Deep Learning for Vision & Language

Transformers I: Introduction

Today

- Sequence-to-sequence (RNNs) for Machine Translation
- Learning to Align and Translate with Soft Attention
- Image Captioning (CNNs + RNNs): Show and Tell
- Image Captioning (CNNs + RNNs + Attention): Show Attend and Tell
- Attention is All you Need!
- Encoder Transformers: BERT
- Decoder Transformers: GPT-2 – maybe next class

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Let's take a look at one of the first papers introducing this idea.

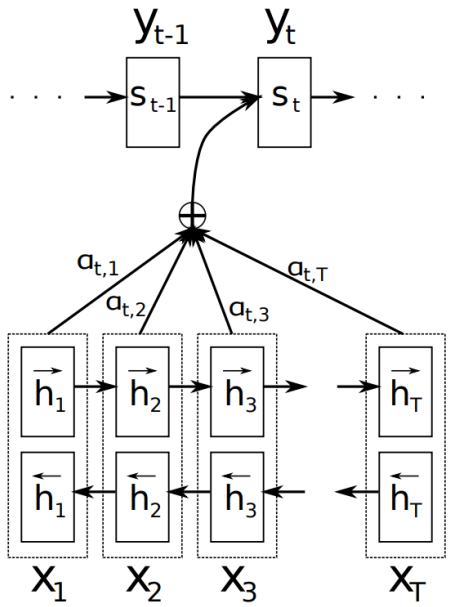


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

The Embedding Layer nn.Embedding

EMBEDDING

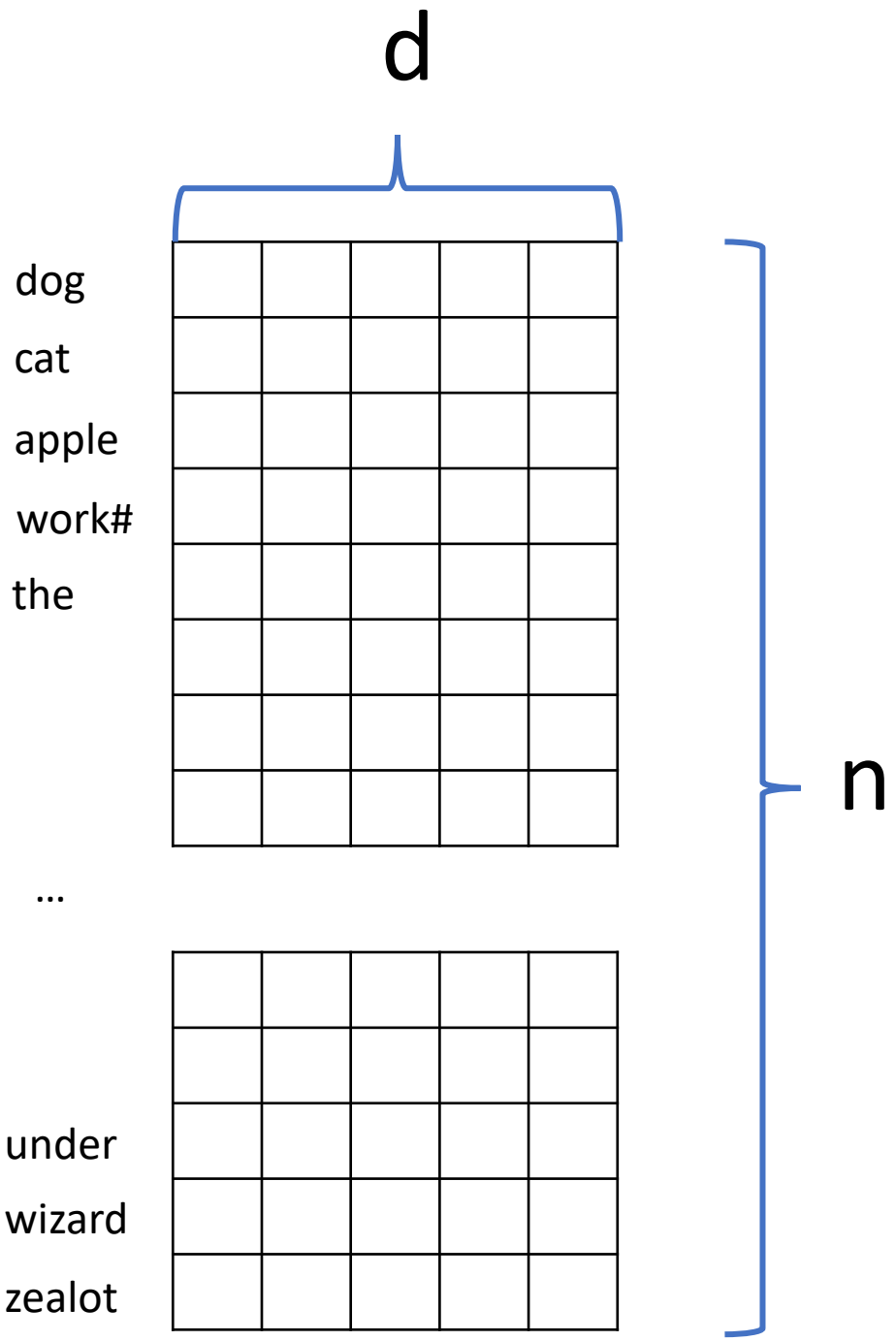
CLASS torch.nn.Embedding(*num_embeddings*, *embedding_dim*, *padding_idx=None*, *max_norm=None*, *norm_type=2.0*, *scale_grad_by_freq=False*, *sparse=False*, *_weight=None*, *device=None*, *dtype=None*) [SOURCE]

A simple lookup table that stores embeddings of a fixed dictionary and size.

This module is often used to store word embeddings and retrieve them using indices. The input to the module is a list of indices, and the output is the corresponding word embeddings.

Parameters:

- **num_embeddings** (*int*) – size of the dictionary of embeddings
- **embedding_dim** (*int*) – the size of each embedding vector
- **padding_idx** (*int*, *optional*) – If specified, the entries at `padding_idx` do not contribute to the gradient; therefore, the embedding vector at `padding_idx` is not updated during training, i.e. it remains as a fixed "pad". For a newly constructed Embedding, the embedding vector at `padding_idx` will default to all zeros, but can be updated to another value to be used as the padding vector.



The Embedding Layer nn.Embedding

nn.Embedding(n, d)

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Let's take a look at one of the first papers introducing this idea.

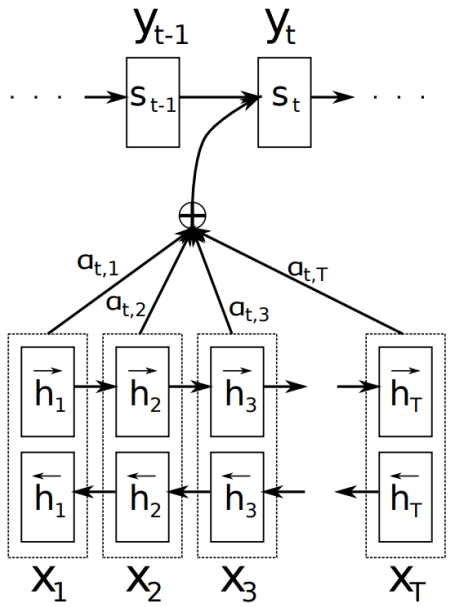


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

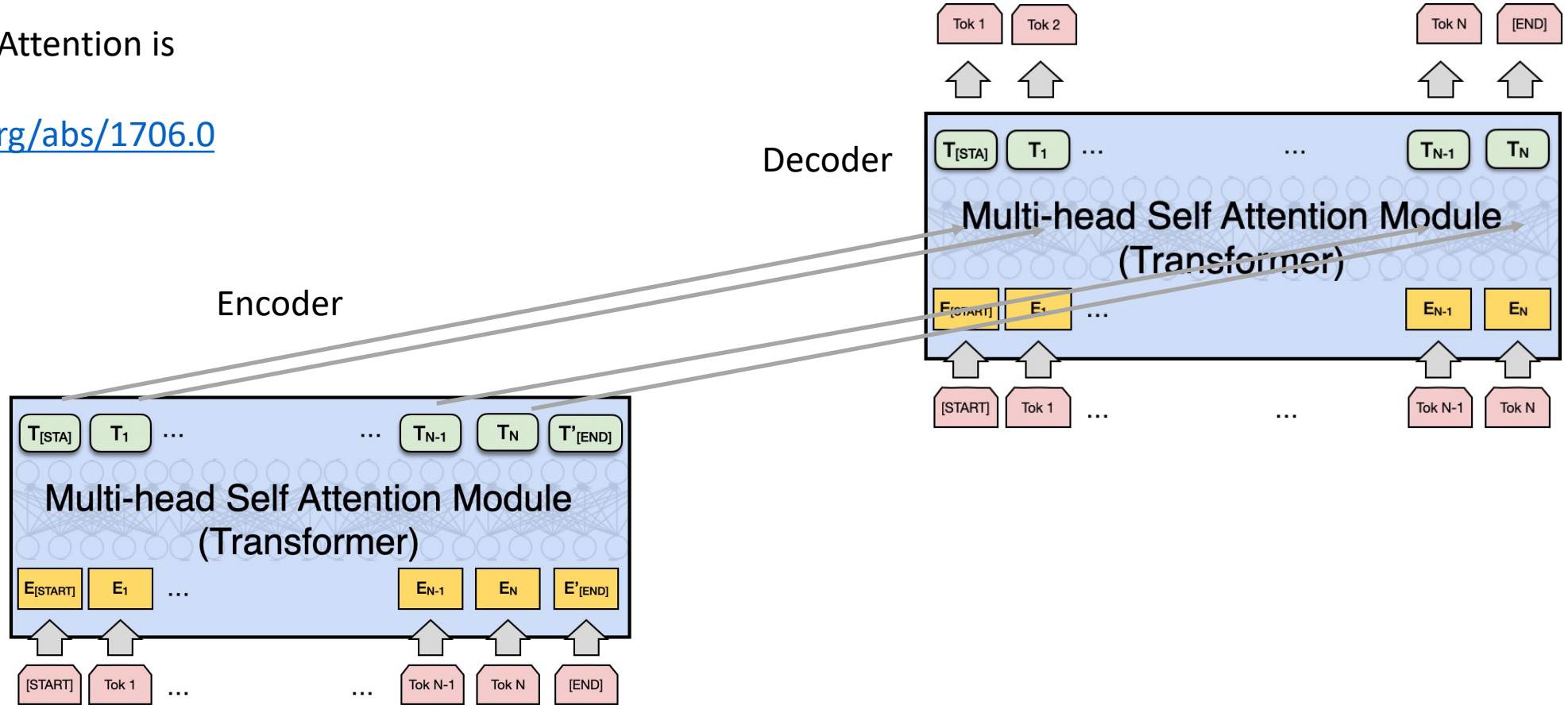
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

