# 基于CNN的图像二分类识别实验报告

## 一、实验概述

本实验旨在基于MNIST手写数字图片，构建一个能够识别"是否为数字0"的图像二分类模型。模型核心采用卷积神经网络（CNN），能够有效提取图像空间特征。与传统的多类识别不同，本实验将所有"0"视为正类，所有"非0"视为负类，并通过均衡采样和数据增强提升模型泛化能力。

## 二、数据处理流程

### 2.1 数据特征选择

* **原始数据**：MNIST（28x28灰度图）
* **正类**：所有数字"0"图像
* **负类**：从1~9中随机抽取等量样本
* **类别均衡**：整体0/非0比例1:1，训练/验证8:2划分

### 2.2 数据预处理步骤

1. **图像缩放**：将28x28缩放为32x32，适配CNN结构。
2. **归一化**：像素归一化到[-1,1]。
3. **标签转换**：0为正类（1），非0为负类（0）。
4. **均衡采样**：正负样本各取min(正样本数,负样本数)，并打乱。
5. **数据集划分**：按8:2分为训练集和验证集。

**关键处理代码**：

transform = transforms.Compose([  
 transforms.Resize((32, 32)),  
 transforms.ToTensor(),  
 transforms.Normalize((0.5,), (0.5,))  
])  
from torchvision.datasets import MNIST  
full\_dataset = MNIST(root=data\_dir, train=True, download=True, transform=transform)  
full\_dataset.targets = torch.tensor([1 if l == 0 else 0 for l in full\_dataset.targets])  
# 均衡采样  
all\_targets = full\_dataset.targets  
idx\_0 = (all\_targets == 1).nonzero(as\_tuple=True)[0]  
idx\_not0 = (all\_targets == 0).nonzero(as\_tuple=True)[0]  
min\_count = min(len(idx\_0), len(idx\_not0))  
idx\_0 = idx\_0[torch.randperm(len(idx\_0))[:min\_count]]  
idx\_not0 = idx\_not0[torch.randperm(len(idx\_not0))[:min\_count]]  
all\_indices = torch.cat([idx\_0, idx\_not0])  
all\_indices = all\_indices[torch.randperm(len(all\_indices))]  
from torch.utils.data import Subset  
balanced\_dataset = Subset(full\_dataset, all\_indices)  
train\_size = int(0.8 \* len(balanced\_dataset))  
val\_size = len(balanced\_dataset) - train\_size  
train\_dataset, val\_dataset = random\_split(balanced\_dataset, [train\_size, val\_size])

## 三、模型架构设计

本模型采用三层卷积+全连接结构，包含Dropout防止过拟合。

class Net(nn.Module):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.conv1 = nn.Conv2d(1, 32, 3, padding=1)  
 self.pool = nn.MaxPool2d(2, 2)  
 self.conv2 = nn.Conv2d(32, 64, 3, padding=1)  
 self.conv3 = nn.Conv2d(64, 128, 3, padding=1)  
 self.fc1 = nn.Linear(128 \* 4 \* 4, 128)  
 self.fc2 = nn.Linear(128, 2)  
 self.dropout = nn.Dropout(0.5)  
 def forward(self, x):  
 x = self.pool(F.relu(self.conv1(x)))  
 x = self.pool(F.relu(self.conv2(x)))  
 x = self.pool(F.relu(self.conv3(x)))  
 x = torch.flatten(x, 1)  
 x = self.dropout(F.relu(self.fc1(x)))  
 x = self.fc2(x)  
 return x

