使用不同優化器、調整 epoch、batch size、隱藏層數、隱藏層節點數

資管數二 7107029022 邱靖詒

MNIST 手寫辨識資料集

A. 分割測試集、訓練集

In [1]:
```
from keras.datasets import mnist
from keras import models
from keras import layers
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
print("[Info] train data={:7,}".format(len(train_images)))
print("[Info] test  data={:7,}".format(len(test_images)))
```

```
D:\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion o
f the second argument of issubdtype from `float` to `np.floating` is deprecate
d. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
[Info] train data= 60,000
[Info] test  data= 10,000
```

B. 前置作業

In [2]:
```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

from keras.utils import to_categorical

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

C. 定義繪製圖形之函式

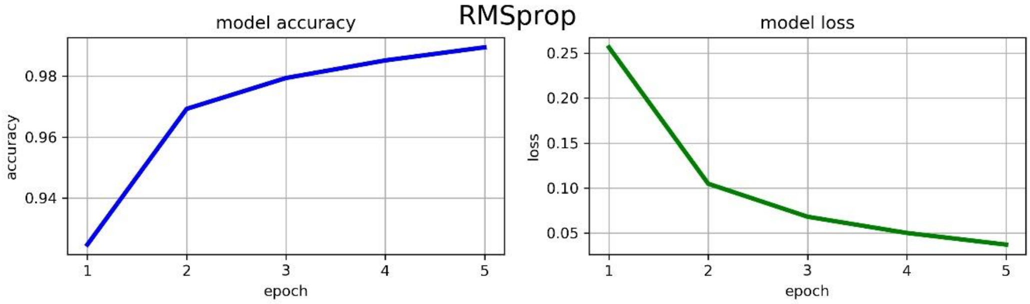
In [7]:
```
def show_train_history(train_history, train, validation, epoch):
    plt.plot(train_history.history[train], linewidth=3)
    plt.plot(train_history.history[validation], linewidth=3)
    plt.title('Train History')
    plt.ylabel(train)
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'], loc='best')
    plt.grid(True)
    if train == 'acc':
        plt.savefig("epochs_acc_" + str(epoch) + ".jpg")
    if train == 'loss':
        plt.savefig("epochs_loss_" + str(epoch) + ".jpg")
    plt.show()
```

優化器 Optimizer

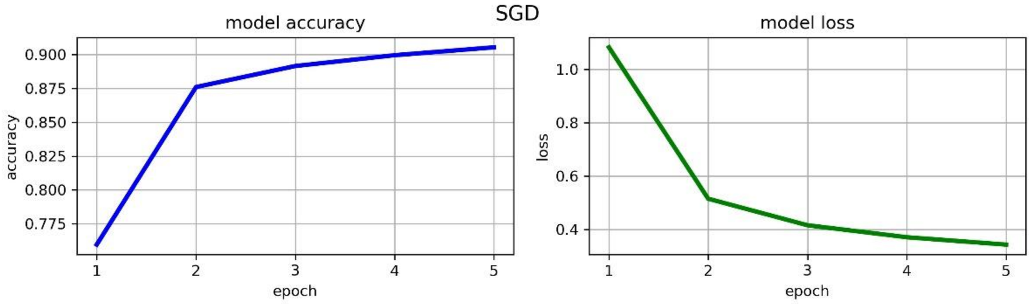
使用以下參數配置：

```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
rmsprop_hist = network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

