DEEP LEARNING BASED FACIAL EXPRESSION RECOGNITION USING FEATURES FUSION

By: Tadele Shimelis Dejene

A Thesis Submitted to the Department of Computing

School of Electrical Engineering and Computing.

Presented in Partial Fulfilment for the Degree of Masters of Science in

Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

Adama,
Ethiopia,

September 23, 2019

TITLE: DEEP LEARNING BASED FACIAL EXPRESSION RECOGNITION USING FEATURES FUSION

By

Tadele Shimelis Dejene

Advisor: Prof. Yun Koo Chung

A Thesis Submitted to the Department of Computing

School of Electrical Engineering and Computing.

Presented in Partial Fulfillment for the Degree of Masters of Science in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

Adama,

Ethiopia

September 23, 2019

DECLARATION

I hereby declare that this MSc thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis have been duly acknowledged

Name: Tadele Shimelis

Signature: _____

This MSc thesis has been submitted for examination with my approval as a university advisor.

Yun Koo Chung (PHD) _____

(Advisor) (Signature)

Submission Date: _____

APPROVAL OF THE BOARD OF EXAMINERS

We, the undersigned, members of the Board of Examiners of the final open defence by Tadele Shimelis Dejene have read and evaluated his thesis entitled "Deep Learning-Based Facial Expression Recognition using Feature Fusion" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfilment of the requirement of the degree of Masters in Computer Science and Engineering (CSE).

Yun Koo Chung (PHD)	_____	_____
(Advisor)	Signature	Date
_____	_____	_____
(Chairperson)	Signature	Date
_____	_____	_____
(Internal Examiner)	Signature	Date
_____	_____	_____
(External Examiner)	Signature	Date

Acknowledgment

Foremost, I would like to thank the Ultimate God for his almighty, and his mother Saint Merry. I am thankful Lord, for everything that you allow to cross my path. I am also thankful for the decisions that you allowed me to make and the lessons that come from these decisions.

I would like to pay a special thankfulness to my advisor Prof. Yun Koo Chung for his support and supervision during the implementation and completion of this thesis.

I respect and I wish to present my special thanks to Mesfin Abebe (PhD.), for his kindly understanding of my research idea and supporting me to get the dataset used in this thesis. I am greatly thankful for his comment on this thesis. I would like to pay my special thankfulness to Lijun (PhD.), for his willingness, support, and approval to get a dataset from Binghamton University New York, America.

I would like to thank Computer Vision and Robotics SIG members especially Mr. Anteneh Tilaye and Miss. Fetlework Kedir for discussing different ideas regarding our thesis titles that helping me to view my research idea from different angles. My special thanks also goes to my family members, especially my Mom and Dad to give me comfort in my study and to encourage me in any activities.

Finally, I would like to thank all friends, specially Andualem Dribe and Yonas Kenenisa for day to day activities we have together

Table of Contents

List of Tables

List of Figures

List of Acronyms

3D	3 Dimension
2D	2 Dimension
BU-3DFE	Binghamton University 3D Facial Expression
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EOH	Edge Orientation Histogram
FER	Facial Expression Recognition
GPU	Graphic Processing Unit
HCI	Human Computer Interaction
HOG	Histogram of Oriented Gradient
KNN	K –Nearest Neighbour
LBP	Local Binary Pattern
LTBP	Local Threshold Binary Pattern
LPQ	Local Phase Quantization
PCA	Principal Component Analysis
SLPP	Supervised Locality Preserving Projections
SVM	Support Vector Machine
RF	Random Forest
ULBP	Uniform Local Binary Pattern

Abstract

*The facial expression which carries rich information on body behaviour is the leading carrier of human affective and symbol of intelligence. Currently, facial expression recognition is becoming an interesting application and research area because of emerging technology like affective computing and human-computer interaction. 3D data can improve the recognition rate of facial expression by overcoming the problem of 2D data. To overcome this limitation most researchers used a feature fusion of both 2D and 3D data. The main purpose of this thesis is to recognize 3D human facial expression. (1) The study proposed a solution using CNN with a dlib landmark detection method to detect face and landmarks from the face. (2) Then using localization, the facial parts named eyebrow, eye, mouth, and nose are extracted from both texture and depth images models by adding some neighbour pixel from the detected landmark coordinates. The features of each facial part are then extracted using Resnet50 pre-trained model. (3) The extracted feature is combined or fused using a deep fusion technique, which has four input layers for each face part and then dimension reduction is implemented for the sake of better performance. (4) For dimension reduction, PCA is used, which is well known and better than other dimension reduction techniques. (5) Finally, the SVM classifier is trained on both 7 and 6 expression types by feature obtained after the dimension reduction steps. For comparison, LR and SoftMax are also trained using the same feature. To check the recognition performance of proposed solutions public dataset (*BU_3D FER) *is used for training and testing. The proposed solution achieved 85.38 and 89.54 average accuracy on 7 and 6 expression type datasets, respectively, where 7 expressions are Angry, Disgust, Fear, Happy, Neutral, Sad and Surprise. The proposed solution achieved a 92% recognition rate for neutral and 93% for surprise facial expression types. Fear facial expression type is recognized 77% which most lower recognition rate. A final average accuracy of the proposed solution is compared with other previous methods with similar problems and it achieved better performance.*

Keywords: *3D Facial Expression, Feature Fusion, Emotion Recognition, Deep-learning Based System*

Chapter One

1 Introduction

1.1 Background of the Study

The human face and its features have been one of the important topics in computer vision. Currently, human facial expression recognition is playing a great role in the day to day activities of human beings. A recent study, contrasting human and humanoid robot facial expressions, suggests that people can recognize the expressions made by the robot explicitly, but may not show the automatic, implicit response. To overcome this limitation researchers and engineers still provide huge task, which means currently most technologies innovated or developed regarding to HCI included human facial recognition system that is used for the different purposes, additionally they use human face to read the emotion of individual by analysing their facial expression and use their emotion characteristics for interaction with robots and other purposes. Currently, facial expression recognition is used in a different application area, the currently developed robots like humanoid robots and others use facial expression recognition as one input to interact with the human, customer satisfaction analysis, character substitution, depression analysis and also used as biometric security depending on the user interest.

In [1], [2] 2D facial expression recognition technique most widely 2D face images are used, because the availabilities of large 2D image collections as a dataset, but 2D face expression recognition techniques are sensitive to external factors such as varying illumination, shadows, pose and even to the use of cosmetics. Moreover, they require a good quality of images. Also, it should be a frontal image and these criteria must be meet for both the probe and the gallery image. 3D facial expression recognition technique is used to overcome the limitation of the 2D facial expression recognition technique. Especially current research papers, [3], [4] used the feature of 2D and 3D facial images together to improve the recognition rate by overcoming the limitation of one another.

This thesis is emphasized on the 3D facial expression recognition to have a solution for the current problem of facial expression recognition. The facial expression has seven commonly known categories named angry, disgusted, fear, happy, neutral, sad, and surprised. To recognize

the emotion of the person from the face image or from the specific frame of video bulk of the researcher's written researches on facial expression recognition. Currently, the average recognition rate of facial expression recognition is 91.32% by wang [5] on six (6) facial expression categories, and on seven (7) facial expression categories 77.3% recognition rate is achieved by Han [6]. It still needs to improve the recognition rate of facial expression regarding expression categories and recognition performance.

1.2 Motivation of the Study

The motivation for this research mainly arises due to the observation of a minimal research contribution in 3D face recognition and 3D facial expression recognition research area. Especially the number of researchers in our country Ethiopia who engaged in this area is very limited. Increasing the contribution of Ethiopian researchers one step forward in the subject area is the main initiation to come up with this research title. Facial expression recognition is an essential computer vision field that can be applied to different real-life tasks. However, the currently available 3D facial expression recognition systems have not provided satisfactory accuracy. Accuracy of a system is very crucial to get the required benefits from the application area under the subject. Therefore, an improvement in the accuracy of facial expression recognition is needed to exploit the better benefits of each application area. This study is proposed with the basic motivation of this fact as well. Understanding the dataset types and ways of combining different types of datasets like texture images and depth images can help to improve the recognition rate of expression. Above all design and develop an algorithm by understanding each expression type in different ethnics is better in order to improve the recognition rate in case real-world implementation. Still, most research papers in this area only used 6 and 7 expression types, but in the case of real-world types of expressions are not limited to the above expression numbers. So, a study in this area can improve the number of expression types used in expression recognition research. Finally, as a facial expression is used in different human and robot interaction, health care, character substitution, and customer satisfaction application it motivates any researcher to study in this research area.

1.3 Statement of the Problem

In order for the computer to interact with humans better, understanding human behaviour is important. The human facial expression can tell the computer a lot of detail. Currently, human and computer interaction is a day to day activity, so to analyse and understand the behaviour of human's, robots must recognize the facial expression from the human face. The development of an accurate facial expression recognition algorithm is important. Several researchers have proposed a different solution using 2D and 3D data. Facial expression recognition based on 2D image is affected by illumination, pose, and cosmotics, so the accuracy of 2D facial expression recognition is not always guaranteed.

Different algorithms based on local features and deep learning extraction techniques are also developed by different researchers for 3D data. Most of the works done on 3D data have the following drawbacks.

- The recognition rate of 3D facial expression recognition is not satisfactory.
- Most researches papers depend on six (6) facial expression categories
- The current research paper which used whole facial features and facial part features not have efficient accuracy to recognize expression.

This study attempts to answer the following research questions.

1. Which facial parts feature best represent facial expression?
2. What is an efficient feature extraction technique for facial expression recognition?
3. How different features are fused to get a better recognition rate?
4. Which classification technique has better performance for 3D facial expression recognition?

1.4 Objectives

1.4.1 General Objective

The general objective of this thesis is to design and develop deep learning-based 3D facial expression recognition using feature fusion.

1.4.2 Specific Objective

- Investigate and understand existing techniques for FER based on 2D and 3D information
- Investigate and understand how to implement deep learning for facial expression recognition.
- To select 3D facial expression dataset for training and testing
- To select accurate pre-processing techniques
- To select a proper algorithm to detect the facial landmarks and extract features
- Identify and select best deep-learning-based feature extraction architectures depending on the datasets
- To select or design proper classification methods

1.5 Scope and Limitation

1.5.1 Scope

In this thesis the main target is to identify the expression of the individuals depending on 7 commonly known facial expressions, using the facial part extracted by a pre-trained deep learning model and it's to investigate some method and methodology that is used in the thesis.

1.5.2 Limitation

In this thesis deep learning is not used directly for the classification because the dataset size is not enough for direct deep learning training, but for comparison and if our dataset is enough to train deep learning the method used for the classification may use handcraft and deep learning classifiers.

1.6 Research Methodology

The methodology is used to describe the detail about the proposed method by discussing data pre-processing, algorithms for feature extraction, feature combination (feature fusion), dimension reduction, classification techniques. and type of data used in the proposed method, software and any simulator used for the proposed method is discussed briefly in deep discussed in chapter 3. The paper discussed some of them in short to highlighting the reader about the methodologies as follow:

A. Pre-Processing

After collecting the necessary data, the next step is performing pre-processing on the acquired face images to remove some noise and fill some holes.

B. Feature extraction technique and Landmark detection technique

In this thesis before feature extraction facial landmark detection is performed to extract facial parts. After facial part extraction by using Convolutional Neural Network (CNN) features are extracted from each face part. Then before the classification Principle Component Analyses (**PCA**) [7], [8] used for reduction from high dimensional space to a lower-dimensional subspace. The dimension reduction space should make clear distinctions between different classes in the dataset.

C. Classification Method

The classification method is the main part of any research. The classification method is depending on the classes used in the classification and the features extracted from the specific dataset. In this thesis Support Vector Machine (**SVM),** Logistic Regression **(LR)** methods are used to classify the expression depending on the input data.

D. Dataset Selection for Training and Testing

To train the proposed method first the method must have trained using public data and tested using public data too. In this thesis dataset (**BU-3DFE)** for the training and testing of the proposed method.

E. Finally, for comparison purpose select previously used methods

To check the newly proposed method is performed better than the previously used methods which are used local feature and global information by combining the features [1] and compare like LBP + SLPP with KNN [2], LTBP + HOG with SVM [3] and Fuzzy SVM [4] methods with the same public dataset.

F. Programming Tools

To achieve the goal of the study we have to use programming tools for writing the code and data acquiring. The programming tools and library planned to use are as follow:

- **Keras**: - is new open-source libraries (frameworks) that contained pre-trained features with most commonly known architectures like Alex Net, VGG-16, VGG-19, Resnet50, InceptionV3, Xception and other libraries used for early stop and model checkpoint identification and others which needed in deep learning.
- **TensorFlow**: - is an open-source library(framework) which is used to implement the deep learning and improved high abstraction, it's easy to get API of TensorFlow than other frameworks. TensorFlow is deployed on CPU and GPU which means it is possible to configure it on the Computer without using the graphics card, in case the system needs high processing performance it is configured on the graphics processing unit. As a conclusion, it's easy to deploy and learn from other open-source libraries.
- **Python**: - do prototyping with Python (easier to setup environment, implement algorithm, make proof-of-concept) which means it is faster than C^{++}.
- **Microsoft Office Word and Visio** - are tools used to write the thesis paper and draw diagrams needed for the thesis.

- Finally, drivers used to configure some devices with the computer are used as programming tools.

1.7 Beneficiary and Application of Results

Facial expression recognition is used in a different area depending on the service provided for the human or machine that means the application area of facial expression is depended on the services. The improvement in the study provides huge improvement for a lot of application areas which is possible to implement the facial expression and benefit users. Some application area of the study is as follow:

- Biometric Security: - users used a facial expression as biometric security for their device and home security by integrating with the face recognition system.
- Intelligent Human-Computer Interaction: - used in humanoid robots to identify the emotion of humans from their facial expressions.
- Clinical medicine for autism: - it is implemented to identify autism on the children before it is getting worse.

- Depression Analysis: - it is implemented for the depression analysis and sleepy face detection
- Customer Satisfaction Analysis: - it is implemented in industry or market areas to analysis customer service satisfaction.
- Film Industry: - used to pass the expression of the actors to the character in the movie especially in the case of animation movies.

1.8 Thesis Organization

The process of this research study is classified into 7 different chapters, from the chapters first chapter discusses about the introduction or background of the study, the motivation, statement of problems, which contain list of specific problems, and the objective of this study, the scope and limitation, the methodology of study and beneficiaries and application of the results.

In the second chapter, the study discusses the background literature on facial expression recognition on the different datasets. Discuss different face detection algorithms from traditional machine learning until deep-learning as well discuss facial expression recognition algorithms, about transfer learning and types of transfer learning, discuss emotion recognition using 3D dataset and emotion recognition using feature fusion and finally, discuss related work and gaps in each related works.

Chapter three discuss methodology in the study in detail, which contain literature review collection, dataset collection, classification, designing, feature extraction, dimension reduction methods with their mathematical representation used in the study and programming tools used in coding, designing and editing and algorithms used for the compression also discussed in this chapter.

In chapter four the main point of discussion is about the proposed solution in detail and discuss the development life cycle like system model, flow chart and pseudocode in training, testing and algorithms with their mathematical equivalent.

Chapter five discusses the implementation step of the proposed solution, which contains training steps, testing steps with implementation environment setup, sample source code of the proposed solution.

Chapter six discusses the proposed solution results based on the collected runtime metrics, accuracy, precession, and recall. Compare the new result of the proposed solution with the previous solution with the same data set.

Finally, chapter seven discusses the conclusion of the study and discover some new perspective of future works that would be a problem area for researchers on the related discussion area

Chapter Two

2 Literature Review and Related Work

2.1 Overview

One or more movements of muscles in the human face can convey the emotional state of an individual. Facial expression can be taken as a non-verbal communication between humans or between humans and computers or robots. In computer vision and artificial intelligence human facial expression or human emotion recognition is very sensitive research area, that means researcher studies new finding using both **2-D**(two-dimensional) and **3-D**(three-dimensional) datasets also use other face datasets like infrareds, Gabor face and other features.

