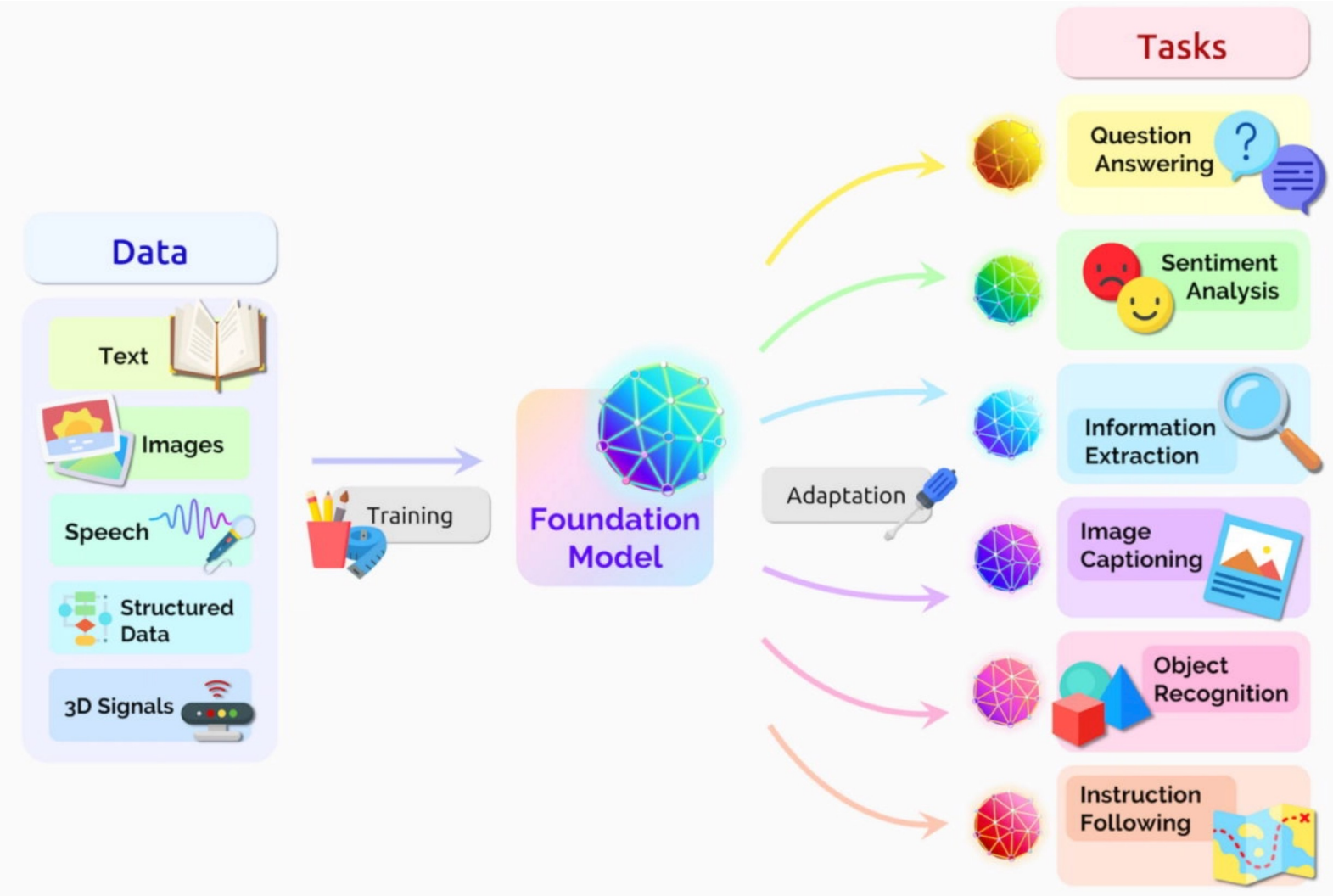
Computer Vision:
Transformer

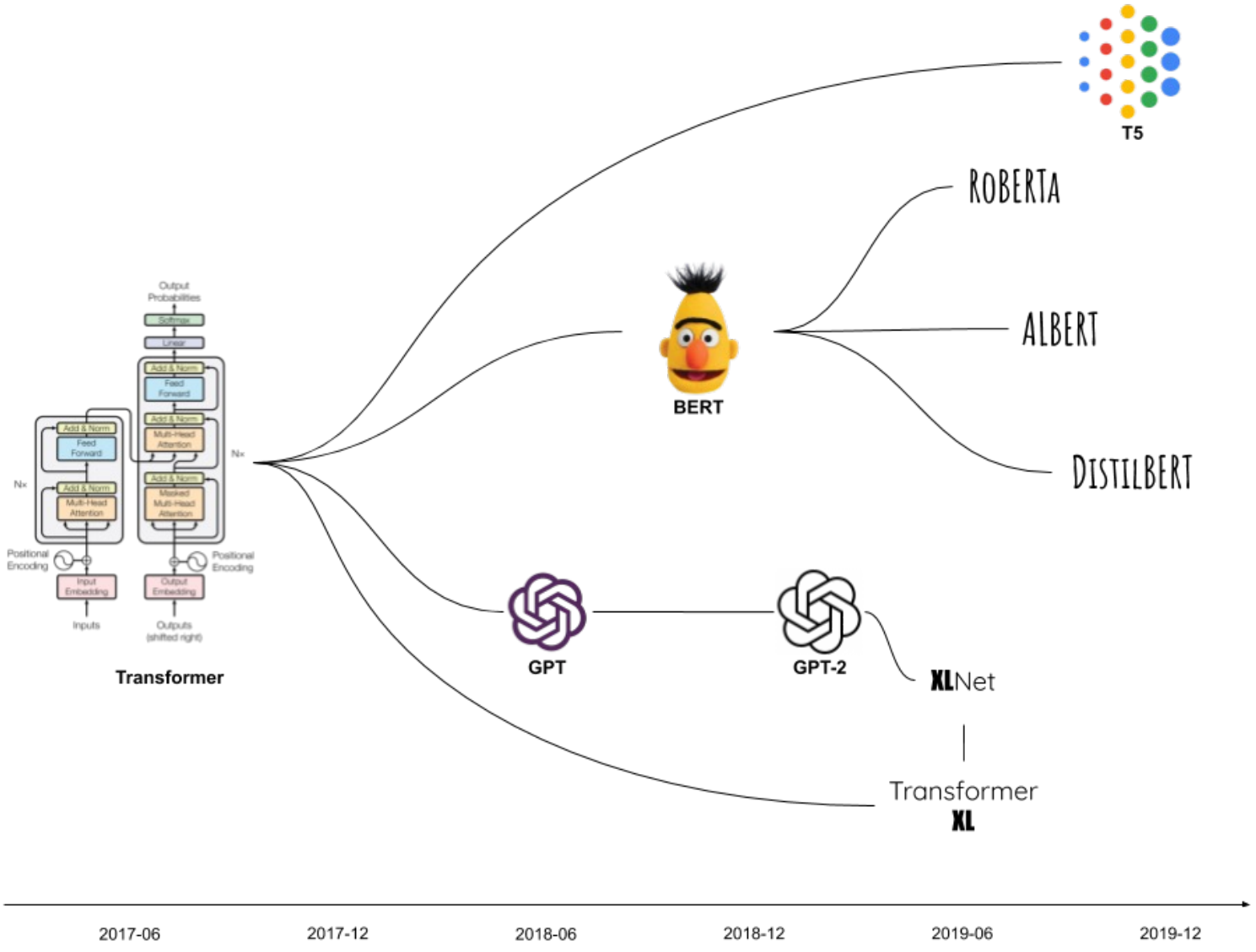
Siheng Chen 陈思衡

Why Transformer?



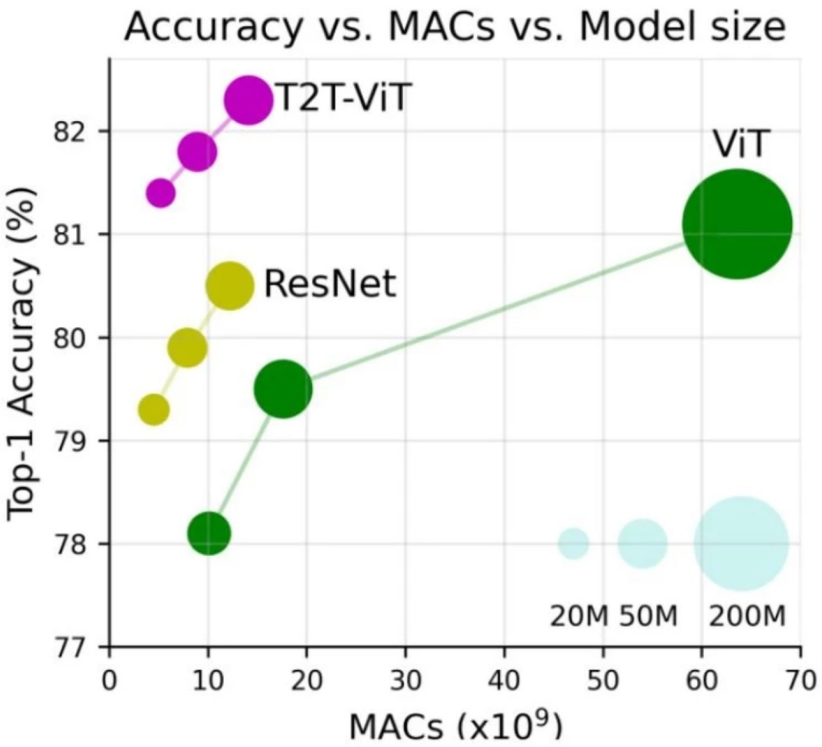
https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/

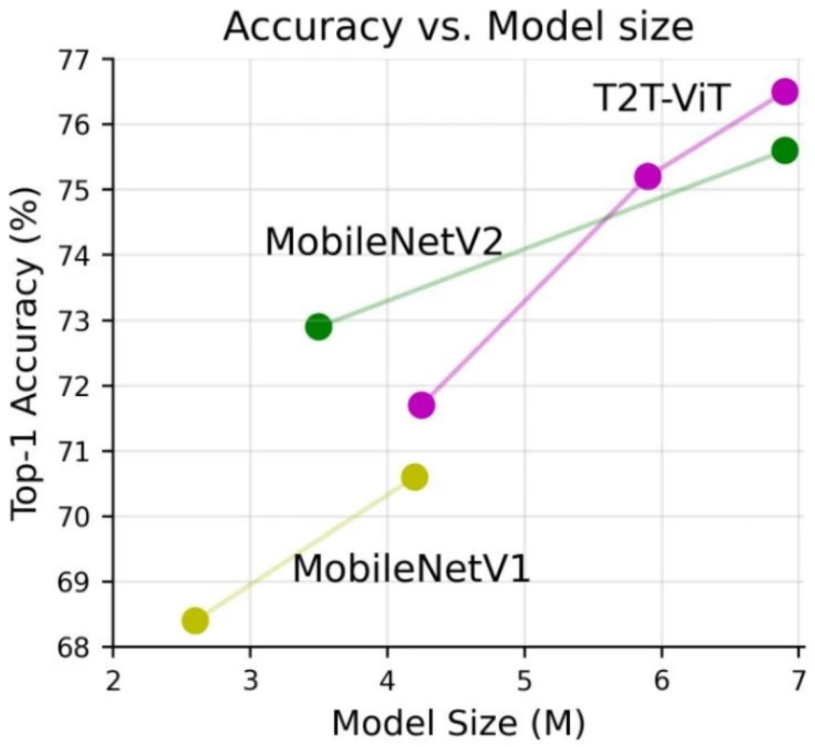
Why Transformer?



https://www.factored.ai/2021/09/21/an-intuitive-explanation-of-transformer-based-models/

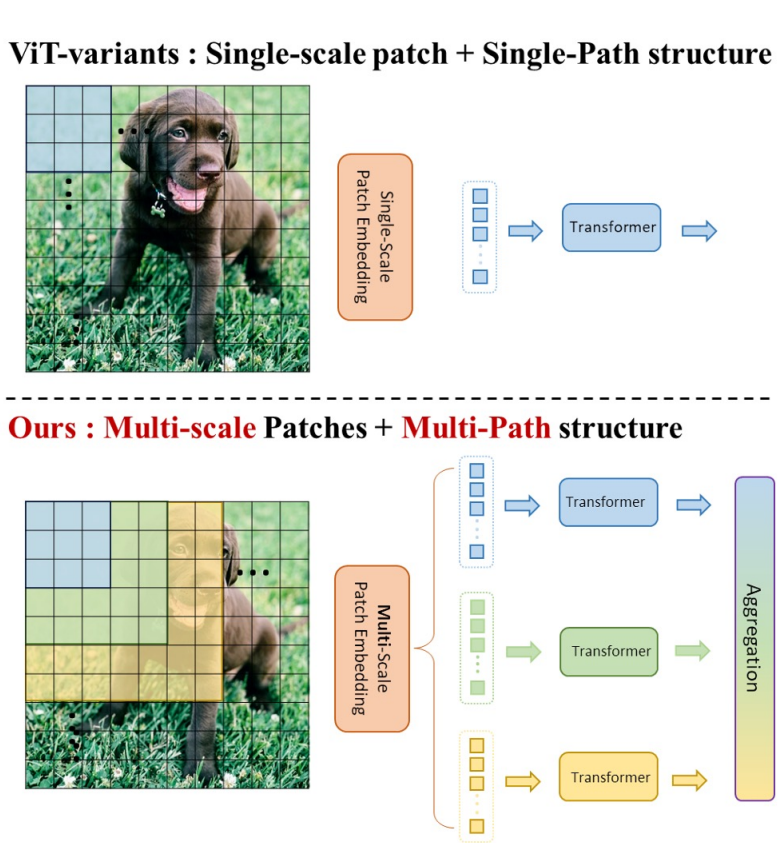
Why Transformer?





https://viso.ai/deep-learning/vision-transformer-vit/

Why Transformer?



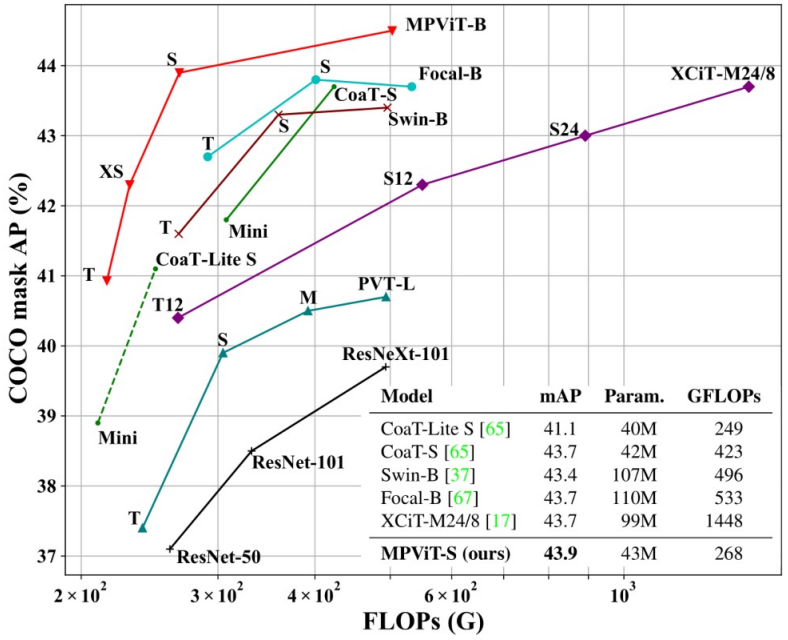
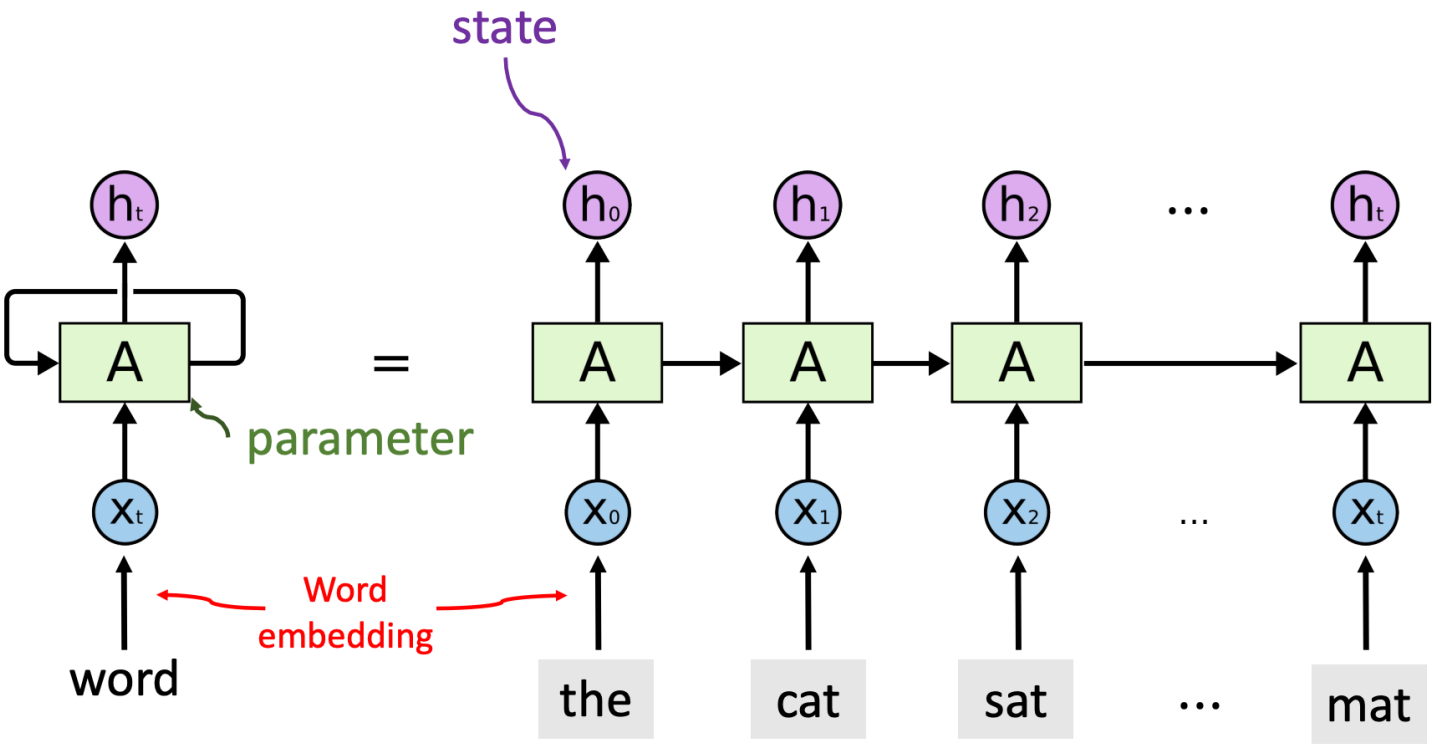


Figure 2. **FLOPs vs. COCO mask AP** on Mask R-CNN. MPViTs outperform state-of-the-art Vision Transformers while having fewer parameters and FLOPs. B, S, XS, and T at the end of the model names denote base, small, extra-small and tiny respectively. Complete results are in Table 3.

