In [1]:
```
#從 keras 的 datasets 匯入 mnist 資料集
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

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

In [2]:
```
#神經網路架構
from keras import models
from keras import layers

network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```

```
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [3]:
```
#編譯步驟
network.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [4]:
```
#準備圖片資料
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
```

In [5]:
```
#準備標籤
from keras.utils import to_categorical

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

In [6]:
```
#檢驗神經網路模型
#加入驗證集validation_data
history = network.fit(train_images,
                      train_labels,
                      epochs=20,
                      batch_size=128,
                      validation_data=(test_images, test_labels))
```

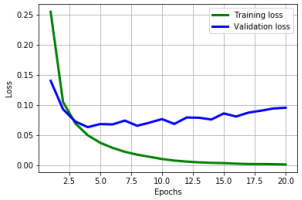
```
WARNING:tensorflow:From D:\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
60000/60000 [==============================] - 5s 77us/step - loss: 0.2547 - acc: 0.9266 - val_loss: 0.1402 - val_acc: 0.9596
Epoch 2/20
60000/60000 [==============================] - 3s 47us/step - loss: 0.1051 - acc: 0.9688 - val_loss: 0.0929 - val_acc: 0.9717
Epoch 3/20
60000/60000 [==============================] - 3s 46us/step - loss: 0.0692 - acc: 0.9789 - val_loss: 0.0724 - val_acc: 0.9784
Epoch 4/20
60000/60000 [==============================] - 3s 47us/step - loss: 0.0497 - acc: 0.9852 - val_loss: 0.0632 - val_acc: 0.9807
Epoch 5/20
60000/60000 [==============================] - 3s 47us/step - loss: 0.0373 - acc: 0.9887 - val_loss: 0.0682 - val_acc: 0.9802
Epoch 6/20
60000/60000 [==============================] - 3s 49us/step - loss: 0.0288 - acc: 0.9910 - val_loss: 0.0677 - val_acc: 0.9802
Epoch 7/20
60000/60000 [==============================] - 3s 47us/step - loss: 0.0222 - acc: 0.9933 - val_loss: 0.0740 - val_acc: 0.9789
Epoch 8/20
60000/60000 [==============================] - 3s 48us/step - loss: 0.0174 - acc: 0.9950 - val_loss: 0.0654 - val_acc: 0.9803
Epoch 9/20
60000/60000 [==============================] - 3s 48us/step - loss: 0.0138 - acc: 0.9960 - val_loss: 0.0707 - val_acc: 0.9805
Epoch 10/20
60000/60000 [==============================] - 3s 49us/step - loss: 0.0102 - acc: 0.9970 - val_loss: 0.0765 - val_acc: 0.9799
Epoch 11/20
60000/60000 [==============================] - 3s 49us/step - loss: 0.0076 - acc: 0.9979 - val_loss: 0.0685 - val_acc: 0.9820
Epoch 12/20
60000/60000 [==============================] - 3s 47us/step - loss: 0.0059 - acc: 0.9985 - val_loss: 0.0792 - val_acc: 0.9817
Epoch 13/20
60000/60000 [==============================] - 3s 50us/step - loss: 0.0046 - acc: 0.9988 - val_loss: 0.0787 - val_acc: 0.9830
Epoch 14/20
60000/60000 [==============================] - 3s 48us/step - loss: 0.0038 - acc: 0.9990 - val_loss: 0.0759 - val_acc: 0.9829
Epoch 15/20
60000/60000 [==============================] - 3s 47us/step - loss: 0.0034 - acc: 0.9992 - val_loss: 0.0859 - val_acc: 0.9816
Epoch 16/20
60000/60000 [==============================] - 3s 48us/step - loss: 0.0025 - acc: 0.9994 - val_loss: 0.0808 - val_acc: 0.9836
Epoch 17/20
60000/60000 [==============================] - 3s 48us/step - loss: 0.0019 - acc: 0.9996 - val_loss: 0.0873 - val_acc: 0.9828
Epoch 18/20
60000/60000 [==============================] - 3s 49us/step - loss: 0.0019 - acc: 0.9996 - val_loss: 0.0905 - val_acc: 0.9825
Epoch 19/20
60000/60000 [==============================] - 3s 50us/step - loss: 0.0016 - acc: 0.9996 - val_loss: 0.0941 - val_acc: 0.9819
Epoch 20/20
60000/60000 [==============================] - 3s 46us/step - loss: 0.0012 - acc: 0.9997 - val_loss: 0.0954 - val_acc: 0.9821
```

In [7]:
```
#評估測試資料的表現
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test_acc:', test_acc)
```

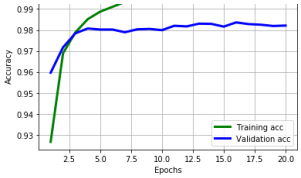
```
10000/10000 [==============================] - 0s 38us/step
test_acc: 0.9821
```

In [8]:
```
#繪製訓練與驗證的損失函數
import matplotlib.pyplot as plt
%matplotlib inline

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']


epochs = range(1, len(loss_values)+ 1)

plt.plot(epochs, loss_values, 'g', label='Training loss', linewidth=3)
plt.plot(epochs, val_loss_values, 'b', label='Validation loss', linewidth=3)
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.grid(True)
plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x29723610470>



In [9]:
```
#繪製訓練和驗證的準確度
plt.clf()
acc = history_dict['acc']
val_acc = history_dict['val_acc']

plt.plot(epochs, acc, 'g', label='Training acc', linewidth=3)
plt.plot(epochs, val_acc, 'b', label='Validation acc', linewidth=3)
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
```


0.99
0.98
0.97
0.96
0.95
0.94
0.93
Accuracy
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs
Training acc
Validation acc

In [ ]:
```
1
```