In early 1978 Ekman and Friesen proposed a new method for facial emotion recognition named Facial action coding system (FACS), this method depends on a change of facial muscles called action units (AUs). After this Facial Landmark (FL) [9], [10] method is proposed, in this method points which visually salient are selected from the face and depending on points calculate some distance from one point to another point and train the model using the distance for the expression or emotion recognition. In facial expression recognition history, 2D features based facial expression recognition methods are still having limitations like invariant to the illumination and also to the head poses, occlusion and makeups. To overcome the above problem many researchers, study the 3D features based facial expression recognition methods.

2.2 Feature Extraction Techniques

Feature extraction is depending on appearance information, geometric information temporal and spatiotemporal information. computer vision has many different feature extraction techniques which are traditional(vision) based feature extraction and deep learning-based feature extraction. From all feature extraction techniques, we only discuss some techniques that have a high relationship or contribution to facial expression recognition.

2.2.1 Vision-based Feature Extraction Techniques

- **LBP (Local Binary Pattern): -** in facial expression recognition LBP [11], feature extraction technique is most common and popular which use 3x3 window or filter that takes the center pixel value of the window as threshold value and compare each

neighbourhood of the threshold value with threshold value in the window, then if the neighbour value is greater than or equal to threshold value binaries neighbour with **1** or binaries with **0** values. The Mathematical expression of a local binary pattern is discussed as follows.

$$LBP_{P,R} = \sum_{i=0}^{p-1} S(g_i - g_c) \times 2^i \qquad s_i = \begin{cases} 1 & if \quad g_i \geq g \\ 0 & \quad\;\; g_i < g \end{cases}$$

[11] The detail discerption of the above mathematical formula is the value of ***p*** is denoted by window size or filter, 'g_c' is the center pixel value of the window/ threshold value, g_i take all threshold surrounding pixels in window one by one and subtract threshold value-form each until '***p'*** reaches window size – 1. Function ***S(x)*** where ***x*** is '$g_i - g_c$' would determine binaries value if '***x'*** is **>= *0*** then assign "**1**", else if ***x*** is < **0** then assign "0". Finally, after all, threshold neighbour's pixel value is binarized, then combine the 8-bit binary value from the left-top pixel and convert to decimal value by multiplying each binary value by 2^i where "***2***" is base and "***i***" represent the position of the value. [11] In addition to LBP, Uniform LPB, Extended LPB, Local Threshold LBP and Number LBP used for feature extraction in different cases and dataset types.

- **SIFT (Scale Invariant Feature Transform): -** key-points extracted using scale-space extrema in difference-of-gaussian (DoG) from a specific image, which used orientation features (Descriptor) with 128-dimension size and it is invariant to scale and rotation also partially invariant to illumination. difference-of-gaussian better of detecting corners, edges, and blobs. It is responsible to provide local maxima between scale and space with a list of (x, y, σ), where (x, y) are key points and σ is scale size. To get DoG gaussian of bluer is implanted between two σ, which is σ and xσ with different octaves. It is possible to use PCA by applying on gradient batch of SIFT to get descriptor with 20-dimension size which is most robust faster than SIFT.

2.2.2 Transfer Learning

Now a day there are two main ways to use deep learning for the specific problem with proper datasets. as discussed in [12], [13] transfer learning is known as a staring task for new tasks and

it has a huge difference with machine learning. From the ways, the first is to develop new architecture using deep learning and train the architecture using the dataset until better accuracy is obtained and store the model to use for the current problem. The second way is transfer learning, as the name indicated transfer previously trained deep learning architecture to solve current problems. In general, there are two types of transfer learning in the context of deep learning.

2.2.2.1 Transfer Learning for Feature Extraction

Feature extracted using a pre-trained model like [14] VGG16, VGG19, Resnet50[14], Inception, Xception [15] and other models, this means take a pre-trained model(architecture) with its previous weight except for the top of the model or without the classifier layer of the model. Using a new model without classifier layer extract feature from the specified dataset and use extracted features to train other machine learning algorithms or design new deep learning architecture and train it using extracted features. This way of feature extraction techniques known as convolutional based feature extraction.

2.2.2.2 Transfer Learning for Fine-tuning

Transfer learning with fine-tuning means a select pre-trained model that most suitable for specific problems and train selected models by changing the classification layer only or freeze part of layers from the pre-trained layer and use with a new classification layer. It is known as classifier transfer learning. The main difference between the transfer learning for feature extraction and fine-tuning is on the classification layer.

2.3 Dimension Reduction Techniques

The dimensional space of feature extracted using deep learning and feature combined together after extracted using two or more different extraction method is high, therefore train machine learning method(classifier) using the extracted features is taking a lot of time and reduce the performance of the classification. Different types of dimension reduction algorithms used to reduce from high dimensional space to low dimensional space before the classifier is trained. The study discusses some poplar linear dimension reduction algorithms like [11][7], [8]PCA, ICA, FA and hybrid of PCA and ICA.

2.3.1 Principle Component Analysis

Principle component analysis is the most common dimension reduction technique which is easy to implement, and faster to reduce feature from high dimensional space to a lower-dimensional subspace. Use the eigenvalue decomposition covariance matrix. Most commonly used for continuous data and project the features to increasing side of variance and select principal component with high variance depending on the garget variable. In PCA first principal component have the maximum variance and the second principal component has less variance than the first principal component and maximum variance than 3rd principal component. This feature reduction technique discussed in chapter 3(three) in detail with its mathematical expressions.

2.3.2 Independent Component Analysis

Independent component analysis is a supervised feature reduction technique and it is used to reduce dimension using independent factors or constraint with there all momentums and second-order statistics while PCA depends on the second-order statistic only. The model is mathematically represented by $\boldsymbol{a} = \boldsymbol{W_X}$, where '**W**' represents mixed matrix and problem formulated as one of source separation, '**x**' represents vectors called sources and '$\boldsymbol{a}$' represents the observations on feature reduction. The limitation of this technique is working with an assumption of subcomponent comparison with signal source are non-gaussian. This technique is most commonly used in medical researches for signal extraction and separation from biomedical datasets.

2.3.3 Factor Analysis

FA is also one of the linear feature reduction techniques which are achieved better results of a dataset with Gaussian noises. It is used to simplify the complex relationships between classes and/or variables. The values of observed data are expressed as functions of a number of possible causes in order to find which are the most important.

2.4 Machine Learning (Classification) Techniques

In computer vision many different machine learning or classification techniques used depending on the dataset, which means if a number of datasets are minimum most commonly used handcraft or vision-based classification techniques are used, but if the number of datasets

approximately closer or greater than 5 thousand deep learning classification techniques is recommended for better accuracy. when classification techniques are selected number of target classes in the dataset must be considered to select different parameters for the selected classification techniques.

2.4.1 Vision-based Classifier(hand-craft)

In computer vision different types of hand-craft classification techniques used for classification problems. The most commonly used classification techniques are SVM with different kernel types, Logistic Regression, K-Nearest Neighbours (KNN), Random Forest and others. [16] From hand-craft techniques specifically SVM with the linear and Gaussian kernel, [17] KNN and Logistic Regression classification techniques used in expression recognition researches.

2.4.2 Deep-learning Classifier

In [3] deep-learning classifiers and parameters of the classifiers are depending on the target class in the dataset. If the target class is binary, the classification layer in deep-learning has an activation function with a sigmoid parameter, and if the target class has more than two target classes in the dataset, the classifier layer has an activation function with a soft-max parameter. In deep-learning to calculate the loss and accuracy of the model different optimizers are used. From optimizers SGD, Adam is used widely in expression recognition researches. In deep-learning other parameter is learning rate, a learning rate has grate factor on the accuracy achieved using deep-learning classifier and has different variation depending on optimizer used in the classification.

2.5 Related Work

Know a day most of the researchers use multi-modal dataset for expression recognition which means use the feature generated from the 2D face image and also use features generated from 3D facial image together by using combining or fusion algorithms to overcome the limitation of one another, but recognition rate of facial expression recognition is still not on the satisfactory stage as face recognition. Recently after 2014 G.C, many researchers studded to solve the problem of facial expression recognition, as a literature review discussed the following studies.

2.5.1 Emotion Recognition Using Feature Fusion (2D+3D)

Currently, feature fusion is a top research area for a lot of researchers, because Feature fusion help to improve recognition performance by overtaking the limitation of one another. Feature fusion can be implemented in two ways, methods feature fusion and data feature fusion. Method feature fusion means more than two methods extract features from the same data/image separately and combine extracted features. Data feature fusion means one method extract features from the same data/image, but with a different dimension, format, pre-processing and so on. As an example, extract features from one person 2D data, 3D data, normal maps, mesh and others, then combine the feature of each data together. Using feature fusion some researcher discussed to new finding as follow.

[3] Proposed partially deep learning-based facial expression recognition method that uses a combined feature of both 2D and 3D facial images. In this study, Facial Landmark detection techniques are used to localizing the face parts like mouth, nose, eye, and eyebrows in case of facial part extraction. After the facial part extracted from the face, VGG-M-DF is used as feature extraction and feature fusion method and PCA is used for the dimension reduction then finally for the classification used SVM with the polynomial kernel. In this study, BU_3D FER public dataset is used as a training and testing dataset and using a proposed method they obtain recognition performance 88.54%, but the recognition rate of the method is still not satisfactory and, in this discussion, the pre-processing is not applied and discussed at all.

Asim Jan and Hongying Meng [18] proposed a method used both 2D and 3D facial image features. In this study to extract features from the 2D facial images they used LPQ, EOH and ULBP algorithms and to extract features from the 3D facial images the used Landmark Detection and Facial Mesh Distance methods, then combine or fuse the features generated from 2D and 3D facial image. After the fusion is done the dimension reduction is implemented to make the feature dimension better for the classifier. In this study, they used Multi-Class SVM as the classifier. The recognition rate of the proposed method is 82.32% on the dataset of BU_3D FER. The performance of this study is poor when compared to other studies and pre-processing is not implemented in this study.

[19] Proposed method for facial expression or emotion recognition using multi-modal 2D and 3D Features of facial images. Incremental Parallel Cascade of Linear Regression (IPar-CLR) method is used to extract features from the 2D facial images and Histogram of Second-Order Gradient (HSOG) and SIFT also used to extract feature from 2D texture images. Mesh of Histogram of Gradient and Mesh of Histogram of Second Order is used to extract features from the 3D facial images. Then as the classifier, they proposed SVM and they obtain the recognition rate 86.80% for the proposed method by using the BU_3D FER dataset. This proposed solution is not performing well on large pose variations and data loss.

[4] Proposed multi-modal 2D and 3D facial expression recognition with deep fusion CNN. In the proposed method the system used 2D and 3D features, to extract the features from the facial images used Deep Fusion Convolutional Neural Network (DF-CNN) with the subset of feature extraction. After feature extracted using DF-CNN with Feature Extraction Subset then extracted features passed to the next level for the dimension reduction. For the dimension reduction, the method used Deep Fusion Convolutional Neural Network (DF-CNN) with the subset of feature fusion. Then the classification is done using two different methods name DF-CNN SoftMax and DF_CNN SVM. The recognition rate of the proposed method is depending on the classification methods. The recognition rate for the DF_CNN SoftMax is 86.82% and DF_CNN SVM is 86.86%. The method is used in two different public datasets named BU_3D FER and Bosphorus. Like any other method, this method has its own gap the performance is unsatisfactory on video-based facial expression recognition.

For more clarity and understanding, in short, the following tables on the next page named show more information regarding the above discussions.

Table 2. 1: Research Review by using feature fusion 1

Research's		*Accurate Facial Parts Localization and Deep Learning for 3D Facial Expression Recognition*	*Automatic 3D Facial Expression using Geometric and Textured Feature Fusion*
Data		2D + 3D	2D + 3D
Feature Extraction Algorithm	***Texture***	Facial Landmark, VGG-M-DF	Local Phase Quantization (LPQ), Edge Oriented Histogram (EOH), Uniform Local Binary Patterns (ULBP)
	Geometric		Landmark Detection and Facial mesh Distance
Dimension Reduction		PCA	PCA
Classification		SVM with Polynomial Kernel	Multi-Class SVM
Accuracy		88.54%	82.32%
Dataset		BU_3D FER	BU_3D FER
No of Expression Type		6(six)	6(six)
Gap		No Pre-processing and Accuracy is Not satisfactory	No Pre-processing and Accuracy is Not satisfactory
Year		2018 G.C	2015 G.C
Authors		Asim Jan et al.	Asim Jan & Hongying Meng

Table 2. 2: Research Review by using feature fusion 2

Research's		*An Efficient Multimodal 2D + 3D Feature-Based Approach to Automatic Facial Expression Recognition*	*Multimodal 2D+3D Facial Expression Recognition with Deep Fusion Convolutional Neural Network*
Data		2D + 3D	2D + 3D
Feature Extraction Algorithm	***Texture***	incremental Parallel Cascade of Linear Regression (iPar–CLR), HSOG (Histogram of Second-Order Gradient) and SIFT	DF-CNN with Feature Extraction Subnet
	Geometric	Mesh HOG and meshHOS (Histogram of Second-Order Gradient)	
Dimension Reduction		__	DF-CNN with Feature Fusion Subnet
Classification		SVM	Linear SVM and SoftMax
Accuracy		Gaussian 88.78%	DF-CNNsvm 86.86% DF-CNNsmax 86.82%
Dataset		BU_3D FER	BU_3D FER and Bosphorus
No of Expression Type		6(six)	6(six)
Gap		Fail on large pose variation and data loss and performance limitation in large pose variation	Accuracy is not satisfactory and poor performance on video-based facial expression
Year		2015 G.C	2017 G.C
Authors		Huibin Li	Huibin Li el.

2.5.2 Emotion Recognition Using 3D Model

To overcome the limitation of 2D data on facial expression recognition the greatest number of researchers diverted to use 3D data in different ways or formats. The most researchers used 3D data/ image features by obtaining different types of mesh, landmark detection and mixing the depth information with the texture data. As emotion classification method researchers used different techniques beginning from LBP to currently used CNN. For more detail knowledge the following researchers proposed and discussed the different methods in deep.

[20] Xiaoli Li et al, proposed a new method only uses features of 3D facial images. This method from the geometric images generates a 3D mesh facial model. After the model is generated for the classification SVM with Linear, Gaussian and sigmoid kernels functions are used. The recognition rate of each kernel function is different because of the performance of the kernels are also different. The recognition rate obtained by the **linear** function is 81.33%, **Gaussian** function is 88.78% and for a **sigmoid** function is 84.67%, by using BU_3D FER public dataset for training and testing the proposed method. As the gap this method depends on the nose tip, this means if the nose tip is not detected accurately the recognition of the expression also dropped and detecting the nose tip is difficult so this makes the expression recognition difficult and poor performance.

[21] Proposed CNN based facial expression method that uses 3D facial features. In this study, the Orthogonal Projection method and Landmark detection are used as the feature extraction techniques from the geometric data. After features extracted the classification of the recognition is done by using CNN. Finally, the recognition rate of the proposed method is 75.90% with the gaps like invariant to pose and illumination variations. In this discussion, the dataset used for the training and testing purpose is BU_3D FER public dataset.

[22] Proposed method using methods depth map, normal map, and shape index map with OTMFA and multi-class SVM. In this study methods depth map, three normal maps and especially shape index map is used to extract features from the geometric data. After the features extracted from the geometric data then Orthogonal Tensor Marginal Fisher Analysis used for dimension reduction. After the dimension is reduced multi-class SVM is used as the recognizer

or classifier. The proposed method is performed 84.27% recognition rate on the BU_3D FER public dataset, but the performance of the proposed method is not satisfactory regarding other methods.

[16] Proposed a new method using Fuzzy SVM with SFFS. In this method 3D facial features used and to extract the features from geometric data, they used the Sequential Forward Feature Selection (SFFS) method. After features extracted Fuzzy SVM method is used as the classifier. The recognition rate of the proposed method is 86.67%. The method used the BU_3D FER public dataset for the training and testing of the performance of the recognition. The performance of the method is reducing for low-complexity practical application of FED.

[23] Proposed a new facial expression method depending on the Kernel methods and Riemannian manifold. In this method to extract the features from the geometric data, they used Landmark detection with covariance matrices and Symmetric Positive Definite (SPS). After the feature's detection is done, then proceed to the classification. The proposed method uses the SVM algorithm for classification. The performance of the proposed method is 86.17% on the BU_3D FER public dataset and also used another public dataset name Bosphorus. Regarding the performance, the method has limitations, which means the recognition rate depends on the covariance matrices. If the performance of the covariance matrices is degraded, then the performance of the recognition is also reduced.

