Transformer Architecture

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

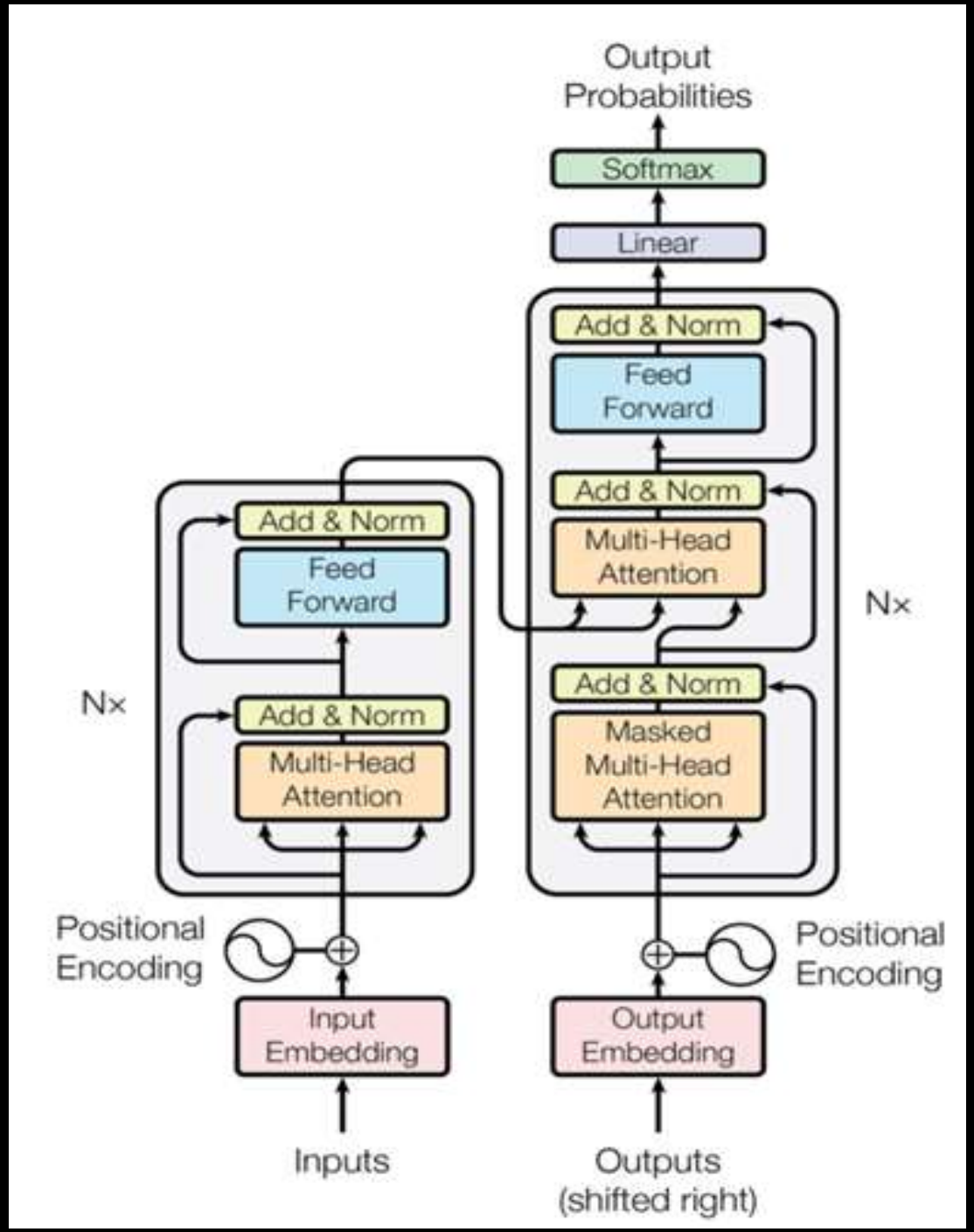
Jakob Uszkoreit[*]
Google Research
usz@google.com

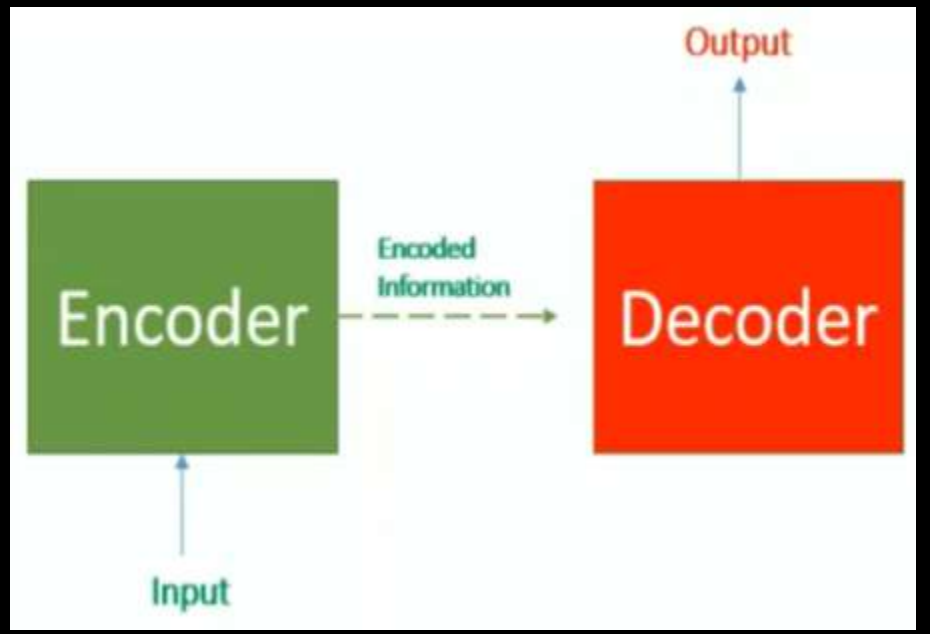
Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com


