UCF Center for Research in Computer Vision

Introduction to Transformers

Advanced Computer Vision Spring2025 Lecture-3-Transformers

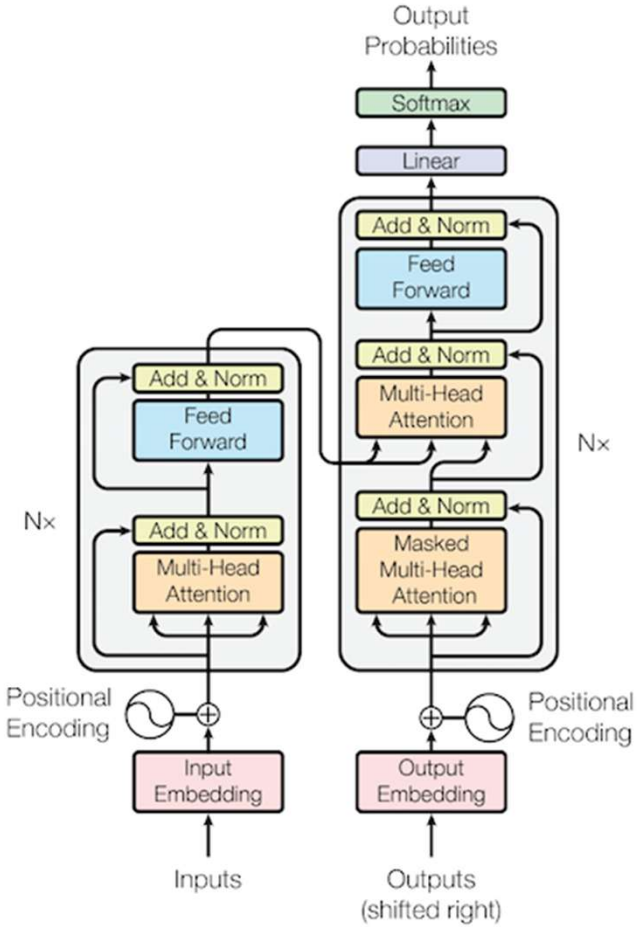
Mubarak Shah

shah@crcv.ucf.edu

UCF Center for Research in Computer Vision

Contents

- Basics
  - What is Transformer?
  - Self-Attention
    - Query, Key, Value
  - Position encoding
  - Encoder-Decoder
- BERT
- Vision Transformers
  - VIT
  - SWIN



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

UCF Center for Research in Computer Vision

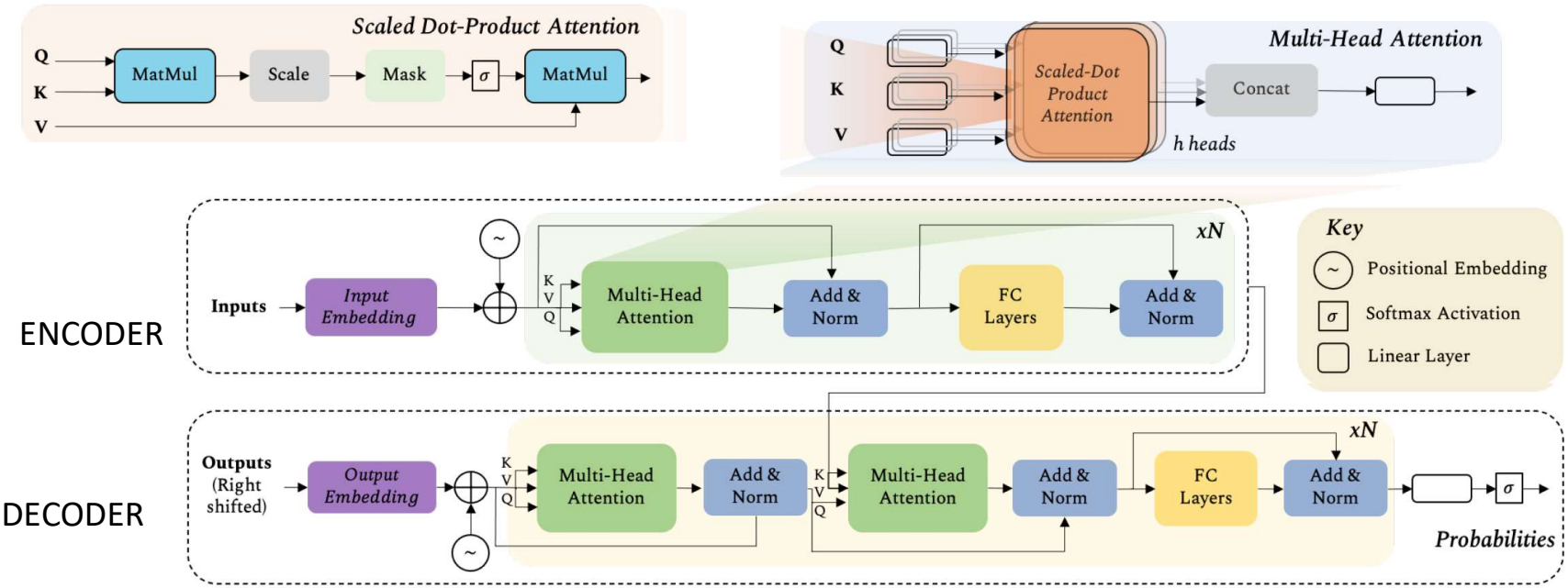
Natural Language Processing

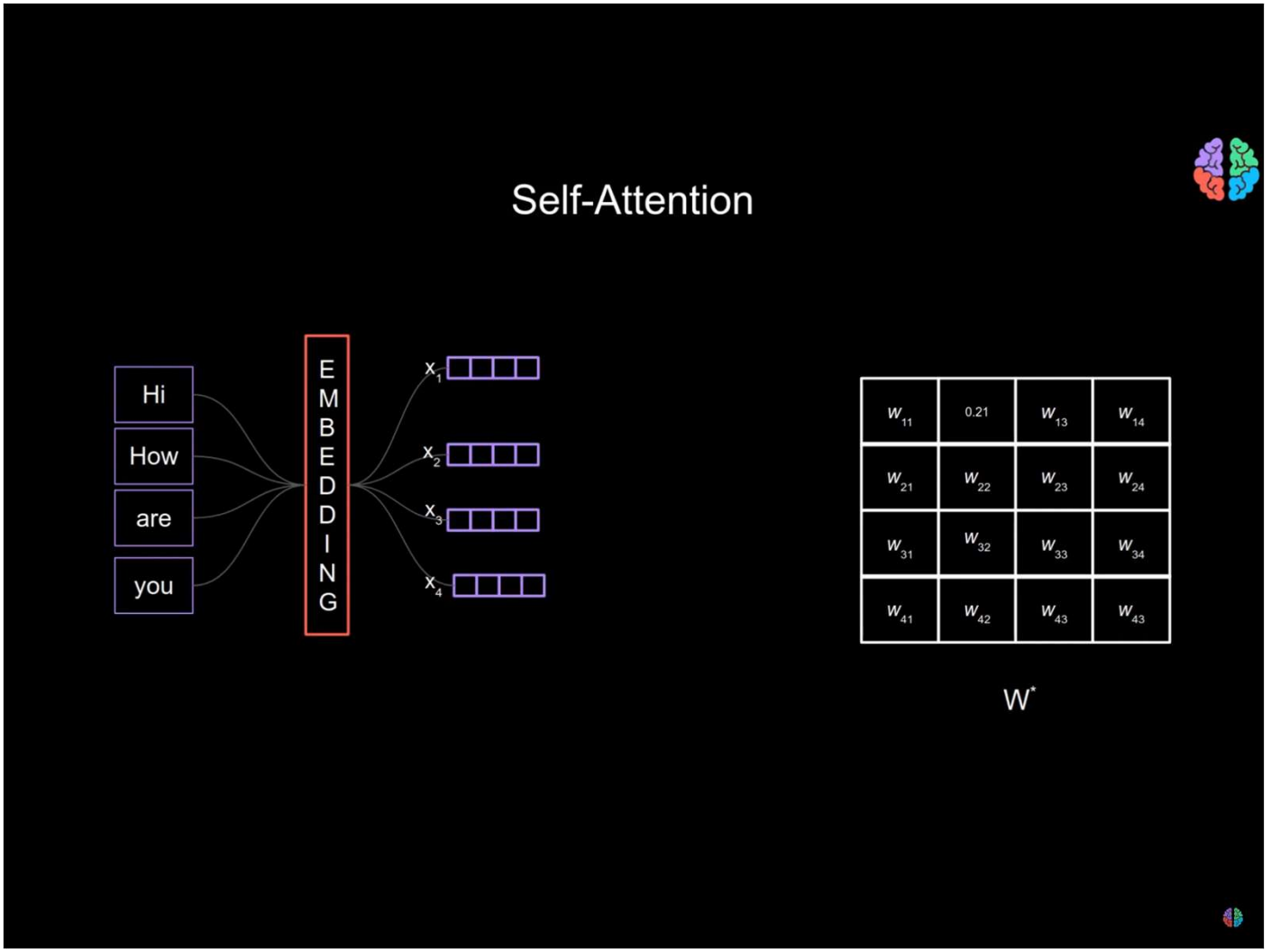
- BERT (Bidirectional Encoder Representations from Transformers),
- GPT (Generative Pre-trained **Transformer**) v1-4,
- RoBERTa (Robustly Optimized BERT Pre-training)
- T5 (Text-to-Text Transfer Transformer)
- Gemini
- LLaMA
- Most recent LLMs and VLMs

Transformer Basics

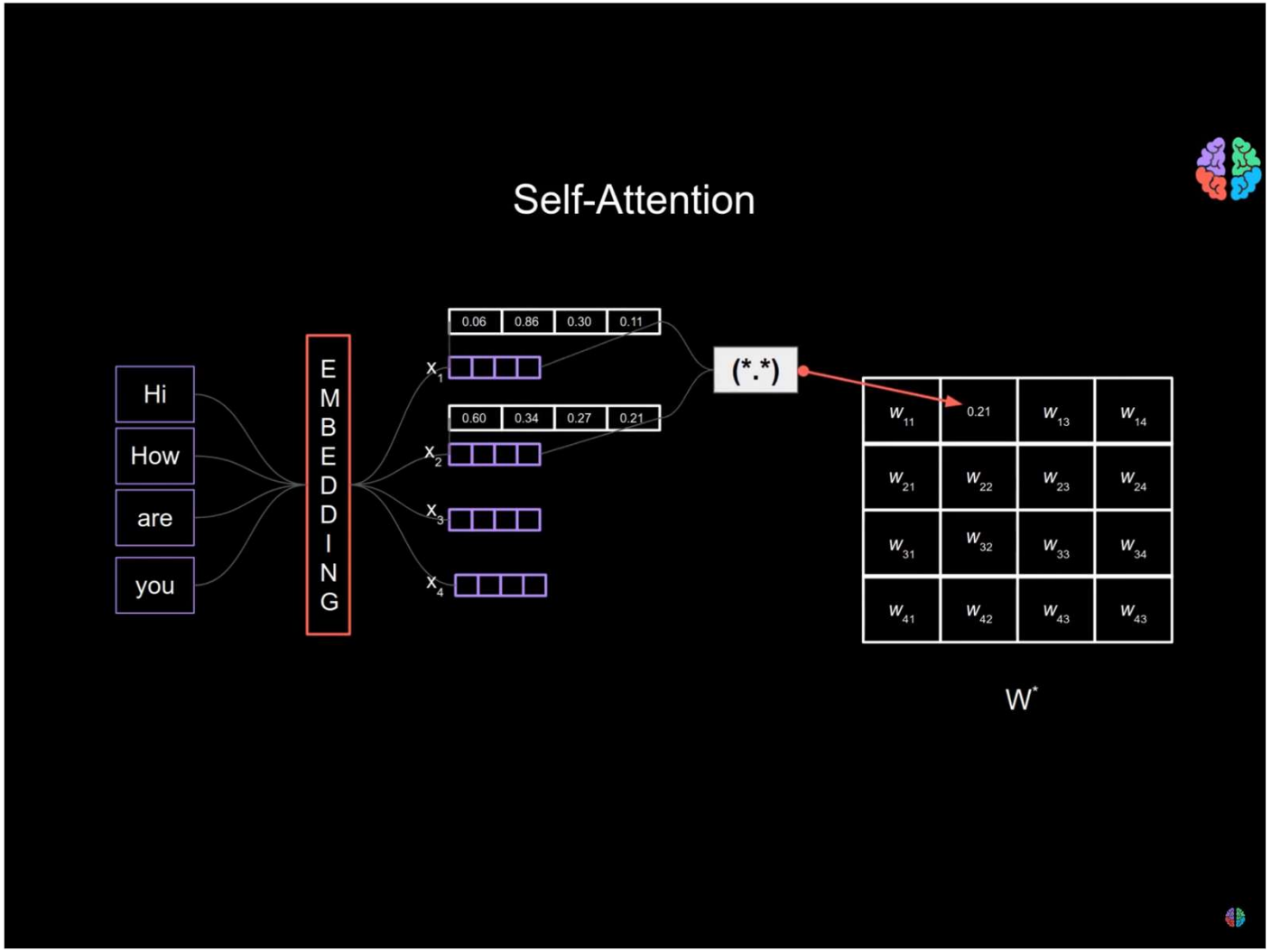
- It consists of Encoder and Decoder Blocks
- Main components of each block:
  - Self-Attention
  - Layer Normalization
  - Feed Forward Network

