



温州大學
WENZHOU UNIVERSITY

深度学习-Vision Transformer (ViT)

黄海广 副教授

2023年06月

- 01** 背景知识
- 02** 模型介绍
- 03** 模型训练策略
- 04** 模型的缺点与改进
- 05** 模型的代码实现

1.背景知识

3

01 背景知识

02 模型介绍

03 模型训练策略

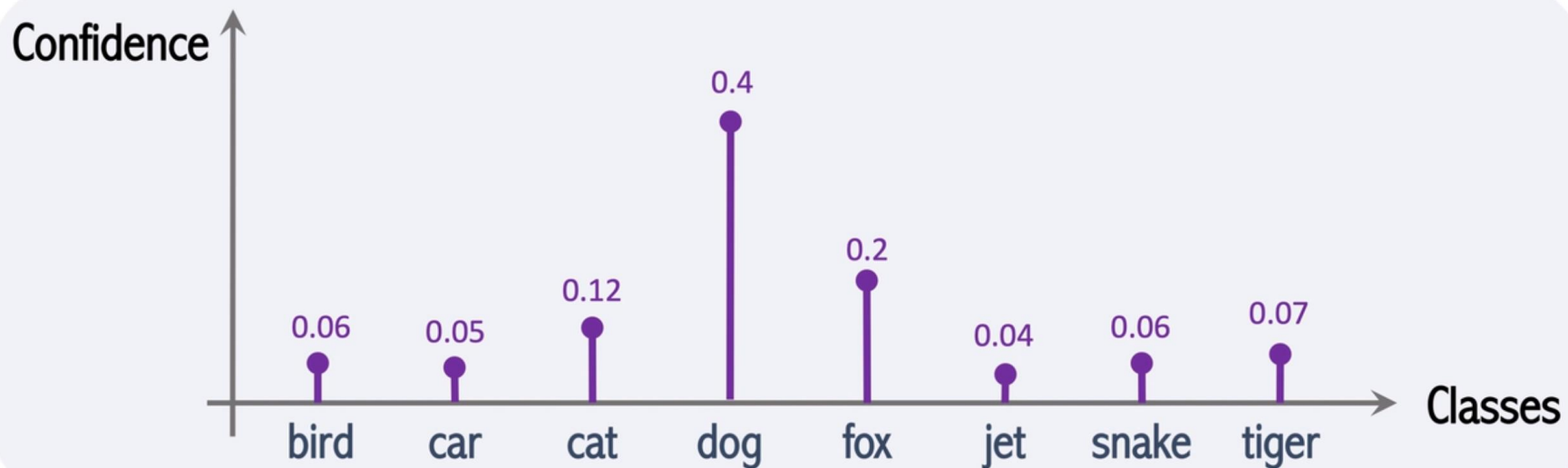
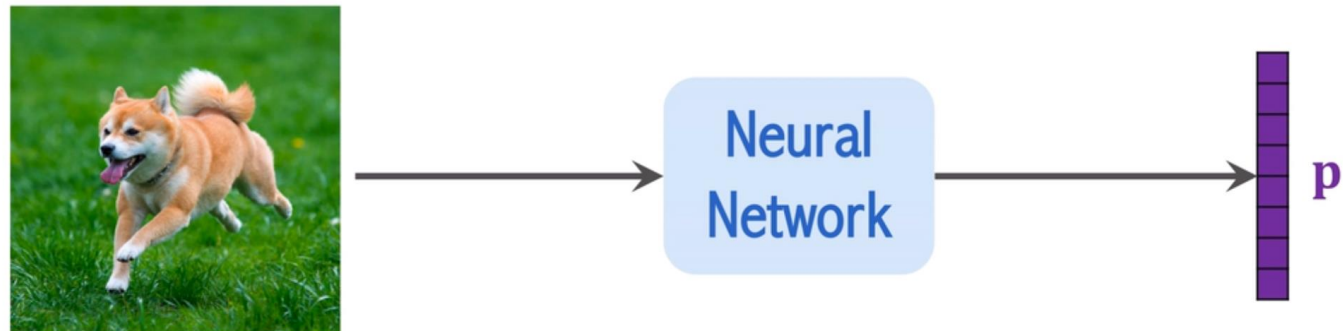
04 模型的缺点与改进

05 模型的代码实现

1.背景知识

4

图片分类的原理

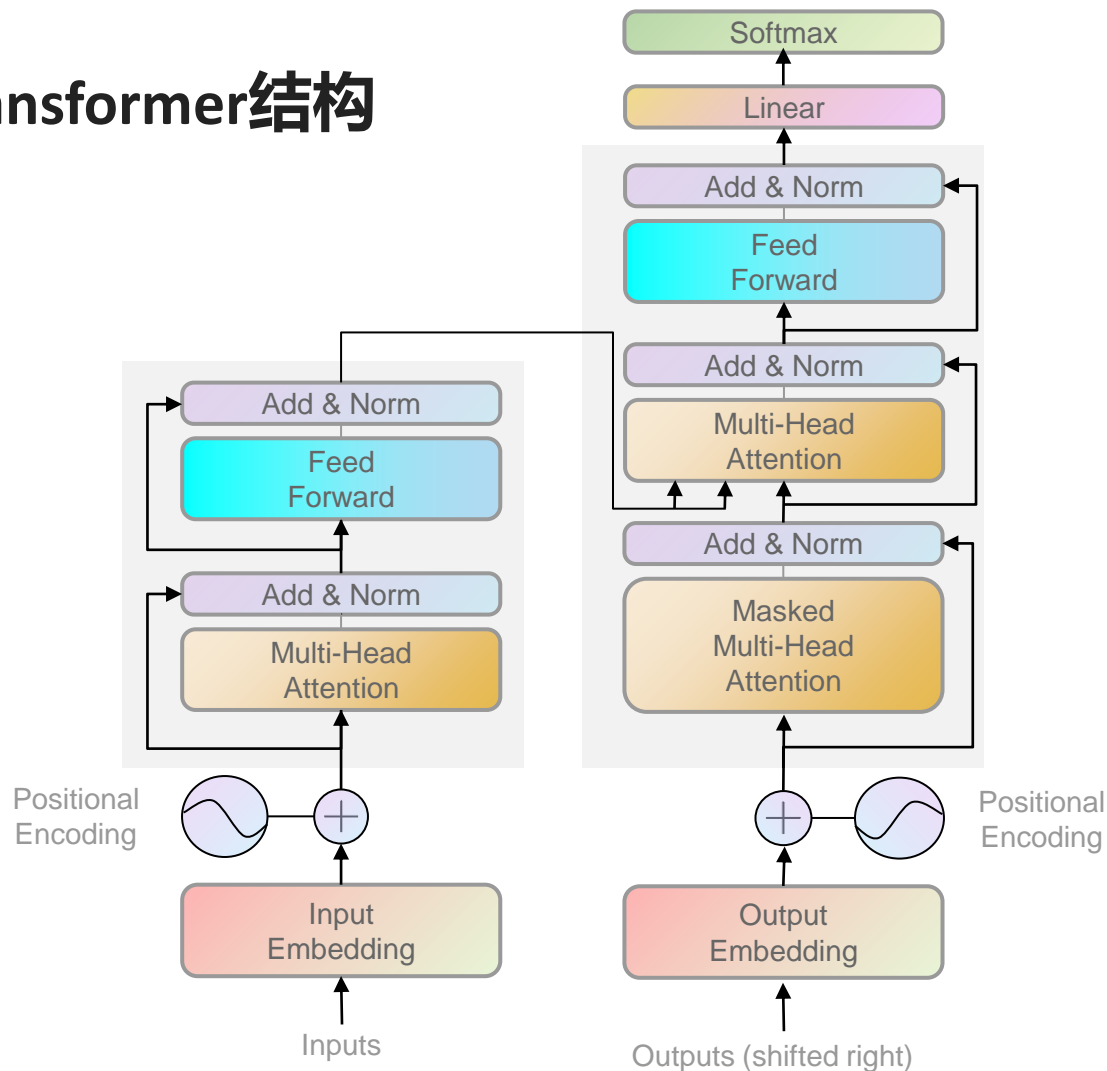


1.背景知识

5

2017年google的机器翻译团队在NIPS上发表了**Attention is all you need**的文章，开创性地提出了在序列转录领域，完全抛弃CNN和RNN，只依赖**Attention-注意力**结构的简单的网络架构，名为**Transformer**；论文实现的任务是机器翻译。

