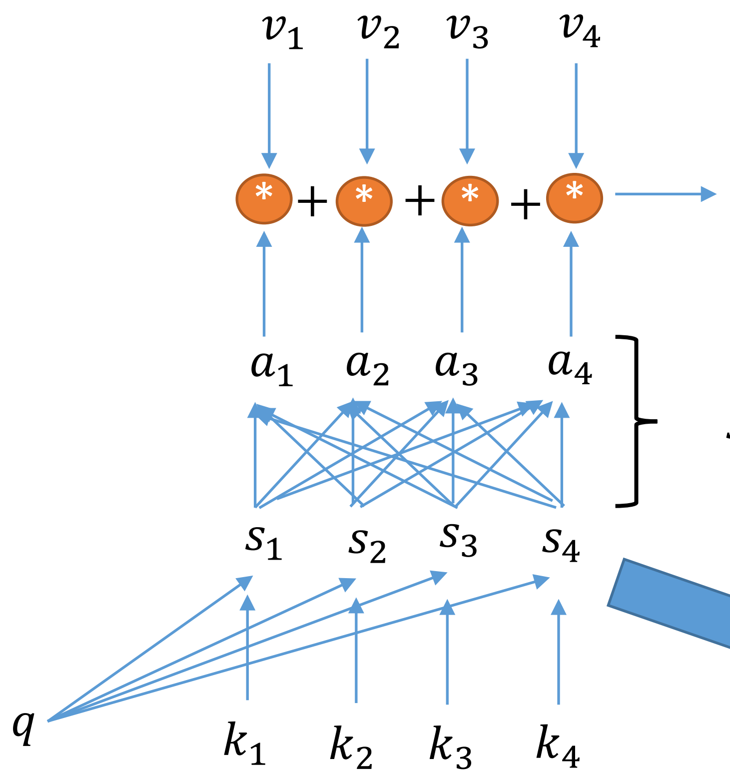
Lecture 10

Transformer

Neural architecture



$$\mathit{attention\ value} = \sum_i a_i v_i$$

$$\mathit{softmax}\colon\ a_i = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

$$s_i = \mathit{similarity}(k_i, q) = \begin{cases} q^T k_i \\ q^T k_i / \sqrt{p} \\ q^T W k_i \\ \vdots \end{cases}$$

(p is the dimensionality of k)

Matrix Forms:

- Words in sequence: $X = [x_1, ..., x_n] \in \mathbb{R}^{d \times n}$
- Queries: $Q = [q_1, ..., q_n] \in \mathbb{R}^{p \times n}$
- Keys: $K = [k_1, ..., k_n] \in \mathbb{R}^{p \times n}$
- Values: $V = [v_1, ..., v_n] \in \mathbb{R}^{m \times n}$

Projection:

- Queries: $q_i = W_Q^T x_i$
- Keys: $k_i = W_K^T x_i$
- Values: $v_i = W_V^T x_i$

Matrix Form

Similarity Measures:

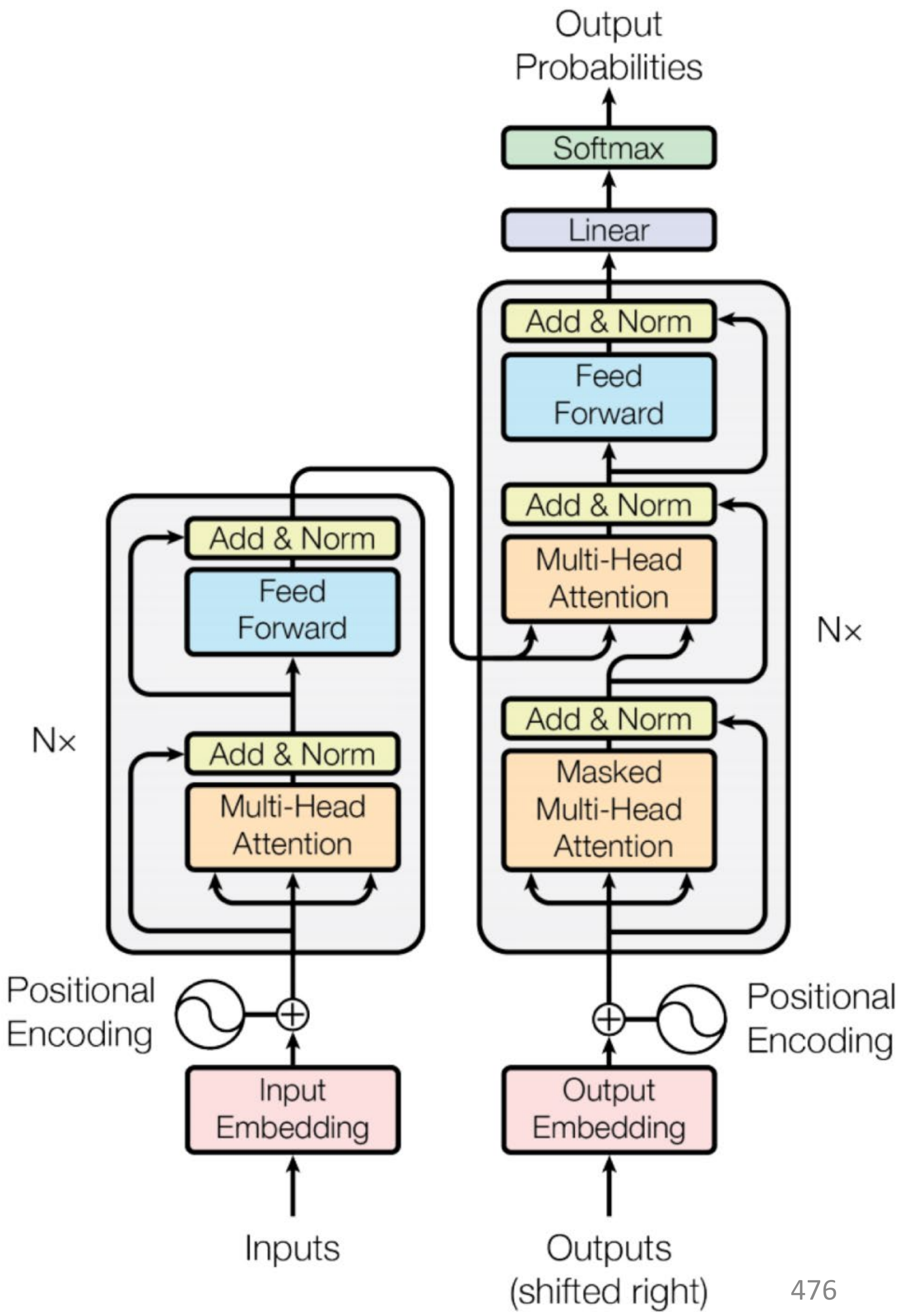
- Inner product: $q^T k_i = x_i^T W_Q (W_K)^T x_i$
- Acts like a kernel matrix, measuring similarity.

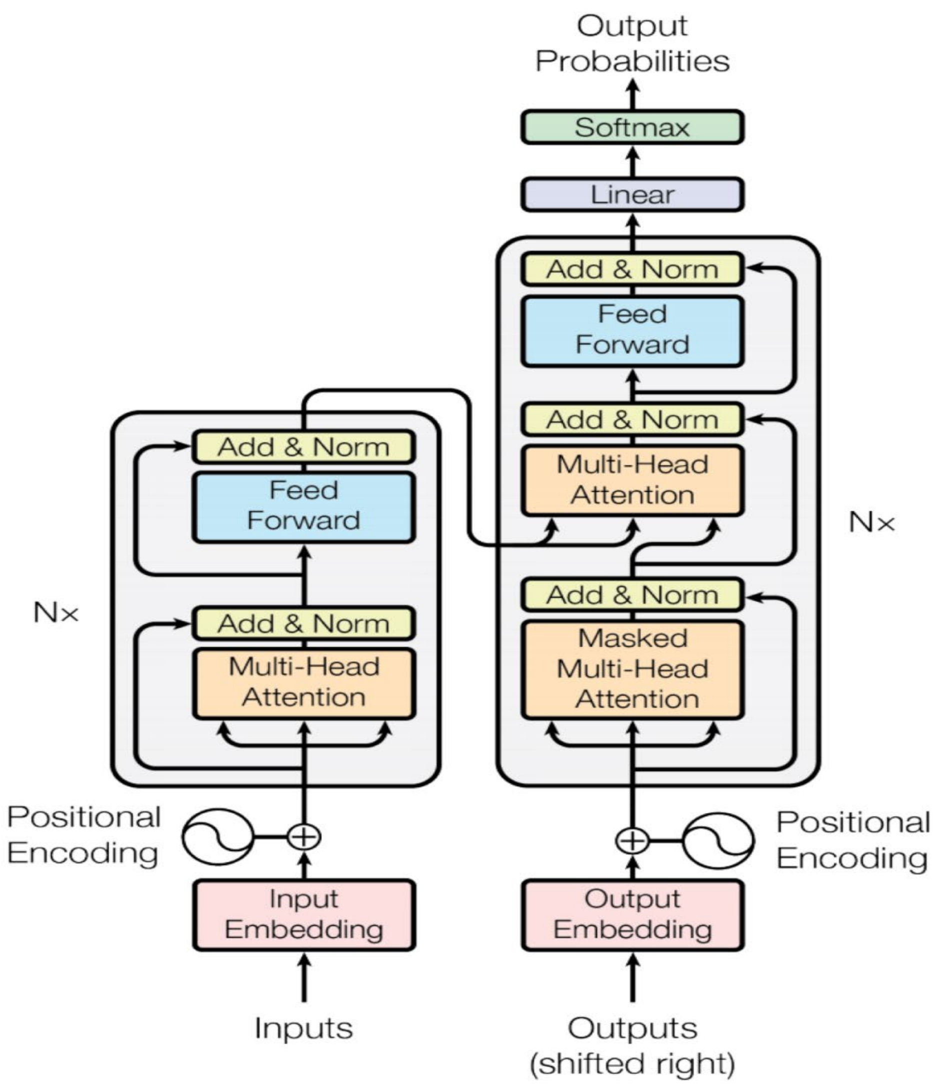
Attention Computation:

