



# 注意力机制及其应用

彭小江 副教授

深圳技术大学 大数据与互联网学院

邮箱: [pengxiaojiang@sztu.edu.cn](mailto:pengxiaojiang@sztu.edu.cn)

个人主页: [pengxj.github.io](https://pengxj.github.io)

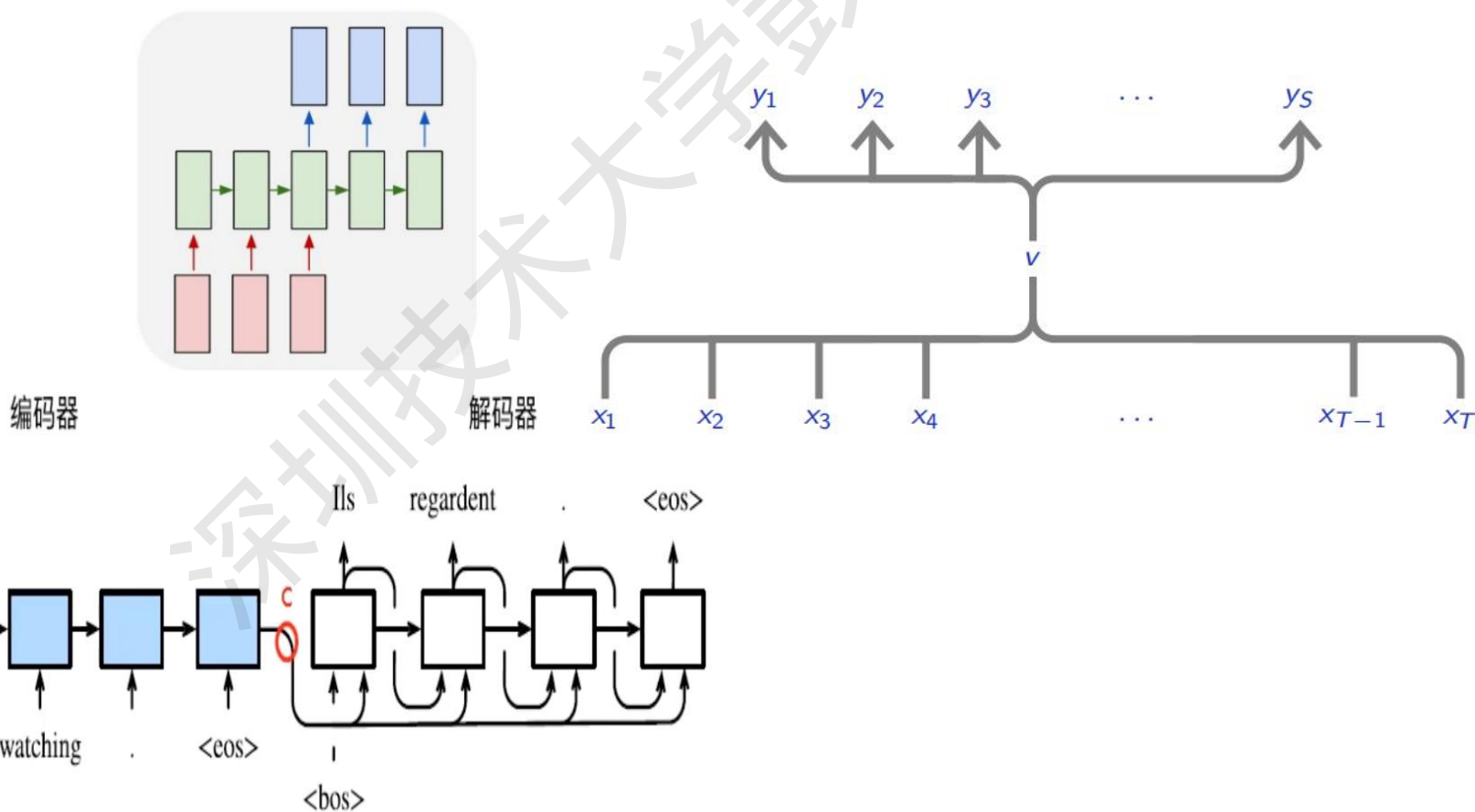


# 主要内容

- 时序模型中的注意力机制
- Transformer
- Vision Transformer
- ViT表情识别及ViT变种

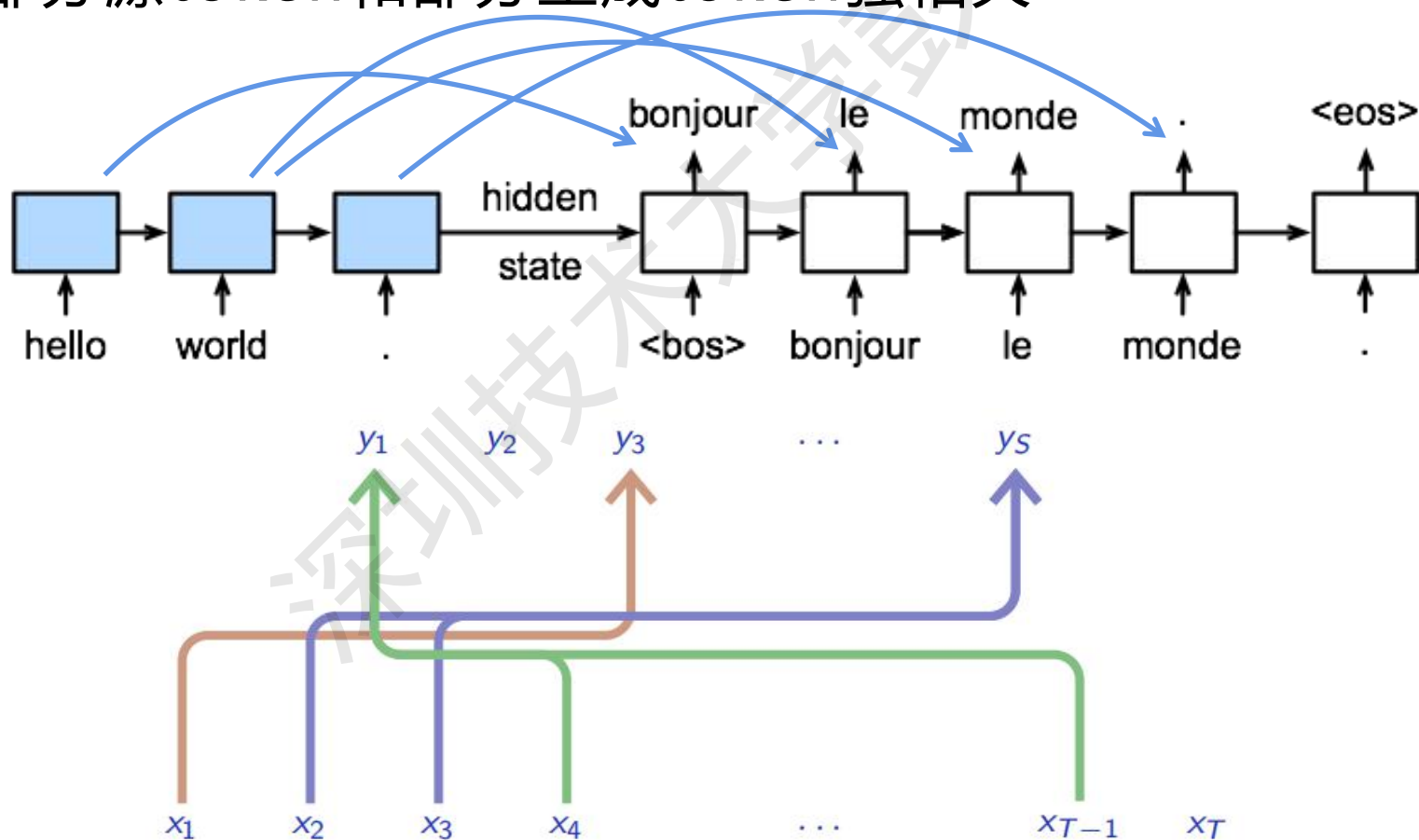
# Seq2Seq典型结构

- 应用：文本翻译、看图说话
- 输入编码成一个隐含状态然后进行编码输出



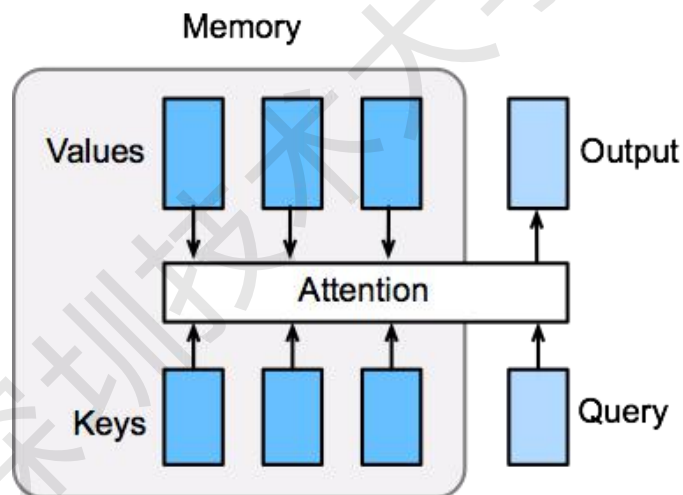
# 动机

- 每个生成的token可能与不同的源token相关
- 部分源token和部分生成token强相关



# 注意力层作用

- 注意力层明确选择相关信息
  - 它的存储器（memory）由“键值对”组成
  - 键和查询越相似，则输出的值越相近



# 注意力聚合存储中向量

- 假设“一条询问”为  $\mathbf{q} \in \mathbb{R}^{d_q}$ ，存储器为  $(\mathbf{k}_1, \mathbf{v}_1), (\dots), (\mathbf{k}_n, \mathbf{v}_n)$ ； $\mathbf{k}_i \in \mathbb{R}^{d_k}$ ， $\mathbf{v}_i \in \mathbb{R}^{d_v}$

- 计算  $n$  分数  $a_1, \dots, a_n$ ； $a_i = \alpha(\mathbf{q}, \mathbf{k}_i)$

- 使用 softmax 获得注意力

改变 $\alpha$ 可以获得不同的注意力层

$$b_1, \dots, b_n = \text{softmax}(a_1, \dots, a_n)$$

- 输出是值的加权和

$$\mathbf{o} = \sum_{i=1}^n b_i \mathbf{v}_i$$

# 点乘注意力

- 假设询问的长度与值相同  $\mathbf{q}, \mathbf{k}_i \in \mathbb{R}^d$

$$\alpha(\mathbf{q}, \mathbf{k}) = \langle \mathbf{q}, \mathbf{k} \rangle / \sqrt{d}$$