Transformer结构

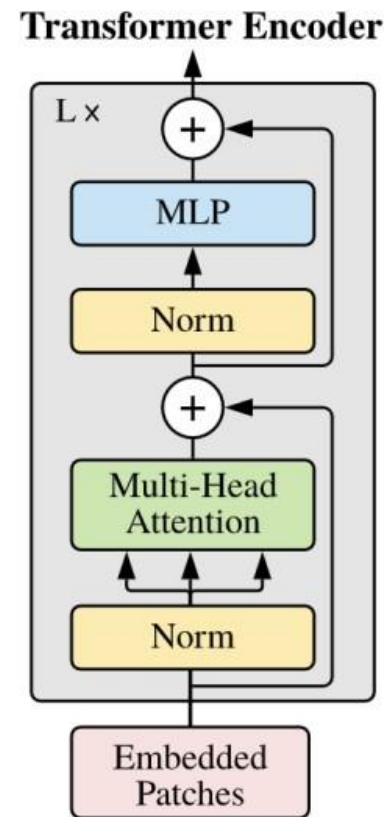
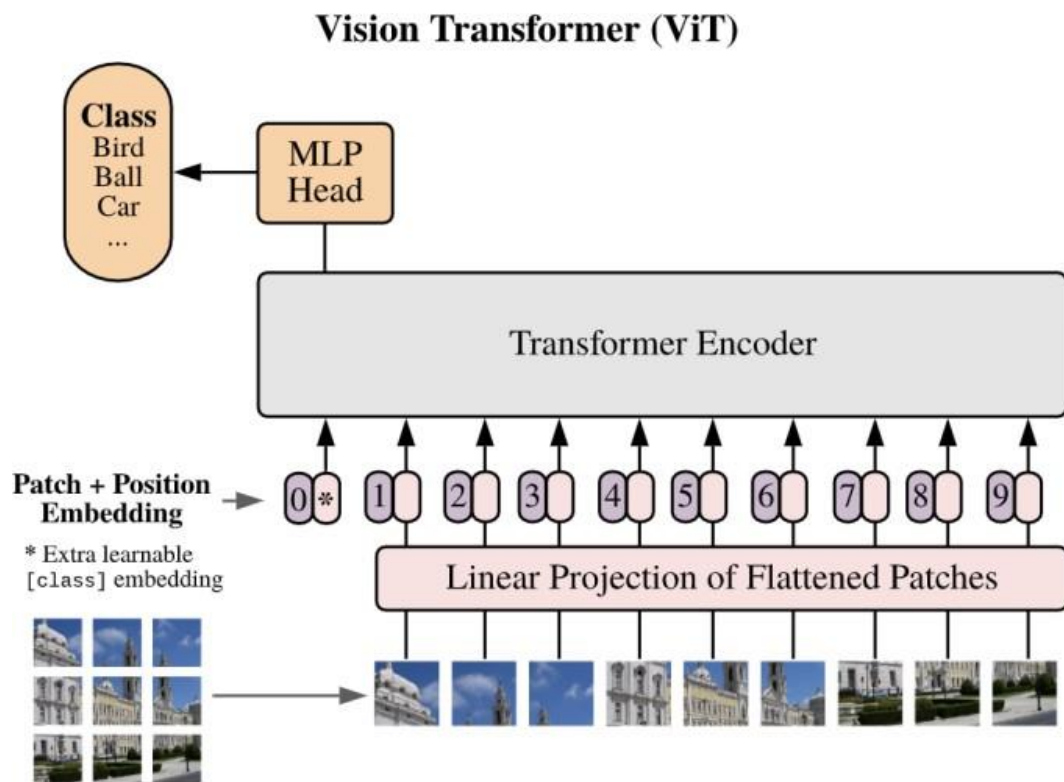


1.背景知识

6

为什么需要用transformer

Transformer原本是用来做NLP的工作的，所以ViT的首要任务是将图转换成词的结构，这里采取的方法是如上图左下角所示，将图片分割成小块，每个小块就相当于句子里的一个词。这里把每个小块称作Patch，而**Patch Embedding**就是把每个Patch再经过一个全连接网络压缩成一定维度的向量。



