Capstone Project
Face Emotion Recognition

Name- Pranav Singhal
Batch-AlmaBetter Pro

Github- https://github.com/pranav168/Emotion-Detection

Problem Description

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.

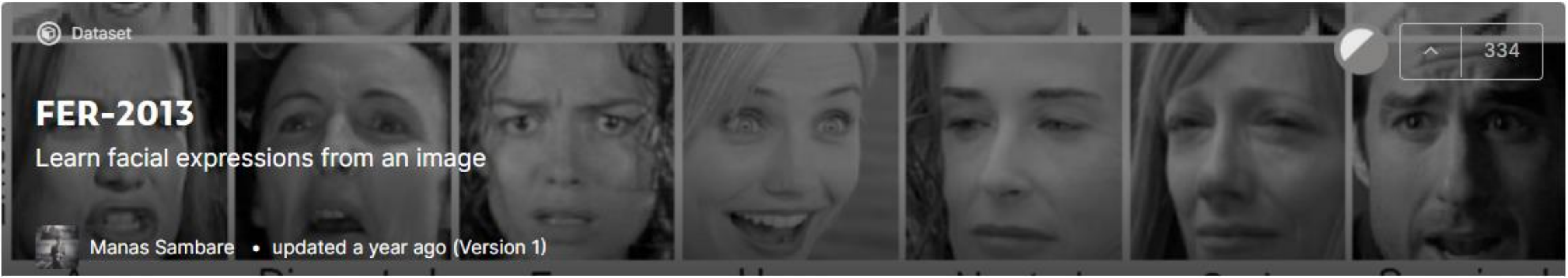
In a physical classroom during a lecturing teacher can see the faces and assess the emotion of the class and tune their lecture accordingly, whether he is going fast or slow. He can identify students who need special attention. Digital classrooms are conducted via video telephony software program (exZoom) where it’s not possible for medium scale class (25-50) to see all students and access the mood. Because of this drawback, students are not focusing on content due to lack of surveillance

How can Face Emotion Recognition with Deep Learning help?

While digital platforms have limitations in terms of physical surveillance but it comes with the power of data and machines which can work for you. It provides data in the form of video, audio, and texts which can be analysed using deep learning algorithms. Deep learning backed system not only solves the surveillance issue, but it also removes the human bias from the system, and all information is no longer in the teacher's brain rather translated in numbers that can be analyzed and tracked.

