Module 1 (Pose detection)

In this module we detect does image have full pose shots of humans.

Pose Detection with tensorflow lite

- https://www.tensorflow.org/lite/models/pose_estimation/overview
- https://medium.com/roonyx/pose-estimation-and-matching-with-tensorflow-lite-posenet-model-ea2e9249abbd **(code Referenced link)**

Pose estimation refers to computer vision techniques that detect human figures in images and videos, so that one could determine, for example, where someone's elbow shows up in an image. It is important to be aware of the fact that pose estimation merely estimates where key body joints are and does not recognize who is in an image or video.

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/53
7.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-
Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7" --header="Referer: https://www.kaggle.c
om/" "https://storage.googleapis.com/kaggle-data-sets/930393/1613771/bundle/archive.zip?X
-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gse
rviceaccount.com%2F20210326%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20210326T054515Z
&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-Signature=2d8f49dc64d3eee28463558
5e324a7f27863186954091e43ed1c451d784d564249ee1a468c6cd9eee3cd6183cc3b5992b25ed83a05215c25
5c0488c05a16f4b481d0a9b84015ee298c9b5b83769d10859a51a07582b4b3ca435d3cc61a923cd22f4f3b929
a3f095cd989ba53826954b092fd9f8ca82b99bccdea4bf6f0250d09cb6e3ed8d426a6e79d38e4083d42b0d5bb
21d673c7d7c699ccfe7919459080d5bd1c3ff59de8e8083cab6098e31a1bc0b5cb93624da56b79af3d6dfb001
0cf2468e48f7bbcca671071fcf9d7506c6bcee5f16aec6ab3ba9a88b45d23ba3b342e190754cbd187a10721a1
b9da9223e60b6d03a1f9c089c940051b2062ba94f9e6" -c -O 'archive.zip'
```

```
--2021-03-26 05:45:32--  https://storage.googleapis.com/kaggle-data-sets/930393/1613771/b
undle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40ka
ggle-161607.iam.gserviceaccount.com%2F20210326%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Da
te=20210326T054515Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-Signature=2d8f
49dc64d3eee284635585e324a7f27863186954091e43ed1c451d784d564249ee1a468c6cd9eee3cd6183cc3b5
992b25ed83a05215c255c0488c05a16f4b481d0a9b84015ee298c9b5b83769d10859a51a07582b4b3ca435d3c
c61a923cd22f4f3b929a3f095cd989ba53826954b092fd9f8ca82b99bccdea4bf6f0250d09cb6e3ed8d426a6e
79d38e4083d42b0d5bb21d673c7d7c699ccfe7919459080d5bd1c3ff59de8e8083cab6098e31a1bc0b5cb9362
4da56b79af3d6dfb0010cf2468e48f7bbcca671071fcf9d7506c6bcee5f16aec6ab3ba9a88b45d23ba3b342e1
90754cbd187a10721a1b9da9223e60b6d03a1f9c089c940051b2062ba94f9e6
Resolving storage.googleapis.com (storage.googleapis.com)... 209.85.147.128, 142.250.136.
128, 209.85.200.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|209.85.147.128|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 11894512126 (11G) [application/zip]
Saving to: 'archive.zip'

archive.zip         100%[===================>]  11.08G  38.4MB/s    in 4m 26s

2021-03-26 05:49:59 (42.6 MB/s) - 'archive.zip' saved [11894512126/11894512126]

time: 4min 30s (started: 2021-03-26 05:45:32 +00:00)
```

In [ ]:

```
import pandas as pd
position = []
for i in range(17):
  position.append(i)
```

```
part = ["nose","leftEye","rightEye","leftEar","rightEar","leftShoulder","rightShoulder","
leftElbow","rightElbow","leftWrist","rightWrist","leftHip","rightHip","leftKnee","rightKn
ee","leftAnkle","rightAnkle"]

position_df = pd.DataFrame(list(zip(position, part)),
               columns =['id', 'part'])

position_df
```

Out[ ]:

	id	part
0	0	nose
1	1	leftEye
2	2	rightEye
3	3	leftEar
4	4	rightEar
5	5	leftShoulder
6	6	rightShoulder
7	7	leftElbow
8	8	rightElbow
9	9	leftWrist
10	10	rightWrist
11	11	leftHip
12	12	rightHip
13	13	leftKnee
14	14	rightKnee
15	15	leftAnkle
16	16	rightAnkle

In [ ]:

```
!unzip archive.zip

!rm -rf archive.zip
```

In [ ]:

```
import tensorflow as tf
```

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/53
7.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-
Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7" --header="Referer: https://www.tensorfl
ow.org/" "https://storage.googleapis.com/download.tensorflow.org/models/tflite/posenet_mo
bilenet_v1_100_257x257_multi_kpt_stripped.tflite" -c -O 'posenet_mobilenet_v1_100_257x257
_multi_kpt_stripped.tflite'
```

```
--2021-04-22 04:14:57--  https://storage.googleapis.com/download.tensorflow.org/models/tf
lite/posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.73.208, 142.250.65.8
0, 172.253.62.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.73.208|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 13269068 (13M) [application/octet-stream]
Saving to: 'posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite'
```

```
posenet_mobilenet_v 100%[===================>]  12.65M  53.5MB/s    in 0.2s

2021-04-22 04:14:58 (53.5 MB/s) - 'posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tf
lite' saved [13269068/13269068]
```

Code Referrence :

https://www.tensorflow.org/api_docs/python/tf/lite/Interpreter

https://programmer.group/analysis-of-official-post-energy-model-of-tensorflow.html

In [ ]:

```
pose_detector = tf.lite.Interpreter(model_path="posenet_mobilenet_v1_100_257x257_multi_kp
t_stripped.tflite")
pose_detector.allocate_tensors()

# Get input and output tensors information from the model file
input_details = pose_detector.get_input_details()
# from input details it's clear that we have shape of 257x257 with dtype float32 [ 'shape
_signature': array([  1, 257, 257,   3], dtype=int32), 'dtype': <class 'numpy.float32'>]
output_details = pose_detector.get_output_details()
# output details [ 'shape_signature': array([ 1,   9,   9, 17], dtype=int32), 'dtype': <cla
ss 'numpy.float32'> ]
```

In [ ]:

```
def heat_offset(path):
    '''
    Input: path of the image
    Output: HeatMap, Offsets'''
    input_image = cv.imread(path)
    input_image = cv.resize(input_image, (257, 257))
    # (257, 257, 3)
    input_image = np.expand_dims(input_image.copy(), axis=0)
    # (1, 257, 257, 3)

    input_image = (np.float32(input_image) - 127.5) / 127.5
    # Sets the value of the input tensor
    pose_detector.set_tensor(input_details[0]['index'], input_image)
    # Runs the computation
    pose_detector.invoke()
    #output data and offset data
    heatmap_data = pose_detector.get_tensor(output_details[0]['index'])
    offset_data = pose_detector.get_tensor(output_details[1]['index'])
    # Getting rid of the extra dimension
    heatmaps = np.squeeze(heatmap_data)
    offsets = np.squeeze(offset_data)
    return heatmaps,offsets
```

In [ ]:

```
def pose_detection(heat_map,offset_data, threshold):

  '''
  Input:
    heat_map - hetmaps for an image. Three dimension array
    offset_data - offset vectors for an image. Three dimension array
    threshold - probability threshold for the keypoints. Scalar value
  Output:
    array with coordinates of the keypoints and flags for those that havelow probability
    Condition : If nose and (two eye) and ( one shoulder ) and (one hip) and (one ankle)
is detected , then it will return TRUE
  '''

  key_points = 17
  # as we have 17 key points like nose, shoulder ..etc;
  pose_points = np.zeros((key_points,3), np.uint32)
  for i in range(17):

      joint_heatmap = heat_map[...,i]
      pos_with_max_val = np.squeeze(np.argwhere(joint_heatmap==np.max(joint_heatmap)))
      #Remove axes of length one from a
      remap_pos = np.array(pos_with_max_val/8*257,dtype=np.int32)
      pose_points[i,0] = int(remap_pos[0] + offset_data[pos_with_max_val[0],pos_with_max
_val[1],i])
      pose_points[i,1] = int(remap_pos[1] + offset_data[pos_with_max_val[0],pos_with_max
_val[1],i+key_points])
      max_prob = np.max(joint_heatmap)

      if max_prob > threshold:
        if pose_points[i,0] < 257 and pose_points[i,1] < 257:
          pose_points[i,2] = 1
      bool = (pose_points[0][2]==1) and ((pose_points[1][2] and pose_points[2][2]) ==1)
and ((pose_points[5][2] or pose_points[6][2]) ==1) and ((pose_points[11][2] or pose_poin
ts[12][2]) ==1) and ((pose_points[15][2] or pose_points[16][2]) ==1)
  return bool
```

