DEEP LEARNING BASED HUMAN EMOTION RECOGNITION FROM FACIAL EXPRESSION

Dhanonjoy Howlader

Ratin Dev Sadhon

Sararah Mahjabin

University of Asia Pacific

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2022

DEEP LEARNING BASED HUMAN EMOTION RECOGNITION FROM FACIAL EXPRESSION

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering of the University of Asia Pacific

By

Dhanonjoy Howlader
Registration ID : 18101062

Ratin Dev Sadhon

Registration ID : 18101059

Sararah Mahjabin

Registration ID : 18101067

Supervised By

Shaila Rahman

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF ASIA PACIFIC

June 2022

DECLARATION

We expressly declare that the work portrayed in this thesis is the result of our exploration conducted under the supervision of Shaila Rahman, Assistant Professor, Department of Computer Science, University of Asia Pacific. We also confirm that no part of this Thesis has been or is being submitted to any other institution for the respect of a degree or diploma.

Countersigned

………………………………..

(Shaila Rahman)

Supervisor

Signature

Dhanonjoy

…………………………......

(Dhanonjoy Howlader)

Ratin

……………………………

(Ratin Dev Sadhon)

Sararah

………………………...

(Sararah Mahjabin)

Certificate of Approval

We at this moment, highly urge that the thesis is prepared by Dhanonjoy Howlader, Ratin Dev Sadhon, Sararah Mahjabin entitled ***“Deep learning based human emotion recognition from Facial expression”*** I accepted it as one of the requisites for a degree of Bachelor of Science in Computer Science and Engineering.

Shaila Rahman
Chairman of the Committee
Assistant Professor
(Supervisor)
Department of Computer Science and Engineering
University of Asia Pacific (UAP)

Prof. Dr. Bilkis Jamal Ferdosi
Member of the Committee
Professor
(External)
Department of Computer Science and Engineering
University of Asia Pacific (UAP)

Dr. Md. Rajibul Islam
Assistant Professor and Head
Head of the Department
Department of Computer Science and Engineering
University of Asia Pacific (UAP)

ACKNOWLEDGEMENTS

First and supreme, I'd like to express my sincere grateful to ALLAH. Today I completed our thesis work successfully and so effortlessly because almighty gave me the ingenuity, chance, and a cooperative supervisor. I would like to take a moment to express my thankfulness to my guidance and respectable supervisor, Shaila Rahman. Despite the fact that she was often busy with various other activities, she made sure I had adequate time for this work. She not only gave me time, but also provided me appropriate instruction and valuable advice anytime I was facing difficulties. Her proper suggestions and guidance encouraged me in completing my thesis paper. We are very happy to express our heartfelt gratitude for our honorable external Prof. Dr. Bilkis Jamal Ferdosi. Though having too busy with her schedules, she was very cooperative and provided us sufficient time. She not only provided us her valuable time and effort, but also benefited us by providing appropriate guidance and whenever we faced any challenges or difficulties. she was available to provide us with important advice based on her extensive experience and knowledge. We would want to express our gratitude to our teachers, my respected teacher, who has inspired us at every stage. Finally, I am grateful to my family, who have always stood by my side throughout my life.

Dedication

We dedicate our final-year work to our family, friends, and well-wishers. Our grateful parents, whose words of reassuring and emphasis on perseverance still ring in our ears, owe us a remarkable respect and gratitude. Our siblings, who have never abandoned us and are extremely beloved to us.

This dissertation is also dedicated to our numerous friends and seniors who have encouraged and helped us throughout the process. We will remain extremely thankful for everything they've done for us. We dedicate this work to our honorable supervisor, Shaila Rahman and admire and respect her for her guidance and support throughout the thesis.

Abstract

Human facial expression instead of speaking convey a large amount of knowledge visually. Facial expression recognition is vital in the realm of human-machine cooperation. Understanding human behavior, identifying mental diseases, and creating polymerized human expressions are just a few of the applications of an automatic facial expression recognition system. Computer recognition of facial expressions with a high recognition rate is still a laborious task. In the previous 20 years, various approaches for Face Expression Recognition have been applied, however they are usually divided into two categories: conventional or traditional FER approach and deep learning-based approach. In this Thesis, we use a Deep Learning Approach. In this approach, there are three basic steps these are: pre-processing, deep feature learning and deep feature classification face detection (mainly utilizing the Viola-John algorithm (with Haar Cascade), face alignment, normalization (illumination, pose), and buildup are usually included in this phase (scaling, rotating, colors, noises, etc.). We used three different Keras applications in this study, these are: VGG-19, ResNet-50V2, and MobileNetV2, and then respectively we have achieved different accuracy 92.89 %, 91.62 %, and 92.76 %. In order to detect the seven emotions in human faces, different parameters and designs of Conventional Neural Networks (CNNs) were used. These are: angry, fear, disgust, neutral, happiness, sadness and surprise.

TABLE OF CONTENTS

Chapter 1: Introduction

Chapter 2: Machine Learning and CNN, Environment Setup

Chapter 3: Related Works

Chapter 4: Datasets

Chapter 7: Conclusion

LIST OF FIGURES

Chapter 1

Introduction

1.1 Introduction

Strong and consequential human interaction is necessary to express emotions and communicate with others. Human communication is indeed the process of transferring information. Information can be conveyed by words, voice tone, facial expressions, and body posture. In addition to verbal communication, nonverbal communication such as body posture, facial expression, attitudes, gesture, and others can be utilized to transmit communicative feelings [1]. Only seven percent (7%) of the information conveyed is articulated in words. Vocal tone takes up Fifty-five percent (55%) of the time while body posture takes up (38%) of the time [2]. Nonverbal communication evolves naturally, and many of us aren't even conscious that we're improving our words with things like body posture or facial expression.

In 1971, Ekman et al. [3] defined six universal facial expressions: anger, disgust, fear, happiness, sadness, and surprise. Nonverbal communication is learnt through social-emotional interactions as an infant, making the face rather than the voice the main mode of communication [4].

Children with superior face-reading skills are much popular at school as noted in [5], and they are more intellectually successful, as exhibited in [6]. Moreover, studies demonstrate that people who can recognize fear signals are kinder and more charitable [7].

On the other side, children who have trouble recognizing emotion in faces are more likely to have companion issues and learning challenges, as demonstrated in [8]. Externalizing 2 behavioral issues are more common in preschoolers who have weak face-reading abilities for their age. such as hyperactivity [9], or, if shy, anxiety [10].

1.2 Motivation

Facial Emotion Recognition (FER) are incredibly useful in a range of real-life situations since they provide crucial information about an individual. That is the reason why they have been thoroughly researched and are utilized in a wide range of systems. Medical treatments, human resources, police investigations, education (on students during lectures), customer service, journalism (during interviews), and more can all benefit from FER. Advertisements, health-care, education, wearable technologies, and more can all benefit from the facial expression recognition (FER) system.

1.3 Objective

The objective of the Face emotion recognition (FER) is to identify an individual’s emotions. Emotions can be expressed in a variety of ways, including facial expressions and verbal communication. Emotions are identified by psychological features such as heartbeat and blood pressure, speech, hand gestures, bodily movements, and facial expressions.

Facial expression recognition, or a computer-based facial expression recognition system, is essential because of its ability to imitate human coding capabilities. In interpersonal relationships, nonverbal communication indicates facial expressions and other mannerisms which are extremely important.

1.4 Social and ethical issues

Facial recognition has countless precedence for society, spanning from taking the edge off gratuitous human reciprocity and toil to fend off misdemeanor and ameliorate well-being and reliability. Scarcely, it can flush succor medical reinforcement.

1.5 Environment & sustainability issues

Recognizing facial emotions is a very useful thing for ensuring security, better healthcare, understanding kids behaviors, and improving robots' smartness. This type of system will help to make our life easier. So easily we can say that this system will be sustainable in the future.

1.6 Our Contribution:

In this work, we proposed a system that can identify Human Emotion. First, we collected some data of Bangladeshi people. Then we up and down sampling those data. Actually, we collect it for testing our Machine learning model. In this work, we consider a few popularly used web applications categorized into Seven classes that can be easily scaled up in the future. We experimented deep learning Keras applications: VGG-19 , ResNet-50V2 and MobileNetV2 to compare the accuracy and performance. The performance and accuracy of the proposed system using a VGG-19 excels compared to the others.

1.7 Critical Challenges

Our test data collection period was under covid-19 pandemic time wearing mask was very mandatory in that time. So, in this situation collects different emotions data of Human was very challenging. For training we used FER2013 dataset for train our model. FER2013 is very big dataset 28,709 data for training. For training model using this amount big dataset we required good hardware support. For getting good hardware support was very challenging also.

1.8 Statement of Originality

There are very few works that focus on our field of interest for solving this problem. According to our knowledge, there is no proposed method analogous to us. We completed the entire study project with relevant citations. The rest of the research is organized as follows.

The background theory is summarized in Chapter 2. Chapter 3 contains the literature reviews for our projects. The details and description of our suggested method are offered in Chapter 5, followed by a discussion of our datasets in Chapter 4. The findings of our proposed system's evaluation are provided in Chapter 6. In Chapter 7, the conclusion and future efforts are presented.

1.7 Background Study

We wanted to understand how to handle a specialized type of picture classification problem because our job is about it. A Convolutional Neural Network (CNN) is a Deep Learning algorithm capable of taking an image as input and distinguishing one from the other, and it may be used to classify activities in our work [14].

We employed VGG-19, Resnet50V2, and MobileNetV2 transfer learning Keras applications. we had to do was train the layers every day to get a rather accurate result. This is considered as transfer learning, and it focuses on accumulating acquired knowledge while resolving one problem and implementing it to a related but distinct problem.

1.8 Layout of Thesis Book

Chapter 1: Introduction

We discuss about the motivation, objective, our contribution critical challenges and so on of our thesis in this chapter.

Chapter 2: Machine Learning and CNN Environment Setup

In this chapter we have discussed all the research and their work and also discussed some important terms like CNN, Transfer Learning, Deep Learning etc.

Chapter 3: Working Model

This chapter is related to all the literature that we have studied from several research papers during our Thesis work.

Chapter 4: Dataset

This chapter provides a good overview of the FER2013 and our collected dataset.

Chapter 5 : Methodology

This chapter gives a clear description about the working methodology i.e which algorithms are applied for this Thesis as well as discussed data collection procedure and method and also gives a statistical analysis about the Thesis.

Chapter 6: Evaluation and Results

This chapter addressed the experimental result of all human emotions and discuss the process of recognizing emotions in descriptive analysis part.

Chapter 7: Conclusion and Implication for Future Research

In this chapter we have discussed all the process in short and our target in future on this project.

Chapter 2

Machine Learning and CNN Environment Setup

2.1 Convolutional Neural Network (CNN)

In Artificial Intelligence, computer vision is a field, which allows machines to observe and perceive objects in the same way that humans do. The Convolutional Neural Network (CNN) algorithm is crucial for the progress and development of computer vision using deep Learning. A CNN is a Deep Learning algorithm which can take images as input, train itself using filters to discern distinct features of images, and distinguish one image from another.

CNNs are neural networks that have one or e more convolutional layers and are widely used within machine learning for image processing, classification, segmentation, and other auto-correlated data. CNN is a supervised deep learning algorithm utilized mostly in image recognition and computer vision. CNN is a pattern recognition and image processing algorithm which is frequently utilized. It has numerous advantages, including a straightforward structure, fewer training parameters, and adaptability

They are complicated feed forward neural networks used in machine learning. Because of its extreme precision. Feed-forward has been around for a long time. fully-connected layers are used in Machine Learning algorithms, with every neuron in a layer coupled to each output neuron in the following layer. To begin the network, CNNs employ a Convolution layer, and in the conclusion, they use fully-connected networking.

Each layer of a CNN applies a distinct set of filters, usually hundreds or thousands, and then aggregates the results by sending the output within the network's next layer. During train a model, the CNN learns the values for these filters automatically. "A CNN may learn to:

- Detects edges from raw pixel data in the first layer in the context of image classification.

- In the second layer, use these edges to detect shapes (i.e., "blobs"); in the highest layers of the network,

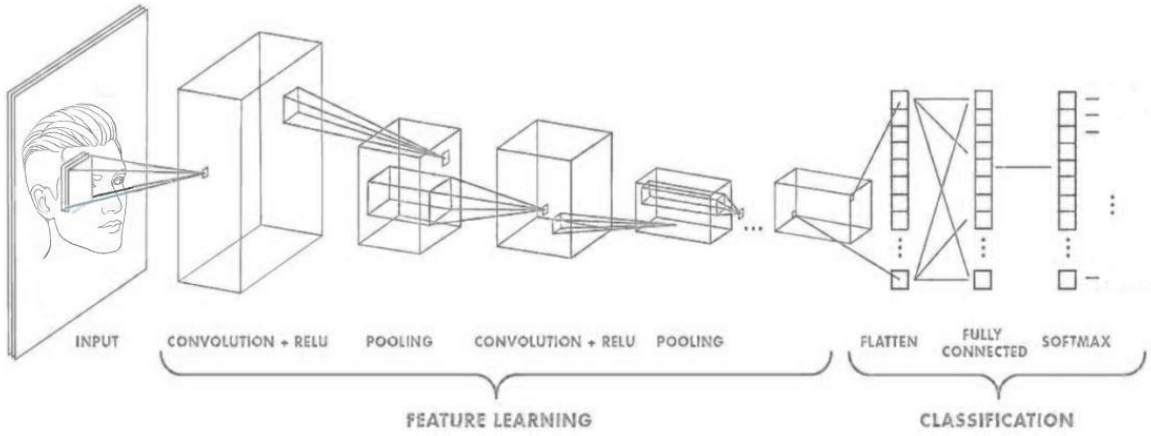


Figure 2.1: Convolutional Neural network Architecture

2.1.1 Usage of CNN for Image Recognition

Prior to CNN's success in computer vision applications, image classification is performed using traditional Machine Learning approaches. However, there was no way to directly input images into the algorithm. At first, features from an image were extracted using a distinct feature extraction approach. The features were then given into a classification algorithm such as Support Vector Machine (SVM), k-Nearest Neighbor, Logistic Regression, and others. Images pixel level values were also used as a feature vector in many algorithms. For SVM, developing a model with 2,304 features would necessitate each and every feature representing a pixel value for a 48×48 image.

The fact that, CNN does not require feature development for making it a better choice for image recognition than other algorithms. This gives the CNN algorithm the ability to extract features as part of its architecture. Those extracted features are learned by the CNN during the model's training process. As a result, CNNs can be viewed as an image feature extractor that works automatically.

Yann Le Cun introduced the concept of CNN in 1998, when he studied numerous methods for handwritten character recognition and compared them on a standard handwritten digit identification assignment. The author only employed one Convolution layer in this paper. AlexNet , later uses many Convolution layers to obtain great performance on the ImageNet dataset, which is the largest dataset used for object detection, popularized in 2012.

2.1.2 CNN Implementation for Facial Expression Recognition

The objective of establishment of the facial expression recognition model is to categorize seven different facial emotions from input images. The CNN is the greatest approach for computer vision applications for image classification, and it has been utilized as the Deep Learning algorithm for the model construction. Without CNN, filters for image processing techniques like smoothing, sharpening, and edge detection will need to be developed further, increasing the process's complexity.

By using convolutional filters, nonlinear activation functions, pooling, and back propagation, CNNs may learn filters that can recognize edges and blob-like structures in lower-level layers of the network. In addition, CNNs can use edges and structures as "building bricks" for detecting high-level objects in the network's deeper layers, such as facial expressions.

The method of applying lower-level layers to learn high-level features is accomplished by stacking a planned collection of layers. These various sorts of layers, often known as the building blocks of a CNN, will be described in the next section.

2.1.3 Building Blocks of a CNN

Since each layer in a feed forward neural network is a fully-connected network, it does not scale well as the size of images grows. It also leaves a lot to be desired in terms of accuracy. A CNN, on the other side, specifies a network architecture in a more logical manner.

The layers of a CNN, dissimilar from a traditional neural network, are placed in a 3D volume in width, height, and depth, where depth refers to the volume's third dimension, including the number of channels in an image or the number of filters in a layer.

A feed forward neural network with a color image of size 24x24 pixels as its input may describe the volume idea. The total number of inputs to the network will be 24x24x3 = 1,728, where 3 seems to be the number of RGB channels in a color image.

Even though this is a small amount for a neural network, if images with a resolution of 500x500 pixels are used, the total inputs to the network will be 500x500x3 = 4,50,000. The other connecting lines in the network's hidden layers and output layer are not included in these values. As a consequence, these variables might soon build up, resulting in poor performance.

on the contrary, the input volume for a CNN will have dimensions of width, height, and depth, respectively, 24x24x3. . Following layers neurons will only connect to a tiny proportion of the layer before them (rather than the fully-connected structure of a conventional neural network) – this is known as local connectivity, and it allows the network to save a large amount of parameters.

Finally, the output layer which will be an 1x1xN volume representing the image as a single vector of class scores. In this thesis, N = 7 which identifies seven different emotions, resulting in a volume of 1x1x7.

In a CNN, there are various types of layers. The following are the most frequent layers of a CNN.

- Convolutional Layer
- Activation Layer
- Pooling Layer
- Fully-Connected Layer
- Batch Normalization Layer
- Dropout Layer

A CNN architecture is nothing more than a specific stacking of these layers. A CNN architecture is frequently defined as a collection of input layers, convolution layers, activation layers, and fully connected layers. A simple CNN is defined here, which accepts an input, then applies a convolution layer, an activation layer, and finally a Fully-Connected layer.

The parameters learned during the training phase are found in the Convolutional, Fully-Connected, and Batch Normalization layers. Convolutional, Fully-Connected, Activation, and Pooling layers are the most significant for defining the real network structure among these layers.

2.1.4 Convolutional Layers

Convolutional Layer is the first layer of a CNN. This called the core building block of a CNN. It is composed of a collection of programmable filters or kernels. These kernels are typically small, like 3x3 or other squared-dimension lengths. These filters are applied to the input image in a sliding motion. A filter/kernel is also a collection of numbers, basically an array known as weights or parameters.

One point to remember is that the filter's depth must match the input's depth. In the upper left corner of (Figure 2.2), it can be seen that the filter is multiplying the values in the filter with the original pixel values of the image as it slides, or convolved, around the input image (aka computing element wise multiplications).

The result of all of these multiplications is a single number, which is the final output of the convolution process. The number represents the position of the filter in the image when it is at the top left. For each place on the input volume, the process is repeated. The next stage is to move the filter to the right by one or two units, known as a stride, then back to the right by one or two units, and so on.

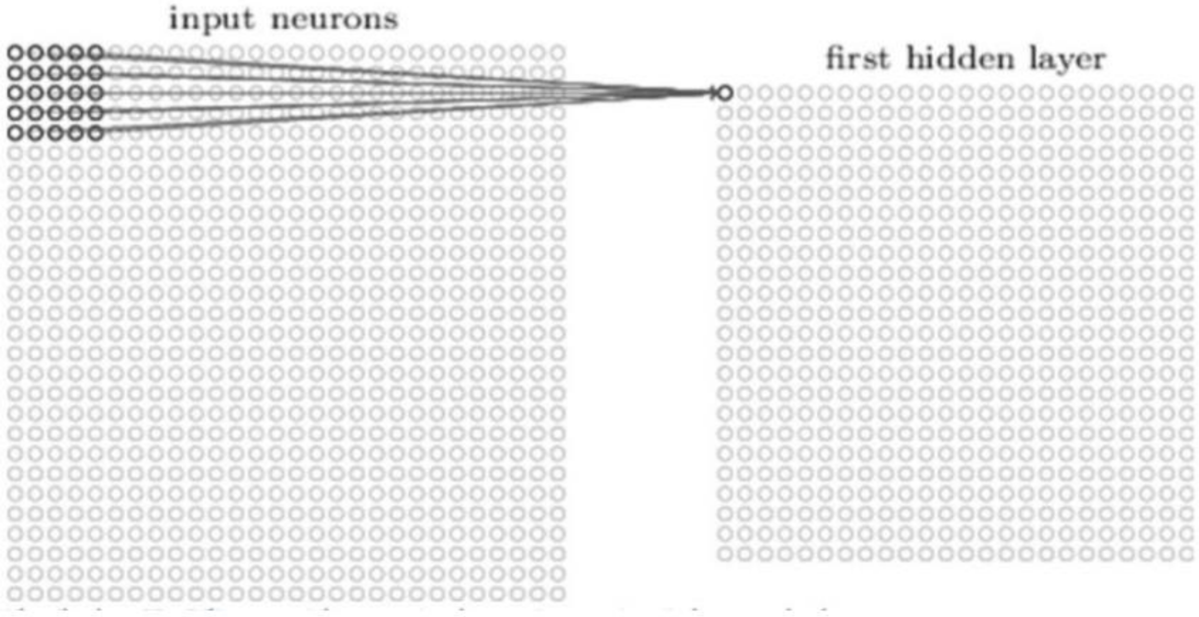


Figure 2.2 : Convolution operation of a CNN

Since application of all N-filters to the input volume, there are N, 2-dimensional activation maps. The final output volume is generated by stacking these N activation maps along the depth dimension of our array, as shown in (Figure 2.3).

