(Vision) Transformers

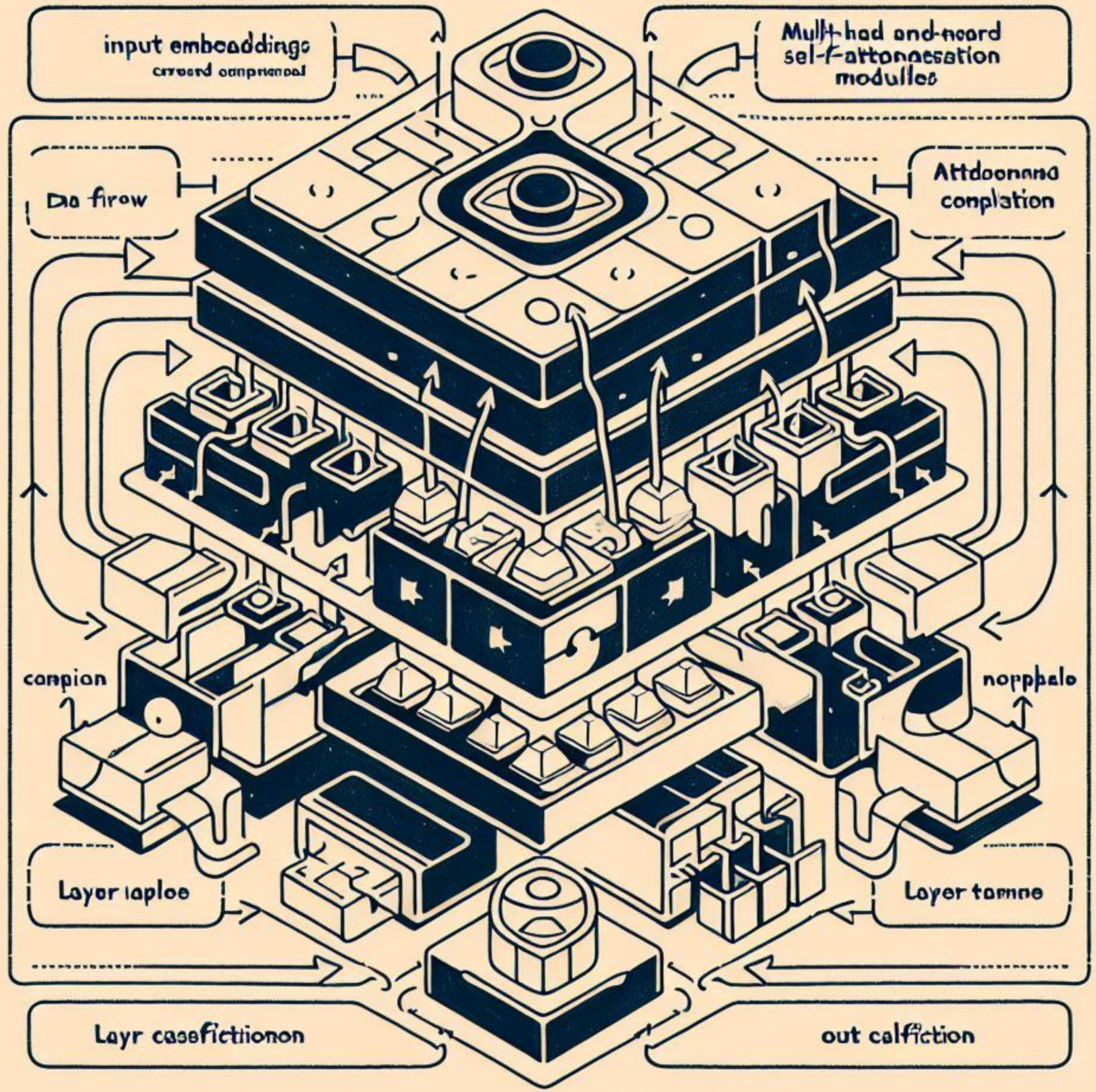
Computer Vision and Artificial Intelligence

Dr Varun Ojha

varun.ojha@ncl.ac.uk

School of Computing

Newcastle University



Apple is sweet

Apple is fast

Learning objectives (Vision Transformer)

By the end of this week, you will be able to:

- Learn the concepts of Transformer Models
- Understand the Self-Attention Mechanisms (the basic building block of Transformers)
- Understand an image classification using Vision Transformers

Content of this week (ViT)

- **Part 1: Self-Attention Block**
  - Basic concept of Transformers
  - Word Embedding
  - Self-Attention Mechanism
- **Part 2: Vision Transformer**
  - Basic concept of Vision Transformers
  - Architecture of a Vision Transformer
  - Performance of Vision Transformer

Word Embedding

Given a word W (e.g. "intelligence") we want W to be a real vector of dimension d. Dimension d is also called word embedding dimension.

$$\boldsymbol{W}: \text{words} \to \mathbb{R}^d$$

$$\text{"intelligence"} \to (w_1, w_2, \ldots, w_d) \to (0.1, -0.8, \ldots, 0.9)$$

Word Embedding

W_1 = “I love artificial intelligence”

W_2 = “I like computational intelligence”

We create a vocabulary V collecting all unique words.

V = {“I”, “love”, “like”, “artificial”, “computational”, “intelligence”}

For this example, vocabulary size $|V| = 6$

Word Embedding: Word → Integer

V = {“I”, “love”, “artificial”, “computational”, “intelligence”, “like”}

I → 0

love → 1

like → 2

artificial → 3

computational → 4

intelligence → 5

Word Embedding: Integer → Word

V = {“I”, “love”, “artificial”, “computational”, “intelligence”, “like”}

0 → I

1 → love

2 → like

3 → artificial

4 → computational

5 → intelligence

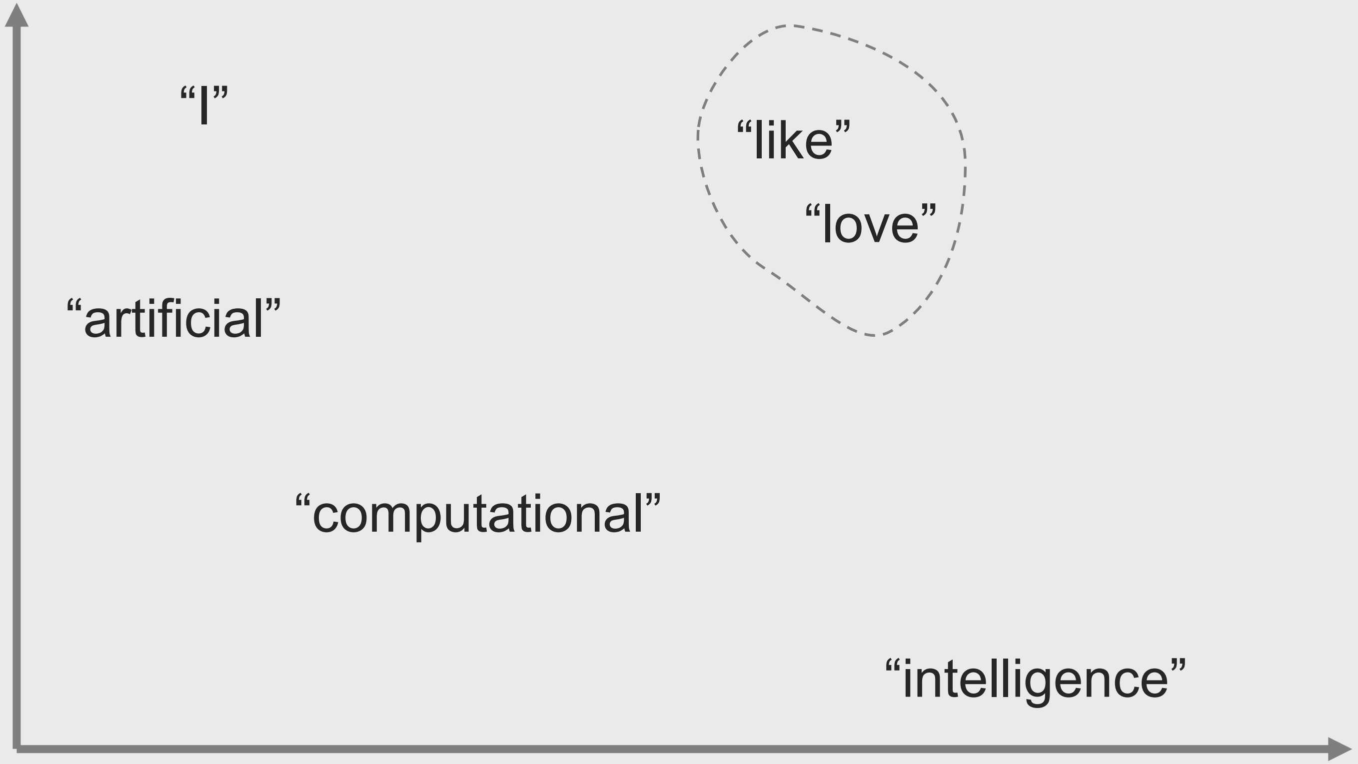
One-Hot Encoding

$V = \{$"I", "love", "like", "artificial", "computational", "intelligence"$\}$

"I" = $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ "love" = $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ "like" = $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ "artificial" = $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

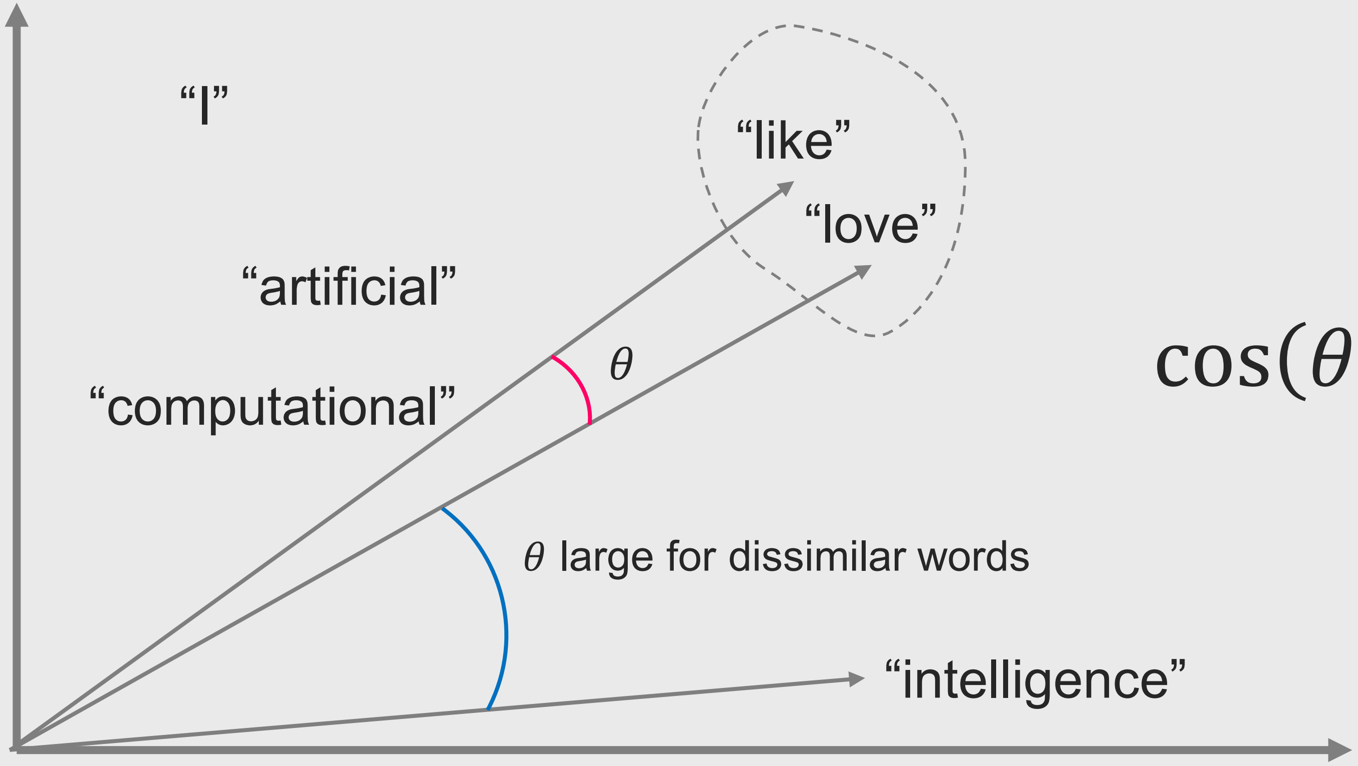
"computational" = $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ "intelligence" = $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Similarity between words?



Objective is to place similar words close to each other

Similarity between words?

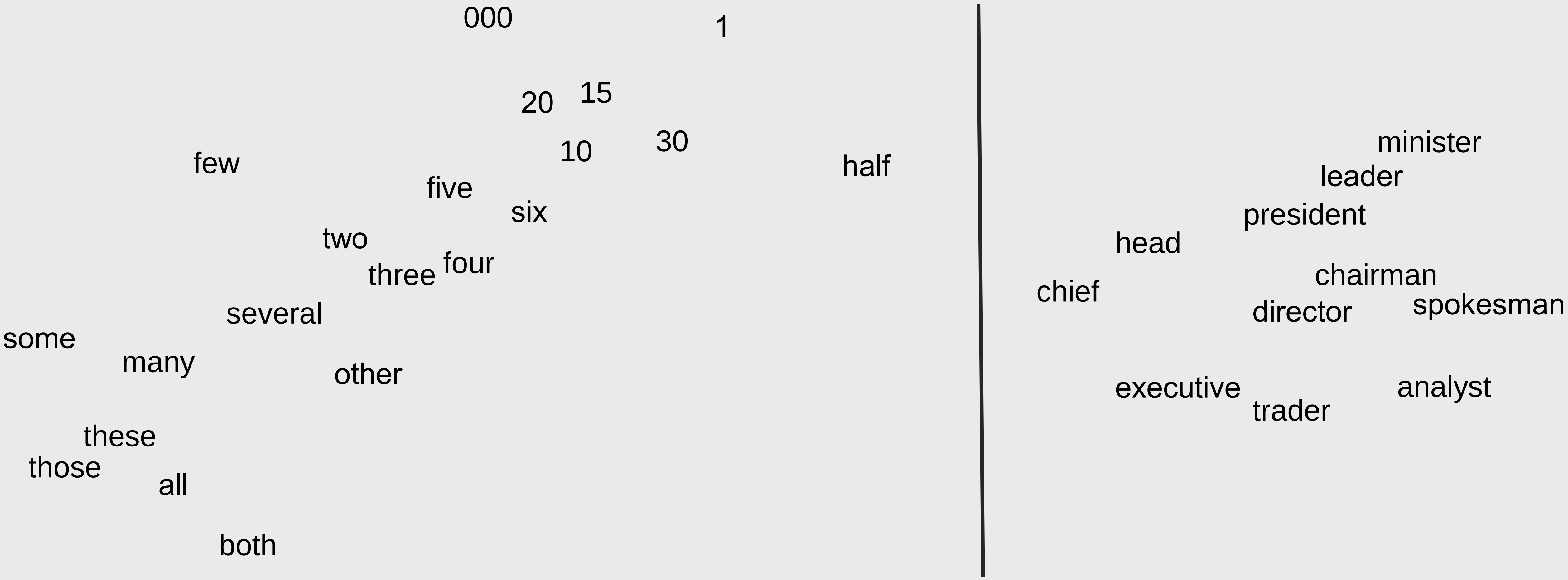


$$\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Objective is to place similar words close to each other

t-SNE visualisation of words

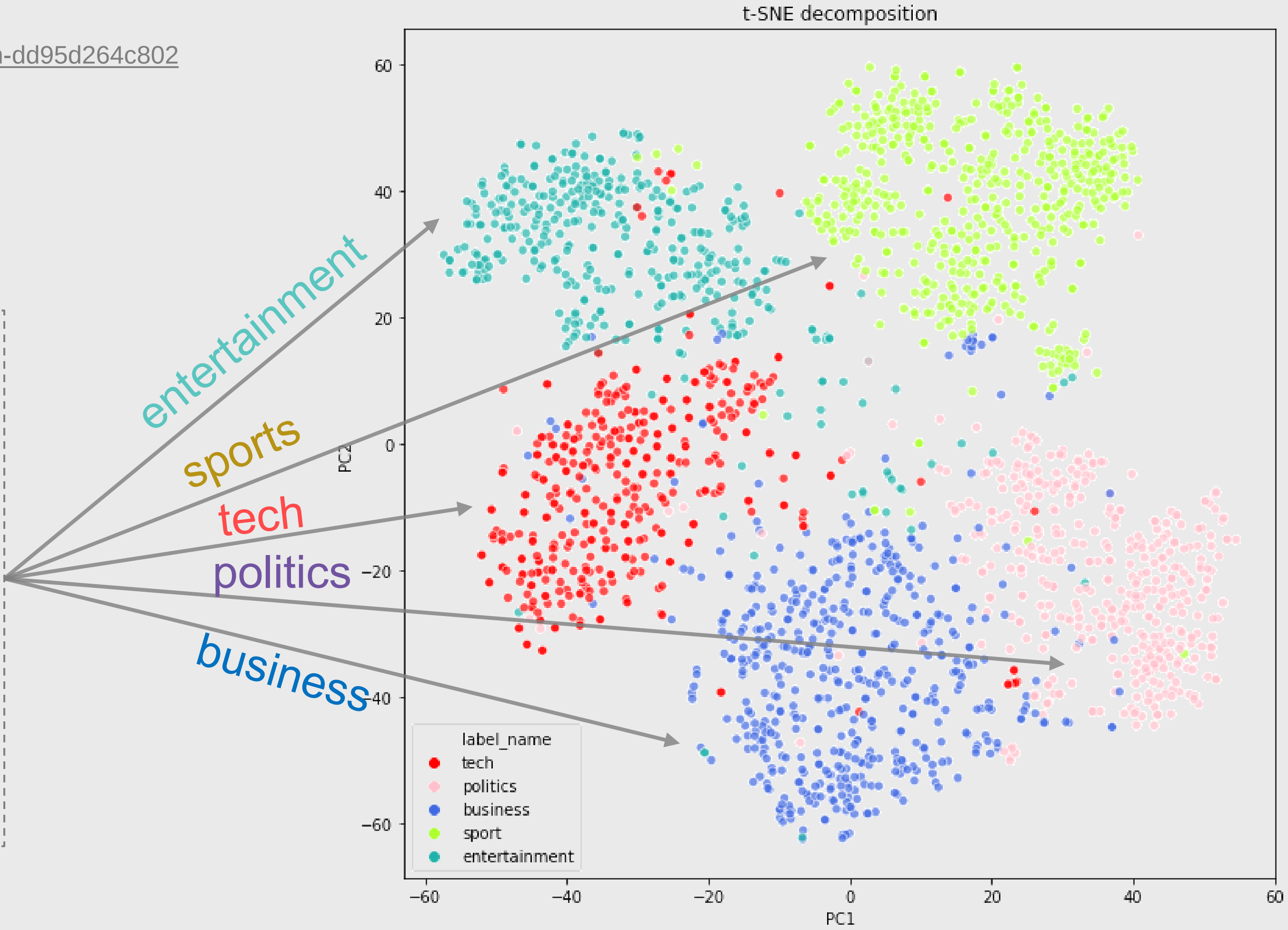
Turian *et al.* (2010)



t-SNE visualisation

Example Source:
https://towardsdatascience.com/text-classification-in-python-dd95d264c802

