FLIP ROBO

Image Scraping and Classification project

Submitted by:

Allen Paul

ACKNOWLEDGMENT

- https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb
- http://www.sc.chula.ac.th/courseware/2309507/Lecture/remote18.htm
- https://www.kdnuggets.com/2018/12/solve-image-classification-problem-quickly-easily.html
- https://www.einfochips.com/blog/understanding-image-recognition-and-its-uses/

INTRODUCTION

• Business Problem Framing

Deep learning is fast becoming a key instrument in artificial intelligence applications. For example, in areas such as computer vision, natural language processing, and speech recognition, deep learning has been producing remarkable results. Therefore, there is a growing interest in deep learning. One of the problems where deep learning excels is image classification. The goal in image classification is to classify a specific picture according to a set of possible categories.

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.

• Conceptual Background of the Domain Problem

The convolutional neural network (CNN) is a class of deep learning neural networks. CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyse visual imagery and are frequently working behind the scenes in image classification. They can be found at the core of everything from Facebook's photo tagging to self-driving cars. They're working hard behind the scenes in everything from healthcare to security. They're fast and they're efficient.

• Review of Literature

Image classification is a complex process that may be affected by many factors. Image classification has made great progress over the past decades in the following three areas: (1) development and use of advanced classification algorithms, such as subpixel, per-field, and knowledge-based classification algorithms; (2) use of multiple remote-sensing features, including spectral, spatial, multitemporal, and multisensor information; and (3) incorporation of ancillary data into classification procedures, including such data as topography, soil, road, and census data. Accuracy assessment is an integral part in an image classification procedure. Accuracy assessment based on error matrix is the most commonly employed approach for evaluating per-pixel classification, while fuzzy approaches are gaining attention for assessing fuzzy classification results. Uncertainty and error propagation in the image-processing chain is an important factor influencing classification accuracy. Identifying the weakest links in the chain and then reducing the uncertainties are critical for improvement of classification accuracy. The study of uncertainty will be an important topic in the future research of image classification.

• Motivation for the Problem Undertaken

Since this task of recognizing a visual concept (e.g. cat) is relatively trivial for a human to perform, it is worth considering the challenges involved from the perspective of a Computer Vision algorithm. As we present (an inexhaustive) list of challenges below, keep in mind the raw representation of images as a 3-D array of brightness values:

- Viewpoint variation. A single instance of an object can be oriented in many ways with respect to the camera.
- Scale variation. Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).
- Deformation. Many objects of interest are not rigid bodies and can be deformed in extreme ways.
- Occlusion. The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.
- Illumination conditions. The effects of illumination are drastic on the pixel level.
- Background clutter. The objects of interest may *blend* into their environment, making them hard to identify.
- Intra-class variation. The classes of interest can often be relatively broad, such as *chair*. There are many different types of these objects, each with their own appearance.

A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

In this Project there are two phases: Data Collection and Mode Building.

Data Collection Phase: In this section, we have scraped images from Amazon.com. The clothing categories used for scraping are:

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

Model Building Phase: Using the above data we need to build an image classification model that will classify between these 3 categories mentioned above.

The convolutional neural network (CNN) is a class of deep learning neural networks. CNNs represent a huge breakthrough in image recognition.

A CNN has Convolutional layers, ReLU layers, Pooling layers, a Fully connected layer. A CNN convolves learned features with input data and uses 2D convolutional layers. This means that this type of network is ideal for processing 2D images. Compared to other image classification algorithms, CNNs actually use very little pre-processing. This means that they can learn the filters that have to be hand-made in other algorithms. CNNs can be used in tons of applications from image and video recognition, image classification, and recommender systems to natural language processing and medical image analysis.

- ## Data Sources and their formats.

  We have scraped images of Sarees, Trousers and Jeans from amazon.in using selenium.

  Sample of the data/images scraped are as shown below:

  Sample of images of Sarees:

  Sample of images of Jeans:

Sample of images of Trousers:

We have then divided them into train and test data and them imported them into the jupyter notebook.

