

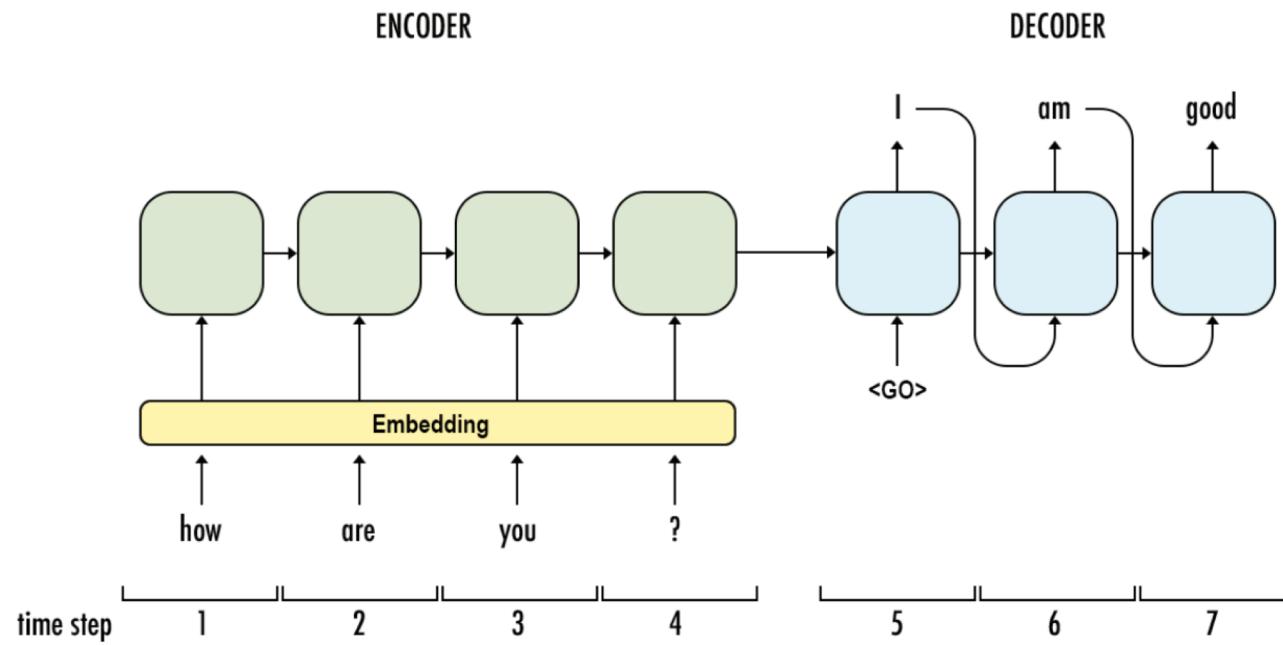
Seq2Seq Learning Advancements

Hardeep Arora

About Me

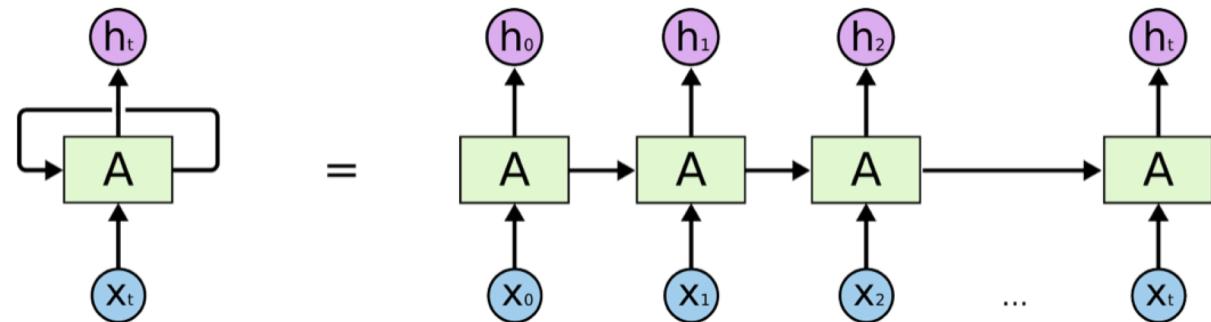
- 18+ years experience in Banking and Finance (MBA Finance, B.E. Computers)
- Have worked as Developer, DBA, Data Scientist, and Leading Analytics functions at Bank.
- Have delivered Data Warehouses and BI platforms for few banks.
- Have worked in Marketing Analytics
- Currently Heading COE AI/Analytics @ Accenture focusing on Finance and Risk domain in ASEAN region.
- Advisor for some startup's on AI Strategy
- Work Life : Credit Risk and Financial Crime
- Night Life : Deep Learning, Blockchain, AGI, Kaggle

