深度学习实验二

1190200708 熊峰

2022 年 5 月 4 日

目录

1 环境配置 **3**
1.1 硬件配置 3
1.2 软件配置 3

2 代码编写 **3**
2.1 数据读取 3
2.2 搭建网络 3
2.2.1 AlexNet 3
2.2.2 VGG16 5
2.3 定义优化器 6
2.4 定义损失函数并训练 6

3 实验验证 **6**
3.1 实验相关设置 6
3.2 训练过程 6
3.2.1 AlexNet 7
3.2.2 VGG16 8
3.2.3 ResNet50 9
3.2.4 InceptionV3 10
3.2.5 DenseNet121 11
3.3 实验结果 12
3.4 实验总结 12

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

将数据集下载好后，使用 torchvision.transforms 对数据预处理，将图片转化为 Tensor，并正则化。

接下来对 torch.utils.data.Dataset 继承，并实现自己的数据集，在此步骤将数据的标签映射为数字，方便计算交叉熵损失函数。

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 AlexNet、VGG16、InceptionV3、ResNet50、DenseNet121 的模型。其中 AlexNet、VGG16 使用 PyTorch 搭建完整模型，其余网络主要加载 PyTorch 的预训练模型，并进一步训练。

2.2.1 AlexNet

AlexNet 网络结构如图所示。

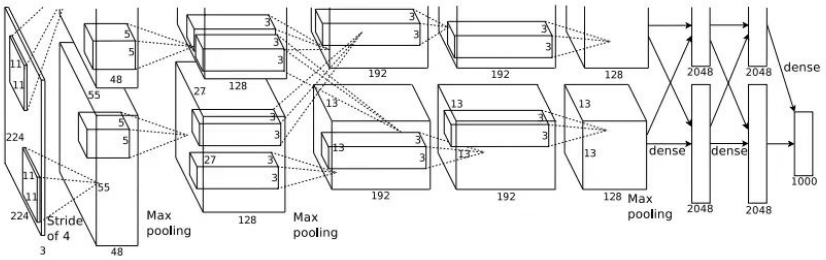

图 1: AlexNet Model

原模型分为两部分在两个 GPU 中训练。

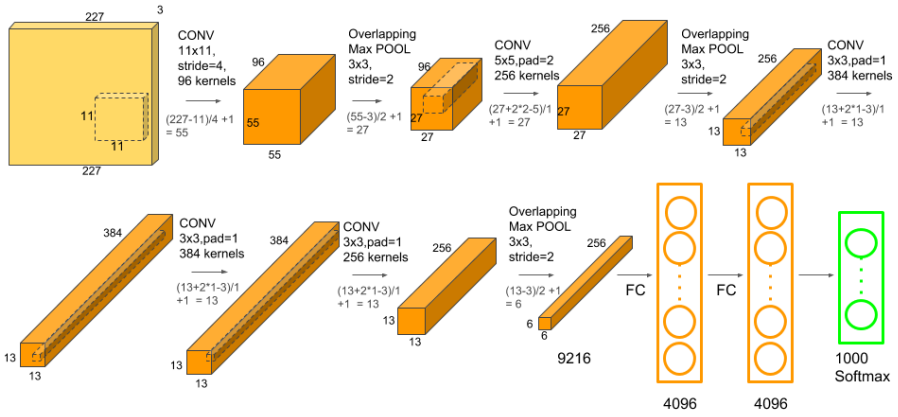

图 2: AlexNet Model

本模型将两部分合并，网络主要由五层卷积层，三层池化层，和三个全连接层组成，原始输入为 3*224*224 的张量。

网络特点: 在每个卷积层后加上 ReLU 激活函数，防止梯度消失问题，使整体收敛更快。使用 Dropout 避免过拟合。

网络初始化方法主要采用何恺明提出的初始化方法。

网络结构 (在卷积层后接 ReLU，在线性层后接 ReLU 和 DropOut):

第一层: 卷积层, kernel_size = (11,11), stride = 4, in_channel = 3,

out_channel = 96, 此时输出的 size = (227-11+0)/4+1 = 55.

第二层: 池化层, kernel_size = (3,3), stride = 2, 输出的 size = (55-3)/2+1 = 27.

第三层: 卷积层, kernel_size = (5,5), padding = 2, 输出的 size = (27-5+4)/1+1 = 27.

