UCF Center for Research in Computer Vision

CAP6412
Advanced Computer Vision

Mubarak Shah

shah@crcv.ucf.edu

Lecture-3-Transformer Introduction

- What is Transformer?
- Self-Attention
  - Query, Key, Value
- Position encoding
- Encoder-Decoder

Transformer

- Used for modeling long dependencies between input sequence elements
- Supports parallel processing of sequence as compared to recurrent networks e.g. (LSTM).
- Requires minimal inductive biases for its design
- 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 capacity networks and huge datasets.

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

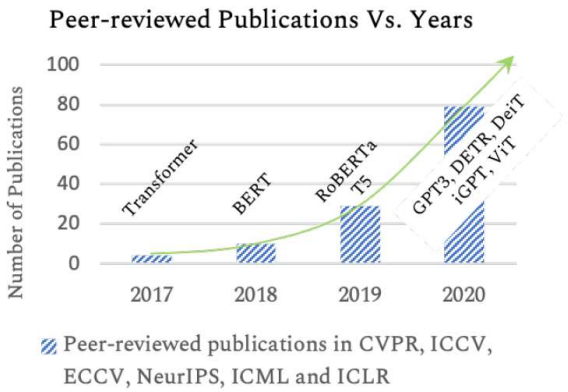
- BERT (Bidirectional Encoder Representations from Transformers),
- GPT (Generative Pre-trained Transformer) v1-3,
- RoBERTa (Robustly Optimized BERT Pre-training)
- T5 (Text-to-Text Transfer Transformer)

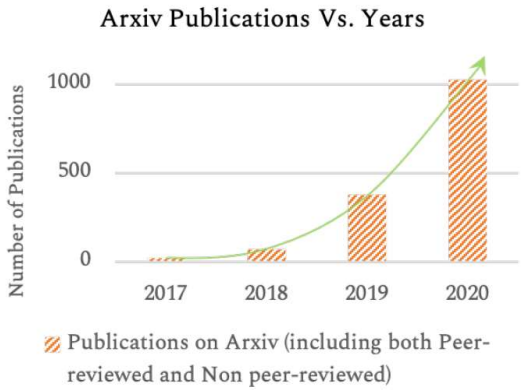
BERT (Bidirectional Encoder Representations from Transformers)

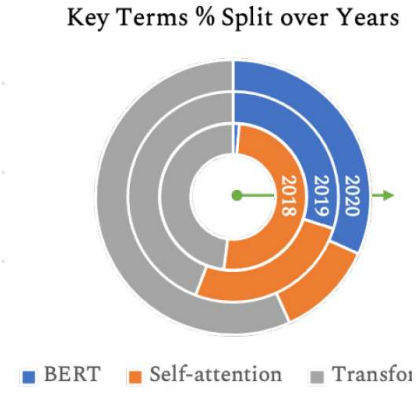
- Original Transformer model could only attend to the context on the left of a given word
- BERT jointly encodes the right and left context of a word in a sentence to improve the learned feature representations
- BERT is trained on two pre-text tasks in self-supervised manner
  - Masked Language Model (MLM)
    - A fixed percentage (15%) of words in a sentence are randomly masked and the model is trained to predict these masked words using cross-entropy loss.
    - In predicting the masked words, the model learns to incorporate 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.

Publications

The breakthroughs from Transformer networks in NLP domain has sparked great interest in the computer vision community to adapt these models



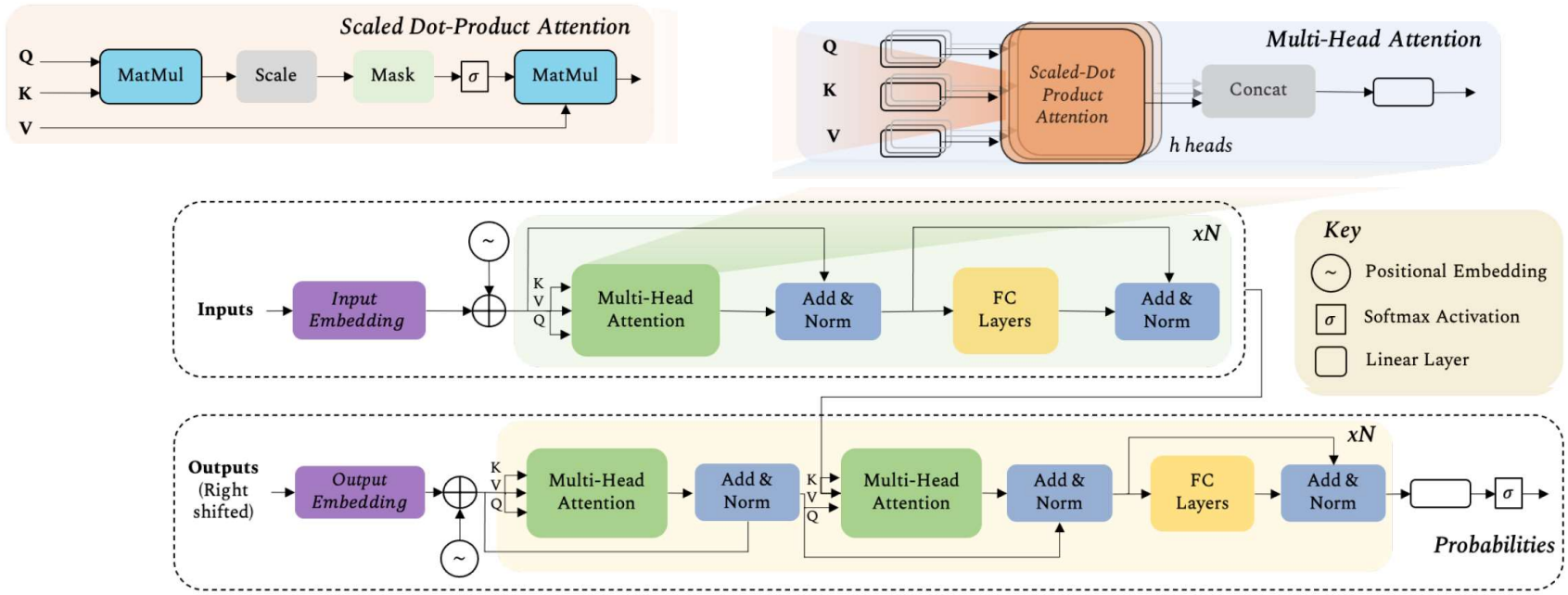


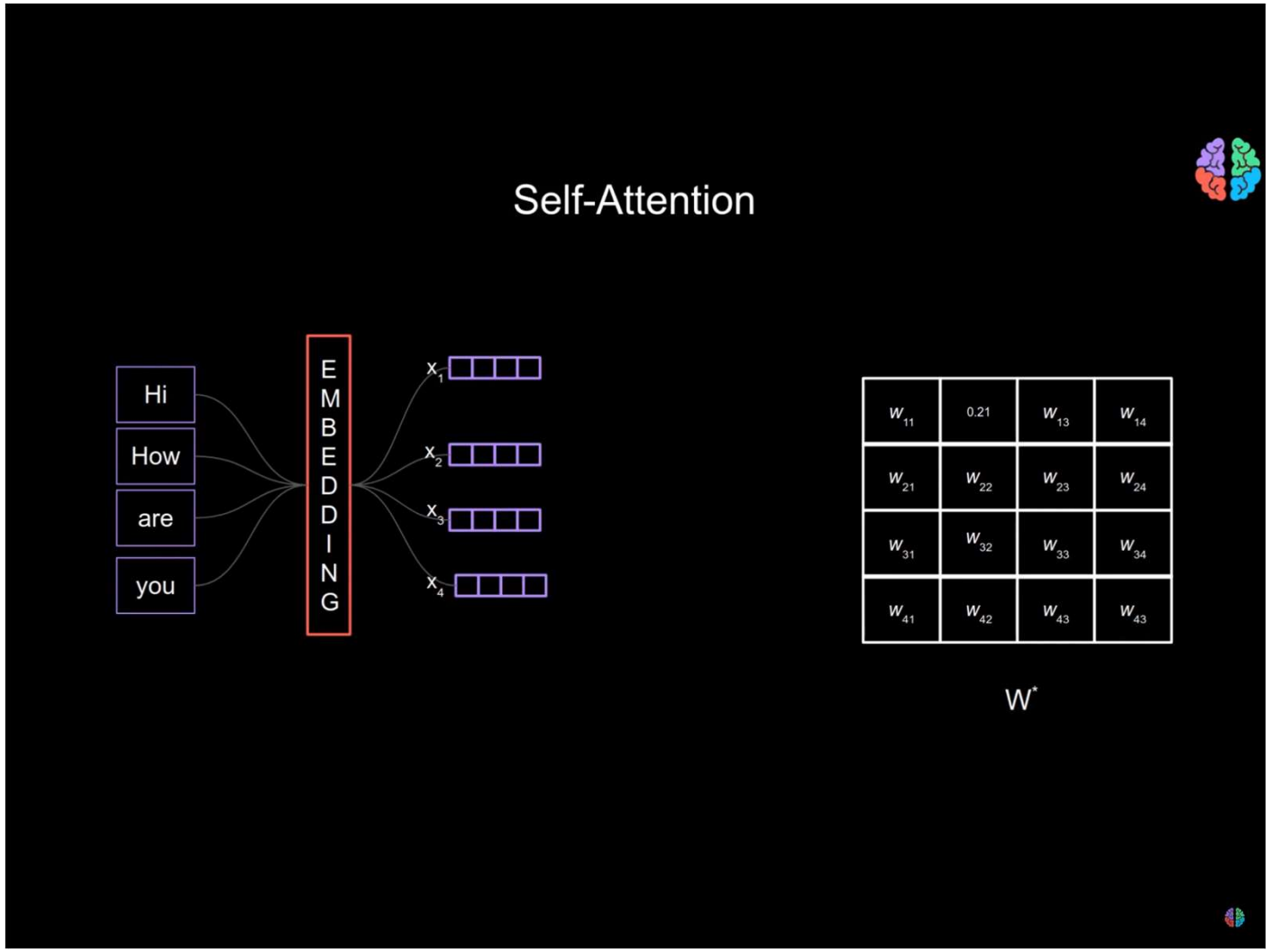


Transformer

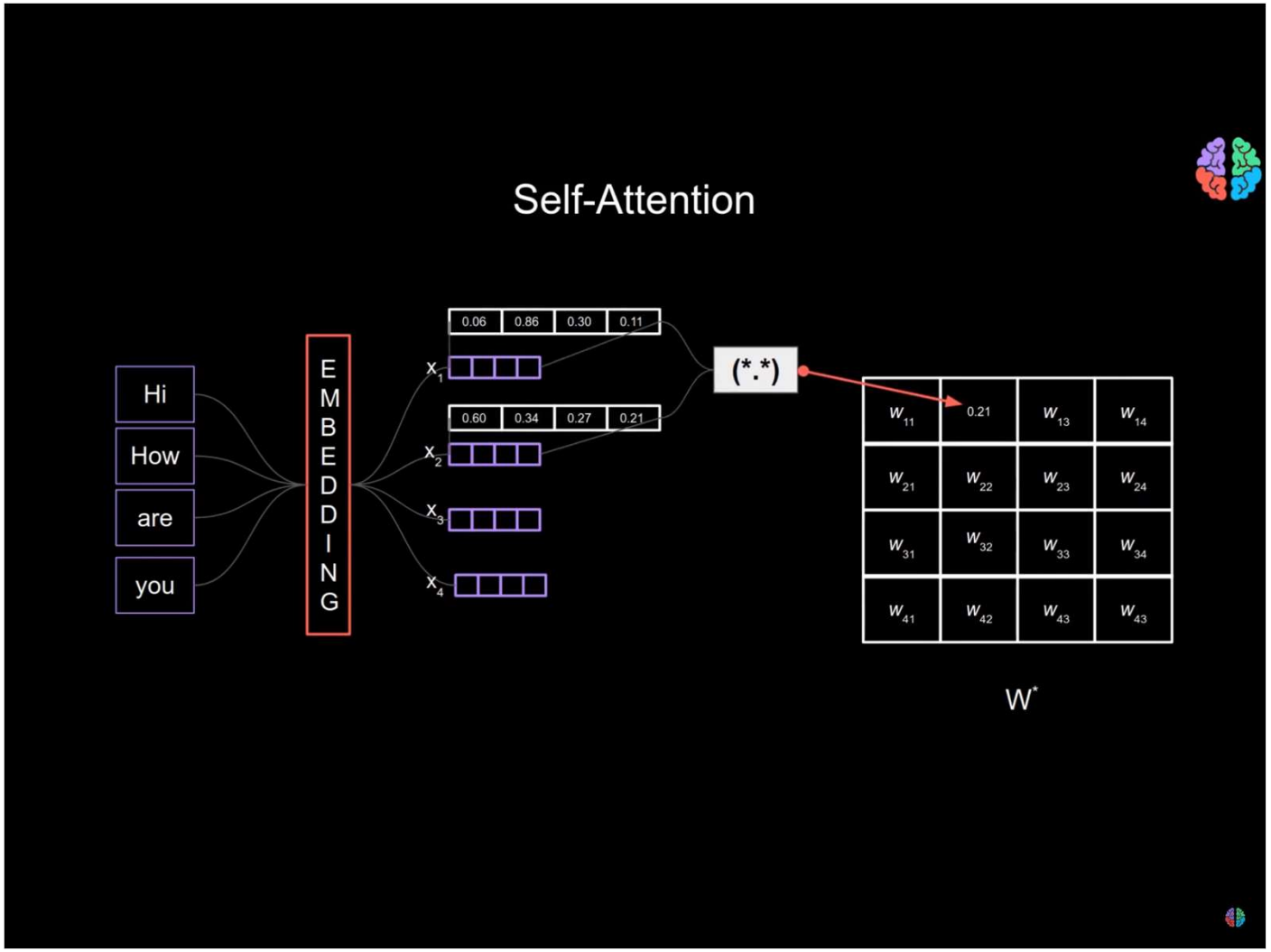
- It consists of Encoder and Decoder Blocks
- Main components of each block:
  - Self-Attention
  - Layer Normalization
  - Feed Forward Network
- The main difference of self-attention with CNN is that the filters are dynamically calculated

