

Transformer Model

Shusen Wang



Transformer Model

- **Original paper:** Vaswani et al. [Attention Is All You Need](#). In *NIPS*, 2017.

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

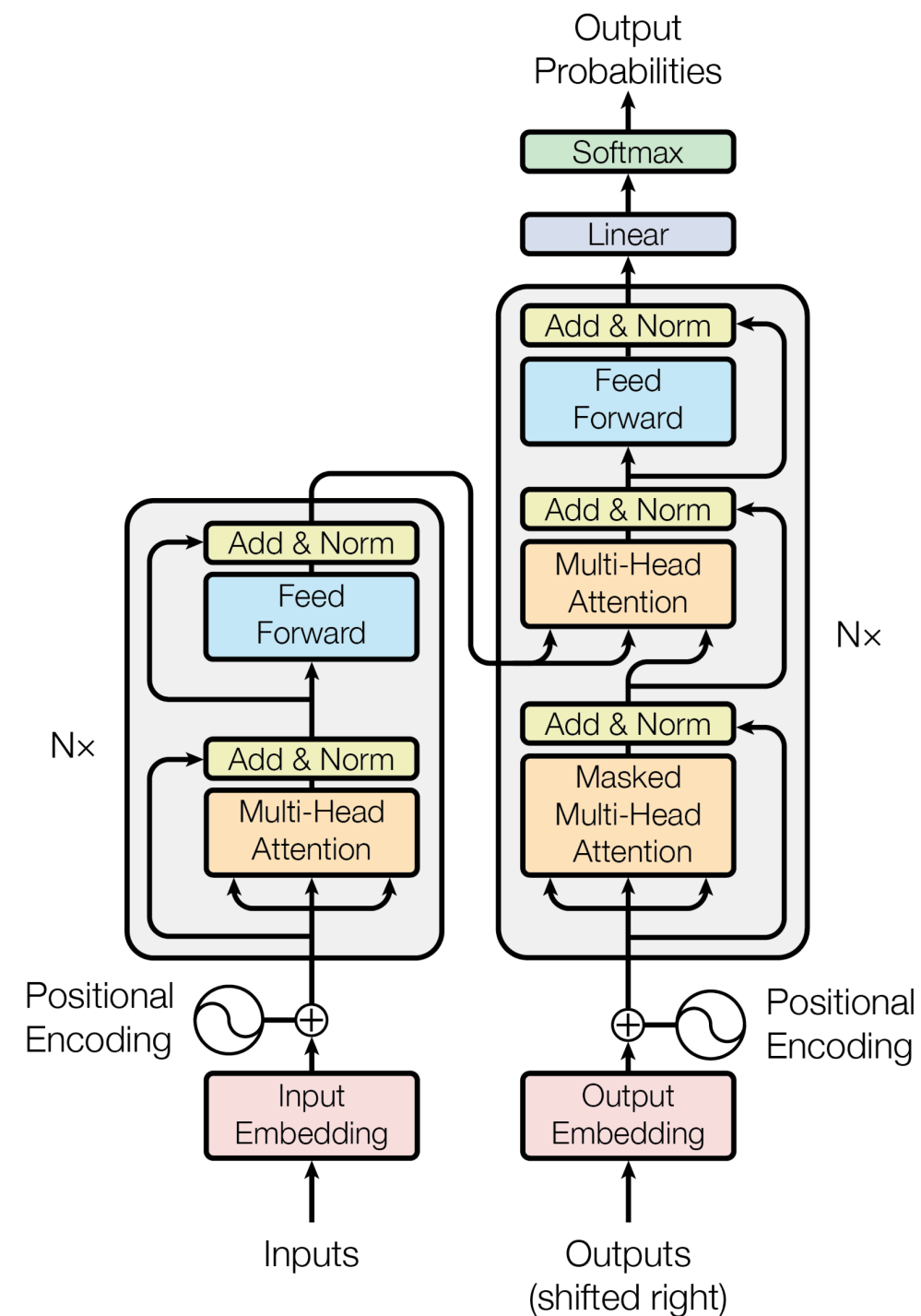
Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

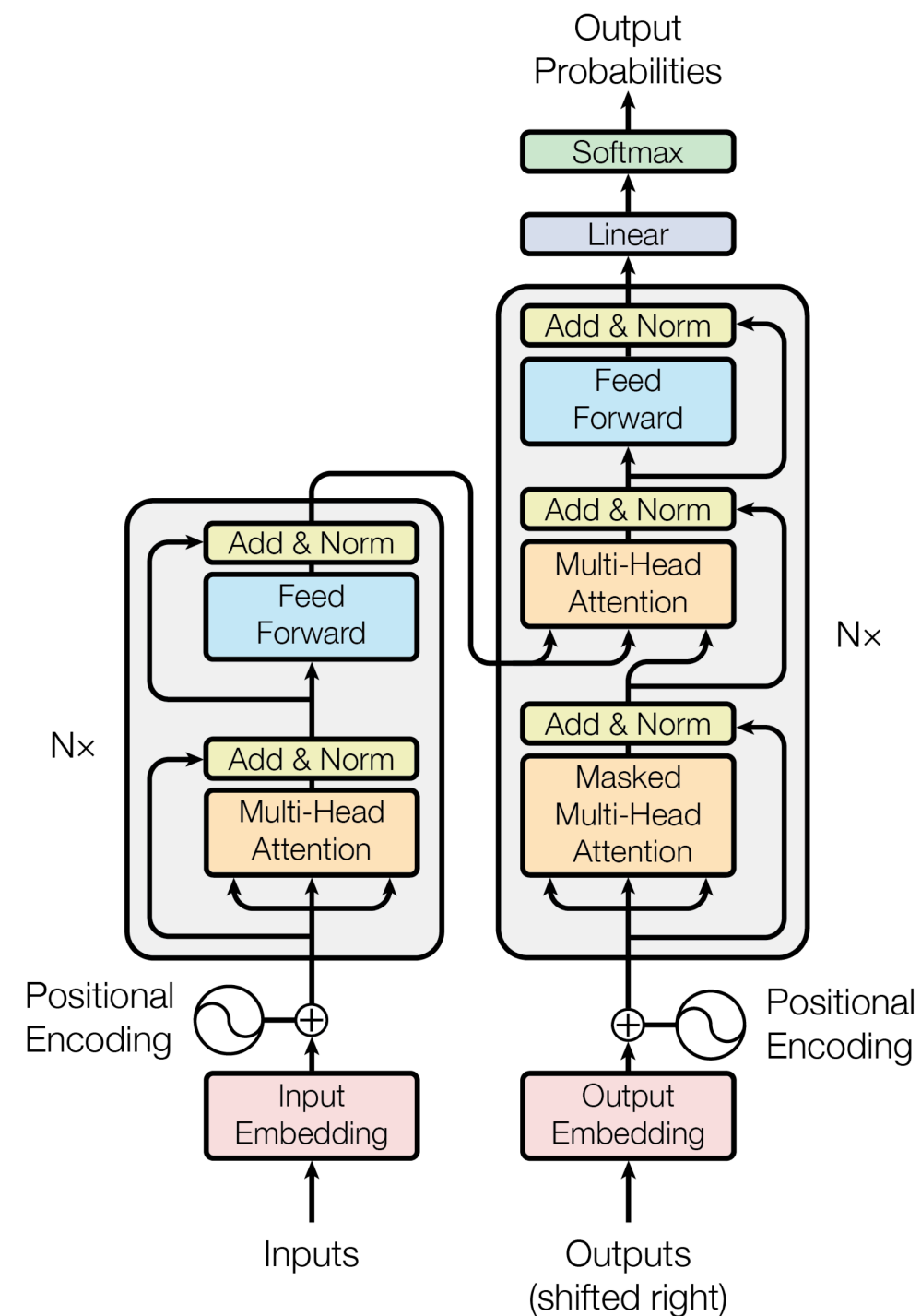
Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com



Transformer Model

- Transformer is a Seq2Seq model.
- Transformer is not RNN.
- Purely based attention and fully-connected layers.
- Much more computations than RNNs.
- Higher performance than RNNs on large datasets.

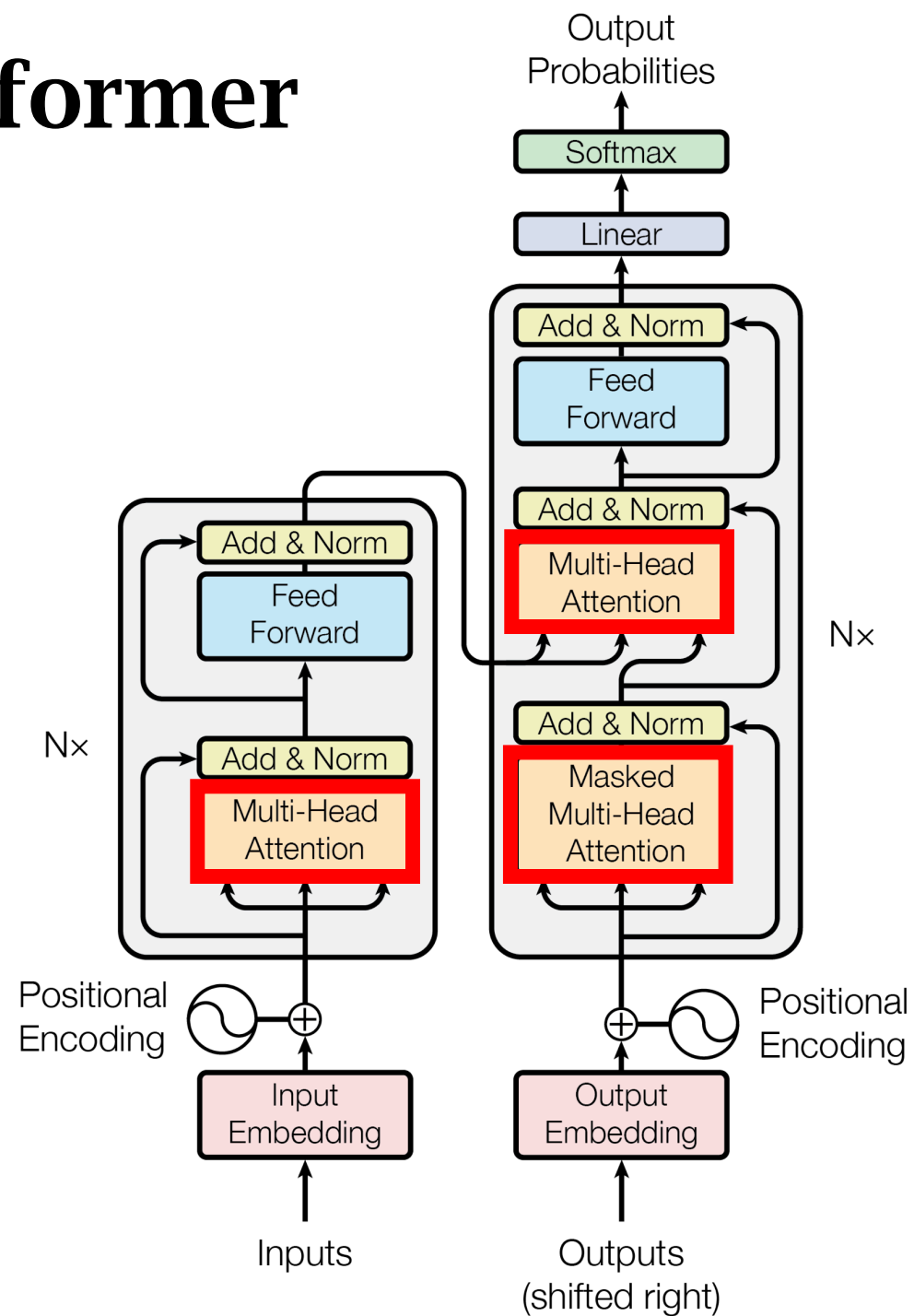


Attention beyond RNNs

Attention in Transformer

Multi-head attention:

- Multiple **single-head attentions**, each has its own parameter matrices.
- Concatenate the outputs.

