

Lab 3 Report - Retinopathy Detection

- Introduction (20%)

此次實驗是基於PyTorch實作兩個neural network - ResNet18及ResNet50，而的dataset的資料內容為視網膜照片，目的是辨別因為糖尿病而造成的視網膜病變嚴重程度。

在neural network方面，兩個network的差異除了深度外，所組成的block也不相同，ResNet18會由basic block組成，而ResNet50則會由bottleneck block組成。

在實作方面，TA提供的 `getData` 及 `RetinopathyDataset` 作為基礎建立data loader，並嘗試利用一個 `ResNet` class處理ResNet18和ResNet50兩個network的建立。

- Experiment setups (30%)

1. The details of your model (ResNet)

- Common parts

1. `ResNet`

- `get_pretrained_weights`

```
183 def get_pretrained_weights(architecture):
184     if architecture == 'resnet18':
185         return ResNet18_Weights.IMAGENET1K_V1
186     else:
187         return ResNet50_Weights.IMAGENET1K_V1
```

根據architecture來決定pretrained weights的給定。

- `generate_blocks`

```

189 # Generate blocks for conv_2, conv_3, conv_4, and conv_5.
190 def generate_blocks(self, block, num_of_blocks, in_channels, stride=1):
191     down_sample = None
192
193     # If the number of channels, width, or height will be changed, do down_sample for residual.
194     if stride != 1 or self.current_channels != in_channels * block.expansion:
195         down_sample = nn.Sequential(
196             nn.Conv2d(
197                 in_channels=self.current_channels,
198                 out_channels=in_channels * block.expansion,
199                 kernel_size=1,
200                 stride=stride,
201                 bias=False,
202             ),
203             nn.BatchNorm2d(
204                 num_features=in_channels * block.expansion,
205             ),
206         )
207
208 # Build first block for conv_2, conv_3, conv_4, and conv_5.
209 layers = [
210     block(
211         in_channels=self.current_channels,
212         out_channels=in_channels,
213         stride=stride,
214         down_sample=down_sample,
215     ),
216 ]
217
218 # After building the first block, the current_channels will be changed (because of convolution
219 # in first block).
220 self.current_channels = in_channels * block.expansion
221 # Build remaining blocks for conv_2, conv_3, conv_4, and conv_5.
222 layers += [
223     block(
224         in_channels=self.current_channels,
225         out_channels=in_channels,
226     ) for _ in range(1, num_of_blocks)]
227
228 return nn.Sequential(*layers)

```

用於建立conv_2, conv_3, conv_4, and conv_5，每層皆由多個block組成，block數可由 `num_of_blocks` 來控制，而由於block可能為basic block或是bottleneck block，因此以參數傳入。

- `__init__`

1. Pretrained

```

1333         if pretrain:
1334             pretrained_weights = self.get_pretrained_weights(architecture)
1335             pretrained_resnet = getattr(torch_models, architecture)(weights=pretrained_weights)
1336             self.conv_1 = nn.Sequential(
1337                 getattr(pretrained_resnet, 'conv1'),
1338                 getattr(pretrained_resnet, 'bn1'),
1339                 getattr(pretrained_resnet, 'relu'),
1340                 getattr(pretrained_resnet, 'maxpool')
1341             )
1342
1343             # Layers
1344             self.conv_2 = getattr(pretrained_resnet, 'layer1')
1345             self.conv_3 = getattr(pretrained_resnet, 'layer2')
1346             self.conv_4 = getattr(pretrained_resnet, 'layer3')
1347             self.conv_5 = getattr(pretrained_resnet, 'layer4')
1348
1349             self.classify = nn.Sequential(
1350                 getattr(pretrained_resnet, 'avgpool'),
1351                 nn.Flatten(),
1352                 nn.Linear(getattr(pretrained_resnet, 'fc').in_features, out_features=50),
1353                 nn.ReLU(
1354                     inplace=True,
1355                 ),
1356                 nn.Dropout(
1357                     p=0.25,
1358                 ),
1359                 nn.Linear(
1360                     in_features=50,
1361                     out_features=5,
1362                 ),
1363             )