Transformer

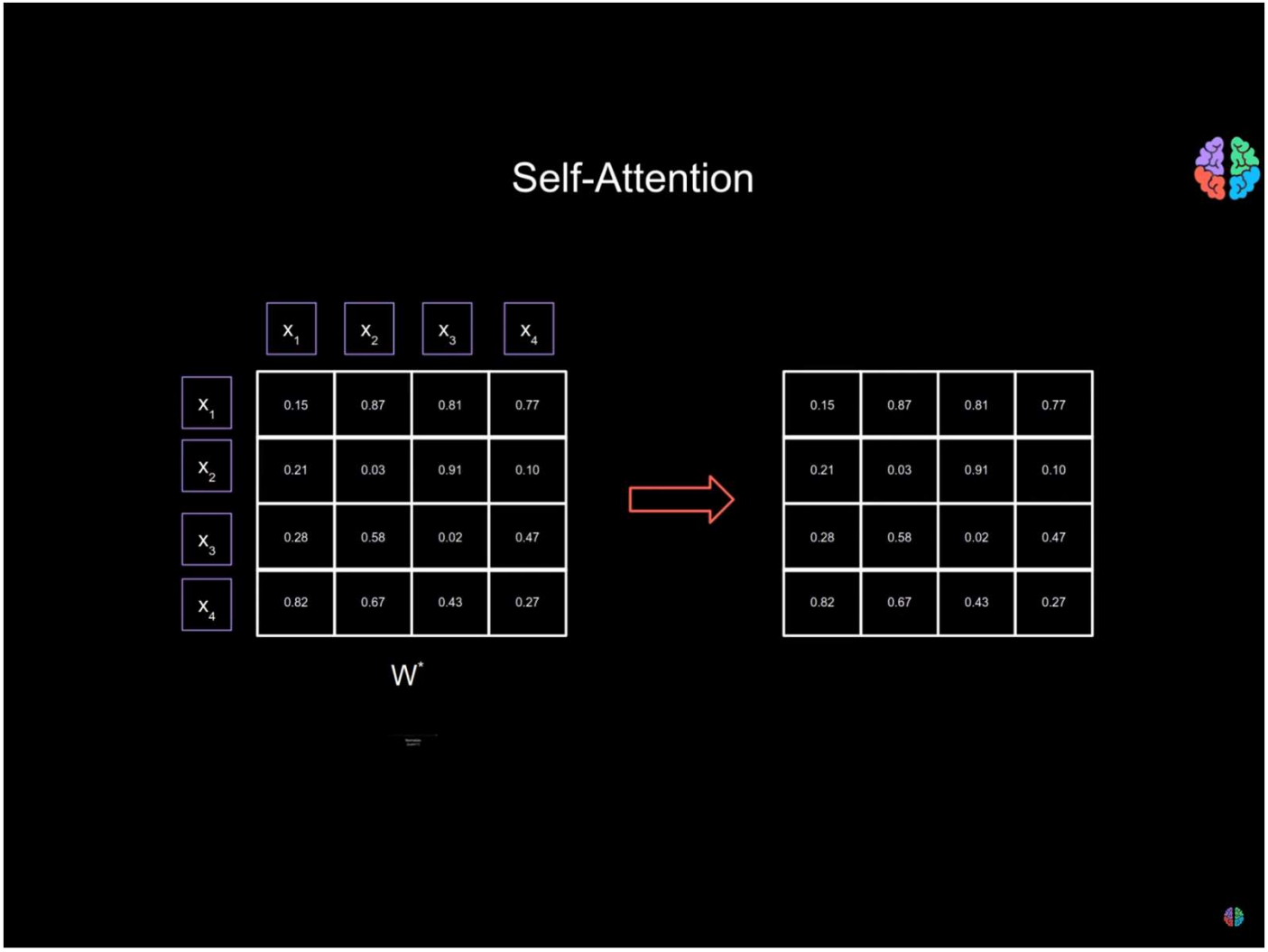




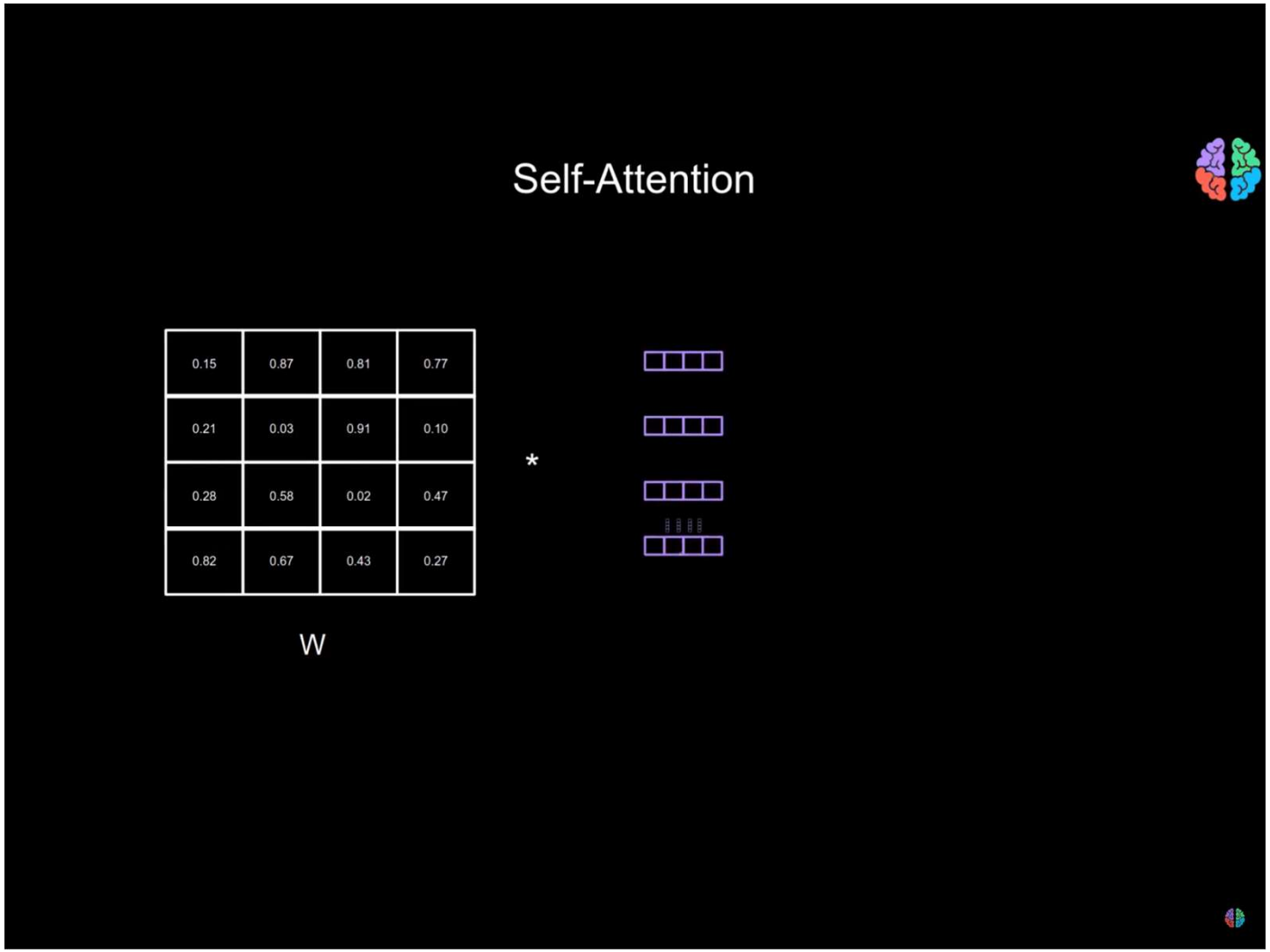
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



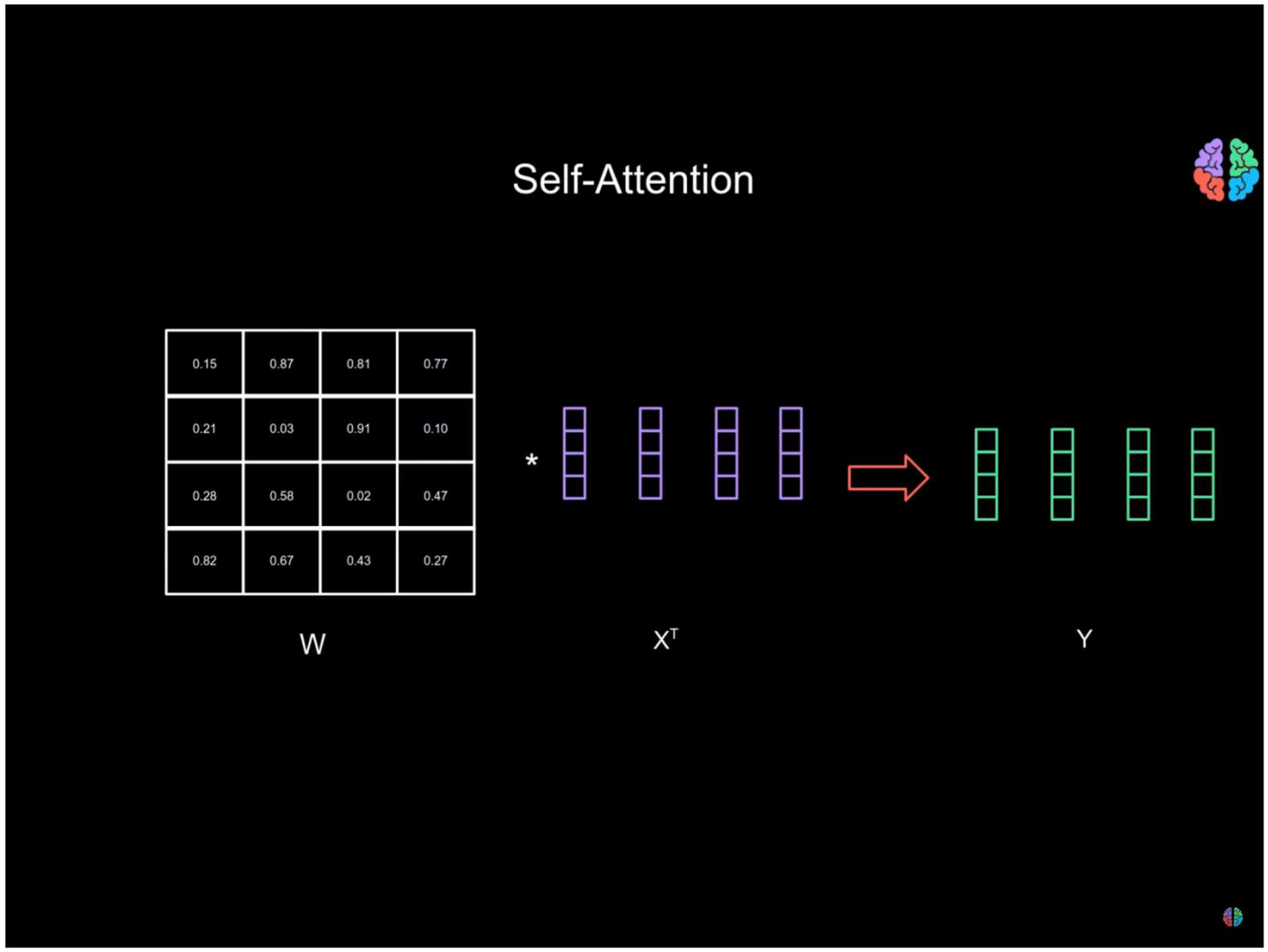
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



