

**深度学习课程**

**实验二：基于ViT的CIFAR10 图像分类**

**学生姓名： 李增圣**

**学生学号： 2024E8009282002**

**指导教师： 徐俊刚 教授 中国科学院大学计算机科学与技术学院**

**培养单位： 中国科学院大学材料科学与光电技术学院**

**2025年 4 月**

**摘 要**

本实验基于Vision Transformer（ViT）模型，采用PyTorch框架在CIFAR-10数据集上实现图像分类。首先将32×32彩色图像按4×4小块切分，并通过LayerNorm与线性映射生成patch嵌入；随后在patch序列前添加可学习的分类token并加入位置编码，通过多头自注意力与前馈网络构建深度Transformer编码器以提取图像特征；最后经全连接分类头输出10类预测。训练阶段使用Adam优化器（学习率3e-4）、交叉熵损失及批次大小256，迭代300轮后，训练集损失从≈1.7稳定下降至<0.05，测试集准确率由初始≈50%提升至≈87%。实验结果验证了ViT在小规模图像分类任务中的有效性与良好泛化性能，为Transformer结构在计算机视觉领域的应用提供了实践依据。

**关键词：**CIFAR-10，Vision Transformer，图像分类

**目 录**

[第1章 实验背景 3](#_Toc395777550)

[第2章 实验过程 4](#_Toc1076428091)

[2.1 导入相关库 4](#_Toc1124683109)

[2.2 数据预处理 4](#_Toc397952069)

[2.3 Transformer 编码器模块 5](#_Toc1116346925)

[2.3.1 多头自注意力模块 5](#_Toc2025628283)

[2.3.2 前馈网络 5](#_Toc676296490)

[2.3.3 Transformer 编码器 6](#_Toc2031647506)

[2.3.4 Vision Transformer 6](#_Toc909646042)

[2.4 训练函数 7](#_Toc484944901)

[2.5 测试函数 8](#_Toc768510742)

[第3章 实验结果 10](#_Toc1393387736)

[3.1 训练参数 10](#_Toc358508417)

[3.2 训练结果 10](#_Toc1759334684)

[3.2.1 训练损失曲线 10](#_Toc435698445)

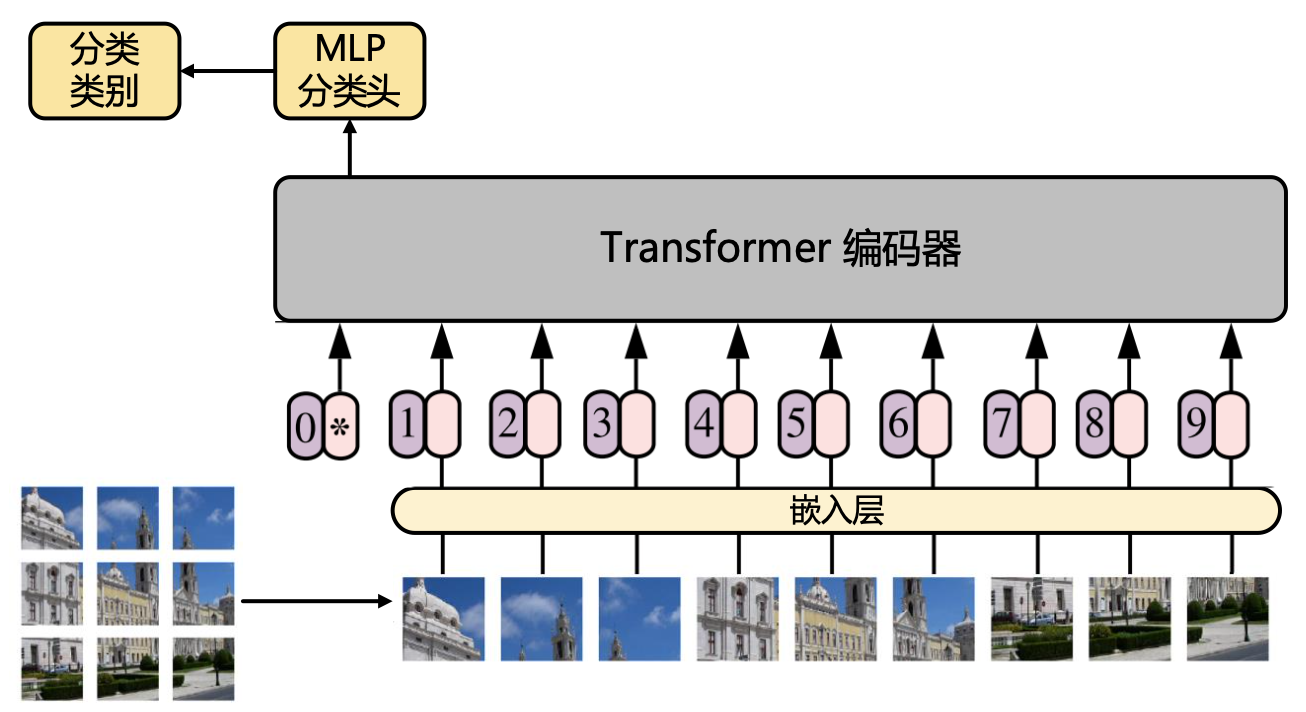
[3.2.2 训练损失曲线 11](#_Toc2012012492)