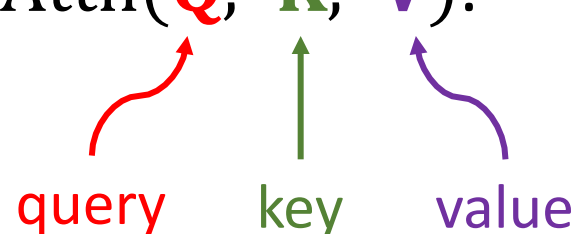


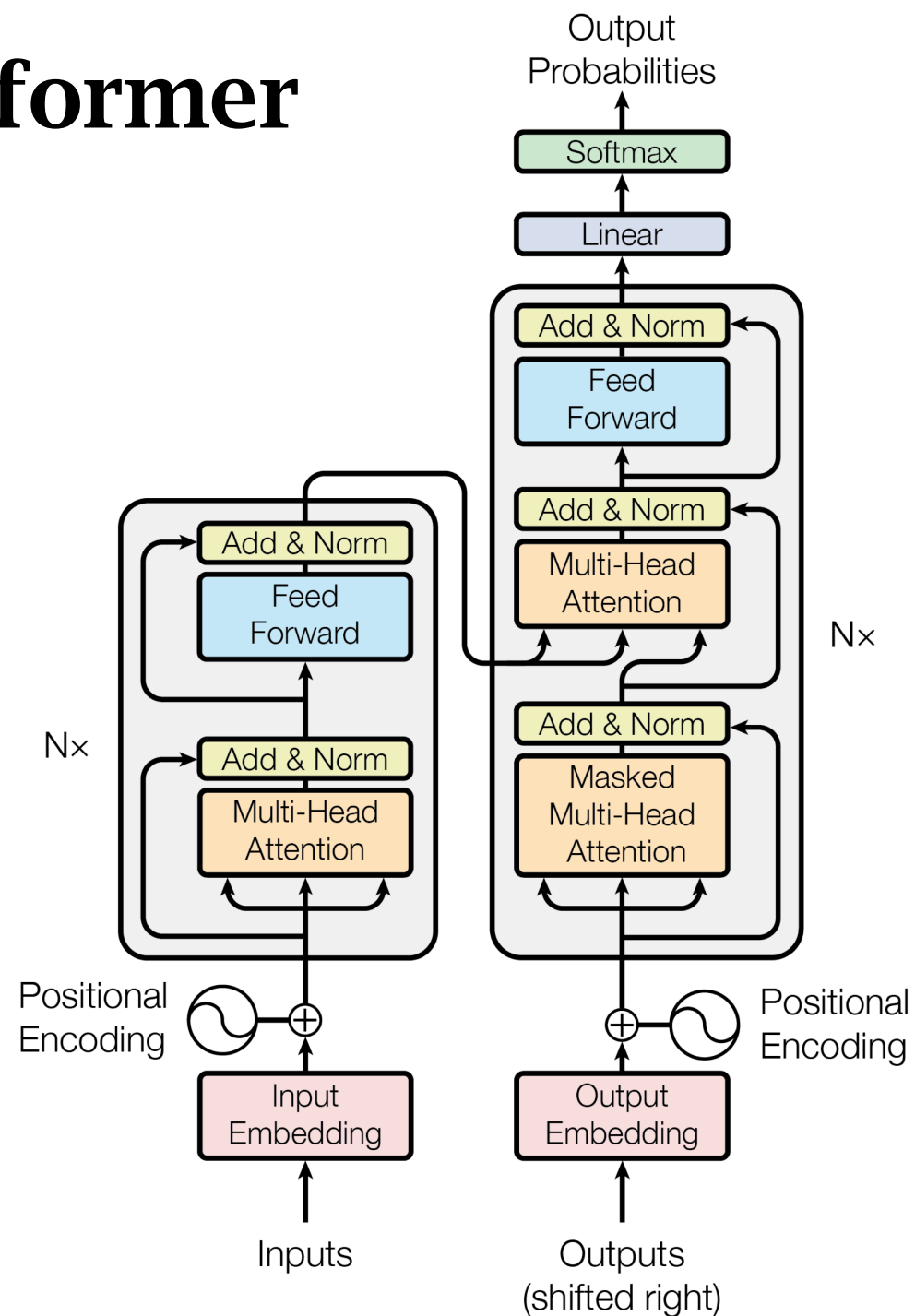
Attention in Transformer

Multi-head attention:

- Multiple single-head attentions, each has its own parameter matrices.
- Concatenate the outputs.

Single-head attention:

- $C = \text{Attn}(Q, K, V)$.




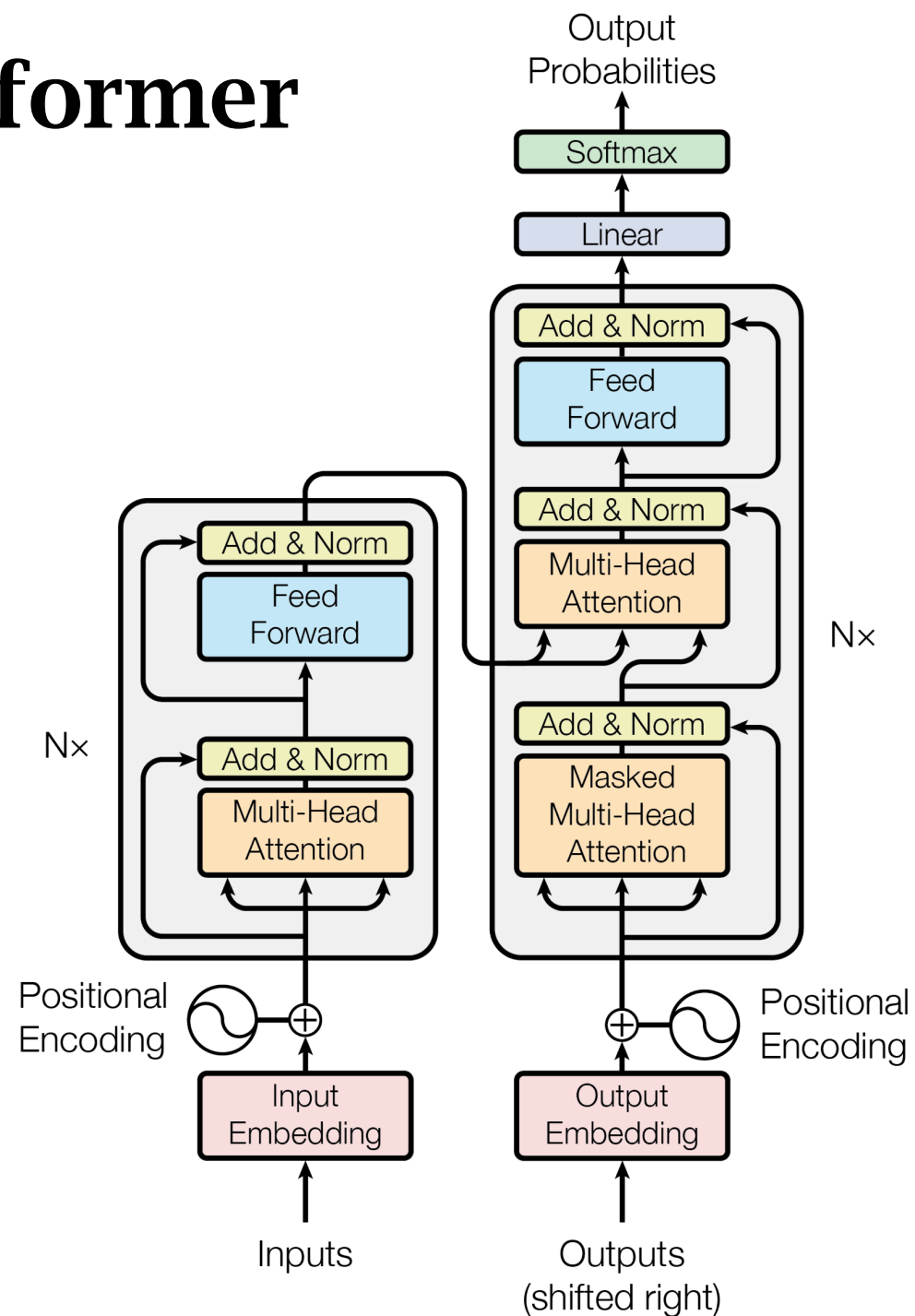
Attention in Transformer

Multi-head attention:

- Multiple single-head attentions, each has its own parameter matrices.
- Concatenate the outputs.

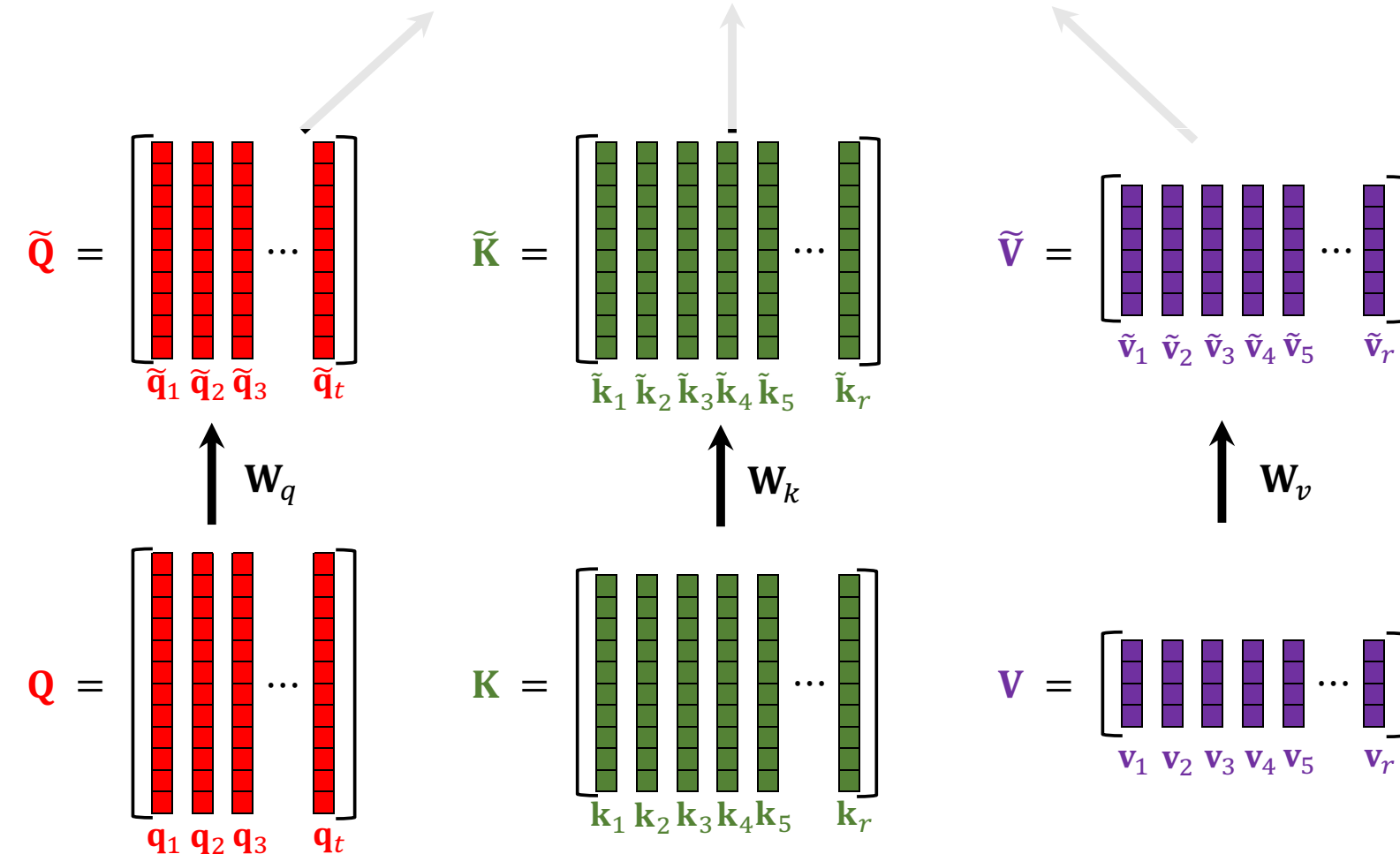
Single-head attention:

- $\mathbf{C} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.
- \mathbf{Q} and \mathbf{C} have t columns.
- t : sequence length.
- \mathbf{K} and \mathbf{V} have r columns (r is arbitrary).



