

Double-click (or enter) to edit

```
!pip install -q keras
import keras
```

↳ Using TensorFlow backend.

```
import numpy as np
from keras.utils import to_categorical
import matplotlib.pyplot as plt
%matplotlib inline
from tqdm import tqdm
import os
from random import shuffle
import cv2
```

```
from google.colab import drive
drive.mount("/content/drive")
```

↳ Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=9](https://accounts.google.com/o/oauth2/auth?client_id=9)

Enter your authorization code:

.....

Mounted at /content/drive

```
train_images=('/file//C:\Users\lenovo\OneDrive\t10k-images-idx3-ubyte.zip')
train_file=('/file//C:\Users\lenovo\OneDrive\t10k-labels-idx1-ubyte.zip')
test_file=('/file//C:\Users\lenovo\OneDrive\train-images-idx3-ubyte.zip')
test_file=('/file//C:\Users\lenovo\OneDrive\train-labels-idx1-ubyte.zip')
```

↳

```
from keras.datasets import fashion_mnist
(train_X,train_Y),(test_X,test_Y)=fashion_mnist.load_data()
```

↳ Downloading data from <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t32768/29515> [=====] - 0s 3us/step  
 Downloading data from <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t26427392/26421880> [=====] - 1s 0us/step  
 Downloading data from <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t8192/5148> [=====] - 0s 0us/step  
 Downloading data from <http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t4423680/4422102> [=====] - 1s 0us/step

```
import numpy as np
from keras.utils import to_categorical
import matplotlib.pyplot as plt
%matplotlib inline

print('training data shape:',train_X.shape,train_Y.shape)
```

```
print('testing data shape:',test_X.shape,test_Y.shape)
```

```
↳ training data shape: (60000, 28, 28) (60000,)
   testing data shape: (10000, 28, 28) (10000,)
```

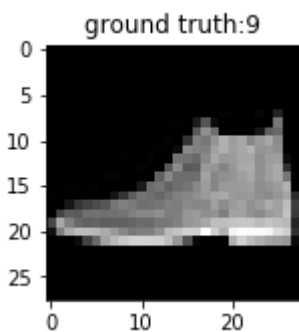
```
classes=np.unique(train_Y)
nclasses=len(classes)
print('total number of outputs:',nclasses)
print('output classes:',classes)
```

```
↳ total number of outputs: 10
   output classes: [0 1 2 3 4 5 6 7 8 9]
```

```
plt.figure(figsize=[5,5])
plt.subplot(121)
plt.imshow(train_X[0],cmap='gray')
plt.title("ground truth:{}".format(train_Y[0]))

plt.subplot(121)
plt.imshow(test_X[0],cmap='gray')
plt.title("ground truth:{}".format(test_Y[0]))
```

```
↳ /usr/local/lib/python3.6/dist-packages/matplotlib/figure.py:98: MatplotlibDeprecat
Adding an axes using the same arguments as a previous axes currently reuses the ea
"Adding an axes using the same arguments as a previous axes "
Text(0.5, 1.0, 'ground truth:9')
```



```
train_X=train_X.reshape(-1,28,28,1)
test_X=test_X.reshape(-1,28,28,1)
train_X.shape,test_X.shape
```

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

```
train_X=train_X.astype('float32')
test_X=test_X.astype('float32')
train_X=train_X/255
test_X=test_X/255
```

```
train_Y_one_hot=to_categorical(train_Y)
test_Y_one_hot=to_categorical(test_Y)

print('original label:',train_Y[0])
print('after conversion to one_hot:',train_Y_one_hot[0])
```

```
↳ original label: 9
   after conversion to one_hot: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

```
from sklearn.model_selection import train_test_split
train_X,valid_X,train_label,valid_label=train_test_split(train_X,train_Y_one_hot,test_si:
```