All similar topics are closer to each other



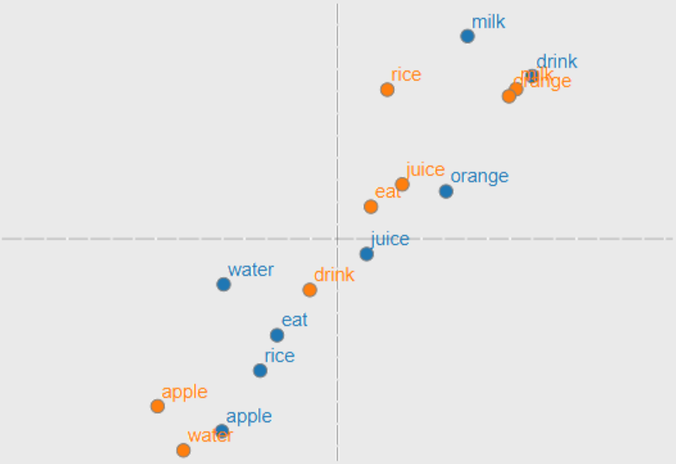
Word Embedding: Objective

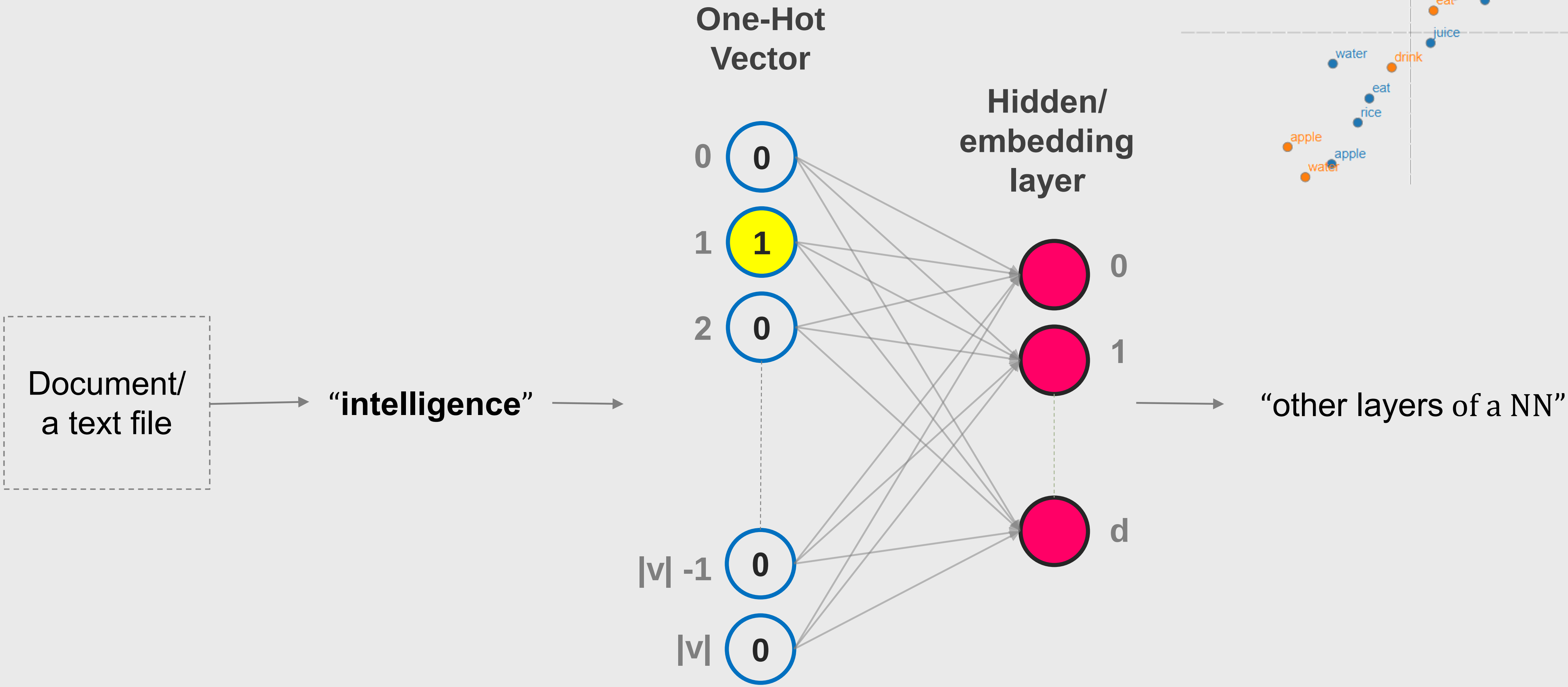
Given a word W (e.g. “intelligence”) we want to W a real vector of dimension d

$$\boldsymbol{W}: \text{words} \rightarrow \mathbb{R}^d$$

“intelligence” $\rightarrow (w_1, w_2, \dots, w_d) \rightarrow (0.1, -0.8, \dots, 0.9)$

Word Embedding

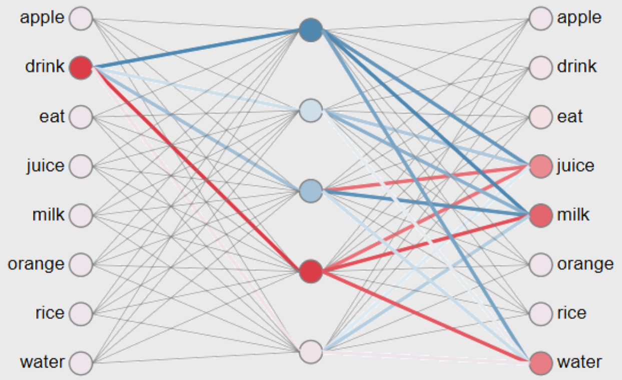


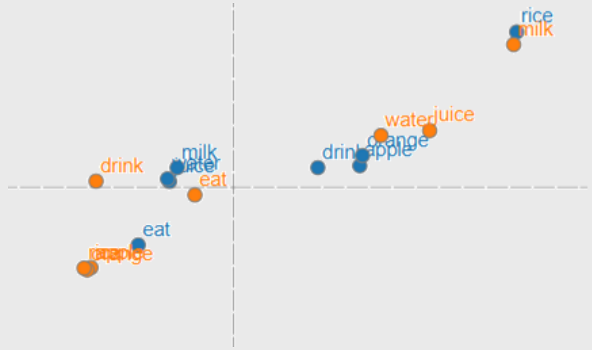


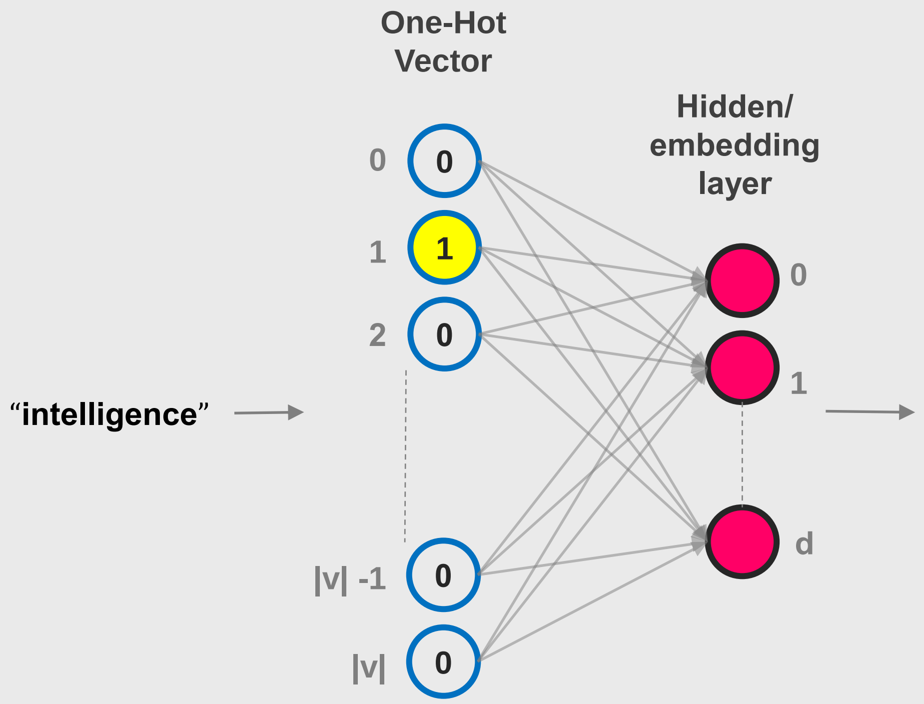
Check online here: https://ronxin.github.io/wevi/

Word Embedding

Check online here: https://ronxin.github.io/wevi/







$$\text{“intelligence”} = [0 \quad 1 \quad 0 \quad \ldots \quad 0 \quad 0] \times \begin{bmatrix} 0.5 & 4.6 & \ldots & 0.9 \\ 0.1 & -0.8 & \ldots & 0.7 \\ \vdots & \vdots & & \vdots \\ 0.6 & 0.8 & \ldots & 0.3 \\ 0.3 & -0.6 & \ldots & -0.8 \\ -0.5 & 0.5 & \ldots & 0.1 \end{bmatrix}$$

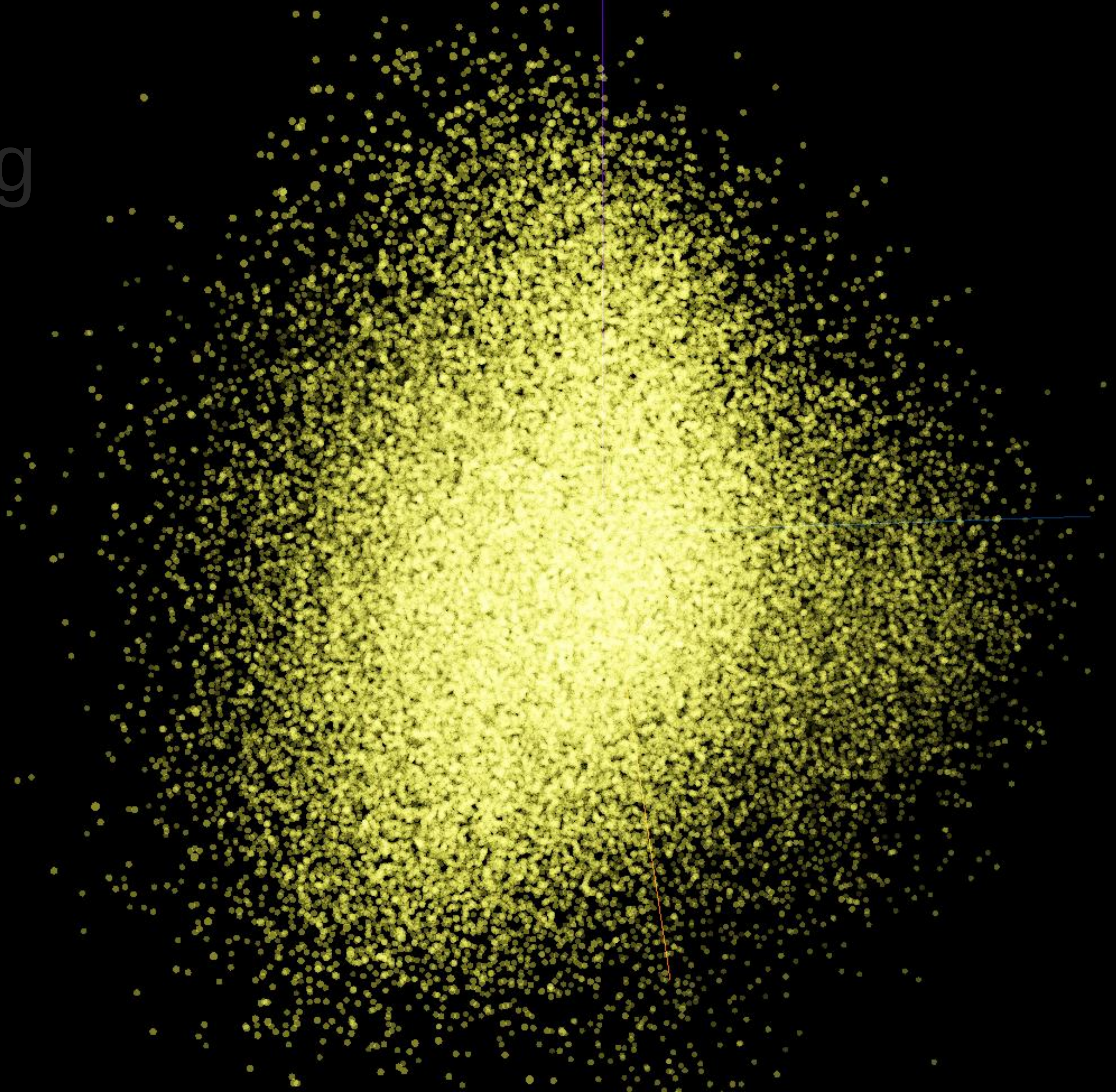
$$(1 \times |V|) \cdot (|V| \times d) \Rightarrow (1 \times d)$$

Word Embedding

$$\text{“intelligence”} = [0 \quad 1 \quad 0 \quad \ldots \quad 0 \quad 0] \quad \times \begin{bmatrix} 0.5 & 4.6 & \ldots & 0.7 \\ 0.1 & -0.8 & \ldots & 0.9 \\ 0.6 & 0.8 & \ldots & 0.3 \\ 0.3 & -0.6 & \ldots & -0.8 \\ -0.5 & 0.5 & \ldots & 0.1 \end{bmatrix}$$

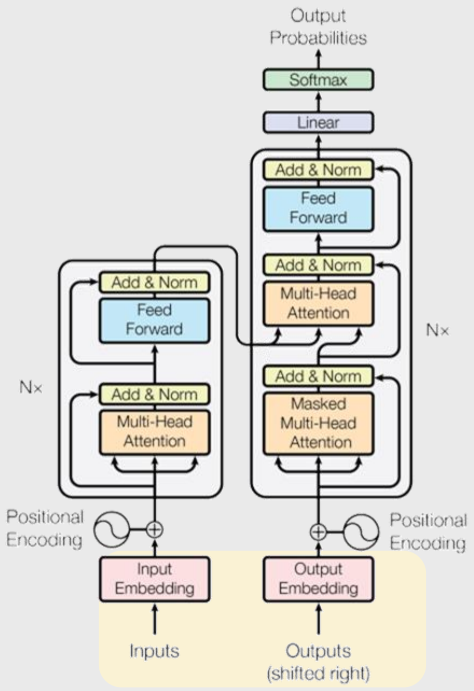
$$= [0.1, -0.8, \ldots, 0.9]$$

Word Embedding



https://projector.tensorflow.org/

Word Embedding



Embedding dimension d

$$\begin{bmatrix} 0.5 & 4.6 & \dots & 0.9 \\ 0.1 & -0.8 & \dots & 0.7 \\ 0.6 & 0.8 & \dots & 0.3 \\ \vdots & \vdots & & \vdots \\ 0.3 & -0.6 & \dots & -0.8 \\ -0.5 & 0.5 & \dots & 0.1 \end{bmatrix}$$

Vocabulary size

“intelligence”

Input token

Lookup Table
Embedding Weight Matrix

Positional Encoding

Let's have a sentence "This is Computer Vision Class" of $n = 5$ sequence length

And each word $\boldsymbol{x_t}$ (e.g., "Computer") is represented by an embedding vector of size for example $d = 10$ (this could be very large number)

That is mathematically t-th word is represented as

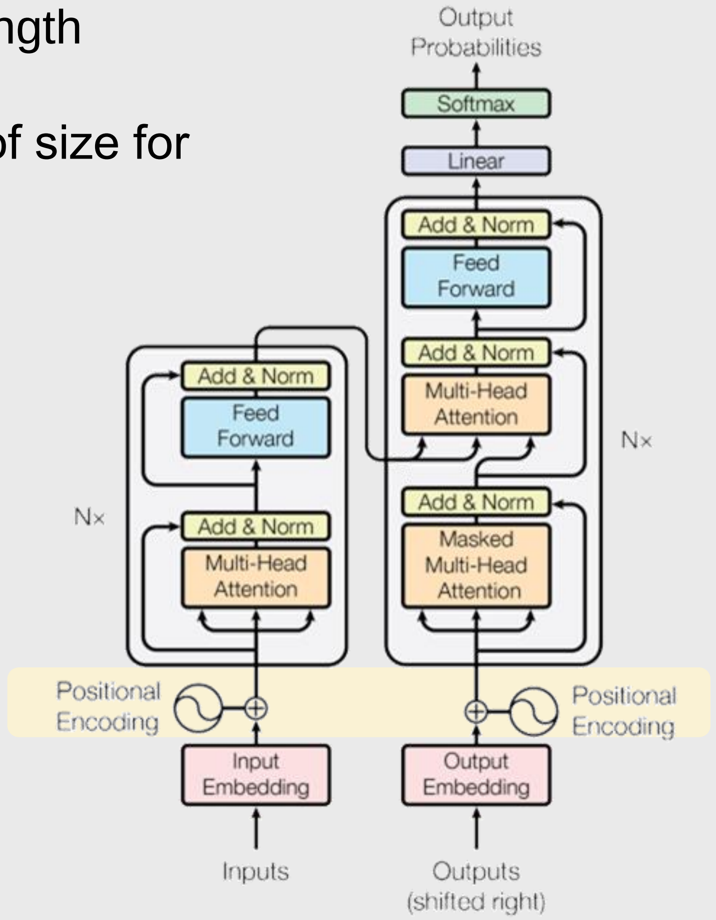
$$\boldsymbol{x_t} \in \mathbb{R}^{\boldsymbol{d}}$$

Then the positional encoding will be presented as:

$$p(pos, 2i) = sin\left(\frac{pos}{10000^{2i/d}}\right)$$

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

For $pos = 0, 1, \ldots n$ and $i = 0,1, \ldots \frac{d}{2}$



Positional Encoding of word: $\boldsymbol{x_t} \in \ \mathbb{R}^{\boldsymbol{d}}$

It assign a value relevant to the position of the word in the sentence

$$\boldsymbol{p_{pos}} = \begin{bmatrix} \sin(\omega_1) \\ \cos(\omega_1) \\ \sin(\omega_2) \\ \cos(\omega_2) \\ \vdots \\ \vdots \\ \sin(\omega_{d/2}) \\ \cos(\omega_{d/2}) \end{bmatrix}$$

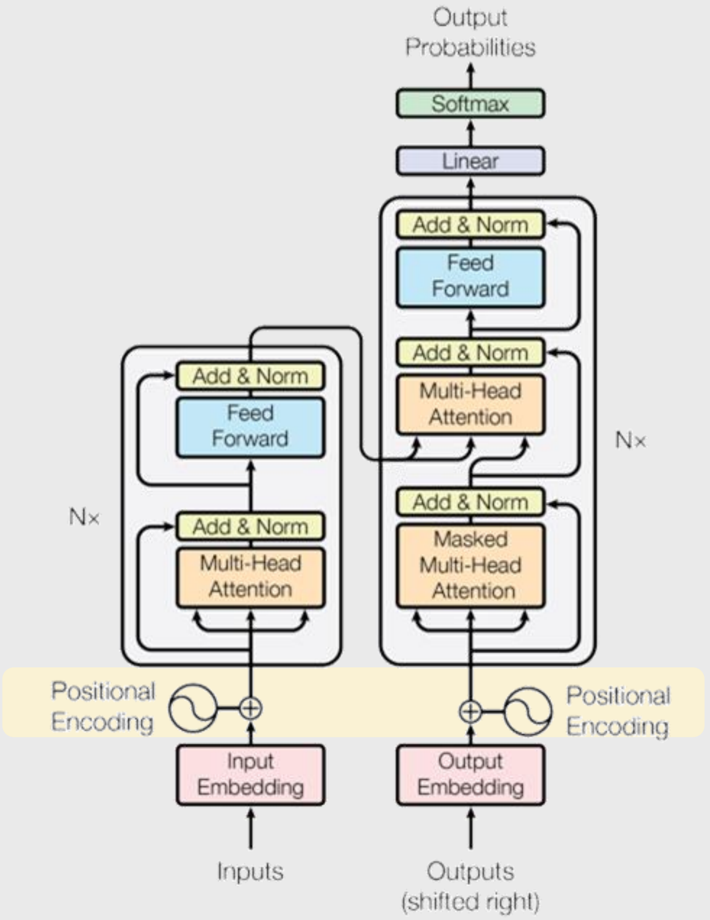
Alternating function and values

where

$$\omega_t = \frac{pos}{10000^{2i/d}}$$

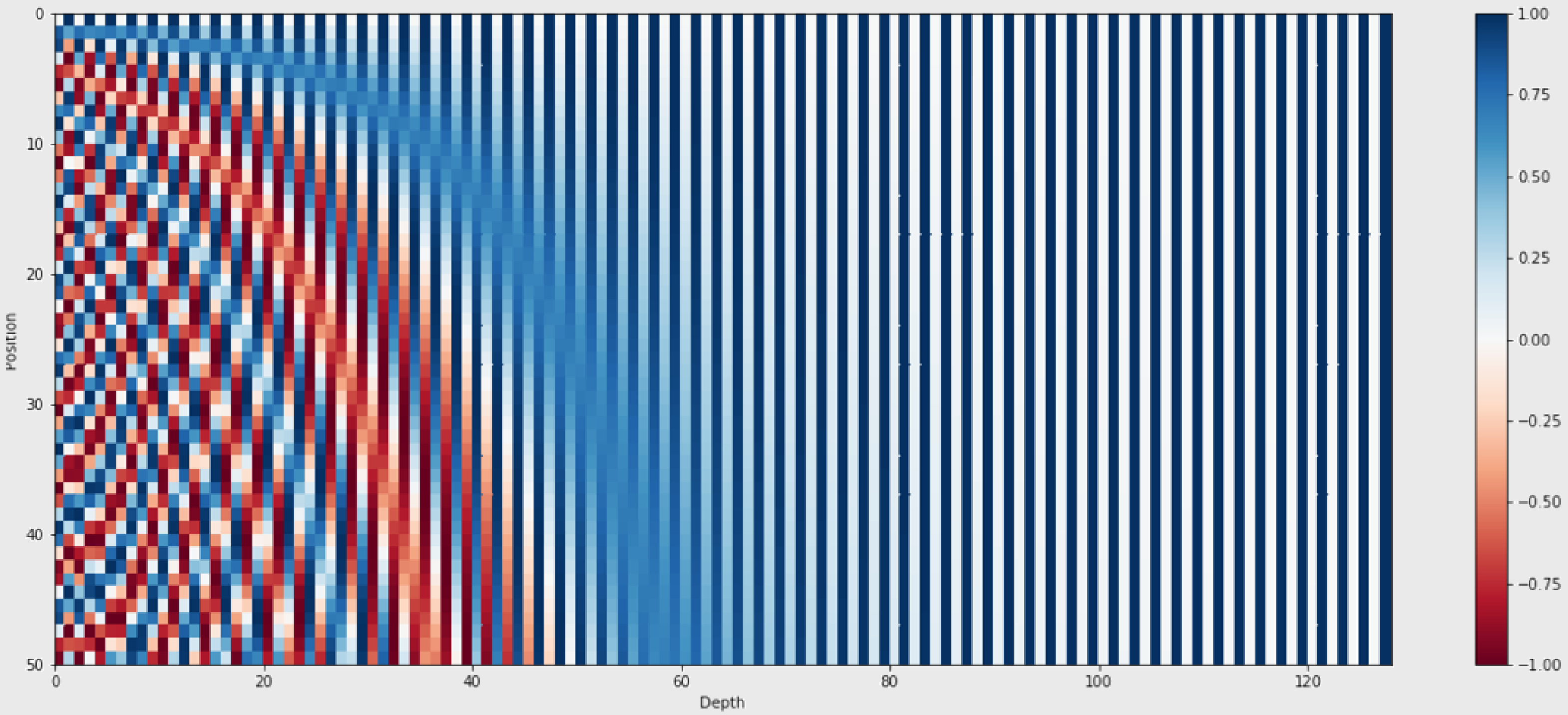
$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p_0} \\ \boldsymbol{p_1} \\ \vdots \\ \boldsymbol{p_n} \end{bmatrix} \quad n \times d$$

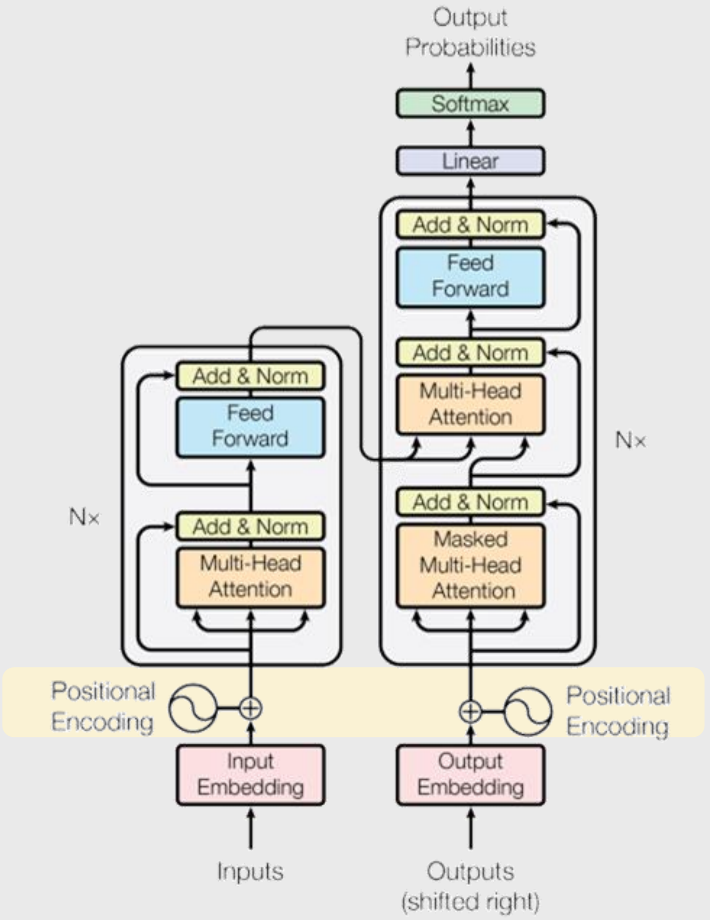
Positional encoding matrix matrix



Positional Encoding of word: $\boldsymbol{x_t} \in \mathbb{R}^{\boldsymbol{d}}$

It assign a value relevant to the position of the word in the sentence





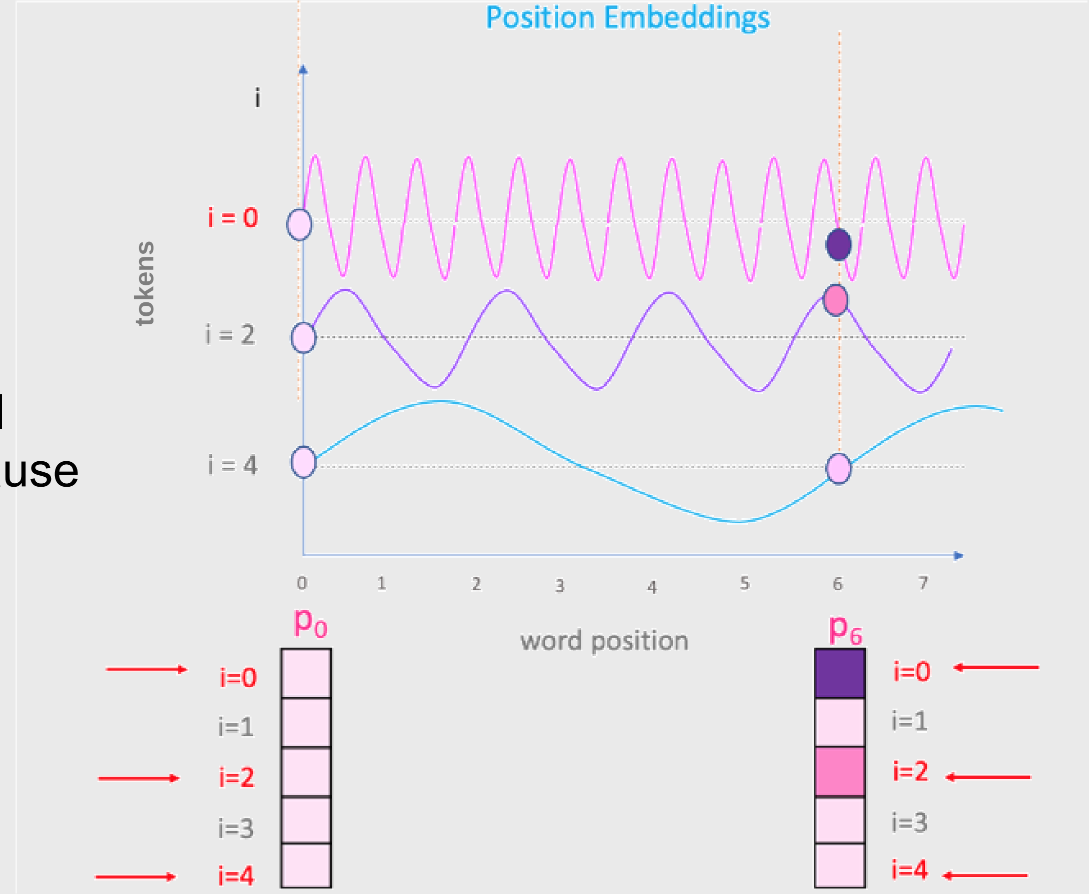
Kazemnejad et al (2024). The impact of positional encoding on length generalization in transformers. *NIPS*

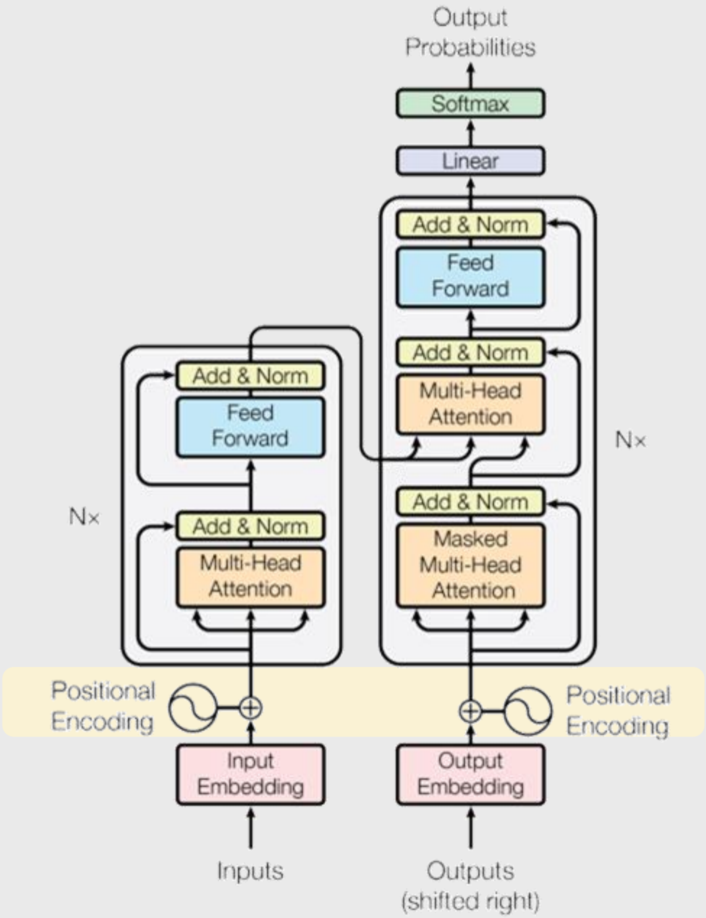
Positional Encoding of word: $\boldsymbol{x_t} \in \mathbb{R}^{\boldsymbol{d}}$

It assign a value relevant to the position of the word in the sentence

$\sin(\omega_1 \cdot t)$

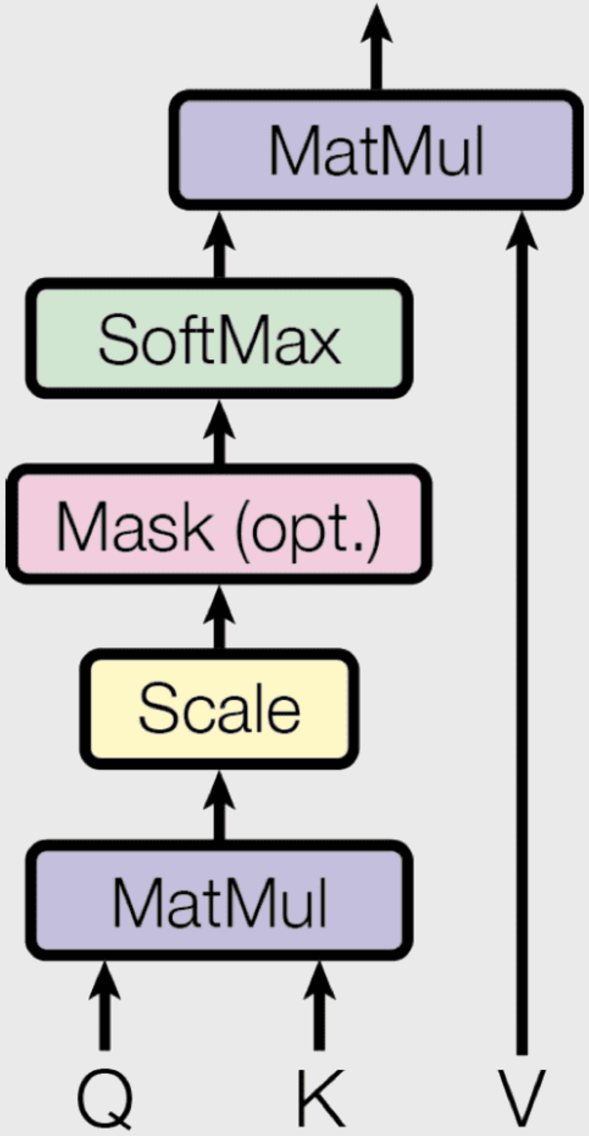
only Sinusoidal function here because $i = 2k$





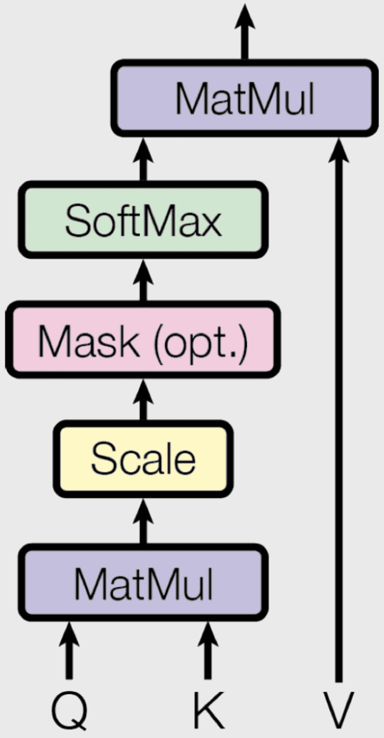
Kazemnejad et al (2024). The impact of positional encoding on length generalization in transformers. *NIPS*

Self-Attention



Vaswani et al. Attention Is All You Need (NIPS 2017)

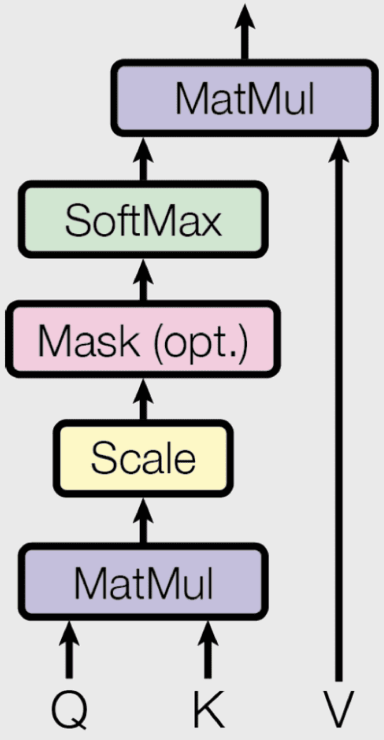
Self-Attention



$$\text{Attention}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \text{softmax}\left(\frac{\mathbf{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

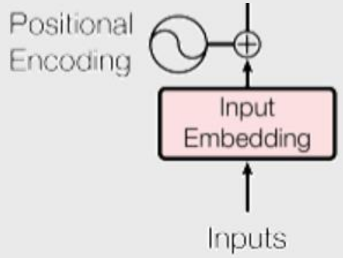
Vaswani et al. Attention Is All You Need (NIPS 2017)

Self-Attention



Let's have a sentence "This is Computer Vision Class" of $n = 5$ sequence length

And each word $\boldsymbol{x}$ (e.g., "Computer") is represented by an embedding vector of size for example $d = 10$ (this could be very large number)



That is mathematically a word is presented as

$$\boldsymbol{x^j} \in \mathbb{R}^{\boldsymbol{d}}$$

And we have weight matrices

10 x 2 10 x 2 10 x 2

$$\textbf{Query } \mathbf{W_Q} = d \times d_k\,, \qquad \textbf{Key } \mathbf{W_K} = d \times d_k, \qquad \textbf{Value } \mathbf{W_v} = d \times d_v$$

Then we perform a liner transformation of the input of $\boldsymbol{x^j}$ via Query, Key and Value matrices to obtain Query, Key and Value vectors as:

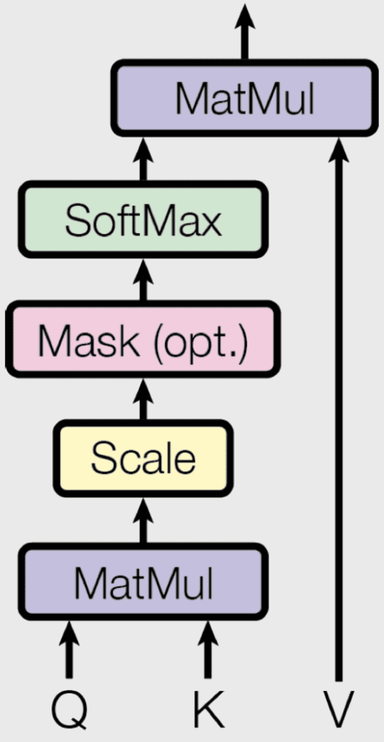
1 x 2 1 x 2 1 x 2

$$\boldsymbol{q_i^{1\times d_k}} = \boldsymbol{x_i^{1\times d}} \times \mathbf{W_Q^{\boldsymbol{d\times d_k}}}, \quad \boldsymbol{k_i^{1\times d_k}} = \boldsymbol{x_i^{1\times d}} \times \mathbf{W_K^{\boldsymbol{d\times d_k}}}, \quad \textbf{and } \boldsymbol{v_i^{1\times d_v}} = \boldsymbol{x_i^{1\times d}} \times \mathbf{W_V^{\boldsymbol{d\times d_v}}}$$

For all words $\boldsymbol{i = 1, \dots, n}$ in the sentence.

Vaswani et al. Attention Is All You Need (NIPS 2017)

Self-Attention



We can pack the following Query, Key and Value vectors into a matrix forms:

1 x 2 $\quad$ 1 x 2 $\quad$ 1 x 2

$q_i^{1\times d_k} = x_i^{1\times d} \times \mathrm{W}_{\mathrm{Q}}^{d\times d_k}$, $k_i^{1\times d_k} = x_i^{1\times d} \times \mathrm{W}_{\mathrm{K}}^{d\times d_k}$, and $v_i^{1\times d_k} = x_i^{1\times d} \times \mathrm{W}_{\mathrm{V}}^{d\times d_k}$

For all words $i = 1, \dots, n$ in the sentence.

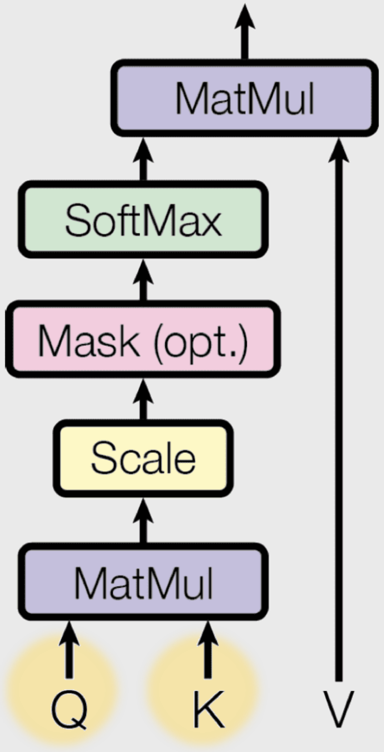
$\mathbf{Q} = [q_1 \quad q_2 \quad \cdots \quad q_n]$ $\quad$ $\mathbf{K} = [k_1 \quad k_2 \quad \dots \quad k_n]$ $\quad$ $\mathbf{V} = [v_1 \quad v_2 \quad \cdots \quad v_n]$

1 2 5 $\quad$ 1 2 5 $\quad$ 1 2 5

Self-Attention

Vaswani et al. Attention Is All You Need (NIPS 2017)

$\boldsymbol{n}$ is the number of tokens in a sentence



$\mathbf{Q}$ is a matrix of size $n \times d_k$

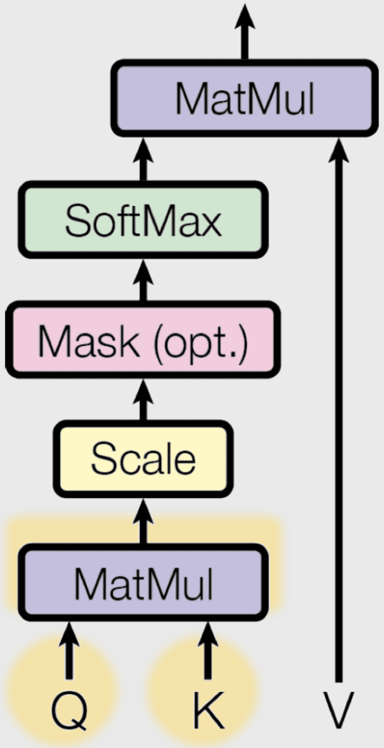
$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix}$$ 5 x 2

$$\text{Attention}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix}$$ 5 x 2

$\mathbf{K}$ is a matrix of size $n \times d_k$

Self-Attention



$\mathbf{Q}$ is a matrix of size $n \times d_k$

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix}$$ 5 x 2

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

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix}$$ 5 x 2

$\mathbf{K}$ is a matrix of size $n \times d_k$

dot product

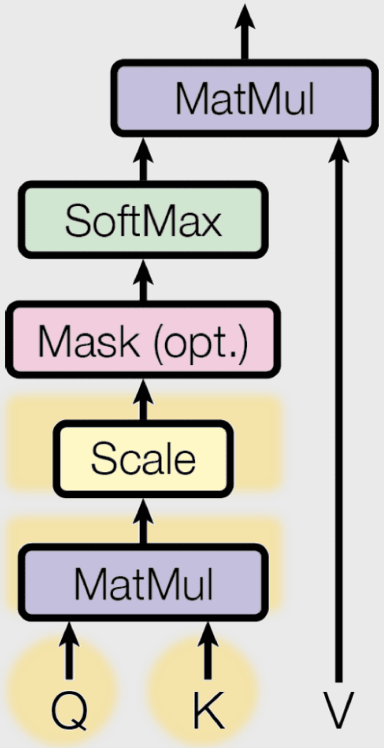
$$\begin{bmatrix} q_{1,1} & \cdots & q_{1,d_k} \\ q_{2,1} & \ddots & q_{2,d_k} \\ \vdots & & \vdots \\ q_{n,1} & \cdots & q_{n,d_k} \end{bmatrix} \bullet \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix}$$

5 x 2   2 x 5

alignment scores

$$\begin{bmatrix} \boldsymbol{e}_{1,1} & \boldsymbol{e}_{1,2} & \cdots & \boldsymbol{e}_{1,n} \\ \boldsymbol{e}_{2,1} & \boldsymbol{e}_{2,2} & \cdots & \boldsymbol{e}_{2,n} \\ \vdots & \cdots & & \vdots \\ \boldsymbol{e}_{n,1} & \boldsymbol{e}_{n,2} & \cdots & \boldsymbol{e}_{n,n} \end{bmatrix}$$ 5 x 5

Self-Attention



Q is a matrix of size $n \times d_k$

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix}$$ 5 x 2

$$\begin{bmatrix} \frac{e_{1,1}}{\sqrt{d_k}} & \frac{e_{1,2}}{\sqrt{d_k}} & \cdots & \frac{e_{1,n}}{\sqrt{d_k}} \\ \frac{e_{2,1}}{\sqrt{d_k}} & \frac{e_{2,2}}{\sqrt{d_k}} & \cdots & \frac{e_{2,n}}{\sqrt{d_k}} \\ \vdots & \cdots & & \vdots \\ \frac{e_{n,1}}{\sqrt{d_k}} & \frac{e_{n,2}}{\sqrt{d_k}} & \cdots & \frac{e_{n,n}}{\sqrt{d_k}} \end{bmatrix}$$ 5 x 5

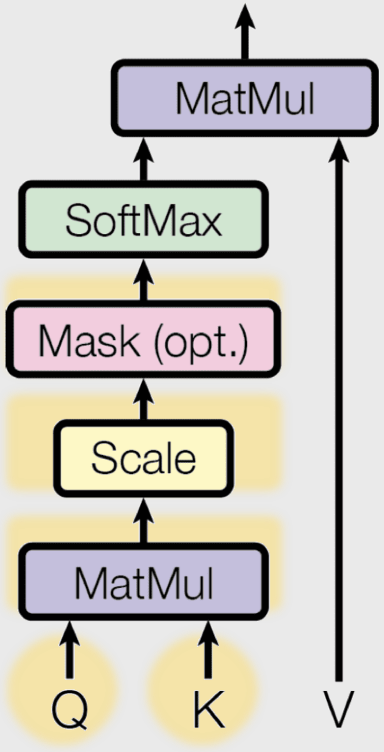
Scaling of alignment scores

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

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix}$$ 5 x 2

K is a matrix of size $n \times d_k$

Self-Attention



$\mathbf{Q}$ is a matrix of size $n \times d_k$

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix} \text{5 x 2}$$

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

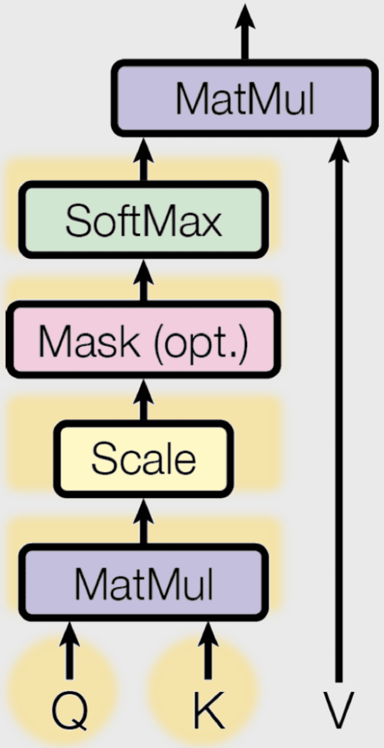
Apply Mask (optional)

$$\begin{bmatrix} \frac{e_{1,1}}{\sqrt{d_k}} & -\infty & & -\infty \\ \frac{e_{2,1}}{\sqrt{d_k}} & \frac{e_{2,2}}{\sqrt{d_k}} & \cdots & -\infty \\ \vdots & \cdots & & \vdots \\ \frac{e_{n,1}}{\sqrt{d_k}} & \frac{e_{n,2}}{\sqrt{d_k}} & \cdots & \frac{e_{n,n}}{\sqrt{d_k}} \end{bmatrix} \text{5 x 5}$$

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix} \text{5 x 2}$$

$\mathbf{K}$ is a matrix of size $n \times d_k$

Self-Attention



$\mathbf{Q}$ is a matrix of size $n \times d_k$

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix}$$ 5 x 2

$$\begin{bmatrix} \mathbf{softmax}\left(\frac{\boldsymbol{e}_{1,1}}{\sqrt{d_k}} \quad \mathbf{0} \quad \cdots \quad \mathbf{0}\right) \\ \mathbf{softmax}\left(\frac{\boldsymbol{e}_{2,1}}{\sqrt{d_k}} \quad \frac{\boldsymbol{e}_{2,2}}{\sqrt{d_k}} \quad \cdots \quad \mathbf{0}\right) \\ \vdots \\ \mathbf{softmax}\left(\frac{\boldsymbol{e}_{n,1}}{\sqrt{d_k}} \quad \frac{\boldsymbol{e}_{n,2}}{\sqrt{d_k}} \quad \cdots \quad \frac{\boldsymbol{e}_{n,n}}{\sqrt{d_k}}\right) \end{bmatrix}$$ 5 x 5

SoftMax of Scaled dot product

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

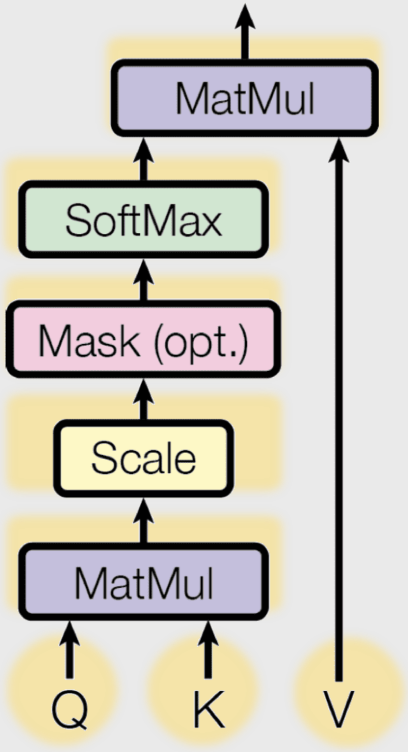
$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix}$$ 5 x 2

$\frac{1}{\sqrt{d_k}}$ is the scaling factor

$\mathbf{K}$ is a matrix of size $n \times d_k$

Self-Attention

Vaswani et al. Attention Is All You Need (NIPS 2017)



$$\begin{bmatrix} \mathrm{softmax}\left(\frac{e_{1,1}}{\sqrt{d_k}} \quad 0 \quad \cdots \quad 0\right) \\ \mathrm{softmax}\left(\frac{e_{2,1}}{\sqrt{d_k}} \quad \frac{e_{2,2}}{\sqrt{d_k}} \quad \cdots \quad 0\right) \\ \vdots \\ \mathrm{softmax}\left(\frac{e_{d_k,1}}{\sqrt{d_k}} \quad \frac{e_{d_k,2}}{\sqrt{d_k}} \quad \cdots \quad \frac{e_{d_k,d_k}}{\sqrt{d_k}}\right) \end{bmatrix}_{5 \times 5} \cdot \begin{bmatrix} v_{1,1} & \cdots & v_{1,d_v} \\ v_{2,1} & \ddots & q_{2,d_v} \\ \vdots & & \vdots \\ v_{n,1} & \cdots & v_{n,d_v} \end{bmatrix}_{5 \times 2}$$

resulting self-attention weights

Q is a matrix of size $n \times d_k$

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix} \quad 5 \times 2$$

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}K^T}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix} \quad 5 \times 2$$

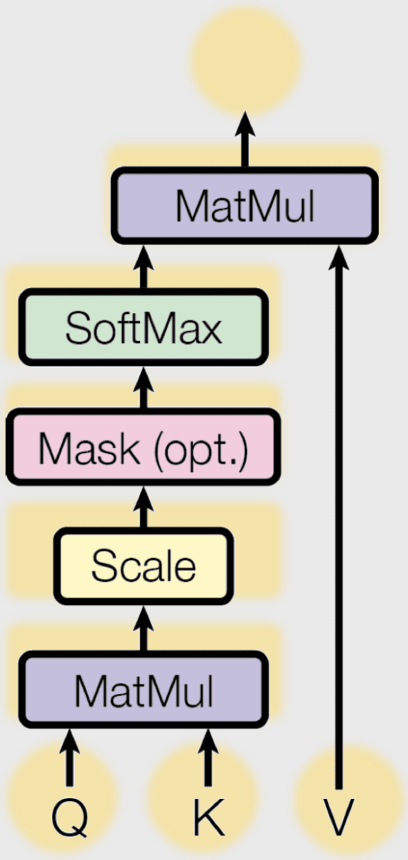
$$\mathbf{V} = \begin{bmatrix} v_{1,1} & \cdots & v_{1,d_v} \\ v_{2,1} & \ddots & q_{2,d_v} \\ \vdots & & \vdots \\ v_{n,1} & \cdots & v_{n,d_v} \end{bmatrix} \quad 5 \times 2$$

K is a matrix of size $n \times d_k$

V is a matrix of size $n \times d_v$

Self-Attention

Vaswani et al. Attention Is All You Need (NIPS 2017)



$\mathbf{Q}$ is a matrix of size $n \times d_k$

$$\mathbf{Q} = \begin{bmatrix} q_{1,1} & \cdots & q_{n,1} \\ q_{1,2} & \ddots & q_{n,2} \\ \vdots & & \vdots \\ q_{1,d_k} & \cdots & q_{n,d_k} \end{bmatrix}$$

$$\mathbf{Head_1} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,d_v} \\ a_{2,1} & \ddots & a_{2,d_v} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,d_v} \end{bmatrix}$$ 5 x 2

resulting self-attention weights

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

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & \cdots & k_{n,1} \\ k_{1,2} & \ddots & k_{n,2} \\ \vdots & & \vdots \\ k_{1,d_k} & \cdots & k_{n,d_k} \end{bmatrix}$$

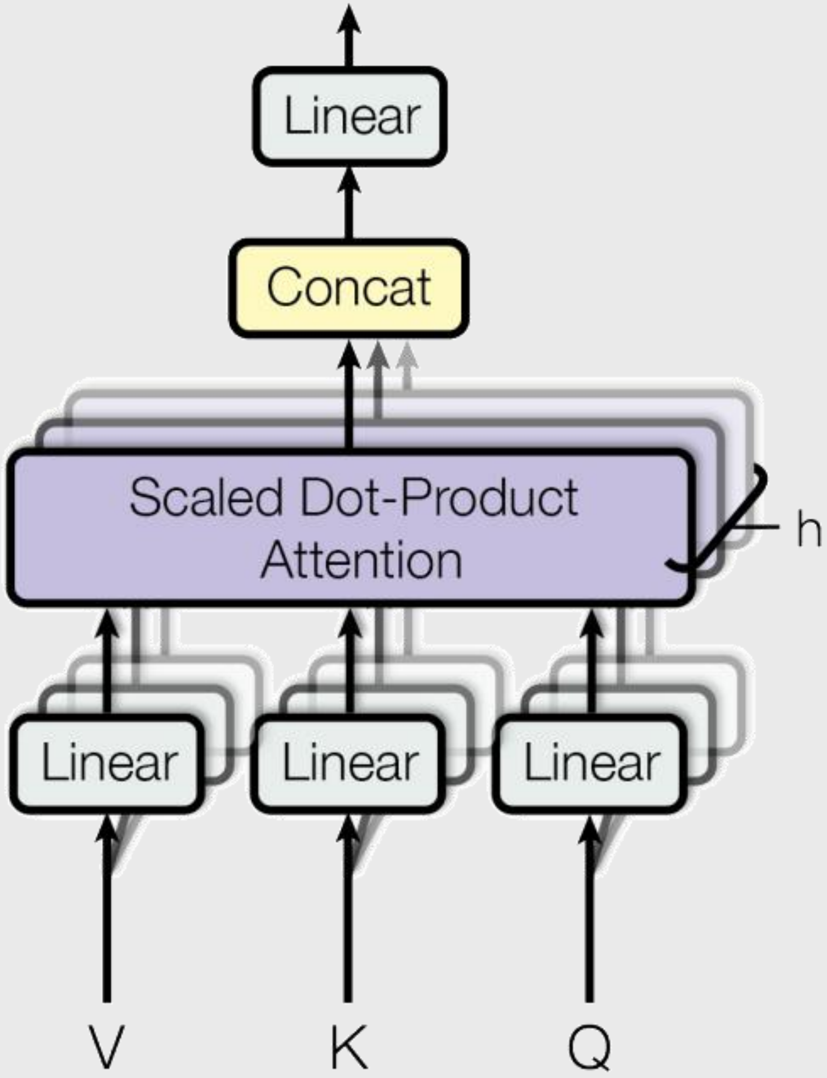
$$\mathbf{V} = \begin{bmatrix} v_{1,1} & \cdots & v_{1,d_v} \\ v_{2,1} & \ddots & q_{2,d_v} \\ \vdots & & \vdots \\ v_{d,1} & \cdots & v_{d,d_v} \end{bmatrix}$$

$\mathbf{K}$ is a matrix of size $n \times d_k$

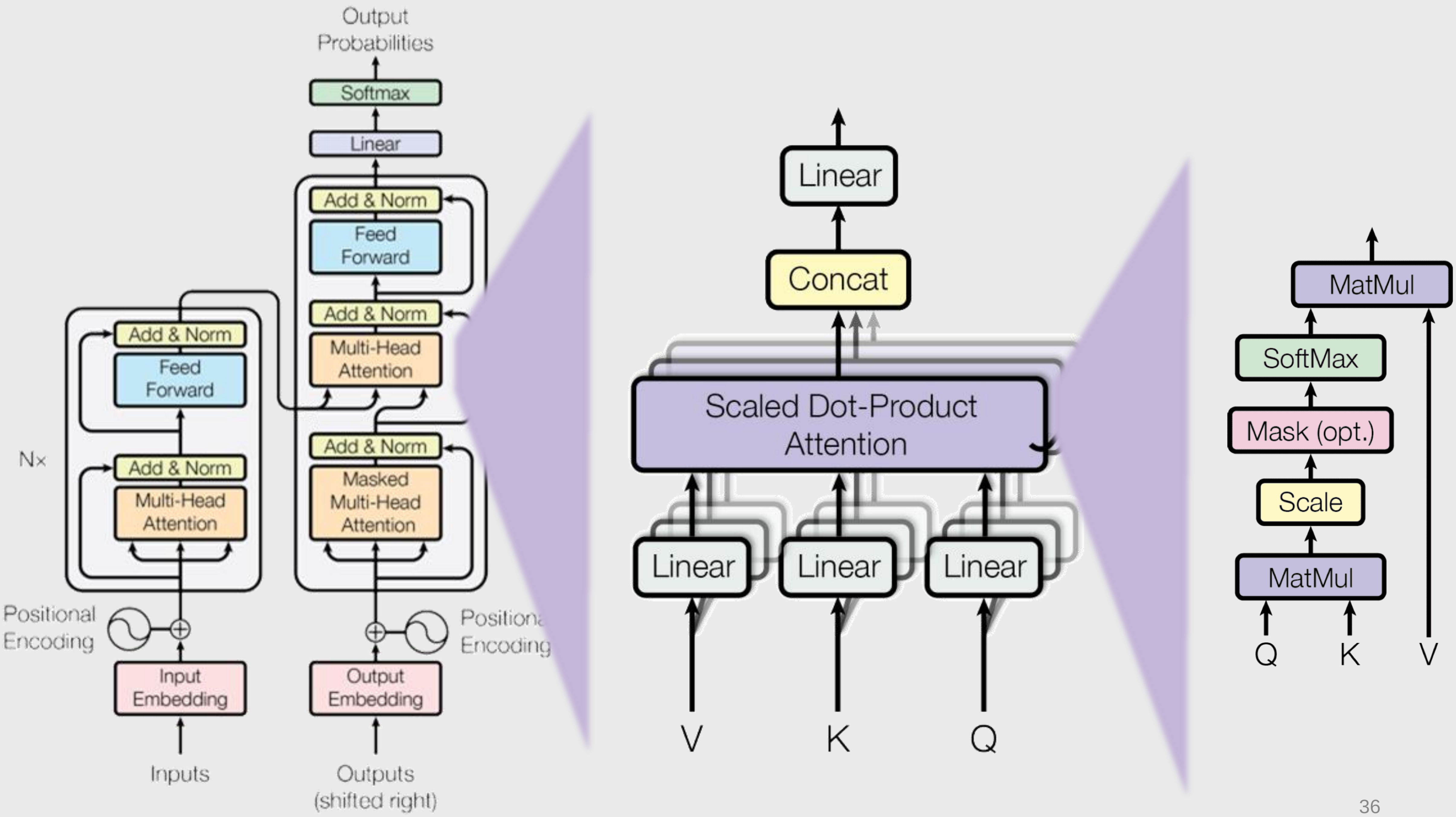
$\mathbf{V}$ is a matrix of size $d \times d_v$

Multi Head Self-Attention

Vaswani et al. Attention Is All You Need (NIPS 2017)



$$\mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{Head}_1, \mathrm{Head}_2, \cdots, \mathrm{Head}_h)\,\mathbf{W}_{\mathbf{Out}}$$

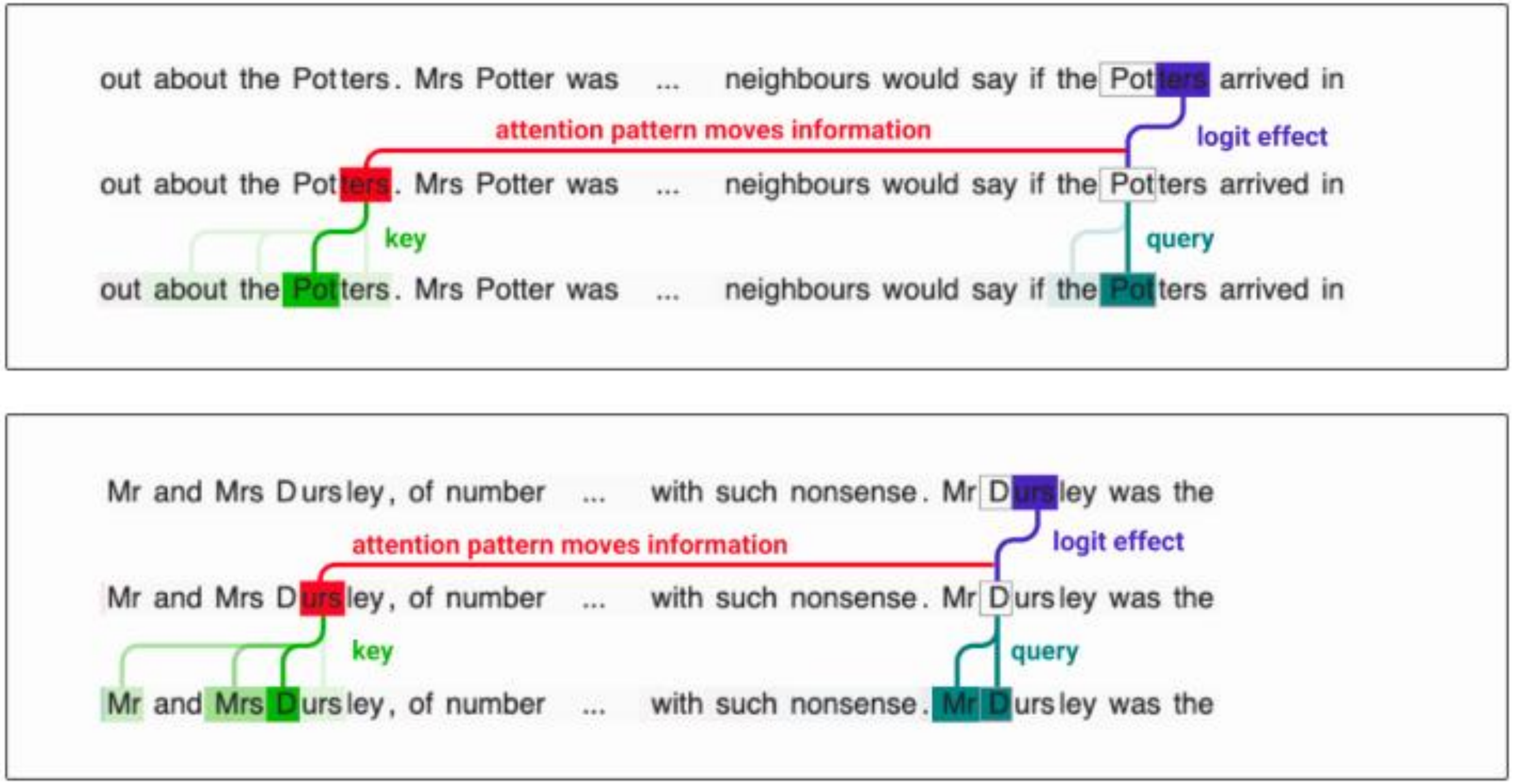

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Linear
Concat
Scaled Dot-Product Attention
Linear
Linear
Linear
V
K
Q
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

Attention Map

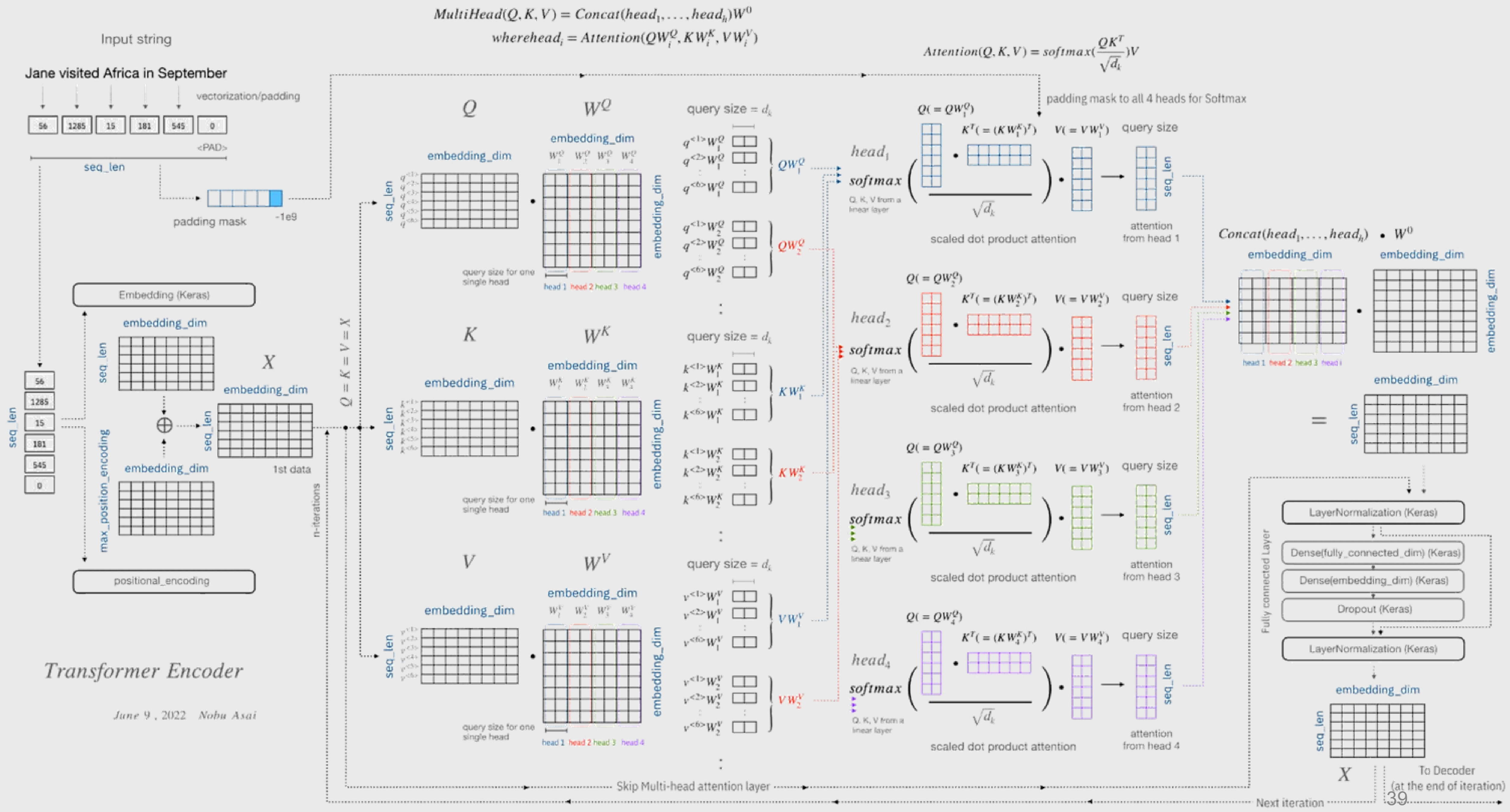
A Mathematical Framework for Transformer Circuitshttps://transformer-circuits.pub/2021/framework/index.html

How Query and Key might work

The query searches for "similar" key vectors, but because keys are shifted, it finds the next token.

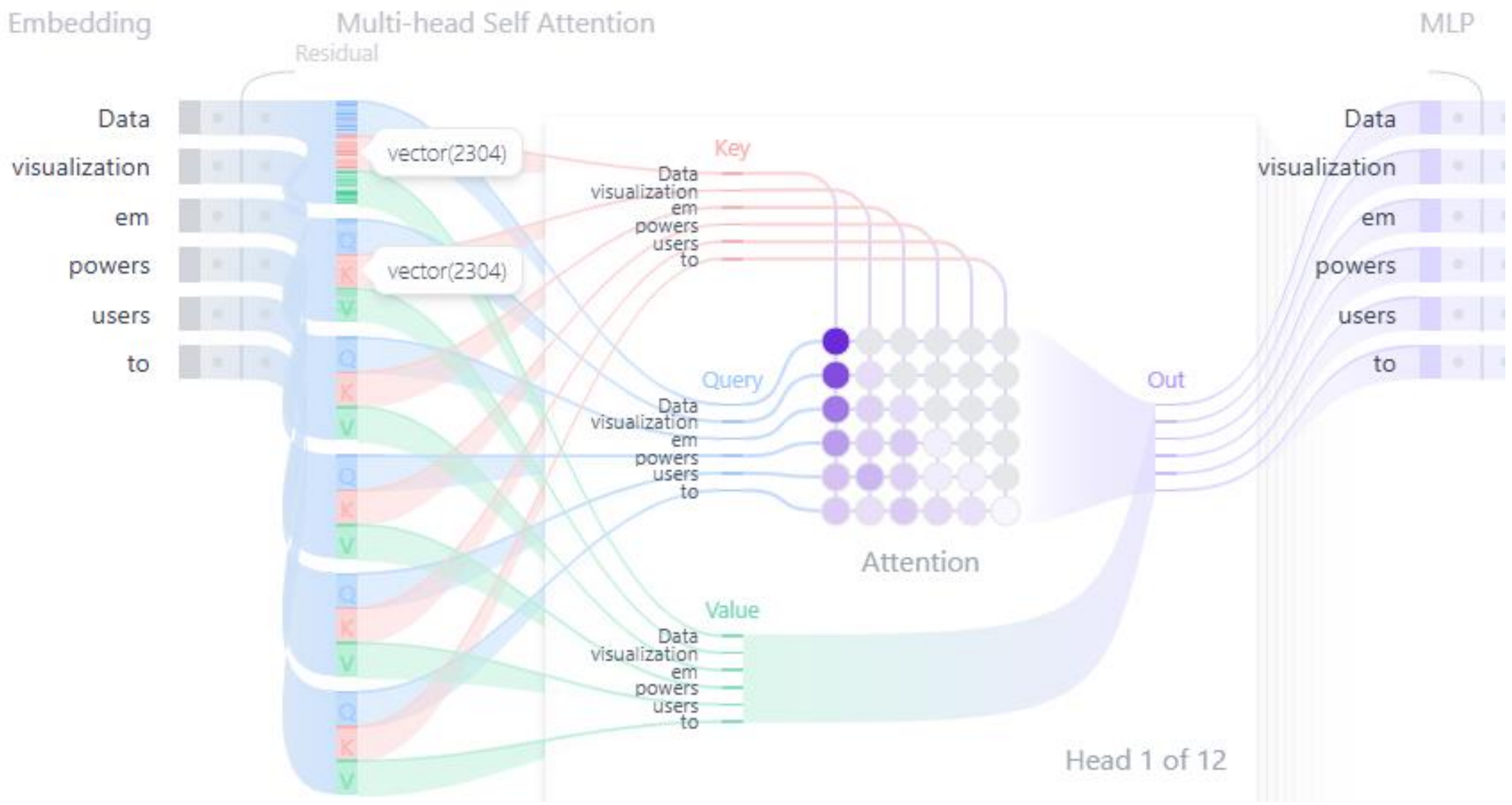


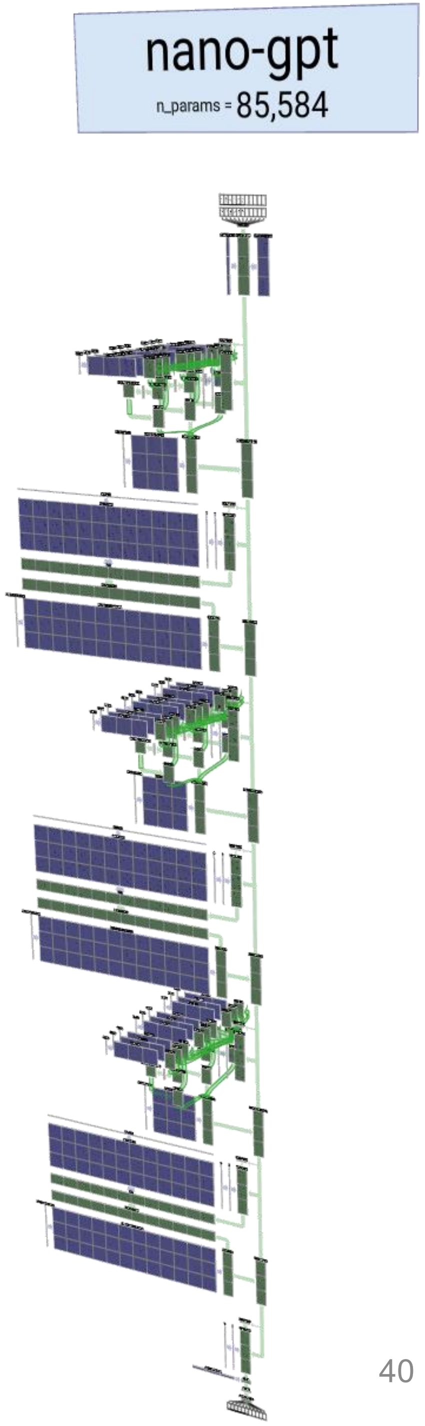
A Mathematical Framework for Transformer Circuitshttps://transformer-circuits.pub/2021/framework/index.html



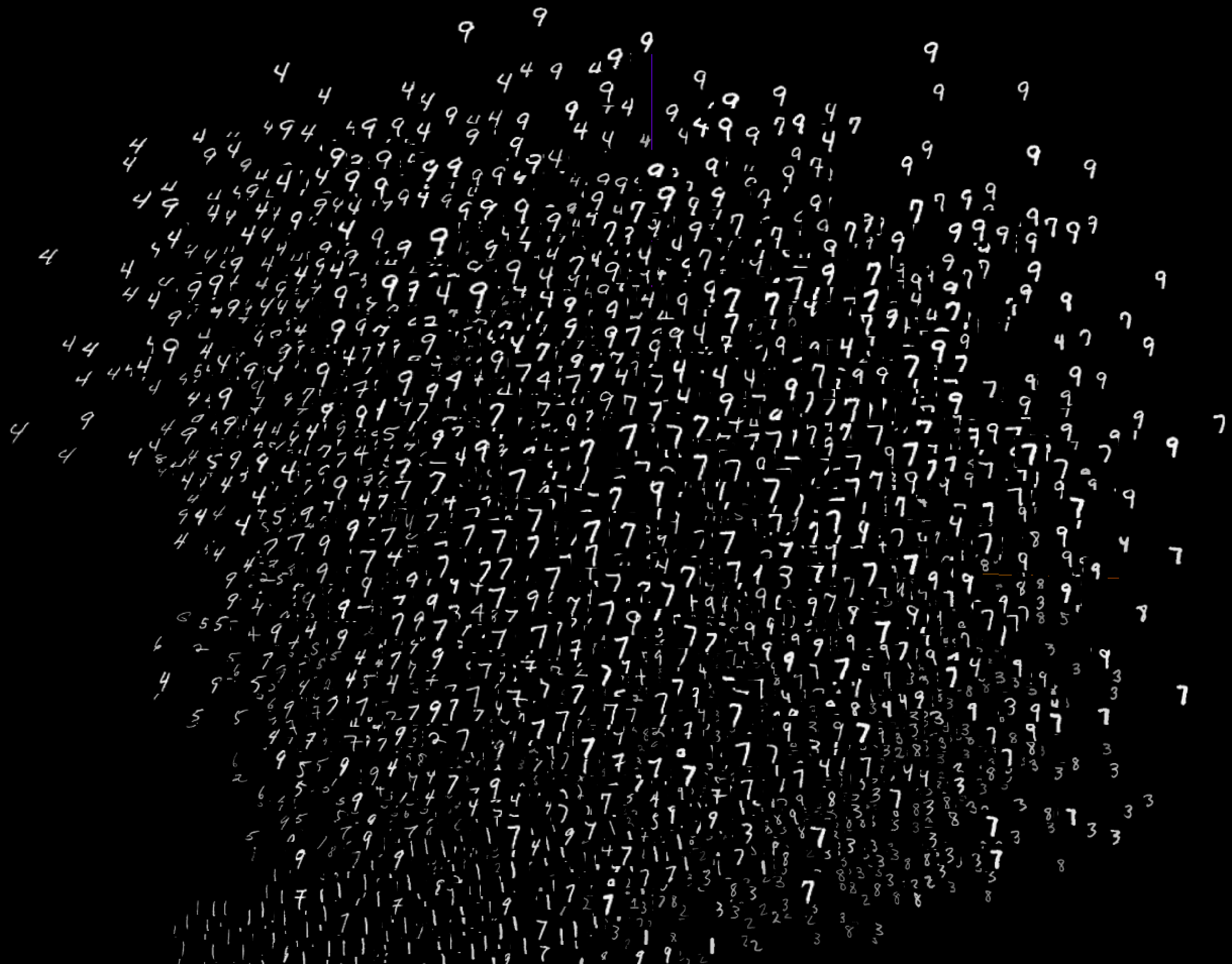
Source: https://community.deeplearning.ai/t/w4-a1-is-there-a-typo-in-multi-head-attention-slides/135478

Transformer Visualisations & Explainers (Online Resources)

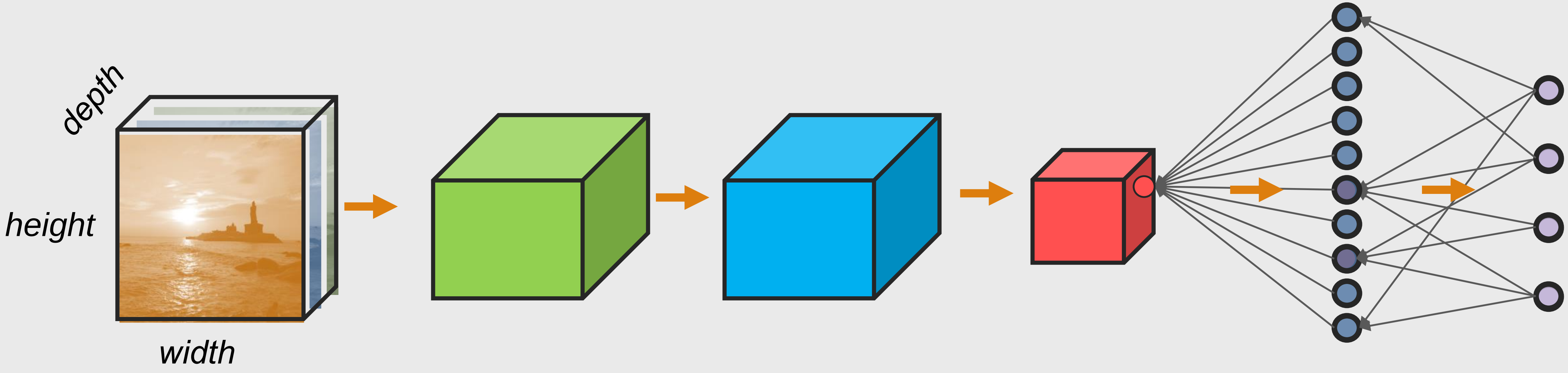




- https://bbycroft.net/llm
- https://poloclub.github.io/transformer-explainer/
- https://jalammar.github.io/illustrated-transformer/



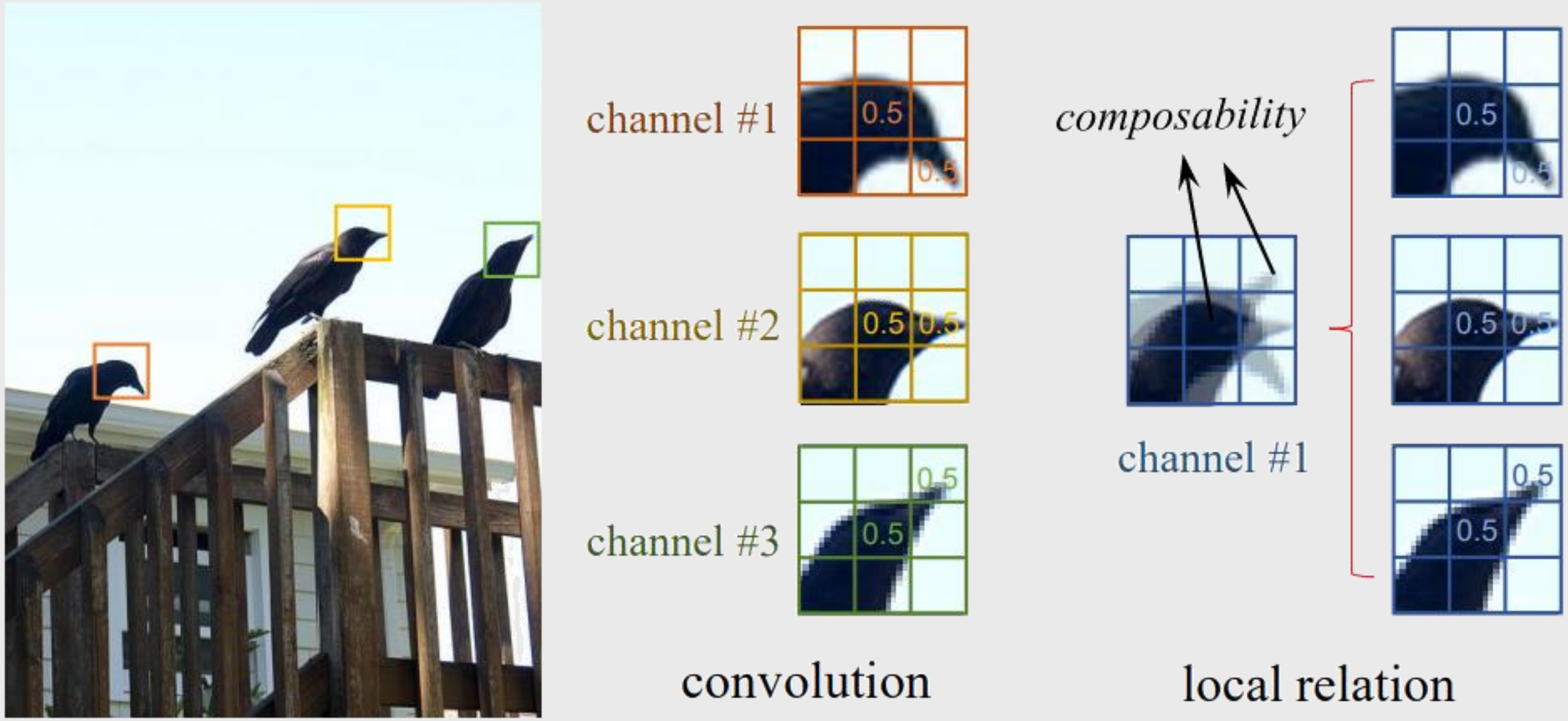
Convolutional Neural Nets



[Input - Convolution - RELU - POOLING - Fully Connected - Output]

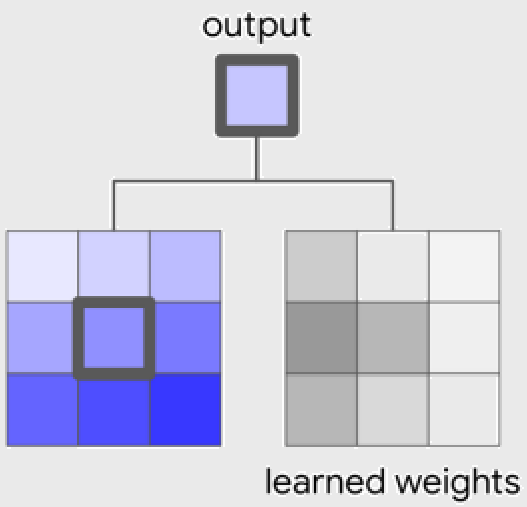
3×3 convolution layer and the 3×3 local relation layer

Hu et al. (2019). Local relation networks for image recognition. ICCV

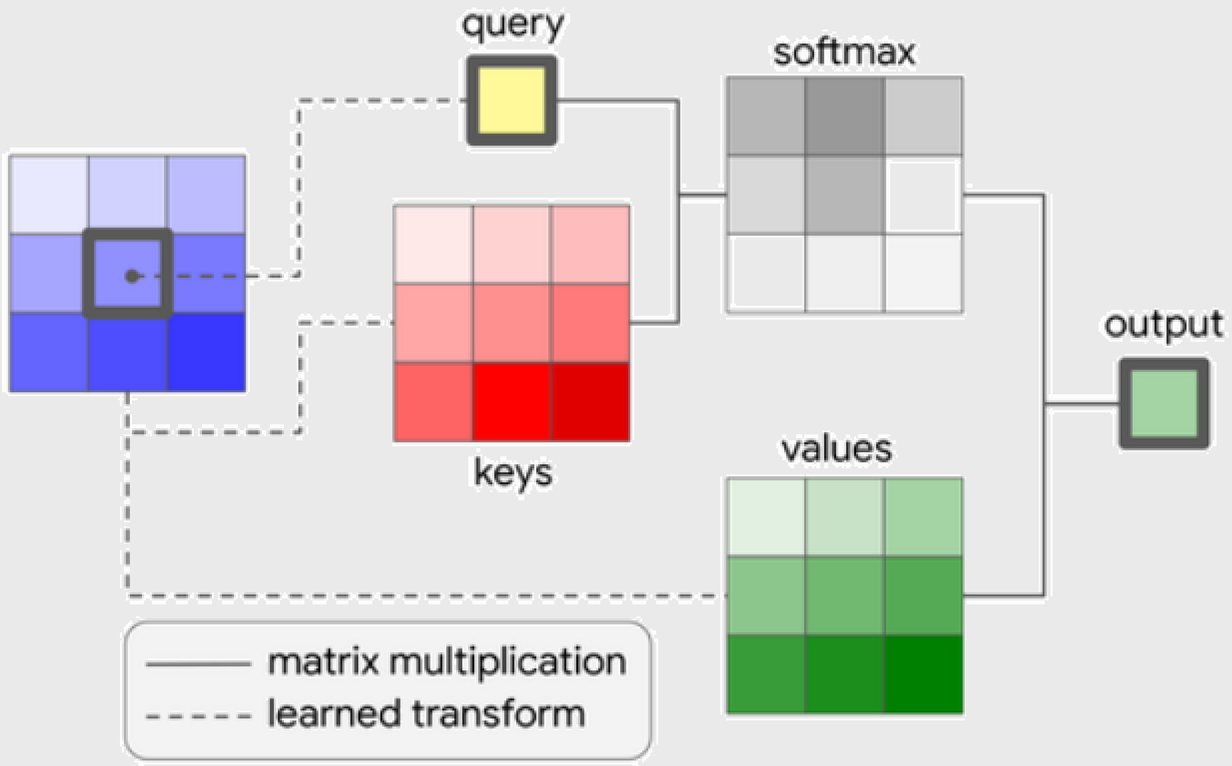


Convolutional Nets Vs Transformer

Ramachandran et al. Stand-alone self-attention in vision models. NIPS 2019



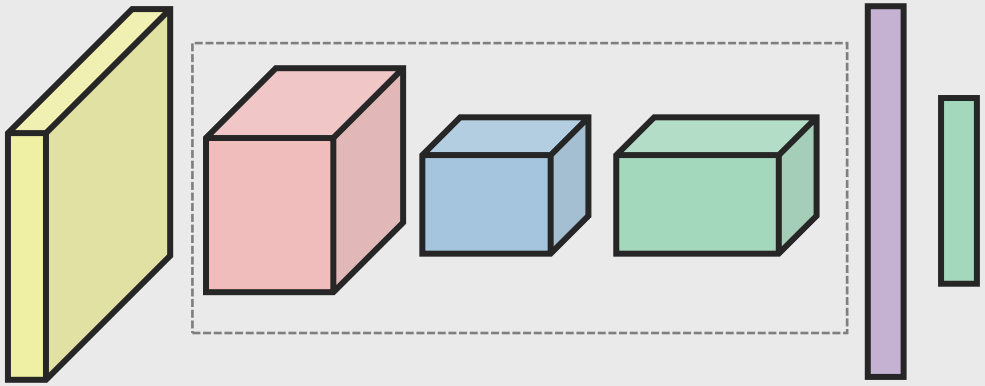
3 × 3 convolution.
The output is the inner product between the local window and the learned weights

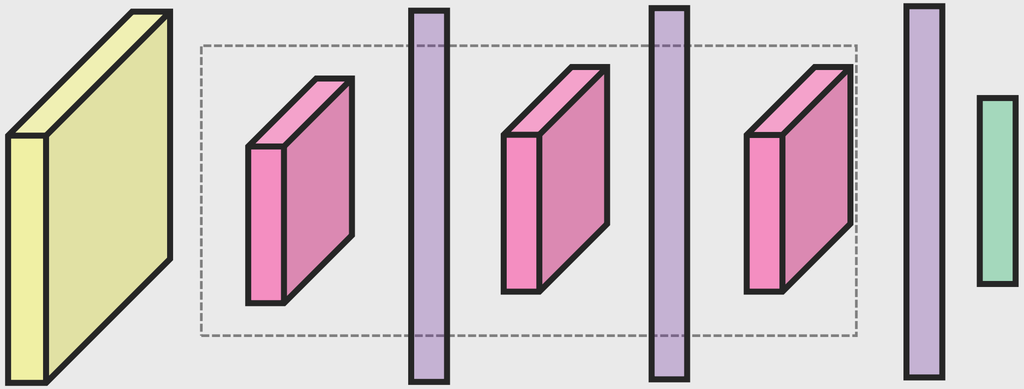


Self-attention around image local region
The output is local self attention

Standalone Self-attention in Vision Models

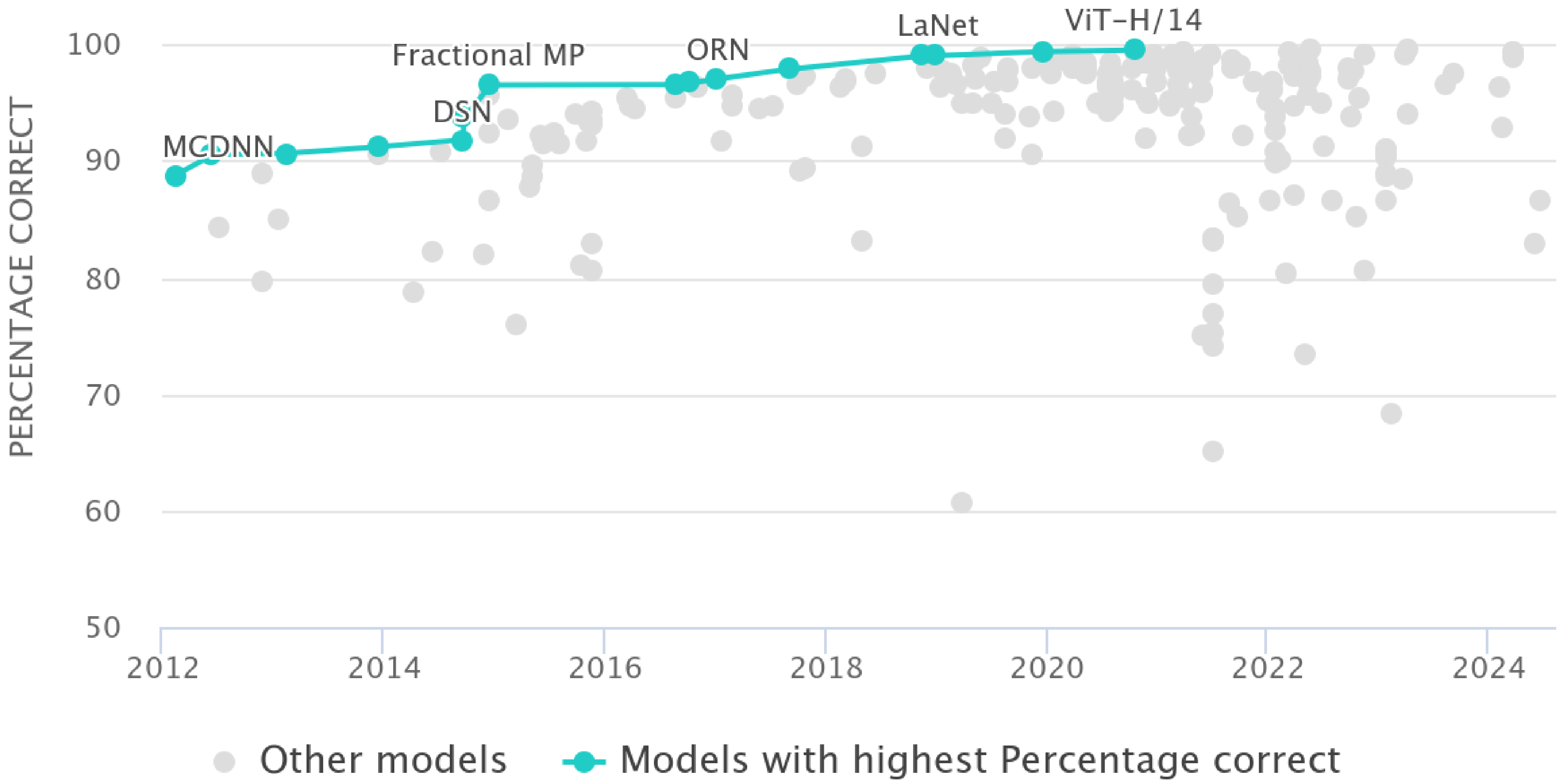
Ramachandran et al. Stand-alone self-attention in vision models. NIPS 2019





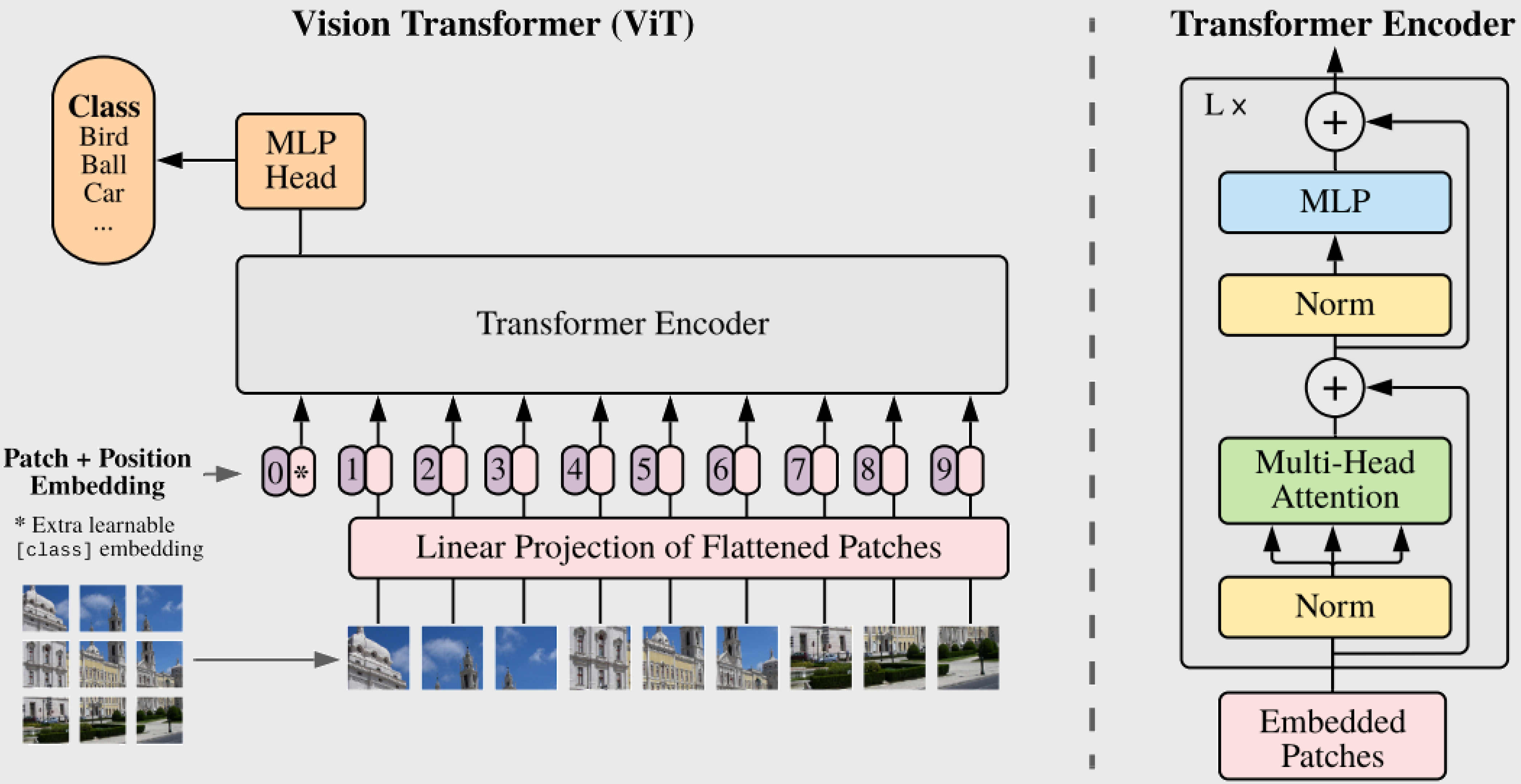
	ResNet-26			ResNet-38			ResNet-50		
	FLOPS (B)	Params (M)	Acc. (%)	FLOPS (B)	Params (M)	Acc. (%)	FLOPS (B)	Params (M)	Acc. (%)
Baseline	4.7	13.7	74.5	6.5	19.6	76.2	8.2	25.6	76.9
Conv-stem + Attention	4.5	10.3	**75.8**	5.7	14.1	**77.1**	7.0	18.0	**77.4**
Full Attention	4.7	10.3	74.8	6.0	14.1	76.9	7.2	18.0	**77.6**

Convolutional Nets Vs Transformers



Check Latest Models Here: https://paperswithcode.com/sota/image-classification-on-cifar-10

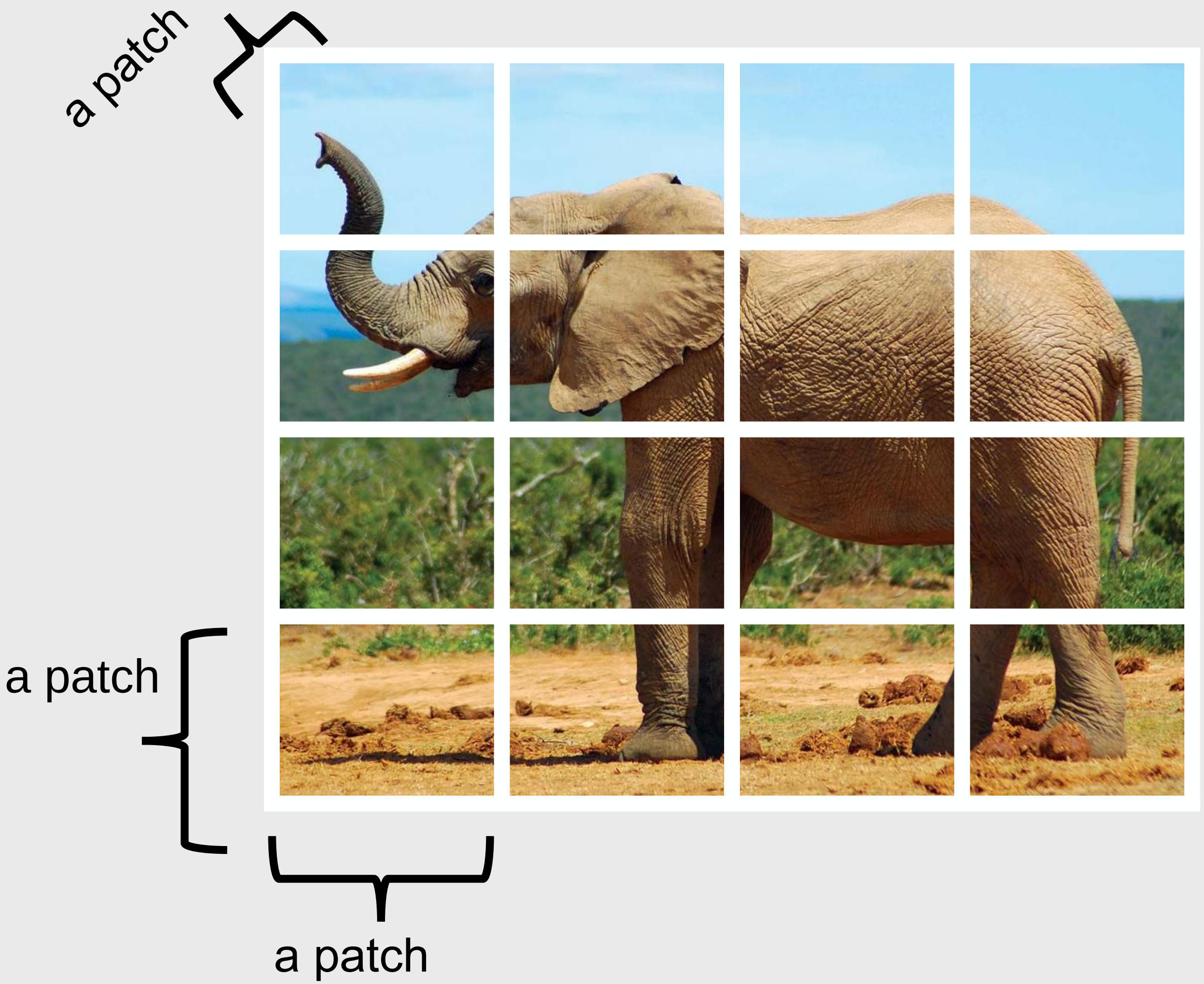
How Vision Transformer Models Works

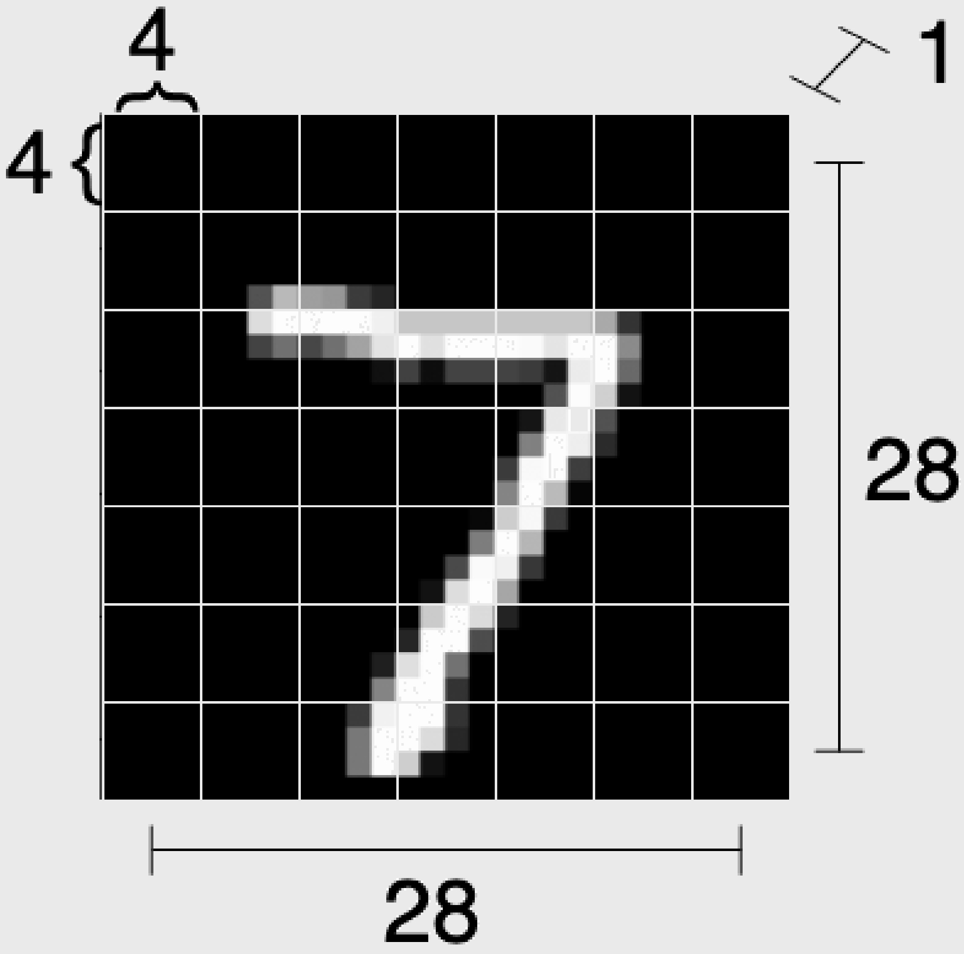


Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2017

Splitting an Image into Patches

Split the image into patches, each of size (H’xW’xD)

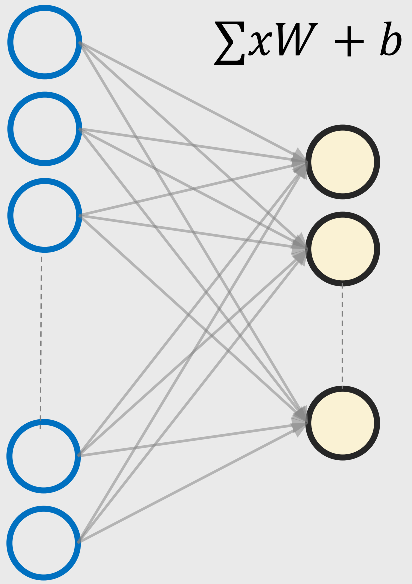


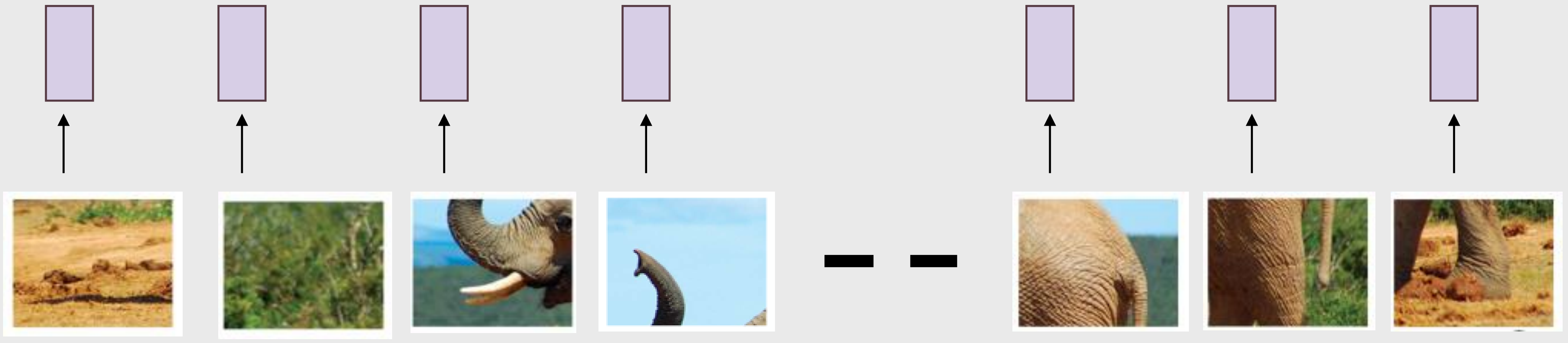


MIST dataset

Linear mapping

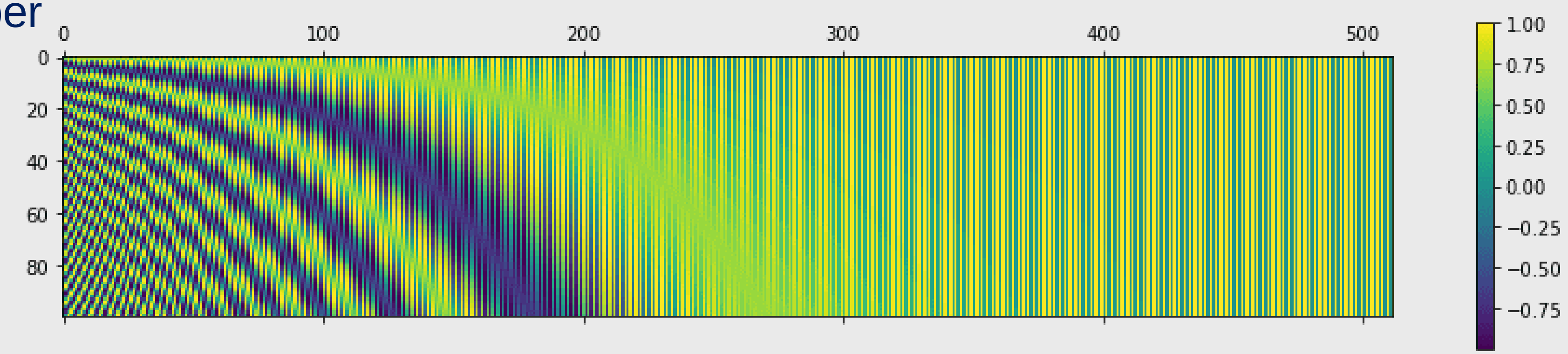
Linear projection to D-dimensional vector





Positional Encoding

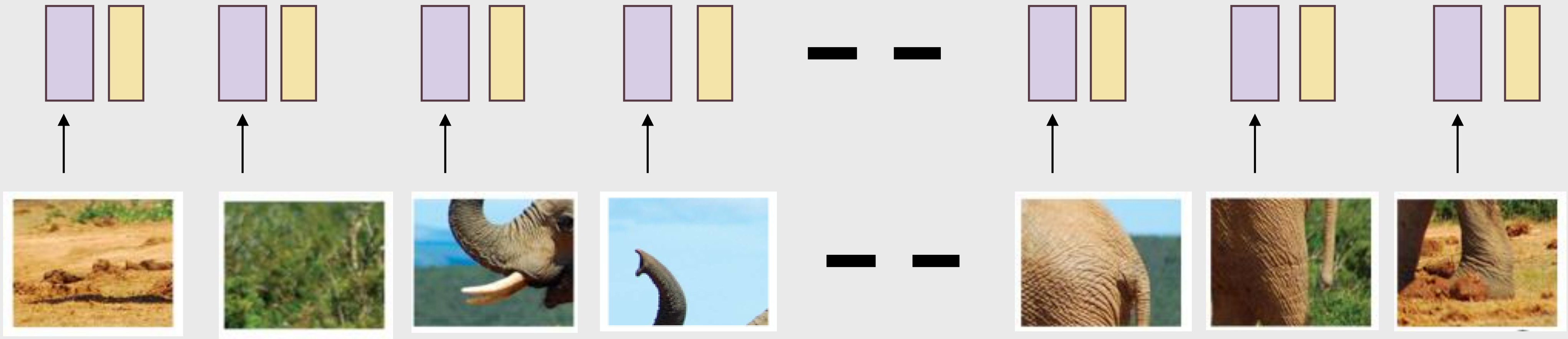
Inform the model where the patch's position in the image is. In other word use sine and cosine values for respective patch number



$$p_{i,j} = \begin{cases} \sin\left(\dfrac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is even} \\ \cos\left(\dfrac{i}{10000^{\frac{j-1}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is odd} \end{cases}$$

Linear Mapping h-dim

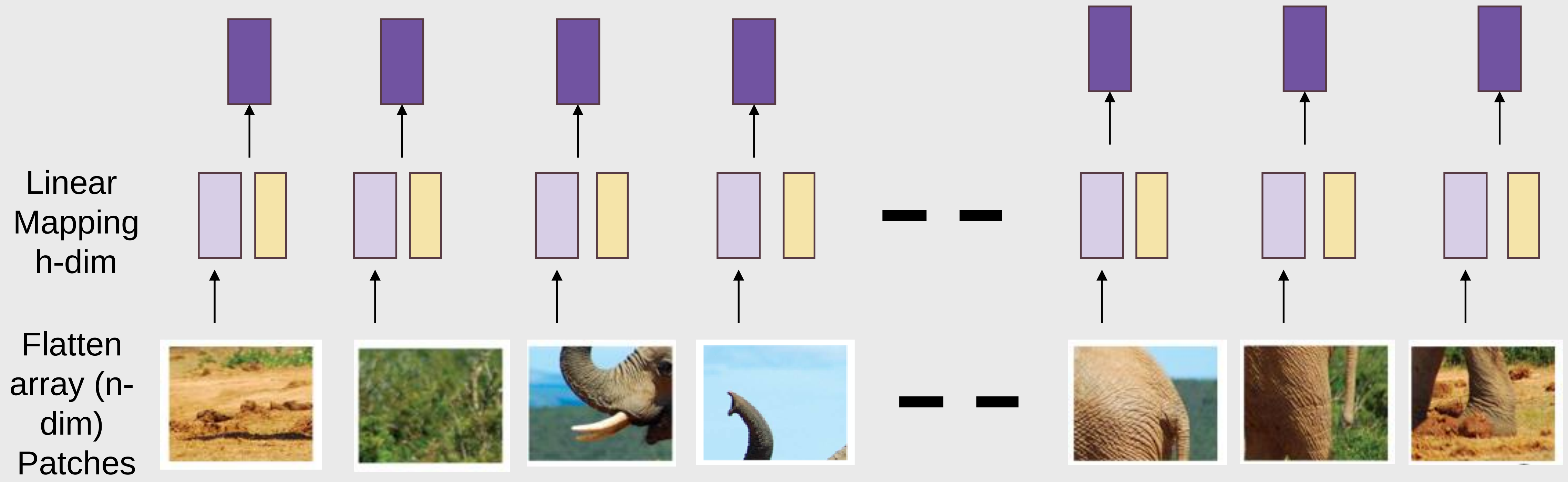
Flatten array (n-dim) Patches



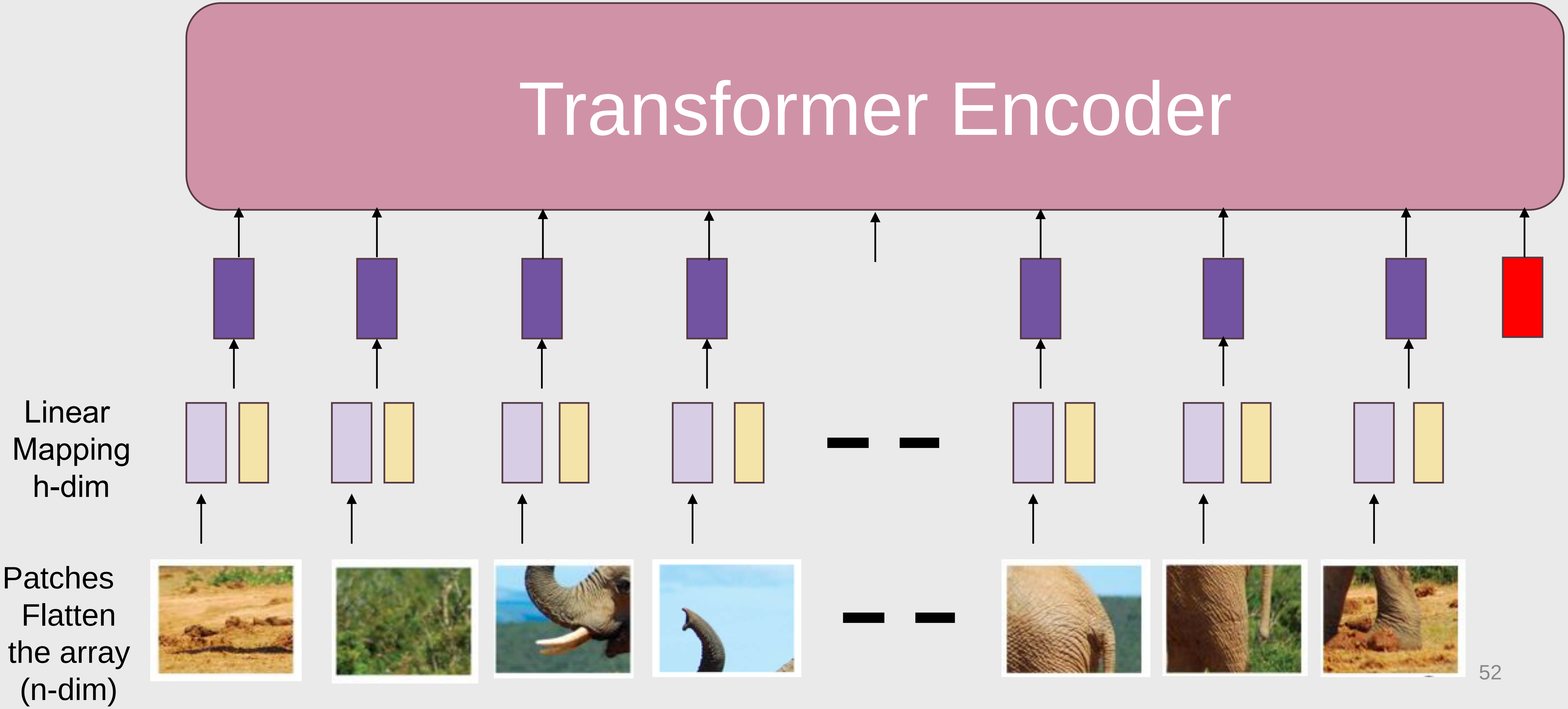
Positional Encoding and Vectors

Inform the model where the patch's position in the image is. In other word use sine and cosine values for respective patch number

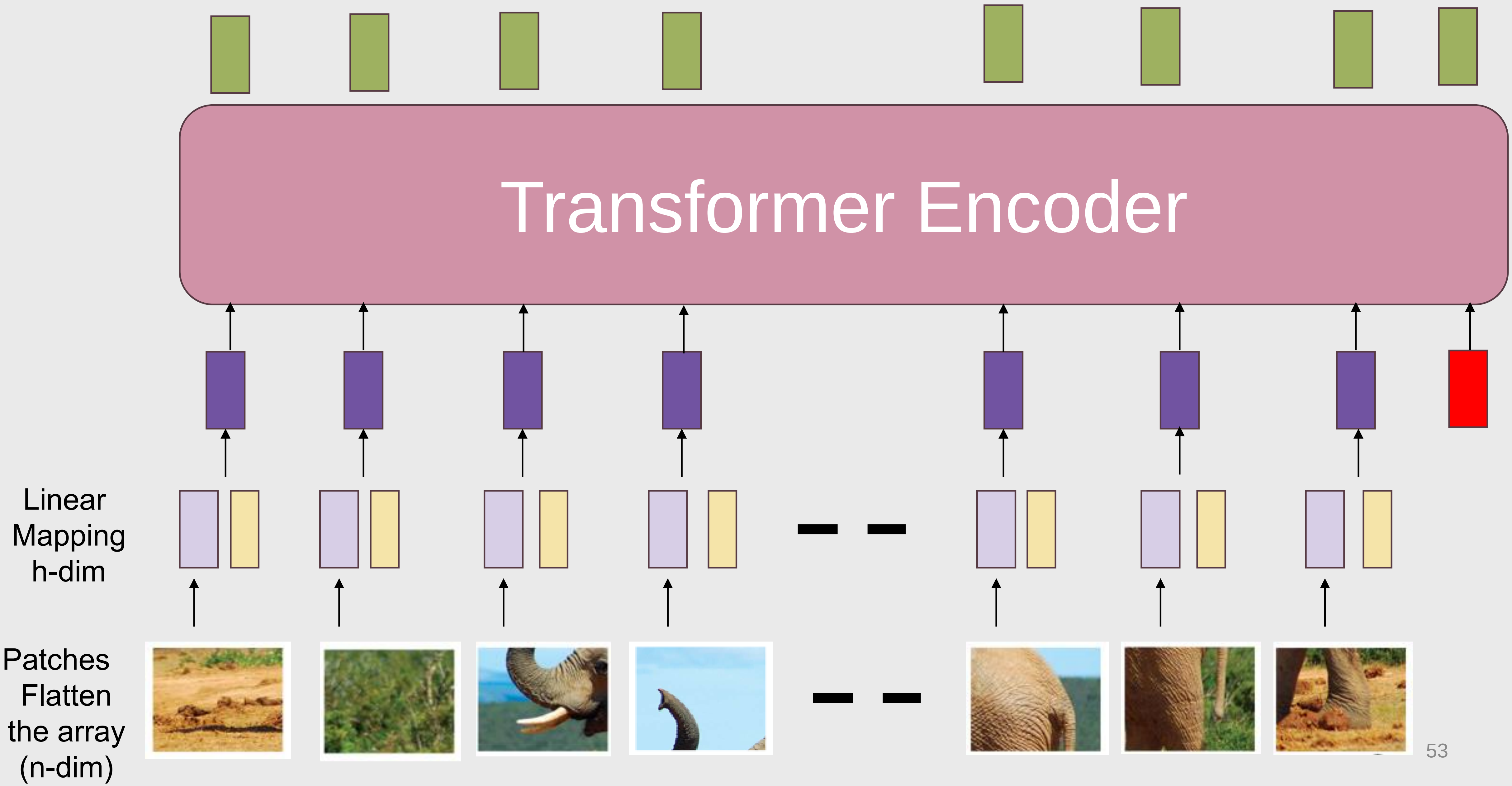
$$p_{i,j} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{\frac{j-1}{d_{emb\_dim}}}}\right) & \text{if } j \text{ is odd} \end{cases}$$



