课程项目二—新冠肺炎的辅助诊断

何东阳 2019011462 自96

1 项目要求

请你设计深度学习算法，来实现新冠/非新冠 CT 影像的二分类。请对给定的数据集进行 合理的数据划分，设计合适的模型评价方法和指标，对所构建的深度学习算法进行性能评估。

2 项目建模

2.1 数据集处理

本次实验中我将给定的COVID影像标注为0，将给定的non-COVID影像标注为1，所有影像按照3：7的比例手动分为测试集和训练集，使用txt文件批量记录图片路径读入`dataloader`中。将照片批量导入txt文件的程序如下：

```
import os

# path表示路径
path = "./新冠辅助诊断数据集/COVID"
# 返回path下所有文件构成的一个list列表
filelist = os.listdir(path)
# 遍历输出每一个文件的名字和类型
file = open('./新冠辅助诊断数据集/train2.txt', 'w+')
for item in filelist:
    # 输出指定后缀类型的文件
    # if(item.endswith('.jpg')):
    file.write('./新冠辅助诊断数据集/COVID/' + item + ' 1' + '\n')
file.close()
```

为了能够使用`pytorch`进行训练，需要设计自己的`dataloader`，结构如下：

```
def default_loader(path):
    return Image.open(path).convert('RGB')


class MyDataset(Dataset):
    # 构造函数带有默认参数
    def __init__(self,
                 txt,
                 transform=None,
                 target_transform=None,
                 loader=default_loader):
        fh = open(txt, 'r', encoding='UTF-8')
        imgs = []
        for line in fh:
            line = line.strip('\n')
            line = line.rstrip()
            words = line.split()
            imgs.append((words[0], int(words[1])))  #imgs中包含有图像路径和标签
        self.imgs = imgs
```

```
        self.transform = transform
        self.target_transform = target_transform
        self.loader = loader

    def __getitem__(self, index):
        fn, label = self.imgs[index]
        img = self.loader(fn)
        if self.transform is not None:
            img = self.transform(img)
        return img, label

    def __len__(self):
        return len(self.imgs)
```

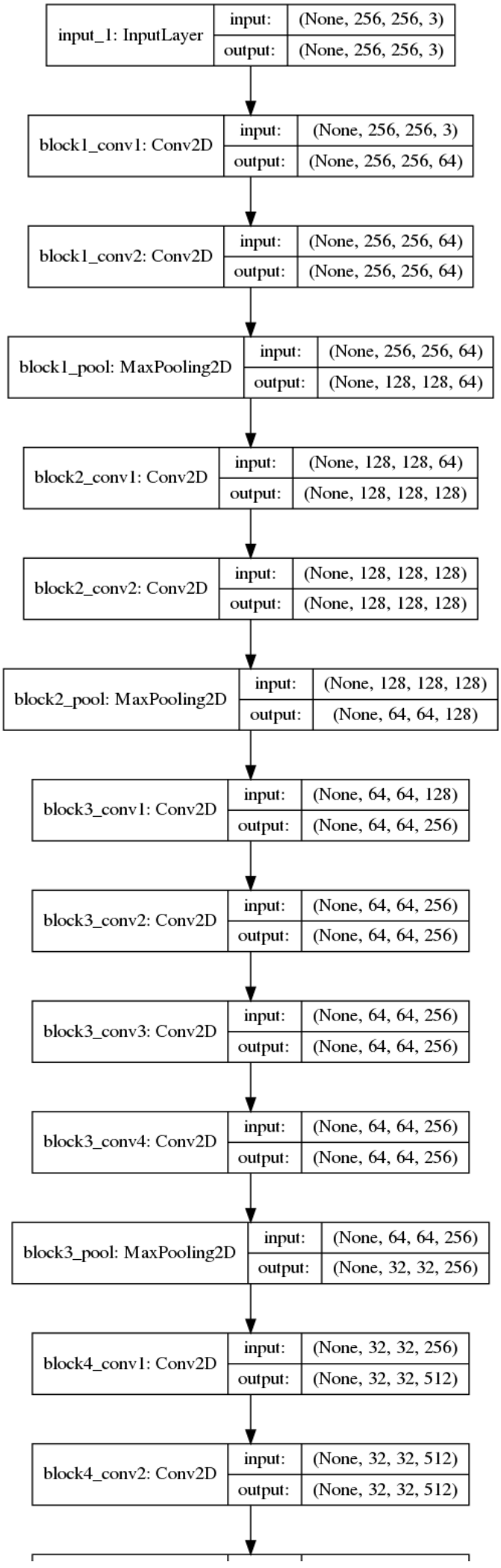
训练前通过`dataloader`导入自己的数据，导入前使用`pytorch`的`transformer`对图片进行预处理：

