

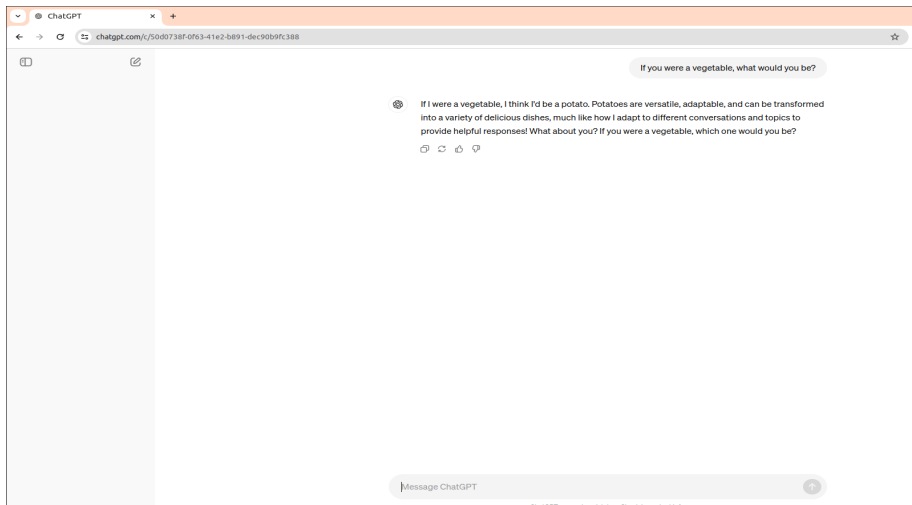
Transformer

Ranjeet Ranjan Jha

Mathematics Department
Indian Institute of Technology Patna

July 25, 2024

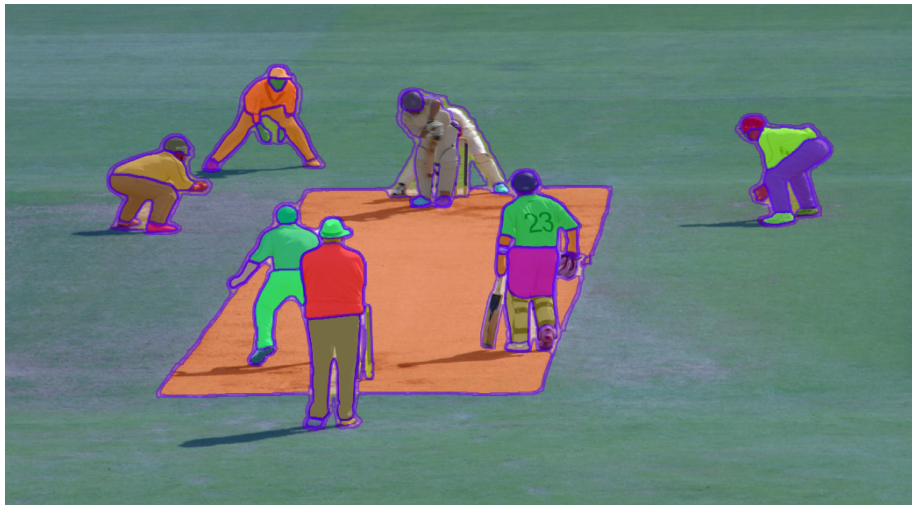
Transformer Applications: ChatGPT



Transformer Applications: Segment Anything



Transformer Applications: Segment Anything



Transformer

- Used for modeling long dependencies between input sequence elements
- Supports parallel processing of sequence as compared to RNN (e.g. LSTM)
- Allows processing multiple modalities
 - (e.g., images, videos, text and speech) using similar processing blocks
- Typically, pre-trained using pretext tasks on largescale (unlabeled) datasets
- Demonstrates excellent scalability to very large networks and huge datasets.
- **GPT-4** (Generative Pretrained Transformer)

Vision Applications

- Recognition tasks (e.g., image classification, object detection, action recognition, and segmentation),
- Generative modeling, multi-modal tasks (e.g., visual-question answering, visual reasoning, and visual grounding),
- Video processing (e.g., activity recognition, video forecasting),
- Low-level vision (e.g., image super-resolution, image enhancement, and colorization)
- 3D analysis (e.g., point cloud classification and segmentation)

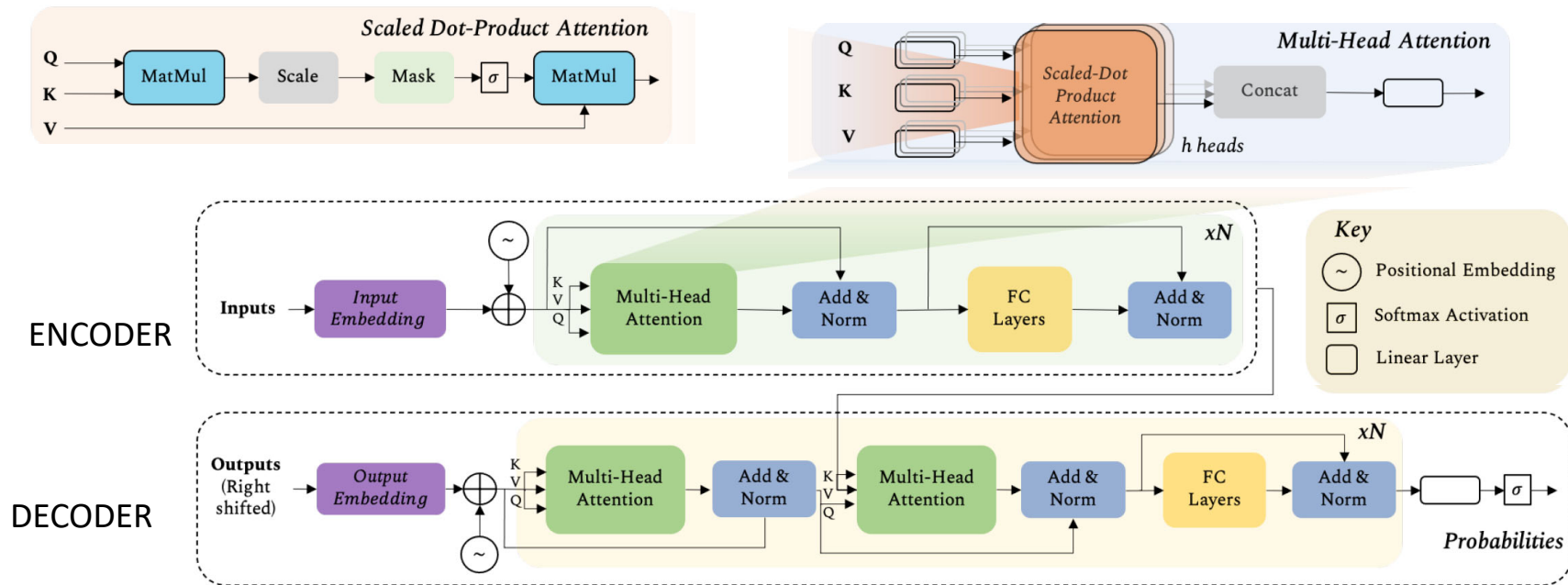
Natural Language Processing

- BERT (Bidirectional Encoder Representations from Transformers),
- GPT (Generative Pre-trained **Transformer**) v1-4,
- RoBERTa (Robustly Optimized BERT Pre-training)
- T5 (Text-to-Text Transfer Transformer)

Transformer Basics

- It consists of Encoder and Decoder Blocks
- Main components of each block:
 - Self-Attention
 - Layer Normalization
 - Feed Forward Network

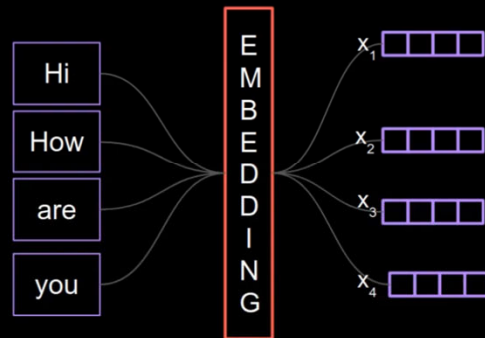
Transformer



Slide courtesy of AI Bites, Youtube Channel:
<https://www.youtube.com/c/AIBites>

1/11/2024

Self-Attention

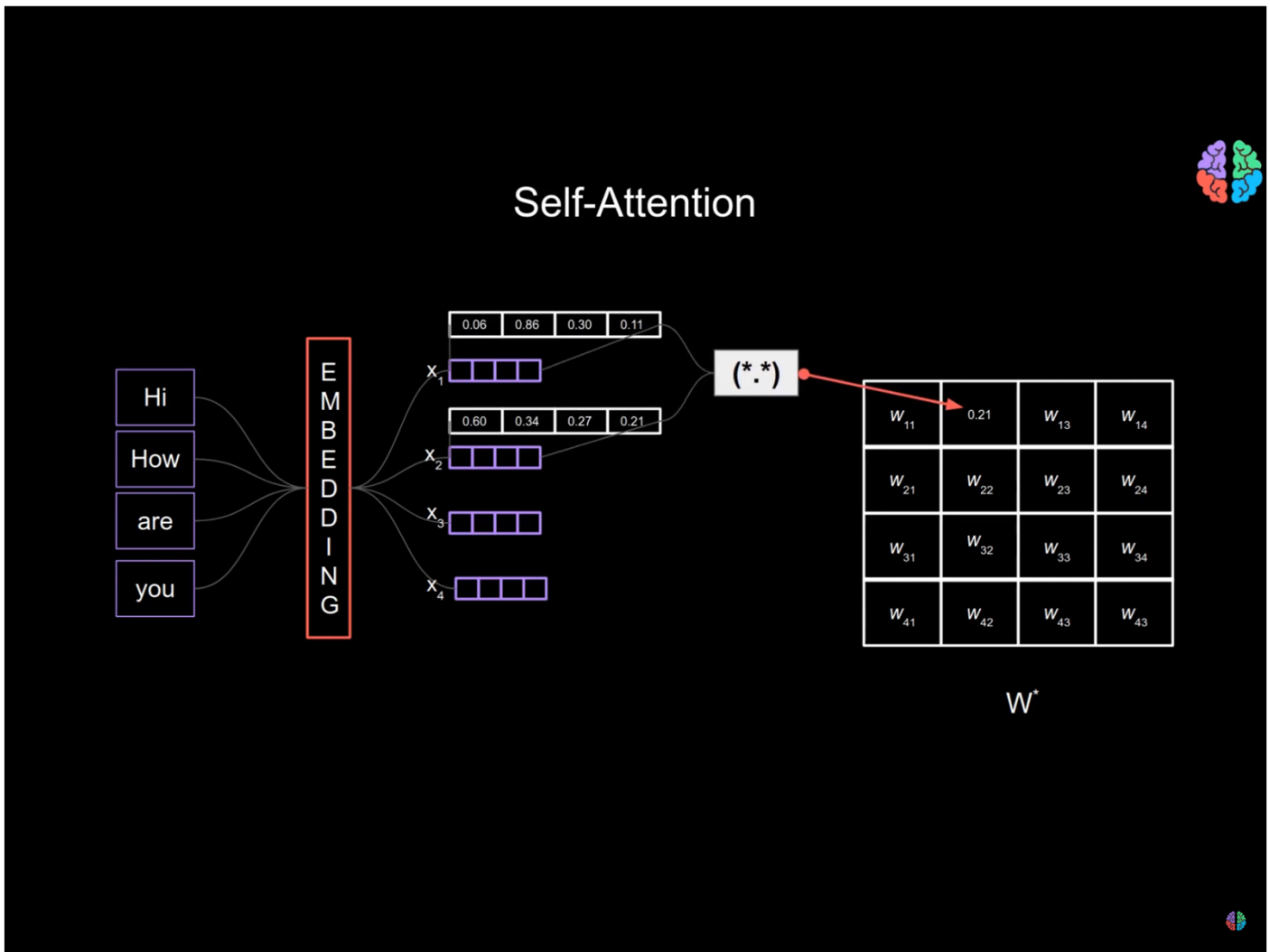


w_{11}	0.21	w_{13}	w_{14}
w_{21}	w_{22}	w_{23}	w_{24}
w_{31}	w_{32}	w_{33}	w_{34}
w_{41}	w_{42}	w_{43}	w_{43}

