Trần Ngọc Đoàn - 19146175 - Chiều T5 tiết 12-16

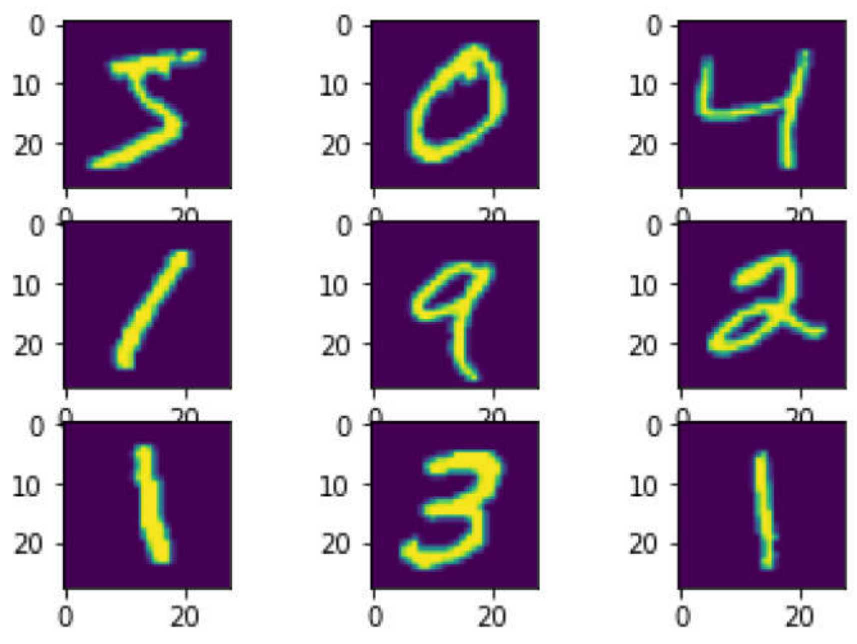
https://github.com/DoanAI/MNIST.git

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

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

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

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



```
x = x_test
from tensorflow.keras.utils import to_categorical
x_train = x_train.reshape((x_train.shape[0], x_train.shape[1], x_train.shape[2], 1))
x_test = x_test .reshape((x_test.shape[0], x_test.shape[1], x_test.shape[2], 1))
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

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

```
x_train = x_train.astype('float32')/255
x_test = x_test.astype('float32')/255
```

```
from keras.models import Sequential
from tensorflow.keras.layers import Flatten, MaxPooling2D, Conv2D
model = Sequential()
```

```
from tensorflow.keras.layers import Dense, Activation, Dropout
model.add(Conv2D(32, (3,3), activation='relu', input_shape= (28,28,1)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(48, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 11, 11, 48)        13872     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 5, 5, 48)         0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 5, 5, 48)          0         
                                                                 
 flatten_1 (Flatten)         (None, 1200)              0         
                                                                 
 dense_2 (Dense)             (None, 500)               600500    
                                                                 
 dense_3 (Dense)             (None, 10)                5010      
                                                                 
=================================================================
Total params: 619,702
Trainable params: 619,702
Non-trainable params: 0
_________________________________________________________________
```

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

```
history = model.fit(x_train, y_train, epochs=10, batch_size = 128, verbose= 2, validation_split= 0.1)
```

```
Epoch 1/10
422/422 - 42s - loss: 0.2490 - accuracy: 0.9236 - val_loss: 0.0625 - val_accuracy: 0.9840 - 42s/
Epoch 2/10
422/422 - 38s - loss: 0.0833 - accuracy: 0.9738 - val_loss: 0.0398 - val_accuracy: 0.9880 - 38s/
Epoch 3/10
422/422 - 40s - loss: 0.0599 - accuracy: 0.9812 - val_loss: 0.0355 - val_accuracy: 0.9898 - 40s/
Epoch 4/10
422/422 - 38s - loss: 0.0493 - accuracy: 0.9842 - val_loss: 0.0342 - val_accuracy: 0.9900 - 38s/
Epoch 5/10
422/422 - 38s - loss: 0.0420 - accuracy: 0.9866 - val_loss: 0.0330 - val_accuracy: 0.9920 - 38s/
Epoch 6/10
422/422 - 38s - loss: 0.0354 - accuracy: 0.9882 - val_loss: 0.0274 - val_accuracy: 0.9925 - 38s/
Epoch 7/10
422/422 - 39s - loss: 0.0323 - accuracy: 0.9897 - val_loss: 0.0264 - val_accuracy: 0.9930 - 39s/
Epoch 8/10
422/422 - 38s - loss: 0.0282 - accuracy: 0.9909 - val_loss: 0.0289 - val_accuracy: 0.9920 - 38s/
Epoch 9/10
422/422 - 38s - loss: 0.0260 - accuracy: 0.9912 - val_loss: 0.0242 - val_accuracy: 0.9932 - 38s/
Epoch 10/10
422/422 - 38s - loss: 0.0238 - accuracy: 0.9921 - val_loss: 0.0271 - val_accuracy: 0.9932 - 38s/
```

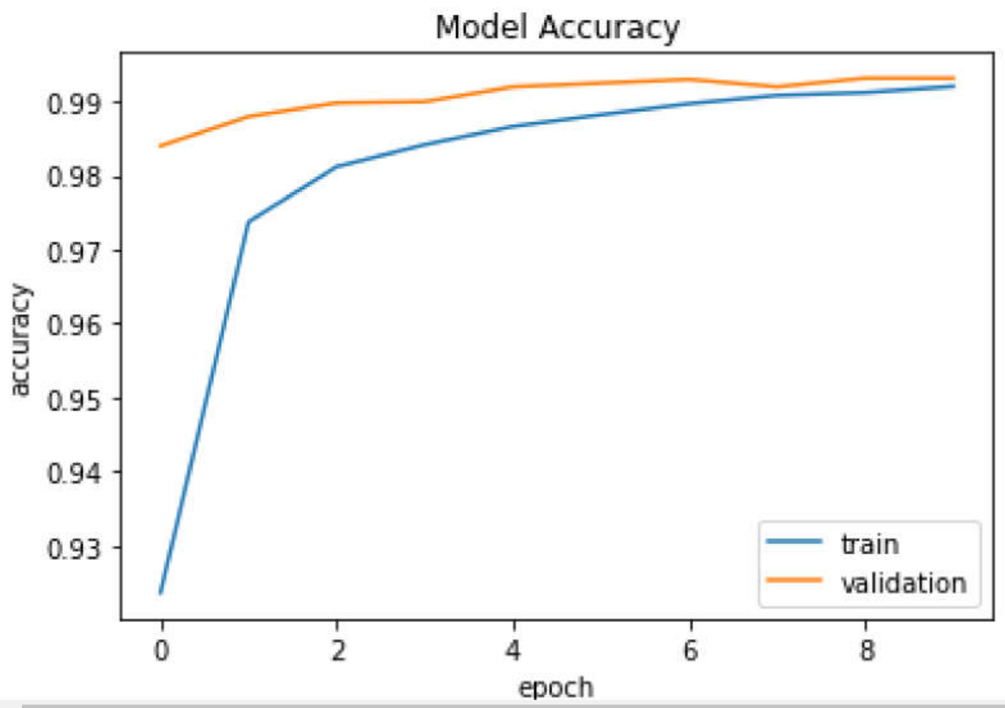
```
score = model.evaluate(x_test, y_test,verbose=0)
print('Sai so kiem tra la:', score[0])
print('Do chinh xac kiem tra la:', score[1]*100, '%')
```

```
Sai so kiem tra la: 0.020718380808883026
Do chinh xac kiem tra la: 99.26000237464905 %
```

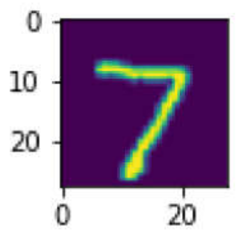
```
model.save('CNN_MNIST.h5')
from keras.models import load_model
model1 = load_model('CNN_MNIST.h5')
```

```
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')
```

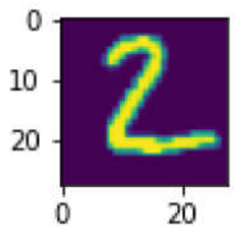
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecationWarning: Un
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center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

<matplotlib.legend.Legend at 0x7fd24ba95910>
```



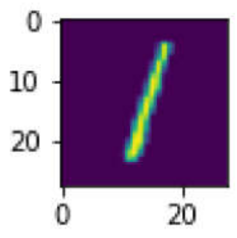
```
import numpy as np
y_pred = model.predict(x_test)
for i in range (9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x[i])
  plt.show()
  print(np.round(y_pred[i]))
```



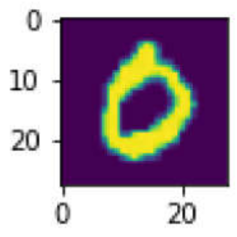
```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



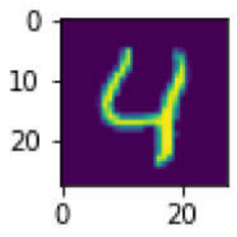
```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



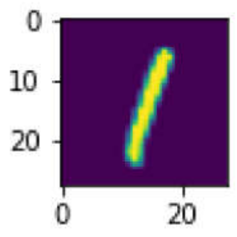
```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



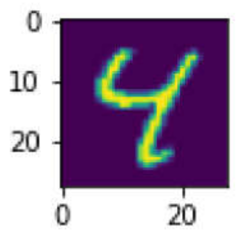
```
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```



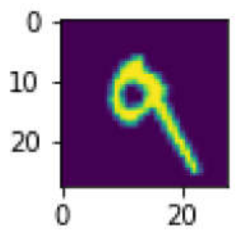
```
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



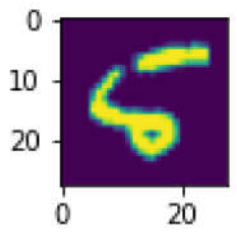
```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



```
[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.]
```