基于Transformer的图像分类任务及其与预训练模型的性能对比研究

PB22000184 李乐琪

一、背景介绍

Transformer模型 最初由Vaswani等人在《Attention is All You Need》中提出，主要用于自然语言处理（NLP）任务。由于其独特的注意力机制和并行化能力，Transformer模型在多个NLP任务中取得了显著的效果。近年来，研究者们开始尝试将Transformer应用于计算机视觉任务，提出了视觉Transformer（Vision Transformer，ViT）等模型，进一步拓宽了Transformer模型的应用领域。

CIFAR-10（Canadian Institute for Advanced Research - 10）数据集是计算机视觉领域中常用的基准数据集之一，用于评估图像分类算法的性能。由于图像分辨率相对较低（仅32x32像素），这使得任务相对简单，但也因此要求模型具备良好的泛化能力。评估模型性能通常使用分类准确率（accuracy）作为主要指标，即模型在测试集上正确分类的比例。

> CIFAR-10数据集由60000张32x32彩色图片组成，分为10个类别，每个类别6000张图片。这些类别包括飞机、汽车、鸟类、猫、鹿、狗、青蛙、马、船和卡车。数据集被分为训练集和测试集，训练集包含50000张图片，测试集包含10000张图片。

本研究的目标是复现《Attention is All You Need》中提出的Transformer模型，并基于此实现基础的Vision Transformer（ViT）模型，进而在CIFAR-10图像分类任务上进行训练和验证。同时，我们将所实现的ViT模型与三个流行的预训练模型进行对比，包括ResNet18、普通ViT和增强版ViT。通过这些对比实验，我们旨在评估ViT模型在小规模图像数据集上的表现，并探讨其在计算机视觉任务中的潜力和优势。

二、科学技术原理

2.1 Transformer 模型

2.1.1 模型架构

编码器（Encoder）与解码器（Decoder）

- Transformer模型主要由编码器（Encoder）和解码器（Decoder）两部分组成。
- 编码器负责处理输入序列，并为每个位置生成一个上下文向量。这个向量包含了输入序列中所有位置的信息。
- 解码器使用这些编码器的输出来生成输出序列。解码器一次只生成一个词，并且使用自回归（auto-regressive）的方式生成，即过去时刻的输出作为当前时刻的输入。

自注意力机制（Self-Attention）

- Transformer模型的核心是自注意力机制。在处理输入序列时，模型会为每个输入位置计算一个向量表示。这些向量表示是通过对所有输入位置进行加权求和得到的，其中每个位置的权重由该位置与其他所有位置之间的相似度计算得出。
- 具体来说，模型将输入序列中的每个位置都视为查询（query）、键（key）和值（value）。对于每个查询，模型会计算它与所有键之间的相似度，并将这些相似度转换为权重。然后，模型使用这些权重对所有值进行加权求和，以获得该查询的向量表示。

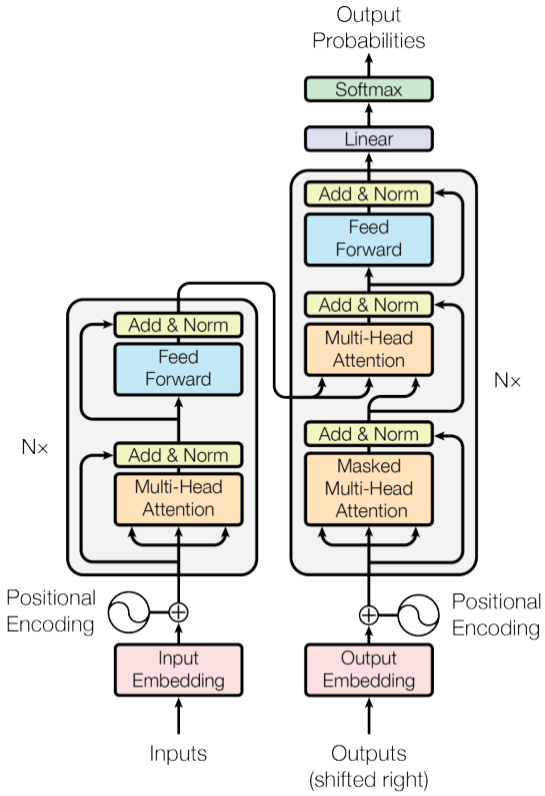


Figure 1: The Transformer - model architecture.

