



Arab Academy for Science, Technology and Maritime Transport

College of Computing and Information Technology

Computer Science Department

B. Sc. Final Year Project

Emotion recognition using facial expressions

Presented By:

Ziad Hamdy El-Nouby Adam

Youssef Khaled

Mohmed Ebrahim Abd Allah

Mohamed Omar Mohamed

Ahmed Ali Ahmed Abd Al Mowla

Supervised By:

Ass.Prof.DR/ Yasser Omar

June-2020

DECLARATION

We hereby certify that this report, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in Computer Science Bachelor, is all my own work and contains no plagiarism. By submitting this report, I agree to the following terms:

Any text, diagrams or other material copied from other sources (including, but not limited to, books, journals, and the internet) have been clearly acknowledged and cited followed by the reference number used; either in the text or in a footnote/endnote. The details of the used references that are listed at the end of the report are confirming to the referencing style dictated by the final year project template and are, to my knowledge, accurate and complete.

I have read the sections on referencing and plagiarism in the final year project template. I understand that plagiarism can lead to a reduced or fail grade, in serious cases, for the Graduation Project course.

Student Name: Ziad Hamdy El-Nouby
Registration Number: 16108596

Signed:
Date: 26/6/2020

Student Name: Youssef Khaled
Registration Number: 16108655

Signed:
Date: 26/6/2020

Student Name: Mohmed Ebrahim
Registration Number: 17208519

Signed:
Date: 26/6/2020

Name: Mohamed Omar Mohamed
Registration Number: 16108727

Signed:
Date: 26/6/2020

Student Name: Ahmed Ali Ahmed Abd Al Mowla`
Registration Number: 16108676
Signed:
Date: 26/6/2020

TABLE OF CONTENTS

List of figures	5
List of acronyms	7
chapter 1	8
1 Introduction	9
1.1 Motivation.....	9
1.2 Problem Statement	9
1.3 Objective	10
1.4 Methodology	10
chapter 2	11
2.1 neural networks (NN).....	12
2.2 Neural network layers.....	12
2.3 Components of NN	13
2.3.1- Neurons	13
2.3.2-Connections	13
2.3.3 Weights and Biases.....	14
2.3.4 Activation Functions or Transfer Function	14
2.3.4.1 Linear Activation Function	15
2.3.4.2 Non-linear Activation Function	15
2.3.5 Neural networks training process	16
2.3.6 Cost function.....	16
2.3.7 Gradient descent.....	16
2.3.8 Backpropagation	17
2.3.9 Hyperparameter	17
2.4 Learning paradigms	18
2.4.1-Supervised learning	18
2.4.2-Unsupervised learning	18
2.4.3-Reinforcement learning	18
2.5 Important Activation Functions	19
2.5.1 ReLU (Rectified Linear Unit)	19
2.5.2 Soft-Max function	19
2.6 Most important types of neural networks	20
2.6.1 RNN(Recurrent Neural Networks)	20
2.6.2 CNN(convolutional neural network)	21
2.6.2.1 Architecture of convolutional neural network	21
2.6.2.1.1 Convolution	21
2.6.2.1.2 Non Linearity (ReLU)	22
2.6.2.1.3 Pooling or Sub Sampling	23
2.6.2.1.4 Classification (Fully Connected Layer).....	24
2.7 Concepts and techniques used in our model	24
2.7.1 Residual modules	24

2.7.2 Depth-wise separable convolutions	25
2.7.2.1-Depth-wise operation	25
2.7.2.2-Point-wise operation	26
2.7.3 Global average pooling	26
2.7.4 Batch normalization	27
2.7.5 Adam optimizer	27
chapter 3	28
3.1 Categorical model.....	29
3.2 FACS model	29
3.3 Dimensional model.....	31
3.4 FEC-16D model	32
chapter 4	34
4.1 Dataset	35
4.1.1-Emotion label	35
4.1.2-Image matrix	35
4.1.3-Usage	35
4.2 Model.....	36
4.2.1 Sequential	36
4.2.2 Functional	37
4.3 Model Architecture.....	38
4.3.1 Residual Block	40
4.4.2 Depth-wise separable convolutions neural networks	41
chapter 5	43
5.1 Results	44
5.2 GUI	46
5.3 Future work	54
5.4 CONCLUSIONS	55
5.5 Reference.....	55

LIST OF FIGURES

Figure 2.1	12
Figure 2.2	13
Figure 2.3	14
Figure 2.4.....	15
Figure 2.5	15
Figure 2.6	18
Figure 2.7	19
Figure 2.8	19
Figure 2.9	20
Figure 2.10	20
Figure 2.11	21
Figure 2.12	22
Figure 2.13	22
Figure 2.14.....	23
Figure 2.15	24
Figure 2.16	25
Figure 2.17	25
Figure 2.18	26
Figure 2.19	26
Figure 3.1	29
Figure 3.2	30
Figure 3.3	30
Figure 3.4.....	31
Figure 3.5	33
Figure 3.6	33
Figure 3.7	33
Figure 4.1	35
Figure 4.2	36
Figure 4.3	36
Figure 4.4.....	37
Figure 4.5	39
Figure 4.6	40
Figure 4.7	41
Figure 4.8	41
Figure 4.9	42
Figure 5.1	44
Figure 5.2	44

Figure 5.3	45
Figure 5.4.....	45
Figure 5.5	46
Figure 5.6	47
Figure 5.7	47
Figure 5.8	48
Figure 5.9	48
Figure 5.10	49
Figure 5.11	50
Figure 5.12	51
Figure 5.13	52
Figure 5.14	52
Figure 5.15	53
Figure 5.16	53
Figure 5.17	54

LIST OF ACRONYMS

AI: artificial intelligence

ML :Machine learning

DL: Deep learning

ANN: Artificial Neural Network

CNN :Convolutional neural network

RNN :Recurrent neural network

ReLU : Rectified Linear Unit

FC : Fully Connected Layer

chapter 1

(Introduction)

Introduction

Inter-personal human communication includes not only spoken language but also non-verbal cues such as hand gestures, facial expression and tone of the voice, which are used in expressing feelings and giving feedback. According to Albert Mehrabian communication study verbal components convey one-third of human communication, and nonverbal components convey two-thirds. Among several nonverbal components facial expressions are one of the main information channels in interpersonal communication. Therefore, Automatic facial expression analysis has received significant attention from the computer vision community due to its numerous applications such as emotion prediction, expression retrieval , photo album summarization, candid portrait selection.

1.1 Motivation

Emotion recognition can be used in many applications to solve a lot of problems in many fields like :

- Automotive industry : Emotion recognition can be used for detecting sleeping drivers and then take actions to alarm the driver or stop the car to save passengers life.
- Learning applications : Emotion recognition may be used to monitor concentration of learners and to analyze their satisfaction of the content and instructor.
- Digital advertising and market research : Emotion recognition can be used to measure target users subjective satisfaction of the displayed advertisement in a real-time manner
- Video game testing : we can use emotion recognition to monitor user's reactions on video games

1.2 Problem Statement

Most of CNN architectures have tons of parameters in their last fully connected layers and this leads to heavy computations that affects performance speed .

1.3 Objective

We aim to build a real-time system that combines speed performance and high accuracy emotion recognition which can be implemented and integrated in multiple business applications with low computing power.

1.4 Methodology

Our project consists of three phases :

1- Dataset downloading and preprocessing:

- Download FER-2013 dataset from **Kaggle**
- Extract pixels' values string from the CSV file
- Reshape images matrices and resize them into 48 X 48 images
- Normalize pixels' values to be in range of [-1 ,1]

2- Building the model:

- Design neural network architecture
- Build the model using (TensorFlow , Keras , Numpy , Pandas) python libraries
- Train our model using dataset collected in phase1
- Evaluate the model

3- Interface design

- Design graphical user interface using Tkinter python library
- Write a python script to open the camera and detect user's face , capture the photo and predict his emotion.

chapter 2

(Theoretical background)

2.1 neural networks (NN)

An NN is based on a collection of connected units or nodes called artificial neurons, neural networks that constitute human brains. And the output of each neuron is calculated by a nonlinear function of its total input. The connections are called edges. Neurons and edges typically have a weight that is adjusted as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the Hidden layers multiple times. generally without being programmed with task-specific rules.

2.2 Neural network layers

The neural network consists of 3 layers types, the input layer, the hidden layer(s), and the output layer,

- 1-The input layer is the layer that introduces inputs (features) to the network
- 2-The hidden layer(s) which is optional may be 0 or more depending on the problem we want to solve and it performs nonlinear transformations of the inputs entered into the network
- 3- The output layer is the last layer and it produces the output (prediction) of the neural network

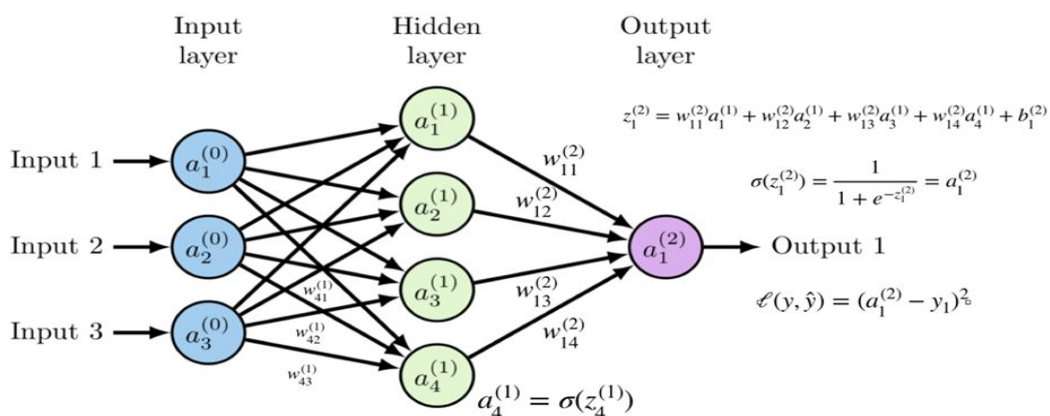


Figure 2.1

2.3 Components of NN :

2.3.1- Neurons

NNs are composed of artificial neurons which retain the biological concept of neurons ,we can consider each neuron in NN as a logistic regression model which receive inputs(signals), combine them with it's own bias using sum of products function (SOP) and provide a mapping between inputs and output using an activation function. The important characteristic of the activation function is that it provides a smooth, differentiable transition as input values change. a small change in input produces a small change in output.

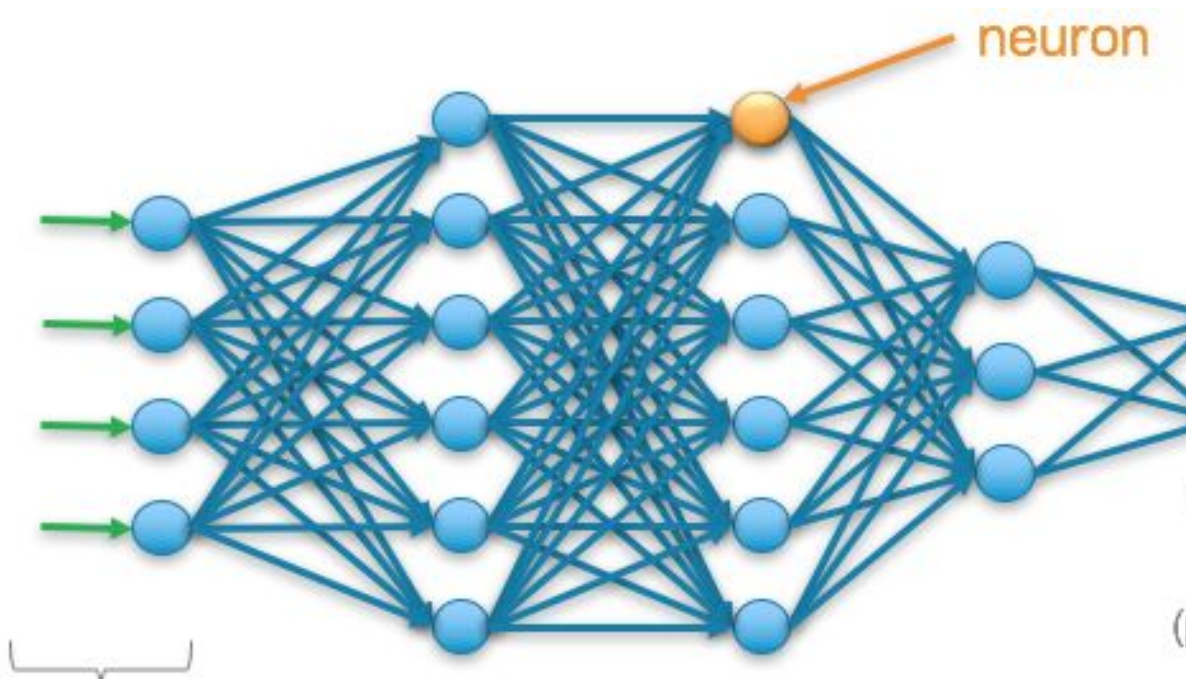


Figure 2.2

2.3.2-Connections :

NN connections used to connect neurons with each other and they have weights inside them that transform input and feed it to the next neuron.

