Human Emotion Recognition

Project By	-	Dhruv Yadav
Course work	-	Digital Image Processing

- Deep Learning
- CNN (Convolutional Neural Networks)

References

- https://openaccess.thecvf.com/content_cvpr_2017/html/Li_Reliable_Crowdsourcing_and_CVPR_2017_paper.html - theory was fantastic and helpful
- https://www.kaggle.com/datasets/muhammadhananasghar/human-emotions-datasethes - for dataset
- https://github.com/WuJie1010/Facial-Expression-Recognition.Pytorch - this link was really helpful for getting to know the structure and ideas of the code writing
- https://chatgpt.com - for debugging and some prompts based codes.

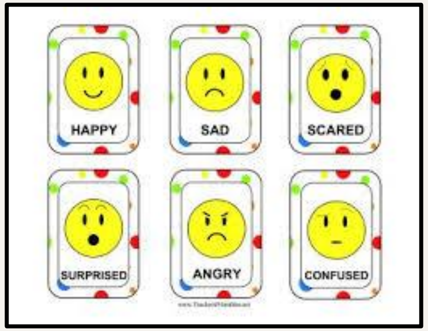
“When dealing with people, remember you are not dealing with creatures of logic, but with creatures of emotion”.

—Dale Carnegie

Problem Statement

Write a code which can recognize human emotions as **happy**, **sad** and **angry.**

I've used python for this purpose



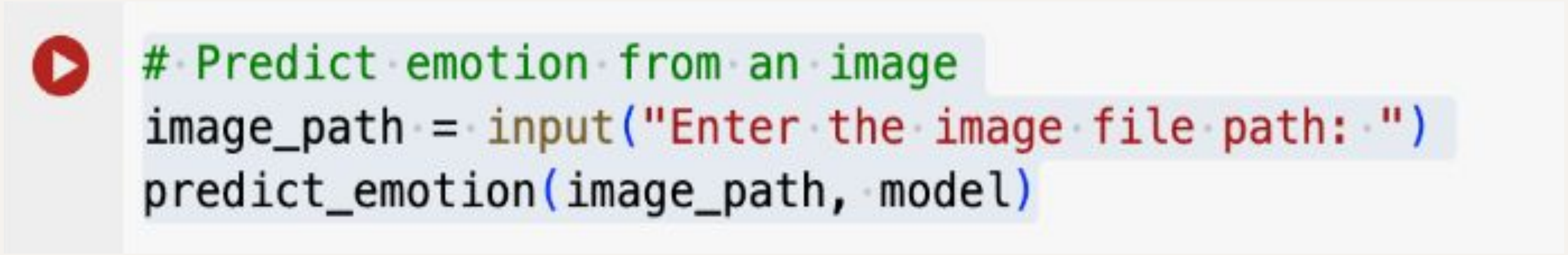
Motivation

As the quote goes, it’s much evident that emotions are indistinguishable aspect of human existence.

- Emotion recognition aims to automatically detect human emotions using various modalities like facial expressions, speech, or physiological signals.
- It has applications in human-computer interaction, mental health monitoring, and affective computing.
- Enhancing human-computer interaction by enabling machines to understand emotions.
- Improving mental health diagnostics through automated emotion analysis and Enhancing recommendation systems and personalized content delivery based on user emotions.

Input Format

My code takes an image as input, which is included in the folder or file path is properly described when asked. Following lines are responsible for getting the input



```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)
```

