BIO-INSPIRED COMPUTING: APPLICATIONS AND INTERFACES

SEQUENCE LEARNING PART 2

Slides created by Jo Plested

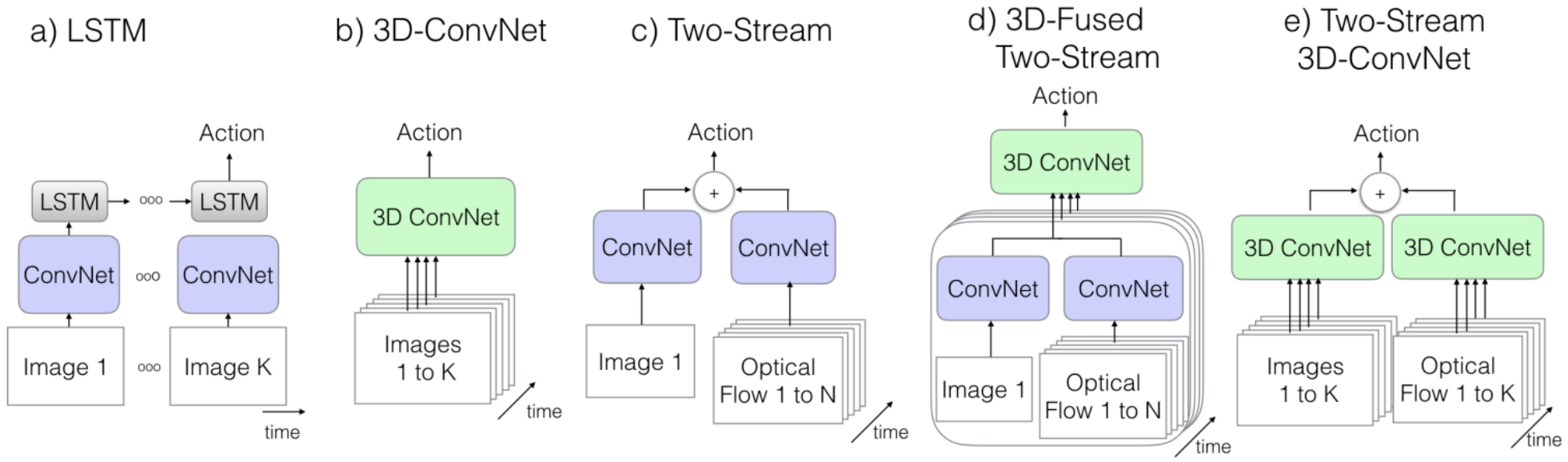
LECTURE OUTLINE

- CNNs for sequence learning
  - how do they work
  - current research
- The transformer model
  - how do they work

FEED FORWARD NNS FOR SEQUENCES

- In domains where tricks can be used to express inputs of variable length with inputs of fixed length feedforward neural networks can be used to model the time domain
- Most commonly this has been done with CNNs, but recently the most successful language prediction models have used multiple blocks of standard fully connected feed forward layers combined with attention layers

CNNS FOR VIDEO MODELLING



https://arxiv.org/abs/1705.07750

- A video can be split into pieces of a predetermined number of frames so a 3D CNN or concatenated 2D CNNs can be applied
- There are a number of different ways the spatial and time dimensions can be fused