- ## Data Pre-processing Done

CNN Architecture:

Images and photos are structured differently from your normal data stored in CSVs and contain even more information if they are coloured. This means that we will have to first transform photos into arrays of numbers that can be understood by the network

A CNN is made up of hidden layers that are attached to a fully-connected layer which handles the classification decisions based on the pixel information flowing through the previous layers

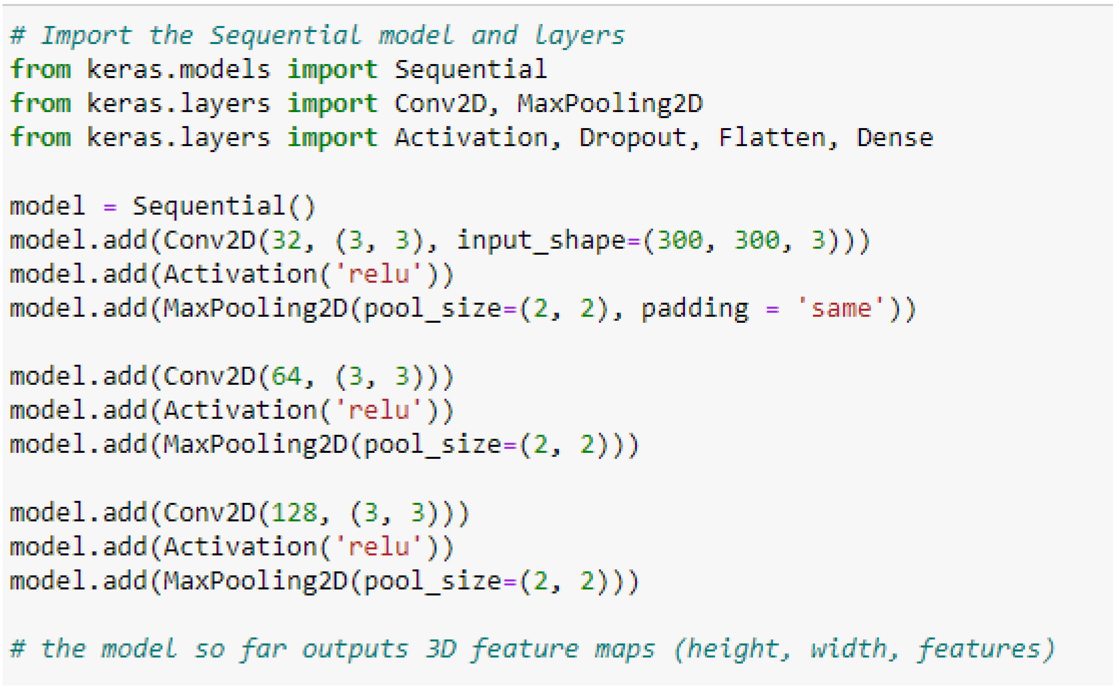
Our CNN Architecture will consist of multiple layers:

Convolutional layer: This layer is the first to extract information from the image by moving over it, a few pixels at a time and applying the convolutional operation. The output of these pixels will be the dot product of the filter value and the pixel value. This process continues for the rest of the pixels and we end with the entire image convolved to a feature map.

Max Pooling: The Pooling layer reduces the size of this feature map to only retain important information and make the training process easier and quicker. In our CNN, we will be using a Max Pooling layer which will move over the feature map in a similar fashion like the previous filter and only take the maximum value of the cluster over which it is focussing.

Fully Connected Layer: The Fully Connected Layer is the usual neural network and in our case, it is composed of 2 layers; one with 64 nodes and the last one with a single output node for binary classification.

