Transformers

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

Transformer

- Issues with RNNs:
  - Cannot parallelize within a sequence due to recurrence.
  - Problem of vanishing/exploding gradients.
  - Requires large number of training steps.
- Vaswani et. al. 2017 proposed a new architecture based on the attention mechanism

Attention Is All You Need

- The architecture is known as Transformer.
- Employs a self-attention mechanism to model relationships between all words in a sentence

Paper: Vaswani *et. al.* Attention Is All You Need, NIPS 2017

Self-attention

- In typical encoder-decoder attention layers mechanisms, the queries come from the decoder layer, while the 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.
- Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.
- Suppose we have an encoder with self-attention layers.
  - In a self-attention layer, the keys, values and queries come from the output of the previous layer in the encoder.
- Self-attention layers in the decoder allow each position in the decoder to attend all positions in the decoder up to and including that position.

Self-attention

- In each step, the transformer applies a self-attention mechanism which directly models relationships between all words in a sentence, regardless of their respective position.

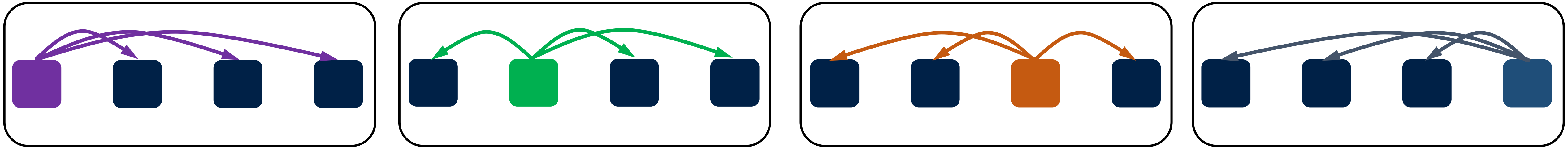
- Consider the following sentence:

We need a crane to lift the item

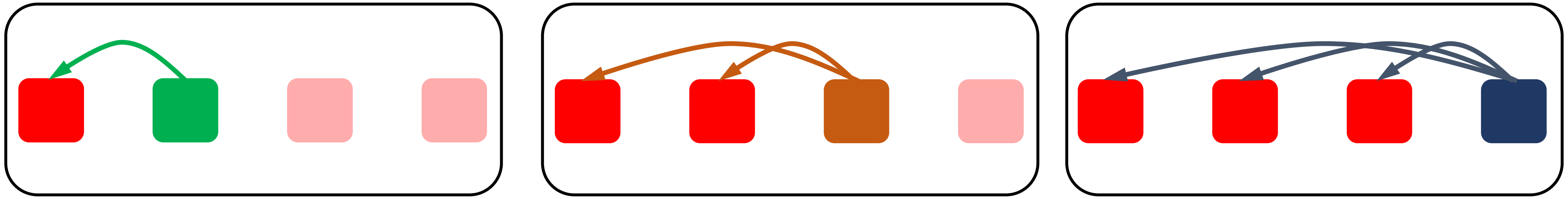
  - The model needs to determine that the word “crane” refers to the a large machine and not the bird.
  - The Transformer computes the next representation of each word by comparing it with every other word in the sentence.

Self-attention

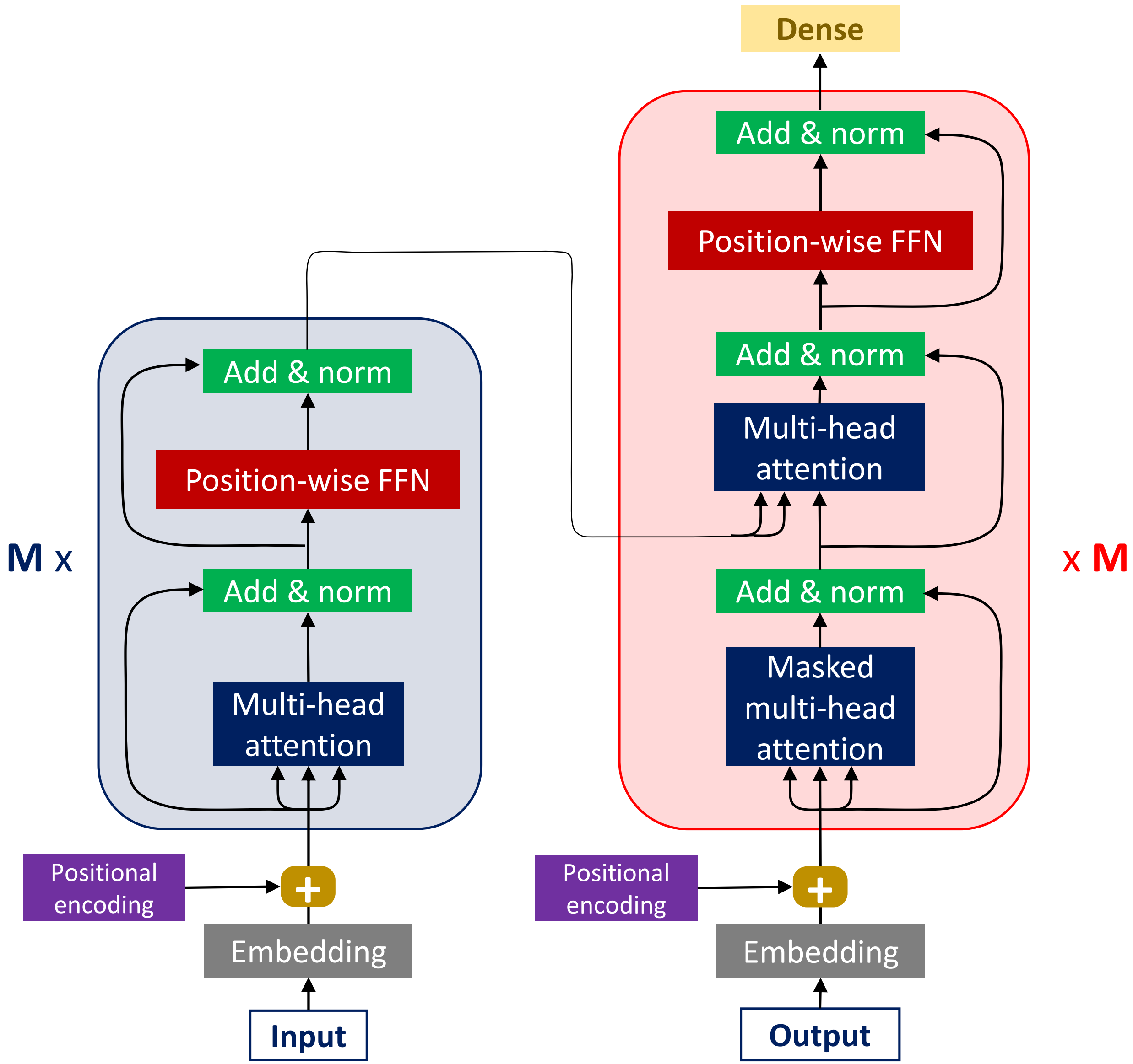
ENCODER SELF-ATTENTION



DECODER SELF-ATTENTION (MASKED)



