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

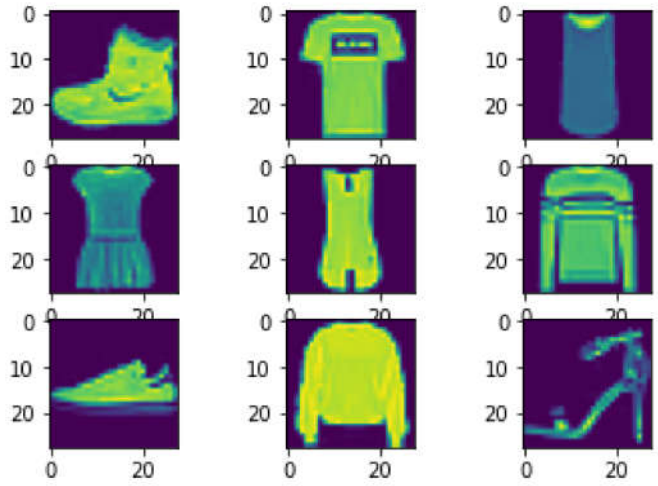
https://github.com/DoanAI/Fashion.git

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

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tra
32768/29515 [================================] - 0s 0us/step
40960/29515 [========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tra
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
16384/5148 [===============================================================================
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

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



```
print(x_train.shape,x_test.shape)
```

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

```
x = x_test
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)
```

```
print(x_train.shape, x_test.shape)
```

```
(60000, 784) (10000, 784)
```

```
# 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_data=(x
```

```
469/469 [==============================] - 9s 18ms/step - loss: 0.2060 - accuracy:
Epoch 73/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2074 - accuracy:
Epoch 74/100
469/469 [==============================] - 9s 18ms/step - loss: 0.2074 - accuracy:
Epoch 75/100
```

```
Epoch 75/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2171 - accuracy:
Epoch 76/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2131 - accuracy:
Epoch 77/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2049 - accuracy:
Epoch 78/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2079 - accuracy:
Epoch 79/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2028 - accuracy:
Epoch 80/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2049 - accuracy:
Epoch 81/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2041 - accuracy:
Epoch 82/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2024 - accuracy:
Epoch 83/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1963 - accuracy:
Epoch 84/100
469/469 [==============================] - 8s 17ms/step - loss: 0.2055 - accuracy:
Epoch 85/100
469/469 [==============================] - 8s 17ms/step - loss: 0.2039 - accuracy:
Epoch 86/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1955 - accuracy:
Epoch 87/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2042 - accuracy:
Epoch 88/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1971 - accuracy:
Epoch 89/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1982 - accuracy:
Epoch 90/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1928 - accuracy:
Epoch 91/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1934 - accuracy:
Epoch 92/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1952 - accuracy:
Epoch 93/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1962 - accuracy:
Epoch 94/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2063 - accuracy:
Epoch 95/100
469/469 [==============================] - 8s 18ms/step - loss: 0.2019 - accuracy:
Epoch 96/100
469/469 [==============================] - 9s 18ms/step - loss: 0.1910 - accuracy:
Epoch 97/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1916 - accuracy:
Epoch 98/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1989 - accuracy:
Epoch 99/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1894 - accuracy:
Epoch 100/100
469/469 [==============================] - 8s 18ms/step - loss: 0.1894 - accuracy:
```

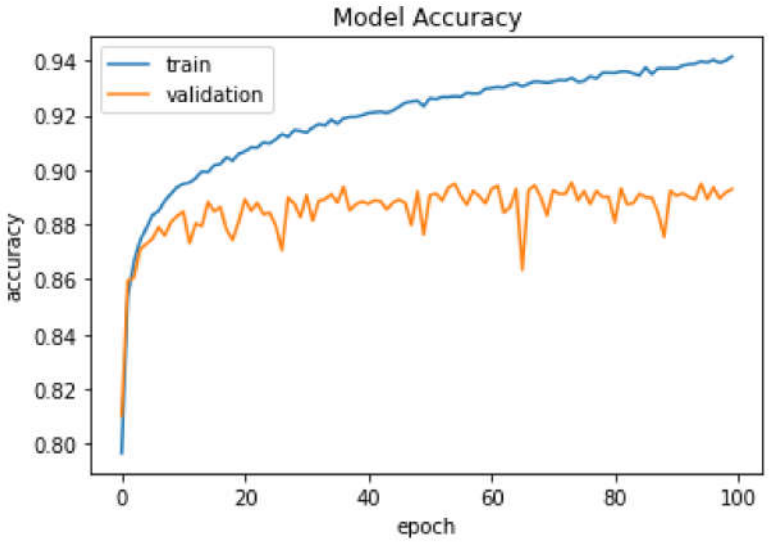
```
score=model.evaluate(x_test, y_test, verbose=1)
print('Test loss =', score)
print('Test accuracy =', score[1])
```

```
313/313 [==============================] - 1s 3ms/step - loss: 1.0513 - accuracy: 0.8
```

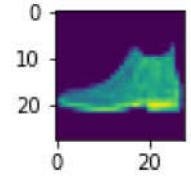
```
Test loss = [1.0512856245040894, 0.8927000164985657]
Test accuracy = 0.8927000164985657
```

```
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')
```

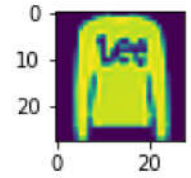
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecatior
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will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

<matplotlib.legend.Legend at 0x7f3e5dbed910>
```



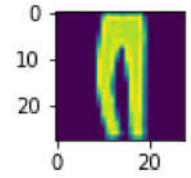

```
import numpy as np
y_pred = model.predict(x_test)
for i in range (9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x[i])
  plt.show()
  print(np.round(y_pred[i]))
```



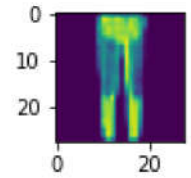

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```



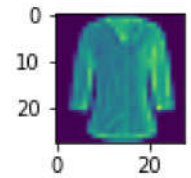

```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



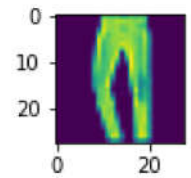

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



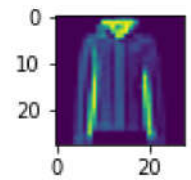

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



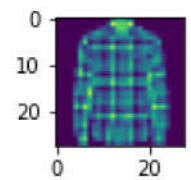

```
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```



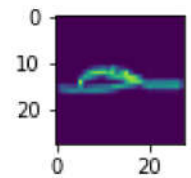

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```




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




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




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