Report

110511118 陳孟頡

Part I.

- **Task:**

  Please explain the pros and cons of “regular convolution” and “depth-wise separable convolution” ?

- **Answer:**
  - 一般 convolution:

    優點:

    1. 能提取出較多特徵: 參數量較多，因此能習得更多的特徵。
    2. 使用簡單: 相較於 dw-separable convolution 較為直觀。
    3. 適合 dataset 較小應用:能解析較複雜特徵，在小 dataset 可能有較佳的 performance。

    缺點:

    1. 計算量龐大: 當資料維度提升，所需乘加法數量增加計算效率下降。
    2. 參數量多: 輸入和輸出的維度較大，參數量跟著提升。
    3. Overfitting 機率較高: 原因為參數與計算量較多。

  - depth-wise separable convolution:

    優點:

    1. 減少參數數量
    2. 減少運算量
    3. 加快訓練及推論時間

    缺點:

    1. 萃取出的特徵較少
    2. 實作較複雜
    3. Performance 可能較差

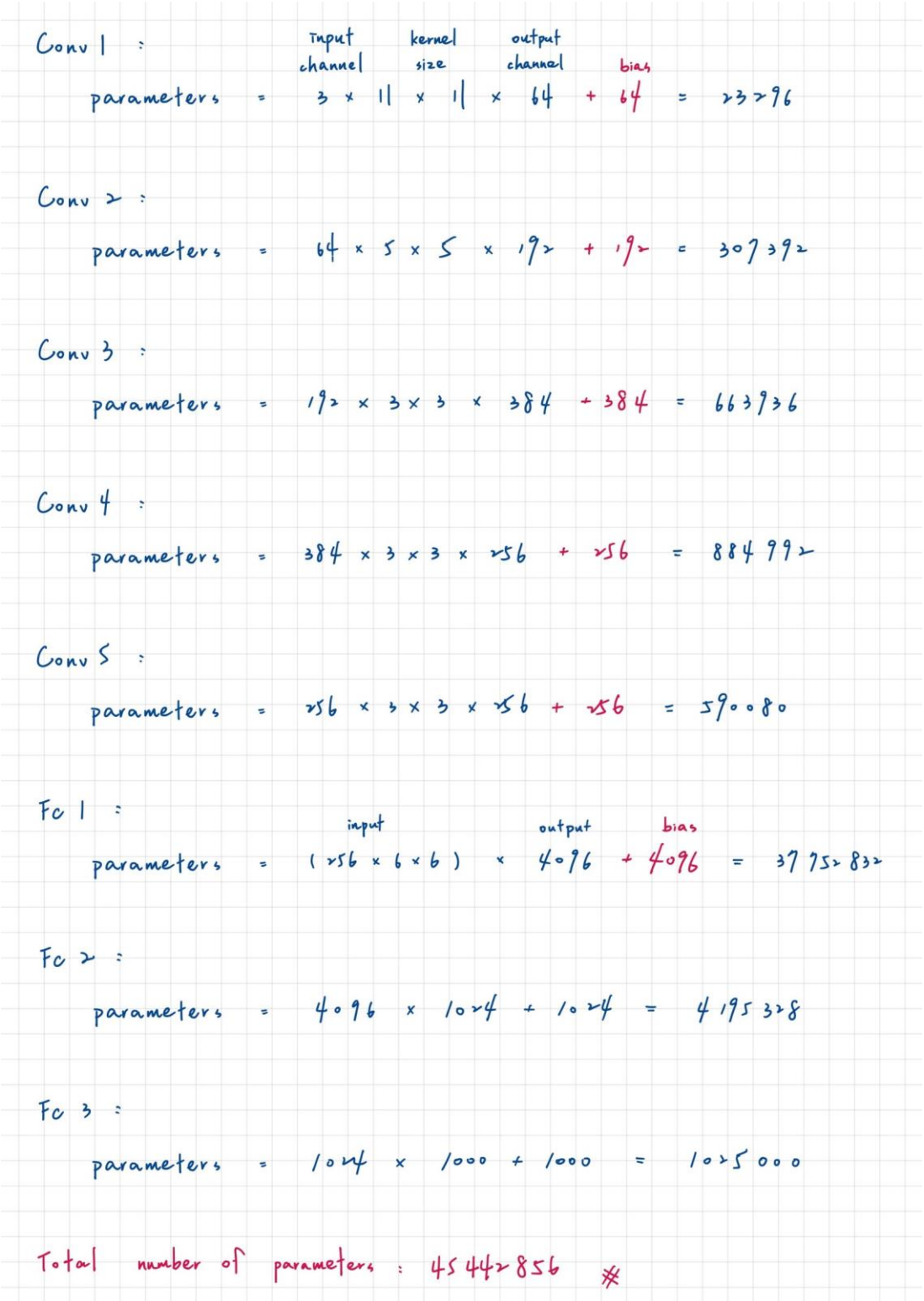
  使用 depth-wise separable convolution 主要的原因是，他能降低參數量，節省記憶空間，同時節省計算量，增佳計算效率。

Part II.

- **Task:**

  Please report the parameters of AlexNet by manual calculations. Show the actual "FLOPS / parameters" reported by code. Attached with Screenshot.

