Step0：環境

作業系統：MacOS Sonoma 14.4
晶片：M3 Pro CPU
記憶體：18G

```
!pip install -U scikit-learn scipy matplotlib
!pip install certifi
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from numpy import mean
from numpy import std
from matplotlib import pyplot
from sklearn.model_selection import KFold
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.optimizers import SGD
from keras.layers import Dropout
from keras.regularizers import l2
```

Step1：定義問題並建立資料

使用Fashion–MNIST 28×28 像素灰階影像資料集使用CNN訓練分類模型，下面列出了所有 0-9 整數到類別標籤的對應。

- 0：T卹/上衣
- 1：褲子
- 2：套頭衫
- 3：連身裙
- 4：外套
- 5：涼鞋
- 6：襯衫
- 7：運動鞋
- 8：袋子
- 9：踝靴

Step2: 選擇一種評量成功的準則

使訓練分數(train accuracy)和測試分數(test accuracy)在不過度擬合的情況下，越高越好。並讓訓練和測試的損失分數(Cross Entropy Loss)越加接近，且使其越低越好。另外，次要考慮時間成本。

Step3：決定驗證程序

使用多次洗牌的K折驗證(Iterated K-fold validation with shuffling)。

Step4: 準備資料

將訓練集和測試集的像素值從整數轉換為浮點數。並將訓練集和測試集的像素值歸一化處理，把像素值範圍對應到0到1之間。

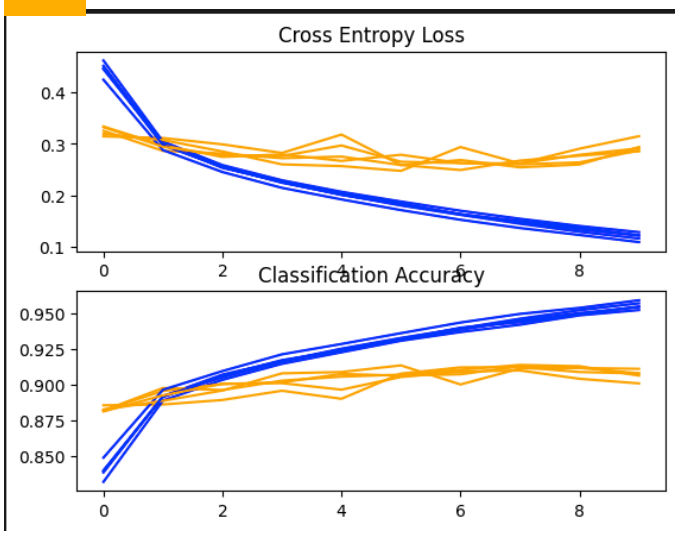
Step5: 開發出基於基準(baseline)的模型

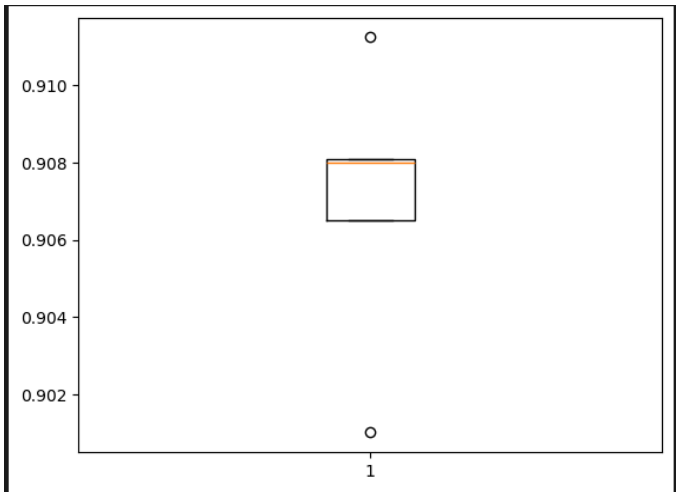
建立有一層卷基層，一層池化層(2x2)，兩層全連結層的CNN當作基準模型，其訓練集和測試集的準確度皆平均有9成，但可從損失分數中看到有些微過擬合的情況發生。

```python
def define_model_v1():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

✓ 0.0s Pyt

V1



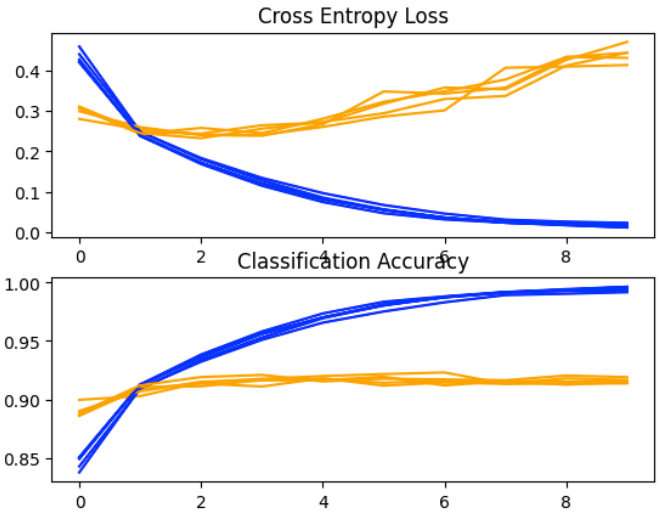


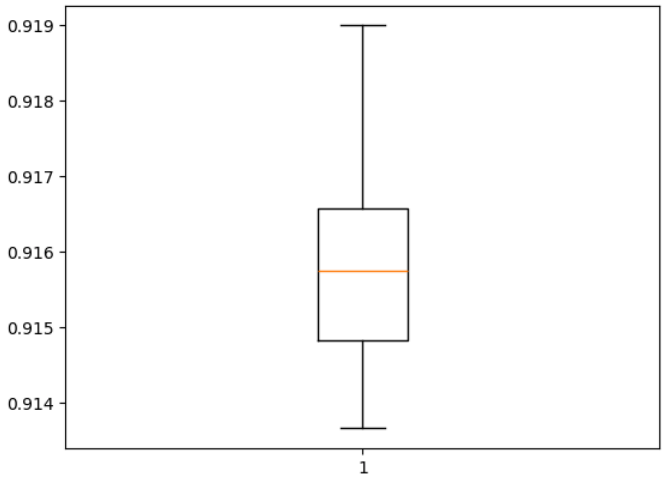
Accuracy: mean=90.697 std=0.336, n=5

Step6: 擴大規模：開發一個過度配適的模型

將池化層拔掉，並增加兩層的卷基層，各有128和256個卷機核，並且在圖形周圍填充零像素，使輸出的圖像大小可以和輸入的大小相同，可保留更多圖像的訊息。訓練集和測試集的準確度比起v1高了1.5分，但可以明顯看到過度擬合的情況發生，且訓練集損失分數比起v1有著持續上升的趨勢。

