STUDENT EMOTION DETECTION IN E-LEARNING

A project report in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

Submitted by

SAI CHARANI ARUMILLI (Reg.No:316506402036)

Under the guidance of

PROF. V. VALLI KUMARI
Professor
Department of CS&SE

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)

ANDHRA UNIVERSITY, VISAKHAPATNAM-530003

MARCH 2020

DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING
ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)
ANDHRA UNIVERSITY
VISAKHAPATNAM

CERTIFICATE

This is to certify that this is a bonafide project work entitled "**STUDENT EMOTION DETECTION IN E-LEARNING**", done by **Ms. SAI CHARANI ARUMILLI(Reg.No:316506402036)** student of Department of Computer Science & Systems Engineering, Andhra University College of Engineering(A) during the period 2016-2020 in the partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering**.

Prof. Kuda Nageswara Rao
Head of the Department
Dept of CS&SE
AUCE(A)
Visakhapatnam

Prof. V. Valli Kumari
Project Guide
Dept of CS&SE
AUCE(A)
Visakhapatnam

DECLARATION

I hereby declare that the project entitled **"STUDENT EMOTION DETECTION IN E-LEARNING"** has been prepared by me during the period November 2019 – March 2020 in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering**.

SAI CHARANI ARUMILLI
(Regd.No:316506402036)

Place: Visakhapatnam
Date:

ACKNOWLEDGEMENT

I have immense pleasure in expressing my earnest gratitude to my Project Guide **Prof. V. Valli Kumari,** Andhra University for her inspiring and scholarly guidance. Despite her pre-occupation with several assignments, she has been kind enough to spare her valuable time and gave us the necessary counsel and guidance at every stage of planning and constitution of this work. I express sincere gratitude for having accorded me permission to take up this project work and for helping me graciously throughout the execution of this work.

I express sincere Thanks to **Prof. K. Nageswara Rao,** Head of the Department Computer Science and Systems Engineering, Andhra University College of Engineering(A) for his keen interest and providing necessary facilities for this project study.

I express sincere gratitude to **Prof. P.V.G.D. Prasad Reddy,** Chairman, Board of Studies, Computer Science and Systems Engineering, Andhra University College of Engineering(A) for his keen interest and for providing necessary facilities for this project study.

I express sincere Thanks to **Prof. P. Srinivas Rao**, Principal, Computer Science and Systems Engineering, Andhra University College of Engineering(A) for his keen interest and for providing necessary facilities for this project study.

I extend our sincere thanks to our academic teaching staff and non-teaching staff for their help throughout our study.

CONTENTS

ABSTRACT

Emotions of a student during course engagement play a vital role in any learning environment whether it's in classrooms or in e-learning. Emotion detection methods have been in focus of the researchers across various disciples to understand the user involvement, effectiveness and usefulness of the system implemented or to be implemented. Our focus is on understanding and interpolating the emotional state of the learner during a learning engagement.

Evaluating the emotion of a learner can progressively help in enhancing learning experience and update the learning contents. In this project, I propose a system that can identify and monitor emotions of the student in an e-learning environment and provide a real-time feedback mechanism to enhance the e-learning aids for a better content delivery.
This helps to identify emotions and classify learner involvement and interest in the topic which are plotted as feedback to the instructor to improve learner experience.

In-order to identify students emotions, a deep learning model inspired from Mini-Xception model is built and is trained with FER-2013 dataset which contains 35,887 grayscale images where each image belongs to one of the following classes {“angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”}. To deal with real-time content, avoiding large number of parameters is the most important thing. So, I adopted Mini-Xception model which replaces the fully-connected layer that has more parameters, with global average pooling. Here, I used Haar Cascade classifier as it is trained with large number of positive and negative images. So that it avoids detecting false positives better than many other classifiers.

This deep learning model is trained to detect emotions of a person in real-time. At the end, it analyses the overall emotional behaviour of the person throughout the course and presents reports. The generated report can be used to analyse and review course content.

LIST OF FIGURES

INTRODUCTION

Emotion plays an important role for analyzing the student's interest in class room lectures. Out of the various ways to detect emotion, the quick way is to understand the emotion symptoms through facial expression .

Capturing the student’s emotion and dynamically changing the topic to elevate the student’s mood is missing in an e-learning. Most of the time in this type of environment even an important lecture or course ends up with the boredom of the student. All the effort of the instructor and their instructional aides become ineffective. In order to overcome these issues, more interactive learning technologies has been introduced in past few decades but was not able to find a better way to analyze the emotions at real-time in an e-learning environment. Thus there is a necessity for a system to analyze the mood of the student in an e- learning environment.

This project report presents the technology I developed for real-time emotion detection using deep learning and generating the analysys reports of there emotions in terms of pie chart. The consequence of this metric can also give clue to the instructor to make the topic interesting as per the student need.

1.2 OBJECTIVE

To capture the emotional behaviour of a person in real-time using facial emotions recognition, a deep learning model and generate report in the form of a pie-chart based on the captured data. The obtained report can be used to review the course content.

1.3 PROBLEM STATEMENTS

- Identifying emotions in real-time through webcam streaming.
- Generating report which can be used to analyse the impact of course on users.

OVERVIEW

2.1 PROJECT OVERVIEW:

Capturing the student's emotion and dynamically changing the topic to elevate the student's mood is missing in an e-learning. Most of the time in this type of environment even an important lecture or course ends up with the boredom of the student. All the effort of the instructor and their instructional aides become ineffective. In order to overcome these issues, more interactive learning technologies has been introduced in past few decades but was not able to find a better way to analyze the emotions at real-time in an e-learning environment. Thus there is a necessity for a system to analyze the mood of the student in an e- learning environment.

In-order to deal with this problem in e learning, a deep learning model is trained to detect emotions of a person. At the end, the overall emotional behaviour of the person is also reported by a pie-chart representation which can be used to analyse and review the course content.

Human emotion recognition plays an important role in the interpersonal relationship. Emotions are reflected from speech, hand and gestures of the body and through facial expressions. Convolutional Neural Networks (CNNs) is used for feature extraction and inference. Data model named "Mini-Xception model" is used along with "Haar Cascade Classifier" to preprocess the input images and extract the required features. The model is trained using FER-2013 dataset, which contains approximately 35,000+ gray images of pixel size 48x48. The dataset has 7 classes namely anger, disgust, fear, sad, happy, surprise and neutral.

In Mini-Xception model, Global Average Pooling is used in last layer which avoids working with large number of parameters and Depth-wise separable convolution is used to decouple spatial correlation and cross-channel correlations. This makes this model optimal to work with real-time content.

Using the webcam of user's system, video streaming is done through which users faces are read. Input images are processed and the classifier detects the faces in the input image. Based on the features extracted, emotions of the detected faces are identified. At the end, the overall emotional behaviour of the user throughout the period will be provided as a report in the form of a pie-chart. Based on the report, the course content can be analysed.

2.2 LIBRARIES USED

2.2.1 Tensorflow:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs.

2.2.2 OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York,

checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

2.2.3 NumPy:

NumPy is the fundamental package needed for scientific computing with Python. This package contains:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- basic linear algebra functions
- basic Fourier transforms
- sophisticated random number capabilities
- tools for integrating Fortran code
- tools for integrating C/C++ code

Besides its obvious scientific uses, *NumPy* can also be used as an efficient multi-dimensional container of generic data. Arbitrary data types can be defined. This allows *NumPy* to seamlessly and speedily integrate with a wide variety of databases.