Transformer

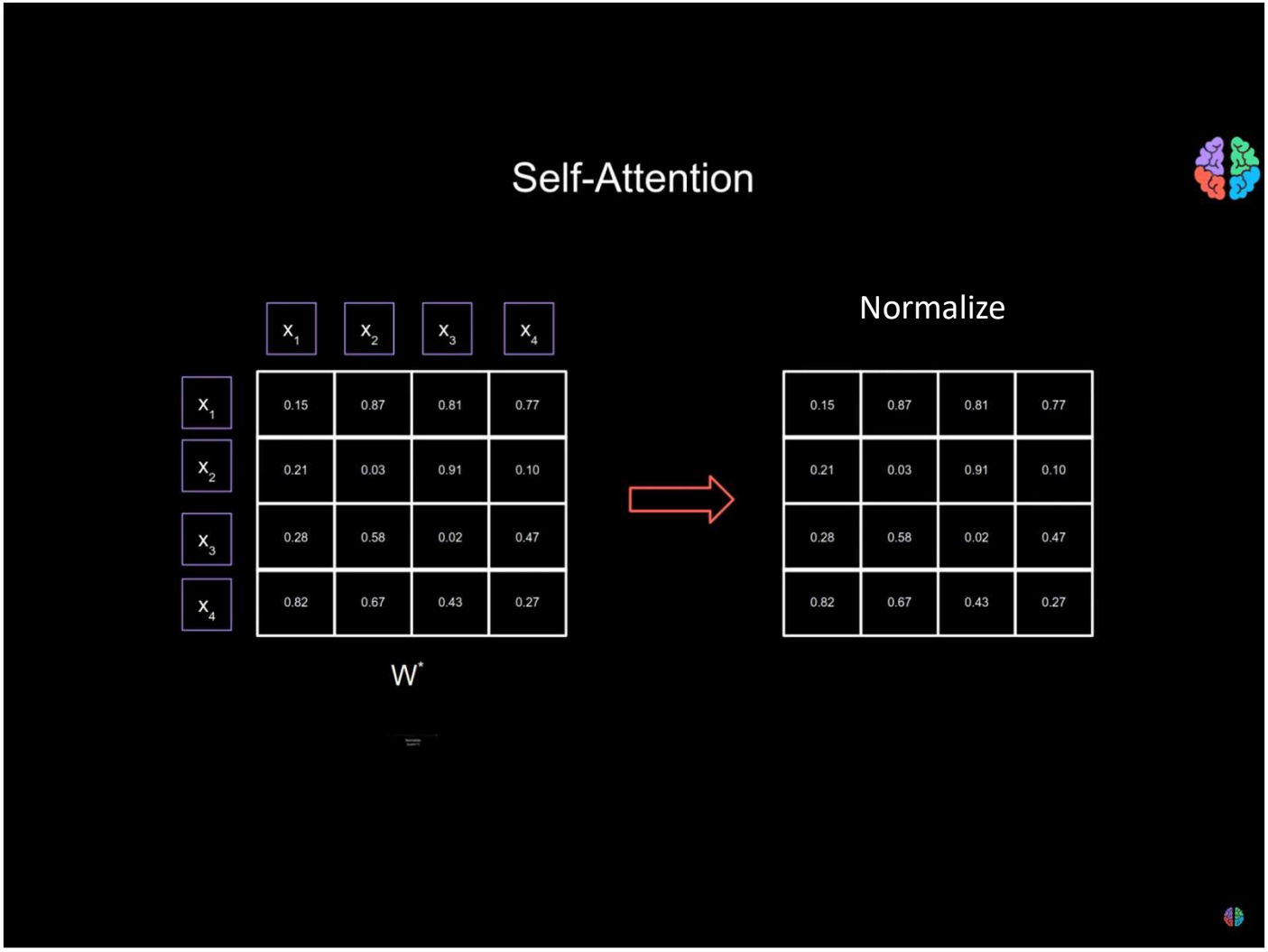




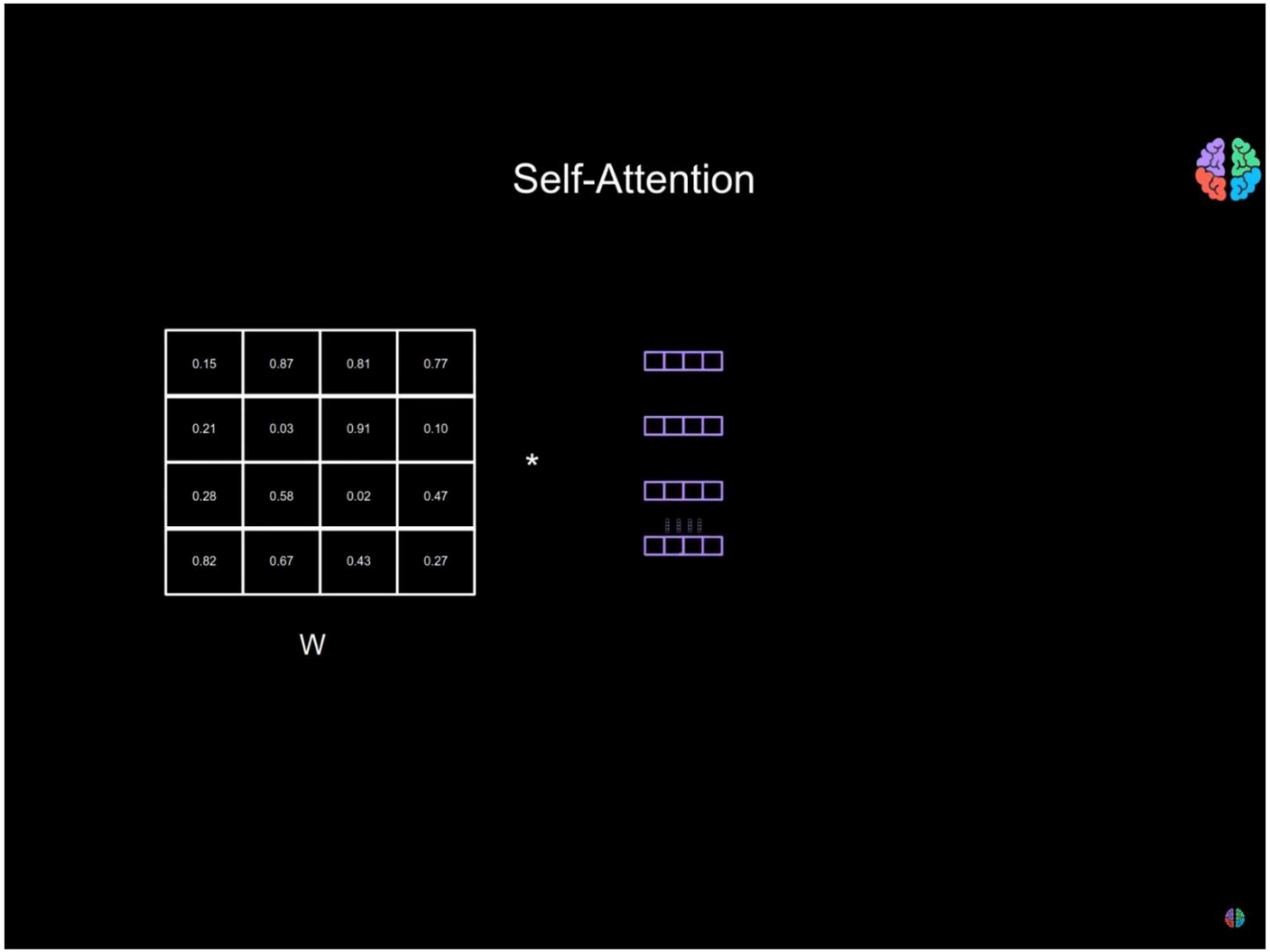
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



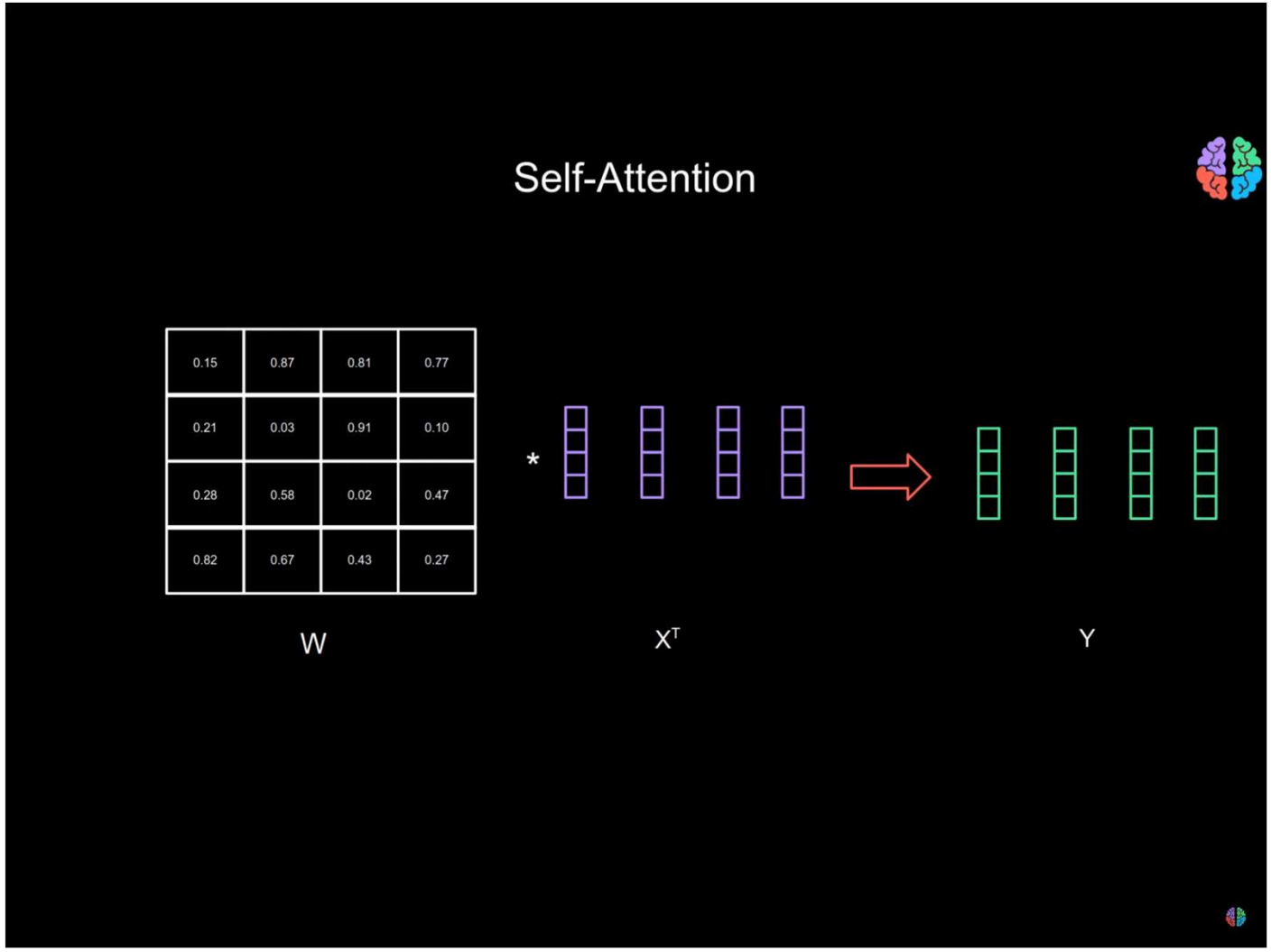
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



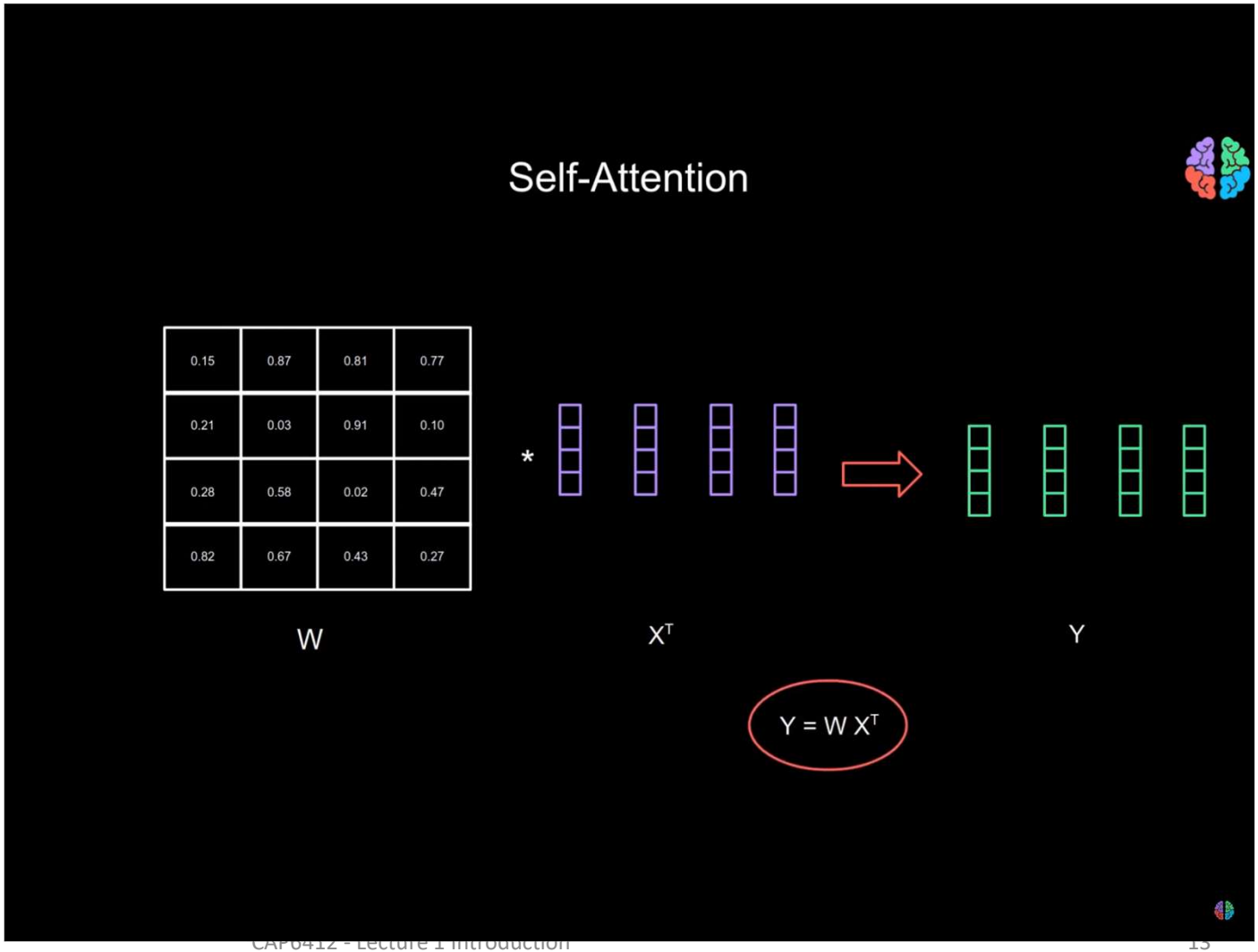
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



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



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



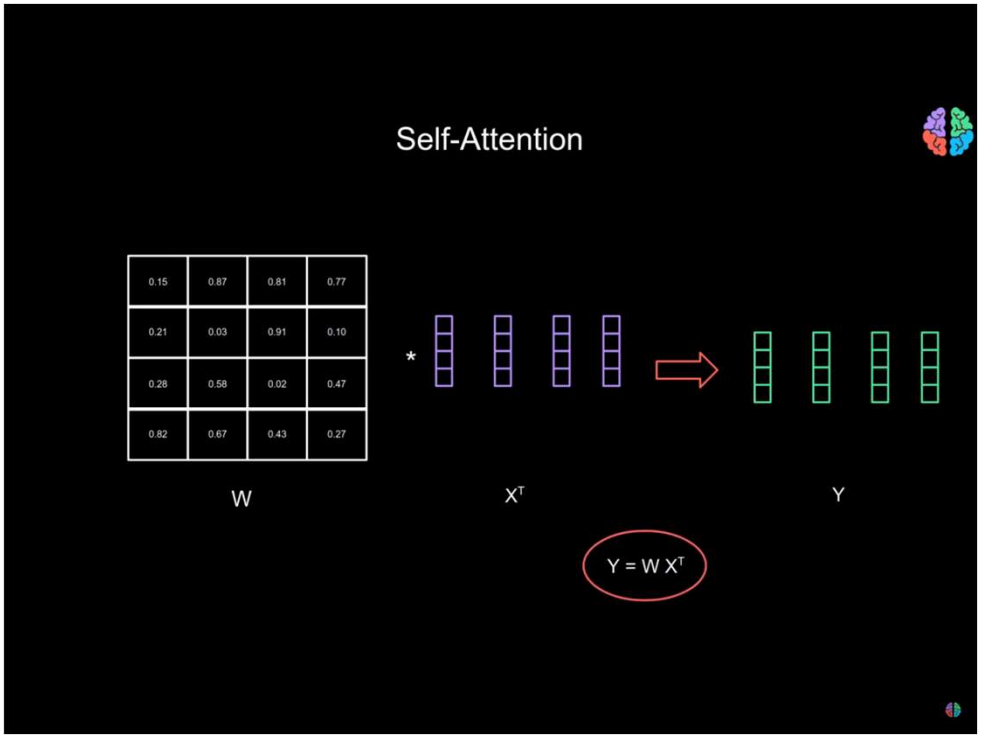
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites

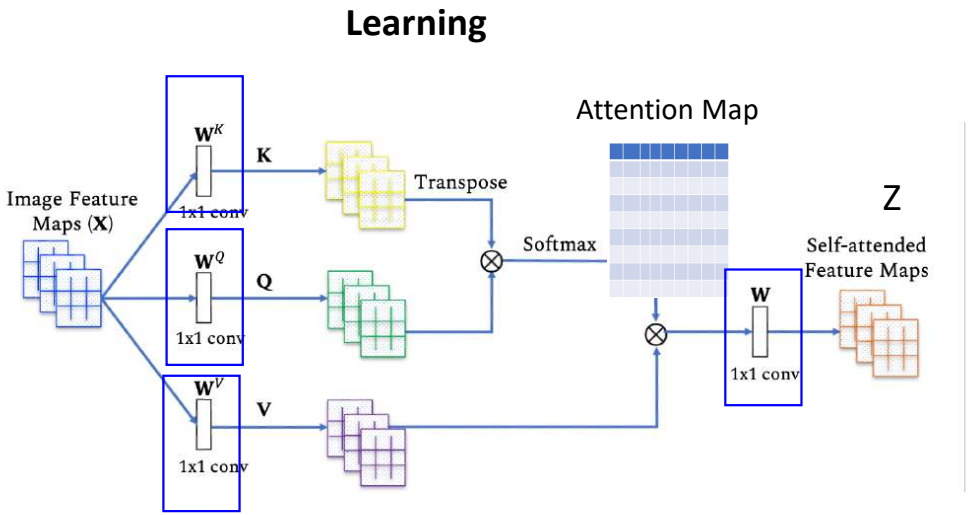
UCF Center for Research in Computer Vision

Self-Attention

- So far no learning!

