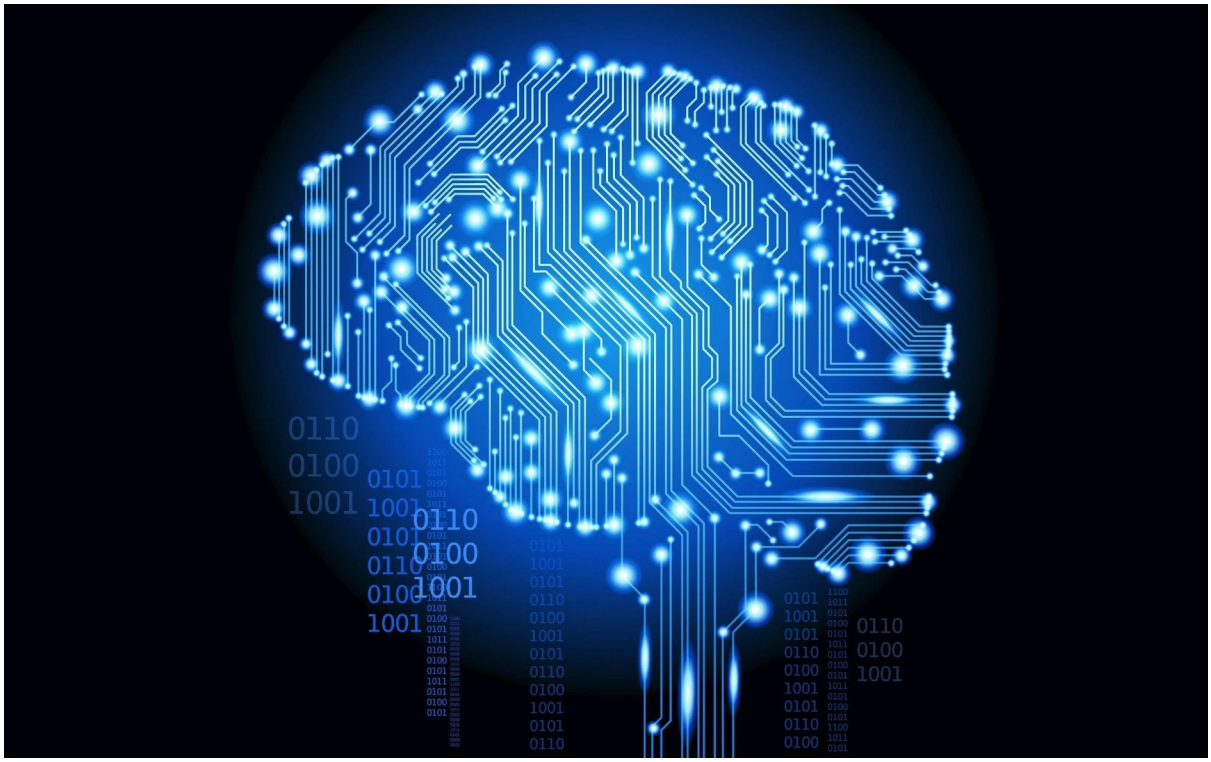


Image Scraping and Classification Project

Submitted by:

Aaron D'souza

ACKNOWLEDGMENT

References:

Paper referred:

https://www.researchgate.net/publication/325116934_Image_classification_using_Deep_learning

Link (GitHub reference):

https://github.com/girishkuniyal/Cat-Dog-CNN-Classifier/blob/master/DogVsCat%20Classifier.ipynb

Link (GitHub reference):

https://github.com/krishnaik06/Cotton-Disease-Prediction-Deep-Learning

Link (Paper):

https://www.hindawi.com/journals/mpe/2021/5843816/

INTRODUCTION

- ## Business Problem Framing

  Image classification has gained lot of attention due to its application in different computer vision tasks such as remote sensing, scene analysis, surveillance, object detection, and image retrieval.

  The primary goal of image classification is to assign the class labels to images according to the image contents.

  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.

  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(Amazon.in). This is done to make the model more and more robust.

- ## Conceptual Background of the Domain Problem

  Image classification and analysis is an active research area and there are many applications of automatic image classification in computer vision domains such as pattern recognition, image retrieval, object recognition, remote sensing, face recognition, textile image analysis, automatic disease detection, geographic mapping, and video processing.

  In any image classification-based model, the primary objective of research is to assign the class labels to images.

  A group of images are used as training samples and learning of classification-based model is done by using a training dataset.

  After training, the test dataset is assigned to the trained model to predict the class labels of images. On the basis of prediction of test dataset, images can be arranged in a semantic and meaningful order.

- ## Review of Literature

  Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data.

  The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision.

  The conventional methods used for image classifying is part and piece of the field of artificial intelligence (AI) formally called as machine learning.

