

**计算机视觉实验二实验报告**

|  |  |
| --- | --- |
| 姓 名： | 王嘉成 |
| 学 院： | 计算机学院 |
| 专 业： | 计算机科学与技术专业 |
| 班 级： | 计算机2207班 |
| 学 号： | U202215576 |
| 指导教师： | 刘康 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2024年 12 月 24 日

目 录

[一、实验要求 1](#_Toc97642866)

[二、实验内容 2](#_Toc97642870)

[1、数据集的下载 2](#_Toc97642871)

[2、数据的加载与预处理 2](#_Toc97642872)

[3、模型的构建 4](#_Toc97642873)

[4、训练与测试 5](#_Toc97642873)

[三、实验框架 6](#_Toc97642874)

[1、实验环境 6](#_Toc97642875)

[2、卷积神经网络结构 6](#_Toc97642876)

[3、实验结构 7](#_Toc97642877)

[四、实验结果 8](#_Toc97642874)

[1、训练参数设置 8](#_Toc97642875)

[2、结果与分析 8](#_Toc97642876)

# 一、实验要求

任务要求：设计一个卷积神经网络，输入为两张MNIST手写体数字图片，如果两张图片为**同一个数字**（注意，非同一张图片），输出为1，否则为0。

从MNIST数据集的训练集中选取10%作为本实验的训练图片，从MNIST数据集的测试集中选取10%作为本实验的测试图片。请将该部分图片经过适当处理形成一定数量的用于本次实验的训练集和测试集。

注意事项：

1. 深度学习框架任选。
2. 实验报告需包含训练集和测试集构成、神经网络架构、每一轮mini-batch训练后的模型在训练集和测试集上的损失、最终的训练集和测试集准确率，以及对应的实验分析。
3. 将代码和实验报告打包成**ZIP**压缩包，以“姓名-学号-实验报告#”命名，比如“张三-2020XXX-实验报告一.zip”，提交到课程平台（https://smartcourse.hust.edu.cn/）。
4. 截止时间为12月25号下午2:00。

二、实验步骤及内容

## 、数据集的下载

首先，我们需要MNIST数据集，所以通过组织一段代码从互联网上下MNIST数据集的训练集和测试集，同时对每个图像进行预处理，将其转换为PyTorch张量并归一化，使得图像的像素值位于[-1.0, 1.0]范围内。

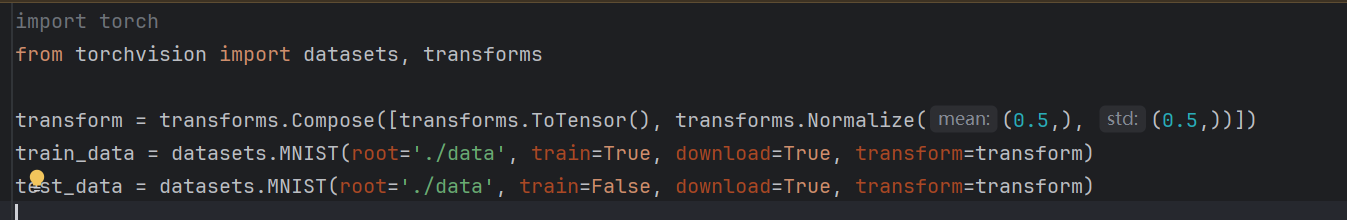


图 1 download代码段

## 、数据的加载与预处理

在正式训练模型前，我们先要对数据集进行加载和预处理。

decompress\_file函数用于解压.gz格式的文件，这些文件包含了MNIST数据集的内容。解压后的数据会以原始格式存储，以便后续加载。这是因为MNIST数据集通常是以压缩格式分发的，解压函数负责从压缩文件中提取数据。

create\_pairs函数接收图像和标签作为输入，生成图像对。对于数据集中的每一张图像，函数随机选择属于同一类别的第二张图像来形成正样本（标签为1），同时选择一张来自不同类别的图像来形成负样本（标签为0）。这些图像对会沿着通道维度堆叠成一个3D数组，每一对图像的形状为(2, 28, 28)，即两张28x28像素的图像堆叠在一起。这是为了完成图像对比任务，模型需要判断这两张图像是否属于同一类别。pair\_labels数组记录了每对图像的标签，正样本为1，负样本为0。最后，这两个数组（图像对和标签）作为numpy数组返回。

