Image Classification Project

Submitted by:

ARGHA KR MAITI

ACKNOWLEDGMENT

I would like to express my thanks of gratitude to mentors Swathi Mahaseth as well as Flip Robo who gave me the golden opportunity to do this wonderful project on the topic Image Classification project, which also helped me in doing a lot of research and I came to know about so many new things. I am really very thankful to them.

I am making this project to increase my knowledge.

INTRODUCTION

Problem Statement:

Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details. We are trying to give you an exposure of how an end-to-end project is developed in this field.

The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portal. This is done to make the model more and more robust.

This task is divided into two phases: Data Collection and Mode Building.

Data Collection Phase

In this section, you need to scrape images from e-commerce portal, Amazon.com. The clothing categories used for scraping will be:

- Sarees (women)
- Trousers (men)
- Jeans (men)

Need to scrape images of these 3 categories and build your data from it. That data will be provided as an input to your deep learning problem. You need to scrape minimum 200 images of each categories. There is no maximum limit to the data collection. You are free to apply image augmentation techniques to increase the size of your data but make sure the quality of data is not compromised.

Model Building Phase

After the data collection and preparation is done, need to build an image classification model that will classify between these 3 categories mentioned above. We can play around with optimizers and learning rates for improving your model's performance.

Analytical Problem Framing

With the help of Selenium I have scrapped data from Amazon.com for three categories Sarees (women), Trousers (men), Jeans (men).

```python
1  # Importing Libraries
2  import selenium
3  import pandas as pd
4  import time
5
6  # Importing selenium webdriver
7  from selenium import webdriver
8
9  # Importing required Exceptions which needs to handled
10 from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
11
12 #Importing requests
13 import requests
```

```python
1  #webdriver
2  driver=webdriver.Chrome(r"C:\Users\SAGAR KADAM\Downloads\chromedriver_win32 (2)\chromedriver.exe")
3  time.sleep(3)
```