CNNS FOR VIDEO MODELLING

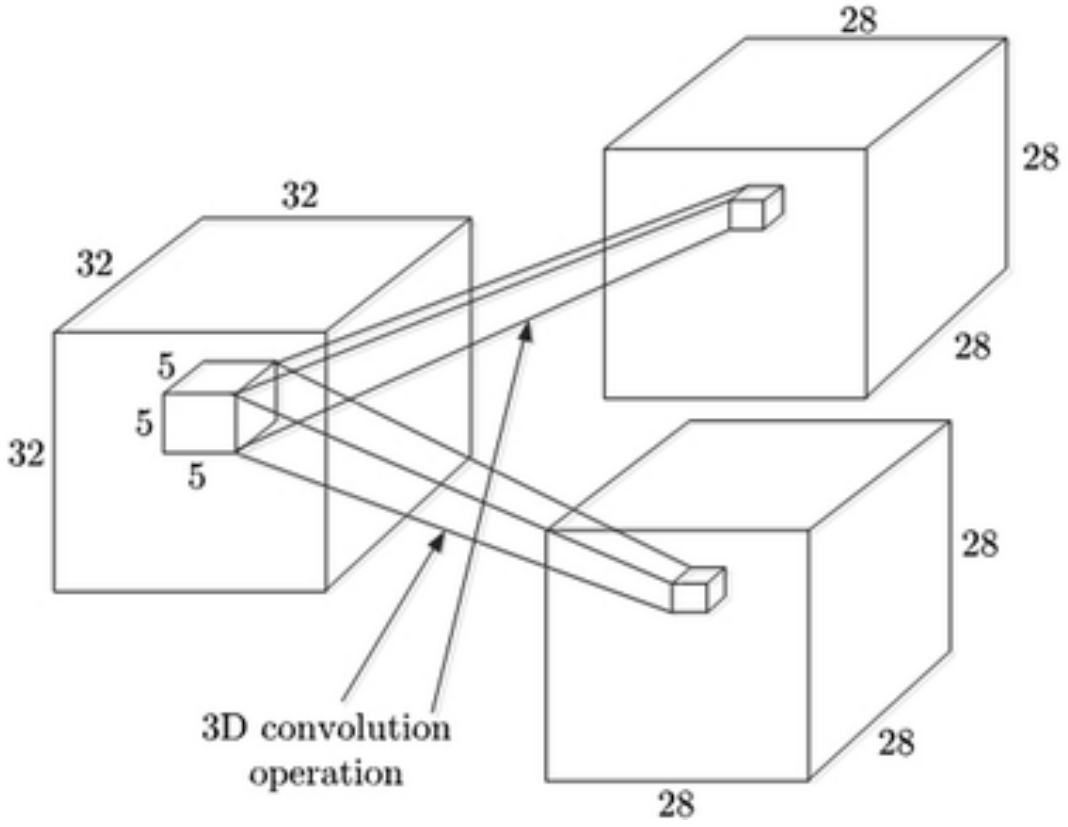
3D Resnet architecture specifications

Layer Name	Architecture	
	18-layer	34-layer
conv1	$7 \times 7 \times 7, 64$, stride 1 (T), 2 (XY)	
conv2_x	$3 \times 3 \times 3$ max pool, stride 2 $\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 64 \\ 3 \times 3 \times 3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 128 \\ 3 \times 3 \times 3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 256 \\ 3 \times 3 \times 3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3 \times 3, 512 \\ 3 \times 3 \times 3, 512 \end{bmatrix} \times 3$
	average pool, 400-d fc, softmax	

https://arxiv.org/pdf/1708.07632.pdf

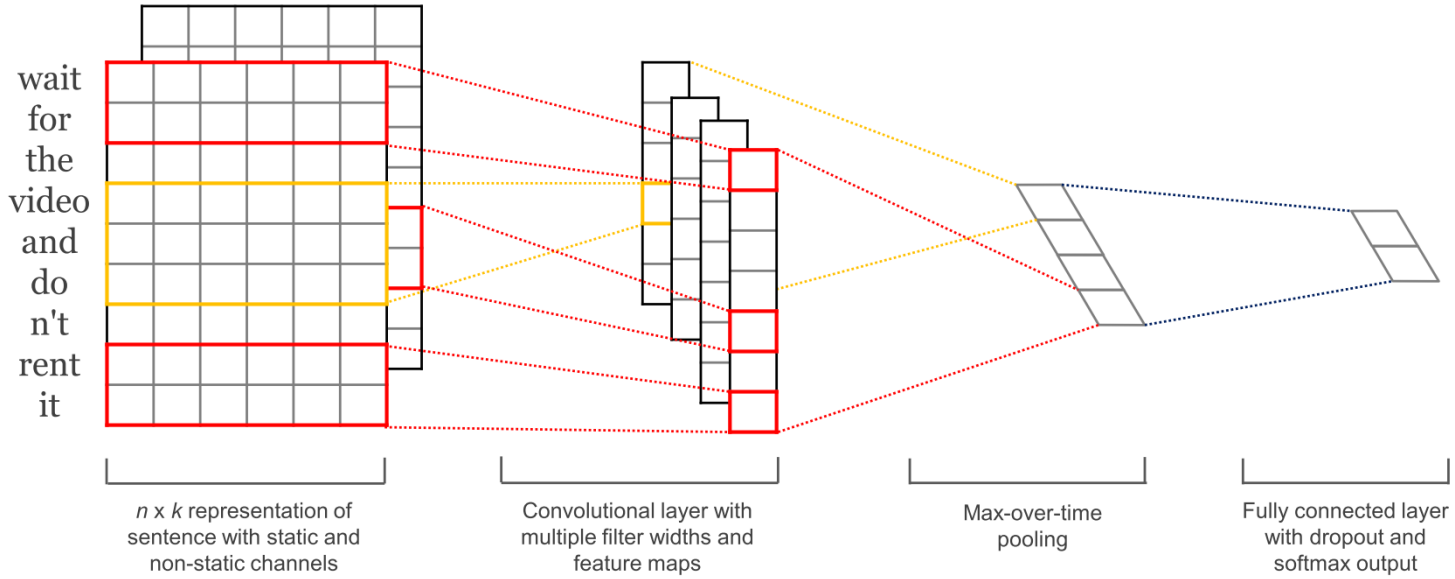
A 3D convolutional layer is just like a 2D convolutional layer, but it is moving across a 3rd dimension – usually time