第四层: 池化层, kernel_size = (3,3), stride = 2, 输出的 size = (27-3)/2+1 = 13.

第五层: 卷积层, kernel_size = (3,3), padding = 1, 输出的 size = (13-3+2)/1+1 = 13.

第六层: 卷积层, kernel_size = (3,3), padding = 1, 输出的 size = (13-3+2)/1+1 = 13.

第七层: 池化层, kernel_size = (3,3), stride = 2, 输出的 size = (13-3)/2+1 = 6.

第八层: 全连接层, in_features = 9216, out_features = 4096.

第九层: 全连接层, in_features = 4096, out_features = 4096.

第十层: 全连接层, in_features = 4096, out_features = 101.

2.2.2 VGG16

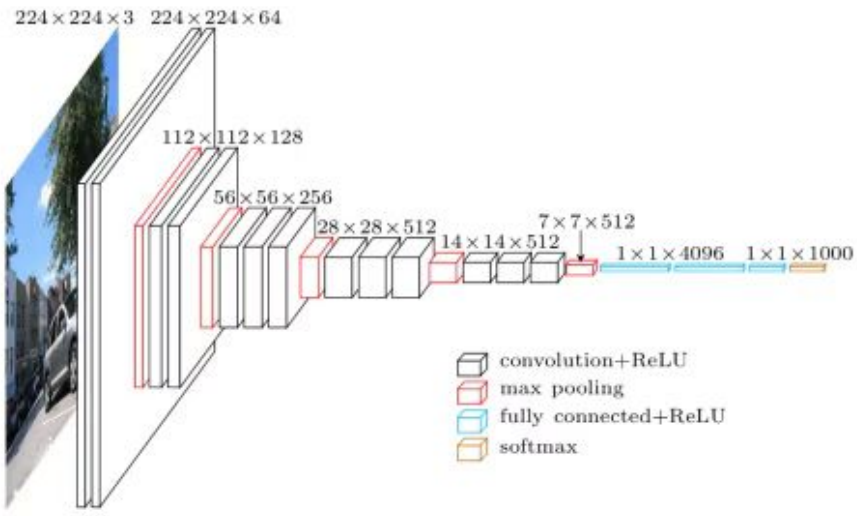

图 3: VGG16 Model

本部分主要实现了 VGG16 模型。

第一部分经过两层卷积 (在所有卷积操作后加上 ReLU)，此时输出为 64*224*224.

第二部分经过一层池化层，再通过两层卷积层，此时输出为 256*56*56.

第三部分经过一层池化层，再通过三层卷积层，此时输出为 512*14*14.

第四部分经过一层池化层，此时输出为 512*7*7.

第五部分主要为全连接层，将上一部分输出映射为 101 维的向量。

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: 500(若五十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。