Single-Head Attention: $\mathbf{C} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.

$$\mathbf{C} = \begin{bmatrix} \text{c}_1 & \text{c}_2 & \text{c}_3 & \dots & \text{c}_t \end{bmatrix}, \text{ where } \mathbf{c}_i = \tilde{\mathbf{V}} \cdot \text{softmax}(\tilde{\mathbf{K}}^T \tilde{\mathbf{q}}_i)$$



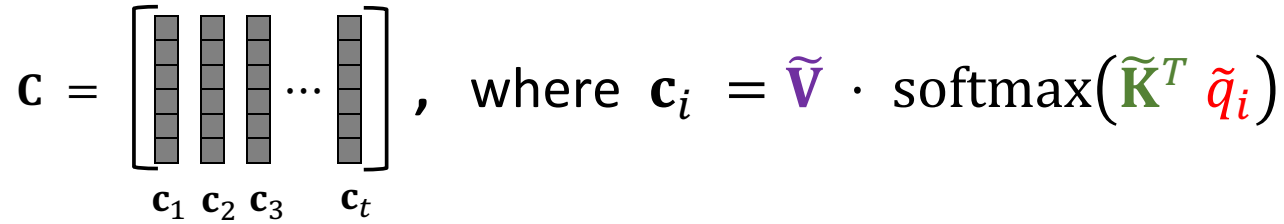
Linear maps:

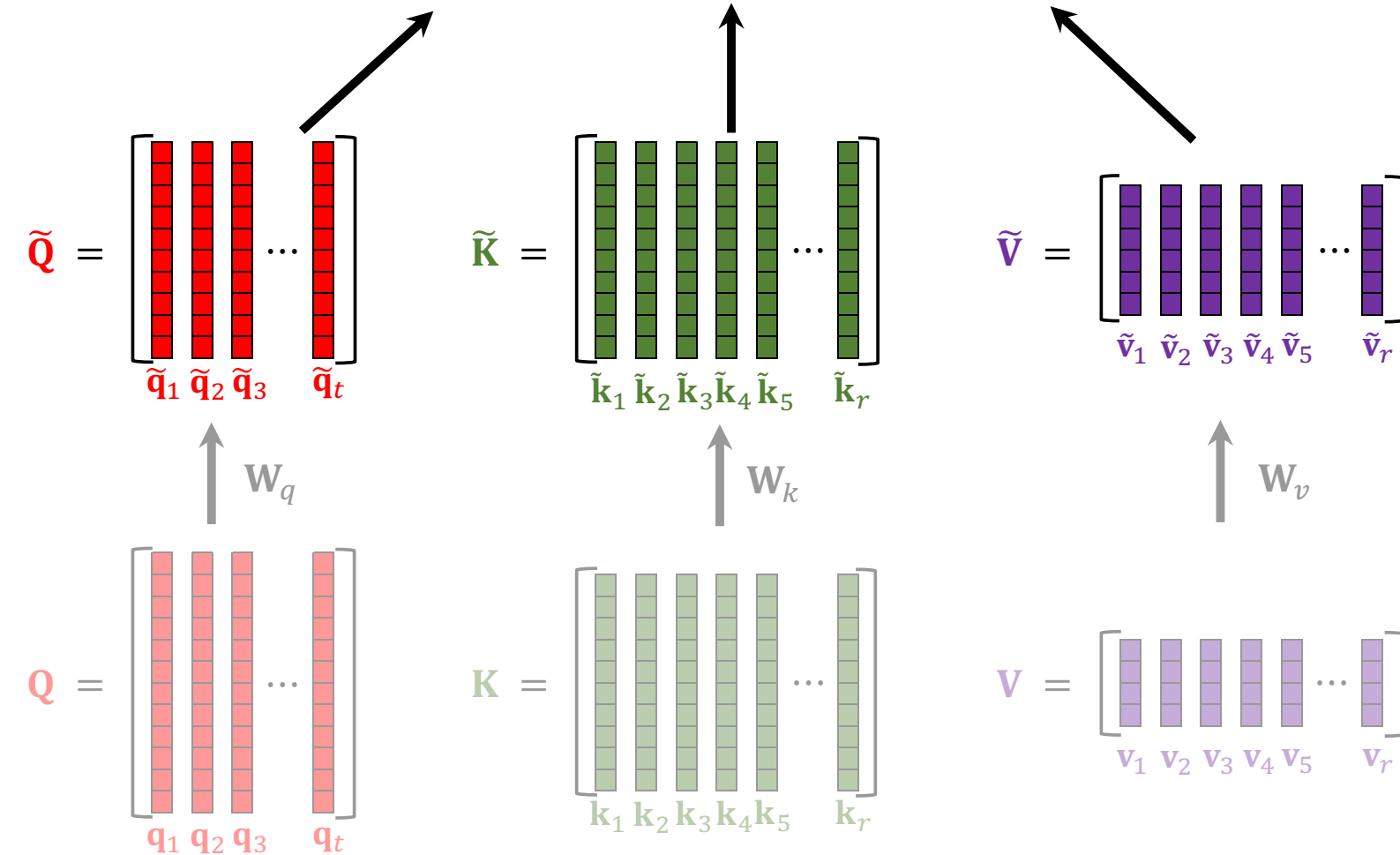
- $\tilde{\mathbf{Q}} = \mathbf{W}_q \mathbf{Q}$,
- $\tilde{\mathbf{K}} = \mathbf{W}_k \mathbf{K}$,
- $\tilde{\mathbf{V}} = \mathbf{W}_v \mathbf{V}$.

Trainable parameters:

- $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$.

Single-Head Attention: $\mathbf{C} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.

$$\mathbf{C} = \begin{bmatrix} \text{col}_1 & \text{col}_2 & \text{col}_3 & \dots & \text{col}_t \end{bmatrix}, \text{ where } \mathbf{c}_i = \tilde{\mathbf{V}} \cdot \text{softmax}(\tilde{\mathbf{K}}^T \tilde{\mathbf{q}}_i)$$




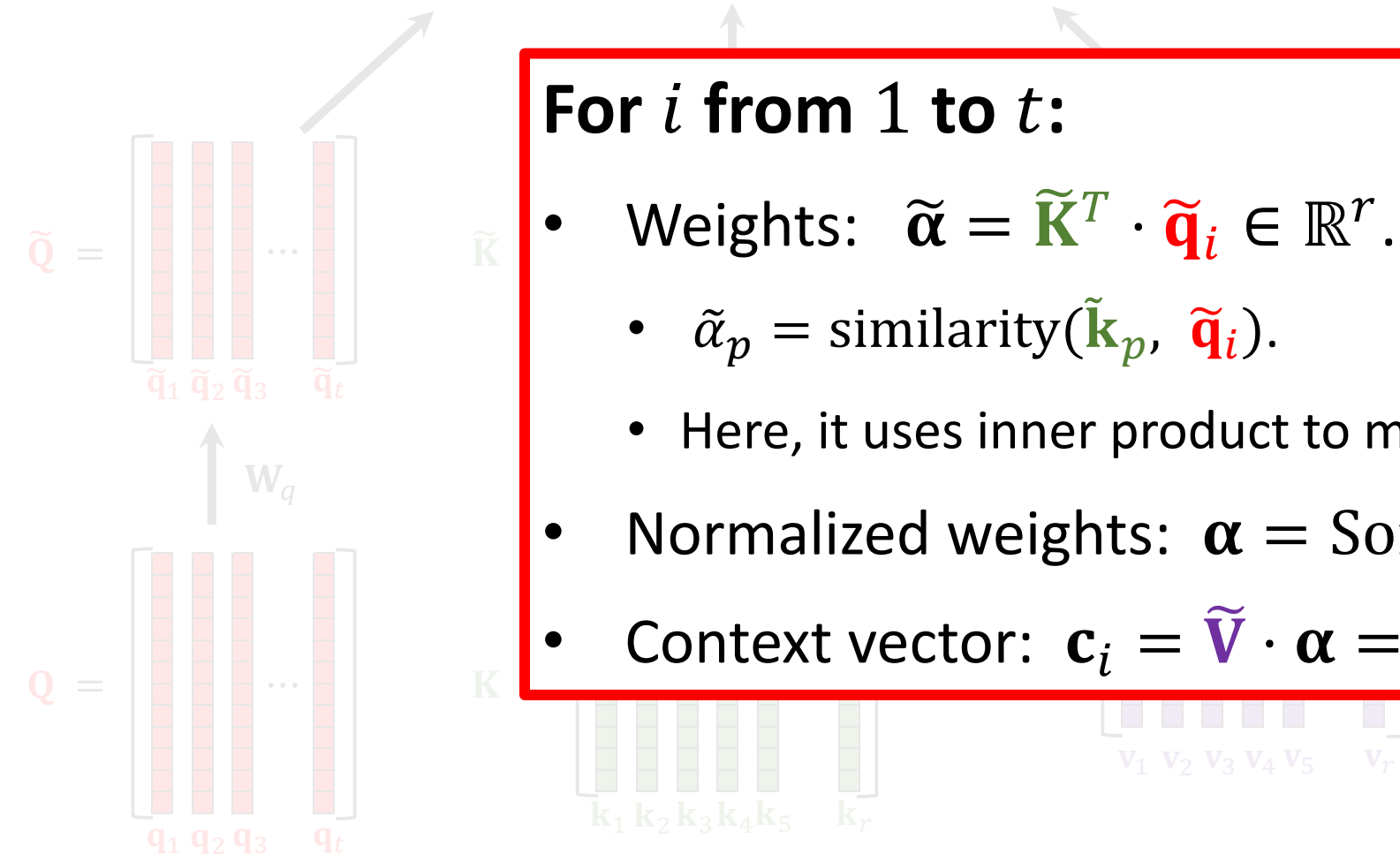
Single-Head Attention: $\mathbf{C} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.

$$\mathbf{C} = \begin{bmatrix} \text{column 1} & \text{column 2} & \text{column 3} & \dots & \text{column } t \end{bmatrix}, \text{ where } \mathbf{c}_i = \tilde{\mathbf{V}} \cdot \text{softmax}(\tilde{\mathbf{K}}^T \tilde{\mathbf{q}}_i)$$

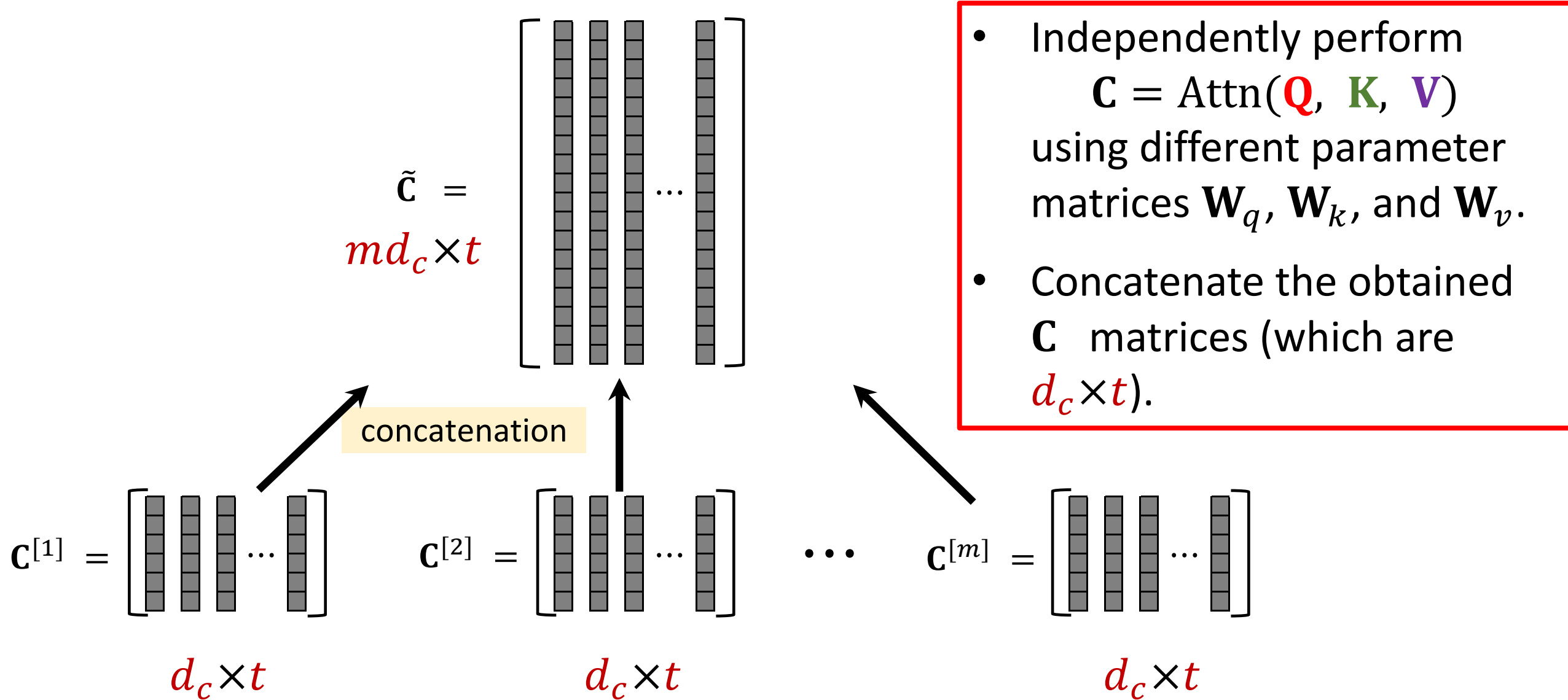
$\mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{c}_3 \quad \dots \quad \mathbf{c}_t$

For i from 1 to t :

- Weights: $\tilde{\boldsymbol{\alpha}} = \tilde{\mathbf{K}}^T \cdot \tilde{\mathbf{q}}_i \in \mathbb{R}^r$.
 - $\tilde{\alpha}_p = \text{similarity}(\tilde{\mathbf{k}}_p, \tilde{\mathbf{q}}_i)$.
 - Here, it uses inner product to measure similarity.
- Normalized weights: $\boldsymbol{\alpha} = \text{Softmax}(\tilde{\boldsymbol{\alpha}})$.
- Context vector: $\mathbf{c}_i = \tilde{\mathbf{V}} \cdot \boldsymbol{\alpha} = \alpha_1 \tilde{\mathbf{v}}_1 + \dots + \alpha_r \tilde{\mathbf{v}}_r$.



