CV4AFRICA GROUP

VISION TRANSFORMER

Dr. Ahmed Shahin

WHY ???

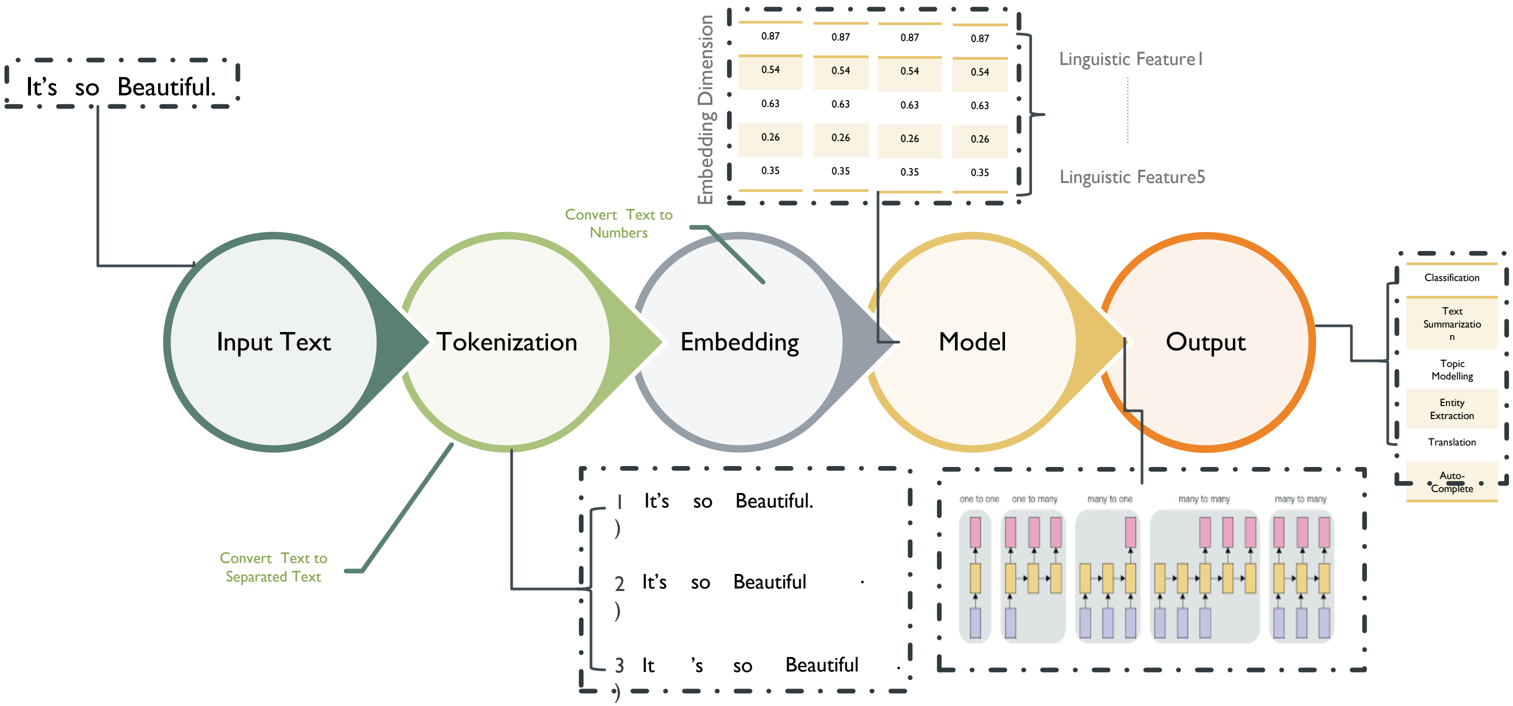
- Most CV Scientist has little experience with NLP, which is the transformers concept.
- Most CV Scientists understand CNN but it's difficult to understand Transformers
- Physical meaning of Transformers is absent.
- Little math experience in Transformers ideas
- Little Implementation experience in Transformers ideas
- Discussion and the update of vision transformers directions

AGENDA

- Introduction to NLP
- Before Transformers in NLP
- Transformers in NLP
- Before Transformers in CV
- Transformers in CV

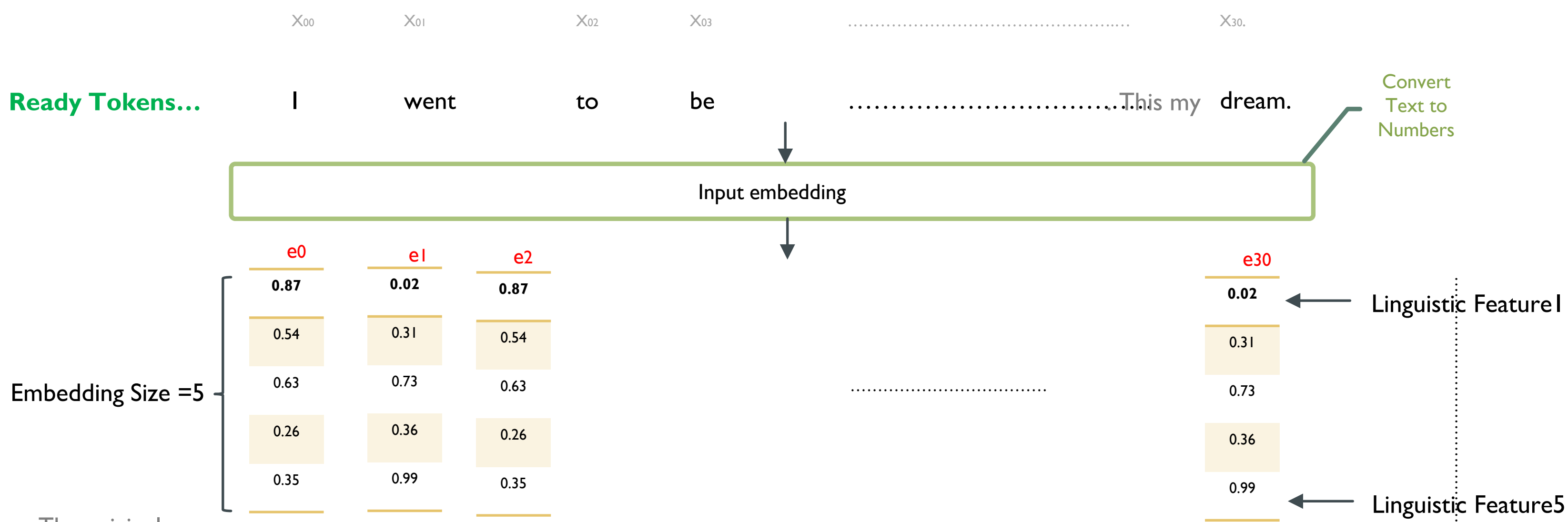
INTRODUCTION TO NLP

NLP Models Pipelines



BEFORE TRANSFORMERS IN NLP

Word Embedding in NLP models



The original transformer **Embedding Size** with 512 vector length

BEFORE TRANSFORMERS IN NLP

Simple Concept, with butterfly effect

Why is the position important in linguistics ????

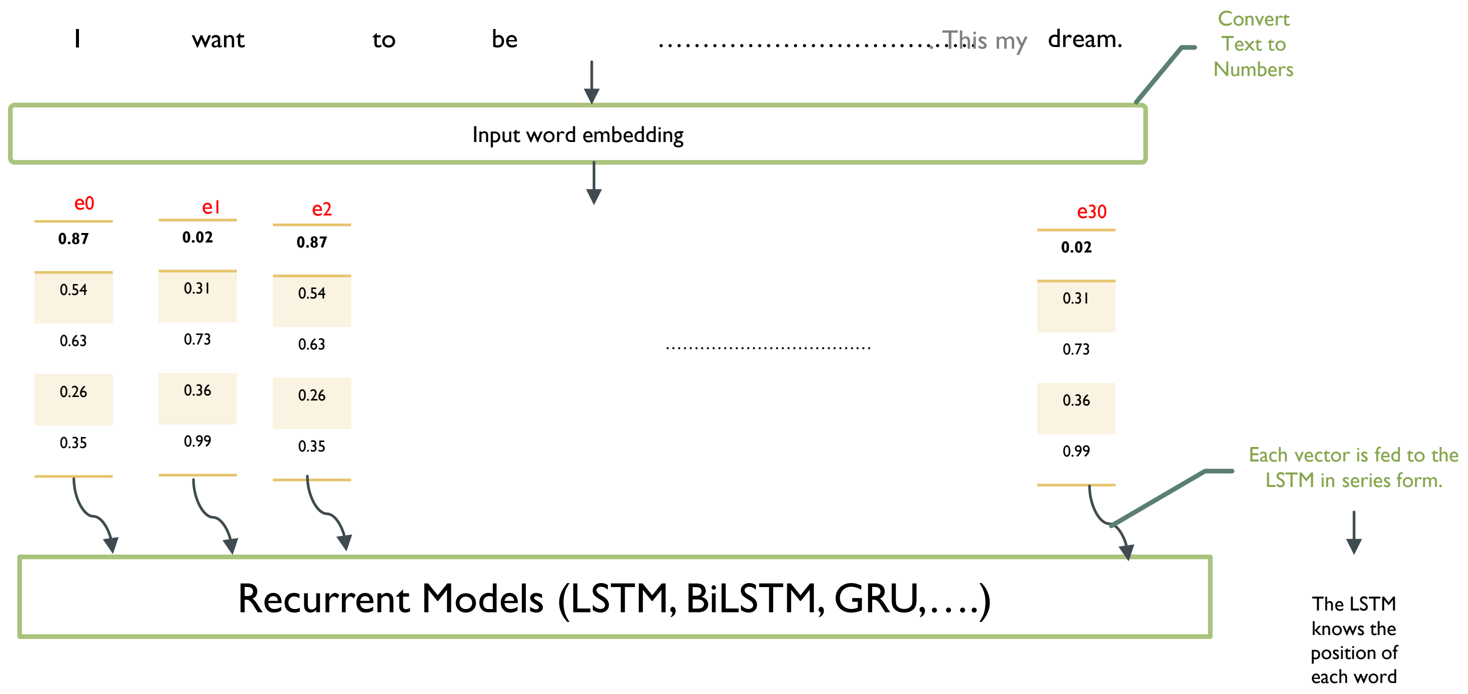
Even though she did not win the award, she was satisfied.

Even though she did win the award, she was not satisfied.

Different positions ➔ different meaning

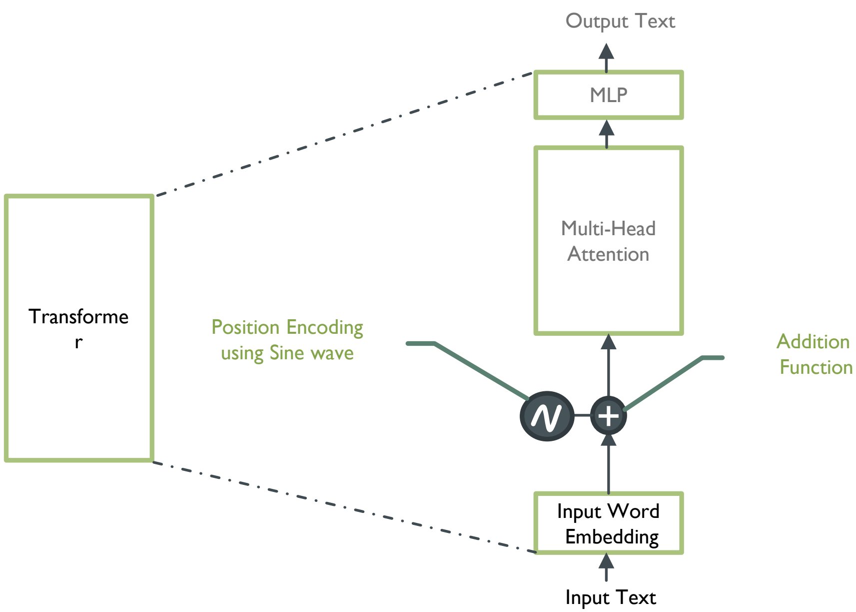
BEFORE TRANSFORMERS IN NLP

How was the embedding happening before?



TRANSFORMERS IN NLP

Transformer Architecture



Attention Is All You Need

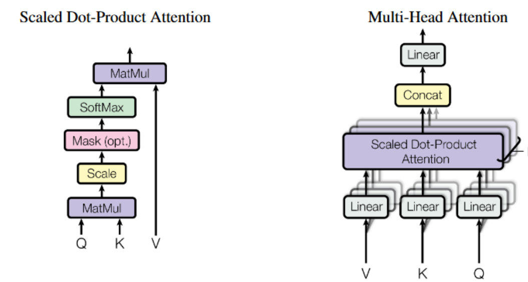
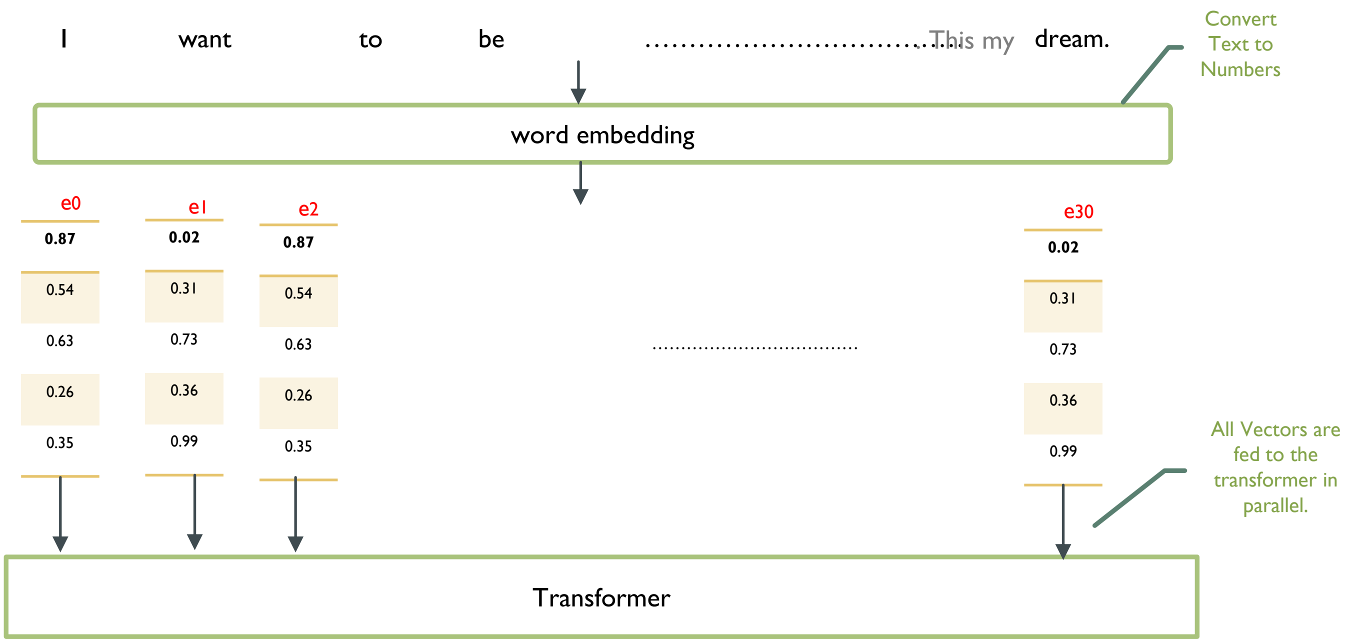


