引用 **Keras** 的 **Mnist** 套件，下載已建立好的手寫數字集

下載需一些時間，建議使用自己網路下載

In [1]:

```
from keras.datasets import mnist
```

```
Using TensorFlow backend.
C:\Users\KAI\Anaconda3\envs\NN\lib\site-packages\tensorflow\python\framework\d
types.py:523: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
is deprecated; in a future version of numpy, it will be understood as (type,
(1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\KAI\Anaconda3\envs\NN\lib\site-packages\tensorflow\python\framework\d
types.py:524: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
is deprecated; in a future version of numpy, it will be understood as (type,
(1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\KAI\Anaconda3\envs\NN\lib\site-packages\tensorflow\python\framework\d
types.py:525: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
is deprecated; in a future version of numpy, it will be understood as (type,
(1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\KAI\Anaconda3\envs\NN\lib\site-packages\tensorflow\python\framework\d
types.py:526: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
is deprecated; in a future version of numpy, it will be understood as (type,
(1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\KAI\Anaconda3\envs\NN\lib\site-packages\tensorflow\python\framework\d
types.py:527: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
is deprecated; in a future version of numpy, it will be understood as (type,
(1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\KAI\Anaconda3\envs\NN\lib\site-packages\tensorflow\python\framework\d
types.py:532: FutureWarning: Passing (type, 1) or '1type' as a synonym of type
is deprecated; in a future version of numpy, it will be understood as (type,
(1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:

```
(train_feature, train_label), (test_feature, test_label) = mnist.load_data()
```

In [3]:

```
print(len(train_feature), len(train_label))
```

```
60000 60000
```

In [4]:

```
print(train_feature.shape, train_label.shape)
```

```
(60000, 28, 28) (60000,)
```

引用 **matplotlib** 套件

%matplotlib inline 讓 matplotlib 畫出來的圖可以在 Jupyter Notebook 上顯示

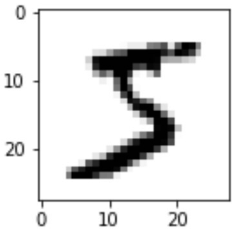
In [5]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

Matplotlib 測試，**show** 出單一張圖片

In [6]:
```
def show_image(image):
    fig=plt.gcf()
    fig.set_size_inches(2, 2) # 圖片大小
    plt.imshow(image, cmap='binary') # 黑白
    plt.show()
```

In [7]:
```
show_image(train_feature[0])
print(train_label[0])
```



```
5
```

一次畫多張圖片

一排畫五張，最多畫25張(可修改)
如果 predictions 裡面有東西，則會顯示 AI 預測的結果以及是否正確、label
反之，只 show 出 label

In [8]:
```
def show_images_labels_predictions(images, labels, predictions, start_id, num=10):
    plt.gcf().set_size_inches(12, 14)
    if num > 25: num = 25
    for i in range(0, num):
        ax = plt.subplot(5, 5, i+1)
        ax.imshow(images[start_id], cmap='binary')

        if len(predictions) > 0:
            title = 'ai = ' + str(predictions[start_id])
            title += (' (o)' if predictions[start_id] == labels[start_id] else ' (x)')
            title += '\nlabel = ' + str(labels[start_id])
        else:
            title = 'label = ' + str(labels[start_id])

        ax.set_title(title, fontsize = 12)
        ax.set_xticks([])
        ax.set_yticks([])
        start_id += 1
    plt.show()
```

In [9]:

```
show_images_labels_predictions(train_feature, train_label, [], 0, 10)
```

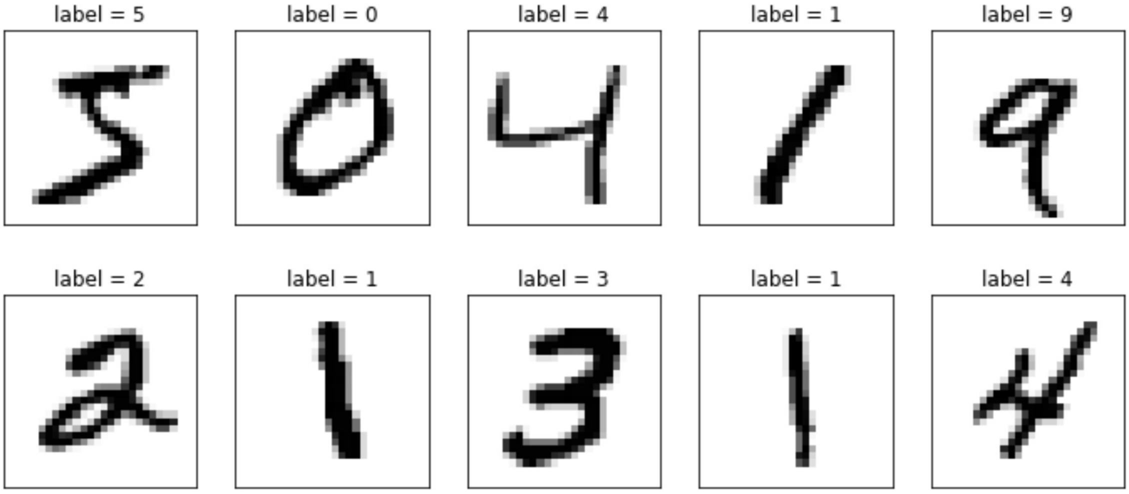


image 轉換

用 reshape() 函式將 28*28 的圖片轉成784個數字的一維向量
再以 astype 將每個數字都轉換維 float

In [10]:

```
train_feature_vector = train_feature.reshape(len(train_feature), 784).astype('float32')
test_feature_vector = test_feature.reshape(len(test_feature), 784).astype('float32')
```

In [11]:

```
print(train_feature_vector.shape, test_feature_vector.shape)
```

```
(60000, 784) (10000, 784)
```

```
In [12]: print(train_feature_vector[0])
```

```
[   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    3.   18.
   18.   18.  126.  136.  175.   26.  166.  255.  247.  127.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.   30.   36.   94.  154.  170.  253.
  253.  253.  253.  253.  225.  172.  253.  242.  195.   64.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.   49.  238.  253.  253.  253.  253.  253.
  253.  253.  253.  251.   93.   82.   82.   56.   39.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.   18.  219.  253.  253.  253.  253.  253.
  198.  182.  247.  241.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.   80.  156.  107.  253.  253.  205.
   11.    0.   43.  154.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.   14.    1.  154.  253.   90.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  139.  253.  190.
    2.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   11.  190.  253.
   70.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   35.  241.
  225.  160.  108.    1.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   81.
  240.  253.  253.  119.   25.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   45.  186.  253.  253.  150.   27.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.   16.   93.  252.  253.  187.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.  249.  253.  249.   64.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   46.  130.  183.  253.  253.  207.    2.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   39.  148.
  229.  253.  253.  253.  250.  182.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   24.  114.  221.  253.
  253.  253.  253.  201.   78.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.   23.   66.  213.  253.  253.  253.
  253.  198.   81.    2.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.   18.  171.  219.  253.  253.  253.  253.  195.
   80.    9.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.   55.  172.  226.  253.  253.  253.  253.  244.  133.   11.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.  136.  253.  253.  253.  212.  135.  132.   16.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
```

```
In [13]: print(test_feature_vector[0])
```

```
[   0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.   84.  185.  159.  151.   60.   36.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.  222.  254.  254.  254.  254.  241.  198.  198.
  198.  198.  198.  198.  198.  198.  170.   52.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.   67.  114.   72.  114.  163.  227.  254.  225.
  254.  254.  254.  250.  229.  254.  254.  140.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   17.   66.   14.
   67.   67.   67.   59.   21.  236.  254.  106.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.   83.  253.  209.   18.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.   22.  233.  255.   83.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.  129.  254.  238.   44.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.   59.  249.  254.   62.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.  133.  254.  187.    5.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    9.  205.  248.   58.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.  126.  254.  182.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
   75.  251.  240.   57.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   19.
  221.  254.  166.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    3.  203.
  254.  219.   35.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   38.  254.
  254.   77.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   31.  224.  254.
  115.    1.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  133.  254.  254.
   52.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   61.  242.  254.  254.
   52.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  121.  254.  254.  219.
   40.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  121.  254.  207.   18.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.]
```

image 標準化

將 0-255 的數字，除以 255 得到 0-1 之間的浮點數，稱為標準化
標準化之後可以提高模型預測的準確度，增加訓練效率

In [14]:

```
train_feature_normalize = train_feature_vector/255
test_feature_normalize = test_feature_vector/255
```

In [15]:

```
print(train_label[0:5])
```

```
[5 0 4 1 9]
```

Label 資料預處理

使用 np_utils.to_categorical() 進行 one-hot encoding
用來增加模型效率

In [16]:

```
from keras.utils import np_utils
train_label_onehot = np_utils.to_categorical(train_label)
test_label_onehot = np_utils.to_categorical(test_label)
```

In [17]:

```
print(train_label_onehot[0:5])
```

```
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

建立模型

匯入 Sequential 可以用來建立模型
Dense 用來建立隱藏層
units: 神經元數目
input_dim: 輸入神經元的數目(第一層神經元必須寫)
kernel_initializer: 'normal'表示使用常態分佈的亂數
activation: 激勵函式，如果用於分類問題，輸出層多為 'softmax'

