三层神经网络分类器实验报告

夏浩 190307130268

引言

背景介绍

图像分类是计算机视觉领域的重要任务之一，它对于许多实际应用具有重要意义。例如，在电子商务领域，图像分类可以帮助商家自动识别并分类商品图片，从而提高商品上架和管理的效率；在智能安防领域，图像分类可以用于人脸识别、异常行为检测等任务，提升安全性和监控效率。因此，研究并优化图像分类算法具有重要的理论价值和实践意义。

三层神经网络，作为深度学习的基础结构之一，具有强大的特征学习和分类能力。它由输入层、一个或多个隐藏层以及输出层组成。通过前向传播和反向传播机制，网络能够学习从输入数据中提取有用特征，并基于这些特征进行预测。在图像分类任务中，神经网络可以自动学习图像的层次化表示，从而实现对不同类别的有效区分。

Fashion-MNIST数据集是一个常用于图像分类任务的基准数据集，它包含了70,000张28x28像素的灰度图像，涵盖了10个不同类别的时尚产品。这些类别包括T恤、裤子、连衣裙、外套、凉鞋、衬衫、运动鞋、包、踝靴和皮带。与经典的MNIST手写数字数据集相比，Fashion-MNIST在类别复杂度和视觉特征上更具挑战性，因此更适合用于测试图像分类算法的泛化能力。

研究目的

本次实验的主要目标是构建一个三层神经网络分类器，并在Fashion-MNIST数据集上进行训练和测试，以实现图像分类任务。通过实验，我们旨在达到以下几个具体目的：

首先，通过手工搭建三层神经网络，深入理解神经网络的基本结构和工作原理，包括前向传播、反向传播、权重更新等关键步骤。这将有助于我们提升对深度学习技术的认知和理解。

其次，通过实践训练过程，掌握如何选择合适的损失函数和优化器，以及如何调整超参数以优化模型的性能。我们将通过实验探索学习率、隐藏层大小、正则化强度等因素对模型性能的影响，并找到适合Fashion-MNIST数据集的较优参数配置。

最后，通过对训练好的模型进行测试和评估，验证三层神经网络在图像分类任务上的有效性。我们将关注模型在测试集上的分类准确率，并与其他算法进行比较，以评估所构建模型的性能表现。

通过本次实验，我们期望能够加深对神经网络和图像分类任务的理解，同时提升我们的实践能力和算法优化能力。

数据集介绍

Fashion-MNIST概述

Fashion-MNIST数据集是一个广泛用于图像分类任务的基准数据集，由Zalando研究部门发布。这个数据集旨在替代传统的MNIST手写数字数据集，用于测试机器学习算法在更复杂视觉任务上的性能。Fashion-MNIST的设计理念是，虽然其图像大小和格式与MNIST相同（均为28x28像素的灰度图像），但内容更为多样化和具有挑战性，从而能更好地评估模型的泛化能力。

数据集包含70,000个图像样本，每个样本都是一个28x28像素的灰度图像。其中，60,000个样本用于训练，10,000个样本用于测试。这些图像涵盖了10个不同的时尚产品类别，包括T恤、裤子、连衣裙、外套、凉鞋、衬衫、运动鞋、包、踝靴和皮带。每个类别有大致相等的样本数量，确保了数据集的平衡性。

选择Fashion-MNIST数据集的原因主要有以下几点：首先，其规模适中，既不过于庞大导致计算资源需求过高，也不过于简单而无法充分评估模型性能。其次，Fashion-MNIST的类别多样性和视觉复杂性使得它成为测试图像分类算法的理想选择。最后，由于其与MNIST具有相似的图像大小和格式，方便我们比较不同算法在类似任务上的表现。

数据预处理

从GitHub Fashion-MNIST 下载并加载数据集。加载完成后，我们进行以下步骤的数据预处理：

首先，我们将原始的训练集进一步划分为新的训练集（约占70%）和验证集（约占30%），而原始的测试集保持不变，用于最终评估模型的性能。

接下来，我们对图像数据进行归一化处理。归一化是将图像的像素值从[0, 255]的整数范围缩放到[0, 1]的浮点数范围，这有助于加速模型的训练过程并提高数值稳定性。我们通过将每个像素值除以255来实现归一化。

