DL HW3

1. Introduction

In this assignment, we should classify the diabetes on retina images. There are two models, which are ResNet18 and ResNet50 can accomplish this task. We should also implement our own dataloader and confusion matrix to evaluate the result.

2. Experiment set up

A. The detail of your model (ResNet)

I use the model from *torchvision.models.* The last layer is modified *out_channel* to 5. (Code with the yellow underline) When loading pretained model, I set *pretrained=True.* (Code with the green underline) and initialize the parameter only for the last layer.

```
class ResNet18(nn.Module):
    """docstring for ResNet18"""
    def __init__(self, is_pretrained):
        super(ResNet18, self).__init__()

        if is_pretrained:
            self.model = models.resnet18(pretrained=True)
            self.model.fc = init_weights(nn.Linear(512, 5))
        else:
            self.model = models.resnet18(pretrained=False)
            self.model.fc = nn.Linear(512, 5)

    def forward(self, img):
        return self.model(img)


class ResNet50(nn.Module):
    """docstring for ResNet50"""
    def __init__(self, is_pretrained):
        super(ResNet50, self).__init__()

        if is_pretrained:
            self.model = models.resnet50(pretrained=True)
            self.model.fc = init_weights(nn.Linear(2048, 5))
        else:
            self.model = models.resnet50(pretrained=False)
            self.model.fc = nn.Linear(2048, 5)

    def forward(self, img):
        return self.model(img)
```

B. The details of your Dataloader

I declare *self.transform = torchvision.transforms.ToTensor()* in the __init__. This function can convert the format from image to tensor. The order of channel also permute by this function.

In __getitem__, I get the path from *img_name.* I load the image by *Image.open* and convert the format by *self.transform.*

```
class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode):
        """
        Args:…
        """
        self.root = root
        self.img_name, self.label = getData(mode)
        self.mode = mode
        self.transform = transforms.ToTensor()
        print("> Found %d images..." % (len(self.img_name)))

    def __len__(self):
        """'return the size of dataset"""
        return len(self.img_name)


    def __getitem__(self, index):
        """something you should implement here"""

        """…
        """
        path = self.root + self.img_name[index] + '.jpeg'
        label = self.label[index]

        img = Image.open(path).convert('RGB')
        img = self.transform(img)

        return img, label
```

C. Describing your evaluation through the confusion matrix

I compute the confusion matrix by *sklearn.metrics.confusion_matrix.* I visualize the result as the following code. My result looks like the result from *sklearn.metrics.plot_confusion_matrix.*

```python
def plot_cm(cm, classes, filename):

    plt.imshow(cm, cmap=plt.cm.Blues)
    plt.title('Confusion matrix')
    plt.colorbar()
    plt.xlabel('Actual Class')
    plt.ylabel('Predicted Class')


    for j, i in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(i, j, "{:.2f}".format(cm[j, i]),
        color="white" if cm[j, i] > cm.max()/2. else "black",
        horizontalalignment="center")


    plt.savefig(filename+'.png')
    plt.close(0)

label_list = np.array(label_list).reshape(1, -1)[0]
pred_list = np.array(pred_list).reshape(1, -1)[0]

np.save('label_list', label_list)
np.save('pred_list', pred_list)

cm = confusion_matrix(label_list, pred_list, normalize='true')
plot_cm(cm, 5, folder_name[:-1])

plt.savefig(folder_name[:-1]+'.png')
plt.close(0)
```

3. Experimental results

A. The highest testing accuracy

- ResNet18

Without pretrained	pretrained
72.15%	80.15%

- ResNet50

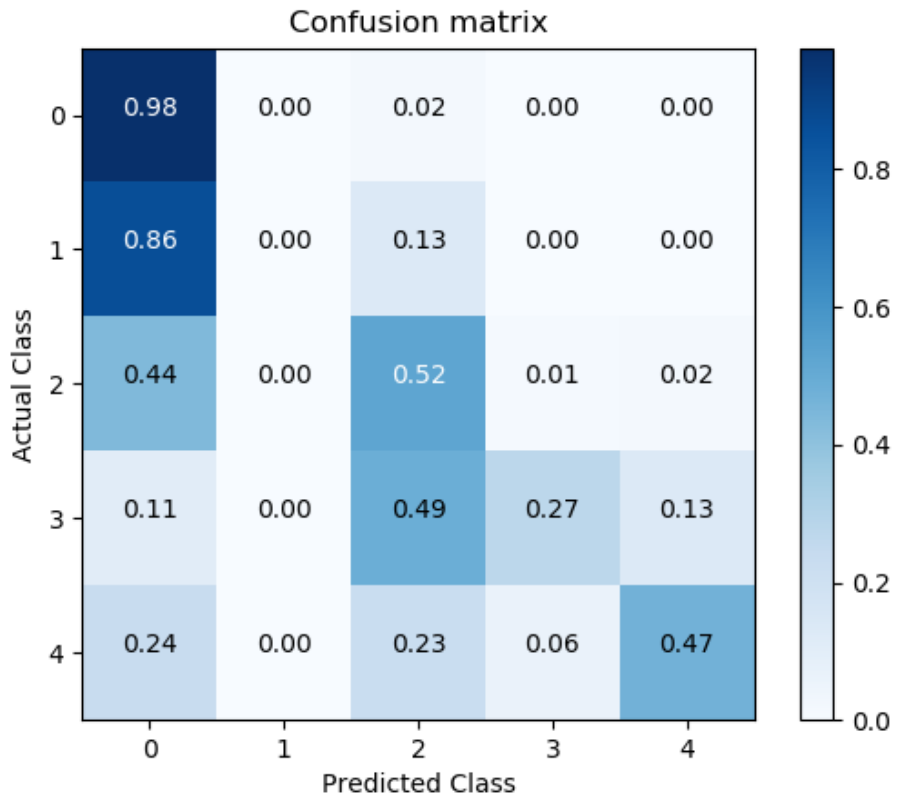
Without pretrained	pretrained
73.35%	**82.18%**

The higher tasing accuracy is resnet50 with pretrained model. The following images is the screenshot and confusion matrix of this model.