CNNS FOR VIDEO MODELLING



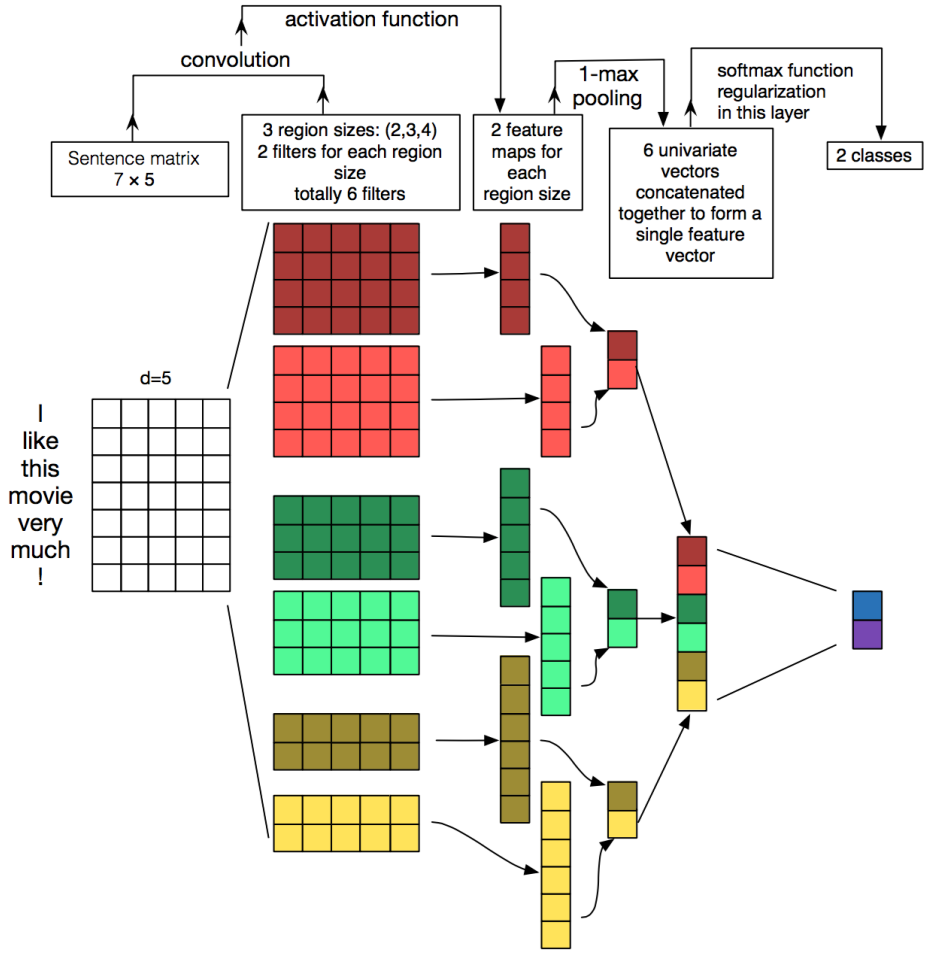
A 3D convolutional layer is just like a 2D convolutional layer, but the input, receptive field and feature map are 3D with the 3rd dimension usually being time, and the receptive field is moving along all 3 dimensions

CNNS FOR LANGUAGE MODELLING



- A variable length sentence with words represented as vectors is convolved with multiple filters with multiple receptive field size.
- This produces feature maps of a variable length in the time dimension which are each reduced to one output with pooling over time.

CNNS FOR LANGUAGE MODELLING



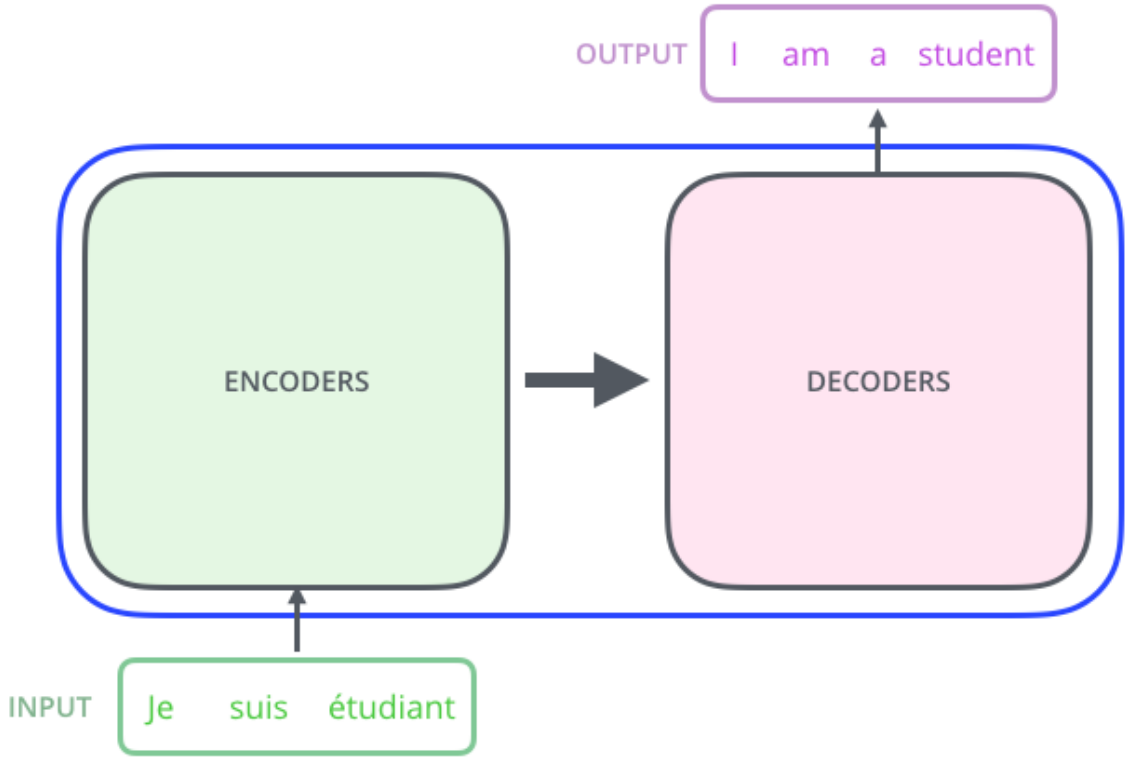
TRANSFORMERS FOR LANGUAGE MODELLING

- The transformer was originally proposed in the paper Attention is all you need
- A combination of attention and scoring is used to allow context in word embeddings created by fully connected feed forward neural network layers