  The machine learning consists of feature extraction module that extracts the important features such as edges, textures etc and a classification module that classify based on the features extracted.

  The main limitation of machine learning is, while separating, it can only extract certain set of features on images and unable to extract differentiating features from the training set of data. This disadvantage is rectified by using the deep learning. Deep learning (DL) is a sub field to the machine learning, capable of learning through its own method of computing.

  A deep learning model is introduced to persistently break down information with a homogeneous structure like how a human would make determinations. To accomplish this, deep learning utilizes a layered structure of several algorithms expressed as an artificial neural system (ANN).

- ## Motivation for the Problem Undertaken

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.

The proposed methodology aims to enhance the image classification accuracy while using CNN model keeping in view the robust performance of the model, we selected Residual Network (ResNet50) for evaluation.

ResNet50 is the short form of Residual Network with 50 layers. When researchers started to follow the phrase “the deeper the better” with deep learning models, they encountered some problems. “The deeper the network is the performance of the network should be better”; this theory was proved wrong when a deep network with 52 layers generated bad results as compared to the networks with 20–30 layers.

Multiple predictions are reported about this decrease in the performance of the model and the most appropriate reason for this is the vanishing gradients. When the network is too deep, the gradient value shrinks to 0, which causes the weights not to update, and as a result no learning is performed.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  As an aspiring Data Scientist the goal is to apply my analytical skills to give findings and conclusions in detailed data analysis written in jupyter notebook.

  The main goal of the project is to scrap image data namely three categories (Sarees (women) , Trousers (men), Jeans (men)) from Amazon.com. This data will be provided as an input to your deep learning model.

- ## Data Sources and their formats

  More than 200 images for each individual categories were scrapped from Amazon.com.

  All the images were saved in JPG (Joint Photographic Group)

  Sarees (women):

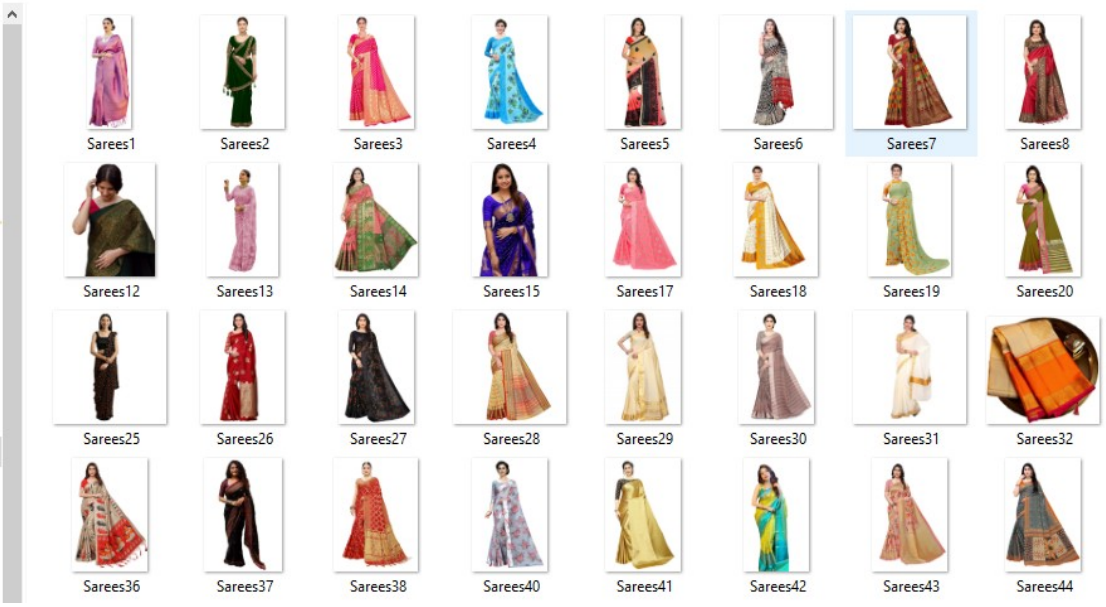


Fig.1 Sarees Images sample

Trousers (men):

Fig.2 Trousers Images sample

Jeans (men):

Fig.3 Jeans Images sample

- # Data Preprocessing Done

Step 1:

We created two folders, Dataset for Training data and Test folder for testing data. Each categories were saved in separate folders with labels as [0, 1, 2]. Label 0 for Sarees, Label 1 for Trousers and Label 2 for Jeans.

