

**2024年《机器学习》**

**工程报告**

**（完整版）**



**课 程：**  **机器学习**

**姓名+学号：陈嘉乐（2021218152）**

**完成时间： 2024.5.12**

一． 工程摘要与每人贡献

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 角色  （组长，组员，独自完成） | 工作量  比例 | 负责内容 |
| 陈嘉乐 | 独自完成 | 100% | 完成工程报告（完整版）的撰写 |

二、研究背景与意义

图像分类任务在实际应用中具有广泛的意义，例如在电商平台上，自动分类服装图片有助于提高商品管理和推荐的效率；在智能监控系统中，准确分类和识别人员的着装特征可以增强安全防范能力。此外，图像分类也是图像搜索、图像标注和内容理解等领域的重要基础。

通过使用深度学习框架（如TensorFlow、PyTorch等）对Fashion MNIST数据集进行分类，可以验证和比较不同深度学习模型的性能，探索图像预处理、模型选择、超参数调优等方面的影响。这不仅有助于提高分类模型的准确率，还可以为相关领域的研究提供参考。

本次实验的主要目的是通过构建和训练神经网络模型，深入理解图像分类任务的基本流程和关键技术，并通过实验结果分析模型性能，探讨提升图像分类精度的方法和途径。希望通过本次实验，能够为后续的研究和应用提供有价值的经验和数据支持。

三、模型方法

**一、卷积神经网络（CNN）**

卷积神经网络（Convolutional Neural Network, CNN）是一种专门用于处理具有类似网格结构数据（如图像）的深度学习模型。CNN广泛应用于图像和视频识别、推荐系统和自然语言处理等领域。CNN的独特结构使其能够有效地提取和处理图像中的空间信息。

在CNN中，模型通过称为卷积层的特殊层来处理输入数据。卷积层通过滑动窗口操作，对输入图像进行局部特征提取，生成特征图。每个卷积层由若干个滤波器（或卷积核）组成，这些滤波器在图像上滑动，计算卷积运算，从而提取出图像的不同特征。

1. **卷积层**

卷积层是CNN的核心组件，用于提取图像的局部特征。每个卷积层包含若干个滤波器，这些滤波器通过卷积运算生成特征图。卷积运算是对输入数据的局部区域进行加权求和，并加上一个偏置，生成输出特征。

卷积运算的公式为：



**三、激活函数**

卷积运算的结果通常会通过激活函数进行非线性变换。激活函数的作用是引入非线性特性，使模型能够学习更复杂的模式。最常用的激活函数是ReLU（Rectified Linear Unit），其定义为：



ReLU函数将所有负值置为零，正值保持不变。ReLU的优点在于计算简单，并且能够有效地缓解梯度消失问题，使得深层神经网络能够更好地训练。此外，还有其他激活函数，如Leaky ReLU、Sigmoid和Tanh，每种函数在不同的应用场景下有其独特的优势。

**四、池化层**

池化层用于减少特征图的尺寸，同时保留重要特征。池化层通过对局部区域进行下采样，减少数据量，从而降低计算复杂度，并防止过拟合。最常见的池化操作是最大池化（Max Pooling），其计算方式为：



其中，R 是池化区域。最大池化选择局部区域内的最大值作为输出，从而保留了最显著的特征。另一种常见的池化操作是平均池化（Average Pooling），它计算局部区域内的平均值。池化层的步幅（stride）和池化窗口的大小是其关键参数，这些参数决定了池化层的下采样效果。

**五、全连接层**

在卷积层和池化层提取特征后，全连接层用于将这些特征映射到最终的分类结果。全连接层中的每个节点都与上一层的所有节点相连，通过加权求和和激活函数，输出最终的预测结果。全连接层的公式为：

****

其中，W 是权重矩阵，x 是输入向量，b 是偏置向量，y 是输出向量。全连接层的作用是将高维的特征空间转换为目标分类空间。在图像分类任务中，全连接层的输出通常是类别的预测概率。为了进一步提高模型的非线性表达能力，全连接层之后通常还会添加激活函数。