- 向量化版本

- $m$  个询问  $\mathbf{Q} \in \mathbb{R}^{m \times d}$  和  $n$  个键  $\mathbf{K} \in \mathbb{R}^{n \times d}$

$$\alpha(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^T / \sqrt{d}$$

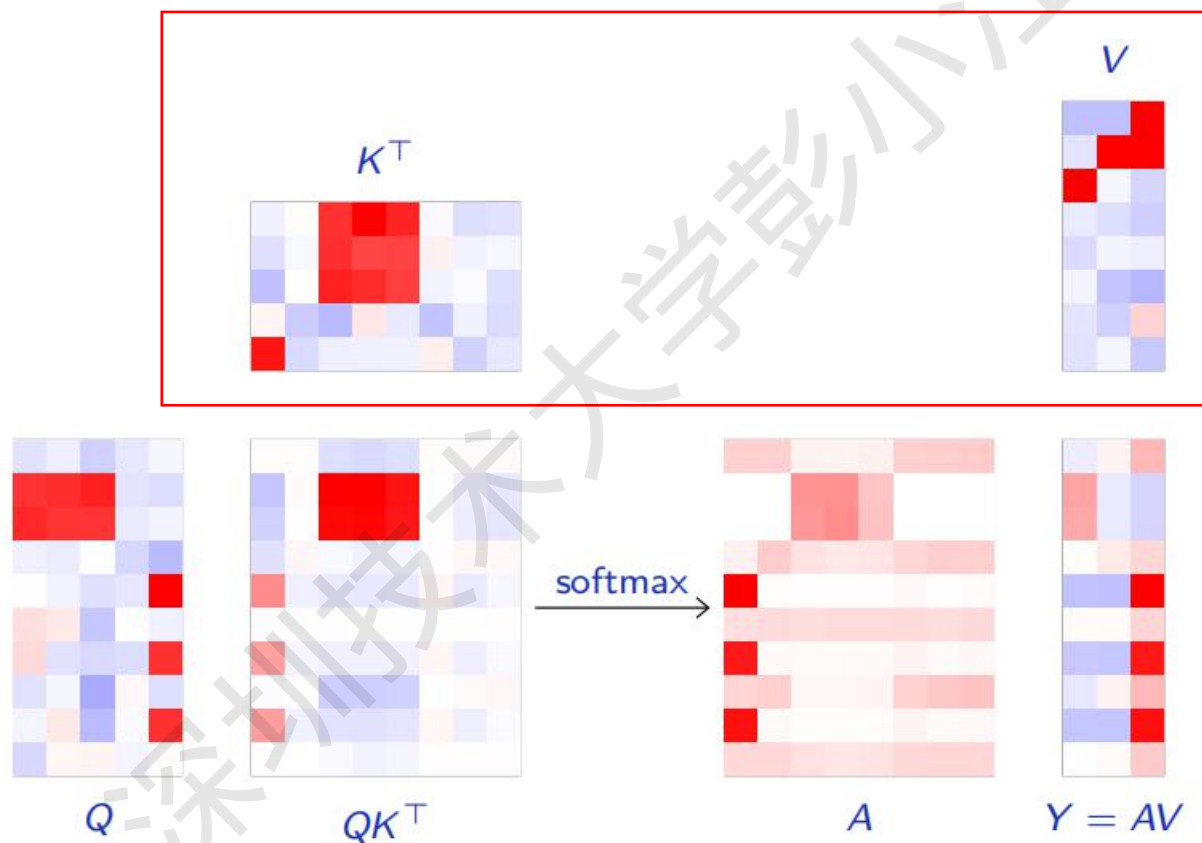
# 多层感知注意力

- 可学习的参数  $\mathbf{W}_k \in \mathbb{R}^{h \times d_k}$ ,  $\mathbf{W}_q \in \mathbb{R}^{h \times d_q}$ ,  $\mathbf{v} \in \mathbb{R}^h$

$$\alpha(\mathbf{k}, \mathbf{q}) = \mathbf{v}^T \tanh(\mathbf{W}_k \mathbf{k} + \mathbf{W}_q \mathbf{q})$$

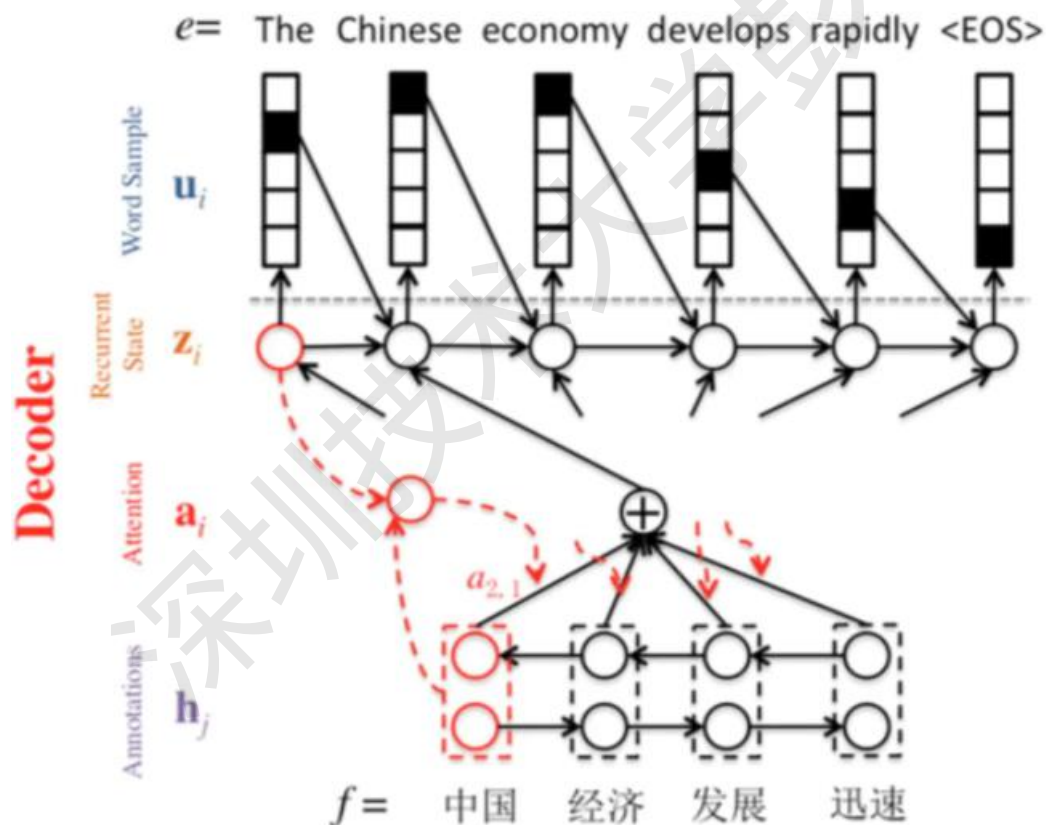


# 注意力的整体效果



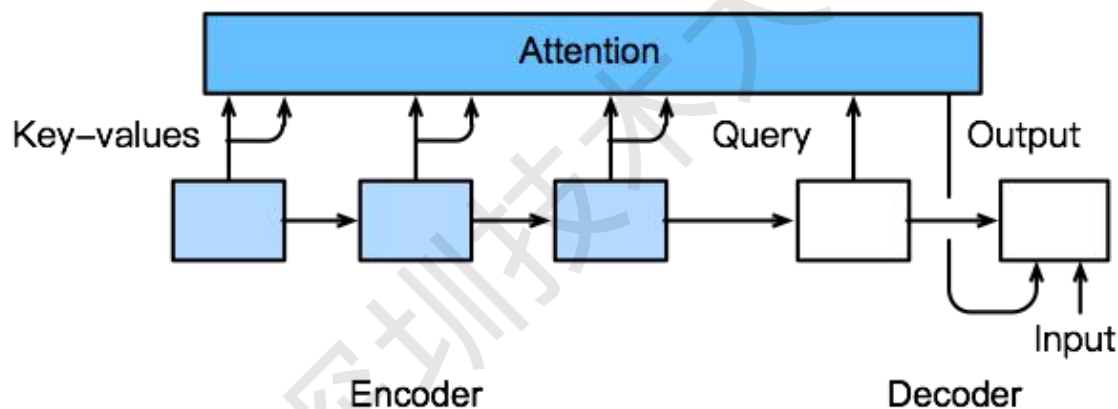
# Seq2Seq中的注意力机制应用

- 在解码器解码时候加入来自注意力模块的向量



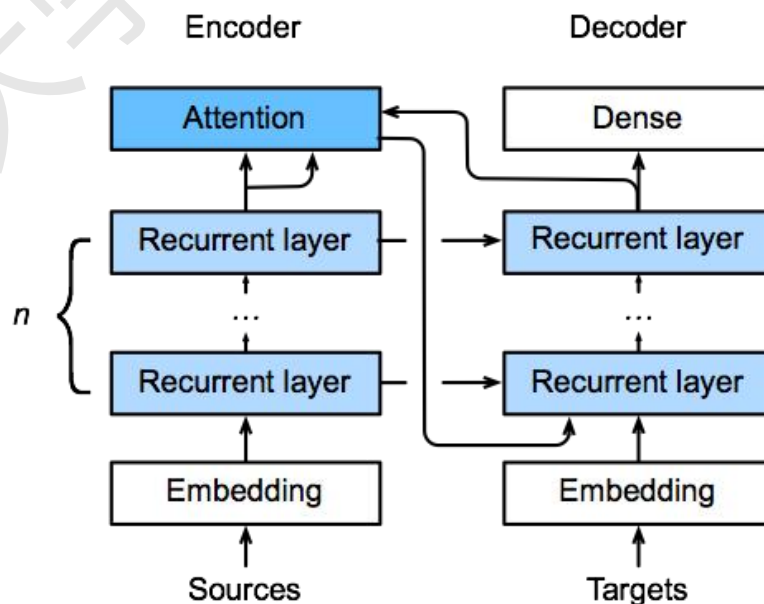
# 模型架构

- 添加额外的注意层以编码器的输出作为存储器
- 注意力的输出用作解码器的输入



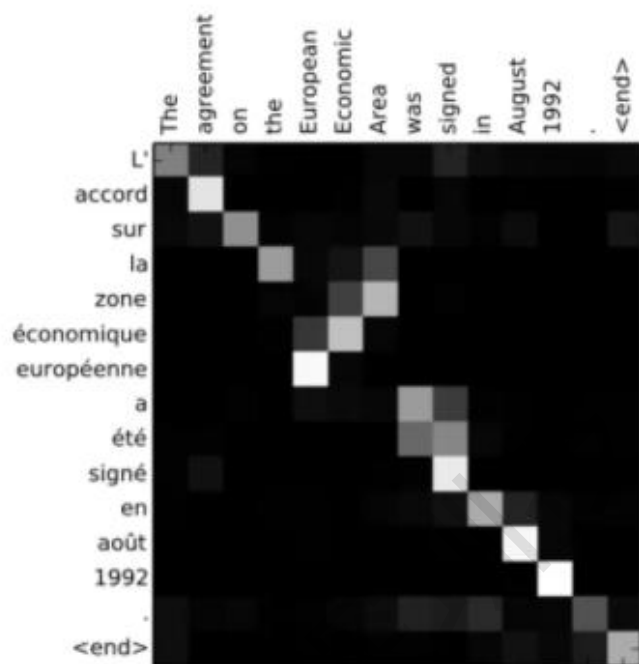
# 编码器—解码器上的注意力机制

- 使用编码器中最后一个循环神经网络层的输出
- 然后，注意力输出与嵌入输出拼接，以输入解码器中的第一个循环神经网络层

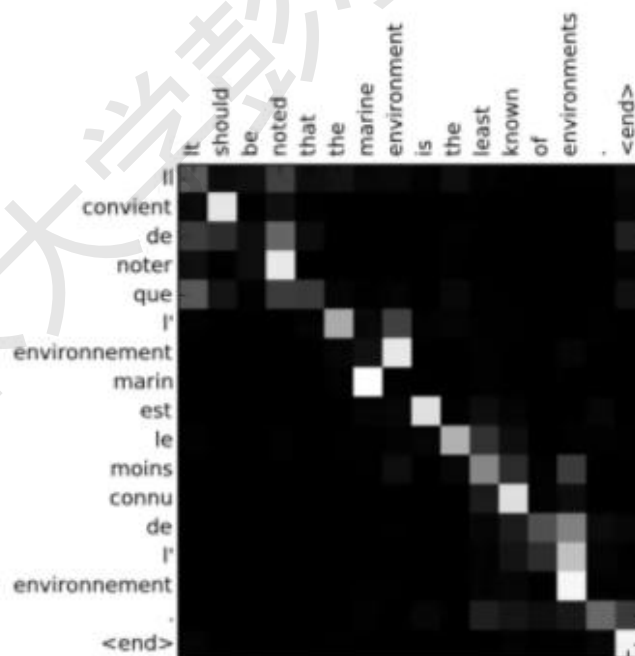


# 注意力的作用

## ● 输入-输出对齐效果



(a)



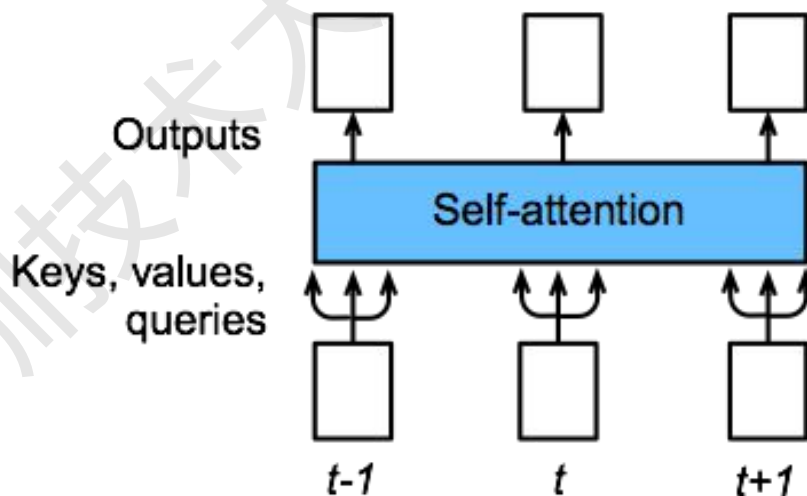
(b)