Self-Attention of Image Features



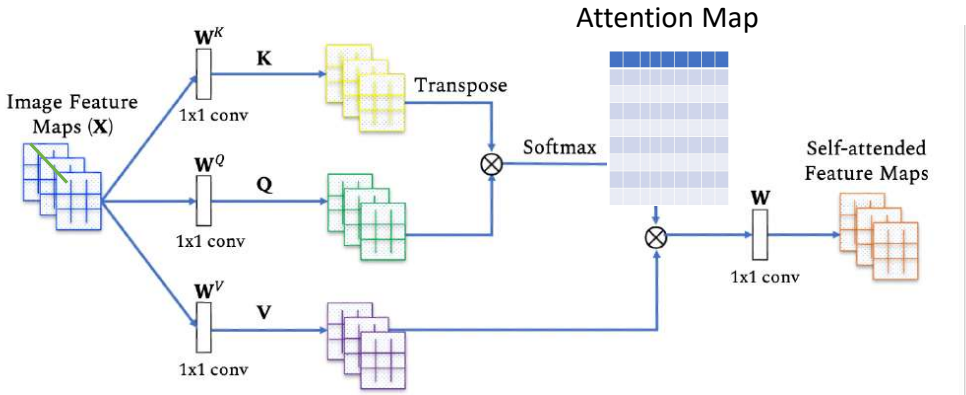


Self-Attention (Matrix Explanation)

Lets denote a sequence of n entities $(\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n)$ 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 $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^K$ and $\mathbf{V} = \mathbf{X}\mathbf{W}^V$. The output $\mathbf{Z} \in \mathbb{R}^{n \times d_v}$ of the

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



Self-Attention

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

by $\mathbf{X} \in \mathbb{R}^{n \times d}$ 9x3

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

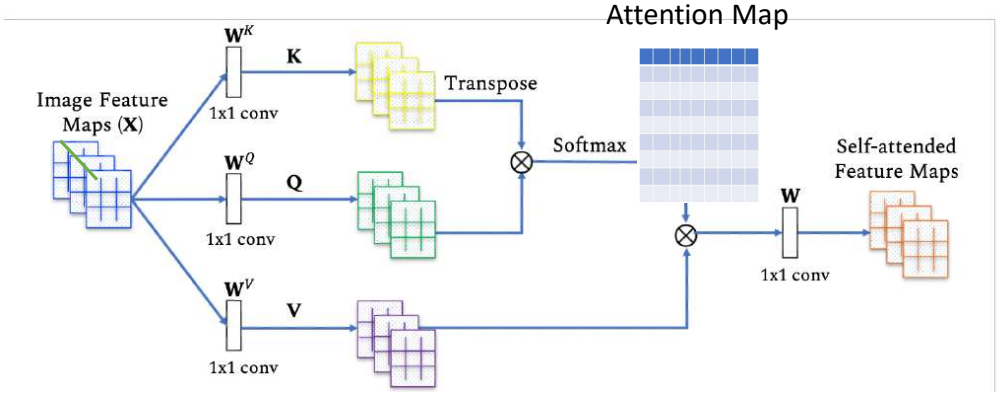
3x3 3x3 3x3

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

9x3 9x3 9x3 9x3

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

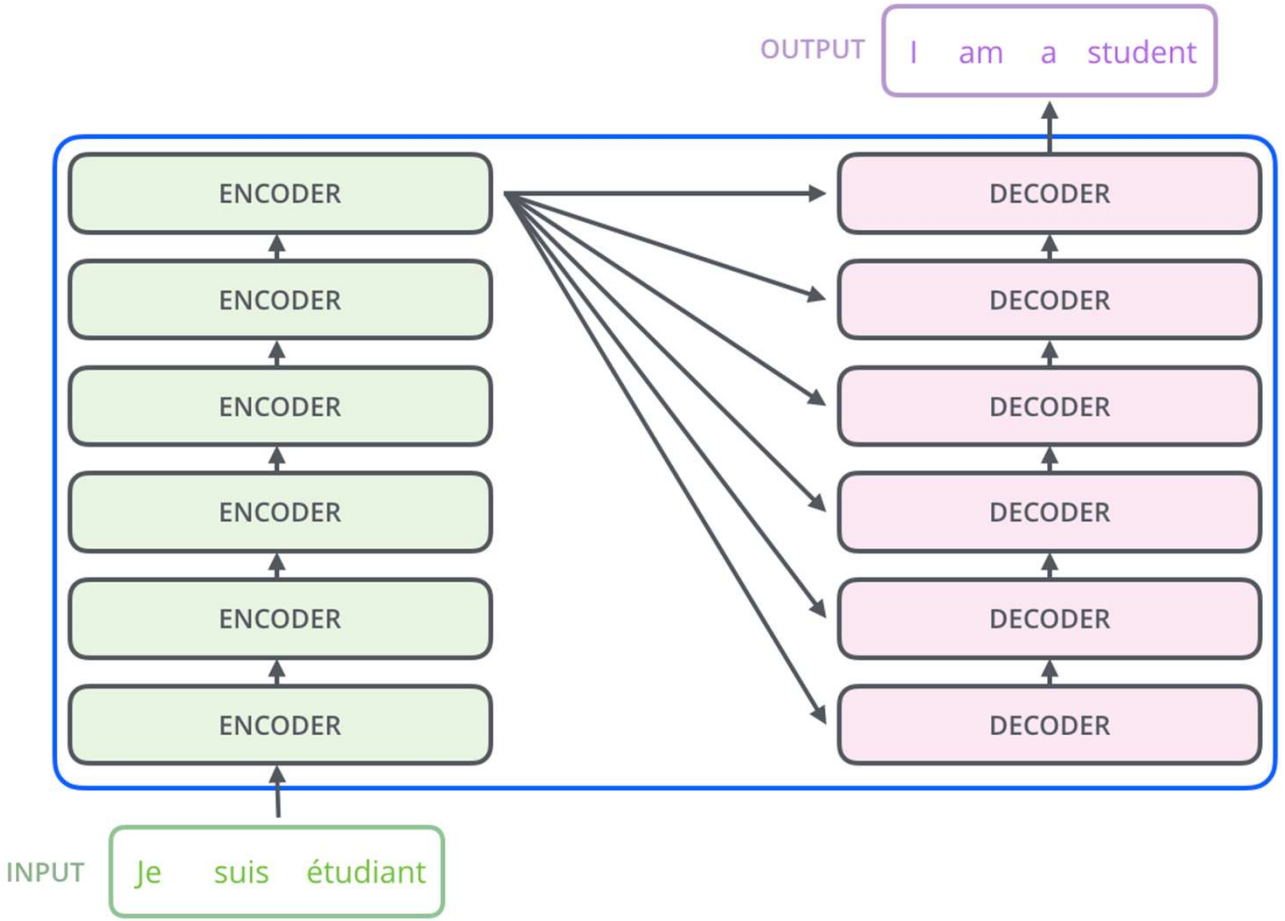
9x3 9x9 9x3



Transformers (Attention is all you need 2017)

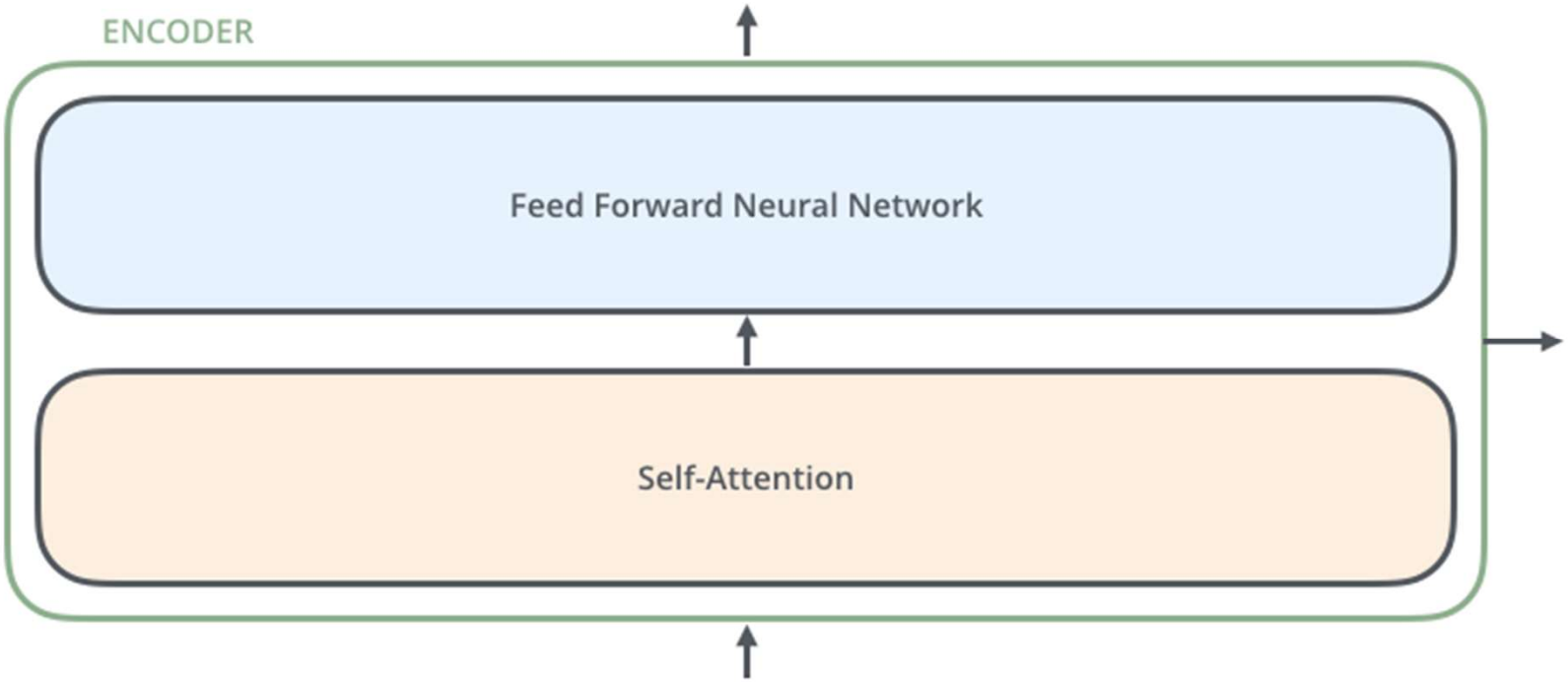
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in NeurIPS, 2017. (**148579** citations)
- Two valuable sources
  - http://nlp.seas.harvard.edu/2018/04/03/attention.html
  - https://jalammar.github.io/illustrated-transformer/ (slides come from this source)
- Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Machine Translation



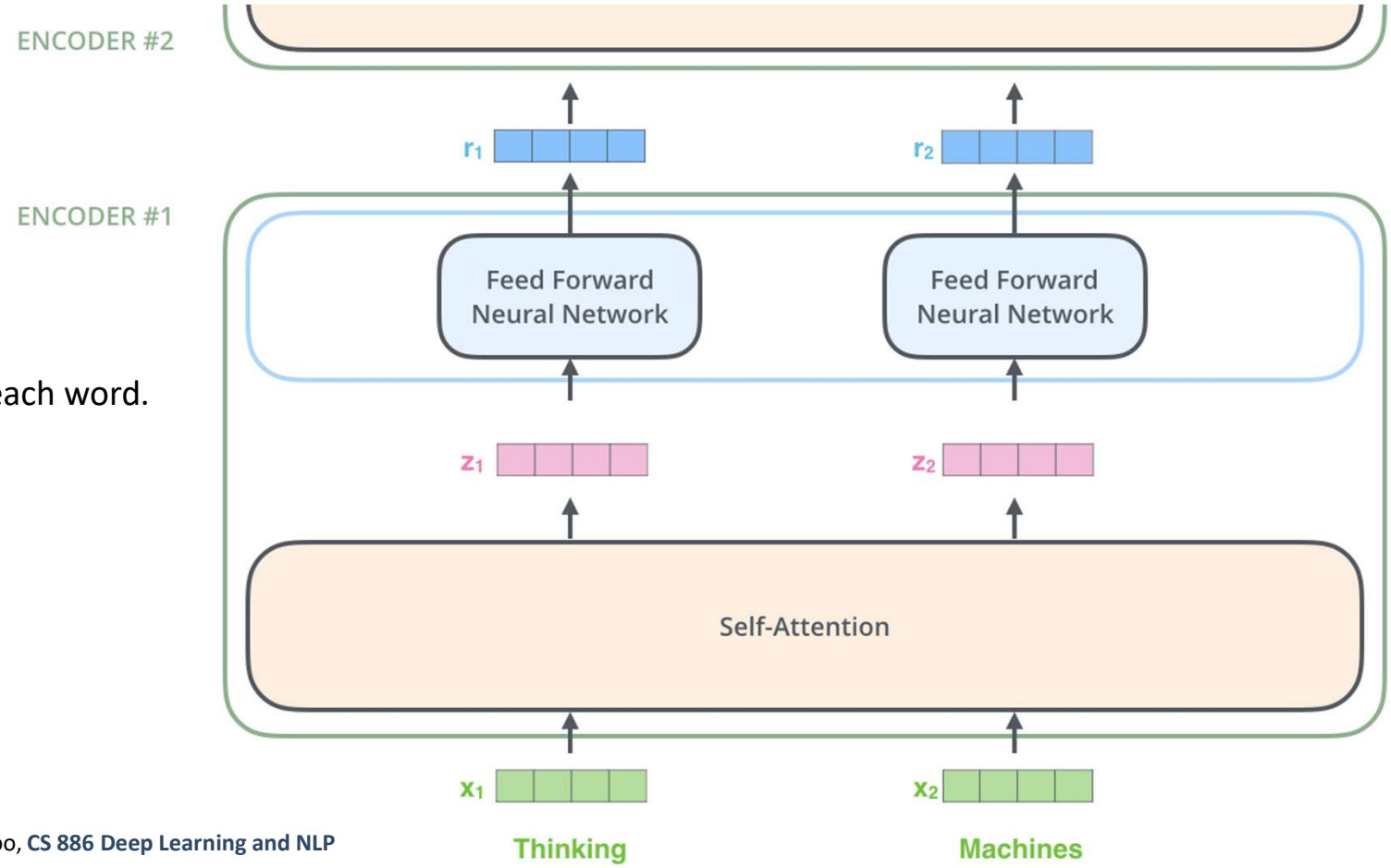
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

