Capstone Project

Live Class Monitoring System (Face Emotion Recognition)

By Sayan Bandopadhyay

Synopsis

Due to the ever changing structure of Indian education landscape, also owing to the COVID lockdown protocols, large number of students started attending online classes. This change comes with its own drawbacks. Unlike the physical classroom, teachers are unable to understand the student's emotion during the class. This tampers the communicational aspect between the students and teacher

This problem can be dealt by applying deep learning algorithms to live video data in order to recognize student's emotion during the class. So, this is basically a problem regarding face emotion detection.

This problem can be solved by creating some neural networks models which will be able to classify and detect facial emotions in real time scenario.

Data Summary

The dataset used to tackle this problem is downloaded and accessed from 'Kaggle'. Following is the link of the dataset: https://www.kaggle.com/msambare/fer2013

The data consists of 35,887 images in total which are 48x48 pixel grayscale images of faces. 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. This particular dataset was pre-split into train and test data directories. Most importantly, the dataset has 7 classes to where each class represents a face emotion. Following are the classes:

0 = Angry
1 = Disgust
2 = Fear
3 = Happy
4 = Sad
5 = Surprise
6 = Neutral

Continued….

After walking through both the train and test directory, below table is created which summarises the number of images in each of the seven class directories of both train and test data.

Emotion Class	Number of images in training data	Number of images in testing data
Angry	3995	958
Disgust	436	111
Fear	4097	1024
Happy	7215	1774
Sad	4830	1247
Surprised	3171	831
Neutral	4965	1233

It is clear from the table that,

- In training data, 'happy' class has the highest number of images that is 7215 and 'disgust' class has the lowest number of images that is 436.
- In testing data, 'happy' class has the highest number of images that is 1774 and 'disgust' class has the lowest number of images that is 111.

Data Visualisation

Now it is time to visualise how the images in the data actually looks like. Following are the random images of each of the seven classes:

ANGRY

DISGUST

FEAR

SURPRISE

HAPPY

NEUTRAL

SAD

Project Dependencies

- Google Colab : Google Colab is a free Jupyter notebook environment that runs entirely in the cloud which allows anybody to write and execute arbitrary python code.
- Python 3.0 : The most popular programming language of latest version is used for the overall process.
- TensorFlow 2.0 : It is a open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning.
- OpenCV : It stands for 'Open-Source Computer Vision' and is a popular computer vision library that incorporates numerous computer vision algorithms.
- GPU : In order to make the training process faster, access to the GPU is must needed. Here, 'Tesla K80' GPU is allocated by Google colab.
- VS Code : It stands for 'Visual Studio Code' which is a code editor redefined and optimized for building and debugging modern web and cloud applications. In our case its used for developing the face emotion detection web application.
- Streamlit : Its an open-source application framework which helps in creating the web applications in minutes.

Model Building

Data is pre-processed and augmented as per requirement followed by model building. The objective here is to try out different models such as CNN and some pre-trained architectures which comes under transfer learning approach. The model with highest accuracy will be taken into consideration and saved in order to use it in creating the web application. Architectures which will be used are as follows:

- ResNet50 (Transfer learning) : "ResNet50" is a convolutional neural network that is 50 layers deep. ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. It is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. First model is created using this pre-trained architecture (Model name – model_A).

- Convolutional Neural Networks (CNN) : A convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Two models are created using CNN architecture (Model names – model_B & model_C).

