Transformers

Ranjeet Ranjan Jha

Mathematics Department
Indian Institute of Technology Patna

October 9, 2024

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)

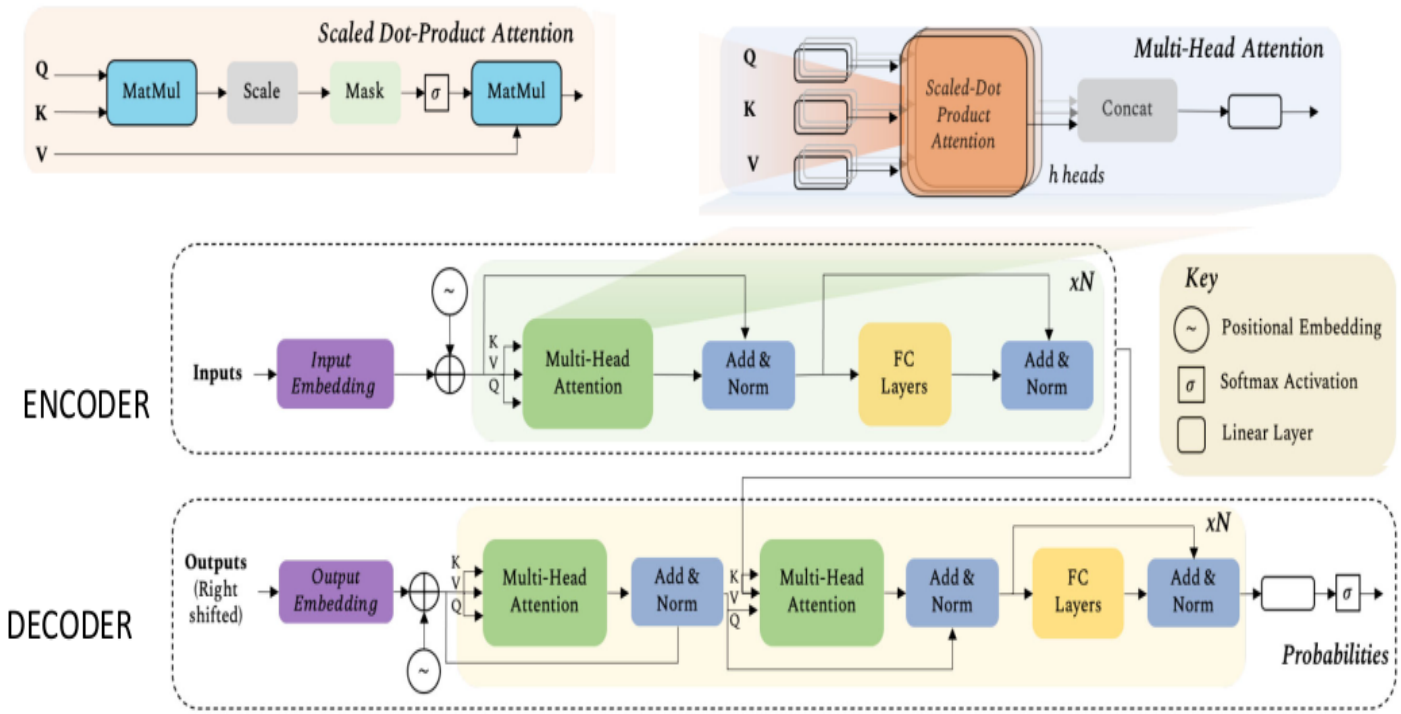
Natural Language Processing

- 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)

