Deep Learning and Practice Lab3 – Diabetic Retinopathy Detection

310554009 杜葳葳

1. Introduction

使用視網膜影像的資料集，比較 Resnet18 與 Resnet50 在 pre-trained 和 without pre-trained 的結果。糖尿病所引發的視網膜病變共分為五種類別，故此次作業為一個五種類別的分類問題。

2. Experiment setups

A. The details of your model (ResNet)

Batch size = 8

Learning rate = 1e-3

Epochs = 10

layer name	output size	18-layer	34-layer	50-layer	101-layer
conv1	112×112	7×7, 64, stride 2			
		3×3 max pool, stride 2			
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax			
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9

為了降低深層神經網路的訓練難度，Resnet 採用 identity mapping 的技術，讓在時間內可以達到足夠高的準確率。另外，在 Resnet50 時，使用 bottleneck 的設計以降低參數量。

◆ ResNet18

ResNet18 先經過一次 conv2d、batch normalization、activation (relu) 和 max pooling 後，再經過 4 層 layer，最後經過 average pooling 後過一個 fully connected layer 輸出 1000 維，再經過 activation (relu) 然後一層 Linear 輸出成 5 維 (ouput 是五個類別)

```
def forward(self, img):
    output = self.conv1(img)
    output = self.bn1(output)
    output = self.relu(output)
    output = self.maxpool(output)

    output = self.layer1(output)
    output = self.layer2(output)
    output = self.layer3(output)
    output = self.layer4(output)

    output = self.avgpool(output)
    output = output.view(output.size(0), -1)
    output = self.fc(output)
    output = self.relu(output)
    output = self.out(output)
    return output
```

```
def forward(self, x):
    residual = x
    output = self.conv1(x)
    output = self.bn1(output)
    output = self.relu(output)

    output = self.conv2(output)
    output = self.bn2(output)

    if self.downsample is not None:
        residual = self.downsample(x)

    output += residual
    output = self.relu(output)

    return output
```

- ResNet50

  與 ResNet18 類似，差別在於它的 layer 是由 [3, 4, 6, 3] 個 bottleneck 組成

```python
def forward(self, img):
    output = self.conv1(img)
    output = self.bn1(output)
    output = self.relu(output)
    output = self.maxpool(output)

    output = self.layer1(output)
    output = self.layer2(output)
    output = self.layer3(output)
    output = self.layer4(output)

    output = self.avgpool(output)
    output = output.view(output.size(0), -1)
    output = self.fc(output)
    output = self.out(output)
    return output
```

```python
def forward(self, x):
    residual = x
    output = self.conv1(x)
    output = self.bn1(output)
    output = self.relu(output)

    output = self.conv2(output)
    output = self.bn2(output)

    output = self.relu(output)

    output = self.conv3(output)
    output = self.bn3(output)

    if self.downsample is not None:
        residual = self.downsample(x)

    output += residual
    output = self.relu(output)

    return output
```

B. The details of your Dataloader

- Dataset → Dataloader (對 Dataset 進行迭代) → 對 Dataloader 進行迭代
- 讀四個 csv 檔，將照片名稱和 label 一一對應

```python
def getData(mode):
    if mode == 'train':
        img = pd.read_csv('train_img.csv')
        label = pd.read_csv('train_label.csv')
        return np.squeeze(img.values), np.squeeze(label.values)
    else:
        img = pd.read_csv('test_img.csv')
        label = pd.read_csv('test_label.csv')
        return np.squeeze(img.values), np.squeeze(label.values)
```

- 用 torchvision.transform.ToTensor 將 input image 從[H, W, C] 轉為 [C, H, W]，並將值 normalize 到 [0, 1] 之間
- __init__: 初始化整個類別，__len__: 回傳長度，__getitem__: 傳入 key、回傳對應的 value

```python
class RetinopathyLoader(data.Dataset):
    def __init__(self, root, mode):
        """
        Args:
            root (string): Root path of the dataset.
            mode : Indicate procedure status(training or testing)

            self.img_name (string list): String list that store all image names.
            self.label (int or float list): Numerical list that store all ground truth label values.
        """
        self.root = root
        self.img_name, self.label = getData(mode)
        self.mode = mode
        print("> Found %d images..." % (len(self.img_name)))

        ## transform
        self.transform = transforms.Compose([
            transforms.ToTensor()
        ])

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

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

    """
       step1. Get the image path from 'self.img_name' and load it.
              hint : path = root + self.img_name[index] + '.jpeg'

       step2. Get the ground truth label from self.label

       step3. Transform the .jpeg rgb images during the training phase, such as resizing, random flipping,
              rotation, cropping, normalization etc. But at the beginning, I suggest you follow the hints.

              In the testing phase, if you have a normalization process during the training phase, you only need
              to normalize the data.

              hints : Convert the pixel value to [0, 1]
                      Transpose the image shape from [H, W, C] to [C, H, W]

        step4. Return processed image and label
    """
    ## step 1
    image_path = './data/' + self.img_name[index] + '.jpeg'
    ## step 2
    label = self.label[index]
    ## step 3
    img = Image.open(image_path).convert('RGB')
    img = self.transform(img).numpy()

    return img, label
```