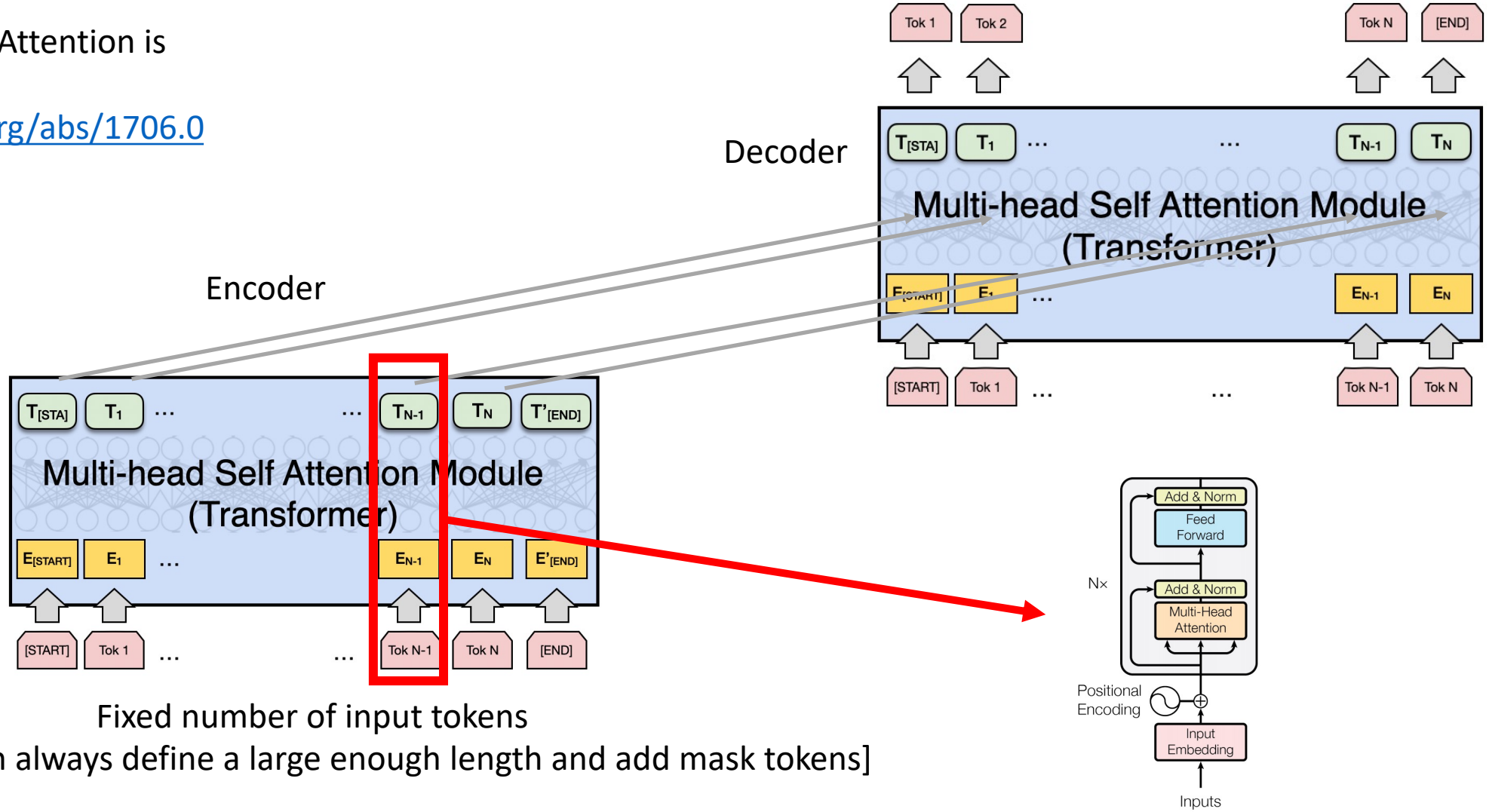


Fixed number of input tokens

[but hey! we can always define a large enough length and add mask tokens]

Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

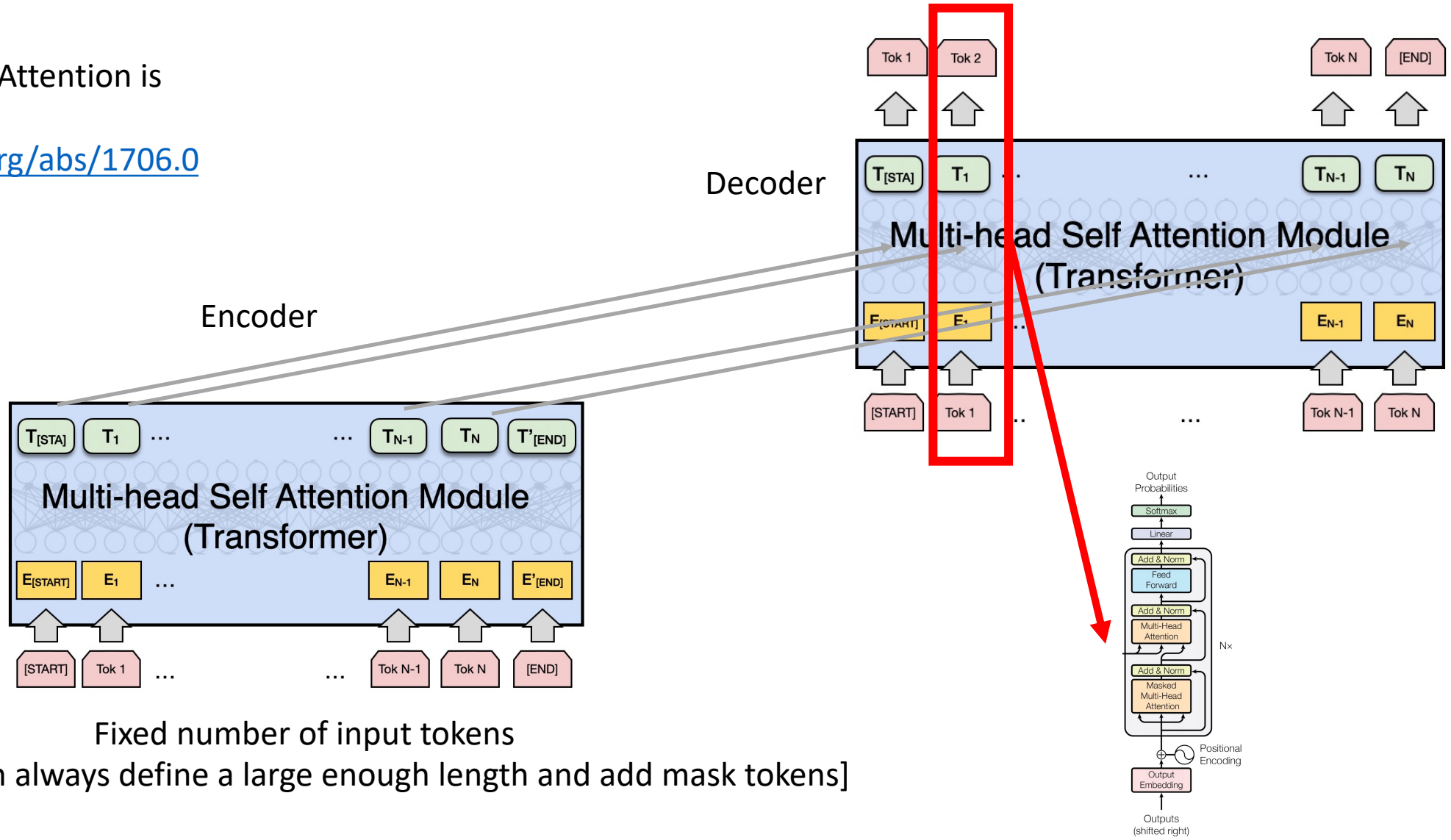


Fixed number of input tokens

[but hey! we can always define a large enough length and add mask tokens]

Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

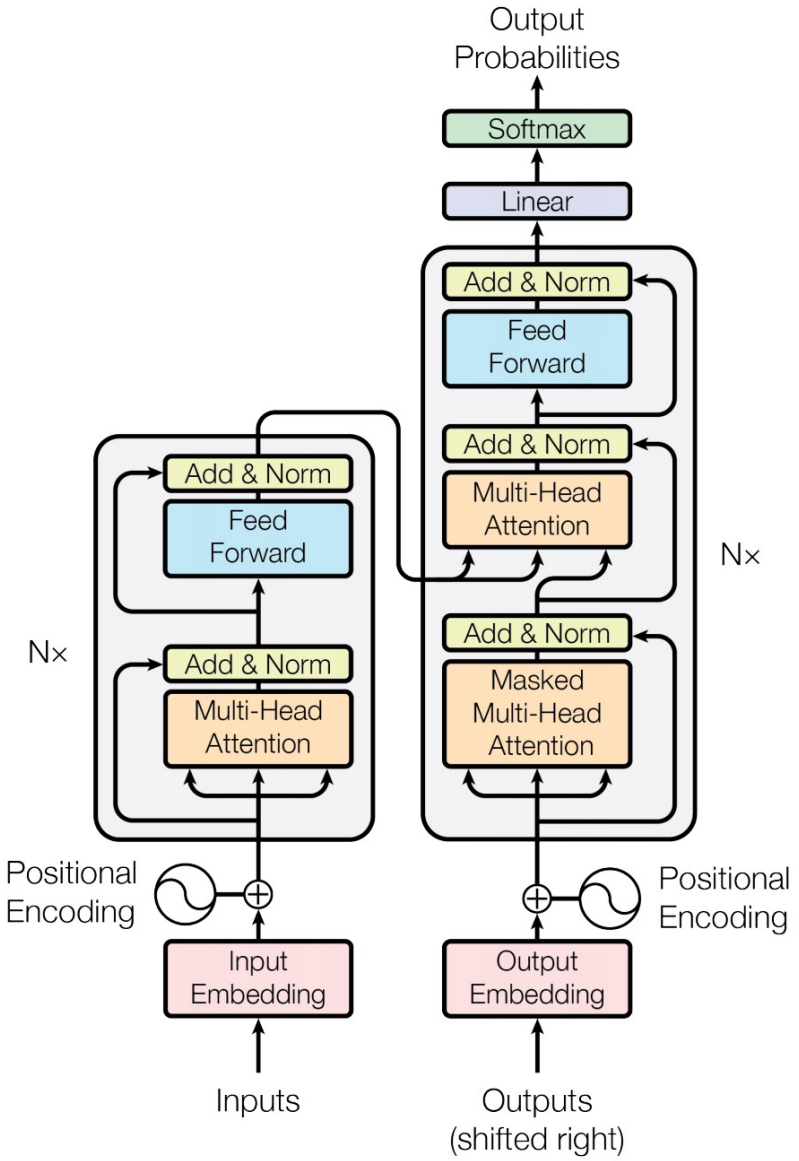


Fixed number of input tokens

[but hey! we can always define a large enough length and add mask tokens]

We can also draw this as in the paper:

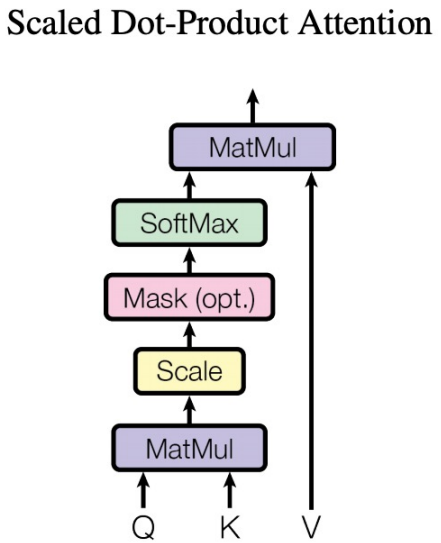
Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762



Regular Attention: + Scaling factor

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

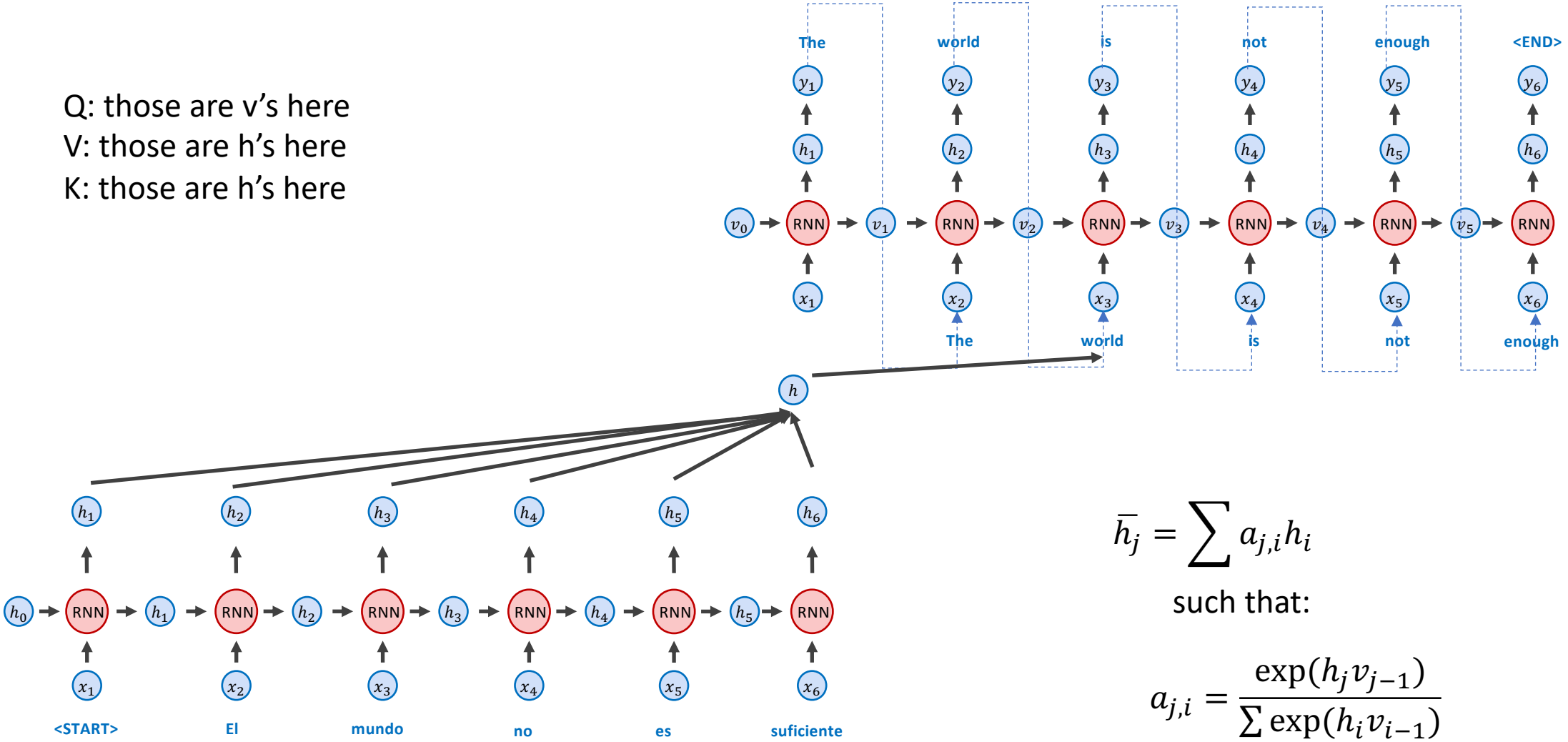
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



This is not unlike what we already used before

Only showing the third time step encoder-decoder connection

Q: those are v's here
V: those are h's here
K: those are h's here



$$\bar{h}_j = \sum a_{j,i} h_i$$

such that:

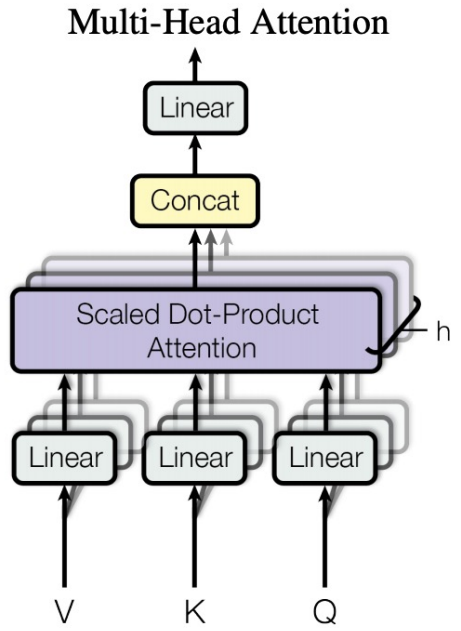
$$a_{j,i} = \frac{\exp(h_j v_{j-1})}{\sum \exp(h_i v_{i-1})}$$

Multi-head Attention: Do not settle for just one set of attention weights.