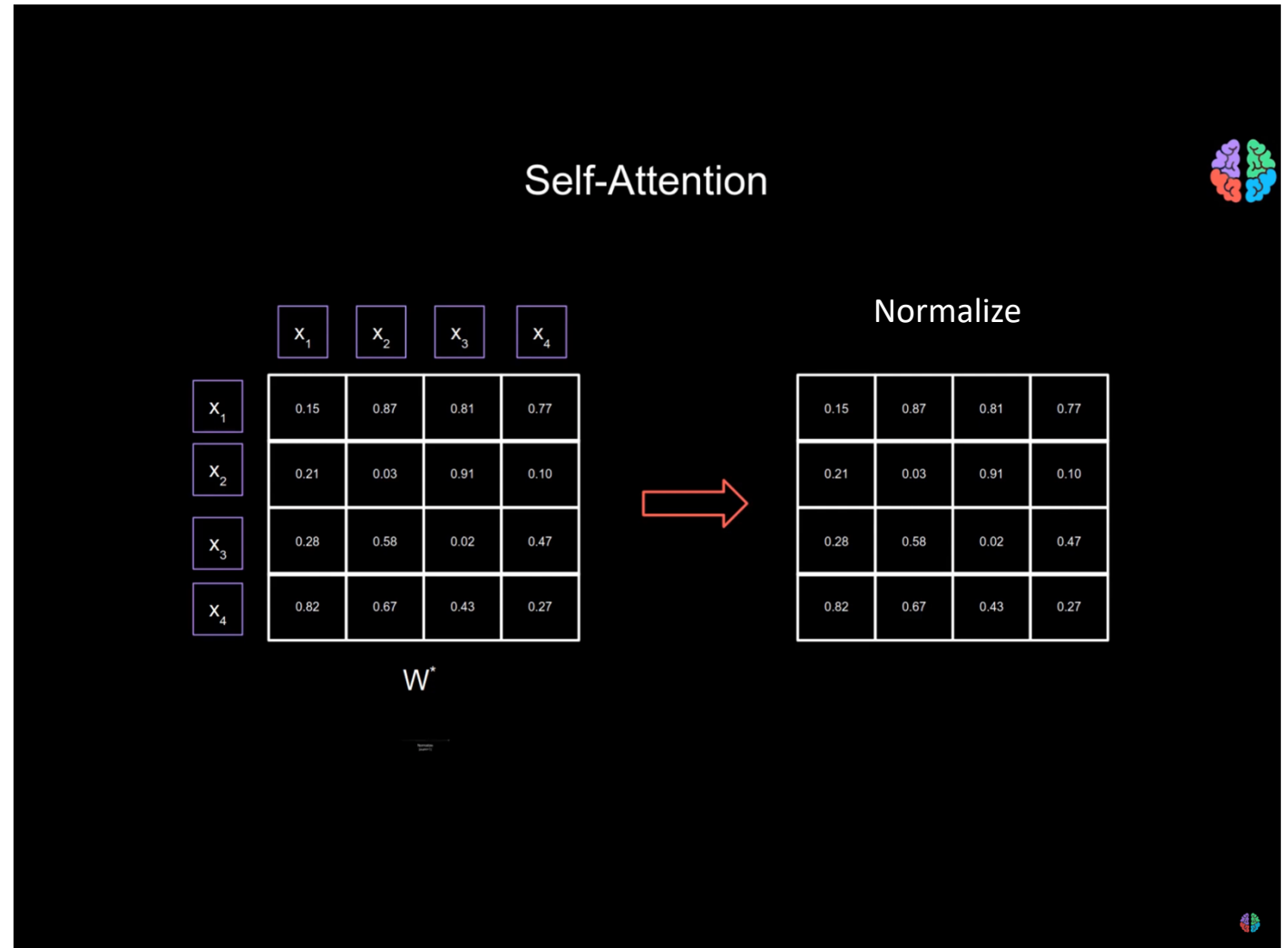
W^*



Slide courtesy of AI Bites, Youtube
Channel:
<https://www.youtube.com/c/AIBites>



Slide courtesy of AI Bites, Youtube
Channel:
<https://www.youtube.com/c/AIBites>



Slide courtesy of AI Bites, Youtube
 Channel:
<https://www.youtube.com/c/AIBites>

Self-Attention



0.15	0.87	0.81	0.77
0.21	0.03	0.91	0.10
0.28	0.58	0.02	0.47
0.82	0.67	0.43	0.27

W

*











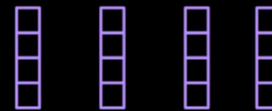
Slide courtesy of AI Bites, Youtube
 Channel:
<https://www.youtube.com/c/AIBites>

Self-Attention

0.15	0.87	0.81	0.77
0.21	0.03	0.91	0.10
0.28	0.58	0.02	0.47
0.82	0.67	0.43	0.27

W

*



X^T



Y

Slide courtesy of AI Bites, Youtube
 Channel:
<https://www.youtube.com/c/AIBites>

Self-Attention



0.15	0.87	0.81	0.77
0.21	0.03	0.91	0.10
0.28	0.58	0.02	0.47
0.82	0.67	0.43	0.27

W



X^T

Y

$$Y = W X^T$$

Self-Attention

- So far no learning!

Self-Attention of Image Features

Self-Attention

0.15	0.87	0.81	0.77
0.21	0.03	0.91	0.10
0.28	0.58	0.02	0.47
0.82	0.67	0.43	0.27

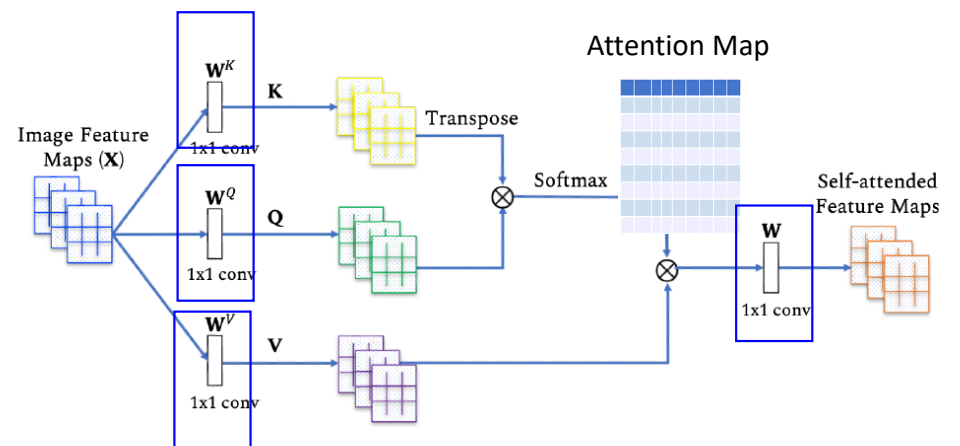
W

*

X^T

Y

$$Y = W X^T$$



Self-Attention (Matrix Explanation)

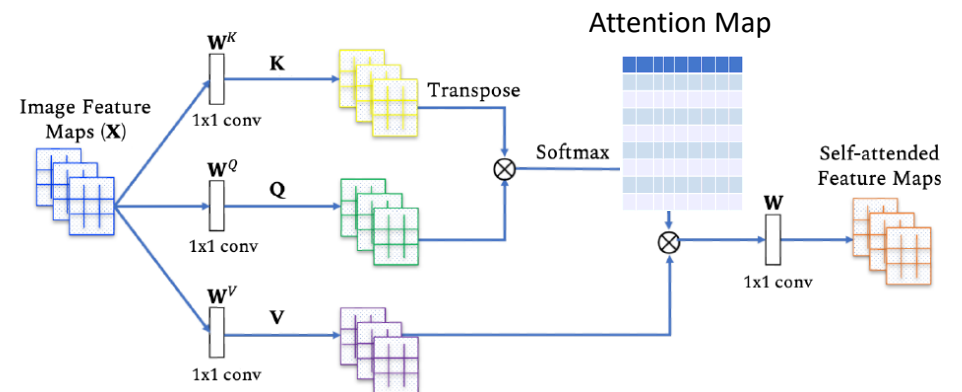
Lets denote a sequence of n entities ($\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$)

by $\mathbf{X} \in \mathbb{R}^{n \times d}$,

Queries ($\mathbf{W}^Q \in \mathbb{R}^{d \times d_q}$), Keys ($\mathbf{W}^K \in \mathbb{R}^{d \times d_k}$) and Values ($\mathbf{W}^V \in \mathbb{R}^{d \times d_v}$), where $d_q = d_k$. The input sequence $\mathbf{X}$ is

first projected onto these weight matrices to get $\hat{\mathbf{Q}} = \mathbf{X}\mathbf{W}^Q$, $\hat{\mathbf{K}} = \mathbf{X}\mathbf{W}^K$ and $\hat{\mathbf{V}} = \mathbf{X}\mathbf{W}^V$. The output $\mathbf{Z} \in \mathbb{R}^{n \times d_v}$ of the

$$\mathbf{Z} = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_q}} \right) \mathbf{V}.$$



Self-Attention

Lets denote a sequence of n entities $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$

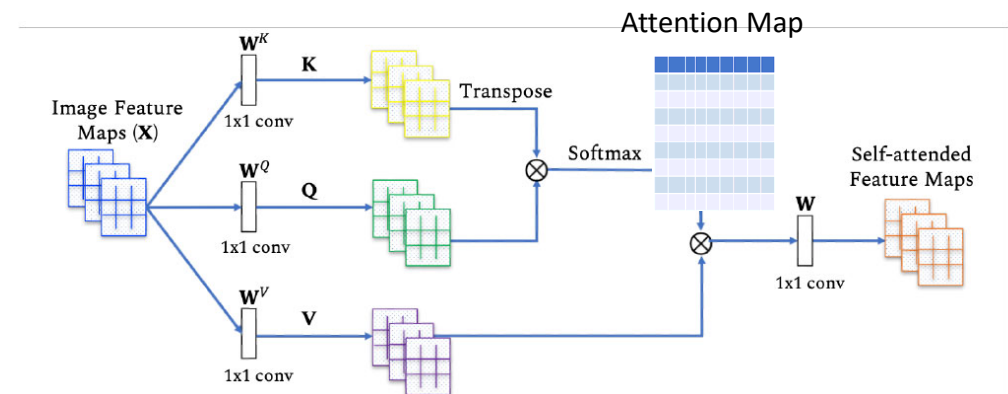
by $\mathbf{X} \in \mathbb{R}^{n \times d}$
 9×3

Queries ($\mathbf{W}^Q \in \mathbb{R}^{d \times d_q}$), Keys ($\mathbf{W}^K \in \mathbb{R}^{d \times d_k}$) and Values ($\mathbf{W}^V \in \mathbb{R}^{d \times d_v}$), where $d_q = d_k$. The input sequence $\mathbf{X}$ is
 3×3

first projected onto these weight matrices to get $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$,
 $\mathbf{K} = \mathbf{X}\mathbf{W}^K$ and $\mathbf{V} = \mathbf{X}\mathbf{W}^V$. The output $\mathbf{Z} \in \mathbb{R}^{n \times d_v}$ of the
 9×3

$$\mathbf{Z} = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_q}} \right) \mathbf{V}.$$