In [ ]:

```
def final(path):
    '''
    Input: path of the Image
    output: Bool true or flase
    Explanation : it takes path and it sends to model to get the
    output and it sends to parse output function to get wheater it is a full pose or not'
''

    heatmaps,offsets = heat_offset(path)
    bool = pose_detection(heatmaps,offsets,0.7)
    return bool
```

In [ ]:

```
def final_custom(path):
    '''
    Input: Image path
    output: Provided image is a full pose or not
    This function will the plot images based on condition. full pose or not. This is visu
lazie the results for Module_1
    '''
    bool = final(path)
    if bool:
        print("Full Pose Image")
        image=cv.imread(path)
        plt.figure(figsize=(6,6))
        plt.imshow(image)
        plt.show()
        print("\n")
    else:
        print("Not Full-Pose-Image")
        image=cv.imread(path)
        plt.figure(figsize=(6,6))
        plt.imshow(image)
        plt.show()
        print("\n")
```

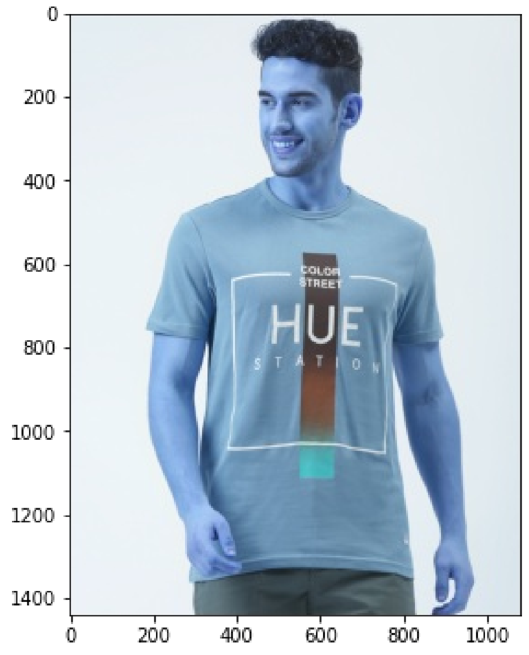
```
time: 7.65 ms (started: 2021-03-26 09:49:56 +00:00)
```

In [ ]:

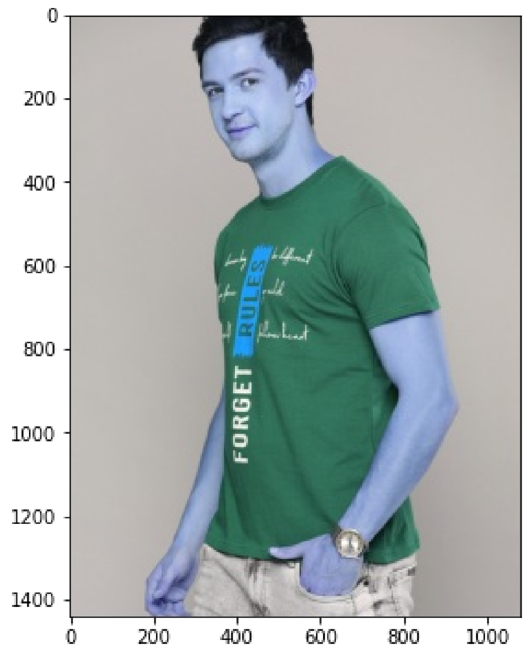
```
import os
path='/content/test'

for image in os.listdir(path):
  Image = os.path.join(path,image)
  final_custom(Image)
```