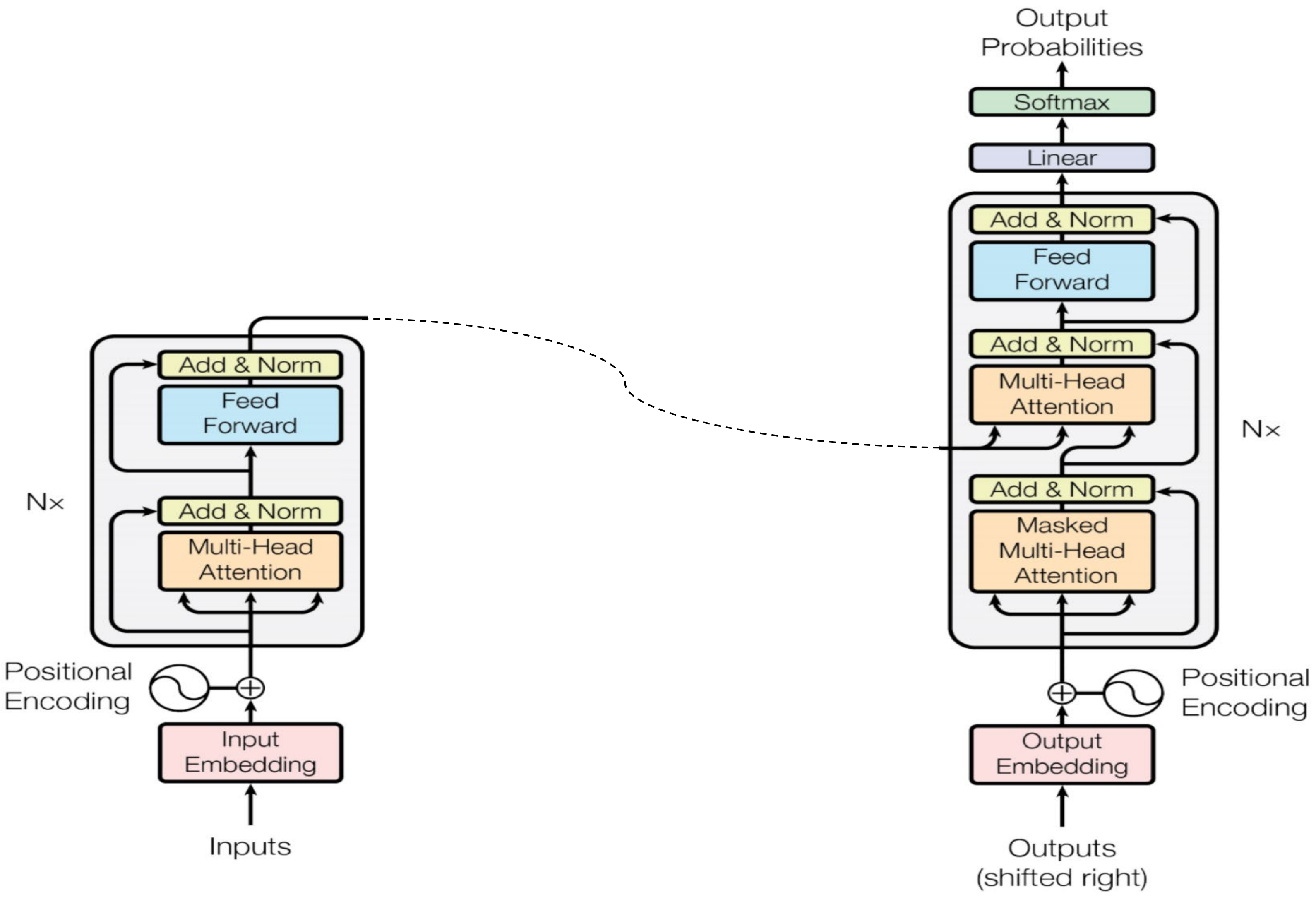
- $Z := \text{attention}(Q, K, V) = V\text{softmax}\left(\frac{1}{\sqrt{p}} Q^T K\right)$

Transformer

Attention Is All You Need (2017)




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)

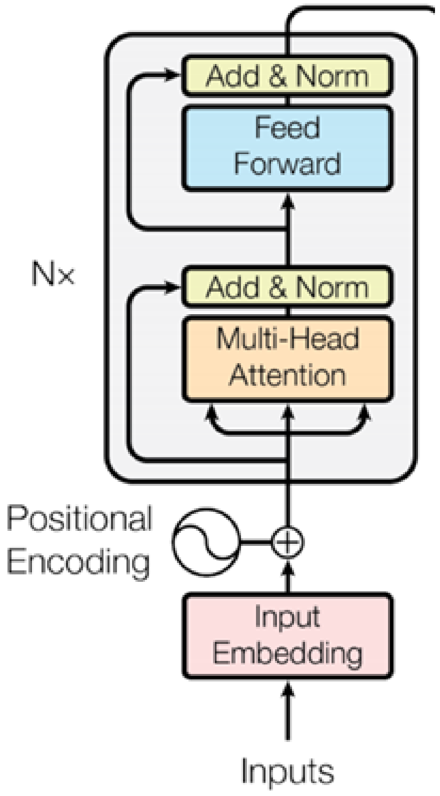


Encoder

Decoder

Encoder

The encoder part of the transformer embeds the input sequence of n words $X \in \mathbb{R}^{d \times n}$ into context vectors with the attention mechanism.



Encoder

- The encoder consists of two main components: Self-Attention and Feedforward Neural Network (FFN).
- **Self-Attention:**
  - Input: Matrix X
  - Linear Transformations to generate Query (Q), Key (K), and Value (V) matrices:

$$Q = W_Q^T X, \quad K = W_K^T X, \quad V = W_V^T X$$

  - Compute attention output Z using the formula:

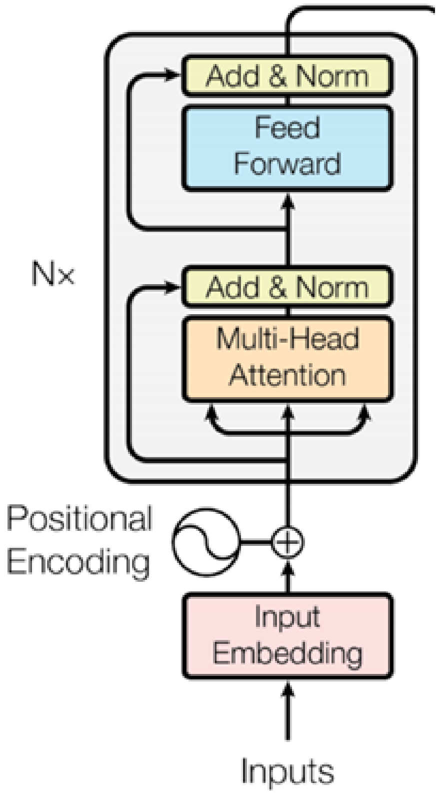
$$Z = V\text{softmax}\left(\frac{Q^T K}{\sqrt{p}}\right)$$

  - Residual Connection:

$$X + Z$$

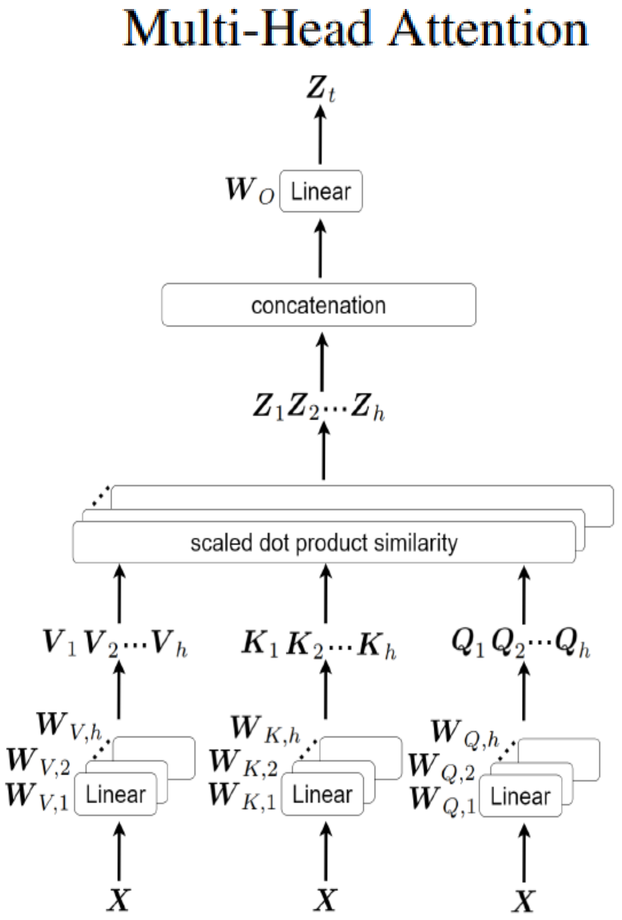
  - Normalization:

$$(X + Z)$$



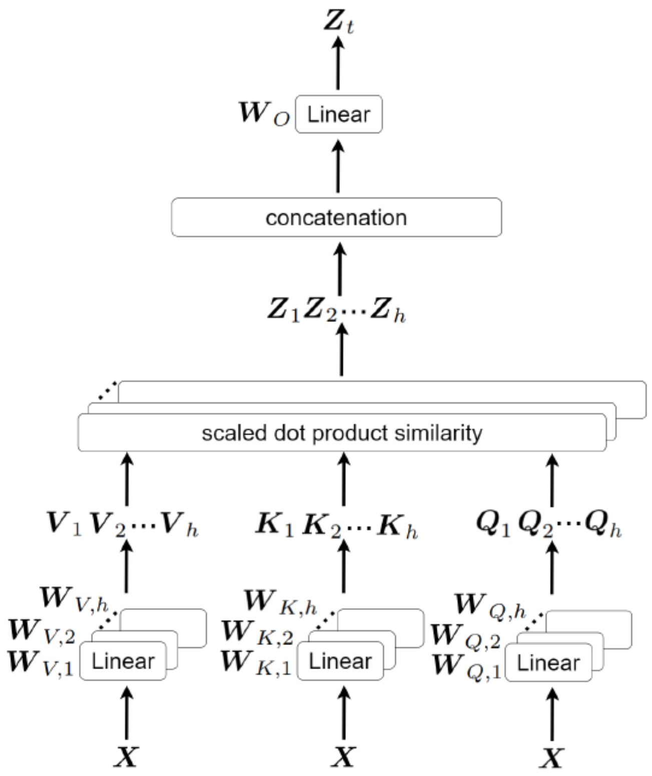
Multi-Headed Attention

$$Q_1 = {W_Q^1}^T X$$
$$K_1 = {W_K^1}^T X$$
$$V_1 = {W_V^1}^T X$$
$$Z_1 = V \mathrm{softmax}\left(\frac{1}{\sqrt{p}} Q_1^T K_1\right)$$
$$\vdots$$
$$Q_h = {W_Q^h}^T X$$
$$K_h = {W_K^h}^T X$$
$$V_h = {W_V^h}^T X$$
$$Z_h = V \mathrm{softmax}\left(\frac{1}{\sqrt{p}} Q_h^T K_h\right)$$



