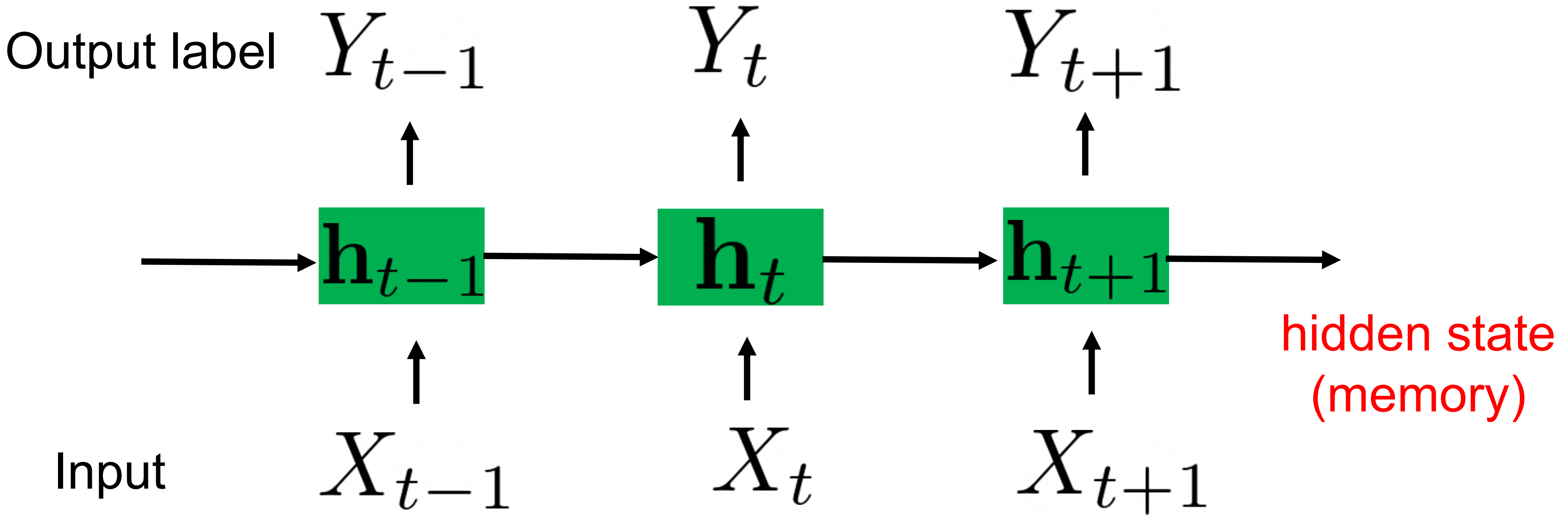
THE UNIVERSITY OF TEXAS AT DALLAS

Transformers

CS 4391 Introduction to Computer Vision
Professor Yapeng Tian
Department of Computer Science

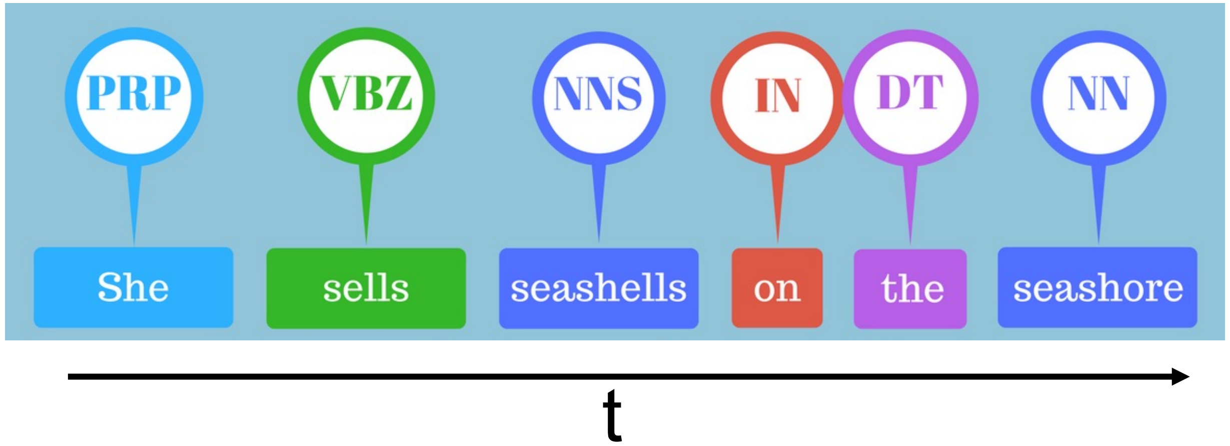
Slides borrowed from Professor Yu Xiang

Recurrent Neural Networks



Sequential Data Labeling

Part-of-speech tagging (grammatical tagging)



Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Machine Translation

Translate a phrase from one language to another

- E.g., English phrase to French phrase

Google Translation

English	French
UT Dallas is a rising public research university in the heart of DFW.	UT Dallas est une université de recherche publique en plein essor au cœur de DFW.
13 words	15 words

Machine Translation

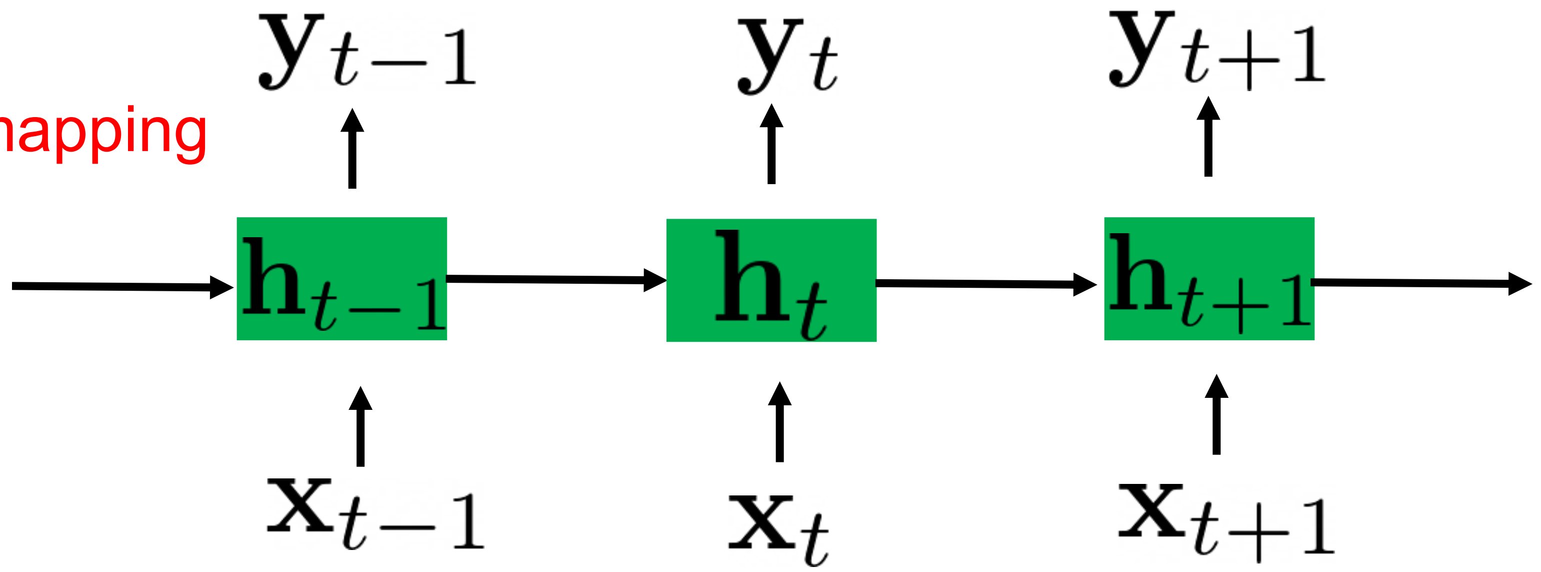
Input $$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$$