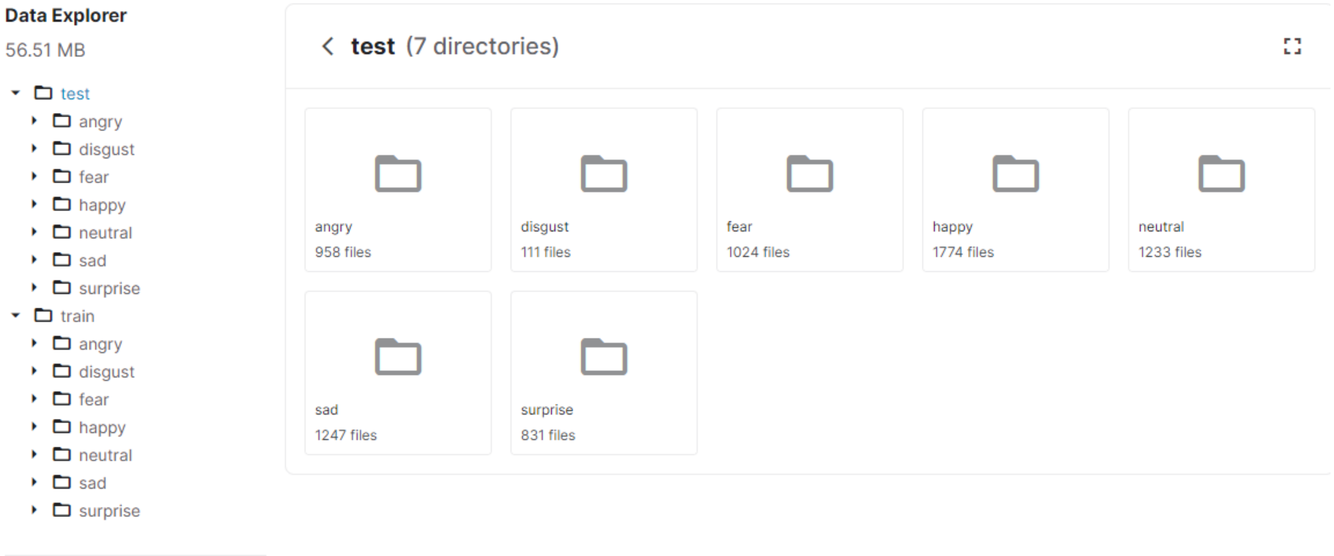
Data – FER2013

Kaggle Link- https://www.kaggle.com/msambare/fer2013



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 task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples and the public test set consists of 3,589 examples.



Data Structure : I would be creating a validation Data set apart from these test and train sets

Loading Data in Notebook

```
training_gen=ImageDataGenerator(rescale=1./255)                                                    # Crea
ting Image generators, for all train, validation, and test set
validation_gen=ImageDataGenerator(rescale=1./255)
testing_gen=ImageDataGenerator(rescale=1./255)
```

```
train_gen_3ch=training_gen.flow_from_directory('archive/train',                   # Creating  Training Dataset with 3 channe
ls for plotting it and check the labelling
                                           target_size=(48,48),
                                           batch_size=32,
                                           class_mode='categorical')
train_gen=training_gen.flow_from_directory('archive/train',                       # Creating  Training Dataset
                                           target_size=(48,48),
                                           batch_size=32,
                                           color_mode='grayscale',
                                           class_mode='categorical')
valid_gen=validation_gen.flow_from_directory('archive/validation',                # Creating Validation Set
                                           target_size=(48,48),
                                           batch_size=32,
                                           color_mode='grayscale',
                                           class_mode='categorical')
test_gen=testing_gen.flow_from_directory('archive/test',                           # Creating Test Set
                                           target_size=(48,48),
                                           batch_size=32,
                                           color_mode='grayscale',
                                           class_mode='categorical',
                                           shuffle= False)
```

```
Found 28207 images belonging to 7 classes.
Found 28207 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
Found 502 images belonging to 7 classes.
```

Loading Data and Splitting it into train, test and Validation set and assigning Class to each of them on the basis of folder they are present in, here I am setting the target size as 48 X 48 and keeping only one channel out of 3 by providing color mode as Grayscale

Plotting Data

```
[ ] imgs,lables=next(train_gen_3ch)                                    # Extracting next batch to plot it, this batch of photos are selected randomly
    list_of_keys=['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']  # list of Classes
    def plotImages(images_arr):
      fig, axes= plt.subplots(1,32,figsize=(32,2))
      axes= axes.flatten()
      for img, ax in zip(images_arr,axes):
        ax.imshow(img)
        ax.axis('off')
      plt.tight_layout()
      plt.show()
    plotImages(imgs)
    for i in lables[:32]:                                              # Printing the lables below the respective image
      for num, j in enumerate(i):
        if j==1:
          print(list_of_keys[num], end='   ')                          # Labelling seems to be spot on
    num+=1
```

```
angry happy angry neutral fear sad fear happy neutral neutral happy fear fear angry happy happy angry surprise surprise angry neutral happy happy surprise angry fear fear sad surprise fear sad fear
```

Here I am Plotting one Batch of train set with their respective class labels, and by looking at the Images and their labels, I can conclude that Labeling is really very accurate.