1.背景知识

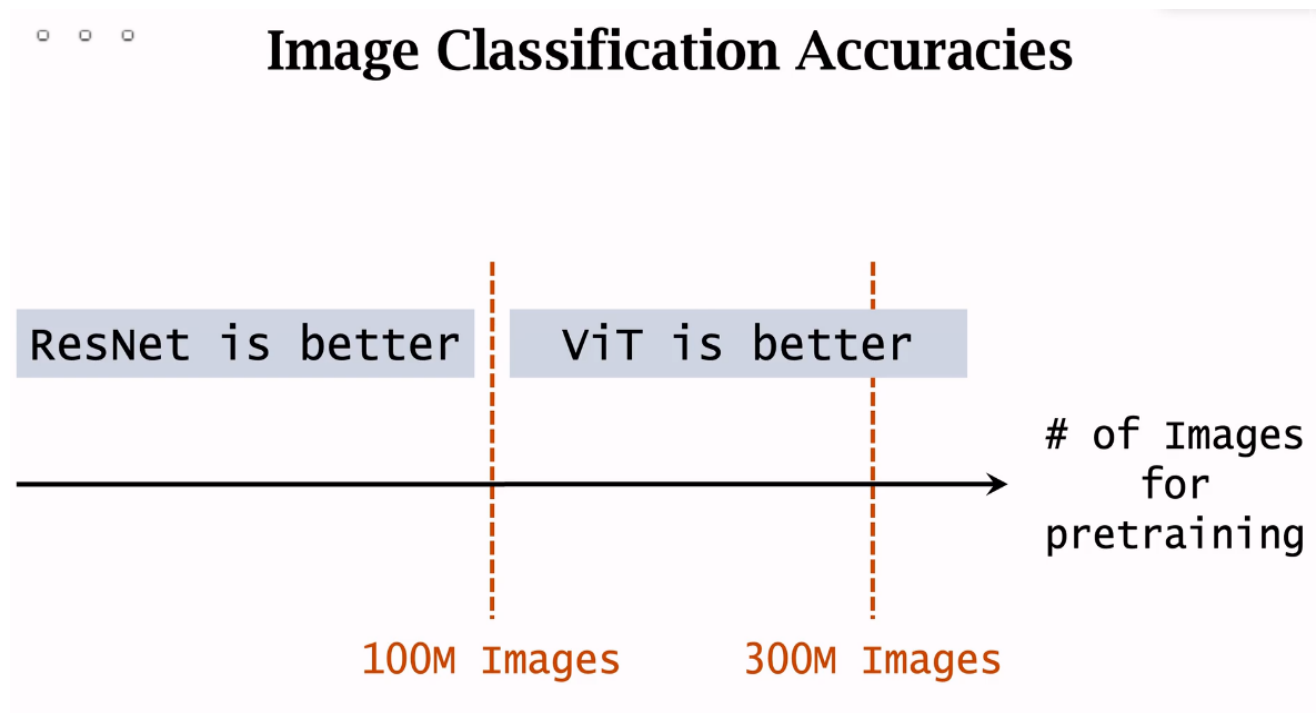
7

为什么需要用transformer

CNN（如ResNet）是图像分类的最佳解决方案。

如果预训练的数据集足够大（至少一亿张图像），则Vision Transformer（ViT）将击败CNN（小幅度）

Vision Transformer（ViT）实际上就是Transformer的encode网络。



2.模型介绍

8

01 背景知识

02 模型介绍

03 模型训练策略

04 模型的缺点与改进

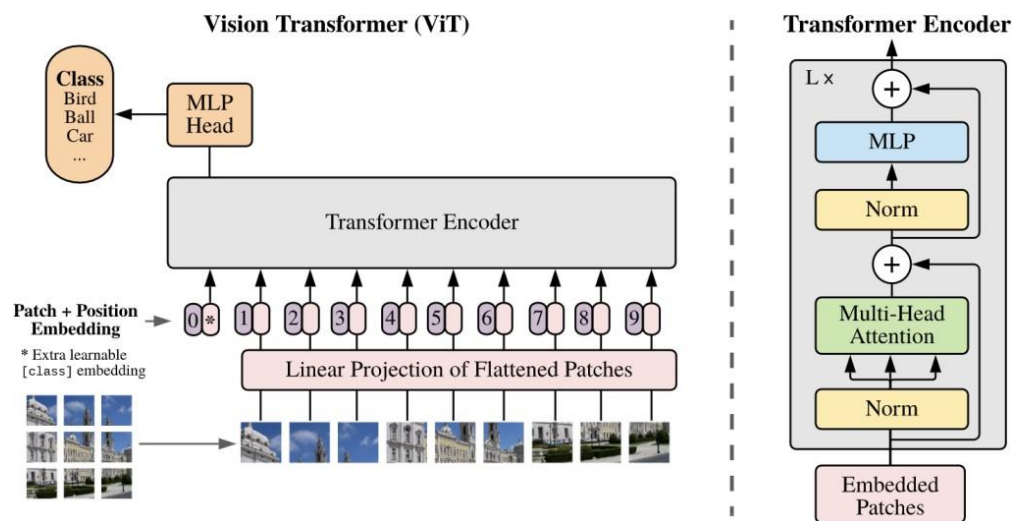
05 模型的代码实现

2.模型介绍

9

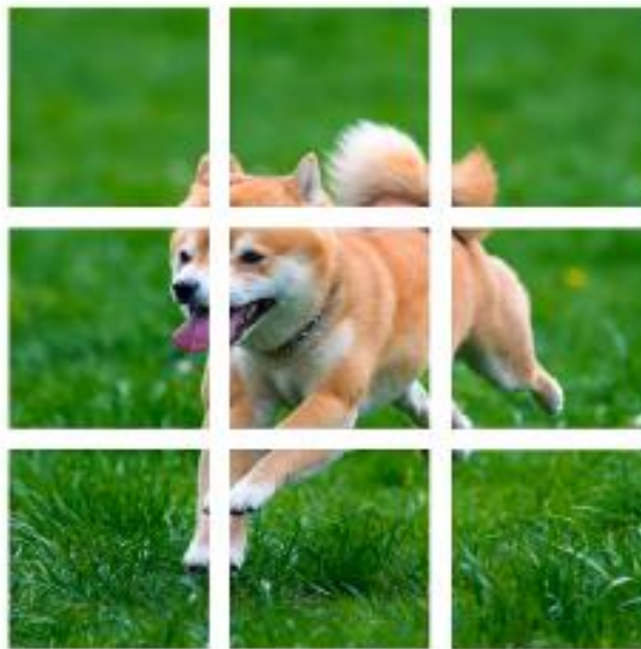
模型思路

- 1.图片切分为patch
- 2.patch转化为embedding
- 3.位置embedding和tokenembedding相加
- 4.输入到Transformer模型
- 5.CLS输出做多分类任务



2.模型介绍

10

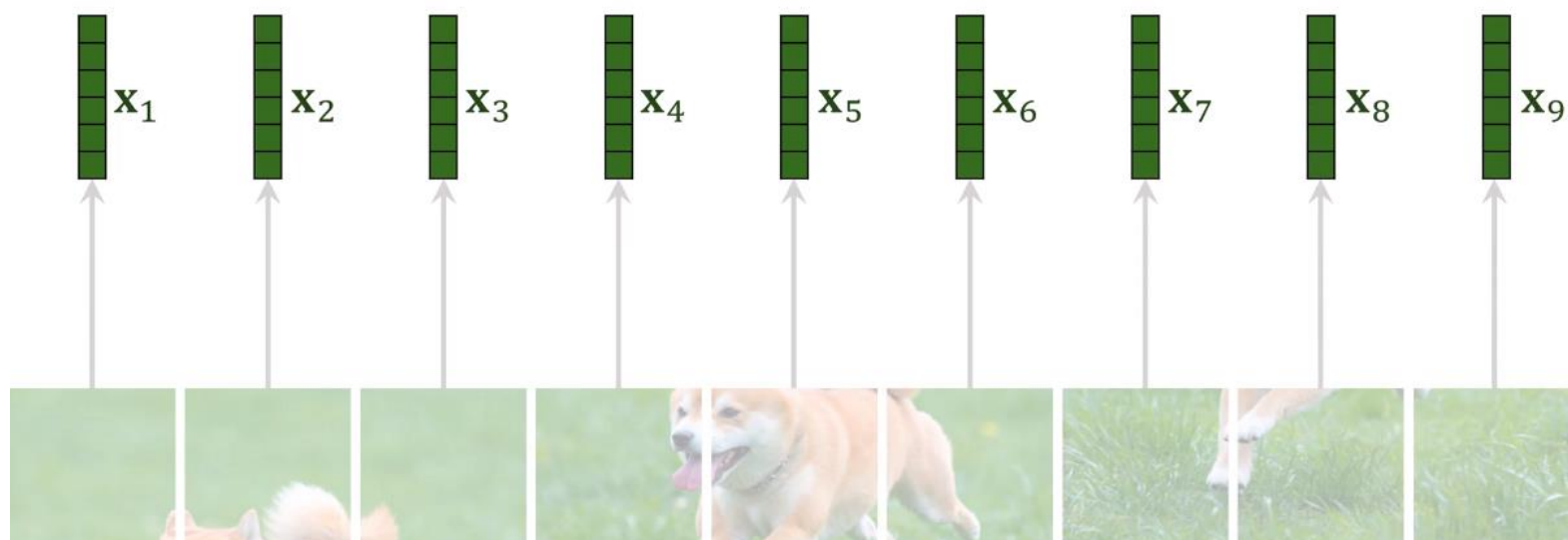
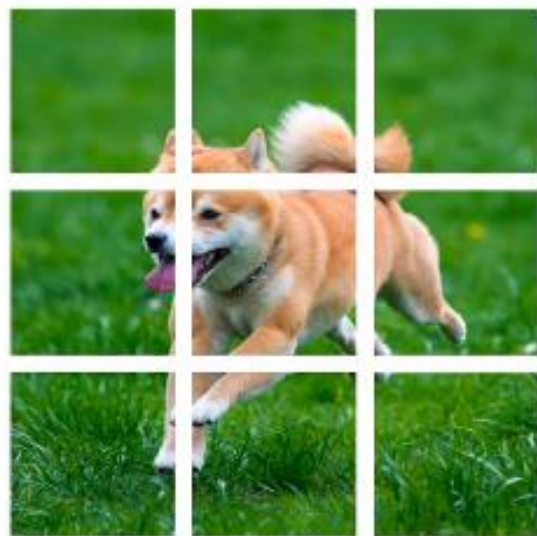


先将图片分成 $N \times N$ 的patch块(原始论文是 16×16)
patch块可以重叠(上图没有重叠, 是 9×9 的patch块)