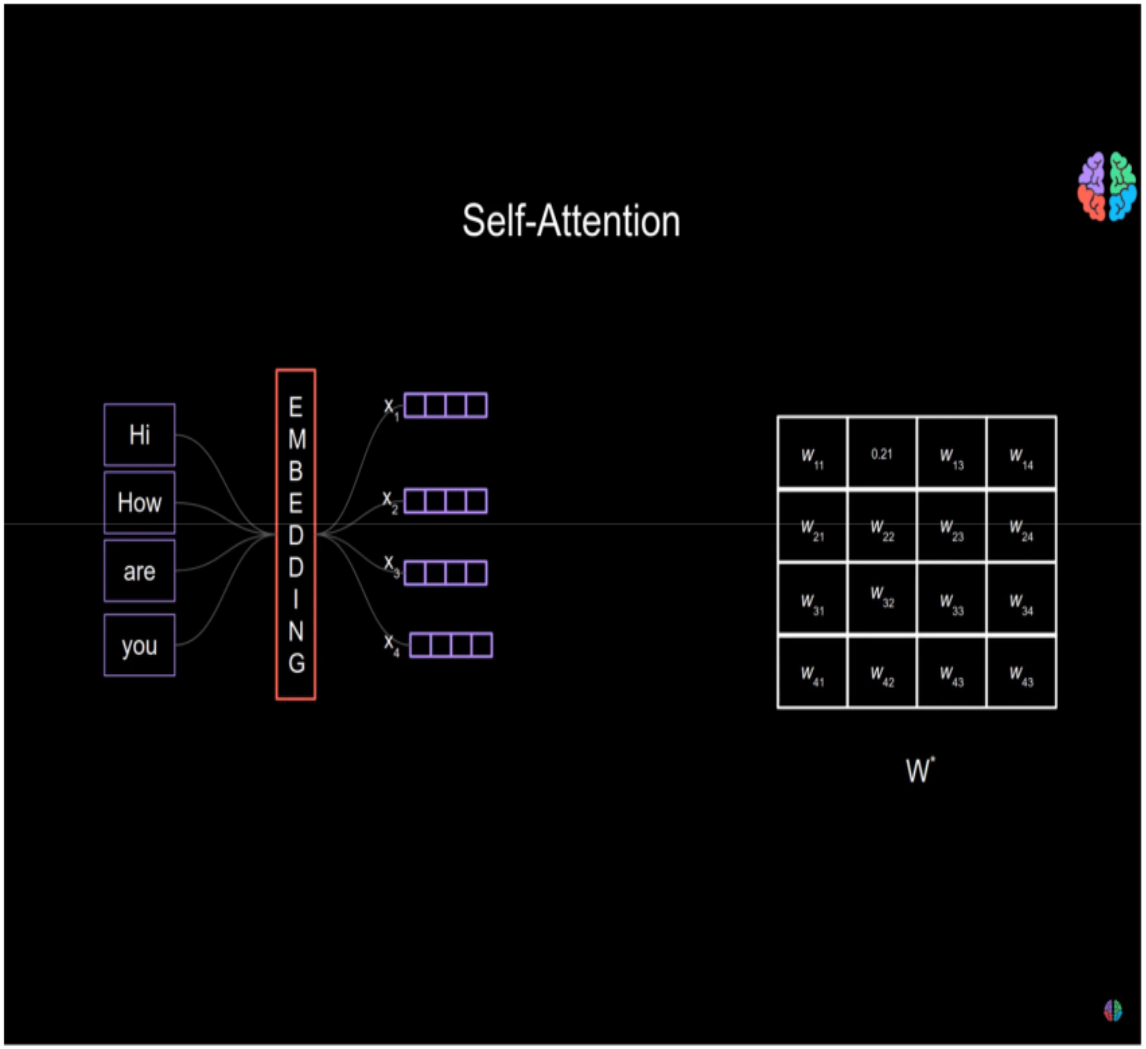
Transformer Basics

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

Transformer

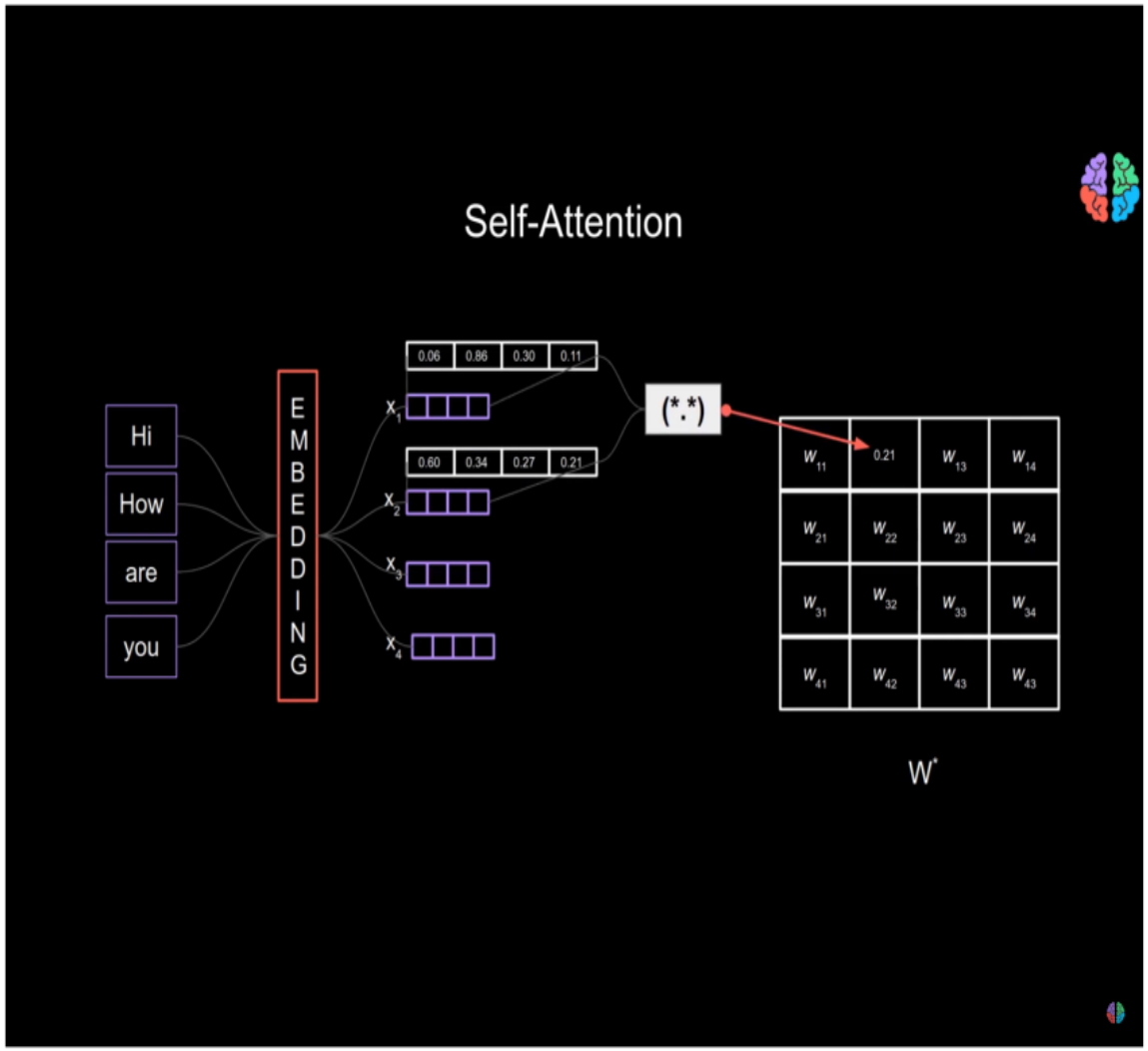


Slide courtesy of AI Bites, Youtube Channel: https://www.youtube.com/c/AIBites



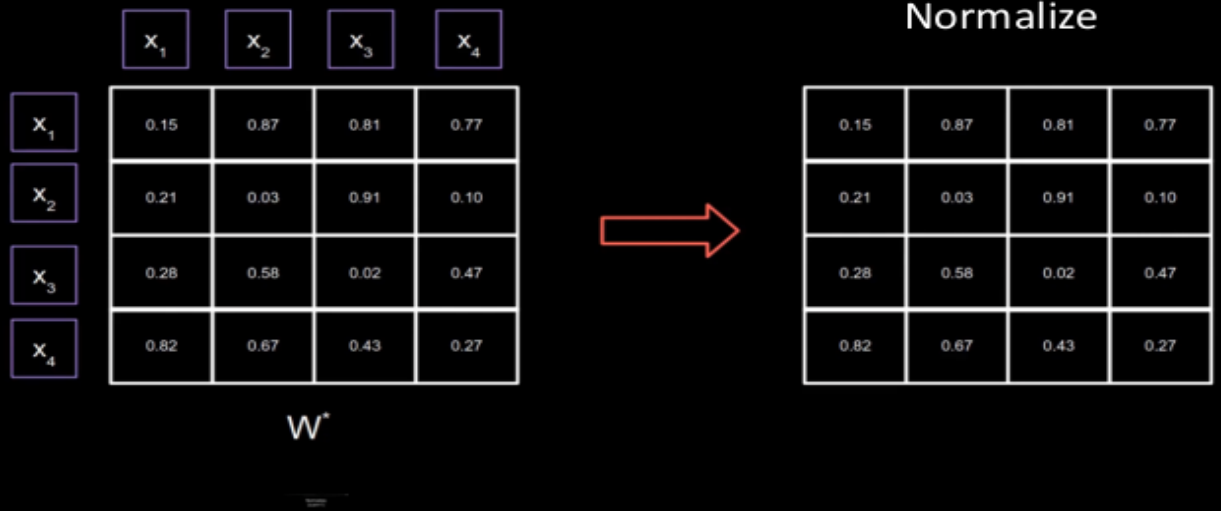
1/11/2024

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

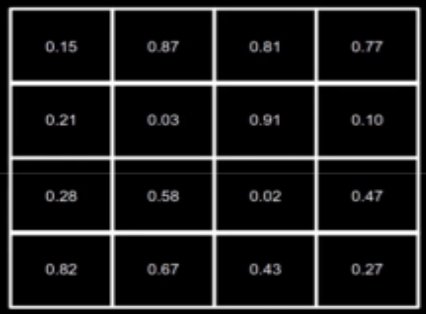


1/11/2024

Self-Attention

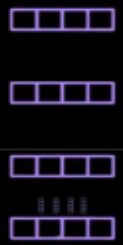


Self-Attention

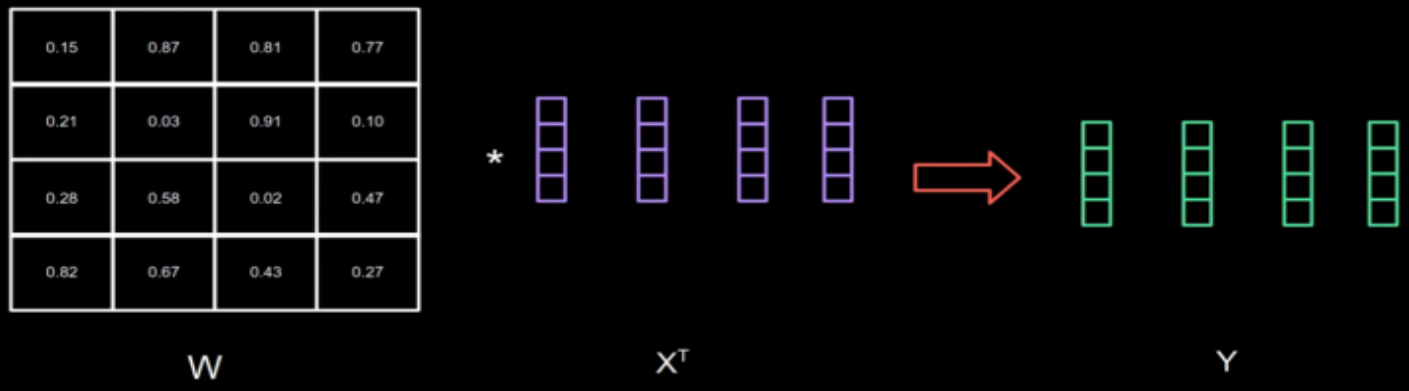


W

*

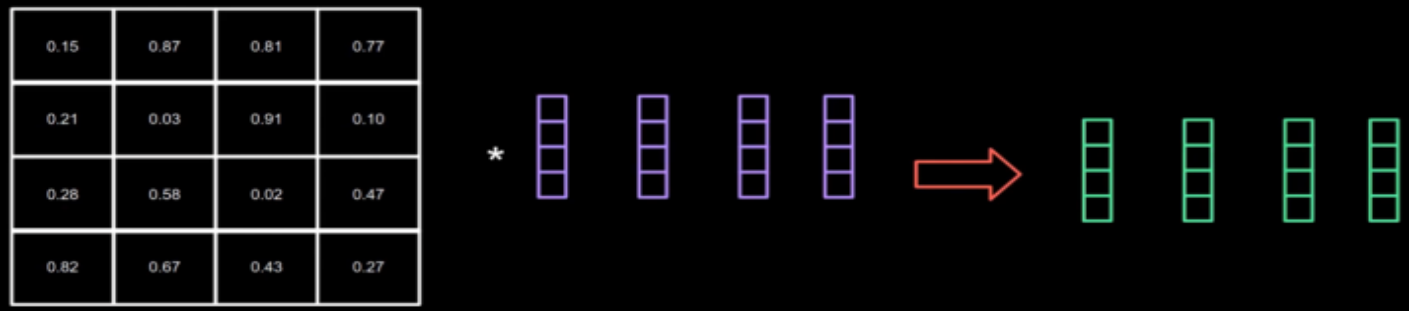


Self-Attention



W X^T Y

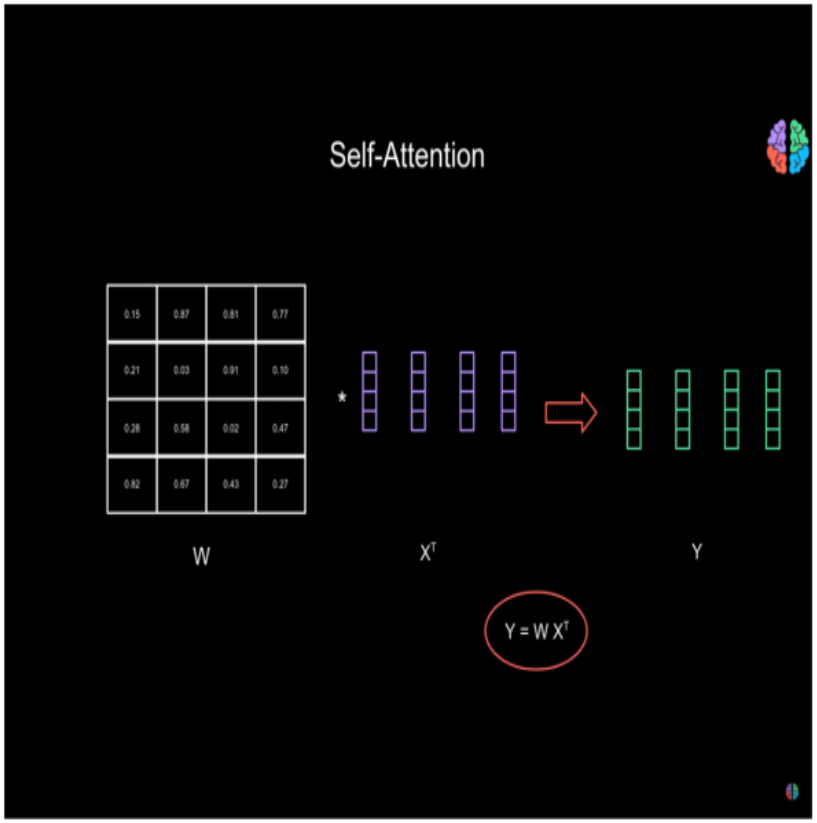
Self-Attention

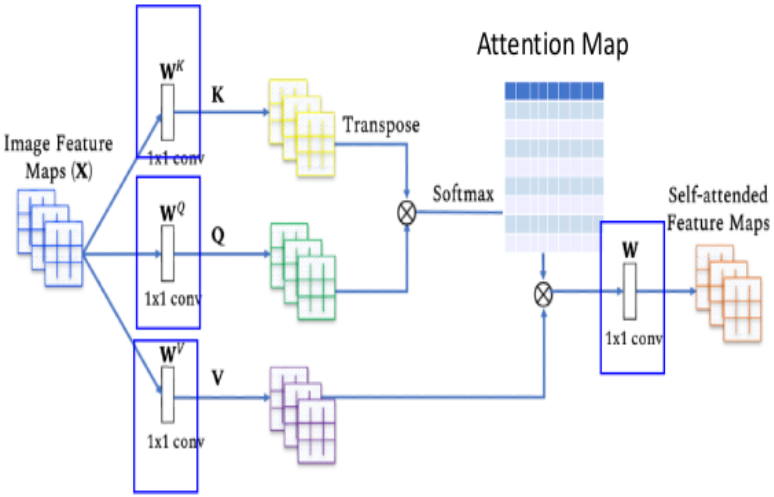


Self-Attention

- So far no learning!