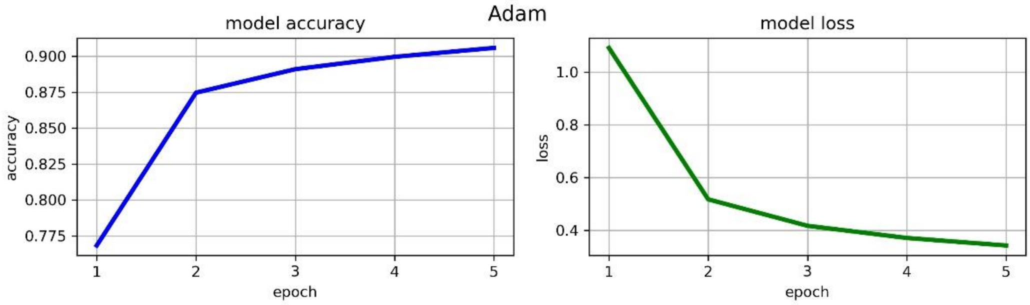
1. RMSprop



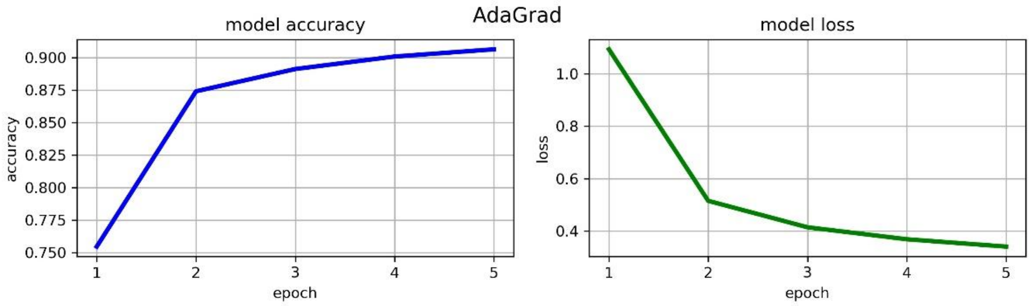
2. SGD



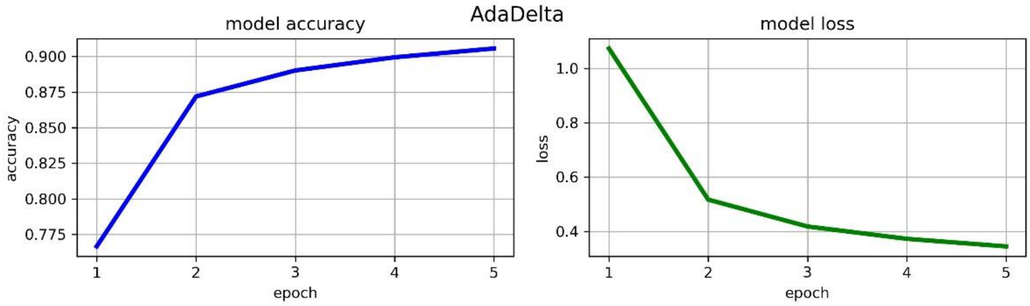
3. Adam



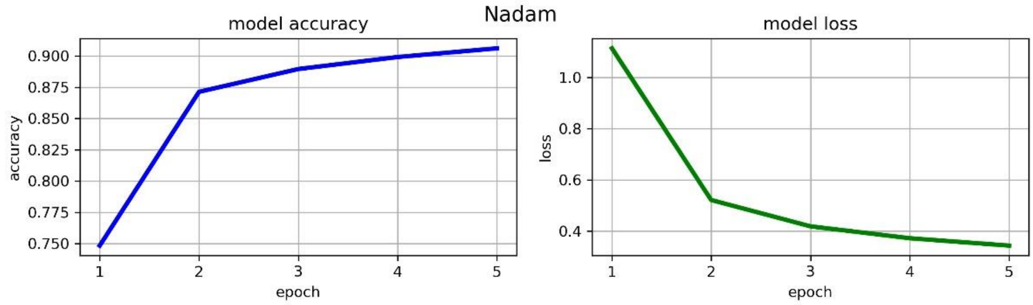
4. AdaGrad



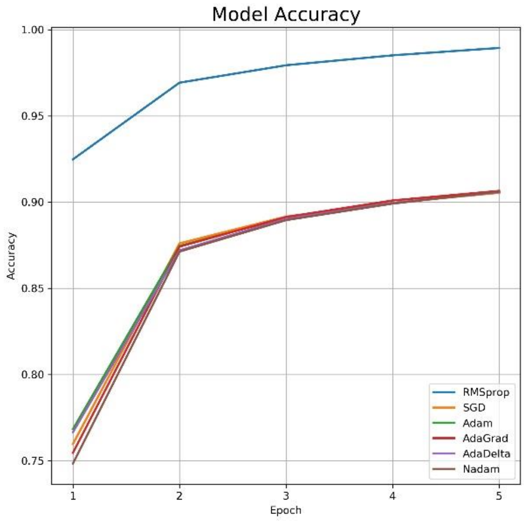
5. AdaDelta



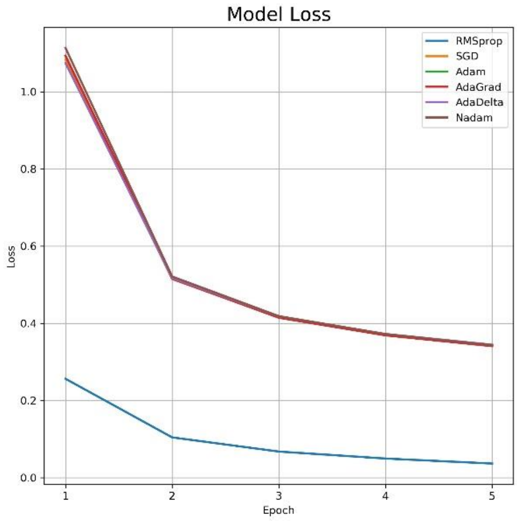
6. Nadam



7. Accuracy of different optimizer



8. Loss of different optimizer



Conclusions

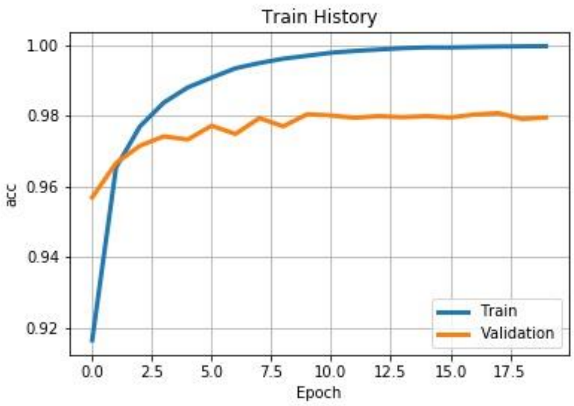
- 以手寫辨識資料集來說，使用 RMSprop 優化器準確度最高、損失最少。
- 其餘優化器在相似的情況下表現差不多。
- RMSprop 是一種自我調整學習速率的方法。
- 任何一種優化器，準確度隨著 Epoch 增加，損失隨著 Epoch 減少。
- 自適應學習率方法 Adagrad、AdaDelta、RMSprop、Adam 幾乎很快就可以達到收斂的效果。

◆ 數據訓練的總輪數 Epoch

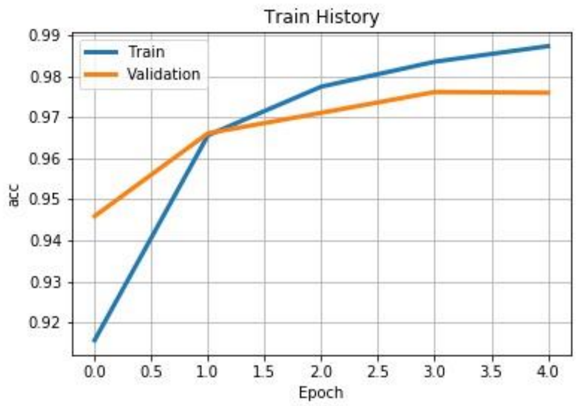
使用以下參數配置：

```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
train_history = network.fit(train_images, train_labels, validation_split=0.2, epochs=5, batch_size=128)
```

