1 任务要求

设计一个前馈神经网络，对一组数据实现分类任务。

下载“dataset.csv”数据集，其中包含四类二维高斯数据和它们的标签。设计至少含有一层隐藏层的前馈神经网络来预测二维高斯样本$(\text{data}_1, \text{data}_2)$所属的分类label。这个数据集需要先进行随机排序，然后选取 90%用于训练，剩下的 10%用于测试。

2 任务设计

2.1 数据集准备

使用 pandas 库读取.csv 文件。获取到的是一个矩阵对象，由于数据集给的分类结果是 1~4，神经网络输出应为 0~3，将标签列减 1，然后打乱，取出 90%作为训练数据，10%作为测试数据。

```
# 读取CSV 文件
data_frame = pd.read_csv('./dataset.csv')
matric = data_frame.values
matric[:,-1] -= 1
np.random.shuffle(matric)
# print(matric)
# 提取特征和标签列
train_data = matric[:3600]
test_data = matric[3600:]
```

然后将矩阵通过 DataLoader 转化即可。

2.2 网络结构

由于数据是四类 2 维高斯数据的分类任务，所以输入 2 个数据，输出 4 维张量。该实验采取较为简单的网络结构。网络均由全连接层组成，后文将尝试不同层次及不同神经元的组合，此处给出一个简单的结构。如图 1 所示。

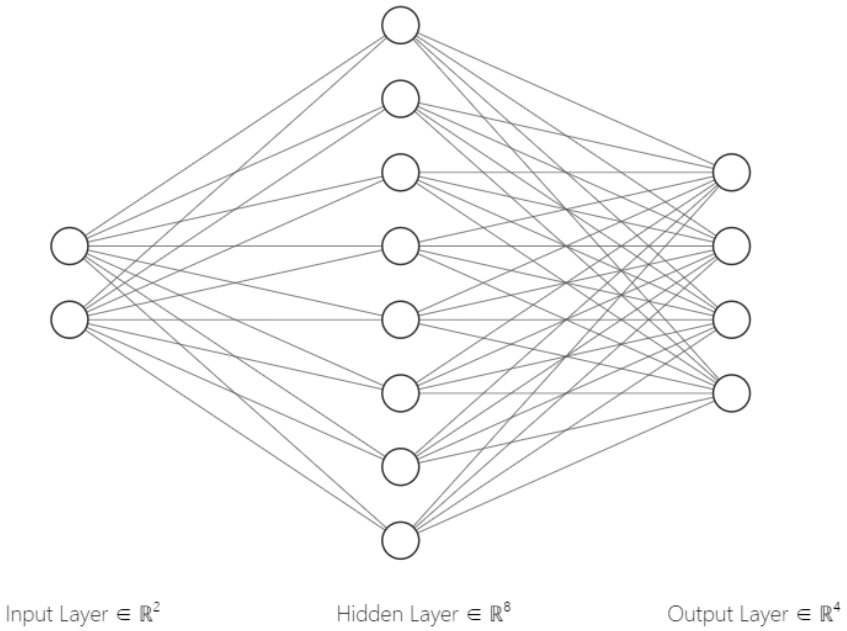


图 1 全连接神经网络

2.3 架构

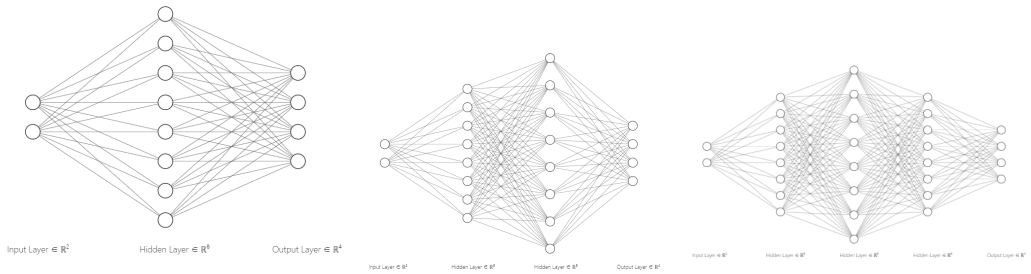
总体采用 pytorch 架构。

3 实验

本实验尝试不同的网络层数、不同的神经元个数、使用不同的激活函数等，观察网络性能。

3.1 网络层数

控制其他参数不变，改变层数观察收敛情况。在以下三个不同层次的模型中均采用：学习率=0.0015，batchsize=400，激活函数=Relu()。每层神经元数均设置为 8。三个神经网络均采用全连接模式，模型如下图所示，分别为 model1、model2、model3。



