Emotion Detection Using Facial Expression

A PROJECT REPORT

Submitted by

AMRITHA.T.N

ATCHAYA.R

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

ANNA UNIVERSITY: CHENNAI 600 025

MARCH 2020

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"EMOTION DETECTION USING FACIAL EXPRESSION"** is the bonafide work of **"AMRITHA.T.N, ATCHAYA.R"** who carried out the project work under my supervision.

SIGNATURE

Dr. R. Manimegalai

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering

PSG Institute of Technology and Applied Research

Avinashi Road, Neelambur

Coimbatore - 641 062

SIGNATURE

Mr. C.P. Shabariram

SUPERVISOR

Assistant Professor

Department of Computer Science and Engineering

PSG Institute of Technology and Applied Research

Avinashi Road, Neelambur

Coimbatore – 641 062

ACKNOWLEDGEMENT

I am very grateful to **Dr.P.V.Mohanram, Principal** for providing me with an environment to complete our project successfully.

I also take the privilege of expressing my gratitude to **Dr. G. Chandramohan, Vice Principal** in extending his support to carry out our study.

I am greatly indebted to **Dr. R. Manimegalai, Head of the Department, Computer Science and Engineering** for her guidance which was instrumental in the completion of my project.

I am very grateful to **Mr. C.P. Shabariram, Project Guide** for her constant encouragement and support throughout my course.

I express my sincere thanks to **Mrs. Bhuvana, Project Coordinator** for her constant encouragement and support throughout my course.

Also, I extend my gratitude to **Mr. C.P. Shabariram, Project Coordinator** for his constant encouragement and support throughout my course.

Finally, I take this opportunity to extend my deepest appreciation to my **family and friends,** who supported me during the crucial times of our project.

AMRITHA T N

ATCHAYA R

ABSTRACT

In recent years, facial recognition plays a major role in the fields of artificial intelligence, robotics, security, trade etc. It can be used in stress analysis in industries by capturing the faces of the employees and by analysing their emotions in order to put forward the suitable therapies for cure in required. Face recognition can also be used in shopping malls to analyse the facial expressions of the users while purchasing various products, thus capturing their interests for purchasing the same. This can then be used together with the other data for the process of data analysis and further promotions for the users in terms of their liking products.

There are multiple methods like support vector machine, k-nearest neighbour algorithm, multilayer perceptron that are used in facial expression analysis. Though there are various methods to identify expressions using Machine learning and Artificial Intelligence techniques, this project attempts to use Deep Convolutional neural networks to recognize various expressions from the faces and classify them accordingly.

This project proposes a modified deep learning neural network to learn face representation from a smaller data set. Convolutional neural networks are well known to have a higher accuracy when compared to traditional machine learning methods. Our system continuously captures the images of the user using Webcam and analyses the facial expressions accordingly using a convolutional neural network and identify the mood of the user which can be used for further analyses.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	LIST OF FIGURES	**xviii**
	LIST OF SYMBOLS	**xxvi**
1	**INTRODUCTION**	**1**
	1.1 Convolution	2
	1.2 2D Convolution	2
	1.3 Pooling	4
	1.4 Fully Connected Layer	5
	1.5 Activation Function	6
	1.6 Optimizers	6
	1.7 Dropout	9
	1.8 Batch Normalization	10
	1.9 Cascade Classifier	10
	1.10 Haar features	11
	1.11 Haar Cascade Classifier	12
2	**LITERATURE REVIEW**	**14**
	2.1 Judgement Based Approaches	14
	2.2 Sign Based Approaches	15
	2.3 Reliability of Ground Truth Coding	15

	2.4 Automatic Face Expression Analysis	16
	2.5 Face Acquisition	16
	2.6 Face Feature Extraction	16
	2.7 Deformation Extraction System	17
	2.8 Motion Extraction System	17
	2.9 Hybrid System	17
	2.10 Speech Based Recognition System	18
3	**NEURAL NETWORK DESIGN**	**19**
	3.1 TensorFlow and Keras	19
	3.2 Architecture Design of CNN	20
	3.3 Dataset	24
	3.4 Training and Testing	25
4	**SYSTEM IMPLEMENTAION**	**26**
	4.1 Conceptual Architecture	26
	4.2 Input Module	28
	4.3 Pre Processing Module	28
	4.4 Analyzer Module	28
	4.5 Emotion Detector	28
5	**OUTPUT AND EVALUATION**	**29**
	5.1 Output during training	29
	5.5 Output from emotion detector	30

6	**APPLICATION AND FUTURE SCOPE**	**31**
	6.1 Application	31
	6.2 Future Scope	32
7	**CONCLUSION**	**33**
	REFERENCES	**34**

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1.1.	Illustration of 2D convolution	3
1.2.	Pooling	4
1.3.	ReLU graph	6
1.4.	Convex and Non -Convex Optimizers	7
1.5.	Convergence of Optimizers	8
1.6.	Illustration of dropouts.	9
1.7.	Square shaped kernels	11
1.8.	Haar features applied	11
1.9.	Haar Cascade classifier	12
2.1	Six Basic Emotions	14
2.2	FACS Example	15
3.1.	CNN Architecture diagram	20
3.2.	CNN Model Details	23
3.3	Dataset details	24
3.4	Confusion matrix	25
4.1.	Conceptual Architecture	26
4.2.	Flow of Components	27
5.1	Training output	29

5.2	Images from internet	30
5.3	Images from web camera	31

LIST OF ABBREVATIONS

ABBREVATION	EXPANSION
CNN	Convolution Neural Network
CONV2D	2D Convolution
FACS	Facial Action Coding System
ICML	International Conference on Machine Learning
PCA	Principal Component Analysis
ReLU	Rectified Neural Network

CHAPTER 1

INTRODUCTION

Facial expression detection, in today's world, is one of the most important features of human emotion recognition. Facial Expression can be defined as the facial changes observed in a person in response to the person's internal emotional state, intentions, or social communication. Automated facial expression recognition has a large variety of applications nowadays, such as neuromarketing, data-driven animation, sociable robotics, interactive games and many other human-computer interaction systems.

Basically, expression recognition is a task performed by human beings on everyday basis, effortlessly. However, it is not yet easily performed by computers. Although there have been many systems developed recently that have also presented accuracies larger than 95% in some conditions like frontal face, controlled environments, high-resolution images. But these simply appear to be misleading high-accuracy. Moreover, these systems do not represent most of the problems in real scenarios faced by face expression recognition systems.

The use of deep learning in expression detection is mainly for obtaining higher accuracy even for dynamic sequence of input received from webcam live. However, deep learning methods require huge dataset for obtaining such accuracies. Facial expression data is not as large as required by deep networks. Hence, the proposed system attempts to use a modified CNN that can work on even small datasets and ensure that the required accuracy is obtained. The modified CNN, which helps obtain high accuracy even with limited dataset uses data augmentation to recreate large datasets from the available data by

processes like applying combinations of translations, rotations and skewing for increasing the database.

1.1 Convolution

Convolution is mathematical operation where, two functions produce a third function as output. The output represents how the shape of one function is affected by another function.

The convolution operation is represented by asterisk (*) symbol. Let f and g be two functions, then convolution of these functions will be represented as f*g. Mathematical representation of convolution is:

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,d\tau.$$

Basically, there are 2 types of convolution:

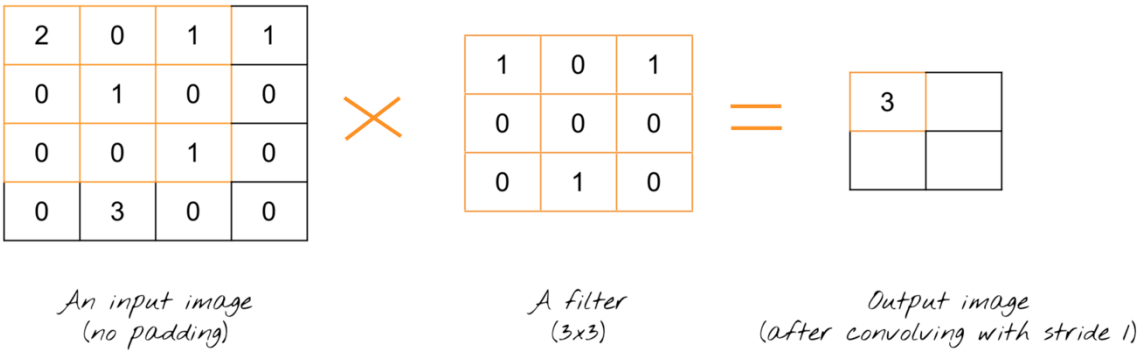
1. Continuous signal convolution
2. Discrete signal convolution

1.2 2D Convolution

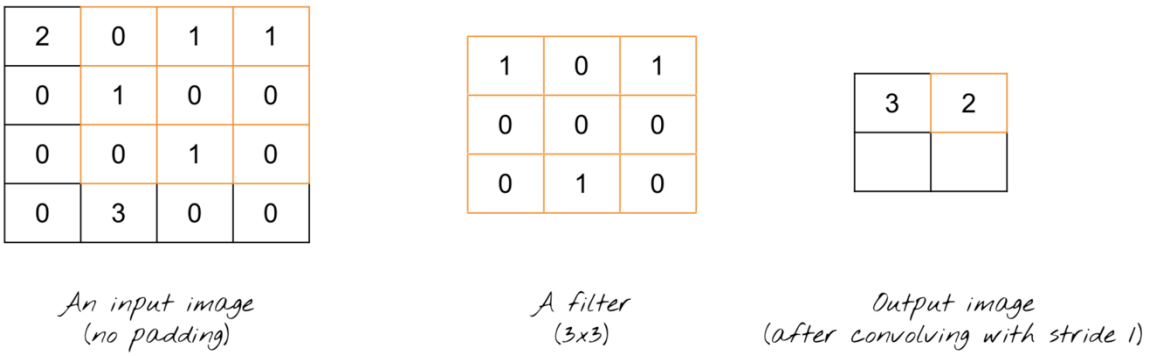
2D convolution network is used for grayscale images. The 2D convolution neural network needs two matrices.