Common Modelling Steps

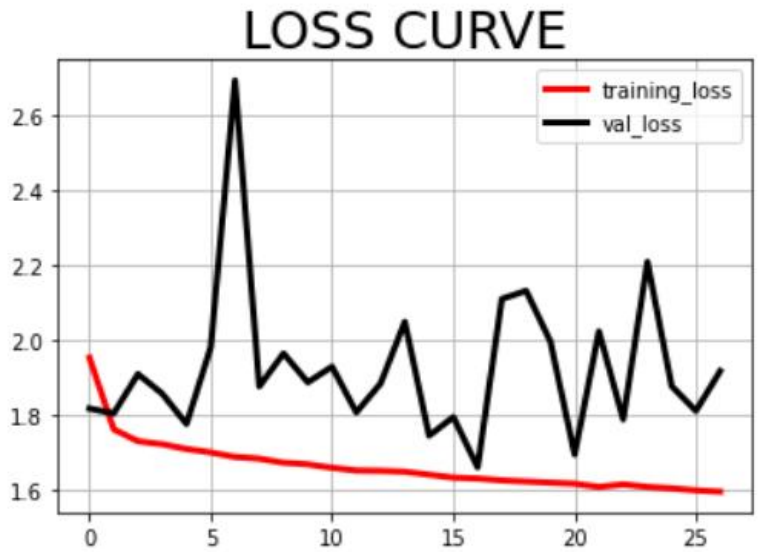
AI

- 'ReLu' is used in all the layers of the model except the output layer. In output layer, 'SoftMax' activation function is used.
- 'Batch Normalization' layer is also added in order to deal with the problem of internal co-variate shift which leads to loss of learning capacity and accuracy.
- 'Dropout' is another add on layer used which randomly drops units just to prevent the model from overfitting.
- Batch Size is set to 32.
- Categorical cross-entropy loss function is used and Adam optimizer is used to compile the model.
- The evaluation metric focused on is 'accuracy' of the model to judge its performance.
- Images are augmented and normalized as neural networks are highly sensitive towards non-normalized data.
- All the models are being trained for 100 epochs.
- 'Early Stopping' is a regularization technique for deep neural networks that stops training when parameter updates no longer begin to yield improves on a validation set & 'ReduceLROnPlateau' reduces learning rate when a metric has stopped improving. (These two are the callback function used in model creation)

ResNet50

The first stack of layer of this model consist of 50 pre-trained layers of ResNet50 architecture followed by one flattening layer and 3 fully connected layers of 512 units, 256 units and 7 units respectively. Last layer is the output layer with 7 units because the number of classes the model is working on is 7.

Following are the loss and accuracy plots of the model (model A):



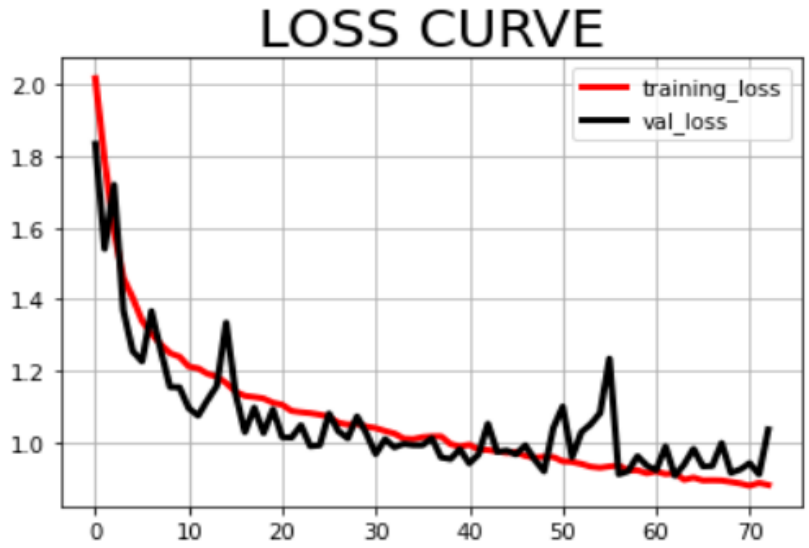
Continued….