2.模型介绍

11

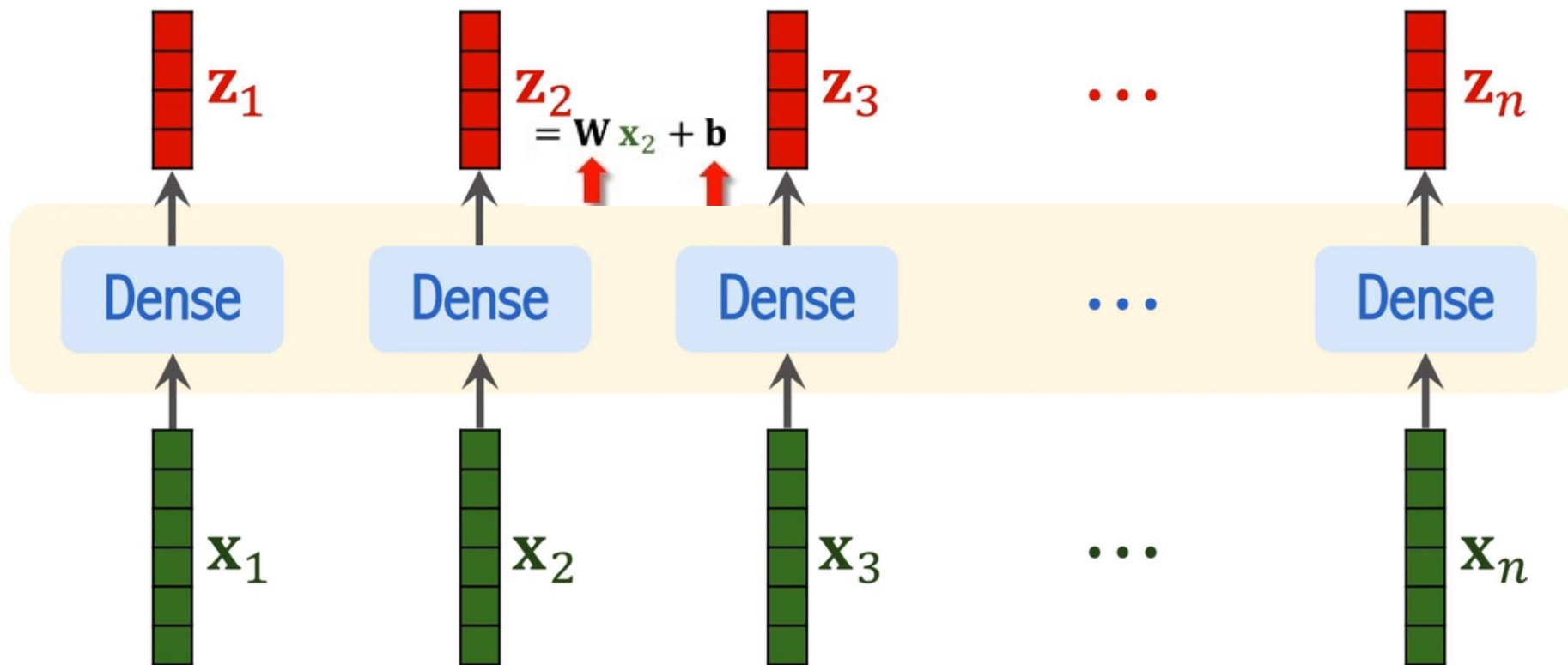
将patch打平，对每个 patch 进行线性映射，提取特征



2.模型介绍

12

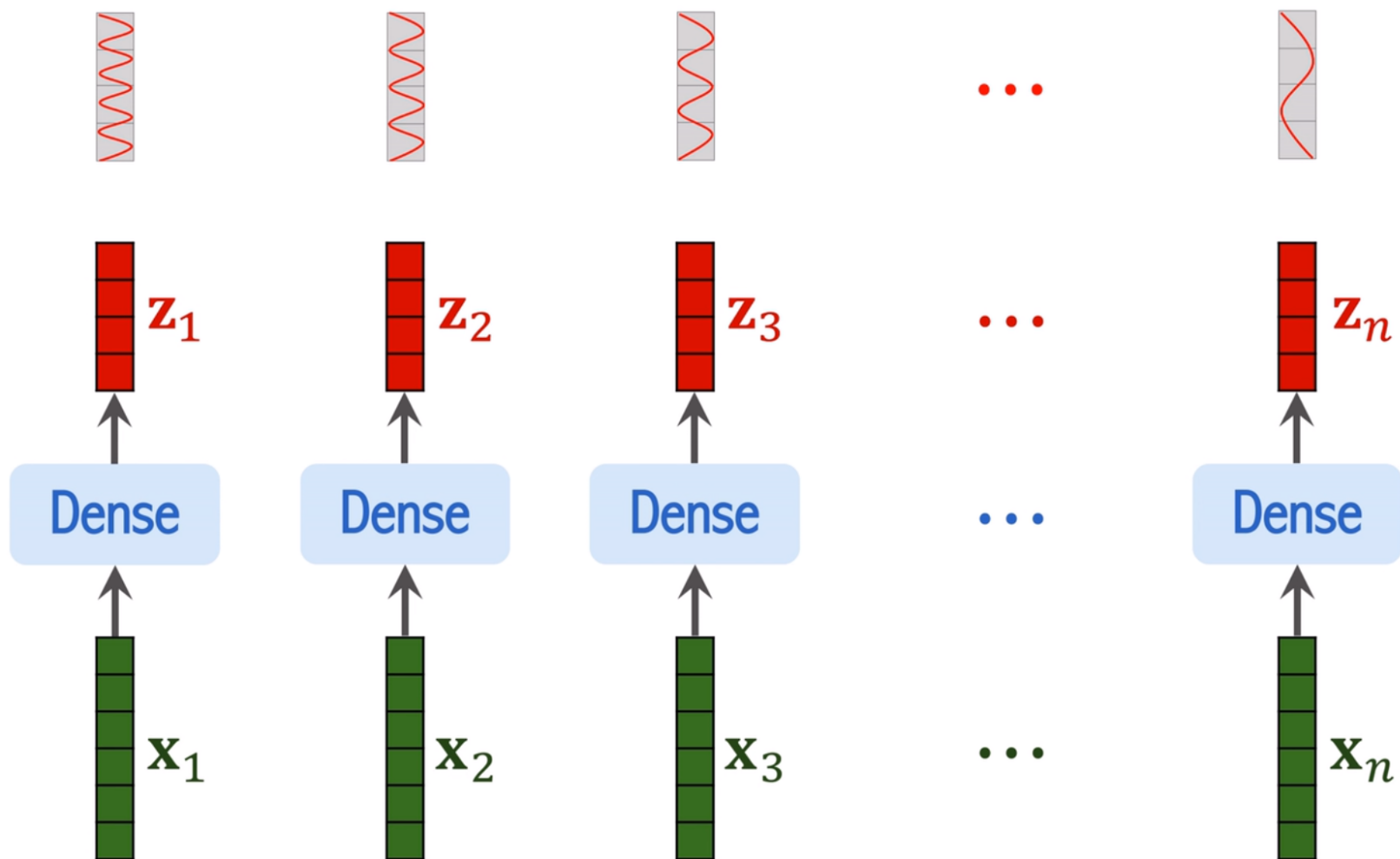
提取特征



2.模型介绍

13

1.将位置编码信息加入提取的特征



2.模型介绍

14

位置编码信息对准确率的影响

结论:编码有用,但是怎么编码影响不大,干脆用简单的得了

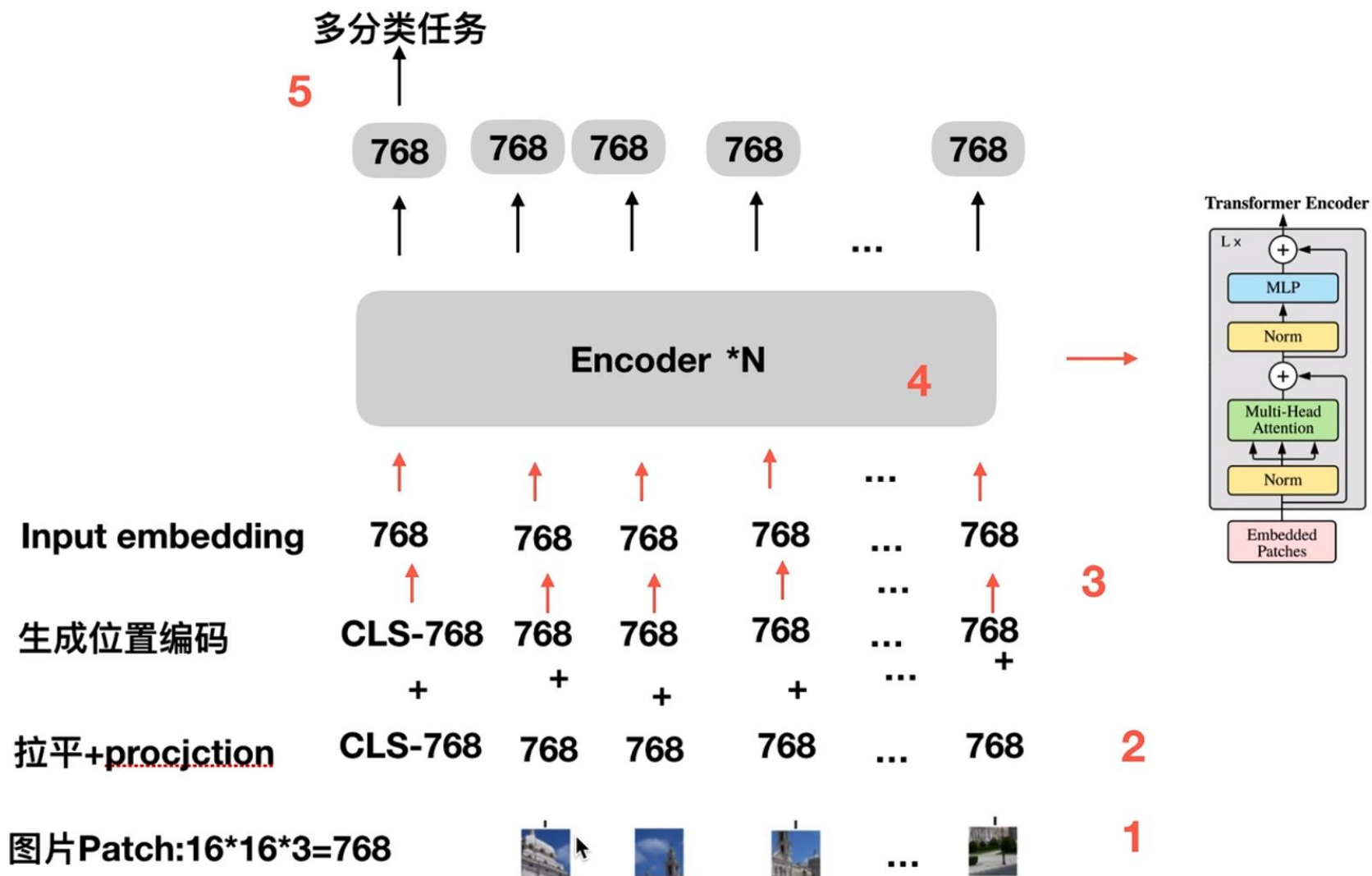
2D(分别计算行和列的编码,然后求和)的效果还不如1D的每一层都加共享的位置编码也没啥太大用

		最开始	每一层都加入 而且独立训练	每一层都加入 但是参数共享
	Pos. Emb.	Default/Stem	Every Layer	Every Layer-Shared
没有位置编码	No Pos. Emb.	0.61382	N/A	N/A
一维位置编码	1-D Pos. Emb.	0.64206	0.63964	0.64292
二维位置编码	2-D Pos. Emb.	0.64001	0.64046	0.64022
相对位置编码	Rel. Pos. Emb.	0.64032	N/A	N/A