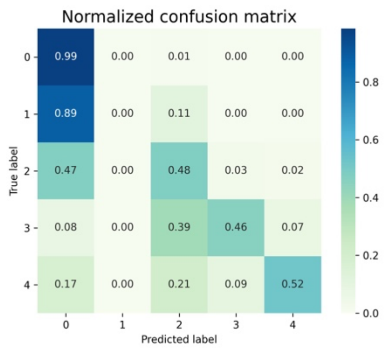
```python
train_dataset = RetinopathyLoader('', 'train')
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)

test_dataset = RetinopathyLoader('', 'test')
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=False)
```

C. Describing your evaluation through the confusion matrix

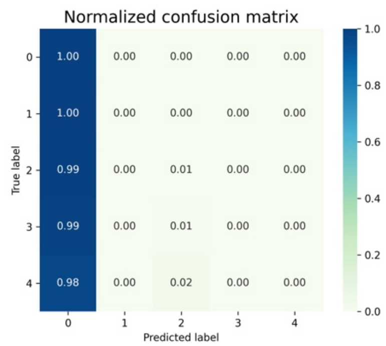
在相同參數下，pretrain 的都比沒有 pretrain 的結果來的好。

另外，因為各類別的數目不平均（label=0 的數目遠高於其他類別），故模型在判斷時傾向全部猜測為 label=0 仍能達到一定的準確率。

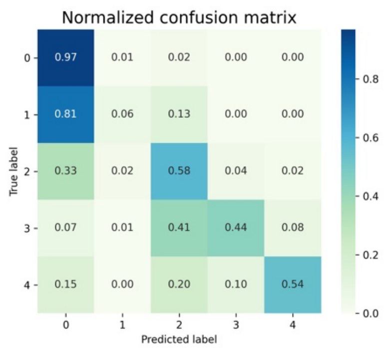
Resnet18, with pretraining



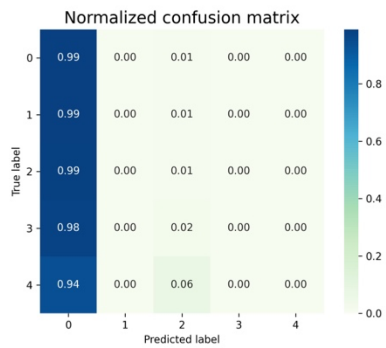
Resnet18, without pretraining



Resnet50, with pretraining



Resnet50, without pretraining



3. Experimental results

A. The highest testing accuracy

- **Screenshot (Resnet50, with pretraining)**

```
[ResNet50 with_pretraining]
Accuracy:  0.8247686832740213
```

- **Anything you want to present**

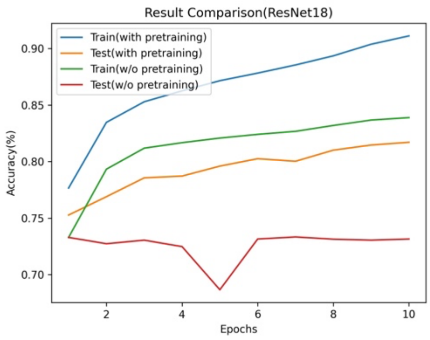
Resnet18 是 Resnet50 的子集合，故 Resnet50 的結果一定大於/等於 Resnet18，可從下表 testing 的結果觀察到。

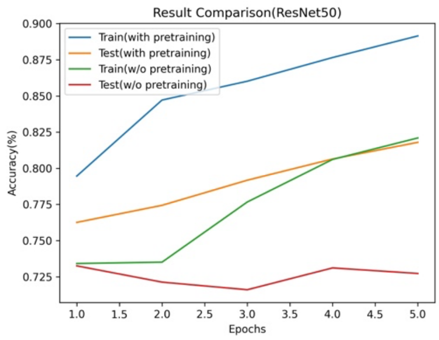
	Resnet18	**Resnet50**
pretraining	0.8173	0.8247
w/o pretraining	0.7333	0.7335

B. Comparison figures

- **Plotting the comparison figures (ResNet18/50, with/without pretraining)**

  可發現沒有 pretrain 的 model 在 epochs 數增加時，正確率並不一定一直持續上升，推斷是自己 tune 參數的 model 穩定度不好。





4. Discussion

A. Anything you want to share

在這次作業中，使用 GeForce RTX 2070 做訓練，發現加大 batch size 可以有效提升 model 的表現，但 GPU memory 只有 8.4G，故 batch size 在加到 8 以上會造成記憶體爆炸。

另外，訓練時間也是其中一個考量因素，10 個 epoch，Resnet18 花費 3 個小時、Resnet50 則花費了 6 個小時，故從折線圖推斷增加 epoch 數應該可以提升成效，但考量到需要花費的時間就沒有再加大。