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.

Import required libraries

In [ ]:

```
# We should be able to directly predict using saved model so I am again importing all mod
ules
!pip install tensorflow==1.15.0
```

In [ ]:

```
!!pip install keras==2.1.5
```

In [1]:

```
import tensorflow as tf
import pandas as pd
```

In [3]:

```
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[3]:

	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

	id	part
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

Import required tensorflow lite pose model

In [4]:

```
!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/89.0.4389.128 Safari/537.36" --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.tensorflow.org/" "https://storage.googleapis.com/download.tensorflow.org/models/tflite/posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite" -c -O 'posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite'
```

```
--2021-04-27 03:57:02--  https://storage.googleapis.com/download.tensorflow.org/models/tflite/posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.11.128, 74.125.31.128, 74.125.141.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.11.128|:443... connected.
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  67.7MB/s    in 0.2s

2021-04-27 03:57:03 (67.7 MB/s) - ‘posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite’ 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 [5]:

```
pose_detector = tf.lite.Interpreter(model_path="posenet_mobilenet_v1_100_257x257_multi_kpt_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': <class 'numpy.float32'> ]
```

In [6]:

```
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

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_kps[i,0] < 257 and pose_kps[i,1] < 257:
          pose_kps[i,2] = 1
      bool = (pose_kps[0][2]==1) and ((pose_kps[1][2] or pose_kps[2][2]) ==1) and ((pose
_kps[11][2] or pose_kps[12][2]) ==1) and ((pose_kps[15][2] or pose_kps[16][2]) ==1)
  return bool
```

Image that are detected as full pose images will be passed to second object localization module training.

Module 2 (Modue-2 Article detection and Localization of Clothes)

In this module we use object detection techniques to localizalize fashion objects in the images

https://www.kaggle.com/pednoi/training-mask-r-cnn-to-be-a-fashionista-lb-0-07

I took Mark Rcnn is because, it is a kind of widely used in Instance segmentation

Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.

This tutorial requires TensorFlow version 1.15.3 and Keras 2.2.4. It does not work with TensorFlow 2.0+ or Keras 2.2.5+ because a third-party library has not been updated at the time of writing.

!pip install --no-deps tensorflow==1.15.0

!pip install --no-deps keras==2.1.5

In [ ]:

```
import tensorflow as tf
import keras
```

```
Using TensorFlow backend.
```

In [ ]:

```
print(tf.__version__)
print(keras.__version__)
```

```
1.15.0
2.1.5
```

In [ ]:

```
## Cloning the Mask_rcnn model from git
!git clone https://www.github.com/matterport/Mask_RCNN.git

## Changing the directroy to make use of existing python files for your code
os.chdir('Mask_RCNN')
!rm -rf .git # to prevent an error when the kernel is committed
!rm -rf images assets # to prevent displaying images at the bottom of a kernel
```

```
Cloning into 'Mask_RCNN'...
warning: redirecting to https://github.com/matterport/Mask_RCNN.git/
remote: Enumerating objects: 956, done.
remote: Total 956 (delta 0), reused 0 (delta 0), pack-reused 956
Receiving objects: 100% (956/956), 125.23 MiB | 33.20 MiB/s, done.
Resolving deltas: 100% (562/562), done.
```

In [ ]:

```
## Downloading the saved weights
!wget --header="Host: doc-10-5k-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36" --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="Cookie: AUTH_gnb78hdmdiks9t0b8kec09hpa7nncs5e_nonce=puni4p4qb0oi6" --header="Connection: keep-alive" "https://doc-10-5k-docs.googleusercontent.com/docs/securesc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/ie0rdgpscha71ofaar1rnr07j6ed489c/1619416350000/03515051603858730688/03515051603858730688/1vxhcFBgz7GrMZwZ-WV0LMkc8bqAcN5SQ?e=download&authuser=0&nonce=puni4p4qb0oi6&user=03515051603858730688&hash=d760jp9k9hda4mqvdg794s0a5t4t90fl" -c -O 'mask_rcnn_fashion_day_2_2_0004-0.32516.h5'
```

```
--2021-04-26 05:53:44--  https://doc-10-5k-docs.googleusercontent.com/docs/securesc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/ie0rdgpscha71ofaar1rnr07j6ed489c/1619416350000/03515051603858730688/03515051603858730688/1vxhcFBgz7GrMZwZ-WV0LMkc8bqAcN5SQ?e=download&authuser=0&nonce=puni4p4qb0oi6&user=03515051603858730688&hash=d760jp9k9hda4mqvdg794s0a5t4t90fl
Resolving doc-10-5k-docs.googleusercontent.com (doc-10-5k-docs.googleusercontent.com)... 74.125.142.132, 2607:f8b0:400e:c08::84
Connecting to doc-10-5k-docs.googleusercontent.com (doc-10-5k-docs.googleusercontent.com)|74.125.142.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/octet-stream]
Saving to: 'mask_rcnn_fashion_day_2_2_0004-0.32516.h5'

mask_rcnn_fashion_d     [       <=>             ] 171.80M   115MB/s    in 1.5s

2021-04-26 05:53:46 (115 MB/s) - 'mask_rcnn_fashion_day_2_2_0004-0.32516.h5' saved [180150208]
```

In [ ]:

```
# Root directory of the project
ROOT_DIR = os.getcwd()
# Import Mask RCNN
sys.path.append(ROOT_DIR) # To find local version of the library
custom_WEIGHTS_PATH = '/content/Mask_RCNN/mask_rcnn_fashion_day_2_2_0004-0.32516.h5'
%matplotlib inline
# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
```

In [ ]:

```
from mrcnn.config import Config
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
from mrcnn.model import log
```

In [ ]:

```
# https://github.com/matterport/Mask_RCNN/blob/master/mrcnn/config.py
# Overwarign the existing config file with our requirements
class FashionConfig(Config):
    NAME = "Article localization"  # Override in sub-classes

    # Number of classification classes (including background)
    NUM_CLASSES = 46 + 1 # +1 for the background class

    # NUMBER OF GPUs to use. When using only a CPU, this needs to be set to 1.
    GPU_COUNT = 1

    # Number of images to train with on each GPU. A 12GB GPU can typically
    # handle 2 images of 1024x1024px.
    # Adjust based on your GPU memory and image sizes. Use the highest
    # number that your GPU can handle for best performance.
    IMAGES_PER_GPU = 4 # a memory error occurs when IMAGES_PER_GPU is too high , 4 becau
se image size is 512x512

    # Backbone network architecture
    # Supported values are: resnet50, resnet101.
    BACKBONE = 'resnet50'

    # Image shape
    IMAGE_MIN_DIM = 512
    IMAGE_MAX_DIM = 512
    IMAGE_RESIZE_MODE = 'none'

    # Length of square anchor side in pixels
    RPN_ANCHOR_SCALES = (16, 32, 64, 128, 256)

    # Number of training steps per epoch
    # This doesn't need to match the size of the training set. Tensorboard
    STEPS_PER_EPOCH = 1000

    # Number of validation steps to run at the end of every training epoch.
    # A bigger number improves accuracy of validation stats, but slows
    # down the training.
    VALIDATION_STEPS = 200

config = FashionConfig()
```

In [ ]:

```
## updating confign file for testing
class InferenceConfig(FashionConfig):
  GPU_COUNT = 1
  IMAGES_PER_GPU = 1

inference_config = InferenceConfig()
```

```
# Loading the model
model = modellib.MaskRCNN(mode='inference',config=inference_config,model_dir=ROOT_DIR)
## Loading saved model
model_path ='/content/Mask_RCNN/mask_rcnn_fashion_day_2_2_0004-0.32516.h5'

print("Loading weights from ", model_path)
model.load_weights(model_path, by_name=True)
```

In [ ]:

```
# Fashion Categories from Json file
feet = ['shoe']
upperbody = ['cardigan',
 'collar',
 'epaulette',
 'hood',
 'jacket',
 'lapel',
 'neckline',
 'shirt, blouse',
 'sleeve',
 'sweater',
 'top, t-shirt, sweatshirt',
 'vest']

lowerbody = ['pants', 'pocket', 'shorts', 'skirt']
wholebody = ['cape', 'coat', 'dress', 'jumpsuit']
```

