# Transformer

Mengxue

# 1 Transformer

## 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

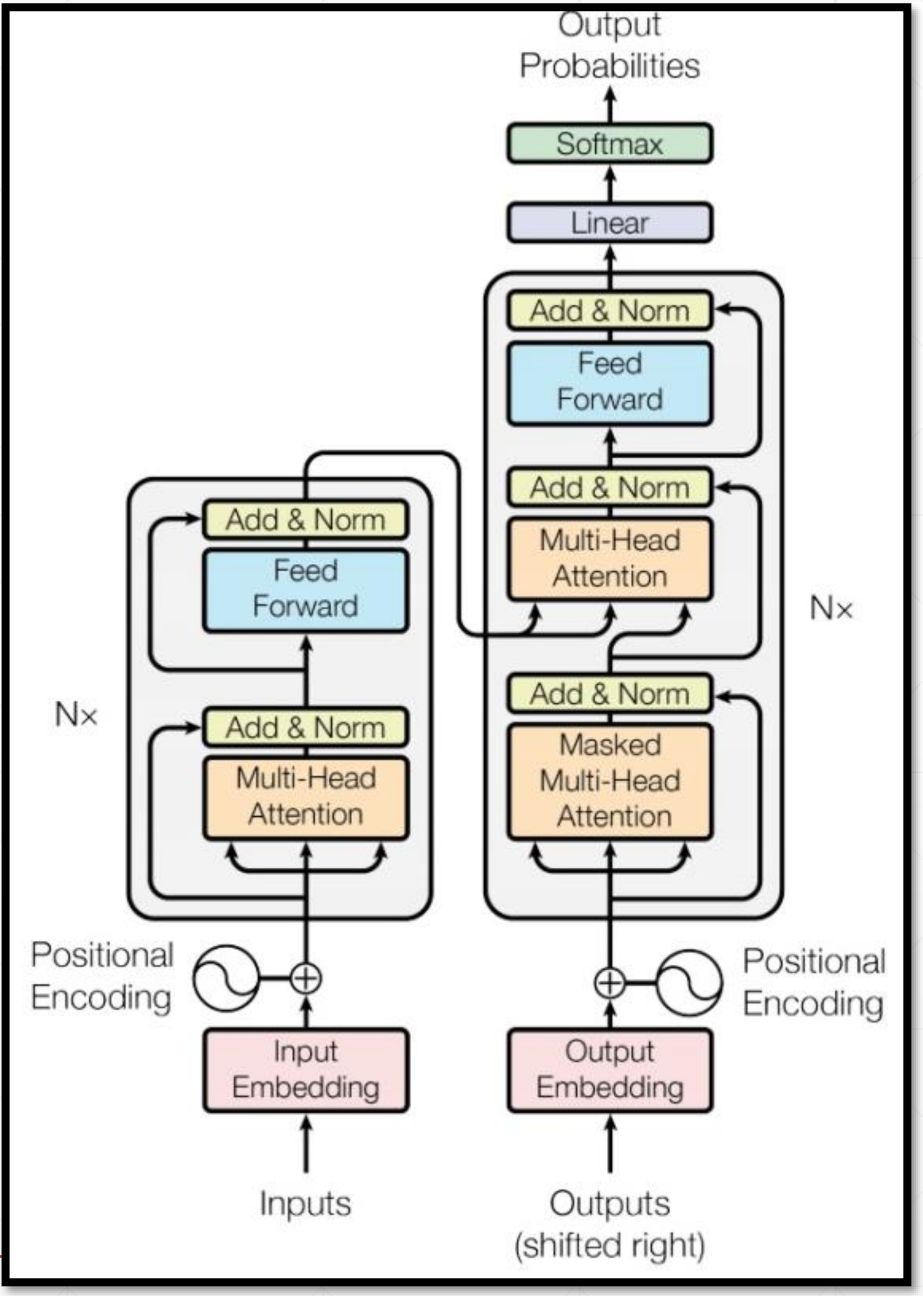
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu

**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com

**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

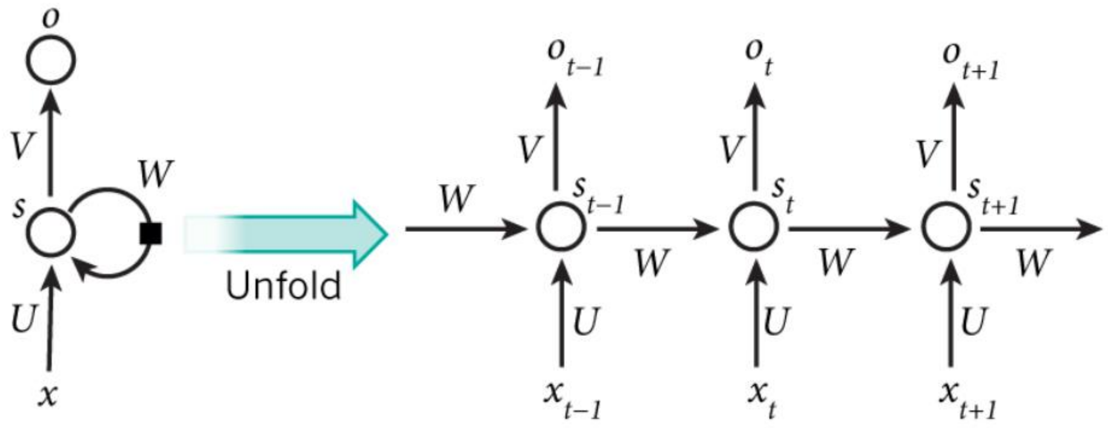


**Neural Information Processing Systems (NIPS 2017)**

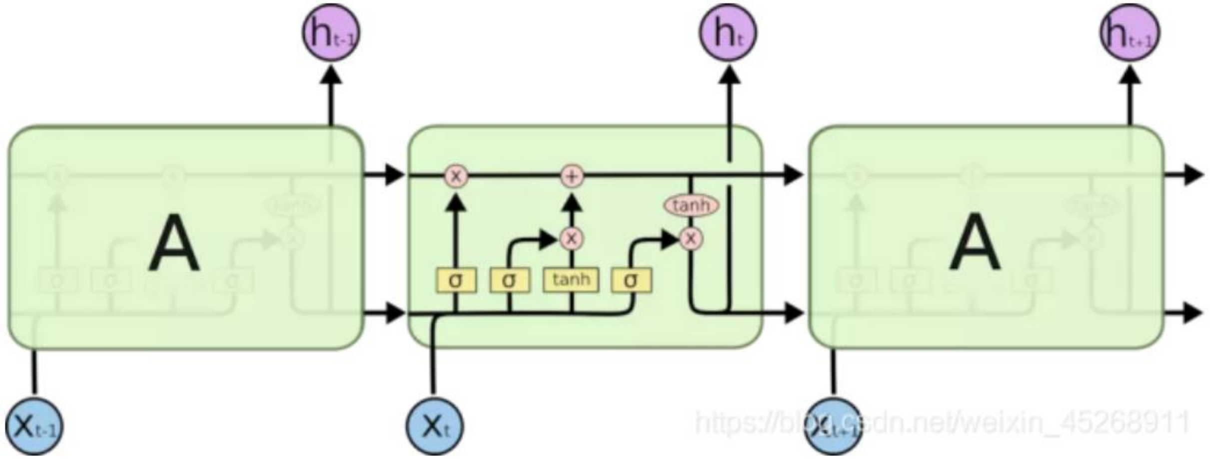



# RNN&LSTM

## →Long-Term Dependencies




RNN




LSTM

# Transformer →

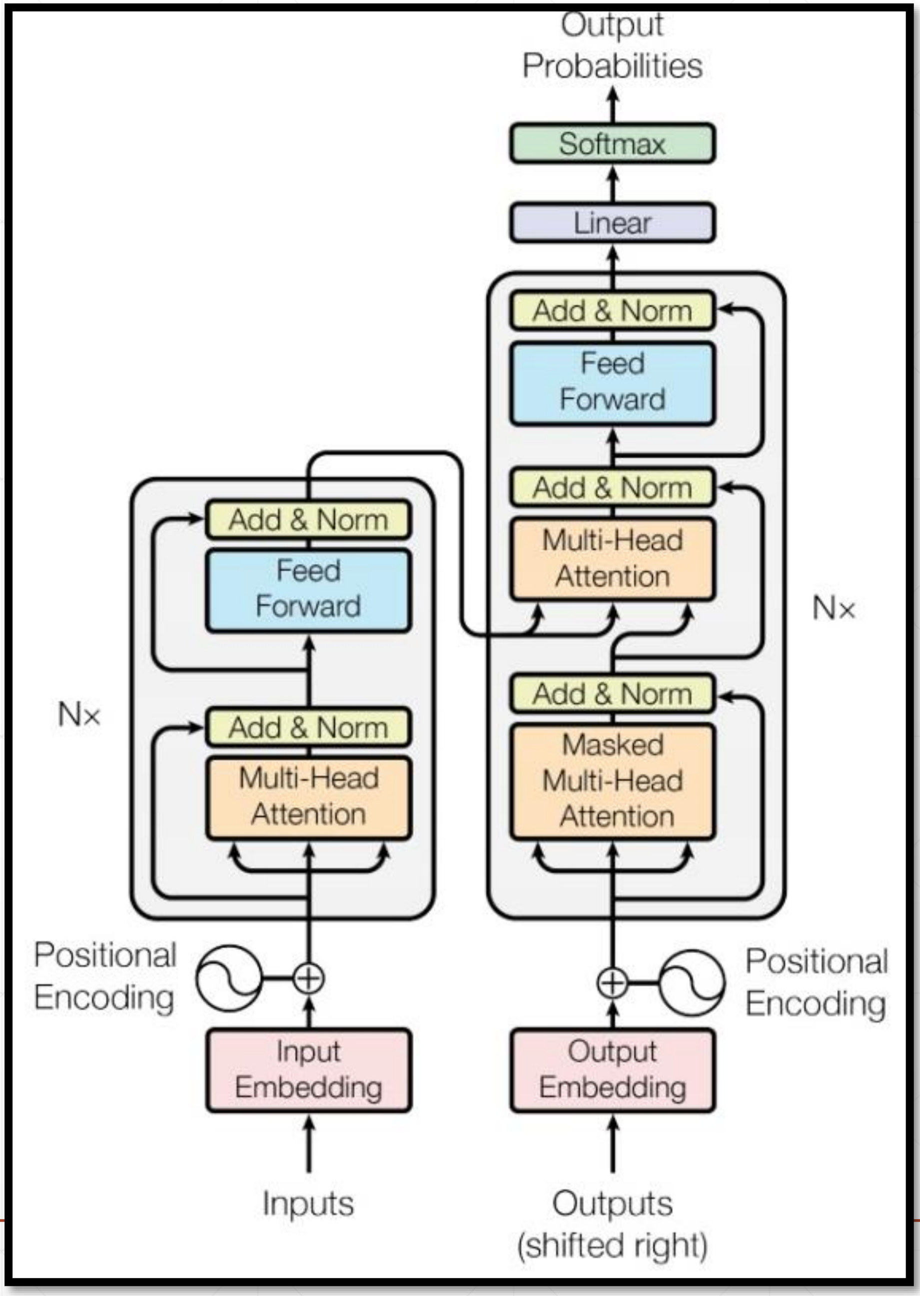
1 可并行计算（RNN只能顺序传播）

2 避免长程依赖的问题（用position embedding，可以快速找到远处）

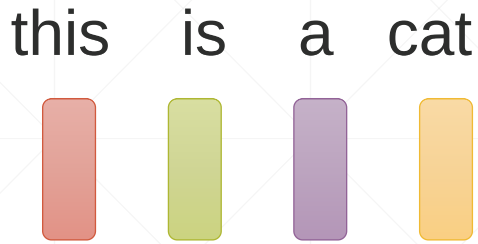
3 自注意力可以产生更具可解释性的模型。我们可以从模型中检查注意力分布。各个注意头 (attention head) 可以学会执行不同的任务

# Tips

- Self-attention
- Multihead Self-attention
- Position encoding
- Masked Multi-Head Attention
- Cross Attention



# Vector Set as Input




***One-hot Encoding***

apple = [ 1 0 0 0 0 …… ]

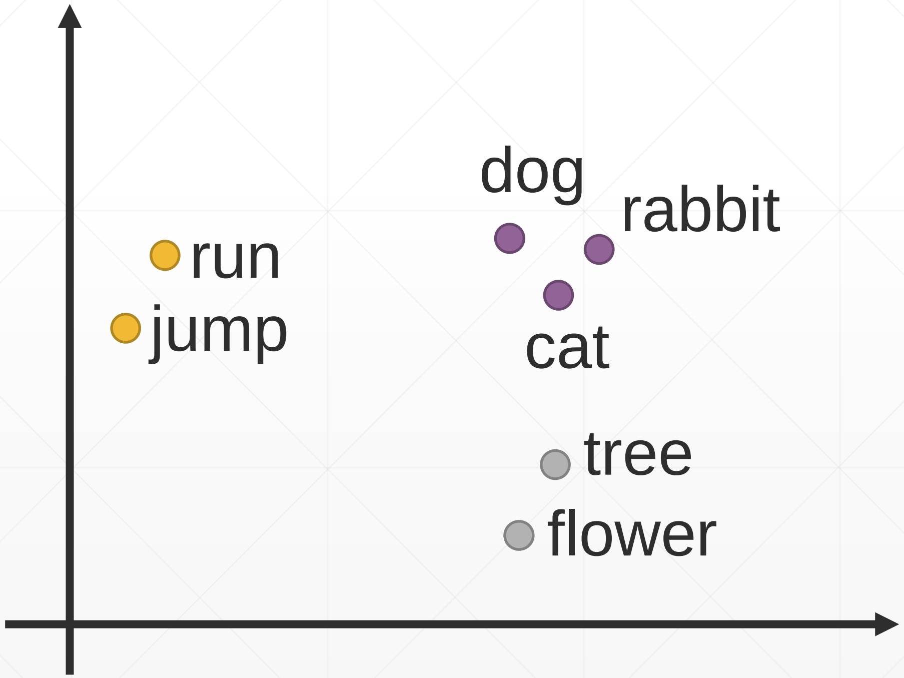
bag = [ 0 1 0 0 0 …… ]

cat = [ 0 0 1 0 0 …… ]

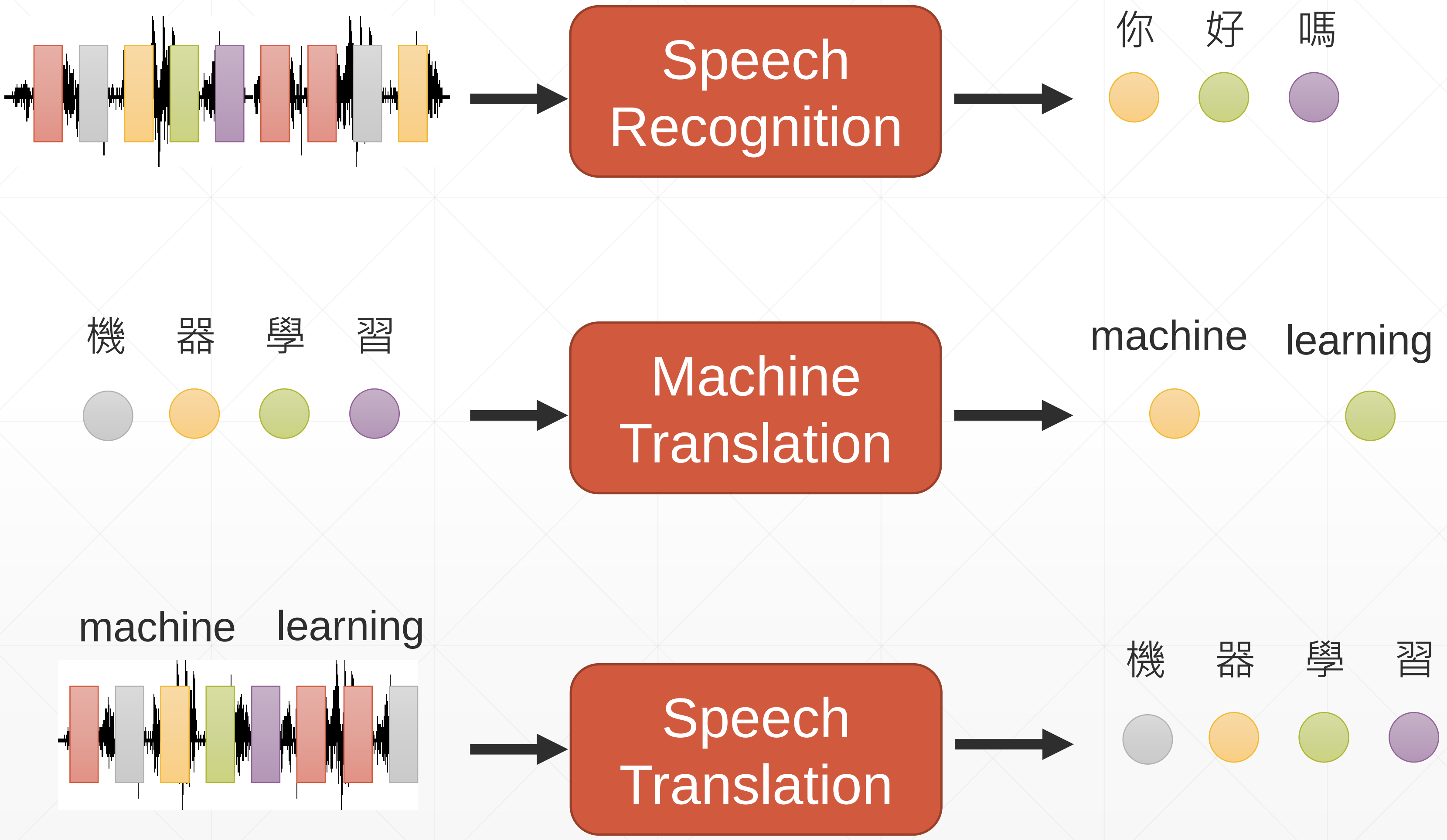
dog = [ 0 0 0 1 0 …… ]

elephant = [ 0 0 0 0 1 …… ]