Self-Attention of Image Features



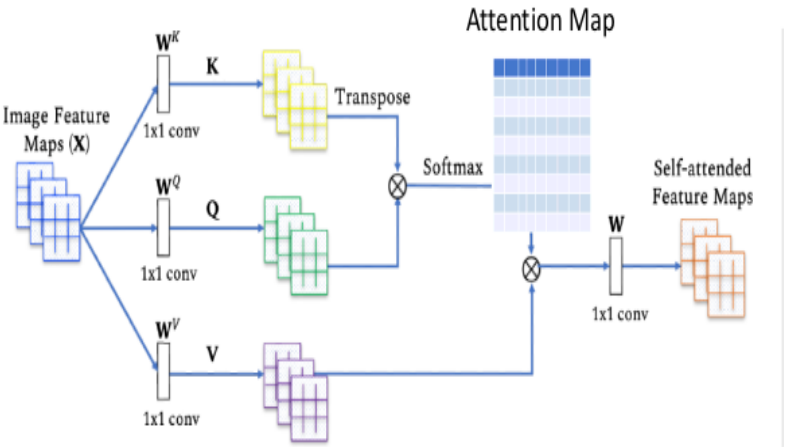


Self-Attention (Matrix Explanation)

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

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

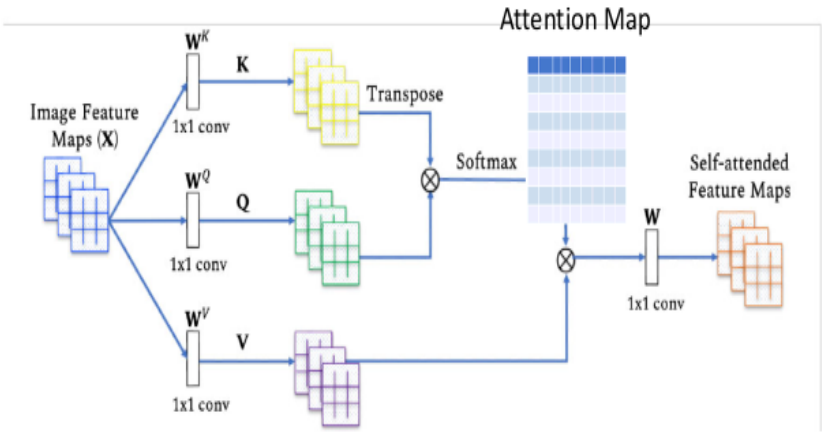


Self-Attention

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

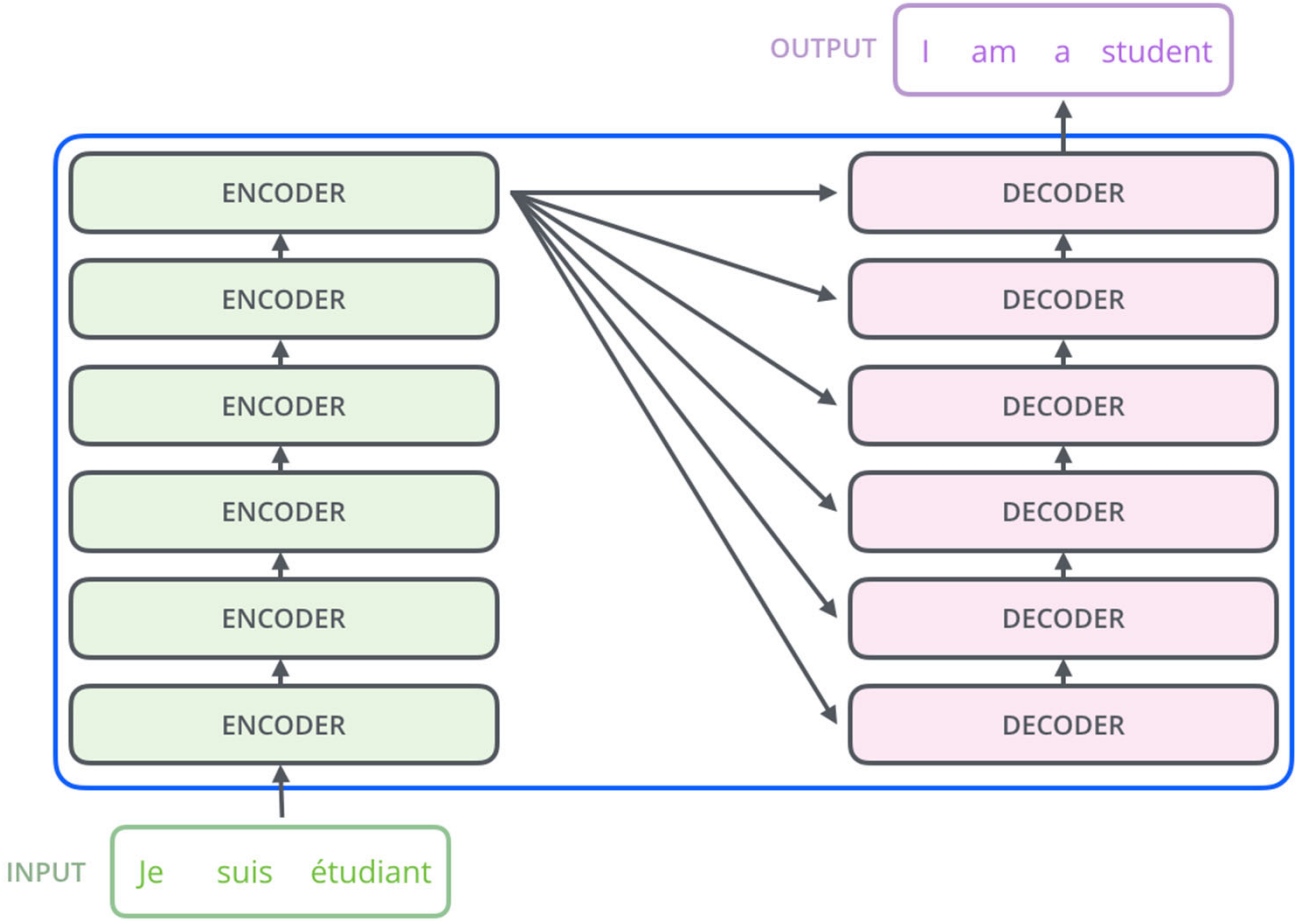
$$\mathbf{Z} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_q}}\right)\mathbf{V}.$$



Transformers (Attention is all you need 2017)

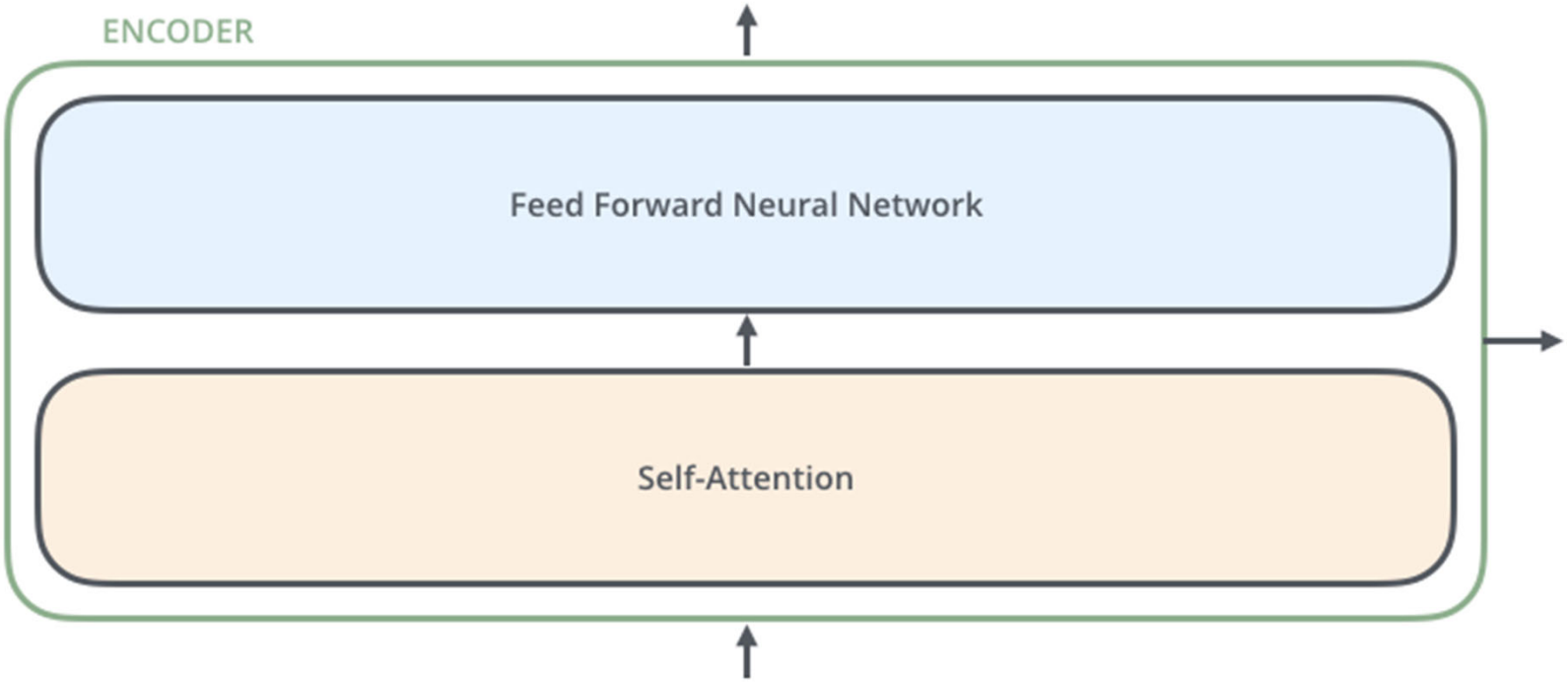
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in NeurIPS, 2017. (102,618 citations)
- Two valuable sources
  - http://nlp.seas.harvard.edu/2018/04/03/attention.html
  - https://jalammar.github.io/illustrated-transformer/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

