Capstone Project

Live Class Monitoring System (Face Emotion Recognition)

Technical Documentation
By

Team

Kanishka Raj, Rakhi Kumari, Raushan Kumar
Sridhar Nagar

AlmaBetter

Date: 30th January 2022

Table of Contents

Abstract:

The Indian education landscape has been undergoing rapid changes for the past 10 years owing to the advancement of web-based learning services, specifically, eLearning platforms.

In a physical classroom during a lecture, the teacher can see the faces and assess the emotion of the class and tune their lecture accordingly, whether he is going fast or slow. He can identify students who need special attention. Digital classrooms are conducted via a video telephony software program (ex-Zoom) where it's not possible Convthe to see all students and access the mood. Because of this drawback, students are not focusing on content due to a lack of surveillance. Digital platforms have limitations in terms of physical surveillance but it comes with the power of data and machines which can work for you. Its data can be analyzed using deep learning algorithms which not only solves the surveillance issue but also removes the human bias from the system.

Problem Statement:

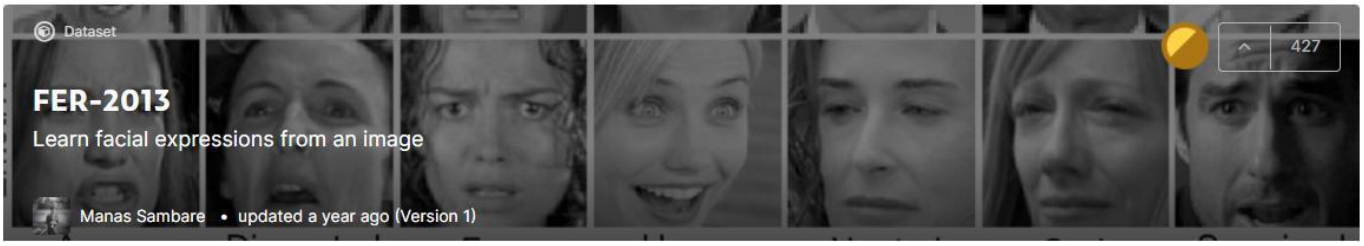
To construct and face emotion recognition model for Live Class Monitoring System.

Human emotions and intentions are expressed through facial expressions and deriving angles. The efficient and effective feature is the fundamental component of the facial expression system. Facial expressions convey non-verbal cues, which play an important role in interpersonal relations. Automatic recognition of facial expressions. can be an important component of natural human-machine interfaces.

during counseling and other healthcare-related fields. In an E-Learning system, the presentation style may be varied depending on the student's state. However, in many cases, static emotion detection is not very useful. It is essential to know the user's feelings over a period of time in a live environment. Thus, the paper proposes a model that is aimed at real-time facial emotion recognition

Data Summary:

The data comes from the past Kaggle competition "Challenges in Representation Learning: Facial Expression Recognition Challenge": we have defined the image size to 48 so each image will be reduced to a size of 48x48. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. Each image corresponds to a facial expression in one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The dataset contains approximately 36K images.



- Dataset link - https://www.kaggle.com/msambare/fer2013

Steps involved:

1. Exploratory Data Analysis

The dataset contains approximately 36K images. Each image corresponds to a facial expression in one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). All the images are 48x48 pixels.

2. Data Image generator

Generate batches of tensor image data with real-time data augmentation.

3. Building models

We started our project with the concept of transfer learning i.e. using the pre-trained weights of ResNet50 for the training of the model and then fine-tuning them to increase the accuracy. Due to our dissatisfaction with this result, we switched to the Custom CNN model.

4. Training the model

We trained the model with 40 epochs and callback as early stopping and ReduceLROnPlateau to avoid overfitting and reach the global minima.

5. Model Evaluation

We evaluated the model using an accuracy plot and categorical cross-entropy loss and confusion matrix to find out in which category the model has inadequate performance and among which category the model is getting confused

6. Model Deployment

We have created a front-end using Streamlit for web app and used streamlit-webrtc which helped to deal with real-time video streams. Image captured from the webcam is sent to the VideoTransformer function to detect the emotion. Then this model was deployed on the Heroku platform with the help of buildpack-apt which is necessary to deploy the OpenCV model on Heroku.