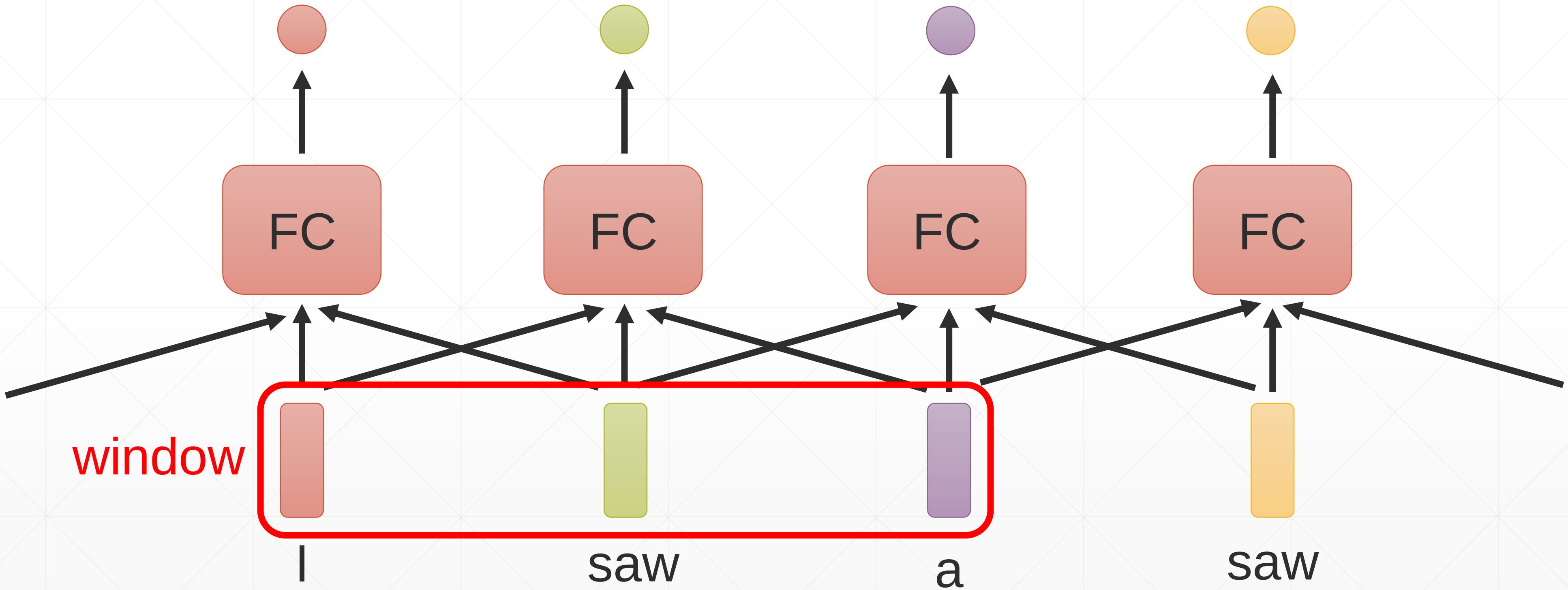
***Word Embedding***



# Sequence-to-sequence (Seq2seq)

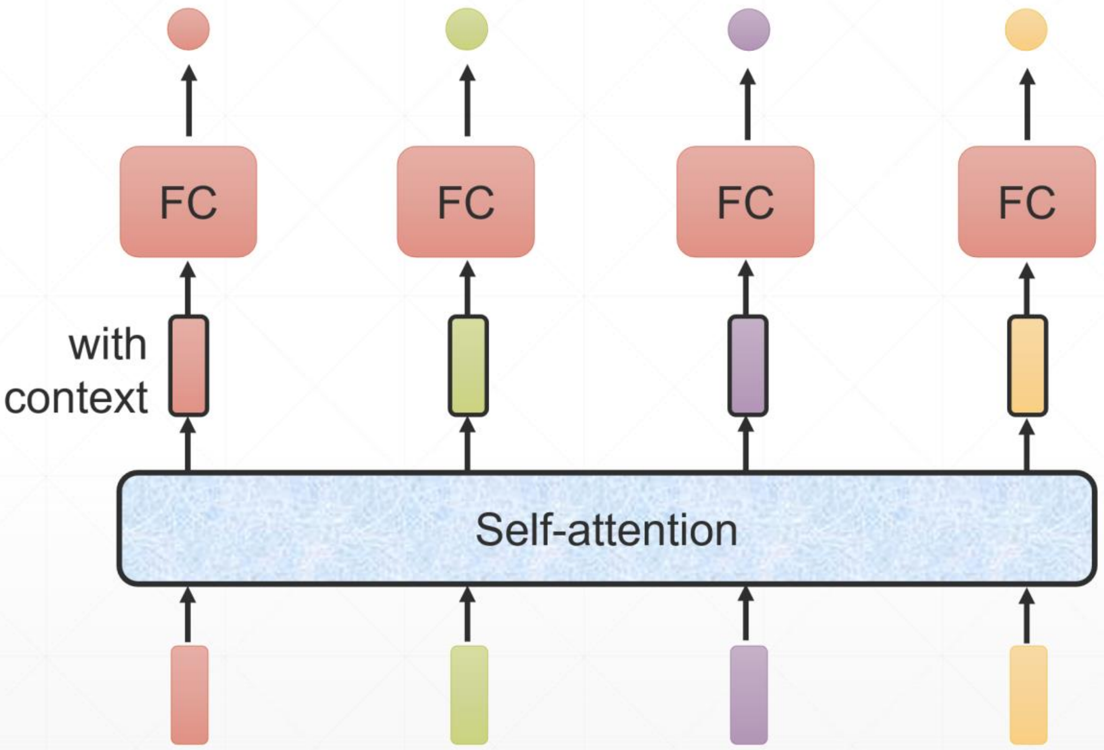


# Start with Fully-connected

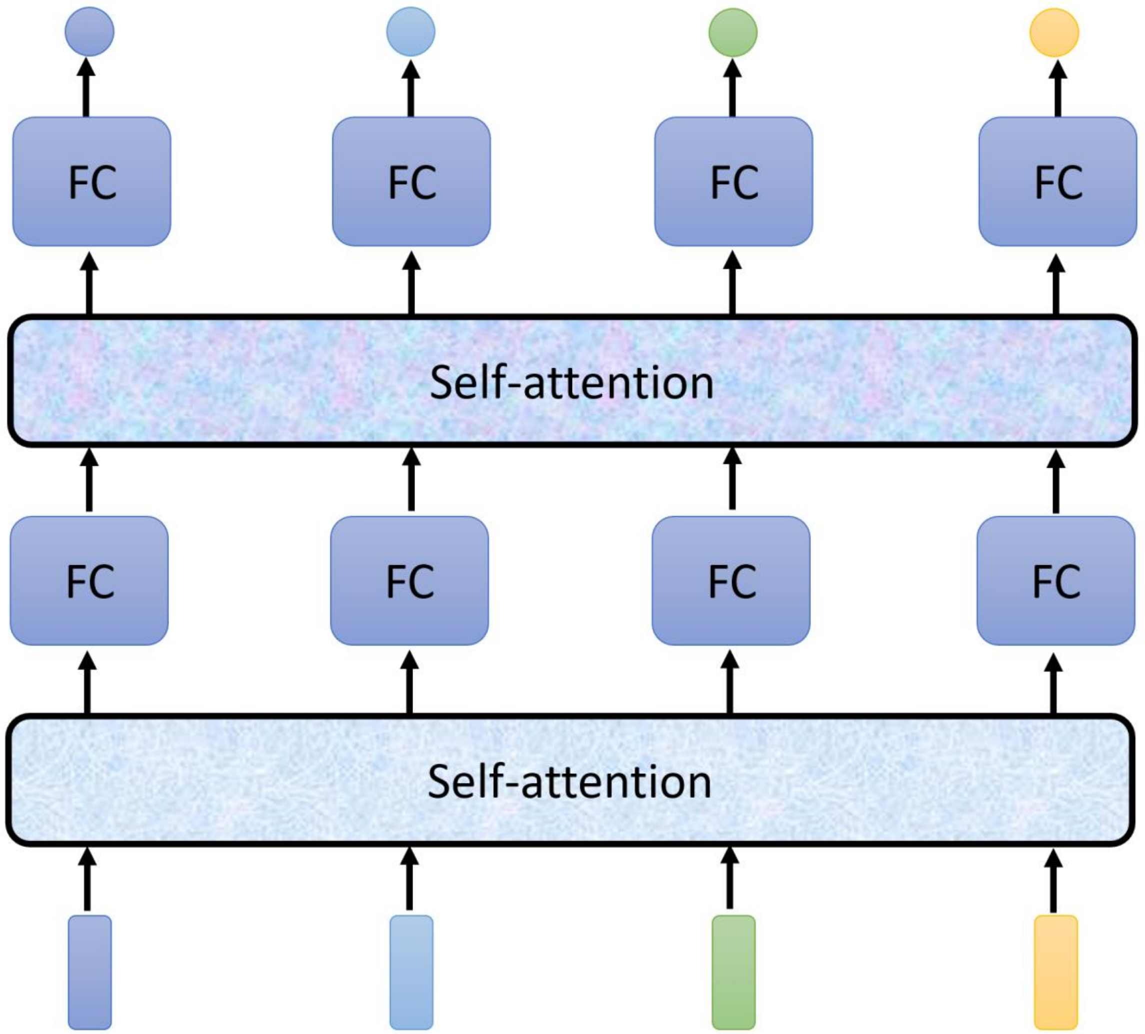



How to consider the whole sequence? → Self-attention

# Self-attention

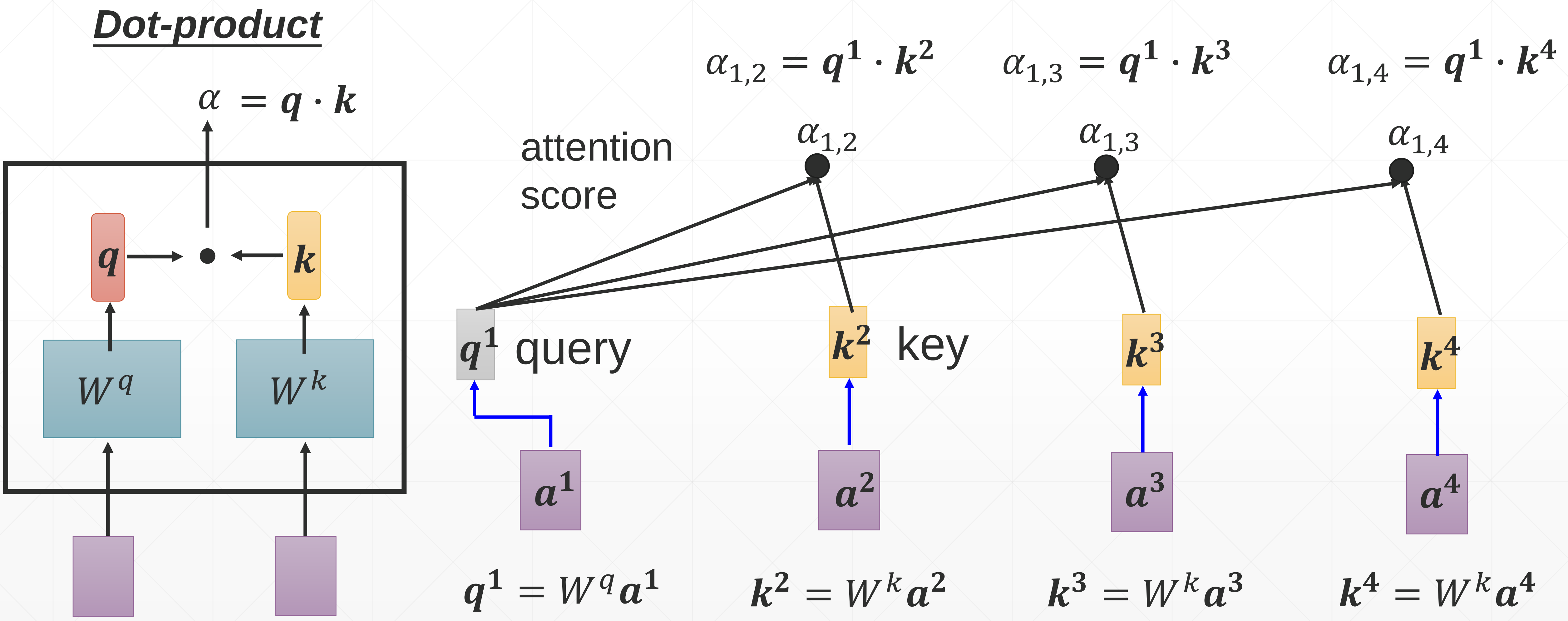



Self-attention

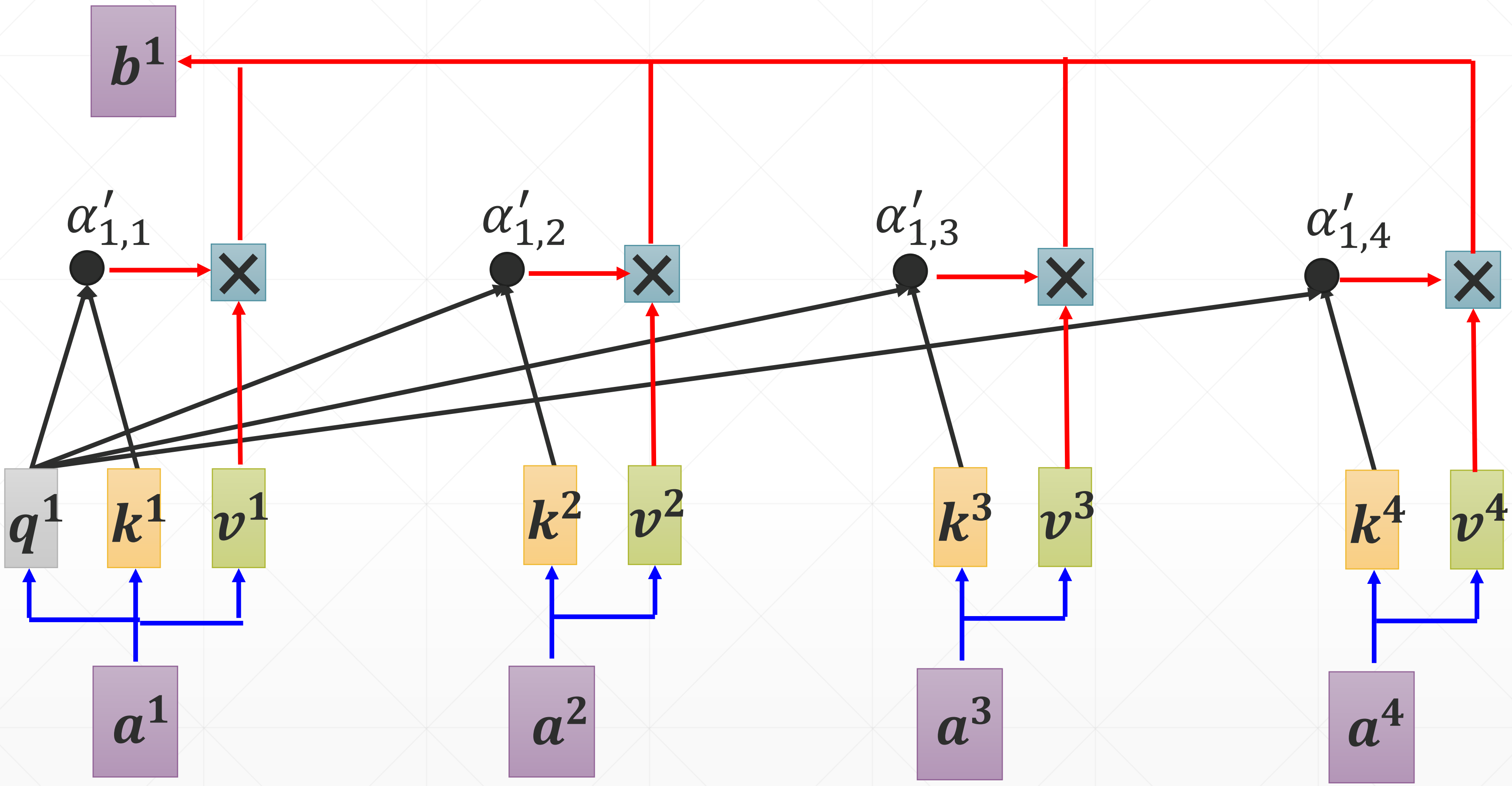



Transformer

# Self-attention

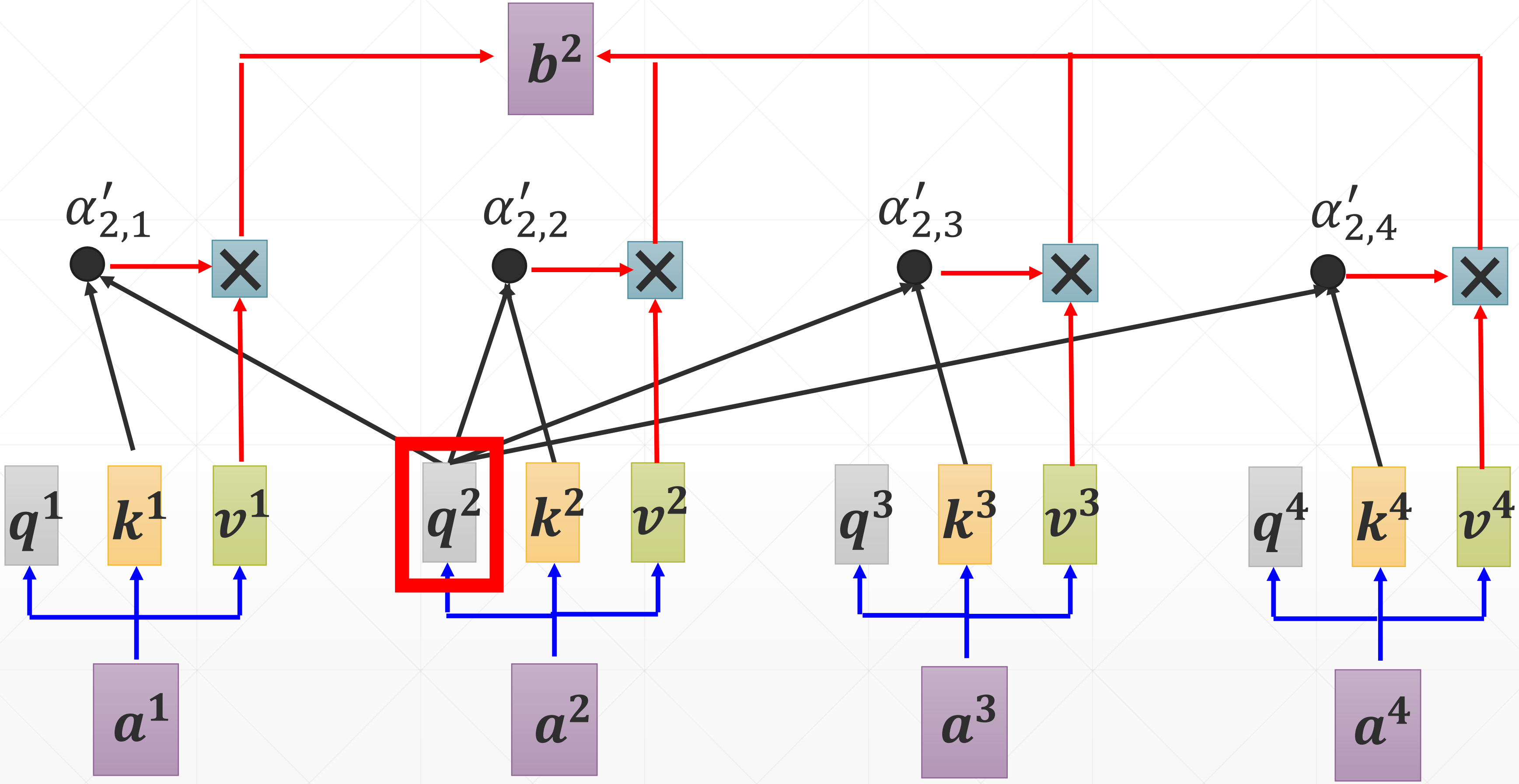


# Self-attention

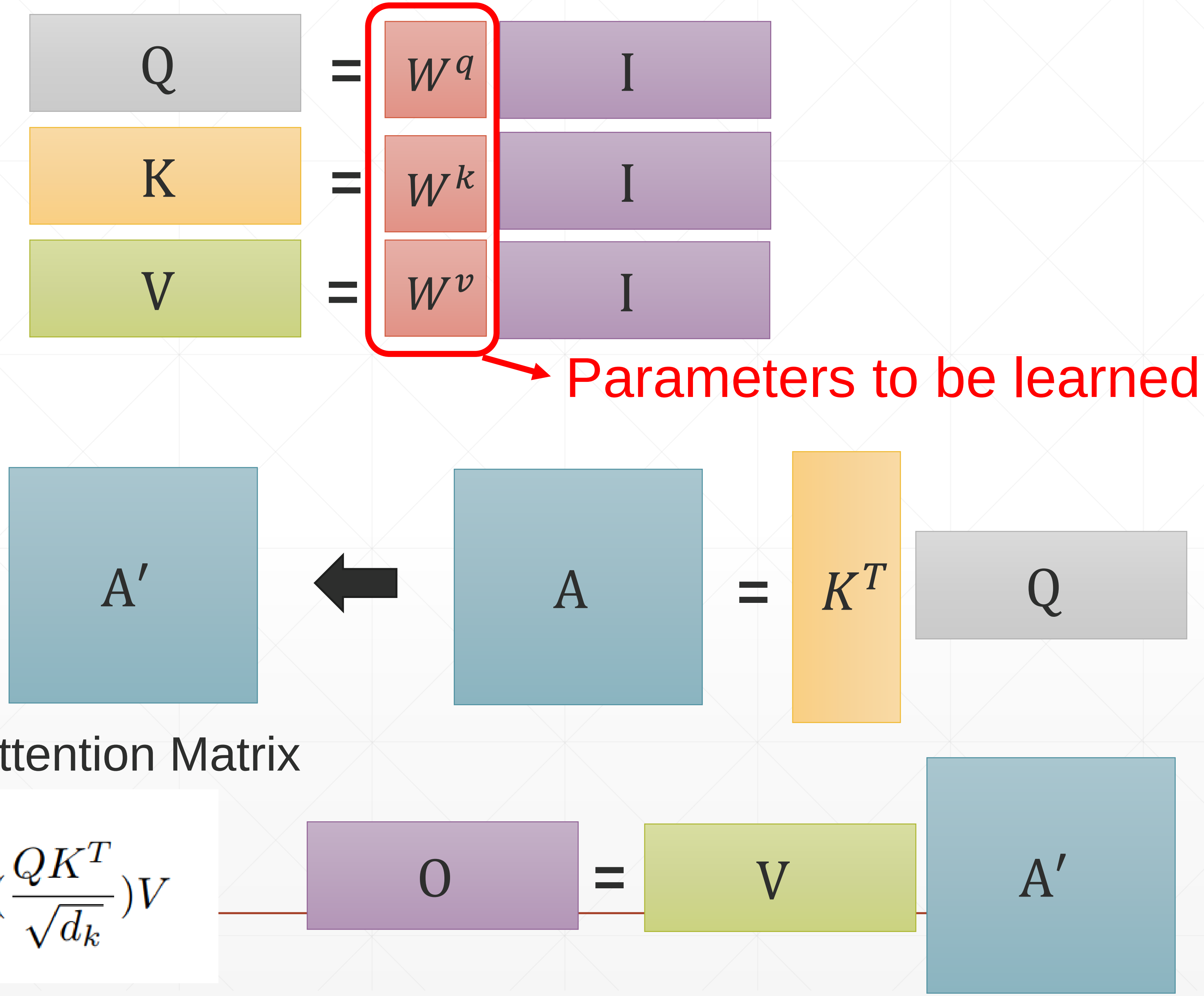



$v^1 = W^v a^1$ $\quad$ $v^2 = W^v a^2$ $\quad$ $v^3 = W^v a^3$ $\quad$ $v^4 = W^v a^4$