Step 2:

We need to Normalize the images and use 20 % of the data for cross validation(Testing)

```
# We need to normalize the images

image_generator = ImageDataGenerator(rescale=1./255,validation_split= 0.2)

# Using 20% of the data for cross validation
```

Step3:

We will Generate a batch of 40 images for both Training data and Validation data.

```
# Generate a batch of 40 images


train_generator = image_generator.flow_from_directory(batch_size=40,directory=SJT_directory,
                                                      shuffle=True,target_size=(256,256),
                                                      class_mode='categorical',subset='training')
```

```
Found 540 images belonging to 3 classes.
```

```
# Creating validation generator

validation_generator = image_generator.flow_from_directory(batch_size=40,directory=SJT_directory,
                                                      shuffle=True,target_size=(256,256),
                                                      class_mode='categorical',subset='validation')
```

```
Found 135 images belonging to 3 classes.
```

Step 4:

Assigning Label names to images.

```
# Label Translator

label_names = {0:"Saree",1:"Trousers",2:"Jeans"}
```

```
label_names
```

```
{0: 'Saree', 1: 'Trousers', 2: 'Jeans'}
```

- Hardware and Software Requirements and Tools Used

Hardware used:

- 🕐 OS: Windows 10 Home Single Language 64 bit
- 🕐 Ram: 8 GB
- 🕐 Processor: Intel I5

Software used:

- 🕐 Jupyter Notebook

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

  ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks.

  This model was the winner of Image-Net challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.

  Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients.

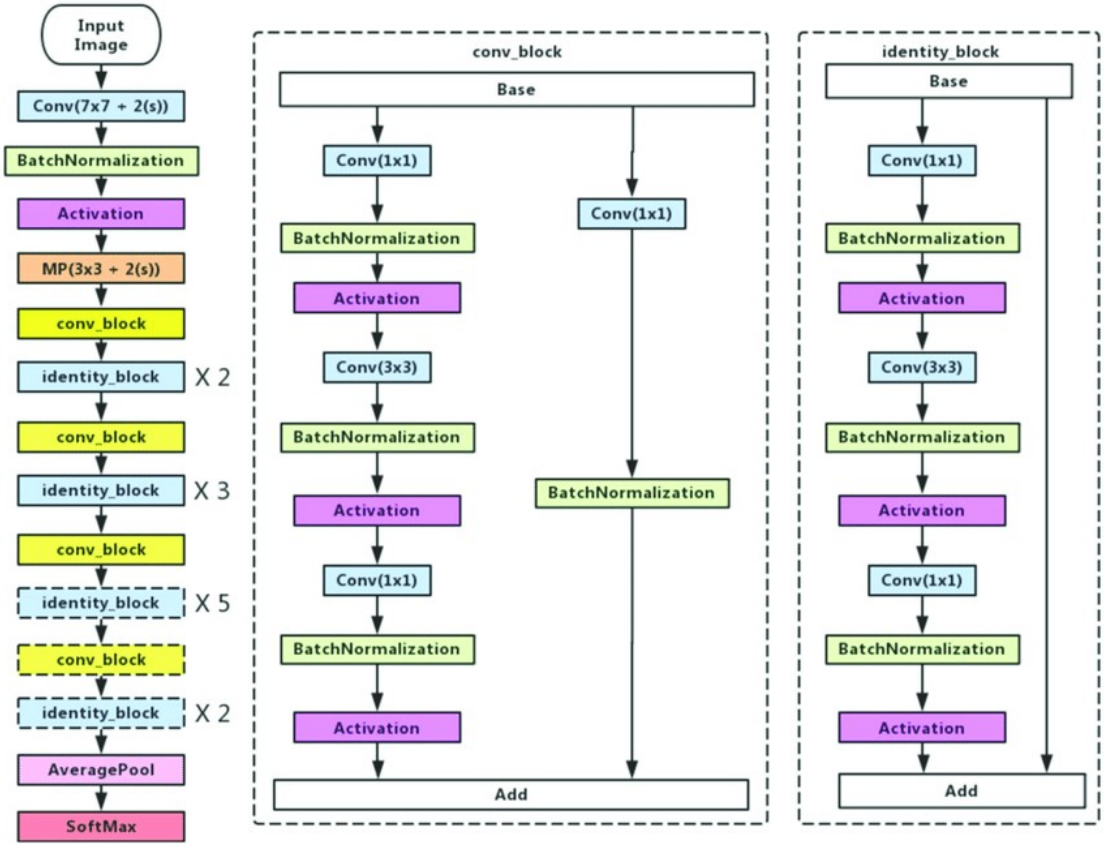


Fig.4 ResNet50 Architecture

A Convolutional Neural Network: (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