下面给出训练集上 loss 和 accuracy 随 epoch 的变化图示，详细数据参见文件夹内“不同网络层数.txt”文件。注意 loss 是每一轮训练的 loss。不是单条数据的 loss。loss1 对应 model1，以此类推。

训练集上：

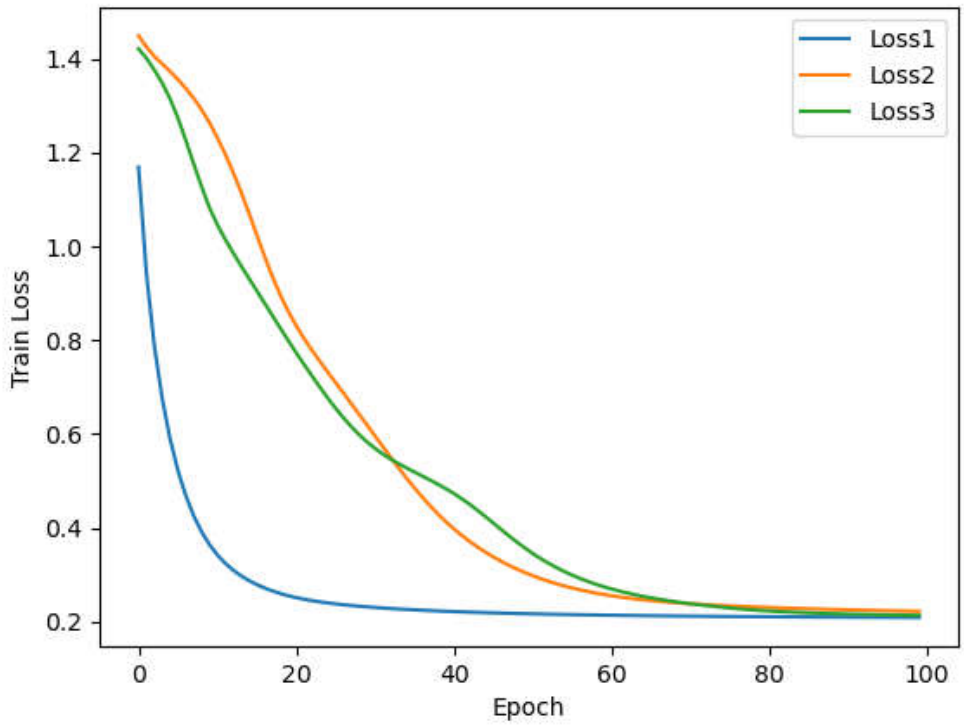


图 2 训练集 loss

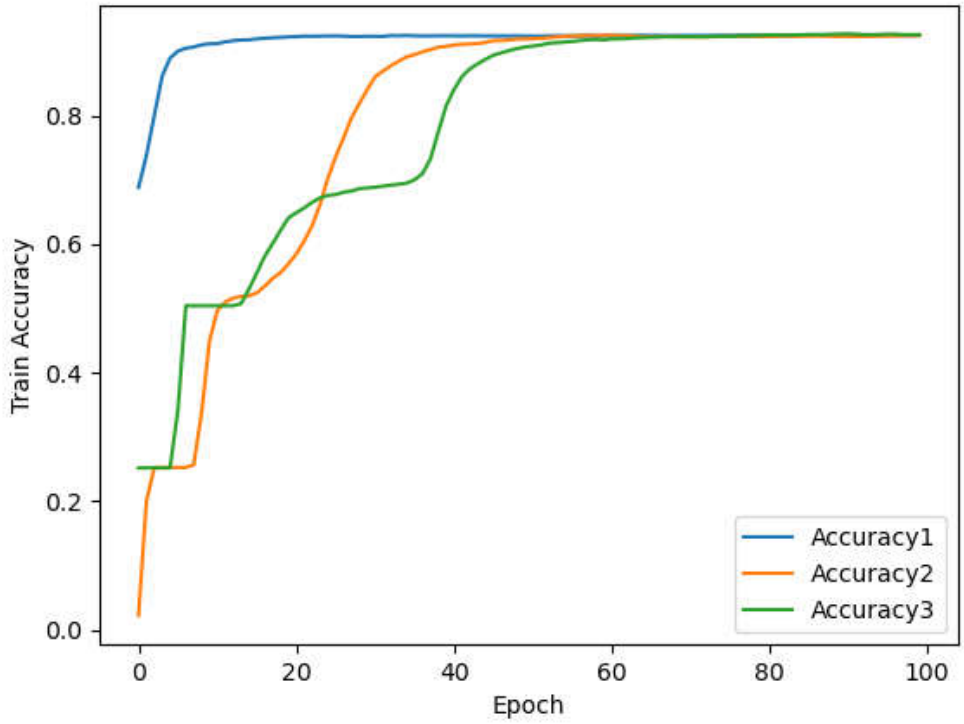


图 3 训练集 accuracy

测试集上：

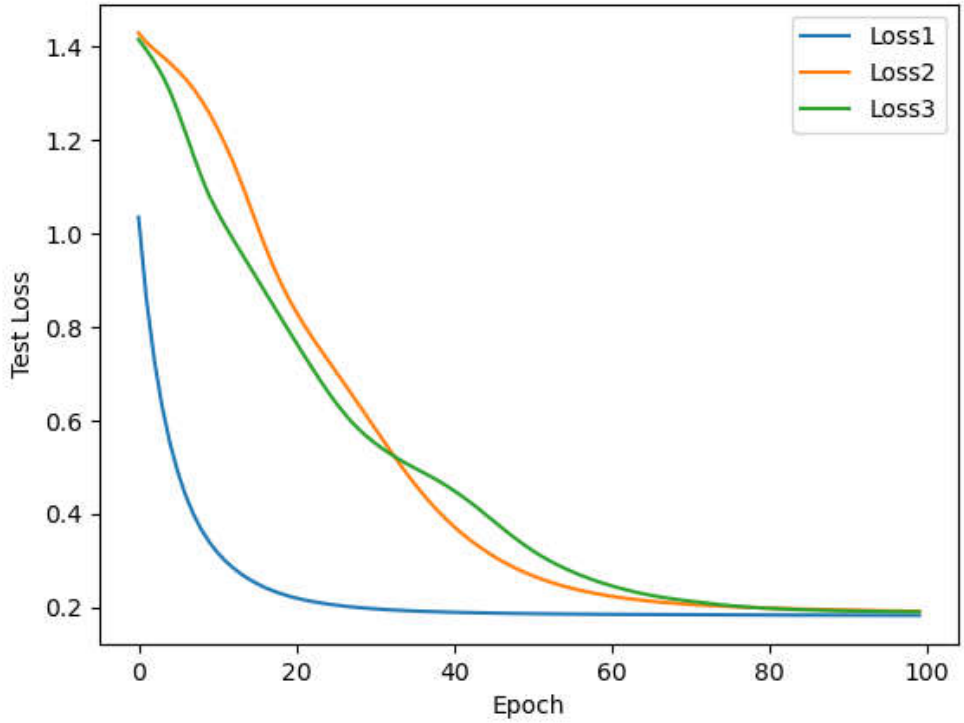