# Self-attention



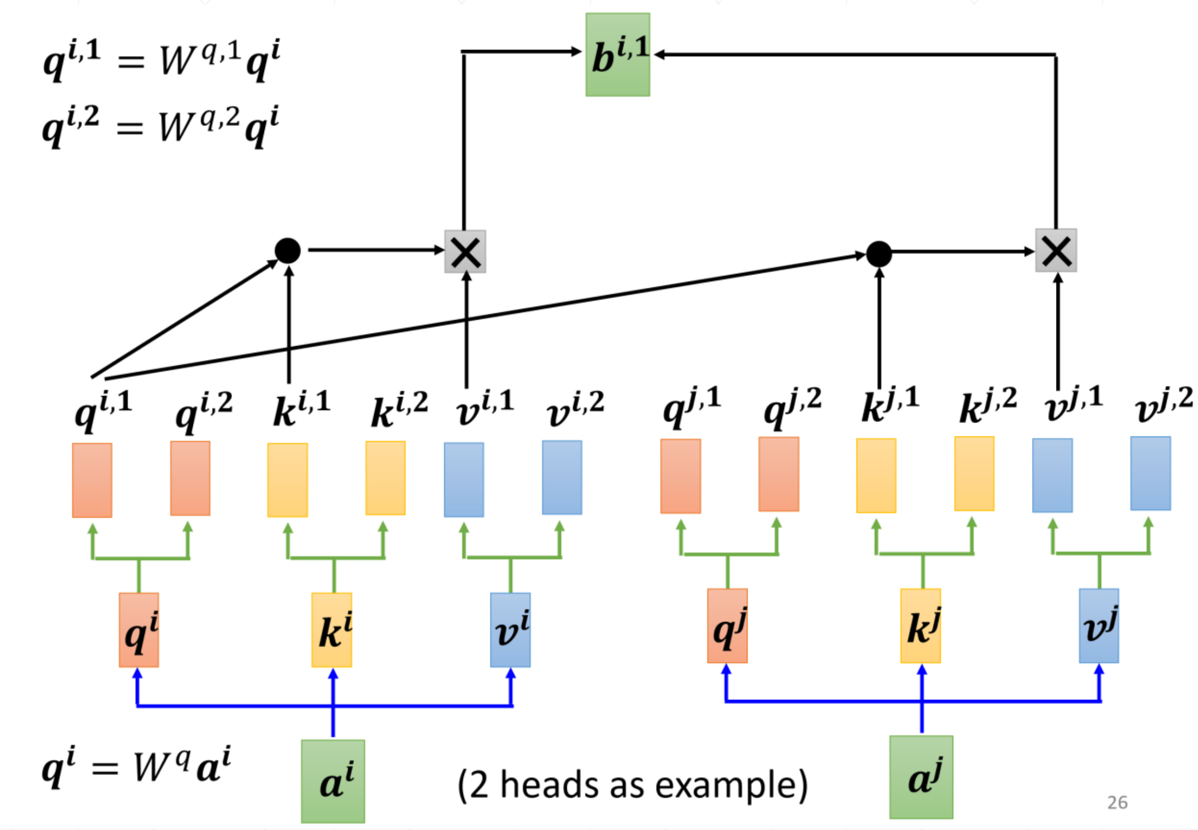
# Self-attention

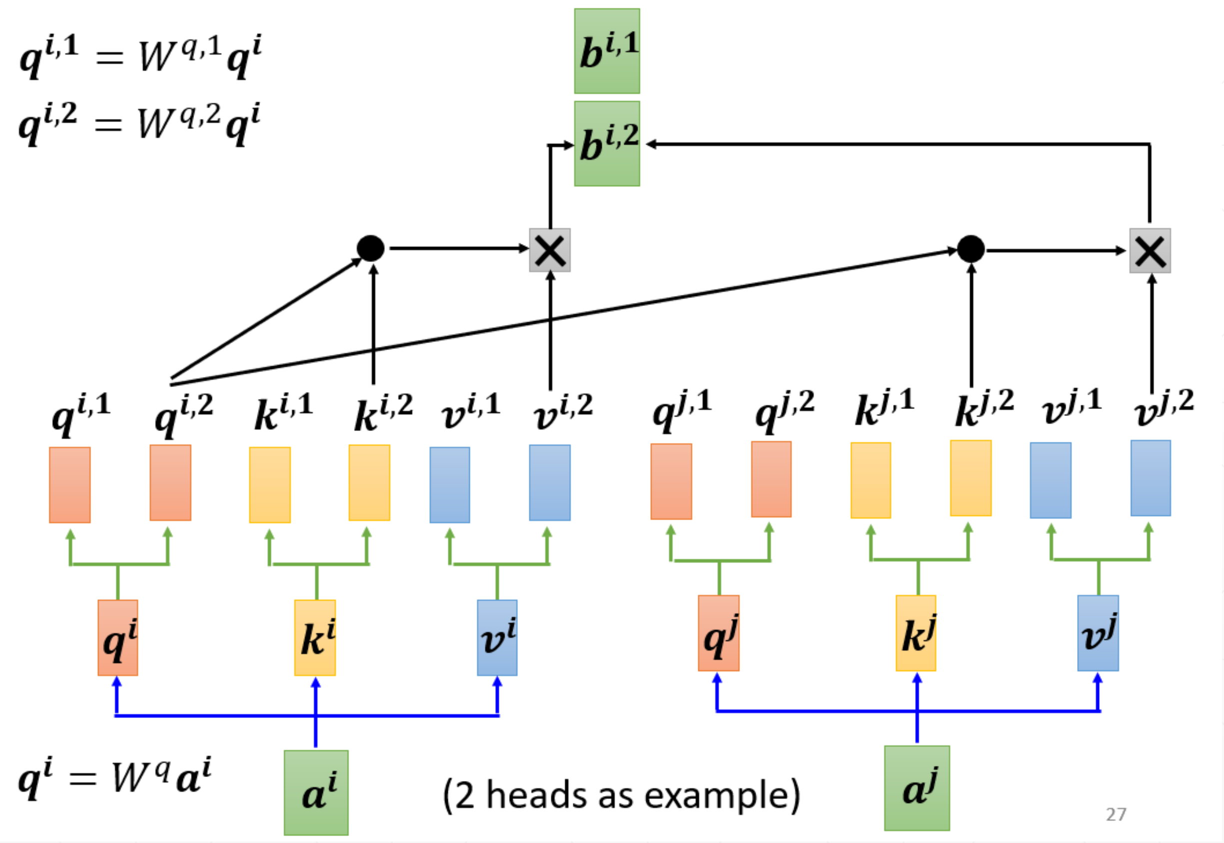



$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

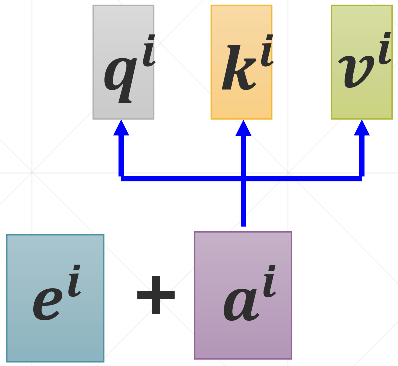
# Multi-head Self-attention

$$b^i = W^O \begin{bmatrix} b^{i,1} \\ b^{i,2} \end{bmatrix}$$



# Positional Encoding




## learned from data →BERT

Word embedding output: (b, N, 512) → self.pos_embedding = nn.Parameter(torch.randn(1,N,512))

In this work, we use sine and cosine functions of different frequencies:

## hand-crafted

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

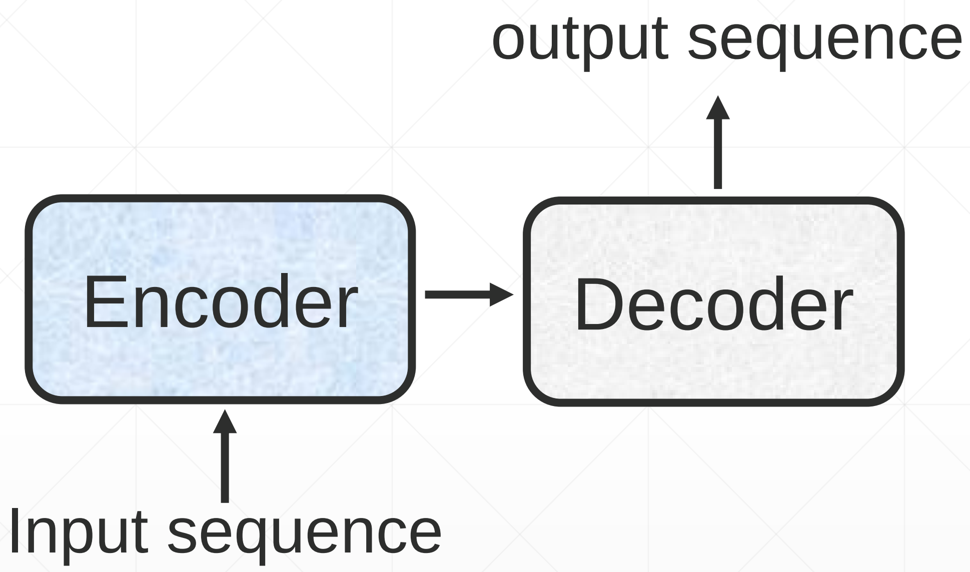
$$\begin{cases} sin(\alpha+\beta) = sin\alpha cos\beta + cos\alpha sin\beta \\ cos(\alpha+\beta) = cos\alpha cos\beta - sin\alpha sin\beta \end{cases}$$



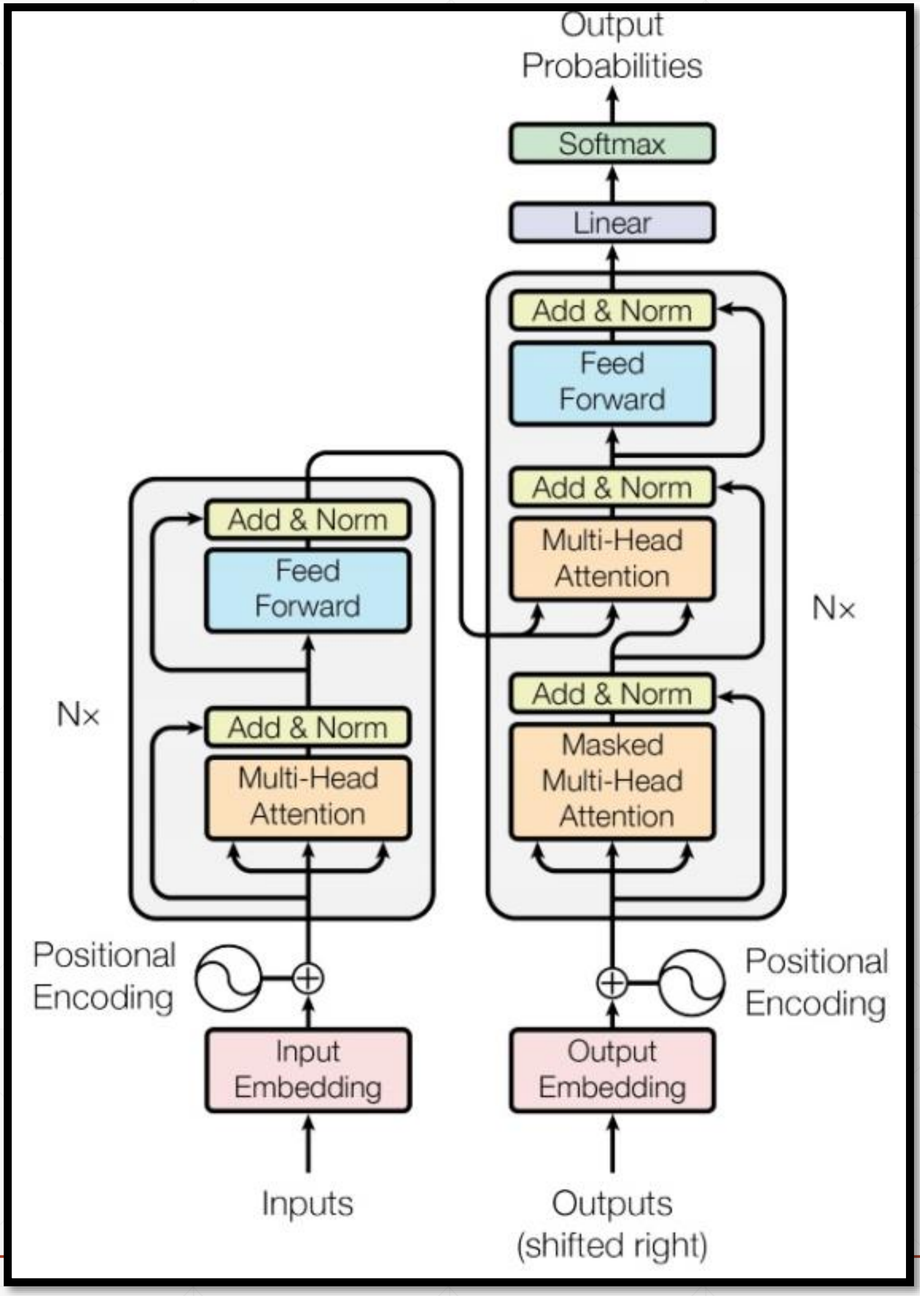
可以将$PE(pos+k)$用$PE(pos)$进行线性表出：

$$\begin{cases} PE(pos+k,2i) = PE(pos,2i) \times PE(k,2i+1) + PE(pos,2i+1) \times PE(k,2i) \\ PE(pos+k,2i+1) = PE(pos,2i+1) \times PE(k,2i+1) - PE(pos,2i) \times PE(k,2i) \end{cases}$$