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

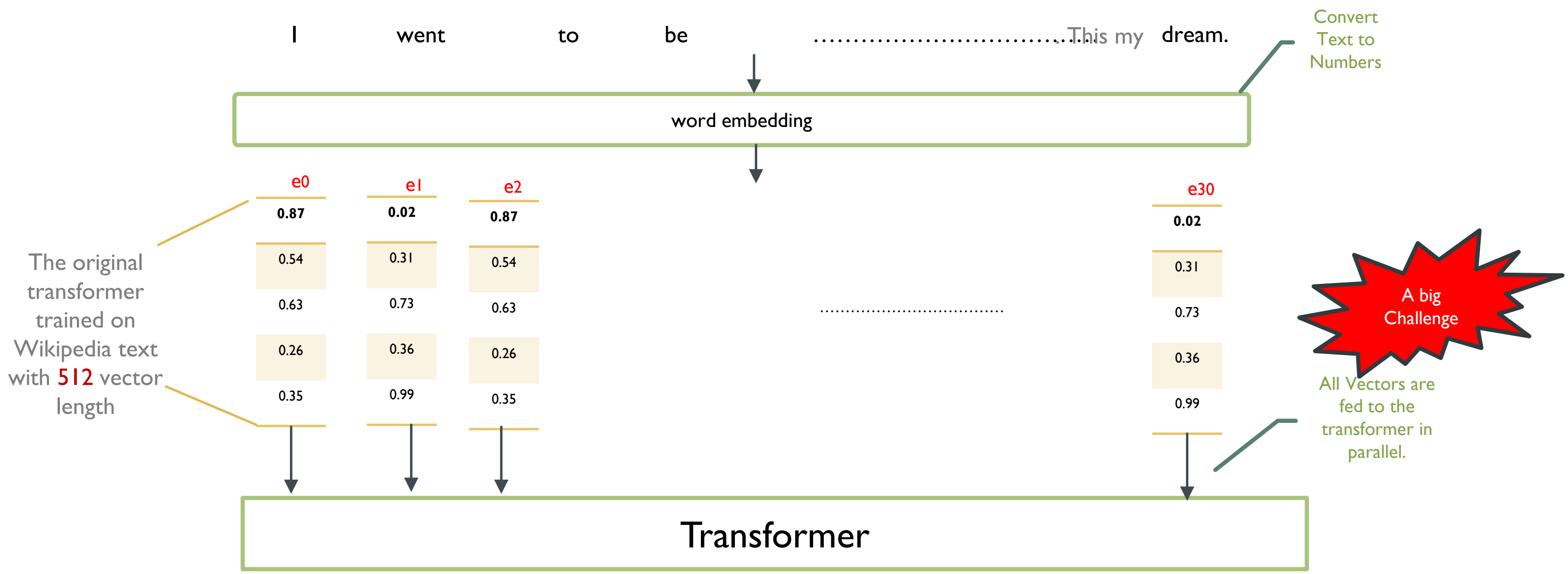
TRANSFORMERS IN NLP

Recurrent Models vs. Transformers



TRANSFORMERS IN NLP

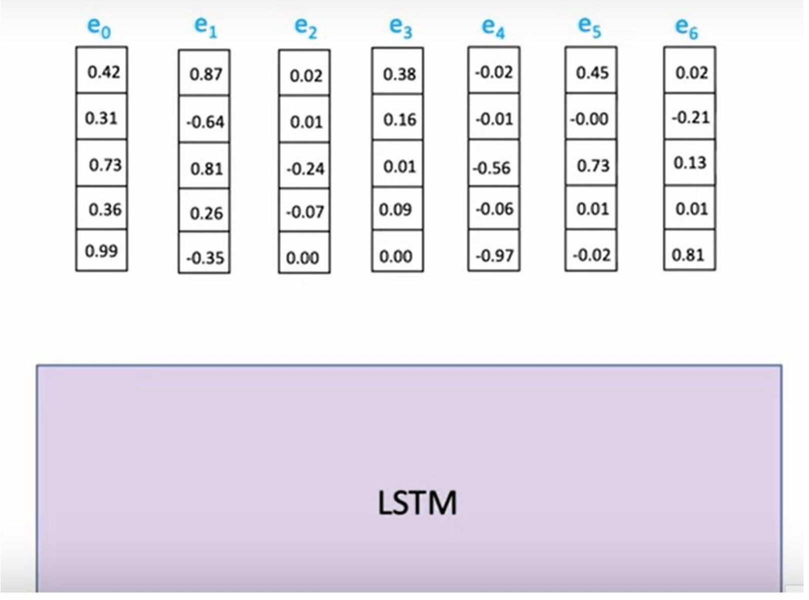
Recurrent Models vs. Transformers



TRANSFORMERS IN NLP

Recurrent Models vs. Transformers

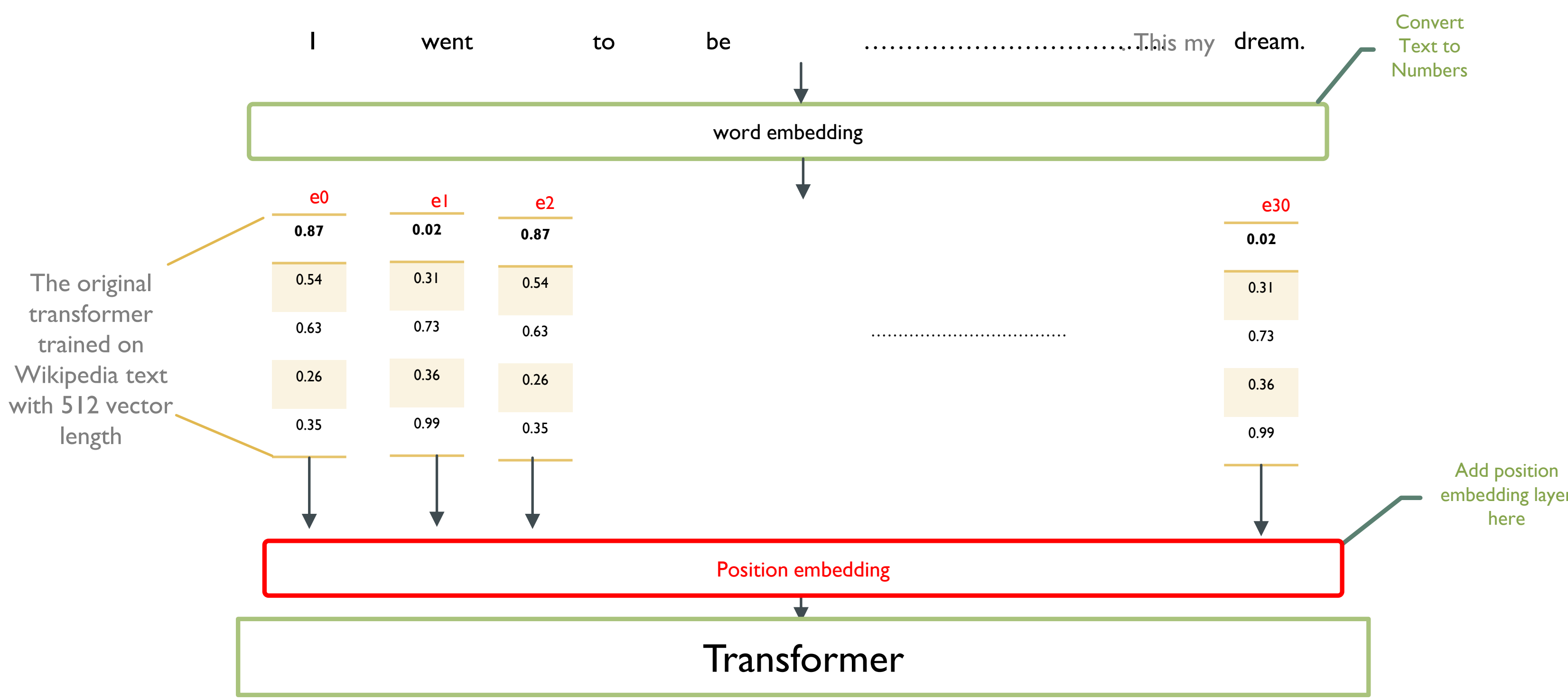
So, Once you fed the input to the LSTM model, the model knew the position of each vector



So, Once you fed the input to the Transformer model, the model lack the position of each vector

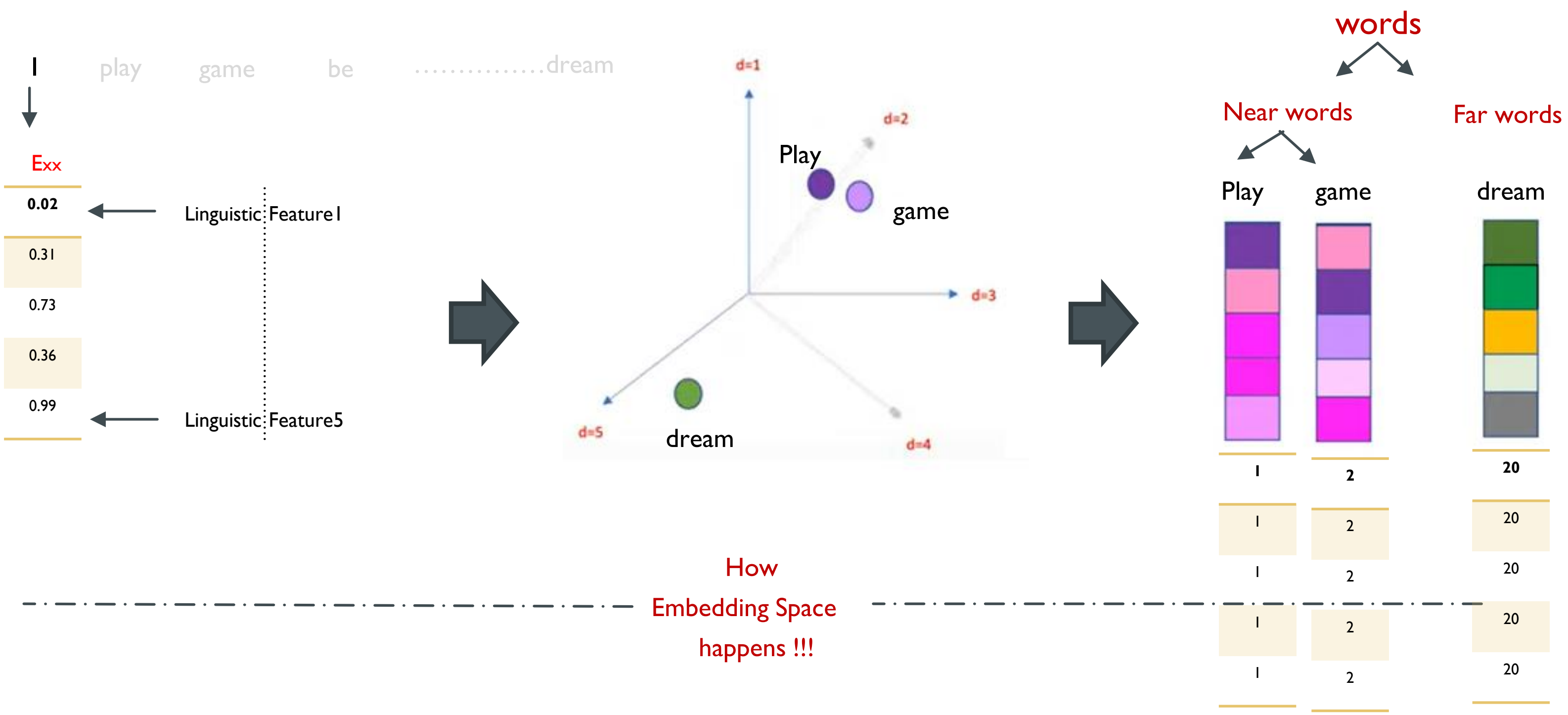
TRANSFORMERS IN NLP

Recurrent Models vs. Transformers



TRANSFORMERS IN NLP

Inside Position embedding space



TRANSFORMERS IN NLP

Inside Position embedding space

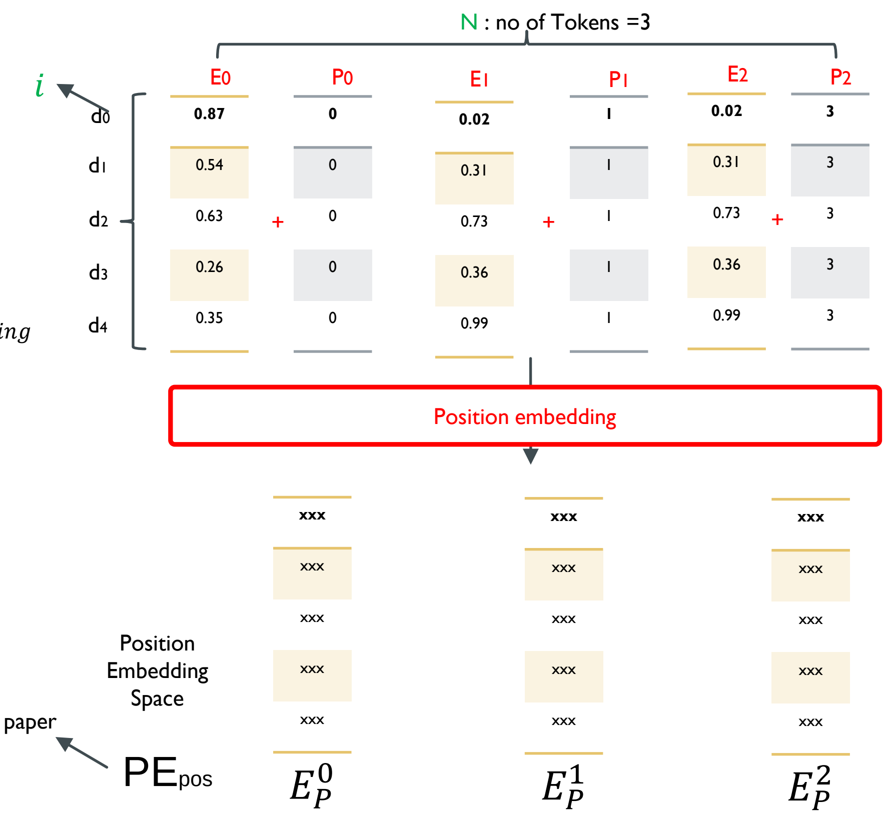
Terms

- If we have a sentence for 3 tokens

i 0,1,2,3
refers to
indices number
“dimension”

d *the size of position embedding* $= 5$

In the original paper



TRANSFORMERS IN NLP

Inside Position embedding space

First Scenario

- If we have a sentence for 3 tokens,
  SOLVE ENCODING BY FIXED INTEGER VALUES

N=3

E0	P0	E1	P1	E3	P3
0.87	0	**0.02**	**1**	**0.02**	**3**
0.54	0	0.31	1	0.31	3
0.63	0	0.73	1	0.73	3
0.26	0	0.36	1	0.36	3
0.35	0	0.99	1	0.99	3

Order 3 is too small to train the language model

TRANSFORMERS IN NLP

Inside Position embedding space

Second Scenario

- If we have a sentence for 30 tokens,
  SOLVE ENCODING BY FIXED INTEGER VALUES

N=30

E0		P0	E1		P1		E30		P30
0.87		**0**	**0.02**		**1**		**0.02**		**30**
0.54		0	0.31		1		0.31		30
0.63	+	0	0.73	+	1	....	0.73	+	30
0.26		0	0.36		1		0.36		30
0.35		0	0.99		1		0.99		30

Order 30 is not applicable,
as it will distort the original info

TRANSFORMERS IN NLP

Inside Position embedding space

Third Scenario

- If we have a sentence for 30 tokens,
  SOLVE ENCODING BY FIXED fraction VALUES

$$P_{pos} = i * \frac{1}{N-1}$$

N=2

E0		P0	E1		P1	E2		P2
0.87		**0**	**0.02**		**0.5**	**0.02**		**1**
0.54			0.31			0.31		
0.63	+		0.73	+		0.73	+	
0.26			0.36			0.36		
0.35			0.99			0.99		

$$P_0 = i * \frac{1}{N-1} = 0 * \frac{1}{3-1} = 0 * \frac{1}{2}$$

$$P_1 = i * \frac{1}{N-1} = 1 * \frac{1}{3-1} = 1 * \frac{1}{2}$$

$$P_2 = i * \frac{1}{N-1} = 2 * \frac{1}{3-1} = 2 * \frac{1}{2}$$

But this encoding will fail to deal with different token sizes in different sentences

TRANSFORMERS IN NLP

N=3

I		play		football	
E0	P0	E1	P1	E2	P2
0.87	**0**	**0.02**	**0.5**	**0.02**	**1**
0.54	0	0.31	0.5	0.31	1
0.63	+ 0	0.73	+ 0.5	0.73	+ 1
0.26	0	0.36	0.5	0.36	1
0.35	0	0.99	0.5	0.99	1

$$P_0 = i * \frac{1}{N-1} = 0 * \frac{1}{3-1} = 0 * \frac{1}{2}$$

$$P_1 = i * \frac{1}{N-1} = 1 * \frac{1}{3-1} = 1 * \frac{1}{2}$$

$$P_2 = i * \frac{1}{N-1} = 2 * \frac{1}{3-1} = 2 * \frac{1}{2}$$

N=4

I		play		a		football	
E0	P0	E1	P1	E2	P2	E3	P3
0.87	**0**	**0.02**	0.3	**0.02**	**0.66**	**0.2**	**1**
0.54	0	0.31	0.3	0.31	0.66	0.31	1
0.63	+ 0	0.73	+ 0.3	0.73	+ 0.66	0.35	+ 1
0.26	0	0.36	0.3	0.36	0.66	0.45	1
0.35	0	0.99	0.3	0.99	0.66	0.91	1

$$P_0 = i * \frac{1}{N-1} = 0 * \frac{1}{4-1} = 0 * \frac{1}{3}$$

$$P_1 = i * \frac{1}{N-1} = 1 * \frac{1}{4-1} = 1 * \frac{1}{3}$$

$$P_2 = i * \frac{1}{N-1} = 2 * \frac{1}{4-1} = 2 * \frac{1}{3}$$

$$P_2 = i * \frac{1}{N-1} = 3 * \frac{1}{4-1} = 3 * \frac{1}{3}$$

The same position in different sentences will be encoded in different way

TRANSFORMERS IN NLP

Inside Position embedding space

Last Scenario

- If we have a sentence for N tokens,
  SOLVE ENCODING BY Sine and Cosine VALUES

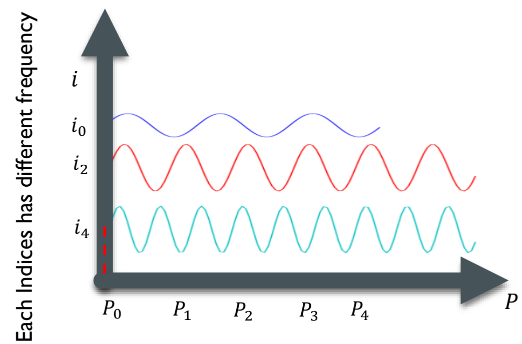
N=4

E0		P0	E1		P1	E2		P2	E3		P3
0.87			**0.02**			**0.02**			**0.02**		
0.54			0.31			0.31			0.31		
0.63	+		0.73	+		0.73	+		0.73	+	
0.26			0.36			0.36			0.36		
0.35			0.99			0.99			0.99		

$$P_{pos,2i} = Sin(\frac{P}{10,000(\frac{2i}{d})})$$

For even positions

Fixed Value assumed in Attention is all u need! paper



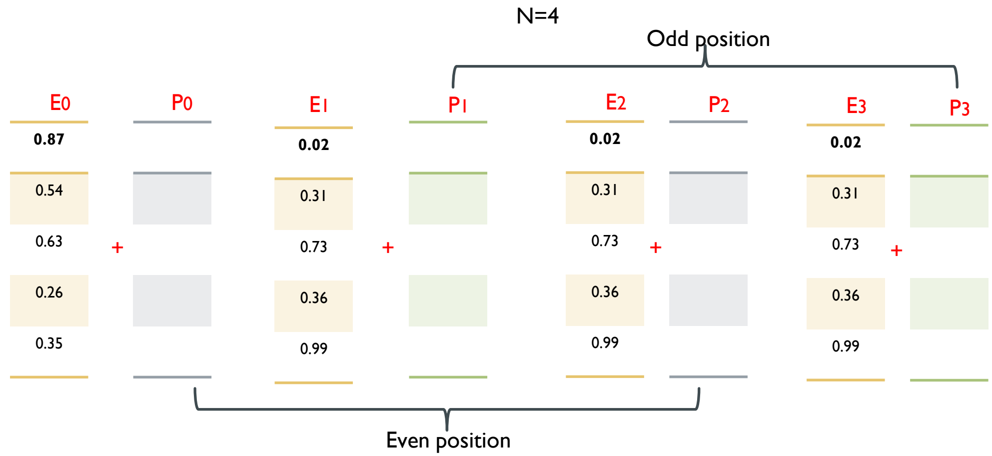
Each word position has different X-axis value

TRANSFORMERS IN NLP

Inside Position embedding space

Last Scenario

- If we have a sentence for N tokens,
  SOLVE ENCODING BY Sine and Cosine VALUES



For even positions

$$P_{pos,2i} = Sin(\frac{P}{10{,}000^{(\frac{2i}{d})}})$$

For odd positions

$$P_{pos,2i+1} = cos(\frac{P}{10{,}000^{(\frac{2i}{d})}})$$

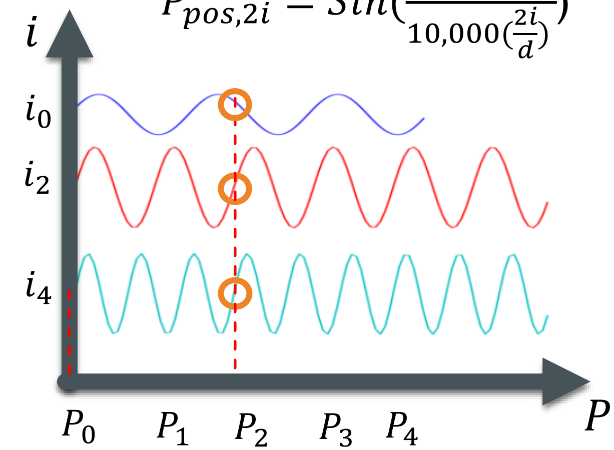
Fixed Value assumed in Attention is all u need! paper

TRANSFORMERS IN NLP

Same position value, different indices

	P_0	P_1	P_2	P_3	P_4
	I	play	football	with	ahmed
	P 0				
i_0			0		
i_1			0.00049		
i_2			0.000249		
i_3			0.000166		
i_4			0.0001249		

$$P_{pos,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})})$$



$$P_{2,2*0} = Sin(\frac{P}{10{,}000(\frac{2i}{d})}) = Sin(\frac{2}{10{,}000(\frac{2*0}{5})}) = 0$$

$$P_{2,2*1} = Sin(\frac{P}{10{,}000(\frac{2i}{d})}) = Sin(\frac{2}{10{,}000(\frac{2*1}{5})}) = 0.00049$$