CNN Model

```
cnn_model= Sequential([                                                                                                    # Usi
ng Keras Sequential model to build out CNN
    Conv2D(32, kernel_size=(3,3), activation='relu',input_shape=(48,48,1)),                                                # Fir
st Layer of convolutional Nural network, input shape is provided only at this instance.
    Conv2D(64, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2,2)),                                                                                            # Max
pool Layer
    Dropout(0.05),

    Conv2D(128, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Dropout(0.05),

    Conv2D(256, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Dropout(0.05),

    Conv2D(512, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2,2)),
    Dropout(0.05),

    Flatten(),                                                                                                             # Fla
ttening out all the layers
    Dense(1024, activation='relu'),
    Dense(7, activation='softmax')]
)
cnn_model.summary()
```

Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 256)         295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 512)         1180160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 7175
=================================================================
Total params: 2,100,487
Trainable params: 2,100,487
Non-trainable params: 0
_________________________________________________________________
```

Compiling and Fitting the Model

```
checkpoint = ModelCheckpoint('./my_model.h5', monitor='val_acc', verbose=1, save_best_only=True, mode='max')

early_stopping=EarlyStopping(monitor='val_loss',
                            min_delta=0.1,
                            patience=4,
                            verbose=2,
                            restore_best_weights=True)
decay_lr= ReduceLROnPlateau(monitor='val_loss',
                           factor=0.2,
                           patience=3,
                           verbose=1,
                           min_delta=0.0001)
callbacks=[early_stopping,checkpoint,decay_lr]
```

```
cnn_model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])                              # Com
piling the Model
cnn_model.fit(train_gen, epochs=20, verbose=2,validation_data=valid_gen, callbacks=callbacks)
```

Here I have used Adam Optimizer, and Loss to be Categorical cross entropy

For training the model to train dataset, a training and validation set is provided and callbacks like early stopping and decay learning rate are added for better training

Accuracy on train and Validation Set

```
Epoch 1/20
882/882 - 186s - loss: 1.7513 - accuracy: 0.2793 - val_loss: 1.5645 - val_accuracy: 0.3835
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 2/20
882/882 - 131s - loss: 1.4599 - accuracy: 0.4289 - val_loss: 1.3645 - val_accuracy: 0.4710
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 3/20
882/882 - 130s - loss: 1.2926 - accuracy: 0.5008 - val_loss: 1.2440 - val_accuracy: 0.5228
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 4/20
882/882 - 130s - loss: 1.1842 - accuracy: 0.5460 - val_loss: 1.1845 - val_accuracy: 0.5460
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 5/20
882/882 - 130s - loss: 1.1053 - accuracy: 0.5818 - val_loss: 1.1593 - val_accuracy: 0.5538
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 6/20
882/882 - 131s - loss: 1.0353 - accuracy: 0.6094 - val_loss: 1.1470 - val_accuracy: 0.5596
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 7/20
882/882 - 131s - loss: 0.9697 - accuracy: 0.6333 - val_loss: 1.1301 - val_accuracy: 0.5747
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 8/20
882/882 - 131s - loss: 0.9109 - accuracy: 0.6581 - val_loss: 1.1307 - val_accuracy: 0.5819
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 9/20
882/882 - 130s - loss: 0.8440 - accuracy: 0.6821 - val_loss: 1.1769 - val_accuracy: 0.5804
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 10/20
882/882 - 130s - loss: 0.7884 - accuracy: 0.7049 - val_loss: 1.1951 - val_accuracy: 0.5761
WARNING:tensorflow:Can save best model only with val_acc available, skipping.

Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
Epoch 11/20
882/882 - 130s - loss: 0.6017 - accuracy: 0.7768 - val_loss: 1.2628 - val_accuracy: 0.5921
Restoring model weights from the end of the best epoch.
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 00011: early stopping
```

- Maximum accuracy for training set is 70% while for validation set is only 60%
- Thus the model is under fitting.

Predicting Facial expression

```
predictions=cnn_model.predict(test_gen)
from sklearn.metrics import confusion_matrix, classification_report
cm=confusion_matrix(y_pred=np.argmax(predictions, axis=-1), y_true=test_gen.classes)
cm
```

```
array([[42,  1,  5,  9,  8,  8,  2],
       [30,  8, 11, 10,  1, 13,  2],
       [ 6,  0, 10,  2,  2,  8,  4],
       [ 0,  0,  3, 67,  2,  7,  1],
       [10,  0, 10, 13, 23, 20,  4],
       [10,  1,  6,  7, 11, 44,  1],
       [ 0,  0,  6,  6,  1,  2, 65]], dtype=int64)
```

```
from sklearn.metrics import accuracy_score
aoc=accuracy_score(y_pred=np.argmax(predictions, axis=-1),y_true=test_gen.classes)
aoc                                                                 # I am able to get an accuracy of only 50% lets try with a pretrain
ed model
```

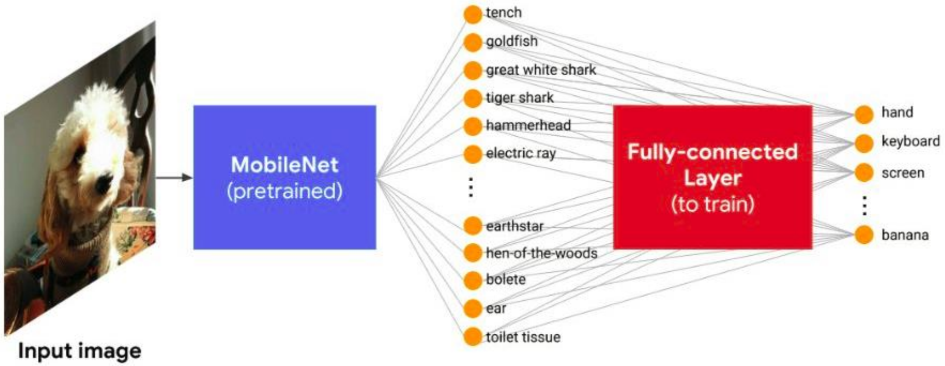
```
0.5159362549800797
```

```
cnn_model.save('my_model.h5')
```

Here on Predicting the Facial Expression of test data set it can be seen that Model is showing average performance as Accuracy score is only 51%

Transfer Learning

Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks.



Architecture of our image recognition model. We trained a fully-connected layer added to the pretrained MobileNet.

Loading Mobile Net Model and Making a few amendments to it

```
import random
random.shuffle(train_set)
```

```
X=[]
y=[]
for features, lables in train_set:
    X.append(features)
    y.append(lables)
X=np.array(X).reshape(-1,224,224,3)
```

```
from sklearn.model_selection import train_test_split
X_train, X_validation,y_train, y_validation= train_test_split(X,y,test_size=0.1)
```

Creating Train and test set yet again as Mobile Net architecture takes input of (None,224,224,3), thus keeping all 3 channels

```
final_output=layers.Dense(128)(base_output)
final_output=layers.Activation('relu')(final_output)
final_output=layers.Dense(128)(final_output)
final_output=layers.Activation('relu')(final_output)
final_output=layers.Dense(7, activation='softmax')(final_output)
```

```
final_output
```

```
<KerasTensor: shape=(None, 7) dtype=float32 (created by layer 'dense_2')>
```

```
final_model=keras.Model(inputs=base_input, outputs=final_output)
```

```
for layer in final_model.layers[:-23]:
    layer.trainable = False
```

Making Few amendments to the final layer of model, instead on classifying 1000 objects, now it will only be able to classify 7.

All I am freezing few weights and making them un-trainable

```
Conv_1 (Conv2D)                 (None, 7, 7, 1280)   409600      block_16_project_BN[0][0]
__________________________________________________________________________________________________
Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)   5120        Conv_1[0][0]
__________________________________________________________________________________________________
out_relu (ReLU)                 (None, 7, 7, 1280)   0           Conv_1_bn[0][0]
__________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          163968      global_average_pooling2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 128)          0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)          16512       activation[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 128)          0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 7)            903         activation_1[0][0]
==================================================================================================
Total params: 2,439,367
Trainable params: 1,231,943
Non-trainable params: 1,207,424
```

- Model Summary : As Few of the Parameters are at freeze, there are only some which are trainable

Model Performance