[3.2.3 收敛与过拟合 12](#_Toc1616447382)

# 实验背景

ViT（vision transformer）是Google在2020年提出的直接将transformer应用在图像分类的模型。ViT直接把图像分成固定大小的patchs，然后通过线性变换得到patch embedding，再将图像的patch embeddings送入transformer后就能够进行特征提取从而分类了。

本实验使用 PyTorch 框架完成了一个基于 ViT 的CIFAR10图像分类任务，ViT架构由三部分组成：（1）图像特征嵌入模块；（2）Transformer 编码器模块； （3）MLP分类模块。

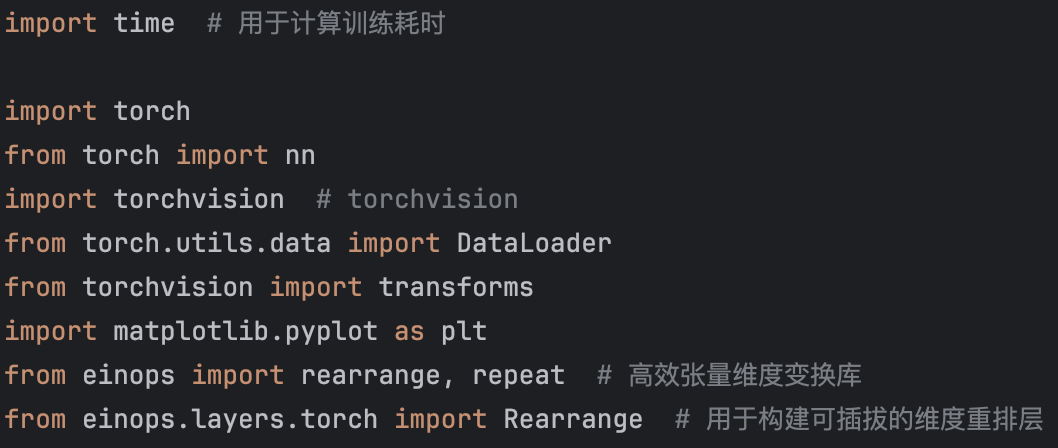


CAFIR10数据集共有60000张彩色图像，这些图像是32\*32，分为10个类，每类6000张图。这里面有50000张用于训练，构成了5个训练批，每一批10000张图；另外10000用于测试，单独构成一批。

# 实验过程

## 导入相关库

torch, nn, DataLoader 等：构建与训练 PyTorch 模型的核心工具。torchvision.datasets, transforms：加载 CIFAR-10 数据集并做图像预处理。einops：方便地做张量维度重排，用于 patch 提取与多头注意力中分头/拼接。matplotlib.pyplot：可视化训练过程中损失与准确率曲线。time：统计训练每轮用时。



## 数据预处理

trans\_train（训练集）：随机裁剪 + 随机翻转 + 张量化 + 归一化。trans\_valid（验证/测试集）：仅张量化 + 归一化。用 DataLoader 封装 CIFAR-10 测试集，批量大小 256，不打乱顺序。