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

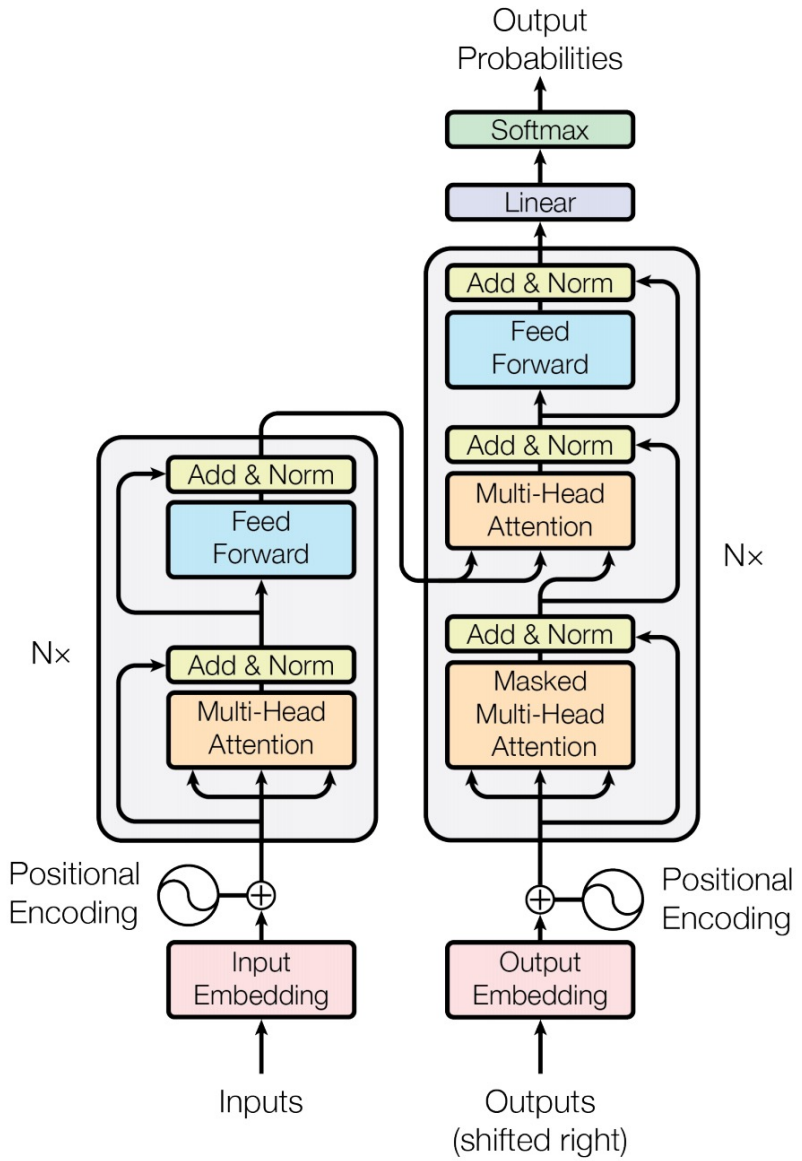
Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.



We can lose track of position since we are aggregating across all locations

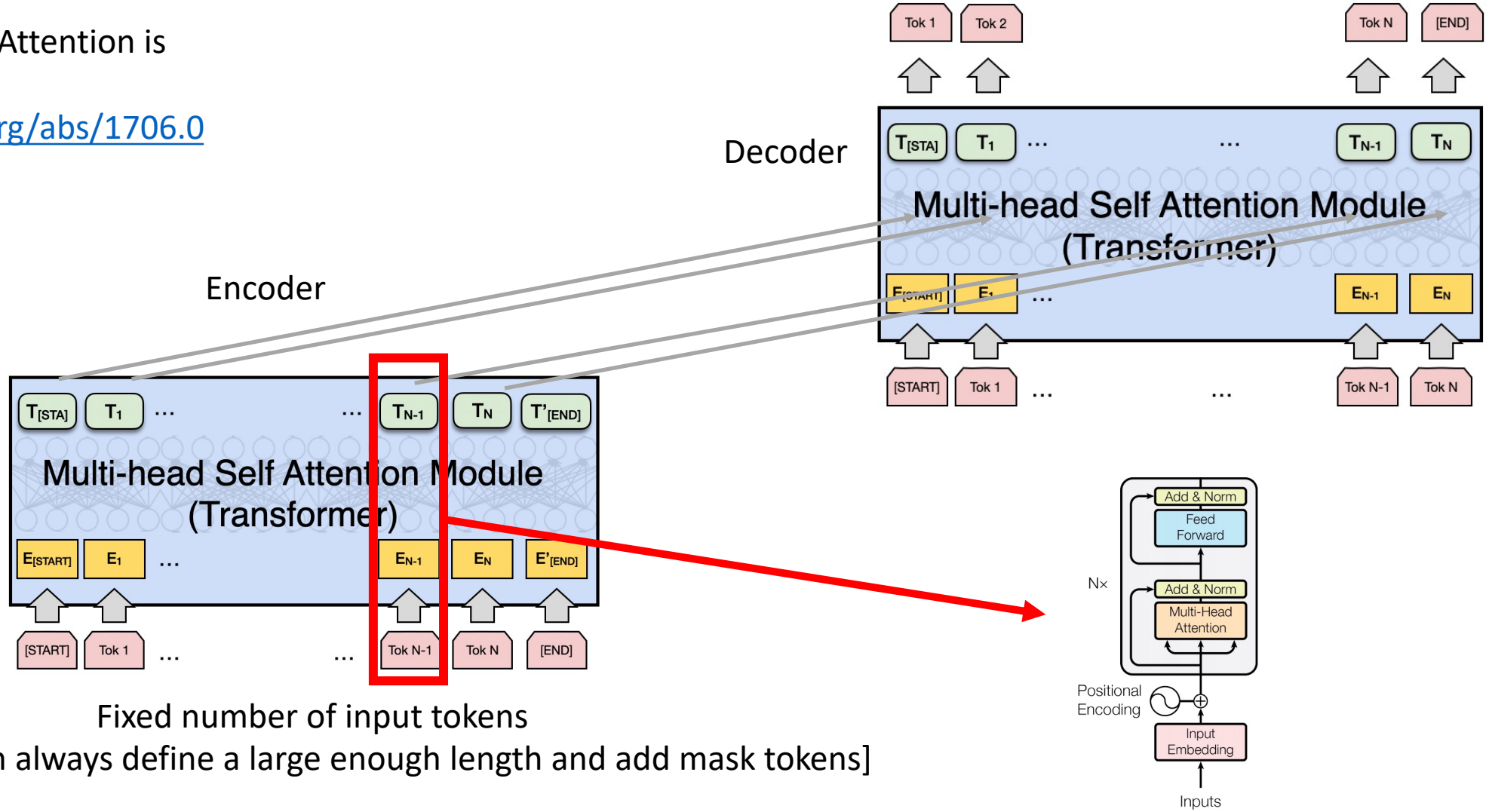
Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