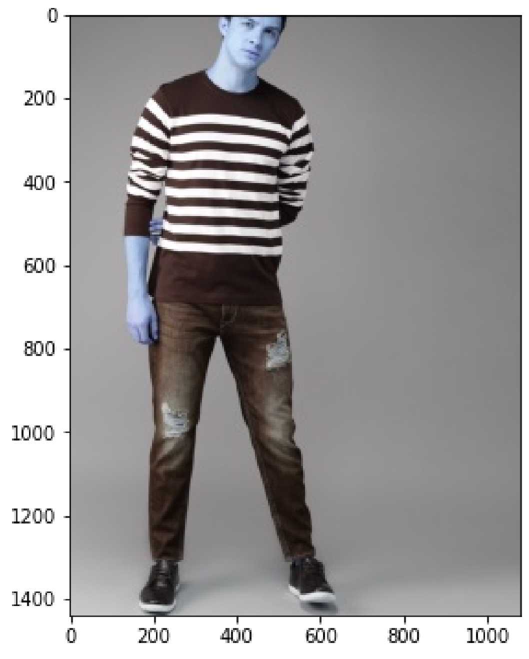
```
Not Full-Pose-Image
```



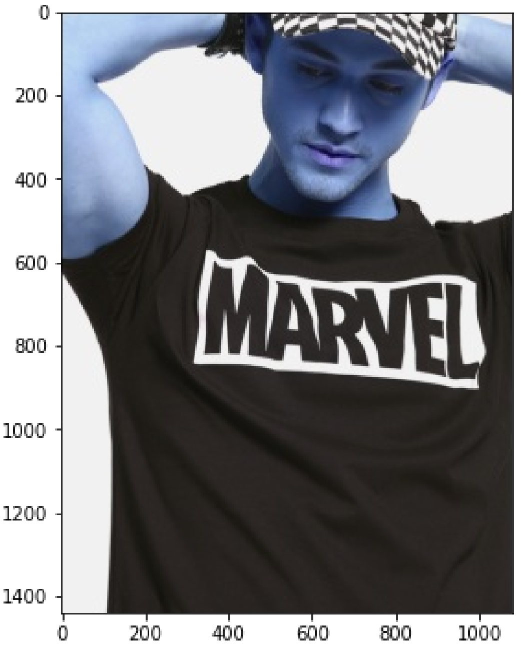
```
Not Full-Pose-Image
```



```
Full Pose Image
```

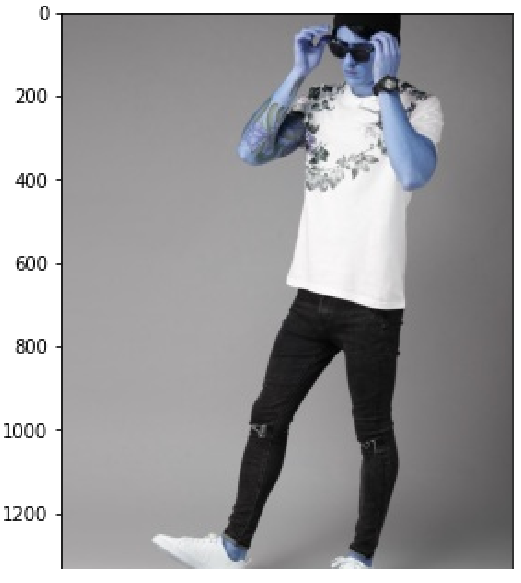


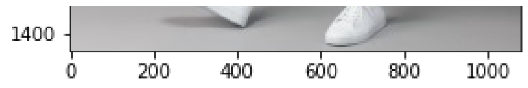
```
Not Full-Pose-Image
```



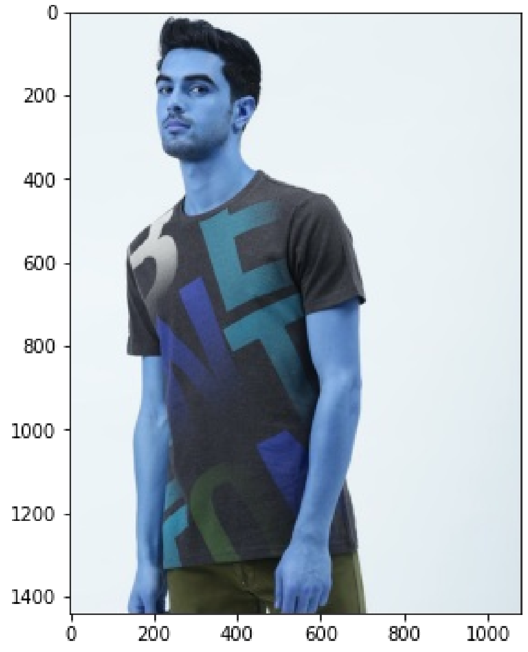
```
Not Full-Pose-Image
```

```
Not Full-Pose-Image
```