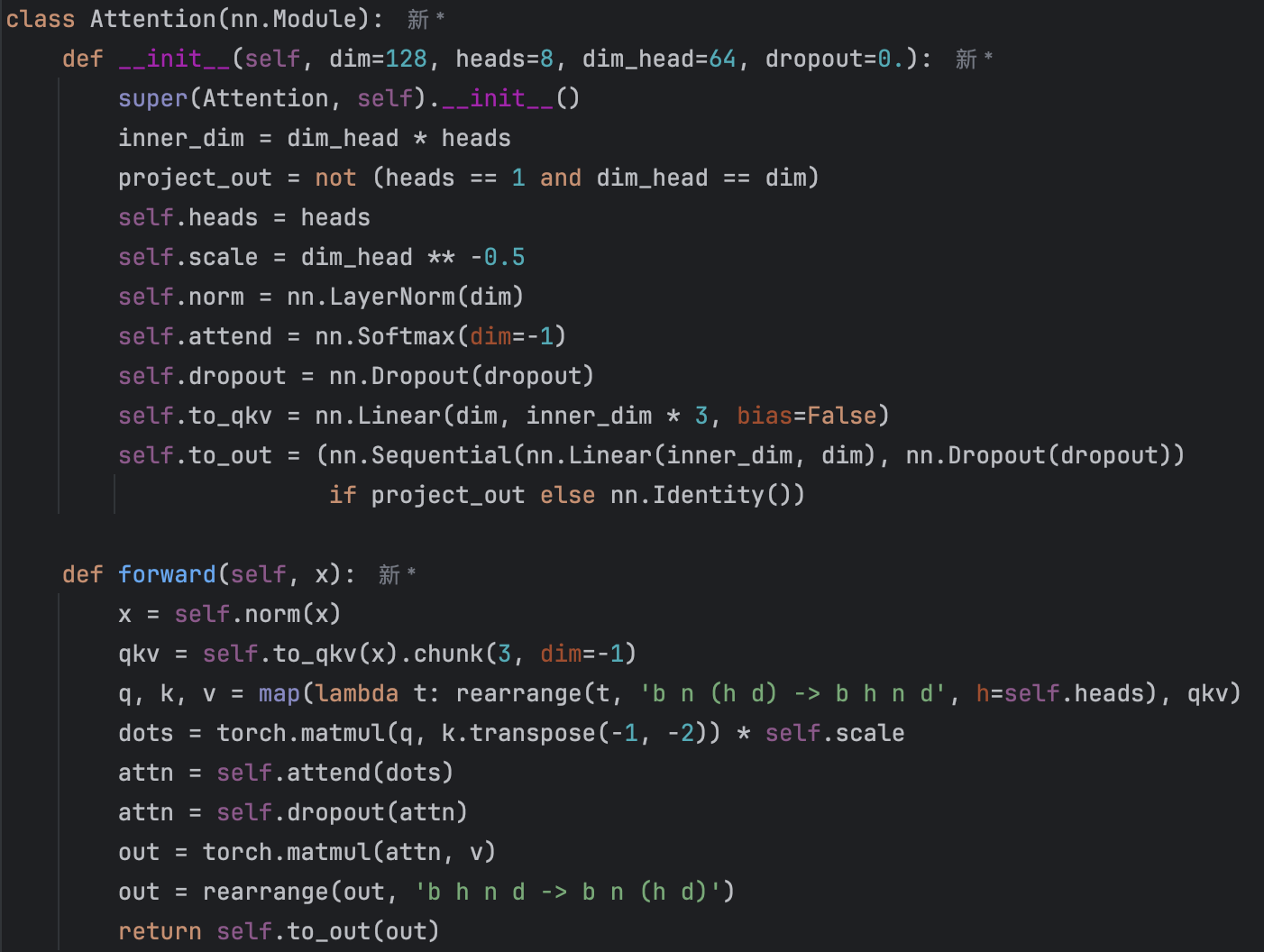
## Transformer 编码器模块

Transformer 编码器模块主要由 LayerNorm 层、多头注意力机制、MLP 模块、残差连接这5个部分组成。

### 多头自注意力模块

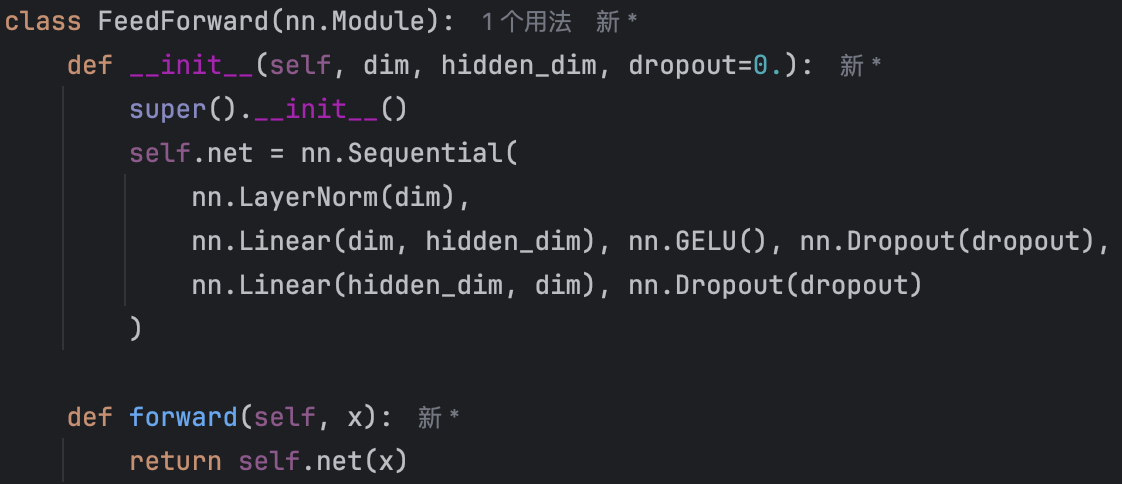
**整体流程：**LayerNorm → 一次线性映射（QKV） → 分头 → 缩放点积 → Softmax → Dropout → 加权求和 → 拼接 → 可选投影

首先对输入张量做 LayerNorm 归一化，然后通过一个线性层一次性生成查询（Q）、键（K）、值（V），并用 einops 将它们拆分成多个头；接着对每个头分别计算缩放点积：先做 QKT再经 Softmax 和 Dropout 得到注意力权重，最后用这些权重加权求和值 V；将各头输出拼回原始维度后（若需要，还通过线性层投影回输入维度并加 Dropout），即可得到最终的注意力输出。



