HW3 手写字符识别

1 数据预处理

本次实验中采用 Omniglot 数据集，进行手写字符识别分类的任务。其中每张图片为 28×28 的黑白像素点，使用助教所给的代码 `utils.py` 进行图片数据的读取以及训练/测试集数据的划分，随机从所有类别中取出 50 类进行分类，每个类别中使用 15 张图片作为训练数据，5 张图片作为测试数据。完成图片像素点的读取后发现，每张黑白图像由接近的两个像素值 0.99607843 和 1 组成，为了使模型能够更好的识别区分特征，即文字和背景，将得到的图像数组二值化，使每个像素点的值只为 0 或 1，进行图片特征的增强。由于本地运行环境问题，`utils.py` 中只**更改了图片的读取方式**，将之前的 `misc.imread()` 更改为了 `imageio.imread()`。

对所有图像处理完成后，每张图像对应一个大小为 28×28 的 0-1 二维数组，通过 pytorch 中的 `TensorDataset` 将**图像-标签对**转化为张量数据集，通过 `Dataloader` 构建迭代器，方便后续的训练过程。

2 模型实现

在本次作业中基于 `pytorch` 实现了全连接神经网络 (MLP) 和卷积神经网络 (CNN) 对该任务进行了尝试。考虑到本次作业的数据集较简单，综合考虑模型的收敛性和训练速度，设置学习率为 0.1，`bacth_size` 为 32，损失函数为交叉熵损失 `CrossEntropyLoss()`，优化器为随机梯度下降 `torch.optim.SGD()`，随机选取训练集中 20% 的数据作为验证集。网络最终的输出为长度为 50 的分类标签向量，根据 `argmax()` 操作得到最终网络预测的分类标签。

2.1 全连接神经网络 (MLP)

使用 `pytorch` 中的 `nn.Linear` 构建全连接神经网络，包含两个隐藏层，网络的结构如下：

```
MLP(
  (linear_layer): Sequential(
    (0): Linear(in_features=784, out_features=1024)
    (1): BatchNorm1d(1024)
    (2): ReLU()
    (3): Dropout(p=0.2)
    (4): Linear(in_features=1024, out_features=512)
    (5): BatchNorm1d(512)
    (6): ReLU()
    (7): Dropout(p=0.8)
```

```
      (8): Linear(in_features=512, out_features=50)
    )
  )
```

对于输入的一张 28×28 的二维图像，将其展平为长度为 784 的向量作为模型的输入，包含大小为 1024 和 512 的两个隐藏层，分别用于特征的展开和提炼，使用 `ReLU()` 作为模型中的激活函数。

使用全连接神经网络进行训练，设置最大训练轮数为 100，当达到最大轮数时停止训练，选取验证集上表现最好的模型作为最终的模型。在测试集上进行验证，得到的分类准确率为：**0.512**。

2.2 卷积神经网络 (CNN)

使用 `pytorch` 中的 `nn.Conv2d` 构建卷积神经网络，网络的结构如下：

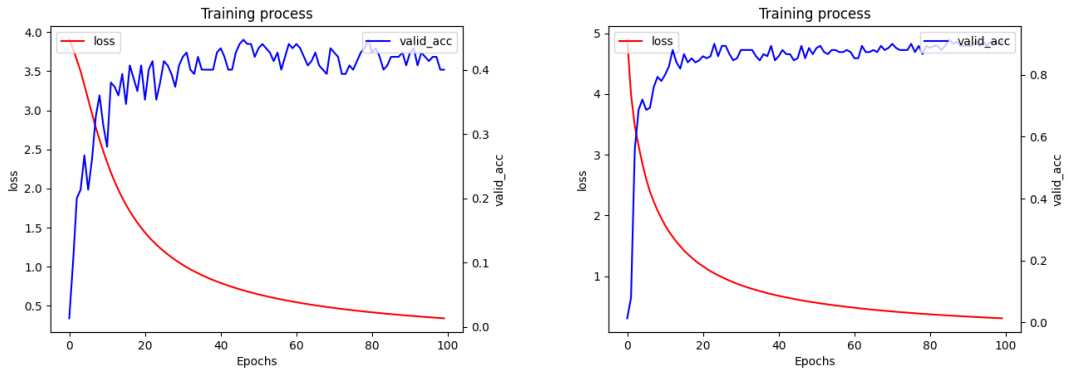
```
CNN(
 (block1): Sequential(
    (0): Conv2d(1,32,kernel_size=(3,3),stride=(1,1),padding=valid)
    (1): BatchNorm2d()
    (2): ReLU()
    (3): Conv2d(32,32,kernel_size=(3,3),stride=(1,1),padding=valid)
    (4): BatchNorm2d()
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2,stride=2,padding=0)
  )
  (block2): Sequential(
    (0): Conv2d(32,64,kernel_size=(3,3),stride=(1,1),padding=valid)
    (1): BatchNorm2d()
    (2): ReLU()
    (3): Conv2d(64,64,kernel_size=(3,3),stride=(1,1),padding=valid)
    (4): BatchNorm2d()
    (5): ReLU()
    (6): MaxPool2d(kernel_size=2,stride=2,padding=0)
  )
  (flatten):Flatten(start_dim=1,end_dim=-1)
  (linear):Sequential(
    (0): Linear(in_features=1024,out_features=512)
    (1): BatchNorm1d()
    (2): ReLU()
    (3): Dropout(p=0.8)
    (4): Linear(in_features=512,out_features=50)
  )
)
```