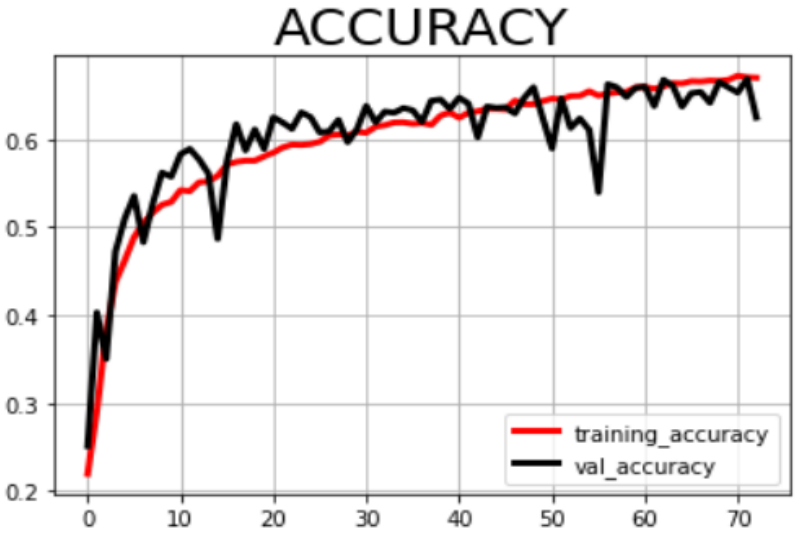
Anyway the model failed to impress by its performance as it only managed to achieve 37.21% training accuracy and 37.56% validation accuracy which is really very less. So, this model was not at all appropriate to be used in real time scenario.

So, moving forward to build the second model which is a custom model using CNN architecture to check if that manages to achieve satisfactory accuracy.

CNN (1st CNN model)

This model (model_B) is the first model created using the CNN architecture which has got 4 CNN layer of 64 units, 128 units, 512 units, 512 units respectively followed by one flattening layer and 3 fully connected layer of 256 units, 512 units and 7 units respectively.
Following are the loss and accuracy plots of the model (model_B):





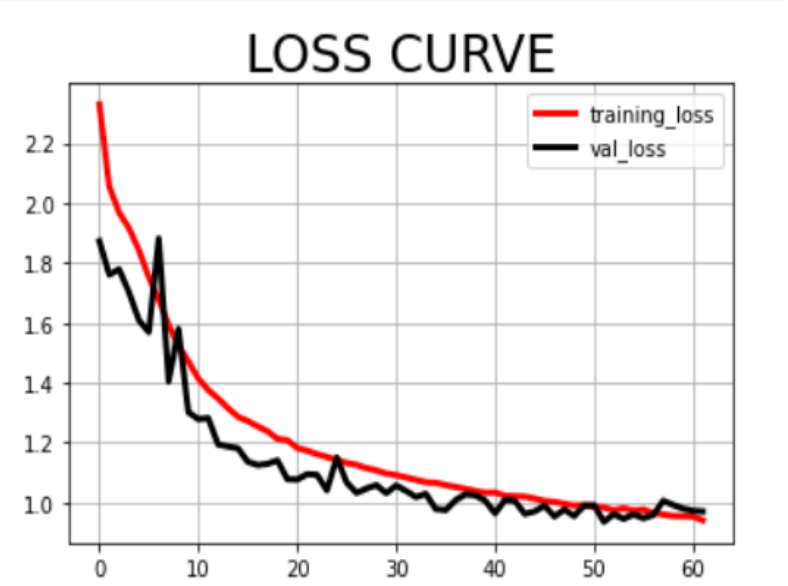
Continued….