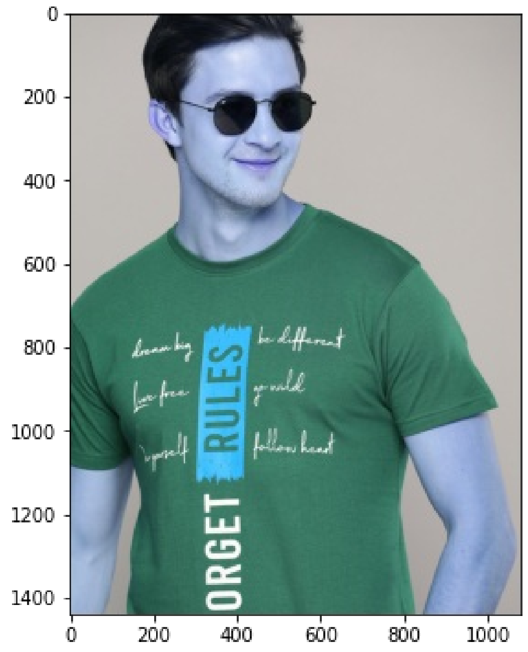




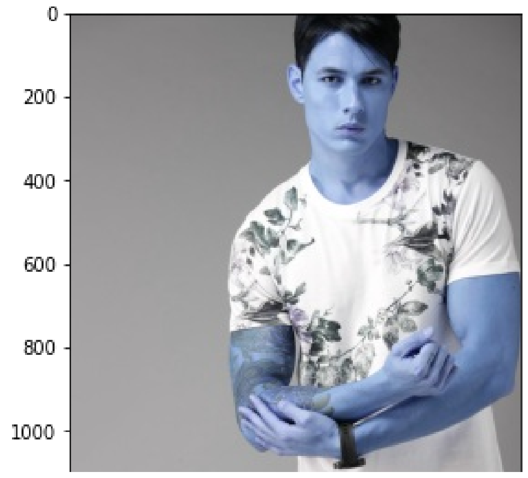
```
Not Full-Pose-Image
```

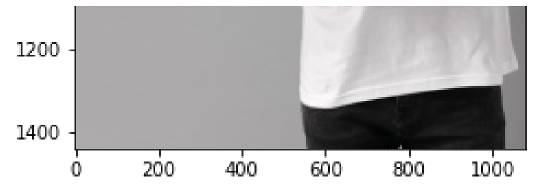


```
Not Full-Pose-Image
```

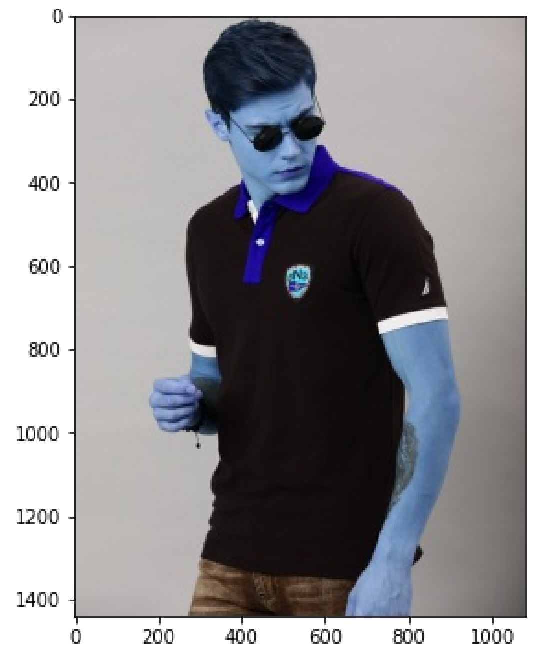


```
Not Full-Pose-Image
```

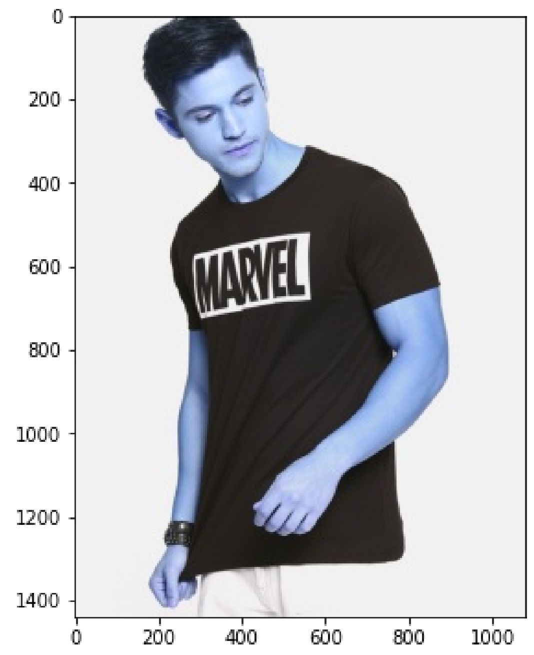




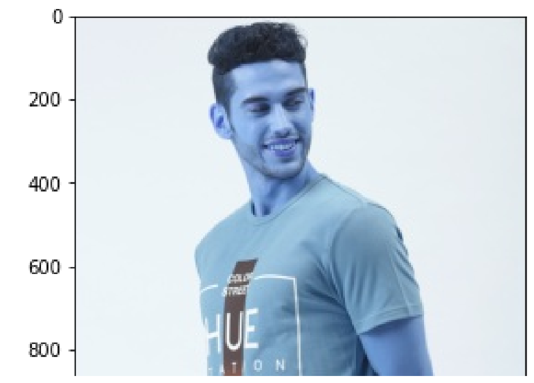
```
Not Full-Pose-Image
```