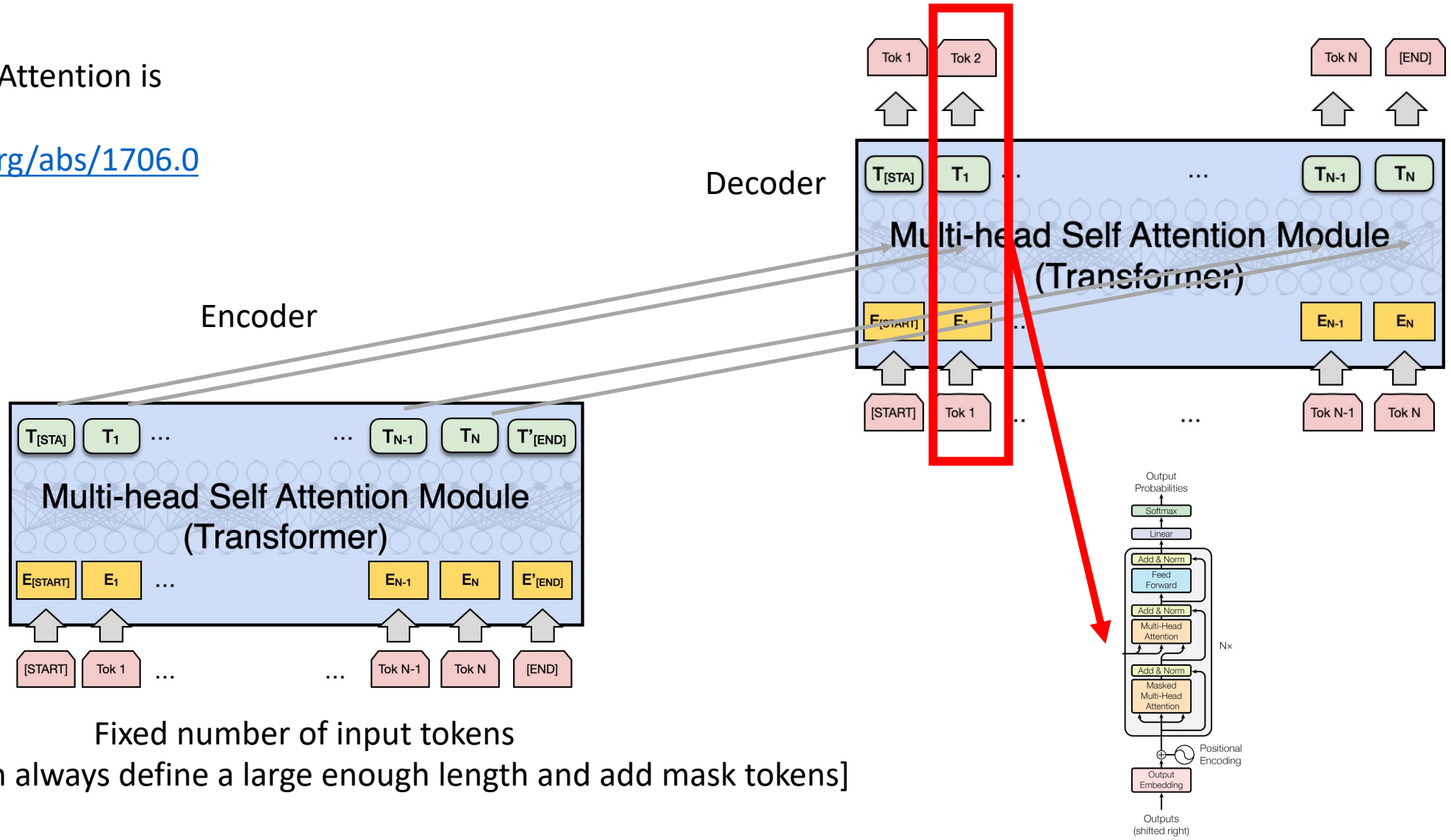


Fixed number of input tokens

[but hey! we can always define a large enough length and add mask tokens]

Attention is All you Need (no RNNs)

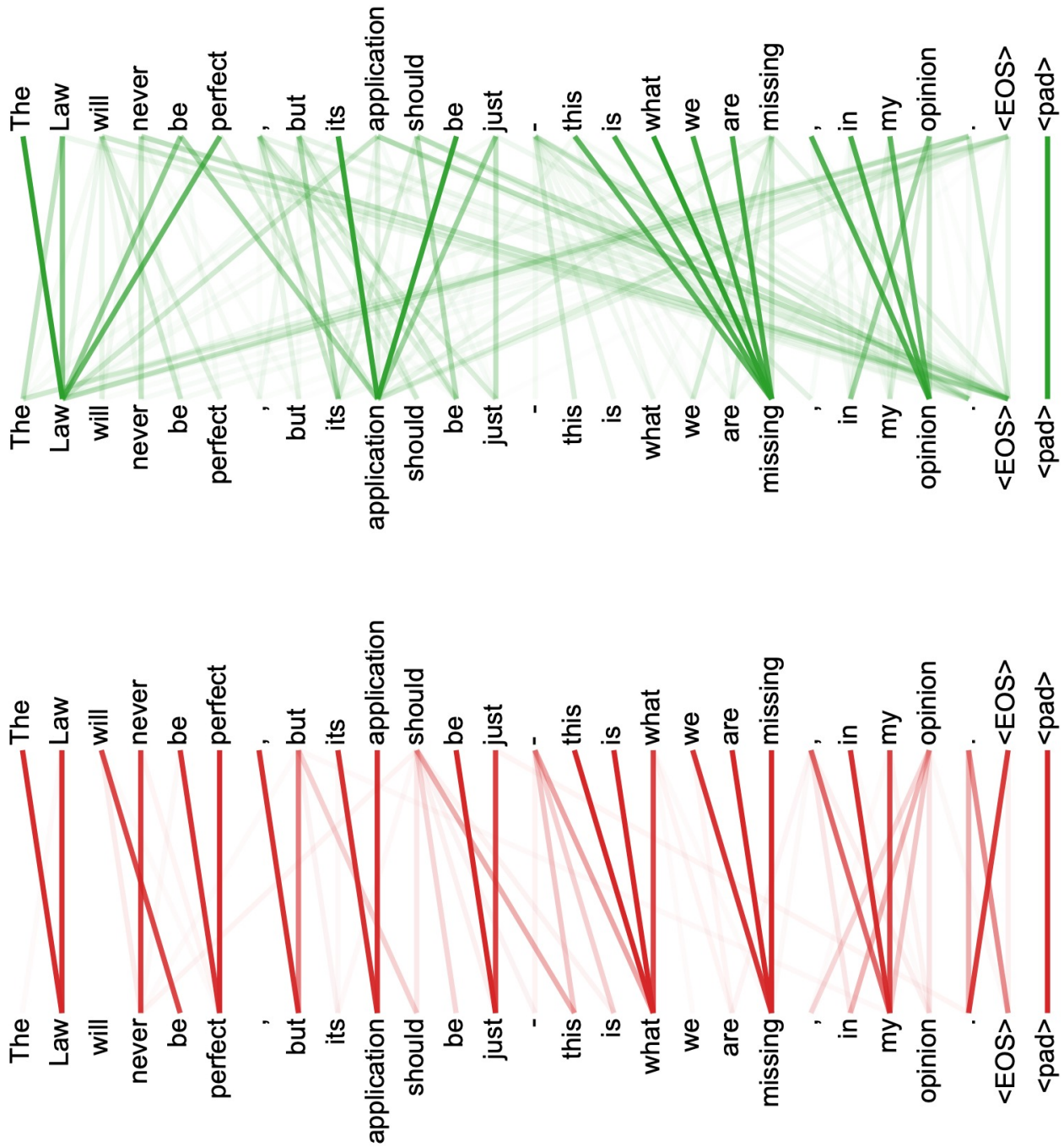
Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762



Fixed number of input tokens

[but hey! we can always define a large enough length and add mask tokens]

Multi-headed attention weights are harder to interpret obviously

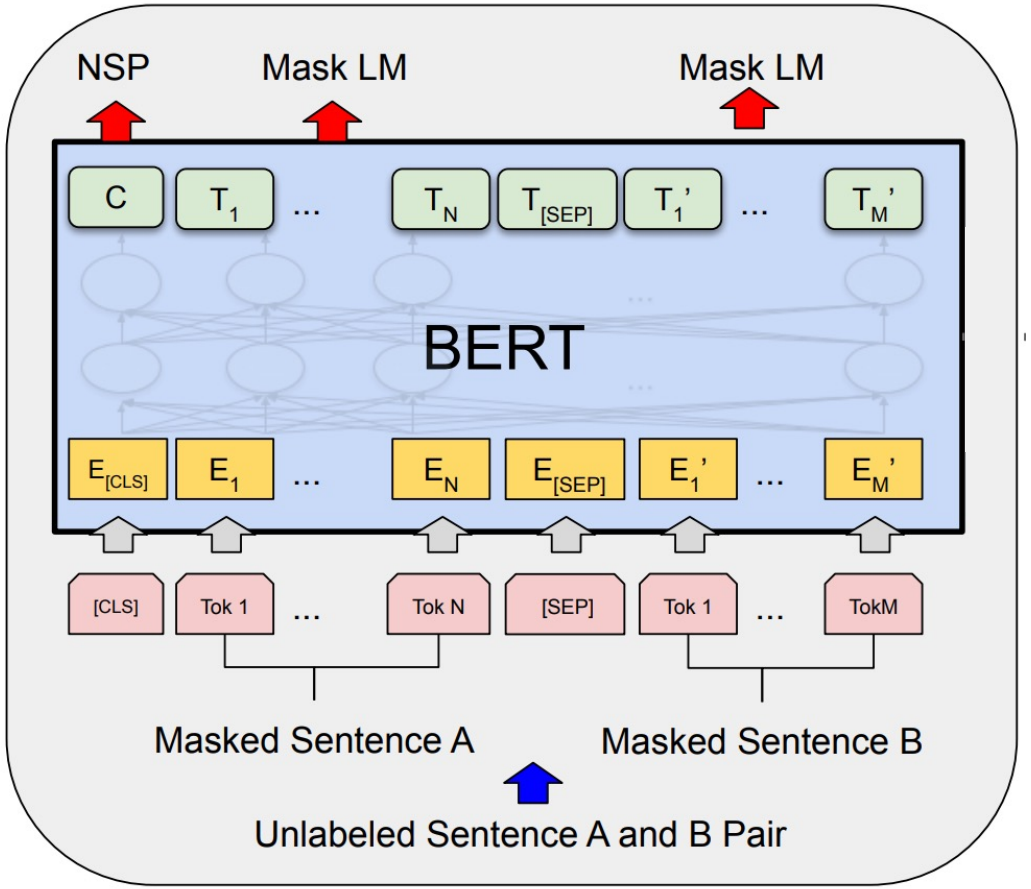


The BERT Encoder Model

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding . https://arxiv.org/abs/1810.04805

