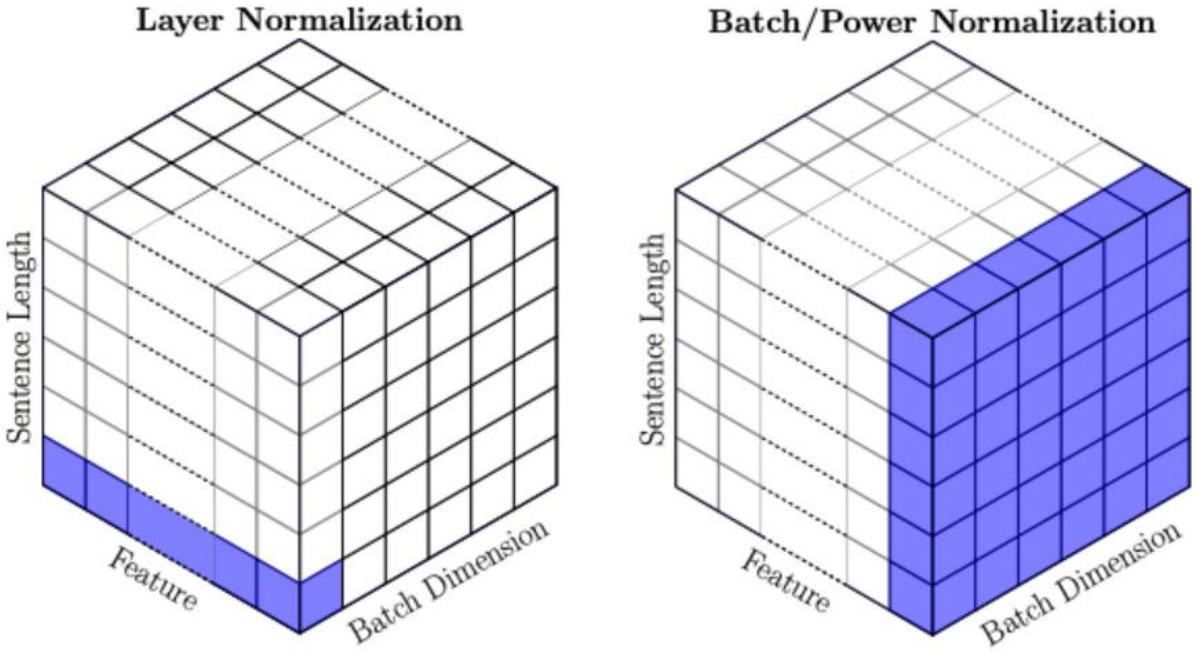
Masked self-attention

```
def scaled_dot_product(q, k, v, mask=None):
    d_k = q.size()[-1]
    attn_logits = torch.matmul(q, k.transpose(-2, -1))
    attn_logits = attn_logits / math.sqrt(d_k)
    if mask is not None:
        attn_logits = attn_logits.masked_fill(mask == 0, -9e15)
    attention = F.softmax(attn_logits, dim=-1)
    values = torch.matmul(attention, v)
    return values, attention
```

