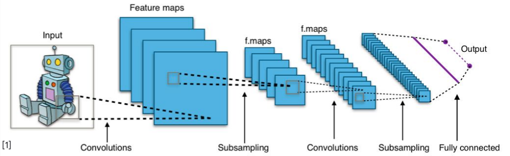
Transformer

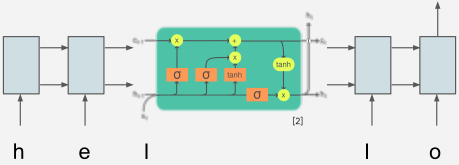
Computer Vision

Convolutional NNs (+ResNets)



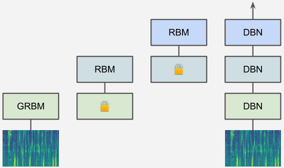
Natural Lang. Proc.

Recurrent NNs (+LSTMs)



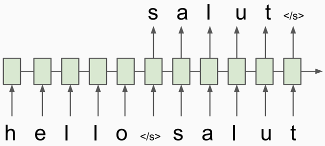
Speech

Deep Belief Nets (+non-DL)



Translation

Seq2Seq



RL

BC/GAIL

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \dots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

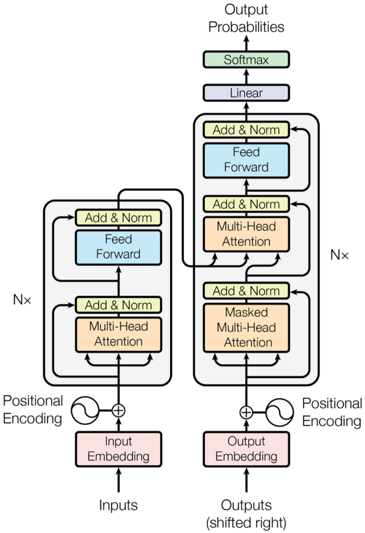
$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

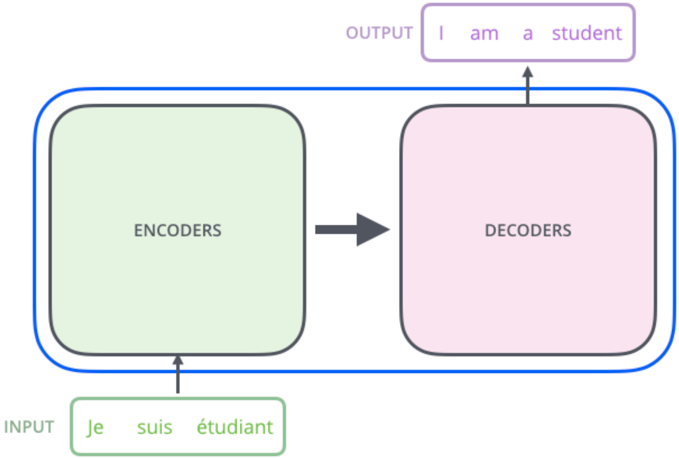
[1] CNN image CC-BY-SA by Aphex34 for Wikipedia https://commons.wikimedia.org/wiki/File:Typical_cnn.png
[2] RNN image CC-BY-SA by GChe for Wikipedia https://commons.wikimedia.org/wiki/File:The_LSTM_Cell.svg

Attention is All you Need

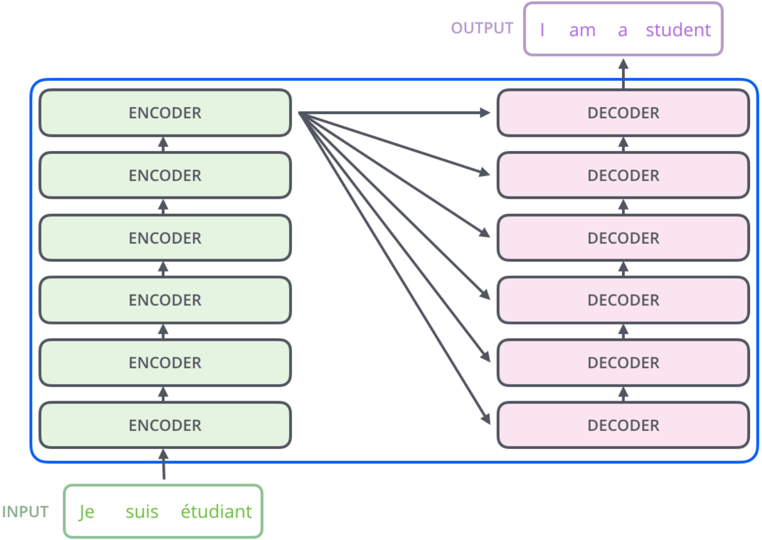
A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin (2017)