2.模型介绍

15

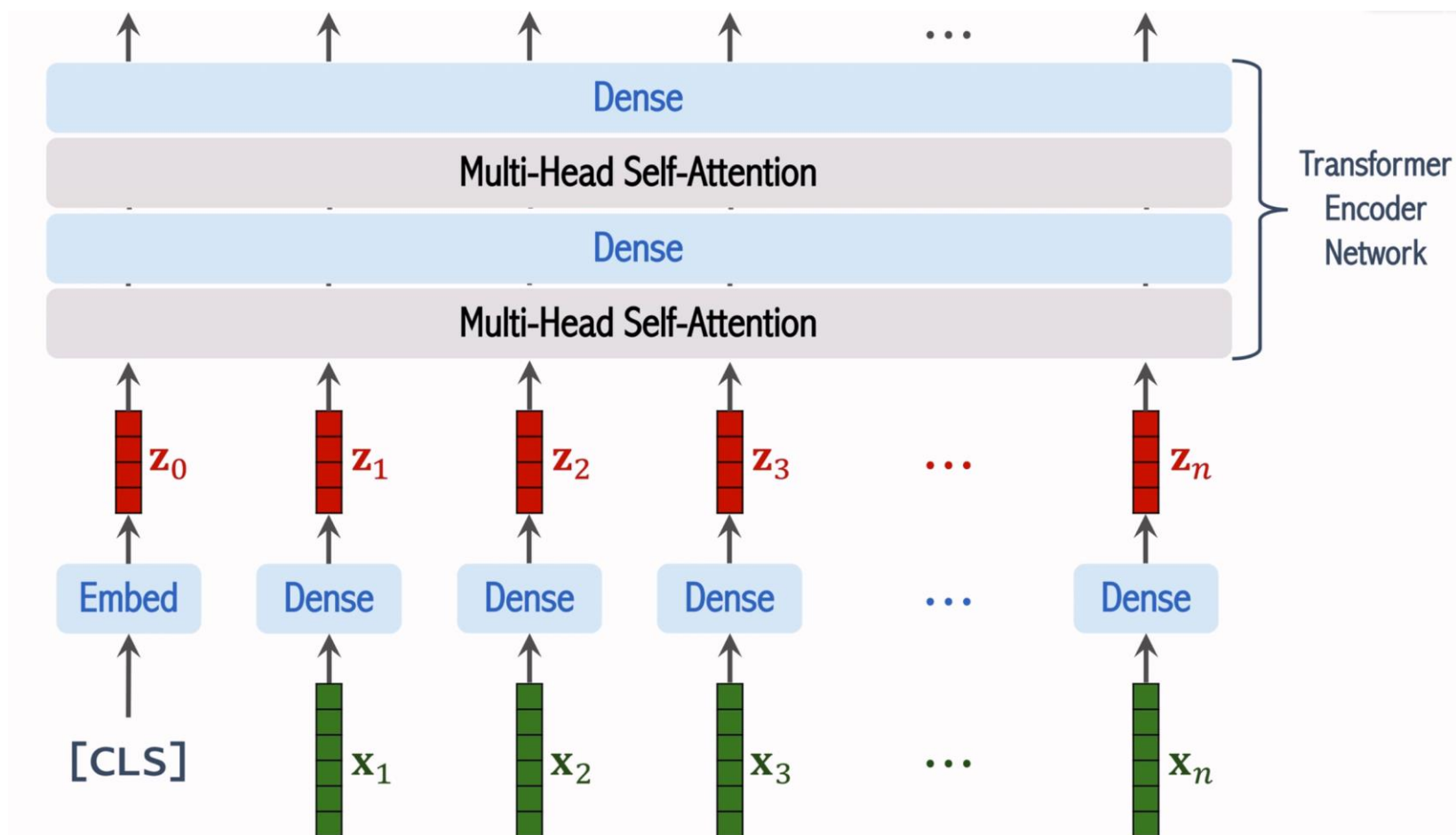
位置编码



2.模型介绍

16

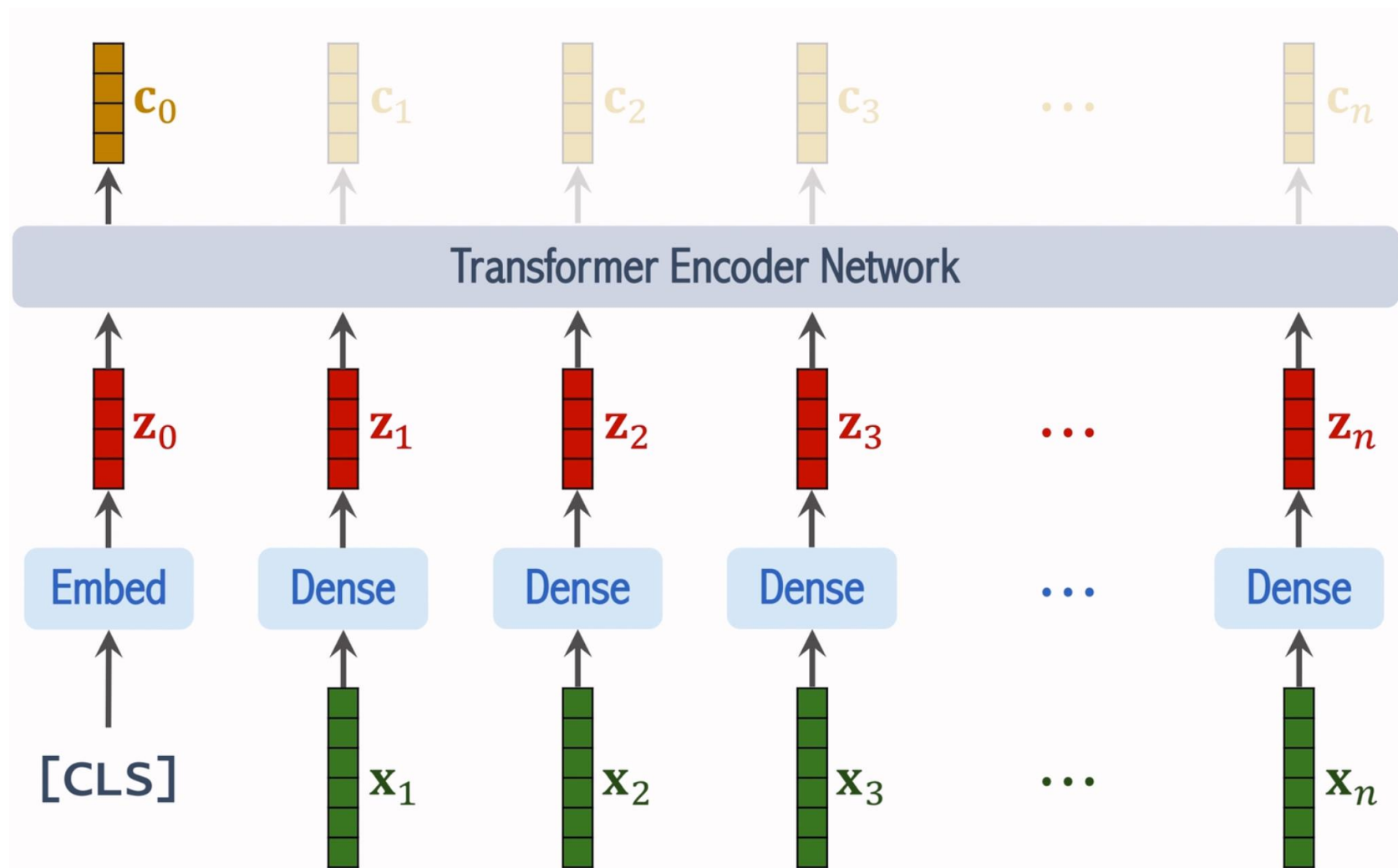
将 3) 的结果喂入标准 Transformer 的 encoder 中作者将类别作为一个可学习的 patch (z_0) 输入模型, 与图像的 patch+pos 信息作为 multi-head attention 的输入。
可以叠加多层 encoder:



2.模型介绍

17

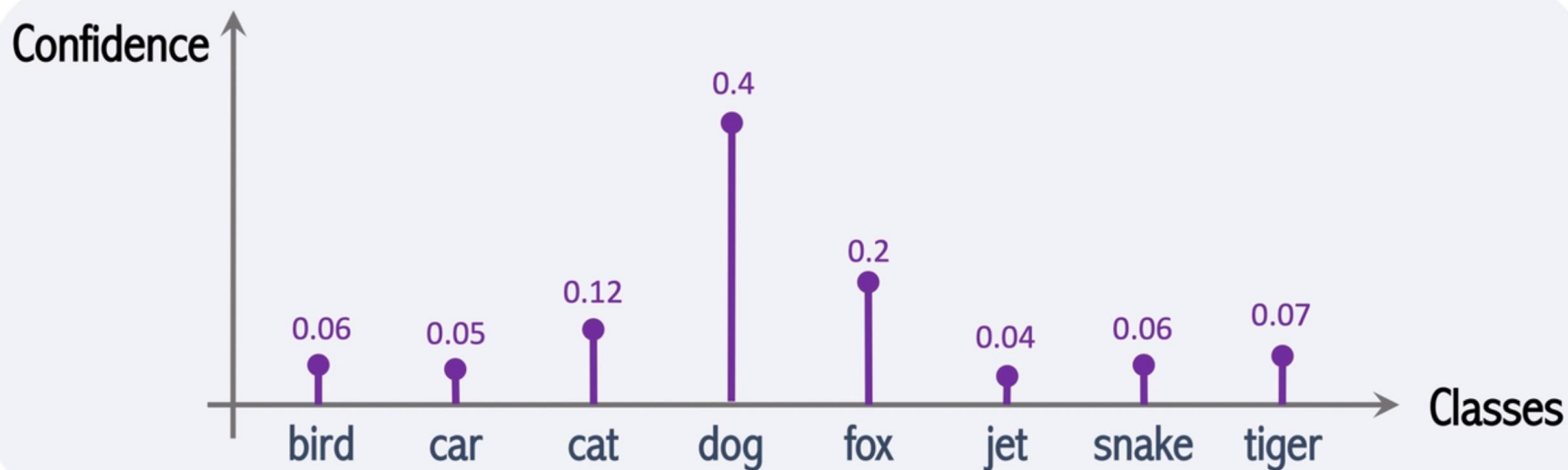
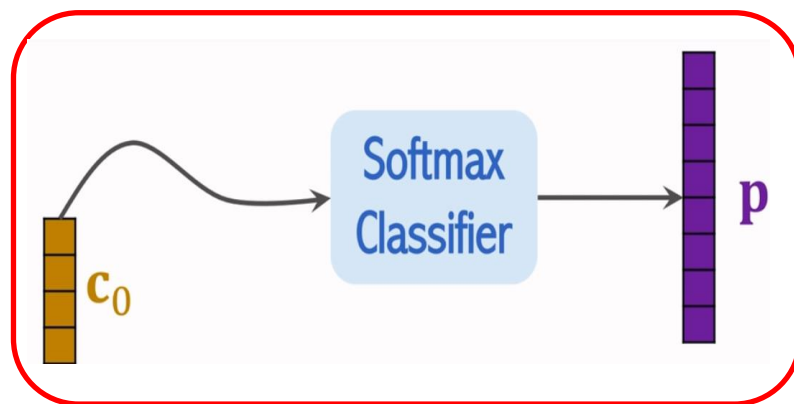
将encoder得到的结果
输入分类层
encoder 会输出多个上
下文向量，对于图像分
类，只需要 c_0 。



1.背景知识

18

将encoder得到的结果
输入分类层
encoder 会输出多个上
下文向量，对于图像分
类，只需要 c_0 。

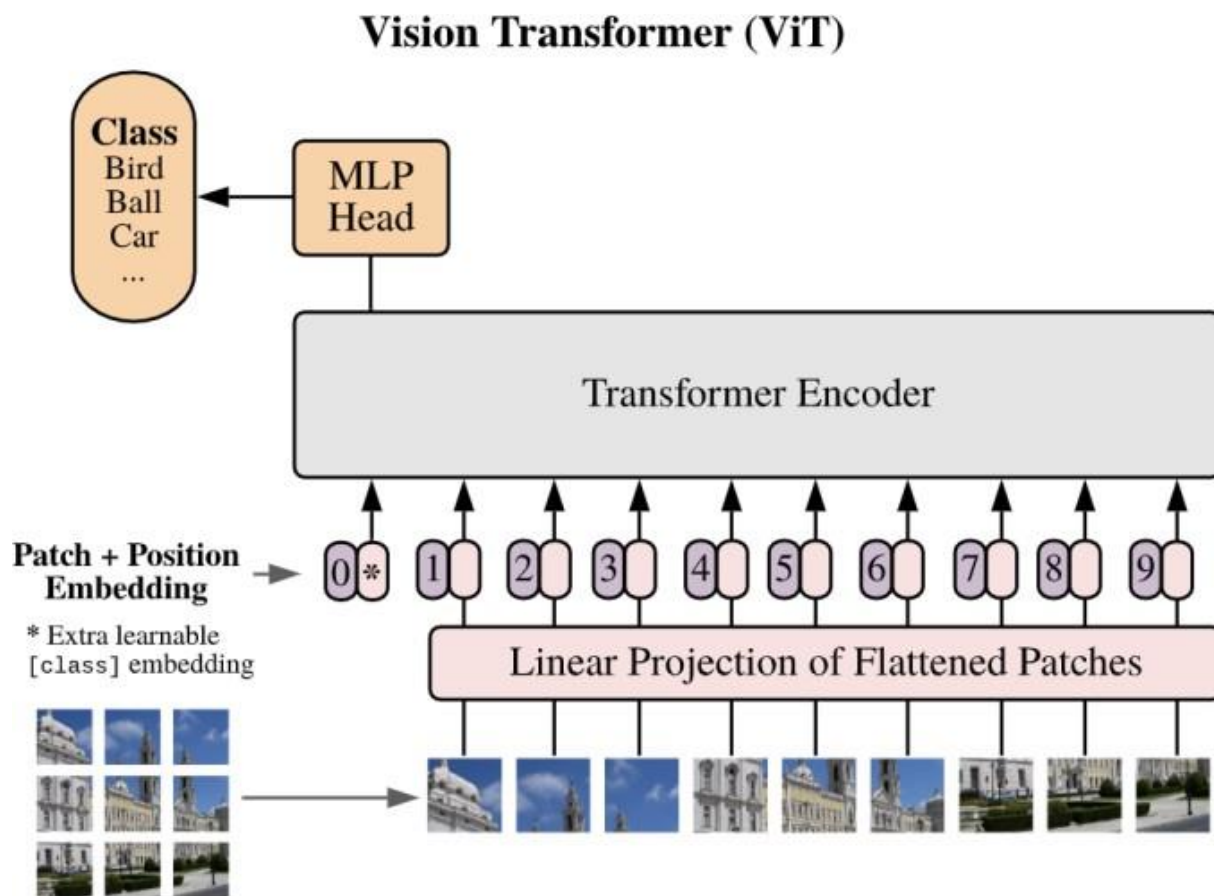


2.模型介绍

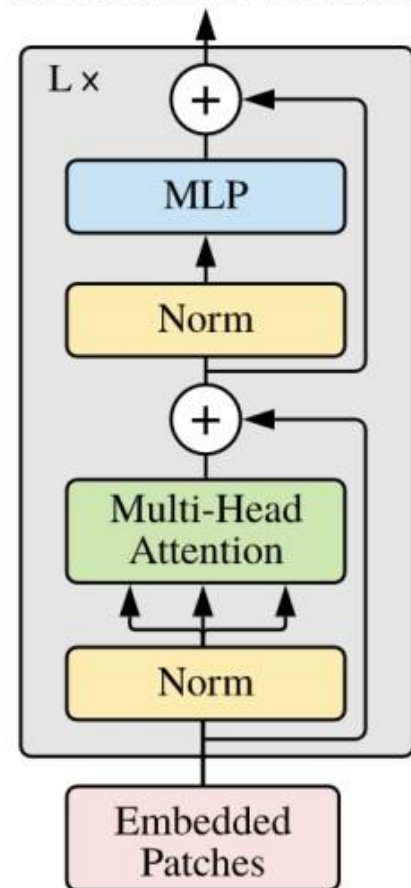
19

模型框架

最简洁的Vision Transformer模型，先将图片分成16x16的patch块，送入transformer encoder，第一个cls token的输出送入mlp head得到预测结果。