Output $$\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T'})$$

$$T \neq T'$$

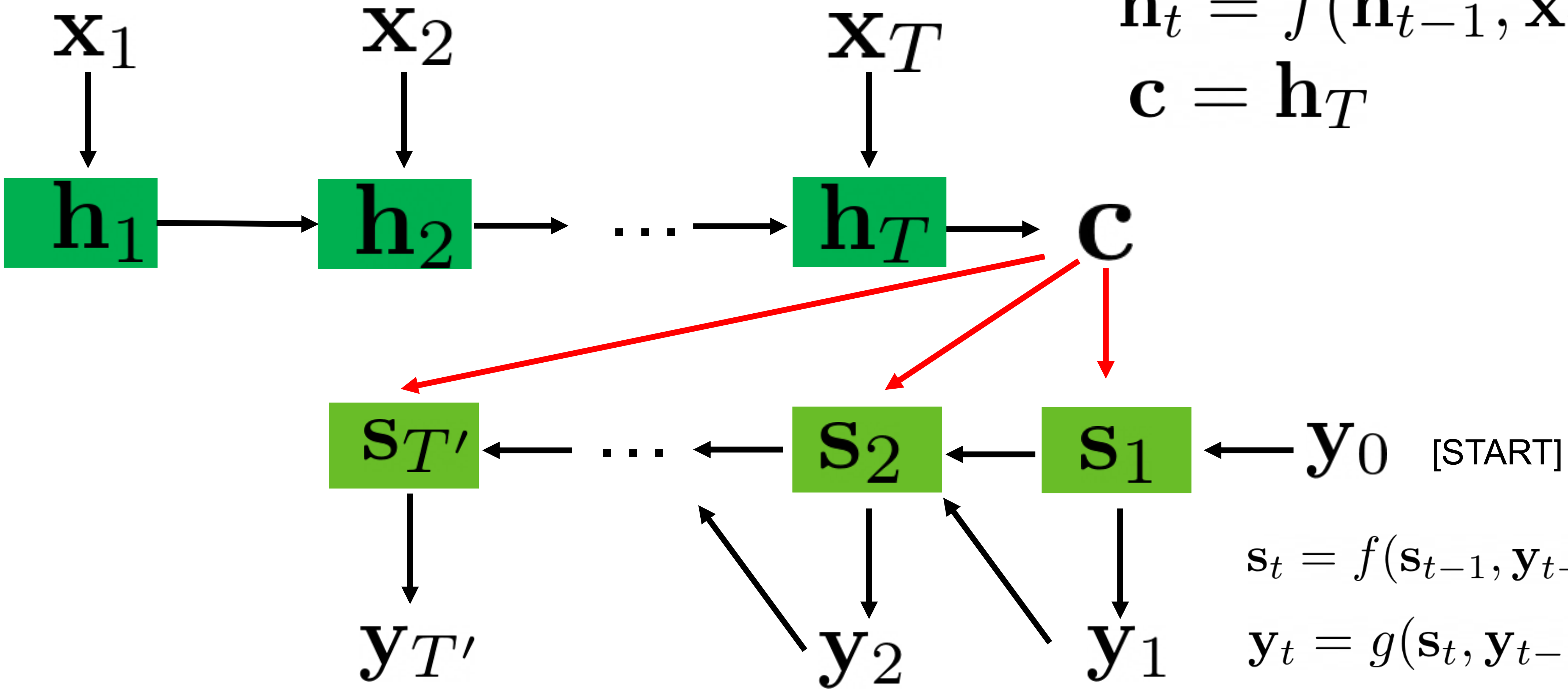
Not one to one mapping

RNN



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

RNN Encoder-Decoder

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

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

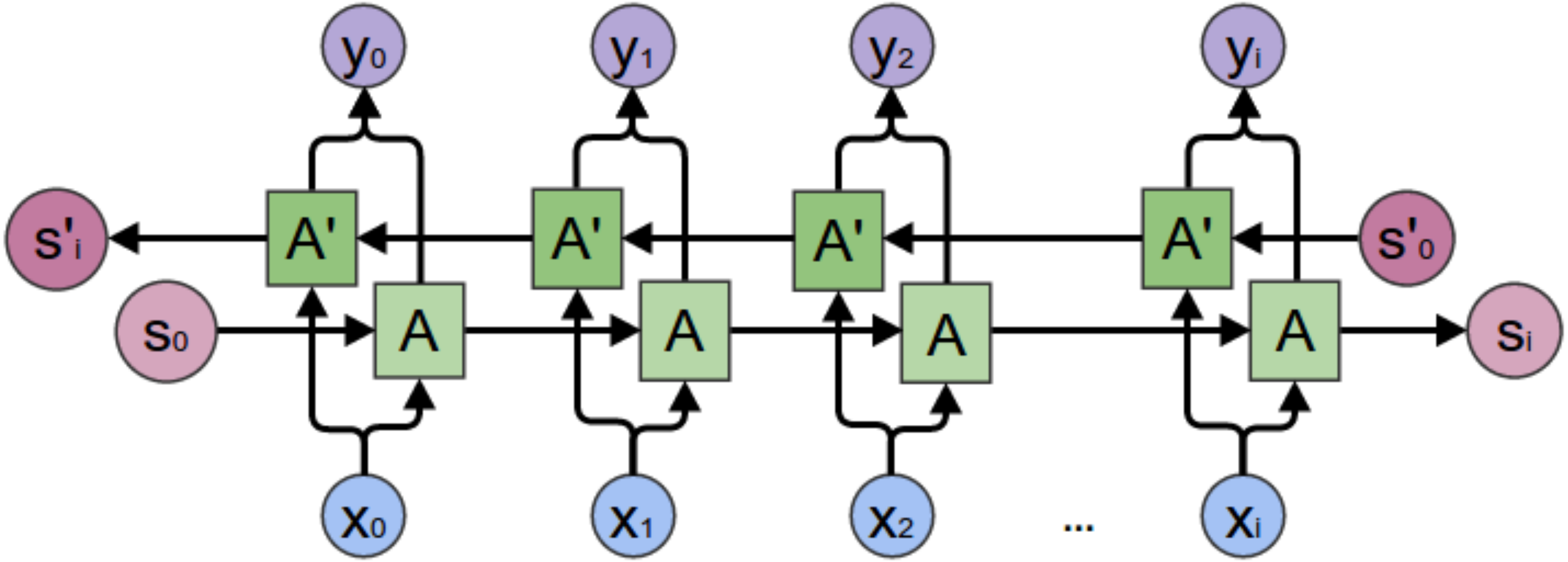
Pros

- Can deal with different input size and output size

Cons

- The fixed length embedding $\mathbf{c}$ cannot handle long sentence well (long-distance dependencies)