Youngwan Lee and Jonghee Kim and Jeffrey Willette and Sung Ju Hwang, MPViT : Multi-Path Vision Transformer for Dense Prediction, CVPR 2022

Recurrent NN

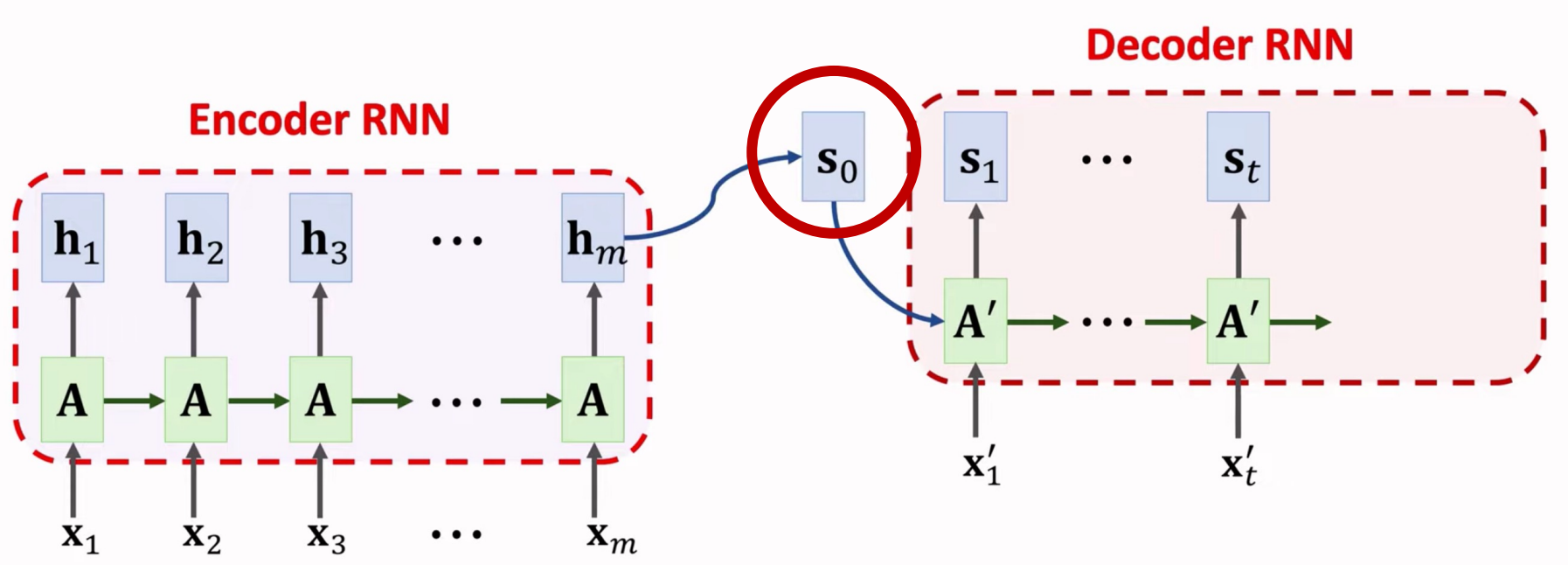


Roadmap

1. Recurrent neural network (RNN)
2. Seq2seq model
3. Attention for Seq2seq model
4. Self-attention for RNN
5. Attention without RNN: Attention layer / self-attention layer
6. Transformer
7. Vision Transformer

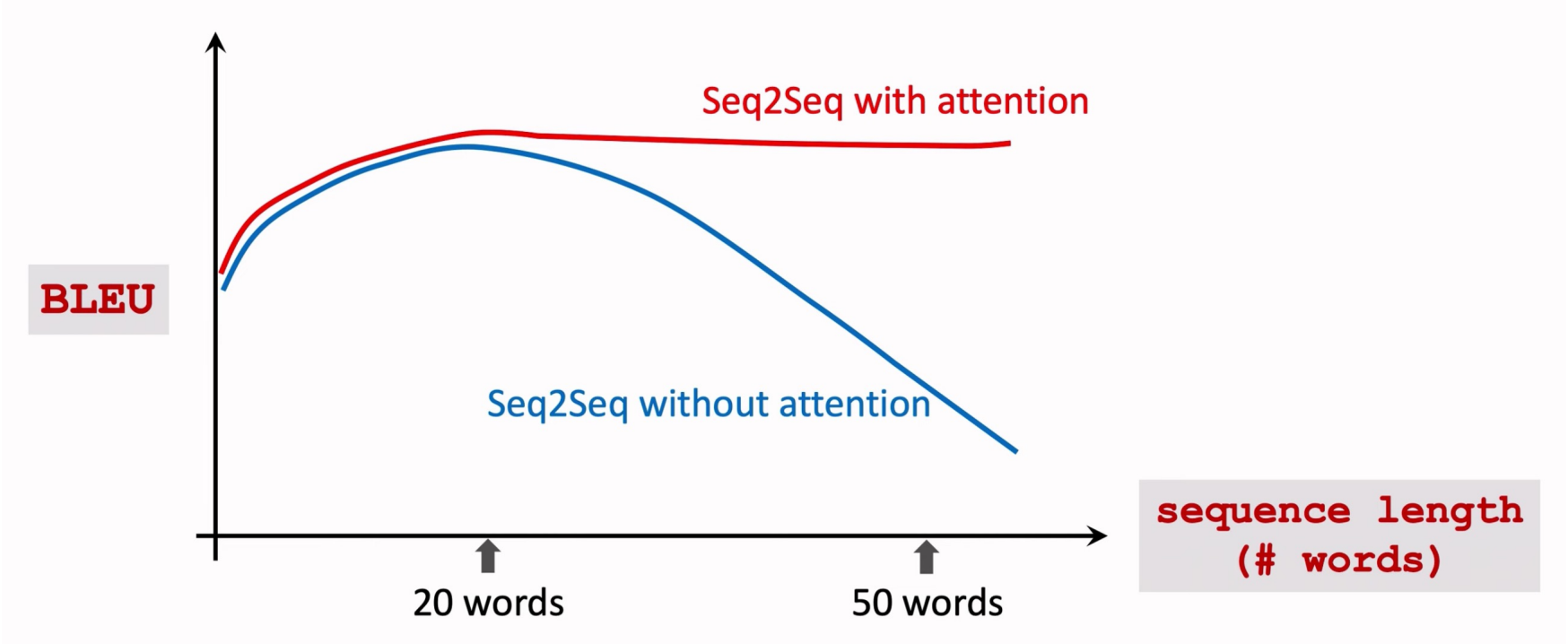
Seq2Seq model

Translation: from one sentence to another sentence



Problem: The final state is incapable of remembering a long sequence

Seq2Seq model



Problem: The final state is incapable of remembering a long sequence

Solution: Attention

Attention for Seq2Seq model

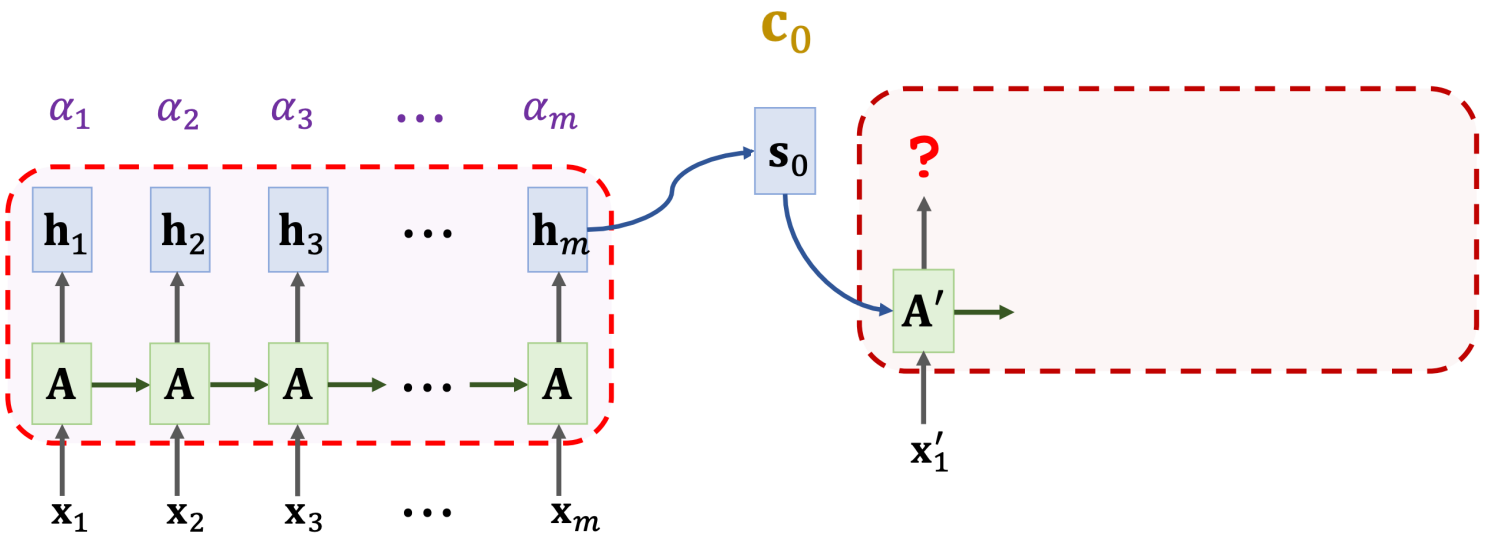
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

SimpleRNN:

$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix}\mathbf{x}_1' \\ \mathbf{s}_0\end{bmatrix} + \mathbf{b}\right)$$

SimpleRNN + Attention:

$$\mathbf{s}_1 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix}\mathbf{x}_1' \\ \mathbf{s}_0 \\ \mathbf{c}_0\end{bmatrix} + \mathbf{b}\right)$$



Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

Attention for Seq2Seq model

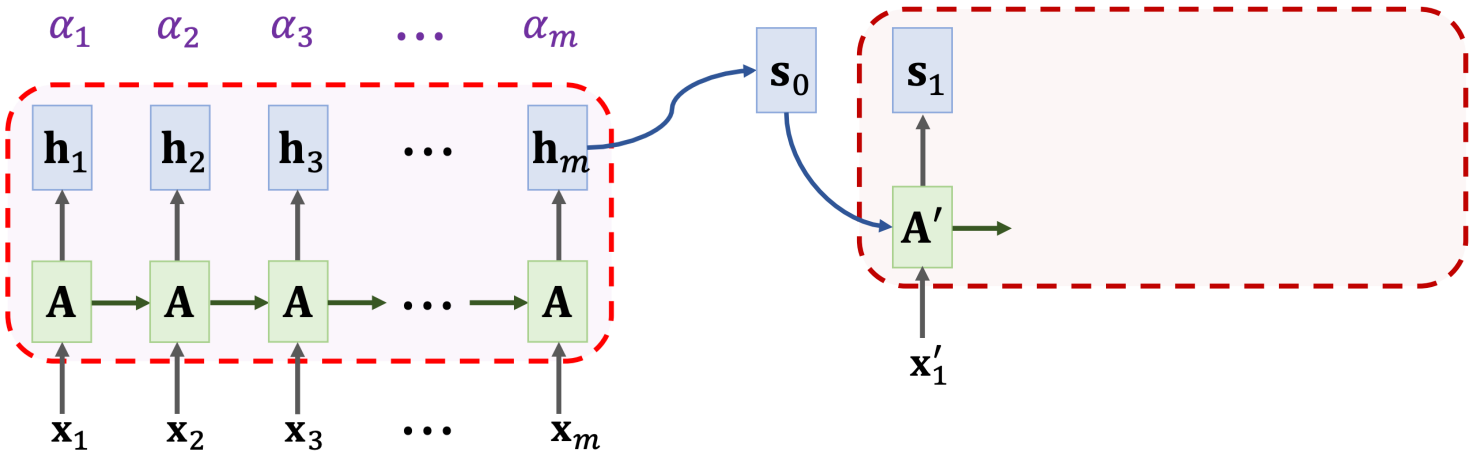
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_1)$.

$$\mathbf{s}_2 = \tanh\left(\mathbf{A}' \cdot \begin{bmatrix} \mathbf{x}'_2 \\ \mathbf{s}_1 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b}\right)$$

re-compute each time!

$\mathbf{c}_0$

$\mathbf{c}_1 = \alpha_1 \mathbf{h}_1 + \cdots + \alpha_m \mathbf{h}_m$



Bahdanau, Cho, & Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

Attention for Seq2Seq model

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

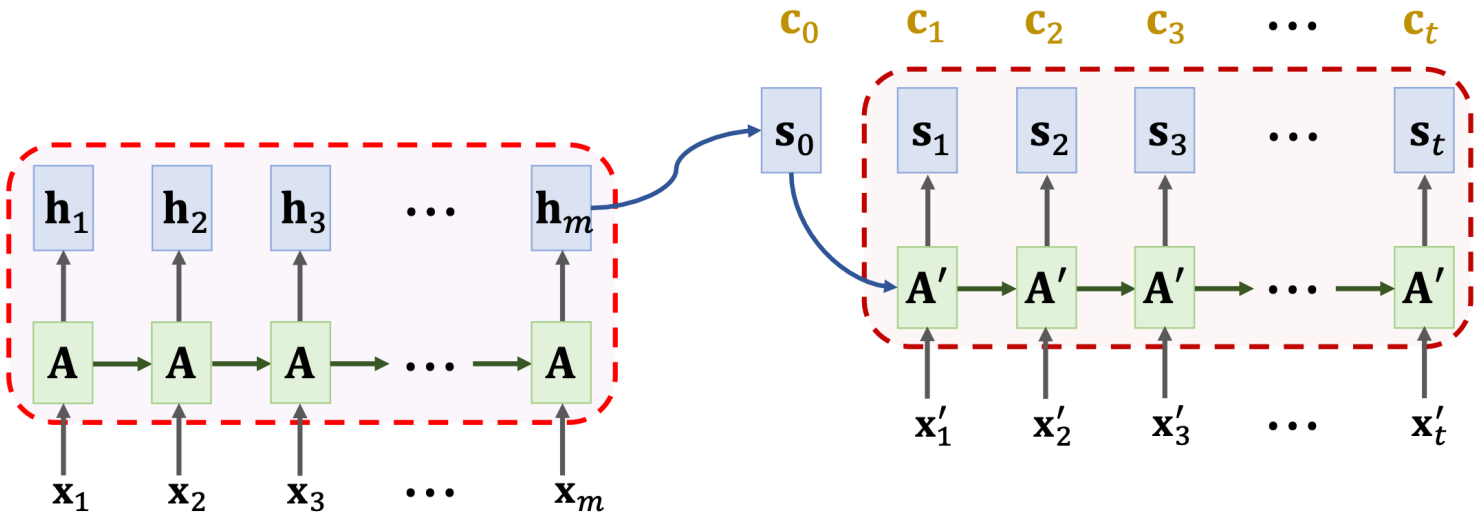
Attention weight computation

1. Linear maps:
   - $\mathbf{k}_i = \mathbf{W}_K \cdot \mathbf{h}_i$, for $i = 1$ to m.
   - $\mathbf{q}_0 = \mathbf{W}_Q \cdot \mathbf{s}_0$.
2. Inner product:
   - $\tilde{\alpha}_i = \mathbf{k}_i^T \mathbf{q}_0$, for $i = 1$ to m.
3. Normalization:
   - $[\alpha_1, \cdots, \alpha_m] = \text{Softmax}\,([\tilde{\alpha}_1, \cdots, \tilde{\alpha}_m])$.