Deployment Link for Heroku -

https://face-emotion-recog-app.herokuapp.com/

Deployment Link for Streamlit Share -

https://share.streamlit.io/raushan9jnv/face-emotion-recognition/main/app.py

Models:

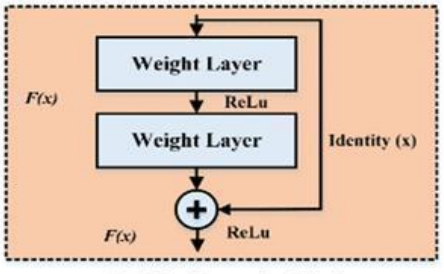
Basic CNN architecture details:

- Input layer - The input layer in CNN should contain image data
- Convo layer - The convo layer is sometimes called the feature extractor layer because features of the image are get extracted within this layer
- Pooling layer - Pooling is used to reduce the dimensionality of each feature while retaining the most important information. It is used between two convolution layer
- Fully CL - Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training and placed before the output layer
- Output Layer - The output layer contains the label which is in the form of a one-hot encoded

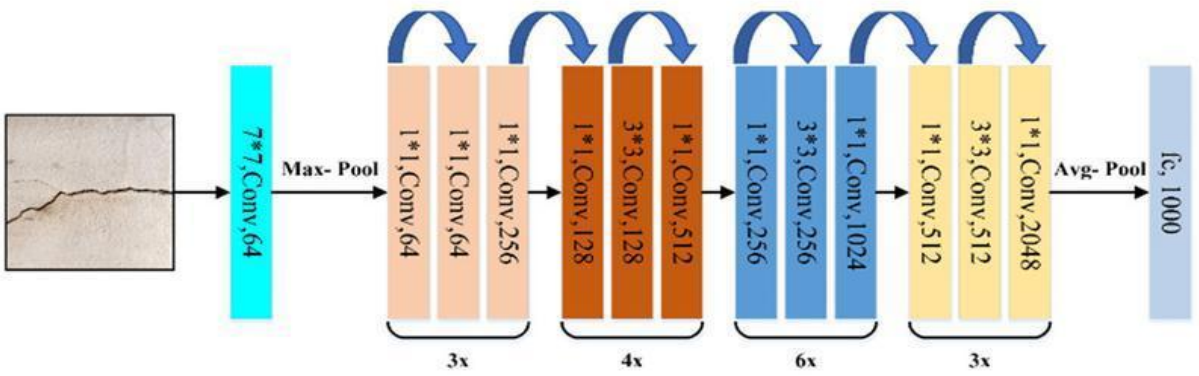
1. ResNet50:

ResNet50 is a variant of the ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations.

