EMOTION DETECTION IN IMAGES USING CONVOLUTIONAL NEURAL NETWORKS

A Project Report

for

Machine Learning (CSE4020)

B. Tech

By

LAVANYA MIDDHA - 18BCE0845

VARTIKA TRIVEDI - 18BCE0962

Winter 2020-2021

Under the Guidance of

Prof Sharmila Banu K.

Associate Professor

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering [SCOPE]

June, 2021

TABLE OF CONTENTS

INTRODUCTION

Emotion recognition is the process of identifying human emotion. Use of technology to help people with emotion recognition is a relatively new research area. Facial expression for emotion detection has always been an easy task for humans, but achieving the same task with a computer algorithm is quite challenging. With the recent advancement in computer vision and machine learning, it is possible to detect emotions from images. Computer vision is the field of study that helps computers to study using different techniques and methods so that it can capture what exists in an image or a video.

This project aims to classify the emotion on a person's face into one of three categories, using deep convolutional neural networks. The model is trained on a dataset compiled and processed by us. This dataset consists of 75, 800x600 sized face images with three emotions –**'Happy'** , **'Sad'** and **'Surprised'**. We have built a computer vision model using Keras. We use this model to check the emotions in real-time using OpenCV. We have also applied a concept of Transfer Learning to detect human faces in an image. To identify human faces, we have used the Haar Cascade Frontal Face Model which is an open source pertained model.

Haar Cascade is an algorithm proposed by Paul Viola and Michael Jones that is capable of detecting objects in images, regardless of their location and scale in an image. Furthermore, this algorithm can run in real-time, making it possible to detect objects in video streams. The model was developed to detect faces in particular but can also be used to detect other objects in an image.

LITERATURE SURVEY

1. vjgpt/Face-and-Emotion-Recognition

They have developed a model that recognizes person's faces and their corresponding emotions from a video or webcam feed. The model is made with the help of OpenCV, Dlib, face_recognition and Deep Learning. Convolutional Networks have been used to train the model. They have used Keras and TensorFlow to train the model using CNN. For the purpose of detecting faces, they have used the Forntal Face Detector provided by Dlib library in python.

2. SanjayMarreddi/Emotion-Investigator

They have developed a Facial Emotion Detection Model using Convolutional Neural Networks, python and Flask. The CNN consists of 4 Convolutional Layers including Batch Normalization, Activation, Max Pooling, Dropout Layers followed by Flatten Layer, 2 Fully Connected dense Layers and finally Dense Layer with SoftMax Activation Function. Model has been compiled using Adam Optimizer and categorical cross entropy loss function. The model has been trained for 15 epochs. The model is finally deployed on web using Flask. They have used Haar Cascade Frontal Face Default Model to render bounding boxes and detect the presence of face in an image.

3. BishalLakha/Facial-Expression-Recognition-with-Keras

They have developed a Facial Expression Recognizer and classified 6 emotions – Angry, Fear, Sad, Happy, Neutral, Surprised. The image dat used is from Kaggle (Facial Expression Recognition Challenge) and Karolinska Directed Emotional Faces datasets. The architectures they employed for their convolutional neural networks were VGG-16 and ResNet50. They used the support vector machine multiclass classifier as their baseline, which had an accuracy performance of 31.8%. To further improve their results, they used ensemble and transfer learning techniques to achieve best results. Thus, the accuracy using ensemble learning was 67.2% and with transfer learning was 78.3%

4. magical2world/Facial-expression-recognition-with-VGG

They have developed a Facial Expression Recognition System with VGG. The facial expression dataset we used is fer2013 which was released as a part of the Facial Expression Recognition Challenge. The VGG model has been trained using TensorFlow.

5. ptbailey/Emotion-Recognition

They have developed a model to Classify a person's emotion, from a list of 7 emotions – anger, disgust, fear, happy, sad, surprise and neutral using their facial expression. The downloaded dataset consisted of face images that came as a csv file with grayscale images converted into binary. They have used a Haar Cascade Frontal Face code to detect faces in live video feed. The output only prints the emotion identified. It does not produce any bounding boxes to highlight the face it detects.

6. oarriaga/paz

They have developed a model using PAZ. It stands for Perception for Autonomous Systems. PAZ is a hierarchical perception library that allow users to manipulate multiple levels of abstraction in accordance to their requirements or skill level. More specifically, PAZ is divided into three hierarchical levels which we refer to as pipelines, processors, and backends. These abstractions allows users to compose functions in a hierarchical modular scheme that can be applied for reprocessing, data-augmentation, prediction and postprocessing of inputs and outputs of machine learning (ML) models. It is used in object detection in images.

GAP IDENTIFIED

The Most commonly used pre - trained model in the projects is Haar Cascade. However, it has some disadvantages. As Haar Features have to be determined manually, there is a certain limit to the types of things it can detect. It works best if object have clear edges and lines. Even as a face detector, if face is manipulated a little (For eg: cover up the eyes with sunglasses, or tilt the head to a side), a Haar-based classifier may not be able to recognize the face. Hence, a model trained with the help of convolutional neural networks has better results provided that model is trained using appropriate data and CNN architecture. The most important factor that always remains a challenge is the availability of a large dataset of images with multiple variations.

DATA SET

Vartika:

https://www.kaggle.com/deadskull7/fer2013

Lavanya:

I Collected my own dataset. The Dataset comprised of a total of 75 images corresponding to 3 emotions –**'Happy', 'Sad', 'Surprised'.**

Training Set consists of 66 images (i.e. 23 for each emotion – Happy, Sad and Surprised)

Testing Set consists of 9 images (i.e. 3 for each emotion – Happy, Sad, Surprised)

Dataset Processing:

Step 1:

Create a Project folder in the Documents folder. For our Project, I have created a folder named **ML.**

Step 2:

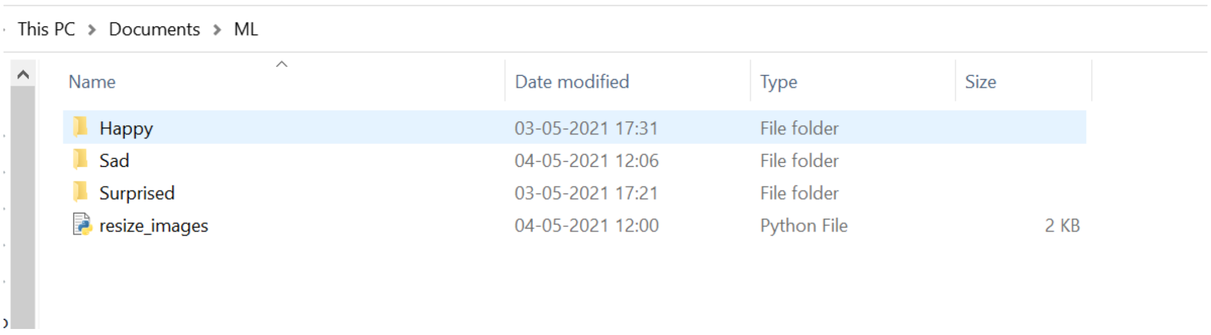
Inside the ML folder, Create folders for each emotion that you want to detect.

For our project, I have taken the following 3 emotions:

Happy

Sad

Surprised:

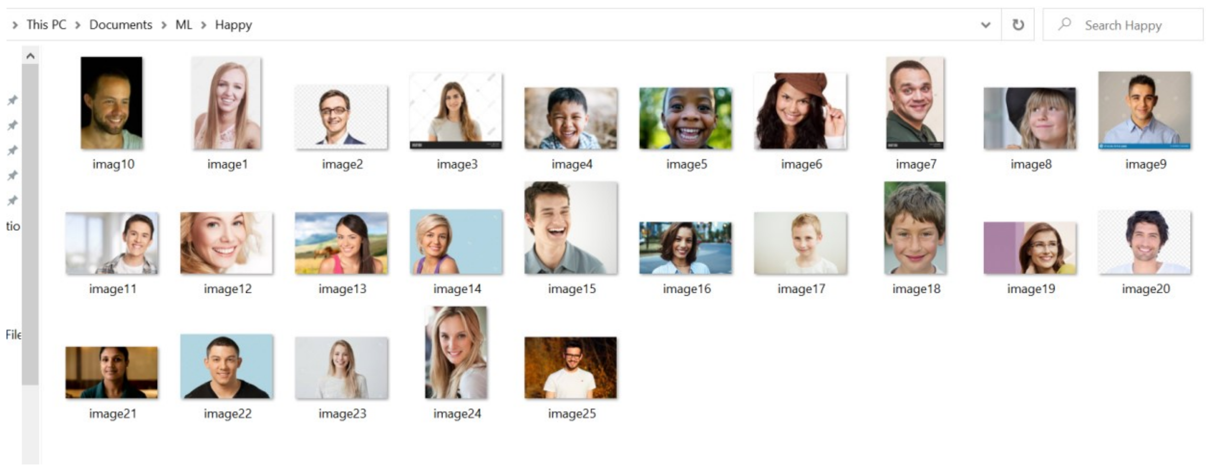


This PC > Documents > ML

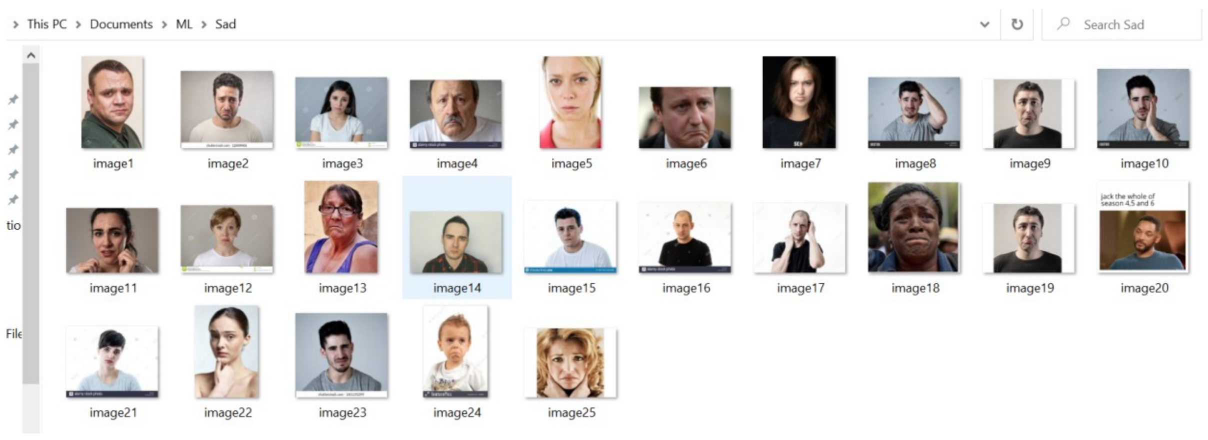
Name	Date modified	Type	Size
Happy	03-05-2021 17:31	File folder	
Sad	04-05-2021 12:06	File folder	
Surprised	03-05-2021 17:21	File folder	
resize_images	04-05-2021 12:00	Python File	2 KB

Step 3:

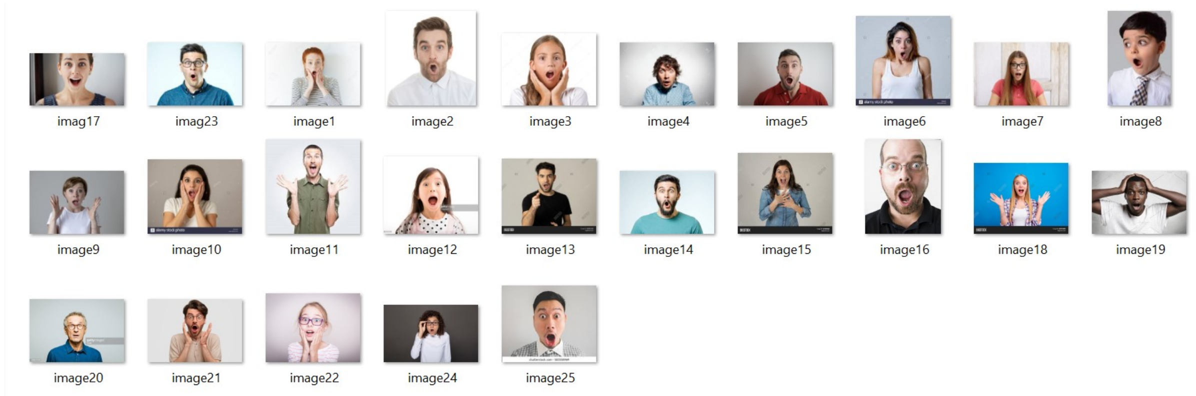
Inside each of these folder, collect images for your dataset. I have collected 25 images per category. 3 images would be kept for testing per category and 22 images for training.



