FLIP ROBO

Image Scraping and Classification Project

Submitted by:

Mekhala Misra

ACKNOWLEDGMENT

I took help from following websites:

1)Geek for geeks

2)tensorflow documentation

3)researchgate.net

4)youtube videos

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. So here we are working on this project in 2 phases:

  1. Data Collection: I collected images of saree, trousers and jeans from amazon.com.
  2. Analysing the image and building the model.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

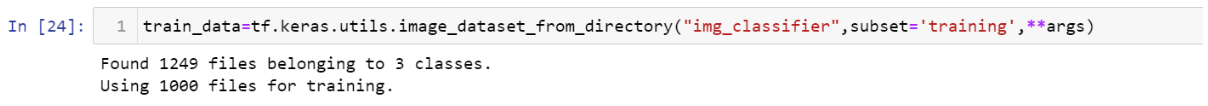
  This is a deep learning problem in which we have to analyse the features of few images and classify them into 3 classes- saree, trousers and jeans.

- Data Sources and their formats

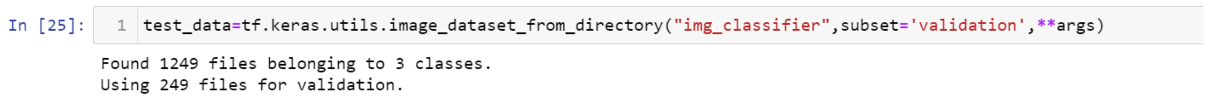
  We collected approx 400 images of each class saree, trousers and jeans from amazon.com in the jpg format and saved each class images in there respective folders . Further all the categorical folders are saved in image_classifier folder.

- ## Data Pre-processing

1) The dataset is divided into training set and testing set.

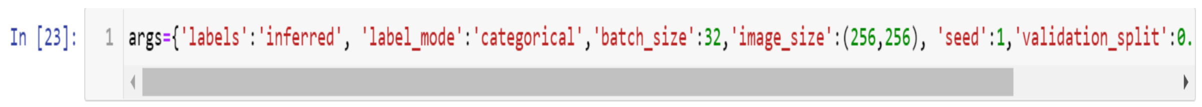


```
In [24]:  train_data=tf.keras.utils.image_dataset_from_directory("img_classifier",subset='training',**args)

Found 1249 files belonging to 3 classes.
Using 1000 files for training.
```



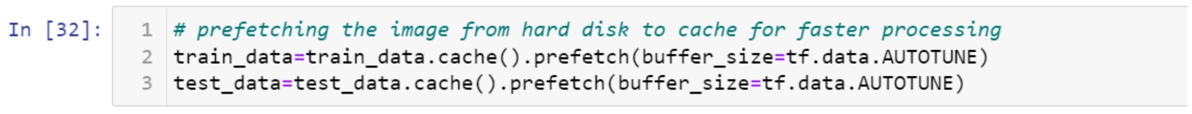
```
In [25]:  test_data=tf.keras.utils.image_dataset_from_directory("img_classifier",subset='validation',**args)

Found 1249 files belonging to 3 classes.
Using 249 files for validation.
```

Here args is a set of following parameters:



```
In [23]:  args={'labels':'inferred', 'label_mode':'categorical','batch_size':32,'image_size':(256,256), 'seed':1,'validation_split':0.
```

2) Further we prefetched the training and testing data from hard disk to cache for faster processing.



```
In [32]:  # prefetching the image from hard disk to cache for faster processing
          train_data=train_data.cache().prefetch(buffer_size=tf.data.AUTOTUNE)
          test_data=test_data.cache().prefetch(buffer_size=tf.data.AUTOTUNE)
```

- ## Hardware and Software Requirements and Tools Used

We imported following packages:

import tensorflow as tf
from PIL import Image
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import pandas as pd
import numpy as np
import itertools
import base64
import iofrom sklearn.feature_extraction.text import TfidfVectorizer
import warnings
warnings.filterwarnings('ignore')
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import stopwords
import string