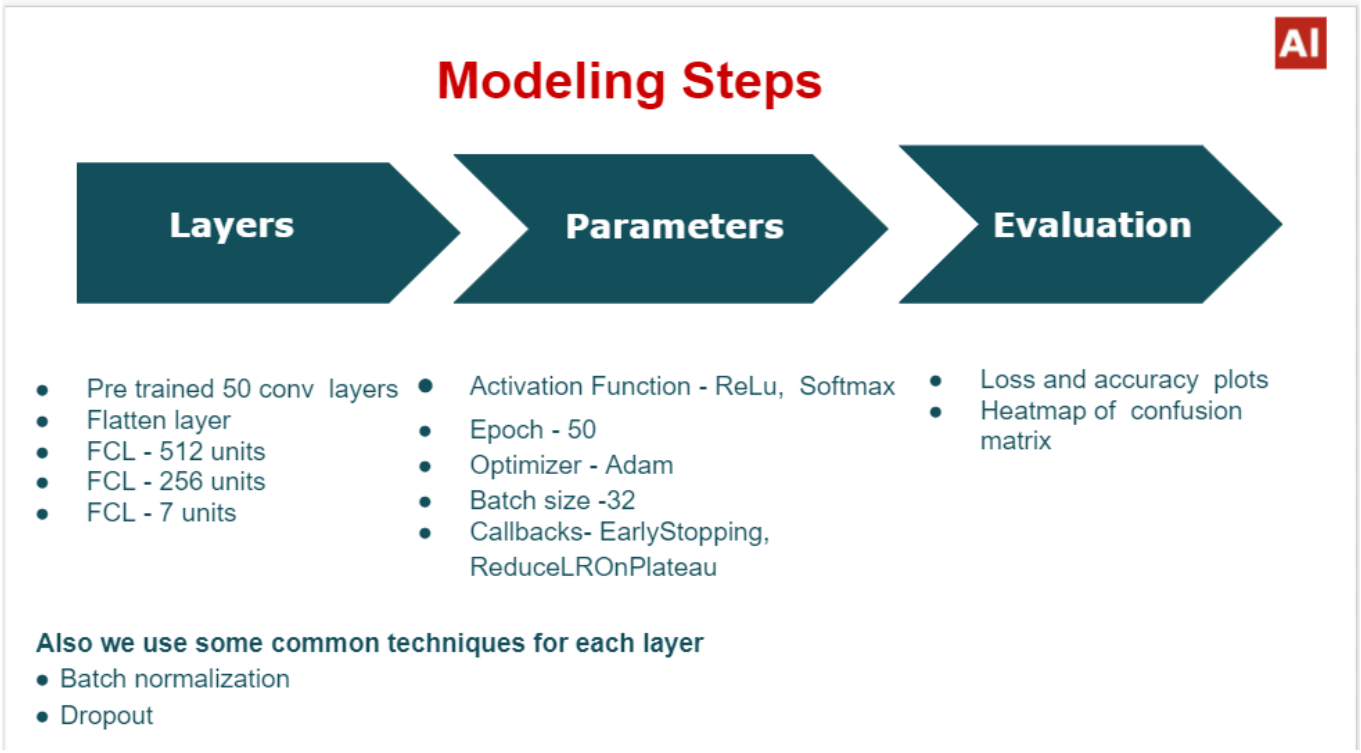
The ResNets were initially applied to the image recognition task but as mentioned in the paper that the framework can also be used for non-computer vision tasks to achieve better accuracy.