An Encoder Block: same structure, different parameters



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

Encoder



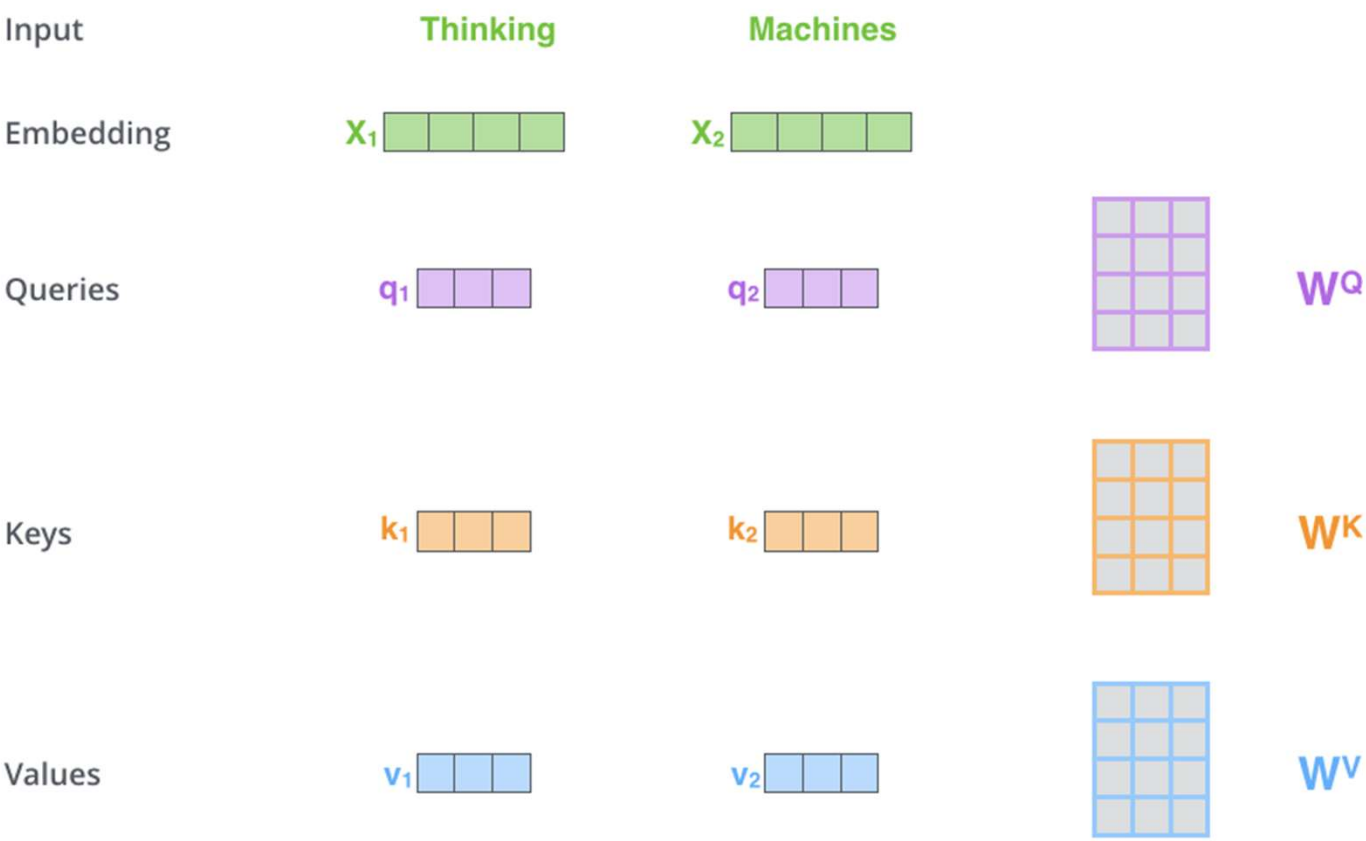
The ffnn is independent for each word.
Hence can be parallelized.

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

Self Attention

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

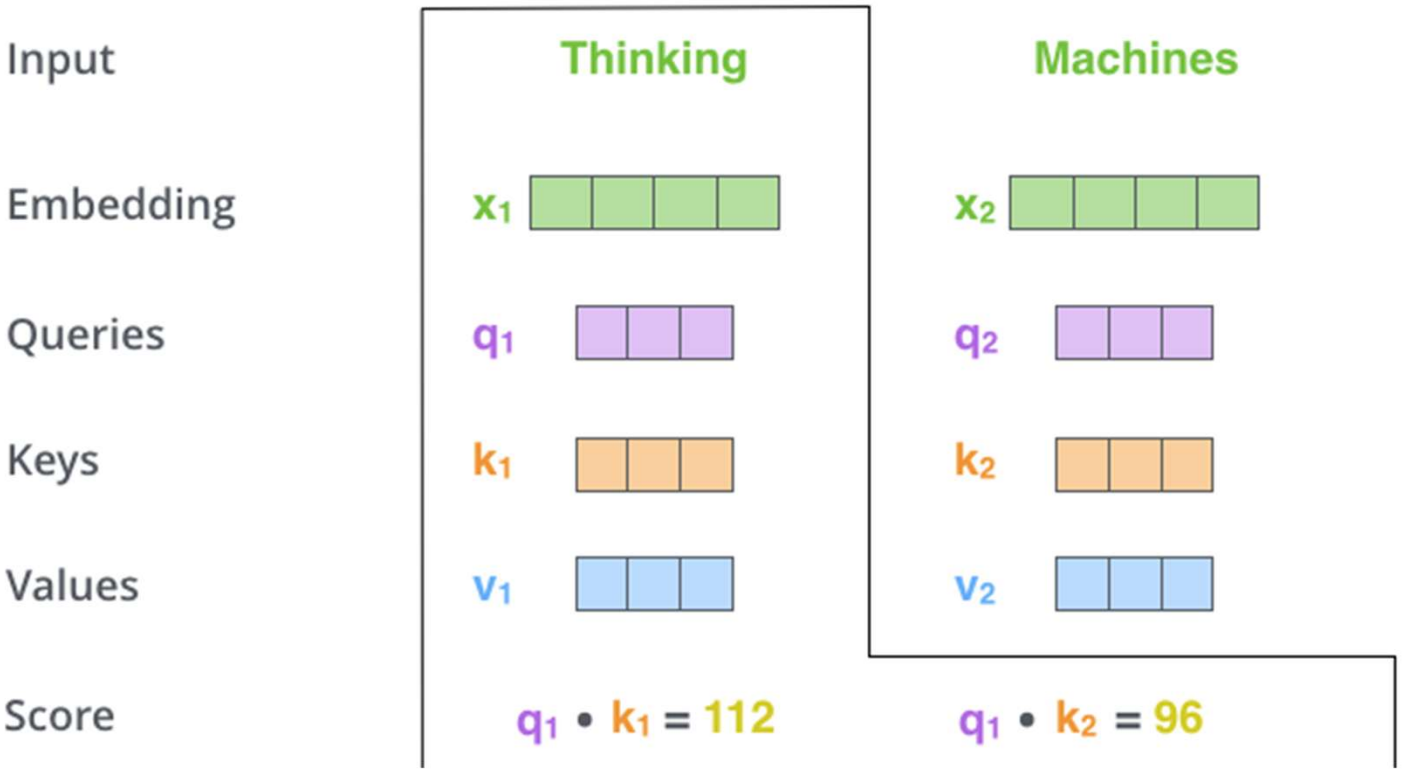
- $q_i = x_i W^Q$
- $K_i = x_i W^K$
- $V_i = x_i W^V$



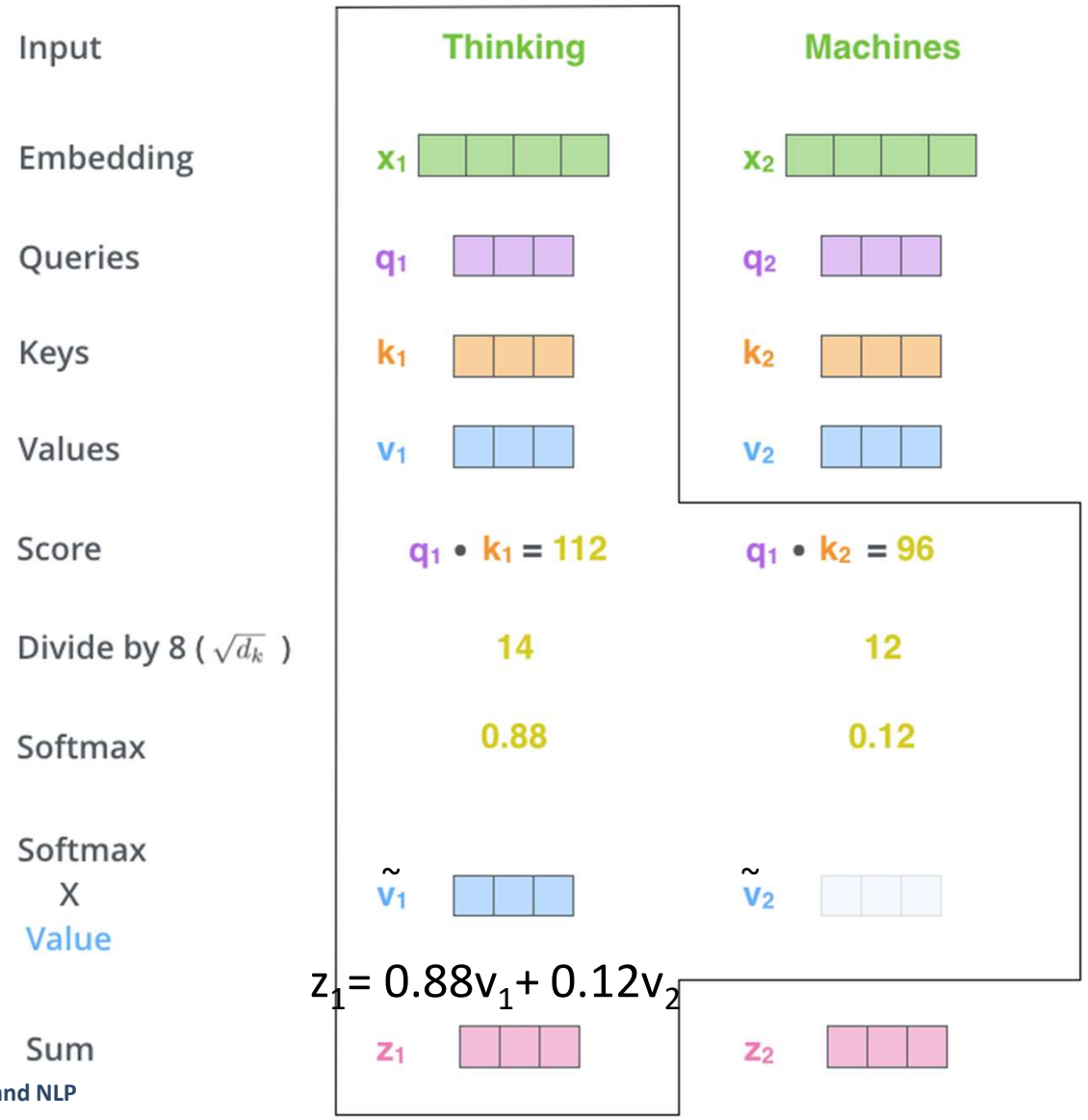
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Self Attention

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



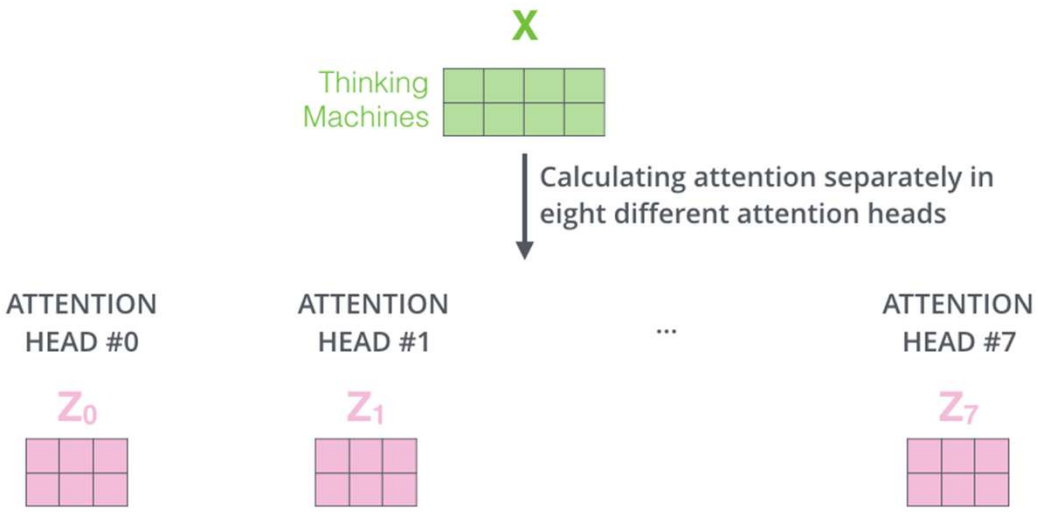
Self Attention



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

Multiple heads

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



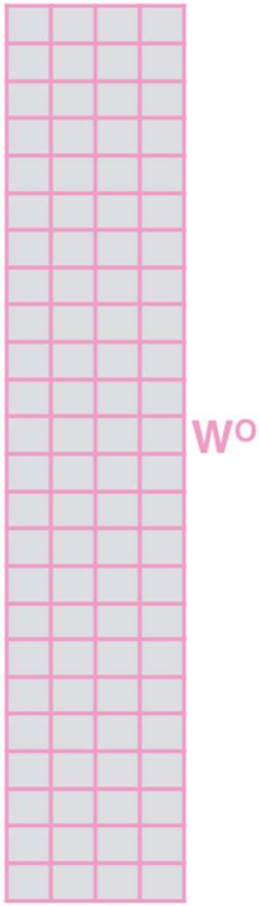
Attention and Transformers

1) Concatenate all the attention heads



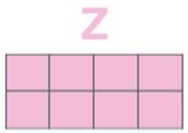
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

