HW10

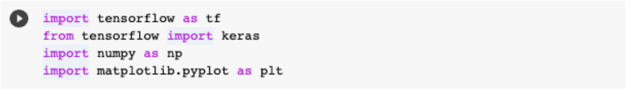
Date: 2021. 12. 06

Student ID: 21600635

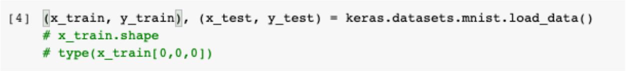
Name: Bomoon JUNG

1. Homework 10

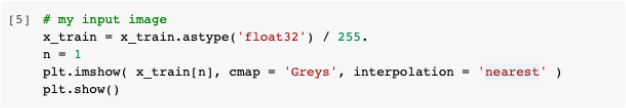
1) show one example of Hand-written Number



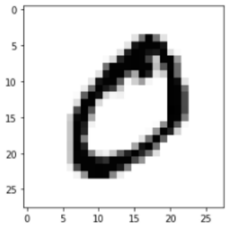
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
```



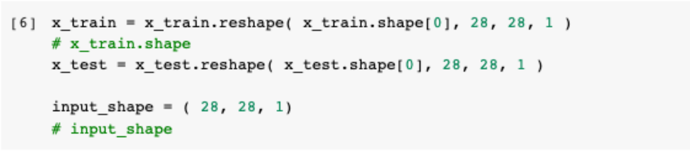
```
[4] (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
    # x_train.shape
    # type(x_train[0,0,0])
```



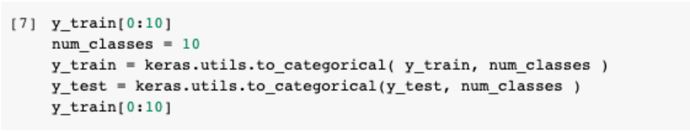
```
[5] # my input image
    x_train = x_train.astype('float32') / 255.
    n = 1
    plt.imshow( x_train[n], cmap = 'Greys', interpolation = 'nearest' )
    plt.show()
```



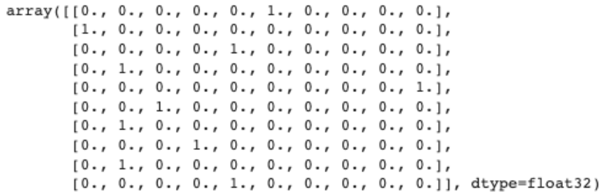
2) show training result



```
[6] x_train = x_train.reshape( x_train.shape[0], 28, 28, 1 )
    # x_train.shape
    x_test = x_test.reshape( x_test.shape[0], 28, 28, 1 )

    input_shape = ( 28, 28, 1)
    # input_shape
```



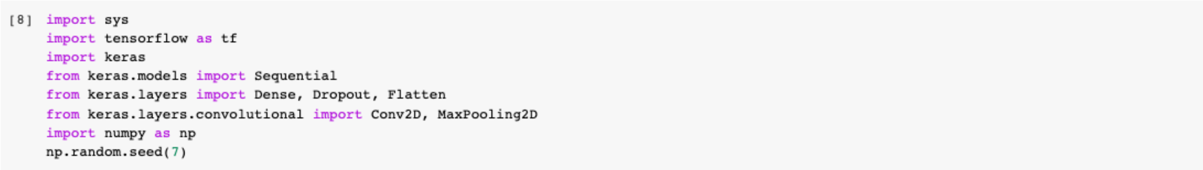
```
[7] y_train[0:10]
    num_classes = 10
    y_train = keras.utils.to_categorical( y_train, num_classes )
    y_test = keras.utils.to_categorical(y_test, num_classes )
    y_train[0:10]
```



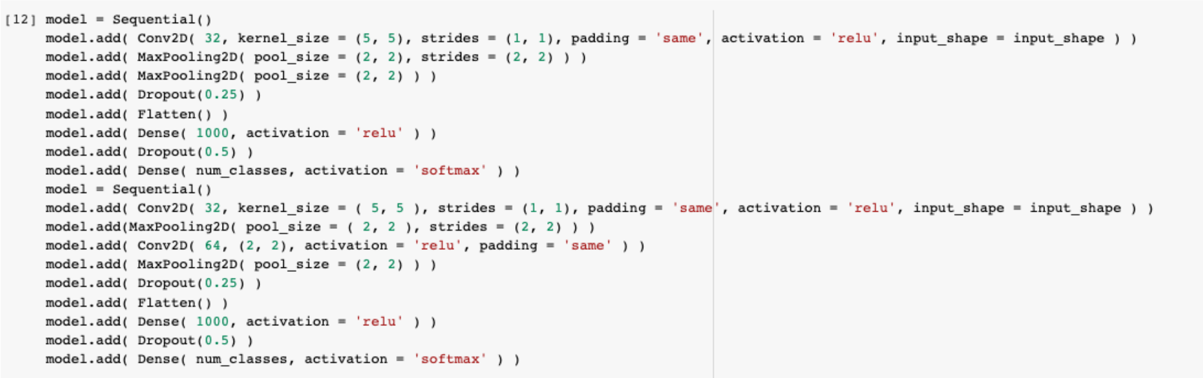
```
array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]], dtype=float32)
```

\

3) modeling

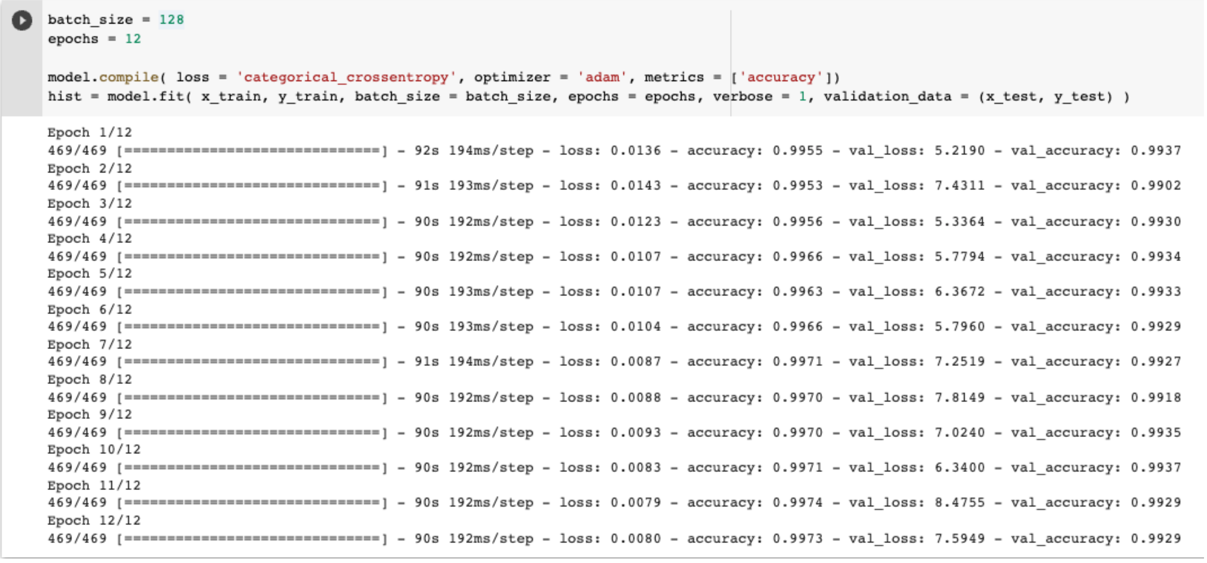


```
[8] import sys
    import tensorflow as tf
    import keras
    from keras.models import Sequential
    from keras.layers import Dense, Dropout, Flatten
    from keras.layers.convolutional import Conv2D, MaxPooling2D
    import numpy as np
    np.random.seed(7)
```



```
[12] model = Sequential()
     model.add( Conv2D( 32, kernel_size = (5, 5), strides = (1, 1), padding = 'same', activation = 'relu', input_shape = input_shape ) )
     model.add( MaxPooling2D( pool_size = (2, 2), strides = (2, 2) ) )
     model.add( MaxPooling2D( pool_size = (2, 2) ) )
     model.add( Dropout(0.25) )
     model.add( Flatten() )
     model.add( Dense( 1000, activation = 'relu' ) )
     model.add( Dropout(0.5) )
     model.add( Dense( num_classes, activation = 'softmax' ) )
     model = Sequential()
     model.add( Conv2D( 32, kernel_size = ( 5, 5 ), strides = (1, 1), padding = 'same', activation = 'relu', input_shape = input_shape ) )
     model.add(MaxPooling2D( pool_size = ( 2, 2 ), strides = (2, 2) ) )
     model.add( Conv2D( 64, (2, 2), activation = 'relu', padding = 'same' ) )
     model.add( MaxPooling2D( pool_size = (2, 2) ) )
     model.add( Dropout(0.25) )
     model.add( Flatten() )
     model.add( Dense( 1000, activation = 'relu' ) )
     model.add( Dropout(0.5) )
     model.add( Dense( num_classes, activation = 'softmax' ) )
```

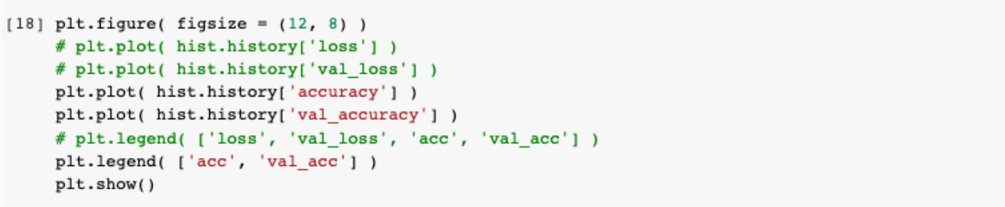
4) learning



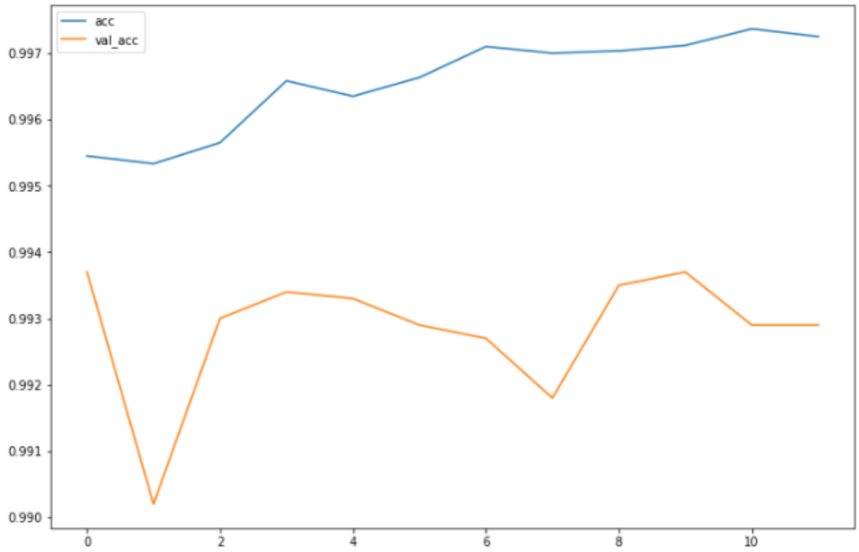
```
batch_size = 128
epochs = 12

model.compile( loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
hist = model.fit( x_train, y_train, batch_size = batch_size, epochs = epochs, verbose = 1, validation_data = (x_test, y_test) )
```

```
Epoch 1/12
469/469 [==============================] - 92s 194ms/step - loss: 0.0136 - accuracy: 0.9955 - val_loss: 5.2190 - val_accuracy: 0.9937
Epoch 2/12
469/469 [==============================] - 91s 193ms/step - loss: 0.0143 - accuracy: 0.9953 - val_loss: 7.4311 - val_accuracy: 0.9902
Epoch 3/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0123 - accuracy: 0.9956 - val_loss: 5.3364 - val_accuracy: 0.9930
Epoch 4/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0107 - accuracy: 0.9966 - val_loss: 5.7794 - val_accuracy: 0.9934
Epoch 5/12
469/469 [==============================] - 90s 193ms/step - loss: 0.0107 - accuracy: 0.9963 - val_loss: 6.3672 - val_accuracy: 0.9933
Epoch 6/12
469/469 [==============================] - 90s 193ms/step - loss: 0.0104 - accuracy: 0.9966 - val_loss: 5.7960 - val_accuracy: 0.9929
Epoch 7/12
469/469 [==============================] - 91s 194ms/step - loss: 0.0087 - accuracy: 0.9971 - val_loss: 7.2519 - val_accuracy: 0.9927
Epoch 8/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0088 - accuracy: 0.9970 - val_loss: 7.8149 - val_accuracy: 0.9918
Epoch 9/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0093 - accuracy: 0.9970 - val_loss: 7.0240 - val_accuracy: 0.9935
Epoch 10/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0083 - accuracy: 0.9971 - val_loss: 6.3400 - val_accuracy: 0.9937
Epoch 11/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0079 - accuracy: 0.9974 - val_loss: 8.4755 - val_accuracy: 0.9929
Epoch 12/12
469/469 [==============================] - 90s 192ms/step - loss: 0.0080 - accuracy: 0.9973 - val_loss: 7.5949 - val_accuracy: 0.9929
```

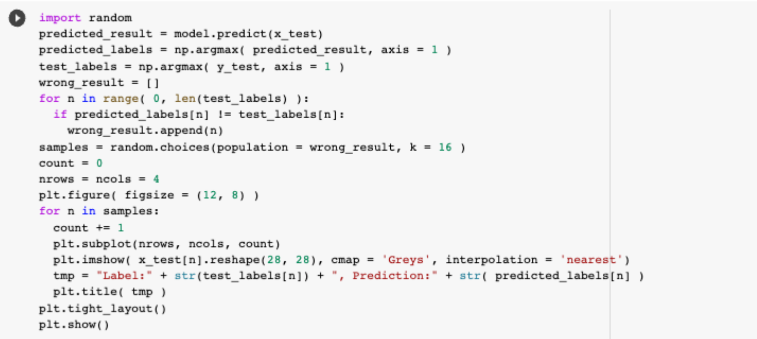
5) graph of accuracy and validate accuracy



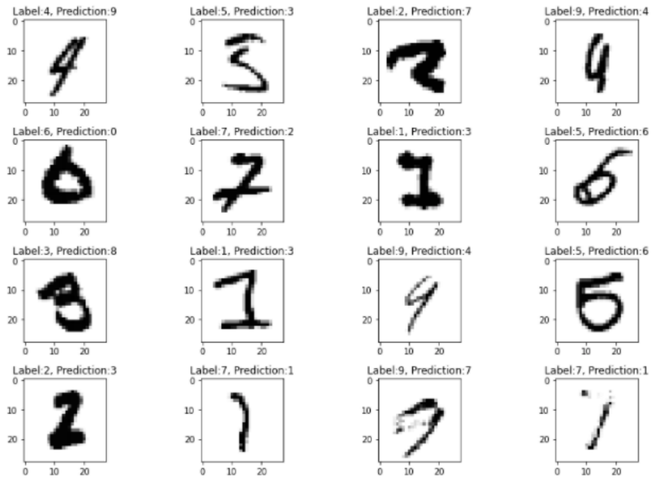
```
[18] plt.figure( figsize = (12, 8) )
     # plt.plot( hist.history['loss'] )
     # plt.plot( hist.history['val_loss'] )
     plt.plot( hist.history['accuracy'] )
     plt.plot( hist.history['val_accuracy'] )
     # plt.legend( ['loss', 'val_loss', 'acc', 'val_acc'] )
     plt.legend( ['acc', 'val_acc'] )
     plt.show()
```



6) results of error case



```
import random
predicted_result = model.predict(x_test)
predicted_labels = np.argmax( predicted_result, axis = 1 )
test_labels = np.argmax( y_test, axis = 1 )
wrong_result = []
for n in range( 0, len(test_labels) ):
  if predicted_labels[n] != test_labels[n]:
    wrong_result.append(n)
samples = random.choices(population = wrong_result, k = 16 )
count = 0
nrows = ncols = 4
plt.figure( figsize = (12, 8) )
for n in samples:
  count += 1
  plt.subplot(nrows, ncols, count)
  plt.imshow( x_test[n].reshape(28, 28), cmap = 'Greys', interpolation = 'nearest')
  tmp = "Label:" + str(test_labels[n]) + ", Prediction:" + str( predicted_labels[n] )
  plt.title( tmp )
plt.tight_layout()
plt.show()
```


Label:4, Prediction:9
Label:5, Prediction:3
Label:2, Prediction:7
Label:9, Prediction:4
Label:6, Prediction:0
Label:7, Prediction:2
Label:1, Prediction:3
Label:5, Prediction:6
Label:3, Prediction:8
Label:1, Prediction:3
Label:9, Prediction:4
Label:5, Prediction:6
Label:2, Prediction:3
Label:7, Prediction:1
Label:9, Prediction:7
Label:7, Prediction:1
0
10
20