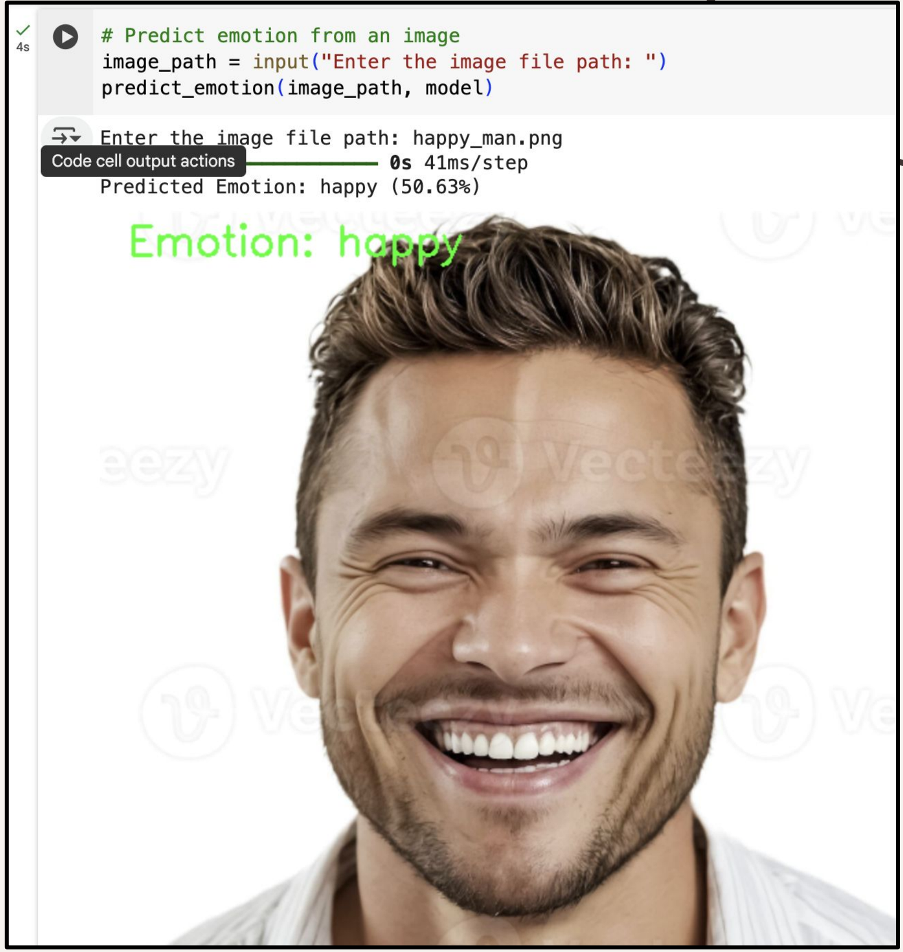
Explanation :

image_path is a string which is being taken as an input from the user.
predict_emotion is a function which predicts the emotion based on the training given to the model using the dataset.

Output Format

My code while returning output, displays the image along with the emotion written. It also claims the emotion with a confidence (based on the training, it tells to what extent is the person angry, happy or sad. It may vary with normal human experience as its knowledge is based on the dataset which it is using.

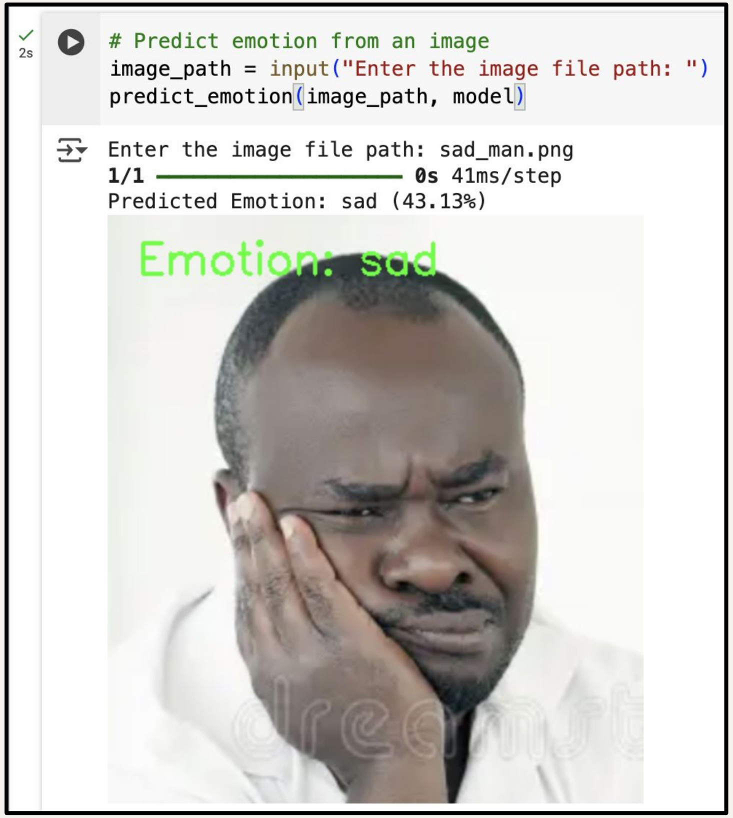
I’ve also included dataset in the folder uploaded in the google form.



```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)
```