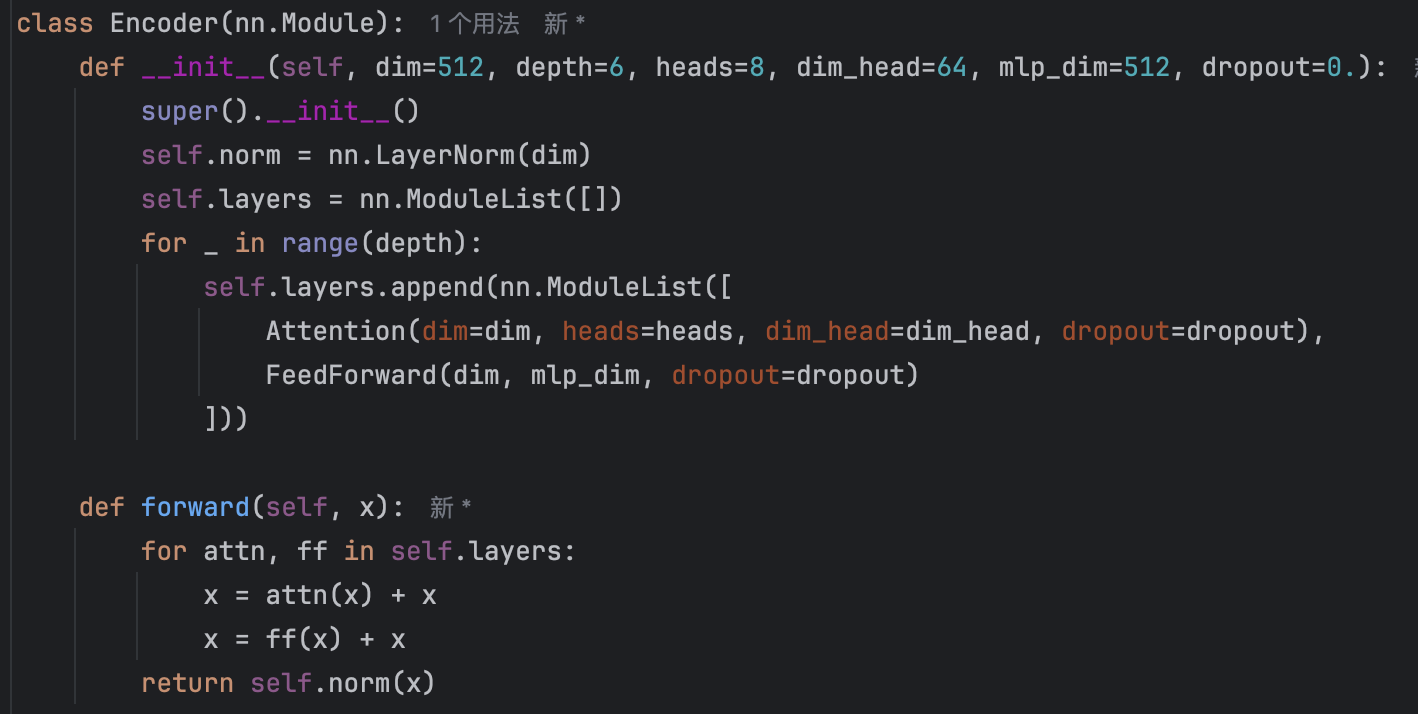
### 前馈网络

FeedForward 模块实现了 Transformer 中常用的两层全连接前馈网络：首先对输入特征做 LayerNorm 归一化，然后通过一个线性层将维度从 dim 扩展到 hidden\_dim，接着使用 GELU 激活并进行 Dropout，以增强非线性和防止过拟合；随后再通过第二个线性层将维度映射回原始的 dim，并再次应用 Dropout。