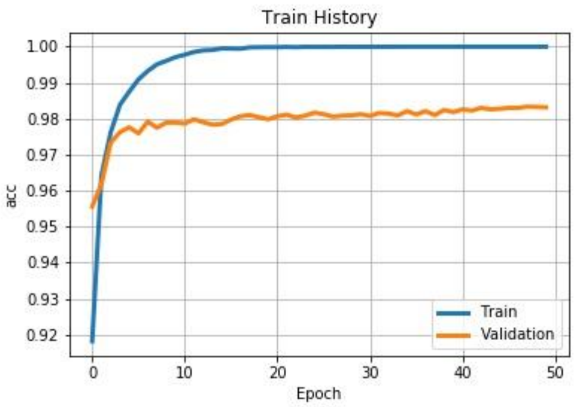
1. epochs=5



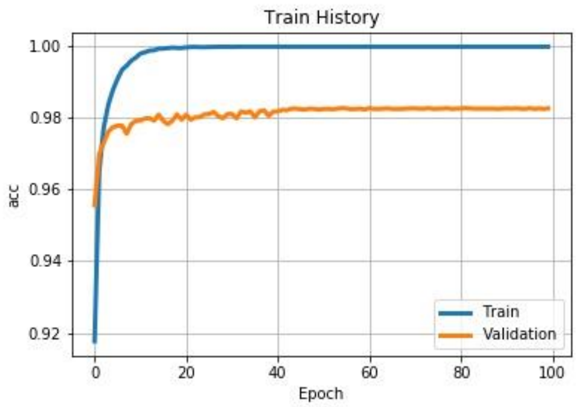
2. epochs=20



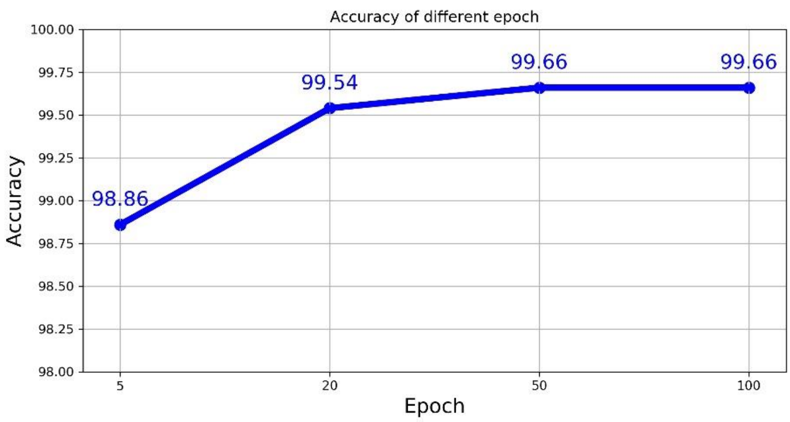
3. epochs=50



4. epochs=100



5. Accuracy of different epochs



Conclusions

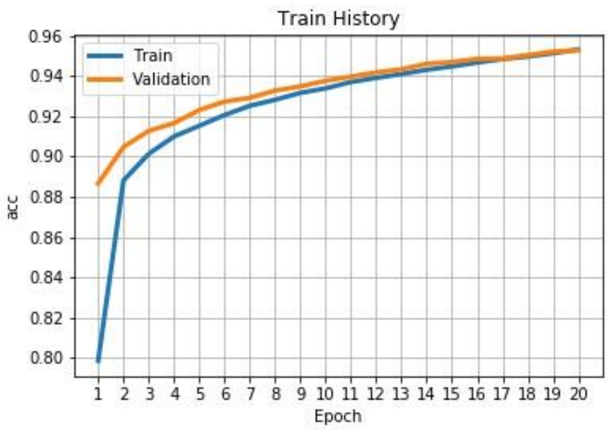
- 隨著 Epoch 的增加，準確率也逐漸提升。
- Epoch 5、Epoch 20、Epoch 50、Epoch 100 準確率都表現得不錯。
- 資料的多樣性會影響合適的 epoch 的數量。

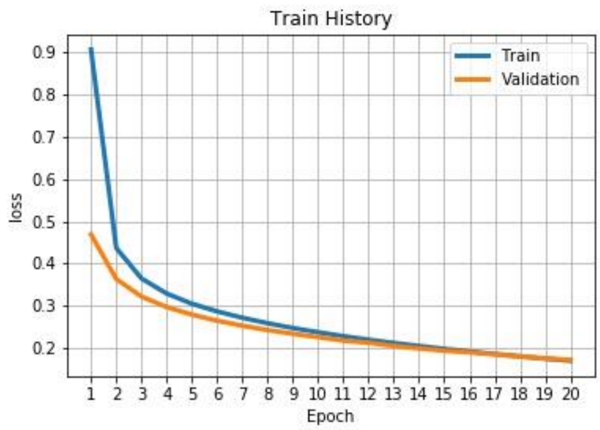
◆ 批次處理數據量 Batch Size

使用以下參數配置：

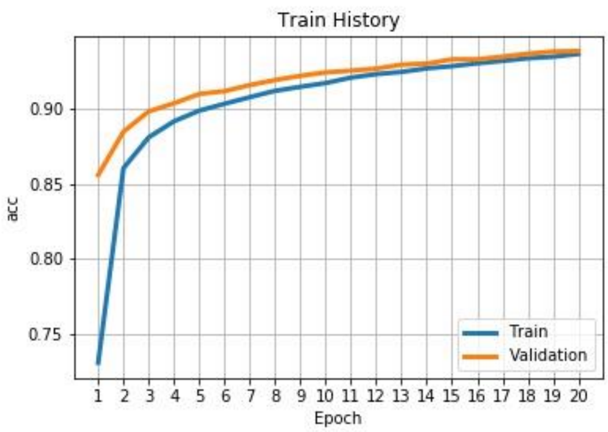
```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history = network.fit(train_images, train_labels, validation_split=0.2, epochs=5, batch_size=64)
```

