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

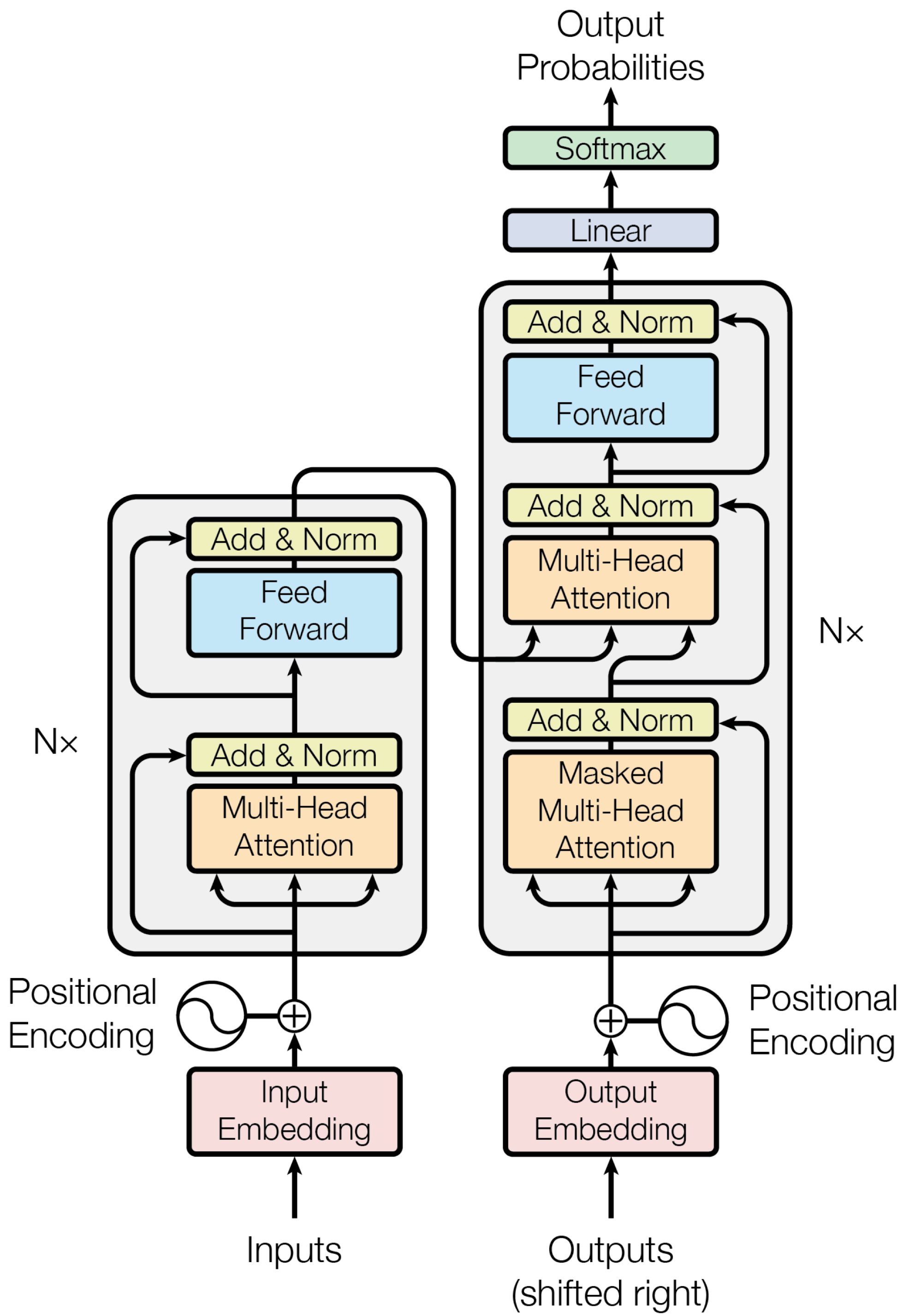
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

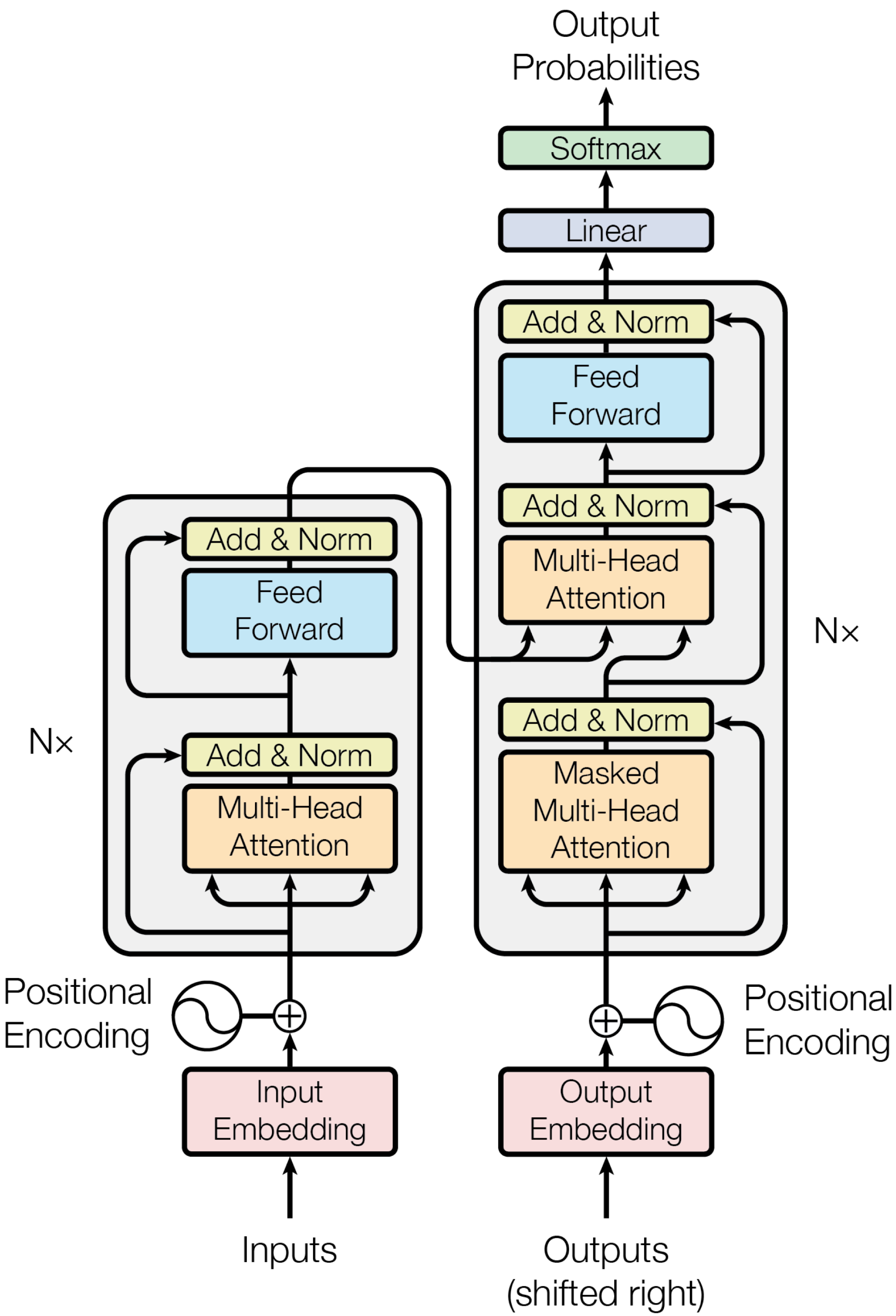
Illia Polosukhin* ‡ illia.polosukhin@gmail.com



Transformer Model

- Transformer is a Seq2Seq model.
- Transformer is not RNN.
- Purely based attention and fully-connected layers.

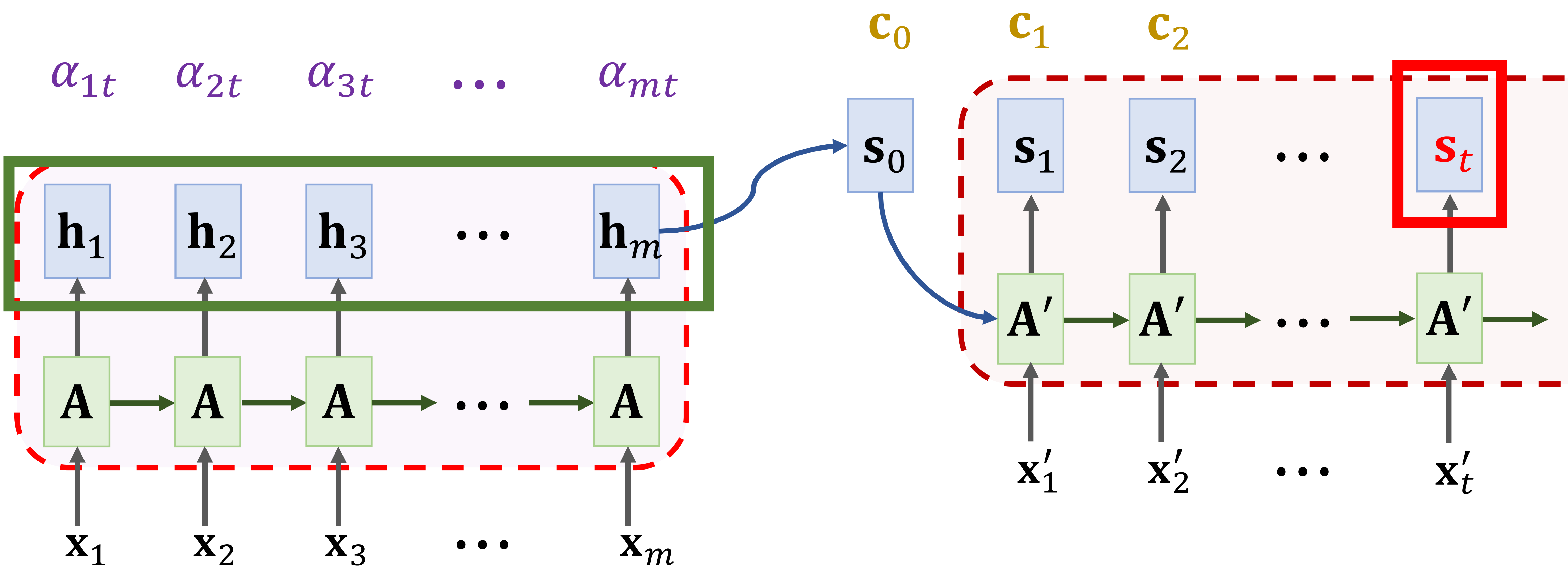
- Much more computations than RNNs.
- Higher performance than RNNs on large datasets.