Referrence for cropping https://medium.com/swlh/buy-me-that-look-bb46174d26ea

In [ ]:

```
def resize_image(image_):
    image = cv2.imread(image_)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (512, 512), interpolation=cv2.INTER_AREA)
    return image


def roi_masks(masks, rois):
    mask_index = np.argsort(np.sum(masks.reshape(-1, masks.shape[-1]), axis=0))
    union_mask = np.zeros(masks.shape[:-1], dtype=bool)
    for mask in mask_index:
        masks[:, :, mask] = np.logical_and(masks[:, :, mask], np.logical_not(union_mask)
)
        union_mask = np.logical_or(masks[:, :, mask], union_mask)
    for mask in range(masks.shape[-1]):
        mask_pos = np.where(masks[:, :, mask]==True)
        if np.any(mask_pos):
            y1, x1 = np.min(mask_pos, axis=1)
            y2, x2 = np.max(mask_pos, axis=1)
            rois[mask, :] = [y1, x1, y2, x2]
    return masks, rois

with open('/content/label_descriptions.json') as f:
    label_descriptions = json.load(f)

label_names = [fashion['name'] for fashion in label_descriptions['categories']]


def roi_box_coordinates(path):

  image = cv2.imread(path)
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

  result = model.detect([resize_image(path)])
  object_lo = result[0]

  if object_lo['masks'].size > 0:
```

```
      masks = np.zeros((image.shape[0], image.shape[1], object_lo['masks'].shape[-1]), d
type=np.uint8)
      for ol in range(object_lo['masks'].shape[-1]):
          masks[:, :, ol] = cv2.resize(object_lo['masks'][:, :, ol].astype('uint8'),
                                       (image.shape[1], image.shape[0]), interpolation=cv
2.INTER_NEAREST)

      y_scale = image.shape[0]/512
      x_scale = image.shape[1]/512
      rois = (object_lo['rois'] * [y_scale, x_scale, y_scale, x_scale]).astype(int)

      masks, rois = roi_masks(masks, rois)
      return image, masks, rois, object_lo
  else:
      masks, rois = object_lo['masks'], object_lo['rois']
      return image,masks, rois, object_lo




# retrieve the boxes of fashion clothes

def get_parts(image, boxes, masks, class_ids, class_names,
                      scores=None, title="",
                      figsize=(16, 16), ax=None,
                      show_mask=True, show_bbox=True,
                      colors=None, captions=None):
    """
    boxes: [num_instance, (y1, x1, y2, x2, class_id)] in image coordinates.
    masks: [height, width, num_instances]
    class_ids: [num_instances]
    class_names: list of class names of the dataset
    scores: (optional) confidence scores for each box
    title: (optional) Figure title
    show_mask, show_bbox: To show masks and bounding boxes or not
    figsize: (optional) the size of the image
    colors: (optional) An array or colors to use with each object
    captions: (optional) A list of strings to use as captions for each object
    """
    # Number of instances
    Objectes = boxes.shape[0]
    if not Objectes:
        print("\n*** No instances to display *** \n")
    else:
        assert boxes.shape[0] == masks.shape[-1] == class_ids.shape[0]

    # Show area outside image boundaries.
    height, width = image.shape[:2]
    masked_image = image.astype(np.uint32).copy()

    feet_ware_ = []
    upper_ware_ = []
    lower_ware_ = []
    whole_body_ = []

    for i in range(N):
        # Bounding box
        if not np.any(boxes[i]):
            continue
        y1, x1, y2, x2 = boxes[i]
        if not captions:
            class_id = class_ids[i]
            score = scores[i] if scores is not None else None
            label = class_names[class_id]
            caption = "{} {:.3f}".format(label, score) if score else label
            if score>0.70:
              if label in feet:
                feet_ware_.append((y1, x1, y2, x2))
              if label in upperbody:
                upper_ware_.append((y1, x1, y2, x2))
              if label in lowerbody:
                lower_ware_.append((y1, x1, y2, x2))
```

```
                if label in wholebody:
                  whole_body_.append((y1, x1, y2, x2))
        else:
            caption = captions[i]

    return  feet_ware_ ,upper_ware_,lower_ware_ , whole_body_



#get only objects from the boxes
def crop(y1,x1,y2,x2,img):
  outimage = img[int(y1):int(y2), int(x1):int(x2)]
  return outimage

def get_min_max(image_list,index,size):
  mini = []
  for crop_image in image_list:
    mini.append(crop_image[index])
  if size=='min' or :
    return min(mini)
  if size=='max':
    return max(mini)

def get_crop_image(image,image_list,name):
  images = []
  if name=='feet':
    for feet in image_list:
      outimage = crop(feet[0],feet[1],feet[2],feet[3],image)
      images.append(outimage)
  if name=='whole':
    for whole in image_list:
      outimage = crop(whole[0],whole[1],whole[2],whole[3],image)
      images.append(outimage)
  if name=='upper':
    y1 = get_min_max(image_list,0,'min')
    x1 = get_min_max(image_list,1,'min')
    y2 = get_min_max(image_list,2,'max')
    x2 = get_min_max(image_list,3,'max')
    outimage = crop(y1,x1,y2,x2,image)
    images.append(outimage)

  if name=='lower':
    y1 = get_min_max(image_list,0,'min')
    x1 = get_min_max(image_list,1,'min')
    y2 = get_min_max(image_list,2,'max')
    x2 = get_min_max(image_list,3,'max')
    outimage = crop(y1,x1,y2,x2,image)
    images.append(outimage)
  return images


def object_detctions(path):
  img , masks, rois, r = roi_box_coordinates(path)
  feet_ware_ ,upper_ware_,lower_ware_ , whole_body_ = get_parts(img, rois, masks, r['cla
ss_ids'],
                              ['bg']+label_names, r['scores'],
                              title=path, figsize=(12, 12),show_mask=False)
  feet_images = []
  whole_images = []
  upperbody_images = []
  lowerbody_images = []
  if len(feet_ware_)>0:
    feet_images = get_crop_image(img,feet_ware_,'feet')
  if len(whole_body_)>0:
    whole_images = get_crop_image(img,whole_body_,'whole')
  if len(upper_ware_)>0:
    upperbody_images = get_crop_image(img,upper_ware_,'upper')
  if len(lower_ware_)>0:
    lowerbody_images = get_crop_image(img,lower_ware_,'lower')

  return img , masks, rois, r , feet_images, whole_images, upperbody_images, lowerbody_i
mages
```

This crop images will be passed to module 3 to convert them to embeddings

Module 3 (Generate embeddings of images)

In this module we will use some pre-trained and get embeddings from their last layer

In [ ]:

```
import tensorflow as tf
import datetime as dt
import cv2
import numpy as np
```

In [ ]:

```
#tf.keras.backend.clear_session()
#Load pre-trained DenseNet121 model
model_embedding = tf.keras.applications.DenseNet121(
    include_top=False, weights='imagenet', input_tensor=None, input_shape=(520,520,3),
    pooling=None,
)
def load_img(img):
  # Reading an image
  image = cv2.imread(path,cv2.IMREAD_UNCHANGED)
  # resizing because pre-trained model image shape is 520x520
  image = cv2.resize(image,(520,520),interpolation=cv2.INTER_AREA)
  # Converting to RBG because it will be saved as a correct image even if it is saved aft
er being converted to a PIL
  image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
  # Preprocessed numpy.array or a tf.Tensor with type float32.
  image = tf.image.convert_image_dtype(image,tf.float32)[tf.newaxis, ...]
  # return image to genearte embedding
  return image

def get_embeddings(path):
  image = load_img(path)
  # Retuns 1024 dimension array/ vector with predicted values
  img_embedding = model_embedding.predict(image,steps=1)
  # Removes dimensions of size 1 from the shape of a tensor.
  img_embedding = tf.squeeze(img_embedding, axis=None, name=None)
  # Computes the mean of elements across dimensions of a tensor. [ Normalize ]
  img_embedding = tf.reduce_mean(img_embedding, axis=(0,1), keepdims=False, name=None)
  # Converting to List
  img_embedding = img_embedding.eval(session=tf.Session())
  img_embedding = img_embedding.tolist()
  # Return image embedding
  return img_embedding
```

Module 4 (Store and Search embeddings)

In this module we will store all our embeddings in Elastic Search so we can easily retrive them using similarity logic

