深度学习实验一

1190200708 熊峰

2022 年 4 月 29 日

目录

1 环境配置

1.1 硬件配置

CPU : Intel(R) Xeon(R) Silver 4214
GPU : TITAN RTX 24G
MEM : 128G RAM

1.2 软件配置

OS : Ubuntu 20.04.1 LTS
PyTorch : Stable 1.11.0 CUDA 11.3
IDE : PyCharm 2021.3.2

2 代码编写

2.1 数据读取

使用 torchvision.datasets.MNIST 函数下载训练集和测试集，并使用 transform 对数据预处理。将数据正则化，并转化为张量。

在正式训练的时候使用 data.DataLoader 加载数据，并添加 batch_size。

2.2 搭建网络

本实验实现了基于 Pytorch 的 MLP 及 CNN 的模型。

本实验中 MLP 网络主要分为三层。

第一层: 将 28*28 维的向量映射为 512 维度的向量。

第二层: 将 512 维的向量映射为 512 维度的向量。

第三层: 将 512 维的向量映射为 10 维的向量。

本实验除了 MLP 网络，还搭建了 CNN 网络，CNN 网络主要分为三层。

第一层为卷积层，使用卷积核大小为 (4,4)，out_channel 设置为 64，用于识别不同的模式。

第二层为池化层，缩小参数矩阵。

第三层为全连接层，将数据映射为所需结果的维度。

2.3 定义优化器

优化器选择为 Adam 优化器。

2.4 定义损失函数并训练