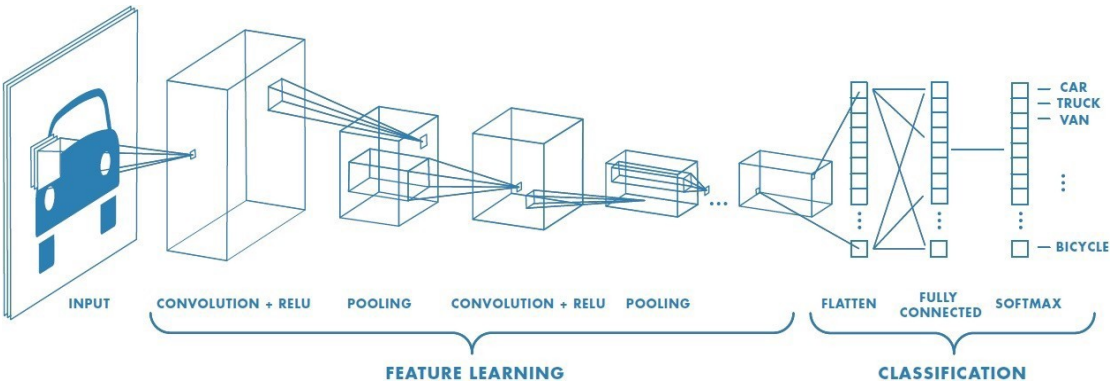


Fig.5 CNN Architecture

- # Run and Evaluate selected models

Step 1:

Creating a Base-model
Importing a model with pre trained weights. Downloading the model.

```
# defining the base model

basemodel = ResNet50(weights = 'imagenet', include_top = False,
                     input_tensor = Input(shape=(256, 256, 3)))
```

Image-Net is basically a project aimed at labelling and categorizing images into almost 22,000 categories based on a defined set of words and phrases.

We do not include top as we will create our custom input.

Step 2:

Checking Base-model summary. A total of 22 million parameters.

```
#Summary
basemodel.summary()
```

```
conv5_block3_1_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block3_1_conv[0][0]
conv5_block3_1_relu (Activation (None, 8, 8, 512)    0           conv5_block3_1_bn[0][0]
conv5_block3_2_conv (Conv2D)    (None, 8, 8, 512)    2359808     conv5_block3_1_relu[0][0]
conv5_block3_2_bn (BatchNormali (None, 8, 8, 512)    2048        conv5_block3_2_conv[0][0]
conv5_block3_2_relu (Activation (None, 8, 8, 512)    0           conv5_block3_2_bn[0][0]
conv5_block3_3_conv (Conv2D)    (None, 8, 8, 2048)   1050624     conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali (None, 8, 8, 2048)   8192        conv5_block3_3_conv[0][0]
```

Step 3:

Freezing layers as the weights should not be modified after a certain threshold.

```
# Freezing layers in model

for layer in basemodel.layers[:-10]:

    layers.trainable = False
```

Step 4:

Building and Training a deep learning Model.

Now be create a Head model for our Base modelling

```
headmodel = basemodel.output

headmodel = AveragePooling2D(pool_size=(4,4))(headmodel)

headmodel = Flatten(name = 'flatten')(headmodel)

headmodel = Dense(256, activation = 'relu') (headmodel)

headmodel = Dropout(0.3)(headmodel)

headmodel = Dense(128, activation = 'relu') (headmodel)

headmodel = Dropout(0.2)(headmodel)

headmodel = Dense(3, activation = 'softmax') (headmodel)
```

Adding an Average Pooling layer with size (4 * 4)

Average Pooling is a pooling operation that calculates the average value for patches of a feature map, and uses it to create a downsampled (pooled) feature map. It is usually used after a convolutional layer.

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector.

Dense layer is the regular deeply connected neural network layer. Using Relu activation function

Adding a Dropout layer to drop some neurons both hidden and visible in a neural network.

Adding another Dense layer

Dense layer is the regular deeply connected neural network layer. Using Softmax activation function as we need to classify three categories.

Step 5:

Connecting the Head model with the Base Model and compiling it.

```
# Building the final model

model = Model(inputs = basemodel.input, outputs = headmodel)
```

```
model.compile(loss = 'categorical_crossentropy', optimizer = optimizers.RMSprop(learning_rate = 1e
              metrics=['accuracy'])
```

Taking evaluation metrics as Accuracy

Taking loss function as categorical cross entropy mostly used for multi class classification problems.

Step 6:

Final Model Summary.

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0         
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 262, 262, 3)       0         
_________________________________________________________________
conv1_conv (Conv2D)          (None, 128, 128, 64)      9472      
_________________________________________________________________
conv1_bn (BatchNormalization (None, 128, 128, 64)      256       
_________________________________________________________________
conv1_relu (Activation)      (None, 128, 128, 64)      0         
_________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 130, 130, 64)      0         
_________________________________________________________________
pool1_pool (MaxPooling2D)    (None, 64, 64, 64)        0         
_________________________________________________________________
conv2_block1_1_conv (Conv2D) (None, 64, 64, 64)        4160      
```

Step 7:

Using Early Stopping to avoid over fitting problems.