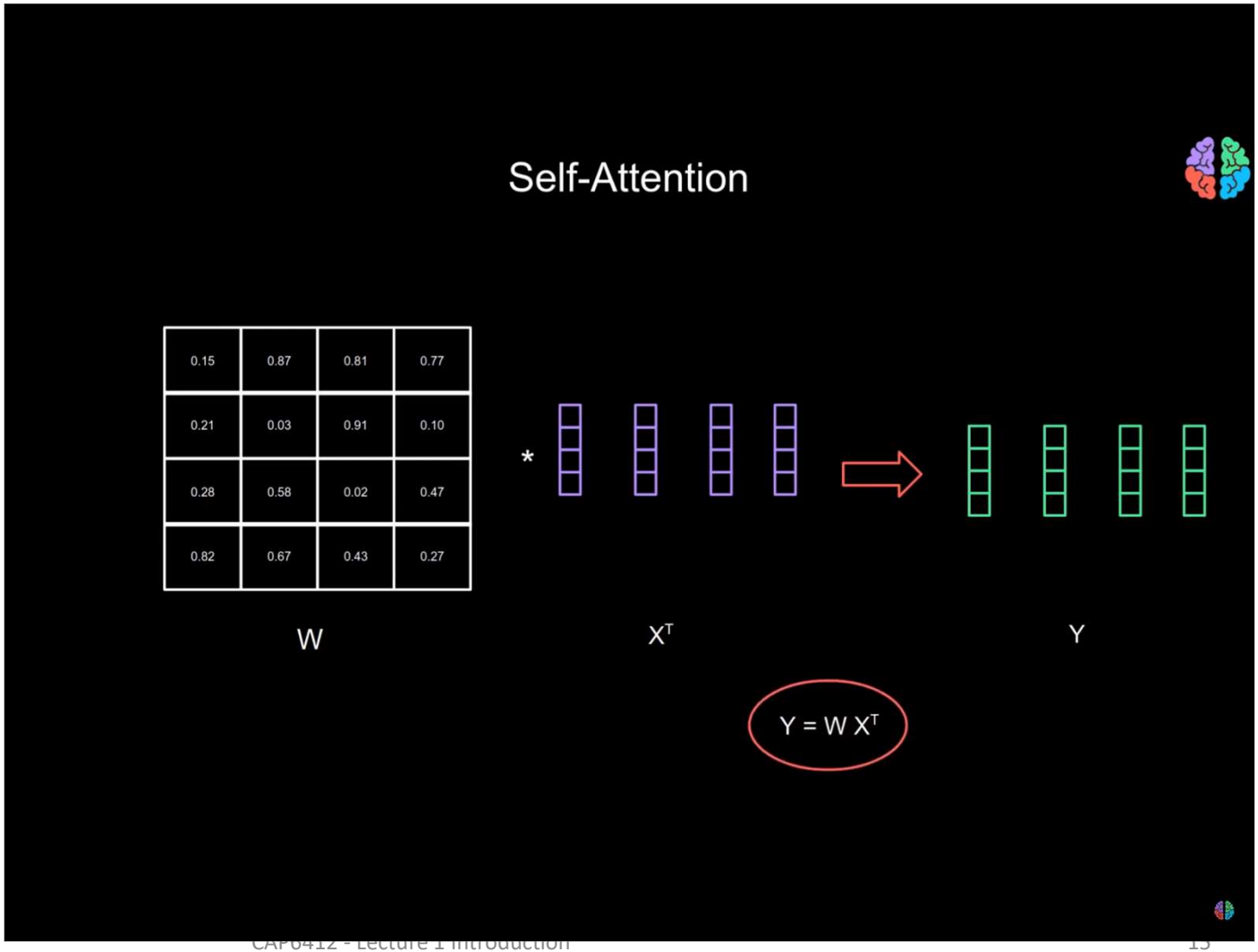
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



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



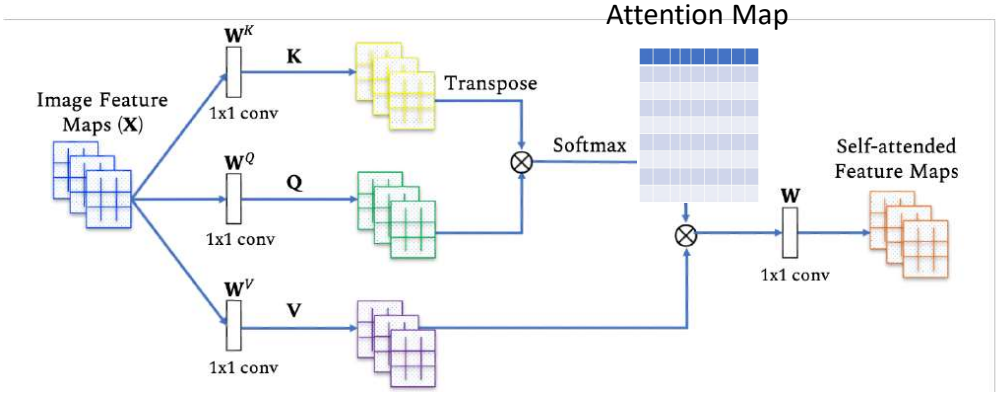
Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites

Self-Attention

Lets denote a sequence of n entities $(\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n)$ by $\mathbf{X} \in \mathbb{R}^{n \times d}$,

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

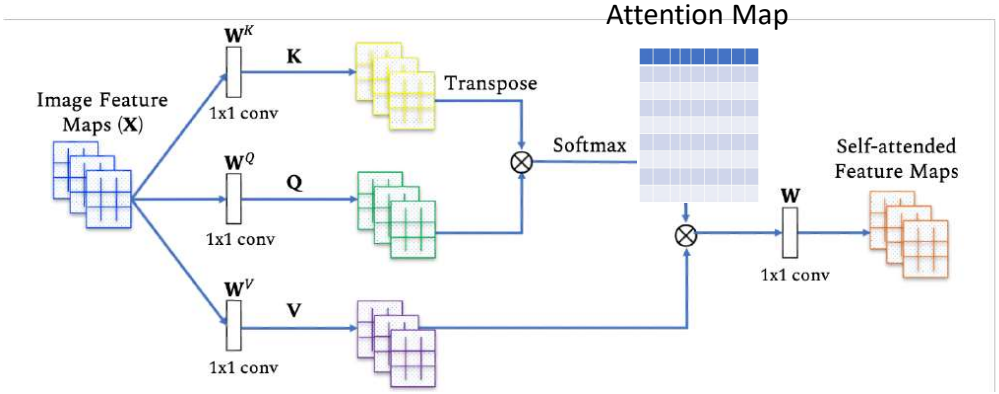
3x3 3x3 3x3

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

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

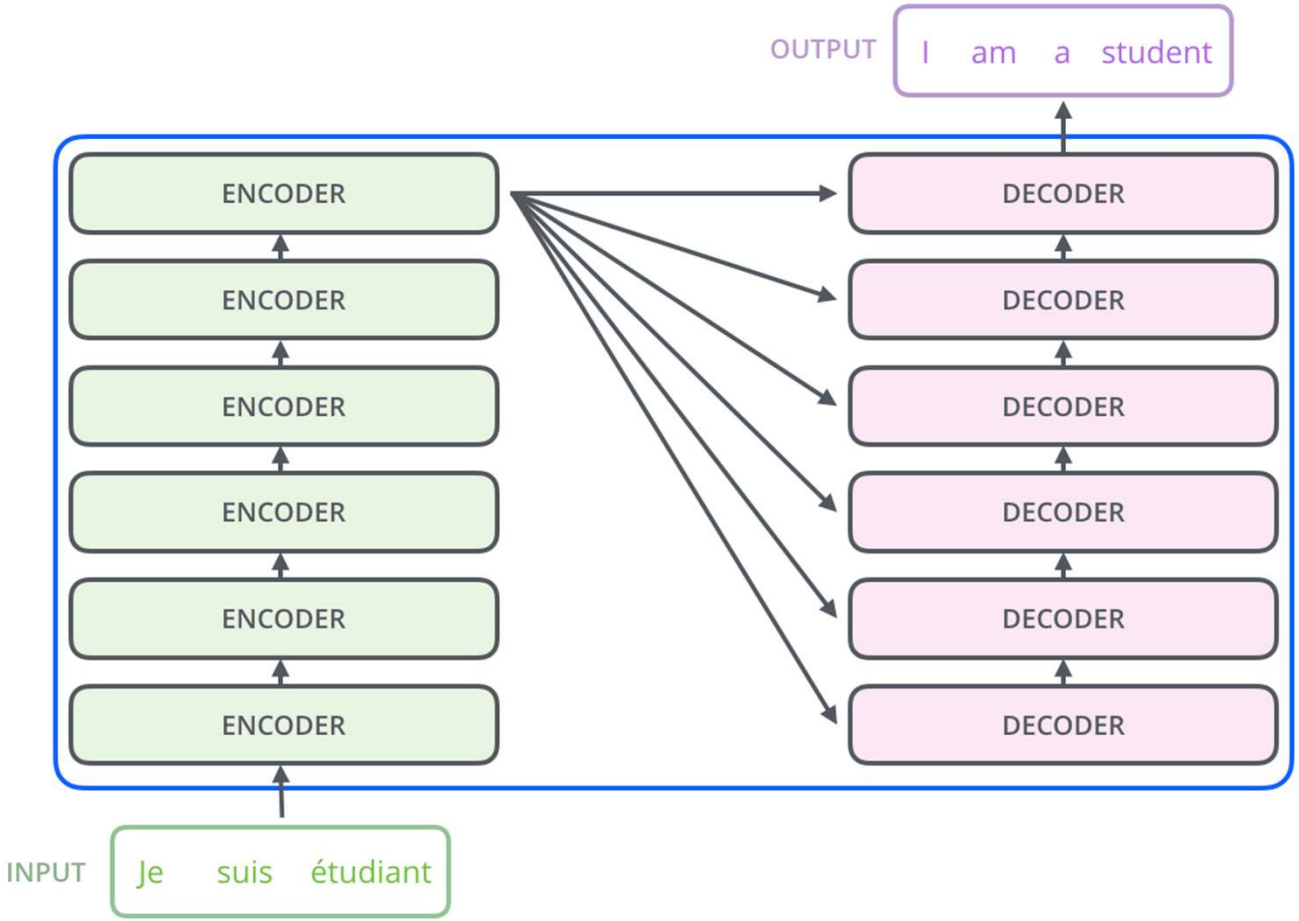
9x3 9x9 9x3



Transformers (Attention is all you need 2017)