本任务为多分类任务，因此将损失函数定义为交叉熵函数。

3 实验验证

3.1 实验相关设置

训练轮数: 100(若十轮以内验证集上没有优化，则结束训练)
训练设备: GPU
学习率: 1e-5

3.2 训练过程

实验使用 tensorboard 记录实验过程的数据。

3.2.1 MLP

在训练 12000 轮后，实验结果趋于收敛。

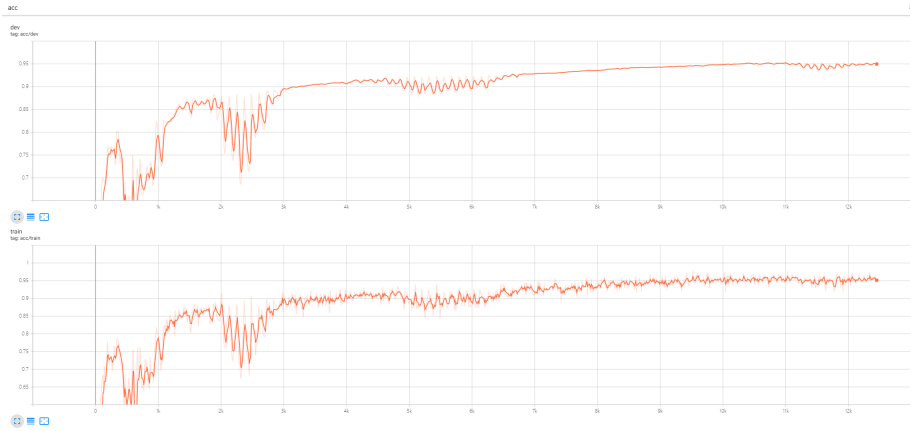

图 1: 准确率

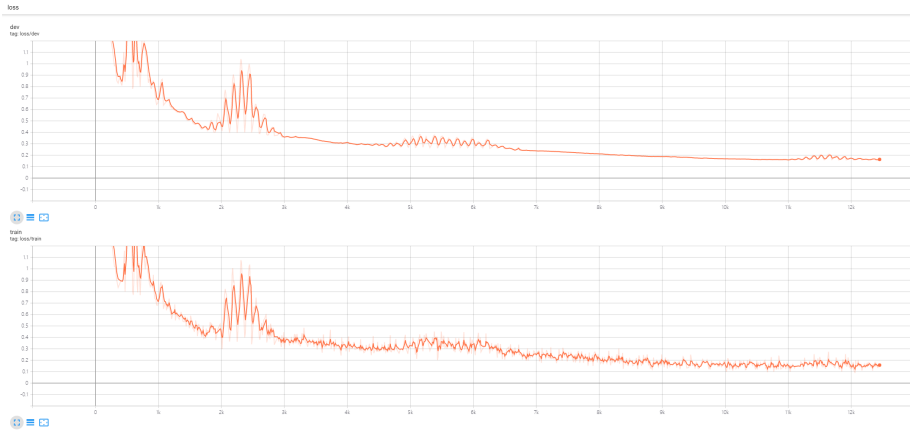

图 2: LOSS

3.2.2 CNN

在训练 8000 轮后，实验结果趋于收敛。

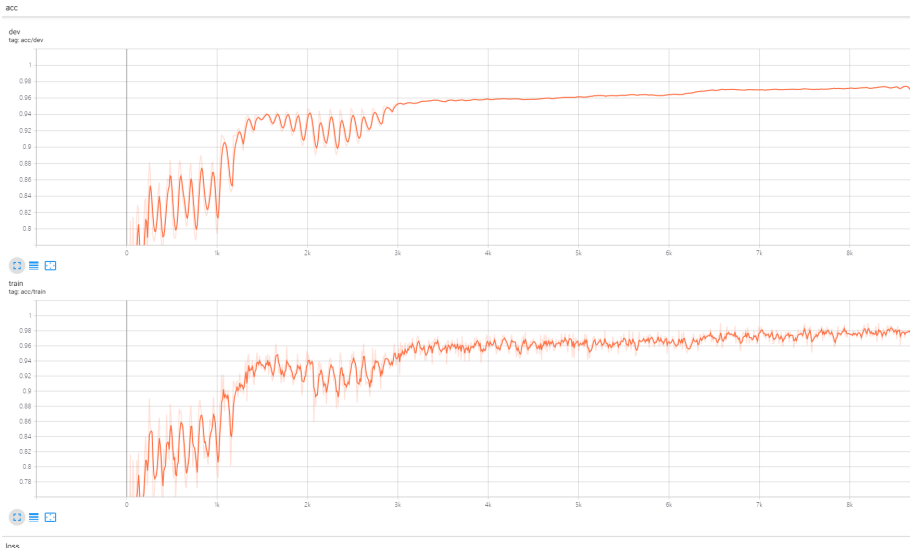

图 3: 准确率

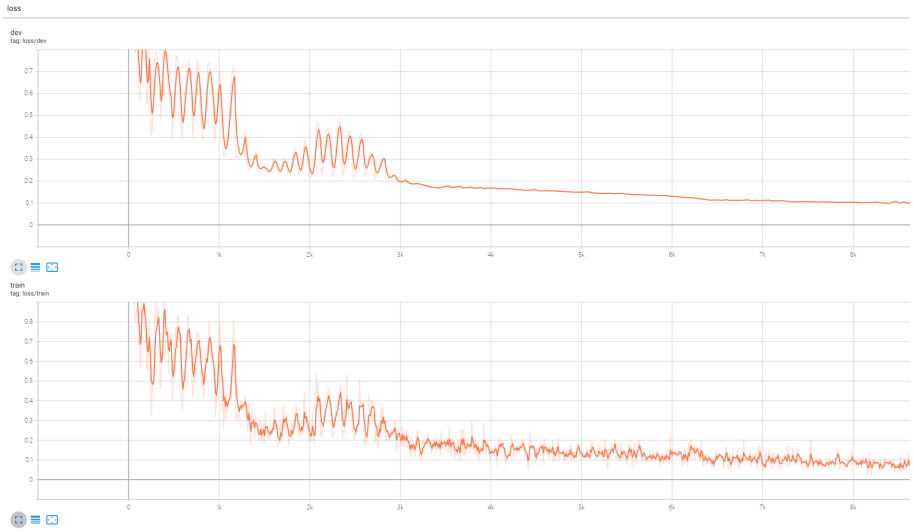

图 4: LOSS

3.3 实验结果

实验得到的 Confusion Matrix 如下图所示：

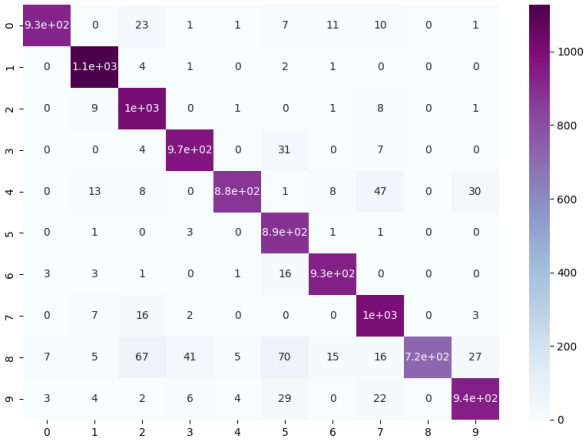

图 5: MLP Confusion Matrix

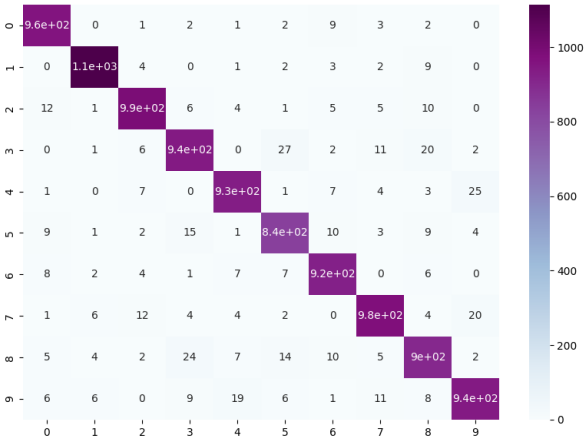

图 6: CNN Confusion Matrix

实验结果如下表所示:

模型	Acc	Micro F1
CNN	97.52%	0.9751
MLP	95.22%	0.9483

3.4 实验总结

通过动手实验 MLP 等模型，充实了理论的学习，收获颇丰。