FLIP ROBO

Image Classification Project

Submitted by:

SUDHANSU MANDAL

INTRODUCTION

➢ Business Problem Framing

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.

➢ Conceptual Background of the Domain Problem

E-commerce industry is fast growing in worldwide. Amazon Company is leading in market in India where they are providing wide range of assortment and choices for customer. For any E commerce company designing the apps and content in apps about the product information play very vital role. Company has to depend on suppliers to get the product images and their details. To have correct information from supplier NLP algorithm plays very important role where the algorithm can predict the correct category of product by processing the images of product.

➢ Review of Literature

Here in this project we are using Image Classification algorithm using Keras as well as Tensor flow. Image Classification is a Machine Learning module that trains itself from an existing dataset of multiclass images and develops a model for future prediction of similar images not encountered during training.To complete the project, we have followed below process:

1. **Understand the problem:** Before getting the data, we need to understand the problem statement and the objective of the project. We need to understand the problem we are trying to solve with the help of data science.
2. **Collect Data:** In this step, we have collected the images of saree, men's jeans and men's trouser from Amazon website through web scrapping..
3. **Data Pre-Processing:** We have done pre-processing through ImageDataGenerator and also downloaded VGG-16 weights, by including the top layer parameter as false. We have also frozen the layer and modified the last layer.
4. **Model Training:** we have trained the model with 5 epochs.
5. **Model Testing:** In this step we have tested our model on test dataset.

➢ Motivation for the Problem Undertaken

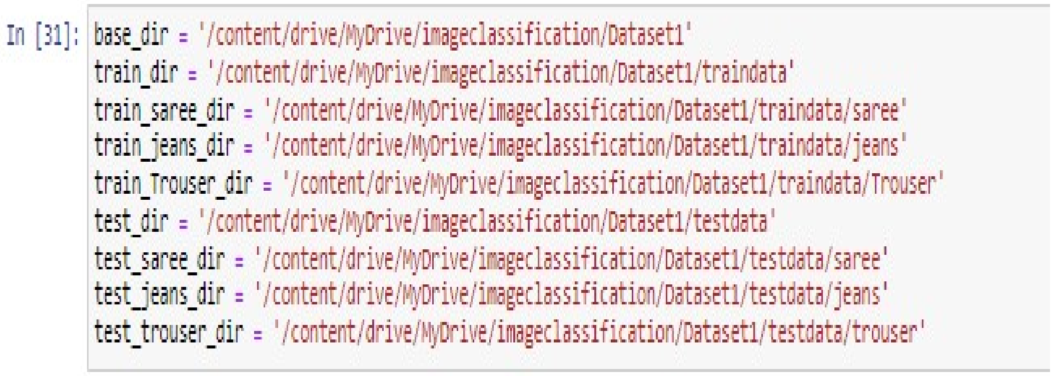
Here in this project we are using Image Classification algorithm using Keras as well as Tensor flow. Image Classification is a Machine Learning module that trains itself from an existing dataset of multiclass images and develops a model for future prediction of similar images not encountered during training.

ANALYTICAL PROBLEM FRAMING

➢ Data Sources and their formats

The data is in .jpg format. We have scrapped 867 images from Amazon website which having saree, men’s jeans and men’s trouser images. For uploading images into the model we have created dictionary path as shown in Fig 1.

Fig 1:



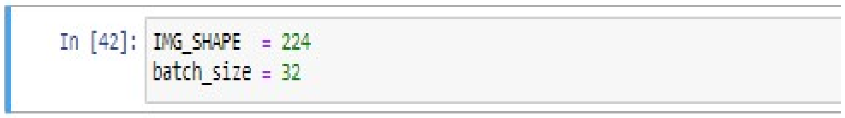
```
In [31]: base_dir = '/content/drive/MyDrive/imageclassification/Dataset1'
         train_dir = '/content/drive/MyDrive/imageclassification/Dataset1/traindata'
         train_saree_dir = '/content/drive/MyDrive/imageclassification/Dataset1/traindata/saree'
         train_jeans_dir = '/content/drive/MyDrive/imageclassification/Dataset1/traindata/jeans'
         train_Trouser_dir = '/content/drive/MyDrive/imageclassification/Dataset1/traindata/Trouser'
         test_dir = '/content/drive/MyDrive/imageclassification/Dataset1/testdata'
         test_saree_dir = '/content/drive/MyDrive/imageclassification/Dataset1/testdata/saree'
         test_jeans_dir = '/content/drive/MyDrive/imageclassification/Dataset1/testdata/jeans'
         test_trouser_dir = '/content/drive/MyDrive/imageclassification/Dataset1/testdata/trouser'
```

➢ Data Pre-processing

As we know Pre-Processing of our data is an important step for our model to perform better but in this project we will apply image data generator algorithm to import images and applying rescaling and shuffling .