[6] Proposed facial expression recognition method with LBP and SLPP algorithms. The data used in this method is only 3D data and to extract the features from the geometric data the method used Local Binary Pattern (LBP). After the features extracted Supervised Locality Preserving Projection (SLPP) used for dimension reduction. Then for the classification, the K-Nearest Neighbour algorithm is used. In this method, the dataset used for training and testing the performance of the recognition is CASIA-3DFace and BU_3D FER which is publicly available. The maximum recognition rate of the proposed method is 83.10% on the BU_3D FER public dataset, but the main problem of this method is it takes a long period of time on the recognition.

Above this many previously proposed, methods are discussed regarding the methodology and algorithm they used and also a dataset used in their discussion finally the recognition rate they obtained from their method is discussed briefly from 3D data/image angle. For more clarity and understanding, in short, the following tables named Table 2.3, 2.4 show detail information regarding the above discussions.

Table 2. 3: Research Review using 3D Model 1

Research's		*Analysis of Range Images Used in 3D Facial Expression Recognition Systems*	*CNN based 3D Facial Expression Recognition Using Masking and Landmark Features*	*3D Facial Expression Recognition Using Orthogonal Tensor Marginal Fisher Analysis on Geometric Maps*
Data		3D	3D	3D
Feature Extraction Algorithm	***Texture***	—	—	—
	Geometric	3D mesh Facial Model	Use Orthogonal Projection Method and Land Mark Detection	DM(Depth Map) NOM(Normal Map) Shape Index Map(SIM)
Dimension Reduction		—	—	Orthogonal Tensor Marginal Fisher Analysis (OTMFA)
Classification		SVM with Linear, Gaussian, Sigmoid kernels functions	Multi-Class SVM CNN	Multi-Class SVM
Accuracy		Gaussian 88.78%	75.90%	84.27%
Dataset		BU_3D FER	BU_3D FER	BU_3D FER
No of Expression Type		6(six)	6(six)	6(six)
Gap		difficult to get nose tip in high pose variation.	invariant to pose and illumination variations and poor performance	poor performance related to others
Year		2016 G.C	2017 G.C	2017 G.C
Authors		Xiaoli Li	Huiyuan Yang & Lijun Yin Qian Li	Gaoyun An & Qiuqi Ruan 2

Table 2. 4: Research Review using 3D Model 2

Research's		*Fuzzy SVM for 3D Facial Expression Classification using Sequential Forward Feature Selection*	*3D Facial Expression Recognition using Kernel Methods on Riemannian Manifold*	*Facial Expression Recognition with LBP and SLPP Combined Method*
Data		3D	3D	3D
Feature Extraction Algorithm	***Texture***	__	__	
	Geometric	Sequential Forward Feature Selection (SFFS)	Landmark Detection with Local Covariance Matrices, Symmetric Positive Definite (SPD)	LBP
Dimension Reduction		__	__	SLPP (Supervised Locality Preserving Projection)
Classification		Fuzzy SVM	SVM	KNN
Accuracy		87.67%	86.17%	83.10%
Dataset		BU_3D FER	BU_3D FER and Bosphorus	CASIA-3D Face and BU_3D FER
No of Expression Type		6(six)	6(six)	7(Seven)
Gap		The Performance reduced for low-complexity applications of FED	If the performance of the covariance matrices is reduced the Performance of the method also decreased.	Recognition time is increased depending on the performance of recognition
Year		2017 G.C	2017 G.C	2014 G.C
Authors		Payam Zarbakhsh and Hasen Demirel	Walid Harir el.	Dan Han & Yue Ming

2.6 Summary

Facial expression recognition system can be implemented on both 2D and 3D dataset and many researchers implemented different methods to improve the accuracy of facial expression and increase the expression recognition types. [20] In most research papers 6 expression types are used and the expression recognition rate is 88.78 by 6 expression types on the BU_3D FER dataset, but the recognition rate on 7 expression types is 77.3 on the same dataset [6]. Using facial parts like eyebrow, mouth, eye, nose as a dataset for expression recognition is helps to achieve better performance than using the whole face. Combining 2D and 3D dataset features also improve the performance of recognition in this problem. [3] In previous literature features are extracted from each facial part of texture image and depth image separately. Those extracted features are then combined together into one feature vector that represents the whole face image. unlike these works, this study proposed different ways to fuse facial features. Features are extracted from all facial part of texture and depth images at a time and will be combined together into one feature vector that represents the whole face image. For the classification technique vision base or handcraft methods with different parameter value and deep learning methods with different activation function, optimizers and learning rate is used and handcraft methods are obtained better performance than deep learning methods because the dataset on 3D facial expression recognition is not enough for deep learning classifier

Chapter Three

3 Research Methodology

3.1 Overview

Most of the researchers discussed their methods to collect data, apply pre-processing, extract features, train model and use the model for the recognition purposes also design and show ways to design system models. Starting from scratch which means, identifying problems in the proposed approach reviewing the previous work and observing the current expression recognition environment are performed. After problem identification, analysing the problem and solving methods like selecting feature extraction algorithms, classification algorithm, and dataset selection are the next steps which are discussed in this stage. After the algorithms are selected the way to integrate and use the selected algorithm and developing the system model for the integrated algorithms is next. Following this develop a model for the classification and finally test the model using public datasets.

3.2 Methods

3.2.1 Data Collection Procedure

The proposed method is trained and tested using public datasets. In this thesis from some 3D dataset from Binghamton University 3D Facial Expression (**BU-3DFE)** [24] datasets are used for the training and testing of the proposed method. The dataset has a 2500 3D dataset from 100 individuals, where 56 individuals are female and 44 individuals are male. 6 expression types have 4 samples with different intensity values for each individual except neutral expression type have a single sample of each individual. In addition to 3D samples, the dataset has a 2D texture sample for each 3D samples.

3.2.2 Pre-Processing and Segmentation

After collecting the necessary data, the next step is performing pre-processing on the acquired face images to remove some noise and fill some holes and generate depth images from each 3D model using a MeshLab application.

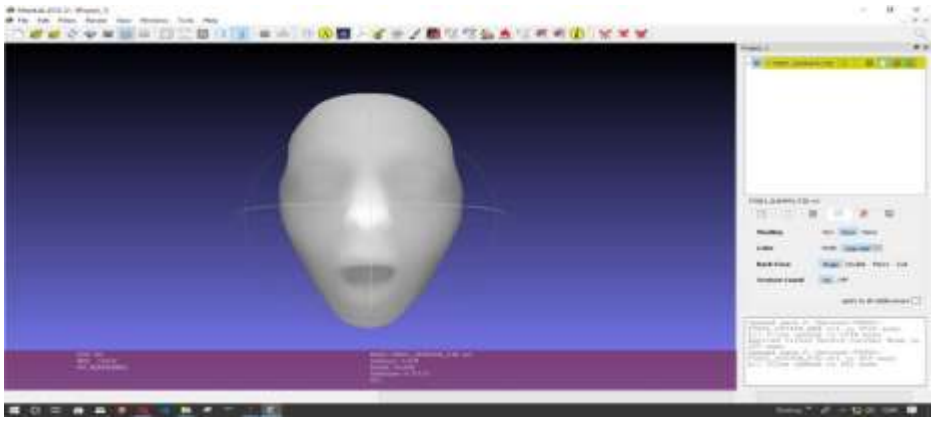
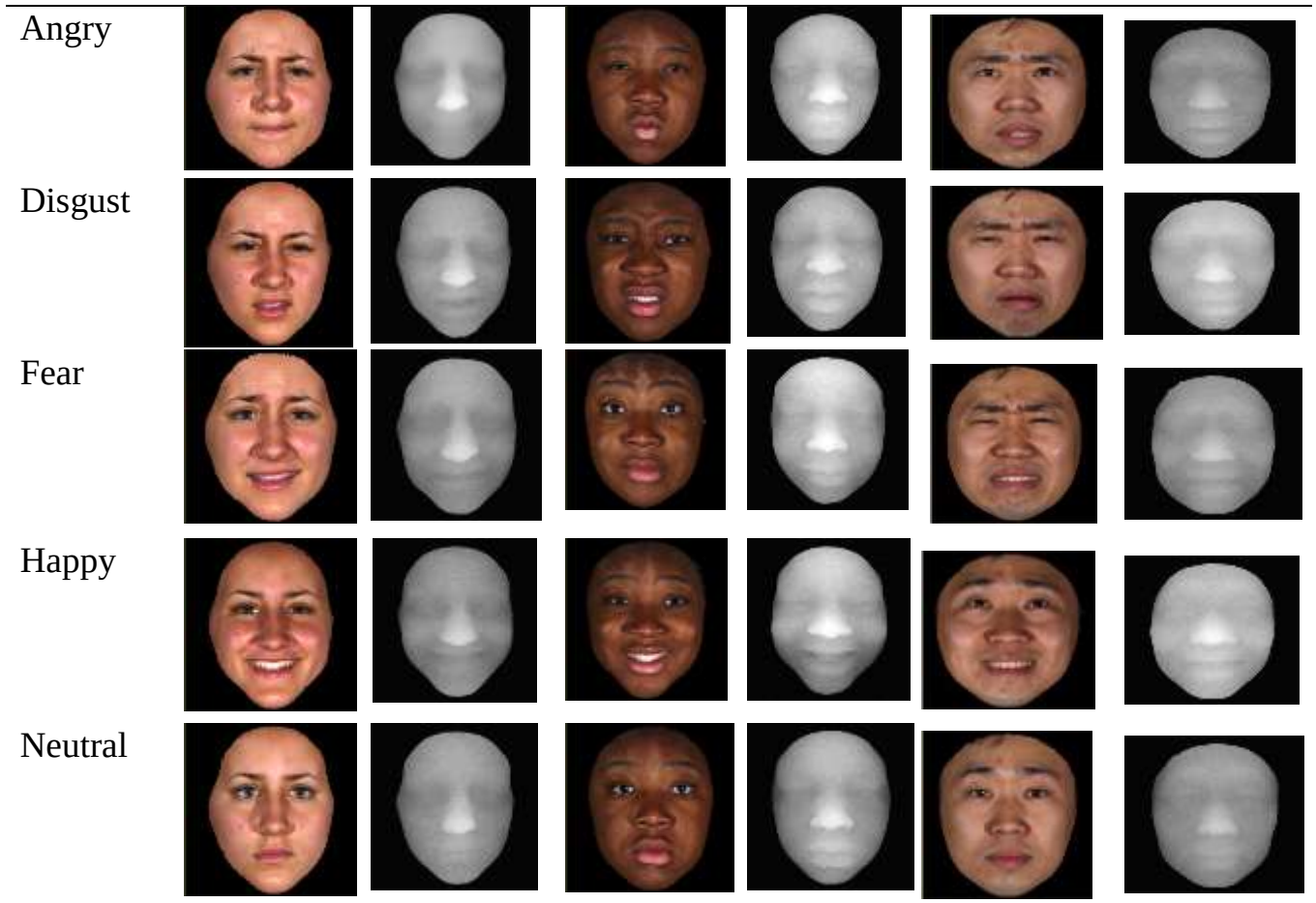


Figure 3. 1: Application to generate a depth image

The following Figure shows the original 3D image and depth image generated from a 3D image.



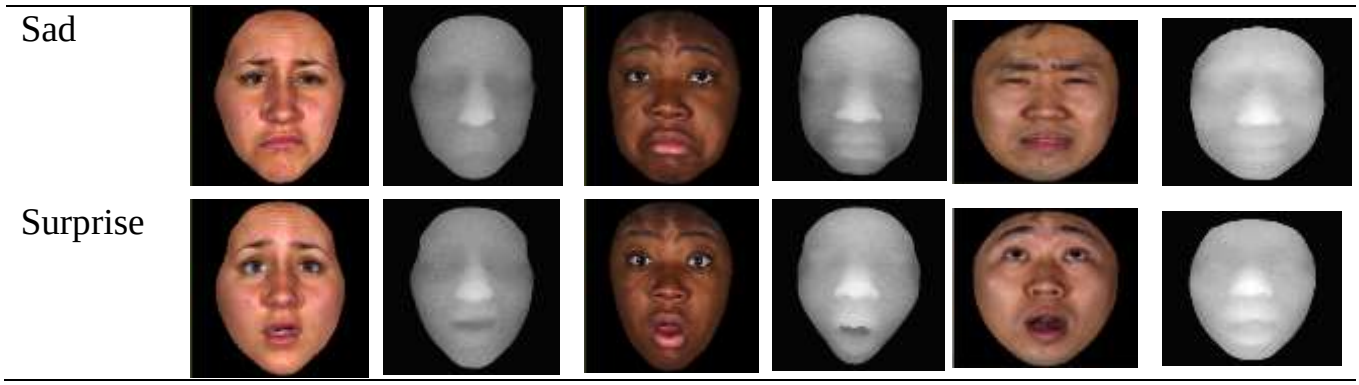


Figure 3.1 Sample Texture Image with generated depth image

3.2.3 Landmark Detection Technique

In this thesis before feature extraction facial landmark detection is performed to extract facial parts. For the proposed method, before feature extraction, facial landmarks are detected using dlib [25] pre-trained model from textured image and extract facial parts like eyebrows, eye, nose and mouth from both depth and texture images by localizing detected landmarks from the textured image on the depth image. Dlib method used to detect facial landmarks is trained by 300-W, AFLW public datasets. The following image shows a landmark detected using the dlib pre-trained model.

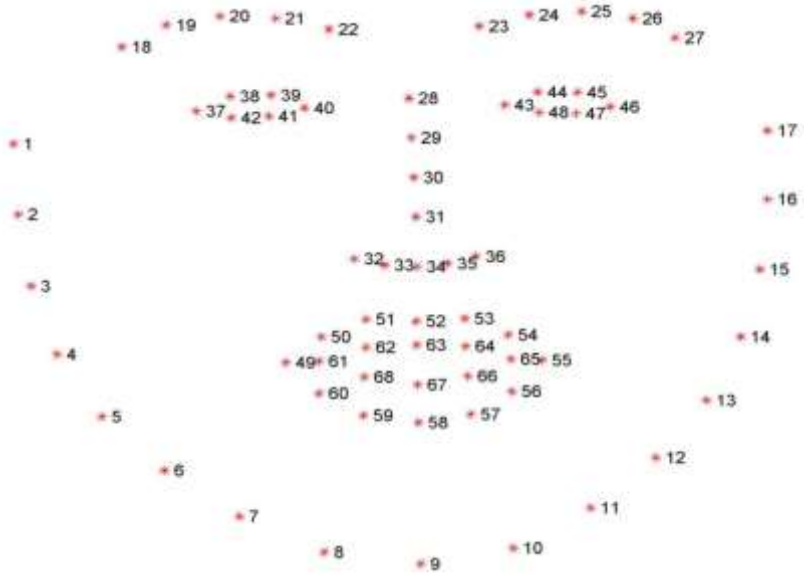
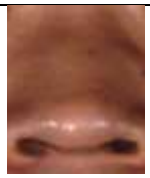
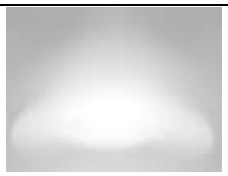


Figure 3. 2: Index of facial landmark

As the above image shows landmark from 1-17 represent yaw of the face, and landmarks from 18-22 represent right eyebrow from the face and 23-27 represent left eyebrow when right eye represented by landmarks from 37-42, left eye represented by landmarks from 43-48, from face nose part is represented by landmarks from 28-36 and mouth represented by landmarks from 49-68. In the proposed system to extract both eyebrows together, landmarks from 18-27 are used. similarly, to extract both eye parts together, landmarks from 37-48 are used. To extract nose and mouth facial parts original landmarks are used. Finally, all facial parts extracted from textured and depth images are reshaped to 100x50 size and stored to a specific directory as a new dataset to use to train and test the proposed classifier.