Important things to know

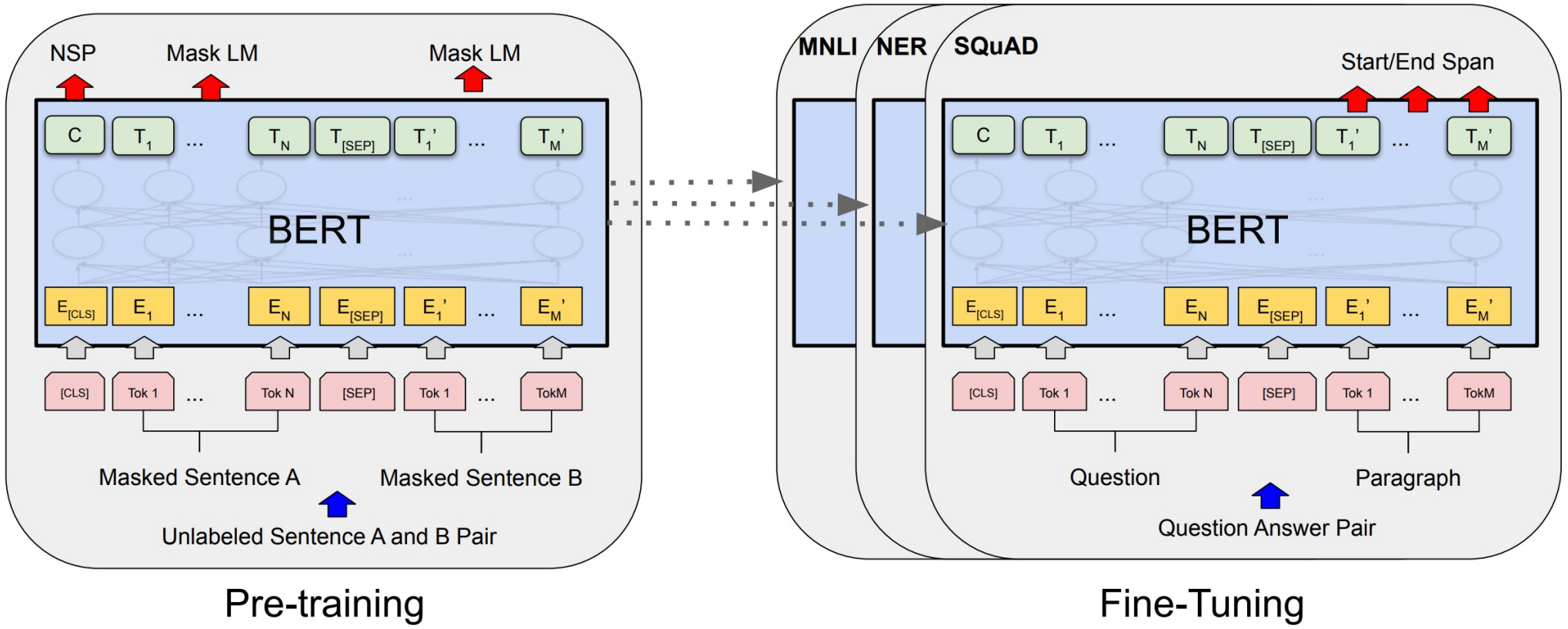
- No decoder
- Train the model to fill-in-the-blank by masking some of the input tokens and trying to recover the full sentence.
- The input is not one sentence but two sentences separated by a [SEP] token.
- Also try to predict whether these two input sentences are consecutive or not.



Pre-training

The BERT Encoder Model

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding . https://arxiv.org/abs/1810.04805



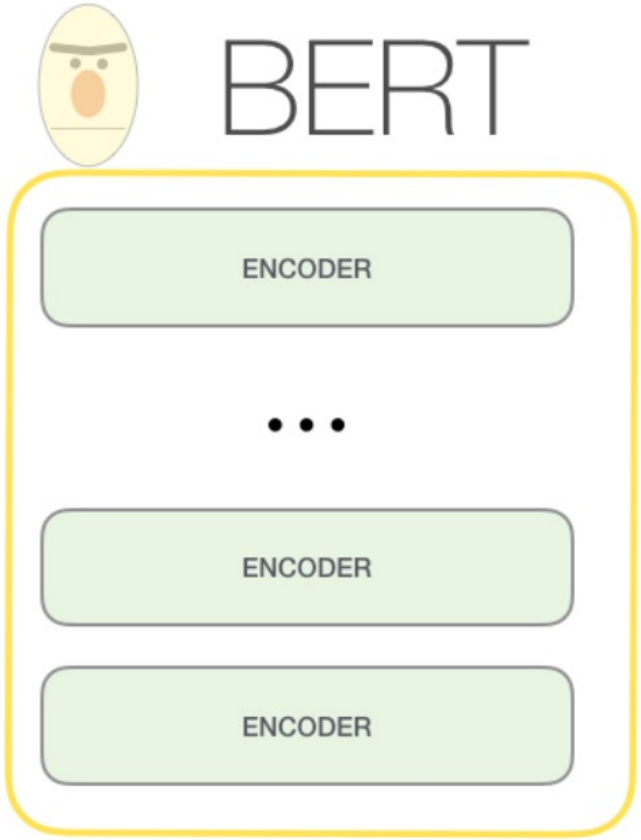
The GPT-2 Model

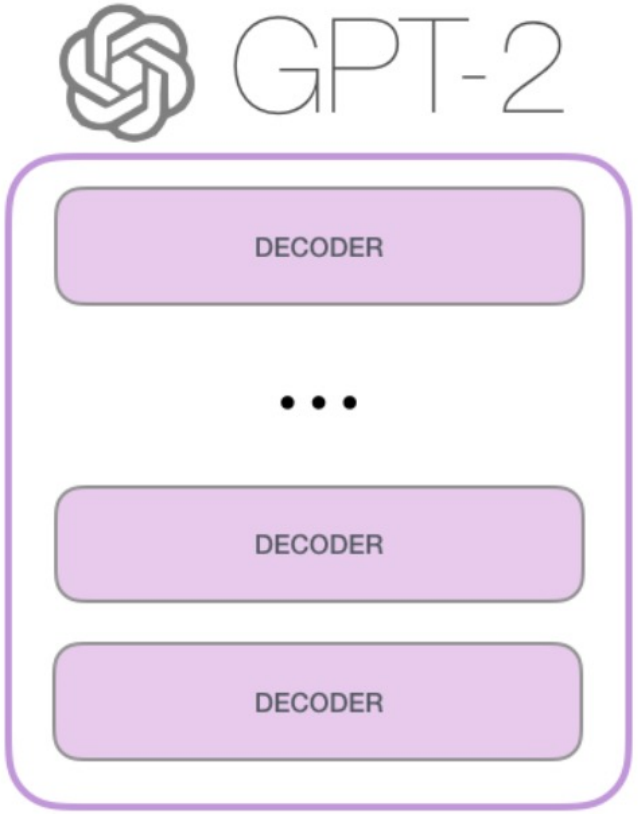
Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