And these two objects are the fundamental building blocks of the neural network. More complex neural networks are just models with more hidden layers and that means

more neurons and more connections between neurons. And this more complex web of connections (and weights and biases) is what allows the neural network to “learn” the complicated relationships hidden in our data.

2.3.3 Weights and Biases

As mentioned each neuron is a logistic regression model so weights and biases of all neurons in the network are the parameters that we should adjust to change the predictions made by the NN . So each neuron output = Activation function (Z) ; And $Z = [W] * [X] + \text{bias}$.

Where $[X]$ is the input vector (features).

, $[W]$ is the weights vector and it is the estimated slope parameter.

And bias is very similar to the intercept term from regression

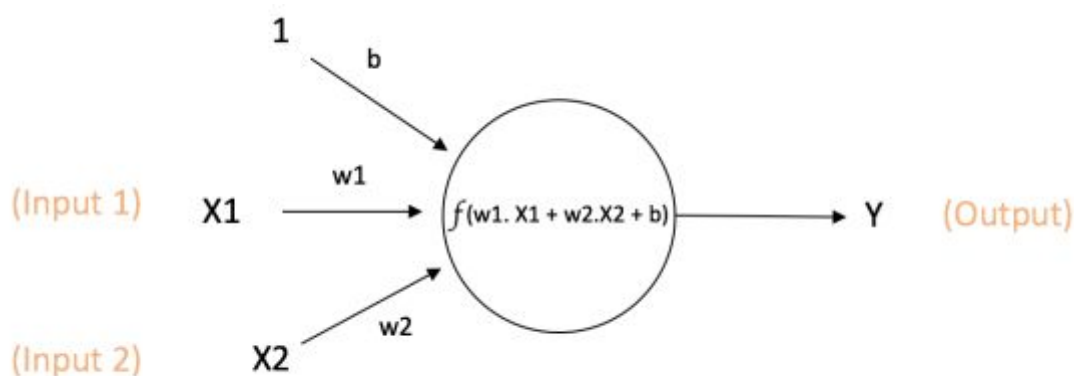


Figure 2.3

2.3.4 Activation Functions or Transfer Function

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1. The Activation Functions can be basically divided into 2 types:

1. Linear Activation Function

2. Non-linear Activation Functions

2.3.4.1 Linear Activation Function

The function's output is a line or linear.

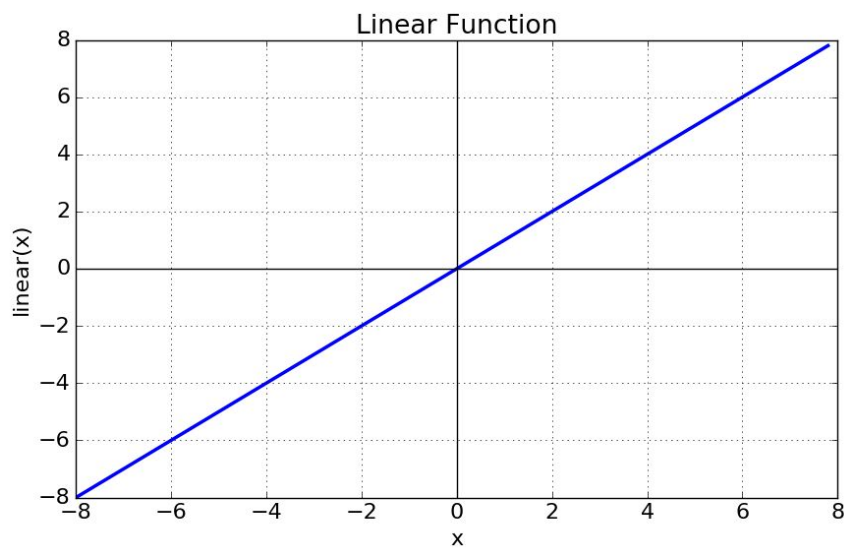


Figure 2.4

2.3.4.2 Non-linear Activation Function

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

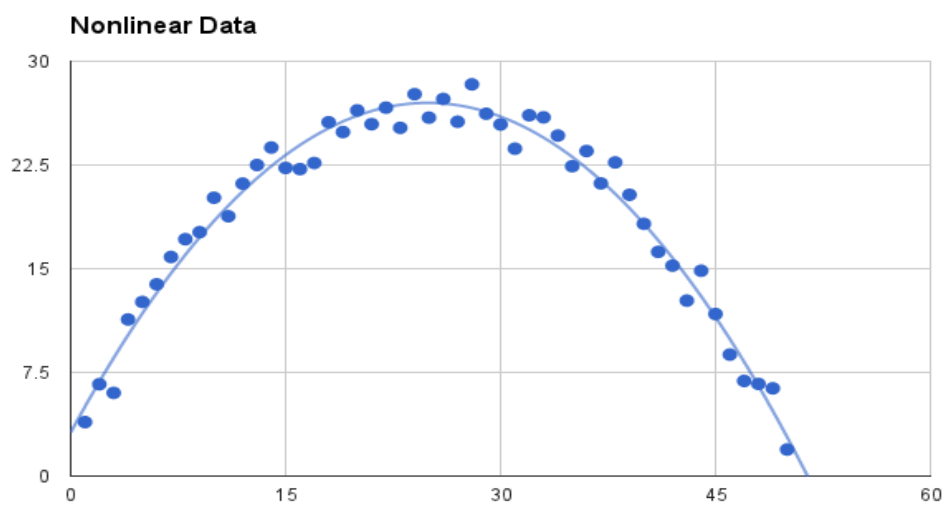


Figure 2.5

2.3.5 Neural networks training process

The training process of neural networks is like that many of learning models which is define a cost function and minimize it using gradient descent optimization. And in order to minimize the cost function we must adjust it's parameters which are weights and biases.

2.3.6 Cost function

The objective of any cost function is given a set of training inputs (features) and outcomes (the target we are trying to predict):

We want to find the set of weights and biases that minimize our cost function — where the cost function is an approximation of how wrong our predictions are relative to the target outcome. And in order to minimize the cost function we need gradient descent optimization algorithm. But before we use gradient descent we need to we need to know the gradient of our cost function, the vector that points in the direction of greatest steepness (we want to repeatedly take steps in the opposite direction of the gradient to eventually arrive at the minimum).

2.3.7 Gradient descent

The gradient of a function is the vector whose elements are its partial derivatives with respect to each parameter. For example, if we were trying to minimize a cost function, $C(B_0, B_1)$, with just two changeable parameters, B_0 and B_1 , the gradient would be:

Gradient of $C(B_0, B_1) = [[dC/dB_0], [dC/dB_1]]$

So each element of the gradient tells us how the cost function would change if we applied that small change to particular parameter. And these are gradient descent steps:

- 1- Calculate the gradient using our current parameter values.
- 2- Modify each parameter by an amount proportional to its gradient element and in the opposite direction of its gradient element. For example if the partial derivative of our cost function with respect to B_0 is positive but tiny and the partial derivative with respect to B_1 is negative and large, then we want to decrease B_0 by a tiny amount and increase B_1 by a large amount to lower our cost function.

3- Recompute the gradient using our new parameter values and repeat the previous steps until we arrive at the minimum.

2.3.8 Backpropagation

First we must mention forward propagation which is the process of moving forward through the neural network (from inputs to the output or prediction). Backpropagation is the reverse. Except instead of signal, we are moving error backwards through our model. While the objective of forward propagation is to calculate the activations at each neuron for each successive hidden layer until we arrive at the output, in contrast in back propagation we move this error backwards through our model via the same weights and connections that we use for forward propagating our signal. So the objective of Backpropagation is to calculate the error of each neuron starting from the layer closest to the output all the way back to the starting layer of our model. So The magnitude of the error of a specific neuron is directly proportional to the impact of that neuron's output on our cost function. So basically the role of backpropagation allows us to calculate the error attributable to each neuron and that in turn allows us to calculate the partial derivatives and ultimately the gradient so that we can utilize gradient descent.

2.3.9 Hyperparameter

A hyperparameter is a constant parameter whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyperparameters include

1-learning rate: determines the step size at each iteration while moving toward a minimum of a loss function

2-the number of hidden layers

Note :The values of some hyperparameters can be dependent on those of other hyperparameters. For example, the size of some layers can depend on the overall number of layers.

3-batch size: is a term used in machine learning and refers to the number of training examples utilized in one iteration. The batch size can be one of two options:

1. **batch mode**: where the batch size is equal to the total dataset thus making the iteration and epoch values equivalent

2. **mini-batch mode**: where the batch size is greater than one but less than the total dataset size. Usually, a number that can be divided into the total dataset size.

2.4 Learning paradigms

The three major learning paradigms are supervised learning, unsupervised learning and reinforcement learning. They each correspond to a particular learning task

2.4.1-Supervised learning

In supervised learning **labelled data** is used, which is a dataset where the target output is provided with each training example.

supervised learning includes two important techniques:

1- Regression used if the target output is a continuous value

2-Classification used if we want to separate data into two or more classes and it has two types: 1- binary classification 2- Multi classification

2.4.2-Unsupervised learning

In Unsupervised learning dataset is provided without labels and the task is to find similarities and dissimilarities between training examples in order to cluster them into related segments or detect anomalies

2.4.3-Reinforcement learning

In reinforced learning an indication whether the prediction matches the target or not is provided to the learning system. The learning system, called an *agent* in this context, can observe the environment, select and perform actions, and get *rewards* in return. It must then learn by itself what is the best strategy, called a *policy*, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

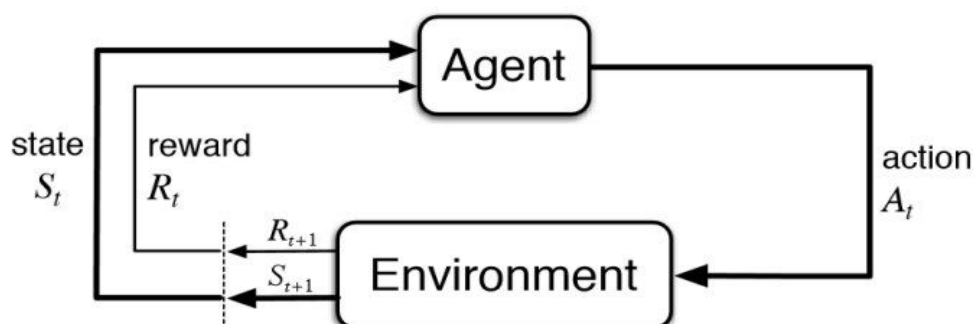


Figure 2.6

2.5 Important Activation Functions

2.5.1 ReLU (Rectified Linear Unit)

The rectified linear activation function is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

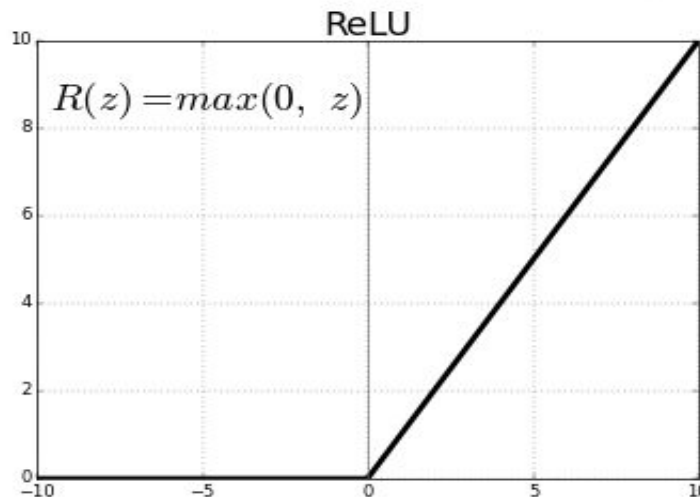


Figure 2.7

2.5.2 Soft-Max function

is a more generalized logistic activation function which is used for multiclass classification. The output of the soft-max function is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

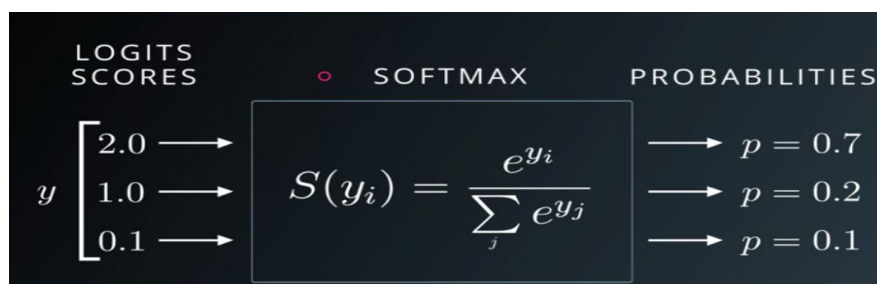


Figure 2.8

2.6 Most important types of neural networks

2.6.1 RNN(Recurrent Neural Networks)

Recurrent Neural Network(RNN) are a type of Neural Network where the output from previous step are fed as input to the current step and they are popular in NLP (Natural Language Processing) models .