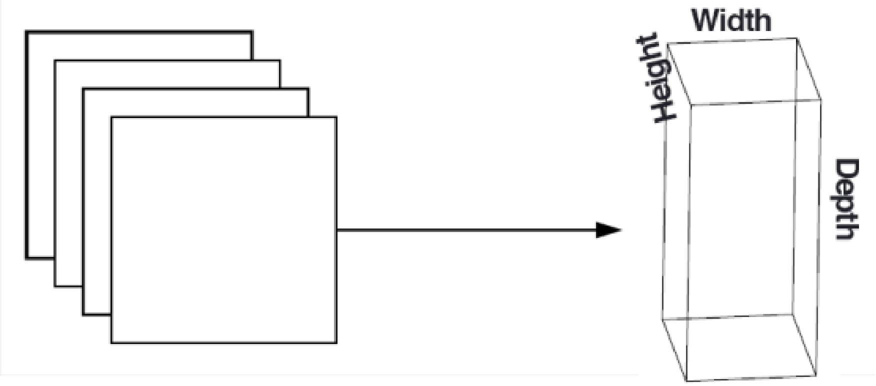


Figure 2.3 : After obtaining the N activation maps, they are stacked together to form the input to the next layer [30]

"Each entry in the output volume is thus the result of a neuron that only 'looks' at a small portion of the input "In this approach, the network learns filters that activate when they see a given type of feature at a specific spatial position in the input volume. Filters in lower layers of the network may activate when they see edge-like or corner-like regions. Then, when high-level elements like portions of the face, a dog's paw, a car's hood, and so on are present in the network's lower layers [30]

2.1.5 Activation Layers

An Activation layer is added to the CNN after each Convolutional layer. Nonlinear functions for example Sigmoid, Rectified Linear Unit (ReLU) , Exponential Linear Unit (ELU) , and others are used for the activation layers. The facial emotion recognition model was created using ReLU and ELU for model development. Figure 2.4 depicts plots of various activation functions along with their related mathematical functions.

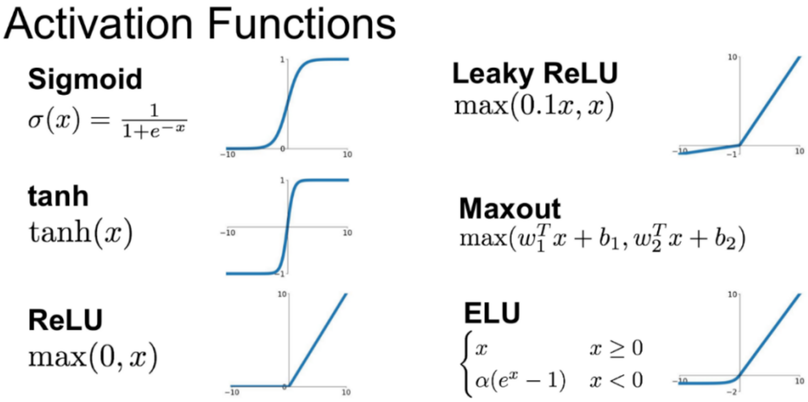
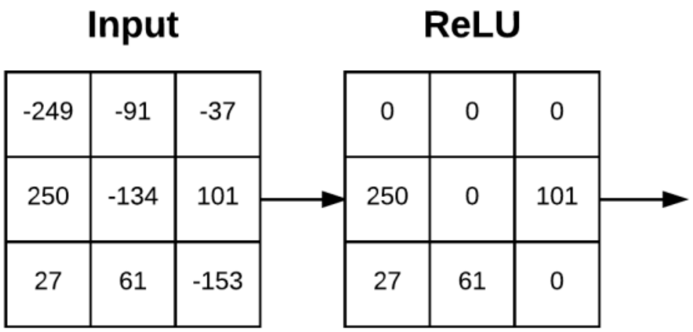


Figure 2.4: Plots of different activation functions

" An activation layer accepts a Winput x Hinput x Dinput input volume and then applies the provided activation function as shown in Figure 13. The activation function is applied element by element, and an activation layer's output is always the same as the input dimension, Winput = Woutput, Hinput = Houtput, Dinput = Doutput." [30]



Input

-249	-91	-37
250	-134	101
27	61	-153

ReLU

0	0	0
250	0	101
27	61	0

Figure 2.5 : An example of an input volume going through a ReLU activation, max (0, x) [30]

2.1.6 Pooling Layers

Pooling layers are generally applied to diminish the spatial size of an input volume which are placed in between consecutive convolutional layers. Pooling helps to control overfitting by decreasing the amount of parameters and operations in the network. Pooling layers use the max or average function to operate with each of the depth pieces of an input simultaneously.

Max Pooling is most often done in the middle of the CNN architecture to diminish spatial size, whereas average Pooling is generally utilized as the network's ultimate layer, including in GoogLeNet, SqueezeNet, and ResNet, where FC layers are omitted entirely. Figure: 2.6 depicts a pooling operation. As shown in the diagram, the size of the pooling layer and the stride affect the input size.

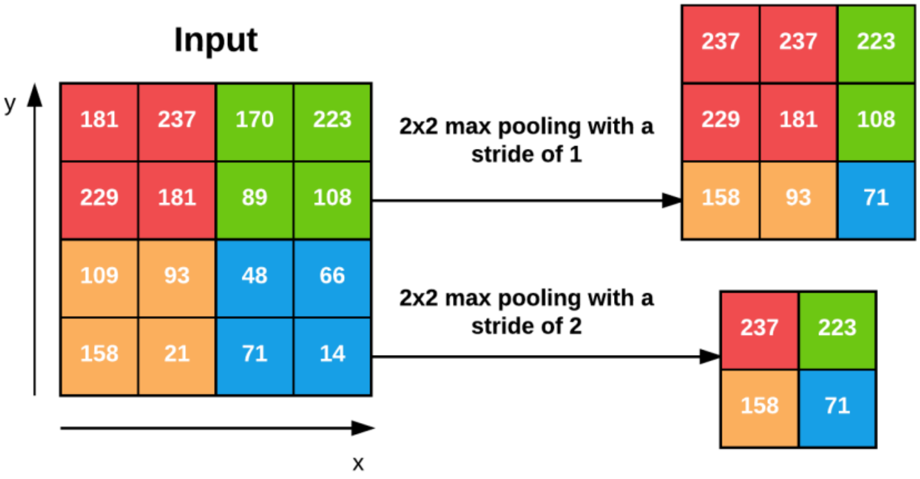


Figure 2.6: An example of a Pooling operation with different strides

2.1.7 Fully-Connected Layers

In a fully-connected layer, neurons are fully connected to all activations in the previous layer. Layers which are fully connected are always placed at the network's end.

This layer receives an input volume (whatever the result of the convolutional layer, Relu, or pooling layer before it is) and outputs an N-dimensional vector, where N is the number of classes the program must realize.

The model will be developed using numerically ordered classes that indicate Angry-0, Disgust-1, Fear-2, Happy-3, Sad-4, Surprise-5, Neutral-6)

2.1.8 Batch Normalization

Before sending a specific input volume within the next layer in the network, batch normalization layers are applied to normalize its activations. If xx considered of as a mini-batch of activations, the normalized xx may be calculated using the equation:

$$\hat{\boldsymbol{x}} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \qquad (1)$$

Here, respectively mean and variance are $\mu\beta$ and $\sigma^2\beta$, over each mini-batch of training images, β. The error ε is set equal to a tiny positive value in the range of 1e-7 for avoiding divide by zero. Applying (1) implies that the activations leaving a Batch Normalization layer will have approximately zero mean and unit variance (i.e., zero-centered).

During testing time, the mini-batch $\mu\beta$ and $\sigma^2\beta$ are replaceable with running averages of $\mu\beta$ and $\sigma^2\beta$ computed at the training process. This gives the surety that through the network images can pass and attain an accurate prediction without being biased by the $\mu\beta$ and $\sigma^2\beta$ from the final mini-batch passed through the network during the training phase [30].

Batch normalization has proven as incredibly beneficial in the network for the following effects.

- It decreases the number of epochs required for a NN to train.

- It allows for a wide range of learning rate and regularization techniques, which helps to stabilize training.

- It aims to reduce loss and gives a consistent loss curve.

2.1.9 Dropout

Dropout is a type of variational that seeks to prevent overfitting by enhancing testing accuracy, even at the expense of training accuracy. Dropout layers, with probability p, randomly detach inputs from the preceding layer to the next layer in the network architecture for each mini-batch in the training set [30]. Figure 2.7 demonstrates a dropout operation with a value of 0.5, which drops 50% of the connections from the previous layer.

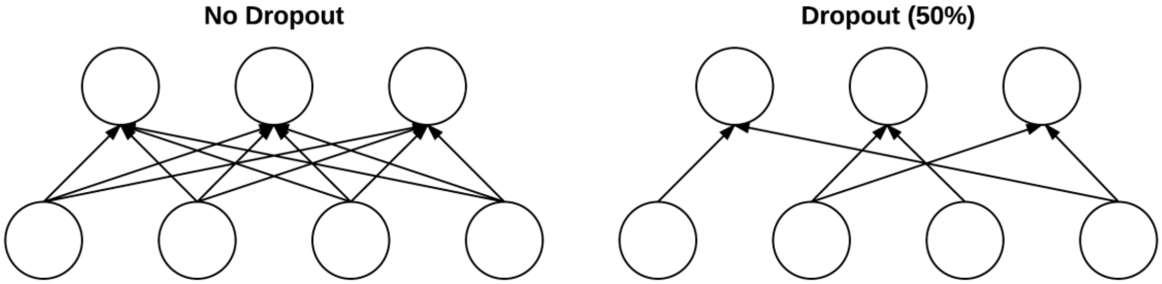


Figure 2.7: Effect of no Dropout (left) and Dropout with a value of 0.5 (right)

Dropping few connections to the preceding layer guarantees that there are numerous nodes instead of one when confronted with a given pattern, for example, filter, which helps the model generalize by decreasing overfitting.

In this section of Convolutional Neural Network (CNN), The importance of a CNN in image recognition is explored, as well as why it is the optimal approach for face expression detection. Finally, the components of a CNN are dissected thoroughly.

2.2 Machine Learning

In general, Machine learning algorithms are classified into four types.: supervised, unsupervised, semi-supervised, and reinforcement learning. The supervised learning category encompasses the Machine Learning algorithm presented in this thesis. As a consequence, supervised learning will be briefly described here.

Supervised learning is the Machine Learning process of learning a function that maps an input to an output which is basically based on an example of input-output pairs . It combines labeled training data and a collection of training examples to presume a function . A Machine Learning Algorithm is provided labeled training data consisting of both a set of inputs and target outputs in supervised learning. The algorithm then attempts to discover patterns that may be utilized to map input data points to their correct target outputs autonomously.

For example, a given dataset containing labeled images of human facial emotions, the algorithm can learn to anticipate and categorize unseen images of human facial emotions into the related classes (labels). This is precisely what this thesis has done. Another example is having a large number of molecules and knowing information about which ones are drugs, and then training a model to determine whether a new molecule is also a drug.

To quantify the contents of an image in earlier, hand-engineered features were utilized. In the past, raw pixel intensities were hardly utilized as inputs to Machine Learning models, but this has changed with the emergence of Deep Learning [30]. Initially, the feature extraction procedure was performed, which is also referred to as the process of taking an input image. The extracted features were evaluated using some algorithm. A feature extractor or image descriptor was the term for this sort of method. Finally, the feature extraction algorithm generated a vector, or a list of numbers, which was used to quantify the contents of an image. These emphasized vectors were then fed into a Machine Learning algorithm as input. To extract features from an image, Deep Learning does not utilize any hand-defined algorithms; rather than, the Deep Learning model learns these features naturally throughout the training process. A DL model, as previously mentioned, contains several hidden layers. A simple linearisation neural network with two hidden layers is shown in (Figure 2.8).

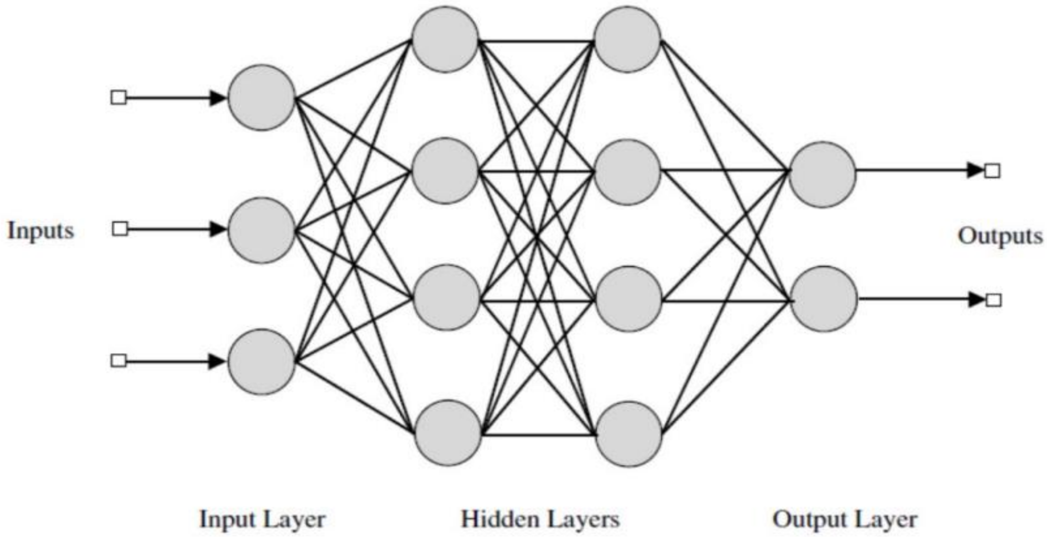


Figure 2.8: Linearisation of Neural network with two hidden layers of a DL model.

The learning process of a DL model can be described in terms of hierarchy.

To extract features from an image, a sequence of hidden layers is required. These hidden layers are built in a hierarchical order from the input layer to the output layer. The hidden layer's bottom layer normally learns to identify edges, shapes, and regions, and then combine them to generate contours and corners. These contours and corners, in turn, produce abstract object portions of the image in the next layer. Filters completed the learning process which are learnt automatically by the model. Finally, a classification algorithm is utilized in the output layer to categorize the image and attain the output class label.

The output of the previous layers is used by each layer in the Deep Learning network to construct more conceptual features of an image, and these features and layers are learned autonomously. As a result, the learning method is known as hierarchical learning.

A comparison between classic image classification algorithms using Machine Learning and current DL algorithm is shown in (Figure 2.9).

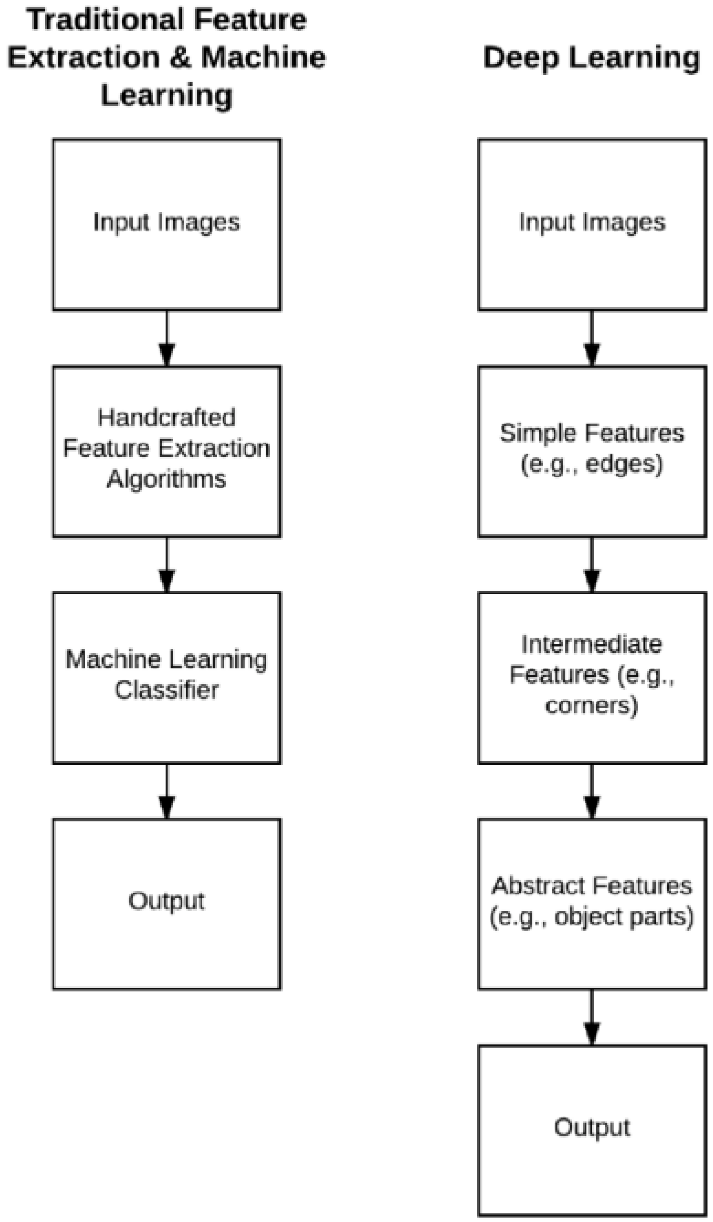

Traditional Feature Extraction & Machine Learning
Deep Learning
Input Images
Handcrafted Feature Extraction Algorithms
Machine Learning Classifier
Output
Input Images
Simple Features (e.g., edges)
Intermediate Features (e.g., corners)
Abstract Features (e.g., object parts)
Output

Figure 2.9: Comparisons between traditional Machine Learning Algorithm and DL algorithm [30]

2.2.1 Types of Machine Learning

In machine learning there are four different types of techniques. In our dataset, we have seven classes. Our training and validation datasets were all labeled with the relevant classes. As a consequence, we employed supervised learning strategy. For your convenience, we have included all four types of machine learning approaches below.

2.2.2 Supervised Learning

Supervised learning algorithms are capable of doing various analyses on new data based on what they have learnt previously, as well as predicting future events. The supervised learning algorithms construct a derived function for anticipating the initial analysis of known training data and output values. The system creates a goal for any new inputs after some effective training. The system may compare its output to the correct output and identify errors which need to be modified.

2.2.3 Unsupervised Learning

Unsupervised learning does not classify or label data sets. The system has no way of knowing what is about the ground. From unlabeled data, it can describe a secret shape. In this kind of learning process, the model investigates the data and tries to identify some structure in the dataset. This system is unable to give accurate results, but it is capable of making critical decisions based on the data set in order to describe a hidden shape from unlabeled data. The unsupervised learning method is very popular in market segmentation and classifying objects based on similarities.

2.2.4 Semi-supervised Learning

Semi-supervised Machine learning algorithms stand between supervised and unsupervised learning. Semi-supervised development employs both labeled and unlabeled data. Most of the cases, the amount of unlabeled data is greater than the number of labeled data. Unlabeled data is used because it is less costly and requires less efforts to process. Where supervised learning isn't an option in that case semi-supervised learning can be utilized alternatively. Using this strategy, the systems can improve their learning precision. Semi-supervised learning is ideal for this type of classification procedure. The model categorizes data based on a small quantity of labeled data while also categorizing a large amount of unlabeled data. When labeling data requires skilled and appropriate resources for training, semi-supervised learning techniques are generally utilized. While classifying a huge number of unlabeled data in a small amount of labeled data.

2.2.5 Reinforcement Learning

The system learns through trial and error in reinforcement learning. The agent (the learner or decision-maker), the environment (everything the agent interacts with), and actions are the three components of this learning process (what the agent can do). For right steps, the agent is rewarded; for wrong steps, the agent gets a penalty.

2.3 Deep Learning

Deep Learning is a subfield of Machine Learning whereas Machine Learning is a subfield of Artificial Intelligence (AI). Deep Learning basically trains computers to perform things that humans do instinctively.

This relationship can be best described by a graphical representation shown in (Figure 2.10).

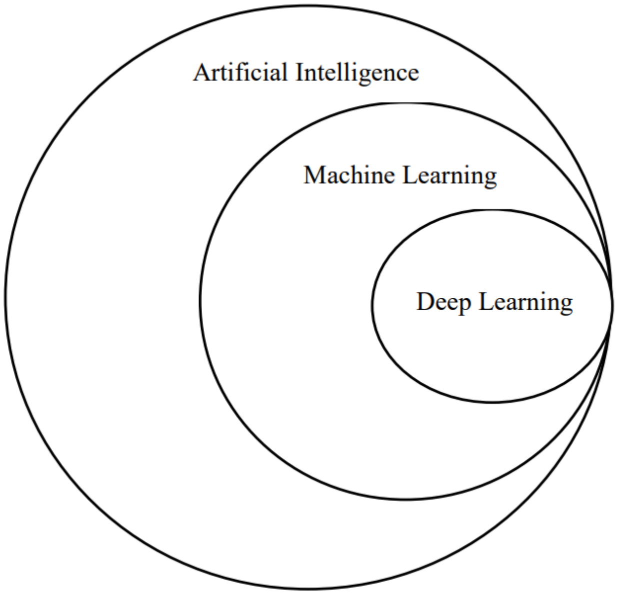


Figure 2.10 : A Venn diagram describing relationship between Deep Learning, Machine Learning and Artificial Intelligence.

