How to use Google Colab

MNIST, Convolutional Neural Network (CNN)

Step - 3

MNIST由手寫阿拉伯數字組成，包含60,000個訓練樣本和10,000個測試樣本。

data from: https://keras.io/datasets/#mnist-database-of-handwritten-digits (https://keras.io/datasets/#mnist-database-of-handwritten-digits)

code modified from: TensorFlow+Keras[深度學習]人工智慧實務應用 / 林大貴

(1) Import the data from Keras

In [0]:

```
from keras.utils import np_utils
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
np.random.seed(3)
```

In [0]:

```
# read in the file
from numpy import load

data = load('mnist.npz')
lst = data.files
print(lst)
```

```
['x_test', 'x_train', 'y_train', 'y_test']
```

In [0]:

```
x_test_image  = data['x_test']
x_train_image = data['x_train']
y_test_label  = data['y_test']
y_train_label = data['y_train']

print(x_train_image.shape)
print(y_train_label.shape)
print(x_test_image.shape)
print(y_test_label.shape)
```

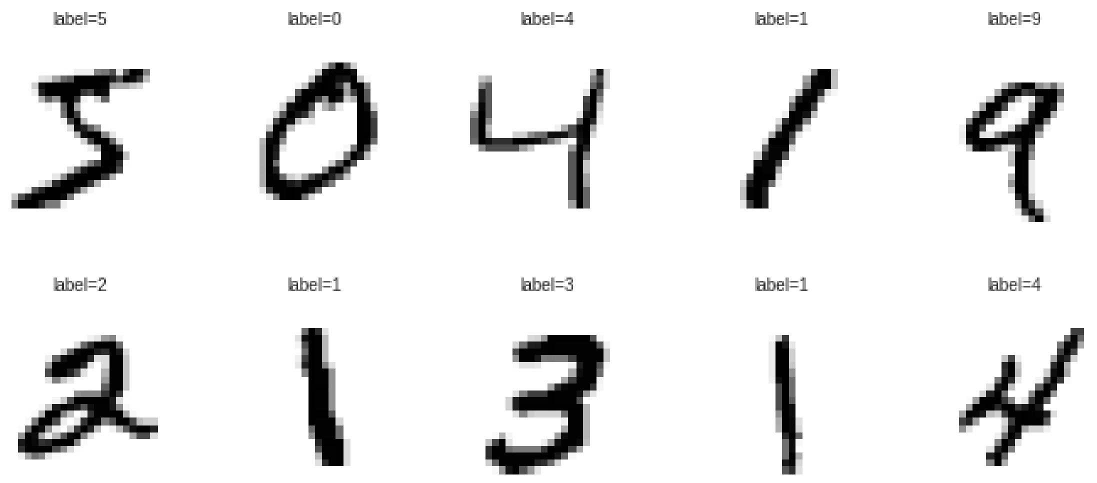
```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

(2) View the first 10 images and labels

In [0]:

```
fig = plt.gcf()
fig.set_size_inches(12,14)

for i in range(0,10):
    ax=plt.subplot(5,5,1+i)
    ax.imshow(x_train_image[i], cmap='binary')
    title= "label=" +str(y_train_label[i])
    ax.set_title(title,fontsize=10)
    ax.set_xticks([]);ax.set_yticks([])
plt.show()
```



(3) Convert 2-D image to nx28x28x1 array, normalize the numbers

In [0]:

```
# convert 2-D 28x28 image to 4-D nx28x28x1  array

x_Train4D=x_train_image.reshape(x_train_image.shape[0],28,28,1).astype('float32')
x_Test4D=x_test_image.reshape(x_test_image.shape[0],28,28,1).astype('float32')
```

In [0]:

```
# normalize the image numbers to 0~1

x_Train4D_normalize = x_Train4D / 255
x_Test4D_normalize = x_Test4D / 255
print(x_Train4D_normalize.shape)
print(x_Test4D_normalize.shape)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

(4) Convert label number to one-hot encoding

In [0]:

```
# convert label numbers to one-hot encoding

y_TrainOneHot = np_utils.to_categorical(y_train_label)
y_TestOneHot = np_utils.to_categorical(y_test_label)
print(y_TrainOneHot.shape)
print(y_TestOneHot.shape)
```

```
(60000, 10)
(10000, 10)
```

(5) Use a Convolutional Neural Network

In [0]:

```
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D
```

In [0]:

```
model = Sequential()
```

In [0]:

```
model.add(Conv2D(filters=16,
                 kernel_size=(5,5),
                 padding='same',
                 input_shape=(28,28,1),
                 activation='relu'))
```

In [0]:

```
# Enable this cell in the second step

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=36,
                 kernel_size=(5,5),
                 padding='same',
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

In [0]:

```
model.add(Flatten())
```

In [0]:

```
# Enable this cell in the second step

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
```

In [0]:

```
model.add(Dense(10,activation='softmax'))
```

In [0]:

```
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 16)        416       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 16)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 36)        14436     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 36)          0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 7, 7, 36)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 1764)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               225920    
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290      
=================================================================
Total params: 242,062
Trainable params: 242,062
Non-trainable params: 0
_________________________________________________________________
None
```

(6) Model training

In [0]:

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

In [0]:

```
train_history=model.fit(x=x_Train4D_normalize,
                        y=y_TrainOneHot,validation_split=0.2,
                        epochs=50, batch_size=300,verbose=2)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/50
 - 3s - loss: 0.4867 - acc: 0.8459 - val_loss: 0.1025 - val_acc: 0.9692
Epoch 2/50
 - 3s - loss: 0.1449 - acc: 0.9568 - val_loss: 0.0668 - val_acc: 0.9786
Epoch 3/50
 - 3s - loss: 0.1045 - acc: 0.9690 - val_loss: 0.0581 - val_acc: 0.9822
Epoch 4/50
 - 3s - loss: 0.0877 - acc: 0.9742 - val_loss: 0.0485 - val_acc: 0.9843
Epoch 5/50
 - 3s - loss: 0.0727 - acc: 0.9782 - val_loss: 0.0428 - val_acc: 0.9878
Epoch 6/50
 - 3s - loss: 0.0634 - acc: 0.9807 - val_loss: 0.0407 - val_acc: 0.9883
Epoch 7/50
 - 3s - loss: 0.0555 - acc: 0.9837 - val_loss: 0.0390 - val_acc: 0.9890
Epoch 8/50
 - 3s - loss: 0.0498 - acc: 0.9846 - val_loss: 0.0350 - val_acc: 0.9899
Epoch 9/50
 - 3s - loss: 0.0462 - acc: 0.9854 - val_loss: 0.0347 - val_acc: 0.9903
Epoch 10/50
 - 3s - loss: 0.0423 - acc: 0.9867 - val_loss: 0.0372 - val_acc: 0.9890
Epoch 11/50
 - 3s - loss: 0.0367 - acc: 0.9884 - val_loss: 0.0334 - val_acc: 0.9909
Epoch 12/50
 - 3s - loss: 0.0366 - acc: 0.9888 - val_loss: 0.0360 - val_acc: 0.9888
Epoch 13/50
 - 3s - loss: 0.0339 - acc: 0.9886 - val_loss: 0.0316 - val_acc: 0.9907
Epoch 14/50
 - 3s - loss: 0.0315 - acc: 0.9896 - val_loss: 0.0340 - val_acc: 0.9901
Epoch 15/50
 - 3s - loss: 0.0292 - acc: 0.9906 - val_loss: 0.0321 - val_acc: 0.9910
Epoch 16/50
 - 3s - loss: 0.0294 - acc: 0.9909 - val_loss: 0.0309 - val_acc: 0.9912
Epoch 17/50
 - 3s - loss: 0.0280 - acc: 0.9912 - val_loss: 0.0323 - val_acc: 0.9906
Epoch 18/50
 - 3s - loss: 0.0251 - acc: 0.9916 - val_loss: 0.0323 - val_acc: 0.9910
Epoch 19/50
 - 3s - loss: 0.0229 - acc: 0.9928 - val_loss: 0.0317 - val_acc: 0.9919
Epoch 20/50
 - 3s - loss: 0.0218 - acc: 0.9933 - val_loss: 0.0309 - val_acc: 0.9918
Epoch 21/50
 - 3s - loss: 0.0224 - acc: 0.9928 - val_loss: 0.0306 - val_acc: 0.9913
Epoch 22/50
 - 3s - loss: 0.0210 - acc: 0.9926 - val_loss: 0.0305 - val_acc: 0.9917
Epoch 23/50
 - 3s - loss: 0.0204 - acc: 0.9931 - val_loss: 0.0285 - val_acc: 0.9918
Epoch 24/50
 - 3s - loss: 0.0192 - acc: 0.9935 - val_loss: 0.0295 - val_acc: 0.9924
Epoch 25/50
 - 3s - loss: 0.0196 - acc: 0.9935 - val_loss: 0.0337 - val_acc: 0.9908
Epoch 26/50
 - 3s - loss: 0.0174 - acc: 0.9941 - val_loss: 0.0314 - val_acc: 0.9925
Epoch 27/50
 - 3s - loss: 0.0170 - acc: 0.9944 - val_loss: 0.0307 - val_acc: 0.9924
Epoch 28/50
 - 3s - loss: 0.0160 - acc: 0.9946 - val_loss: 0.0312 - val_acc: 0.9921
Epoch 29/50
 - 3s - loss: 0.0161 - acc: 0.9947 - val_loss: 0.0339 - val_acc: 0.9912
Epoch 30/50
 - 3s - loss: 0.0164 - acc: 0.9943 - val_loss: 0.0319 - val_acc: 0.9919
```

```
Epoch 31/50
 - 3s - loss: 0.0153 - acc: 0.9947 - val_loss: 0.0339 - val_acc: 0.9915
Epoch 32/50
 - 3s - loss: 0.0155 - acc: 0.9947 - val_loss: 0.0342 - val_acc: 0.9913
Epoch 33/50
 - 3s - loss: 0.0146 - acc: 0.9951 - val_loss: 0.0319 - val_acc: 0.9918
Epoch 34/50
 - 3s - loss: 0.0139 - acc: 0.9951 - val_loss: 0.0370 - val_acc: 0.9918
Epoch 35/50
 - 3s - loss: 0.0134 - acc: 0.9955 - val_loss: 0.0373 - val_acc: 0.9921
Epoch 36/50
 - 3s - loss: 0.0142 - acc: 0.9951 - val_loss: 0.0308 - val_acc: 0.9926
Epoch 37/50
 - 3s - loss: 0.0130 - acc: 0.9953 - val_loss: 0.0330 - val_acc: 0.9922
Epoch 38/50
 - 3s - loss: 0.0120 - acc: 0.9960 - val_loss: 0.0300 - val_acc: 0.9921
Epoch 39/50
 - 3s - loss: 0.0107 - acc: 0.9963 - val_loss: 0.0353 - val_acc: 0.9926
Epoch 40/50
 - 3s - loss: 0.0135 - acc: 0.9956 - val_loss: 0.0406 - val_acc: 0.9916
Epoch 41/50
 - 3s - loss: 0.0125 - acc: 0.9959 - val_loss: 0.0320 - val_acc: 0.9925
Epoch 42/50
 - 3s - loss: 0.0102 - acc: 0.9964 - val_loss: 0.0333 - val_acc: 0.9922
Epoch 43/50
 - 3s - loss: 0.0108 - acc: 0.9964 - val_loss: 0.0326 - val_acc: 0.9928
Epoch 44/50
 - 3s - loss: 0.0100 - acc: 0.9967 - val_loss: 0.0310 - val_acc: 0.9929
Epoch 45/50
 - 3s - loss: 0.0116 - acc: 0.9960 - val_loss: 0.0354 - val_acc: 0.9915
Epoch 46/50
 - 3s - loss: 0.0118 - acc: 0.9960 - val_loss: 0.0305 - val_acc: 0.9928
Epoch 47/50
 - 3s - loss: 0.0102 - acc: 0.9965 - val_loss: 0.0366 - val_acc: 0.9923
Epoch 48/50
 - 3s - loss: 0.0104 - acc: 0.9964 - val_loss: 0.0384 - val_acc: 0.9918
Epoch 49/50
 - 3s - loss: 0.0111 - acc: 0.9959 - val_loss: 0.0372 - val_acc: 0.9922
Epoch 50/50
 - 3s - loss: 0.0089 - acc: 0.9972 - val_loss: 0.0377 - val_acc: 0.9919
```

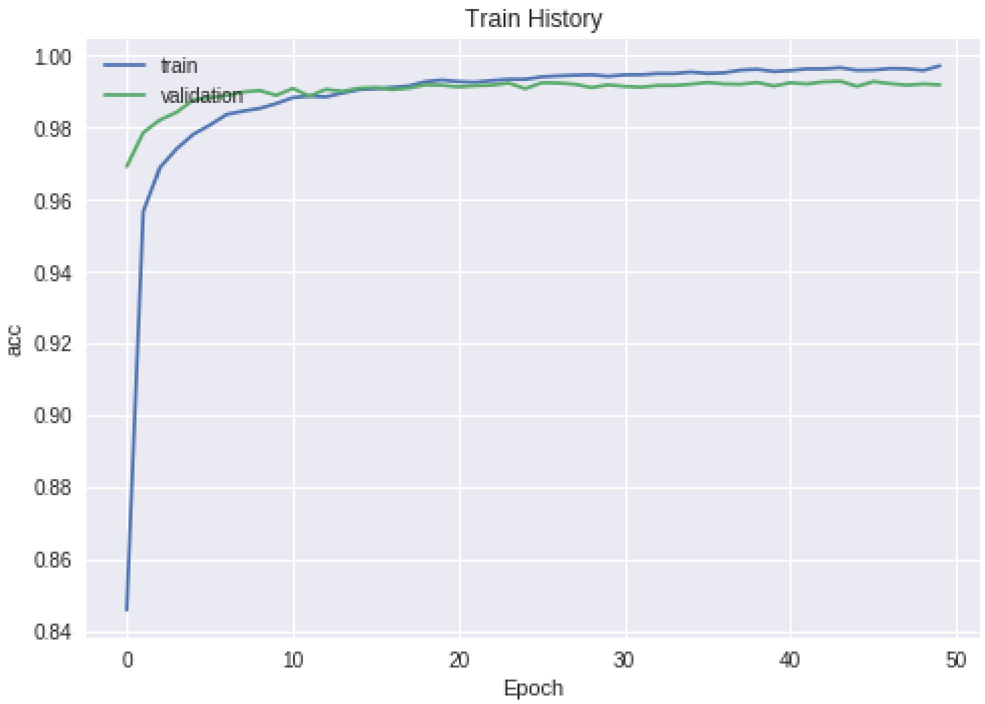
(7) Training history

In [0]:

```
def show_train_history(train_history,train,validation):
    plt.plot(train_history.history[train])
    plt.plot(train_history.history[validation])
    plt.title('Train History')
    plt.ylabel(train)
    plt.xlabel('Epoch')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
```

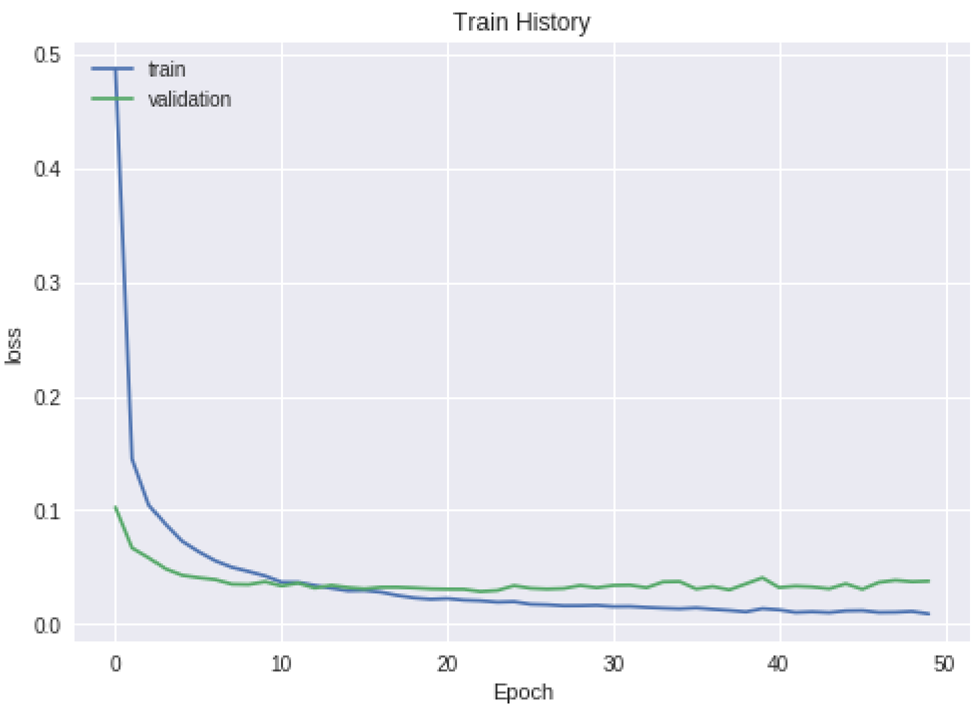
In [0]:

```
show_train_history(train_history,'acc','val_acc')
```



In [0]:

```
show_train_history(train_history,'loss','val_loss')
```



(8) Accuracy

In [0]:

```
scores = model.evaluate(x_Test4D_normalize, y_TestOneHot)
print()
print('accuracy=',scores[1])
```

```
10000/10000 [==============================] - 1s 78us/step

accuracy= 0.9936
```

(9) Prediction

In [0]:

```
prediction=model.predict_classes(x_Test4D_normalize)
```

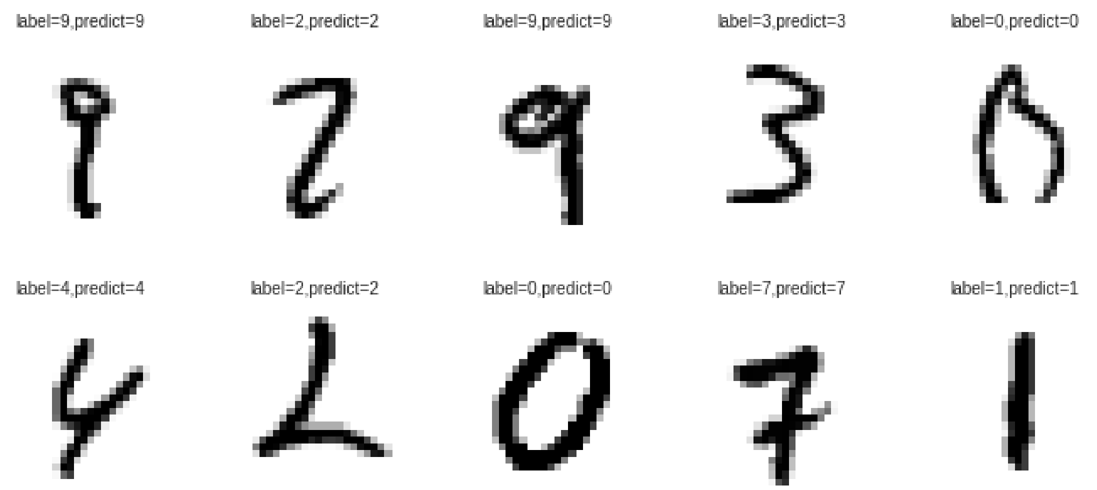
In [0]:

```
def plot_images_labels_prediction(images,labels,prediction,
                                  idx,num=10):
    fig = plt.gcf()
    fig.set_size_inches(12, 14)
    if num>25: num=25
    for i in range(0, num):
        ax=plt.subplot(5,5, 1+i)
        ax.imshow(images[idx], cmap='binary')
        title= "label=" +str(labels[idx])
        if len(prediction)>0:
            title+=",predict="+str(prediction[idx])

        ax.set_title(title,fontsize=10)
        ax.set_xticks([]);ax.set_yticks([])
        idx+=1
    plt.show()
```

In [0]:

```
plot_images_labels_prediction(x_test_image,y_test_label,
                              prediction,idx=320)
```



(10) Confusion matrix

In [0]:

```
pd.crosstab(y_test_label,prediction,
            rownames=['label'],colnames=['predict'])
```

Out[0]:

predict	0	1	2	3	4	5	6	7	8	9
label										
0	975	0	1	0	0	0	2	1	1	0
1	0	1134	1	0	0	0	0	0	0	0
2	1	1	1029	0	0	0	0	1	0	0
3	0	0	1	1006	0	2	0	0	1	0
4	0	0	0	0	979	0	1	0	0	2
5	1	0	0	3	0	887	1	0	0	0
6	3	2	0	0	2	3	947	0	1	0
7	0	2	6	1	0	0	0	1018	1	0
8	2	0	3	1	0	0	0	1	965	2
9	0	1	0	0	5	2	0	4	1	996

In [0]:

```
# save and load weights
model.save_weights('my_model_weights.h5')
model.load_weights('my_model_weights.h5')
```

In [0]:

```
model.save('my_model.h5')
del model  # deletes the existing model

from keras.models import load_model
model = load_model('my_model.h5')
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 16)        416       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 16)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 36)        14436     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 36)          0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 7, 7, 36)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 1764)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               225920    
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290      
=================================================================
Total params: 242,062
Trainable params: 242,062
Non-trainable params: 0
_________________________________________________________________
```