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

TRANSFORMERS FOR LANGUAGE MODELLING

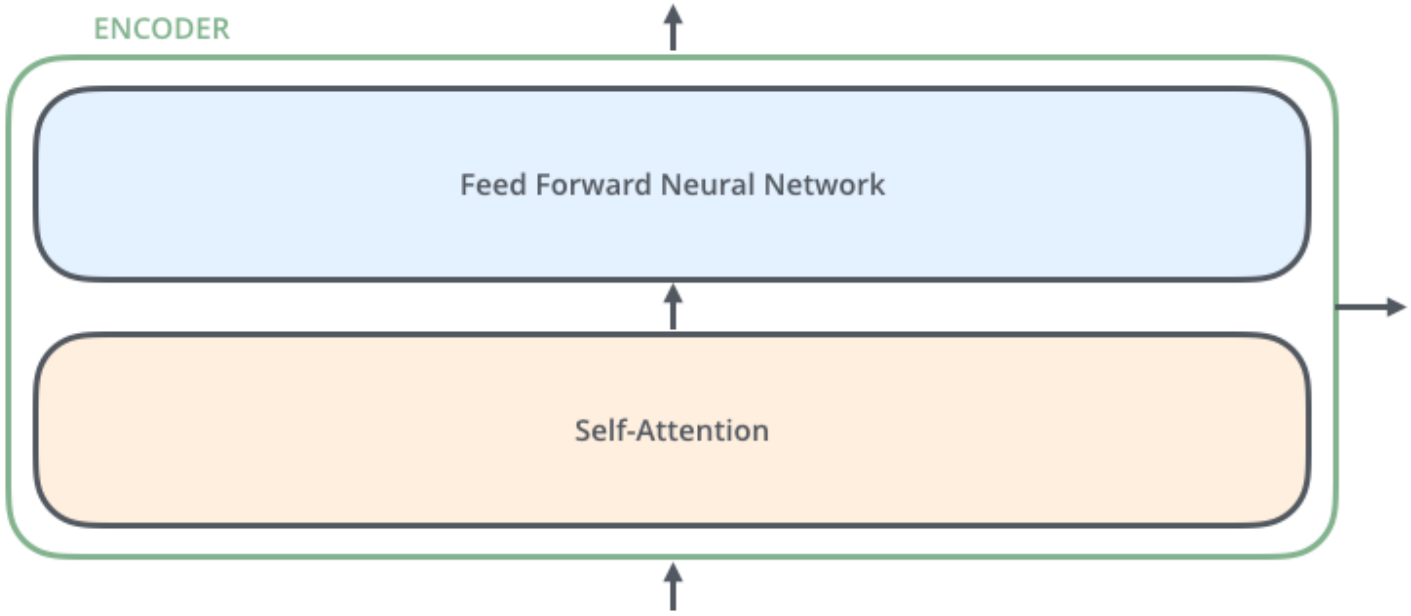
Just like an RNN the model consists of an encoder and a decoder



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

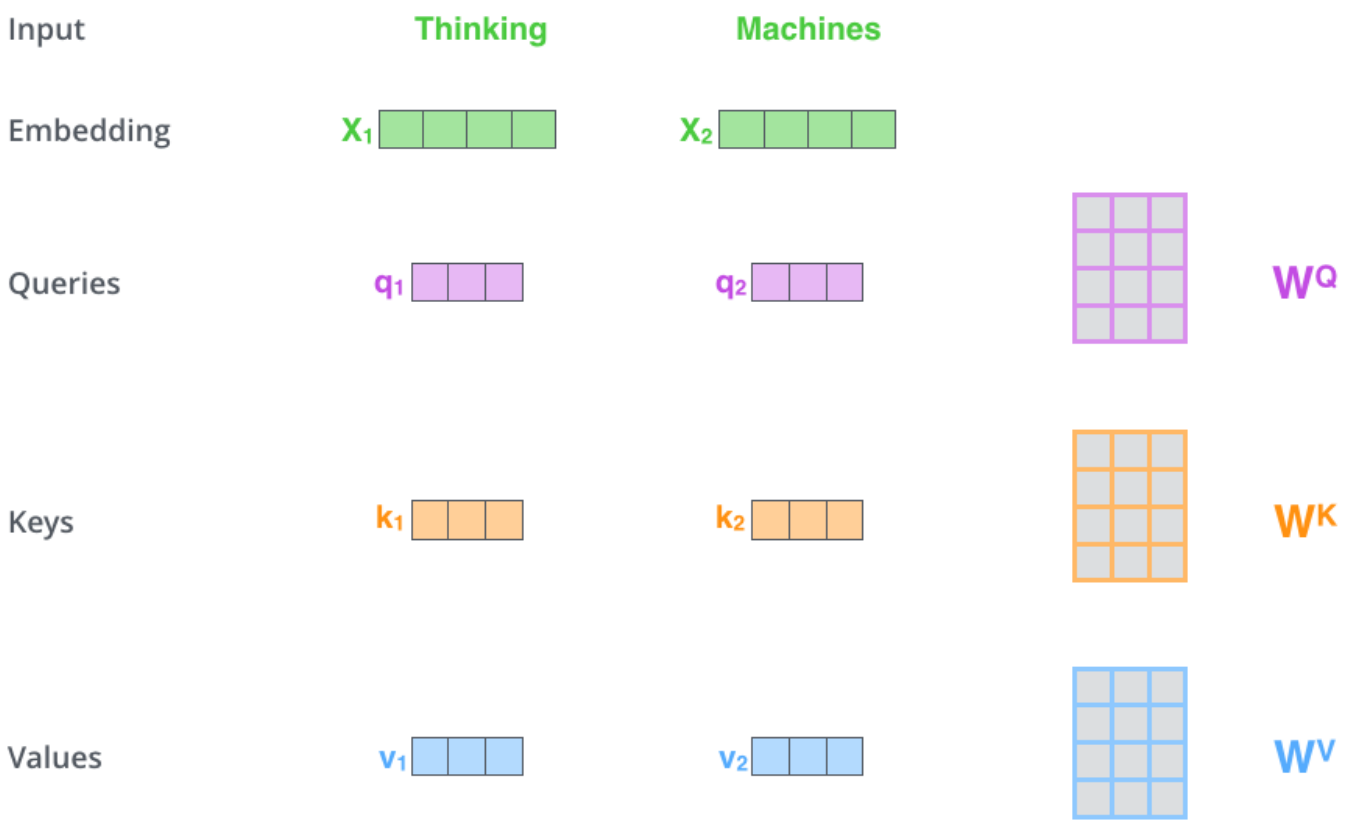
TRANSFORMERS FOR LANGUAGE MODELLING

Each block of the encoder consists of a self-attention layer and a feed forward NN layer