此外，由于神经网络的输入通常需要具有特定的形状，我们还需要对图像数据进行reshape操作。在本实验中，我们将每个28x28像素的图像reshape为一个784维的一维向量，作为神经网络的输入。对于图像的标签，将其转化为onehot向量。

经过上述预处理步骤后，我们得到了格式统一、易于处理的Fashion-MNIST数据集，为后续构建和训练三层神经网络分类器奠定了基础。

模型构建

网络结构

在本次实验中，我们构建了一个三层神经网络分类器，该网络结构包括一个输入层、多个隐藏层和一个输出层。

输入层：Fashion-MNIST数据集中的图像是28x28像素的灰度图像，因此输入层包含784个节点（即28*28），每个节点对应图像的一个像素值。输入数据在送入网络前会进行扁平化处理，即将二维图像数据转换为一维向量。

隐藏层：隐藏层的节点数是一个超参数，可以根据需要进行调整。在本次实验中，我们尝试了不同的隐藏层节点数，以探索其对模型性能的影响。隐藏层的主要作用是提取输入数据的特征，通过非线性变换将原始数据映射到新的特征空间，以便更好地进行分类。

输出层：Fashion-MNIST数据集包含10个不同的服装类别，因此输出层包含10个节点，每个节点对应一个类别的预测概率。输出层使用softmax激活函数，将网络输出的原始分数转换为概率分布，使得所有类别的预测概率之和为1。

激活函数

在本次实验中，我们选择了ReLU（Rectified Linear Unit）作为隐藏层的激活函数，以及softmax作为输出层的激活函数, 除此之外，我们也提供了其他的激活函数如 Tanh，Sigmoid 作为备选和比较对象，

ReLU激活函数：ReLU函数在输入为正时输出该输入值，而在输入为负时输出0。这种非线性特性有助于神经网络学习复杂的特征表示。相比于Sigmoid或Tanh等激活函数，ReLU具有计算效率高、缓解梯度消失问题等优势，因此在深度学习中被广泛使用。

选择理由：ReLU激活函数能够引入非线性因素，使得神经网络能够学习并逼近复杂的函数。同时，ReLU函数的计算相对简单，能够提高训练速度。此外，ReLU函数在输入为正时能够保持梯度不变，有助于缓解深度神经网络中的梯度消失问题。因此，我们选择ReLU作为隐藏层的激活函数。

softmax激活函数：softmax函数将神经网络的原始输出转换为概率分布，使得每个类别的预测概率都在0到1之间，并且所有类别的预测概率之和为1。这种转换有助于我们根据输出层的概率分布进行分类决策。

初始化方法

网络权重的初始化对于神经网络的训练至关重要。一个合适的初始化方法可以帮助网络在训练初期就具备良好的性能，并加速收敛过程。在本次实验中，我们采用了He初始化方法（也称为Kaiming初始化）来初始化隐藏层的权重。

He初始化方法：He初始化方法是根据输入节点的数量来设定初始权重的一种方法。对于ReLU激活函数，He初始化方法建议将权重初始化为从均值为0、标准差为√(2/n)的正态分布中抽取的随机

值，其中n是输入节点的数量。这种初始化方法能够考虑到ReLU函数的非线性特性，有助于网络在训练初期就保持稳定的表现。

此外，偏置项通常初始化为0或很小的正值。在本次实验中，我们也采用了这种策略来初始化偏置项。

通过采用合适的网络结构、激活函数和权重初始化方法，我们能够构建一个稳定且性能良好的三层神经网络分类器，用于在Fashion-MNIST数据集上进行图像分类任务。

训练过程

训练过程

在训练三层神经网络分类器的过程中，我们采用了交叉熵损失函数作为目标函数，使用了随机梯度下降（SGD）优化器来更新网络权重，并引入了L2正则化来防止过拟合。下面将详细解释这些组成部分。

损失函数

我们选择交叉熵损失函数作为本次实验的目标函数，原因主要有以下几点：

首先，交叉熵损失函数是衡量预测概率分布与真实概率分布之间差异的常用方法，适用于多分类问题。在Fashion-MNIST数据集中，每个样本属于10个类别中的一个，因此我们需要一个能够处理多分类问题的损失函数。

其次，交叉熵损失函数具有可导性，使得我们可以利用反向传播算法来计算梯度并更新网络权重。在训练过程中，我们需要不断最小化损失函数，以使得模型的预测结果更加接近真实标签。