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


Output
Encoder
Encoded Information
Decoder
Input

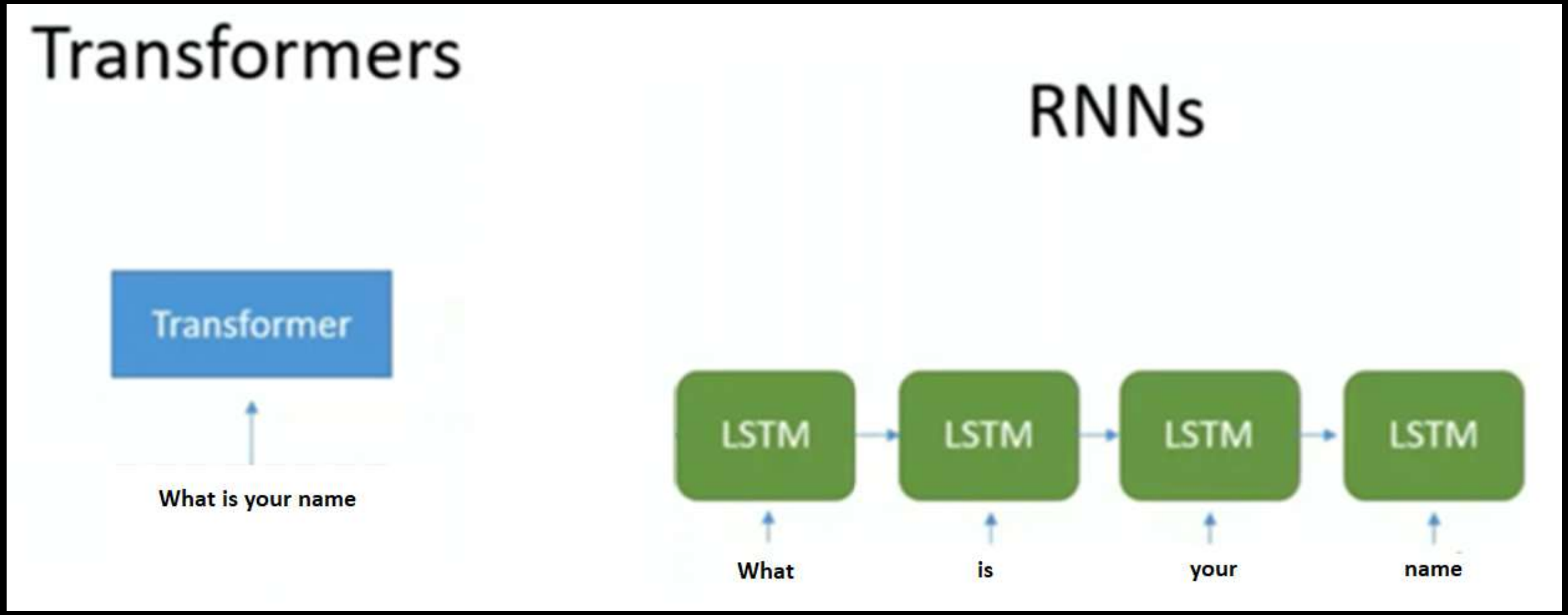
Three keypoints of the Transformer

- *Not* sequential like RNNs, all the input (ex. sentence) is fed once through the model and calculation is performed one time.
- Attention is generated from the model's own input (self-attention)
- More than one attention is generated each time (multi-head attention)

Not Sequential

Sequential means that you cannot generate next word until you know the previous word. It is sequential, the current time stamp can only generate words until the previous time stamps (or previous hidden layer) has generated a word. Transformer takes all input at once and process all the words simultaneously. There is no concept of time stamp anymore in transformer

Not Sequential



Attention

How is self-attention useful?

- Consider two sentences:

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

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

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

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

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

5th Layer of the Encoder for one out of 8 attention heads

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

Context Vector

The context vector is a sum of the hidden states of the input sequence, weighted by alignment scores. Each word in the input sequence is represented by a concatenation of the two (i.e., forward and backward) RNNs hidden states..