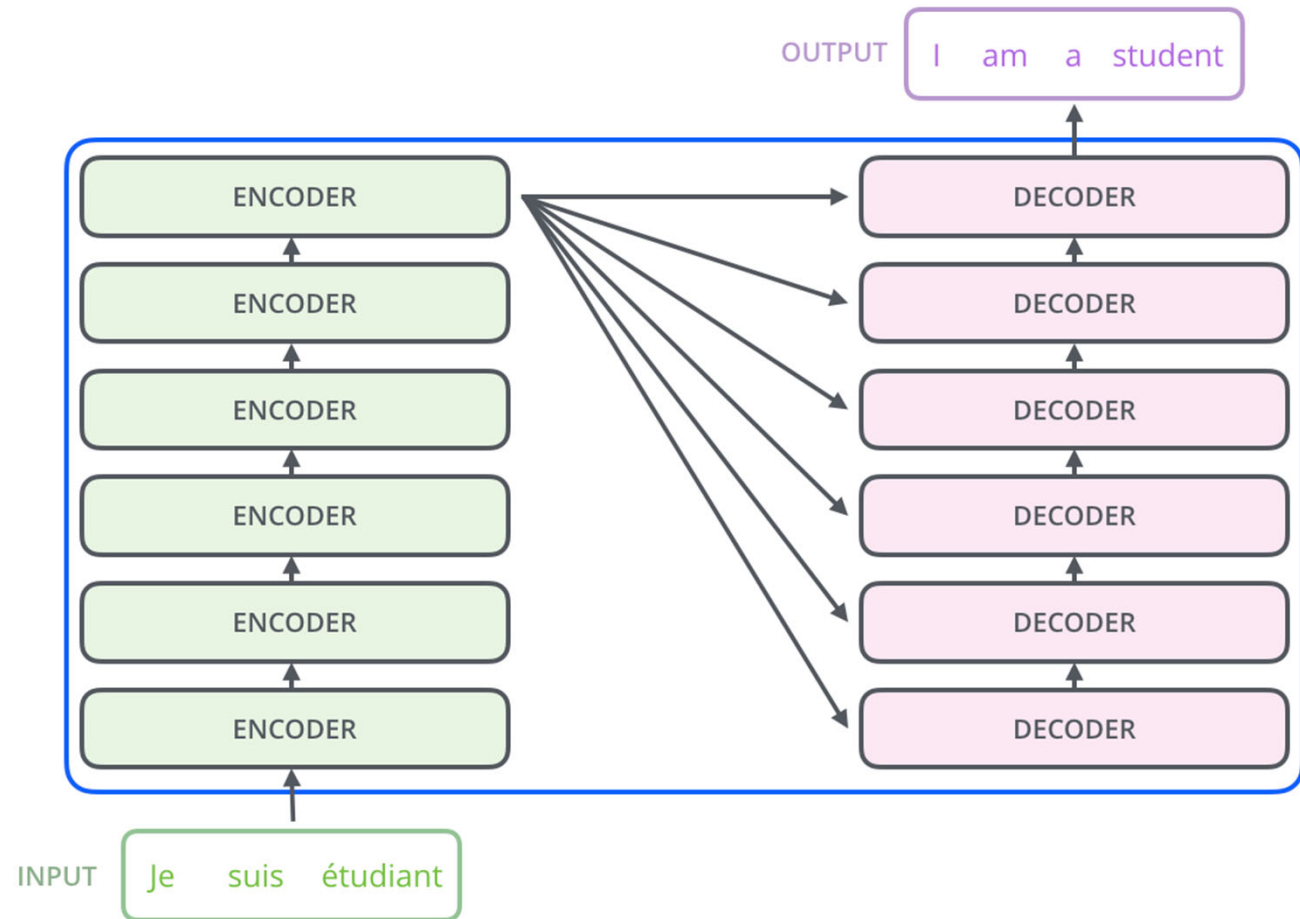
9×3



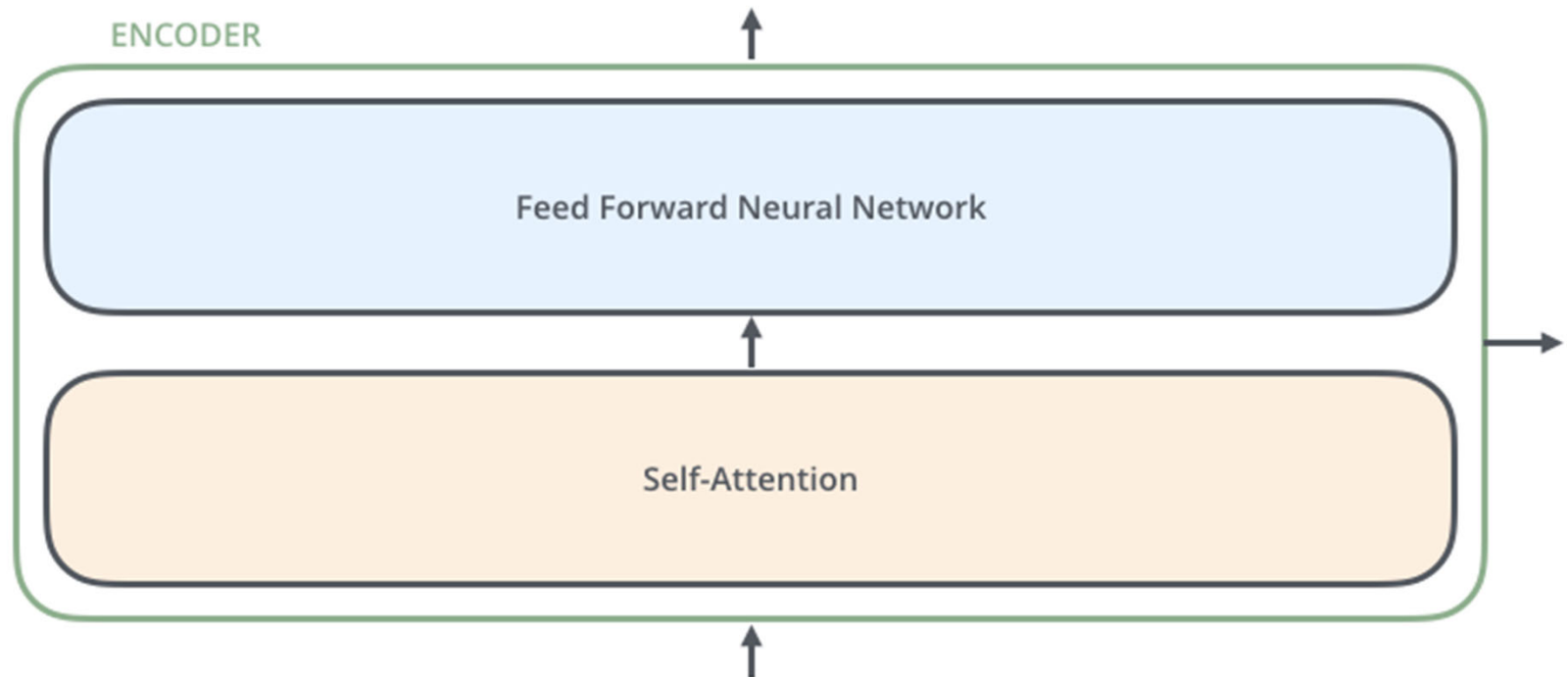
Transformers (Attention is all you need 2017)

- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in NeurIPS, 2017. (102,618 citations)
- Two valuable sources
 - <http://nlp.seas.harvard.edu/2018/04/03/attention.html>
 - <https://jalammar.github.io/illustrated-transformer/> (slides come from this source)
- Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

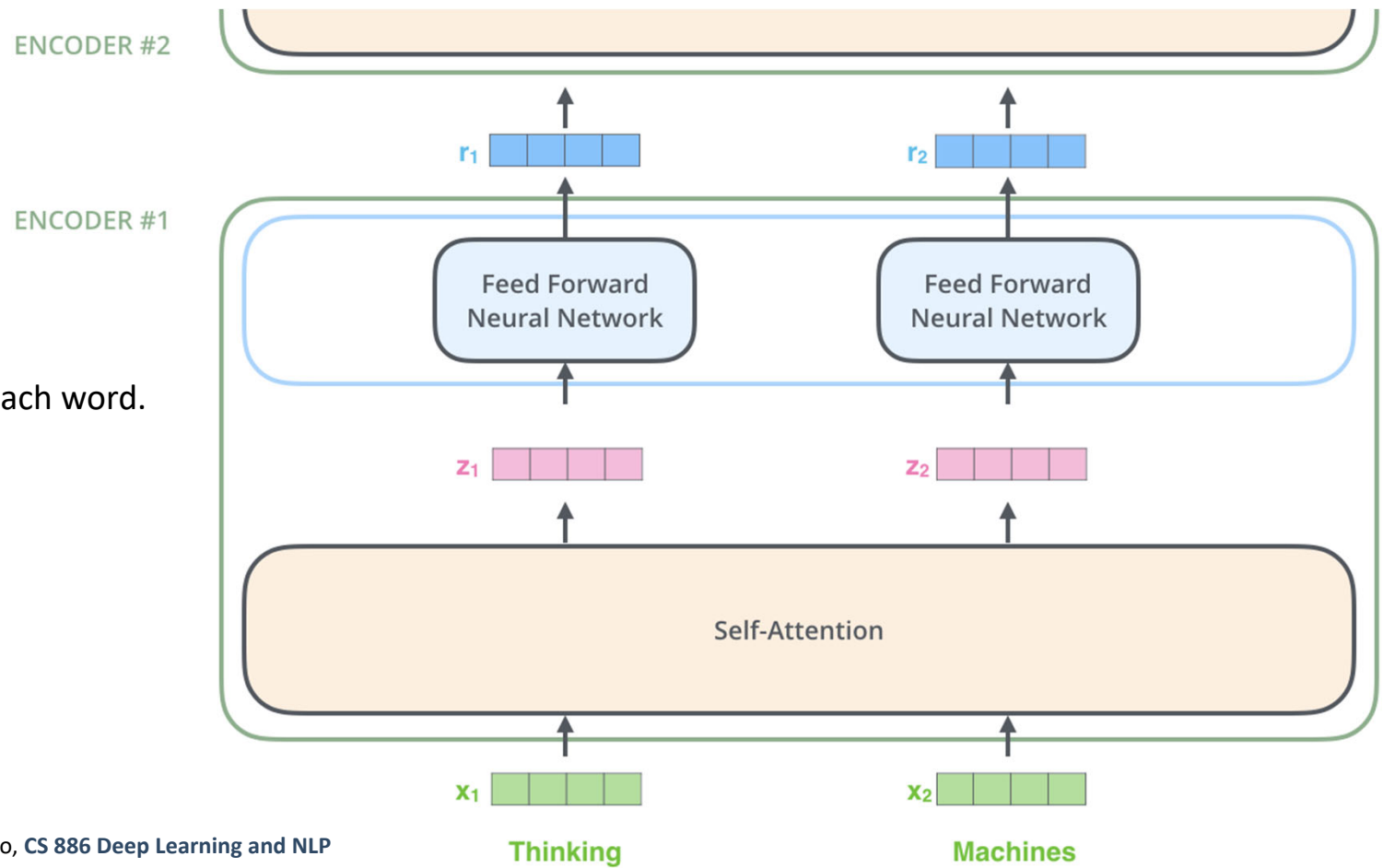
Transformer



An Encoder Block: same structure, different parameters



Encoder

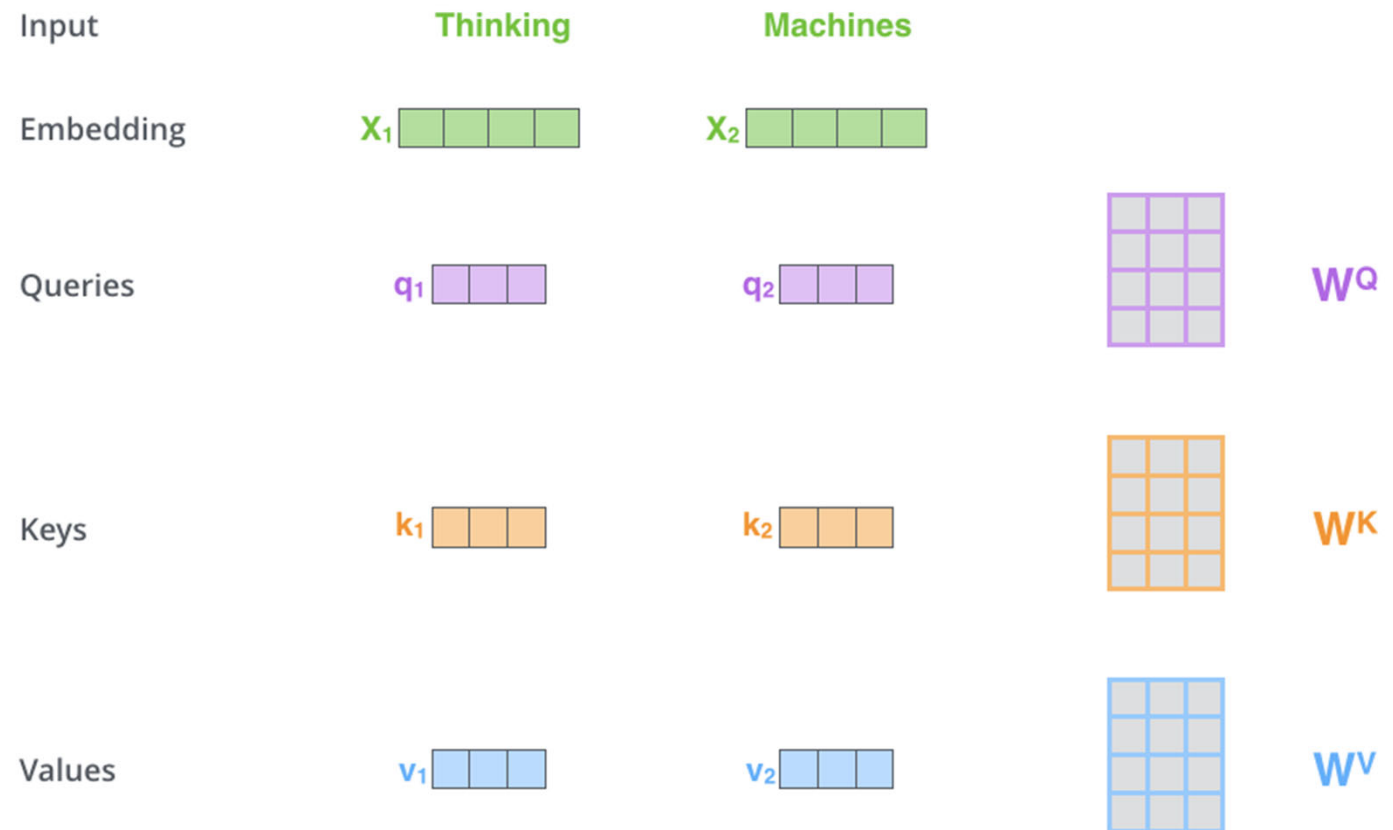


The ffnn is independent for each word.
Hence can be parallelized.