Table 3. 1: Extracted Face part from texture and depth image

No	Image	Texture Face part	Depth Face part	Landmarks (AU)	Name
1				AU (49-68)	Mouth
2				AU (18-27)	Eye brow
3				AU (37-42 and 43-48)	Eye
4				AU (28-36)	Nose

3.2.4 Feature Extraction Technique Selection

After the facial part extracted, Pre-trained Convolutional Neural Network (CNN) is used to extract features from both depth and textured face parts. As a method there are many different

types of pre-trained CNN model is available publicly. As an example, VGG-16, VGG-19 [14], InceptionV3, Xception [15] and ResNet50 [14] are some.

In order to select a feature extraction technique, different experiments are conducted. The following tables present the result of the experiments based on the feature extraction time (3.3) and the accuracy of the extracted features on the Linear Regression (LR) model.

In the following table, 3.3 shows detail information about the pre-trained model from the angle of feature extraction time and table 3.4 shows the performance of extracted features with the LR model on core i3, 4GB ram, and Windows 10 operating system working environment.

Table 3. 2: Time taken to extract feature from a single facial part

No	**Pre-trained model (1)**	**Feature Extraction time (Minute)**
1	VGG-16	0.020
2	VGG-19	0.021
3	XceptionV3	0.059
4	Resnet50	0.060

Table 3. 3: Accuracy of the extracted feature on LR model without feature fusion

No	**Pre-trained model**	**Feature testing method**	**Reduction method**	**Accuracy**
1	VGG-16	LR	PCA	56.8%
			_	50.8%
2	VGG-19	LR	PCA	51.6%
			_	75.2%
3	Resnet50	LR	PCA	79.6%
			_	76.0%
4	XceptionV3	LR	PCA	59.2%
			_	55.3%

As discussed in table 3.3 and 3.4 Resnet50 pre-trained model is better than other models in terms of feature testing accuracy by LR classifier with Gaussian kernel. 90% of the feature is used for training and 10% for testing using a computer vision dataset splitting convention. In feature testing, feature reduction technique also used as one parameter and resnet50 pre-trained model is performed better than other pre-trained models after feature reduction implemented on the extracted feature. Finally, depending on the result from table 3 and table 4, the resnet50 pre-trained model is selected as a feature extraction technique for the proposed solution.

3.2.5 Dimension Reduction Method

Dimensionality reduction is a statistical technique used to extract important pixel information from the original image. To combine/fusion features extracted from each face parts deep fusion is used. When two or more features are combined, they provide features with a high dimension. To change the dimension from high-level to low-level dimension different methods like PCA [8], ICA [7], FA used, but, from all methods, PCA performed well on the dataset.

PCA has different steps to reduce dimension. First, it takes all features in the dataset consisting of d-dimension samples ignoring the class labels then calculates the d-dimensional mean vector by computing the mean for every dimension of the dataset.

$$\mathbf{m} = \mathbf{1}/\boldsymbol{n} \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{n}} \boldsymbol{x_k}$$ Mean of d-dimension vector

After the mean of d-dimension vector is calculated then compute scatter matrix or covariance matrices for the whole dataset of features

$$\mathbf{S} = \mathbf{1}/\boldsymbol{n} \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{n}} (\boldsymbol{x_k} - \mathbf{m})\,(\boldsymbol{x_k} - \mathbf{m})^{\boldsymbol{T}}$$ Scatter matrix of d-dimension

Where **m** is the mean vector

As the third step compute eigenvectors ($e_{1,}\, e_{2,} \ldots\ldots\ldots\ldots\ldots\ldots, e_d$) and corresponding eigenvalue ($\lambda_{1,}\, \lambda_{2,} \ldots\ldots\ldots\ldots\ldots\ldots, \lambda_d$)

$$\boldsymbol{\Sigma_v} = \boldsymbol{\lambda_v}$$

Where $\sum$ = Covariance matrix

λ = Eigenvector

v = Eigenvalue

Finally, use d x k eigenvector matrix to transform the samples onto the new subspace where x is a d x 1-dimension vector representing one sample, and y is the transformed k x 1- dimension sample in the new subspace

$$\mathbf{Y} = \boldsymbol{W}^{\boldsymbol{T}} * \mathbf{X}$$

In the proposed solution Principle Component Analyses (**PCA**) used for reduction from high dimensional space to a lower-dimensional subspace. The image features in the reduced dimension space should make clear distinctions between different classes in the dataset [6].

3.2.6 Classification Method

Different classification method provides different performance depending on training data. Classification techniques are divided into two main parts named as supervised and unsupervised classification techniques. In most previous researches on facial expression recognition supervised classification methods like KNN, SVM, LR, and other machine learning algorithms are used for classification. Each classification method has different parameters like kernel type, learning rate, a degree which is used to assign a degree for the polynomial kernel, gamma which is used to assign a coefficient for Gaussian, polynomial and sigmoid kernels. For the proposed approach from supervised classification method LR with multinomial solver which used stochastic average gradient descent are used as classifier method. The performance of SVM with Gaussian kernel and automatic gamma is also evaluated to choose best. To use deep learning as a classifier soft-max classifier is also used with different learning rates, and with two commonly known optimizers, namely stochastic gradient descent (SGD) and Adam are used for classification. The classification method is depending on the classes used in the classification and the features extracted from the specific acquired data. For the proposed approach LR is the method used to classify the expression depending on the input data.

3.2.7 Model Evaluation Method

In this study, the performance of the proposed solution is evaluated based on the Confusion matrix on training and testing dataset and response time for the training and testing dataset. The evaluation implementation is depending on the dataset which is divided into 3 main parts using

computer vision dataset splitting conventions. The evaluation dataset can be divided as 70/30, 80/20 and 90/10 which 70, 80, 90 represent the training dataset and 30, 20, 10 represented testing datasets for each split and tested using 10-ford dataset splitting technique used for the evaluation of the proposed solution. For more detail information on the evaluation of the proposed method is discussed in chapter 6 with its excremental results.

3.2.8 Programming Tools

To achieve the goal of the study we have to use programming tools for writing the code and data acquiring. The programming tools and library planned to use are as follow:

- **Jupyter Notebook**: - is a web-based application that helps to configure, load python API and write python code, it also provides help for each method used in python code and to plot accuracy, confusion matrix and other information needed to write proposed method results.
- **Anaconda**: - is an application used to install python programming language with its all modules depending on the version of python and it is helpful to configure different environments depending on the experiment implemented in the proposed method. Additionally, anaconda provides anaconda navigator application to view different environments with their applications like Jupiter notebook, spider, vs-code, and modules installed in the environment.
- **MeshLab**: - is an application used to edit and change the format of 3D data. Using this application used for the pre-processing 3D model by filling holes and removing the noises. In the proposed method, this application mainly used to generate depth images from each 3D dataset samples.
- **TensorFlow**: - is an open-source library(framework) which is used to implement the deep learning and improved high abstraction, it's easy to get API of TensorFlow than other frameworks. TensorFlow is deployed on CPU and GPU which means it is possible to configure it on the Computer without using the graphics card, in case the system needs high processing performance it is configured on the graphics processing unit. As a conclusion, it's easy to deploy and learn from other open-source libraries.

- **Python**: - do prototyping with Python (easier to setup environment, implement algorithm, make proof-of-concept) which means it is faster than C^{++}.
- **Microsoft Office Word and Visio**: - Ms-office is a tool used to write the thesis paper and Visio is used to design the proposed method.
- As the last programming tools in this thesis, the drivers used to configure some device with the computer is used

Chapter Four

4 Proposed Facial Expression Recognition using Feature Fusion

4.1 Overview of Proposed Solution

In order to achieve described problems in previous chapters some methods are used in each stages and in this chapter each proposed methods are discussed in detail, which means methods proposed in each step from landmark detection method until classification methods are discussed in detail and the integration between each proposed method is described using framework and each part in framework are discussed either using diagram or algorithm, pseudocode, and flowcharts. Figure 4.1 discuss in deep about methods used in each stage and follow of proposed solution and data used in each stage for the specified problem.

4.2 The pipeline of Proposed Solution

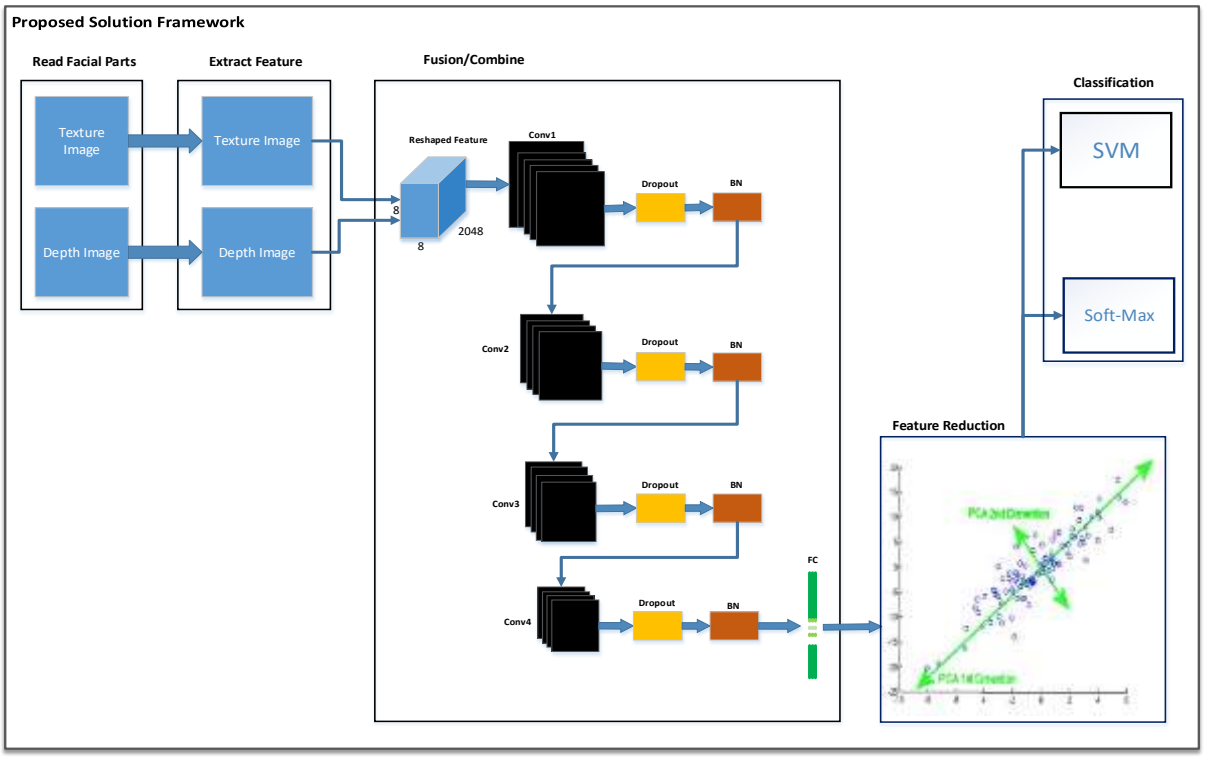


Figure 4. 1: Framework of the proposed solution

4.2.1 Facial Parts Reading

Before reading, face parts for each sample in the dataset facial parts are extracted using the proposed method and stored as a new dataset for the training and testing the prosed model. To extract facial parts proposed method take two input images which are textured image and depth image with (512 x 512 x 3) same size both images, where first 512 shows width and next shows height and 3 color dimension in for each sample. In facial part extraction dlib landmark detecting algorithm is used to detect and localize face parts on both textured and depth image, after facial parts are localized each part from both images are cropped and resized in (50 x 100 x 3), where 50 specifies the size of width and 100 specifies the height and 3 specifies color dimension for each part and finally each facial parts are stored as new dataset for training and testing the proposed solution. The following diagram shows the facial part extraction step in detail.

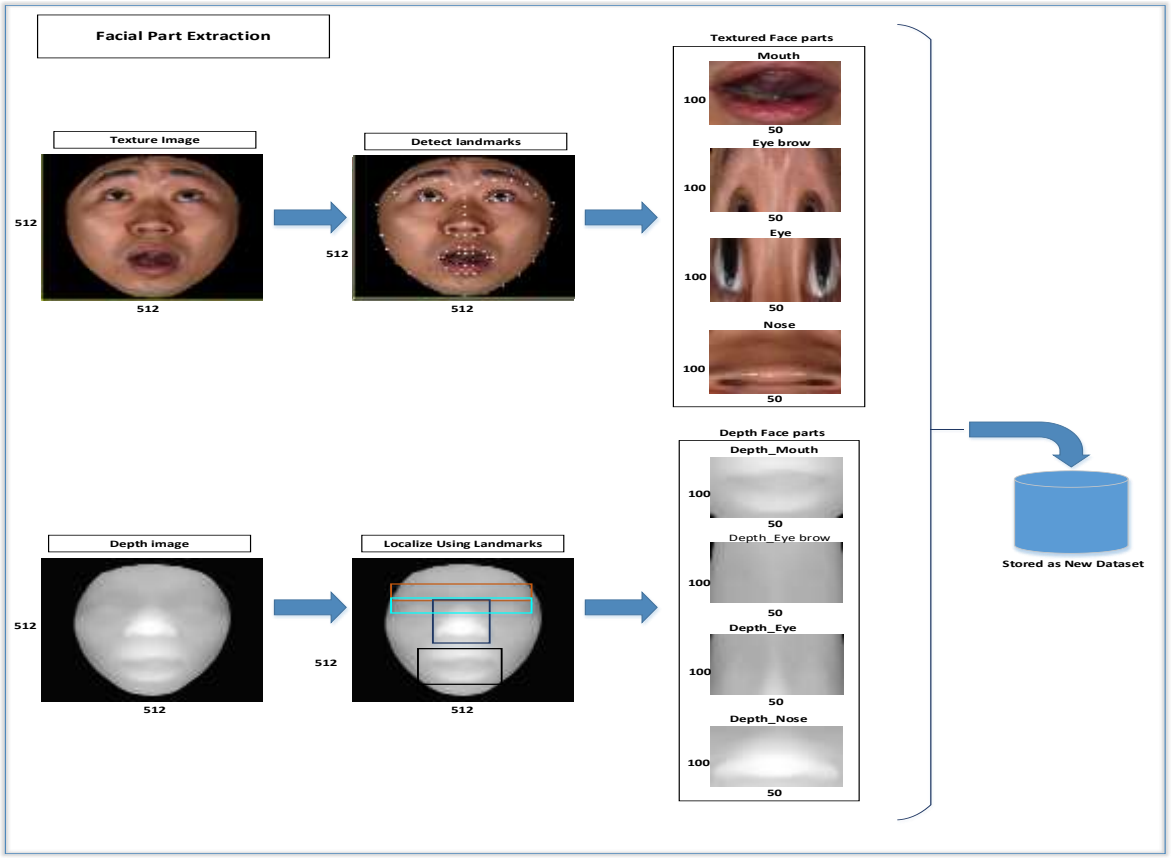


Figure 4. 2: Facial part extraction diagram

4.2.1.1 Pseudo Code for Facial Part Extraction

1. *Start*
2. *Read n number of texture image assign to **image***
   *Read n number of depth image assign to **depth***
3. *Initialize **i** = 1*
4. *While **i** <= len (Landmarks)*
    a. *Texture_image = image[i]*
       Depth_image = depth[i]
    b. *Landmarks = detect landmark (Texture_image)*
    c. *While **a** <= len (Landmarks)*
        i. *If (a = 49)*
            1. *J = 68*
                a. *Add neighbour pixel to detected landmarks **X**, **Y** coordinates*
                   X – 30 to left side
                   Y – 20 to up side
                   (X+J) + 30 to right side
                   (Y+J) + 20 to down side
            2. *Texture_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on Texture_image*
               Depth_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on depth_image
            3. *Mouth_Texture_crop = crop (Texture_loc)*
               Mouth_depth_crop = crop (depth_loc)
            4. *Write Mouth_Texture_crop[i]*
               Write Mouth_depth_crop[j]
        ii. *If (a = 18)*
            1. *J = 27*
                a. *Add neighbour pixel to detected landmarks **X**, **Y** coordinates*
                   X – 0 to left side
                   Y – 20 to up side
                   (X+J) + 0 to right side
                   (Y+J) + 0 to down side
            2. *Texture_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on Texture_image*
               Depth_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on depth_image
            3. *Eyebrow_Texture_crop = crop (Texture_loc)*
               Eyebrow_depth_crop = crop (depth_loc)
            4. *Write Eyebrow_Texture_crop[i]*
               Write Eyebrow_depth_crop[j]
        iii. *If (a = 37)*
            1. *J = 48*
                a. *Add neighbour pixel to detected landmarks **X**, **Y** coordinates*
                   X – 30 to left side