全连接神经网络使用两个卷积块 (block) 对图像进行卷积运算，进行特征提取，所有卷积操作均采用大小为 3×3 的卷积核，每个 block 中有两个卷积层，第一个 block 中通道数为 32，第二个 block 中通道数为 64，进行了特征的提取和富化，激活函数采用 `ReLU()`，在每个 block 最后对卷积提取的特征进行二维的最大池化操作。最后将经过卷积层得到的二维特征展平输入一个全连接网络，进行特征的提炼得到最后的分类标签输出。

采用与全连接网络相同的训练参数设置，使用卷积神经网络分类得到最终的准确率为 **0.868**，相比于全连接神经网络有了比较明显的提升。

2.3 两种模型效果的对比

两种模型训练的过程如图1所示:



(a) MLP (b) CNN

Figure 1: Training Process

在模型方面，全连接神经网络结构更加简单，在收敛过程中验证集上的表现相对卷积神经网络波动更大。同时，全连接网络相比卷积神经网络中权重共享的特性来说，参数量更大，训练成本更高。

在最终效果方面，卷积神经网络的效果明显优于全连接神经网络，因为图片中包含的特征天然是二维的，而卷积神经网络中二维的卷积核更适用于这种二位特征的提取；全连接神经网络将图片展平为一维的向量，会造成图像中二位空间位置相关特征的丢失，所以效果不好。

3 实验及对比

在这一小节中，将从不同的方面对实验条件进行改变，观察对实验结果的影响。

3.1 数据集划分

首先对数据集采用不同的划分策略进行实验。首先对类别数量、训练/测试样本数量的组合进行不同的尝试，采用效果较好的 CNN 模型进行实验，得到分类的准确率：

Table 1: 不同数据集划分下的结果

classes/split	(5,15)	(10,10)	(15,5)	(18,2)
25	0.603	0.788	0.848	0.92
50	0.573	0.808	0.868	0.88
100	0.575	0.778	0.836	0.87

从结果可以看出，对于类别的数量，随着分类数量的逐渐增大，最终模型的分类准确率逐渐降低。这种变化趋势是符合常识认知的，因为类别数目越多，说明分类问题的难度越大，模型分类错误的概率越大，所以准确率降低。对于数据集的划分，随训练集比例的增大，模型在测试集上的准确率表现逐渐提升，因为训练集越大，模型“见过”的图片输入越多，就能更好地提取各种特征，更精确地逼近手写数字图片的特征分布，取得更高的分类准确率。

综合分类任务的难易程度以及模型的泛化能力，最终选择 50 种不同的类别对模型进行训练和测试，每个类别中使用 15 张图片做为训练集，5 张图片做为测试集。

3.2 模型结构

在本小节中尝试对模型结构进行更改，观察是否对最终模型的效果产生影响。

首先对于全连接网络 MLP，去掉其中一个隐藏层，得到测试集上分类的准确率为 0.484；将原始的全连接网络先扩充特征，再压缩特征改为从输入开始逐层压缩的如下结构：

```
    MLP(
  (linear_layer): Sequential(
    (0): Linear(in_features=784, out_features=512)
    ...
    (4): Linear(in_features=512, out_features=256)
    ...
    (8): Linear(in_features=256, out_features=50)
  )
)
```

得到最终测试集上的分类准确率为 0.444。相比之前 0.512 的分类准确率可以得出结论，先对输入特征进行扩充，再对特征进行压缩的结构有利于提高模型的效果，且加深 MLP 的深度有助于提高预测的准确率。

对于卷积神经网络 CNN 不做过于复杂的更改，尝试去掉第二个 64 通道的卷积块，只保留第一个卷积块，得到最后的分类准确率为 0.788，效果变差。因为底层的卷积层用于筛选图片各个局部的特征，上层的卷积层用于将这些局部特征进行聚合，学习更上层的全局特征。只保留第一个卷积块可能会导致网络最终只学到了图片的局部特征，缺少了部分对分类很重要的聚合全局图像特征，所以最终的分类准确率降低。