```

pretrained model 包含 NN 架構及 weights，讀入後寫入各層 layer 即可。

2. Not pretrained

```

1667         else:
1668             self.current_channels = 64
1669
1670             self.conv_1 = nn.Sequential(
1671                 nn.Conv2d(
1672                     in_channels=3,
1673                     out_channels=64,
1674                     kernel_size=7,
1675                     stride=2,
1676                     padding=3,
1677                     bias=False,
1678                 ),
1679                 nn.BatchNorm2d(
1680                     num_features=64,
1681                 ),
1682                 nn.ReLU(
1683                     inplace=True,
1684                 ),
1685                 nn.MaxPool2d(
1686                     kernel_size=3,
1687                     stride=2,
1688                     padding=1,
1689                 ),

```

```

291         # Layers
292         self.conv_2 = self.generate_blocks(
293             block=block,
294             num_of_blocks=layers[0],
295             in_channels=64,
296         )
297         self.conv_3 = self.generate_blocks(
298             block=block,
299             num_of_blocks=layers[1],
300             in_channels=128,
301             stride=2,
302         )
303         self.conv_4 = self.generate_blocks(
304             block=block,
305             num_of_blocks=layers[2],
306             in_channels=256,
307             stride=2,
308         )
309         self.conv_5 = self.generate_blocks(
310             block=block,
311             num_of_blocks=layers[3],
312             in_channels=512,
313             stride=2,
314         )

```

```

315         self.classify = nn.Sequential(
316             nn.AdaptiveAvgPool2d(
317                 output_size=(1, 1),
318             ),
319             nn.Flatten(),
320             nn.Linear(
321                 in_features=512 * block.expansion,
322                 out_features=50,
323             ),
324             nn.ReLU(
325                 inplace=True,
326             ),
327             nn.Dropout(
328                 p=0.25,
329             ),
330             nn.Linear(
331                 in_features=50,
332                 out_features=5,
333             ),
334         )

```

除了 `conv_1`，其他layer皆以 `generate_blocks` 來生成，實現共用
構建ResNet18及ResNet50的程式碼。

- `forward`

```

336     def forward(self, inputs: TensorDataset):
337         partial_results = inputs
338         for idx in range(1, 6):
339             partial_results = getattr(self, f'conv_{idx}')(partial_results)
340         return self.classify(partial_results)

```

- ReseNet18

1. `BasicBlock`

- `__init__`

```

81     def __init__(self, in_channels, out_channels, stride=1, down_sample=None):
82         super(BasicBlock, self).__init__()
83
84         self.activation = nn.ReLU(
85             inplace=True,
86         )
87         self.block = nn.Sequential(
88             nn.Conv2d(
89                 in_channels=in_channels,
90                 out_channels=out_channels,
91                 kernel_size=3,
92                 stride=stride,
93                 padding=1,
94                 bias=False,
95             ),
96             nn.BatchNorm2d(
97                 num_features=out_channels,
98             ),
99             self.activation,
100             nn.Conv2d(
101                 in_channels=out_channels,
102                 out_channels=out_channels,
103                 kernel_size=3,
104                 padding=1,
105                 bias=False,
106             ),
107             nn.BatchNorm2d(
108                 num_features=out_channels,
109             ),
110         )
111         self.down_sample = down_sample

```

基於spec上的方式建立basic block，主要包含兩次的convolution，並保持輸入輸出channel數不變。

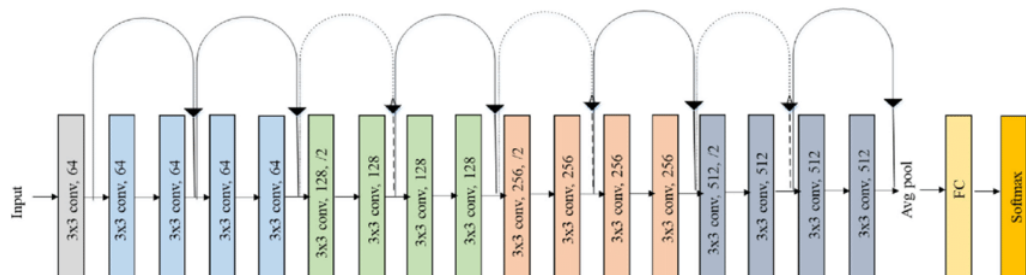
- forward

```

122     def forward(self, inputs):
123         outputs = self.block(inputs)
124         if self.down_sample is not None:
125             inputs = self.down_sample(inputs)
126             outputs = self.activation(outputs + inputs)
127
128         return outputs

