Building a neural network model for classification problem

xxx[a], 马雯芯[a], xxx[b], xxx[c]

[a] 生物医学工程学院
[b] 计算机学院
[c] 合肥物质科学研究院

Dr.Wenjie Ruan

Contents

1. Introduction

1.1. Task

本项目旨在构建 MLP 和 CNN 两种神经网络模型来解决 EMNIST 分类问题。项目链接：https://github.com/yuhangz/cv_task.git

1.2. Dataset

EMNIST 是一组手写字符数字，源自 NIST 特别数据库 19，所有数据被转换为 28x28 像素图像格式，数据集结构与 MNIST 数据集完全匹配。本次任务使用 EMNIST 数据集中的"Balanced" 子集，分为训练集（大小为 112800）和测试集（大小为 18800）。可视化的前 20 个数据如下，其中图像标题为对应的 label：

1.3. Model Design

本次模型搭建通过 Pytorch 实现，具体的结构及设计理由如下：

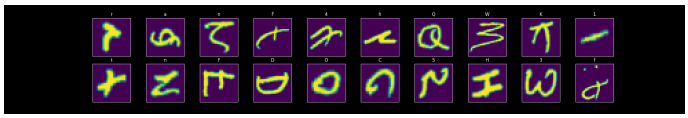

Figure 1. 前 20 个数据

1.3.1. CNN

我们的 CNN 模型中包含三个卷积块和最后的 47 类 Linear 分类层，其中一个卷积块包含一层卷积层、一层激活层、一层池化层和可选的归一化层及 Dropout 层。三层卷积层分别完成 $1 \to 256$、$256 \to 128$、$128 \to 64$ 的维度转变，卷积核大小为 3，步距为 2。通过对图像模式的先放大后压缩，我们充分提取到了对分类最为关键的特征。

1.3.2. MLP

在 MLP 的设计过程中，我们采用了相似的思想，三层 Linear 层分别完成 $784, \to 512$、$512 \to 256$、$256 \to 128$ 的维度转变，最后通过 $128 \to 47$ 的分类层完成分类。因为图像特征较为简单，我们通过不断减少特征维度完成了对关键特征的筛选。

1.4. Metrics

混淆矩阵（Confusion Matrix）是一种用于评估分类模型性能的工具，能够详细展示模型在分类任务中的表现。它以表格的形式显示模型的预测结果与实际结果之间的对比，通常用于二分类问题。

实际/预测	正类（P）	负类（N）
正类（P）	真正类 (TP)	假负类 (FN)
负类（N）	假正类 (FP)	真负类 (TN)

Table 1. 混淆矩阵

- **TP（真正类）**：模型正确预测为正类的样本数
- **FN（假负类）**：模型错误预测为负类的实际正类样本数
- **FP（假正类）**：模型错误预测为正类的实际负类样本数
- **TN（真负类）**：模型正确预测为负类的样本数

准确率（Accuracy）表示正确预测的样本数占总样本数的比例

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

精确率（Precision）是评价分类模型性能的一个重要指标，特别是在关注正类预测的准确性时。它定义为在所有被模型预测为正类的样本中，实际为正类的样本所占的比例。

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

召回率（Recall）是评价分类模型性能的另一个重要指标，尤其在关注模型检测正类样本的能力时。它定义为在所有实际为正类的样本中，被模型正确预测为正类的样本所占的比例

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 分数（F1 Score）是综合精确率（Precision）和召回率（Recall）的指标，特别适用于当精确率和召回率都同等重要的场合。F1 分数是精确率和召回率的调和平均数：

$$\text{F1 Score} = 2 \frac{\text{Precision} \quad \text{Recall}}{\text{Precision} + \text{Recall}}$$

1.5. 损失函数

交叉熵（Cross Entropy）是衡量两个概率分布之间差异性的一种方法，在机器学习中常用于评估模型的性能。对于离散型随机变量，交叉熵定义为：

$$H(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$$

其中，y 是真实的概率分布，$\hat{y}$ 是模型预测的概率分布，y_i 和 $\hat{y}_i$ 分别表示真实类别和预测类别 i 的概率。交叉熵越小，表示模型的预测结果越接近真实分布。