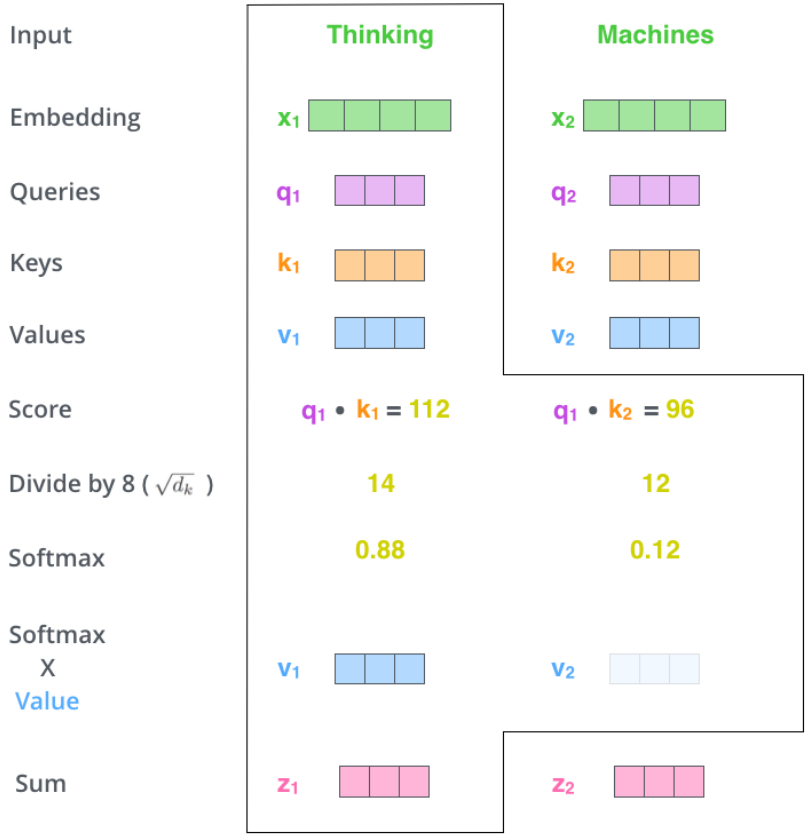
TRANSFORMERS FOR LANGUAGE MODELLING

Self attention – weight matrices and calculation of vectors



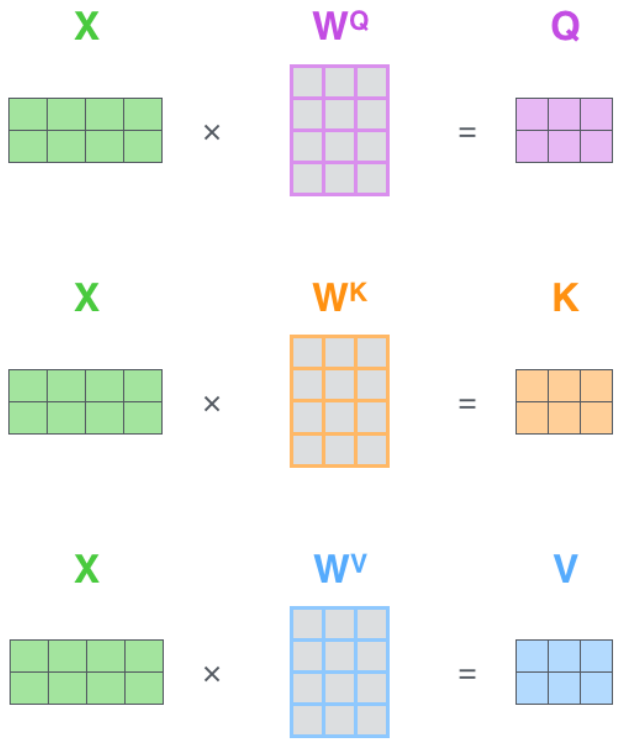
TRANSFORMERS FOR LANGUAGE MODELLING

Self attention – scoring calculation



TRANSFORMERS FOR LANGUAGE MODELLING

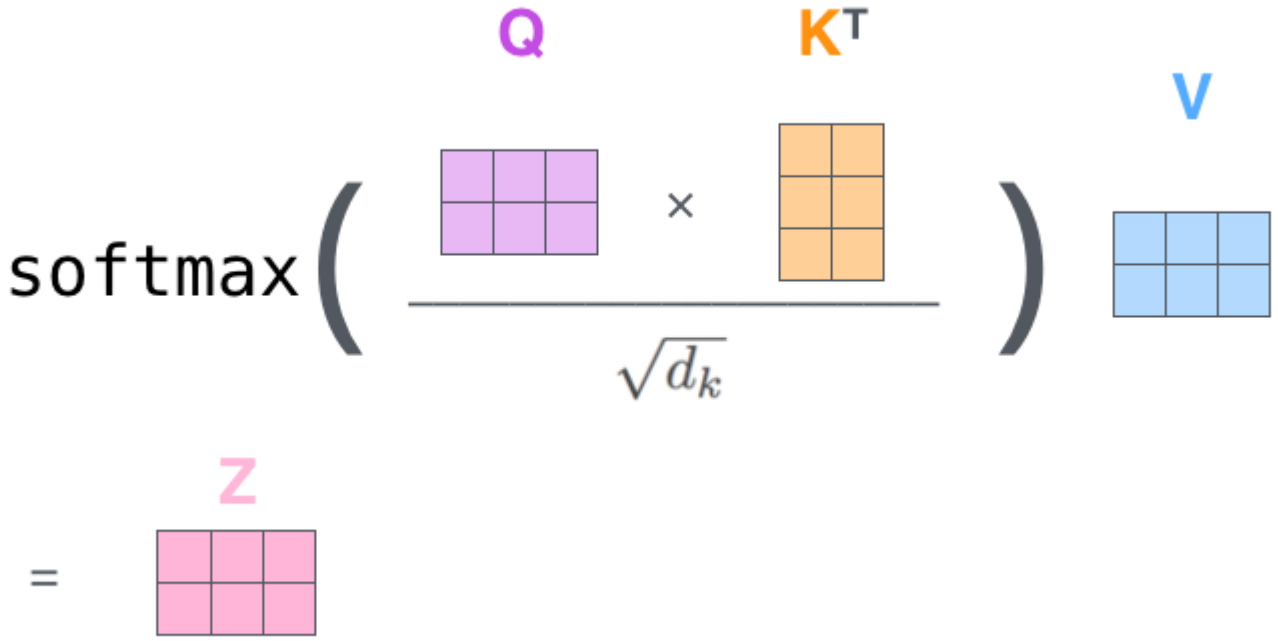