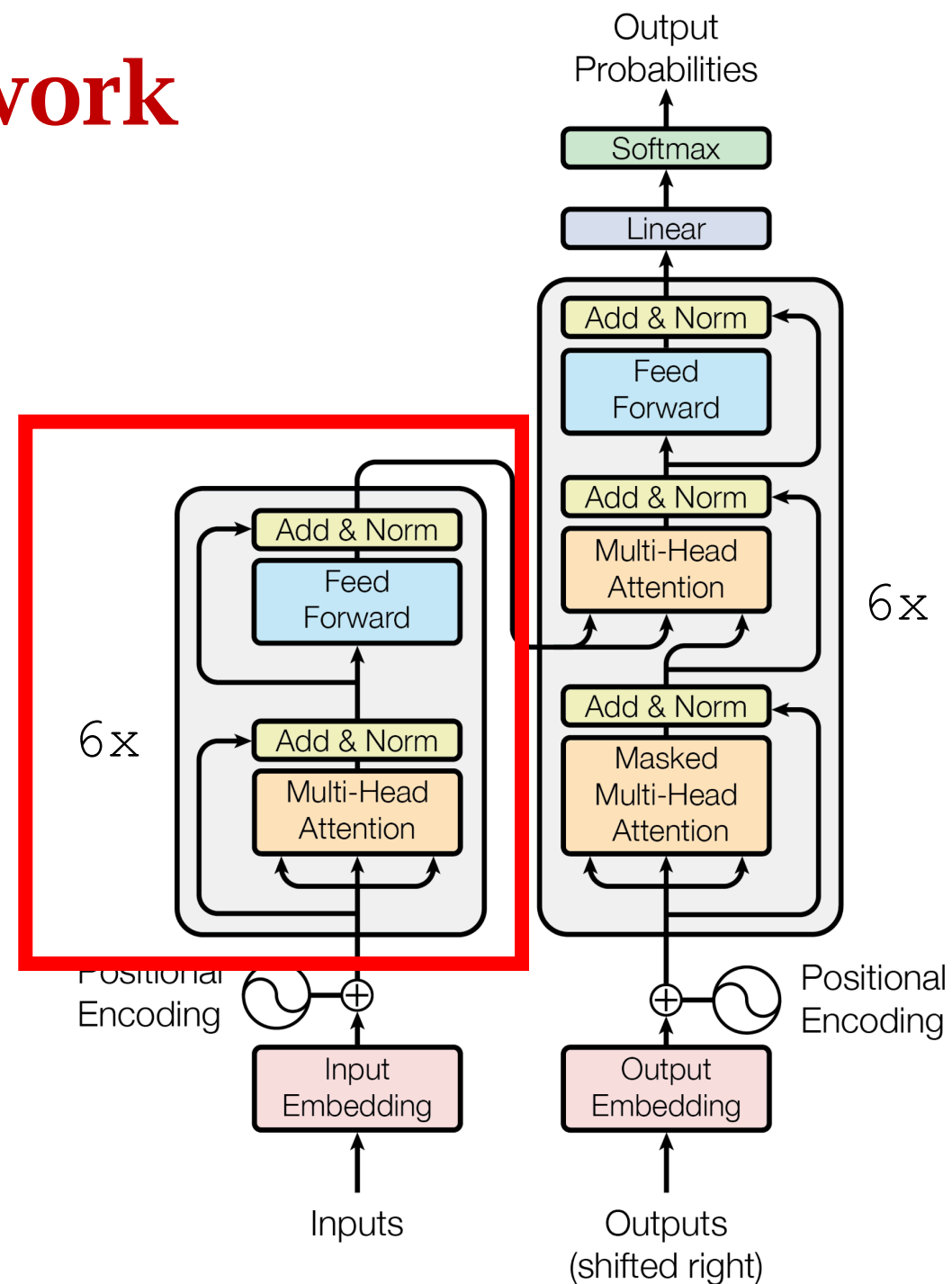
Multi-Head Attention



Encoder of Transformer

Encoder Network

- Encoder has 6 **blocks**.
- **1 block** = **Multi-head attention** + **Dense**.
- 6 is the result of hyper-parameter tuning; nothing magical about 6.
- Other tricks:
 - Skip connection.
 - Normalization.



Multi-Head Attention + Dense Layer

Multi-Head Attention

$$\tilde{\mathbf{C}} = [\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \tilde{\mathbf{c}}_3, \dots, \tilde{\mathbf{c}}_t] \in \mathbb{R}^{md_c \times t}$$

Concatenation

$$\mathbf{C}^{[1]} \in \mathbb{R}^{d_c \times t}$$

$$\mathbf{C}^{[2]} \in \mathbb{R}^{d_c \times t}$$

...

$$\mathbf{C}^{[m]} \in \mathbb{R}^{d_c \times t}$$

Attention w/ different
parameter matrices.

(**Q**, **K**, **V**)

Multi-Head Attention + Dense Layer

$\tilde{\mathbf{C}}$'s number of columns, t , is determined by \mathbf{Q} .

Multi-Head Attention

$$\tilde{\mathbf{C}} = [\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \tilde{\mathbf{c}}_3, \dots, \tilde{\mathbf{c}}_t] \in \mathbb{R}^{md_c \times t}$$

Concatenation

$$\mathbf{C}^{[1]} \in \mathbb{R}^{d_c \times t}$$

$$\mathbf{C}^{[2]} \in \mathbb{R}^{d_c \times t}$$

...

$$\mathbf{C}^{[m]} \in \mathbb{R}^{d_c \times t}$$

Attention w/ different
parameter matrices.

(\mathbf{Q} , \mathbf{K} , \mathbf{V})

Multi-Head Attention + Dense Layer

$$\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \tilde{\mathbf{u}}_3, \dots, \tilde{\mathbf{u}}_t] \in \mathbb{R}^{d_u \times t}$$

Dense layer is applied to every column independently.

$$\tilde{\mathbf{C}} = [\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \tilde{\mathbf{c}}_3, \dots, \tilde{\mathbf{c}}_t] \in \mathbb{R}^{md_c \times t}$$

Concatenation

$$\mathbf{C}^{[1]} \in \mathbb{R}^{d_c \times t}$$

$$\mathbf{C}^{[2]} \in \mathbb{R}^{d_c \times t}$$

...

$$\mathbf{C}^{[m]} \in \mathbb{R}^{d_c \times t}$$

Attention w/ different parameter matrices.

(**Q**, **K**, **V**)

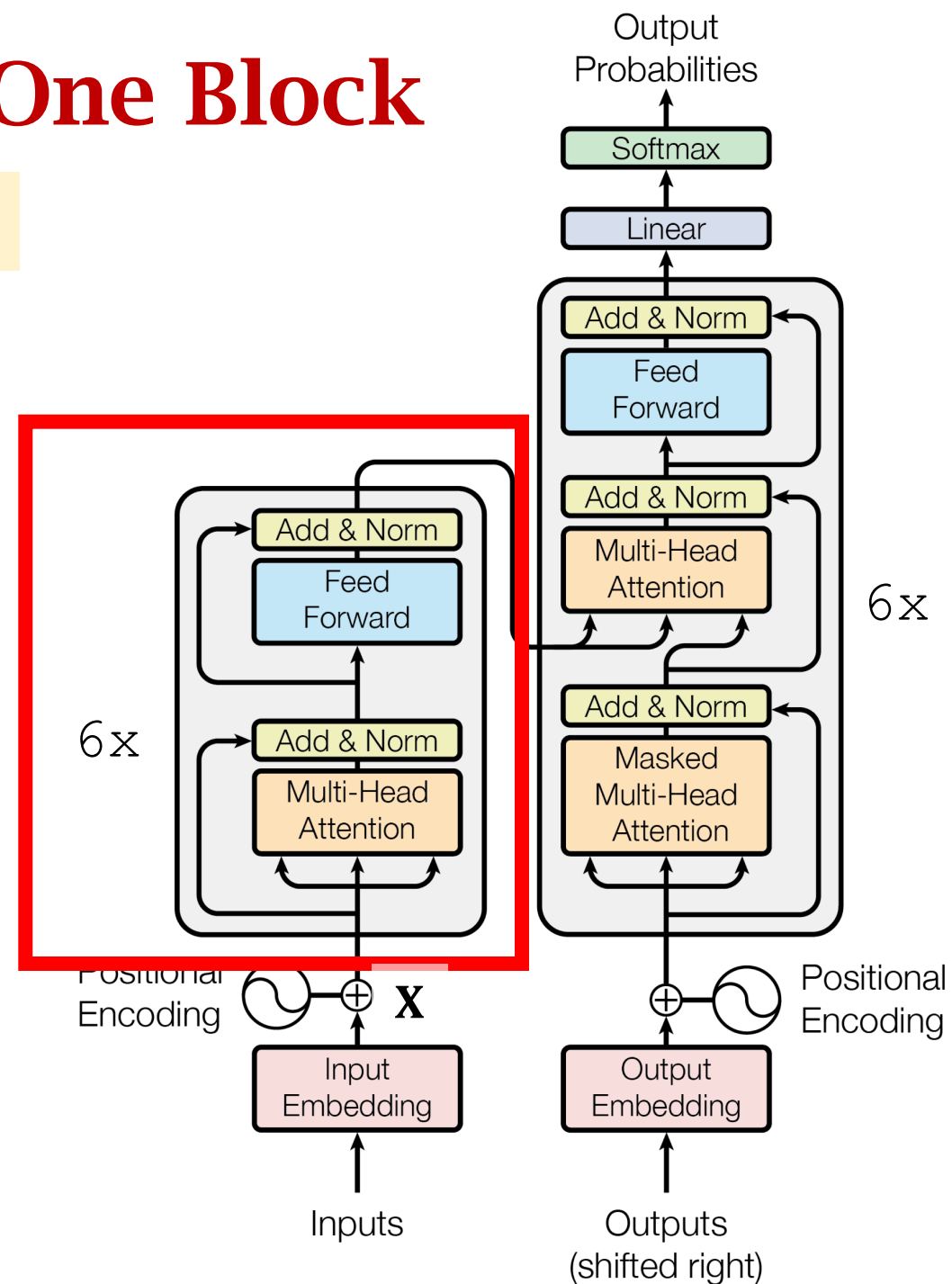
Encoder Network: One Block

Ignore skip connection and normalization.

- Input: $\mathbf{X} \in \mathbb{R}^{512 \times t}$; (t is the seq length.)
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$.

query key value

Similar to self-attention.

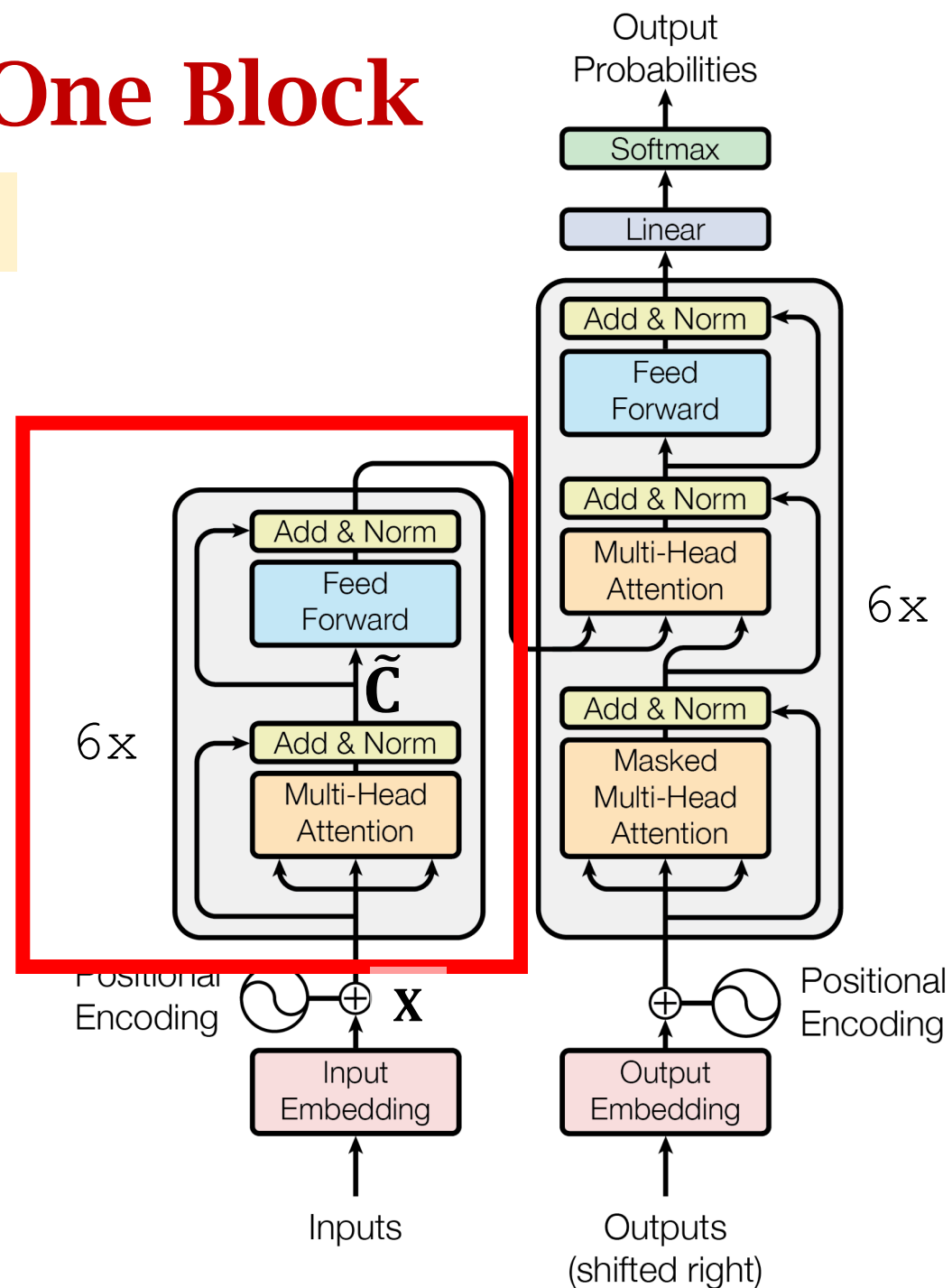


Encoder Network: One Block

Ignore skip connection and normalization.