Translating and recognizing the contents of an image has proven to be a challenging problem for a computer. Humans can execute the same action with ease and with little effort. The main objective of AI is to create a collection of algorithms that can be utilized to solve issues that humans can solve naturally and effortlessly but that are otherwise difficult for computers to solve [30].

Machine Learning is a broad subfield of AI focused with the area of research that permits computers to learn without being explicitly programmed [31]. This means that, once created, a single software will be capable of teaching itself how to perform intelligent tasks that are not related to programming. As a consequence, Machine Learning may be defined as a method for achieving AI.

This is just the exact form of how humans learn. The human brain learns the features of an object instantly, and when that thing is observed again, it may be recognized by a person based on memory and experience. The human brain is without a doubt one of the most capable machines for learning, thinking and solving problems. The neuron is considered to be the brain's fundamental computational component. Neurons interact with one another via the brain's basic working unit, a specialized cell that transmits data to other nerve cells, muscle or gland cells.

As demonstrated in (Figure 2.11), the complex connected network of neurons serves as the foundation for all decisions made depending on the diverse information acquired. Artificial Neural Networks (ANNs) are a simplified model of the human brain's neural system that perform similarly to the human brain. As illustrated in Figure 4, a basic ANN has an input layer, a hidden layer, and an output layer.

Deep Learning (DL) is a subset of neural networks that include more than three layers, for example, more than one hidden layer, inside the neural network domain. Deep Neural Networks (DNNs) are the neural networks utilized in DL [18]. The remainder of this chapter will go through when a neural network is considered "deep," the idea of "hierarchical learning," and when deep learning networks should be used for classification issues.

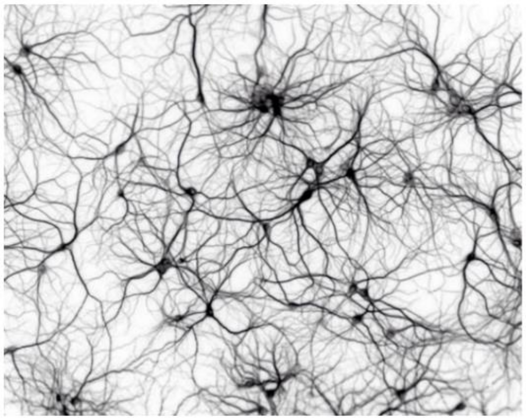


Figure 2.11 : Complex connected network of neurons of a human brain

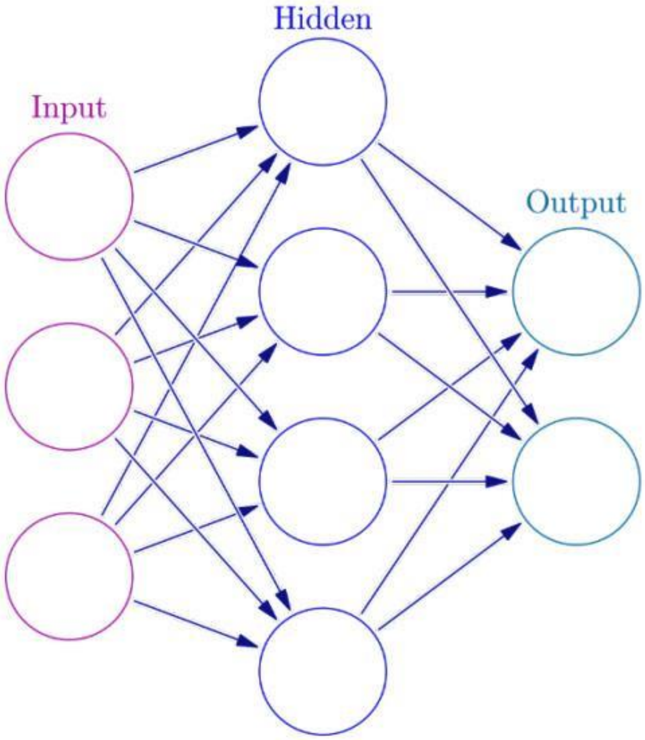


Figure 2.12: An Artificial Neural Network. Here, each circular node is an artificial neuron an arrow is a connection between nodes

2.3.1 Using Deep Learning

The term "deep" refers to the network's type and depth. For their special form, CNNs, Recurrent Neural Networks, and Long-Short Term Memory (LSTM) networks are regarded as DNNs. When the depth of a neural network is greater than two, it is referred to as a DNN. The number of hidden layers in the network is referred to as the depth in this case. The network is said to be very deep if its depth is greater than 10. In contrast to standard Machine Learning techniques, as the depth of a DNN increases, classification accuracy increases as well. Figure 8 depicts this tendency using a figure based on Andrew Ng's 2015 lecture, What Data Scientists Should Know About DL [25]. The plot shows that as the amount of training data increases, DNN accuracy enhances as well, but traditional Machine Learning techniques come to a halt. Large amounts of data are associated with DL for this reason. If the dataset comprises more than 1000 images, it is thought that image categorization will be more difficult, DL will undoubtedly beat other Machine Learning methods such as logistic regression, SVMs, decision trees, and so on.

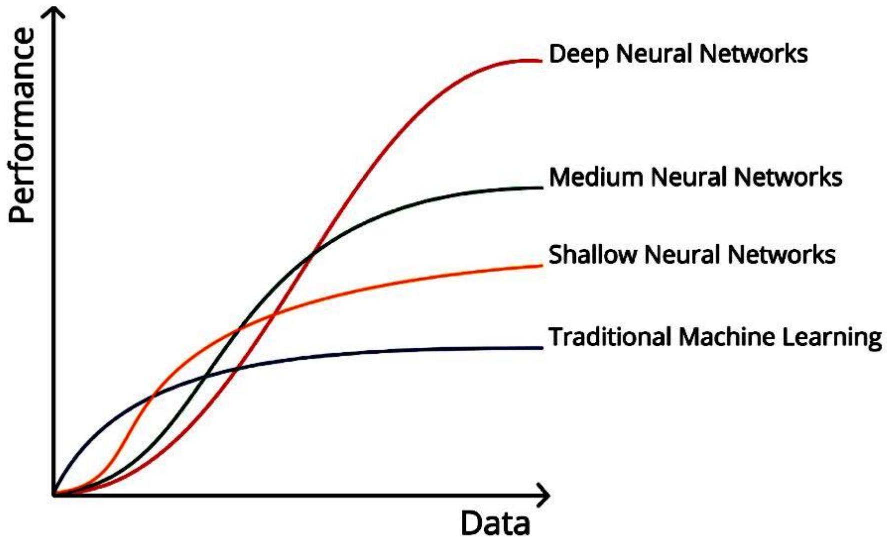


Figure 2.13: performance and training data for different neural networks

The fundamentals of DL are presented in this chapter. It belongs to the Artificial Neural Networks (ANN) family (ANNs). In compared to typical Machine Learning techniques, the utilization of DL and hierarchical learning of DL networks are also discussed.

2.4 Transfer Learning

Transfer Learning can train deep neural networks with a limited quantity of data, transfer learning is very useful in deep learning. The reuse of a pre-trained model on a new issue is referred to as transfer learning in machine learning. In this learning, a machine uses previous activity knowledge to enhance generalization about a new task. This is especially valuable in the field of data science, because most real-world scenarios do not necessitate millions of labeled data points for complex models to be trained. In transfer learning, we don't begin learning from scratch, we rather start with patterns which are learned from solving related tasks. Because of the huge amount of Computational power required, transfer learning is basically employed in computer vision and natural language processing applications like sentiment analysis. Neural networks are used in computer vision to recognize edges in the first layer, forms in the middle layer, and task-specific properties in the latter layers. The early and intermediate layers are utilized in transfer learning, whereas the latter layers are just retrained. It makes use of the labeled data from the task it was initially trained on. In transfer learning, we strive to transfer as much information as possible from the prior task that the model was trained on to the new task at hand. Depending on the problem and the data, this knowledge might take many different forms. For example, it may be the way models are constructed, which makes it easier to recognize new objects.

2.4.1 Necessity of Transfer Learning

Transfer learning has a number of benefits, the most prominent of which are diminish training time, increased neural network performance (in most circumstances), and the elimination of the need for a large amount of data. A substantial amount of data is usually necessary to train a neural network from start, but access to such data isn't always feasible. This is when transfer learning is beneficial. Transfer learning can be used to build a powerful machine learning model with relatively minimal training data because the model has already been pre-trained. This is particularly useful in natural language processing, where large labeled datasets need a great deal of expertise. Furthermore, training time is reduced because constructing a deep neural network from the ground up is a difficult task that might take days or even weeks.

2.4.2 Usage of Transfer Learning

It's difficult to develop generally applicable rules in machine learning, however here are some suggestions for when transfer learning could be useful:

- There isn't sufficient labeled training data to start from laceration and train your network.
- There is already a network that has been pre-trained on a similar task, and it is usually trained on large amounts of data.
- When the input for task 1 and task 2 is the same.

If the unique model was trained using TensorFlow, you can simply reinstate it and perpetuate some layers for your task

It's important to remember that transfer learning only works if the features learned in the first task are generic, i.e., they can be appeal to other tasks. Moreover, the input to the model must be the same size as when it was first trained. Add a pre-processing step to resize your input to the required size if you don't already have one.

We have used transfer learning models such as VGG19, Resnet-50V2, MobilenetV2.

2.4.3 VGG-19

VGG19 is a 19-layer deep Convolutional Neural Network design. The main goal for which the VGG net was developed was to win the ILSVRC ImageNet competition.

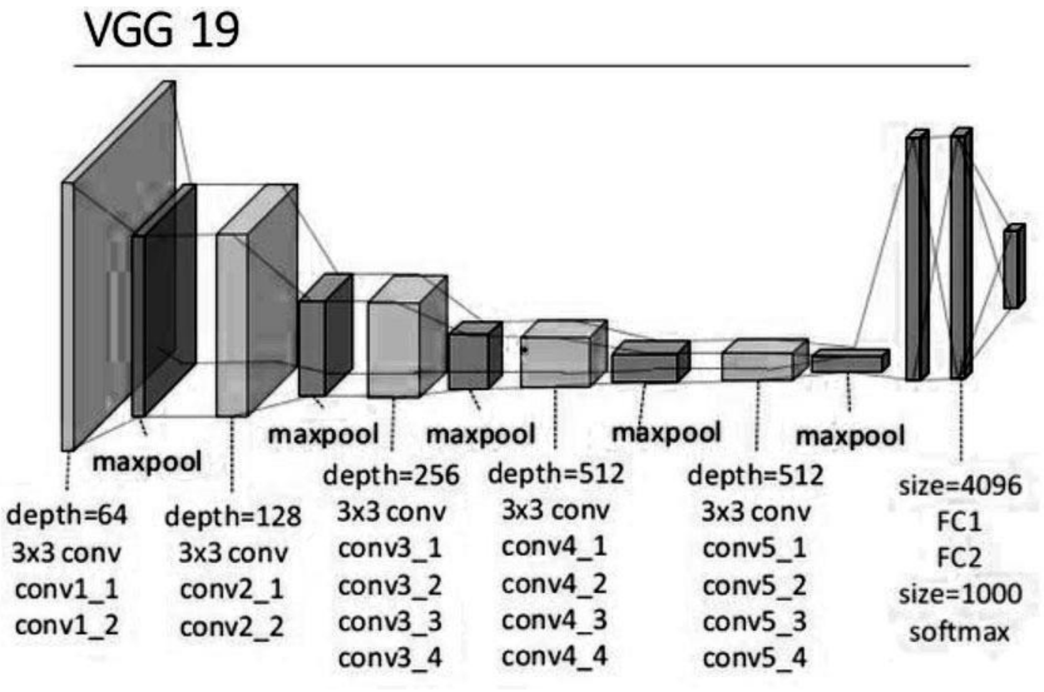


Figure 2.14 : VGG-19 Architecture

Let’s take a brief look at the architecture of VGG19 to understand how it works

- **Input:** At first, The image input size for the VGG19 is 224x224 pixels.

- **Convolutional Layers**: VGG uses a narrow receptive field in its convolutional layers. i.e. 33%, the cramped size achievable while still capturing up/down and left/right gesture. A ReLU activation function is also available. Rectified Linear Unit activation function (ReLU) is a holomorphic linear function that outputs the input if it is positive and zero if it is negative. To maintain spatial resolution after convolution, the stride is set to 1 pixel.

- **Fully-Connected Layers:** The VGG19 is made up of three layers that are all connected. The first two levels have 4096 nodes each, whereas the third layer has 1000 nodes, which is the entire number of classes in the ImageNet dataset.

2.4.4 Resnet-50V2

ResNet-50V2 is a deep convolutional neural network with 50 layers. A pre-trained version of the network has been trained on over a million pictures and is available in the ImageNet database. The network was pre-trained to recognize 1000 different object categories, including keyboards, mice, pens, and a variety of animals. We can see the architecture of ResNet-50V2 below (Fig 2.15).

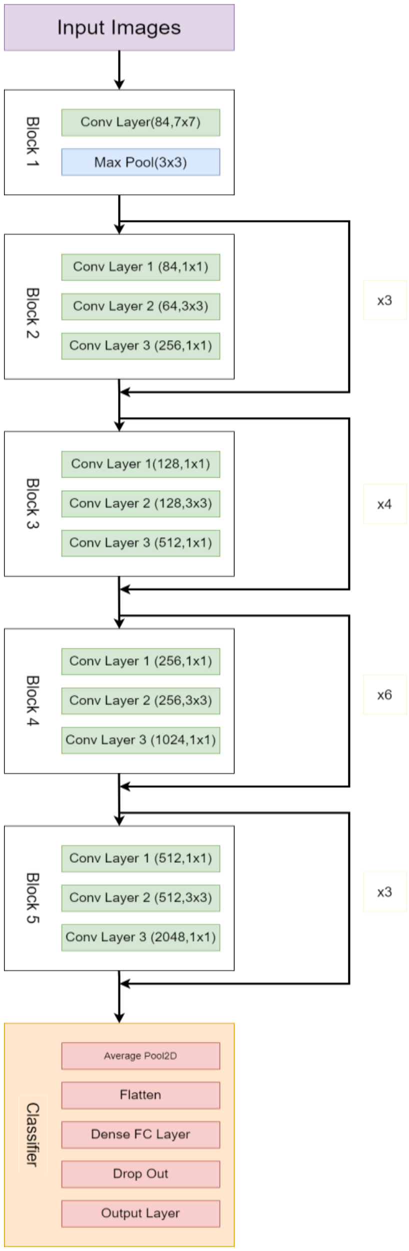


Figure 2.15 : ResNet-50V2 Architecture

2.4.5 MobileNetV2

MobileNetV2 is a convolutional neural network architecture. It aspires to be user-friendly on mobile devices. It's based on an inverted residual structure, with residual connections connecting bottleneck levels. Lighter depth wise convolutions are used as a source of non-linearity in the intermediate expansion layer filters. MobileNetV2's architecture includes a fully convolutional layer with 32 filters, followed by 19 residual bottleneck layers. (Fig 2.16).

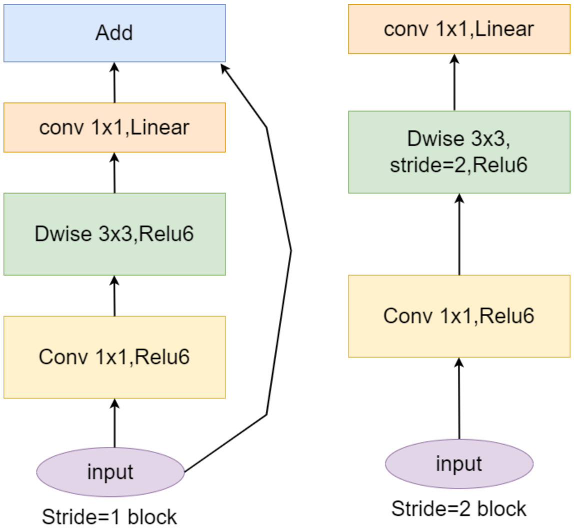


Figure 2.16 : MobileNetV2 Architecture

2.5 Basic ML Paradigm

- Collecting data such as images, text, and so on.
- "Training data" can be used to create a model.
- "Validation data" can be used to fine-tune the model.
- "Testing data" can be used to test the model.
- Make predictions using the model.

2.6 Environment Setups

2.6.1 Keras

Keras is a Python-based machine learning framework which is basically built on top of TensorFlow. It was designed specifically for deep learning applications. Keras is compatible with a variety of machine learning models and layers. One of Keras' most fundamental models is the sequential model. Keras may also be used to create layers like the Dense layer, flatten layer, convolutional LSTM, dropout layer, and so on. It supports a number of optimizers, including Adam, SGD, and others, allowing us to customize our optimizer to our specific needs. Keras can also be used to train, evaluate, and test models.

We needed Keras for importing sequential models, ResNet-50V2 models, average pooling layer, dense layer, dropout layer, and optimizer routines as well as other things.

2.6.2 TensorFlow

TensorFlow is an open-source toolkit that performs complicated computations to speed up and simplify machine learning. TensorFlow was built by the Google Brain team. Its framework is designed in the C++ and Python programming languages. Model training, data acquisition, prediction, and refining processes are all made easier using TensorFlow. TensorFlow may be used to quickly implement machine learning models and algorithms in applications like handwritten digit categorization, image recognition, natural language processing, and more.

We used Keras in our model, however Keras is generally built on TensorFlow. TensorFlow was deployed as a solution. When we use Keras, TensorFlow is constantly operating on the backend.

Chapter 3
Related Works

3.1 Introduction

The performance of recognizing human facial expressions and the technical creation of a model with CNN to classify distinct face expressions are the subjects of this thesis. Face expression recognition and deep learning models for facial expression classification were used to build the thesis concept. By combining these two components, the novel idea for this thesis was born, allowing different features and reasons of the thesis to be better comprehended.

3.2 Facial Expression Recognition based literature

In the last few years, deep learning in CNN has made great progress in recognizing human facial expressions. The authors of [11] examined the present state of the art in image-based facial expression recognition with CNNs, with emphasis on algorithmic variants and their impact on performance. They exhibited that removing one of these bottlenecks – the relatively simplistic designs of the CNNs utilized in this field – improves expression recognition performance significantly. By developing an ensemble of current deep CNNs, the authors were able to achieve a FER2013 test accuracy of 75.2 percent, outperforming previous work that did not include extra training data or face registration. The authors used multiple deep learning CNN advances to identify the key seven human emotions: angry, disgust, fear, happy, sad, surprise, and neutral as part of another project [12]. They used ensemble and transfer learning approaches to attain the best results using the FER2013 dataset. The authors were able to achieve an accuracy of 67.2 percent using ensemble learning and 78.3 percent utilizing transfer learning. The winner of the Kaggle Facial Expression Recognition Challenge had a 71.2 percent accuracy rate, which was great. Those who placed in the top ten of the same competition had accuracy rates of roughly 60% on average. The authors from [13] obtained test accuracies of 74.79 percent and 95.71 percent using visual salience and deep learning on the Compound Facial Expressions of Emotion Dataset (CFEE) and Radbound Faces 10 Database (RaFD) datasets. The authors offered strategies for refining individual models and integrating several CNNs in their work published in [14], and their Linear SVM methodology generated considerable end results, with a high-test accuracy of 87.4 percent. The author makes appropriate use of transfer learning approaches to handle emotion recognition. Resnet50, vgg19, Inception V3, and Mobile Net pre-trained networks are chosen for this research.

The experiment was conducted by using the CK+ database and achieved an average accuracy of 96% for emotion detection problems. [26].

This paper extends the deep Convolutional Neural Network (CNN) approach to facial expression recognition task. This task is done by detecting the occurrence of facial Action Units (AUs) as a subpart of Facial Action Coding System (FACS) which represents human emotion. In the CNN fully-connected layers we employ a regularization method called "dropout" that proved to be very effective to reduce overfitting. The system performance gain average accuracy rate of 92.81%. [27].

This study proposed a Machine Learning (ML) approach based on a statistical analysis of emotion recognition using facial expression through a digital image. The analysis part was divided into two phases: firstly boosting algorithms-based ML classifiers (named as LogitBoost, AdaboostM1, and Stacking) which obtained 94.11%, 92.15%, and 89.21% accuracy, respectively. Secondly, decision tree algorithms named J48, Random Forest, and Random Committee were obtained with 97.05%, 93.14%, and 92.15% accuracy, respectively [28].

In the paper, a deep learning-based human emotion detection framework (DL-HEDF) has been proposed to evaluate the probability of digital representation, identification, and estimation of feelings. The proposed DL-HEDF analyzes the impact of emotional models on multimodal identification. The paper introduces emerging works that use existing methods like convolutional neural networks (CNN) for human emotion identification based on language, sound, image, video, and physiological signals. The output results are obtained as an analysis of the ratio of the facial expression of 87.16%, accuracy evaluation ratio being 88.7%, improving facial recognition ratio is 84.5%, and the expression intensity ratio is 82.2%. The emotional simulation ratio is 93.0% [29].

