基于ViT的CIFAR-10分类任务

深度学习实验题（二）

学号：111
姓名：李明
学院：xx学院
单位：xx研究院
时间：2035年13月

中国科学院大学
University of Chinese Academy of Sciences

目录

CONTENTS

1 概述

本实验旨在使用Vision Transformer（ViT）模型解决CIFAR-10图像分类问题。CIFAR-10数据集包含10个类别的60000张32x32彩色图像，是一个广泛用于图像分类任务的基准数据集。该模型通过将输入图像分块并经过多个Transformer块进行编码，从而实现了在视觉任务中的表现。

2.1 网络结构设计

2.1.1 Patch Embedding模块：

- `PatchEmbedding`模块负责将输入图像分块为固定大小的patch，并将每个patch通过卷积层投影到嵌入空间。同时，引入一个分类token（用于表示整个图像）和位置嵌入。
- `PatchEmbedding`模块通过卷积操作将输入图像分块，并将每个patch展平后进行转置，以便按照Transformer的输入要求进行处理。位置嵌入（`pos_embed`）提供了每个patch的位置信息，而分类token（`cls_token`）则引入了整个图像的全局信息。

核心代码如下：

```python
# Patch Embedding 模块
class PatchEmbedding(nn.Module):
    def __init__(self, img_size, patch_size, in_channels, embed_dim):
        super(PatchEmbedding, self).__init__()
        self.img_size = img_size
        self.patch_size = patch_size
        self.num_patches = (img_size // patch_size) ** 2
        self.proj = nn.Conv2d(in_channels, embed_dim, kernel_size=patch_size, stride=patch_size)  # 图像分块卷积
        self.cls_token = nn.Parameter(torch.randn(1, 1, embed_dim))  # 分类 token
        self.pos_embed = nn.Parameter(torch.randn(1, self.num_patches + 1, embed_dim))  # 位置嵌入

    def forward(self, x):
        B = x.size(0)  # 获取批次大小
        x = self.proj(x).flatten(2).transpose(1, 2)  # 将图像分割成 patch 并展平
        cls_tokens = self.cls_token.expand(B, -1, -1)  # 扩展分类 token
        x = torch.cat((cls_tokens, x), dim=1)  # 将分类 token 拼接到 patch 之前
        x += self.pos_embed  # 添加位置嵌入
        return x
```

2.1.2 Transformer Block：

- `TransformerBlock`由多头自注意力层（`MultiHeadSelfAttention`）和MLP层（多层感知机）组成，通过残差连接和LayerNorm层进行信息传递和归一化。
- `TransformerBlock`首先使用LayerNorm对输入进行归一化处理，然后通过多头自注意力层（`MultiHeadSelfAttention`）捕捉序列中各位置的依赖关系，再经过MLP层（`MLP`）进行非线性映射和特征提取。

```
class TransformerBlock(nn.Module):
 def __init__(self, embed_dim, num_heads, mlp_dim, dropout):
     super(TransformerBlock, self).__init__()
     self.norm1 = nn.LayerNorm(embed_dim)  # LayerNorm 层
     self.attn = MultiHeadSelfAttention(embed_dim, num_heads)  # 多头自注意力层
     self.norm2 = nn.LayerNorm(embed_dim)  # LayerNorm 层
     self.mlp = MLP(embed_dim, mlp_dim, dropout)  # MLP 层
 def forward(self, x):
     x = x + self.attn(self.norm1(x))  # 残差连接和 LayerNorm
     x = x + self.mlp(self.norm2(x))  # 残差连接和 MLP
     return x
```