1. Input Matrix
2. Kernel Matrix (Filter Matrix)

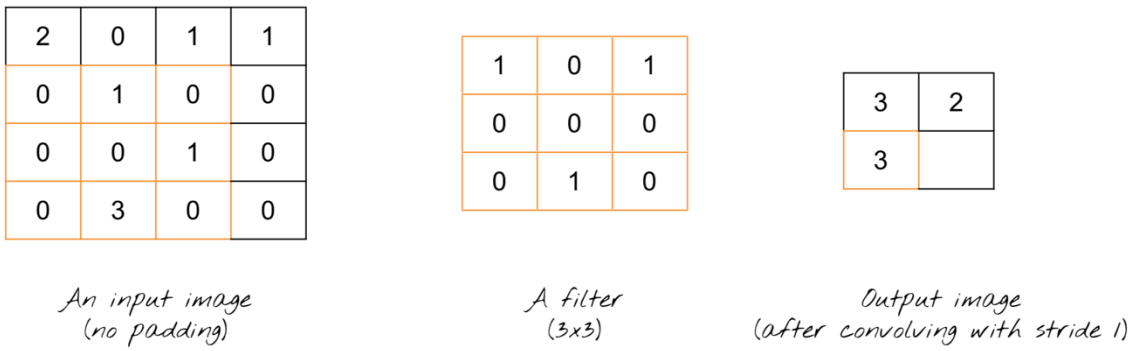
In 2D Convolution, two basic operations are involved: Multiplication and addition. The kernel matrix acts as a sliding window.



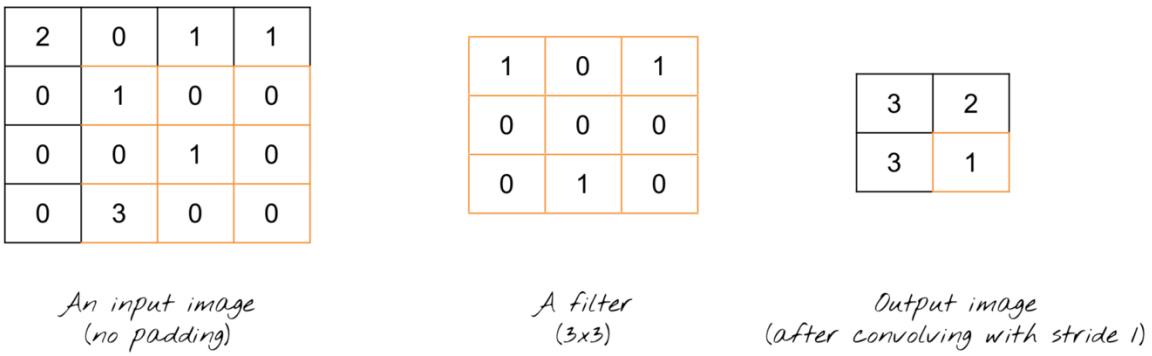
$$(2*1) + (0*0) + (1*1) + (0*0) + (1*0) + (0*0) + (0*0) + (0*1) + (1 * 0) = 3$$



$$(0*1) + (1*0) + (1*1) + (1*0) + (0*0) + (0*0) + (0*0) + (1*1) + (0*0) = 2$$



$$(0*1) + (1*0) + (0*1) + (0*0) + (0*0) + (1*0) + (0*0) + (3*1) + (0*0) = 3$$



$$(1*1) + (0*0) + (0*1) + (0*0) + (1*0) + (0*0) + (3*0) + (0*1) + (0*0) = 1$$

Figure 1.1: Illustration of 2D convolution

Notice that you moved the filter by only one column. The step size as the filter slides across the image is called a **stride**. Here, the stride is 1. The same operation is repeated to get the third output. A stride size greater than 1 will always downsize the image. If the size is 1, the size of the image will stay the same.

Convolution layer is the layer where feature learning happens.

1.3 Pooling

The next step following convolution is called pooling operation. Pooling is an operation done to reduce the dimensionality. Feature selection is done here. There are two types of pooling in general,

- Max Pooling:

  Max pooling takes the maximum value in the pooling window and provides it as the output.

  Thus, only the dominant features are selected for processing in the next round.

- Average Pooling:

  Here the average of all values present in the pooling window will be provided as the output.

  Thus, average pooling smoothens the image features.

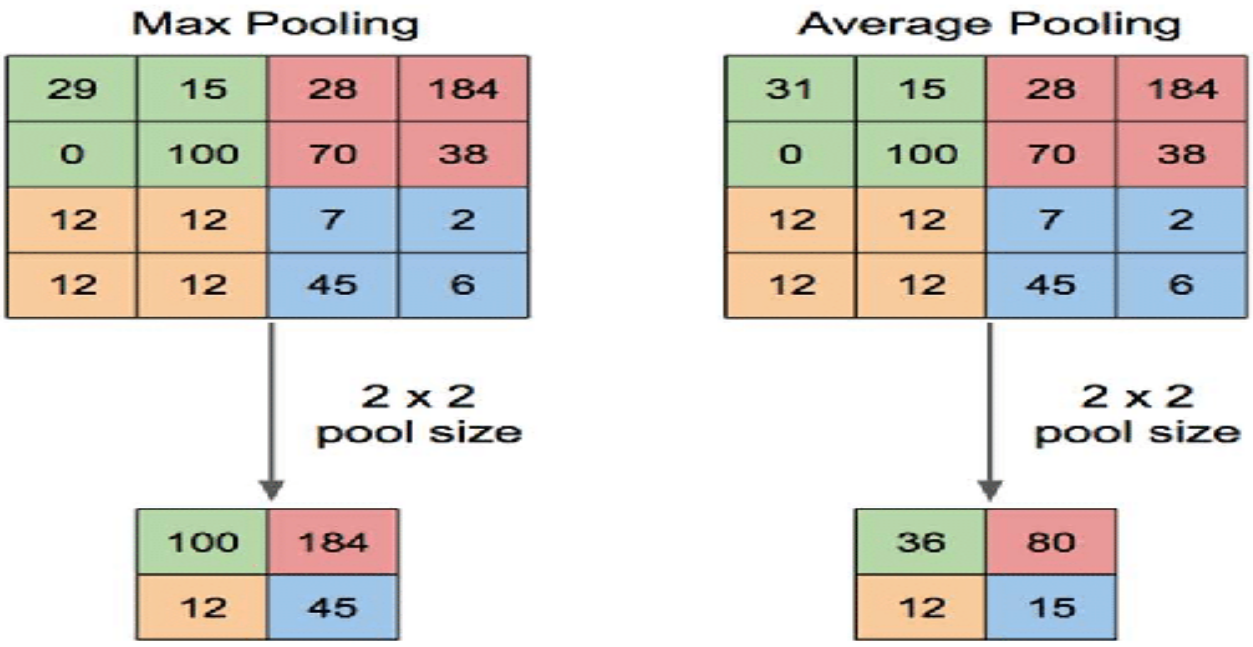


Figure 1.2: Pooling

1.4 Fully Connected Layer

After the convolution and pooling layer, a couple of fully connected layer was added. The output from convolution and pooling layer will be 2D or 3D but the fully connected layer requires 1D vector of numbers, so the output from the final pooling layer will be flattened. In this layer classification will happen.

1.5 Activation Function

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

In the proposed model Rectified Linear Units(ReLU) activation function is used. ReLU is defined as,

$$R(x) = \max(0,x)$$

i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x.

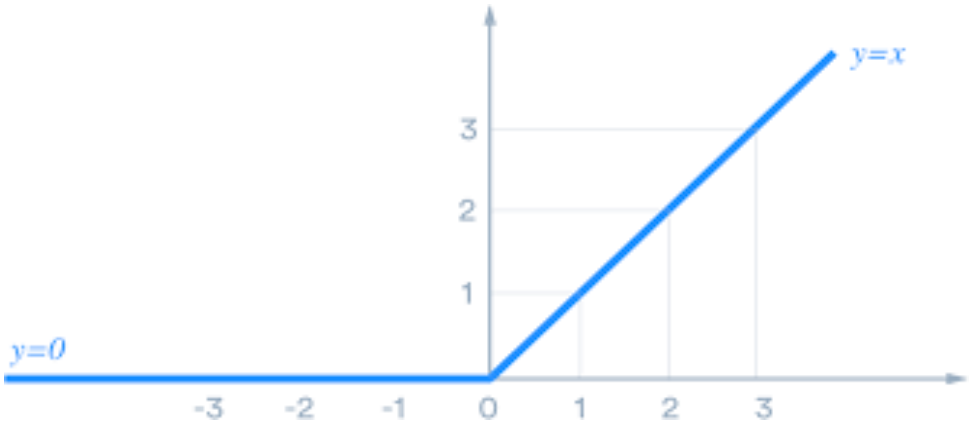


Figure 1.3: ReLU graph

1.6 Optimizer

Optimizers are functions which are used to change the internal learnable parameters of neural network like bias and weight in order to reduce the losses. Optimizers increases the learning rate of the neural networks.

Loss functions are used to indicate if the optimizers are working in the right direction to reach the global optimum.

There are mainly 2 types of optimization:

1. **Convex Optimization:** A function which involves only one optimum point with respect to the global optimum, which can either be minimum or maximum. There will be no concept of local optima here. Due to this, convex optimization problems are easier to solve.
   Example: Linear Programming, Least Squares.