- **Setting up Images size:**
  As shown in Fig 2, here we have set the size (height, width) of images. This step mostly needs when dataset images have different sizes, it will speed up the training process. I used an image shape of (224,224) because VGG-16 algorithm accepts only this image size.

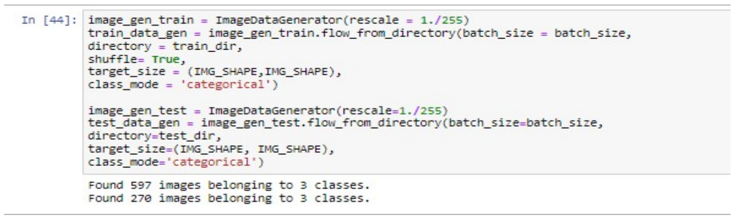
  Fig 2.



```
In [42]: IMG_SHAPE  = 224
         batch_size = 32
```

- **Image importing:**
  As shown in Fig 3, In this step we have pre-processed our data (train and test), which includes, rescaling and shuffling. also using the Image Data Generator to import the images from the dataset

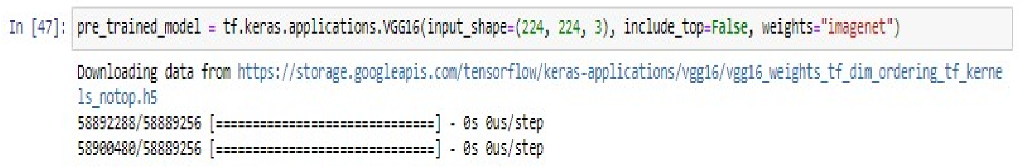
  Fig 3.



```
In [44]: image_gen_train = ImageDataGenerator(rescale = 1./255)
         train_data_gen = image_gen_train.flow_from_directory(batch_size = batch_size,
         directory = train_dir,
         shuffle= True,
         target_size = (IMG_SHAPE,IMG_SHAPE),
         class_mode = 'categorical')

         image_gen_test = ImageDataGenerator(rescale=1./255)
         test_data_gen = image_gen_test.flow_from_directory(batch_size=batch_size,
         directory=test_dir,
         target_size=(IMG_SHAPE, IMG_SHAPE),
         class_mode='categorical')

         Found 597 images belonging to 3 classes.
         Found 270 images belonging to 3 classes.
```

- **Downloading VGG-16:**
  As shown in Fig 4, In this step we have downloaded VGG-16 weights, by including the top layer parameter as false.

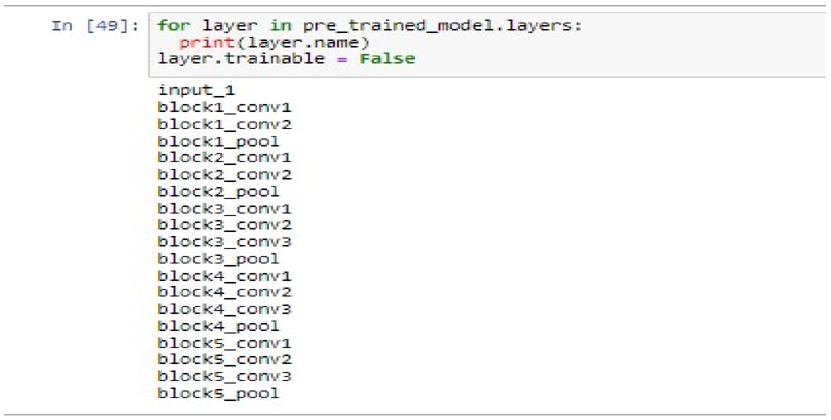
Fig 4.



```
In [47]: pre_trained_model = tf.keras.applications.VGG16(input_shape=(224, 224, 3), include_top=False, weights="imagenet")

         Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kerne
         ls_notop.h5
         58892288/58889256 [==============================] - 0s 0us/step
         58900480/58889256 [==============================] - 0s 0us/step
```

- **Freezing the training layer:**
  As shown in Fig 5, In this step we have freeze the training layers of VGG-16 because VGG-16, already trained on huge data.