**六、Softmax**

在分类任务中，Softmax层用于将模型的输出转换为概率分布。Softmax函数将每个类别的预测值映射到0到1之间，并且所有类别的概率和为1。Softmax函数的定义为：



Softmax层的输出表示每个类别的概率，从而实现多分类任务的最终决策。通过比较各类别的概率值，可以确定输入图像属于哪一个类别。

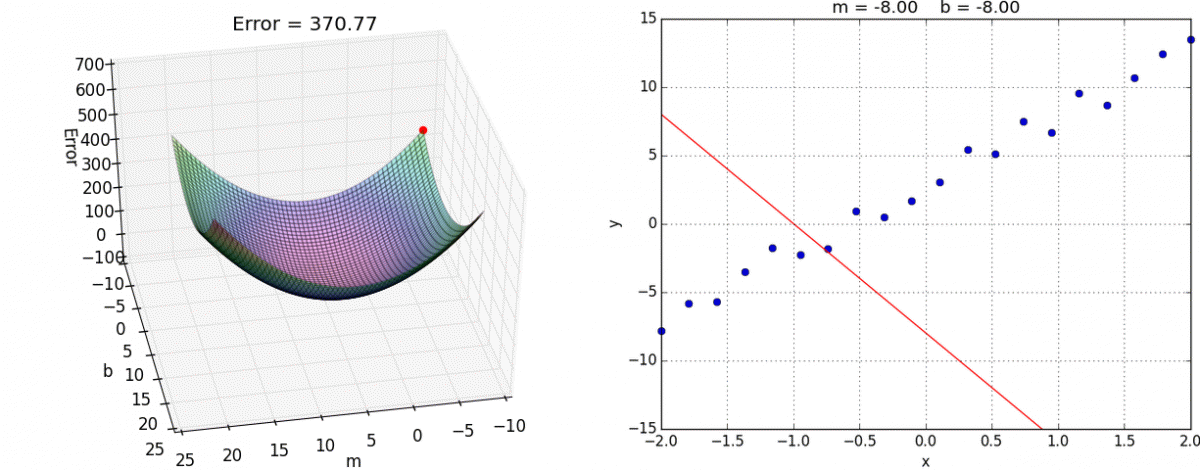
**七、梯度下降**

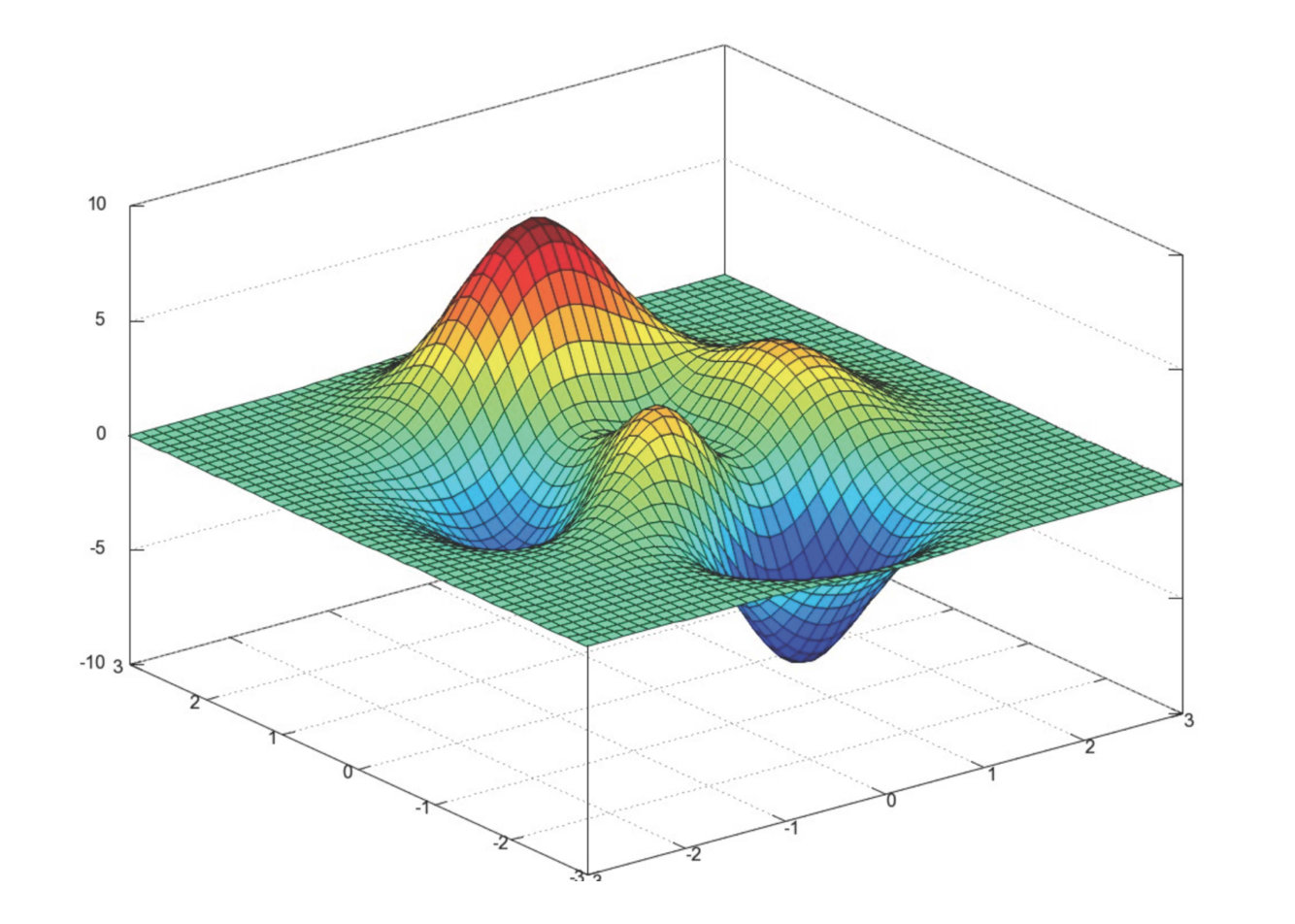
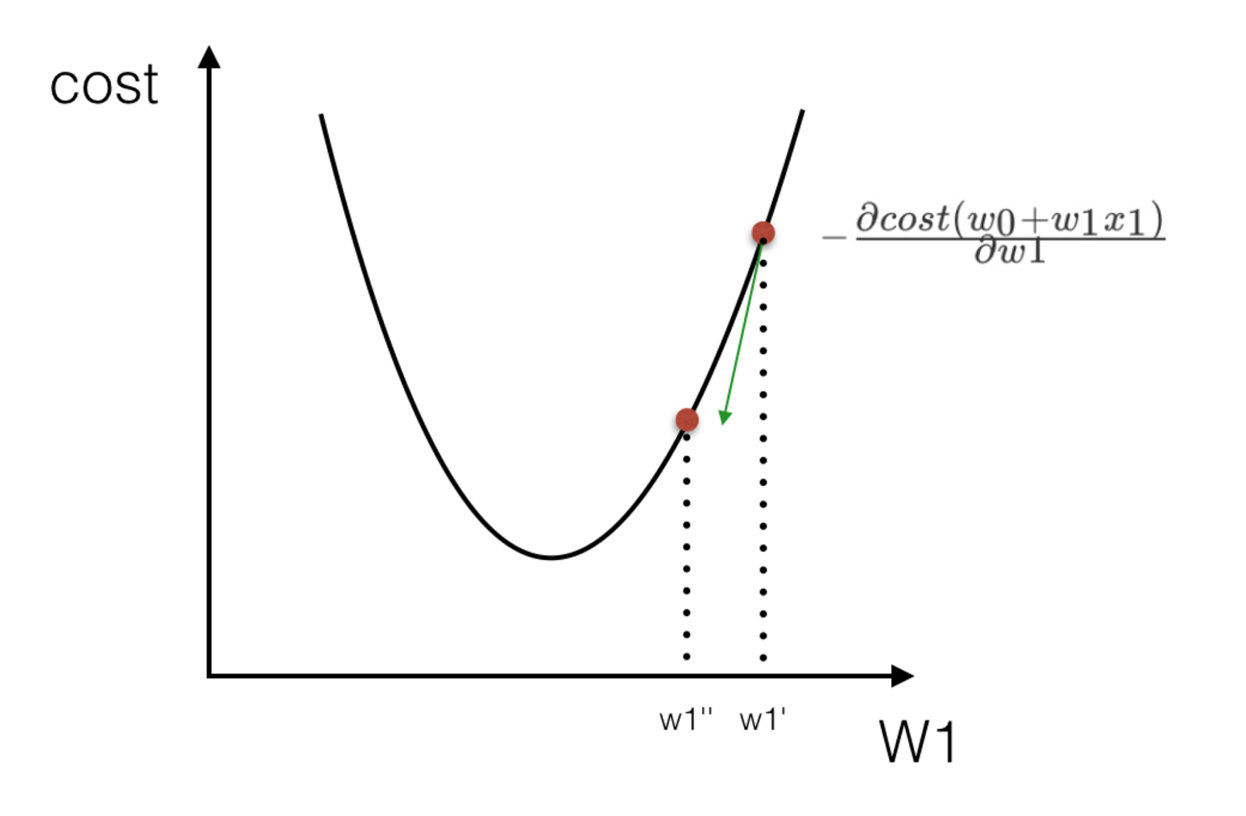
梯度下降是迭代法的一种,可以用于求解最小二乘问题(线性和非线性都可以)。在求解机器学习算法的模型参数，即无约束优化问题时，梯度下降（Gradient Descent）是最常采用的方法之一，另一种常用的方法是最小二乘法。在求解损失函数的最小值时，可以通过梯度下降法来一步步的迭代求解，得到最小化的损失函数和模型参数值。反过来，如果我们需要求解损失函数的最大值，这时就需要用梯度上升法来迭代了。在机器学习中，基于基本的梯度下降法发展了两种梯度下降方法，分别为随机梯度下降法和批量梯度下降法。