Multi-Headed Attention

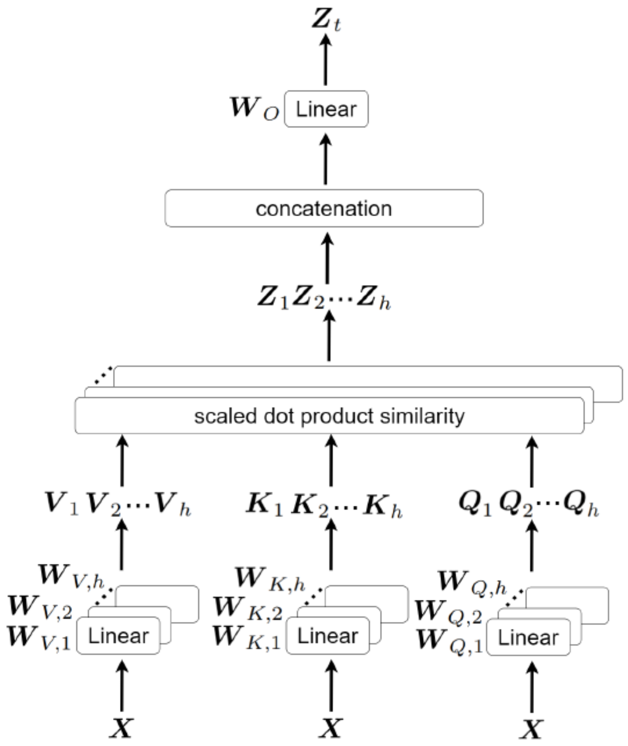
$$\begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_h \end{pmatrix} = \text{Concat}(head_1, \ldots, head_h)$$



Multi-Headed Attention

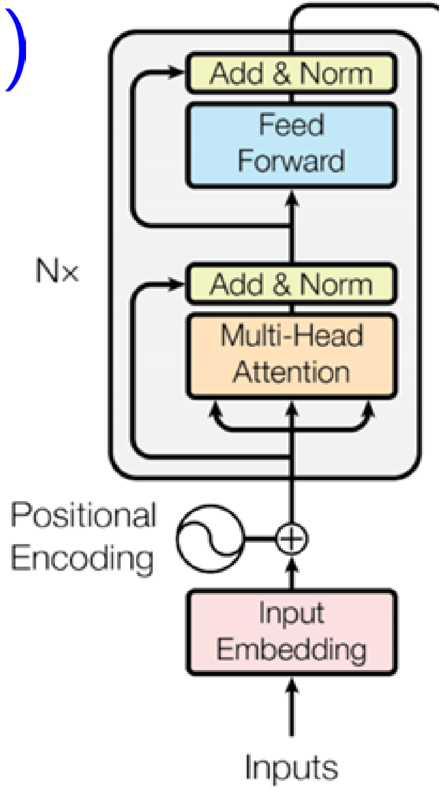
$$\begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_h \end{pmatrix} = \mathrm{Concat}(head_1, \ldots, head_h)$$

$$Z = \mathrm{MultiHead}(Q, K, V) = {W_0}^T \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_h \end{pmatrix}$$



Multi-Headed Attention

$$\text{MultiHead}(Q, K, V) = {W_O}^T \text{Concat}(head_1, \ldots, head_h)$$



Structure of the Feed Forward Network

- Linear Layer 1
- ReLU Activation
- Linear Layer 2

$$FFN(x) = W_2^T \max(0, W_1^T X + b_1) + b_2$$

Two linear transformations with ReLU activation in between.

Application of FFN to Each Position

- The Feed Forward Network (FFN) is applied independently to each position in the input sequence.
- Despite individual processing, all positions share the same set of weights and biases in the FFN.
- Key Points:
  - Shared parameters ensure consistency in processing across all positions.
  - Enables the model to generalize learnings from one position to all positions.
  - Facilitates parallel processing of the sequence, enhancing computational efficiency.

Global vs Local

- Attention Mechanism:
- **Global Understanding**: Captures relationships among different positions in the sequence.
- **Context Aggregation**: Spreads relevant information across the sequence, enabling each position to see a broader context.

Global vs Local

- Attention Mechanism:
- **Global Understanding**: Captures relationships among different positions in the sequence.
- **Context Aggregation**: Spreads relevant information across the sequence, enabling each position to see a broader context.
- Feed-Forward Networks (FFN):
- **Local Processing**: While attention looks across the entire sequence, FFN zooms back in to process each position independently.
- **Individual Refinement**: Enhances the representation of each position based on its own value, refining the information gathered so far.

A Classroom Analogy

- Attention Mechanism - Classroom Discussion:
- **Interactions:** Students (positions) in a classroom engaging in a discussion, sharing ideas, and interacting.
- Teacher's Role: The teacher (attention mechanism) observes who is interacting with whom, gaining a global understanding of the discussion dynamics.

A Classroom Analogy

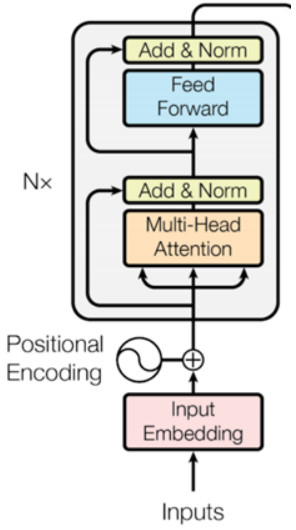
- Feed-Forward Network - Individual Assessment:
- **Teacher's Role**: The teacher (FFN) interacts with each student (position) independently, assessing their understanding and knowledge.
- **Independent Processing**: Each student is evaluated individually, akin to how the FFN processes each position independently.
- **Outcome**: Enhanced understanding and refined representation of each student's performance, akin to the FFN refining representations at each position.

A Classroom Analogy

- Synergy of Attention and FFN:
- **Holistic Understanding**: The combination of global interaction observation (attention) and individual assessment (FFN) provides a holistic understanding of both group dynamics and individual performances.
- **Balanced Processing**: A balanced approach to processing global relationships and local, position-specific information, leading to richer representations and enhanced learning.

Encoder

If the output of the FFN is denoted by R, then a residual connection is established from the output of the previous layer $(X + Z)$ to the output of the FFN, resulting in $(X + Z) + R$. This will be normalized $((X + Z) + R)$ to form the output of the encoder.



