Lab2 :
Butterfly & Moth Classification

312551133
鄧長軒

1. Introduction

在這次的Lab中：

- 我實作了VGG19與ResNet50的模型，了解模型的結構與概念。
- 自定義了DataLoader與data前處理，了解DataLoader的運作原理。
- 從零開始訓練模型，發現了overfitting的問題，知道了解決方法。
- 最後在test data中獲得了77.2%與89.2%的準確率。

2. Implementation Details

The details of model (VGG19)：

第一階段使用nn.Sequential把Conv2d、ReLU、MaxPool2d串起來。

```
self.layer = nn.Sequential(
    nn.Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(inplace=True),
    nn.Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(inplace=True),

    nn.MaxPool2d(kernel_size=2, stride=2, padding=0),

    nn.Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(inplace=True),
    nn.Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)),
    nn.ReLU(inplace=True),

    nn.MaxPool2d(kernel_size=2, stride=2, padding=0),
```

2. Implementation Details

The details of model (VGG19)：

第二階段使用nn.Sequential把Linear與ReLU串起來。

```
self.classifier = nn.Sequential(
    nn.Linear(7*7*512, 4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, 4096),
    nn.ReLU(inplace=True),
    nn.Linear(4096, 100),
)
```

2. Implementation Details

The details of model (VGG19)：

定義forward，

讓input通過第一階段→攤平→第二階段，

得到最後結果。

```
def forward(self, x):
    x = self.layer(x)
    x = torch.flatten(x,1)
    x = self.classifier(x)
    return x
```

2. Implementation Details

The details of model (VGG19)：

使用了nn.init.kaiming_uniform_()初始化權重。

```
for m in self.modules():
    if isinstance(m, nn.Conv2d) or isinstance(m, nn.Linear):
        nn.init.kaiming_uniform_(m.weight)
```

2. Implementation Details

The details of model (ResNet50)：

定義Bottleneck，skip代表是否需要維度轉換，是的話要將stride設為2，而當in_channels=64時不需做維度轉換。

```python
class Bottleneck(nn.Module):
    def __init__(self, in_channels, out_channels, skip):
        super(Bottleneck, self).__init__()
        self.conv1 = nn.Conv2d(in_channels, out_channels//4, kernel_size=(1, 1), stride=(1, 1), padding=(0, 0))
        self.bn1 = nn.BatchNorm2d(out_channels//4)

        if skip==1 and in_channels!=64:
            self.conv2 = nn.Conv2d(out_channels//4, out_channels//4, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
        else:
            self.conv2 = nn.Conv2d(out_channels//4, out_channels//4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        self.bn2 = nn.BatchNorm2d(out_channels//4)

        self.conv3 = nn.Conv2d(out_channels//4, out_channels, kernel_size=(1, 1), stride=(1, 1), padding=(0, 0))
        self.bn3 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)

        self.skip = skip
        if skip==1 and in_channels!=64:
            self.skipconv = nn.Conv2d(in_channels, out_channels, kernel_size=(1, 1), stride=(2, 2), padding=(0, 0))
            self.skipbn = nn.BatchNorm2d(out_channels)
        elif skip==1:
            self.skipconv = nn.Conv2d(in_channels, out_channels, kernel_size=(1, 1), stride=(1, 1), padding=(0, 0))
            self.skipbn = nn.BatchNorm2d(out_channels)
```

```python
def forward(self, x):
    residual = x

    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)

    x = self.conv2(x)
    x = self.bn2(x)
    x = self.relu(x)

    x = self.conv3(x)
    x = self.bn3(x)

    if self.skip==1:
        residual = self.skipconv(residual)
        residual = self.skipbn(residual)

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

    return x
```

2. Implementation Details

The details of model (ResNet50) ：

把中間的Bottleneck串起來，

最後定義forward，讓input通過全部，得到最後輸出。

```
self.layer1 = nn.Sequential(
    Bottleneck(64, 256, 1),
    Bottleneck(256, 256, 0),
    Bottleneck(256, 256, 0),
)
self.layer2 = nn.Sequential(
    Bottleneck(256, 512, 1),
    Bottleneck(512, 512, 0),
    Bottleneck(512, 512, 0),
    Bottleneck(512, 512, 0),
)
self.layer3 = nn.Sequential(
    Bottleneck(512, 1024, 1),
    Bottleneck(1024, 1024, 0),
    Bottleneck(1024, 1024, 0),
    Bottleneck(1024, 1024, 0),
    Bottleneck(1024, 1024, 0),
    Bottleneck(1024, 1024, 0),
)
self.layer4 = nn.Sequential(
    Bottleneck(1024, 2048, 1),
    Bottleneck(2048, 2048, 0),
    Bottleneck(2048, 2048, 0),
)
```

```
def forward(self, x):
    x = self.conv(x)
    x = self.bn(x)
    x = self.relu(x)
    x = self.maxpool(x)

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

    x = self.avgpool(x)
    x = torch.flatten(x, 1)
    x = self.fc(x)

    return x
```

2. Implementation Details

The details of model (ResNet50) ：

使用了nn.init.kaiming_normal_()初始化Conv2d的權重。

用常數初始化BatchNorm2d的權重。

```
for m in self.modules():
    if isinstance(m, nn.Conv2d):
        nn.init.kaiming_normal_(m.weight)
    elif isinstance(m, nn.BatchNorm2d):
        nn.init.constant_(m.weight, 1)
        nn.init.constant_(m.bias, 0)
```

2. Implementation Details

The details of Dataloader：

getData根據mode的不同，
取得相對應的path與label。

```python
def getData(mode):
    if mode == 'train':
        df = pd.read_csv('Python/deep_learning/Lab2/dataset/train.csv')
        path = df['filepaths'].tolist()
        label = df['label_id'].tolist()
        return path, label
    elif mode == 'test':
        df = pd.read_csv('Python/deep_learning/Lab2/dataset/test.csv')
        path = df['filepaths'].tolist()
        label = df['label_id'].tolist()
        return path, label
    else:
        df = pd.read_csv('Python/deep_learning/Lab2/dataset/valid.csv')
        path = df['filepaths'].tolist()
        label = df['label_id'].tolist()
        return path, label
```

2. Implementation Details

The details of Dataloader：

__getitem__根據index來取得單獨一張圖片與label

```
def __getitem__(self, index):
    path = self.root + '/dataset/' +self.img_name[index] #+ '.jpg'
    label = self.label[index]
    img = Image.open(path)

    if self.mode == 'train':
        transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225])
        ])
    else:
        transform = transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225])
        ])

    img = transform(img)
    return img, label
```

3. Data Preprocessing

How you preprocessed your data?