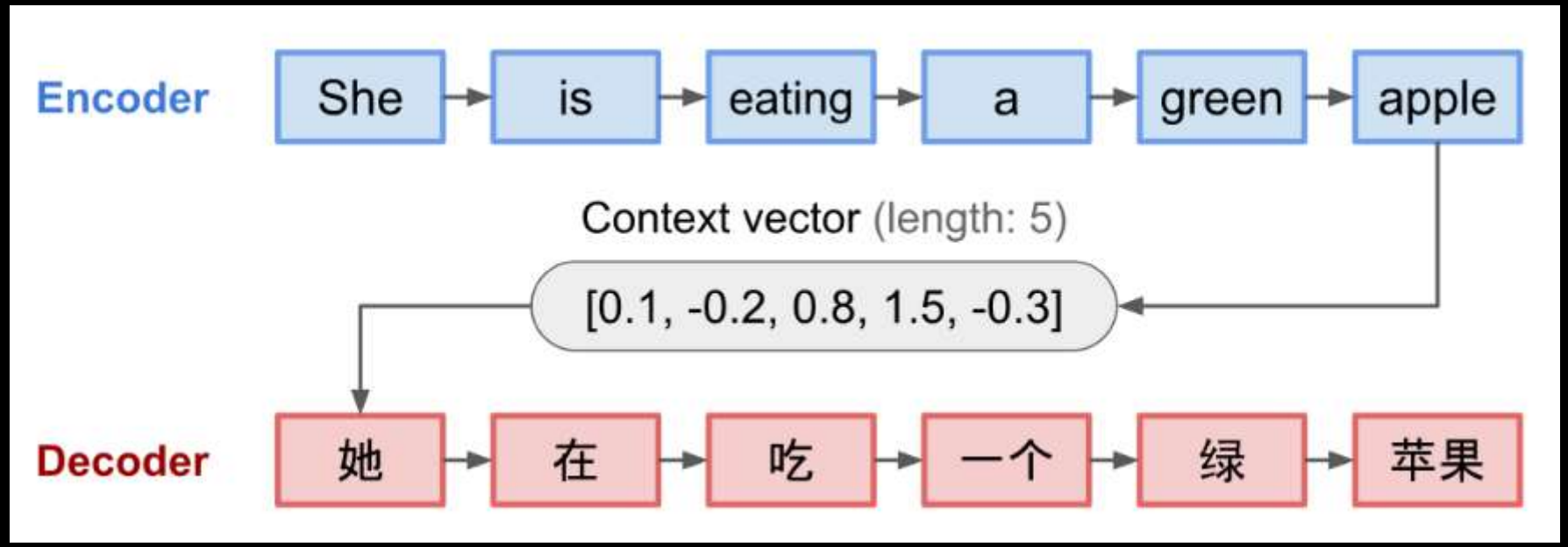
Context Vector

The seq2seq model normally has an encoder-decoder architecture.

An **encoder** processes the input sequence and compresses the information into a context vector (also known as sentence embedding or “thought” vector) of a *fixed lengt*
This representation is expected to be a good summary of the meaning of the *whole* sou sequence.

A **decoder** is initialized with the context vector to emit the transformed output.
The early work only used the last state of the encoder network as the decoder initial sta
.