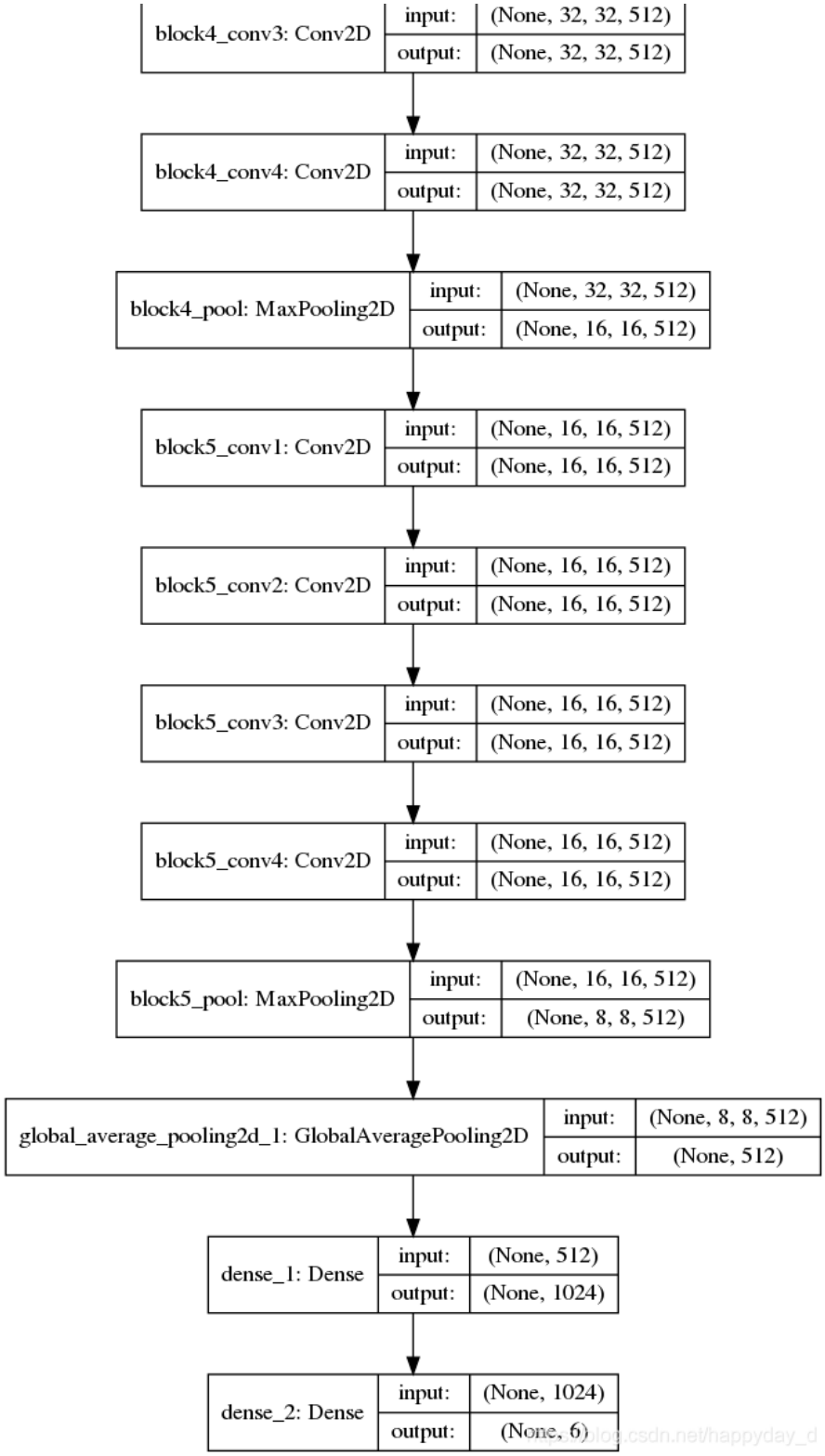
```
root = "./新冠辅助诊断数据集/"
transform_train = transforms.Compose([
    transforms.Resize([224, 224]),
    transforms.RandomGrayscale(1),
    transforms.ToTensor(),
    transforms.Normalize((0.569971, 0.569971, 0.569971),
                         (0.279340, 0.279340, 0.279340))
])

transform_test = transforms.Compose([
    transforms.Resize([224, 224]),
    transforms.ToTensor(),
    transforms.Normalize((0.569971, 0.569971, 0.569971),
                         (0.279340, 0.279340, 0.279340))
])

train_data = MyDataset(txt=root + 'train.txt', transform=transform_train)
test_data = MyDataset(txt=root + 'test.txt', transform=transform_test)

#train_data 和test_data包含多有的训练与测试数据，调用DataLoader批量加载
train_loader = DataLoader(dataset=train_data, batch_size=10, shuffle=True)
test_loader = DataLoader(dataset=test_data, batch_size=10)
```