Transformer layer



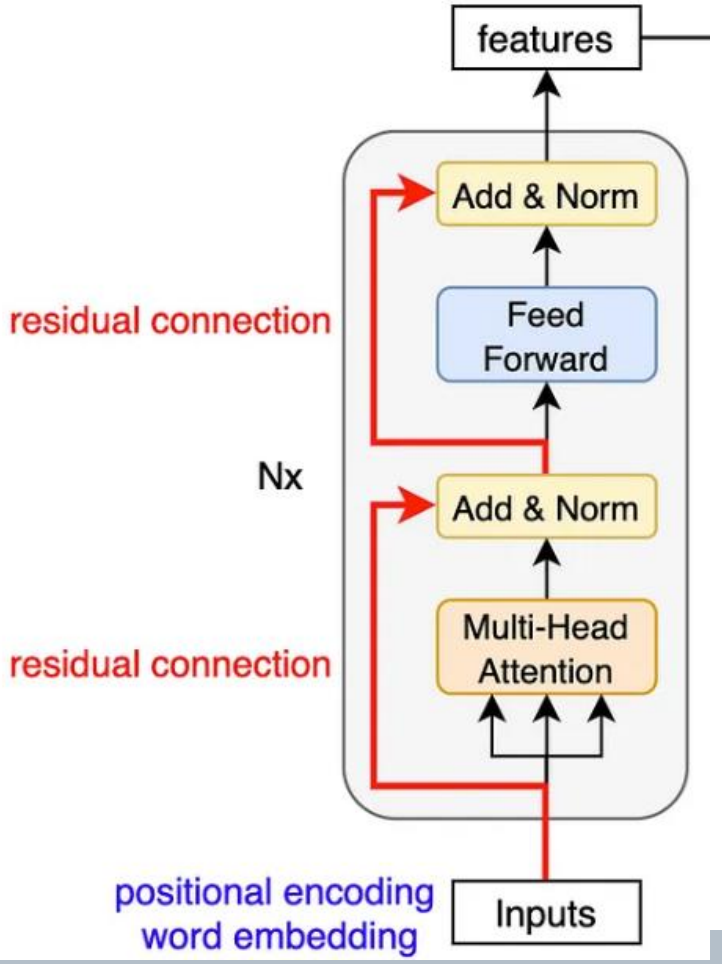
Layer normalization



- Has two trainable parameters!!

Residual connections and positional encodings

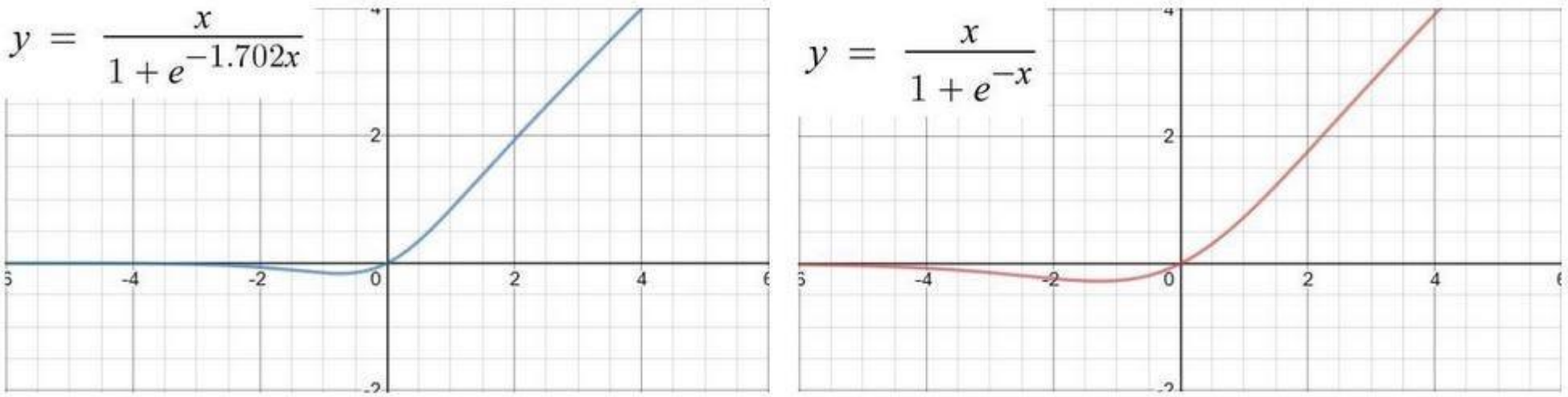
- Residual connections help to propagate positional information through layers



Batchnorm vs. layernorm

- Sequences can have different lengths, each batch can be normalized differently (bad)
- It is also found that "statistics of NLP data across the batch dimension exhibit large fluctuations throughout training. This results in instability, if BN is naively implemented”
- Tradition: It was the common method to normalize in RNN networks

Activation functions



- Transformer replace ReLU by GELU or Swish
- https://www.youtube.com/watch?v=uiB97cPEVxM