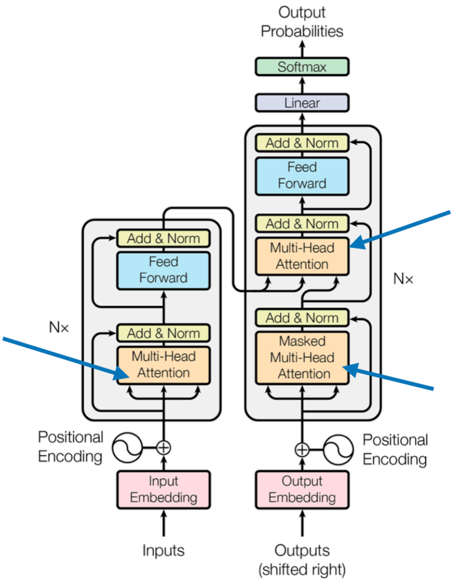
A big picture



A smaller picture

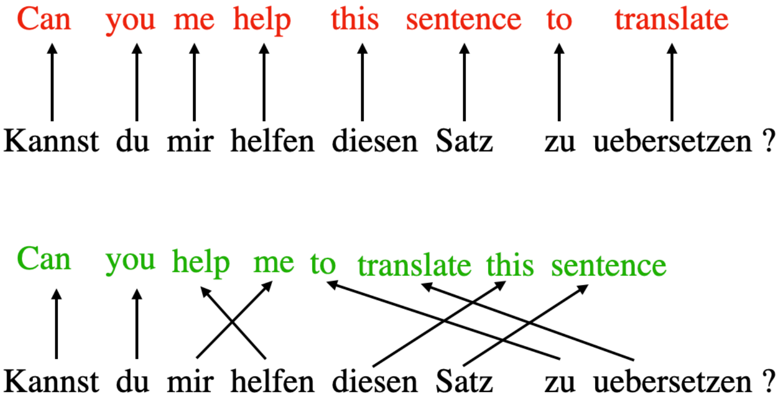


Attention

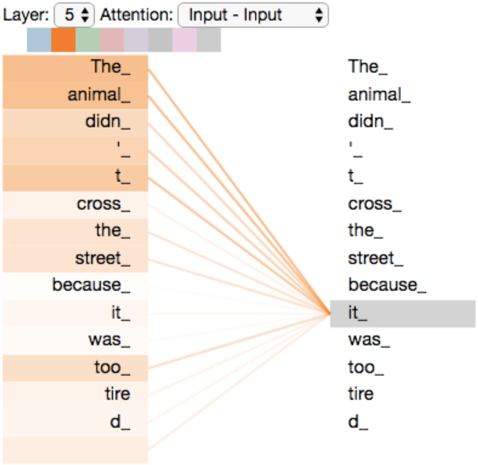


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Idea behind Self-Attention



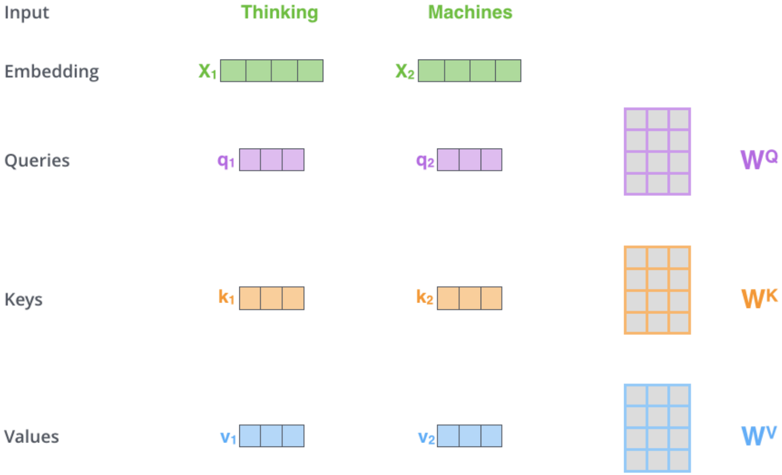
Self-Attention as finding relevant words



Try it yourself

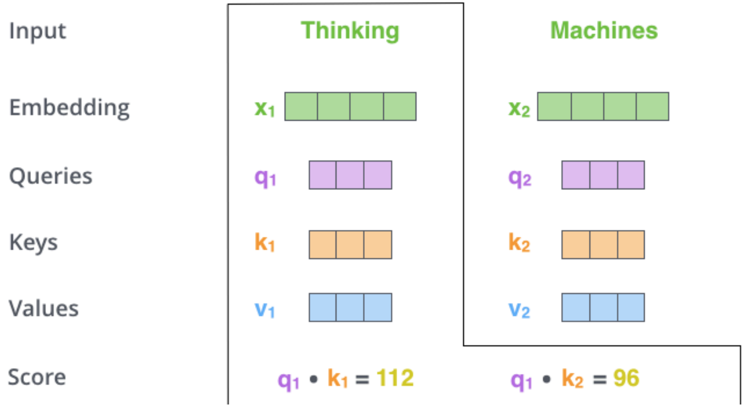
Calculating Self-Attention

Step 1: From an input vectors, create **Q**uery vector, **K**ey vector and **V**alue vector