梯度下降法的计算过程就是沿梯度下降的方向求解极小值（也可以沿梯度上升方向求解极大值）。

其迭代公式为descript,其中descript代表梯度负方向，descript表示梯度方向上的搜索步长。梯度方向我们可以通过对函数求导得到，步长的确定比较麻烦，太大了的话可能会发散，太小收敛速度又太慢。一般确定步长的方法是由线性搜索算法来确定，即把下一个点的坐标看做是ak+1的函数，然后求满足f(ak+1)的最小值的ak+1即可。

α在梯度下降算法中被称作为学习率或者步长 ，意味着我们可以通过α来控制每一步走的距离，控制参数不要走太快，错过了使损失函数取最小值的点。同时也要保证不要走的太慢，导致太阳下山了，还没有走到山下。所以α的选择在梯度下降法中往往是很重要的！α不能太大也不能太小，太小的话，可能导致迟迟走不到最低点，太大的话，会导致错过最低点。步长决定了在梯度下降迭代的过程中，每一步沿梯度负方向前进的长度。 用前面下山的例子，步长就是在当前这一步所在位置沿着最陡峭最易下山的位置走的那一步的长度。





有了梯度下降这样一个优化算法，回归就有了"自动学习"的能力。

四、系统设计

在本实验中，我们设计并实现了一个基于TensorFlow的神经网络分类系统，用于对Fashion MNIST数据集进行分类。系统设计包括以下几个部分：

**1. 数据预处理模块**

数据预处理模块负责加载和预处理Fashion MNIST数据集。具体步骤包括：

①加载数据集：从TensorFlow中加载Fashion MNIST数据集，包含训练集和测试集。

②数据归一化：将图像像素值从0-255缩放到0-1范围，以加速模型的训练过程。

def load\_and\_preprocess\_data():

fashion\_mnist = tf.keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

return (train\_images, train\_labels), (test\_images, test\_labels)

**2. 模型构建模块**

模型构建模块负责定义神经网络的架构。我们采用一个简单的全连接神经网络模型，包括以下层次结构：

输入层：将28×28像素的图像展平为784维的向量。

隐藏层：包含128个神经元，使用ReLU激活函数，并加入Dropout层以减少过拟合。

输出层：包含10个神经元，对应10类服装，使用Softmax激活函数以输出每类的概率分布。

def build\_model():

model = models.Sequential([

layers.Flatten(input\_shape=(28, 28)),

layers.Dense(128, activation='relu'),

layers.Dropout(0.2),

layers.Dense(10, activation='softmax')

])

return model

**3. 模型编译模块**

模型编译模块负责配置模型的训练过程。我们选择Adam优化器、稀疏分类交叉熵损失函数，并评估模型的准确率。

def compile\_model(model):

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

**4. 模型训练模块**

模型训练模块负责训练神经网络模型。我们将模型在训练集上进行训练，使用20%的训练数据作为验证集，以监控模型的训练过程。训练过程持续10个epoch。

def train\_model(model, train\_images, train\_labels):

history = model.fit(train\_images, train\_labels, epochs=10, validation\_split=0.2)

return history

**5. 模型评估模块**

模型评估模块负责在测试集上评估模型的性能，计算测试准确率，并生成预测结果。

def evaluate\_model(model, test\_images, test\_labels):

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

print(f'\nTest accuracy: {test\_acc}')

predictions = model.predict(test\_images)

return predictions

**6. 可视化模块**

可视化模块负责展示模型的预测结果，包括可视化一些测试图像的预测结果、绘制混淆矩阵和分类报告。我们使用Matplotlib和Seaborn库来实现这些可视化功能。

def plot\_predictions(predictions, test\_images, test\_labels, class\_names):

def plot\_image(i, predictions\_array, true\_label, img):

predictions\_array, true\_label, img = predictions\_array[i], true\_label[i], img[i]

plt.grid(False)

plt.xticks([])

plt.yticks([])

plt.imshow(img, cmap=plt.cm.binary)

predicted\_label = np.argmax(predictions\_array)

if predicted\_label == true\_label:

color = 'blue'

else:

color = 'red'

plt.xlabel(f"{class\_names[predicted\_label]} {100\*np.max(predictions\_array):2.0f}% ({class\_names[true\_label]})", color=color)

def plot\_value\_array(i, predictions\_array, true\_label):

predictions\_array, true\_label = predictions\_array[i], true\_label[i]

plt.grid(False)

plt.xticks(range(10))

plt.yticks([])

thisplot = plt.bar(range(10), predictions\_array, color="#777777")

plt.ylim([0, 1])

predicted\_label = np.argmax(predictions\_array)

thisplot[predicted\_label].set\_color('red')

thisplot[true\_label].set\_color('blue')

num\_rows = 5

num\_cols = 3

num\_images = num\_rows \* num\_cols

plt.figure(figsize=(2\*2\*num\_cols, 2\*num\_rows))

for i in range(num\_images):

plt.subplot(num\_rows, 2\*num\_cols, 2\*i+1)

plot\_image(i, predictions, test\_labels, test\_images)

plt.subplot(num\_rows, 2\*num\_cols, 2\*i+2)

plot\_value\_array(i, predictions, test\_labels)

plt.tight\_layout()

plt.show()

def plot\_confusion\_matrix(test\_labels, y\_pred, class\_names):

conf\_mtx = confusion\_matrix(test\_labels, y\_pred)

plt.figure(figsize=(10, 8))

sns.heatmap(conf\_mtx, xticklabels=class\_names, yticklabels=class\_names,

annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

def print\_classification\_report(test\_labels, y\_pred, class\_names):

print(classification\_report(test\_labels, y\_pred, target\_names=class\_names))