This PC > Documents > ML > Happy
Search Happy
imag10 image1 image2 image3 image4 image5 image6 image7 image8 image9
image11 image12 image13 image14 image15 image16 image17 image18 image19 image20
image21 image22 image23 image24 image25



This PC > Documents > ML > Sad
Search Sad
image1 image2 image3 image4 image5 image6 image7 image8 image9 image10
image11 image12 image13 image14 image15 image16 image17 image18 image19 image20
image21 image22 image23 image24 image25



imag17 imag23 image1 image2 image3 image4 image5 image6 image7 image8
image9 image10 image11 image12 image13 image14 image15 image16 image18 image19
image20 image21 image22 image24 image25

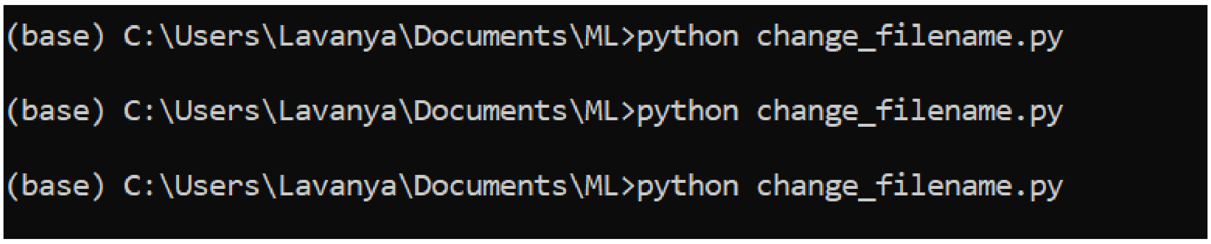
Step 4:

I am going to rename the pictures so that I can place them in the same folder. For this, I will run the following code. I will save this code in a file **Change_filename.py** in the **ML** folder.

```
import os
path = 'Surprised'                    #change file name
files = os.listdir(path)
i = 1                                 #change value of i
```

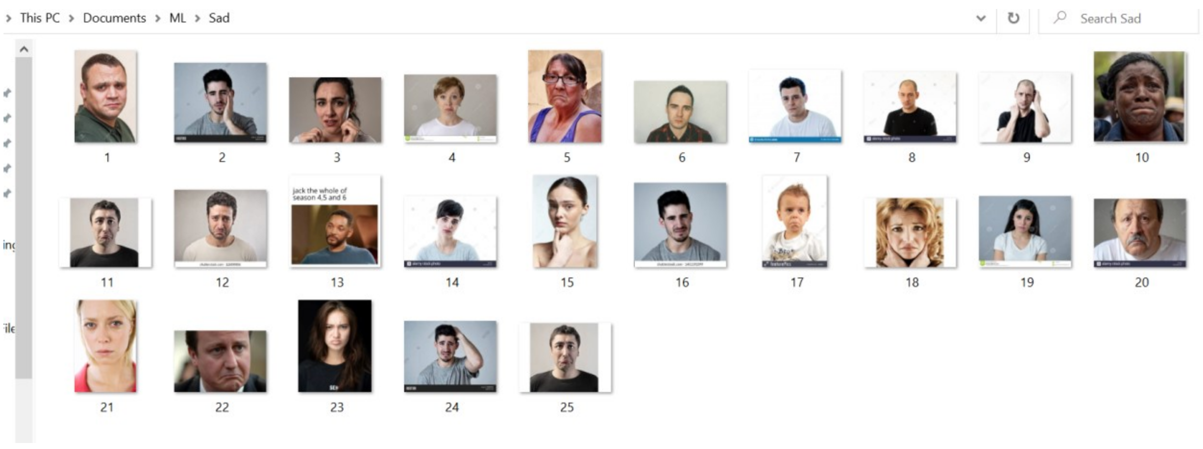
```
for file in files:
    os.rename(os.path.join(path, file), os.path.join(path, str(i)+'.jpg'))
    i = i+1
```

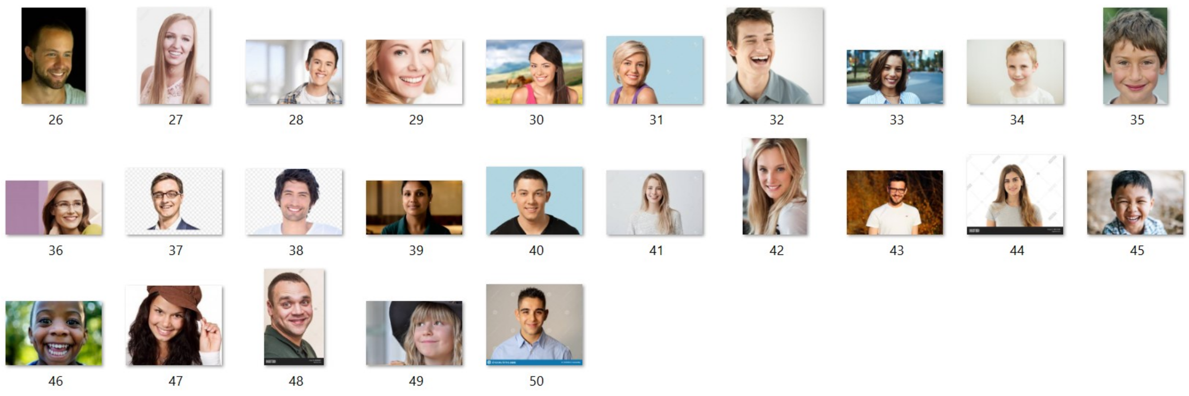
Note: Change the folder name each time you run the file and also the value of i

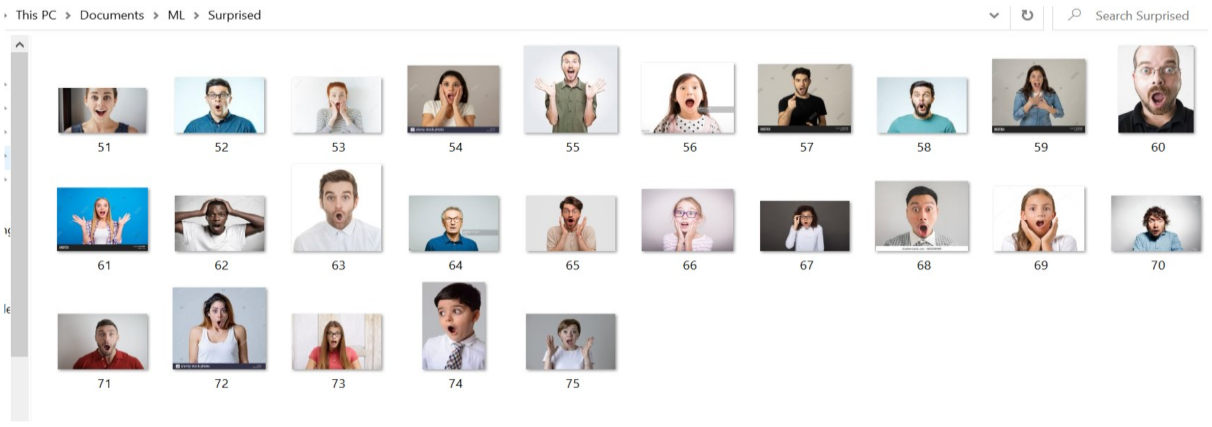


```
(base) C:\Users\Lavanya\Documents\ML>python change_filename.py

(base) C:\Users\Lavanya\Documents\ML>python change_filename.py

(base) C:\Users\Lavanya\Documents\ML>python change_filename.py
```

After running the above commands, the folder will look like this.

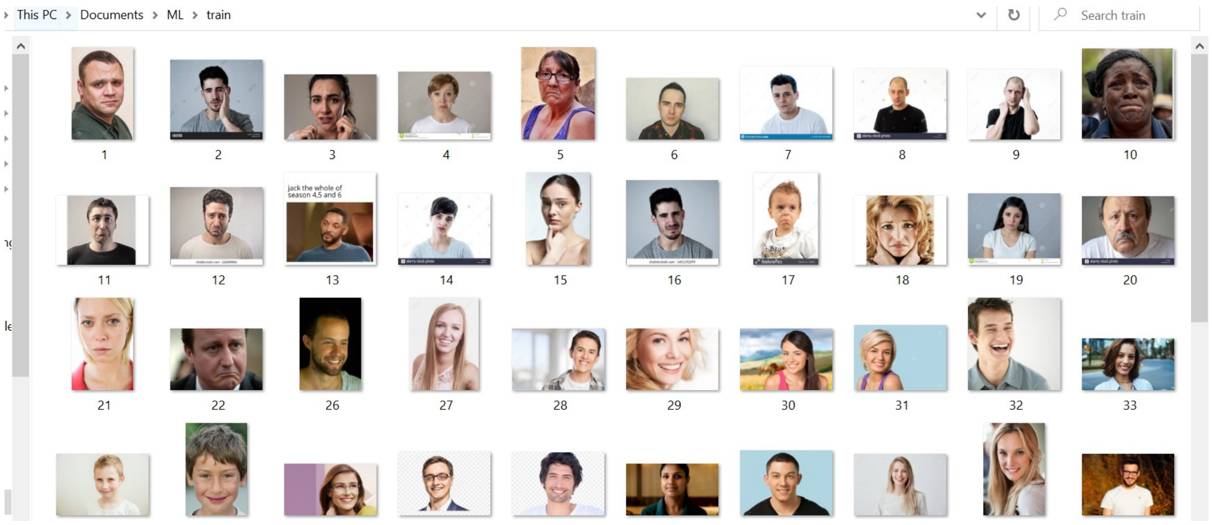


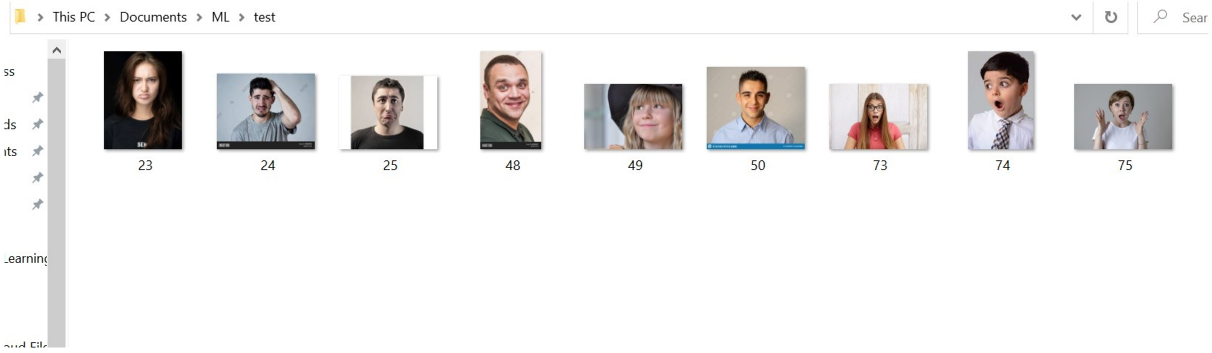




Step 5:

I am going to create two folders, train and test. I will place 22 images from each category in the train folder and 3 images from each category in the test folder.