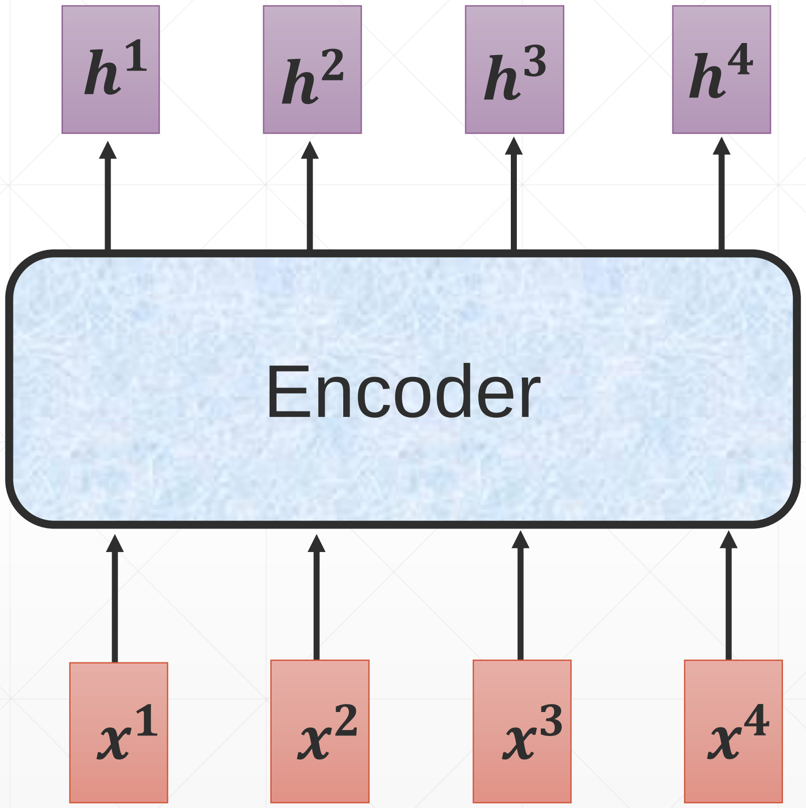
# Seq2seq Model



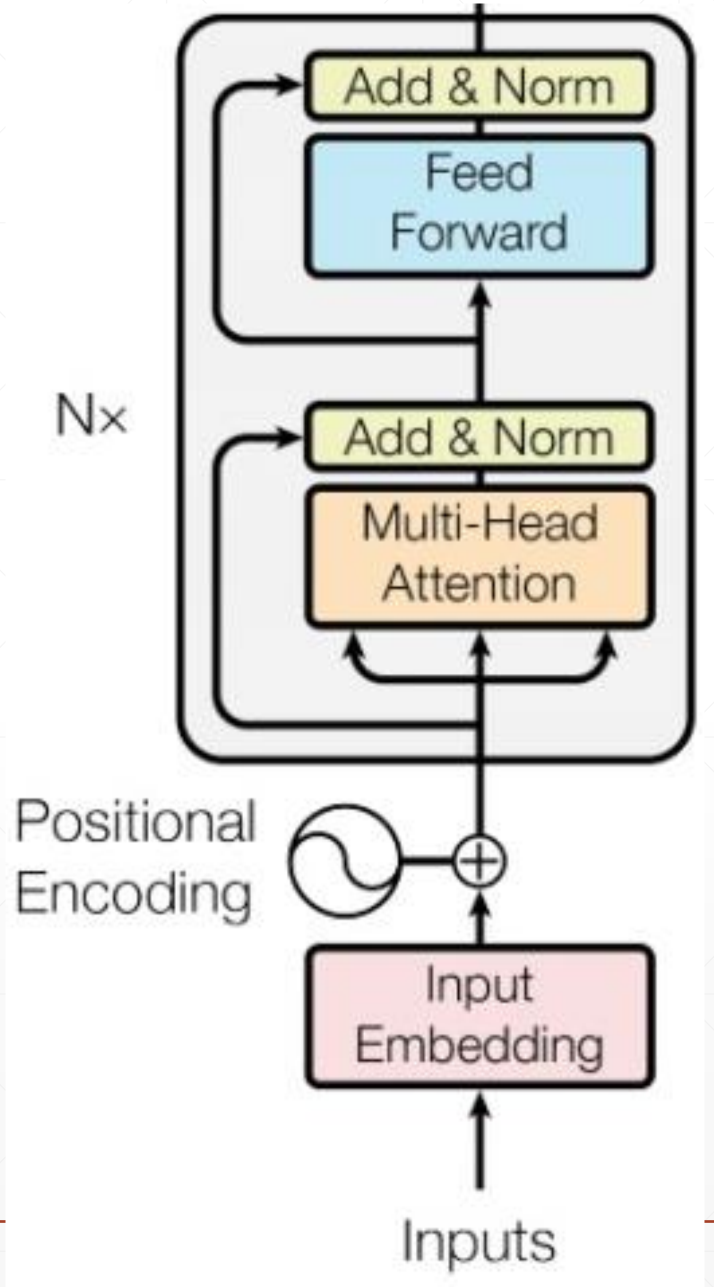

**Transformer→**



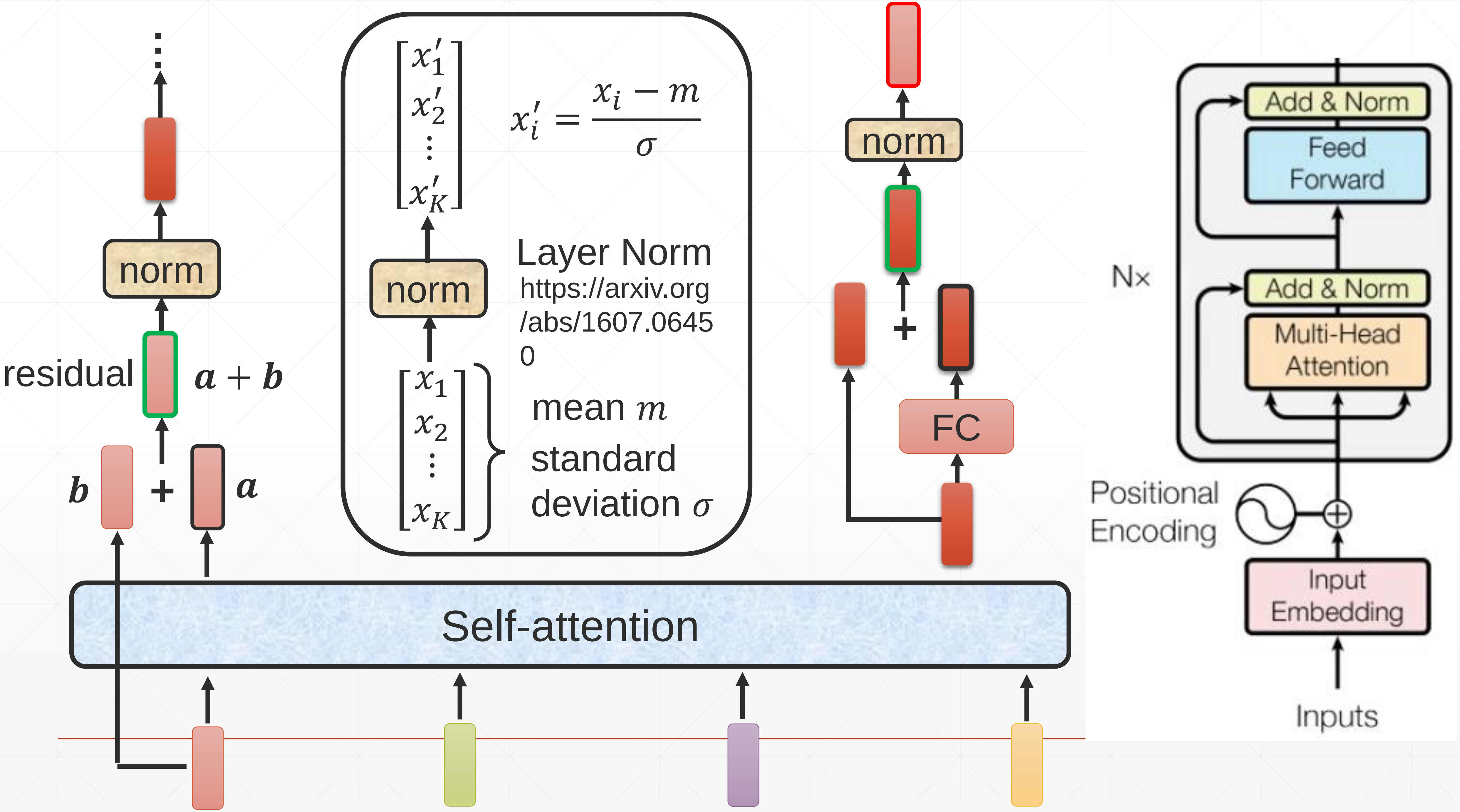
# Encoder



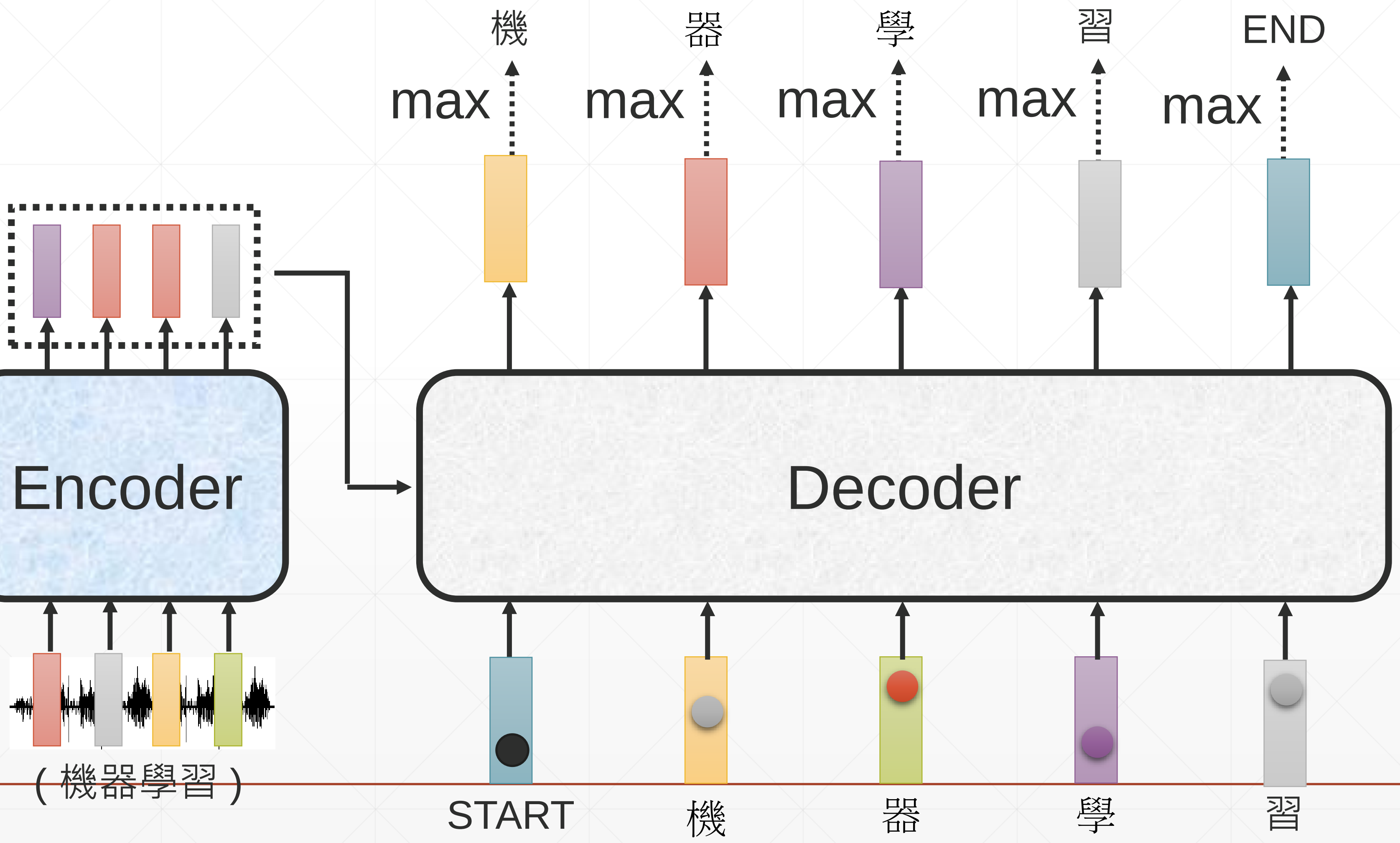
$h^1$
$h^2$
$h^3$
$h^4$
Encoder
$x^1$
$x^2$
$x^3$
$x^4$

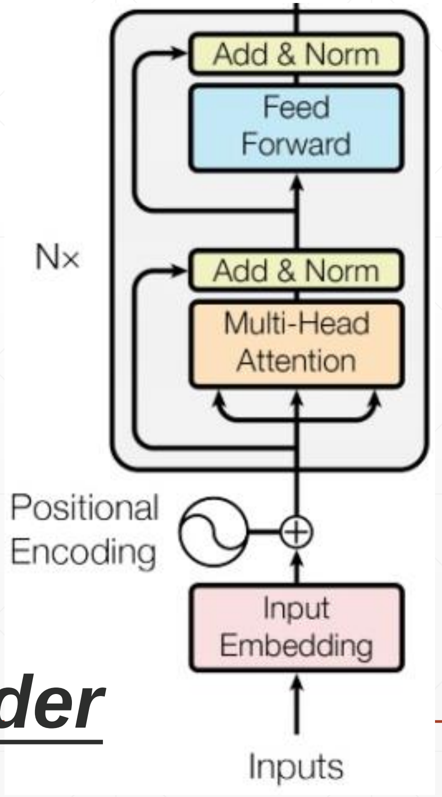
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Input
Embedding
Inputs


residual
$a + b$
$b$
+
$a$
norm
$x_i' = \frac{x_i - m}{\sigma}$
Layer Norm
https://arxiv.org/abs/1607.06450
mean $m$
standard deviation $\sigma$
norm
+
FC
norm
Self-attention
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

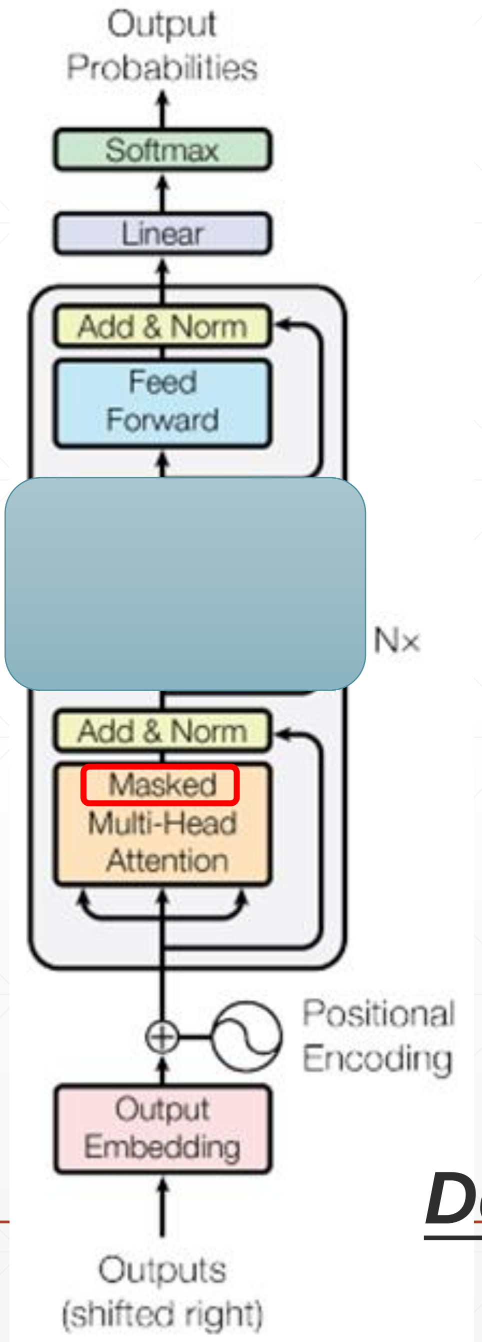
# Decoder



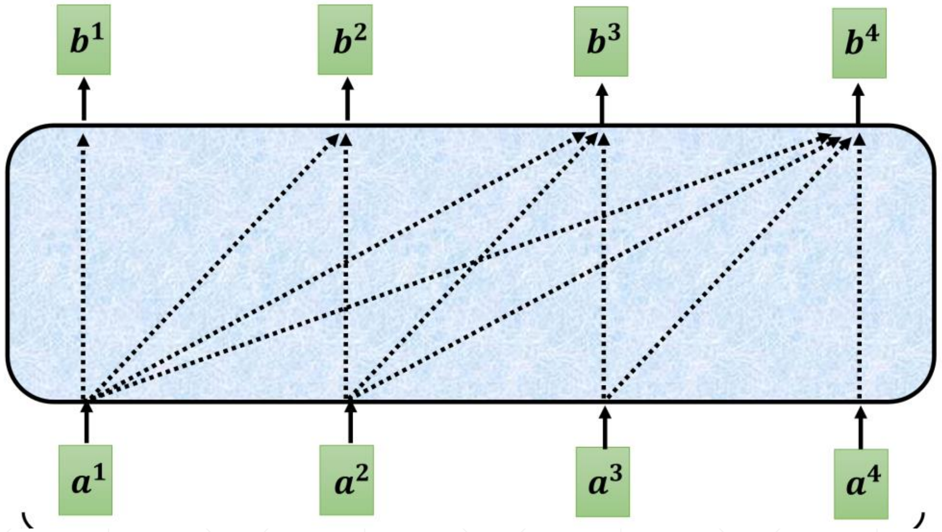
# Decoder



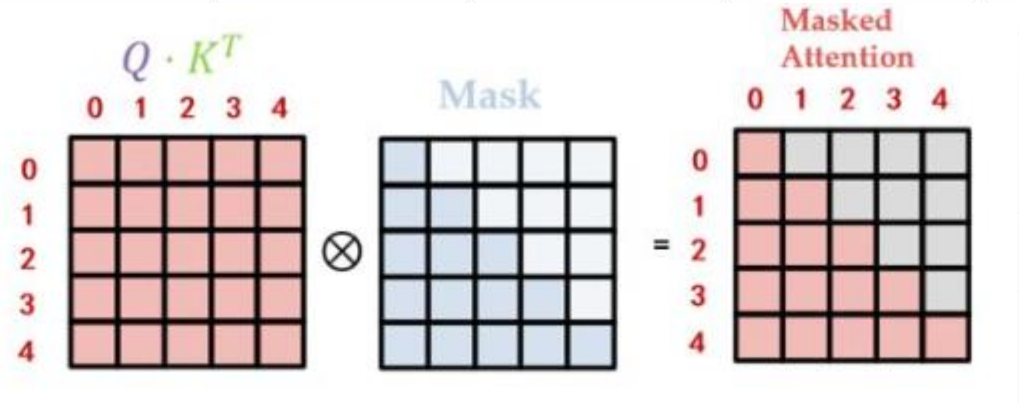

**Encoder**



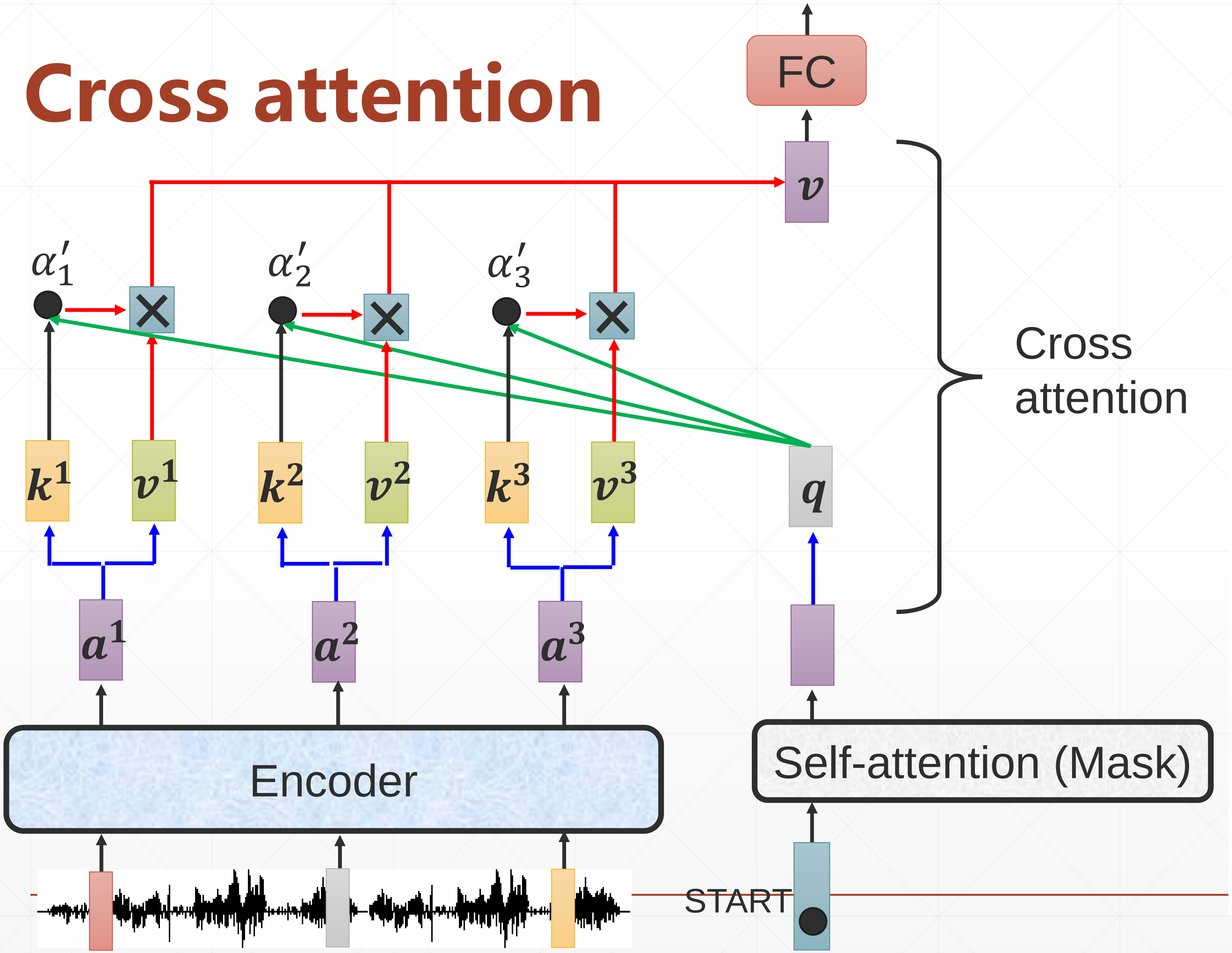

**Decoder**

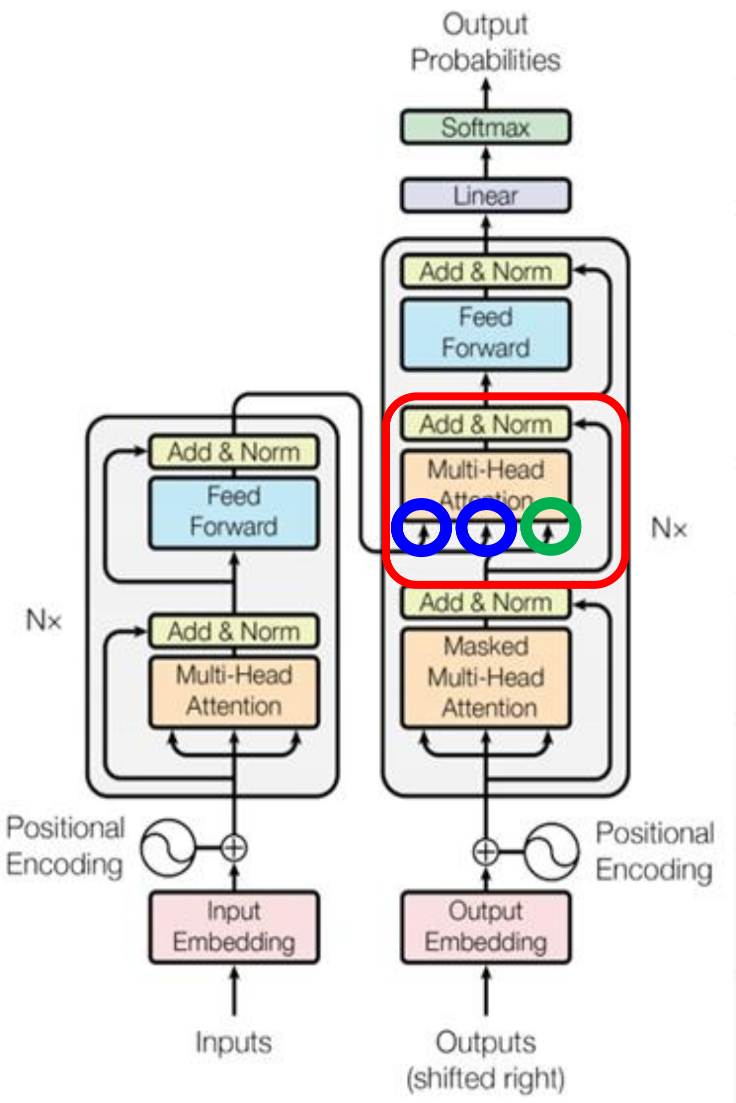



Masked Multi-Head Attention



# Cross attention



2

# An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

**Alexey Dosovitskiy[*,†], Lucas Beyer[*], Alexander Kolesnikov[*], Dirk Weissenborn[*], Xiaohua Zhai[*], Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby[*,†]**