Transformer Encoder



2.模型介绍

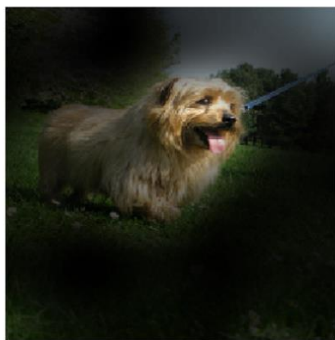
20

来自输入空间的注意力表达

输入



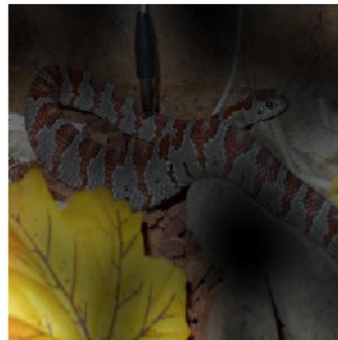
注意力



输入



注意力



输入



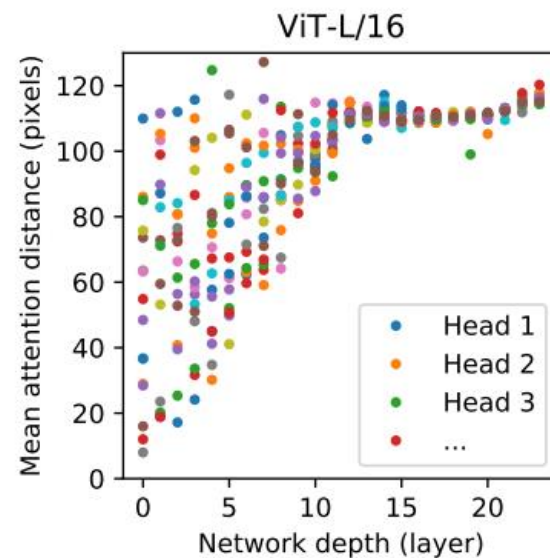
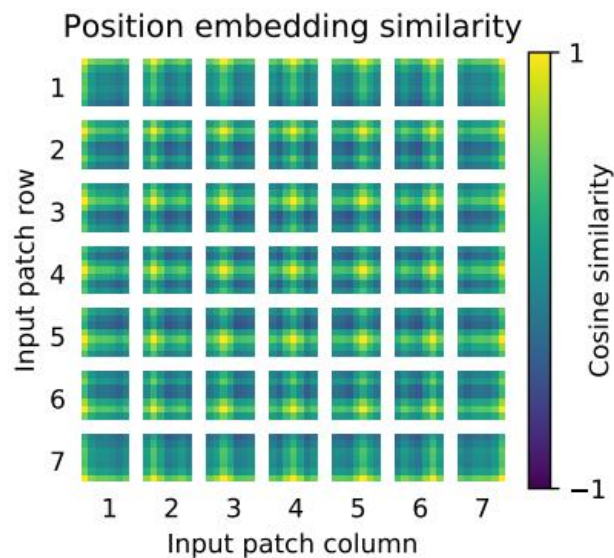
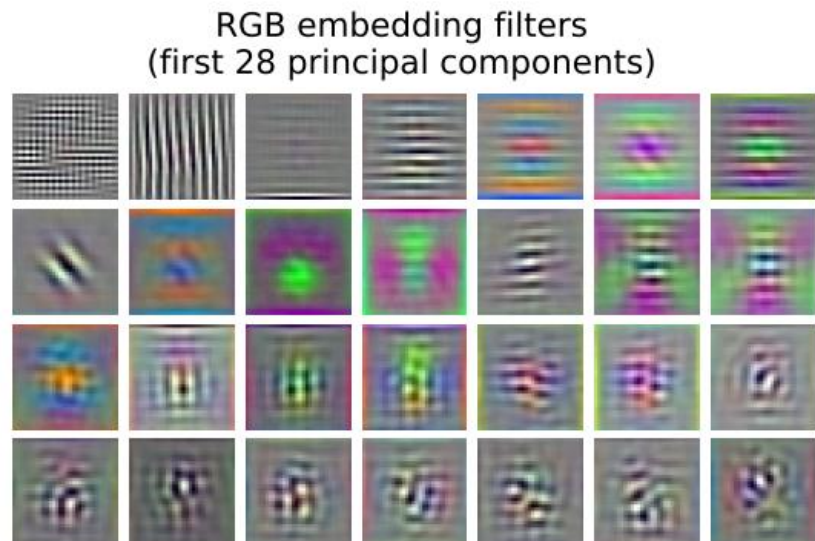
注意力



2.模型介绍

21

左图展示了模型学习到的图嵌入，中图展示了学习到的位置嵌入，右图展示了不同层注意力的平均距离。

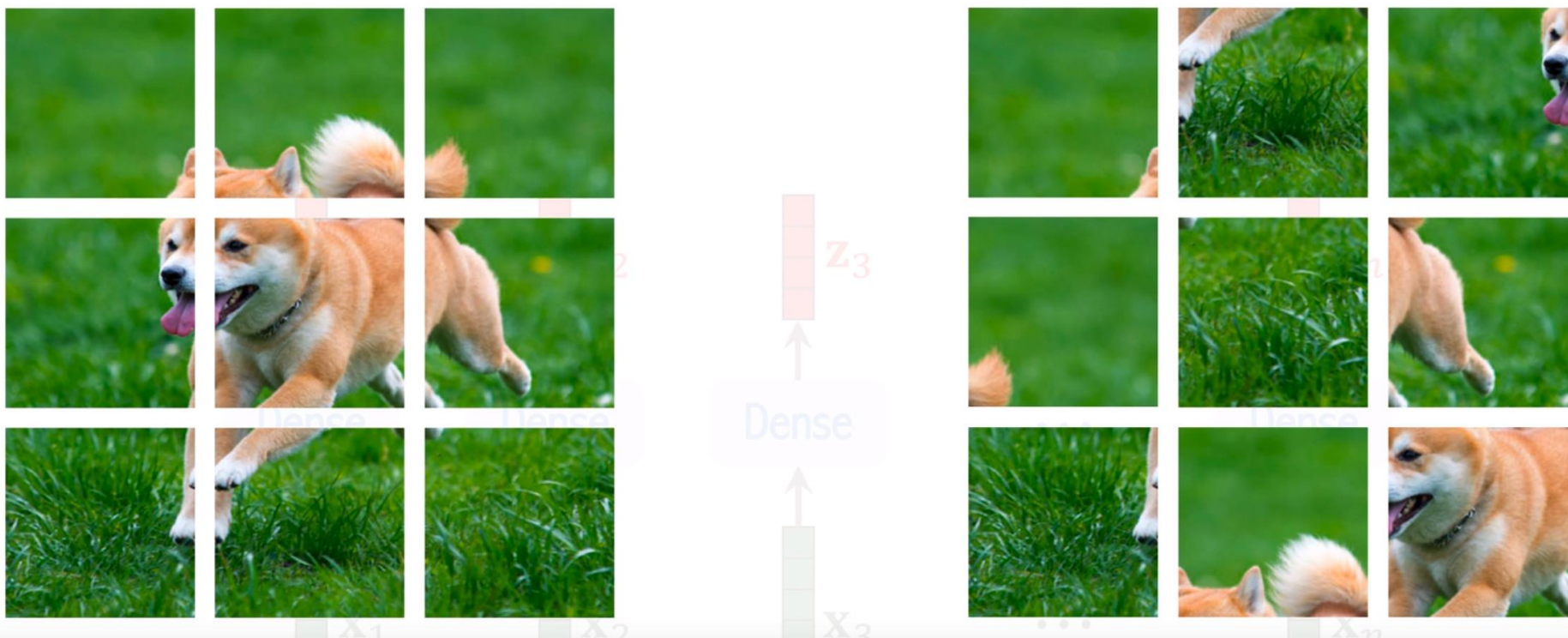


2.模型介绍

22

加入位置信息的原因

如下图所示，将左图的patch打乱，则两个图是不同的，但对于Transformer的最后一层来说会得到相同的特征(认为是一个图)，为了让其能够识别是两个图，加入位置信息(使两个图不一样)。



2.模型介绍

23

Patch 打平的具体做法

标准Transformer的输入是1D序列，对于图像 $\mathbf{x} \in R^{H*W*C}$ ，将其reshape成 $\mathbf{x}_p \in R^{N*(P^2 \cdot C)}$ 的序列。