NumPy is a successor for two earlier scientific Python libraries: Numeric and Numarray**.**

2.2.4 datetime:

In Python, **date, time and datetime** classes provides a number of function to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Whenever you manipulate dates or time, you need to import datetime function.

The datetime classes in Python are categorized into main 5 classes.

- date – Manipulate just date (Month, day, year)

- time – Time independent of the day (Hour, minute, second, microsecond)
- datetime – Combination of time and date (Month, day, year, hour, second, microsecond)
- timedelta— A duration of time used for manipulating dates
- tzinfo— An abstract class for dealing with time zones

2.2.5 matplotlib.pyplot:

Python's matplotlib library, it fortunately takes far less than a thousand words of code to create aproduction-quality graphic.

However, matplotlib is also a massive library, and getting a plot to look just right is often achievedthrough trial and error. Using one-liners to generate basic plots in matplotlib is fairly simple, butskillfully commanding the remaining 98% of the library can be daunting.

It is a collection of command style functions that make matplotlib work like MATLAB.
Eachpyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

2.2.6 keras.models import load_model:

Keras is a simple and powerful Python library for deep learning. Given that deep learning models can take hours, days and even weeks to train, it is important to know how to save and load them from disk.
Keras separates the concerns of saving your model architecture and saving your model weights.Model weights are saved to HDF5 format. This is a grid format that is ideal for storing multi-

dimensional arrays of numbers. Keras.models import load_model is used to import the data model or csv file.

2.2.7 keras.preprocessing import image:

Function that will be implied on each input. The function will run after the image is resized and augmented. The function should take one argument: one image (Numpy tensor with rank 3), and should output a Numpy tensor with the same shape. This is used to import and load an image for preprocessing.

2.2.8 statistics import mode:

Return the most common data point from discrete or nominal data. The mode (when it exists) is the most typical value, and is a robust measure of central location. If data is empty, or if there is not exactly one most common value, StatisticsError is raised. This assumes discrete data, and returns a single value.

2.2.9 os:

The OS module in Python provides a way of using operating system dependent functionality.

The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux. You can find important information about your location or about the process.

2.3 Mini Xception model

It’s complicated to establish a balance between their classification accuracy and unnecessary parameters. Commonly used CNNs for feature extraction include a set of fully connected layers at the end. Fully connected layers tend to contain most of the parameters in a CNN. Specifically, VGG16 contains approximately 90% of all its parameters in their last fully connected layers. Recent architectures such as Inception V3, reduced the number of parameters in their last layers by including a Global Average Pooling operation. Global Average Pooling reduces each feature map into a scalar value by taking the average over all elements in the feature map. The average operation forces the network to extract global features from the input image. Modern CNN architectures such as Xception leverage from the combination of two of the most successful experimental assumptions in CNNs: the use of residual modules and depth-wise separable convolutions. Depth-wise separable convolutions reduce further the number of parameters by separating the processes of feature extraction and combination within a convolutional layer.

The model was designed with the idea of creating the best accuracy over number of parameters ratio. Reducing the number of parameters help us overcoming two important problems.First, the use of small CNNs alleviate us from slow performances in hardware-constrained systems.And second, the reduction of parameters provides a better generalization. This architecture is trained with the ADAM optimizer. Global Average Pooling was used to completely remove any fully connected layers. In the last convolutional layer, the same number of feature maps as number of classes were maintained, and a softmax activation function was applied to each reduced feature map.

This architecture is a standard fully-convolutional neural network composed of 9 convolution layers, ReLUs, Batch Normalization and Global Average Pooling.

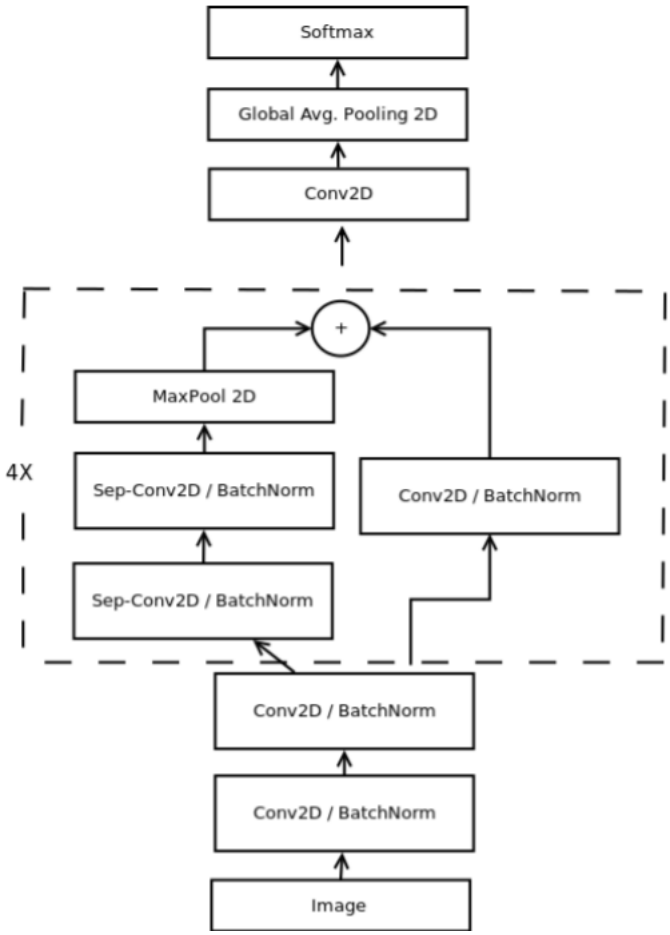


Figure – 2.1: Mini-Xception model architecture

2.3.1 CONVOLUTION 2D:

Convolution involving one-dimensional signals is referred to as 1D convolution or just convolution. Otherwise, if the convolution is performed between two signals spanning along two mutually perpendicular dimensions (i.e., if signals are two-dimensional in nature), then it will be referred to as 2D convolution. This concept can be extended to involve multi-dimensional signals due to which we can have multi-dimensional convolution.

In the digital domain, convolution is performed by multiplying and accumulating the instantaneous values of the overlapping samples corresponding to two input signals, one of which is flipped. This definition of 1D convolution is applicable even for 2D convolution except that, in the latter case, one of the inputs is flipped twice.

This kind of operation is extensively used in the field of digital image processing wherein the 2D matrix representing the image will be convolved with a comparatively smaller matrix called 2D kernel.

2.3.2 SEPARABLE CONVOLUTION:

There are two main types of separable convolutions: spatial separable convolutions, and depthwise separable convolutions.

2.3.2.1 Spatial Separable Convolutions:

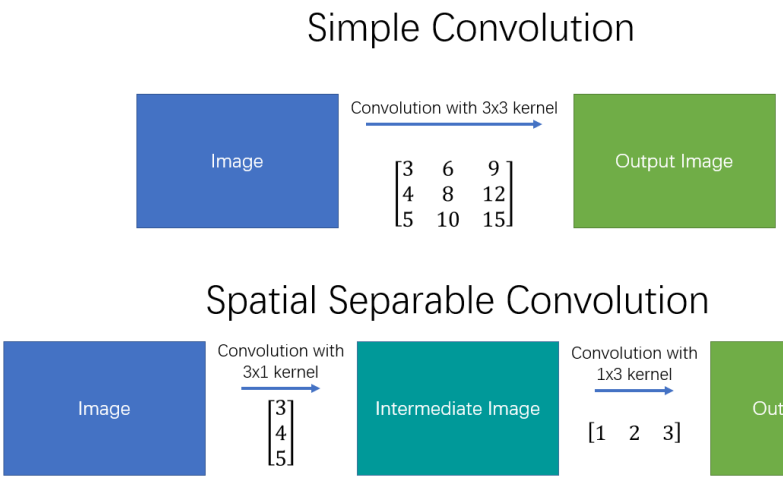
Conceptually, this is the easier one out of the two, and illustrates the idea of separating one convolution into two well, so I'll start with this. Unfortunately, spatial separable convolutions have some significant limitations, meaning that it is not heavily used in deep learning.