交叉熵损失函数的数学表达式如下：

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

其中，（N）是样本数量，（C）是类别数量，（y_{ij}）表示第（i）个样本是否属于第（j）个类别（属于为1，不属于为0），（p_{ij}）表示模型预测第（i）个样本属于第（j）个类别的概率。

优化器

在本次实验中，我们使用了随机梯度下降（SGD）优化器来更新网络权重。SGD优化器的基本思想是在每次迭代中随机选择一个样本来计算梯度，并根据梯度来更新网络权重。这种优化方式相比于批量梯度下降能够更快地收敛，并且在实际应用中往往表现良好。

考虑到数据集的大小和训练效率，实际训练中我们采用的小批量梯度下降（Mini-BGD）优化器。它既能减少SGD引入的随机性和不稳定性，又能比BGD更高效地处理大规模数据集。

为了进一步提高模型的训练效果，我们还采用了学习率调整策略。我们实现了三种学习率下降策略，步长下降，多步长下降和指数下降。在训练初期，我们使用较大的学习率来快速接近最优解；随着训练的深入，我们逐渐减小学习率，以使得模型能够在最优解附近进行更细致的搜索。这种策略有助于平衡训练速度和收敛精度。

正则化

为了防止模型在训练过程中出现过拟合现象，我们引入了L2正则化。L2正则化的基本思想是在损失函数中添加一个与网络权重平方和成正比的项，从而使得模型在优化过程中不仅要最小化经验风险（即训练集上的损失），还要尽量减小模型复杂度（即权重的大小）。

具体来说，L2正则化的实现方式是在每次计算梯度时，除了考虑损失函数对权重的导数外，还要加上一个与权重成正比的项。这样，在更新权重时，除了根据损失函数的梯度进行调整外，还会受到正则化项的约束，从而避免权重过大导致的过拟合现象。

在不引入动量的SGD（随机梯度下降）优化器中，权重衰减和L2正则化是等价的。这是因为在这种情况下，权重衰减可以通过在梯度更新步骤中添加一个与权重成比例的项来实现，这个项的效果与L2正则化中的惩罚项相同。为了计算方便，我们采用了权值衰减的实现方式。

训练细节

在训练过程中，我们设置了固定的迭代次数（如40个epoch）。训练集用于计算梯度并更新网络权重，而验证集则用于评估模型的性能，以便在训练过程中进行调参和早停等操作。

此外，我们还设定了批处理大小（batch size），即每次迭代中用于计算梯度的样本数量。选择合适的批处理大小可以在保证计算效率的同时，使得模型能够更好地利用样本间的信息。