load\_images和load\_labels函数负责从原始二进制文件中读取图像和标签数据。这些函数会打开文件，读取文件头信息，并将数据转换为numpy数组。图像数据会被重塑成一个3D数组，形状为(num\_images, rows, cols)，其中num\_images是图像数量，rows和cols是每张图像的尺寸。

load\_data函数首先调用decompress\_data来解压数据，然后使用其中的10%数据用于load\_images和load\_labels函数加载图像和标签文件。接着，它调用create\_pairs函数来生成图像对，这些图像对将用于后续的模型训练。生成的图像对会被标准化，将像素值除以255，使其值范围变为[0, 1]，这是神经网络常见的预处理步骤。

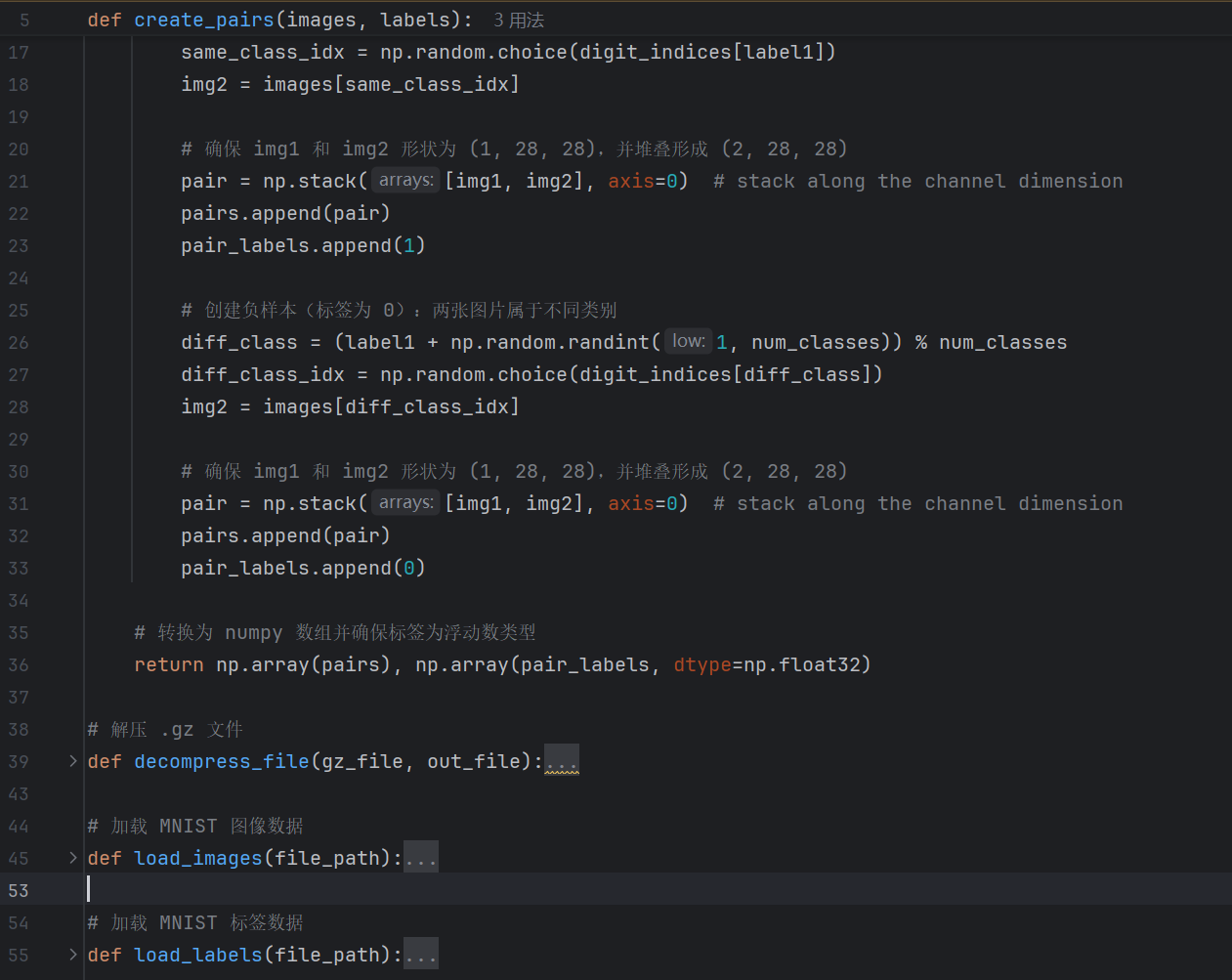


图 2 loader部分代码段



图 3 main部分代码段

## 、模型的构建

根据实验要求，我们要设计一个卷积神经网络来完成实验任务，我们定义了一个简单的卷积神经网络模型，类名为SimpleCNN，继承自PyTorch的nn.Module类。该模型旨在处理成对的图像数据，用于二分类任务，例如，判断两张图像是否属于同一类别。

在\_\_init\_\_方法中，网络的各个层被定义。首先是两层卷积层conv1和conv2，conv1接受2个输入通道，输出32个通道，卷积核大小为3x3，且在每个卷积层后都使用了padding=1，使得卷积后的图像大小保持不变。接下来是一个全连接层fc1，该层将卷积层输出的64×28×28的特征图展平成一维向量后，映射到一个128维的空间。最后是另一个全连接层fc2，将128维的向量映射为一个值，表示图像对的标签，即判断这两张图像是否属于同一类别。