The code to built a CNN is shown Below:



```python
# Import the Sequential model and layers
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(300, 300, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), padding = 'same'))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# the model so far outputs 3D feature maps (height, width, features)
```

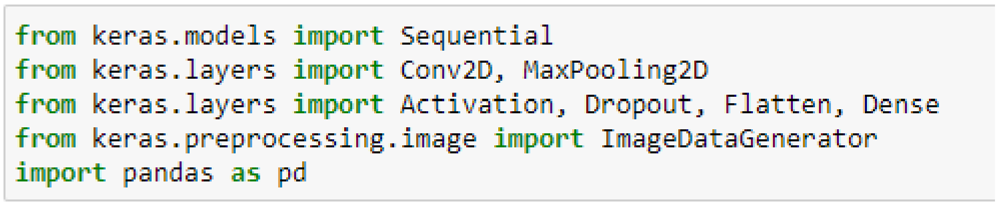
```
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('sigmoid'))

model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
model.summary()

batch_size = 5
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 298, 298, 32)      896
_________________________________________________________________
activation (Activation)      (None, 298, 298, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 147, 147, 64)      18496
_________________________________________________________________
activation_1 (Activation)    (None, 147, 147, 64)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 128)       73856
_________________________________________________________________
activation_2 (Activation)    (None, 71, 71, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 35, 35, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 156800)            0
_________________________________________________________________
dense (Dense)                (None, 64)                10035264
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
_________________________________________________________________
activation_4 (Activation)    (None, 3)                 0
=================================================================
Total params: 10,128,707
Trainable params: 10,128,707
Non-trainable params: 0
_________________________________________________________________
```

- Data Inputs- Logic- Output Relationships

  In this problem the images of jeans, saree and trousers scraped from the amazon.in are the inputs and after training the model using deep learning we use this model to predict the probability of an image to be jeans, saree and trousers is the output.

- Hardware and Software Requirements and Tools Used



```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.preprocessing.image import ImageDataGenerator
import pandas as pd
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

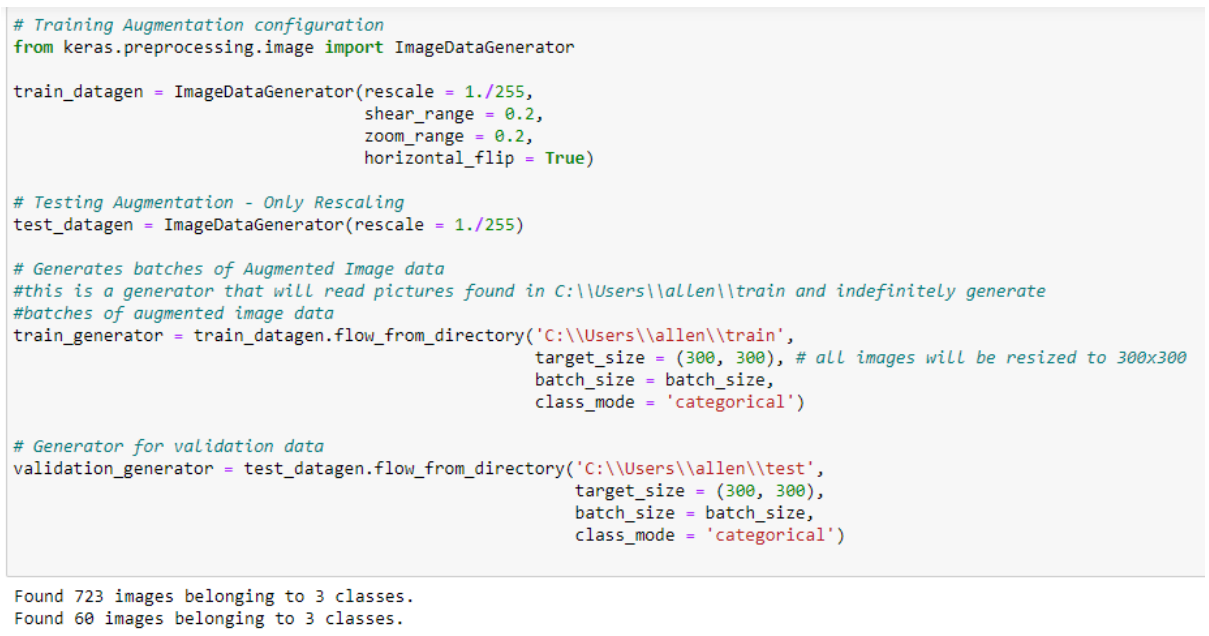
  CNN is one of the most useful techniques in solving the image classification problem. We must see to it that all the images have proper resolution, scaling and all the images should be of same size in order to make a model that predict accurately. Data pre-processing on the input/image data must be done properly.

- Testing of Identified Approaches (Algorithms)

  In this problem we have used ImageDataGenerator model in order to train the model.

- Run and Evaluate selected models

Augmentation:



```
# Training Augmentation configuration
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

# Testing Augmentation - Only Rescaling
test_datagen = ImageDataGenerator(rescale = 1./255)

# Generates batches of Augmented Image data
#this is a generator that will read pictures found in C:\\Users\\allen\\train and indefinitely generate
#batches of augmented image data
train_generator = train_datagen.flow_from_directory('C:\\Users\\allen\\train',
                                                    target_size = (300, 300), # all images will be resized to 300x300
                                                    batch_size = batch_size,
                                                    class_mode = 'categorical')

# Generator for validation data
validation_generator = test_datagen.flow_from_directory('C:\\Users\\allen\\test',
                                                        target_size = (300, 300),
                                                        batch_size = batch_size,
                                                        class_mode = 'categorical')
```

```
Found 723 images belonging to 3 classes.
Found 60 images belonging to 3 classes.
```

We can see from above that there are 723 images in training set and 60 images in testing set and they belong to 3 classes. Namely saree, jeans and trouser. Having 20 images each in testing set and 241 images each in training.

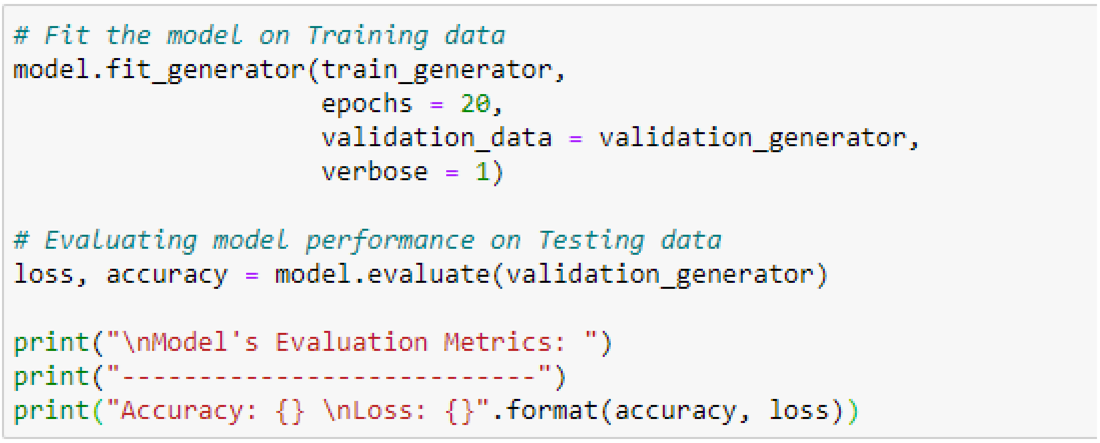
In Keras, this can be done via the keras.preprocessing.image.ImageDataGenerator class. This class allows you to

- Configure random transformations and normalization operations to be done on your image data during training
- Rescale is a value by which we will multiply the data before any other processing. Our original images consist in RGB coefficients in the 0-255, but such values would be too high for our models to process (given a typical learning rate), so we target values between 0 and 1 instead by scaling with a 1/255. factor.
- Shear range is for randomly applying shearing transformations
- Zoom range is for randomly zooming inside pictures
- Horizontal flip is for randomly flipping half of the images horizontally --relevant when there are no assumptions of horizontal asymmetry (e.g. real-world pictures).

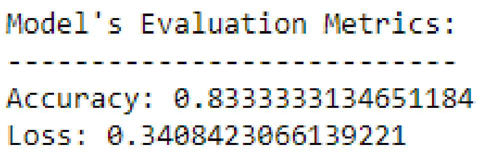
- ## Key Metrics for success in solving problem under consideration

  The success of the model is governed by accuracy score and loss. The more the accuracy and lower the loss, the better the model.

- ## Interpretation of the Results

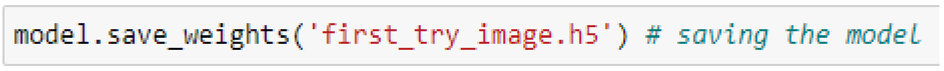