Self attention – using matrices to represent sentences – each row in X is one word in the input sentence



TRANSFORMERS FOR LANGUAGE MODELLING

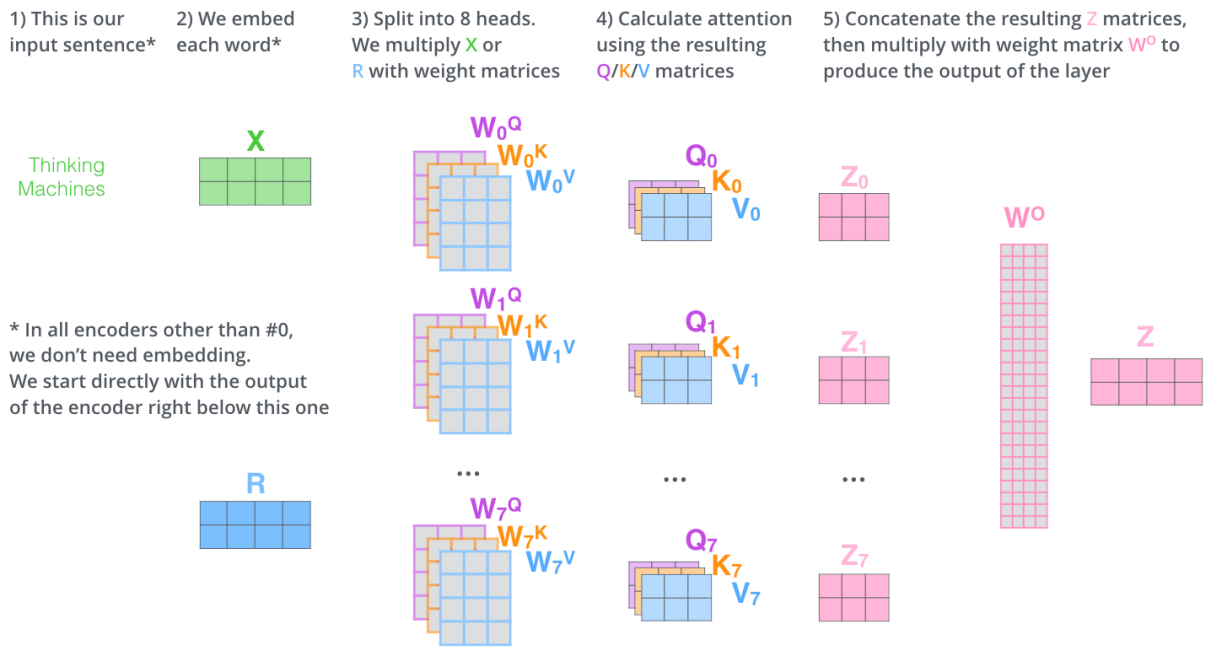
Self attention – using matrix representation allows us to calculate scoring and final output embedding in one step

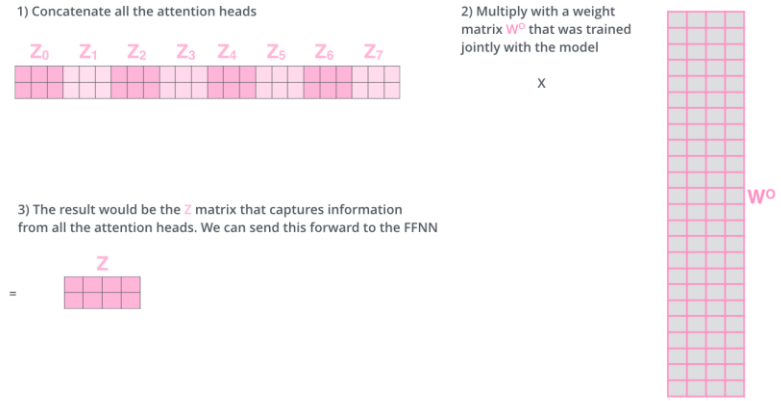
z is the final embedding of the input sentence, each row is one word