- **Manual calculation:**



- **Actual parameters/FLOPS:**

  FLOPS: 698,533,568

  Parameters: 45,442,856

```
[INFO] Register count_convNd() for <class 'torch.nn.modules.conv.Conv2d'>.
[INFO] Register count_linear() for <class 'torch.nn.modules.linear.Linear'>.
FLOPS: 698533568.0
Parameters: 45442856.0
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 55, 55]          23,296
            Conv2d-2          [-1, 192, 27, 27]         307,392
            Conv2d-3          [-1, 384, 13, 13]         663,936
            Conv2d-4          [-1, 256, 13, 13]         884,992
            Conv2d-5          [-1, 256, 13, 13]         590,080
            Linear-6                [-1, 4096]      37,752,832
            Linear-7                [-1, 1024]       4,195,328
            Linear-8                [-1, 1000]       1,025,000
================================================================
Total params: 45,442,856
Trainable params: 45,442,856
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 3.75
Params size (MB): 173.35
Estimated Total Size (MB): 177.67
----------------------------------------------------------------
```

Part III.

- **Task:**

  With hw4.py, train a CNN-based model without pre-trained weights. (Dataset: MNIST)

- **Model:**

  VGG-16

- **Implementation:**

  因 data preparation、training、testing 程式皆已提供於 hw4.py 檔案中，因此主要修改部分為 import model 以及 first layer of the model，修改後程式如下:

```
from torchvision.models import vgg16
Net = vgg16(pretrained=False)

# Modify the first convolutional layer to accept 1 channel
(grayscale) input
Net.features[0] = nn.Conv2d(1, 64, kernel_size=3, stride=1,
padding=1)

# Modify the final fully connected layer to output 10 classes
(for MNIST)
num_ftrs = Net.classifier[6].in_features
Net.classifier[6] = nn.Linear(num_ftrs, 10)
```

  因 VGG model 較大，訓練時間較久，且在發現訓練 epoch 數少情形下也能達到高準確度，因此只訓練 3 個 epoch。

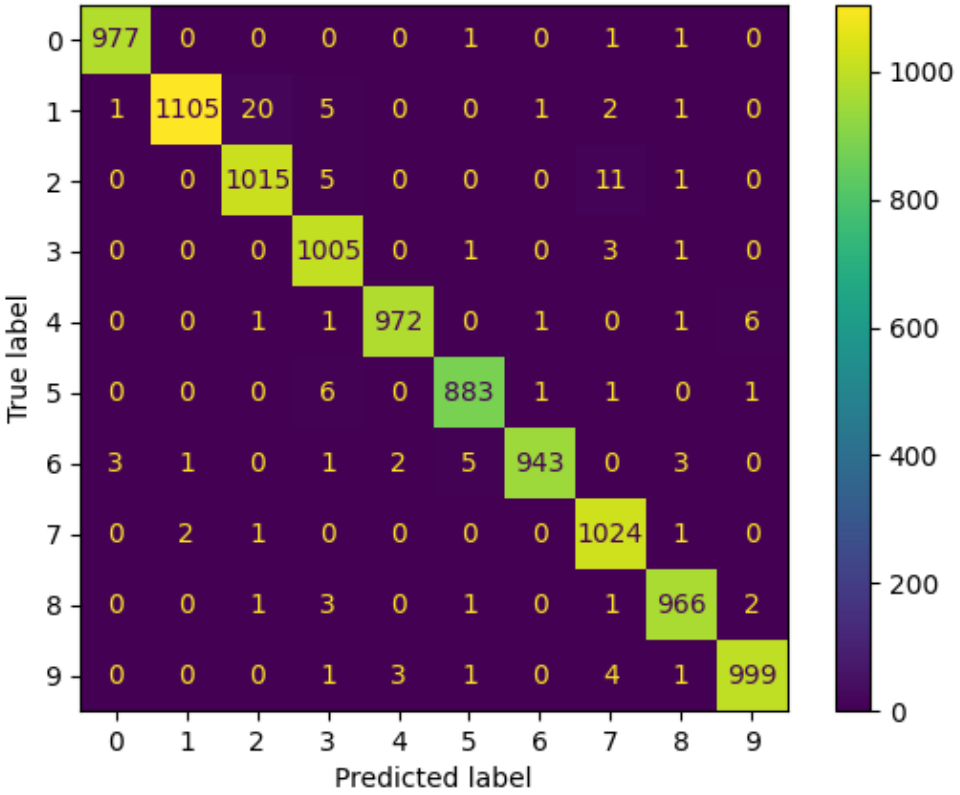
```
opt = parser.parse_args(
        args=[
            '--epochs', '3',
            '--lr', '1e-4',
            '--batch_size', '16',
            '--resume', ''
        ]
    )
```

  因 VGG-16 model 輸入大小為 224*224，因此需進行 data resize:

```
train_tfms = transforms.Compose([
        transforms.Resize(224),
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
    valid_tfms = transforms.Compose([
        transforms.Resize(224),
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
```

- Summary:
  1. Accuracy: 0.9889

```
Accuracy: 0.9889
Precision: 0.9890114694243893
Recall: 0.989113768293248
F1_score: 0.9890276312544868
```



2. Model selection:

VGG 模型保留 Alexnet 模型優點，並使用小 kernel 代替大 kernel 進行 convolution，來將模型加深，達成更高的準確度。

同時有嘗試參數量較小的 MobileNet，雖訓練速度較快，但若要達到與 VGG model 相同的準確度，需要較多的 epoch 數。附上 MobileNet 準確度(10 epoch):

```
Accuracy: 0.9852
Precision: 0.9851379207326361
Recall: 0.9852004671020858
F1_score: 0.9851475617060694
```