```python
# Fit the model on Training data
model.fit_generator(train_generator,
                    epochs = 20,
                    validation_data = validation_generator,
                    verbose = 1)

# Evaluating model performance on Testing data
loss, accuracy = model.evaluate(validation_generator)

print("\nModel's Evaluation Metrics: ")
print("---------------------------")
print("Accuracy: {} \nLoss: {}".format(accuracy, loss))
```



```
Model's Evaluation Metrics: 
---------------------------
Accuracy: 0.8333333134651184 
Loss: 0.3408423066139221
```

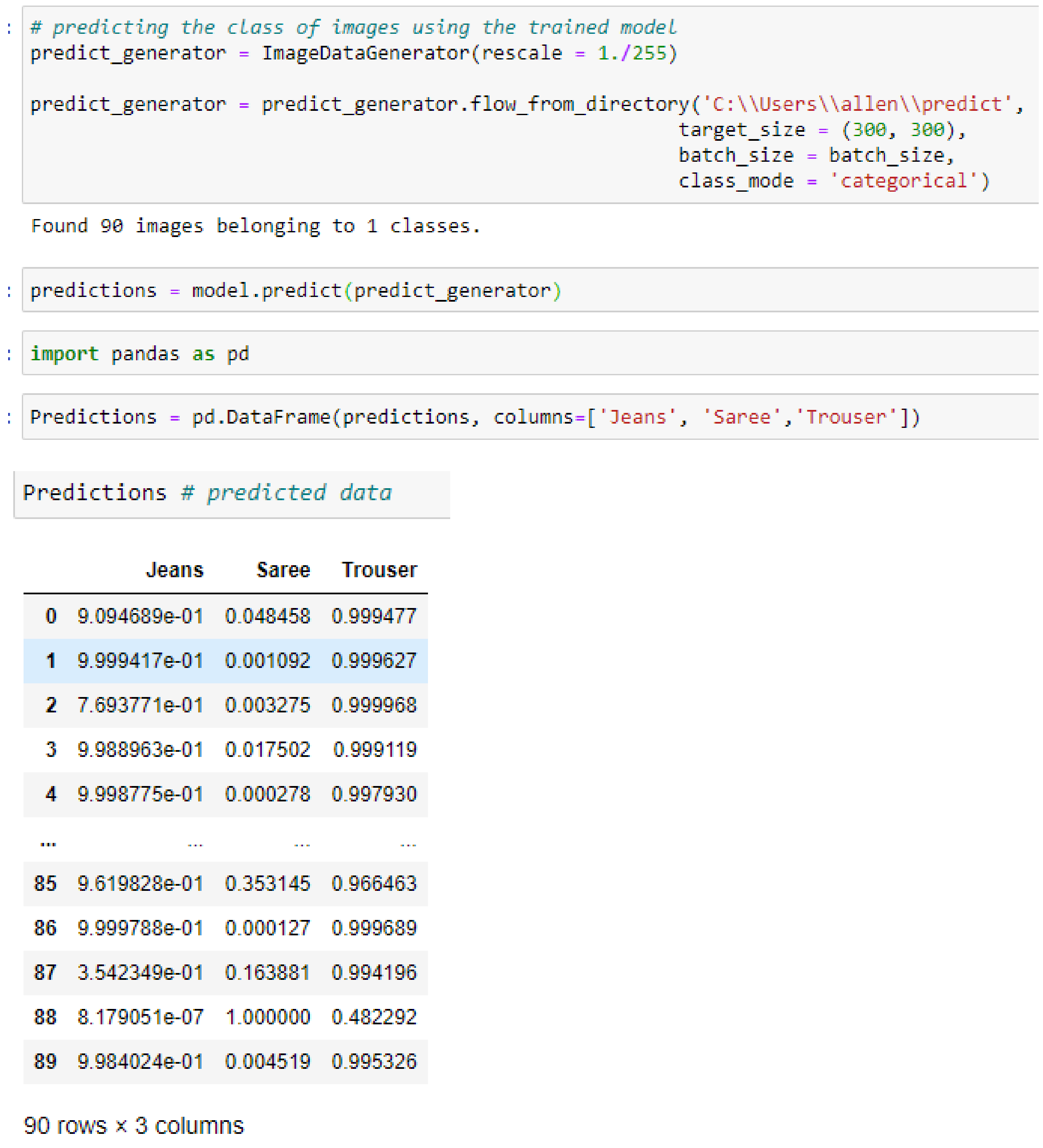
Here we can see that the model has an accuracy of 83.33% and loss of 0.34. Therefore the model is performing well