多头注意力（Multi-Head Attention）

- 为了捕捉输入序列中不同方面的信息，Transformer使用了多头注意力机制。这意味着模型将输入序列分割成多个“头”，每个头独立地执行自注意力操作，然后将结果合并起来。

前馈神经网络（Feed-Forward Neural Network）

- 除了自注意力层外，Transformer的每个编码器和解码器层还包含一个前馈神经网络。这个网络对自注意力层的输出进行进一步处理，并产生该层的最终输出。

残差连接（Residual Connections）和层归一化（Layer Normalization）

- 为了使模型能够训练得更深，Transformer在层之间使用了残差连接和层归一化。这种设计有助于梯度的反向传播，并防止模型在训练过程中出现过拟合。

2.1.2 训练策略

并行化训练：由于Transformer完全基于注意力机制，因此它可以并行地处理输入序列中的每个位置。这使得模型在训练时能够更快地收敛，并提高了训练效率。

优化器：在训练过程中，通常使用如Adam等优化器来更新模型的参数。

损失函数：对于诸如机器翻译等任务，Transformer模型通常使用交叉熵损失函数来评估模型的性能。

正则化：为了防止过拟合，模型可能会使用诸如dropout等正则化技术。

2.2 ResNet18

ResNet18（*Residual Network-18*）是深度残差网络的一种经典架构，旨在解决深层神经网络训练中的梯度消失和退化问题。ResNet 通过引入残差块（Residual Block）来实现更深的网络结构，有效地减少了训练过程中的信息损失和梯度消失，从而提高了网络的收敛速度和泛化能力。

- **架构细节**

ResNet18 整体结构相对简洁明了，共包含18个层次，主要由卷积层、池化层和全连接层构成。其核心是残差块，每个残差块包含两个3x3卷积层，中间有批量归一化（Batch Normalization）和ReLU激活函数。此外，每个残差块的输入通过跳跃连接（skip connection）直接添加到后续卷积的输出，形成残差学习的机制。

2.4 ViT（Vision Transformer）

Vision Transformer（ViT）是一种基于Transformer架构的图像分类模型。与传统的卷积神经网络（CNN）不同，ViT将图像分割为固定大小的小块（patch），并将这些patch视为序列输入到Transformer模型中，利用自注意力机制捕捉图像的全局信息，从而实现图像分类任务。

- **架构细节**

ViT的核心思想是将图像视为一系列的图像块（patches），然后将这些块转换为序列数据输入到Transformer模型中。具体来说，ViT包含一个图像块的嵌入层（patch embedding layer），将每个图像块映射到一个低维度的特征向量，然后将这些特征向量作为序列输入到Transformer的编码器中。在Transformer编码器中，使用多头自注意力机制来学习图像块之间的关系，并通过位置编码来处理图像块的位置信息。最后，通过全局平均池化（global average pooling）将序列输出转换为图像级别的表示，然后经过全连接层进行分类。

2.5 ViT增强（ViT+ToMe）

ViT增强（ViT+ToMe）是一种对Vision Transformer（ViT）模型的增强版本，通过结合Transformer和Token-merge（ToMe）机制来提高模型的性能和泛化能力。ToMe机制通过将多个相邻的token合并为一个token，从而降低了模型的计算复杂度和内存消耗，同时保持了原始图像信息的表达能力。

三、具体实现

3.1 Transform 复现

由开源代码在 Multi30k 数据集上训练得出。

3.2 基于Transform的基础 Vit 实现

针对CIFAR-10图像分类任务，每张输入图片的尺寸为 $224 \times 224 \times 3$，其中 224×224 表示每张图片的像素点个数，每个像素点由三个通道（红、绿、蓝）组成，每个通道的取值范围为0到255。为了适应Transformer模型和Vision Transformer（ViT），我们采取以下步骤预处理图像数据：