Transformer



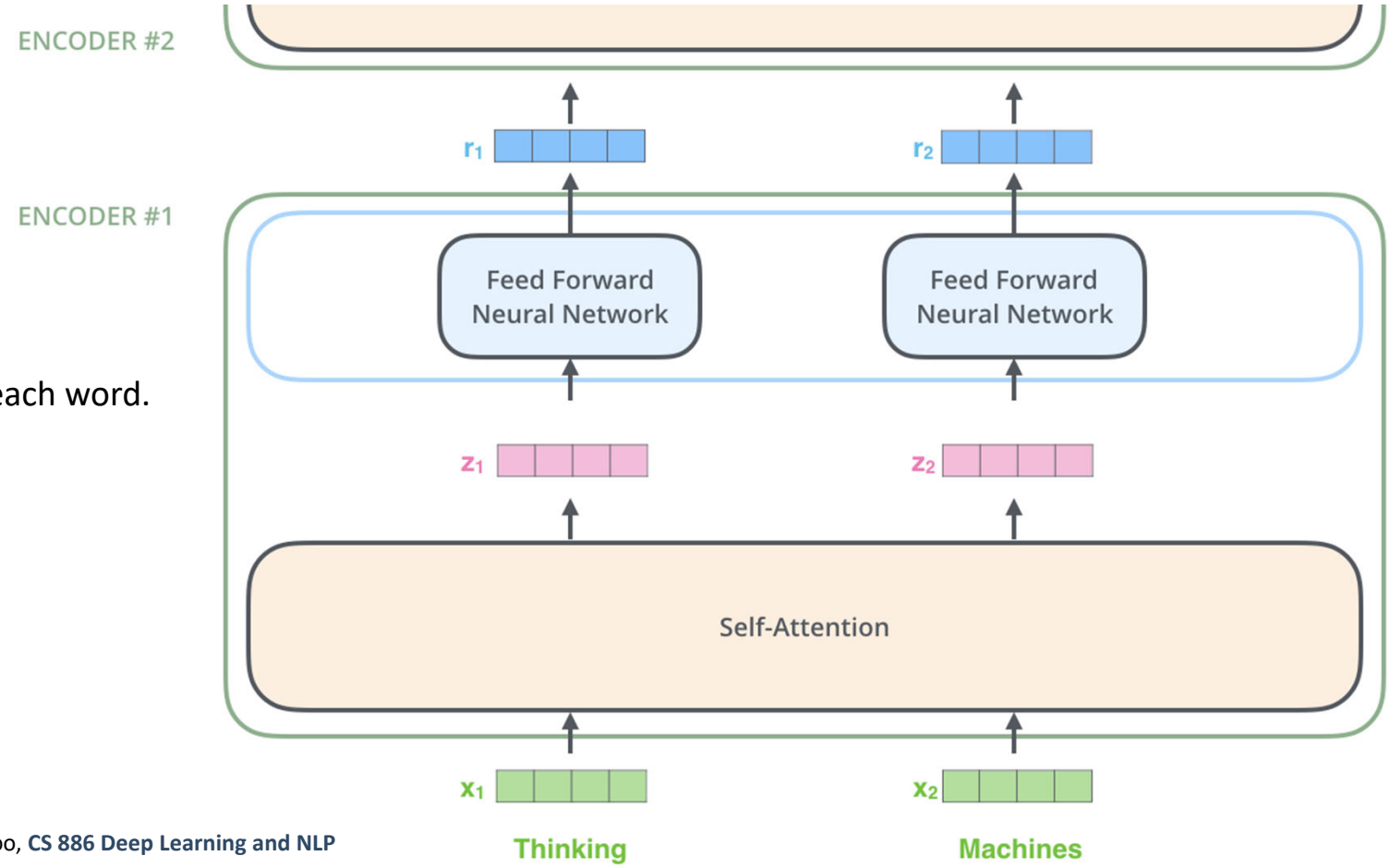
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

An Encoder Block: same structure, different parameters



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

Encoder



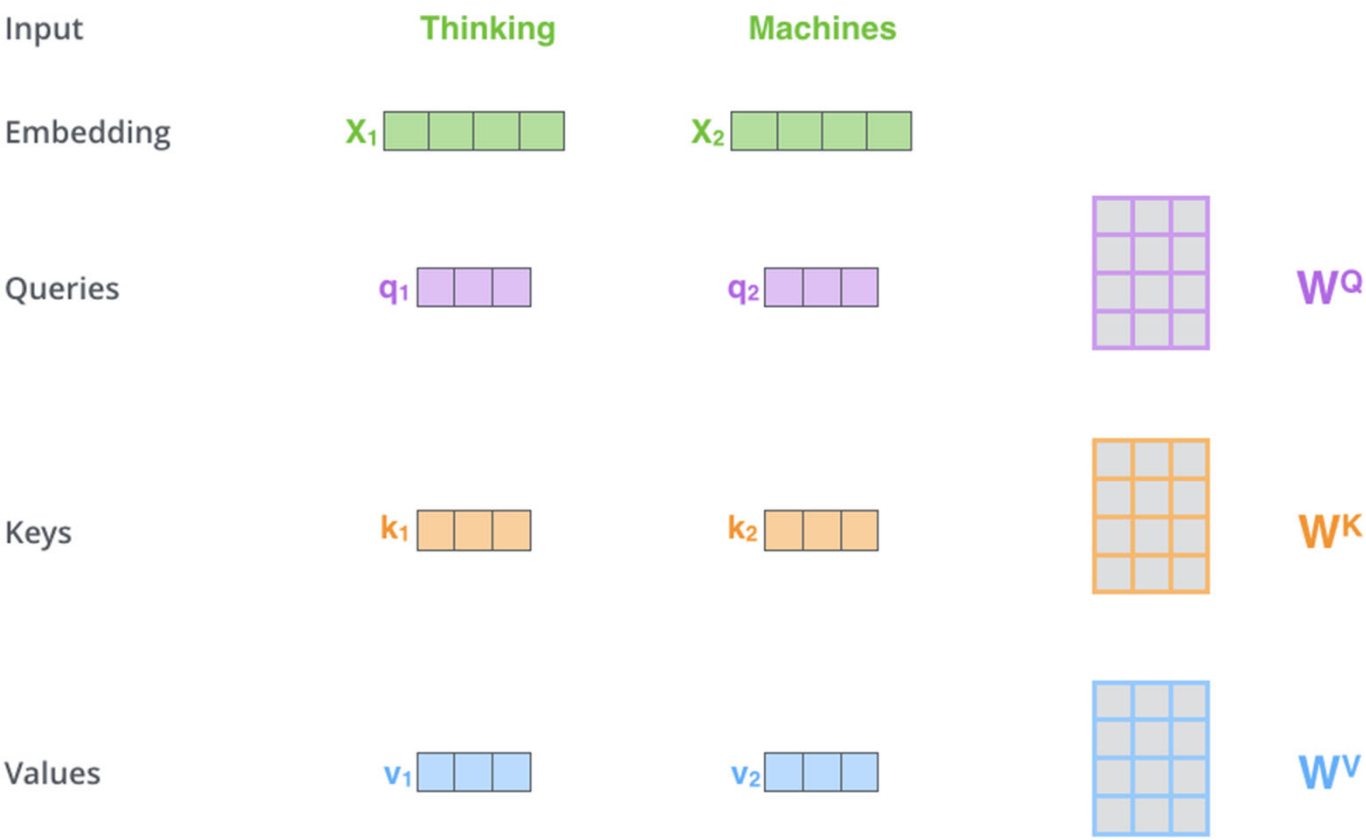
The ffnn is independent for each word.
Hence can be parallelized.

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

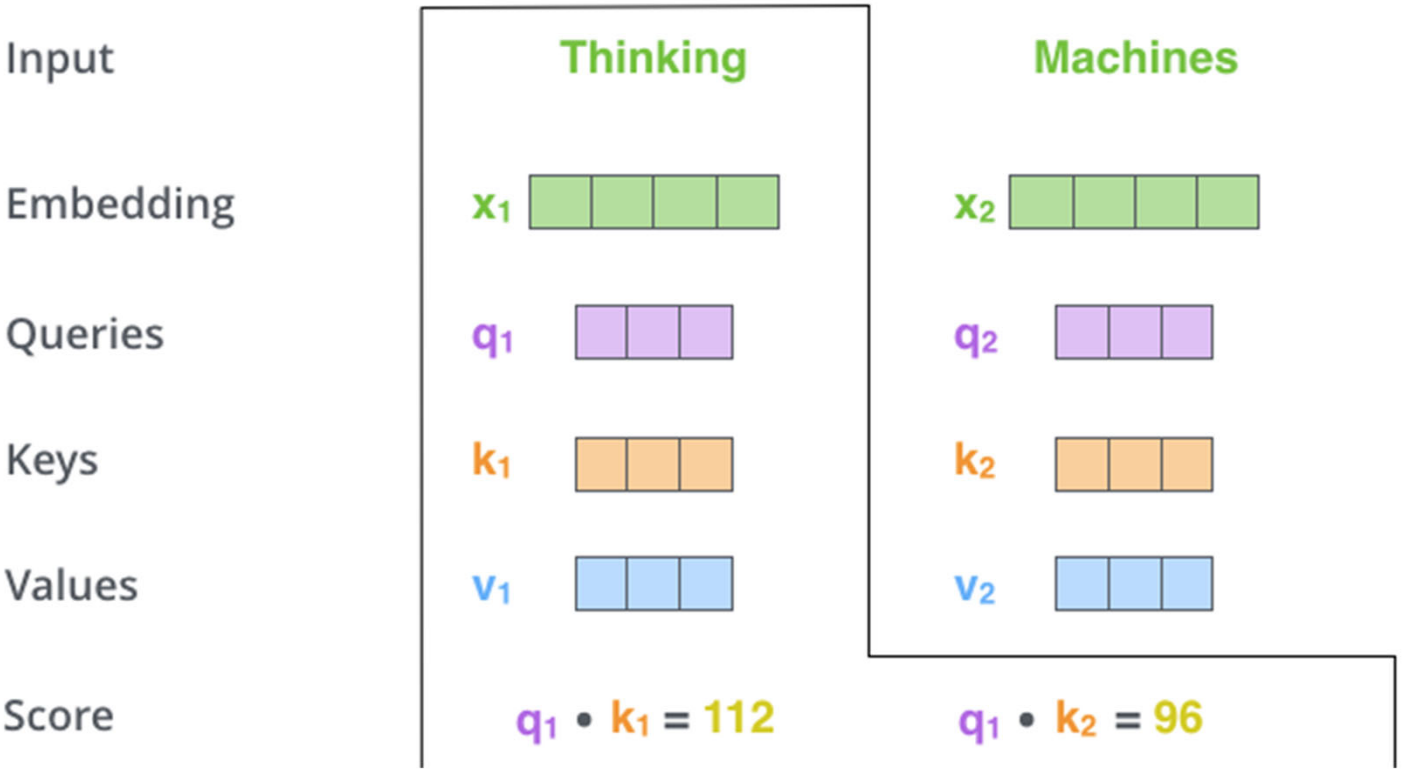
- $q_i = x_i W^Q$
- $K_i = x_i W^K$
- $V_i = x_i W^V$