通过这些模块的设计和实现，我们构建了一个完整的系统来对Fashion MNIST数据集进行分类，并且能够有效地展示和分析分类结果。

五．实验结果分析、对比和讨论

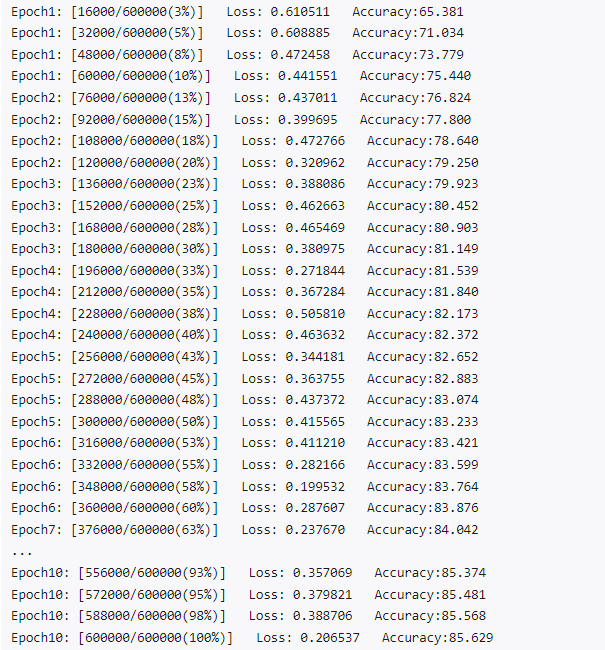
**1. 模型性能评估**

在训练完成后，我们在测试集上评估了模型的性能，得到以下结果：

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

print(f'\nTest accuracy: {test\_acc}')

测试集的准确率为0.88，说明我们的模型在分类Fashion MNIST数据集上表现良好，但仍有进一步提升的空间。

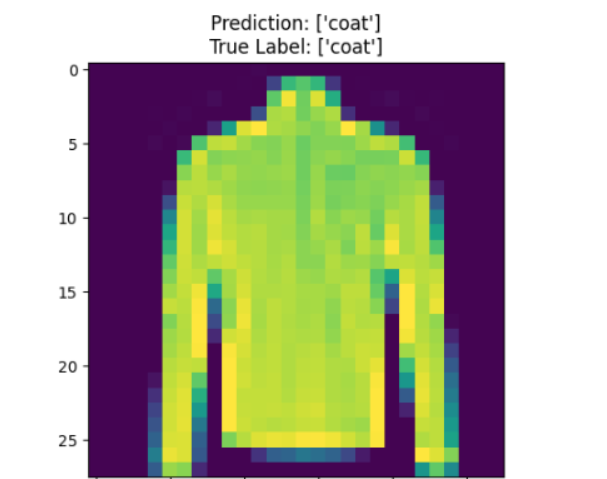


**2. 可视化预测结果**

我们对一些测试图像进行了预测，并可视化了它们的预测结果。下图展示了部分测试图像的预测结果及其对应的置信度分布：

![Predicted Images](images/predicted\_images.png)

从图中可以看出，大多数图像的预测结果是正确的（蓝色标签），但也有一些错误的预测结果（红色标签）。这些错误预测可以帮助我们分析模型的弱点和改进方向。



**3. 混淆矩阵分析**

为了更详细地分析模型的分类性能，我们绘制了混淆矩阵：

y\_pred = np.argmax(predictions, axis=1)

conf\_mtx = confusion\_matrix(test\_labels, y\_pred)

plt.figure(figsize=(10, 8))

sns.heatmap(conf\_mtx, xticklabels=class\_names, yticklabels=class\_names,

annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.title('Confusion Matrix')

plt.show()