每一张图片都被预处理成了`Tensor`的数据结构，是一种可以在GPU上加速运算的矩阵结构，在训练时可以进行被网络进行多种矩阵运算。

2.2 神经网络设计

观察图片可以发现，COIVD和non-COVID的局部区别非常明显，因此可以采用卷积神经网络才提取特征训练。本次作业我使用的是`vgg19`，这是一种非常深的卷积神经网络结构，适合本题对CT影像分类的需求。


input_1: InputLayer
input: (None, 256, 256, 3)
output: (None, 256, 256, 3)
block1_conv1: Conv2D
input: (None, 256, 256, 3)
output: (None, 256, 256, 64)
block1_conv2: Conv2D
input: (None, 256, 256, 64)
output: (None, 256, 256, 64)
block1_pool: MaxPooling2D
input: (None, 256, 256, 64)
output: (None, 128, 128, 64)
block2_conv1: Conv2D
input: (None, 128, 128, 64)
output: (None, 128, 128, 128)
block2_conv2: Conv2D
input: (None, 128, 128, 128)
output: (None, 128, 128, 128)
block2_pool: MaxPooling2D
input: (None, 128, 128, 128)
output: (None, 64, 64, 128)
block3_conv1: Conv2D
input: (None, 64, 64, 128)
output: (None, 64, 64, 256)
block3_conv2: Conv2D
input: (None, 64, 64, 256)
output: (None, 64, 64, 256)
block3_conv3: Conv2D
input: (None, 64, 64, 256)
output: (None, 64, 64, 256)
block3_conv4: Conv2D
input: (None, 64, 64, 256)
output: (None, 64, 64, 256)
block3_pool: MaxPooling2D
input: (None, 64, 64, 256)
output: (None, 32, 32, 256)
block4_conv1: Conv2D
input: (None, 32, 32, 256)
output: (None, 32, 32, 512)
block4_conv2: Conv2D
input: (None, 32, 32, 512)
output: (None, 32, 32, 512)



上图为vgg19网络的整体结构，一共有19层，整个网络都使用的是同样大小的3*3卷积核和2*2最大池化尺寸，这种深度增加和小卷积核的使用对网络的最终分类识别效果有最大的作用。

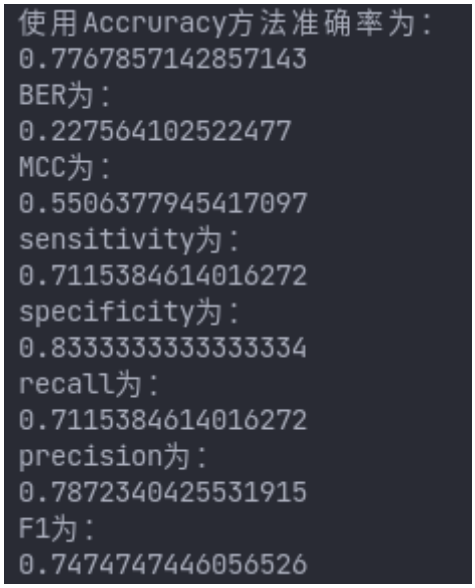
2.3 开始训练

计算`loss`的函数我使用的是交叉熵损失函数，调用的`pytorch`的接口`nn.CrossEntropyLoss()`，优化器使用的是优化效果比较好的`optim.Adam`，整个训练过程如下

```
def train(epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data = data.to(device)
        target = target.to(device)
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

3 性能评估

由于本题是二分类，因此我采用课上提到过的评估方式进行评估。



4 影响因素

4.1 超参数

在神经网络的训练过程中，对于超参数进行适当的微调对于实验最终的成功十分重要。经过多次调试，我最终确定的参数如下：