3.3 Existing System

1 . Multi-Modal Emotion Recognition, using EEG enabled Emotion Tracking and Speech Emotion Recognition [21]

The design of an affective service, that uses SER and EEG enabled Emotion Recognition to understand a person’s complex inner state, could pave the way for a new era of Human-Computer Interaction. multi-modal emotion recognition systems have been examined. Such can be visual and audio signals, as they are complementary to each other. Those systems combine features of different modalities to construct a new feature, that is then processed by the classifiers.

2. Music player based on emotion recognition of voice signals [22]

In this paper, a smart music system is designed by recognizing the emotion using voice speech signal as an input. The objective of the speech emotion recognition (SER) system is to determine the state of emotion of a human being's voice. This study recognizes five emotions- anger, anxiety, boredom, happiness and sadness.

3. Facial emotion recognition based on LDA and Facial Landmark Detection [23]

This paper is to extract facial image features by using Linear Discriminant Analysis (LDA) and Facial Landmark Detection after grayscale processing and cropping, and then compare the accuracy after emotion recognition and classification to determine which feature extraction method is more effective.

4. Facial Expression Recognition Based on Arousal-Valence Emotion Model and Deep L earning Method [24]

this paper uses the arousal-valence continuous emotion space model, which can enrich emotion expression. The arousal reflects emotional intensity, and the valence indicates positive and negative emotion. The arousal and valence all have the value in the same range, which is between -1 and 1. In the experiments, it uses convolutional neural network (CNN) in the pre-trained models and support vector regression (SVR).

5. Classroom monitoring system based on facial expression recognition [25]

Recognizing dynamic expressions of students in class, 8 kinds of emotions are selected for application: positive emotions: “happy”; negative emotions: “disgust, Sadness, doubts, contempt, anger”; neutral emotion: “focus, surprise”. In this design, the classroom performance scoring system in normal hours is split into four functions: wireless network list acquisition and verification, face recognition, emotion analysis, and scoring record storage. On this basis, SVM and Softmax are used.

Chapter 4

Datasets

4.1 Datasets

For this thesis, facial expressions were needed to be classified by a CNN model and for that it was essential to have datasets which contain images of seven different facial expressions. The facial expression recognition CNN model was trained using two datasets. The dataset is the first step in training a neural network; it defines our ultimate goals, along with the problem we're trying to address. The first dataset that was used in this thesis is the dataset used in the Kaggle's "Challenges in Representation Learning: Facial Expression Recognition Challenge" [17]. It is known as the FER2013 dataset and can be found in [18]. The collection includes 48x48 pixel grayscale photos of faces divided into seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set includes 28,709 distinct examples from each of the seven categories. There are 3,589 cases in the public test set utilized for the leaderboard. Another 3,589 samples make up the final test set, which was utilized to decide the competition's winner. A preview of images of the FER2013 dataset is shown in Figure (4.1)

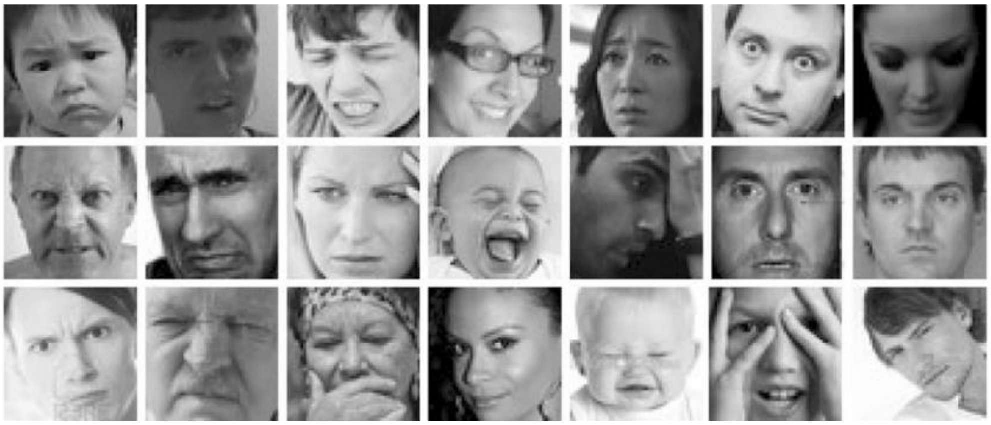
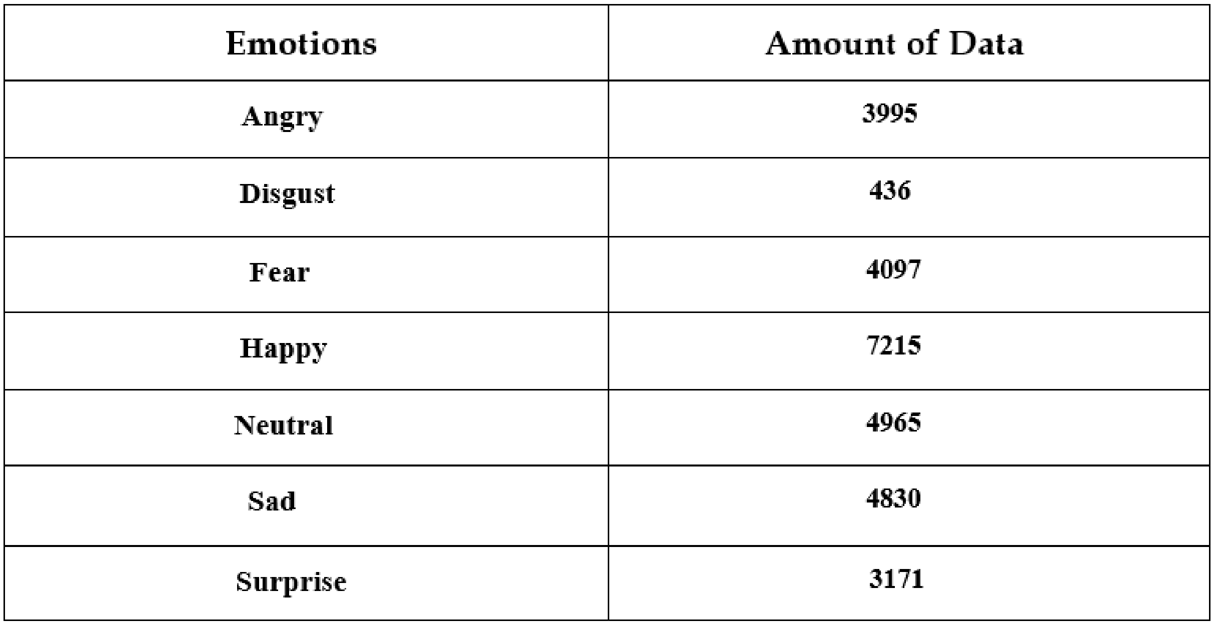


Figure 4.1 : Image contents of FER2013 dataset showing different emotions



Emotions	Amount of Data
Angry	3995
Disgust	436
Fear	4097
Happy	7215
Neutral	4965
Sad	4830
Surprise	3171

Table 4.1: Emotion labels and Amount of Dataset

4.2 Random Dataset for Test Sample

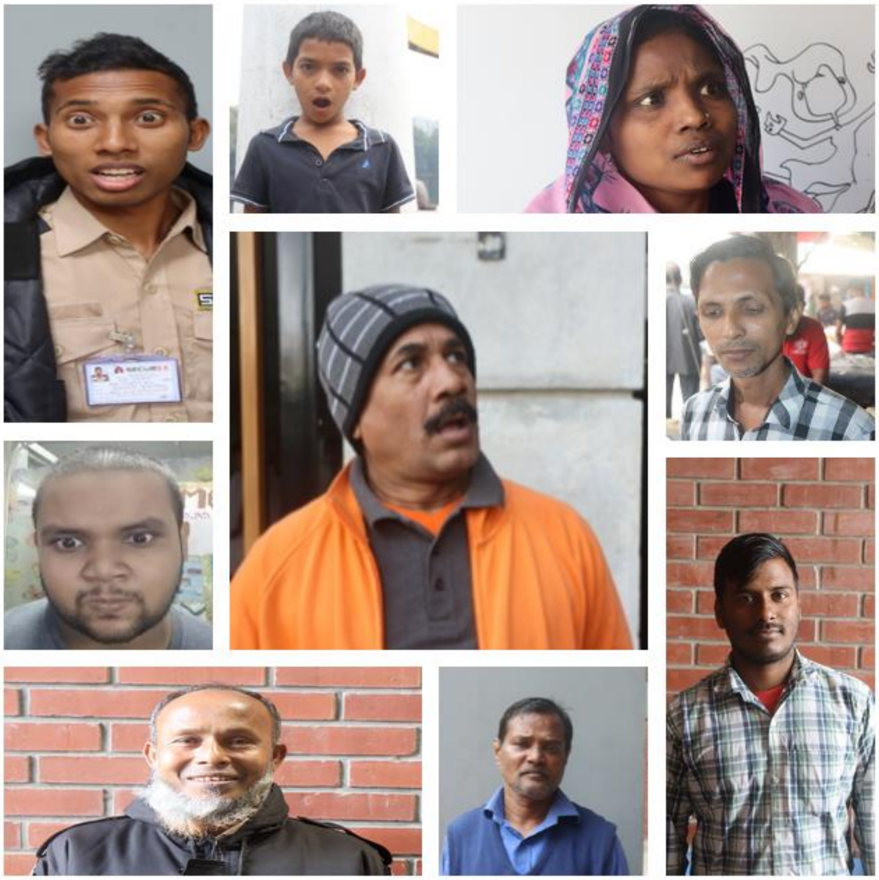
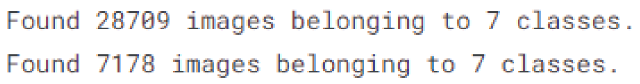


Figure 4.2: Collected Dataset

4.3 Data Preparation



```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

Figure 4.3: Total class and Dataset result

Chapter 5

Methodology

5.1 Methodology

In this chapter, we'll go through the framework of our suggested strategy as well as the methodologies we'll employ. The general structure of the approach is depicted in (Figure 5.1)

STEP 1

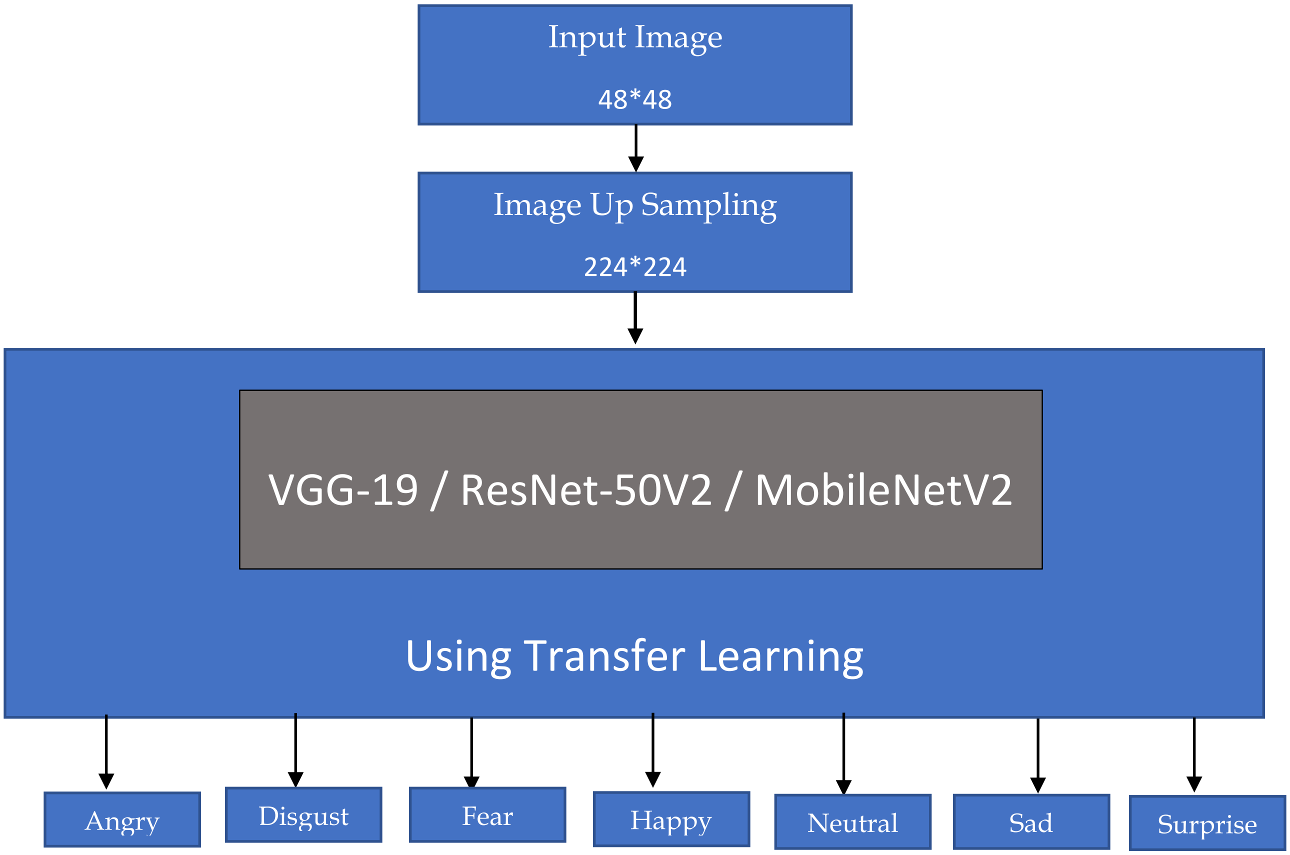


Figure 5.1a : Methodology

STEP 2

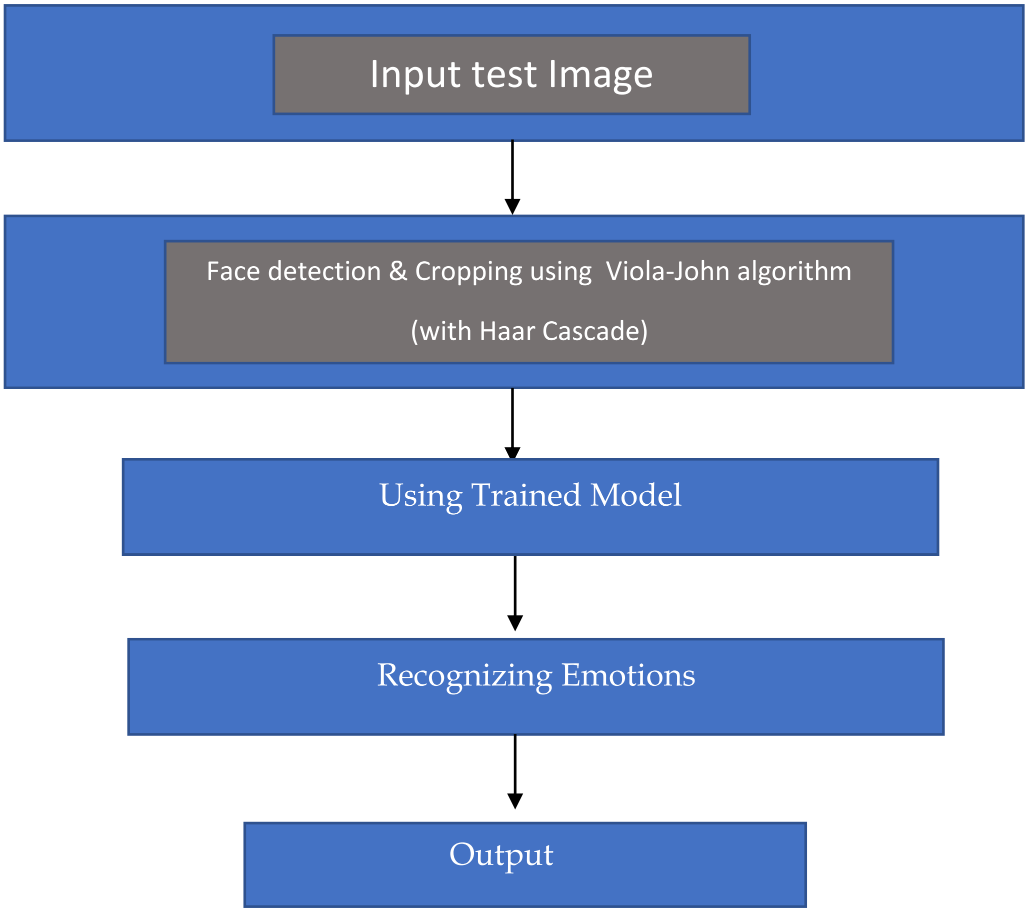


Figure 5.1b : Methodology

5.1.1 Description of Methodology

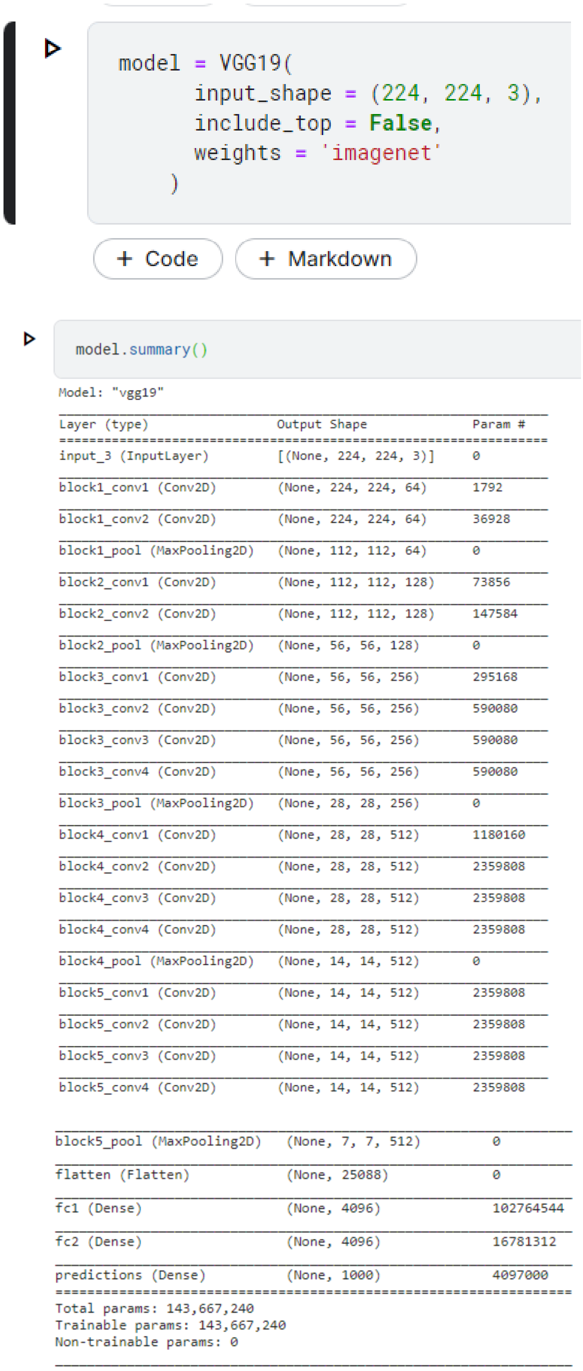
The face expression is used in our suggested method to detect seven emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. The work flow of the methodology starts with collecting and labeling datasets followed by specifying the CNN model architecture. In this (Figure 5.1) at first, we take input image from FER2013 dataset its image size 48×48 but our Keras applications size required 224 × 224 that’s why we up sampling it. After that we did transfer learning with different keras applications. Then for test image detection and cropping we use Viola-John algorithm (with Haar Cascade) Then Recognizing emotions and its Output.

5.1.2 Using Purpose of These Three Applications

Compared with VGG-16, this method has quicker training speed, fewer training samples per time, and superior accuracy. Secondly, VGG-16 has 16 layers and VGG-19 has 19 layers. As the number of layers in a CNN model raises, the model's capacity to fit more complex functions raises as well. As a result, more layers imply better performance. Now, ResNet50V2 is a revised form of ResNet50V2 that performs best on the ImageNet dataset than ResNet50 and ResNet101. We were searching for good keras applications for get good accuracy that is the on of the reason to choose ResNet50V2. And then MobileNetV2 was our first choice when we start work on this thesis, and also, we get good output from MobileNetV2 and accuracy also as you can see.

5.1.3 VGG-19

VGG-19 is one of the best-performing architectures that obtain a accuracy of 92.89%. The ImageNet challenge (ILSVRC) 1000-class classification job was used to train the VGG-19. The image input size for the VGG19 is 224×224 pixels. The VGG19 is made up of three layers that are all connected. The first two layers each have 4096 nodes, whereas the third layer has 1000 nodes, which is equal to the entire number of classes in the ImageNet dataset.