Seq2Seq

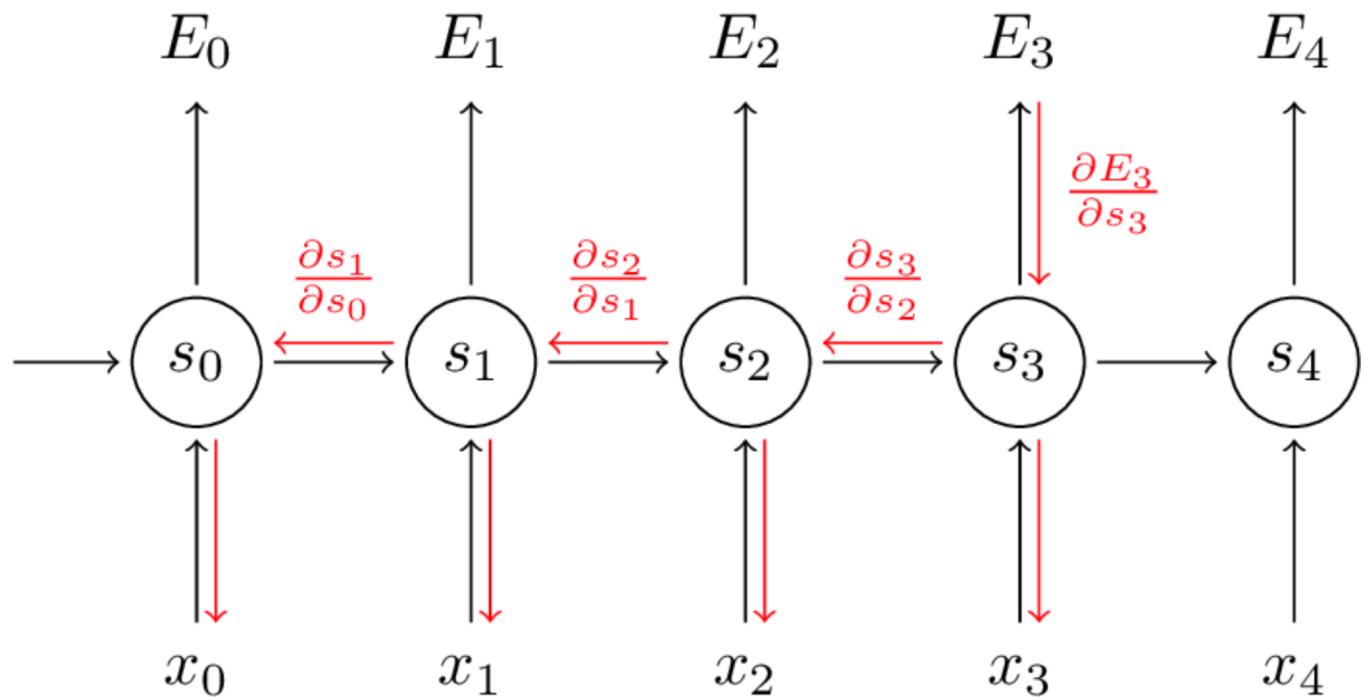


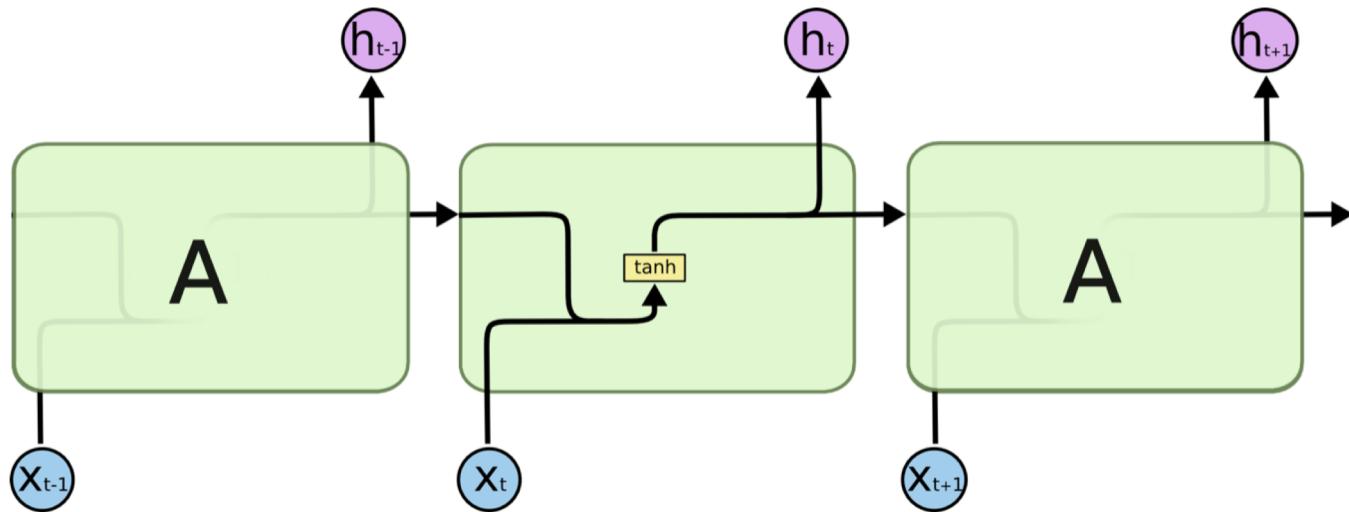
RNN

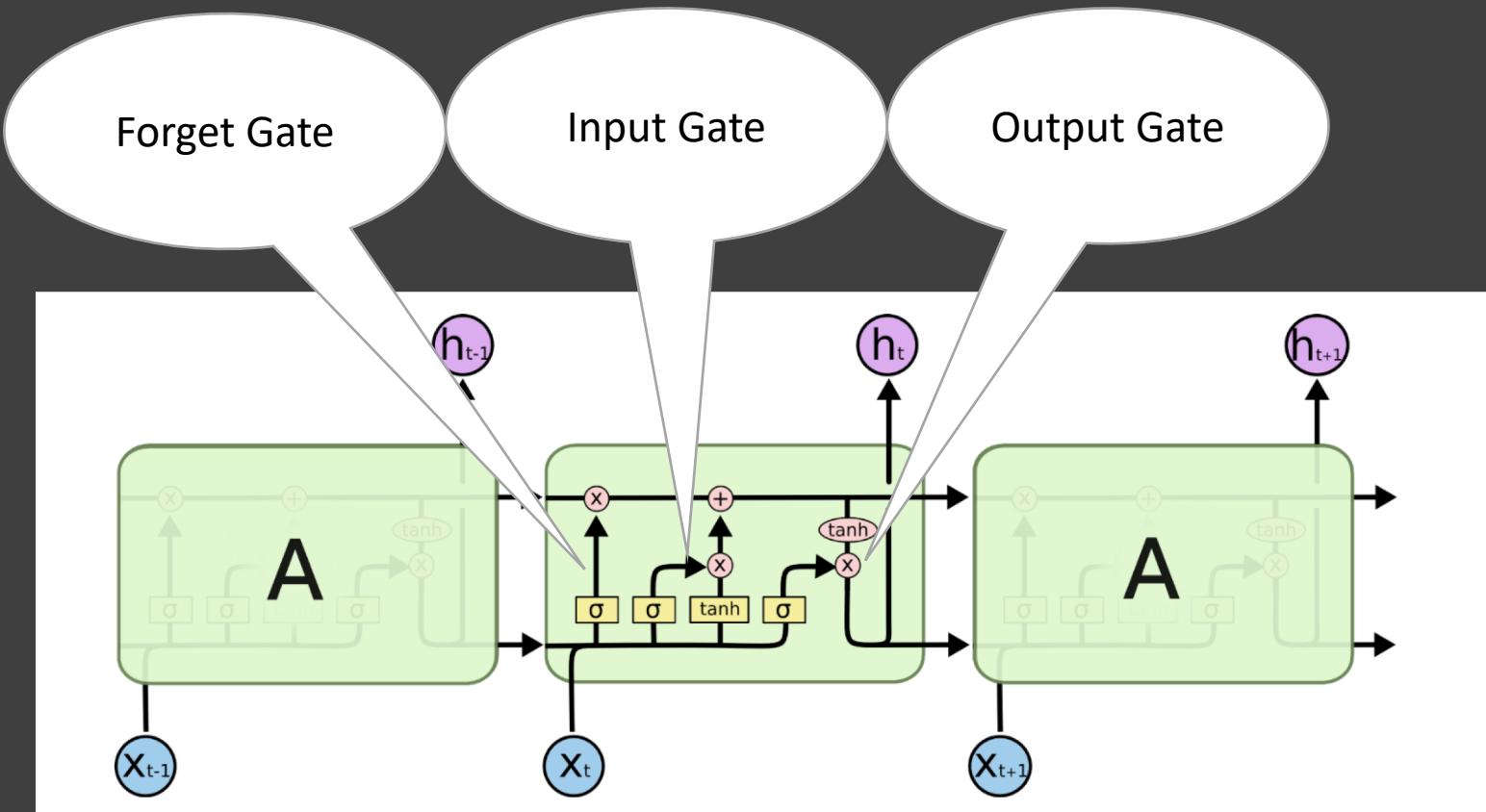
- RNN can be the encoder and decoder part of the Seq2Seq network seen in previous slide
- Can carry short-term context over. E.g. “the clouds are in the sky,”
- But struggles with long-term context. E.g. “I grew up in France... I speak fluent French.”



RNN Vanishing Gradients







To the rescue, came the LSTM module, which today can be seen as multiple switch gates, and a bit like [ResNet](#) it can bypass units and thus remember for longer time steps. LSTM and GRU and derivatives are able to learn a lot of longer term information! But they can remember sequences of 100s, not 1000s or 10,000s or more.