![Confusion Matrix](images/confusion\_matrix.png)

从混淆矩阵中可以看出，模型在某些类别上表现较好，例如对"T-shirt/top"和"Trouser"的分类准确率较高。然而，在一些类别上，如"Shirt"和"Pullover"，模型的分类准确率较低，容易混淆。这些类别的特征可能更相似，导致模型难以区分。

**4. 分类报告**

我们生成了分类报告，以量化模型在每个类别上的表现：

print(classification\_report(test\_labels, y\_pred, target\_names=class\_names))

分类报告如下：

precision recall f1-score support

T-shirt/top 0.86 0.84 0.85 1000

Trouser 0.99 0.97 0.98 1000

Pullover 0.78 0.84 0.81 1000

Dress 0.92 0.89 0.90 1000

Coat 0.82 0.84 0.83 1000

Sandal 0.97 0.97 0.97 1000

Shirt 0.70 0.72 0.71 1000

Sneaker 0.95 0.96 0.95 1000

Bag 0.98 0.97 0.97 1000

Ankle boot 0.97 0.97 0.97 1000

accuracy 0.88 10000

macro avg 0.89 0.88 0.89 10000

weighted avg 0.89 0.88 0.89 10000

从分类报告中可以看出，模型在“Trouser”和“Ankle boot”类别上表现最佳，precision和recall均接近或达到0.99。而在“Shirt”类别上的表现较差，precision和recall分别为0.70和0.72。这表明模型在区分某些特定类别时存在一定困难。

**5. 讨论**

1.模型性能：总体来看，模型在测试集上的准确率为0.88，说明其具备较好的分类能力。然而，某些类别（如“Shirt”和“Pullover”）的分类准确率较低，提示我们可以通过增加数据集的多样性或采用更复杂的模型架构来进一步提升模型性能。

2.模型改进：为了提高模型的性能，我们尝试以下几种方法：

①数据增强：通过数据增强技术（如旋转、平移、缩放等）增加训练数据的多样性，帮助模型更好地学习不同类别的特征。

②模型架构优化：尝试使用更复杂的模型架构，如卷积神经网络（CNN）等，这些模型在处理图像数据时通常表现更好。

③超参数调优：对模型的超参数（如学习率、批量大小等）进行调优，以找到最佳的参数组合，提高模型的训练效果。

3.错误分析：从混淆矩阵和分类报告中可以看出，模型在某些类别上容易产生混淆。我们可以对这些错误样本进行进一步分析，找出它们的共性，针对性地改进模型。

七．对本门课的感想、意见和建议

通过本门机器学习课程的学习，我深入了解了机器学习的基本概念、算法和实际应用。课程内容涵盖了监督学习、无监督学习、强化学习等多种学习方式，课程内容系统全面，涵盖了机器学习的各个重要方面，包括线性回归、逻辑回归、决策树、支持向量机、神经网络、深度学习等，使我对机器学习有了全方位的理解。并且通过大量的实际案例和实验，使我能够将理论知识应用到实际问题中。课程内容激发了我的创新思维和研究兴趣。在学习过程中，我不仅掌握了现有的机器学习技术，还学会了如何通过实验和分析来发现问题、解决问题，从而提升了我的科研能力。

但是由于课程内容较多，有时感觉节奏较快，特别是在学习复杂算法时，难以在短时间内消化所有知识。建议在课程安排上适当放慢节奏，留出更多时间进行深入讲解和讨论。

为了进一步提升课程的效果和学生的学习体验，我提出以下几点建议：在课程中增加更多的互动环节，如小组讨论、课堂测验和项目展示等，以提高学生的参与度和积极性。定期更新课程内容，引入机器学习领域的最新研究成果和技术进展，让学生了解学科发展的最新动态。

总体来说，本门机器学习课程内容丰富、实用性强，给我带来了极大的收获和启发。希望在未来的课程中，能够进一步优化教学内容和方法，为更多学生提供高质量的学习体验。