```
model = VGG19(
        input_shape = (224, 224, 3),
        include_top = False,
        weights = 'imagenet'
    )
```

```
model.summary()
```

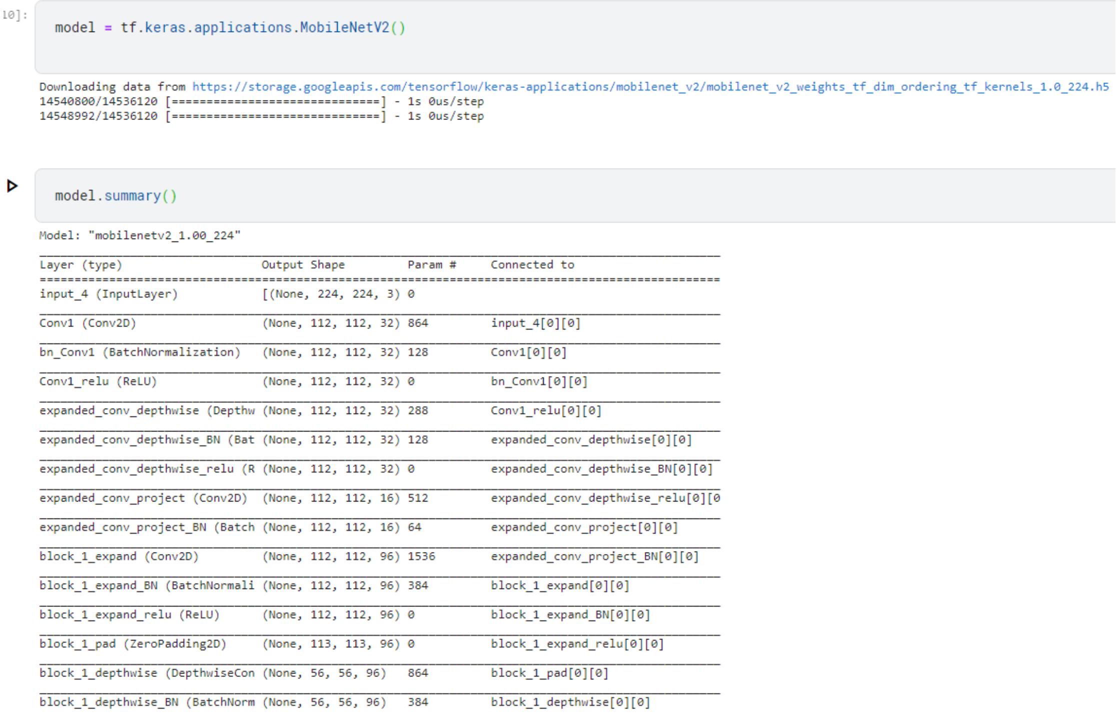
```
Model: "vgg19"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_3 (InputLayer)        [(None, 224, 224, 3)]     0

block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080

block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv4 (Conv2D)       (None, 14, 14, 512)       2359808


_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 143,667,240
Trainable params: 143,667,240
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.2 : Structure of VGG-19 model in code

5.1.4 MobileNetV2

In MobileNetV2, there are two genre of blocks. An one-march superfluous block is either one of them. A two-superfluous block is one more recourse for bashful. on that point three levels in both category of blocks.

This period, the rudimentary layer is 11 convolution using ReLU6. The subsequent layer is profundity convolution. One more 1×1 convolution is make use of in the third layer, nonetheless this time there is no inconsistent. If ReLU solicited afresh, on the report of the proclamation, deep networks will only have the potentiality of a linear morpheme on the non-zero volume part of the output domain.

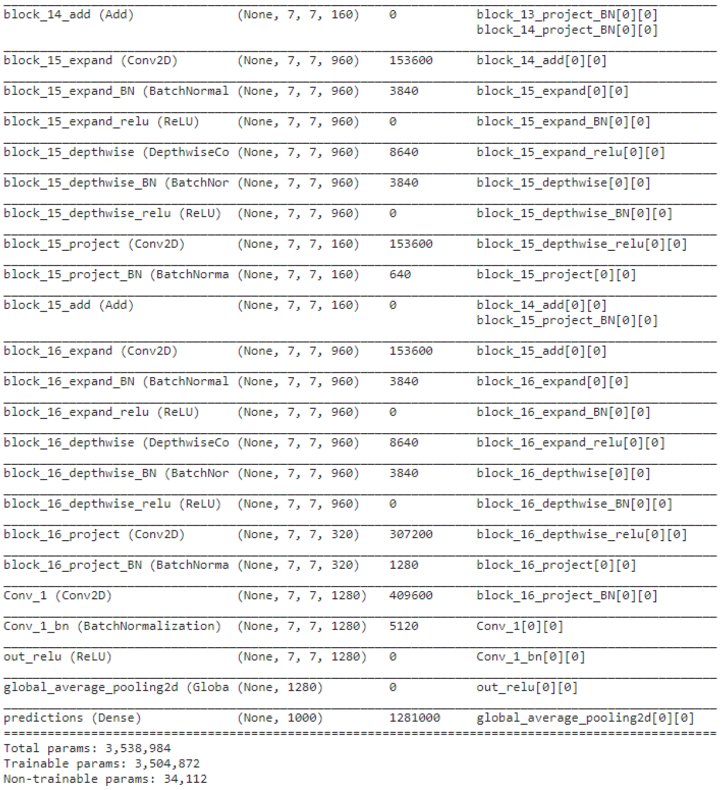


```
10]:  model = tf.keras.applications.MobileNetV2()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224.h5
14540800/14536120 [==============================] - 1s 0us/step
14548992/14536120 [==============================] - 1s 0us/step
```

```
model.summary()
```

```
Model: "mobilenetv2_1.00_224"
____________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
====================================================================================================
input_4 (InputLayer)            [(None, 224, 224, 3) 0
____________________________________________________________________________________________________
Conv1 (Conv2D)                  (None, 112, 112, 32) 864         input_4[0][0]
____________________________________________________________________________________________________
bn_Conv1 (BatchNormalization)   (None, 112, 112, 32) 128         Conv1[0][0]
____________________________________________________________________________________________________
Conv1_relu (ReLU)               (None, 112, 112, 32) 0           bn_Conv1[0][0]
____________________________________________________________________________________________________
expanded_conv_depthwise (Depthw (None, 112, 112, 32) 288         Conv1_relu[0][0]
____________________________________________________________________________________________________
expanded_conv_depthwise_BN (Bat (None, 112, 112, 32) 128         expanded_conv_depthwise[0][0]
____________________________________________________________________________________________________
expanded_conv_depthwise_relu (R (None, 112, 112, 32) 0           expanded_conv_depthwise_BN[0][0]
____________________________________________________________________________________________________
expanded_conv_project (Conv2D)  (None, 112, 112, 16) 512         expanded_conv_depthwise_relu[0][0
____________________________________________________________________________________________________
expanded_conv_project_BN (Batch (None, 112, 112, 16) 64          expanded_conv_project[0][0]
____________________________________________________________________________________________________
block_1_expand (Conv2D)         (None, 112, 112, 96) 1536        expanded_conv_project_BN[0][0]
____________________________________________________________________________________________________
block_1_expand_BN (BatchNormali (None, 112, 112, 96) 384         block_1_expand[0][0]
____________________________________________________________________________________________________
block_1_expand_relu (ReLU)      (None, 112, 112, 96) 0           block_1_expand_BN[0][0]
____________________________________________________________________________________________________
block_1_pad (ZeroPadding2D)     (None, 113, 113, 96) 0           block_1_expand_relu[0][0]
____________________________________________________________________________________________________
block_1_depthwise (DepthwiseCon (None, 56, 56, 96)   864         block_1_pad[0][0]
____________________________________________________________________________________________________
block_1_depthwise_BN (BatchNorm (None, 56, 56, 96)   384         block_1_depthwise[0][0]
```



```
__________________________________________________________________________________________________
block_14_add (Add)              (None, 7, 7, 160)    0           block_13_project_BN[0][0]
                                                                 block_14_project_BN[0][0]
__________________________________________________________________________________________________
block_15_expand (Conv2D)        (None, 7, 7, 960)    153600      block_14_add[0][0]
__________________________________________________________________________________________________
block_15_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_15_expand[0][0]
__________________________________________________________________________________________________
block_15_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_15_expand_BN[0][0]
__________________________________________________________________________________________________
block_15_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_15_expand_relu[0][0]
__________________________________________________________________________________________________
block_15_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_15_depthwise[0][0]
__________________________________________________________________________________________________
block_15_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_15_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_15_project (Conv2D)       (None, 7, 7, 160)    153600      block_15_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_15_project_BN (BatchNorma (None, 7, 7, 160)    640         block_15_project[0][0]
__________________________________________________________________________________________________
block_15_add (Add)              (None, 7, 7, 160)    0           block_14_add[0][0]
                                                                 block_15_project_BN[0][0]
__________________________________________________________________________________________________
block_16_expand (Conv2D)        (None, 7, 7, 960)    153600      block_15_add[0][0]
__________________________________________________________________________________________________
block_16_expand_BN (BatchNormal (None, 7, 7, 960)    3840        block_16_expand[0][0]
__________________________________________________________________________________________________
block_16_expand_relu (ReLU)     (None, 7, 7, 960)    0           block_16_expand_BN[0][0]
__________________________________________________________________________________________________
block_16_depthwise (DepthwiseCo (None, 7, 7, 960)    8640        block_16_expand_relu[0][0]
__________________________________________________________________________________________________
block_16_depthwise_BN (BatchNor (None, 7, 7, 960)    3840        block_16_depthwise[0][0]
__________________________________________________________________________________________________
block_16_depthwise_relu (ReLU)  (None, 7, 7, 960)    0           block_16_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_16_project (Conv2D)       (None, 7, 7, 320)    307200      block_16_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_16_project_BN (BatchNorma (None, 7, 7, 320)    1280        block_16_project[0][0]
__________________________________________________________________________________________________
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]
__________________________________________________________________________________________________
predictions (Dense)             (None, 1000)         1281000     global_average_pooling2d[0][0]
==================================================================================================
Total params: 3,538,984
Trainable params: 3,504,872
Non-trainable params: 34,112
```

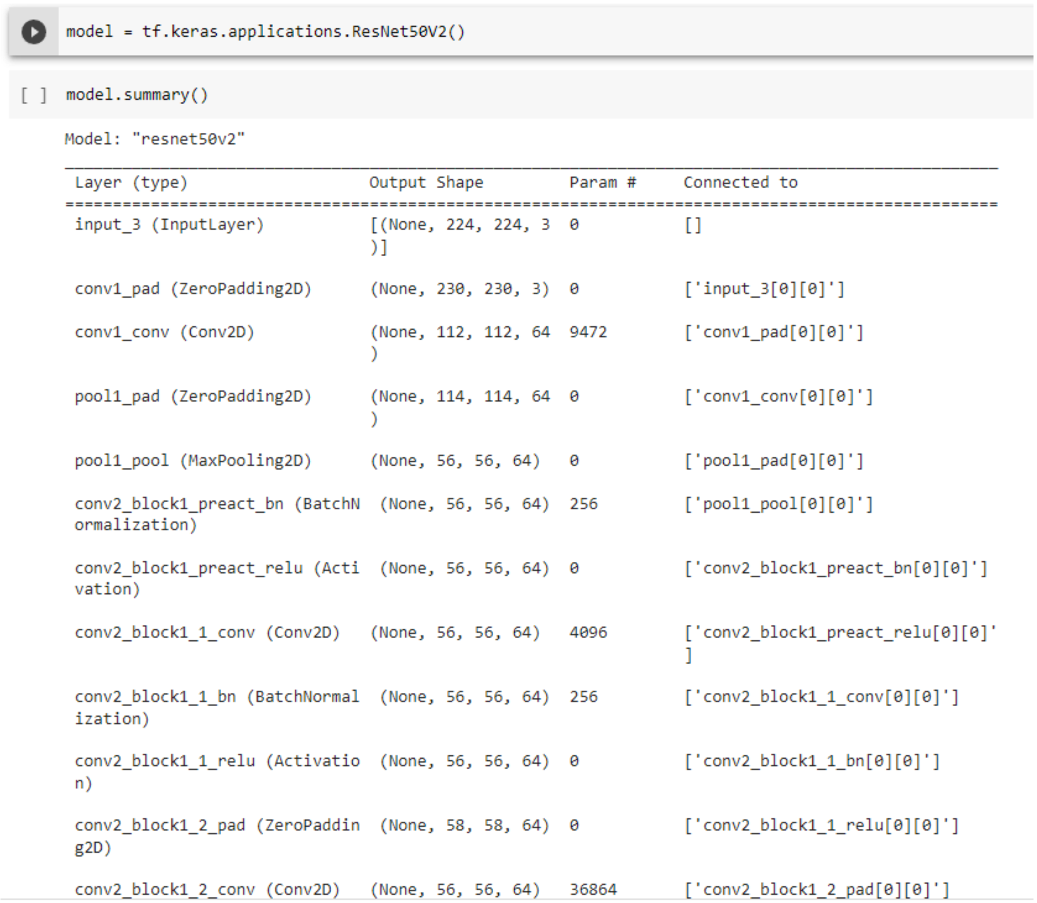
Figure 5.3 : Structure of MobileNetV2 model in code.

5.1.5 ResNet50V2

After AlexNet obtained leading position in the LSVRC2012 classification challenge in 2012, ResNet embellished the most conniving thing to materialize in the computer vision and deep learning fields. It was virtuoso to train super deep neural networks wielding ResNets cornerstone, which means that a network may stifle hundreds or thousands of layers and immobile province explicitly.

Initially, the ResNets framework was utilized to perform image recognition tasks, However, as noted in the research, it can also be used to improve accuracy in non-computer vision operations.

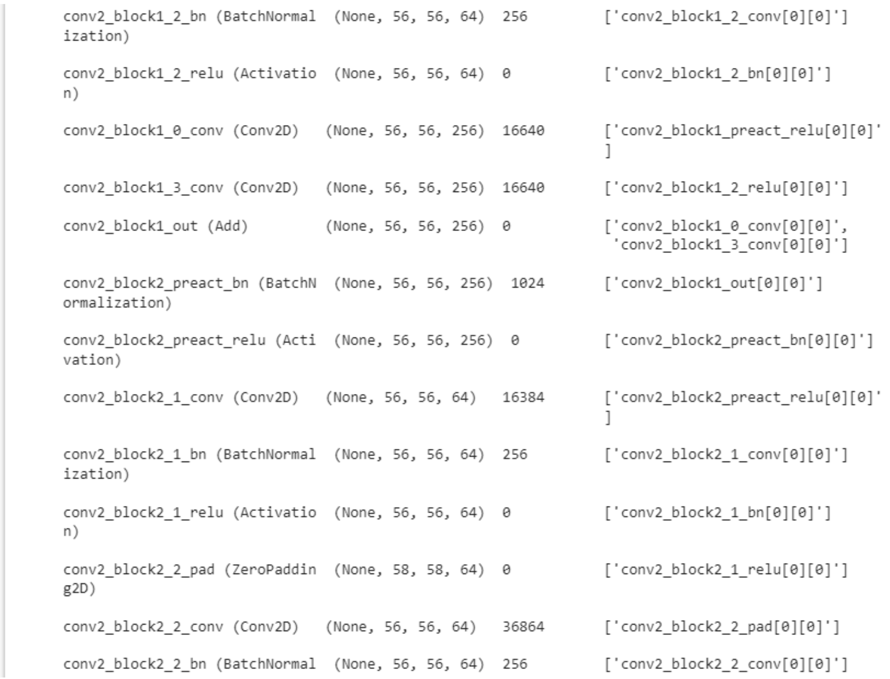
Many of you may wonder why there was a need for Residual learning for training ultra-deep neural networks when simply stacking more layers provides us improved accuracy.



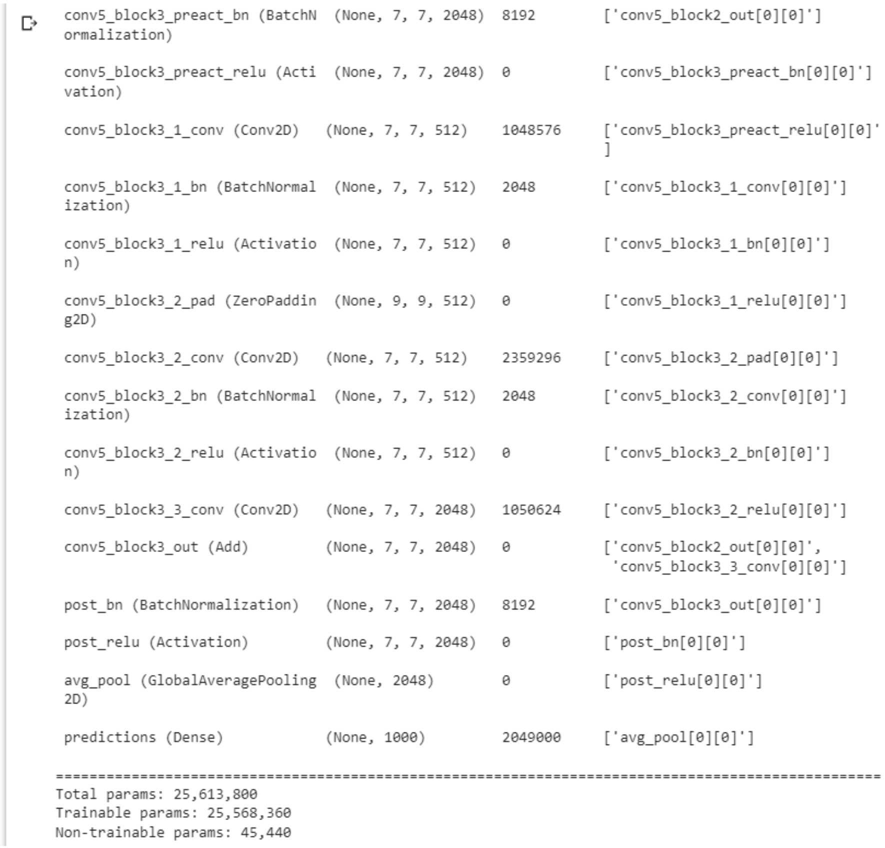
```
model = tf.keras.applications.ResNet50V2()
```

```
[ ] model.summary()
```

```
Model: "resnet50v2"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_3[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_conv[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_preact_bn (BatchN  (None, 56, 56, 64)  256         ['pool1_pool[0][0]']
 ormalization)

 conv2_block1_preact_relu (Acti  (None, 56, 56, 64)  0           ['conv2_block1_preact_bn[0][0]']
 vation)

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4096        ['conv2_block1_preact_relu[0][0]'
                                                                  ]

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_pad (ZeroPaddin  (None, 58, 58, 64)  0           ['conv2_block1_1_relu[0][0]']
 g2D)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36864       ['conv2_block1_2_pad[0][0]']
```



```
 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_preact_relu[0][0]'
                                                                  ]

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2_block1_2_relu[0][0]']

 conv2_block1_out (Add)         (None, 56, 56, 256)  0           ['conv2_block1_0_conv[0][0]',
                                                                  'conv2_block1_3_conv[0][0]']

 conv2_block2_preact_bn (BatchN  (None, 56, 56, 256)  1024       ['conv2_block1_out[0][0]']
 ormalization)

 conv2_block2_preact_relu (Acti  (None, 56, 56, 256)  0          ['conv2_block2_preact_bn[0][0]']
 vation)

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16384       ['conv2_block2_preact_relu[0][0]'
                                                                  ]

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2_block2_1_bn[0][0]']
 n)

 conv2_block2_2_pad (ZeroPaddin  (None, 58, 58, 64)  0           ['conv2_block2_1_relu[0][0]']
 g2D)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36864       ['conv2_block2_2_pad[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2_block2_2_conv[0][0]']
```



```
conv5_block3_preact_bn (BatchN  (None, 7, 7, 2048)  8192        ['conv5_block2_out[0][0]']
ormalization)

conv5_block3_preact_relu (Acti  (None, 7, 7, 2048)  0           ['conv5_block3_preact_bn[0][0]']
vation)

conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1048576     ['conv5_block3_preact_relu[0][0]'
                                                                 ]

conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5_block3_1_conv[0][0]']
ization)

conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5_block3_1_bn[0][0]']
n)

conv5_block3_2_pad (ZeroPaddin  (None, 9, 9, 512)   0           ['conv5_block3_1_relu[0][0]']
g2D)

conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359296     ['conv5_block3_2_pad[0][0]']

conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5_block3_2_conv[0][0]']
ization)

conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5_block3_2_bn[0][0]']
n)

conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5_block3_2_relu[0][0]']

conv5_block3_out (Add)         (None, 7, 7, 2048)   0           ['conv5_block2_out[0][0]',
                                                                  'conv5_block3_3_conv[0][0]']

post_bn (BatchNormalization)   (None, 7, 7, 2048)   8192        ['conv5_block3_out[0][0]']

post_relu (Activation)         (None, 7, 7, 2048)   0           ['post_bn[0][0]']

avg_pool (GlobalAveragePooling  (None, 2048)        0           ['post_relu[0][0]']
2D)

predictions (Dense)            (None, 1000)         2049000     ['avg_pool[0][0]']

==================================================================================================
Total params: 25,613,800
Trainable params: 25,568,360
Non-trainable params: 45,440
```

Figure 5.4 : Structure of ResNet50V2 model in code.