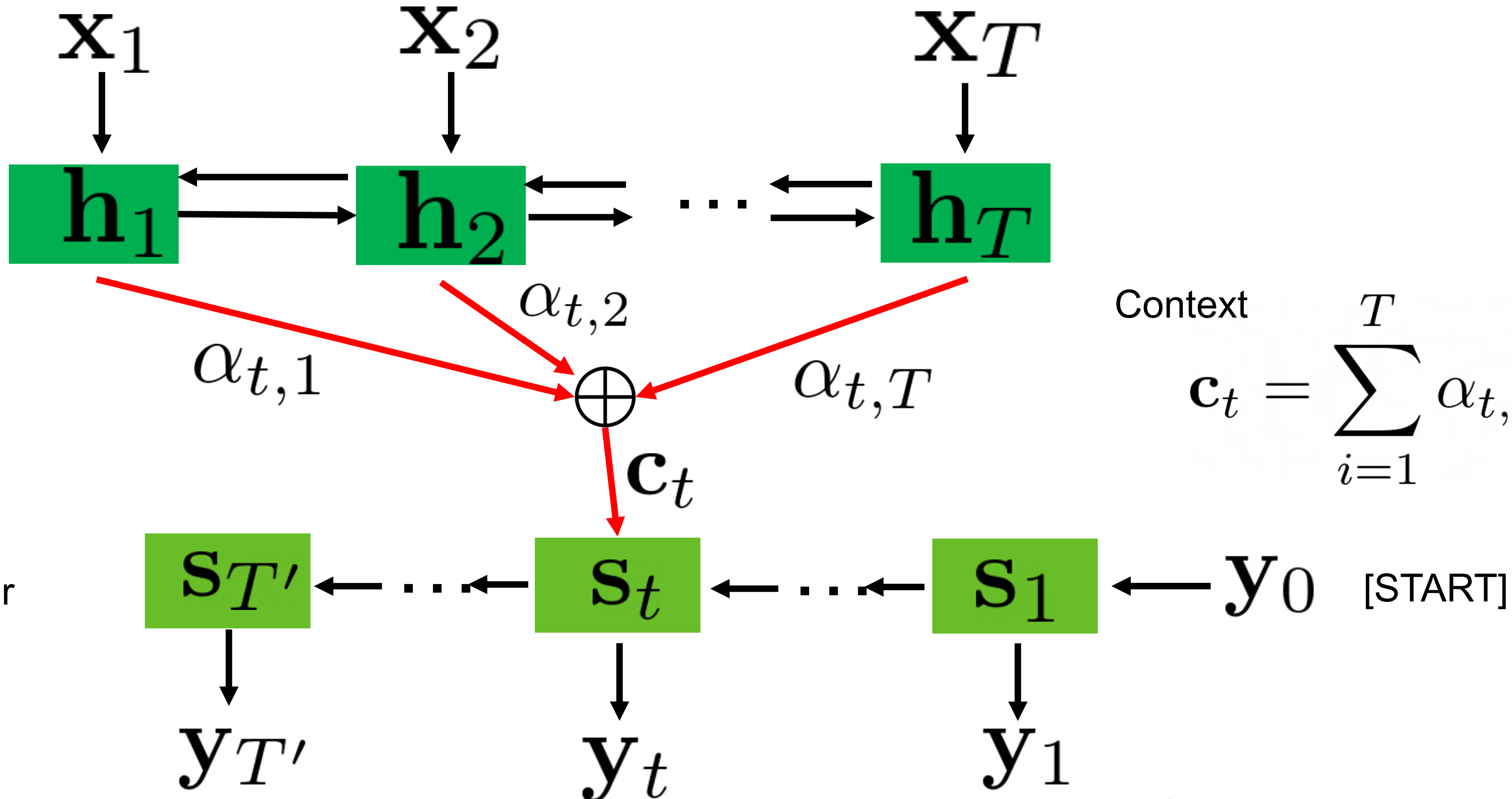
Bi-directional RNNs



https://blog.paperspace.com/bidirectional-rnn-keras/

RNN Encoder-Decoder with Attentions

Encoder: Bi-directional RNN



Decoder

Context

$$\mathbf{c}_t = \sum_{i=1}^{T} \alpha_{t,i} \mathbf{h}_i$$

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. Bahdanau et al., ICLR'15

RNN Encoder-Decoder with Attentions

Alignment model (attention)

$$e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$$

Feedforward network

Hidden state of output

Hidden state of input

Softmax $$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Attending to different parts of the input

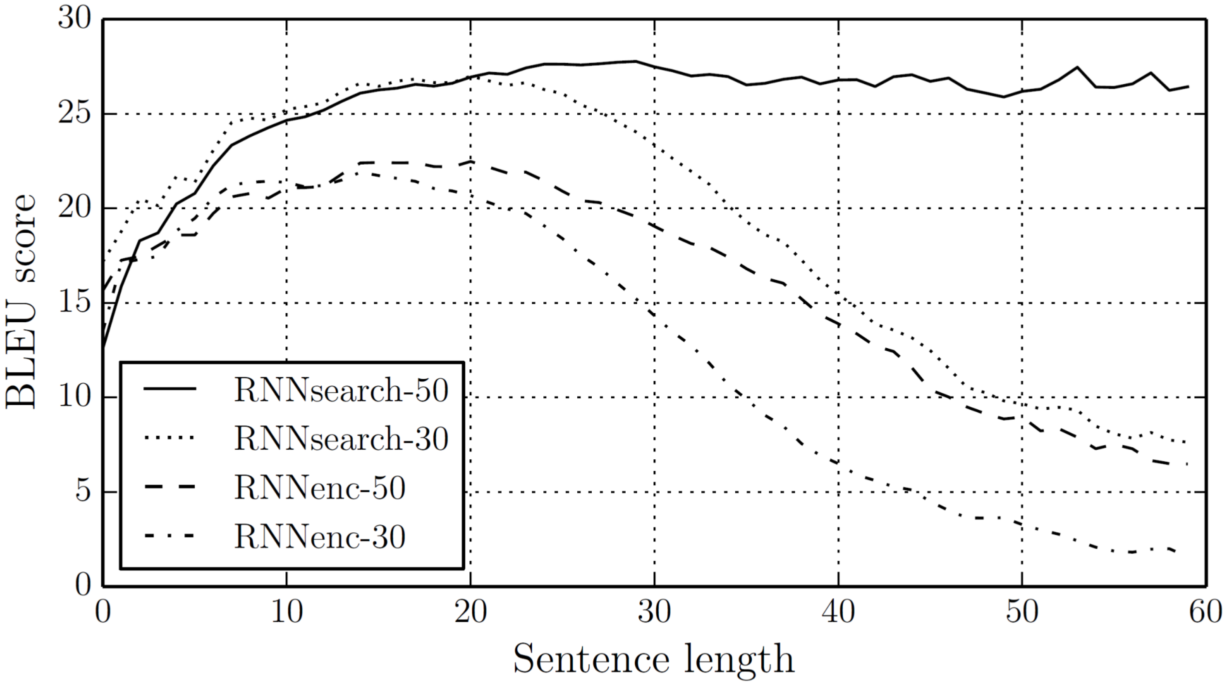
Context $$\mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j$$

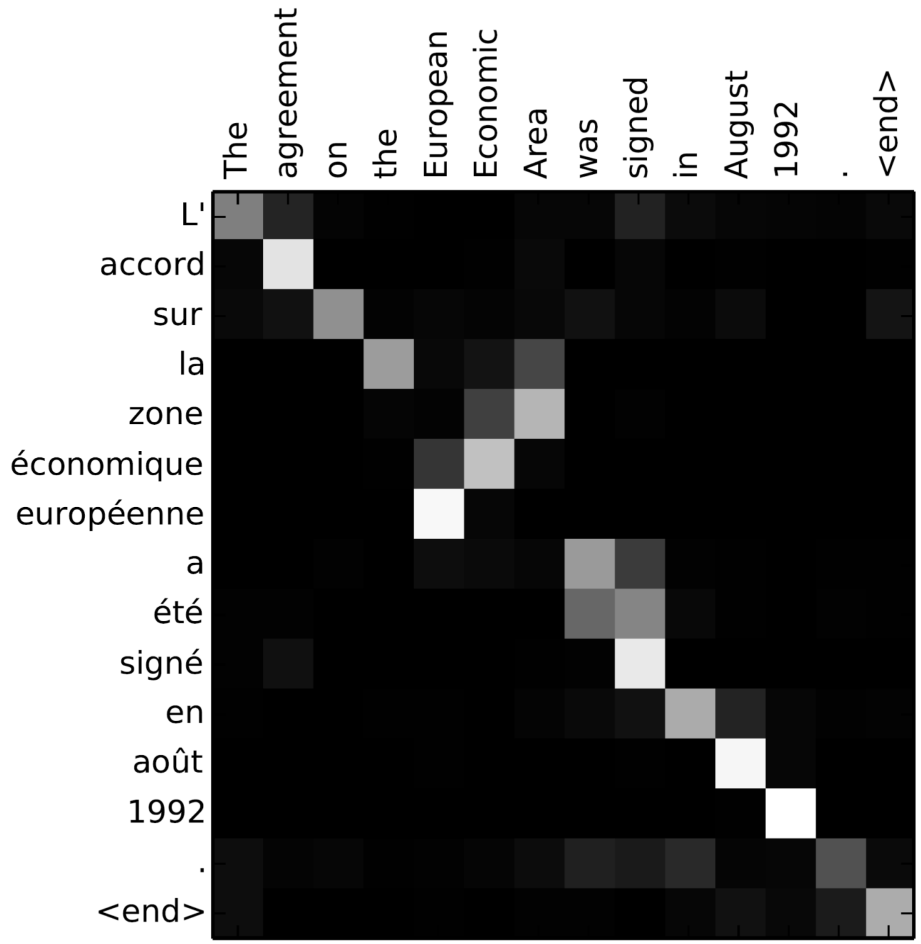
$$\mathbf{s}_i = f(\mathbf{s}_{i-1}, \mathbf{y}_{i-1}, \mathbf{c}_i)$$