Attention for Seq2Seq model

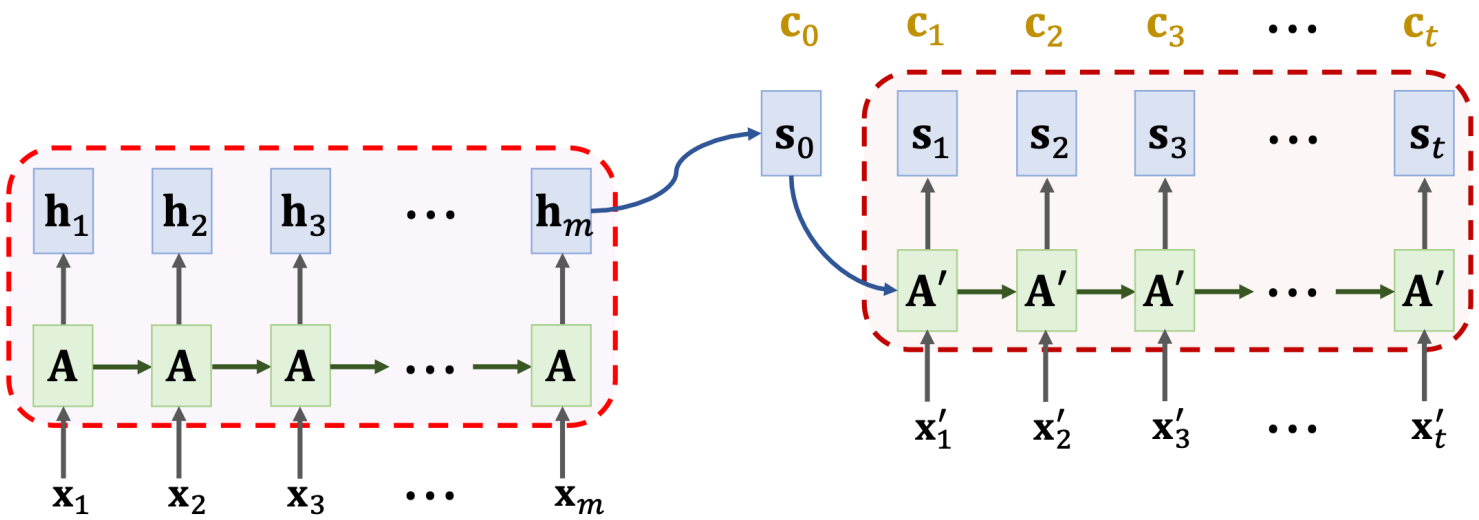
Question: How many attention weights have been computed?

- To compute each c_i, we compute ***m*** weights
- The decode has ***t*** states, so there are totally ***mt*** weights



Attention for Seq2Seq model

Without Attention	With Attention
Time complexity: $O(m+t)$	Time complexity: $O(mt)$



Attention for Seq2Seq model

Advantages

- Standard Seq2Seq model: the decoder looks at only its current state
- Attention: decoder additionally looks at all the states of the encoder
- Attention: decoder knows where to focus

Disadvantages

- Much higher time complexity.

Self-Attention for RNN

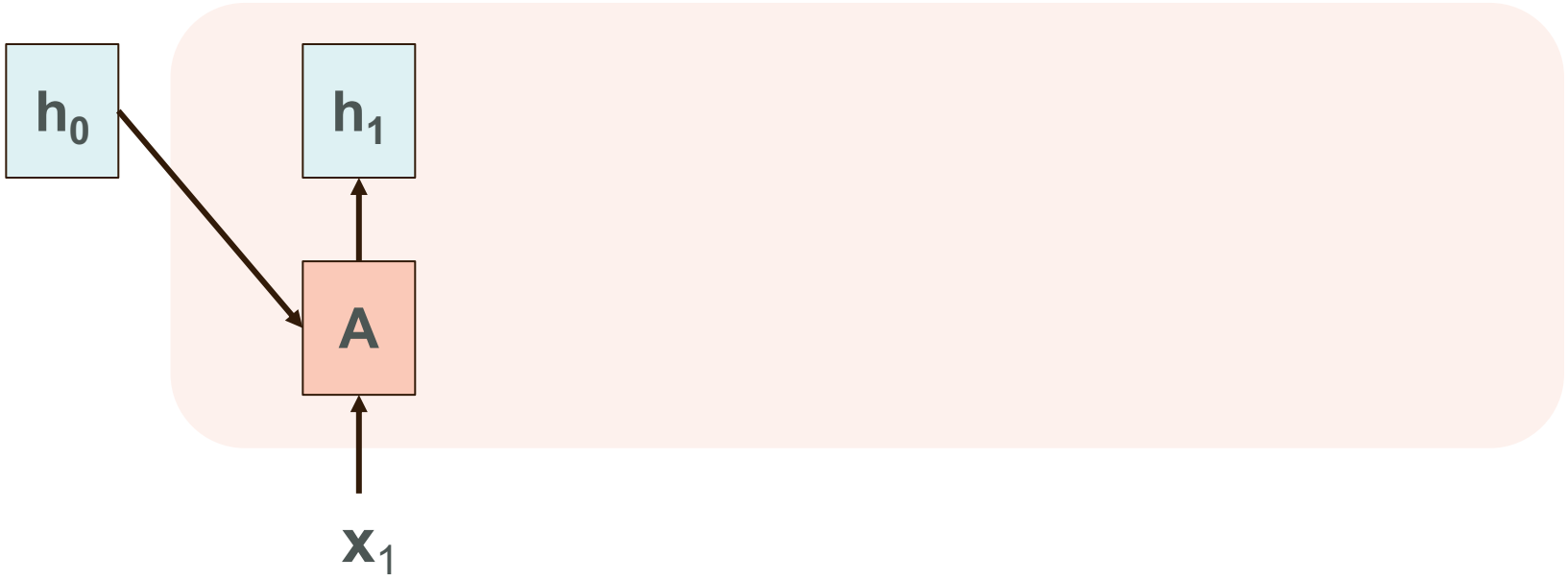
SimpleRNN:

$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{h}_0 \end{bmatrix} + \mathbf{b}\right)$$

SimpleRNN + Self-Attention:

$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{c}_1 \end{bmatrix} + \mathbf{b}\right)$$

$c_0 = 0$ $c_1 = h_1$



Cheng, Dong, Lapata, Long Short-Term Memory-Networks for Machine Reading, In EMNLP, 2016.

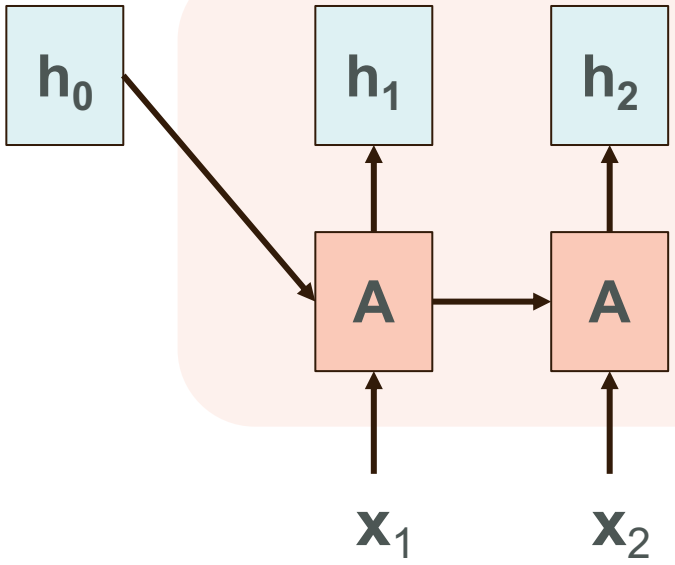
Self-Attention for RNN

SimpleRNN + Self-Attention:

$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{c}_0 \end{bmatrix} + \mathbf{b}\right)$$

Weights: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{h}_2)$.

$\mathbf{c}_0 = 0$ $\mathbf{c}_1 = \mathbf{h}_1$ $\mathbf{c}_2 = \alpha_1 \mathbf{h}_1 + \alpha_2 \mathbf{h}_2$

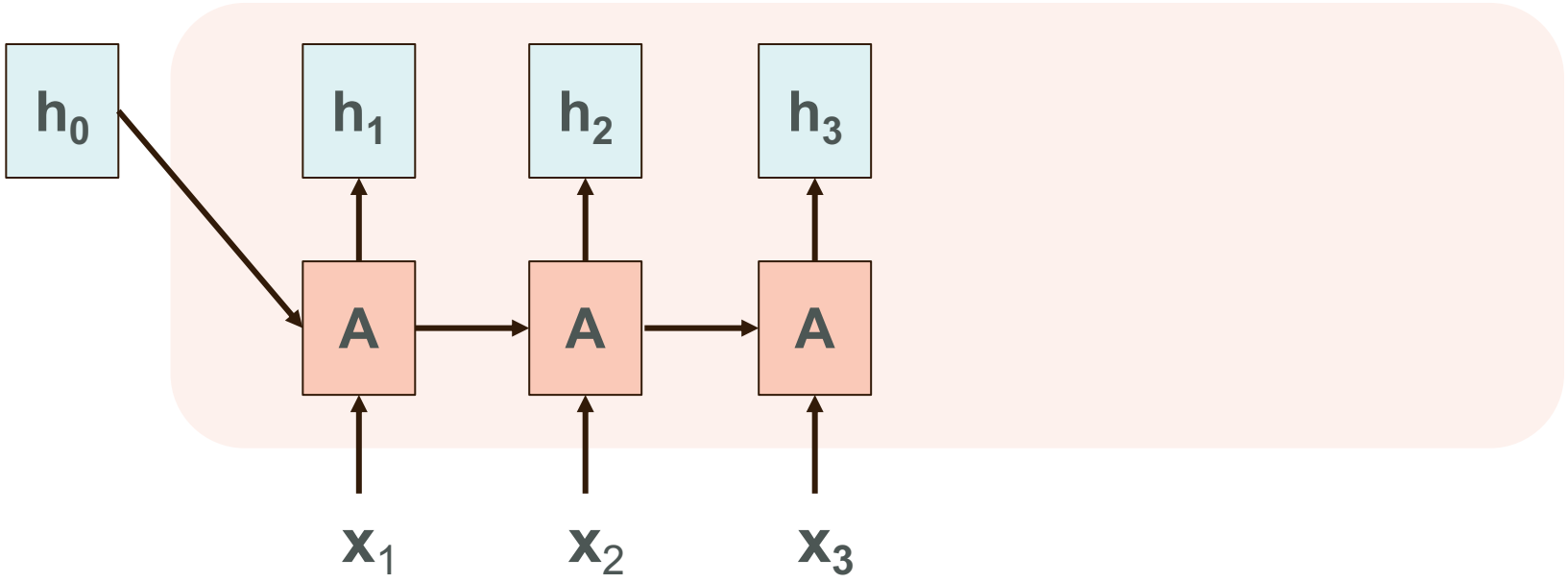


Cheng, Dong, Lapata, Long Short-Term Memory-Networks for Machine Reading, In EMNLP, 2016.

Self-Attention for RNN

$h_i \longrightarrow \alpha_1, \alpha_2, \ldots, \alpha_i \longrightarrow c_i$

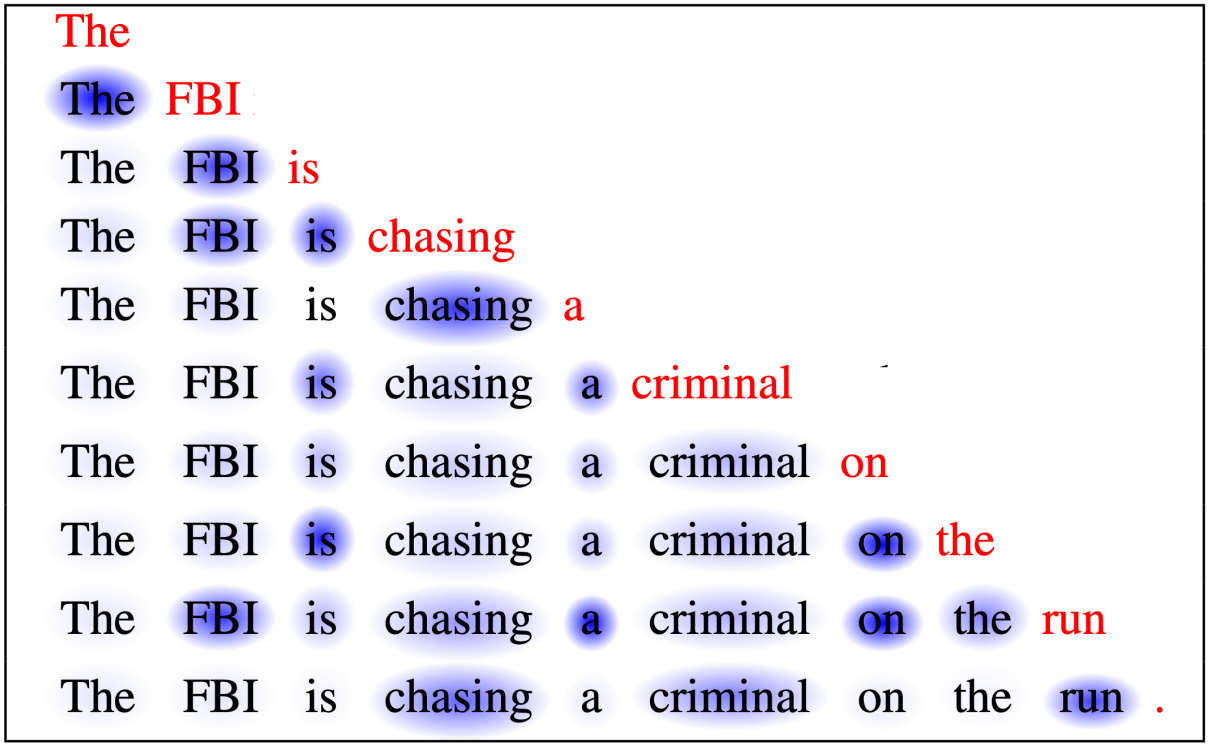
$c_0 = 0 \qquad c_1 = h_1 \qquad c_2 \qquad c_3 = \alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$



Cheng, Dong, Lapata, Long Short-Term Memory-Networks for Machine Reading, In EMNLP, 2016.

Self-Attention for RNN

- With self-attention, RNN is less likely to forget
- Pay attention to the context relevant to the new input



Cheng, Dong, Lapata, Long Short-Term Memory-Networks for Machine Reading, In EMNLP, 2016.

Attention without RNN

Encoder's inputs:

x_1 x_2 ... x_m

Decoder's inputs:

x'_1 x'_2 ... x'_t

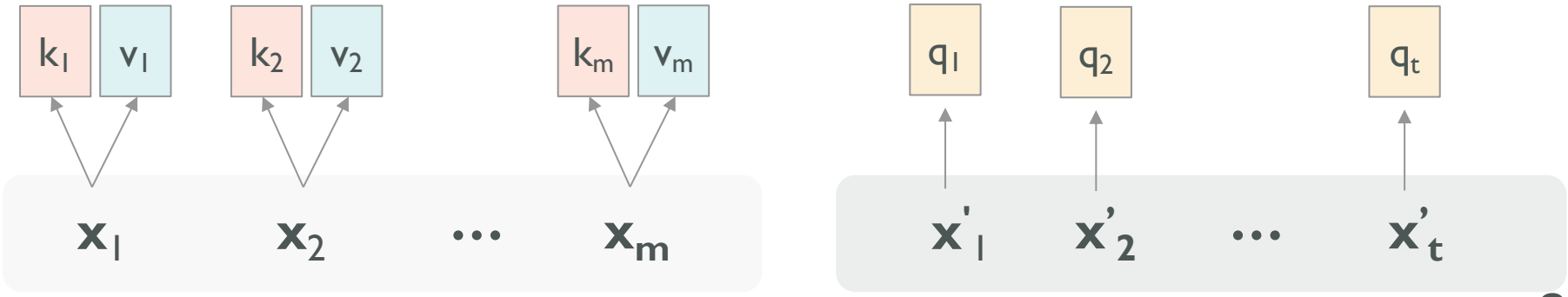
Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention without RNN