2. **Non-Convex Optimization:** A function involving many local optima among which only one is the global optimum. Locating the global optimum faster on the loss surface is a non-convex optimization problem. This is trickier than the convex optimization.
   Example: Maximum likelihood estimation, Deep neural networks.

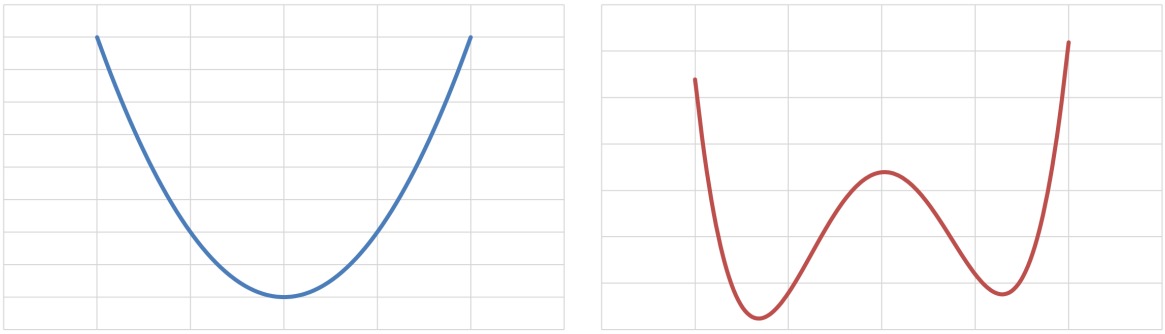


Figure 1.4: Convex Optimization (L) and Non - Convex Optimization(R)

There are different types of optimizers available, some of them are:

1. Momentum
2. Nesterov accelerated gradient (NAG)
3. Adaptive gradient algorithm (Adagrad)
4. Adaptive Moment Estimation (Adam)

For CNN, Adam optimizers are used since it converges faster than other optimizers. The convergence of the optimizers are visualised in figure 1.5.

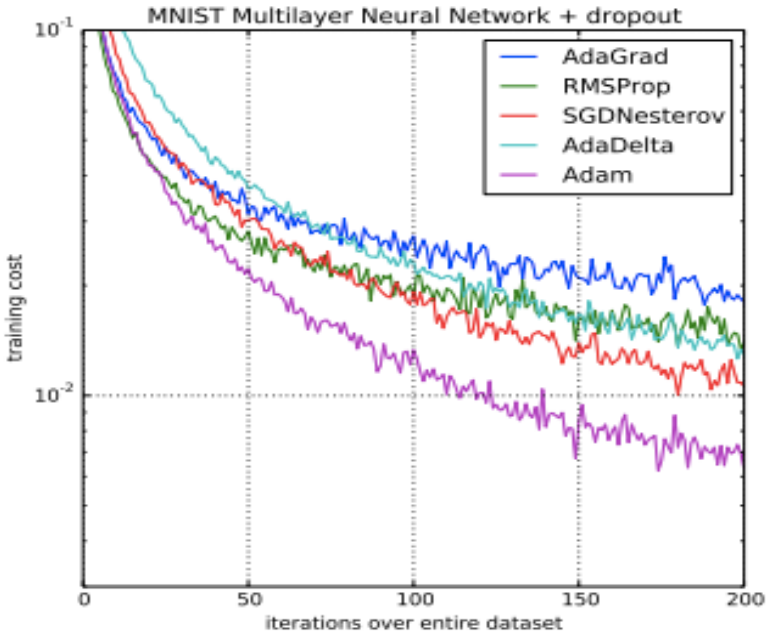


Figure 1.5: Convergence of Optimizers

Adam Configuration Parameters for our model is,

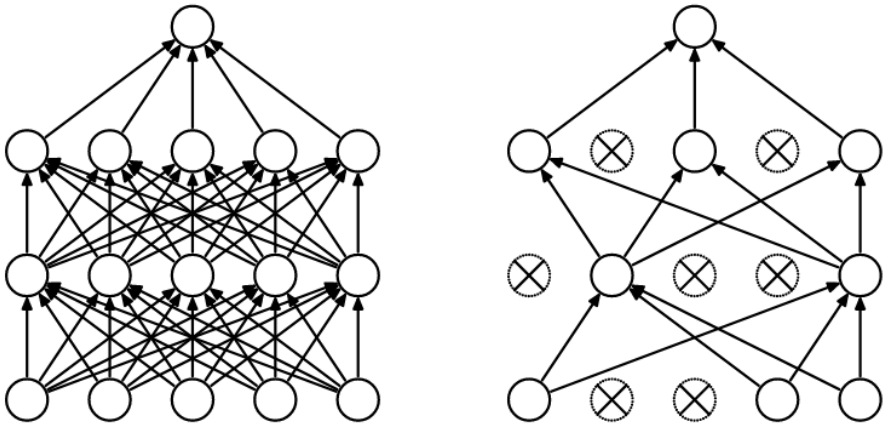
- **Alpha** : Also referred to as the learning rate or step size. The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training
- **beta1** : The exponential decay rate for the first moment estimates (e.g. 0.9).
- **beta2** : The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems).
- **Epsilon :** Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

1.7 Dropout

Dropout is a process where random neurons are ignored during the training of the neural network. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. Neighbouring neurons become to rely on this specialization, which if taken too far can result in a fragile model too specialized to the training data. This reliant on context for a neuron during training is referred to complex co-adaptations.

If neurons are randomly dropped out of the network during training, that other neurons will have to step in and handle the representation required to make predictions for the missing neurons. This is believed to result in multiple independent internal representations being learned by the network.



(a) Standard Neural Net (b) After applying dropout.

Figure 1.6 Dropout Illustration

The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

1.8 Batch Normalization

Batch normalization is used to reduce the amount by which the hidden unit values shift around (Covariance shift). Normalization is usually performed in the first layer, which increased the speed of learning. So using this on hidden layers tremendously increase the learning rate. Batch normalization allows layer of the neural network to train independently. This also helps in reducing the problem of overfitting.

1.9 Cascade Classifiers

Classifiers in computer vision are models trained with a few thousand samples of a specific object, which are known as positive examples and other images called as the negative examples of the same sizes. After training the classifiers will be able to detect the particular object for which it was trained for in a input image.

Cascade classifiers is a collection of several simple classifiers which are applied to an image to find the region of interest (Ex: A car or A face). The basic classifiers are mostly decision tree classifiers with a minimum of two leaves. When it comes to object detection, the input to the classifiers are Haar-like features.

1.10 Haar-like features

Haar features are digital image features, which uses square shaped kernels. The image is sectioned to rectangles and the kernels are applied on it.

For example:

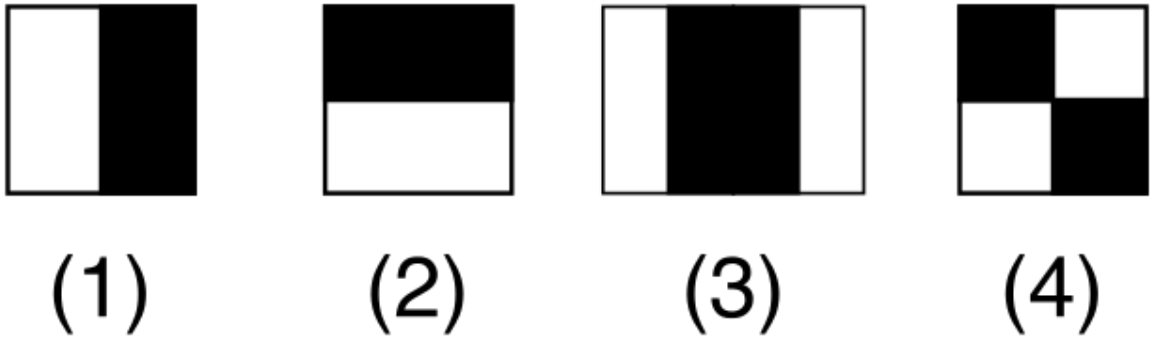


Figure 1.7 : Square Shaped Kernels

These kernels are applied to black and white image in order to detect the object.

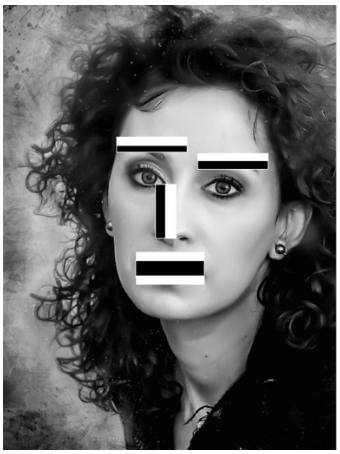


Figure 1.8: Haar features applied on relevant parts of the face

The figure 1.8 shows the application of the square shaped kernels to an image to detect the features of the face. To detect eyebrow, the Haar feature will be used (Fig 1.7 (1)) because forehead and eyebrow form lighter pixels- darker pixel like image. Similarly all the other features are also detected.

1.11 Haar Cascade Classifier

This classifier uses Haar-like features for object detection in an image. This algorithm basically has 4 important steps:

1. Collecting the Haar-like features
2. Constructing integral images
3. Adaboost training
4. Cascading Classifiers

- Haar-like features: The kernels are applied and the Haar-features are first extracted from the image.
- Integral images: Is used to quickly and efficiently calculate the sum of pixel intensities in a rectangular grid.
- Adaboost: From the chosen features, the best features are selected and the classifiers are trained on them.
- Cascading classifier: A series of week classifiers are organized to produce a strong classifier in order to increase the accuracy of the detection.

