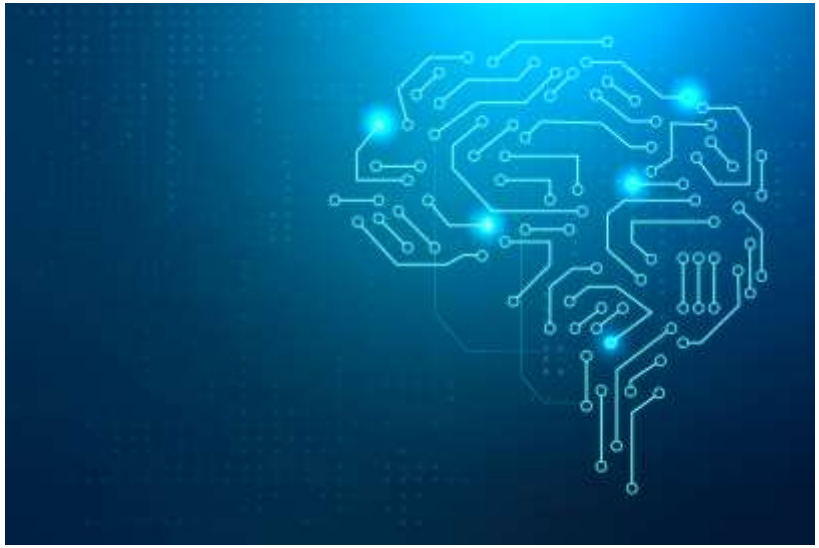
# Capstone Project - 5

# Live Class Monitoring System
## (Face Emotion Recognition)

Kiran Mamtani

# Content

- Introduction
- Defining problem statement
- Dataset overview & Data Pre-processing
- CNN model
- Model Deployment
- Real Time Facial Emotion Detection
- Conclusion
- Challenges



Source: https://www.freepik.com/

# Introduction

- 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
- Global E-learning is estimated to witness an 8X over the next 5 years to reach USD 2B in 2021. India is expected to grow with a CAGR of 44% crossing the 10M users mark in 2021
- Although the market is growing, one of many challenges is how to ensure quality learning for students
- In a physical classroom during a lecturing teacher can see the faces and assess the emotion of the class and tune their lecture accordingly, whether he is going fast or slow
- Digital platforms have limitations in terms of physical surveillance but it comes with the power of data and machines which can work for you
- Deep learning backed system not only solves the surveillance issue, but it also removes the human bias from the system

# Defining the problem statement

Our objective is to solve the mentioned challenge by applying deep learning algorithms to live video data in order to recognize the facial emotions and categorize them accordingly

Source: https://www.freepik.com/

# Dataset Overview

- We have utilized the FER 2013 dataset provided on Kaggle. The data consists of 48x48 pixel grayscale images of faces
- The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral)

# Data Pre-processing

- Rescaled the data between values 0 to 1 by multiplying with 1/255
- Applied data augmentation technique to create a new training data from existing training data. It helps us to increase variability in the dataset
- We have total 35887 images in our dataset out of which 28709 images belongs to training set and 7178 images belongs to testing set

# CNN Model

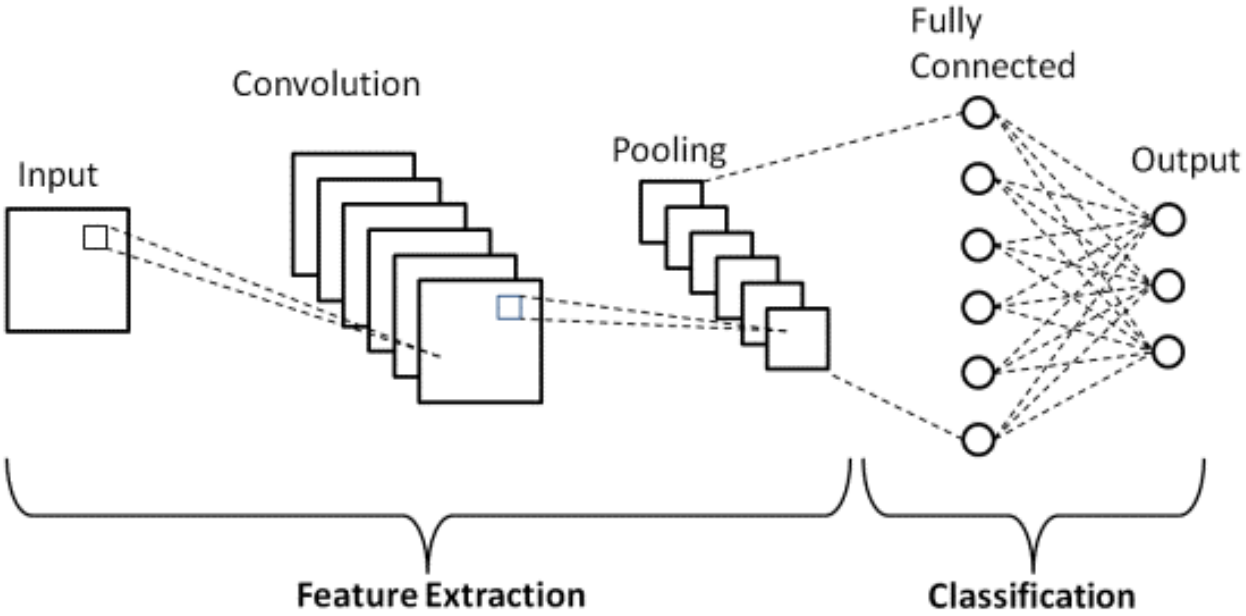
## What is Convolutional Neural Network (CNN)?

- A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

- A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

# CNN Model

## Architecture of Convolutional Neural Network (CNN)



# CNN Model

**Defining the model:** We will use Keras to create a Convolutional Neural Network. This network will have the following components:

- **Convolutional Layers:** This layer is the first layer that is used to extract the various features from the input images. These compute dot product between their weights and the small regions to which they are linked

- **Pooling Layers:** In most cases, a Convolutional Layer is followed by a Pooling Layer. These layers will down sample the operation along the dimensions. This helps reduce the spatial data and minimize the processing power that is required. In our project we will use *Max Pooling*.

- **Activation functions:** are those functions which are applied to the outputs of all layers in the network. In this project, we will resort to the use of two functions— *Relu* and *Softmax*.

- **Dense layers:** These layers are present at the end of a C.N.N. They take in all the feature data generated by the convolution layers and do the decision making.

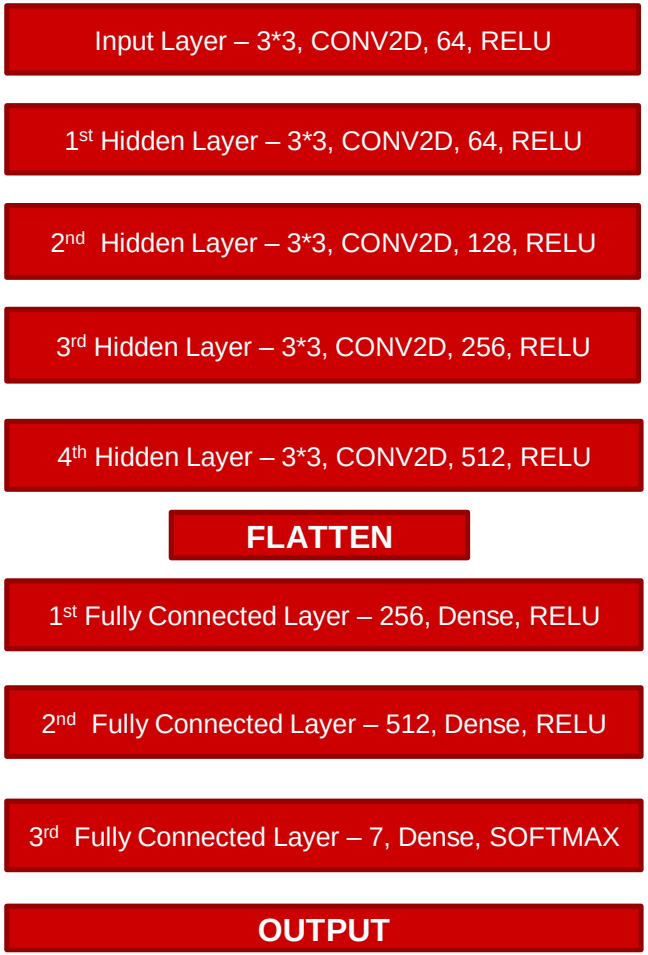
- **Dropout Layers:** randomly turns off a few neurons in the network to prevent overfitting.

- **Batch Normalization:** normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This speeds up the training process

# CNN Model

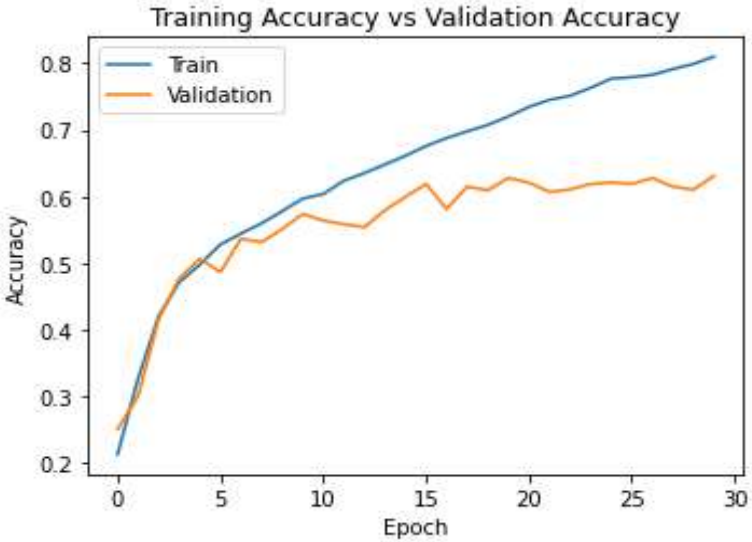
## Our model for face emotion detection

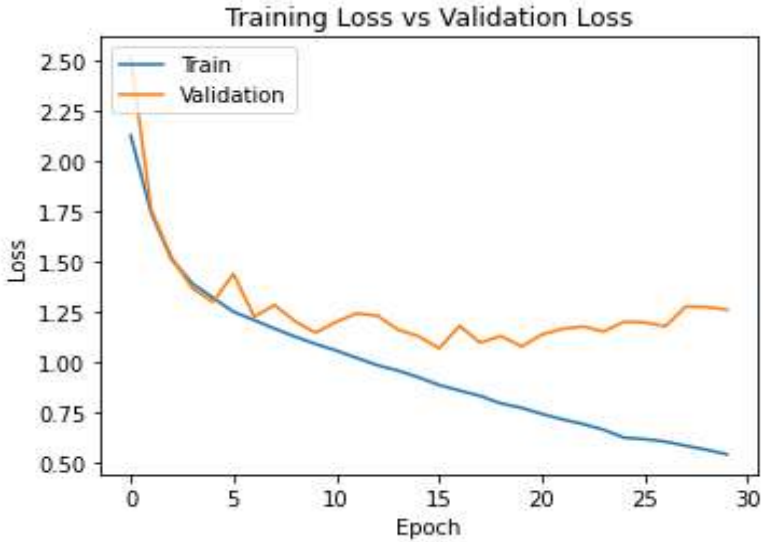
- We created a CNN model using Keras and Tensorflow libraries
- The neural network have input layer which is the first layer of our model. Following by four hidden layers and three fully connected layers. This will process the image and give the output.
- BatchNormalization is used in all the hidden layers and fully connected layers. MaxPooling is used in all the hidden layers. Dropout 0.25 is used in all the hidden layers and dropout 0.30 is used in fully connected layers.
- After the 4th hidden layer, flatten layer is used to convert all the resultant 2-D arrays from pooled feature maps into a single long continuous linear vector.
- As this is the multi-class problem, Softmax activation function is used in the last layer to get the final output



# CNN Model

## Accuracy and Loss Visualization



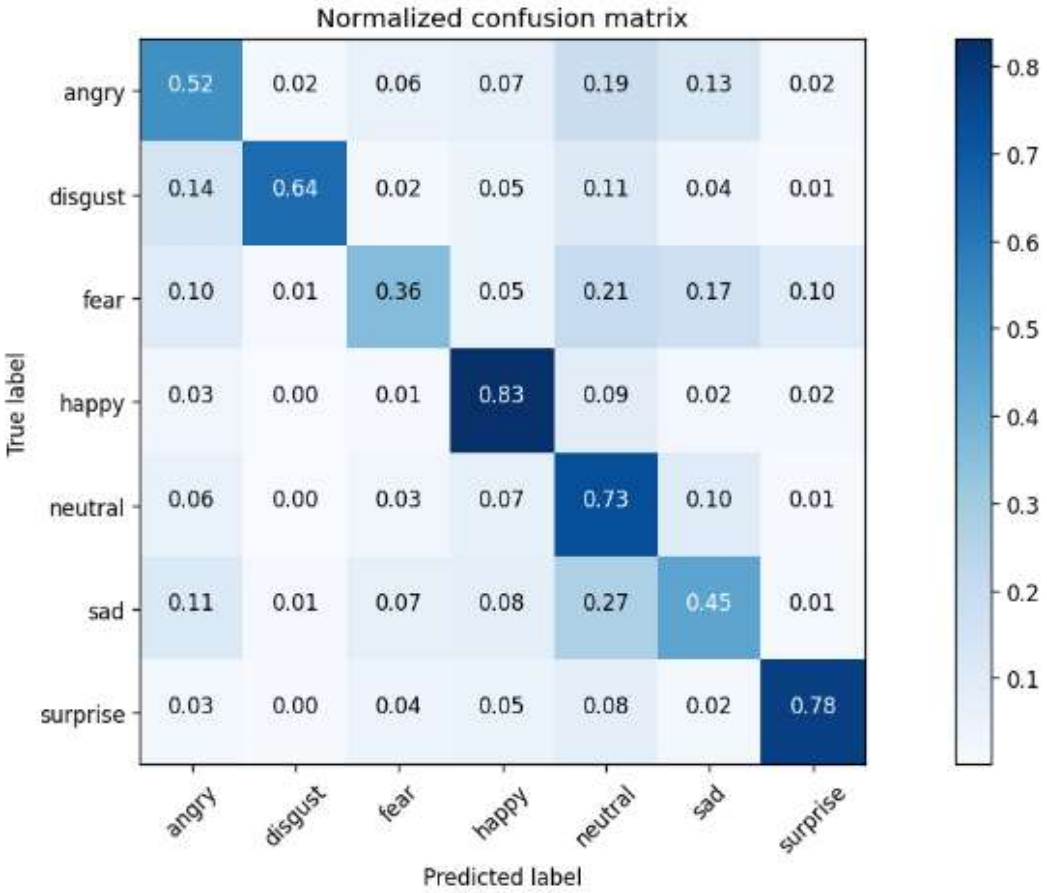




The model was fitted and we have run till 30 epochs. Got accuracy of 80.96% in training data & 63.09% in testing data. The loss also decreased from 2.12 to 0.54 in training data and from 2.51 to 1.27 in testing data

# CNN Model

## Confusion Matrix

- With the matrix, we can analyze why the model performs poorly on ‘fear’ by looking at the row of confusion matrix
- Ah, it is clear now. Most ‘fear’ faces are incorrectly predicted as angry, sad or neutral.



# CNN Model

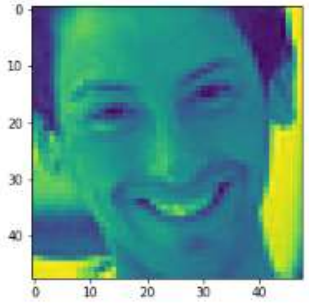
## Let’s evaluate our model

- Finally the model was tested on an  image and it was successful in prediction of the face emotion which is ‘Happy”

### Let's test our model

```
img = image.load_img("/content/train/happy/Training_371241.jpg",target_size = (48,48),color_mode = "grayscale")
img = np.array(img)
plt.imshow(img)
print(img.shape)
```

```
(48, 48)
```




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

```
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)
```

```
[1.6149664e-20, 2.439993e-36, 0.0, 1.0, 1.8555509e-33, 4.200219e-19, 1.3918004e-35]
```

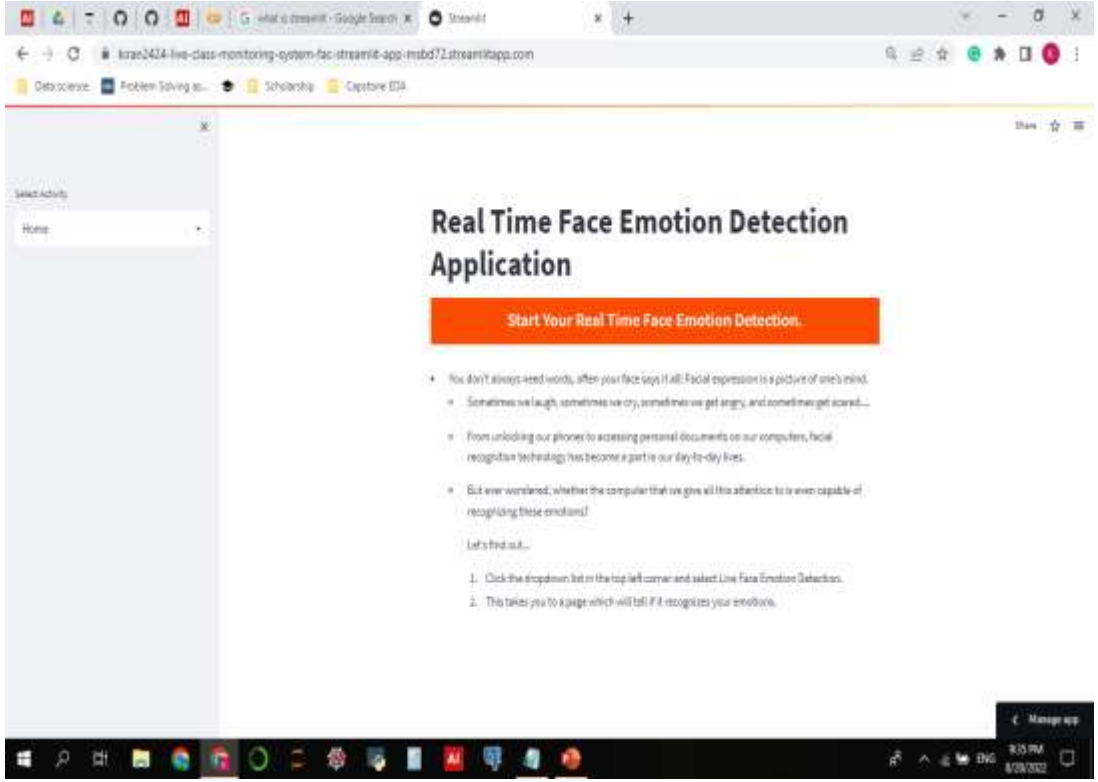
```
img_index = result.index(max(result))
print(label_dict[img_index])
plt.show()
```

```
Happy
```

# Model Deployment

## On Streamlit Cloud

- Streamlit is **a free and open-source framework** to rapidly build and share beautiful machine learning and data science web apps
- It is a Python-based library specifically designed for machine learning engineers
- This model is deployed on Streamlit and the link is accessible here – https://kiran2424-live-class-monitoring-system-fac-streamlit-app-msbd72.streamlitapp.com/
- Our model is also deployed on AWS cloud using PuTTY tool



# Real Time Facial Emotion Detection

- Here we have seven different categories in our model which are Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise.
- Applied in our app and here is the result of live webcam
- Most of the prediction was correct but still few images was undetectable
- Here are some examples…

# Conclusion

- We started by uploading the FER 2013 dataset from kaggle to solve out problem statement of recognising the emotions in live webcam.
- Then we splitted our data into training and testing sets followed by converting it between 0 to 1 pixel size by multiplying with 1/255.
- We build a CNN model which is capable of recognition the facial expression of the user.
- The model obtained accuracy of 80.96% in training data & 63.09% in testing data and the loss also decreased from 2.12 to 0.54 in training data and from 2.51 to 1.27 in testing data.
- From the confusion matrix we saw that our model does a good job in predicting most of the classes but the performance is comparatively lower in class fear. Most ‘fear’ faces are incorrectly predicted as angry, sad or fearful. .
- Finally, we tested the model by using an image from the dataset and it mostly predicted it correctly but still our model need to retrain which is only possible in high GPU.

# Challenges

- The most difficult thing I faced in this project was fitting the model and running epochs because it was too time consuming because of low GPU and accuracy was also low in the first few trials of CNN model. But I kept changing few things in the model and finally got the acceptable accuracy

- Second most difficult thing I faced was to deploy the project on streamlit. I had done coding using colab notebook because of google GPU. Making local environment was easy using anaconda but when I tried deploy it on streamlit, I was geting error again and again due to requirements file and the versions. But I surfed on google and tried few things which finally deployed my project on Streamlit.

# Thank You!