The spatial separable convolution is so named because it deals primarily with the **spatial dimensions** of an image and kernel: the width and the height. (The other dimension, the "depth" dimension, is the number of channels of each image).

A spatial separable convolution simply divides a kernel into two, smaller kernels. The most common case would be to divide a 3x3 kernel into a 3x1 and 1x3 kernel, like so:

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Now, instead of doing one convolution with 9 multiplications, we do two convolutions with 3 multiplications each (6 in total) to achieve the same effect. With less multiplications, computational complexity goes down, and the network is able to run faster.



2.3.2.2 Depthwise Separable Convolutions:

Unlike spatial separable convolutions, depthwise separable convolutions work with kernels that cannot be "factored" into two smaller kernels. Hence, it is more commonly used. This is the type of separable convolution seen in keras.layers.SeparableConv2D or tf.layers.separable_conv2d.

The depthwise separable convolution is so named because it deals not just with the spatial dimensions, but with the depth dimension — the number of channels — as well. An input image may have 3 channels: RGB. After a few convolutions, an image may have multiple channels. You can image each channel as a particular interpretation of that image; in for example, the "red" channel interprets the "redness" of each pixel, the "blue" channel interprets the "blueness" of each pixel, and the "green" channel interprets the "greenness" of each pixel. An image with 64 channels has 64 different interpretations of that image.

Similar to the spatial separable convolution, a depthwise separable convolution splits a kernel into 2 separate kernels that do two convolutions: the depthwise convolution and the pointwise convolution. But first of all, let's see how a normal convolution works.

2.3.3 BATCH NORMALIZATION:

We normalize the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning. If the input layer is benefiting from it, why not do the same thing also for the values in the hidden layers, that are changing all the time, and get 10 times or more improvement in the training speed.

Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). To explain covariance shift, let's have a deep network on cat detection. We train our data on only black cats' images. So, if we now try to apply this network to data with colored cats, it is obvious; we're not going to do well. The training set and the prediction set are both cats' images but they differ a little bit. In other words, if an algorithm learned some X to Y mapping, and if the distribution of X changes, then we might need to retrain the learning algorithm by trying to align the distribution of X with the distribution of Y.

Also, batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.

We can use higher learning rates because batch normalization makes sure that there's no activation that's gone really high or really low. And by that, things that previously couldn't get to train, it will start to train.

It reduces overfitting because it has a slight regularization effect. Similar to dropout, it adds some noise to each hidden layer's activations. Therefore, if we use batch normalization, we will use less dropout, which is a good thing because we are not going to lose a lot of information. However, we should not depend only on batch normalization for regularization; we should better use it together with dropout.

To increase the stability of a neural network, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

However, after this shift/scale of activation outputs by some randomly initialized parameters, the weights in the next layer are no longer optimal. SGD (Stochastic gradient descent) undoes this normalization if it's a way for it to minimize the loss function.

Consequently, batch normalization adds two trainable parameters to each layer, so the normalized output is multiplied by a "standard deviation" parameter (gamma) and add a "mean" parameter (beta). In other words, batch normalization lets SGD do the denormalization by changing only these two weights for each activation, instead of losing the stability of the network by changing all the weights.

2.3.4 ACTIVATION FUNCTION:

In artificial neural networks, the activation function of a node defines the output of that node, or "neuron," given an input or set of inputs. This output is then used as input for the next node and so on until a desired solution to the original problem is found.

The Activation Functions can be basically divided into 2 types-

- Linear Activation Function
- Non-linear Activation Functions

2.3.4.1 Linear or Identity Activation Function

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

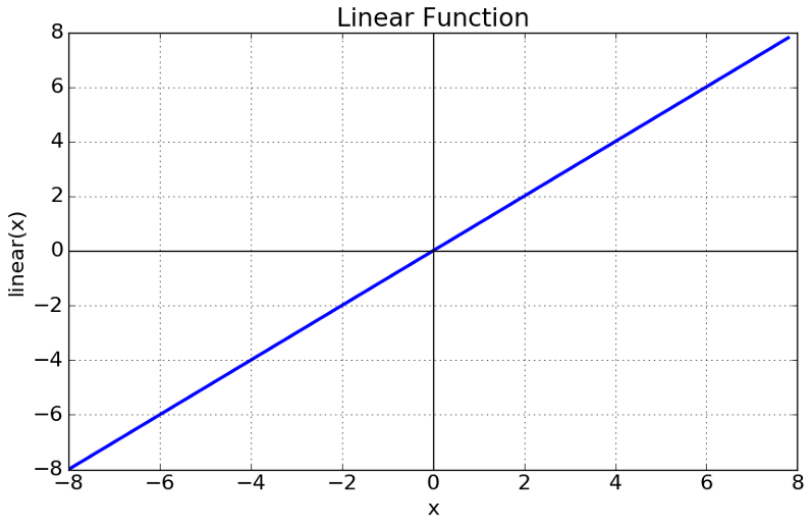


Figure – 2.2: Sample graph of a linear function

2.3.4.2 Non-linear Activation Function:

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this.

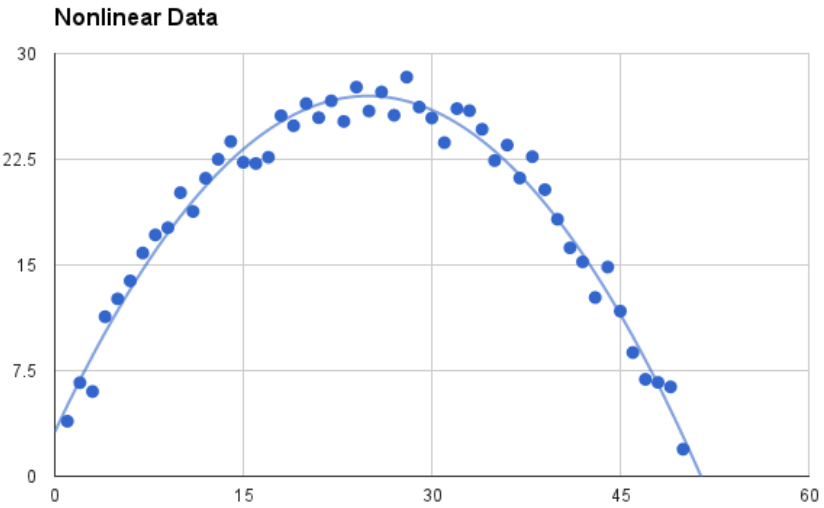


Figure - 2.3: Sample graph of a non-linear function

It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

ReLU (Rectified Linear Unit) Activation Function:

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as .***y = max(0, x)*.** Visually, it looks like the following:

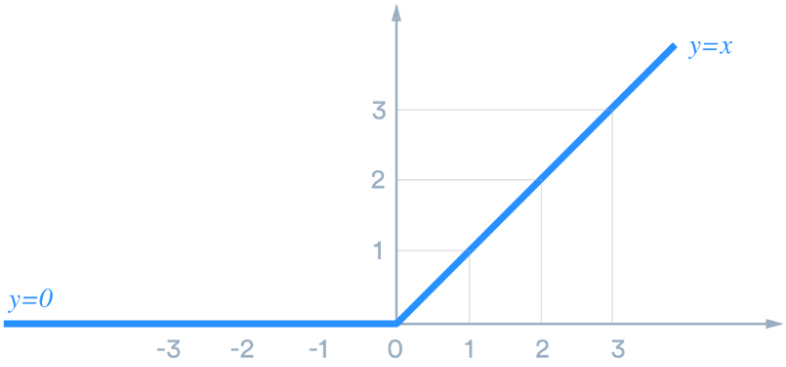


Figure - 2.4:Sample graph of ReLU Activation Function

ReLU is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network, ReLU is usually a good first choice.