图 4 测试集 loss

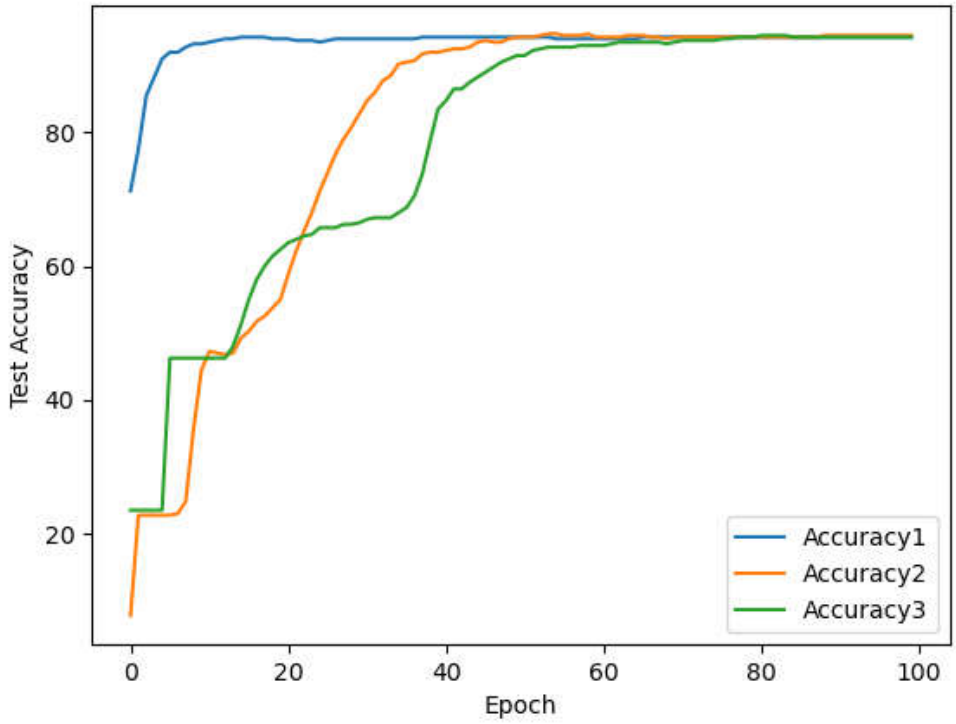


图 5 测试集 accuracy

最终 loss 和 accuracy 如下表所示。每一轮数据见附带文件。

	训练集		测试集	
	loss	accuracy	loss	accuracy
model1	0.209308	0.925556	0.183124	0.942500
model2	0.222393	0.924722	0.192026	0.945000
model3	0.213838	0.926111	0.190490	0.942500