In [ ]:

```
!pip install wget
!pip install faiss-cpu --no-cache
```

In [ ]:

```
!wget --header="Host: doc-08-5k-docs.googleusercontent.com" --header="User-Agent: Mozill
a/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.44
30.85 Safari/537.36" --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="Cookie: AUTH_g
```

```
nb78hdmdiks9t0b8kec09hpa7nncs5e=03515051603858730688|1619416350000|bc85h42ilg10cpnf37d50s
bmju30rr3s" --header="Connection: keep-alive" "https://doc-08-5k-docs.googleusercontent.c
om/docs/securesc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/7c02pqulq5l7nihepnpcilsgtvo0vlej/161941
6500000/03515051603858730688/03515051603858730688/1Yj4AwMISDq3YbKU8S3SERa73GT9XBP1f?e=dow
nload&authuser=0" -c -O 'foot_ware.csv'
!wget --header="Host: doc-04-5k-docs.googleusercontent.com" --header="User-Agent: Mozill
a/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.44
30.85 Safari/537.36" --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="Cookie: AUTH_g
nb78hdmdiks9t0b8kec09hpa7nncs5e=03515051603858730688|1619416350000|bc85h42ilg10cpnf37d50s
bmju30rr3s" --header="Connection: keep-alive" "https://doc-04-5k-docs.googleusercontent.c
om/docs/securesc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/46fg1uq7f1b3oqhoca3gt6in13l9q1oj/161941
6500000/03515051603858730688/03515051603858730688/11HRqh7hTkDn_67NkDMYYpbnJqgEmXJep?e=dow
nload&authuser=0" -c -O 'lower_ware.csv'
!wget --header="Host: doc-0c-5k-docs.googleusercontent.com" --header="User-Agent: Mozill
a/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.44
30.85 Safari/537.36" --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="Cookie: AUTH_g
nb78hdmdiks9t0b8kec09hpa7nncs5e=03515051603858730688|1619416350000|bc85h42ilg10cpnf37d50s
bmju30rr3s" --header="Connection: keep-alive" "https://doc-0c-5k-docs.googleusercontent.c
om/docs/securesc/lcn000d4f5ncb3531bgn3uus2eb0i5pv/npa3skc7snkm4c69j484q10i816i05ld/161941
6500000/03515051603858730688/03515051603858730688/1FkqQ5M3eP9pTApiMSLZn_ygC_5ITgi-S?e=dow
nload&authuser=0" -c -O 'upper_ware.csv'
```

```
--2021-04-26 05:56:26--  https://doc-08-5k-docs.googleusercontent.com/docs/securesc/lcn00
0d4f5ncb3531bgn3uus2eb0i5pv/7c02pqulq5l7nihepnpcilsgtvo0vlej/1619416500000/03515051603858
730688/03515051603858730688/1Yj4AwMISDq3YbKU8S3SERa73GT9XBP1f?e=download&authuser=0
Resolving doc-08-5k-docs.googleusercontent.com (doc-08-5k-docs.googleusercontent.com)...
74.125.142.132, 2607:f8b0:400e:c08::84
Connecting to doc-08-5k-docs.googleusercontent.com (doc-08-5k-docs.googleusercontent.com)
|74.125.142.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/csv]
Saving to: 'foot_ware.csv'

foot_ware.csv           [        <=>         ] 784.96M   143MB/s    in 5.6s

2021-04-26 05:56:31 (140 MB/s) - 'foot_ware.csv' saved [823085689]

--2021-04-26 05:56:32--  https://doc-04-5k-docs.googleusercontent.com/docs/securesc/lcn00
0d4f5ncb3531bgn3uus2eb0i5pv/46fg1uq7f1b3oqhoca3gt6in13l9q1oj/1619416500000/03515051603858
730688/03515051603858730688/11HRqh7hTkDn_67NkDMYYpbnJqgEmXJep?e=download&authuser=0
Resolving doc-04-5k-docs.googleusercontent.com (doc-04-5k-docs.googleusercontent.com)...
74.125.142.132, 2607:f8b0:400e:c08::84
Connecting to doc-04-5k-docs.googleusercontent.com (doc-04-5k-docs.googleusercontent.com)
|74.125.142.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/csv]
Saving to: 'lower_ware.csv'

lower_ware.csv          [              <=>   ] 460.79M   120MB/s    in 3.8s

2021-04-26 05:56:36 (120 MB/s) - 'lower_ware.csv' saved [483178840]

--2021-04-26 05:56:36--  https://doc-0c-5k-docs.googleusercontent.com/docs/securesc/lcn00
0d4f5ncb3531bgn3uus2eb0i5pv/npa3skc7snkm4c69j484q10i816i05ld/1619416500000/03515051603858
730688/03515051603858730688/1FkqQ5M3eP9pTApiMSLZn_ygC_5ITgi-S?e=download&authuser=0
Resolving doc-0c-5k-docs.googleusercontent.com (doc-0c-5k-docs.googleusercontent.com)...
74.125.142.132, 2607:f8b0:400e:c08::84
Connecting to doc-0c-5k-docs.googleusercontent.com (doc-0c-5k-docs.googleusercontent.com)
|74.125.142.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/csv]
Saving to: 'upper_ware.csv'

upper_ware.csv          [            <=>     ] 353.25M   141MB/s    in 2.5s

2021-04-26 05:56:39 (141 MB/s) - 'upper_ware.csv' saved [370405321]
```

In [ ]:

```
upper_ware = pd.read_csv("/content/upper_ware.csv")
lower_ware = pd.read_csv("/content/lower_ware.csv")
foot_ware = pd.read_csv("/content/foot_ware.csv")
```

In [ ]:

```
upper_img_embedding = upper_ware.img_embedding.tolist()
lower_img_embedding = lower_ware.img_embedding.tolist()
foot_img_embedding = foot_ware.img_embedding.tolist()
```

In [ ]:

```
import faiss
import wget
# # As the embedding stored in from os strings converting them to list by using EVAL meth
od
upper_list_ = []
lower_list_ = []
foot_list_  = []
for i in range(len(upper_img_embedding)):
  upper_list_.append(eval(upper_img_embedding[i]))

for i in range(len(lower_img_embedding)):
  lower_list_.append(eval(lower_img_embedding[i]))

for i in range(len(foot_img_embedding)):
  foot_list_.append(eval(foot_img_embedding[i]))

## Converting the list to Arrya and chaning type to float32 as the FAISS takes only array
with type float32
upper_ =np.array([np.array(xi) for xi in upper_list_])
upper_ = upper_.astype('float32')

lower_ =np.array([np.array(xi) for xi in lower_list_])
lower_ = lower_.astype('float32')

foot_ =np.array([np.array(xi) for xi in foot_list_])
foot_ = foot_.astype('float32')

## Creating three index for upper ware, lower ware and foot ware.

## build the index with len of list[ as embedding length is 1024 , have to pass argument
as 1024] L2 distance
upper_index = faiss.IndexFlatL2(1024)   # build the index
lower_index = faiss.IndexFlatL2(1024)
foot_index = faiss.IndexFlatL2(1024)
print(upper_index.is_trained)
print(lower_index.is_trained)
print(foot_index.is_trained)
upper_index.add(upper_)
lower_index.add(lower_)
foot_index.add(foot_)                  # add vectors to the index
```

```
True
True
True
```

In [ ]:

```
def get_image_links(df,url,img_emb):
  # Filtering duplicates and retreiving only unqiue one image URL's
  image_links = []
  df_new = img_emb[img_emb['url'] == url]
  df_new.reset_index(inplace = True)
  if df_new.shape[0] > 4:
    for i in range(int(df_new.shape[0]/2)):
      image_links.append(df_new["img_url"][i])
  else:
    for i in range(int(df_new.shape[0]/2)):
```

```
      image_links.append(df_new["img_url"][i])
  return image_links



def url_to_image(link):
  # Downlaoding recomamnded images to display
  filename = wget.download(link)
  img = cv2.imread(filename,cv2.IMREAD_UNCHANGED)
  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
  return img


def plot_images(li):
  # Plotting images
  try:
    _, axs = plt.subplots(1, len(li), figsize=(12, 12))
    axs = axs.flatten()
    for img, ax in zip(li, axs):
      ax.imshow(img)
    plt.show()
  except:
    pass



def print_recommendations(I,img_emb):
  c = 1
  url_exist = []

  returned_embedding = []
  for i in range(I.shape[1]):
    returned_embedding.append(I[0][i])

  new_img_emb = img_emb[img_emb.index.isin(returned_embedding, level=None)]
  info = new_img_emb["url"].tolist()

  for buy_url in info:
    if buy_url not in url_exist:
      url_exist.append(buy_url)
      images_links = get_image_links(new_img_emb,url,img_emb)

      images = []
      for link in images_links:
        img = url_to_image(link)
        images.append(img)

      print(f"recomendation {c}")
      print("You can buy product from below link")
      # Print Buy Url to buy the products
      print(buy_url)
      # Plotting similar images under a buy URL
      plot_images(images)
      print("\n\n")
      c+=1
      if c==4:
        break
```