The main and most important feature of RNN is **Hidden state**, which remembers some information about a sequence. Also RNN have a “memory” which remembers all information about what has been calculated.

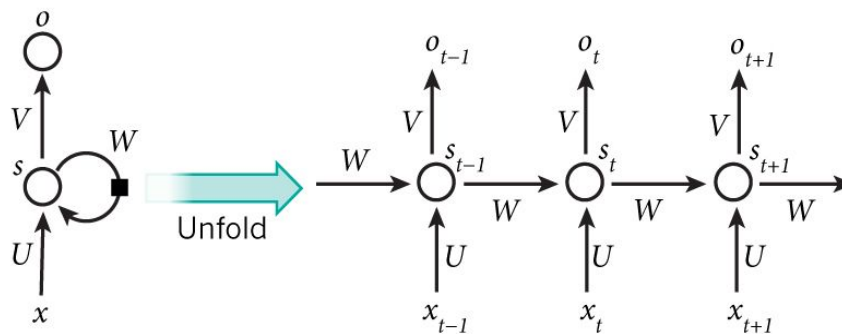


Figure 2.9

2.6.2 CNN(convolutional neural network)

convolutional neural network is a type of deep neural networks, most commonly applied to analysing visual imagery and they have applications in image and video recognition, recommender systems, image classification and self driving cars.

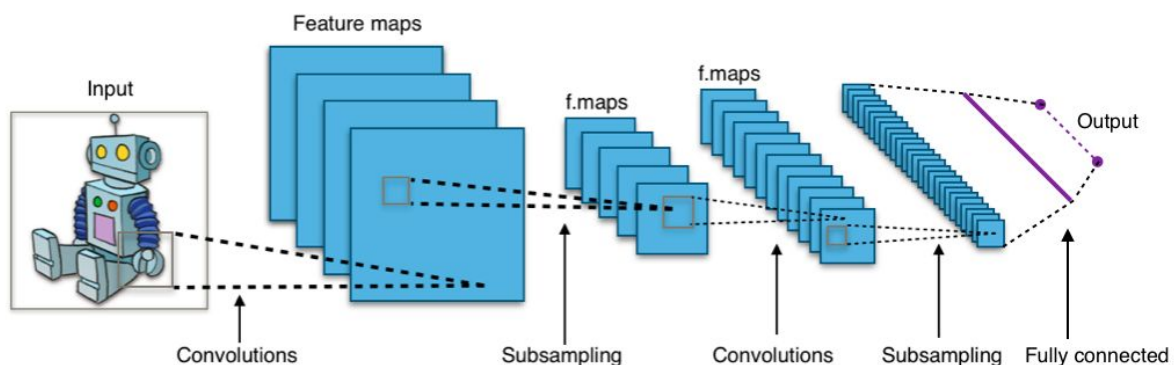


Figure 2.10

2.6.2.1 Architecture of convolutional neural network

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of

1. Convolution
2. Non Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

2.6.2.1.1 Convolution

ConvNets derive their name from the “convolution” operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image.

Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

every image can be considered as a matrix of pixel values. Consider a 5 x 5 image whose pixel values are only 0 and 1 (note that for a grayscale image, pixel values range from 0 to 255, the green matrix(matrix 1) below is a special case where pixel values are only 0 and 1) , Also, consider another 3 x 3 matrix(matrix 2) ,the Convolution of the 5 x 5 image and the 3 x 3 matrix can be result in 3x3 matrix (matrix 3)

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

*

1	0	1
0	1	0
1	0	1

=

4	3	4
2	4	3
2	3	4

Convolved
Feature

Figure 2.11

2.6.2.1.2 Non Linearity (ReLU)

An additional operation called ReLU has been used after every Convolution operation . ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by: ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet

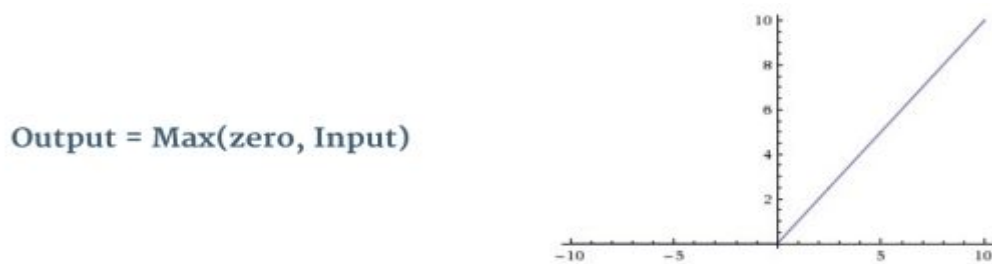


Figure 2.12

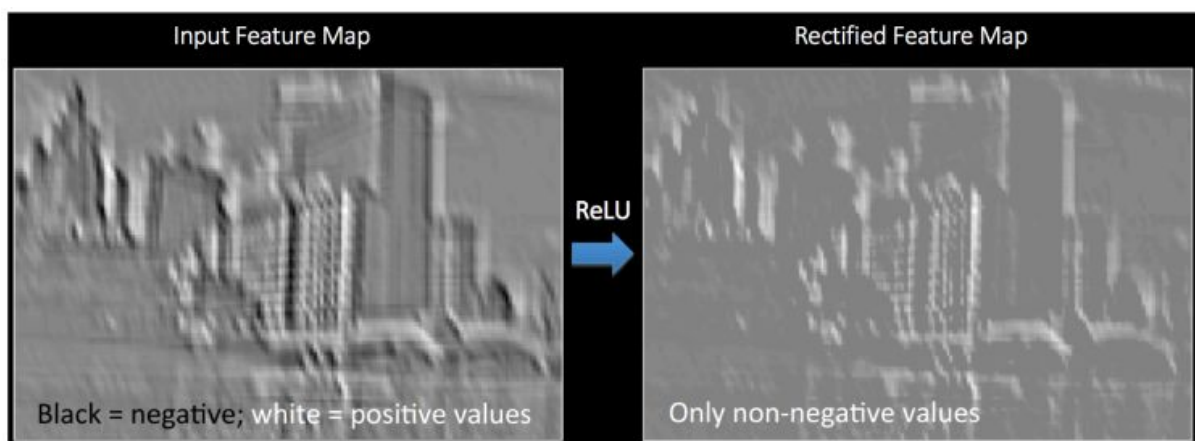


Figure 2.13

Other non linear functions such as **tanh** or **sigmoid** can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

2.6.2.1.3 Pooling or Sub Sampling

Pooling reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: **Max**, **Average**, **Sum**.

in most case use of Max Pooling, we define a spatial neighbourhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

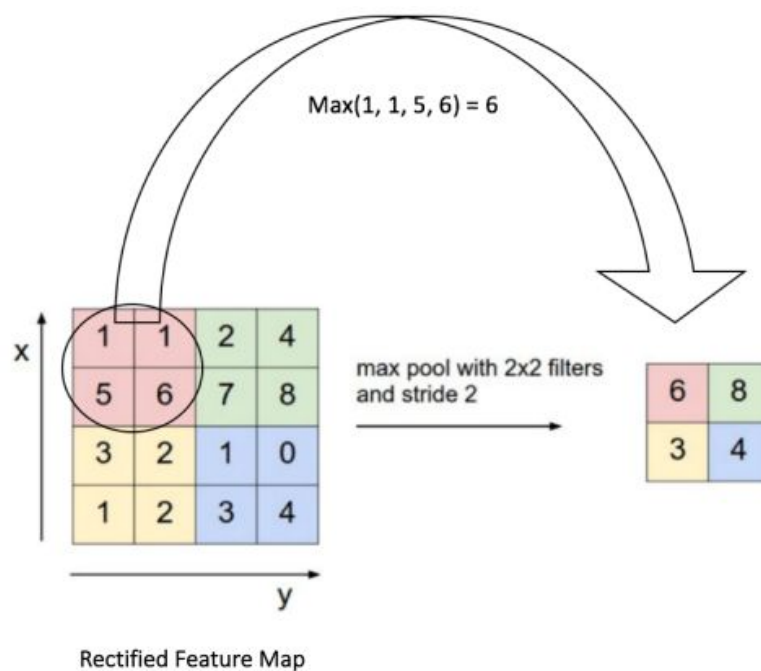


Figure 2.14

2.6.2.1.4 Classification (Fully Connected Layer)

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer

The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. For example, the image classification task we set out to perform has four possible outputs as shown in Figure below.

The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer.

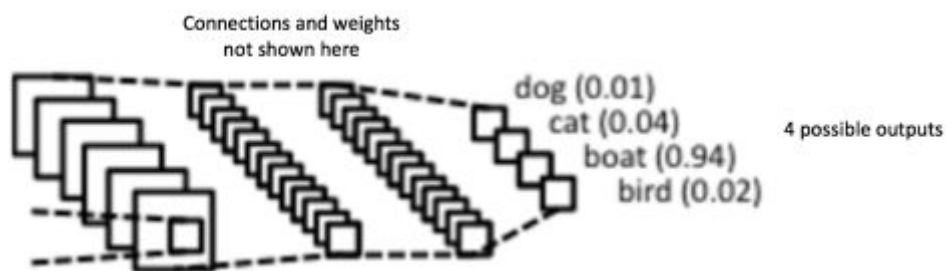


Figure 2.15

2.7 Concepts and techniques used in our model

2.7.1 Residual modules

In order to solve the problem of the vanishing/exploding gradient, this architecture introduced the concept called Residual Network. In this network we use a technique called *skip connections*. The skip connection skips training from a few layers and connects directly to the output.

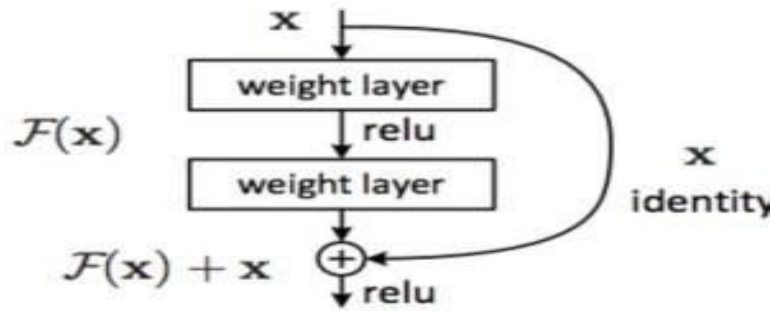


Figure 2.16

The advantage of adding this type of skip connection is because if any layer hurt the performance of architecture then it will be skipped.

2.7.2 Depth-wise separable convolutions

One class of CNN's are depth wise separable convolutional neural networks. These type of CNN's are widely used because of the following two reasons :

- They have lesser number of parameters to adjust as compared to the standard CNN's, which reduces overfitting
- They are computationally cheaper because of fewer computations which makes them suitable for mobile vision applications

Depth-wise separable convolutions has two main operations :

2.7.2.1-Depth-wise operation

In depth-wise operation, convolution is applied to a single channel at a time unlike standard CNN's in which it is done for all the M channels. So here the filters/kernels will be of size $D_k \times D_k \times 1$. Given there are M channels in the input data, then M such filters are required. Output will be of size $D_p \times D_p \times M$.