```
final_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001), metrics=['accura
final_model.fit(X_train,y_train, epochs=20, verbose=2,validation_data=(X_validation, y_validat

Epoch 1/20
794/794 - 406s - loss: 1.5786 - accuracy: 0.3765 - val_loss: 2.2255 - val_accuracy: 0.1581
Epoch 2/20
794/794 - 401s - loss: 1.3411 - accuracy: 0.4864 - val_loss: 2.0981 - val_accuracy: 0.2233
Epoch 3/20
794/794 - 401s - loss: 1.1818 - accuracy: 0.5559 - val_loss: 2.3961 - val_accuracy: 0.2205
Epoch 4/20
794/794 - 401s - loss: 1.0205 - accuracy: 0.6248 - val_loss: 1.7945 - val_accuracy: 0.3559
Epoch 5/20
794/794 - 403s - loss: 0.8426 - accuracy: 0.6978 - val_loss: 1.5884 - val_accuracy: 0.4325
Epoch 6/20
794/794 - 406s - loss: 0.6698 - accuracy: 0.7661 - val_loss: 1.8180 - val_accuracy: 0.4403
Epoch 7/20
794/794 - 400s - loss: 0.5158 - accuracy: 0.8242 - val_loss: 1.9420 - val_accuracy: 0.4172
Epoch 8/20
794/794 - 396s - loss: 0.3797 - accuracy: 0.8767 - val_loss: 2.0276 - val_accuracy: 0.4537
Epoch 9/20
794/794 - 396s - loss: 0.2783 - accuracy: 0.9123 - val_loss: 2.0764 - val_accuracy: 0.4782
Epoch 10/20
794/794 - 394s - loss: 0.2056 - accuracy: 0.9379 - val_loss: 2.5549 - val_accuracy: 0.4488
Epoch 11/20
794/794 - 395s - loss: 0.1700 - accuracy: 0.9480 - val_loss: 2.7138 - val_accuracy: 0.4619
Epoch 12/20
794/794 - 393s - loss: 0.1461 - accuracy: 0.9560 - val_loss: 2.6793 - val_accuracy: 0.4697
Epoch 13/20
794/794 - 395s - loss: 0.1244 - accuracy: 0.9619 - val_loss: 3.2451 - val_accuracy: 0.4169
Epoch 14/20
794/794 - 394s - loss: 0.1162 - accuracy: 0.9641 - val_loss: 2.9203 - val_accuracy: 0.4413
Epoch 15/20