Self Attention

• First, we create three vectors by multiplying input embedding x_i with three matrices

- $q_i = x_i W^Q$
- $K_i = x_i W^K$
- $V_i = x_i W^V$



Self Attention

Now we calculate
a score to determine how
much focus to place on other
Parts of the input.

Input

Embedding

Queries

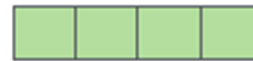
Keys

Values

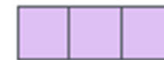
Score

Thinking

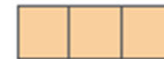
x_1



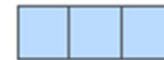
q_1



k_1



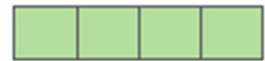
v_1



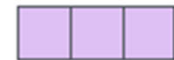
$$q_1 \cdot k_1 = 112$$

Machines

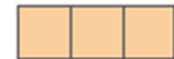
x_2



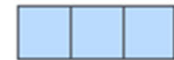
q_2



k_2



v_2



$$q_1 \cdot k_2 = 96$$



Attention and Transformer

Self Attention

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Softmax
X
Value
Value

Sum

Thinking

x_1

q_1

k_1

v_1

$q_1 \cdot k_1 = 112$

14

0.88

$\tilde{v}_1$

z_1

Machines

x_2

q_2

k_2

v_2

$q_2 \cdot k_2 = 96$

12

0.12

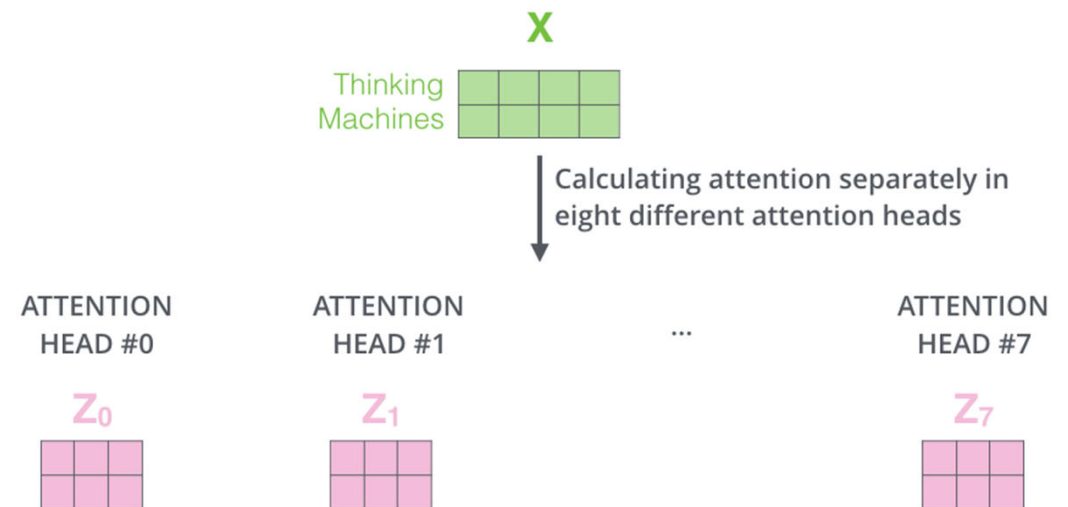
$\tilde{v}_2$

z_2

$$z_1 = 0.88v_1 + 0.12v_2$$

Multiple heads

1. It expands the model's ability to focus on different positions.
2. It gives the attention layer multiple “representation subspaces”



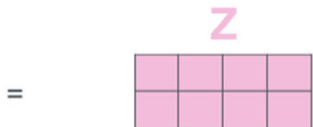


Attention and Transformers

1) Concatenate all the attention heads

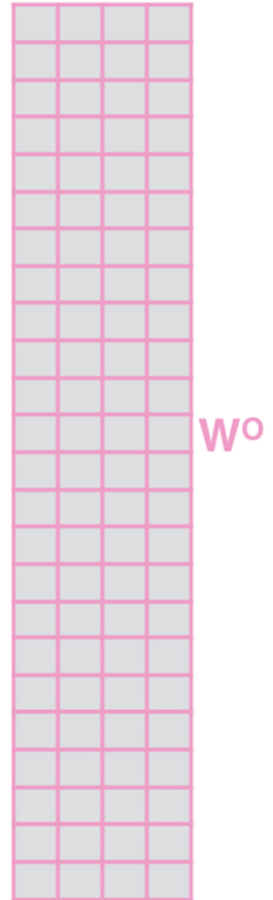


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



2) Multiply with a weight matrix W^O that was trained jointly with the model

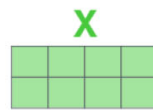
$\times$



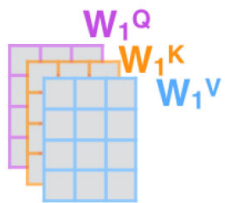
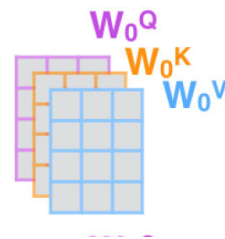
1) This is our input sentence*

Thinking
Machines

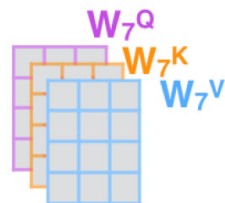
2) We embed each word*



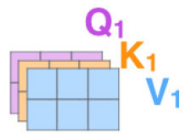
3) Split into 8 heads.
We multiply X or R with weight matrices



...



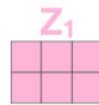
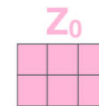
4) Calculate attention using the resulting $Q/K/V$ matrices



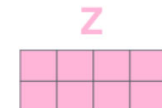
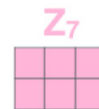
...

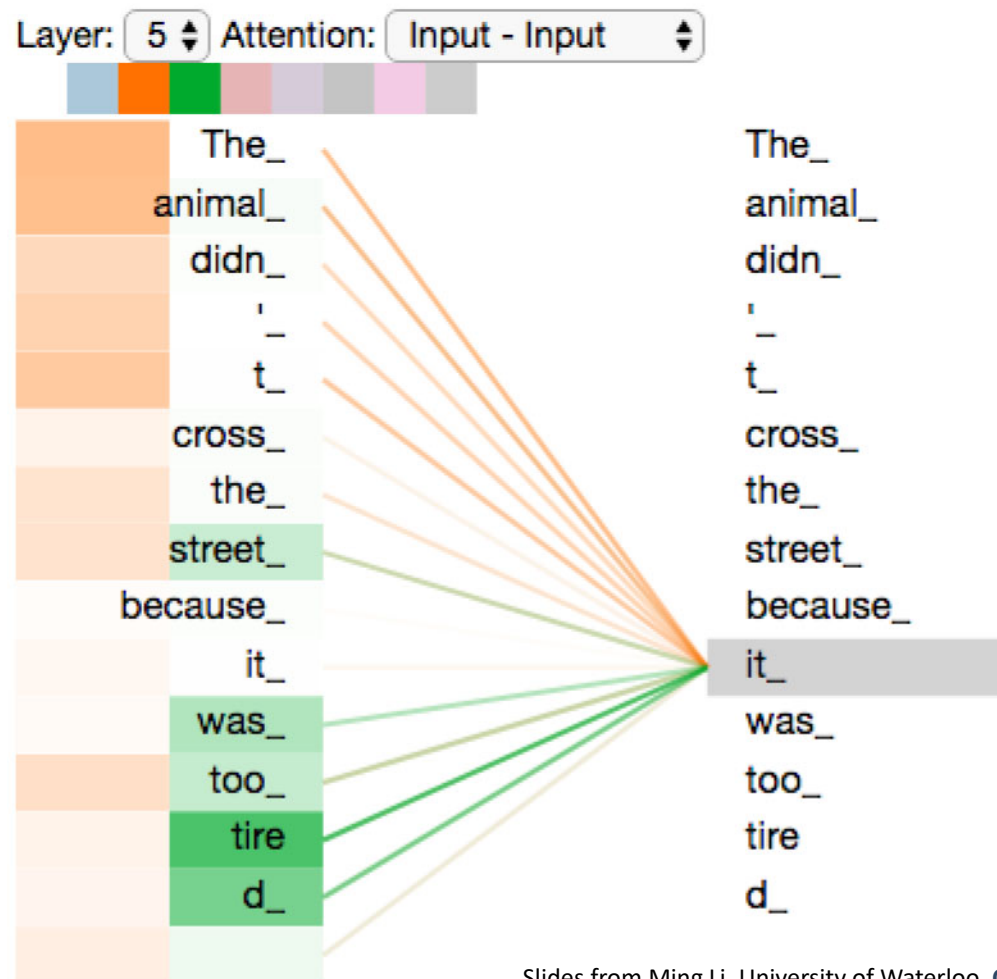


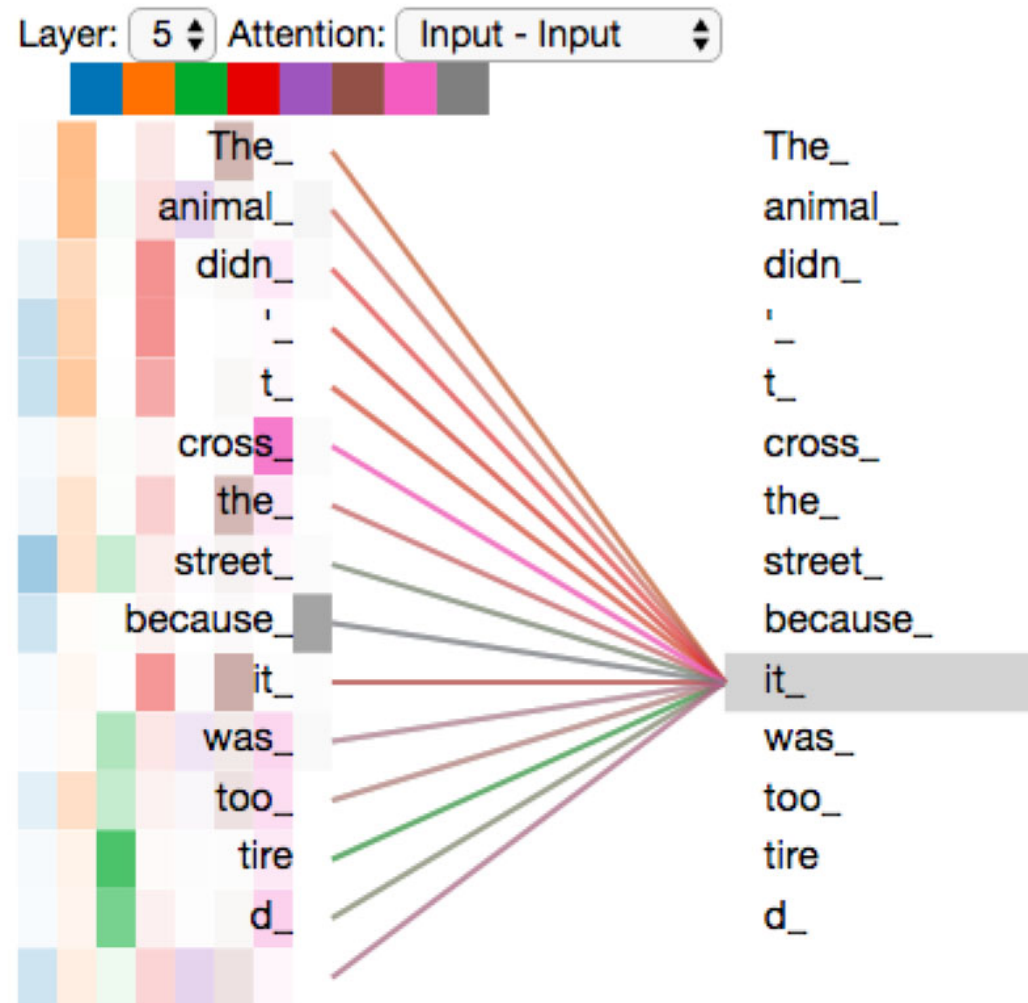
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



...







