# Transformers I

CS 4391 Introduction Computer Vision

Professor Yu Xiang

The University of Texas at Dallas

# Machine Translation

- Translate a phrase from one language to anther
  - E.g., English phrase to French phrase

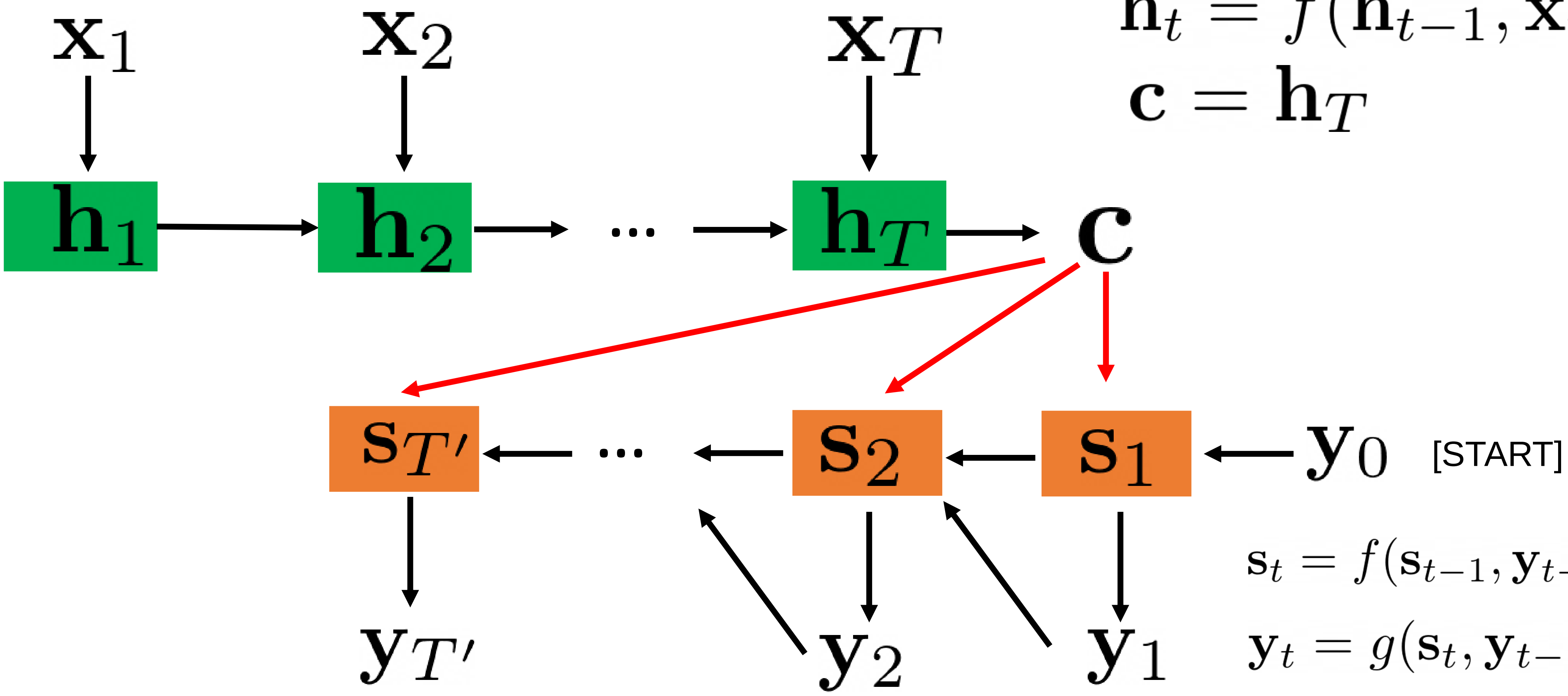
Google Translation