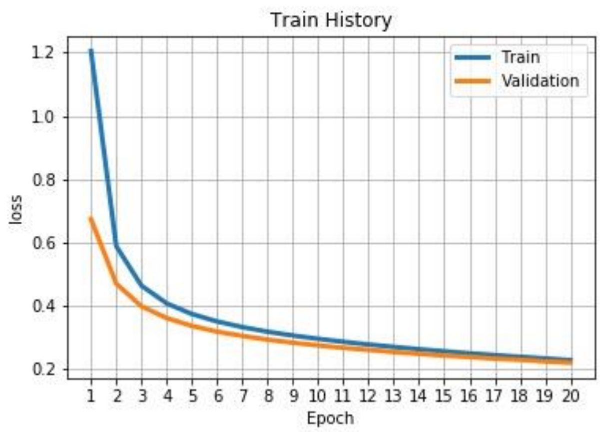
1. batch_size=64



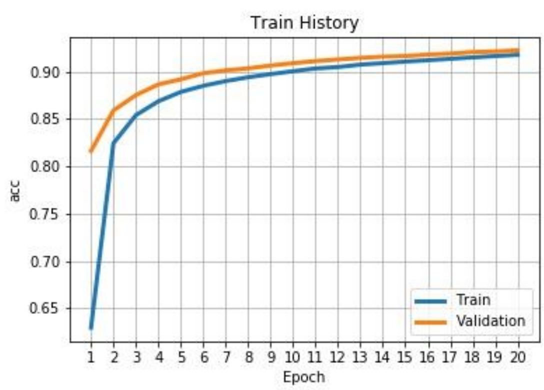


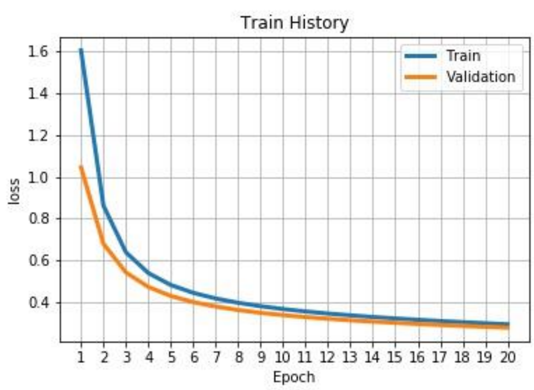
2. batch_size=128



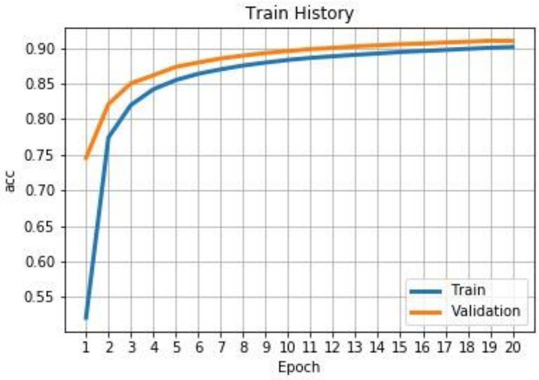


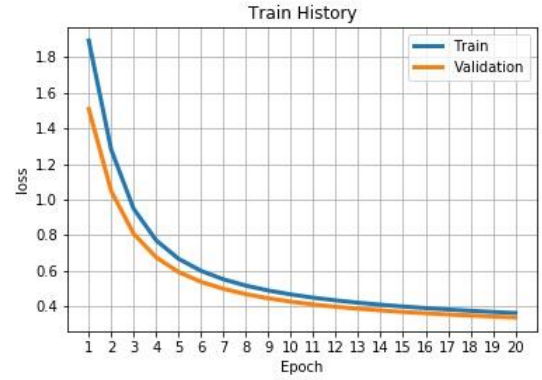
3. batch_size=256



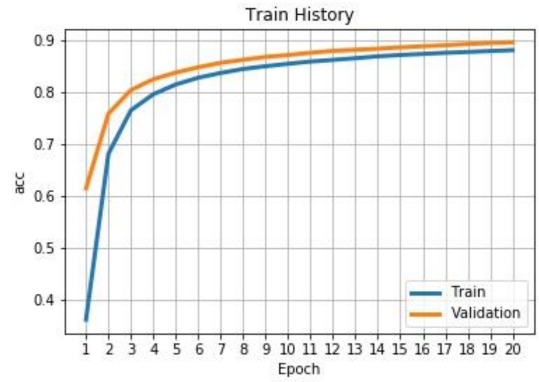


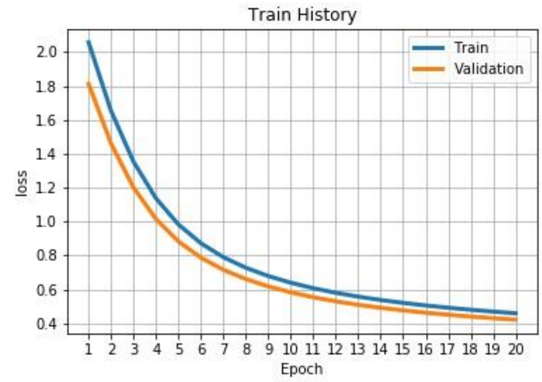
4. batch_size=512



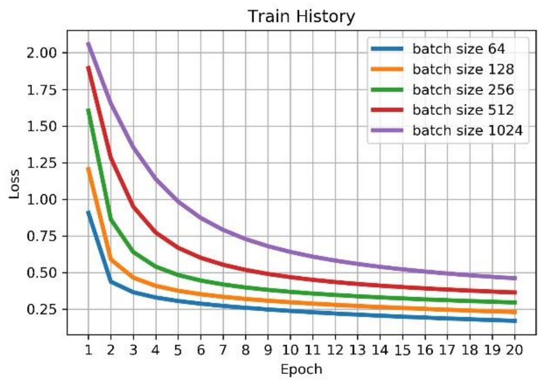


5. batch_size=1024

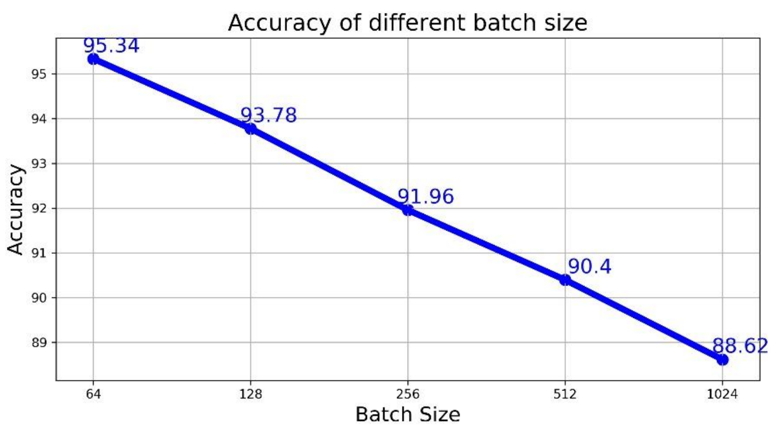




6. Accuracy/Loss of different batch size



7. Accuracy of different batch size



Conclusions