https://openai.com/blog/better-language-models/

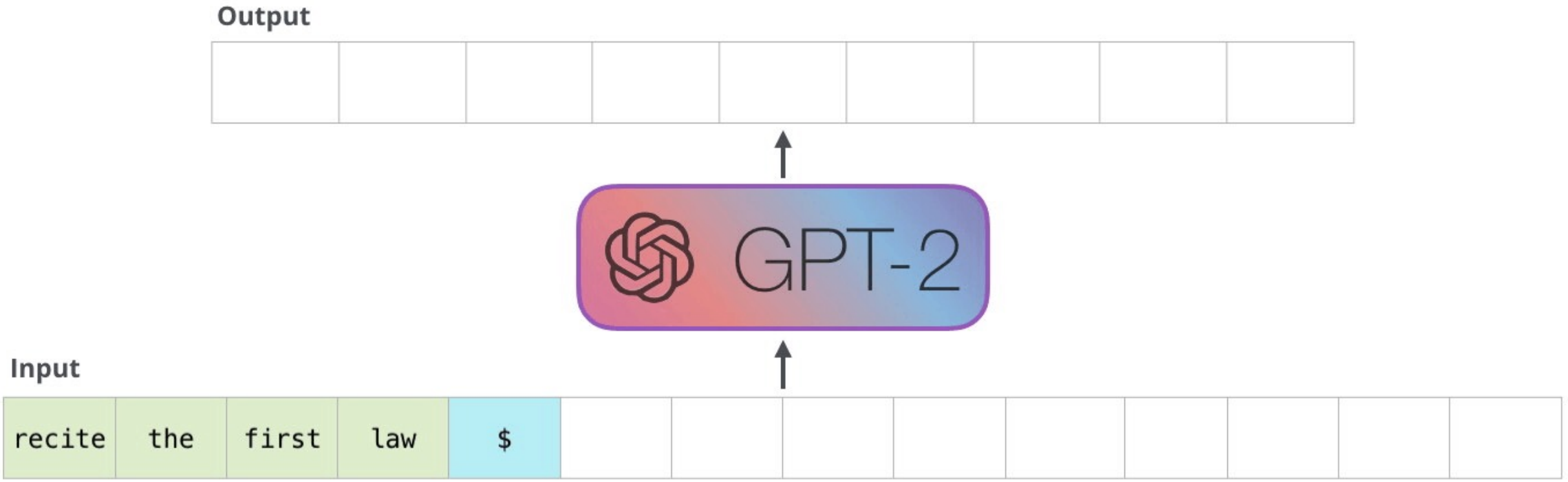
The GPT-2 Model





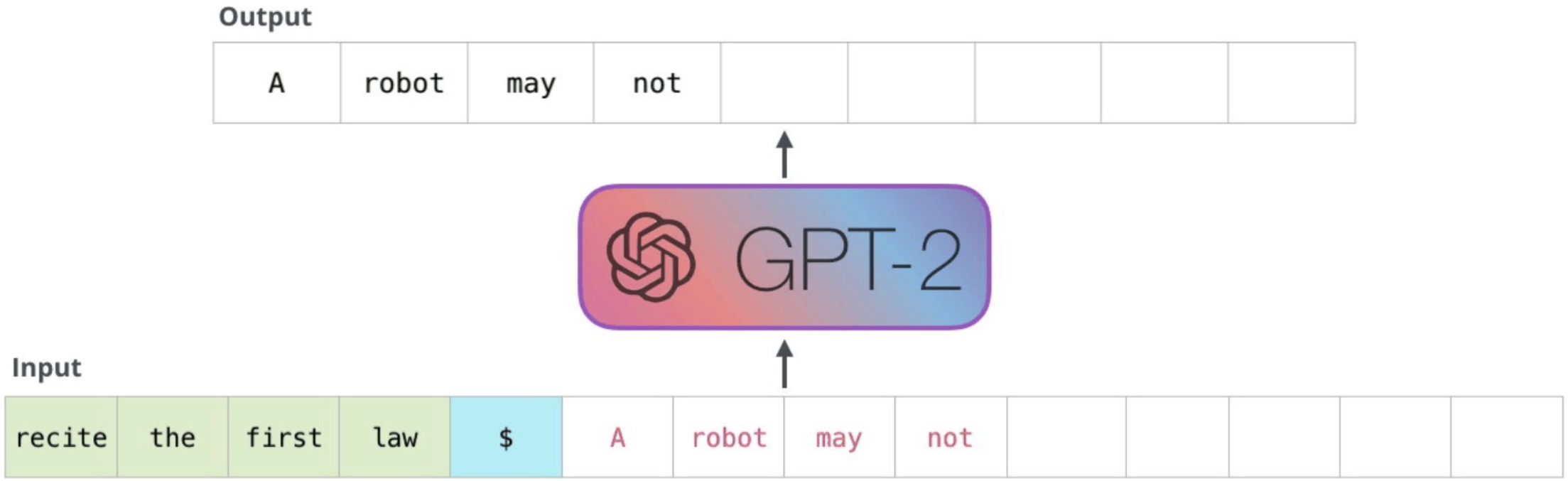
https://jalammar.github.io/illustrated-gpt2/

The GPT-2 Model



https://jalammar.github.io/illustrated-gpt2/

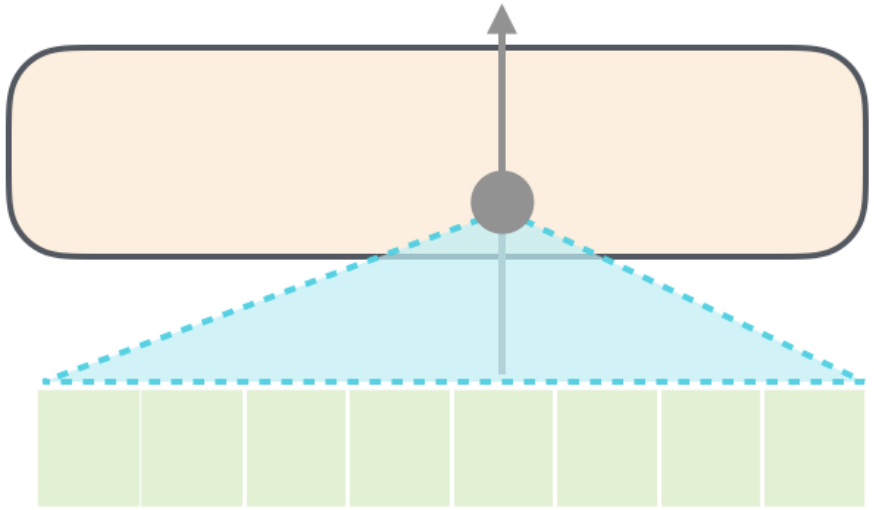
The GPT-2 Model



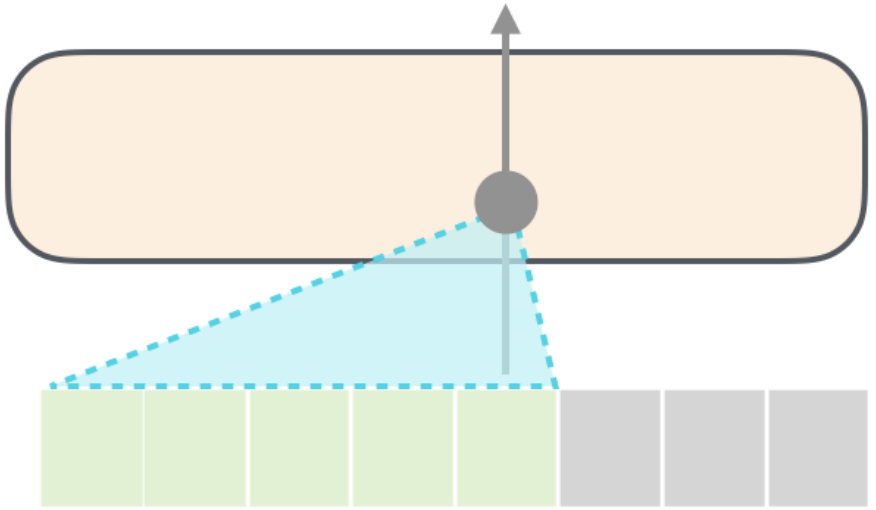
https://jalammar.github.io/illustrated-gpt2/

The GPT-2 Model

BERT



GPT



https://jalammar.github.io/illustrated-gpt2/

GPT-1 vs GPT-2 vs GPT-3

	GPT-1	GPT-2	GPT-3
Parameters	117 Million	1.5 Billion	175 Billion
Decoder Layers	12	48	96
Context Token Size	512	1024	2048
Hidden Layer	768	1600	12288
Batch Size	64	512	3.2M

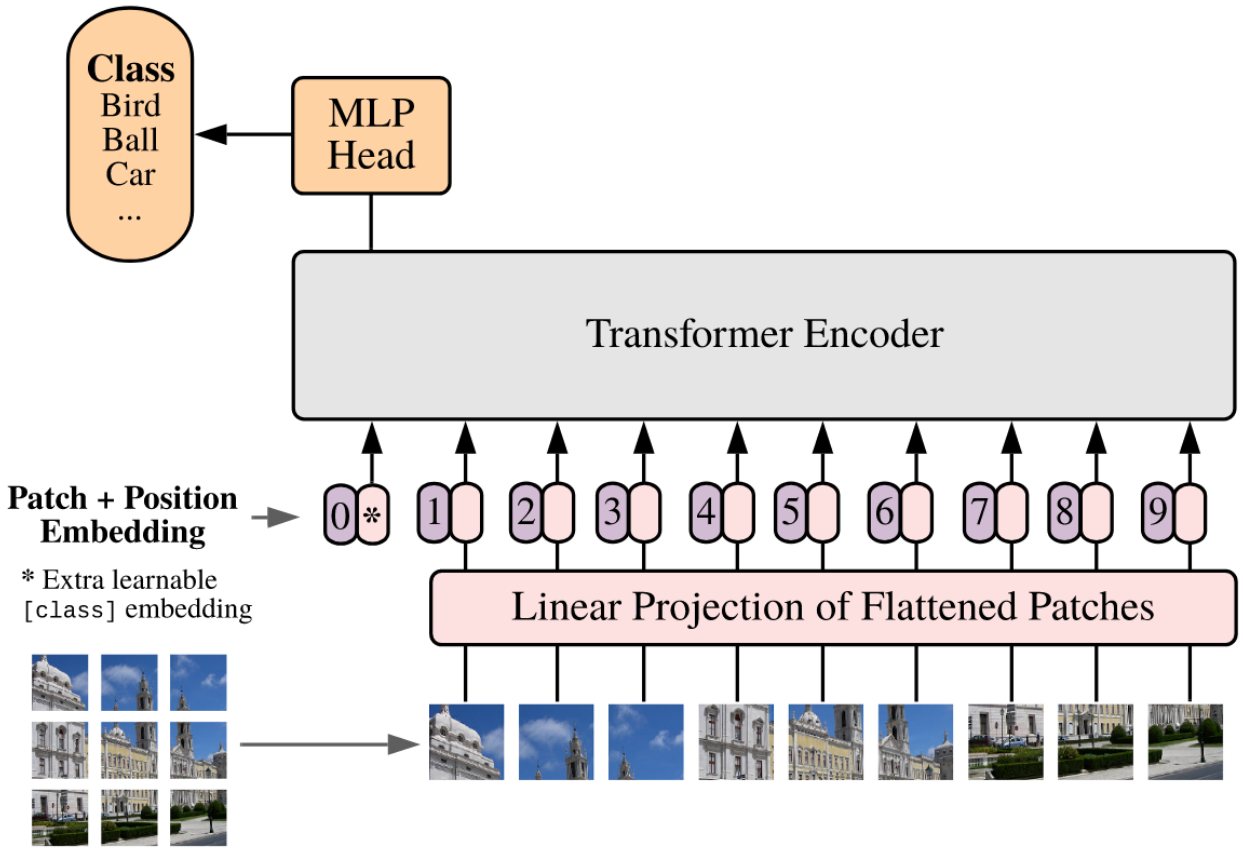
https://360digitmg.com/blog/types-of-gpt-in-artificial-intelligence