TRANSFORMERS FOR LANGUAGE MODELLING

Multi head attention – separate weight matrices for W^Q, W^K and W^V are maintained for each attention head 0, 1,..7 in the below figure, creating separate embedding matrices z that are combined into one final output embedding



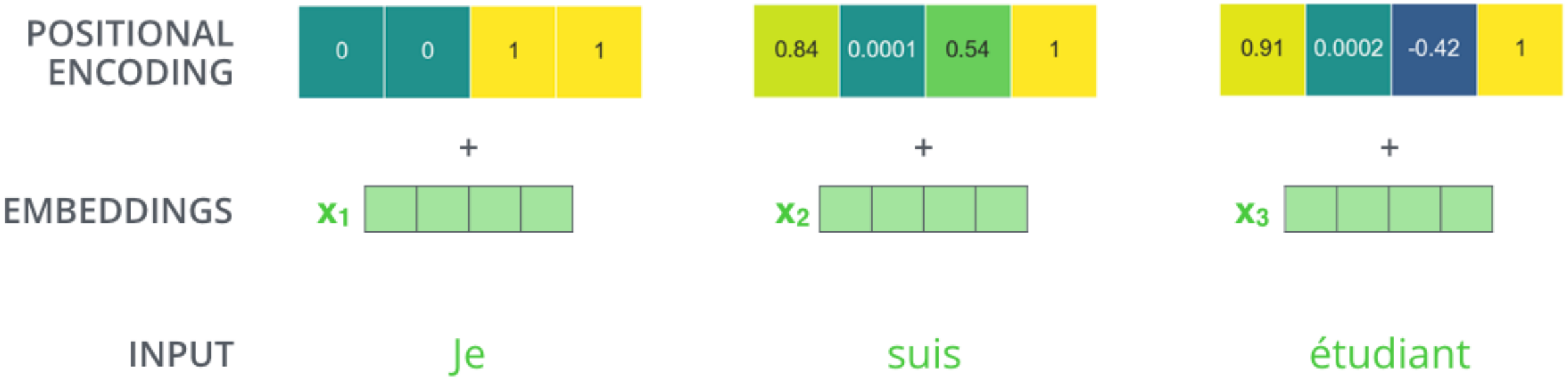


Step 5 in detail

(steps 1-4 detail as per slides 12-15)

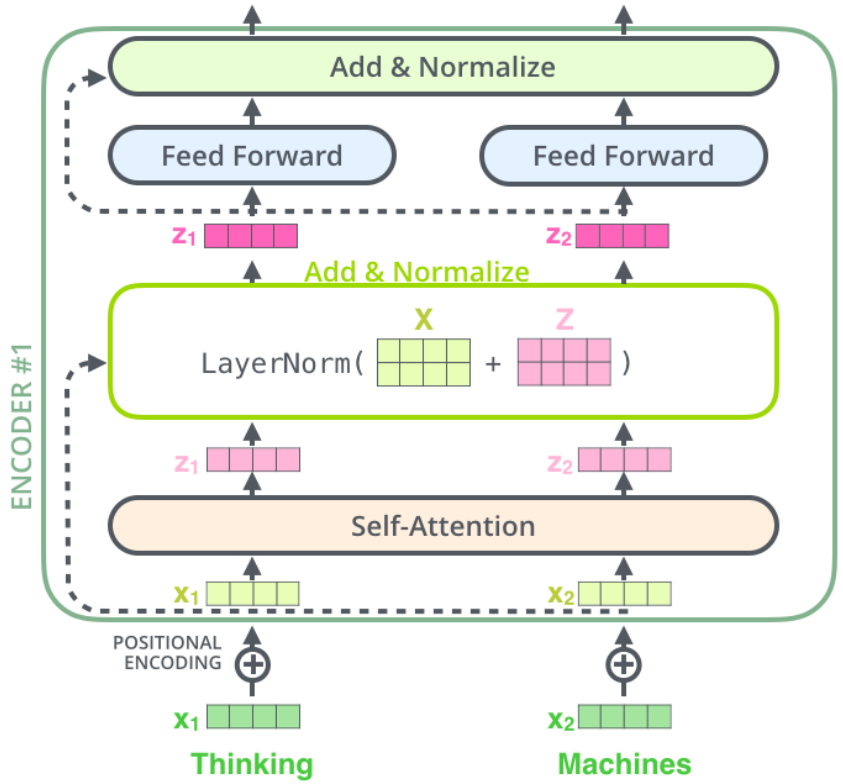
TRANSFORMERS FOR LANGUAGE MODELLING

Positional encoding vectors are added to the input vectors to give the model information about the order of the words