Revisit Attention

Attention for RNN

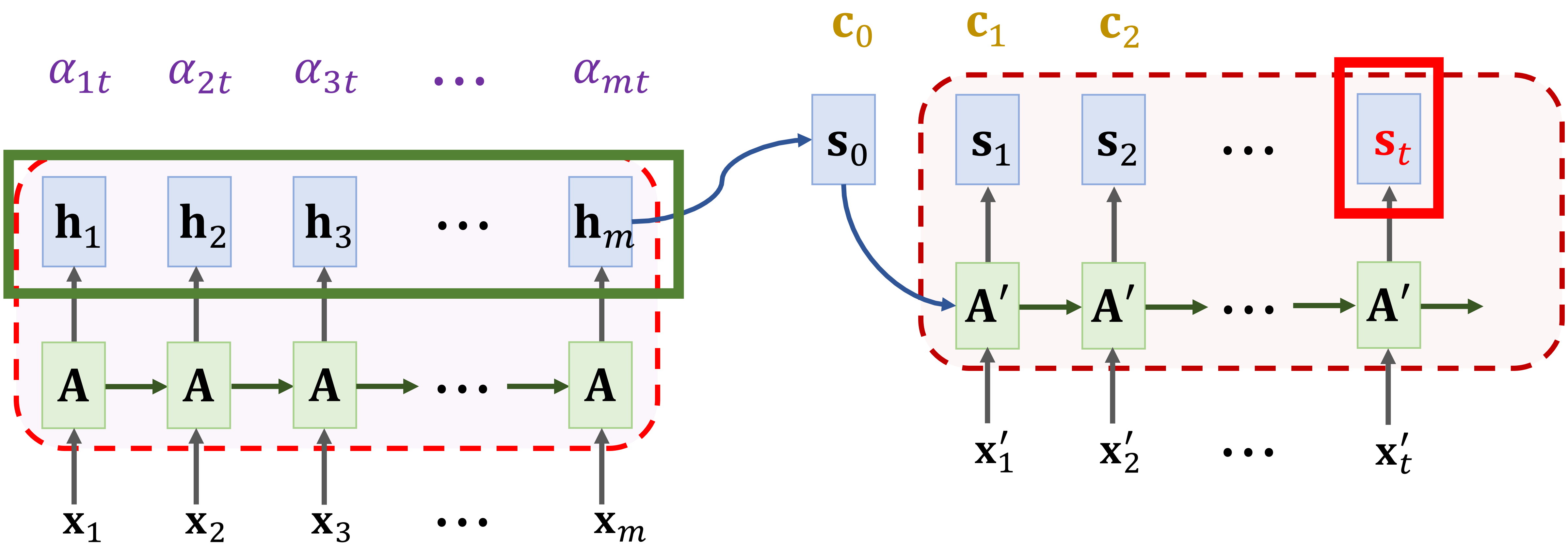
Weights: $\alpha_{it} = \text{similarity}(\mathbf{h}_i, \mathbf{s}_t)$



Attention for RNN

Weights: $\alpha_{it} = \text{similarity}(\mathbf{h}_i, \mathbf{s}_t)$

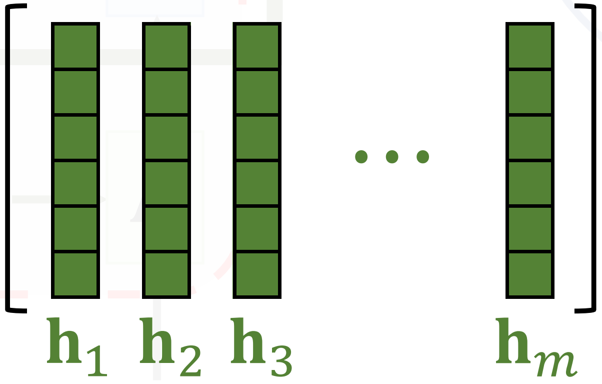
Here, $\mathbf{h}_i$ is called "key", and $\mathbf{s}_t$ is called "query".



Attention for RNN

Weights: $\alpha_{it} = \text{similarity}(\mathbf{h}_i, \mathbf{s}_t)$

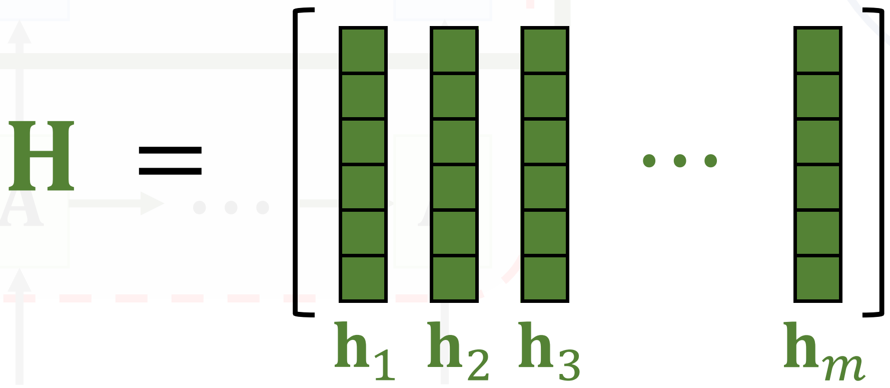
- Define $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_m] \in \mathbb{R}^{d \times m}$.
- Compute weights: $\boldsymbol{\alpha}_{:t} = \text{Softmax}\Big((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \Big) \in \mathbb{R}^m$.
- $\alpha_{1t}, \alpha_{2t}, \cdots, \alpha_{mt}$ are the entries of vector $\boldsymbol{\alpha}_{:t} \in \mathbb{R}^m$.

$\mathbf{H} =$ 

Attention for RNN

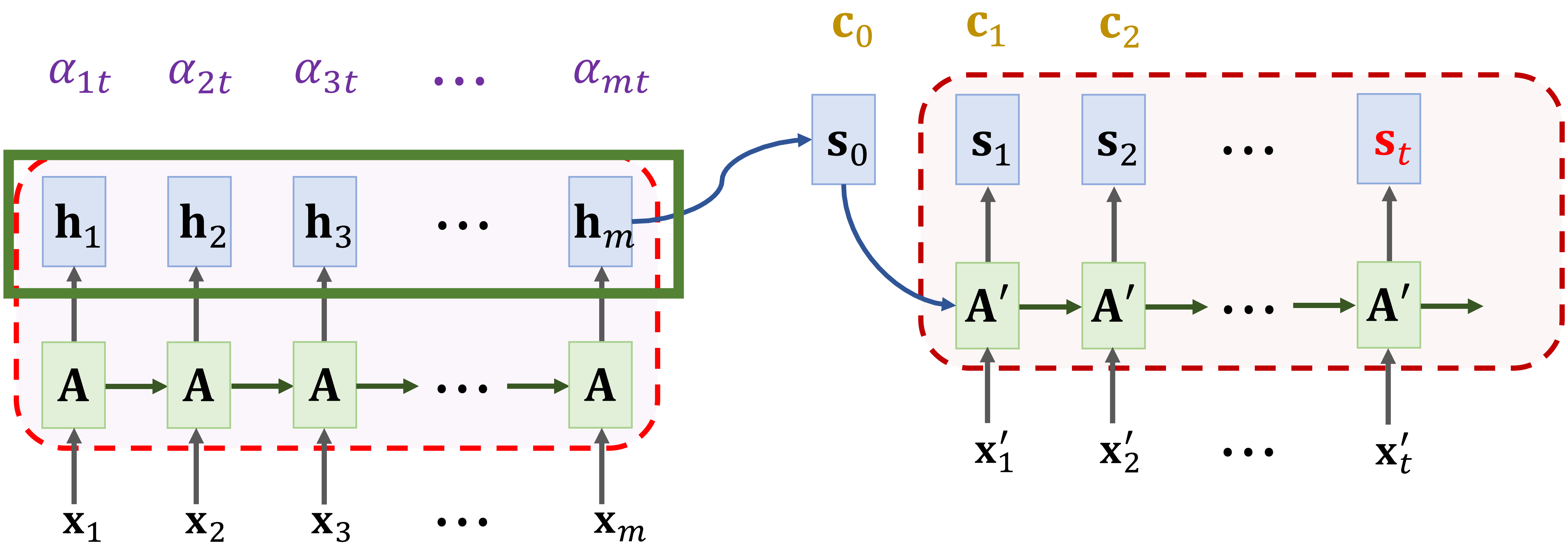
Weights: $\boldsymbol{\alpha}_{:t} = \text{Softmax}\left((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \right)$

- Define $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_m] \in \mathbb{R}^{d \times m}$.
- Compute weights: $\boldsymbol{\alpha}_{:t} = \text{Softmax}\left((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \right) \in \mathbb{R}^m$.
- $\alpha_{1t}, \alpha_{2t}, \cdots, \alpha_{mt}$ are the entries of vector $\boldsymbol{\alpha}_{:t} \in \mathbb{R}^m$.



Attention for RNN

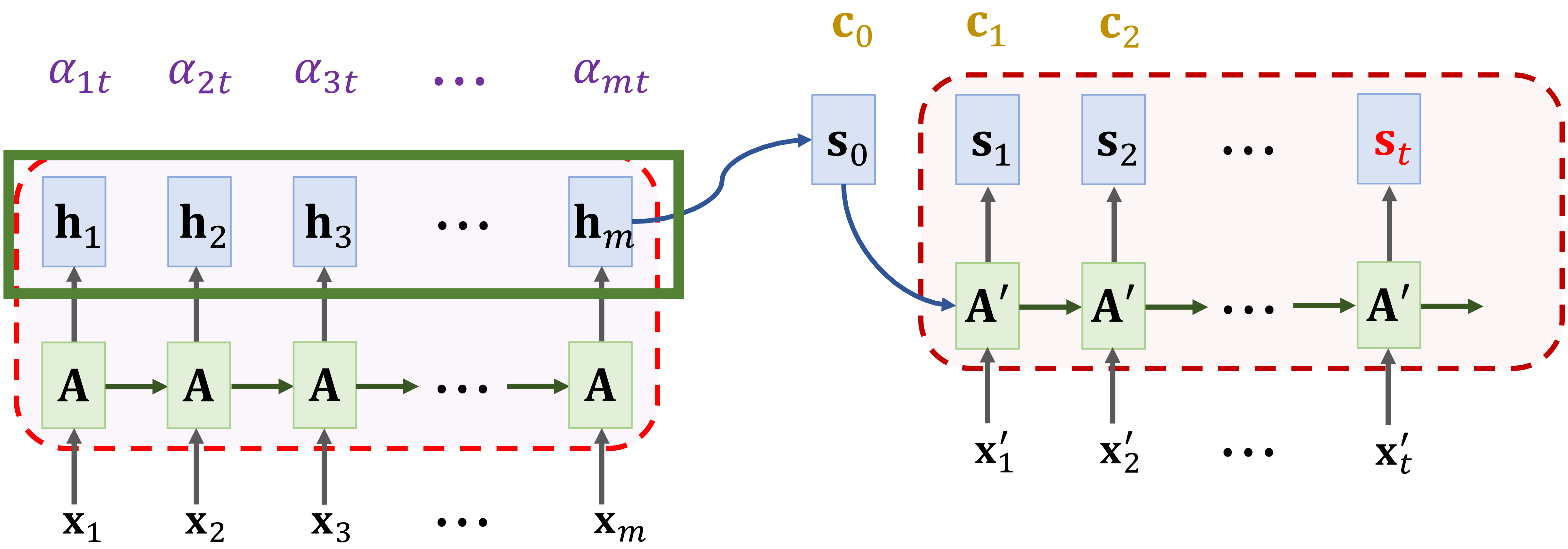
Weights: $\boldsymbol{\alpha}_{:t} = \mathrm{Softmax}\Big((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \Big)$



Attention for RNN

Weights: $\boldsymbol{\alpha}_{:t} = \text{Softmax}\left((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \right)$