Y – 20 to up side
(X+J) + 30 to right side
(Y+J) + 20 to down side

2. *Texture_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on Texture_image*
   Depth_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on depth_image
3. *Eye_Texture_crop = crop (Texture_loc)*
   Eye_depth_crop = crop (depth_loc)
4. *Write Eye_Texture_crop[i]*
   Write Eye_depth_crop[j]

iv. *If (a = 28)*

1. *J = 36*
   a. *Add neighbour pixel to detected landmarks **X**, **Y** coordinates*
      X – 10 to left side
      Y – 40 to up side
      (X+J) + 10 to right side
      (Y+J) + 0 to down side
2. *Texture_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on Texture_image*
3. *Depth_loc = Localize X, Y coordinates from (X:Y) to cordinates (X+J:Y+J) on depth_image*
4. *Nose_Texture_crop = crop (Texture_loc)*
   Nose_depth_crop = crop (depth_loc)
5. *Write Nose_Texture_crop[i]*
   Write Nose_depth_crop[j]

v. *Increment **a** by one*

d. *Increment **i** by one*

4.2.2 Feature Extraction

As shown in figure [4.1] framework of proposed system feature extraction is one layer of the proposed solution, which takes extracted facial parts as an input and provides extracted vector as an output that can be feed to fusion/combination layer. In proposed solution for feature extraction from facial parts Resnet50 pre-trained model is used. Resnet50 [14] pre-trained model is able to extract image with [‘?’, ‘?’, 3], the first and second ‘?’ specifically indicate any size of width and height which is specified depending on the input image and 3 indicate color dimension for input image. an output dimension of resnet50 pre-trained model is [‘?’, ‘?’, 2048], as the same to input, the first two ‘?’ of output depend on the width and height size of the input image.

in proposed solution pre-trained model take an input image with size (50, 100, 3) for each face parts and provide NumPy array with [1, 2, 4, 2048] for each image parts, these indicate for single expression image have [8, 2, 4, 2048] where 8 shows the number of facial parts and 2048 indicate the dimension of the output NumPy array. The next step of this layer can be done in the fusion layer by combining 8 facial parts together using concatenate () deep learning method. The following diagram shows detail about extracting features.

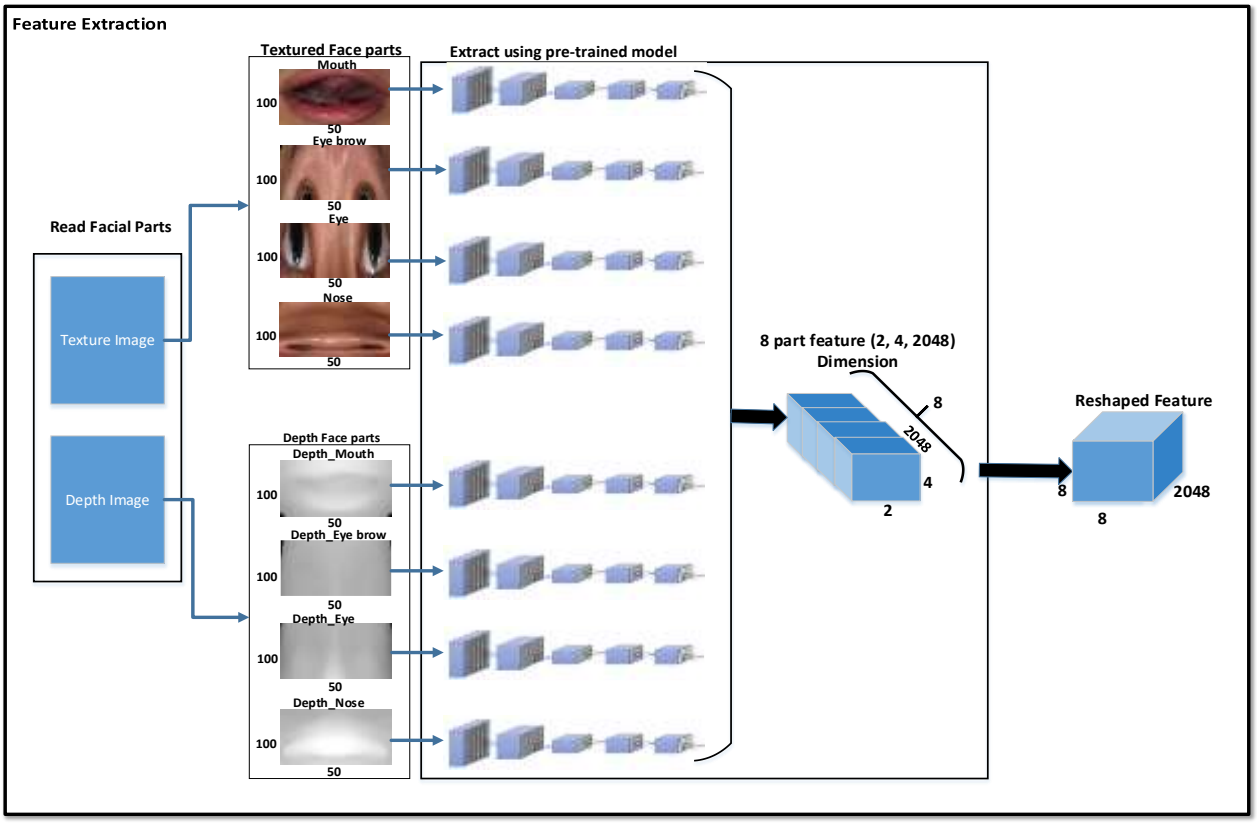


Figure 4. 3: Feature extraction diagram

4.2.2.1 Pseudo Code for Feature Extraction from Face Parts

1. *Read n number of face parts as face part*
2. *Features = [ ], Initialize **I** = 1*
3. *Label = split decretory path (-2)*
4. *Model = Load Resnet50 pre-trained model*
5. *While **I** <= len(x)*
   i. *New_feature = Extract feature of face_part[i]using Model*
   ii. *Append New_feature and Label to Features*
   iii. *Increment **I** by one*
6. *Write features as CSV file*

4.2.3 Feature Fusion

Currently feature fusion [5] is used to combine features extracted using two or more models from the same data also used to combine features extracted from different data using one extraction model. In the proposed solution feature extracted from different data using a resnet50 pre-trained model where each data has a great contribution to the target value. Thanks to Keras concatenating module features extracted from each facial part are combined together easily. After concatenating extracted feature concatenated features are reshaped from [8, 2, 4, 2048] to [8, 8, 2048] because it is easy to apply convolution, polling and others deep learning methods. In this layer 4 convolutions are applied to reshape feature with different kernel size and number of filters, in first convolution (4,3) kernel size and 1024 number of filters are used, from second to last convolution (1,1) kernel size is used and 512 number of filters second convolution, 256 number of filters for third convolution and 128 number of filters used for last convolutional layer. In each convolutional layer dropout with (0.25) value and batch-normalization are used to prevent overfitting problems because a number of datasets for neutral extraction type are not balanced with a number of others expression datasets. After the fourth convolutional layer, new features with [8, 8, 128] shape is the final output of the convolutional layer in the fusion step. After the last convolutional layer output, new features can be changed to vectors by using the flattening method which provides new shape [1, 8192]. By changing the kernel size used in the convolution it is possible to increase or decrease the parameters used to train in deep learning. The following diagram shows some discussion in feature fusion. The implementation of the feature fusion layer discussed in chapter five in detail.

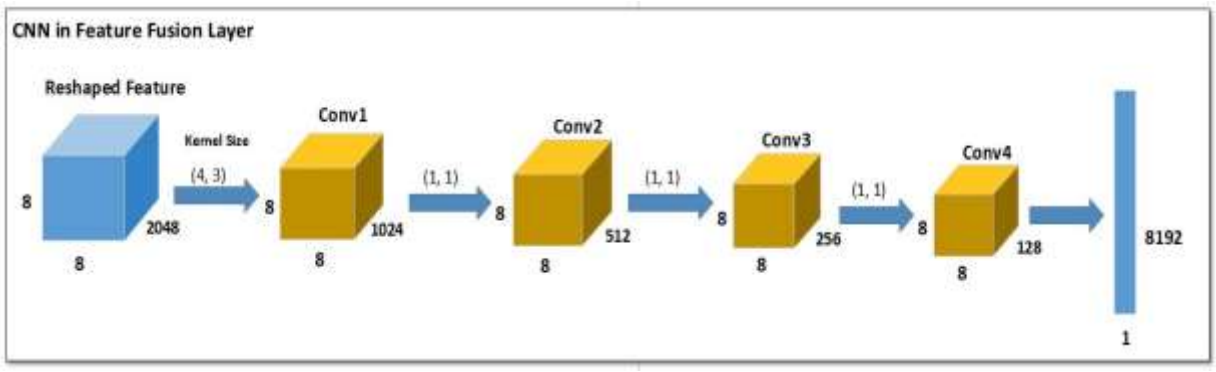


Figure 4. 4: CNN Diagram in Feature Fusion Layer

4.2.4 Dimension Reduction

To improve the recognition performance feature with a high dimension must be redacted to lower dimension space. In the case of dimension reduction proposed solution used PCA which is a better dimension reduction method for continuous-valued datasets. As discussed in chapter 3 in subtitle 3.2.5 in detail principle component analysis used covariance matrix, eigenvector, and eigenvalue to calculation the coloration between the attributes and target value, which means PCA takes attribute which has a high variance to the target value as the first attribute and take next higher attribute as the second attribute from attributes. In propose solution after final fully connected layer 99(%) of fully connected layer features are reduced using principal component analysis method.

4.2.4.1 Pseudo Code for Dimension Reduction

1. *Read n number of fusion features (1, 8192)*
2. *Assign (99%) of fusion features as number component*
3. *Compute d-dimension mean vector for all training dataset/ features*
4. *Compute covariance matrix for all features*
5. *Compute Eigenvector and corresponding Eigenvalues*
6. *Sort Eigenvectors by decreasing Eigenvalue and choose Eigenvector with large Eigenvalue.*
7. *Use new dimensional matrices to transform the samples onto new subspace*

4.2.5 Classification

In most FER researches hand-craft classification method is used because the soft-max classification technique needs a large number of datasets in order to provide better accuracy. In the case of proposed solution SVM with Gaussian kernel, automatic learning rate, and default epoch are used as a classification method. Kernel type in SVM is used to specify the classification type in training, where classification types are binary and multi-class classification. The proposed classification method is better on preventing overfitting than other methods.

In addition to the SVM classification method, LR classification method with ‘lbfgs’ solver, a multi-class attribute with multinomial value is initialized with a max-iter attribute which

specifies an epoch for the classifier is used. finally, for comparison purposes, Soft-Max is used. all the above classifiers are taken reduced feature of training dataset with its target value and after training, store model as an output. By loading the stored model evaluate the performance of the proposed solution using a reduced testing set. Finally, compute some performance evaluation matrices.

Chapter Five

5 Implementation of Facial Expression Recognition Method

5.1 Overviews

To achieve the goal of the proposed solution, methods in each part of a proposed solution must be implemented using some specific environment with a specific dataset. The proposed solution has more than 6 different stages, but mainly 5 stages of proposed solution from facial part extraction to classification going to discussed in this chapter, which is not including data pre-processing and depth image generation. As discussed in chapter 3 in heading 3.2.3 and chapter 4 heading 4.2.1 in detail facial part extraction stage of the proposed solution is the stage that prepares the dataset for the proposed solution in addition to facial part extraction and localization. To train the proposed solution 90% used from the whole dataset which means 2250 samples are used and for testing 250 samples are used from the dataset. in training and testing the proposed solution method used in different stages must be implemented so, through training and testing implementation of facial part extraction, dimension reduction, feature fusion also passed.

5.1.1 Working Environments

- Desktop Computer
    - Processor: Intel(R) Core (TM) i5-6500 CPU @ 3.20GHz, 3192 Mhz, 4 Core(s), 4 Logical Processor(s)
    - Installed Memory: Physical Memory (RAM) 8.00 GB
    - OS: Microsoft Windows 10 Pro, x64-based PC, Version 10.0.17134 Build 17134
    - System Model: - Dell OptiPlex 3040
- Application and IDE for programming languages
    - Application
        - Anaconda: - is an application used to install the latest version of python with its different module and IDEs, for implementing the proposed solution an anaconda application version 1.9.7 with 64-bit support used. For detail, information read chapter 3 subsection 3.2.7.

        - MeshLab: - in an application used to edit the 3D objects, change the format of an object and used to generate depth image from the 3D object file. For implementation latest version of this application is used which is V2016.12 with 64-bit support.
    - IDE
        - Jupyter Notebook: - is an IDE that is the most popular and recommended IDE to researchers to edit python code, to visualize some diagram, training and testing charts and processing progresses. To implement the proposed solution Jupiter notebook with 6.0.0 version is used which is installed through anaconda installation.
        - Visual-Studio-Code: - is an IDE that is easily configured with different python environments and used to edit python code, c++ code and other programming language code also editable using this IDE. A version 1.38.1 with community and 64-bit support is installed for implementation in addition to Jupiter notebook.
- Programming Language
    - Python: - to implement the proposed solution python programming language with version 3.5 is used because the dlib module does not install properly on the latest version of python.
        - The module used in python most necessary for implementation
            - Keras with version 2.2.4 which used to load the pre-trained model, used to apply convolution, pooling, and other deep learning processes.
            - Tensorflow with version 1.12.0 and above used to connect pre-trained model and feature fusion model graphs
            - Tensorboard with version 1.12.0 and above to visualize training progress in deep learning, in the proposed approach to visualize the feature fusion progress.
            - Seaborn with version 0.9.0 used to plot confusion matrix of testing data and used to plot the accuracy for each target class.

5.2 Read Face Parts

The starting step of the proposed solution is reading texture image and depth image from the dataset and extract face parts from both texture and depth images. For more detail information about the facial part extraction technique please read chapter 3 subtitle 3.2.3 which is discussing each step in deep. In this chapter, we discussed the code implementation of a facial part in detail. The face must be detected before landmarks detected from the input image. for detecting face from input image deep CNN based method is used. the following images show the performance of landmark detection with face detection where facial landmark detection depends on face detection performance.