Transformer Architecture

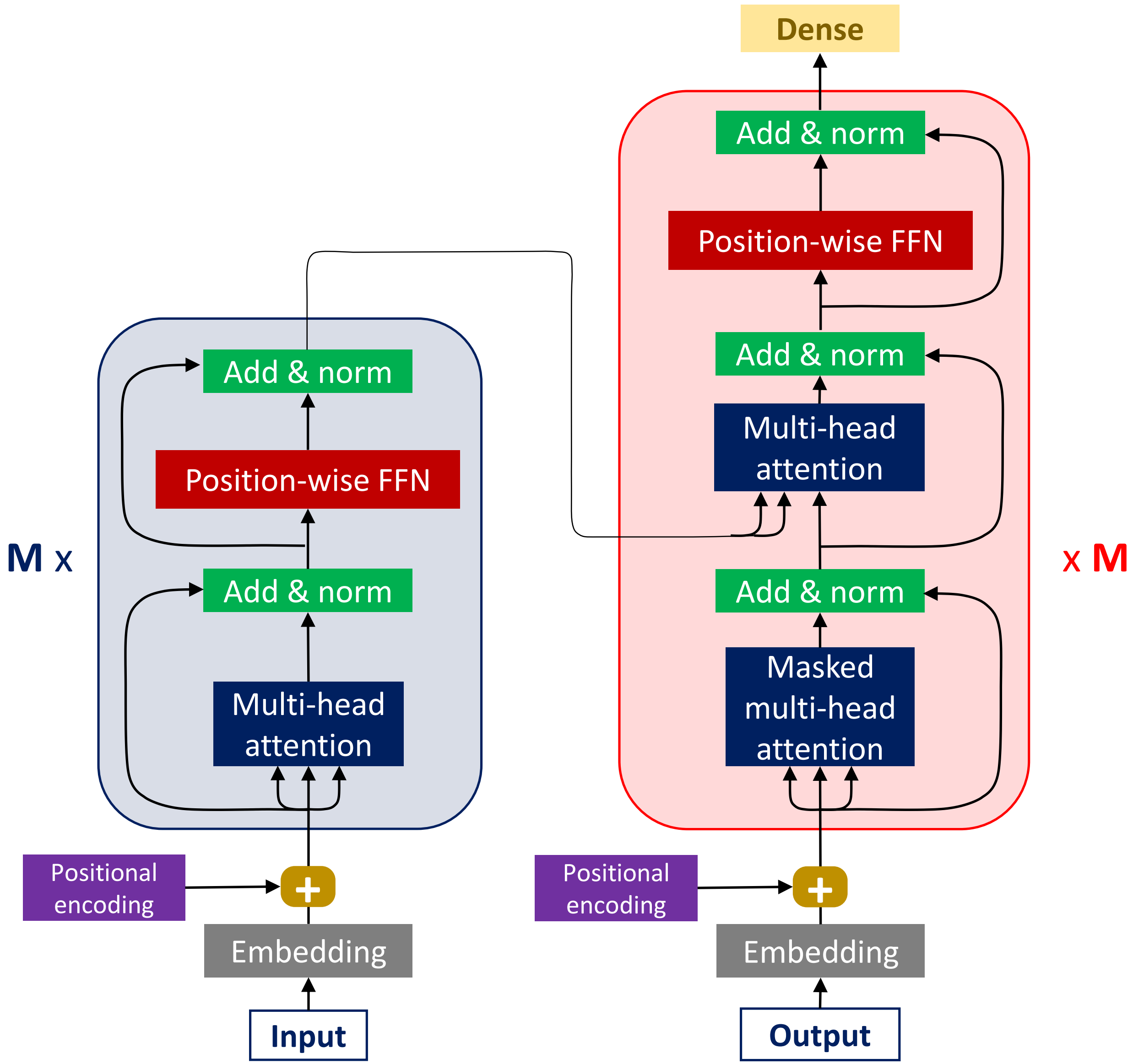


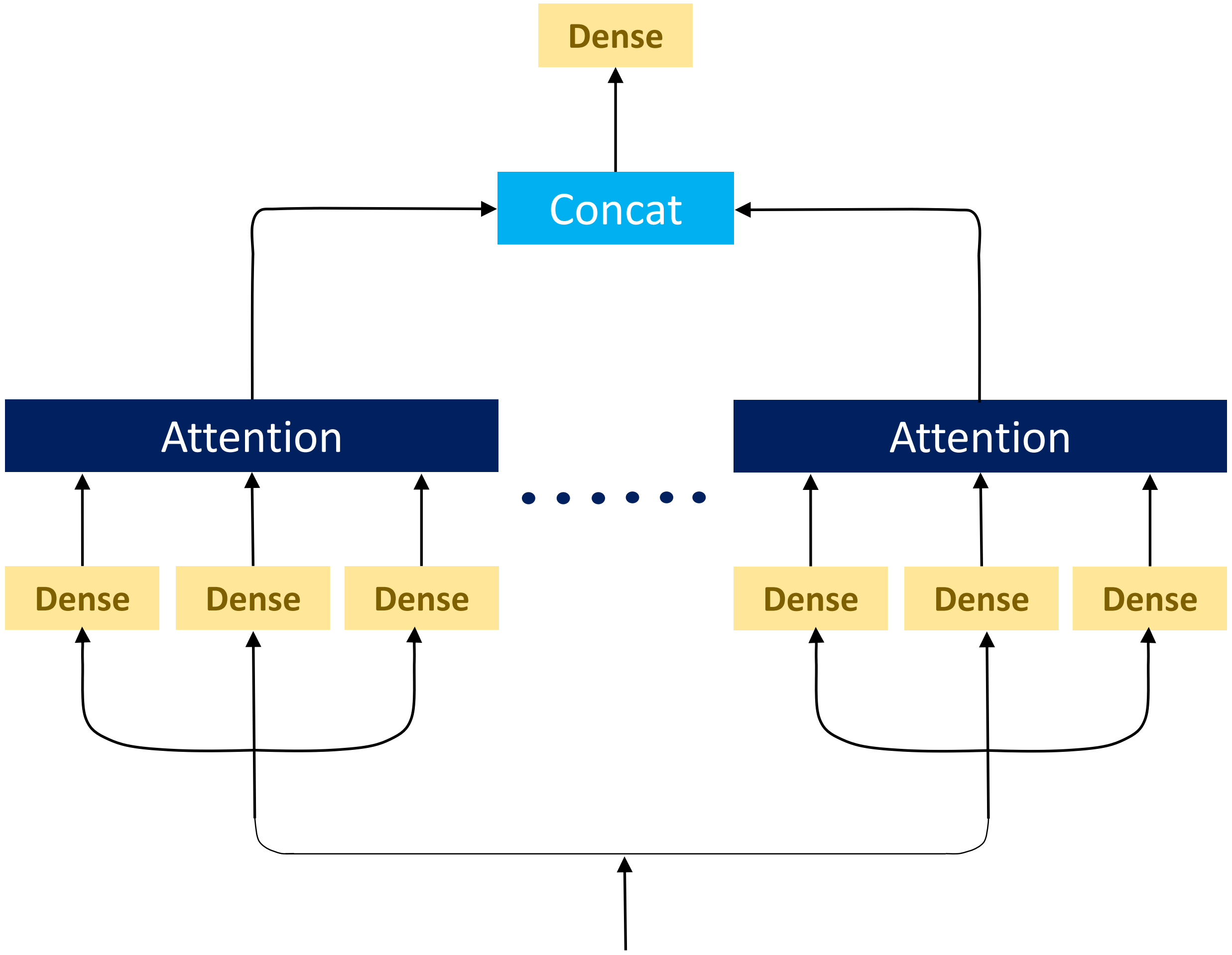
- The figure shows the basic unit of a transformer.
- Embeddings for both inputs and outputs.
- Encoder major components:
  - Self-attention mechanism
  - Feed-forward neural network
- Decoder major components:
  - Self-attention mechanism
  - Attention mechanism over encodings
  - Feed-forward neural network

Animation

Source: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

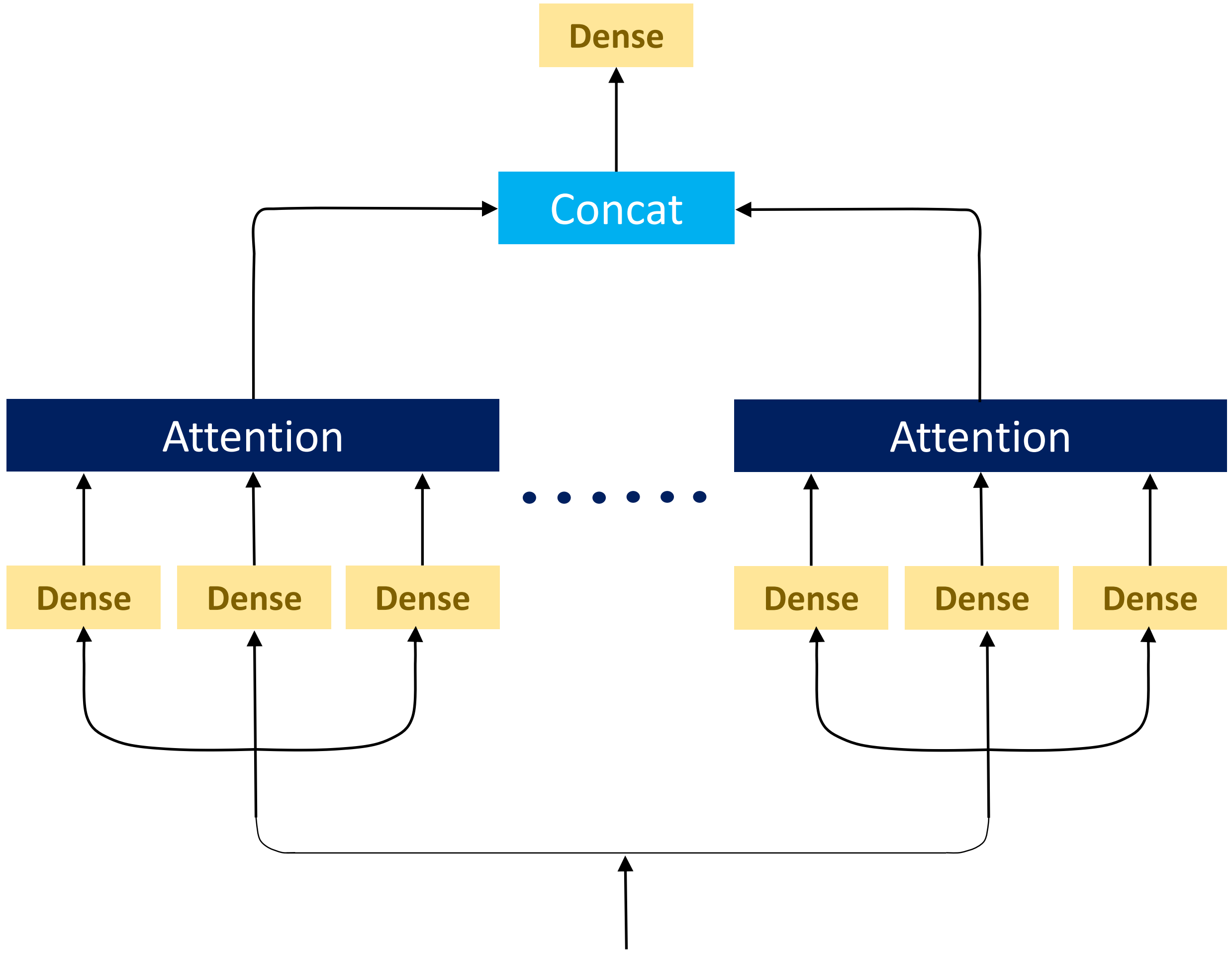
Multi-head attention





Multi-head attention

- In multi-head attention layer there are multiple number (say M) of parallel self-attention layers. Each of them is called a head.
  - In a self-attention layer the query, key and value vectors are the same (the input).
- The dense layers before attention layer project the queries, keys and values into different dimensions.
  - Suppose the query, key and value vectors are projected into vectors of length p_q, p_k and p_v respectively.
- Suppose the dimension of query is l_q, key is l_k and value is l_v.