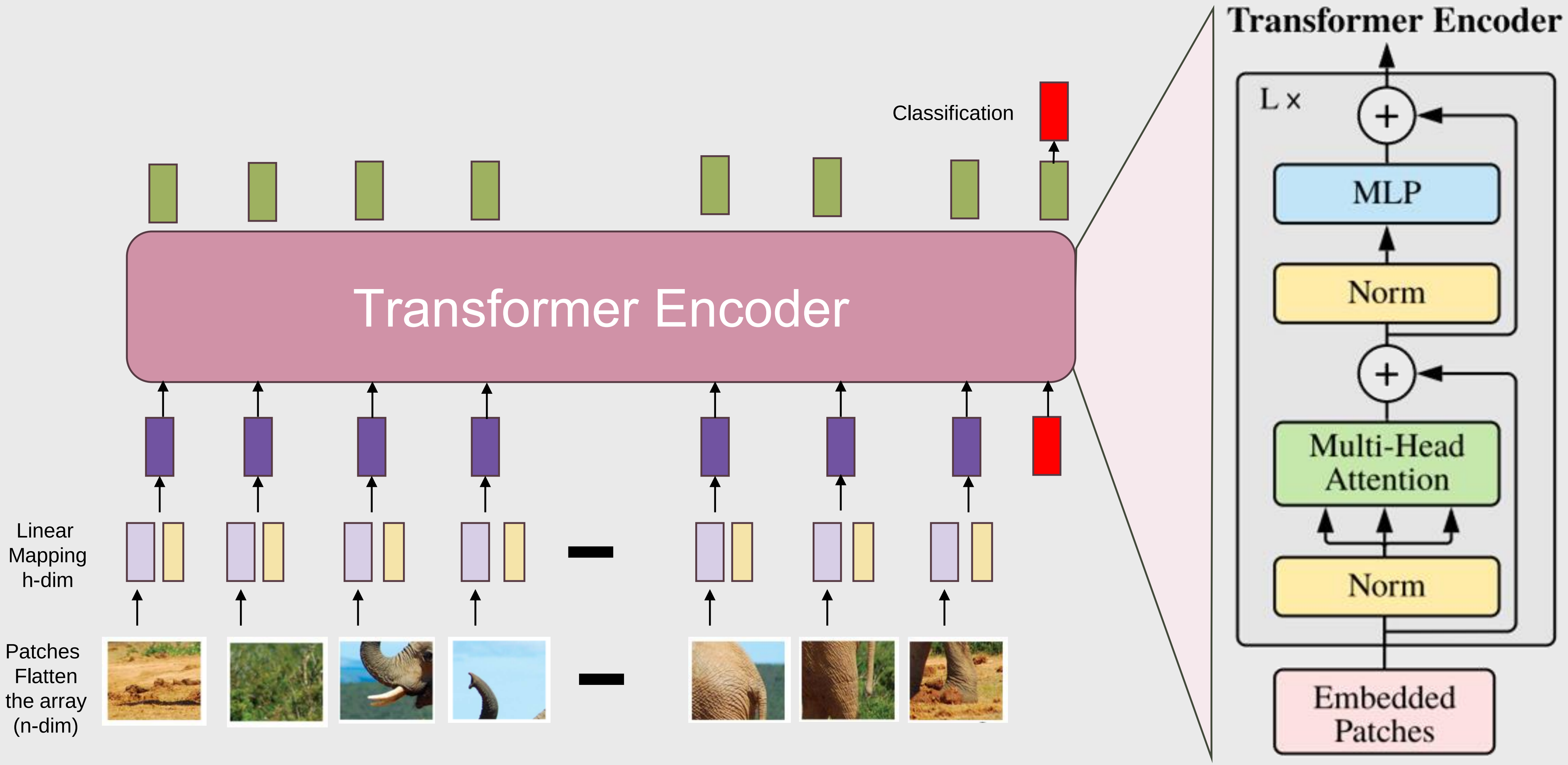
Add a Learnable Classification Token



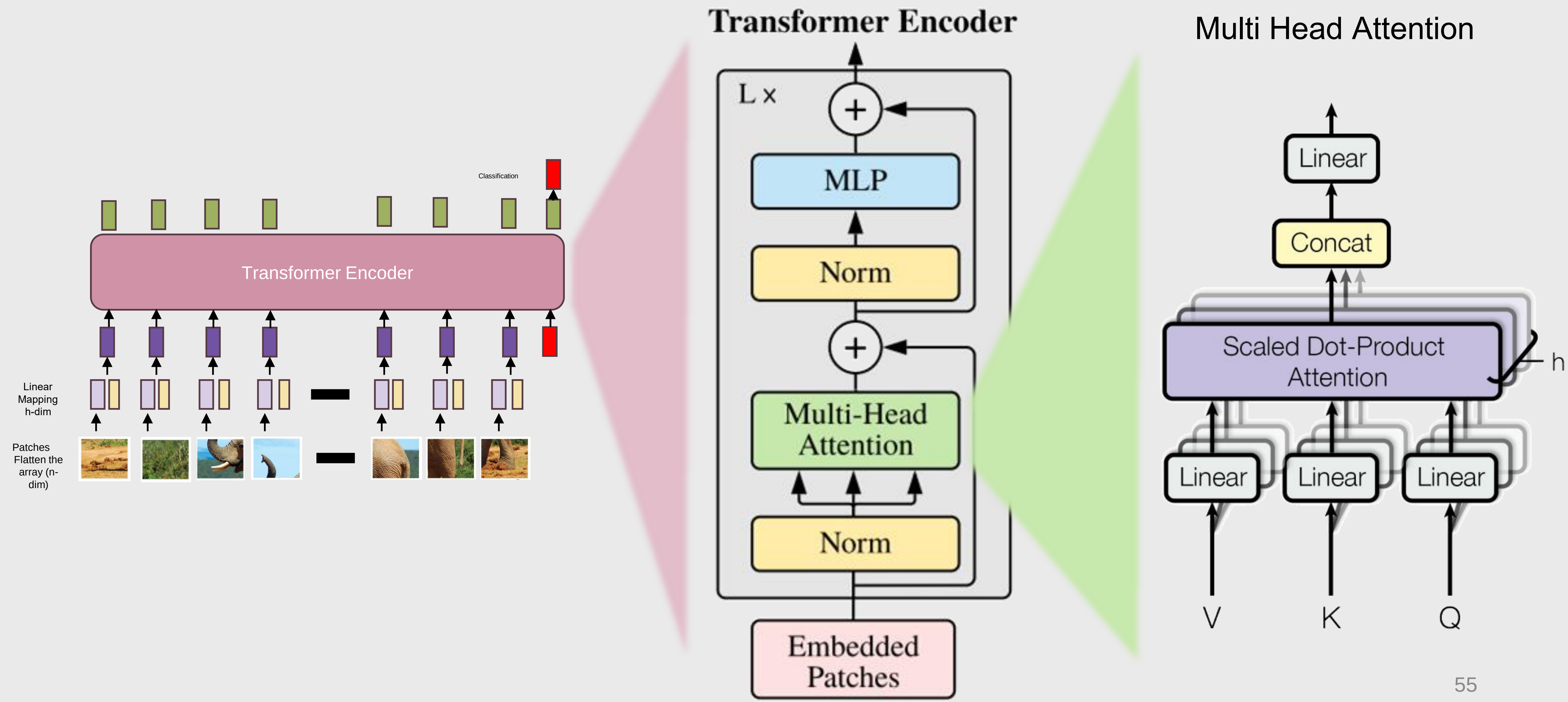
Output vector of Transformer Encoder



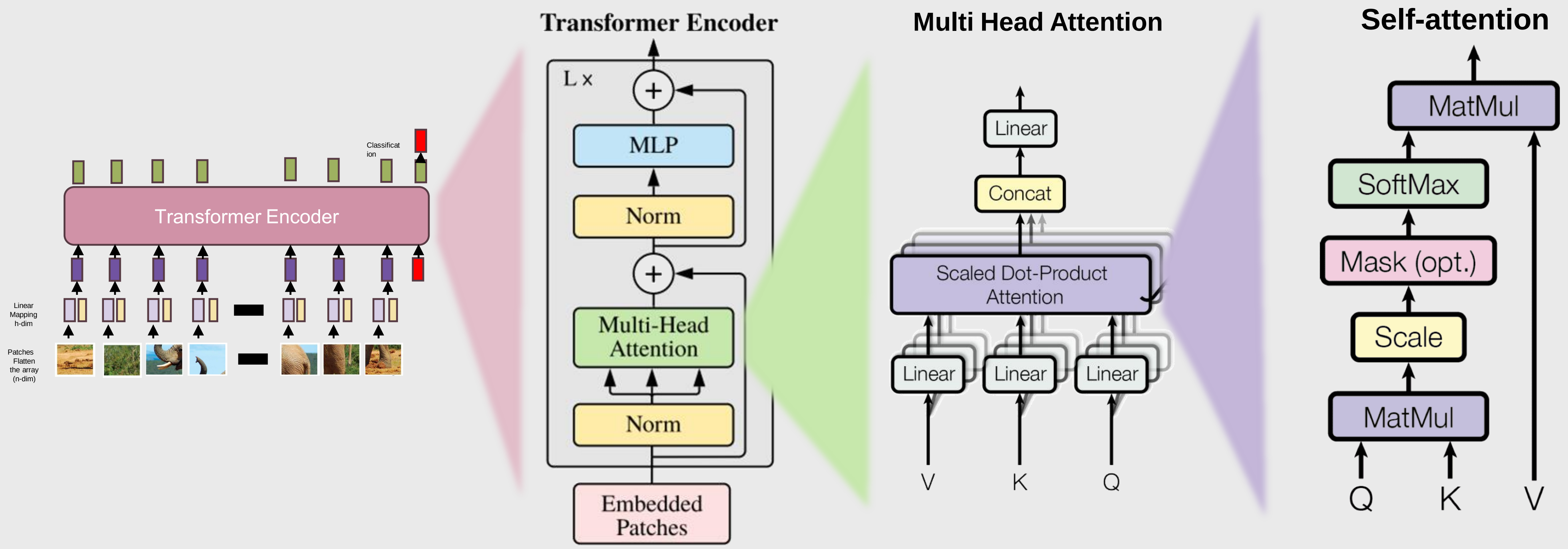
Same as ChatGPT Transformer



Same as ChatGPT Transformer



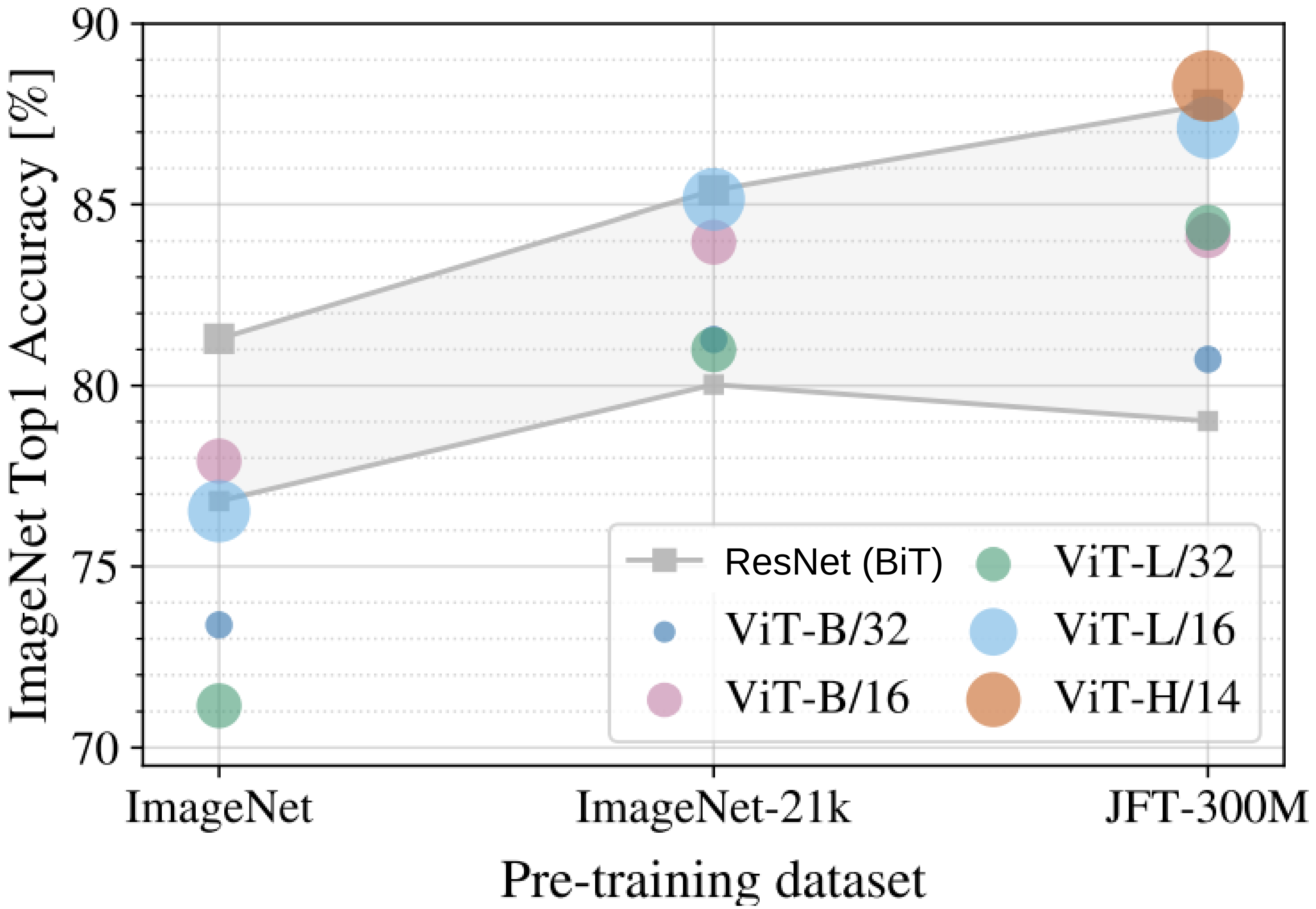
Same as ChatGPT Transformer



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

ViT Performance

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2017

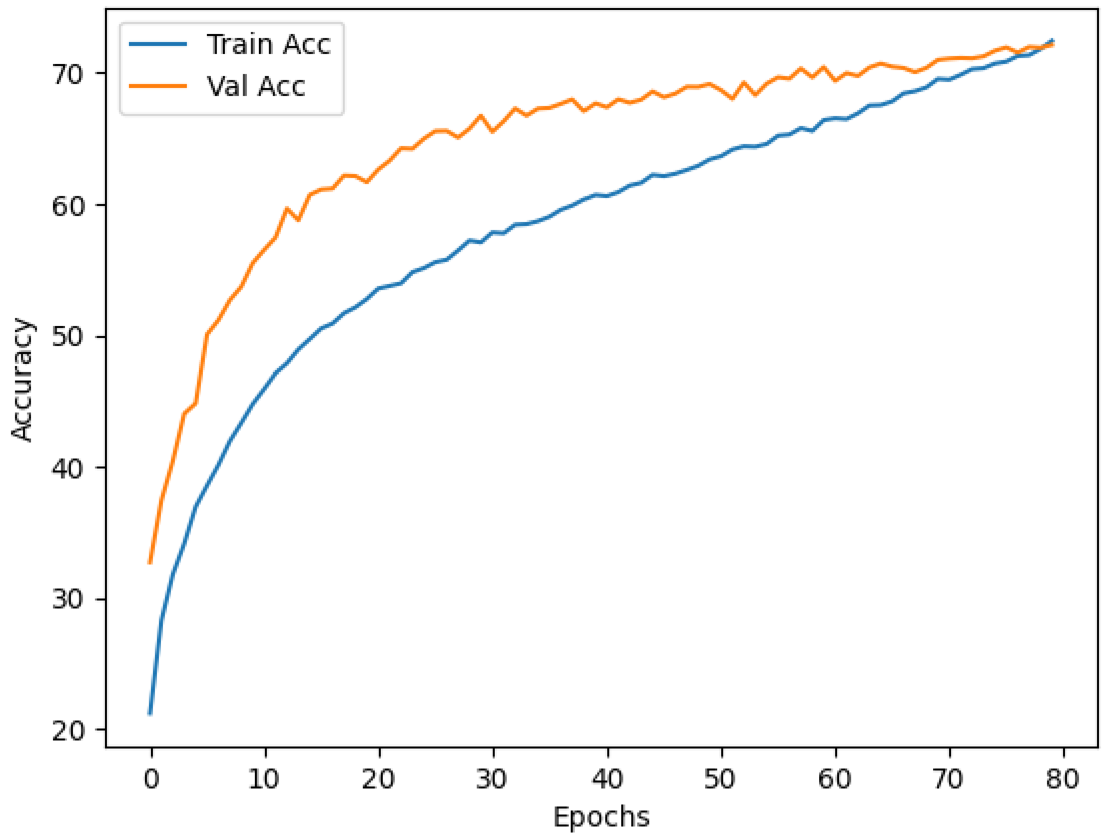


Note that this performance is only achieved when ViT is pre-trained on large dataset (in this case JFT-300M is a 300 million image dataset of Google)

Check Latest Models Here: https://paperswithcode.com/sota/image-classification-on-cifar-10

ViT on CIFAR-10 (without Pre-Training)

Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2017



Note that the performance depends on hyper parameter tuning models' size etc.

Source: https://github.com/ShivamRajSharma/Vision-Transformer?tab=readme-ov-file

ViT Performance

- Worse than ResNet when trained just on ImageNet
- Performance improved when pre-trained on very large dataset
- Pretrained outperforms much bigger CNNs
- You need large GPUs (Computational Cost is very high)

Coursework Brief (Part III)

More details to be released this week (before Practical Session)

Implement Convolutional Neural Networks (specifically using VGG16) on CIFAR-10 dataset and solve following three problems:

- For the training use early stopping and save the model that produce best validation results. (you will need to use some of training data as validation set) [*Marks 10*: 5+3+2]

- What would be the performance of VGG16 with or without batch normalization to it. Show using a convergence graph [*Marks 10*: 5+5]

- Visualise the Convolutional Features / Filters. This could be done by using `imshow` or similar methods. Show how filters features changes over different layers over a test image. [*Marks 20*: 10+10]