<u>Step 6:</u>

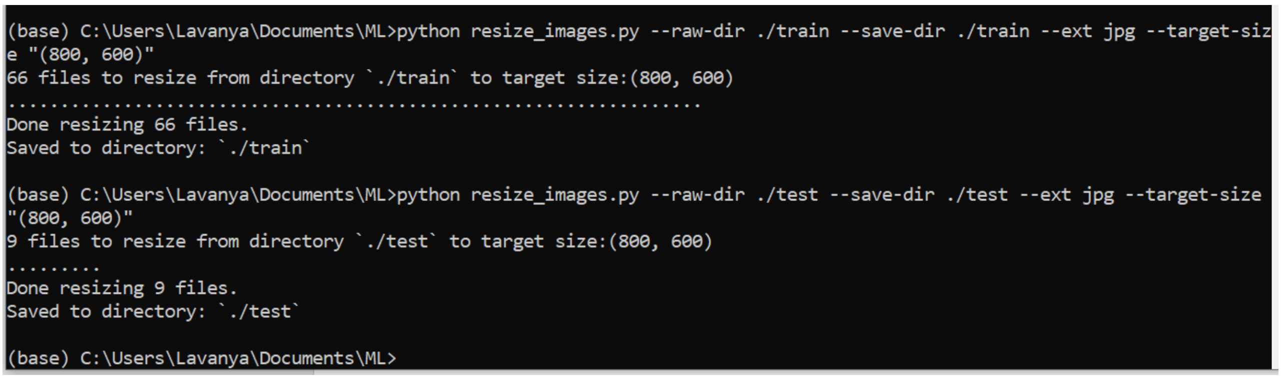
Resize all images to a standard image size.

To standardize all images, download the file **resize_images.py**

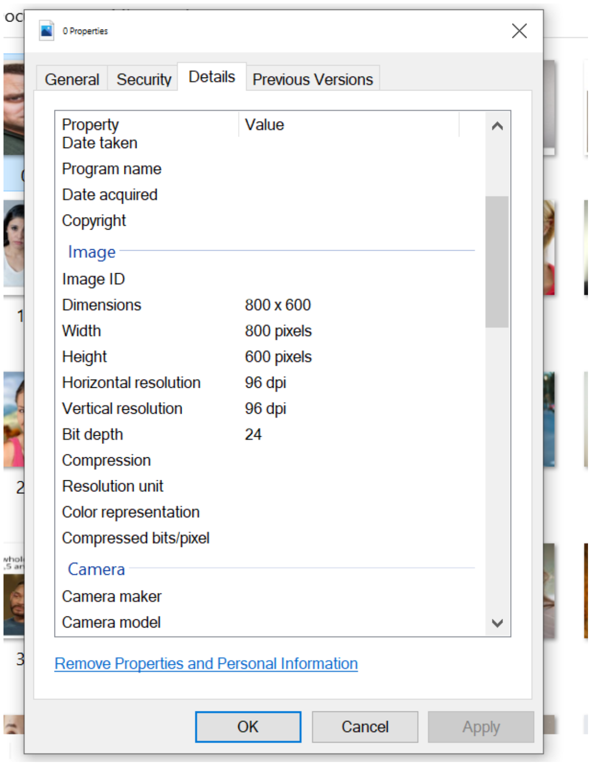
Go to the directory of your project folder and run the following command:

python resize_images.py --raw-dir ./train --save-dir ./train --ext jpg --target-size "(800, 600)"

python resize_images.py --raw-dir ./test --save-dir ./test --ext jpg --target-size "(800, 600)"



```
(base) C:\Users\Lavanya\Documents\ML>python resize_images.py --raw-dir ./train --save-dir ./train --ext jpg --target-siz
e "(800, 600)"
66 files to resize from directory `./train` to target size:(800, 600)
..................................................................
Done resizing 66 files.
Saved to directory: `./train`

(base) C:\Users\Lavanya\Documents\ML>python resize_images.py --raw-dir ./test --save-dir ./test --ext jpg --target-size
"(800, 600)"
9 files to resize from directory `./test` to target size:(800, 600)
.........
Done resizing 9 files.
Saved to directory: `./test`

(base) C:\Users\Lavanya\Documents\ML>
```



Property	Value
Date taken	
Program name	
Date acquired	
Copyright	
Image	
Image ID	
Dimensions	800 x 600
Width	800 pixels
Height	600 pixels
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	24
Compression	
Resolution unit	
Color representation	
Compressed bits/pixel	
Camera	
Camera maker	
Camera model	

Here I can see that all the files have been resized to a standard size of 800 x 600 pixels.

Step 7:

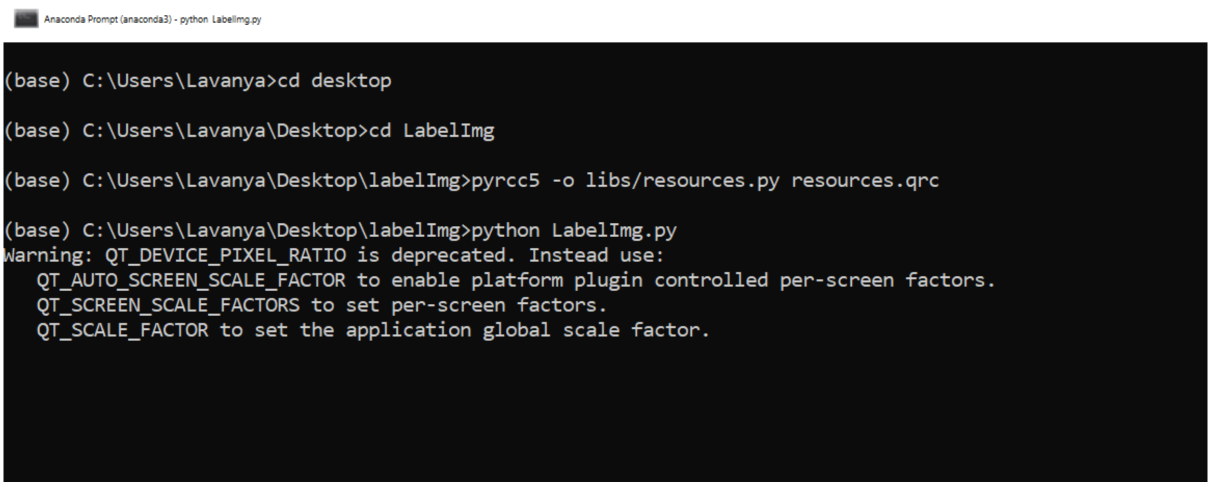
The next step would be to annotate the images. For this I will be using a tool called LabelImg. In this step I will set up the labelImg tool.

To use this python based tool, I have to follow the following steps:

a. Download the folder from the following GitHub link:
   https://github.com/tzutalin/labelImg

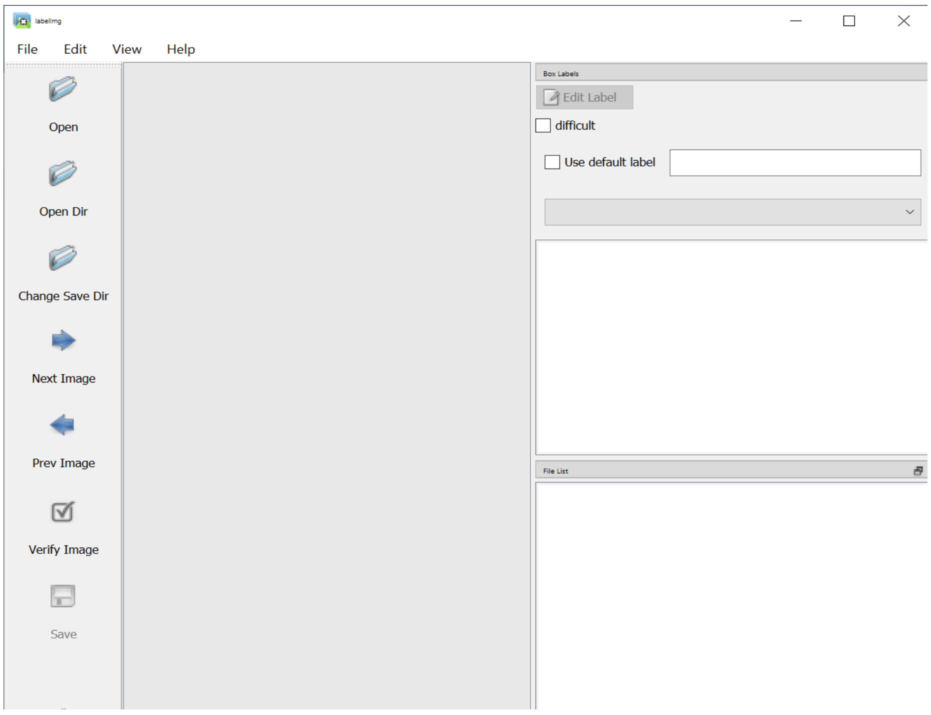
b. Extract the folder to desktop and rename it as LabelImg
   then go to anaconda and install the following:
   conda isntall pyqt=5
   conda install -c anaconda lxml

c. Then on the command prompt, Navigate to the folder LabelImg and execute the following:
   pyrcc5 -o libs/resources.py resources.qrc
   python LabelImg.py



```
(base) C:\Users\Lavanya>cd desktop

(base) C:\Users\Lavanya\Desktop>cd LabelImg

(base) C:\Users\Lavanya\Desktop\labelImg>pyrcc5 -o libs/resources.py resources.qrc

(base) C:\Users\Lavanya\Desktop\labelImg>python LabelImg.py
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen factors.
   QT_SCALE_FACTOR to set the application global scale factor.
```

d. On running the above commands, the tool window will open as follows:



Step 8:

Now I will start annotating the images.

In the tool, I will click on the option **open Dir** and then select the folder in which I have to annotate images.

I have to do this for both, Test and Train folder.

The most important thing is to keep the label class names same.