Seq2Seq based on LSTM applications

- Speech to Text – Siri, Cortana, Google Home, Alexa
- Neural Machine Translation
- Image to text and text to image
- Video captioning

Compute Challenges lead to use of CNN's

- RNNs operate in a strict left-to-right or right-to-left order, one word at a time. This is a less natural fit to the highly parallel GPU hardware that powers modern machine learning. The computation cannot be fully parallelized, because each word must wait until the network is done with the previous word
- In comparison, CNNs can compute all elements simultaneously, taking full advantage of GPU parallelism. They therefore are computationally more efficient. Another advantage of CNNs is that information is processed hierarchically, which makes it easier to capture complex relationships in the data.

Attention

$$\alpha_{ts} = \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'=1}^S \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))}$$

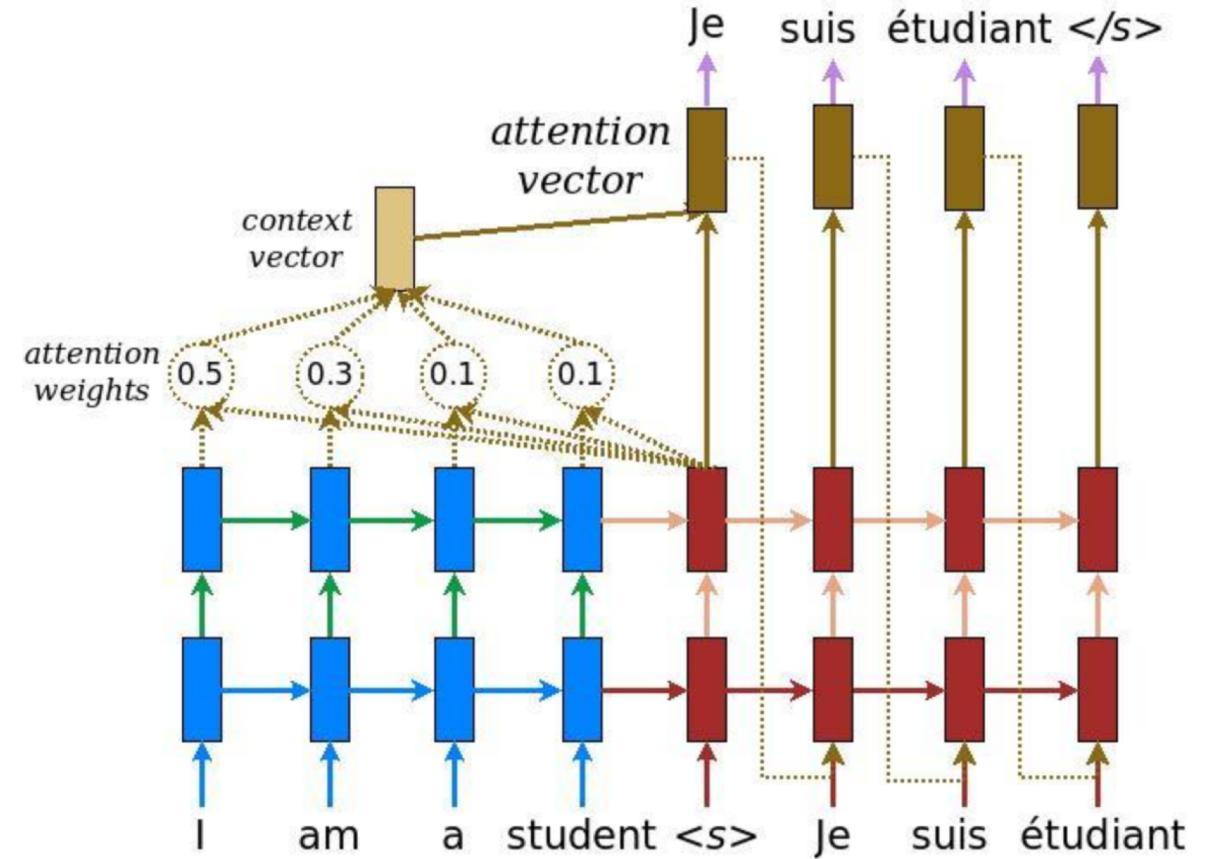
[Attention weights]

$$\mathbf{c}_t = \sum_s \alpha_{ts} \bar{\mathbf{h}}_s$$

[Context vector]

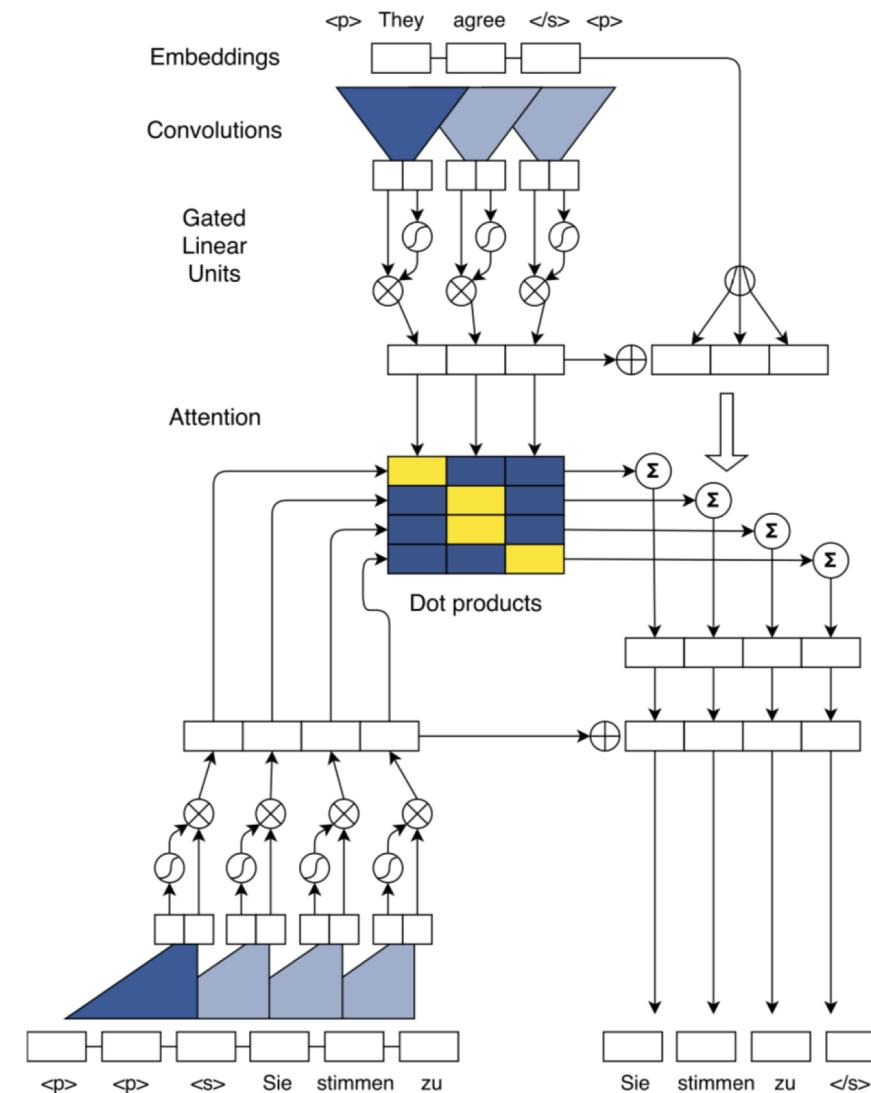
$$\mathbf{a}_t = f(\mathbf{c}_t, \mathbf{h}_t) = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t])$$