13 words

15 words

# RNN Encoder-Decoder

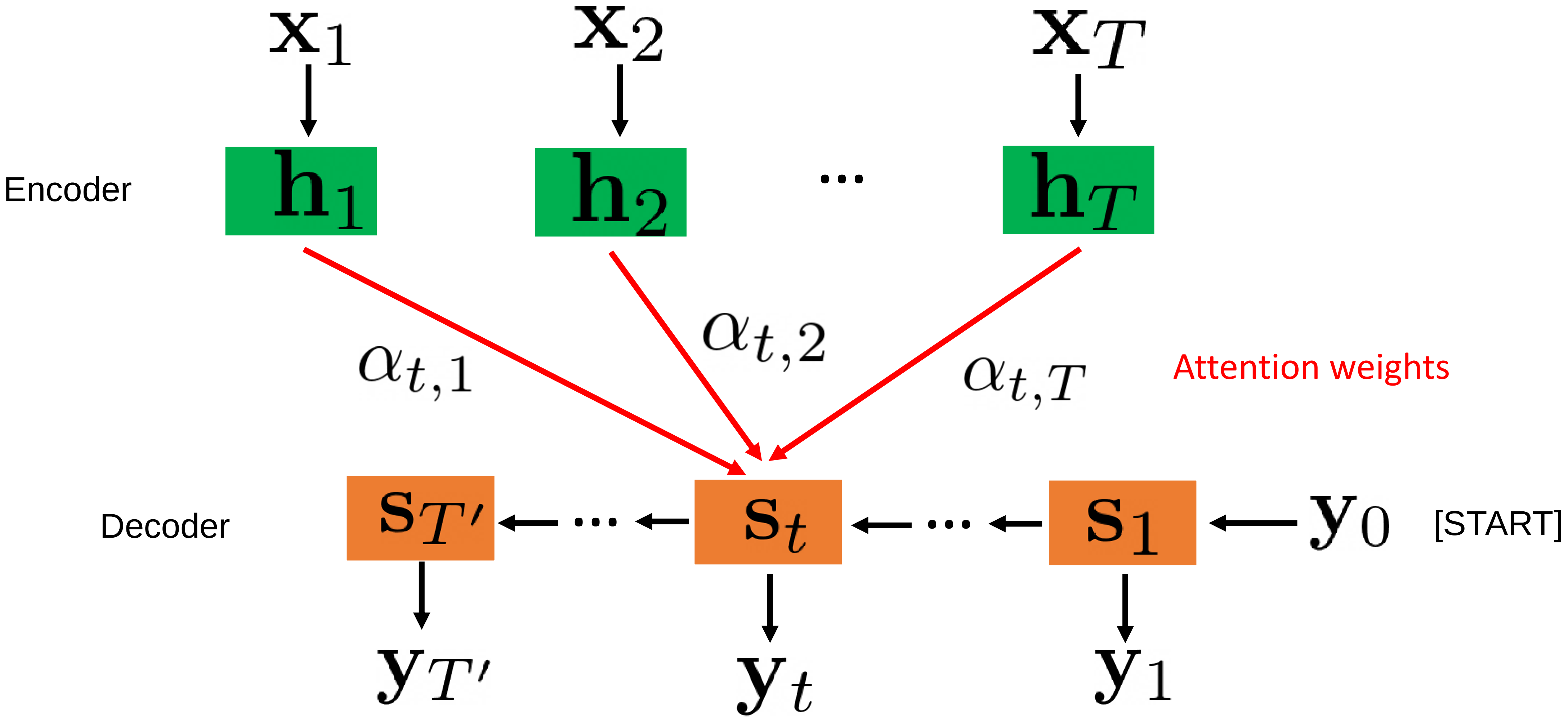
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$\mathbf{c} = \mathbf{h}_T$$




$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c})$$
$$\mathbf{y}_t = g(\mathbf{s}_t, \mathbf{y}_{t-1}, \mathbf{c})$$