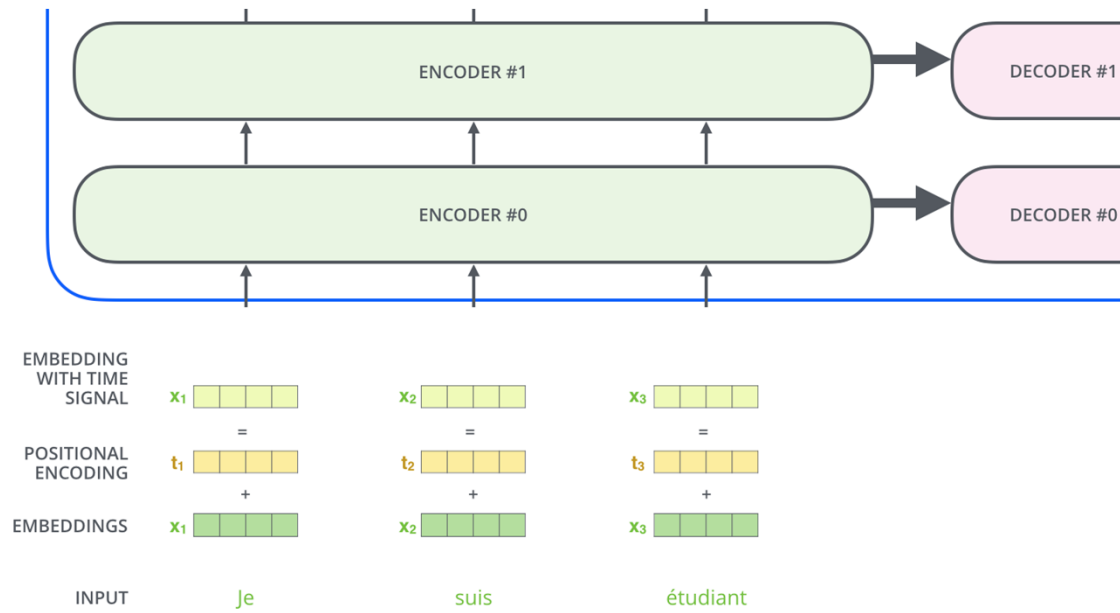


Attention and Transformers

Representing the input order (positional encoding)

- Transformer is permutation invariant
- The transformer adds a vector to each input embedding.
- These vectors follow a specific pattern that the model learns
- Learned pattern helps model
 - to determine the position of each word, or
 - the distance between different words.

Representing the input order (positional encoding)



Position Encoding

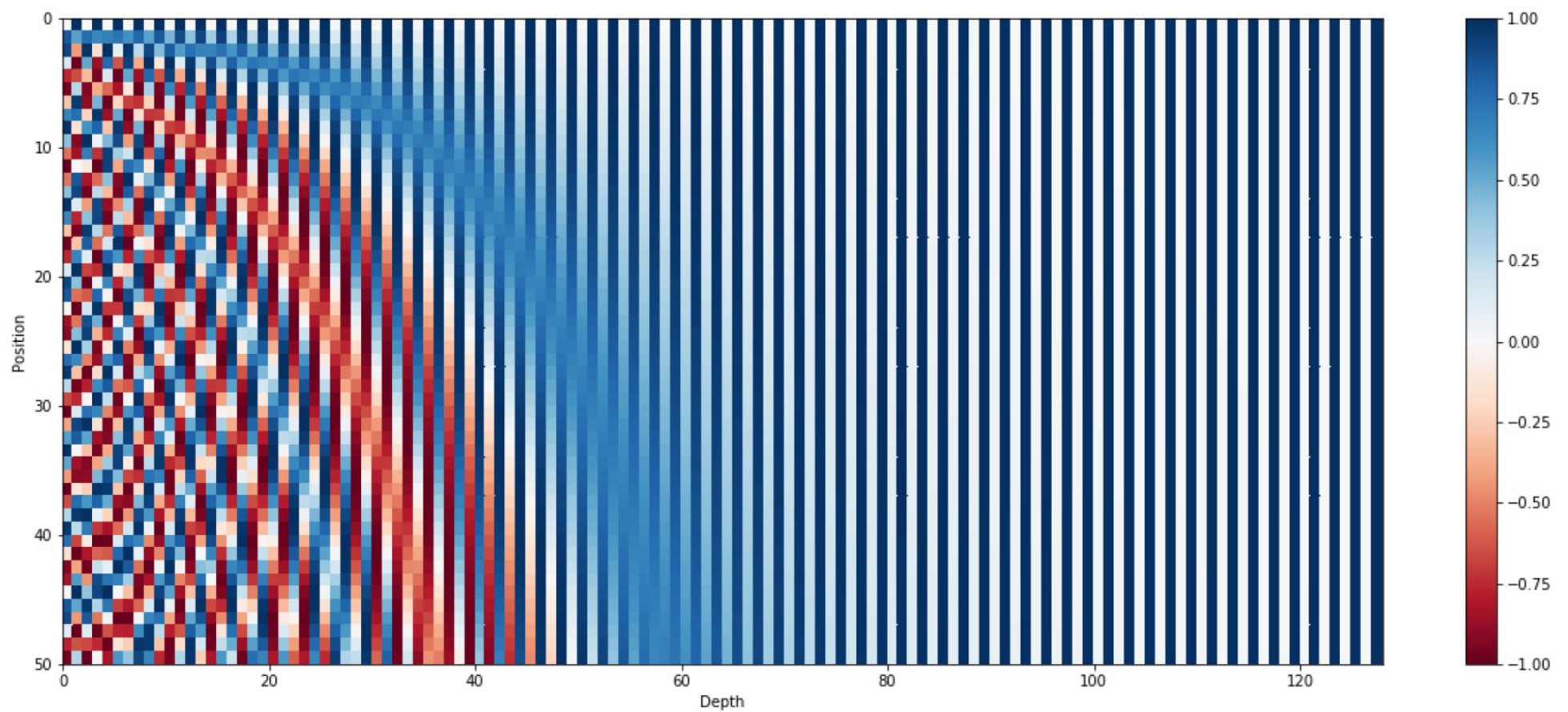
$$\vec{p}_t^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases}$$

where

$$\omega_k = \frac{1}{10000^{2k/d}}$$

$$\vec{p}_t = \begin{bmatrix} \sin(\omega_1 \cdot t) \\ \cos(\omega_1 \cdot t) \\ \\ \sin(\omega_2 \cdot t) \\ \cos(\omega_2 \cdot t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} \cdot t) \\ \cos(\omega_{d/2} \cdot t) \end{bmatrix}_{d \times 1}$$

Position Encoding



$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

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

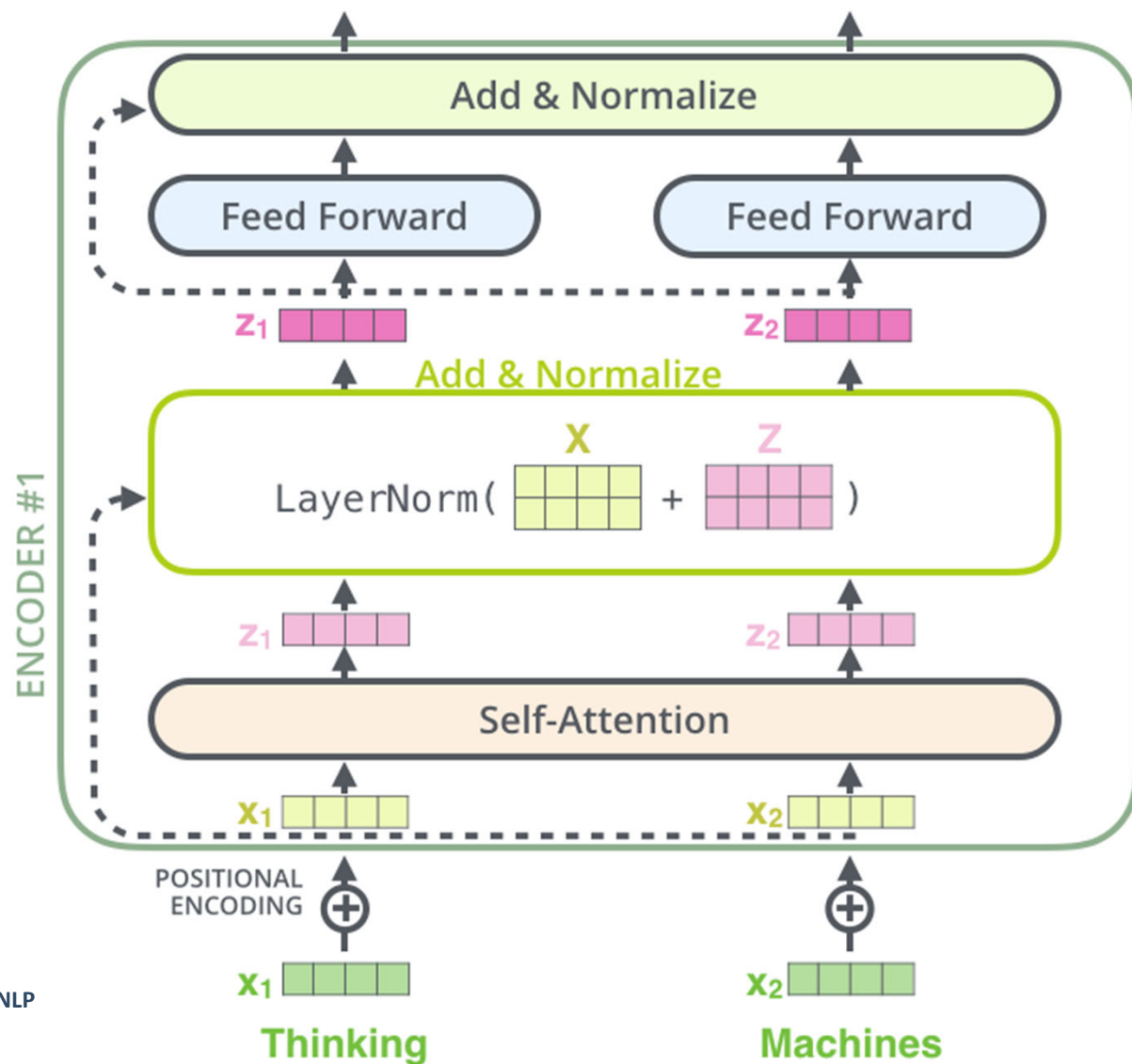
Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	$P_{00}=\sin(0)$ = 0	$P_{01}=\cos(0)$ = 1	$P_{02}=\sin(0)$ = 0	$P_{03}=\cos(0)$ = 1
am	1	$P_{10}=\sin(1/1)$ = 0.84	$P_{11}=\cos(1/1)$ = 0.54	$P_{12}=\sin(1/10)$ = 0.10	$P_{13}=\cos(1/10)$ = 1.0
a	2	$P_{20}=\sin(2/1)$ = 0.91	$P_{21}=\cos(2/1)$ = -0.42	$P_{22}=\sin(2/10)$ = 0.20	$P_{23}=\cos(2/10)$ = 0.98
Robot	3	$P_{30}=\sin(3/1)$ = 0.14	$P_{31}=\cos(3/1)$ = -0.99	$P_{32}=\sin(3/10)$ = 0.30	$P_{33}=\cos(3/10)$ = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