1. **图片分块处理：** 将每张图片分割为 $14 \times 14 = 196$ 个 16×16 的小块（patch）。

2. **线性映射转换：** 每个小块 $16 \times 16 \times 3$ 的像素信息被展平为一个 1×768 的向量。这些向量经过线性变换，与一个 768×768 的权重矩阵相乘，得到一个 1×768 的特征向量。
3. **矩阵组合：** 所有196个小块生成的特征向量被组合成一个 196×768 的矩阵，类似于Transformer中处理文本时的“句向量”。
4. **特殊处理：** 在这个矩阵中添加一个额外的全零行（1×768），形成一个 197×768 的输入矩阵，作为Transformer模型的输入进行训练。
5. **分类输出：** 经过训练后，将全零行提出，经过一个 768×10 的输出层，得到一个 1×10 的输出向量，表示对应10个类别的分类概率。

这种处理方式以类似Vit的方式，将图像数据转换为适合Transformer模型处理的格式，利用Transformer的自注意力机制有效地捕捉图像中的空间信息和像素关系，以实现高效的图像分类任务。

3.3 ResNet 18、Vit与ViT增强

导入预训练模型并设置参数。

四、实验结果与分析

4.1 硬件和时间表

本文在远程SSH服务器上进行代码的运行，使用的硬件设备为：

```
========实例配置========
核数：16
内存：30 GB
磁盘：23% 6.9G/30G
显卡：NVIDIA GeForce RTX 3090, 1
```

