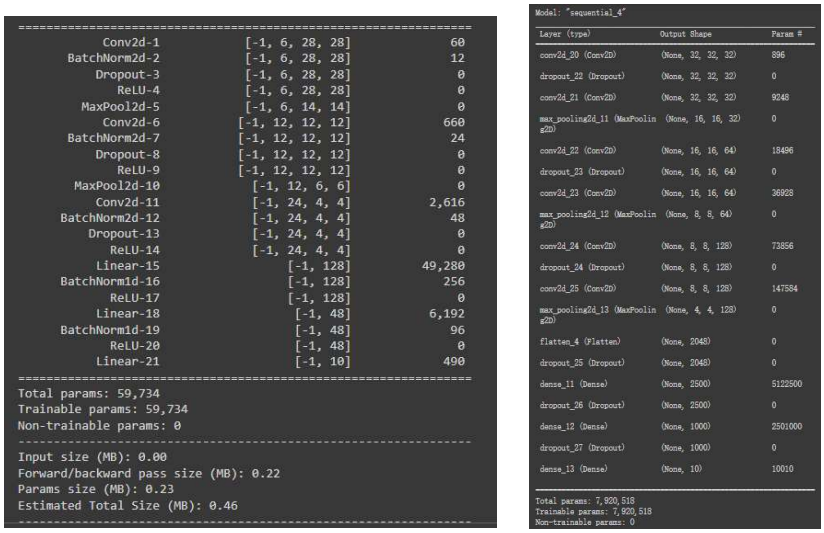
(2) LeNet CNN Classify (framework: PyTorch/Keras)

甲、摘要

搭建三層 Convolution layer 加上三層 Fully-Connected layer 拿去 Inference 於 MNIST/CIFAR-10 dataset 上。



```
----------------------------------------------------------------
        Conv2d-1          [-1, 6, 28, 28]              60
   BatchNorm2d-2          [-1, 6, 28, 28]              12
     Dropout-3            [-1, 6, 28, 28]               0
       ReLU-4             [-1, 6, 28, 28]               0
  MaxPool2d-5             [-1, 6, 14, 14]               0
        Conv2d-6          [-1, 12, 12, 12]             660
   BatchNorm2d-7          [-1, 12, 12, 12]              24
     Dropout-8            [-1, 12, 12, 12]               0
       ReLU-9             [-1, 12, 12, 12]               0
  MaxPool2d-10             [-1, 12, 6, 6]               0
       Conv2d-11           [-1, 24, 4, 4]           2,616
  BatchNorm2d-12           [-1, 24, 4, 4]              48
     Dropout-13            [-1, 24, 4, 4]               0
       ReLU-14             [-1, 24, 4, 4]               0
       Linear-15                 [-1, 128]          49,280
  BatchNorm1d-16                 [-1, 128]             256
       ReLU-17                   [-1, 128]               0
       Linear-18                  [-1, 48]           6,192
  BatchNorm1d-19                  [-1, 48]              96
       ReLU-20                    [-1, 48]               0
       Linear-21                  [-1, 10]             490
================================================================
Total params: 59,734
Trainable params: 59,734
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.22
Params size (MB): 0.23
Estimated Total Size (MB): 0.46
----------------------------------------------------------------
```

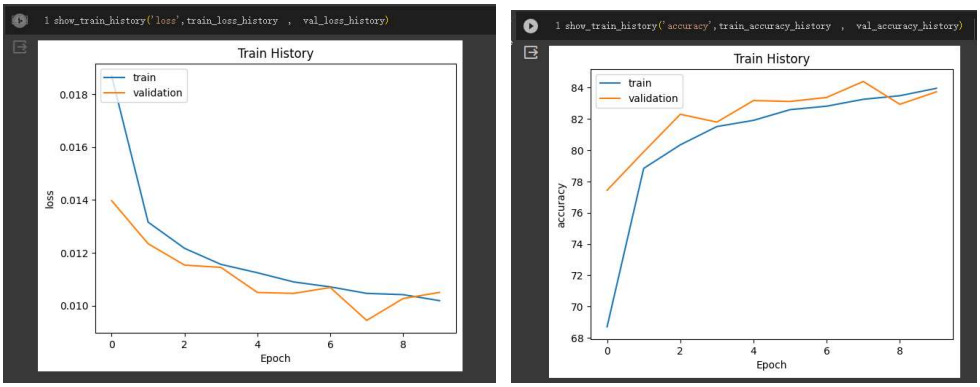
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
dropout_22 (Dropout)         (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_11 (MaxPoolin  (None, 16, 16, 32)        0
g2D)
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
dropout_23 (Dropout)         (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_12 (MaxPoolin  (None, 8, 8, 64)          0
g2D)
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
dropout_24 (Dropout)         (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_13 (MaxPoolin  (None, 4, 4, 128)         0
g2D)
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dropout_25 (Dropout)         (None, 2048)              0
_________________________________________________________________
dense_11 (Dense)             (None, 2500)              5122500
_________________________________________________________________
dropout_26 (Dropout)         (None, 2500)              0
_________________________________________________________________
dense_12 (Dense)             (None, 1000)              2501000
_________________________________________________________________
dropout_27 (Dropout)         (None, 1000)              0
_________________________________________________________________
dense_13 (Dense)             (None, 10)                10010
=================================================================
Total params: 7,920,518
Trainable params: 7,920,518
Non-trainable params: 0
```

圖一二、Model summary(左:PyTorch / 右:Keras)

乙、想法

利用 sequential 的方式搭建每一層 layer，而 optimizer 選用 Adam，其中有比較有無 Batch normalization 與 Dropout 的差別，與嘗試縮減網路深度，發現均有更好的成效，前兩者是為了防止梯度消失與過過擬合，而第三者我推測是由於 dataset 的解析度太小，若不斷 Down-sampling 反而會造成後面層數的 feature 太小或消失，造成準確率反而下降。

丙、結果



圖三四、Train history

圖五六、Mnist 與 CIFAR-10 test accuracy