# CIFAR-10图像分类实验报告

## 摘要

部分A基于Pytorch框架从零实现了深度学习模型，在CIFAR-10数据集上进行图像分类任务。CIFAR-10是计算机视觉领域的经典数据集，包含10个类别的60,000张32×32彩色图像。本实验旨在探索不同的深度学习技术对模型性能的影响。

通过一系列优化策略，包括网络结构改进、批归一化、数据增强、学习率调整等方法，最终将分类准确率从基线模型的86.97%提升至93.21%。同时，用最佳模型验证了模型在MNIST数据集上的泛化性能。

## 2. 实验方法

### 2.1 基线模型

基线模型采用简单的CNN架构，包括5层卷积层、4个全连接层，采用Adam和weight decay优化器，训练了128个epoch。在数据增强方面，基线模型在训练集上使用了RandamRotation，RandomHorizontalFlip，RandomAffine，在测试集上采用Normalize。在网络架构上，首先CIFAR-10是有3个通道的图像数据集，因此输入通道为3。经过卷积层，通道数逐步增加（3→128→256→512→512→256），每层后接ReLU激活函数。同时采用池化层降维，在第1、2、5卷积层后各有一个2×2的最大池化层来下采样。4层全连接层的神经元数量依次为512、256、128，最后输出10个类别。

\begin{verbatim}

net = Sequential(

Conv2d(3, 128, 3, padding=1), ReLU(), MaxPool2d(2), Dropout(0.3),

Conv2d(128, 256, 3, padding=1), ReLU(), MaxPool2d(2), Dropout(0.3),

Conv2d(256, 512, 3, padding=1), ReLU(),

Conv2d(512, 512, 3, padding=1), ReLU(),

Conv2d(512, 256, 3, padding=1), ReLU(), MaxPool2d(2), Dropout(0.3),

Flatten(),

Linear(256 \* 4 \* 4, 512), ReLU(), Dropout(0.5),

Linear(512, 256), ReLU(), Dropout(0.5),

Linear(256, 128), ReLU(), Dropout(0.5),

Linear(128, 10)

)

\end{description}

### 2.2 改进策略

我们依次尝试了5种变体，记录了模型在训练集和测试集上的表现，测试精确度逐步提升(Table )。其中由于运行时间长且表现提升不大，V3的残差连接的尝试被舍弃，V4仍然基于V2的网络架构进行改进。

#### 2.2.1 BatchNorm & Dropout Ratio（Variant 1）

首先，在基线模型的基础上，本文考虑使用归一化减少内部协变量偏移，添加BatchNorm2d和BatchNorm1d层，使每层的输入分布更加稳定，从而加快模型的收敛速度。第二，为了保持学习信息，移除了卷积层的Dropout并降低全连接层Dropout率至0.3。结果：准确率提升至89.31%。

从学习曲线来看，training loss在前20个epoch快速下降，在40个epoch之后缓慢下降，最终loss为0.033.testing loss在20个epoch之后出现上升，表明模型出现过拟合，最终test loss为0472，精确度却没有显著变化。loss和accuracy的差异表现，可能的原因是Entropy更聚焦模型预测的概率分布，而accuracy只要求最高概率的类别是否正确。因此，虽然精确度变高了，模型仍然是过拟合的。

\begin{verbatim}

net = nn.Sequential(

    nn.Conv2d(3, 128, 3, padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2),nn.BatchNorm2d(128),

    nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2),nn.BatchNorm2d(256),

    nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(inplace=True),nn.BatchNorm2d(512),

    nn.Conv2d(512, 512, 3, padding=1), nn.ReLU(inplace=True),nn.BatchNorm2d(512),

    nn.Conv2d(512, 256, 3, padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2),nn.BatchNorm2d(256),

    nn.Flatten(),

    nn.Linear(256 \* 4 \* 4, 512), nn.ReLU(inplace=True), nn.Dropout(0.3),nn.BatchNorm1d(512),

    nn.Linear(512, 256), nn.ReLU(inplace=True), nn.Dropout(0.3),nn.BatchNorm1d(256),

    nn.Linear(256, 128), nn.ReLU(inplace=True), nn.Dropout(0.3),nn.BatchNorm1d(128),

    nn.Linear(128, 10),

)