```
tabnine: test | explain | document | ask
def define_model_overfitting():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     input_shape=(28, 28, 1)))
    model.add(Conv2D(128, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform'))
    model.add(Conv2D(256, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform'))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
0.0s
```





Accuracy: mean=91.597 std=0.180, n=5

Step7: 常規化模型並調整超參數

(1).參數測試：

此章節為測試模型參數的比較，下列為程式碼部分，其輸出在下頁顯示。v1如step5所顯示有著32個卷積核；v2如下，有著64個卷積核其餘不變、v3如下，有32個卷積核並使用L2正規化；v4如下，有64個卷積核並使用L2正規化；v5如下，有32個卷積核並使用L2正規化且增加丟棄層。

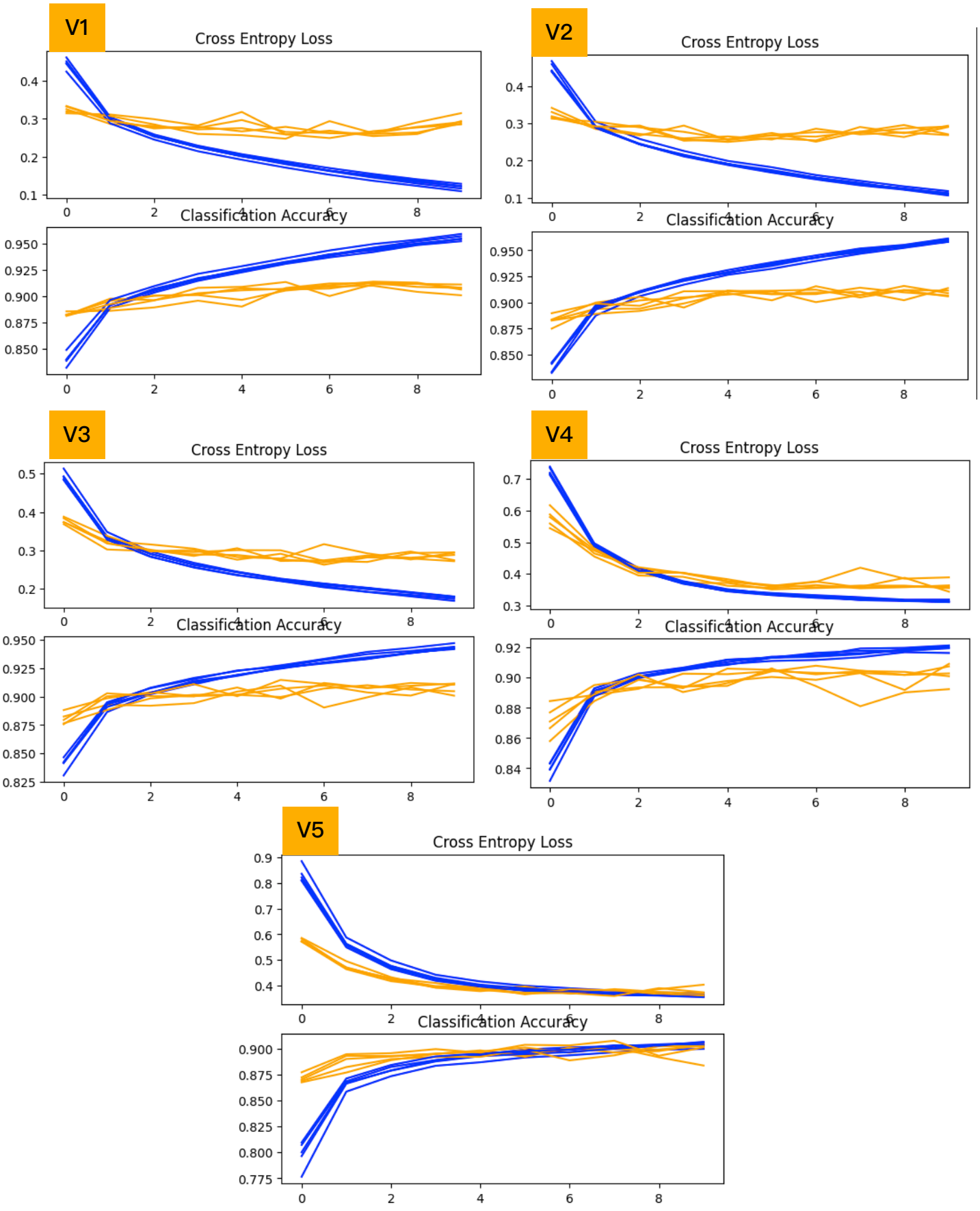
```python
# define cnn model
tabnine: test | explain | document | ask
def define_model_v2():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
✓ 0.0s
```

```python
def define_model_v3():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.001), input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
✓ 0.0s
```

```python
tabnine: test | explain | document | ask
def define_model_v4():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.001), input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
✓ 0.0s
```