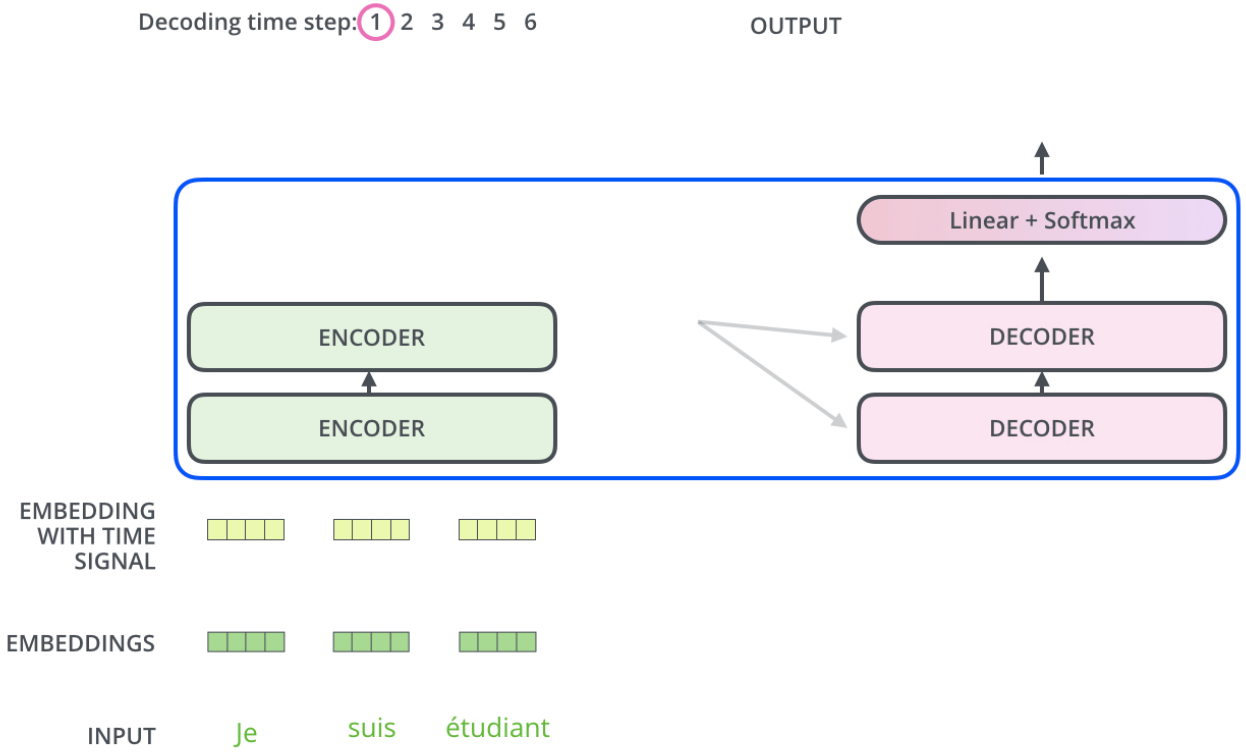
TRANSFORMERS FOR LANGUAGE MODELLING

The last step in the encoding block is to add a residual connection around each layer and a layer-normalisation step (similar to modern CNN models)



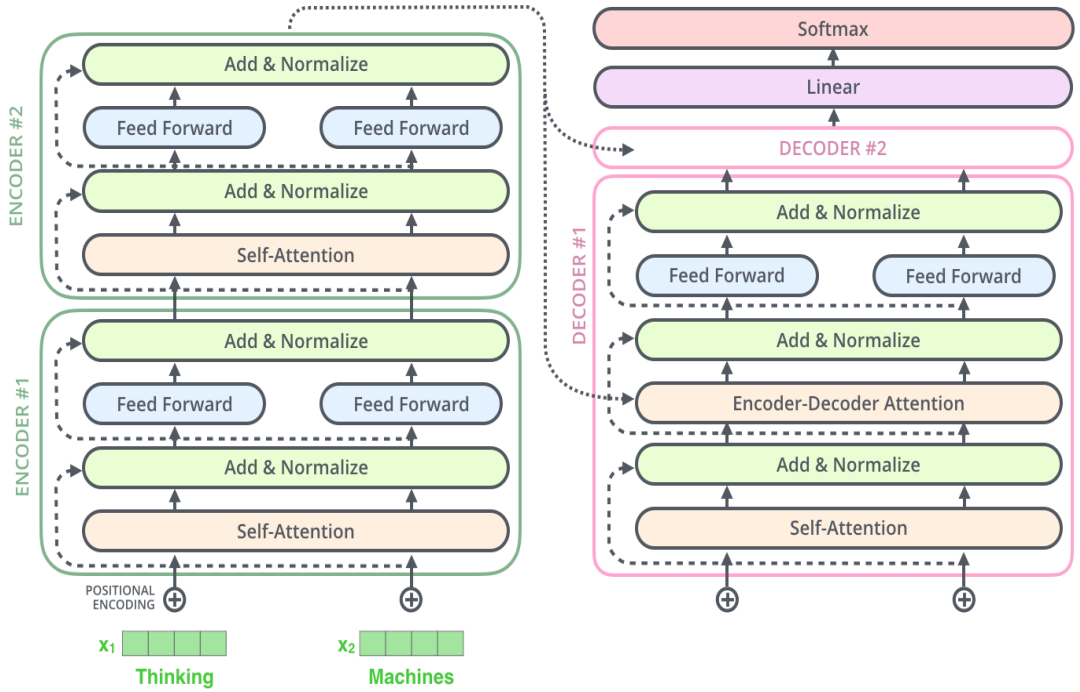
TRANSFORMERS FOR LANGUAGE MODELLING

Multiple encoder blocks are stacked on top of each other (6 in the original paper) to produce the final output and set of attention vectors K and V



TRANSFORMERS FOR LANGUAGE MODELLING

The decoding block looks very similar, except that it has two attention layers:



1. Self-attention as per the encoder takes in the output embedding for the previous word created + positional encoding. The raw logit values of all output words still to come in the sequence are set to –inf before the softmax is applied – called masked attention
2. Encoder-decoder attention as per the encoder self-attention, but the keys K and values V come from the output of the encoder and the queries come from the previous decoder layer

TRANSFORMERS FOR LANGUAGE MODELLING

For more detailed information and annotation of code

https://nlp.seas.harvard.edu/2018/04/03/attention.html