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

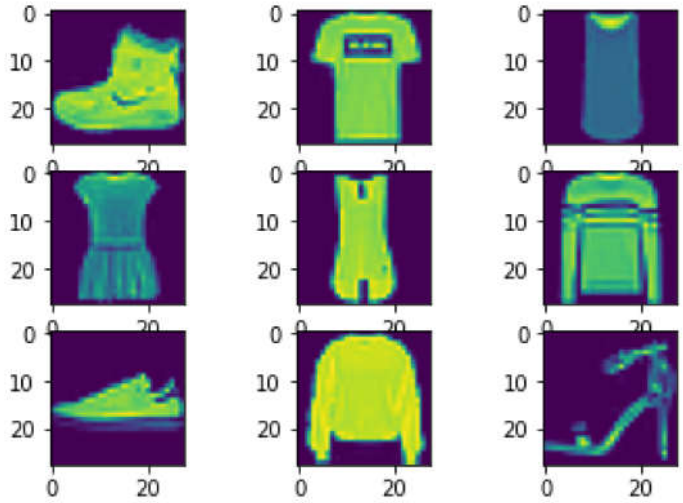
https://github.com/DoanAI/Fashion.git

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

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets
32768/29515 [================================] - 0s 0us/step
40960/29515 [========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets
16384/5148 [===============================================================================
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
(60000, 28, 28)
(10000, 28, 28)
```

```
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
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
```

Create Model

```
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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 48)        13872

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 48)         0
 2D)

 dropout (Dropout)           (None, 5, 5, 48)          0

 flatten (Flatten)           (None, 1200)              0

 dense (Dense)               (None, 500)               600500
```

```
 dense_1 (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=['accur
```

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

```
Epoch 1/10
422/422 - 41s - loss: 0.5863 - accuracy: 0.7853 - val_loss: 0.4044 - val_accuracy
Epoch 2/10
422/422 - 42s - loss: 0.4057 - accuracy: 0.8516 - val_loss: 0.3283 - val_accuracy
Epoch 3/10
422/422 - 43s - loss: 0.3542 - accuracy: 0.8701 - val_loss: 0.3142 - val_accuracy
Epoch 4/10
422/422 - 40s - loss: 0.3200 - accuracy: 0.8809 - val_loss: 0.2915 - val_accuracy
Epoch 5/10
422/422 - 40s - loss: 0.2981 - accuracy: 0.8897 - val_loss: 0.2724 - val_accuracy
Epoch 6/10
422/422 - 39s - loss: 0.2792 - accuracy: 0.8968 - val_loss: 0.2660 - val_accuracy
Epoch 7/10
422/422 - 39s - loss: 0.2651 - accuracy: 0.9016 - val_loss: 0.2633 - val_accuracy
Epoch 8/10
422/422 - 39s - loss: 0.2535 - accuracy: 0.9046 - val_loss: 0.2457 - val_accuracy
Epoch 9/10
422/422 - 39s - loss: 0.2446 - accuracy: 0.9084 - val_loss: 0.2481 - val_accuracy
Epoch 10/10
422/422 - 39s - loss: 0.2348 - accuracy: 0.9121 - val_loss: 0.2388 - val_accuracy
```

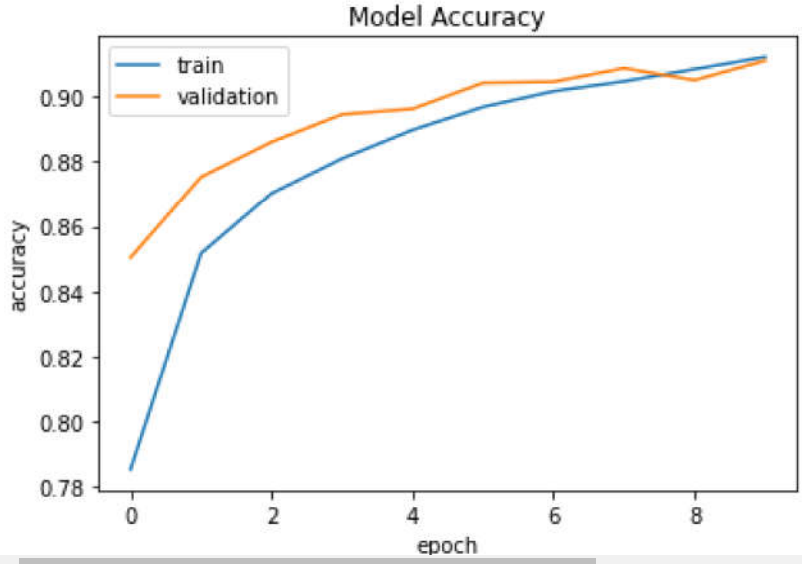
```
score = model.evaluate(x_test, y_test,verbose=0)
print('Sai so kiem tra la:', score[0])
print('Do chinh xac kiem tra la:', score[1]*100)
```

```
Sai so kiem tra la: 0.25306054949760437
Do chinh xac kiem tra la: 90.61999917030334
```

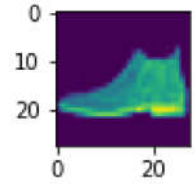
```
model.save('CNN_FASHION_MNIST.h5')
from keras.models import load_model
model1 = load_model('CNN_FASHION_MNIST.h5')
```

```
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')
```

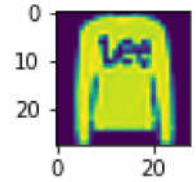
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: MatplotlibDeprecation
        best
        upper right
        upper left
        lower left
        lower right
        right
        center left
        center right
        lower center
        upper center
        center
This will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

<matplotlib.legend.Legend at 0x7f7e2572ff50>
```



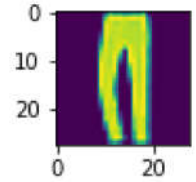

```
import numpy as np
y_pred = model.predict(x_test)
for i in range (9):
  plt.subplot(330+i+1)    # 330 mean: 3 hang 3 cot
  plt.imshow(x[i])
  plt.show()
  print(np.round(y_pred[i]))
```



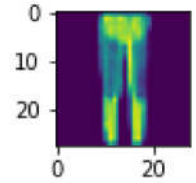

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```



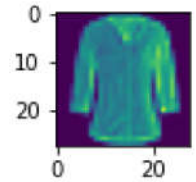

```
[0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```



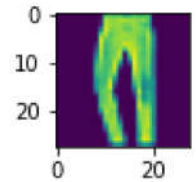

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



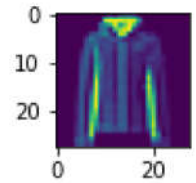

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



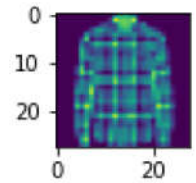

```
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
```




```
[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.]
```