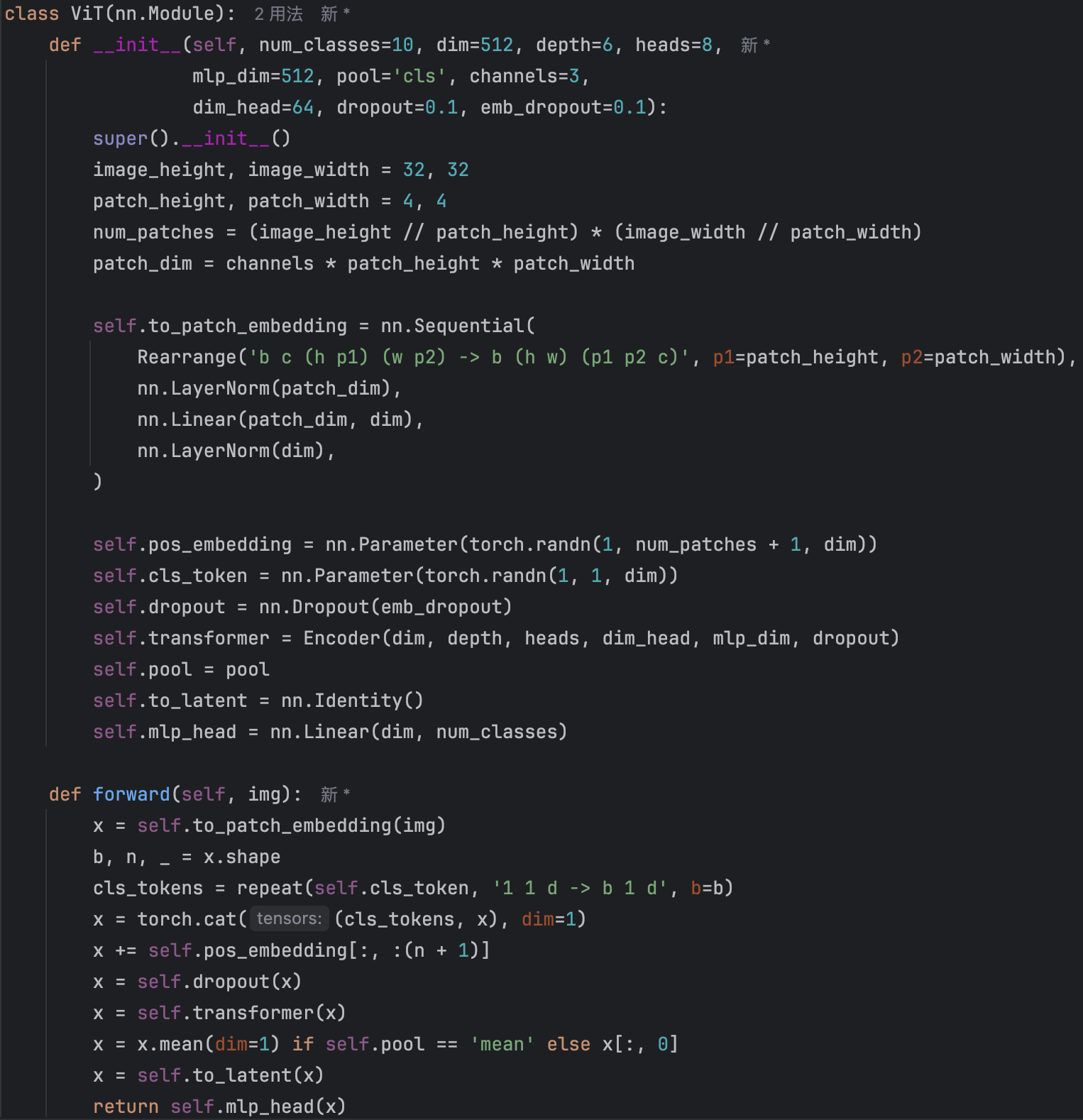
### Transformer 编码器

Encoder 模块将多个自注意力层和前馈网络层按顺序堆叠起来，构成 Transformer 编码器的主体结构。初始化时，它会创建 depth 个子模块列表，每个子模块包含一个多头自注意力 (Attention) 和一个前馈网络 (FeedForward)。在前向传播中，对于输入特征 x，依次执行：先通过自注意力计算并与原始 x 做残差相加，再将结果送入前馈网络并同样做残差相加，循环 depth 次后，最后对全序列做一次 LayerNorm 归一化输出，从而实现多层次的特征融合和非线性变换。