# 变换器模型 (Transformer)



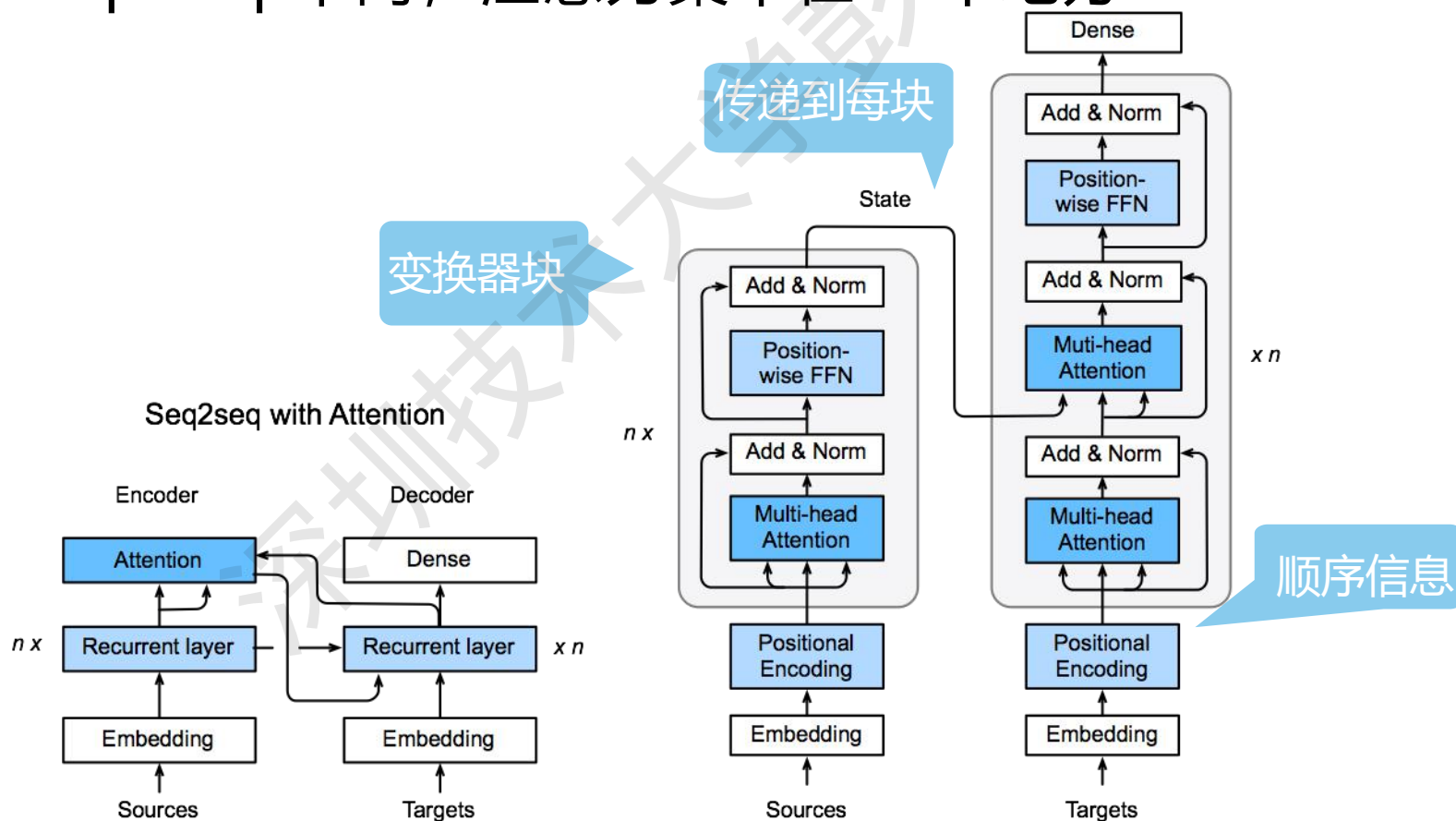
# 自注意力机制

- 要使用  $n$  个输入生成  $n$  个输出，我们可以将每个输入复制为键（key），值（value）和查询（query）中
- 不保留顺序信息
- 并行计算



# Transformer模型架构

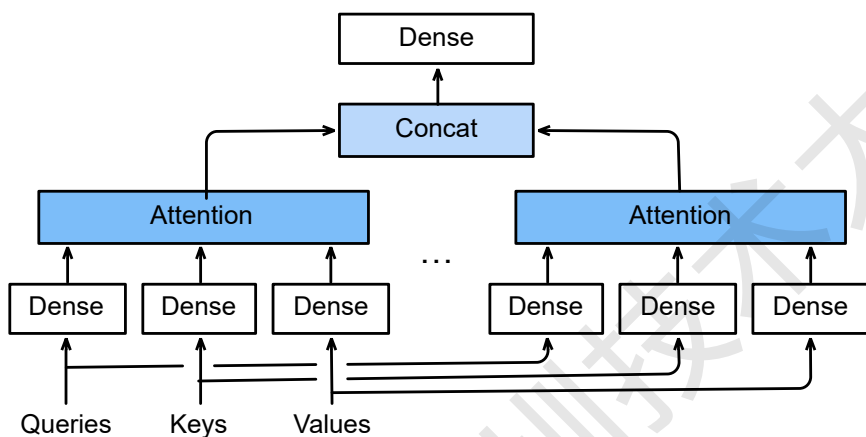
- 它是一个编码器 - 解码器架构
- 与 seq2seq 不同，注意力集中在 3 个地方





# 多头注意力机制 (Multi-head Attention)

- 有时候注意力方式可能有多种，且都是合理的，这时候可以使用多个注意力模块

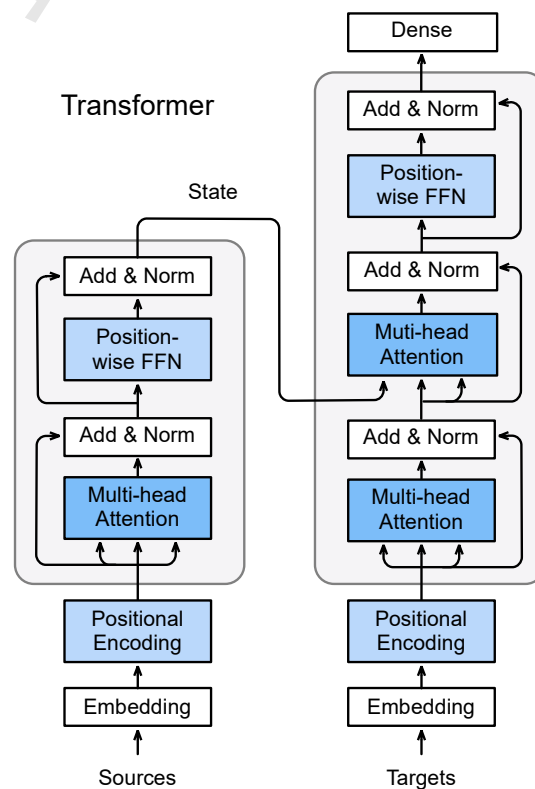


$$\mathbf{W}_q^{(i)} \in \mathbb{R}^{p_q \times d_q}, \mathbf{W}_k^{(i)} \in \mathbb{R}^{p_k \times d_k}, \text{ and } \mathbf{W}_v^{(i)} \in \mathbb{R}^{p_v \times d_v}$$

$$\mathbf{o}^{(i)} = \text{attention}(\mathbf{W}_q^{(i)} \mathbf{q}, \mathbf{W}_k^{(i)} \mathbf{k}, \mathbf{W}_v^{(i)} \mathbf{v})$$

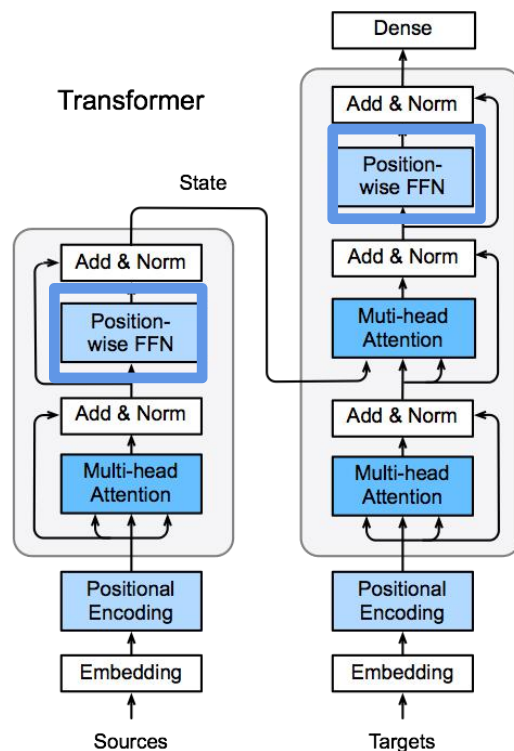
$$\text{for } i = 1, \dots, h$$

$$\mathbf{o} = \mathbf{W}_o \begin{bmatrix} \mathbf{o}^{(1)} \\ \vdots \\ \mathbf{o}^{(h)} \end{bmatrix}$$



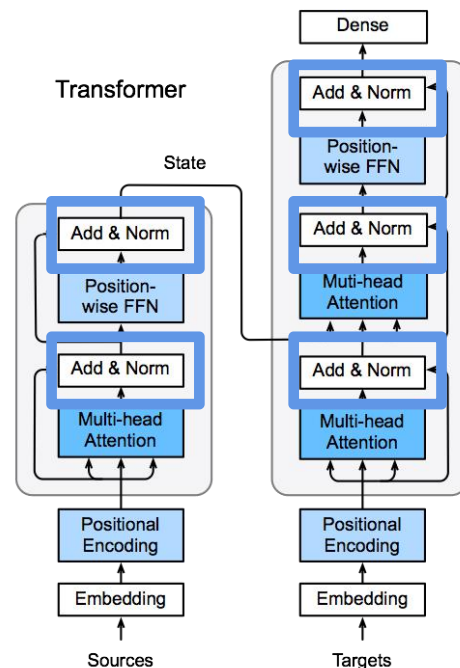
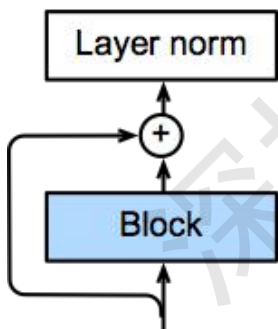
# 位置前馈网络

- 将输入（批量大小，序列长度，特征大小）重新整形为（批量\*序列长度，特征大小）
- 用两层 MLP



# Res结构和归一化

- 层规范 (Layer Norm) 类似于批量规范 (Batch Norm)
- 但是平均值和方差是沿最后一个维度计算的
  - $X.\text{mean}(\text{axis} = -1)$  而不是批量归一化
  - $X.\text{mean}$  中的第一个批次维度 ( $\text{axis} = 0$ )