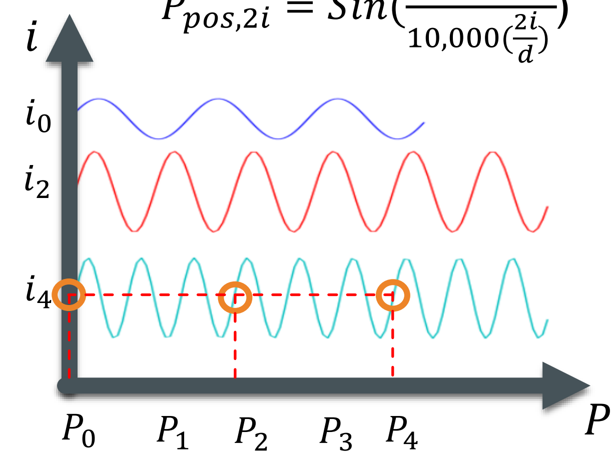
$$P_{4,2*2} = Sin(\frac{P}{10{,}000(\frac{2i}{d})}) = Sin(\frac{2}{10{,}000(\frac{2*2}{5})}) = 0.000249$$

TRANSFORMERS IN NLP

Same indices value, different position

	P_0	P_1	P_2	P_3	P_4
	I	play	football	with	ahmed
	P0	P1	P2	P3	P4
i_0	0				
i_1	0				
i_2	0				
i_3	0				
i_4	0		0.0001249		0.000249

$$P_{pos,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})})$$



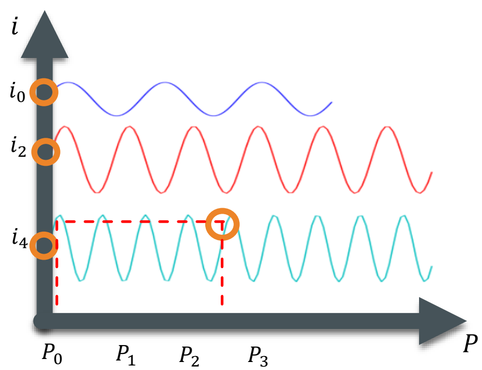
$$P_{0,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})}) = Sin(\frac{0}{10{,}000(\frac{2*4}{5})}) = 0$$

$$P_{2,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})}) = Sin(\frac{2}{10{,}000(\frac{2*4}{5})}) = 0.0001249$$

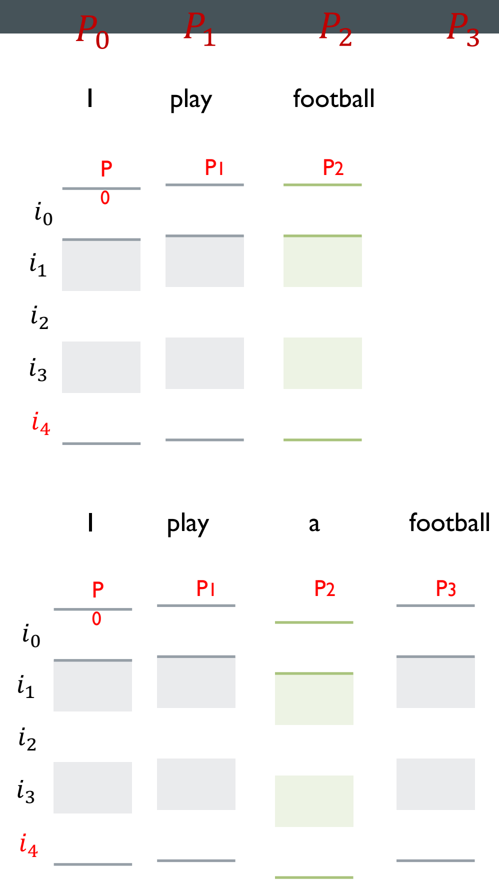
$$P_{4,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})}) = Sin(\frac{4}{10{,}000(\frac{2*4}{5})}) = 0.000249$$

TRANSFORMERS IN NLP

Different Sentences



$$P_{pos,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})})$$

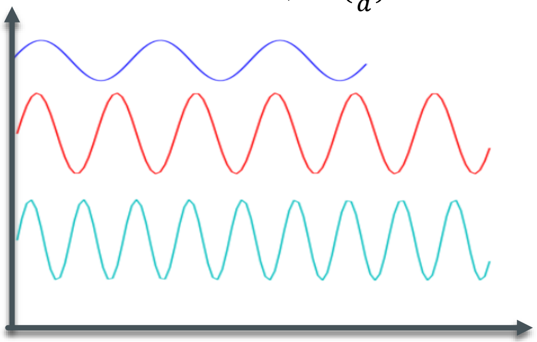


The same position in different sentences will be encoded in the same way

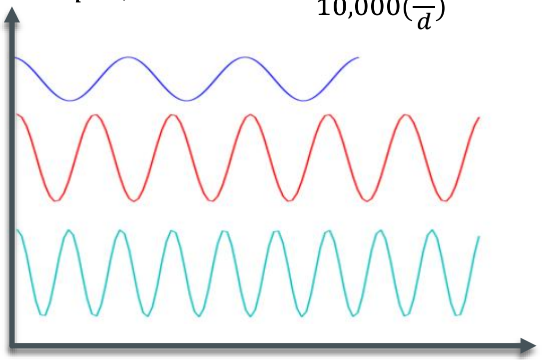
TRANSFORMERS IN NLP

Sine and cosine for positional embedding

$$P_{pos,2i} = Sin(\frac{P}{10{,}000^{(\frac{2i}{d})}})$$



$$P_{pos,2i+1} = cos(\frac{P}{10{,}000^{(\frac{2i}{d})}})$$

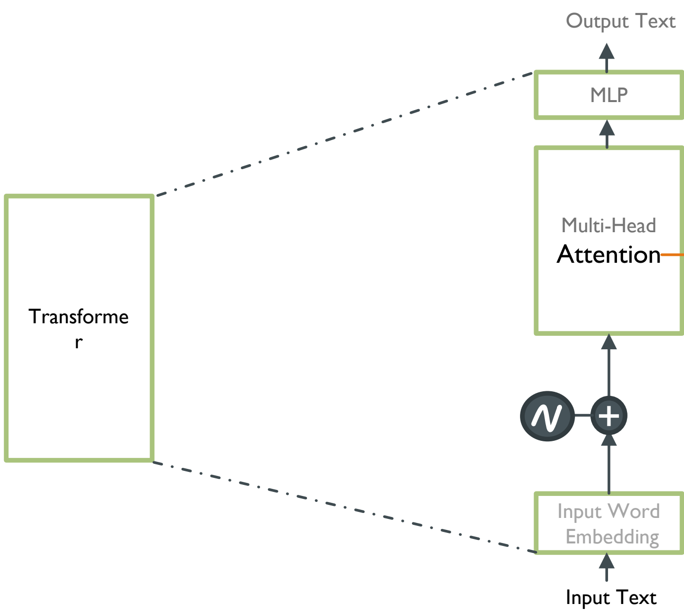


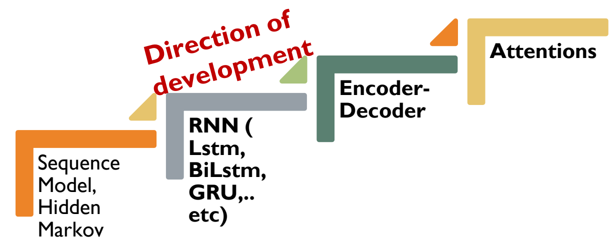
Advantages of position encoding using sine and cosine

1. The sine and cosine functions have values in [-1, 1], which keeps the values of the positional encoding matrix in a normalized range.
2. As the sinusoid for each position is different; we have a unique way of encoding each position.
3. We have a way of measuring or quantifying the similarity between different positions, enabling us to encode relative positions of words.

TRANSFORMERS IN NLP

Attention in NLP



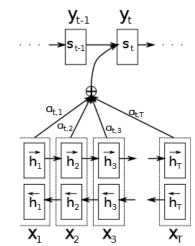


Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

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



The attention mechanism is proposed as an improvement **after the encoder-decoder-based neural machine translation system** in natural language processing (NLP). Later, this mechanism, or its variants, was used in other applications, including **computer vision**, speech processing, etc.

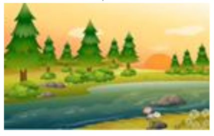
TRANSFORMERS IN NLP

Attention in NLP

- The Simple Question here how to know the difference between each bank in this sentence ?

Answer is the Context of each word

He went to the bank and learned of his empty account, after which he went to a river bank and cried



He went to the bank and learned of his empty account, after which he went to a river bank and cried.

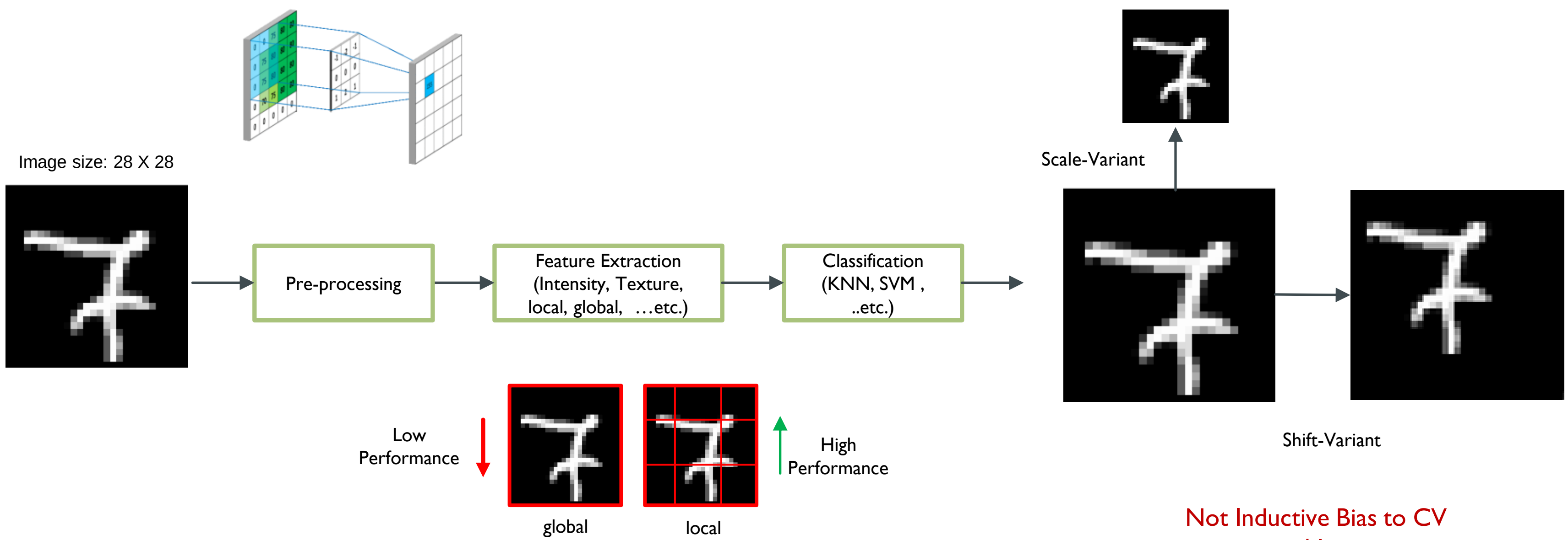
To achieve this numerically in computers, You are talking about self-attention

He went to the bank and learned of his empty account, after which he went to a river bank and cried

BEFORE TRANSFORMERS IN CV

From Traditional Systems to CNN revolution

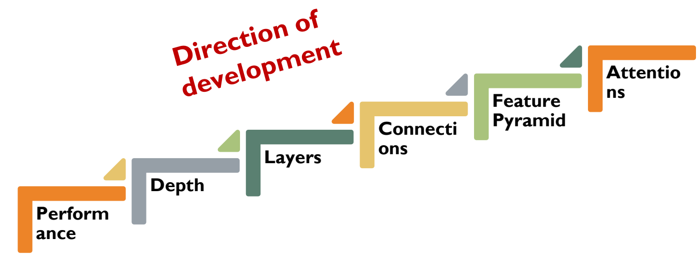
Most of the pre-processing step was based on the **convolution process (**different filter values ➔ different output



Not Inductive Bias to CV problems

BEFORE TRANSFORMERS IN CV

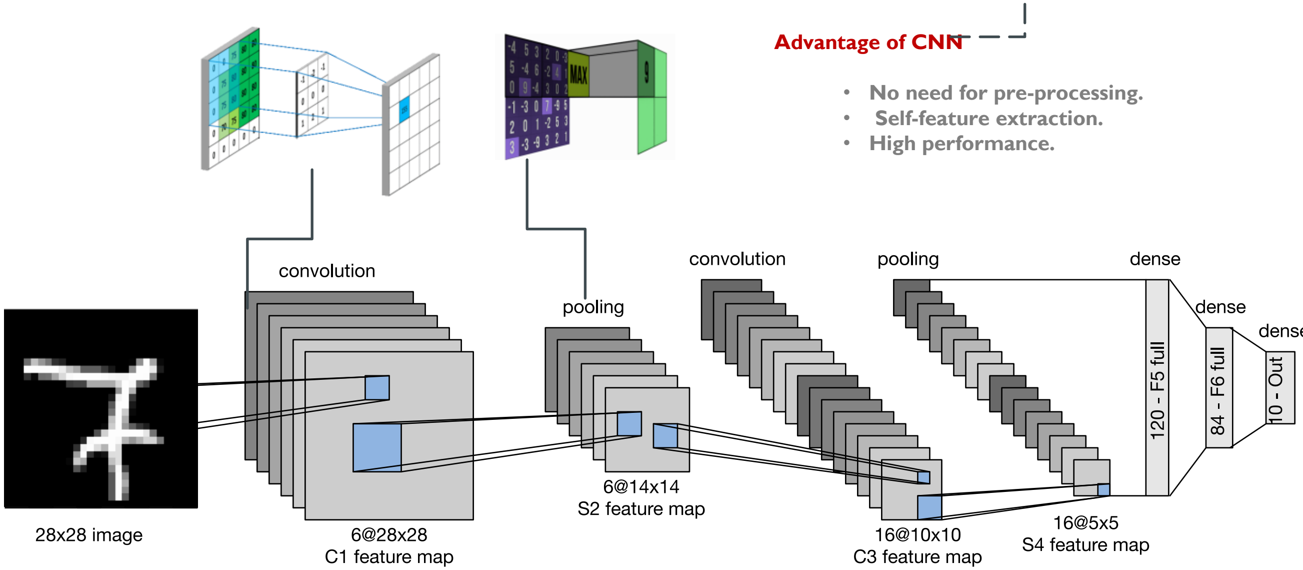
From Traditional Systems to CNN revolution



Most of the pre-processing step was based on the **convolution process (**different filter values ➔ different output

Advantage of CNN

- **No need for pre-processing.**
- **Self-feature extraction.**
- **High performance.**



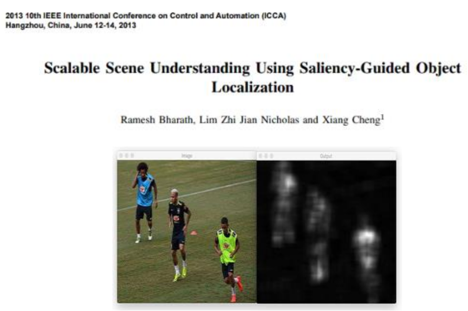
Inductive Bias to CV problems

BEFORE TRANSFORMERS IN CV

Attention in Image ➔ Where to focus in the Image

Attention in Image

The attention source is simple calculations


2013 10th IEEE International Conference on Control and Automation (ICCA)
Hangzhou, China, June 12-14, 2013

Scalable Scene Understanding Using Saliency-Guided Object Localization

Ramesh Bharath, Lim Zhi Jian Nicholas and Xiang Cheng[1]

BEFORE TRANSFORMERS IN CV

Attention in Image ➔ Where to focus in the Image

Attention in Image

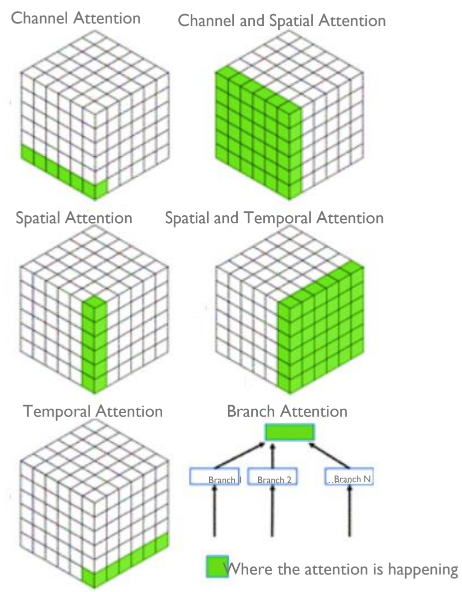
2013 10th IEEE International Conference on Control and Automation (ICCA)
Hangzhou, China, June 12-14, 2013

Scalable Scene Understanding Using Saliency-Guided Object Localization

Ramesh Bharath, Lim Zhi Jian Nicholas and Xiang Cheng[1]

After CNN
Attention in Feature Maps

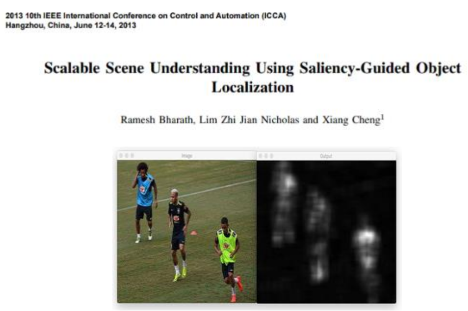
The attention source is the features maps
Now, we call this self attention



BEFORE TRANSFORMERS IN CV

Attention in Image ➔ Where to focus in the Image

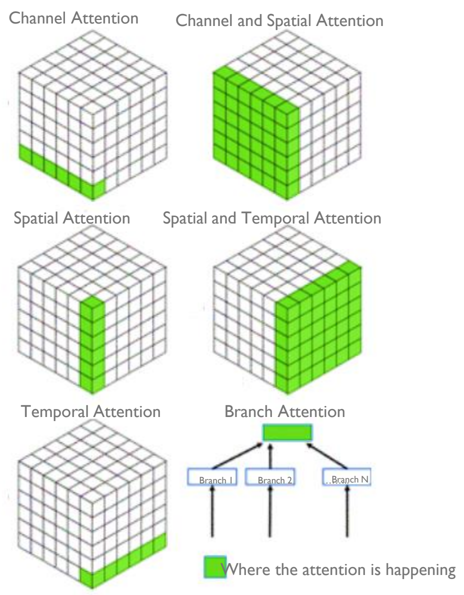
Attention in Image


2013 10th IEEE International Conference on Control and Automation (ICCA)
Hangzhou, China, June 12-14, 2013

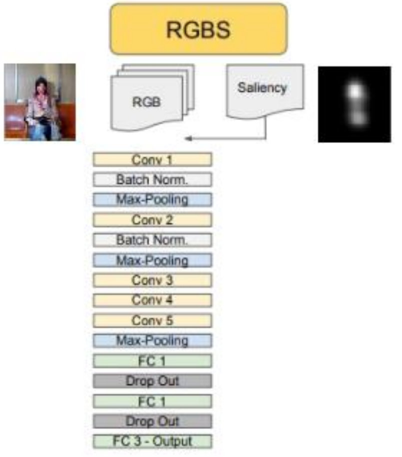
Scalable Scene Understanding Using Saliency-Guided Object Localization

Ramesh Bharath, Lim Zhi Jian Nicholas and Xiang Cheng[1]