In [ ]:

```
# Final funtion

def Fashion_Recommandation(path):
  print("Searching for fashion objects..........")
  img , masks, rois, r , feet_images,whole_images,upperbody_images,lowerbody_images = ob
ject_detctions(path)#Module-2
  visualize.display_instances(img, rois, masks, r['class_ids'],
                              ['bg']+label_names, r['scores'],
                              title=path, figsize=(8,8),show_mask=False)
  if len(upperbody_images)>0:
    print("Upper Ware")
```

```
    for image in upperbody_images:
      _, ax = plt.subplots(1, figsize=(4,4))
      ax.imshow(image.astype(np.uint8))
      plt.show()
      query_vec = get_embeddings(image)
      print("Searching for upper ware")
      #Embedding is list type. Now we have convert it into array and change dtype to floa
t32
      emb = np.asarray(query_vec)
      emb = emb.astype('float32')
      # Convert it to row vector by adding a new axis
      upper_vec = emb[np.newaxis, :]
      D, I = upper_index.search(upper_vec,10)
      print_recommendations(I, upper_ware)

  if len(lowerbody_images)>0:
    print("lower Ware")
    for image in lowerbody_images:
      _, ax = plt.subplots(1, figsize=(4,4))
      ax.imshow(image.astype(np.uint8))
      plt.show()
      query_vec = get_embeddings(image)
      print("Searching for Lower ware")
      # Embedding is list type. Now we have convert it into array and change dtype to flo
at32
      emb = np.asarray(query_vec)
      emb = emb.astype('float32')
      # Convert it to row vector by adding a new axis
      lower_vec = emb[np.newaxis, :]
      D, I = lower_index.search(lower_vec,10)
      print_recommendations(I,lower_ware)


  if len(feet_images)>0:
    print("Foot Ware")
    for image in feet_images:
      _, ax = plt.subplots(1, figsize=(4,4))
      ax.imshow(image.astype(np.uint8))
      plt.show()
    query_vec = get_embeddings(feet_images[0])
    # Embedding is list type. Now we have convert it into array and change dtype to float
32
    print("Searching for footware")
    emb = np.asarray(query_vec)
    emb = emb.astype('float32')
    # Convert it to row vector by adding a new axis
    lower_vec = emb[np.newaxis, :]
    D, I = lower_index.search(lower_vec,10)
    print_recommendations(I,foot_ware)
```

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/90.0.4430.85 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/compressed/women_shi
rts_tops_tees.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40ka
ggle-161607.iam.gserviceaccount.com%2F20210426%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Da
te=20210426T082623Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-Signature=7ef4
c55c1b546cd645834cb579a53b8b88da5e5c579b5790317b2a99e1208daebb5beb73919b7fb88c94529b5a872
62c24c97760b952a0bf7c344b9705fa3f44cea1fbc8c48ce9f339f5dfccd4d12976c24d1894328d7be6a5ac62
28d2c564920c953d99ab9b558c8c5d8577e633af61bffaa00e963f30ccf40f4c1f98623894c3b14b6fca6a4e2
a8cb7305c63adae711506607f95245c62a8e3570adcaa53172bd5ddc9204700092ae4c3c3626dfa00ea6ea420
f5e23d4eeecfa6fb1b7379973678009642acd9714ad7205c1fa970ba804ad9cf1ceeb51cd89e062e7c7a32624
8b361ba0f2f26f1bfeea42c6fe64e204b3d744ee68b667247a2869d0f291efc" -c -O 'women_shirts_tops
_tees.zip'
```

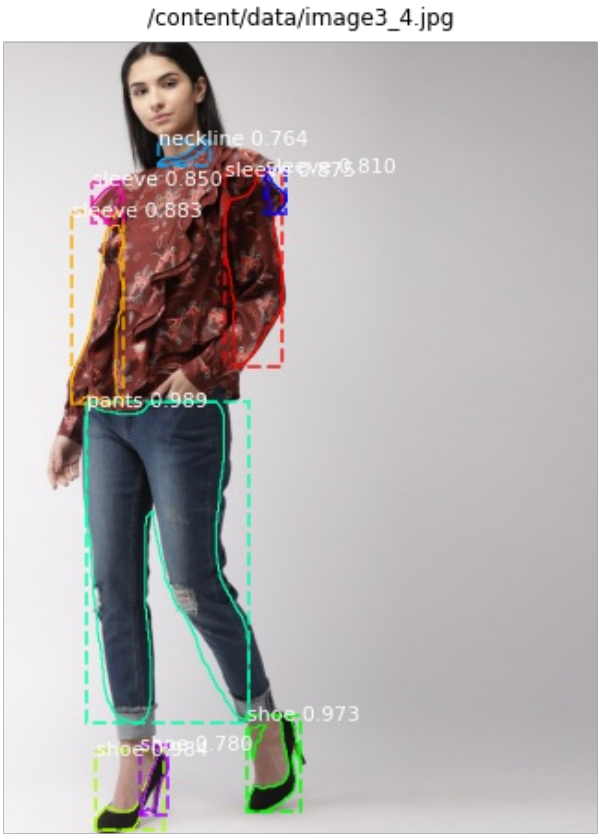
```
--2021-04-26 08:26:40--  https://storage.googleapis.com/kaggle-data-sets/930393/1613771/c
ompressed/women_shirts_tops_tees.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=
gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20210426%2Fauto%2Fstorage%2Fgoog
```

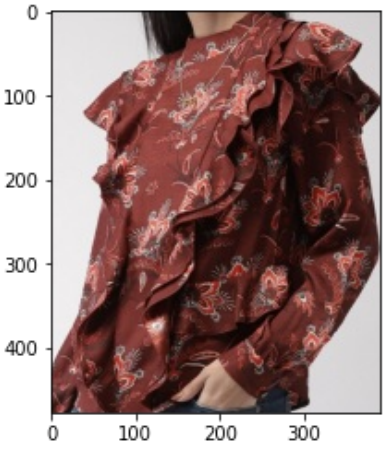
```
4_request&X-Goog-Date=20210426T082623Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-
Goog-Signature=7ef4c55c1b546cd645834cb579a53b8b88da5e5c579b5790317b2a99e1208daebb5beb7391
9b7fb88c94529b5a87262c24c97760b952a0bf7c344b9705fa3f44cea1fbc8c48ce9f339f5dfccd4d12976c24
d1894328d7be6a5ac6228d2c564920c953d99ab9b558c8c5d8577e633af61bffaa00e963f30ccf40f4c1f9862
3894c3b14b6fca6a4e2a8cb7305c63adae711506607f95245c62a8e3570adcaa53172bd5ddc9204700092ae4c
3c3626dfa00ea6ea420f5e23d4eeecfa6fb1b7379973678009642acd9714ad7205c1fa970ba804ad9cf1ceeb5
1cd89e062e7c7a326248b361ba0f2f26f1bfeea42c6fe64e204b3d744ee68b667247a2869d0f291efc
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.135.128, 74.125.142.1
28, 74.125.195.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.135.128|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 3594186014 (3.3G) [application/zip]
Saving to: 'women_shirts_tops_tees.zip'

women_shirts_tops_t 100%[===================>]   3.35G  37.9MB/s    in 61s

2021-04-26 08:27:41 (56.4 MB/s) - 'women_shirts_tops_tees.zip' saved [3594186014/35941860
14]
```

In [ ]:

```
path = '/content/data/image3_4.jpg'
Fashion_Recommandation(path)
```

```
Searching for fashion objects..........
```



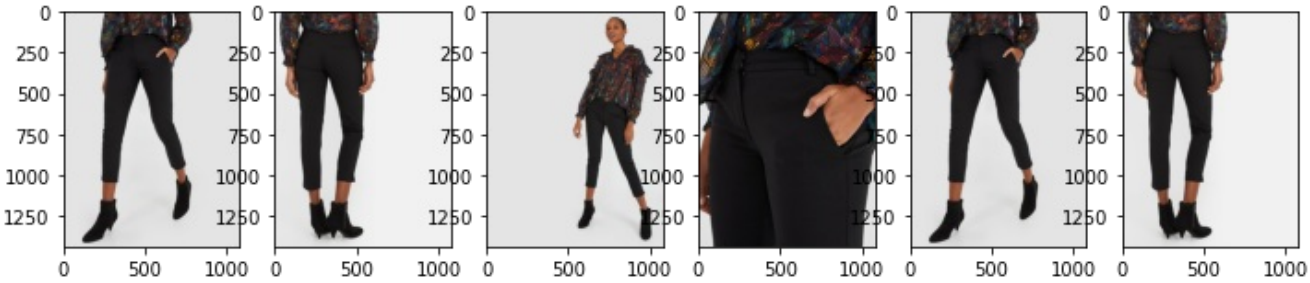


Searching for upper ware
recomendation 1
You can buy product from below link
https://www.myntra.com/tops/dorothy-perkins/dorothy-perkins-women-peach-coloured-petite-self-design-styled-back-top/12409818/buy

recomendation 2
You can buy product from below link
https://www.myntra.com/tops/scotch--soda/scotch--soda-women-black-floral-print-top/8532617/buy

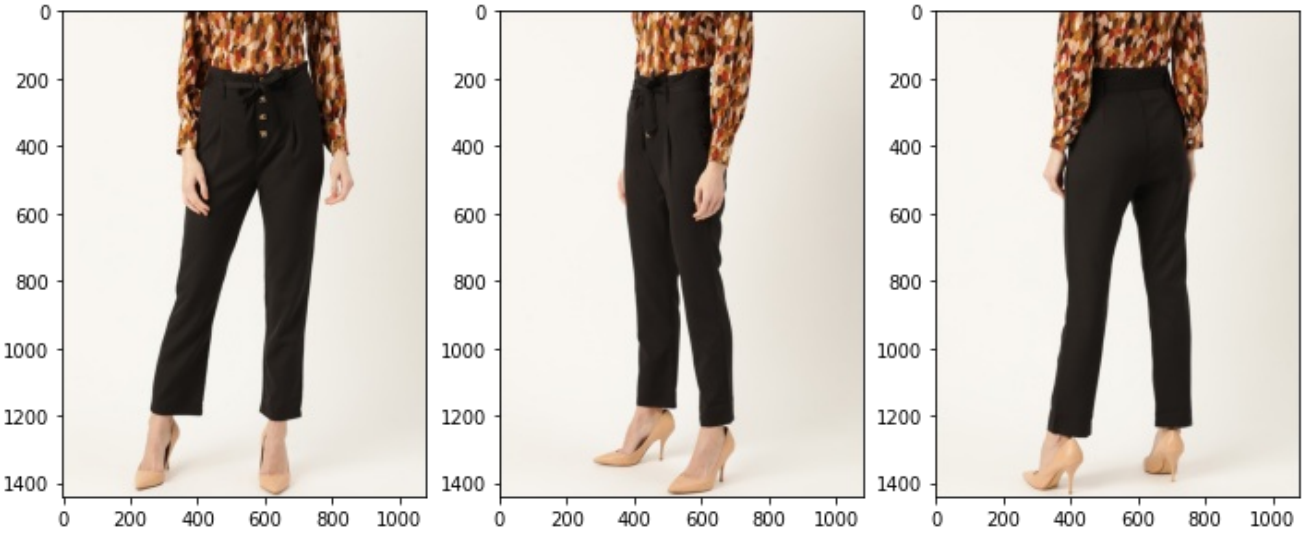
recomendation 3
You can buy product from below link
https://www.myntra.com/tops/scotch--soda/scotch--soda-women-burgundy-printed-top/8581649/buy

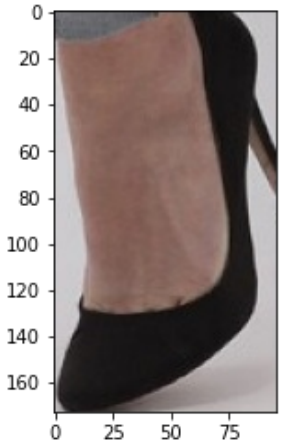
Searching for Lower ware
recomendation 1
You can buy product from below link
https://www.myntra.com/trousers/promod/promod-women-black-pencil-slim-fit-solid-regular-trousers/11388718/buy



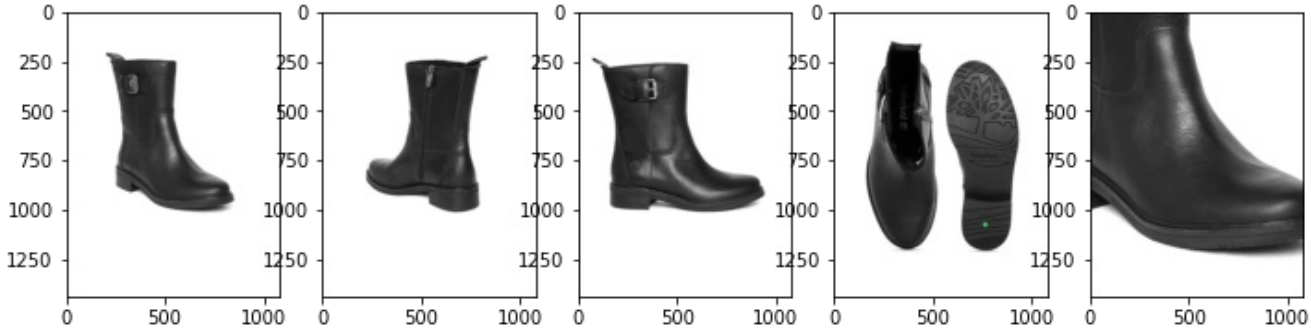
recomendation 2
You can buy product from below link
https://www.myntra.com/trousers/label-ritu-kumar/label-ritu-kumar-women-beige-slim-fit-solid-cropped-trousers/10672638/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/trousers/20dresses/20dresses-women-black-regular-fit-solid-regular-trousers/11680610/buy

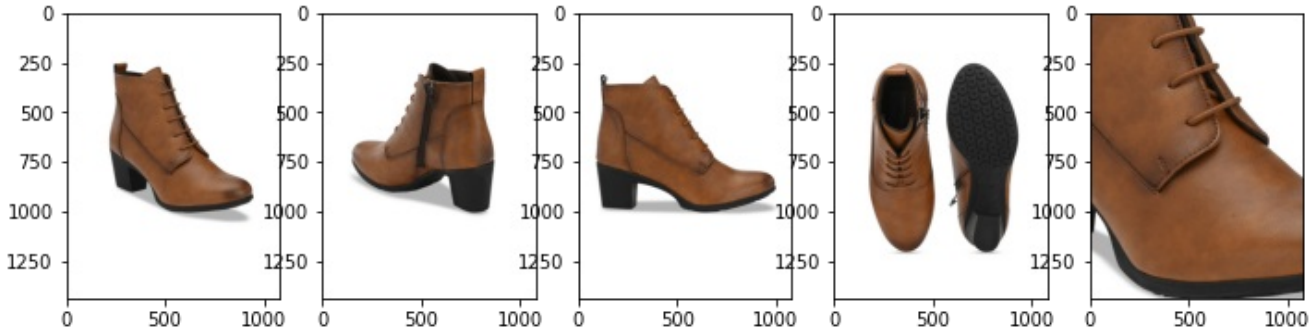




Searching for footware ware
recomendation 1
You can buy product from below link
https://www.myntra.com/casual-shoes/timberland/timberland-women-black-solid-leather-chevalier-biker-high-top-flat-boots/9161717/buy



recomendation 2
You can buy product from below link
https://www.myntra.com/heels/delize/delize-women-tan-brown-solid-heeled-mid-top-leather-boots/11381650/buy



recomendation 3
You can buy product from below link

https://www.myntra.com/heels/her-by-invictus/her-by-invictus-women-tan-brown-solid-heeled-cushioned-boots/11178524/buy

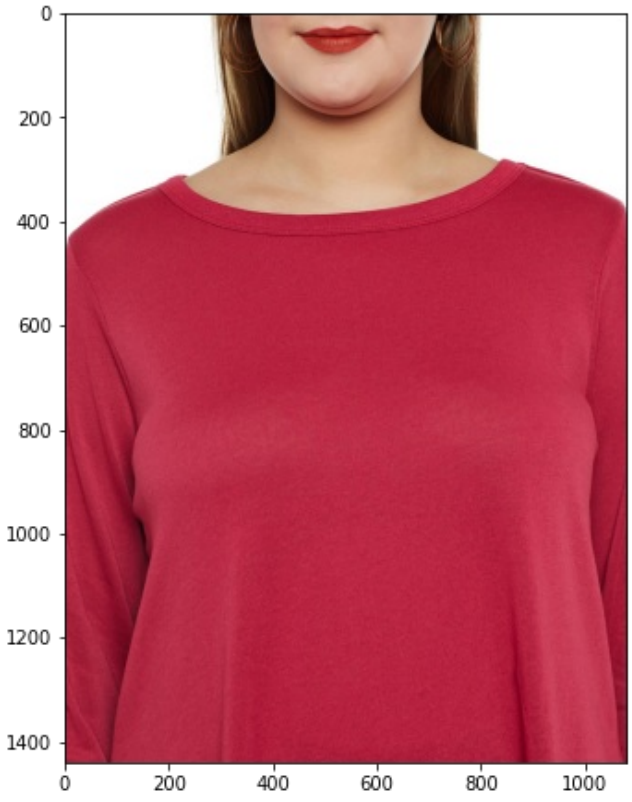
In [ ]:

```
path = '/content/data/image11_0.jpg'
Fashion_Recommandation(path)
```

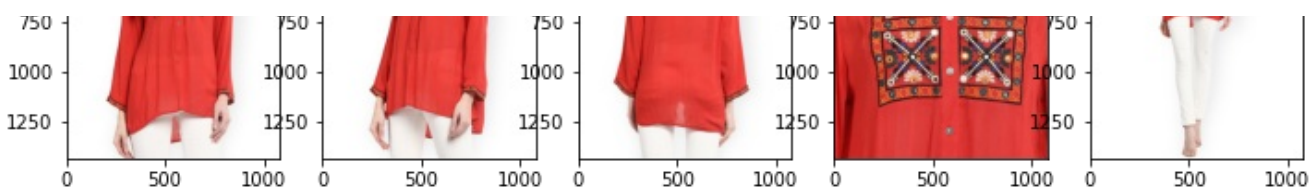
```
Searching for fashion objects..........
```

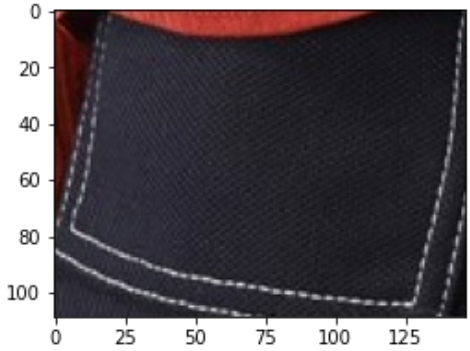
```
Searching for upper ware
recomendation 1
You can buy product from below link
https://www.myntra.com/tops/next/next-women-yellow-solid-cap-sleeves-top/8181709/buy
```


recomendation 2
You can buy product from below link
https://www.myntra.com/tops/junarose/junarose-plus-size-women-magenta-solid-top/2531823/buy

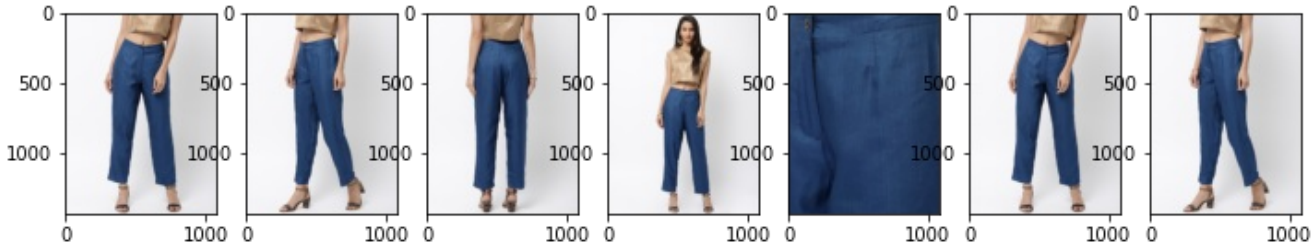



recomendation 3
You can buy product from below link
https://www.myntra.com/tops/bhama-couture/bhama-couture-women-red-embroidered-top/1984526/buy



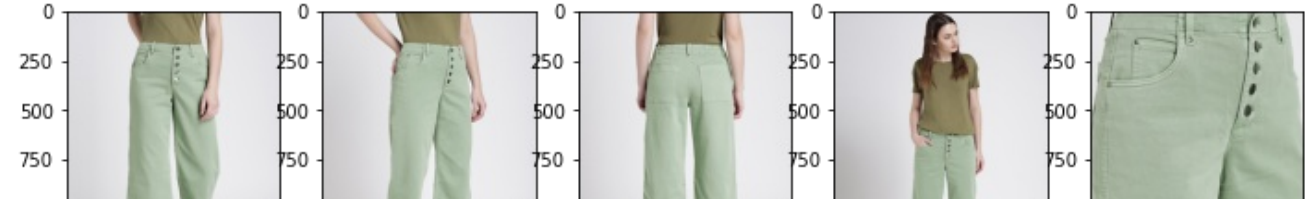


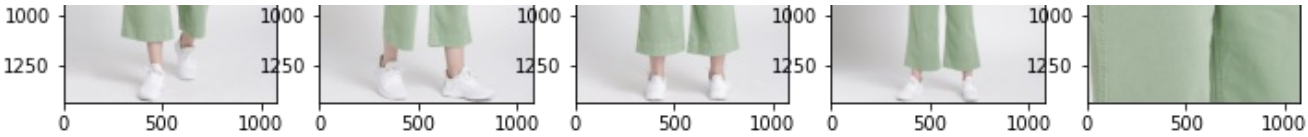
Searching for Lower ware
recomendation 1
You can buy product from below link
https://www.myntra.com/trousers/fabindia/fabindia-women-blue-regular-fit-solid-ethnic-trousers/7329920/buy



recomendation 2
You can buy product from below link
https://www.myntra.com/trousers/cotton-on/cotton-on-women-pink-loose-fit-solid-linen-culottes/9851469/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/trousers/gap/gap-womens-green-high-rise-crop-wide-leg-pants-with-button-fly/7088503/buy





In [ ]:

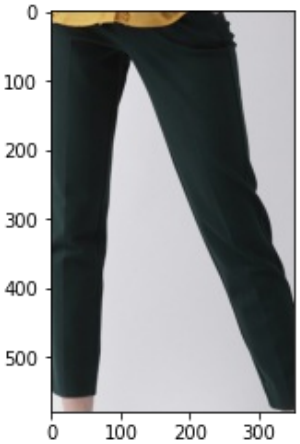
```
path = '/content/data/image18_4.jpg'
Fashion_Recommandation(path)
```