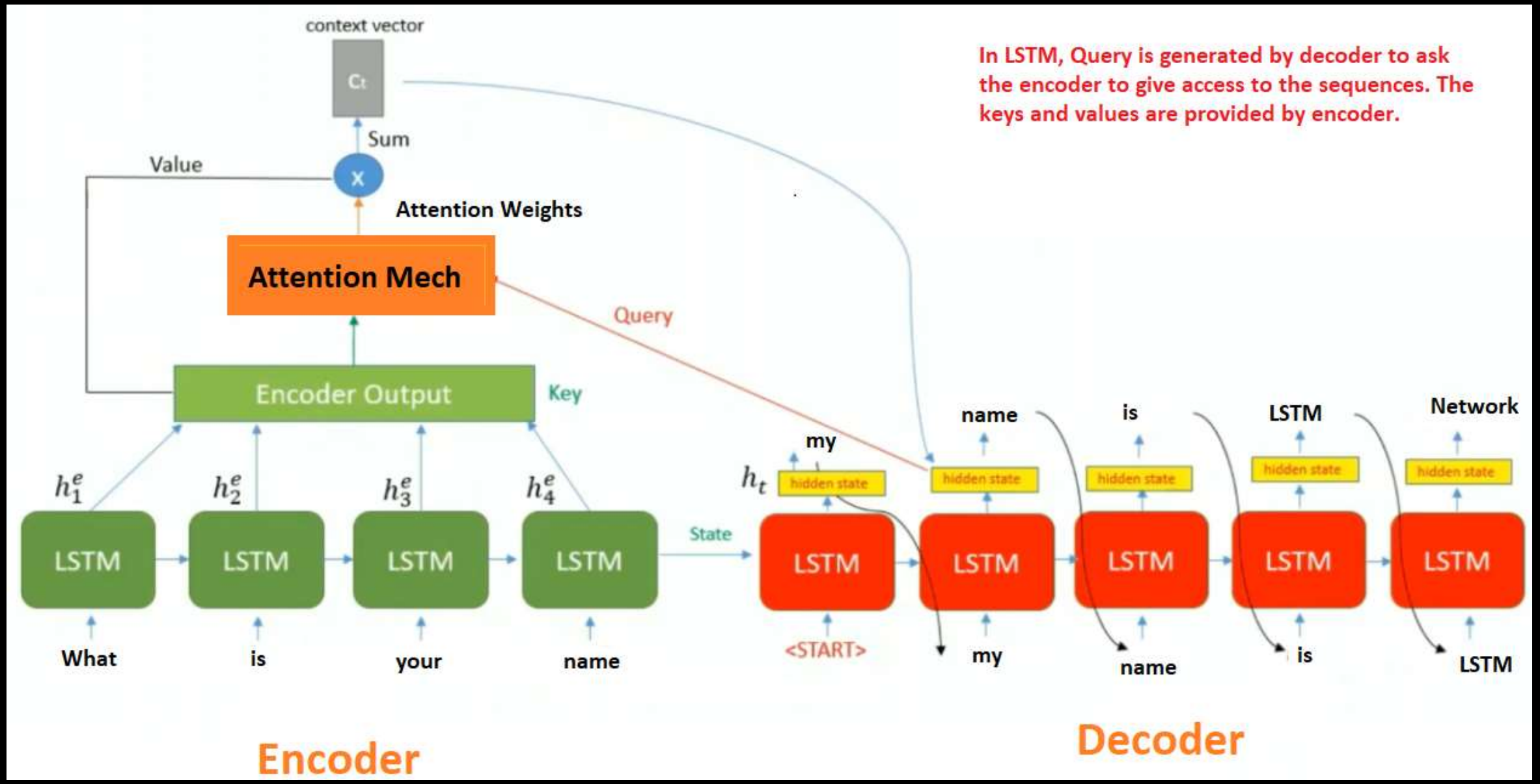
Context Vector



https://lilianweng.github.io/posts/2018-06-24-attention/

Attention Mechanism

In attention mechanism we give the access to decoder to focus on the sequences of encoder rather than waiting for the final output value of the encoder

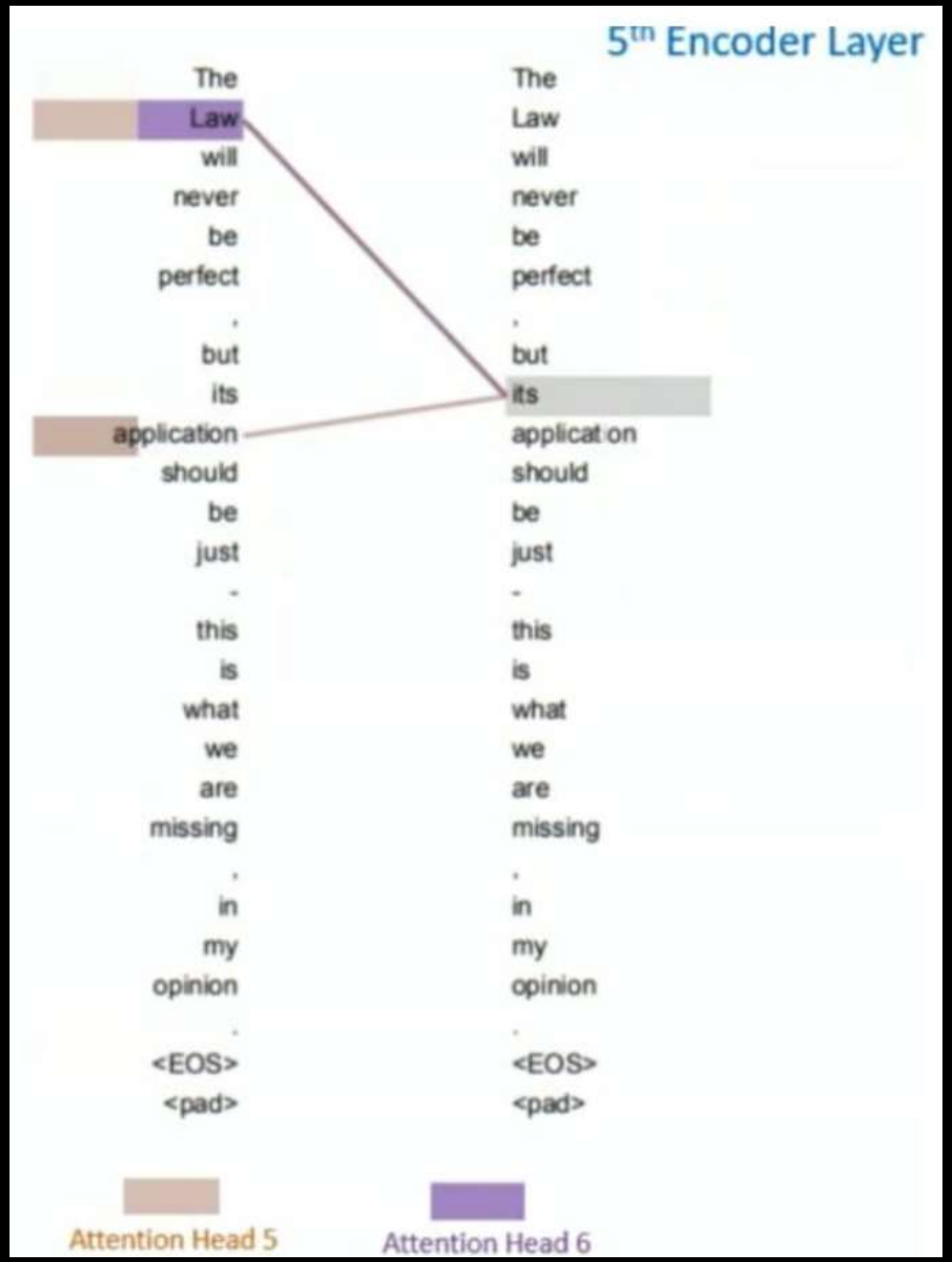

context vector
Ct
Sum
Value
X
Attention Weights
Attention Mech
Query
Encoder Output
Key
In LSTM, Query is generated by decoder to ask the encoder to give access to the sequences. The keys and values are provided by encoder.
h_1^e
h_2^e
h_3^e
h_4^e
LSTM
LSTM
LSTM
LSTM
What
is
your
name
State
h_t
my
hidden state
<START>
name
hidden state
my
is
hidden state
name
LSTM
hidden state
is
Network
hidden state
LSTM
Encoder
Decoder

Multihead Attention

Multihead attention allows the model to focus on different words in h ways.
h is the number of heads. It increases the model capacity to focus on different positions and give the attention layer multiple representations. We have h different Queries, Keys and Values.