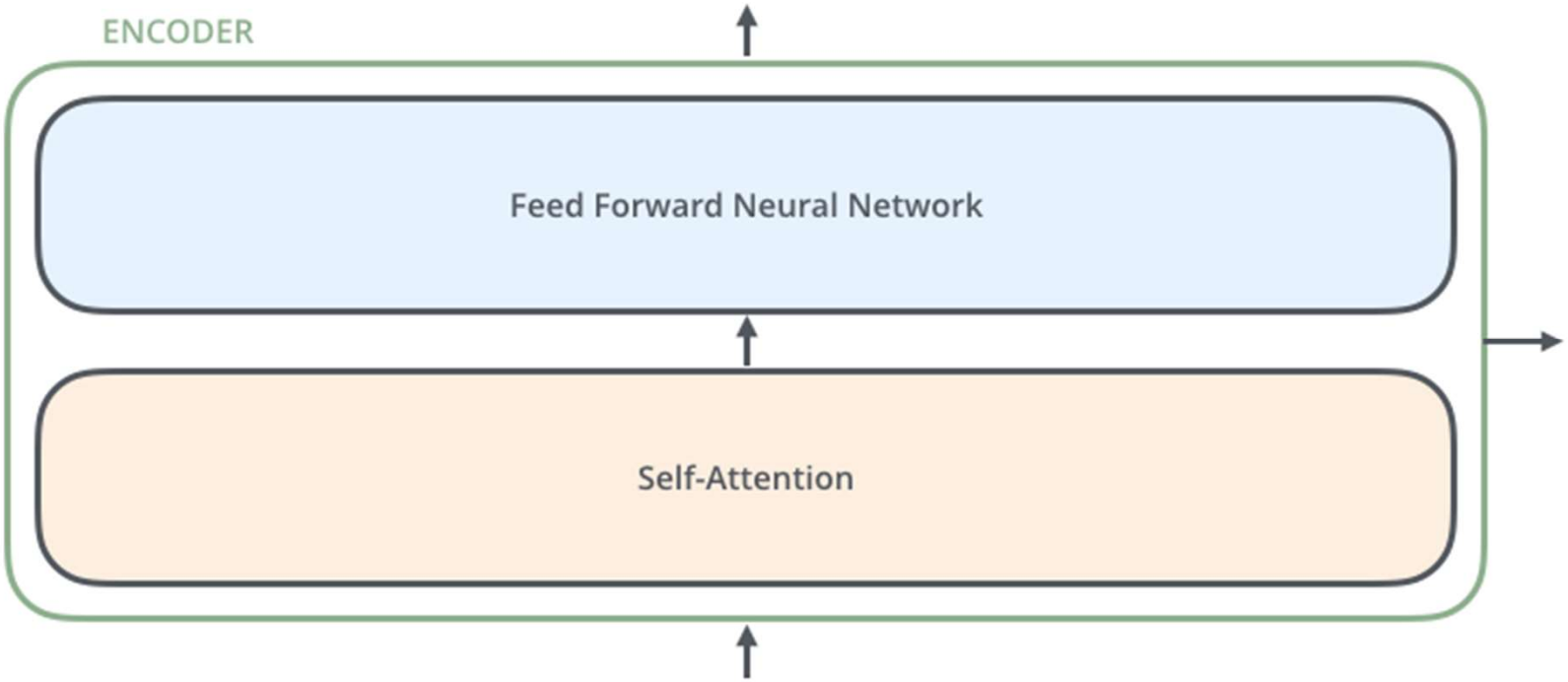
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in NeurIPS, 2017. (34,000 citations)
- Two valuable sources
  - http://nlp.seas.harvard.edu/2018/04/03/attention.html
  - https://jalammar.github.io/illustrated-transformer/ (slides come from this source)
- Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Transformer



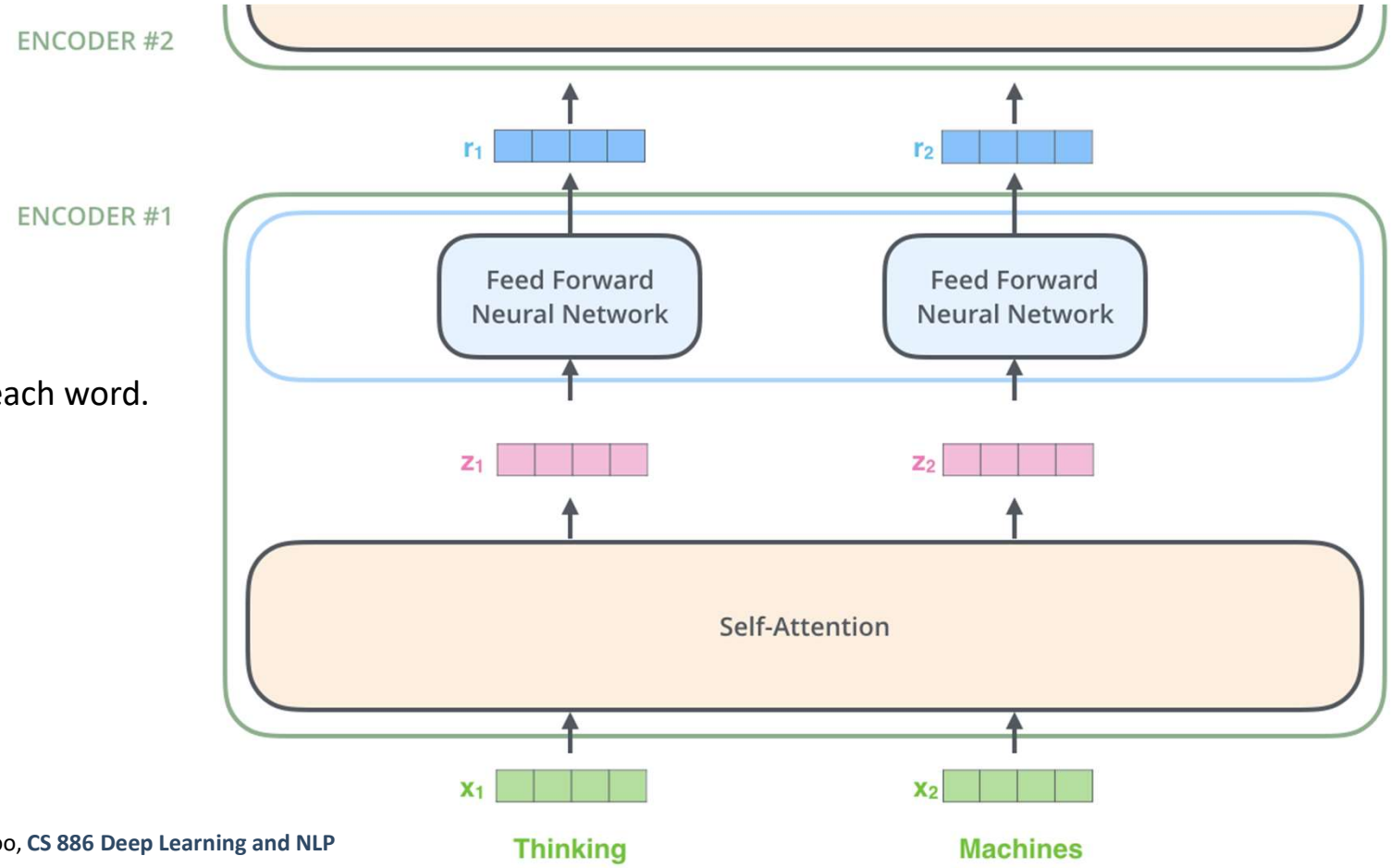
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

An Encoder Block: same structure, different parameters



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

Encoder



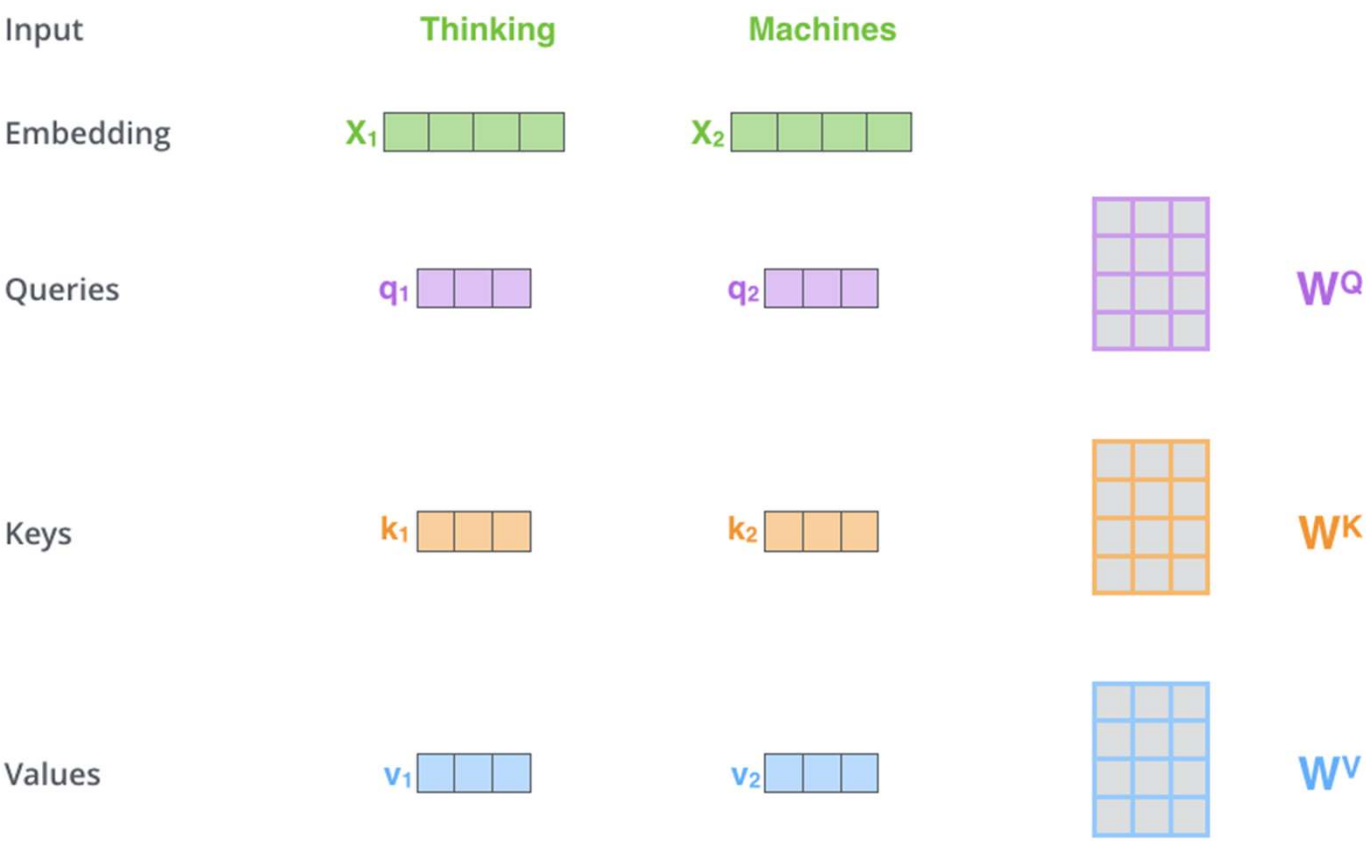
The ffnn is independent for each word.
Hence can be parallelized.