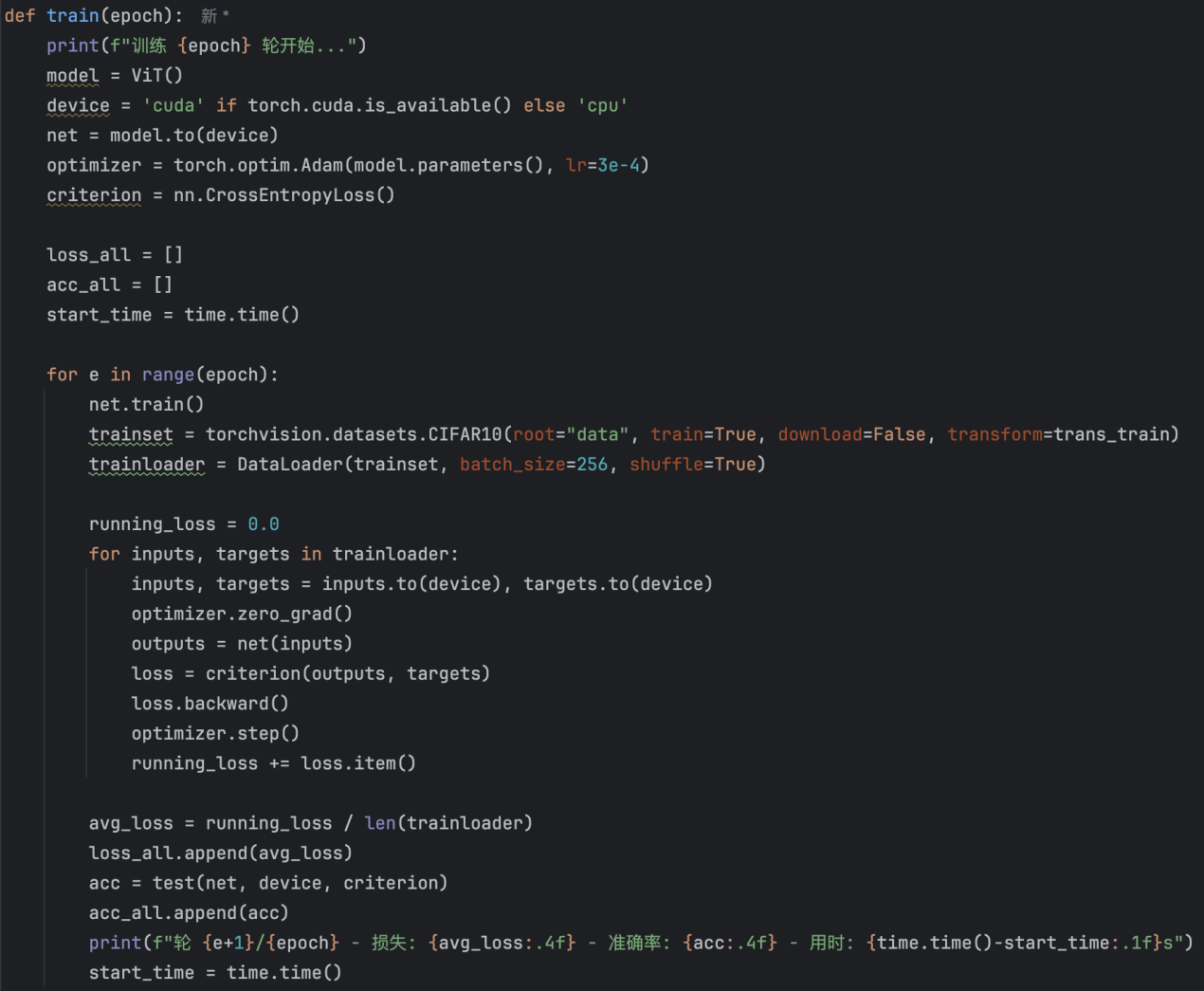
### Vision Transformer

定义了一个基于 Vision Transformer（ViT）结构的图像分类网络：首先用 to\_patch\_embedding 模块将输入的 32×32 RGB 图像按 4×4 小块切分（共 64 个 patch），并通过 LayerNorm 和线性映射将每个 patch 映射到维度 dim 的特征向量；然后在这些 patch 特征前插入一个可学习的分类 token（cls\_token），并加上可学习的位置编码（pos\_embedding），再经过一次 Dropout；接着将序列送入由多层自注意力和 MLP 子层组成的 Transformer 编码器（Encoder）；编码器输出后，根据 pool 参数选择对所有 token 求均值得到全局表示或直接取分类 token（第 0 位）；最后通过一个线性层（mlp\_head）将该表示映射到类别数 num\_classes ，实现图像分类的预测。