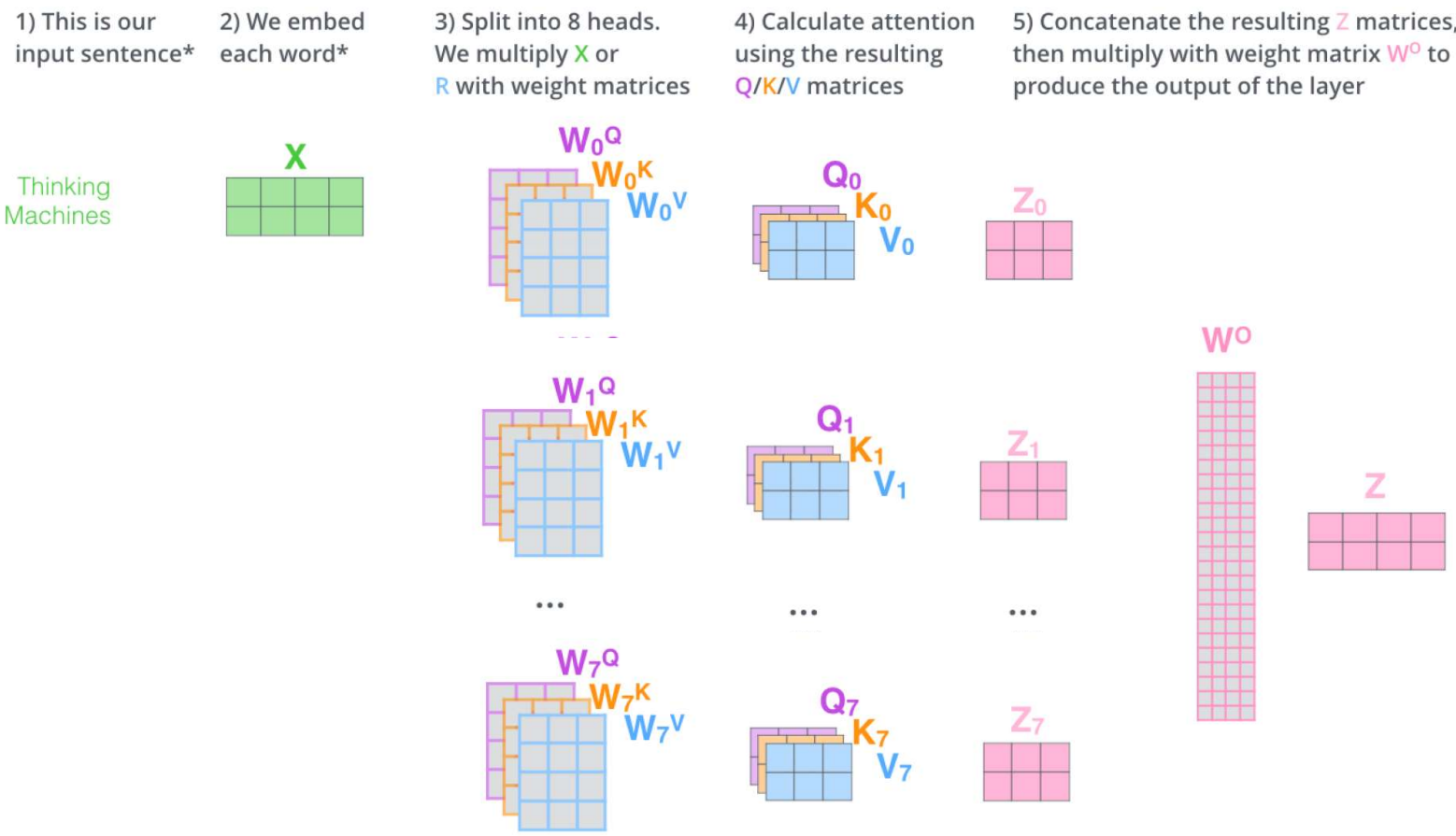


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

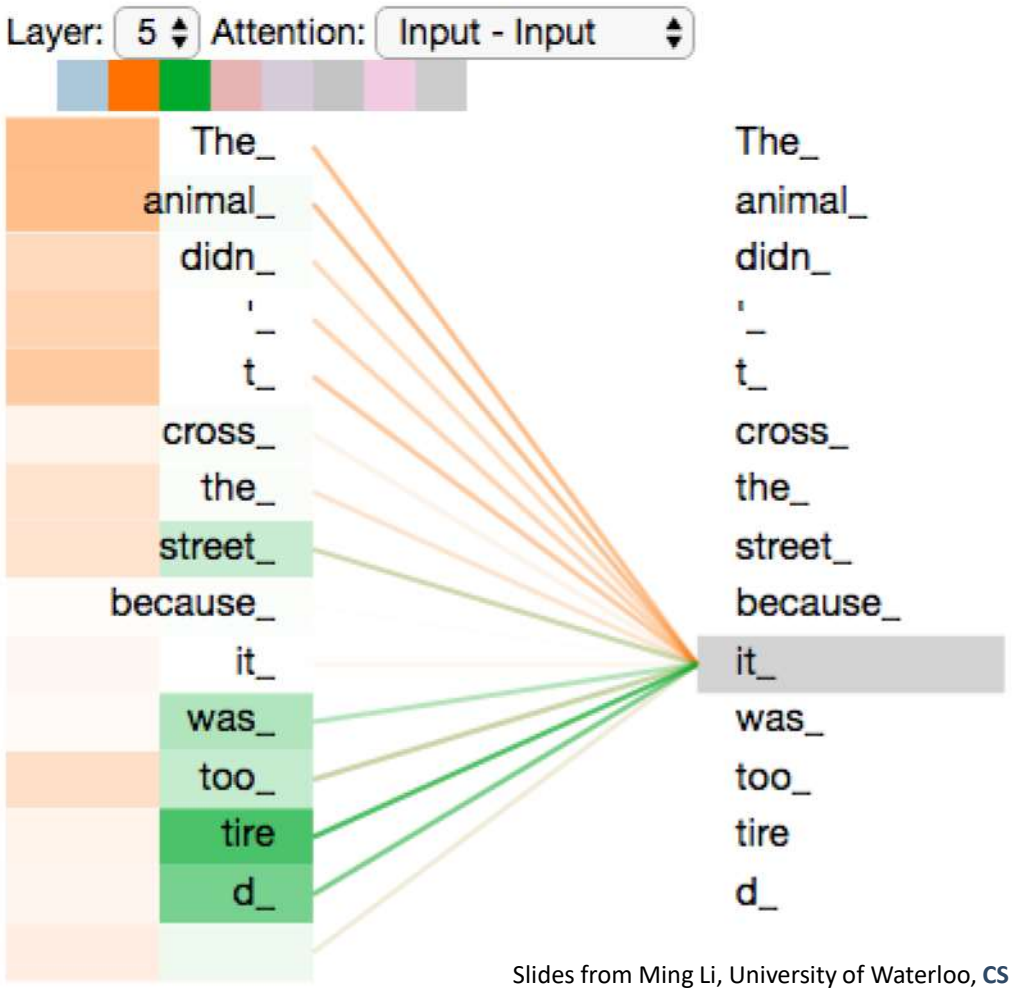
=



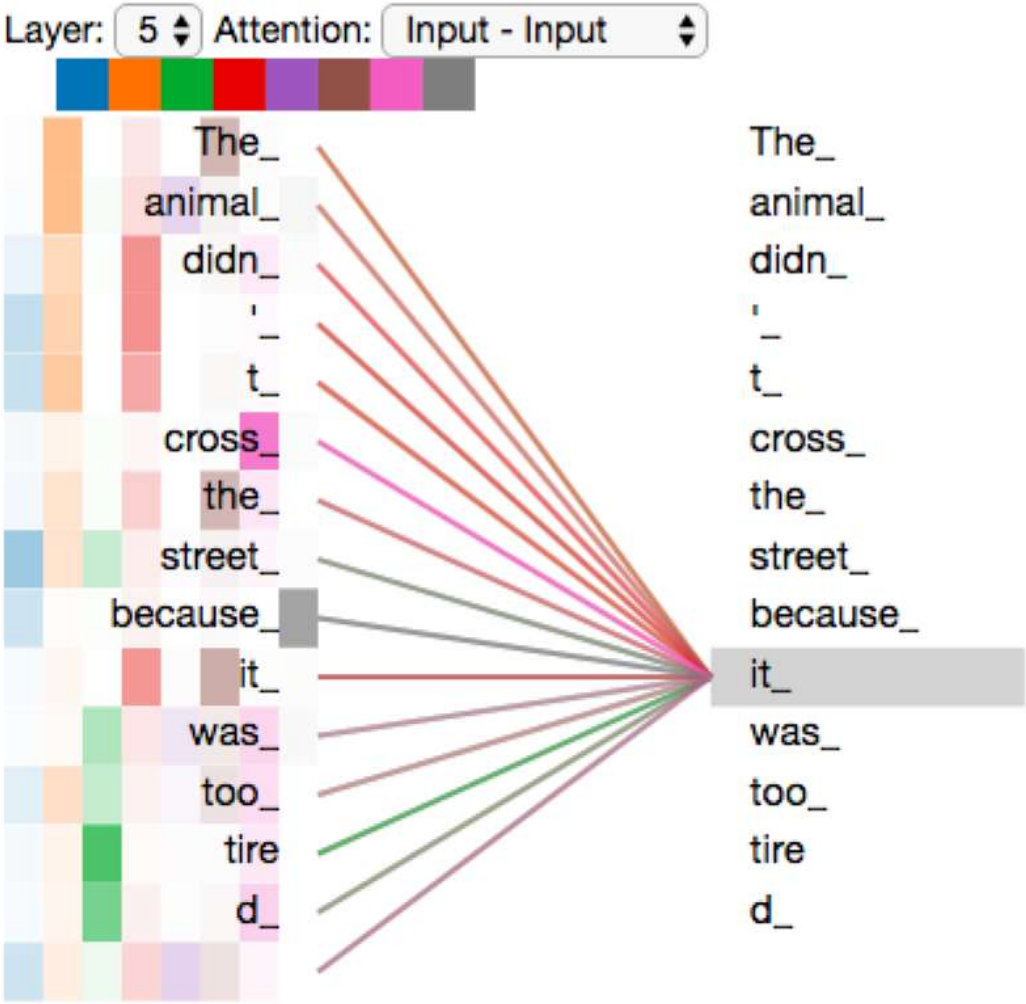
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**



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



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



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



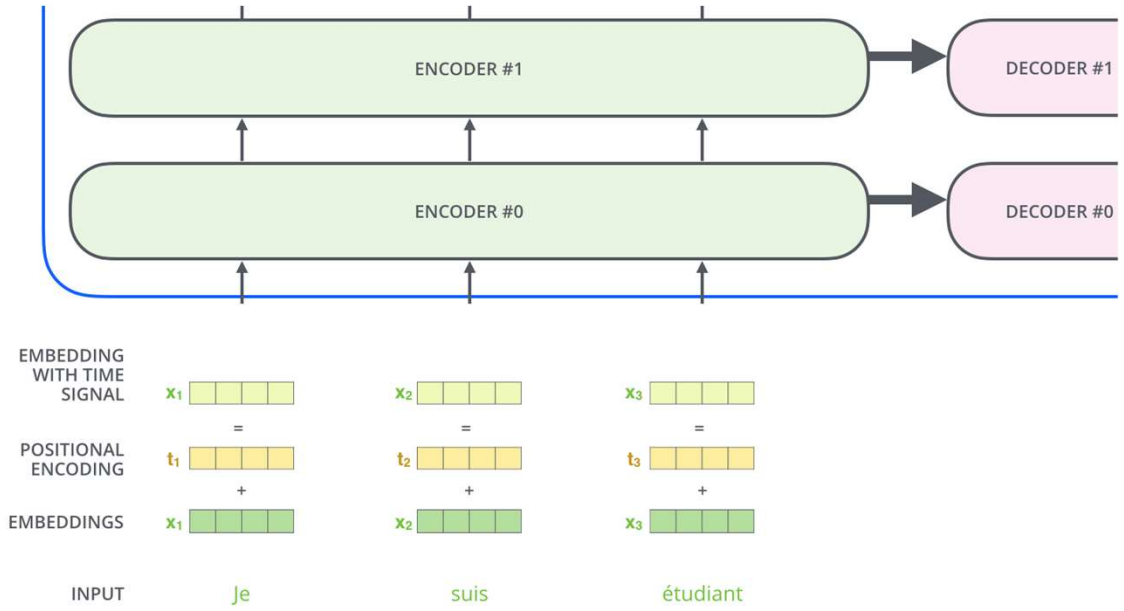
Representing the input order (positional encoding)

- Transformer is permutation invariant
- Transformer process tokens in parallel
- Attention mechanism does not take into account the position of a word in the sequence
- 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.

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



Representing the input order (positional encoding)



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

Position Encoding

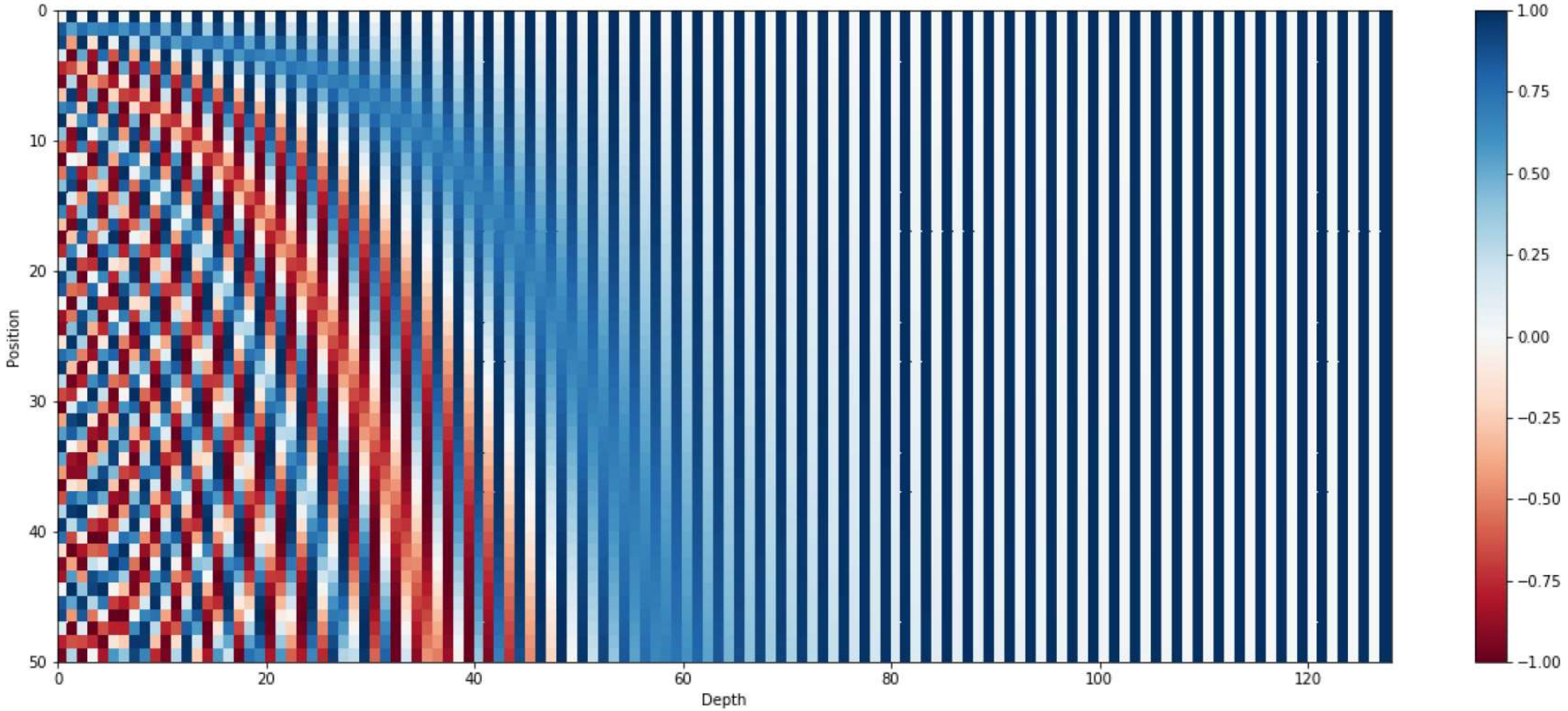
$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