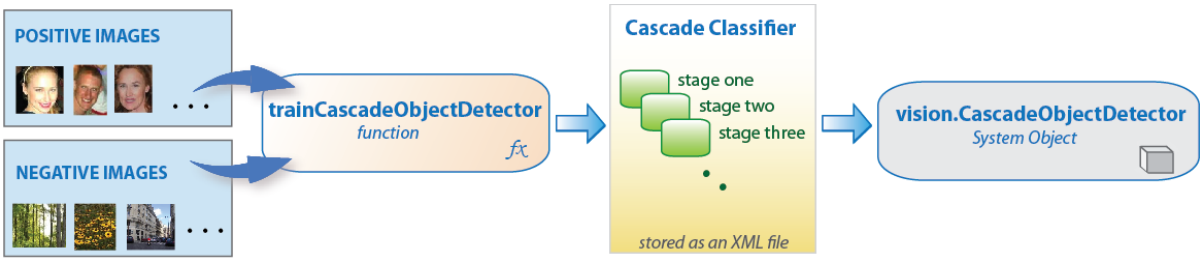


Figure 1.9 Haar Cascade Classifier

The proposed automated facial expression recognition systems receive the expected input (static image, image sequence or voice input) and typically give as output one of six basic expressions namely anger, sad, surprise, happy, disgust and fear. Expression recognition systems basically use a three-stage training procedure: feature learning, feature selection and classifier construction,

in the same order. The feature learning stage is responsible for the extraction of all features related to the facial expression. The feature selection selects the best features to represent the facial expression. At the end of the whole process, a classifier (or a set of classifiers, with one for each expression) is used to infer the facial expression, given the selected features.

CHAPTER 2

LITERATURE REVIEW

Facial expressions are generated by the contractions of various facial muscles which results in the temporary deformation of our eye, nose, lips, cheeks etc. It varies from person to person. Analysis of facial expressions plays a major role in various fields. This survey explains in detail about the various methods that are used for facial expression analysis.

2.1 Judgement based approach

Judgement based approaches are centered around the messages conveyed by the facial expressions. The facial expressions are classified into a set of emotions based on the opinion of the group of experts or coders as ground truth. Most of the facial expression analysis group into one of the six basic emotion classes specified by **Ekmen and Friesen** in **'Constants across culture in the face and emotions'** in 1971.

Figure 2.1: Six Basic Emotions

2.2 Sign based approaches

In sign-based approaches facial expressions are abstracted and described by their location and intensity. Therefore, a complete description framework consists of all possible perceptions that can occur on a face. This FACS (Facial Action Coding system) was proposed by **Ekman and Friesen** in **'A technique for the measurement of facial movement '** in 1978. FACS uses 44 action units for the description of facial expressions with respect to location and intensity. Similar coding schemes like EMFACS, MAX and AFFEX were proposed by **Izard, Dougherty and Hembree** in **'A system for indentifying Affect expressions by holistic judgements'** in 1983.

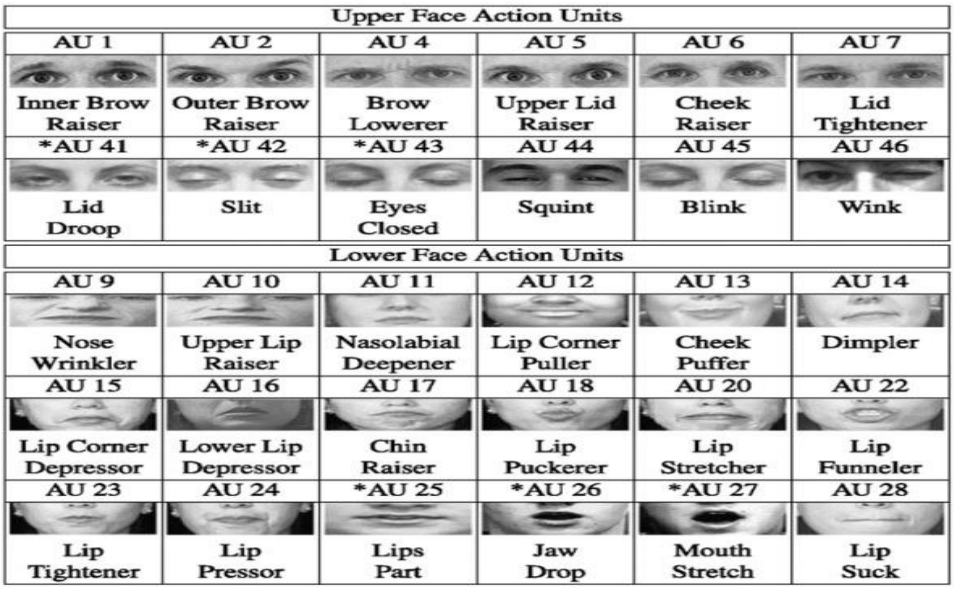


Figure 2.2: FACS Example

2.3 Reliability of ground truth coding

The labeling of employed databases to recognize or interpret facial expressions has improved recognition accuracy with respect to timing and intensity estimations. Ekman specifies certain points to be addressed while measuring facial expressions in **'Handbook of methods in non-verbal behavioural research'** in 1982.

2.4 Automatic face expression analysis

Automatic facial expression analysis is a complex task as the faces vary from person to person. Automatic facial expression analysis involves facial acquisition, facial feature extraction and analysis.

2.5 Face acquisition

Face acquisition features automatic face recognition in complex scenes with cluttered backgrounds. This can be done using active appearance models specified by **Lanitis, Taylor, Cootes** in **'Automatic interpretation and coding of face images using flexible models'** in 1997. Hong used the person spotter system by steffens in order to perform real time tracking of faces. The appearance of facial expressions depends on the angle and distance at which a given face is being observed. The distinctive facial features like eye, ear, nose serve as reference points in order to normalize test faces according to some generic face models specified in **'Transactions on pattern Analysis and machine intelligence'.** Illumination also plays a major role in face recognition. Light variations can be reduced by using Gabor wavelets in **'Proceedings of Circuits, Systems, Communications and Computers'** in 1999.

2.6 Facial feature extraction and representation

Facial feature extraction can be classified according to the features such as deformation or facial motion. Facial feature can be processed wholly or locally by focusing on specific areas. Facial features can be classified as intransient or transient as specified in **'Sixth international conference on Computer vision'** by **Black, Fleet and Yacoob** in 1998.

2.7 Deformation extraction based systems

Padgett and **Cottrell** presented an automatic facial expression interpretation system that could identify six basic emotions. Facial data was extracted from 32*32 pixel blocks. The blocks were placed on the eyes as well as the mouth and projected onto the top 15 PCA eigen vectors of 900 random patches which were extracted from the training images. For classification, the normalized projections were fed into an ensemble of 11 neural networks. Their output was summed and normalized again by dividing the average outputs for each emotions across all networks by their respective deviation over the entire training set. Altogether 97 images of 6 emotions from 6 males and 6 females were analysed and 86 percent performance was measured. This was presented in 'Proceedings of the 18th annual conference of the cognitive science society' in 1996.

2.8 Motion extraction based systems

Black and Yacoob analysed facial expressions with parametrized models for the mouth, the eyes, and the eyebrows and represented image flow with low-order polynomials. A concise description of facial motion was achieved with the aid of a small number of parameters from which they derived mid and high-level description of facial actions. The facial expression was modelled by registering the intensities of the mid-level parameters within temporal segments.

2.9 Hybrid systems

Hybrid facial expression analysis systems combine several facial expression analysis methods. Barlett proposes a system that integrates holistic difference image-based motion extraction coupled with PCA. Three different methods were employed together to produce less error patterns and a high accuracy. This was proposed in **'Advances in Neutral Information processing systems'** by **Barlett, Viola, Larsen** in 1996.

2.10 Speech based recognition systems

‘**Speech emotion recognition using Fourier parameters’** by **Kunxia Wang, Ning An, Bing Nan Li, Yanyong Zhang, Lian Li** states that the first- and second-order differences of harmony features also play an important role in speech emotion recognition. Therefore, a new Fourier parameter model using the perceptual content of voice quality and the first- and second-order differences for speaker-independent speech emotion recognition was proposed. Experimental results show that the proposed Fourier parameter (FP) features are effective in identifying various emotional states in speech signals. They improve the recognition rates over the methods using Mel frequency cepstral coefficient (MFCC) features by 16.2, 6.8 and 16.6 points on the German database (EMODB), Chinese language database (CASIA) and Chinese elderly emotion database (EESDB**)**

CHAPTER 3

NEURAL NETWORK DESIGN

The main component in an emotion detection system built using deep learning is the neural network. The neural network deployed in this system will be Convolution Neural Network (CNN). This chapter will discuss about the CNN model designed for this system and all the related details.

3.1 TensorFlow and Keras

The model is designed using TensorFlow and Keras package.

Usually, a neural network that is developed is trained by means of forward propagation and backward propagation. Computational graphs are constructed to compute gradients effectively. These gradients are used in the back propagation process while training our model, thus implementing gradient descent.

The data structure for the computational graph is implemented by it, as the functions of each of the weights are defined. Hence, while training, TensorFlow makes it easier for the model to compile and run faster and effectively.