在自己搭建的网络中，训练收敛较慢，基于预训练模型训练速度较快。

3.2.1 AlexNet

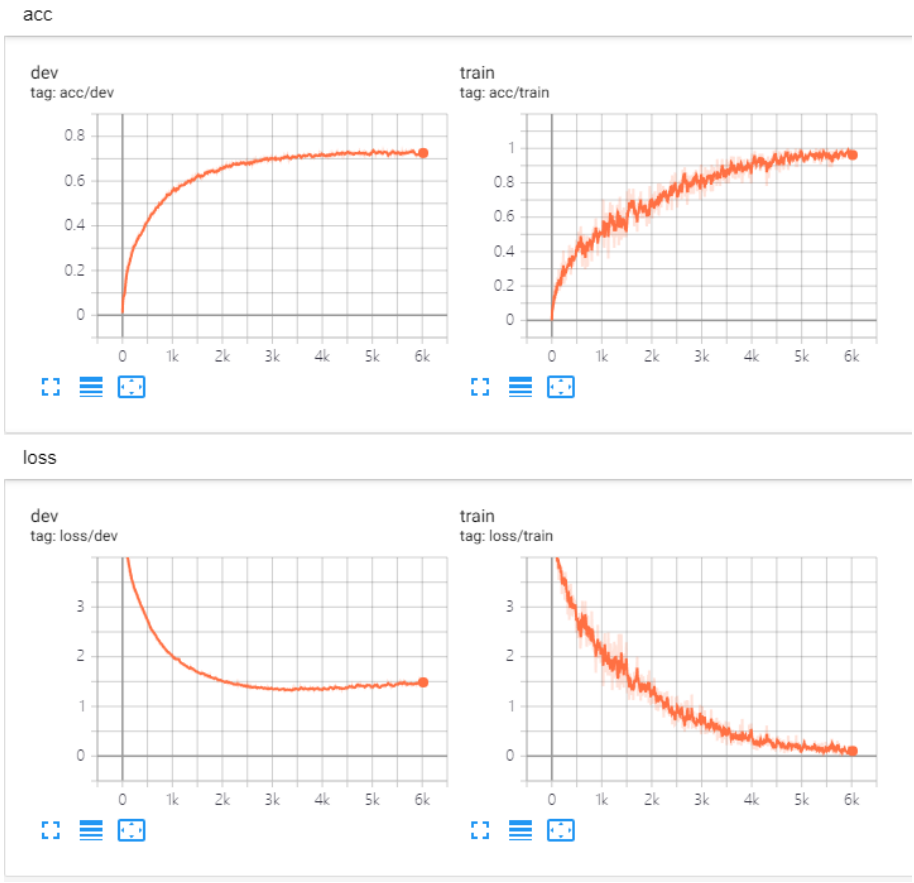

图 4: AlexNet Tensorboard

AlexNet 网络训练约 6000 轮后收敛，在验证集上准确率大约为 71%。实验发现，对网络参数初始化对网络性能有较为明显的提升。

3.2.2 VGG16

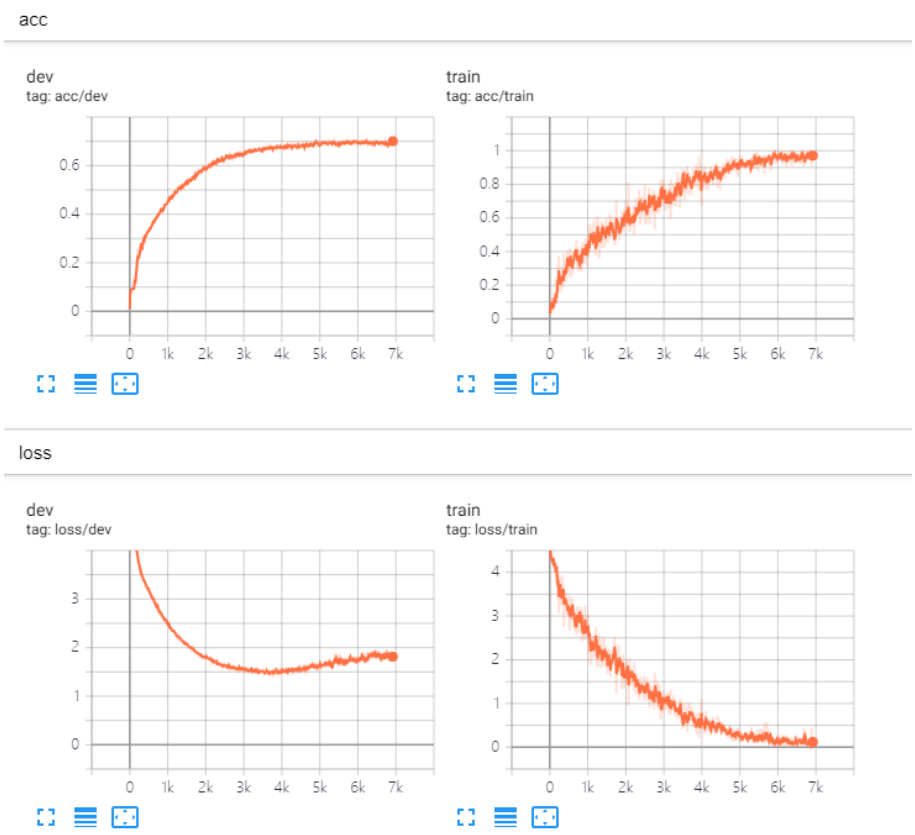

图 5: VGG16 Tensorboard

VGG16 网络训练约 7000 轮后收敛，在验证集上准确率大约为 70%。

3.2.3 ResNet50

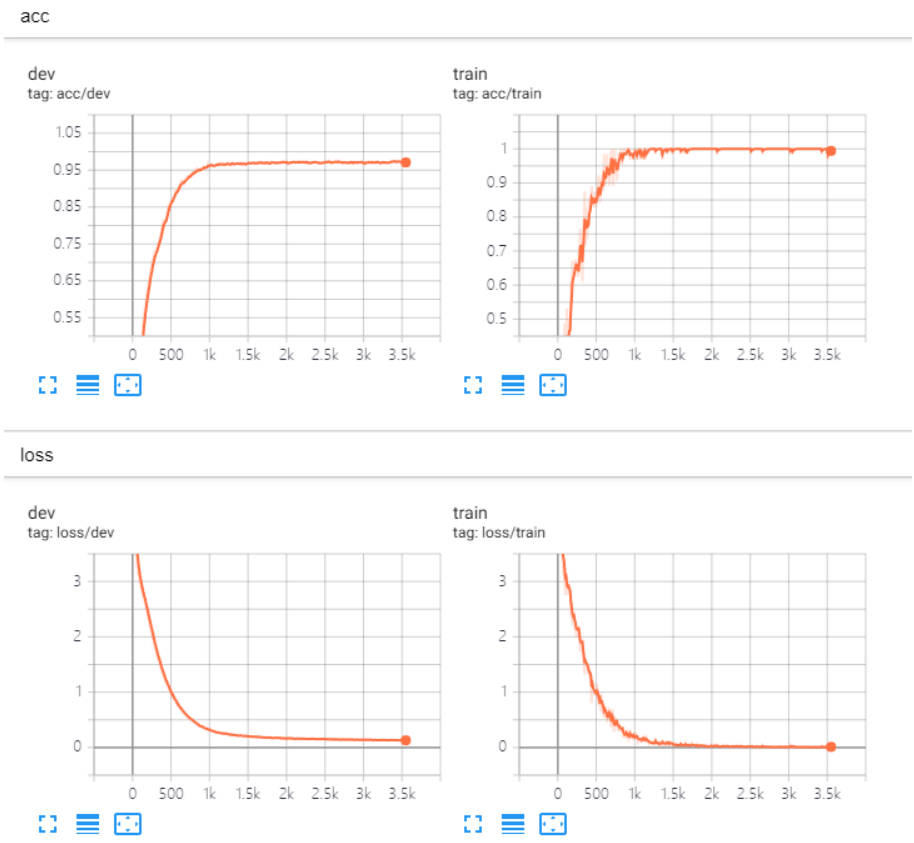