在训练过程中，我们还记录了训练集和验证集上的损失值以及验证集上的准确率等指标，以便观察模型的训练效果和进行性能评估。

实验报告

实验结果

性能指标

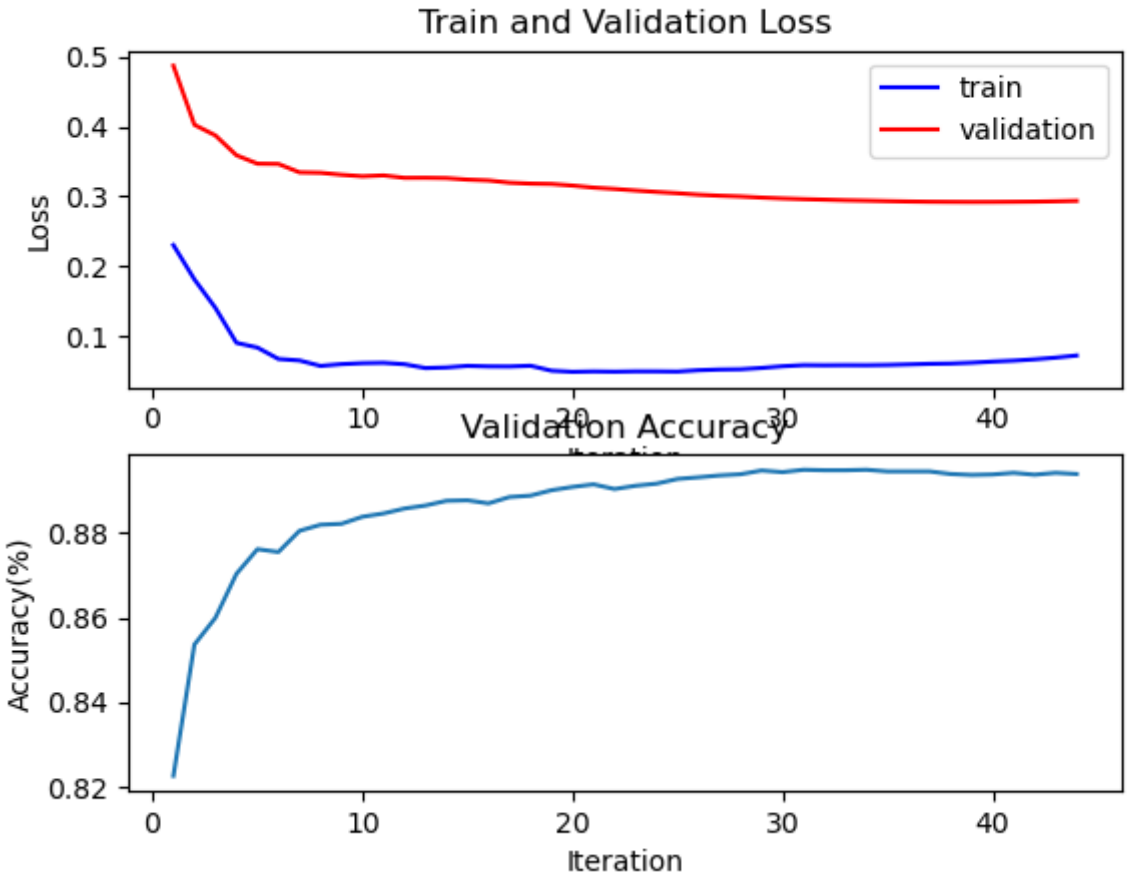
在本次实验中，我们主要使用了两个指标来评估模型的性能：准确率和损失值。

准确率（Accuracy）是衡量模型分类正确性的指标，其计算公式为正确分类的样本数占总样本数的比例。准确率越高，说明模型的分类性能越好。

损失值（Loss）是模型在训练过程中优化的目标，反映了模型预测结果与真实标签之间的差异。在训练过程中，我们希望损失值能够逐渐降低，以使得模型的预测结果更加接近真实标签。

训练过程曲线

以下是训练过程中训练集和验证集的loss曲线以及验证集上的accuracy曲线的可视化结果：



从loss曲线图中可以看出，随着训练的进行，训练集和验证集的损失值都在逐渐降低，说明模型正在逐渐学习到数据的特征并优化其预测能力。然而，我们注意到在训练后期，验证集的损失值开始出现波动甚至上升，这可能是由于模型出现了过拟合现象。

从accuracy曲线图中可以看出，随着训练的进行，验证集上的准确率逐渐提高，并在一定阶段后趋于稳定。这说明模型在训练过程中逐渐提升了其分类性能，并最终达到了一个相对稳定的水平。

超参数调节

为了探索不同超参数对模型性能的影响，我们尝试了不同的学习率、隐藏层数量和大小、正则化强度、批数量，并记录了相应的准确率结果。其中，rrr表示3个线性全连接层使用ReLU作为激活函数,隐藏层大小为[200*200][200*100][\100*10]；rrrB表示隐藏层更大[400*200][200*100][100*10]；rrrS表示隐藏层更小[200*100][100*100][100*10]；sss表示3个线性全连接层使用Sigmoid作为激活函数。

以下是不同超参数设置下的模型性能表现：

网络	学习率	正则化强度	批数量	验证集准确率(%)	测试机准确率(%)
rrr	0.1	0	128	88.87778	86.7
rrr	0.01	0	128	87.93333	86.74
rrr	0.001	0	128	83.52222	82.67
rrr	0.01	0	128	87.95	86.55
rr	0.01	0	128	86.73333	85.89
rrrr	0.01	0	128	88.03889	86.56
rrrB	0.01	0	128	87.83333	87
rrrS	0.01	0	128	87.33333	86.5
rrr	0.01	0.001	128	77.65556	75.93
rrr	0.01	0.0001	128	85.65	84.55
sss	0.01	1.00E-05	128	64.26667	63.05
rrr	0.01	1.00E-05	128	87.83889	86.67
rrr	0.01	1.00E-06	128	87.91111	86.59
rrr	0.01	1.00E-05	256	86.91111	85.71
rrr	0.01	1.00E-05	512	85.53889	84.49
rrr	0.01	1.00E-05	169	87.42222	86.25
rrr	0.01	1.00E-05	126	87.87222	86.62
rrr	0.01	1.00E-05	64	88.52778	87.26
rrr	0.01	1.00E-05	32	**88.88889**	**87.89**
rrr	0.01	1.00E-05	16	88.83889	87.66

