[Lab3] Diabetic Retinopathy Detection

0616014 楊政道

April 20, 2020

1 Introduction

1.1 Lab Objective

In this assignment, I will use pytorch to construct ResNet to analysis diabetic retinopathy. Finally, plot the confusion matrix and accuracy plot during trainging and testing.

1.2 Dataset

There are 28099 images as training dataset and 7025 images as testing dataset. The image size is 512 by 512.

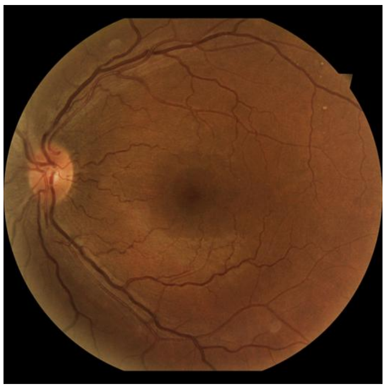

Figure 1: Dataset Sample Image

1.3 Project Structure

```
DLP_LAB3_0616014_ 楊政道.zip
├─data/
│ └─dataloader.py
├─models.py .2 weight/
│ └─record.json
├─train.py
└─evaluate.py
```

1.3.1 data/dataloader.py

To generate the pytorch dataloader for our training and testing data.

1.3.2 models.py

To define the structure of ResNet and some api for ResNet18 and ResNet50.

1.3.3 weight/record.json

To mentain the maximum test accuracy. If we have higher test accuracy, we will replace the weight and update the record file. The model parameters will be stored in weight directory.

1.3.4 train.py

To define the Trainer class and train networks.

1.3.5 evaluate.py

It will load the network parameters we have stored and evaluate the network.

2 Experiment Setup

2.1 The details of ResNet model

2.1.1 ResNet

I use pytorch api to construct the ResNet architecture. The only different part is the last layer of the model, we use a new fully-connected layer as the classification layer.

The reason why the residual shortcut can avoid vanishing gradient problems is that the gradient of the weight at least 1, so there are still gradient to be propagated back to the previous layer.

```
class ResNet(nn.Module):
    def __init__(self, num, pretrained=False, lr=1e-4):
        super(ResNet, self).__init__()

        self.pretrained = pretrained
        self.num = num
        pretrained_model = torchvision.models.__dict__['resnet{}'.format(num)](pretrained=
            pretrained)
        self.conv1 = pretrained_model._modules['conv1']
        self.bn1 = pretrained_model._modules['bn1']
        self.relu = pretrained_model._modules['relu']
        self.maxpool = pretrained_model._modules['maxpool']
        self.layer1 = pretrained_model._modules['layer1']
        self.layer2 = pretrained_model._modules['layer2']
        self.layer3 = pretrained_model._modules['layer3']
        self.layer4 = pretrained_model._modules['layer4']
        self.avgpool = nn.AdaptiveAvgPool2d(1)
        self.classify = nn.Linear(pretrained_model._modules['fc'].in_features, 5)

        del pretrained_model
```

2.1.2 ResNet18

```
def ResNet18(device, pretrained=False):
    return ResNet(18, pretrained).to(device)
```

2.1.3 ResNet50

```
def ResNet50(device, pretrained=False):
    return ResNet(50, pretrained).to(device)
```

2.2 The details of Dataloader

There are two main function need to be implemented, ___len___ and ___getitem___. The details of ___len___ function are returning the length of the dataset simply. The details of ___getitem___ are loading the image data according to the given index and doing some transformations(RandomHorizontalFlip, ToTensor, normalize).

```
class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode, arg=None):
        self.root = root
        self.img_name, self.label = getData(root, mode)
        self.mode = mode
        trans=[]
        if arg:
            trans += arg
        self.transforms = transforms.Compose(trans)

    def __len__(self):
        return len(self.img_name)

    def __getitem__(self, index):
        img = PIL.Image.open(self.root + 'jpeg/' + self.img_name[index] + '.jpeg')
        img = self.transforms(img)
        label = self.label[index]
        return img, label
```