Output $$\mathbf{y}_i = g(\mathbf{s}_i, \mathbf{y}_{i-1}, \mathbf{c}_i)$$

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. Bahdanau et al., ICLR'15

RNN Encoder-Decoder with Attentions





α_{ij}

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE. Bahdanau et al., ICLR'15

Limitations of RNNs

The sequential computation of hidden states precludes parallelization within training examples

$$\rightarrow \mathbf{h}_{t-1} \rightarrow \mathbf{h}_t \rightarrow \mathbf{h}_{t+1} \rightarrow$$

Cannot handle long sequences well

- Truncated back-propagation due to memory limits
- Difficult to capture dependencies in long distances

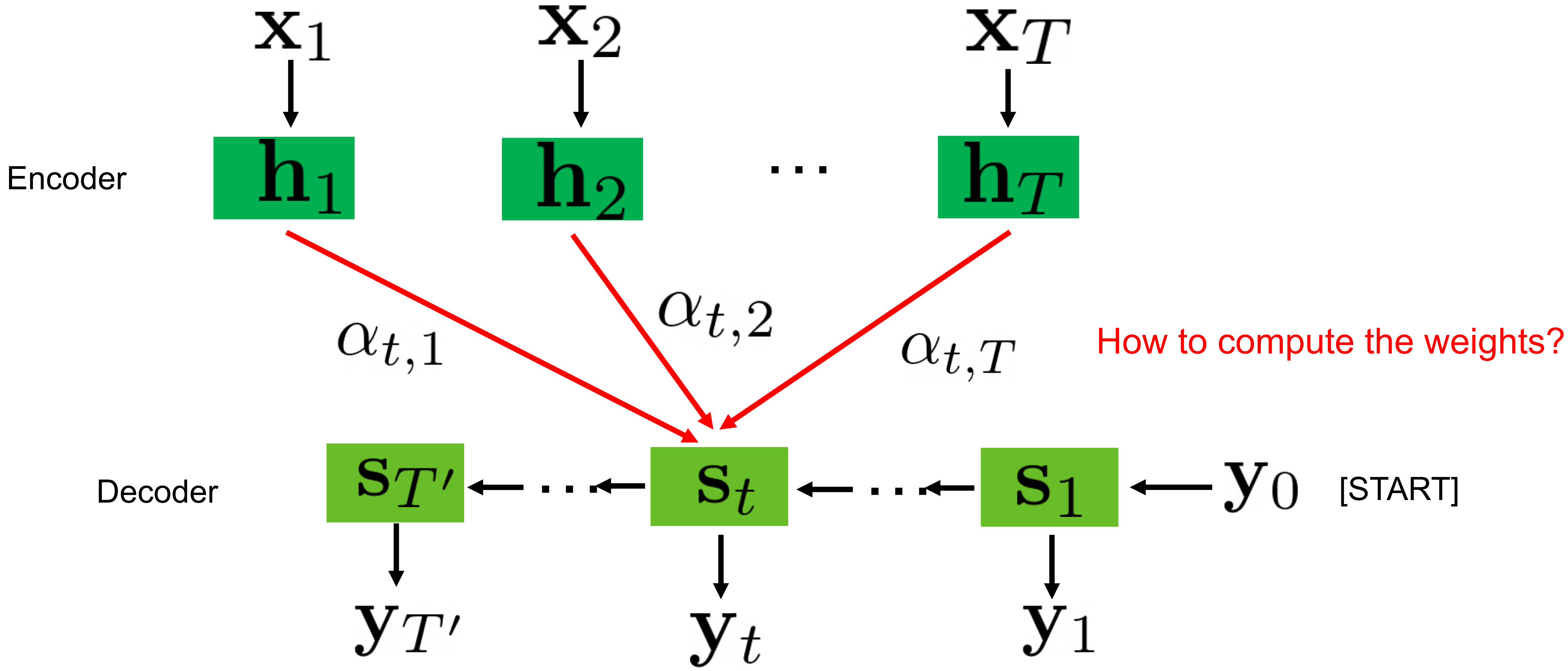
Transformer

No recurrence

Attention only

- Global dependencies between input and output
- More parallelization compared to RNNs

Transformer: Encoder-Decoder with 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

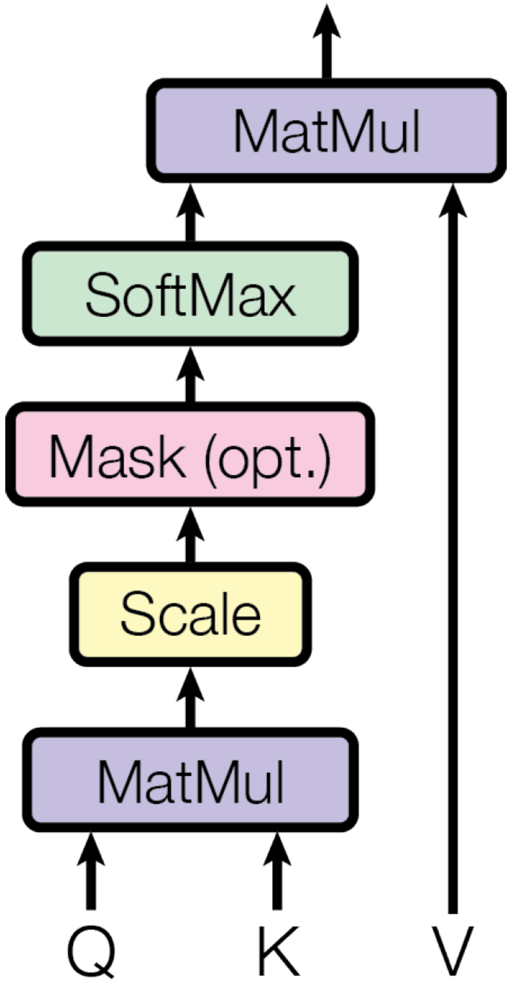
Scaled Dot-Product Attention

- Keys $K : m \times d_k$
- Values $V : m \times d_v$
- n queries $Q : n \times d_k$

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

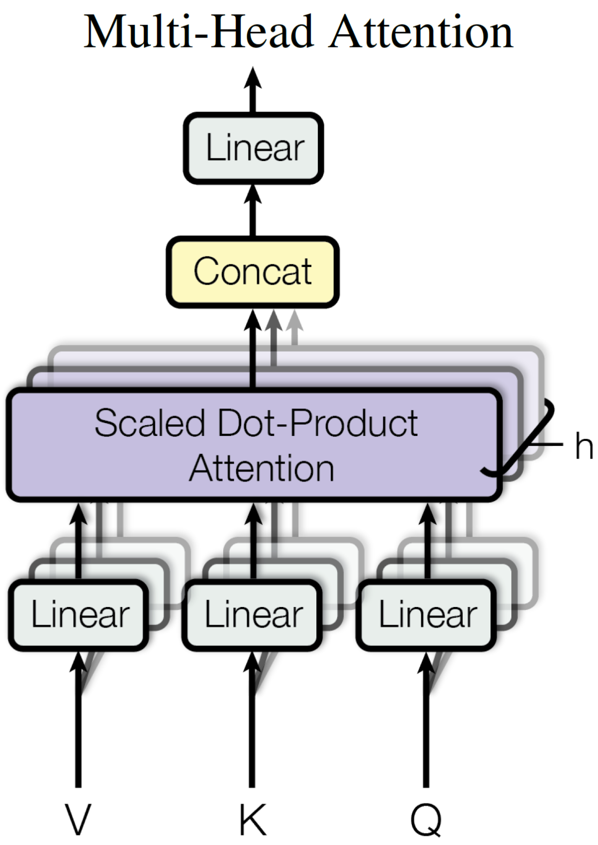
$$n \times d_v$$

weights



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

Transformer: Attention



Multi-Head Attention

- Suppose the latent vector is with dimension d_{model}

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

$$\text{head}_{\text{i}} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$ Projection