3.3 超参数

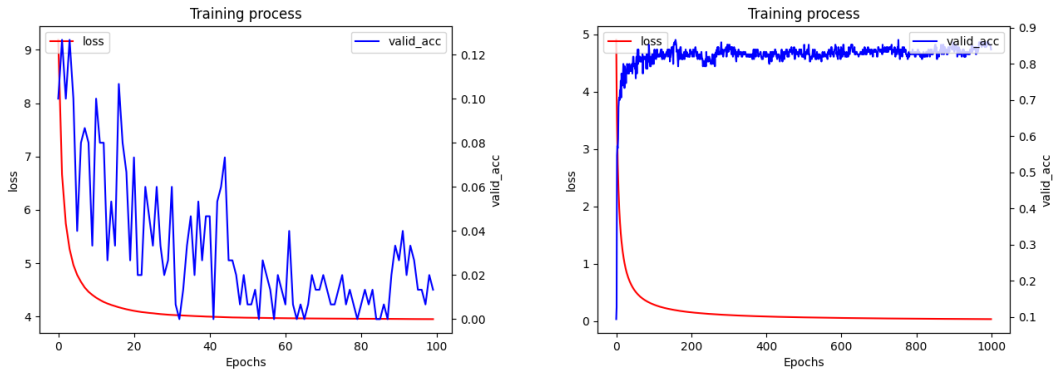
使用效果较好的卷积神经网络进行超参数的调参。首先测试批量大小 (batch size) 和学习率 (leraning rate) 对模型效果的影响，固定训练轮数为 100，尝试不同的批量大小与学习率，得到模

型最终的分类准确率如表 2：

Table 2: 不同超参数下的结果对比

bsz/lr	1	0.1	0.01	1e-3
16	0.032	0.852	0.816	0.704
32	0.752	**0.856**	0.784	0.576
64	0.804	0.852	0.732	0.488

从结果中可以看出，最终效果最好的超参数组合为 $batch\_size = 32, lr = 0.1$。对于批量大小 (batch_size) 来说，较小的批量大小可以节省内存和计算资源，但是会增大单轮的训练时间，同时小批量包含的数据较少，得到梯度的偏度和随机性更大，梯度下降过程中的稳定性更差。以表2第一个数据点的特殊现象为例，小批量配合一个大的学习率导致模型在梯度更新优化的过程中完全走向了一个错误的方向，导致模型最后的表现极差，训练过程中验证集的准确率如图2(a)展示不升高反而逐渐下降。当批量大小增大时，可以有效利用训练的数据并行提高训练的效率，但是会给内存和计算资源带来更大的压力；同时增大批量大小可以有效提高模型在训练过程中的稳定性，但过大的批量也会导致模型在训练过程中出现过拟合，降低模型的泛化能力，所以选择一个合适的批量大小进行训练是十分重要的。