```
# Using early stopping to avoid overfitting

earlystopping = EarlyStopping(monitor = 'val_loss',mode = 'min', patience = 20)
```

Step 8:

Using Model checkpoint to save only the best model.

```
# Only save the best model

checkpointer = ModelCheckpoint(filepath='stjweights.hdf5', verbose = 1, save_best_only = True)
```

Step 9:

Training and Testing Data.

```
# For Train

train_generator = image_generator.flow_from_directory(batch_size = 4, directory = SJT_directory,
                                                      shuffle = True, target_size = (256,256),
                                                      class_mode='categorical',subset='training')
```

```
Found 540 images belonging to 3 classes.
```

```
# For validation

val_generator = image_generator.flow_from_directory(batch_size = 4, directory = SJT_directory,
                                                    shuffle = True, target_size = (256,256),
                                                    class_mode='categorical',subset='validation')
```

```
Found 135 images belonging to 3 classes.
```

Step 10:

Training the model with 25 epochs

```
# Training the model

history = model.fit_generator(train_generator, epochs = 25, validation_data = val_generator,
                              callbacks = [checkpointer,earlystopping])
```

```
Epoch 00003: val_loss improved from 1.09050 to 1.00652, saving model to stjweights.hdf5
Epoch 4/25
135/135 [==============================] - 347s 3s/step - loss: 0.1758 - accuracy: 0.9481 - val_los
s: 0.8266 - val_accuracy: 0.5111

Epoch 00004: val_loss improved from 1.00652 to 0.82661, saving model to stjweights.hdf5
Epoch 5/25
135/135 [==============================] - 340s 3s/step - loss: 0.1309 - accuracy: 0.9722 - val_los
s: 1.4039 - val_accuracy: 0.6370

Epoch 00005: val_loss did not improve from 0.82661
Epoch 6/25
135/135 [==============================] - 341s 3s/step - loss: 0.1426 - accuracy: 0.9574 - val_los
s: 0.2059 - val_accuracy: 0.9185
```

The best model was saved at epoch number 18

Training Score 0.99%

Validation Score 0.98%

- Key Metrics for success in solving problem under consideration

  Evaluating the model on Test images

  Step 1:

  Normalizing the test images and creating a batch size of 40 images

```
test_gen = ImageDataGenerator(rescale = 1./255)
```

```
test_generator = test_gen.flow_from_directory(batch_size = 40,
                                              directory= test_directory,
                                              shuffle= True, target_size=(256,256),
                                              class_mode= 'categorical')
```

```
Found 30 images belonging to 3 classes.
```

  Step 2:

  Model Accuracy for Test images

```
evaluate = model.evaluate_generator(test_generator, steps = test_generator.n // 3, verbose = 1)
```

```
 1/10 [==>.....................] - ETA: 27s - loss: 0.2834 - accuracy: 0.9667WARNING:tensorflo
w:Your input ran out of data; interrupting training. Make sure that your dataset or generator can gen
erate at least `steps_per_epoch * epochs` batches (in this case, 10 batches). You may need to use the
repeat() function when building your dataset.
10/10 [==============================] - 4s 148ms/step - loss: 0.2834 - accuracy: 0.9667
```

```
print('Accuracy Test : {}'.format(evaluate[1]))
```

```
Accuracy Test : 0.9666666388511658
```

Step 3:

Actual Images Vs Prediction

Step 4:

Classification Report

```
1 print(classification_report(np.asarray(original), np.asarray(prediction)))
```

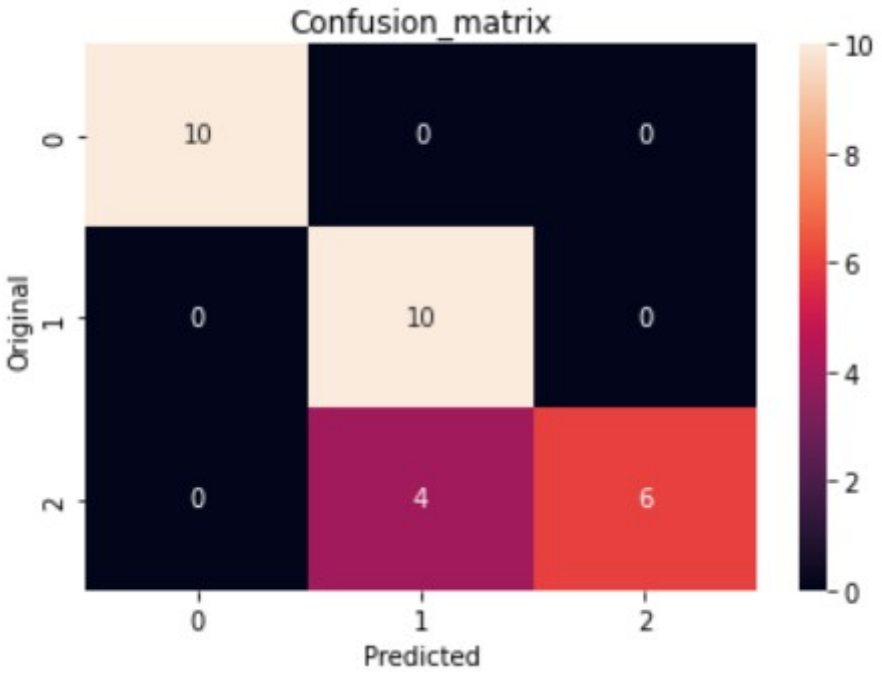
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.71      1.00      0.83        10
           2       1.00      0.60      0.75        10

    accuracy                           0.87        30
   macro avg       0.90      0.87      0.86        30
weighted avg       0.90      0.87      0.86        30
```