- Input: $\mathbf{X} \in \mathbb{R}^{512 \times t}$; (t is the seq length.)
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$.
- Repeat single-head attention $m = 8$ times:
 $\mathbf{C}^{[i]} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{64 \times t}$.
- $\tilde{\mathbf{C}} = \text{Concatenate}(\mathbf{C}^{[1]}, \dots, \mathbf{C}^{[m]}) \in \mathbb{R}^{512 \times t}$.

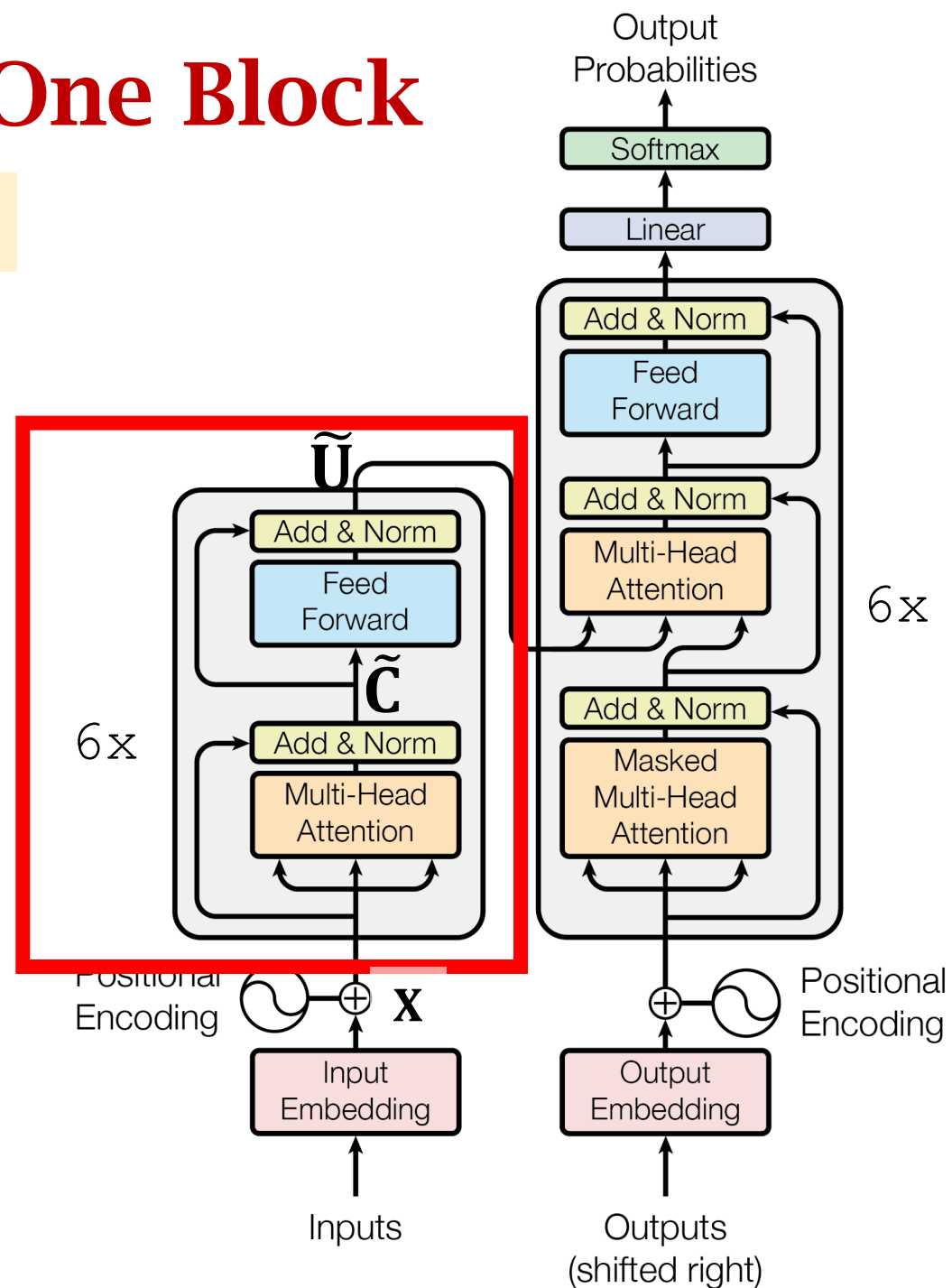
- Make sure the **input shape** and **output shape** are the same.
- Otherwise, skip connection cannot be applied.



Encoder Network: One Block

Ignore skip connection and normalization.

- Input: $\mathbf{X} \in \mathbb{R}^{512 \times t}$; (t is the seq length.)
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$.
- Repeat single-head attention $m = 8$ times:
 $\mathbf{C}^{[i]} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{64 \times t}$.
- $\tilde{\mathbf{C}} = \text{Concatenate}(\mathbf{C}^{[1]}, \dots, \mathbf{C}^{[m]}) \in \mathbb{R}^{512 \times t}$.
- $\tilde{\mathbf{U}} = \text{DenseLayer}(\tilde{\mathbf{C}}) \in \mathbb{R}^{512 \times t}$.
- Output: $\tilde{\mathbf{U}} \in \mathbb{R}^{512 \times t}$. (The same shape as \mathbf{X} .)