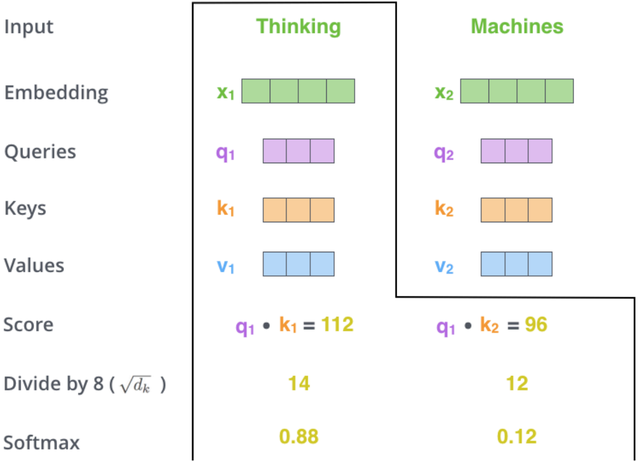
Calculating Self-Attention

Step 2: Compute the **scores** of each word by taking dot-product of its **query** and the **keys** of the all words



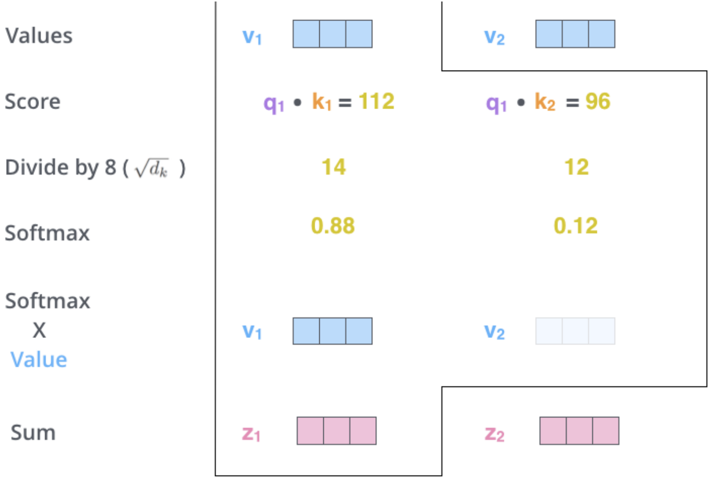
Calculating Self-Attention

Step 3: Divide by 8 (or the square root of the dimension of the key vectors) and apply Softmax



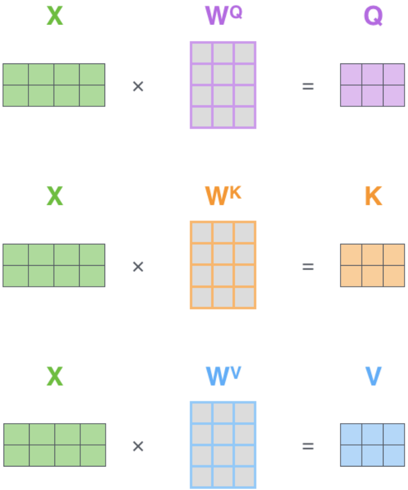
Calculating Self-Attention

Step 4: Multiply each value vector by the softmax score, then sum the vectors. Irrelevant words with low softmax scores will not contribute much to the sum

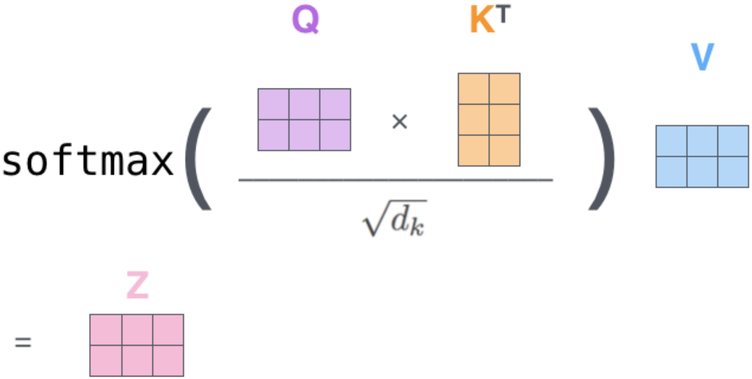


Self-Attention with Multiple inputs

We can write multiple dot-products as a matrix multiplication



Self-Attention in one equation



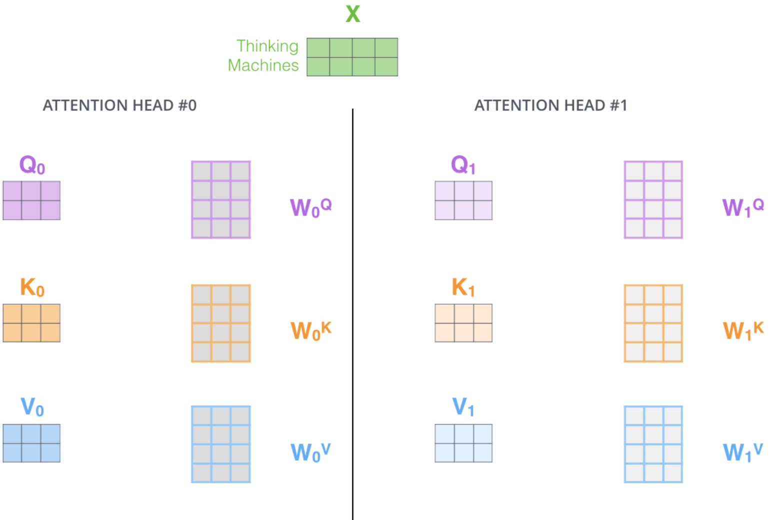
Multi-head Attention

Motivation: Same words can have multiple meanings:

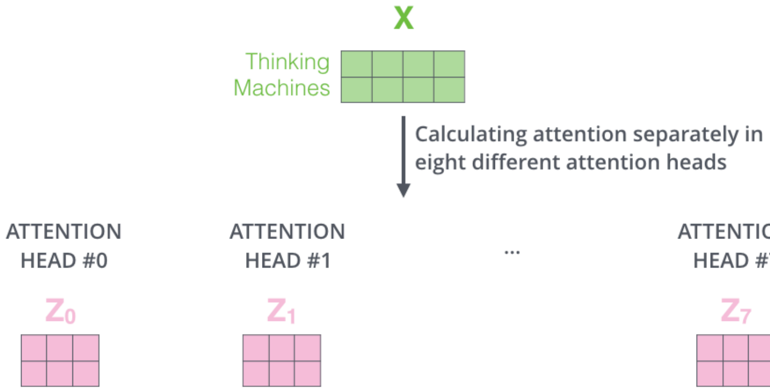
"She turned on the **light**"   vs   "The bag was very **light**"

Multi-head Attention

Motivation: Same words can have multiple meanings:



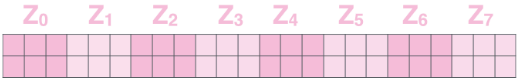
Multi-head Attention



but the feed-forward layer only takes a single matrix. How do we combine these into a single matrix?

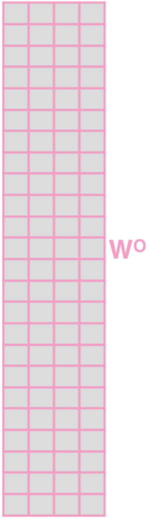
Combining matrices

1) Concatenate all the attention heads



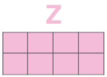
2) Multiply with a weight matrix W^O that was trained jointly with the model

X



3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



Summary of Multi-head Self-Attention

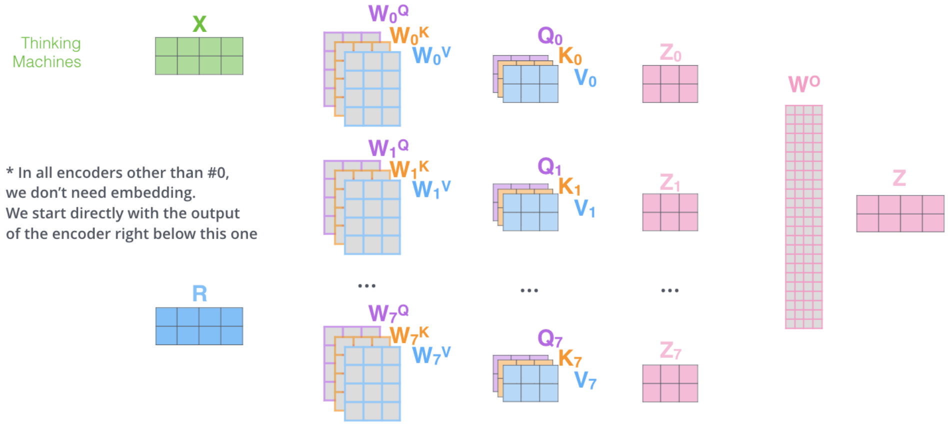
1) This is our input sentence*

2) We embed each word*

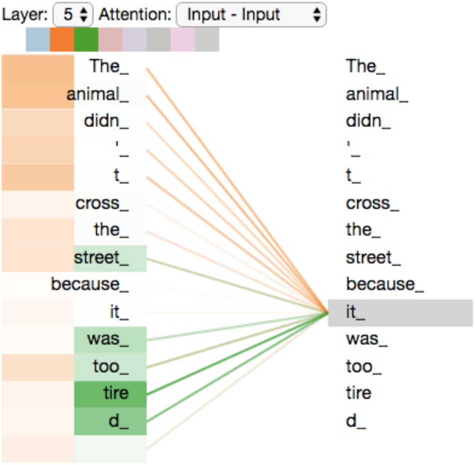
3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