$n \times d_v$

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

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_{\text{h}})W^O$$

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

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

Transformer: Encoder

Self-attention

- Keys, values and queries are all the same
- n input tokens $n \times d_{\text{model}}$

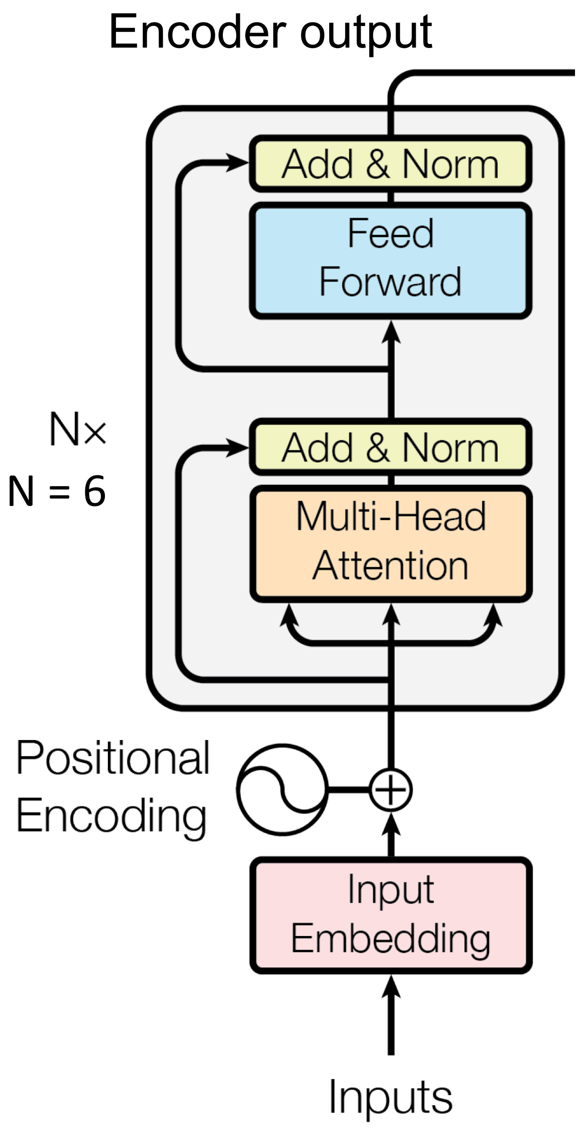
$$\mathrm{MultiHead}(Q, K, V)$$

- Residual connection

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

- Layer normalization $a^l \coloneqq \gamma \hat{a}^l + \beta = LN_{\gamma,\beta}(a^l)$

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



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

Transformer: Encoder

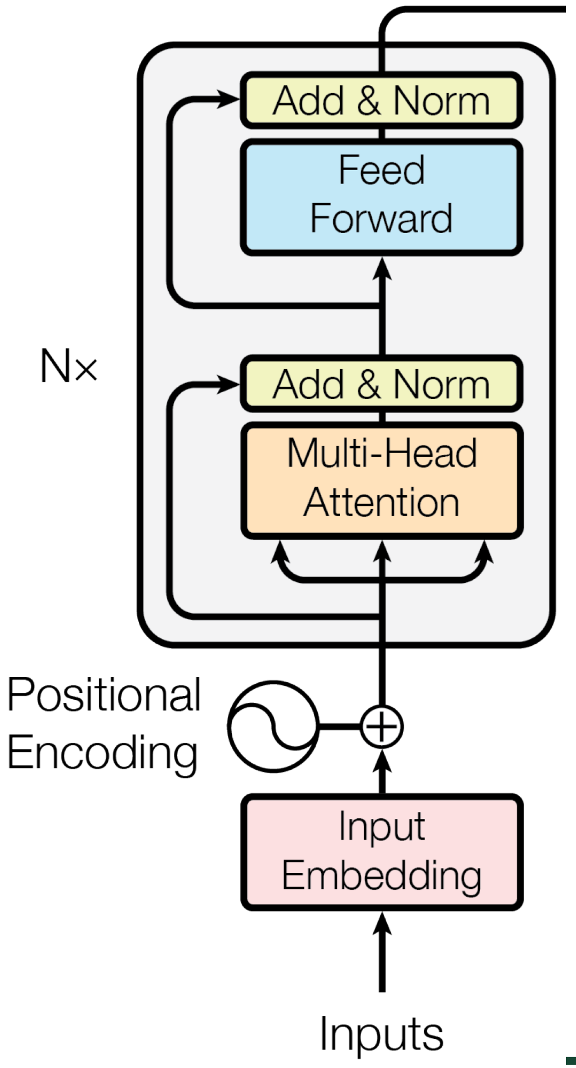
Feed Forward Network

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

Positional encoding

- Make use the order of the sequence
- With dimension d_{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

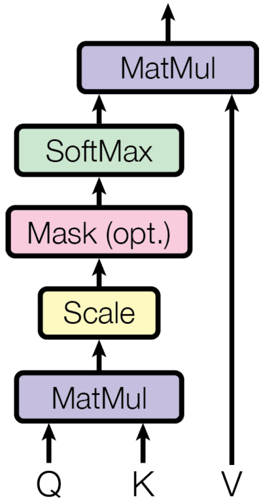
Transformer: Decoder

Output embedding

[START]

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

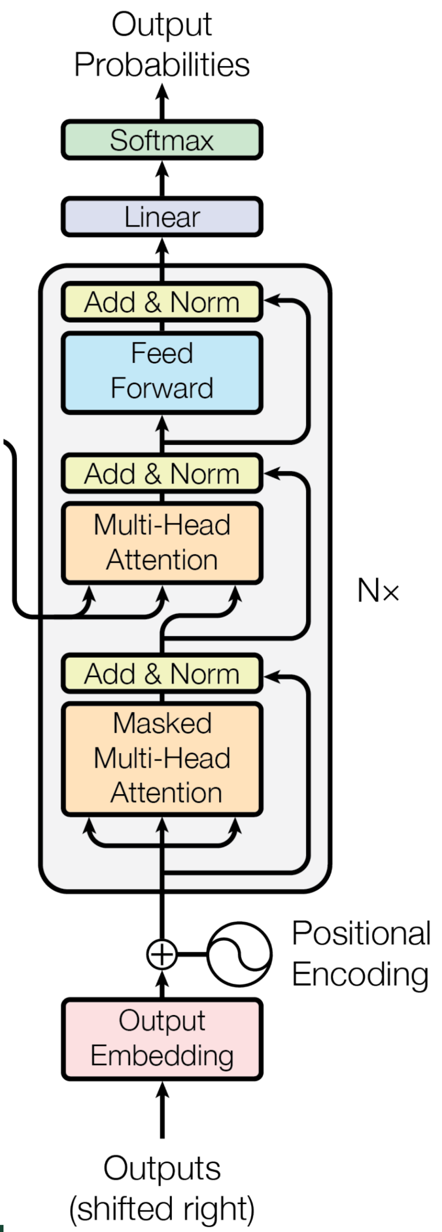
Shifted right by one position and insert the start token