```python
tabnine: test | explain | document | ask
def define_model_v5():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.001), input_shape=(28, 28, 1)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform', kernel_regularizer=l2(0.001)))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
✓ 0.0s
```

V2和V1相比沒有明顯差異；V4和V3相比有改善過擬合，但準確度有下降的情形；V3 vs. V1以及V4 vs.V2 加入L2正規化可些微改善過度擬合；V5和V4相比，可更加避免過度擬合，但會損失模型準確度，且訓練集分數明顯下降許多，可能因此有欠擬合情況發生。



(2)最佳化模型

經過幾輪調適參數，得出下列兩個表現不錯的模型架構，這兩個模型皆拔掉一層FC層，並使用兩層卷積層，但不同數量的卷積核，一層池化層，且不使用丟棄層，因為可能會使重要特徵資料丟失，導致準確度無法提升。準確度各如下，皆比V1有微幅提升，其VII提升較多、其VIII訓練時間比VII少了5倍，且準確度差異不大。

VII ：Accuracy:mean=91.200 std=0.150, n=5
VIII ：Accuracy:mean=90.910 std=0.210, n=5

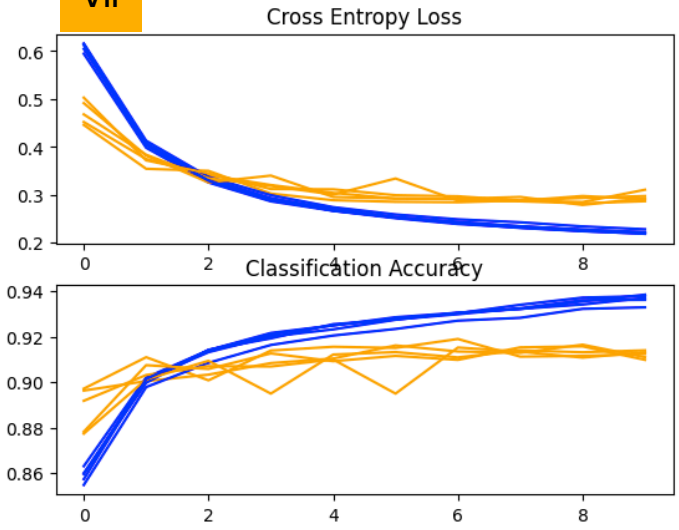
```python
def define_model_final_VII():
    model = Sequential()
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.001),  input_shape=(28, 28, 1)))
    model.add(Conv2D(64, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.001)))
    model.add(MaxPooling2D((3, 3), strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

✓ 0.0s

```python
def define_model_final_VIII():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.01),  input_shape=(28, 28, 1)))
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu', kernel_initializer='he_uniform',
                     kernel_regularizer=l2(0.01)))
    model.add(MaxPooling2D((3, 3), strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

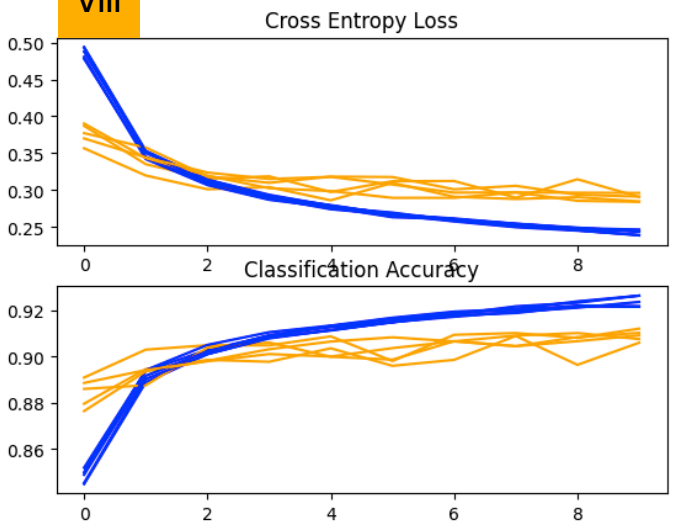
✓ 0.0s

VII




```
run_test_harness(7)
✓ 30m 17.3s
```

VIII




```
run_test_harness(8)
✓ 6m 22.8s
```

Step8：結論

經過本實驗，準確度無法有顯著提升，可能是因為資料集的照片為灰階，使其訓練出來的模型最好情況只能達到9成附近，如要有更近一步的提升準確度，因從提高資料量和彩度。