- Sarees (women)

```python
1  #opening the homepage of amazon.in
2  driver.get('https://www.amazon.in/s?rh=n%3A1968256031&fs=true&ref=lp_1968256031_sar')
```

```python
1  urls=[]
2  max_page=7 #maximum number of pages to extract
3  max_p_d=2 #page number digits
4  for i in range (2, max_page +1):
5      url=f"https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page={i}&qid=1632122277&ref=sr_pg_{i}"
6      urls.append(url)
7
8  urls
```

```
['https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page=2&qid=1632122277&ref=sr_pg_2',
 'https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page=3&qid=1632122277&ref=sr_pg_3',
 'https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page=4&qid=1632122277&ref=sr_pg_4',
 'https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page=5&qid=1632122277&ref=sr_pg_5',
 'https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page=6&qid=1632122277&ref=sr_pg_6',
 'https://www.amazon.in/s?i=apparel&rh=n%3A1968256031&fs=true&page=7&qid=1632122277&ref=sr_pg_7']
```

```python
1  links1=[]
2  driver.execute_script("window.scrollTo(0,document.body.scrollHeight)")
3  time.sleep(2)
4  try:
5      for l in driver.find_elements_by_xpath('//*[@class="a-section aok-relative s-image-tall-aspect"]/img'):
6          if l.get_attribute('src') and 'https' in l.get_attribute('src'):
7              links1.append(l.get_attribute('src'))
8  except NoSuchElementException:
9      links1.append("-")
10 for i in urls:
11     driver.get(i)
12     try:
13         for l in driver.find_elements_by_xpath('//*[@class="a-section aok-relative s-image-tall-aspect"]/img'):
14             if l.get_attribute('src') and 'https' in l.get_attribute('src'):
15                 links1.append(l.get_attribute('src'))
16     except NoSuchElementException:
17         links1.append("-")
```

```python
1  len(links1)
```

```
364
```

```python
import os
import io
import requests
from PIL import Image

for i, url in enumerate(links1):
    file_name = f"S{i}.jpg"

    if i <=280:
        os.chdir(r'D:\Clothing\train\Saree')
        baseDir=os.getcwd()
        try:
            image_content = requests.get(url).content

        except Exception:
            print(f"ERROR - unable to download {url} \n")

        try:
            image_file = io.BytesIO(image_content)
            image = Image.open(image_file).convert('RGB')

            loc = os.path.join(baseDir, file_name)

            with open(loc, 'wb') as f:
                image.save(f, "JPEG", quality=85)
            print(f"SAVED - {url} - AT: {loc}")
        except Exception:
            print(f"ERROR - unable to download {url} \n")
    elif i <=350:
        os.chdir(r'D:\Clothing\validate\Saree')
        baseDir=os.getcwd()
        try:
            image_content = requests.get(url).content

        except Exception:
            print(f"ERROR - unable to download {url} \n")

        try:
            image_file = io.BytesIO(image_content)
            image = Image.open(image_file).convert('RGB')

            loc = os.path.join(baseDir, file_name)

            with open(loc, 'wb') as f:
                image.save(f, "JPEG", quality=85)
            print(f"SAVED - {url} - AT: {loc}")
        except Exception:
            print(f"ERROR - unable to download {url} \n")
    else:
        os.chdir(r'D:\Clothing\test')
        baseDir=os.getcwd()
        try:
            image_content = requests.get(url).content

        except Exception:
            print(f"ERROR - unable to download {url} \n")

        try:
            image_file = io.BytesIO(image_content)
            image = Image.open(image_file).convert('RGB')

            loc = os.path.join(baseDir, file_name)

            with open(loc, 'wb') as f:
                image.save(f, "JPEG", quality=85)
            print(f"SAVED - {url} - AT: {loc}")
        except Exception:
            print(f"ERROR - unable to download {url} \n")
```

```
SAVED - https://m.media-amazon.com/images/I/91H5HBp9qBL._AC_UL320_.jpg - AT: D:\Clothing\train\Saree\S0.jpg
SAVED - https://m.media-amazon.com/images/I/91jBPCeWCAS._AC_UL320_.jpg - AT: D:\Clothing\train\Saree\S1.jpg
SAVED - https://m.media-amazon.com/images/I/310VyTHkonL._AC_UL320_.jpg - AT: D:\Clothing\train\Saree\S2.jpg
SAVED - https://m.media-amazon.com/images/I/91GKFgnS5uL._AC_UL320_.jpg - AT: D:\Clothing\train\Saree\S3.jpg
SAVED - https://m.media-amazon.com/images/I/91WCQ88UV5L._AC_UL320_.jpg - AT: D:\Clothing\train\Saree\S4.jpg
```

Firstly, we will start by importing required libraries and databases.

```
1   # import the Libraries as shown below
2
3   from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
4   from tensorflow.keras.models import Model
5   from tensorflow.keras.applications.resnet50 import ResNet50
6   from tensorflow.keras.applications.resnet50 import preprocess_input
7   from tensorflow.keras.preprocessing import image
8   from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_ima
9   from tensorflow.keras.models import Sequential
10  import numpy as np
11  from glob import glob
12  import matplotlib.pyplot as plt
13  import warnings
14  warnings.filterwarnings('ignore')
```

```
1   # re-size all the images to this
2   IMAGE_SIZE = [224, 224]
3
4   train_path = 'D:/Clothing/train' #location of train path
5   valid_path = 'D:/Clothing/validate' #location of test path
```

Import the library as shown below and add preprocessing layer to the front. Here we will be using imagenet weights.

```
resnet = ResNet50(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
# don't train existing weights
for layer in resnet.layers:
    layer.trainable = False
```

```
1   # useful for getting number of output classes
2   folders = glob('D:/Clothing/train/*')
```

```
1   # our Layers - you can add more if you want
2   x = Flatten()(resnet.output)
```

```
1   prediction = Dense(len(folders), activation='softmax')(x)
2
3   # create a model object
4   model = Model(inputs=resnet.input, outputs=prediction)
```