- 隨著 Batch Size 的增加，準確率逐漸下降，損失也逐漸下降。
- Batch Size 64、Batch Size 128、Batch Size 256、Batch Size 512、Batch Size 1024，以 Batch Size 64 表現最好，準確率高達 95.34%。
- 對於大的資料集，不能使用全批次(大 Batch Size)，因為會得到更差的結果。

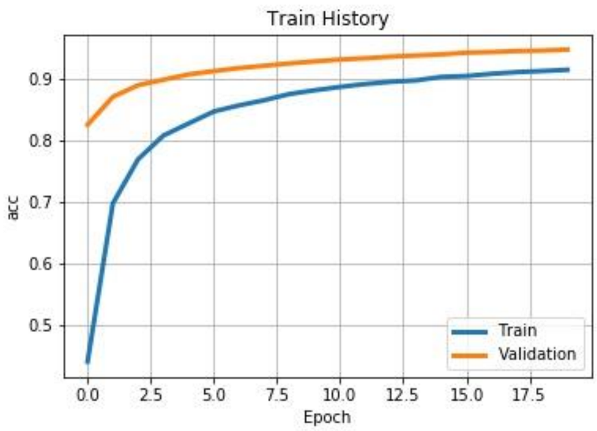
◆ 隱藏層 Hidden Layer

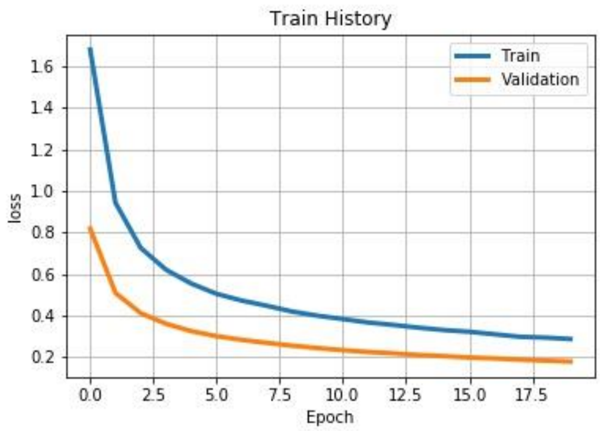
使用以下參數配置：

```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history = network.fit(train_images,train_labels, validation_split=0.2, epochs=20, batch_size=128)
```

1. 1 hidden layer

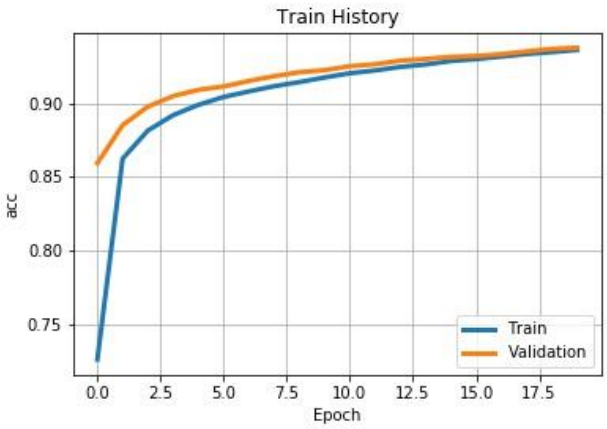
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