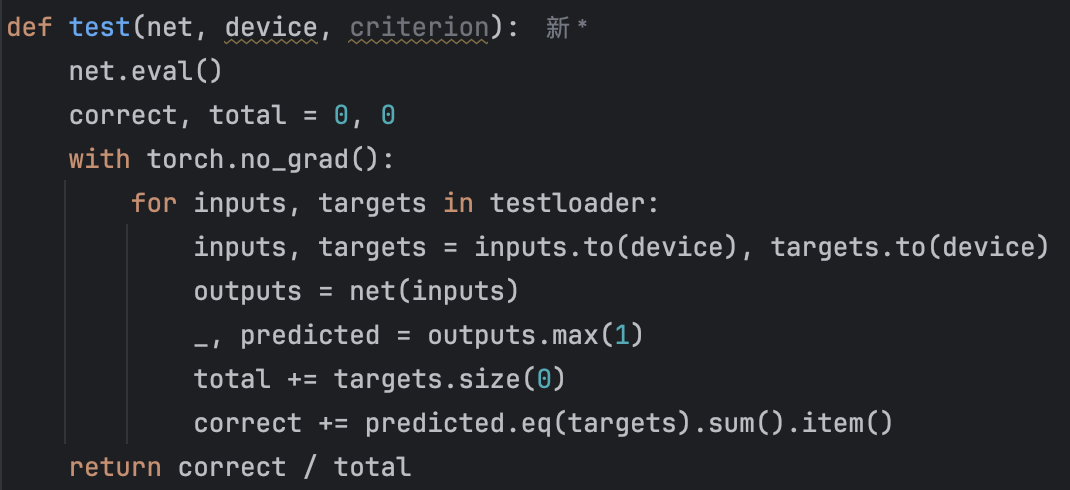
## 训练函数

首先实例化一个 Vision Transformer 模型并将其移动到 GPU（若可用）或 CPU，然后用 Adam 优化器（学习率 3e-4）和交叉熵损失函数进行配置。函数内部维护两个列表 loss\_all 和 acc\_all 来记录每一轮的训练平均损失和测试准确率，并使用 time.time() 统计每轮耗时。对于指定的轮数，它先将模型设为训练模式，从 CIFAR-10 训练集构造 DataLoader（批量大小 256，随机打乱），对每个小批量执行前向传播、计算损失、反向传播和参数更新，并累计损失值。每轮结束后，计算平均损失，调用 test 函数在验证集上评估准确率，将结果追加到记录列表中，并打印本轮的损失、准确率和耗时。全部训练完成后，会将 loss\_all 和 acc\_all 分别写入文本文件，并将模型参数保存到 model1.pt。