- Screenshot

```
------epoch0------
loss:0.920249
acc_train:0.740275
acc_test:0.766548
[array([3, 3, 3, 3]) array([3, 3, 3, 3]) array([3, 3, 3, 3]) ...
 array([2, 0, 0, 0]) array([0, 0, 0, 0]) array([0, 0, 0])]
------epoch1------
loss:1.274549
acc_train:0.767429
acc_test:0.791174
[array([0, 0, 0, 2]) array([0, 0, 0, 0]) array([0, 0, 0, 0]) ...
 array([0, 0, 0, 0]) array([0, 0, 0, 0]) array([0, 0, 0])]
------epoch2------
loss:0.228856
acc_train:0.786007
acc_test:0.797011
[array([0, 0, 2, 0]) array([0, 0, 0, 0]) array([0, 0, 0, 0]) ...
 array([0, 0, 0, 0]) array([0, 3, 0, 0]) array([0, 0, 0])]
------epoch3------
loss:0.659064
acc_train:0.801310
acc_test:0.805979
[array([0, 0, 0, 0]) array([0, 0, 0, 0]) array([0, 0, 2, 0]) ...
 array([0, 0, 0, 0]) array([0, 0, 0, 0]) array([0, 0, 0])]
------epoch4------
loss:0.142079
acc_train:0.823908
acc_test:0.821811
[array([0, 0, 0, 0]) array([0, 0, 2, 0]) array([0, 0, 0, 0]) ...
 array([0, 0, 0, 2]) array([0, 0, 0, 0]) array([0, 0, 0])]
```

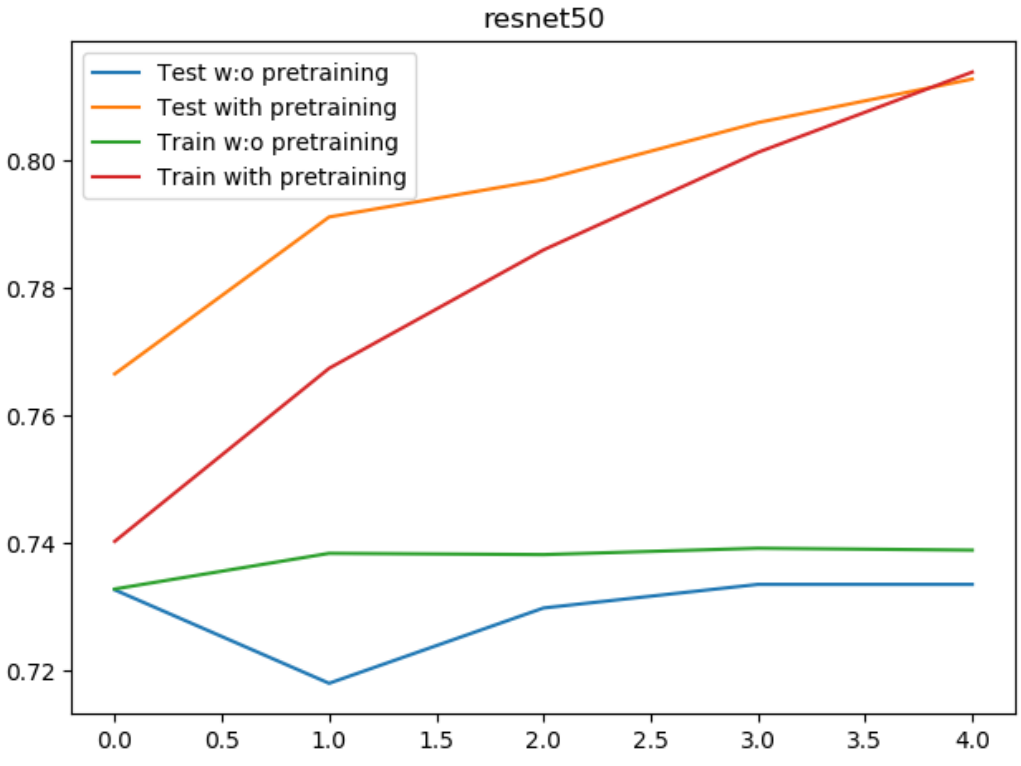
- Confusion matrix



B. Comparison figures

- ResNet18

- ResNet50



4. **Discussion**

In this assignment, I notice that the model with pretrained parameters is better than without pretrained one in some aspect. For example, with pretrained model, we can converge fast in few epoch. Also, the accuracy of pretrained one is higher than the model that learns from scratch. The reason is that the pretrained model have trained many epoch and the dataset it used may be much bigger.
To conclude, pretrained model can help us train a model faster and improve the performance.

Based on the observation from confusion matrix, I notice that many prediction are class 0. I check the training data and find the distribution of data is not uniform. Most of cases are class 0. If we want to detect the diabetes, these cases(class 0) are viewed as negative. Although the accuracy is high, most of prediction is negative positive. This can't reflect the truly situation for the accuracy of diabetes detection. Therefore, confusion matrix is important matrices we can analyze the performance of model more clearly.