```python
model.save_weights('first_try_image.h5') # saving the model
```

Saving the model so that we can use it by loading the model and predict on different dataset.

Predicting data using the trained model:



```
# predicting the class of images using the trained model
predict_generator = ImageDataGenerator(rescale = 1./255)

predict_generator = predict_generator.flow_from_directory('C:\\Users\\allen\\predict',
                                                         target_size = (300, 300),
                                                         batch_size = batch_size,
                                                         class_mode = 'categorical')
```

```
Found 90 images belonging to 1 classes.
```

```
predictions = model.predict(predict_generator)
```

```
import pandas as pd
```

```
Predictions = pd.DataFrame(predictions, columns=['Jeans', 'Saree','Trouser'])
```

```
Predictions # predicted data
```

	Jeans	Saree	Trouser
0	9.094689e-01	0.048458	0.999477
1	9.999417e-01	0.001092	0.999627
2	7.693771e-01	0.003275	0.999968
3	9.988963e-01	0.017502	0.999119
4	9.998775e-01	0.000278	0.997930
...	...	...	...
85	9.619828e-01	0.353145	0.966463
86	9.999788e-01	0.000127	0.999689
87	3.542349e-01	0.163881	0.994196
88	8.179051e-07	1.000000	0.482292
89	9.984024e-01	0.004519	0.995326

90 rows × 3 columns

CONCLUSION

- ## Key Findings and Conclusions of the Study

  We can see that by using CNN, Image augmentation and multiple layers we can develop a good image classification model

- ## Learning Outcomes of the Study in respect of Data Science

  Here the quality of the data plays a very major role in order to make a good model. We must see to it that all the images have proper resolution, scaling and all the images should be of same size in order to make a model that predict accurately. Data pre-processing on the input/image data must be done properly.

- ## Limitations of this work and Scope for Future Work

Viewpoint Variation: In a real world, the entities within the image are aligned in different directions and when such images are fed to the system, the system predicts inaccurate values. In short, the system fails to understand that changing the alignment of the image (left, right, bottom, top) will not make it different and that is why it creates challenges in image recognition.

Scale Variation: Variations in size affect the classification of the object. The closer you view the object the bigger it looks in size and vice-versa

Deformation: Objects do not change even if they are deformed. The system learns from the perfect image and forms a perception that a particular object can be in specific shape only. We know that in the real world, shape changes and as a result, there are inaccuracies when the system encounters a deformed image of an object.

Inter-class Variation: Certain object varies within the class. They can be of different shape, size, but still represents the same class. For example, buttons, chairs, bottles, bags come in different sizes and appearances.