Key: $k_i = W_K\, x_i$

Value: $v_i = W_V\, x_i$

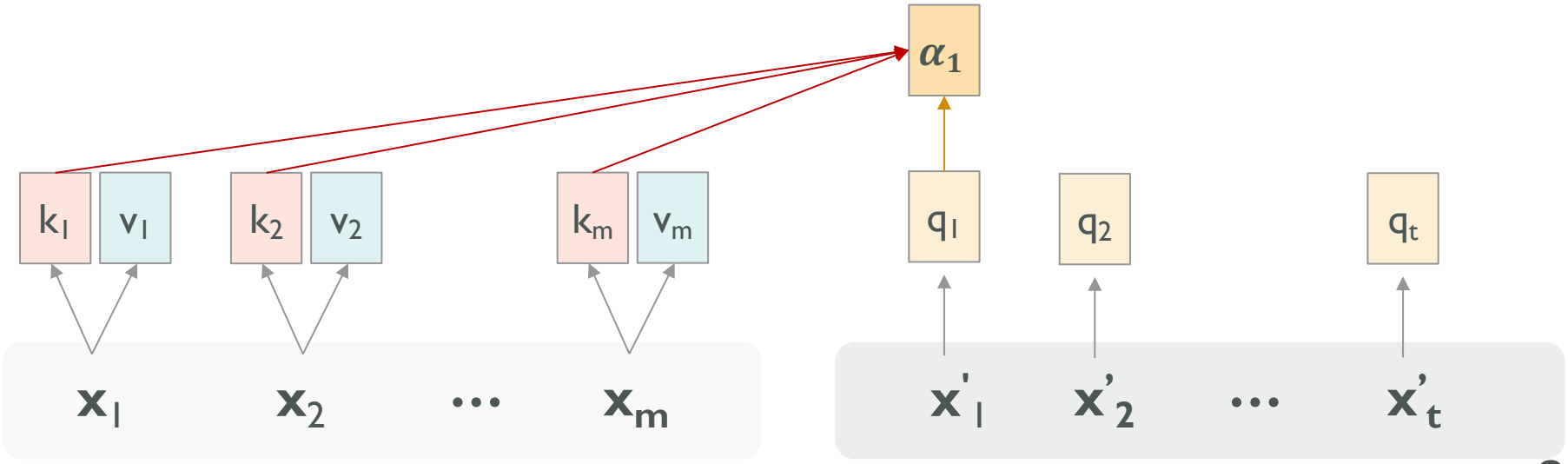
Query: $q_j = W_Q\, x'_j$



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention without RNN

Compute attention weights: $\alpha_1 = \text{softmax}(K^T q_1)$

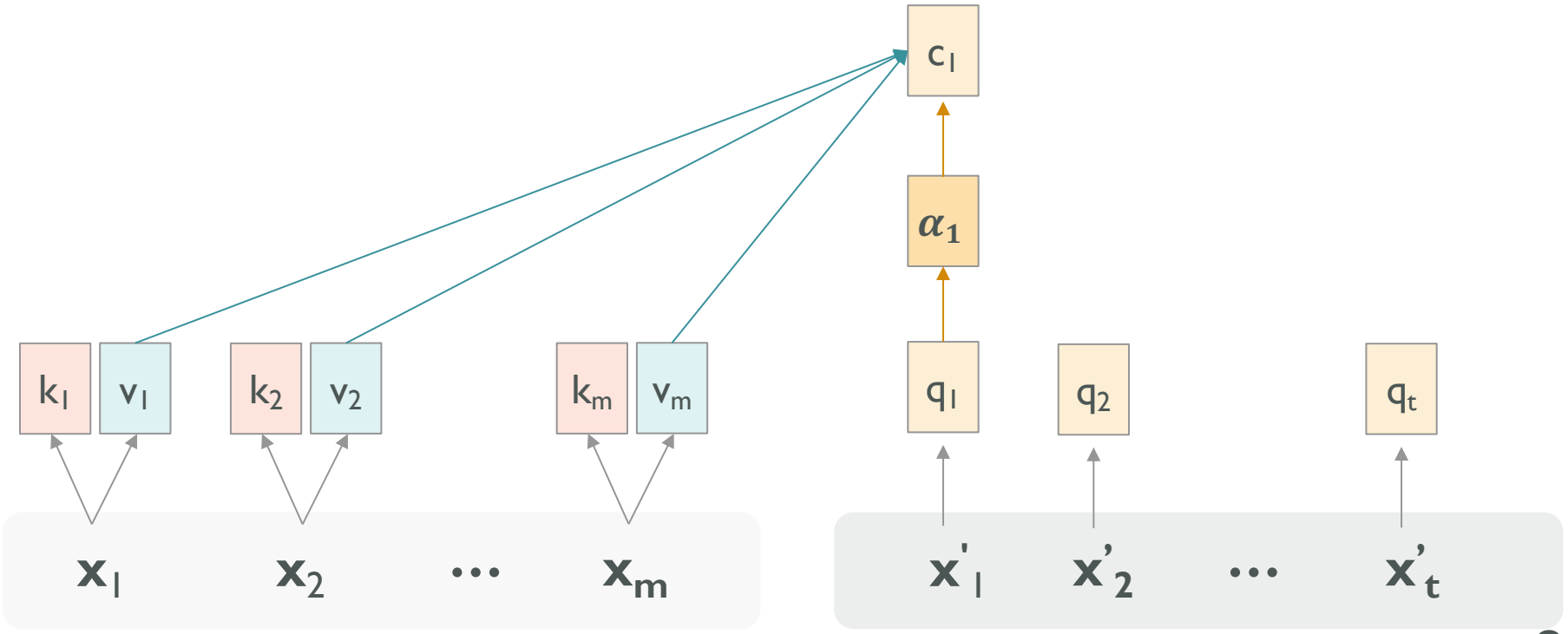


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention without RNN

Compute attention weights: $\boldsymbol{\alpha_1}$ = softmax ($K^T q_1$)

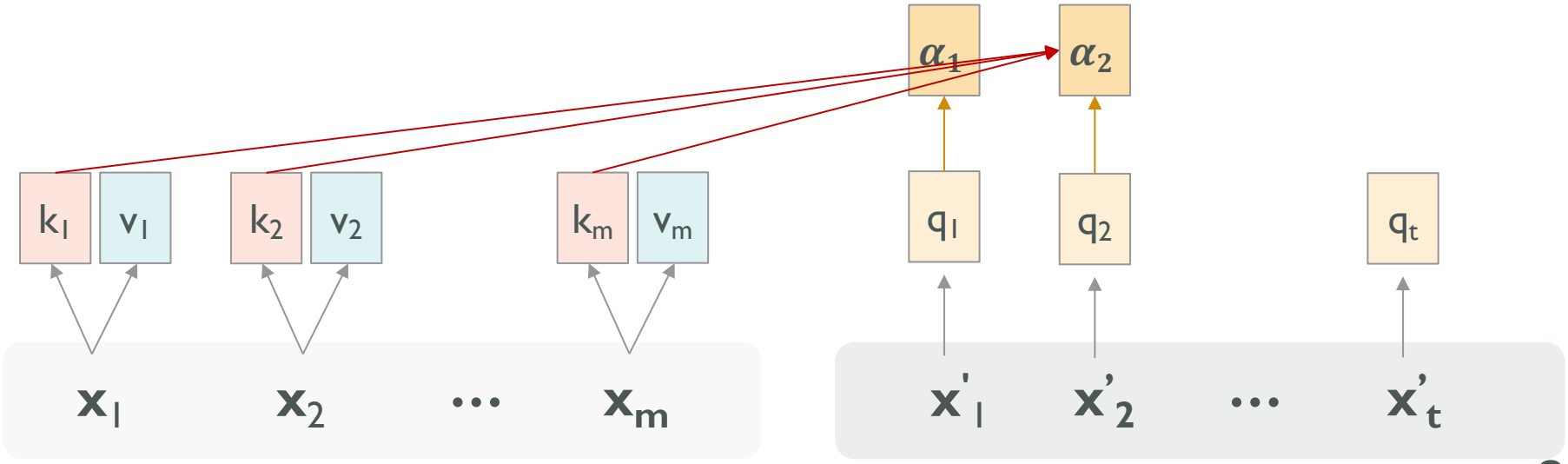
Compute context vector: $\boldsymbol{c_1} = V^T \boldsymbol{\alpha_1}$



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention without RNN

Compute attention weights: $\alpha_2 = \text{softmax}(K^T q_2)$

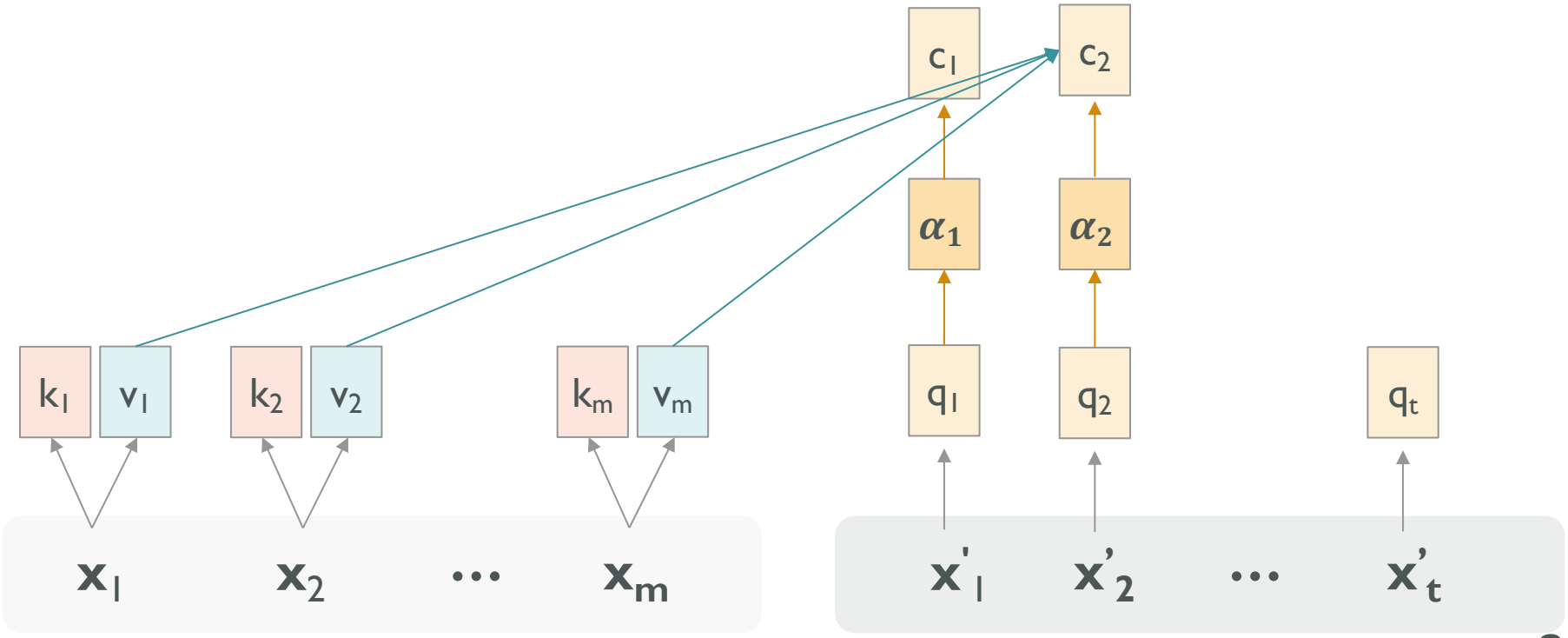


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention without RNN

Compute attention weights: $\alpha_2 = \text{softmax}(K^T q_2)$

Compute context vector: $c_2 = V^T \alpha_2$



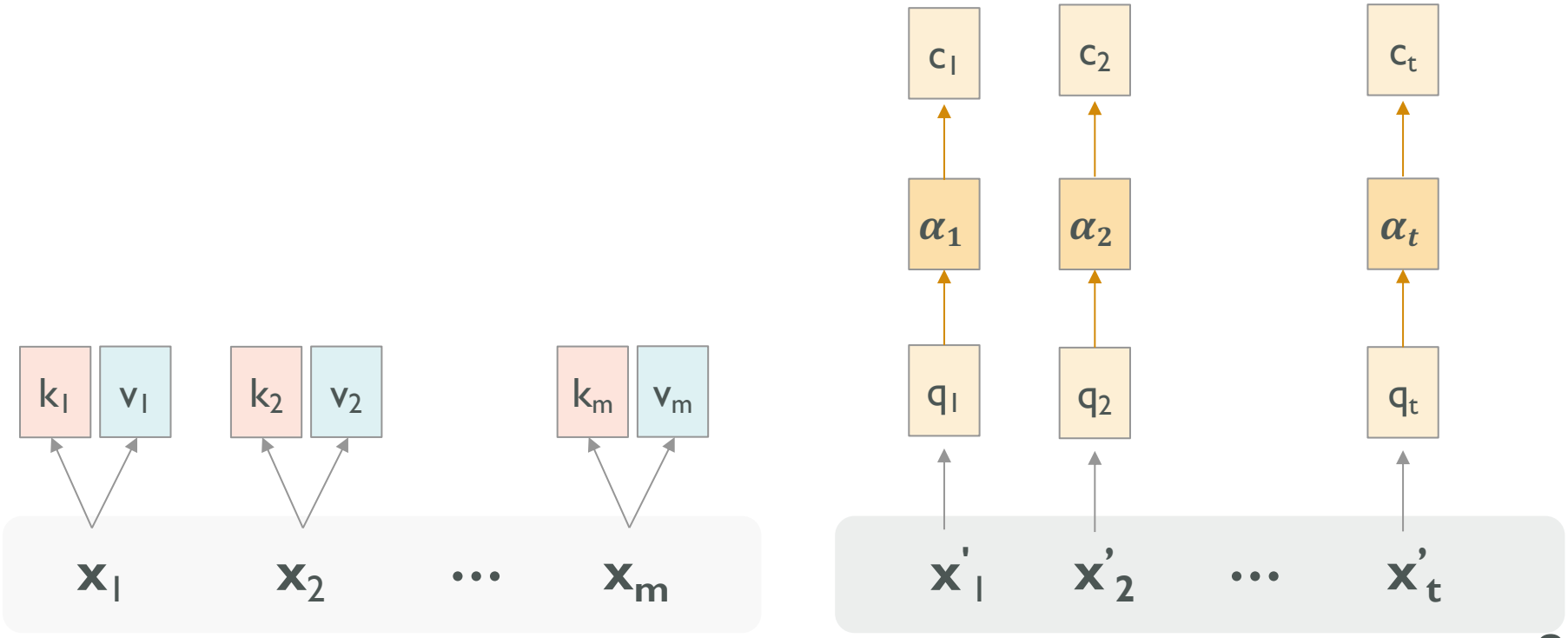
Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention layer

Compute attention weights: $\boldsymbol{\alpha_i}$ = softmax ($K^T q_i$)

Compute context vector: $\boldsymbol{c_i} = V^T \boldsymbol{\alpha_i}$

Output of attention: $\boldsymbol{C} = [\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_i}]$

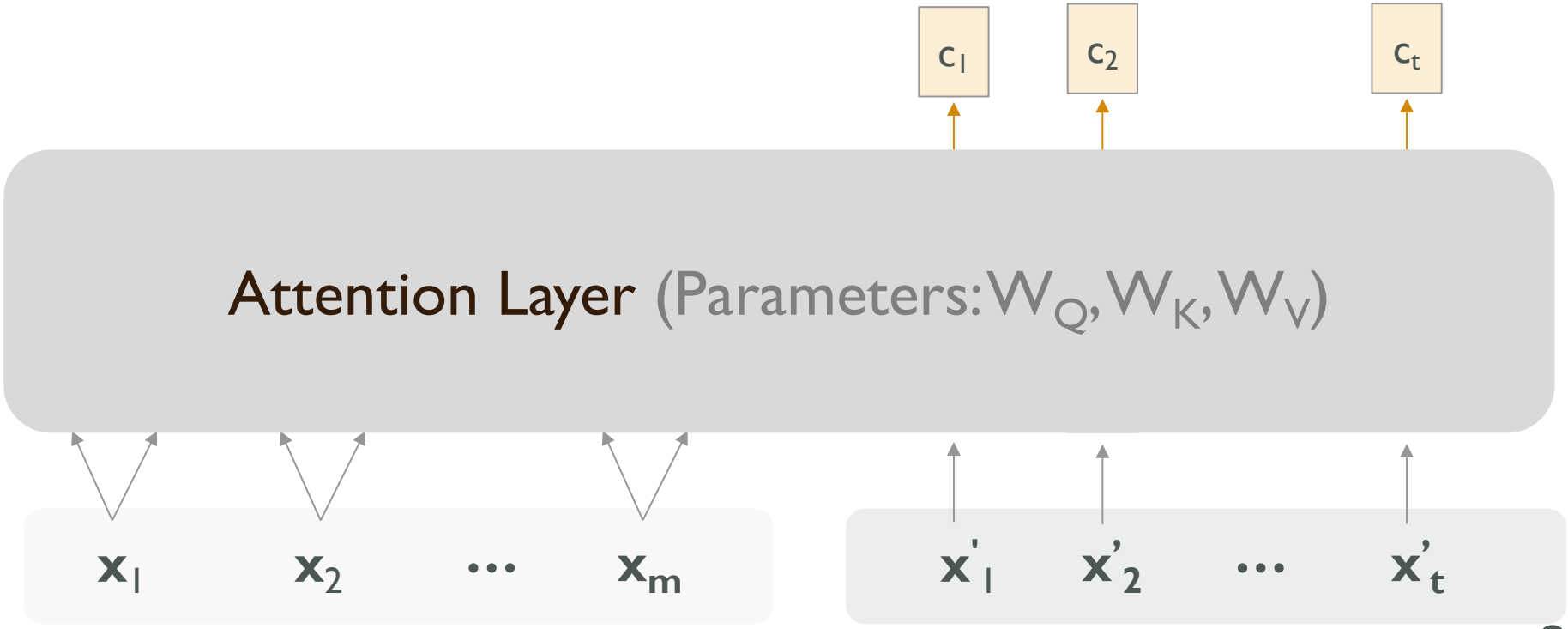


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Attention layer

Attention layer: $\boldsymbol{C} = attn(\boldsymbol{X}, \boldsymbol{X}')$

- Encoder's input: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{xm}]$
- Decoder's input: $\boldsymbol{X}' = [\boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_m]$