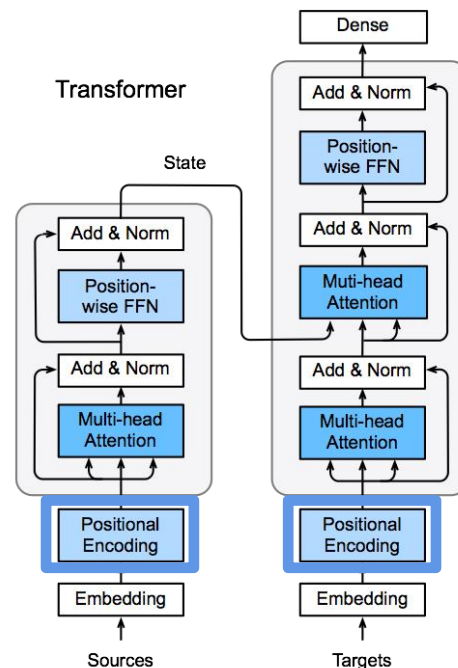
# 位置编码

- 假设嵌入  $X \in \mathbb{R}^{l \times d}$  输出的形状 (序列长度, 嵌入维度)
- 创建  $P \in \mathbb{R}^{l \times d}$

$$P_{i,2j} = \sin(i/10000^{2j/d})$$

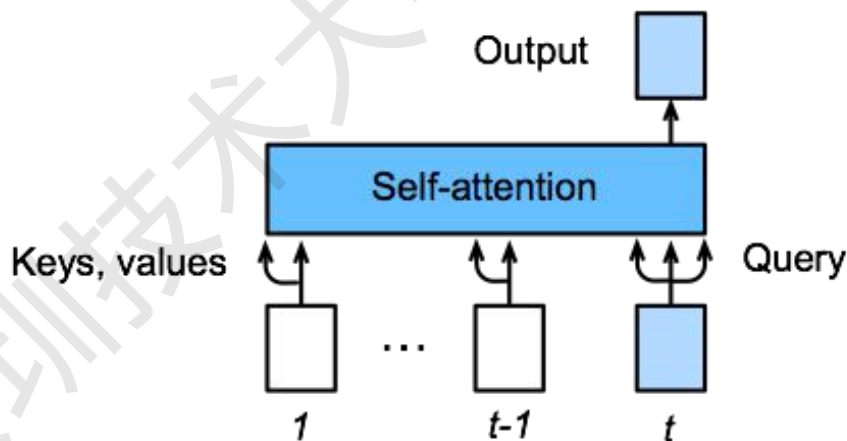
$$P_{i,2j+1} = \cos(i/10000^{2jd})$$

- 输出  $X + P$



# 预测下一单词

- 在时刻  $t$  预测：
  - 用之前输入的键和值
  - 时刻  $t$  输入作为查询，以及键和值，以预测输出



# BERT

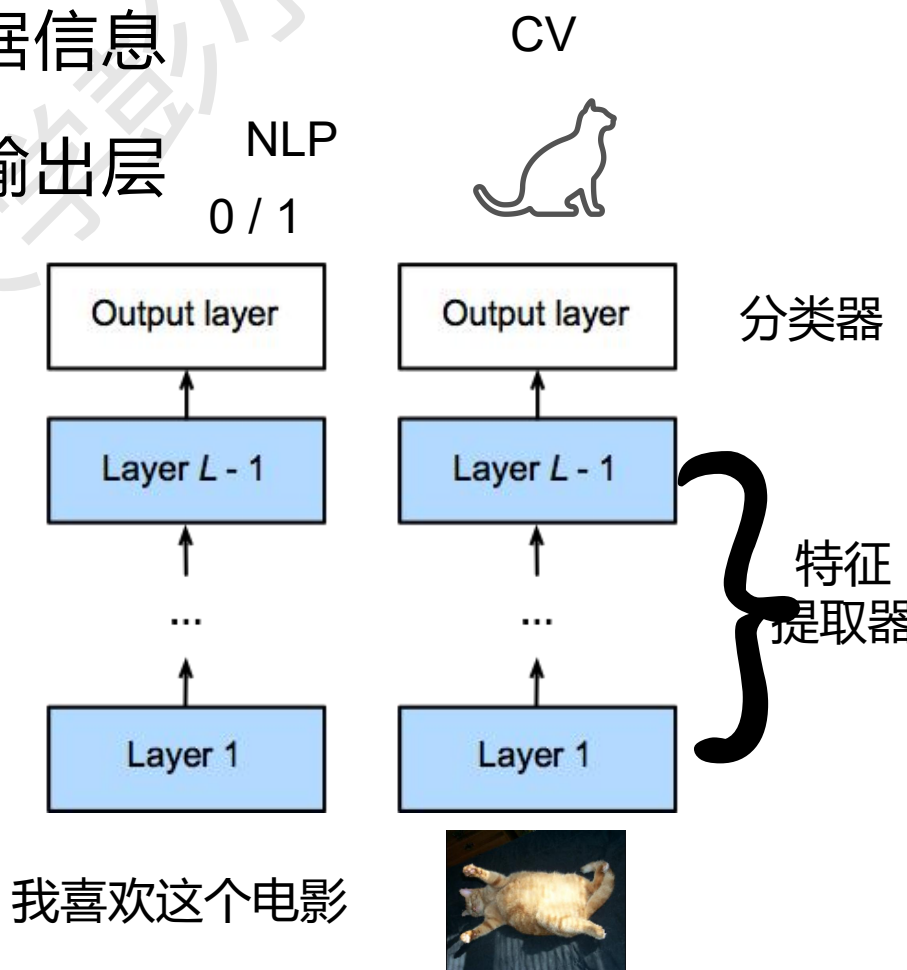
- 全称Bidirectional Encoder Representation from Transformers, 是一个预训练的语言表征模型
- 类似CV任务, 通常使用ImageNet或者更大的数据库进行预训练模型, 然后根据自己任务进行微调, 该过程叫做迁移学习 (Transfer learning)

# NLP中的迁移学习 (Transfer Learning)

- 使用预先训练的模型为新任务提取单词/句子功能
  - 例如：word2vec 或语言模型
- 通常不更新预先训练的模型的权重
- 需要构建一个新模型来捕获新任务所需的信息
  - Word2vec 忽略顺序信息，这个语言模型只查看单一方向

# BERT 的动机

- 基于微调的 NLP 方法
- 预训练模型捕获足够多的数据信息
- 只需要为新任务添加简单的输出层



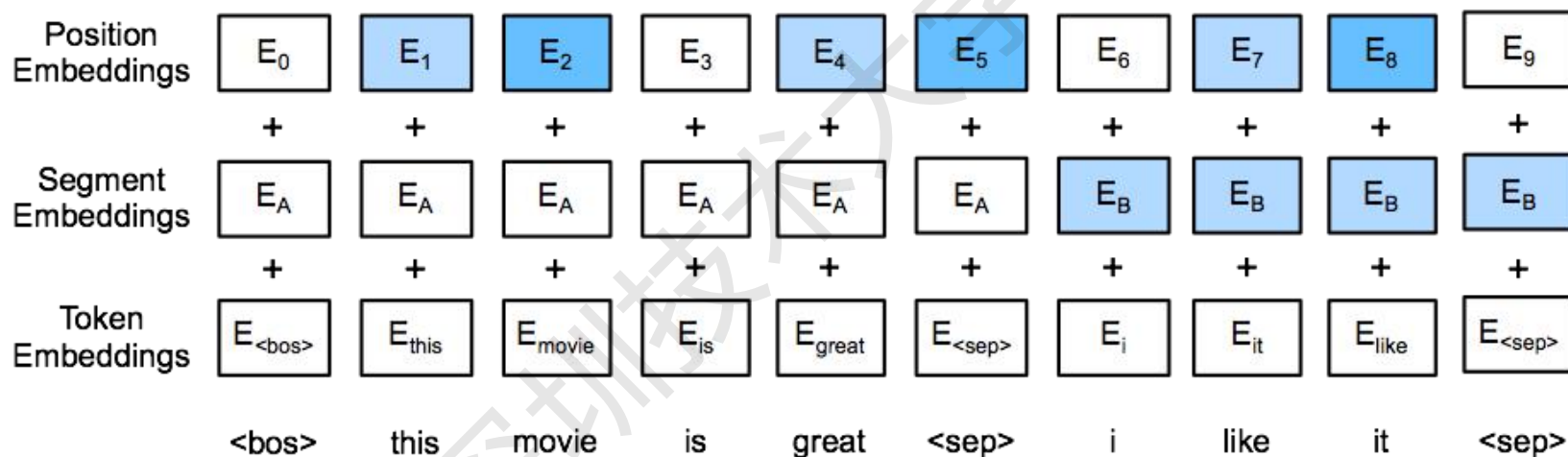


# BERT 架构

- 一个（巨大的）变换器模型编码器（没有解码器）
- 两种模型大小：
  - 基础版：# blocks = 12, 隐含大小 = 768, # heads = 12, # 参数 = 110M
  - 增强版：# blocks = 24, 隐含大小 = 1024, # heads = 16, # 参数 = 340M
- 使用超过30亿单词的大型语料库（书籍和维基百科）  
训练

# 输入

- 每个样本都是一对句子
- 添加其他细分嵌入



# 预训练任务1：掩码语言模型

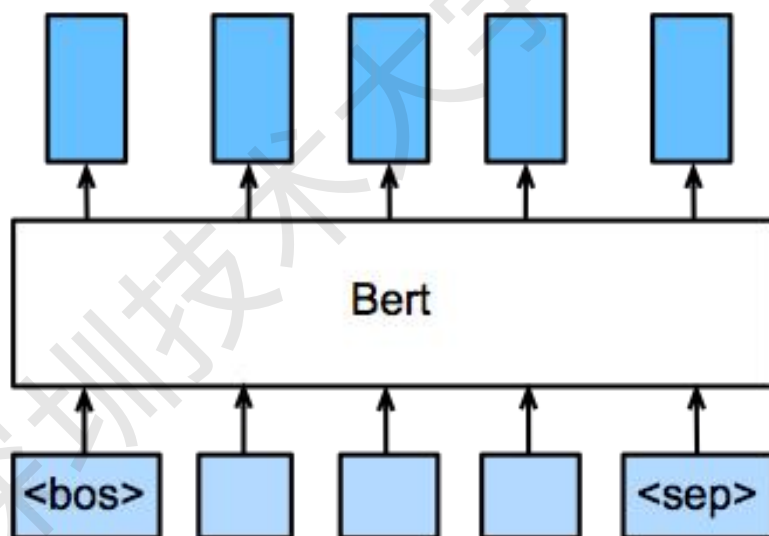
- 在每个句子中随机掩盖（例如 15%）标记，预测这些掩码标记（<mask>）
  - 变换器模型是双向的，它打破了标准语言模型的单向限制
- 微调任务中没有掩码标记（<mask>）
  - 80% 的时间，用 <mask> 替换选定的标记
  - 10% 的时间，用随机挑选的picked tokens替换
  - 10% 的时间，保留原始标记

## 预训练任务2：下一句话预测

- 50% 的时间，选择一个连续的句子对
  - `<bos>` 这部电影很棒 `<sep>` 我喜欢它 `<sep>`
- 50% 的时间，选择一个随机的句子对
  - `<bos>` 这部电影很棒 `<sep>` hello world `<sep>`
- 将变换器模型的输出 `<bos>` 输入到稠密层以预测它是否是顺序对 ( sequential pair)