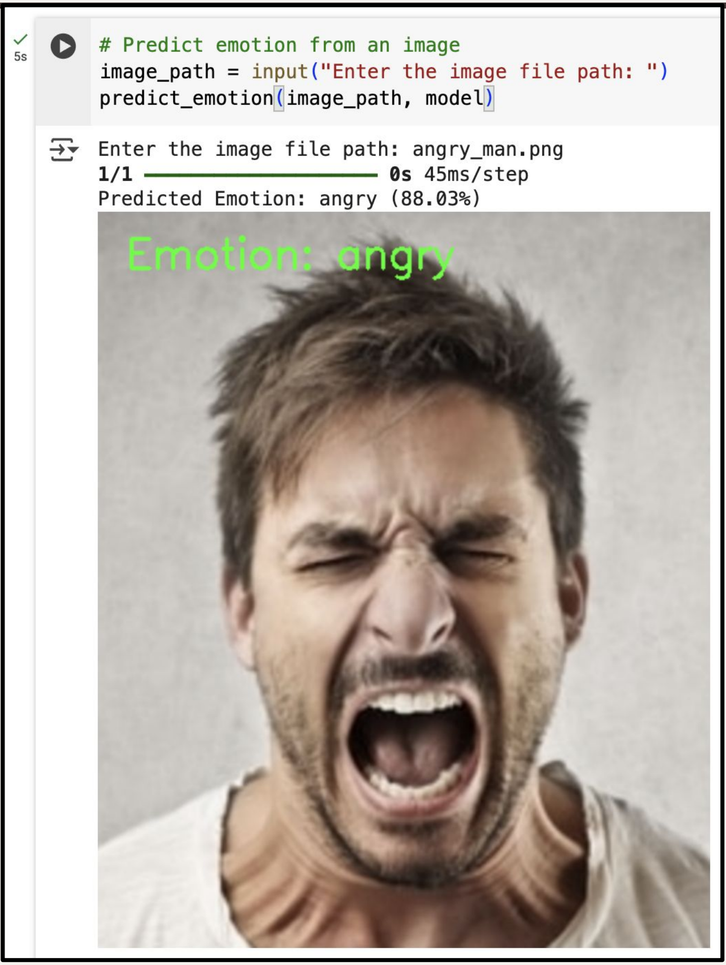
```
Enter the image file path: happy_man.png
                                 0s 41ms/step
Predicted Emotion: happy (50.63%)
```

Output Format



```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)
```

```
Enter the image file path: sad_man.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Predicted Emotion: sad (43.13%)
```



```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)
```