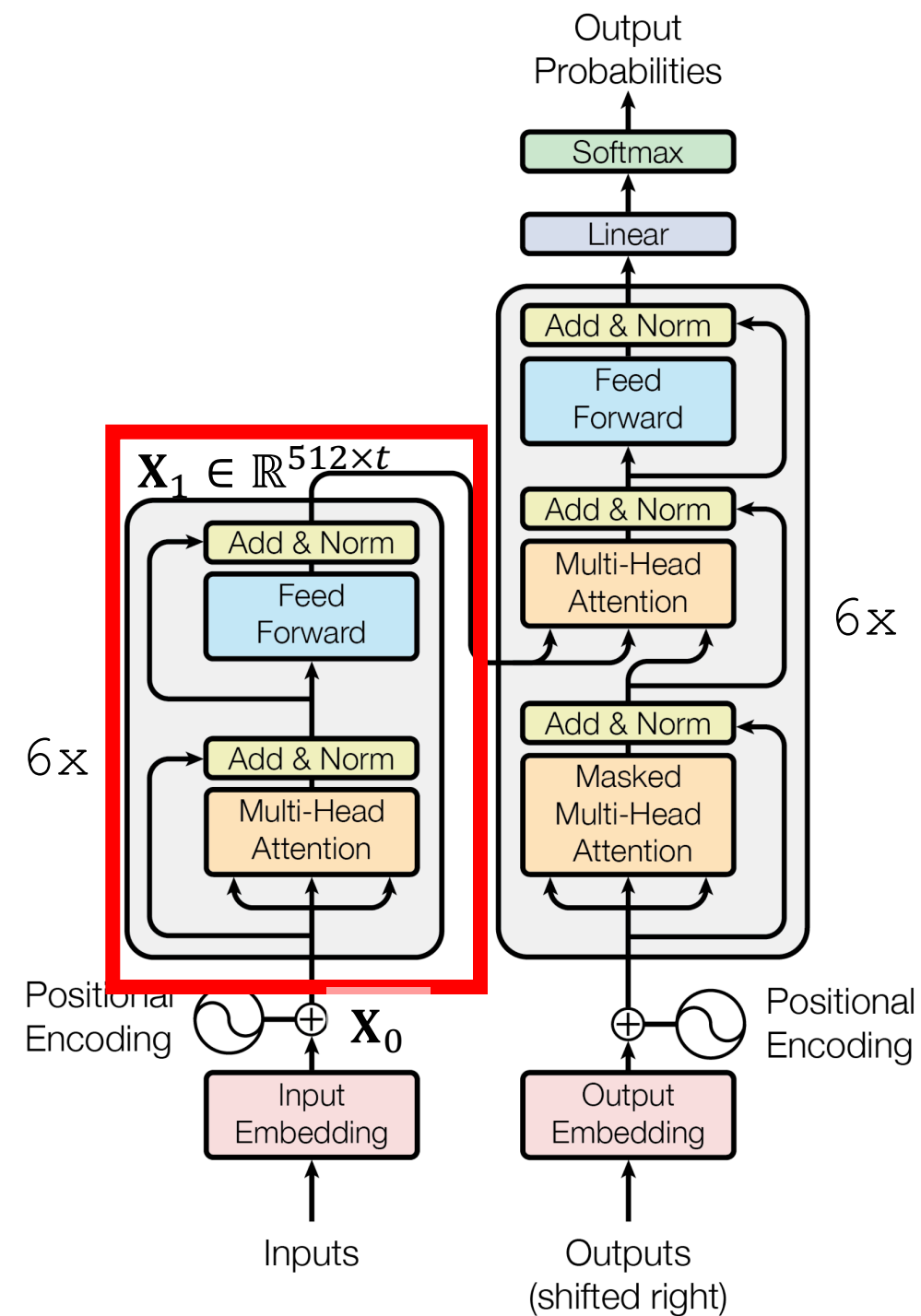


Encoder Network

$$\mathbf{X}_{(1)} \in \mathbb{R}^{512 \times t}$$

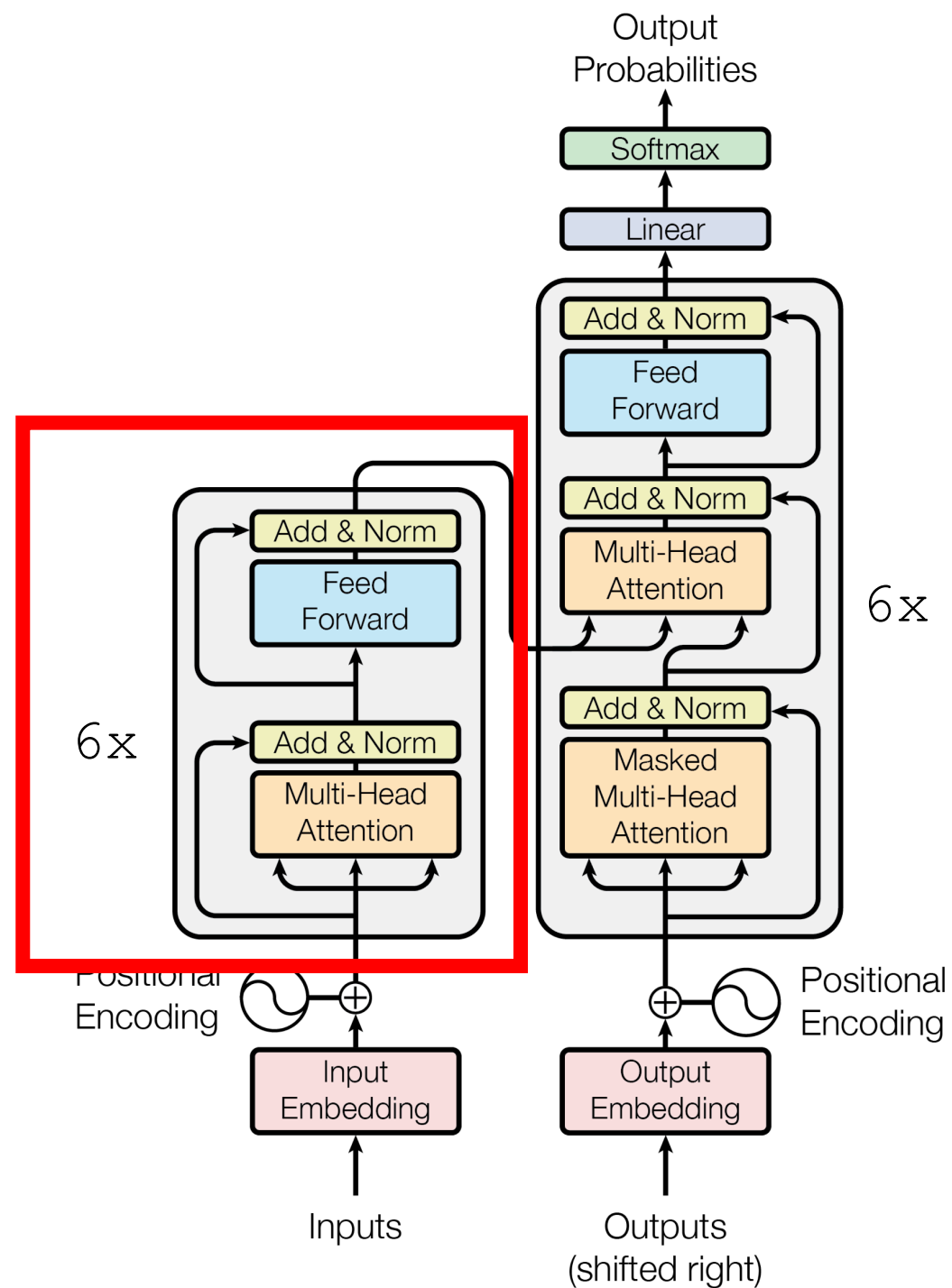
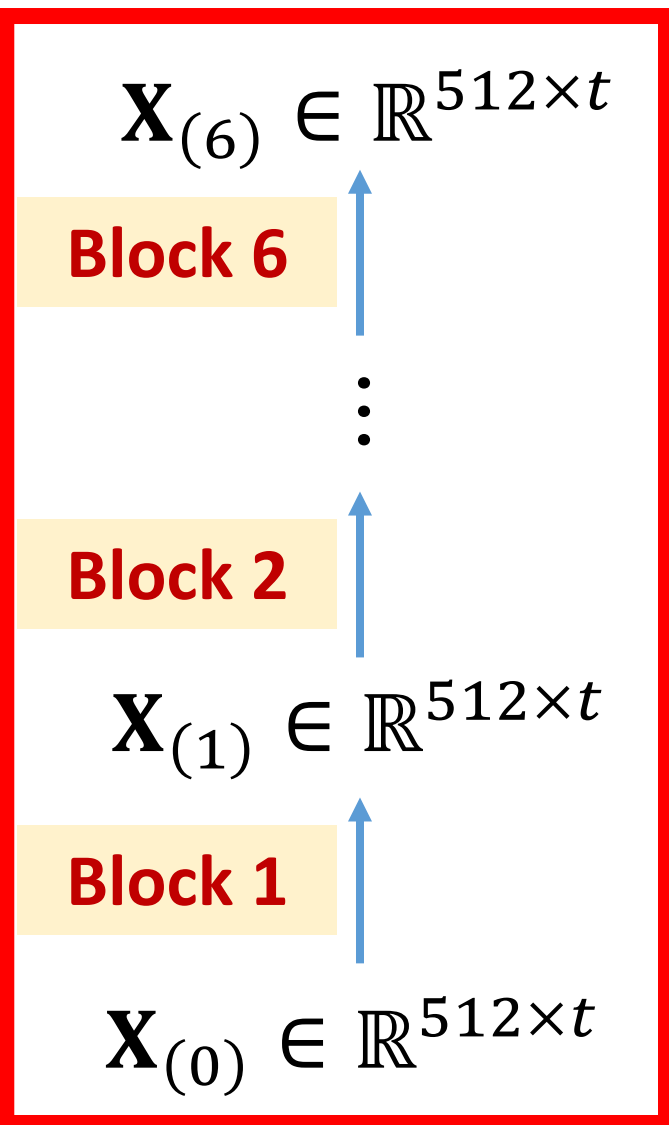
Block 1

$$\mathbf{X}_{(0)} \in \mathbb{R}^{512 \times t}$$



Encoder Network

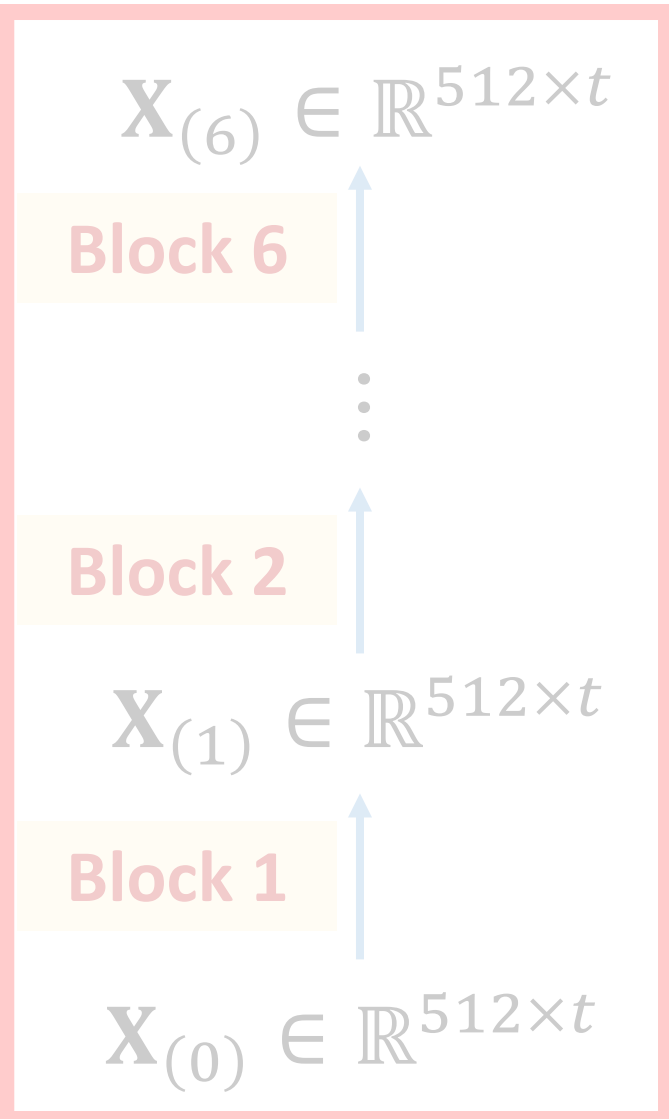
Encoder



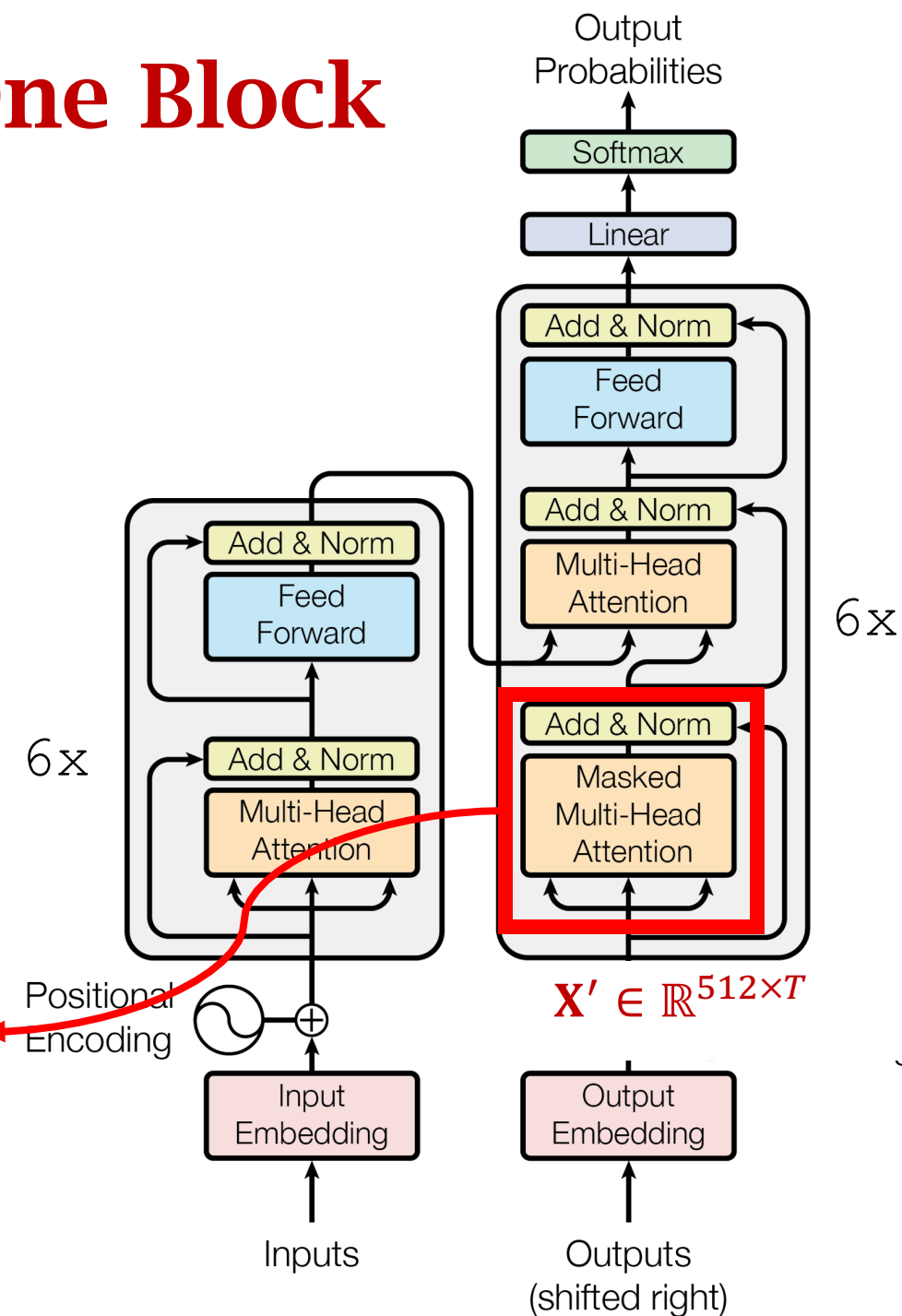
Decoder of Transformer

Decoder Network: One Block

Encoder

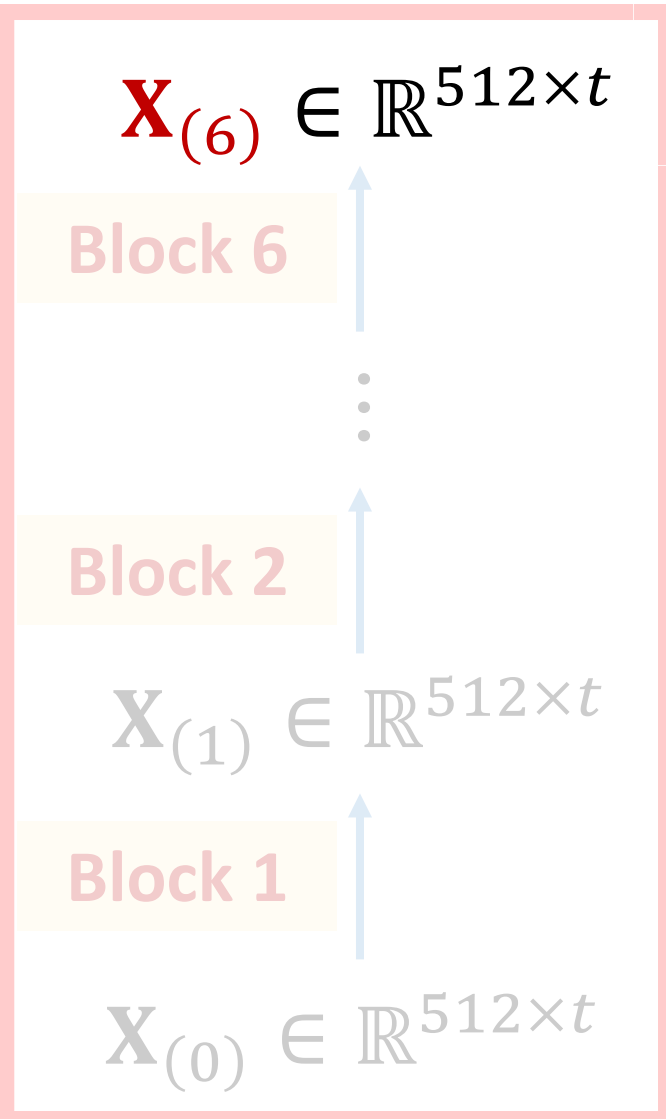


- Similar to encoder.
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}'$.