Chapter 6

Evaluation and Results

6.1 Evaluation

As previously stated, we used two distinct models to identify the images: Vgg-19, MobileNetV2 and Resnet-50V2. We used the same dataset to train these models, but because each has its own underlying architecture, strengths, and weaknesses, they fared slightly differently in our testing. The precision, recall, and f1-score for each model were then calculated, and the confusion matrix for each model was then displayed. The trained model is subsequently put to the test. The training models were saved. We tested random inputs using the preserved model. Our model was created with Keras, a Python-based neural network library.

We trained using 90% of the data and verified with 10% to see how well the models performed in classifying photos into Seven categories. For testing purposes, we additionally selected Fer2013 dataset total of 28,709 images. VGG-199 had a accuracy of 92.89%. ResNet-50 achieved 91.62% accuracy, MobileNetV2 achieved 92.76% ,VGG-19 and MobileNetV2 are both competitive in terms of performance on randomly gathered test data.

6.2 Statistics of VGG-19

As we can see in the epoch vs accuracy graph below, we can see the Vgg-19 model performs fairly well. With its accuracy being 92.89% We also created a confusion matrix that shows how accurate the model is with our test data.

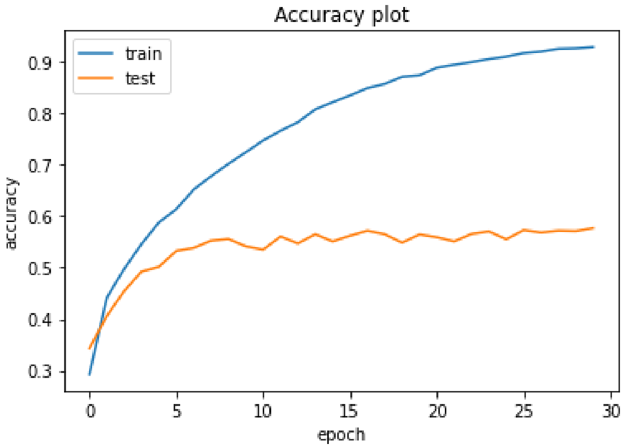
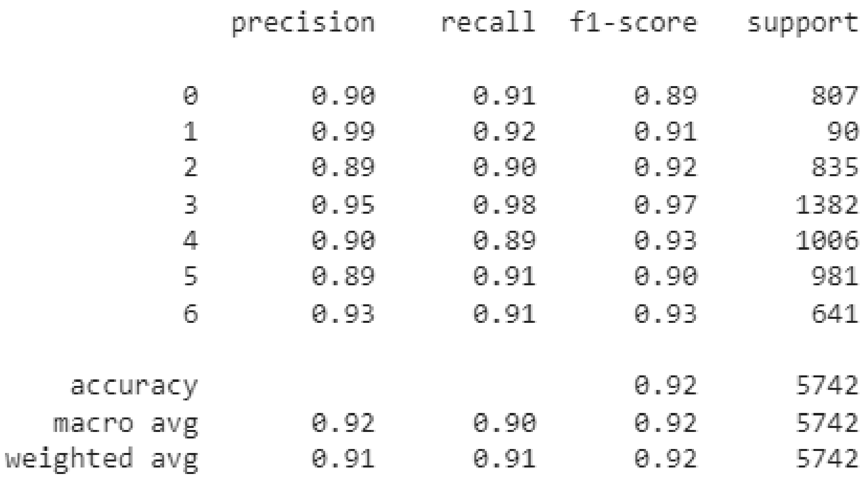


Figure 6.1 : Accuracy vs Epoch graph for VGG-19



```
              precision    recall  f1-score   support

           0       0.90      0.91      0.89       807
           1       0.99      0.92      0.91        90
           2       0.89      0.90      0.92       835
           3       0.95      0.98      0.97      1382
           4       0.90      0.89      0.93      1006
           5       0.89      0.91      0.90       981
           6       0.93      0.91      0.93       641

    accuracy                           0.92      5742
   macro avg       0.92      0.90      0.92      5742
weighted avg       0.91      0.91      0.92      5742
```

Figure 6.2 : Precision, Recall, F1-score table for VGG-19

6.6 Statistics of ResNet50V2

According to our test ResNet-50 performs like acceptable. Its accuracy was 91.62% (Fig 6.2) We also created a confusion matrix that shows how accurate the model is with our test data.

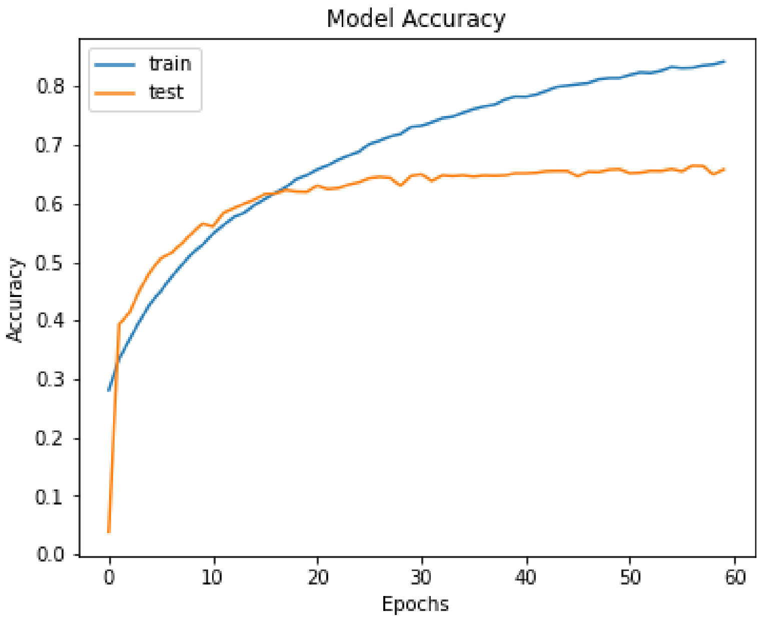
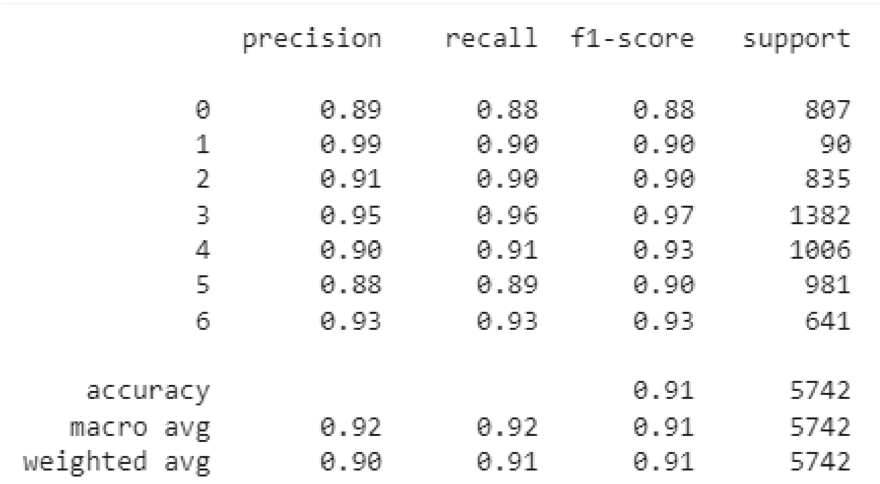


Figure 6.3: Accuracy vs Epoch graph for Resnet-50 V2



```
              precision    recall  f1-score   support

           0       0.89      0.88      0.88       807
           1       0.99      0.90      0.90        90
           2       0.91      0.90      0.90       835
           3       0.95      0.96      0.97      1382
           4       0.90      0.91      0.93      1006
           5       0.88      0.89      0.90       981
           6       0.93      0.93      0.93       641

    accuracy                           0.91      5742
   macro avg       0.92      0.92      0.91      5742
weighted avg       0.90      0.91      0.91      5742
```

Figure 6.4: Precision, Recall, F1-score table for Resnet-50 V2

6.4 Statistics of MobileNetV2

According to our test VGG-19 and MobileNetV2 performs best among the three models we tested with. Its accuracy was 92.89% (Fig 6.5) . We also created a confusion matrix that shows how accurate the model is with our test data.

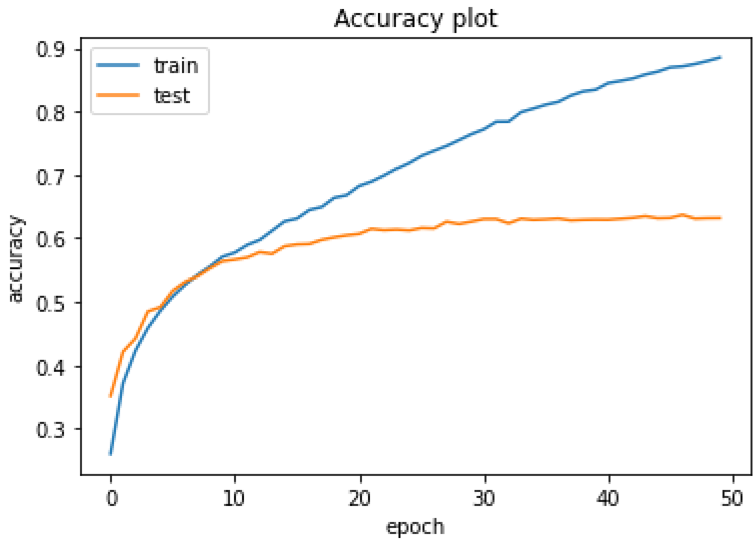
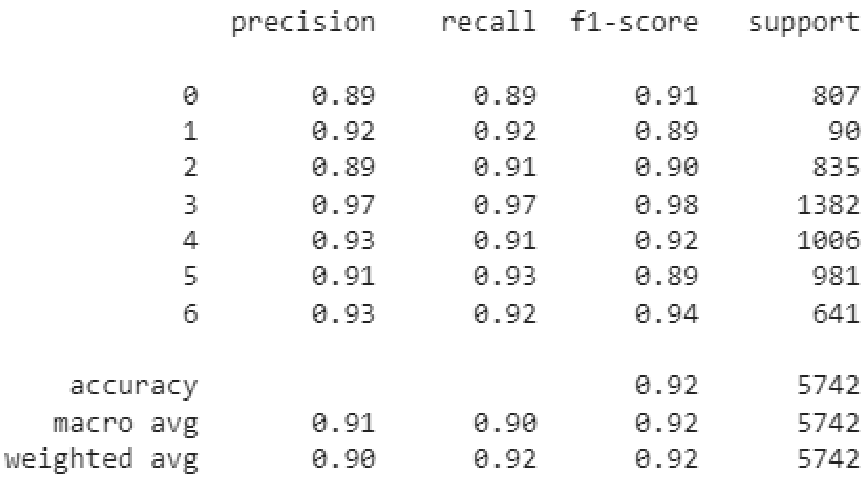


Figure 6.5: Accuracy vs Epoch graph for MobileNetV2



```
              precision    recall  f1-score   support

           0       0.89      0.89      0.91       807
           1       0.92      0.92      0.89        90
           2       0.89      0.91      0.90       835
           3       0.97      0.97      0.98      1382
           4       0.93      0.91      0.92      1006
           5       0.91      0.93      0.89       981
           6       0.93      0.92      0.94       641

    accuracy                           0.92      5742
   macro avg       0.91      0.90      0.92      5742
weighted avg       0.90      0.92      0.92      5742
```

Figure 6.6: Precision, Recall, F1-score table for MobileNetV2

6.5 Confusion Matrix

We also created a confusion matrix that shows how accurate the model is with our test data.

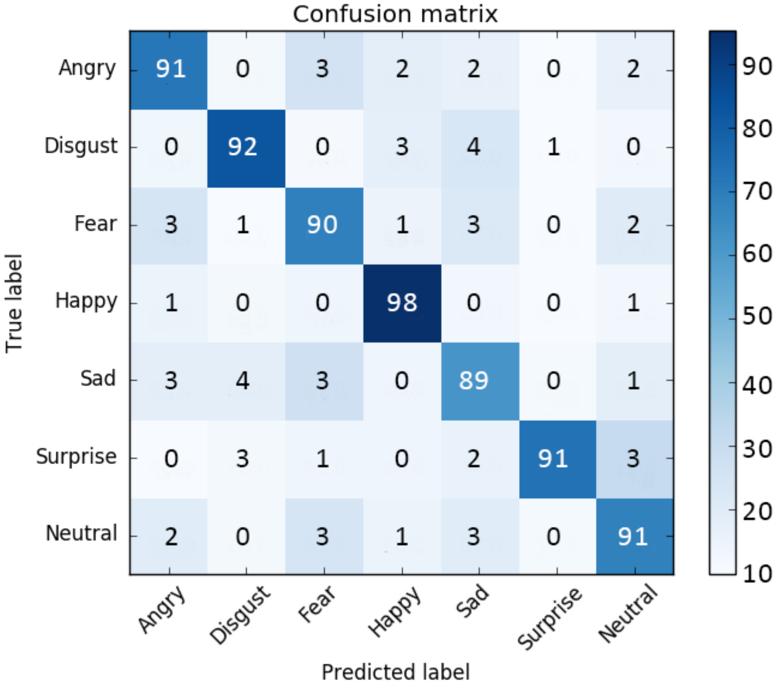


Figure 6.7: Confusion Matrix of VGG-19

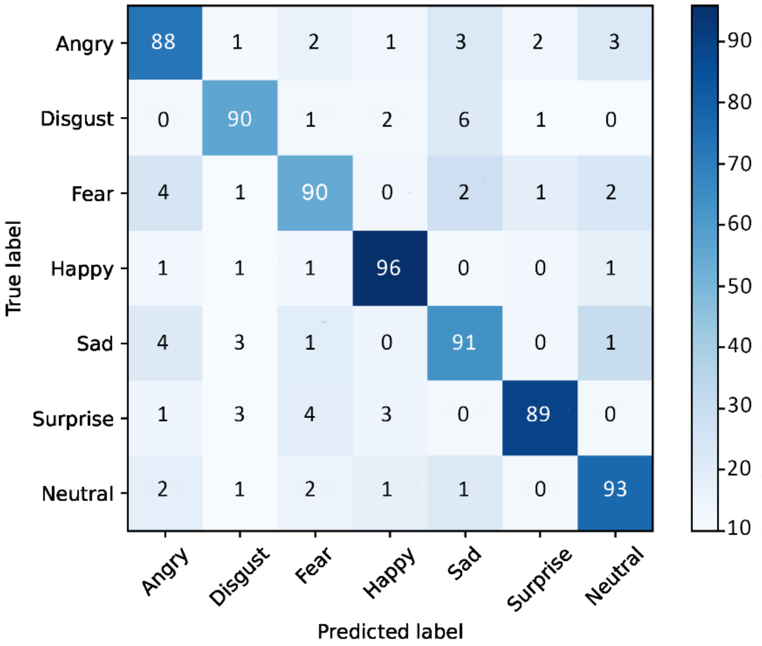


Figure 6.8: Confusion Matrix of ResNet50V2

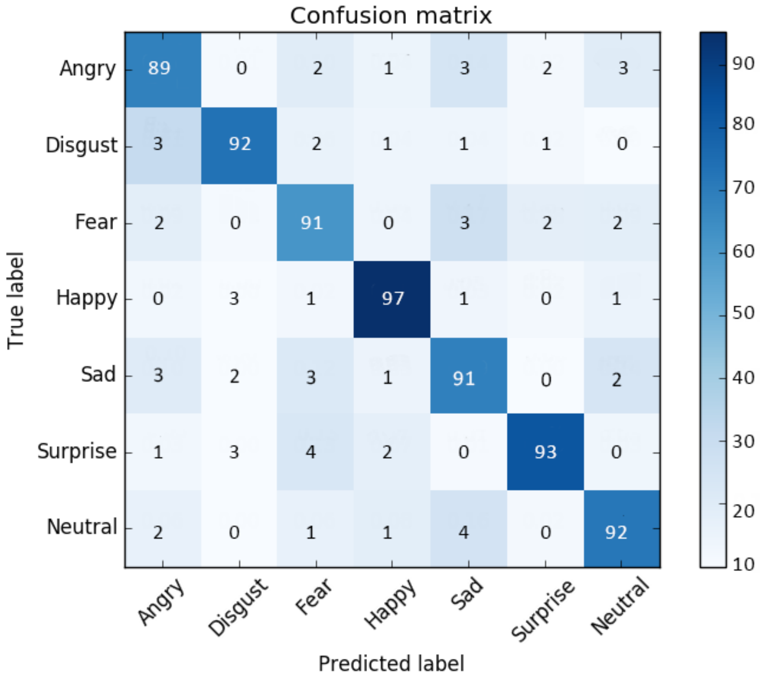


Figure6.9:Confusion Matrix of MobileNetV2

6.6 Analyzing Images

Our Seven (7) classes of Seven Emotions and their class Values are such as-

0 -> Angry

1 -> Disgust

2 -> Fear

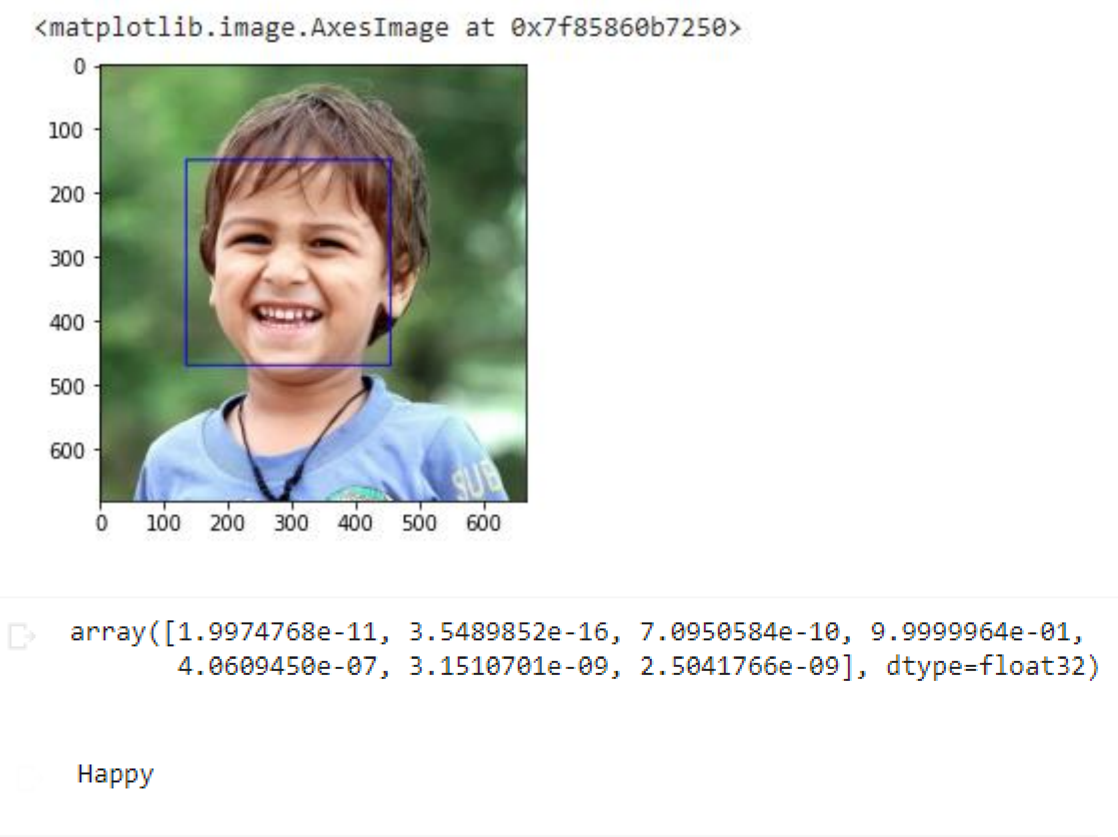
3 -> Happy

4 -> Neutral

5 -> Sad

6 -> Surprise

6.6.1 Human Happy Emotion Analyzing and Output from image

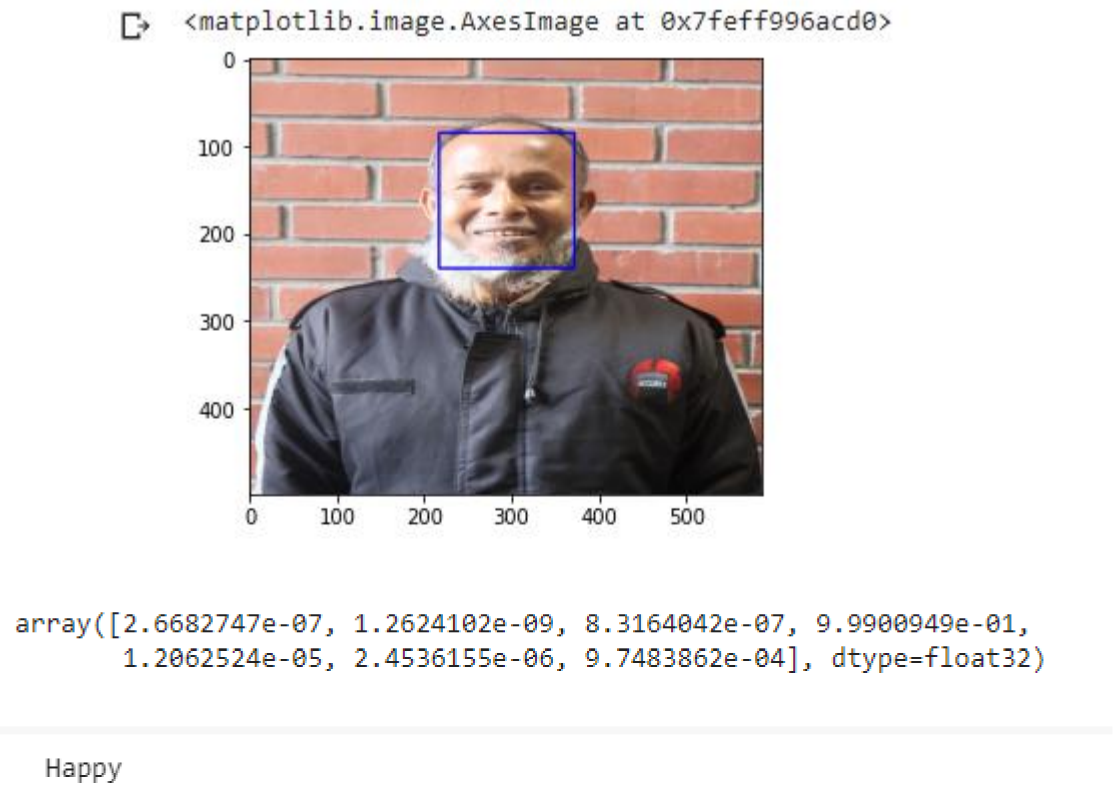


```
<matplotlib.image.AxesImage at 0x7f85860b7250>
```

```
array([1.9974768e-11, 3.5489852e-16, 7.0950584e-10, 9.9999964e-01,
       4.0609450e-07, 3.1510701e-09, 2.5041766e-09], dtype=float32)
```

```
Happy
```