Let's view the structure of the model:

```
2   model.summary()
```

Model: "model"

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]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_1_conv[0][0]
conv2_block1_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][0]
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block1_1_relu[0][0]
conv2_block1_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block1_2_conv[0][0]
conv2_block1_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][0]
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block1_2_relu[0][0]
conv2_block1_0_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_0_conv[0][0]
conv2_block1_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block1_3_conv[0][0]
conv2_block1_add (Add)	(None, 56, 56, 256)	0	conv2_block1_0_bn[0][0] conv2_block1_3_bn[0][0]

Lets specify cost and optimization methods to use:

```
2  model.compile(
3     loss='categorical_crossentropy',
4     optimizer='adam',
5     metrics=['accuracy']
6  )|
```

I have used the Image Data Generator to import the images from the dataset.

```
2  from tensorflow.keras.preprocessing.image import ImageDataGenerator
3
4  train_datagen = ImageDataGenerator(rescale = 1./255,
5                                     shear_range = 0.2,
6                                     zoom_range = 0.2,
7                                     horizontal_flip = True)
8
9  test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
1  # Make sure you provide the same target size as initialied for the image size
2  training_set = train_datagen.flow_from_directory('D:/Clothing/train',
3                                     target_size = (224, 224),
4                                     batch_size = 32,
5                                     class_mode = 'categorical')
```

Found 983 images belonging to 3 classes.

```
1  test_set = test_datagen.flow_from_directory('D:/Clothing/validate',
2                                     target_size = (224, 224),
3                                     batch_size = 32,
4                                     class_mode = 'categorical')
```

Found 170 images belonging to 3 classes.

Let's fit the model:

```
2  # Run the cell. It will take some time to execute
3  r = model.fit_generator(
4     training_set,
5     validation_data=test_set,
6     epochs=50,
7     steps_per_epoch=len(training_set),
8     validation_steps=len(test_set)
9  )
```

```
Epoch 1/50
31/31 [==============================] - 107s 3s/step - loss: 3.6734 - accuracy: 0.4893 - val_loss: 1.2987 - val_accuracy:
0.6824
Epoch 2/50
31/31 [==============================] - 98s 3s/step - loss: 1.1206 - accuracy: 0.6714 - val_loss: 0.6916 - val_accuracy: 0.
7353
Epoch 3/50
31/31 [==============================] - 98s 3s/step - loss: 0.6520 - accuracy: 0.7314 - val_loss: 0.5791 - val_accuracy: 0.
7588
Epoch 4/50
31/31 [==============================] - 97s 3s/step - loss: 0.6079 - accuracy: 0.7447 - val_loss: 0.5520 - val_accuracy: 0.
7588
Epoch 5/50
31/31 [==============================] - 97s 3s/step - loss: 0.5000 - accuracy: 0.7650 - val_loss: 0.6892 - val_accuracy: 0.
7118
Epoch 6/50
31/31 [==============================] - 101s 3s/step - loss: 0.5994 - accuracy: 0.7447 - val_loss: 0.5347 - val_accuracy:
0.7471
Epoch 7/50
```