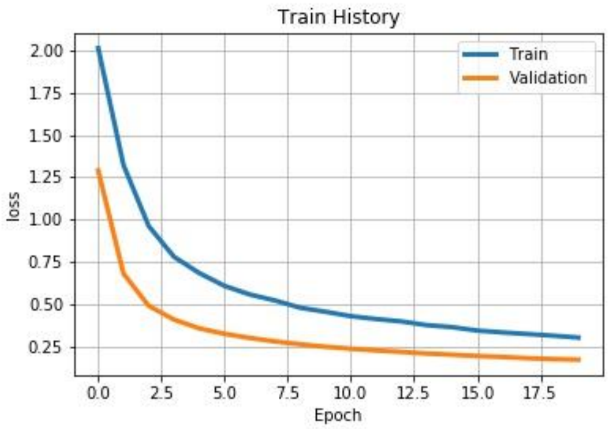




2. 2 hidden layers

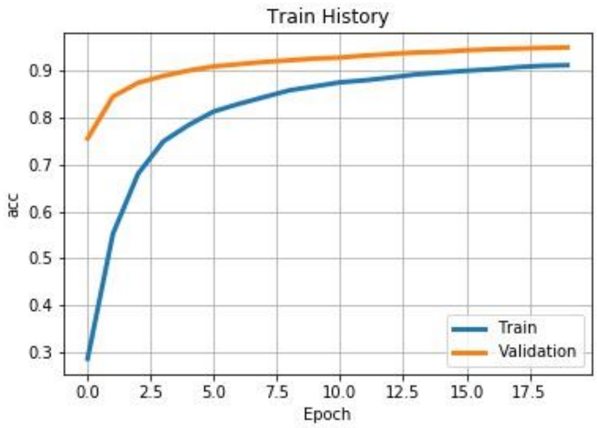
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_3 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664     
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290      
=================================================================
Total params: 468,874
Trainable params: 468,874
Non-trainable params: 0
_________________________________________________________________
```

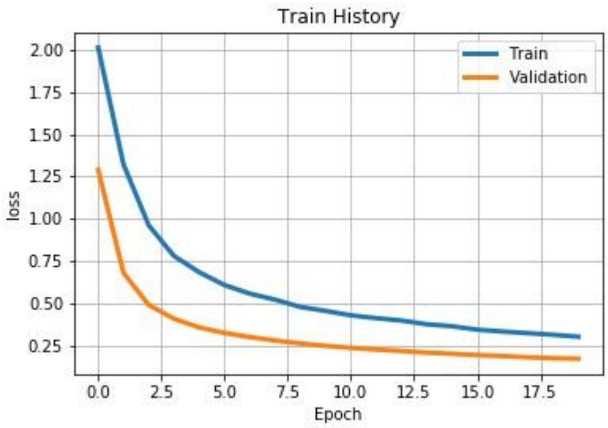




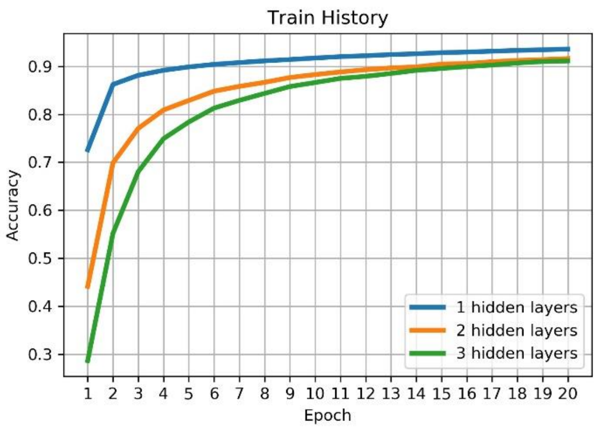
3. 3 hidden layers

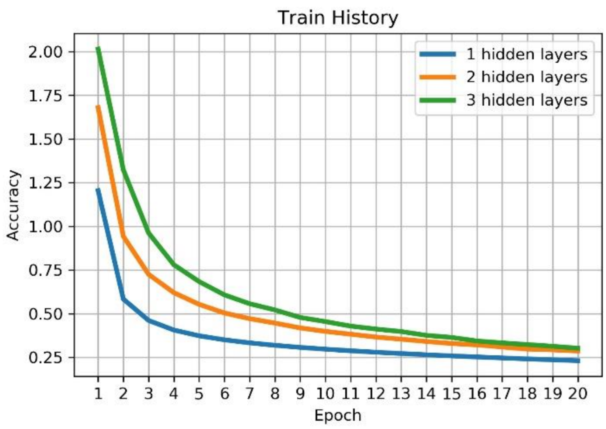
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_6 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_7 (Dense)              (None, 256)               131328    
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_8 (Dense)              (None, 128)               32896     
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1290      
=================================================================
Total params: 567,434
Trainable params: 567,434
Non-trainable params: 0
_________________________________________________________________
```



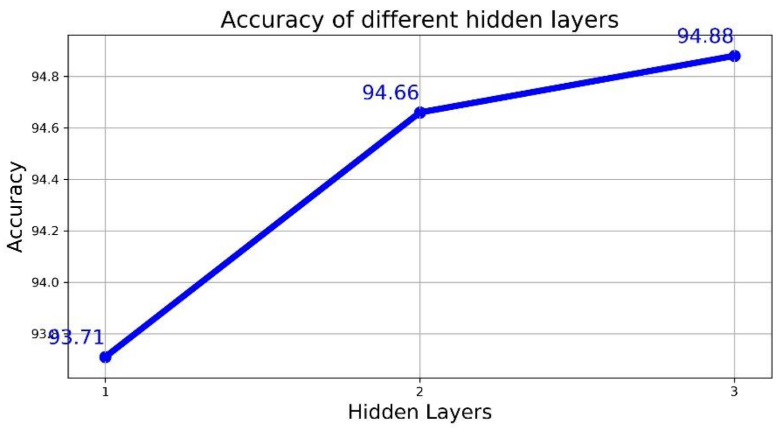


4. Accuracy/Loss of different hidden layers





5. Accuracy of different hidden layers



Conclusions

- 隨著 Hidden Layers 的增加，準確率逐漸上升，損失也逐漸下降。
- 1 Hidden Layer、2 Hidden Layers、3 Hidden Layers，以 3 Hidden Layers 表現最好，準確率高達 94.88%。
- 輸入（可見層）和第一個隱藏層之間加入一層 Dropout。丟棄率設為 50%，就是說每輪迭代時每 2 個輸入值就會被隨機拋棄 1 個，準確率完美地提升。
- Dropout 被用於兩個隱藏層之間和隱藏層與輸出層之間。丟棄率同樣設為 50%，準確率效果更好。

- 實驗過程中，以 Dropout 控制在 20%~50%，可從 20%開始嘗試。如果比例太低則起不到效果，比例太高則會導致模型的欠學習。

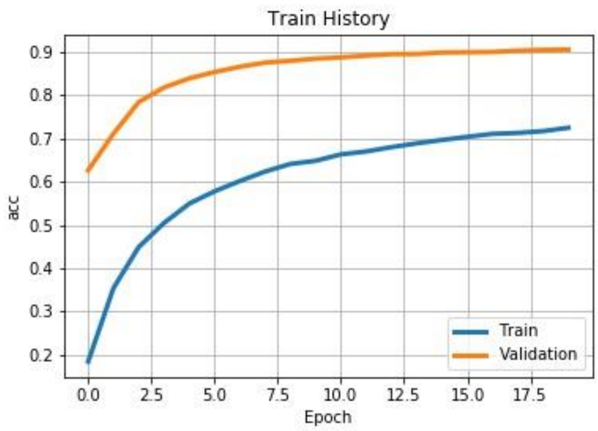
◆ 隱藏層節點 Hidden Node

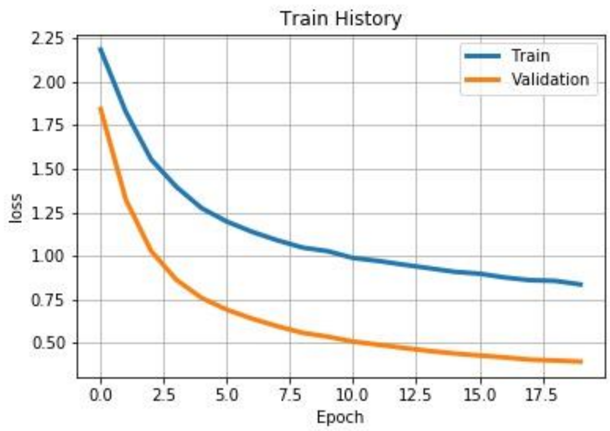
使用以下參數配置：

```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
train_history = network.fit(train_images,train_labels, validation_split=0.2, epochs=20, batch_size=128)
```