```
Searching for fashion objects.....
```

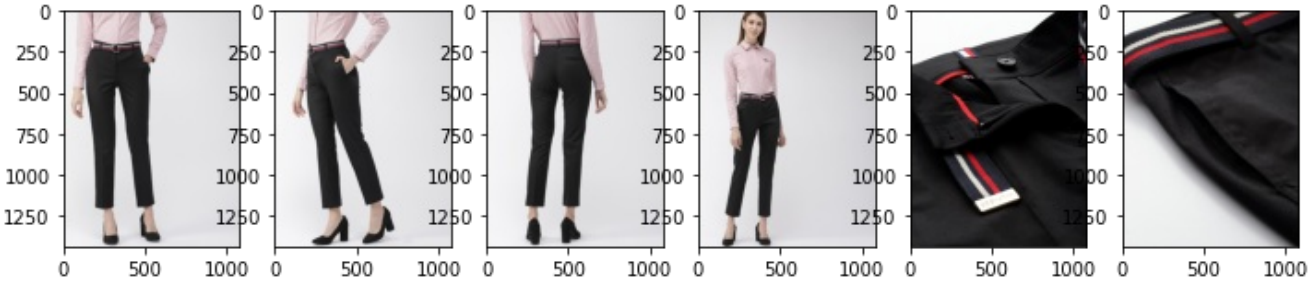
```
Searching for upper ware
recomendation 1
You can buy product from below link
https://www.myntra.com/shirts/mango/mango-women-burgundy-regular-fit-solid-high-low-casua
l-shirt/11926024/buy
```

recomendation 2
You can buy product from below link
https://www.myntra.com/shirts/tommy-hilfiger/tommy-hilfiger-women-blue-regular-fit-solid-casual-chambray-shirt/2201422/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/shirts/scotch--soda/scotch--soda-women-mustard-yellow-regular-fit-solid-casual-shirt/8532621/buy

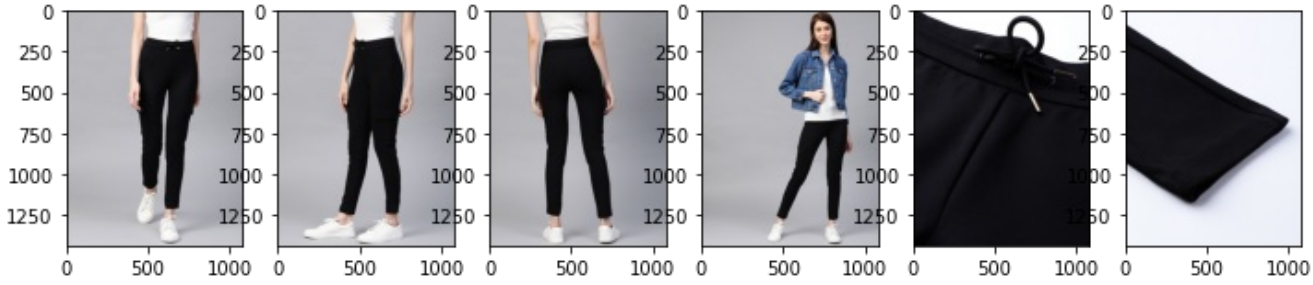


Searching for Lower ware
recomendation 1
You can buy product from below link
https://www.myntra.com/trousers/tommy-hilfiger/tommy-hilfiger-women-black-regular-fit-solid-regular-trousers/10785874/buy



recomendation 2
You can buy product from below link
https://www.myntra.com/trousers/jump-usa/jump-usa-women-black-pencil-slim-fit-solid-regular-trousers/4448637/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/trousers/marks--spencer/marks--spencer-women-black-regular-fit-solid-trousers/12382826/buy

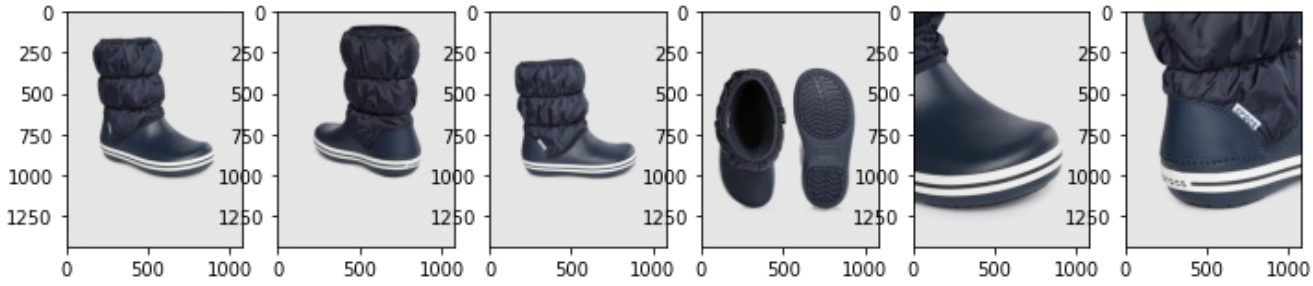


Searching for footware ware
recomendation 1
You can buy product from below link

https://www.myntra.com/heels/saint-g/saint-g-women-tan-brown-solid-leather-heeled-boots/9203925/buy

recomendation 2
You can buy product from below link
https://www.myntra.com/casual-shoes/timberland/timberland-women-black-leather-kamela-pt-gaiter-high-top-flat-boots/9161631/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/casual-shoes/crocs/crocs-women-navy-blue-flat-boots/10624076/buy



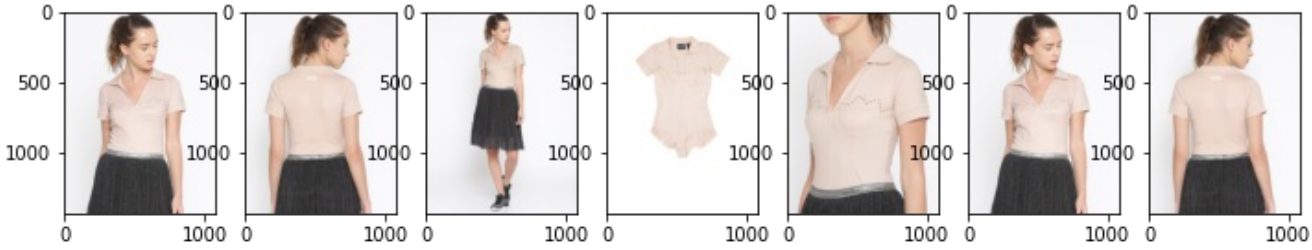
In [ ]:

```
path = '/content/9c022ea15de1f73e2c136bdae2c5d78d.jpg'
Fashion_Recommandation(path)
```