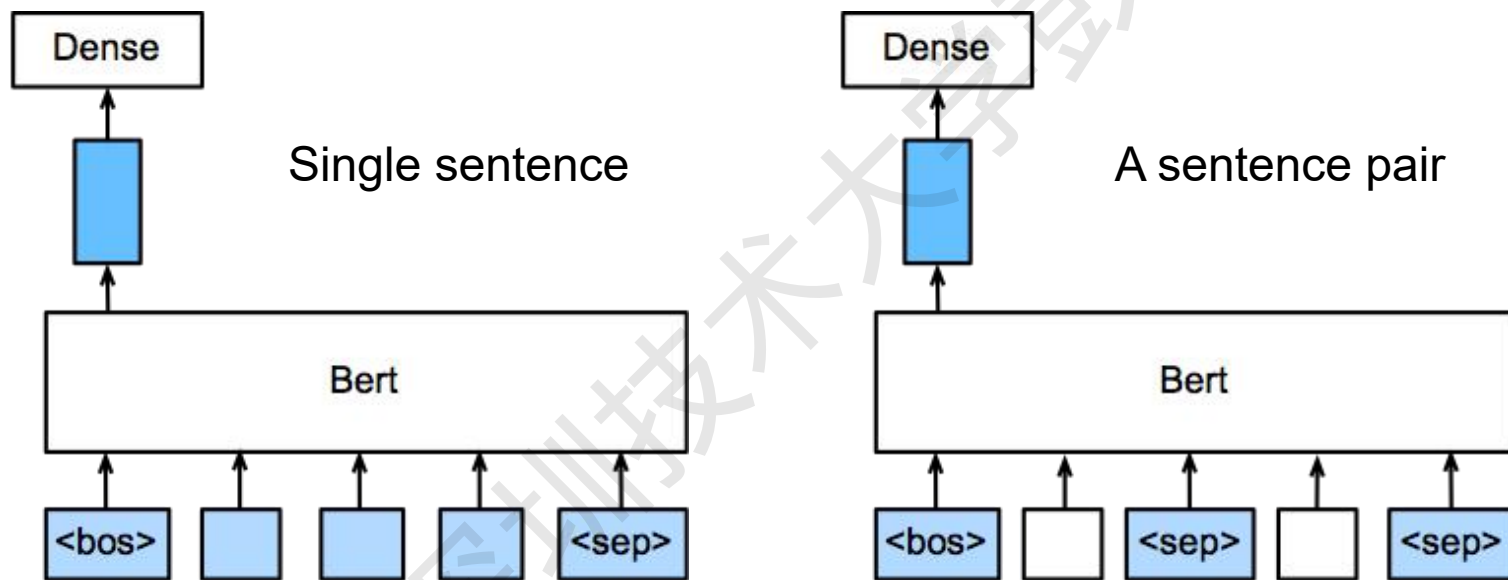
## 用 Bert 微调

- Bert 为捕获上下文信息的每个标记返回一个特征向量
- 不同的微调任务使用不同的向量集



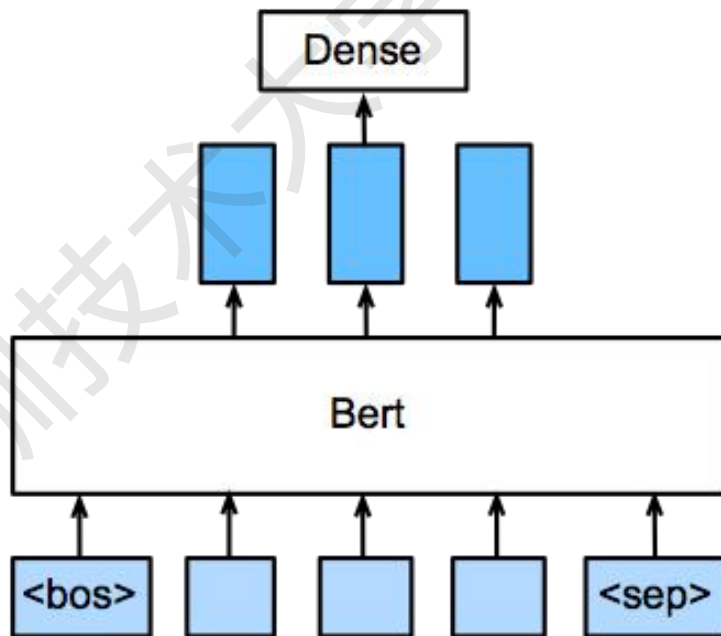
# 语句分类

- 将 `<bos>` 标记向量输入稠密输出层



# 命名实体识别

- 确定标记是否是命名实体，例如人员，组织和位置等等
- 将每个非特殊标记向量馈送到稠密输出层

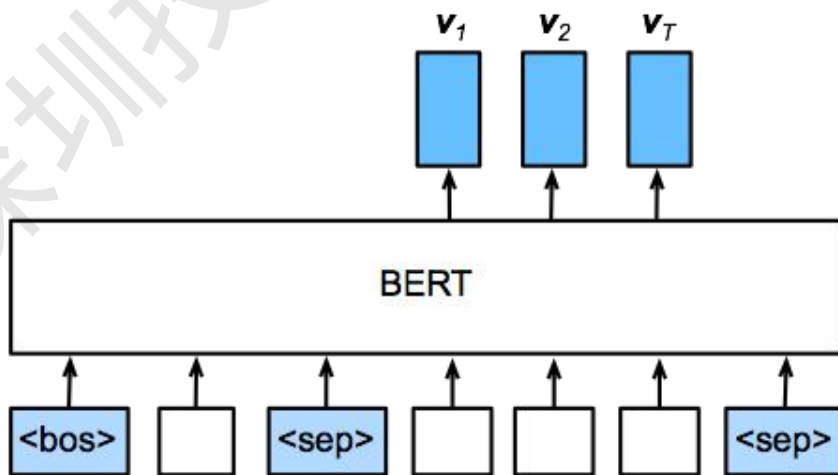


# 自动问答

- 给定问题和描述文本，找到答案，这是描述中的文本段
- 给定  $p_i$ ，描述中的第  $i$  个标记，学习  $s$  中的  $p_i$ ，第  $i$  个标记是这段开始的概率：

$$p_1, \dots, p_T = \text{softmax}(\langle \mathbf{s}, \mathbf{v}_1 \rangle, \dots, \langle \mathbf{s}, \mathbf{v}_T \rangle)$$

同样可以学习第  $i$  个标记是这段结局的概率







# Vision Transformers

视觉Transformers

<https://github.com/lucidrains/vit-pytorch>

# Transformers in Vision

- 识别任务：图像分类、目标检测、动作识别、分割
- 生成模型
- 多模式任务：视觉问题解答、视觉推理
- 视频处理：活动识别、视频预测
- low-level视觉：图像超分辨率、彩色化
- 3D分析：点云分类、分割

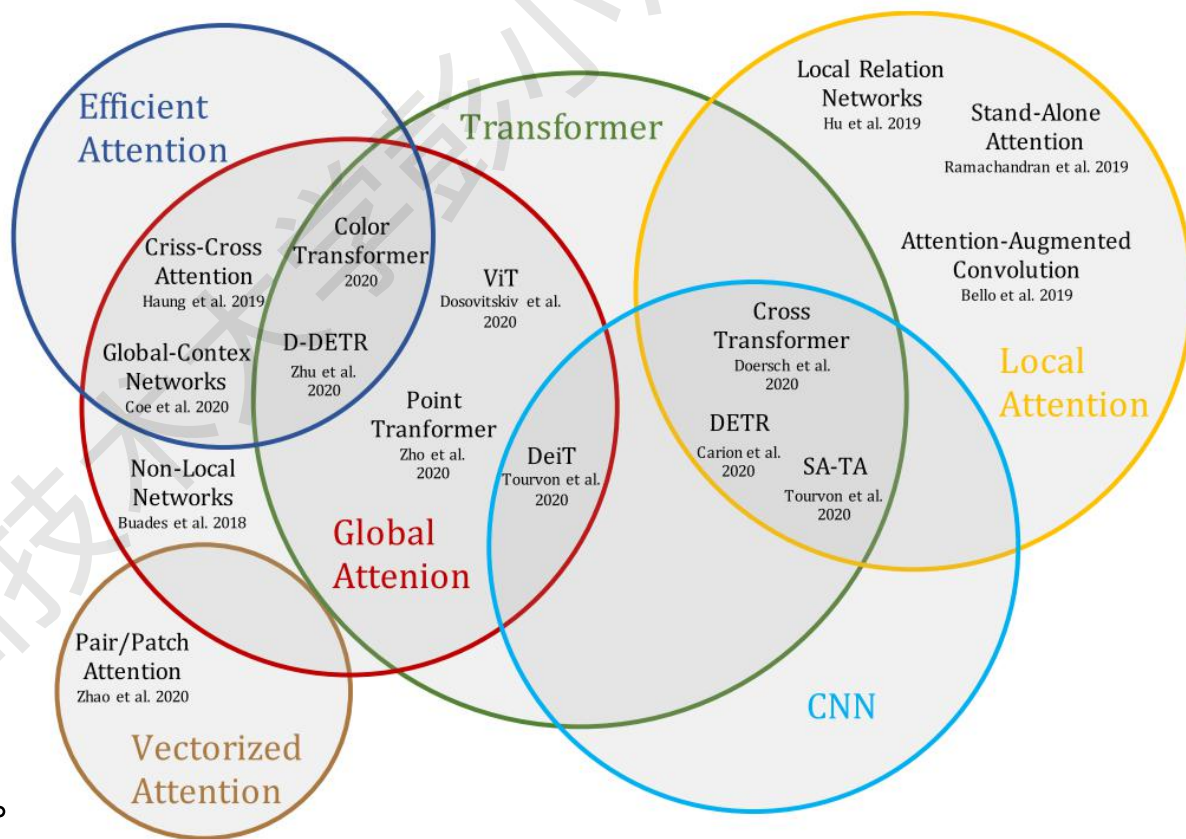
# Transformers in Vision

Task	Input Data Type	Label Type	Method
Image Classification	2D Image	Class labels	ViT [9]
Image Classification	2D Image	Class labels, Soft CNN output labels	DeiT [10]
Object Detection	2D Image	Class labels & Bounding Boxes	DETR [11]
Object Detection	2D Image	Class labels & Bounding Boxes	D-DETR [12]
Low Shot Learning	2D Image	Pretraining without labels and few-shot learning with Class labels	CT [22]
Image Colorization	2D Image	2D Image	ColTran [21]
Action Recognition	Skeleton	Action Classes	ST-TR [66]

Task	Input Data Type	Label Type	Method
Super-resolution	2D Image	2D Image	TTSR [14]
Multi-Model Learning	2D Images	Captions, Class labels, Object tags	Oscar [67]
3D Classification/Segmentation	CAD models, 3D object part segmentation	Object and shape categories	PT [68]
3D Mesh Reconstruction	2D Image	3D Mesh + Human Pose	METRO [69]
Vision and Language Navigation	Instruction text + RGBD panorama + Topological Environment Map	Navigation Plan	Chen <i>et al.</i> [70]
Referring Image Segmentation	2D Image + Language expression	Segmentation mask	CMSA [13]
Video Classification	Audio-Visual	Activity labels	Lee <i>et al.</i> [71]

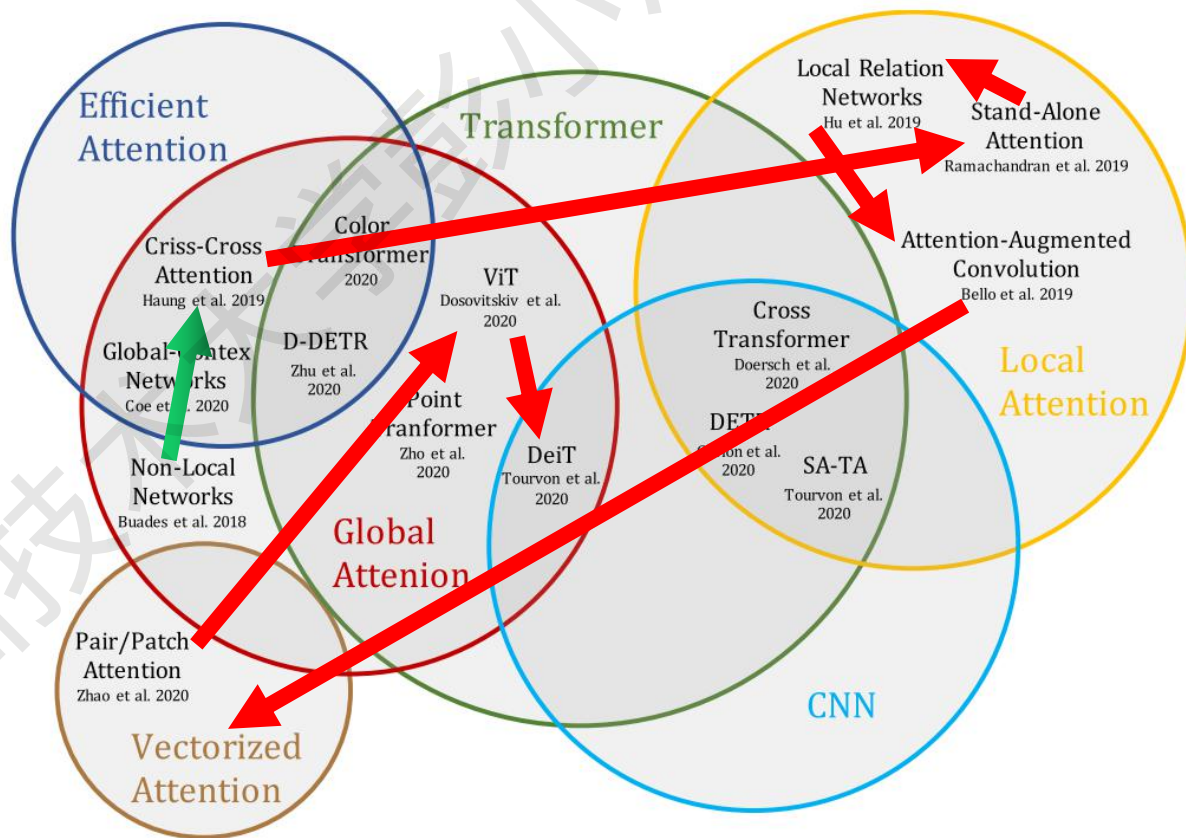
# Transformers in Vision

- Efficient attention: 改善训练Transformer模型需要长时间、大显存、大数据集的问题。
- Global/Local attention: 注意力是应用于整张feature map, 还是应用于确定范围的近邻。
- CNN: 用Transformer实现了某个组件, 其他组件还是用到了CNN, 比如feature map用ResNet提取。