Multi-head attention

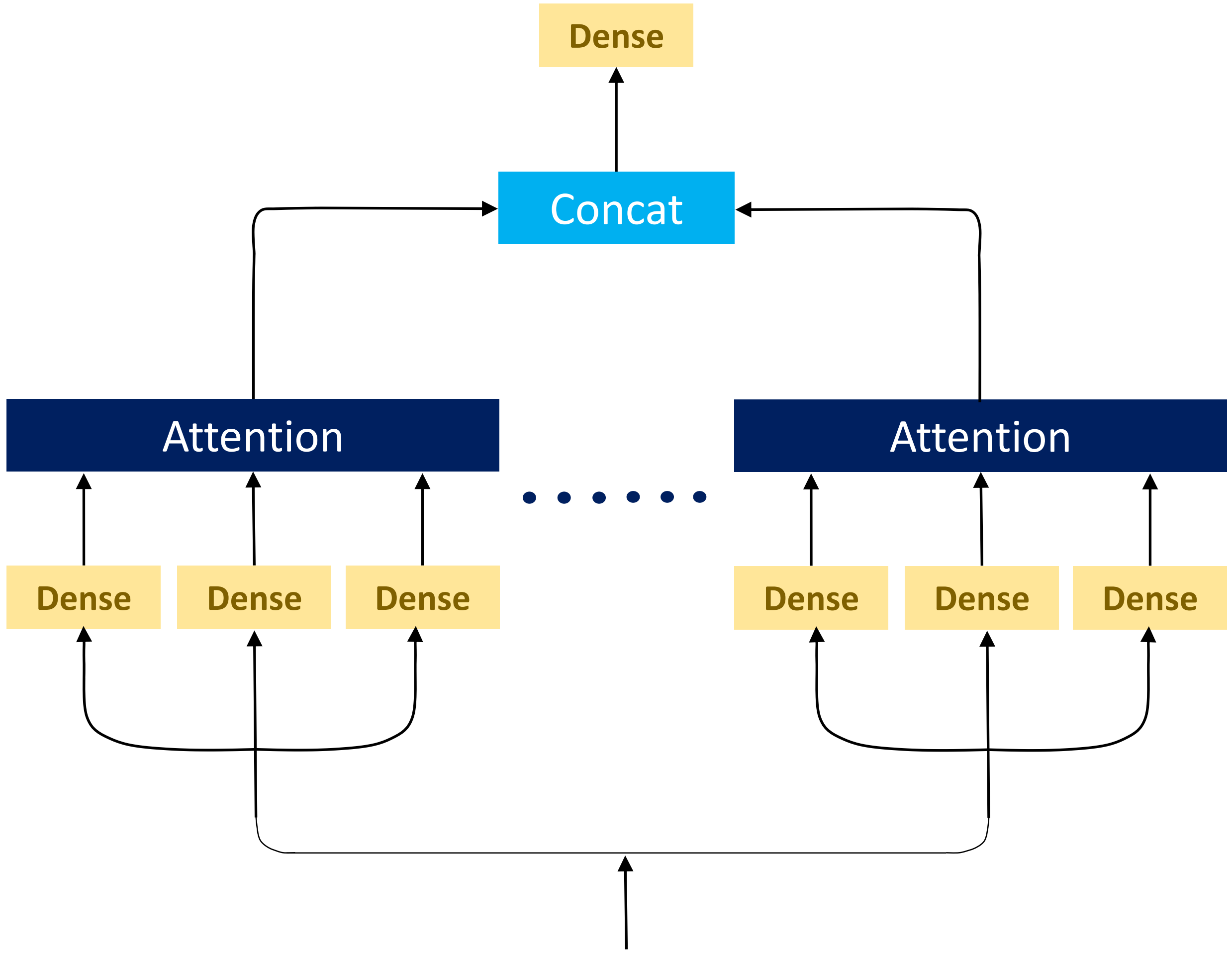
- Then the parameters associated with the dense layers prior to the mth attention layer are
  - $\mathbf{W}_q^m \in \mathbb{R}^{p_q \times l_q}$ for the query dense layer
  - $\mathbf{W}_k^m \in \mathbb{R}^{p_k \times l_k}$ for the key dense layer
  - $\mathbf{W}_v^m \in \mathbb{R}^{p_v \times l_v}$ for the value dense layer

- The projections for the mth head are:

$$\mathbf{q}^{m'} = \mathbf{W}_q^m \mathbf{q}$$
$$\mathbf{k}^{m'} = \mathbf{W}_k^m \mathbf{k}$$
$$\mathbf{v}^{m'} = \mathbf{W}_v^m \mathbf{v}$$