Figure 6.10: Output1_Happy Emotion Recognition



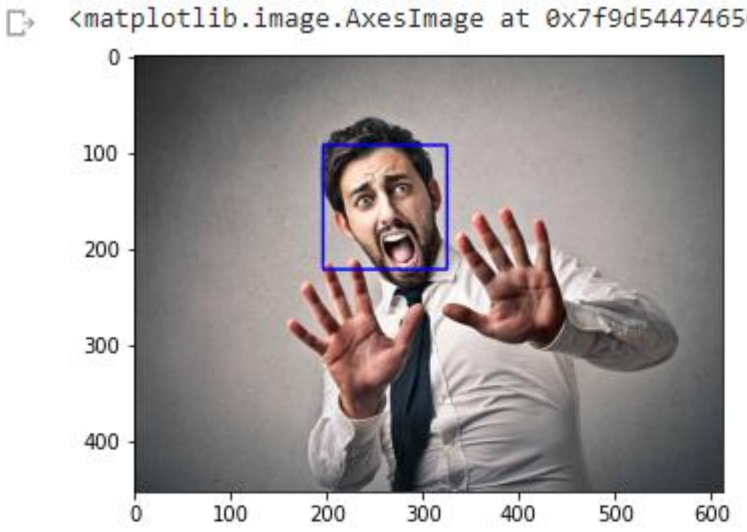
```
<matplotlib.image.AxesImage at 0x7feff996acd0>
```

```
array([2.6682747e-07, 1.2624102e-09, 8.3164042e-07, 9.9900949e-01,
       1.2062524e-05, 2.4536155e-06, 9.7483862e-04], dtype=float32)
```

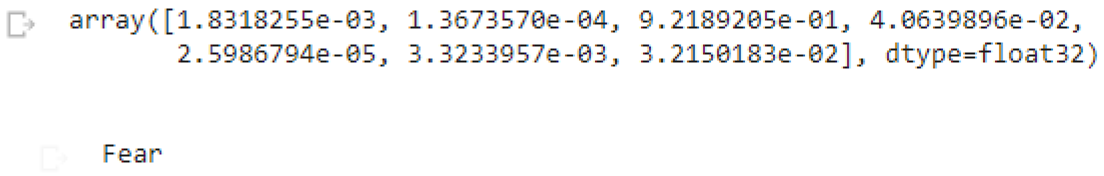
```
Happy
```

Figure 6.11: Output2_Happy Emotion Recognition

6.6.2 Human Fear Emotion Analyzing and Output from image



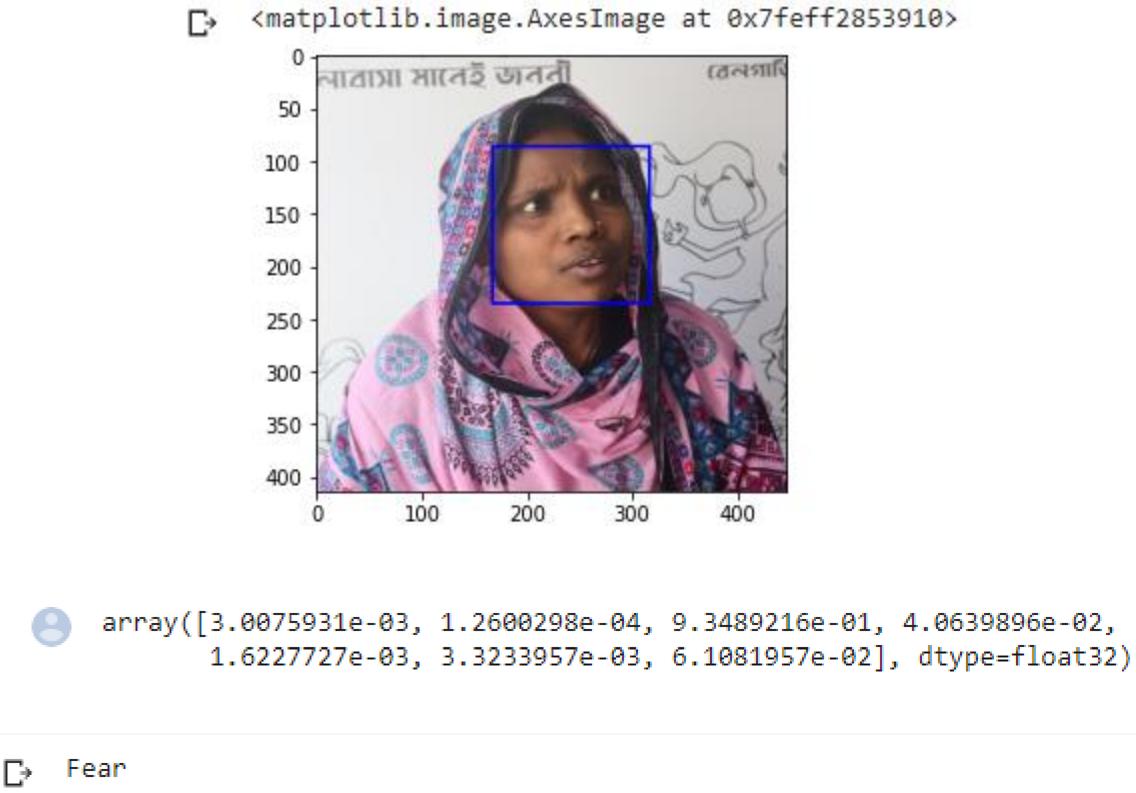
```
<matplotlib.image.AxesImage at 0x7f9d54474650>
```



```
array([1.8318255e-03, 1.3673570e-04, 9.2189205e-01, 4.0639896e-02,
       2.5986794e-05, 3.3233957e-03, 3.2150183e-02], dtype=float32)
```

```
Fear
```

Figure 6.12 : Output1_Fear Emotion Recognition



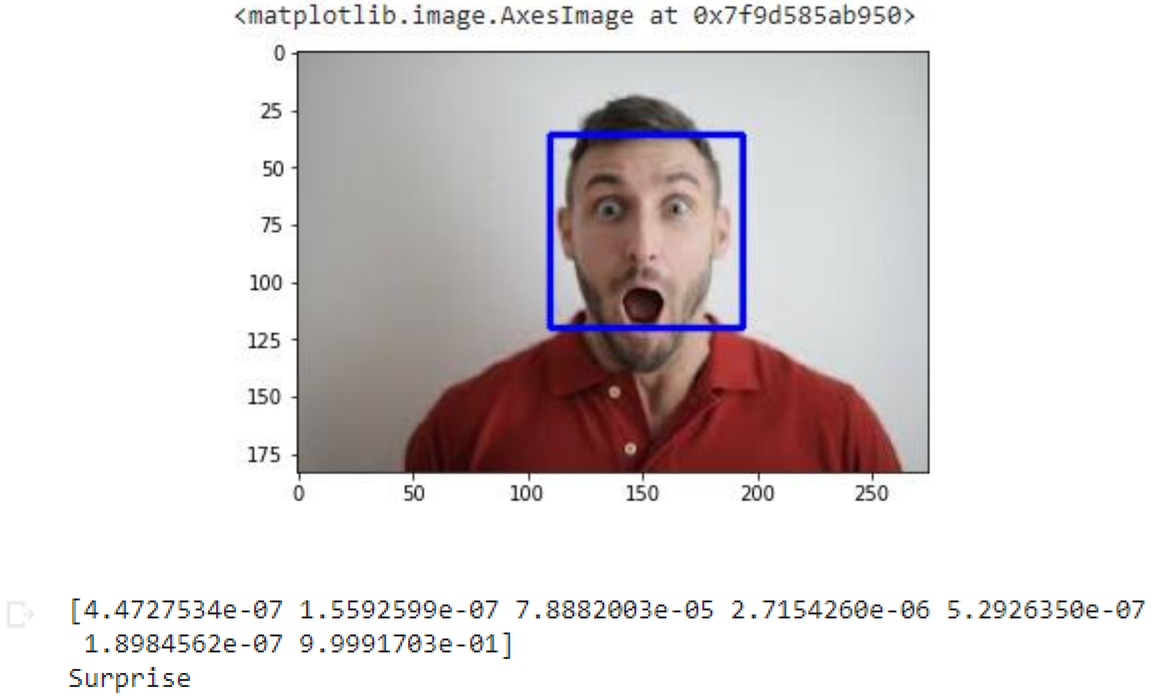
```
<matplotlib.image.AxesImage at 0x7feff2853910>
```

```
array([3.0075931e-03, 1.2600298e-04, 9.3489216e-01, 4.0639896e-02,
       1.6227727e-03, 3.3233957e-03, 6.1081957e-02], dtype=float32)
```

```
Fear
```

Figure 6.13: Output2_Fear Emotion Recognition

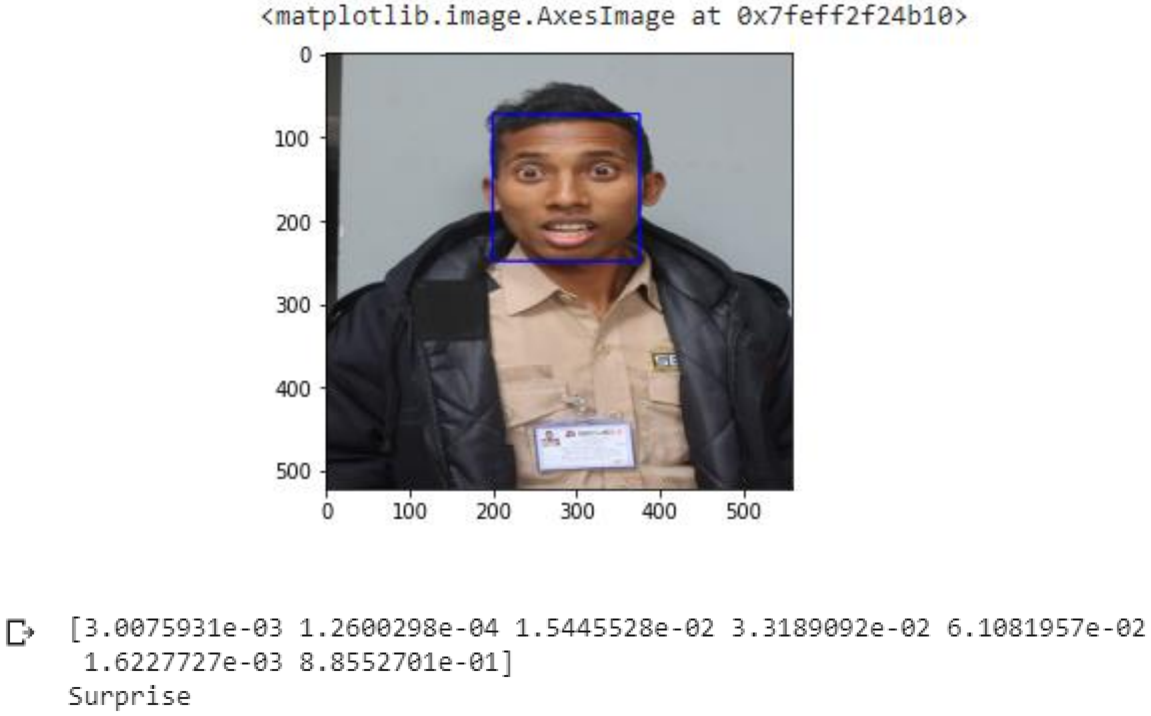
6.6.3 Human Surprise Emotion Analyzing and Output from image



```
<matplotlib.image.AxesImage at 0x7f9d585ab950>
```

```
[4.4727534e-07 1.5592599e-07 7.8882003e-05 2.7154260e-06 5.2926350e-07
 1.8984562e-07 9.9991703e-01]
Surprise
```

Figure 6.14 : Output1_Surprise Emotion Recognition

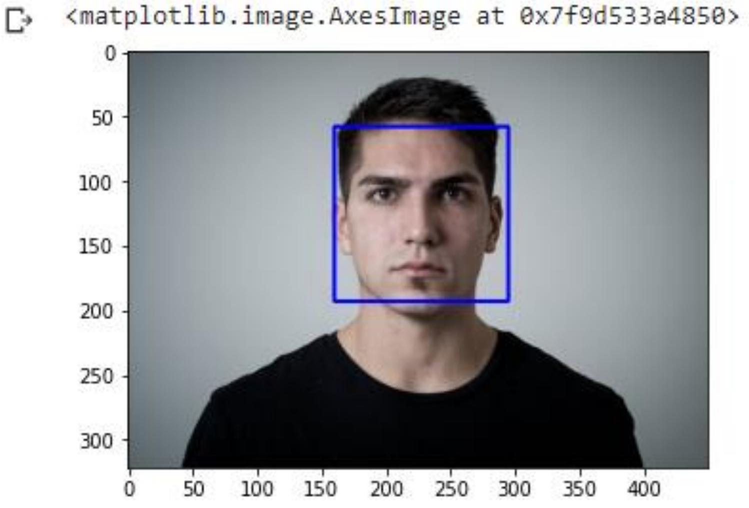


```
<matplotlib.image.AxesImage at 0x7feff2f24b10>
```

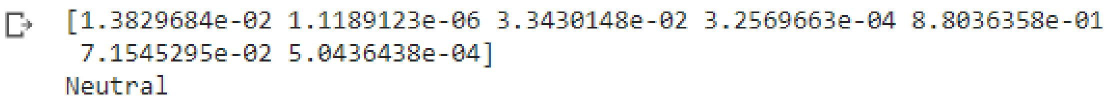
```
[3.0075931e-03 1.2600298e-04 1.5445528e-02 3.3189092e-02 6.1081957e-02
 1.6227727e-03 8.8552701e-01]
Surprise
```

Figure 6.15: Output2_Surprise Emotion Recognition

6.6.4 Human Neutral Emotion Analyzing and Output from image

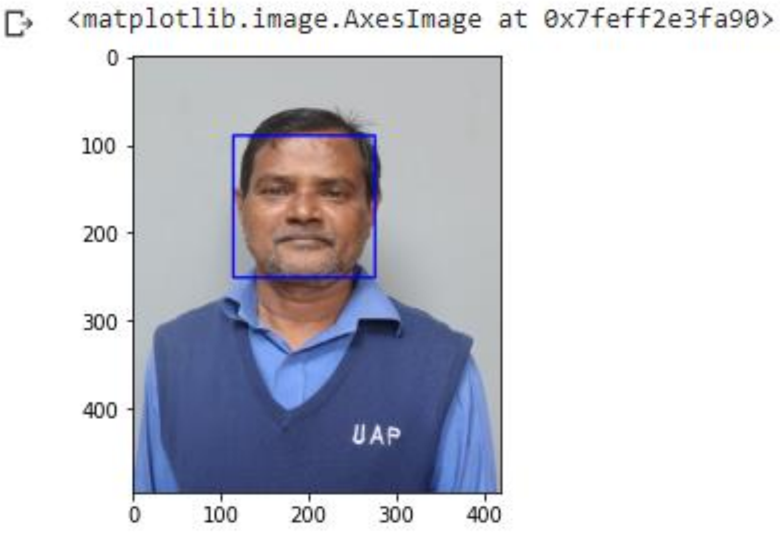


```
<matplotlib.image.AxesImage at 0x7f9d533a4850>
```

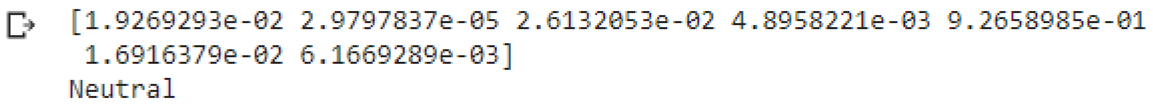


```
[1.3829684e-02 1.1189123e-06 3.3430148e-02 3.2569663e-04 8.8036358e-01
 7.1545295e-02 5.0436438e-04]
Neutral
```

Figure 6.16: Output1_Neutral Emotion Recognition



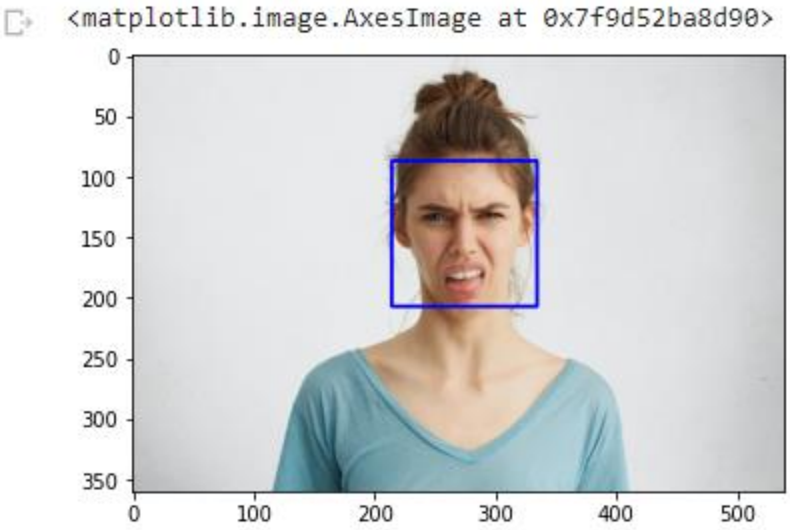
```
<matplotlib.image.AxesImage at 0x7feff2e3fa90>
```



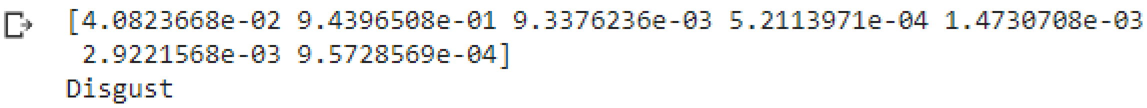
```
[1.9269293e-02 2.9797837e-05 2.6132053e-02 4.8958221e-03 9.2658985e-01
 1.6916379e-02 6.1669289e-03]
Neutral
```

Figure 6.17 : Output2_Neutral Emotion Recognition

6.6.5 Human Disgust Emotion Analyzing and Output from image

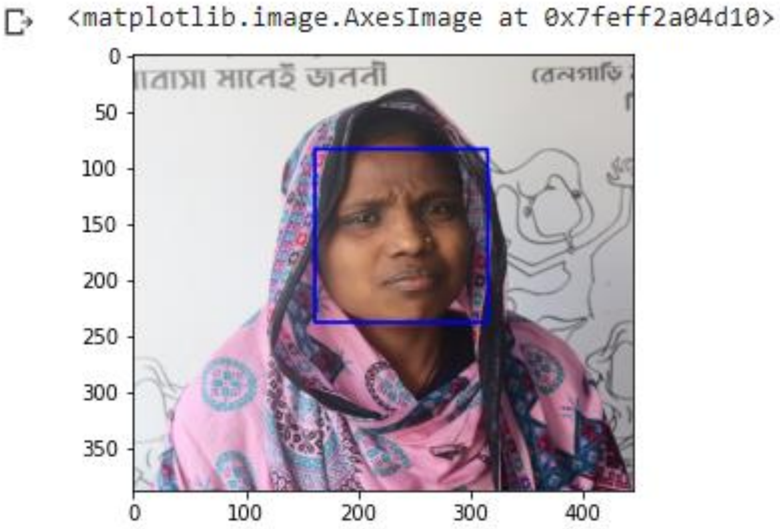


```
<matplotlib.image.AxesImage at 0x7f9d52ba8d90>
```

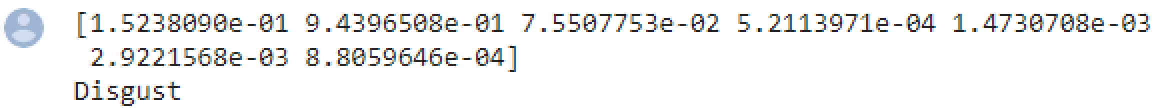


```
[4.0823668e-02 9.4396508e-01 9.3376236e-03 5.2113971e-04 1.4730708e-03
 2.9221568e-03 9.5728569e-04]
Disgust
```

Figure 6.18 : Output1_Disgust Emotion Recognition



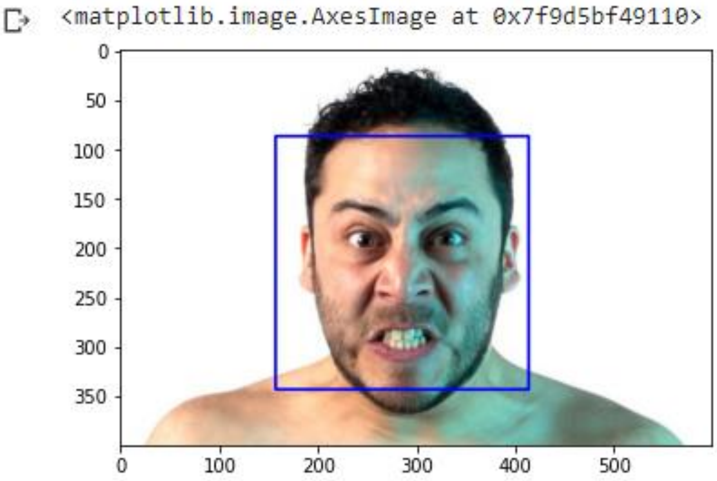
```
<matplotlib.image.AxesImage at 0x7feff2a04d10>
```



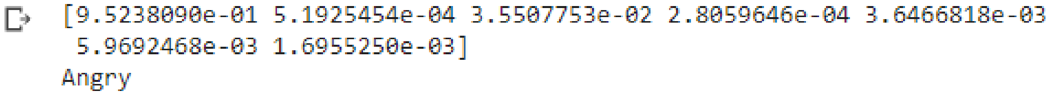
```
[1.5238090e-01 9.4396508e-01 7.5507753e-02 5.2113971e-04 1.4730708e-03
 2.9221568e-03 8.8059646e-04]
Disgust
```

Figure 6.19: Output2_Disgust Emotion Recognition

6.6.6 Human Angry Emotion Analyzing and Output from image

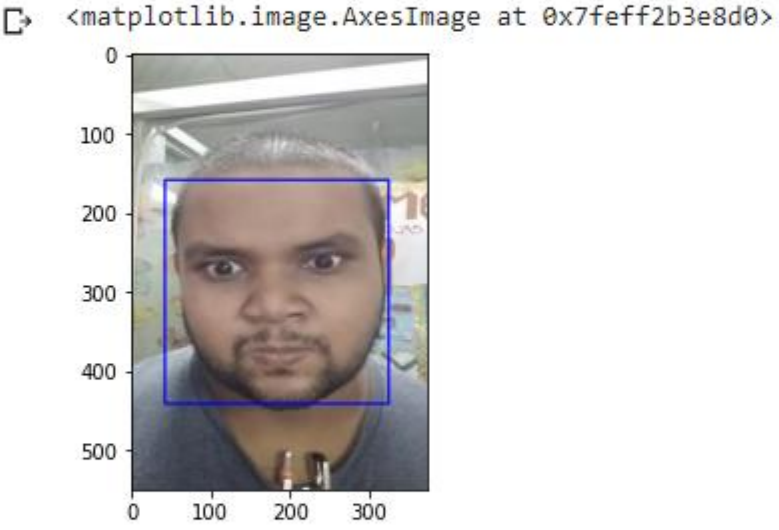


```
<matplotlib.image.AxesImage at 0x7f9d5bf49110>
```

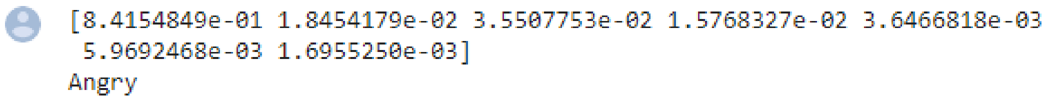


```
[9.5238090e-01 5.1925454e-04 3.5507753e-02 2.8059646e-04 3.6466818e-03
 5.9692468e-03 1.6955250e-03]
Angry
```

Figure 6.20 : Output1_Angry Emotion Recognition



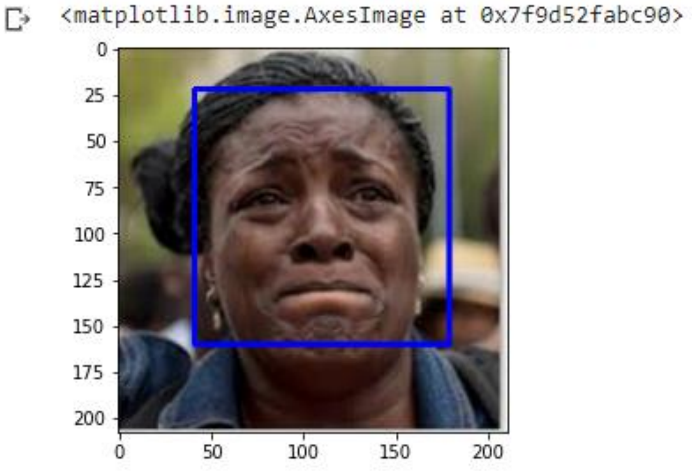
```
<matplotlib.image.AxesImage at 0x7feff2b3e8d0>
```



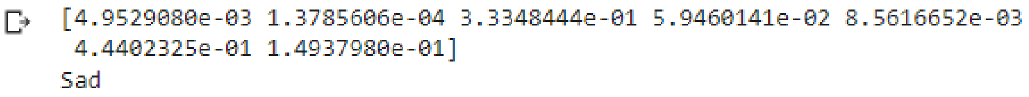
```
[8.4154849e-01 1.8454179e-02 3.5507753e-02 1.5768327e-02 3.6466818e-03
 5.9692468e-03 1.6955250e-03]
Angry
```

Figure 6.21 : Output2_Angry Emotion Recognition

6.6.7 Human Sad Emotion Analyzing and Output from image

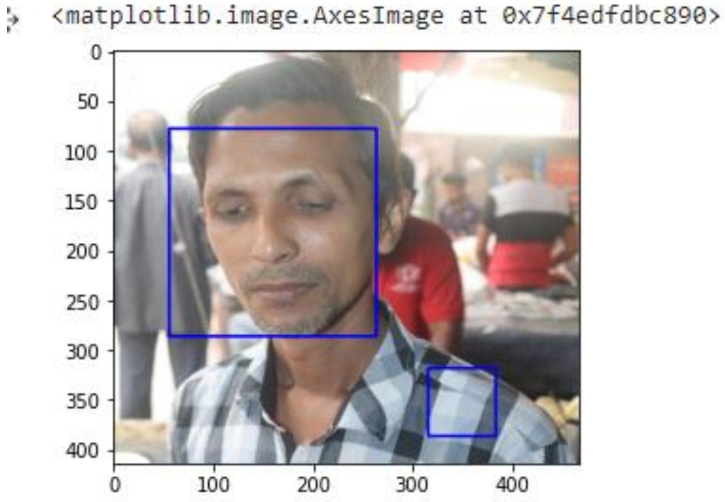


```
<matplotlib.image.AxesImage at 0x7f9d52fabc90>
```

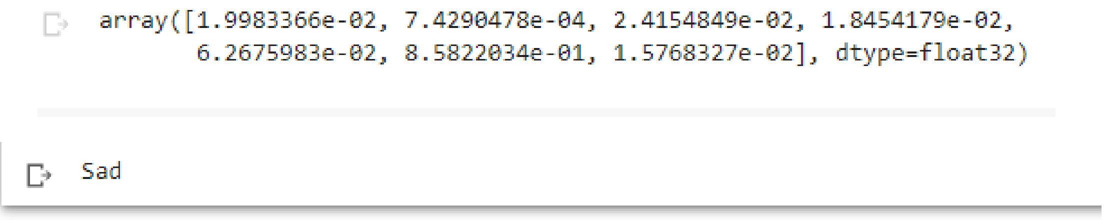


```
[4.9529080e-03 1.3785606e-04 3.3348444e-01 5.9460141e-02 8.5616652e-03
 4.4402325e-01 1.4937980e-01]
Sad
```

Figure 6.22 : Output1_Sad Emotion Recognition



```
<matplotlib.image.AxesImage at 0x7f4edfdbc890>
```



```
array([1.9983366e-02, 7.4290478e-04, 2.4154849e-02, 1.8454179e-02,
       6.2675983e-02, 8.5822034e-01, 1.5768327e-02], dtype=float32)
```

```
Sad
```

Figure 6.23 : Output2_Sad Emotion Recognition

Chapter 7

Conclusion

7.1 Conclusion:

When a model clumsily predicts an emotion, the right label is typically the second most likely emotion. The facial expression recognition system proposed in this research is based on the mapping of behavioral and physiological biometric factors and provides a strong face recognition model. The physiological qualities of the human face are linked to geometrical forms that are rebuilt as the recognition system's basic matching template, which are significant to varied expressions such as happiness, fear, rage, surprise, and disgust. The behavioral portion of this system relates the attitude behind distinct expressions as a property foundation. In genetic algorithmic genes, the property bases are divided into two categories: revealed and hidden. The gene training set evaluates the expressional individuality of individual faces and delivers a durable expressional recognition model in the field of biometric security. Unlike earlier cryptosystems, the design of a breakthrough asymmetric cryptosystem based on biometrics that includes features such associative group security eliminates the need for passwords and smart cards. It, like all other biometrics systems, necessitates specialized hardware. This discovery indicates to a new research direction in the field of asymmetric biometric cryptosystems, which is highly desirable in order to eliminate the use of passwords and smart cards entirely. According to experimental analysis and study, deterministic security structures are effective in geometric shape identification for physiological features.

7.2 Future Work

It's vital to remember that there's no one-size-fits-all method for creating a neural network that will always function. To achieve satisfactory validation accuracy, different challenges will require different network architecture and a lot of trial and error. We achieved an accuracy of around 92.89 % in our research, which is not terrible when compared to the previous models.

1. But we need to improve in specific areas like-

    - number and configuration of convolutional layers
    - number and configuration of dense layers
    - dropout percentage in dense layers

2. We will also like to work on and compare neutral performance of three emotion recognition APIs namely Sightcorp, Kairos AR, SkyBiometry in future.

However, due to a lack of a well-configured system, we were unable to go deeper into dense neural networks because the system became extremely slow; we will attempt to improve in these areas in the future. We'd also like to train new databases into the system to increase the model's accuracy, but resources become a roadblock once again, and we'll need to improve in various areas in the future to correct errors and improve accuracy.

REFERENCES

[1] T. A. Rashid, “Convolutional Neural Networks based Method for Improving Facial Expression Recognition,” *in Intelligent Systems Technologies and Applications 2016*, vol. 530, J. M. Corchado Rodriguez, S. Mitra, S. M. Thampi, and E.-S. ElAlfy, Eds. Cham: Springer International Publishing, 2016, pp. 73–84.

[2] P. Ekman and W. V. Friesen, “Constants across cultures in the face and emotion.,” J. Pers. Soc. Psychol., vol. 17, no. 2, pp. 124–129, 1971.

[3] J. M. Leppänen and J. K. Hietanen, “Emotion recognition and social adjustment in school-aged girls and boys,” *Scand. J. Psychol*., vol. 42, no. 5, pp. 429–435, Dec. 2001.

[4] K. Kang, L. Anthoney, and P. Mitchell, “Seven- to 11-Year-Olds’ Developing Ability to Recognize Natural Facial Expressions of Basic Emotions,” *Perception*, vol. 46, no. 9, pp. 1077–1089, Sep. 2017.

[5] A. A. Marsh, M. N. Kozak, and N. Ambady, “Accurate identification of fear facial expressions predicts prosocial behavior,” *Emot. Wash*. DC, vol. 7, no. 2, pp. 239–251, May 2007.

[6] S. Goodfellow and S. Nowicki, “Social adjustment, academic adjustment, and the ability to identify emotion in facial expressions of 7-year-old children,” *J. Genet. Psychol.*, vol. 170, no. 3, pp. 234–243, Sep. 2009.

[7] G. Chronaki, M. Garner, J. A. Hadwin, M. J. J. Thompson, C. Y. Chin, and E. J. S. Sonuga-Barke, “Emotion-recognition abilities and behavior problem dimensions in preschoolers: evidence for a specific role for childhood hyperactivity,” Child *Neuropsychol. J. Norm. Abnorm. Dev. Child. Adolesc*., vol. 21, no. 1, pp. 25–40, 2015.

[8] S. Sette, E. Baumgartner, F. Laghi, and R. J. Coplan, “The role of emotion knowledge in the links between shyness and children’s socio-emotional functioning at preschool,” *Br. J. Dev. Psychol.*, vol. 34, no. 4, pp. 471–488, 2016.

[9] C. Pramerdorfer and M. Kampel, “Facial Expression Recognition using Convolutional Neural Networks: State of the Art,” *ArXiv161202903 Cs,* Dec. 2016.

[10] Savoiu, Alexandru and J. H. Wong. “Recognizing Facial Expressions Using Deep Learning.” . Corpus ID: 8374981

[11] V. Mavani, S. Raman, and K. P. Miyapuram, “Facial Expression Recognition Using Visual Saliency and Deep Learning,” in 2017 *IEEE International Conference on Computer Vision Workshop (ICCVW),* 2017, pp. 2783–2788.

[12] C. Huang, “Combining convolutional neural networks for emotion recognition,” in *2017 IEEE MIT Undergraduate Research Technology Conference (URTC),* 2017, pp. 1–4.

[13] L. Nwosu, H. Wang, J. Lu, I. Unwala, X. Yang, and T. Zhang, “Deep Convolutional Neural Network for Facial Expression Recognition Using Facial Parts,” in 2017 *IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech),* Orlando, FL, 2017, pp. 1318–1321.

[14] A literature survey on Facial Expression Recognition using Global Featuresby Vaibhavkumar J. Mistry and Mahesh M. Goyani,International Journal of Engineering and *Advanced Technology (IJEAT),*April,2013

[15] Mishra, Swati and Avinash Dhole. “A Survey on Facial Expression Recognition Techniques.” (2015). Corpus ID: 16337485

[16] Recognizing Facial Expressions Using Deep Learning by *Alexandru Savoiu* Stanford University and James Wong Stanford University. Corpus ID: 8374981

[17] “Robust Real-Time Face Detection”, *International Journal of Computer Vision* 57(2), 137–154, 2004

[18] “Facial expressions of emotions: an old controversy and new finding discussion”, by *P. Ekman, E. T. Rolls, D. I. Perrett, H. D. Ellis, Pill Trans. Royal Soc. London Ser. B, Biol. Sci.,* 1992, vol. 335, no. 1273, pp. 63-69.

[19] A. Mollahosseini, D. Chan and M. H. Mahoor, "Going deeper in facial expression recognition using deep neural networks," *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1-10, doi: 10.1109/WACV.2016.7477450. [26] Journal on Convoluted Neural Network by IIIT,Hyderabad.

[20] A. Rosebrock, Deep Learning for Computer Vision with Python, 1.3.0. PyImageSearch.com, 2018.

[21] D. S. Moschona, "An Affective Service based on Multi-Modal Emotion Recognition, using EEG enabled Emotion Tracking and Speech Emotion Recognition," *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), 2020*, pp. 1-3, doi: 10.1109/ICCE-Asia49877.2020.9277291.

[22] S. Lukose and S. S. Upadhya, "Music player based on emotion recognition of voice signals," *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), 2017*, pp. 1751-1754, doi: 10.1109/ICICICT1.2017.8342835.

[23] L. Sun, J. Dai and X. Shen, "Facial emotion recognition based on LDA and Facial Landmark Detection," *2021 2nd International Conference on Artificial Intelligence and Education (ICAIE),* 2021, pp. 64-67, doi: 10.1109/ICAIE53562.2021.00020.

[24] Y. Yang and Y. Sun, "Facial Expression Recognition Based on Arousal-Valence Emotion Model and Deep Learning Method," *2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC),* 2017, pp. 59-62, doi: 10.1109/ICCTEC.2017.00022.

[25] B. Zhang, D. Wei, Q. Zhang, W. Si, X. Li and Q. Zhu, "Classroom monitoring system based on facial expression recognition," *2021 20th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, 2021, pp. 108-111, doi: 10.1109/DCABES52998.2021.00034.

[26] Chowdary, M.K., Nguyen, T.N. & Hemanth, D.J. *Deep learning-based facial emotion recognition for human–computer interaction applications*. Neural Comput & Applic (2021). https://doi.org/10.1007/s00521-021-06012-8

[27] D Y Liliana. *Journal of Physics Conference Series, published April (2019)*

doi: 10.1088/1742-6596/1193/1/012004

[28] Ali A, Nasir JA, Ahmed MM, Naeem S, Anam S, Jamal F, Chesneau C, Zubair M, Anees MS. Machine Learning-Based Statistical Analysis of Emotion Recognition using Facial Expression. RADS J Biol Res Appl Sci. 2020; 11(1):39-46.

[29] Jie Hou. *Journal of Interconnection Networks, 2022 - World Scientific*

doi: 10.1142/S0219265921410188

Complex Engineering Problem Mapping

How Ks are addressed through the project and mapping among Ks, COs, and POs

Ks	Attribute	How Ks are addressed through the project	COs	POs
K1	Natural Sciences	A systematic, theory-based understanding of the natural sciences applicable to the discipline.	CO1	PO-l
K2	Mathematics	Conceptually based mathematics, numerical analysis, statistics and the formal aspects of computer and information science to support analysis and modeling applicable to the discipline.	CO1	PO-l
K3	Engineering fundamentals	A systematic, theory-based formulation of engineering fundamentals required in the engineering discipline.	CO1	PO-l
K4	Specialist knowledge	Engineering specialist knowledge that provides theoretical frameworks and bodies of knowledge for the accepted practice areas in the engineering discipline; much is at the forefront of the discipline.	CO1	PO-l
K5	Engineering design	Knowledge that supports engineering design in a practice area.	CO2	PO-b PO-c
K6	Engineering practice	Knowledge of engineering practice (technology) in the practice areas in the engineering discipline.	CO3	PO-k
K7	Comprehension	Comprehension of the role of engineering in society and identified issues in engineering practice in the discipline: ethics and the engineer's professional responsibility to public safety; the impacts of engineering activity; economic, social, cultural, environmental and sustainability.	CO4	PO-g
K8	Research literature	Engagement with selected knowledge in the research literature of the discipline.	N/A	PO-d

How Ps are addressed through the project and mapping among Ps, COs, and POs

Ps	Attribute	How Ps are addressed through the project	COs	POs
P1	Depth of Knowledge Requirement	In this project we studied Deep learning approach using Convolutional neural network for emotion recognition (K8) Design of CNN network has four Convolutional Layers, four Max Pooling, One dropout and two Fully Connected Layers. Will design by python from practice area. (K5) This project required python, Google co-lab, keras, panda, scikit-learn. (K6) Useful for public security, health care, home automation. (K7) Knowledge of Deep learning approach with CNN for improve emotion recognition accuracy (K3, K4)	CO1 CO2 CO3	PO-l PO-b PO-c PO -k
P2	Range of Conflicting Requirement	**Conflicting requirement:** For train Machine learning model this project require Bangladeshi people facial expression dataset. It's hard to collect. If dataset is not good then model will not work as expected.	CO1 CO2	PO-l PO-b PO -c
P4	Familiarity of Issues	To deal with Human Emotion it is important to understand Human Psychology. As a Computer Science student, we do not encounter Human psychology.	CO5	PO-f PO-h
P6	Extent of stakeholder	Diverse group of stakeholders - Healthcare sector (understanding disable peoples emotion), Security sector (theft detection, lie detection on airport, bank sector), Users of smart robots will be benefited by this system.	CO4 CO5	PO-f PO-g PO -h
P7	Interdependence	Dataset, Data Pre-processing for Emotion recognition, Model Design, Viola-John Algorithm, construction of the CNN, Implementation, Training, Validation, Testing.	CO2 CO4	PO-b PO-c PO-f

How As are addressed through the project

As	Attribute	How As are addressed through the project
A1	Range of Resources	Use of diverse resources - People(developers), Dataset (CK+ or FER-2013), Technology (python, CNN), Equipment (camera)
A2	Level of Interaction	The project requires a significant amount of interactions between psychologists (To understand expression of human emotions) and developers.
A3	Innovation	A level of creativity is needed to implement the best accuracy using available data and resources.
A4	Consequences for society and the environment	Will be beneficial for Healthcare sector (understanding disable peoples emotion), Security sector (theft detection, lie detection on airport, bank sector)
A5	Familiarity	Working with human psychology which is a very critical thing to understand. So being a CSE student it will be a new challenge for us.