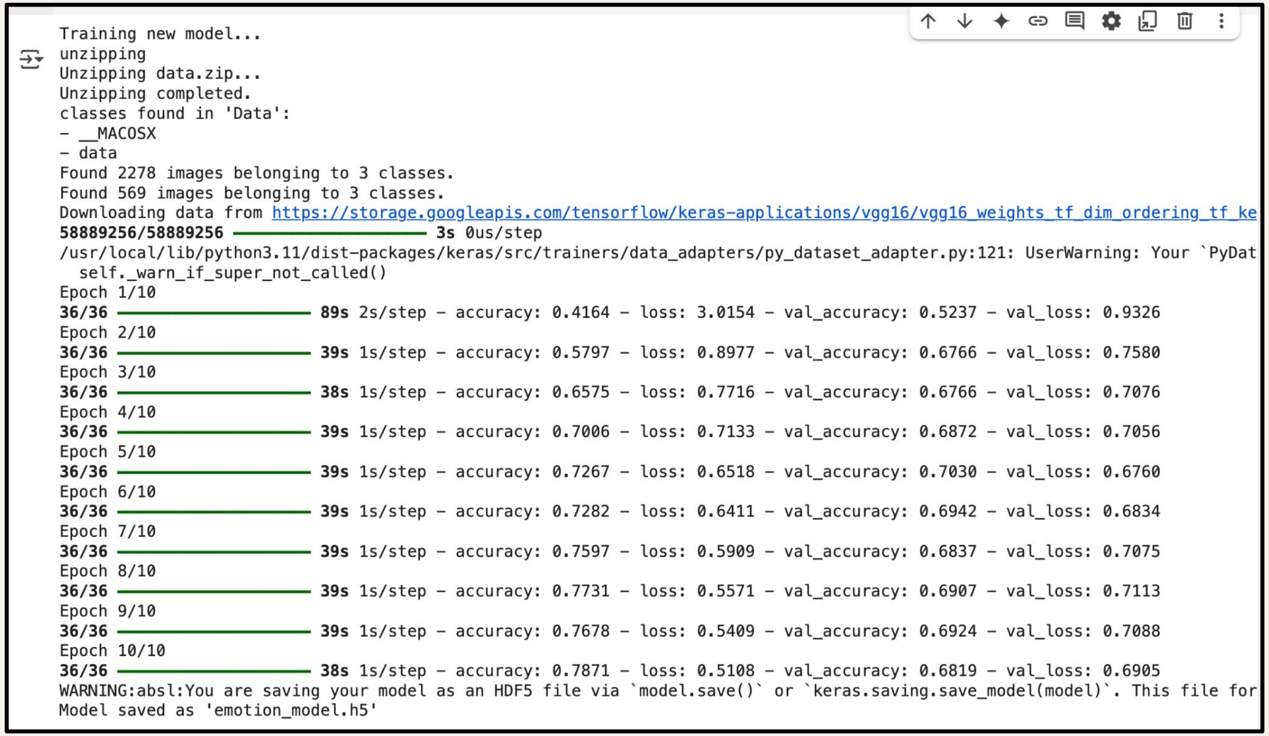
```
Enter the image file path: angry_man.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Predicted Emotion: angry (88.03%)
```

1 Standard Data-sets in Human Emotion Recognition

My experience with Standard Data-sets was not at all good in Human Emotion Recognition. I couldn’t get non-erratic data set. Those which were without / with less errors were not in decent resolution (they were like 50 x 50 etc.) and those which were with errors gave indecent results.

I had to used a data-set which was not with proper expressions (for training purpose). My results were bad due to that, it recognised my furious photo as sad. I then removed the erratic photos and retrained it with that data-set. But then data-set left was not of enough size. Finally I had to use the one which gave the best among all after experiments.

2 Evaluation Metrics - accuracy and loss



```
Training new model...
unzipping
Unzipping data.zip...
Unzipping completed.
classes found in 'Data':
- __MACOSX
- data
Found 2278 images belonging to 3 classes.
Found 569 images belonging to 3 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_ke
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDat
  self._warn_if_super_not_called()
Epoch 1/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 89s 2s/step - accuracy: 0.4164 - loss: 3.0154 - val_accuracy: 0.5237 - val_loss: 0.9326
Epoch 2/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.5797 - loss: 0.8977 - val_accuracy: 0.6766 - val_loss: 0.7580
Epoch 3/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 38s 1s/step - accuracy: 0.6575 - loss: 0.7716 - val_accuracy: 0.6766 - val_loss: 0.7076
Epoch 4/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7006 - loss: 0.7133 - val_accuracy: 0.6872 - val_loss: 0.7056
Epoch 5/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7267 - loss: 0.6518 - val_accuracy: 0.7030 - val_loss: 0.6760
Epoch 6/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7282 - loss: 0.6411 - val_accuracy: 0.6942 - val_loss: 0.6834
Epoch 7/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7597 - loss: 0.5909 - val_accuracy: 0.6837 - val_loss: 0.7075
Epoch 8/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7731 - loss: 0.5571 - val_accuracy: 0.6907 - val_loss: 0.7113
Epoch 9/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 39s 1s/step - accuracy: 0.7678 - loss: 0.5409 - val_accuracy: 0.6924 - val_loss: 0.7088
Epoch 10/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 38s 1s/step - accuracy: 0.7871 - loss: 0.5108 - val_accuracy: 0.6819 - val_loss: 0.6905
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file for
Model saved as 'emotion_model.h5'
```

Here you can notice that there are four metrics in each Epoch:

1. accuracy
2. loss
3. val_accuracy
4. val_loss

Each of them are the metrics which define the compatibility of the model and the data-set for the purpose.

