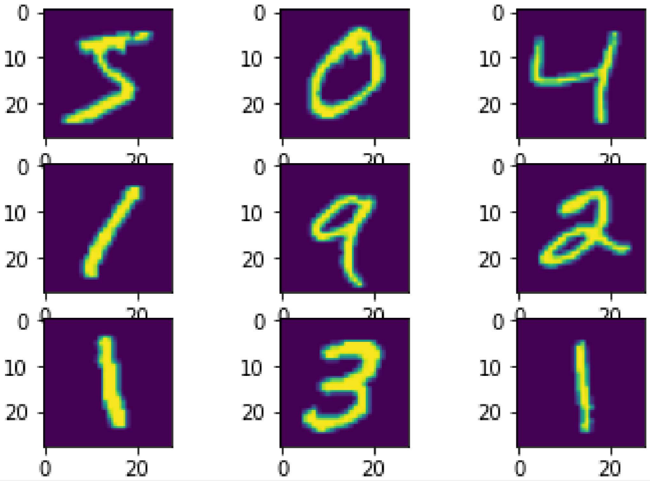
```
from keras.datasets import mnist
import matplotlib.pyplot as plt
(x_train,y_train),(x_test,y_test)=mnist.load_data()
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i])
plt.show()
```

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



```
x=x_test
```

```
x_train =x_train.reshape(60000,784)
```

```
x_train =x_train.reshape(60000,784)
```

```
x_test = x_test.reshape(10000,784)
```

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

```
x_train/=255
x_test/=255
```

```
from tensorflow.keras.utils import to_categorical

y_train=to_categorical(y_train,10)
y_test=to_categorical(y_test,10)
y_train.shape
```

```
(60000, 10)
```

```
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
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(0.1))
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
_________________________________________________________________
```

```
from tensorflow.keras.optimizers import RMSprop
model.compile(loss='categorical_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
model.fit(x_train, y_train, batch_size=128,epochs =20, verbose=1,validation_data=(x_test,y_te
```

```
Epoch 1/20
469/469 [==============================] - 10s 19ms/step - loss: 0.2418 - accuracy: 0.92
Epoch 2/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0996 - accuracy: 0.969
Epoch 3/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0744 - accuracy: 0.977
```

```
Epoch 4/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0583 - accuracy: 0.982
Epoch 5/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0488 - accuracy: 0.985
Epoch 6/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0409 - accuracy: 0.987
Epoch 7/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0362 - accuracy: 0.988
Epoch 8/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0321 - accuracy: 0.990
Epoch 9/20
469/469 [==============================] - 11s 23ms/step - loss: 0.0295 - accuracy: 0.99
Epoch 10/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0278 - accuracy: 0.991
Epoch 11/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0253 - accuracy: 0.992
Epoch 12/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0226 - accuracy: 0.993
Epoch 13/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0211 - accuracy: 0.993
Epoch 14/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0211 - accuracy: 0.994
Epoch 15/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0183 - accuracy: 0.995
Epoch 16/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0191 - accuracy: 0.995
Epoch 17/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0191 - accuracy: 0.994
Epoch 18/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0181 - accuracy: 0.995
Epoch 19/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0166 - accuracy: 0.995
Epoch 20/20
469/469 [==============================] - 9s 19ms/step - loss: 0.0152 - accuracy: 0.995
<keras.callbacks.History at 0x7f91f4f56d90>
```

```
score=model.evaluate(x_test,y_test,verbose=1)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.1145 - accuracy: 0.9837
```

```
model.save('final.h5')
```

```
history=model.fit(x_train, y_train, batch_size=128,epochs =5, verbose=1,validation_data=(x_te
```

```
Epoch 1/5
469/469 [==============================] - 9s 19ms/step - loss: 0.0149 - accuracy: 0.995
Epoch 2/5
469/469 [==============================] - 9s 19ms/step - loss: 0.0139 - accuracy: 0.996
Epoch 3/5
469/469 [==============================] - 9s 19ms/step - loss: 0.0134 - accuracy: 0.996
```

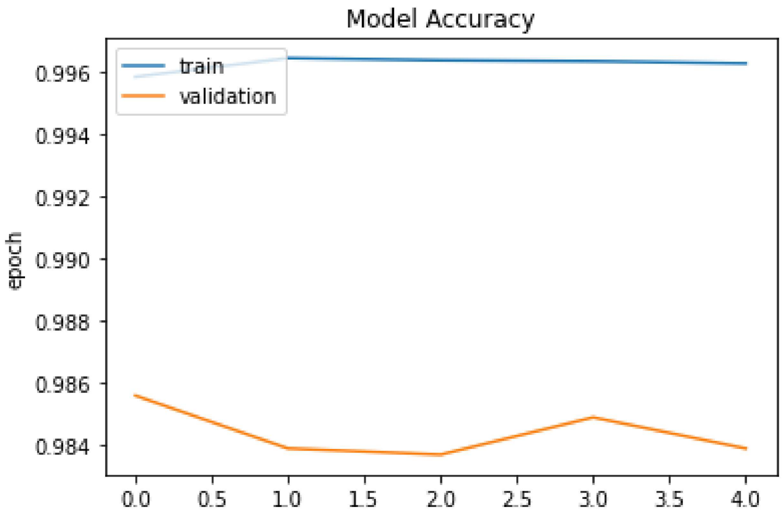
```
Epoch 4/5
469/469 [==============================] - 9s 19ms/step - loss: 0.0138 - accuracy: 0.996
Epoch 5/5
469/469 [==============================] - 9s 19ms/step - loss: 0.0150 - accuracy: 0.996
```

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

```
313/313 [==============================] - 1s 4ms/step - loss: 0.1415 - accuracy: 0.983
Test loss = 0.14153271913528442
Test accurracy = 0.9839000105857849
```

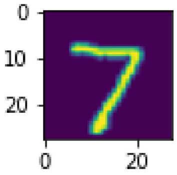
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.ylabel('epoch')
plt.legend(['train','validation'],loc='upper left')
```

```
<matplotlib.legend.Legend at 0x7f91f4e033d0>
```

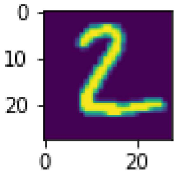


```
from keras.datasets import fashion_mnist
y_pred=model.predict(x_test)
for i in range(9):
  plt.subplot(330+i+1)
  plt.imshow(x[i])
  print(y_pred[i])
  plt.show()
```

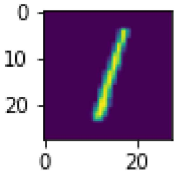
```
[2.4944056e-31 2.1060819e-26 3.8928057e-24 1.7494020e-20 1.2162793e-31
 1.5769648e-29 0.0000000e+00 1.0000000e+00 1.1831653e-31 1.3656826e-20]
```



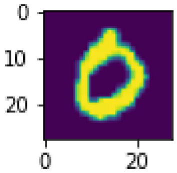
```
[1.4417255e-38 6.2308293e-38 1.0000000e+00 1.2504142e-33 0.0000000e+00
 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00]
```



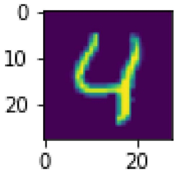
```
[1.6620423e-26 1.0000000e+00 8.0100262e-19 7.0488222e-25 7.4167628e-14
 7.5883228e-24 1.2274318e-18 1.6607280e-14 3.4545728e-18 7.8897947e-24]
```



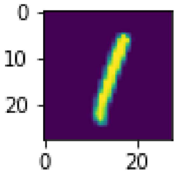
```
[1.0000000e+00 3.7548158e-24 1.2766455e-15 3.2373960e-17 9.1569241e-18
 3.7680715e-14 6.8683071e-12 5.4574172e-15 5.7371538e-23 5.4673518e-13]
```



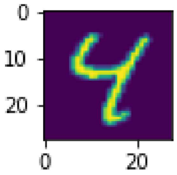
```
[8.3746564e-22 1.9547149e-26 5.3597927e-21 8.9621181e-26 1.0000000e+00
 3.4913161e-22 2.8091970e-21 3.5117302e-12 2.4528856e-25 1.0106165e-12]
```



```
[6.3096118e-25 1.0000000e+00 6.6595744e-18 2.3646699e-21 5.7668297e-12
 5.1550221e-25 7.3765893e-20 8.4513039e-12 4.5898571e-16 1.2441229e-20]
```



```
[2.4269308e-37 4.3861167e-28 1.4363220e-33 5.6444061e-34 1.0000000e+00
 2.1414523e-26 5.4055365e-30 1.8070610e-22 1.2439717e-19 1.4373039e-17]
```



```
[1.6043744e-19 1.2239954e-17 9.4947084e-10 4.6468189e-04 3.2800472e-12
 9.0335101e-07 4.8740649e-27 3.2499013e-08 9.4537191e-11 9.9953437e-01]
```

✓ 4 giây hoàn thành lúc 21:51