It's surprising that such a simple function (and one composed of two linear pieces) can allow your model to account for non-linearities and interactions so well. But the ReLU function works great in most applications, and it is very widely used as a result.

Interactions: Imagine a single node in a neural network model. For simplicity, assume it has two inputs, called A and B. The weights from A and B into our node are 2 and 3 respectively. So the node output is f(2A+3B)f(2A+3B). We'll use the ReLU function for our f. So, if 2A+3B2A+3B is positive, the output value of our node is also 2A+3B2A+3B. If 2A+3B2A+3B is negative, the output value of our node is 0.

For concreteness, consider a case where A=1 and B=1. The output is 2A+3B2A+3B, and if A increases, then the output increases too. On the other hand, if B=-100 then the output is 0, and if A increases moderately, the output remains 0. So A might increase our output, or it might not. It just depends what the value of B is.

This is a simple case where the node captured an interaction. As you add more nodes and more layers, the potential complexity of interactions only increases. But you should now see how the activation function helped capture an interaction.

Non-linearities: A function is non-linear if the slope isn't constant. So, the ReLU function is non-linear around 0, but the slope is always either 0 (for negative values) or 1 (for positive values). That's a very limited type of non-linearity.

But two facts about deep learning models allow us to create many different types of non-linearities from how we combine ReLU nodes.

First, most models include a **bias** term for each node. The bias term is just a constant number that is determined during model training. For simplicity, consider a node with a single input called A, and a bias. If the bias term takes a value of 7, then the node output is f(7+A). In this case, if A is less than -7, the output is 0 and the slope is 0. If A is greater than -7, then the node's output is 7+A, and the slope is 1.

Softmax Activation Function:

Softmax function calculates the probabilities distribution of the event over 'n' different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

The main advantage of using Softmax is the output probabilities range. The range will **0 to 1**, and the sum of all the probabilities will be **equal to one**. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

The formula computes the **exponential (e-power)** of the given input value and the **sum of exponential values** of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function.

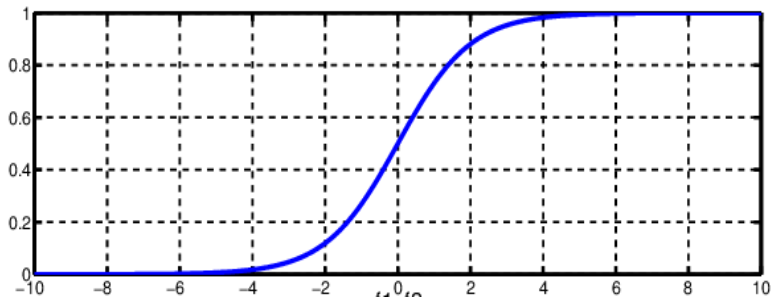


Figure - 2.5: Sample graph of Softmax function

Below are the few properties of softmax function:

- The calculated probabilities will be in the range of 0 to 1.
- The sum of all the probabilities is equals to 1.

Softmax Function Usage:

- Used in multiple classification logistic regression model.
- In building neural networks softmax functions used in different layer level.

2.3.5 GLOBAL AVERAGE POOLING:

In the last few years, experts have turned to global average pooling (GAP) layers to minimize over-fitting by reducing the total number of parameters in the model. Similar to max pooling layers, GAP layers are used to reduce the spatial dimensions of a three-dimensional tensor. However, GAP layers perform a more extreme type of dimensionality reduction, where a tensor with dimensions h × w × d is reduced in size to have dimensions 1×1×d. GAP layers reduce each h × w feature map to a single number by simply taking the average of all hw values.

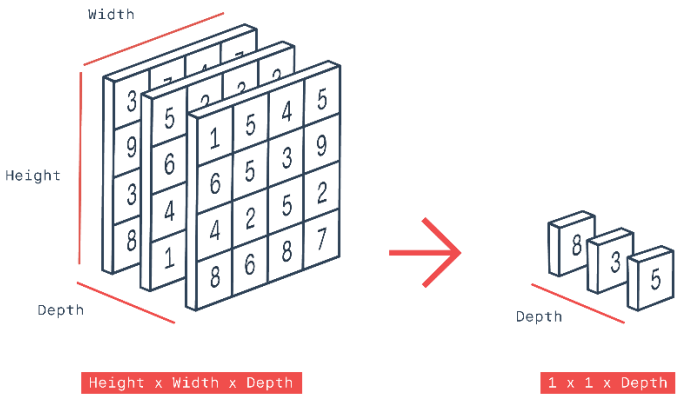


Figure - 2.6: Example demonstrating global average pooling

GAP layers designed an architecture where the final max pooling layer contained one activation map for each image category in the dataset. The max pooling layer was then fed to a GAP layer, which yielded a vector with a single entry for each possible object in the classification task. The authors then applied a softmax activation function to yield the predicted probability of each class.

In mid-2016, researchers at MIT demonstrated that CNNs with GAP layers (a.k.a. GAP-CNNs) that have been trained for a classification task can also be used for object localization. That is, a GAP-CNN not only tells us *what* object is contained in the image - it also tells us *where* the object is in the image, and through no additional work on our part! The localization is expressed as a heat map

(referred to as a **class activation map**), where the color-coding scheme identifies regions that are relatively important for the GAP-CNN to perform the object identification task.