1. 32 hidden nodes

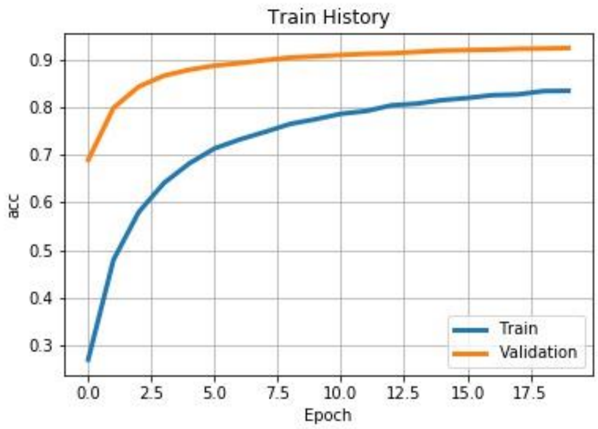
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 32)                25120
_________________________________________________________________
dropout_7 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_11 (Dense)             (None, 32)                1056
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                330
=================================================================
Total params: 26,506
Trainable params: 26,506
Non-trainable params: 0
_________________________________________________________________
```

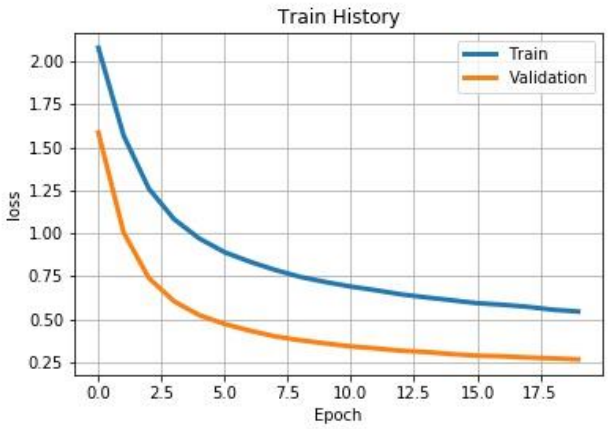




2. 64 hidden nodes

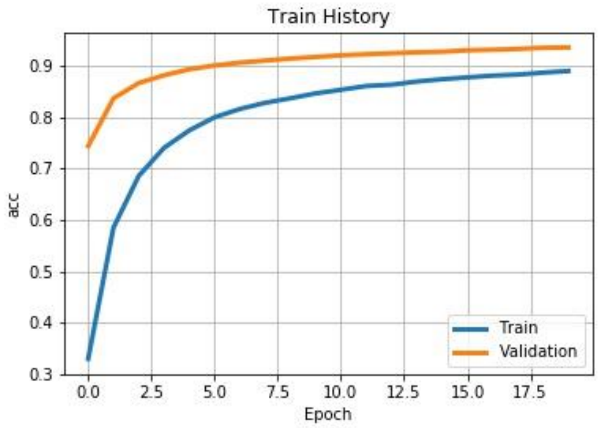
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                50240
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 55,050
Trainable params: 55,050
Non-trainable params: 0
_________________________________________________________________
```

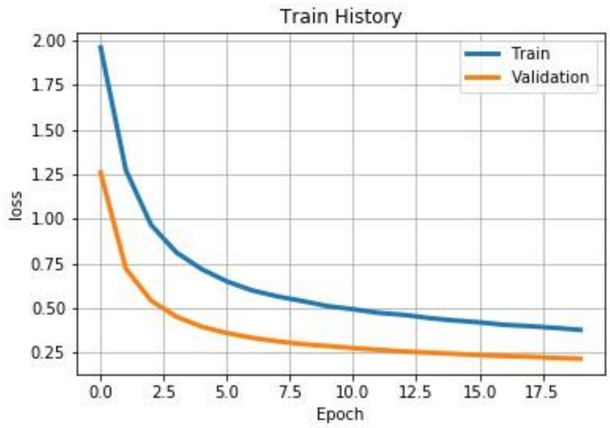




3. 128 hidden nodes

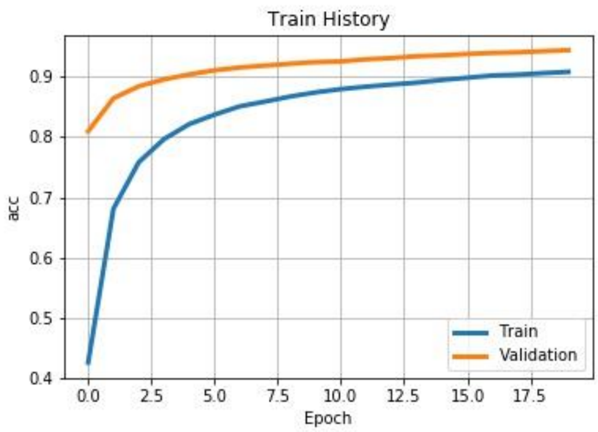
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 128)               100480
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

