IMAGE BASED SEARCH ENGINE

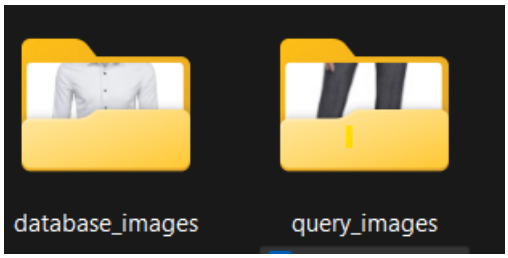
Using Computer Vision Technique

VEENA GOVIND

1 PROJECT DESIGN

1.1 Dataset

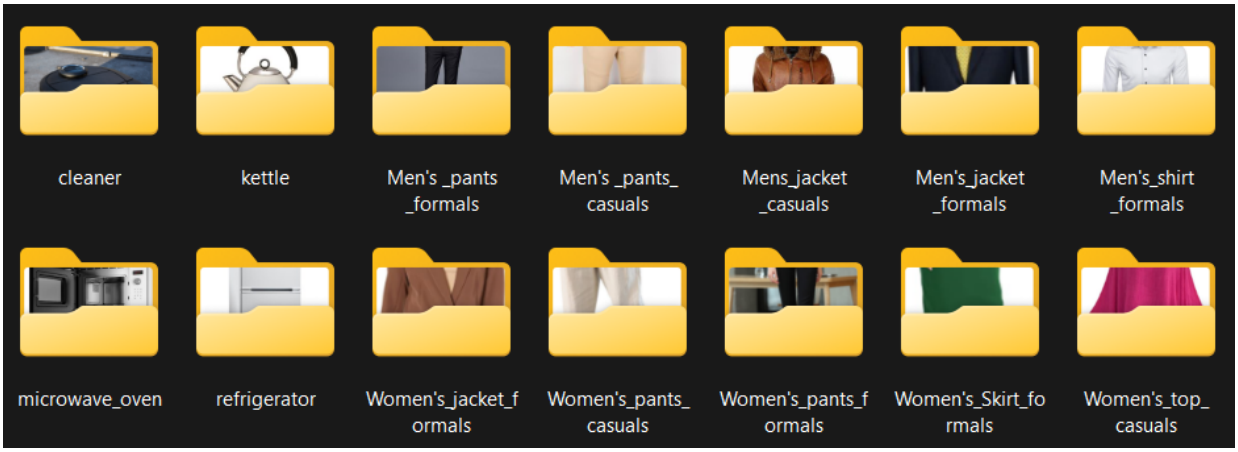
Since the proposed model is based on a transfer learning technique, there is no requirement of training data. The data gathered in different categories used for testing the model. For better understanding, the dataset splits into 2. One set for query images which are considered as user's input data and the other set for database images which are considered as the database in the particular online shopping platform.



Dataset

1.2 Data Management Methodology

The dataset contains different categories in main 2 segments, clothing and home appliances. In clothing, there are different styles in men's and women's categories. All are kept in different folders. The home appliances are also arranged in specific folders based on the categories which includes refrigerator, electric oven, robot cleaner and kettle. All images neatly labelled and formatted.



Data management methodology

2 DEVELOPMENT PROCESS

In this first code snippet, necessary libraries are imported and loaded the bottom layers of the ResNet-101 model.

As illustrated below, it provides the summary of the model, details of layers, output shape and parameters.

```
import tensorflow as tf
import cv2
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import os
```

```
base_model=tf.keras.applications.ResNet101(input_shape=(224,224,3),include_top=False)
```

```
base_model.summary()
```

```
Model: "resnet101"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
```

The features are extracted from the intermediate layers and stacked one feature map on top of another.