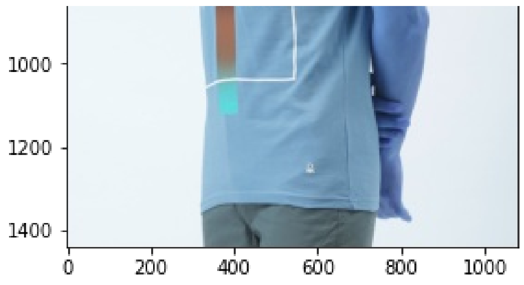


```
Not Full-Pose-Image
```

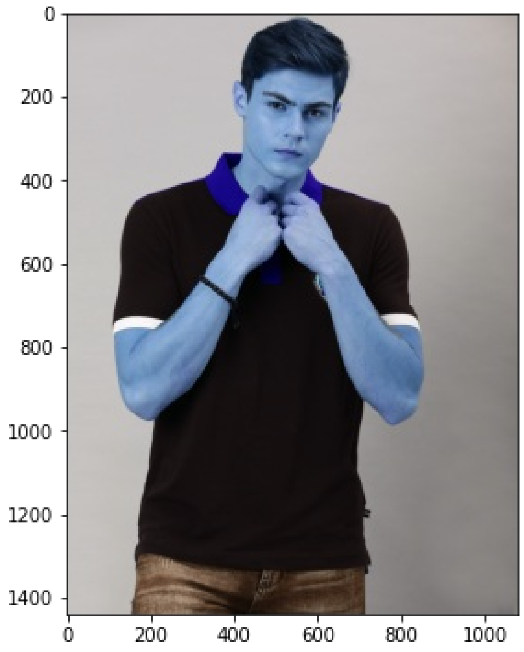


```
Not Full-Pose-Image
```

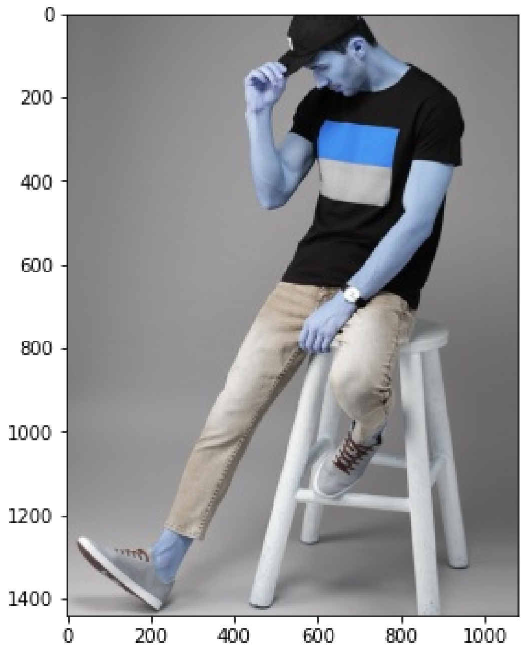




```
Not Full-Pose-Image
```



```
Not Full-Pose-Image
```



Out[ ]:

```
'for i in image:\n    pa = os.path.join(path,i)\n    final_custom(pa)'
```

```
time: 6.12 s (started: 2021-03-26 09:54:19 +00:00)
```