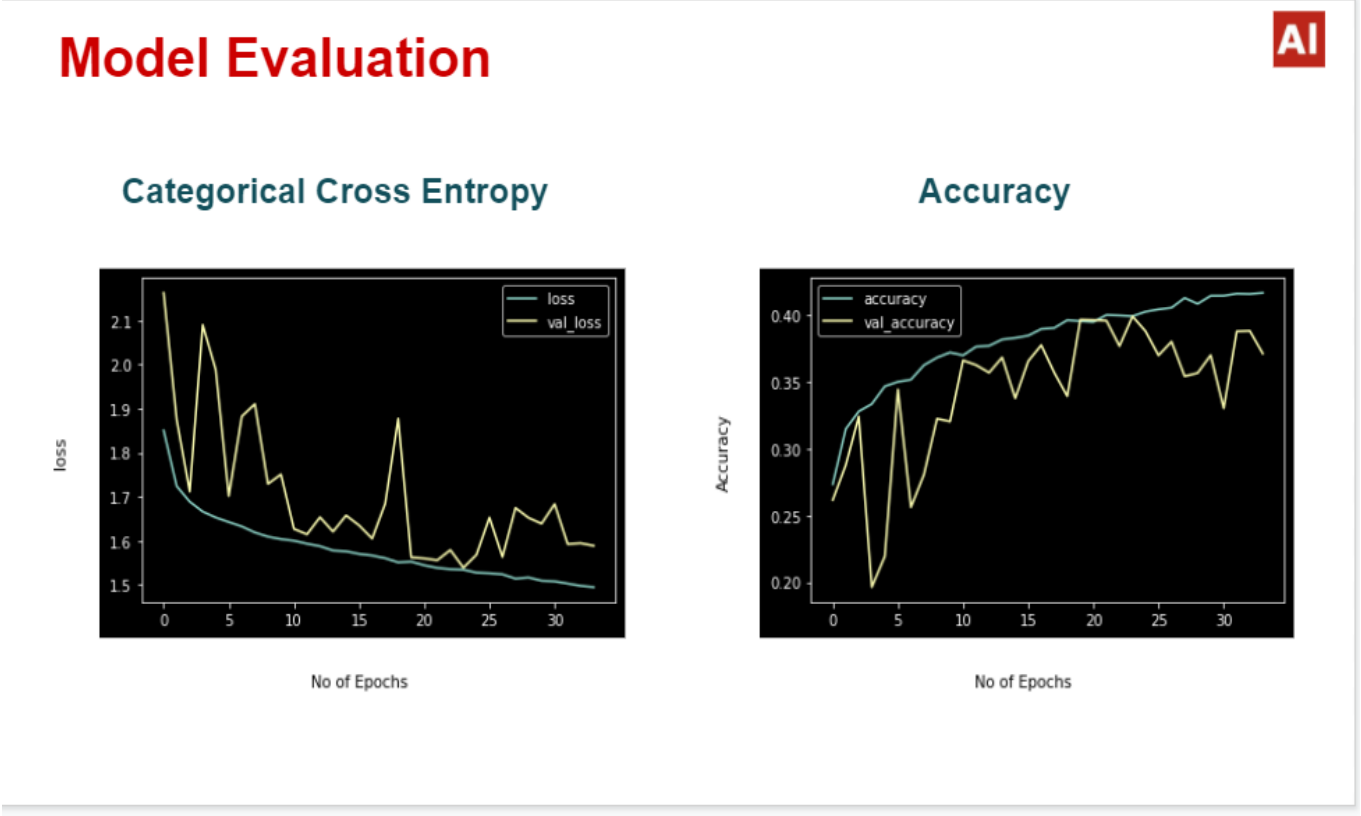
Residual Learning Block



We started with ResNet50 and added 2 FC layers and trained the model freezing all Conv layers except the last 4 and on the second run, we finetuned the model by unfreezing all the layers. And got an accuracy of 40 and 60% respectively.

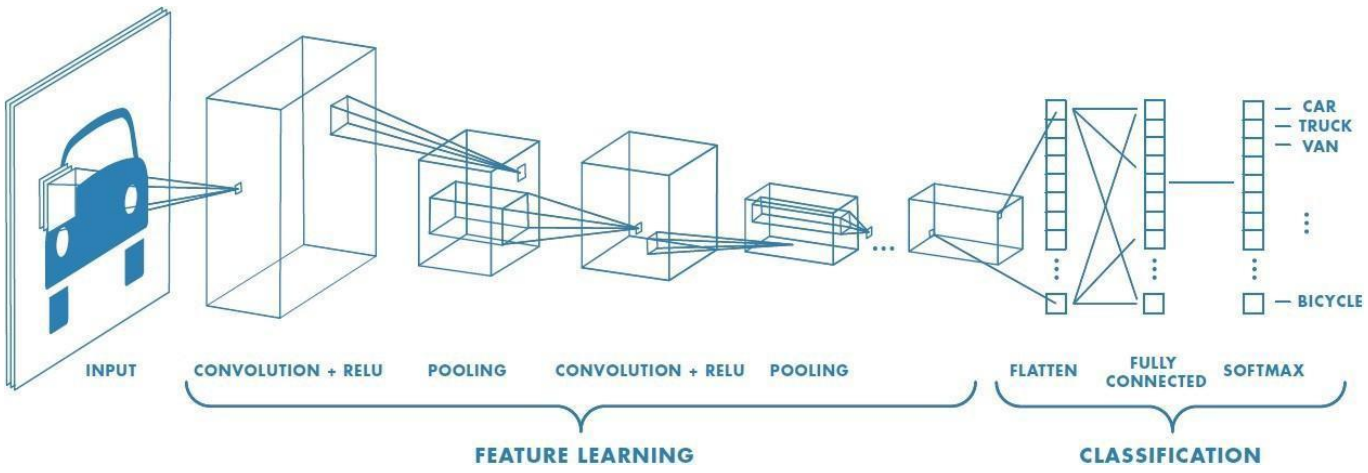


Model evaluation:



2. Convolutional Neural Network:

CNN's are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition and processing that is specifically designed to process pixel data.