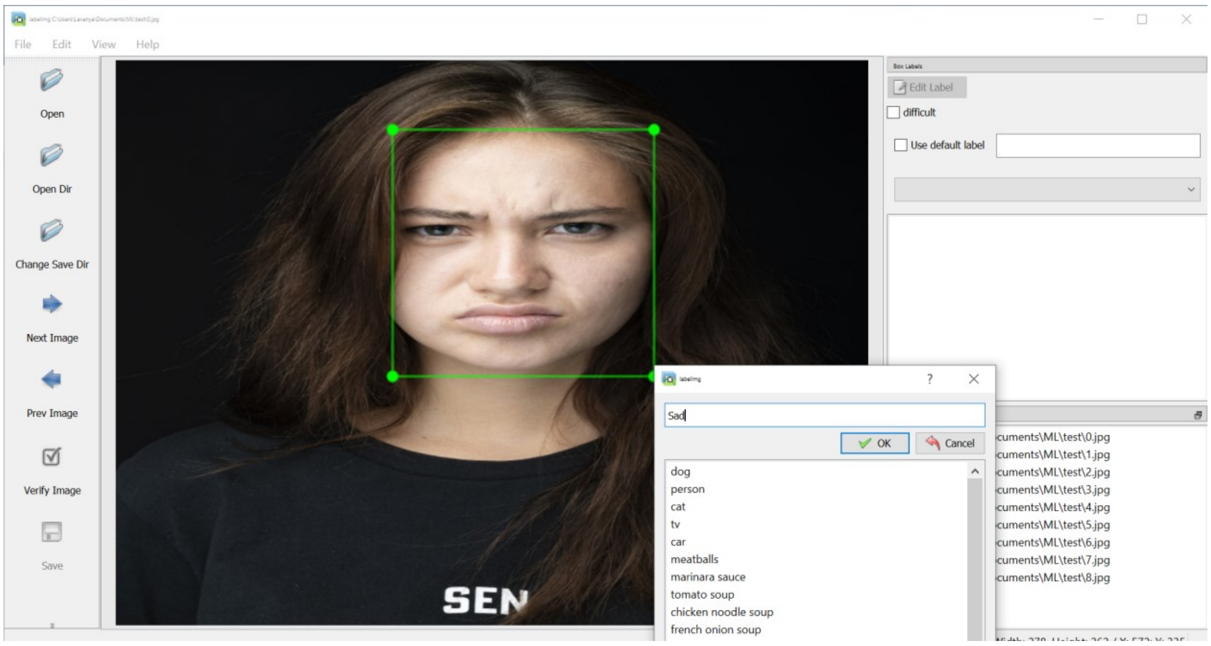
For out project, I would be having 3 labels:

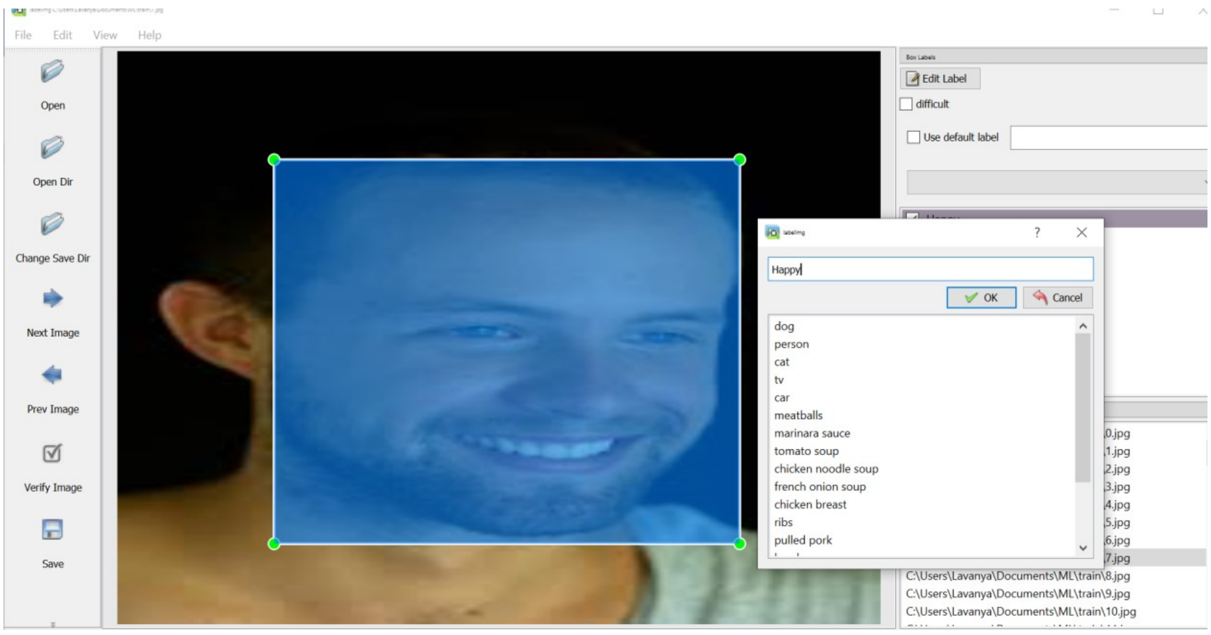
Happy

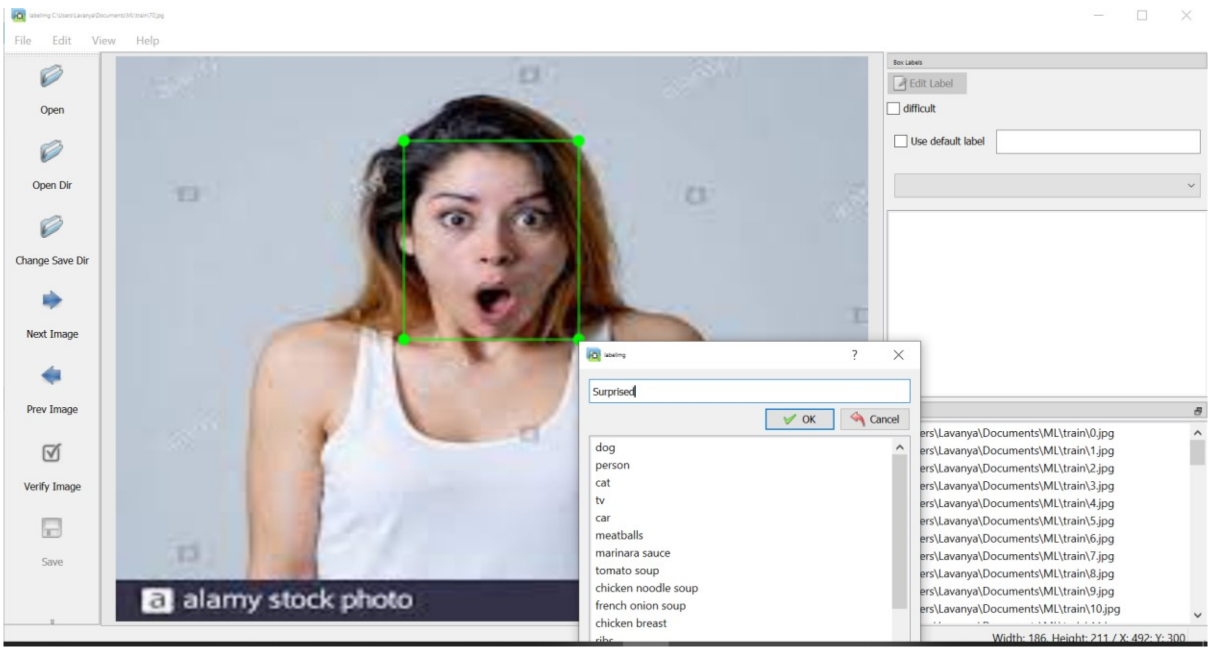
Sad

Serious

Note: These labels are case sensitive, so one should be very careful with labelling the images.


File
Edit
View
Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
Box Labels
Edit Label
difficult
Use default label
labelImg
Sad
OK
Cancel
dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
chicken noodle soup
french onion soup
cuments\ML\test\0.jpg
cuments\ML\test\1.jpg
cuments\ML\test\2.jpg
cuments\ML\test\3.jpg
cuments\ML\test\4.jpg
cuments\ML\test\5.jpg
cuments\ML\test\6.jpg
cuments\ML\test\7.jpg
cuments\ML\test\8.jpg


File
Edit
View
Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
Box Labels
Edit Label
difficult
Use default label
labelImg
Happy
OK
Cancel
dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
chicken noodle soup
french onion soup
chicken breast
ribs
pulled pork
C:\Users\Lavanya\Documents\ML\train\8.jpg
C:\Users\Lavanya\Documents\ML\train\9.jpg
C:\Users\Lavanya\Documents\ML\train\10.jpg



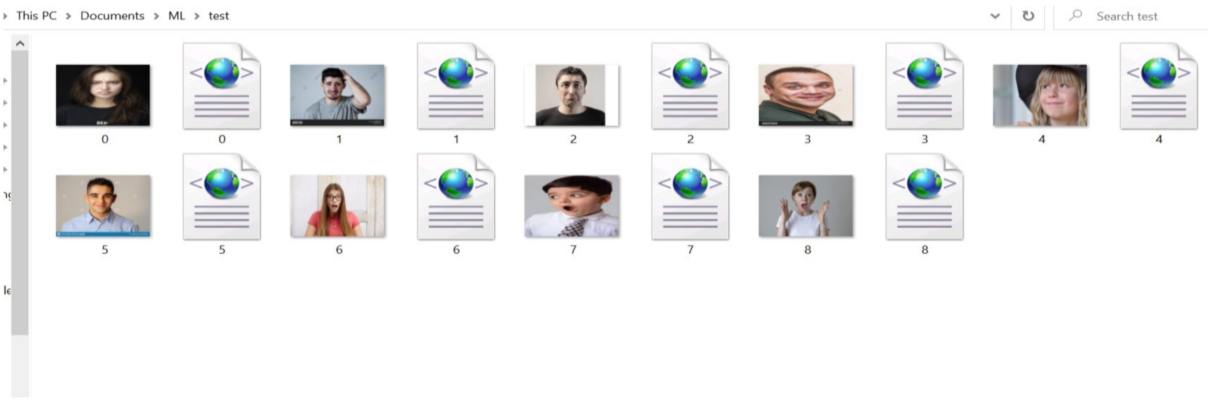
Surprised

dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
chicken noodle soup
french onion soup
chicken breast

ers\Lavanya\Documents\ML\train\0.jpg
ers\Lavanya\Documents\ML\train\1.jpg
ers\Lavanya\Documents\ML\train\2.jpg
ers\Lavanya\Documents\ML\train\3.jpg
ers\Lavanya\Documents\ML\train\4.jpg
ers\Lavanya\Documents\ML\train\5.jpg
ers\Lavanya\Documents\ML\train\6.jpg
ers\Lavanya\Documents\ML\train\7.jpg
ers\Lavanya\Documents\ML\train\8.jpg
ers\Lavanya\Documents\ML\train\9.jpg
ers\Lavanya\Documents\ML\train\10.jpg

Width: 186, Height: 211 / X: 492; Y: 300

The folder after annotation will consist of xml files as follows:



<u>Step 9:</u>

When I annotate the images, the information of the same would get stored in a .xml file for each image.