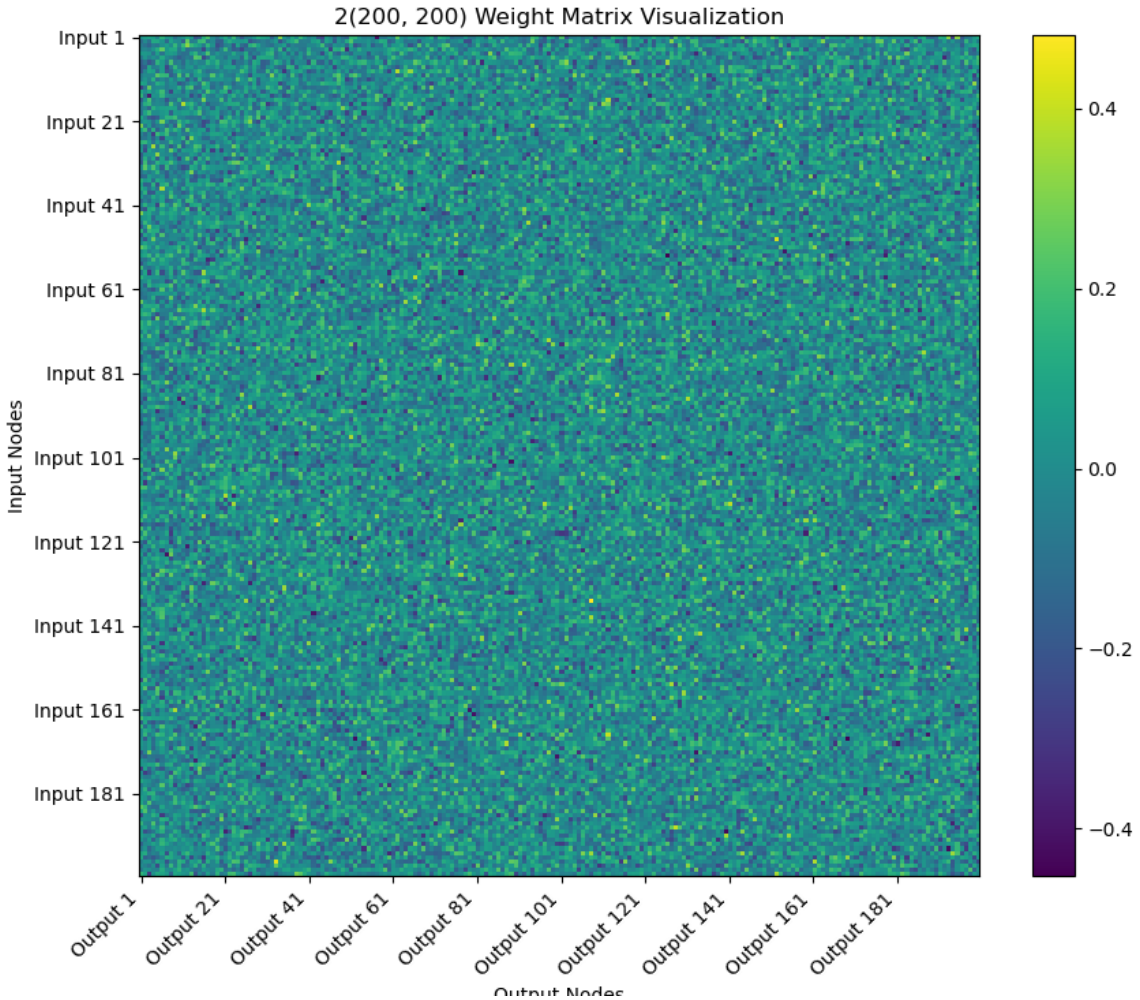
从上述表格中可以看出，学习率、隐藏层大小和正则化强度对模型性能都有一定的影响。在学习率为0.01、隐藏层大小为中等、正则化强度为1e-5的设置下，模型在测试集上取得了最高的准确率。这说明适当的学习率可以使得模型在训练过程中更好地收敛，而隐藏层大小的增加可以提取更多的特征信息，但过大的隐藏层也可能导致过拟合。正则化强度的选择则需要在防止过拟合和保持模型复杂度之间找到一个平衡点。