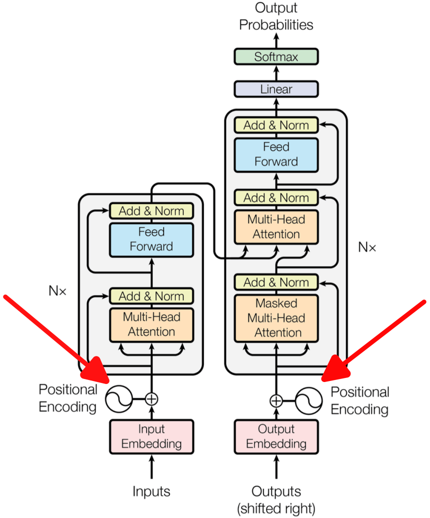


Revisit the Visualization



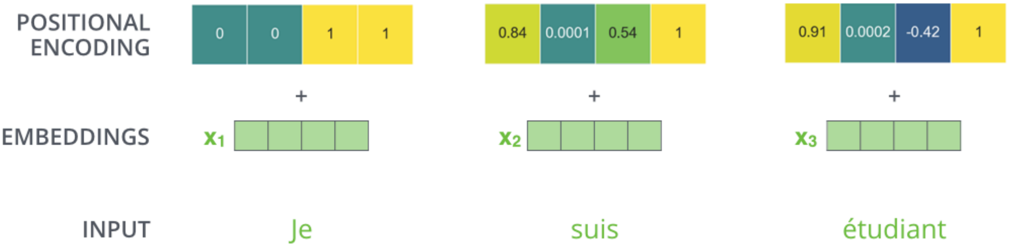
Positional Encoding

Issue: The model does not know the order of words



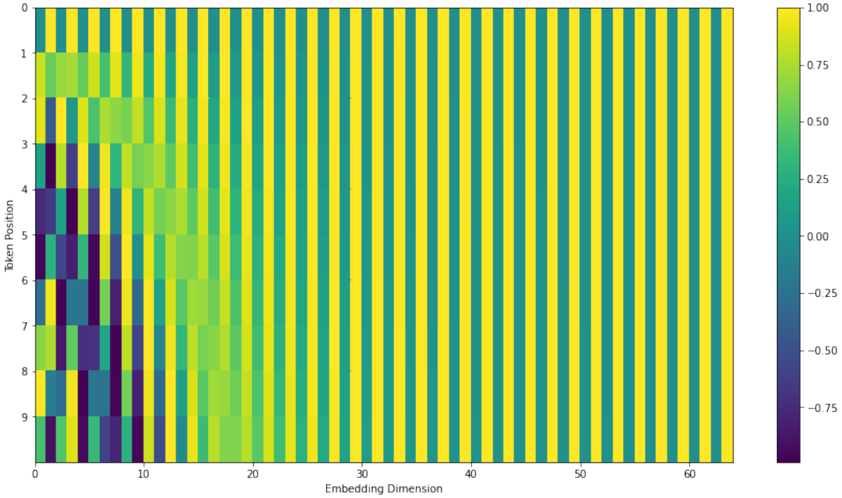
Positional Encoding

Solution: Add different vectors to the sequence of vectors (**positional encoding**)

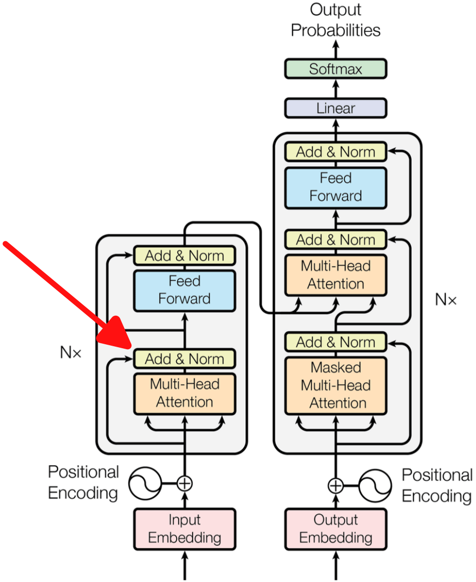


Positional Encodings of first 10 words

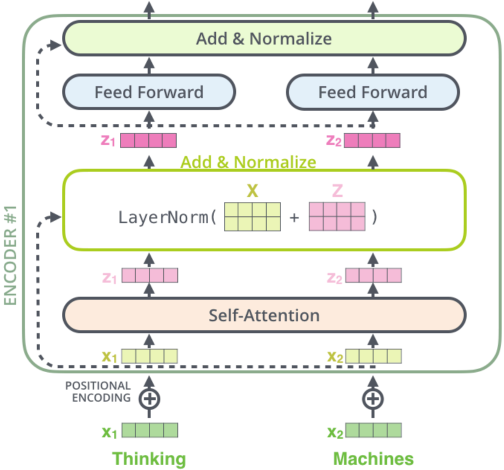
If each word is a 64-dim vector. Below are positional encodings of first 10 words