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

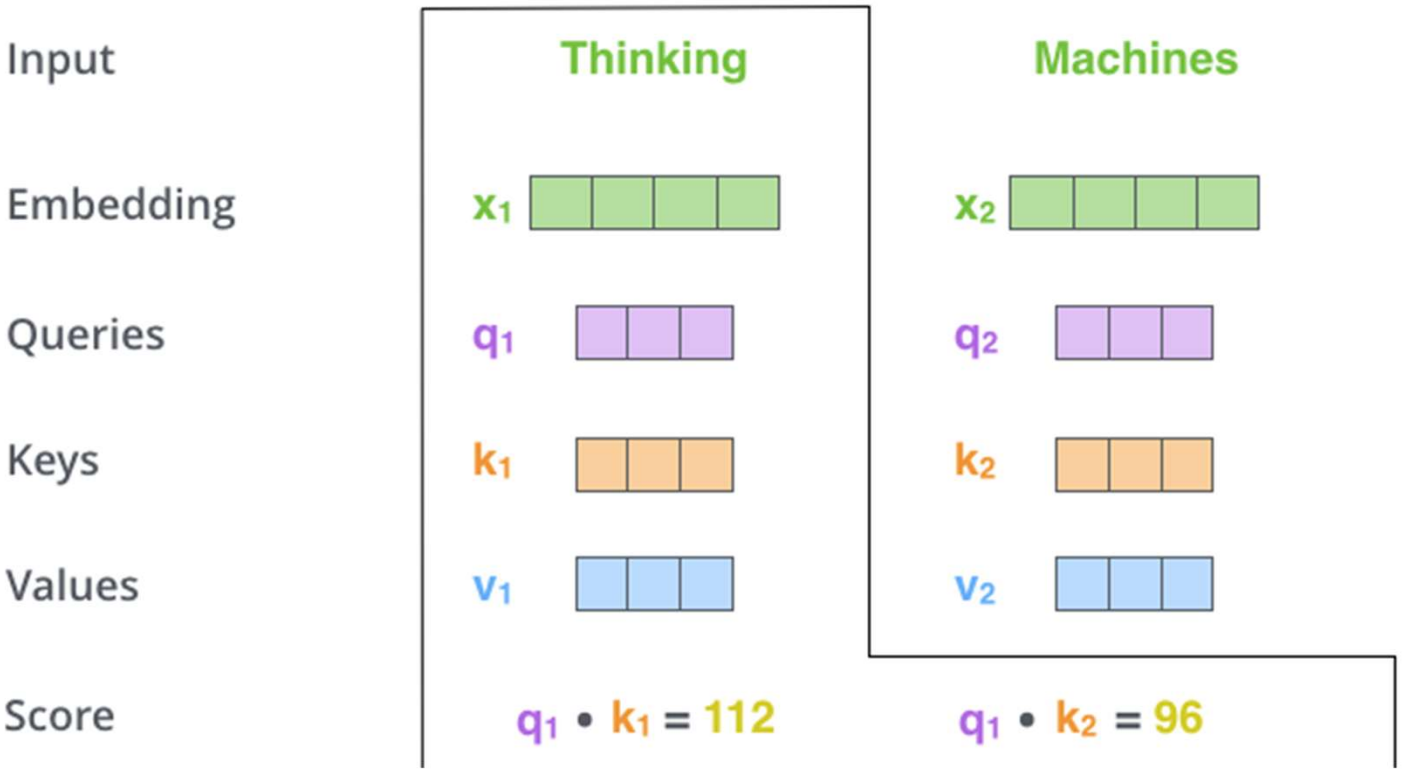
- $q_i = x_i W^Q$
- $K_i = x_i W^K$
- $V_i = x_i W^V$



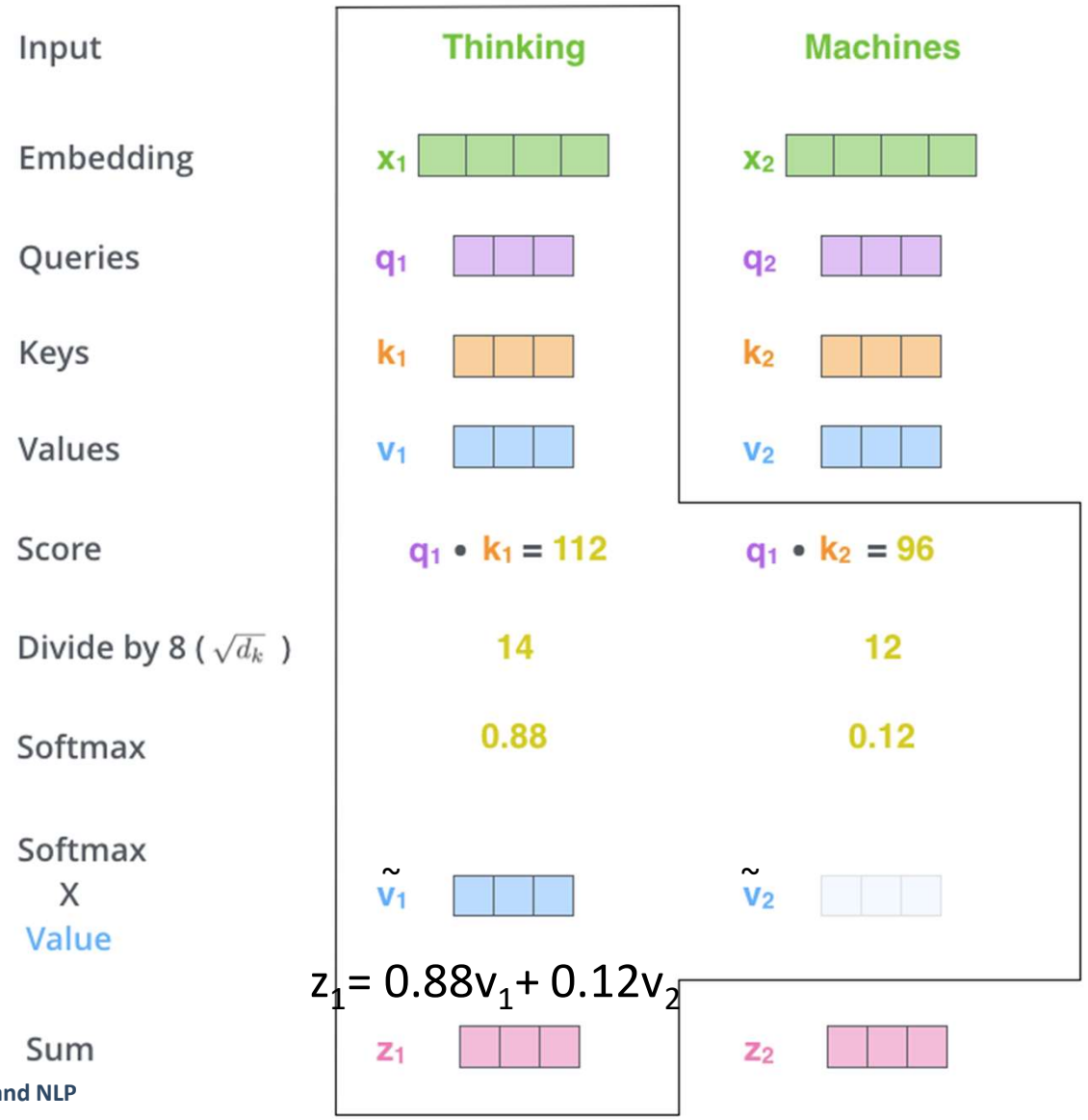
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Self Attention

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



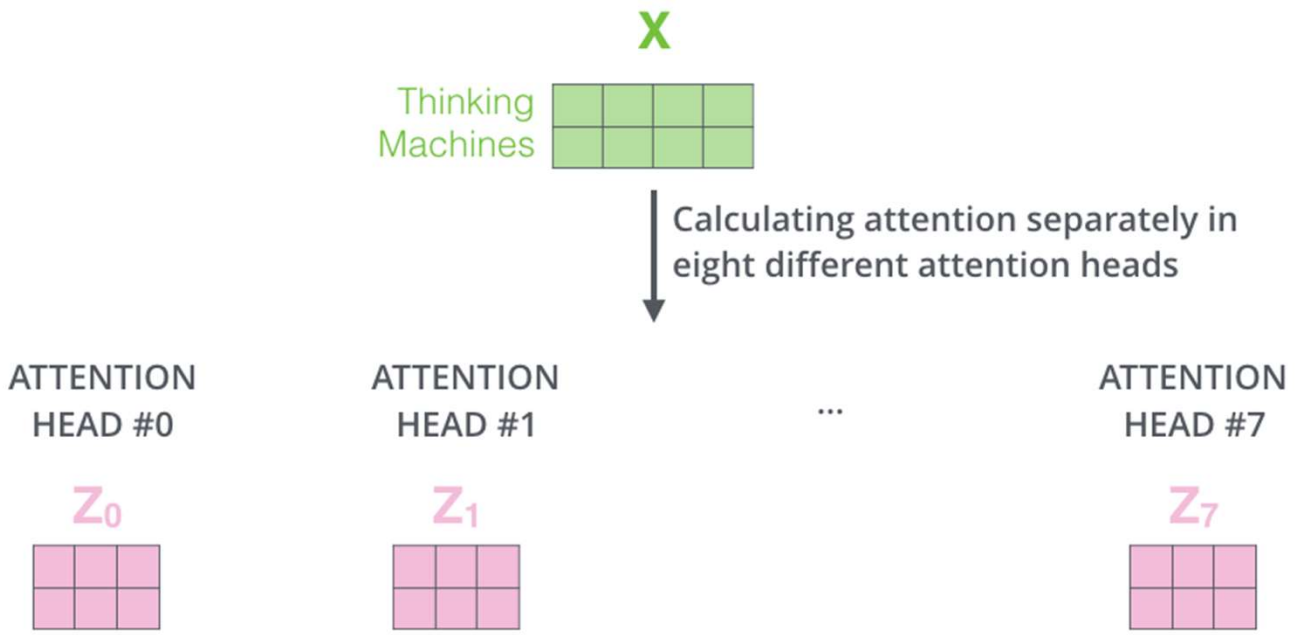
Self Attention



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

Multiple heads

1. It expands the mode ability to focus on different positions.
2. It gives the attention multiple “representa subspaces”



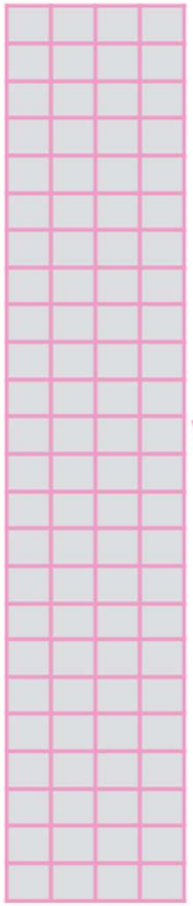
Attention and Transformers

1) Concatenate all the attention heads



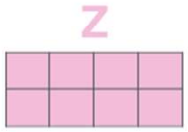
2) Multiply with a weight matrix W^O that was trained jointly with the model