Context vector: $\mathbf{c}_t = \alpha_{1t}\mathbf{h}_1 + \cdots + \alpha_{mt}\mathbf{h}_m.$



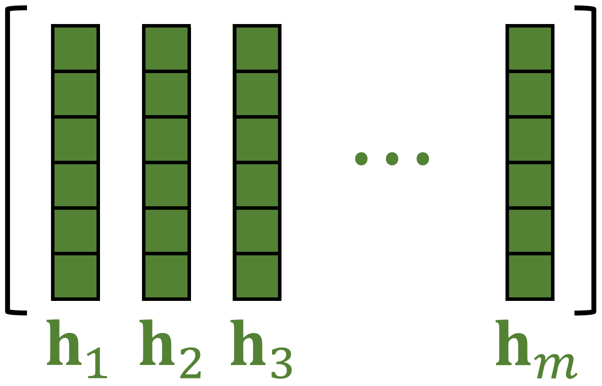
Attention for RNN

Weights: $\boldsymbol{\alpha}_{:t} = \mathrm{Softmax}\Big(\big(\mathbf{W}_K \mathbf{H}\big)^T \big(\mathbf{W}_Q \mathbf{s}_t\big)\Big)$

Context vector: $\mathbf{c}_t = \alpha_{1t}\mathbf{h}_1 + \cdots + \alpha_{mt}\mathbf{h}_m.$

- Note that $\mathbf{H}\,\boldsymbol{\alpha}_{:t} = \alpha_{1t}\mathbf{h}_1 + \cdots + \alpha_{mt}\mathbf{h}_m.$
- Thus $\mathbf{c}_t = \alpha_{1t}\mathbf{h}_1 + \cdots + \alpha_{mt}\mathbf{h}_m = \mathbf{H}\,\boldsymbol{\alpha}_{:t}.$

$\mathbf{H} =$



Attention for RNN

Weights: $\boldsymbol{\alpha}_{:t} = \mathrm{Softmax}\left((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \right)$

Context vector: $\mathbf{c}_t = \mathbf{H}\, \boldsymbol{\alpha}_{:t}.$

Attention for RNN

Weights: $\boldsymbol{\alpha}_{:t} = \mathrm{Softmax}\left((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \right)$

Context vector: $\mathbf{c}_t = \mathbf{W}_V \mathbf{H} \, \boldsymbol{\alpha}_{:t}$.

A different way to computing context vector.

Single-Head & Multi-Head Attention

Attention for RNN

Context vector: $\mathbf{c}_t = (\mathbf{W}_V \mathbf{H}) \cdot \mathrm{Softmax}\left((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t) \right).$

Single-Head Attention

Context vector: $\mathbf{c}_t = (\mathbf{W}_V \mathbf{H}) \cdot \text{Softmax}\Big((\mathbf{W}_K \mathbf{H})^T (\mathbf{W}_Q \mathbf{s}_t)\Big).$

Single-Head Attention:

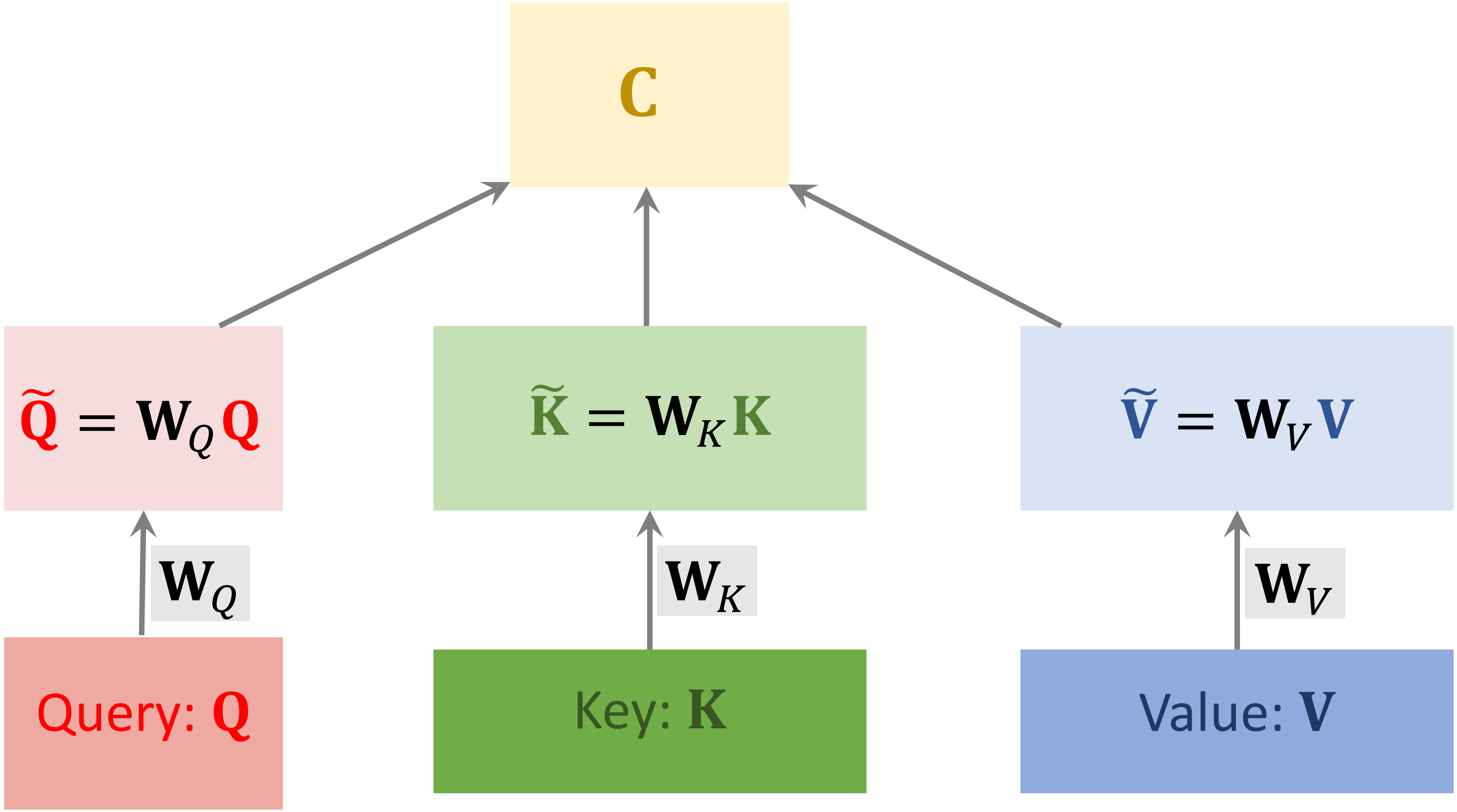
- $\mathbf{C} = \text{Attn}(\mathbf{Q}, \ \mathbf{K}, \ \mathbf{V}).$

  query key value

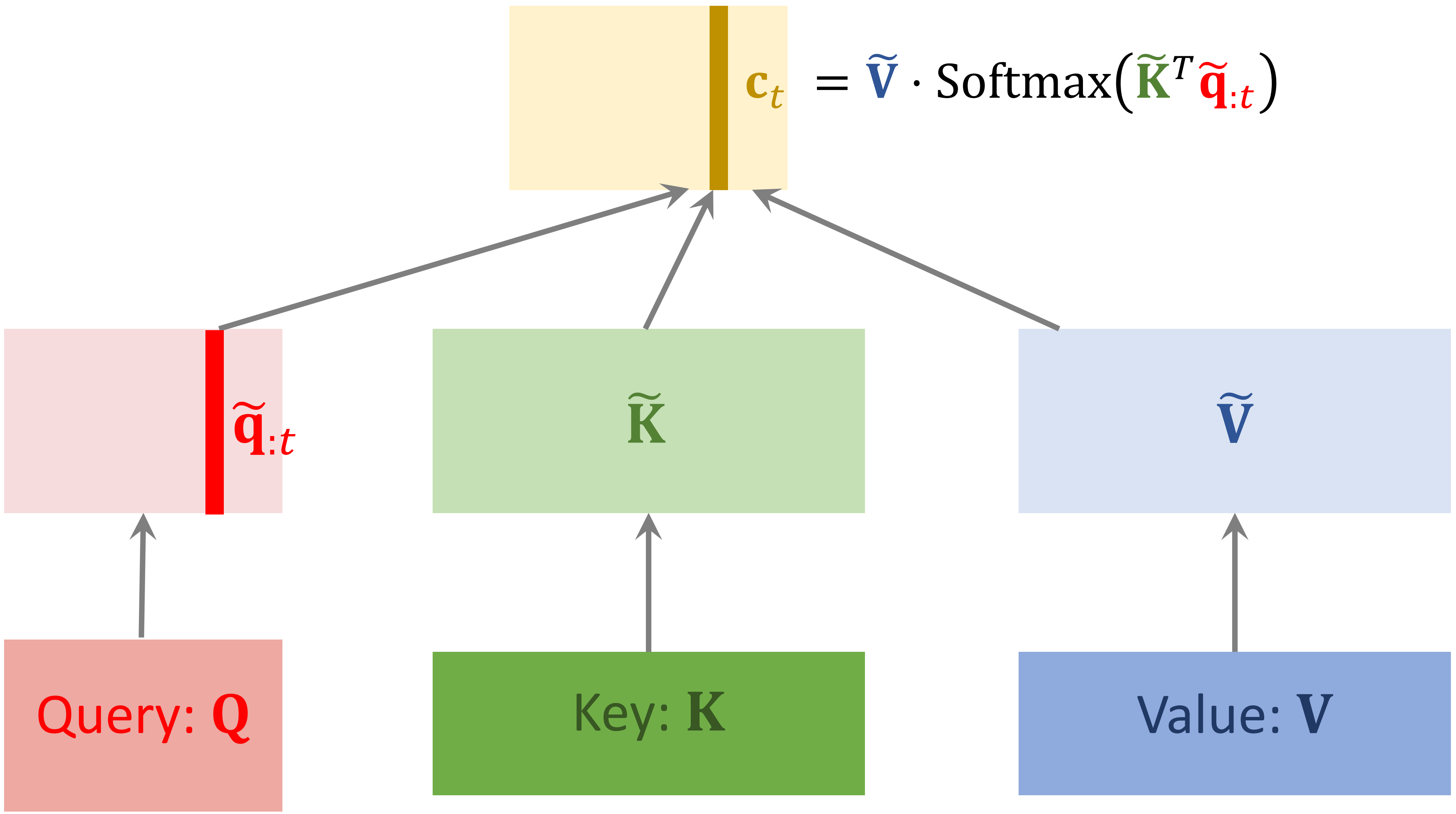
- The t-th column of $\mathbf{C}$ is:

$$\mathbf{c}_t = (\mathbf{W}_V \mathbf{V}) \cdot \text{Softmax}\Big((\mathbf{W}_K \mathbf{K})^T (\mathbf{W}_Q \mathbf{q}_{:t})\Big).$$