After CNN
Attention in Feature Maps

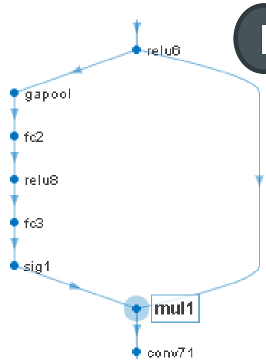


Hybrid Attention The attention source is from another calculations out of the network



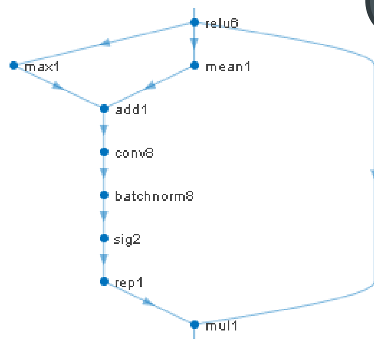
[Channel wise attention or squeeze excitation]

1



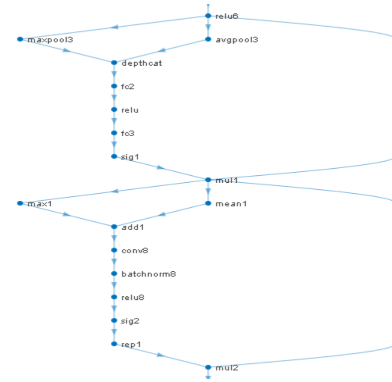
[Spatial attention]

2



[Spatial followed by Channel attention]

3

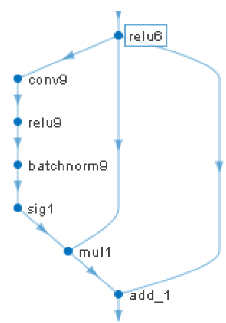


[Simplest:Channel spatial attention]

4

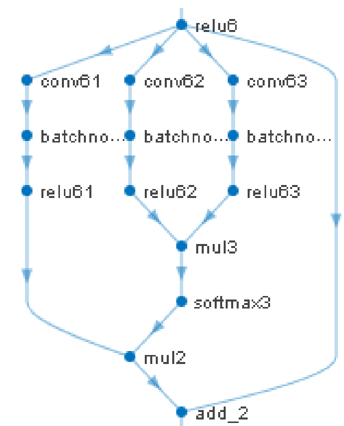
[Simplest:Channel spatial attention with Residual]

5



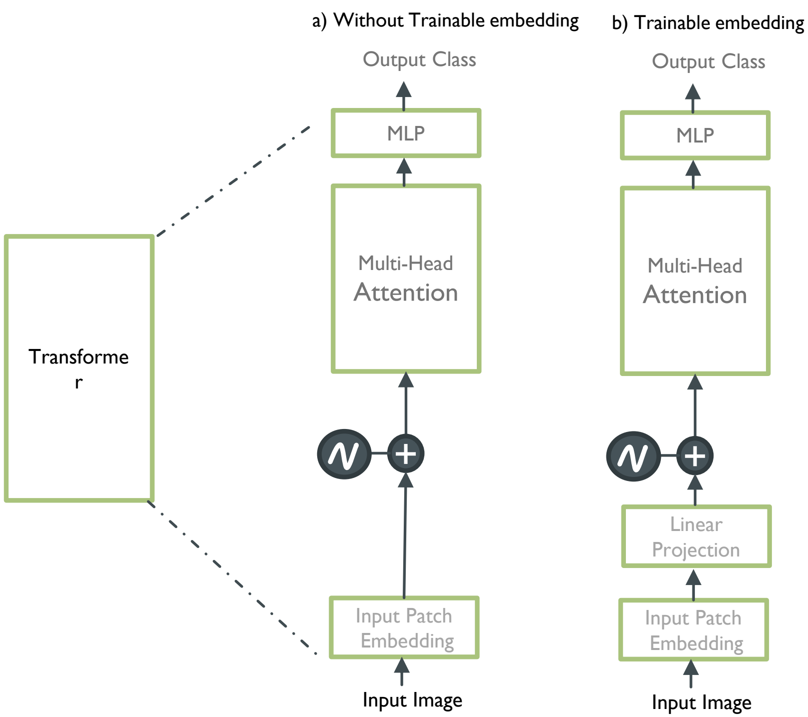
[Q,U,V] attention

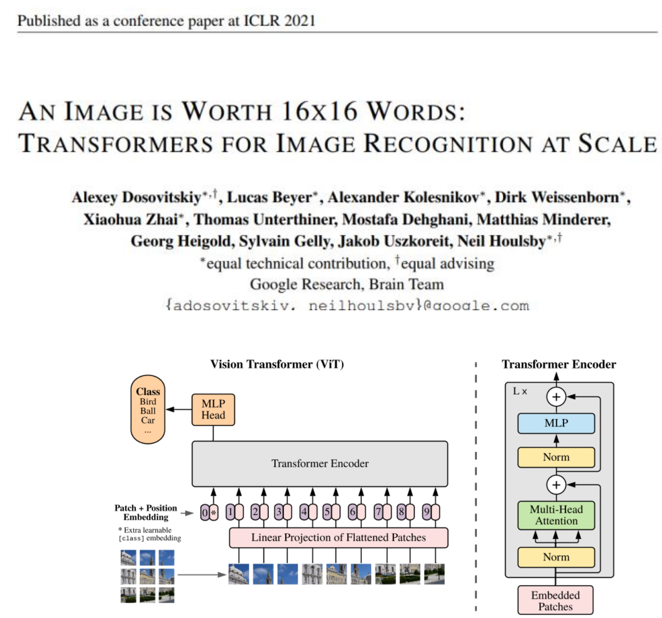
6



TRANSFORMERS IN CV

With non- Trainable parameters (No linear projection) in embedding layer





Published as a conference paper at ICLR 2021

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

TRANSFORMERS IN CV

Simple Concept, with butterfly effect

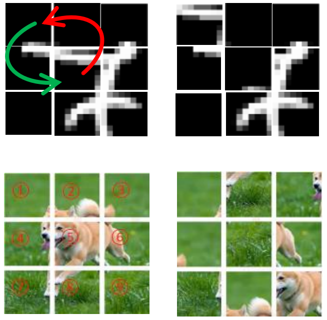
Why is the position important in linguistics ????

Even though she did not win the award, she was satisfied.

Even though she did win the award, she was not satisfied.

Different positions → different meaning

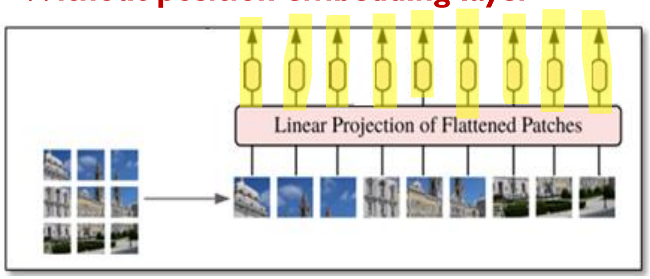
The same issue in CV

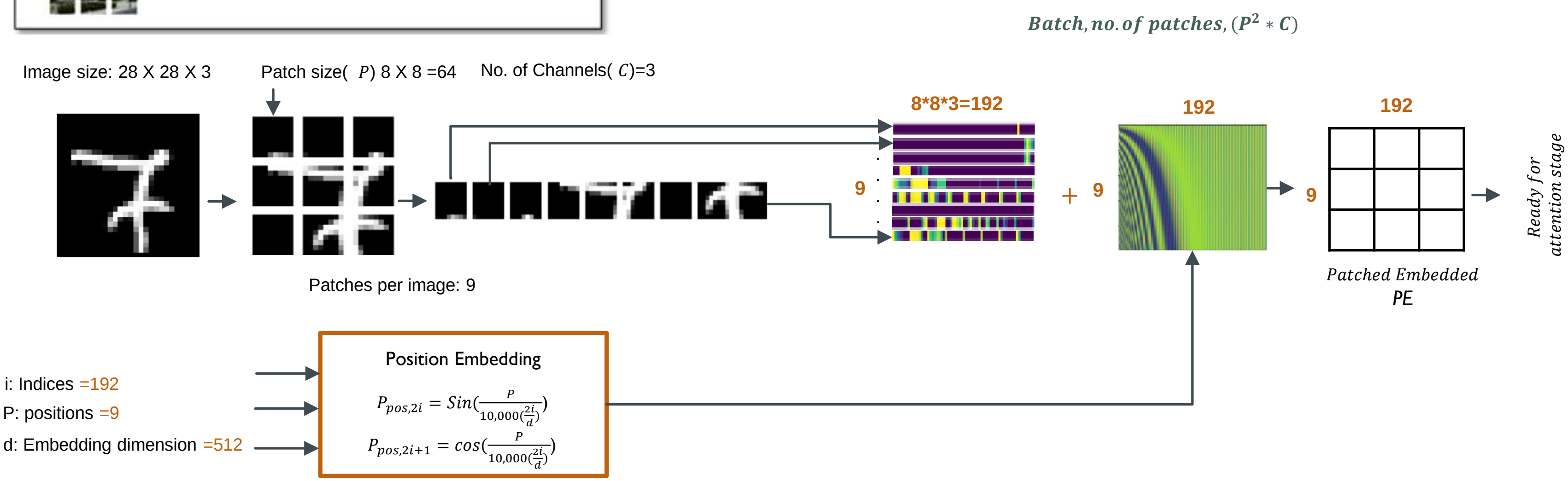


Different positions → different meaning

TRANSFORMERS IN CV

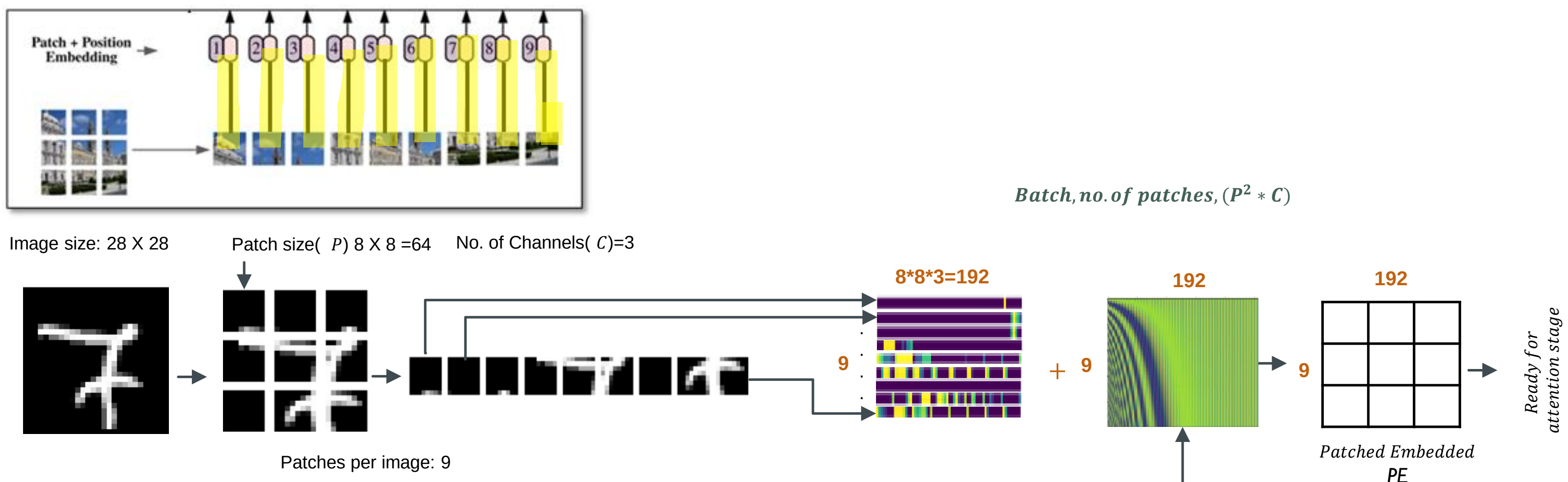
Without position embedding layer





TRANSFORMERS IN CV

Without Trainable parameters in embedding layer



i: Indices =192

P: positions =9

d: Embedding dimension =512

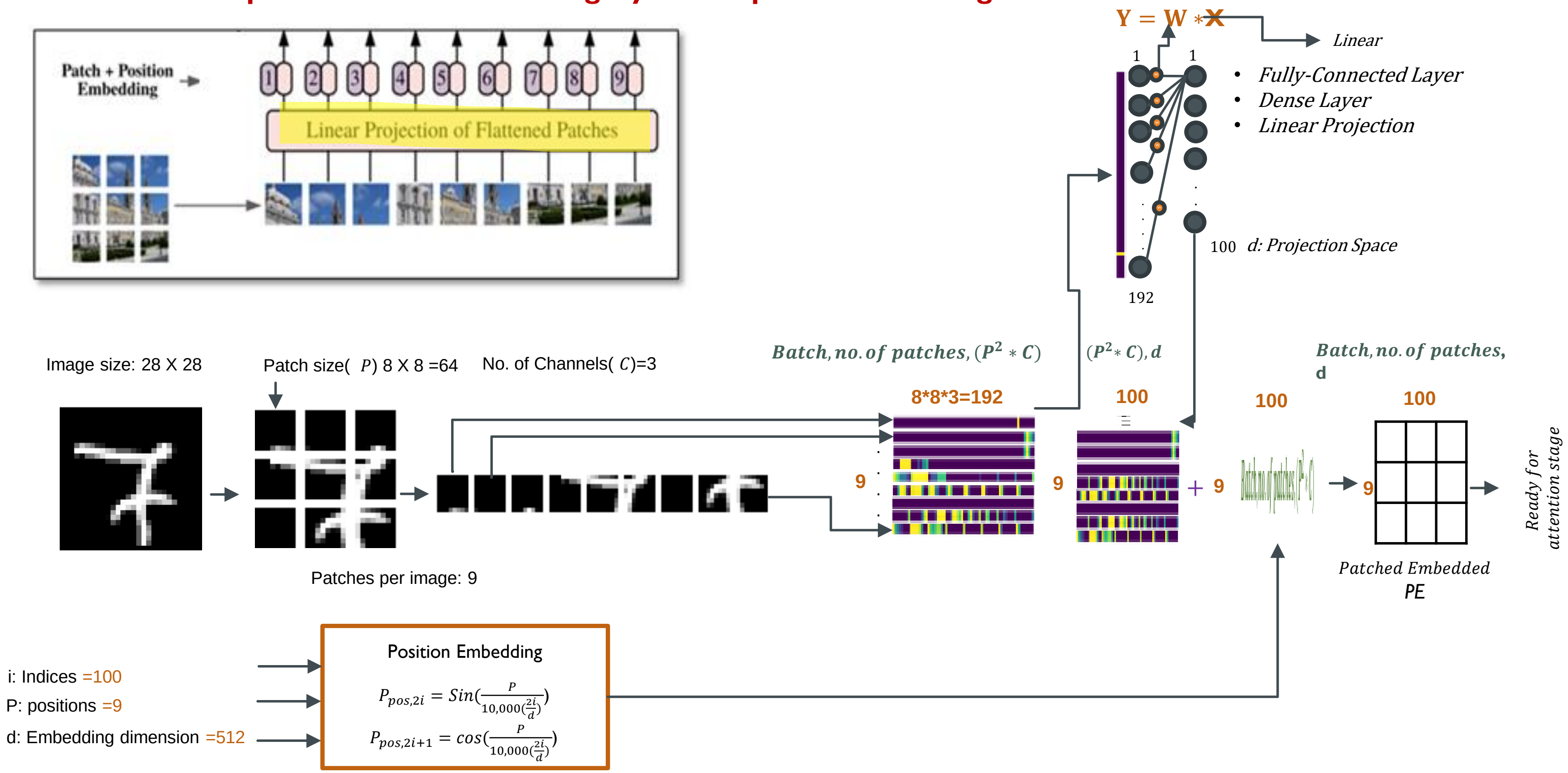
Position Embedding

$$P_{pos,2i} = Sin(\frac{P}{10{,}000(\frac{2i}{d})})$$

$$P_{pos,2i+1} = cos(\frac{P}{10{,}000(\frac{2i}{d})})$$

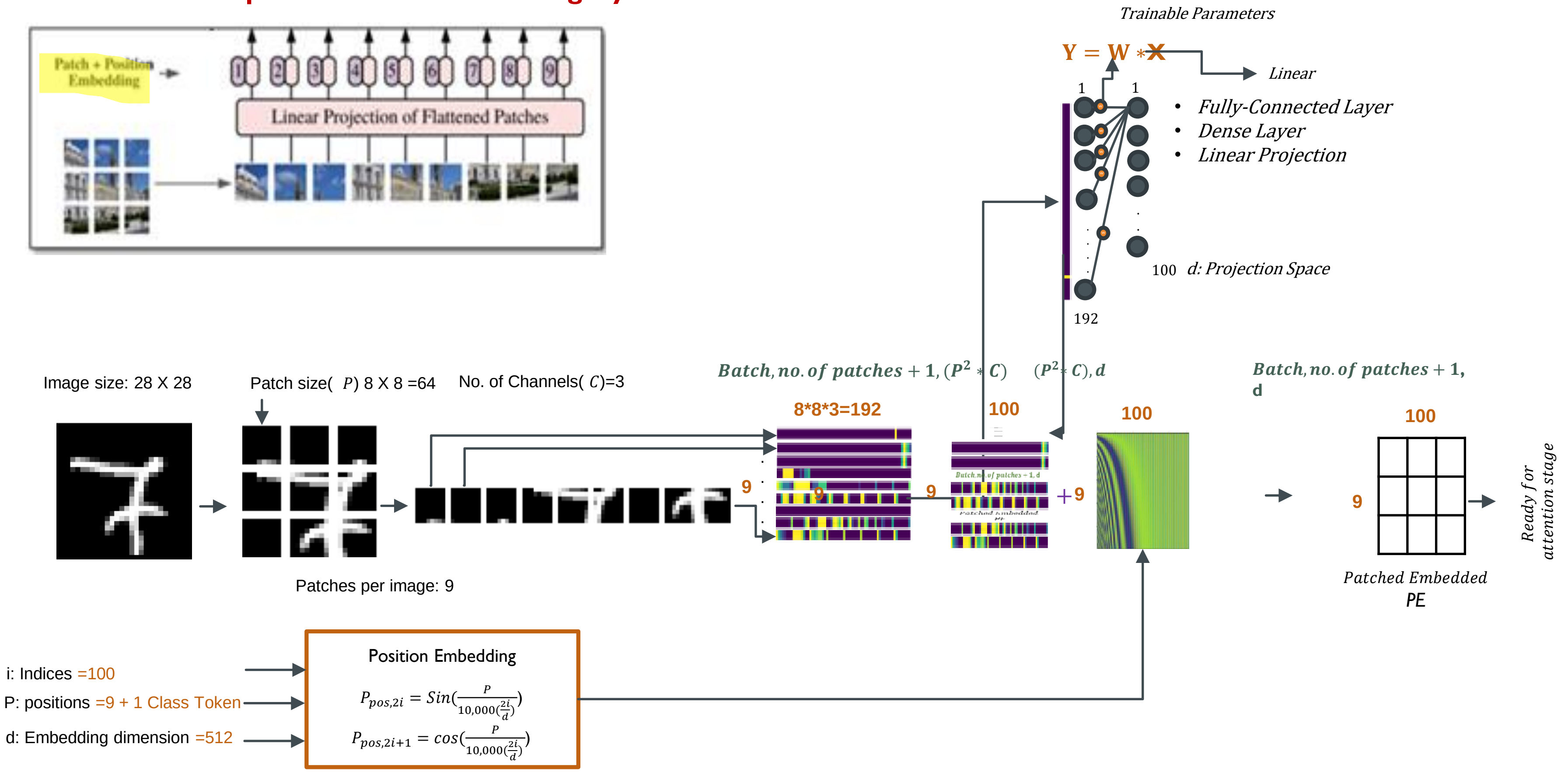
TRANSFORMERS IN CV

With Trainable parameters in embedding layer with position encoding



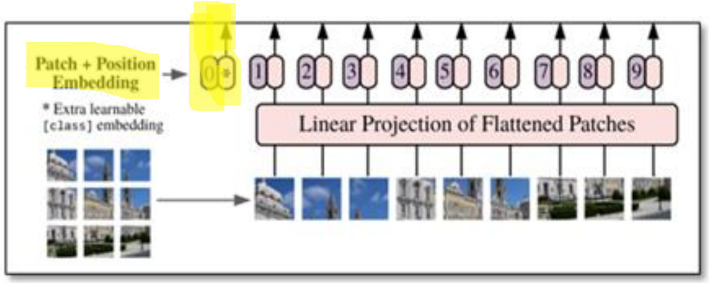
TRANSFORMERS IN CV

With Trainable parameters in embedding layer



TRANSFORMERS IN CV

With Trainable parameters in embedding layer and CLASS TOKEN

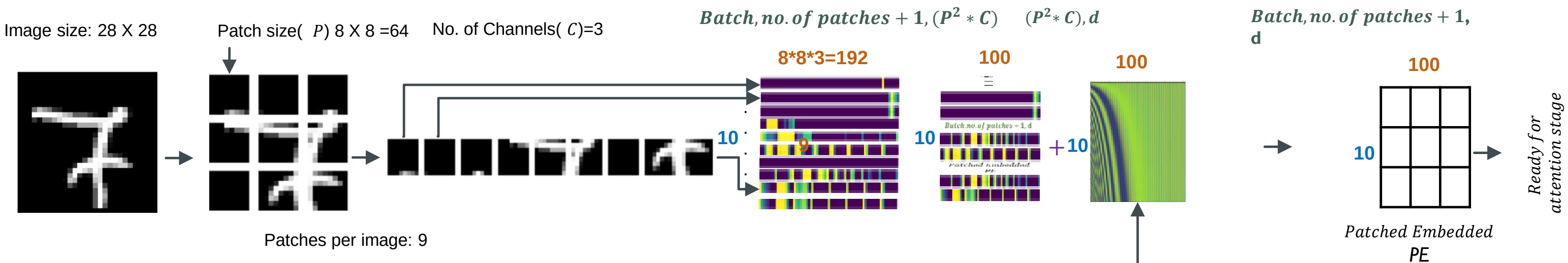


Class Token is a randomly initialized vector embedded at the beginning of your input sequence within the patches token. *It has no relation with classes.*

As an example, if you have 9 patches for an input image with a projection dimension of 100 units. The total input matrix will be (9+1)10 * 100

Why it was useful?