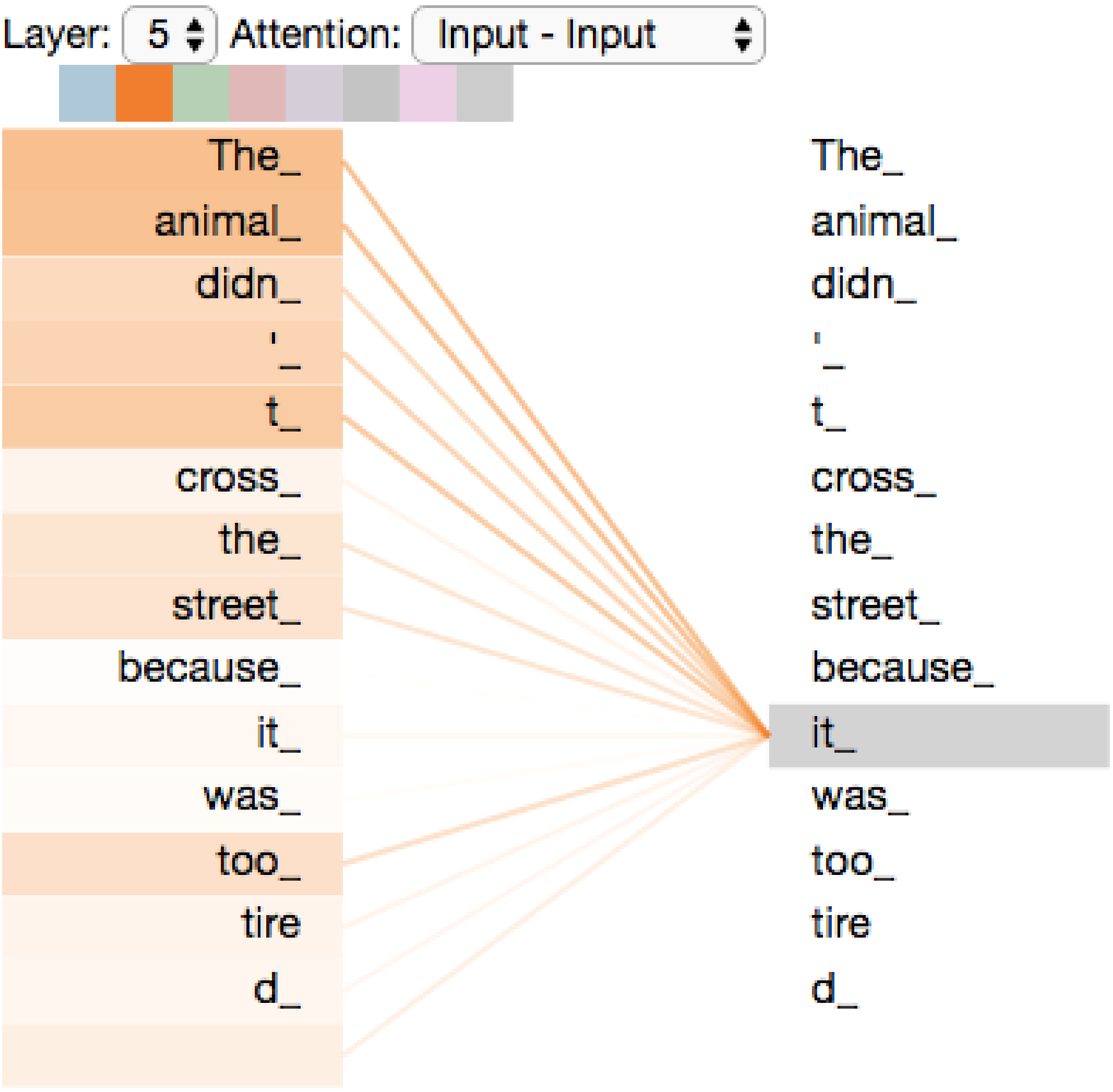


Figure: Jay Alammarz

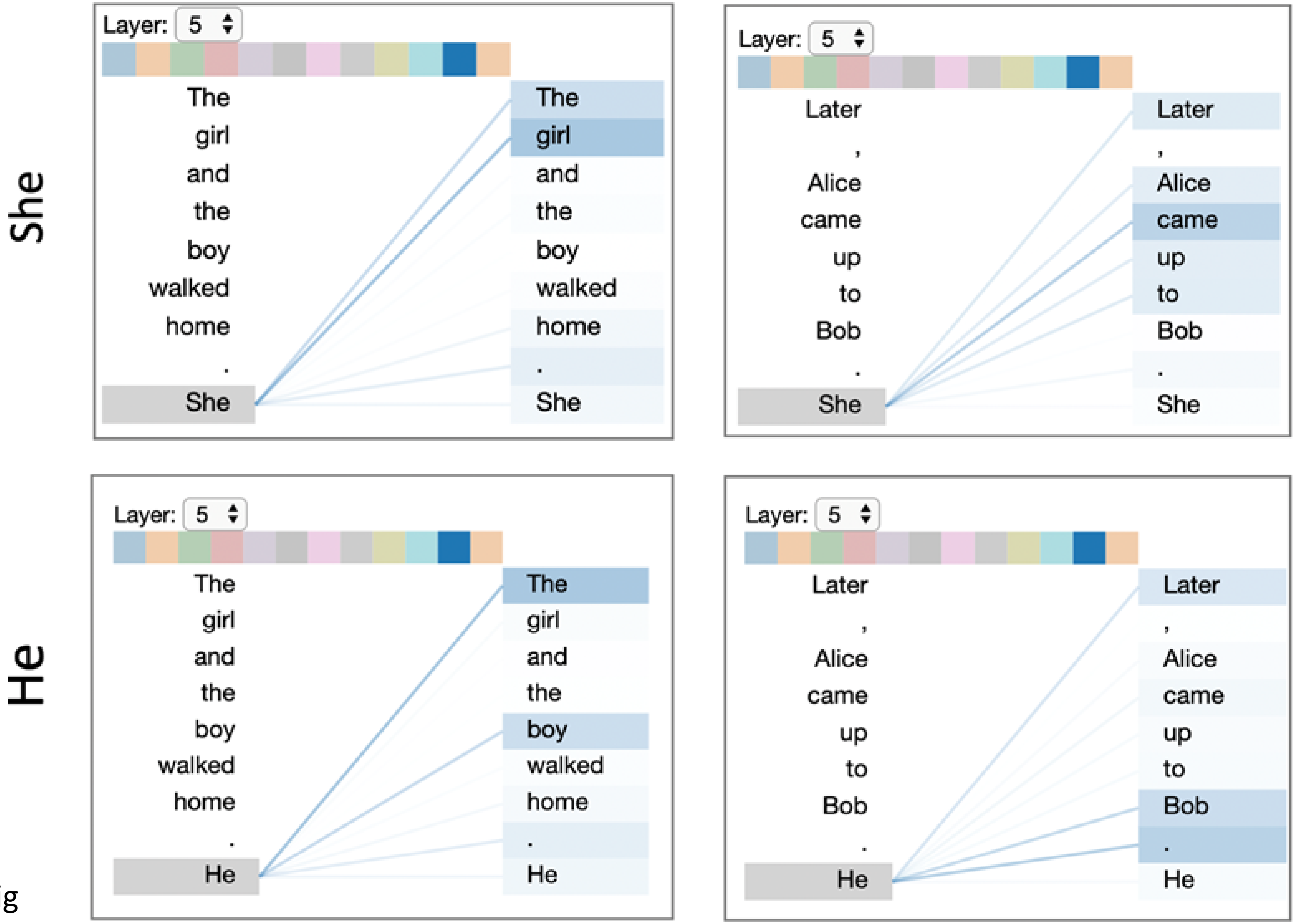
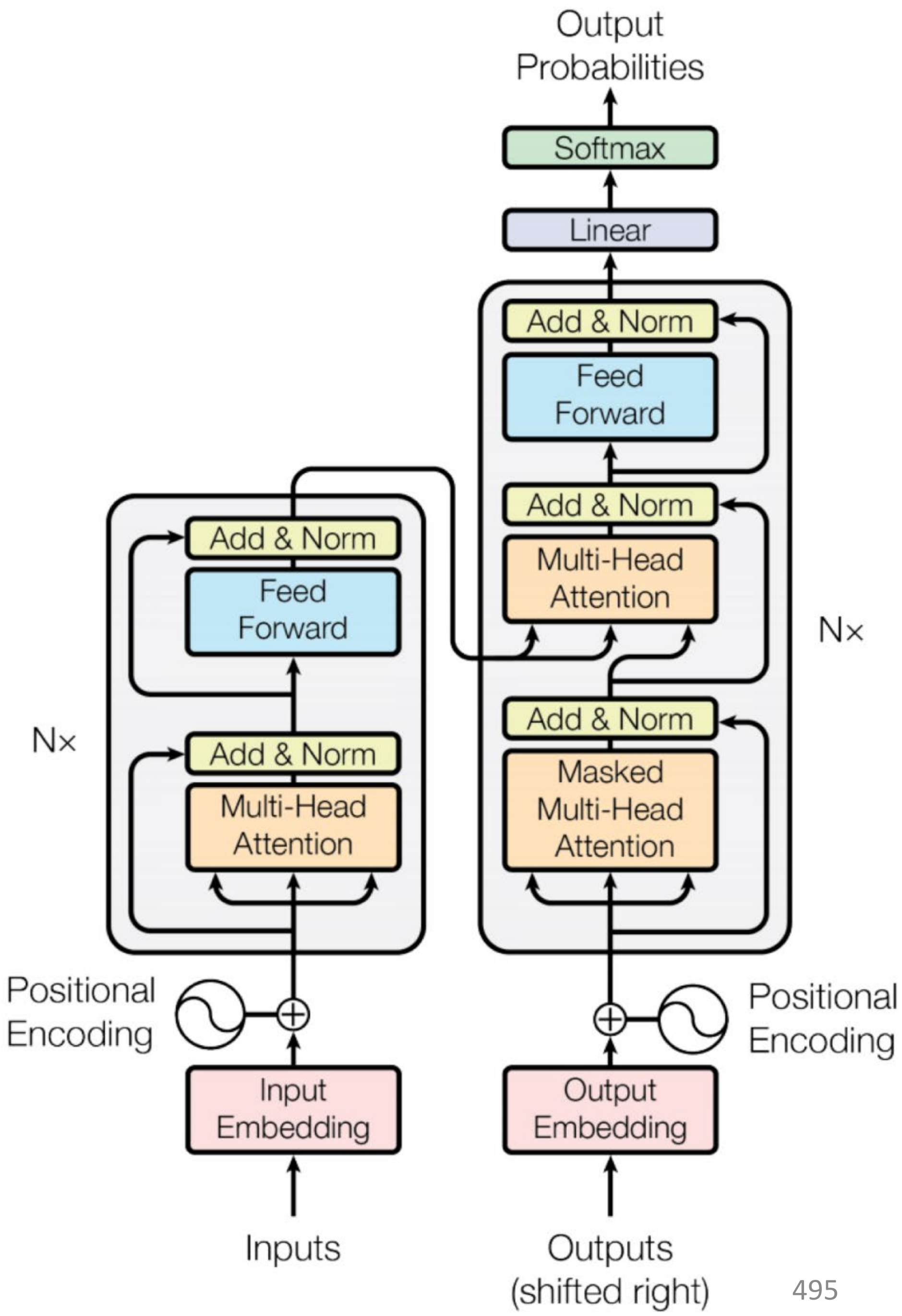


Figure: Jesse Vig

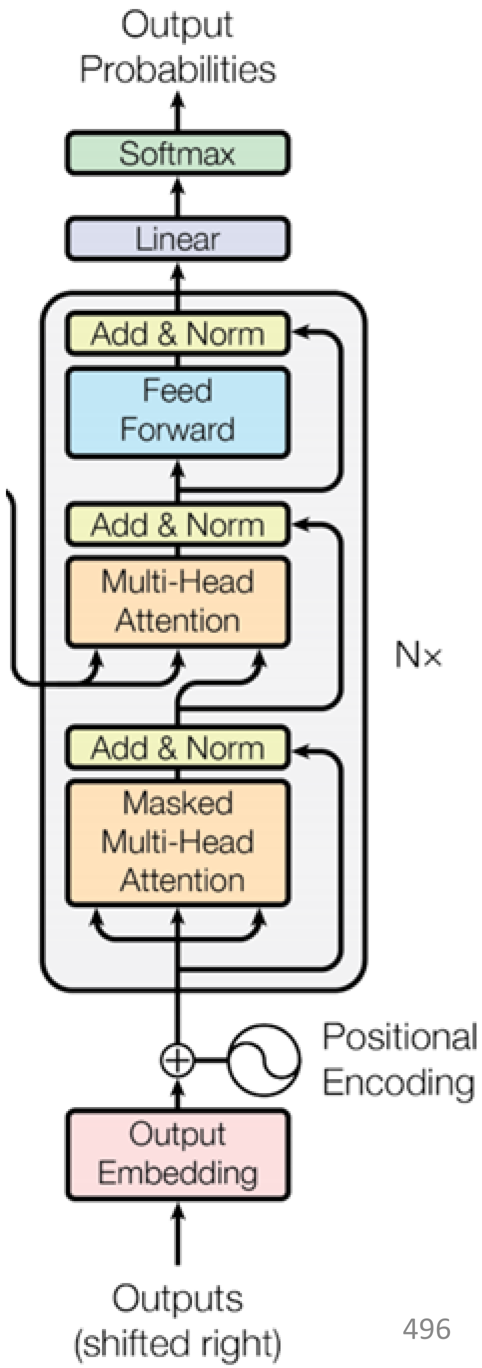
Transformer

Attention Is All You Need (2017)



Decoder

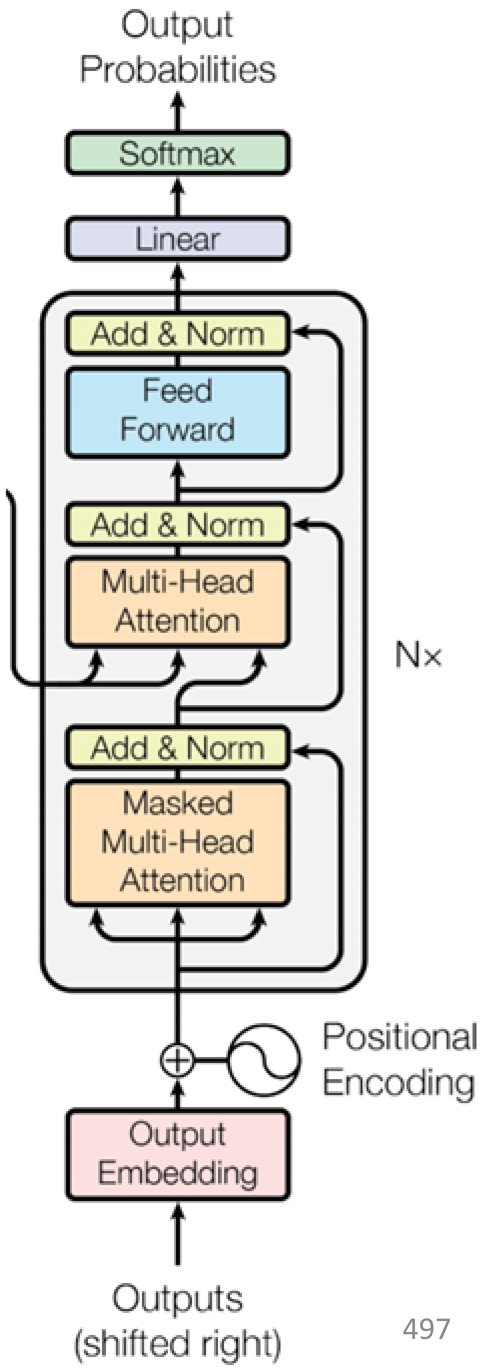
- **Masked self attention**.