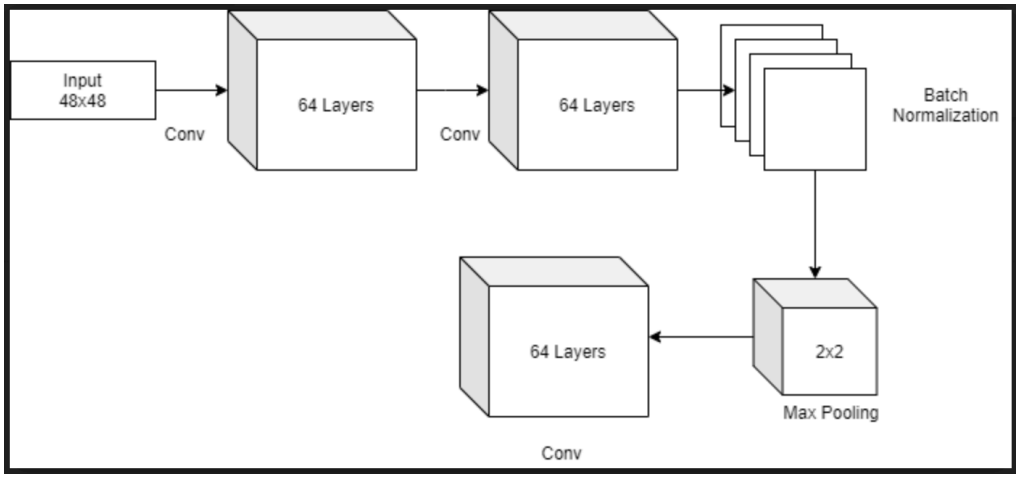
TensorFlow is an open source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least RankBrain in Google search and the fun DeepDream project.

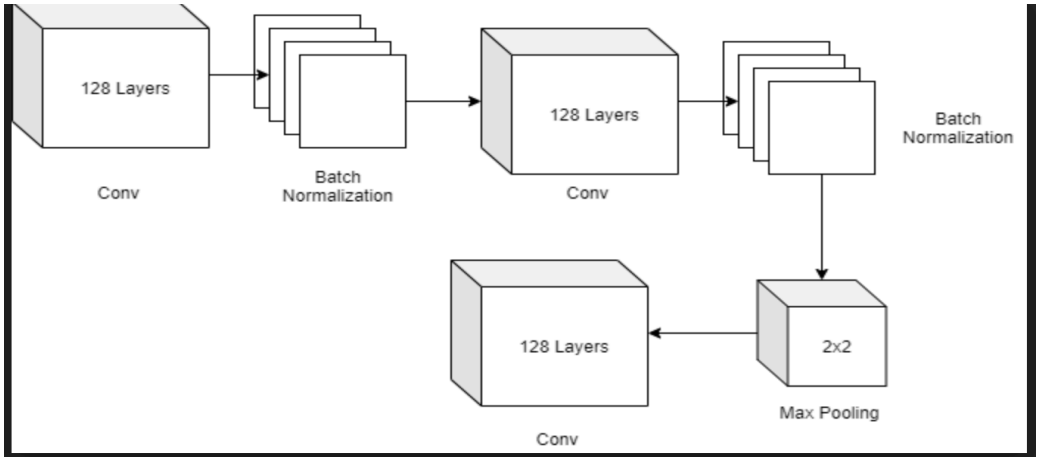
It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

Keras is a high level API that runs on the TensorFlow API. Hence, the models can be easily created using Keras, but it uses TensorFlow as its backend. Therefore, there is no issues related to performance when Keras is compared with TensorFlow. Keras has various types of models, layers, activations, optimizers, initializers, and many more to build a neural network easily.

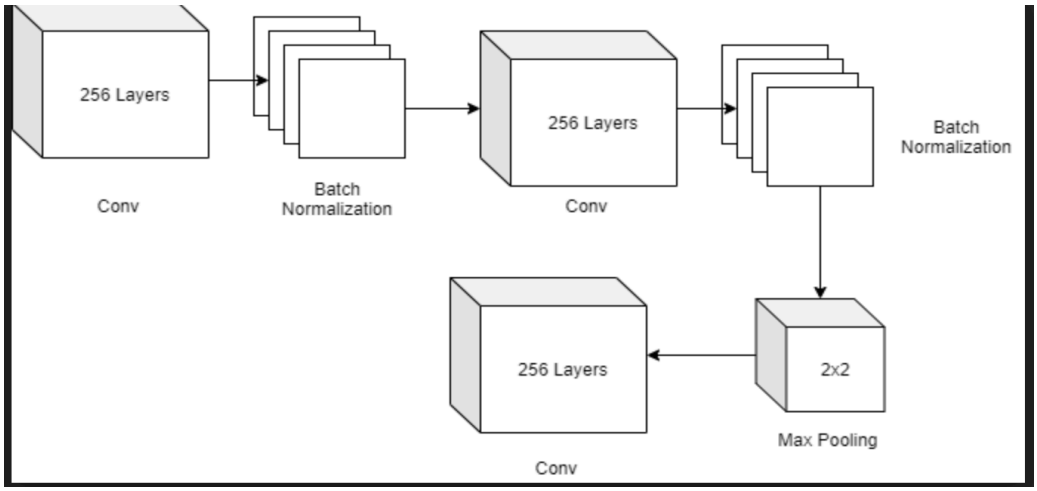
3.2 Architecture design of CNN



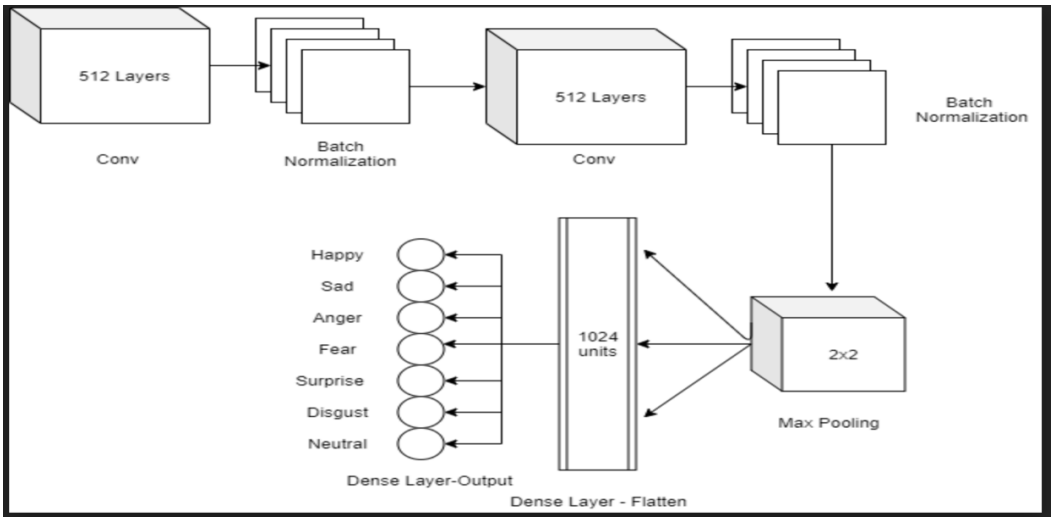
1.3 (a)



1.3 (b)



1.3 (c)



1.3 (d)

Figure 3.1: CNN architecture design

The structure of the architecture is as follows:

- The image input is fed having a dimension of 48*48
- Then there is two convolution layer with 64 filters with kernel size 3*3 applied to the input.

- Followed by batch normalization of the output from the convolution layer
- Then max pooling of size 2*2 is done with stride 2*2.
- The output from max pooling is fed to another convolution layer of 64 filters with 3*3 kernel size.

All the above layers are illustrated in the block diagram 1.3(a)

- The output from the convolution layer of 64 filters is fed to a convolution layer of 128 filters with kernel size 3*3.
- This is followed by a batch normalization layer and another convolution layer with 128 filters.
- Then max pooling of size 2*2 is performed and the output from this process is fed to a convolution layer with 128 filters.

All the above layers are illustrated in the block diagram 1.3 (b)