```

2. resnet_18



根據此圖建立ResNet18，主要包含四層 (藍綠橙灰)，每層由兩個 `BasicBlock` 所組成，因此 `layers` 設為[2, 2, 2, 2] (後四層)，如下圖:

```

343 def resnet_18(pretrain=False):
344     return ResNet(
345         architecture='resnet18',
346         block=BasicBlock,
347         layers=[2, 2, 2, 2],
348         pretrain=pretrain,
349     )

```

- ReseNet50

1. BottleneckBlock

- `__init__`

```

128 def __init__(self, in_channels, out_channels, stride=1, down_sample=None):
129     super(BottleneckBlock, self).__init__()
130
131     external_channels = out_channels * self.expansion
132     self.activation = nn.ReLU(
133         inplace=True,
134     )
135     self.block = nn.Sequential(
136         nn.Conv2d(
137             in_channels=in_channels,
138             out_channels=out_channels,
139             kernel_size=1,
140             bias=False,
141         ),
142         nn.BatchNorm2d(
143             num_features=out_channels,
144         ),
145         self.activation,
146         nn.Conv2d(
147             in_channels=out_channels,
148             out_channels=out_channels,
149             kernel_size=3,
150             stride=stride,
151             padding=1,
152             bias=False,
153         ),
154     )

```

```

154         nn.BatchNorm2d(
155             num_features=out_channels,
156         ),
157         self.activation,
158         nn.Conv2d(
159             in_channels=out_channels,
160             out_channels=external_channels,
161             kernel_size=1,
162             bias=False,
163         ),
164         nn.BatchNorm2d(
165             num_features=external_channels,
166         ),
167     )
168     self.down_sample = down_sample

```

基於spec上的方式建立basic block，主要包含三次的convolution，且輸出channel數為輸入的channel數的四倍。

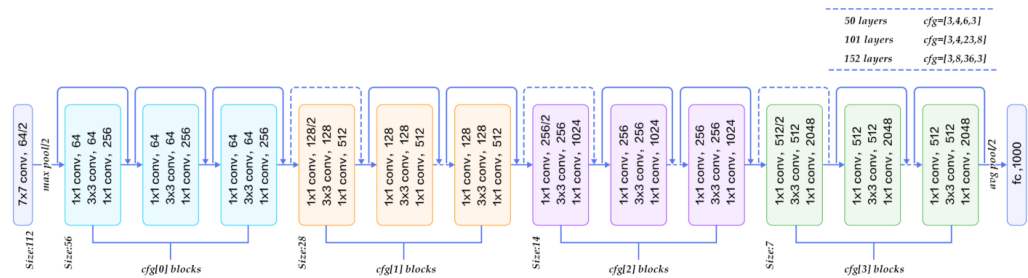
- `forward`

```

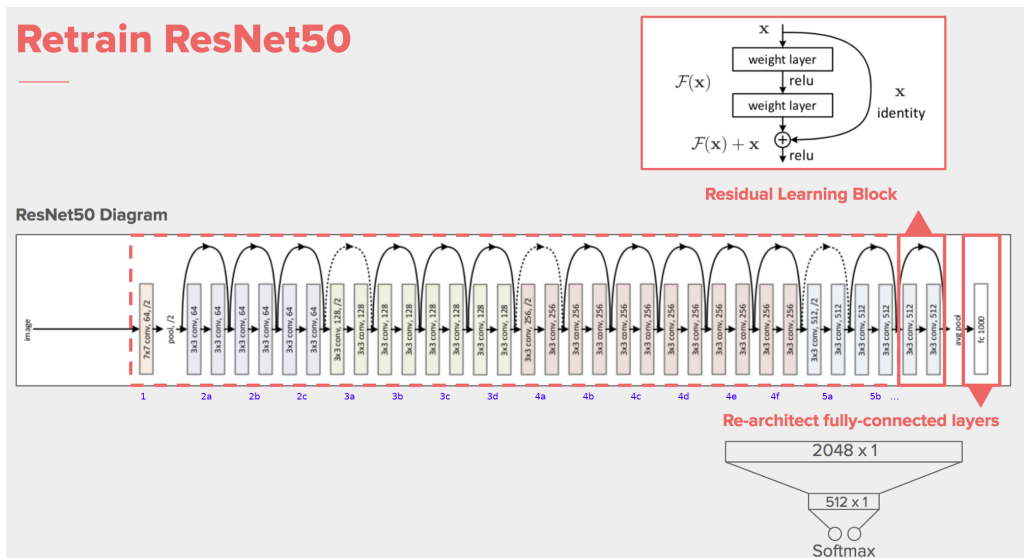
177 def forward(self, inputs):
178     outputs = self.block(inputs)
179     if self.down_sample is not None:
180         inputs = self.down_sample(inputs)
181         outputs = self.activation(outputs + inputs)
182
183     return outputs