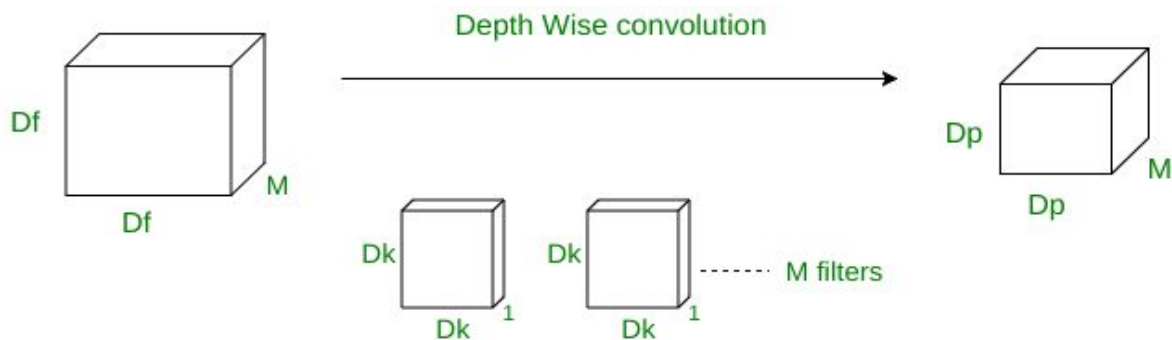


Figure 2.17

2.7.2.2-Point-wise operation

In point-wise operation, a 1×1 convolution operation is applied on the M channels. So the filter size for this operation will be $1 \times 1 \times M$. Say we use N such filters, the output size becomes $D_p \times D_p \times N$.

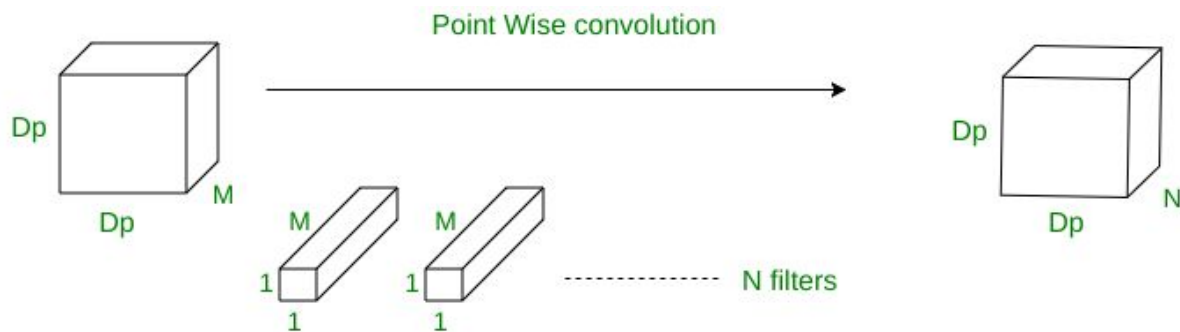


Figure 2.18

2.7.3 Global average pooling

Global Average Pooling reduces each feature map into a scalar value by taking the average over all elements in the feature map like the figure bellow. The average operation forces the network to extract global features from the input image.

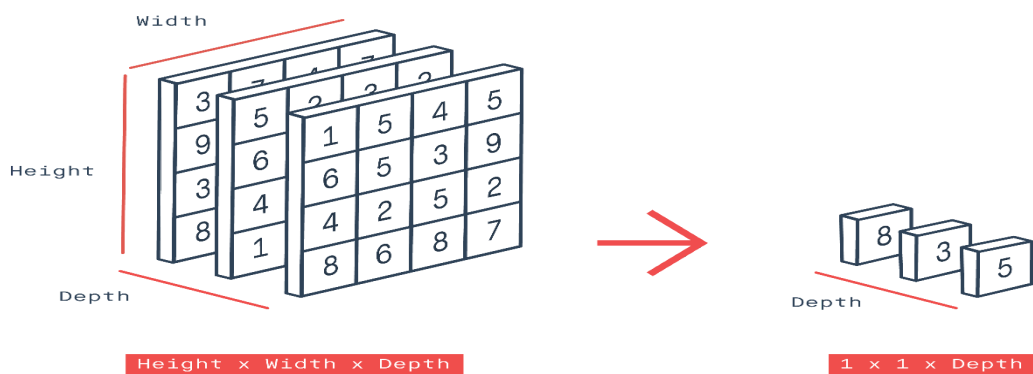


Figure 2.19

They're often used to replace the full-connected layers in a classifier. Instead, the model ends with a convolutional layer that generates as many feature maps as the number of target classes, and applies global average pooling to each in order to convert each feature map into one value. As feature maps can recognize certain elements within the input data, the maps in the final layer effectively learn to "recognize" the presence of a particular class in this architecture. By feeding the values generated by global average pooling into a Softmax activation function, you once again obtain the multiclass probability distribution that you want.

2.7.4 Batch normalization

Batch normalization is a technique to address the vanishing/exploding gradients problems. The technique consists of adding an operation in the model just before or after the activation function of each hidden layer, simply zero-centering and normalizing each input, then scaling and shifting the result using two new parameter vectors per layer: one for scaling, the other for shifting.

In order to zero-center and normalize the inputs, the algorithm needs to estimate each input's mean and standard deviation. It does so by evaluating the mean and standard deviation of each input over the current mini-batch.

2.7.5 Adam optimizer

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

Adam optimizer combines the benefits of :

Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing).

Instead of adapting the parameter learning rates based on the mean as in RMSProp, Adam also makes use of the uncentered variance. And it has a lot of advantages:

- Straightforward to implement.
- Computationally efficient.
- Little memory requirements.
- Well suited for problems that are large in terms of data and/or parameters.
- Appropriate for problems with very noisy/or sparse gradients.

chapter 3

(Related work)

Many current research in automatic facial expression analysis based on the following three approaches :

1- **Categorical model**: appoint discrete emotions class labels.



Figure 3.1

2- **FACS model**: detecting the existence / absence of different action units defined by Facial Action Coding System.

Facial Action Coding System : refers to a collection of facial muscle movements which corresponds to the emotion displayed.

For example:

Happiness: cheek raise and lip corner pull

Disgust: nose wrinkle, lip corner depress, lower lip depress

Sadness: inner brow raise, brow lower, lip corner depress

Surprise: inner brow raise, outer brow raise, upper lid raise, jaw drop

Anger: brow lower, upper lid raise, lid tightener, lip tightener









AU 1	AU 2	AU 4	AU 5
			
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser
*AU 41	*AU 42	*AU 43	AU 44
			
Lid Droop	Slit	Eyes Closed	Squint

Figure 3.2



Figure 3.3

3- Dimensional model : describing emotions using dimensional models like valence-arousal .

In valence-arousal dimensional model emotions can be described by two factors:

Valence: positive or negative affectivity

Arousal: how calming or exciting

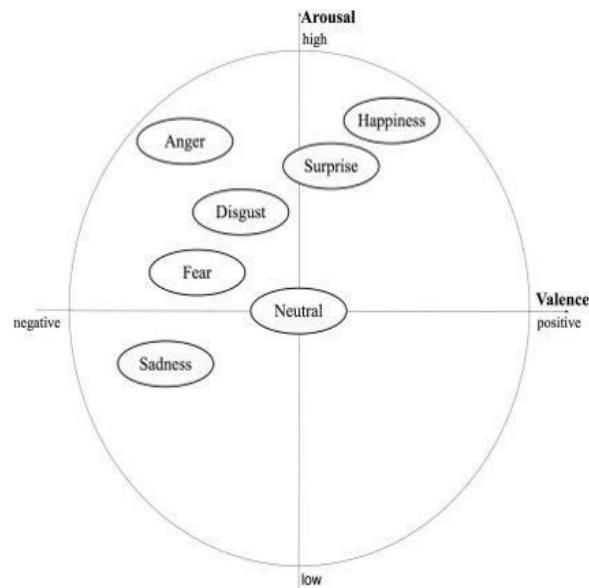


Figure 3.4

FEC-16D model : fully-supervised network for learning 16-dimensional embedding based on FEC dataset

Expression datasets:

In the past, several datasets of facial expression have been created , consisting of face images labeled with discrete emotion classes , facial action units and strength of valence and arousal. Although these datasets played a considerable role in improvement of automatic facial expression analysis , they are not the perfect result for learning a compact expression embedding space that simulates human visual preferences.

Expression Embedding:

neural networks trained using label-based triplets may not produce a detailed expression representation because emotions labels don't provide information about with-in class variations.

In the other hand we used a fully-supervised network for learning 16-dimensional embedding based on FEC dataset that solved this problem by including expression comparison annotations for within-class triplets.

Triplet loss-based representation learning:

Majority of current works category label-based triplets but FEC dataset use human raters to annotate the triplet.

Facial expression comparison dataset:

FEC is a large scale dataset with expression comparisons annotations introduced by human raters.

Each example in the FEC dataset include a face image triplet (I_1, I_2, I_3) with a label $L \in \{1,2,3\}$

That shows which pair of images are most similar in the triplet. For example , $L=1$ means that the second and the third images int the triplet are more visually similar when compared to the first one and so on . In contrast to commonly-used triplet annotations that consist of an anchor , a positive image and a negative image FEC triplets provides two annotations : I_2 is closer to I_3 than I_1 and I_3 is closer to I_2 than I_1 . Also in this data set an image A might be relatively closer to another image B in one triplet and relatively farther from the same image in a another triplet .

Six human raters annotated each triplet and instructed to concentrate on expression only and to ignore other factors like identity, gender , ethnicity, pose and age. A total of 40 raters took part in this process. Each annotated a subset of the entire dataset. Any triplet in this dataset can be classified into one of the following :

- One-class triplets: All the three images share the same class label, These triplets are useful for learning a detailed expression representation.

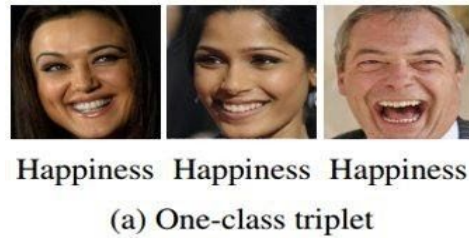


Figure 3.5

- Two-class triplets: Only two images share a class label and the third image belongs to a different category, and images sharing the same label need not form the (visually) most similar pair in these triplets.

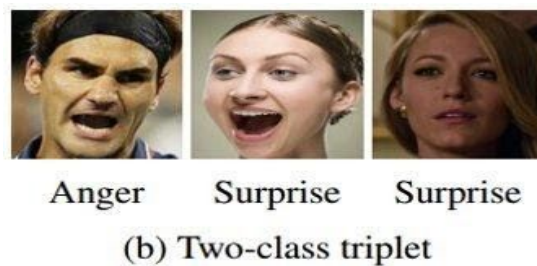


Figure 3.6

- Three-class triplets: None of the images share a common category label, These triplets are useful for learning long-range visual similarity relationships between different categories.



Figure 3.7

chapter 4

(Proposed Model)

4.1-Dataset

This data consists of 35,887 rows and contains 3 columns

1	emotion	pixels	Usage
2	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 91 84 84 90 99 110 126 143 153 158 171 169 172 169 165 129 110 11	Training
3	0	151 150 147 155 148 133 111 140 170 174 182 154 153 164 173 178 185 185 189 187 186 193 194 185 183 186 180 173 166 161 147	Training
4	2	231 212 156 164 174 138 161 173 182 200 106 38 39 74 138 161 164 179 190 201 210 216 220 224 222 218 216 213 217 220 220 218	Training
5	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 19 43 52 13 26 40 59 65 12 20 63 99 98 98 111 75 62 41 73 118 140 192 186 187 188 19	Training
6	6	4 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84 115 127 137 142 151 156 155 149 153 152 157 160 162 159 145 121 83 58 48 38 21 17 7	Training
7	2	55 55 55 55 55 54 60 68 54 85 151 163 170 179 181 185 188 188 191 196 189 194 198 197 195 194 190 193 195 184 175 172 161 159	Training
8	4	20 17 19 21 25 38 42 42 46 54 56 62 63 66 82 108 118 130 139 134 132 126 113 97 126 148 157 161 155 154 154 164 189 204 194 16	Training
9	3	77 78 79 79 78 75 60 55 47 48 58 73 77 79 57 50 37 44 56 70 80 82 87 91 86 80 73 66 54 57 68 69 68 68 49 46 75 71 69 70 70 72 72 71	Training
10	3	85 84 90 121 101 102 133 153 153 169 177 189 195 199 205 207 209 216 221 225 221 220 218 222 223 217 220 217 211 196 188 173	Training

Figure 4.1=====

4.1.1-Emotion label:

It contains 7 values from 0 to 6, each number from 0 to 6 represents a specific expression

- 0 ☐ Angry
- 1 ☐ Disgust
- 2 ☐ Scared
- 3 ☐ Happy
- 4 ☐ Sad
- 5 ☐ Surprised
- 6 ☐ Neutral

4.1.2-Image matrix: all images are grayscale images and their size is 48 X 48 pixels so this column contain 2D matrix whose size is 48 X 48 values between 0 ☐ 255 which express color intensity 0 for black and 255 for white and numbers in between is grayscale

4.1.3-Usage: Contains a value of two {Training, Testing} that specify whether this image is used for training or testing.