Let's have a view at accuracy and loss with the help of graph:

```
1   # plot the loss
2   plt.plot(r.history['loss'], label='train loss')
3   plt.plot(r.history['val_loss'], label='val loss')
4   plt.legend()
5   plt.show()
6   plt.savefig('LossVal_loss')
7
8   # plot the accuracy
9   plt.plot(r.history['accuracy'], label='train acc')
10  plt.plot(r.history['val_accuracy'], label='val acc')
11  plt.legend()
12  plt.show()
13  plt.savefig('AccVal_acc')
```
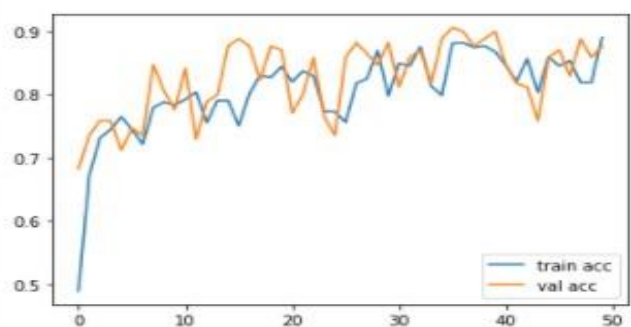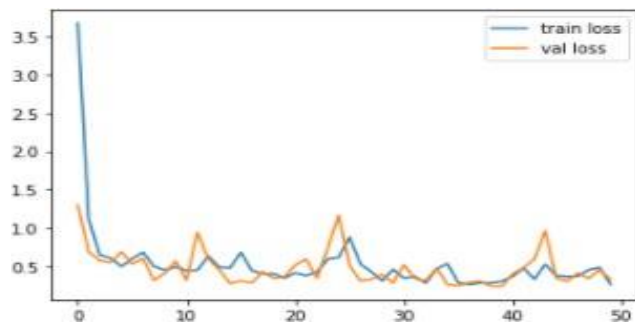
I have saved the model with the help of h5 file.

```
from tensorflow.keras.models import load_model

model.save('model_resnet50.h5')
```

With the help of saved model lets predict the data for test set:

```
1  y_pred = model.predict(test_set)
```

```
1  y_pred
```

```
[9.52050567e-01, 5.39246692e-08, 4.79494184e-02],
[2.56385114e-02, 5.07229148e-10, 9.74361420e-01],
[3.45090001e-07, 9.99999642e-01, 1.74707786e-08],
[1.87611300e-03, 3.52652059e-08, 9.98123825e-01],
[1.78535911e-03, 9.98054266e-01, 1.60412688e-04],
[2.81328010e-08, 1.00000000e+00, 3.71023070e-08],
[9.99128640e-01, 1.00156349e-07, 8.71315366e-04],
[8.85318995e-01, 5.89021568e-07, 1.14680342e-01],
[3.99160216e-09, 2.59644450e-16, 1.00000000e+00],
[1.13305659e-05, 9.99971986e-01, 1.66418977e-05],
[4.66057332e-04, 9.98090446e-01, 1.44349260e-03],
[1.29598344e-03, 9.98689234e-01, 1.48085574e-05],
[4.76938821e-02, 1.04874394e-08, 9.52306092e-01],
[9.96194243e-01, 7.78229492e-07, 3.80491395e-03],
[3.89383110e-16, 1.00000000e+00, 1.38062018e-11],
[9.55161989e-01, 2.37732376e-08, 4.48380597e-02],
[1.60035864e-01, 1.34241782e-04, 8.39829862e-01],
[7.05225579e-03, 5.45047962e-09, 9.92947698e-01],
[1.29695937e-01, 4.13365484e-08, 8.70303988e-01],
[8.29987228e-01, 2.09644786e-05, 1.69991747e-01]
```