## 测试函数

首先调用 net.eval() 将模型切换到评估模式（关闭 dropout、BatchNorm 等训练特性），然后在 torch.no\_grad() 环境中遍历全局的 testloader，将每个批次的输入和标签搬到指定设备上，执行前向推理得到输出 logits，再用 outputs.max(1) 取概率最大的类别作为预测值；同时累计所有样本的总数 total 和预测正确的样本数 correct。最后，函数返回 correct/total 作为整体测试准确率。



# 实验结果

## 训练参数

**Vision Transformer模型结构：**  
 – 输入图像：32×32 RGB  
 – Patch 大小：4×4（共 64个 patch）  
 – 嵌入维度 dim=512  
 – Transformer 层数 depth=6  
 – 注意力头数 heads=8，每头维度 dim\_head=64  
 – MLP 隐藏维度 mlp\_dim=512  
 – Dropout (嵌入和各子层) 均为 0.1

**优化器：**Adam，学习率 lr=3e-4

**损失函数：**交叉熵损失 nn.CrossEntropyLoss()

**批大小：**batch\_size=256

**训练轮数：**epoch=300

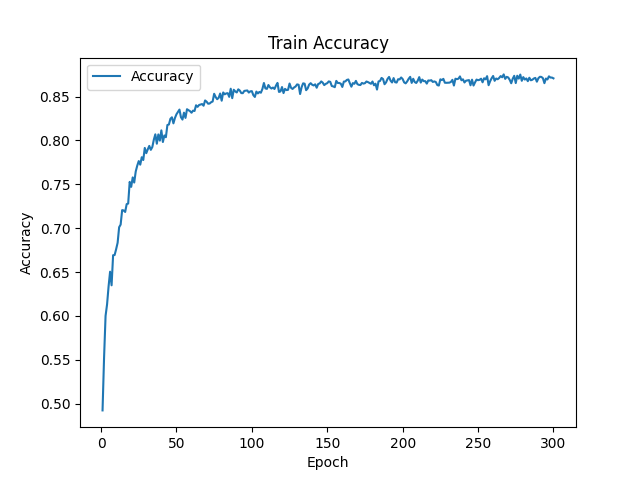
**硬件环境：**GPU（CUDA）

## 训练结果

### 训练损失曲线

初始损失约 1.7，快速下降：前 50 轮内已跌至 ≈0.5；100 轮左右下降至 ≈0.15；200–300 轮收敛到 ≈0.05 左右。

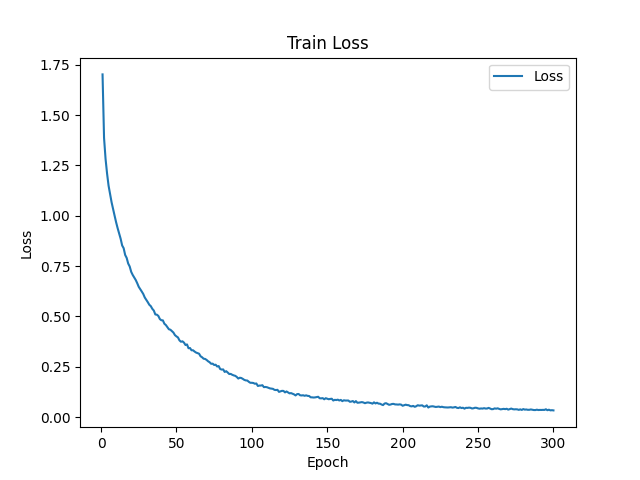
结论：模型能稳定地将训练集损失压低，并在中早期获得大部分收敛，后续下降趋于平缓。



### 训练准确率曲线

起始准确率仅 ≈50%；前 30–50 轮内迅速提升到 ≈85%；100 轮后精度稳定在 86%–88% 之间，300 轮末约 87%。

结论：模型泛化性能在前期快速提升，中期后续增益有限，整体测试准确率 plateau 于 ~0.87。



### 收敛与过拟合

训练损失持续下降，而测试精度在 ~100 轮后波动不大，未出现明显精度下滑，说明暂未严重过拟合；

但测试精度与训练损失的差距（训练损失接近 0，但精度仅0.87）提示仍有泛化空间。