2 Evaluation Metrics - accuracy and loss

Metrics Used in Training :

1. **Accuracy :**
   - Measures the proportion of correctly classified images.
   - Formula : $$\text{Accuracy} = \frac{\text{Total Predictions}}{\text{Correct Predictions}}$$
   - **Intuition :** Higher accuracy means better classification

2. **Loss :**
   - Measures the difference between predicted and actual class probabilities.
   - Formula : $$\text{Loss} = -\sum y_i \log(y_i)$$
   - **Intuition :** Lower loss means better performance

2 Evaluation Metrics - accuracy and loss

Validation Metrics (During testing) :

1. **Validation Accuracy (val_accuracy) :**

   - Accuracy on unseen test data.

   - Intuition : If train accuracy is high but val accuracy is low, the model is **overfitting**.

2. **Validation Loss (val_loss) :**

   - Measures loss on validation data set.

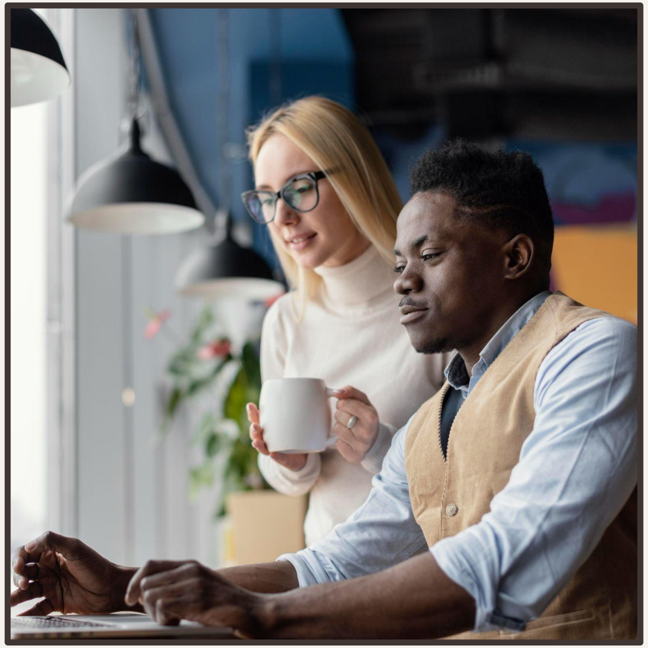
   - Intuition : If training loss decreases but val_loss increases, the model is **overfitting**.

3

Approach used in code

I have used Deep Learning
Specifically the Architecture Named CNN
CNN stands for Convolutional Neural Networks

It is one of the most popular Architecture used
for Image Processing and Pattern detection
which works on mask / filter convolution.

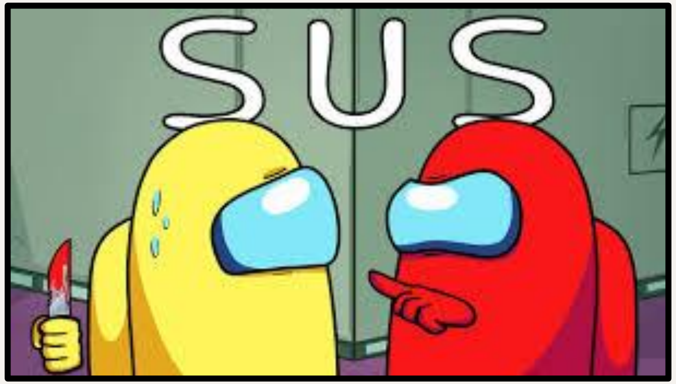


Whoa!

How to do this Human Emotion Recognition !

Deep Learning :

Deep learning is a subset of machine learning that focuses on using artificial neural networks to process and learn from large amounts of data. It is inspired by the structure and function of the human brain and is particularly effective for tasks involving complex patterns, such as image recognition, natural language processing, and autonomous systems.



How does it work !

CNN (Convolutional Neural Networks) which is used for image processing.

CNN is a popular architecture used in Deep Learning for image processing. We already know about convolution. CNN is related.