Position Encoding

- Can also be learned
- Learn like other parameters

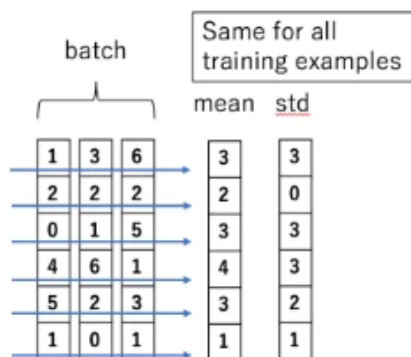
Add and Normalize



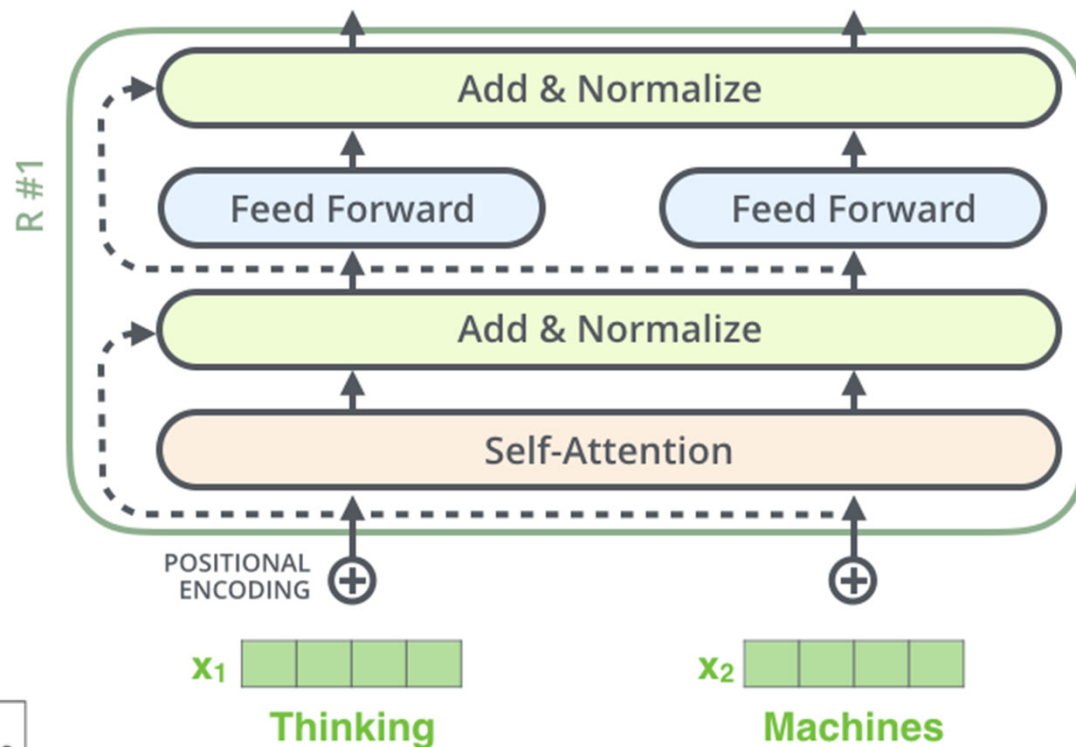
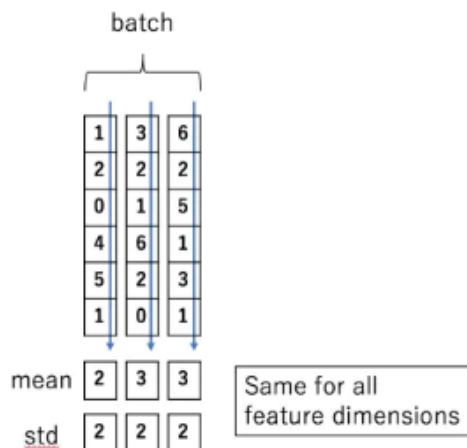
Layer Normalization (Hinton)

Layer normalization normalizes the inputs across the features.

Batch Normalization

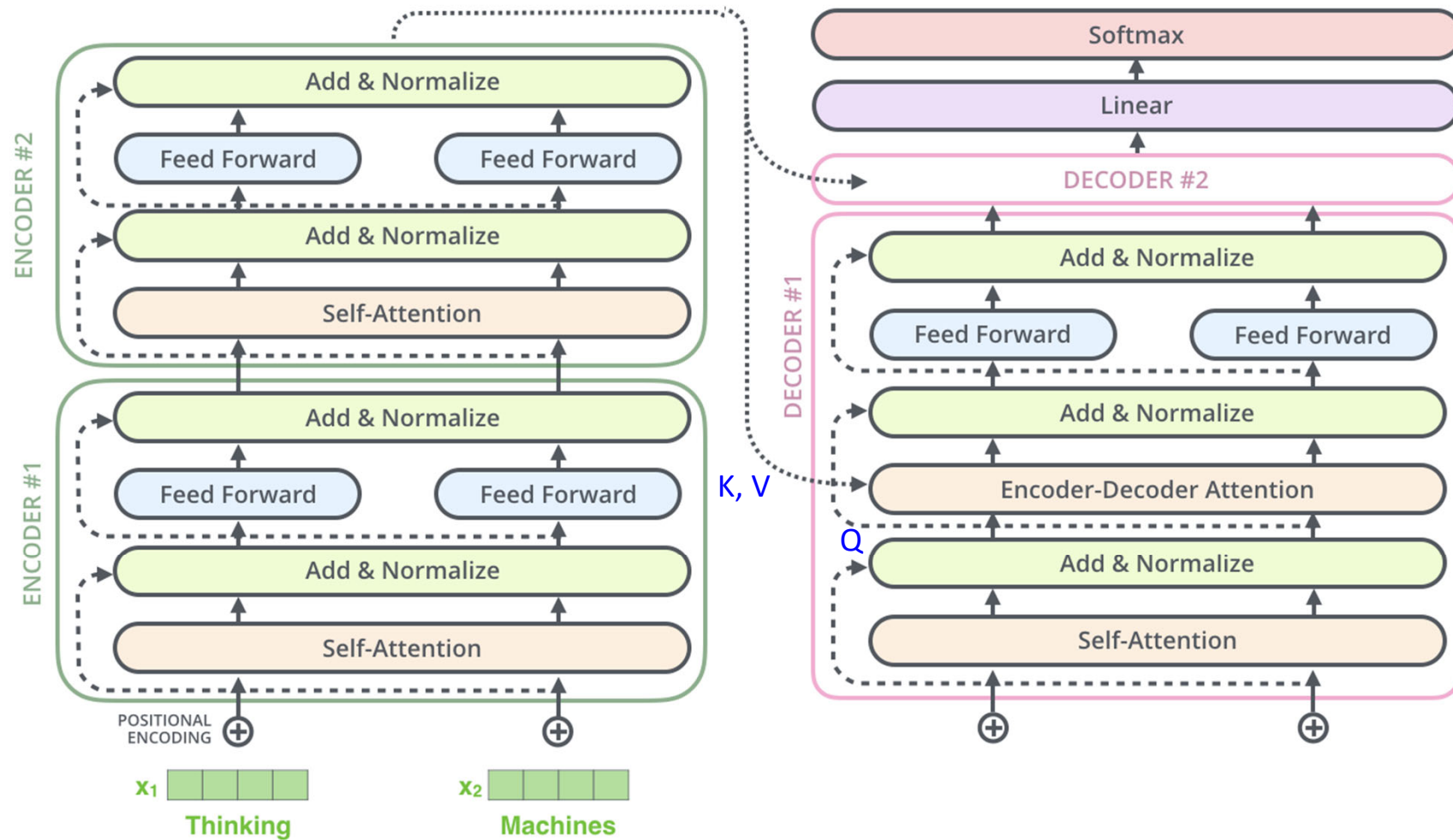


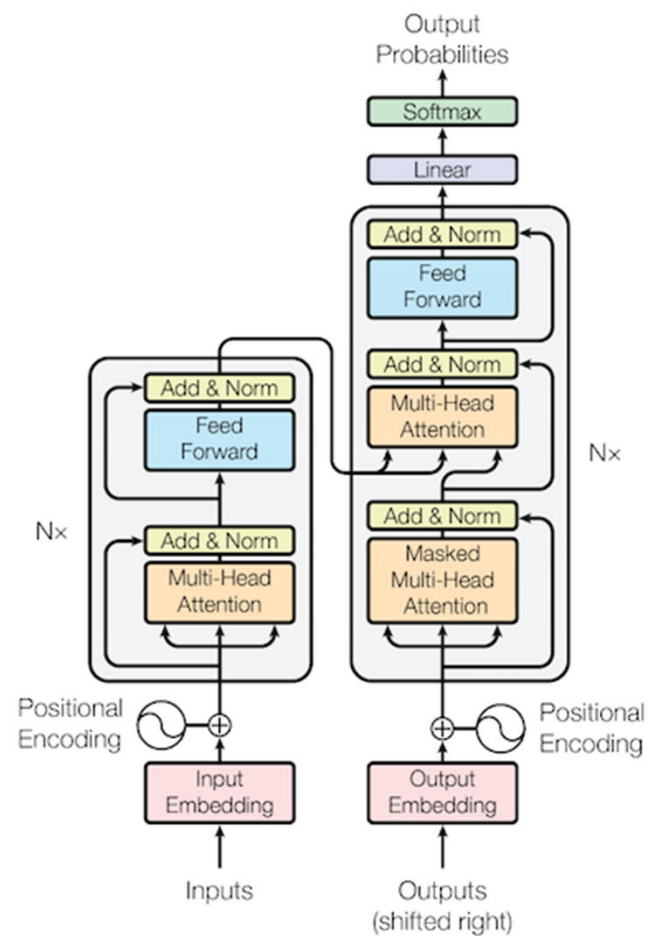
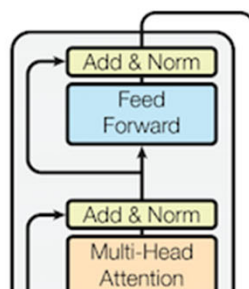
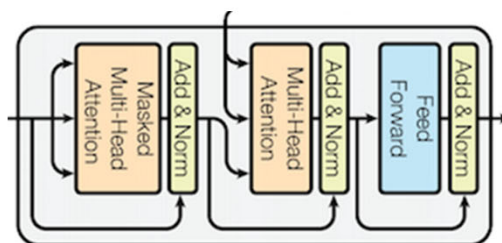
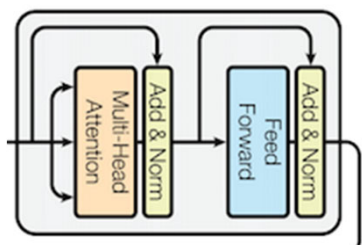
Layer Normalization



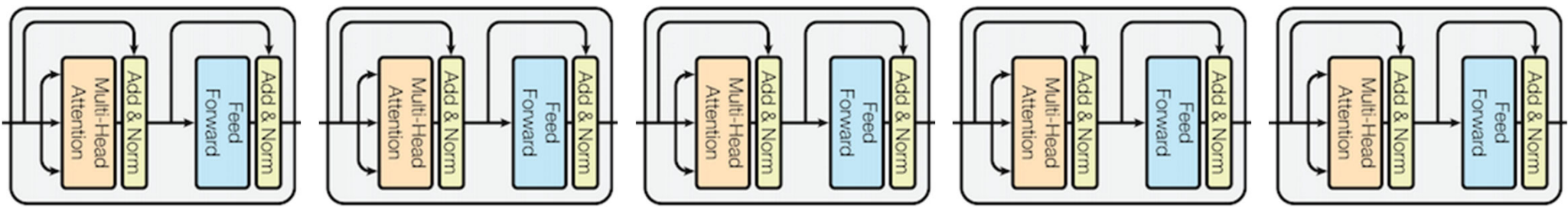
Slides from Ming Li, University of Waterloo, CS 886 Deep Learning and NLP

