Trần Ngọc Đoàn - 19146175 - Chiều T5 tiết 12-15

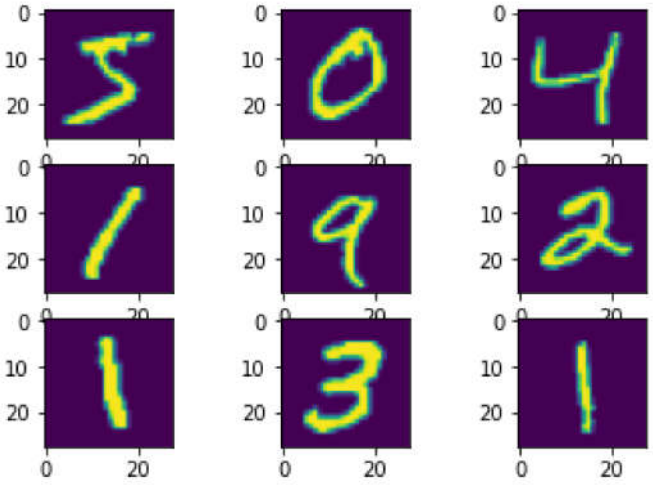
https://github.com/DoanAI/MNIST.git

```
from keras.datasets import mnist
import matplotlib.pyplot as plt
```

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
for i in range(9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x_train[i])
plt.show()
```



```
x = x_test
```

```
x_train = x_train.reshape(60000,784)
x_test = x_test.reshape(10000,784)
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255
```

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

```
# Tao model
```

```
from keras.models import Sequential
from keras.layers import Activation, Dropout, Dense
```

```
from tensorflow.keras.optimizers import RMSprop
```

```
model = Sequential()
```

```
model.add(Dense(512,activation='relu',input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

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

```
history = model.fit(x_train, y_train, batch_size=128, epochs=100, verbose=1, validation_da
```

```
469/469 [==============================] - 2s 4ms/step - loss: 0.0120 - accuracy: (
Epoch 73/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0109 - accuracy: (
Epoch 74/100
469/469 [==============================] - 2s 5ms/step - loss: 0.0099 - accuracy: (
Epoch 75/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0127 - accuracy: (
Epoch 76/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0133 - accuracy: (
Epoch 77/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0076 - accuracy: (
Epoch 78/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0089 - accuracy: (
Epoch 79/100
469/469 [==============================] - 2s 5ms/step - loss: 0.0116 - accuracy: (
Epoch 80/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0093 - accuracy: (
Epoch 81/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0096 - accuracy: (
Epoch 82/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0109 - accuracy: (
Epoch 83/100
```

```
469/469 [==============================] - 2s 4ms/step - loss: 0.0088 - accuracy: 
Epoch 84/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0096 - accuracy: 
Epoch 85/100
469/469 [==============================] - 2s 5ms/step - loss: 0.0099 - accuracy: 
Epoch 86/100
469/469 [==============================] - 3s 5ms/step - loss: 0.0102 - accuracy: 
Epoch 87/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0091 - accuracy: 
Epoch 88/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0099 - accuracy: 
Epoch 89/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0112 - accuracy: 
Epoch 90/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0111 - accuracy: 
Epoch 91/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0089 - accuracy: 
Epoch 92/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0137 - accuracy: 
Epoch 93/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0085 - accuracy: 
Epoch 94/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0083 - accuracy: 
Epoch 95/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0084 - accuracy: 
Epoch 96/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0070 - accuracy: 
Epoch 97/100
469/469 [==============================] - 4s 8ms/step - loss: 0.0094 - accuracy: 
Epoch 98/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0104 - accuracy: 
Epoch 99/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0067 - accuracy: 
Epoch 100/100
469/469 [==============================] - 2s 4ms/step - loss: 0.0092 - accuracy: 
```

```
model.save("ANN_MNIST.h5")
from tensorflow.keras.models import load_model
model=load_model('ANN_MNIST.h5')
```

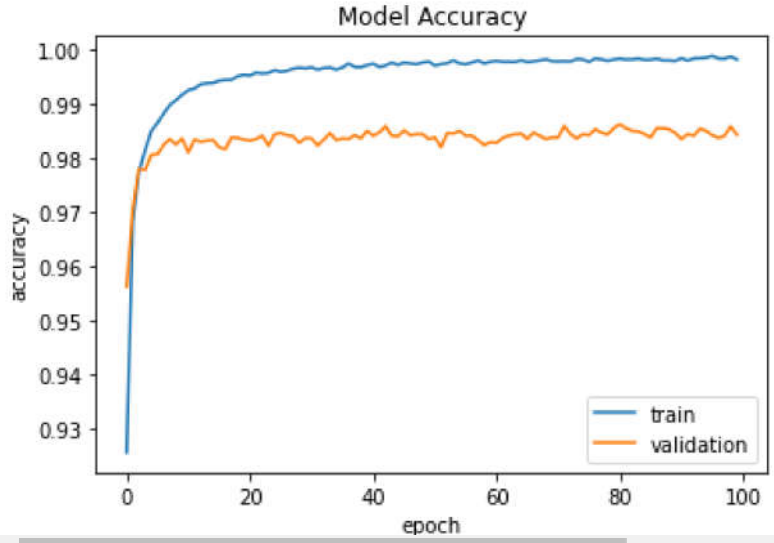
```
score=model.evaluate(x_test, y_test, verbose=1)
print('Test loss =', score[0])
print('Test accuracy =', score[1])
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.2831 - accuracy: 0.9
Test loss = 0.2831094563007355
Test accuracy = 0.984499990940094
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='upper-left')
```

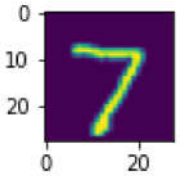
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecatior
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will raise an exception in 3.3.

<matplotlib.legend.Legend at 0x7fbcd00baa10>
```

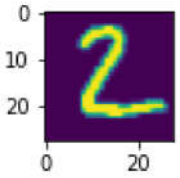



```
import numpy as np
y_pred = model.predict(x_test)
for i in range (9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x[i])
  print(np.round(y_pred[i]))
  plt.show()
```

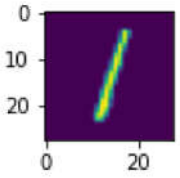
```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



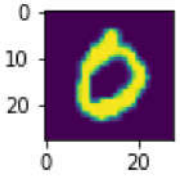

```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



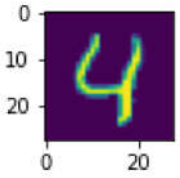

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



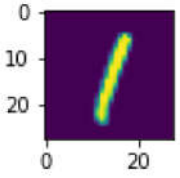

```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



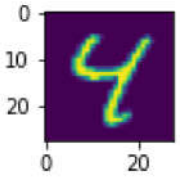

```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



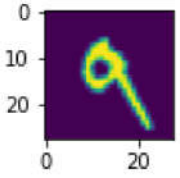

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```




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




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




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