- The output of the mth head is given as

$$\mathbf{o}^{(m)} = A(\mathbf{q}^{m'}, \mathbf{k}^{m'}, \mathbf{v}^{m'})$$

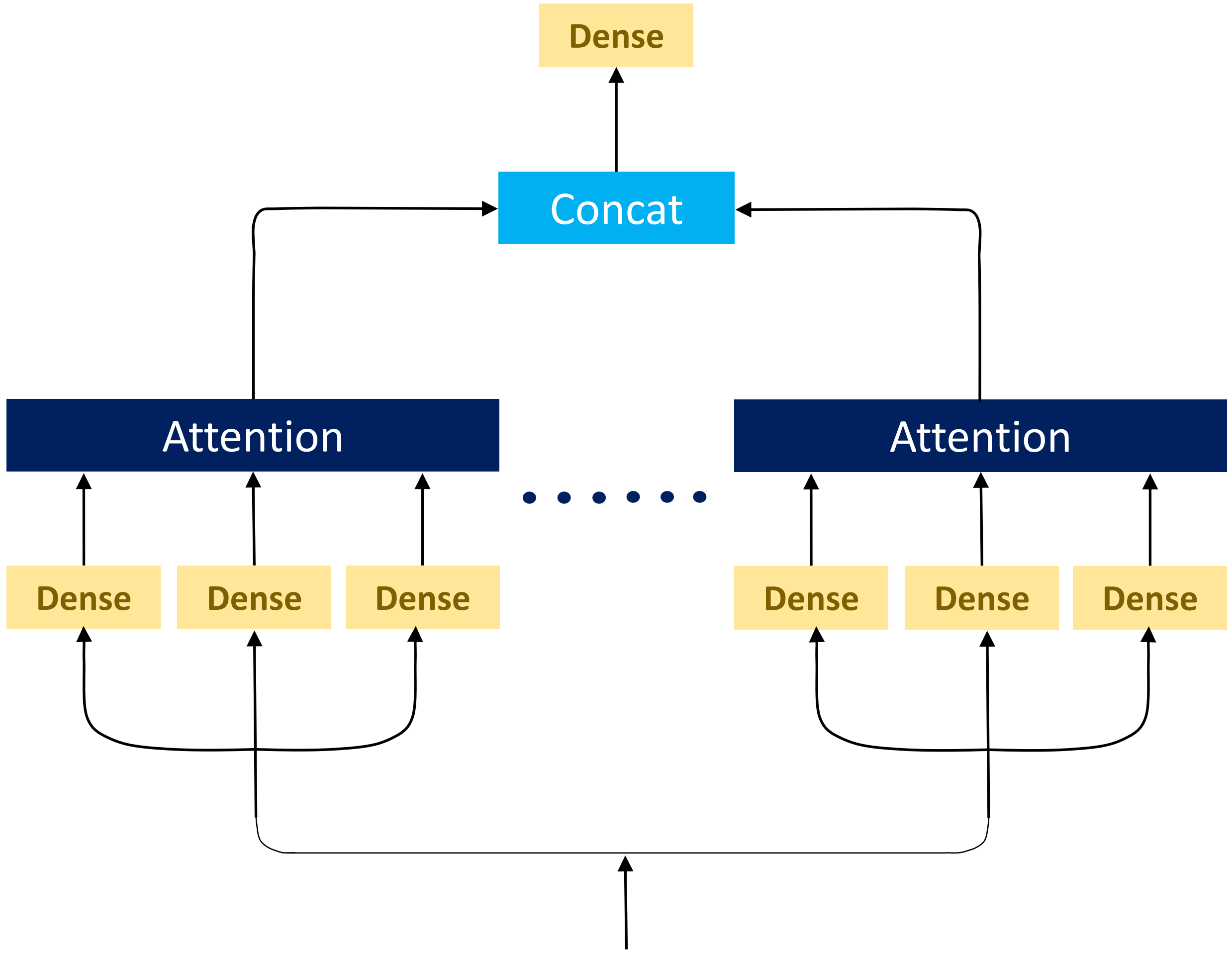


Multi-head attention

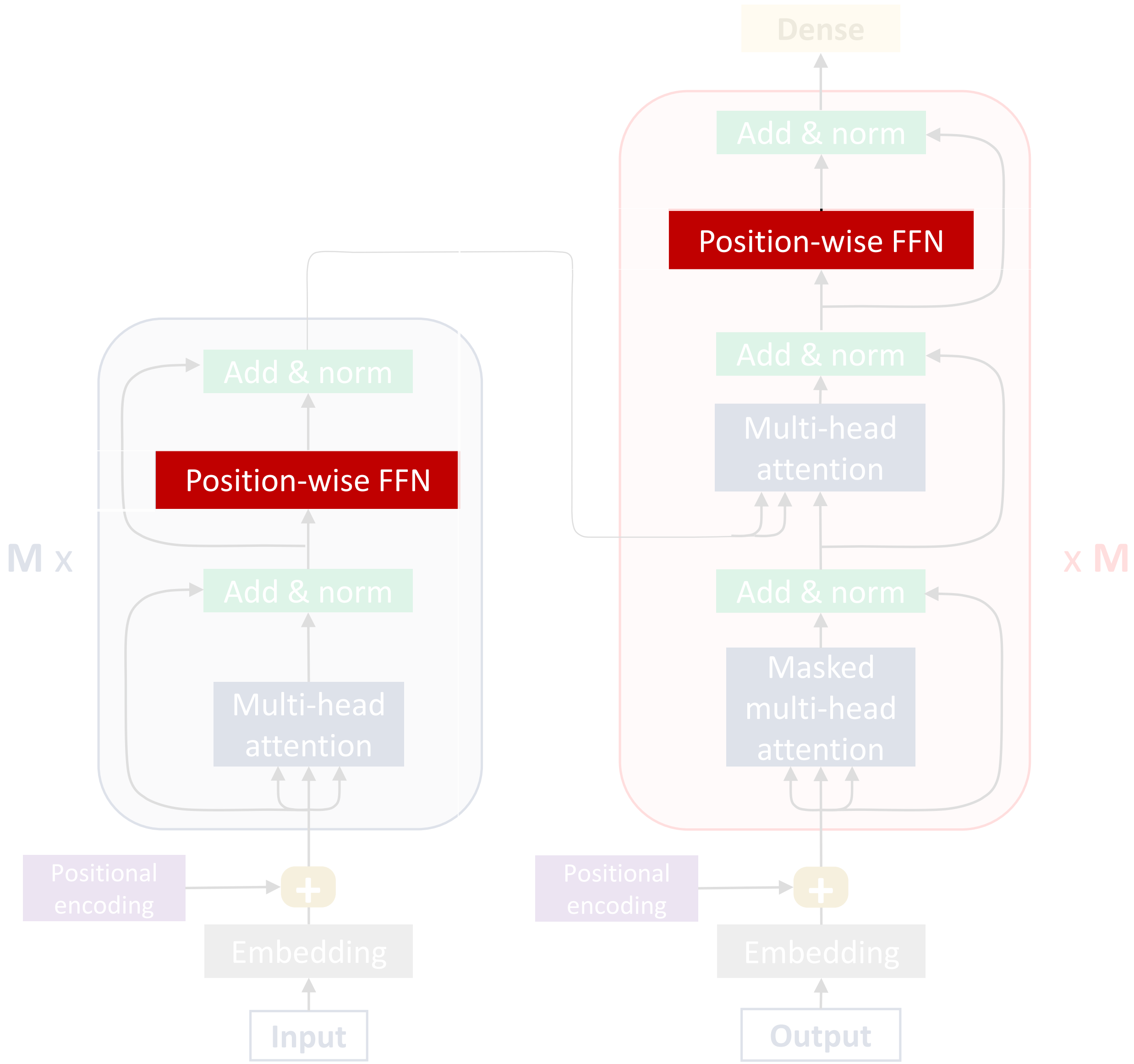
- Finally, the output of the multi-head attention

$$\mathbf{o} = \mathbf{W}_f \begin{bmatrix} \mathbf{o}^{(1)} \\ \mathbf{o}^{(2)} \\ \cdot \\ \cdot \\ \mathbf{o}^{(M)} \end{bmatrix}$$

where $\mathbf{W}_f$ is the weight matrix corresponding to the final dense layer.



Position-wise FFN



- Receives input with shape:

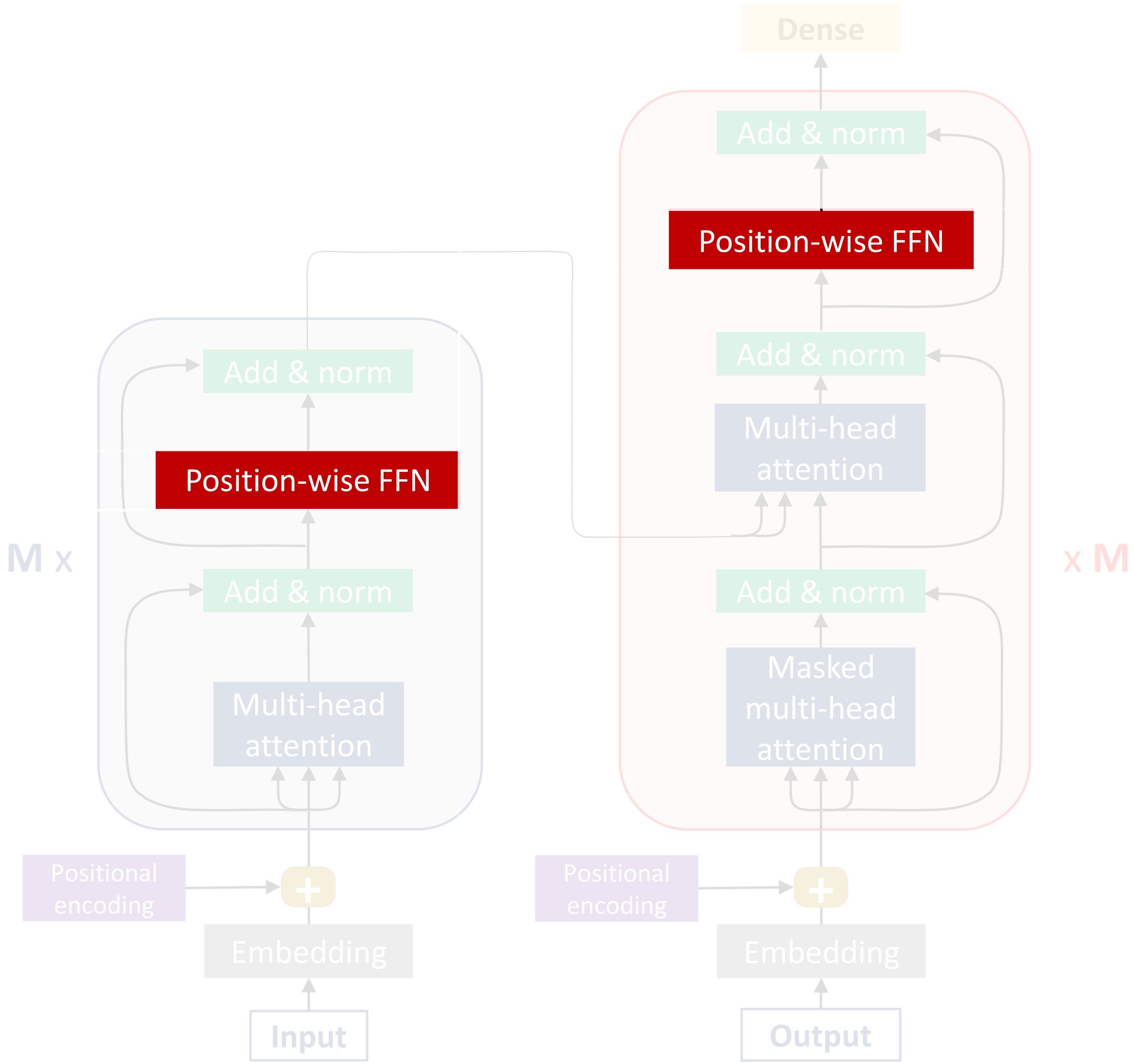
  (batch size, sequence length, feature dimension)

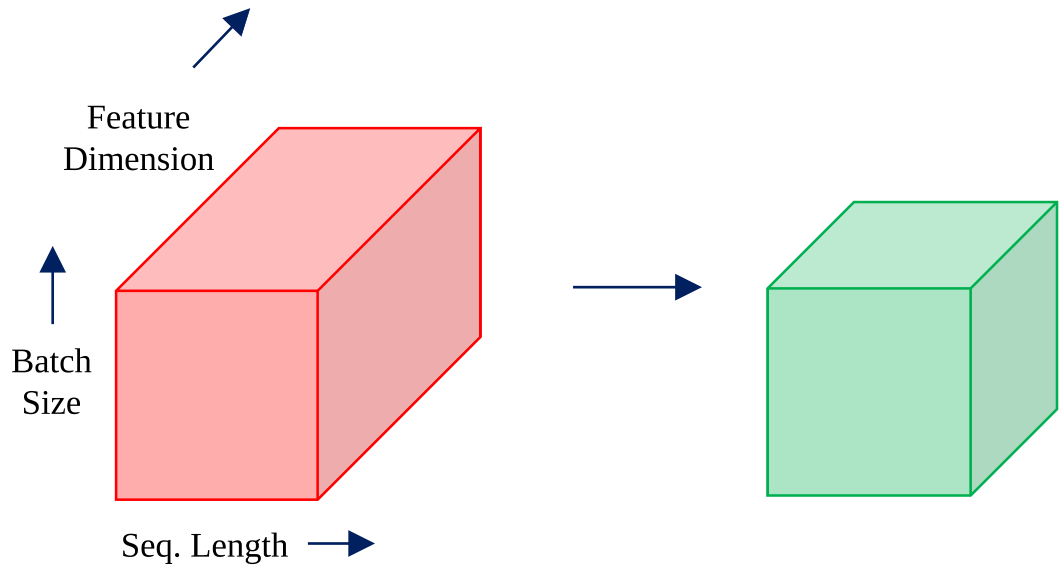
- Reshapes the input to

  (batch size $\times$ sequence length, feature dimension)