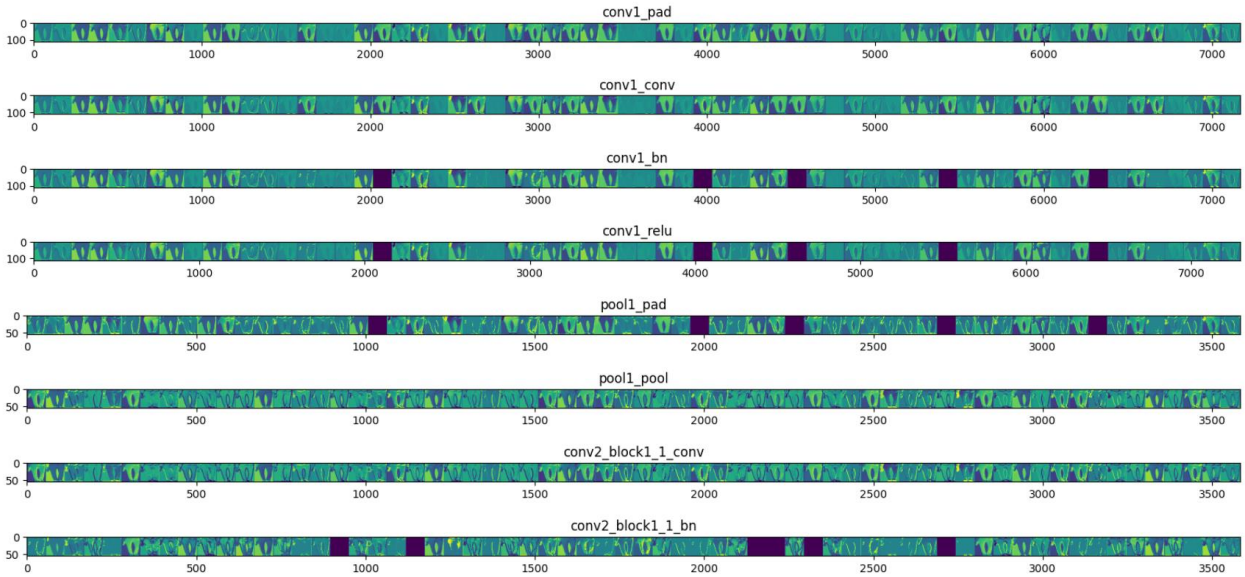
A women's pants has been taken as an example for showing feature extraction. It has been preprocessed and passed to the model and extracted the features, which are displayed along with the details of the layers.

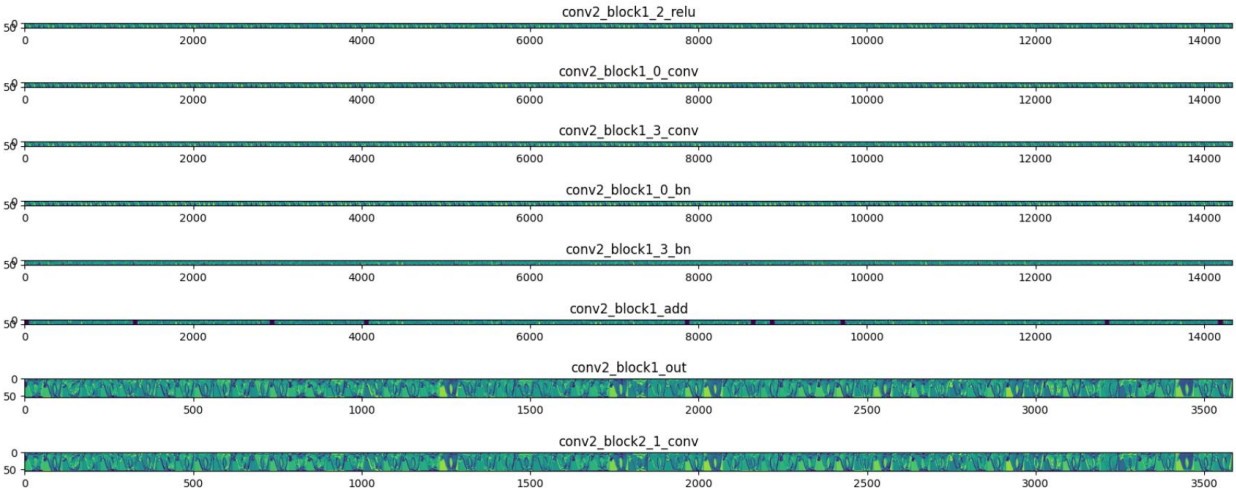
The code snippet used for visualization of the features is given below.

```
import numpy as np
%matplotlib inline
for layer_names,feature_maps in zip(layer_names,feature_maps):
    print(feature_maps.shape)
    if len(feature_maps.shape) == 4:
        channels = feature_maps.shape[-1]
        size = feature_maps.shape[1]
        display_grid = np.zeros((size,size *channels))
        for i in range(channels):
            x = feature_maps[0,:,:,i]
            x-= x.mean()
            x /=x.std()
            x *=64
            x+=128
            x=np.clip(x,0,255).astype('uint8')
            display_grid[:,i*size : (i+1) * size] = x
        scale = 20./channels
        plt.figure(figsize=(scale * channels, scale))
        plt.title(layer_names)
        plt.grid(False)
        plt.imshow(display_grid, aspect='auto',cmap='viridis')
```