```
Searching for fashion objects.....
```

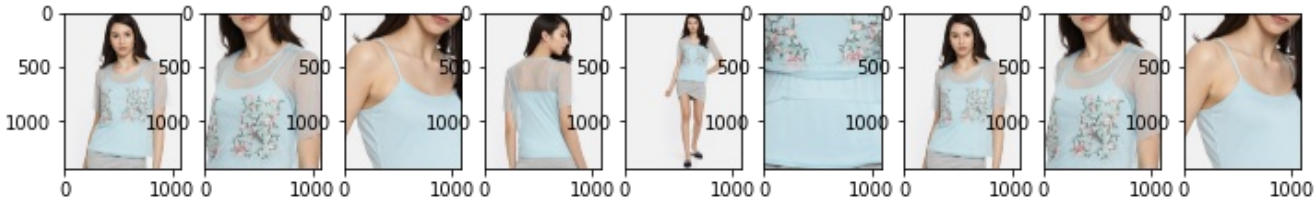
/content/9c022ea15de1f73e2c136bdae2c5d78d.jpg

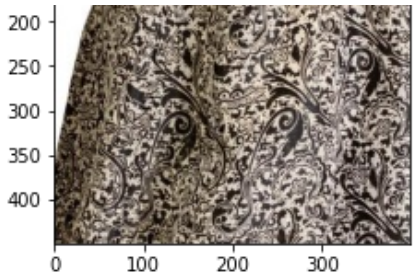
Searching for upper ware
recomendation 1
You can buy product from below link
https://www.myntra.com/tshirts/puma/puma-women-pink-solid-bodysuit-polo-t-shirt/2315046/buy



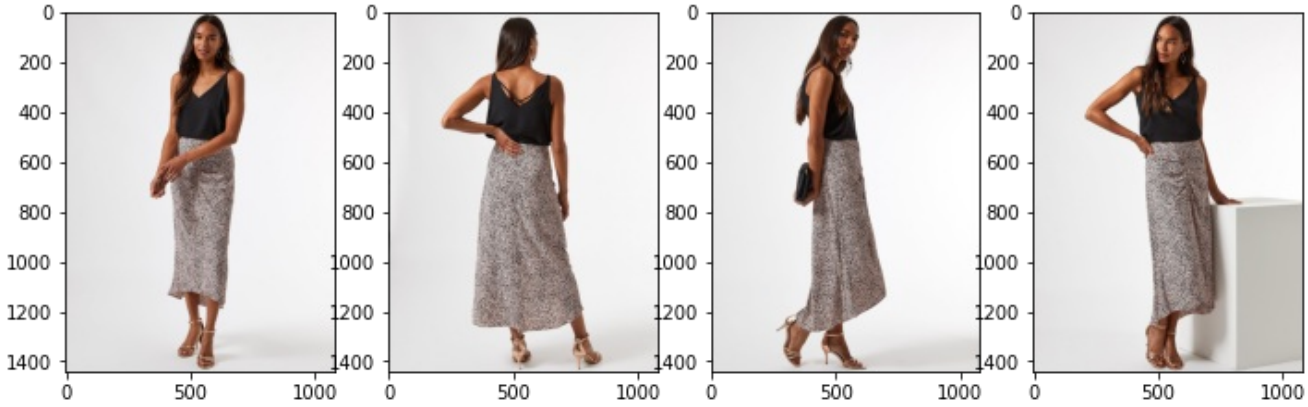
recomendation 2
You can buy product from below link
https://www.myntra.com/tops/junarose/junarose-plus-size-women-pink-solid-semi-sheer-top/9450705/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/tops/vero-moda/vero-moda-women-blue-self-design-sheer-top/6555041/buy



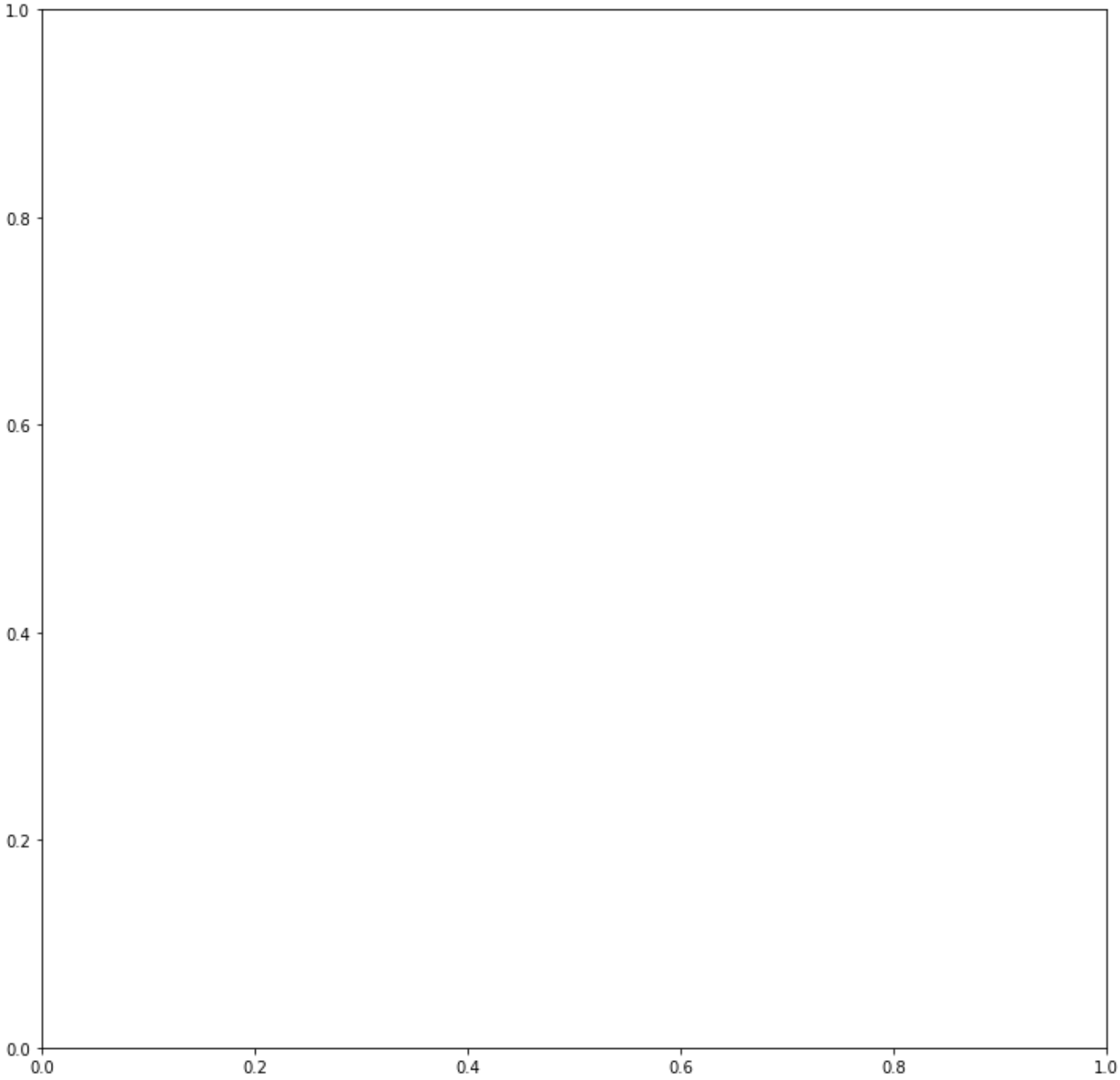


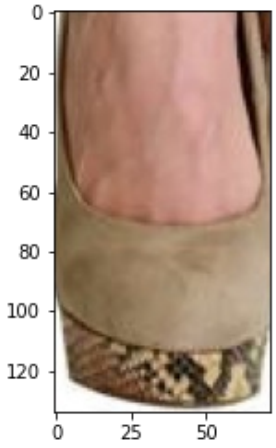
Searching for Lower ware
recomendation 1
You can buy product from below link
https://www.myntra.com/skirts/dorothy-perkins/dorothy-perkins-women-off-white--black-leopard-print-ruched-high-low-maxi-skirt/12409794/buy



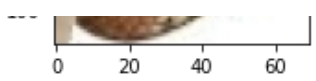
recomendation 2
You can buy product from below link
https://www.myntra.com/skirts/forever-new/forever-new-women-brown-floral-print-pleated-skirt/10829940/buy

recomendation 3
You can buy product from below link
https://www.myntra.com/skirts/dorothy-perkins/dorothy-perkins-women-black--pink-floral-print-accordian-pleated-a-line-skirt/12666314/buy

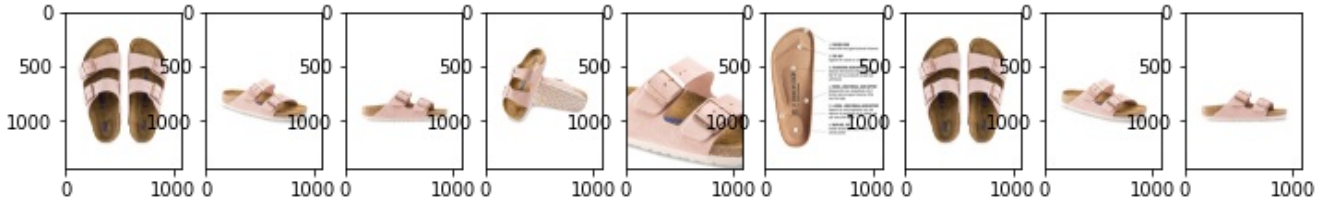

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


0
20
40
60
80
100
120
0
25
50

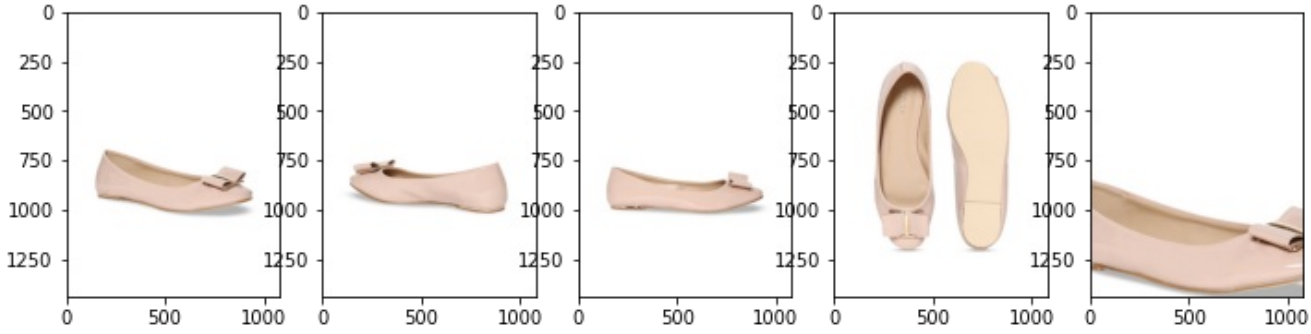

0
20
40
60
80
100



Searching for footware ware
recomendation 1
You can buy product from below link
https://www.myntra.com/flats/birkenstock/birkenstock-unisex-pink-arizona-suede-leather-open-toe-flats/11661950/buy



recomendation 2
You can buy product from below link
https://www.myntra.com/flats/stride/stride-women-pink-solid-ballerinas/10877410/buy



recomendation 3
You can buy product from below link
https://www.myntra.com/flats/charles--keith/charles--keith-women-white-solid-pu-open-toe-flats/12305860/buy