The next step would be to compile these xml files and make csv file from this for both train and test. To do so I have to run the file **xml_to_csv.py**

To do this, navigate to the project folder and run the following command

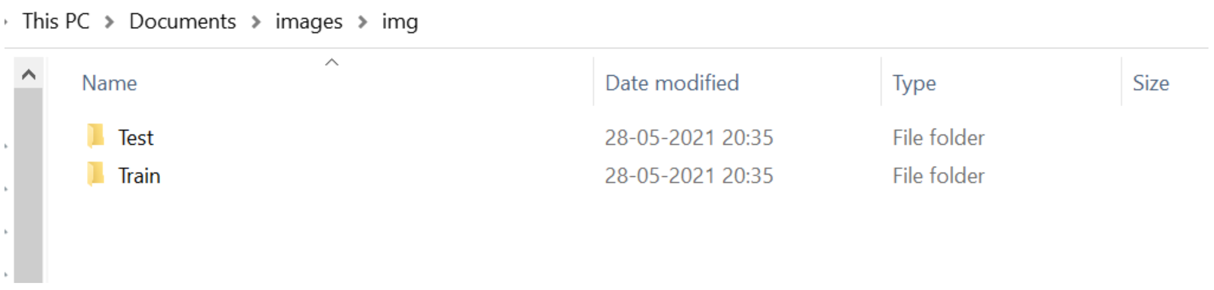
# Create train data:

python xml_to_csv.py -i train -o train/train_labels.csv

# Create test data:

python xml_to_csv.py -i test -o test/test_labels.csv

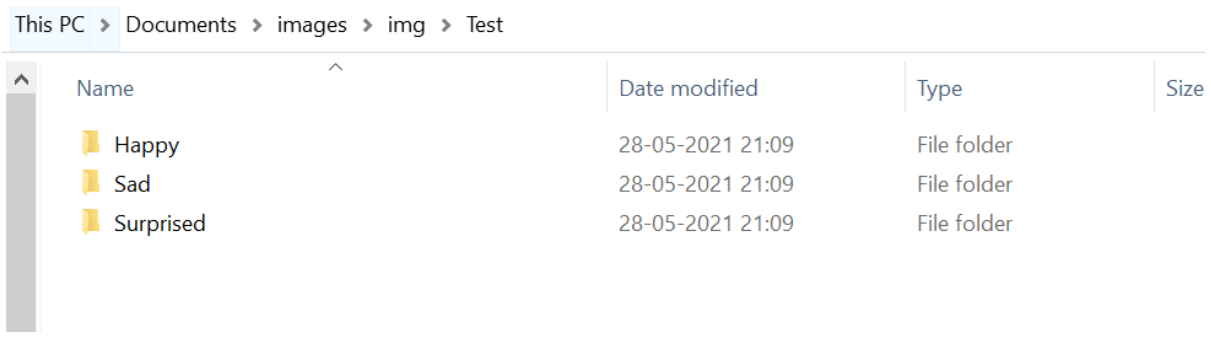
Create a New folder **Images.** Paste the Train and test Folders here. Create another folder **Img** inside the Images folder. The folder should have the following two empty folders.



This PC > Documents > images > img

Name	Date modified	Type	Size
Test	28-05-2021 20:35	File folder	
Train	28-05-2021 20:35	File folder	

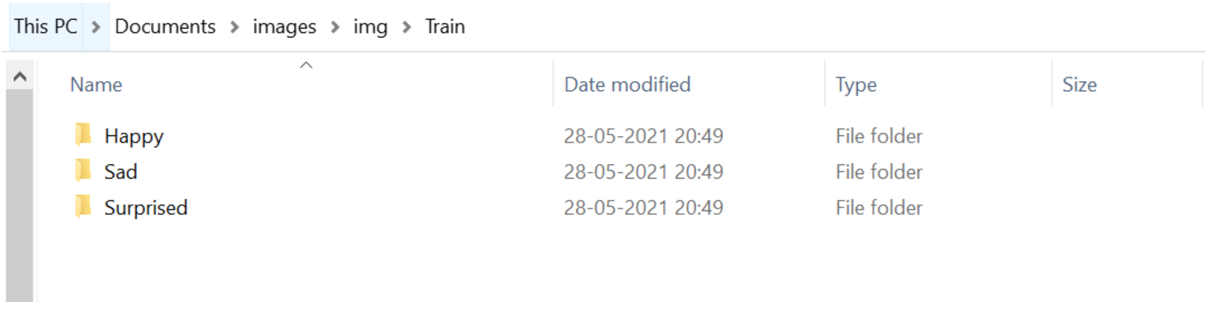
The **Test** folder inside the **Img** folder should have the following 3 empty folders.



This PC > Documents > images > img > Test

Name	Date modified	Type	Size
Happy	28-05-2021 21:09	File folder	
Sad	28-05-2021 21:09	File folder	
Surprised	28-05-2021 21:09	File folder	

Similarly, the **Train** folder inside the **Img** folder should have the following 3 empty folders.



This PC > Documents > images > img > Train

Name	Date modified	Type	Size
Happy	28-05-2021 20:49	File folder	
Sad	28-05-2021 20:49	File folder	
Surprised	28-05-2021 20:49	File folder	

Now I would be storing the cropped images in these folder with the help of the code I have written and the annotations CSV files.

The link for the code can be found in our GitHub repository. Name of the file is ***Code.py***

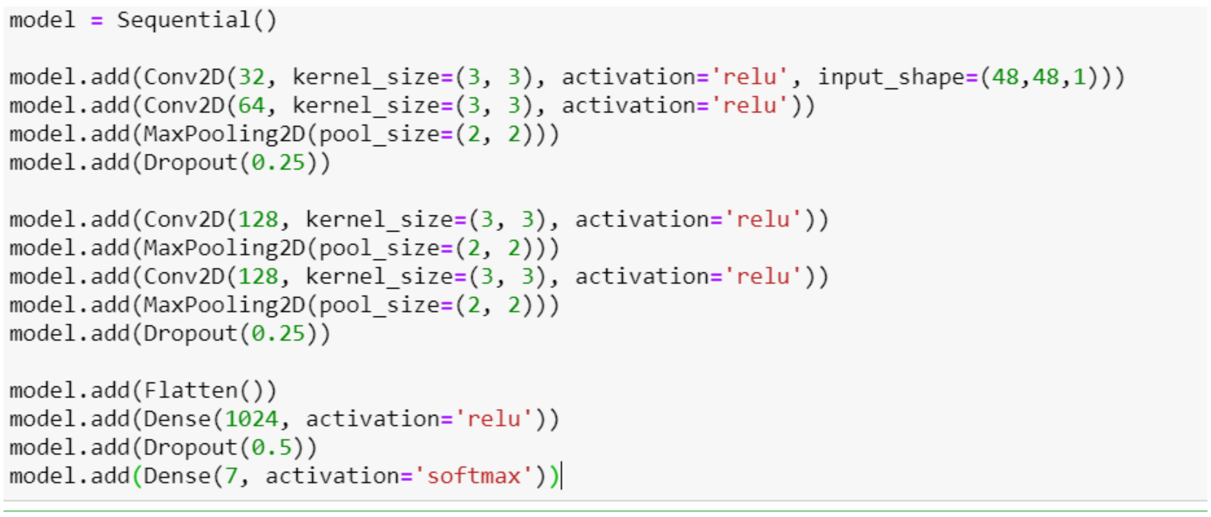
METHODOLOGY ADOPTED

Vartika's Model

Convolutional Neural Networks:

I have used Convolutional Neural Networks to create our Machine Learning Model and Train our data. The architecture for our CNN is as follows:

1. I have used a Sequential Model
2. The Model consist of 2 convolutional Blocks. Each block consists of the following:
    a. Convolutional Layer
    b. Rectified Linear Activation (ReLu) Layer to increase the non-linearity in our images
    c. MaxPooling Layer to downsize the dimensions
    d. Dropout
3. These 2 convolutional blocks are then connected to a Fully Connected Layer.
4. At the end there is a Softmax Dense layer to predict the final output using the model.



```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
```

Convolutional Block 1:

- The Conv2D layer has 32 (3x3) filters and defines the image input dimensions as (48,48,1). I have trained out model using black and white images.
- Another Conv2D layer is added which has 64 (3x3) filters and activation as ReLu layer

- The MaxPooling Layer has pool size as (2,2). Hence it helps in reducing the dimensions by 50%.
- Dropout Layer with dropout rate as 25%

Convolutional Block 2:

- Similar to Convolutional Block 1.
- The Conv2D layers have 128 (3x3) filters

Fully Connected Layer:

- A dense layer with an output dimensionality of 1024
- A Relu Layer
- Dropout Layer with dropout rate as 50%

SoftMax Layer:

- This is Last Fully Connected Layer
- Output is in the form of a vector k with dimensions (1,7) for predicting the final Emotion as '***Happy***', '***Sad***', '***Surprised***', '**Angry**', '***Disgusted***', '**Fearful**' or '***Neutral***'.

The Model had been compiled using ***Adam Optimizer*** with a learning rate of ***0.0001*** and loss function as ***Categorical Cross Entropy***

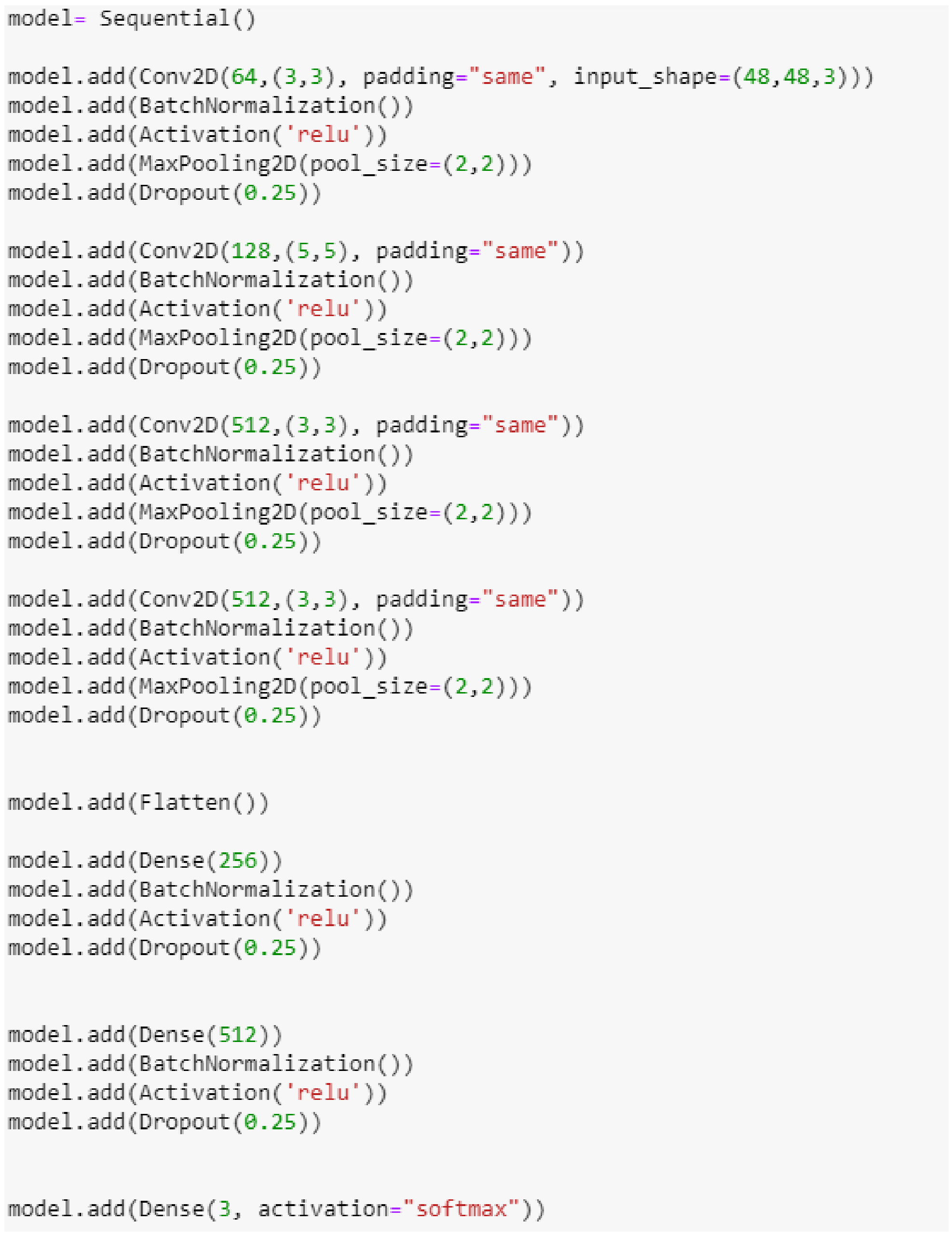
<u>*Lavanya's Model:*</u>

Convolutional Neural Networks:

I have used Convolutional Neural Networks to create our Machine Learning Model and Train our data. The architecture for our CNN is as follows:

5. I have used a Sequential Model
6. The Model consist of 4 convolutional Blocks. Each block consists of the following:
    a. Convolutional Layer
    b. Batch Normalization Layer to normalizes the output using the mean and standard deviation of the current batch of inputs
    c. Rectified Linear Activation (ReLu) Layer to increase the non-linearity in our images

   d. MaxPooling Layer to downsize the dimensions
   e. Dropout
7. These 4 convolutional blocks are then connected to two Fully Connected Layers.
8. At the end there is a Softmax Dense layer to predict the final output using the model.



```
model= Sequential()

model.add(Conv2D(64,(3,3), padding="same", input_shape=(48,48,3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding="same"))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding="same"))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding="same"))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))


model.add(Flatten())

model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))


model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))


model.add(Dense(3, activation="softmax"))
```

Convolutional Block 1:

- The Conv2D layer has 64 (3X3) filters and defines the image input dimensions as (48,48,3). I have trained out model using coloured images.
- A Batch Normalisation Layer
- ReLu Layer
- The MaxPooling Layer has pool size as (2,2). Hence it helps in reducing the dimensions by 50%.
- Dropout Layer with dropout rate as 25%

Convolutional Block 2:

- Similar to Convolutional Block 1 but with the some changes.
- The Conv2D layer has 128 (5X5) filters

Convolutional Block 3:

- Similar to Convolutional Block 1 but with the some changes.
- The Conv2D layer has 512 (3X3) filters

Convolutional Block 4:

- Similar to Convolutional Block 1 but with the some changes.
- The Conv2D layer has 512 (3X3) filters

Fully Connected Layer 1:

- A dense layer with an output dimensionality of 256
- A batch Normalization Layer
- A Relu Layer
- Dropout Layer with dropout rate as 25%

Fully Connected Layer 2:

- A dense layer with an output dimensionality of 512
- A batch Normalization Layer
- A Relu Layer
- Dropout Layer with dropout rate as 25%

SoftMax Layer:

- This is Last Fully Connected Layer
- Output is in the form of a vector k with dimensions (1,3) for predicting the final Emotion as '***Happy***', '***Sad***' or '***Surprised***'

The Model had been compiled using ***Adam Optimizer*** with a learning rate of ***0.0005*** and loss function as ***Categorical Cross Entropy***

Transfer Learning:

The model developed by us can predict the emotion of a person through his facial expressions. However, we also need to a model to detect faces. For this purpose we have applied the concept of Transfer Learning where the knowledge of one model is applied to solve to a different but related problem.

In our Project we have used the ***Haar Cascade Frontal Face*** model which can detect the presence of a face in an image. This has then been used to draw bounding boxes and capture faces in the image. Once the face is captured, it can then be used to predict the emotion of the person with the help of our custom model.

RESULTS

Vartika's Model

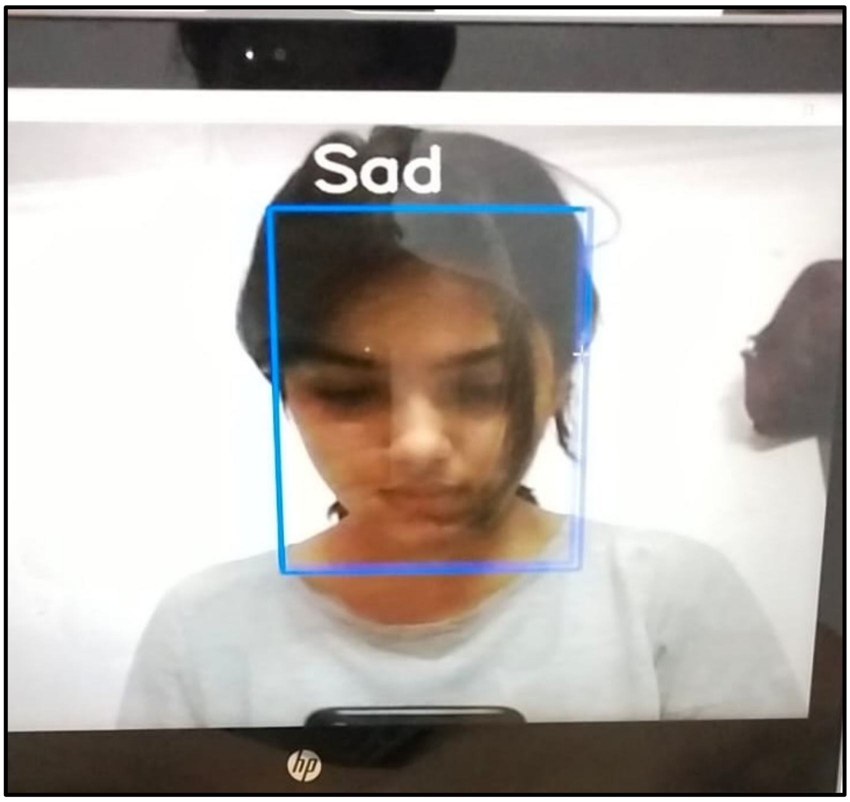


Fig 1: Web camera Frame with Sad Emotion detection

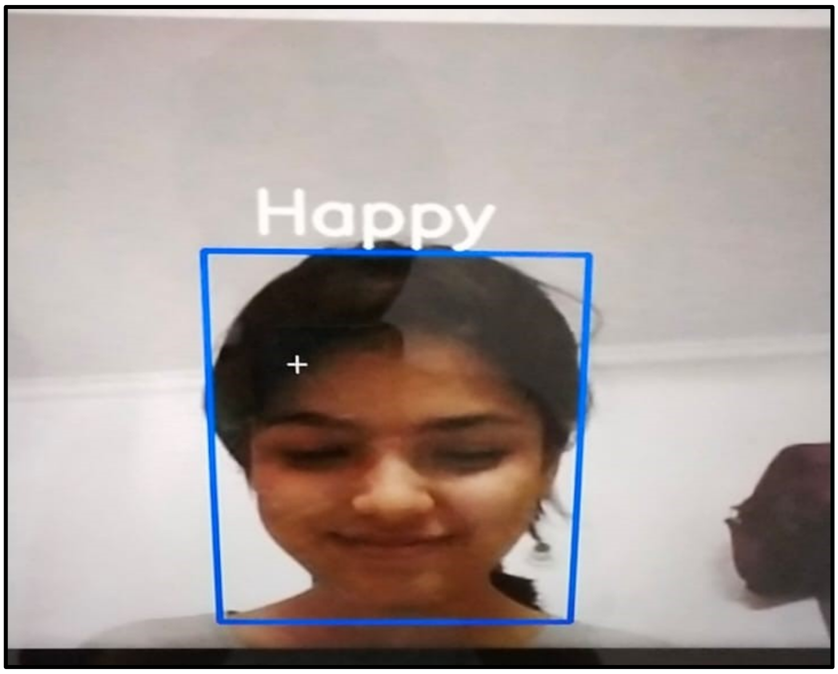


Fig 2: Web camera Frame with Happy Emotion detection

Lavanya's Model

Video Frame 1

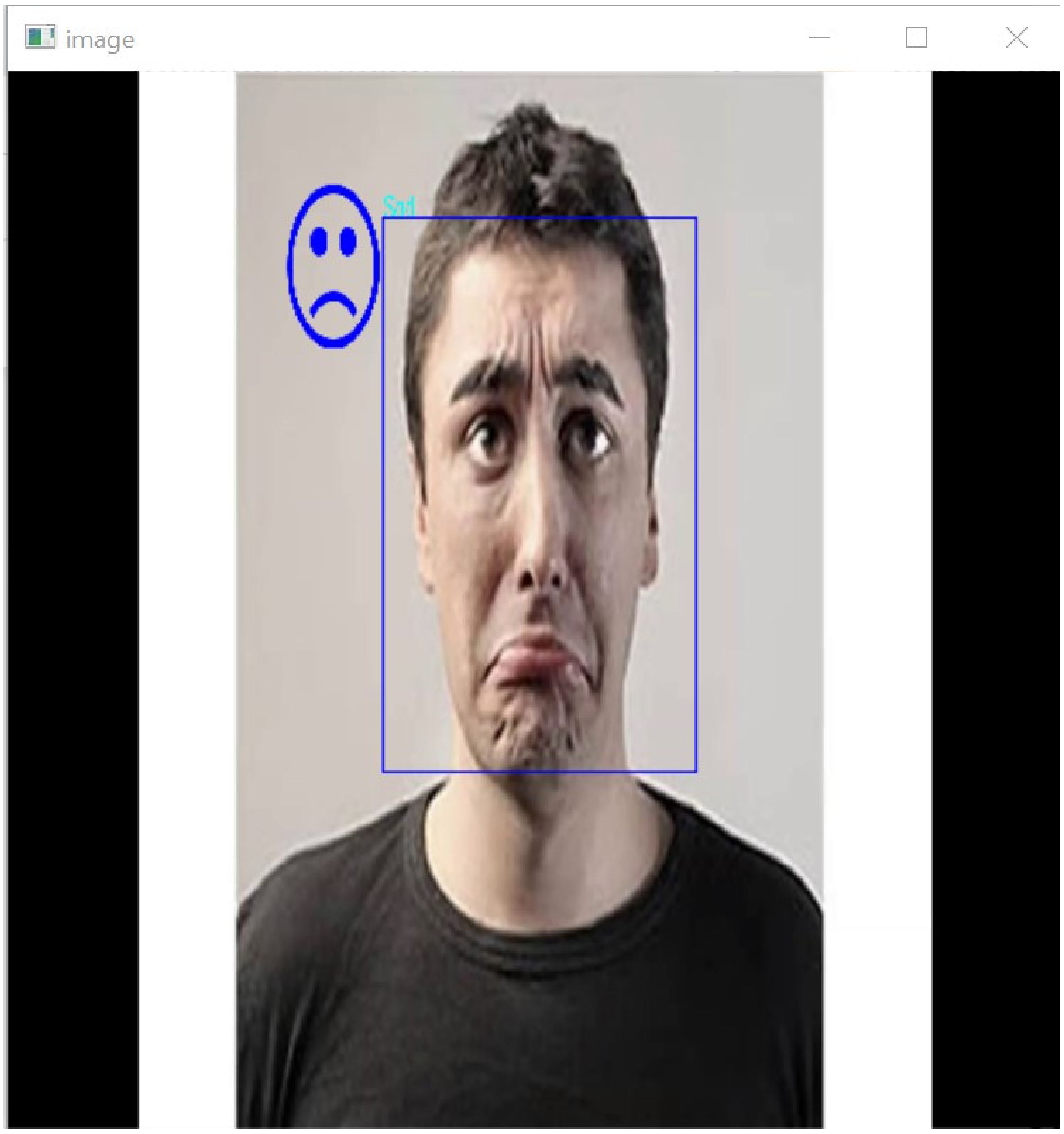


Fig 3: Video Frame with Sad Emotion and a Sad Emoji

I can see that the model has predicted the emotion to be '**Sad**'. It has formed a bounding box around the face. An emoji corresponding to the '**Sad**' Emotion has also been displayed.