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

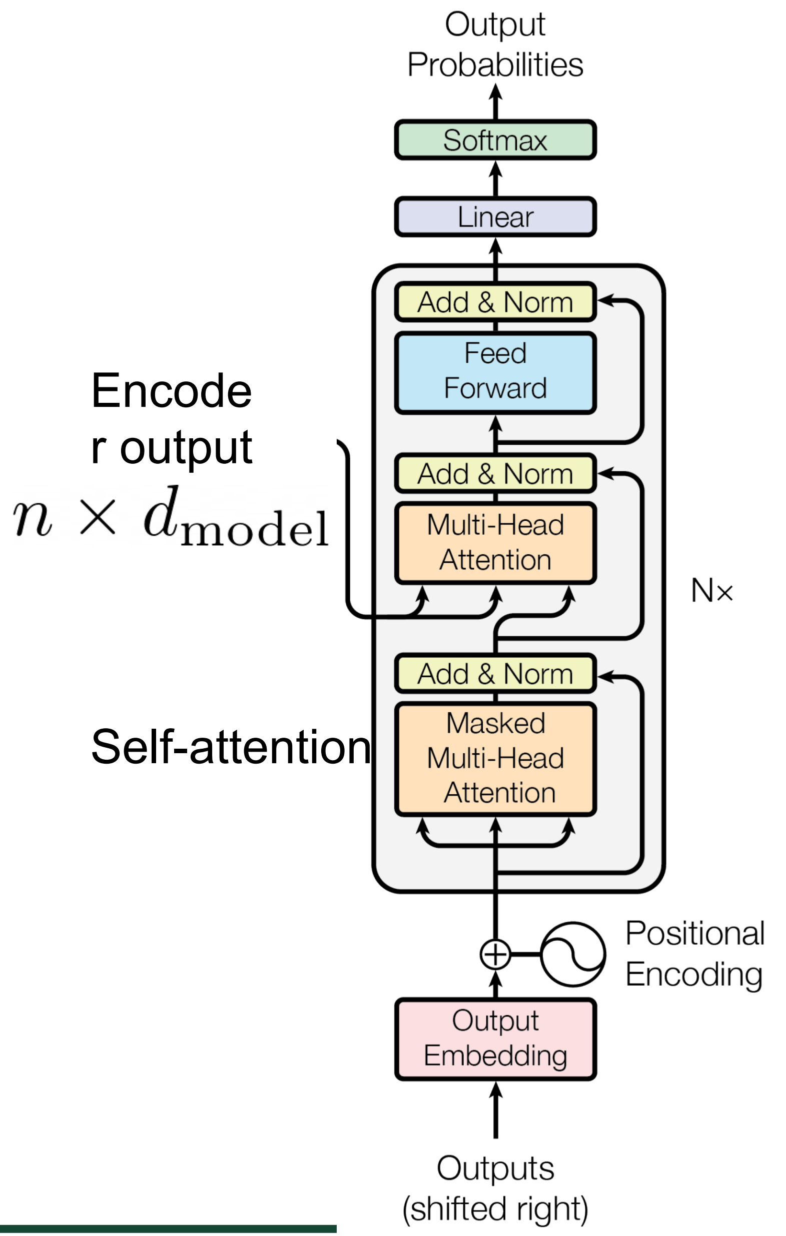
Transformer: Decoder

Encoder-decoder attention

- (Key, value): encoder output
- Queries: decoder output
- Every position in the decoder attends to all positions in the input sequence