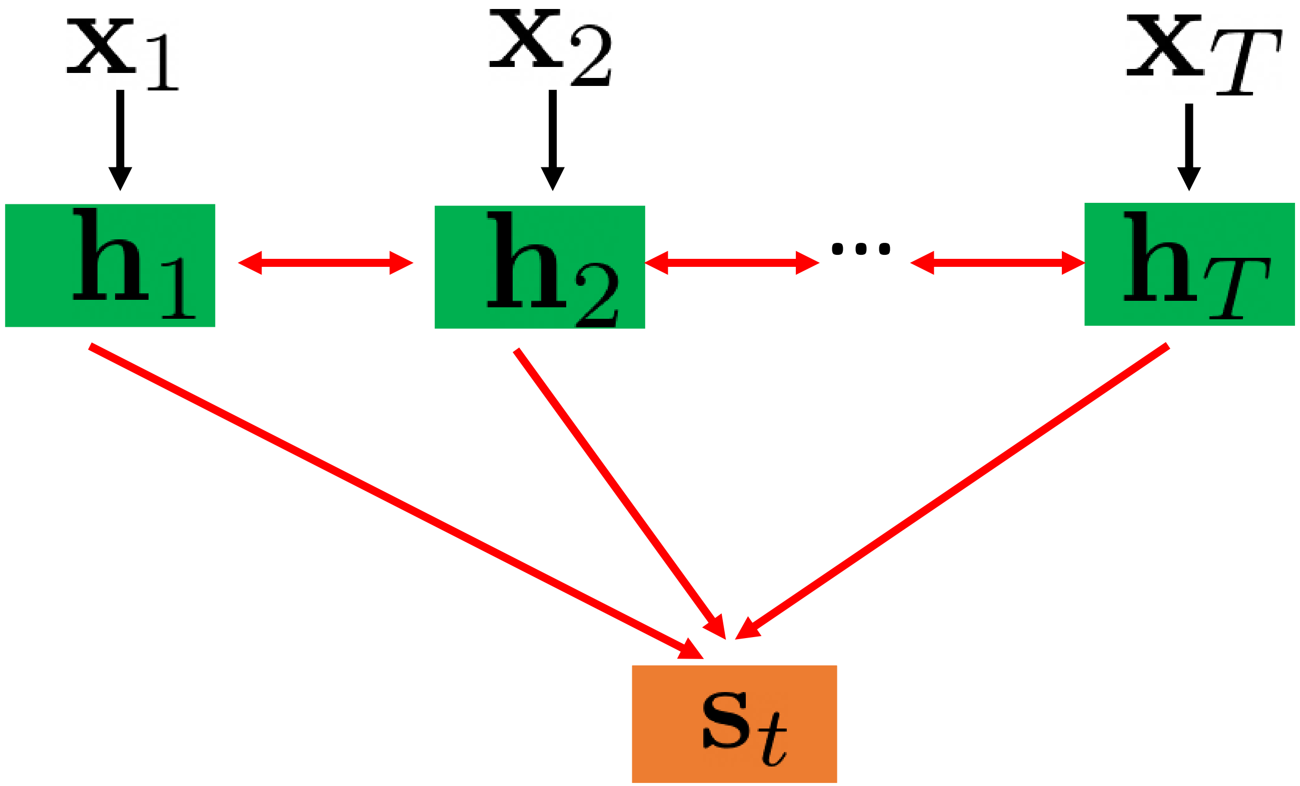
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Cho et al., EMNLP’14

# Transformer: Encoder-Decoder with Attention



# Transformer: Attention

- Two types of attentions
- Self-attention
- Cross-attention



# Transformer: Attention

- Input
  - (key, value) pairs (think about python dictionary)
  - A query

- Output
  - Compare the query to all the keys to compute weights
  - Weighted sum of the values

Attention is all you need. Vaswani et al., NeurIPS'17

# Transformer: Attention

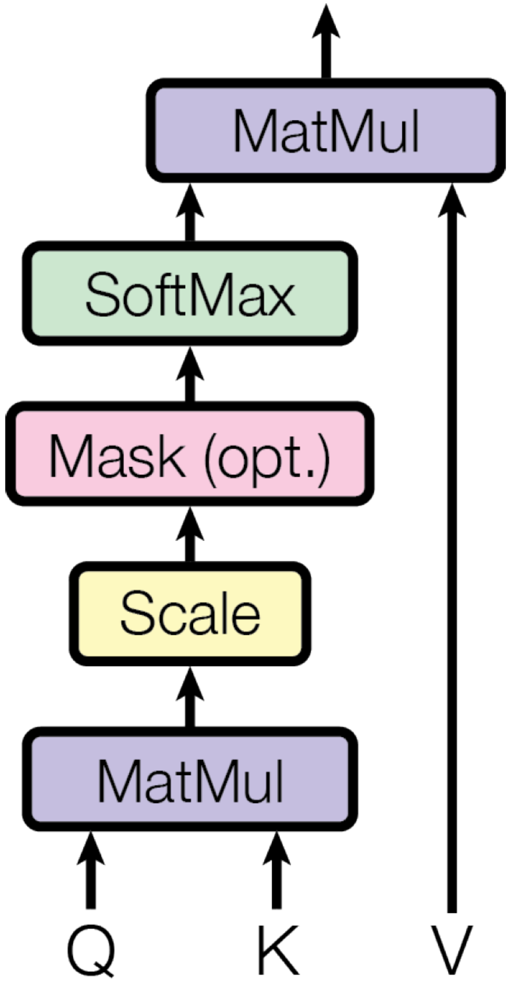
- Scaled Dot-Product Attention
  - Keys $K : m \times d_k$
  - Values $V : m \times d_v$
  - n queries $Q : n \times d_k$

$$\text{Attention}(Q,K,V) = \underbrace{\text{softmax}(\frac{QK^T}{\sqrt{d_k}})}_{\text{weights}} V$$

$$n \times d_v$$

Softmax function

$$\sigma(\mathbf{y})_i = \frac{e^{y_i}}{\sum_j^m e^{y_i}}$$




Attention is all you need. Vaswani et al., NeurIPS'17

# Transformer: Attention

- Multi-Head Attention
  - Suppose the latent vector is with dimension $d_{\text{model}}$