forward方法定义了数据通过模型的前向传播过程。首先，输入数据通过conv1和conv2两层卷积网络，经过ReLU激活函数进行非线性变换，然后，卷积层输出的特征图被展平，以便送入全连接层fc1，同样，fc1的输出经过ReLU激活后进入fc2层，最终通过Sigmoid激活函数得到一个介于0和1之间的值，表示图像对是否属于同一类别。

该网络的输出是一个标量，范围在0到1之间，通过Sigmoid函数实现二分类。输出值接近0表示这两张图像是不同类别，接近1表示这两张图像属于同一类别。



图 4 model代码段

## 、训练与测试

完成卷积神经网络模型设计后，我们还需要设计训练过程及数据收集的函数。我们定义了calculate\_accuracy函数来计算模型在给定数据集上的准确率，利用了PyTorch的DataLoader来遍历数据，并使用torch.no\_grad()来禁用梯度计算以进行推理。并且我们在train\_model函数中，首先设置了损失函数和优化器，然后将训练数据和测试数据转换为PyTorch张量，并使用TensorDataset和DataLoader创建数据加载器。

在训练过程中，模型首先进入训练模式，使用反向传播更新模型的权重。每个epoch中，模型会在训练集上执行前向传播，计算损失，进行反向传播并优化参数。同时，模型还会计算并记录每个epoch的训练损失和准确率，训练轮数设置为10轮，足够进行多次参数更新，同时避免过拟合，学习率被设置为 0.001，这是一个通常选择的较小值，适合Adam优化器的默认设置。

测试集部分在每个epoch结束后计算，模型切换到评估模式，计算测试损失和准确率。每个epoch的训练和测试损失、准确率会被输出并记录，并且最终，损失曲线和准确率曲线会被绘制并保存为图片文件。