[Attention vector]



Convolutional Seq2Seq

- Top is the encoder
- Bottom left is decoder
- Attention is dot product of encoder and decoder

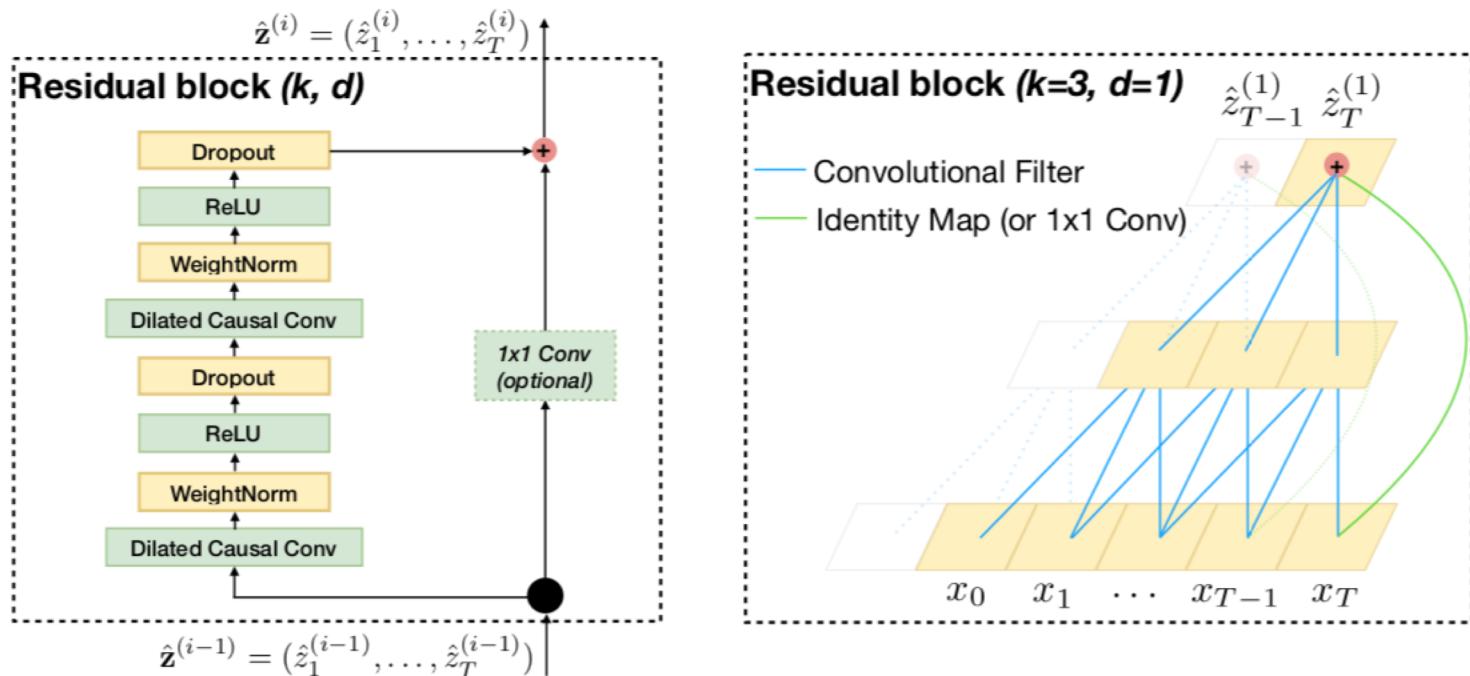
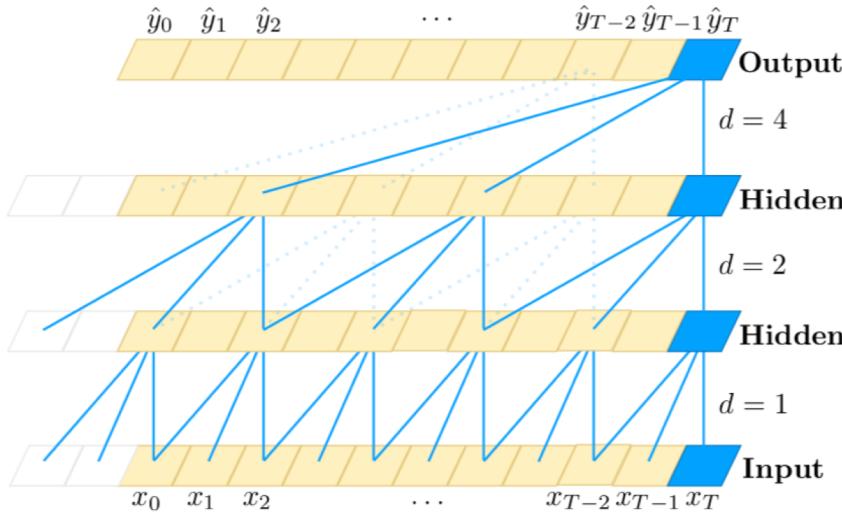


NMT – Application of CNN (Facebook)