2.3.6 ADAM OPTIMIZER:

Stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training. Whereas in ADAM method, it computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

- **Adaptive Gradient Algorithm** (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).
- **Root Mean Square Propagation** (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

2.3.7 GUIDED BACKPROPAGATION:

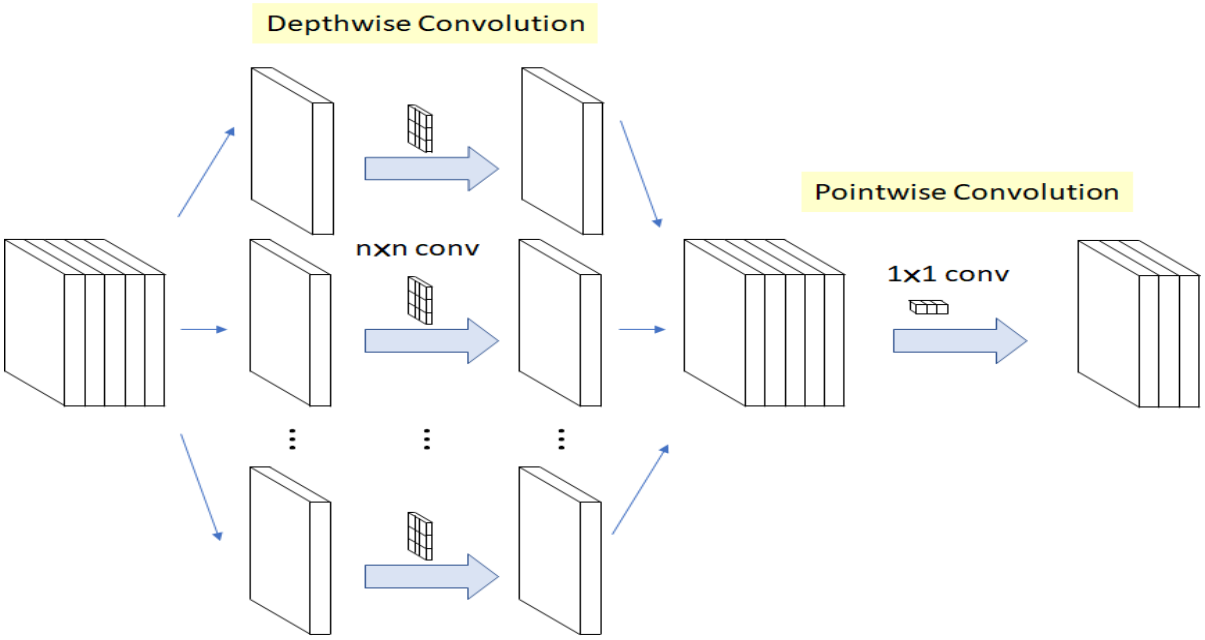
Neurons act like detectors of particular image features

- We are only interested in what image features the neuron detects, not in what kind of stuff it doesn't detect
- So, when propagating the gradient, we set all the negative gradients to 0
- We don't care if a pixel "suppresses" a neuron somewhere along the part to our neuron.

Figure - 2.7:Backprop vs Guided Backprop on a sample image

Mini-Xception model contains approximately 600,000 parameters. This model was also trained and validated with the FER-2013 dataset. This dataset contains 35,887 grayscale images where each image belongs to one of the following classes {“angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”}. This model achieved an accuracy of 66% for the emotion classification task on FER-2013 dataset.

This architecture combines the use of residual modules and depth-wise separable convolutions. Residual modules modify the desired mapping between two subsequent layers, so that the learned features become the difference of the original feature map and the desired features. Consequently, the desired features H(x) are modified in order to solve an easier learning problem F(X) such that: H(x) = F(x) + x



. Figure - 2.8: Depthwise Separable Convolution

The large number of parameters were eliminated in the last fully connected layer which was done through the use of depth-wise separable convolutions. Depth-wise separable convolutions are composed of two different layers: depth-wise convolutions and pointwise convolutions**.** Depth-wise separable convolutions reduces the computation with respect to the standard convolutions by a factor of (1/ N)+ (1/ D2).

This architecture is a fully-convolutional neural network that contains 4 residual depth-wise separable convolutions where each convolution is followed by a batch normalization operation and a ReLUs activation function. The last layer applies a global average pooling and a soft-max activation function to produce a prediction. This architecture has approximately 60, 000 parameters; which corresponds to a reduction of 10× when compared to our initial naive implementation, and 80× when compared to the original CNN.

This model was also included with a real-time guided back-propagation visualization to observe which pixels in the image activate an element of a higher-level feature map i.e., to validate the features learned by CNN. Given a CNN with only ReLUs as activation functions for the intermediate layers, guided backpropagation takes the derivative of every element (x, y) of the input image I with respect to an element (i, j) of the feature map f^L in layer L. The reconstructed image R filters all the negative gradients; consequently, the remaining gradients are chosen such that they only increase the value of the chosen element of the feature map. A fully ReLU CNN reconstructed image in layer l is given by: $R^l_{i,j} = (R^{l+1}{}_{i,j} > 0) * R^{l+1}{}_{i,j.}$

A comparison of the learned features between several emotions can be observed in the following Figure. The white areas in figure correspond to the pixel values that activate a selected neuron in our last convolution layer. The selected neuron was always selected in accordance to the highest activation. We can observe that the CNN learned to get activated by considering features such as the frown, the teeth, the eyebrows and the widening of one's eyes, and that each feature remains constant within the same class. These results reassure that the CNN learned to interpret understandable human-like features, that provide generalizable elements.

Figure - 2.9: Results of the real-time classification using mini-Xception

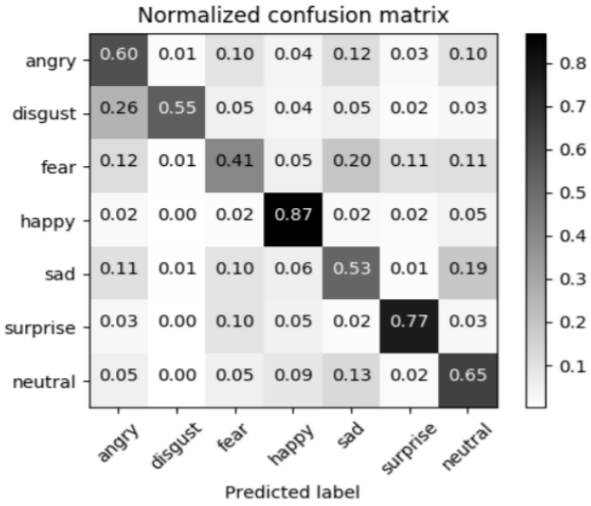


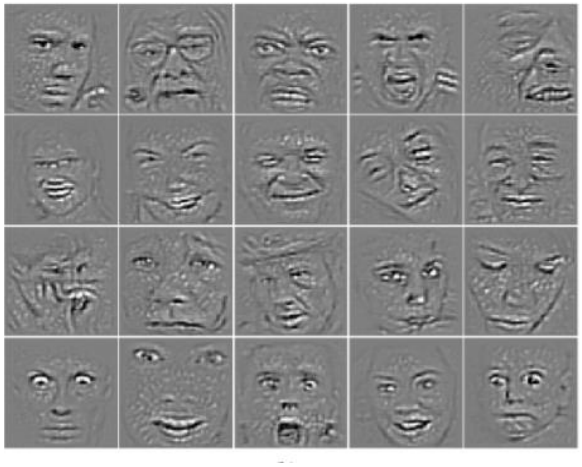
Figure - 2.10: Confusion matrix of emotion classification using mini-Xception

Challenges:

- Persons with glasses are being classified as "angry". This happens since the label "angry" is highly activated when it believes a person is frowning and frowning features get confused with darker glass frames.
- We can observe several common misclassifications such as predicting "sad" instead of "fear" and predicting "angry" instead "disgust" by the confusion matrix.

The sub-figures contain the same images in the same order. Every row starting from the top corresponds respectively to the emotions {"angry", "happy", "sad", "surprise"}

(a) Samples from the FER-2013 dataset



(b) Guided backpropagation

Figure - 2.11 visualization of mini-Xception model

DATASET

A **data set** (or **dataset**) is a collection of data. The term data set may also be used to refer to the data in a collection of closely related tables, corresponding to a particular experiment or event.Selecting the right dataset for Machine learning is very important to make the AI model functional with right approach. Though selecting the right quality and amount of data is challenging task but there are few rules needs to be followed for machine learning on big data.