Decoder

- **Masked self attention.**

$$Z := \text{Attention}(Q, K, V) = V\text{softmax}\left(\frac{1}{\sqrt{p}}(Q^\top K)\right),$$



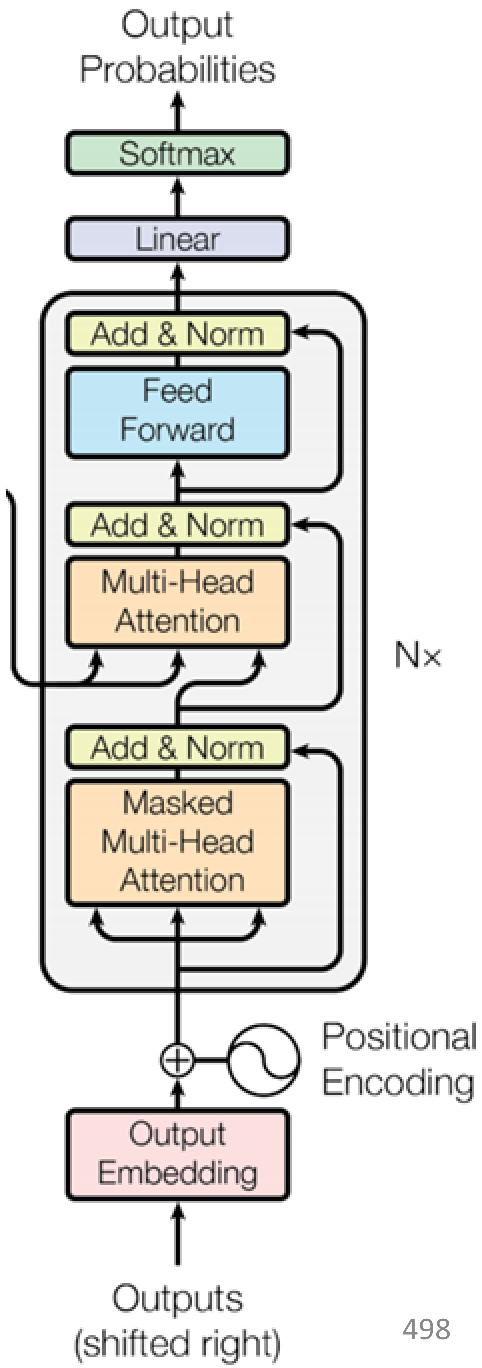
Decoder

- **Masked self attention**.

$$Z := \text{maskedAttention}(Q, K, V) = V\text{softmax}\left(\frac{1}{\sqrt{p}}(Q^\top K + M)\right),$$

where the mask matrix $M \in \mathbb{R}^{n\times n}$ is:

$$M(i,j) := \begin{cases} 0 & \text{if } j \le i, \\ -\infty & \text{if } j > i. \end{cases}$$

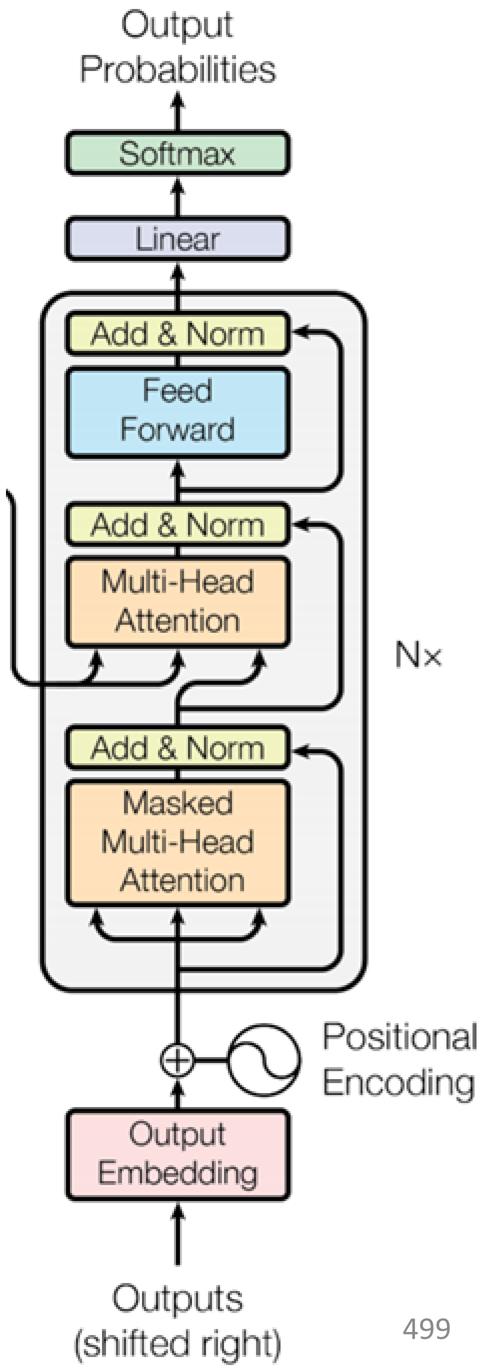


Decoder

- **Cross Attention**
- Cross attention allows each position in one sequence to attend over all positions in another sequence.

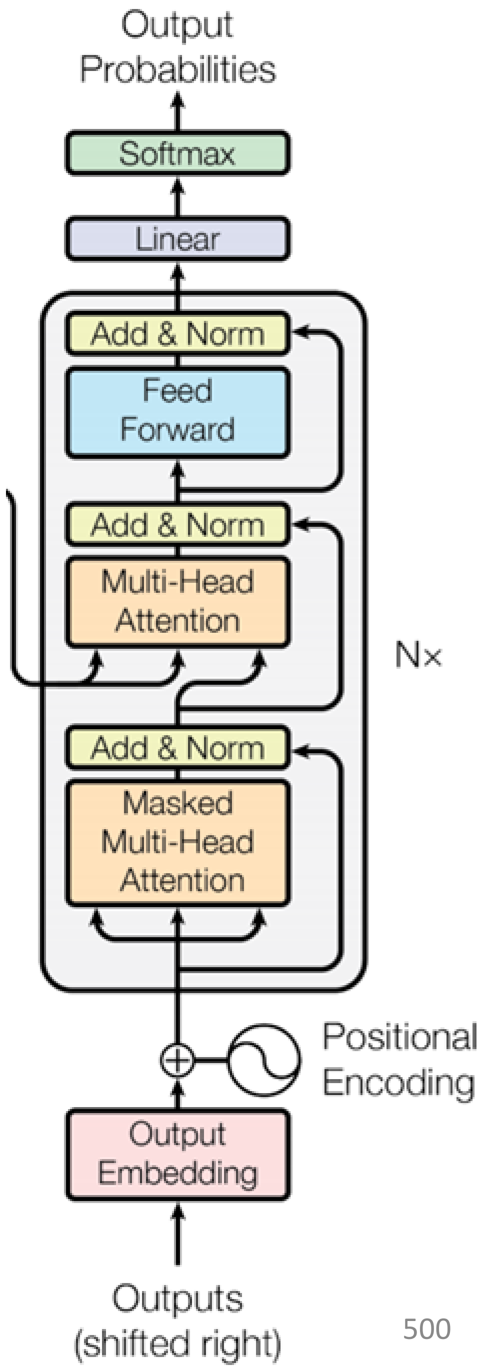
•**Query (Q)**: Originates from a position in the first sequence, i.e. the output of a previous layer in the decoder.

•**Memory Keys (K) and Values (V)**: Both come from all positions in the second sequence, i.e. the output of the encoder.



Decoder

- Masked self attention.
- Cross attention layer is like what attention does in sequence-to-sequence models.

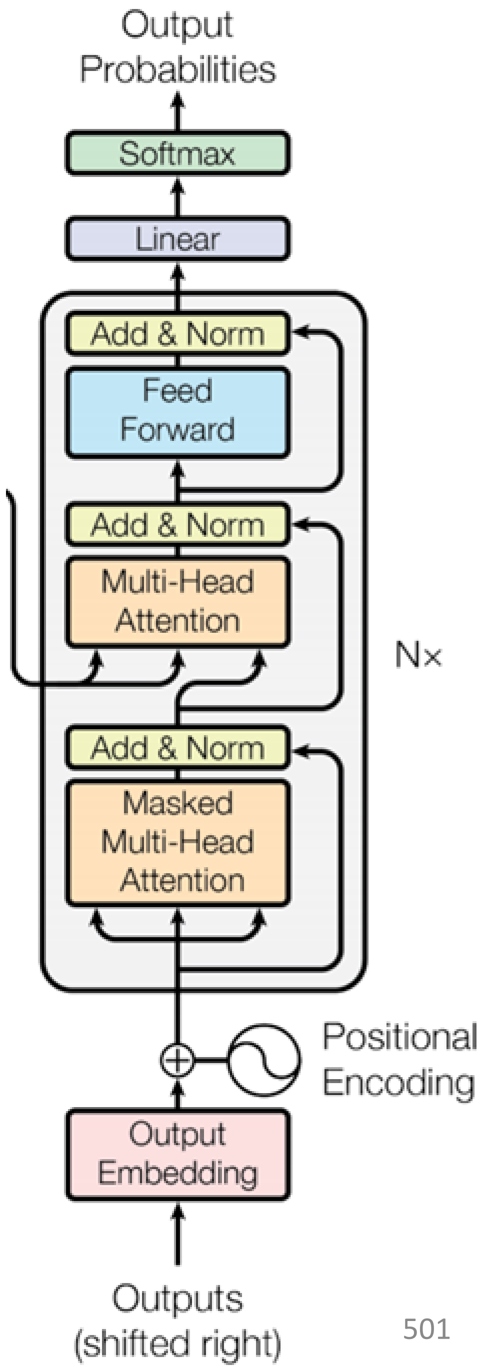


Decoder

- Masked self attention.

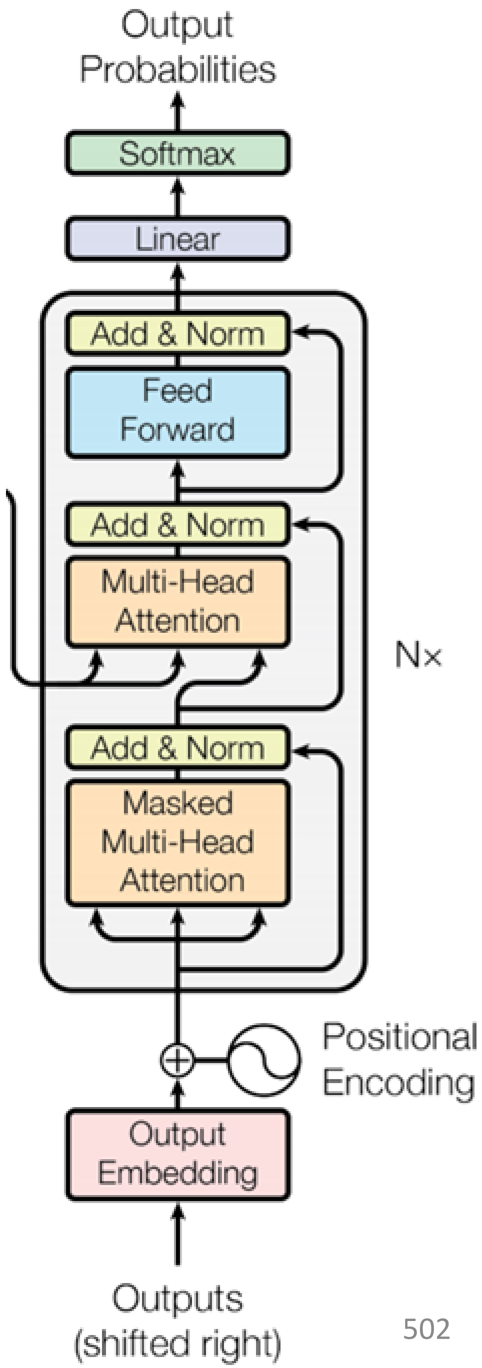
- Cross attention attention layer is like what attention does in sequence-to-sequence models.