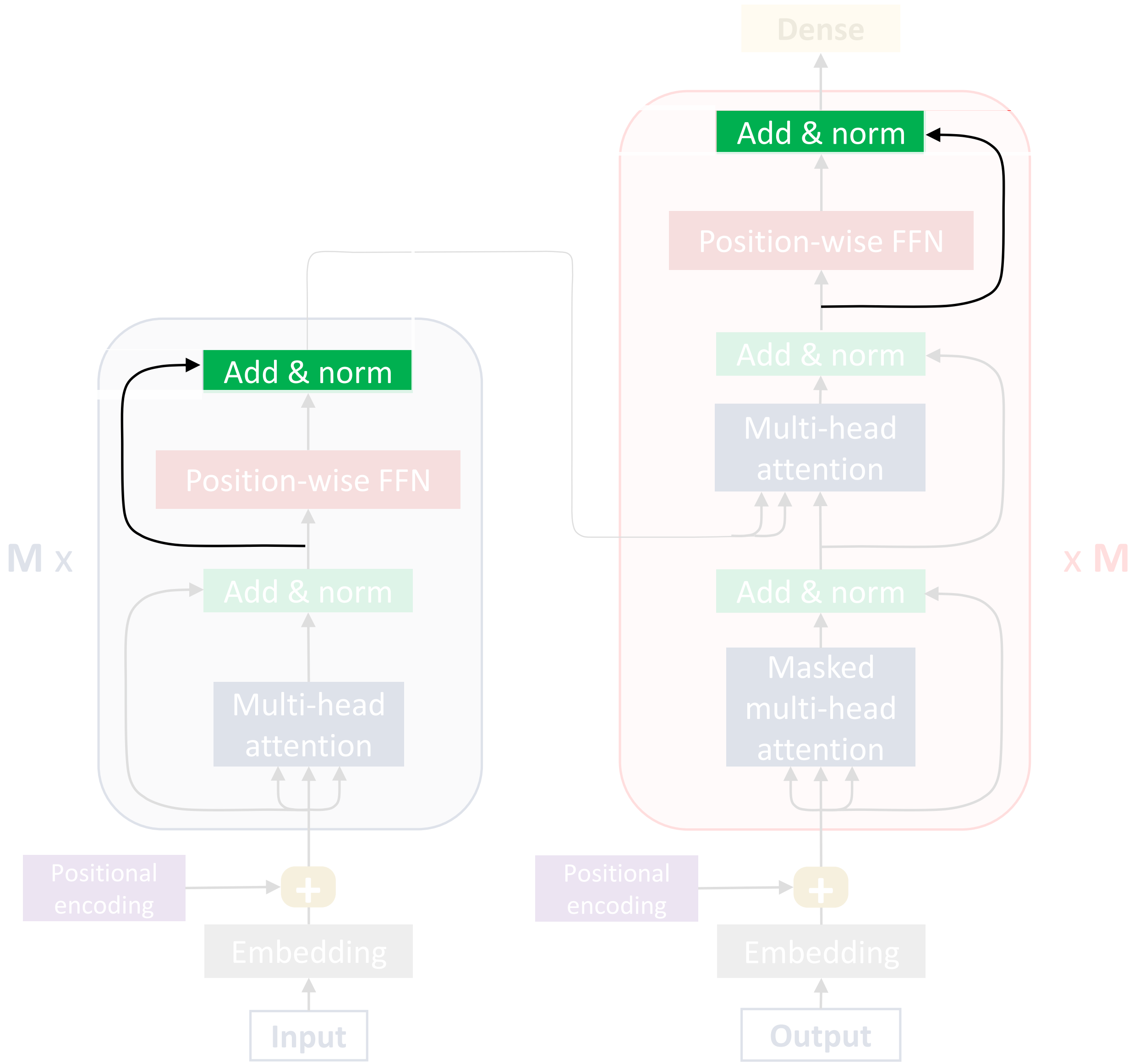
- Pass the reshaped inputs through the two dense layers.
- Reshape back to 3D.
- Equivalent to application of of two (1×1) convolutions.

Position-wise FFN



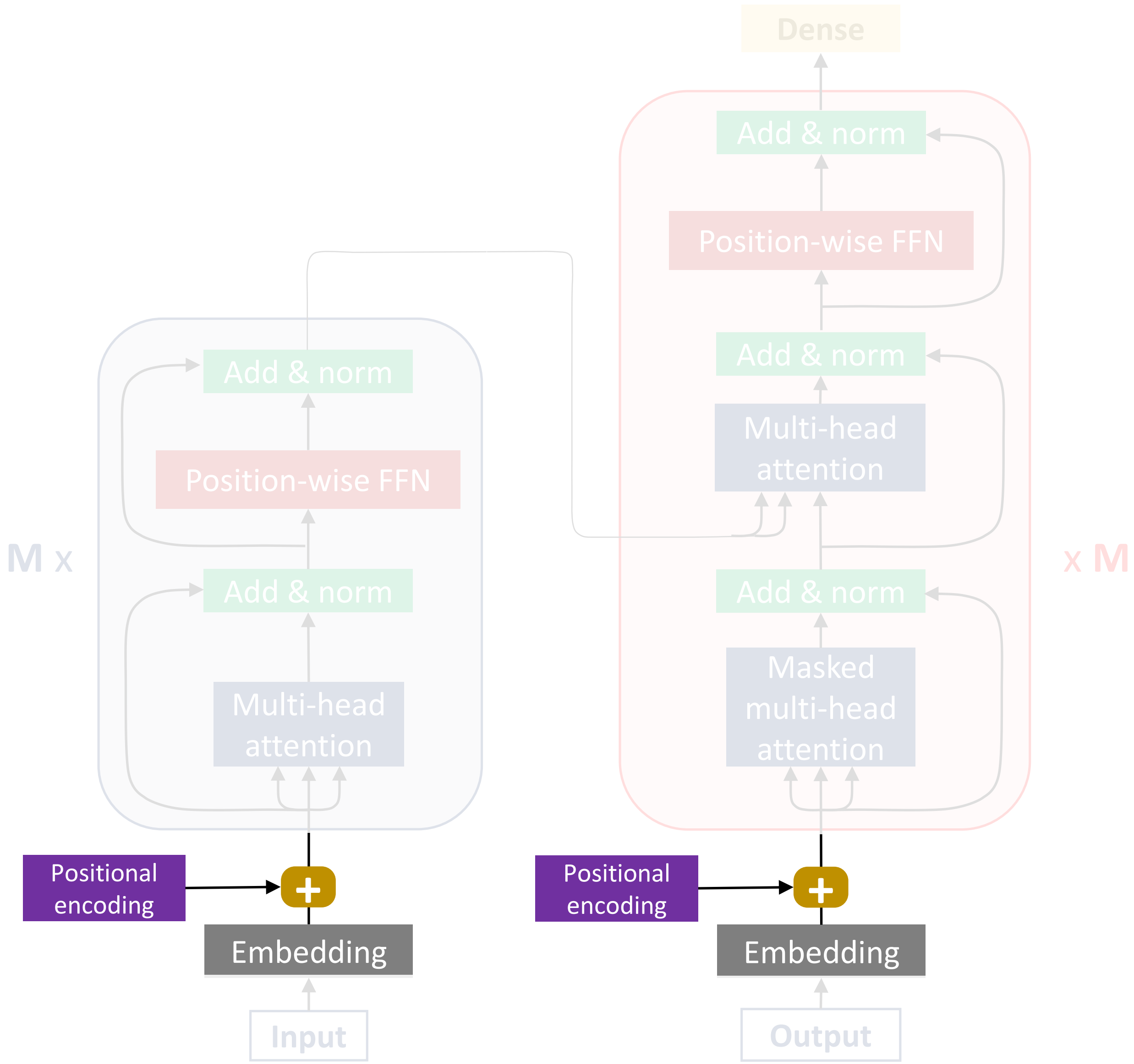


Add & norm



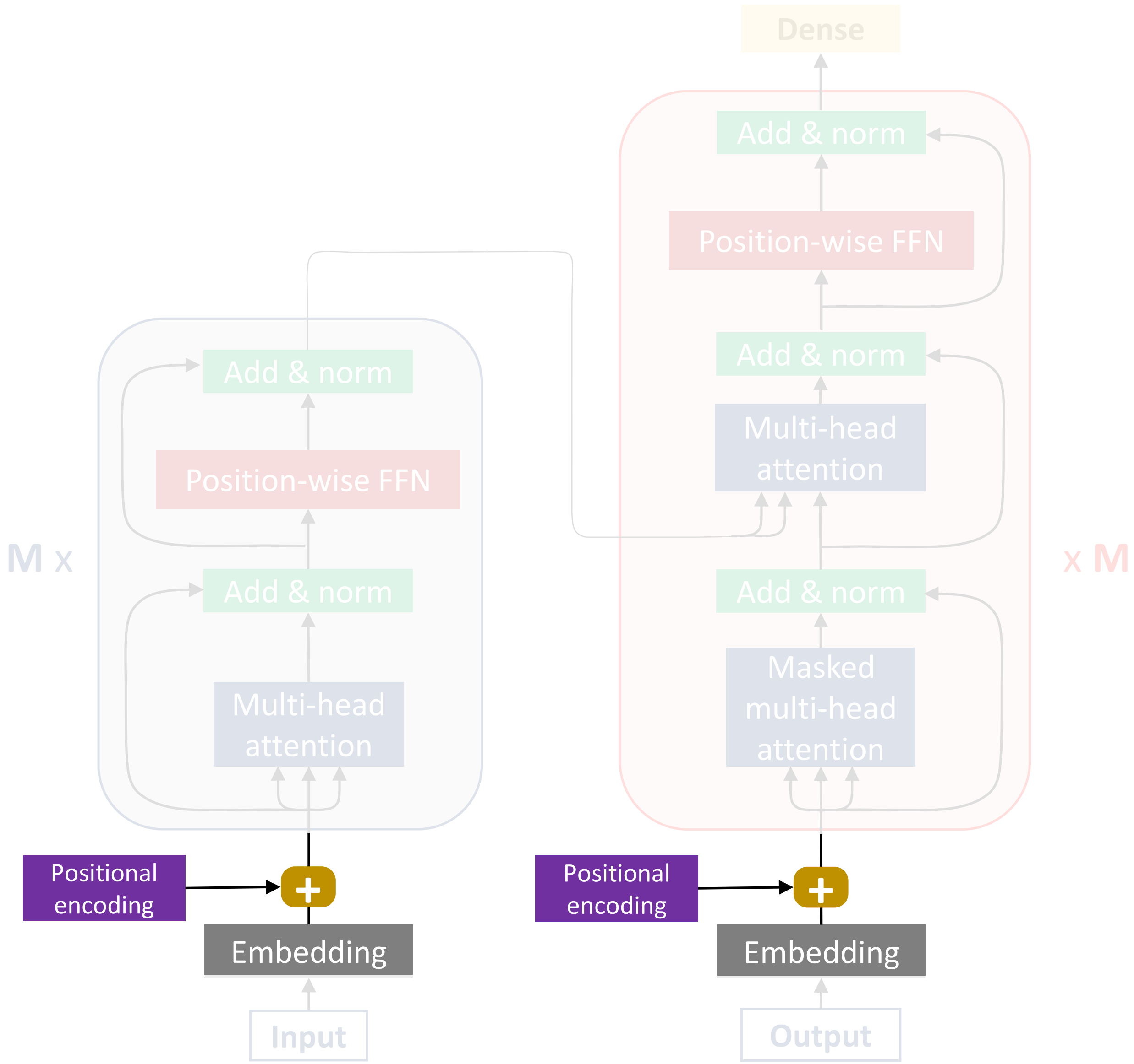
- Residual connection is employed around each of the sublayer.
- Performs layer normalization
  - Mean and variance for layer normalization are calculated w.r.t. the feature dimension.