Model/s Development and Evaluation

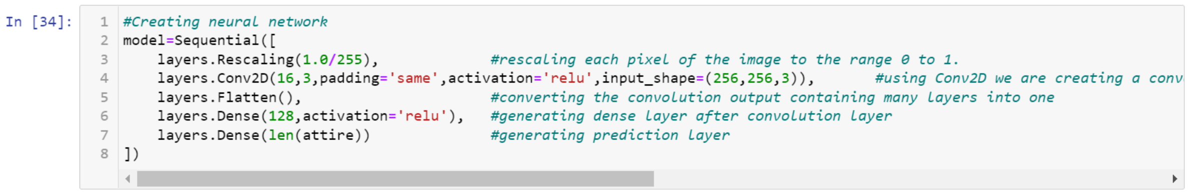
- ### Identification of possible problem-solving approaches (methods)
- Since this is a deep learning classification problem so we are using CNN algorithm for building neural network with 2-3 layers and making predictions.

- ### Testing of Identified Approaches (Algorithms)

- We applied CNN algorithm and build a neural network of 2 layers a convolution layer and a dense layer.
- Further we compiled the model to reduce loss by using keras module categorical crossentropy method.
- Then the model was trained with training data with epochs as 5.
- We checked the performance and concluded that the model is overfitting as in the last epoch training accuracy is 91% and testing accuracy is 73%.
- In order to increase accuracy and remove the problem of overfitting we added more layers to our neural network with higher feature extraction from images.
- This increases training accuracy to 97% and testing to 91% thus removing overfitting as well.

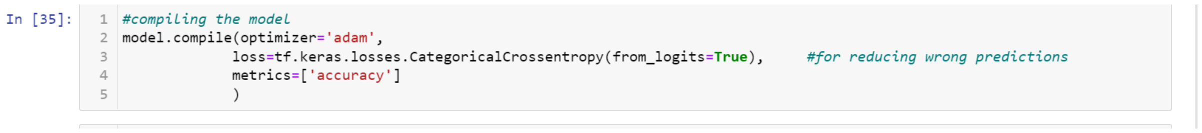
- ### Run and Evaluate selected models

- We applied CNN algorithm and build a neural network of 2 layers a convolution layer and a dense layer.



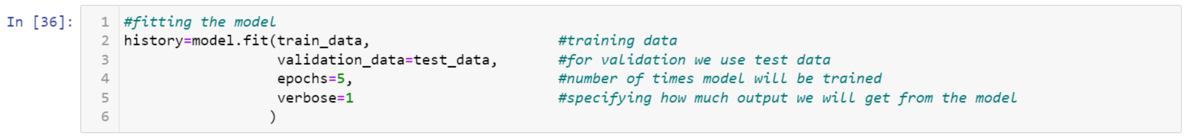
```
In [34]:
#Creating neural network
model=Sequential([
    layers.Rescaling(1.0/255),              #rescaling each pixel of the image to the range 0 to 1.
    layers.Conv2D(16,3,padding='same',activation='relu',input_shape=(256,256,3)),       #using Conv2D we are creating a conv
    layers.Flatten(),                       #converting the convolution output containing many layers into one
    layers.Dense(128,activation='relu'),    #generating dense layer after convolution layer
    layers.Dense(len(attire))               #generating prediction layer
])
```

- Further we compiled the model to reduce loss by using keras module categorical crossentropy method.



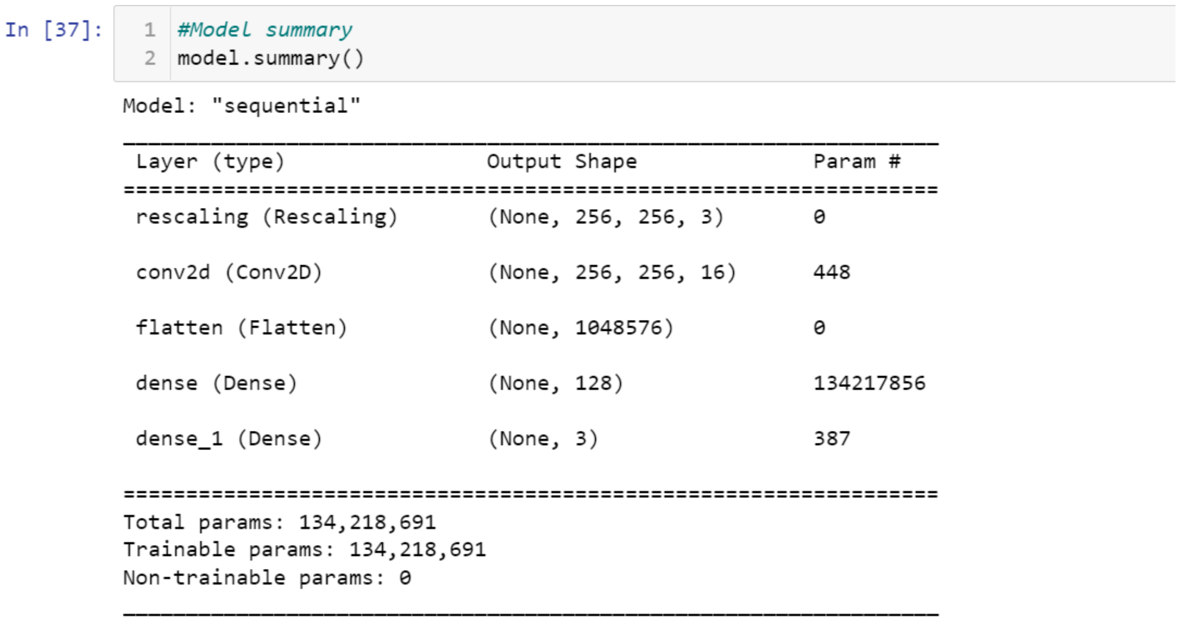
```
In [35]:
#compiling the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),     #for reducing wrong predictions
              metrics=['accuracy']
              )
```

- Then the model was trained with training data with epochs as 5.



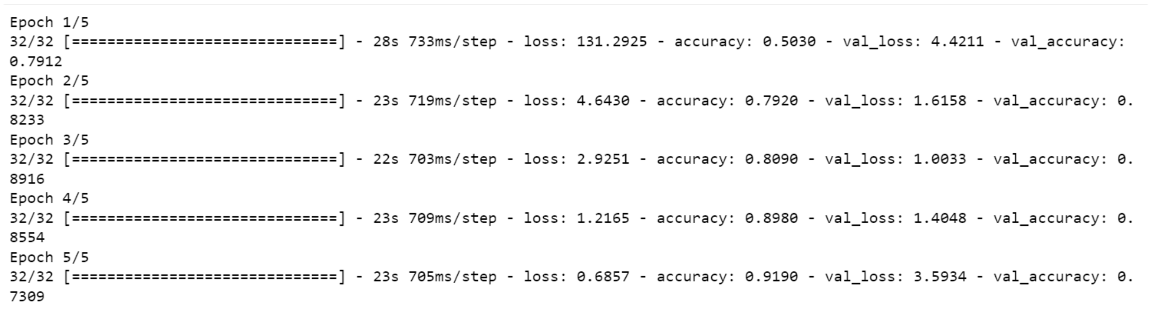
```
In [36]:
#fitting the model
history=model.fit(train_data,                           #training data
                  validation_data=test_data,            #for validation we use test data
                  epochs=5,                             #number of times model will be trained
                  verbose=1                             #specifying how much output we will get from the model
                 )
```

- Here is the model summary, it has 5 layers for rescaling the image, convolution layer, flattening the image and then two dense layers:



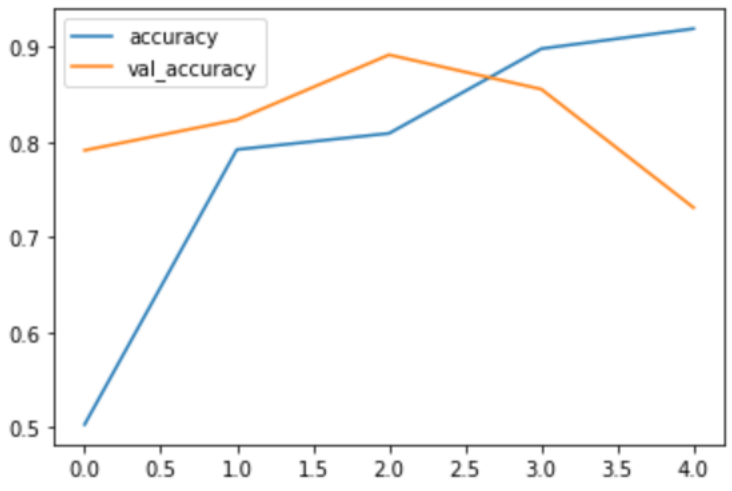
```
In [37]:
#Model summary
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 256, 256, 3)       0

 conv2d (Conv2D)             (None, 256, 256, 16)      448

 flatten (Flatten)           (None, 1048576)           0

 dense (Dense)               (None, 128)               134217856

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 134,218,691
Trainable params: 134,218,691
Non-trainable params: 0
_________________________________________________________________
```

- We checked the performance and concluded that the model is overfitting as in the last epoch training accuracy is 91% and testing accuracy is 73%.

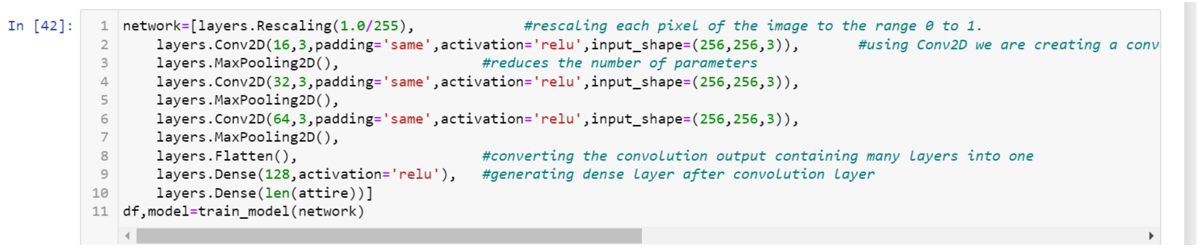


```
Epoch 1/5
32/32 [==============================] - 28s 733ms/step - loss: 131.2925 - accuracy: 0.5030 - val_loss: 4.4211 - val_accuracy:
0.7912
Epoch 2/5
32/32 [==============================] - 23s 719ms/step - loss: 4.6430 - accuracy: 0.7920 - val_loss: 1.6158 - val_accuracy: 0.
8233
Epoch 3/5
32/32 [==============================] - 22s 703ms/step - loss: 2.9251 - accuracy: 0.8090 - val_loss: 1.0033 - val_accuracy: 0.
8916
Epoch 4/5
32/32 [==============================] - 23s 709ms/step - loss: 1.2165 - accuracy: 0.8980 - val_loss: 1.4048 - val_accuracy: 0.
8554
Epoch 5/5
32/32 [==============================] - 23s 705ms/step - loss: 0.6857 - accuracy: 0.9190 - val_loss: 3.5934 - val_accuracy: 0.
7309
```

We could notice validation accuracy after 5 epoch is 0.739 or 73%.

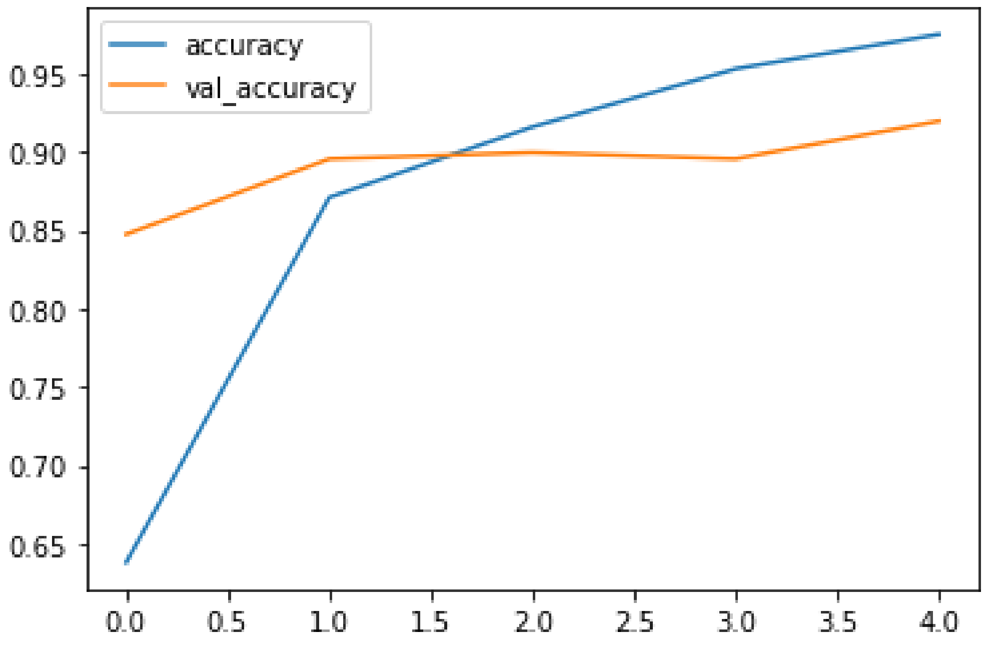


- In order to increase accuracy and remove the problem of overfitting we added more layers to our neural network with higher feature extraction from images.

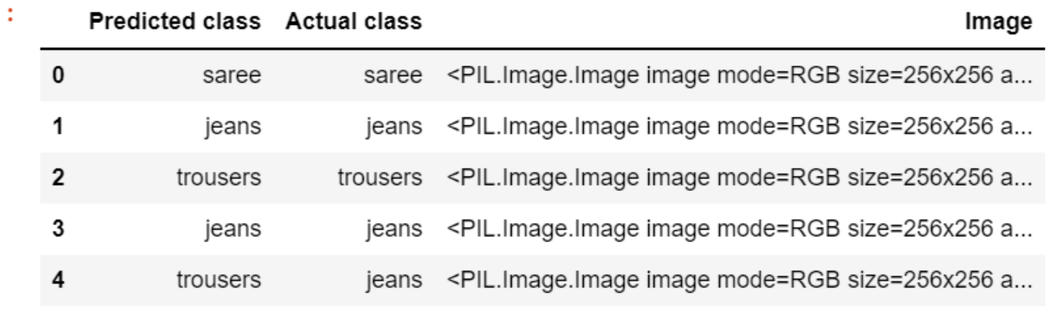


In [42]:
```
network=[layers.Rescaling(1.0/255),            #rescaling each pixel of the image to the range 0 to 1.
    layers.Conv2D(16,3,padding='same',activation='relu',input_shape=(256,256,3)),       #using Conv2D we are creating a conv
    layers.MaxPooling2D(),                #reduces the number of parameters
    layers.Conv2D(32,3,padding='same',activation='relu',input_shape=(256,256,3)),
    layers.MaxPooling2D(),
    layers.Conv2D(64,3,padding='same',activation='relu',input_shape=(256,256,3)),
    layers.MaxPooling2D(),
    layers.Flatten(),                     #converting the convolution output containing many layers into one
    layers.Dense(128,activation='relu'),   #generating dense layer after convolution layer
    layers.Dense(len(attire))]
df,model=train_model(network)
```

- This neural network removed overfitting problem and increases model accuracy to 91%.



- Later on we used this model to make predictions and here is the glimpse of predicted and true value.



	Predicted class	Actual class	Image
0	saree	saree	<PIL.Image.Image image mode=RGB size=256x256 a...
1	jeans	jeans	<PIL.Image.Image image mode=RGB size=256x256 a...
2	trousers	trousers	<PIL.Image.Image image mode=RGB size=256x256 a...
3	jeans	jeans	<PIL.Image.Image image mode=RGB size=256x256 a...
4	trousers	jeans	<PIL.Image.Image image mode=RGB size=256x256 a...