- The same pattern of layers as in the block diagram 1.3(b) is followed in the next set of layers also but with each convolution layer having 256 filters and later with 512 filters.
- Finally, there are a set of dense layers with 1024 units, produced by flattening the output of the final pooling layer.
- This is followed by the final dense layer with 7 classification outputs.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 64)        640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 64)        36928
_________________________________________________________________
batch_normalization_1 (Batch (None, 46, 46, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 23, 23, 128)       512
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 23, 23, 128)       147584
_________________________________________________________________
batch_normalization_3 (Batch (None, 23, 23, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 11, 11, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 11, 11, 256)       295168
_________________________________________________________________
batch_normalization_4 (Batch (None, 11, 11, 256)       1024
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 11, 11, 256)       590080
_________________________________________________________________
batch_normalization_5 (Batch (None, 11, 11, 256)       1024
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 5, 5, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 5, 5, 512)         1180160
_________________________________________________________________
batch_normalization_6 (Batch (None, 5, 5, 512)         2048
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 5, 5, 512)         2359808
_________________________________________________________________
batch_normalization_7 (Batch (None, 5, 5, 512)         2048
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1049088
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_7 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 903
=================================================================
Total params: 5,905,863
Trainable params: 5,902,151
Non-trainable params: 3,712
_________________________________________________________________
```

Figure 3.2: CNN Model details

3.3 Dataset

The dataset used for training and testing the CNN was obtained from a machine learning platform called Kaggle. This dataset was provided as a part of a challenge hosted by the ICML 2013.

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.

Train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

The training set consists of 28,709 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project.

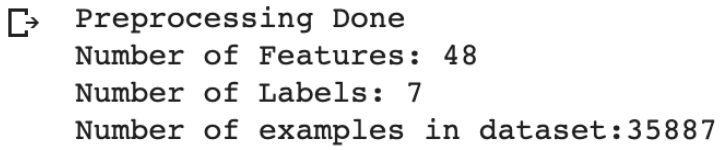


Figure 3.3 Dataset details

3.4 Training and Testing on dataset

Any CNN network needs to be trained and tested before deployment on dataset. For training and testing the dataset mentioned in section 3.2 was used. The testing accuracy of 66.369 % is obtained.

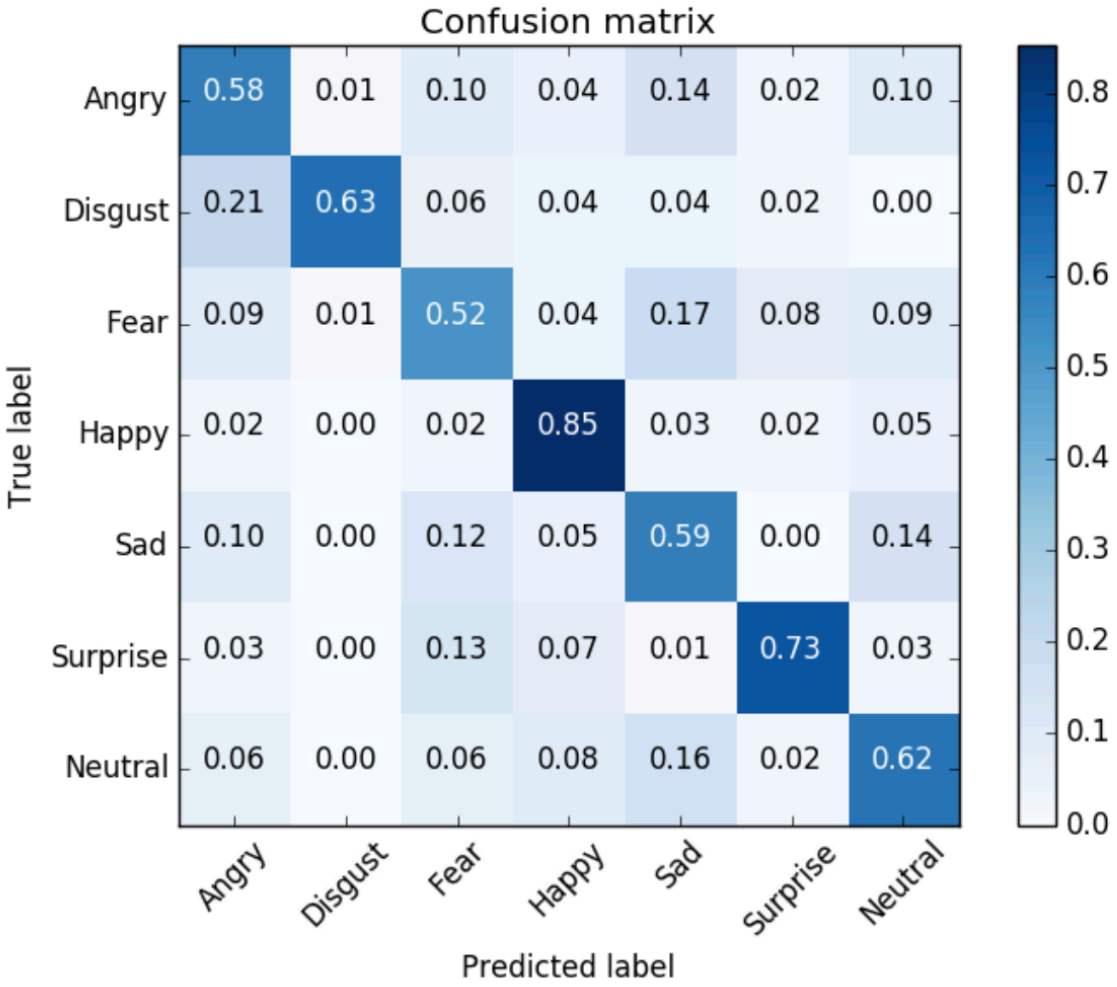


Figure 3.4 Confusion Matrix

CHAPTER 4

SYSTEM IMPLEMENTATION

This chapter is about the prototype implementation. It discusses about the various modules implemented and overall conceptual architecture of the system.

4.1 Conceptual architecture

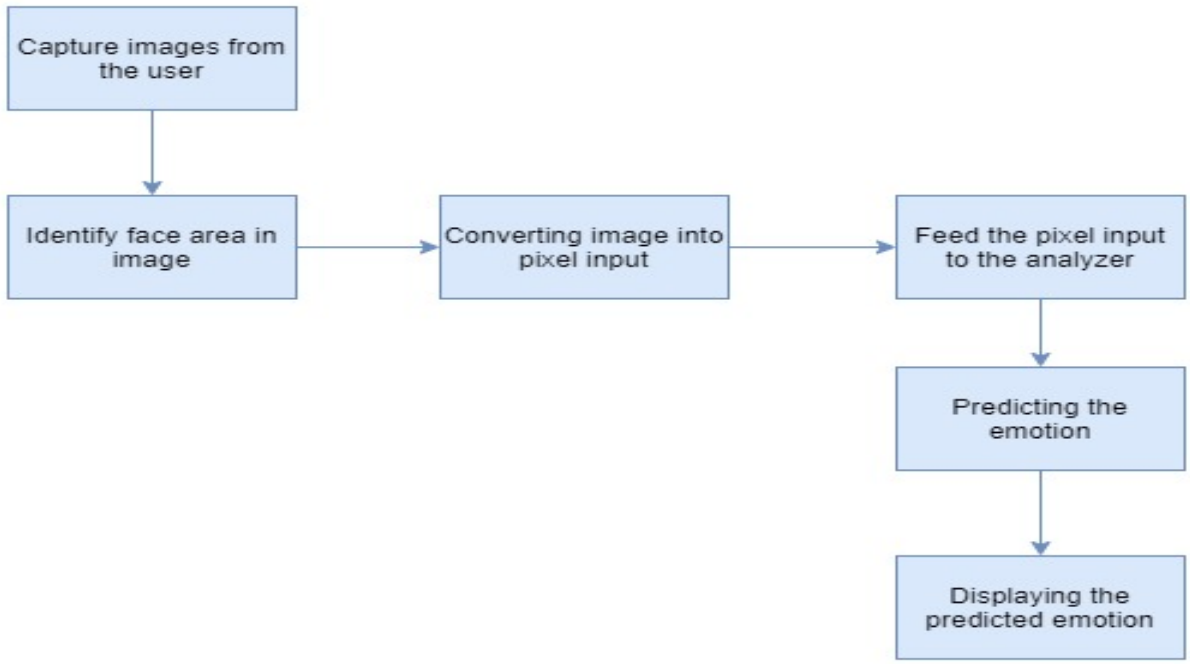


Fig 4.1 Conceptual Architecture

The conceptual architecture visually describes the various aspects of the system at a higher level. The purpose of this Conceptual Architecture is to provide a clear understanding of the system. The system takes the image from web camera as input data. The captured image input is given to the pre-processor in which the face area is recognized from the image input. The pre-processed image is converted into pixels and fed into the model for prediction. The model trained using CNN is used for prediction. The model generates the suitable emotion prediction using the trained data. The prediction is then displayed to the user as output.

The overall workflow between the various components is described in the following block diagram of the proposed system.

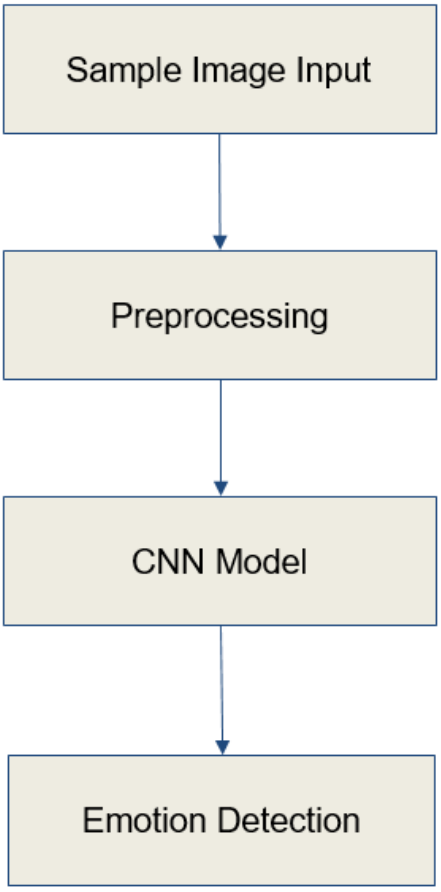


Fig 4.2 Flow of Components

4.2 Input module

Input module is used to recognize voice commands from the user. It captures facial inputs of the user from the web camera and feeds the input for further processing and for final prediction of the mood of user. This is done using **IPython.display** module in python. The image is captured and returned as jpeg image. The image name is then sent to preprocessor module.

4.3 Pre-processing module

Pre-processing module is used to process the input image so that it can be fed into the trained model for prediction. The image input generated from the camera is pre-processed in this module. The image is processed, and the facial part of the image is identified. This is done with the cascade classifier haarcascade_frontalface _default.xml. All this is done using **cv2 module** in python.

4.4 Analyzer module

The faces detected using the cascade classifier are resized to 48*48 size and normalized. After completing all this, the resultant array will be provided as the input to the CNN. All this is also done using **cv2 module** and **NumPy** in python.

4.5 Emotion detector

The Emotion Detector module is designed based on **Convolutional Neural Networks** model which is popularly used for image classification. The CNN model designed in chapter 3 is used here.

CHAPTER 5

OUTPUT AND EVALUATION

5.1 Output during training