where

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

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Position Encoding



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

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

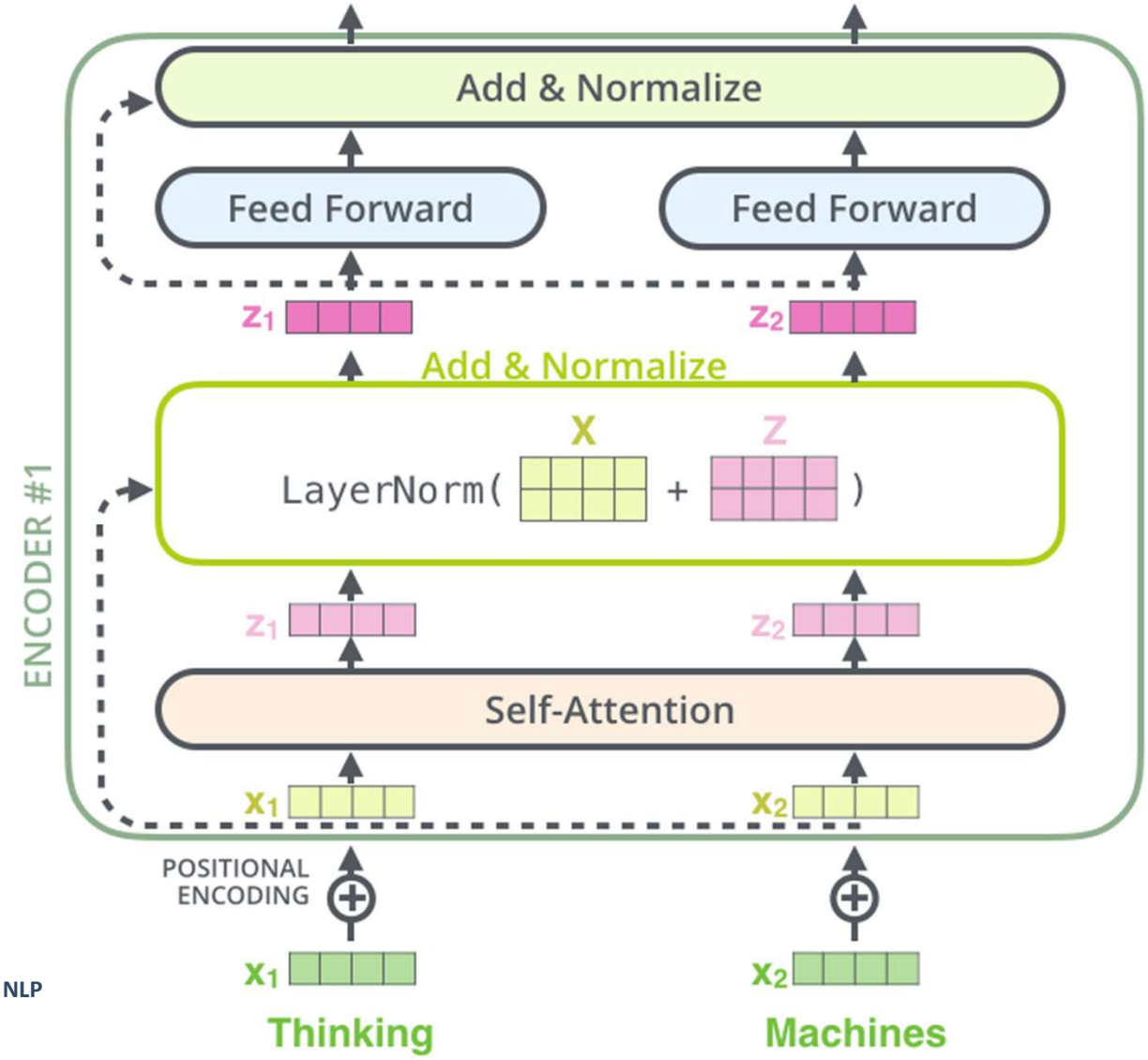
Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	$P_{00}=\sin(0)$ = 0	$P_{01}=\cos(0)$ = 1	$P_{02}=\sin(0)$ = 0	$P_{03}=\cos(0)$ = 1
am	1	$P_{10}=\sin(1/1)$ = 0.84	$P_{11}=\cos(1/1)$ = 0.54	$P_{12}=\sin(1/10)$ = 0.10	$P_{13}=\cos(1/10)$ = 1.0
a	2	$P_{20}=\sin(2/1)$ = 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



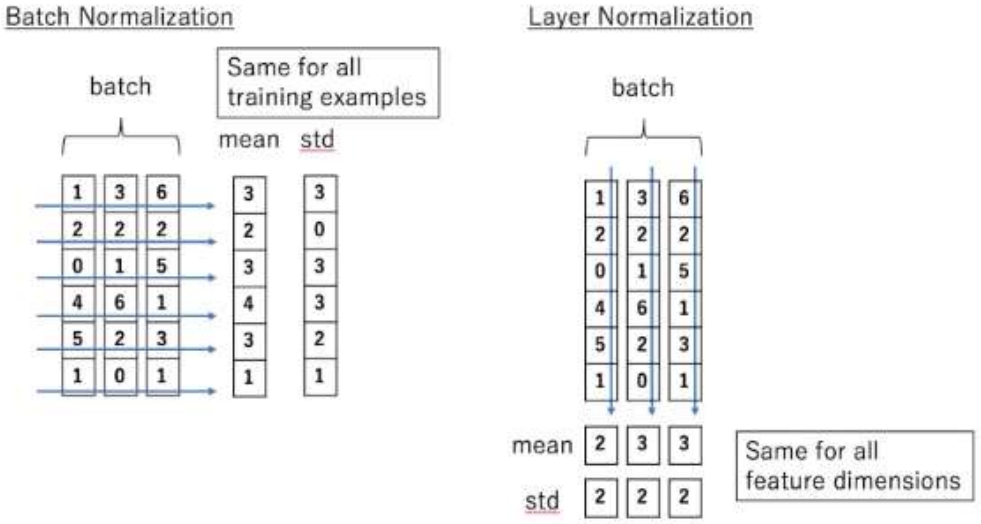
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

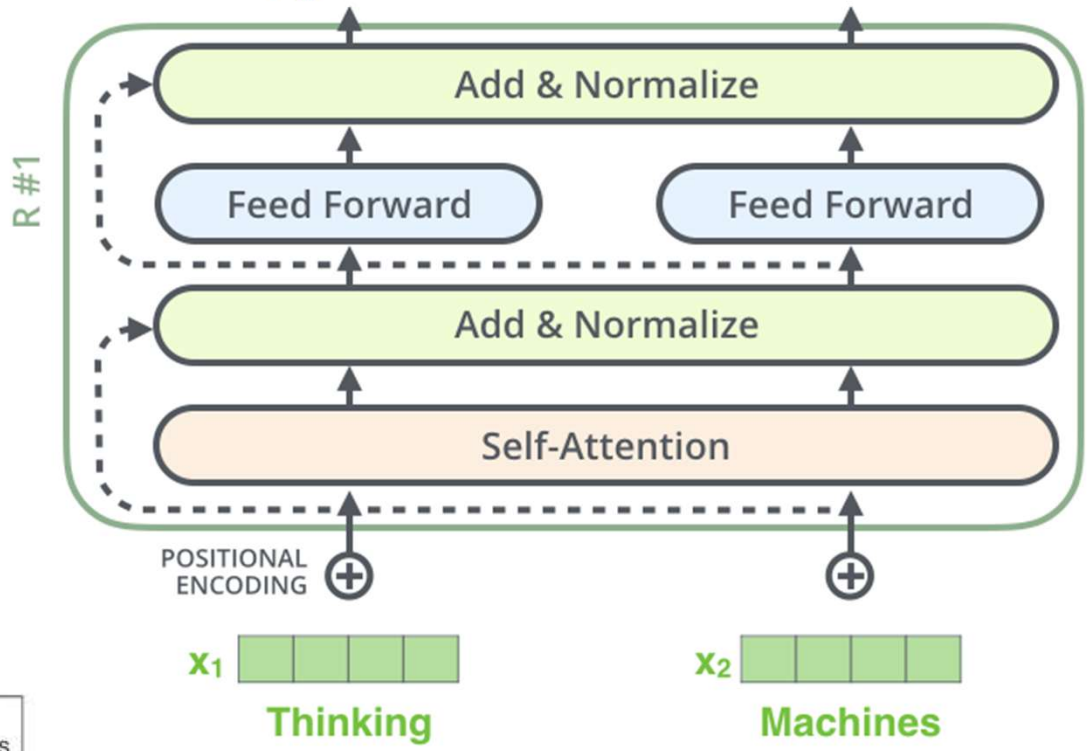
Layer Normalization (Hinton)

Layer normalization normalizes the inputs across the features.

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \sigma_{\mathcal{B}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 + \epsilon} \qquad (1)$$

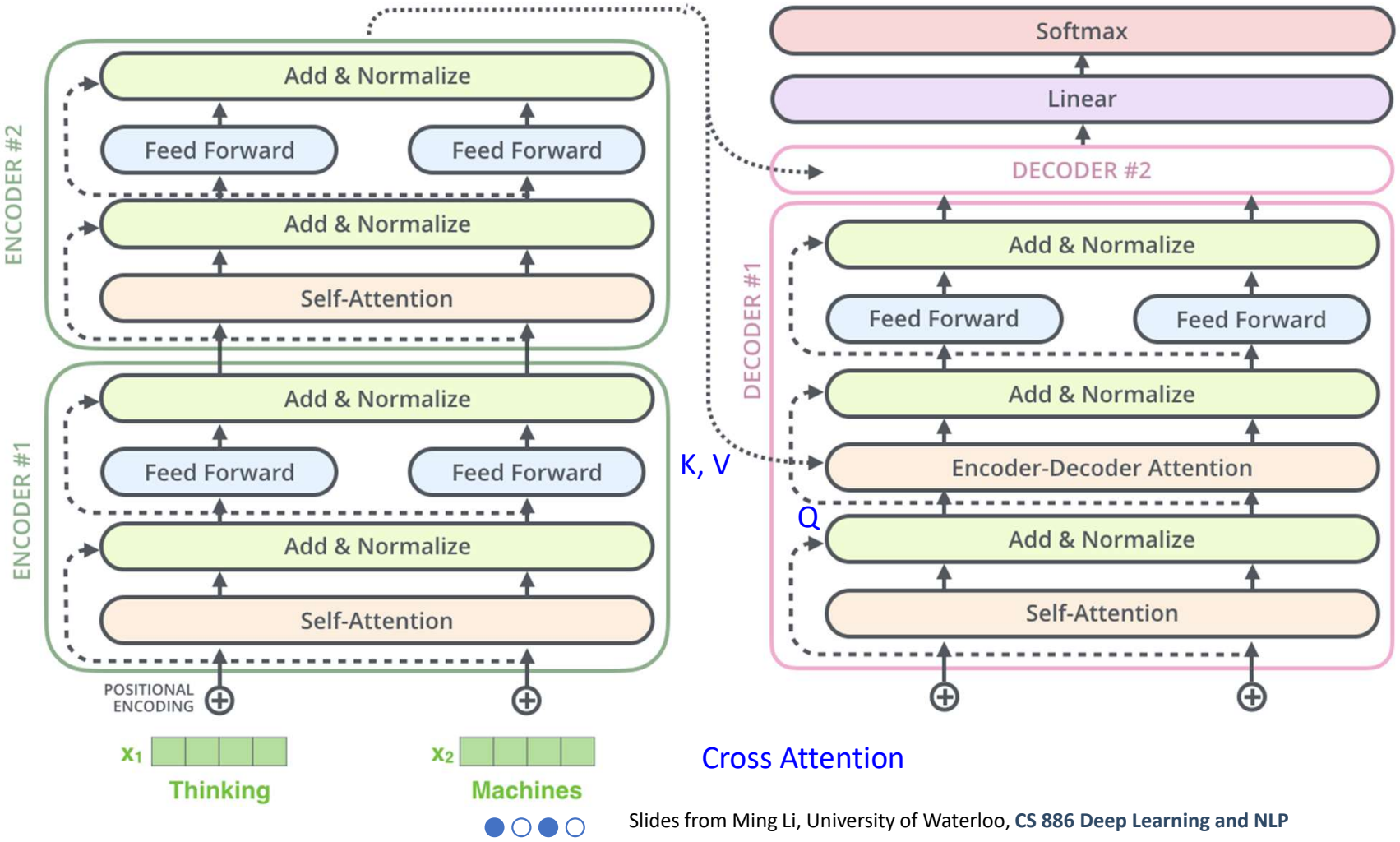
$$\hat{x}_i = \frac{1}{\sigma_{\mathcal{B}}}(x_i - \mu_{\mathcal{B}}) \quad y_i = \gamma\hat{x}_i + \beta \qquad (2)$$