The complete transformer





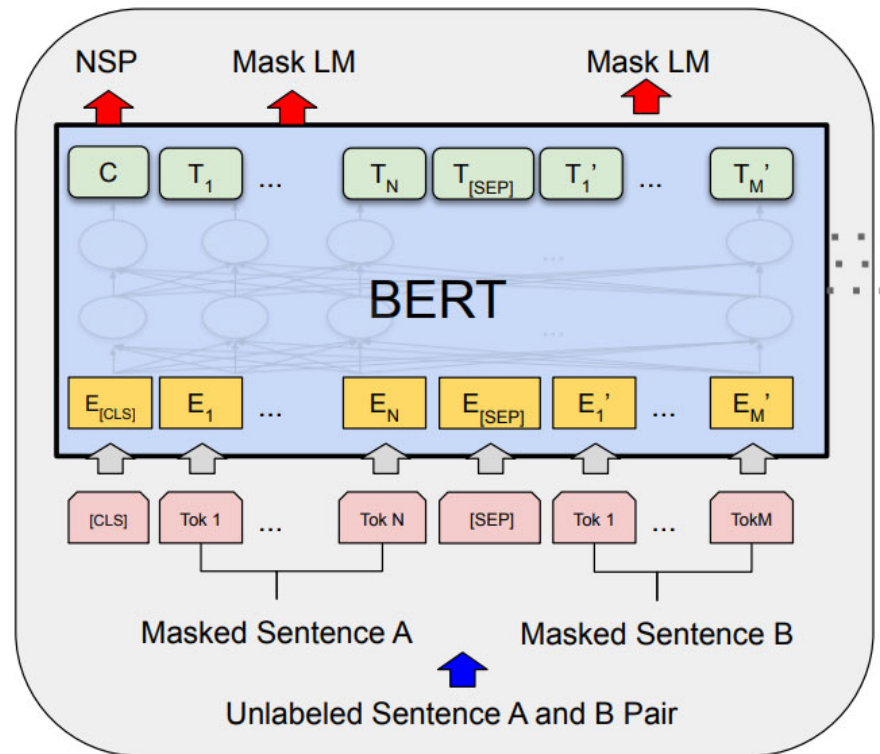
BERT (Stack Encoder Blocks)

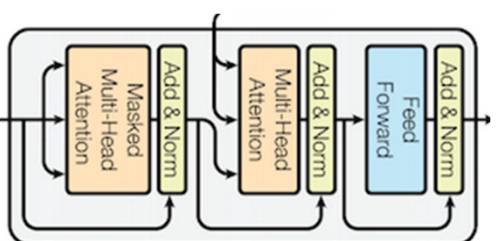
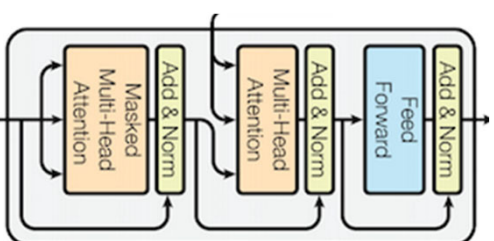
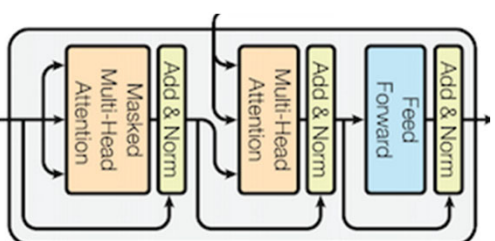
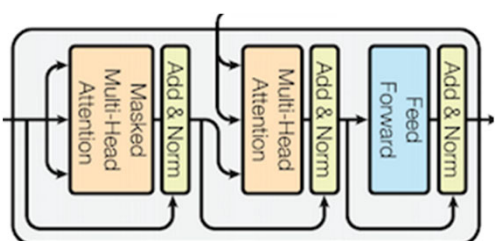


BERT (Bidirectional Encoder Representations from Transformers)

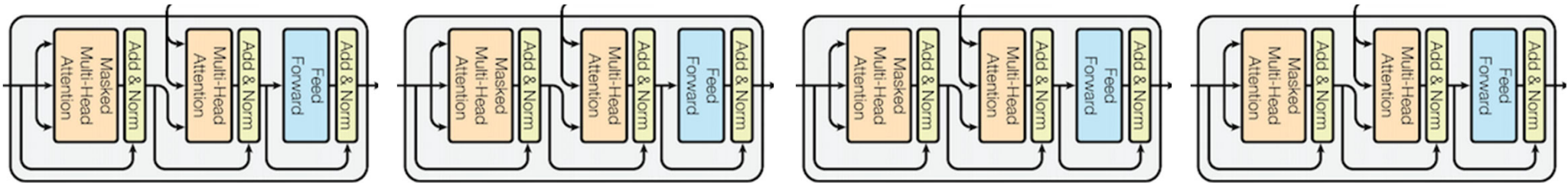
- BERT jointly encodes the right and left context of a word in a sentence to improve the learned feature representations
- BERT is trained on two pre-text tasks in self-supervised manner
 - Masked Language Model (MLM)
 - Mask fixed percentage (15%) of words in a sentence predict these masked words
 - In predicting the masked words, the model learns the bidirectional context.
 - Next Sentence Prediction (NSP)
 - Given a pair of sentences A and B the model predicts a binary label i.e., whether the pair is valid or not from the original document
 - Pair is formed such that B is the actual sentence (next to A) 50% of the time, and B is a random sentence for other 50% of the time.

BERT





GPT (Stack Decoder Blocks)



BERT and GPT

https://www.youtube.com/shorts/BEt_BACGw6g

Vision Transformers

Mubarak Shah
shah@crcv.ucf.edu

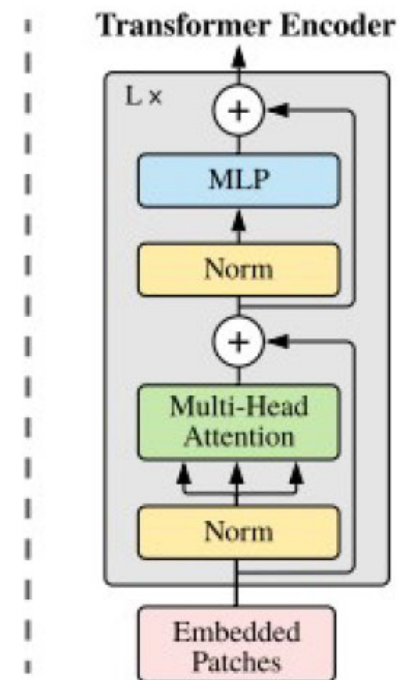
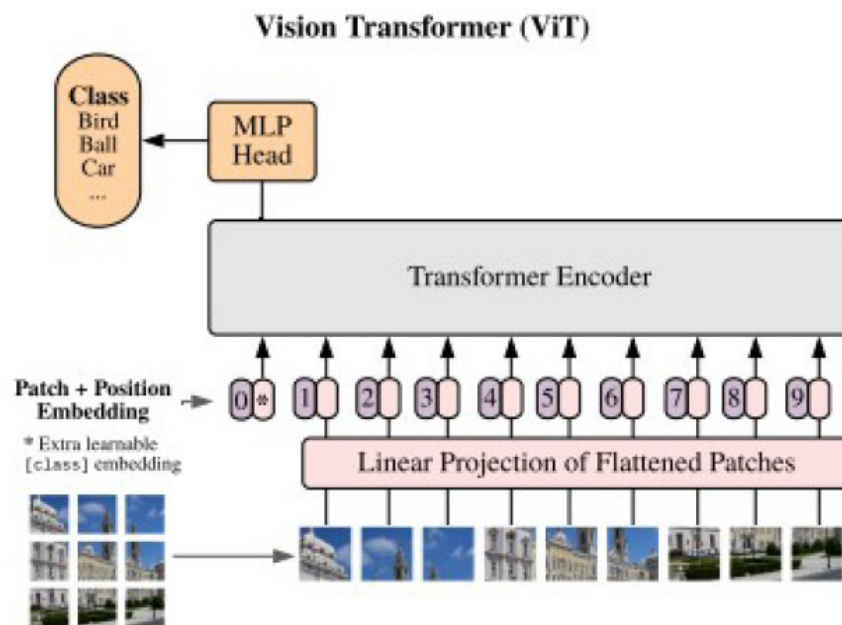
Vision Transformer (ViT)

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” arXiv preprint arXiv:2010.11929, 2020. ([ICLR 2021; 26,938 citations!](#))

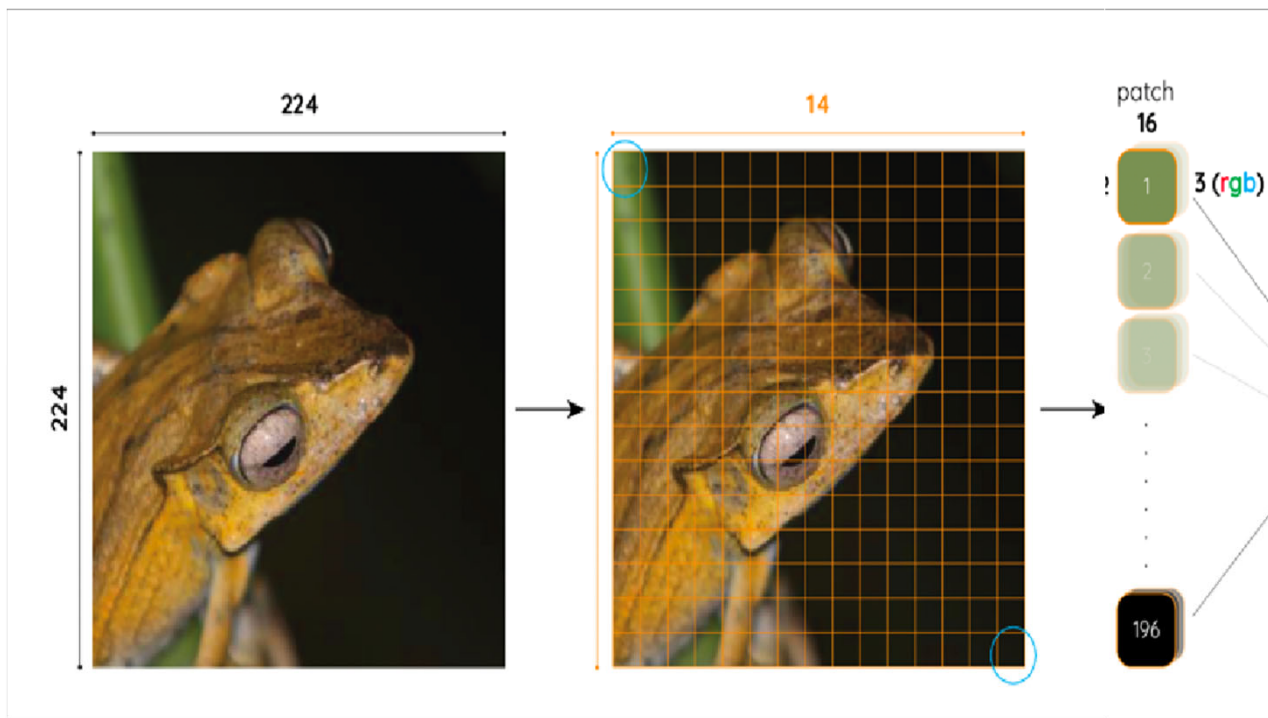
Vision Transformer (ViT)

- Naive application of self-attention to images requires high computation
- Divide an image into 16x16 patches (tokens)
- Transformers need to be trained on large datasets
- ViT attains excellent results when pre-trained on JFT-300M
 - 88:55% on ImageNet,
 - 90:72% on ImageNet-Real,
 - 94:55% on CIFAR-100,
 - 77:63% on the VTAB suite of 19 tasks