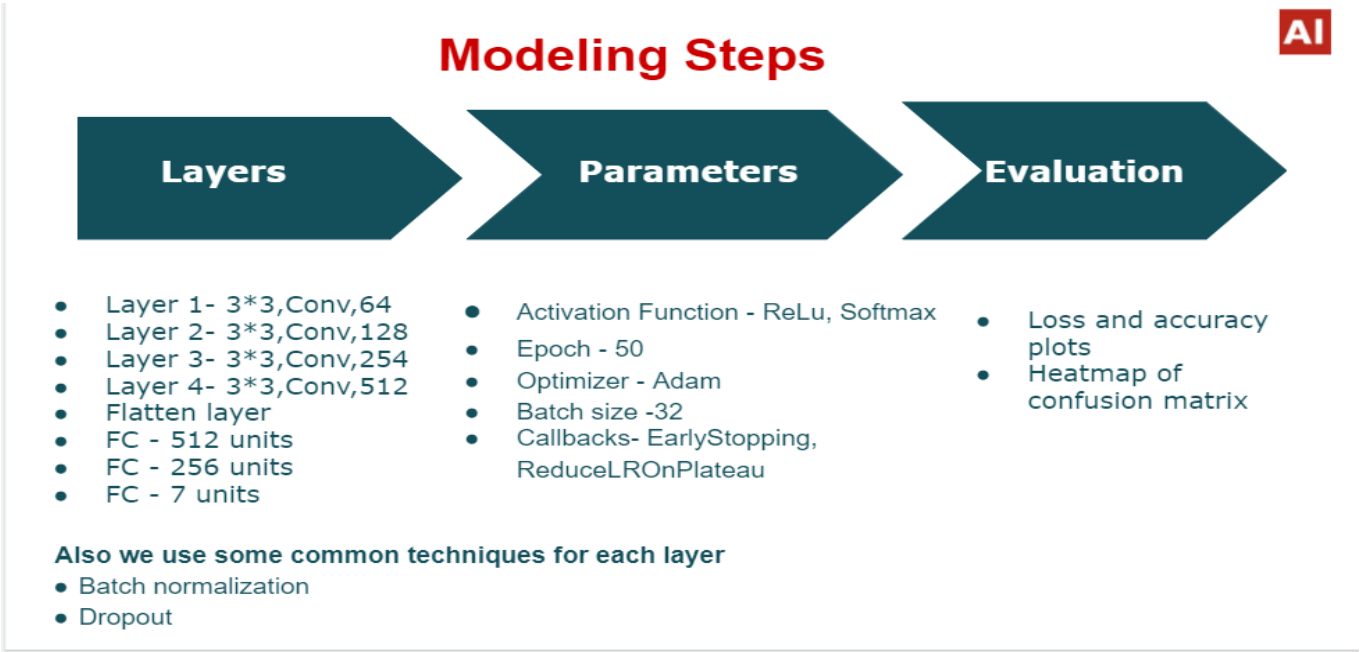


We define our CNN with the following global architecture:

- 4 convolutional layers
- 2 fully connected layers

Also, we use some common techniques for each layer

Batch normalization: improves the performance and stability of NNs by providing inputs with zero mean and unit variance. Dropout: reduces overfitting by randomly not updating the weights of some nodes. This helps prevent the NN from relying on one node in the layer too much.



Model performance:

The model can be evaluated by various metrics such as:

1. Accuracy

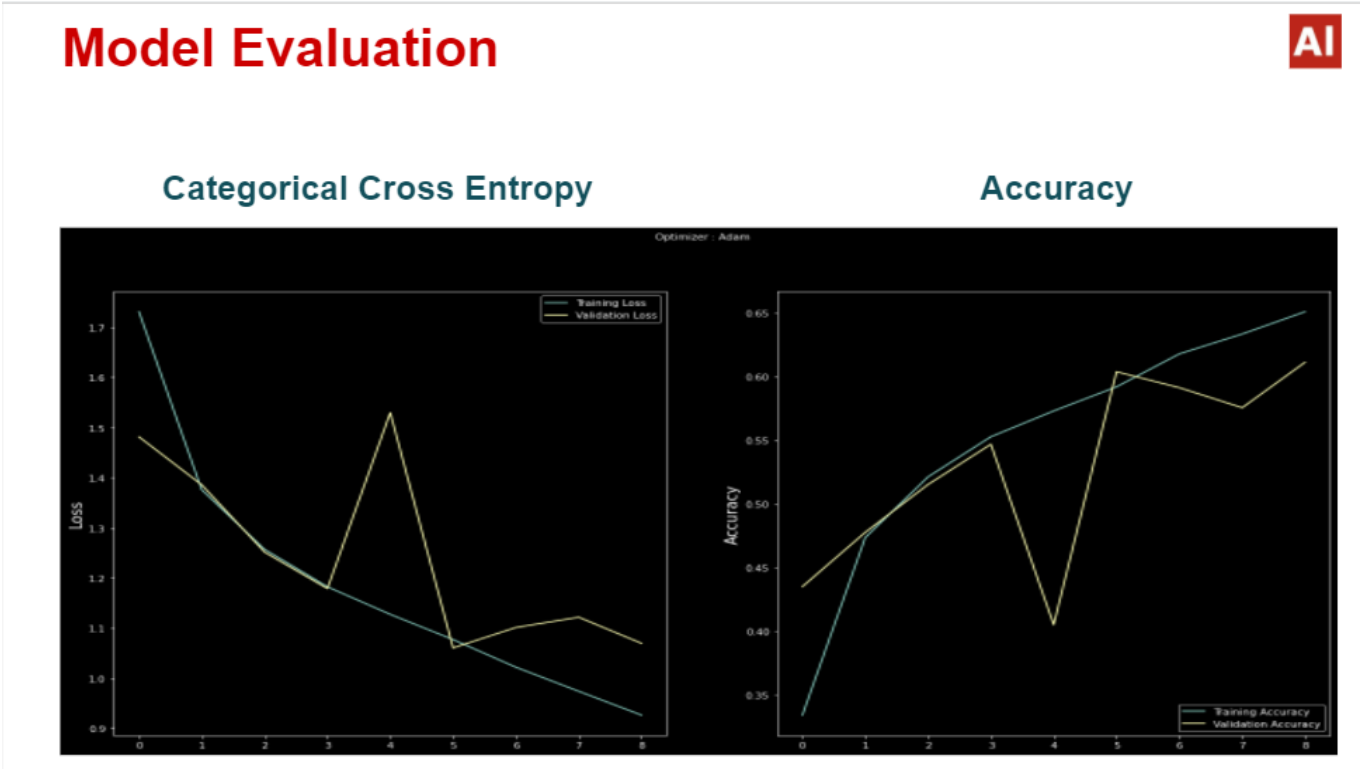
Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observations to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positives and false negatives are almost the same. Therefore, you have to look at another parameter to evaluate the performance of your model. For our model, we have got 0.66 which means our model is approx. 66% accurate.

2. Loss(categorical cross-entropy)

The categorical cross-entropy loss function calculates the loss of an example by computing the following sum:

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

where y^i is the i-th scalar value in the model output,y_i is the corresponding target value, and the output size is the number of scalar values in the model output.



3. Confusion Matrix (Normalized)

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa

Our model is very good for predicting happy and surprised faces. However, it predicts quite poorly feared faces maybe because it confuses them with sad faces.it also gets confused between angry and disgusted faces.

Challenges

- Large image dataset to handle.
- limited GPU access on Google colab notebook.
- Every training epoch takes too much time so experimenting with different data splits takes more time to give useful results.
- Hyperparameter tuning is very sensitive for these Neural Network models. So we make changes very careful
- After seeing all the model scores we need to find the best model based on different matrix scores which is time-consuming because models take a lot of time to compute the result.
- Running webcam on Google Colab
- Deployment

Conclusion

- We built the WebApp using streamlit and deployed it in Heroku and Streamlit Sharing.

- The model which was created by a custom CNN model gave training accuracy of 74% and test accuracy of 60%.which is maximum from the rest of all four models we used in this project.

- We have also included the video of my WebApp working in Local. Local face detection helps us to evaluate the performance after every small change in the code cell.

- Codes, which we deployed are present in Github Repository.

- It was such an amazing and interesting project. We learned a lot from this. We did lots of experiments with respect to data set distribution for the model and saw the effects on the performance.