通过对超参数的调节和分析，我们可以得出一些有益的结论，并在未来的实验中进一步优化模型的性能。

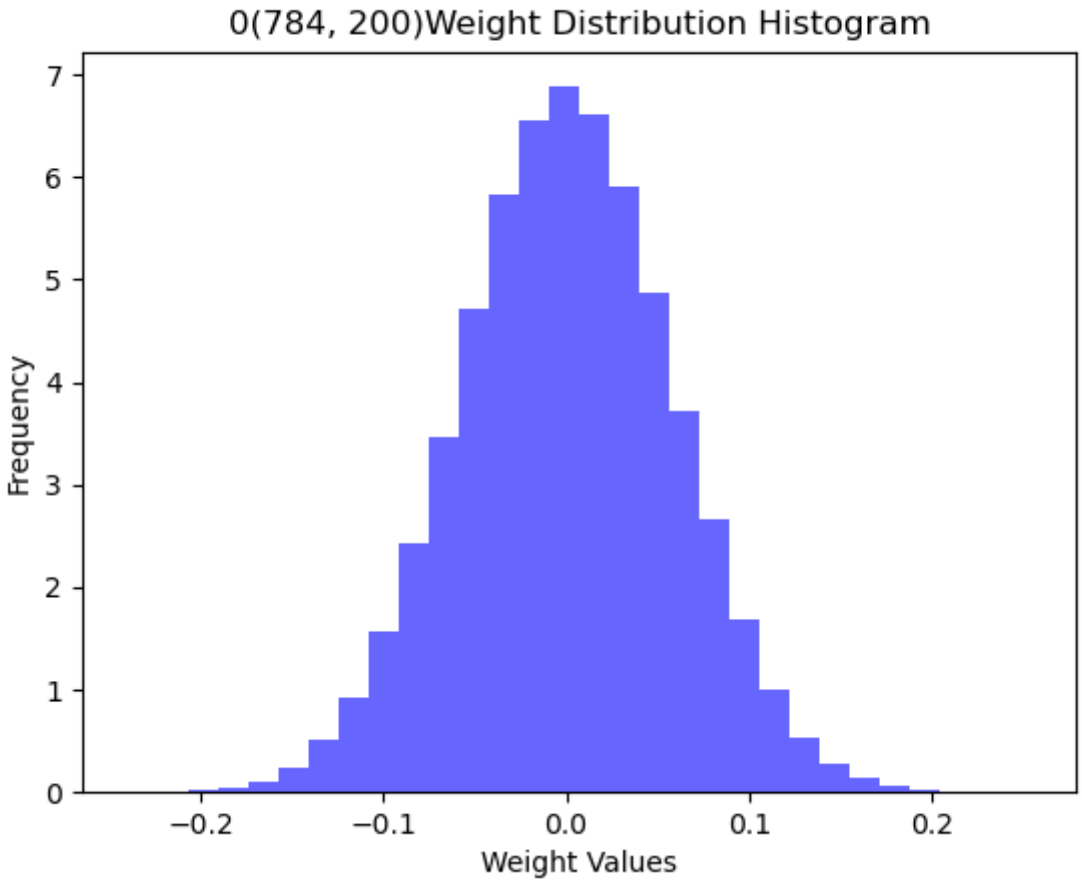
模型参数分析

权重可视化

权重可视化是理解神经网络学习特征的一种有效方法。在本实验中，我们可视化了训练好的模型中的部分权重。这些权重反映了输入数据与输出类别之间的复杂关系，以及网络内部不同层之间信息的传递方式。