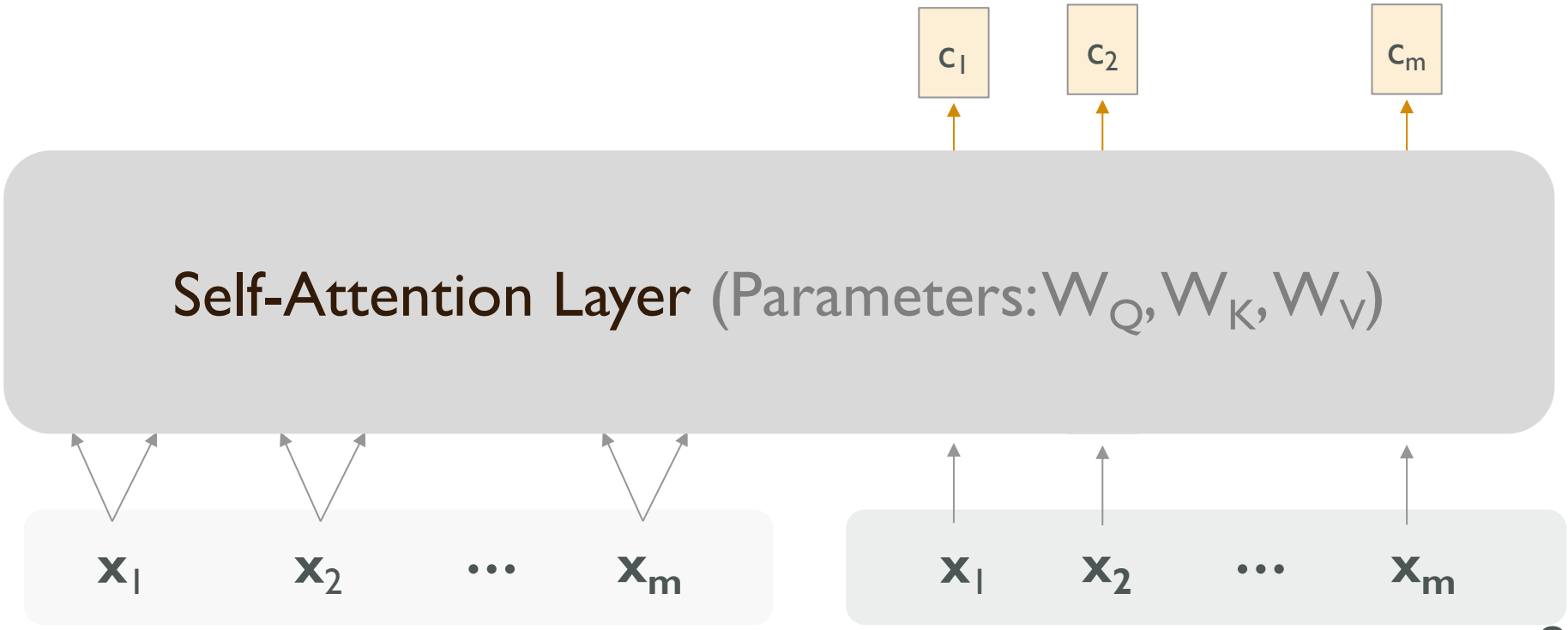


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Self-attention layer

Self-attention layer: $\boldsymbol{C} = attn(\boldsymbol{X}, \boldsymbol{X})$

- Input: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{xm}]$



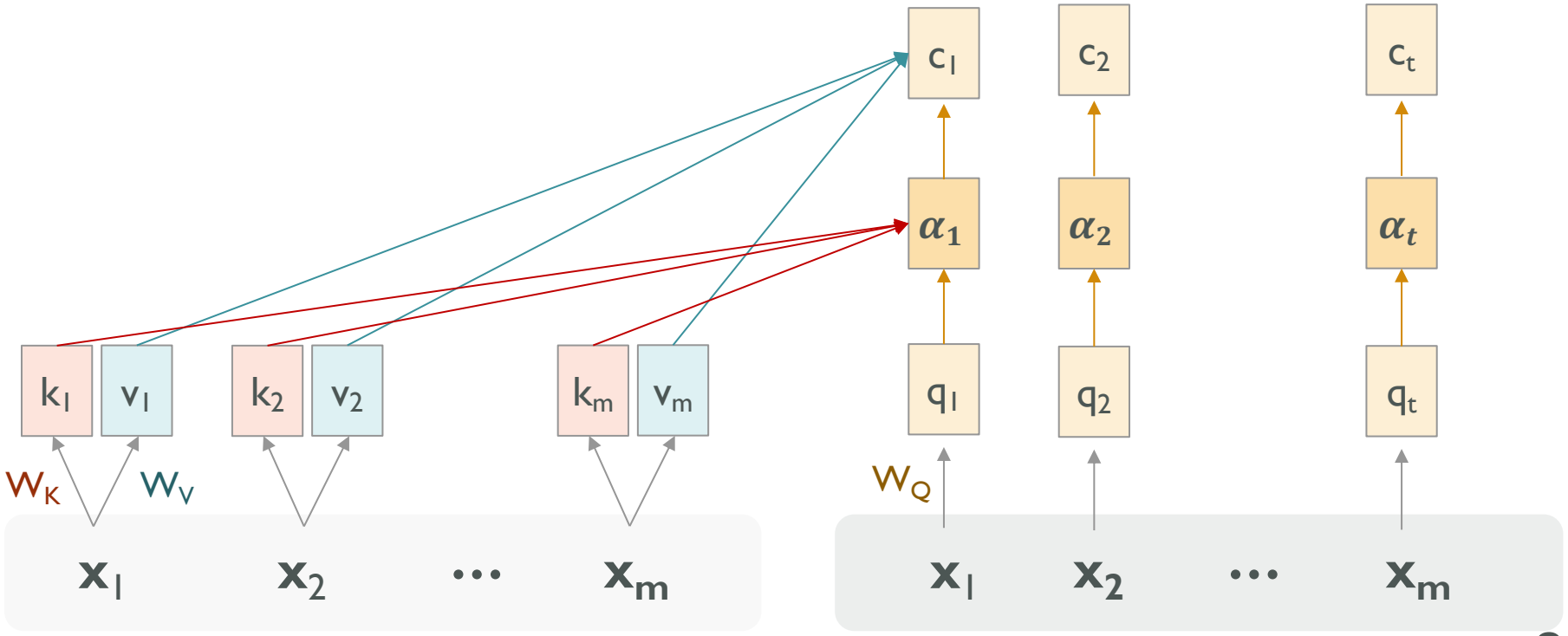
Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Self-attention layer

Compute attention weights: $\boldsymbol{\alpha_i}$ = softmax ($K^T q_i$)

Compute context vector: $\boldsymbol{c_i} = V^T \boldsymbol{\alpha_i}$

Output of attention: $\boldsymbol{C} = [\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_i}]$

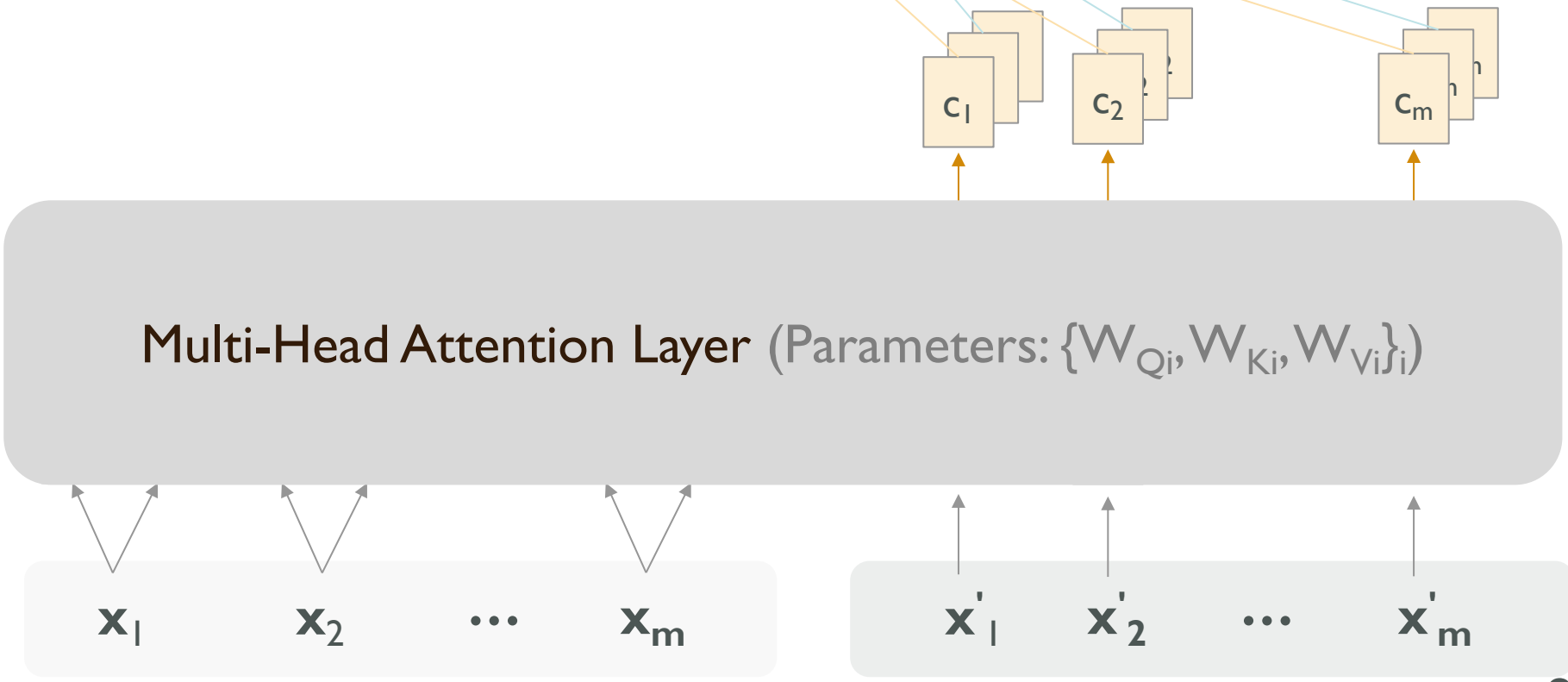


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Multi-head attention layer

Multi-head attention layer: $\boldsymbol{C^1, C^2, \ldots, Cl} = mha(\boldsymbol{X}, \boldsymbol{X})$

- Input: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{xm}]$

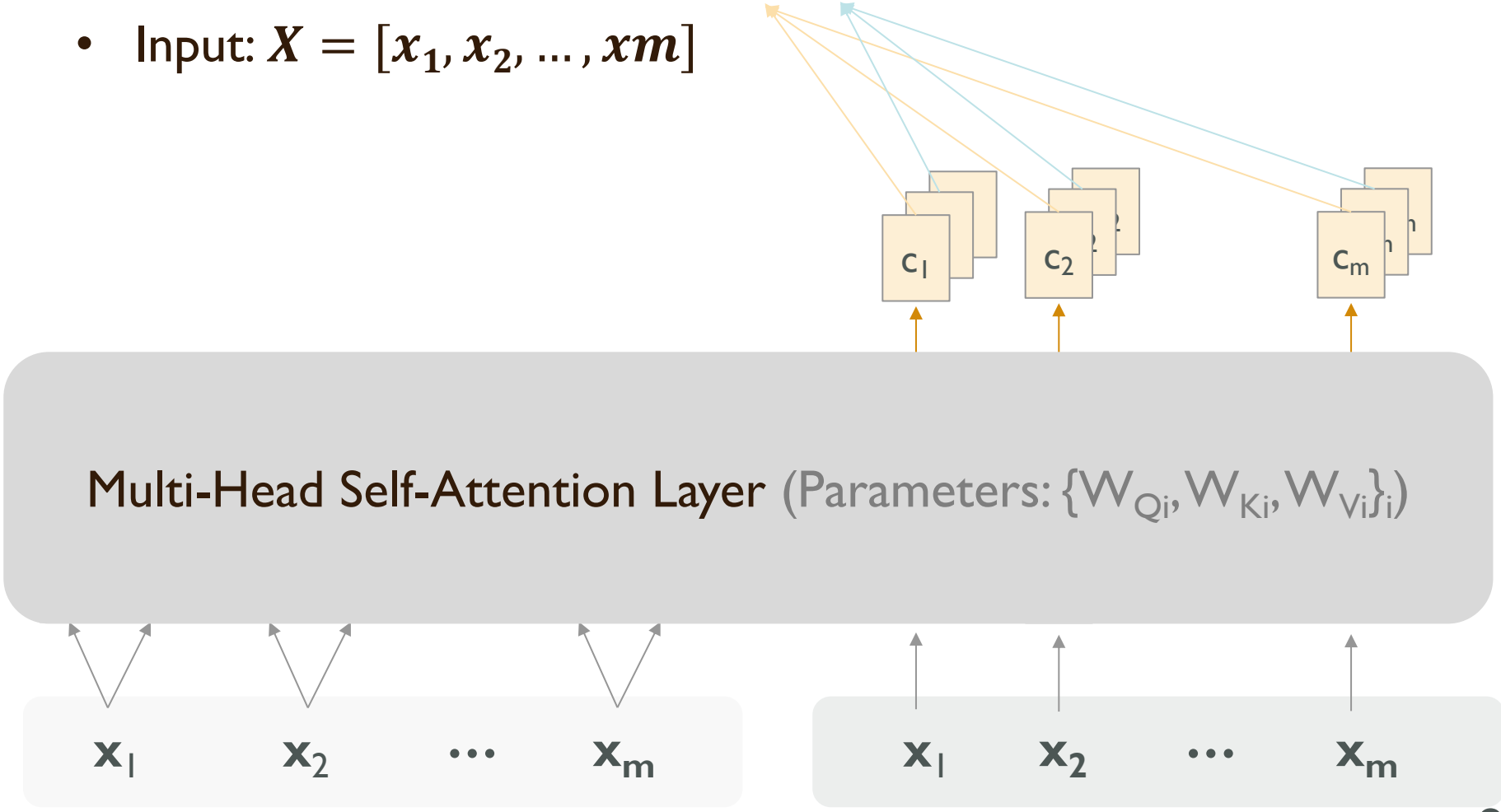


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Multi-head self-attention layer

Multi-head self-attention layer: $\boldsymbol{C^1, C^2, \ldots, Cl} = mhsa(\boldsymbol{X}, \boldsymbol{X})$

- Input: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{xm}]$

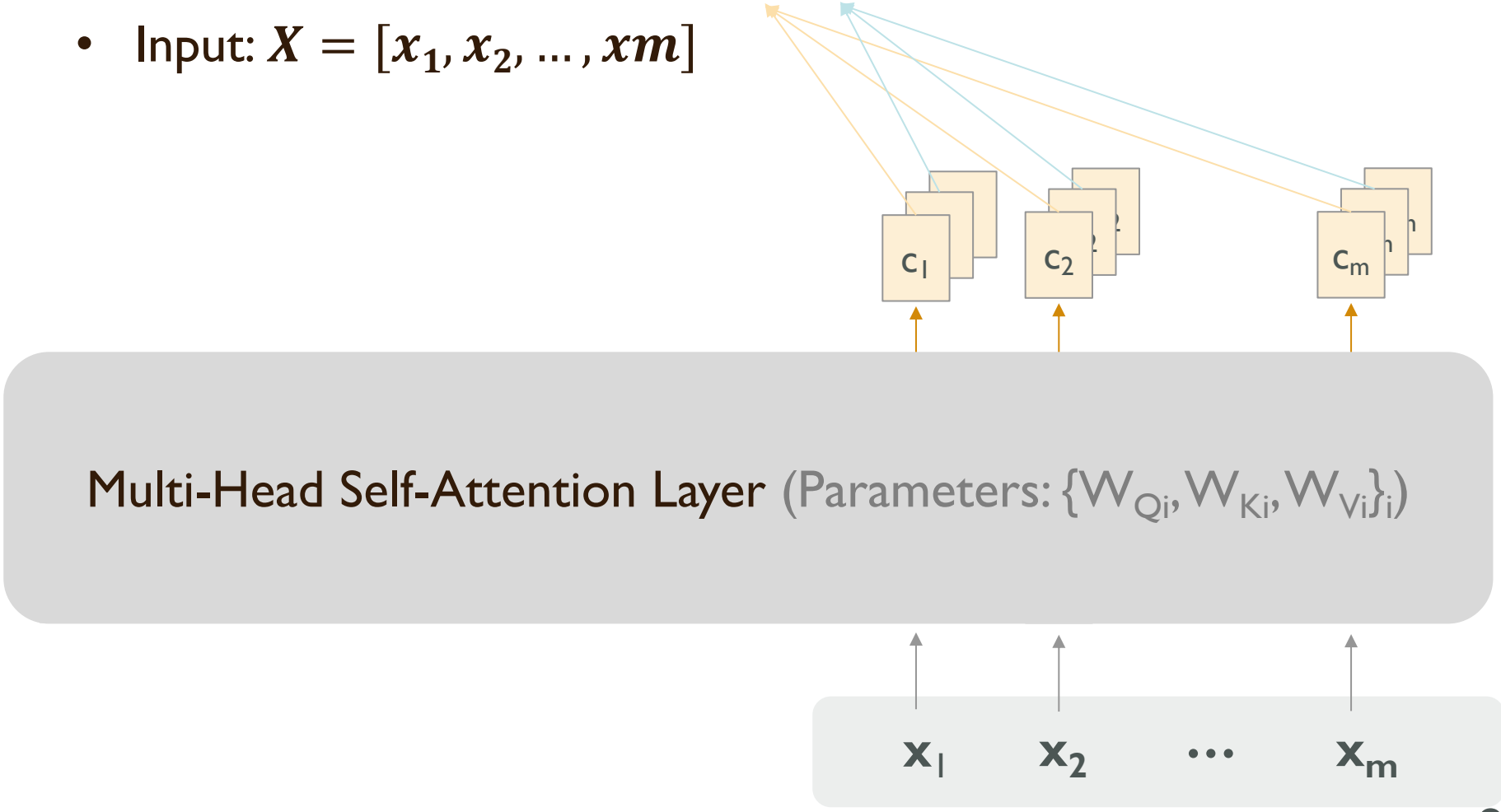


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Multi-head self-attention layer

Multi-head self-attention layer: $\boldsymbol{C^1, C^2, \ldots, Cl} = mhsa(\boldsymbol{X}, \boldsymbol{X})$