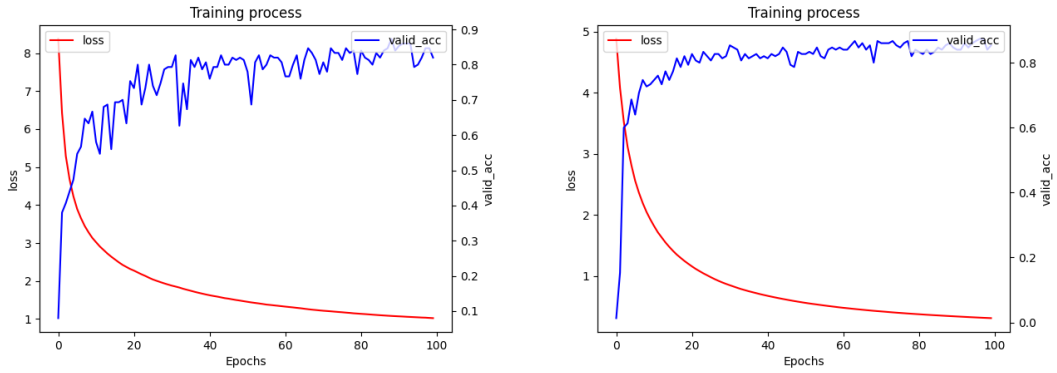


(a) batch_size=16,lr=1　　(b) epoch=1000

Figure 2: Training Process

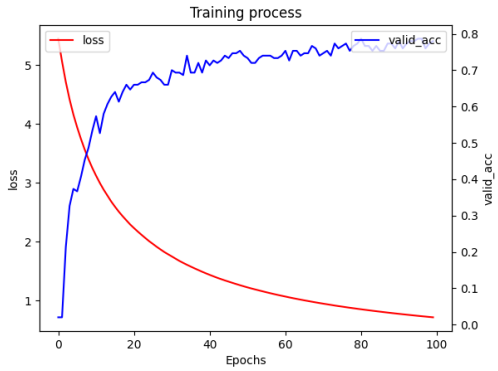
学习率决定了模型梯度在反向传播过程中优化的步长，lr 过小时，会导致模型梯度每次更新的幅度小，使得模型收敛过慢，需要很长时间才能取得一个可以接受的效果，且容易陷入局部最优当中；lr 过大时，会导致梯度更新幅度过大，使得模型在训练过程中不停震荡，收敛困难，且在优化过程中很难达到一个小临域中的局部最优值，往往模型不能达到很好的表现。图3展示了 $batch\_size = 32$ 时不同学习率下的训练过程，实验中模型分类准确率随学习率的变化趋势也说明了这一点。

对于迭代次数，测试了训练 1000 个 epoch 模型在训练过程中的表现，如图2(b)所示，可以看出，训练轮数在达到一定数值后模型在验证集上的表现就会出现比较明显的波动，出现验证集上准确率下降的过拟合现象，对于这种情况最终选取在验证集上表现最好的模型作为最终的模型结果。

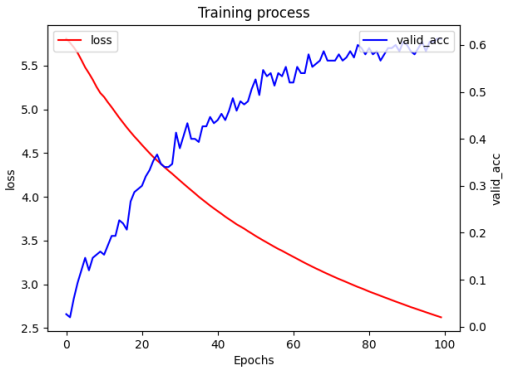


(a) lr=1

(b) lr=0.1



(c) lr=0.01



(d) lr=0.001

Figure 3: 不同学习率下的训练过程

综合以上因素考虑模型的泛化能力，训练速度以及收敛过程中的稳定性，最终选取批量大小 $batch\_size = 32$，学习率 $lr = 0.1$，训练轮数为 200，取训练过程中在验证集上表现最好的模型作为最终结果。

3.4 深度学习技巧

在这一部分中对模型中的 `dropout` 和 `btach_normalization` 进行调整，讨论这两个因素对与模型效果的影响。在 MLP 和 CNN 两个模型中分别加入 BN 层，同时调整 Dropout 层的概率，得到的分类准确率结果如表3所示：

Table 3: 不同设置下的模型效果对比

	MLP(normal)	MLP(BN)	CNN(normal)	CNN(BN)
No Dropout	0.484	0.452	0.648	0.772
Dropout(0.1)	0.460	0.456	0.684	0.756
Dropout(0.5)	0.492	0.476	0.728	0.820
Dropout(0.8)	0.484	0.512	0.776	0.868

可以看出，对于同一个模型，Dropout 的概率越大，模型在测试集上的分类准确率越高，因为 Dropout 可以有效地增强模型的泛化能力，能够一定程度上避免模型在训练集上过拟合，所以对于两种模型都在一定程度上提升了在测试集上的准确率表现。

在同一 Dropout 的概率下，BN 层全连接神经网络的作用不是很明显，但对于卷积神经网络来说，加入 Batch Normalization 有效地提高了网络最终的分类准确率，因为 BN 对每个 batch 进行标准化，相当于对每个样本都进行了微调，可以降低模型对某些特殊偏度样本的依赖，减少这些样本的影响，同时 BN 相当于在训练过程中增加了噪声，能够有效防止模型过拟合，提升模型的泛化能力，故提升了模型在测试集上的表现。

所以最终对于 MLP 和 CNN 选取 dropout 值为 0.8，且两个模型中均加入 BN 层。

4 思考及讨论

在完成这次任务的过程中，我体会到了在实际面对数据处理问题时，除模型结构外很多的特征工程和调试模型的技巧也很重要。

首先是对于模型的输入特征，在最开始使用原始 `utils.py` 读取图片特征时，模型训练的准确率非常低，在进行图像特征二值化增强后，效果得到了明显的提升，所以对于模型的输入，区分和增强不同的特征对结果是很重要的。另外便是模型调试的过程，包括超参数、Batch Normalization 以及 Dropout 概率，这些因素往往也会使相同的模型最终的结果有比较大的差异，需要大量的经验和尝试。

对于模型来说便是要选择符合场景及输入特征的模型，比如在这次二维图片任务的场景下选择卷积神经网络会有更好的结果，一般来说选择正确的模型类别，不需要过于复杂的结构也能够取得比较好的结果。