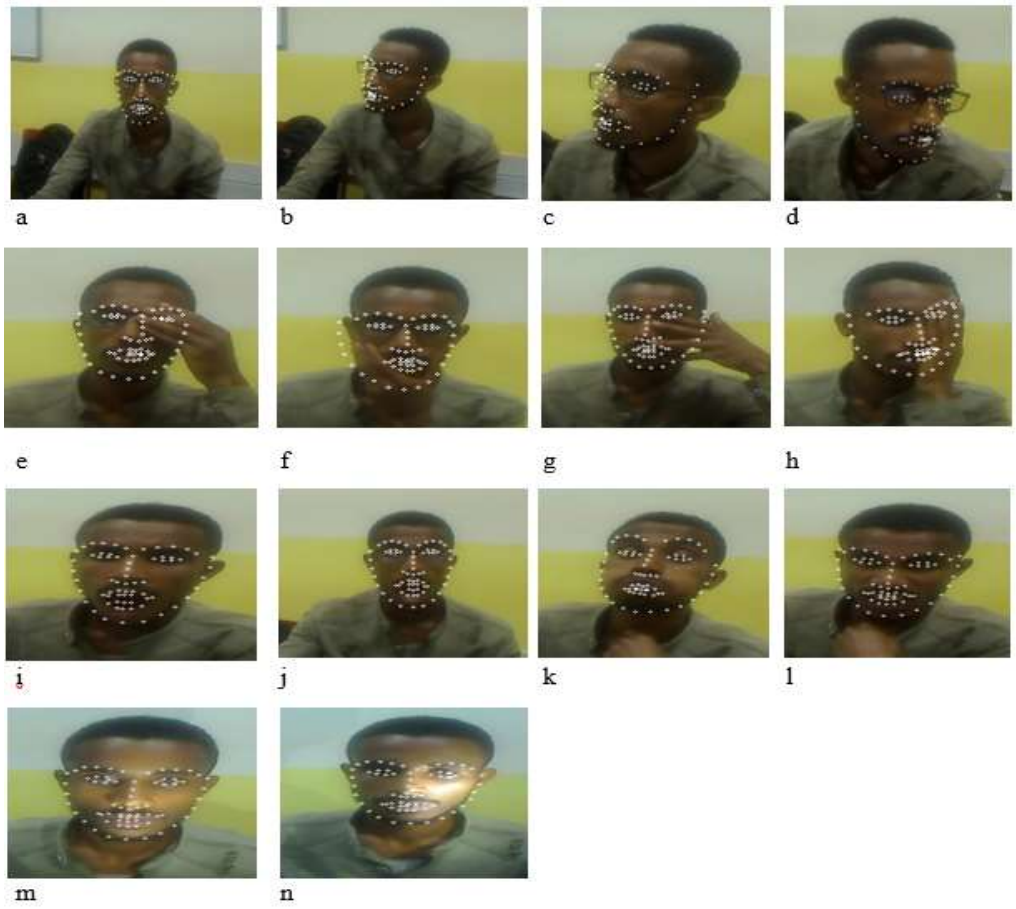


Figure 5. 1: Performance of landmark detection method

In figure 4.5 image **a-d** shows the performance of the CNN face detector and performance of the dlib landmark detector in a different pose, yaw, and angle. Image **e-h** show the performance in different occlusion and image **i-l** show the performance in different expression and same expression with different camera distance showed using image **i** and **j**. image **m** and **n** show the performance in different illumination. the following figure show sample code for face detection and landmark detection.

```
# import the necessary packages
from imutils import face_utils
import imutils
import time
import dlib
import cv2
# initialize dlib's face detector (CNN-based) and then create
# the facial landmark predictor
print("[INFO] loading facial landmark predictor...")
detector = dlib.cnn_face_detection_model_v1("mmod_human_face_detector.dat")
predictor = dlib.shape_predictor(args["shape_predictor"])
image = cv2.imread('M0043_SU04AE_F2D.bmp')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# detect faces in the grayscale frame
rects = detector(gray, 0)
# determine the facial landmarks for the face region, then
# convert the facial landmark (x, y)-coordinates to a NumPy array
shape = predictor(gray, rect)
shape = face_utils.shape_to_np(shape)
# loop over the (x, y)-coordinates for the facial landmarks
# and draw them on the image
for (x, y) in shape:
    cv2.circle(image, (x, y), 3, (255, 255, 255), 1)
# show the image with landmark
cv2.imshow("Land Marked_Image", image)
cv2.imwrite("All_land.jpg", image)
cv2.waitKey(0)
```

Figure 5. 2: A sample code to detect landmark

After landmark detected the next step is to extract facial parts from both texture and depth image. to extract facial parts from depth image, localize each facial part on depth image is helpful to extract the equivalent depth facial part. the following sample code show how to extract feature from both texture and depth images for a single facial part, where i and j are shown x and y coordinate of detected shape.

```
In [12]: # load the input image, resize it, and convert it to grayscale
         image = cv2.imread('images/F0052_FE04WH_F2D_texture_new.jpg')
         depth = cv2.imread('images/F0052_FE04WH_F2D_depth_new.jpg')
```

```
In [13]: if i == 17:
             j = 27
             (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
             roi = image[y:y + h, x:x + w]
             depth_roi = depth[y:y + h, x:x + w]
             roi = imutils.resize(roi, width=50, height=100, inter=cv2.INTER_CUBIC)
             cv2.imwrite('eye_brow_part1'+ str(classes) + str(im_num) + '.jpg', roi)
             cv2.imwrite('depth_eye_brow_part1'+ str(classes)  + str(im_num) + '.jpg', depth_roi)
```

Figure 5. 3: A sample code to extract facial part

5.3 Feature Extraction Steps

The second step to implement the proposed solution is feature extraction from extracted face parts in face parts extraction. to show the step of feature extraction from both texture and depth images using the proposed pre-trained model, the Sample feature extracted from single sample facial parts are used as the input dataset. As discussed in detail in chapter 4 in subtitle 4.2.2 feature extracted from each facial part using Resnet50 pre-trained model. The following same code show detail feature extraction steps from facial parts.

A model proposed to extract features from both texture and depth facial parts is part of the Keras module, so to import and used the model the following sample code is used.

```
In [1]: from keras.applications import ResNet50
        model = ResNet50(weights="imagenet", include_top=False)

        Using TensorFlow backend.
```

Figure 5. 4: A sample code to import model

After the model loaded successfully features extracted from texture image before image passed to the model to extract feature, first image reshaped to 50 x 100 width and height, the following sample code show feature extraction from texture facial part.

```
In [6]: eye_brow = cv2.imread('good/croped_Land mark image23.jpg')
        eye_brow = cv2.resize(eye_brow, (100, 50))

In [7]: eye_brow = img_to_array(eye_brow)
        eye_brow = np.expand_dims(eye_brow, axis=0)
        eye_brow = imagenet_utils.preprocess_input(eye_brow)
        eye_brow.shape
Out[7]: (1, 50, 100, 3)

In [8]: eyeb_predicted = model.predict(eye_brow)
        eyeb_predicted.shape
Out[8]: (1, 2, 4, 2048)
```

Figure 5. 5: A sample code to extract feature from texture part

After features extracted from textured facial parts, the next step is to extract features from depth facial parts using the same step used to extract from the texture facial part. the following code show detail how to extract feature from depth facial part.

```
In [9]: debth_eyeb = cv2.imread('good/depth_croped_Land mark image23.jpg')
        debth_eyeb = cv2.resize(debth_eyeb, (100, 50))

In [10]: debth_eyeb = img_to_array(debth_eyeb)
         debth_eyeb = np.expand_dims(debth_eyeb, axis=0)
         debth_eyeb = imagenet_utils.preprocess_input(debth_eyeb)
         debth_eyeb.shape
Out[10]: (1, 50, 100, 3)

In [11]: debth_eyeb_predicted = model.predict(debth_eyeb)
         debth_eyeb_predicted.shape
Out[11]: (1, 2, 4, 2048)
```

Figure 5. 6: A sample code to extract feature from depth part

5.4 Feature Fusion Step

After feature extraction in the previous step is done, the next step is to combine features extracted from each facial part. To combine or fusion feature in deep learning concatenate module or package is used. The requirement for this module is the dimension of combined feature or models must be equal, which mean in case of the proposed method a feature dimension for each facial part must be equal. The detail about feature fusion is discussed in chapter four subtitle 4.2.3 with a size of input dimension and size of the output dimension after applying some convolution and flattening.

In feature fusion steps the first step is to create an input layer for the texture facial part that is used as input to the fusion model with input size which is similar with the dimension of an extracted feature using a pre-trained model, the following sample code shows the creation of input layer for each facial part.

```
In [186]: # Create model to read texture image of facial part
          mouth_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='mouth_input')
          eye_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='eye_input')
          eye_brow_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='eye_brow_input')
          nose_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='nose_input')
```

Figure 5. 7: A sample code to create a model that read texture image

After the input layer for texture image is created the next step is to create an input layer for depth facial parts, which uses the same code but, the name of the input layer must be different from the previous step as shown in the following sample code.

```
In [187]: # Create model to read depth image of facial part
          depth_mouth_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='depth_mouth_input')
          depth_eye_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='depth_eye_input')
          depth_eye_brow_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='depth_eye_brow_input')
          depth_nose_input = Input(shape=(2 , 4 , 2048), dtype='float32', name='depth_nose_input')
```

Figure 5. 8: A sample code to create a model that read depth image

The next step in feature fusion combines all created input layers together and form one feature which has dimension with the sum of all input layers. The following code shows how to combine all input layers.

```
In [188]: # Combine all facial part model
          ALL_Method = concatenate([mouth_input, eye_input, eye_brow_input, nose_input,
                                    depth_mouth_input, depth_eye_input, depth_eye_brow_input, depth_nose_input],
                                    name='all_models')
```

Figure 5. 9: A sample code to combine the input model

After the combination of the input layer, creating a new model by using combined layers and extract features from both texture and depth facial parts using a new model. The following sample code show in detail how to create a new model and extract feature using the combined model.

```
In [190]: # Crate new model using combine model
          fusion = Model(inputs = [mouth_input, eye_input, eye_brow_input, nose_input,
                                   depth_mouth_input, depth_eye_input, depth_eye_brow_input, depth_nose_input], outputs = ALL_Method)

In [192]: # Extract feature using new combined model
          new_feature = fusion.predict([np.array(mouth_predicted), np.array(eye_predicted),
                                        np.array(eyeb_predicted), np.array(nose_predicted),
                                       np.array(debth_mouth_predicted), np.array(debth_eye_predicted),
                                        np.array(debth_eyeb_predicted), np.array(depth_nose_predicted)])
```

Figure 5. 10: A sample code to create a new model and extract feature

After feature extraction using a new combined model the next step is to reshape extracted features and create a new input layer for convolutional model input. The following code shows how to reshape new features and create a new input layer.

```
In [194]: # Reshape newly extracted feature
          new_feature = np.reshape(new_feature, (1,8,8,2048))

In [195]: # Display reshaped feature dimension
          new_feature.shape

Out[195]: (1, 8, 8, 2048)

In [196]: # Create final fusion model
          # Create input layer that takes reshaped new feature
          part_input = Input(shape=(8, 8, 2048), dtype='float32', name='part_input')
```

Figure 5. 11: A sample code to reshape and create input layer

The next step is applying 4 stage convolutions with different kernel sizes and flattening on the last layer of the convolution apply to flatten. In each convolutional layer dropout and batch-normalization are used to prevent overfitting. The following code show detail about implementation in this step.

```
In [197]: # Create convolutional, Batchnormalization, Dropout and Flatten layers
          # Convolution 1
          conv_1 = Conv2D(1024, (4,3), padding="same", activation='relu', name = 'final_conv1')(part_input)
          conv_1 = Dropout(0.25)(conv_1)
          conv_1 = BatchNormalization()(conv_1)
          # Convolution 2
          conv_2 = Conv2D(512, (1,1), padding="same", activation='relu', name = 'final_conv2')(conv_1)
          conv_2 = Dropout(0.25)(conv_2)
          conv_2 = BatchNormalization()(conv_2)
          # Convolution 3
          conv_3 = Conv2D(256, (1,1), padding="same", activation='relu', name = 'final_conv3')(conv_2)
          conv_3 = Dropout(0.25)(conv_3)
          conv_3 = BatchNormalization()(conv_3)
          # Convolution 4
          conv_4 = Conv2D(128, (1,1), padding="same", activation='relu', name = 'final_conv4')(conv_3)
          conv_4 = Dropout(0.25)(conv_4)
          conv_4 = BatchNormalization()(conv_4)
          # flatten layer
          flat = Flatten()(conv_4)
```

Figure 5. 12: A sample code to apply convolution

Finally, create a model using a flattened layer and used a new model to extract features from features extracted using a combined model and display the final dimension of the extracted feature. The following code shows the final progress of feature fusion.

```
In [39]: # Creaate model using final fusion model
         last_fusion = Model(part_input, flat)

In [41]: # Extract feature using lasf fusion
         final_feature = last_fusion.predict(new_feature)

In [42]: # Display final_feature dimansion
         final_feature.shape

Out[42]: (1, 8192)
```

Figure 5. 13: A sample code to extract feature by fusion model

5.5 Dimension Reduction Step

In the proposed solution PCA is used as a method to redact features from high dimension space to low dimension space. To get a final redacted feature using the proposed technique, more than two most important steps must be pasted. As discussed in chapter 3 in subtitle 3.2.5 in detail with its mathematical description, before new feature using a new dimension matrix, mean for all dimension, covariance matrix, engine value, engine vector must be calculated.

The implementation of dimension reduction steps is so easy, using a sklearn which is a python package or module, by importing Standard Scaler and PCA, as shown in the following sample code.

```
In [73]: from sklearn.preprocessing import StandardScaler
         from sklearn.decomposition import PCA
```

Figure 5. 14: A sample code to import modules

After importing both the python module fit a dataset to standard scaler and then apply a transformation on both training and testing dataset splits, the following code shows the implementation of standard scaler on training and testing dataset.

```
In [74]: scalar = StandardScaler()
         scalar.fit(x_train)

Out[74]: StandardScaler(copy=True, with_mean=True, with_std=True)

In [75]: scalar_train = scalar.transform(x_train)
         scalar_test = scalar.transform(x_test)
```

Figure 5. 15: A sample code to transform using standard scaler

After transformation applied on training and testing dataset, the next step is use PCA to reduce the dimension space depending on specified number of component or present for number of components and using number of component PCA reduce dimension from high dimension space to low dimension space, the following code show detail how to implement PCA for feature reduction.

```
In [76]: pca = PCA(.99)
         pca.fit(scalar_train)
         print('PCA components = :', pca.n_components)

         PCA components = : 0.99

In [77]: pca_train = pca.transform(scalar_train)
         pca_test = pca.transform(scalar_test)
```

Figure 5. 16: A sample code to redact dimension using PCA

5.6 Classifier Training Step

A classifier proposed for this study is SMV with the Gaussian kernel with an automatic learning rate and uses regularization by specifying the value of C in the classifier initialization step. again, in this step sklearn module is responsible to load SVM handcraft machine learning method from python packages. The following code shows how to load and initialize SVM and LR classifiers.

```
In [5]:  from sklearn.svm import SVC
         from sklearn.linear_model import LogisticRegression
         from sklearn.model_selection import train_test_split

In [118]: svm_clf = SVC(kernel='rbf', C=15, gamma='auto')
          lr_model = LogisticRegression(solver='liblinear', multi_class='auto', max_iter=1000)
```

Figure 5. 17: A sample code to load and initialize classifier

After classifiers loaded and initialized properly next step has fit the classifiers with training data set which have attribute and target value separately and store trained model in the form of pickle file as shown in the following sample code.

```
In [120]: start = time.time()
          lr_model.fit(np.array(x_train),np.array(y_train))
          end = time.time()
          print('\n\n time spend to train LR: ', (end-start)/60, 'minutes \n\n')
```

Figure 5. 18: A sample code to train LR classifier

```
In [119]: start = time.time()
          svm_clf.fit(np.array(x_train),np.array(y_train))
          end = time.time()
          print('\n\n time spend to train SVM: ', (end-start)/60, 'minutes \n\n')
```

Figure 5. 19: A sample code to train the SVM classifier

```
In [17]: # Save the model as pickle file
         svm_model = 'SVM_Model.sav'
         pickle.dump(svm_clf, open(svm_model, 'wb'))
```

Figure 5. 20: A sample code to save the trained model

5.7 Classifier Testing Step

In the proposed solution the classifier tested by split dataset for testing and also tested by selecting a sample dataset that is not selected as a training set. To check the performance of the classifier that is trained on a training data set the first step is load trained model and predict the performance of the proposed solution using the testing dataset, as shown in the following sample code.

```
In [18]: loaded_model = pickle.load(open(svm_model, 'rb'))
         result = loaded_model.score(np.array(x_test), np.array(y_test))
         print(result)

         0.8538461538461538
```

Figure 5. 21: A sample code to load and test trained model

Chapter Six

6 Evaluation, Result, and Discussion

6.1 Result and Discussion

To prove the proposed solution is a better solution for facial expression recognition the result of two mostly used handcraft machine learning techniques are evaluated and from deep learning soft-max classifier with different optimizers, learning rate, decay value, early stop value and parameters is used. the dataset for the evaluation is split using computer vision convention into 70/30, 80/20 and 90/10, all classifier is trained and tested using each split and used 10-ford dataset splitting method to evaluate the performance of the proposed solution.

6.1.1 Landmark Detection and Localization

To train the classifier first step is to extract facial parts from both depth and texture images, so to extract facial part landmark detection is implemented and on texture image and by using the landmark detected from texture image, localization facial parts on both depth and texture images. The following figure shows the detail about landmark detection and facial part localization.