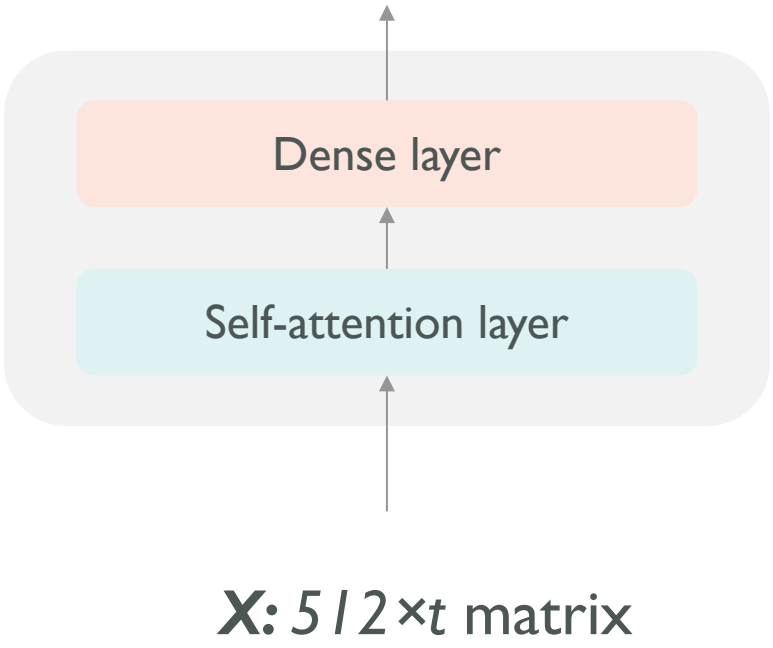
- Input: $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{xm}]$

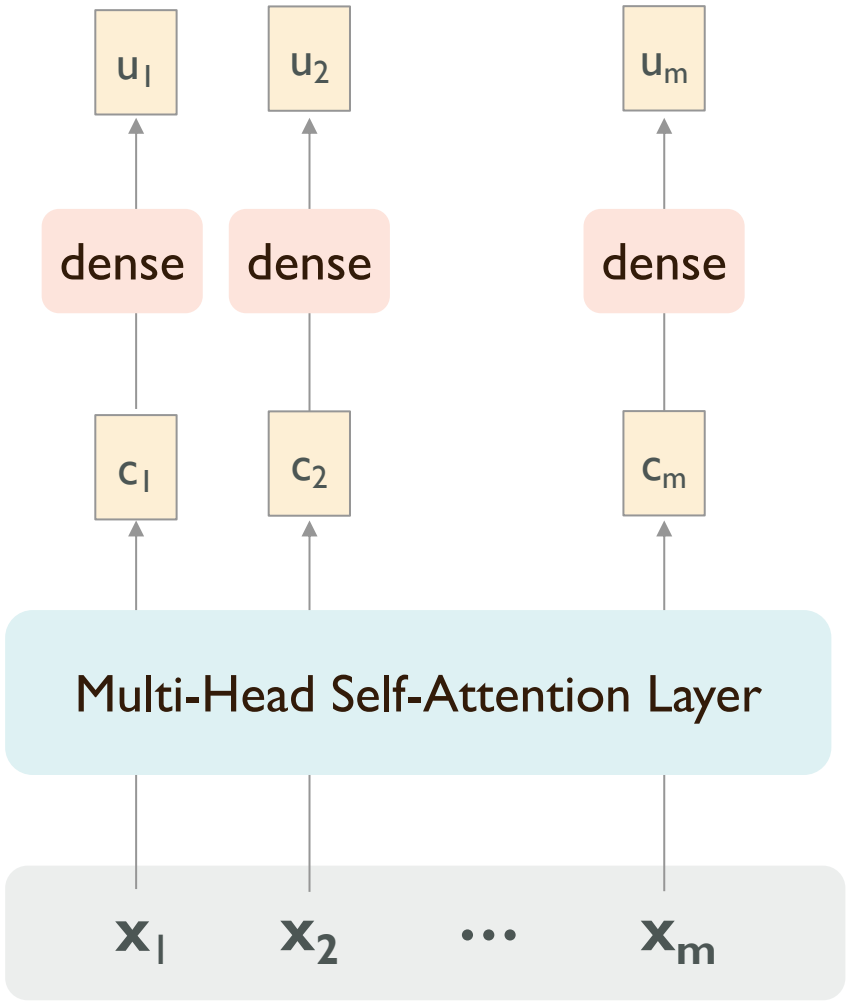


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer

One **block** of encoder



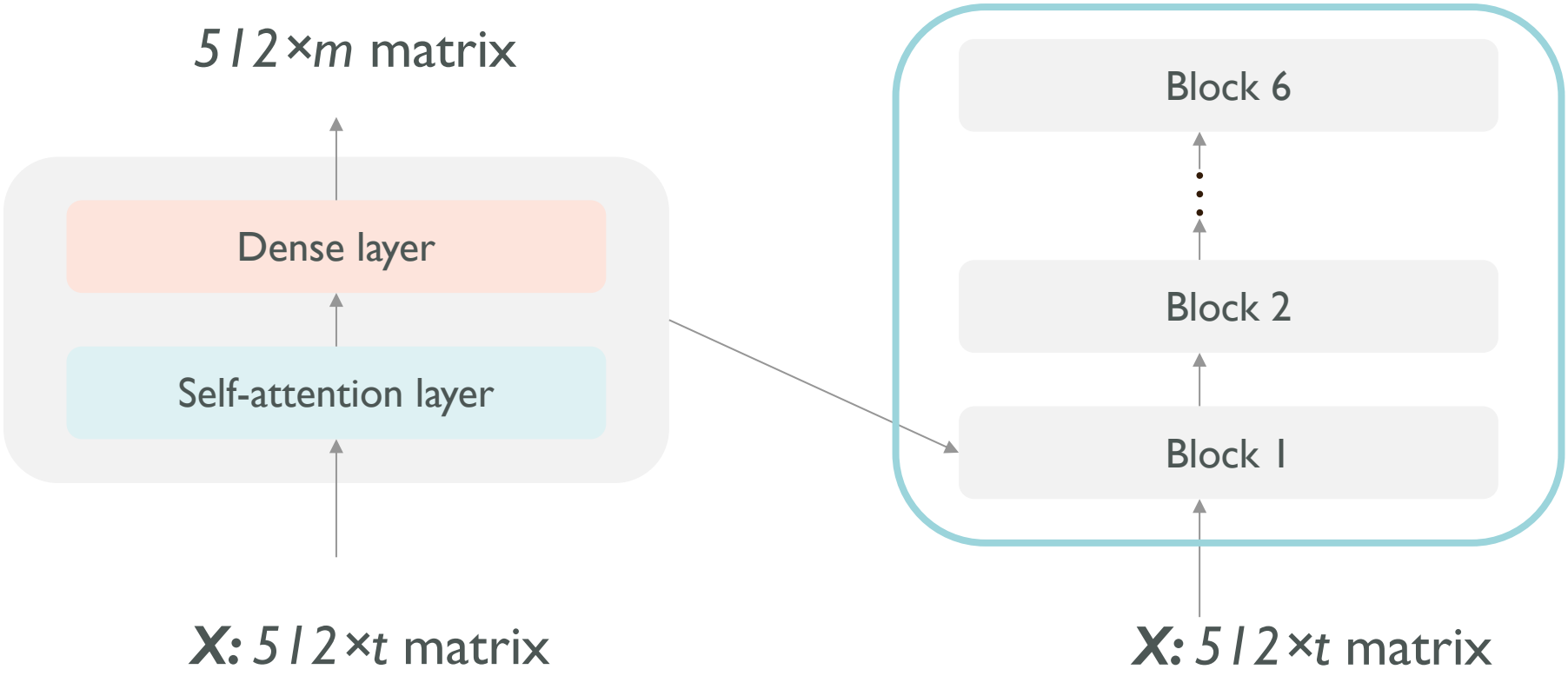


Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer

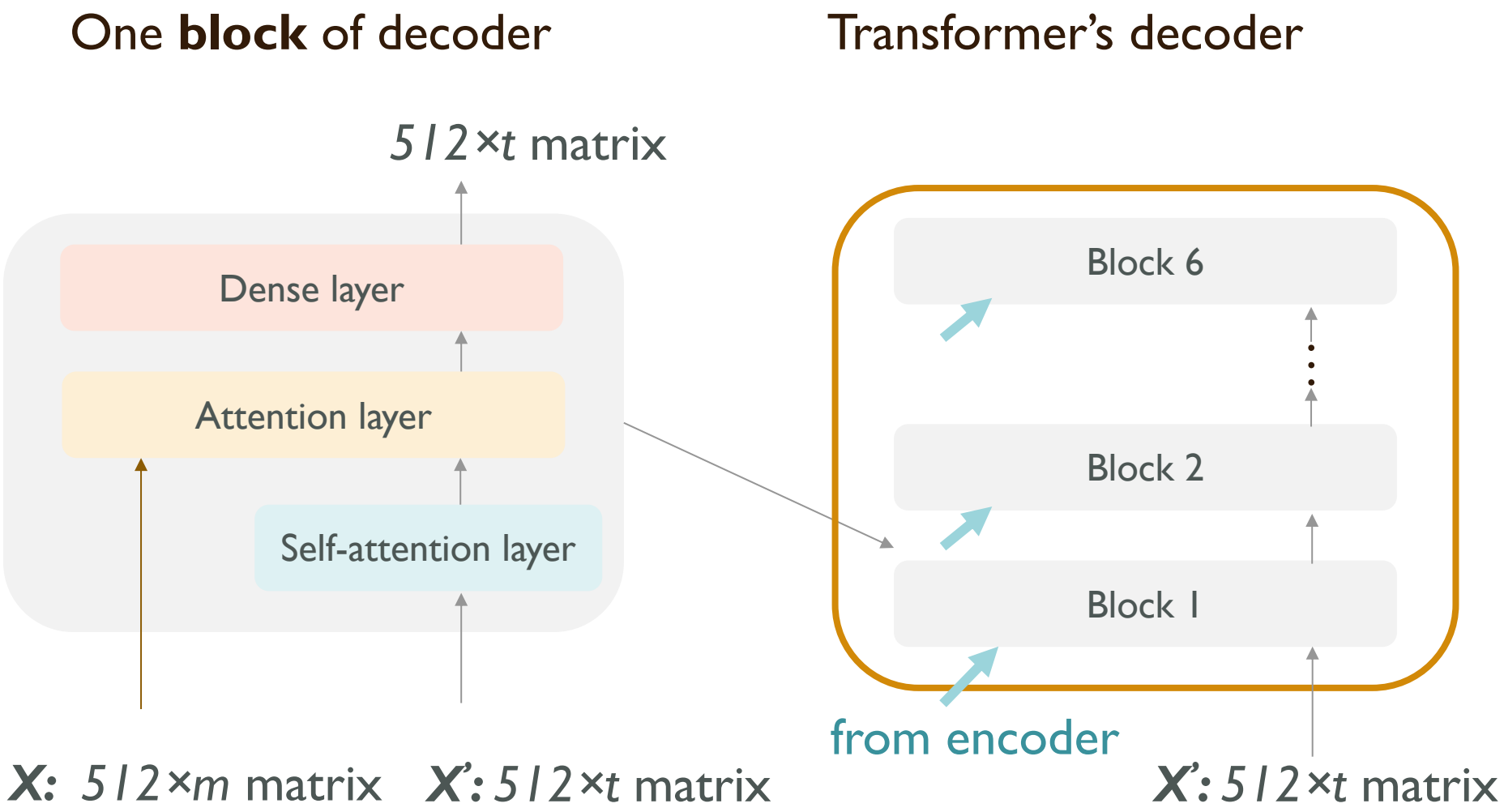
One **block** of encoder

Transformer's encoder



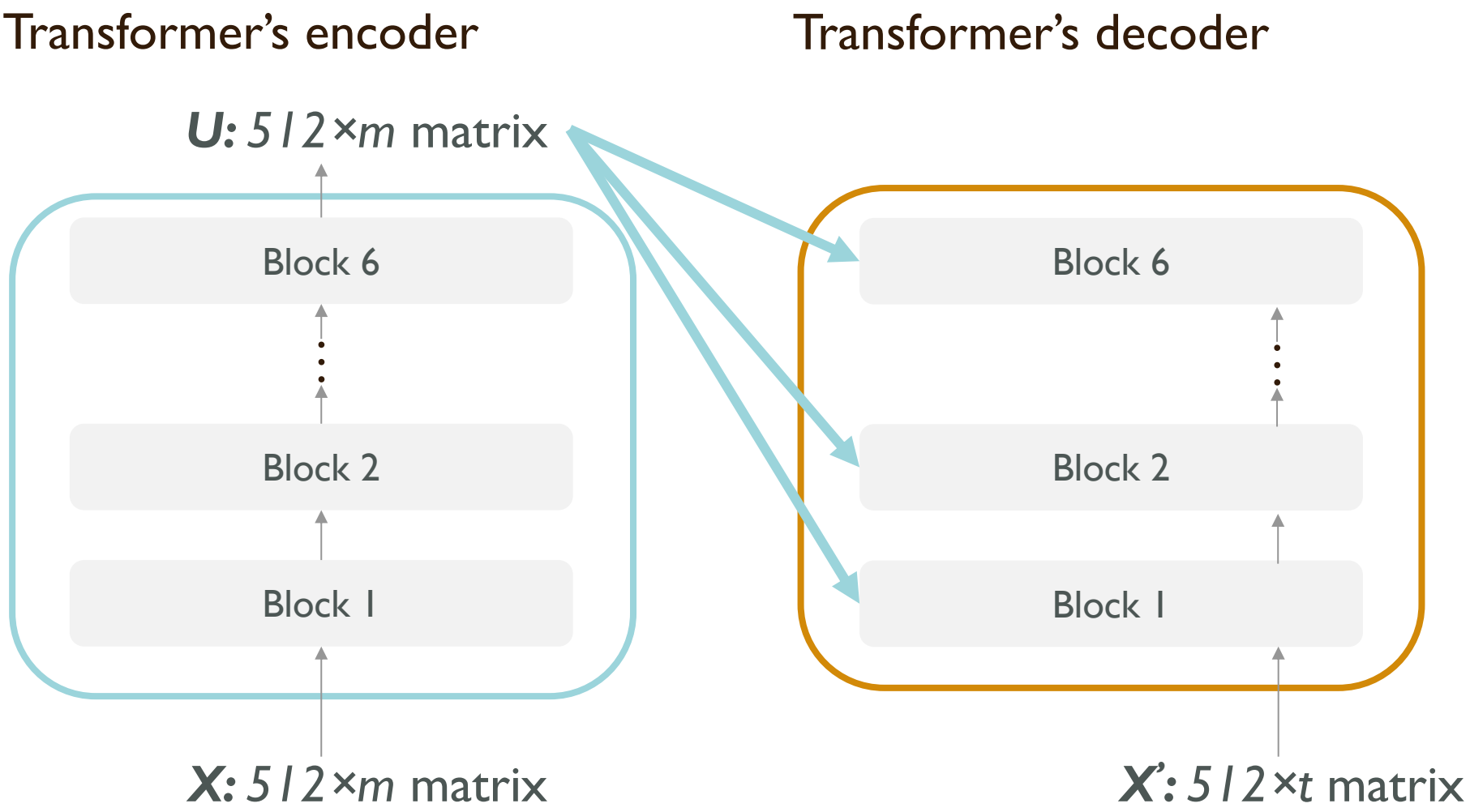
Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