Video Frame 2:

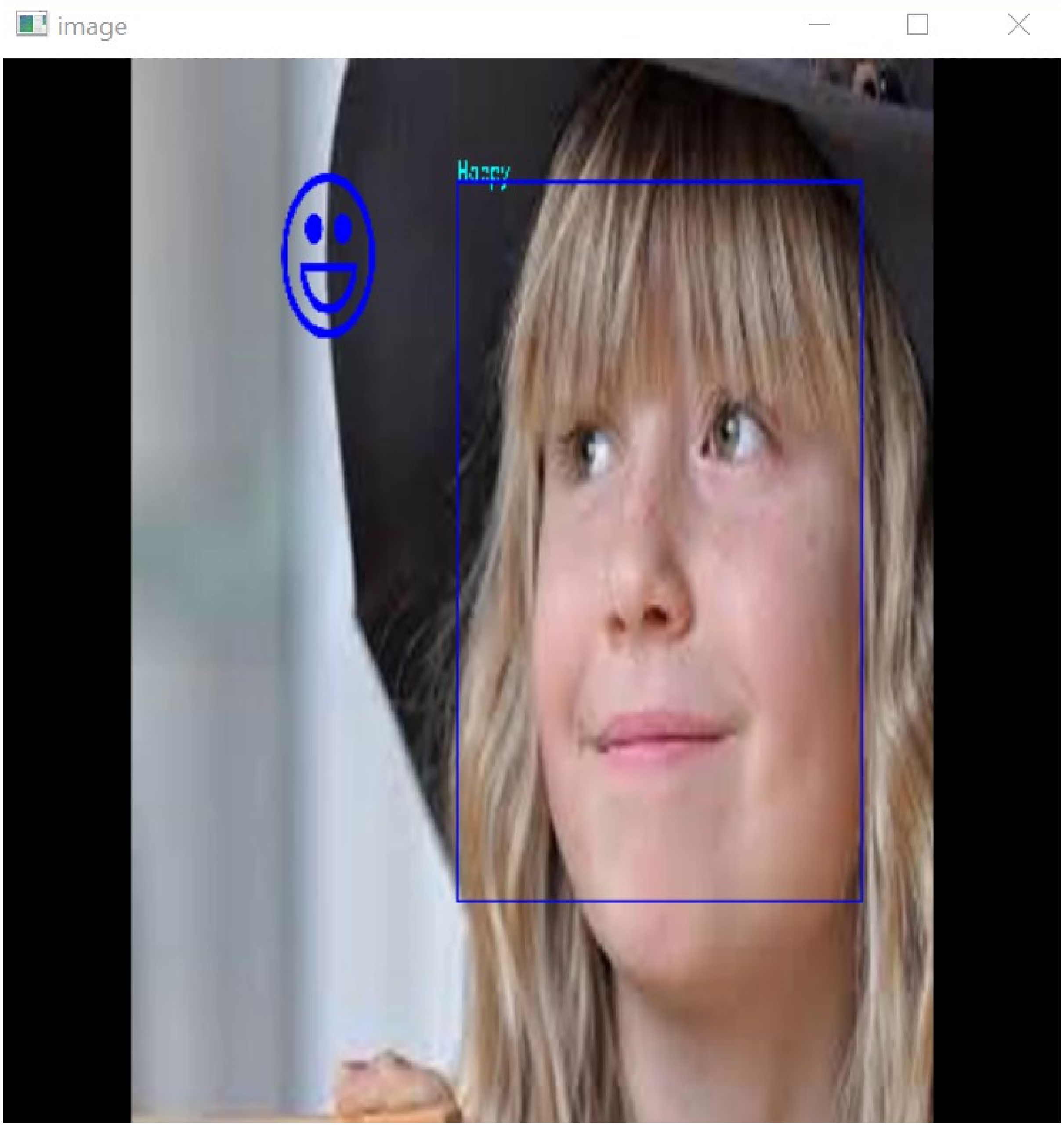


Fig 4: Video Frame with Happy Emotion and a Happy Emoji

I can see that the model has predicted the emotion to be '**Happy**'. It has formed a bounding box around the face. An emoji corresponding to the '**Happy**' Emotion has also been displayed.

Video Frame 3:

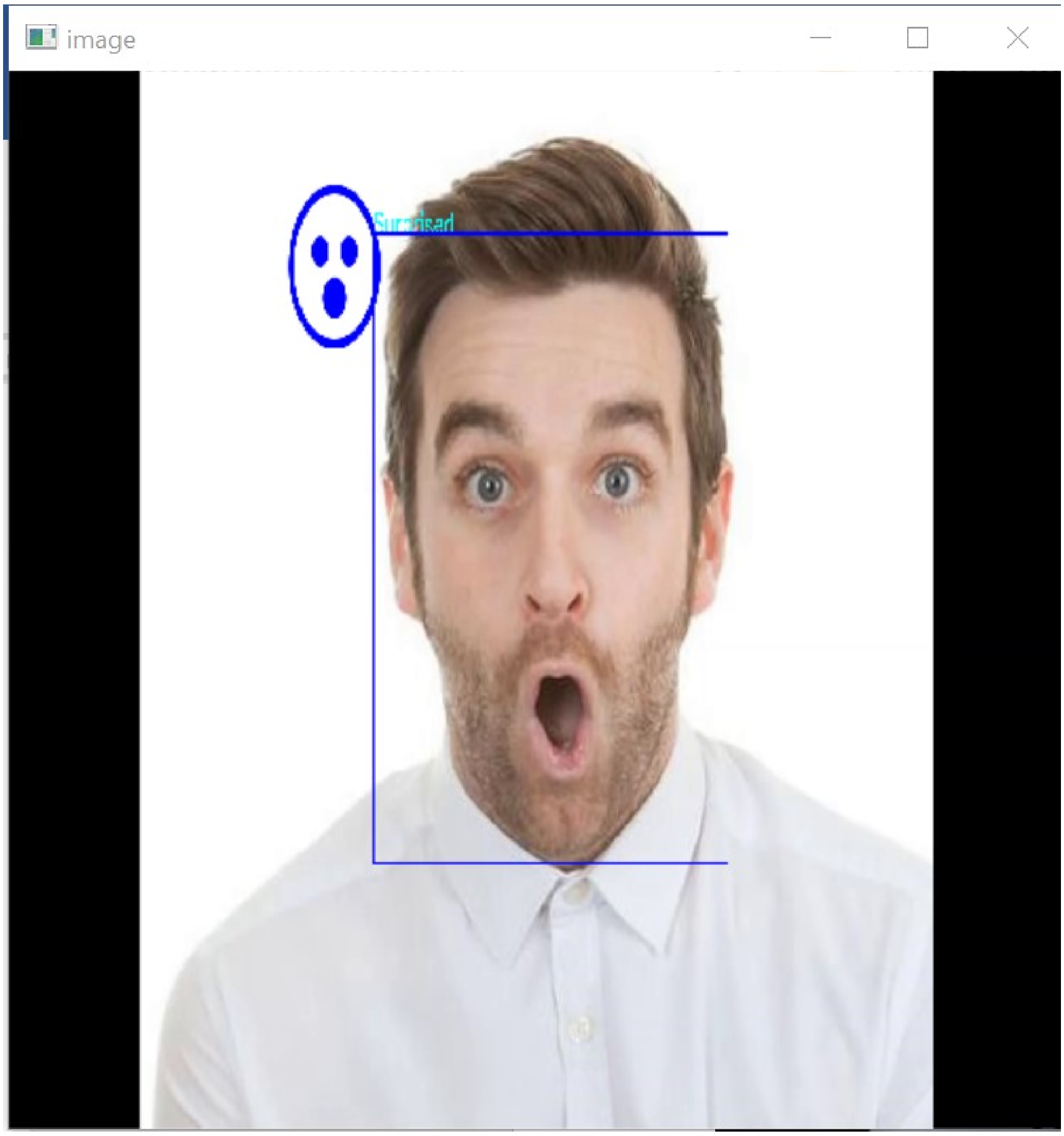


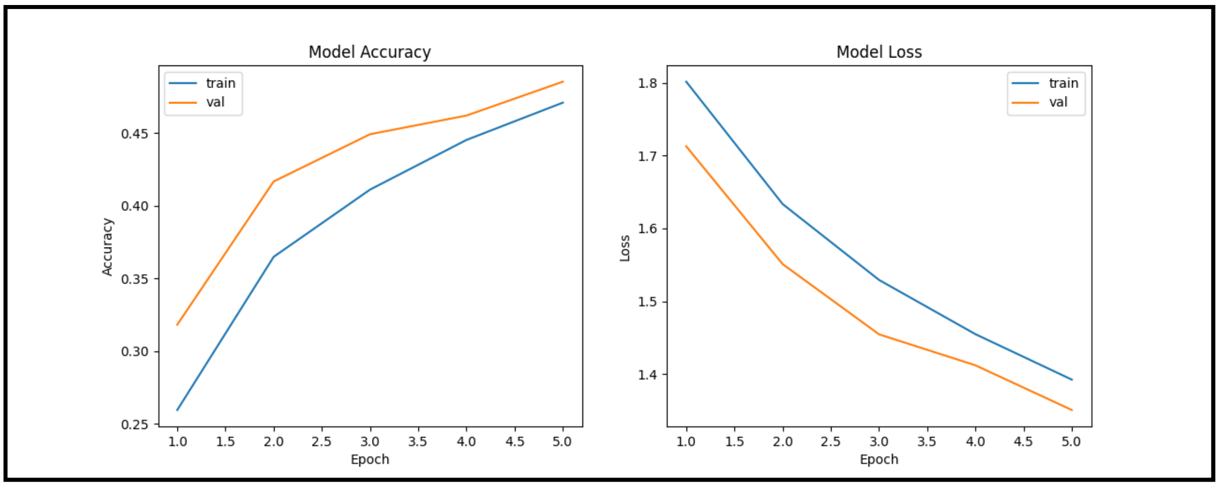
Fig 5: Video Frame with Surprised Emotion and a Surprised Emoji

I can see that the model has predicted the emotion to be '**Surprised**'. It has formed a bounding box around the face. An emoji corresponding to the '**Surprised**' Emotion has also been displayed.