\end{description}

#### 2.2.2 Go Deeper and Wider（Variant 2）

Variant 2尝试增加网络层数、扩大通道数来提高模型对复杂pattern的学习能力。本文在Variant1的基础上，将卷积层数量增加到7层，并扩大通道数(3→256→512→1024→1024→2048→2048)，在第1,2,7层后面增加MaxPool层，每一层之间都有归一化层。模型参数量由7.29M变成127.06M。结果：准确率进一步提升至89.60%，而学习曲线也呈现出与Variant1相似的形状。

\begin{verbatim}

net = nn.Sequential(

nn.Conv2d(3, 256, 3, padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2), nn.BatchNorm2d(256),

nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2), nn.BatchNorm2d(512),

nn.Conv2d(512, 1024, 3, padding=1), nn.ReLU(inplace=True), nn.BatchNorm2d(1024),

nn.Conv2d(1024, 1024, 3, padding=1), nn.ReLU(inplace=True), nn.BatchNorm2d(1024),

nn.Conv2d(1024, 2048, 3, padding=1), nn.ReLU(inplace=True), nn.BatchNorm2d(2048),

nn.Conv2d(2048, 2048, 3, padding=1), nn.ReLU(inplace=True), nn.BatchNorm2d(2048),

nn.Conv2d(2048, 1024, 3, padding=1), nn.ReLU(inplace=True), nn.MaxPool2d(2), nn.BatchNorm2d(1024),

nn.Flatten(),

nn.Linear(1024 \* 4 \* 4, 2048), nn.ReLU(inplace=True), nn.Dropout(0.3), nn.BatchNorm1d(2048),

nn.Linear(2048, 1024), nn.ReLU(inplace=True), nn.Dropout(0.3), nn.BatchNorm1d(1024),

nn.Linear(1024, 512), nn.ReLU(inplace=True), nn.Dropout(0.3), nn.BatchNorm1d(512),

nn.Linear(512, 10),)

\end{description}

#### 2.2.3 Residual Connections（Variant 3）

To avoid the problems of vanishing or exploding gradients，考虑加入residual connections。我们首先直接实现了ResNet34架构，发现其training loss为0.129，测试精确度为84.33%，比基线模型还要差，运行时间为1h13分钟。于是，我们仍考虑在Variant2的基础上加入residual connections。

第一个卷积层与Variant2一致，256个3×3卷积核，padding=1，后接ReLU激活函数和BatchNorm2d。然后，使用7个残差块，通道数依次为256→512→512→1024→1024→2048→2048→1024。并在第1,3,7个残差块后面连接MaxPool2d。全连接层与Variant2保持一致。最终，在测试集上表现出92.04%的准确度。从学习曲线上看，会出现波动，尤其在test loss上曲线非常不稳定，因此模型表现会对epoch number较为敏感。由于训练时间达到6h27m，参数量134.87M，本文并没有将其作为接下来的网络架构，而是延续了Variant2的网络结构，也能取得更好的效果。

\begin{verbatim}

net = nn.Sequential(

  nn.Conv2d(3, 256, 3, padding=1), nn.ReLU(inplace=True), nn.BatchNorm2d(256),

    ResBlock(256, 256),  # 添加残差连接

    nn.MaxPool2d(2),

    ResBlock(256, 512),

    ResBlock(512, 512),

    nn.MaxPool2d(2),

    ResBlock(512, 1024),

    ResBlock(1024, 1024),

    ResBlock(1024, 2048),

    ResBlock(2048, 2048),

    ResBlock(2048, 1024),

    nn.MaxPool2d(2),

nn.Flatten(),

nn.Linear(1024 \* 4 \* 4, 2048), nn.ReLU(inplace=True), nn.Dropout(0.3), nn.BatchNorm1d(2048),

    nn.Linear(2048, 1024), nn.ReLU(inplace=True), nn.Dropout(0.3), nn.BatchNorm1d(1024),

 nn.Linear(1024, 512), nn.ReLU(inplace=True), nn.Dropout(0.3), nn.BatchNorm1d(512),

    nn.Linear(512, 10),

)

\end{verbatim}

#### 2.2.4 数据增强（Variant 4）

在Variant2更宽更深的CNN网络的基础上，对图像使用更多的数据增强，来提高模型的泛化能力。具体而言，本文对训练集新增了RandomCrop、ColorJitter和RandomErasing等数据增强方法，而测试集仍然只进行归一化处理。结果，在测试集上也能达到92.04%的准确度，这与ResNet一致。而从学习曲线上看，training loss并没有快速下降，在50个epoch之后才趋于收敛，而test loss却在20个epoch时就趋于稳定，accuracy曲线变化也比较smooth。因此，对比Variant4和Variant3，数据增强更能使模型获得好的泛化表现和稳定性。