Texture Image

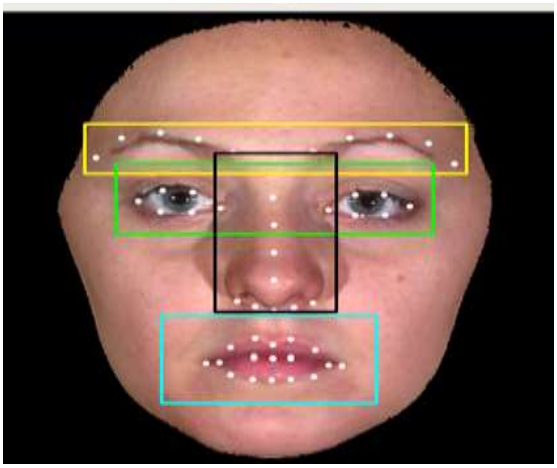


Figure 6. 1: Landmark detection and localization

Depth Image

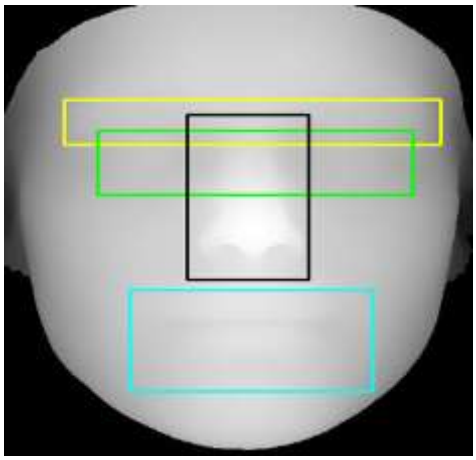


Figure 6. 2: Localization on depth image

6.1.2 Facial Part Extraction

As discussed in the above topic on the landmark detection section, the first step to extract the facial part is landmark detection. After the landmark is detected using the dlib method and localized on both depth and texture images before the facial part extracted for each facial part different neighbour pixels added to prevent information loss when the facial part extracted. For mouth facial part 30 neighbour pixel to left, 20 neighbour pixels to upside, 30 neighbour pixels to right and 20 neighbour pixels to upside are added as its part, for eyebrow 20 neighbour pixels to upside and 0 pixels to other sides are localized as its part. for eye part 30 neighbour pixels to the left side, 30 neighbour pixels to the right side and 20 neighbour pixels to upper and bottom side added as eye part. Finally, for nose facial part 40 neighbour pixels to the upper side, 10 neighbour pixels to left and right side added as nose part. the following image in table 6.1 and 6.2 show result after neighbour pixels added to each facial part for both texture and depth image

Table 6. 1: Localize and Extract Facial part from a texture image

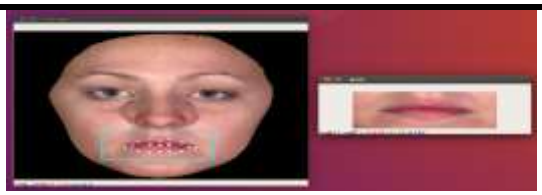
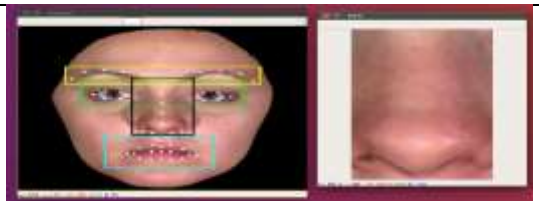
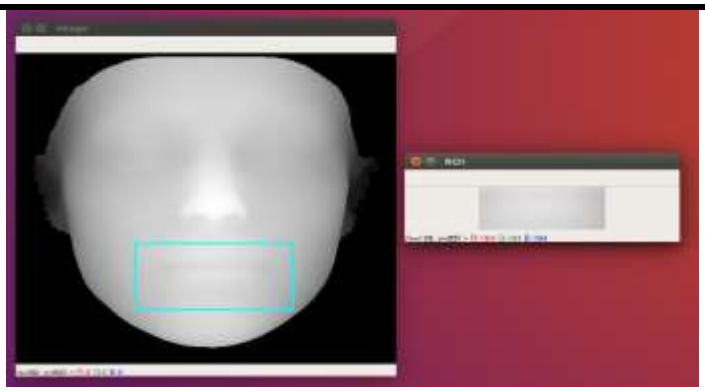
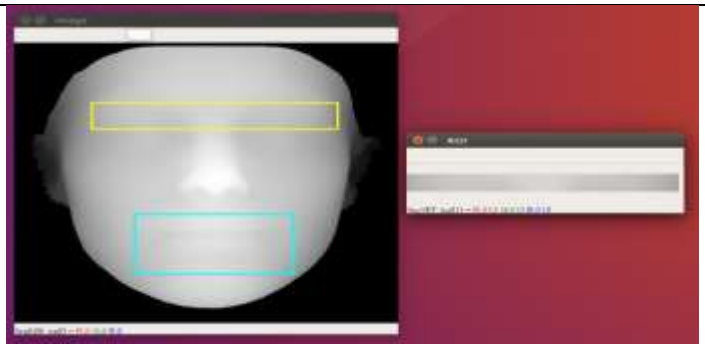
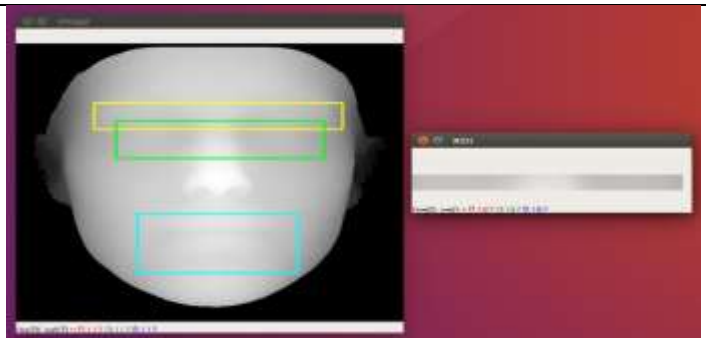
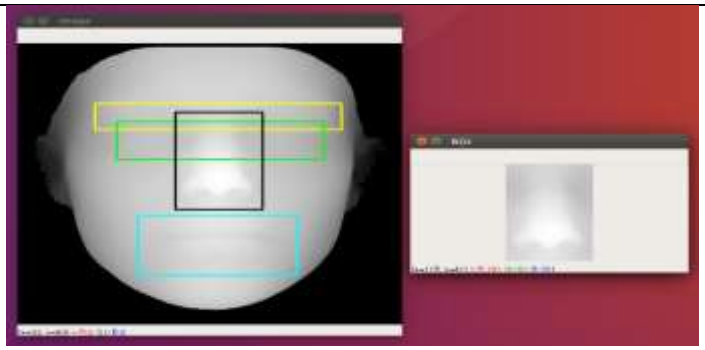
No	Part Name	Localized and Extracted Part
1	Mouth	
2	Eye Brow	
3	Eye	
4	Nose	

Table 6. 2: Localize and Extract Facial part from a depth image

No	Part Name	**Localized and Extracted Part**
1	Mouth	
2	Eye Brow	
3	Eye	
4	Nose	

6.1.3 Feature Extraction

To prove the proposed pre-trained feature extraction method is better than other pre-trained models, features extracted from the dataset using pre-trained models like VGG16, VGG19, Xception and train each classifier using extracted features to check the performance of each model and compare their performance with proposed feature extraction mothed. the following table discusses the input and output shape of each pre-trained feature extraction method.

Table 6. 3: a Pre-trained model with input and output shape

No	**Pre-trained model**	**Input shape**	**Output shape**	**# of Layer**
1	VGG16	[?, ?, 3]	[?, ?, ?, 512]	16
2	VGG19	[?, ?, 3]	[?, ?, ?, 512]	19
3	Xcpetion	[?, ?, 3]	[?, ?, ?, 2048]	71
4	Resnet50	[?, ?, 3]	[?, ?, ?, 2048]	50

In the above table in the input shape column [?, ?, 3], indicate that the first two(?) show the width and height of input image and 3 shows the color channel of input image where two (?) are changed depending on the input image. in output shape column [?, ?, ?, 512] the first (?) show the number of images or batches and second and third (?) represents width and height of output feature, which is dependent on width and height size of the input image. finally, '512' represent the dimension of output shape for each image. the following table shows the performance of the proposed solution by extract features using each pre-trained model.

Table 6. 4: Performance of the proposed solution using different pre-trained extraction model

No	**Pre-trained model**	**Classifier**	**Average Accuracy**
1	VGG16	SVM	48.46%
		LR	78.08%
		Soft-Max	67.39%
2	VGG19	SVM	51.51%
		LR	85.38%

		Soft-Max	66.92%
	Xception	SVM	67.30%
		LR	70.00%
		Soft-Max	42.35%
4	Resnet50	SVM	85.05%
		LR	86.22%
		Soft-Max	84.45%

6.1.4 Hand-Craft Based Classifier

6.1.4.1 SVM Classifier

The Support vector machine is the most popular classifier on machine learning problems. This classifier is almost used in all researches which studied to solve the problem of facial expression recognition. SVM has different parameters staring from kernel type that is used to use classifier as multiple classifier or binary classifier, C parameter is used to assign the value of margin to prevent the classifier from the overfitting, so in proposed solution SVM is used as a classifier with the following parameters with the respective value.

- Kernel = gaussian (rbf)
- C = 12.5
- Gamma = auto

Table 6. 5: Performance of proposed solution Using SVM classifier

No	**Classifier**	**Dataset Split**	**Average Accuracy**
1	SVM with above parameters	90/10	82.97%
		80/20	82.52%
		70/30	83.60%
		10-Fold	85.38%

The following figure (6.4) shows the confusion matric of SVM classifier in case of the dataset split is 90% for training and 10% for the testing, figure (6.3) show the result of the same classifier in case the dataset split is 80 % for training and 20% for testing and next figure (6.5)

show the performance of the classifier in case the dataset split is 70% for the training and 30% for testing and figure 6.6 show accuracy obtained by 10-Fold mechanism.

As discussed above the performance of the classifier is decreases as the number of the training dataset is decreased and if the number of the training dataset is increased the performance of the classifier also increased.

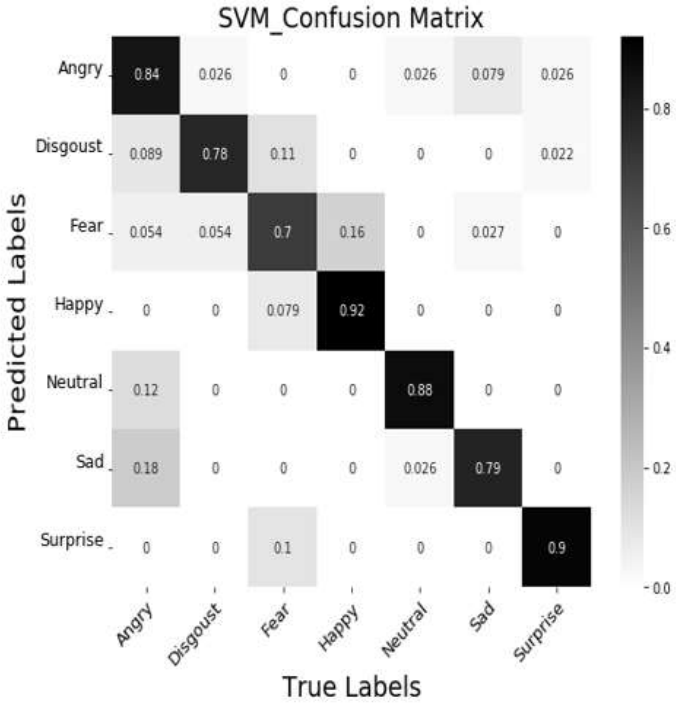


Figure 6. 4: Confusion matrix of SVM 90/10 split

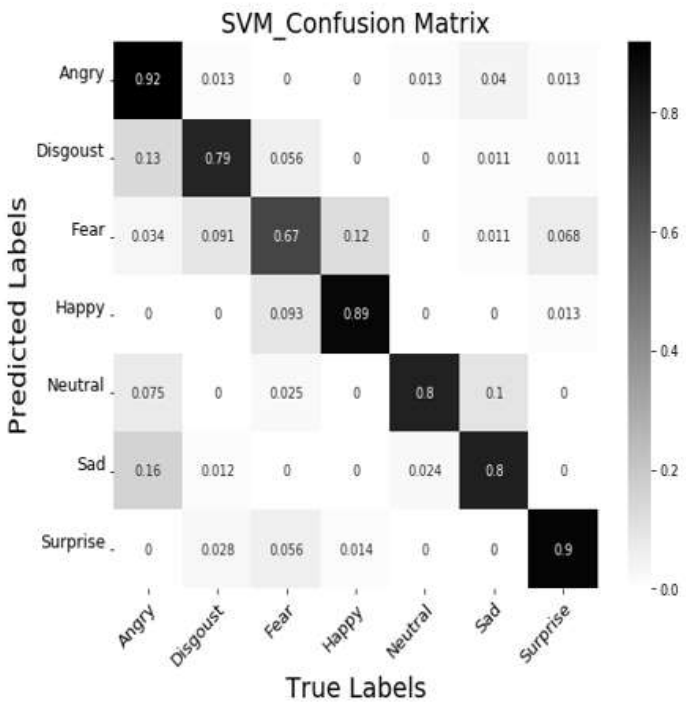


Figure 6. 3: Confusion matrix of SVM 80/20 split

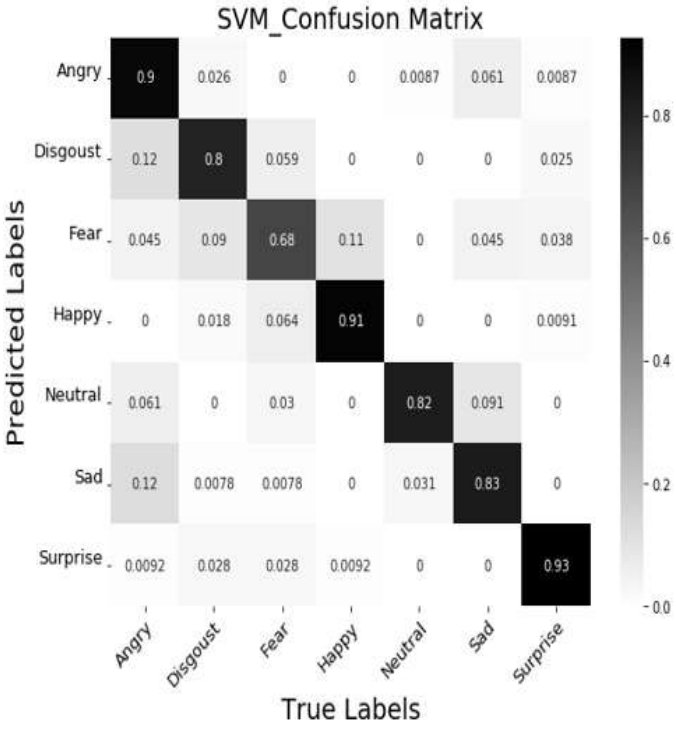


Figure 6. 6: Confusion matrix of SVM 70/30 split

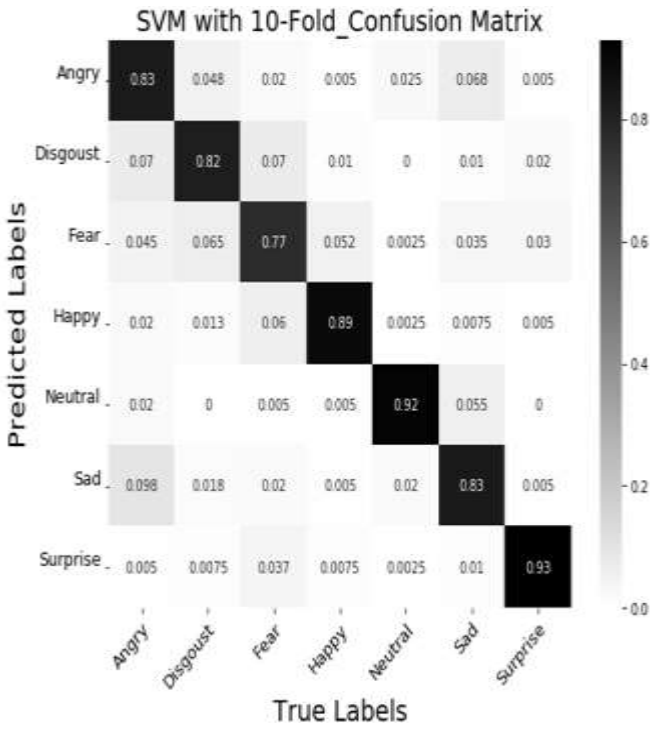


Figure 6. 5: Confusion matrix of SVM 10-Fold

The following chart shows the better accuracy of each class achieved using a 10-Fold training mechanism, which manages performance between each class as shown in figure 6.7.