```
Train on 29068 samples, validate on 3230 samples
Epoch 1/100
29068/29068 [==============================] - 27s 935us/step - loss: 2.0038 - accuracy: 0.2102 - val_loss: 1.8188 - val_accuracy: 0.2594
Epoch 2/100
29068/29068 [==============================] - 19s 655us/step - loss: 1.8354 - accuracy: 0.2464 - val_loss: 1.7917 - val_accuracy: 0.2594
Epoch 3/100
29068/29068 [==============================] - 19s 653us/step - loss: 1.7719 - accuracy: 0.2756 - val_loss: 1.6620 - val_accuracy: 0.3232
Epoch 4/100
29068/29068 [==============================] - 19s 652us/step - loss: 1.6490 - accuracy: 0.3373 - val_loss: 1.5581 - val_accuracy: 0.3718
Epoch 5/100
29068/29068 [==============================] - 19s 650us/step - loss: 1.5435 - accuracy: 0.3886 - val_loss: 1.4342 - val_accuracy: 0.4161
Epoch 6/100
29068/29068 [==============================] - 19s 652us/step - loss: 1.4704 - accuracy: 0.4221 - val_loss: 1.3637 - val_accuracy: 0.4588
Epoch 7/100
29068/29068 [==============================] - 19s 652us/step - loss: 1.4203 - accuracy: 0.4477 - val_loss: 1.2992 - val_accuracy: 0.4978
Epoch 8/100
29068/29068 [==============================] - 19s 653us/step - loss: 1.3775 - accuracy: 0.4667 - val_loss: 1.2873 - val_accuracy: 0.5111
Epoch 9/100
29068/29068 [==============================] - 19s 653us/step - loss: 1.3451 - accuracy: 0.4872 - val_loss: 1.2403 - val_accuracy: 0.5331
Epoch 10/100
29068/29068 [==============================] - 19s 652us/step - loss: 1.3168 - accuracy: 0.4985 - val_loss: 1.2157 - val_accuracy: 0.5387
Epoch 11/100
29068/29068 [==============================] - 19s 649us/step - loss: 1.2888 - accuracy: 0.5126 - val_loss: 1.1855 - val_accuracy: 0.5533
Epoch 12/100
29068/29068 [==============================] - 19s 651us/step - loss: 1.2597 - accuracy: 0.5234 - val_loss: 1.1816 - val_accuracy: 0.5440
Epoch 13/100
```

```
Epoch 84/100
29068/29068 [==============================] - 19s 655us/step - loss: 0.3513 - accuracy: 0.8868 - val_loss: 1.3616 - val_accuracy: 0.6659
Epoch 85/100
29068/29068 [==============================] - 19s 657us/step - loss: 0.3334 - accuracy: 0.8928 - val_loss: 1.3509 - val_accuracy: 0.6724
Epoch 86/100
29068/29068 [==============================] - 19s 656us/step - loss: 0.3397 - accuracy: 0.8916 - val_loss: 1.3001 - val_accuracy: 0.6709
Epoch 87/100
29068/29068 [==============================] - 19s 655us/step - loss: 0.3304 - accuracy: 0.8948 - val_loss: 1.4114 - val_accuracy: 0.6706
Epoch 88/100
29068/29068 [==============================] - 19s 656us/step - loss: 0.3196 - accuracy: 0.8988 - val_loss: 1.3851 - val_accuracy: 0.6619
Epoch 89/100
29068/29068 [==============================] - 19s 654us/step - loss: 0.3139 - accuracy: 0.9006 - val_loss: 1.3524 - val_accuracy: 0.6740
Epoch 90/100
29068/29068 [==============================] - 19s 657us/step - loss: 0.3339 - accuracy: 0.8926 - val_loss: 1.3434 - val_accuracy: 0.6768
Epoch 91/100
29068/29068 [==============================] - 19s 655us/step - loss: 0.3129 - accuracy: 0.9028 - val_loss: 1.3136 - val_accuracy: 0.6737
Epoch 92/100
29068/29068 [==============================] - 19s 654us/step - loss: 0.3139 - accuracy: 0.9002 - val_loss: 1.2969 - val_accuracy: 0.6681
Epoch 93/100
29068/29068 [==============================] - 19s 653us/step - loss: 0.3043 - accuracy: 0.9024 - val_loss: 1.3537 - val_accuracy: 0.6687
Epoch 94/100
29068/29068 [==============================] - 19s 654us/step - loss: 0.3046 - accuracy: 0.9039 - val_loss: 1.3835 - val_accuracy: 0.6777
Epoch 95/100
29068/29068 [==============================] - 19s 653us/step - loss: 0.3002 - accuracy: 0.9065 - val_loss: 1.3399 - val_accuracy: 0.6628
Epoch 96/100
29068/29068 [==============================] - 19s 653us/step - loss: 0.3017 - accuracy: 0.9046 - val_loss: 1.3796 - val_accuracy: 0.6740
Epoch 97/100
29068/29068 [==============================] - 19s 653us/step - loss: 0.2968 - accuracy: 0.9071 - val_loss: 1.4488 - val_accuracy: 0.6650
Epoch 98/100
29068/29068 [==============================] - 19s 658us/step - loss: 0.2883 - accuracy: 0.9097 - val_loss: 1.4585 - val_accuracy: 0.6619
Epoch 99/100
29068/29068 [==============================] - 19s 653us/step - loss: 0.2839 - accuracy: 0.9108 - val_loss: 1.4661 - val_accuracy: 0.6706
Epoch 100/100
29068/29068 [==============================] - 19s 654us/step - loss: 0.2841 - accuracy: 0.9126 - val_loss: 1.4795 - val_accuracy: 0.6687
Saved model to disk
```

Figure 5.1 Training output

5.2 Output from the emotion detector

- Output Evaluation on images taken from internet. These images are taken in well lighted conditions in professional camera. This was processed in the implemented system.

Figure 5.2 Images from internet

- Evaluation of images taken using the web camera and processed using the implemented system.

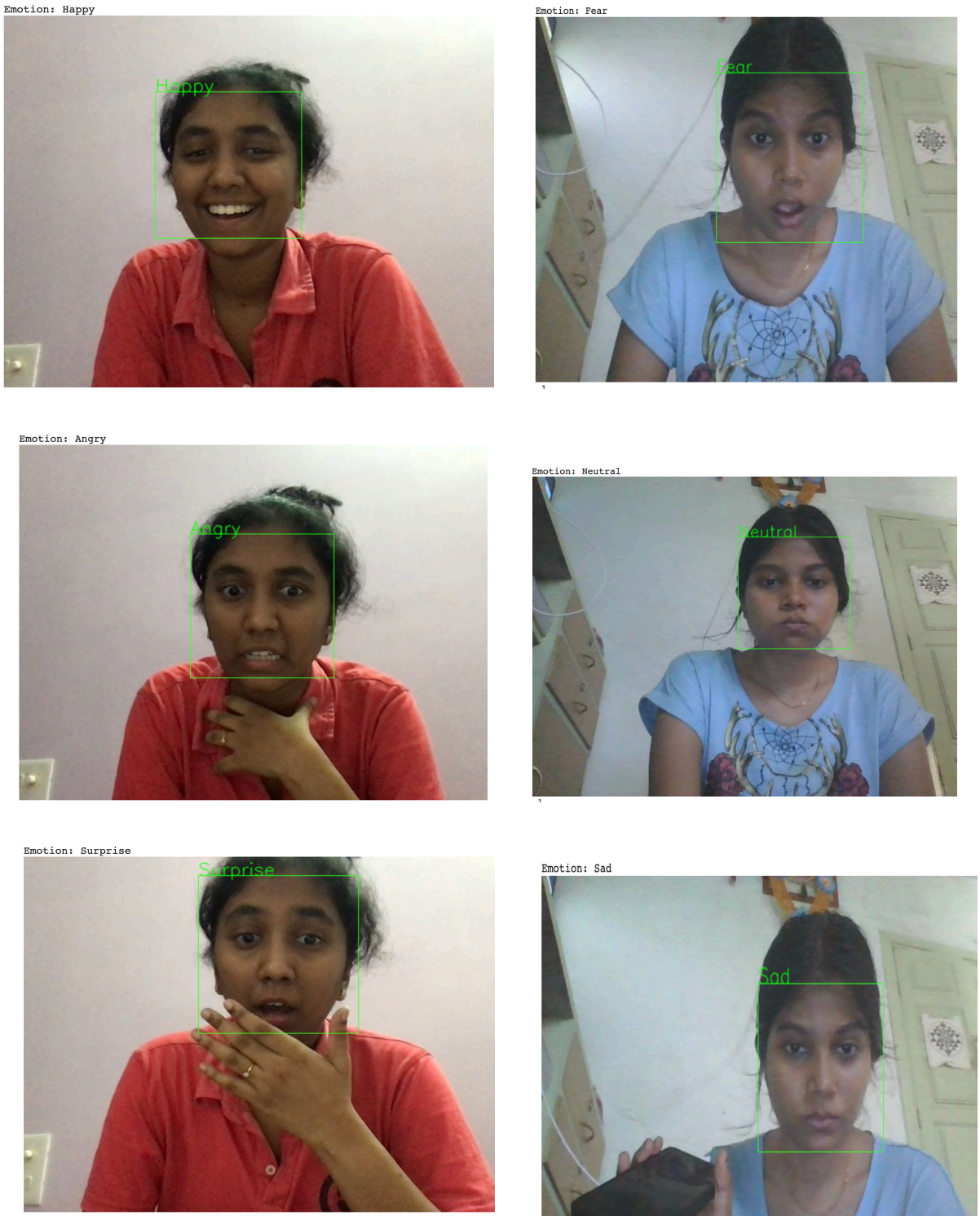


Figure 5.3 Images from webcam

CHAPTER 6

APPLICATION AND FUTURE SCOPE

6.1 Application

The product's scope is multi-fold. It can be used as a part of numerous applications. Some of the scopes of this system are discussed below. The system can be used to detect and track a user's state of mind. The system also finds its application in mini-marts or in shopping centre to view the feedback of the customers in order to enhance the business. Clever marketing is feasible using emotional knowledge of a person which can be identified by this system. The system can also be installed at busy places like airport, railway station or bus stands for detecting human faces and facial expressions of each person. If any suspicious emotions like angry or fear detected, the system could trigger an internal alarm. The system can further be used for educational purposes like getting feedback based on how the student is reacting during the class.

This system can help people in emotion related research fields to improve the processing of emotion data. It can also be used as plugins to monitor mentally disordered patient's behaviour at difficult level. System is fully automatic and has the capability to work with video feeds as well as images. It can recognize spontaneous expressions. This makes the system capable of being used even in Digital Cameras where in it is tuned such that the image is captured only when the person smiles or similar emotions. It can feature as added plugin in home automation. For instance, rooms in the house can set the lights, television to a person's current mood by figuring out his/her expression when he/she enters.

6.2 Future scope

The developed system is restricted only to predict the emotion of the facial expression captured at an instance of time by the system. The system can be extended to predict the emotion of the user captured over a sequence of period i.e a video input as a sequence of images to train the model. It can also be extended to predict the emotion based on the vocal input of the user i.e predict emotion based on sound patterns of user. This system can be delivered as a plug-in feature so that it can be incorporated with many applications that use this feature (emotion detection) in it.

CHAPTER 7

CONCLUSION

7.1 Conclusion

Thus, the purpose of this project has been achieved by efficiently predicting the emotion of the corresponding facial expression captured by the system. This has been achieved by using a trained Convolutional Neural Network architecture. The model has been successfully trained with sufficient input output samples and has produced a decent prediction accuracy as well. The performance has been improved by using increased kernel filters and sufficient dropout percentage so as to ensure the required features are captured and learnt by the model as well as the model does not overfit for the training samples.

VeriGuide - Originality Report
Individual Report

Background Information

File Name:	Emotion_Detection_Using_Facial_Expression.pdf
Report Generated On:	03/06/2020, 02:12:51 PM

Similarity Statistics Overview

Similar Sentence(s) Found By VeriGuide:	49 out of 342 sentences = 14.33%
Similar Sentence(s) Filtered by User:	49 out of 342 sentences = 14.33%
Sentence(s) Selected By User To Export:	0

Similarity Statistics for Each Source

Entry	Source	From	Similarity
1	https://core.ac.uk/download/pdf/48630142.pdf	Internet	15 / 342 = 4.39%
2	https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/	Internet	8 / 342 = 2.34%
3	https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/	Internet	5 / 342 = 1.46%
4	https://arxiv.org/pdf/1905.01118	Internet	4 / 342 = 1.17%
5	https://link.springer.com/article/10.1186/s13673-018-0156-3	Internet	4 / 342 = 1.17%
6	https://uu.diva-portal.org/smash/get/diva2:952138/FULLTEXT01.pdf	Internet	3 / 342 = 0.88%
7	https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/	Internet	2 / 342 = 0.58%
8	https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/	Internet	2 / 342 = 0.58%
9	https://www.embecosm.com/2020/03/04/face-detection-with-the-edgetpu-using-haar-cascades/	Internet	2 / 342 = 0.58%
10	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5856145/	Internet	2 / 342 = 0.58%
11	https://www.researchwithrutgers.com/en/publications/speech-emotion-recognition-using-fourier-parameters	Internet	2 / 342 = 0.58%
12	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.2604&rep=rep1&type=pdf	Internet	1 / 342 = 0.29%

13	https://cs231n.github.io/neural-networks-1/	Internet	1 / 342 = 0.29%
14	https://pdfs.semanticscholar.org/0316/7776e17bde31b50f294403f97ee068515578.pdf	Internet	1 / 342 = 0.29%
15	https://stackoverflow.com/questions/30857908/face-detection-using-cascade-classifier-in-opencv-python/54495524	Internet	1 / 342 = 0.29%
16	https://towardsdatascience.com/an-intro-to-deep-learning-for-face-recognition-aa8dfbbc51fb	Internet	1 / 342 = 0.29%
17	https://www.cin.ufpe.br/~rps/Artigos/Face%20Detection%20-%20%20A%20Survey.pdf	Internet	1 / 342 = 0.29%
18	https://www.ri.cmu.edu/pub_files/pub3/sayette_m_a_2001_1/sayette_m_a_2001_1.pdf	Internet	1 / 342 = 0.29%
19	https://www.sciencedirect.com/topics/psychology/facial-expression	Internet	1 / 342 = 0.29%

REFERENCES

[1] Jie Wang, Zihao Li, “Research on Face Recognition Based on CNN”, 2018 IOP Conf. Ser.: Earth Environ. Sci. 170 032110.

[2] https://towardsdatascience.com/an-intro-to-deep-learning-for-face-recognition.

[3] Andre Teixeira Lopesa, Edilson de Aguiarb, Alberto F. De Souzaa, Thiago Oliveira-Santosa, “Facial Expression Recognition with Convolutional Neural Networks: Coping with Few Data and the Training Sample Order”.

[4] Ankit S. Vyas, Harshadkumar B. Prajapati , Vipul K. Dabhi, “Survey on Face Expression Recognition using CNN” .

[5] Gholamreza Anbarjafari, Alvo Aabloo, “Expression Recognition by Using Facial And Vocal Expressions”.

[6] Kunxia Wang, Ning An, Bing Nan Li, Yanyong Zhang, Lian Li, “Speech emotion recognition using Fourier parameters”

[7] Lanitis, Taylor, Cootes, “Automatic interpretation and coding of face images using flexible models”.

APPENDIXES

Data preprocessing