Like in convolution, CNN uses **mask / filter** to scan small regions of input image; extracts features such as **edges**, **textures**, and **patterns**. Each filter detects a different feature (e.g., vertical edges, horizontal edges, curves), and finally it uses **dot product** operations and slides over the image with **strides** (steps).

In one algorithm, multiple CNN masks may be used for detection of different features.

3 Approach used in code - brief overview

Emotion Recognition using CNNs and Transfer Learning :

- **Objective** **:** Classify images into three emotions : **Angry, Happy, Sad**.
- **Method** **:** Use **VGG16(Pretrained CNN)** + Custom Layers

Data Preparation :

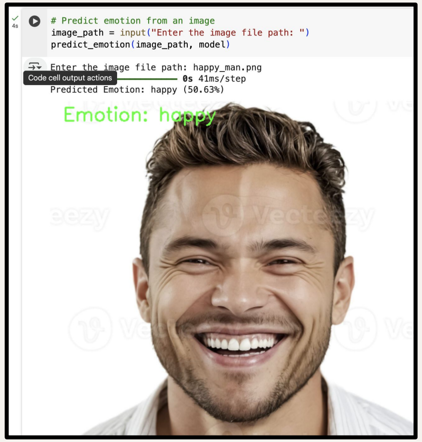
- **Image Preprocessing :**
  - **Resizing** **: not needed here** (images already in **224 x 224**)
  - **Normalization** **:** scaling pixel values to [0, 1]
  - **Argumentation** **:** Zoom, rotation to make model more robust **(optional)**

VGG16 is a pre-trained model trained on **ImageNet** (1M+ images).
We **freeze its convolutional layers** to retain feature extraction.
Custom layers : Flatten Layer, Dense Layer (128 neurons, ReLU activation), Dropout (0.1), Final Layer (3 neurons, Softmax activation) etc. are added

3 Approach used in code - brief overview

Custom Layers added :

- Flatten Layer	: Converts feature maps into a 1D array	
- Dense Layer	**: (128 neurons, ReLU activation)**	: Learn patterns
- Dropout (0.1)	: Prevents overfitting	
- Final Layer	**: (3 neurons, Softmax activation)**	: Outputs class probabilities

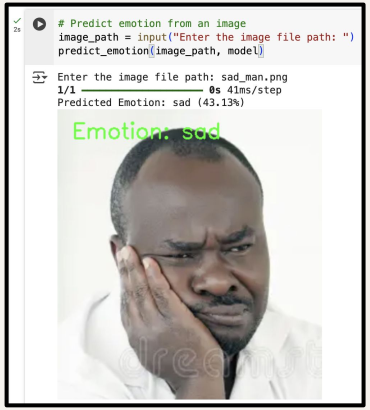


```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)
```

```
Enter the image file path: happy_man.png
                    0s 41ms/step
Predicted Emotion: happy (50.63%)
```

Prediction Pipeline : Making predictions on new images

- Load image & resize to **224 x 224**
- Convert to array and normalize
- Predict Class using Softmax Output
- Display Image with Predicted Emotion



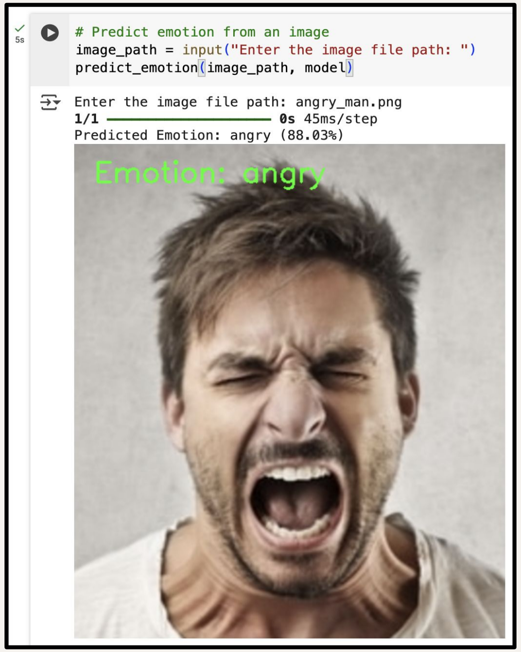
```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)
```

```
Enter the image file path: sad_man.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Predicted Emotion: sad (43.13%)
```