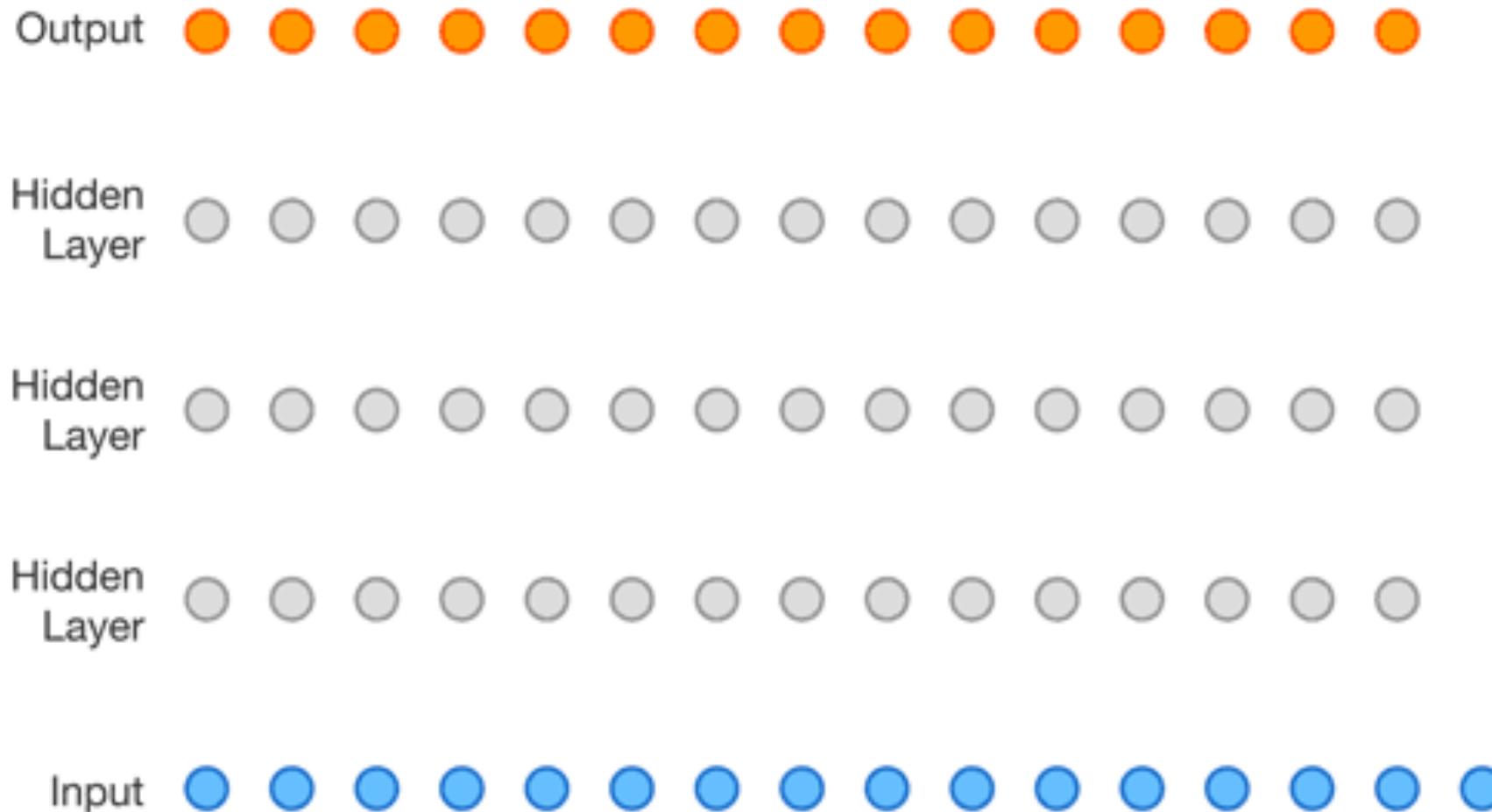
la maison de Léa <end>

Temporal convolutional network (TCN)

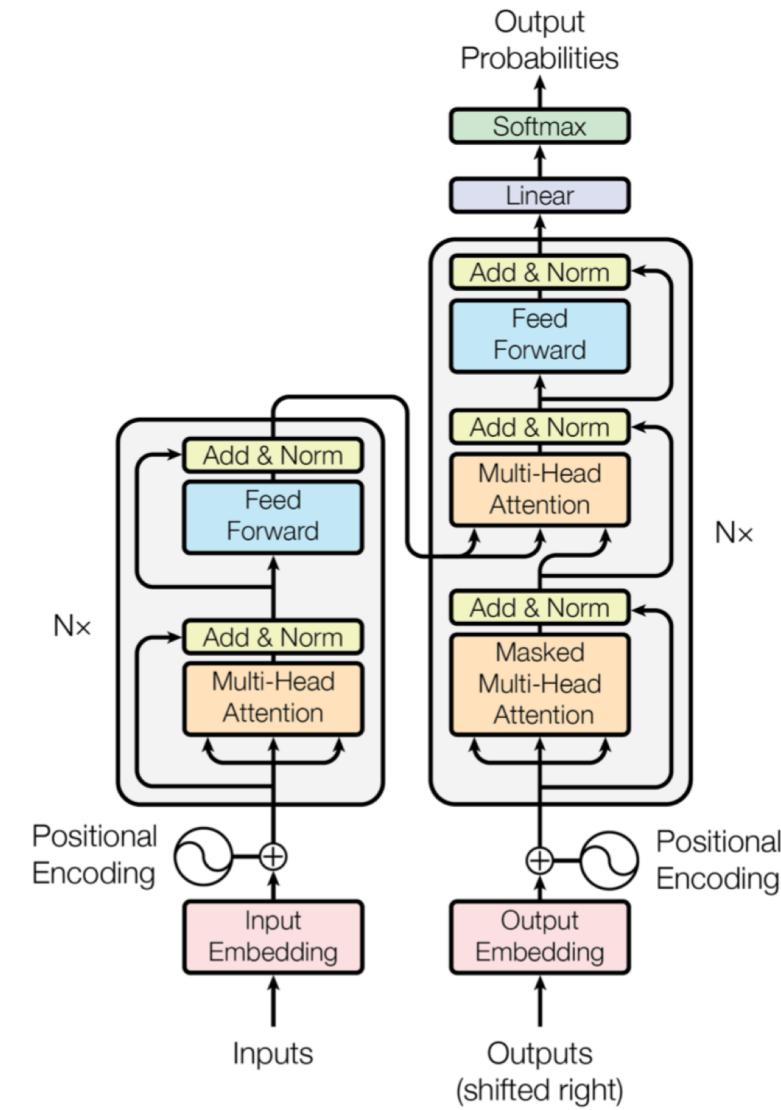
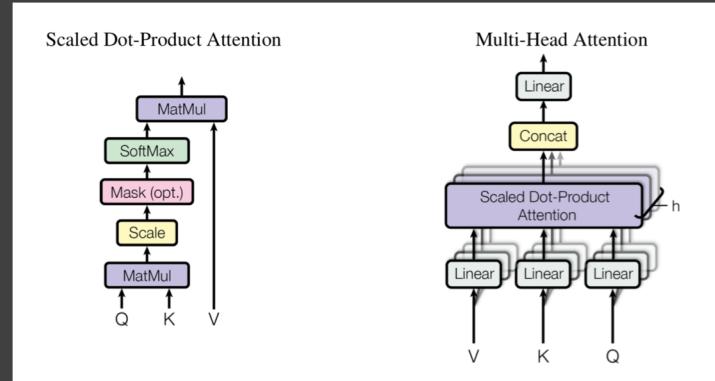
- TCNs convincingly outperform baseline recurrent architectures across a broad range of sequence modelling tasks.
- TCN's also exhibit substantially longer memory, and are thus more suitable for domains where a long history is required.