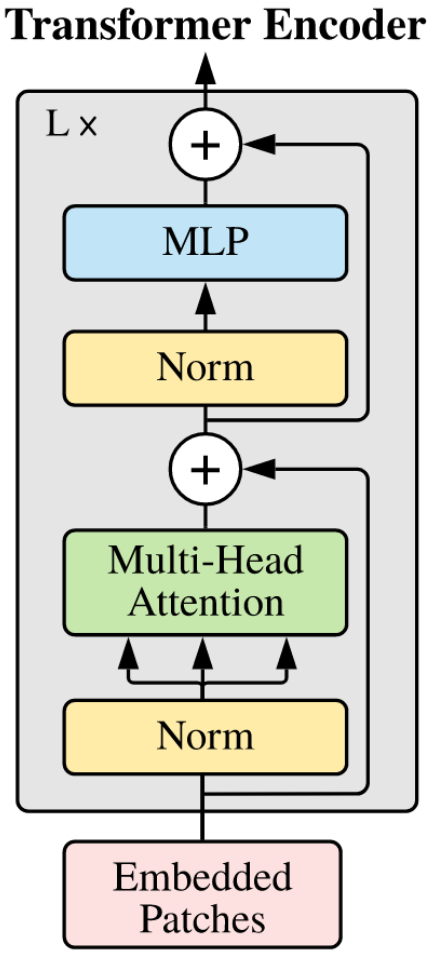
GPT-3

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

https://arxiv.org/abs/2005.14165

Vision Transformers

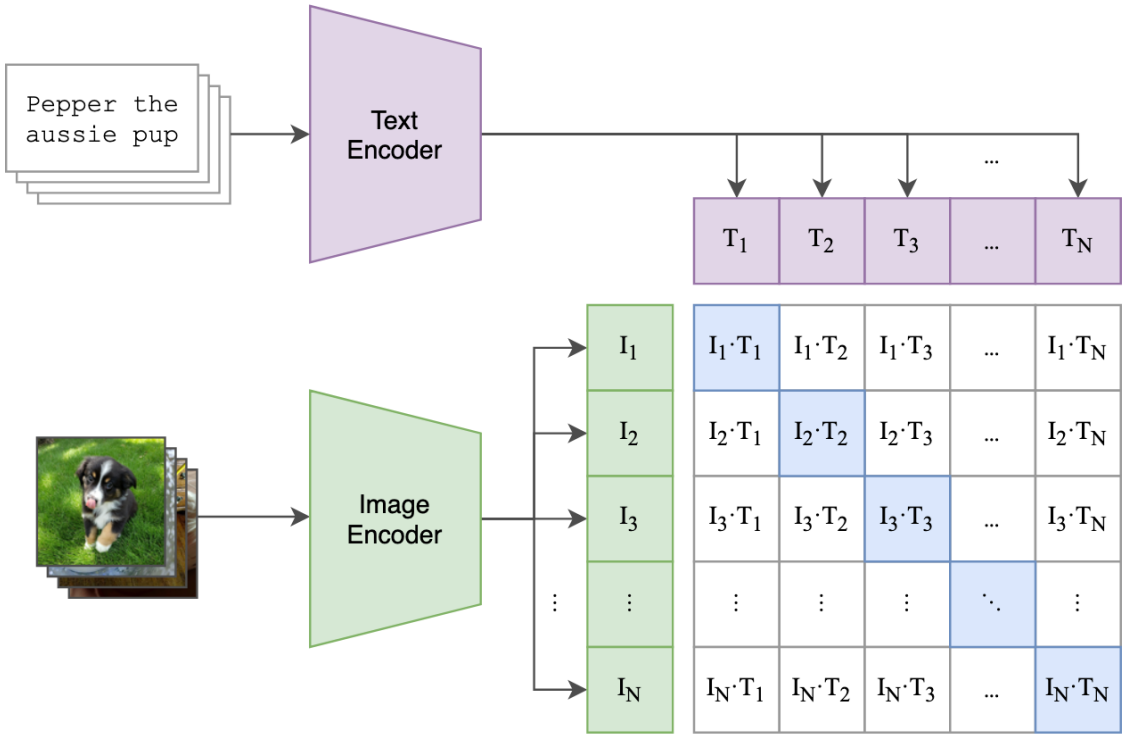




https://arxiv.org/abs/2010.11929

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby

The CLIP Model



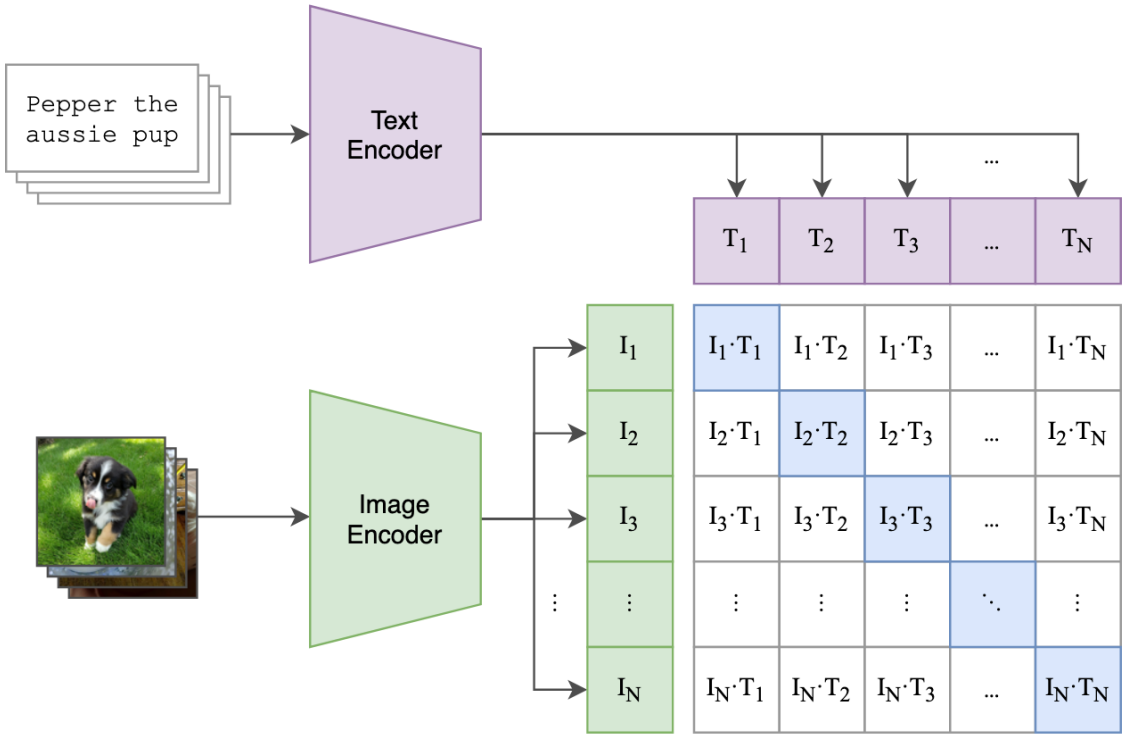
$$L = \sum_{k} \ell(I_k T_k)$$

$$\ell(I_k T_k) = -\log\left(\frac{\exp(sim(I_k, T_k))}{\sum_{t=1}^{2N} 1[k \neq i]\exp(sim(I_k, T_t))}\right)$$

https://arxiv.org/abs/2103.00020

Learning Transferable Visual Models From Natural Language Supervision
Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever

The CLIP Model



$$L = \sum_k \ell_1(I_k T_k) + \ell_2(I_k T_k)$$

$$\ell_1(I_k T_k) = -\log\left(\frac{\exp(sim(I_k, T_k))}{\sum_{t=1}^{2N} 1[k \neq i]\exp(sim(I_k, T_t))}\right)$$

$$\ell_2(I_k T_k) = -\log\left(\frac{\exp(sim(I_k, T_k))}{\sum_{t=1}^{2N} 1[k \neq i]\exp(sim(I_t, T_k))}\right)$$

https://arxiv.org/abs/2103.00020

Learning Transferable Visual Models From Natural Language Supervision
Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever

Questions?