So, at first, it doesn't contain any helpful information on its own. However, the Class Token is able to accumulate information from the other tokens in the sequence the deeper and more layers the Transformer is. When the Vision Transformer finally performs the final classification of the sequence, it uses an MLP head which only looks at data from the last layer's Class Token and no other information. *This operation suggests that the Class Token is a placeholder data structure that's used to store information that is extracted from other tokens in the sequence.* By allocating an empty token for this procedure, it seems like the Vision Transformer makes it less likely to bias the final output towards or against any single one of the other individual tokens.



i: Indices =100

P: positions =9 + 1 Class Token

d: Embedding dimension =512

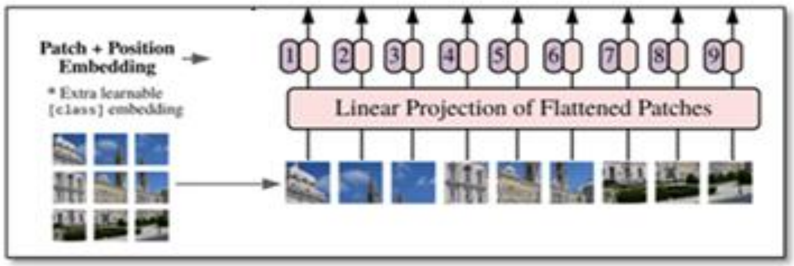
Position Embedding

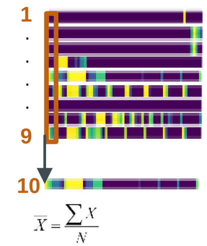
$$P_{pos,2i} = Sin\left(\frac{P}{10{,}000^{\left(\frac{2i}{d}\right)}}\right)$$

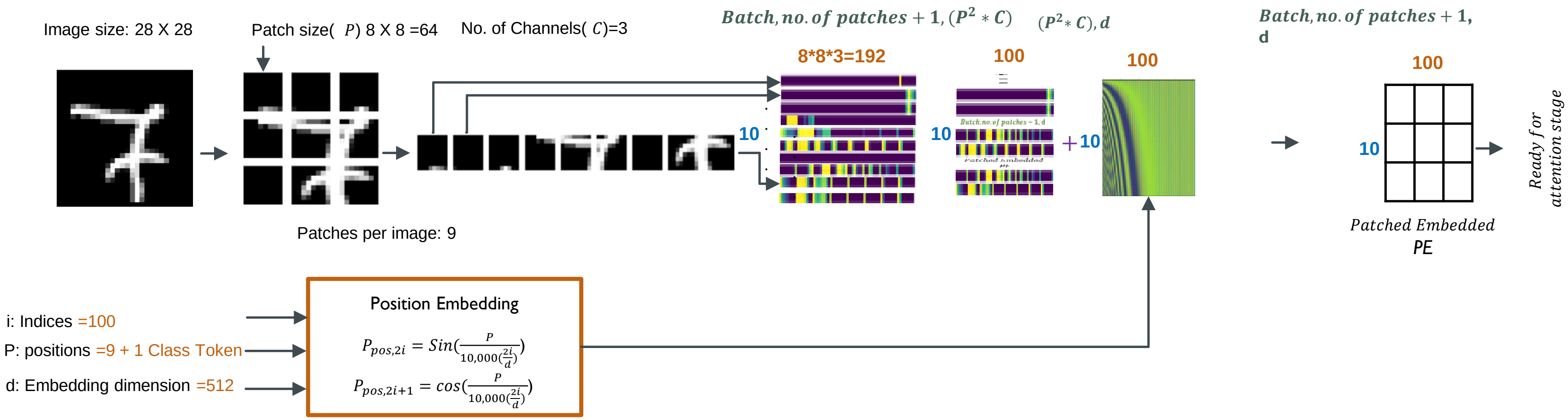
$$P_{pos,2i+1} = cos\left(\frac{P}{10{,}000^{\left(\frac{2i}{d}\right)}}\right)$$

With Trainable parameters in embedding layer and GAP

GAP is the global average pooling of all features of all tokens. Assumption that the GAP is computed at first (however, it can be before MLP head)







MODEL SETUP EXPERIMENT

Model Variants. We base ViT configurations on those used for BERT (Devlin et al., 2019), as summarized in Table 1. The "Base" and "Large" models are directly adopted from BERT and we add the larger "Huge" model. In what follows we use brief notation to indicate the model size and the input patch size: for instance, ViT-L/16 means the "Large" variant with 16×16 input patch size. Note that the Transformer's sequence length is inversely proportional to the square of the patch size, thus models with smaller patch size are computationally more expensive.

For the baseline CNNs, we use ResNet (He et al., 2016), but replace the Batch Normalization layers (Ioffe & Szegedy, 2015) with Group Normalization (Wu & He, 2018), and used standardized convolutions (Qiao et al., 2019). These modifications improve transfer (Kolesnikov et al., 2020), and we denote the modified model "ResNet (BiT)". For the hybrids, we feed the intermediate feature maps into ViT with patch size of one "pixel". To experiment with different sequence lengths, we either (i) take the output of stage 4 of a regular ResNet50 or (ii) remove stage 4, place the same number of layers in stage 3 (keeping the total number of layers), and take the output of this extended stage 3. Option (ii) results in a 4x longer sequence length, and a more expensive ViT model.

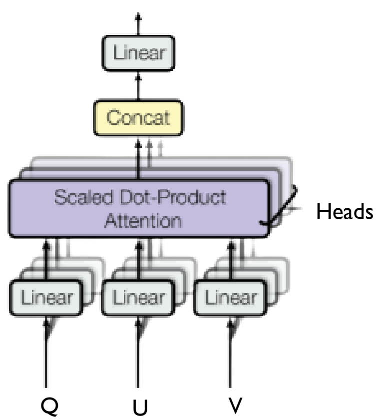
Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

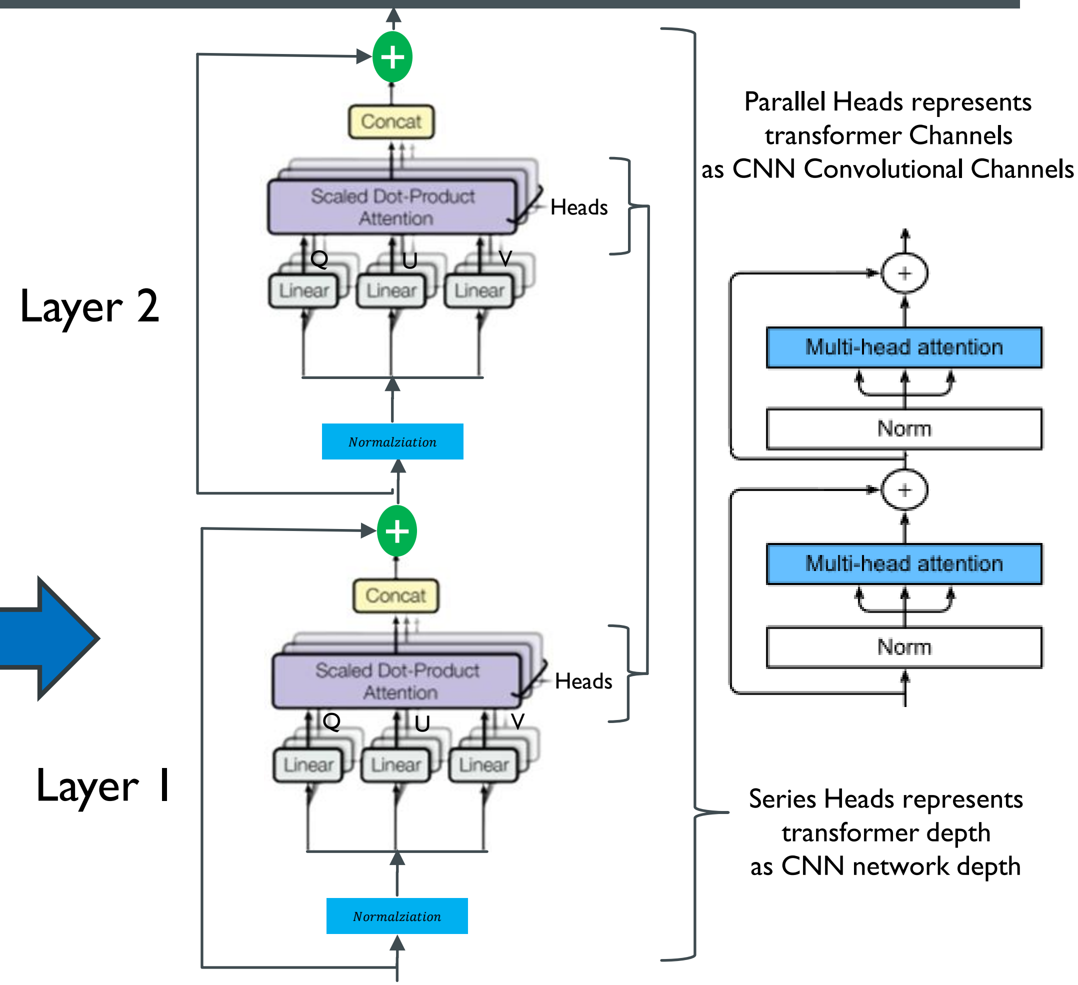
Table 1: Details of Vision Transformer model variants.

TRANSFORMERS IN CV

Transformer Dimensions and parameters

- Heads in Parallel
- Heads in Series





POSITION EMBEDDING EXPERIMENT

D.4 Positional Embedding

We ran ablations on different ways of encoding spatial information using positional embedding. We tried the following cases:

- Providing no positional information: Considering the inputs as a *bag of patches*.
- 1-dimensional positional embedding: Considering the inputs as a sequence of patches in the raster order (default across all other experiments in this paper).
- 2-dimensional positional embedding: Considering the inputs as a grid of patches in two dimensions. In this case, two sets of embeddings are learned, each for one of the axes, X-embedding, and Y-embedding, each with size $D/2$. Then, based on the coordinate on the path in the input, we concatenate the X and Y embedding to get the final positional embedding for that patch.
- Relative positional embeddings: Considering the relative distance between patches to encode the spatial information as instead of their absolute position. To do so, we use 1-dimensional Relative Attention, in which we define the relative distance all possible pairs of patches. Thus, for every given pair (one as query, and the other as key/value in the attention mechanism), we have an offset $p_q - p_k$, where each offset is associated with an embedding. Then, we simply run extra attention, where we use the original query (the content of query), but use relative positional embeddings as keys. We then use the logits from the relative attention as a bias term and add it to the logits of the main attention (content-based attention) before applying the softmax.

In addition to different ways of encoding spatial information, we also tried different ways of incorporating this information in our model. For the 1-dimensional and 2-dimensional positional embeddings, we tried three different cases: (1) add positional embeddings to the inputs right after the stem of them model and before feeding the inputs to the Transformer encoder (default across all other experiments in this paper); (2) learn and add positional embeddings to the inputs at the beginning of each layer; (3) add a learned positional embeddings to the inputs at the beginning of each layer (shared between layers).

Table 8 summarizes the results from this ablation study on a ViT-B/16 model. As we can see, while there is a large gap between the performances of the model with no positional embedding and models with positional embedding, there is little to no difference between different ways of encoding positional information. We speculate that since our Transformer encoder operates on patch-level inputs, as opposed to pixel-level, the differences in how to encode spatial information is less important. More precisely, in patch-level inputs, the spatial dimensions are much smaller than the original pixel-level inputs, e.g., 14×14 as opposed to 224×224, and learning to represent the spatial relations in this resolution is equally easy for these different positional encoding strategies. Even so, the specific pattern of position embedding similarity learned by the network depends on the training hyperparameters (Figure 10).

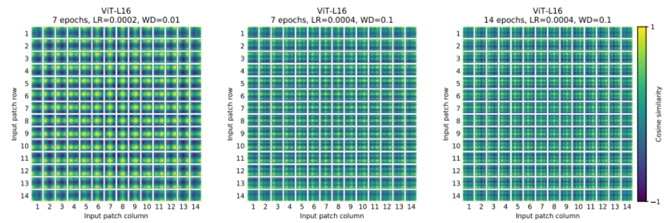


Figure 10: Position embeddings of models trained with different hyperparameters.

Pos. Emb.	Default/Stem	Every Layer	Every Layer-Shared
No Pos. Emb.	0.61382	N/A	N/A
1-D Pos. Emb.	0.64206	0.63964	0.64292
2-D Pos. Emb.	0.64001	0.64046	0.64022
Rel. Pos. Emb.	0.64032	N/A	N/A

Different PE Methods

Table 8: Results of the ablation study on positional embeddings with ViT-B/16 model evaluated on ImageNet 5-shot linear.

CLASS TOKEN EXPERIMENT

D.3 HEAD TYPE AND CLASS TOKEN

In order to stay as close as possible to the original Transformer model, we made use of an additional [class] token, which is taken as image representation. The output of this token is then transformed into a class prediction via a small multi-layer perceptron (MLP) with tanh as non-linearity in the single hidden layer.

This design is inherited from the Transformer model for text, and we use it throughout the main paper. An initial attempt at using only image-patch embeddings, globally average-pooling (GAP) them, followed by a linear classifier—just like ResNet's final feature map—performed very poorly. However, we found that this is neither due to the extra token, nor to the GAP operation. Instead, the difference in performance is fully explained by the requirement for a different learning-rate, see Figure 9.

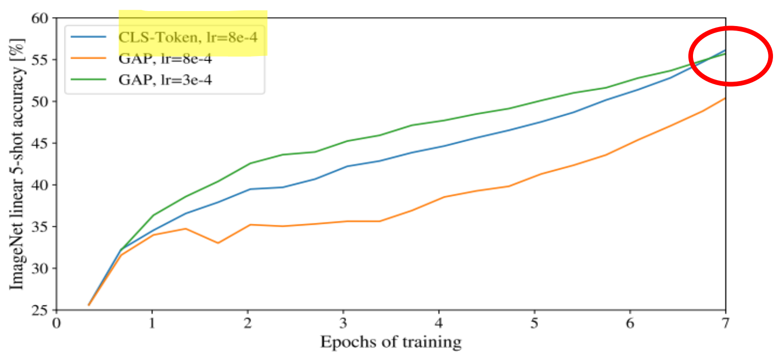
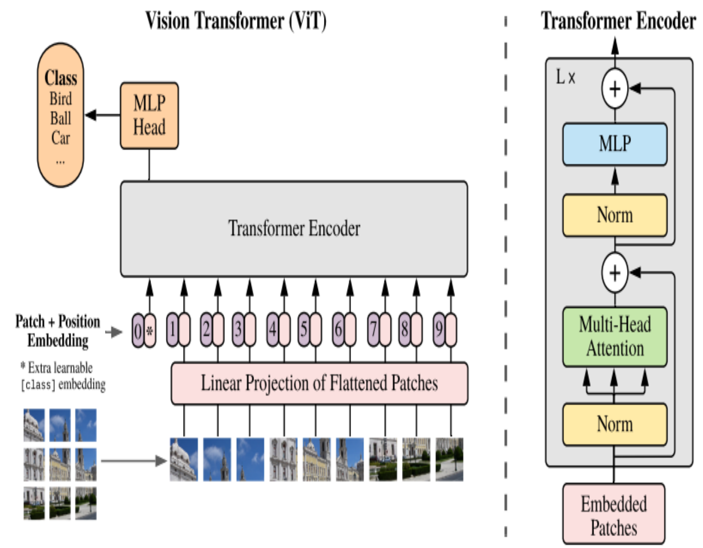
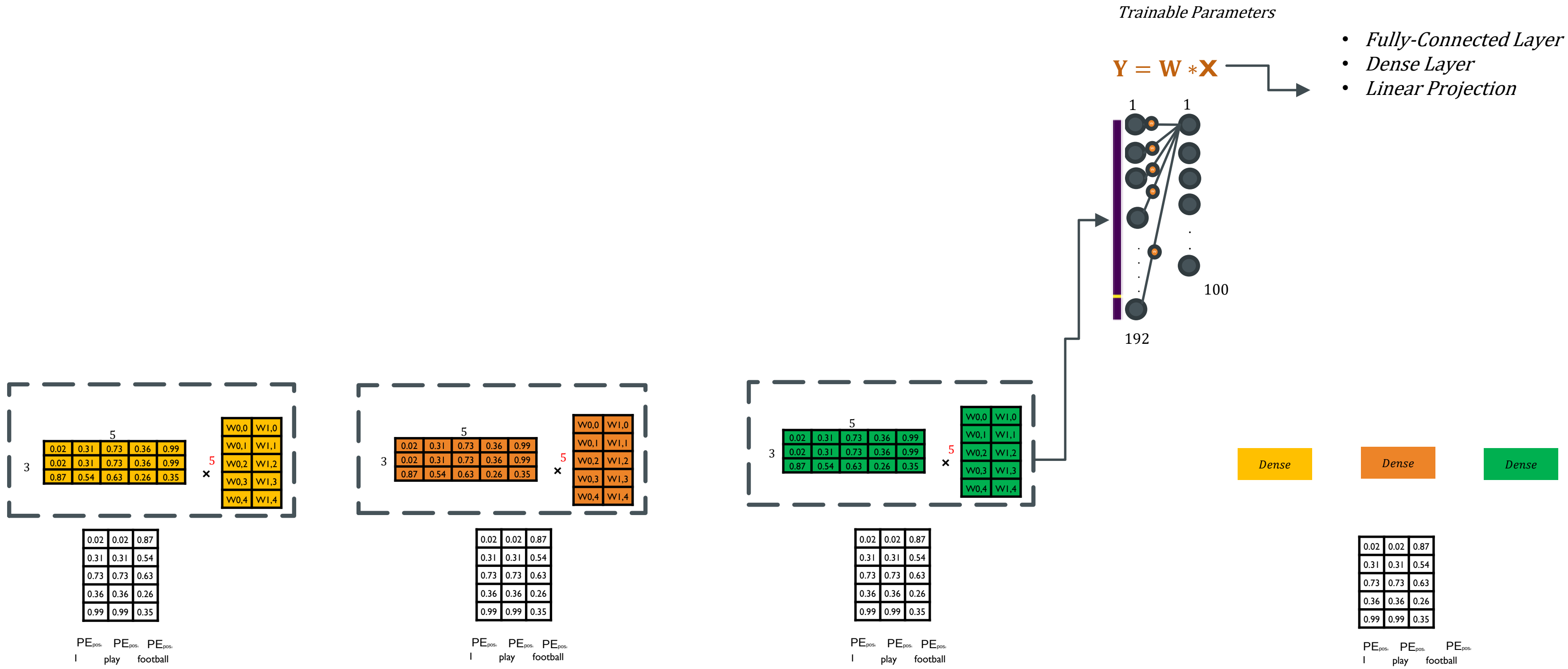


Figure 9: Comparison of class-token and global average pooling classifiers. Both work similarly well, but require different learning-rates.