$m \times d_{\text{model}}$ $\quad d_{\text{model}} \times d_k$

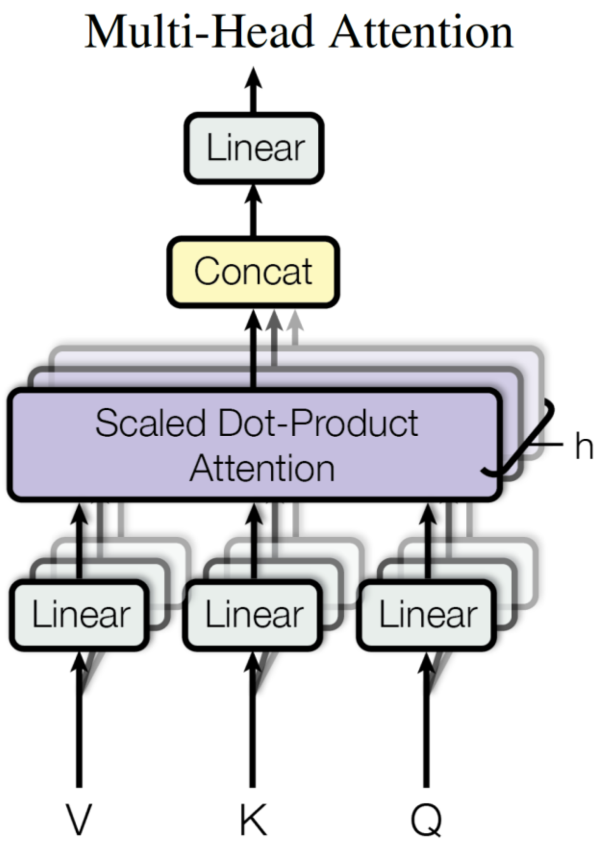
$$\mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$ Projection

$n \times d_v$

$n \times d_{\text{model}}$ $\quad d_{\text{model}} \times d_k$ $\quad m \times d_{\text{model}}$ $\quad d_{\text{model}} \times d_v$

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$

$n \times d_{\text{model}}$ $\quad n \times hd_v$ $\quad hd_v \times d_{\text{model}}$




Attention is all you need. Vaswani et al., NeurIPS'17

# Transformer: Encoder

- Self-attention (repeat N times)
  - Keys, values and queries are all the same
  - n input tokens $n \times d_{\text{model}}$

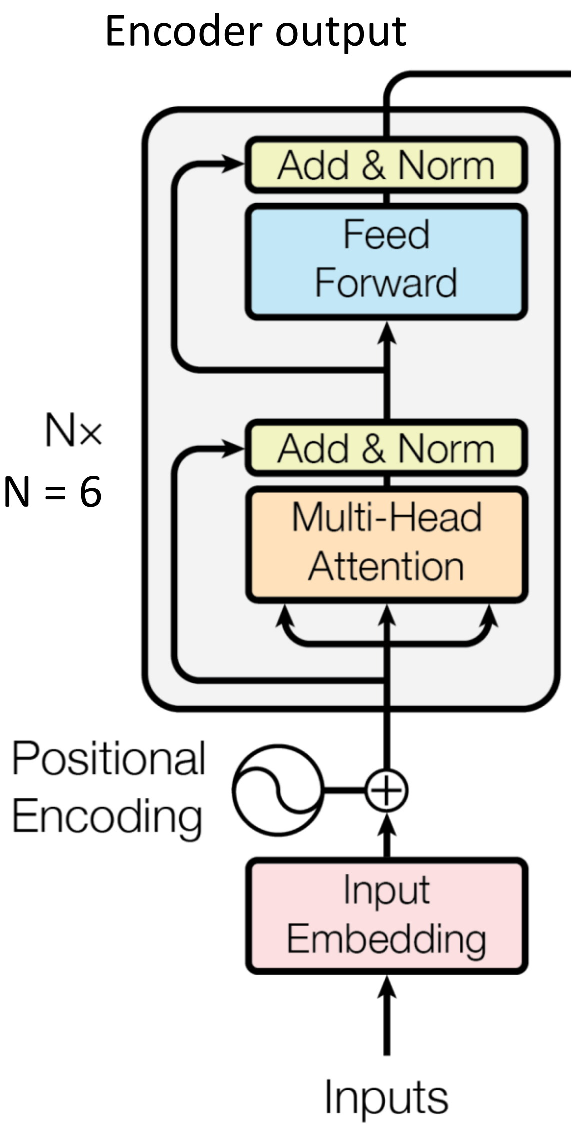
$$\text{MultiHead}(Q, K, V)$$

  - Residual connection

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

  - Layer normalization

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad \frac{a^l - \mu^l}{\sigma^l}$$