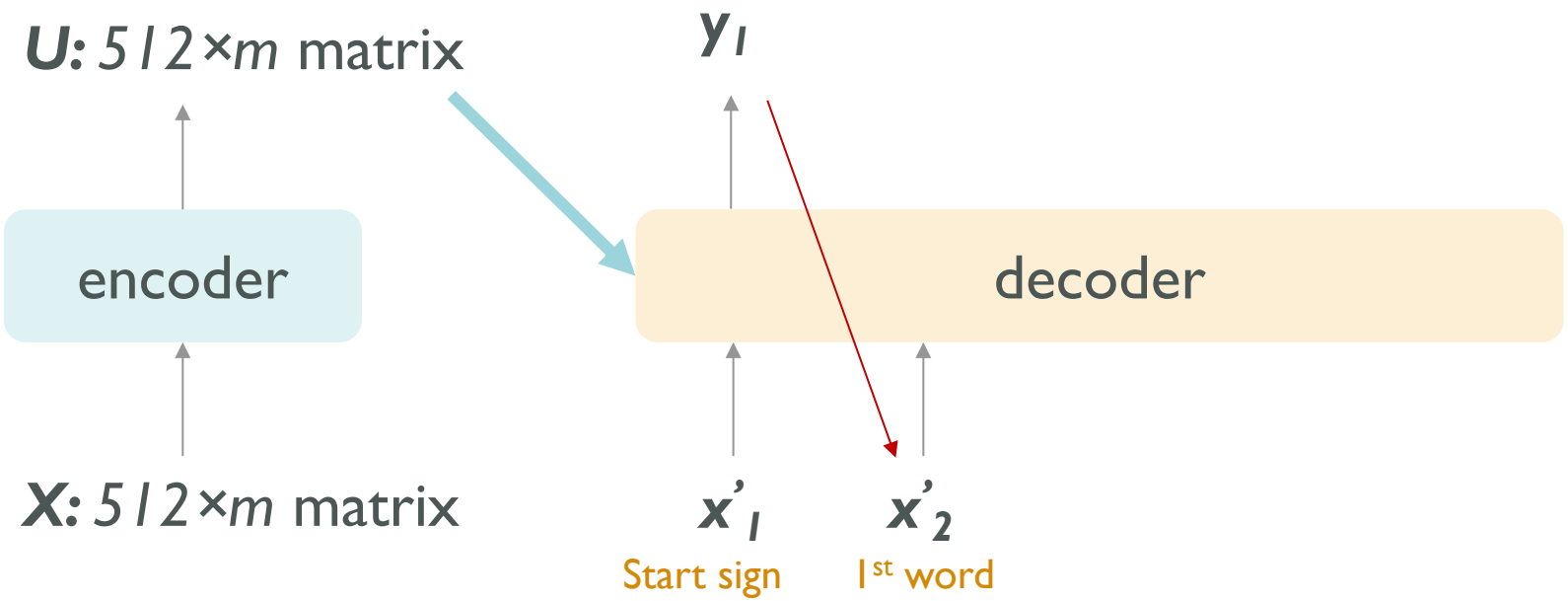
Transformer



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer

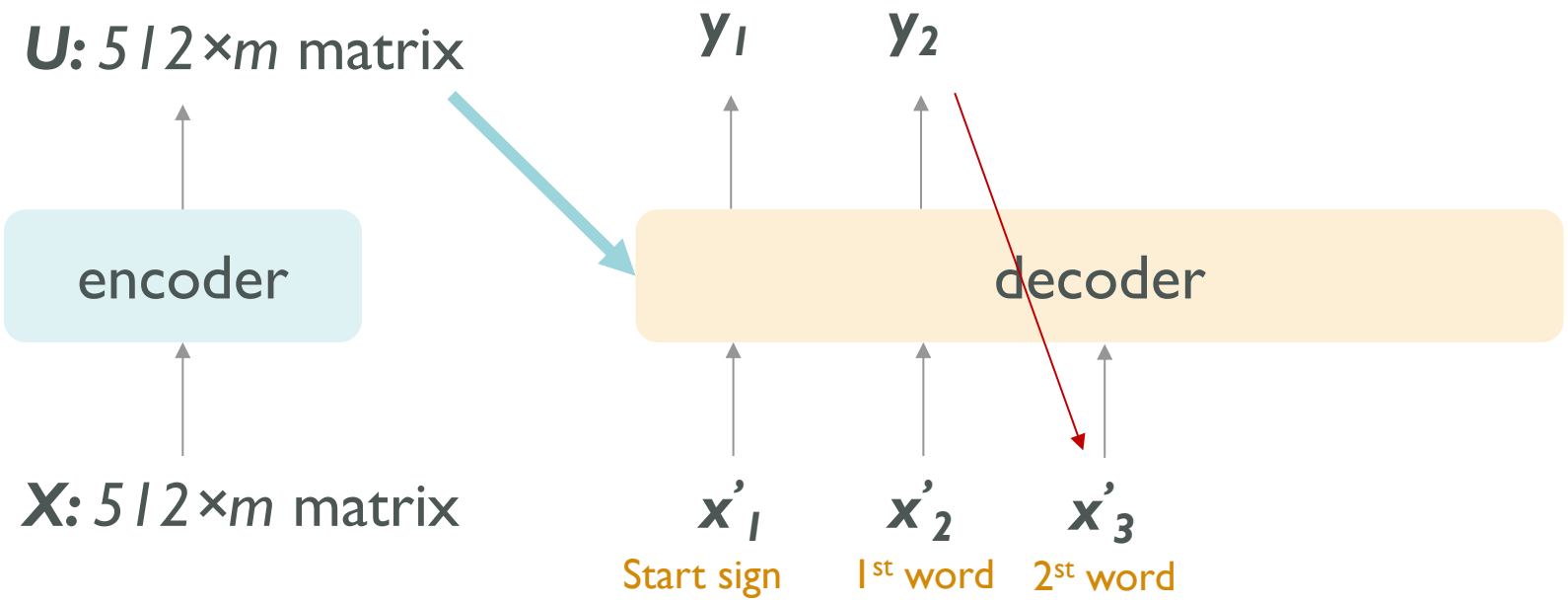
Example: Translation



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer

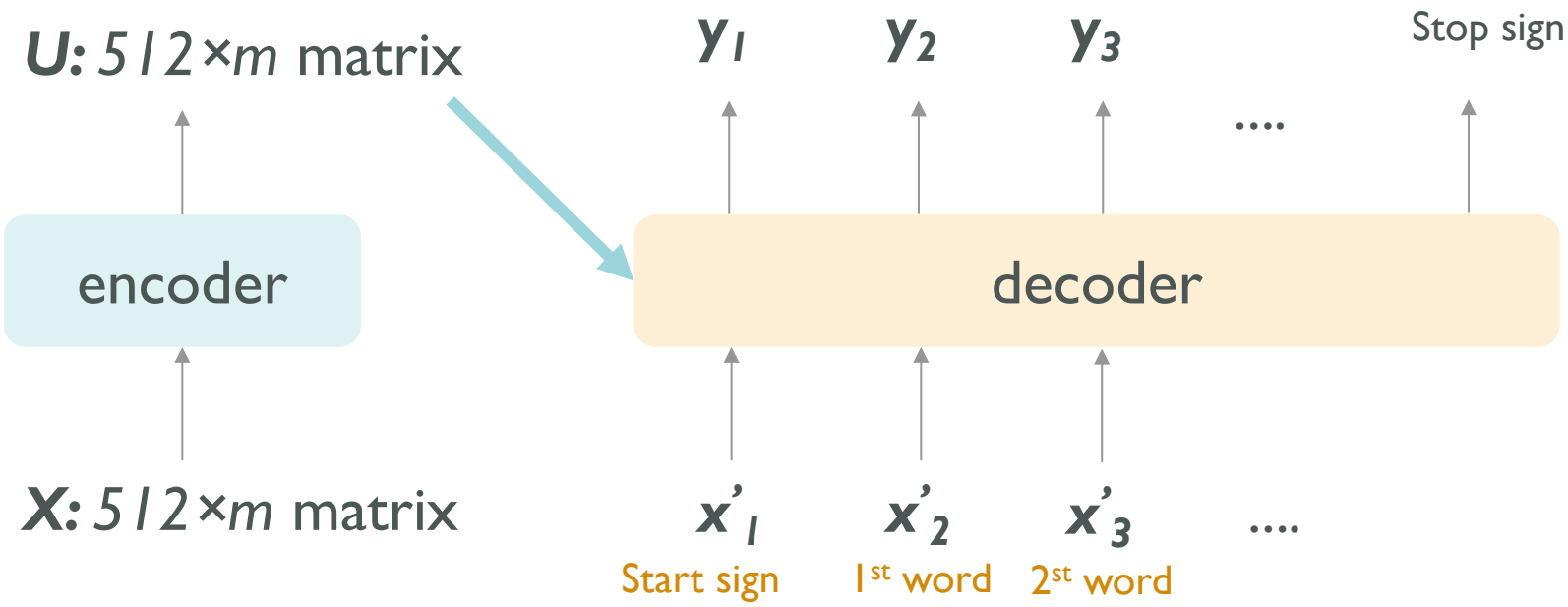
Example: Translation



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer

Example: Translation



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Transformer

- Transformer is Seq2Seq model; it has an encoder and a decoder.
- Transformer model is not RNN
- Transformer is based on attention and self-attention
- Transformer outperforms all the state-of-the-art RNN models.

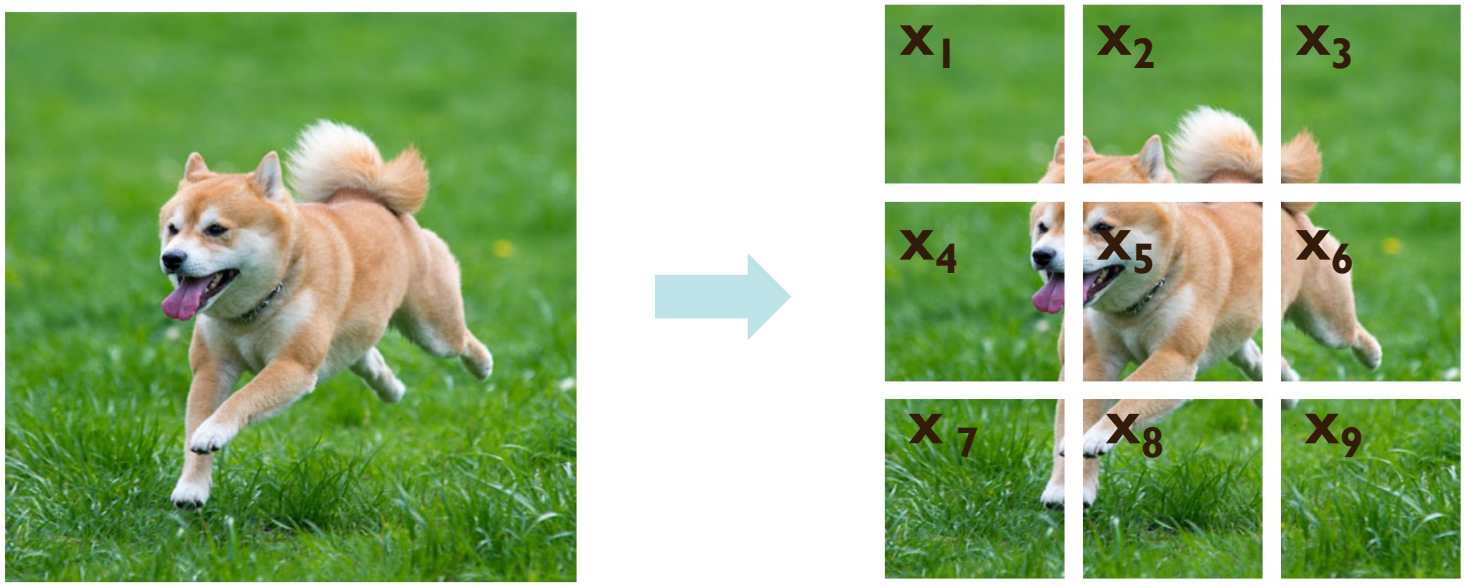
Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Vision Transformer (ViT)

- CNNs, e.g., ResNet, were the best solutions to image classification
- Vision Transformer (ViT) [1] beats CNNs (by a small margin), if the dataset for pretraining is sufficiently large (at least 100 million images)
- ViT is based on Transformer (for NLP)

Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

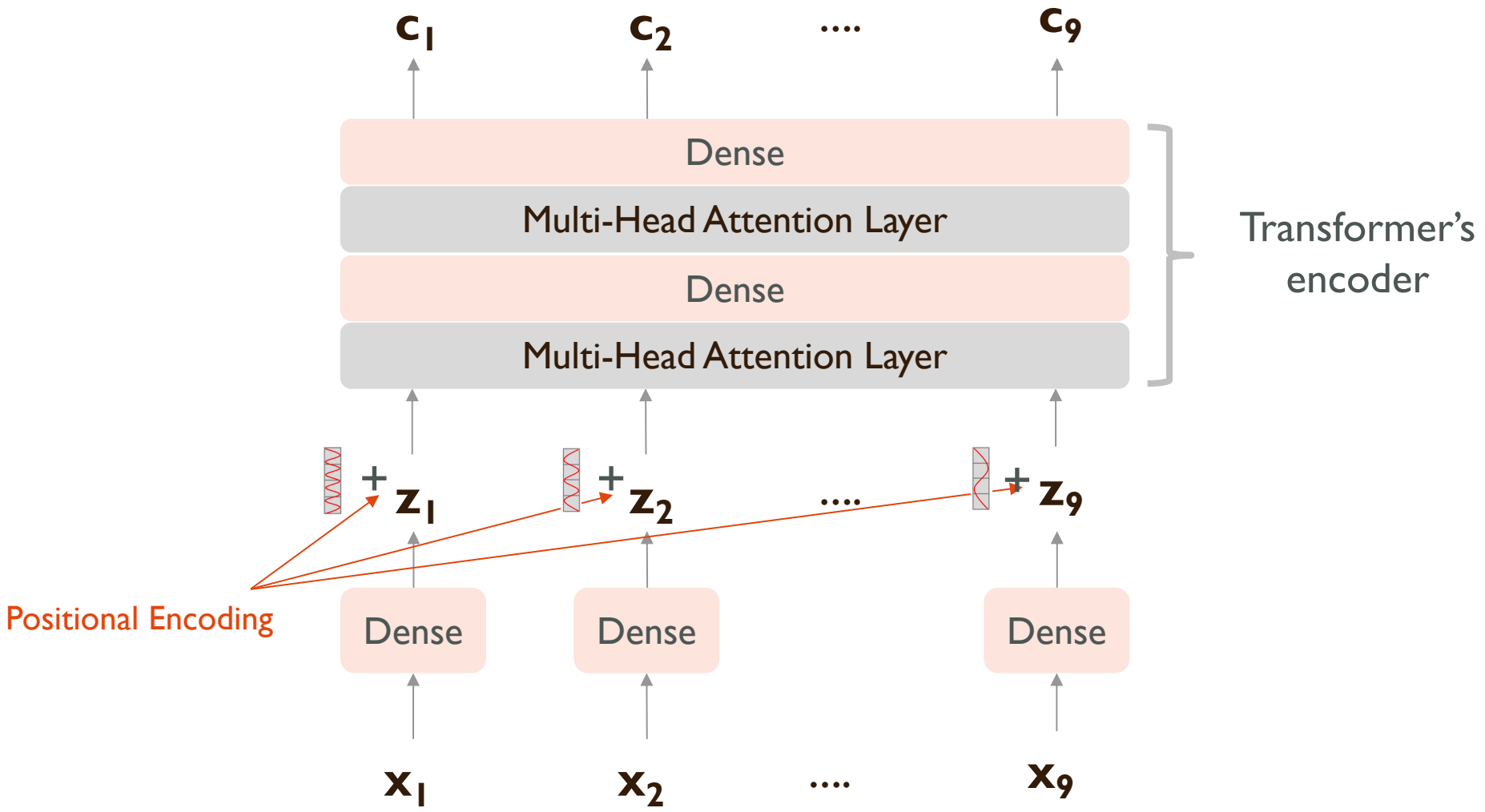
Vision Transformer (ViT)



- Represent each image as several patches
- Treat each patch as one word
- Use Transformer's encoder to embed each patch

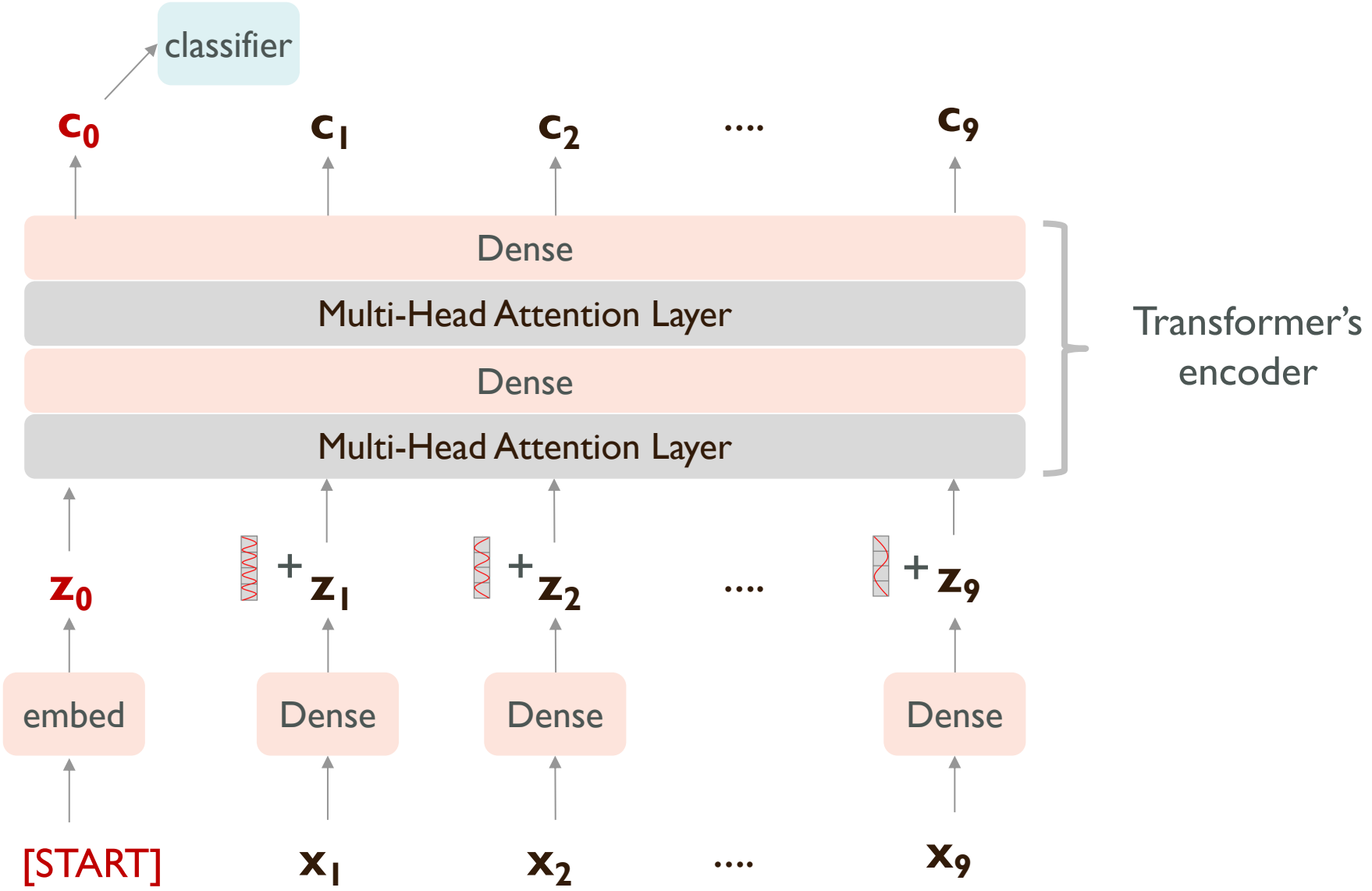
Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

Vision Transformer (ViT)