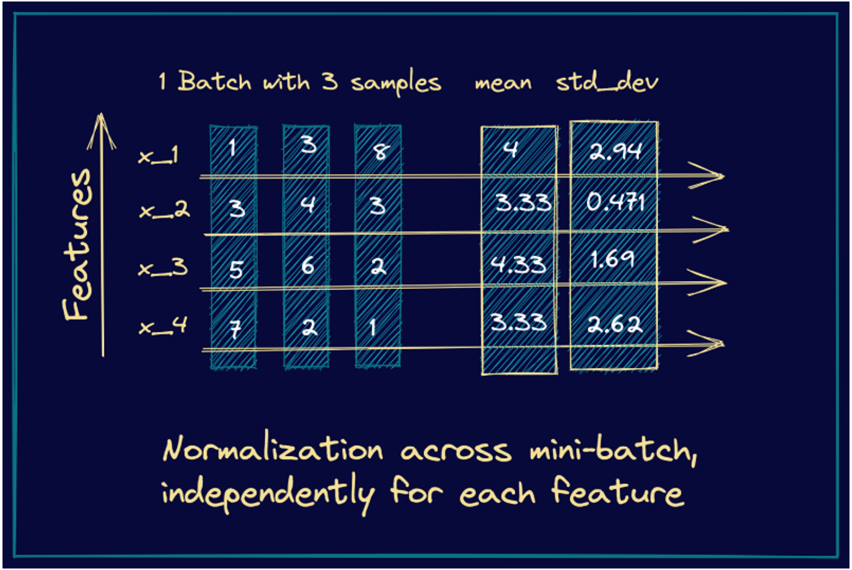
Addition & Normalization



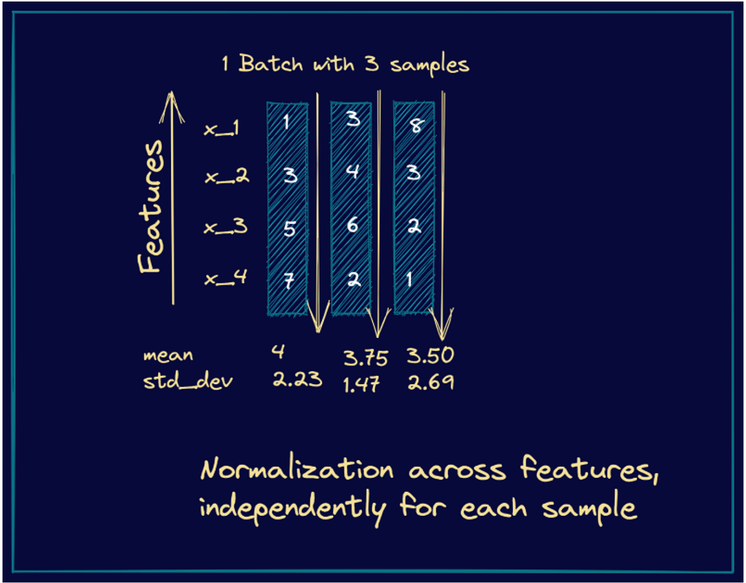
Addition & Normalization



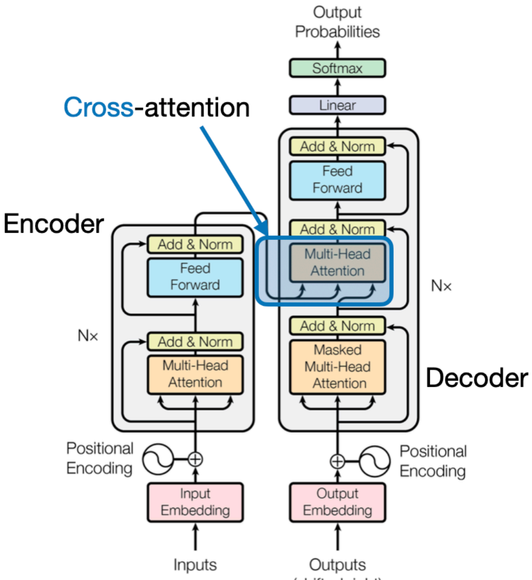
Review: Batch normalization



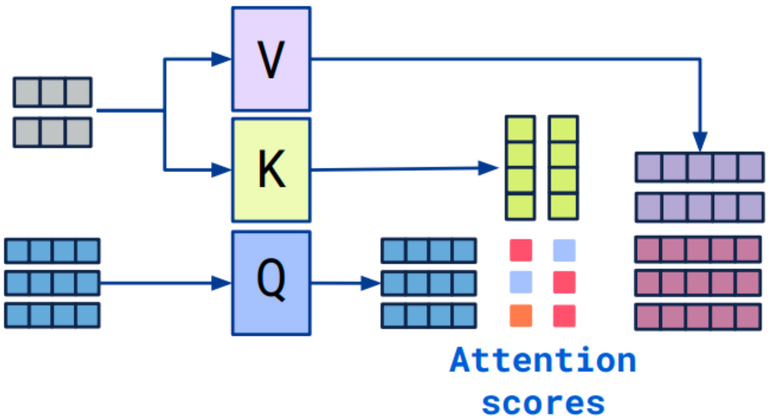
Layer normalization



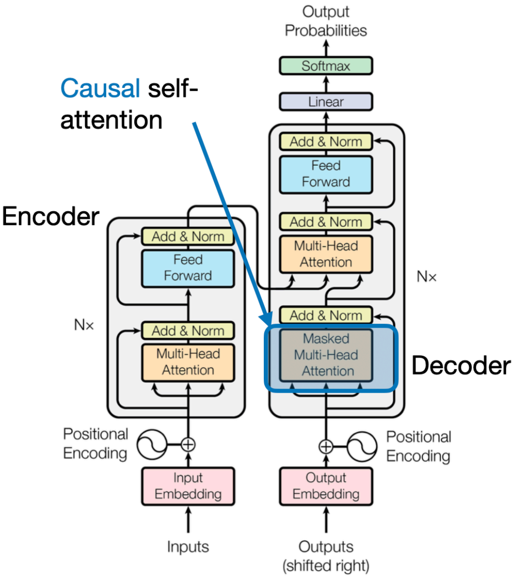
The Decoder: Cross-Attention



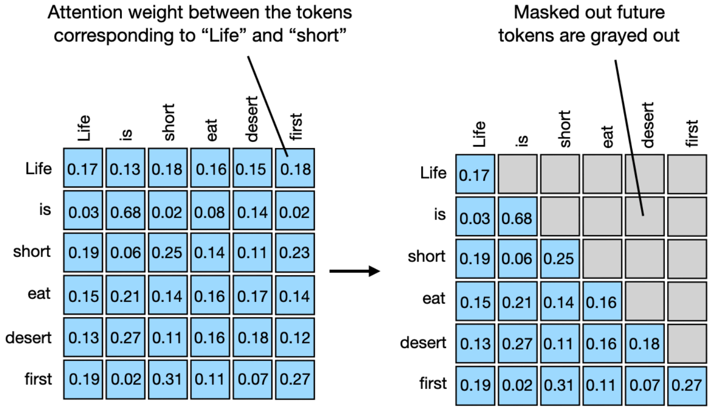
Cross-Attention



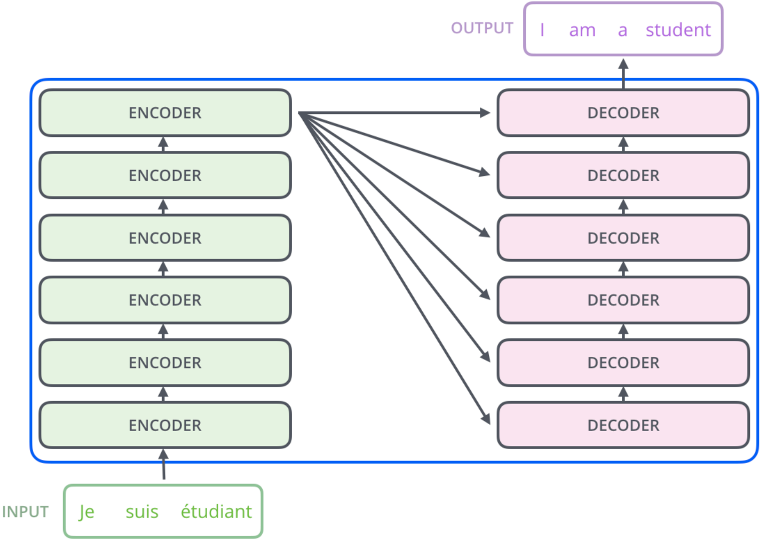
Causal attention



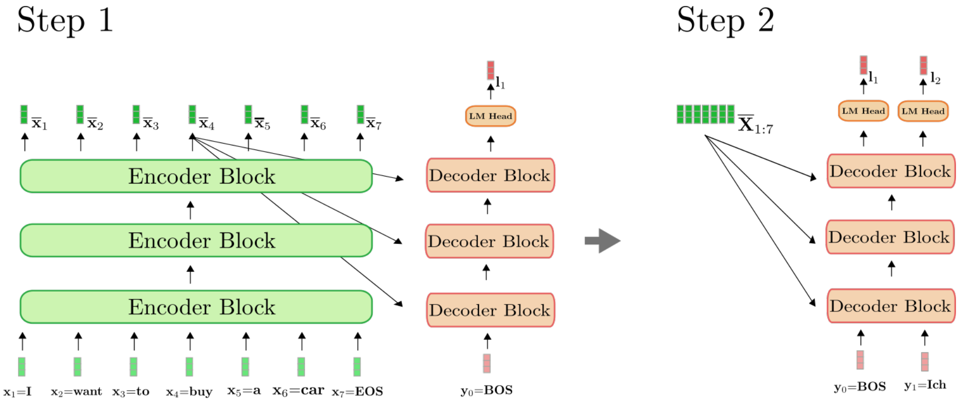
Causal attention



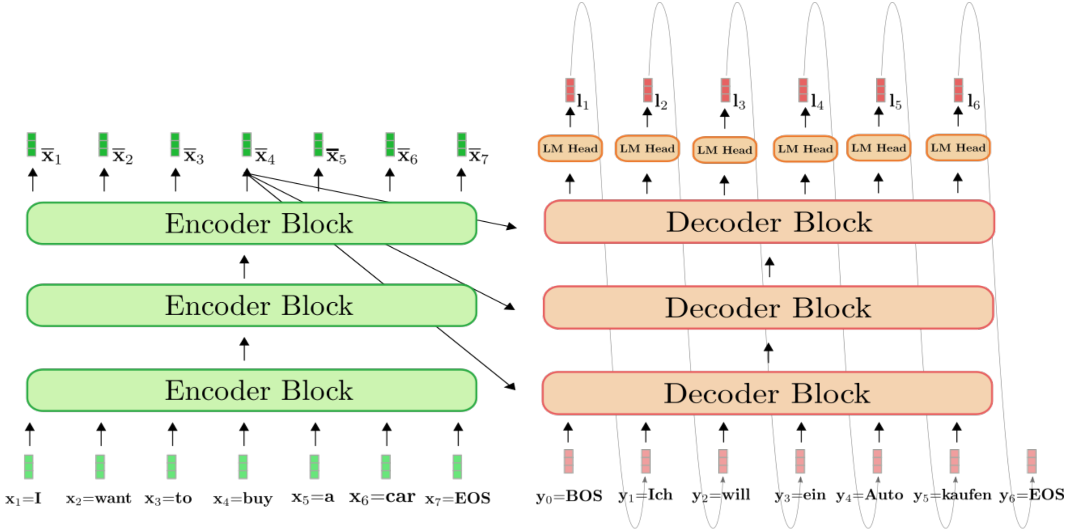
Training: Teacher forcing



Predictions



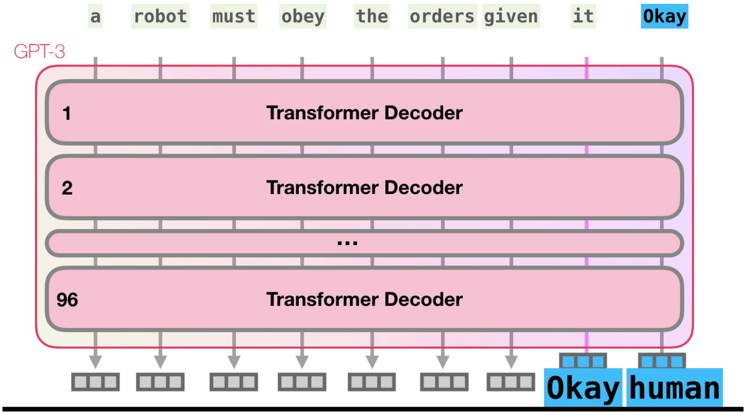
Predictions