I have used Numpy library to make the prediction readable.

```
1  import numpy as np
2  y_pred = np.argmax(y_pred, axis=1)
```

```
1  y_pred
```

```
array([1, 0, 1, 0, 1, 2, 0, 1, 1, 2, 1, 1, 2, 0, 1, 0, 1, 2, 1, 2, 1, 2,
       2, 1, 0, 1, 0, 0, 1, 2, 2, 0, 2, 2, 2, 0, 2, 1, 2, 1, 1, 0, 0, 2,
       1, 1, 1, 2, 0, 1, 0, 2, 2, 2, 0, 2, 0, 1, 2, 2, 0, 2, 2, 0, 1, 1,
       2, 2, 1, 2, 1, 1, 1, 0, 1, 2, 0, 1, 2, 0, 0, 1, 0, 0, 1, 2, 1, 0,
       1, 2, 0, 1, 2, 1, 1, 1, 2, 1, 2, 0, 1, 1, 2, 1, 0, 1, 2, 2, 2, 1,
       1, 2, 2, 0, 2, 1, 1, 1, 2, 0, 0, 0, 0, 0, 2, 1, 1, 2, 0, 2, 0, 2,
       1, 0, 1, 0, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 0, 2, 1, 2, 2,
       2, 1, 0, 1, 1, 2, 0, 0, 0, 1, 1, 2, 2, 2, 1, 0], dtype=int64)
```

```
1  from tensorflow.keras.models import load_model
```

```
1  model=load_model('model_resnet50.h5')  # Load the saved model
```

```
1  img=image.load_img('D:/Clothing/test/S361.jpg',target_size=(224,224))  #testing on new image
```

```
1  x=image.img_to_array(img)
2  x
```

```
array([[[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       ...,

       [[255., 252., 254.],
        [255., 252., 254.],
        [255., 253., 254.],
        ...,
        [251., 255., 255.],
        [251., 255., 255.],
        [254., 255., 255.]],

       [[255., 253., 252.],
        [255., 254., 252.],
        [254., 254., 254.],
        ...,
        [253., 254., 255.],
        [253., 254., 255.],
        [254., 255., 255.]],

       [[254., 254., 252.],
        [254., 254., 252.],
        [253., 255., 252.],
        ...,
        [255., 253., 255.],
        [255., 254., 255.],
        [255., 253., 255.]]], dtype=float32)
```

```
1  x.shape #shape of the image
```

```
(224, 224, 3)
```

```
1  x=x/255 # rescaling the image
```

```
1  import numpy as np
2  x=np.expand_dims(x,axis=0)
3  img_data=preprocess_input(x)
4  img_data.shape # new shape of the image
```

```
(1, 224, 224, 3)
```

```
1    img_data
```

```
array([[[[-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         ...,
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ]],

        [[-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         ...,
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ]],

        [[-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         ...,
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ],
         [-102.939    , -115.779    , -122.68     ]],

        ...,

        [[-102.942924, -115.790764, -122.68     ],
         [-102.942924, -115.790764, -122.68     ],
         [-102.942924, -115.78684 , -122.68     ],
         ...,
         [-102.939    , -115.779    , -122.69569 ],
         [-102.939    , -115.779    , -122.69569 ],
         [-102.939    , -115.779    , -122.68392 ]],

        [[-102.95077 , -115.78684 , -122.68     ],
         [-102.95077 , -115.78292 , -122.68     ],
         [-102.942924, -115.78292 , -122.68392 ],
         ...,
         [-102.939    , -115.78292 , -122.68784 ],
         [-102.939    , -115.78292 , -122.68784 ],
         [-102.939    , -115.779    , -122.68392 ]],

        [[-102.95077 , -115.78292 , -122.68392 ],
         [-102.95077 , -115.78292 , -122.68392 ],
         [-102.95077 , -115.779    , -122.68784 ],
         ...,
         [-102.939    , -115.78684 , -122.68     ],
         [-102.939    , -115.78292 , -122.68     ],
         [-102.939    , -115.78684 , -122.68     ]]]], dtype=float32)
```

```
1    model.predict(img_data)
```

```
array([[7.423415e-14, 1.000000e+00, 6.745252e-10]], dtype=float32)
```

```
1    a=np.argmax(model.predict(img_data), axis=1) #prediction over test image
```

```
1    a==1 # checking that its a saree or not as 1 is denotation for saree
```

```
array([ True])
```

```
1    a==0 # checking that its a jeans or not as 0 is denotation for jeans
```

```
array([False])
```

```
1    a==2 # checking that its a trouser or not as 0 is denotation for trouser
```

```
array([False])
```

CONCLUSION

In this Image Scraping and Classification Project, firstly with the help of selenium I have scrapped images of saree, jeans and trouser from Amazon.com. With the help of tensorflow library I have build image classification model. Started with adding pre-processing layer and then created model object.

With the help of compile method specified cost and optimization method to use. Image data Generator and flow from directory methods to import images and equalized same target size as initialized for the image size. Fitted the model.

Lastly, I have tried to review data loss and accuracy with the help of graph. Saved file with the help of h5. I have the loaded the saved model and then predicted over new image.