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

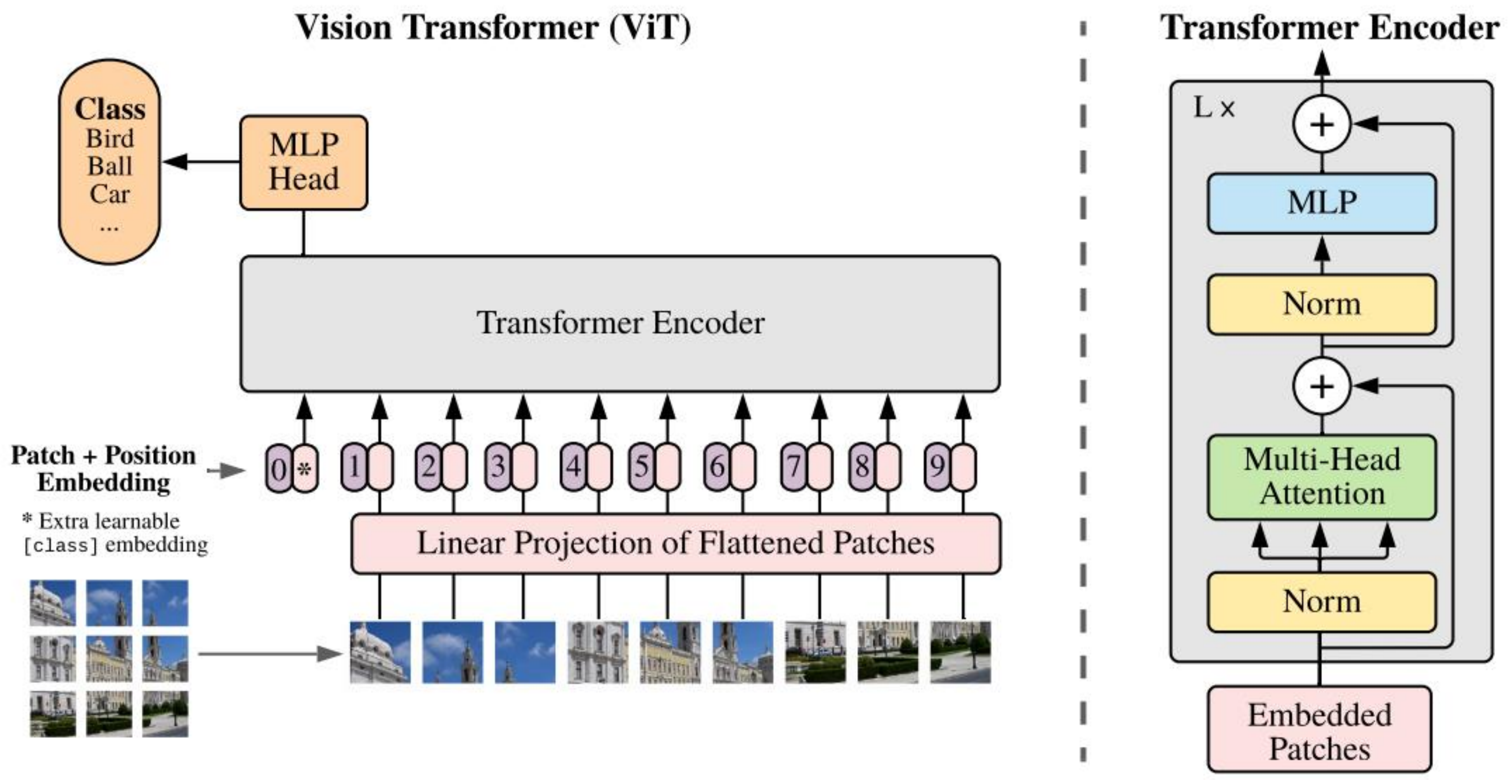
**ICLR 2021 oral**

**ViT→Visual Transformer**

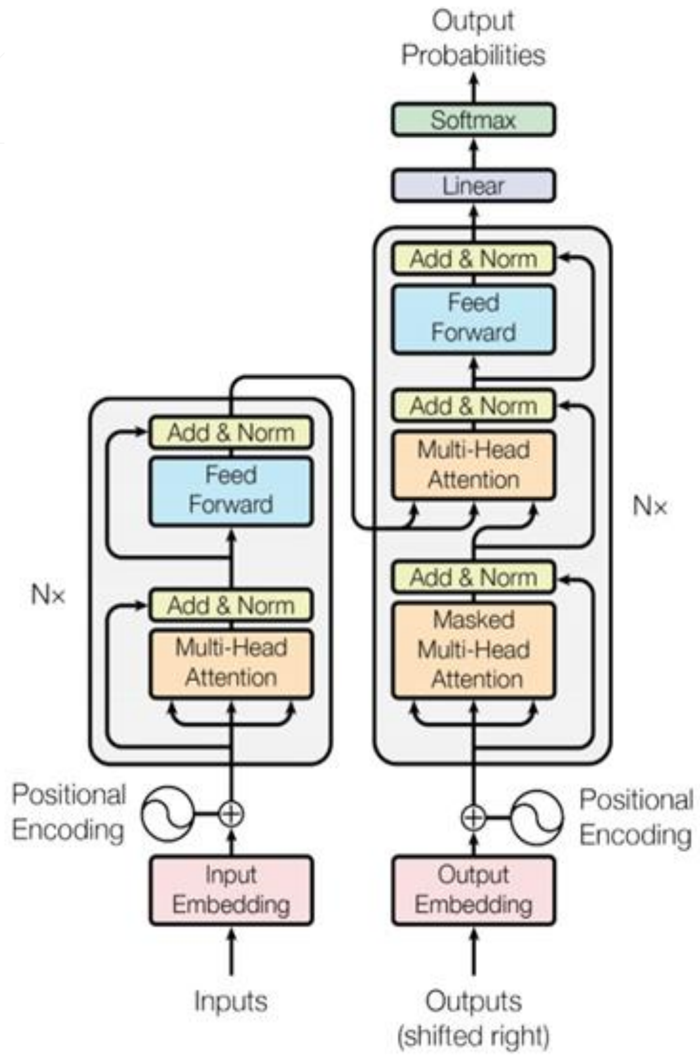
**Motivation:** 将Transformer直接应用到图像领域，做**图像分类**任务，

不修改Transformer结构，完全不加CNN。

# Model



Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

# Patch Embedding+Positional Encoding

标准的接受token的一维嵌入向量作为输入。为了处理二维数据，要进行reshape。
原始图像输入：（H,W）是图片分辨率，C是通道数

$$\mathbf{x} \in \mathbb{R}^{H\times W\times C}$$

reshape（分割patch）：P是patch的大小，N是patch的个数

$$\mathbf{x} \in \mathbb{R}^{N\times(P^2\cdot C)},\quad N = HW/P^2$$

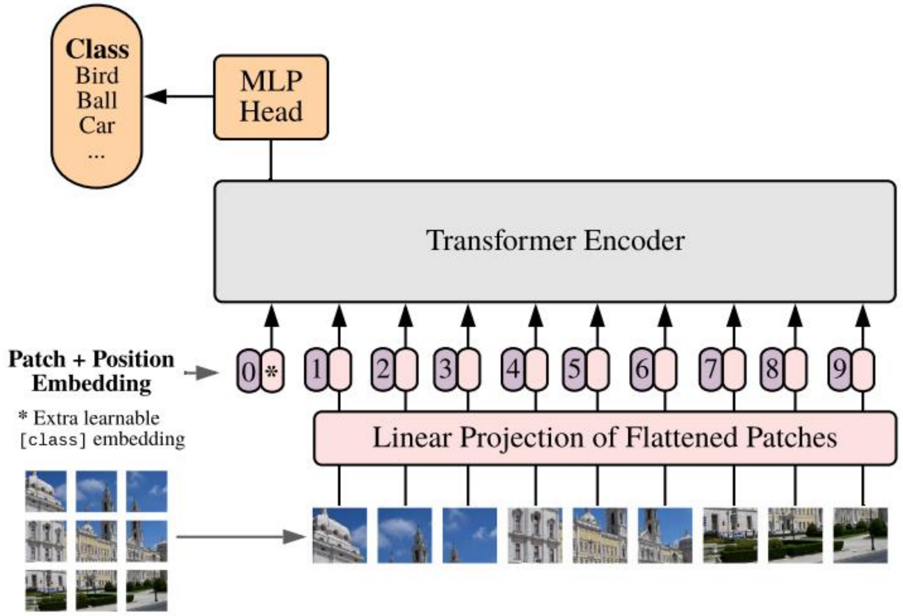
→ 分块

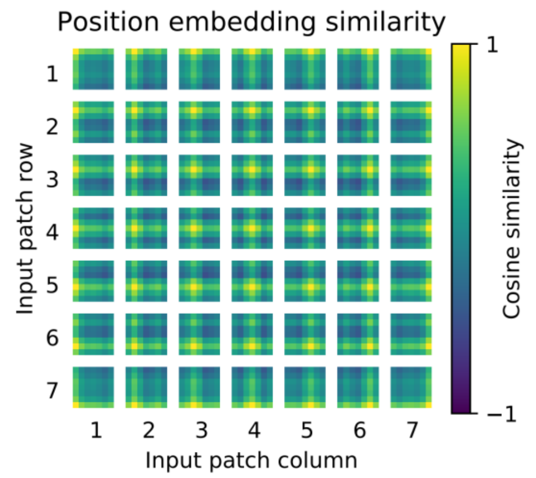
flatten(拍平，映射成Transformer接受的固定大小D，映射E是可学习的):

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos},\ \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$$

→ 维度转换

映射后的结果称为 patch embeddings。
在patch前面添加一个可学习的xclass，代表着图片的标签信息（全局信息）

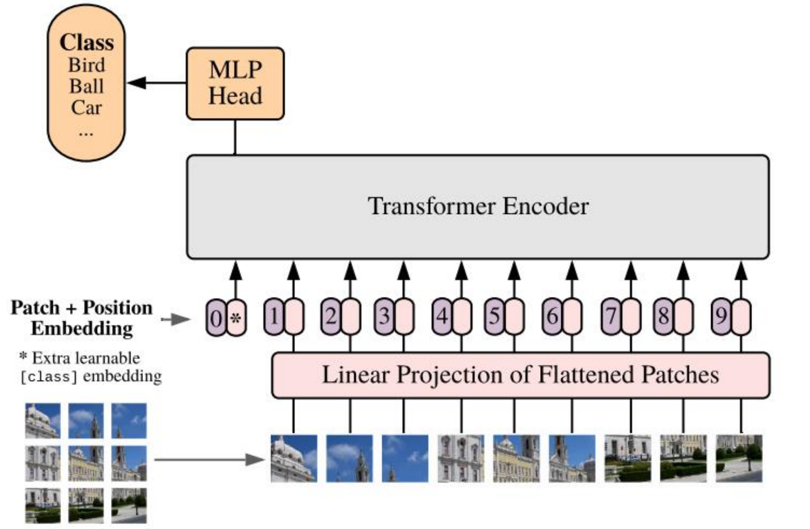


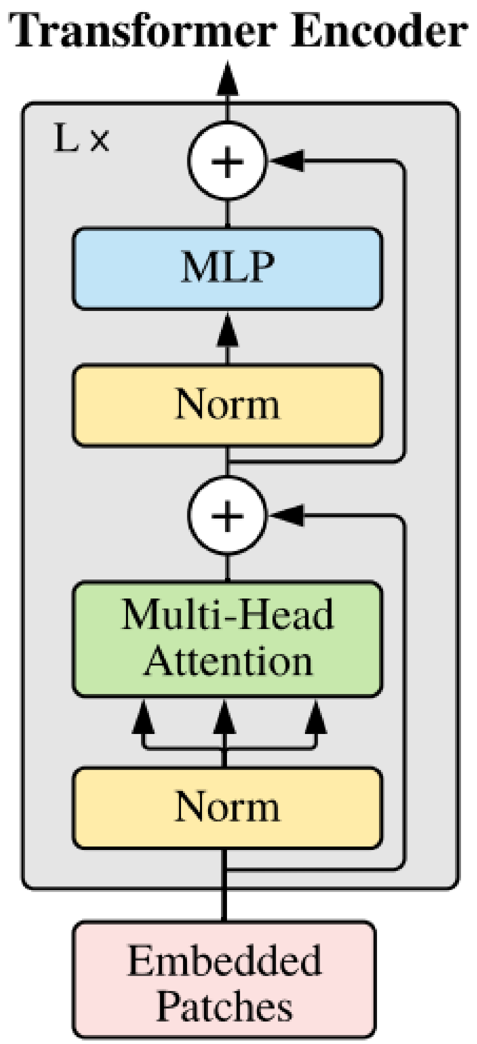
# Transformer Encoder

$$
\begin{aligned}
\mathbf{z'}_\ell &= \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, && \ell = 1 \ldots L \\
\mathbf{z}_\ell &= \mathrm{MLP}(\mathrm{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, && \ell = 1 \ldots L \\
\mathbf{y} &= \mathrm{LN}(\mathbf{z}_L^0)
\end{aligned}
$$

# Fine-tuning and high resolution




1. Pre-train ViT on large datasets
   Fine-tune to (smaller) downstream tasks
2. Remove the pre-trained prediction head
   Attach a D×K feed forward layer
   K is the number of downstream classes
3. If high resolution, keep the patch size the same
   but a larger effective sequence length



# Experiments

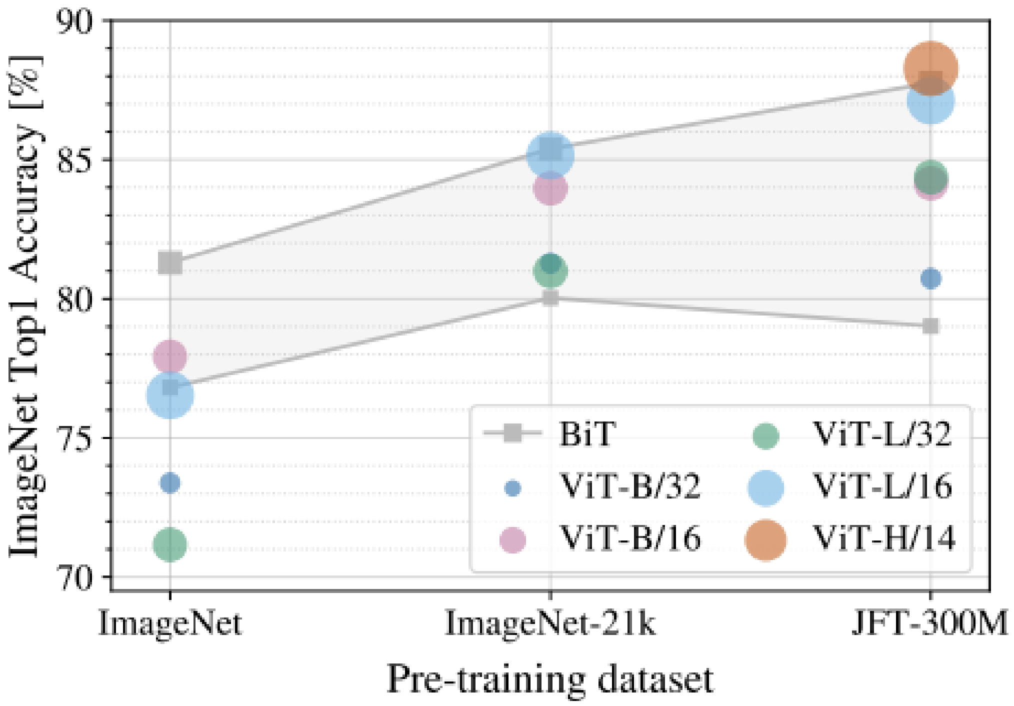
| Model | Layers | Hidden size $D$ | MLP size | Heads | Params |
|---|---|---|---|---|---|
| ViT-Base | 12 | 768 | 3072 | 12 | 86M |
| ViT-Large | 24 | 1024 | 4096 | 16 | 307M |
| ViT-Huge | 32 | 1280 | 5120 | 16 | 632M |

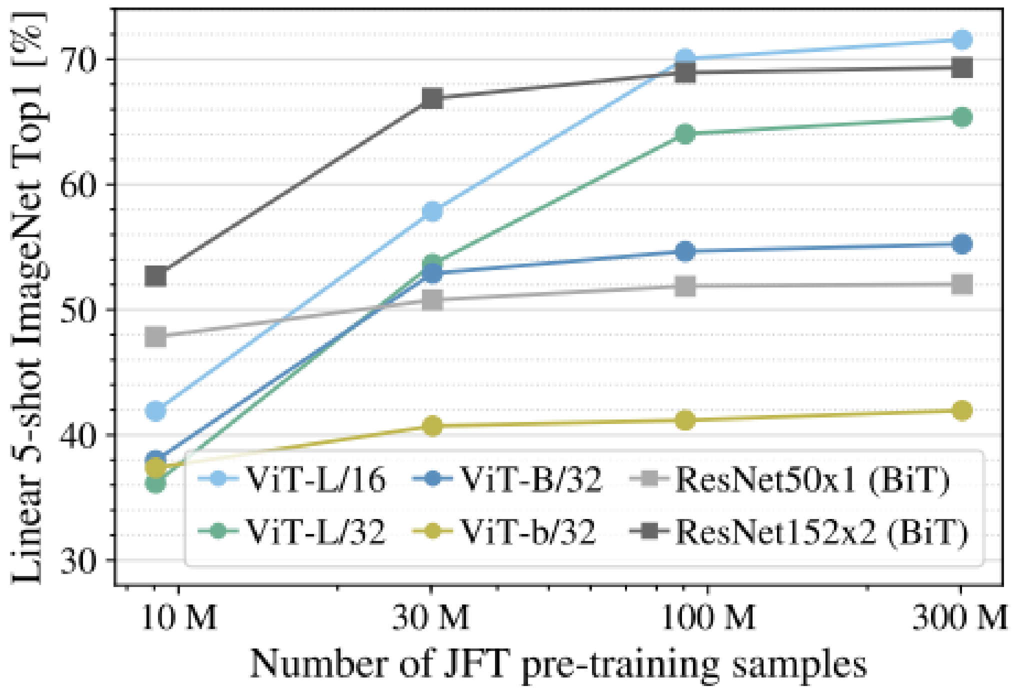
Table 1: Details of Vision Transformer model variants.