```
train_X.shape,valid_X.shape,train_label.shape,valid_label.shape
```

```
↳ ((48000, 28, 28, 1), (12000, 28, 28, 1), (48000, 10), (12000, 10))
```

```
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU,PRELU
```

```
batch_size=64
epochs=20
num_classes=10
```

```
fashion_model=Sequential()
fashion_model.add(Conv2D(32,kernel_size=(3,3),activation='linear',input_shape=(28,28,1),l
fashion_model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
fashion_model.add(Conv2D(64,(3,3),activation='linear',padding='same'))
fashion_model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
fashion_model.add(Conv2D(128,(3,3),activation='linear',padding='same'))
fashion_model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
fashion_model.add(Flatten())
fashion_model.add(Dense(128,activation='linear'))
fashion_model.add(Dense(num_classes,activation='softmax'))
```

```
fashion_model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimiz
```

```
fashion_model.summary()
```

```
↳
```

```
fashion_train=fashion_model.fit(train_X,train_label,batch_size=batch_size,epochs=epochs,
```

W0615 17:29:49.448983 140106155558784 deprecation.py:323] From /usr/local/lib/python3.6/tensorflow/python/ops/stack\_ops.py:119: tf.nn.conv2d (from tensorflow.python.ops.nn\_ops) is deprecated and will be removed in a future version. Instructions for updating:  
Use tf.nn.conv2d in 2.0, which has the same broadcast rule as np.conv2d.  
W0615 17:29:49.529005 140106155558784 deprecation\_wrapper.py:119] From /usr/local/lib/python3.6/tensorflow/python/ops/stack\_ops.py:119: tf.nn.conv2d (from tensorflow.python.ops.nn\_ops) is deprecated and will be removed in a future version. Instructions for updating:  
Use tf.nn.conv2d in 2.0, which has the same broadcast rule as np.conv2d.

Train on 48000 samples, validate on 12000 samples

```
Epoch 1/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.4485 - acc: 0.1000
Epoch 2/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.2936 - acc: 0.1875
Epoch 3/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.2490 - acc: 0.2500
Epoch 4/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.2129 - acc: 0.3125
Epoch 5/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.1881 - acc: 0.3750
Epoch 6/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.1610 - acc: 0.4375
Epoch 7/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.1406 - acc: 0.5000
Epoch 8/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.1185 - acc: 0.5625
Epoch 9/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.1036 - acc: 0.6250
Epoch 10/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0896 - acc: 0.6875
Epoch 11/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0839 - acc: 0.7500
Epoch 12/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0705 - acc: 0.8125
Epoch 13/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0622 - acc: 0.8750
Epoch 14/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0588 - acc: 0.9375
Epoch 15/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0535 - acc: 0.9375
Epoch 16/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0430 - acc: 0.9375
Epoch 17/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0481 - acc: 0.9375
Epoch 18/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0497 - acc: 0.9375
Epoch 19/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0391 - acc: 0.9375
Epoch 20/20
48000/48000 [=====] - 87s 2ms/step - loss: 0.0375 - acc: 0.9375
```