In Multihead attention, we have h context vector and thus h representations of Values.

Multihead Attention



Embedding in Transformers

1. Word Embeddings

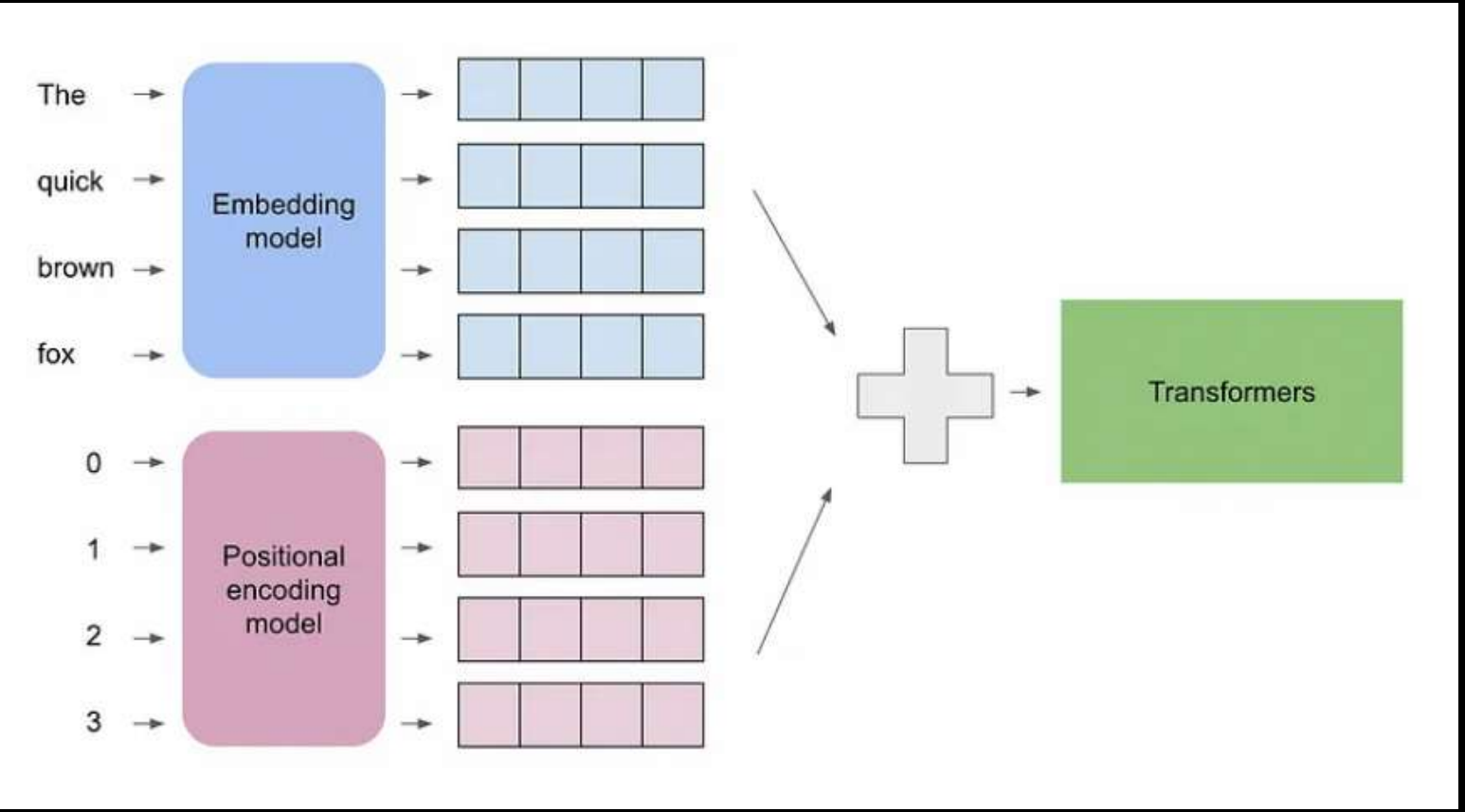
2. Positional Embeddings

Positional Embedding

In positional embedding, we use the position of the word as index.

These are added to the token embeddings before feeding into the Transformer mo
allowing the model to understand the order of the sequence.