有上述图像及数据可知，虽然不同层神经网络最后都收敛到差不多的水平，但是 model1 明显收敛更快。说明神经网络不是层数越多越好。

3.2 神经元个数

采用单层网络结构，仅改变中间层神经元数目。其他参数同 3.1 所述。

依然设计三个模型，分别是 model1、model2、model3.对应神经元个数为 8，32,128。

训练集上：

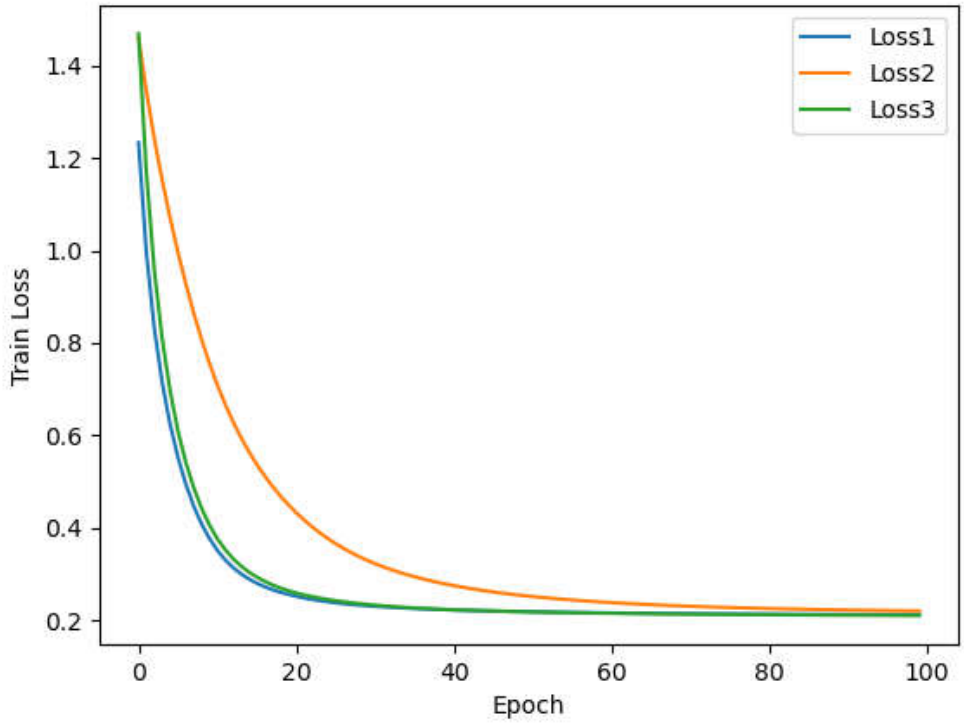


图 6 训练集 loss

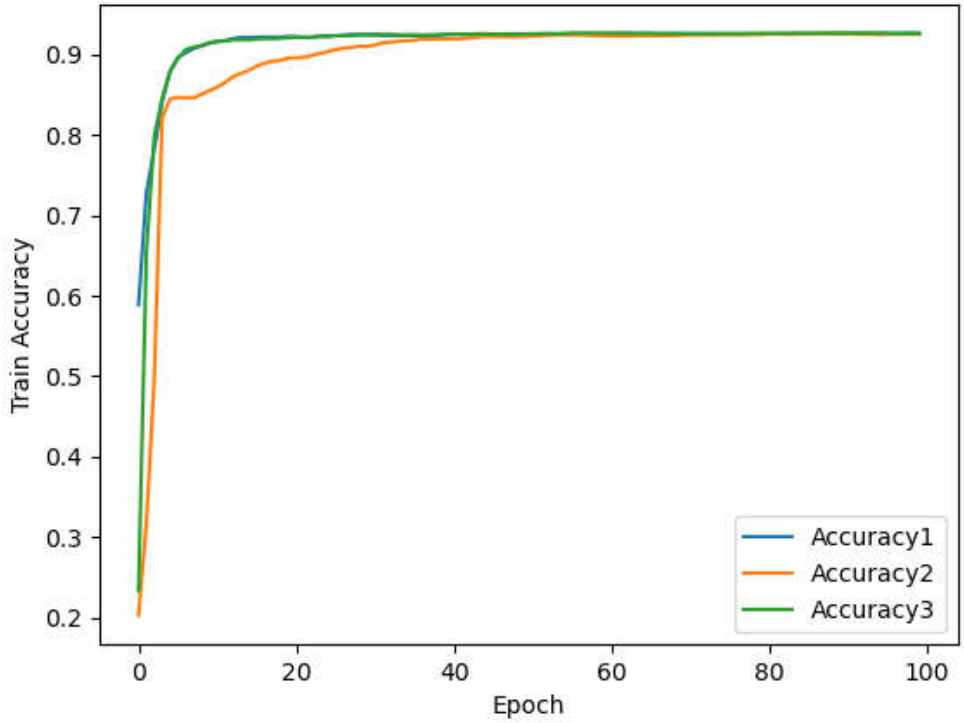