Transformer-based Language Models

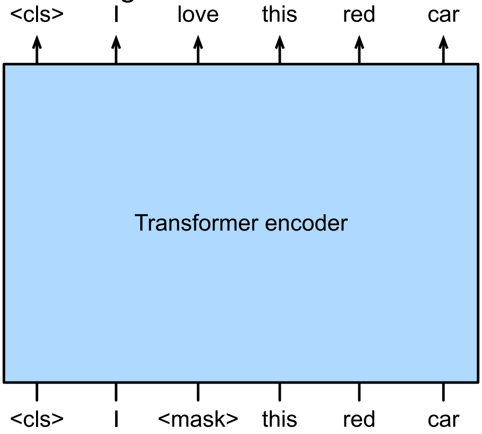
GPT (Generative Pretrained Transformer)

Invented by researchers at OpenAI
96 stacks of Transformer Decoder

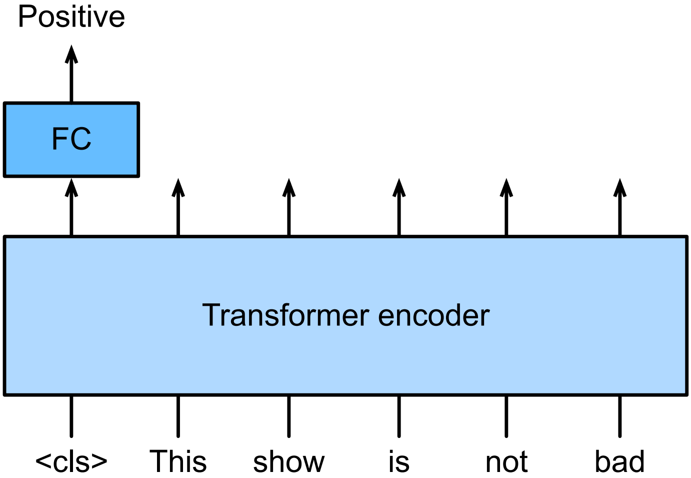


BERT (Bidirectional Encoder Representations from Transformers)

Invented by researchers at Google AI



Fine-tuning BERT



ChatGPT

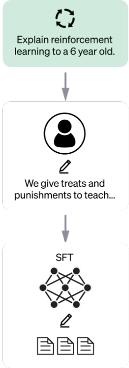
Step 1

Collect demonstration data and train a supervised policy.

A prompt is sample from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



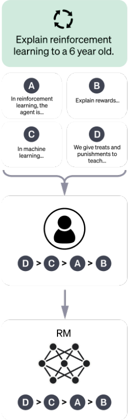
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

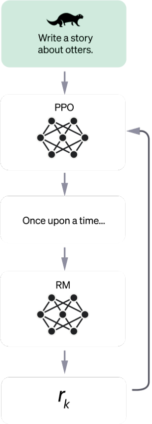
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



References

- Jay Alammar. The Illustrated Transformer.
  `https://jalammar.github.io/illustrated-transformer`
- Bala Priya. Build Better Deep Learning Models with Batch and Layer Normalization.
  `https://www.pinecone.io/learn/batch-layer-normalization`
- Sebastian Raschka. Understanding and Coding Self-Attention, Multi-Head Attention, Cross-Attention, and Causal-Attention in LLMs.
  `https://magazine.sebastianraschka.com/p/understanding-and-coding-self-attention`
- Patrick von Platen. Transformers-based Encoder-Decoder Models.
  `https://huggingface.co/blog/encoder-decoder`