# Transformers in Vision

- Efficient attention: 改善训练Transformer模型需要长时间、大显存、大数据集的问题。
- Global/Local attention: 注意力是应用于整张feature map, 还是应用于确定范围的近邻。
- CNN: 用Transformer实现了某个组件, 其他组件还是用到了CNN, 比如feature map用ResNet提取。



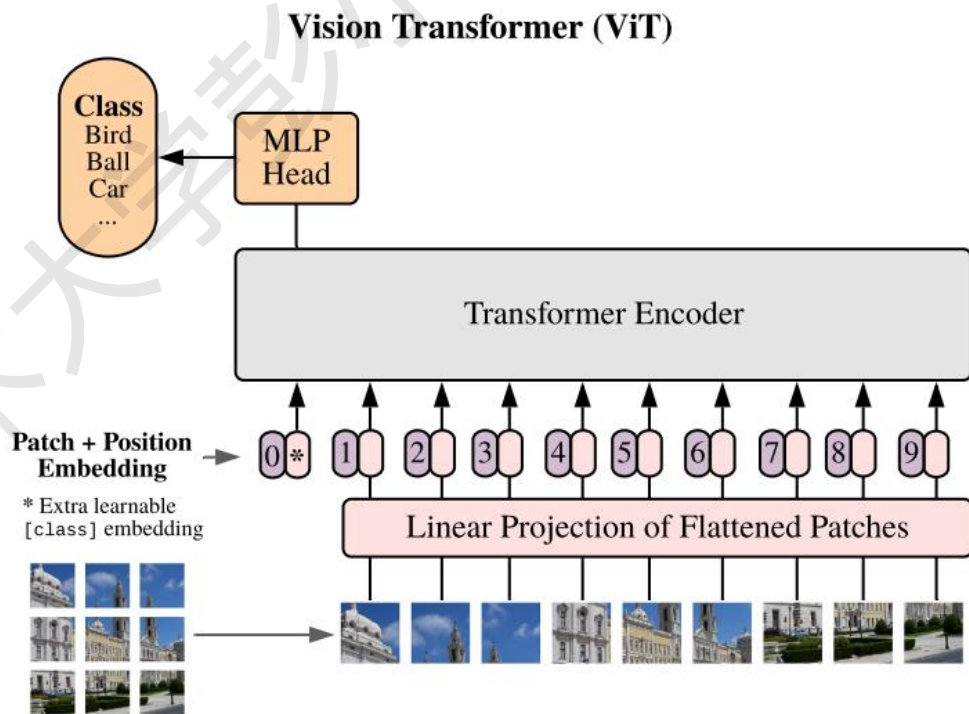




# Vision transformer

## 主要流程

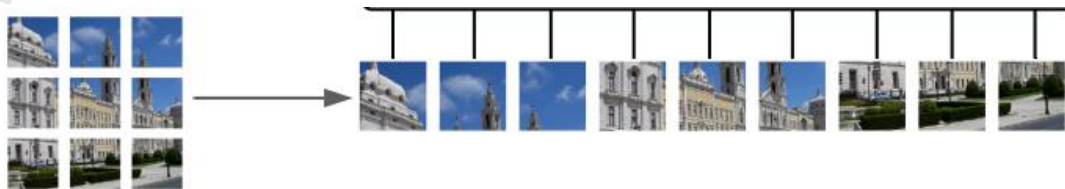
1. 把img 输入后先做一个卷积的操作, 然后根据需要reshape 到合理的尺寸
2. 在channel 维度加上一维cls 的token 用来做分类任务
3. 使用position\_embedding 在每个batch上添加学习到位置相关的特征
4. 进行基于self attention的transformer
5. 拿出整个特征的第一个维度的特征, 然后最后跑一个mlp\_head 得到最终 的分类结果



# Vision transformer

## 数据处理部分

1. 原始输入的图片数据是  $H \times W \times C$ ，我们先对图片作分块，再进行展平。  
假设每个块的长宽为  $(P, P)$ ，那么分块的数目为：  
$$N = H * W / (P * P)$$
2. 然后对每个图片块变成一维向量，  
每个向量大小为：  
$$P * P * C$$
3. 总的输入变换为，即每个patch变成  $(P^2 * C) \times 1$  的列向量：  
$$N \times (P^2 * C)$$





# Vision transformer

## Patch Embedding

接着对每个向量都做一个线性变换（即全连接层），压缩维度为D，这里我们称其为 Patch Embedding

```
self.patch_to_embedding = nn.Linear(patch_h_dim, dim)

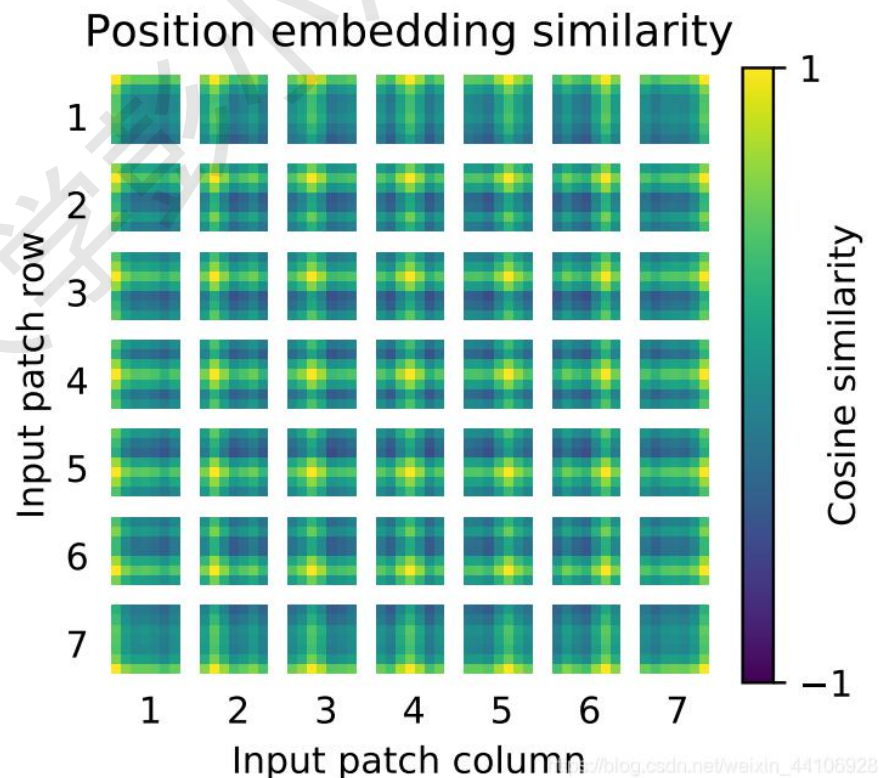
# forward前向代码
x = rearrange(img, 'b c (h p1) (w p2) -> b (h w) (p1 p2 c)', p1=p, p2=p)
x = self.patch_to_embedding(x)
```

# Vision transformer

## Positional Encoding

原始的Transformer引入了一个 Positional encoding 来加入序列的位置信息，同样在这里也引入了pos\_embedding，是用一个可训练的变量替代。

```
self.pos_embedding = nn.Parameter(torch.randn(1, num_patches + 1, dim))
```