Input Embeddings



Vision Transformer

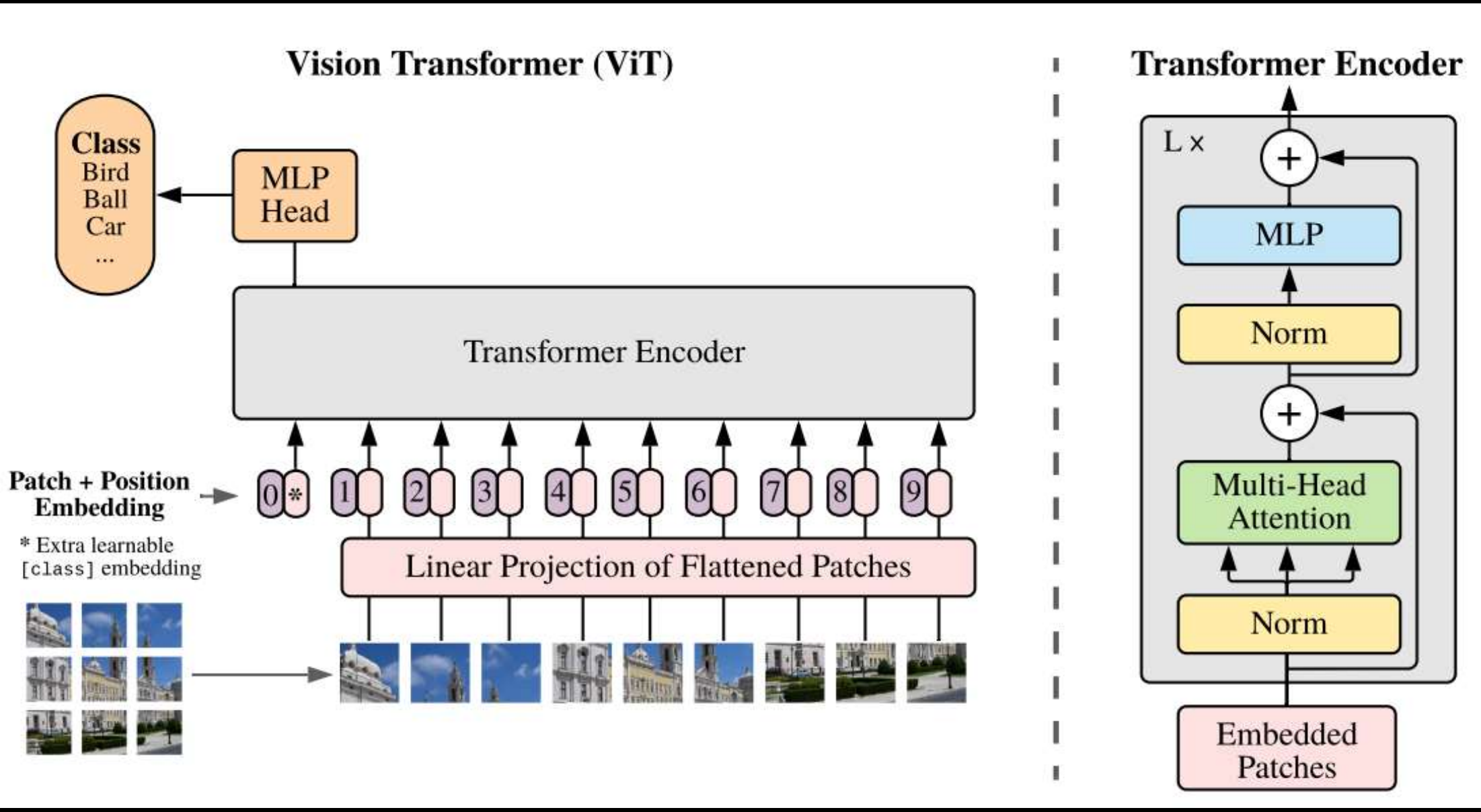
AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

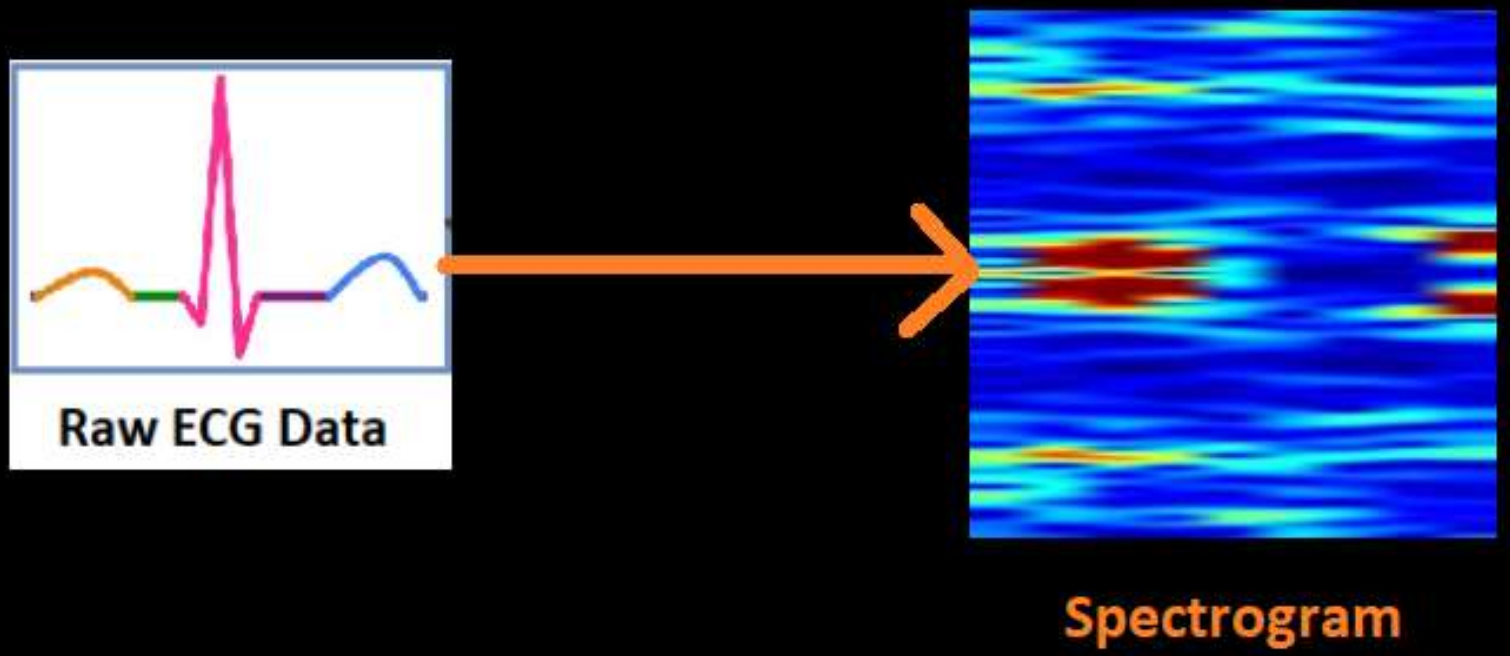
[*]equal technical contribution, [†]equal advising