2.1.3 Vision Transformer模型：

- VisionTransformer模型集成了PatchEmbedding和多个TransformerBlock，并包括最终的分类全连接层。
- VisionTransformer首先使用PatchEmbedding将输入图像进行分块和嵌入，然后通过多个TransformerBlock进行特征提取和编码，最后通过全连接层（fc）进行图像分类。

```python
# Vision Transformer
class VisionTransformer(nn.Module):
    def __init__(self, img_size, patch_size, in_channels, num_classes, embed_dim, depth, num_heads, mlp_dim, dropout):
        super(VisionTransformer, self).__init__()
        self.patch_embed = PatchEmbedding(img_size, patch_size, in_channels, embed_dim)  # Patch Embedding
        self.transformer_blocks = nn.Sequential(*[
            TransformerBlock(embed_dim, num_heads, mlp_dim, dropout) for _ in range(depth)  # 多个 Transformer Block
        ])
        self.norm = nn.LayerNorm(embed_dim)  # LayerNorm 层
        self.fc = nn.Linear(embed_dim, num_classes)  # 分类全连接层

    def forward(self, x):
        x = self.patch_embed(x)  # 图像分块并嵌入
        x = self.transformer_blocks(x)  # Transformer 编码
        x = self.norm(x)[:, 0]  # 取出分类 token
        x = self.fc(x)  # 最后的分类层
        return x
```

2.2 损失函数和优化器设计

- 使用交叉熵损失作为分类任务的损失函数。
- 选择Adam优化器，并使用学习率调度器根据验证损失动态调整学习率。
- 学习率调度器（`ReduceLROnPlateau`）根据验证损失的变化自动调整学习率，有助于加速模型的收敛过程。

核心代码如下：

```
criterion = nn.CrossEntropyLoss()  # 交叉熵损失
optimizer = optim.Adam(model.parameters(), lr=learning_rate)  # Adam 优化器
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min', patience=3, factor=0.5)  # 学习率调整器
```

2.3 图像尺寸调整

标准的 ViT 模型对图像的输入尺寸有要求，必须为224*224， 图像输入之后首先是需要进行 Patch 分块，一般设置 Patch 的尺寸为16*16，那么一共能生成(224/16)*(224/16)=196 个 Patch 块。

核心代码如下：

```
# 训练数据的变换：包括随机裁剪、水平翻转、归一化
transform_train = transforms.Compose([
    transforms.Resize(224),  # 调整图像大小到 224x224
    transforms.RandomCrop(224, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])
```

3.1 实验结果

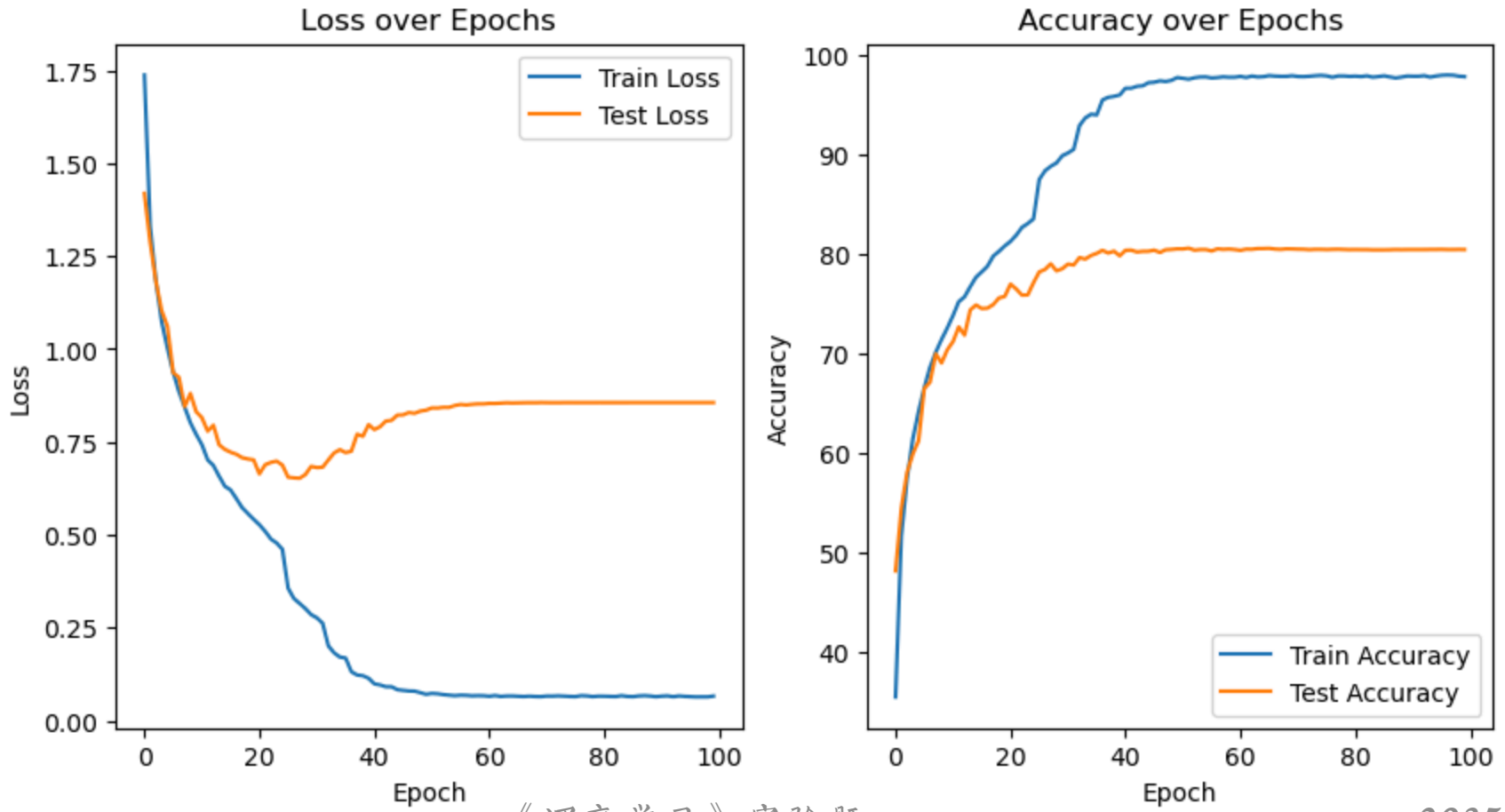
在经过100个epoch的训练后，模型在训练集上的损失逐步降低，同时训练精度提升。最终模型在测试集上达到了80%的准确率，表明ViT模型在CIFAR-10上取得了良好的分类性能。