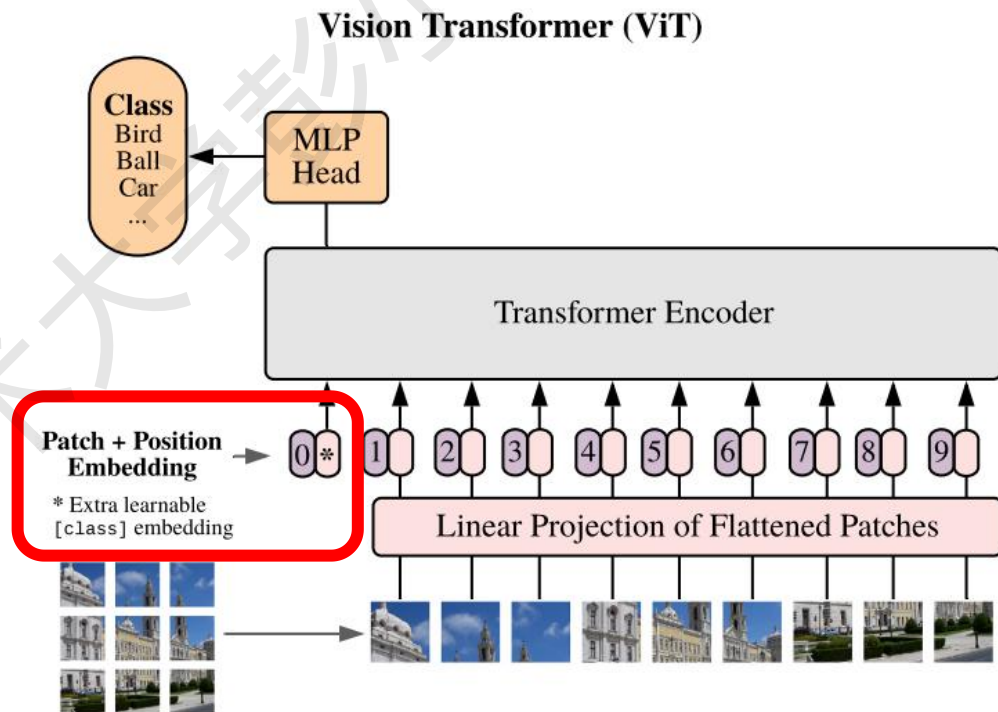
# Vision transformer

## Class\_token

因为传统的Transformer采取的是类似seq2seq编解码的结构

而ViT只用到了Encoder编码器结构，缺少了解码的过程，假设9个向量经过编码器之后，该选择哪一个向量进入到最后的分类头呢？

因此这里作者给了额外的一个用于分类的向量，与输入进行拼接。同样这是一个可学习的变量

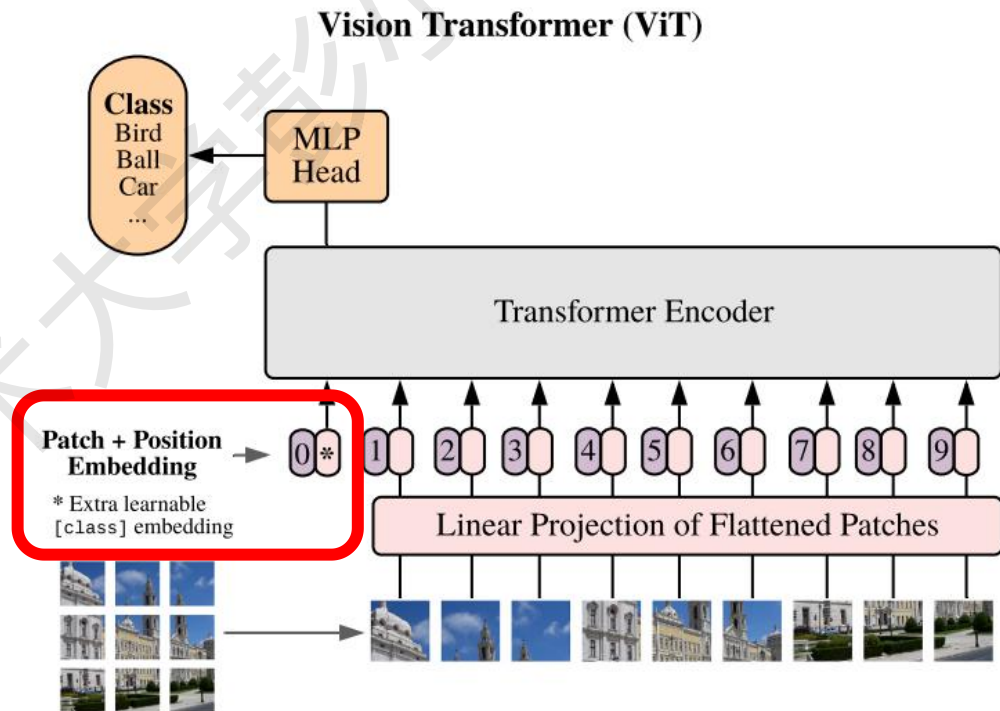


# Vision transformer

## Class\_token

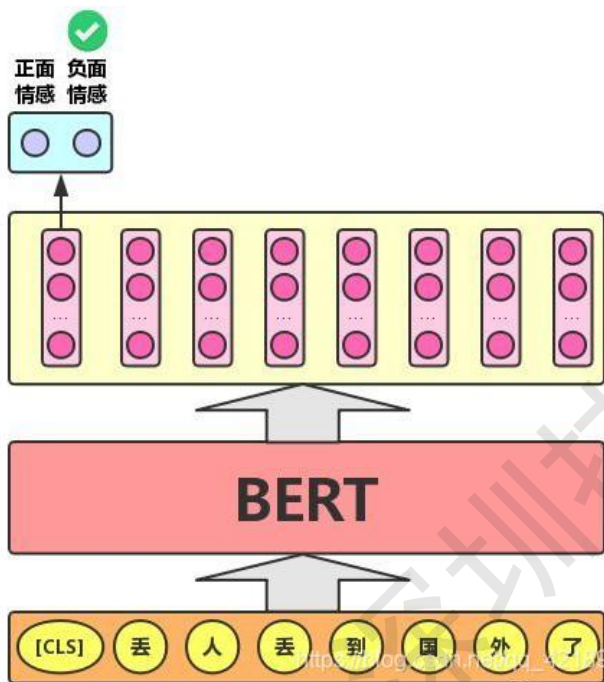
**CLS:** 该符号对应的输出向量作为整张图的语义表示，用于图片分类

可以理解为：与各个patch相比，这个无明显语义信息的符号会更“公平”地融合文本中各个patch的语义信息。  
以下给出NLP的解释：

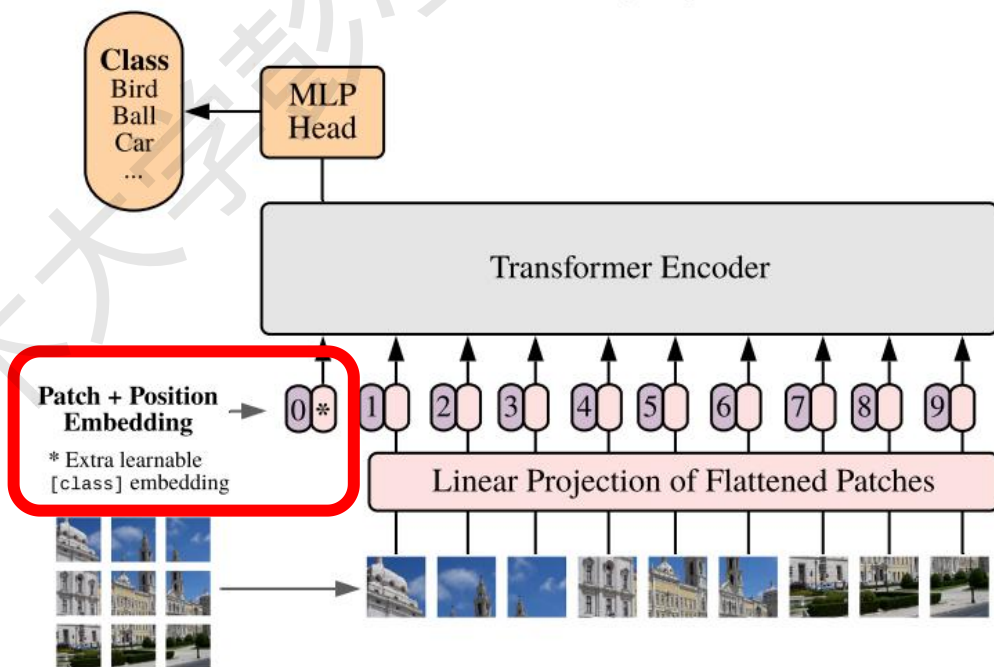


# Vision transformer

## Class\_token



## Vision Transformer (ViT)



# Vision transformer

## 分类

分类头很简单，加入了LayerNorm和两层全连接层实现的，采用的是GELU激活函数。

Python

```
self.mlp_head = nn.Sequential(
    nn.LayerNorm(dim),
    nn.Linear(dim, mlp_dim),
    nn.GELU(),
    nn.Dropout(dropout),
    nn.Linear(mlp_dim, num_classes)
)
```

Copy Caption ...

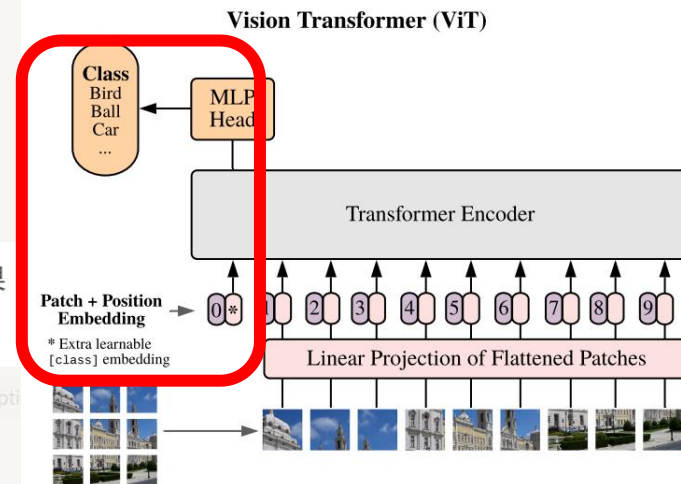
最终分类我们只取第一个，也就是用于分类的token，输入到分类头里，得到最后的分类结果

MLP = 多层FC layer 构成的NN

Python

```
self.to_cls_token = nn.Identity()
forward前向部分
x = self.transformer(x, mask)
x = self.to_cls_token(x[:, 0])
return self.mlp_head(x)
```

Copy Caption ...





# DeiT: Transformer+Distillation

Training data-efficient image transformers  
& distillation through attention

Hugo Touvron<sup>\*,†</sup> Matthieu Cord<sup>†</sup> Matthijs Douze<sup>\*</sup>

Francisco Massa<sup>\*</sup> Alexandre Sablayrolles<sup>\*</sup> Hervé Jégou<sup>\*</sup>

<sup>\*</sup>Facebook AI

<sup>†</sup>Sorbonne University

5555@over计算机视觉

<https://blog.csdn.net/arnusi1994>

# DeiT

## • 动机

ViT要非常大的数据集 (JFT-300M) 训练才有SOTA的性能，而这个数据集Google还没有发布，此外训练所需的计算资源也非常庞大，最简单的模型需要8TPU v3 跑30天。DeiT只要在ImageNet上，用8GPU训3天，性能还能超过ViT。

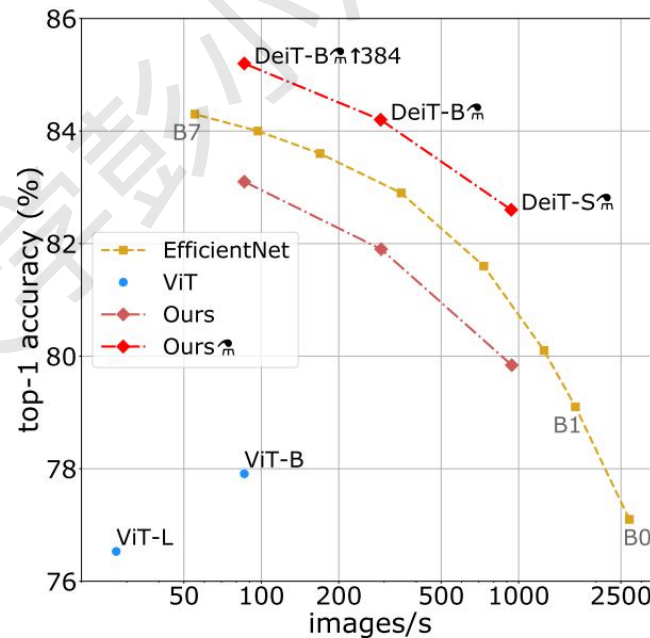



Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to ViT-B, but the training is more adapted to a data-starving regime. It is learned in a few days on one machine. The symbol  refers to models trained with our transformer-specific distillation. See Table 5 for details and more models.



# DeiT

## • 优点

用ImageNet训练，在ImageNet测试的性能。ViT在小数据集上的性能不如使用CNN的SOTA

EfficientNet[3]，跟ViT结构相同，使用更好的训练策略的DeiT比ViT的性能已经有了很大的提升，在此基础上，再加上蒸馏

(distillation) 操作，性能超过了EfficientNet。

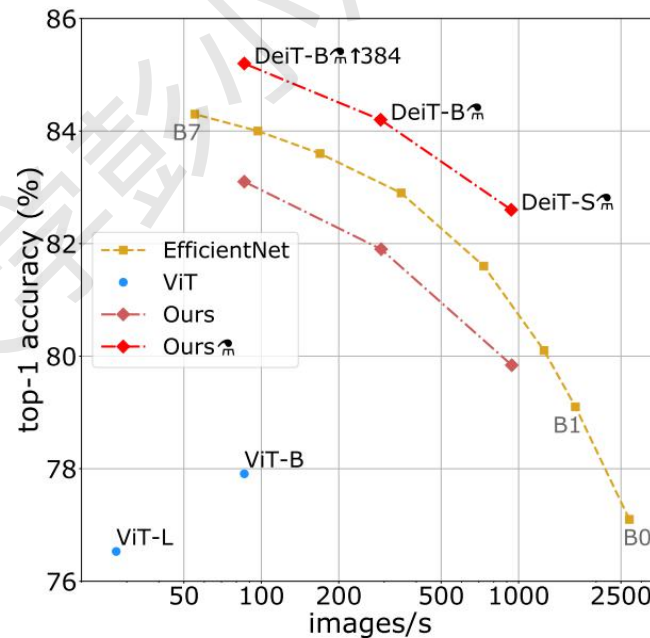



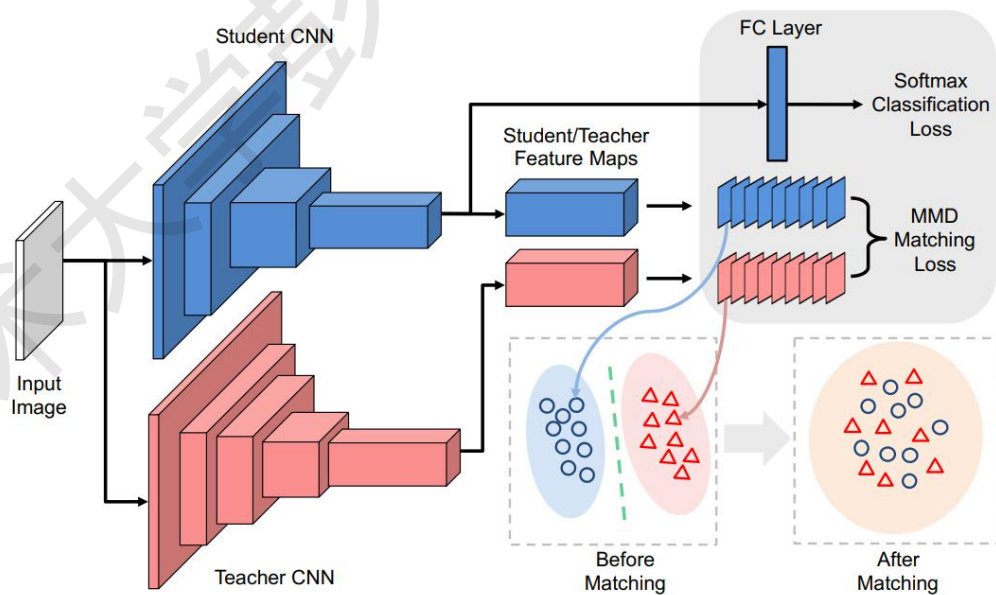
Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to ViT-B, but the training is more adapted to a data-starving regime. It is learned in a few days on one machine. The symbol  refers to models trained with our transformer-specific distillation. See Table 5 for details and more models.