训练 Transform 模型大约需要20分钟，TRAIN_SIZE = 20000的基础Vit模型大约需要 15分钟。

4.2 Attention 复现结果

本次实验中使用 Multi30k 数据集复现了论文结果，以下是第零周期（Epoch 0）和最后一个周期（Epoch 7）的训练日志输出：

```
Vocabulary sizes:
8317
6384
Train worker process using GPU: 0 for training
[GPU0] Epoch 0 Training ====
Epoch Step:      1 | Accumulation Step:   1 | Loss:   7.61 | Tokens / Sec:  2465.8 | Learning Rate: 5.4e-07
Epoch Step:     41 | Accumulation Step:   5 | Loss:   7.38 | Tokens / Sec:  3829.8 | Learning Rate: 1.1e-05
Epoch Step:     81 | Accumulation Step:   9 | Loss:   6.95 | Tokens / Sec:  4032.1 | Learning Rate: 2.2e-05
Epoch Step:    121 | Accumulation Step:  13 | Loss:   6.60 | Tokens / Sec:  3973.4 | Learning Rate: 3.3e-05
Epoch Step:    161 | Accumulation Step:  17 | Loss:   6.41 | Tokens / Sec:  3978.0 | Learning Rate: 4.4e-05
Epoch Step:    201 | Accumulation Step:  21 | Loss:   6.22 | Tokens / Sec:  3957.2 | Learning Rate: 5.4e-05
Epoch Step:    241 | Accumulation Step:  25 | Loss:   6.16 | Tokens / Sec:  4010.9 | Learning Rate: 6.5e-05
Epoch Step:    281 | Accumulation Step:  29 | Loss:   6.02 | Tokens / Sec:  3776.2 | Learning Rate: 7.6e-05
Epoch Step:    321 | Accumulation Step:  33 | Loss:   5.88 | Tokens / Sec:  3951.3 | Learning Rate: 8.7e-05
Epoch Step:    361 | Accumulation Step:  37 | Loss:   5.60 | Tokens / Sec:  4089.8 | Learning Rate: 9.7e-05
Epoch Step:    401 | Accumulation Step:  41 | Loss:   5.33 | Tokens / Sec:  4241.4 | Learning Rate: 1.1e-04
Epoch Step:    441 | Accumulation Step:  45 | Loss:   5.19 | Tokens / Sec:  3891.9 | Learning Rate: 1.2e-04
Epoch Step:    481 | Accumulation Step:  49 | Loss:   4.81 | Tokens / Sec:  3705.6 | Learning Rate: 1.3e-04
Epoch Step:    521 | Accumulation Step:  53 | Loss:   4.57 | Tokens / Sec:  4065.9 | Learning Rate: 1.4e-04
Epoch Step:    561 | Accumulation Step:  57 | Loss:   4.51 | Tokens / Sec:  4114.3 | Learning Rate: 1.5e-04
Epoch Step:    601 | Accumulation Step:  61 | Loss:   4.32 | Tokens / Sec:  4112.3 | Learning Rate: 1.6e-04
Epoch Step:    641 | Accumulation Step:  65 | Loss:   4.32 | Tokens / Sec:  4081.6 | Learning Rate: 1.7e-04
Epoch Step:    681 | Accumulation Step:  69 | Loss:   4.19 | Tokens / Sec:  4130.6 | Learning Rate: 1.8e-04
Epoch Step:    721 | Accumulation Step:  73 | Loss:   4.26 | Tokens / Sec:  4119.8 | Learning Rate: 1.9e-04
Epoch Step:    761 | Accumulation Step:  77 | Loss:   4.23 | Tokens / Sec:  4059.4 | Learning Rate: 2.0e-04
Epoch Step:    801 | Accumulation Step:  81 | Loss:   4.02 | Tokens / Sec:  4085.5 | Learning Rate: 2.2e-04
Epoch Step:    841 | Accumulation Step:  85 | Loss:   4.08 | Tokens / Sec:  4113.0 | Learning Rate: 2.3e-04
Epoch Step:    881 | Accumulation Step:  89 | Loss:   4.08 | Tokens / Sec:  4152.6 | Learning Rate: 2.4e-04