Single-Head Attention



Single-Head Attention



Multi-Head Attention

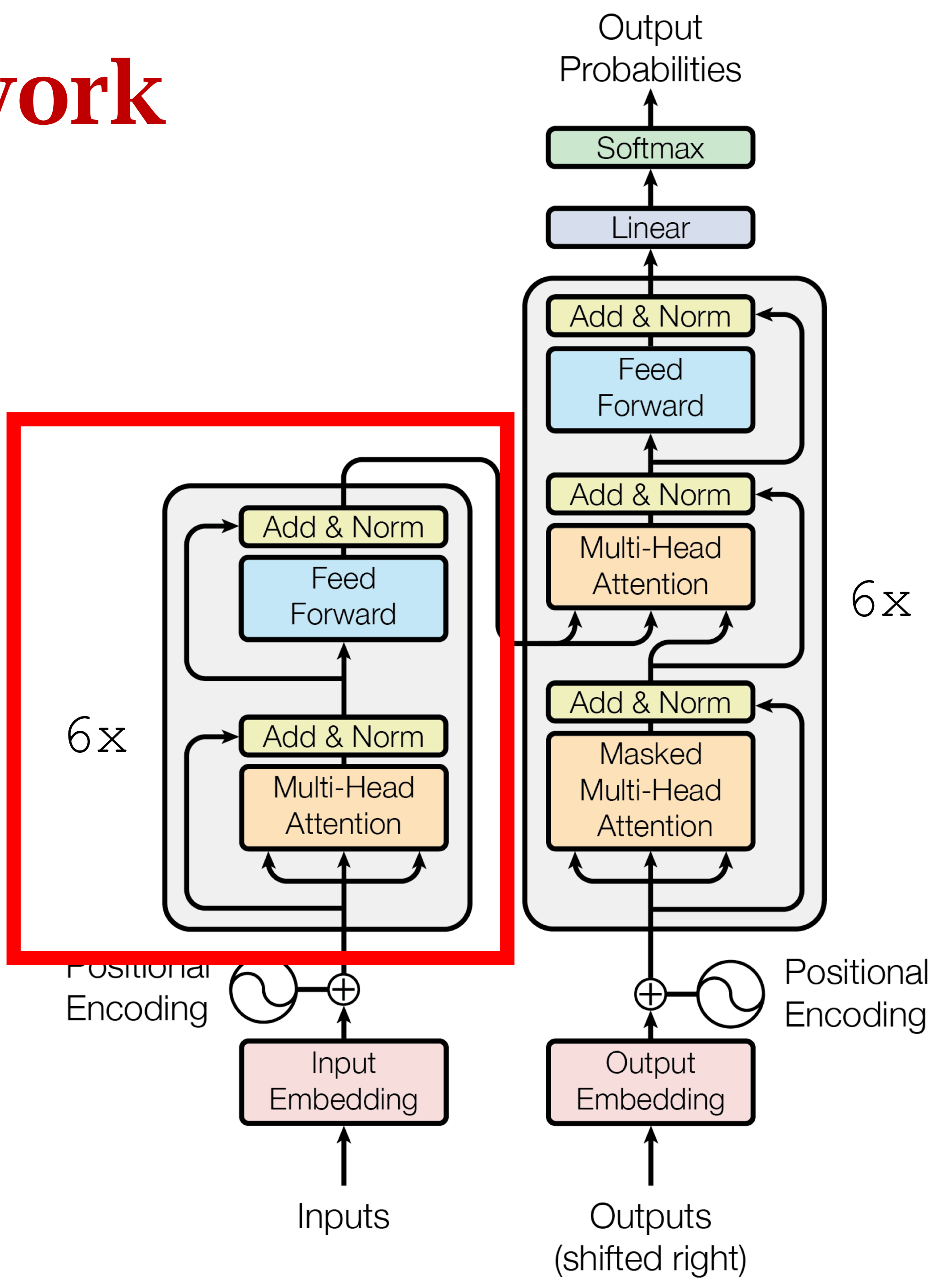
- **Single-head attention:** $\mathbf{C} = \text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$.
  - Its t-th column is $\mathbf{c}_t = (\mathbf{W}_V\mathbf{V}) \cdot \text{Softmax}\left((\mathbf{W}_K\mathbf{K})^T(\mathbf{W}_Q\mathbf{q}_{:t})\right)$.
  - $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are trainable parameters.
  - Output shape: $d_c \times t$. (Matrix $\mathbf{Q}$ has t columns.)

- **Multi-head attention:**
  - Using l single-head attentions (which do not share parameters.)
  - Totally $3l$ parameters matrices $\mathbf{W}$.
  - Concatenating the output $\mathbf{C}$ matrices of the single-head attentions.
  - Output shape: $(ld_c) \times t$.

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

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

Concatenation

$$\mathbf{C}^{[1]} \in \mathbb{R}^{d_c \times t} \qquad \mathbf{C}^{[2]} \in \mathbb{R}^{d_c \times t} \qquad \cdots \qquad \mathbf{C}^{[l]} \in \mathbb{R}^{d_c \times t}$$

Attention w/ different parameter matrices.

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

Multi-Head Attention + Dense Layer

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

Multi-Head Attention

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

Concatenation

$$\mathbf{C}^{[1]} \in \mathbb{R}^{d_c \times t} \qquad \mathbf{C}^{[2]} \in \mathbb{R}^{d_c \times t} \qquad \cdots \qquad \mathbf{C}^{[l]} \in \mathbb{R}^{d_c \times t}$$

Attention w/ different parameter matrices.

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

Multi-Head Attention + Dense Layer

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

Dense layer is applied to every column independently.

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

Concatenation

$\mathbf{C}^{[1]} \in \mathbb{R}^{d_c \times t}$ $\quad$ $\mathbf{C}^{[2]} \in \mathbb{R}^{d_c \times t}$ $\quad \cdots \quad$ $\mathbf{C}^{[l]} \in \mathbb{R}^{d_c \times t}$

Attention w/ different parameter matrices.

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

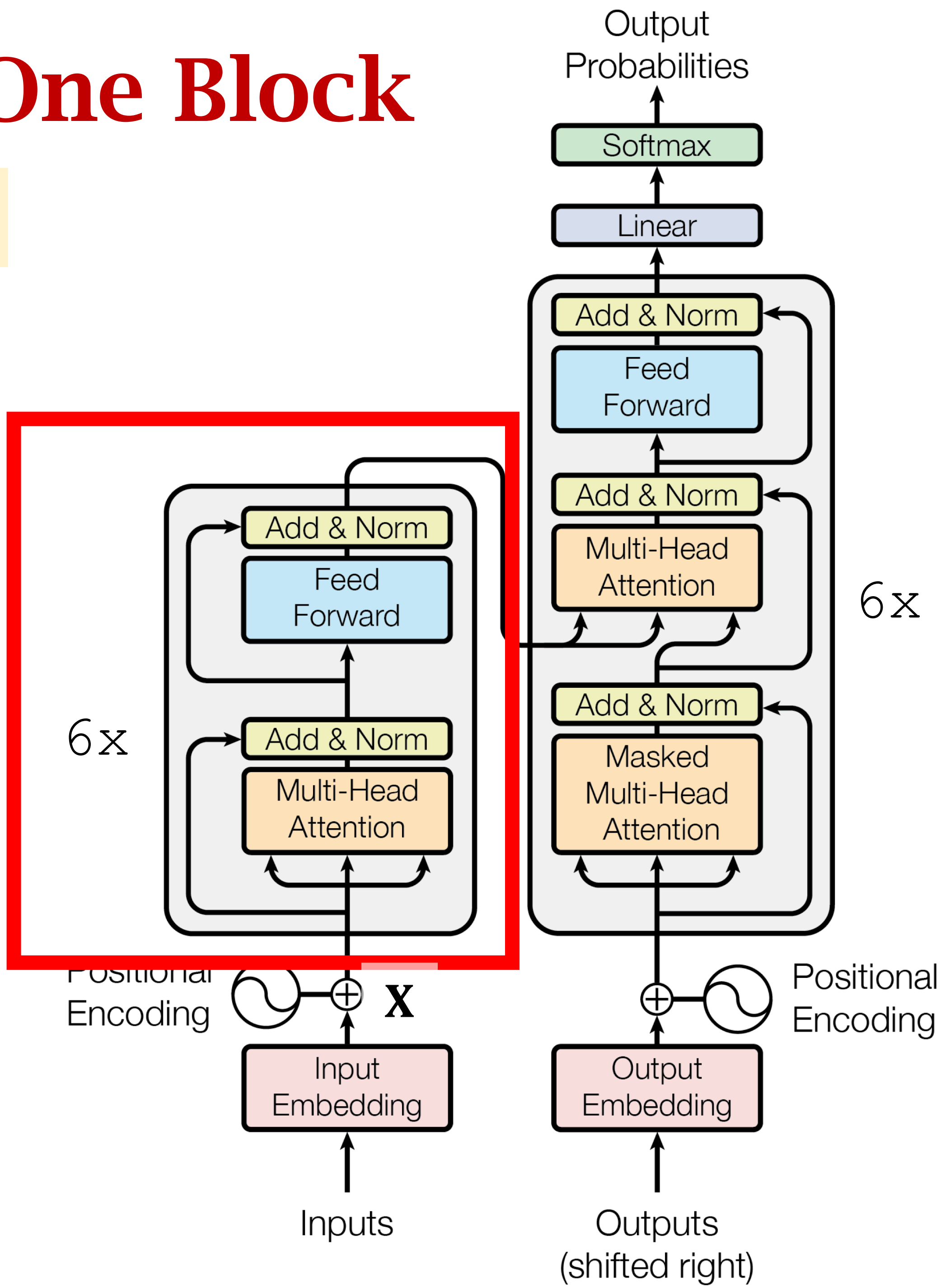
Encoder Network: One Block

Ignore skip connection and normalization.

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

query key value

Similar to self-attention.