4.2-Model

The core of Deep learning project is a model, model divided into two types:

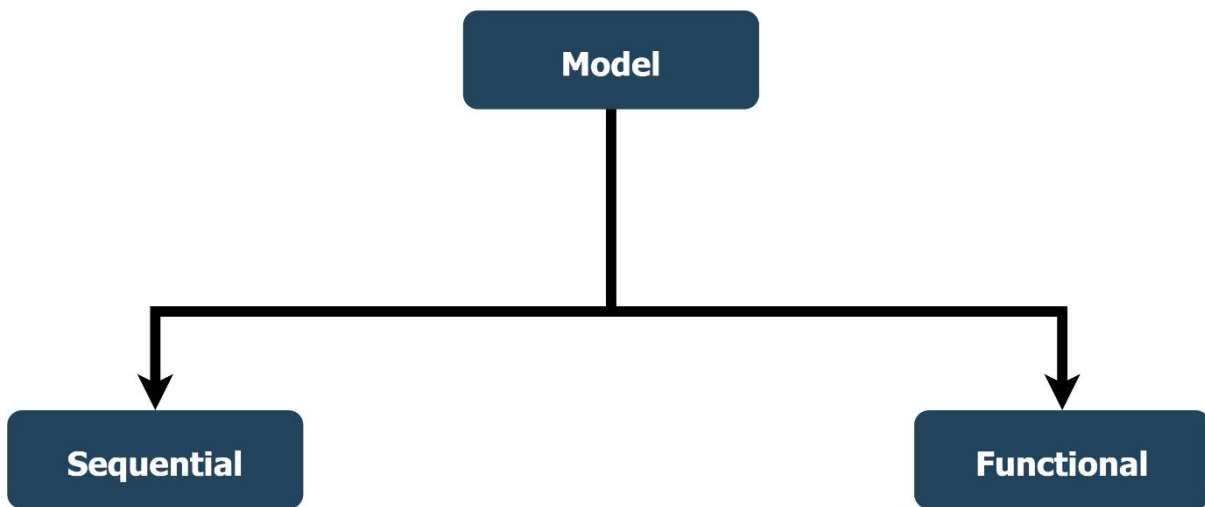


Figure 4.2

4.2.1 Sequential:

The Sequential model API is great for developing deep learning models in most situations, but it also has some limitations. For example, it is not straightforward to define models that may have multiple different input sources, produce multiple output destinations or models that re-use layers.

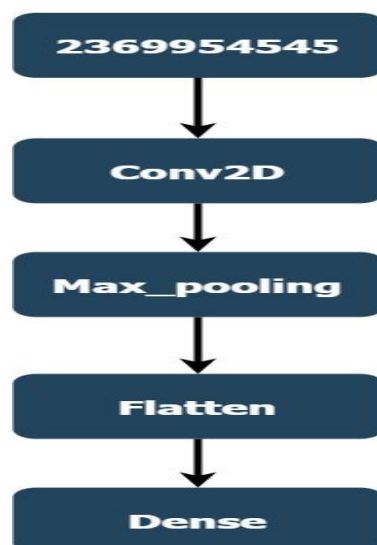


Figure 4.3

4.2.2 Functional:

It specifically allows you to define multiple input or output models as well as models that share layers. More than that, it allows you to define ad hoc acyclic network graphs.

Models are defined by creating instances of layers and connecting them directly to each other in pairs, then defining a Model that specifies the layers to act as the input and output to the model.

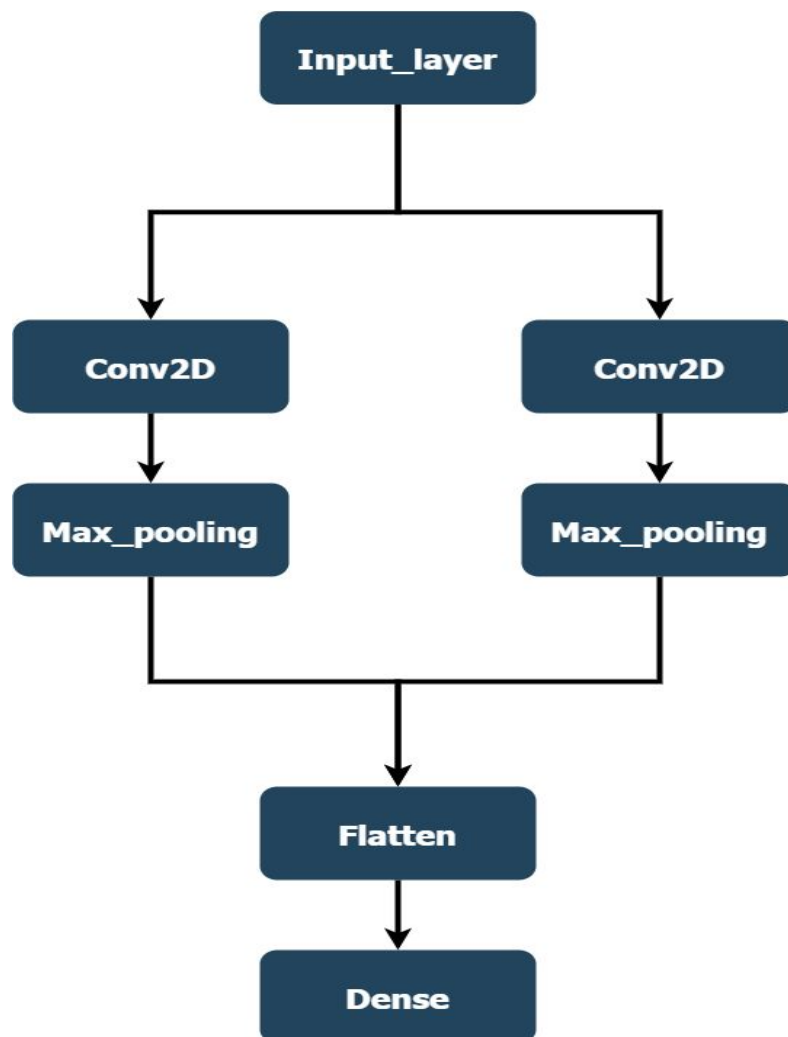


Figure 4.4

4.3 Model Architecture

Let's explain project model architecture:

In this model, we will use **functional** model because we use residual technique and this means we will skip connection, and we know cannot skip connection in a **sequential** model.

- The architecture is a CNN model combines the **deletion of the fully connected** Layer and the inclusion of the combined:
 - 1- depth-wise separable convolutions
 - 2-residual modules.
- Architecture used Global Average Pooling to completely remove any fully Connected layers.
- Model used Categorical cross – entropy loss function
- Model was trained with the ADAM optimizer

Reason for **deletion of fully-connected layers**:

- We aim to use this model in a real time system so we want to reduce computations as much as possible and this conflicts with the tons of parameters in the fully connected layers
- For example VGG16 contains approximately 90% of all its parameters in their last fully connected layers
- This architecture has approximately 60,000 parameters; which corresponds to a reduction of 80×when compared other original CNN

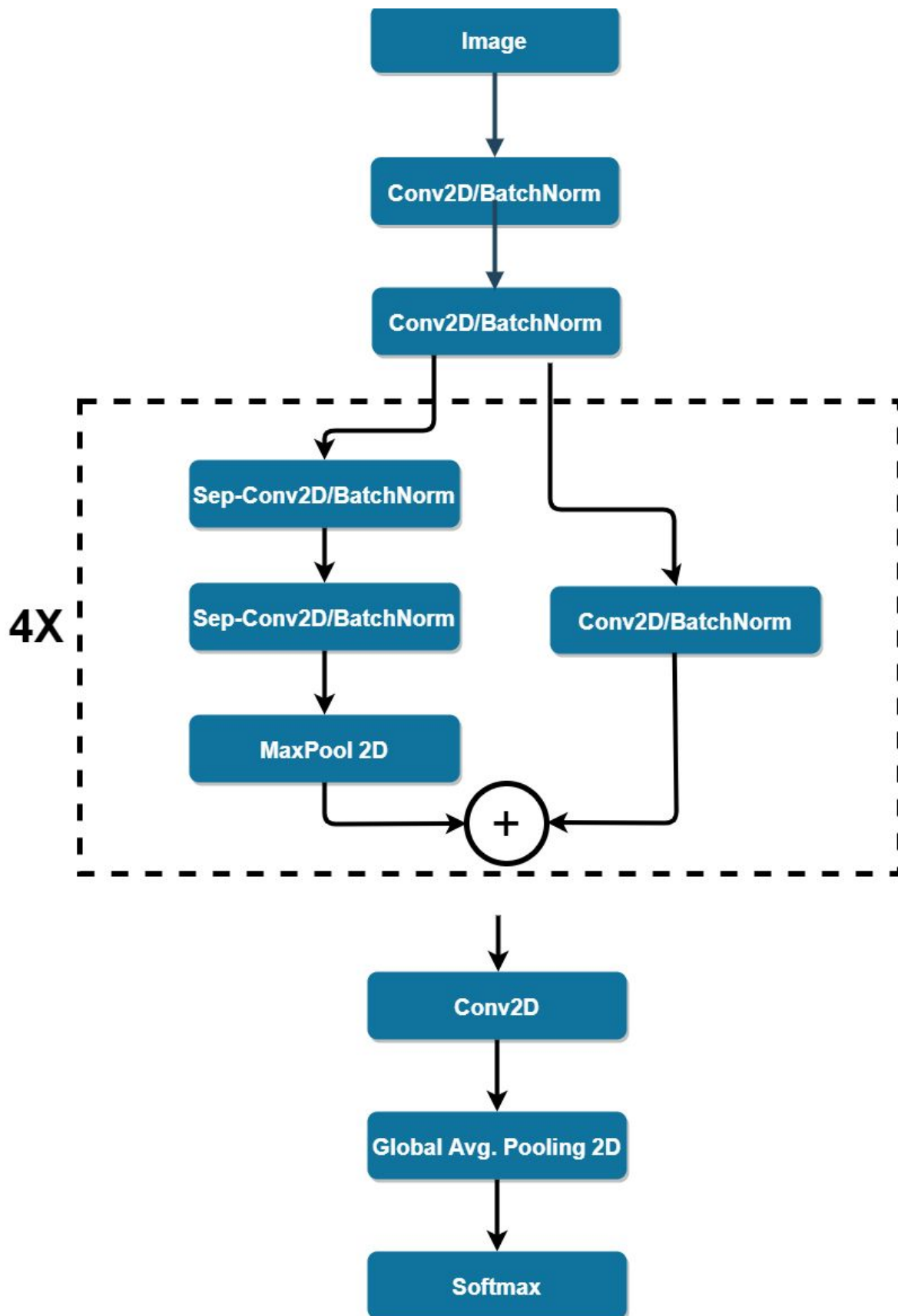


Figure 4.5 :Model Network Architecture

4.3.1 Residual Block:

In traditional neural networks, each layer feeds into the next layer. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away. ResNet, which was proposed in 2015 by researchers at Microsoft Research introduced a new architecture called Residual Network.

In order to solve the problem of the vanishing/exploding gradient, this architecture introduced the concept called Residual Network. In this network we use a technique called *skip connections*. The skip connection skips training from a few layers and connects directly to the output.

The approach behind this network is instead of layers learn the underlying mapping, we allow network fit the residual mapping. So, instead of say $H(x)$, initial mapping, let the network fit, $F(x) := H(x) - x$ which gives $H(x) := F(x) + x$.

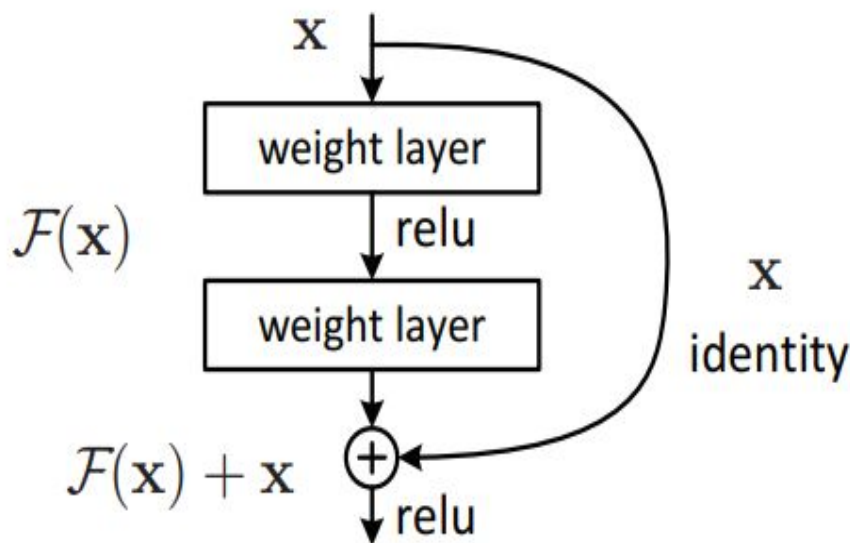


Figure 4.6

-The advantage of adding this type of skip connection is because if any layer hurt the performance of architecture then it will be skipped by regularization. So, this results in training very deep neural network without the problems caused by vanishing/exploding gradient.

4.3.2 Depth-wise separable convolutions neural networks:

These type of CNN's are widely used because of the following two reasons –

- They have lesser number of parameters to adjust as compared to the standard CNN's, which **reduces overfitting**
- They are **computationally cheaper** because of fewer computations which makes them suitable for mobile vision applications

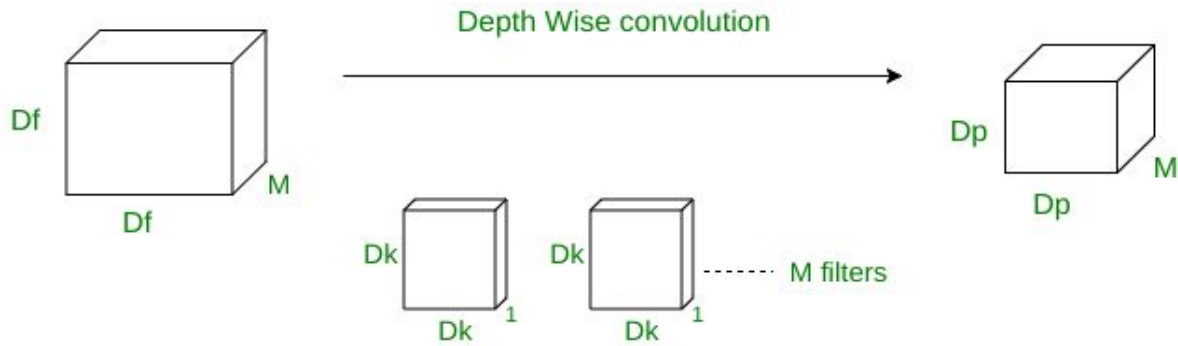


Figure 4.7

- Difference between (a) standard convolutions and (b) depth-wise separable convolutions.

Normal Convolution operation:

Suppose there is an input data of size $D_f \times D_f \times M$, where $D_f \times D_f$ can be the image size and M is the number of channels (3 for an RGB image). Suppose there are N filters/kernels of size $D_k \times D_k \times M$. If a normal convolution operation is done, then, the output size will be $D_p \times D_p \times N$.

Depth-Wise Separable Convolutions:

In *depth-wise operation*, convolution is applied to a **single channel** at a time unlike standard CNN's in which it is done for all the M channels. So here the filters/kernels will be of size $D_k \times D_k \times 1$. Given there are M channels in the input data, then M such filters are required. Output will be of size $D_p \times D_p \times M$.

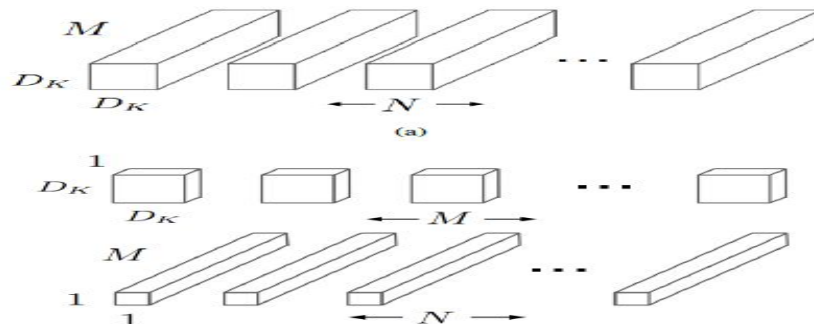


Figure 4.8

Difference between Fully Connected layers and Global average pooling.

- A solution to reduce the number of parameters is to use global average pooling.

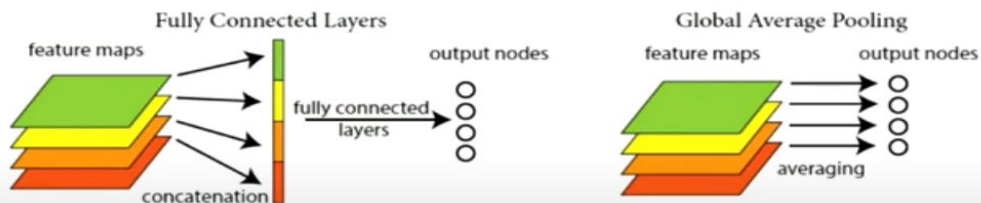


Figure: Global average pooling [3]

Figure 4.9

chapter 5

(Results & Discussion)

5.1 Results

we turned to the second technique which has smaller dataset that contains almost 36K images and the images matrix given simply in the dataset, Also the model architecture is not as complex as the first one. Therefore, we succeeded to train the model and we had fairly good results although it's not the desired results but this is due to the small size of the dataset. **We achieved 70% accuracy on the test images which were 7178 images(20%)**

```
In [4]: 1,acc=model.evaluate(xtest,ytest)
7178/7178 [=====] - 23s 3ms/stepETA: 4s -
ETA: 0s

In [5]: print("Accuracy= ",round(acc*100),"%")
Accuracy= 70 %
```

Figure 5.1

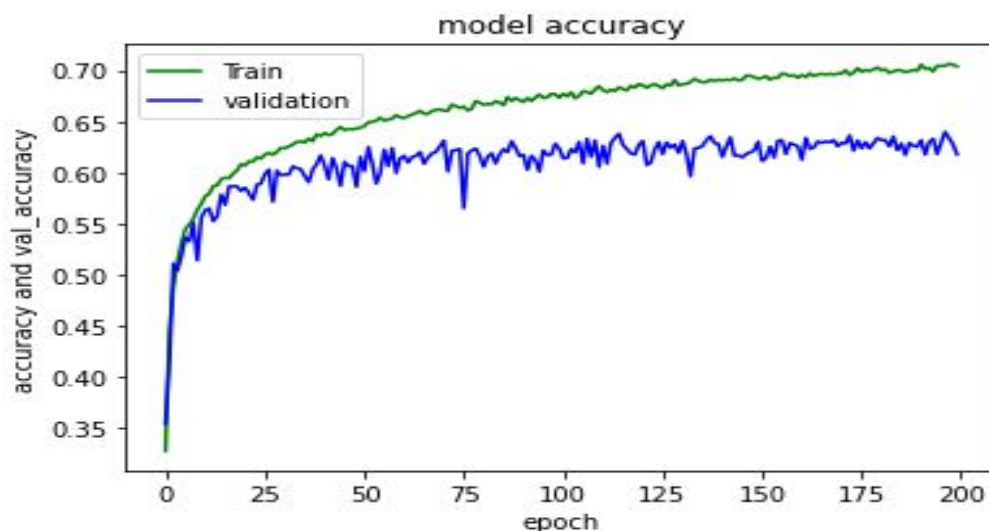


Figure 5.2

Accuracy =70%

As you can see in this plot there is a direct proportion between accuracy and the number of epochs

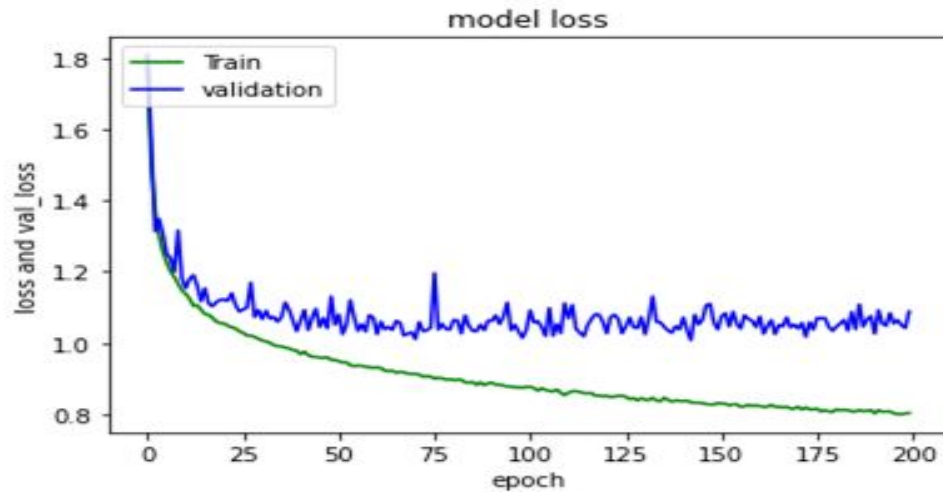


Figure 5.3

-Loss function

As you can see in this plot there is a reversed proportion between loss function and the number of epochs

Apply our model to streaming video:

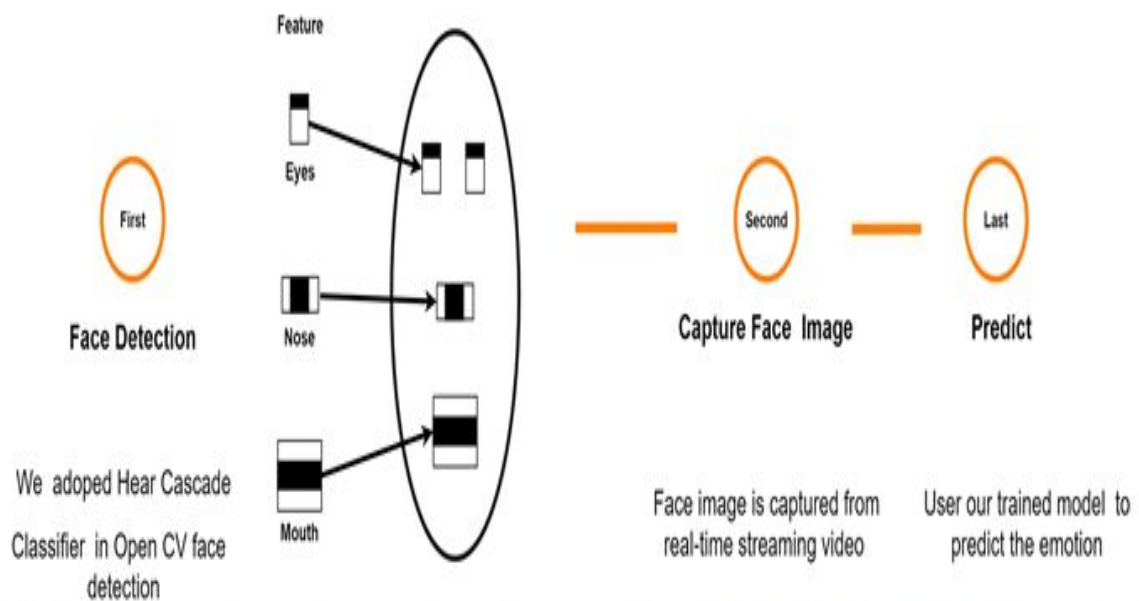


Figure 5.4

5.2 GUI

we have a simple interface for our project which has two options whether to select an image from your computer to predict the emotion of the face in the photo or to test the model live on your face after opening your camera and perform various emotion



Figure 5.5

Testing the model on an image

we tested the model on various faces with different emotions to make sure it works well and these are some results

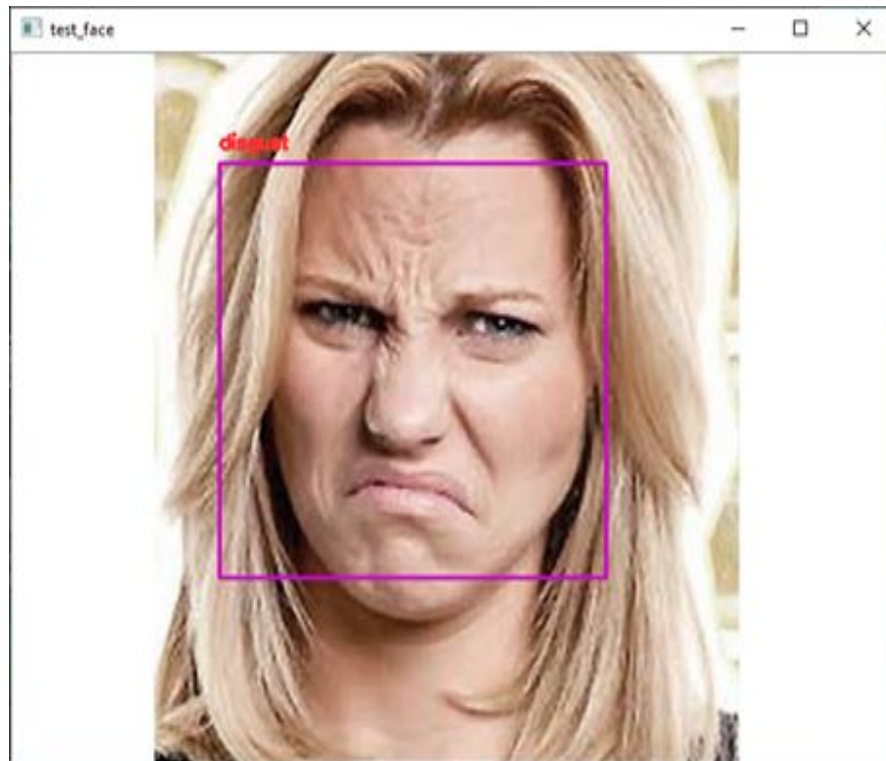


Figure 5.6



Figure 5.7

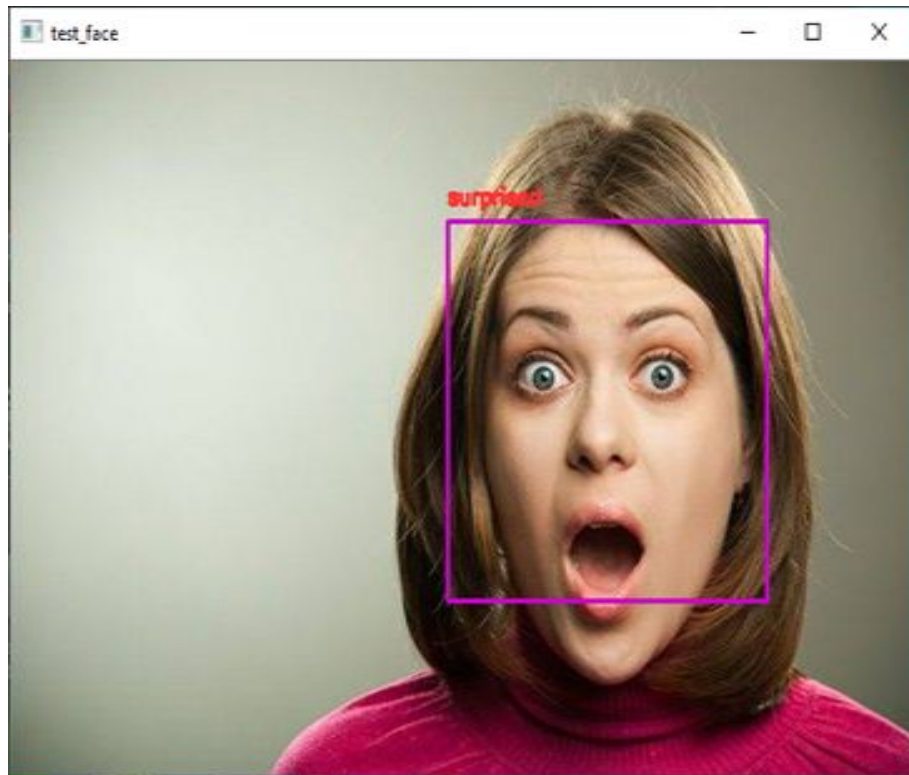


Figure 5.8

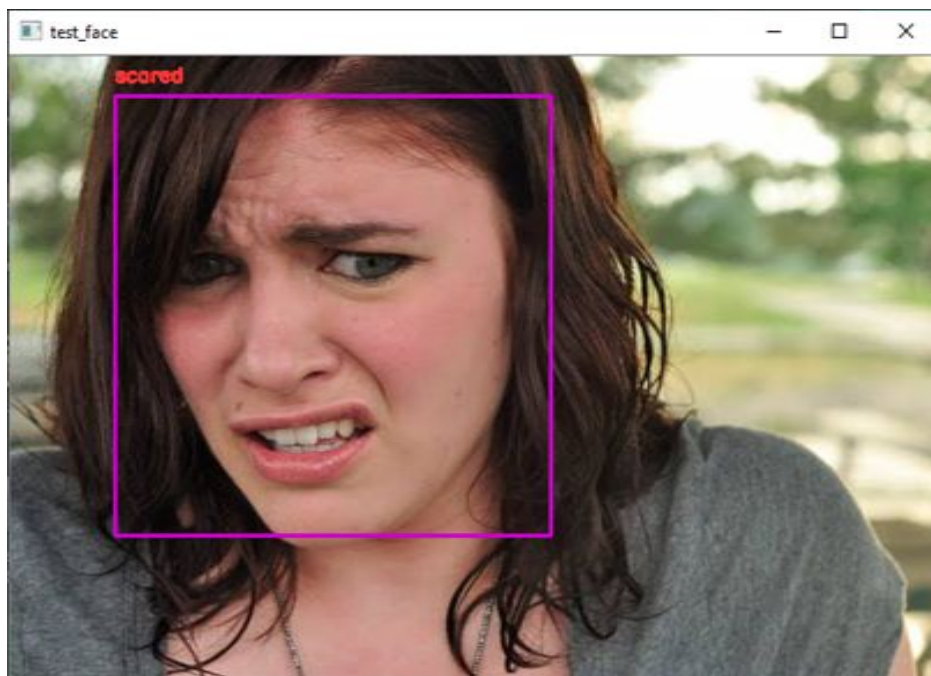


Figure 5.9



Figure 5.10

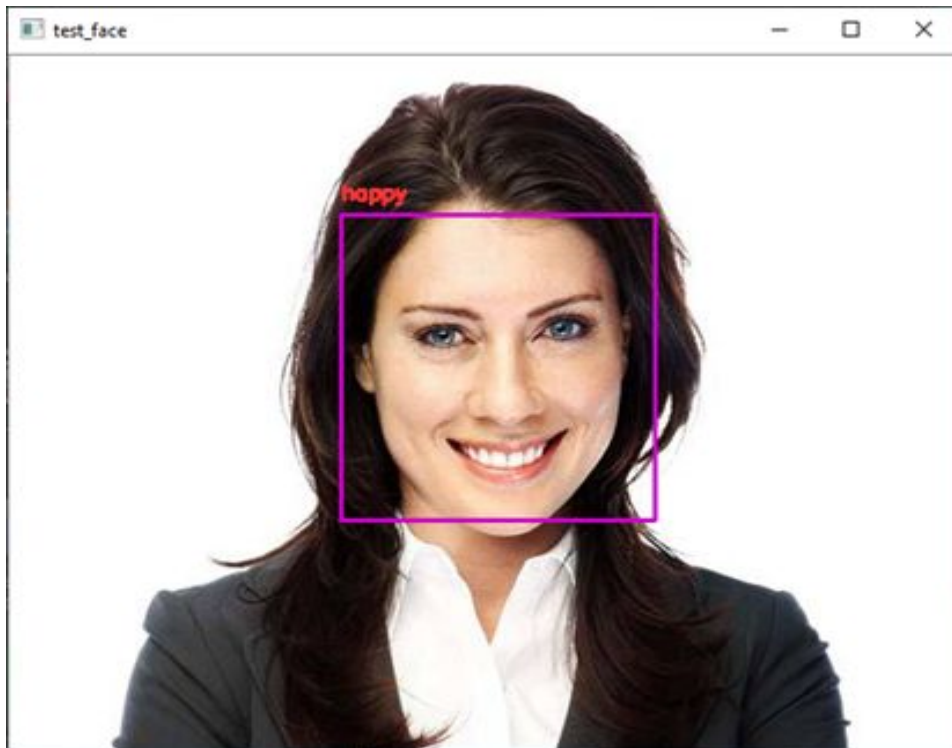


Figure 5.11

Live testing

opening the camera and try to perform different emotions and test the model performance on different measures like: head pose, room lightening and age in addition to the emotion itself and these are some of the **results:**