Step 5:

Confusion Matrix



- Visualizations

Fig.6 Sample Images

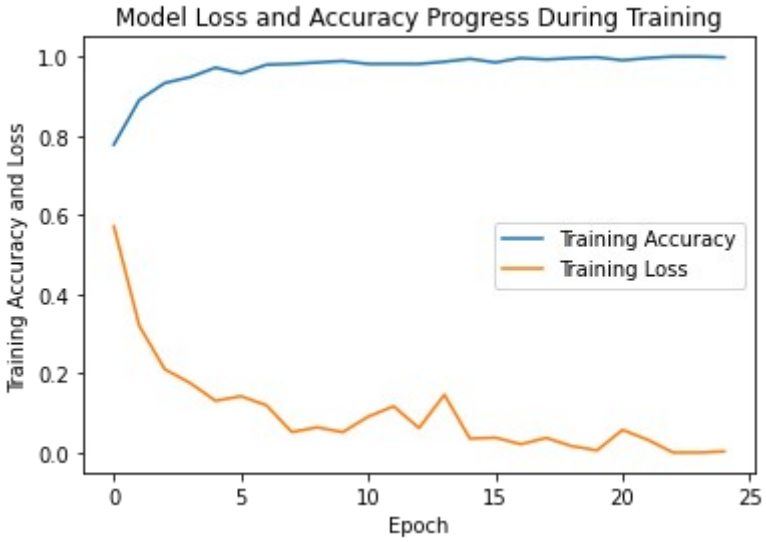


Fig.7 Epoch Vs Accuracy

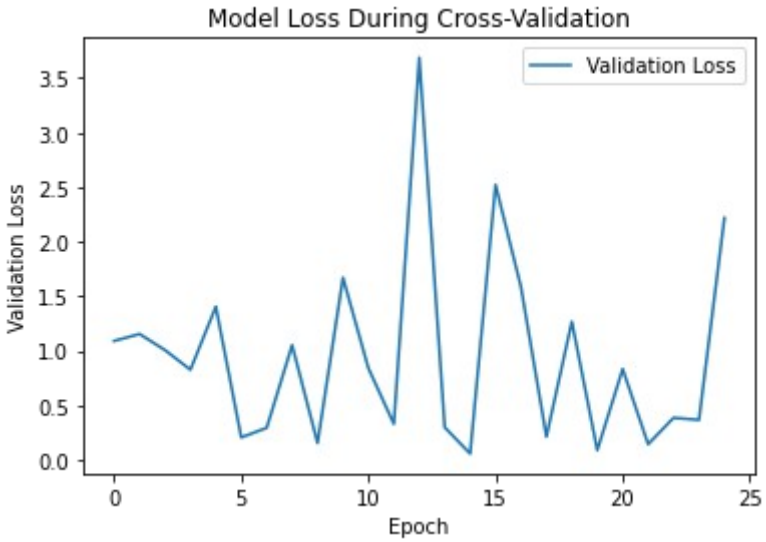


Fig.8 Epoch Vs Validation Loss