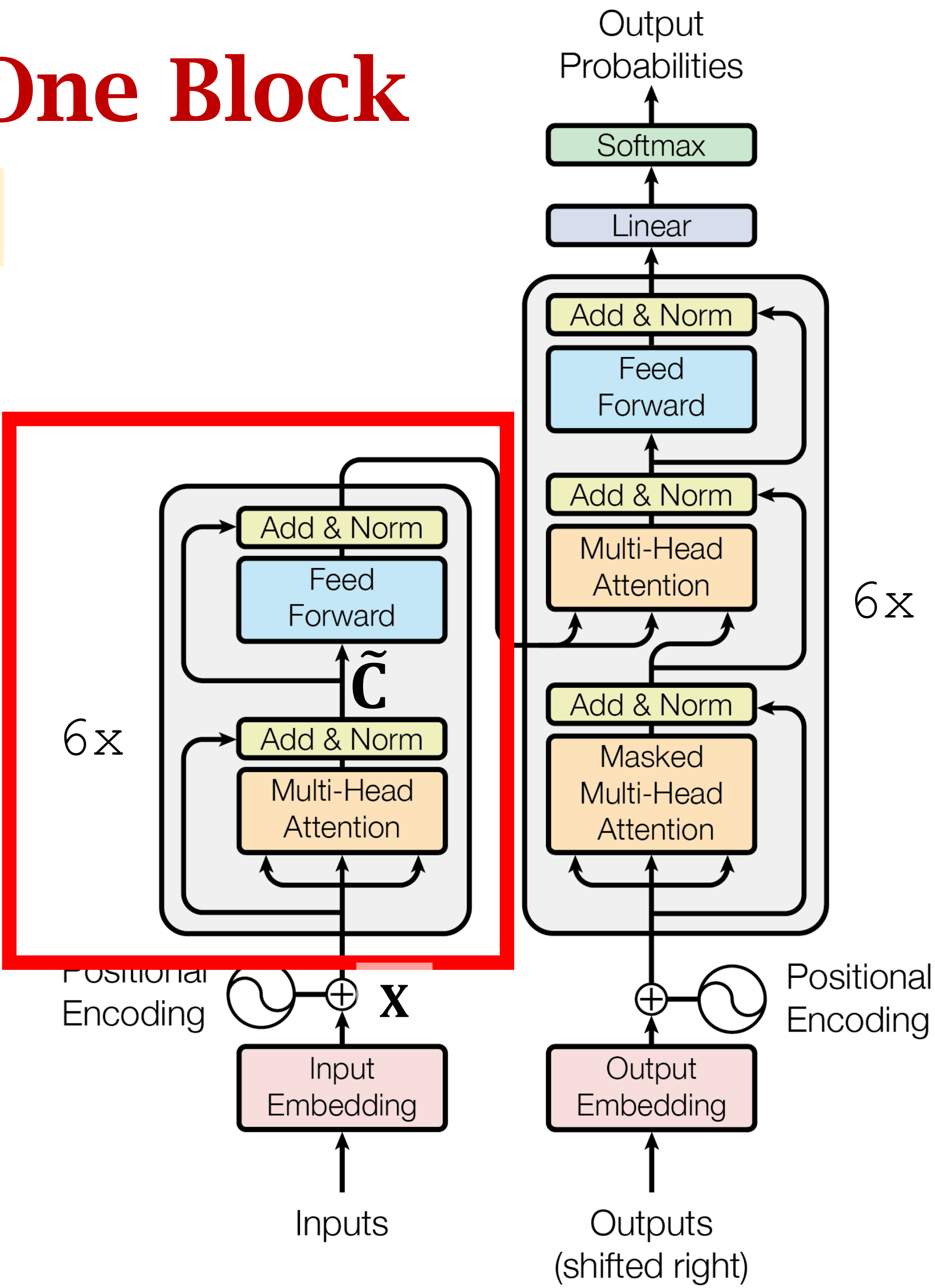
Encoder Network: One Block

Ignore skip connection and normalization.

- Input: $\mathbf{X} \in \mathbb{R}^{512 \times m}$; ($m$ is the seq length.)
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$.
- Repeat single-head attention $l = 8$ times:

$$\mathbf{C}^{[i]} = \text{Attn}(\mathbf{Q},\ \mathbf{K},\ \mathbf{V}) \in \mathbb{R}^{64 \times m}.$$

- $\tilde{\mathbf{C}} = \text{Concatenate}\left(\mathbf{C}^{[1]}, \cdots, \mathbf{C}^{[l]}\right) \in \mathbb{R}^{512 \times m}$.



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

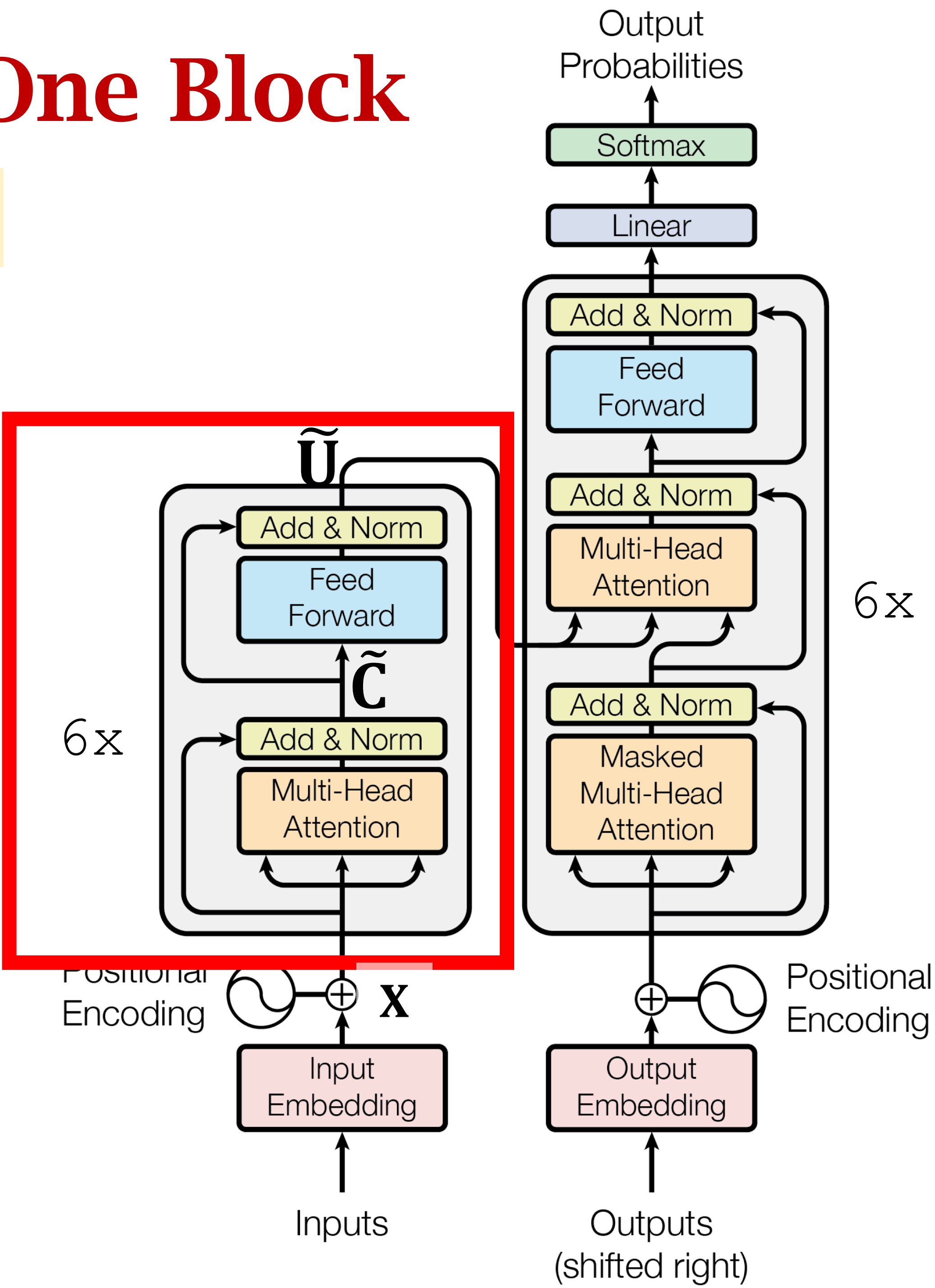
Encoder Network: One Block

Ignore skip connection and normalization.

- Input: $\mathbf{X} \in \mathbb{R}^{512\times m}$; ($m$ is the seq length.)
- Set $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$.
- Repeat single-head attention $m = 8$ times:

$$\mathbf{C}^{[i]} = \text{Attn}(\mathbf{Q},\ \mathbf{K},\ \mathbf{V}) \in \mathbb{R}^{64\times m}.$$

- $\tilde{\mathbf{C}} = \text{Concatenate}\left(\mathbf{C}^{[1]}, \cdots, \mathbf{C}^{[l]}\right) \in \mathbb{R}^{512\times m}$.