In [18]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [19]:

```
model = Sequential()
model.add(Dense(units=256,
                input_dim=784,
                kernel_initializer='normal',
                activation='relu'))
model.add(Dense(units=10,
                kernel_initializer='normal',
                activation='softmax'))
```

訓練模型

loss: 損失函式
optimizer: 優化器
metrics: 'accuracy'為準

In [20]:

```
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['accuracy'])
```

進行訓練

x: 特徵值
y: 標籤(答案)
validation_split: 驗證資料百分比
epochs: 訓練次數
batch_size: 設定每批次讀取多少筆資料
verbose: 設定是否顯示訓練過程，0不顯示，1詳細顯示，2簡易顯示

In [21]:
```
train_history = model.fit(x=train_feature_normalize,
                          y=train_label_onehot,
                          validation_split=0.2,
                          epochs=10,
                          batch_size=200,
                          verbose=2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/10
 - 1s - loss: 0.0189 - acc: 0.8840 - val_loss: 0.0099 - val_acc: 0.9382
Epoch 2/10
 - 1s - loss: 0.0086 - acc: 0.9459 - val_loss: 0.0073 - val_acc: 0.9547
Epoch 3/10
 - 1s - loss: 0.0063 - acc: 0.9615 - val_loss: 0.0061 - val_acc: 0.9631
Epoch 4/10
 - 1s - loss: 0.0050 - acc: 0.9694 - val_loss: 0.0053 - val_acc: 0.9666
Epoch 5/10
 - 1s - loss: 0.0040 - acc: 0.9766 - val_loss: 0.0049 - val_acc: 0.9694
Epoch 6/10
 - 1s - loss: 0.0033 - acc: 0.9813 - val_loss: 0.0045 - val_acc: 0.9715
Epoch 7/10
 - 1s - loss: 0.0028 - acc: 0.9844 - val_loss: 0.0042 - val_acc: 0.9730
Epoch 8/10
 - 1s - loss: 0.0024 - acc: 0.9865 - val_loss: 0.0042 - val_acc: 0.9725
Epoch 9/10
 - 1s - loss: 0.0021 - acc: 0.9891 - val_loss: 0.0039 - val_acc: 0.9741
Epoch 10/10
 - 1s - loss: 0.0018 - acc: 0.9909 - val_loss: 0.0038 - val_acc: 0.9746
```

In [22]:
```
scores = model.evaluate(test_feature_normalize, test_label_onehot)
print('\n準確率 = ', scores[1])
```

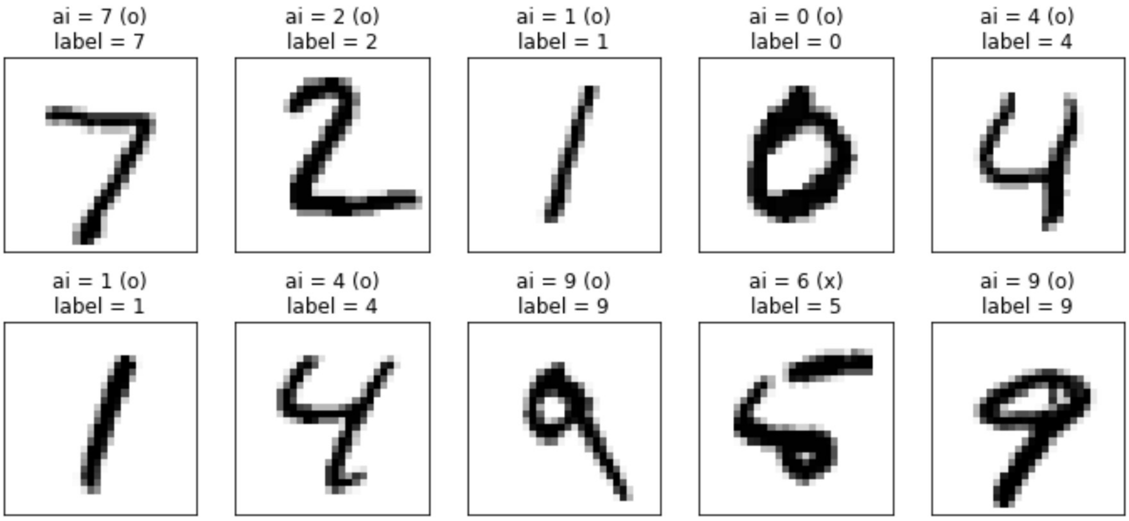
```
10000/10000 [==============================] - 0s 11us/step

準確率 =  0.9782
```

