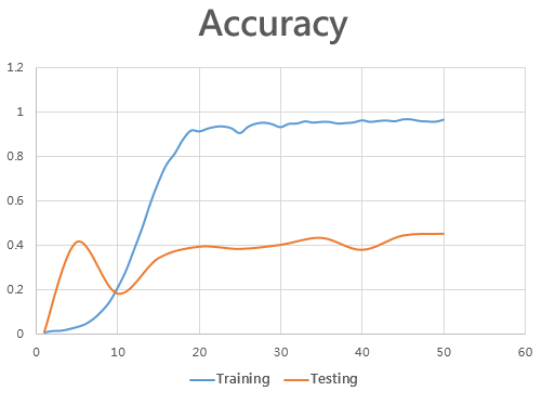
機器學習 作業二

405235035 資工三 王博輝

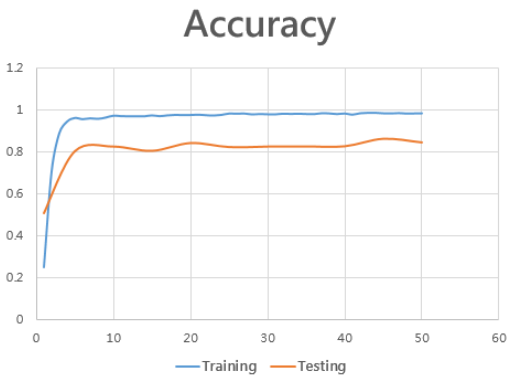
1. 無使用預訓練權重的實驗結果




```
Epoch: 46 / 50
----------------------------
Training loss:  0.157749    accuracy:  0.965128
Epoch: 47 / 50
----------------------------
Training loss:  0.165194    accuracy:  0.957883
Epoch: 48 / 50
----------------------------
Training loss:  0.182144    accuracy:  0.955796
Epoch: 49 / 50
----------------------------
Training loss:  0.195826    accuracy:  0.954199
Epoch: 50 / 50
----------------------------
Training loss:  0.151991    accuracy:  0.963409

Total Correct:  3643
Accuracy on the ALL test images:  45.3053 %
Testing loss:  3.173684
```

2. 有使用預訓練權重的實驗結果




```
Epoch: 46 / 50
----------------------------
Training loss:  0.0969    accuracy:  0.9817
Epoch: 47 / 50
----------------------------
Training loss:  0.0898    accuracy:  0.9834
Epoch: 48 / 50
----------------------------
Training loss:  0.0938    accuracy:  0.9812
Epoch: 49 / 50
----------------------------
Training loss:  0.0962    accuracy:  0.9818
Epoch: 50 / 50
----------------------------
Training loss:  0.0943    accuracy:  0.9826

Accuracy Number:  6780
Accuracy on the ALL test images:  84.32 %
Testing loss:  0.6643
```

3. 方法討論

程式主要使用作業一的程式碼，但是將從 dataset 中讀取資料的方法變更，在這邊分為兩個步驟：

(1) 預處理圖片資料

在這邊我使用了 preprocessing.ipynb 中的程式碼將訓練圖片和測試圖片做預處理，根據 mat 檔中的(x1,y1)和(x2,y2)來將圖片裁切成我們想要的影像內容，裁切完後再將裁切完的訓練及測試圖片，保留原本名稱分別存進不同的資料夾中，以便進行模型的訓練。

(2) 訓練模型

訓練模型就是從已裁切完的資料夾中讀取訓練及測試資料，原本作業一是根據父資料夾的名稱來當作是圖片的 label，這次變為使用 mat 檔中 class 欄位的值來作為圖片的 label，其他部分和上次大同小異，都是使用 Dataloader 來幫我們完成其餘工作，最後將 VGG19 的部分更換成 Resnet101 就大功告成，做圖仍是使用 excel 幫忙。

4. 問題與討論

(1) 在實驗中，雖然主體程式碼沒什麼變動，但花了非常多的時間在學習資料如何讀取，如何使用 python 從 mat 檔中讀取資料並保存至 numpy array 中是這次作業最大的挑戰，就算有了助教所提供的範例程式碼，但在真正使用時仍必須要根據情況來做調整，像是原本我在做資料的讀取時，我是將裁切圖片和訓練模型的程式碼寫在一塊，圖片讀進來裁切完後直接丟進模型做訓練，但是這樣子資料的維度會發生錯誤，就算我做過一些基本的檢查(利用 print、type、size、shape)，發現修改前後的兩個陣列的形狀維度都一樣，仍會發生錯誤，之後就放棄改為分成兩部分來做資料的讀取，才解決這個問題。

(2) 另外在實驗中，發現有無預訓練權重最大的影響是在於模型一般性的強弱，有預訓練參數的模型在面對測試資料時

有較好的效果，大概能到 85%的準確率，而無預訓練權重的模型，訓練完後面對測試資料最高大概只能到達 45%的準確率，我試過多種 seed 值的設定，準確率大概就只能在 35%~45%之間徘徊。我想這次的實驗十分能夠證明預設參數的設定對於模型的好壞有著決定性的影響，有預訓練參數的模型，對於圖像的特徵應該有更一般化的辨別能力，因為它是從 1000 類的分類問題中所得到的特徵權重，而不是單從訓練資料所得到的特徵權重，模型的一般性較好我想也是來自於此原因。

(3) 這次模型訓練還是有發生 over-fitting 的問題，無論是有無預訓練參數，模型在訓練時的準確率都能夠達到 96%~98%的準確率，但經過測試後所得到的準確率都無法堪比訓練時的結果，分別只有 45%和 85%的準確率，下次的作業就要自己動作實做一個模型了，希望到時候能夠有效地解決這個問題，讓訓練和測試的準確率不要相差太多。

(4) 有時候訓練時 accurancy 和 loss 的值會出現 nan，檢查程式碼後也沒發現到有什麼問題，不知是因為什麼樣的原因而導致這種情況發生，是因為變數的值太大或太小的原因嗎？

(5) 關於第一點，問題發生的地方是在 dataset.py 中，

```
def __getitem__(self, index):
    image = Image.open(self.x[index]).convert('RGB')
    #image = im.crop( (int(self.labels[0]), int(self.labels[1]), int(self.labels[2]), int(self.labels[3])) )

    if self.transform:
        image = self.transform(image)

    return image, self.y[index]
```

若只有第一行的敘述則沒問題，但若加上的第二行來對圖片資料做裁切，就會發生問題(第一行的變數名稱有先改成 im)。

```
Epoch: 1 / 50
----------
Traceback (most recent call last):
  File "train.py", line 95, in <module>
    train()
  File "train.py", line 52, in train
    for i, (inputs, labels) in enumerate(data_loader):
  File "Z:\environments\benzoic\lib\site-packages\torch\utils\data\dataloader.py", line 637, in __next__
    return self._process_next_batch(batch)
  File "Z:\environments\benzoic\lib\site-packages\torch\utils\data\dataloader.py", line 658, in _process_next_batch
    raise batch.exc_type(batch.exc_msg)
TypeError: Traceback (most recent call last):
  File "Z:\environments\benzoic\lib\site-packages\torch\utils\data\dataloader.py", line 138, in _worker_loop
    samples = collate_fn([dataset[i] for i in batch_indices])
  File "Z:\environments\benzoic\lib\site-packages\torch\utils\data\dataloader.py", line 138, in <listcomp>
    samples = collate_fn([dataset[i] for i in batch_indices])
  File "C:\Users\ugs01\Downloads\dataset.py", line 38, in __getitem__
    image = im.crop( (int(self.labels[0]), int(self.labels[1]), int(self.labels[2]), int(self.labels[3])) )
TypeError: only size-1 arrays can be converted to Python scalars
```

```
In [7]: im = Image.open("./cars_train/00001.jpg").convert('RGB')
        print("裁切前的type: ",type(im))
        print("裁切前的size: ",im.size)
        image = im.crop( (39, 116, 569, 375) )
        print("裁切後的type: ",type(image))
        print("裁切後的size: ",image.size)

        im = data_transform(im)
        print("裁切前並轉換後的type: ",type(im))
        print("裁切前並轉換後的shape: ",im.shape)

        image = data_transform(image)
        print("裁切後並轉換後的type: ",type(image))
        print("裁切後並轉換後的shape: ",image.shape)

        裁切前的type:  <class 'PIL.Image.Image'>
        裁切前的size:  (600, 400)
        裁切後的type:  <class 'PIL.Image.Image'>
        裁切後的size:  (530, 259)
        裁切前並轉換後的type:  <class 'torch.Tensor'>
        裁切前並轉換後的shape:  torch.Size([3, 224, 224])
        裁切後並轉換後的type:  <class 'torch.Tensor'>
        裁切後並轉換後的shape:  torch.Size([3, 224, 224])
```

很明確可以看到不論有無裁切，在經過轉換後的形狀和型態都一樣，為何有裁切的就無法順利讀進，而會顯示 TypeError: only size-1 arrays can be converted to Python scalars，這讓我十分納悶，究竟為什麼會發生這種問題呢？測試的程式碼在 problemtest.ipynb 中。