# DeiT

## ●知识蒸馏：教师-学生网络

广泛的用于模型压缩和迁移学习当中。做法是先训练一个teacher网络，然后使用这个teacher网络的输出和数据的真实标签去训练student网络。

1. 将网络从大网络转化成一个网络，并保留接近于大网络的性能
2. 将多个网络的学到的知识转移到一个网络中，使得单个网络的性能接近ensemble的结果



# DeiT

## ● 蒸馏输出模仿教师网络输出

在ViT的基础上，加上一个distillation token，与class token很像，唯一的区别在于，class token的目标是跟真实的label一致，而distillation token是要跟teacher model预测的label一致。

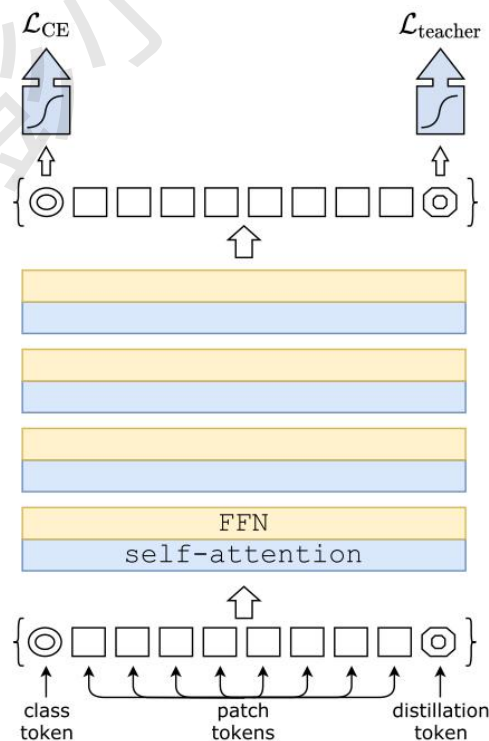
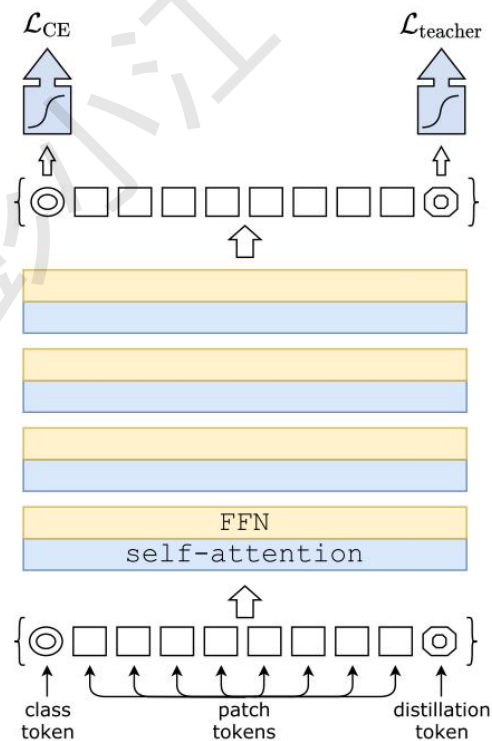


Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce the (hard) label predicted by the teacher, instead of true label. Both the class and distillation tokens input to the transformers are learned by back-propagation.

# DeiT

蒸馏分两种，软蒸馏soft distillation、硬蒸馏hard distillation

- 软蒸馏，如下式所示， $Z_s$  和  $Z_t$  分别是 student model 和 teacher model 的输出，CE 表示交叉熵，KL 表示 KL 散度， $\psi$  表示 SoftMax 函数， $\lambda$  和  $\tau$  是超参数。



$$\mathcal{L}_{\text{global}} = (1 - \lambda) \mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \lambda \tau^2 \text{KL}(\psi(Z_s/\tau), \psi(Z_t/\tau)).$$

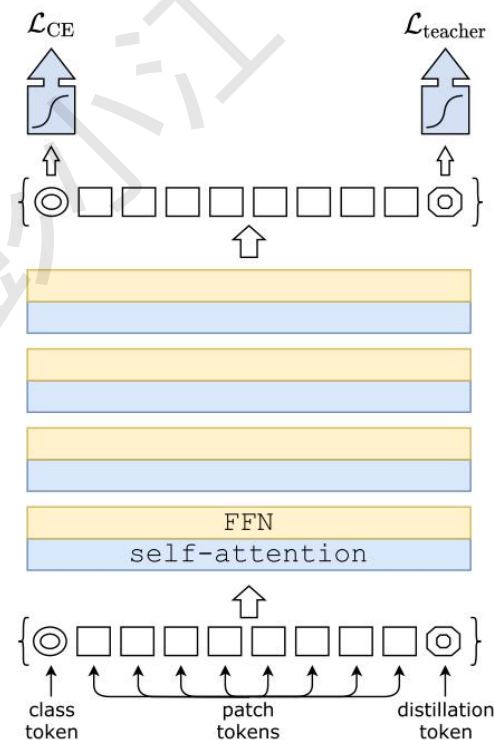
# DeiT

蒸馏分两种，软蒸馏soft distillation、硬蒸馏hard distillation

- 硬蒸馏，如下式所示， $Z_s$  是student model的输出， $y_t$  是teacher model输出的one-hot标签，CE表示交叉熵， $\psi$  表示SoftMax函数。

$$y_t = \operatorname{argmax}_c Z_t(c)$$

$$\mathcal{L}_{\text{global}}^{\text{hardDistill}} = \frac{1}{2} \mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \frac{1}{2} \mathcal{L}_{\text{CE}}(\psi(Z_s), y_t).$$



# DeiT

class token和distillation token是朝着不同的方向收敛的，作者对各个layer的这两个token计算余弦相似度，平均值只有0.06，不过随着网络逐层计算，它们在网络中变得越来越相似，在最后一层是0.93，也就是相似但不相同。

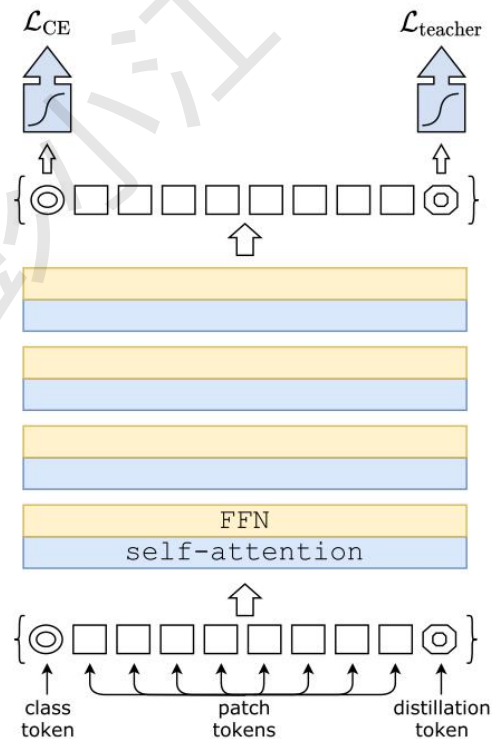


Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce the (hard) label predicted by the teacher, instead of true label. Both the class and distillation tokens input to the transformers are learned by back-propagation.



# DeiT

## Transformer models

Table 1: Variants of our DeiT architecture. The larger model, DeiT-B, has the same architecture as the ViT-B [15]. The only parameters that vary across models are the embedding dimension and the number of heads, and we keep the dimension per head constant (equal to 64). Smaller models have a lower parameter count, and a faster throughput. The throughput is measured for images at resolution  $224 \times 224$ .

Model	ViT model	embedding dimension	#heads	#layers	#params	training resolution	throughput (im/sec)
DeiT-Ti	N/A	192	3	12	5M	224	2536
DeiT-S	N/A	384	6	12	22M	224	940
DeiT-B	ViT-B	768	12	12	86M	224	292

# DeiT

## • Teacher models comparison

## • 用CNN作为教师网络能够使得ViT学习更丰富特征

在teacher model上，卷积模型比Transformer好

Table 2: We compare on ImageNet [42] the performance (top-1 acc., %) of the student as a function of the teacher model used for distillation.

Teacher Models	acc.	Student: DeiT-B $\uparrow$ 384	
		pretrain	
DeiT-B	81.8	81.9	83.1
RegNetY-4GF	80.0	82.7	83.6
RegNetY-8GF	81.7	82.7	83.8
RegNetY-12GF	82.4	83.1	84.1
RegNetY-16GF	82.9	83.1	84.2



# DeiT

## • Soft distillation & Hard distillation

## Class token & Distillation token & both

Table 3: Distillation experiments on Imagenet with DeiT, 300 epochs of pre-training. We report the results for our new distillation method in the last three rows. We separately report the performance when classifying with only one of the class or distillation embeddings, and then with a classifier taking both of them as input. In the last row (class+distillation), the result correspond to the late fusion of the class and distillation classifiers.

method ↓	Supervision		ImageNet top-1 (%)			
	label	teacher	Ti 224	S 224	B 224	B↑384
DeiT– no distillation	✓	✗	72.2	79.8	81.8	83.1
DeiT– usual distillation	✗	soft	72.2	79.8	81.8	83.2
DeiT– hard distillation	✗	hard	74.3	80.9	83.0	84.0
DeiT <sub>ms</sub> : class embedding	✓	hard	73.9	80.9	83.0	84.2
DeiT <sub>ms</sub> : distil. embedding	✓	hard	74.6	81.1	83.1	84.4
DeiT <sub>ms</sub> : class+distillation	✓	hard	74.5	81.2	83.4	84.5

# DeiT

## • 其他的分类数据集上的泛化能力

Table 7: We compare Transformers based models on different transfer learning task with ImageNet pre-training. We also report results with convolutional architectures for reference.

Model	ImageNet	CIFAR-10	CIFAR-100	Flowers	Cars	iNat-18	iNat-19	im/sec
Grafit ResNet-50 [49]	79.6	-	-	98.2	92.5	69.8	75.9	1226.1
Grafit RegNetY-8GF [49]	-	-	-	99.0	94.0	76.8	80.0	591.6
ResNet-152 [10]	-	-	-	-	-	69.1	-	526.3
EfficientNet-B7 [48]	84.3	98.9	91.7	98.8	94.7	-	-	55.1
ViT-B/32 [15]	73.4	97.8	86.3	85.4	-	-	-	394.5
ViT-B/16 [15]	77.9	98.1	87.1	89.5	-	-	-	85.9
ViT-L/32 [15]	71.2	97.9	87.1	86.4	-	-	-	124.1
ViT-L/16 [15]	76.5	97.9	86.4	89.7	-	-	-	27.3
DeiT-B	81.8	99.1	90.8	98.4	92.1	73.2	77.7	292.3
DeiT-B $\uparrow$ 384	83.1	99.1	90.8	98.5	93.3	79.5	81.4	85.9
DeiT-B $\uparrow$	83.4	99.1	91.3	98.8	92.9	73.7	78.4	290.9
DeiT-B $\uparrow$ 384	84.4	99.2	91.4	98.9	93.9	80.1	83.0	85.9



# PyTorch进行ViT图片表情识别应用

- RAF-DB数据 112x112大小

代码见project-ViTemotion

# 参考文献

1. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. CoRR, abs/1706.03762, 2017.
2. Dosovitskiy A , Beyer L , Kolesnikov A , et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale[J]. 2020.
3. Touvron H , Cord M , Douze M , et al. Training data-efficient image transformers & distillation through attention[J]. 2020.