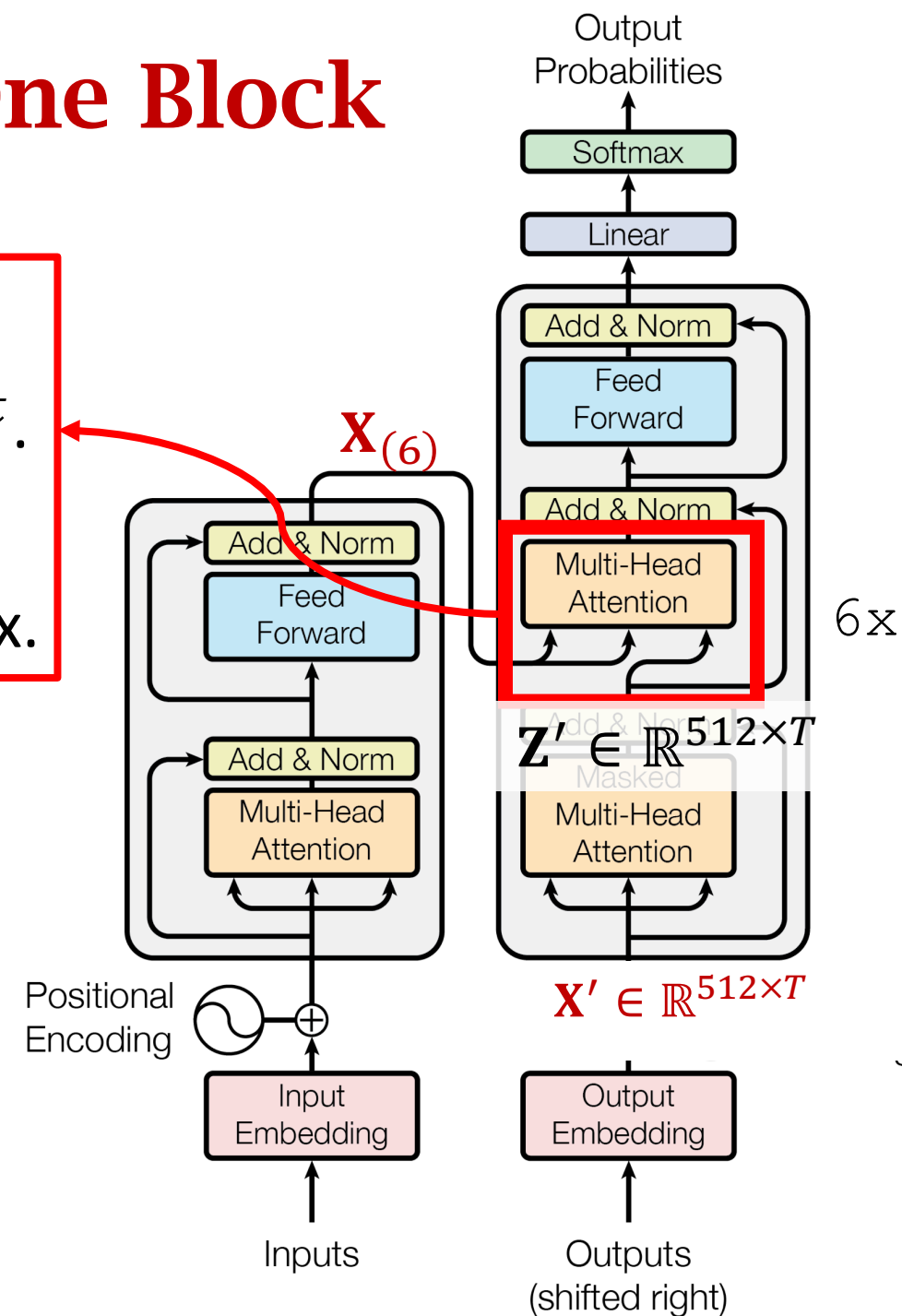
Decoder Network: One Block

Encoder



- Set $\mathbf{Q} = \mathbf{Z}' \in \mathbb{R}^{512 \times T}$.
- $\mathbf{K} = \mathbf{V} = \mathbf{X}_{(6)} \in \mathbb{R}^{512 \times t}$.
- Multi-head attention outputs a $512 \times T$ matrix.

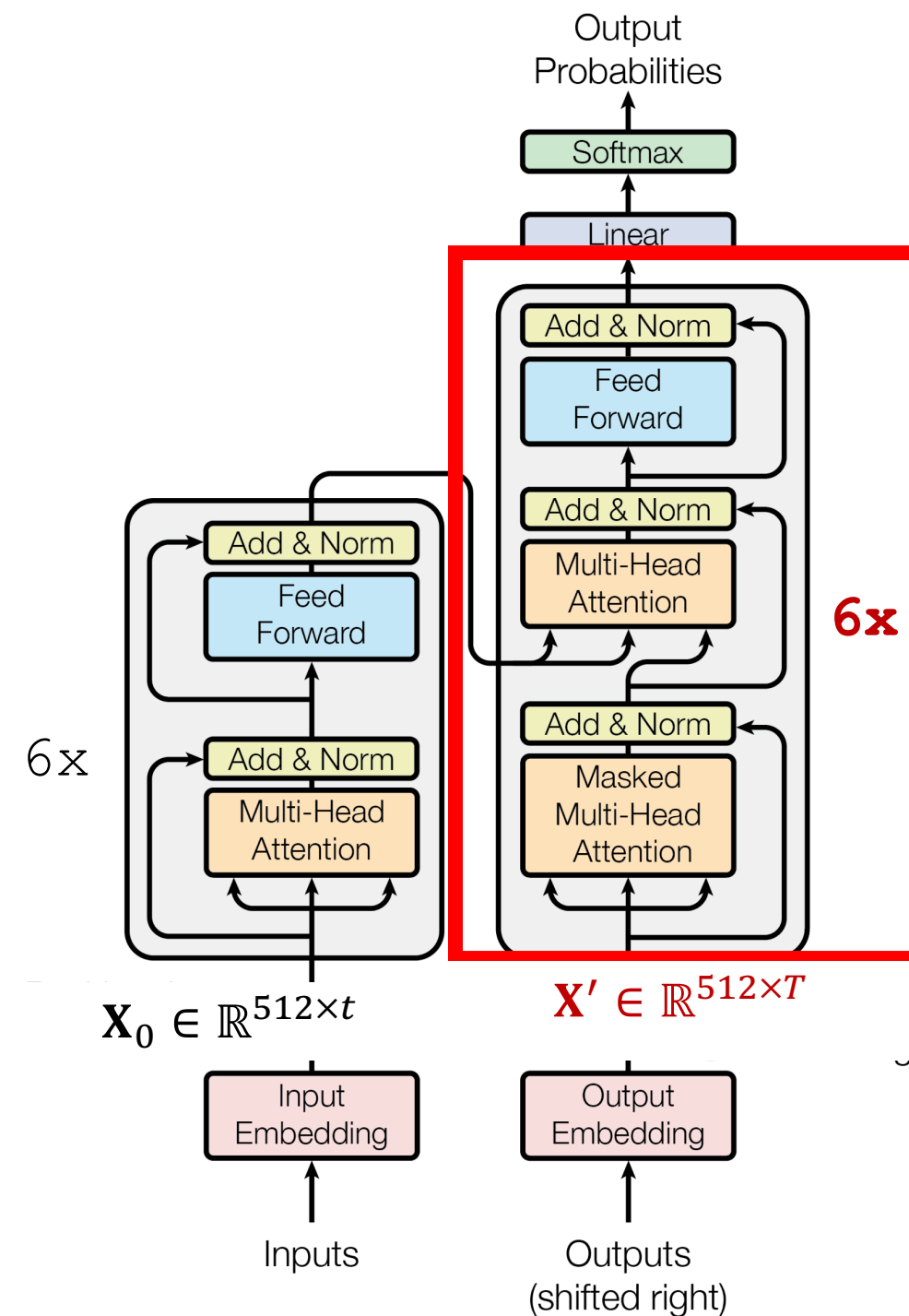
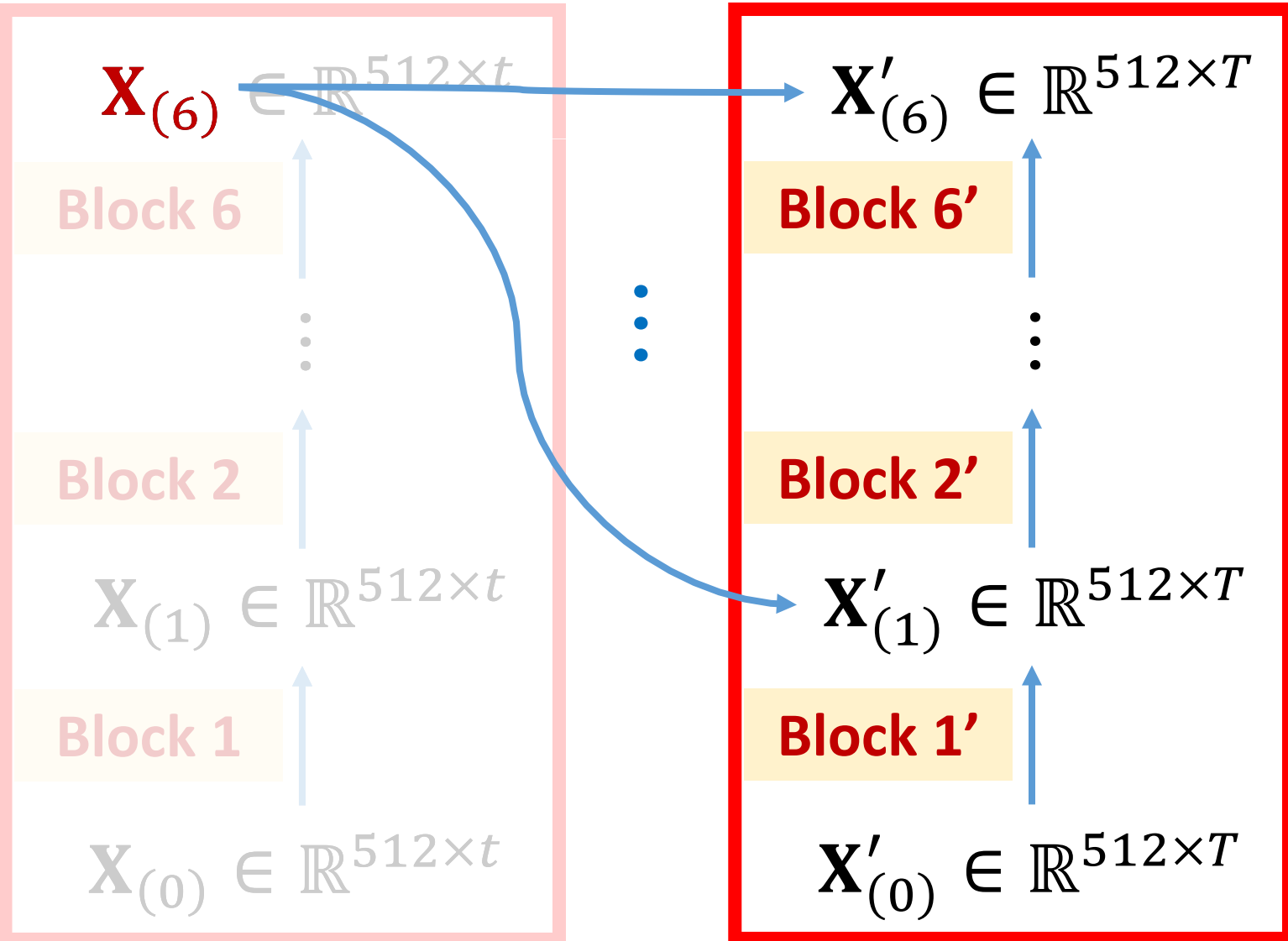
- Similar to encoder.
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}'$.