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

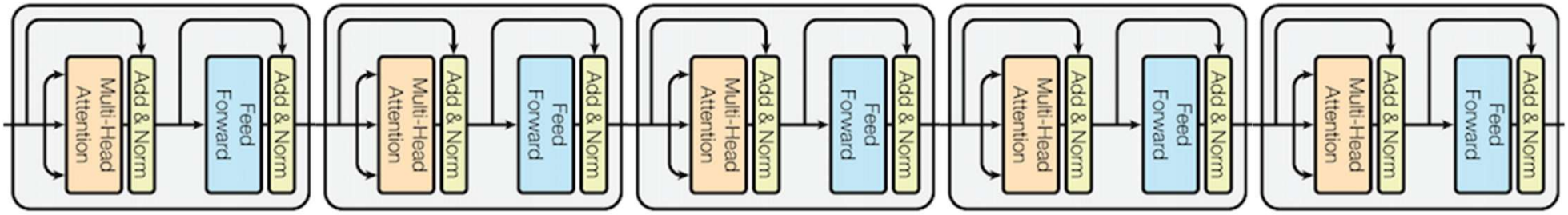
The complete transformer



Cross Attention

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

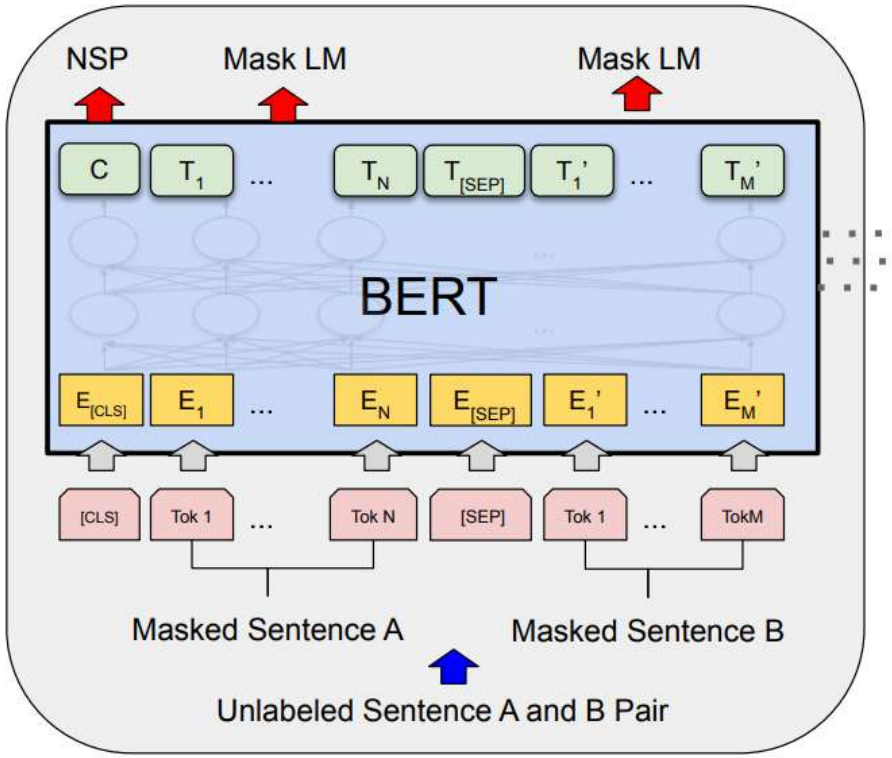
BERT (Stack Encoder Blocks)



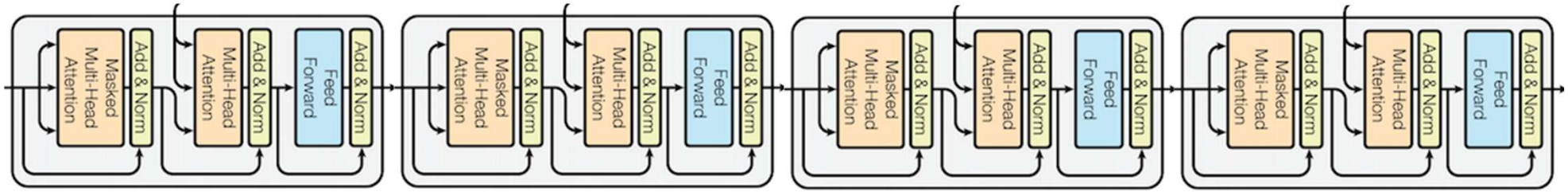
BERT (Bidirectional Encoder Representations from Transformers)

- 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.

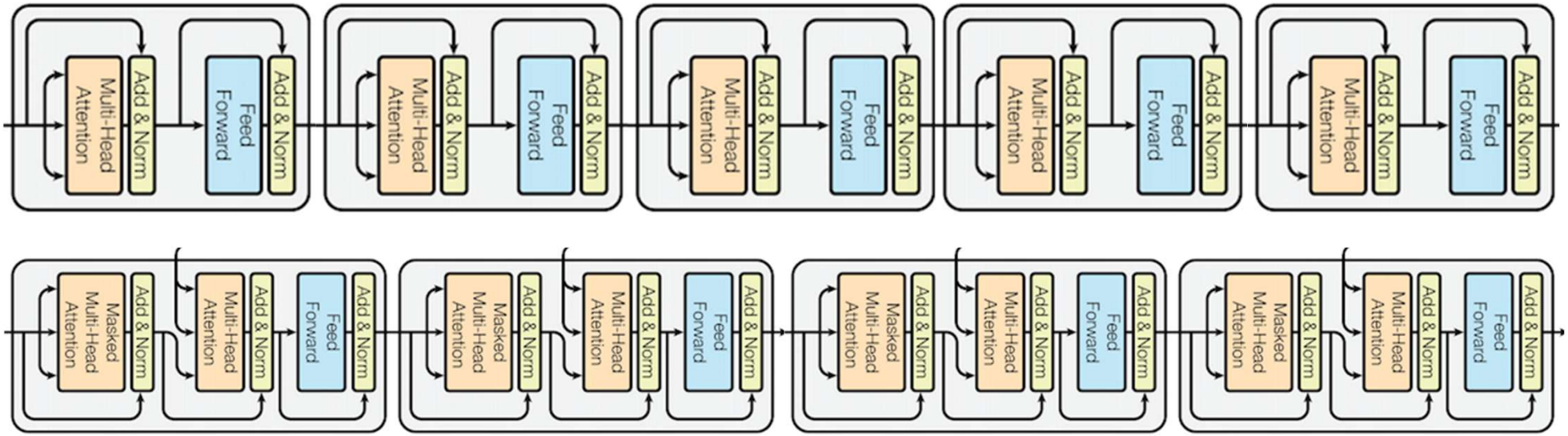
BERT


NSP
Mask LM
Mask LM
C
T1
...
TN
T[SEP]
T1'
...
TM'
BERT
E[CLS]
E1
...
EN
E[SEP]
E1'
...
EM'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair

GPT (Stack Decoder Blocks)



BERT Vs GPT


Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Masked Multi-Head Attention
Add & Norm
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm

BERT and GPT

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

Vision Transformers

Mubarak Shah

shah@crcv.ucf.edu

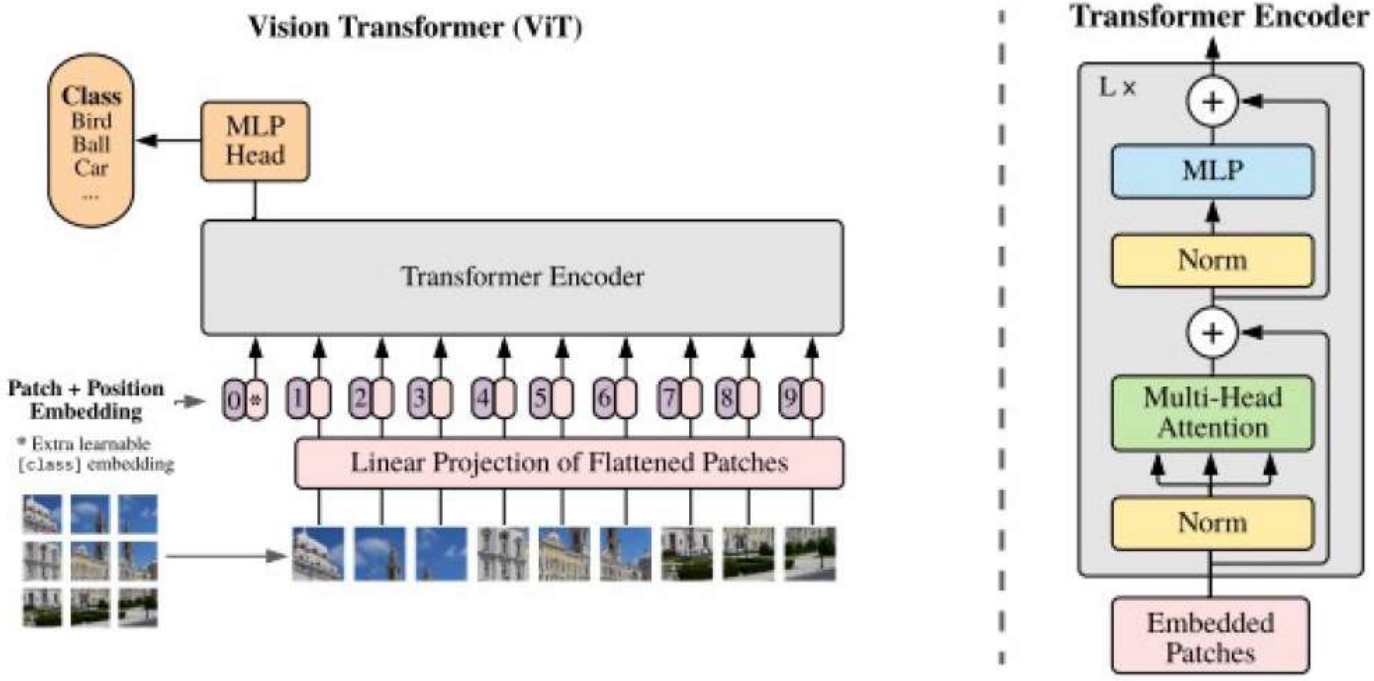
Vision Transformer (VIT)

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; 50,577 citations!)

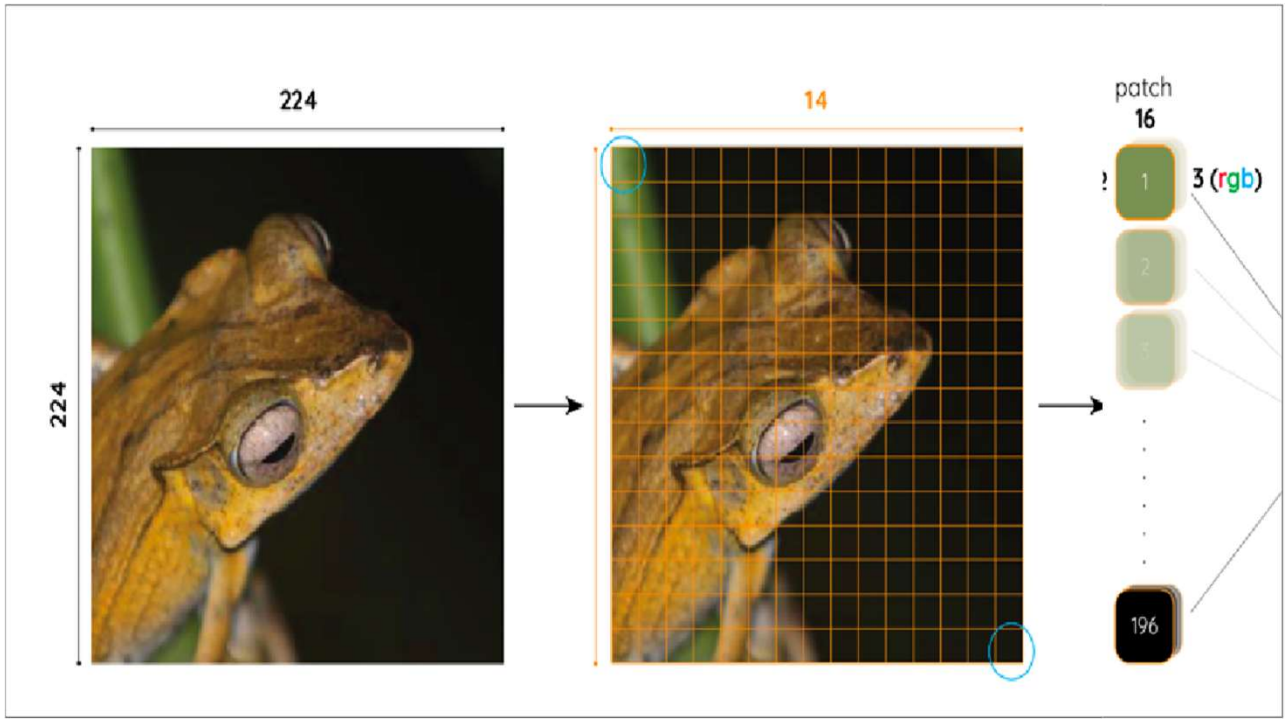
Vision Transformer (VIT)

- 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)