- Pre-train datasets:
  - ILSVRC-2012 ImageNet dataset: 1000 classes
  - ImagetNet-21k: 21k classes
  - JFT: 18k High Resolution Images
- Fine-tuning datasets:
  - CIFAR-10/100
  - Oxford-IIIT Pets
  - Oxford Flowers-102
  - VTAB

| | Ours-JFT (ViT-H/14) | Ours-JFT (ViT-L/16) | Ours-I21K (ViT-L/16) | BiT-L (ResNet152x4) | Noisy Student (EfficientNet-L2) |
|---|---|---|---|---|---|
| ImageNet | **88.55** $\pm$ 0.04 | 87.76 $\pm$ 0.03 | 85.30 $\pm$ 0.02 | 87.54 $\pm$ 0.02 | 88.4/88.5* |
| ImageNet ReaL | **90.72** $\pm$ 0.05 | 90.54 $\pm$ 0.03 | 88.62 $\pm$ 0.05 | 90.54 | 90.55 |
| CIFAR-10 | **99.50** $\pm$ 0.06 | 99.42 $\pm$ 0.03 | 99.15 $\pm$ 0.03 | 99.37 $\pm$ 0.06 | – |
| CIFAR-100 | **94.55** $\pm$ 0.04 | 93.90 $\pm$ 0.05 | 93.25 $\pm$ 0.05 | 93.51 $\pm$ 0.08 | – |
| Oxford-IIIT Pets | **97.56** $\pm$ 0.03 | 97.32 $\pm$ 0.11 | 94.67 $\pm$ 0.15 | 96.62 $\pm$ 0.23 | – |
| Oxford Flowers-102 | 99.68 $\pm$ 0.02 | **99.74** $\pm$ 0.00 | 99.61 $\pm$ 0.02 | 99.63 $\pm$ 0.03 | – |
| VTAB (19 tasks) | **77.63** $\pm$ 0.23 | 76.28 $\pm$ 0.46 | 72.72 $\pm$ 0.21 | 76.29 $\pm$ 1.70 | – |
| TPUv3-core-days | 2.5k | 0.68k | 0.23k | 9.9k | 12.3k |

# Experiments



# 3 SETR

**Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers**

Sixiao Zheng[1*] Jiachen Lu[1] Hengshuang Zhao[2] Xiatian Zhu[3] Zekun Luo[4] Yabiao Wang[4]
Yanwei Fu[1] Jianfeng Feng[1] Tao Xiang[3, 5] Philip H.S. Torr[2] Li Zhang[1†]
[1]Fudan University [2]University of Oxford [3]University of Surrey
[4]Tencent Youtu Lab [5]Facebook AI

**CVPR 2021**

https://fudan-zvg.github.io/SETR

## Motivation:

To break the standard FCN segmentation model, which has an encoder-decoder architecture: the encoder is for feature representation learning, while the decoder for pixel-level classification of the feature representations yielded by the encoder.

We propose to replace the stacked convolution layers based encoder with gradually reduced spatial resolution with a pure transformer , resulting in a new segmentation model termed SEgmentation TRansformer (SETR).

# Model

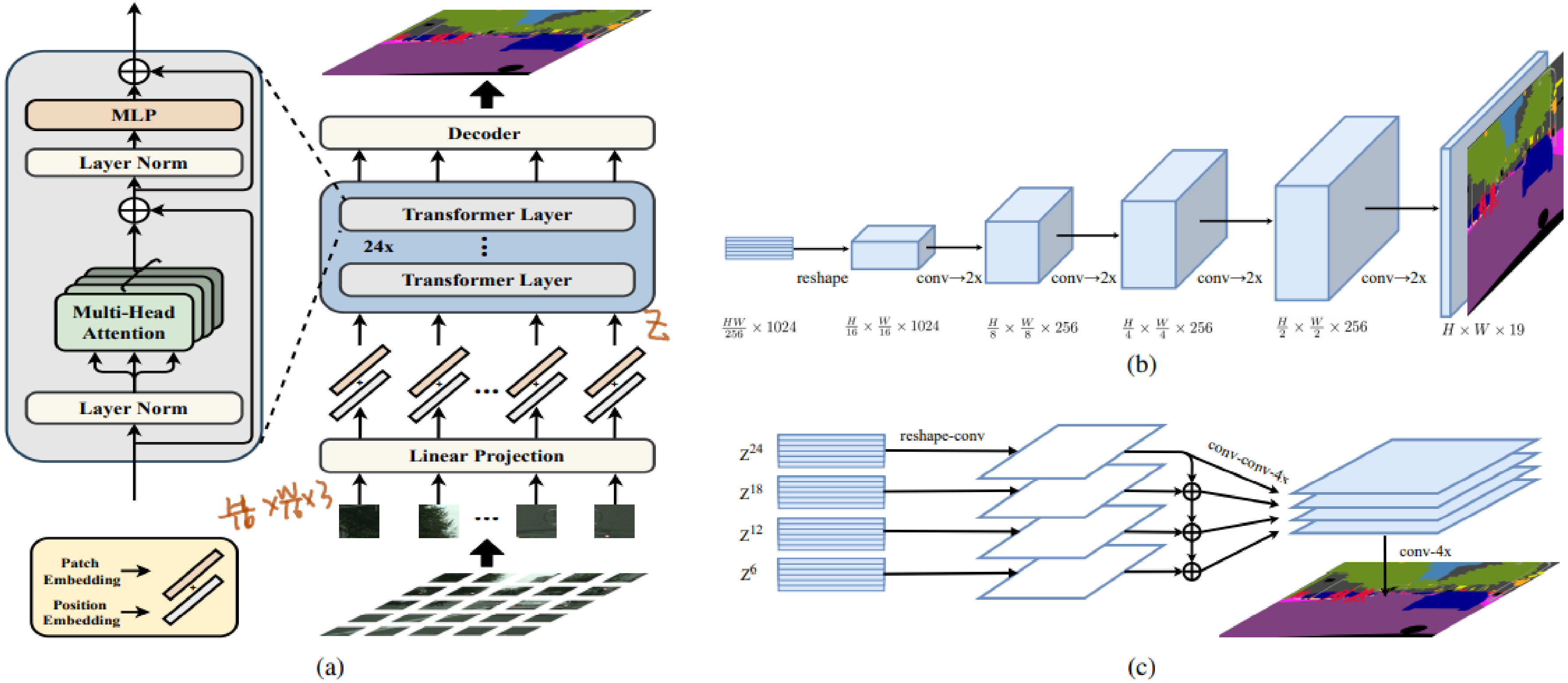



Figure 1. **Schematic illustration of the proposed** ***SEgmentation TRansformer*** **(SETR)** (a). We first split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. To perform pixel-wise segmentation, we introduce different decoder designs: (b) progressive upsampling (resulting in a variant called SETR-*PUP*); and (c) multi-level feature aggregation (a variant called SETR-*MLA*).

# Method

## Image to sequence:

1. Image to grid patches → 分块

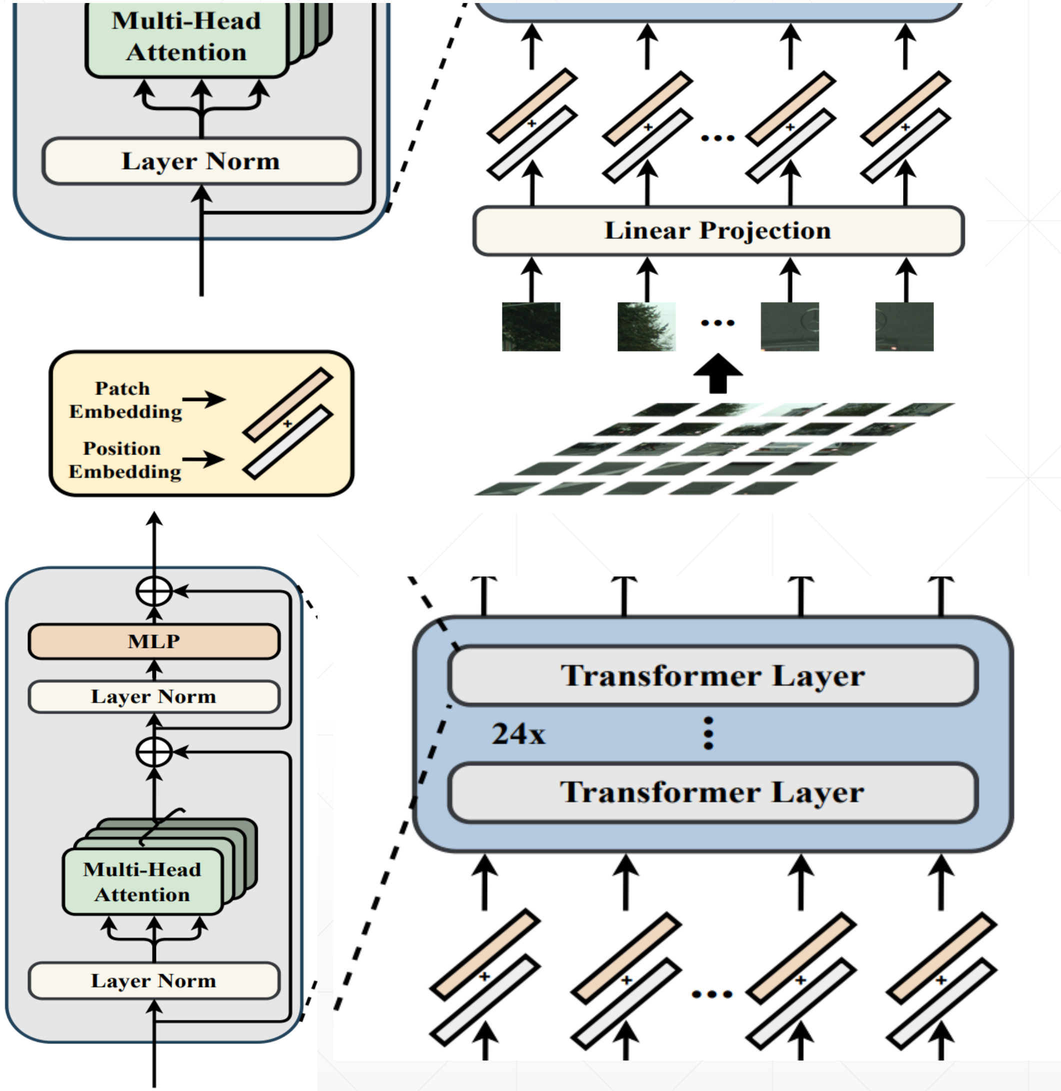
$$x_f \in \mathbb{R}^{\frac{H}{16}\times\frac{W}{16}\times C} \quad \text{Length L: } \frac{H}{16}\times\frac{W}{16}=\frac{HW}{256}$$

2. Linear Projection → 维度转换

$$f\colon p \longrightarrow e \in \mathbb{R}^C$$

3. Position Embedding → 位置嵌入

$$E=\{e_1+p_1,\ e_2+p_2,\ \cdots,\ e_L+p_L\}$$




**Transformer:** $\{Z^1,\ Z^2,\ \cdots,\ Z^{L_e}\}$ as the features of transformer layers.

# Decoder designs

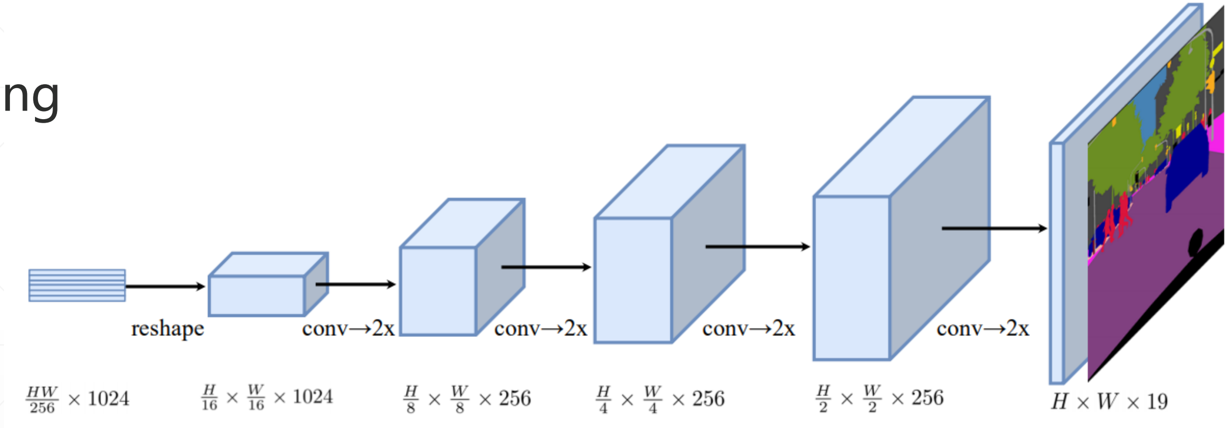
1. Naive: Naive upsampling

1×1 conv + batch norm + 1×1 conv

**bilinearly upsample** the output to the full image resolution, followed by a classification layer with **pixel-wise cross-entropy loss**.

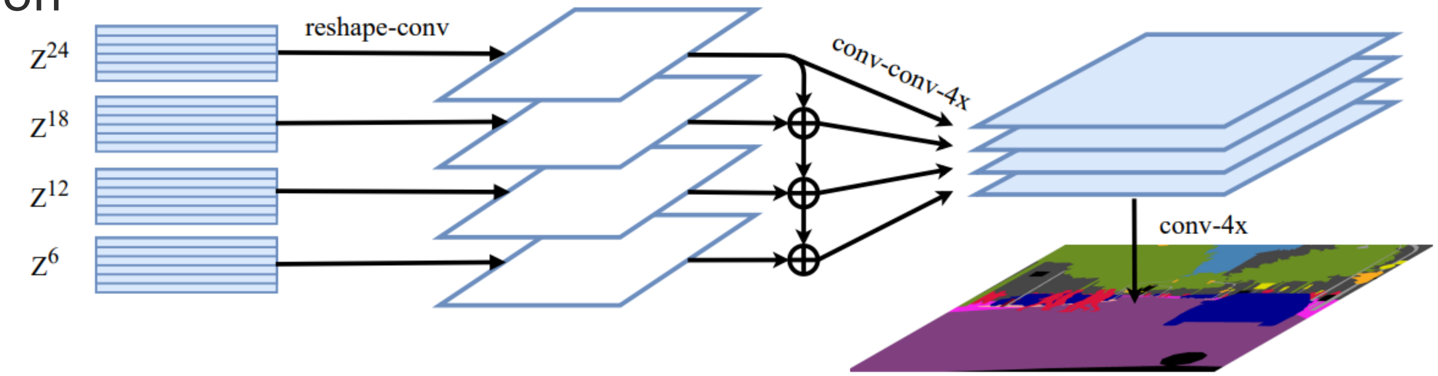
2. PUP: Progressive Upsampling

restrict upsampling to 2×



(b)

3. MLA: Multi-Level feature Aggregation



# Experiments

- Datasets: Cityscapes、ADE20K、PASCAL Context

| Method | Backbone | mIoU | Pixel Acc. |
|---|---|---|---|
| FCN (16, 160k, SS) [40] | ResNet-101 | 39.91 | 79.52 |
| FCN (16, 160k, MS) [40] | ResNet-101 | 41.40 | 80.65 |
| EncNet [55] | ResNet-101 | 44.65 | 81.69 |
| PSPNet [60] | ResNet-269 | 44.94 | 81.69 |
| DMNet [19] | ResNet-101 | 45.50 | - |
| CCNet [26] | ResNet-101 | 45.22 | - |
| Strip pooling [24] | ResNet-101 | 45.60 | 82.09 |
| APCNet [20] | ResNet-101 | 45.38 | - |
| OCNet [54] | ResNet-101 | 45.45 | - |
| SETR-*MLA* (8, 160k, SS) | T-Large | 48.27 | 82.52 |
| SETR-*MLA* (8, 160k, MS) | T-Large | 50.03 | 83.41 |
| SETR-*PUP* (16, 160k, SS) | T-Large | 48.58 | 82.90 |
| SETR-*PUP* (16, 160k, MS) | T-Large | 50.09 | **83.58** |
| SETR-*MLA* (16, 160k, SS) | T-Large | 48.64 | 82.64 |
| SETR-*MLA* (16, 160k, MS) | T-Large | **50.28** | 83.46 |

Table 3. **State-of-the-art comparison on the ADE20K dataset.** Performances of different model variants and batch size (*e.g.*, 8 or 16) are reported. SS: Single-scale inference. MS: Multi-scale inference.

| Method | Backbone | mIoU |
|---|---|---|
| FCN (16, 80k, SS) [40] | ResNet-101 | 44.47 |
| FCN (16, 80k, MS) [40] | ResNet-101 | 45.74 |
| PSPNet [60] | ResNet-101 | 47.80 |
| DANet [18] | ResNet-101 | 52.60 |
| EMANet [32] | ResNet-101 | 53.10 |
| SVCNet [15] | ResNet-101 | 53.20 |
| ACNet [16] | ResNet-101 | 54.10 |
| GFFNet [31] | ResNet-101 | 54.20 |
| APCNet [20] | ResNet-101 | 54.70 |
| SETR-*MLA* (8, 80k, SS) | T-Large | 54.39 |
| SETR-*MLA* (8, 80k, MS) | T-Large | 55.39 |
| SETR-*PUP* (16, 80k, SS) | T-Large | 54.40 |
| SETR-*PUP* (16, 80k, MS) | T-Large | 55.27 |
| SETR-*MLA* (16, 80k, SS) | T-Large | 54.87 |
| SETR-*MLA* (16, 80k, MS) | T-Large | **55.83** |

Table 4. **State-of-the-art comparison on the Pascal Context dataset.** Performances of different model variants and batch sizes (*e.g.*, 8 or 16) are reported. SS: Single-scale inference. MS: Multi-scale inference.