```
n_epochs = 30
learning_rate = 0.0001
weight_decay = 0.00001
batch_size = 10
log_interval = 10
random_seed = 1
```

主要影响的参数有

- 训练轮数

  由于我选取的`batch_size`比较小，因此每一轮迭代次数较多，训练几十轮基本上就收敛了。通过尝试，我发现在30轮左右训练准确率收敛到一个比较高的水平，因此设置`n_epochs`为30，不过这不是关键，较少的训练论数可以节约我们的时间。

- 学习率

对于`vgg19`和`resnet`这种大网络，一般学习率需要设置的很小才能很好的收敛到最优值，经过尝试，我发现在`learning_rate`为0.0001时最终准确率较高，所以设置此值。

- weight_decay

  这是使用`ADAM`优化器进行优化的关键参数，我分别设置了0.001、0.0001、0.00001、0等值，发现`weight_decay`为0时，最终收敛效果较好。

- batch_size

  `batch_size`是一个非常重要的参数，不同的`batch_size`会影响训练速度、训练效率、训练准确率。我发现在`batch_size`较大时，每一轮的训练速度较快，但是对于内存较小的电脑会显示内存不足的错误（这也是随机梯度下降出现的原因之一）。在`batch_size`较小时，最终的准确率又会降低，因此我选择了`batch_size`为10。

4.2 模型结构

本次题目我尝试了两个网络

- resnet50

  resnet是一个残差网络结构，预测效果较好，最终可以收敛到80%左右。

- vgg19

  vggnet的有关介绍见前方，和resnet相比，vgg的训练速度慢了许多，但是最严重收敛效果比较稳定，在80%左右。于是我使用他来作为最终的预测模型。

4.3 预处理

预处理主要分为尺寸变化、数据增强、标准化几个部分。

- 尺寸变化

  我将图片变化为了（224，224）的格式，这样既保证了图片尺寸不会过大，又保证了图片数据不失真

- 数据增强

  一开始时我考虑过旋转、缩放、噪点等数据增强方法，经过测试，我发现这些方法反而导致了最终训练准确率下降，于是放弃了数据增强的预处理。

- 标准化

  标准化是深度学习训练前的必备操作之一，首先需要计算出图片数据的方差和均值，这一部分使用网路的代码即可。经过标准化后，最终数据的测试集准确率上升了百分之三左右。

5 选做

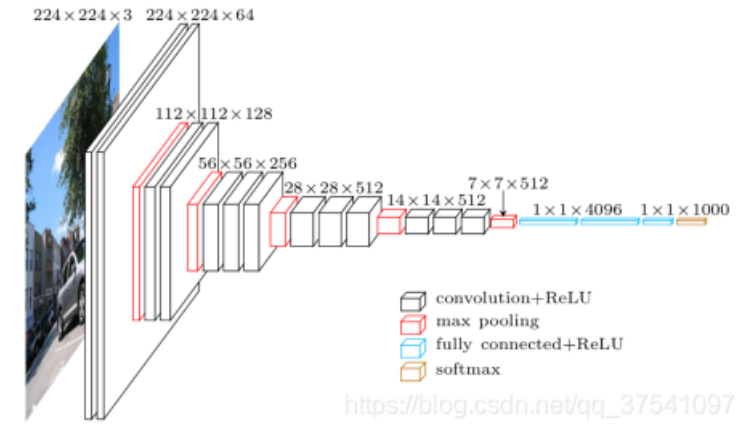
目前为止算法的效率是比较高的，但仍可以改进，可以从鲁棒性和可解释性两个层面做更多说明：

- 鲁棒性

  目前测试集是从给定的数据集中提取出来的，但是在测试中我发现随着轮数增大，测试机的准确率偶尔会突然降低百分之五左右，这说明模型预测的稳定性还有提升的空间。我认为为了增加模型的鲁棒性，需要使用数据增强等手段，消除非关键数据的影响（如重复空间、噪声等），使得模型识别的关键在于特征区域。目前学术界已经有多篇新冠状肺炎病理诊断的论文，针对于这种特定场景的二分类需要使用准确率最高的使用模型，这样才能保证最高程度的鲁棒性。

- 可解释性

有学者认为vgg网络需要消耗大量内存资源用于运算，且认为去掉中间的部分全连接层不会影响最终计算结果。神经网络的可解释性一直是学术界研究的难题，以vgg为例的网络的解释需要深入研究其中的各个卷基层、池化层、全连接层，查看各个参数的相关性和关键特征提取情况，有助于我们加深对vgg网络识别新冠病例的理解。



vgg 网络结构可视化

6 出现的问题与总结

1. 一开始的时候始终没有训练效果，后来发现是因为我将`optimizer.zero_grad()`写在了`loss.backward()`和`optimizer.step()`之间
2. 考虑使用数据增强，但我发现没有什么用，反而使得效果变得更不好了。
3. 一开始没有使用GPU进行训练，速度特别慢，后来安装CUDA后使用GPU进行训练，速度提升了十倍不止。

- 总结

  本次作业是我第一次独立完成大型神经网络的训练，实现了数据读取、网络选择、准确率测试等关键步骤，收货颇丰，希望下次我可以自己从头设计一个准确率更高的网络，进一步加深对人工智能的理解。