Decoder Network

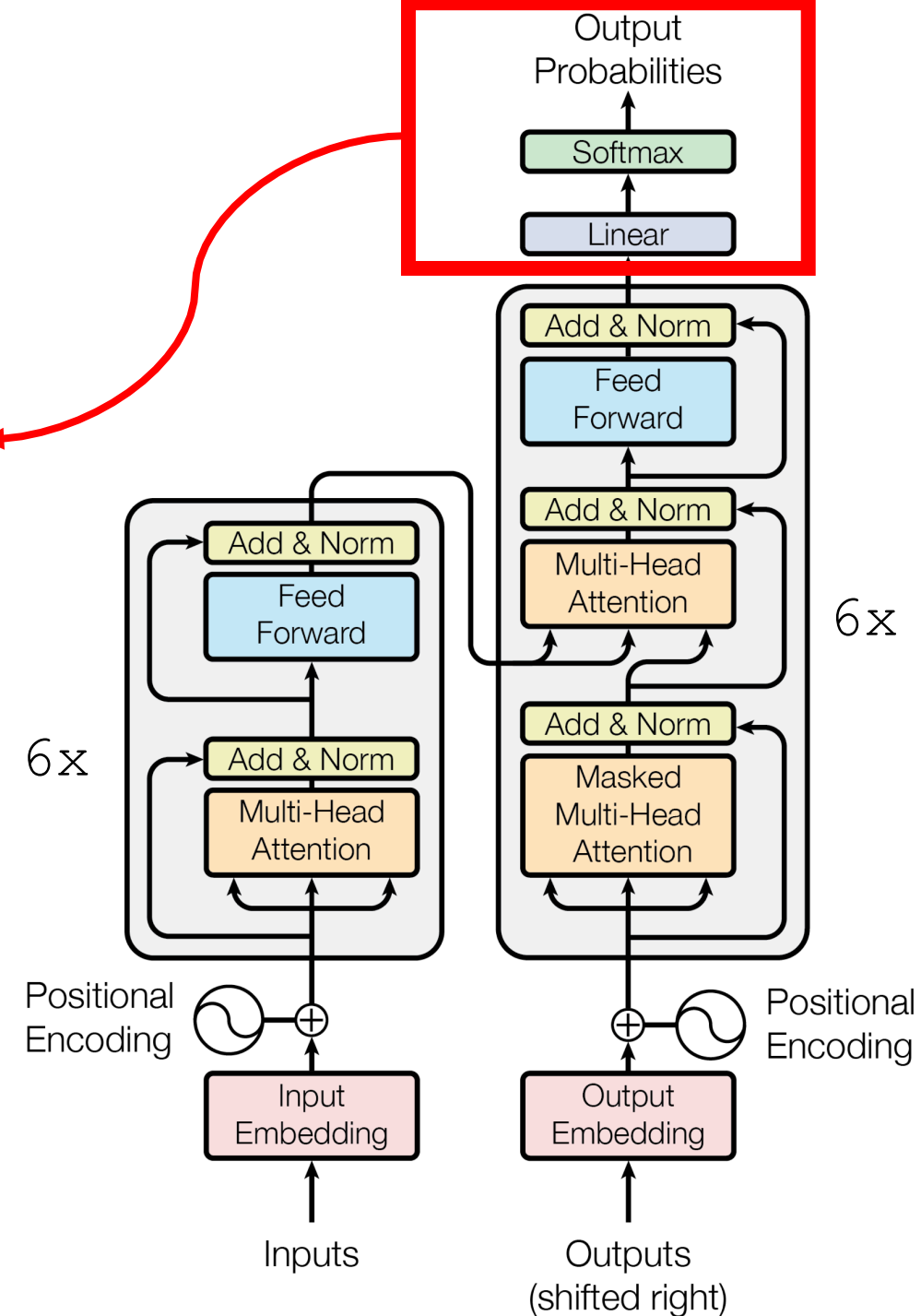
Encoder

Decoder



Decoder Network

- Output a distribution over the vocabulary.
- Compare the distribution with the one-hot encode of the label.
- ➔ Loss, e.g., cross-entropy.
- ➔ Gradient.
- ➔ Update model parameters.



Summary

Summary

- Transformer model is **not RNN**.
 - Transformer is based on **attention** and **self-attention**.
 - **Upside**: Outperform all the state-of-the-art RNN models.
 - **Downside**: Much more expensive than RNN models.
-
- Read the original paper: Vaswani et al. [Attention Is All You Need](#). In *NIPS*, 2017.
 - Google “*transformer model explained*” and read the articles.

Key Concept: Multi-Head Attention

- Inputs: query \mathbf{Q} , key \mathbf{K} , and value \mathbf{V} .
- Linear maps: $\tilde{\mathbf{Q}} = \mathbf{W}_q \mathbf{Q}$, $\tilde{\mathbf{K}} = \mathbf{W}_k \mathbf{K}$, and $\tilde{\mathbf{V}} = \mathbf{W}_v \mathbf{V}$.
- Single-head attention:

$$\mathbf{C} = \text{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \tilde{\mathbf{V}} \cdot \text{softmax}(\tilde{\mathbf{K}}^T \tilde{\mathbf{Q}}).$$

Key Concept: Multi-Head Attention

- Inputs: query \mathbf{Q} , key \mathbf{K} , and value \mathbf{V} .
- Linear maps: $\tilde{\mathbf{Q}} = \mathbf{W}_q \mathbf{Q}$, $\tilde{\mathbf{K}} = \mathbf{W}_k \mathbf{K}$, and $\tilde{\mathbf{V}} = \mathbf{W}_v \mathbf{V}$.
- Single-head attention:

$$\mathbf{C} = \text{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \tilde{\mathbf{V}} \cdot \text{softmax}(\tilde{\mathbf{K}}^T \tilde{\mathbf{Q}}).$$

- Multi-head attention:
 - Repeat $\text{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ using different parameters $\mathbf{W}_q, \mathbf{W}_v, \mathbf{W}_v$.
 - Get $\mathbf{C}^{[1]}, \mathbf{C}^{[2]}, \dots, \mathbf{C}^{[m]} \in \mathbb{R}^{d_z \times t}$.
 - Concatenate the m matrices to get $\tilde{\mathbf{C}} \in \mathbb{R}^{md_z \times t}$.

Attention in the **encoder**:

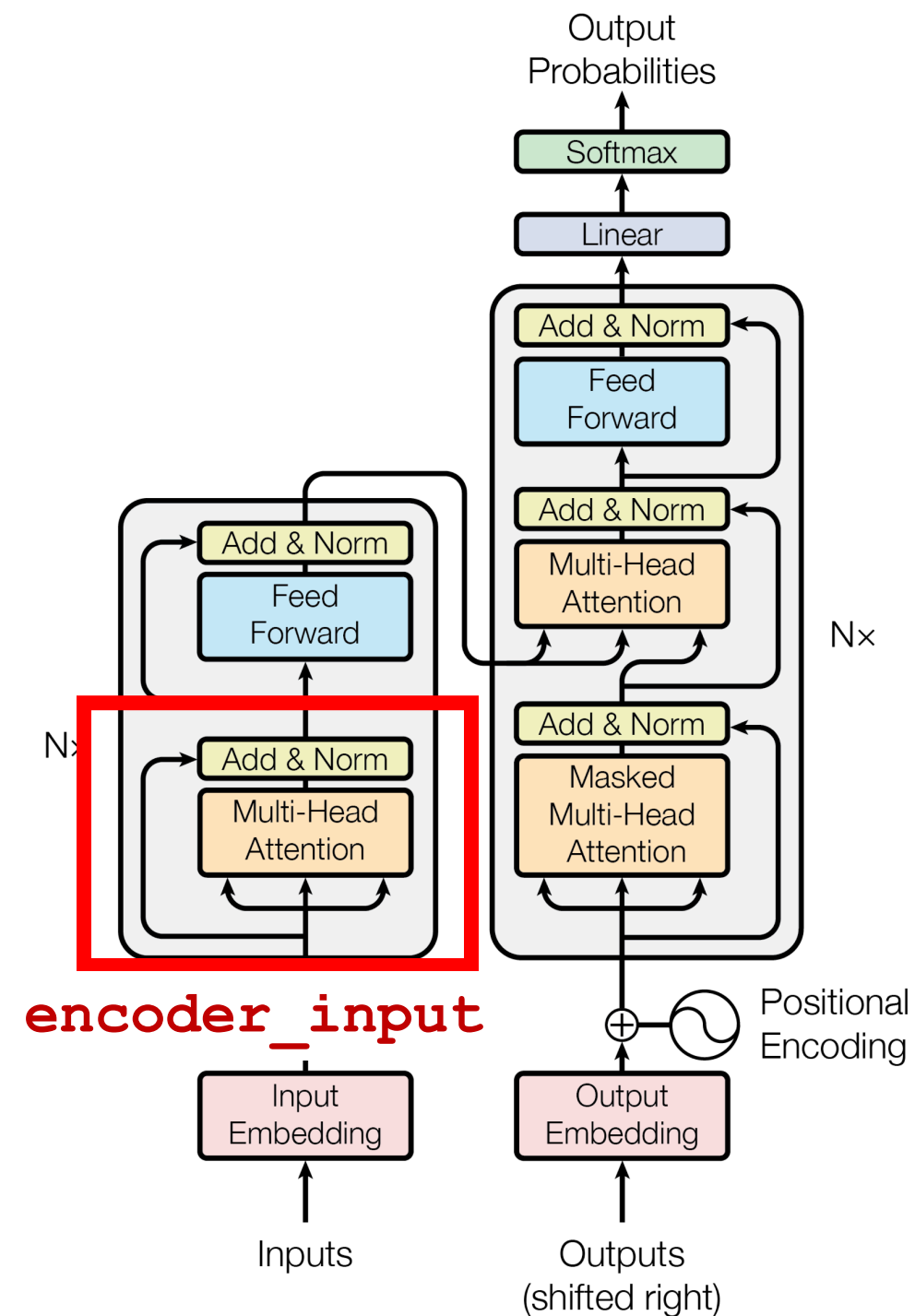
- $Q = K = V = \text{encoder_input}$.

1st attention in the **decoder**:

- $Q = K = V = \text{decoder_input}$.

2nd attention in the **decoder**

- $Q = \text{decoder_input}$
- $K = V = \text{encoder_output}$.



Attention in the **encoder**:

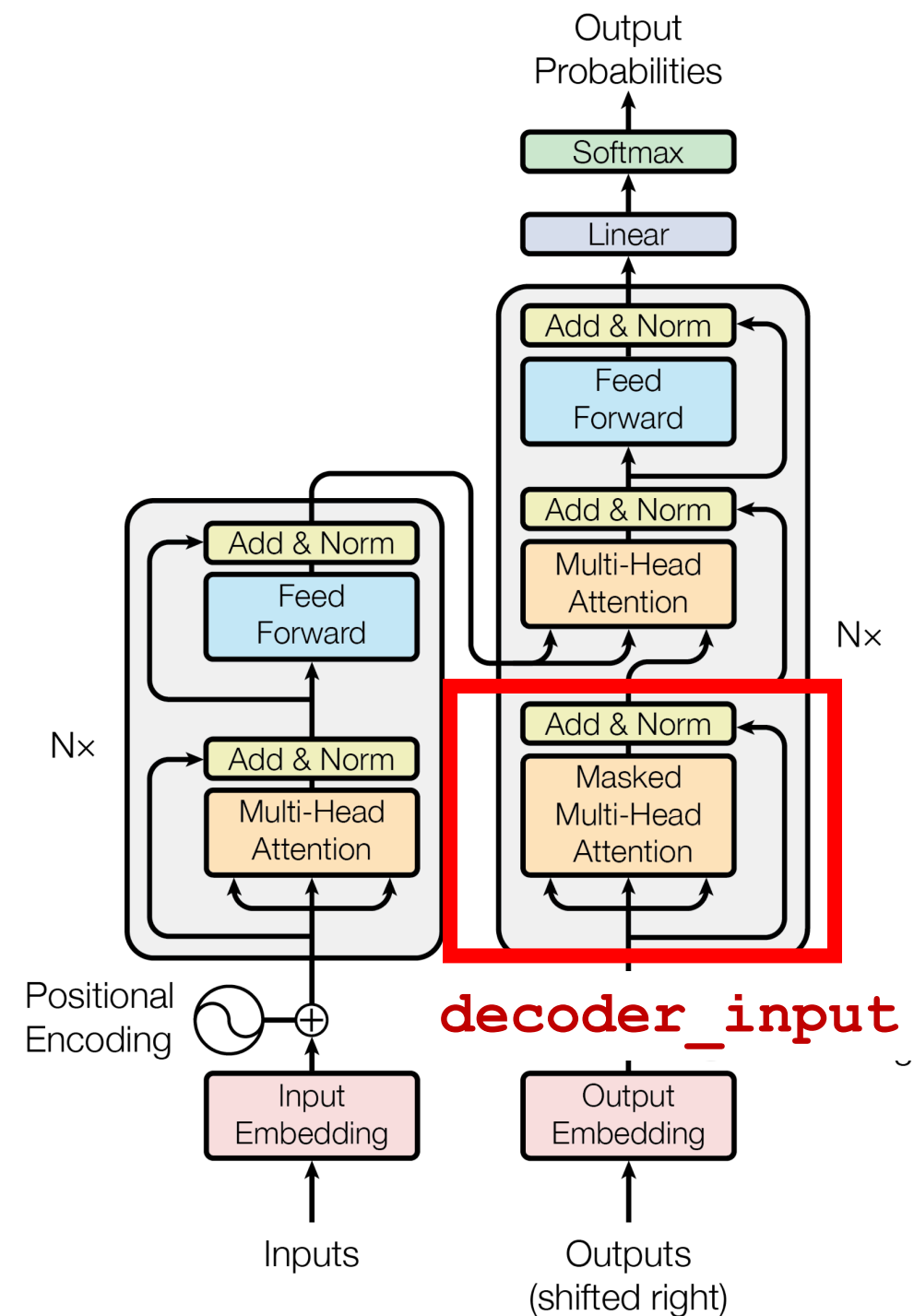
- $Q = K = V = \text{encoder_input}.$

1st attention in the **decoder**:

- $Q = K = V = \text{decoder_input}.$

2nd attention in the **decoder**

- $Q = \text{decoder_input}$
- $K = V = \text{encoder_output}.$



Attention in the **encoder**:

- $Q = K = V = \text{encoder_input}.$

1st attention in the **decoder**:

- $Q = K = V = \text{decoder_input}.$

2nd attention in the **decoder**

- $Q = \text{decoder_input}$
- $K = V = \text{encoder_output}.$