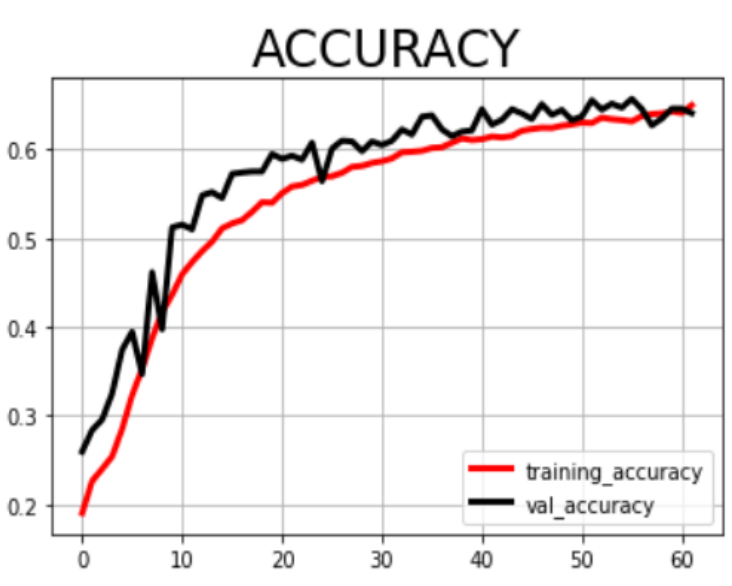
The performance of this model was satisfactory with a training accuracy of 66.98% and validation accuracy of 66.75%. It was trained using the default Adam learning rate which 0.001.

Before finalising the model and saving it for application building, I thought of creating another CNN model which is exactly the same as this model but only difference is that the second model is trained using 0.0001 learning rate instead of default learning rate (0.001) just to check whether tweaking the learning rate helps in increasing the accuracy of the model.

CNN (2nd CNN model)

It is the second CNN model (model_C) which has got the same architecture as the 1st CNN model but trained using 0.0001 learning rate.
Following are the loss and accuracy plots of the model (model_C):





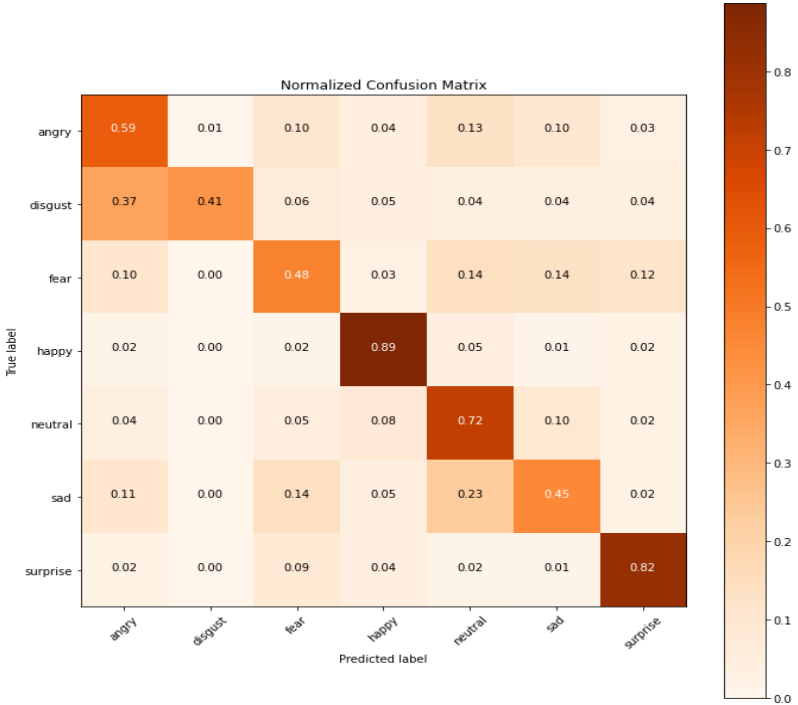
Continued….

This model too performed well and also very close to the 1st CNN model with a training accuracy of 64.96% and validation accuracy of 65.58%.

However, this model was not able to beat the accuracy of the previous CNN model and hence 1st CNN model named as ‘model_B’ was the best performing model out all the three models and saved for further use.

