MNIST 手写 0~9 辨识

MNIST 是一個高層神經網絡 API，由 Python 編寫。爲了完成這個作業，我查看了很多 Keras 的文檔，學習了很多關於 DNN 的知識，例如：網絡層，損失函數，優化器，激活函數等，瞭解這些内容后，我才能稍微順利的完成代碼的編寫。

作業的要求是訓練及測試的正確率達到 90%以上，雖然有些曲折，但是完成了目標。其實拿到了範例代碼，修改訓練正確率到 90%以上并不困難，而且可以相當高，但是很容易就會出現 overfitting 的問題。

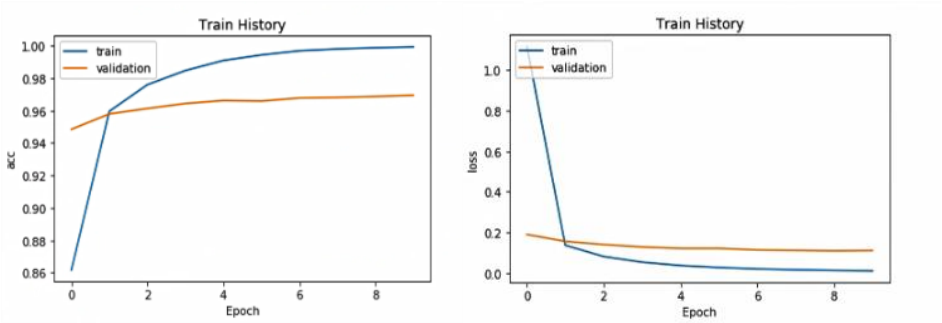
想要使正確率達到 90%以上，衹要修改這裏的代碼就好了：

```
model.add(
    Dense(
        784,
        kernel_initializer='normal',
        input_shape=(784, ),
        activation='selu'))
model.add(Dropout(0.2))      You, 2 hours ago • add MNIST
model.add(Dense(3136, kernel_initializer='normal', activation='selu'))
model.add(Dense(3136, kernel_initializer='normal', activation='selu'))
model.add(Dropout(0.2))
model.add(Dense(784, kernel_initializer='normal', activation='selu'))
#keras.optimizers.SGD(lr=0.01, momentum=0.8, decay=0.2, nesterov=True)
#sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

#model.add(Dropout(0.5))#rate: 在 0 和 1 之间浮动。需要丢弃的输入比例。
model.add(Dense(units=10, kernel_initializer='normal',
                activation='softmax'))  # Add Hidden/output layer
```

簡單修改每一層的神經元個數、每一層的激活函數、隱藏層的層數 只修改這三的參數就可以輕鬆將訓練正確率達到 95%以上，驗證的正確率下文再説。最初的代碼衹有兩層，通過修改激活函數就可以明顯提高正確率，而想要進一步提高，就需要增加隱藏層了，具體增加幾個隱藏層，以及每個隱藏層的神經元的個數還有每一層的激活函數可以自己去嘗試，但不要太多，否則訓練過程會很漫長。具體如何修改，可以參照 keras 的文檔去做。

經過前面的修改，validation 要想达到 95%以上并不太困难，最大的问题是 overfitting。我的代码一开始只追求 accuracy，结果很快就可以达到 99.94%，但是 validation 不足 97%，overfitting 问题非常严重，就像下图：



從圖中就可以看到 validation 的值變化並不大，也就是説訓練出的參數并不完全適合 validation 的數據，這些參數都是在盡可能的符合 train 的數據，造成了上圖 over fitting 的結果。

Overfitting 有三種解決方案：

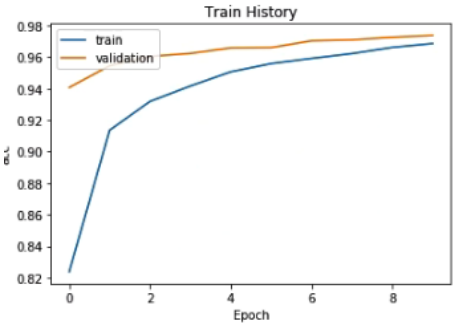
1、 dropout
2、 earlystop
3、 regulation

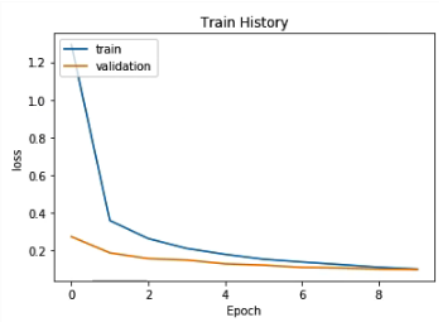
我的代碼使用 dropout，這個最簡單，衹要加一行代碼就好：

```
model.add(
    Dense(
        784,
        kernel_initializer='normal',
        input_shape=(784, ),
        activation='selu'))
model.add(Dropout(0.2))       You, 2 hours ago • add MNIST
model.add(Dense(3136, kernel_initializer='normal', activation='selu'))
model.add(Dense(3136, kernel_initializer='normal', activation='selu'))
model.add(Dropout(0.2))
model.add(Dense(784, kernel_initializer='normal', activation='selu'))
#keras.optimizers.SGD(lr=0.01, momentum=0.8, decay=0.2, nesterov=True)
#sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
#model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

#model.add(Dropout(0.5))#rate: 在 0 和 1 之间浮动。需要丢弃的输入比例。
model.add(Dense(units=10, kernel_initializer='normal',
                activation='softmax'))  # Add Hidden/output layer
```

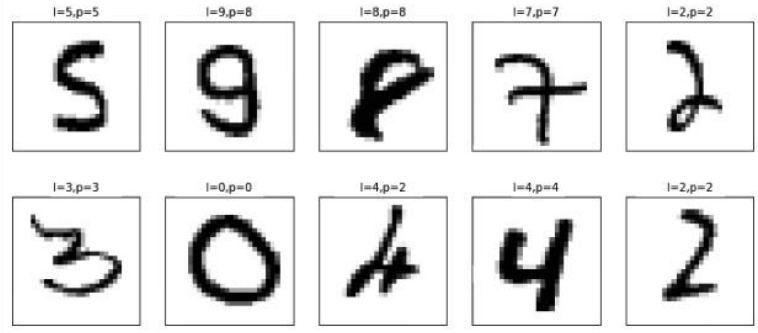
仍然是使用上面的圖片，第七、十行，分別定義要隨機關閉的神經元的數量。我的實驗結果告訴我這裏的數值不要太大，去掉的神經元太多，雖然解決了 overfitting 的問題，但同時會影響正確率，可以多次少量的去掉一些，這樣就能解決問題了。（dropout 會拉低 train 的正確率）






```
[Info] Accuracy of testing data = 97.4%
[Info] Making prediction to x_Test_norm

[Info] Show 10 prediction result (From 240):
[5 8 8 7 2 3 0 2 4 2]
```



後面兩種方法我沒有嘗試。

具體的內容請看代碼，裏面我還嘗試了使用其他的優化器等等的，validation 的結果差不多。