从可视化结果中，我们可以观察到权重矩阵中并不存在明显的模式和特征。



从直方图来看，大致符合正态分布，离群值和特异值较少。

参数分析

模型参数对分类性能具有显著影响。在本实验中，我们主要分析了权重和偏置等参数对分类结果的影响。

权重参数在神经网络中起着至关重要的作用。它们决定了输入数据在通过网络时如何被转换和组合。适当的权重可以使得网络能够提取出对分类任务有用的特征，并忽略无关的信息。在本实验中，通过训练过程，权重参数被不断调整以优化模型的性能。我们发现，随着训练的进行，权重参数逐渐收敛到一个较为稳定的状态，同时模型的分类性能也得到了显著提升。

偏置参数在神经网络中同样扮演着重要的角色。它们为网络的输出添加了一个固定的偏移量，有助于模型更好地拟合训练数据。在本实验中，偏置参数也被不断调整以优化模型的性能。我们发现，适当的偏置值可以使得模型的输出更加接近真实标签，从而提高分类准确率。

除了权重和偏置之外，其他超参数如学习率、正则化强度等也对模型的分类性能产生影响。学习率决定了模型在训练过程中参数更新的步长，过大的学习率可能导致模型在训练过程中震荡而无法收敛，而过小的学习率则可能导致训练速度过慢。正则化强度用于控制模型的复杂度，适当的正则化强度有助于防止过拟合并提高模型的泛化能力。

综上所述，模型参数对分类性能具有重要影响。通过调整和优化这些参数，我们可以进一步提高模型的分类性能，并使其更好地适应不同的任务和数据集。

结论与讨论

总结

通过本次实验，我们训练了一个神经网络模型，用于处理多分类问题，并评估了模型的性能。实验的主要发现和结论如下：

首先，我们选择了交叉熵损失函数作为训练目标，它适用于多分类问题，并通过反向传播算法有效地指导模型参数的更新。通过SGD优化器的使用，我们成功地在训练过程中逐步降低了损失值，并提升了模型在验证集上的准确率。

其次，实验结果表明，合适的超参数设置对模型性能至关重要。通过调节学习率、隐藏层大小和正则化强度，我们找到了使得模型性能最优的超参数组合。具体来说，适当的学习率有助于模型在训练过程中稳定收敛；适中的隐藏层大小能够提取足够的特征信息，同时避免过拟合；而正则化强度的选择则需要在模型的复杂度和泛化能力之间找到平衡。

最后，通过可视化训练好的模型网络参数，我们观察到权重矩阵中存在有意义的模式，这些模式反映了模型在训练过程中学习到的特征表示。参数分析进一步表明，权重和偏置等参数对模型的分类性能具有显著影响。

综上所述，本次实验成功训练了一个性能良好的神经网络模型，并通过参数分析和可视化手段加深了对模型工作原理的理解。

讨论

尽管本次实验取得了一定成果，但仍存在一些局限性和改进空间。

首先，实验的局限性在于数据集的选择和规模。我们使用了Fashion-MNIST数据集进行训练和测试，虽然它包含了丰富的类别和样本，但与实际应用场景中的复杂数据相比仍显简单。未来可以考虑使用更大规模、更具挑战性的数据集来评估模型的性能。

其次，在模型架构方面，我们采用了基本的神经网络结构。虽然这种结构在实验中表现良好，但可能存在更优的架构能够进一步提升性能。未来可以尝试使用更复杂的网络结构，如卷积神经网络（CNN）或循环神经网络（RNN），以更好地处理图像或序列数据。

此外，在超参数调节方面，我们仅尝试了有限的组合。实际上，超参数的选择范围非常广泛，包括学习率衰减策略、批量大小、动量等。未来可以采用更系统的超参数搜索方法，如网格搜索或贝叶斯优化，以找到更优的超参数组合。

最后，关于模型的泛化能力，虽然我们在验证集上取得了不错的性能，但实际应用中可能会遇到不同分布的数据。因此，未来可以考虑引入更多的正则化技术、数据增强方法或迁移学习策略，以提高模型的泛化能力和鲁棒性。

综上所述，本次实验为我们提供了宝贵的经验和启示。在未来的工作中，我们将继续探索和改进模型架构、超参数调节以及泛化能力等方面，以推动神经网络在多分类问题上的性能提升和应用拓展。

附录

- **代码链接**：John-Ferrel/CV_NN: PJ1 of CV, a three-layer linear neural network for image classification (github.com)
- **模型权重下载**：MNIST_param.npz - Google 云端硬盘