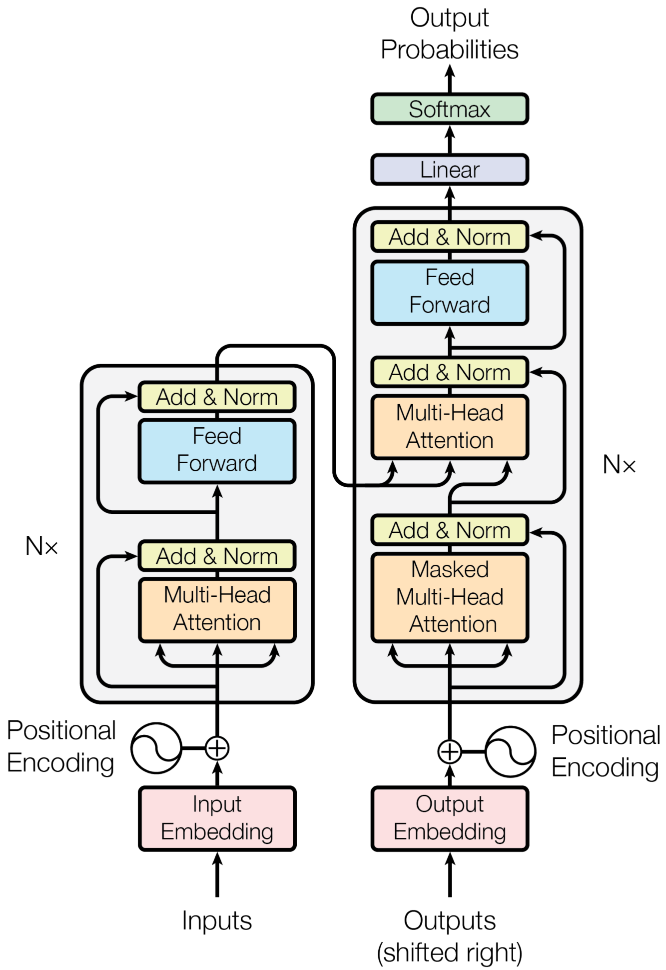
Softmax

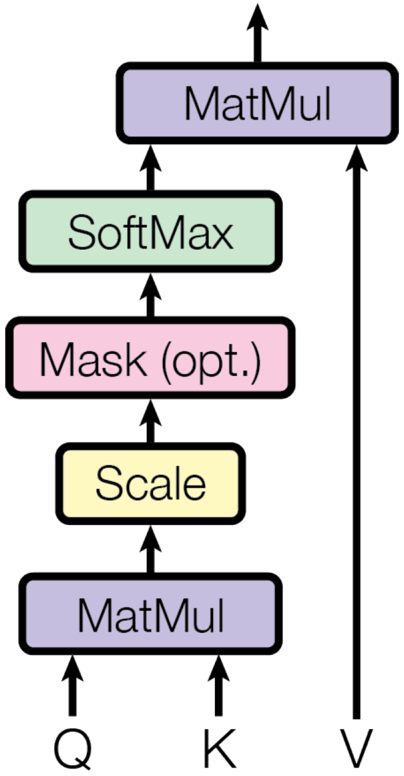
- Predicts next-token probabilities

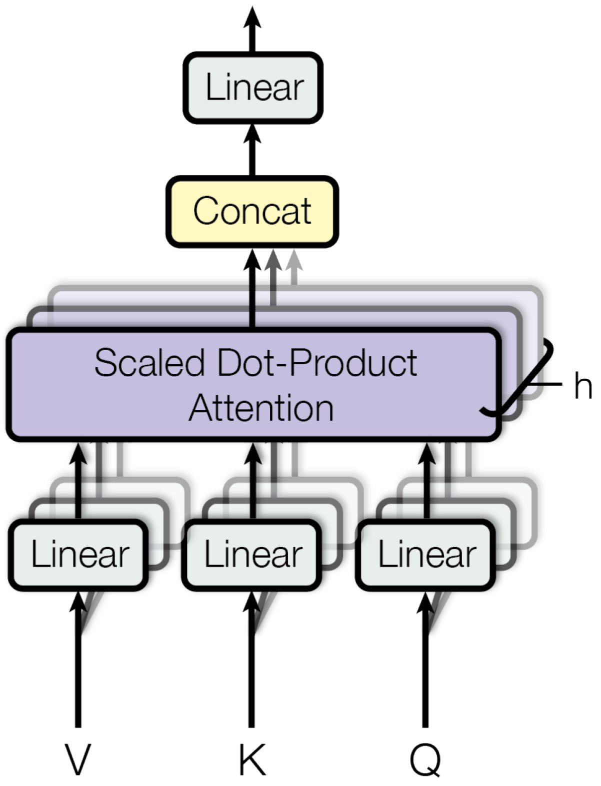


Attention is all you need. Vaswani et al., NeurIPS'17

Transformer

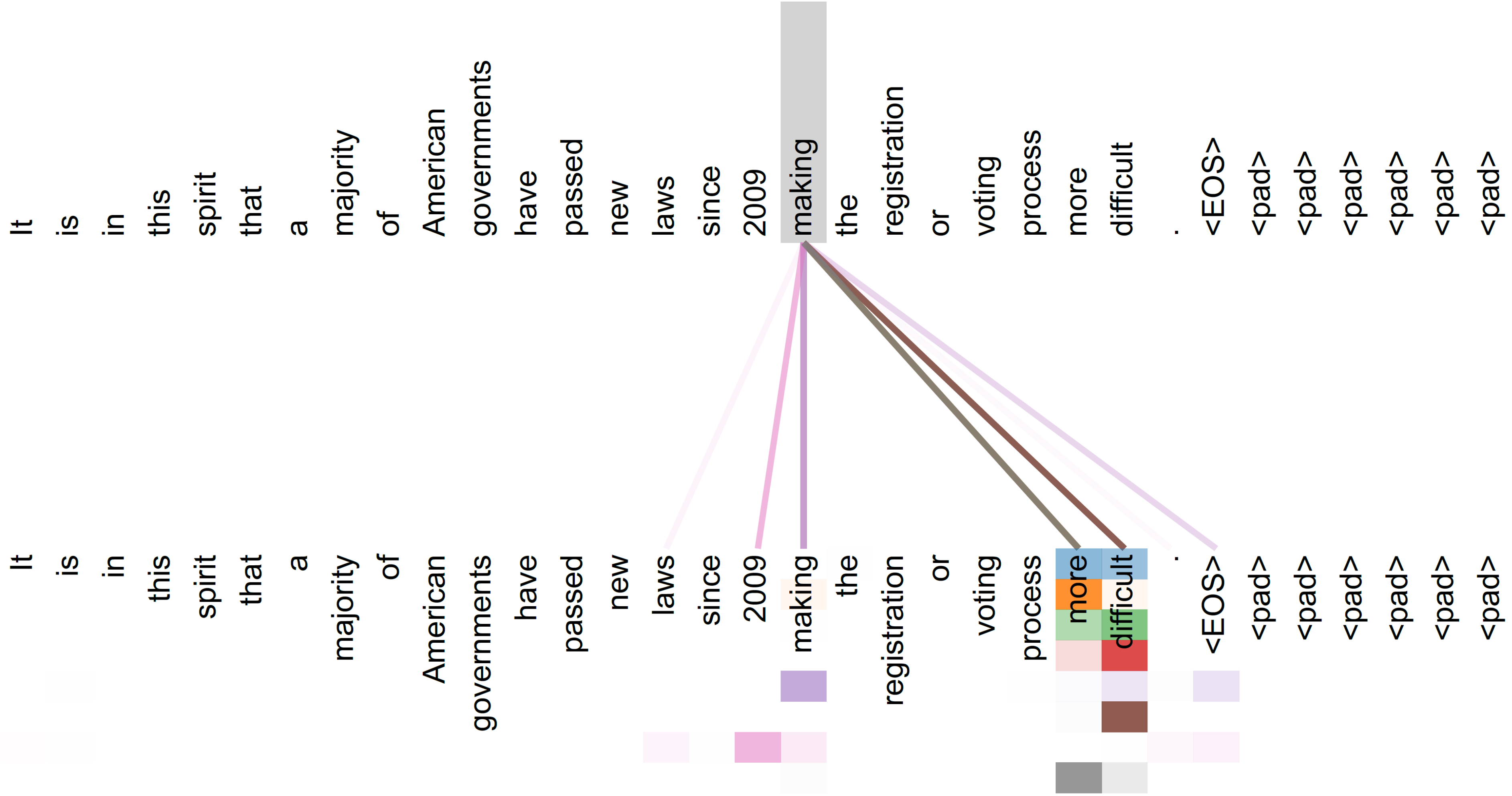






Attention is all you need. Vaswani et al., NeurIPS’17

Transformer: Attention Visualization



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

Vision Transformer

Convert an image into a sequence of “token”

Input embedding by linear projection

d_{model}

$$\mathbf{x}_p^1\mathbf{E};\ \mathbf{x}_p^2\mathbf{E};\ \cdots;\ \mathbf{x}_p^N\mathbf{E} \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}$$

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. Dosovitskiy et al., ICLR’21