X

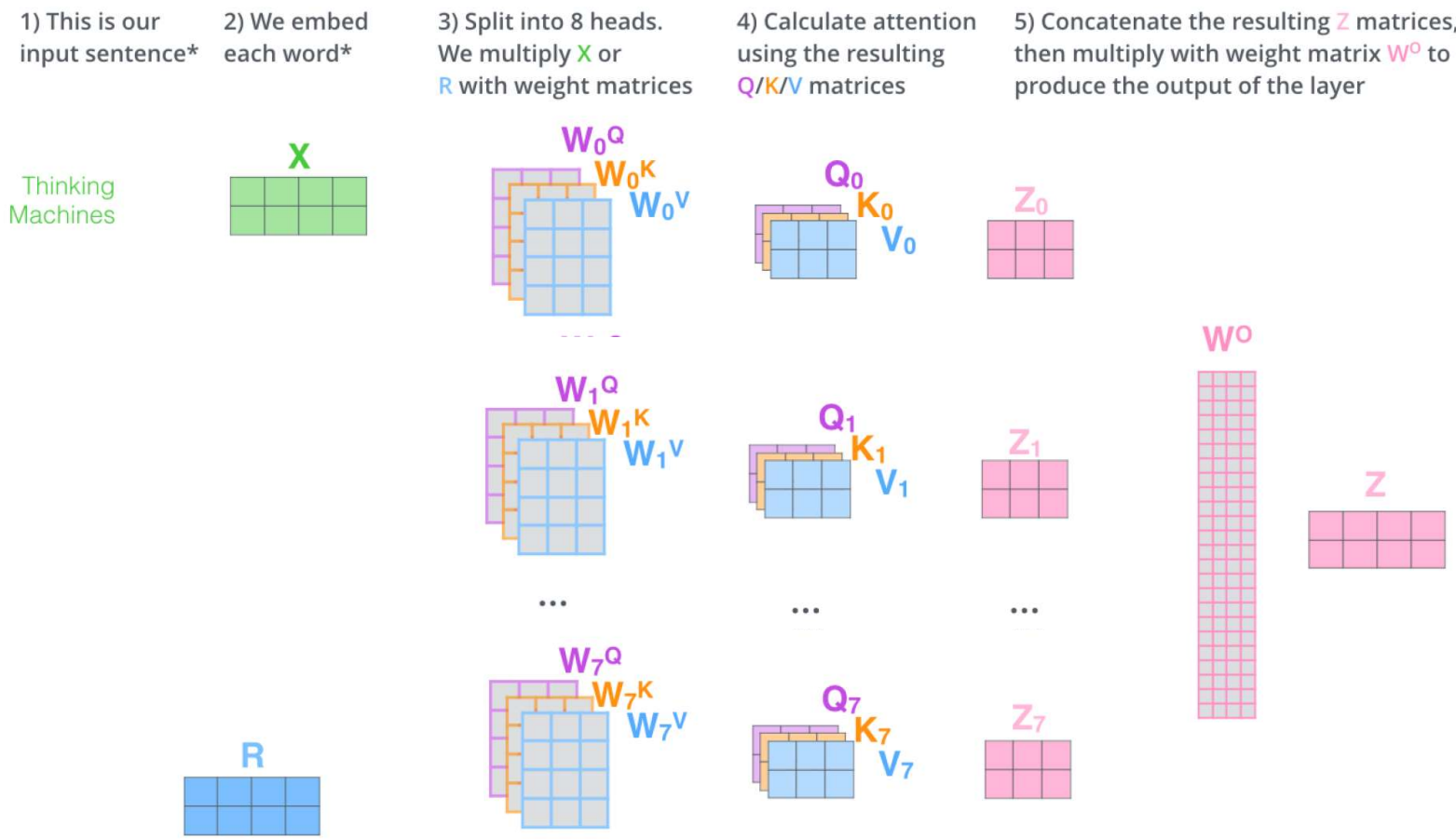


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

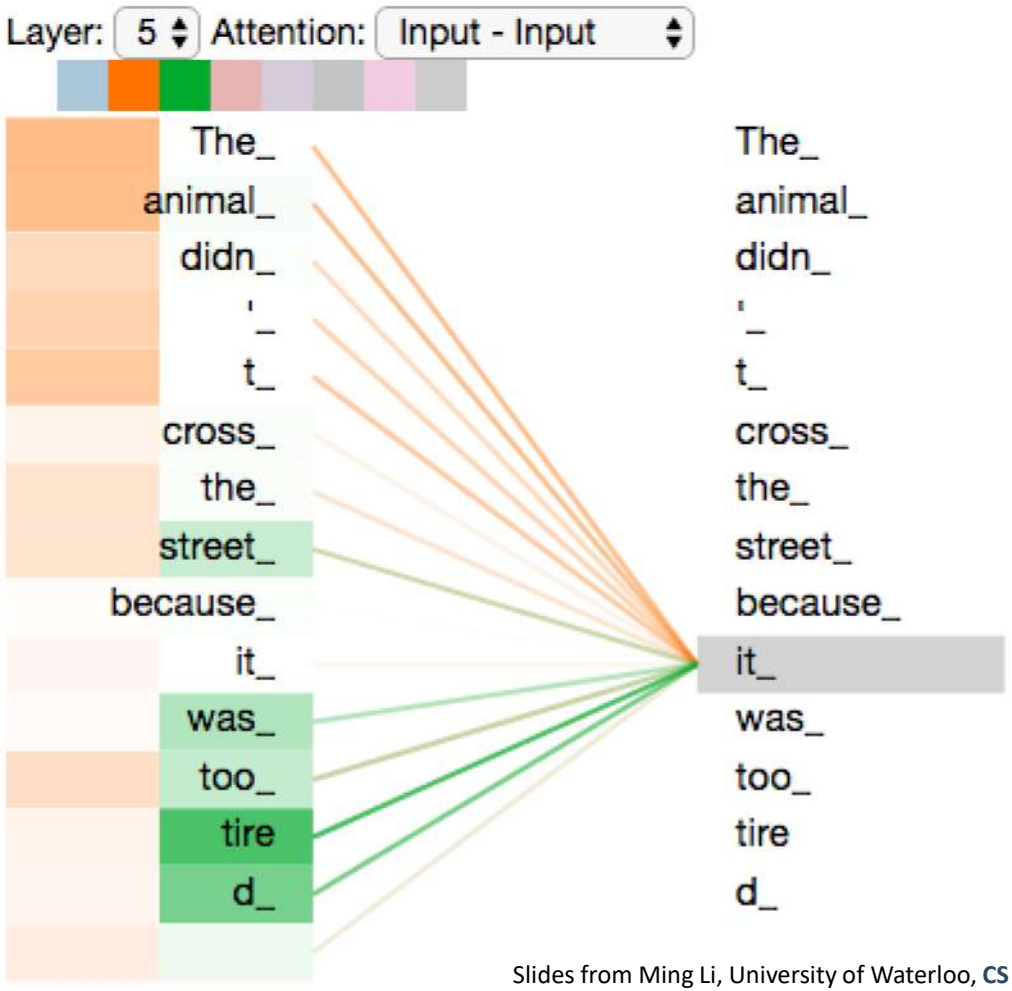
=



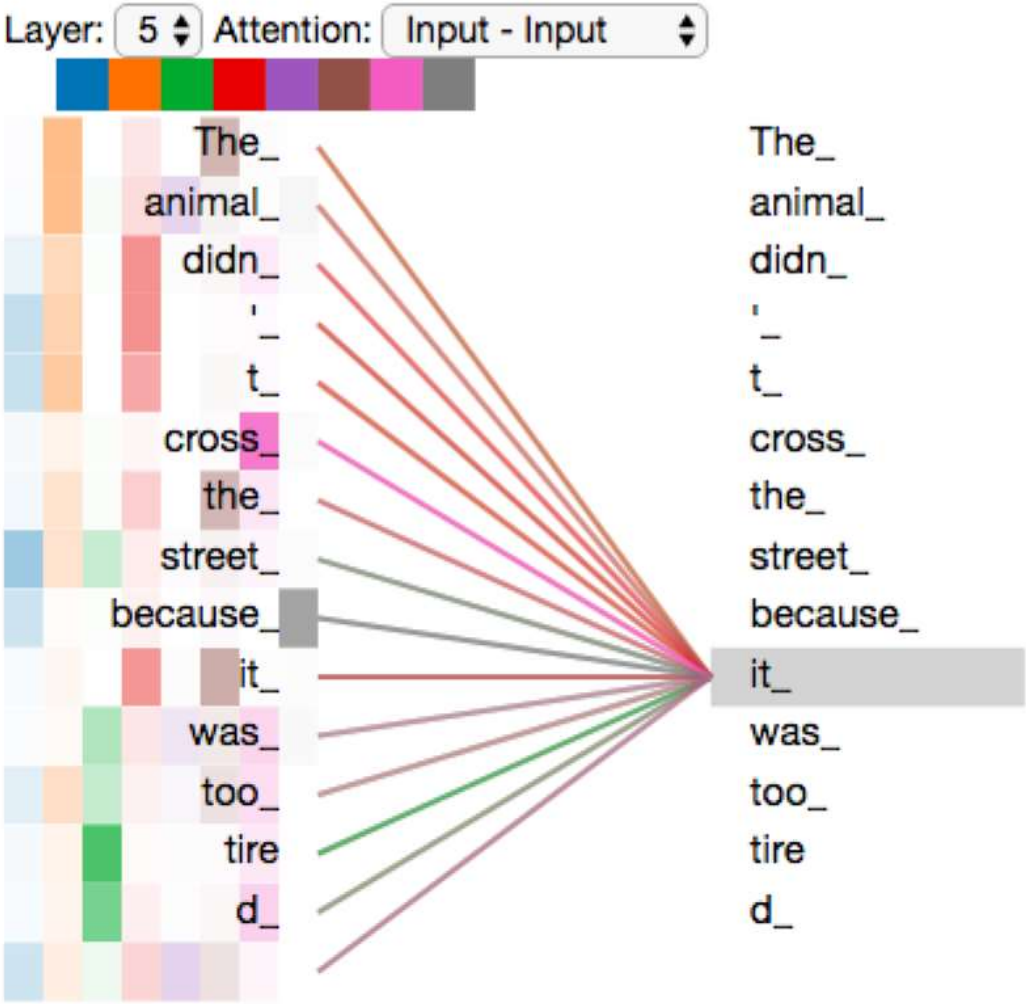
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**



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**

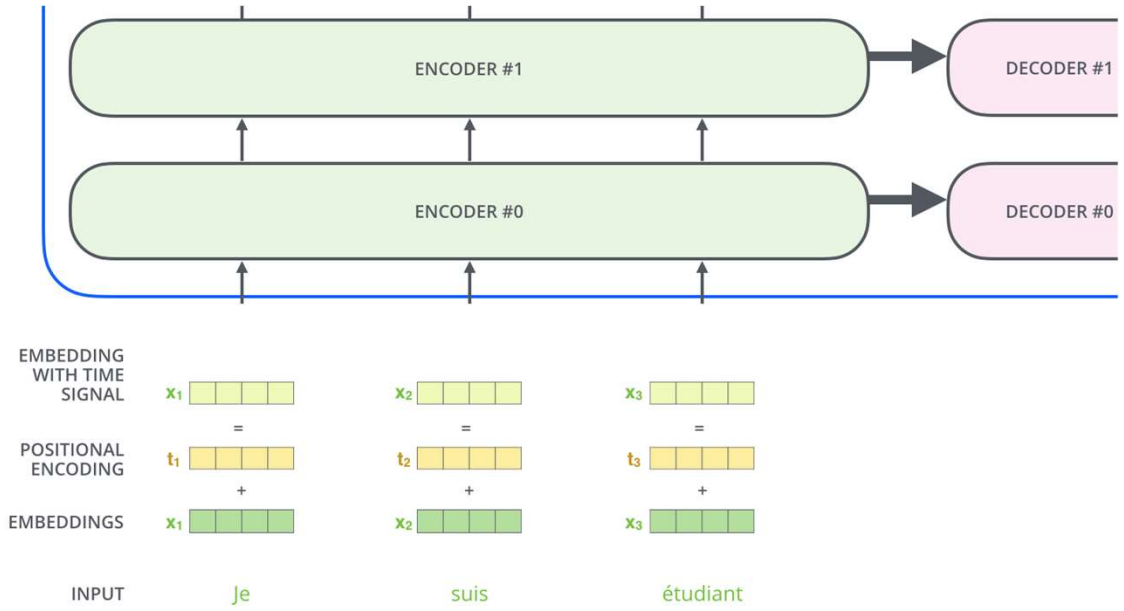


Representing the input order (positional encoding)

- 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)



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

Position Encoding

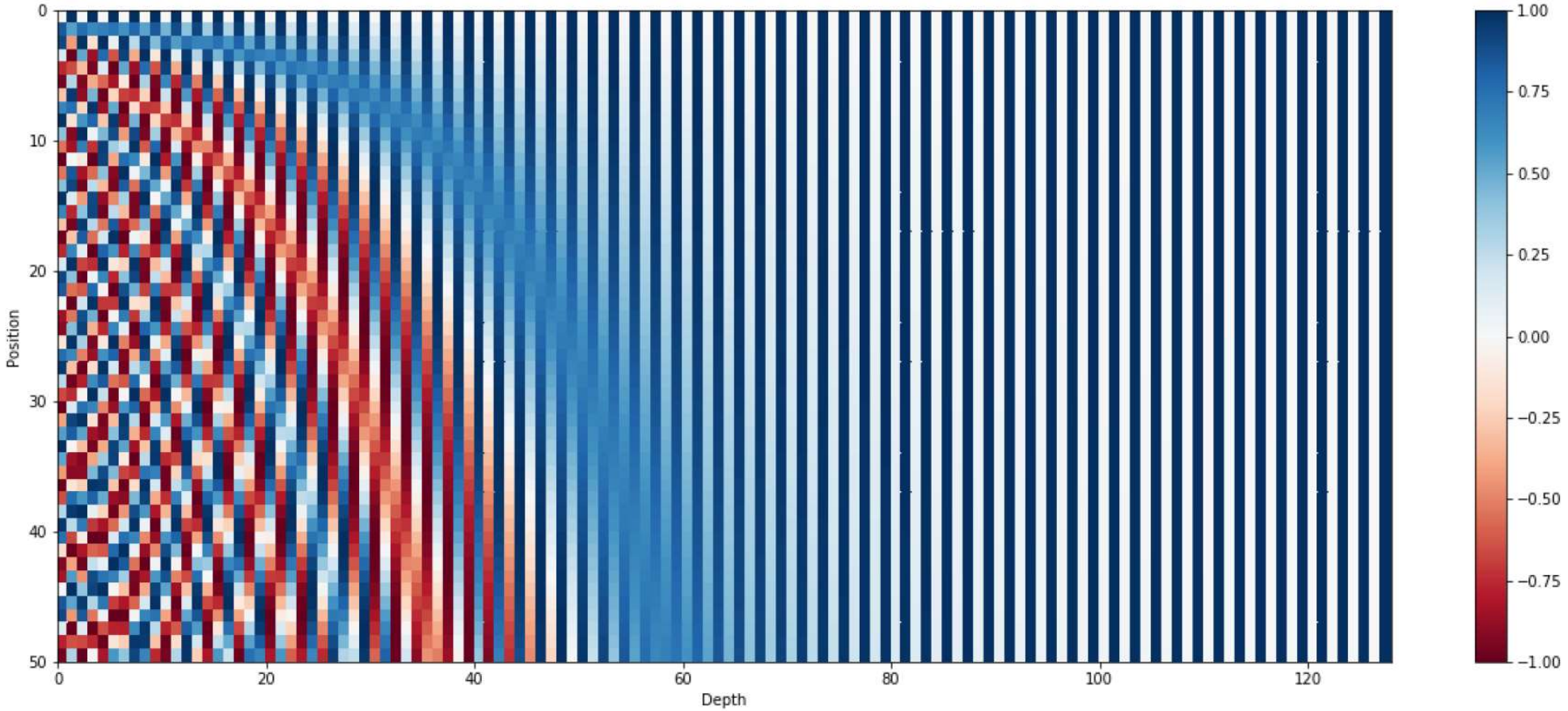
$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