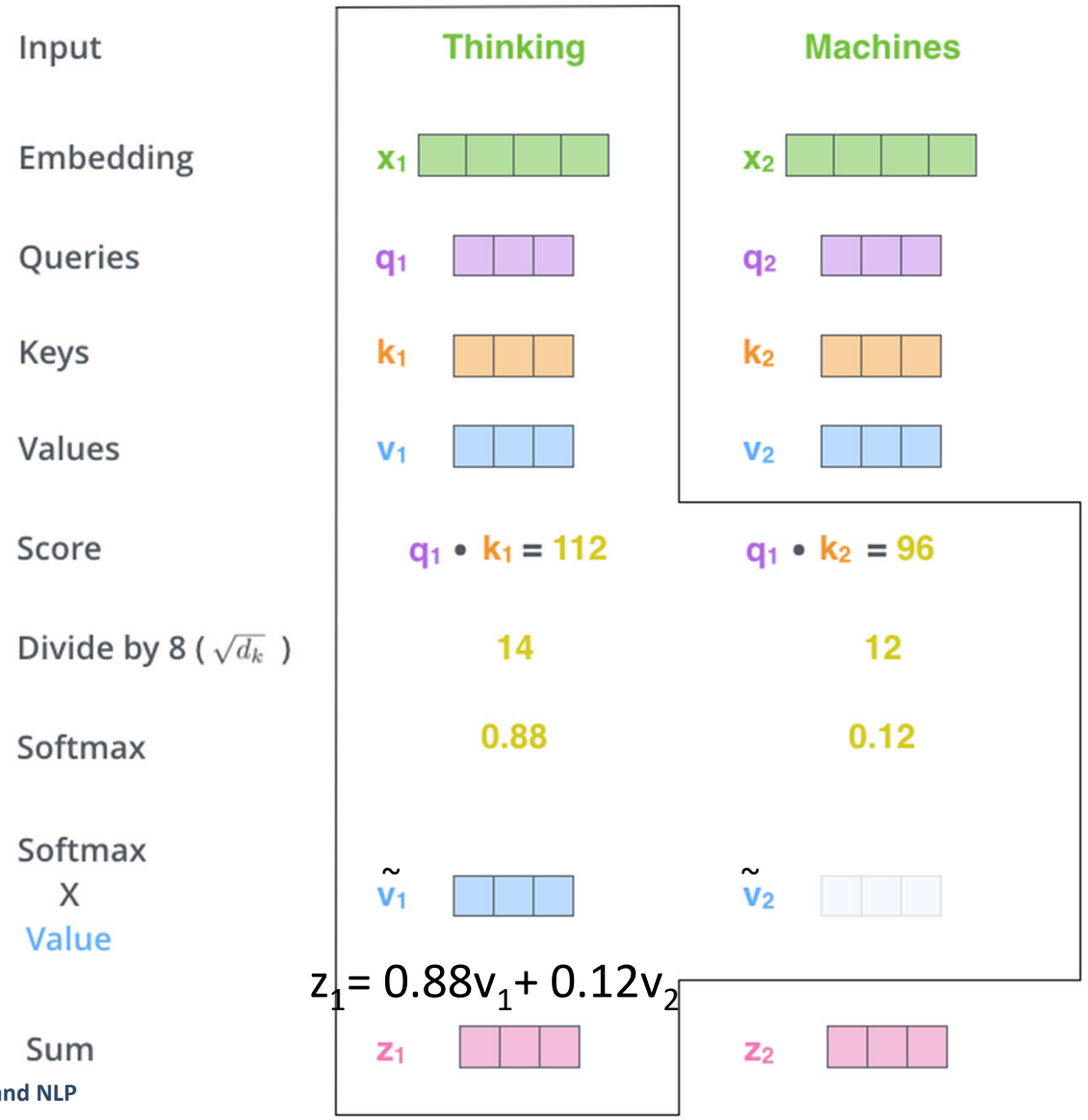
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Self Attention

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



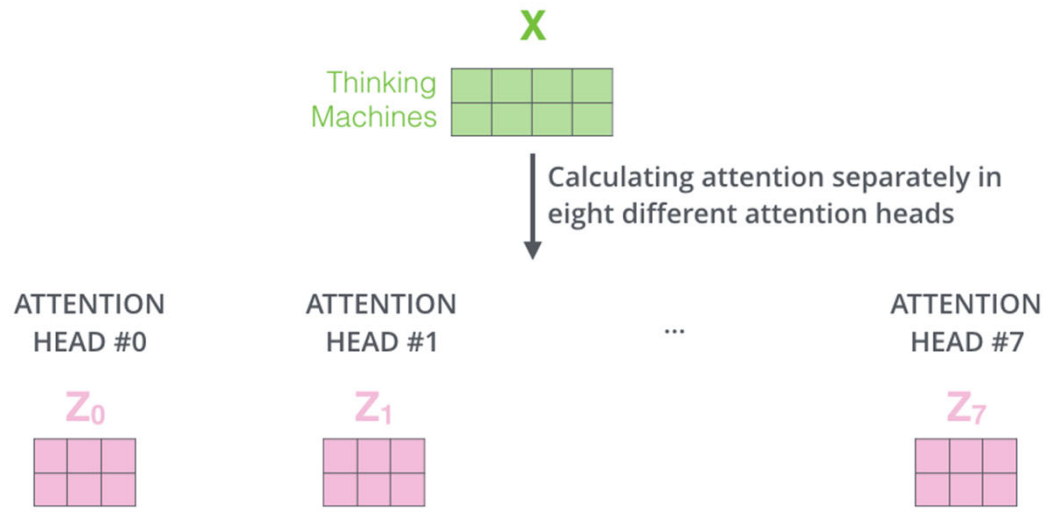
Self Attention



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"