- It helps the decoder emphasize on relevant parts of the input.



Decoder

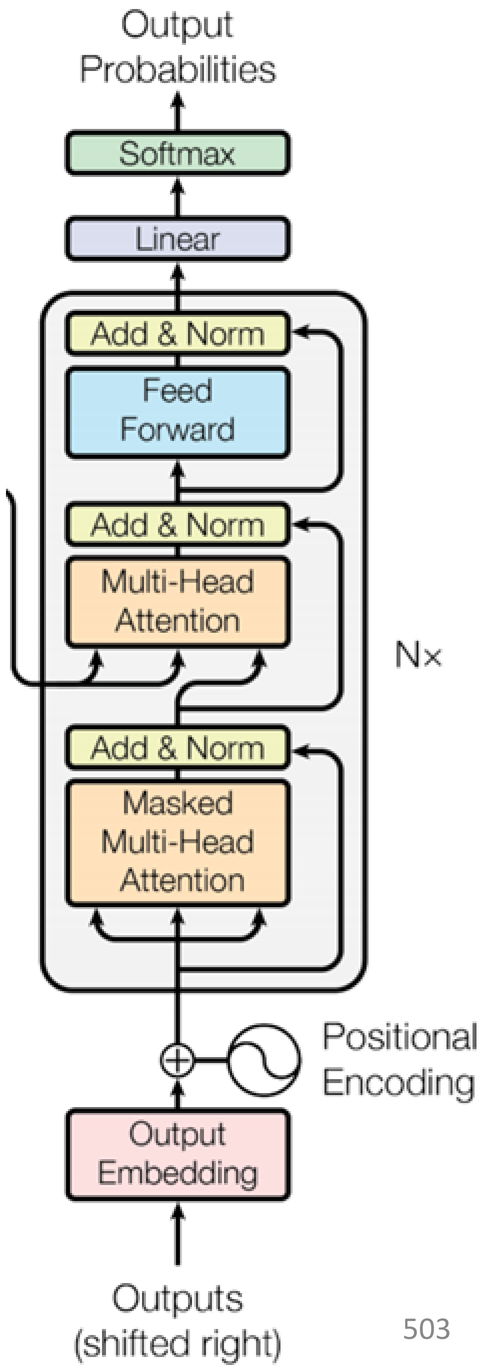
- Masked self attention.
- Cross attention attention layer is like what attention does in sequence-to-sequence models.
- It helps the decoder emphasize on relevant parts of the input.
- The same feed-forward network is applied to each position.



From Feedforward Network to Word Prediction

Linear Projection:

- Primary Role: Adjusting dimensionality.
- The linear layer serves to change the dimensionality of the feedforward network's output to match the size of the vocabulary.
- This ensures that the output has a dimension corresponding to every word in the dictionary.

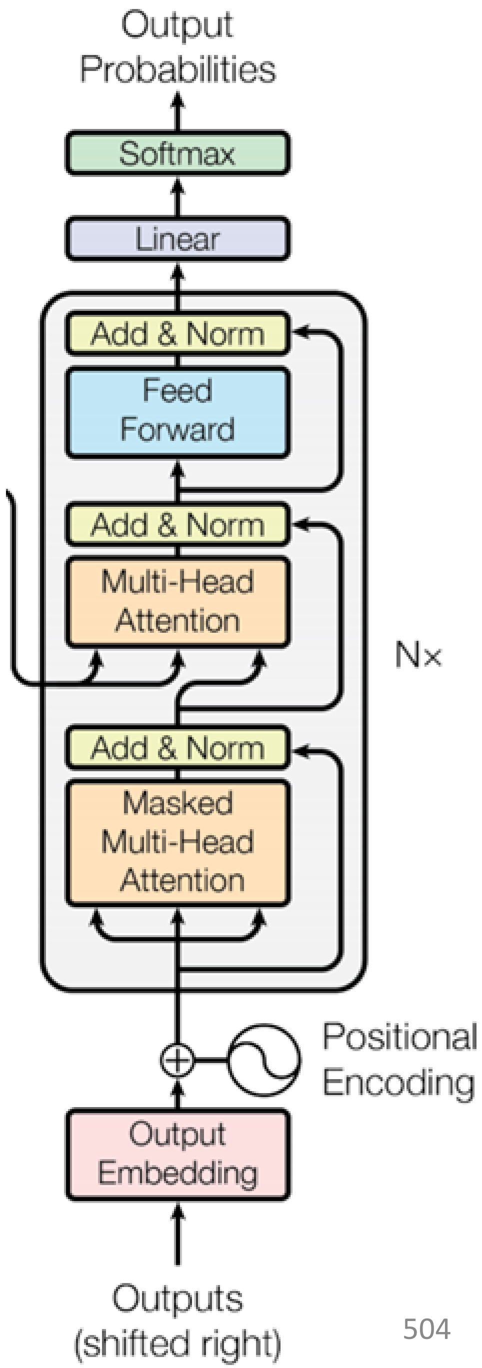


From Feedforward Network to Word Prediction

Softmax Activation:

•This function transforms the linear layer's output into probabilities.

•Representing the likelihood of a respective word being the next word in the sequence.



Positional Encoding

- Problem: no recurrence and no convolution, the model has no sense of the sequence.

Positional Encoding

- Problem: no recurrence and no convolution, the model has no sense of the sequence.

- We need a way to account for the order of the tokens in the sequence.

Positional Encoding

- Problem: no recurrence and no convolution, the model has no sense of the sequence.

- We need a way to account for the order of the tokens in the sequence.

- Solution: Adds a vector accounting for the position to each input embedding.

Positional Encoding

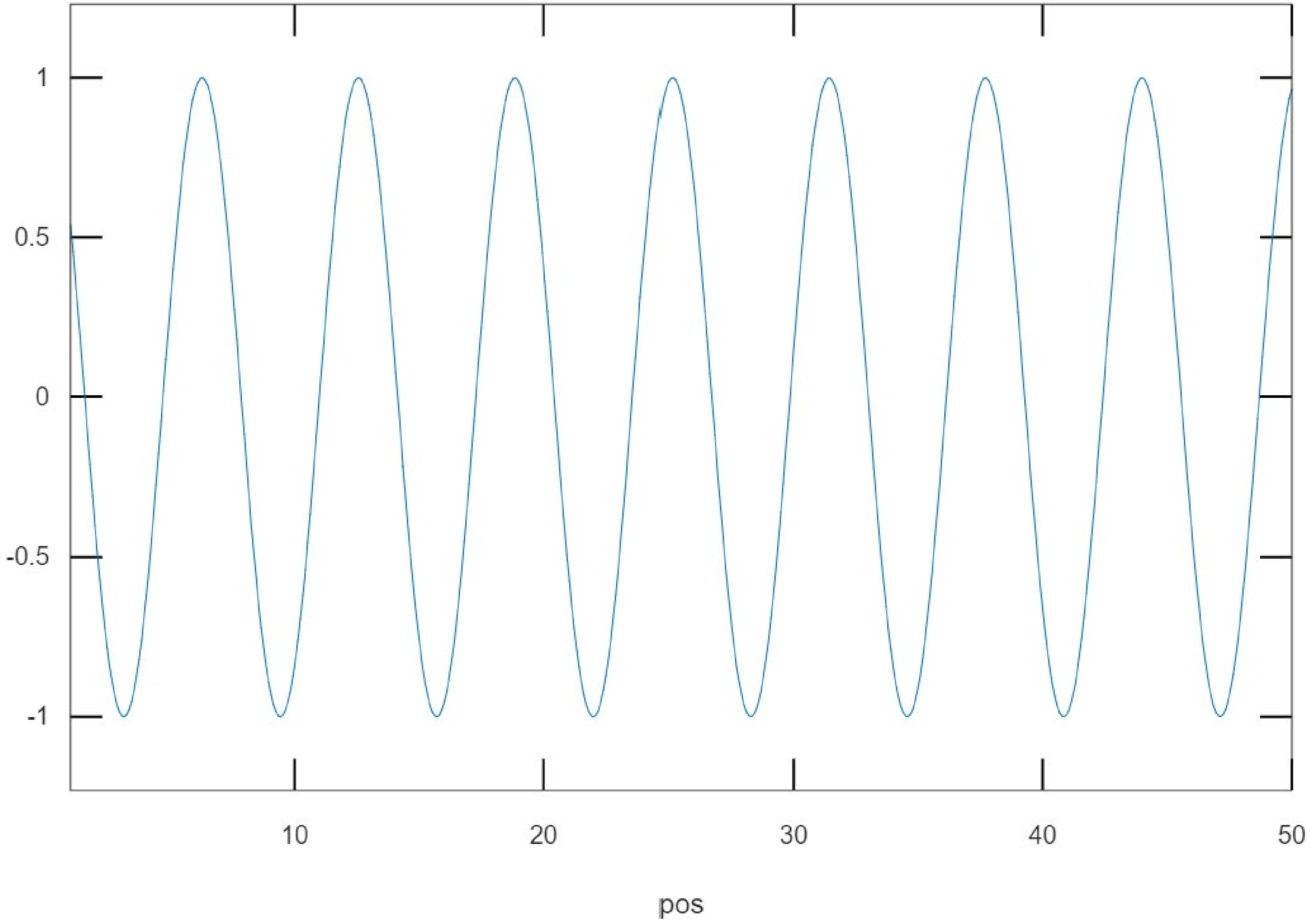
$$PE(pos, 2i + 1 \quad) = \cos\left(\frac{pos}{10000^{\frac{2i}{p}}}\right)$$