where

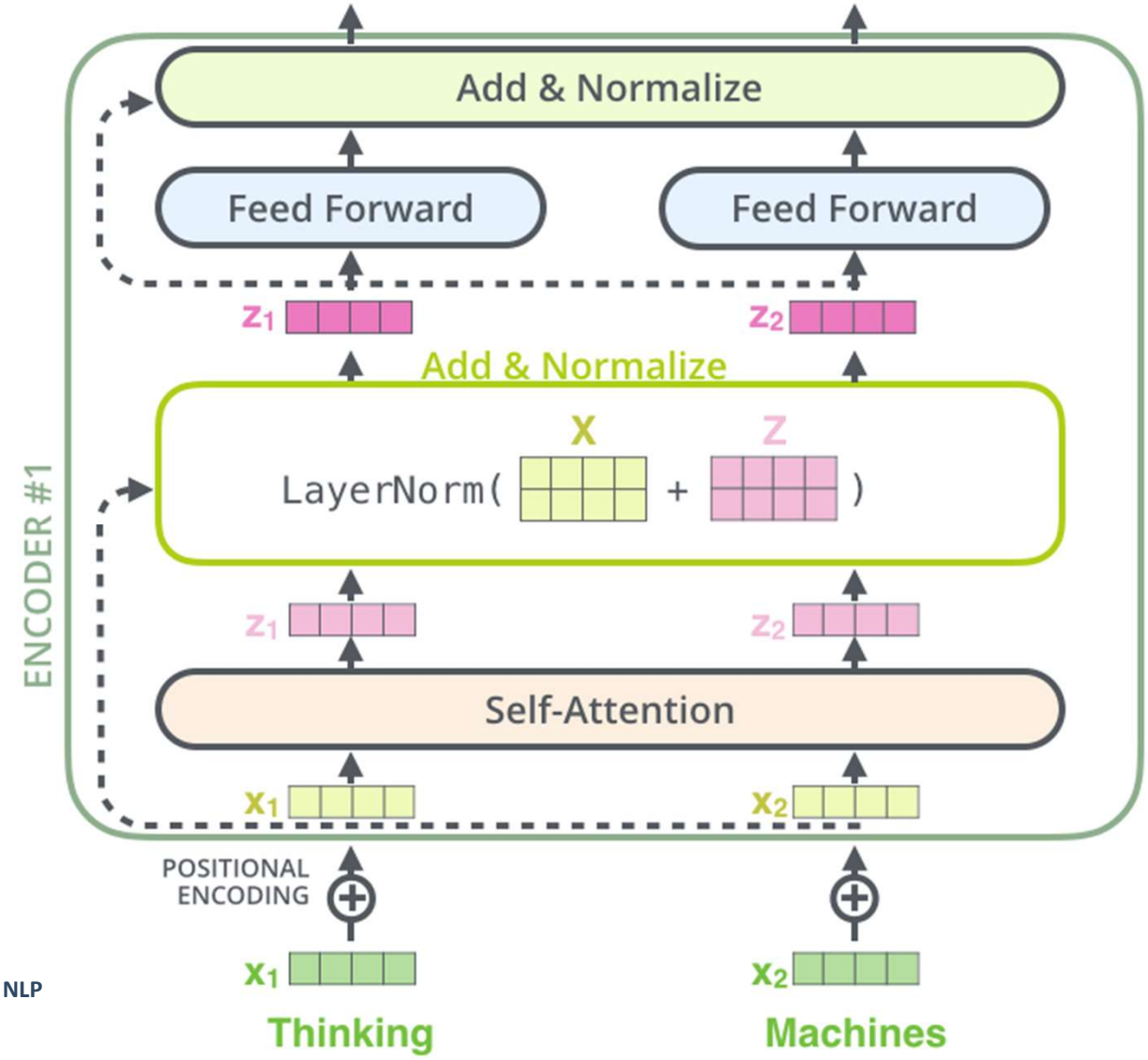
$$\omega_k = \frac{1}{10000^{2k/d}}$$

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

Position Encoding



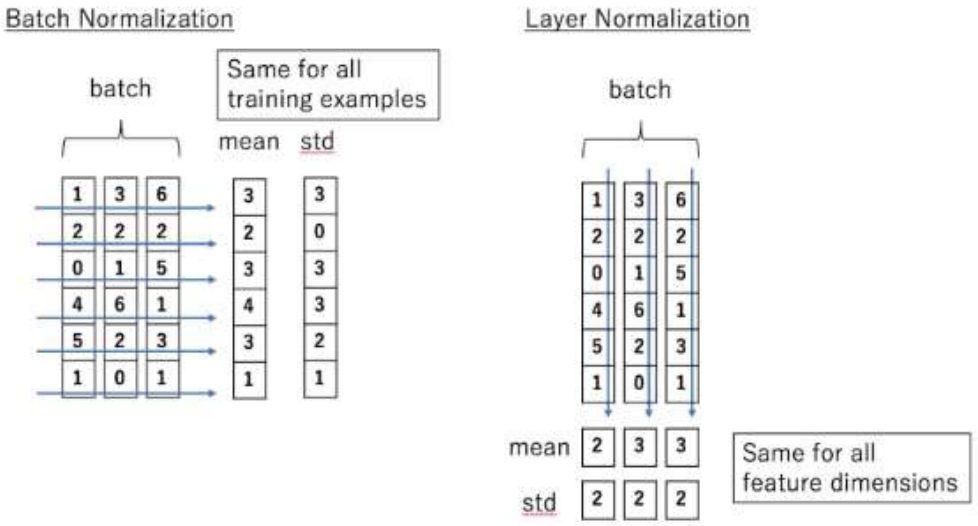
Add and Normalize

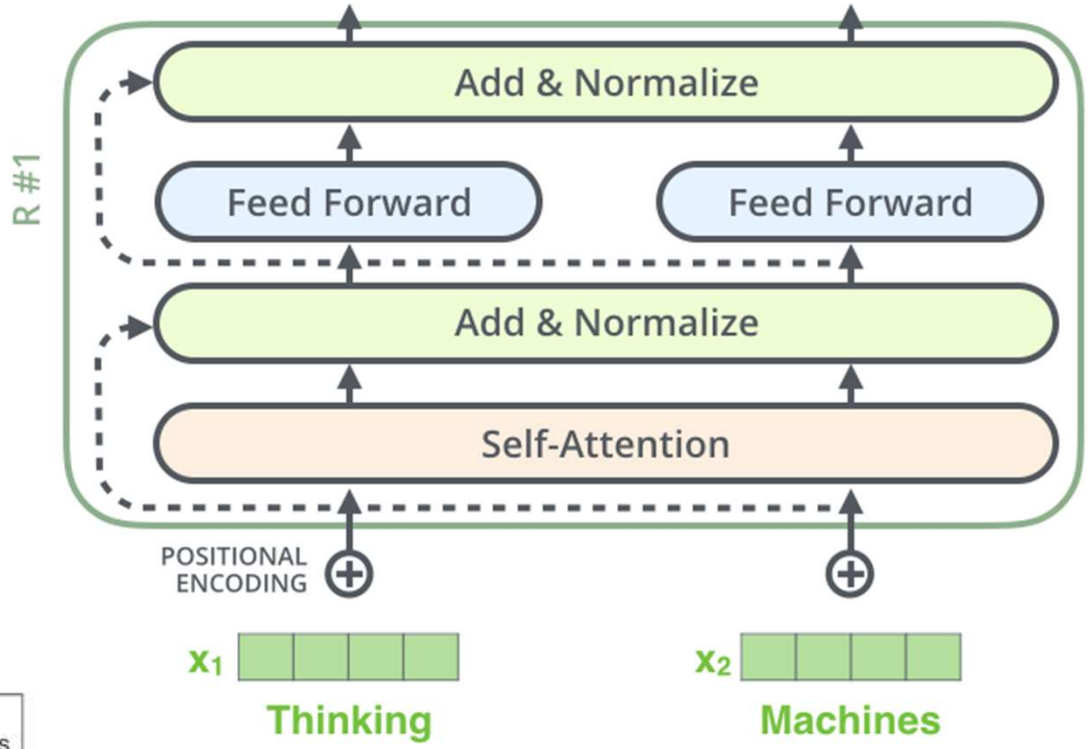


Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Layer Normalization (Hinton)

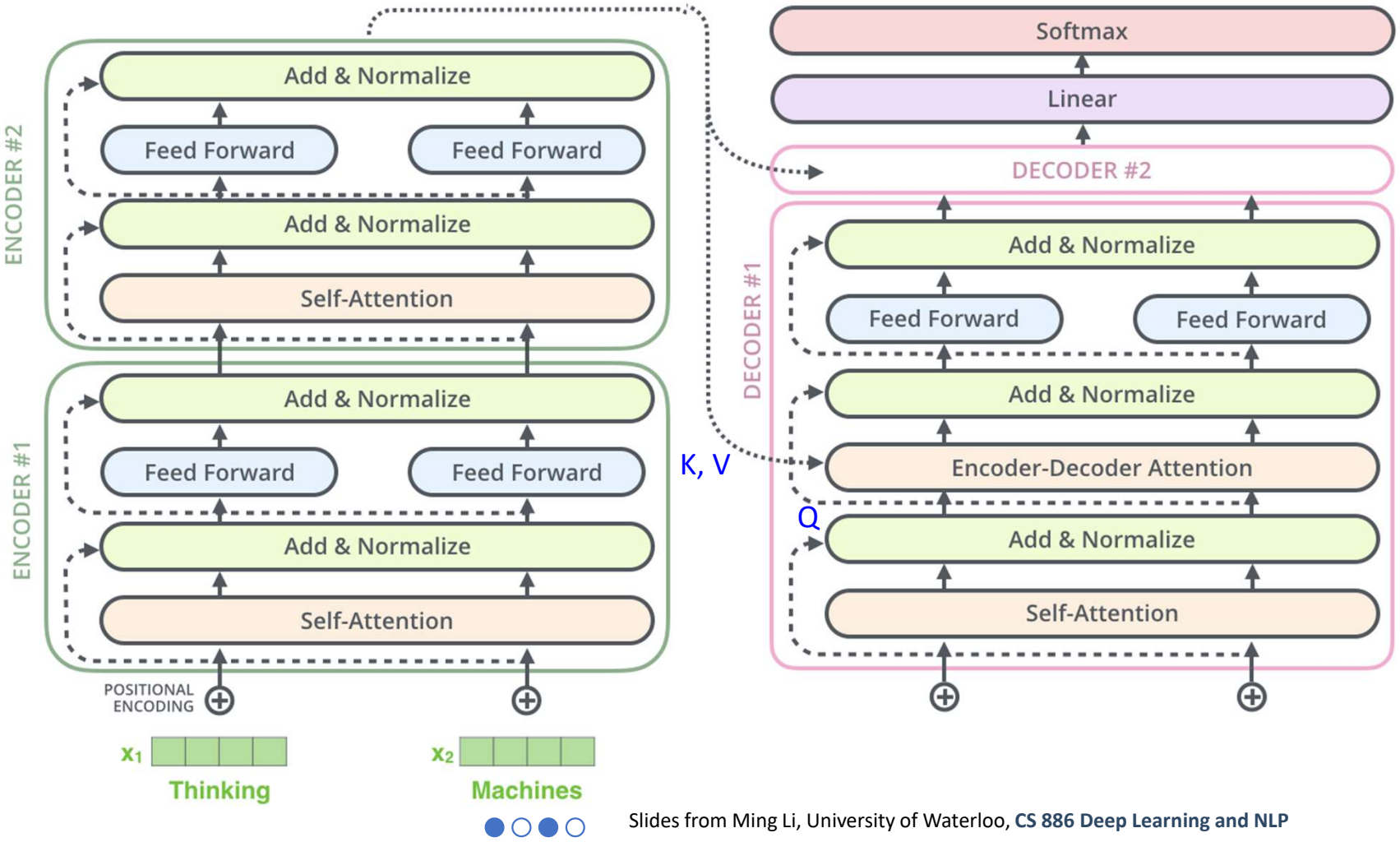
Layer normalization normalizes the inputs across the features.