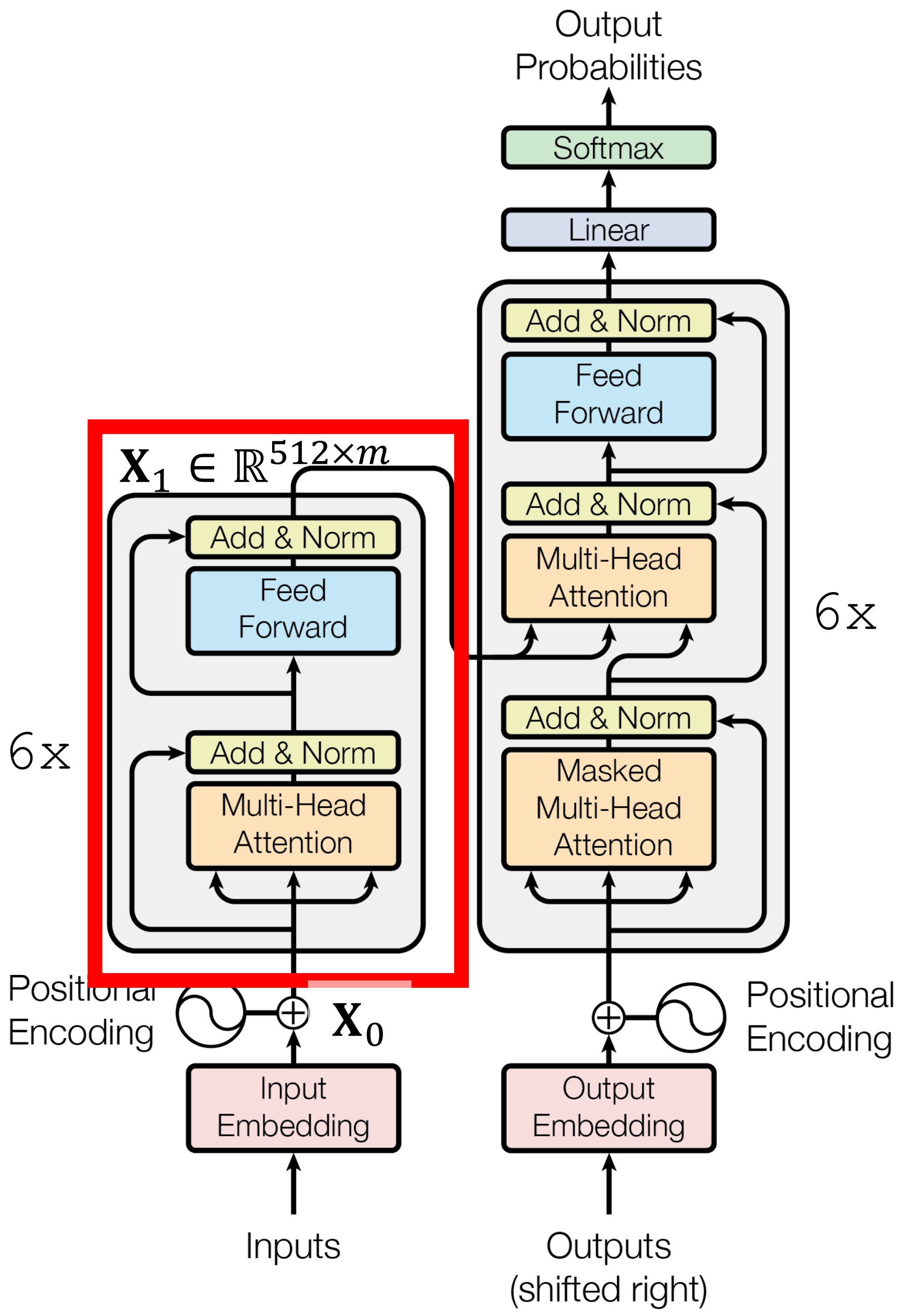


Encoder Network

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

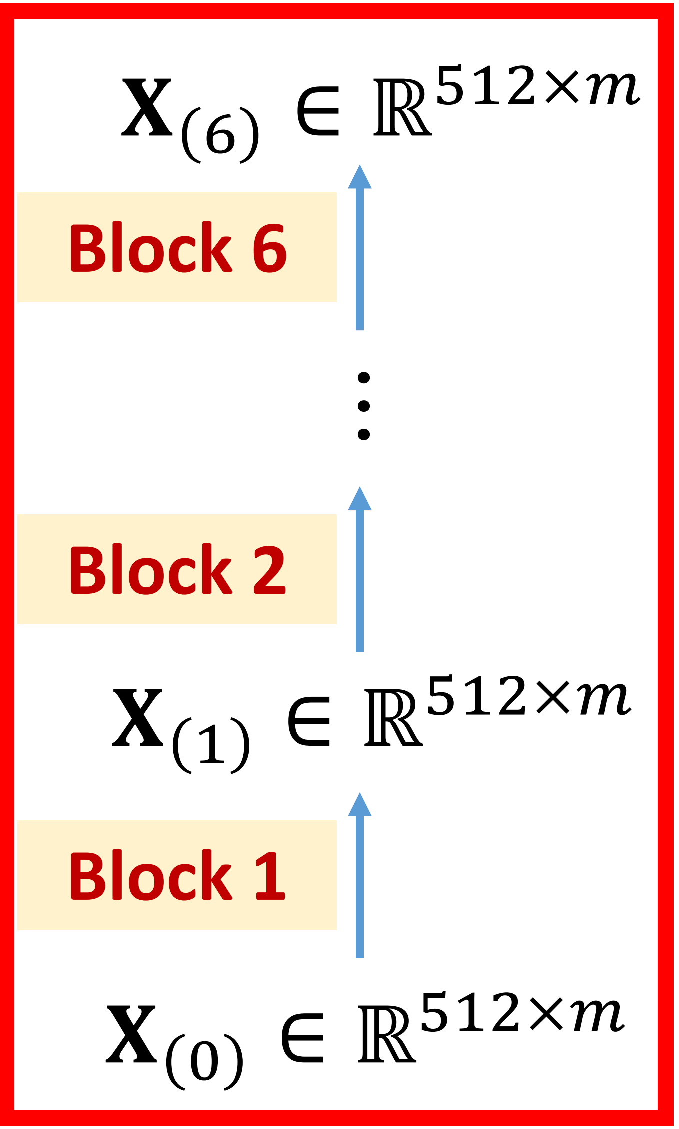
Block 1

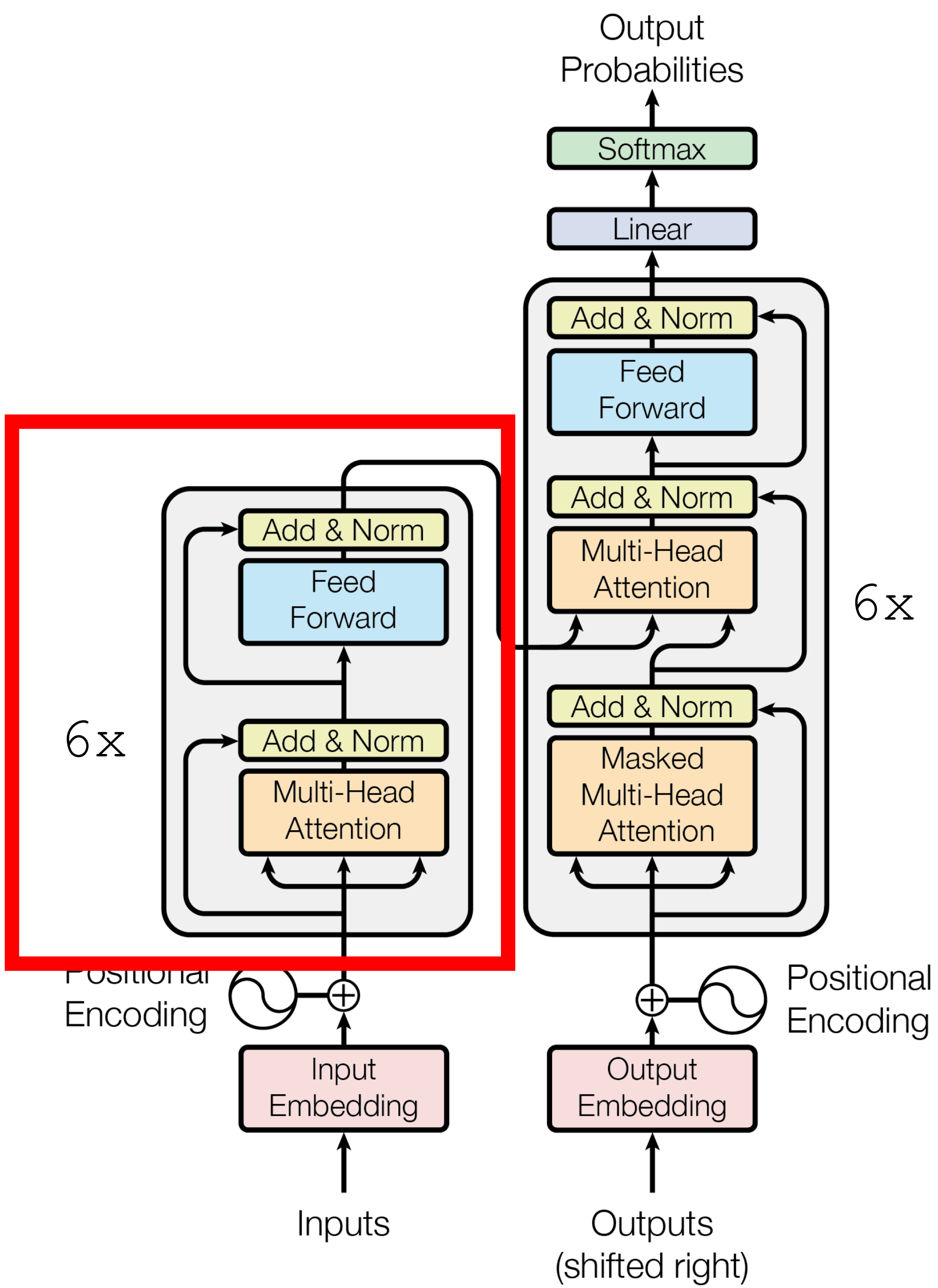
$$\mathbf{X}_{(0)} \in \mathbb{R}^{512\times m}$$