評估準確率並印出結果

In [23]:
```
prediction = model.predict_classes(test_feature_normalize)
```

```
In [24]: show_images_labels_predictions(test_feature, test_label, prediction, 0)
```



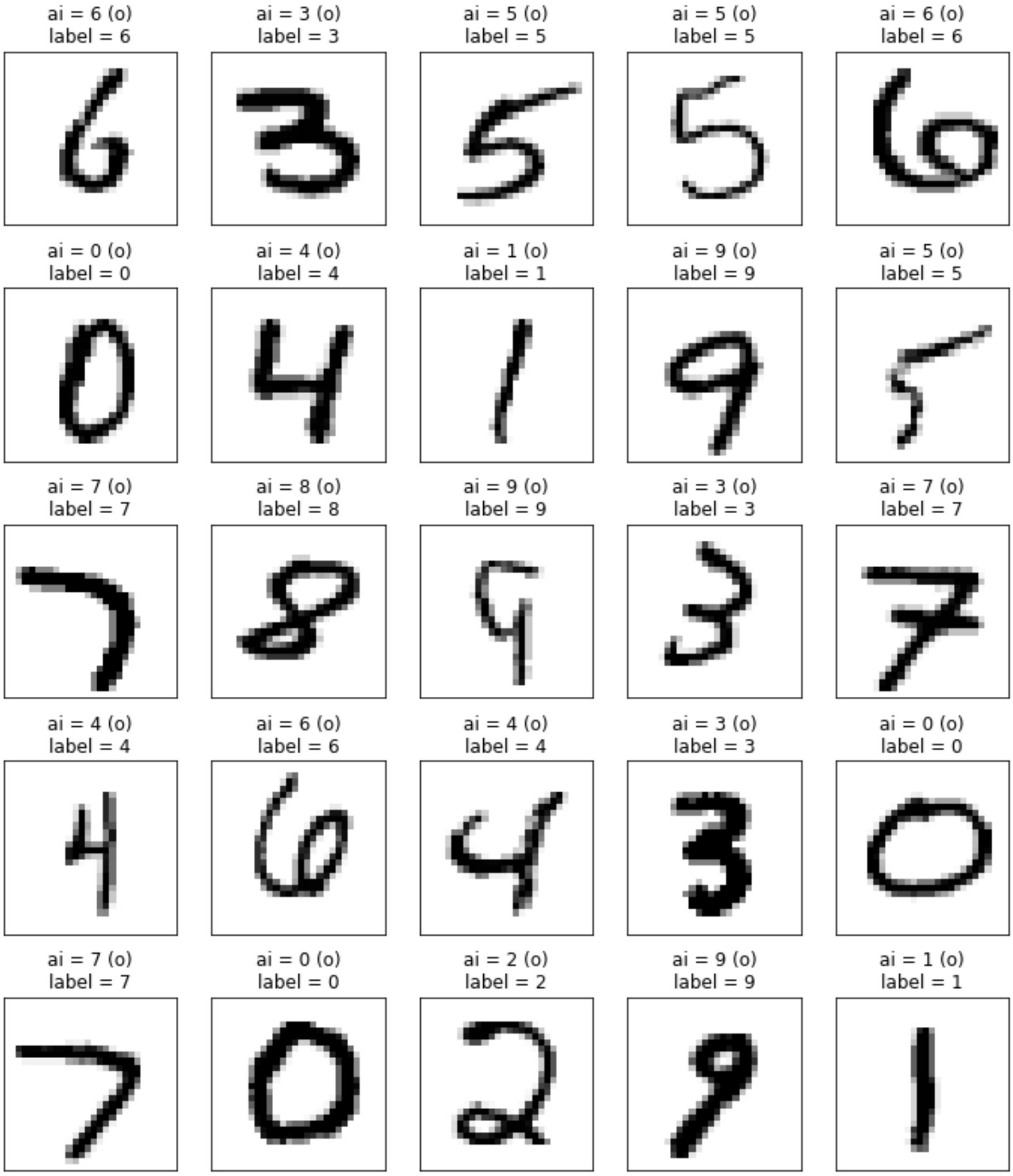
儲存訓練好的模型

```
In [25]: model.save('Mnist_mlp_model.h5')
         print('Mnist_mlp_model.h5 儲存完畢！')
         del model
```

```
Mnist_mlp_model.h5 儲存完畢！
```

讀取訓練好的模型

In [26]:
```
from keras.models import load_model
model = load_model('Mnist_mlp_model.h5')
prediction = model.predict_classes(test_feature_normalize)
show_images_labels_predictions(test_feature, test_label, prediction, 50, num=30)
```



In [ ]:

In [ ]: