Capstone Project - 5

LIVE CLASS MONITORING SYSTEM

(Face Emotion Recognition)

Individual Project By:

Anish Johnson.

Contents:

- Introduction
- Problem Statement
- Dataset used
- Data Preprocessing
- Model Building and Evaluation
- Real Time Facial Emotion Detection
- Model Deployment
- Conclusions

Introduction:

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

Digital platforms might **overpower physical classrooms in terms of content quality**, but in a physical classroom, a **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.**

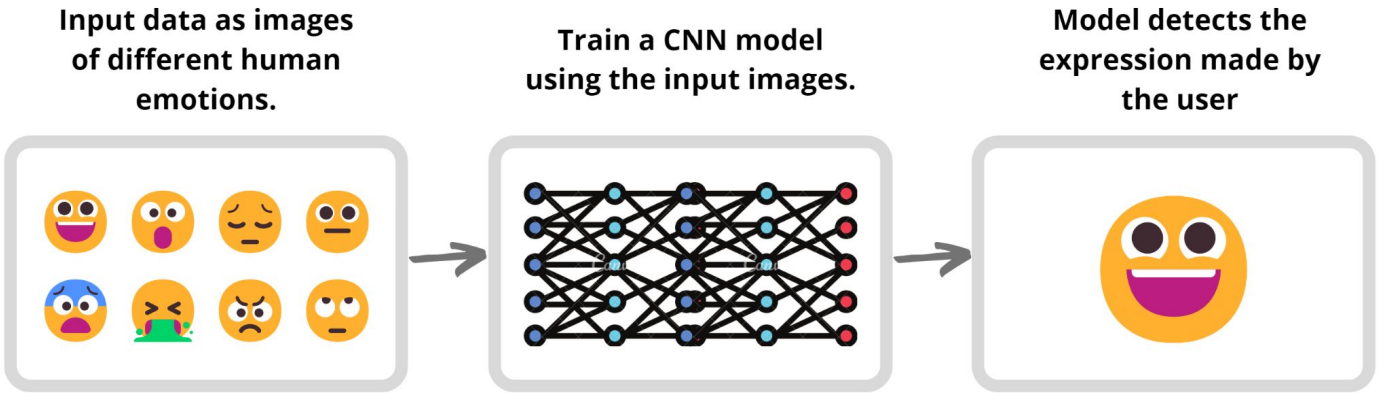
While digital platforms have limitations in terms of physical surveillance, it comes with the **power of data and machines**, which can work for you.

It provides **data in form of video, audio, and texts**, which can be analyzed using deep learning algorithms. A deep learning-backed system not only solves the surveillance issue, but also removes the human bias from the system, and all information is no longer in the teacher's brain but rather translated into numbers that can be analyzed and tracked.

Problem statement:

We aim to solve one of the challenges faced by digital platforms by applying deep learning algorithms to live video data.

We do this by recognizing the facial emotions of the participants using the CNN model we create which will categorize the observed emotion accordingly.



Dataset used:

We have utilized the FER 2013 dataset provided on Kaggle.

The data consists of 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.

The training set consists of 28,709 examples and the public test set consists of 3,589 examples.

Data Preprocessing:

Data Pipeline

1 - GET THE DATA

First we download the required data from Kaggle and unzip it for further analysis.

2 - DEFINE THE DATASETS

Now that we have downloaded our dataset, lets separate it into TRAIN and VALIDATION sets.

3 - RESCALE THE DATA

Before we proceed we need to rescale our data by multiplying it to 1/255.
This is done so we target values between 0 and 1.

4 - APPLY DATA AUGMENTATION

Data augmentation is a technique to artificially create new training data from existing training data. It helps us to increase the size of the dataset and introduce variability in the dataset.

5 - TIME TO BUILD OUR MODEL

Now that we have preprocessed our data, we can start building a neural network to detect those emotions.

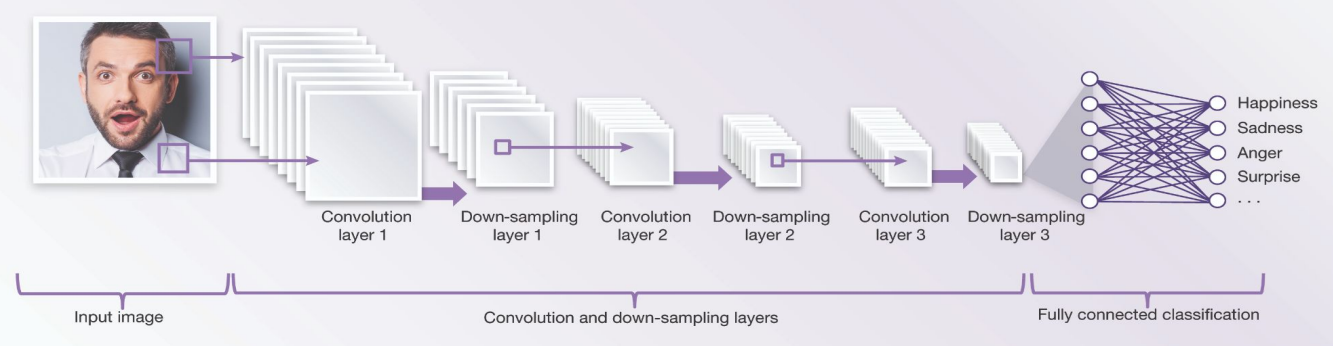
Model Building and Evaluation:

Convolutional Neural Network (CNN):

A **neural network** is a way for a computer to process data input. They're inspired by biological processes found in human and animal brains. Neural networks are comprised of various layers of 'nodes' or 'artificial neurons'. Each node processes the input and communicates with the other nodes.In this way, input filters through the processing of a neural network to create the output, or answer.

Convolutional neural networks were inspired by animal vision. The way the nodes in a CNN communicate with each other resembles the way some animals see the world.
So, rather than taking everything in as a whole, small areas of an image are taken.
And these small areas overlap to cover the whole image.

What happens inside an CNN?



The different layers in CNN

1 - Input Layer

An image is fed into the model as an input.

2 - Convolutional Layers

Instead of looking at the whole picture at once, it scans it in overlapping blocks of pixels.
In simple terms, the filters assign a value to the pixels that match them. The more they match, the higher the value.

3 - Rectified linear unit (ReLU)

ReLU allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as activation, because only the activated features are carried forward into the next layer.

4 - Pooling Layers

Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

5 - Fully Connected Layers

In a fully connected layer, every node receives the input from every node in the previous layer.
This is where all the features extracted by the convolutional neural network get combined.

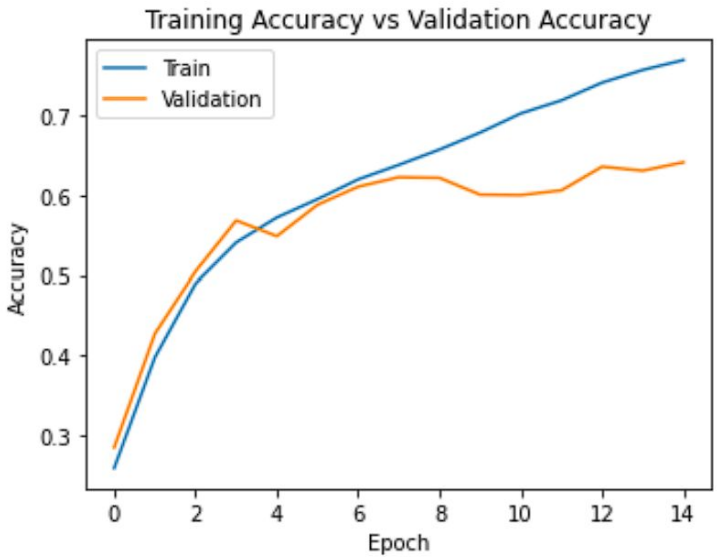
Custom CNN model for recognizing facial emotions:

To achieve our aim to recognize the facial emotions of a user, we created a CNN model using Keras and Tensorflow libraries.

The neural network contained an input layer, four hidden layers, and three fully-connected layers to process the image data and predict the emotions.

In between each layer, a Max Pooling and Dropout layer was added for downsampling the data and preventing our model from overfitting.

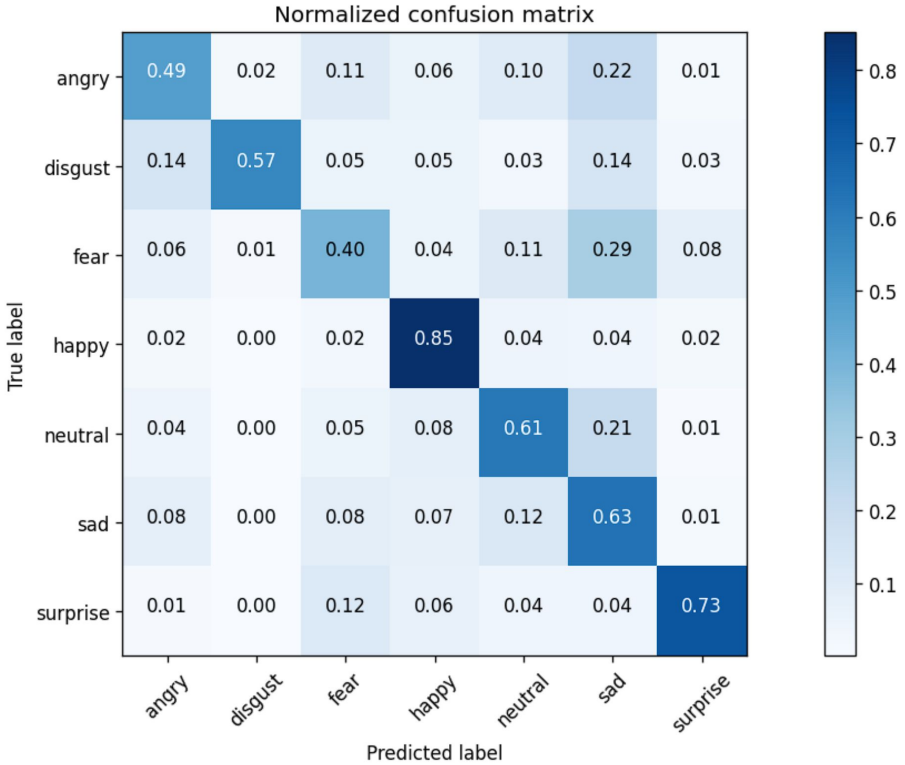
Plots for accuracy and loss.



- The created model was run for 15 epochs in order to get an accuracy of 77% on training set and 64% on validation set.
- The training and validation loss was reduced to 0.6325 and 1.1195.

Confusion Matrix.

From the confusion matrix, we saw; that the model accurately predicts most classes, but the performance is comparatively lower in classes angry and fear. Less amount of data present for these classes might be the reason for this.



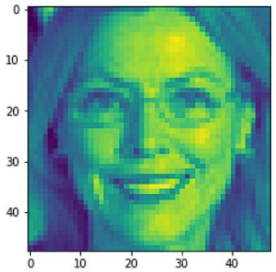
Testing the created model.

Finally, the model was tested using an image and it was successfully capable to recognize the facial emotion displayed in the image to be happy.

Test the model:

```
img = image.load_img("/content/train/happy/Training_1018317.jpg",target_size = (48,48),color_mode = "grayscale")
img = np.array(img)
plt.imshow(img)
print(img.shape)
```

```
(48, 48)
```




```
[263] label_dict = {0:'Angry',1:'Disgust',2:'Fear',3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}
```

```
[264] img = np.expand_dims(img,axis = 0) #makes image shape (1,48,48)
      img = img.reshape(1,48,48,1)
      result = model.predict(img)
      result = list(result[0])
      print(result)
```

```
[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
```

```
img_index = result.index(max(result))
print(label_dict[img_index])
plt.show()
```