```

2. resnet_50



根據此圖建立ResNet50，主要包含四層 (藍黃紅綠)，每層由兩個 `BottleneckBlock` 所組，但 `layers` 設為[3, 4, 6, 3] (後四層)，目的是與 paper保持一致，如下圖:



```

352 def resnet_50(pretrain=False):
353     return ResNet(
354         architecture='resnet50',
355         block=BottleneckBlock,
356         layers=[3, 4, 6, 3],
357         pretrain=pretrain,
358     )

```

2. The details of your data loader

- `__init__`

```

42 def __init__(self, root, mode):
43     """
44     Args:
45         root (string): Root path of the dataset.
46         mode : Indicate procedure status(training or testing)
47
48         self.img_name (string list): String list that store all image names.
49         self.label (int or float list): Numerical list that store all ground truth label values.
50     """
51     self.root = root
52     self.image_name, self.label = getData(mode)
53     self.mode = mode
54     print("> Found %d images..." % (len(self.image_name)))

```

與TA預設的 `__init__` function相同。

- `__getitem__`

```

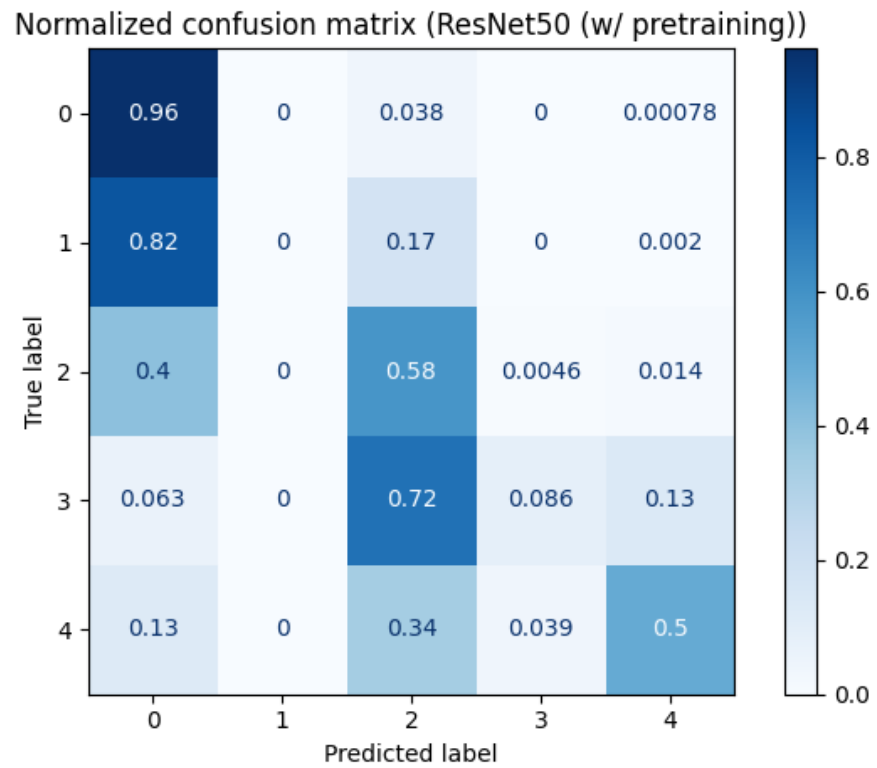
62 def __getitem__(self, index):
63     # Load image.
64     image_path = os.path.join(self.root, f'{self.image_name[index]}.jpeg')
65     # image = mpimg.imread(image_path)
66     image = PIL.Image.open(image_path)
67
68     # Get the ground truth label.
69     label = self.label[index]
70
71     # Transform the .jpeg rgb images during the training phase.
72     # Convert the pixel value to [0, 1].
73     # image = np.where(image < 128, 0, 1)
74     # Transpose the image shape from [H, W, C] to [C, H, W].
75     # image = np.transpose(image, (2, 0, 1))
76     transform = transforms.Compose([
77         transforms.RandomHorizontalFlip(p=0.5),
78         transforms.RandomVerticalFlip(p=0.5),
79         transforms.ToTensor()
80     ])
81     image = transform(image)