WaveNet2 – Using Dilated CNN's



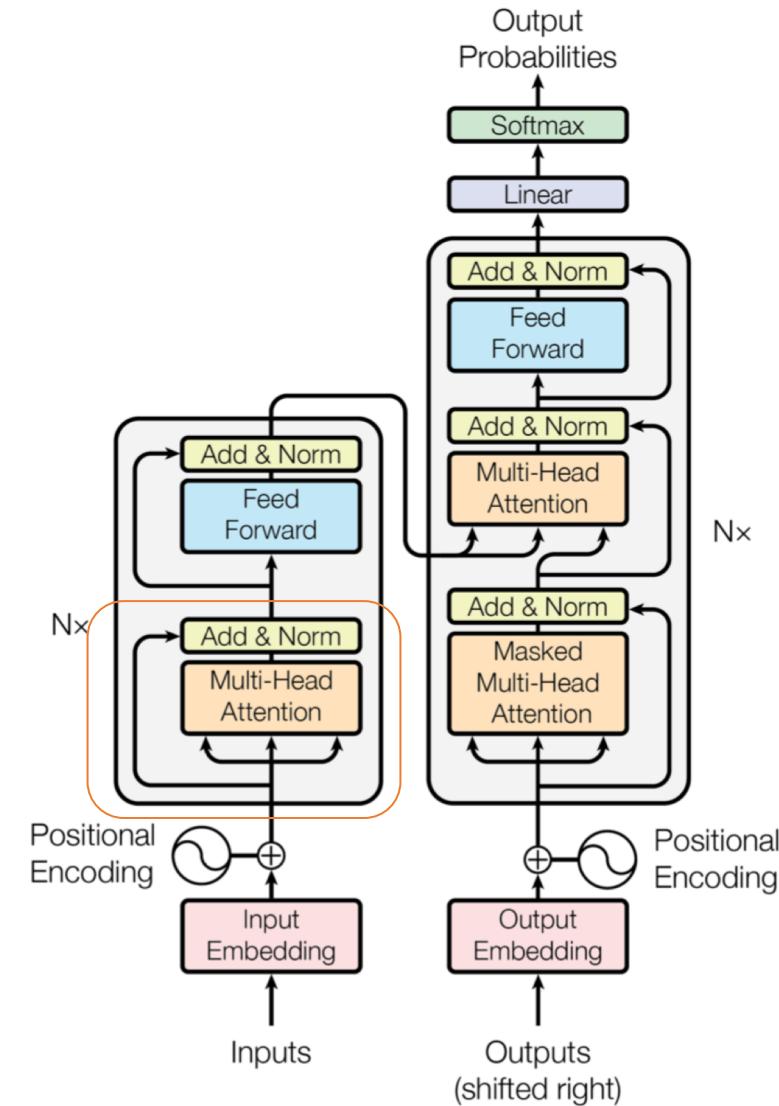
Attention is all you need



<https://arxiv.org/pdf/1706.03762.pdf>

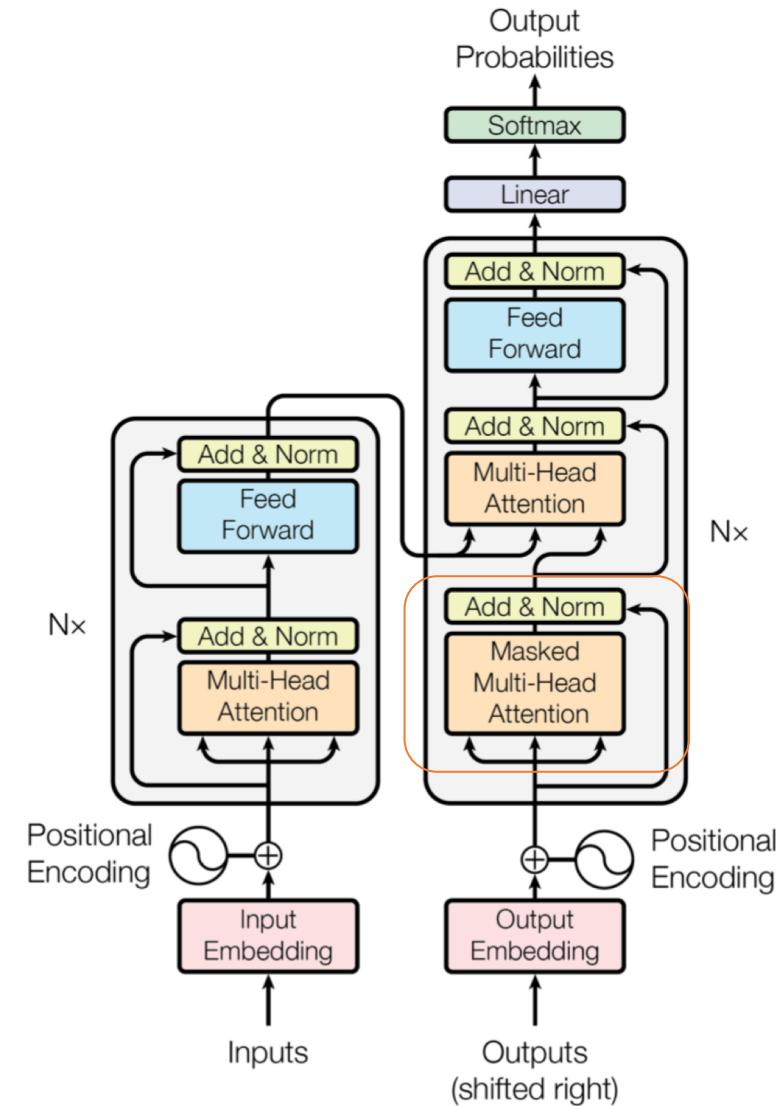
Attention is all you need

- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.



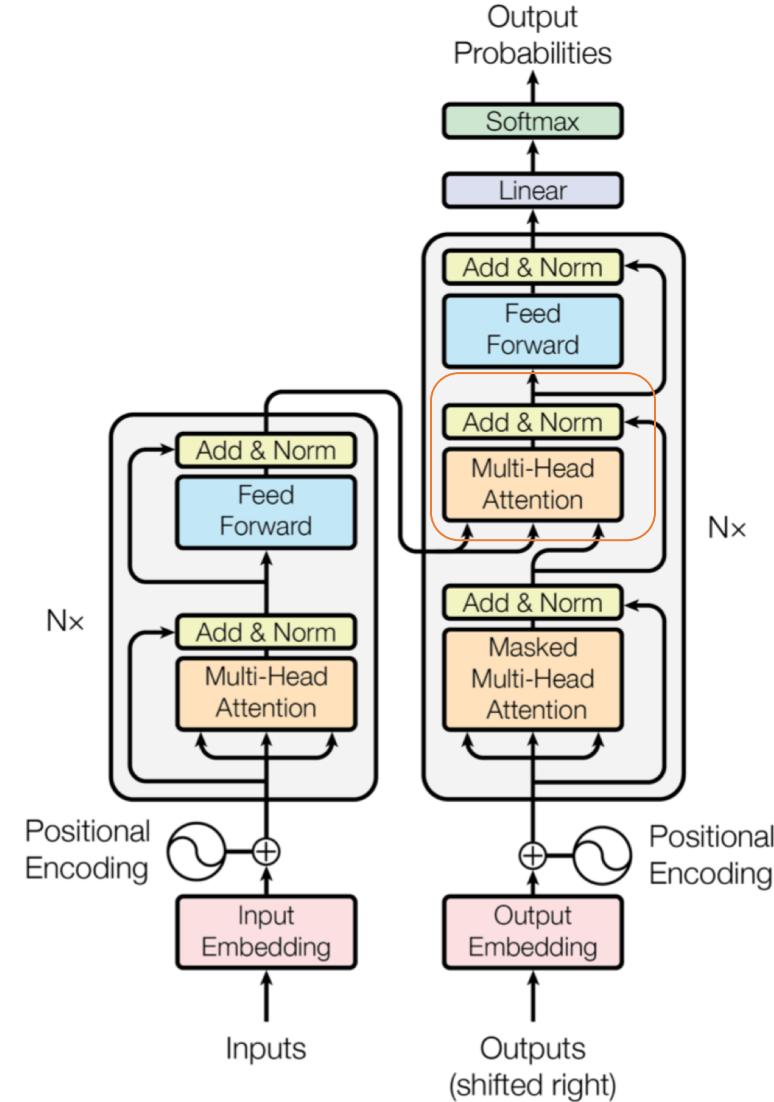
Attention is all you need

- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections.