This code results to the following output.

```
(1, 56, 56, 256)
(1, 56, 56, 256)
(1, 56, 56, 64)
(1, 56, 56, 64)
(1, 56, 56, 64)
(1, 56, 56, 64)
(1, 56, 56, 64)
(1, 56, 56, 64)
(1, 56, 56, 256)
(1, 56, 56, 256)
(1, 56, 56, 256)
(1, 56, 56, 256)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 512)
(1, 28, 28, 512)
(1, 28, 28, 512)
(1, 28, 28, 512)
(1, 28, 28, 512)
(1, 28, 28, 512)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
(1, 28, 28, 128)
```


conv1_pad
conv1_conv
conv1_bn
conv1_relu
pool1_pad
pool1_pool
conv2_block1_1_conv
conv2_block1_1_bn


conv2_block1_2_relu
conv2_block1_0_conv
conv2_block1_3_conv
conv2_block1_0_bn
conv2_block1_3_bn
conv2_block1_add
conv2_block1_out
conv2_block2_1_conv

3 Testing methods and Model deployment

3.1 Testing model

The query image has been loaded, preprocessed, predicted feature maps and converted to vectors.

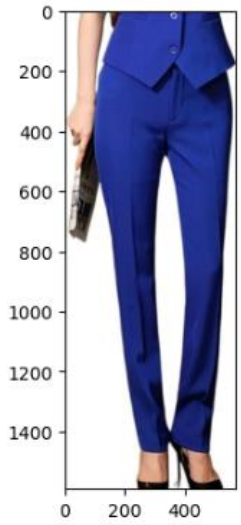
The database images have also been loaded and followed the same processing steps of query image.

Then the feature of query image has been compared with the features of database images and produced similarity scores using the correlation method. Then the scores are sorted in descending order and retrieved top matching images. This is illustrated further with the help of two examples related to different types of images.

The first example based on the image of an item of clothing is illustrated below.

```
query_img=cv2.imread('./query images/Trousers6.jpg')
query_img1=cv2.resize(query_img,[224,224])
query_img2=tf.keras.applications.resnet.preprocess_input(query_img1)
query_img2=np.expand_dims(query_img2,axis=0)

feat=base_model.predict(query_img2)
feat1=feat[0,:,:,:]
feat_query=feat1.mean(axis=0).mean(axis=0)
print('user query image')
plt.imshow(cv2.cvtColor(query_img,cv2.COLOR_BGR2RGB))
plt.show()

items=[]
file_name=os.listdir('./database_images/Ladies Pants formal/')
for i in file_name:
    image=cv2.imread('./database_images/Ladies Pants formal/'+i)
    image1=cv2.resize(image,[224,224])
    image1=tf.keras.applications.resnet.preprocess_input(image1)
    image1=np.expand_dims(image1,axis=0)
    feat=base_model.predict(image1)
    feat1=feat[0,:,:,:]
    feat_image=feat1.mean(axis=0).mean(axis=0)

    cor=pd.DataFrame(np.vstack((feat_query,feat_image)).T).corr().loc[0,1]
    items.append((i, cor, image))

items.sort(key=lambda x: x[1], reverse=True)
print("Top 5 similar items from data base images:")
print()

num_items = len(items)
num_cols = 5
num_rows = 1

fig, axs = plt.subplots(num_rows, num_cols, figsize=(15, 5))

for i, (item, cor, image) in enumerate(items[:5]):
    row_idx = i // num_cols
    col_idx = i % num_cols

    if num_rows > 1:
```

user query image



Top 5 similar items from database images:

Top similar items retrieved from database based on clothing

The second example based on the image of a home appliance is illustrated below.

```
query_img=cv2.imread('./query images/Robot vaccum cleaner 2.jpg')
query_img1=cv2.resize(query_img,[224,224])
query_img2=tf.keras.applications.resnet.preprocess_input(query_img1)
query_img2=np.expand_dims(query_img2,axis=0)

feat=base_model.predict(query_img2)
feat1=feat[0,:,:,:]
feat_query=feat1.mean(axis=0).mean(axis=0)
print('user query image')
plt.imshow(cv2.cvtColor(query_img,cv2.COLOR_BGR2RGB))
plt.show()

items=[]
file_name=os.listdir('./database_images/cleaner/')
for i in file_name:
    image=cv2.imread('./database_images/cleaner/'+i)
    image1=cv2.resize(image,[224,224])
    image1=tf.keras.applications.resnet.preprocess_input(image1)
    image1=np.expand_dims(image1,axis=0)
    feat=base_model.predict(image1)
    feat1=feat[0,:,:,:]
    feat_image=feat1.mean(axis=0).mean(axis=0)

    cor=pd.DataFrame(np.vstack((feat_query,feat_image)).T).corr().loc[0,1]
    items.append((i, cor, image))

items.sort(key=lambda x: x[1], reverse=True)
print("Top 5 similar items from data base images:")
print()

num_items = len(items)
num_cols = 5
num_rows = 1

fig, axs = plt.subplots(num_rows, num_cols, figsize=(15, 5))

for i, (item, cor, image) in enumerate(items[:5]):
    row_idx = i // num_cols
    col_idx = i % num_cols
```