结果如下（完整结果见实验报告和程序文件）：

```
Epoch [90/100] Train Loss:0.0646, Train Acc: 97.86%, Test Loss: 0.8561, Test Acc: 80.42%
Epoch [91/100] Train Loss: 0.0660, Train Acc: 97.85%, Test Loss: 0.8561, Test Acc: 80.42%
Epoch [92/100] Train Loss: 0.0668, Train Acc: 97.84%, Test Loss: 0.8561, Test Acc: 80.42%
Epoch [93/100] Train Loss: 0.0646, Train Acc: 97.90%, Test Loss: 0.8561, Test Acc: 80.42%
Epoch [94/100] Train Loss: 0.0666, Train Acc: 97.76%, Test Loss: 0.8561, Test Acc: 80.43%
Epoch [95/100] Train Loss: 0.0655, Train Acc: 97.86%, Test Loss: 0.8561, Test Acc: 80.43%
Epoch [96/100] Train Loss: 0.0647, Train Acc: 97.94%, Test Loss: 0.8561, Test Acc: 80.44%
Epoch [97/100] Train Loss: 0.0641, Train Acc: 97.97%, Test Loss: 0.8561, Test Acc: 80.42%
Epoch [98/100] Train Loss: 0.0643, Train Acc: 97.95%, Test Loss: 0.8561, Test Acc: 80.42%
Epoch [99/100] Train Loss: 0.0642, Train Acc: 97.85%, Test Loss: 0.8561, Test Acc: 80.43%
Epoch [100/100] Train Loss: 0.0663, Train Acc: 97.81%, Test Loss: 0.8561, Test Acc: 80.42%
```

3.2 损失和准确率曲线

训练过程中损失和准确率曲线显示出模型在训练和测试集上的表现趋势，验证了模型的稳定性和收敛性。变化曲线如下：



4 总结

本实验利用Vision Transformer模型成功解决了CIFAR-10图像分类任务，通过详细分析和实验结果展示了ViT在小尺寸图像数据集上的有效性和性能优势。

未来的研究可以进一步探索ViT在更大规模和复杂图像数据集上的应用，并尝试优化模型结构和训练策略，以提升其在视觉任务中的表现。

恳请指正