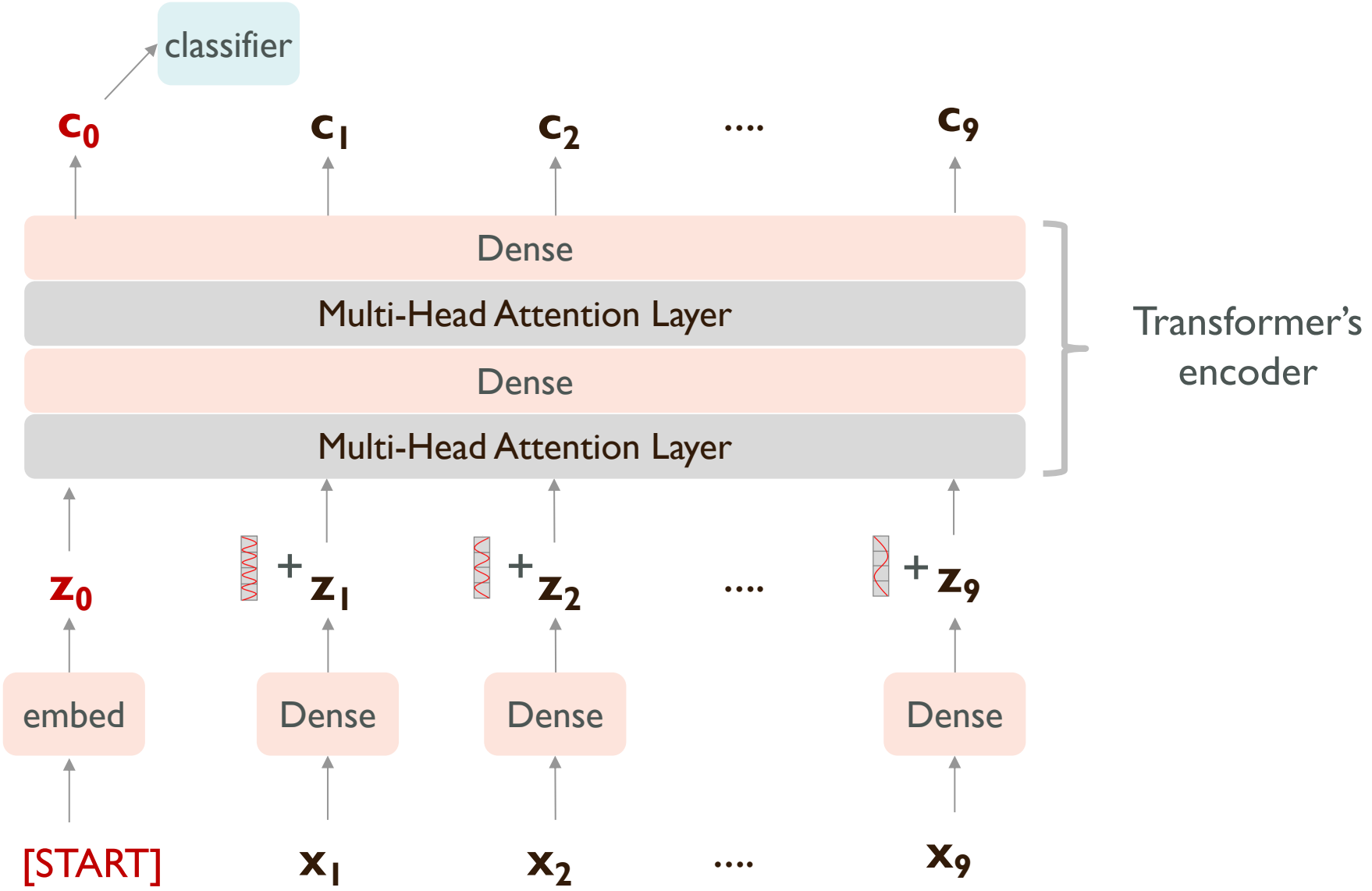
Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

Vision Transformer (ViT)



Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

Vision Transformer (ViT)



Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

Vision Transformer (ViT)

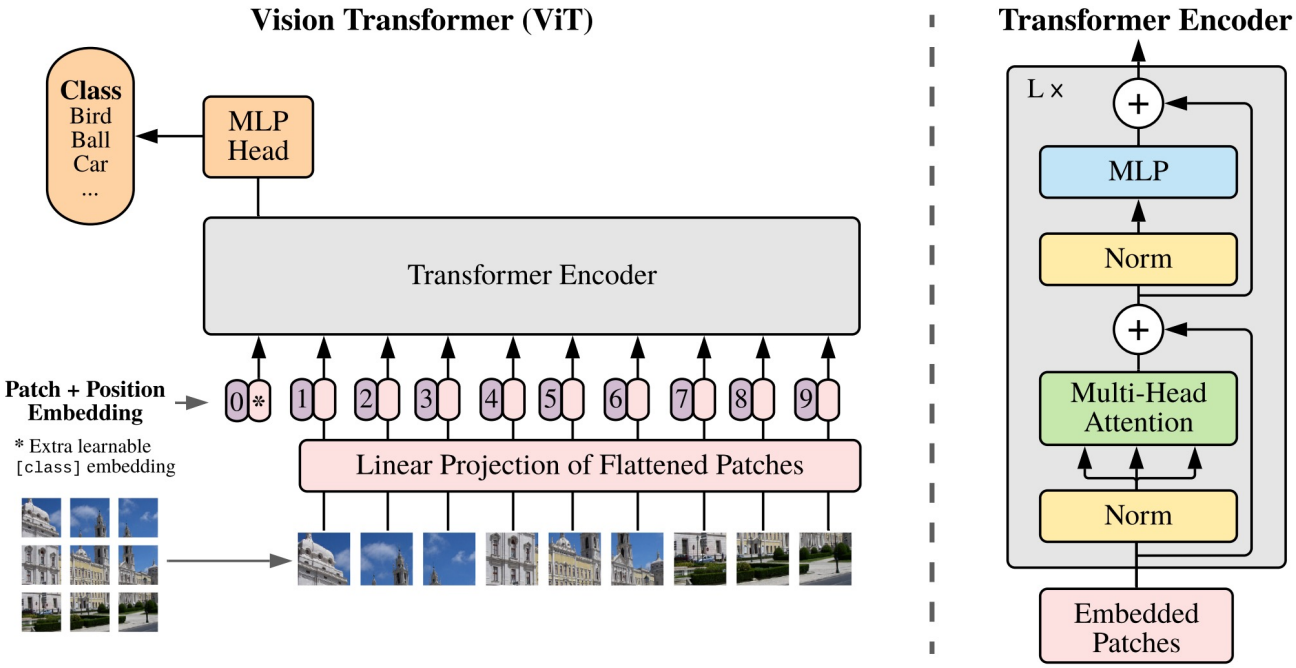
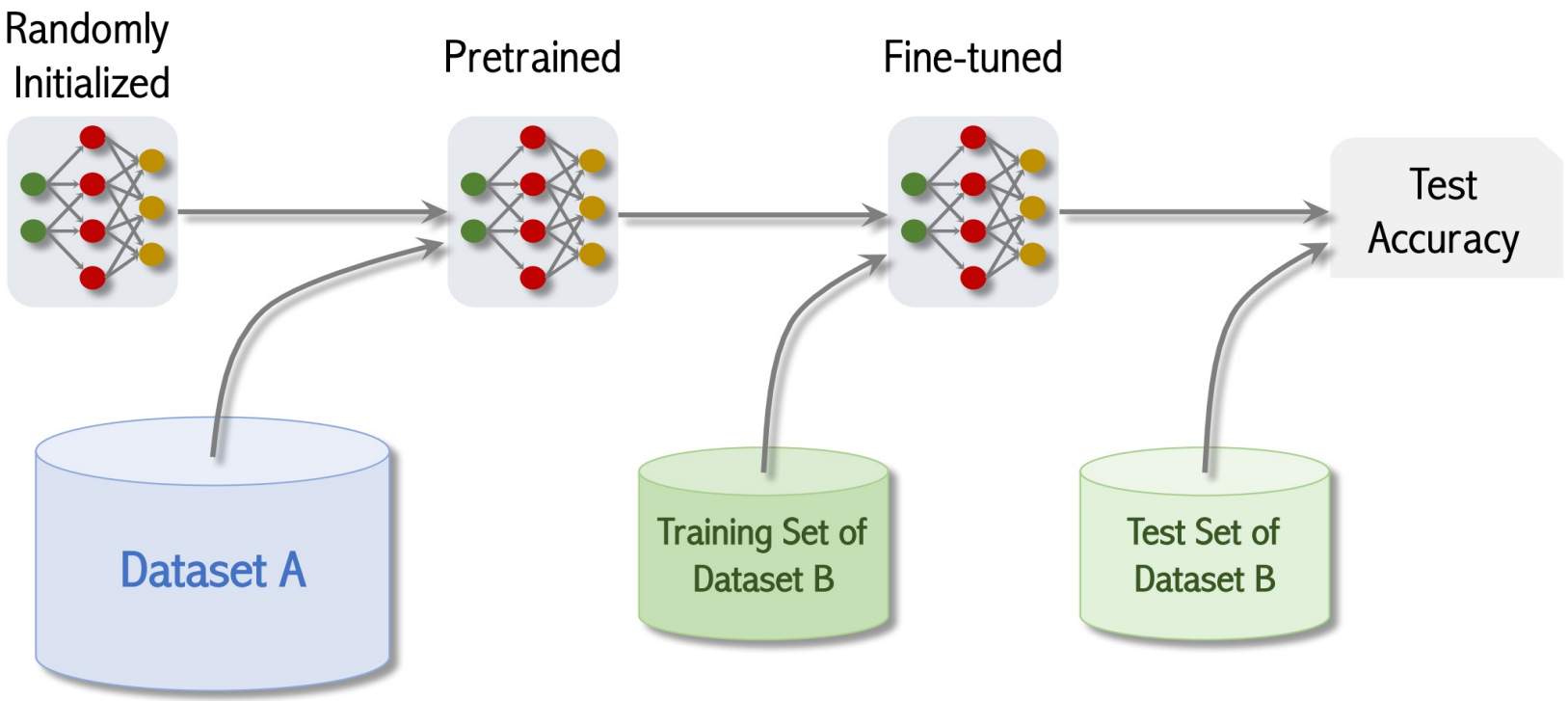


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

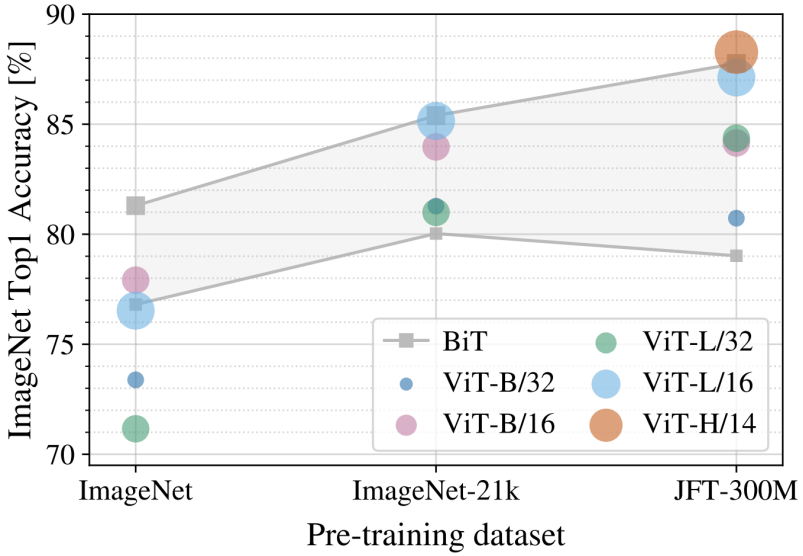
Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

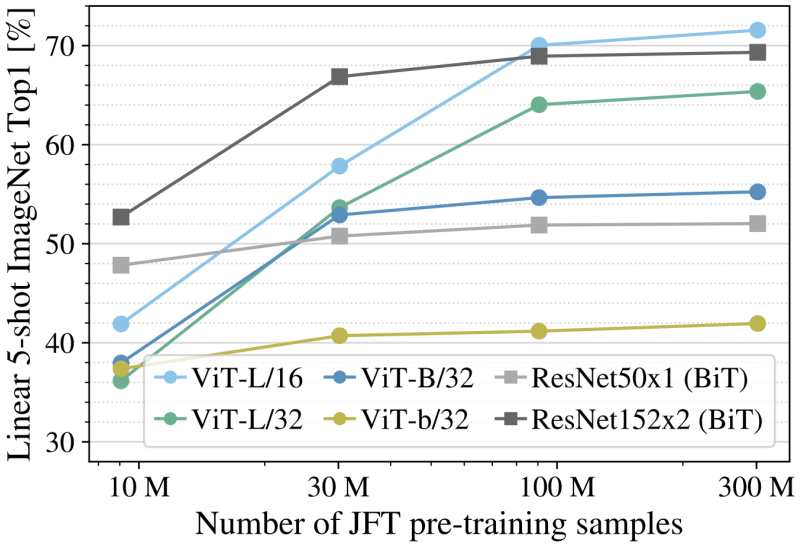
Vision Transformer (ViT)



Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

Vision Transformer (ViT)





- Pretrained on ImageNet (small), ViT is slightly worse than ResNet
- Pretrained on ImageNet-21K (medium), ViT is comparable to ResNet
- Pretrained on JFT (large), ViT is slightly better than ResNet.

Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

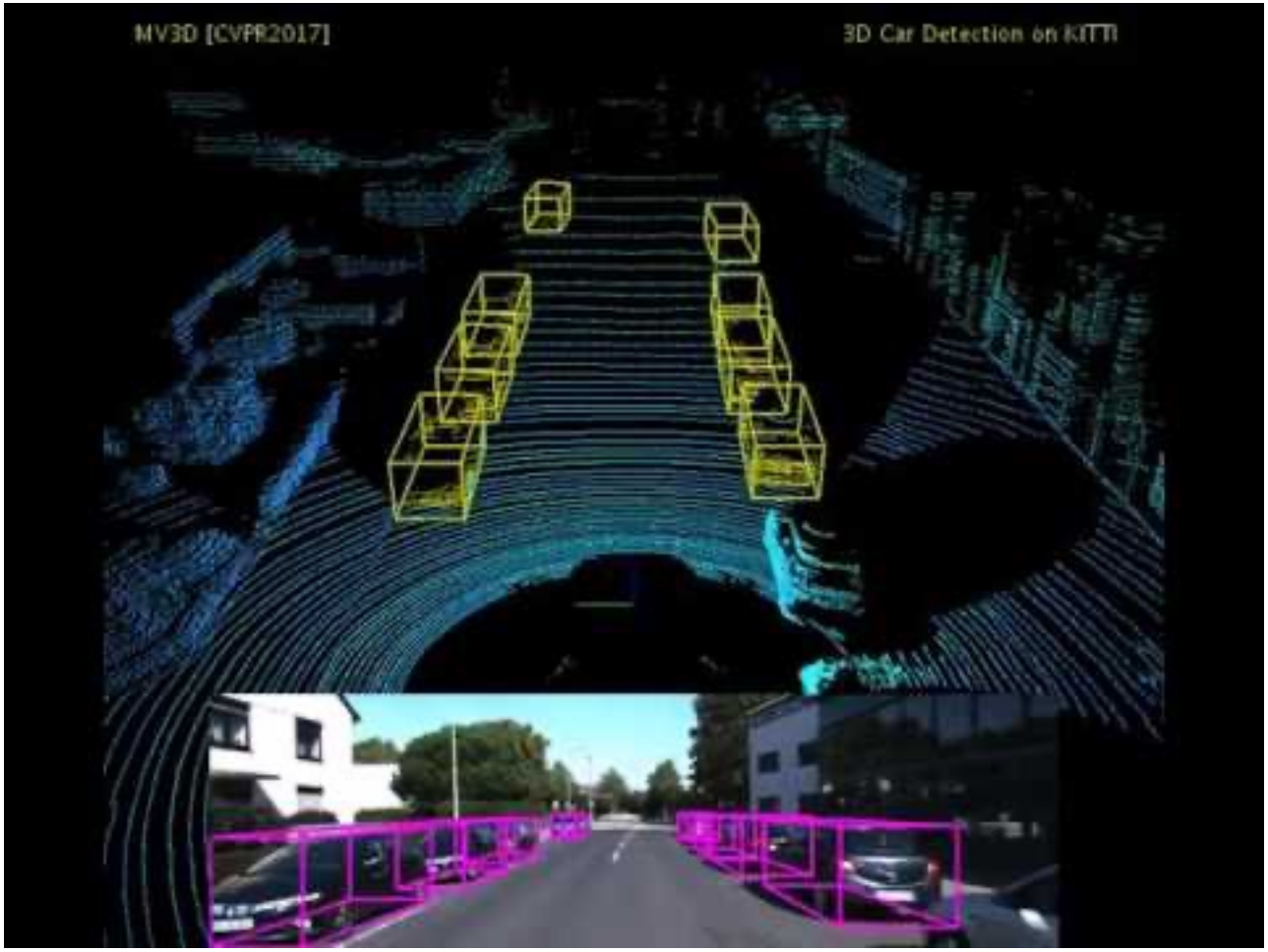
Vision Transformer (ViT)

Pretrain the model on Dataset A, fine-tune the model on Dataset B, and evaluate the model on Dataset B.

	# of Images	# of Classes
ImageNet (Small)	1.3 Million	1 Thousand
ImageNet-21K (Medium)	14 Million	21 Thousand
JFT (Big)	300 Million	18 Thousand

Dosovitskiy et al. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR, 2021.

Next lecture: Object detection



Thank you very much!

sihengc@sjtu.edu.cn