Attention is all you need. Vaswani et al., NeurIPS'17

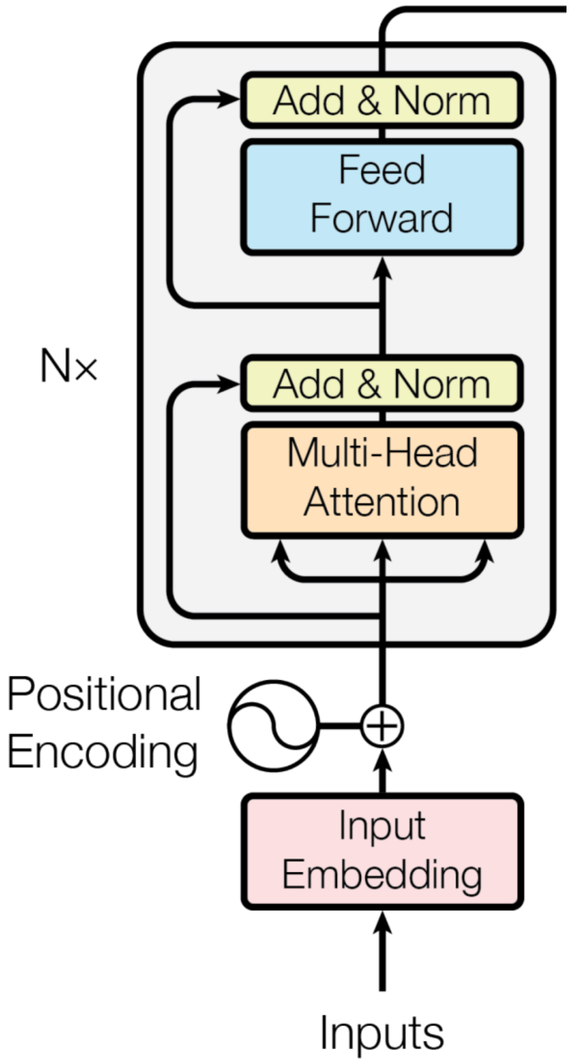
# Transformer: Encoder

- Feed Forward Network

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- Positional encoding
  - Make use the order of the sequence
  - With dimension $d_{\text{model}}$ for each input

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$




Attention is all you need. Vaswani et al., NeurIPS’17

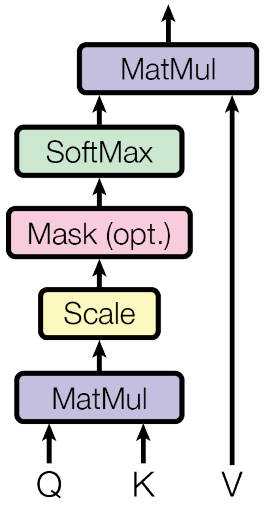
# Transformer: Decoder

- Output embedding

[START]

$$\mathbf{y}_0\ \mathbf{y}_1 \cdots \mathbf{y}_{t-1} \boxed{\mathbf{y}_t\ \ \mathbf{y}_{t+1} \cdots \mathbf{y}_{T'}}$$

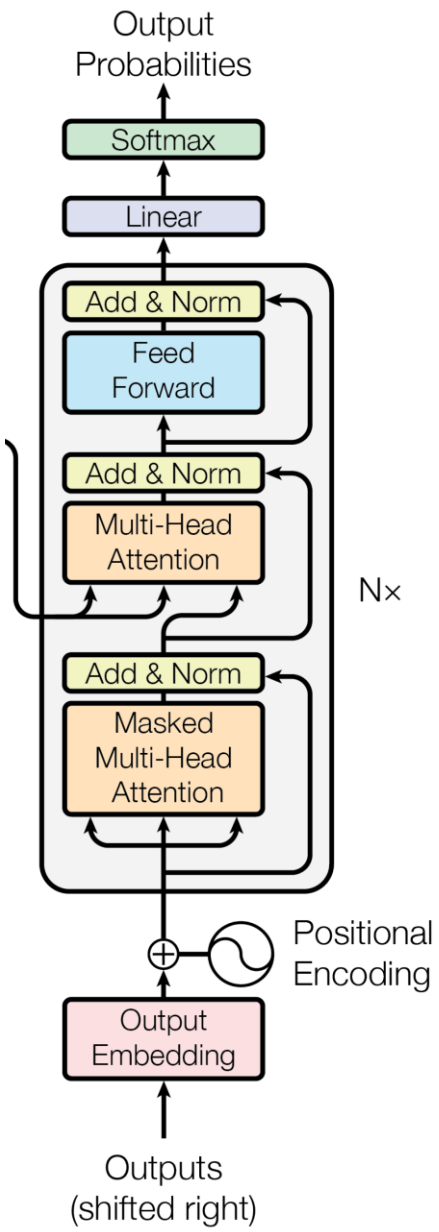
Shifted right by one position and insert the start token