```
test_eval=fashion_model.evaluate(test_X,test_Y_one_hot,verbose=0)
```

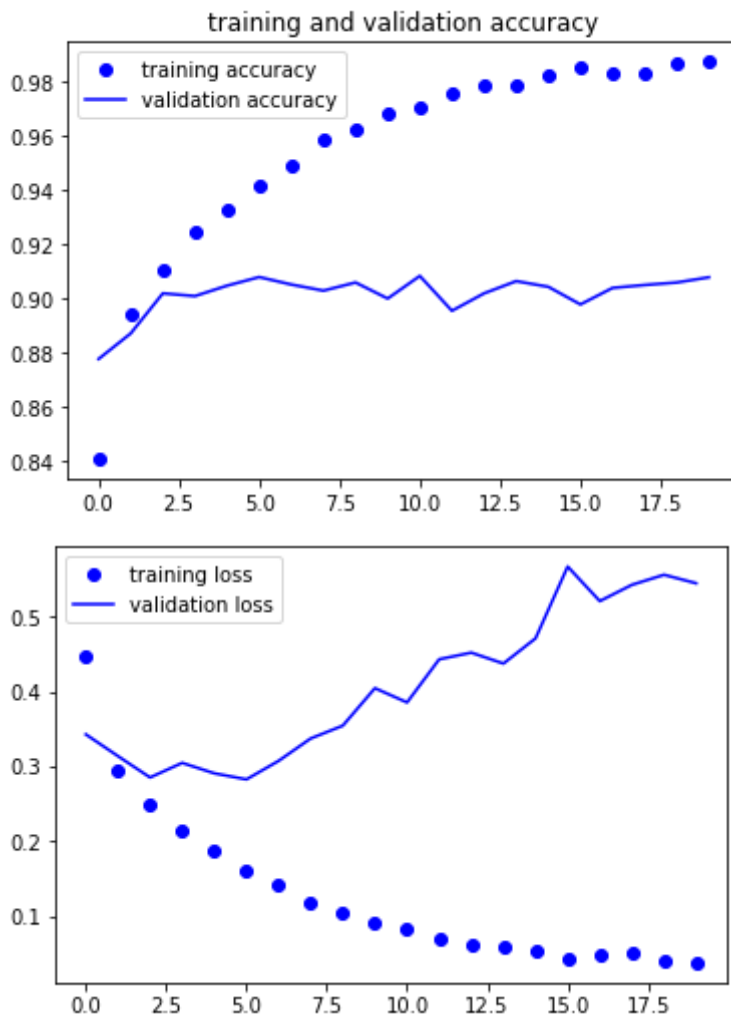
```
print('test loss:',test_eval[0])
print('test accuracy:',test_eval[1])
```

↳

```

accuracy=fashion_train.history['acc']
val_accuracy=fashion_train.history['val_acc']
loss=fashion_train.history['loss']
val_loss=fashion_train.history['val_loss']
epochs=range(len(accuracy))
plt.plot(epochs,accuracy,'bo',label='training accuracy')
plt.plot(epochs,val_accuracy,'b',label='validation accuracy')
plt.title('training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs,loss,'bo',label='training loss')
plt.plot(epochs,val_loss,'b',label='validation loss')
plt.legend()
plt.show()

```



```

batch_size=64
epochs=20
num_classes=10

```

```

fashion_model=Sequential()
fashion_model.add(Conv2D(32,kernel_size=(3,3),activation='linear',input_shape=(28,28,1),))
fashion_model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
fashion_model.add(Dropout(0.25))
fashion_model.add(Conv2D(64,(3,3),activation='linear',padding='same'))
fashion_model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
fashion_model.add(Dropout(0.25))
fashion_model.add(Conv2D(128,(3,3),activation='linear',padding='same'))
fashion_model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
fashion_model.add(Dropout(0.4))
fashion_model.add(Flatten())
fashion_model.add(Dense(128,activation='linear'))
fashion_model.add(Dropout(0.3))

```

```
fashion_model.add(Dense(num_classes,activation='softmax'))
```

```
fashion_model.summary()
```

↗

Layer (type)	Output Shape	Param #
conv2d_31 (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d_28 (MaxPooling)	(None, 14, 14, 32)	0
dropout_4 (Dropout)	(None, 14, 14, 32)	0
conv2d_32 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_29 (MaxPooling)	(None, 7, 7, 64)	0
dropout_5 (Dropout)	(None, 7, 7, 64)	0
conv2d_33 (Conv2D)	(None, 7, 7, 128)	73856
max_pooling2d_30 (MaxPooling)	(None, 4, 4, 128)	0
dropout_6 (Dropout)	(None, 4, 4, 128)	0
flatten_9 (Flatten)	(None, 2048)	0
dense_10 (Dense)	(None, 128)	262272
dropout_7 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 10)	1290
Total params: 356,234		
Trainable params: 356,234		
Non-trainable params: 0		

```
fashion_model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adam())
```

```
fashion_train_dropout=fashion_model.fit(train_X,train_label,batch_size=batch_size,epochs=10)
```



Train on 48000 samples, validate on 12000 samples

