Identify the Apparels

More than 25% of entire revenue in E-Commerce is attributed to apparels & accessories. A major problem they face is categorizing these apparels from just the images especially when the categories provided by the brands are inconsistent. This poses an interesting computer vision problem which has caught the eyes of several deep learning researchers.

Fashion MNIST is a drop-in replacement for the very well known, machine learning hello world - MNIST dataset which can be checked out at 'Identify the digits' practice problem. Instead of digits, the images show a type of apparel e.g. T-shirt, trousers, bag, etc.

```
!wget --header="Host: datahack-prod.s3.amazonaws.com" --header="User-Agent: Mozilla/5.0 (W
```

```
--2020-10-30 20:07:19--  https://datahack-prod.s3.amazonaws.com/train_zip/train_LbEL1
Resolving datahack-prod.s3.amazonaws.com (datahack-prod.s3.amazonaws.com)... 52.219.6
Connecting to datahack-prod.s3.amazonaws.com (datahack-prod.s3.amazonaws.com)|52.219
HTTP request sent, awaiting response... 206 Partial Content
Length: 82498602 (79M), 19584042 (19M) remaining [application/zip]
Saving to: 'train_LbELtWX.zip'

train_LbELtWX.zip   100%[+++++++++++++++++====>]  78.68M  6.22MB/s    in 3.0s

2020-10-30 20:07:23 (6.22 MB/s) - 'train_LbELtWX.zip' saved [82498602/82498602]
```

```
!7z x train_LbELtWX.zip
```

```
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R)

Scanning the drive for archives:
1 file, 102082644 bytes (98 MiB)

Extracting archive: train_LbELtWX.zip

WARNINGS:
There are data after the end of archive

--
Path = train_LbELtWX.zip
Type = zip
WARNINGS:
There are data after the end of archive
Physical Size = 82498602
Tail Size = 19584042

Everything is Ok

Archives with Warnings: 1
```

```
Warnings: 1
Folders: 1
Files: 60001
Size:       73348466
Compressed: 102082644
```

```
!unzip test_ScVgIM0.zip
```

```
!ls
```

```
sample_data  test.csv          train      train_LbELtWX.zip
test         test_ScVgIM0.zip  train.csv
```

```
import pandas as pd
import numpy as np
from skimage.io import imread
from skimage.transform import resize
import warnings
warnings.filterwarnings("ignore")
#from __future__ import print_function
exec('from __future__ import absolute_import, division, print_function')
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

```
train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')
```

```
train.head()
```

	id	label
0	1	9
1	2	0
2	3	0
3	4	3
4	5	0

```
test.head()
```

	id
0	60001
1	60002

```
batch_size=128
num_classes=train.label.nunique()
y_train=train.label
y_train=keras.utils.to_categorical(y_train,num_classes)
```

```
import os
from tqdm import tqdm
train_path=os.path.join(os.getcwd(),'train/')
print(train_path)
test_path=os.path.join(os.getcwd(),'test/')
print(test_path)
```

```
/content/train/
/content/test/
```

```
train_img=[]
for i in tqdm(train['id']):
  img=str(i)+'.png'
  image=imread(train_path+img)
  image=image/255.
  image=resize(image,(28,28,1),mode='constant')
  image=image.astype('float')
  train_img.append(image)
```

```
100%|██████████| 60000/60000 [00:55<00:00, 1072.19it/s]
```

```
test_img=[]
for i in tqdm(test['id']):
  img=str(i)+'.png'
  image=imread(test_path+img)
  image=image/255.
  image=resize(image,(28,28,1),mode='constant')
  image=image.astype('float')
  test_img.append(image)
test_img=np.array(test_img)
test_img.shape
```

```
100%|██████████| 10000/10000 [00:09<00:00, 1051.40it/s]
(10000, 28, 28, 1)
```

```
np.save('train_img.npy',train_img)
np.save('test_img.npy',test_img)
```

```
X_train=np.load('./train_img.npy',allow_pickle=True)
X_test=np.load('./test_img.npy',allow_pickle=True)
```

```
# Network Architecture
# input -> conv -> conv -> pooling -> conv -> conv -> pooling ->dropout-> FC -> output
# 16 16 32 32 512

inp_shape=X_test.shape[1:]
model=Sequential()
model.add(Conv2D(16,kernel_size=(3,3),padding='same',activation='relu',input_shape=inp_sha
model.add(Conv2D(16,5,padding='same',activation='relu'))
model.add(MaxPooling2D(strides=2))
model.add(Conv2D(32,5,activation='relu',padding='same'))
model.add(Conv2D(32,5,activation='relu',padding='same'))
model.add(MaxPooling2D(strides=2))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dense(num_classes,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 28, 28, 16)        160
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 16)        6416
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 14, 14, 32)        12832
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 14, 14, 32)        25632
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               803328
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 853,498
Trainable params: 853,498
Non-trainable params: 0
_________________________________________________________________
```

```
history2=model.fit(X_train,y_train,epochs=10,batch_size=256,verbose=1,validation_split=0.2
```

```
Epoch 1/10
188/188 [==============================] - 184s 978ms/step - loss: 0.6494 - accuracy
Epoch 2/10
188/188 [==============================] - 181s 964ms/step - loss: 0.3607 - accuracy
Epoch 3/10
188/188 [==============================] - 179s 952ms/step - loss: 0.3051 - accuracy
```

```
Epoch 4/10
188/188 [==============================] - 183s 975ms/step - loss: 0.2735 - accuracy
Epoch 5/10
188/188 [==============================] - 179s 951ms/step - loss: 0.2490 - accuracy
Epoch 6/10
188/188 [==============================] - 180s 956ms/step - loss: 0.2337 - accuracy
Epoch 7/10
188/188 [==============================] - 185s 982ms/step - loss: 0.2114 - accuracy
Epoch 8/10
188/188 [==============================] - 182s 967ms/step - loss: 0.1959 - accuracy
Epoch 9/10
188/188 [==============================] - 181s 964ms/step - loss: 0.1812 - accuracy
Epoch 10/10
188/188 [==============================] - 183s 972ms/step - loss: 0.1647 - accuracy
```

```
pred =np.array(model.predict(X_test))
predictions=[]
for i in pred:
  predictions.append(np.argmax(i))


sub=test['id']


sub.head()
```

```
0    60001
1    60002
2    60003
3    60004
4    60005
Name: id, dtype: int64
```

```
sub['label']=predictions


predict = pd.DataFrame(data=predictions ,columns=["label"])



sub = test['id']
DT = pd.merge(sub , predict, on=None, left_index= True,
    right_index=True)


DT.to_csv('brahm_submssion_mnist.csv',index=False)
```