```

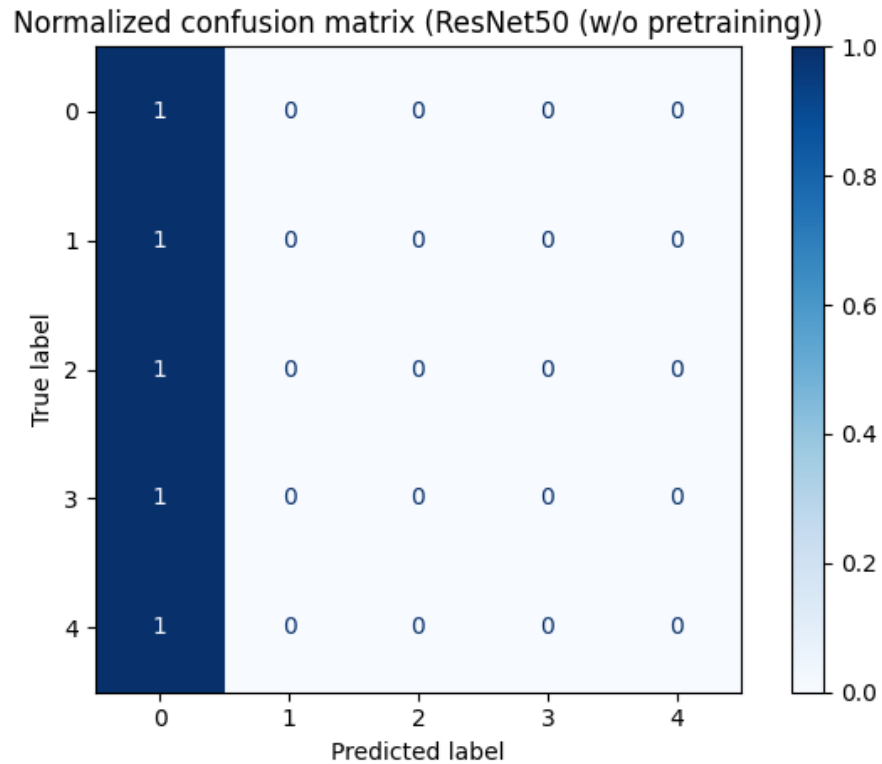
嘗試使用TA推薦的方式，先做一次二質化，並將shape由[H, W, C] 轉為[C, H, W]，及嘗試把圖片做垂直及水平反轉，增加圖片多樣性，發現第二種方式比較好。

3. Describing your evaluation through the confusion matrix

- Pretrained (ResNet50)



- Not pretrained (ResNet50)



- Discussion

由上面兩張圖可以發現，pretrained效果對比沒有pretrained來的更好，在相同的epoch數目下，model更佳general，且由comparison graph還可發現，沒有pretrained的accuracy還有下降的趨勢。