Positional encoding



- Positional encoding enables the transformer to capture sequential information.
  - The attention and the FFN layers do not retain any sequential information.
- Suppose $\mathbf{z} \in \mathbb{R}^{s \times d_e}$ is the embedding of a given example $\mathbf{x}$.
  - s is the length of the sequence
  - d_e is the size of the embedding
- Positional encoding generates a matrix $\mathbf{P}$ which encodes a certain form of sequential information
  - The no. of rows of $\mathbf{P}$ is equal to sequence length.
  - The no. of columns of $\mathbf{P}$ is equal to the embedding size.

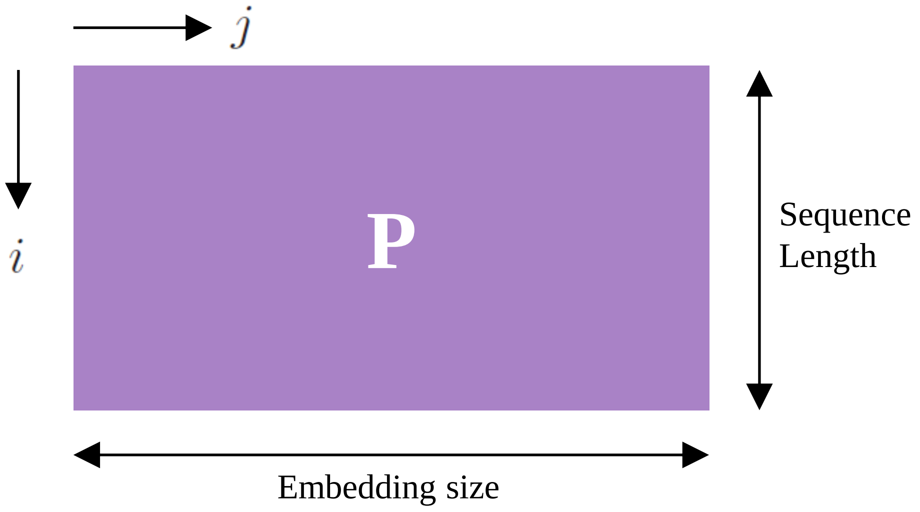
Positional encoding



- The matrix $\mathbf{P}$ is defined as

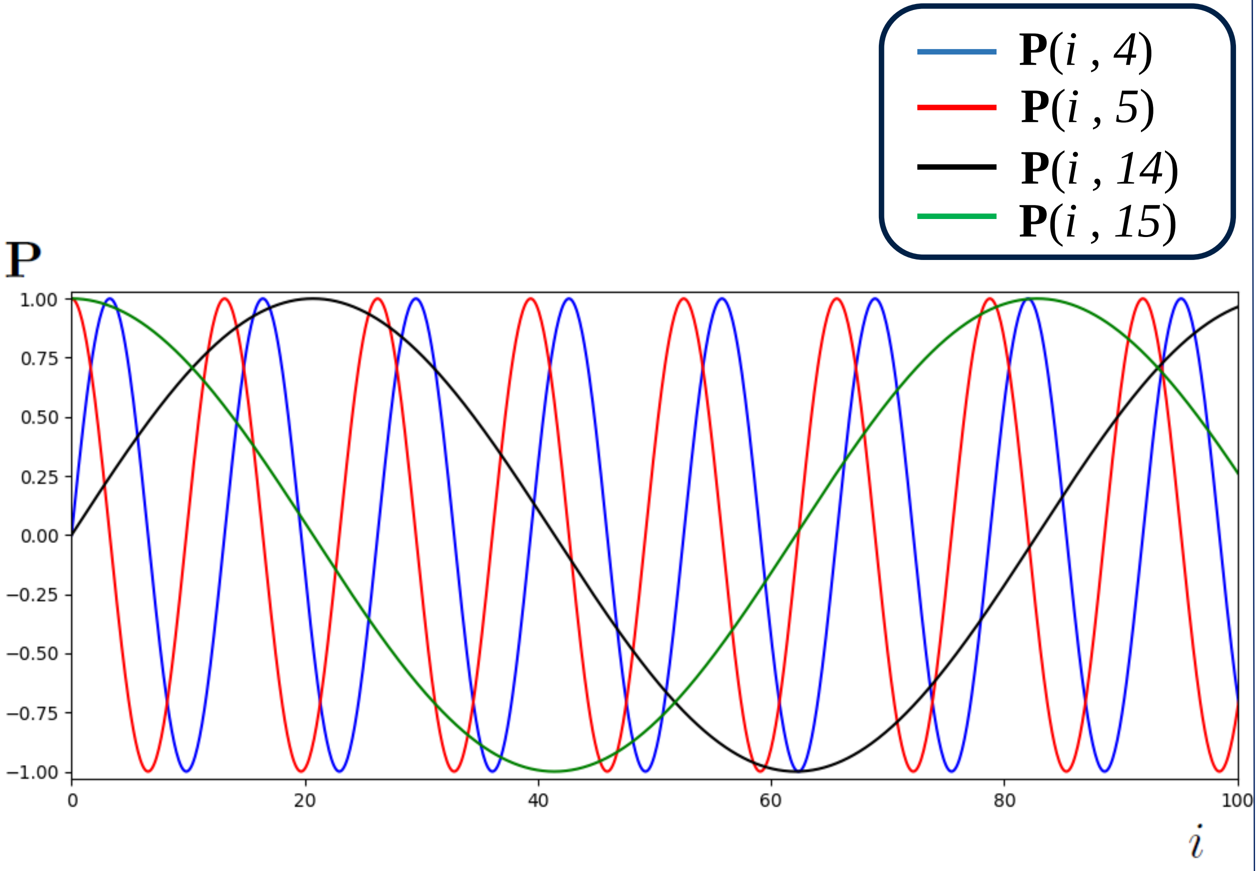
$$\mathbf{P}(i, 2j) = \sin\left(\frac{i}{10000^{2j/d}}\right)$$