Fig 5.

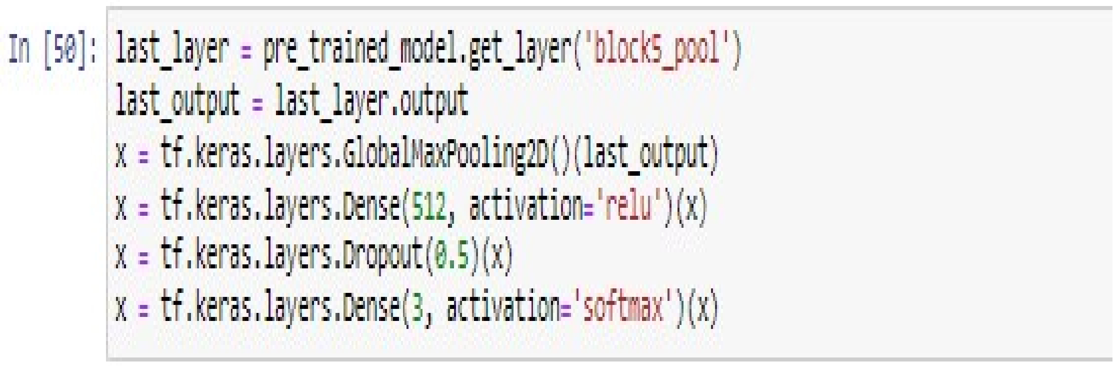


```
In [49]: for layer in pre_trained_model.layers:
           print(layer.name)
         layer.trainable = False

         input_1
         block1_conv1
         block1_conv2
         block1_pool
         block2_conv1
         block2_conv2
         block2_pool
         block3_conv1
         block3_conv2
         block3_conv3
         block3_pool
         block4_conv1
         block4_conv2
         block4_conv3
         block4_pool
         block5_conv1
         block5_conv2
         block5_conv3
         block5_pool
```

- **Modifying the last layer:**

  As shown in Fig 6, In this step we have modified the last layer for our classes as all layers of VGG-16 are frozen. We have added one max polling, one dense layer, one dropout, and one output with the last layer of VGG-16. Since the problem having multiclass so the last dense layer has chosen with 3 and activation with softmax.

Fig 6.



```
In [50]: last_layer = pre_trained_model.get_layer('block5_pool')
         last_output = last_layer.output
         x = tf.keras.layers.GlobalMaxPooling2D()(last_output)
         x = tf.keras.layers.Dense(512, activation='relu')(x)
         x = tf.keras.layers.Dropout(0.5)(x)
         x = tf.keras.layers.Dense(3, activation='softmax')(x)
```

➢ Hardware and Software Requirements and Tools Used

In this project the mention below hardware, software and tools used to complete this project:

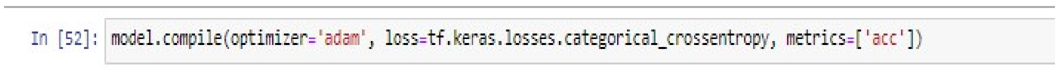
✓ Hardware:	
Processor	Intel(R) Pentium(R) CPU 3825U @ 1.90GHz 1.90 GHz
Installed RAM	4.00 GB
System type	64-bit operating system, x64-based processor
✓ Software:	
Edition	Windows 10 Home Single Language
Anaconda	

✓ Library & Tools:
Jupyter Notebook
Pandas
NumPy
Matplotlib
Seaborn
Keras
Tenser flow