首先將圖片縮放成224X224，然後轉成Tensor，同時將像素值除以255，最後進行標準化，使用常用的平均值與標準差。

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225])
])
```

What makes your method special?

我的方法很基本，因為訓練60 epoch要一整天，所以沒時間優化。

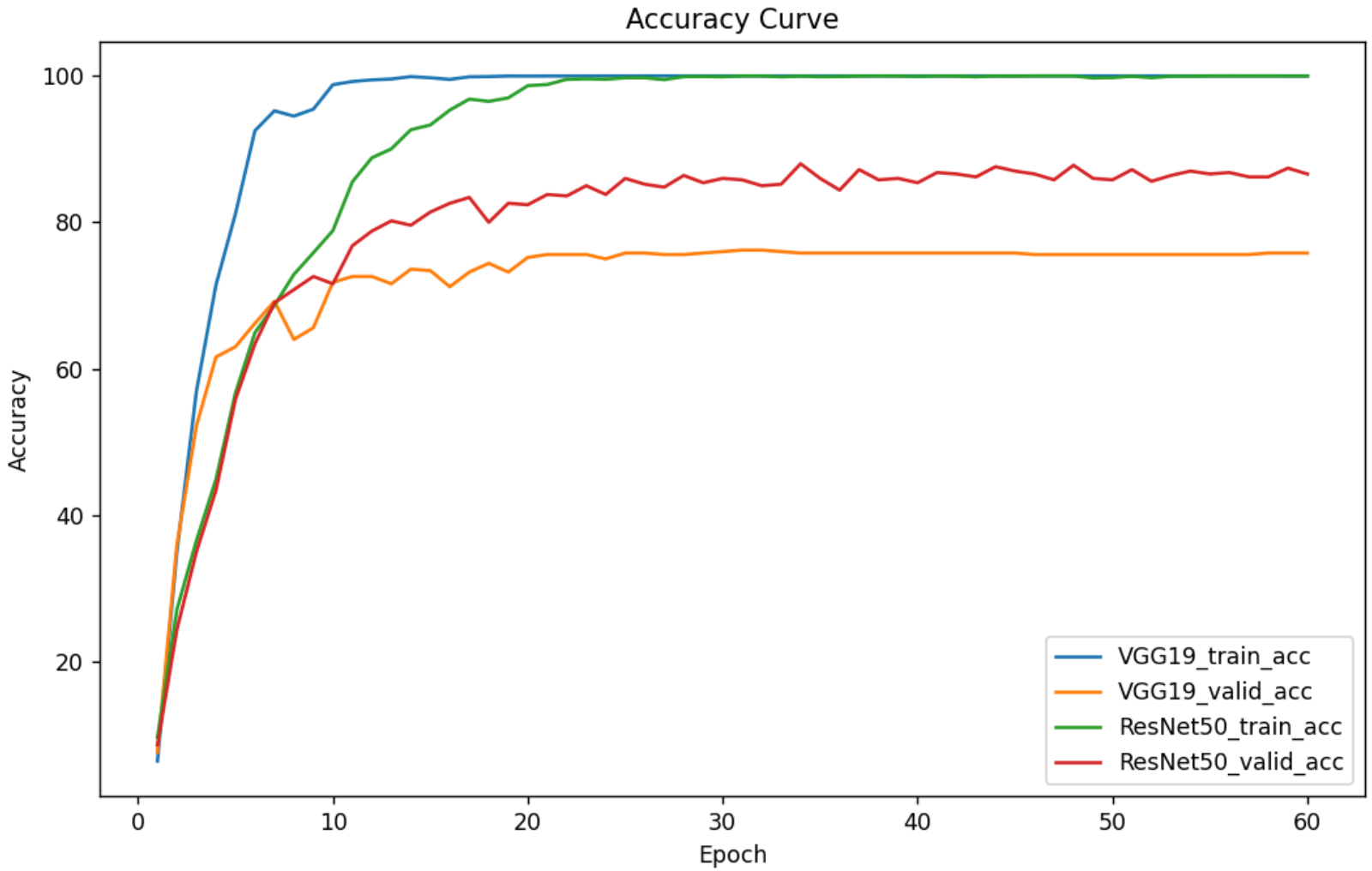
4. Experimental results

The highest testing accuracy

```
> Found 12594 images...
> Found 500 images...
----------VGG19----------
VGG19          |   Train accuracy:  100.00%|   Test accuracy:   77.20%
----------ResNet50----------
ResNet50       |   Train accuracy:   99.98%|   Test accuracy:   89.20%
```

4. Experimental results

Comparison figures



5. Discussion

- 一開始learning_rate設0.1，發現權重都不會更新，檢查了好久才發現loss出來的值都是nan，才知道learning_rate設太大了，後來調成0.001就能正常訓練。

- 從accuracy curve可以看到train_accuracy都是100%，但valid_accuacy卻只有70~90%，代表有overfitting，可用dropout()來解決，但重新訓練很花時間，所以並沒有優化，之後會記得加dropout()。