| ID | GPU | MEM |
------------------
|  0 | 11% | 29% |
[GPU0] Epoch 0 Validation ====
```

```
[GPU0] Epoch 7 Training ====
Epoch Step:      1 | Accumulation Step:   1 | Loss:   1.00 | Tokens / Sec:  3479.8 | Learning Rate: 5.5e-04
Epoch Step:     41 | Accumulation Step:   5 | Loss:   0.89 | Tokens / Sec:  4120.2 | Learning Rate: 5.5e-04
Epoch Step:     81 | Accumulation Step:   9 | Loss:   0.89 | Tokens / Sec:  4062.8 | Learning Rate: 5.5e-04
Epoch Step:    121 | Accumulation Step:  13 | Loss:   0.91 | Tokens / Sec:  4037.1 | Learning Rate: 5.5e-04
Epoch Step:    161 | Accumulation Step:  17 | Loss:   0.97 | Tokens / Sec:  3650.3 | Learning Rate: 5.5e-04
Epoch Step:    201 | Accumulation Step:  21 | Loss:   1.20 | Tokens / Sec:  3703.5 | Learning Rate: 5.5e-04
Epoch Step:    241 | Accumulation Step:  25 | Loss:   1.05 | Tokens / Sec:  3798.1 | Learning Rate: 5.4e-04
Epoch Step:    281 | Accumulation Step:  29 | Loss:   0.99 | Tokens / Sec:  4196.5 | Learning Rate: 5.4e-04
Epoch Step:    321 | Accumulation Step:  33 | Loss:   0.97 | Tokens / Sec:  4178.9 | Learning Rate: 5.4e-04
Epoch Step:    361 | Accumulation Step:  37 | Loss:   1.07 | Tokens / Sec:  4177.1 | Learning Rate: 5.4e-04
Epoch Step:    401 | Accumulation Step:  41 | Loss:   1.00 | Tokens / Sec:  4085.0 | Learning Rate: 5.4e-04
Epoch Step:    441 | Accumulation Step:  45 | Loss:   1.01 | Tokens / Sec:  3785.2 | Learning Rate: 5.4e-04
Epoch Step:    481 | Accumulation Step:  49 | Loss:   1.13 | Tokens / Sec:  3722.8 | Learning Rate: 5.3e-04
Epoch Step:    521 | Accumulation Step:  53 | Loss:   0.97 | Tokens / Sec:  3926.1 | Learning Rate: 5.3e-04
Epoch Step:    561 | Accumulation Step:  57 | Loss:   1.15 | Tokens / Sec:  4078.5 | Learning Rate: 5.3e-04
Epoch Step:    601 | Accumulation Step:  61 | Loss:   0.91 | Tokens / Sec:  4113.8 | Learning Rate: 5.3e-04
Epoch Step:    641 | Accumulation Step:  65 | Loss:   1.04 | Tokens / Sec:  4229.8 | Learning Rate: 5.3e-04
Epoch Step:    681 | Accumulation Step:  69 | Loss:   1.35 | Tokens / Sec:  4072.1 | Learning Rate: 5.3e-04
Epoch Step:    721 | Accumulation Step:  73 | Loss:   1.01 | Tokens / Sec:  4147.8 | Learning Rate: 5.3e-04
Epoch Step:    761 | Accumulation Step:  77 | Loss:   0.82 | Tokens / Sec:  4159.1 | Learning Rate: 5.2e-04
Epoch Step:    801 | Accumulation Step:  81 | Loss:   1.03 | Tokens / Sec:  4116.5 | Learning Rate: 5.2e-04
Epoch Step:    841 | Accumulation Step:  85 | Loss:   1.11 | Tokens / Sec:  4164.7 | Learning Rate: 5.2e-04
Epoch Step:    881 | Accumulation Step:  89 | Loss:   1.13 | Tokens / Sec:  4209.2 | Learning Rate: 5.2e-04
| ID | GPU | MEM |
------------------
|  0 | 33% | 33% |
[GPU0] Epoch 7 Validation ====
```