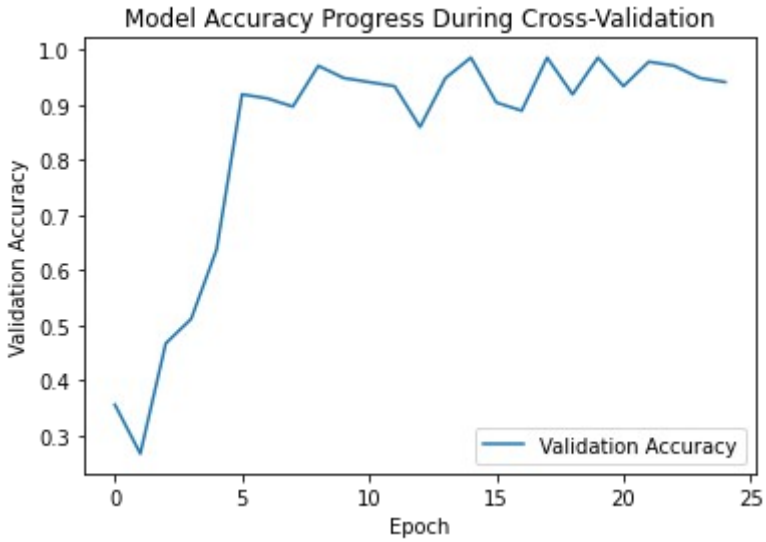


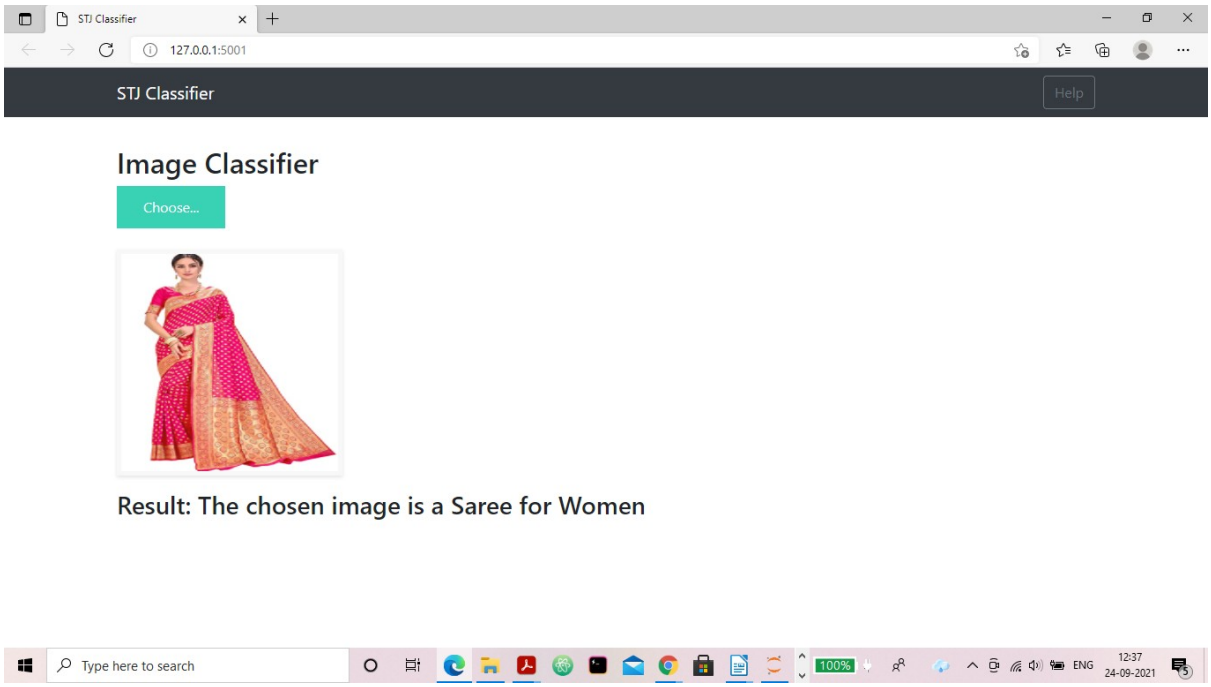
Fig.8 Epoch Vs Validation Accuracy

IMPLEMENTATION

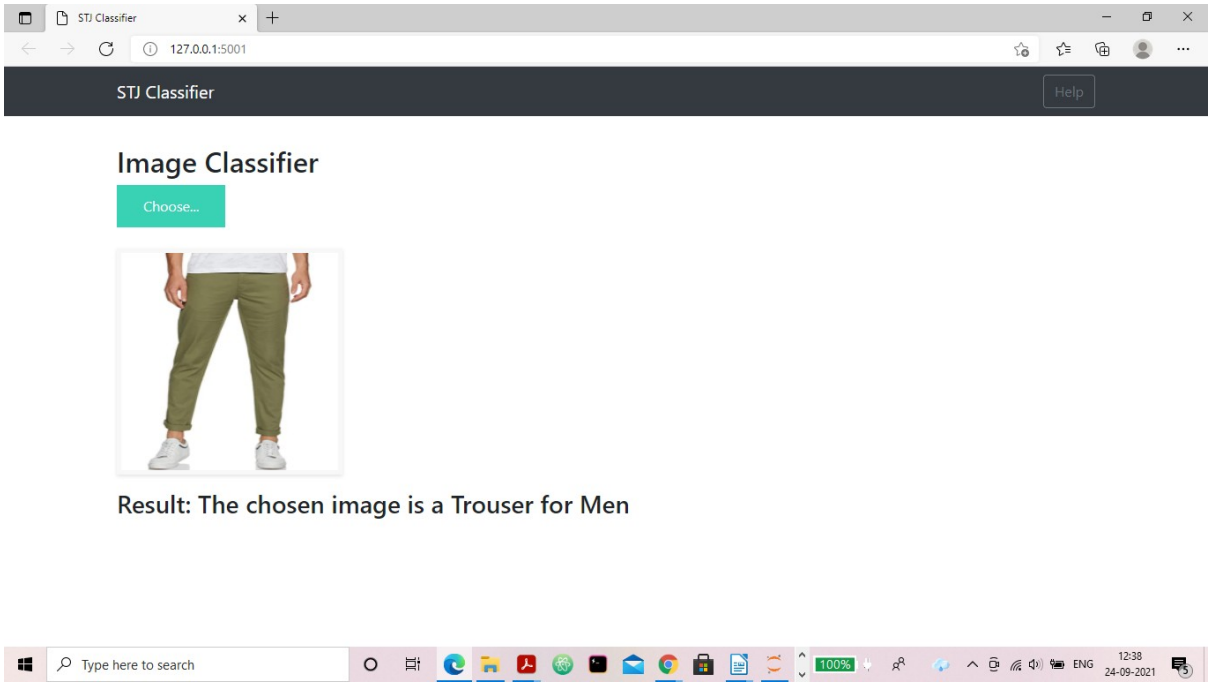
We Have implemented this project using Flask Framework.