Encoder Network

Encoder

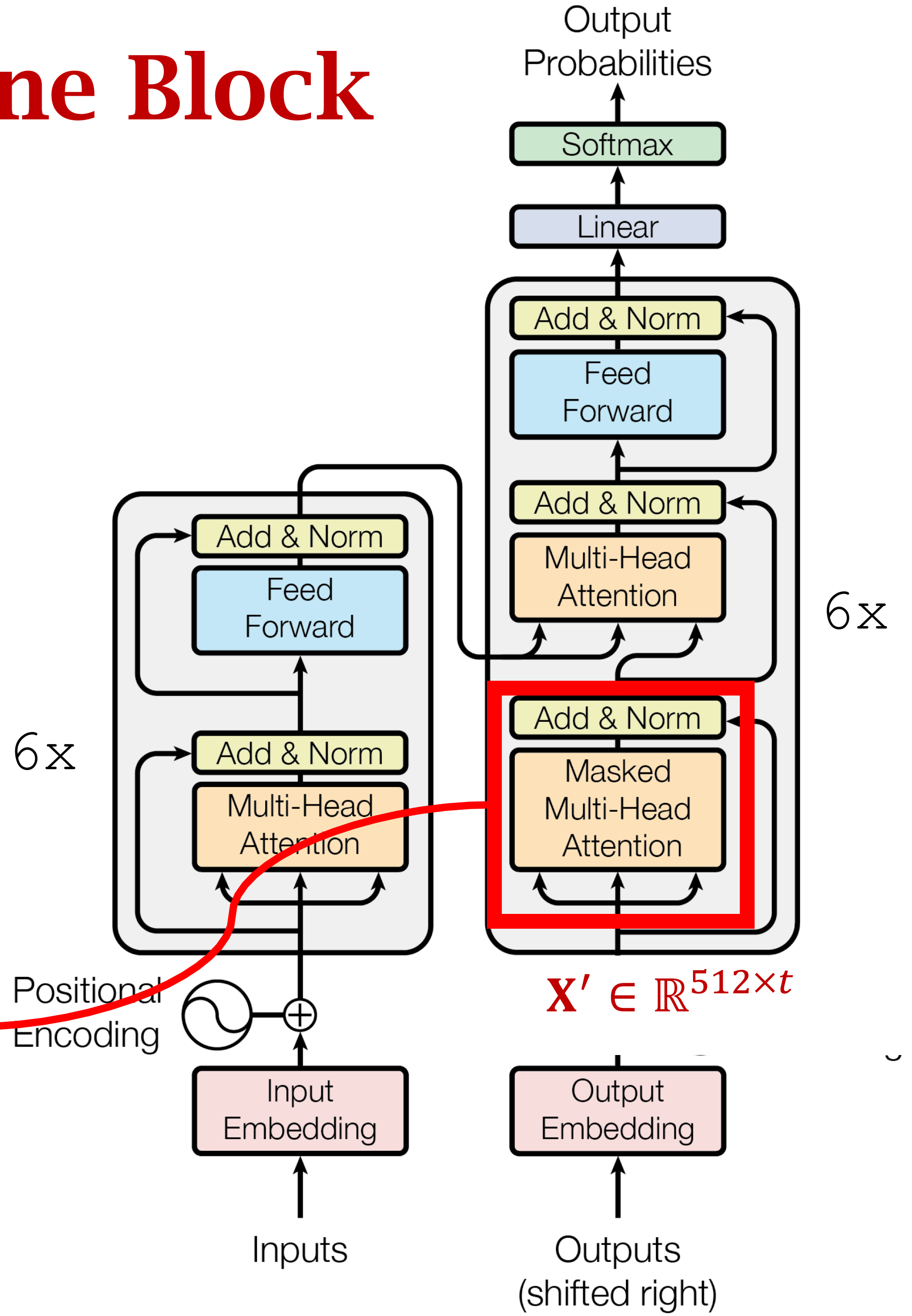




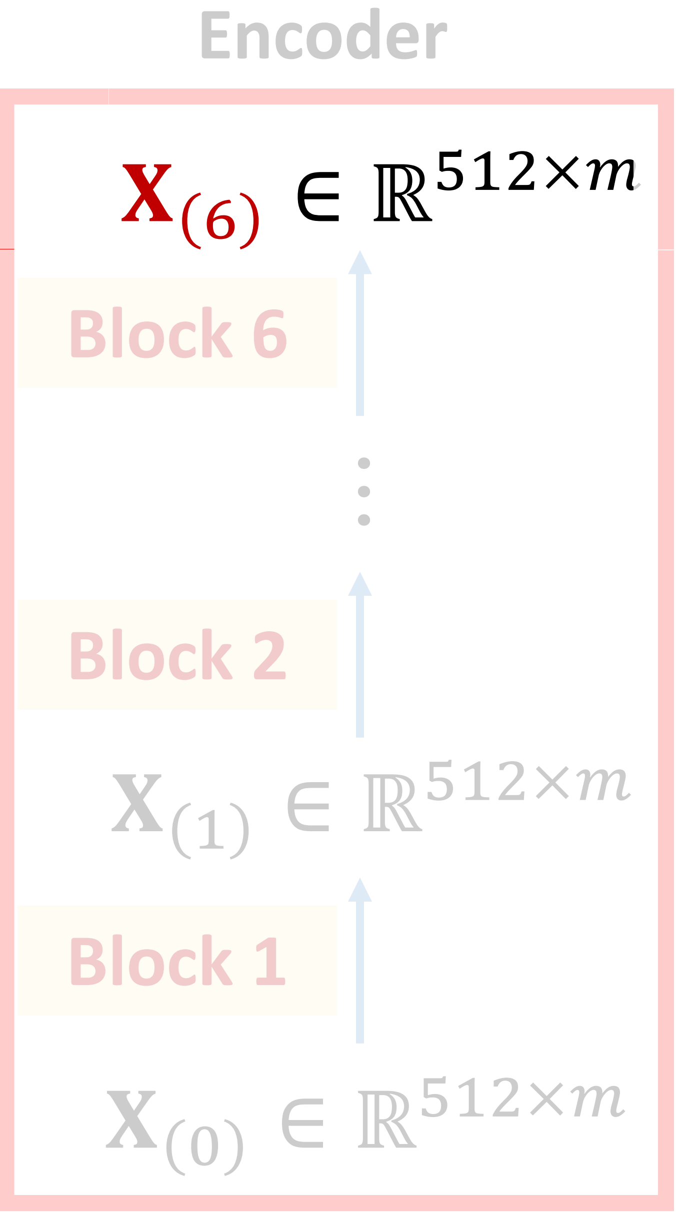
Decoder of Transformer

Decoder Network: One Block

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

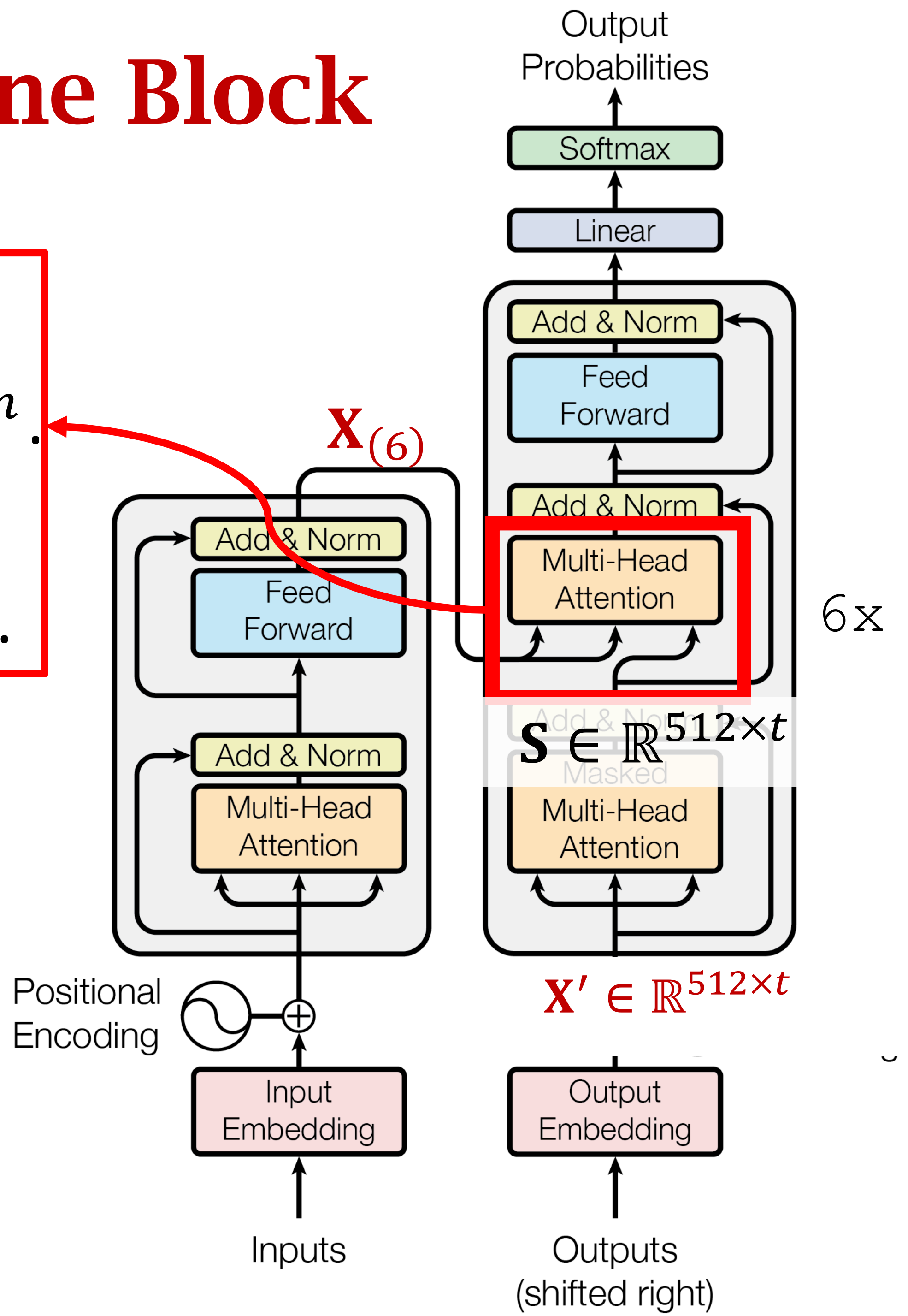


Decoder Network: One Block

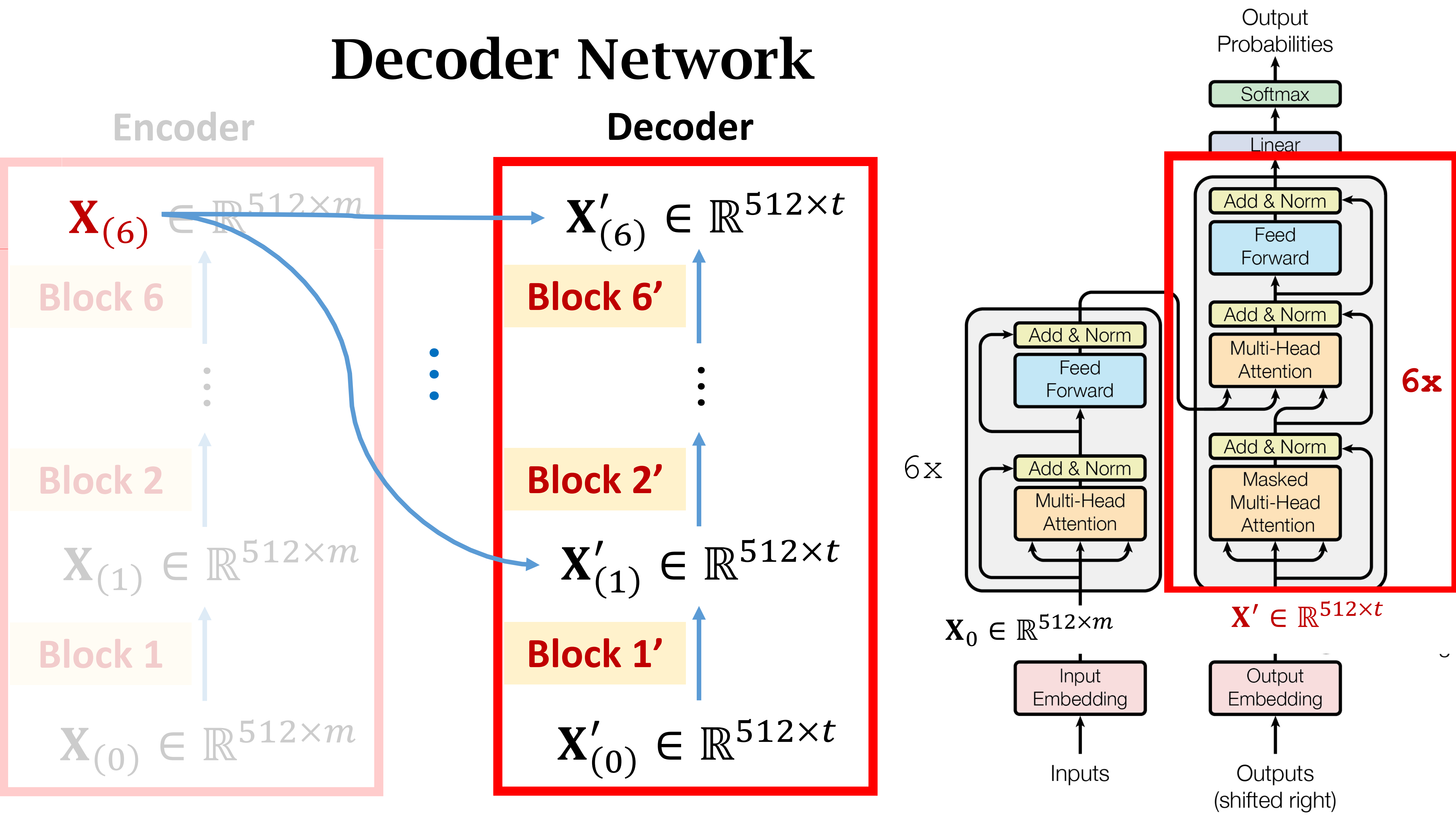


- Set $\mathbf{Q} = \mathbf{S} \in \mathbb{R}^{512\times t}$.
- $\mathbf{K} = \mathbf{V} = \mathbf{X}_{(6)} \in \mathbb{R}^{512\times m}$.
- Multi-head attention outputs a $512\times t$ matrix.