$$\mathbf{P}(i, 2j+1) = \cos\left(\frac{i}{10000^{2j/d}}\right)$$



- The output of the positional encoding layer is $\mathbf{P} + \mathbf{z}$

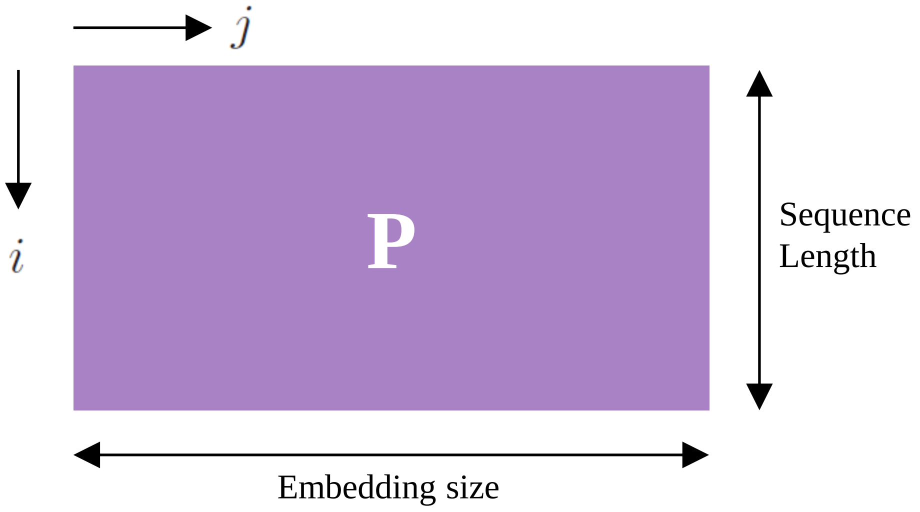
Positional encoding



- The matrix $\mathbf{P}$ is defined as

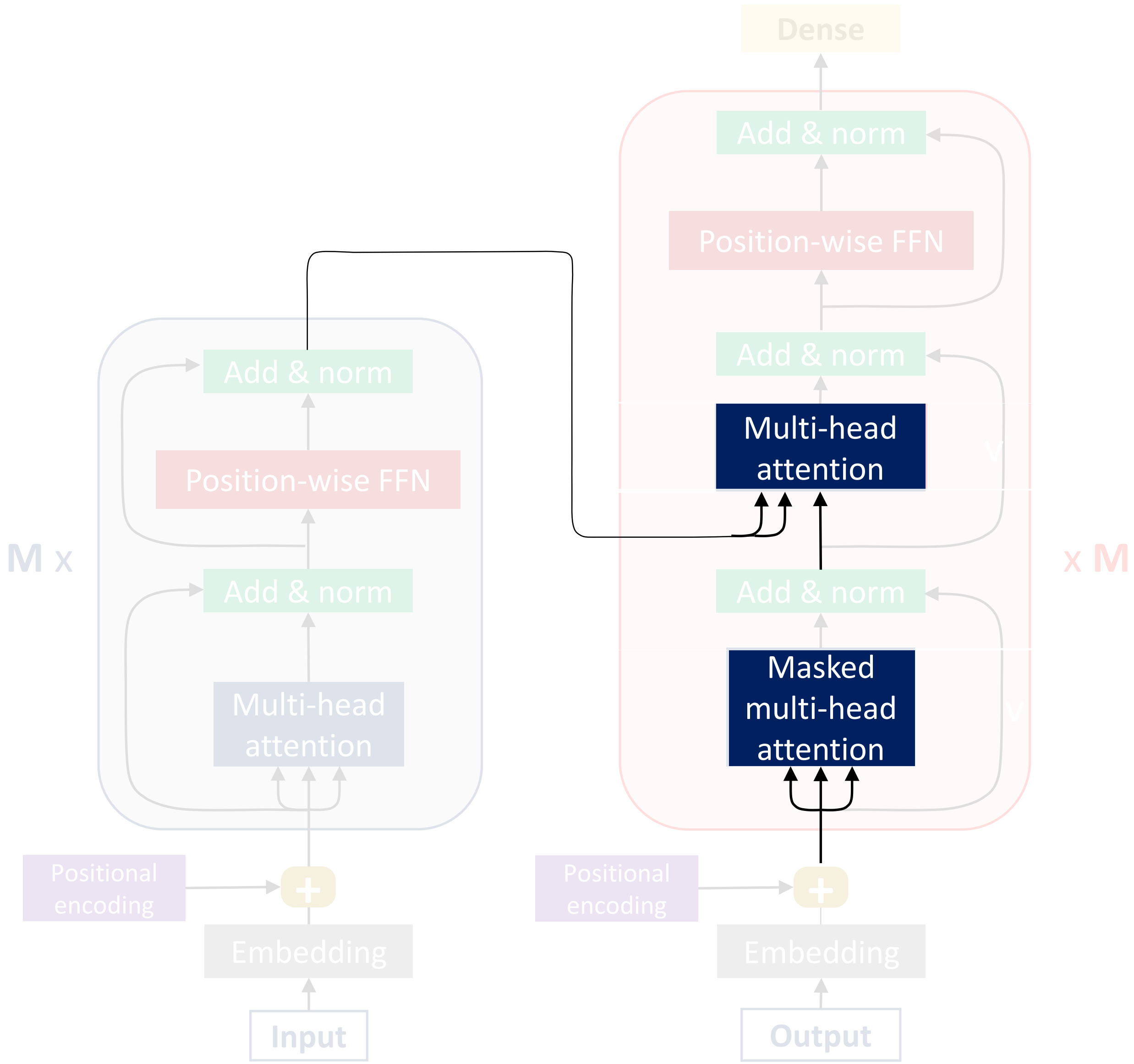
$$\mathbf{P}(i, 2j) = \sin\left(\frac{i}{10000^{2j/d}}\right)$$

$$\mathbf{P}(i, 2j+1) = \cos\left(\frac{i}{10000^{2j/d}}\right)$$



- The output of the positional encoding layer is $\mathbf{P} + \mathbf{z}$

Multi-head attention layers in Decoder



- Masked attention layer:

  This self-attention mechanism attends to all positions in the decoder up to and including that position.

- Encoder-Decoder attention layer:
  - Queries come from the previous decoder layer
  - Keys and Values correspond to encoder outputs

Flow of information

- Let us look at what part of a sentence a transformer network is attending to when processing/translating a given word.
- Consider the two sentence given below:

The animal didn't cross the street because it was too tired.

– Here "it" refers to the animal

The animal didn't cross the street because it was too wide.

– Here "it" refers to the street

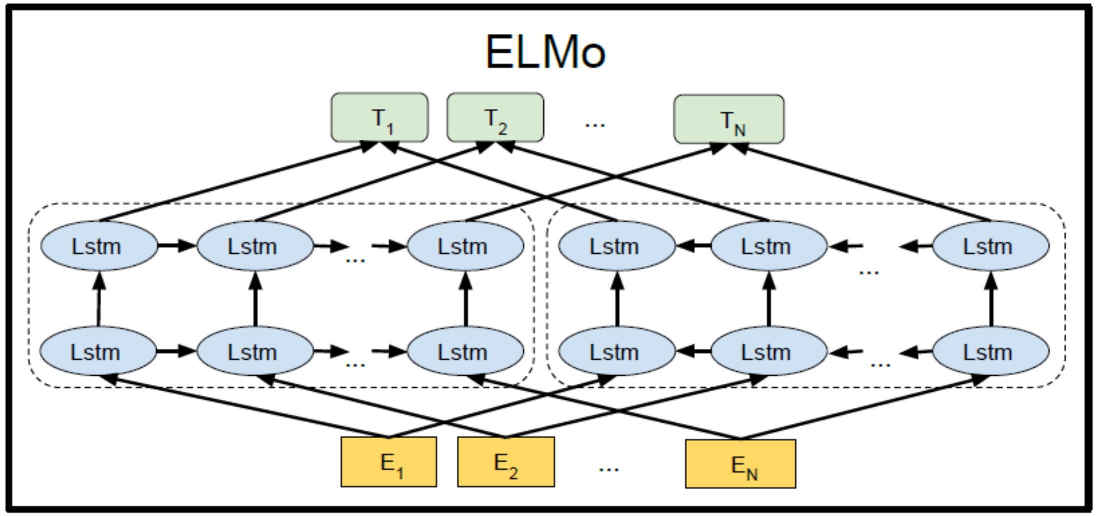
- When translating these sentences to French, the translation for "it" depends on the gender of the noun that is referred to.

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

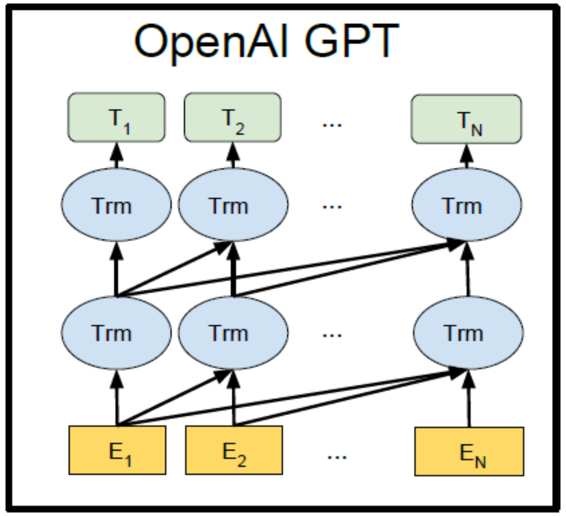
BERT

Devlin *et. al.* BERT: Pre-training Deep Bidirectional Transformers for Language Understanding, Proceedings of NAACL-HLT 2019

Earlier models



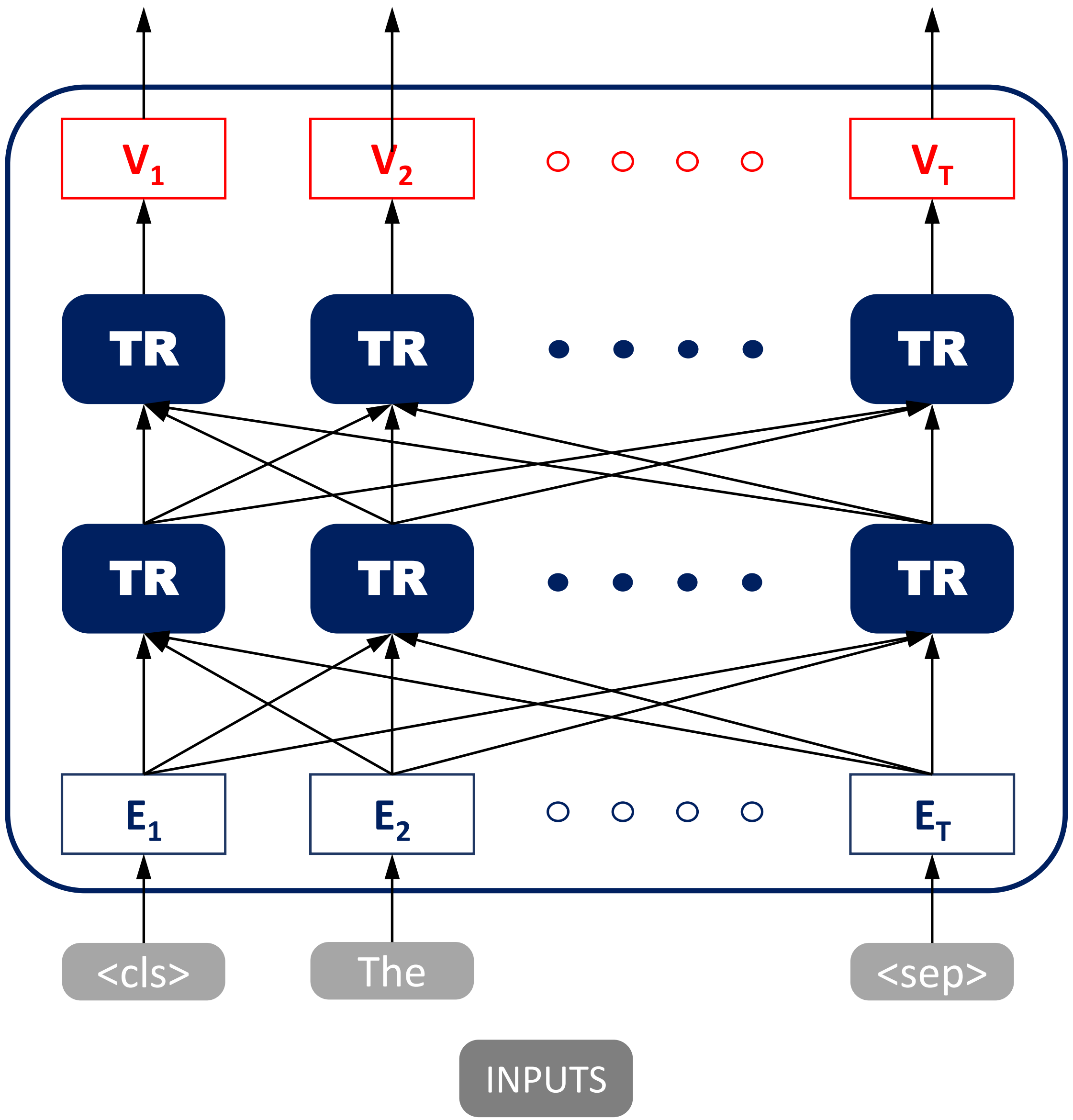
- Two language models:
  1. Left-to-right language model
  2. Right-to-left language model
- The contextual representation of each token is taken to be the concatenation of the representations of the two language models
- Both the model are trained independently