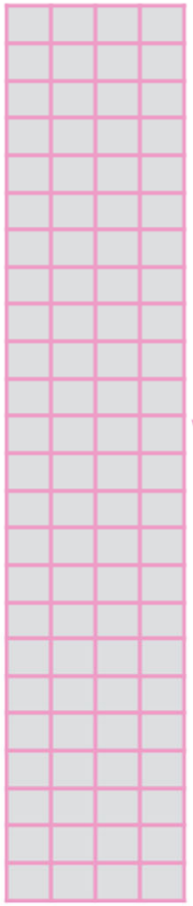


Attention and Transformers

1) Concatenate all the attention heads

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

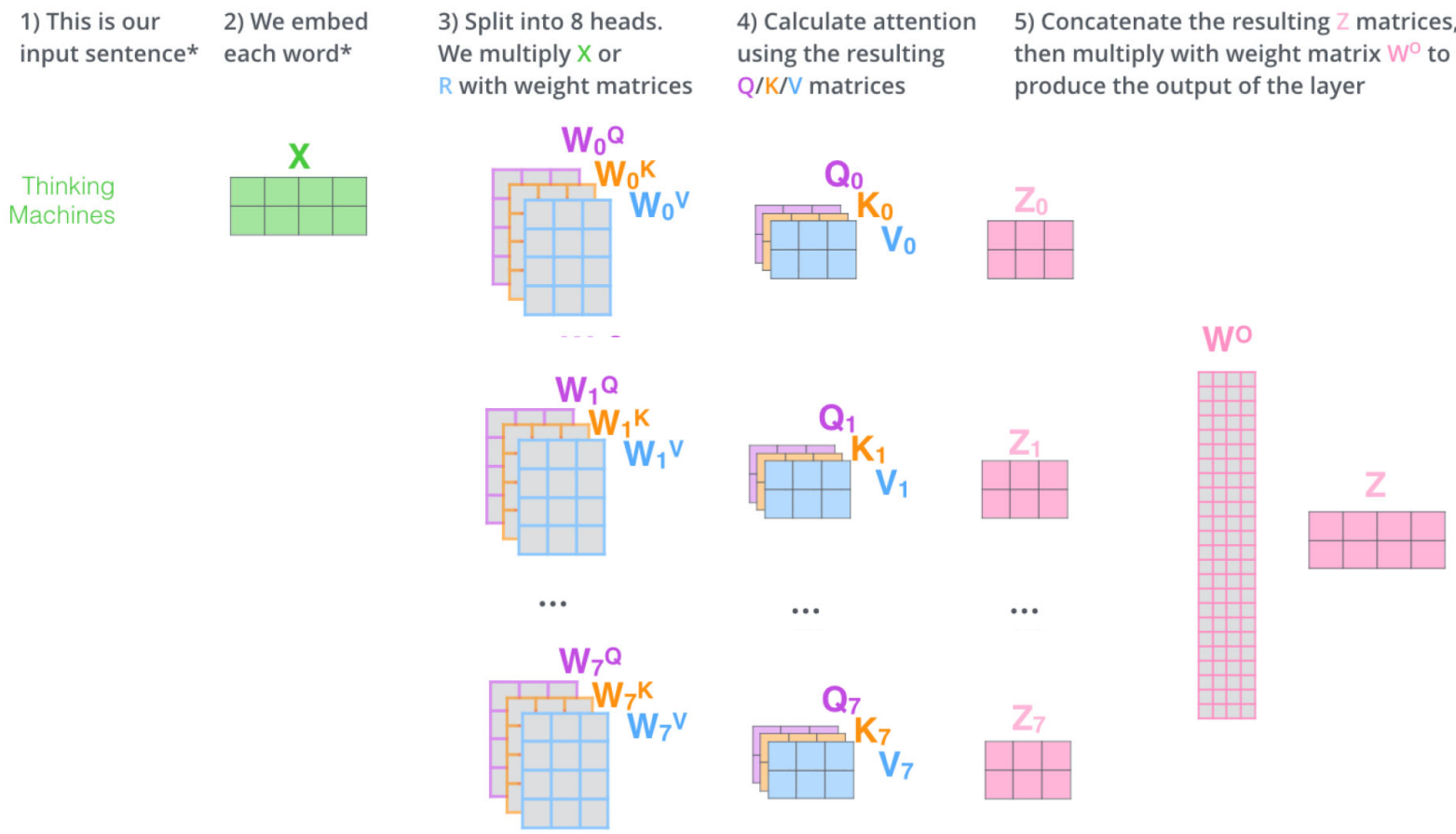
X



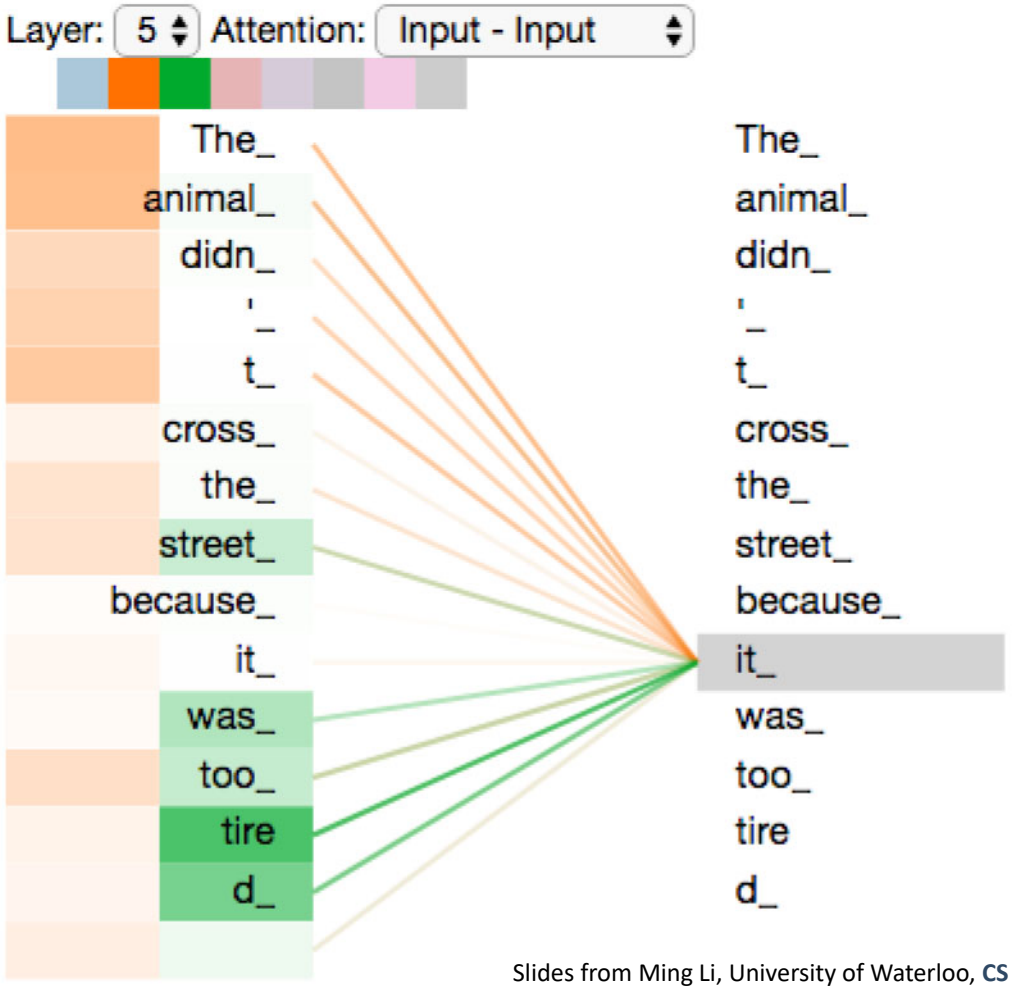
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=

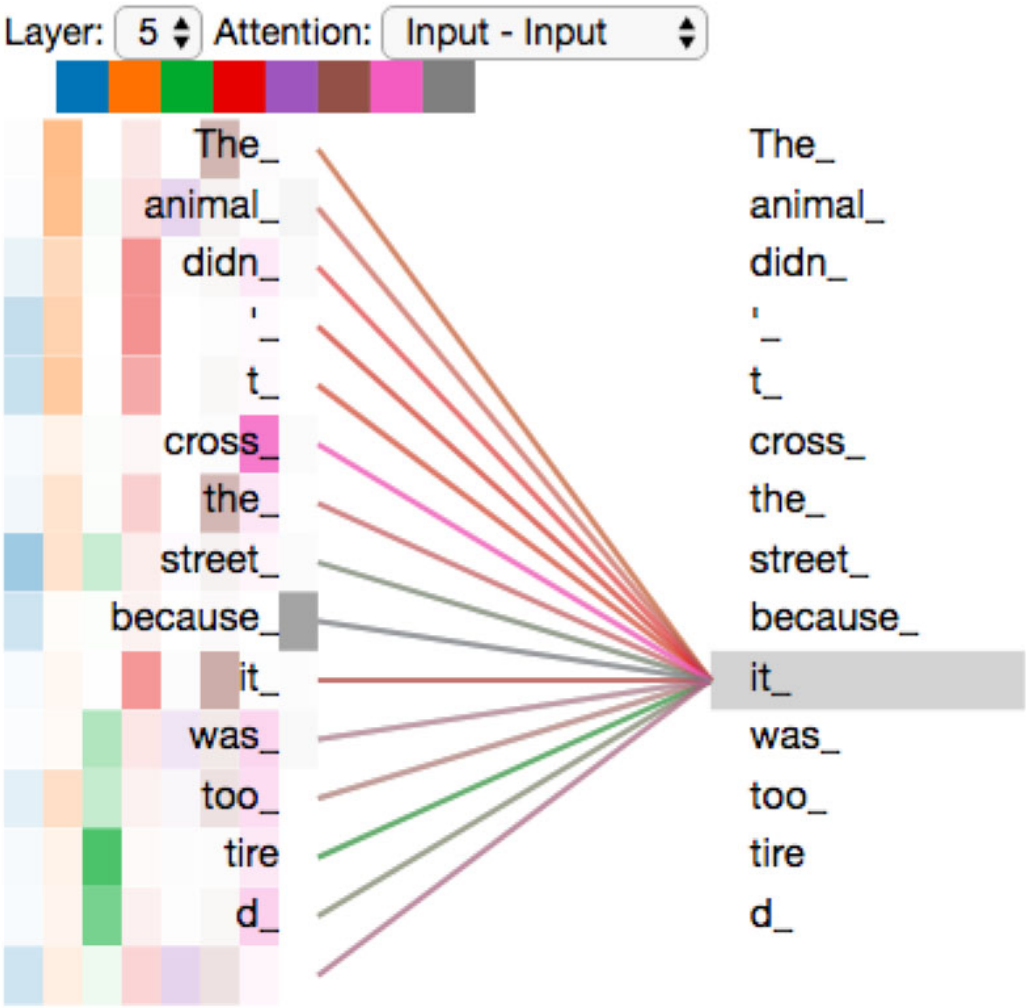
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**



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**

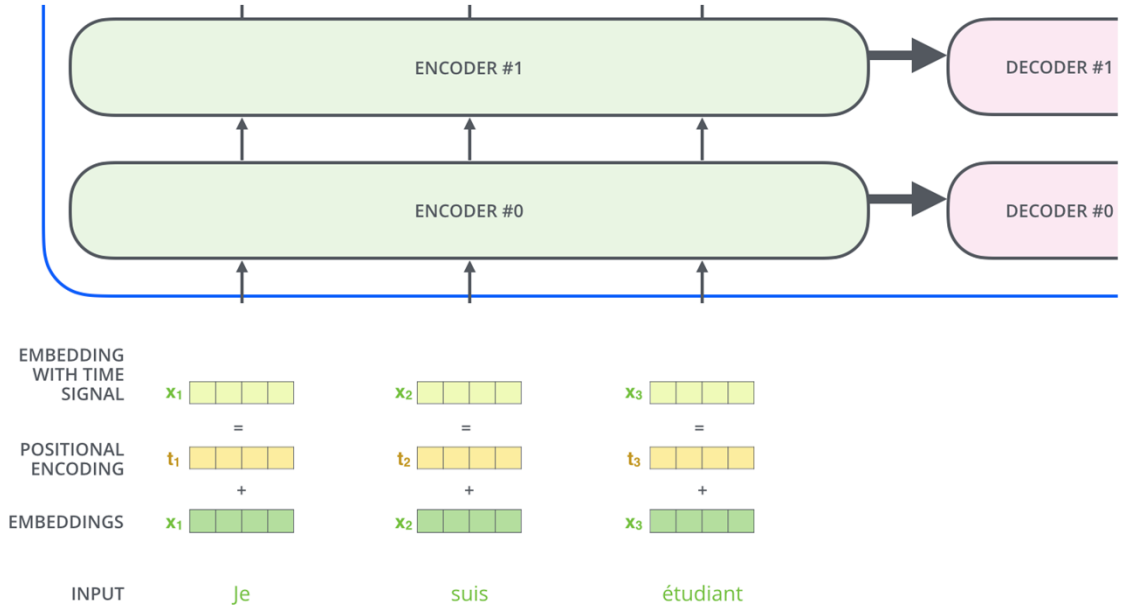


Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Representing the input order (positional encoding)

- Transformer is permutation invariant
- 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

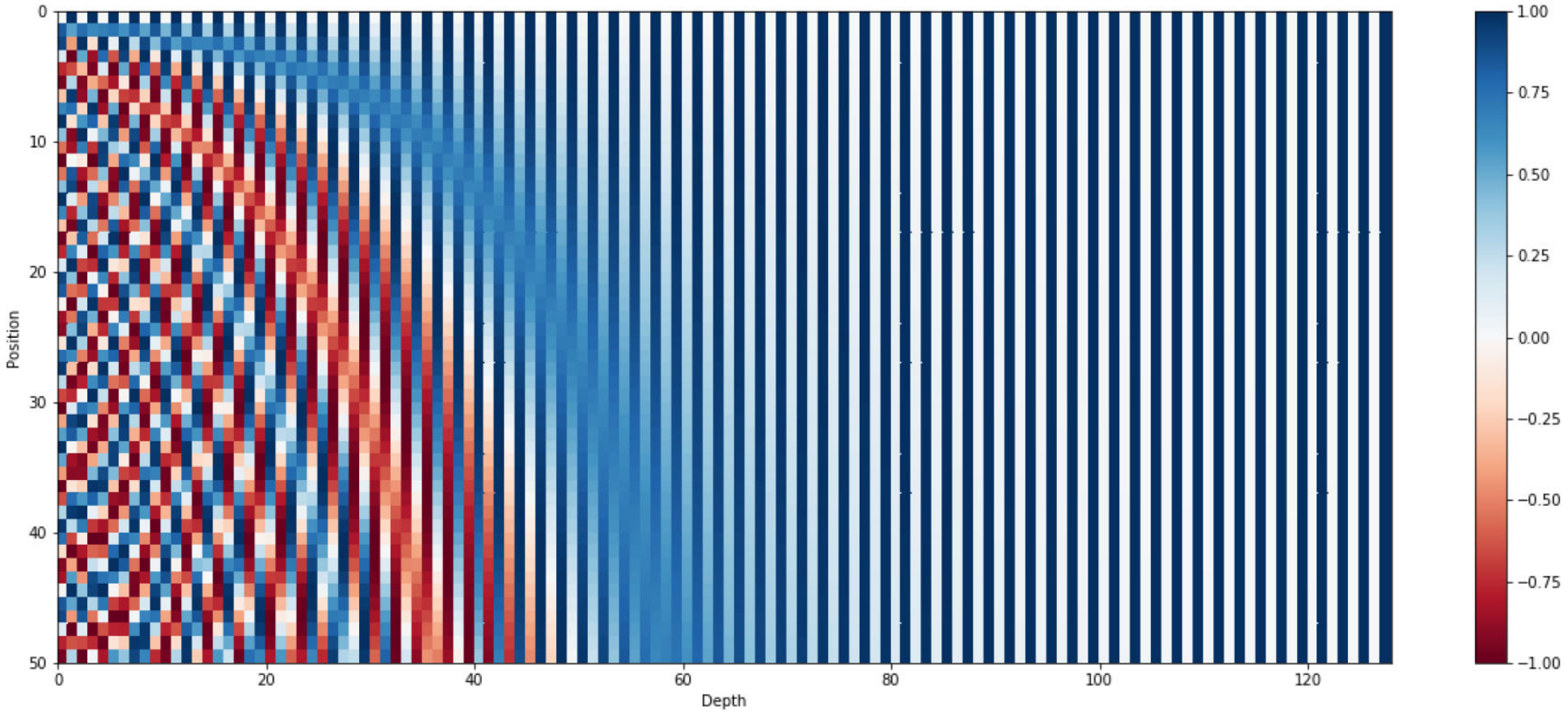
$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k+1 \end{cases}$$