2. Hyperparameters

本次实验中，我们并没有过多调整超参数。考虑到实验数据集较小，我们选择的模型结构也较为简单，具体见 section 1.3。本次训练在 GeForce RTX 3090 上进行，训练共 30 个 epoch，batch size 为 128，初始学习率为 0.001。当使用 StepLR 时，每 10 步进行一次学习率调整，调整比例为 0.9（与 ExponentialLR 一致）。当使用 Dropout 时，其比例设置为 0.1。当使用正则化项时，其比例系数为 0.01。

3. Techniques

本次实验分析了一下训练技术对最终结果的影响（括号中为默认值）：

- 学习率调整方式（StepLR）
- 网络激活函数选择（ELU）
- 优化器的选择（Adagrad）
- 网络是否采用 Batch Normalization（True）
- 正则项的选择（L1）
- 网络中是否采用 Dropout（True）

在实验中，我们假设上述若干超参对于优化结果是正交的。我们在每次实验中只更改一个变量的值，依次选择分类准确率更高和损失函数值更小的方法改变。我们均采用 4 折交叉验证的结果作为交叉验证算法的度量，加粗行为最终最优结果。

3.1. 学习率调整方式

一般情况下，随着训练的进行，我们会逐渐减小学习率从而达到促使模型更好地收敛。我们研究了 StepLR 和 ExponentialLR 两种 scheduler 对模型训练效果的影响。

Table 2. CNN/MLP 结构下不同 scheduler 选择及其效果

Scheduler	Acc.(%)	Loss
StepLR	**100.00/100.00**	**13.2524/12.6537**
ExponentialLR	100.00/100.00	15.6297/15.0863

结果：采用不同学习率调整策略时，模型均能取得很好的预测效果。使用 StepLR 时损失函数收敛到更低值。

分析：ExponentialLR 调整策略中，随着训练进行，学习率呈指数级下降，以至于到训练后期学习率过小，每轮训练参数更新缓慢。

3.2. 网络激活函数选择

深度学习模型的不同层之间往往会加入不同的激活函数来增强模型的非线性性。由于 Sigmoid、tahn 等激活函数存在梯度消失的问题，它们在当前深度学习实践中往往被用于特定任务或特定层中。因此，在本次实验中，我们主要讨论当前更常用的 ReLU 函数、LeakyReLU 函数和 ELU 函数。

Table 3. CNN/MLP 结构下不同激活函数选择及其效果

激活函数	Acc.(%)	Loss
ReLU	100.00/100.00	13.8518/13.3259
LeakyReLU	100.00/100.00	13.9398/13.2725
ELU	**100.00/100.00**	**13.2524/12.6283**

结果：采用不同激活函数时，模型均能取得很好的分类效果。对负值有一定响应的 LeakyReLU 函数和 ELU 函数时，模型预测的损失更小。

分析：神经网络中间层为负数时，数据中仍然包含一定信息，ReLU 这种直接忽略负值的方法会造成一定的信息损失。考虑负值时，模型的表示能力得到增强，能够更好地处理复杂的任务和数据。ELU 函数与 LeakyReLU 函数相比，梯度中能包含更多信息，有助于优化算法的收敛性，使得模型能够更快地学习和收敛。

3.3. 优化器的选择

在反向传播更新参数的过程中，采用不同的更新方法，即不同的优化器时，模型会有不同的收敛效果。随机梯度下降法是当前几乎所有优化器的主要方法，但在具体实现时，不同优化器也引入了不同的信息（如二阶梯度信息、momentum 等）以便更好地优化。我们选择的优化器有：随机梯度下降（SGD）、平均随机梯度下降（ASGD）、包含二阶动量信息的 AdaGrad、进一步引入全部历史梯度的 RMSProp、同时考虑一二阶动量信息的 Adam。