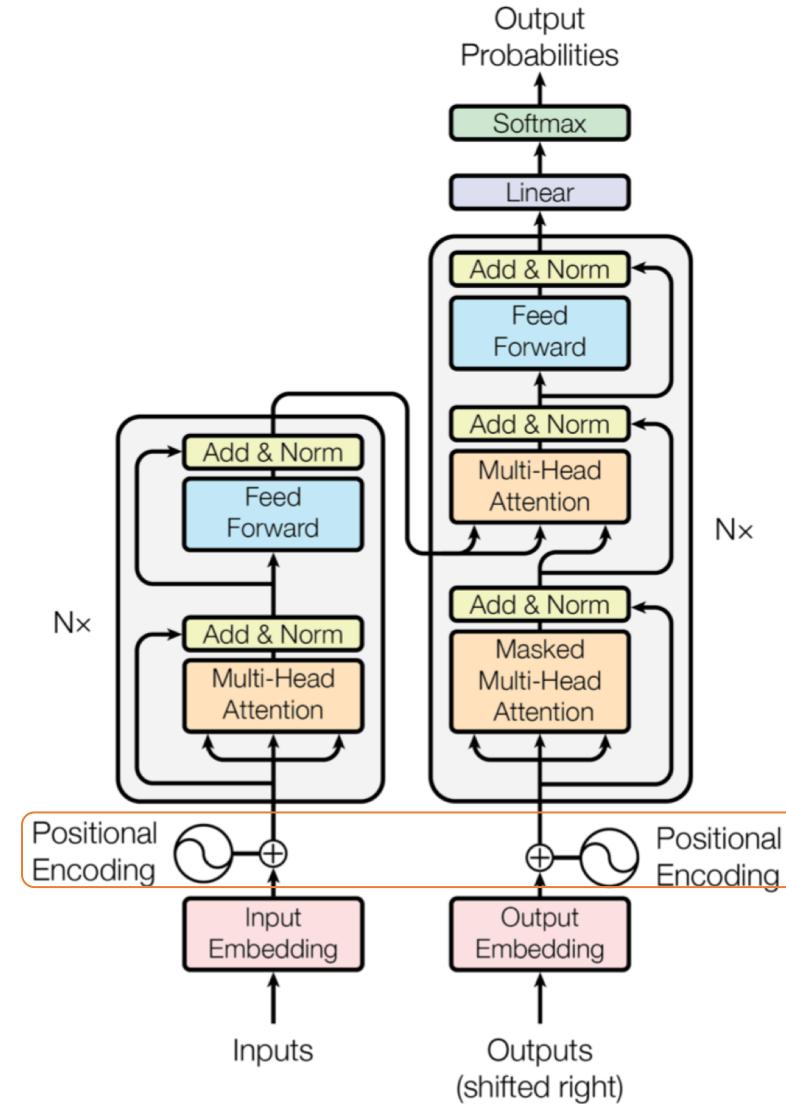
Attention is all you need

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models



Attention is all you need

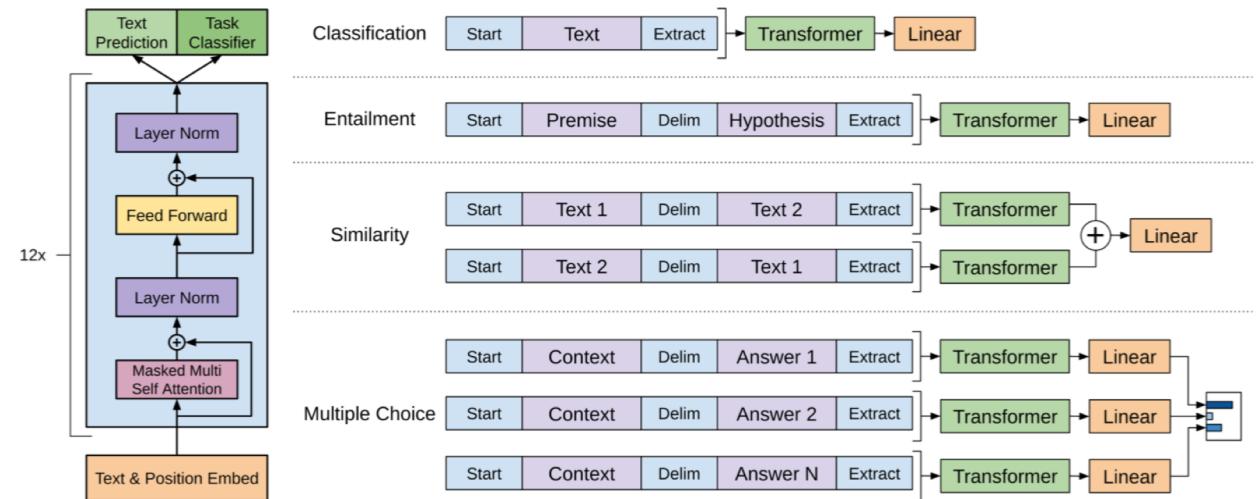
- Add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed



Improving Language Understanding by Generative Pre-Training

Stage 1 - Unsupervised

We use a multi-layer *Transformer decoder* for the language model, which is a variant of the transformer. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens



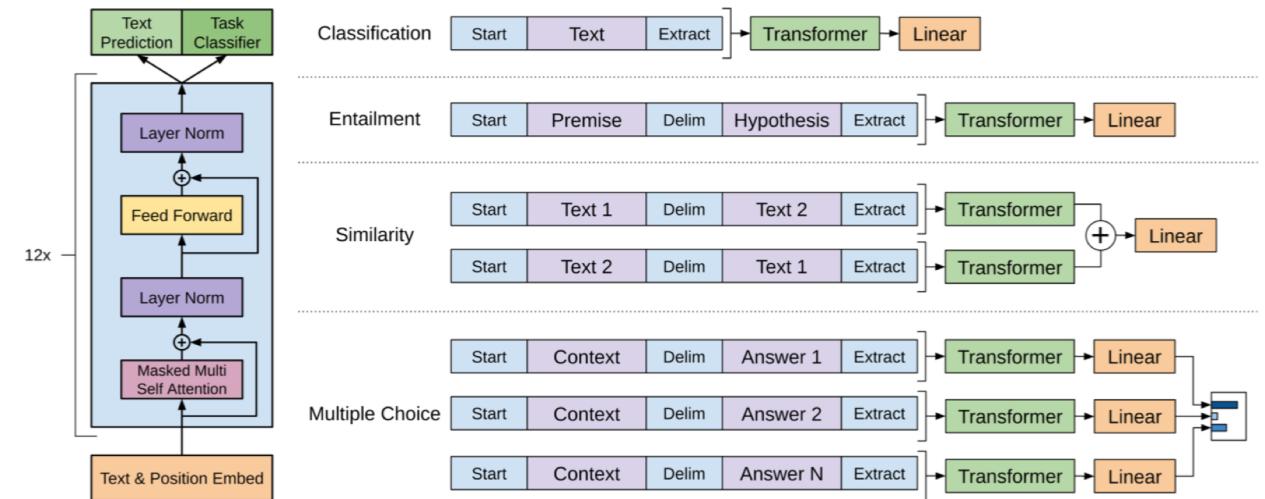
<https://blog.openai.com/language-unsupervised/>
<https://github.com/openai/finetune-transformer-lm>