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



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

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

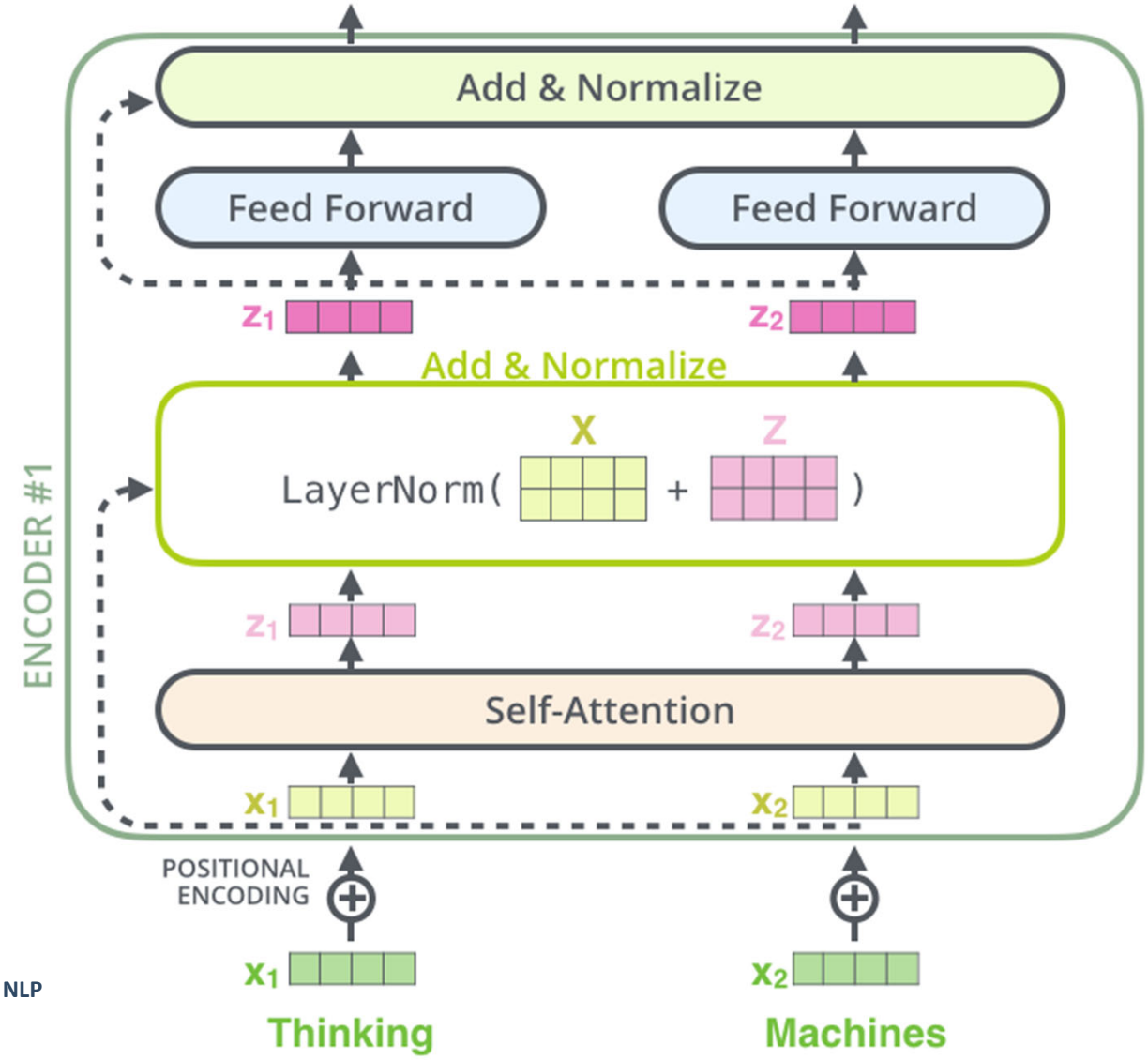
Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	$P_{00}=\sin(0)$ = 0	$P_{01}=\cos(0)$ = 1	$P_{02}=\sin(0)$ = 0	$P_{03}=\cos(0)$ = 1
am	1	$P_{10}=\sin(1/1)$ = 0.84	$P_{11}=\cos(1/1)$ = 0.54	$P_{12}=\sin(1/10)$ = 0.10	$P_{13}=\cos(1/10)$ = 1.0
a	2	$P_{20}=\sin(2/1)$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

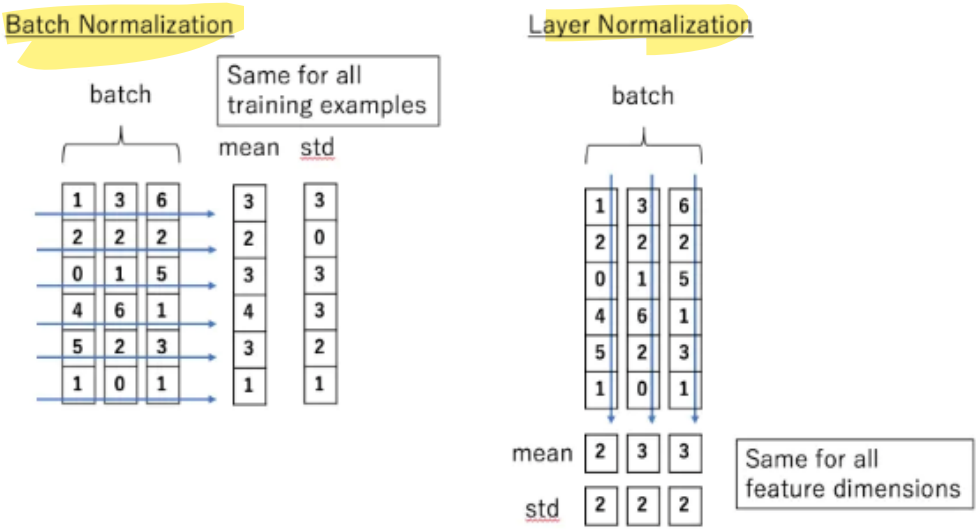
Add and Normalize

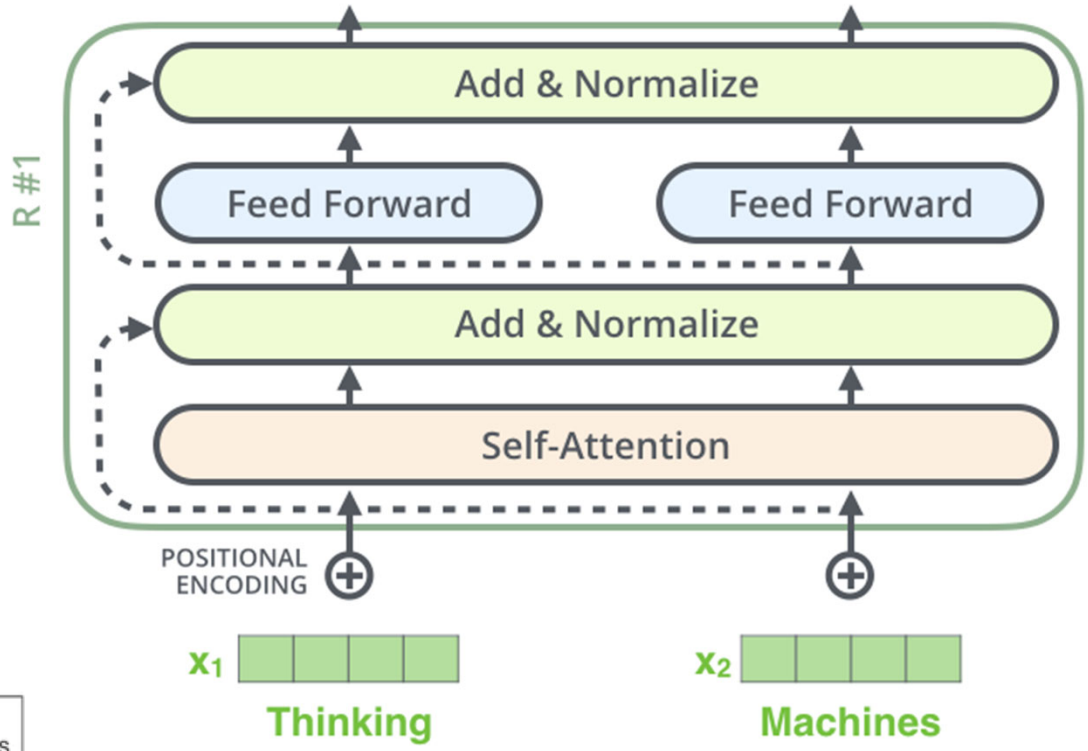


Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

Layer Normalization (Hinton)