ACCURACY AND EVALUATION METRICS

Vartika's Model:

On plotting the Model Training and Testing Accuracy scores, I obtain the following graph:

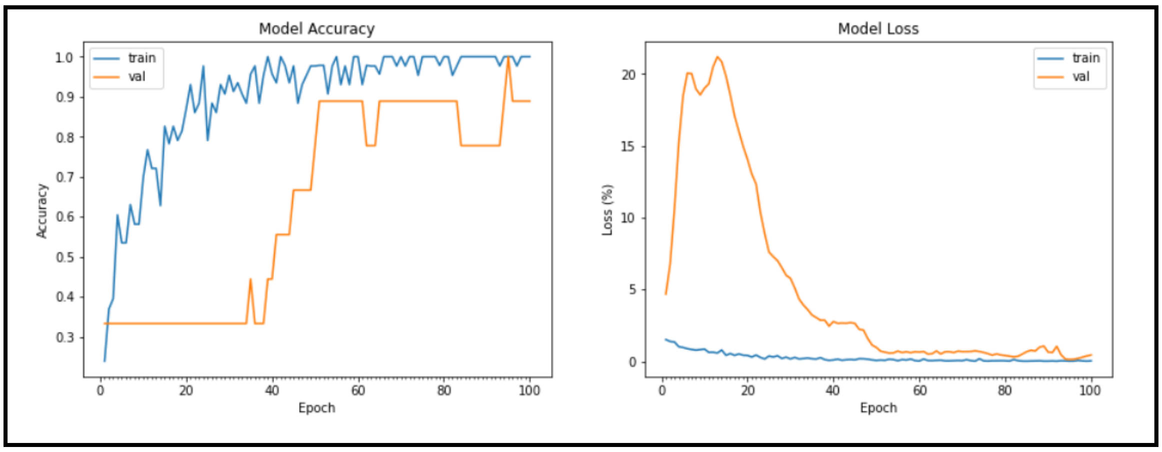


Model Accuracy on Training set was 48.54%

Lavanya's Model:

On plotting the Model Training and Testing Accuracy scores, I obtain the following graphs:

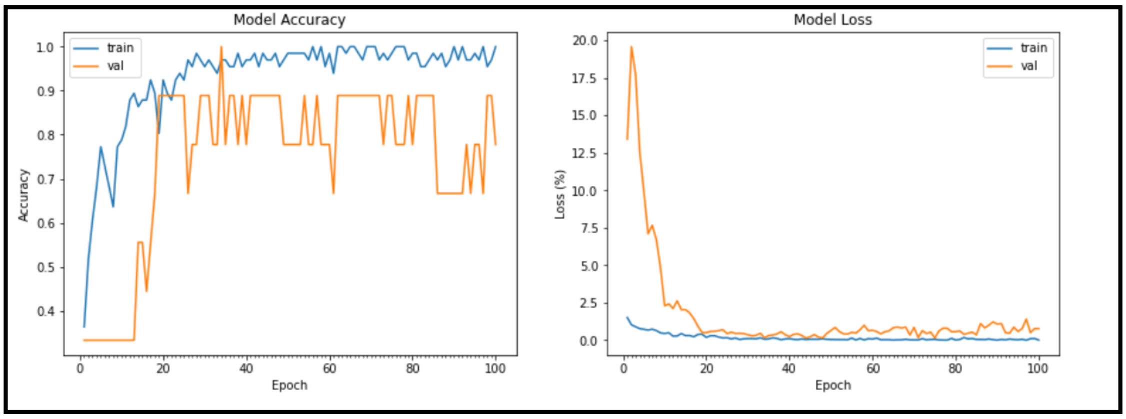
1. ***Batch size= 22***



Emotion	Images Classified Correctly	Images Classified Incorrectly	Total Number of Images
Happy	3	0	3
Sad	2	1	3
Surprised	3	0	3
Total			**9**

On Evaluating the Test Data, I get an accuracy of 88.89%

2. *Batch size =11*

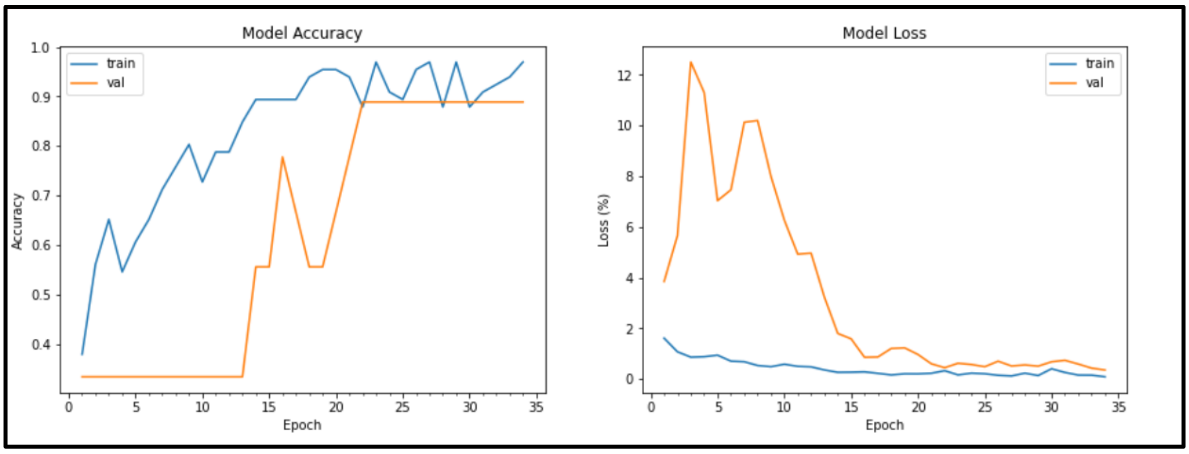


Emotion	Images Classified Correctly	Images Classified Incorrectly	Total Number of Images
Happy	2	1	3
Sad	2	1	3
Surprised	3	0	3
Total			**9**

On Evaluating the Test Data, I get an accuracy of 77.78%

From the above plot I can see that after 20th epoch, The Training and validation Accuracy graph is going in a zig-zig fashion. This indicates that the model is not learning anything new. Hence, by the end of 100 epochs, the model has resulted in overfitting. From the plot, I see that an optimal number of epochs would be 20.

3. *Batch Size =11 and number of epochs =34*



Emotion	**Images Classified Correctly**	**Images Classified Incorrectly**	**Total Number of Images**
Happy	2	1	3
Sad	2	1	3
Surprised	3	0	3
Total			**9**

On Evaluating the Test Data, I get an accuracy of 77.78%

Even after changing the Number of epochs the accuracy does not improve with respect to test data. However, the model performs better on unseen data as compared to the previous model (tested separately)

SYNTAX ERRORS AND LOGICAL ERRORS

Vartika:

1. RuntimeWarning: compiletime version 3.4 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.5 return f(*args, **kwds)-

   Solution- I Updated Tensorflow version to 1.5

2. ipykernel_launcher.py: error: unrecognized arguments.

   Solution- I change ap.add_argument("--mode",help="train/display") to ap.add_argument("-f","--mode",help="train/display")

3. Jupyter: The-kernel-appears-to-have-died-it-will-restart-automatically.

   Solution- the issue is a bottleneck in compute/memory resources in training big models, and places like colab have a lot more bandwidth than our machine. So I switched to Colab

4. Environment error- Tensorflow was installed in a different environment, had to reinstall

5. NameError: name 'plt' is not defined

   Solution. - Add "import matplotlib.pyplot as plt" to code

Lavanya:

1. The first issue that I faced was that the dataset that I collected consisted of images of different dimensions. For model training it is advised to standardise the dimensions of all images to remove bias.

   Solution – This was rectified by standardising all the dimensions to 800 x 600 pixels before I started annotating the images.

2. I wanted to train my model on only the annotated part i.e. the Region of Interest. For this purpose I had to crop my original images as per the annotations available in the csv I generated.

Solution -To successfully achieve the same, I had to read about OpenCV and its crop function.

3. datagen_test.flow_from_directory() reads data from inside the sub folders in the main testing folder.
   Solution- I had to create sub folder for each Emotion inside the testing folder to read the corresponding images. Earlier I had a common test folder with the image corresponding to different emotions

4. While Making the Neural Network, I had to identify that the input image dimensions need to match the image dimensions used in the datagen_test and datagen_train.
   Solution - Hence, for both of them I made it common by ensuring the image Dimensions to be (48X48)

5. While passing the images for prediction and for visualisation of bounding boxes, I faced an error 'mismatch of dimensions- Tensor must be of the same dimensions'. This was due to the fact that TensorFlow adds an extra dimension to the image while processing it for training.
   Solution – I solved this by adding this: np.expand_dims(img, axis = 0)

6. The biggest issue I faced was to render the emoji corresponding to the facial Expression identified. When the Unicode for the same was used, it displayed just empty boxes.
   Solution – I solved this issue by downloading the Symbola.ttf file which is a font style used to display emojis.

INDIVIDUAL CONTRIBUTION

Lavanya Middha

- Literature Survey
- Collection of Dataset
- Pre-processing of Images
- Script for Model Training using Convolutional Neural Networks.
- Script for Capturing Video frames and Displaying the Predicted Results
- Script for Rendering the Corresponding emoji for the emotion detected

Vartika Trivedi

- Literature Survey
- Pre Processing of Images
- Script for Model Training using Convolutional Neural Networks
- Script for Capturing real time from Laptop's camera and Displaying the Predicted Results

Vartika Trivedi – 18BCE0962

Lavanya Middha – 18BCE0845

CONCLUSION

We developed two Emotion Detection Models with the help of Convolutional Neural Networks.

Model 1 – Vartika's Model:

An input through the laptops' Web camera was passed and the trained model was successfully able to detect faces in video frames and identify the emotion on those faces. An output in the form of bounding boxes and the corresponding emotion was also displayed.

Model 2 – Lavanya's Model:

An input video was passed and the Model was successfully able to detect faces in video frames and identify the emotion on those faces. An output in the form of bounding boxes and the corresponding emoji was also displayed. The Model had an accuracy of 88.89%.

CONSTRAINTS

1. The biggest constraint that we faced was the size of the dataset. Our custom dataset had only 75 images. As the dataset was small, we could provide less instances for training and testing which resulted in misclassifications as the model could not learn to the best of its ability.
2. Our dataset was not very diverse. The dataset should have contained images corresponding to facial images with variations to remove bias. For example:
    a. More pictures with people wearing spectacles. Adding spectacles to the face, may cause some of the facial features to not be explicitly visible and hence creating a huge difference in identifying emotions.
    b. More variation in age groups of people. As people grow old, they tend to gain wrinkles on their skin. If adequate data is not provided, then the model may confuse these wrinkles with some of the facial characteristics it has learnt to identify emotion. (For eg: mixing wrinkles with a neutral face emotion may lead to a detection of different kind of emotion)
    c. More Variation with respect to gender. Male and Female face structures are different and so are their facial features. If the model should perform perfectly, it must see all kind of possible input data to avoid misclassification.

3. Both the Models used Haar Cascade for detecting faces. However, we felt that this is not the best model for doing so. In many images, the Haar Cascade model could not even detect the presence of face and hence emotion detection could not be performed on those images.

FUTURE SCOPE

1. I can increase the precision of our model by adding more dataset which is more diverse.
2. I can use different techniques and architectures for training our model.
3. Instead of using Haar Cascade Frontal Face pretrained model for detection of faces, I can use other models such as VGGface which are more robust in detecting faces.
4. Hyperparameter Tuning can be done to arrive at better results.
5. These Models developed can be integrated with various apps, especially social media platforms and chatting platforms, where an emoji can be generated based on their mood.

PROJECT LINK

Vartika:

https://github.com/Vartika11t/EmotionDetection

Lavanya:

https://github.com/LavanyaMiddha/Emotion-Detection

REFERENCES

[1] vjgpt/Face-and-Emotion-Recognition

https://github.com/vjgpt/Face-and-Emotion-Recognition

[1] SanjayMarreddi/Emotion-Investigator

https://github.com/SanjayMarreddi/Emotion-Investigator

[3] BishalLakha/Facial-Expression-Recognition-with-Keras

https://github.com/BishalLakha/Facial-Expression-Recognition-with-Keras

[4] magical2world/Facial-expression-recognition-with-VGG

https://github.com/magical2world/Facial-expression-recognition-with-VGG

[5] ptbailey/Emotion-Recognition

https://github.com/ptbailey/Emotion-Recognition

[6] oarriaga/paz

https://github.com/oarriaga/paz