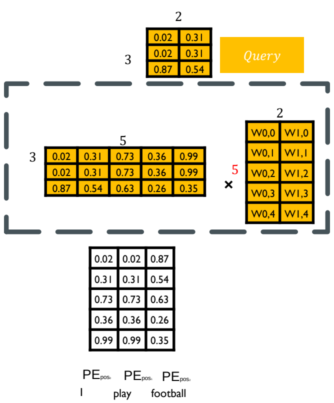


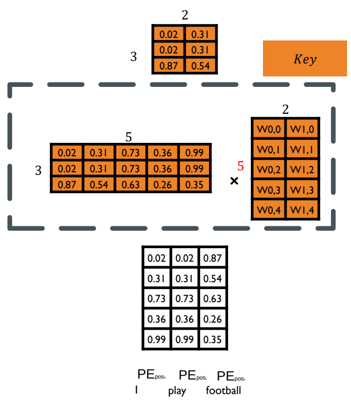
Transformer Workflow
Math Idea

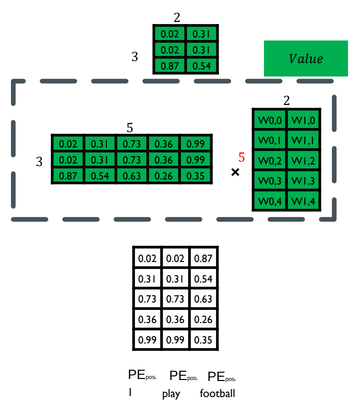

0.02 0.02 0.87
0.31 0.31 0.54
0.73 0.73 0.63
0.36 0.36 0.26
0.99 0.99 0.35
PE pos. I
PE pos. play
PE pos. football
The Same input is repeated three times

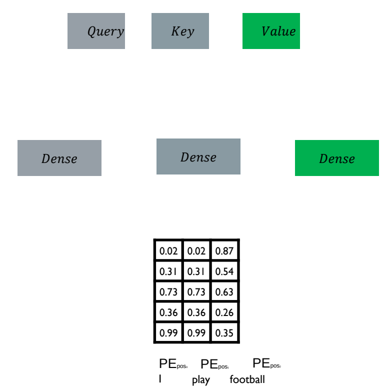


DR. AHMED I. SHAHIN
a.shahin@mu.edu.sa

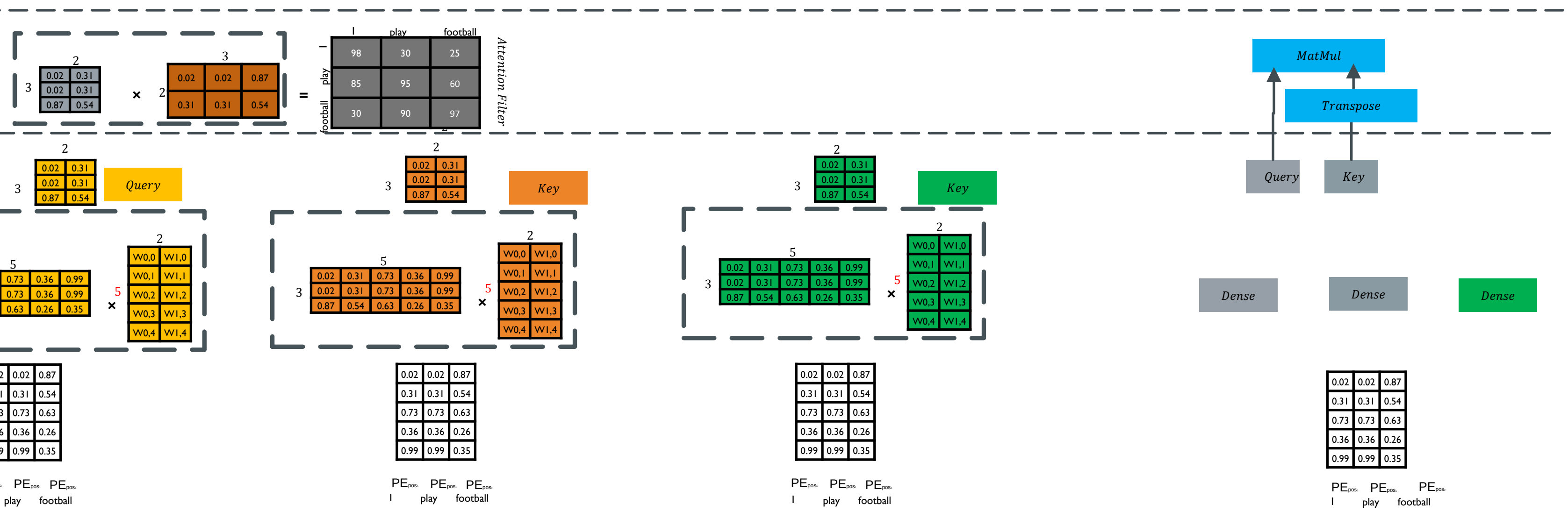

Query
W0,0 W1,0
W0,1 W1,1
W0,2 W1,2
W0,3 W1,3
W0,4 W1,4
PEpos. I
PEpos. play
PEpos. football


Key
W0,0 W1,0
W0,1 W1,1
W0,2 W1,2
W0,3 W1,3
W0,4 W1,4
PEpos. I
PEpos. play
PEpos. football

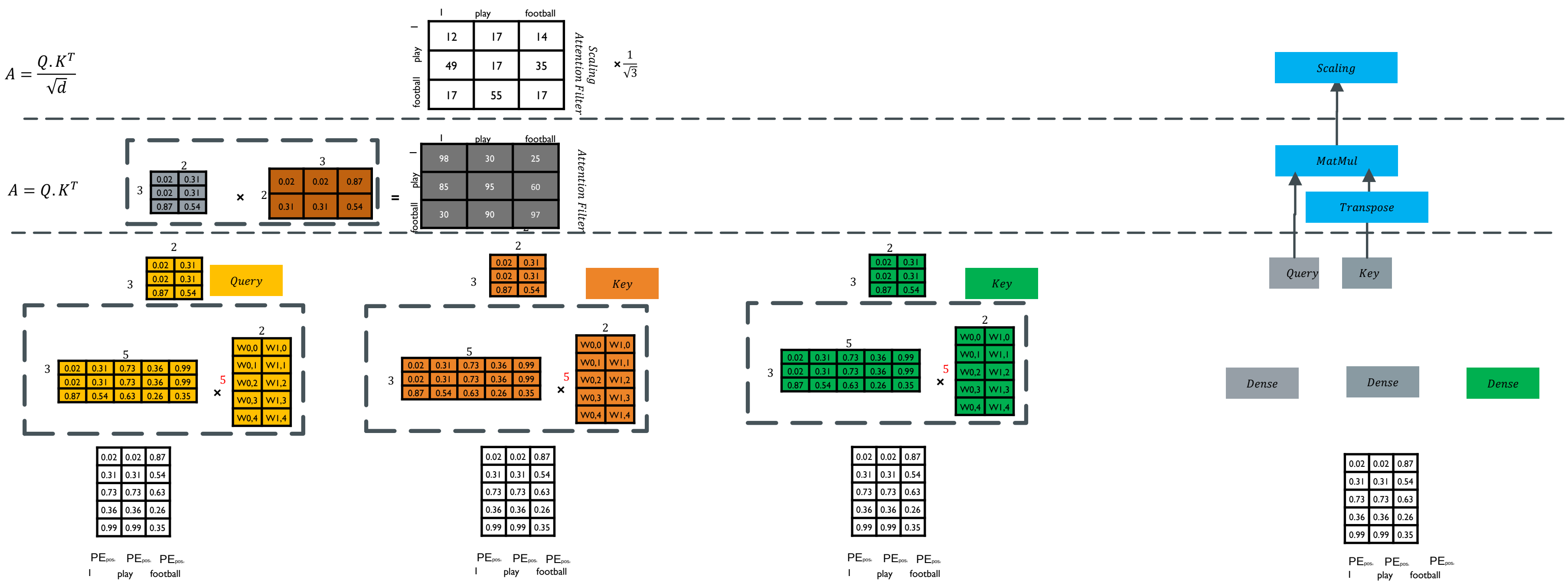

Value
W0,0 W1,0
W0,1 W1,1
W0,2 W1,2
W0,3 W1,3
W0,4 W1,4
PEpos. I
PEpos. play
PEpos. football


Query
Key
Value
Dense
Dense
Dense
PEpos. I
PEpos. play
PEpos. football

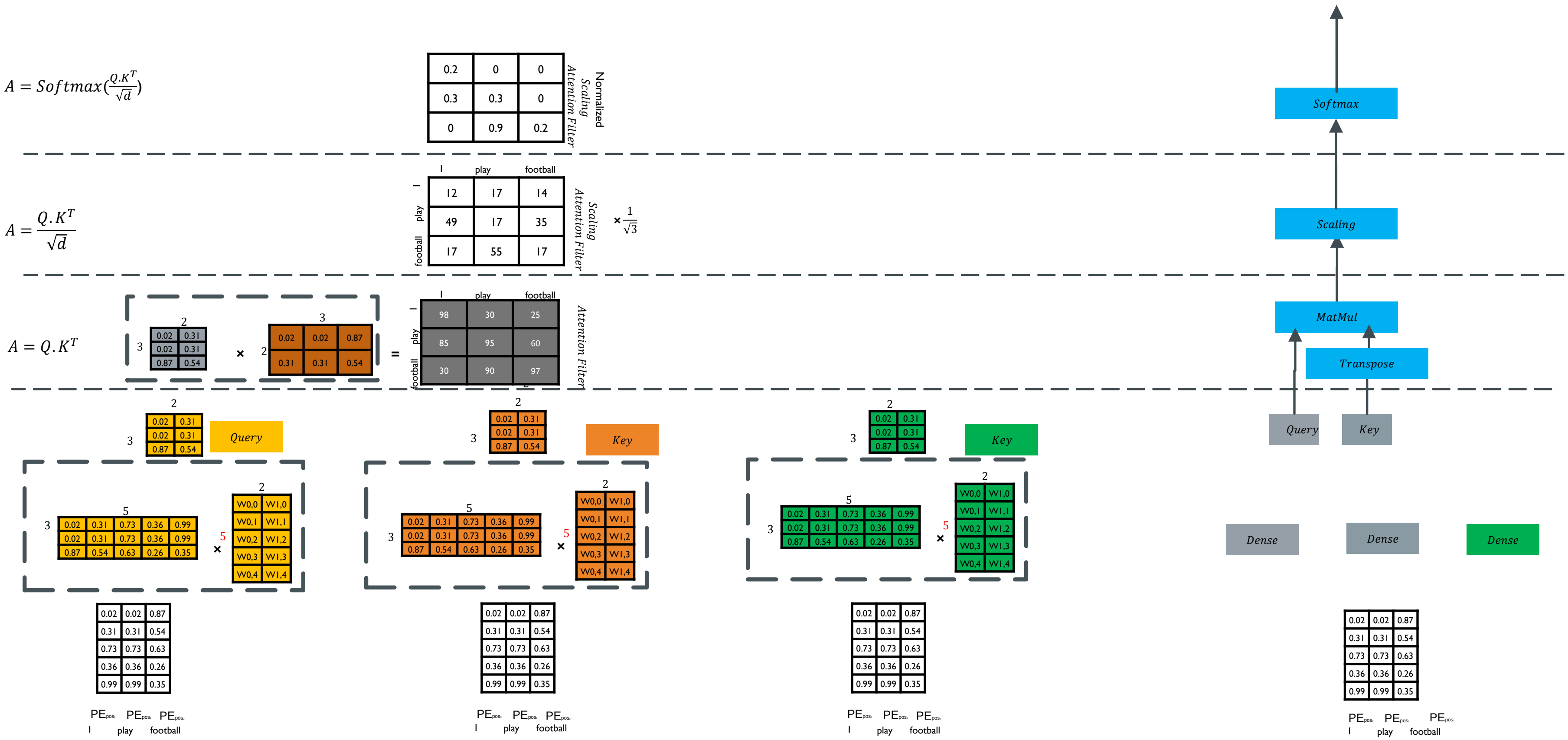
$$A = Q.K^T$$



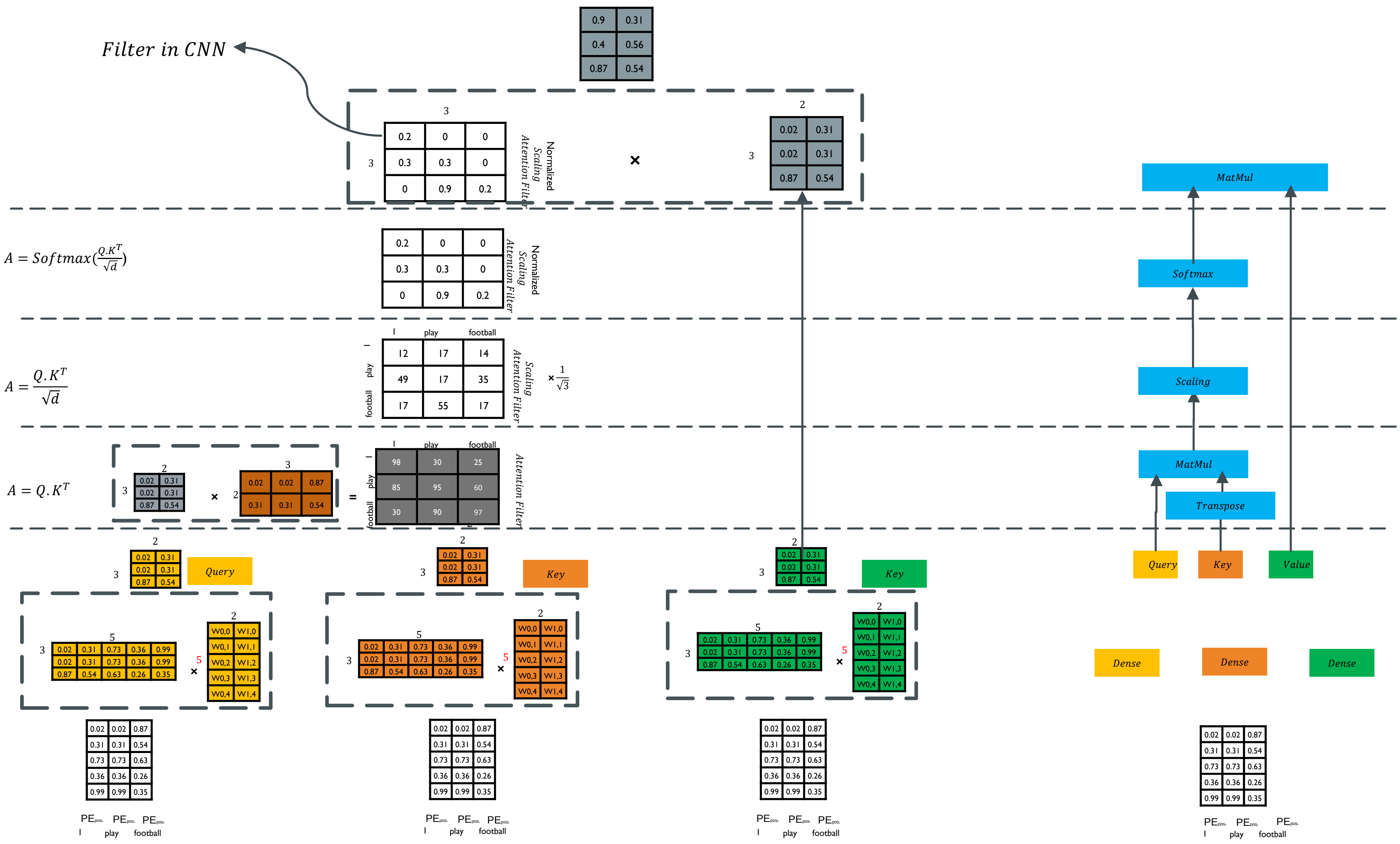
Attention Filter
I
play
football
98
30
25
85
95
60
30
90
97
Query
Key
Key
MatMul
Transpose
Query
Key
Dense
Dense
Dense
PE pos. I
PE pos. play
PE pos. football



$A = \frac{Q.K^T}{\sqrt{d}}$
$A = Q.K^T$
Scaling Attention Filter
Attention Filter
$\times \frac{1}{\sqrt{3}}$
I
play
football
Query
Key
Key
Scaling
MatMul
Transpose
Query
Key
Dense
Dense
Dense
PEpos. I
PEpos. play
PEpos. football



A = Softmax(Q.K^T/√d)
Normalized Scaling Attention Filter
A = Q.K^T/√d
Scaling Attention Filter
× 1/√3
A = Q.K^T
Attention Filter
I play football
Query
Key
Key
Softmax
Scaling
MatMul
Transpose
Query
Key
Dense
Dense
Dense
PEpos. I
PEpos. play
PEpos. football

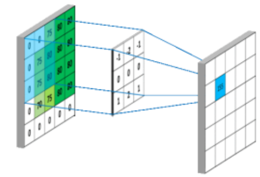


Filter in CNN
Normalized Scaling Attention Filter
$A = Softmax(\frac{Q.K^T}{\sqrt{d}})$
$A = \frac{Q.K^T}{\sqrt{d}}$
Scaling Attention Filter
$\times \frac{1}{\sqrt{3}}$
$A = Q.K^T$
Attention Filter
I
play
football
Query
Key
Value
MatMul
Softmax
Scaling
Transpose
Dense
PEpos.