\start{verbatim}

for data\_type in ("train", "test"):

is\_train = data\_type=="train"

if is\_train:

transformation[data\_type] = tv\_transforms.Compose([

tv\_transforms.RandomCrop(32, padding=4, padding\_mode='reflect'),

tv\_transforms.RandomHorizontalFlip(),

tv\_transforms.RandomRotation(degrees=15),

tv\_transforms.RandomAffine(degrees=0, translate=(0.1, 0.1), scale=(0.9, 1.1)),

tv\_transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),

tv\_transforms.ToTensor(),

tv\_transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),

tv\_transforms.RandomErasing(p=0.5, scale=(0.02, 0.33), ratio=(0.3, 3.3)),

])

else:

transformation[data\_type] = tv\_transforms.Compose([

tv\_transforms.ToTensor(),

tv\_transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),

])

\end{verbatim}

#### 2.2.5 学习率调度（Variant 5）

最后，本文尝试了对优化器和学习率的调整。首先，在baseline model上尝试了AdamW和更大的weight\_decay(从1e-6提高至0.01)，同时用StepLR，精度下降到了83.82%。然后，把StepLR改成constant+cooldown的模式，精度下降到了68.85%。于是，我们仍然采用Adam优化器，保持原来的初始学习率(3e-4)和weight decay(1e-6)。在此基础上，加入了Warmup + Cosine学习率调整，具体为前10%从初始学习率(3e-4)直线上升到最大学习率(3e-3)，在之后的40%个epoch阶段保持最大学习率，在最后的50%采用CosineAnnealingLR scheduler,学习率变化曲线见图X。最终能够达到test accuracy of 93.04%。

\start{verbatim}

def get\_lr(epoch):

if epoch < warmup\_epochs:

return initial\_lr + (max\_lr - initial\_lr) \* (epoch / warmup\_epochs)

return max\_lr

scheduler = torch.optim.lr\_scheduler.CosineAnnealingLR(optimizer, T\_max=num\_epochs\*0.5, eta\_min=1e-6)

for epoch in range(num\_epochs):

if epoch <= num\_epochs\*0.5:

current\_lr = get\_lr(epoch)

else:

scheduler.step()

current\_lr = optimizer.param\_groups[0]['lr']

\end{verbatim}

#### 2.3 MNIST

以Variant5作为最终的模型，在MNIST数据集上retrain并测试精度。由于MNIST数据集是单通道灰度图像，像素为28×28。模型的一些设置需要发生变化：

数据增强中的RandomCrop尺寸从32改为28，对normalization使用经验值mean=0.1307, std=0.3081。由于MNIST的数字翻转后可能改变语义（如6翻转后可能变成9），移除了RandomHorizontalFlip变换。

在网络结构中，MNIST的输入通道数为1，经过3次MaxPool2d(2)：28 → 14 → 7 → 3，所以最后特征图大小是3×3。另外，由于MNIST数据集相对简单，只进行了32个epoch的训练。最终，测试集精确度达到了99.71%，但学习曲线并不平滑，虽然loss很小，但没有达到水平程度，后续还可以再增大训练。

模型信息

CIFAR-10结果对比

| Model | Test acc | Time | Train loss | Test loss | Param. |

|---------|-----------|---------|---------|--------|

| Baseline | 86.50% | xx min | xx | xx| 7.28M |

| Variant 1 | 89.31% | 50 min | 0.033 | xx| xx M |

| Variant 2 | 89.60% | 70 min | 0.042 | xx| xx M |

| Variant 3 | 84.33% | 73 min | 0.129 | xx| xx M |

| Variant 4 | xx% | xx min | xx | xx| xx M |

| Variant 5 | xx% | xx min | xx | xx| xx M |

## 4. 分析与讨论

### 4.1 改进效果分析

1. 批归一化的影响：

- 加速训练收敛

- 提高模型稳定性

- 允许使用更大学习率

2. 网络深度与宽度：

- 更深的网络提供更强的特征提取能力

- 更宽的通道数增加特征多样性

- 需要平衡模型容量和计算成本

3. 残差连接的优劣：

- 性能不如预期

- 可能原因：实现细节、优化器配置、学习率设置

4. 数据增强的重要性：

- 提高模型泛化能力

- 减少过拟合

- 增加训练样本多样性

### 4.2 经验总结

1. 模型设计：

- 批归一化是提升性能的关键

- 网络深度和宽度需要权衡

- 简单架构可能优于复杂架构

2. 训练策略：

- 学习率调度很重要

- 数据增强是必要的

- 优化器选择影响收敛

3. 实现细节：

- CuPy实现需要注意内存管理

- GPU加速提供显著性能提升

- 批处理大小影响训练稳定性

## 5. 结论

通过系统的实验和优化，我们成功将CIFAR-10分类准确率提升至93.04%。关键改进包括：

1. 添加批归一化层

2. 加深加宽网络结构

3. 使用强数据增强

4. 实现自适应学习率调度

模型在MNIST数据集上的良好表现也证明了其架构的通用性。