最后，train\_model函数返回训练和测试的最终准确率，该过程的输出包括每个epoch的损失和准确率，经过训练后，最终的模型能够对图像对进行分类判断。

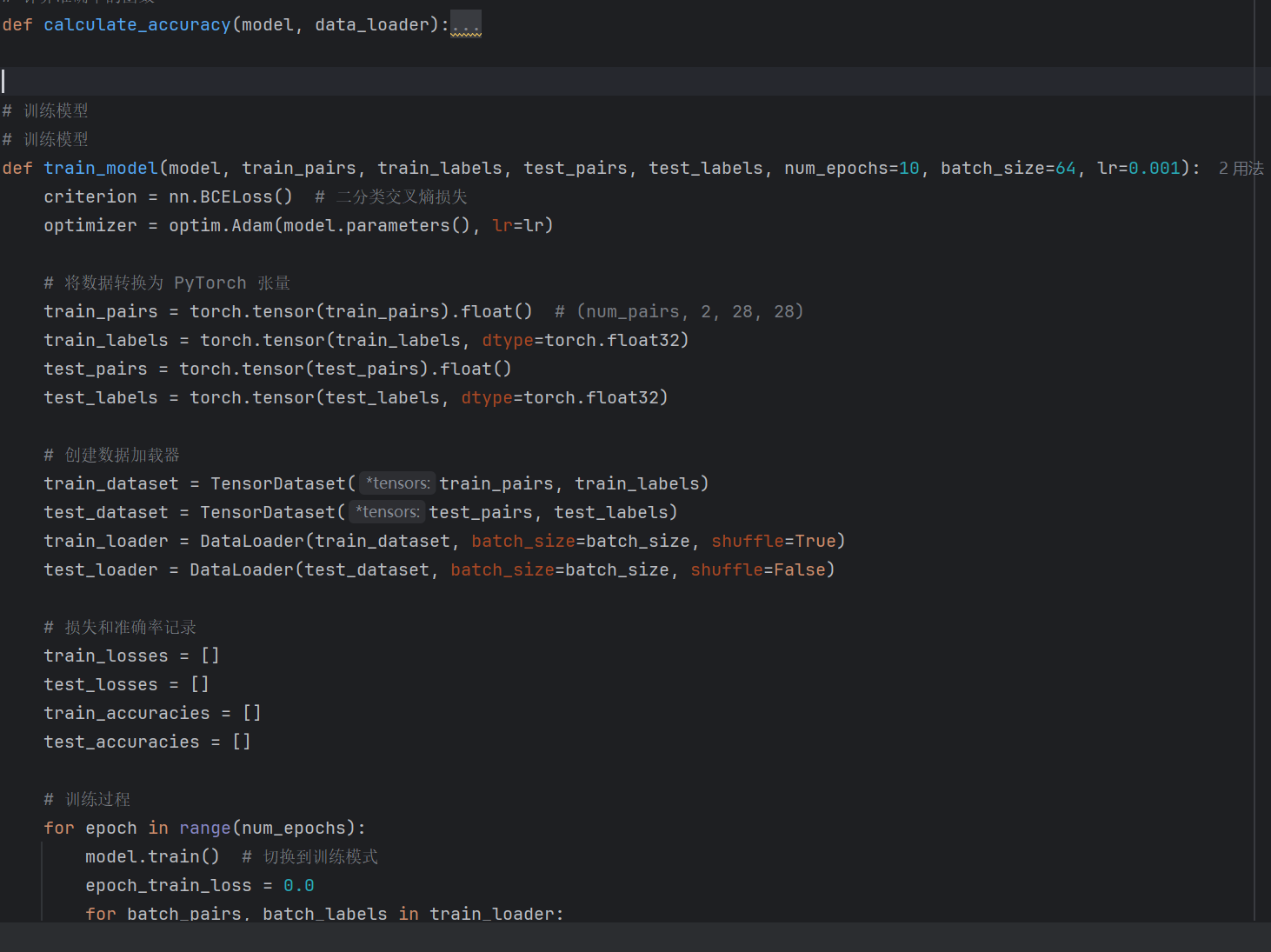


图 5 train中部分代码段

# 三、实验框架

## 、实验环境

操作系统：windows 11

CUDA：11.8

显卡：NVIDIA GeForce RTX 3070 Ti Laptop GPU

深度学习框架：PyTorch

开发工具：PyCharm Community Edition 2024.3.1

## 、卷积神经网络结构

本实验中设计的卷积神经网络结构如图所示，整体由卷积层、激活函数、全连接层组成，用于二分类任务。输入为两个28x28的图像，经过第一个3x3卷积层提取初步特征，紧接着通过ReLU激活函数引入非线性，然后进入第二个3x3卷积层进一步提取更深层次特征。特征提取完成后，数据展平并通过全连接层（128个神经元）进行特征映射，最后通过一个全连接层输出一个二分类的概率值，使用Sigmoid激活函数进行归一化处理。整个网络以轻量化的卷积结构为核心，简单高效地实现图像对的二分类任务。

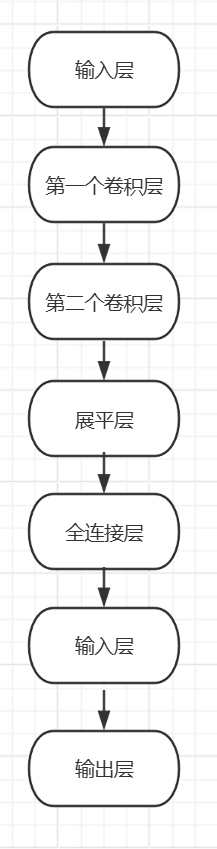


图 6

## 、实验结构

实验工程中，download为下载数据集文件，loader为数据加载和预处理文件，evaluate为实验结果分析问及那，main为主函数，moder为卷积模型文件，train为训练文件，每个.py文件分工明确，方便进行删改。