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

Positional Encoding

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

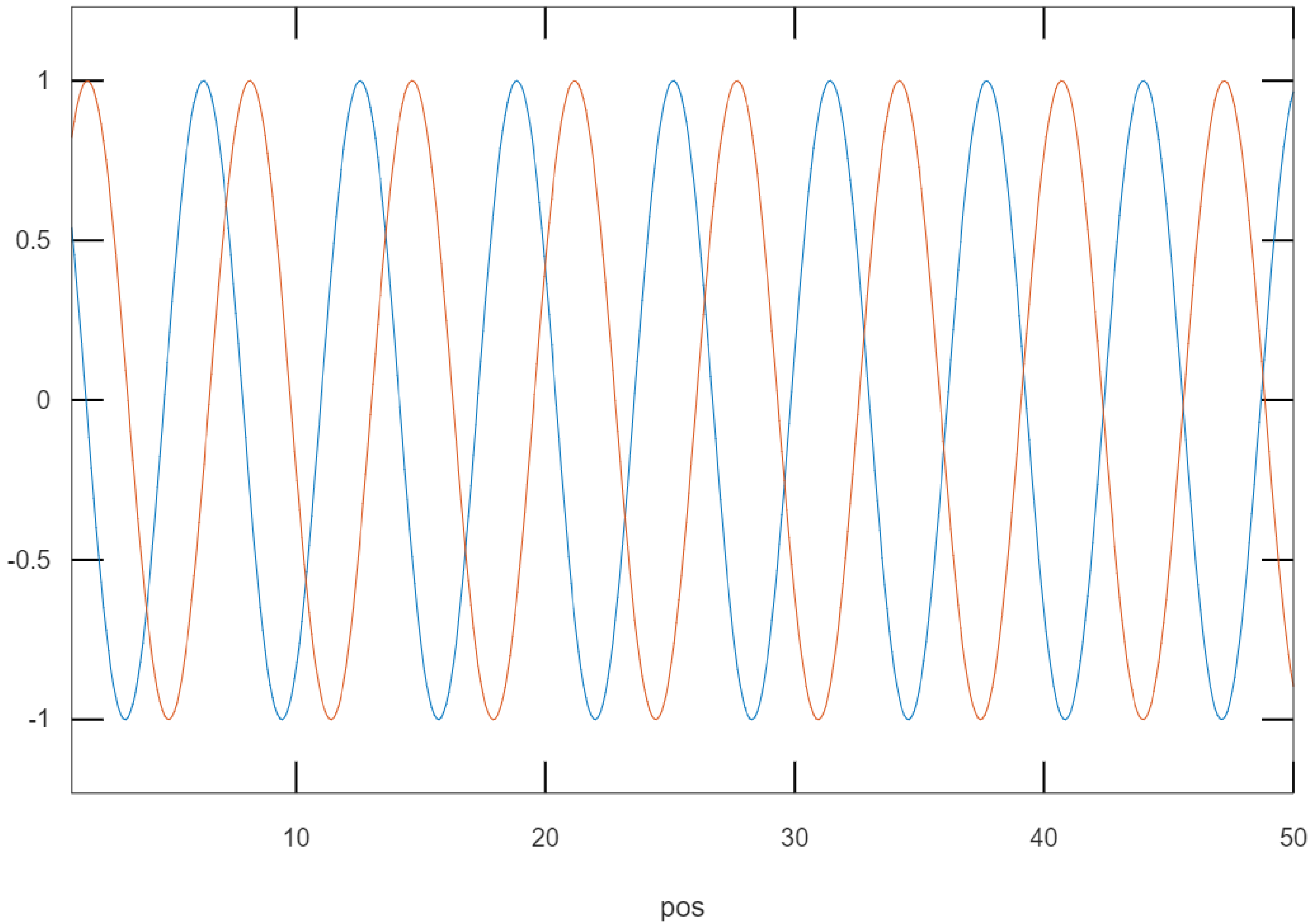
Positional Encoding



$i = 0$

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

Positional Encoding



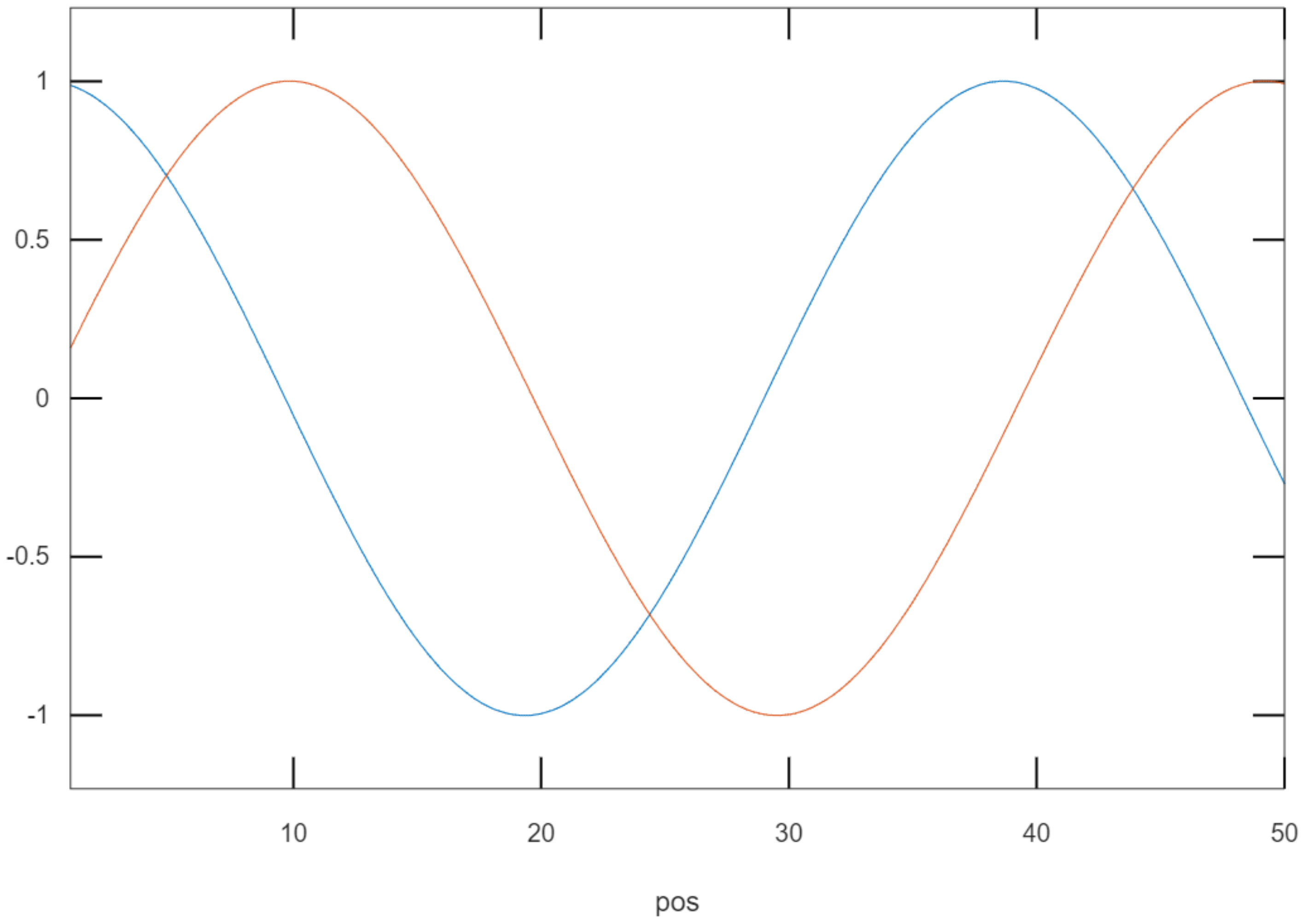
$i = 0$

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

$i = 1$

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

Positional Encoding

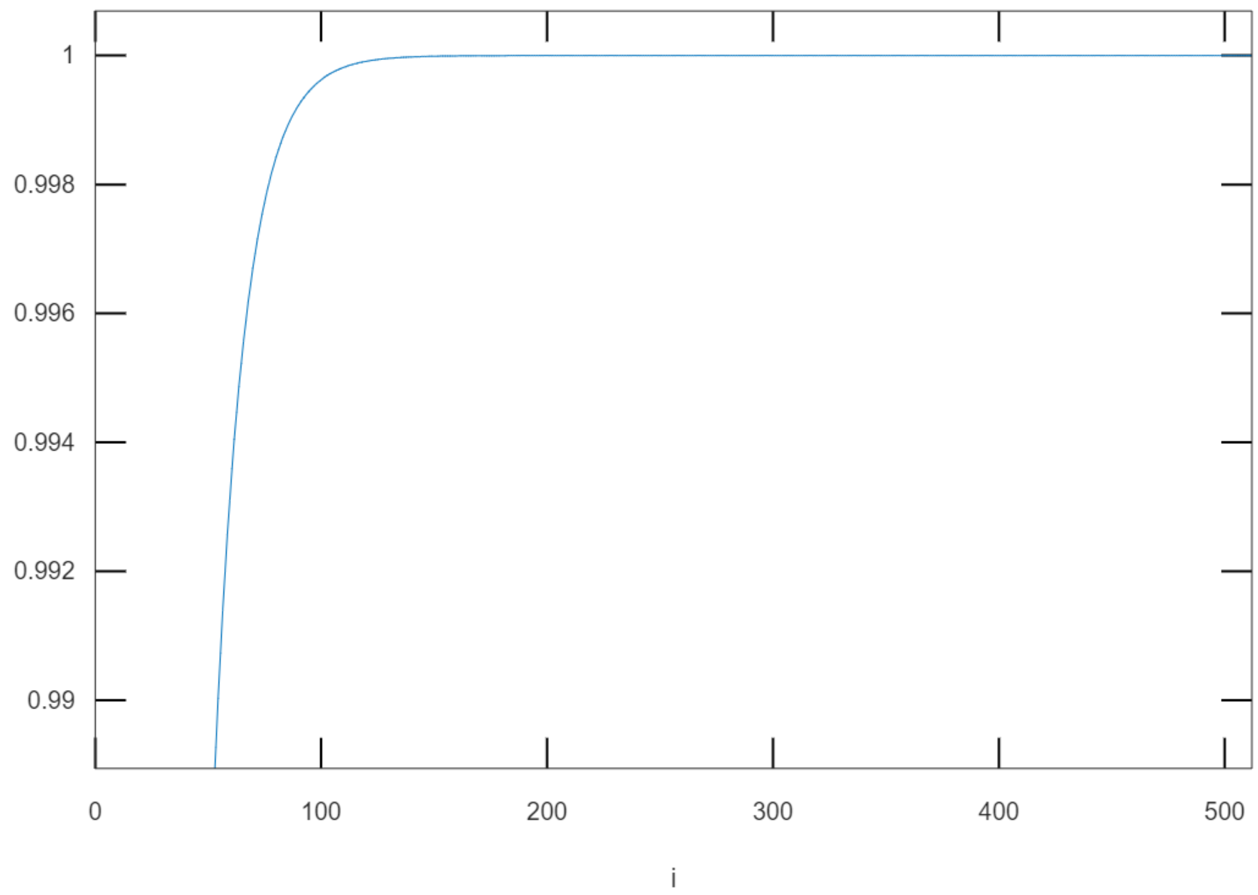


$i = 50$