Model Evaluation

The best performing model (model_B) was evaluated on the basis of accuracy and confusion matrix. Following is the confusion matrix of 'model_B':



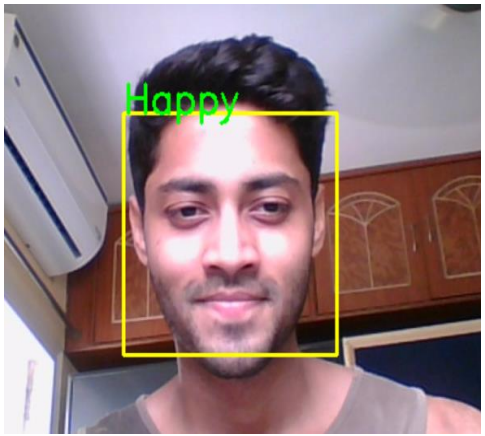
- Accuracy of model_B is 66.75% which is satisfactory.
- Observing the confusion matrix, it can be stated that the model is quite good at predicting 'happy' and 'surprised' faces whereas fails to predict 'disgust' and 'fear' faces appropriately. This may be due to biased dataset and similarity of 'fear' and 'disgust' emotion.

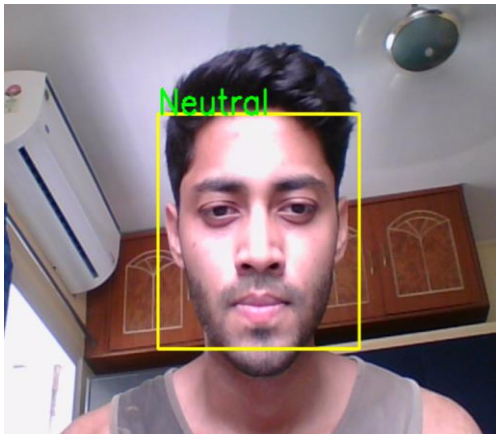
Real Time Face Emotion Detection

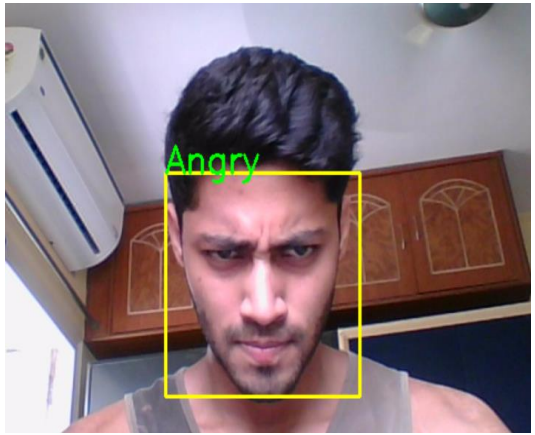
- A program is created for detecting and predicting the emotion for single as well as for the multiple faces with the help of OpenCV video capture in local system.
- Streamlit-webRTC is used for the creation of front end web application.
- The created web application is then deployed in Streamlit cloud.

Streamlit app : https://share.streamlit.io/arka1212/live-class-monitoring-system-face-emotion-recognition/main/app.py

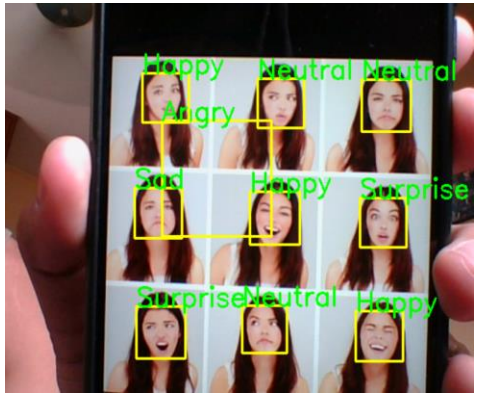
Following are the demo samples of face emotion recognition system:

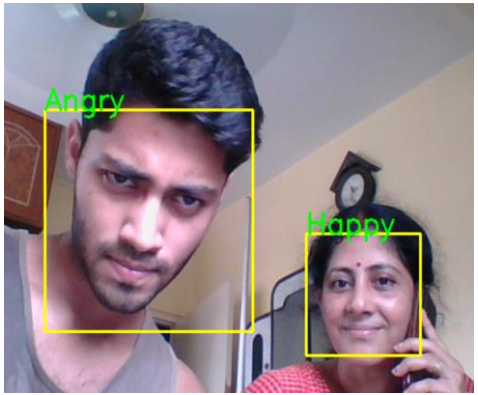


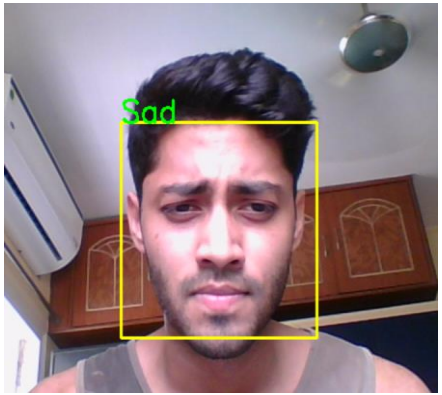


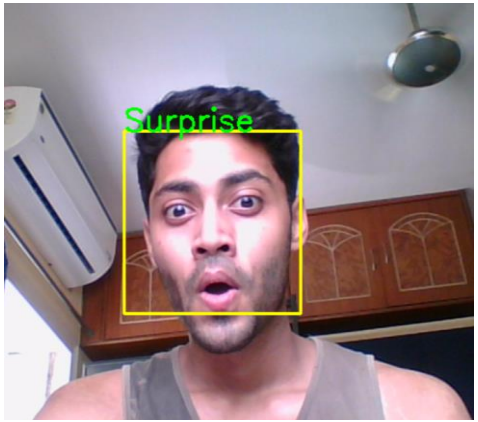


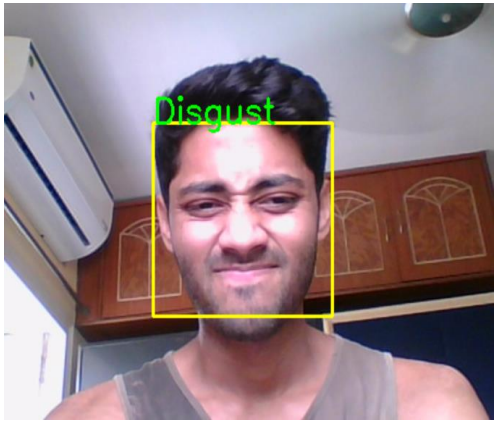
Continued....


Happy
Neutral
Neutral
Angry
Sad
Happy
Surprise
Surprise
Neutral
Happy


Angry
Happy


Sad


Surprise


Disgust

Challenges Faced

- Models took a lot of time to train with a limited GPU access on Google colab notebook.
- Took long time to experiment with different models and find the optimal one.
- In local machine, version issues were causing troubles leading to numerous attempts of installation and uninstallation.
- Faced a lot of unknown error while running the web application created using ‘Streamlit’.
- Deployment was tricky and need to research a lot about the dependencies and the code structure in order to get everything in place.
- Accessing webcam in local machine needed a bit of extra research.

Conclusions

- Three models are built in this project which are model_A, model_B, model_C.
- First model (model_A) is trained using 'ResNet50' architecture which failed to get good accuracy which is 37.21% & 37.56% as training and validation accuracy respectively.
- Second model (model_B) is trained using CNN architecture which gave a quite impressive result with an accuracy of 66.98% & 66.75% as training and validation accuracy respectively.
- Third model (model_C) is exactly the same as second model but the only difference is its trained using 0.0001 learning rate instead of default learning rate (0.001). This model also managed to perform well with an accuracy of 64.96% & 65.58% as training and validation accuracy respectively.
- 'model_B' performed the best and was saved and later used in creating the web application.

Thank You