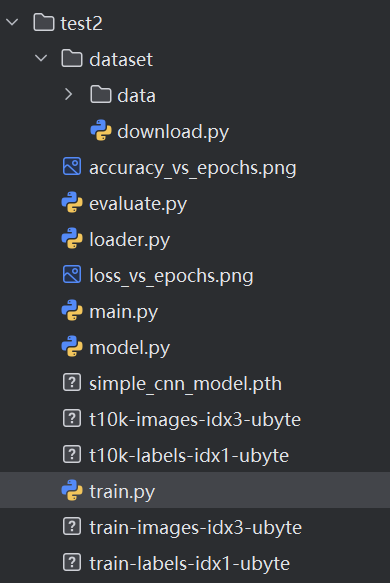


图 6 代码段4

# 实验结果

## 1、训练参数设置

在这个实验中，训练模型的参数设置如下：在网格搜索的过程中，最佳的超参数配置为：

* 激活函数：ReLU
* 学习率 ：0.001，使用的是Adam优化器，通常是一个较常用的学习率。
* 训练轮数：10轮，指的是整个训练数据集将会被模型处理的次数。
* 批量大小：64，表示每次梯度更新时使用的训练样本数量。
* 损失函数：使用的是二分类交叉熵损失函数BCELoss，适用于输出概率值并进行二分类的任务。

## 结果与分析

运行完试验后，根据所得到的实验数据可以分析：

· 训练损失：从Epoch 1到Epoch 10，训练损失从0.3462下降到0.0178，训练损失显著下降，说明模型在训练数据上逐渐学习得更好，正在不断优化，误差减少，拟合效果逐渐提升。

· 测试损失：测试损失的变化较为平稳，尽管每个epoch后有所波动，但整体上没有大幅度上升，保持在一个较低的范围内，大约在0.2附近。这种变化表明模型在测试集上表现稳定，没有出现过拟合或欠拟合的情况，表明模型能够很好地泛化。

· 训练准确率：训练准确率从92.15%提高到99.50%，说明模型成功地学习到了训练数据中的模式，并且达到了几乎完美的表现。

· 测试准确率：测试准确率从91.97%逐渐提高到95.51%，虽然测试准确率没有完全达到训练准确率的水平，但提高幅度较大，且在最后几轮基本稳定在95%左右。

训练损失下降幅度较大，表明模型在训练集上的拟合效果非常好；测试损失维持较低水平，且波动幅度较小，说明模型对新数据的泛化能力较强，训练准确率接近100%，测试准确率虽然稍低，但最终稳定在95%以上，显示了模型较强的识别能力。