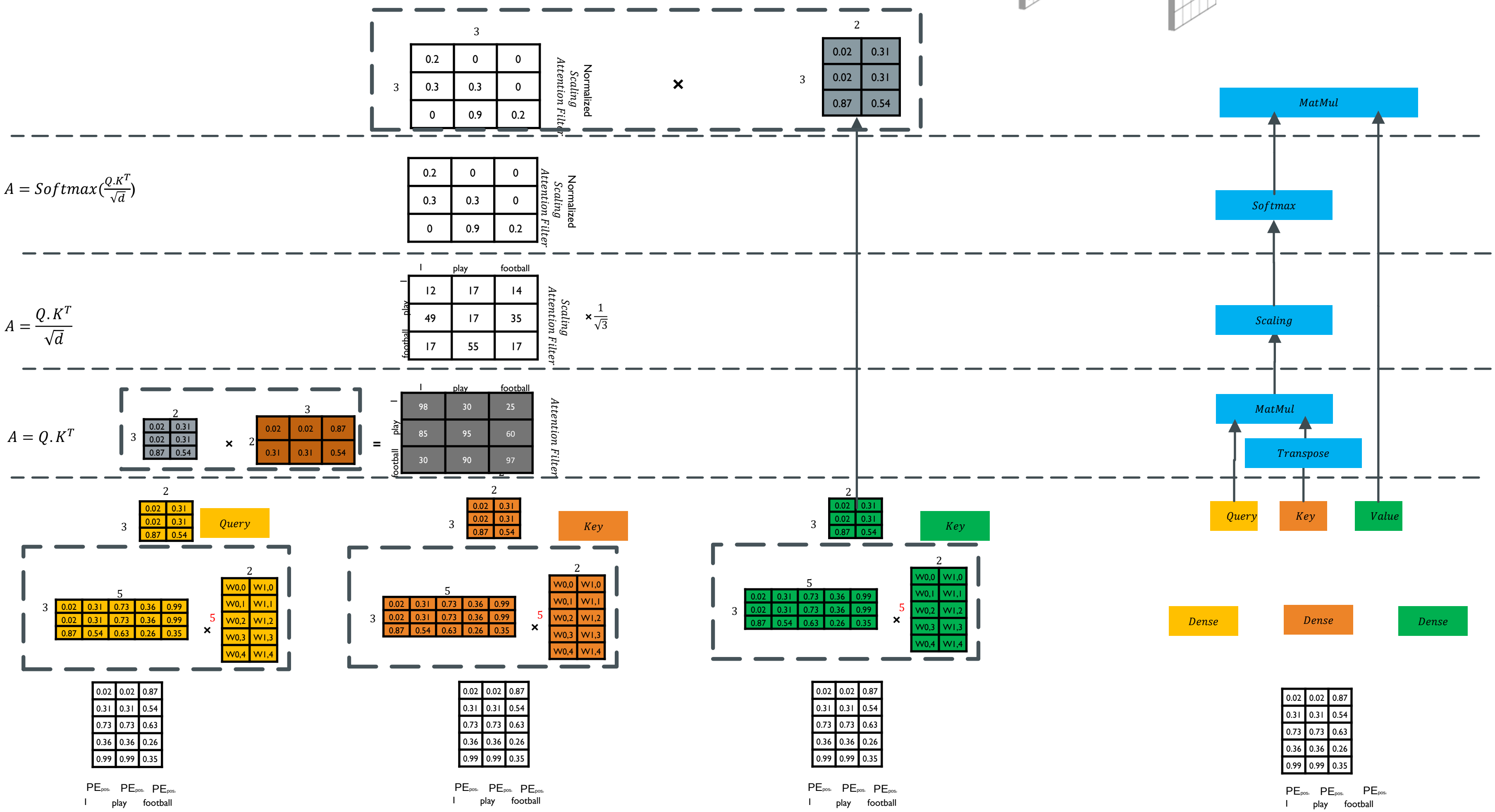
Transformer Vs. CNN

- Not Inductive Bias
  - Shift- variant
  - Scale- variant

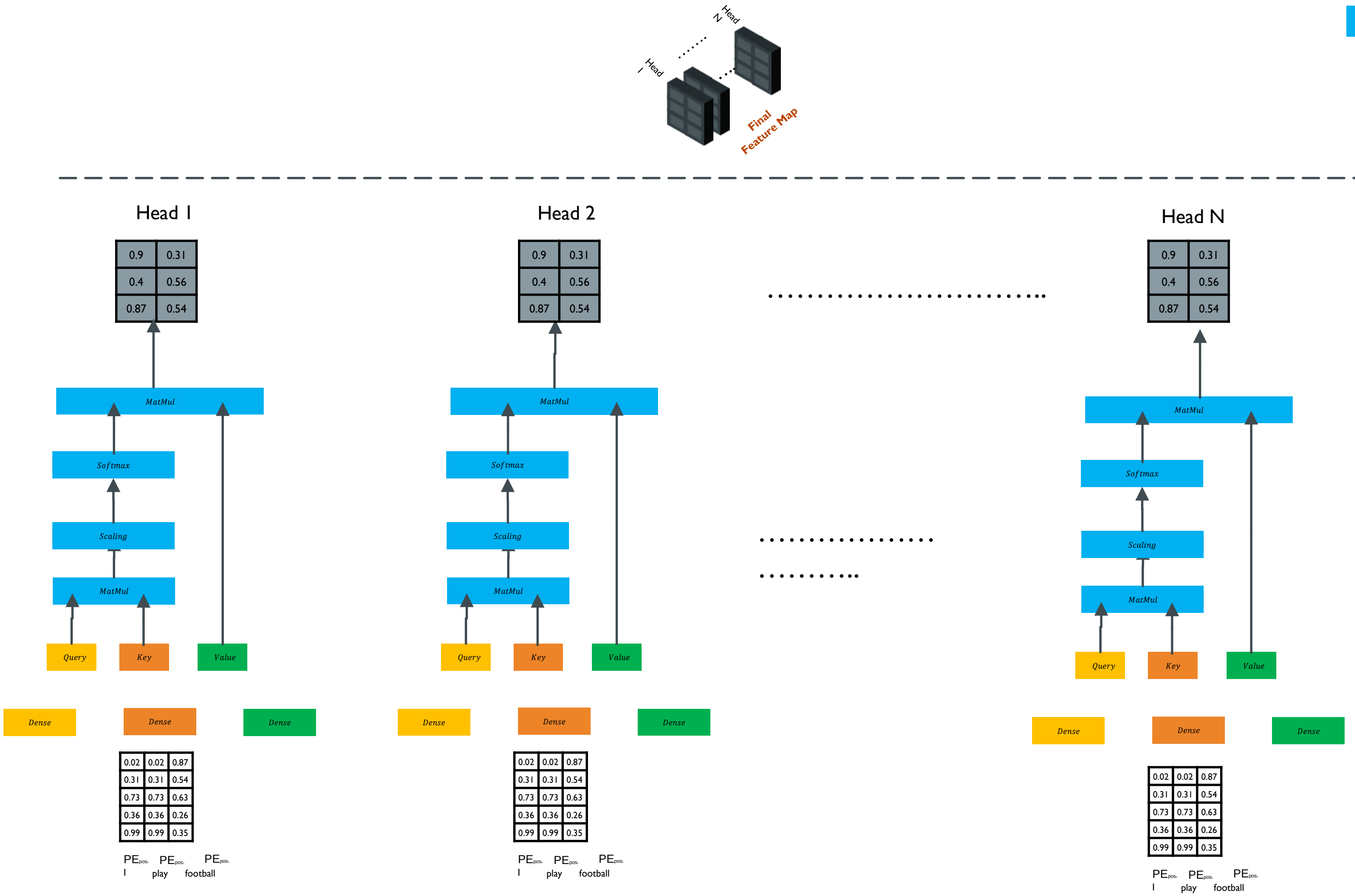
0.9	0.31
0.4	0.56
0.87	0.54

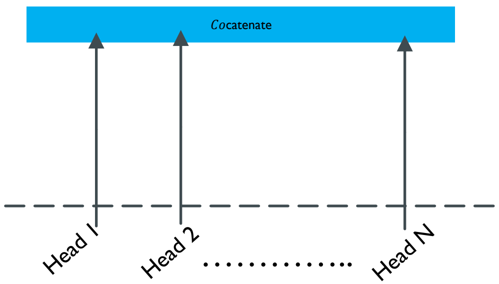


- Inductive Bias
  - Shift- Invariant
  - Scale- Invariant

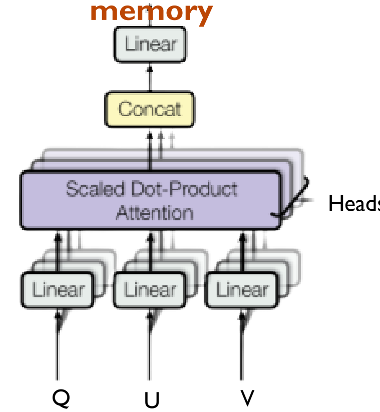


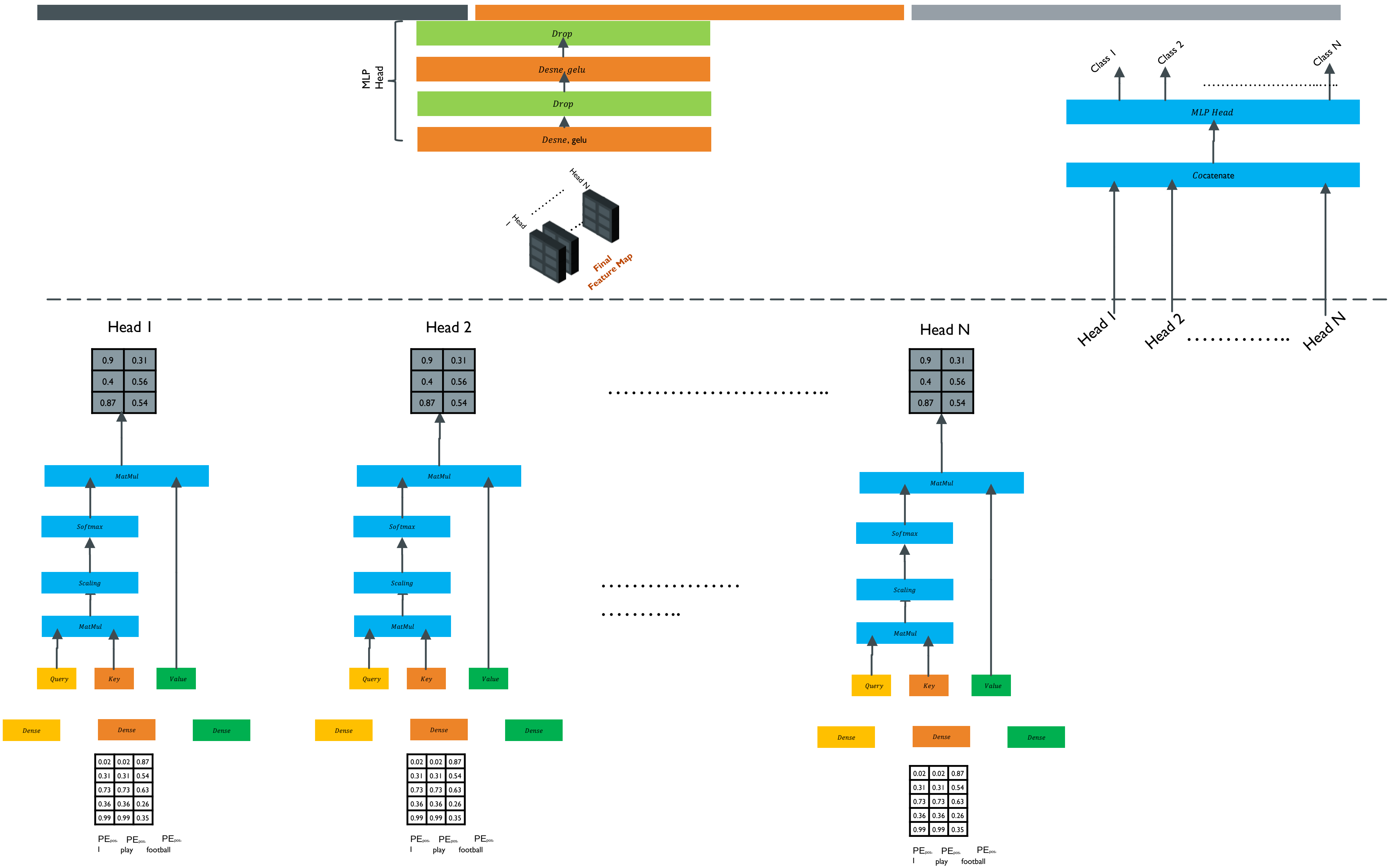
a.shahin@mu.edu.sa





Something is happening in parallel way at the same time, which causes out of memory



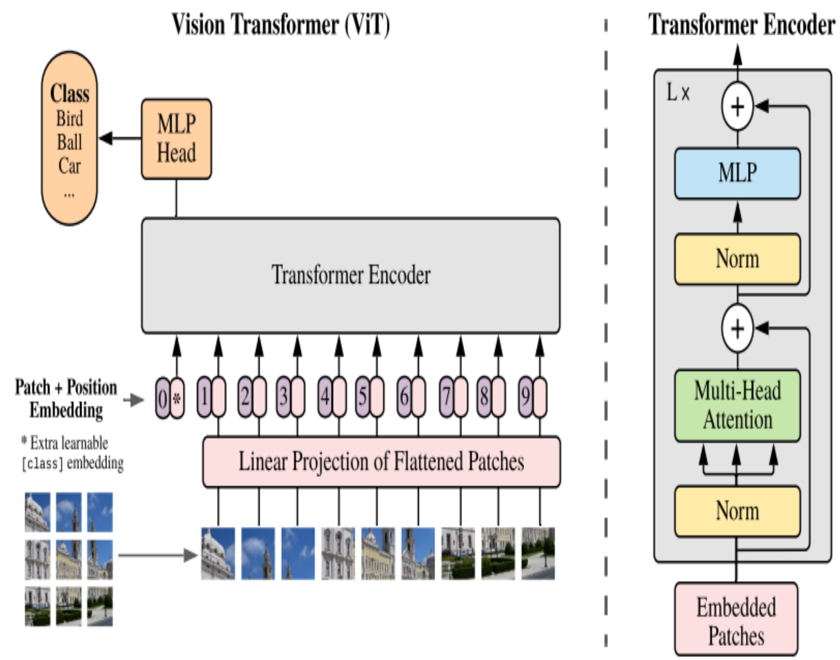


DR. AHMED I. SHAHIN
a.shahin@mu.edu.sa

TRANSFORMERS IN CV

Disadvantage in Transformer

- Not Inductive Bias to CV problems
- Shift-Scale Variant
- Position Encoding effect which represents the spatial info is weak.
- Model Complexity is High.(16 Heads, 12:32 Layers)



Linear Regression

Transofrmer

CNN

CV

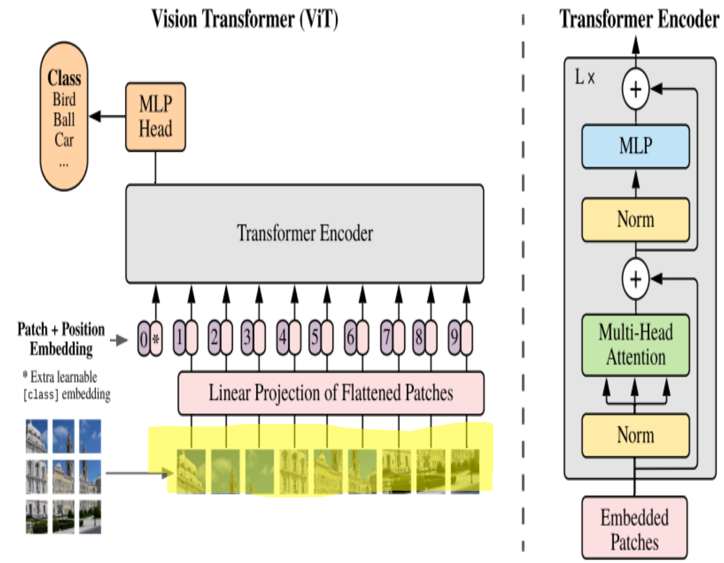
Transformer Workflow Implementation Idea

TensorFlow

TRANSFORMERS IN CV

ViT in Tensorflow

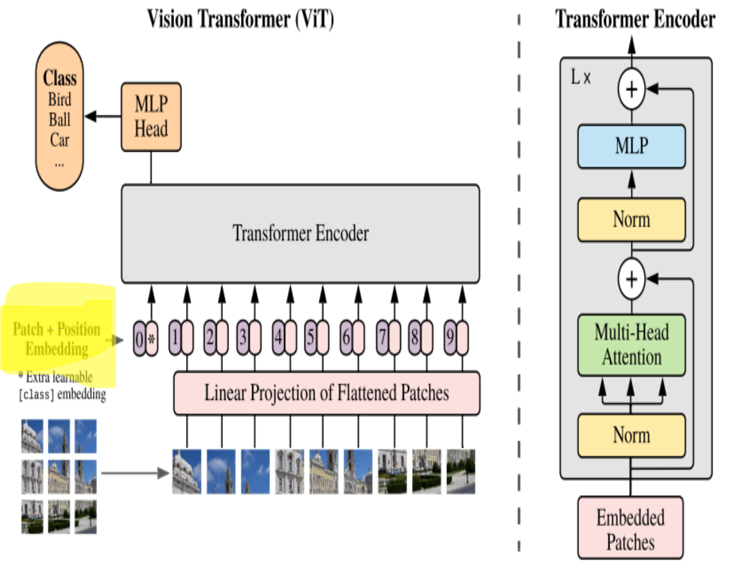
```
class Patches(layers.Layer):
    def __init__(self, patch_size):
        super(Patches, self).__init__()
        self.patch_size = patch_size

    def call(self, images):
        batch_size = tf.shape(images)[0]
        patches = tf.image.extract_patches(
            images=images,
            sizes=[1, self.patch_size, self.patch_size, 1],
            strides=[1, self.patch_size, self.patch_size, 1],
            rates=[1, 1, 1, 1],
            padding="VALID",

        )
        patch_dims = patches.shape[-1]
        patches = tf.reshape(patches, [batch_size, 1, patch_dims])
        return patches
```



TRANSFORMERS IN CV

ViT in Tensorflow

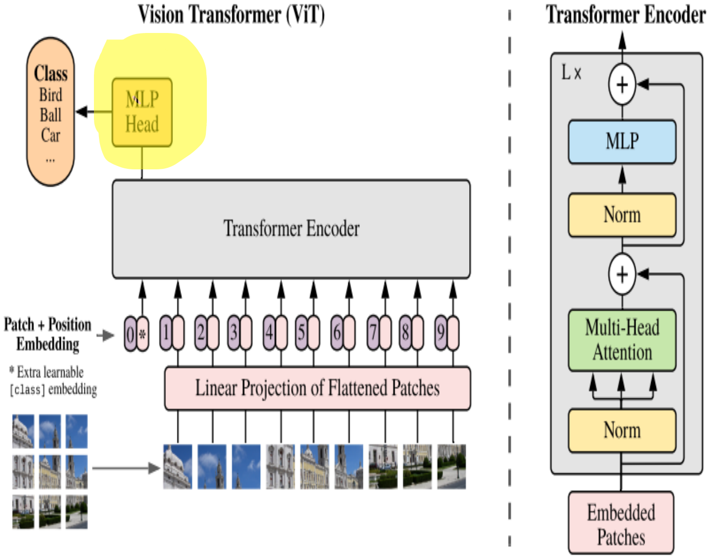
```
class PatchEncoder(layers.Layer):
    def __init__(self, num_patches, projection_dim):
        super(PatchEncoder, self).__init__()
        self.num_patches = num_patches
        self.projection = layers.Dense(units=projection_dim)
        self.position_embedding = layers.Embedding(
            input_dim=num_patches, output_dim=projection_dim
        )

    def call(self, patch):
        positions = tf.range(start=0, limit=self.num_patches, delta
=1)
        encoded = self.projection(patch) + self.position_embedding(
positions)
        return encoded
```