Mask out current and future outputs during training (setting to $-\infty$ )

Encoder output

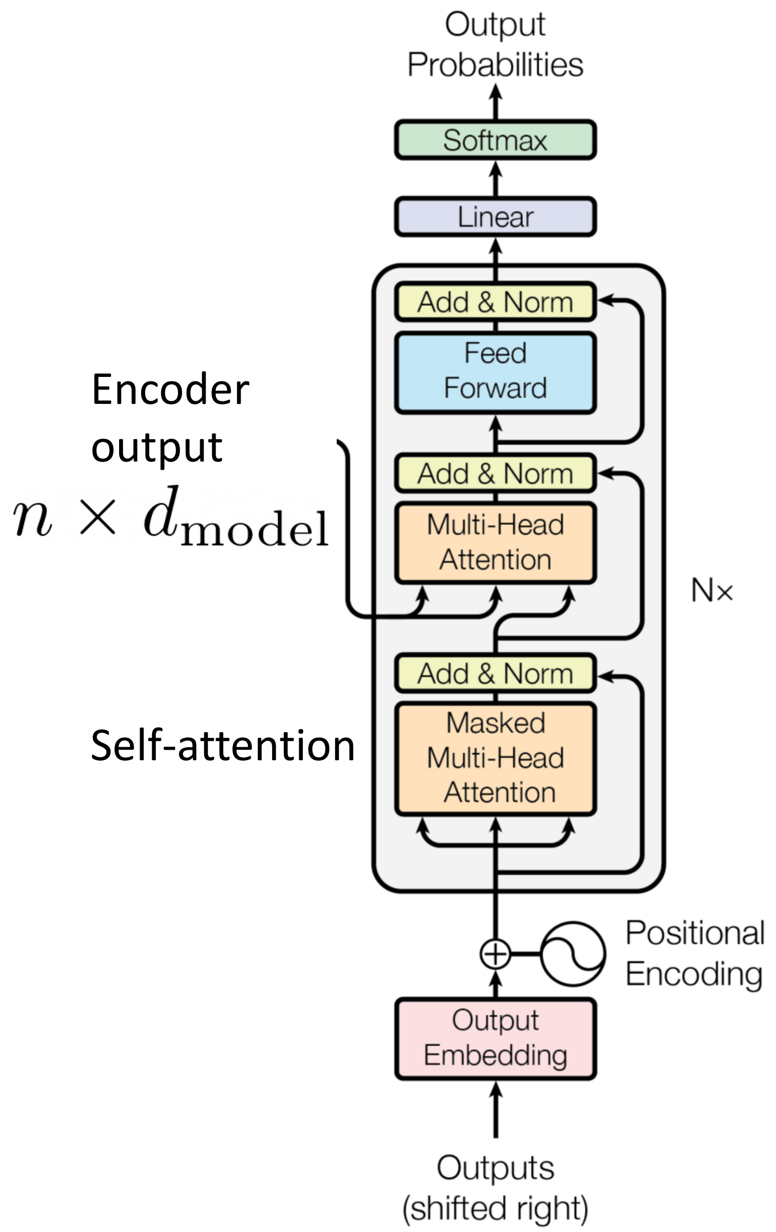
$n \times d_{\text{model}}$




Attention is all you need. Vaswani et al., NeurIPS'17

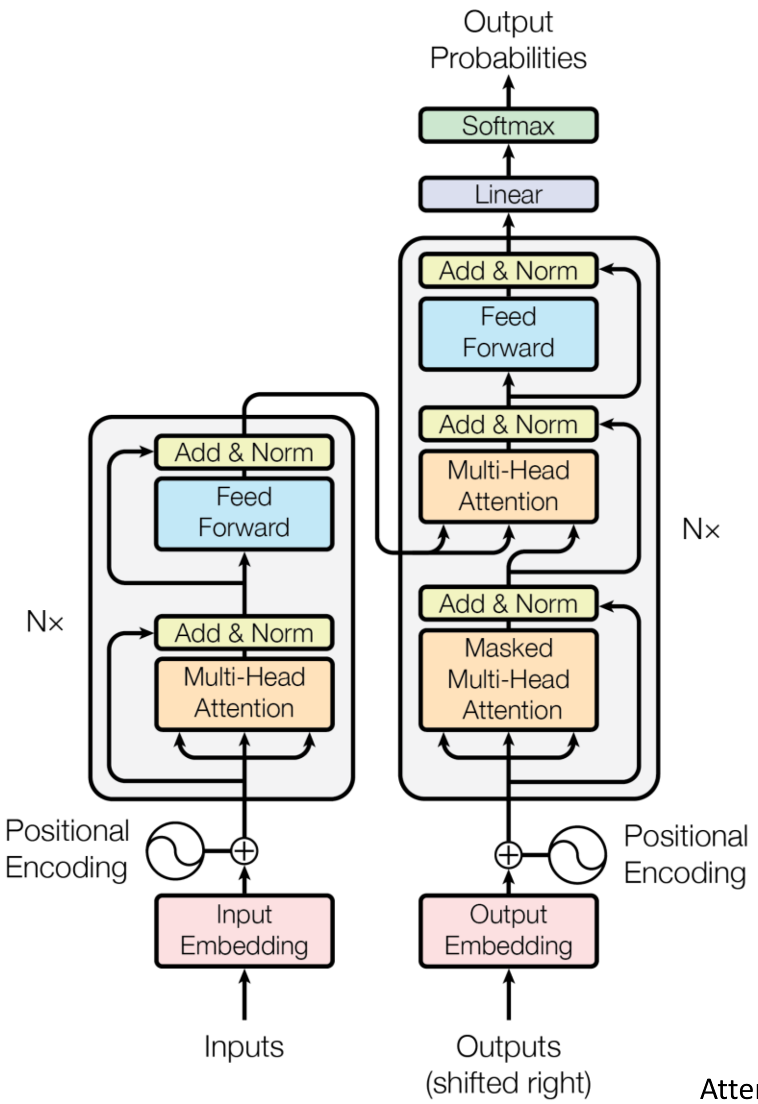