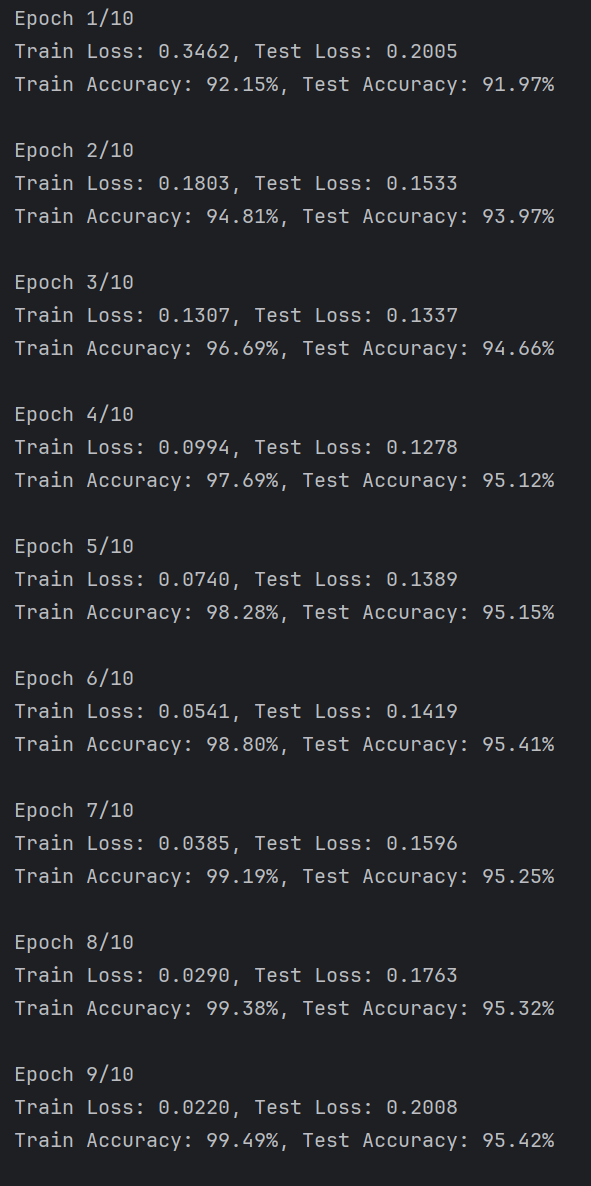


图 7 实验结果

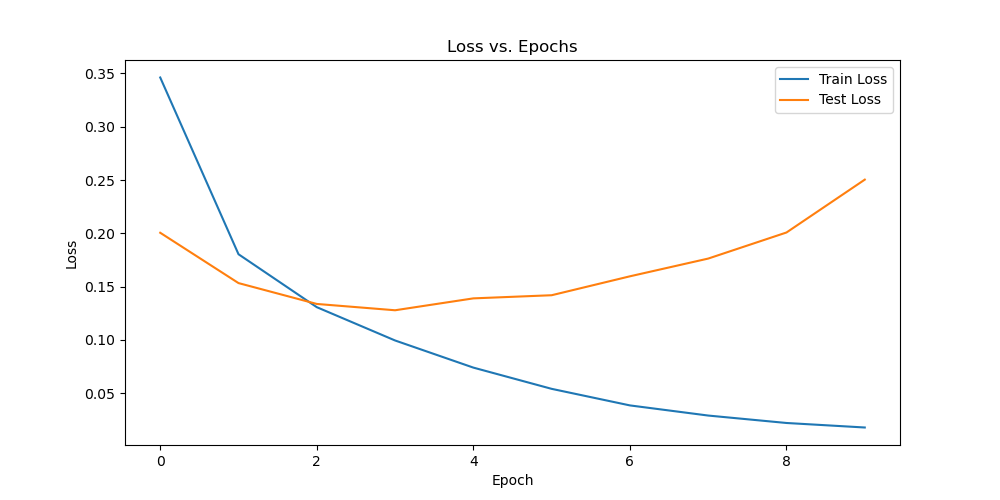


图 8 loss变化图

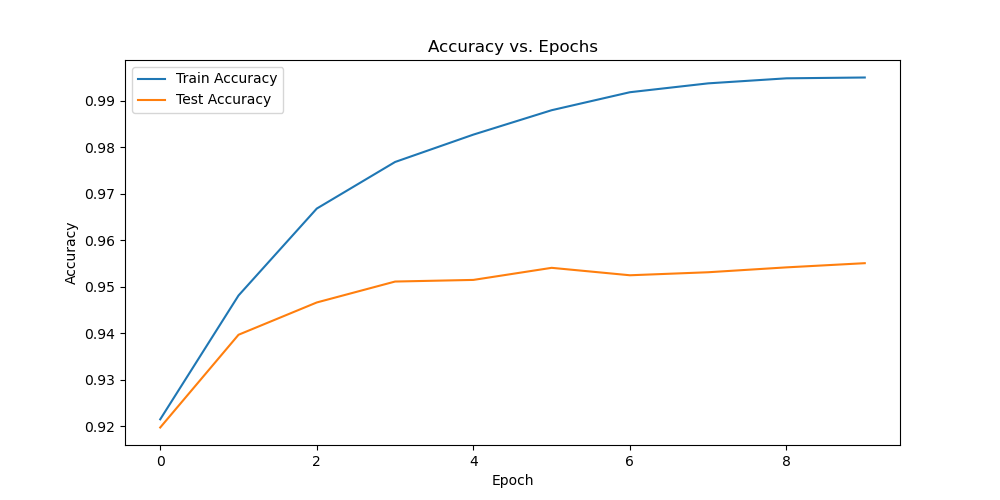


图 9 accuracy变化