可以从结果中明显观察到：

1. 损失 *Loss* 有持续降低的趋势，初始为7.61，在 *Epoch1*中降到4.08，在训练结束降至1.13，正确率显著提高，表明模型在 *Epoch7*的训练后能够更好地拟合验证集；同时 *Loss* 在 *Epoch7* 中无规律波动在 1 附近，说明模型已经进行充分训练，得到最优结果。
2. 学习率 Learning Rate 先增后减，在 Epoch 3 Step 281 达到峰值 8.1×10^{-4} ，符合设定：

```
[GPU0] Epoch 3 Training ====
......
Epoch Step:    241 | Accumulation Step:  25 | Loss:   1.89 | Tokens / Sec: 4329.7 | Learning Rate: 8.0e-04
Epoch Step:    281 | Accumulation Step:  29 | Loss:   2.10 | Tokens / Sec: 4276.3 | Learning Rate: 8.1e-04
Epoch Step:    321 | Accumulation Step:  33 | Loss:   1.82 | Tokens / Sec: 4233.8 | Learning Rate: 8.0e-04
Epoch Step:    361 | Accumulation Step:  37 | Loss:   1.93 | Tokens / Sec: 4338.3 | Learning Rate: 8.0e-04
......
```

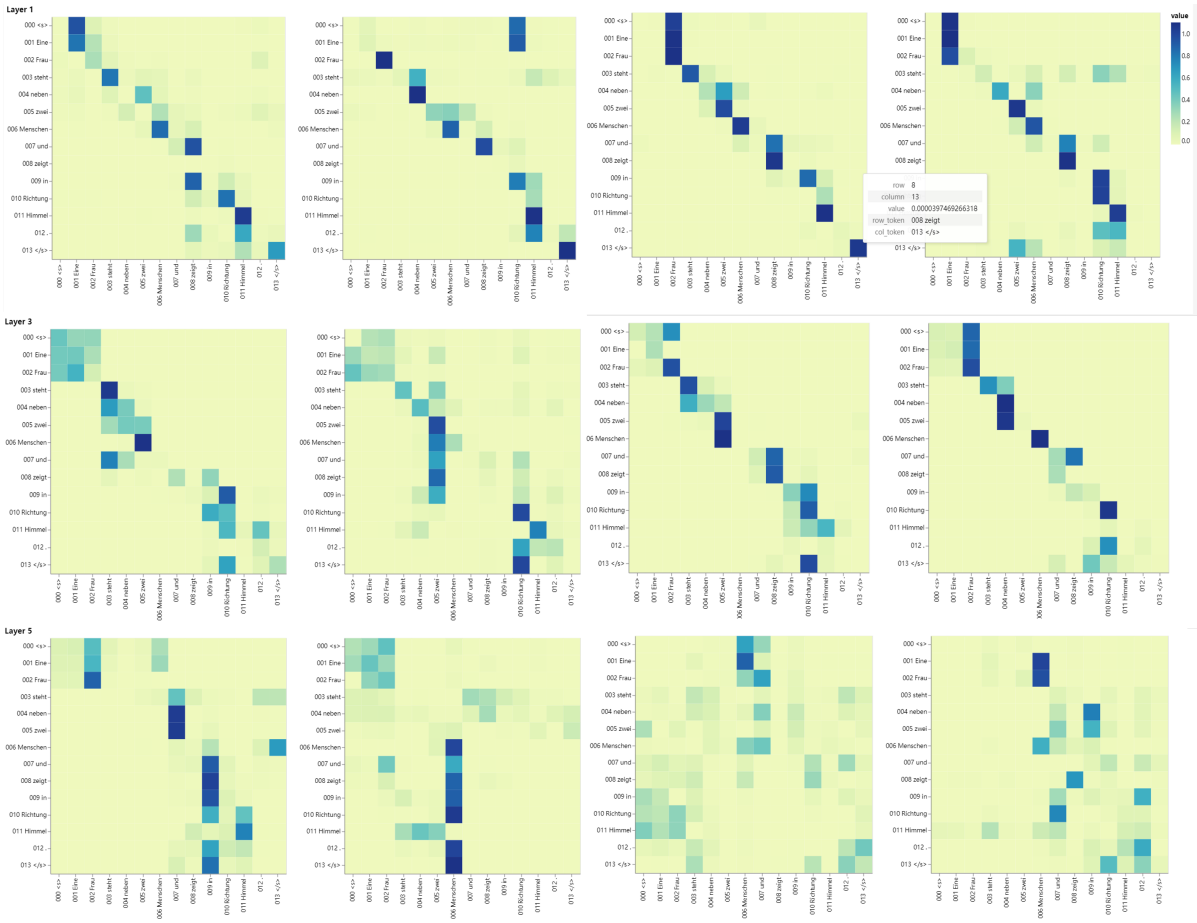
4.3 Attention 可视化

- 编码器注意力可视化

```
True
Preparing Data ...
Loading Trained Model ...
Checking Model Outputs:

Example 0 ========

Source Text (Input)        : <s> Eine Frau steht neben zwei Menschen und zeigt in Richtung Himmel . </s>
Target Text (Ground Truth) : <s> A woman standing next to two people is pointing to the sky . </s>
Model Output               : <s> A woman stands by two people and pointing to the sky . </s>
```



> 初始时 Layer1 图中深色色块主要集中在对角线上，说明注意力主要集中在自身；随着Layer的增加，深色色块逐渐变得模糊且无规律，说明注意力随着模型参数的迭代更新不断变化。

4.4 基础 Vit 与图像分类任务

训练集个数为500时：

```
TRAIN_SIZE = 500
Epoch 1 Accuracy: 21.10%
train_time= 6.15s; eval_time= 2.20s
Epoch 2 Accuracy: 21.90%
train_time= 2.56s; eval_time= 2.22s
......
train_time= 2.53s; eval_time= 2.23s
Epoch 10 Accuracy: 23.60%
train_time= 2.55s; eval_time= 2.21s
```

训练集个数为10000时：

```
TRAIN_SIZE = 10000
- - - - - - - - - - - - - - - - Epoch 1 Accuracy: 22.00%
train_time= 46.55s; eval_time= 4.48s
- - - - - - - - - - - - - - - - Epoch 2 Accuracy: 26.35%
train_time= 41.97s; eval_time= 4.52s
......
- - - - - - - - - - - - - - - - Epoch 9 Accuracy: 35.30%
train_time= 42.56s; eval_time= 4.52s
- - - - - - - - - - - - - - - - Epoch 10 Accuracy: 36.05%
train_time= 42.51s; eval_time= 4.55s
```

训练集个数为20000时：

```
TRAIN_SIZE = 20000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Epoch 1
Accuracy: 29.45%
train_time= 86.52s; eval_time= 5.23s
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Epoch 2
Accuracy: 28.80%
train_time= 83.74s; eval_time= 5.19s
......
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Epoch 9
Accuracy: 34.20%
train_time= 84.27s; eval_time= 5.20s
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Epoch 10
Accuracy: 37.00%
train_time= 84.26s; eval_time= 5.19s
```

可以观察到，随着训练集数据的增加，准确率逐步提升。在训练数据很少时，准确率很低；在20000数据集时依然未超过40%，但并未收敛，说明随着训练数据集的不断扩大能进一步提高准确率。

4.5 预训练模型与图像分类任务

以下训练均取训练集个数为500

ResNet18

```
# ResNet18
- Epoch 1 Accuracy: 9.20%
train_time= 4.91s; eval_time= 0.86s
- Epoch 2 Accuracy: 22.20%
train_time= 1.08s; eval_time= 0.87s
......
- Epoch 9 Accuracy: 56.80%
train_time= 1.11s; eval_time= 0.98s
Epoch 10 Accuracy: 57.60%
train_time= 1.13s; eval_time= 0.88s
```