3.1 FER-2013 DATASET:

Due to the important role of facial expressions in human interaction, the ability to perform Facial Expression Recognition (FER) automatically via computer vision enables a range of novel applications in fields such as human-computer interaction and data analytics.

Consequently, Facial Expression Recognition (FER) has been widely studied and significant progress has been made in this field. In fact, recognizing basic expressions under controlled conditions (e.g. frontal faces and posed expressions) can now be considered a solved problem. The term basic expression refers to a set of expressions that convey universal emotions, usually anger, disgust, fear, happiness, sadness, and surprise. Recognizing such expressions under naturalistic conditions is, however, more challenging. This is due to variations in head pose and illumination, occlusions, and the fact that unposed expressions are often subtle. Reliable FER under naturalistic conditions is mandatory in the real-time applications, yet still an unsolved problem.

Convolutional Neural Networks (CNNs) have the potential to overcome these challenges. CNNs have enabled significant performance improvements in related tasks, and several recent works on FER successfully utilize CNNs for feature extraction and inference.

Figure – 3.1: Images of Fer-2013 of all 7 classes

FER2013 is a large, publicly available FER dataset consisting of 35,887 face crops. The dataset is challenging as the depicted faces vary significantly in terms of person age, face pose, and other factors, reflecting realistic conditions. The dataset is split into training, validation, and test sets with 28,709, 3,589, and 3,589 samples, respectively. Basic expression labels are provided for all samples. All images are grayscale and have a resolution of 48 by 48 pixels. The human accuracy on this dataset is around 65.5%.

CLASSIFICATION

4.1 HAAR CASCADE CLASSIFIER

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

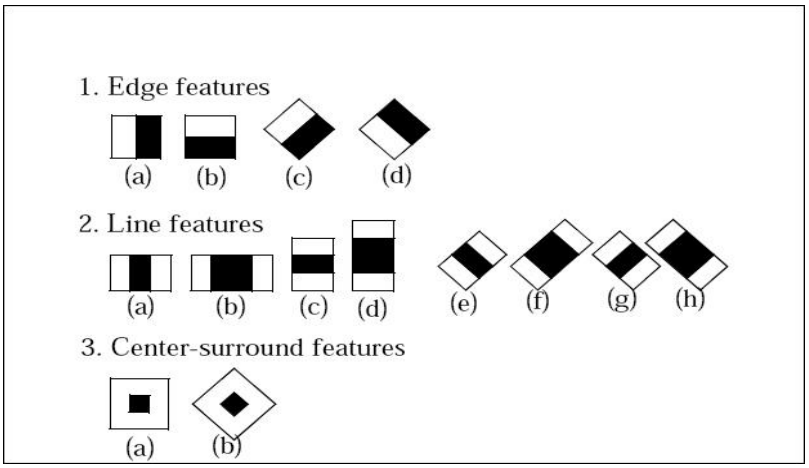


Figure – 4.1: Different types of features of Haar Cascade Classifier

Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

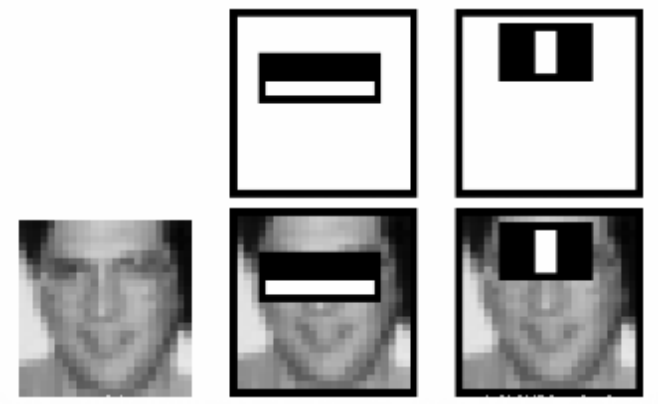


Figure - 4.2: Sample to extract features using Haar Cascade Classifier

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found)

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow. Isn't it a little inefficient and time consuming? Yes, it is. The authors have a good solution for that.

In an image, most of the image is non-face region. So, it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one.

(Normally the first few stages will contain very many fewer features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. How is that plan!

The authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in the first five stages. (The two features in the above image are actually obtained as the best two features from Adaboost). According to the authors, on average 10 features out of 6000+ are evaluated per sub-window.

EXPERIMENTAL RESULTS

5.1 CODE MODULES

The Cascade Classifier to extract features is used in this, is Haar Cascade Classifier. The following module shows how the model and the classifier are loaded.

```
emotion_model_path = './models/emotion_model.hdf5'
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)

emotion_labels = get_labels('fer2013')
```

The definition of the function to get labels of classes in FER-2013 dataset is as shown.

```
def get_labels(dataset_name):
    if dataset_name == 'fer2013':
        return {0:'angry',
        1:'disgust',
        2:'fear',
        3:'happy ',
        4:'sad',
        5:'surprise',
        6:'neutral'}
    else:
        raise Exception('Invalid dataset name')
```

The following code module represents how the video streaming is done using opencv. The video is captured only if the webcam is available, if not available the video at the given directory path will be played.

```
# starting video streaming
cv2.namedWindow('window_frame', cv2.WINDOW_NORMAL)
video_capture = cv2.VideoCapture(0)

# Select video or webcam feed
cap = None
if (USE_WEBCAM == True):
   cap = cv2.VideoCapture(0)   # Webcam source
else:
  cap = cv2.VideoCapture('./demo/dinner.mp4')   # Video file source to play if webcam
                                                    source isn't found
```

For **cap.isOpened(),** the program captures the video and each frame is converted into a gray image because the dataset FER-2013 has gray images of size(48,48). Using Haar Cascade Classifier faces are detected in each captured frame with minNeighbours count '5' as shown in the following code module. (The count is picked in order to avoid false positives and detect every true positive)

```
    ret, bgr_image = cap.read()
gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
    # returns rectangle like coordinates
    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5,
                                minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)
```

Offsets are applied to each face detected by the classifier and resized as shown below,

```
x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
gray_face = gray_image[y1:y2, x1:x2]
    try:
gray_face = cv2.resize(gray_face, (emotion_target_size))
    except:
          continue
```

The definition of the function to apply offsets is as shown

```
  def apply_offsets(face_coordinates, offsets):
          x, y, width, height = face_coordinates
          x_off, y_off = offsets
          return (x - x_off, x + width + x_off, y - y_off, y + height + y_off)
```

The detected faces are matched and the emotion of each detected face is predicted. All the emotions that are predicted, along with their matching probabilities are considered. Then the emotion with highest matching probability is considered as the emotion of detected face.

```
gray_face = preprocess_input(gray_face, True)
gray_face = np.expand_dims(gray_face, 0)
gray_face = np.expand_dims(gray_face, -1)
emotion_prediction = emotion_classifier.predict(gray_face)
emotion_probability = np.max(emotion_prediction)    #gives max value
emotion_label_arg = np.argmax(emotion_prediction)   #returns max value's coordinates
      # getting labels for the matched emotion image
emotion_text = emotion_labels[emotion_label_arg]
face_label = emotion_text + " " + str(emotion_probability)
      print(face_label)
emotion_window.append(emotion_text)
```