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

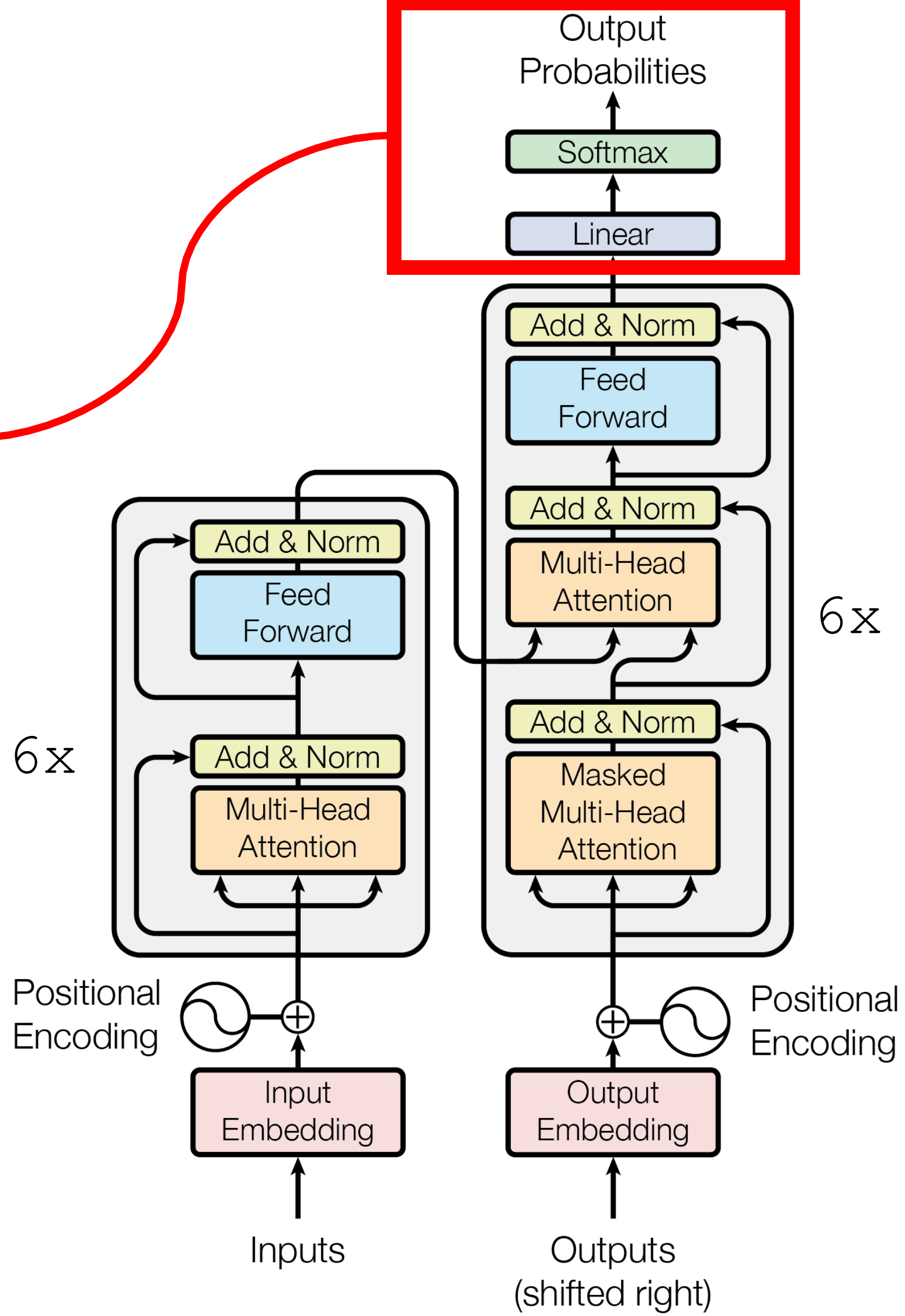


Decoder Network



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:

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

Key Concept: Multi-Head Attention

- Inputs: query $\mathbf{Q}$, key $\mathbf{K}$, and value $\mathbf{V}$.
- Linear maps: $\widetilde{\mathbf{Q}} = \mathbf{W}_Q \mathbf{Q}$, $\widetilde{\mathbf{K}} = \mathbf{W}_K \mathbf{K}$, and $\widetilde{\mathbf{V}} = \mathbf{W}_V \mathbf{V}$.
- Single-head attention:

$$\mathbf{C} = \mathrm{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \widetilde{\mathbf{V}} \cdot \mathrm{softmax}\big(\widetilde{\mathbf{K}}^T \widetilde{\mathbf{Q}}\big).$$

- Multi-head attention:
  - Repeat $\mathrm{attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ using different parameters $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$.
  - Get $\mathbf{C}^{[1]}, \mathbf{C}^{[2]}, \cdots, \mathbf{C}^{[l]} \in \mathbb{R}^{d_c \times t}$.
  - Concatenate the m matrices to get $\tilde{\mathbf{C}} \in \mathbb{R}^{l d_c \times t}$.

Attention in the encoder:

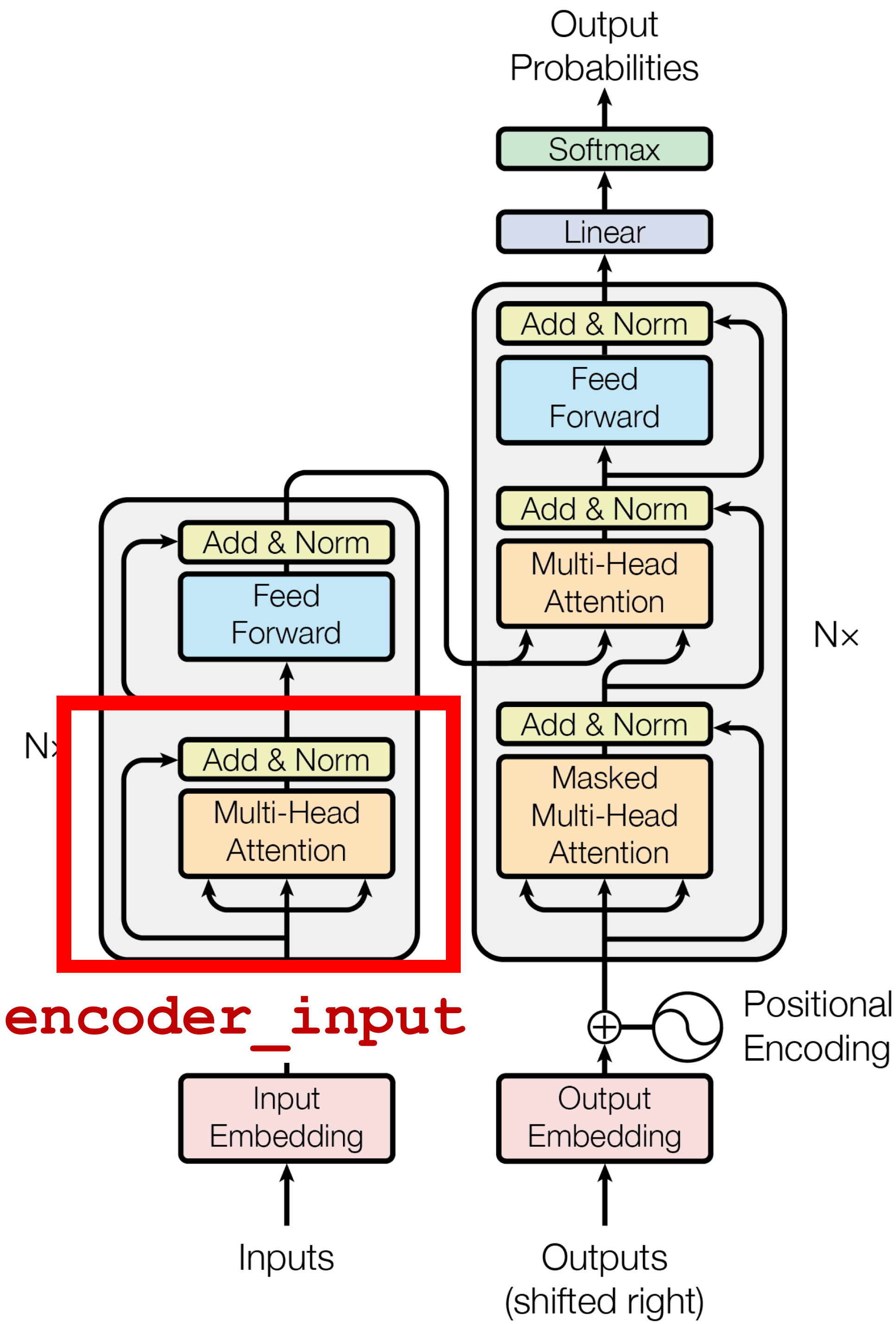
- $Q = K = V =$ encoder_input.

1st attention in the decoder:

- $Q = K = V =$ decoder_input.

2nd attention in the decoder

- $Q =$ decoder_input
- $K = V =$ encoder_output.



Attention in the encoder:

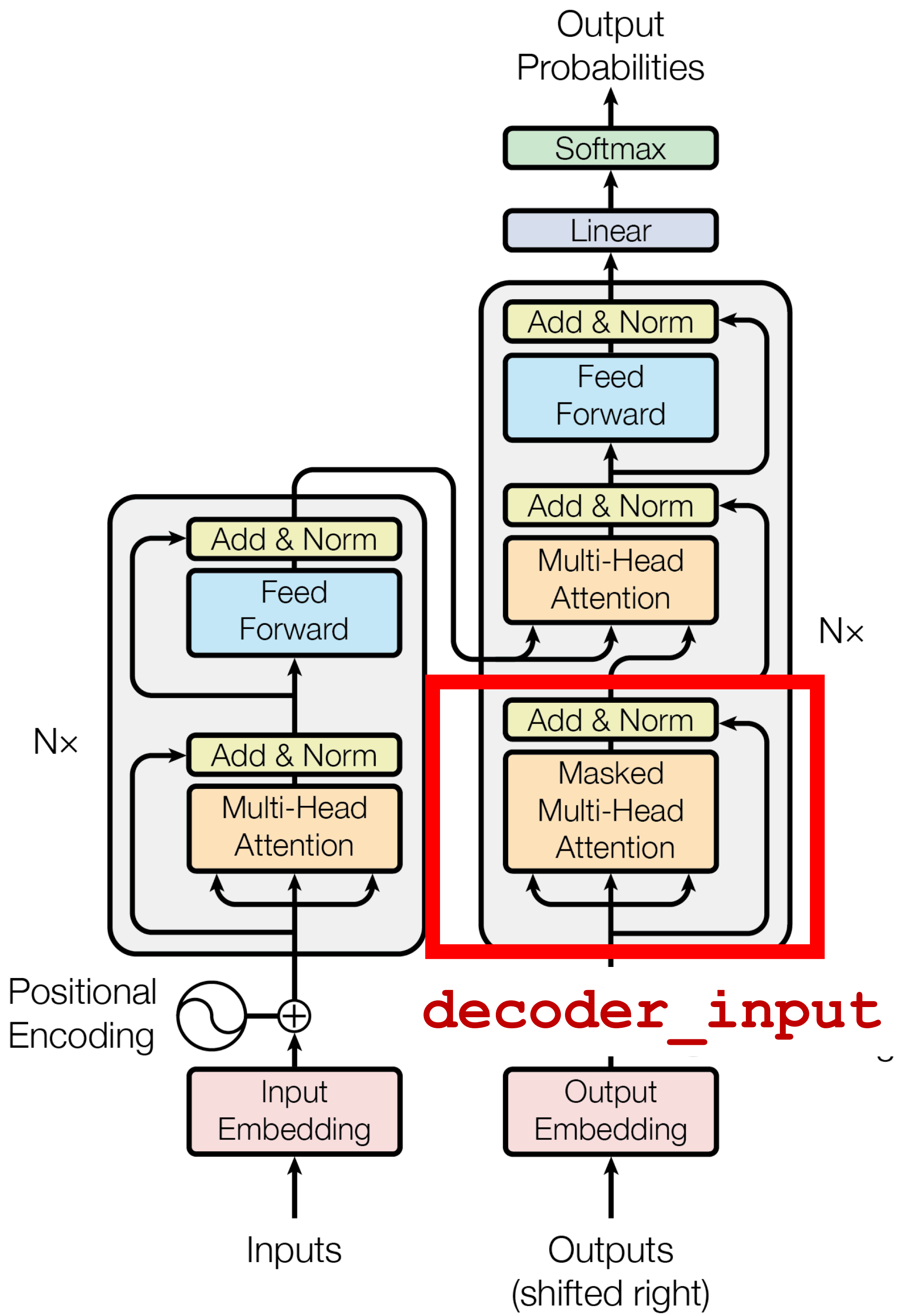
- $\mathbf{Q} = \mathbf{K} = \mathbf{V}$ = encoder_input.

1st attention in the decoder:

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

2nd attention in the decoder

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



Attention in the encoder:

- $Q = K = V =$ encoder_input.

1^{st} attention in the decoder:

- $Q = K = V =$ decoder_input.

2^{nd} attention in the decoder

- $Q =$ decoder_input
- $K = V =$ encoder_output.