4 Results obtained by testing my code and comparison with paper

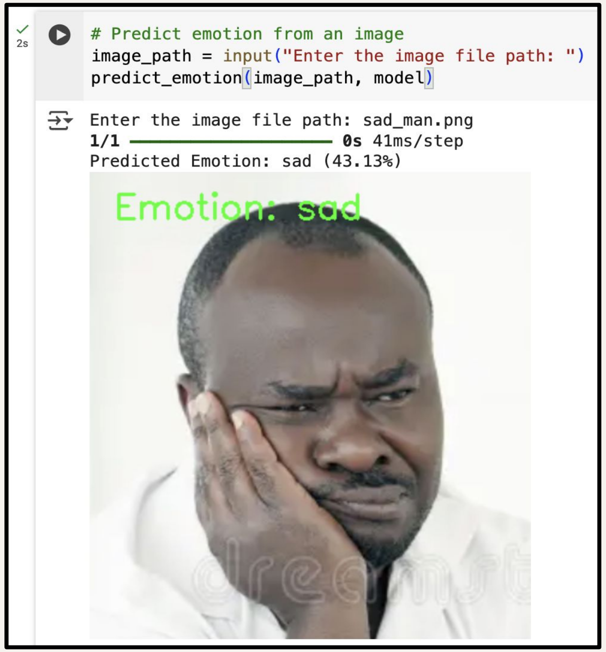
I noticed that the results that I had obtained were not always correct. However, for the images in which the emotions were clearly visible were identified correctly. Other than that, it sometimes recongnized the frowning which is observed while crying as an act of anger and similarly some other weird results were observed. In some clear expressions the recognition percentage were less. However, emotions were identified. I have attached some images that showed these kind of results, in the next slide.

4 Results obtained by testing my code and comparison with paper



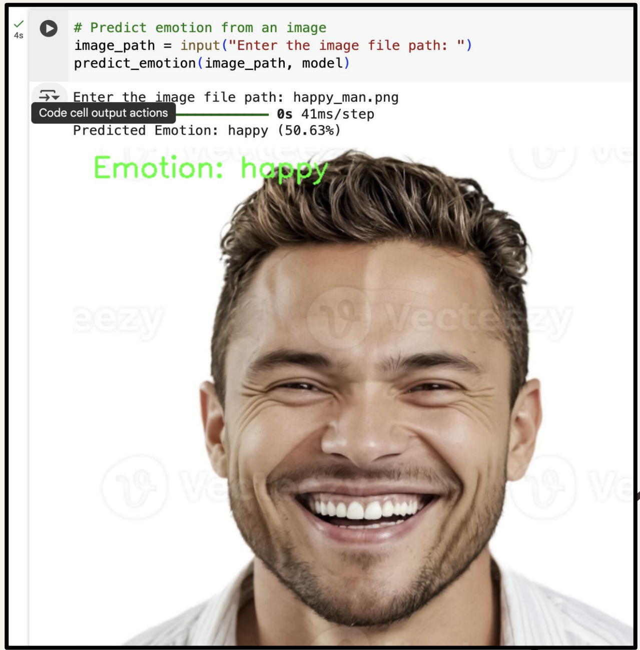
```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)

Enter the image file path: angry_man.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
Predicted Emotion: angry (88.03%)
```



```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)

Enter the image file path: sad_man.png
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Predicted Emotion: sad (43.13%)
```



```
# Predict emotion from an image
image_path = input("Enter the image file path: ")
predict_emotion(image_path, model)

Enter the image file path: happy_man.png
━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Predicted Emotion: happy (50.63%)
```

6 My Learning from this project

I learnt about some fascinating concepts of **Deep Learning** and **Machine Learning**. Especially about **CNN** (Convolutional Neural Networks).

In context of this project, I tried a few things :-

- Training with small datasets **(undersampling).**
- Training extensively so that model starts to cram **(overfitting).**
- What happens when one of emotions is overfed (it’s almost always returned)

Other than that, I tried different data sets and realized that feeding correct data is more important that the quality of model you use. I also tried using the models such as **ResNet.** However, I rejected those because **VGG16** was giving best results for me.