# 4 VL-BERT: PRE-TRAINING OF GENERIC VISUAL-LINGUISTIC REPRESENTATIONS

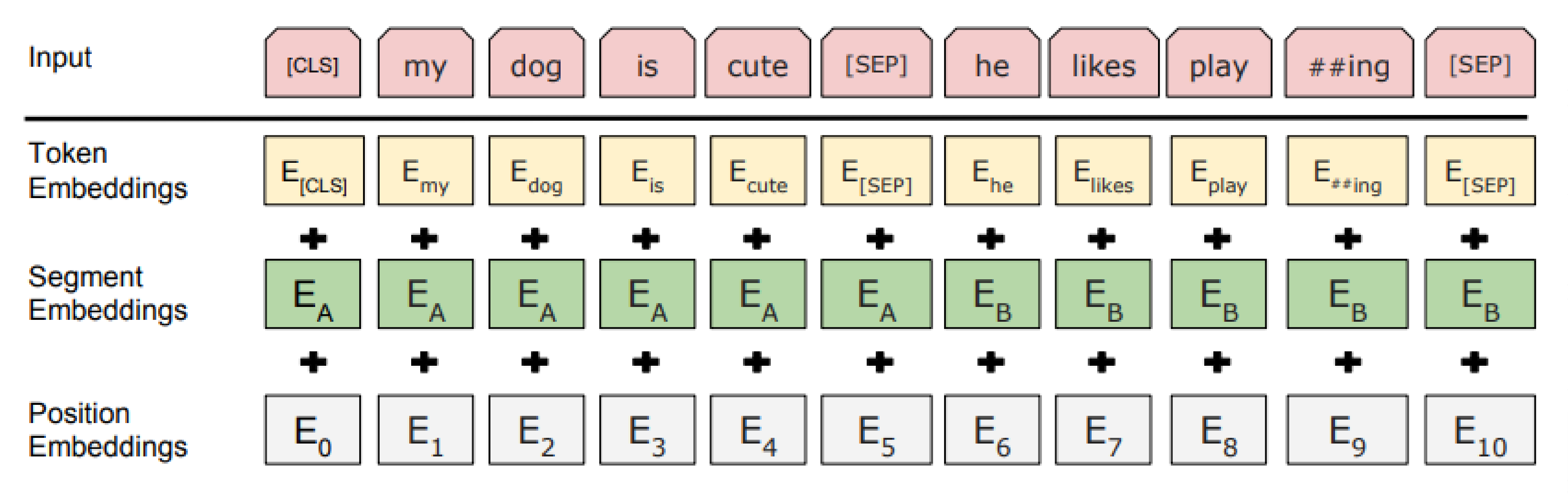
Weijie Su[1,2*], Xizhou Zhu[1,2*], Yue Cao[2], Bin Li[1], Lewei Lu[2], Furu Wei[2], Jifeng Dai[2]
[1]University of Science and Technology of China
[2]Microsoft Research Asia
{jackroos,ezra0408}@mail.ustc.edu.cn, binli@ustc.edu.cn
{yuecao,lewlu,fuwei,jifdai}@microsoft.com

ICLR 2020

**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**

**Jacob Devlin** **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

# Embedding&Pre-Training Task in BERT




→ 上游任务：Mask Language Modelling & Next Sentence Prediction

→ 下游任务：句子对分类任务、单句子分类任务、问答任务、单句标注任务

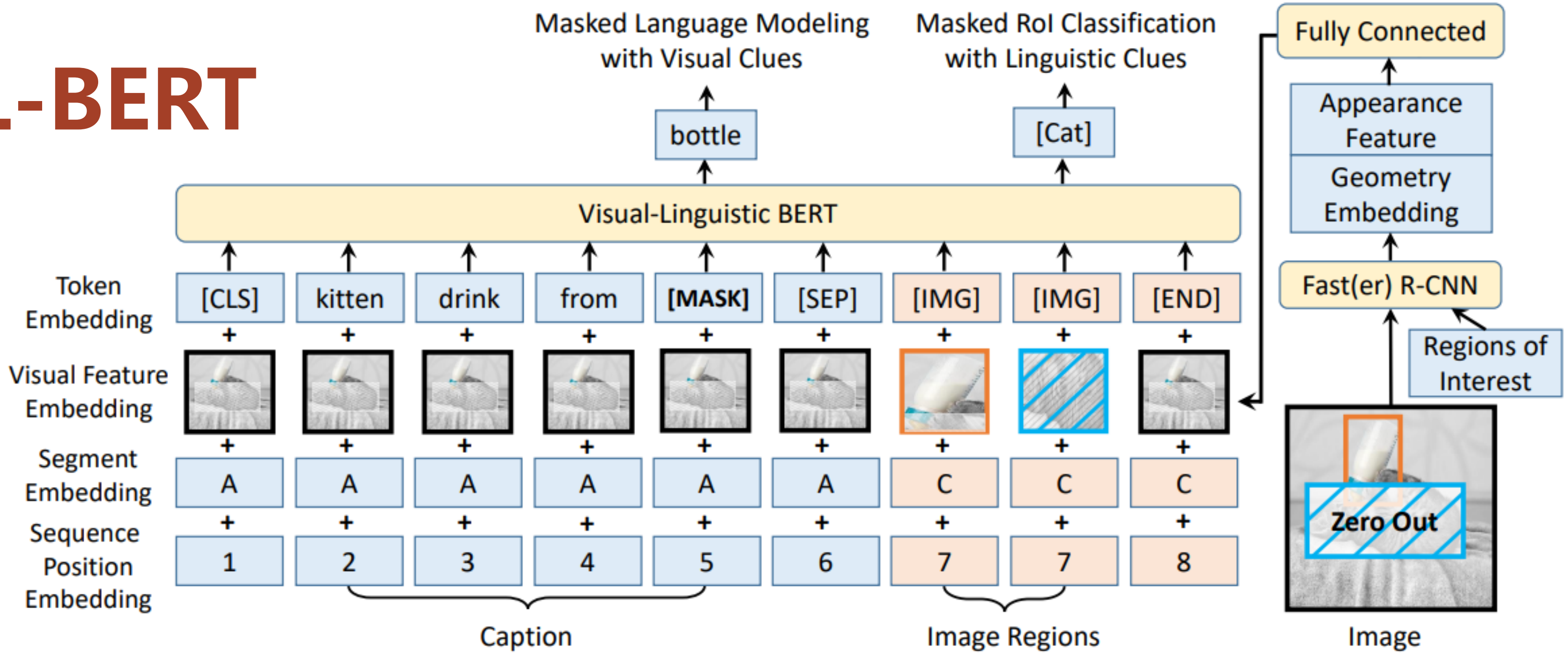
# Embedding in VL-BERT

下游任务：

Visual Commonsense Reasoning

Visual Question Answering (VQA)

Referring Expression task




Visual Feature Embedding： concat(Appearance Feature, Geometry Embedding)

Appearance Feature: For visual element → RoI feature extracted by Fast(er) R-CNN

For non-visual elment → feature extracted on the whole image

Geometry Embedding: $\left(\frac{x_{\mathrm{LT}}}{W}, \frac{y_{\mathrm{LT}}}{H}, \frac{x_{\mathrm{RB}}}{W}, \frac{h_{\mathrm{RB}}}{H}\right)$,

Segment Embedding : Three types of segment, A, B, C, which means different input format.

- For example, for input format of , A denotes Question, B denotes Answer, and C denotes Image. For input format of , A denotes Caption, and C denotes Image.

# Pre-Training Task in VL-BERT

Task#1: Masked Language Modeling with Visual Clues

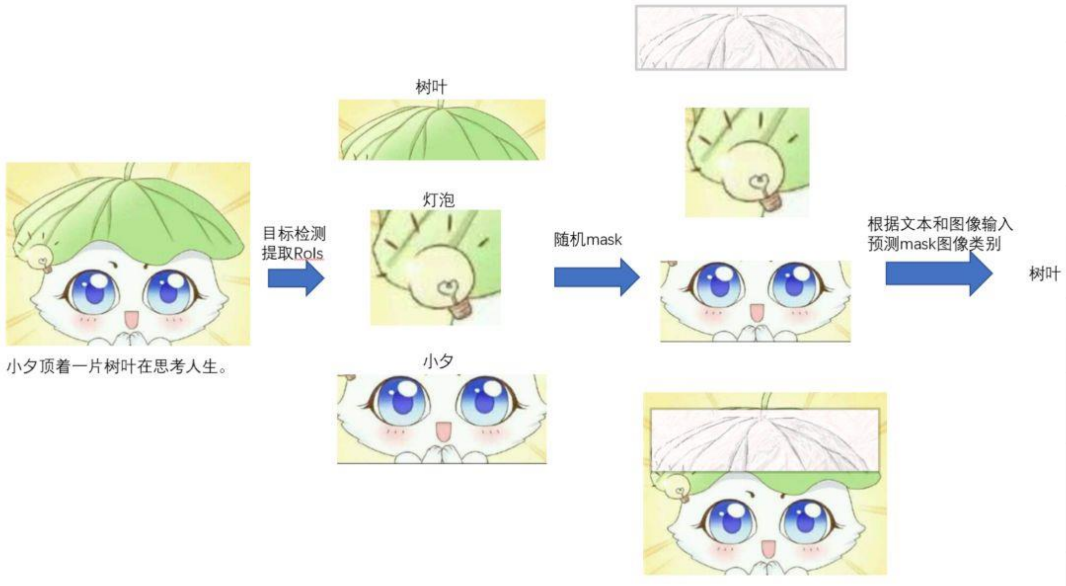
· Predict the masked words ← unmasked words and the visual features.

· Final output feature([mask]) → classifier → word (cross-entropy loss)

Task#2: Masked RoI Classification with Linguistic Clues

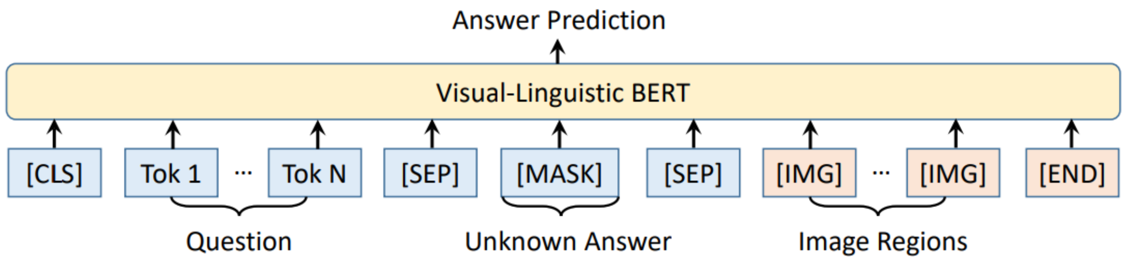
· The pixels laid in the masked RoI are set as zeros before applying Fast R-CNN

· Final output feature([masked RoI]) → classifier → object category classification

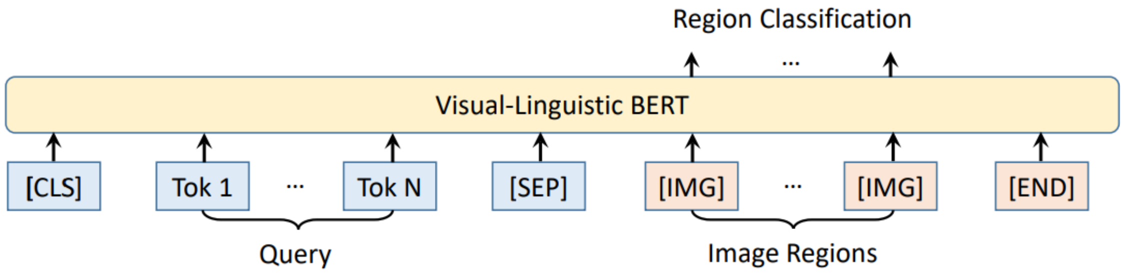



Dataset for pre-training : Conceptual Captions

# FINE-TUNING ON DOWNSTREAM TASKS




(b) Input and output format for Visual Question Answering (VQA) dataset




(c) Input and output format for Referring Expression task on RefCOCO+ dataset

| Model | test-dev | test-std |
|---|---|---|
| BUTD (Anderson et al., 2018) | 65.32 | 65.67 |
| ViLBERT (Lu et al., 2019)† | 70.55 | 70.92 |
| VisualBERT (Li et al., 2019b)† | 70.80 | 71.00 |
| LXMERT (Tan & Bansal, 2019)† | 72.42 | 72.54 |
| $\text{VL-BERT}_{\text{BASE}}$ w/o pre-training | 69.58 | - |
| $\text{VL-BERT}_{\text{BASE}}$ | 71.16 | - |
| $\text{VL-BERT}_{\text{LARGE}}$ | 71.79 | 72.22 |

Table 2: Comparison to the state-of-the-art methods with single model on the VQA dataset. † indicates concurrent works.

| Model | Ground-truth Regions | | | Detected Regions | | |
|---|---|---|---|---|---|---|
| | val | testA | testB | val | testA | testB |
| MAttNet (Yu et al., 2018) | 71.01 | 75.13 | 66.17 | 65.33 | 71.62 | 56.02 |
| ViLBERT (Lu et al., 2019)† | - | - | - | 72.34 | 78.52 | 62.61 |
| $\text{VL-BERT}_{\text{BASE}}$ w/o pre-training | 74.41 | 77.28 | 67.52 | 66.03 | 71.87 | 56.13 |
| $\text{VL-BERT}_{\text{BASE}}$ | 79.88 | 82.40 | 75.01 | 71.60 | 77.72 | 60.99 |
| $\text{VL-BERT}_{\text{LARGE}}$ | 80.31 | 83.62 | 75.45 | 72.59 | 78.57 | 62.30 |

Table 3: Comparison to the state-of-the-art methods with single model on the RefCOCO+ dataset. † indicates concurrent work.

## VQA

## Referring Expression

# 5 Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu[†*] Yutong Lin[†*] Yue Cao[*] Han Hu[*‡] Yixuan Wei[†]
Zheng Zhang Stephen Lin Baining Guo
Microsoft Research Asia
{v-zeliu1,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com

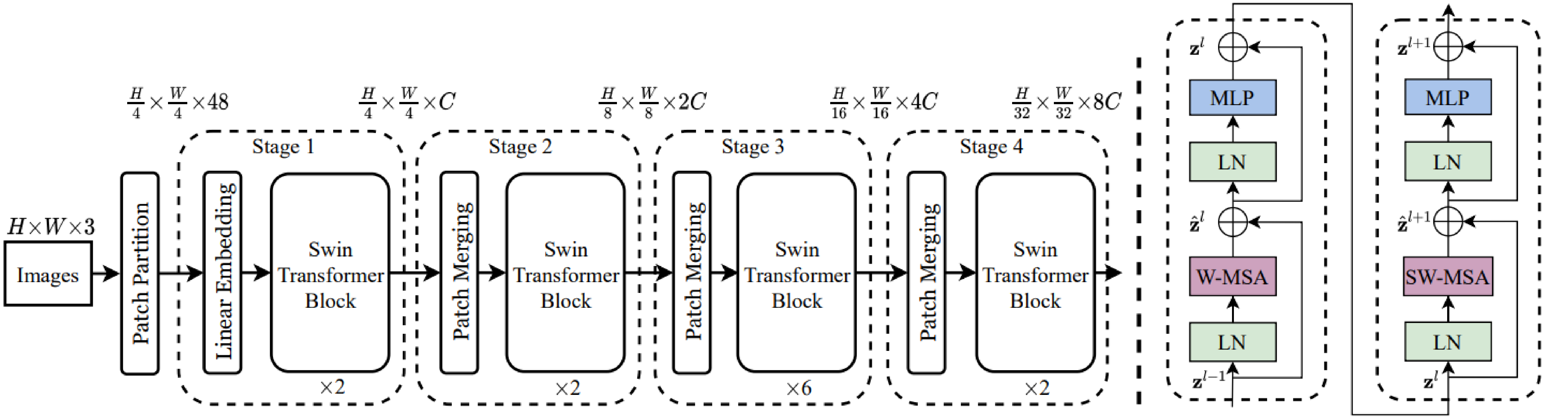
**CVPR 2021**

**Motivation:**

Tokens are all of a fixed scale ,which is unsuitable for vision applications

Higher resolution of pixels in images compared to words in passages of text.

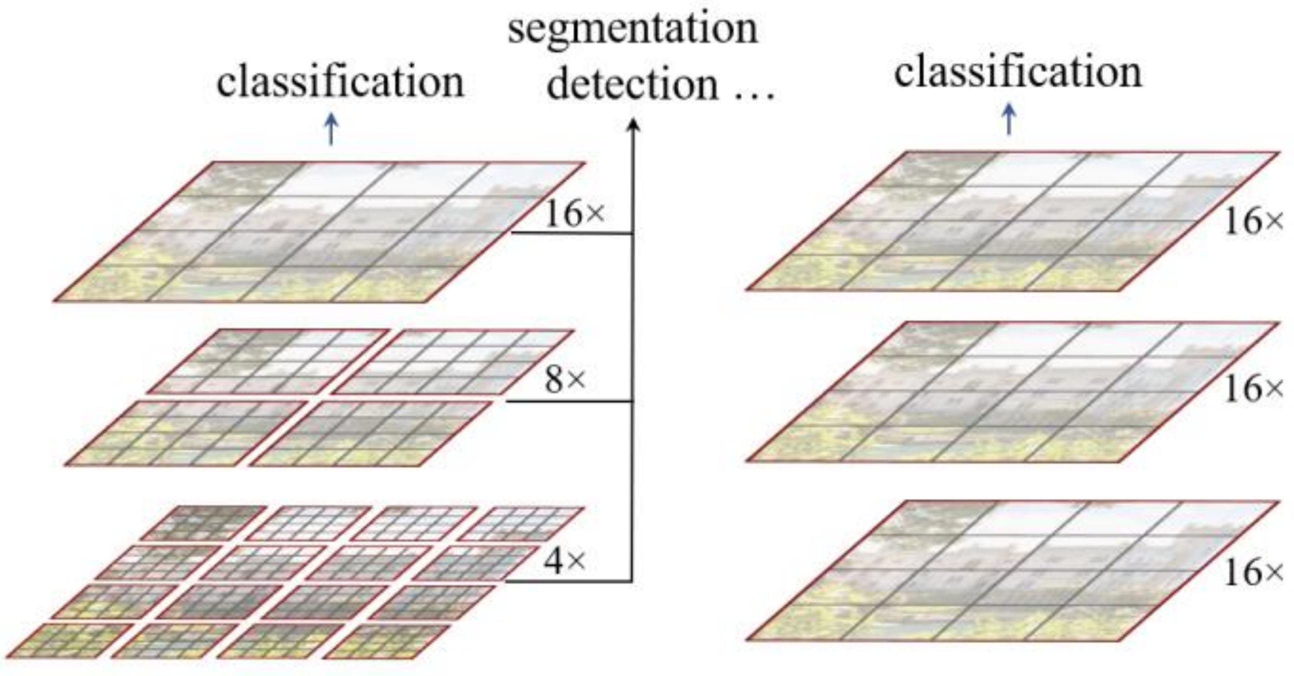
# Overall Architecture




(a) Architecture (b) Two Successive Swin Transformer Blocks

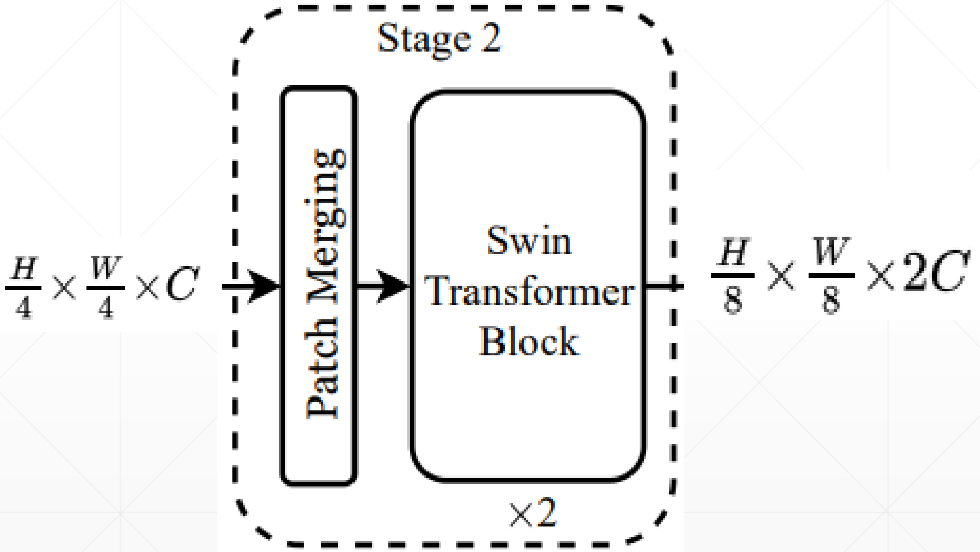
Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

# Hierarchical feature maps → Patch Merging

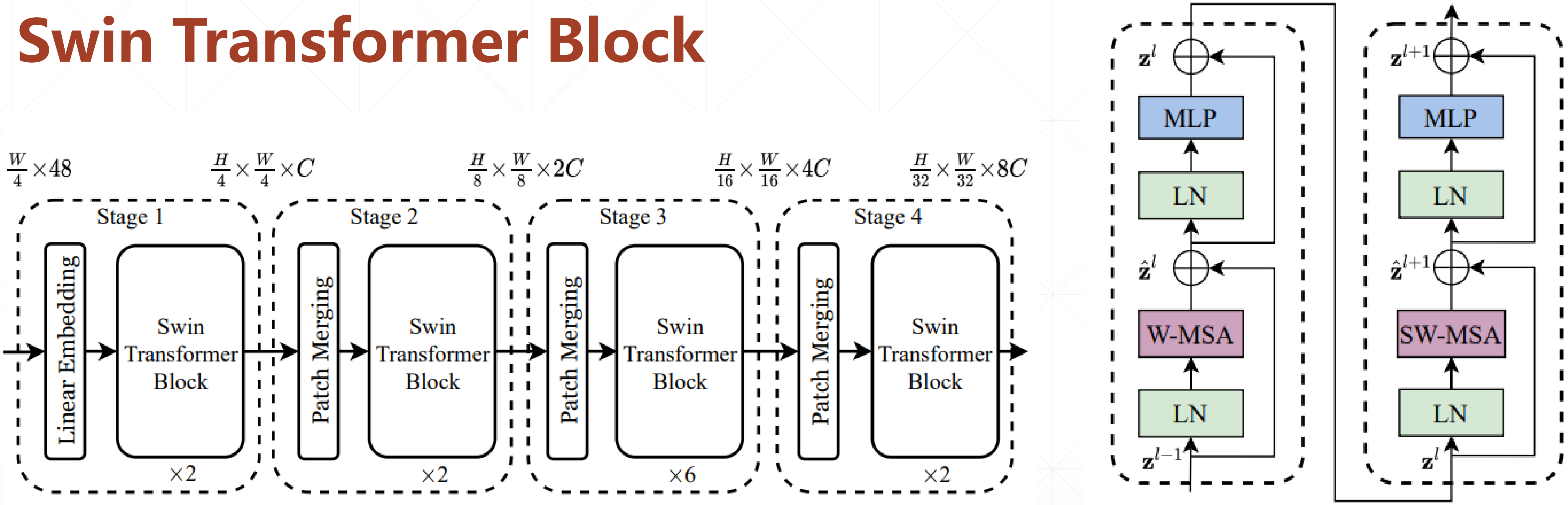



(a) Swin Transformer (ours)