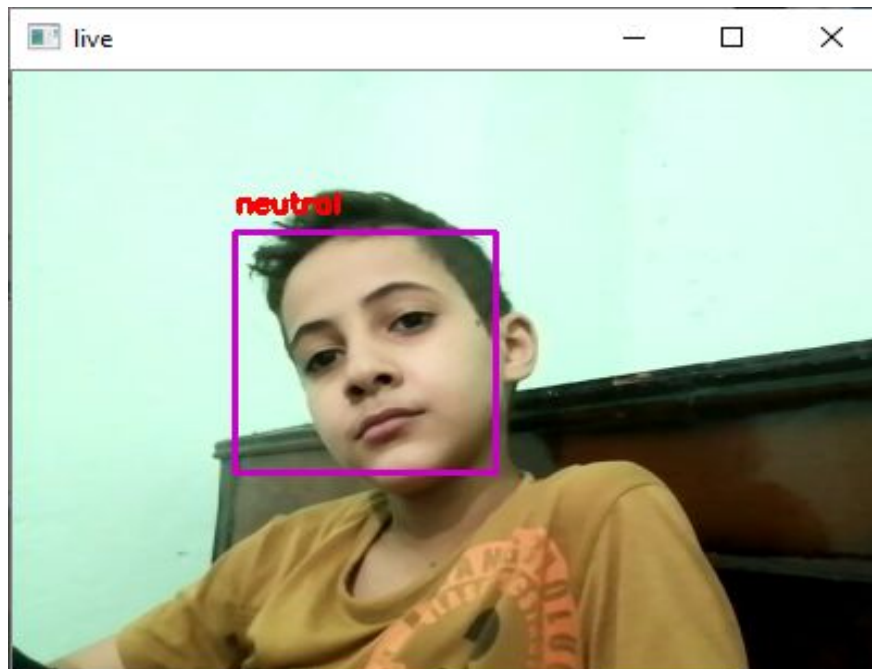


Figure 5.12

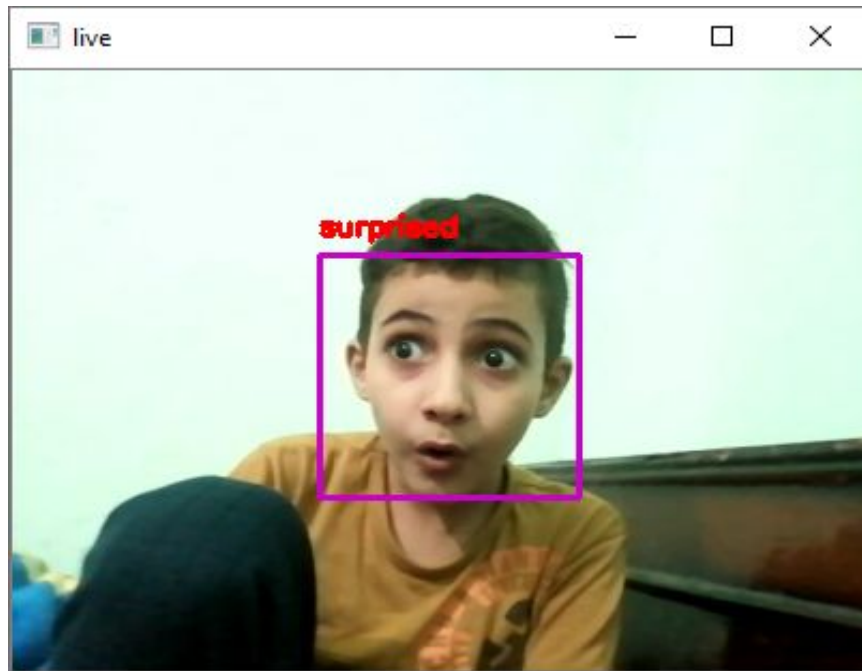


Figure 5.13

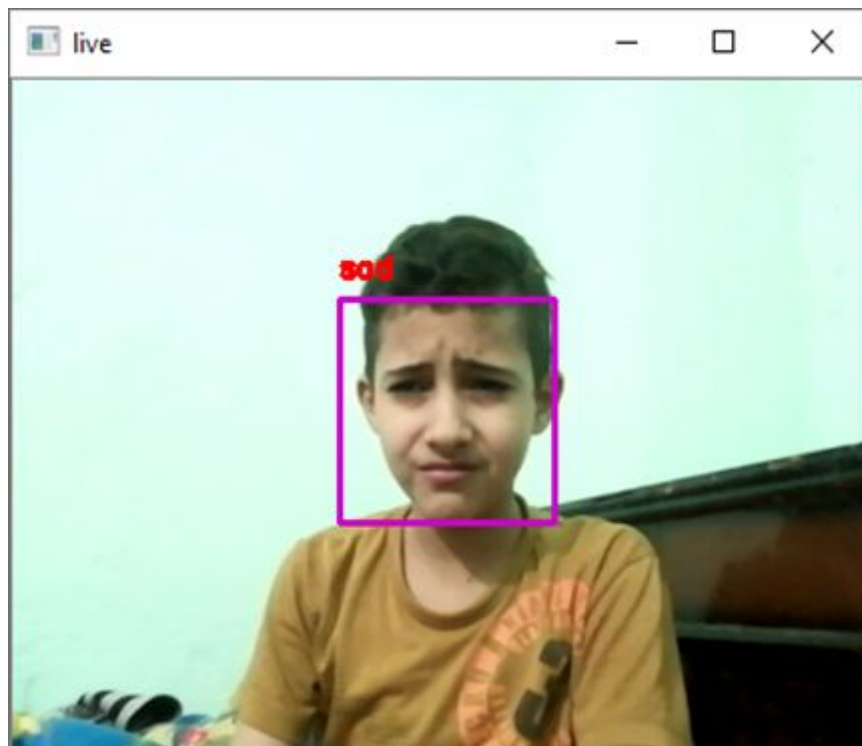


Figure 5.14

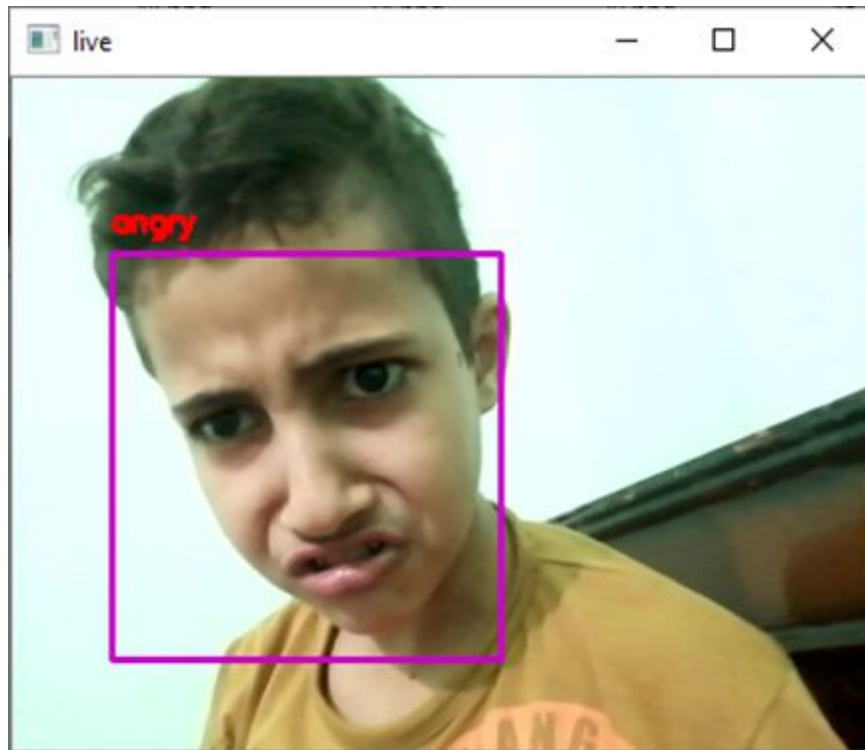


Figure 5.15

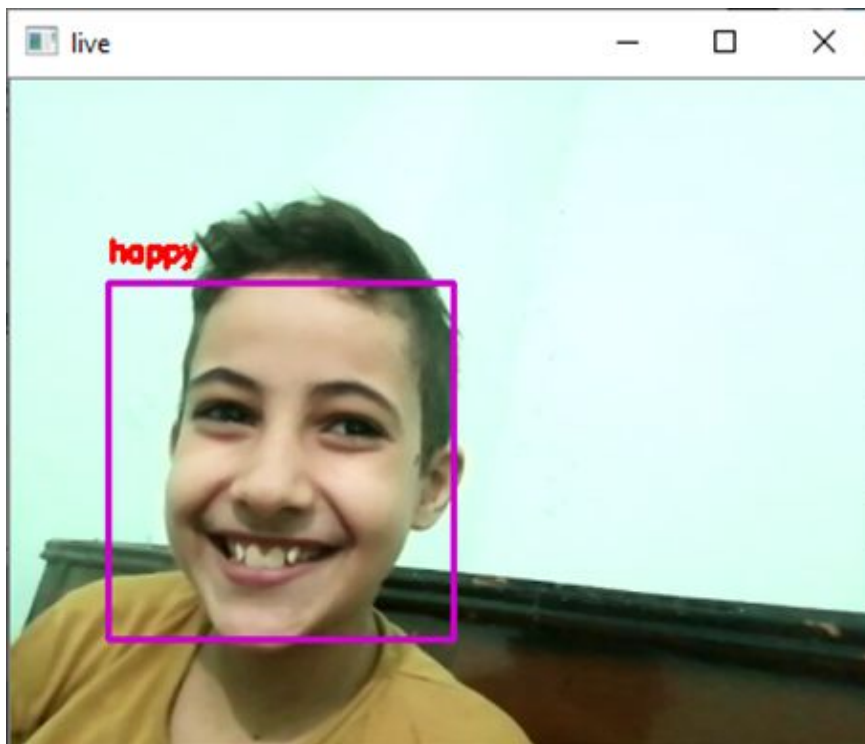


Figure 5.16

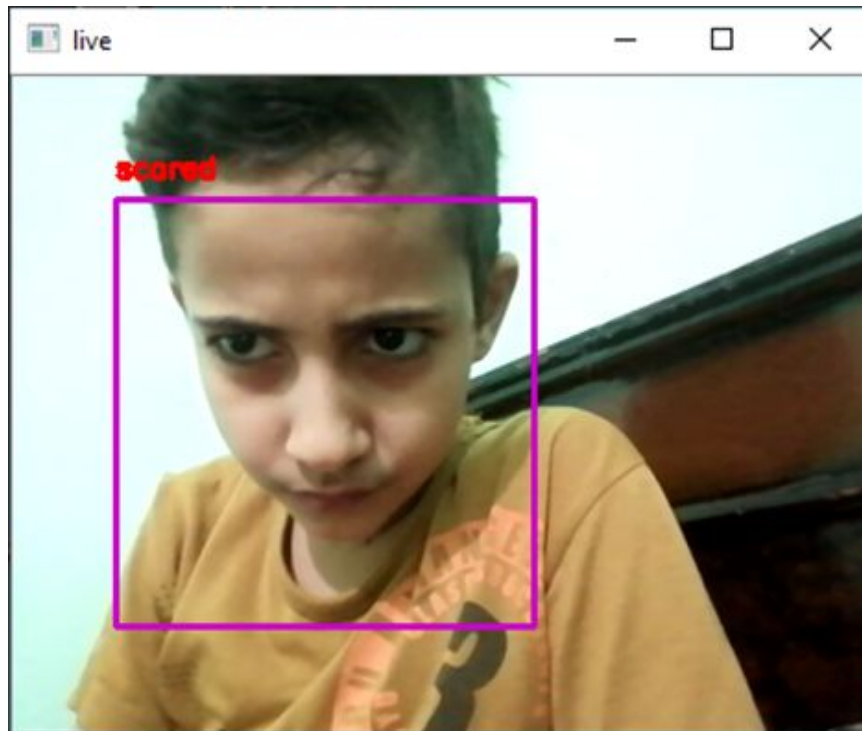


Figure 5.17

5.3 Future work

As future work, we plan to improve your results analysing through train the CNNs on more than one data set to avoid classification interference and Increases the Accuracy. Furthermore, we present a glass type wearable device to detect emotions from a human face response from the user face naturally and continuously while the user is wearing it. take responses from face through sensors the device can capture the image region representing facial expressions

5.4 CONCLUSIONS

We have proposed and tested a general building designs for creating real-time CNNs. Our proposed architectures have been systematically built in order to reduce the number of parameters. We began by eliminating completely the fully connected layers and by reducing the number of parameters in the remaining convolutional layers via depth-wise separable convolutions. We have shown that our proposed models can be stacked for multi-class classifications while maintaining real-time inferences. Specifically, we have developed a vision system that performs face detection, and emotion classification in a single integrated module. We have achieved human-level performance in our classification's tasks using a single CNN that leverages modern architecture constructs .and In the carried-out experiments, for 7 emotional states, we achieved a very good classification

5.5 References

-Real-time Convolutional Neural Networks for Emotion and Gender Classification

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

-Under standing neural network

<https://towardsdatascience.com/understanding-neural-networks-19020b758230>

- Batch normalization:

<https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/>

ReLU and Softmax Activation Functions:

<https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions>

Adam optimizer:

<https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

pooling types:

https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/#global-average-pooling_1

RNN:

<https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>

residual modules:

<https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/>

Approach	research
Categorical model	Real-time Convolutional Neural Networks for Emotion and Gender Classification.Octavio Arriaga, Matias Valdenegro-Toro, Paul Plöger,Oct 2017
FACS model	P. Ekman, W. V. Friesen, and J. C. Hager. Facial Action Coding System - Manual. A Human Face, 2002
Dimensional model	J. A. Russell. A circumplex model of affect. <i>Journal of Personality and Social Psychology</i> , 39(6):1161–1178, 1980.
FEC-16d model	A Compact Embedding for Facial Expression Similarity,Raviteja Vemulapalli, Aseem Agarwala,Nov 2018

-Artificial neural network

https://en.wikipedia.org/wiki/Artificial_neural_network

-Activation Functions

<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

-Data compression

https://en.wikipedia.org/wiki/Data_compression

-Reinforcement learning

https://en.wikipedia.org/wiki/Reinforcement_learning

- Forward propagation

<https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250>

-cnn

<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

- types of Artificial Neural Networks

<https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2>

-Going Deeper with Convolutions

<https://arxiv.org/abs/1409.4842>

-FaceNet: A Unified Embedding for Face Recognition and Clustering

<https://arxiv.org/abs/1503.03832>

-A Compact Embedding for Facial Expression Similarity

<https://arxiv.org/abs/1811.11283>

-Densely Connected Convolutional Networks

<https://arxiv.org/abs/1608.06993>

- Dataset to mode 1(FEC)

<https://ai.google/tools/datasets/google-facial-expression/>

13- Dataset to mode 2

https://www.kaggle.com/deadskull7/fer2013?fbclid=IwAR1EnKBqJdOHvm_japwctOm1gaTzz5Qew9iVHFGGrSJYwhXNUPjL6AL4HI5s