EDA

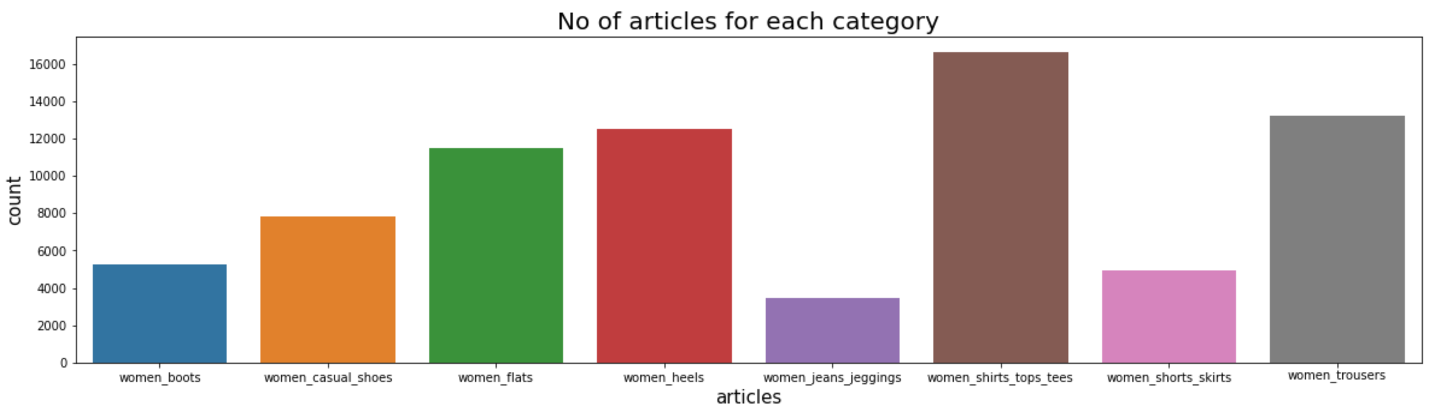
In [ ]:

```
def count_images(path):
    count=0
    for json in os.listdir(path):
        if json.split('.')[1]=='json':
            count = count+1
    return count
women_boots                = count_images('women_boots')
women_casual_shoes         = count_images('women_casual_shoes')
women_flats                = count_images('women_flats')
women_heels                = count_images('women_heels')
women_jeans_jeggings       = count_images('women_jeans_jeggings')
women_shirts_tops_tees     = count_images('women_shirts_tops_tees')
women_shorts_skirts        = count_images('women_shorts_skirts')
women_trousers             = count_images('women_trousers')
```

```
time: 74.3 ms (started: 2021-03-26 08:31:37 +00:00)
```

In [ ]:

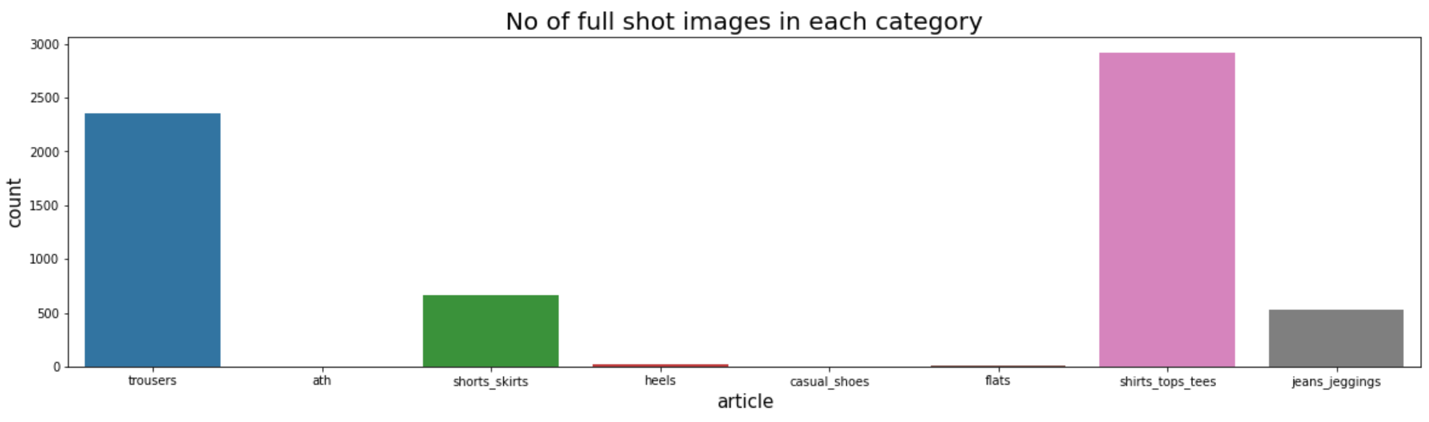
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(20,5))
sns.barplot(
    ['women_boots','women_casual_shoes','women_flats','women_heels','women_jeans_jeggings
','women_shirts_tops_tees','women_shorts_skirts','women_trousers'],
     [women_boots,women_casual_shoes,women_flats,women_heels,women_jeans_jeggings,women_
shirts_tops_tees,women_shorts_skirts,women_trousers])
plt.title('No of articles for each category',fontsize=20)
plt.xlabel("articles",fontsize=15)
plt.ylabel('count',fontsize=15)
plt.show()
```