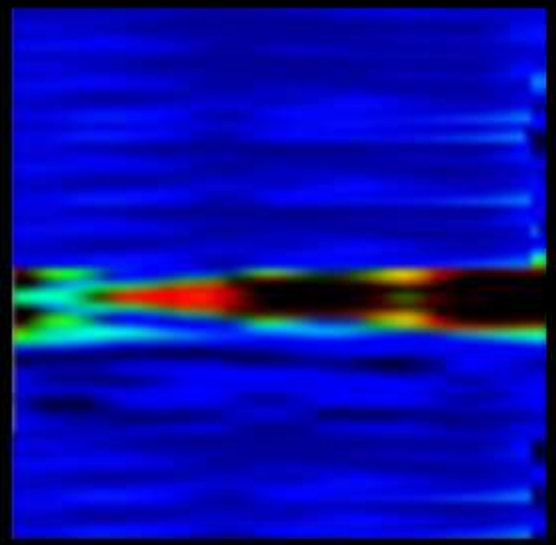
Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

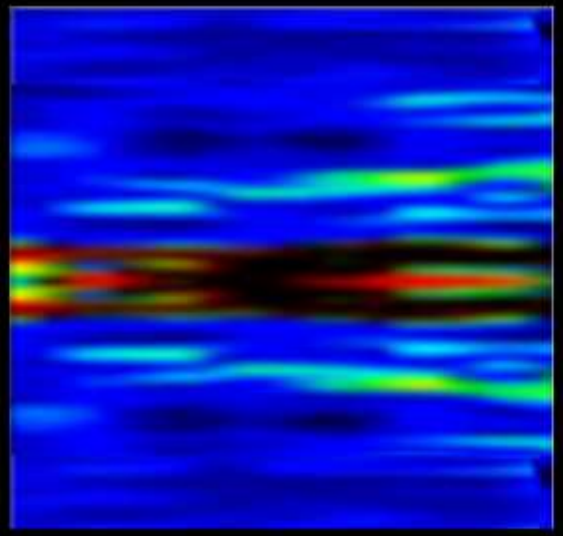

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

ECG to Spectrogram Transformation

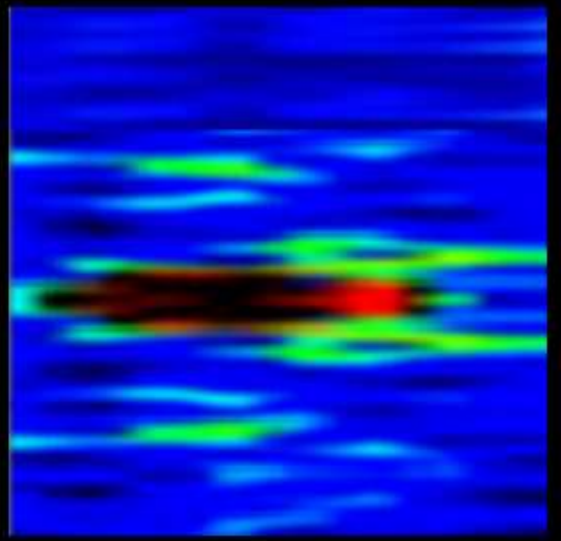




Low

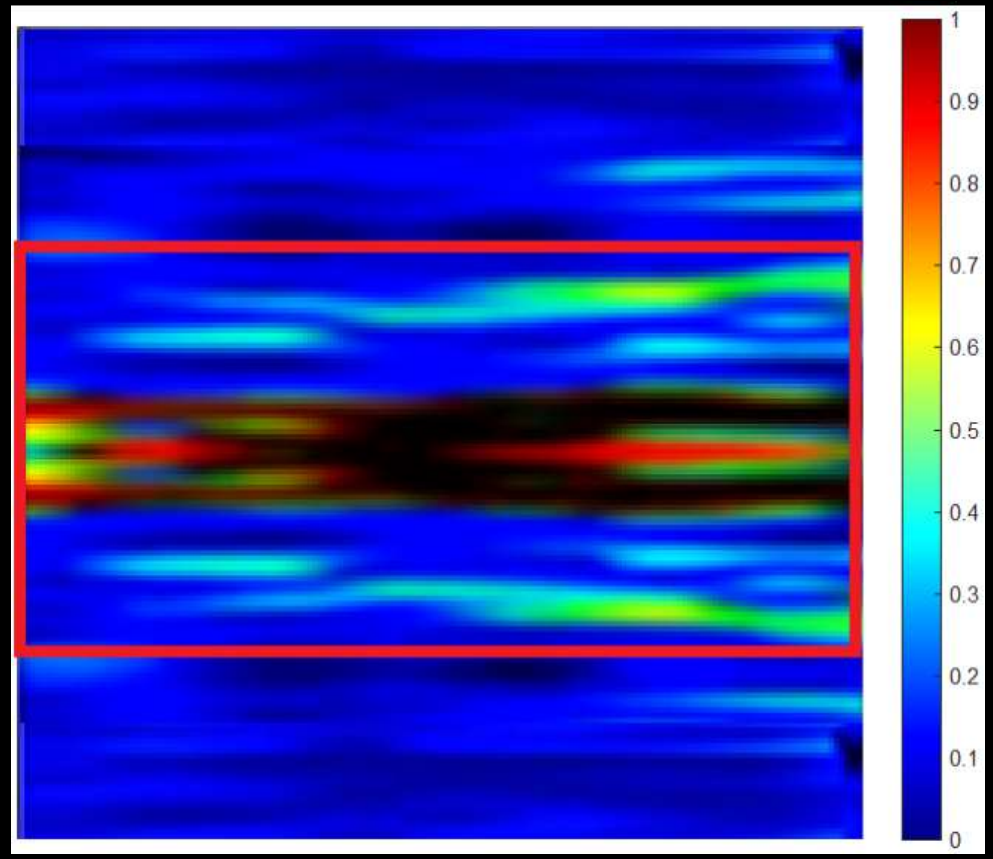