- Experimental results (30%)

1. The highest testing accuracy

- Parameters

1. model → ResNet18
2. batch_size → 32
3. learning_rate → 1e-3
4. epochs → 10
5. optimizer → SGD

6. momentum \rightarrow 0.9
 7. weight_decay \rightarrow 5e-4
- Screenshot

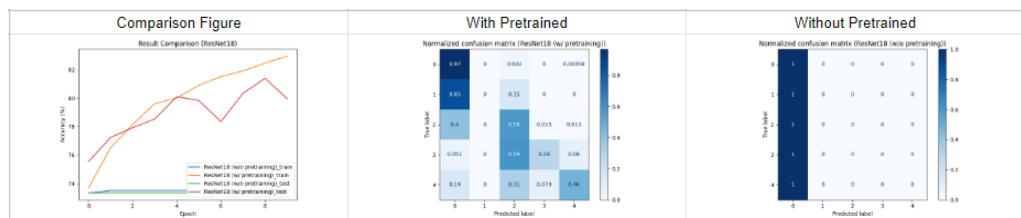
```
ResNet18 (w/o pretraining)_train: 73.52 %
ResNet18 (w/ pretraining)_train: 82.96 %
ResNet18 (w/o pretraining)_test: 73.35 %
ResNet18 (w/ pretraining)_test: 81.40 %

Process finished with exit code 0
```

2. Comparison figures

- ResNet18
 1. Parameters
 - model \rightarrow ResNet18
 - batch_size \rightarrow 32
 - learning_rate \rightarrow 1e-3
 - epochs \rightarrow 10
 - optimizer \rightarrow SGD
 - momentum \rightarrow 0.9
 - weight_decay \rightarrow 5e-4

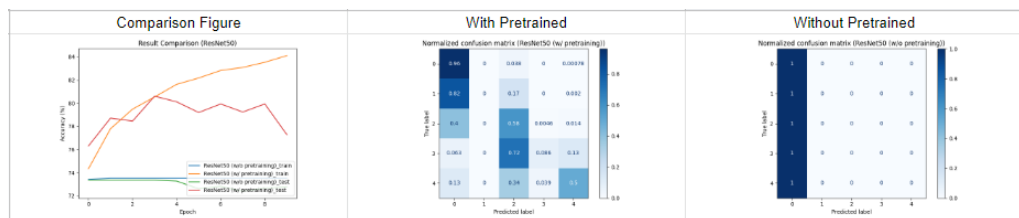
2. Results



- ResNet50
 1. Parameters

- model → ResNet18
- batch_size → 16
- learning_rate → 1e-3
- epochs → 10
- optimizer → SGD
- momentum → 0.9
- weight_decay → 5e-4

2. Results



• Discussion (20%)

1. Accuracy trend of ResNet18 & ResNet50

由上面兩張comparison figure的結果可以發現，pretrained model在epoch提高時，train的accuracy但同時test的accuracy卻下降，可能的原因是epoch還不夠多。

2. Gradient Vanishing vs. Degradation

一開始在做這個project時認為由於層數上升導致gradient在淺層的weight值變小，因此產生gradient vanishing的問題，而ResNet可以改善此問題。但在看過幾篇文章後發現，實際問題可能並非gradient vanishing，原因是他們有嘗試利用其他方式解決，諸如batch norm或drop out都無法改善gradient vanishing，因此ResNet作者提出了一個新的名詞，稱為degradation problem。而ResNet能改善此問題的原因在於Identify mapping，當輸出變成輸入 + 殘差，當殘差很小就可視為輸入，便可以視為此層無作用，我覺得可以理解為ResNet可以在training過程自己適度調節實際運作的layer層數，以最適當的符合training dataset的特性。

- Referenc

1. https://pytorch.org/tutorials/beginner/data_loading_tutorial.html
2. https://github.com/utkuozbulak/pytorch-custom-dataset-examples/blob/master/src/custom_dataset_from_file.py
3. <https://medium.com/@hupinwei/深度學習-resnet之殘差學習-f3ac36701b2f>