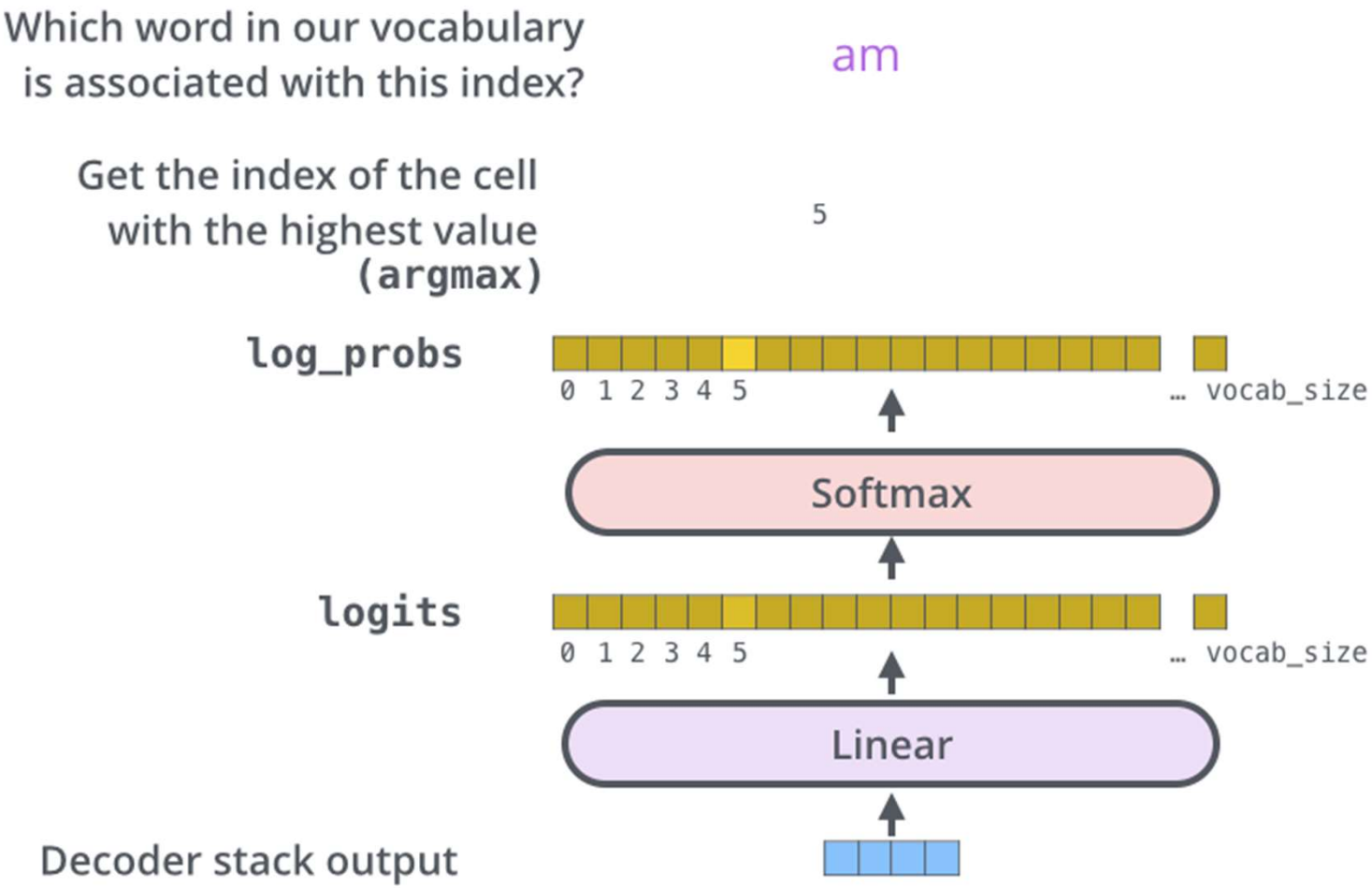


Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

The complete transformer

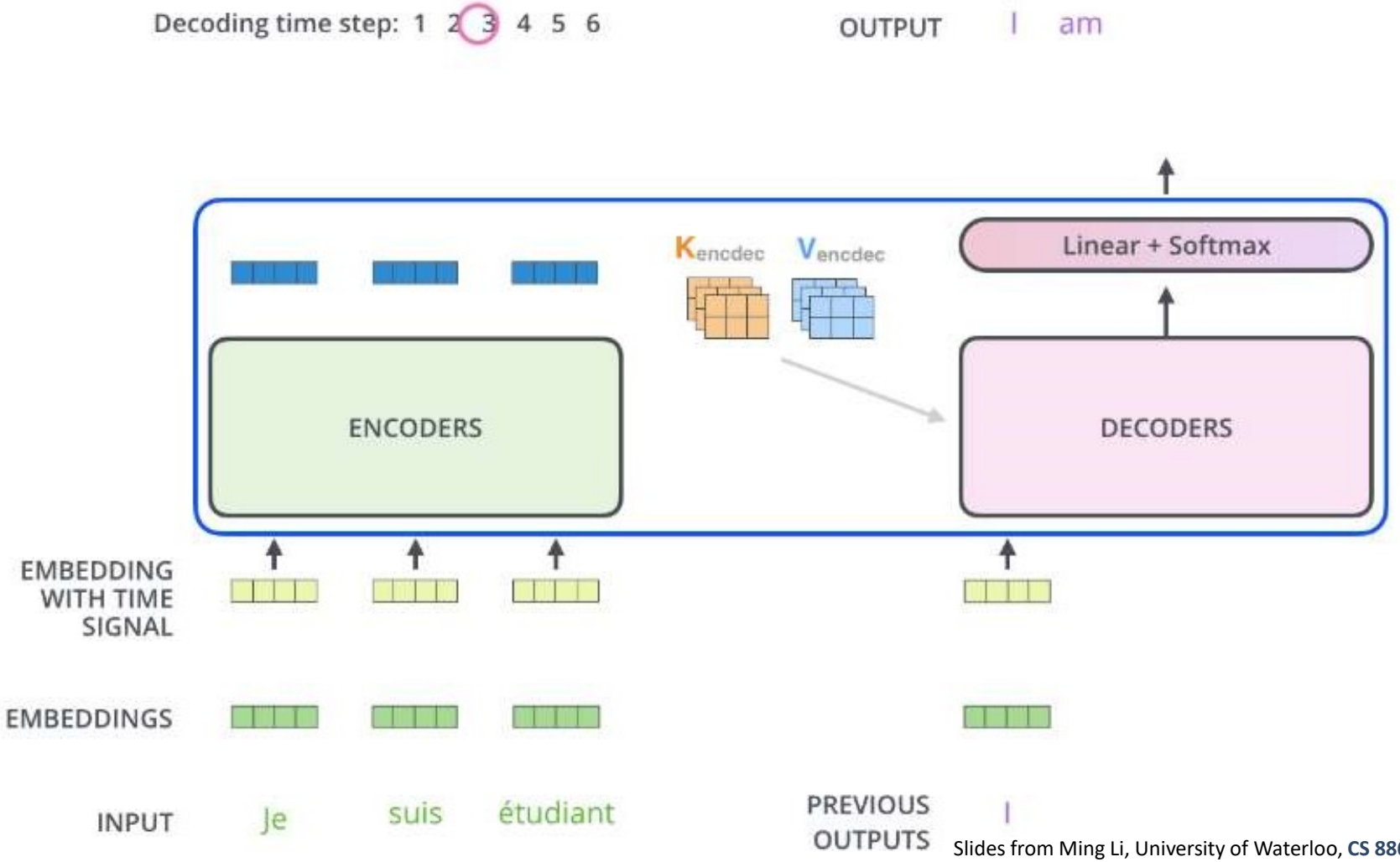


Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**



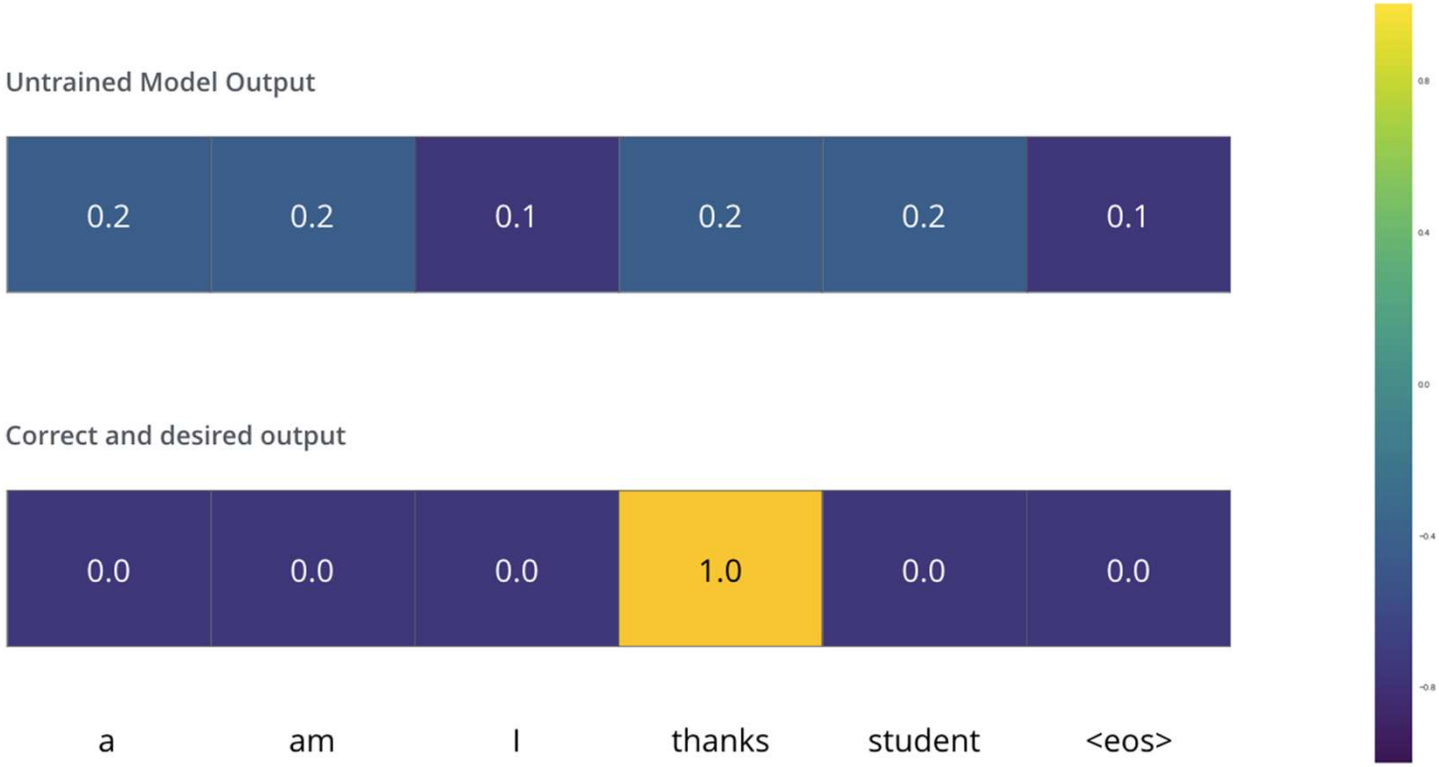
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

How it works



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

Training and the Loss Function



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

Cross Entropy and KL (Kullback-Leibler) divergence

- Entropy: $E(P) = - \Sigma_i P(i) \log P(i)$
- Cross Entropy: $C(P) = - \Sigma_i P(i) \log Q(i)$
- KL divergence: $D_{KL}(P \,||\, Q) = \Sigma_i P(i) \log[P(i)/Q(i)] = \Sigma_i P(i)[\log P(i) - \log Q(i)]$
- JSD$(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$, $M = \frac{1}{2}(P+Q)$, symmetric KL

* JSD = Jensen-Shannon Divergency