图 6: ResNet50 Tensorboard

ResNet50 网络训练约 3500 轮后收敛，在验证集上准确率大约为 97%。

3.2.4 InceptionV3

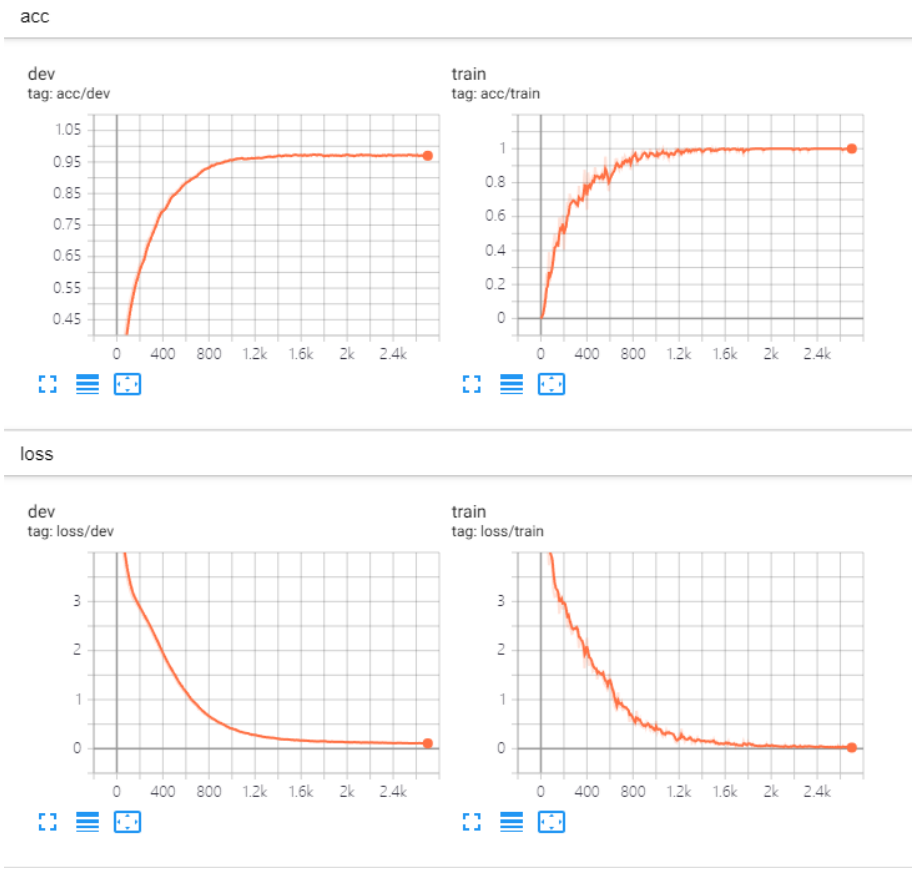

图 7: InceptionV3 Tensorboard

InceptionV3 网络训练约 2400 轮后收敛，在验证集上准确率大约为 96%。

3.2.5 DenseNet121

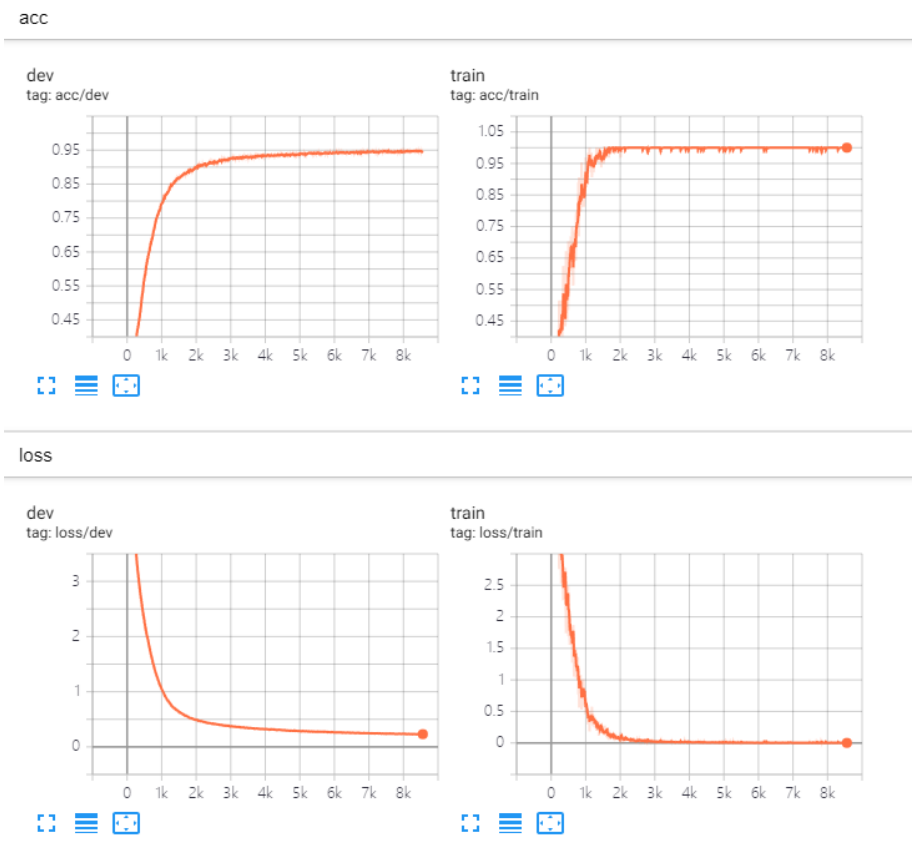

图 8: DenseNet121 Tensorboard

DenseNet121 网络训练约 8000 轮后收敛，在验证集上准确率大约为 95%。

3.3 实验结果

模型	Acc	Micro F1-Score	Macro F1-Score	Cross Entopy Loss
AlexNet	69.48%	0.694841	0.559827	0.741160
VGG16	68.28%	0.682766	0.530221	1.800057
InceptionV3	96.49%	0.964874	0.941263	0.187301
ResNet50	96.93%	0.969264	0.953177	0.137866
DenseNet121	92.43%	0.924259	0.869438	0.338787

3.4 实验总结

本次实验自己搭建网络，使我对深度学习的认知更加深刻，也学习到了更多。