# Transformer: Decoder

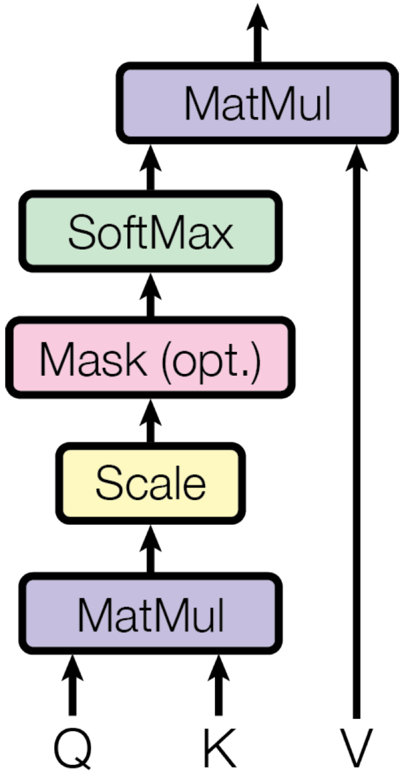
- Encoder-decoder attention
  - (Key, value): encoder output
  - Queries: decoder output
  - Every position in the decoder attends to all positions in the input sequence
- Softmax
  - Predicts next-token probabilities