P是patch的大小；(H,W)是图像的高和宽；C是图像通道数； $N = HW / P^2$ ，即patch的个数。

3.模型训练策略

24

01 背景知识

02 模型介绍

03 模型训练策略

04 模型的缺点与改进

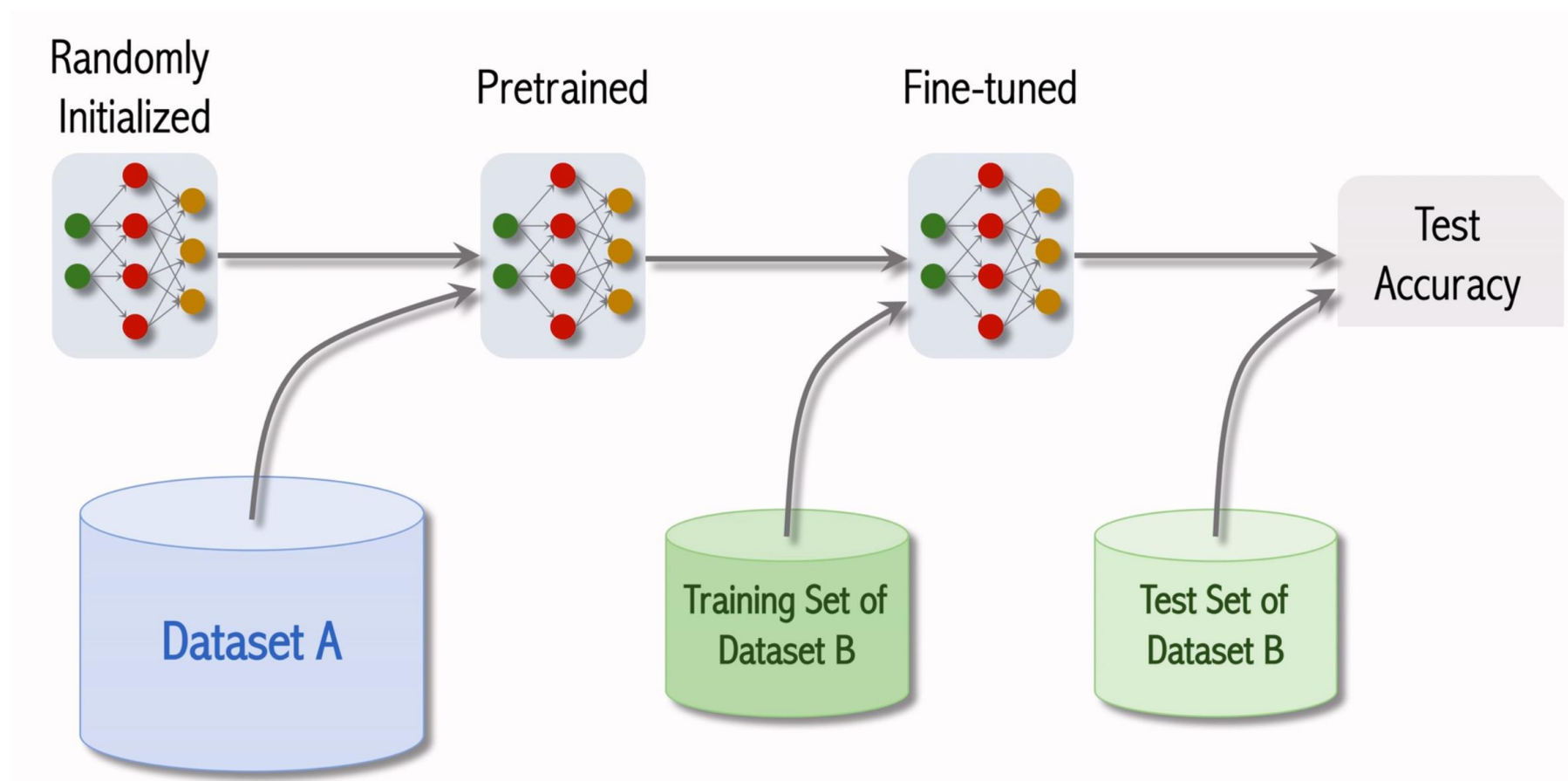
05 模型的代码实现

3.模型训练策略

25

训练策略

模型在Dataset A上预训练，在Dataset B上微调，在Dataset B上评估



3.模型训练策略

26

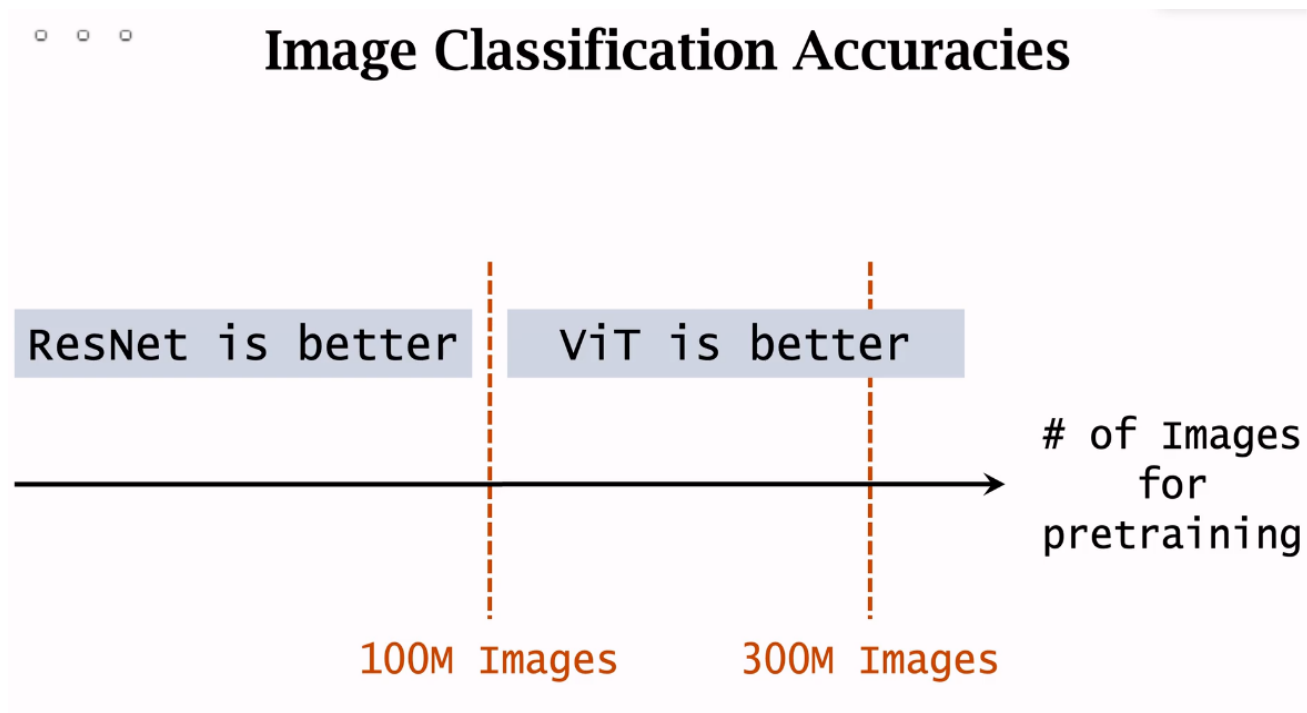
数据集介绍

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

在ImageNet(small)预训练, ViT的效果低于Resnet。

在ImageNet-21K(medium)预训练, ViT的效果接近Resnet。

在JFT(large)预训练, ViT的效果优于Resnet。



4.模型的缺点与改进

27

01 背景知识

02 模型介绍

03 模型训练策略

04 模型的缺点与改进

05 模型的代码实现

4.模型缺点与改进

28

ViT缺点

Vision Transformer比CNN具有更少的图像特异性归纳偏差。

在CNN中，局部性、二维邻域结构和平移等方差被融入到整个模型的每一层中。

在ViT中，只有MLP层是局部的、平移等变的，而自注意层是全局的。

二维邻域结构的使用非常少：在模型的开始通过将图像分割成小块，在微调时调整不同分辨率图像的位置嵌入。

除此之外，初始化时的位置嵌入不携带关于patch二维位置的信息，并且patch之间的所有空间关系都需要从头学习。

4.模型缺点与改进

29

改进

作为原始图像块的替代方法，输入序列可以由CNN的特征图形成。

在该混合模型中，将patch嵌入投影E应用于从CNN feature map中提取的patch。

作为一种特殊情况，patches的空间大小可以是 1×1 ，这意味着输入序列是通过简单地打平feature map的空间维度并投射到Transformer维度来获得的。如前所述，增加分类输入嵌入和位置嵌入。

5.模型的代码实现

30

01 背景知识

02 模型介绍

03 模型训练策略

04 模型的缺点与改进

05 模型的代码实现

5. 模型的代码实现

31

主要思路

- 一个图片224x224，分成了49个32x32的patch；
- 对这么多的patch做embedding，成49个128向量；
- 再拼接一个cls_tokens，变成50个128向量；
- 再加上pos_embedding，还是50个128向量；
- 这些向量输入到transformer中进行自注意力的特征提取；
- 输出的是50个128向量，然后对这个50个求均值，变成一个128向量；
- 然后线性层把128维变成2维从而完成二分类任务的transformer模型。

5. 模型的代码实现

32

```
import torch
from vit_pytorch import ViT

v = ViT(
    image_size = 256,
    patch_size = 32,
    num_classes = 1000,
    dim = 1024,
    depth = 6,
    heads = 16,
    mlp_dim = 2048,
    dropout = 0.1,
    emb_dropout = 0.1
)

img = torch.randn(1, 3, 256, 256)

preds = v(img)
print(preds.shape) # 1000, 与ViT定义的num_classes一致
```

image_size: int 类型参数, 图片大小。如果您有矩形图像, 请确保图像尺寸为宽度和高度的最大值

patch_size: int 类型参数, patches数目。

image_size 必须能够被 patch_size整除。

num_classes: int 类型参数, 分类数目。

dim: int 类型参数, 线性变换nn.Linear(..., dim)后输出张量的尺寸。

depth: int 类型参数, Transformer模块的个数。

heads: int 类型参数, 多头注意力中“头”的个数。

mlp_dim: int 类型参数, 多层感知机中隐藏层的神经元个数。

channels: int 类型参数, 输入图像的通道数, 默认为3。

dropout: float类型参数, Dropout几率, 取值范围为[0, 1], 默认为0。

emb_dropout: float类型参数, 进行Embedding操作时Dropout几率, 取值范围为[0, 1], 默认为0。

pool: string类型参数, 取值为 cls或者 mean。

1. <https://jalammarm.github.io/illustrated-transformer>
2. <https://www.bilibili.com/video/BV18Q4y1o7NY>
3. Dosovitskiy. An image is worth 16×16 words: transformers for image recognition at scale. In ICLR.
4. 唐宇迪, <https://www.bilibili.com/>
5. <https://www.bilibili.com/video/BV1Uu411o7oY>

谢谢!