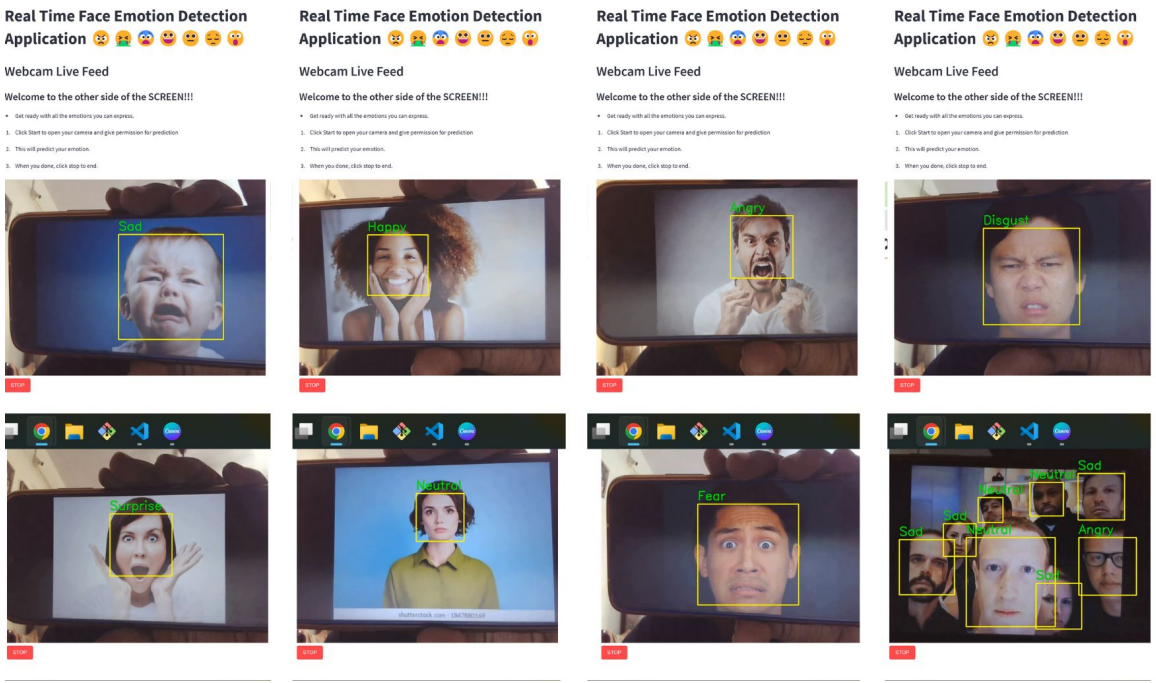
```
Happy
```

Real Time Facial Emotion Detection:

The CNN model we created accurately predicted the seven different emotions we had trained it.

It was also capable of detecting multiple faces and their respective emotions successfully.

The face detection was done using OpenCV, and the output was displayed on the webpage using Streamlit package.



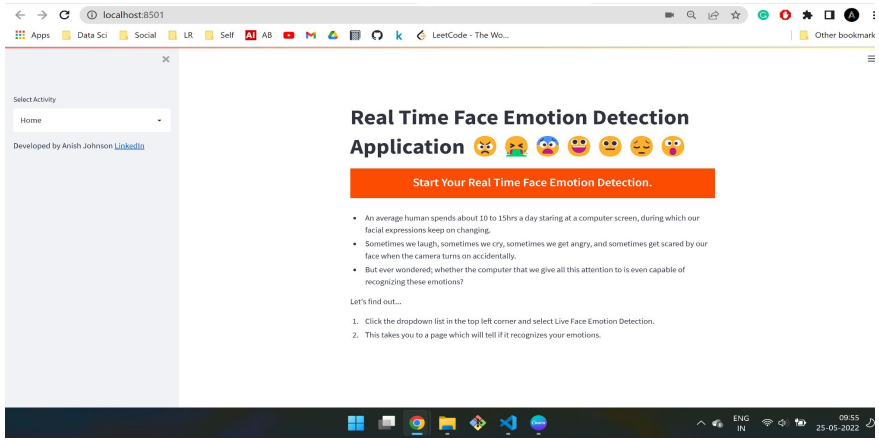
Model Deployment:

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science.

It allows you to write an app the same way you write a python code and makes it seamless to work on the interactive loop of coding and viewing results in the web app.

Our model has been deployed using streamlit and can be accessed using the below link.

https://share.streamlit.io/anishjohnson/face-emotion-recognition/main



Conclusions:

- This project explores the FER 2013 dataset provided by Kaggle to solve one of the issues faced by digital learning by creating a Live Class Monitoring System using CNN.
- We successfully achieved our objective and created a model capable of recognizing seven different classes of emotions mentioned in the dataset.
- For it to be accessible by others, a web app was created using OpenCV and Steamlit library.
- Both locally deployed app and the webapp can recognize the facial emotions of a user using the webcam of his/her laptop or computer.
- The above mentioned webapp can be accessed using the below link: https://share.streamlit.io/anishjohnson/face-emotion-recognition/main

Challenges Faced:

- Creating and testing a neural network is time consuming when you don't have the right tools, our model created for recognizing facial emotions required a gpu server in order to process the results faster, free gpu servers available on Google Colab had time constraints and often crashed when running for high number of epochs.

- Hyperparameter tuning for neural networks is a tedious task as it requires fast gpu's to run multiple iterations which on a normal computer/server might take days or weeks.

- Once the model is created, deploying it successfully is bit difficult since most of the popular cloud servers that offer these kind of services require certain prerequisites.

Future Scope:

- Face detection may be used for a broad range of purposes, from defense to ads. Any examples in usage include Smartphone makers, including Apple, for public protection.
- Law enforcement by gathering mugshots can evaluate local, national, and federal assets repositories too.
- Business protection, as businesses may use facial recognition to access their buildings.
- Marketing, where advertisers may use facial recognition to assess particular age, gender, and ethnicity.

Dependencies:

The following python libraries were used to successfully build and deploy our model:

Important Links:

Github Repository link:
https://github.com/anishjohnson/Face-Emotion-Recognition.git

Streamlit web app link:
https://share.streamlit.io/anishjohnson/face-emotion-recognition/main

Thank You.