App is named as STJ classifier

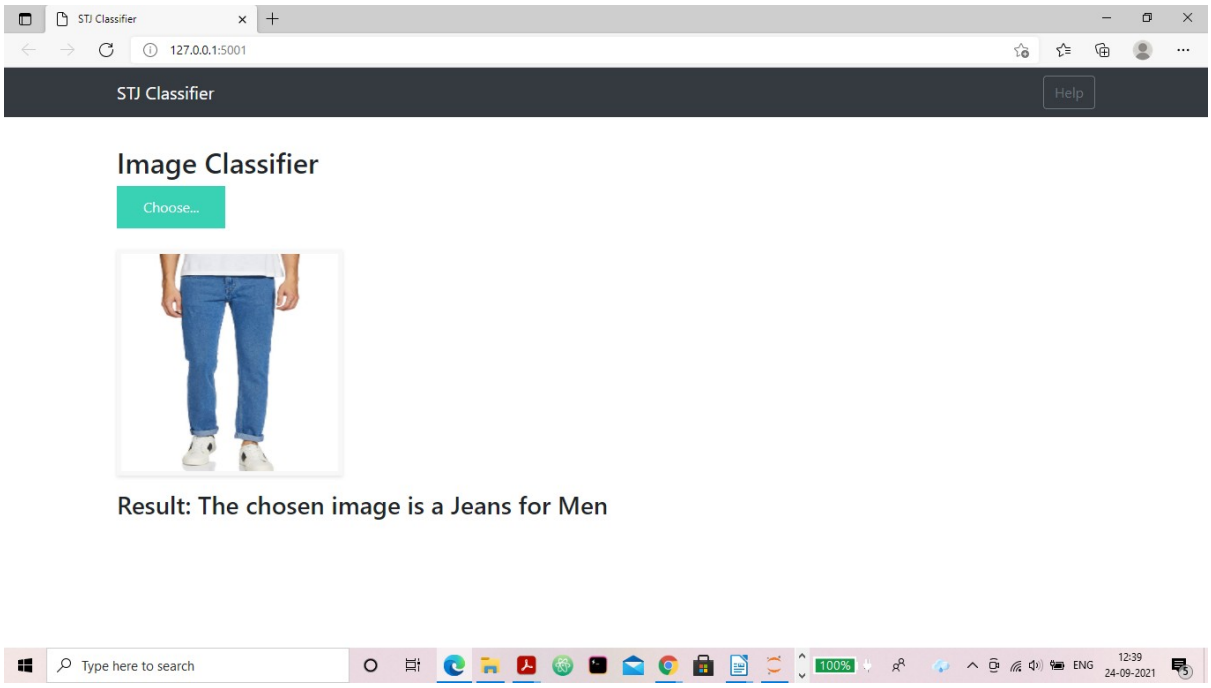
Testing Flask App for Saree (Women)



Testing Flask App for Trousers (Men)



Testing Flask App for Jeans (Men)



CONCLUSION

- ## Key Findings and Conclusions of the Study

  Thirty test images including 10 for Sarees (Women),10 images for Trousers (Men) and 10 images for Jeans (Men) were chosen for testing and validation of image classification using deep learning. The convolutional neural network is used along with REsNet50 CNN with 50 layers and over 22 million pre trained images for classification purpose. From the experiments, it is observed that the images are classified correctly even for the portion of the test images and shows the effectiveness of deep learning algorithm.