- Transformer based model
- Unidirectional: left-to-right architecture
  - Does not incorporate context from both directions

Figures source: Devlin *et. al.* BERT: Pre-training Deep Bidirectional Transformers for Language Understanding, Proceedings of NAACL-HLT 2019

BERT

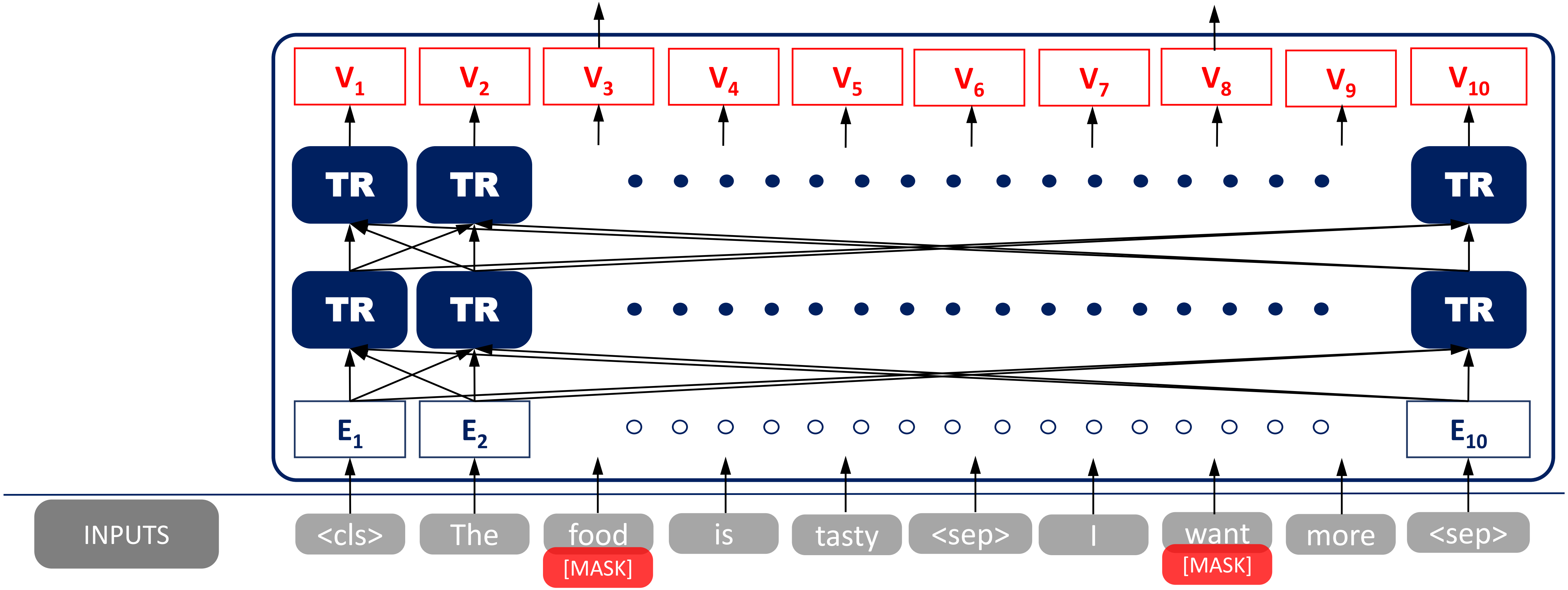


- Bidirectional Encoder Representations from Transformers
- Architecture comprise multiple layers of transformer encoders
- Motivation: Need pre-trained language models that can achieve state-of-the-art performance.
- A big shortcoming of standard language models is unidirectionality.
  - Such limitations are sub-optimal for sentence level tasks.
- In many NLP tasks, e.g. question-answering, it is important the context from both sides are incorporated.

Input representation

Net Embeddings	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
	=	=	=	=	=	=	=	=	=	=
Positional Embeddings	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	S_1	S_1	S_1	S_1	S_1	S_1	S_2	S_2	S_2	S_2
	+	+	+	+	+	+	+	+	+	+
Token Embeddings	T(<cls>)	T(The)	T(food)	T(is)	T(tasty)	T(<sep>)	T(I)	T(want)	T(more)	T(<sep>)
INPUTS	<cls>	The	food	is	tasty	<sep>	I	want	more	<sep>

Pre-training BERT: Task 1



- General procedure:
  - Some percentage of the input tokens are masked at random.
  - Final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary.

Pre-training BERT: Task 1

- The training data generator chooses 15% of tokens at random for prediction.
- If the t-th token is chosen, then it is replaced with

  1. the [MASK] token 80% of the time

INPUTS	<cls>	The	food	is	tasty	<sep>	I	want	more	<sep>
			[MASK]					[MASK]		

  2. a random token 10% of the time

INPUTS	<cls>	The	angry	is	tasty	<sep>	I	Hello	more	<sep>

  3. unchanged token 10% of the time

INPUTS	<cls>	The	food	is	tasty	<sep>	I	want	more	<sep>

- Finally, the vector $\mathbf{v}_t$ is used to predict the original token.

Pre-training BERT: Task 2

- The purpose is to make the model understand sentence relationships.
- Idea: Pre-train a binarized next-sentence prediction task.
- The training data generator chooses two sentences $\mathcal{S}_A$ and $\mathcal{S}_B$, such that
  - 50% of the time $\mathcal{S}_B$ is the actual sentence following $\mathcal{S}_A$. This case is labelled as $IsNext$.

INPUTS	<cls> The food is tasty <sep> I want more <sep>
OUTPUT	$IsNext$

  - 50% of the time $\mathcal{S}_B$ is a random sentence from the corpus. This case is labelled as $NotNext$.

INPUTS	<cls> The food is tasty <sep> Delhi is in India <sep>
OUTPUT	$NotNext$

GLUE Benchmark

- Multi-Genre Natural Language Inference (MNLI): For a pair of sentences, predict if the 2nd sentence neutral w.r.t. 1st sentence.
- Quora Question Pairs (QQP): Binary classification task to determine if two questions are semantically different.
- Question Natural Language Inference (QNLI): Binary classification task where positive example are Q-A pairs with the correct answer and negative examples Q-A pairs which do not contain the answer.
- Stanford Sentiment Treebank (SST-2): Binary sentiment classification task with sentences from movie reviews.
- Corpus of Linguistic Acceptability (CoLA): Binary classification task to predict if a given sentence is linguistically acceptable.
- Semantic Textual Similarity Benchmark (STS-B):: Indicate the (semantic) similarity of 2 sentences on a scale of 1 to 5.

Pre-training → Fine-Tuning

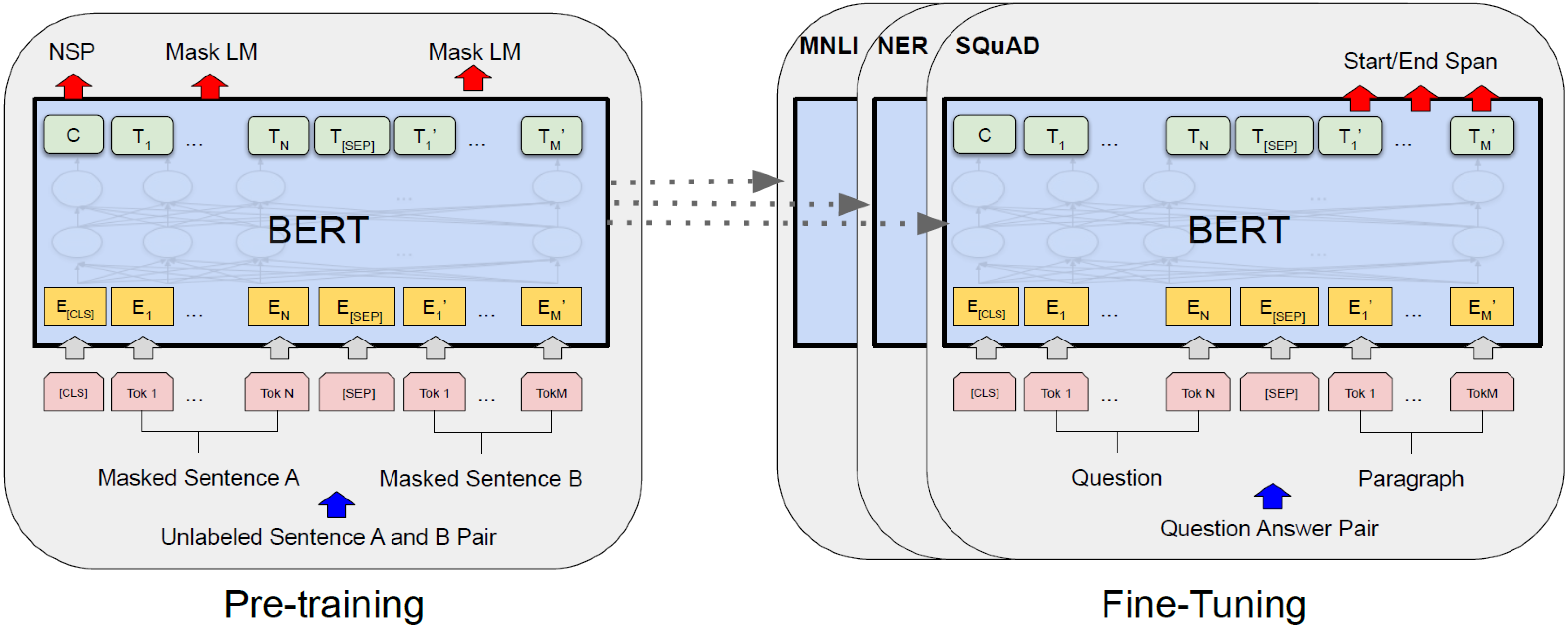


Figure source: Devlin *et. al.* BERT: Pre-training Deep Bidirectional Transformers for Language Understanding, Proceedings of NAACL-HLT 2019