User query image

Top 5 similar items from database images:

Top similar items retrieved from database based on home appliance

3.2 Model Deployment

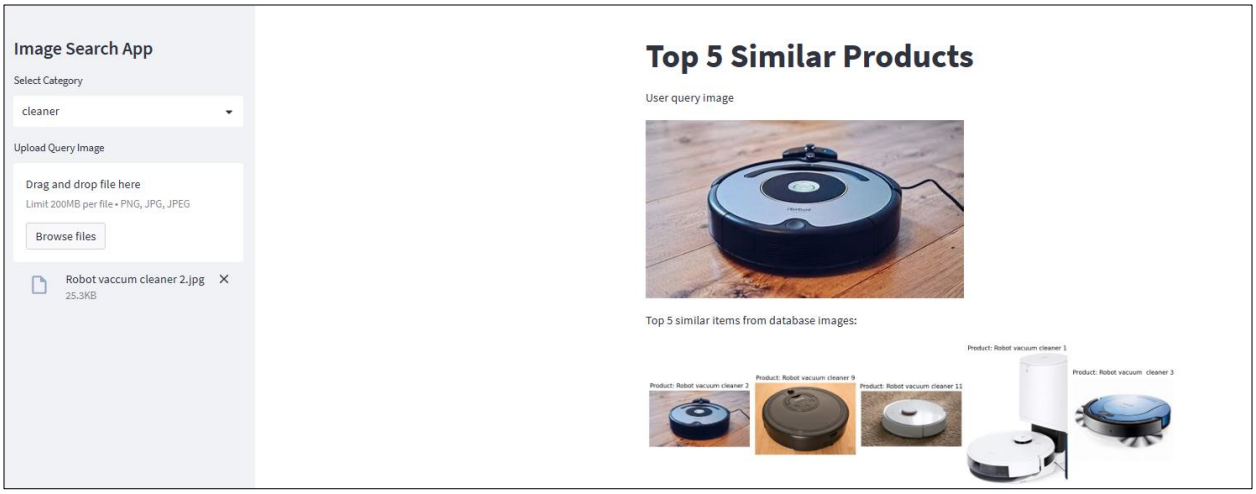
The model is deployed with the help of Streamlit package in python. On the web page, a file uploader is created to enter the query image. Also, a selection box is created for choosing the product category of the query image.

In [36]:
```
%%writefile image_search_engine2.py

import streamlit as st
import cv2
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import os
import pandas as pd

def main():
    st.sidebar.title("Image Search App")
    st.title("Top 5 Similar Products")
    category_folders = os.listdir("./database_images/")
    category_folders.sort()

    category = st.sidebar.selectbox("Select Category", category_folders)

    uploaded_file = st.sidebar.file_uploader("Upload Query Image", type=['png', 'jpg', 'jpeg'])

    if uploaded_file is not None:
        # Read and preprocess the uploaded image
        query_img = cv2.imdecode(np.frombuffer(uploaded_file.read(), np.uint8), 1)
        query_img = cv2.cvtColor(query_img, cv2.COLOR_BGR2RGB)
        query_img1 = cv2.resize(query_img, (224, 224))
        query_img2 = tf.keras.applications.resnet.preprocess_input(query_img1)
        query_img2 = np.expand_dims(query_img2, axis=0)

        base_model = tf.keras.applications.ResNet101(input_shape=(224, 224, 3), include_top=False)
        feat = base_model.predict(query_img2)
        feat1 = feat[0, :, :, :]
        feat_query = feat1.mean(axis=0).mean(axis=0)

        # Display the query image
        st.write('User query image')
        st.image(query_img)
```

Next, run the script file as per the command given below. The app will start and we can access it in our web browser.

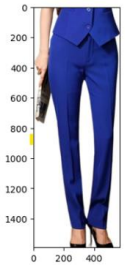
In [25]:
```
!streamlit run image_search_engine2.py
```

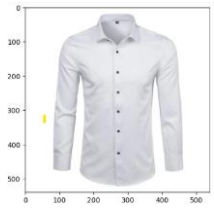
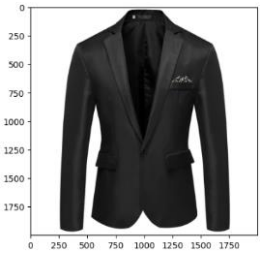
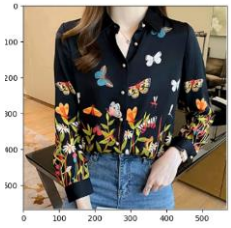
As shown in the below image, vacuum cleaner is given as query image and the similar items are retrieved.



3.3 Results

The results of the study have been illustrated below.

Input Image	Top 5 Similar Items
	
	

Input Image	Top 5 Similar Items
	
	
	
	

Input Image	Top 5 Similar Items