```
Epoch 1/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.5484 - acc:
Epoch 2/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.3833 - acc:
Epoch 3/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.3475 - acc:
Epoch 4/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.3257 - acc:
Epoch 5/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.3117 - acc:
Epoch 6/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.3012 - acc:
Epoch 7/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.2924 - acc:
Epoch 8/20
48000/48000 [=====] - 100s 2ms/step - loss: 0.2833 - acc:
Epoch 9/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.2759 - acc:
Epoch 10/20
48000/48000 [=====] - 99s 2ms/step - loss: 0.2726 - acc:
Epoch 11/20
48000/48000 [=====] - 100s 2ms/step - loss: 0.2678 - acc:
Epoch 12/20
48000/48000 [=====] - 100s 2ms/step - loss: 0.2634 - acc:
```

```
fashion_model.save("fashion_model_dropout.h5py")
```

```
test_eval=fashion_model.evaluate(test_X,test_Y_one_hot,verbose=1)
```

```
10000/10000 [=====] - 5s 525us/step
48000/48000 [=====] - 100s 2ms/step - loss: 0.2734 - acc:
```

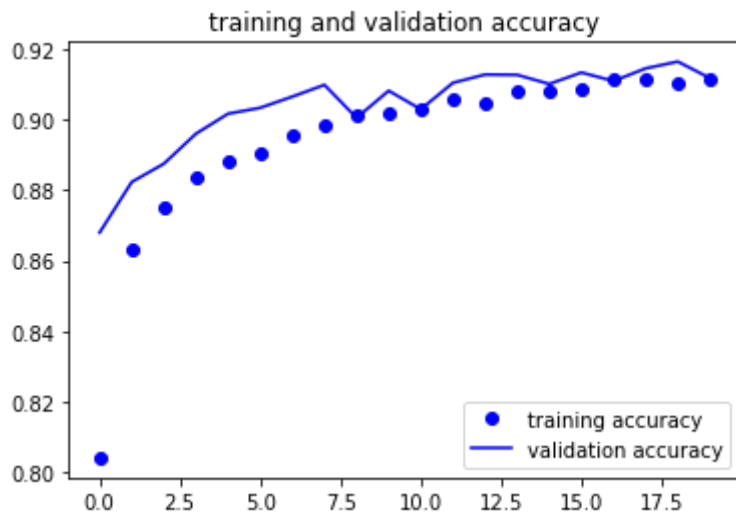
```
print('test loss:',test_eval[0])
print('test accuracy:',test_eval[1])
```

```
test loss: 0.2723303992450237
test accuracy: 0.905
```

```
Epoch 20/20
```

```
accuracy=fashion_train_dropout.history['acc']
val_accuracy=fashion_train_dropout.history['val_acc']
loss=fashion_train_dropout.history['loss']
val_loss=fashion_train_dropout.history['val_loss']
epochs=range(len(accuracy))
plt.plot(epochs,accuracy,'bo',label='training accuracy')
plt.plot(epochs,val_accuracy,'b',label='validation accuracy')
plt.title('training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs,loss,'bo',label='training loss')
plt.plot(epochs,val_loss,'b',label='validation loss')
plt.title('training and validation loss')
plt.legend()
plt.show()
```