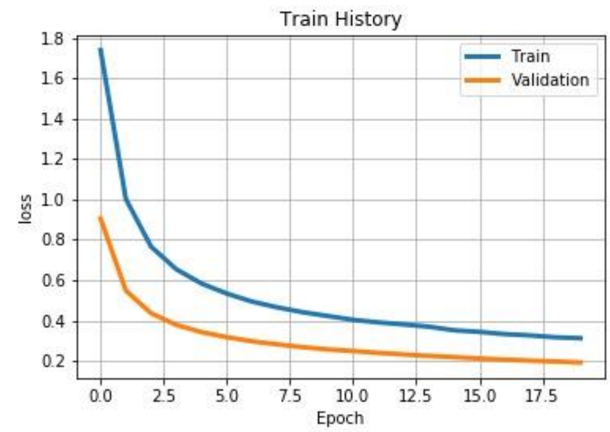




4. 256 hidden nodes

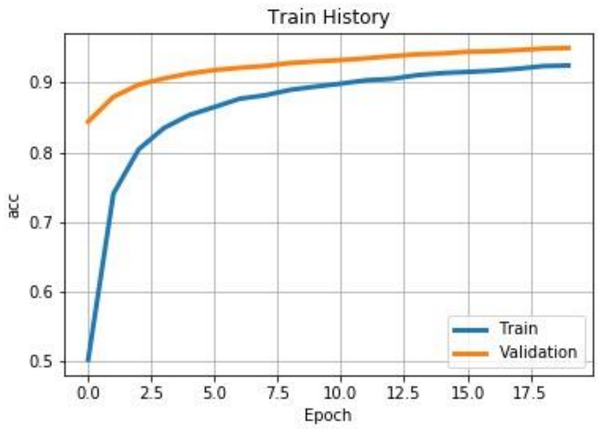
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 256)               200960
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                2570
=================================================================
Total params: 269,322
Trainable params: 269,322
Non-trainable params: 0
_________________________________________________________________
```

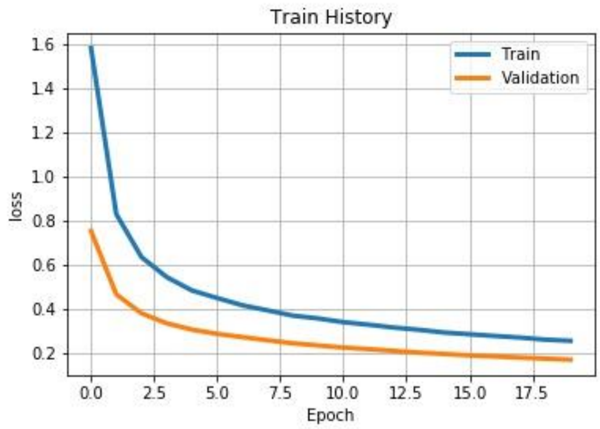




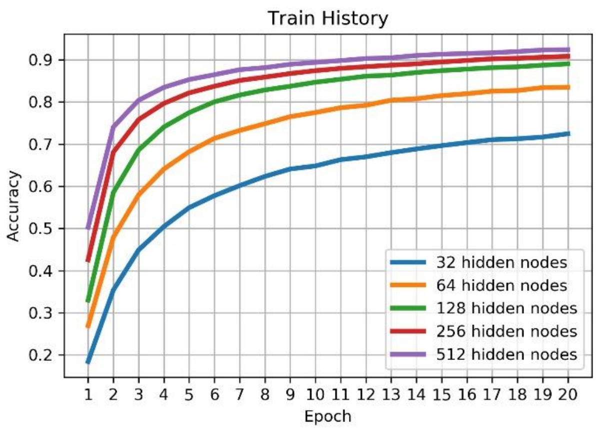
5. 512 hidden nodes

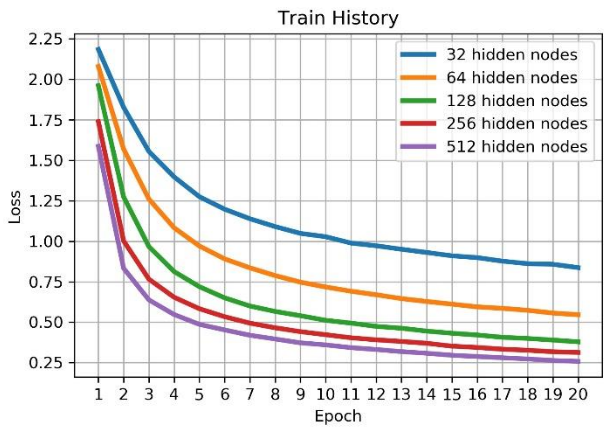
```
[Info] Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 512)               401920
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_14 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_15 (Dense)             (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```



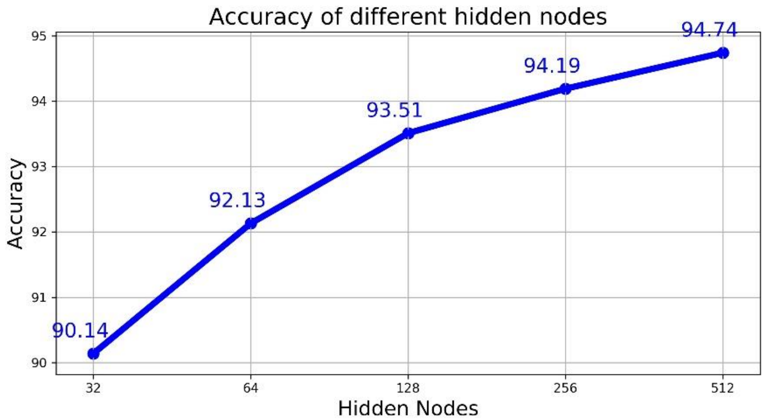


6. Accuracy/Loss of different hidden nodes





7. Accuracy of different hidden nodes



Conclusions

- 隨著 Hidden Nodes 的增加，準確率逐漸上升，損失也逐漸下降。
- 32 Hidden Nodes、64 Hidden Nodes、128 Hidden Nodes、256 Hidden Nodes、512 Hidden Nodes，以 512 Hidden Nodes 表現最好，準確率高達 94.74%。