2.3 Evaluation through the confusion matrix

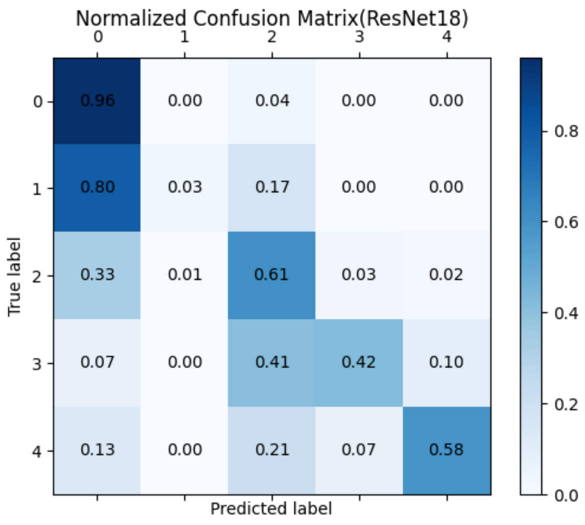

Figure 2: Confusion Matrix in ResNet18

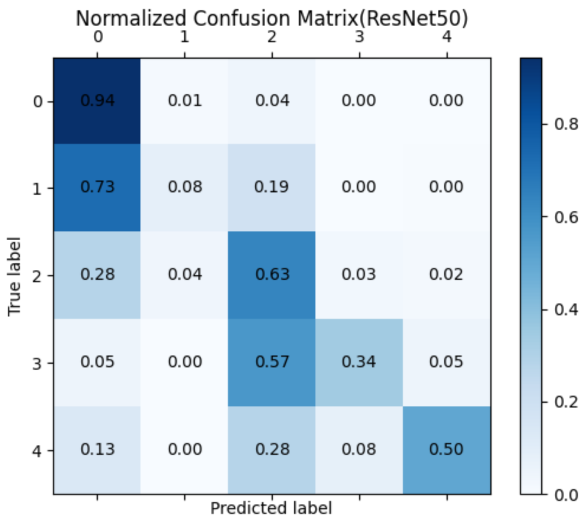

Figure 3: Confusion matrix in ResNet50

3 Experimental Result

3.1 Highest Test Accuracy

Model	Max Test Acc
ResNet18_pretrained	82.16%
ResNet18	73.35%
ResNet50_pretrained	81.34%
ResNet50	75.00%

First, I train pretrained ResNet18 with 10 epochs and pretrained ResNet50 with 5 epochs and store the weights which performs best at test dataset.

Then, I decrease the initial learning rate and train from the weight I store at the previous stage.

3.2 Comparison figures

3.2.1 ResNet18

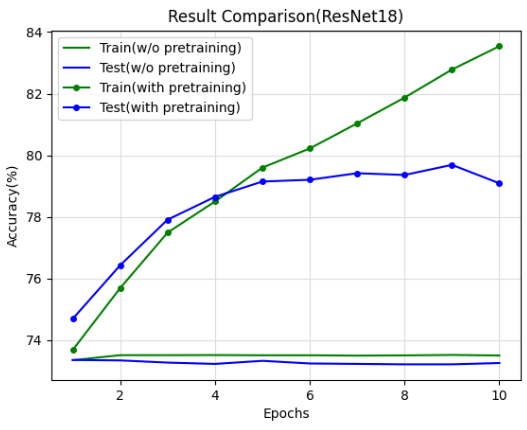

Figure 4: Result comparison(ResNet18)

3.2.2 ResNet50

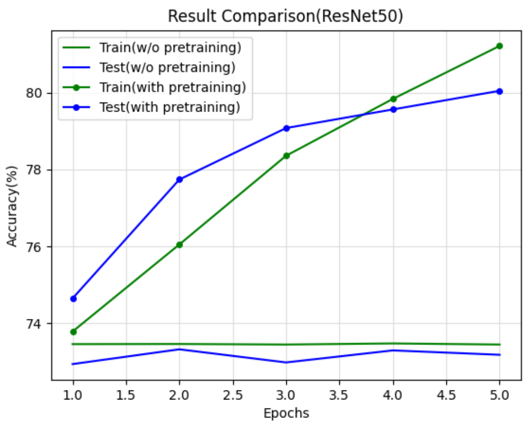

Figure 5: Result comparison(ResNet50)

4 Discussion

4.1 Data augmentation

4.1.1 RandomHorizontalFlip

Because there are left-eye and right-eye images in this dataset, I can use random horizontal flip to strong the dataset.

4.1.2 Normalize

In order to use pretrain weight, I need to make the input value into correct range. According the code on the pytorch github, the input need to be scale into [0, 1] by ToTensor transformation and use normalize transformation(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

4.1.3 Some observations on prediction

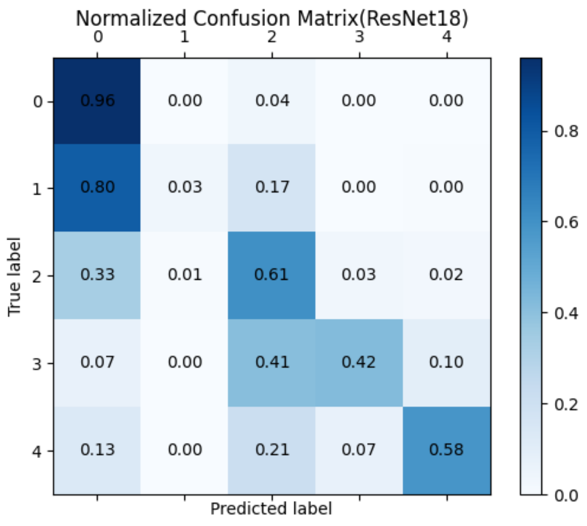

Figure 6: Confusion Matrix in ResNet18

My model predicts many class1 to class0. There are some reasons.

- The number of the images in class0 is more than class1 significantly.
- There are no obvious features to seperate class0 and class1.