```
time: 207 ms (started: 2021-03-26 09:36:48 +00:00)
```

In [ ]:

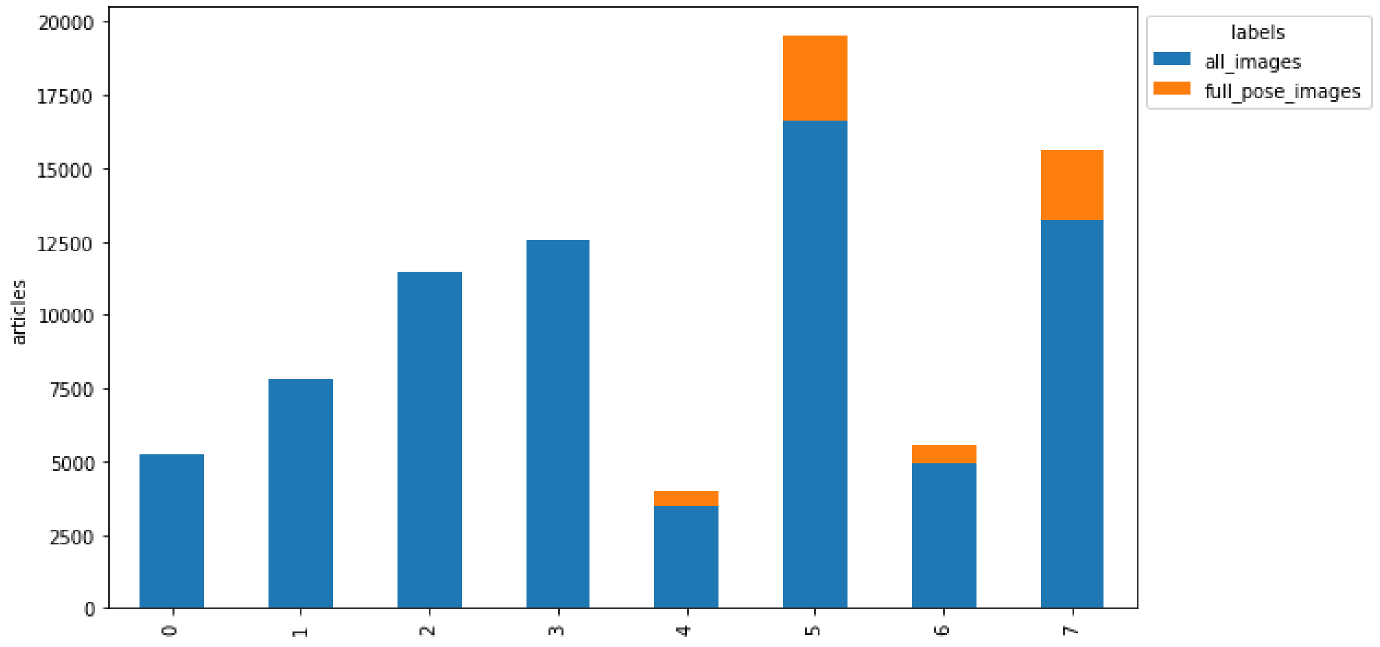
```
plt.figure(figsize=(20,5))
sns.barplot(list(full_pose.keys()),list(full_pose.values()))
plt.title('No of full shot images in each category',fontsize=20)
plt.xlabel("article",fontsize=15)
plt.ylabel('count',fontsize=15)
plt.show()
```



```
time: 190 ms (started: 2021-03-26 09:36:28 +00:00)
```

```
In [ ]:
```

```python
ax = articles_.plot(kind='bar', stacked=True, figsize=(10, 6))
ax.set_ylabel('articles')
plt.legend(title='labels', bbox_to_anchor=(1.0, 1), loc='upper left')
# plt.savefig('stacked.png')  # if needed
plt.show()
```



```
time: 194 ms (started: 2021-03-26 09:48:33 +00:00)
```