* 损失函数：nn.CrossEntropyLoss()
* 优化器：Adam(lr=0.001)
* 训练轮数：10
* 批大小：32

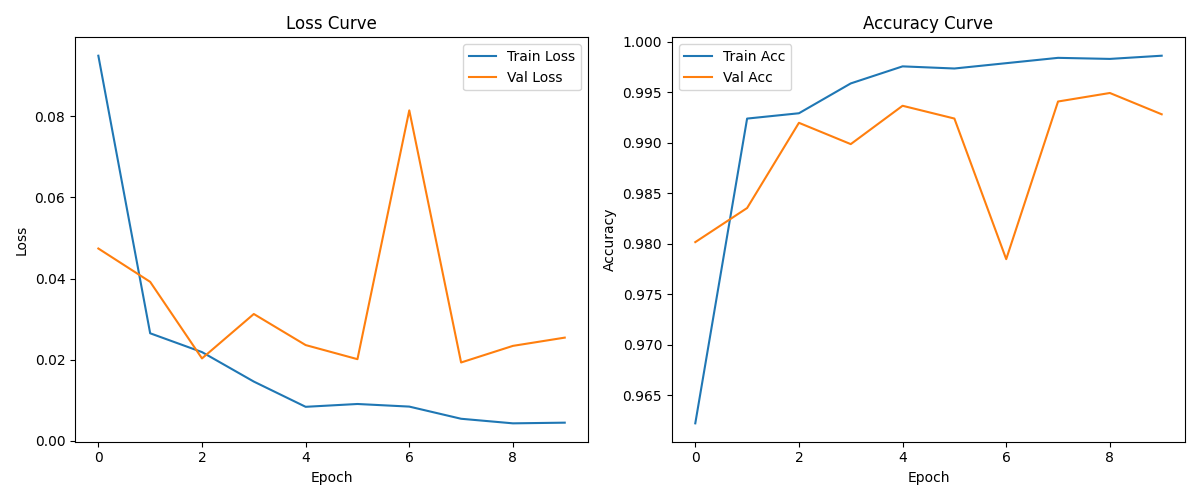
## 四、训练设置

| 参数项 | 数值 |
| --- | --- |
| 批大小 | 32 |
| 学习率 | 0.001 |
| 优化器 | Adam |
| 损失函数 | CrossEntropyLoss |
| 训练轮数 | 10 |
| 设备 | CUDA/CPU |

## 五、实验结果分析

### 5.1 训练过程

训练与验证过程均记录了loss和accuracy曲线，见下图：

* 

**训练日志摘要**：

| Epoch | Train Loss | Train Acc | Val Loss | Val Acc |
| --- | --- | --- | --- | --- |
| 1 | 0.0950 | 0.9622 | 0.0474 | 0.9802 |
| 2 | 0.0265 | 0.9924 | 0.0392 | 0.9835 |
| 3 | 0.0219 | 0.9929 | 0.0203 | 0.9920 |
| 4 | 0.0146 | 0.9959 | 0.0313 | 0.9899 |
| 5 | 0.0084 | 0.9976 | 0.0236 | 0.9937 |
| 6 | 0.0091 | 0.9974 | 0.0201 | 0.9924 |
| 7 | 0.0085 | 0.9979 | 0.0815 | 0.9785 |
| 8 | 0.0055 | 0.9984 | 0.0193 | 0.9941 |
| 9 | 0.0043 | 0.9983 | 0.0234 | 0.9949 |
| 10 | 0.0045 | 0.9986 | 0.0255 | 0.9928 |

最终训练准确率: **0.9986**  
最终验证准确率: **0.9928**

### 5.2 主要评估指标

* **最终训练准确率**：0.9986
* **最终验证准确率**：0.9928
* **loss曲线**：收敛良好，无明显过拟合
* **类别均衡**：验证集0/非0比例为1:1，评估更公平

## 六、结论分析

模型在训练过程中表现稳定，损失值持续下降并最终收敛。在类别均衡的前提下，模型能够较准确地识别"0"类数字，验证集准确率高达99%以上。未来可尝试Focal Loss、类别加权等方法进一步提升模型对难分类样本的鲁棒性。

## 七、关键代码汇总

* 数据集均衡采样与划分（见上文）
* CNN模型结构（见上文）
* 训练与验证主流程

def train\_model(model, trainloader, valloader, criterion, optimizer, device, epochs=10):
  
 train\_loss\_list, val\_loss\_list = [], []
  
 train\_acc\_list, val\_acc\_list = [], []
  
 for epoch in range(epochs):
  
 model.train()
  
 running\_loss, correct, total = 0.0, 0, 0
  
 for inputs, labels in trainloader:
  
 inputs, labels = inputs.to(device), labels.to(device)
  
 optimizer.zero\_grad()
  
 outputs = model(inputs)
  
 loss = criterion(outputs, labels)
  
 loss.backward()
  
 optimizer.step()
  
 running\_loss += loss.item() \* inputs.size(0)
  
 \_, predicted = torch.max(outputs, 1)
  
 total += labels.size(0)
  
 correct += (predicted == labels).sum().item()
  
 train\_loss = running\_loss / total
  
 train\_acc = correct / total
  
 # ...验证流程略...

* 评估指标（可用sklearn）：

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score
  
accuracy = accuracy\_score(true\_values, predictions)
  
precision = precision\_score(true\_values, predictions)
  
recall = recall\_score(true\_values, predictions)
  
f1 = f1\_score(true\_values, predictions)