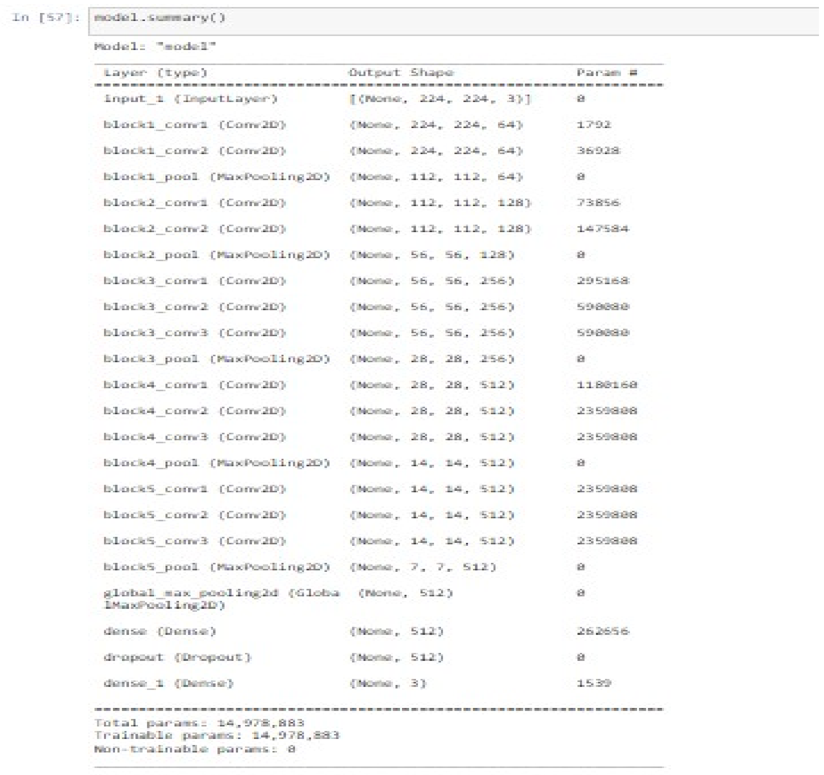
Model/s Development and Evaluation

As shown in Fig 7 here we have compiling the model before starting training and since problem is multiclass have chosen categorical_crossentropy and checking model summary.

Fig 7



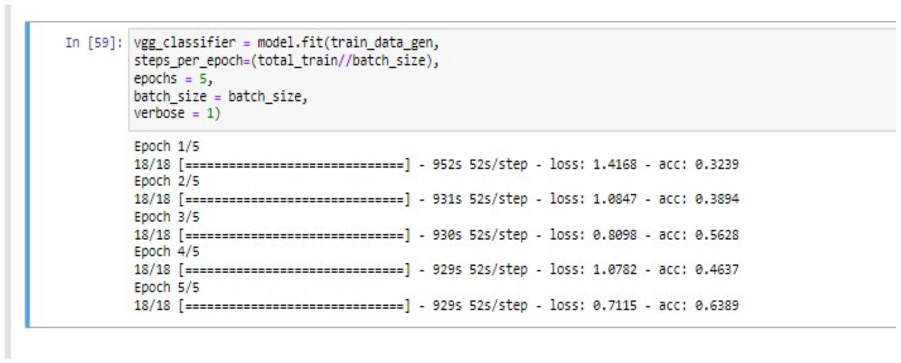
```
In [52]: model.compile(optimizer='adam', loss=tf.keras.losses.categorical_crossentropy, metrics=['acc'])
```



```
In [57]: model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 global_max_pooling2d (Globa  (None, 512)              0
 lMaxPooling2D)

 dense (Dense)               (None, 512)               262656

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 3)                 1539

=================================================================
Total params: 14,978,883
Trainable params: 14,978,883
Non-trainable params: 0
_________________________________________________________________
```

As shown in Fig 8. We have trained the model and test the model in testing dataset.

Fig 8.



```
In [59]: vgg_classifier = model.fit(train_data_gen,
         steps_per_epoch=(total_train//batch_size),
         epochs = 5,
         batch_size = batch_size,
         verbose = 1)

         Epoch 1/5
         18/18 [==============================] - 952s 52s/step - loss: 1.4168 - acc: 0.3239
         Epoch 2/5
         18/18 [==============================] - 931s 52s/step - loss: 1.0847 - acc: 0.3894
         Epoch 3/5
         18/18 [==============================] - 930s 52s/step - loss: 0.8098 - acc: 0.5628
         Epoch 4/5
         18/18 [==============================] - 929s 52s/step - loss: 1.0782 - acc: 0.4637
         Epoch 5/5
         18/18 [==============================] - 929s 52s/step - loss: 0.7115 - acc: 0.6389
```

CONCLUSION

➢ Learning Outcomes of the Study in respect of Data Science

Below are the Learning Outcome of the Study:

- ✓ Collecting the image data from website, where after scraping data would be not in proper format.
- ✓ Our data are in image type so have applied image generator for uploading the data into the model.

➢ Limitations of this work and Scope for Future Work

Mention below are the limitations and future scope steps:

- ✓ We could train our data with more number of Epochs like 50-60 Epochs to get more accuracy.