Table 4. CNN/MLP 结构下不同优化器选择及其效果

优化器	Acc.(%)	Loss
SGD	100.00/100.00	16.5751/ 16.2426
ASGD	100.00/100.00	16.5755/16.2284
Adagrad	100.00/100.00	13.2618/14.0104
RMSProp	**100.00/100.00**	**4.2354/7.1439**
Adam	100.00/100.00	6.3006/12.6636

结果： 采用不同优化器时，最终的分类结果都很好。但不考虑历史梯度信息时（SGD、ASGD），最终损失函数值较大。使用指数移动平均更新历史梯度信息时（即使用 RMSProp 优化器），损失函数值最小。

分析： 由于不考虑历史梯度信息，SGD 容易收敛到局部最优，并且在某些情况下可能被困在鞍点。引入 momentum 项后，在梯度方向改变时，momentum 能够降低参数更新速度，从而减少震荡；在梯度方向相同时，momentum 可以加速参数更新，从而加速收敛。因此考虑历史梯度信息的其他优化器效果普遍更好。

3.4. 网络是否采用 Batch Normalization

神经网络希望输入的数据能够满足独立同分布性，并且希望数据能分布在激活函数梯度较大的范围之内。可是随着网络的加深，底层的参数更新会对高层的输入分布产生很大的影响，也就是说即使一开始我们的数据满足了要求，可是随着网络的传递，数据也会发生变化，不能满足要求。我们需要在网络中加入更多的归一化层来保证数据的独立同分布性。

当前常用的 Normalization 方法包括 Batch Normalization（BN）、Layer Normalization（LN）、Group Normalization（GN）等。由于我们的 batch size 值较大，我们只验证了 Batch Normalization 的有效性。

Table 5. CNN/MLP 结构下使用 (w/-) 及不使用 (w/o)BN 时的分类效果及损失函数值

BN	Acc.(%)	Loss
w/-	100.00/100.00	4.2254/7.1604
w/o	**100.00/100.00**	**4.0334/4.0596**

结果： BN 对分类效果影响不大，模型均能取得很好的分类效果。不使用 BN 时损失函数收敛到更小的值。

分析： 由于本次实验采用的数据集分布简单且模型结构简单，我们推测在没有 BN 时模型已经能够较好地完成预测。加入 BN 后，算法有了更强的约束，对数据集的拟合能力进而被限制。

3.5. 正则项的选择

深度学习中，模型的复杂度往往很高。为了反正模型在数据集上出现过拟合现象，损失函数中可以加上对于参数的约束，即正则项。常用的正则项有 L1、L2 两种。L1 会使参数趋于稀疏化，L2 会使参数的值尽可能小。我们探究了这两种正则化对本实验的影响。

Table 6. CNN/MLP 结构下使用 L1,L2 及不使用 (None) 正则化时的分类效果及损失函数值

Norm	Acc.(%)	Loss
L1	100.00/100.00	4.0357/4.0611
L2	100.00/100.00	3.0192/3.0089
None	**100.00/100.00**	**2.0184/2.4837**

结果： 使用及不使用正则化项对最终的分类结果影响不大，模型并不呈现过拟合状态，均能取得较好结果。由于正则项值大于 0，引入正则项后，损失函数值均有增加。

分析： 由于我们的交叉验证是将训练集拆分后进行，数据集较小且分布较为简单，我们的模型在分类任务上均表现很好，没有呈现明显的过拟合趋势。考虑到正则项的引入会对优化造成更重的负担，我们并未在损失函数中正则化项。

3.6. 网络中是否采用 Dropout

当一个复杂的前馈神经网络被训练在小的数据集时，容易造成过拟合。为了防止过拟合，可以通过阻止特征检测器的共同作用来提高神经网络的性能。Dropout 是一种避免过拟合的方式，它是指在每个 epoch 中，模型按一定比例随机忽略一部分神经元以减少隐层节点间的相互依赖。在我们的实验中，我们对比了不采用 Dropout、采用比例为 0.1 的 dropout 和比例为 0.5 的 dropout。