(b) ViT



# Swin Transformer Block




- W-MSA (Window-Multihead Self Attention)
- SW-MSA (Shifted Window-Multihead Self Attention)

# W-MSA and SW-MSA

$$\Omega(\mathrm{MSA}) = 4hwC^2 + 2(hw)^2C, \qquad (1)$$
$$\Omega(\mathrm{W\text{-}MSA}) = 4hwC^2 + 2M^2hwC, \qquad (2)$$

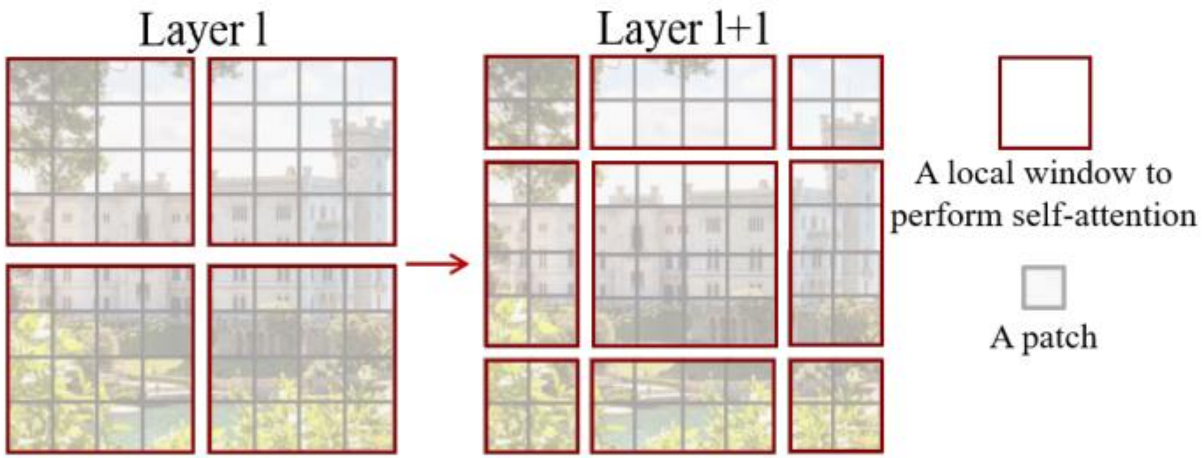
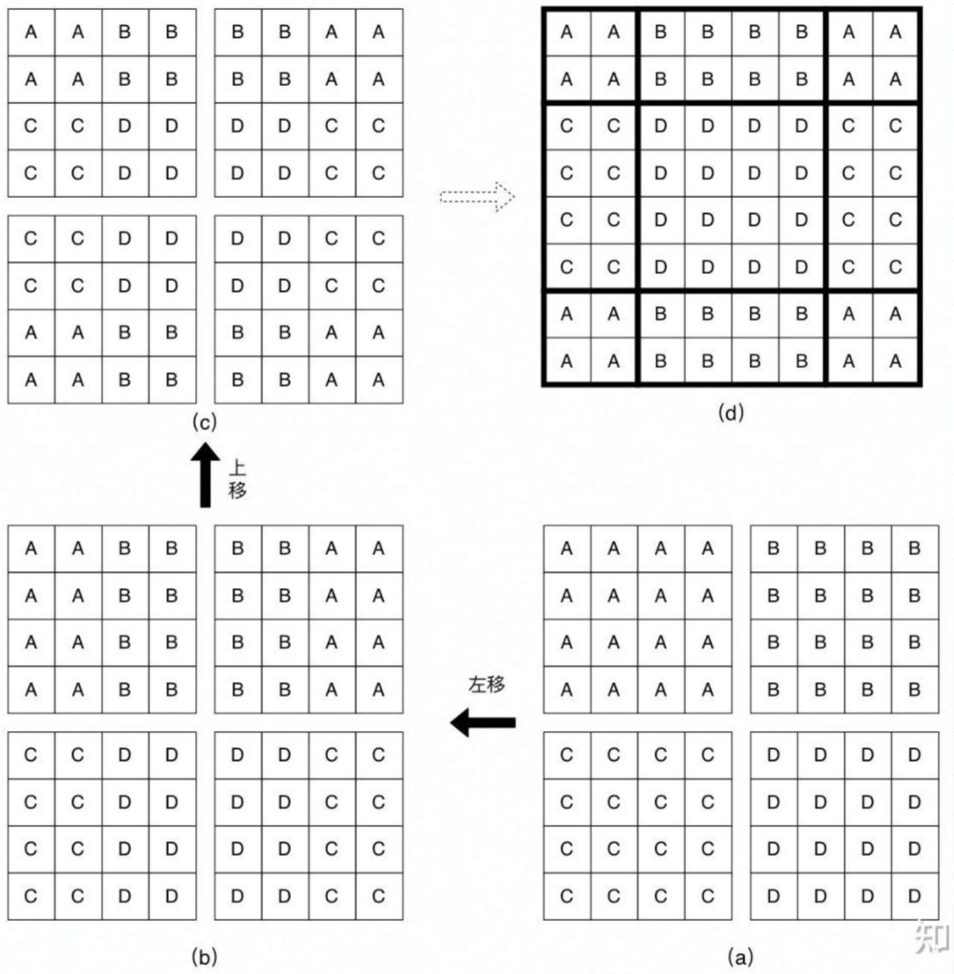



Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer $l$ (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer $l$, providing connections among them.



# Efficient batch computation for shifted configuration

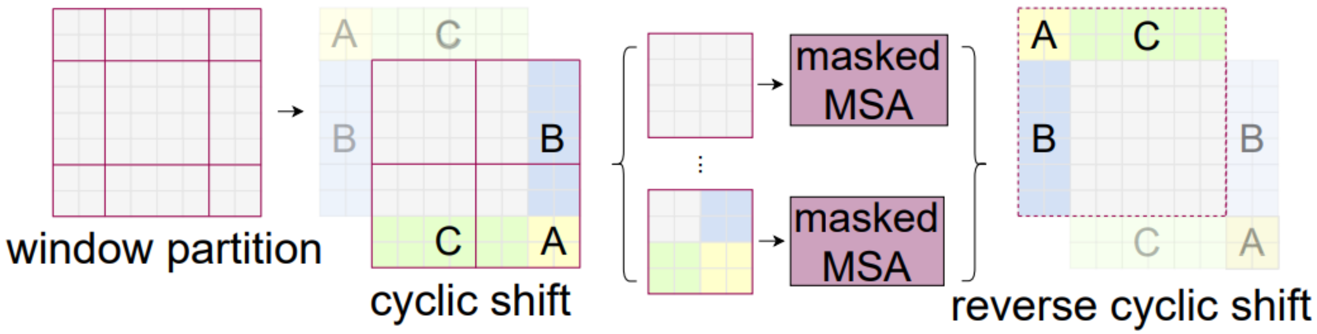



Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

| method | MSA in a stage (ms) S1 | S2 | S3 | S4 | Arch. (FPS) T | S | B |
|---|---|---|---|---|---|---|---|
| sliding window (naive) | 122.5 | 38.3 | 12.1 | 7.6 | 183 | 109 | 77 |
| sliding window (kernel) | 7.6 | 4.7 | 2.7 | 1.8 | 488 | 283 | 187 |
| Performer [13] | 4.8 | 2.8 | 1.8 | 1.5 | 638 | 370 | 241 |
| window (w/o shifting) | 2.8 | 1.7 | 1.2 | 0.9 | 770 | 444 | 280 |
| shifted window (padding) | 3.3 | 2.3 | 1.9 | 2.2 | 670 | 371 | 236 |
| shifted window (cyclic) | 3.0 | 1.9 | 1.3 | 1.0 | 755 | 437 | 278 |

Table 5. Real speed of different self-attention computation methods and implementations on a V100 GPU.

| | Backbone | ImageNet top-1 | top-5 | COCO $AP^{box}$ | $AP^{mask}$ | ADE20k mIoU |
|---|---|---|---|---|---|---|
| sliding window | Swin-T | 81.4 | 95.6 | 50.2 | 43.5 | 45.8 |
| Performer [13] | Swin-T | 79.0 | 94.2 | - | - | - |
| shifted window | Swin-T | 81.3 | 95.6 | 50.5 | 43.7 | 46.1 |

Table 6. Accuracy of Swin Transformer using different methods for self-attention computation on three benchmarks.

# Relative position bias

$$Attention(Q, K, V) = SoftMax(QK^T/\sqrt{d} + B)V$$

具体思想参考UniLMV2

```
def get_relative_distances(window_size):
    indices = torch.tensor(np.array([[x, y] for x in range(window_size) for y in range(window_size)]))
    distances = indices[None, :, :] - indices[:, None, :]
    return distances
```

# Architecture Variants

- Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers =$\{2, 2, 18, 2\}$

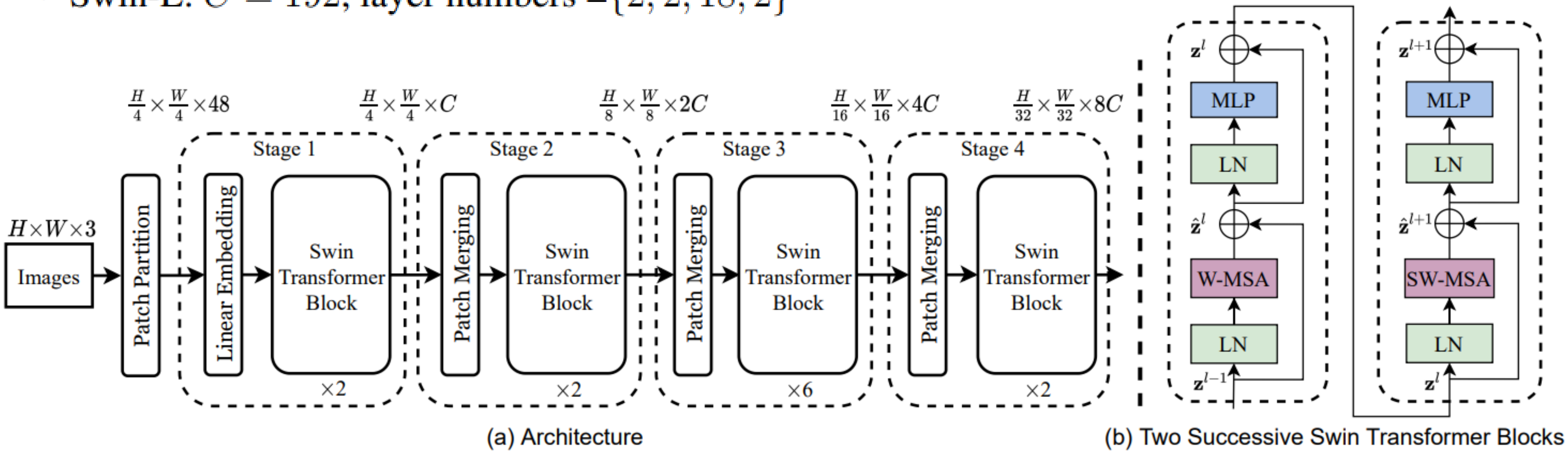



Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

# Experiments

| method | image size | #param. | FLOPs | throughput (image / s) | ImageNet top-1 acc. |
|---|---|---|---|---|---|
| **(a) Regular ImageNet-1K trained models** | | | | | |
| RegNetY-4G [47] | $224^2$ | 21M | 4.0G | 1156.7 | 80.0 |
| RegNetY-8G [47] | $224^2$ | 39M | 8.0G | 591.6 | 81.7 |
| RegNetY-16G [47] | $224^2$ | 84M | 16.0G | 334.7 | 82.9 |
| EffNet-B3 [57] | $300^2$ | 12M | 1.8G | 732.1 | 81.6 |
| EffNet-B4 [57] | $380^2$ | 19M | 4.2G | 349.4 | 82.9 |
| EffNet-B5 [57] | $456^2$ | 30M | 9.9G | 169.1 | 83.6 |
| EffNet-B6 [57] | $528^2$ | 43M | 19.0G | 96.9 | 84.0 |
| EffNet-B7 [57] | $600^2$ | 66M | 37.0G | 55.1 | 84.3 |
| ViT-B/16 [19] | $384^2$ | 86M | 55.4G | 85.9 | 77.9 |
| ViT-L/16 [19] | $384^2$ | 307M | 190.7G | 27.3 | 76.5 |
| DeiT-S [60] | $224^2$ | 22M | 4.6G | 940.4 | 79.8 |
| DeiT-B [60] | $224^2$ | 86M | 17.5G | 292.3 | 81.8 |
| DeiT-B [60] | $384^2$ | 86M | 55.4G | 85.9 | 83.1 |
| Swin-T | $224^2$ | 29M | 4.5G | 755.2 | 81.3 |
| Swin-S | $224^2$ | 50M | 8.7G | 436.9 | 83.0 |
| Swin-B | $224^2$ | 88M | 15.4G | 278.1 | 83.3 |
| Swin-B | $384^2$ | 88M | 47.0G | 84.7 | 84.2 |
| **(b) ImageNet-22K pre-trained models** | | | | | |
| method | image size | #param. | FLOPs | throughput (image / s) | ImageNet top-1 acc. |
| R-101x3 [37] | $384^2$ | 388M | 204.6G | - | 84.4 |
| R-152x4 [37] | $480^2$ | 937M | 840.5G | - | 85.4 |
| ViT-B/16 [19] | $384^2$ | 86M | 55.4G | 85.9 | 84.0 |
| ViT-L/16 [19] | $384^2$ | 307M | 190.7G | 27.3 | 85.2 |
| Swin-B | $224^2$ | 88M | 15.4G | 278.1 | 85.2 |
| Swin-B | $384^2$ | 88M | 47.0G | 84.7 | 86.0 |
| Swin-L | $384^2$ | 197M | 103.9G | 42.1 | 86.4 |

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [65] and a V100 GPU, following [60].

**(a) Various frameworks**

| Method | Backbone | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | #param. | FLOPs | FPS |
|---|---|---|---|---|---|---|---|
| Cascade Mask R-CNN | R-50 | 46.3 | 64.3 | 50.5 | 82M | 739G | 18.0 |
| | Swin-T | **50.5** | **69.3** | **54.9** | 86M | 745G | 15.3 |
| ATSS | R-50 | 43.5 | 61.9 | 47.0 | 32M | 205G | 28.3 |
| | Swin-T | **47.2** | **66.5** | **51.3** | 36M | 215G | 22.3 |
| RepPointsV2 | R-50 | 46.5 | 64.6 | 50.3 | 42M | 274G | 13.6 |
| | Swin-T | **50.0** | **68.5** | **54.2** | 45M | 283G | 12.0 |
| Sparse R-CNN | R-50 | 44.5 | 63.4 | 48.2 | 106M | 166G | 21.0 |
| | Swin-T | **47.9** | **67.3** | **52.3** | 110M | 172G | 18.4 |

**(b) Various backbones w. Cascade Mask R-CNN**

| | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ | param | FLOPs | FPS |
|---|---|---|---|---|---|---|---|---|---|
| DeiT-S† | 48.0 | 67.2 | 51.7 | 41.4 | 64.2 | 44.3 | 80M | 889G | 10.4 |
| R50 | 46.3 | 64.3 | 50.5 | 40.1 | 61.7 | 43.4 | 82M | 739G | 18.0 |
| Swin-T | **50.5** | **69.3** | **54.9** | **43.7** | **66.6** | **47.1** | 86M | 745G | 15.3 |
| X101-32 | 48.1 | 66.5 | 52.4 | 41.6 | 63.9 | 45.2 | 101M | 819G | 12.8 |
| Swin-S | **51.8** | **70.4** | **56.3** | **44.7** | **67.9** | **48.5** | 107M | 838G | 12.0 |
| X101-64 | 48.3 | 66.4 | 52.3 | 41.7 | 64.0 | 45.1 | 140M | 972G | 10.4 |
| Swin-B | **51.9** | **70.9** | **56.5** | **45.0** | **68.4** | **48.7** | 145M | 982G | 11.6 |

**(c) System-level Comparison**

| Method | mini-val $AP^{box}$ | mini-val $AP^{mask}$ | test-dev $AP^{box}$ | test-dev $AP^{mask}$ | #param. | FLOPs |
|---|---|---|---|---|---|---|
| RepPointsV2* [11] | - | - | 52.1 | - | - | - |
| GCNet* [6] | 51.8 | 44.7 | 52.3 | 45.4 | - | 1041G |
| RelationNet++* [12] | - | - | 52.7 | - | - | - |
| SpineNet-190 [20] | 52.6 | - | 52.8 | - | 164M | 1885G |
| ResNeSt-200* [75] | 52.5 | - | 53.3 | 47.1 | - | - |
| EfficientDet-D7 [58] | 54.4 | - | 55.1 | - | 77M | 410G |
| DetectoRS* [45] | - | - | 55.7 | 48.5 | - | - |
| YOLOv4 P7* [3] | - | - | 55.8 | - | - | - |
| Copy-paste [25] | 55.9 | 47.2 | 56.0 | 47.4 | 185M | 1440G |
| X101-64 (HTC++) | 52.3 | 46.0 | - | - | 155M | 1033G |
| Swin-B (HTC++) | 56.4 | 49.1 | - | - | 160M | 1043G |
| Swin-L (HTC++) | 57.1 | 49.5 | 57.7 | 50.2 | 284M | 1470G |
| Swin-L (HTC++)* | **58.0** | **50.4** | **58.7** | **51.1** | 284M | - |

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

| ADE20K Method | Backbone | val mIoU | test score | #param. | FLOPs | FPS |
|---|---|---|---|---|---|---|
| DANet [22] | ResNet-101 | 45.2 | - | 69M | 1119G | 15.2 |
| DLab.v3+ [10] | ResNet-101 | 44.1 | - | 63M | 1021G | 16.0 |
| ACNet [23] | ResNet-101 | 45.9 | 38.5 | - | | |
| DNL [68] | ResNet-101 | 46.0 | 56.2 | 69M | 1249G | 14.8 |
| OCRNet [70] | ResNet-101 | 45.3 | 56.0 | 56M | 923G | 19.3 |
| UperNet [66] | ResNet-101 | 44.9 | - | 86M | 1029G | 20.1 |
| OCRNet [70] | HRNet-w48 | 45.7 | - | 71M | 664G | 12.5 |
| DLab.v3+ [10] | ResNeSt-101 | 46.9 | 55.1 | 66M | 1051G | 11.9 |
| DLab.v3+ [10] | ResNeSt-200 | 48.4 | - | 88M | 1381G | 8.1 |
| SETR [78] | T-Large‡ | 50.3 | 61.7 | 308M | - | - |
| UperNet | DeiT-S† | 44.0 | - | 52M | 1099G | 16.2 |
| UperNet | Swin-T | 46.1 | - | 60M | 945G | 18.5 |
| UperNet | Swin-S | 49.3 | - | 81M | 1038G | 15.2 |
| UperNet | Swin-B‡ | 51.6 | - | 121M | 1841G | 8.7 |
| UperNet | Swin-L‡ | **53.5** | **62.8** | 234M | 3230G | 6.2 |

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

# Review

## Paper List :

**Transformer**

Attention is all you need

**Visual Transformer**

AN IMAGE IS WORTH 16X16 WORDS:TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

**SETR**

Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers

**Swin Transformer**

Hierarchical Vision Transformer using Shifted Windows

**BERT**

Pre-training of Deep Bidirectional Transformers for Language Understanding

**VL-BERT**

PRE-TRAINING OF GENERIC VISUAL LINGUISTIC REPRESENTATIONS

# Thank you