图 7 训练集 accuracy

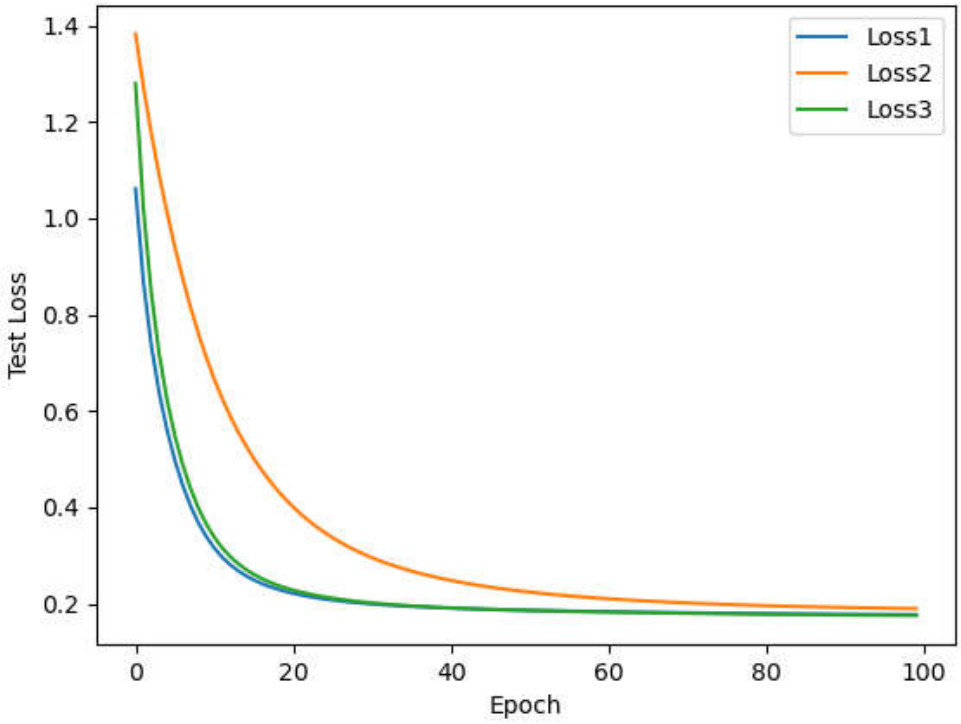


图 8 测试集 loss

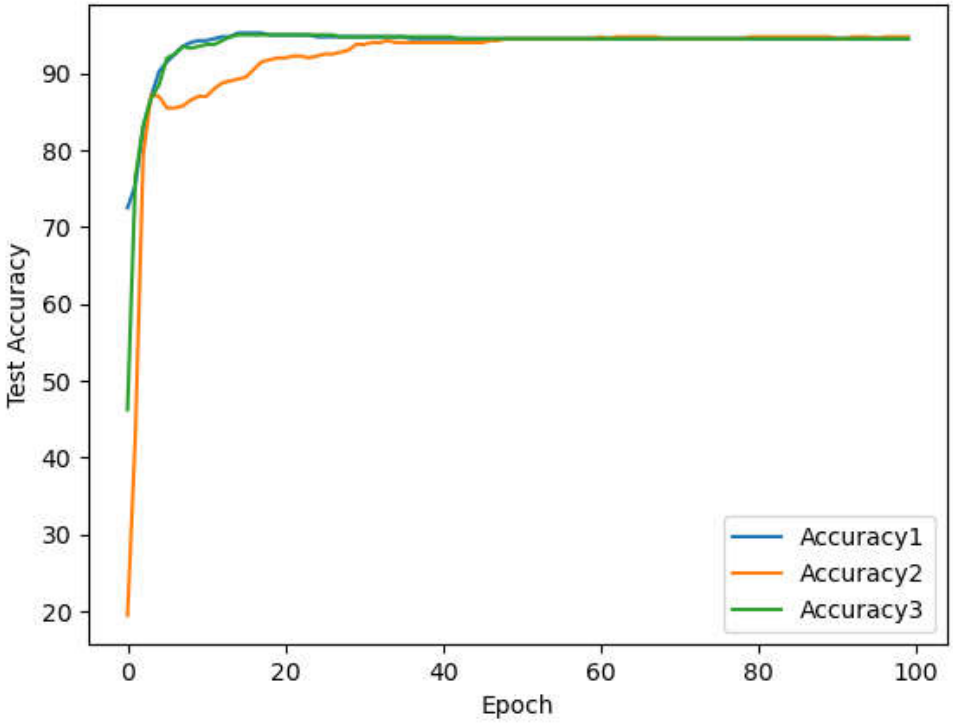


图 9 测试集 accuracy

最终准确率如下：

	训练集		测试集	
	loss	accuracy	loss	accuracy
model1	0.211472	0.926111	0.176955	0.945000
model2	0.219787	0.924722	0.189548	0.947500
model3	0.209980	0.925556	0.175238	0.945000

loss 和 accuracy 也收敛到大概相同的水平。但是从图像可以看出，神经元为 32 个时，模型表现显然没有神经元个数为 8 和 128 的模型好，而且神经元为 8 和 128 的模型表现差不多。说明，模型的识别性能并不与神经元个数成正比。

3.3 激活函数

采用单层 8 神经元网络结构，使用不同的激活函数。依然对比三种不同的激活函数。model1，model2，model3 分别对应 ReLu,Sigmoid,Tanh。