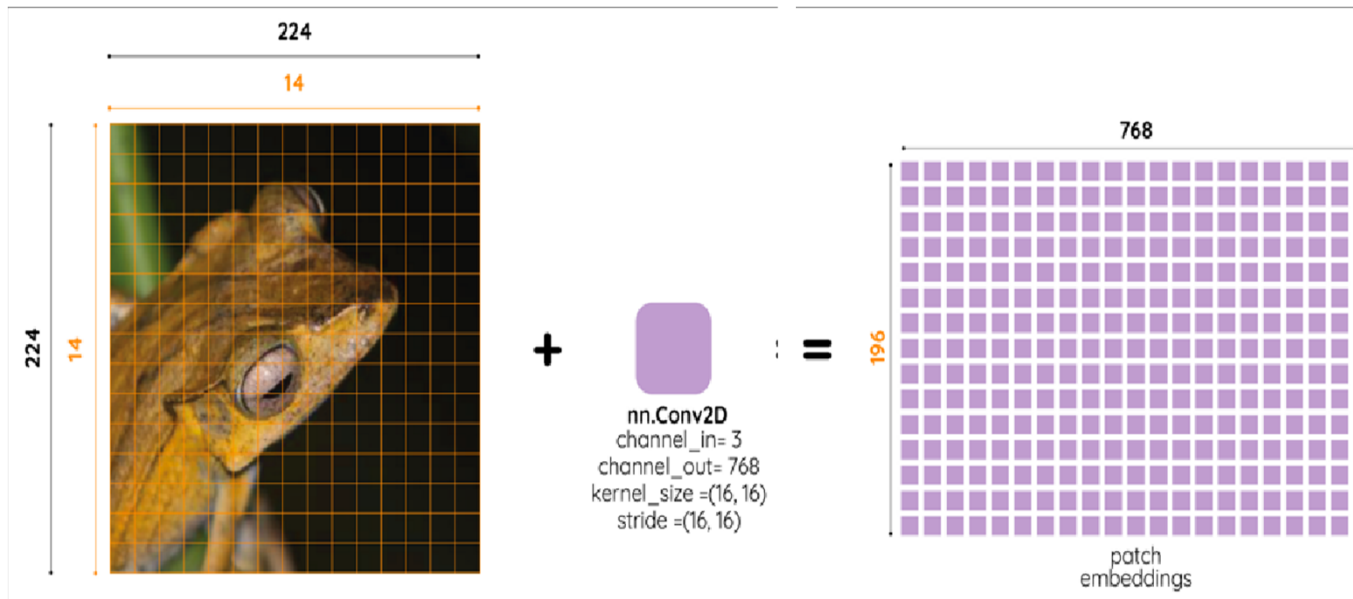
Vision Transformer (ViT)



Divide image into 16x16 patches

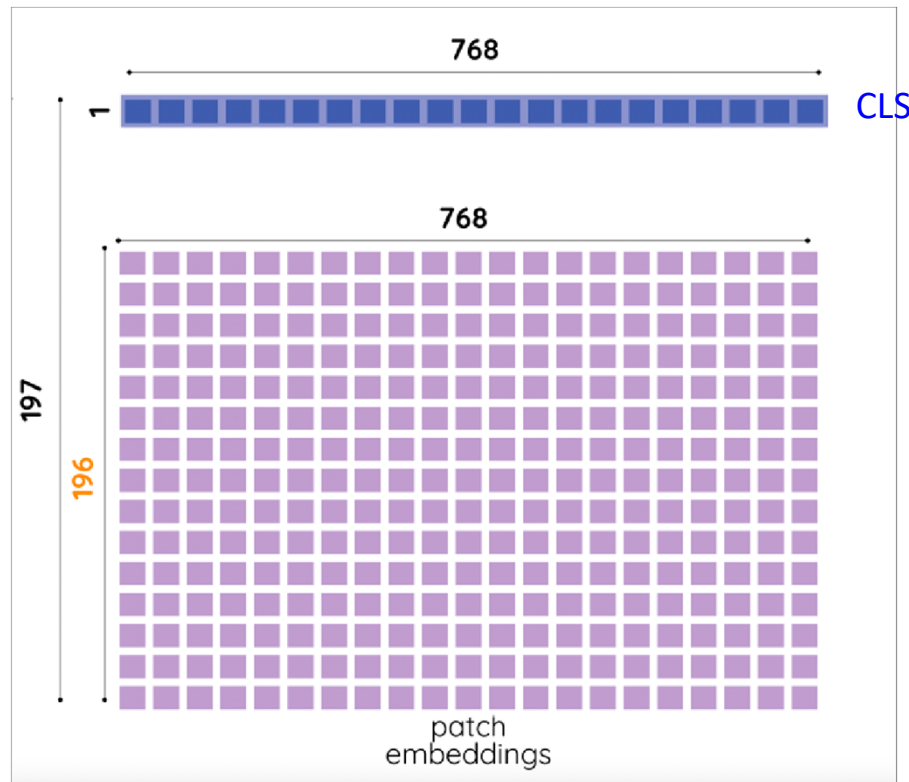


Generate embedding for each patch

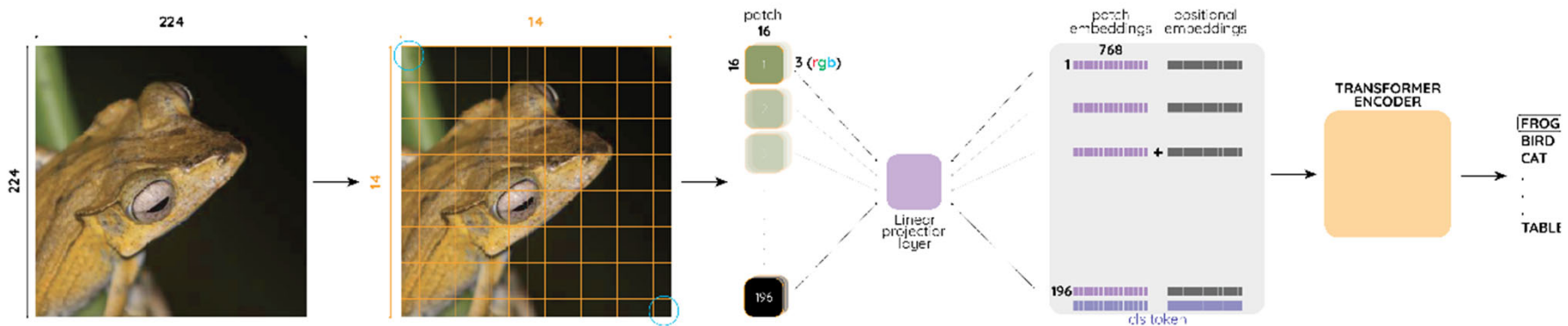


Slide credit: Piotr Mazurek

CLS (Classification) Token



Complete ViT



Slide credit: Piotr Mazurek

ViT Model Variants

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.

Results

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	88.55 ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	90.72 ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	99.50 ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	—
CIFAR-100	94.55 ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	—
Oxford-IIIT Pets	97.56 ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	—
Oxford Flowers-102	99.68 ± 0.02	99.74 ± 0.00	99.61 ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	77.63 ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	—
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Attention

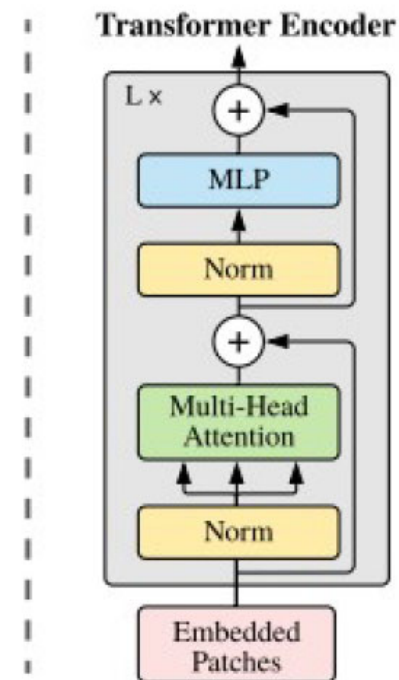
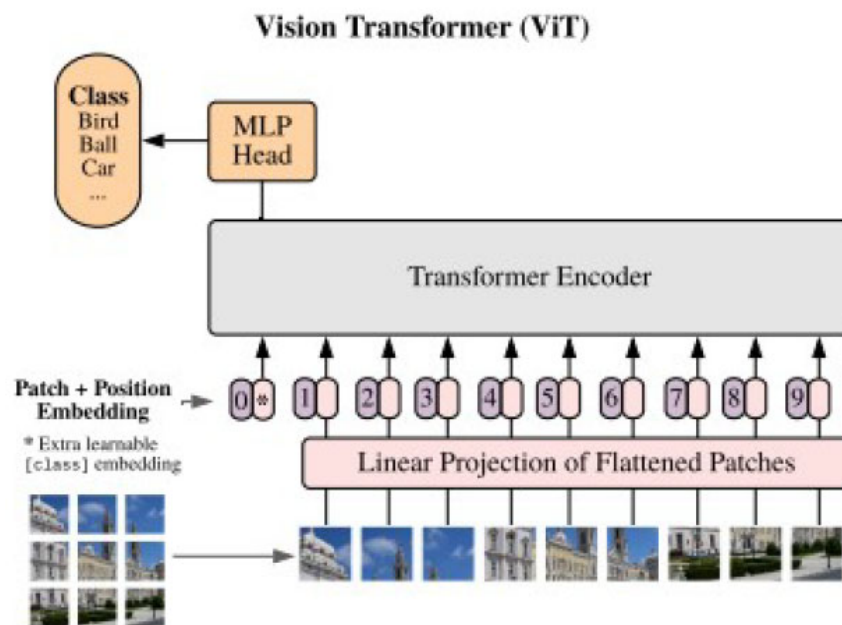
Input



Attention



Vision Transformer (ViT)



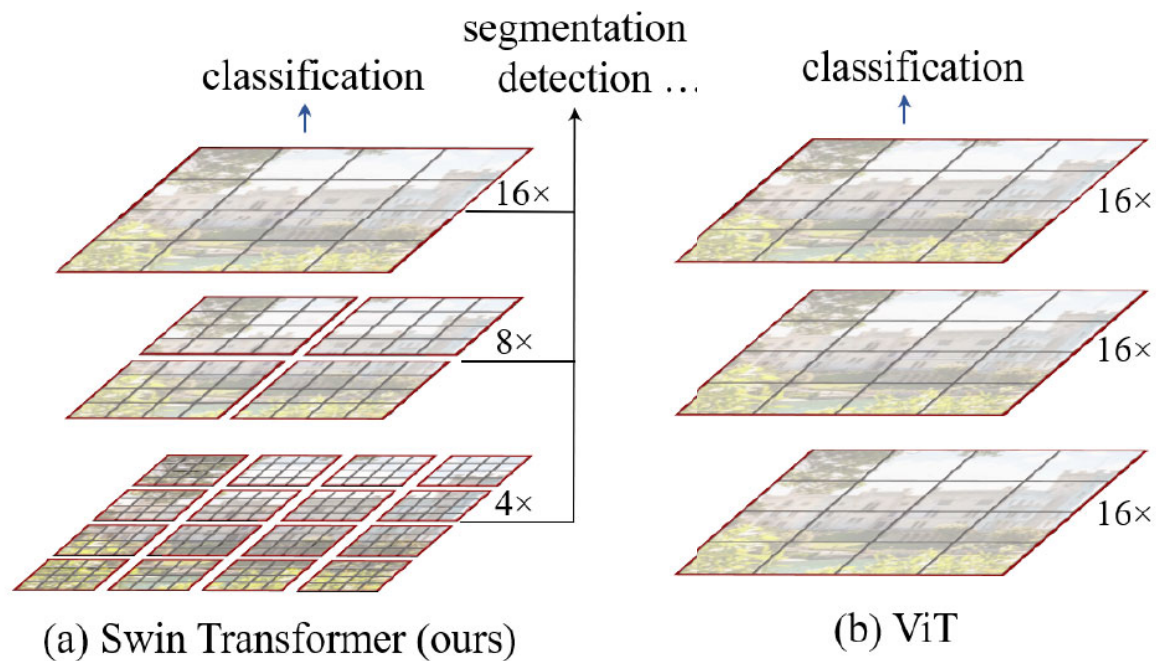
SWIN

- Z Liu, Y Lin, Y Cao, H Hu, Y Wei, Z Zhang, S Lin, “Swin transformer: Hierarchical vision transformer using shifted windows”, ICCV-2021. (Marr Prize) 13,035 Citations

SWIN

- Adapting Transformer from language to vision is challenging
- Differences between language and vision Domains
 - Large variations in the scale of visual entities
 - High resolution of pixels in images compared to words in text
- To address these differences, SWIN proposes
 - A hierarchical Transformer whose representation is computed with Shifted windows
 - Shifted Windowing limit attention
 - To local windows and
 - Allowing cross window connections

Hierarchical Feature Maps and Local Attention



SWIN

- 87.3 top-1 accuracy on ImageNet-1K
- Dense prediction tasks such as
 - Object detection (58.7 box AP and 51.1 mask AP on COCO)
 - Semantic segmentation (53.5 mIoU on ADE20K)
- Performance surpasses the previous state-of-the art by
 - +2.7 box AP and +2.6 mask AP on COCO, and
 - +3.2 mIoU on ADE20K,

Thank You!

Any questions?