```
↳
```



```
predicted_classes=fashion_model.predict(test_X)
```

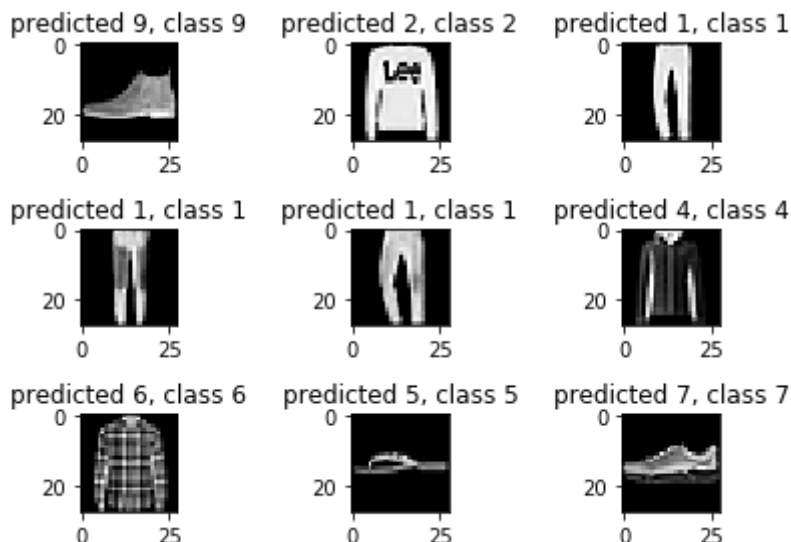
```
predicted_classes=np.argmax(np.round(predicted_classes),axis=1)
```

```
predicted_classes.shape,test_Y.shape
```

```
↳ ((10000,), (10000,))
```

```
correct=np.where(predicted_classes==test_Y)[0]
print("Found %d correct labels" % len(correct))
for i,correct in enumerate(correct[:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(test_X[correct].reshape(28,28),cmap='gray',interpolation='none')
    plt.title("predicted {}, class {}".format(predicted_classes[correct],test_Y[correct]))
    plt.tight_layout()
```

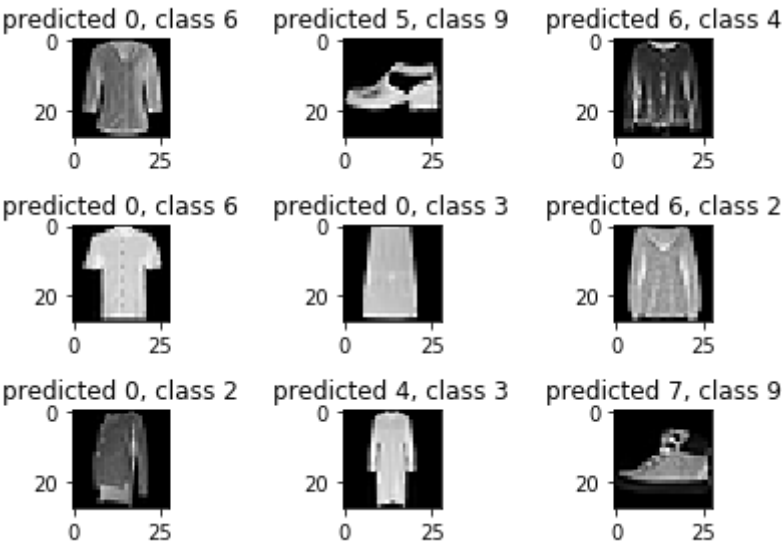
```
↳ Found 8999 correct labels
```





```
incorrect=np.where(predicted_classes!=test_Y)[0]
print("Found %d incorrect labels" % len(incorrect))
for i,incorrect in enumerate(incorrect[:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(test_X[incorrect].reshape(28,28),cmap='gray',interpolation='none')
    plt.title("predicted {}, class {}".format(predicted_classes[incorrect],test_Y[incorrect]))
plt.tight_layout()
```

Found 1001 incorrect labels



```
from sklearn.metrics import classification_report
target_names=["class{}".format(i) for i in range(num_classes)]
print(classification_report(test_Y,predicted_classes,target_names=target_names))
```

	precision	recall	f1-score	support
class0	0.70	0.95	0.81	1000
class1	0.99	0.98	0.99	1000
class2	0.83	0.88	0.85	1000
class3	0.94	0.88	0.91	1000
class4	0.83	0.86	0.85	1000
class5	0.99	0.97	0.98	1000
class6	0.89	0.55	0.68	1000
class7	0.95	0.98	0.96	1000
class8	0.99	0.98	0.99	1000
class9	0.97	0.96	0.97	1000
accuracy			0.90	10000
macro avg	0.91	0.90	0.90	10000
weighted avg	0.91	0.90	0.90	10000