训练集上：

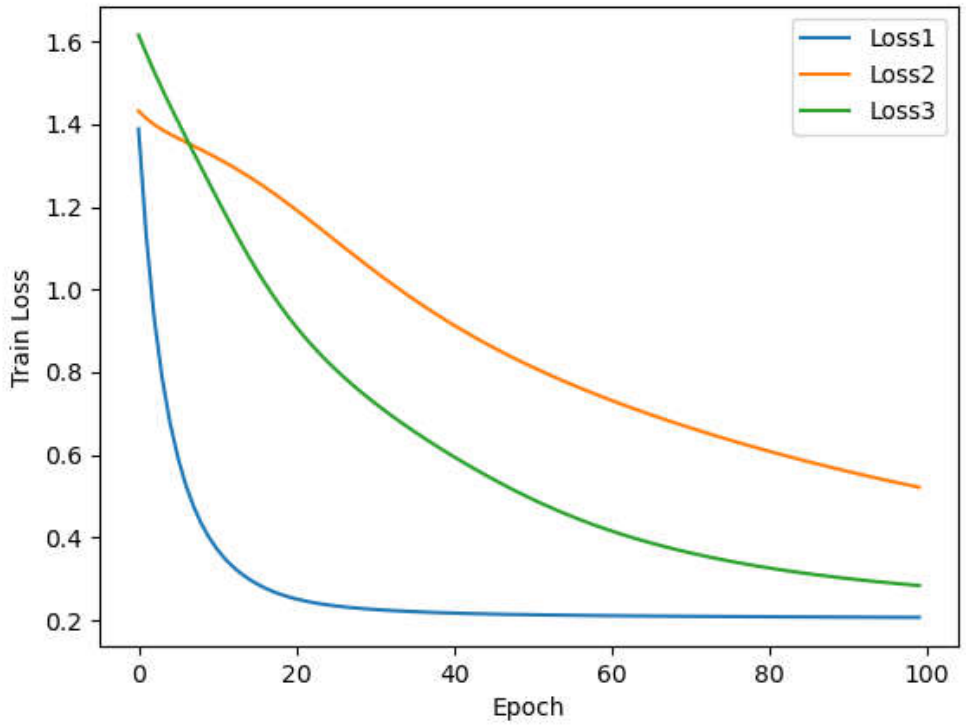


图 10 训练集 loss

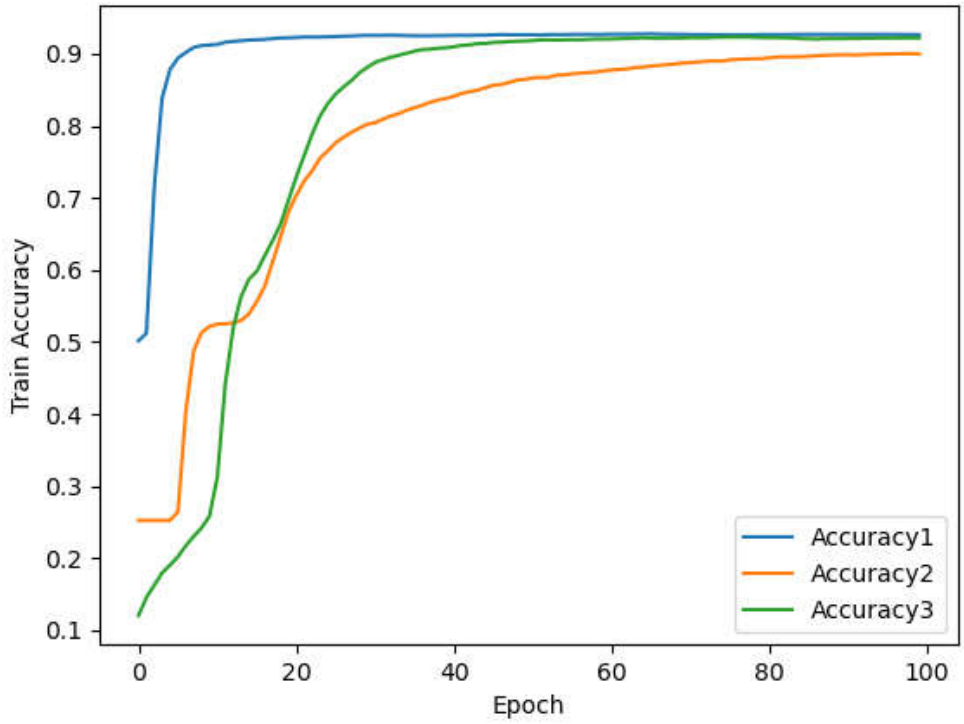


图 11 训练集 accuracy

测试集上：

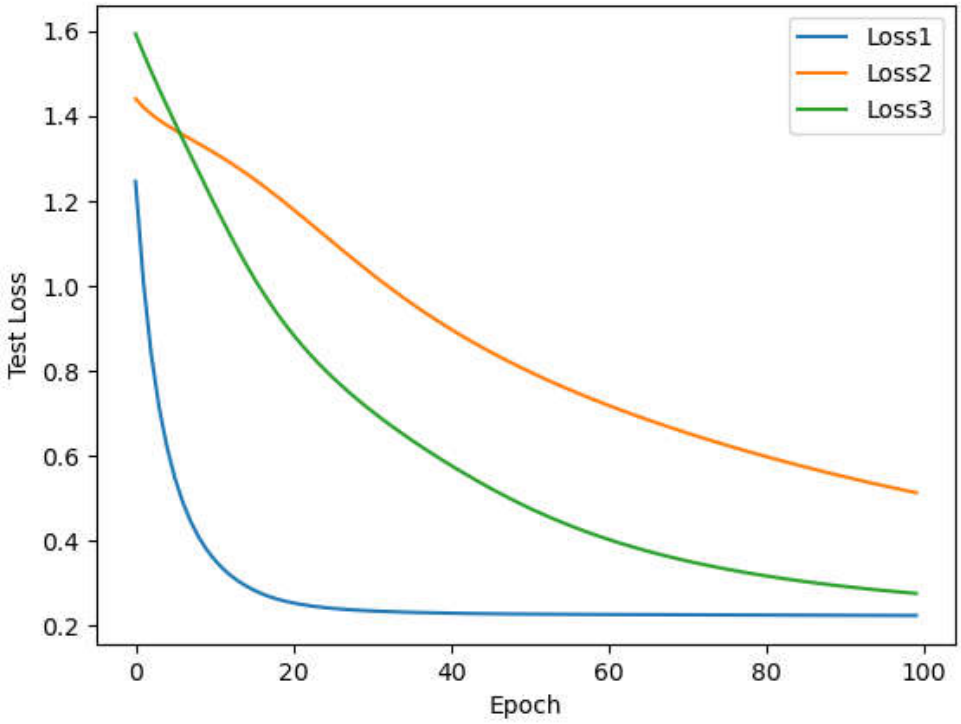


图 12 测试集 loss

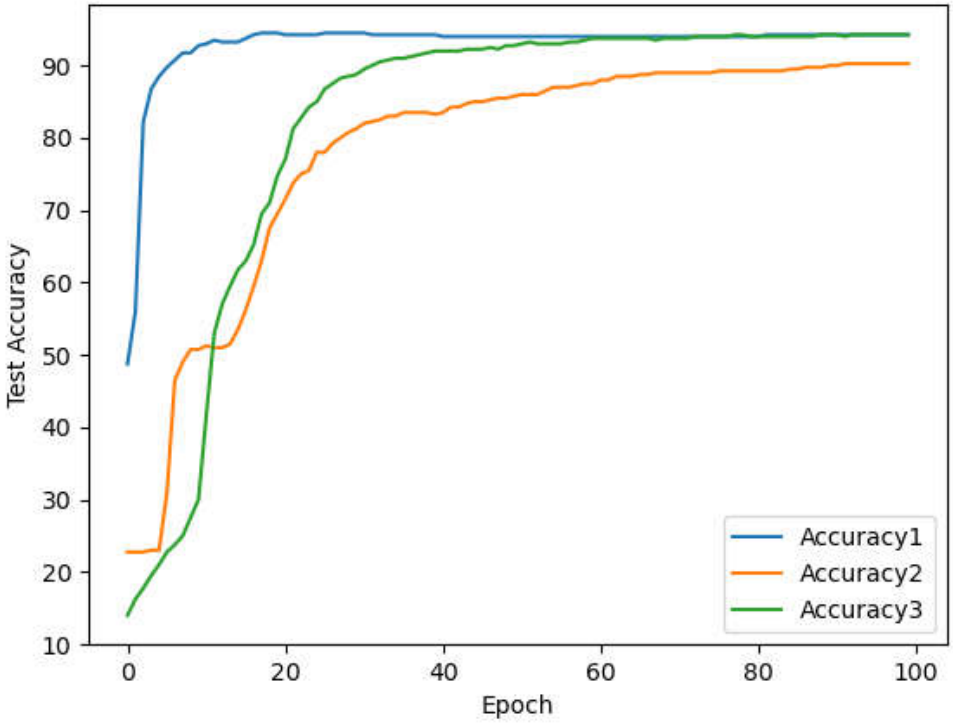


图 13 测试集 accuracy

最终准确率如下：

	训练集		测试集	
	loss	accuracy	loss	accuracy
ReLu	0.206964	0.926667	0.224355	0.942500
Sigmoid	0.521656	0.899722	0.513456	0.902500
Tanh	0.283988	0.921944	0.276381	0.942500

正确率上，ReLu 和 Tanh 最后均收敛到同一数值，不过 ReLu 收敛更为迅速。Sigmoid 在此数据集上各方面表现均不怎么样。总体而言，ReLu 性能最好。

4 结论

网络层数、神经元个数、激活函数等均会对模型的性能产生影响。同时，也尝试过更改学习率和 batch 大小。学习率较小时，模型收敛较慢，学习率较大时 loss 和 accuracy 曲线出现强烈震荡。batch 在训练集上好像越大越好，我将其设置为整个训练集数据大小，一次训练就收敛了。在测试集上就适中比较好。所以，构建一个性能比较好的神经网络需要不断地探索、不断地尝试。