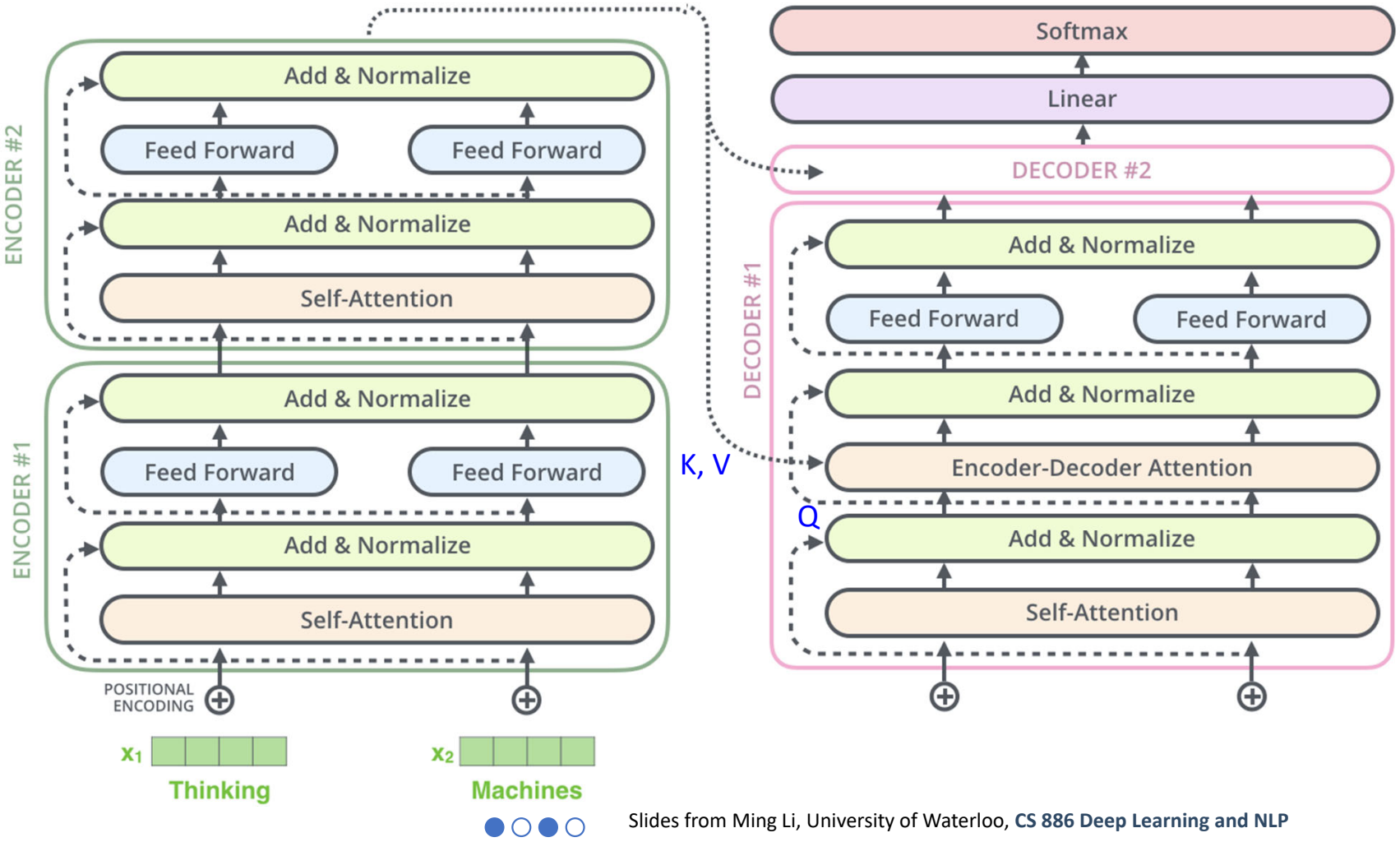
Layer normalization normalizes the inputs across the features.



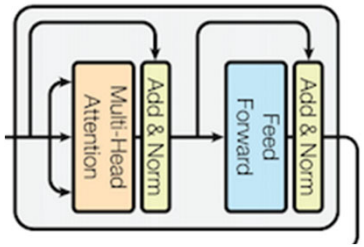


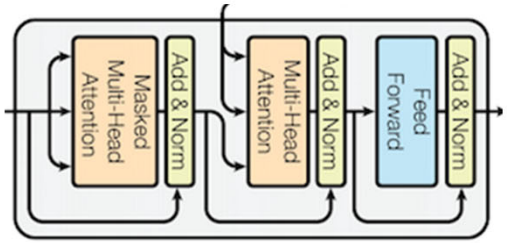
Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

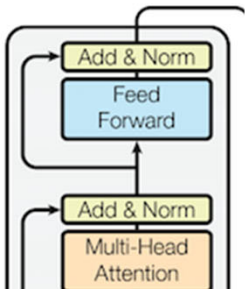
The complete transformer

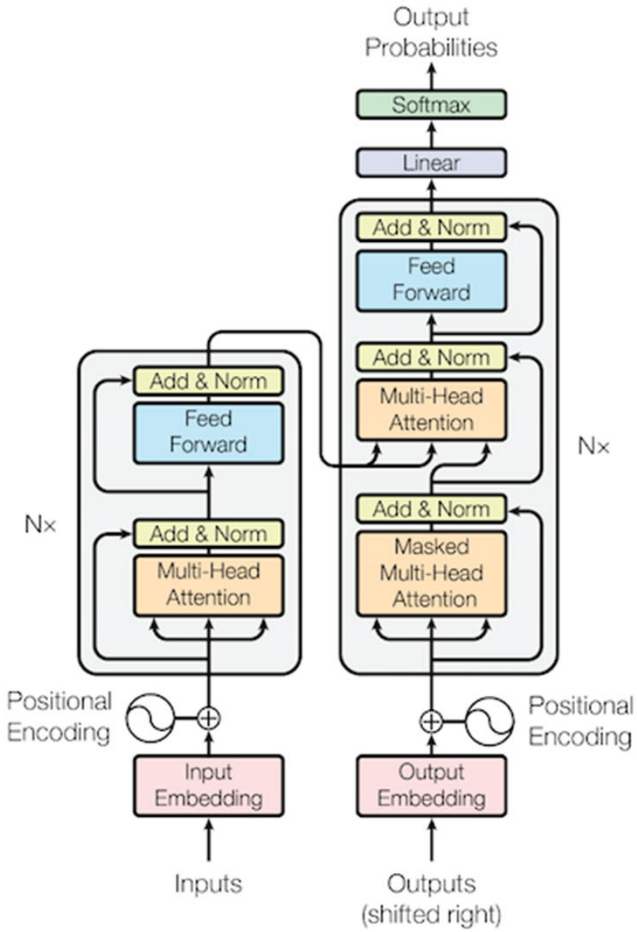


Slides from Ming Li, University of Waterloo, **CS 886 Deep Learning and NLP**

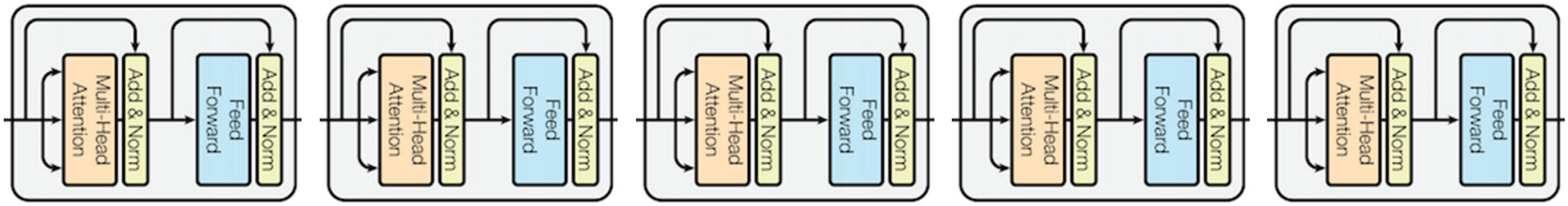

Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm


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


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


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

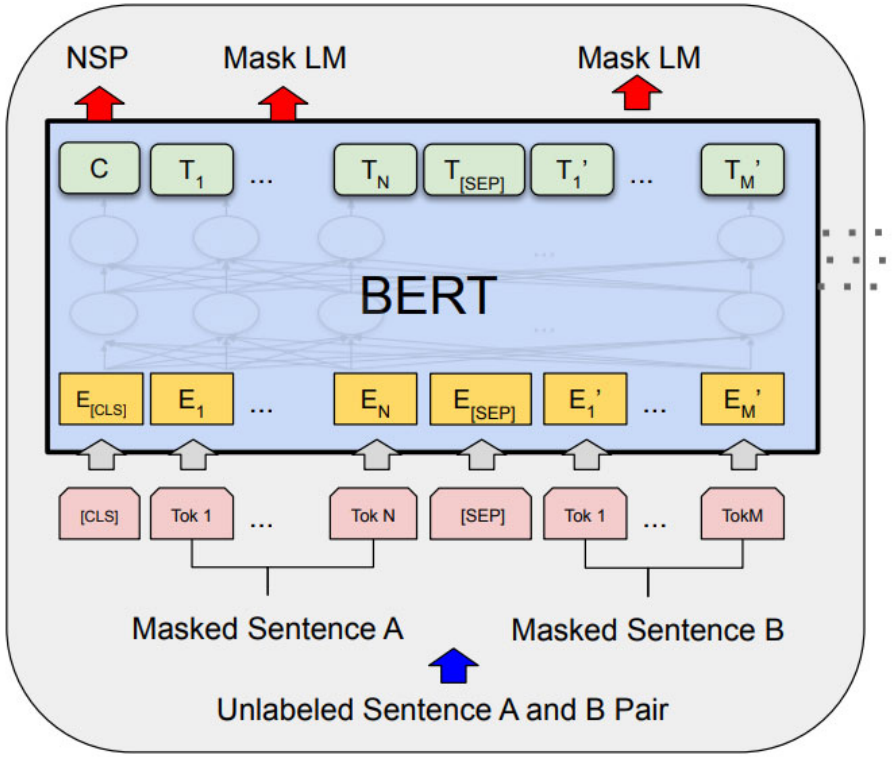
BERT (Stack Encoder Blocks)

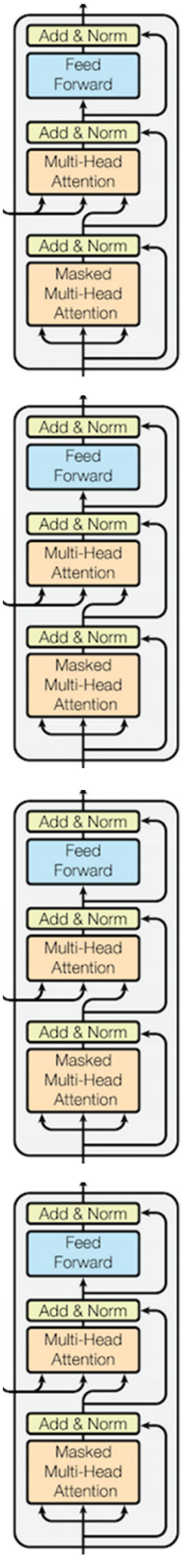


BERT (Bidirectional Encoder Representations from Transformers)

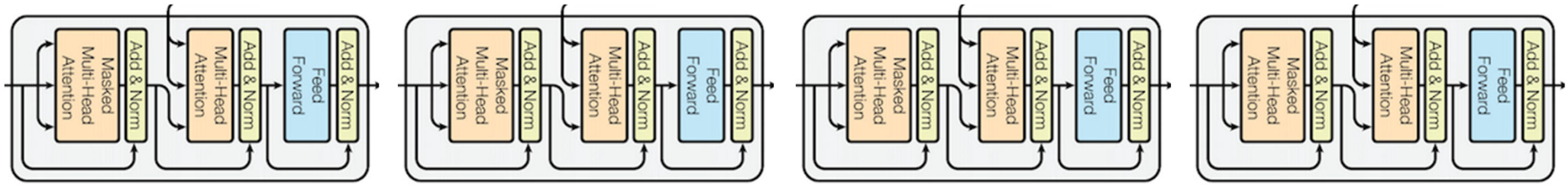
- BERT jointly encodes the right and left context of a word in a sentence to improve the learned feature representations
- BERT is trained on two pre-text tasks in self-supervised manner
  - Masked Language Model (MLM)
    - Mask fixed percentage (15%) of words in a sentence predict these masked words
    - 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


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

GPT (Stack Decoder Blocks)



BERT and GPT

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