794/794 - 395s - loss: 0.1090 - accuracy: 0.9657 - val_loss: 3.0480 - val_accuracy: 0.4307
Epoch 16/20
794/794 - 394s - loss: 0.1055 - accuracy: 0.9672 - val_loss: 3.0312 - val_accuracy: 0.4541
Epoch 17/20
794/794 - 394s - loss: 0.0931 - accuracy: 0.9721 - val_loss: 3.1519 - val_accuracy: 0.4225
Epoch 18/20
794/794 - 395s - loss: 0.0964 - accuracy: 0.9688 - val_loss: 3.1034 - val_accuracy: 0.4378
Epoch 19/20
794/794 - 394s - loss: 0.0972 - accuracy: 0.9681 - val_loss: 3.3477 - val_accuracy: 0.4495
Epoch 20/20
794/794 - 408s - loss: 0.0837 - accuracy: 0.9733 - val_loss: 3.5165 - val_accuracy: 0.4413
<keras.callbacks.History at 0x22e1d4f9580>
```

This Time Around the Model is overfitting as Training Accuracy is around 97.3 % while validation accuracy is only 44 % thus this model is not Robust enough and Model I made previously was better

Creating Web App Using Streamlit

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app.

Deploy Model Using AWS

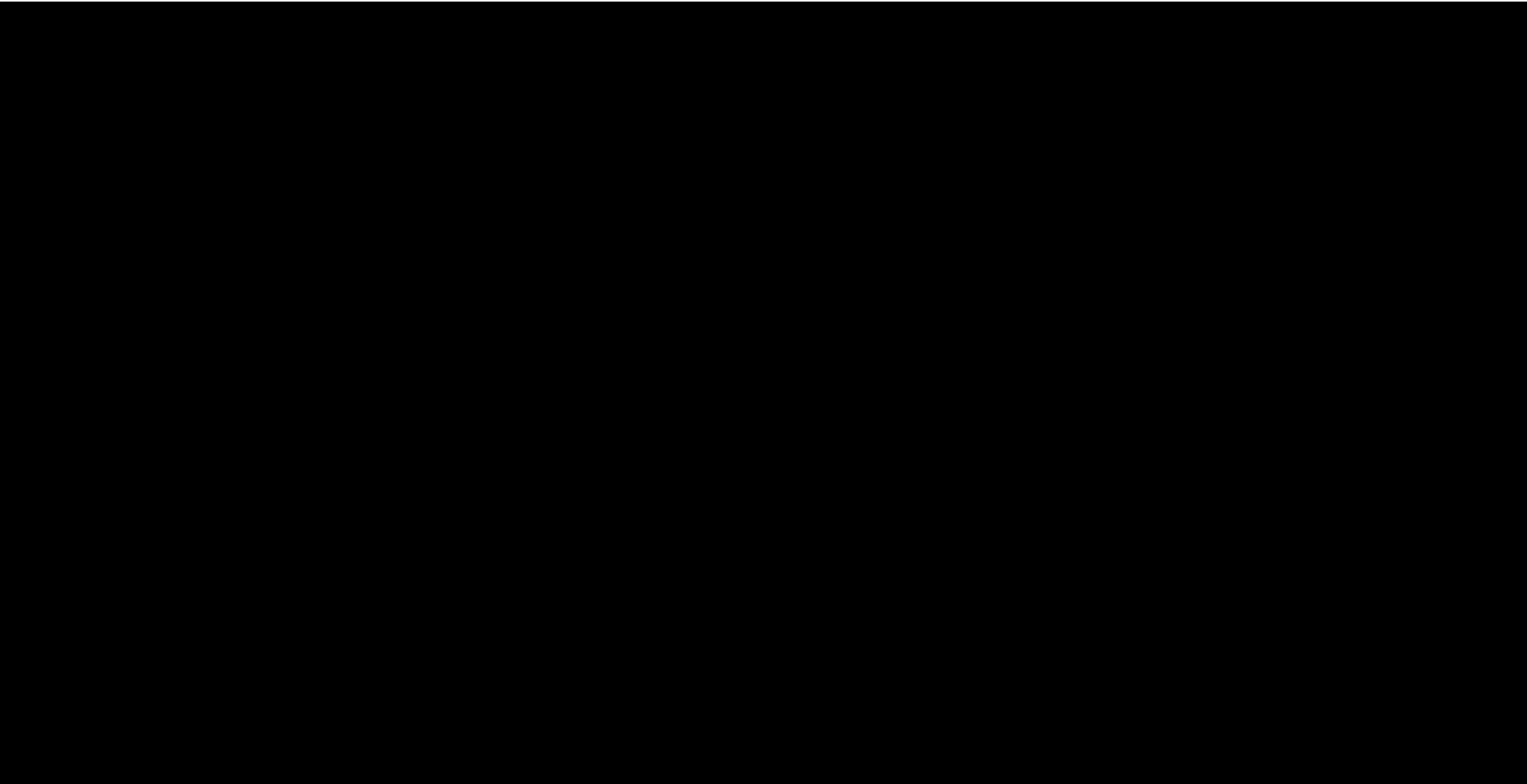
Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

Amazon Web Services (AWS) is the world’s most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS.

Face Emotion Recognition From Images

Face Emotion Recognition From Videos

Live Face Emotion Recognition



Conclusion

In the End we have a CNN Model which can detect the facial expressions, using which we can monitor the facial expression of all the students in the class simultaneously, using this feed teacher can assess the emotion of theclass and tune their lecture accordingly, whether he is going fast or slow. He can identify students who need special attention.

THANK YOU