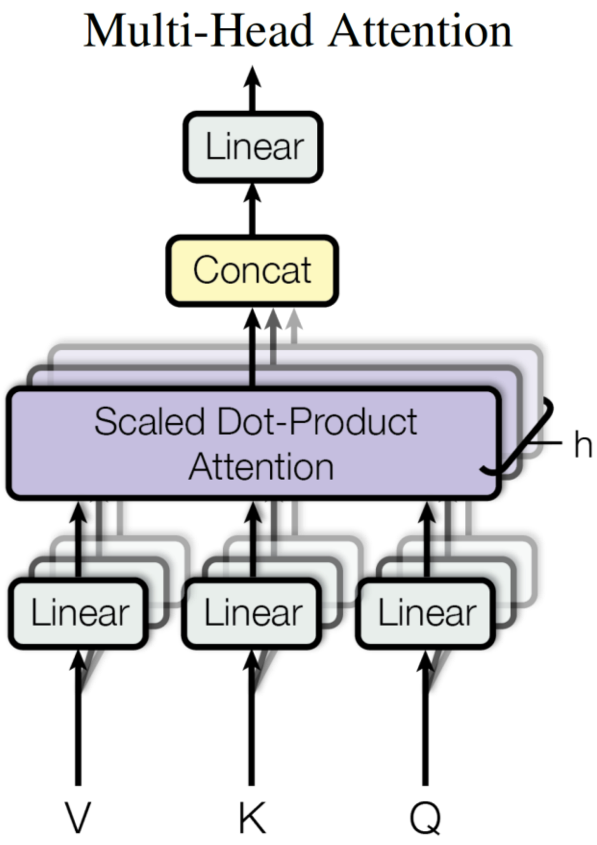



Attention is all you need. Vaswani et al., NeurIPS'17

# Transformer

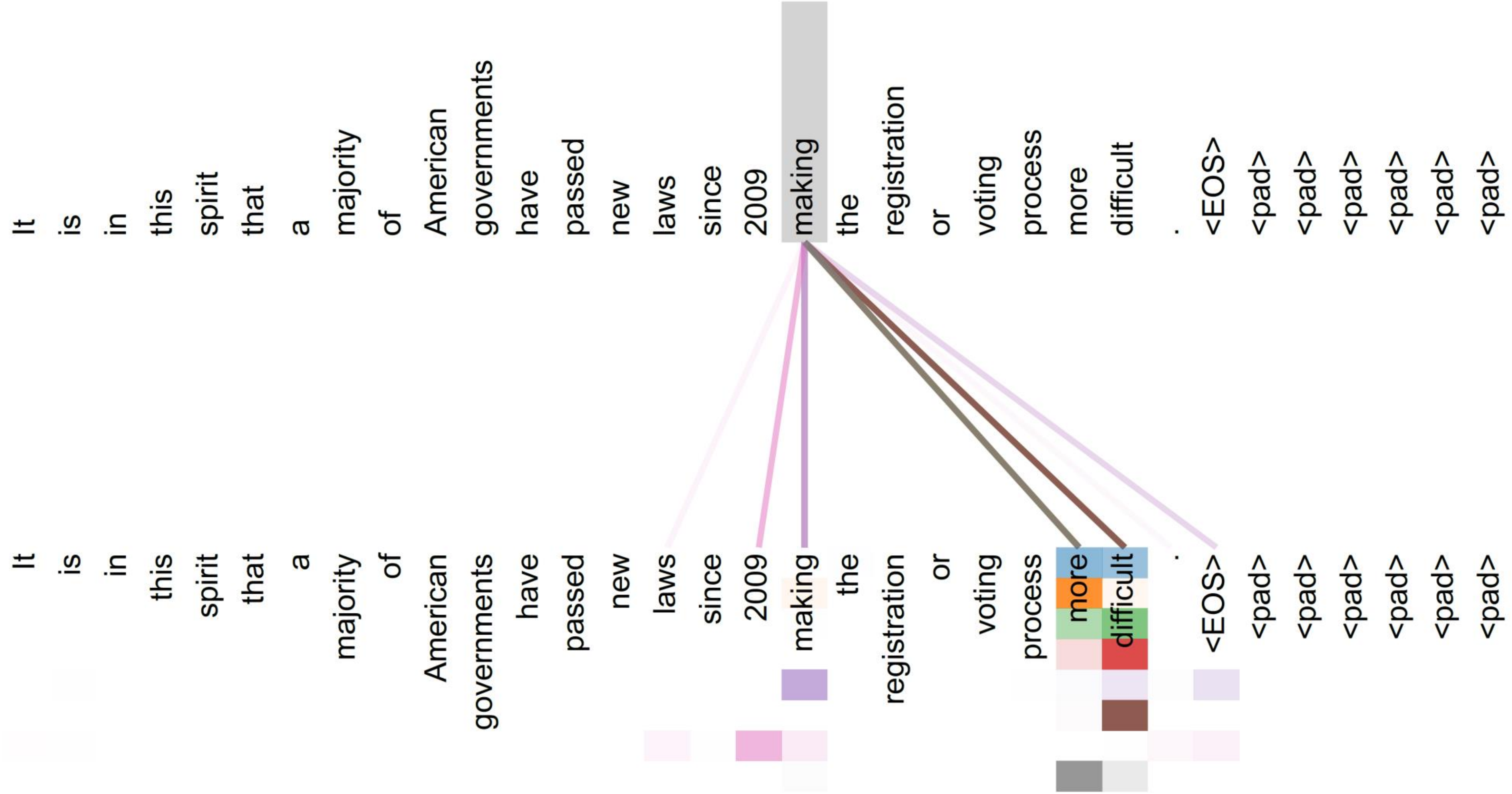



Attention is all you need. Vaswani et al., NeurIPS'17

# Transformer: Attention Visualization




Attention is all you need. Vaswani et al., NeurIPS’17

# Summary

- Transformers
  - Can capture long-distance dependencies (global attention)
  - Computationally efficient, more parallelizable

# Further Reading

- Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation https://arxiv.org/abs/1406.1078
- Neural Machine Translation by Jointly Learning to Align and Translate https://arxiv.org/abs/1409.0473
- Transformer: Attention is all you need https://arxiv.org/abs/1706.03762