Vit

```
# Vit
- Epoch 1 Accuracy: 83.20%
train_time= 6.23s; eval_time= 1.40s
- Epoch 2 Accuracy: 90.20%
train_time= 2.50s; eval_time= 1.40s
......
- Epoch 9 Accuracy: 91.60%
train_time= 2.51s; eval_time= 1.44s
Epoch 10 Accuracy: 91.20%
train_time= 2.51s; eval_time= 1.44s
```

Vit增强

```
# Vit增强
- Epoch 1 Accuracy: 92.80%
train_time= 7.72s; eval_time= 1.99s
- Epoch 2 Accuracy: 96.00%
train_time= 4.01s; eval_time= 2.01s
......
- Epoch 9 Accuracy: 96.00%
train_time= 4.03s; eval_time= 2.01s
Epoch 10 Accuracy: 96.20%
train_time= 4.03s; eval_time= 2.01s
```

可以观察到：

- ResNet18训练速度最快，但准确率较低，最终只有57.60%；
- Vit增强和Vit都始终保持着很高的准确率；
- Vit增强较Vit在准确率上略有下降，但训练速度显著提高。

五、创新性

- 模型复现与扩展

本次大作业不仅成功复现了《Attention is All You Need》中的Transformer模型，而且进一步实现了基础的Vision Transformer（ViT）模型,它展示了Transformer架构从自然语言处理到计算机视觉领域的泛化能力。

- 实验设计的全面性

在CIFAR-10图像分类任务上训练和验证ViT模型，并与多个流行的预训练模型（ResNet18、普通ViT和增强版ViT）进行对比，这一实验设计是全面的。通过对比实验，能够更准确地评估ViT模型在小规模图像数据集上的表现，为未来的研究提供了有价值的参考。

- 对小规模数据集性能的评估

大多数关于ViT的研究都集中在大型数据集上，如ImageNet。本次大作业专注于在CIFAR-10这样的小规模数据集上评估ViT的性能，这一选择具有创新性。通过这一研究，我们可以更好地理解ViT在小规模数据集上的表现，为资源受限环境下的视觉任务提供新的解决方案。

六、学习心得和收获

在本次大作业的实现中，我深入学习了Transformer模型和 Vit 的相关知识，并将其应用于CIFAR-10图像分类任务。这一过程中，我不仅提高了编程能力，也加深了对深度学习理论的理解和实际应用的能力，获得了丰富的学习心得和收获。

1. 深度学习理论的深入理解

   在实现Transformer模型和ViT模型的过程中，我深入学习了注意力机制的原理，理解了它们如何帮助模型捕捉输入序列中的长距离依赖关系。

2. 编程技能的提升

   在实现模型的过程中，我使用了Python的多种科学计算库，如NumPy、PyTorch、timm、tome等，有效训练了我面对复杂问题时的编程思维；同时，在配置环境的过程中，经历了一系列问题并最终解决，提高了我的问题解决能力。

3. 实验设计与执行的严谨性

   在进行CIFAR-10图像分类任务时，我精心设计了实验方案，包括数据预处理、模型训练、参数调整等步骤。这一过程中，我体会到了实验设计的严谨性和重要性，也学会了如何根据实际情况调整实验方案。

4. 对比分析的能力

   在与ResNet18、普通ViT和增强版ViT的对比实验中，我学会了如何分析和解释实验结果，通过对比分析来评估不同模型的性能优劣。这一能力对于科学研究和实际应用都非常重要。

综上，这次Python科学计算课程的大作业让我收获颇丰。我不仅提高了编程能力和深度学习理论的理解能力，还积累了宝贵的实践经验和问题解决能力。同时，我也对深度学习和计算机视觉领域产生了更浓厚的兴趣。这些收获将对我未来的学习和工作产生积极影响。

七、参考文献

[1] Python 科学计算基础 罗奇鸣

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems 30* (NIPS 2017), pp. 5998-6008.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[4] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*.

[5] Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., & Hoffman, J. (2023). TOKEN MERGING: YOUR VIT BUT FASTER. In *Proceedings of the International Conference on Learning Representations (ICLR)* . arXiv:2210.09461.