Based on the emotion and the matching probability colors are picked to mark the detected faces as shown,

```
if emotion_text == 'angry':
        # red color
        color = emotion_probability * np.asarray((255, 0, 0))
elifemotion_text == 'sad':
        # blue color
        color = emotion_probability * np.asarray((0, 0, 255))
elifemotion_text == 'happy':
        # yellow color
        color = emotion_probability * np.asarray((255, 255, 0))
elifemotion_text == 'surprise':
        # aqua color
        color = emotion_probability * np.asarray((0, 255, 255))
```

The following code module is to draw boundary boxes around the faces detected and labels above them.

```
color = color.astype(int)
color = color.tolist()
draw_bounding_box(face_coordinates, rgb_image, color)
draw_text(face_coordinates, rgb_image, emotion_mode,
          color, 0, -45, 1, 1)
```

The definition of the function to draw boundary boxes, is as shown

```
def draw_bounding_box(face_coordinates, image_array, color):
    x, y, w, h = face_coordinates
    cv2.rectangle(image_array, (x, y), (x + w, y + h), color, 2)
```

The definition of the function to display the emotions of detected faces as labels, is as shown

```
def draw_text(coordinates, image_array, text, color, x_offset=0, y_offset=0,
font_scale=2, thickness=2):
    x, y = coordinates[:2]
    cv2.putText(image_array, text, (x + x_offset, y + y_offset),
            cv2.FONT_HERSHEY_SIMPLEX,
            font_scale, color, thickness, cv2.LINE_AA)
```

The following code module is to display the color image(as video stream),

```
bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
cv2.imshow('window_frame', bgr_image)
```

Inorder to close the window showing the video streaming the key ‘q’ is supposed to be pressed, as shown below. As soon as the key specified is pressed, ‘cap.release()’ function will be called.

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

After the window is closed, the user will be shown with a plot of his/her overall emotions throughout the period. The following code shows how the plot can be plot. ‘plt.show()’ function will be called after ‘cap.release()’ to display the plot.

```
normal= Interested - " + str(okper)
ntnormal= "not Interestedl - " + str(ntokper)
labels = [normal,ntnormal]
sizes = [okper,ntokper]
colors = ['yellowgreen','lightskyblue']
patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
```