Improving Language Understanding by Generative Pre-Training

Stage 2 - Supervised

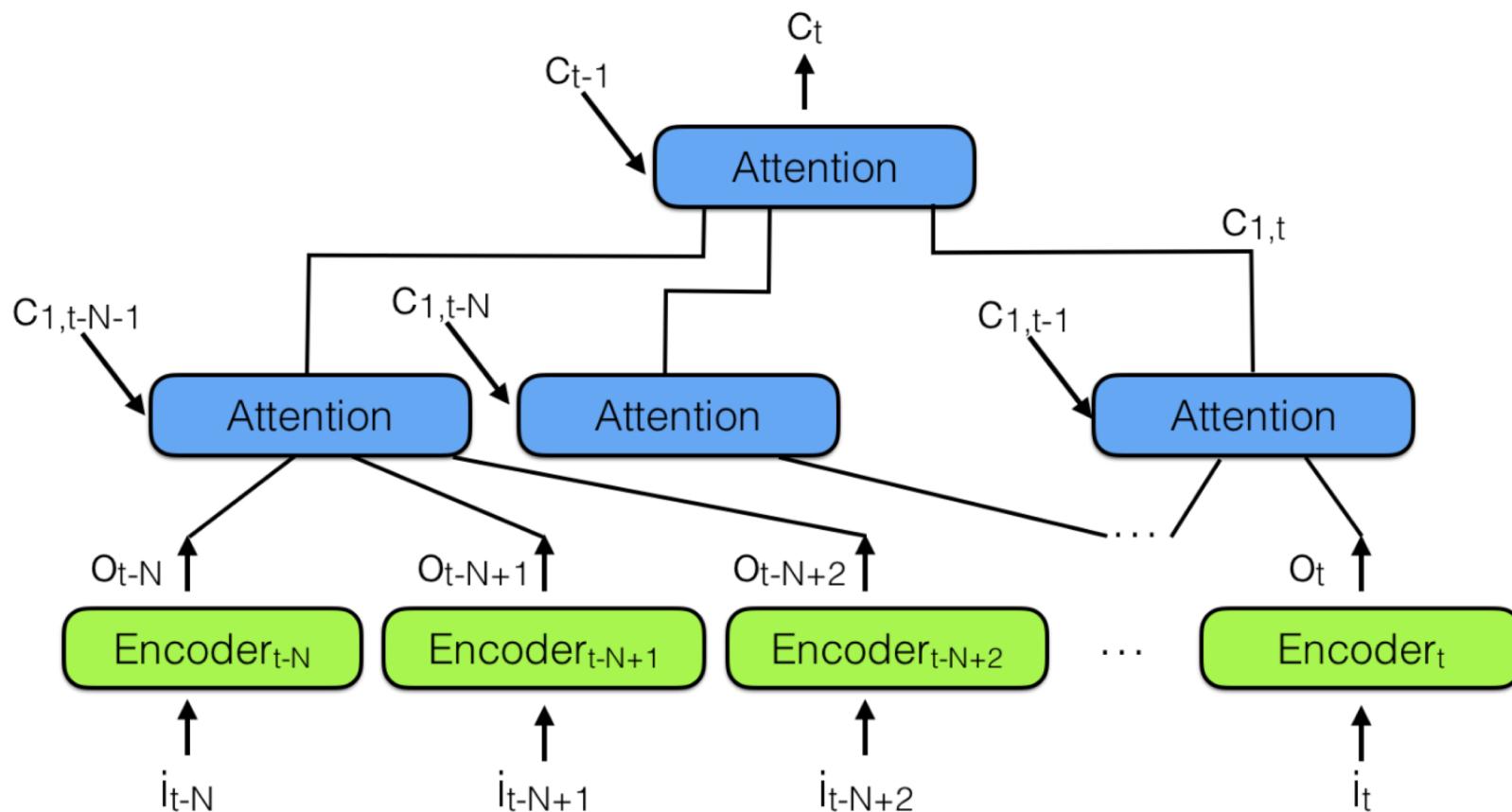
The fine tune the trained model on few label tasks to get task specific outputs.

They used the knowledge gained in the Transformer to transfer it on specific tasks, similar to transfer learning

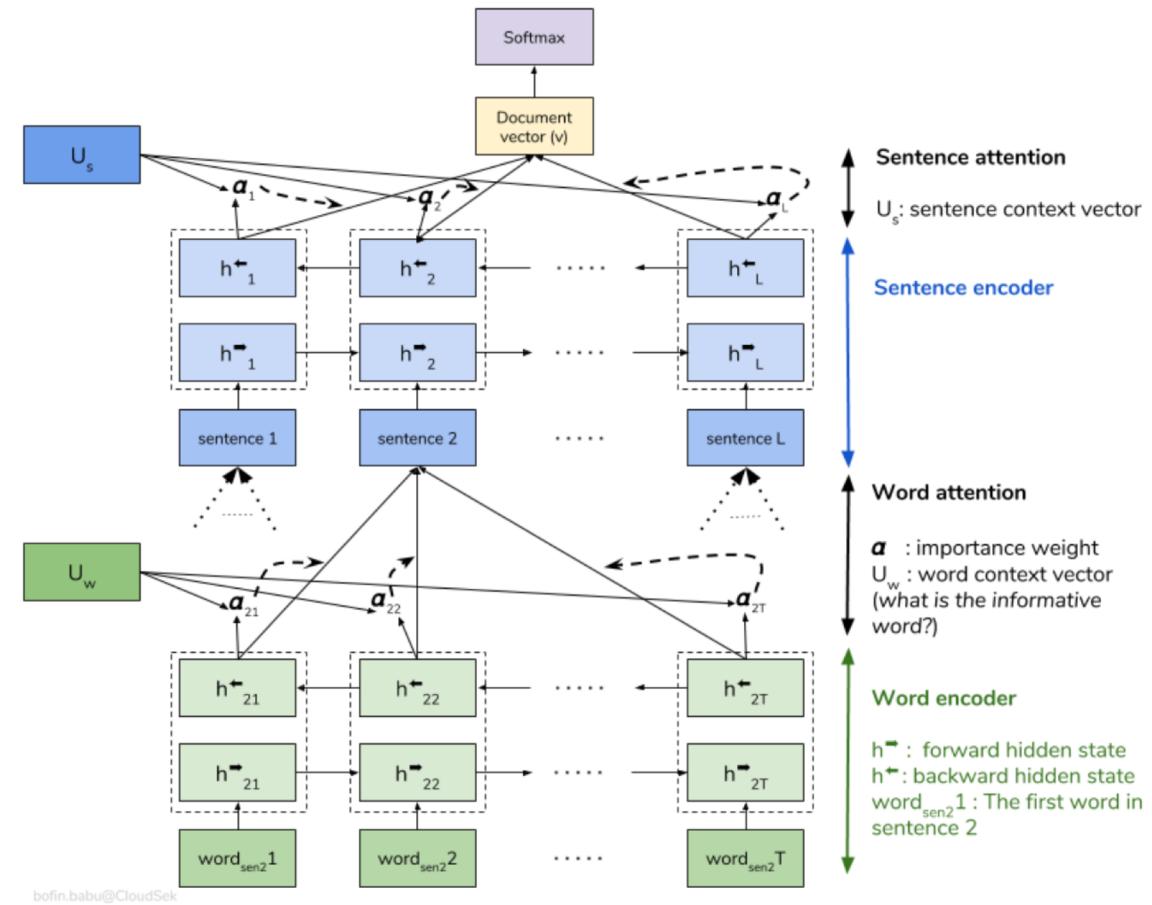


<https://blog.openai.com/language-unsupervised/>
<https://github.com/openai/finetune-transformer-lm>

HAN – Hierarchical Neural Attention Encoder



HAN - Examples



Thanks

- Connect with me on LinkedIn - www.linkedin.com/in/arorahardeep