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

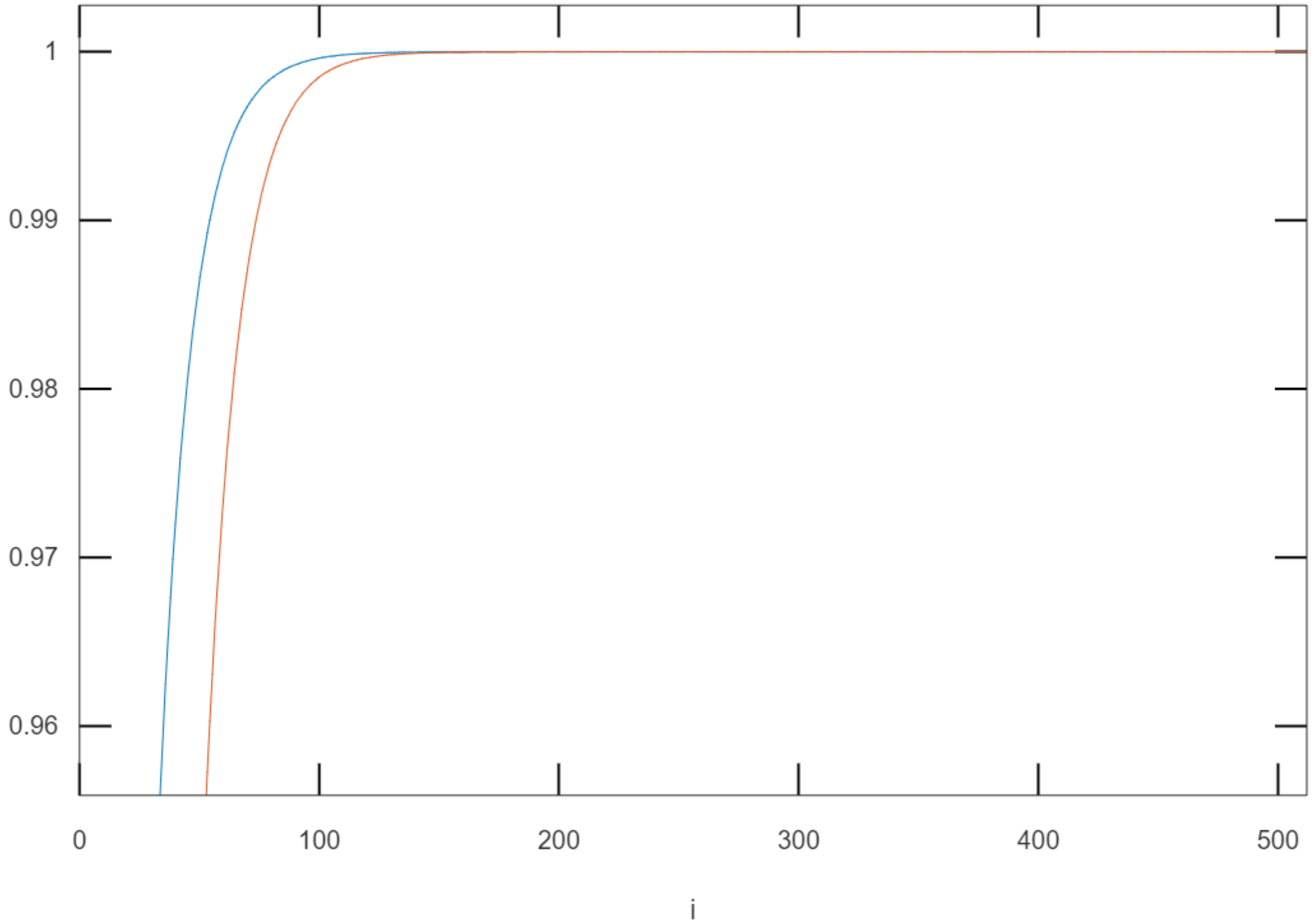
$$PE(pos, 2i\ \ ) = \sin\left(\frac{pos}{10000^{\frac{2i}{p}}}\right)$$

Positional Encoding



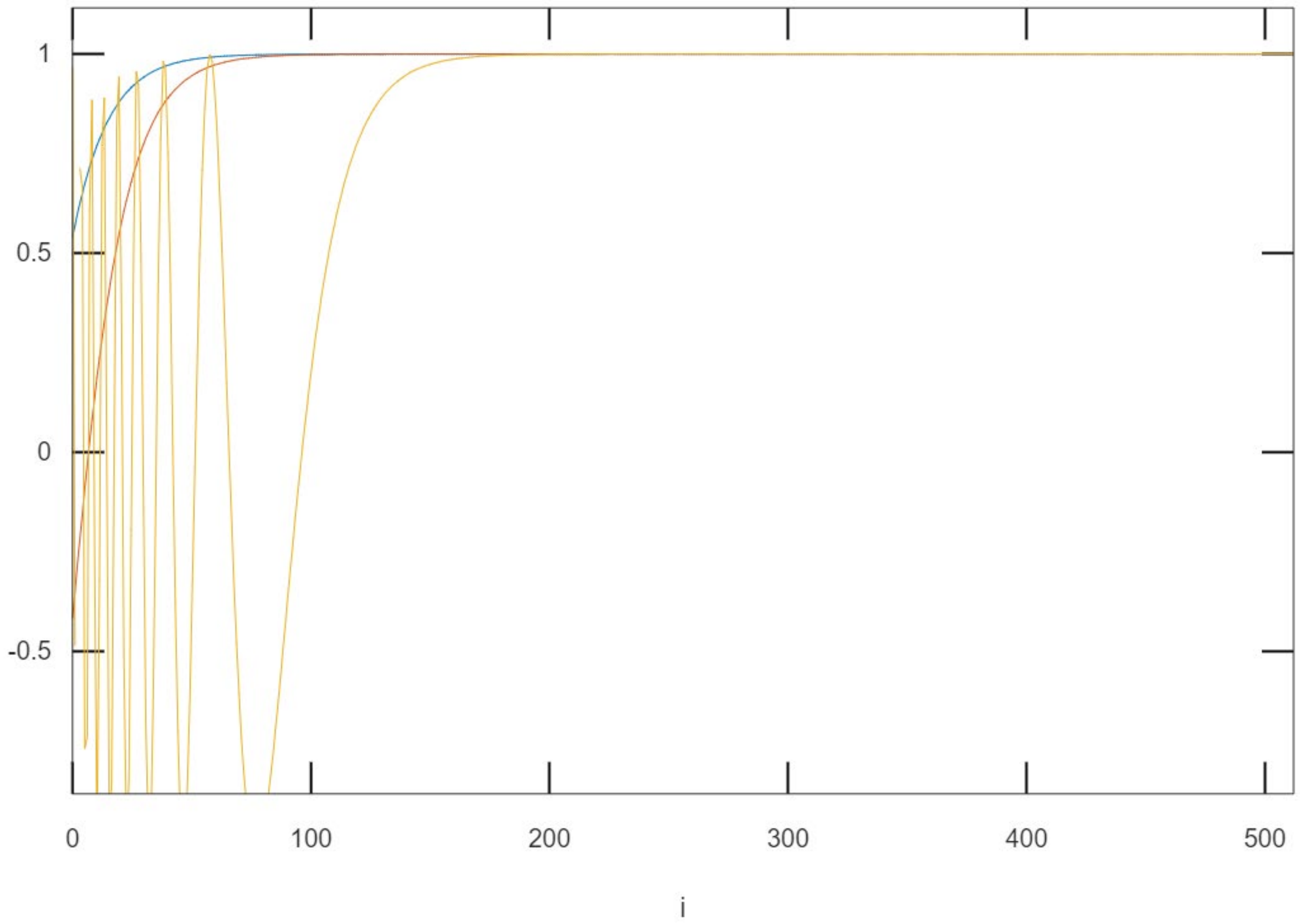
$$pos = 1$$

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



$pos = 2$

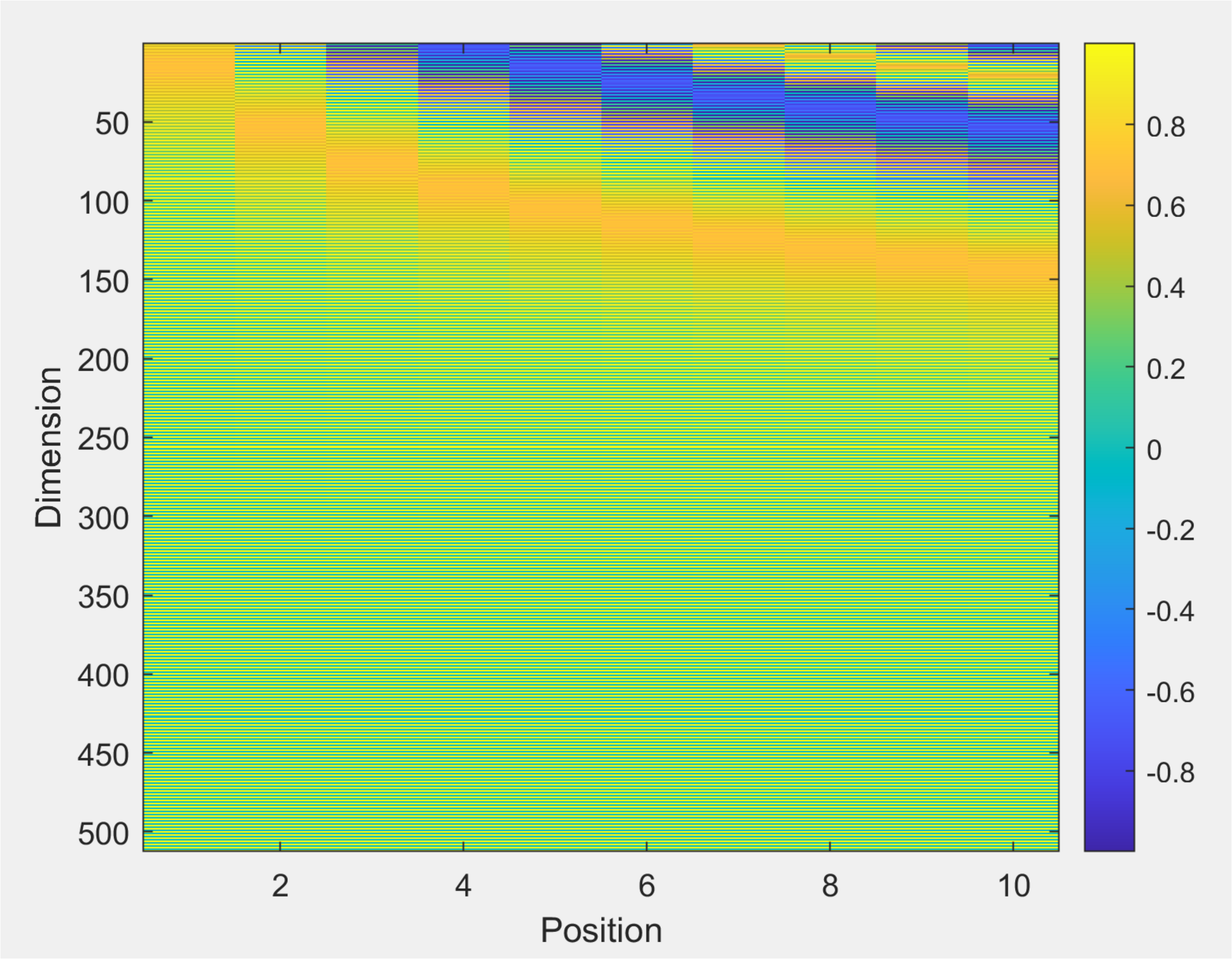
$$PE(pos, 2i+1\ \ ) = \cos\left(\frac{pos}{10000^{\frac{2i}{p}}}\right)$$



pos =50

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

Positional Encoding Visualization



Binary representation