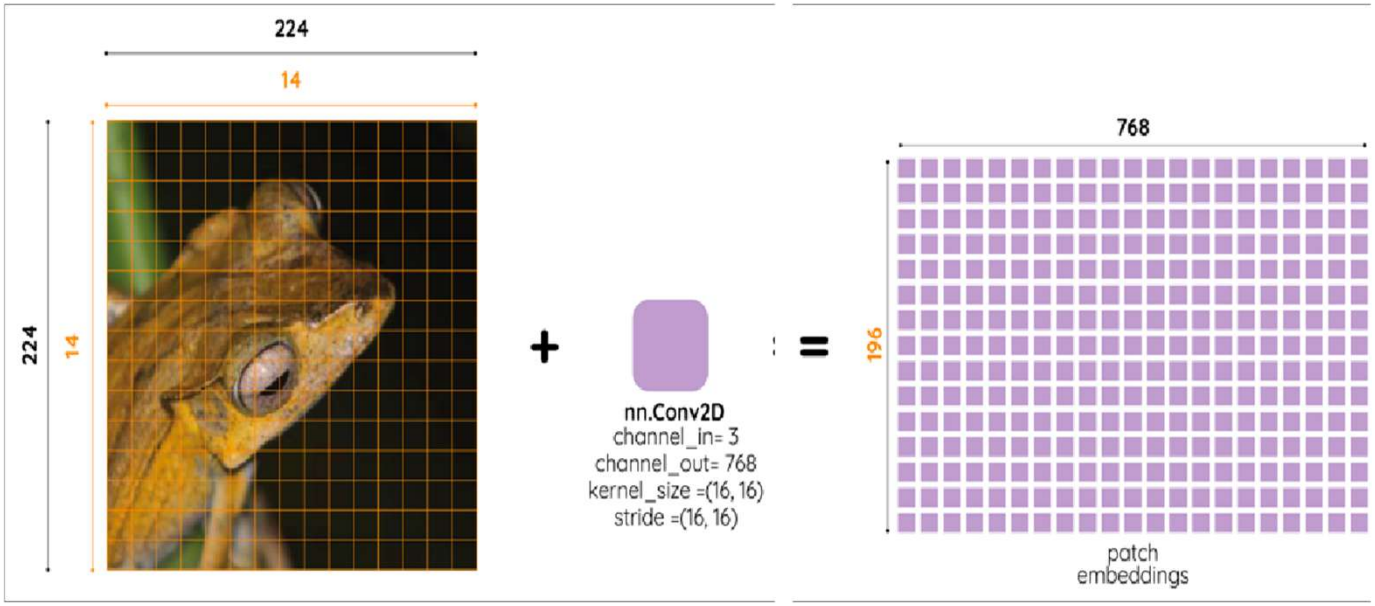


Divide image into 16x16 patches



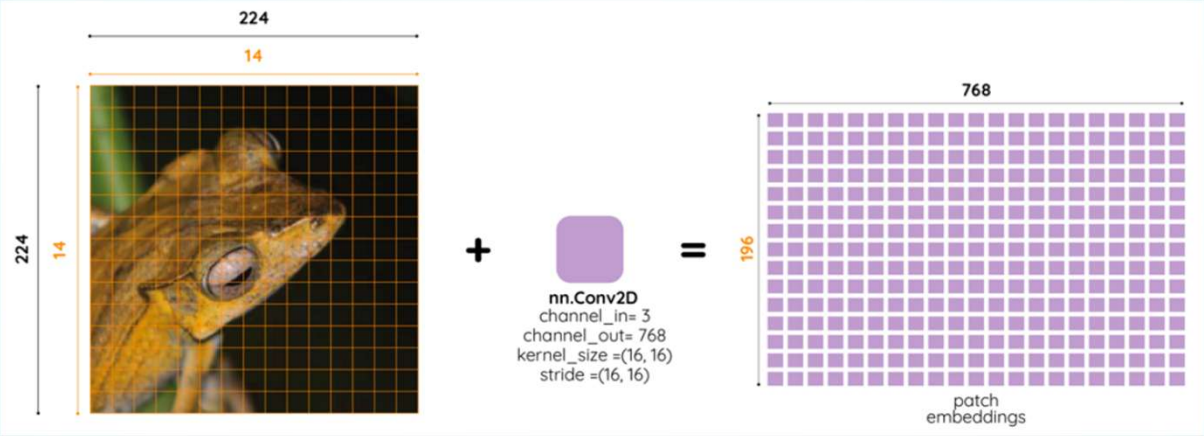
Slide credit: Piotr Mazurek

Generate embedding for each patch



Slide credit: Piotr Mazurek

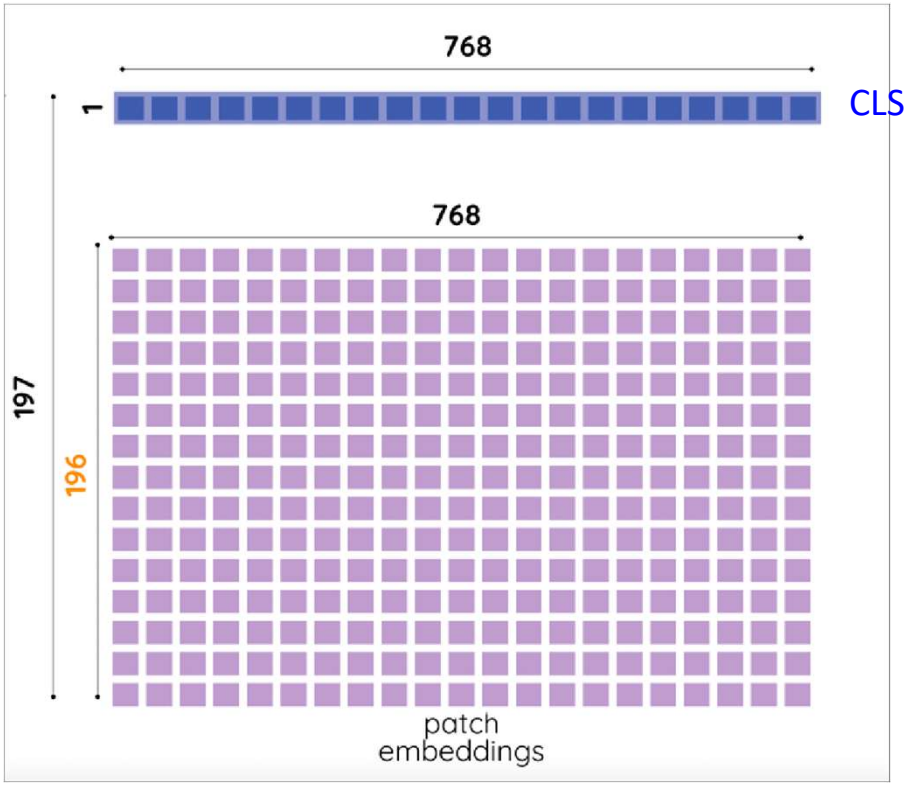
PATCH EMBEDDINGS



Source: Committed towards better future
Bukhari 2021

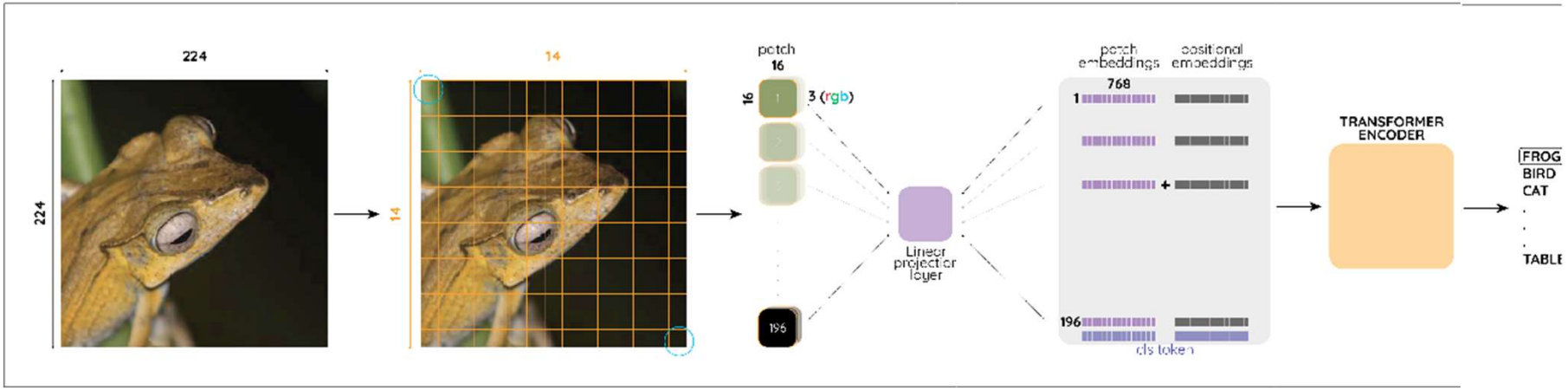
```
x = torch.randn(1, 3, 224, 224)
# 2D conv
conv = nn.Conv2d(3, 768, 16, 16)
conv(x).reshape(-1, 196).transpose(0,1).shape
>> torch.Size([196, 768])
```

CLS (Classification) Token



Slide credit: Piotr Mazurek

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

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; 50,577 citations!)

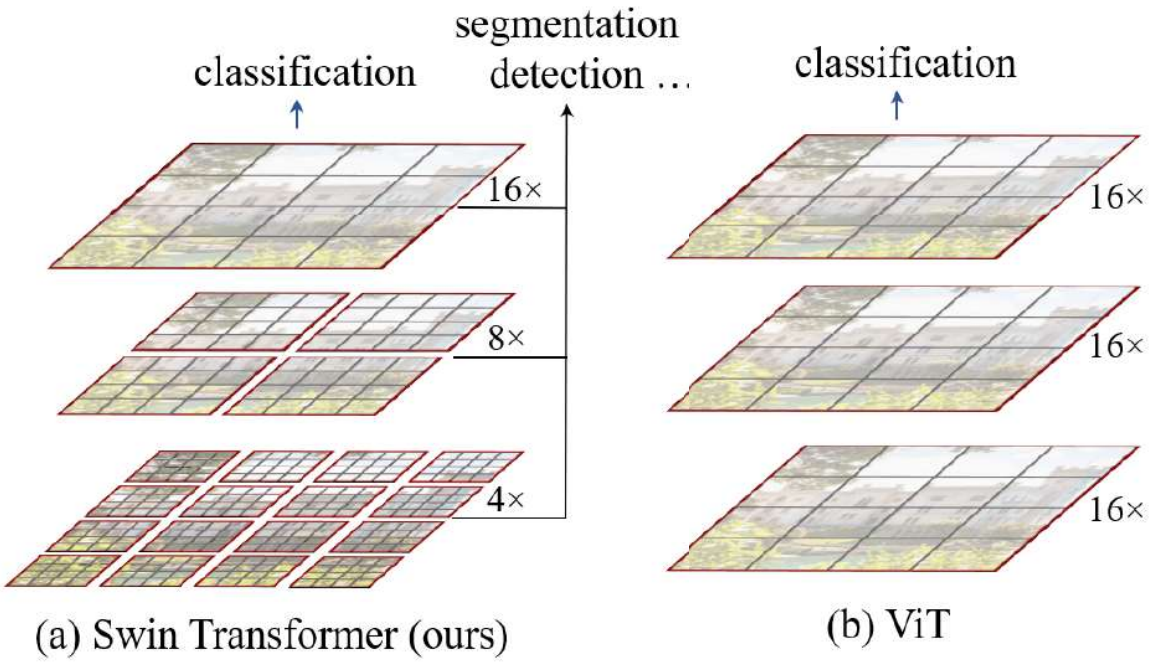
SWIN

- 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) 25,675 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

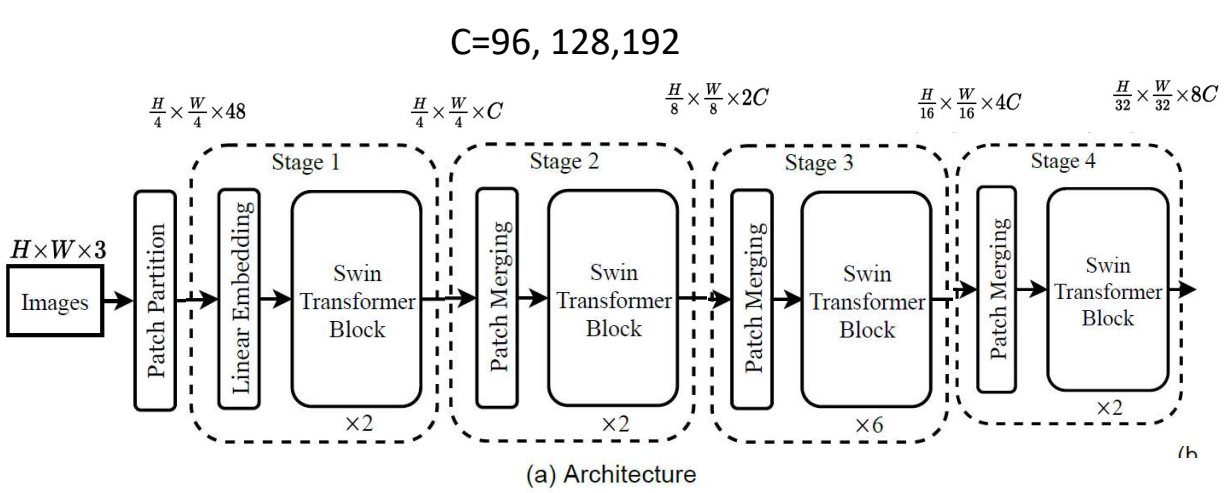
Hierarchical Feature Maps and Local Attention

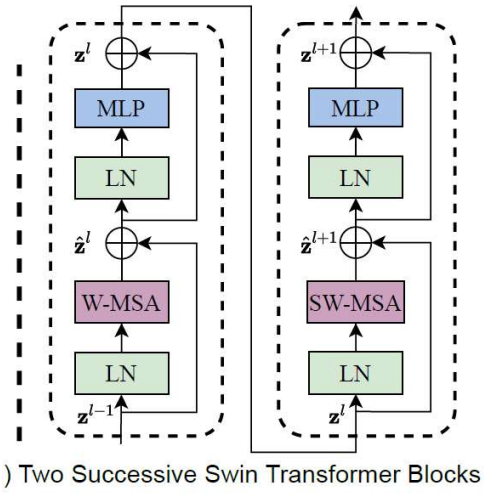


SWIN

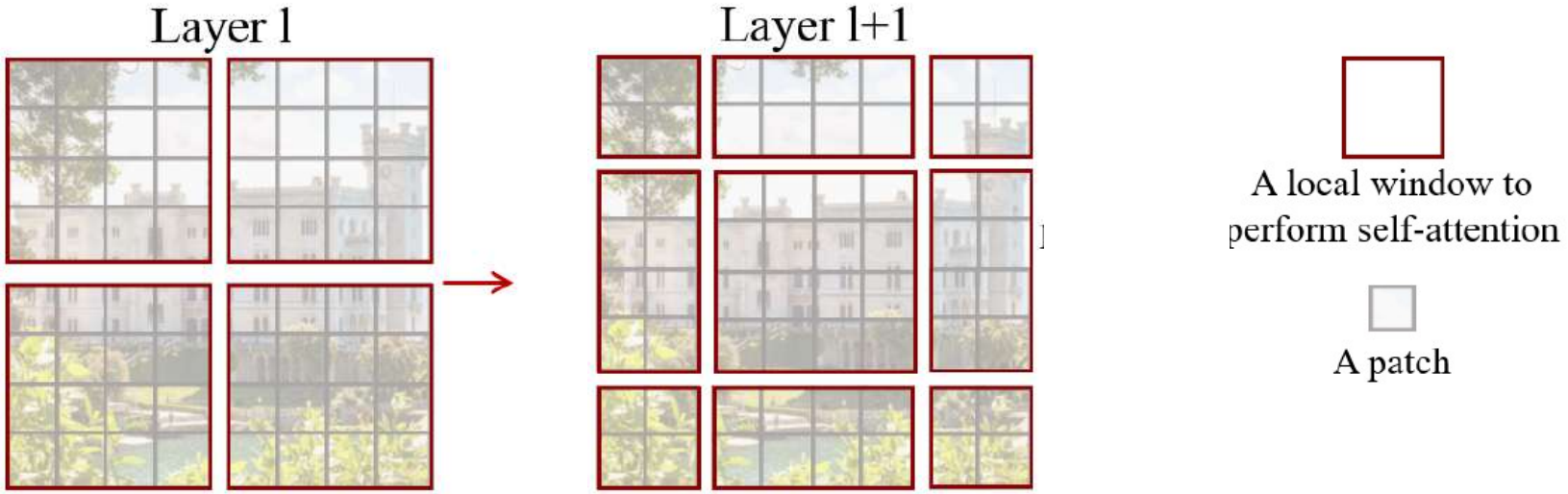
- 87.3 top-1 accuracy on ImageNet-1K
- Dense prediction tasks such as
  - Object detection (58.7 box AP and 51.1 mask AP on COCO)
  - Semantic segmentation (53.5 mIoU on ADE20K)
- Performance surpasses the previous state-of-the art by
  - +2.7 box AP and +2.6 mask AP on COCO, and
  - +3.2 mIoU on ADE20K,

SWIN Architecture





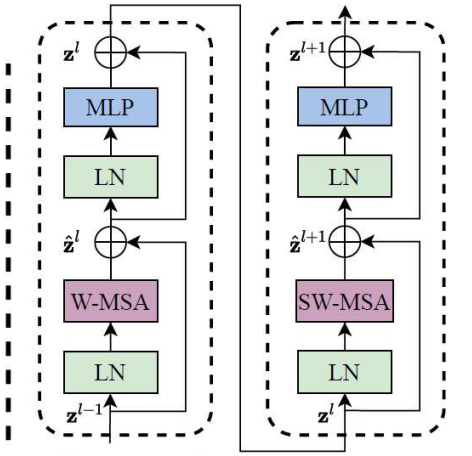
Self-Attention within each window and shifted windows



Transformer Blocks

$$\hat{\mathbf{z}}^{l} = \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1},$$
$$\mathbf{z}^{l} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l}\right)\right) + \hat{\mathbf{z}}^{l},$$

$$\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^{l}\right)\right) + \mathbf{z}^{l},$$
$$\mathbf{z}^{l+1} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},$$



) Two Successive Swin Transformer Blocks

Different Configurations

- Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers =$\{2, 2, 18, 2\}$

Results

(a) Regular ImageNet-1K trained models					
method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

Results

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [34]	384^2	388M	204.6G	-	84.4
R-152x4 [34]	480^2	937M	840.5G	-	85.4
ViT-B/16 [19]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [19]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Results

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

ADE20K Method	Backbone	val mIoU	test score	#param.	FLOPs	FPS
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
DNL [65]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [67]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [63]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [67]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [73]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val

Summary

- VIT is first Vision Transformer, but trained on huge dataset of 300M

- SWIN employs window attention
- Performs well on other tasks: object detection, semantic segmentation