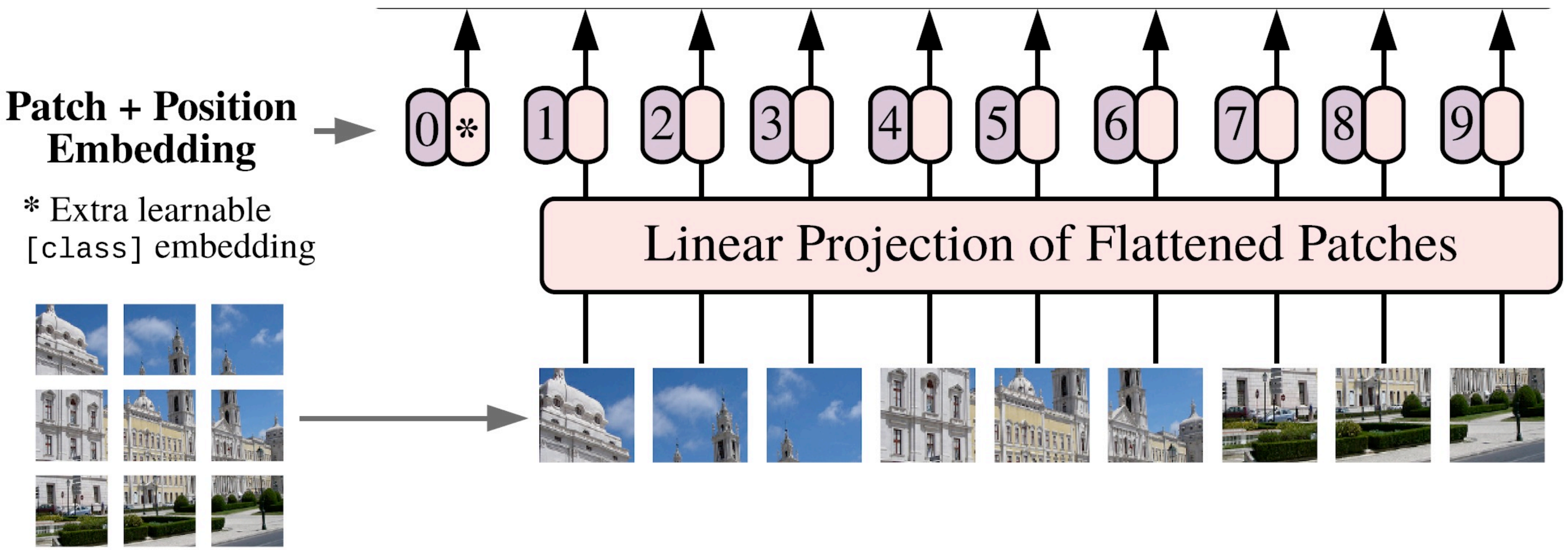
Vision Transformer

Adding positional embedding

Prepend a learnable embedding $\mathbf{z}_0^0$

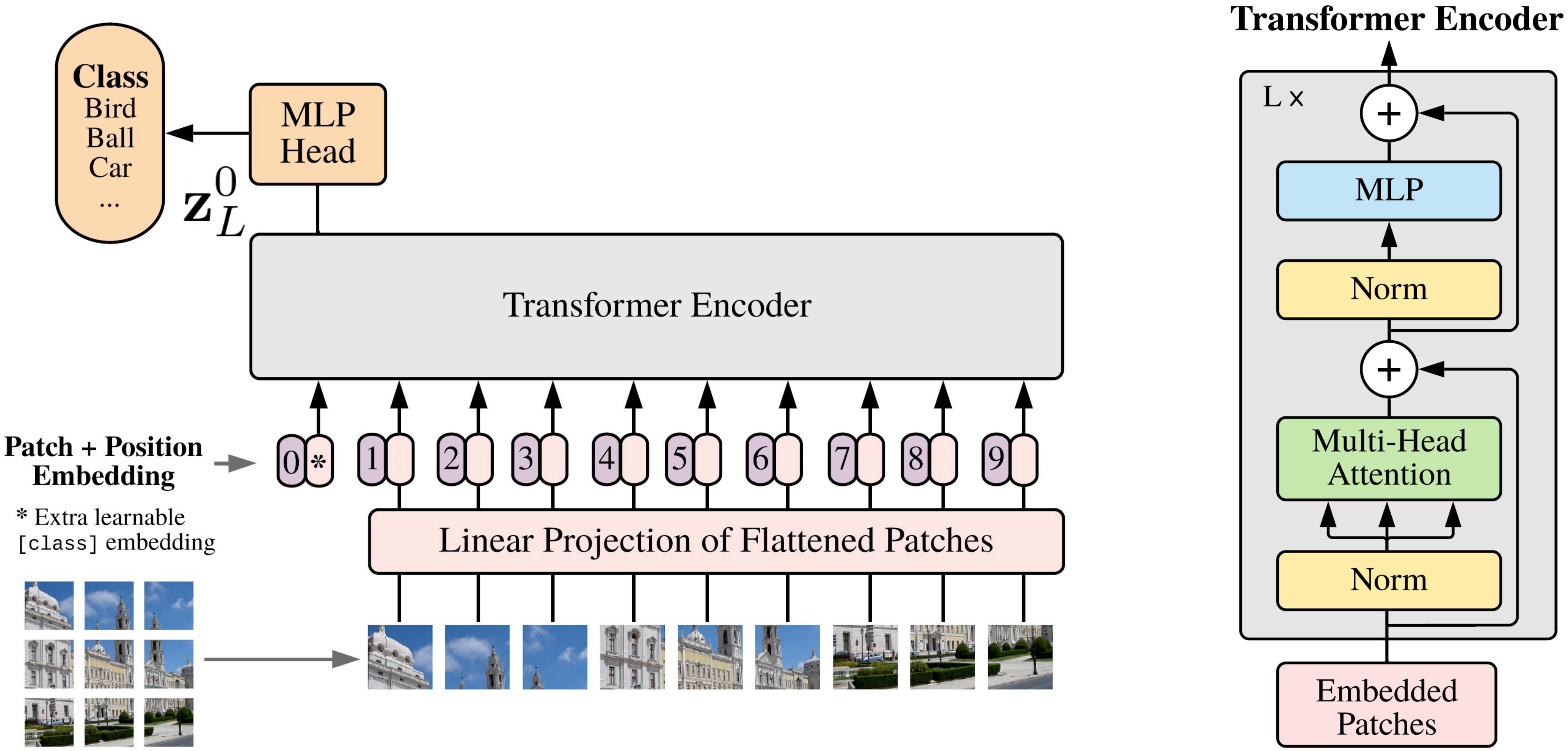
$\mathbf{z}_L^0$ Will be used as the image representation

After L attention layers



AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. Dosovitskiy et al., ICLR’21

Vision Transformer



AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. Dosovitskiy et al., ICLR'21

Vision Transformer

Pretrain on a large-scale dataset

Fine-tune on different tasks

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

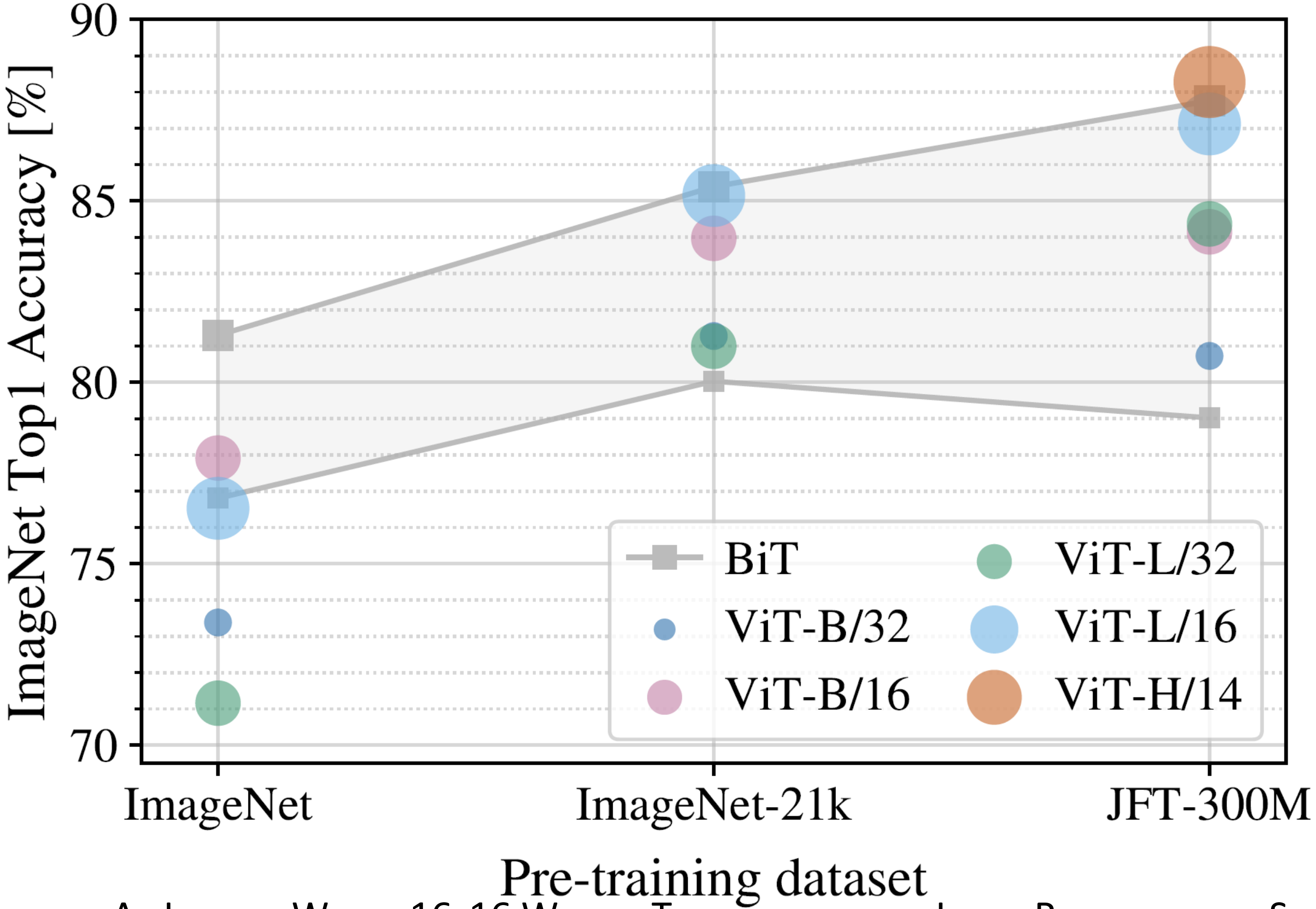
An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Dosovitskiy et al., ICLR'21

Vision Transformer

1k classes
and 1.3M images

21k classes and
14M images

18k classes and
303M



Big Transfer (BiT)

- ResNets-based transfer

Vision transformer works better when pre-trained on large-scale dataset

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. Dosovitskiy et al., ICLR'21

Summary

Transformers

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

Vision transformers

- Works better when pre-trained on large scale datasets (e.g., 300M images)

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

Vision transformer: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale https://arxiv.org/abs/2010.11929