```
import warnings
import sys, os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from keras.losses import categorical_crossentropy
from keras.optimizers import Adam
from keras.regularizers import l2

warnings.filterwarnings("ignore")

data = pd.read_csv('fer2013/fer2013.csv')

width, height = 48, 48

datapoints = data['pixels'].tolist()

#getting features for training
X = []
for xseq in datapoints:
    xx = [int(xp) for xp in xseq.split(' ')]
    xx = np.asarray(xx).reshape(width, height)
    X.append(xx.astype('float32'))

X = np.asarray(X)
X = np.expand_dims(X, -1)

#getting labels for training
y = pd.get_dummies(data['emotion']).to_numpy()

#storing them using numpy
np.save('fdataX', X)
np.save('flabels', y)

print("Preprocessing Done")
```

```
print("Number of Features: "+str(len(X[0])))
print("Number of Labels: "+ str(len(y[0])))
print("Number of examples in dataset:"+str(len(X)))
print("X,y stored in fdataX.npy and flabels.npy respectively")

num_features = 64
num_labels = 7
batch_size = 64
epochs = 100
width, height = 48, 48

x = np.load('./fdataX.npy')
y = np.load('./flabels.npy')

x -= np.mean(x, axis=0)
x /= np.std(x, axis=0)

#for xx in range(10):
#    plt.figure(xx)
#    plt.imshow(x[xx].reshape((48, 48)), interpolation='none',
cmap='gray')
#plt.show()

#splitting into training, validation and testing data
X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.1, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,
test_size=0.1, random_state=41)

#saving the test samples to be used later
np.save('modXtest', X_test)
np.save('modytest', y_test)
```

CNN Model

```
model = Sequential()

model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu',
input_shape=(width, height, 1), data_format='channels_last',
kernel_regularizer=l2(0.01)))
model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))
```

```
model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*num_features, kernel_size=(3, 3), activation='relu',
padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*2*num_features, kernel_size=(3, 3),
activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*2*num_features, kernel_size=(3, 3),
activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3),
activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(2*2*2*num_features, kernel_size=(3, 3),
activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(2*2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*num_features, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(num_labels, activation='softmax'))

model.compile(loss=categorical_crossentropy,
              optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999,
epsilon=1e-7),
              metrics=['accuracy'])
```

Training and saving the model

```
model.fit(np.array(X_train), np.array(y_train),
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(np.array(X_valid), np.array(y_valid)),
          shuffle=True)

#saving the  model to be used later
fer_json = model.to_json()
with open("fer.json", "w") as json_file:
    json_file.write(fer_json)
model.save_weights("fer.h5")
print("Saved model to disk")
```

Module to take picture using web cam

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video:
true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.

google.colab.output.setIframeHeight(document.documentElement.scrollHeig
ht, true);
```

```
      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

Emotion Detection module

```
from __future__ import division
from keras.models import Sequential
from keras.layers import Dense
from keras.models import model_from_json
import numpy
import os
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

#loading the model
json_file = open('fer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("fer.h5")
print("Loaded model from disk")

#setting image resizing parameters
WIDTH = 48
HEIGHT = 48
x=None
y=None
```

```
labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise',
'Neutral']

#loading image
full_size_image = cv2.imread("/content/test2.jpg")
print("Image Loaded")

gray=cv2.cvtColor(full_size_image,cv2.COLOR_RGB2GRAY)

face =
cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')

faces = face.detectMultiScale(gray)

print(face.empty())

#detecting faces
for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
        cropped_img =
np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
        cv2.normalize(cropped_img, cropped_img, alpha=0, beta=1,
norm_type=cv2.NORM_L2, dtype=cv2.CV_32F)
        cv2.rectangle(full_size_image, (x, y), (x + w, y + h), (0, 255,
0), 1)
        #predicting the emotion
        yhat= loaded_model.predict(cropped_img)
        cv2.putText(full_size_image, labels[int(np.argmax(yhat))], (x,
y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1, cv2.LINE_AA)
        print("Emotion: "+labels[int(np.argmax(yhat))])

cv2_imshow(full_size_image)
cv2.waitKey()
```