TRANSFORMERS IN CV

ViT in Tensorflow

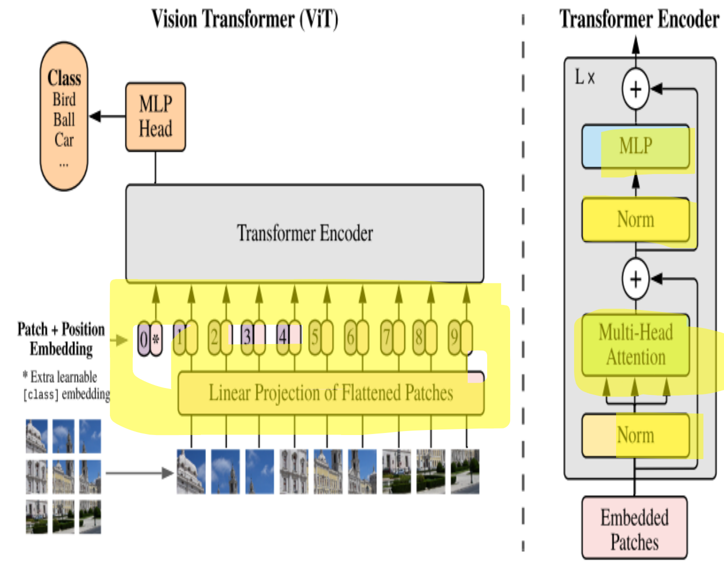
```
def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = layers.Dense(units, activation=tf.nn.gelu)(x)
        x = layers.Dropout(dropout_rate)(x)
    return x
```



TRANSFORMERS IN CV

ViT in Tensorflow

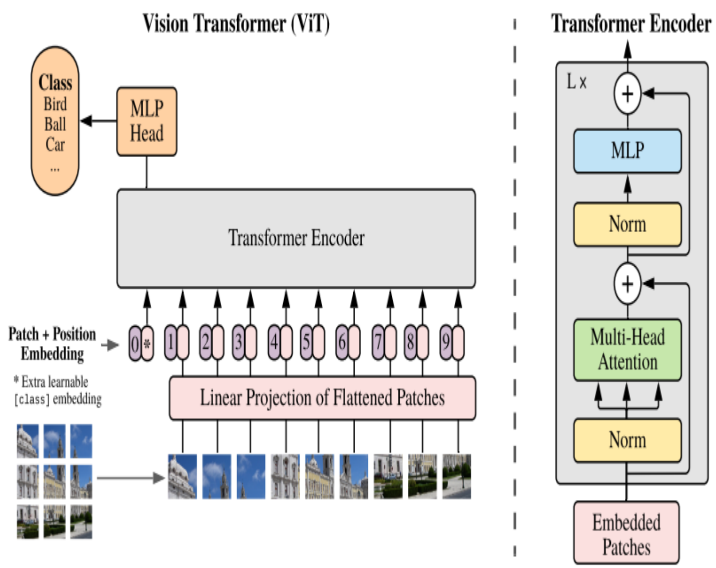
```
def create_vit_classifier():
    inputs = layers.Input(shape=input_shape)
    patches = Patches(patch_size)(inputs)
    # Encode patches.
    encoded_patches = PatchEncoder(num_patches, projection_dim)(patche

    # Create multiple layers of the Transformer block.
    for _ in range(transformer_layers):
        # Layer normalization 1.
        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        # Create a multi-head attention layer.
        attention_output = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=projection_dim, dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = layers.Add()([attention_output, encoded_patches])
        # Layer normalization 2.
        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
        # MLP.
        x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
        # Skip connection 2.
        encoded_patches = layers.Add()([x3, x2])
```



TRANSFORMERS IN CV

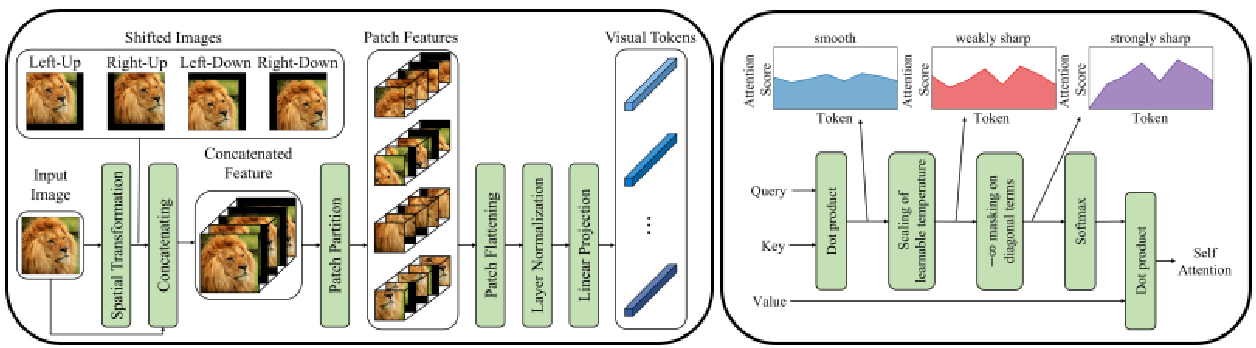
ViT in Tensorflow

```
 # Create a [batch_size, projection_dim] tensor.
    representation = layers.LayerNormalization(epsilon=1e-
6)(encoded_patches)
    representation = layers.Flatten()(representation)
    representation = layers.Dropout(0.5)(representation)
    # Add MLP.
    features = mlp(representation, hidden_units=mlp_head_units, dropou
t_rate=0.5)
    # Classify outputs.
    logits = layers.Dense(num_classes)(features)
    # Create the Keras model.
    model = keras.Model(inputs=inputs, outputs=logits)
    return model
```

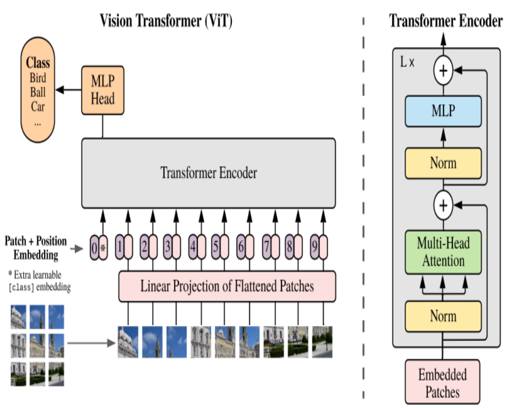


Vision Transformer for Small-Size Datasets

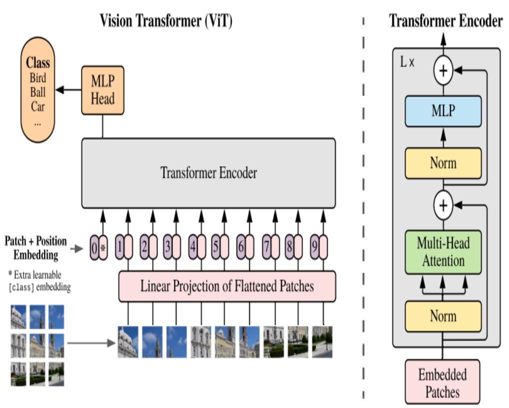
ViT



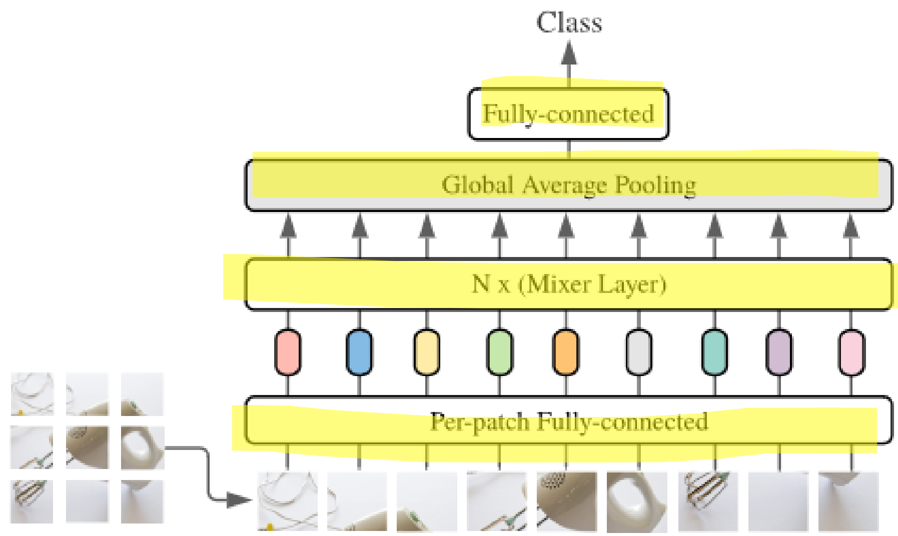
(a) Shifted Patch Tokenization (b) Locality Self-Attention



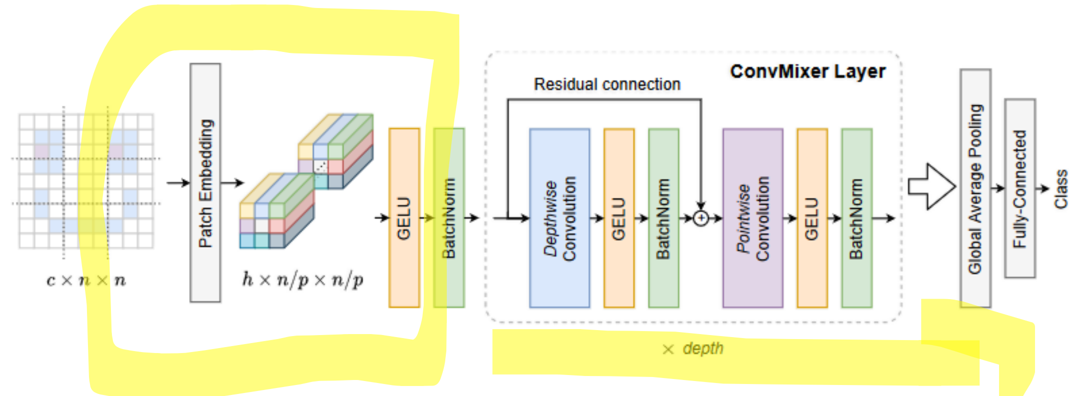
ViT



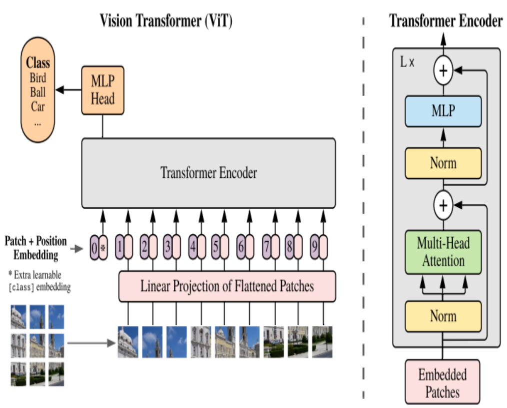
MLP Mixer



Convmixer

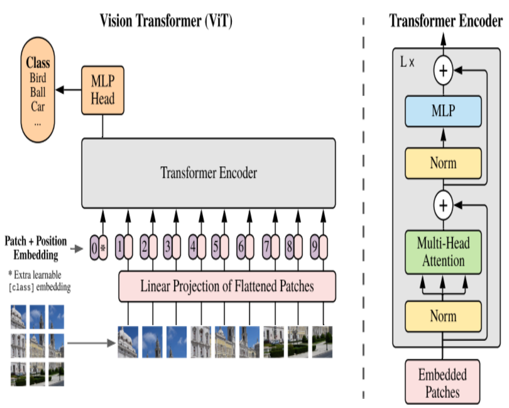


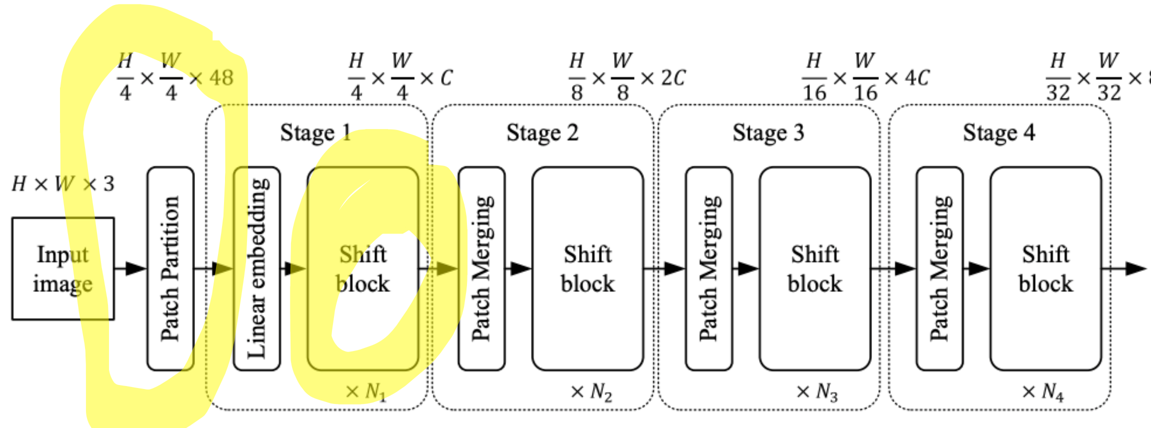
ViT



A Vision Transformer without Attention

ViT

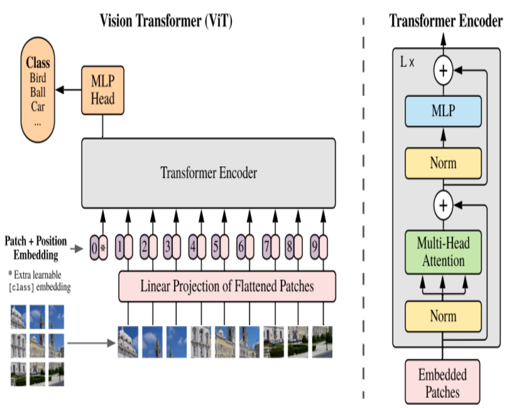




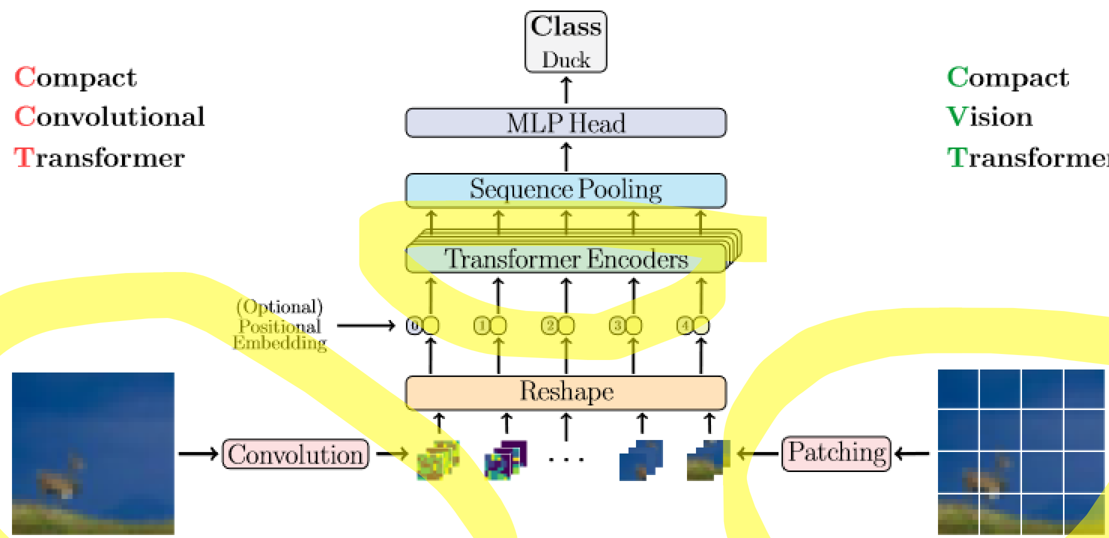
a.shahin@mu.edu.sa

Compact Convolutional Transformers

ViT



Compact
Convolutional
Transformer



Compact
Vision
Transformer

STUDY PLAN

- **ViT**

https://keras.io/examples/vision/image_classification_with_vision_transformer/

https://medium.com/geekculture/vision-transformer-tensorflow-82ef13a9279

https://arxiv.org/pdf/2010.11929.pdf

- **ViT fine-tuning**

https://www.kaggle.com/code/raufmomin/vision-transformer-vit-fine-tuning

- **ViT to Object detection**

https://keras.io/examples/vision/object_detection_using_vision_transformer/

- **Investigating Vision Transformer representations**

https://keras.io/examples/vision/probing_vits/

- **Vision Transformer for Small-Size Datasets**

https://keras.io/examples/vision/vit_small_ds/

https://arxiv.org/pdf/2112.13492v1.pdf

- **MLP Mixer**

https://keras.io/examples/vision/mlp_image_classification/

https://arxiv.org/pdf/2105.01601.pdf

https://arxiv.org/pdf/2201.09792.pdf

- **FNet**

https://keras.io/examples/vision/mlp_image_classification/

https://arxiv.org/pdf/2105.03824.pdf

- **Pay Attention to MLPs**

https://keras.io/examples/vision/mlp_image_classification/

https://arxiv.org/pdf/2105.08050.pdf

- **ConvMixer**

https://keras.io/examples/vision/convmixer/

https://arxiv.org/pdf/2201.09792.pdf

- **EANet**

https://keras.io/examples/vision/eanet/

https://arxiv.org/pdf/2105.02358.pdf

- **A Vision Transformer without Attention**

https://keras.io/examples/vision/shiftvit/

https://arxiv.org/pdf/2201.10801.pdf

- **Compact Convolutional Transformers**

https://keras.io/examples/vision/cct/

https://arxiv.org/pdf/2104.05704.pdf

- **Talking-Heads Attention**

https://keras.io/examples/vision/cait/

https://arxiv.org/pdf/2003.02436.pdf

REFERENCES

- https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1
- https://www.youtube.com/watch?v=dichIcUZfOw
- https://www.youtube.com/watch?v=mMa2PmYJlCo
- https://towardsdatascience.com/concepts-about-positional-encoding-you-might-not-know-about-1f247f4e4e23
- https://medium.com/analytics-vidhya/understanding-the-vision-transformer-and-counting-its-parameters-988a4ea2b8f3

TUTORIAL FOR POSITION ENCODING IN NLP

- https://machinelearningmastery.com/the-transformer-positional-encoding-layer-in-keras-part-2/

COVID Example

https://www.kaggle.com/code/basu369victor/covid19-detection-with-vit-and-heatmap/notebook