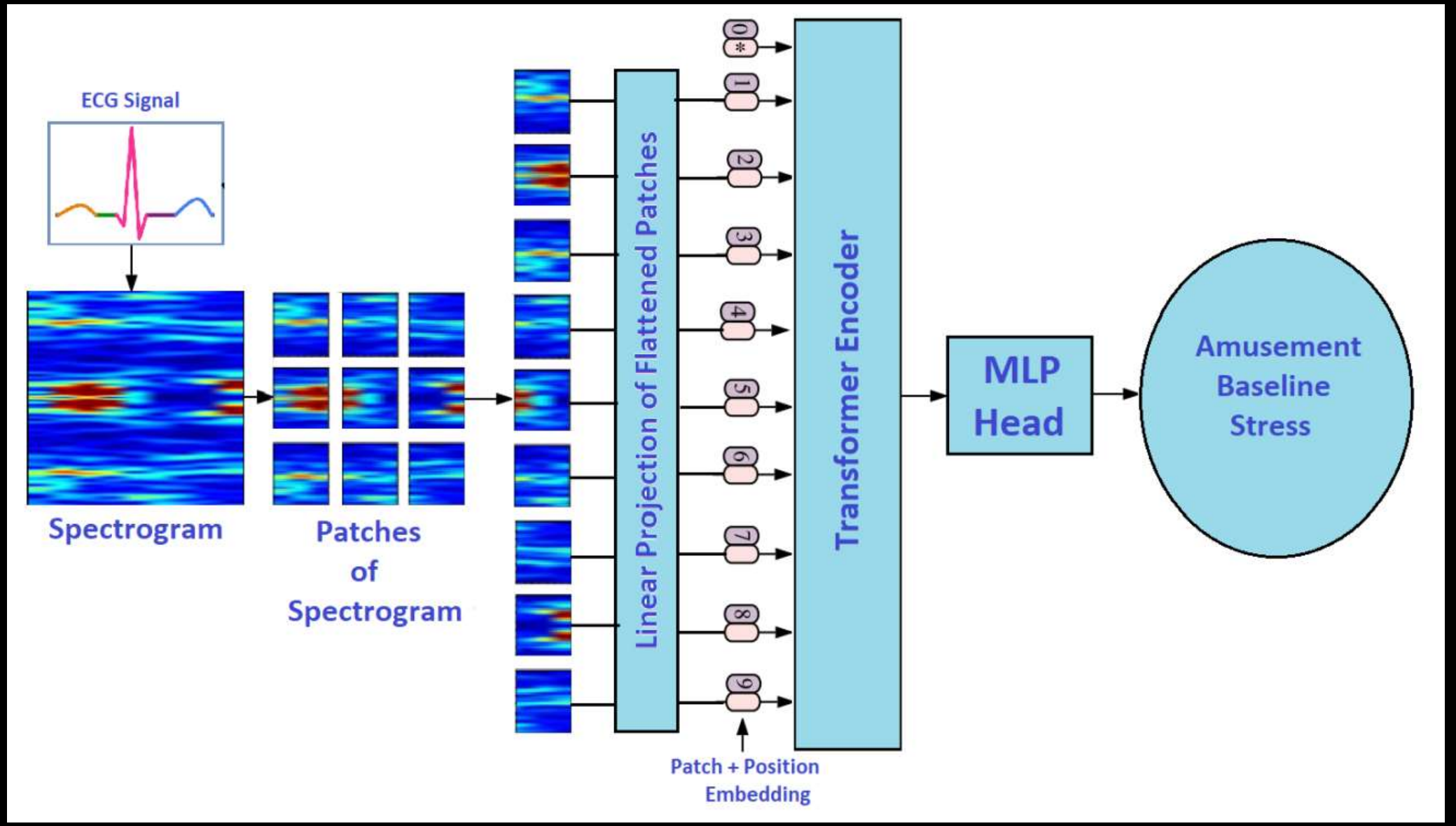
Medium



High

Feature Visualization




ECG Signal
Spectrogram
Patches of Spectrogram
Linear Projection of Flattened Patches
0*
1
2
3
4
5
6
7
8
9
Transformer Encoder
Patch + Position Embedding
MLP Head
Amusement
Baseline
Stress

Input Attention