5.2 OUTPUT SCREENS

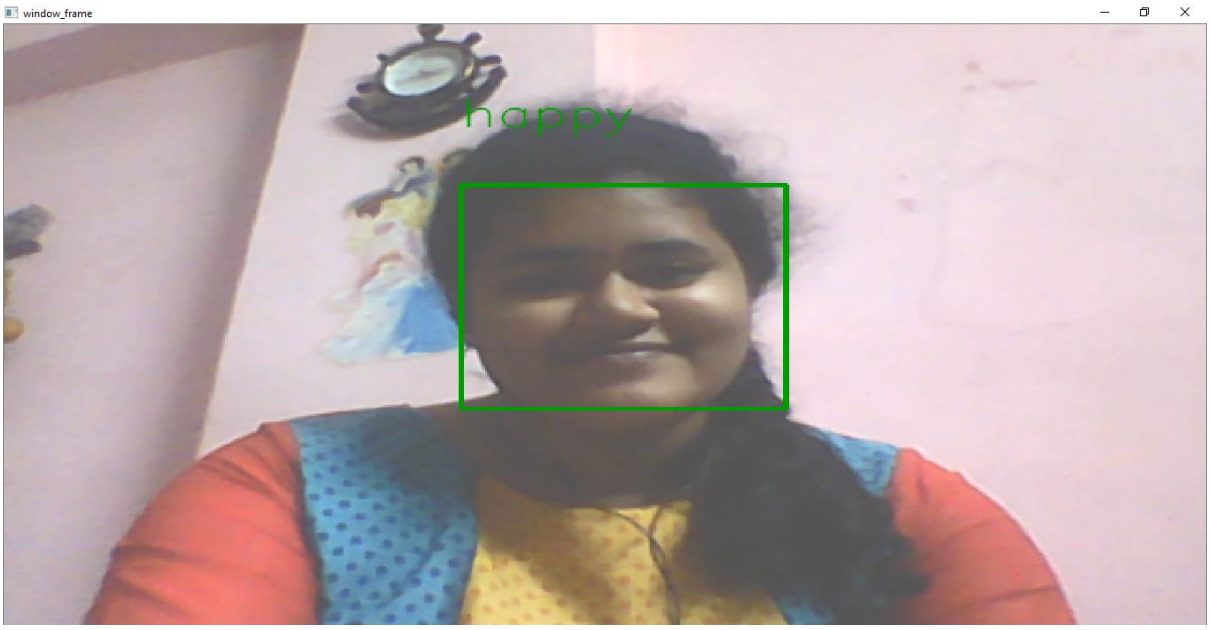


Figure:5.1

Figure:5.2

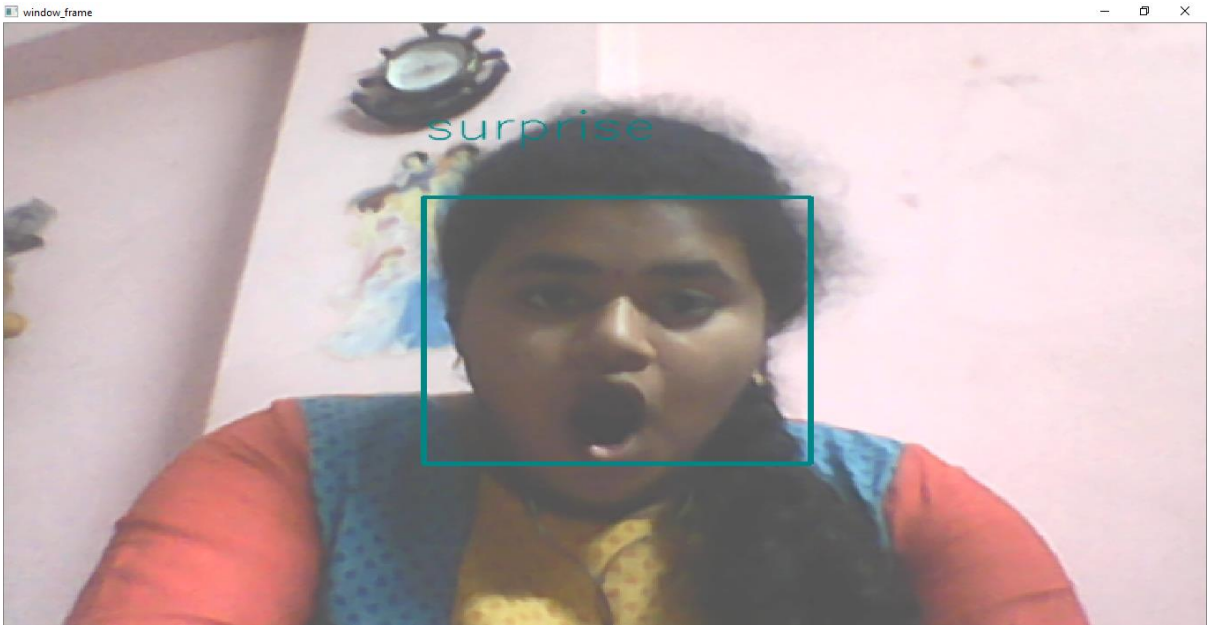


Figure:5.3

Figure:5.4

5.3 RATIO OF INTERESTED TO NOT INTERESTED EMOTIONS

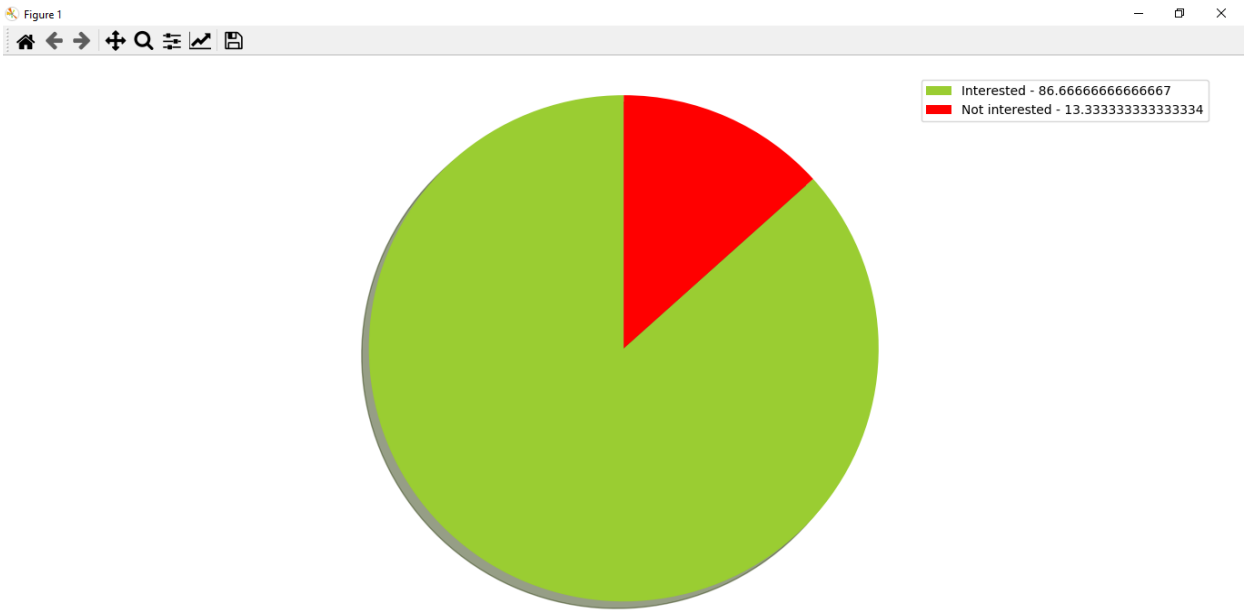


Figure :5.5

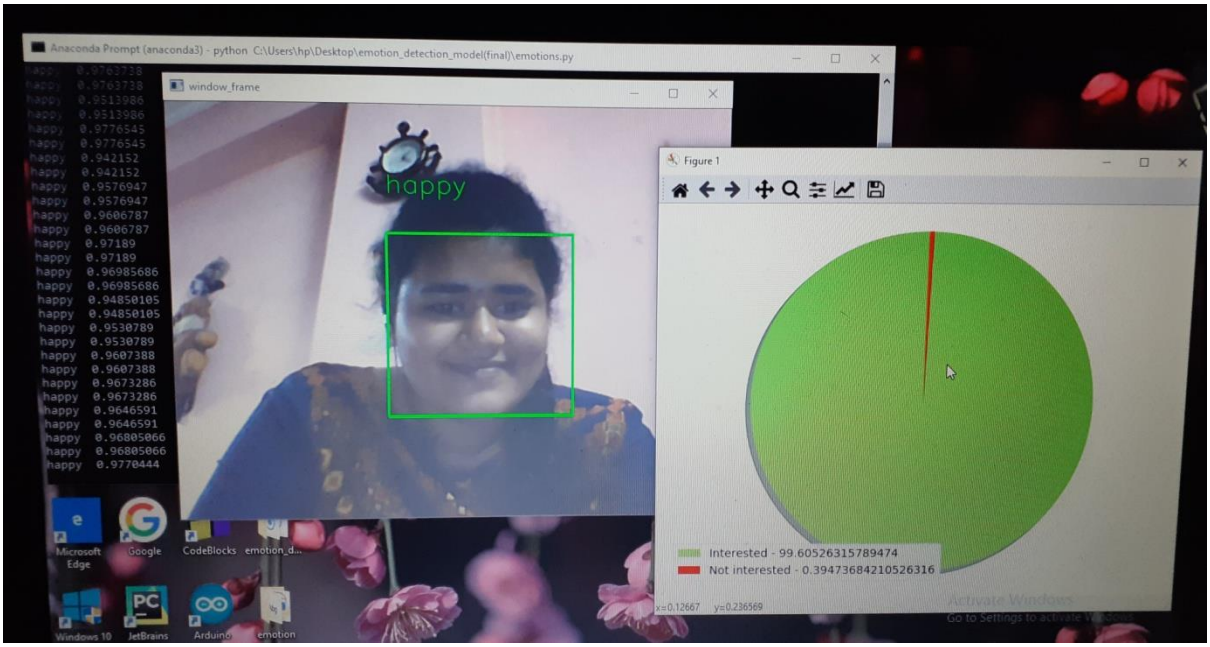


Figure: 5.6

CONCLUSION

The objective of this project i.e., to identify students emotions, a deep learning model inspired from Mini-Xception model is built and is trained with FER-2013 dataset which contains 35,887 grayscale images where each image belongs to one of the following classes {“angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”}. To deal with real-time content, avoiding large number of parameters is the most important thing. So, I adopted Mini-Xception model which replaces the fully-connected layer that has more parameters, with global average pooling. Here, I used Haar Cascade classifier as it is trained with large number of positive and negative images. So that it avoids detecting false positives better than many other classifiers.

This deep learning model is trained to detect emotions of a person in real-time. At the end, it analyses the overall emotional behaviour of the person throughout the course and reports in the form of a pie-chart. The generated report can be used to analyse and review the course content.

REFERENCES

[1]. Real-time Convolutional Neural Networks for Emotion and gender Classificationby **Octavio** Arriaga, Paul G. Ploger, Matias Valdenegro

https://arxiv.org/pdf/1710.07557.pdf

[2]. Xception: Deep learning with depthwise separable convolutions by Franc¸oisChollet.

CoRR, abs/1610.02357, 2016.

[3]. Student Emotion Recognition System (SERS) for e-learning improvement based on learner concentration metric Krithika.L.Ba , Lakshmi Priya GGb.

[4]. Handbook of Learning Analytics , Charles LANG, George SIEMENS, Alyssa WISE, and Dragan GAŠEVIĆ (Eds)

[5]. Facial Expression Recognition using Convolutional Neural Networks: State of the Artby Christopher Pramerdorfer, Martin Kampel

https://www.researchgate.net/figure/Example-images-from-the-FER2013-dataset-3-illustrating-variabilities-in-illumination_fig1_311573401

[6]. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shiftby Sergey loffe, Christian Szegedy

[7]. Guided Backpropagation: How ConvNets See + Applications:

https://www.cs.toronto.edu/~guerzhoy/321/lec/W07/HowConvNetsSee.pdf

[8]. Batch normalization:Accelerating deep network training using reducing internal covariate shift by Sergey Ioffe and Christian Szegedy in International Conference on Machine Learning, pages 448–456, 2015.

[9]. Adam: A method for stochastic optimizationbyDiederikKingma and Jimmy Ba https://arxiv.org/pdf/1412.6980.pdf

[10]. Face Detection using Haar Cascade Classifier

https://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html