Table 7. CNN/MLP 结构下使用不同比例的 dropout 时的分类效果及损失函数值

Dropout rate	Acc.(%)	Loss
0.5	100.00/100.00	3.2241/2.9035
0.1	100.00/100.00	2.1592/2.4584
0	**100.00/100.00**	**1.8041/2.4417**

结果： 模型最终的分类效果很好，受 dropout 影响小。由于 dropout 在模型训练中引入了随机性，模型的损失函数值有所增加。

分析： dropout 对训练的帮助与正则化类似，主要是起到防止过拟合的作用。在本次实验中，我们采用拆分训练集的方式进行交叉验证，在四轮验证中我们的模型并没有呈现明显的过拟合趋势，因此我们没有引入 dropout。

3.7. 总结

对于 CNN 和 MLP 的训练，我们最终采用了如下的训练方案：

- 学习率调整策略 x: StepLR
- 激活函数: ELU
- 优化器: RMSprop
- 是否采用 BN: False
- 是否采用正则化: False
- Dropout rate: 0

4. Overfitting Discussion

我们根据上述实验进行训练，过程分别如 fig. 2,fig. 3所示，随着训练的进行，训练损失在不断下降，且测试表现没有明显下降。我们认为在此数据集上我们的模型没有明显的过拟合趋势。

但是模型在前述预实验中，训练集上分类准确率呈现饱和状态，而测试集准确率低于 90%，我们认为模型的泛化性能较差。可以推测训练数据的分布较为集中，且与测试集差异较大。

5. Results

5.1. Training & Testing

CNN 训练的 loss 和 acc 曲线如图 fig. 2所示，MLP 训练的 loss 和 acc 曲线如图 fig. 3所示。随着训练迭代次数的增加，训练损失逐渐减小，接近于 0。这表示模型在训练数据上的拟合效果逐渐

提升。随着训练迭代次数的增加，测试集准确率呈现先上升后波动的趋势。CNN 模型在训练过程中表现稳定，损失较低且准确率较高，基本保持在 0.87 左右。MLP 模型的训练损失虽然也逐渐下降，但在某些阶段的波动较大，准确率保持在 0.83 左右。

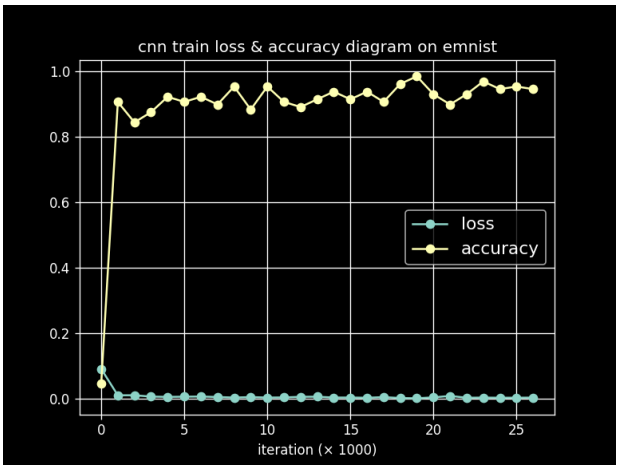

Figure 2. CNN 的训练过程

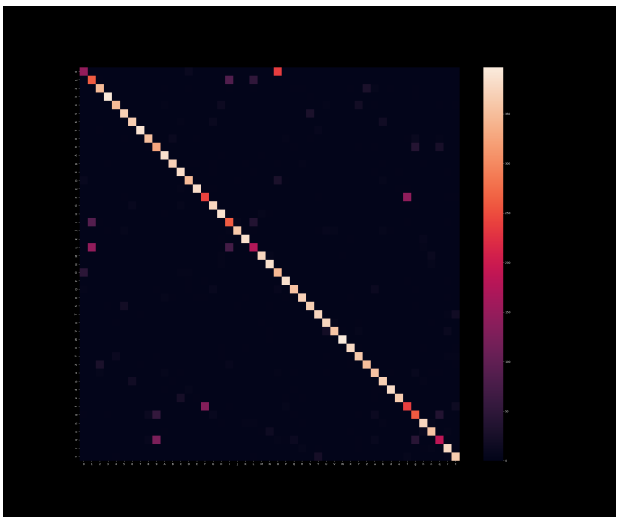

Figure 4. CNN 混淆矩阵

5.2. Results

CNN 与 MLP 的（Micro/Macro）精确率、召回率、F1 分数和准确率如 table 8。综合结果可见，CNN 的结果整体优于 MLP，可见 CNN 在图像任务中的优越性。

模型	精确率		召回率		F1 分数		准确率
	Micro	Macro	Micro	Macro	Micro	Macro	
MLP	0.84	0.84	0.84	0.83	0.83	0.83	0.83
CNN	0.86	0.86	0.86	0.86	0.86	0.86	0.86

Table 8. 模型性能指标比较

我们可视化了 CNN 与 MLP 的预测混淆矩阵，如 fig. 4和 fig. 5所示。CNN 模型的混淆矩阵显示，模型在绝大部分类别上的预测都比较准确，只有少数错误分类。MLP 模型的混淆矩阵显示，相对于 CNN 模型，MLP 模型在某些类别上的预测错误更多，且误差分布较为分散。

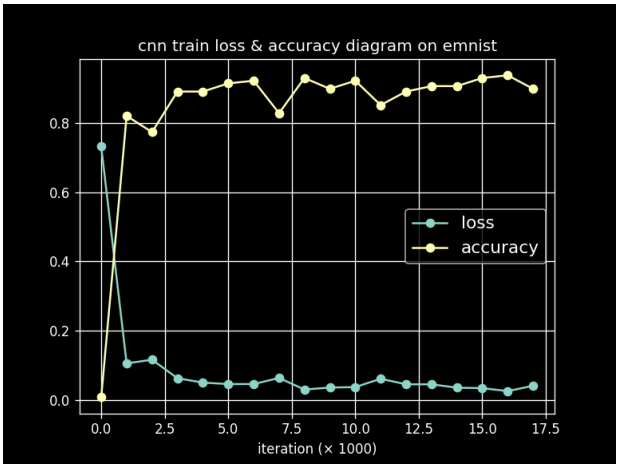

Figure 3. MLP 的训练过程

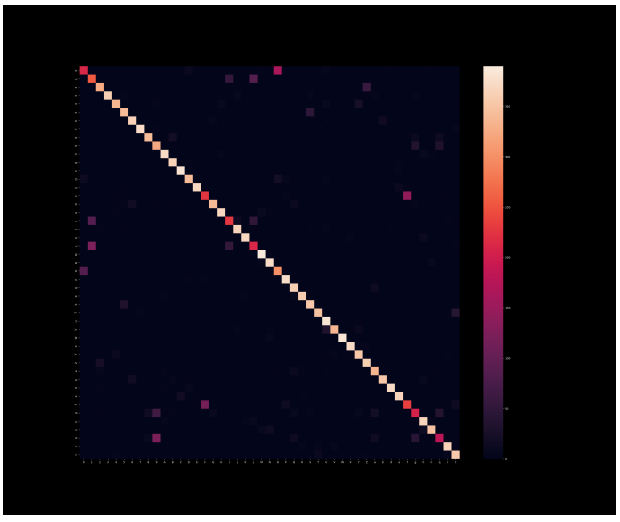

Figure 5. MLP 混淆矩阵

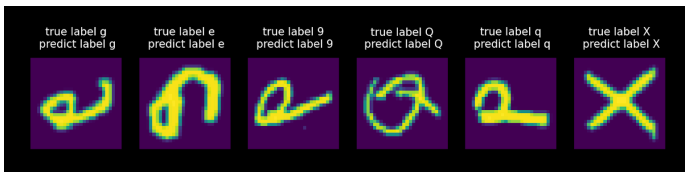

Figure 6. CNN 预测结果

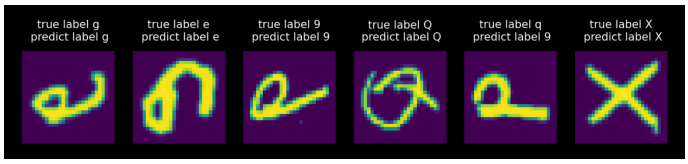

Figure 7. MLP 预测结果

5.3. Visualization

我们展示了前 6 个样本在 CNN 和 MLP 中的预测结果，如 fig. 6和 fig. 7所示。

5.4. Analysis

CNN（卷积神经网络）在图像任务中优于 MLP（多层感知器）的原因主要有以下几点：

- 局部连接和权值共享：CNN 利用局部连接和权值共享的特性，能够更好地处理图像数据的空间结构。由于图像中的相邻像素之间存在相关性，CNN 可以通过局部连接和权值共享来提取图像的局部特征，从而有效地降低了模型的参数量和计算复杂度。
- 卷积和池化层：CNN 通过卷积层和池化层的组合，能够逐渐提取图像的抽象特征。卷积层通过滤波器对图像进行特征提取，而池化层则能够降低特征图的维度，提高模型的计算效率。这种层次化的特征提取方式使得 CNN 能够更好地捕获图像中的特征信息。
- 平移不变性：CNN 具有平移不变性的特性，即当图像中的对象发生平移时，CNN 能够仍能够正确识别对象。这是因为卷积操作在提取特征时会同时考虑到图像中的局部位置信息，而不是依赖于全局位置信息。
- 自动学习特征表示：与 MLP 不同，CNN 能够通过反向传播算法自动学习特征表示，无需手动设计特征提取器。这使得 CNN 更适用于复杂的图像任务，能够学习到更具有区分性的特征表示，从而提高了模型的泛化能力和性能。

6. Conclusion

通过这个项目，我们团队学到了很多关于构建和优化神经网络的知识和技能：首先在深度学习模型的设计方面，我们学会了如何设计和实现多层感知器（MLP）和卷积神经网络（CNN）。学会了根据任务需求选择适当的网络结构，包括隐藏层的数量和每层的神经元数量。其次在超参数优化方面，学习了如何通过交叉验证和超参数搜索来优化模型性能。掌握了自适应学习率调度、激活函数选择、优化器选择、批标准化、正则化和 Dropout 等技术。模型评估和分析方面，掌握了使用准确率、精确率、召回率和 F1 分数等评估指标来评估模型性能。学会了使用混淆矩阵来分析模型的分类效果。

通过比较这两种模型，我们得出了以下几点总结与思考：

- CNN 在图像分类任务中表现出色，能够利用局部连接、权值共享和卷积操作来提取图像的特征，从而实现更高的分类精度。MLP 在一些简单的数据集上也能够取得不错的效果，但对于复杂的图像数据集，其性能往往不如 CNN。
- 我们采用了一些常见的技巧，如学习率衰减、批量归一化和数据增强等，以提高模型的收敛速度和泛化能力。
- 为了防止过拟合，我们采用了正则化技术，如 L2 正则化和 dropout 等，以减少模型的复杂度和提高泛化能力。
- 在未来的研究中，我们可以进一步探索 CNN 和 MLP 结合使用的方法，利用 CNN 提取图像特征，然后将特征输入到 MLP 中进行分类，以进一步提高模型的性能。同时，我们也可以尝试更先进的模型架构和优化算法，如深度残差网络 (ResNet)，以进一步提高模型的性能和稳定性。

综上所述，我们通过对 CNN 和 MLP 的研究与比较，深入了解了它们的特点、训练技巧和防止过拟合的机制，为未来的深度学习研究和应用提供了有益的启示和思考。