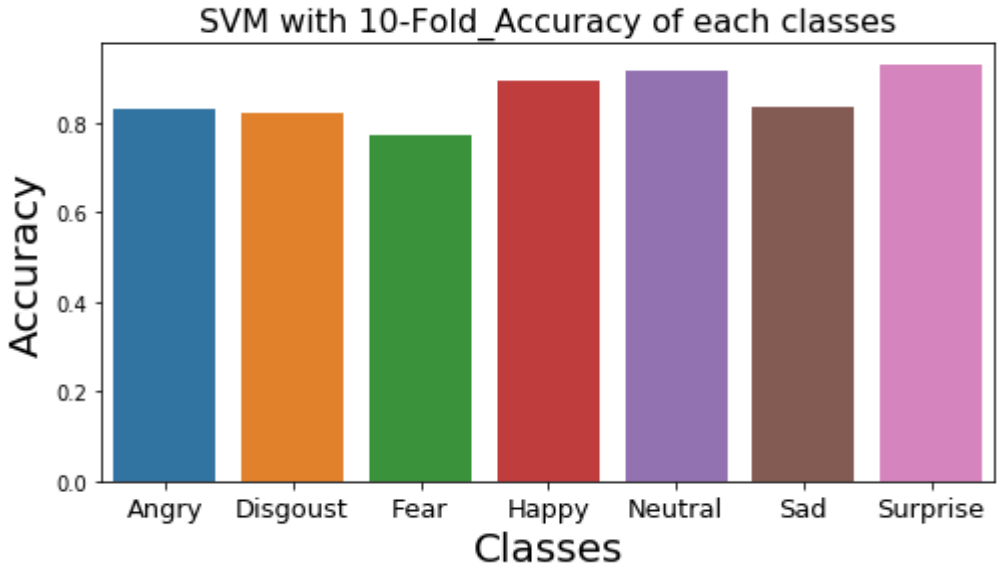


Figure 6. 7. Accuracy of each classes using SVM classifier

6.1.4.2 Logistic Regression Classifier

After the support vector machine classifier next popular classifier used for facial expression problems is logistic regression. This classifier also has different parameters like solver used to specify the model is used as binary or multiple classifiers and a multiclass attribute is used to calculate the loss of the model in training when the model is used for multiple class problems. For this classification method, the following parameters are used.

- Kernel / Solver = lbfgs
- Multi_class = multinominal
- Iteration = 1000

Table 6. 6: Performance of proposed solution Using LR classifier

No	**Classifier**	**Dataset Split**	**Average Accuracy**
1	Logistic Regression with above parameters	90/10	86.22%
		80/20	85.56%
		70/30	83.15%
		10-Fold	84.77%

The performance of the logistic regression classifier is tested by training the classifier by the same dataset split used to train the SVM classifier. From the following figures, figure 6.8 shows the confusion matrix for the 90/10 dataset split, figure 6.9 shows the confusion matrix for the 80/20 dataset split and figure 6.10 shows the confusion matrix for the 70/10 dataset split with respect to the above parameters and figure 6.11 shows the confusion matrix using 10-ford mechanism.

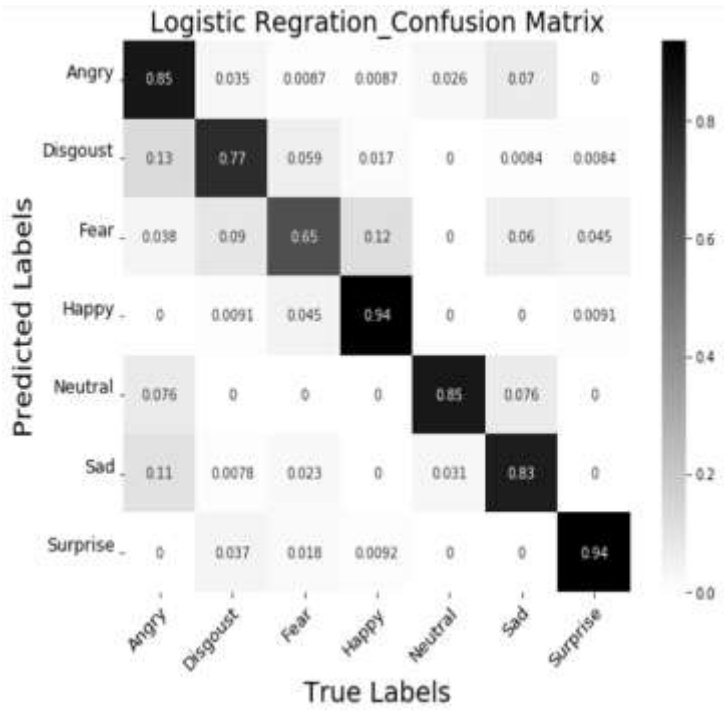


Figure 6. 10: Confusion matrix of LR 70/30 split

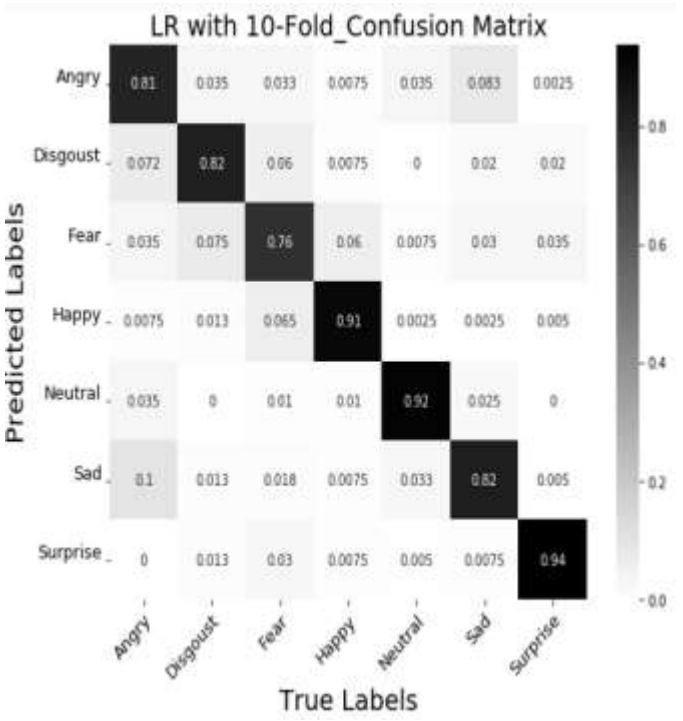


Figure 6. 11: Confusion matrix of LR 10-Fold

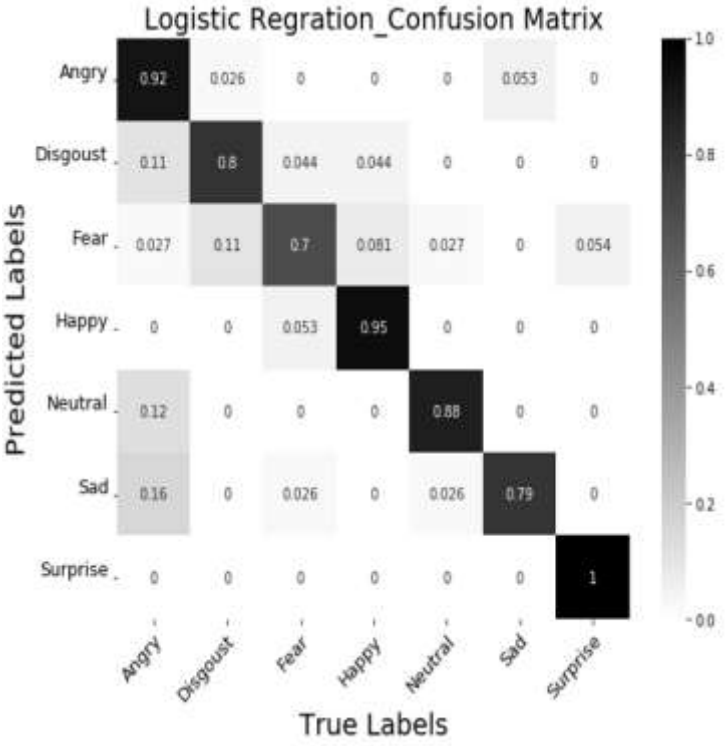


Figure 6. 9: Confusion matrix of LR 90/10 split

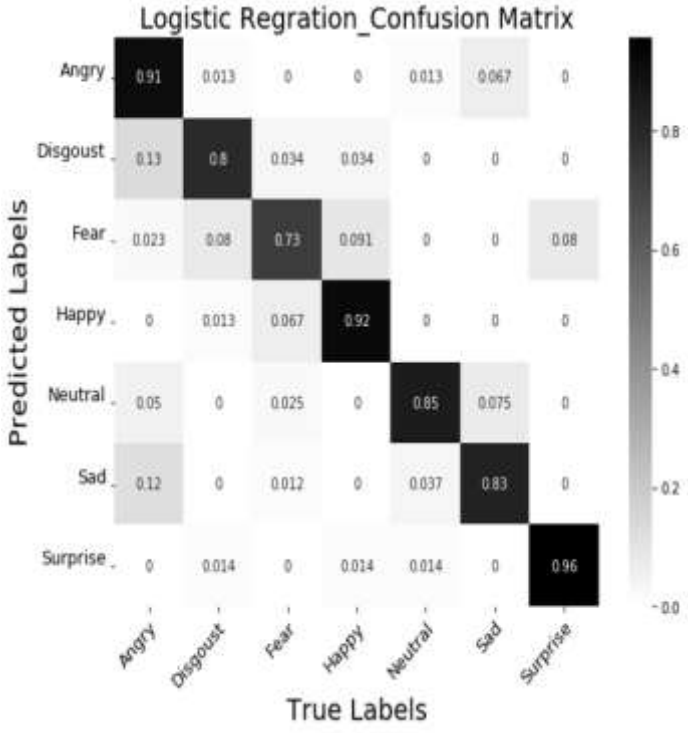


Figure 6. 8: Confusion matrix of LR 80/20 split

The LR classifier performs better on the 90/10 dataset split, in which the performance of fear is lower than others and the performance for surprise and happy are higher as shown in figure 6.12.

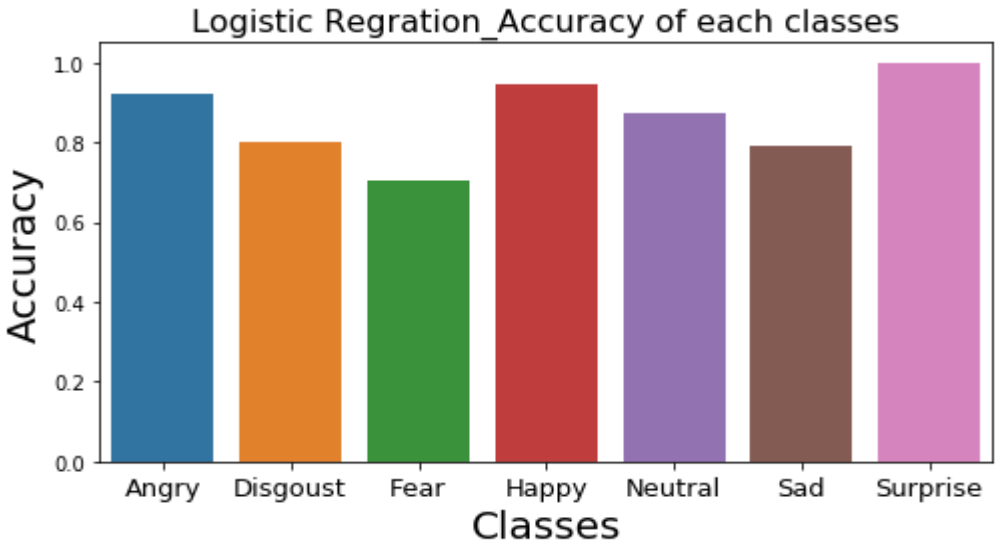


Figure 6. 12: Accuracy of each classes using LR classifier

6.1.5 Deep-Learning based Classifier

In deep learning header layer serve as a classifier and to differentiate weather the classifier is for binary target class or for multiple activation functions with different parameters are used and the other most commonly used parameter is an optimizer with Adam and stochastic gradient descent for the calculation of training loss. Depending on the problems some other parameters like decay, learning rate, momentum, an early stop is used.

6.1.5.1 Soft-max Classifier with Adam Optimizer

Soft-max classifier is used for multiple target class problems, so as an optimizer Adam optimizer used in the evaluation step of the proposed solution and soft-max classifier trained using different numbers of parameters in deep learning architecture. The following parameter values are used with Adam optimizer.

- Learning -rate = 0.0001, 0.000001, 0.00000001
- Number of parameters = 25M, 3M (where M represent million)
- Patience = 50 used for an early stop in case model overfit

The performance of the soft-max classifier with Adam optimizer 18 combined experiments is implemented depending on the dataset split discussed in 6.1 subtitles. From all experiments, the following table discusses the performance of the proposed solution by the soft-max classifier with Adam optimizer in a different parameter which is performed better than all experiments using a 90/10 dataset split.

Table 6. 7: Performance of proposed solution Using Soft-max classifier with Adam optimizer

No	Optimizer	Learning-rate	# Param	Average Accuracy
1	Adam	1e - 3	25 M	22.25%
		1e - 3	3 M	45.50%
		1e - 5	25 M	72.08%
		1e - 5	3 M	84.45%
		1e - 7	25 M	48.75%
		1e - 7	3 M	57.08%

From the above table if the number of parameters increases the performance of the proposed solution also increased. In addition to the parameter, if the learning rate decrease or increases from learning rate 1e-5 the accuracy of the proposed solution is decreased.

6.1.5.2 Soft-max Classifier with SGD Optimizer

Previous works on facial expression recognition used stochastic gradient descent as an optimizer in case of using soft-max as a classifier. This optimizer is implemented with different parameters to evaluate the performance of the proposed solution. The following parameters used with this optimizer.

- Learning -rate = 0.0001, 0.000001, 0.00000001
- Number of parameters = 25M, 3M (where M represent million)
- Patience = 50 used for the early stop in case model overfit
- Decay = 0.0001/no of epoch, 0.000001/ no of epoch, 0.00000001/ no of epoch
- Momentum = 0.9

As the same discussion for the soft-max classifier with Adam optimizer for a soft-max classifier with an SGD optimizer has 18 combinations of experiments are implemented. From all implemented experiments the following table discusses experiment which obtained the best performance using a 90/10 dataset split.

Table 6. 8: Performance of proposed solution Using Soft-max classifier with SGD optimizer

No	**Optimizer**	**Learning-rate**	**# Param**	**Average Accuracy**
1	SGD	1e - 3	25 M	15.80%
		1e - 3	3 M	42.62%
		1e - 5	25 M	72.08%
		1e - 5	3 M	82.50%
		1e - 7	25 M	48.75%
		1e - 7	3 M	50.80%

From the above table, the proposed solution is performed better in the case of 3 million parameters with a 1e-5 learning rate. By using other learning rates and the number of parameters the performance of the proposed solution is reduced.

As the discussion in subtitle 6.1.2.1 and 6.1.2.2, 18 experiments are implemented for each Adam and SGD optimizers to obtain better performance for the proposed solution, which means to get a better recognition rate using soft-max classifier for proposed solution 36 combinations of an experiment is implemented. Using the above discussion soft-max classifier with Adam optimizer with 1e-5 learning rate and 3 million number of parameters is performed better with 84.45 recognition rate as the following training and validation plot shows the performance of the deep-learning model using 7(seven) expression types.

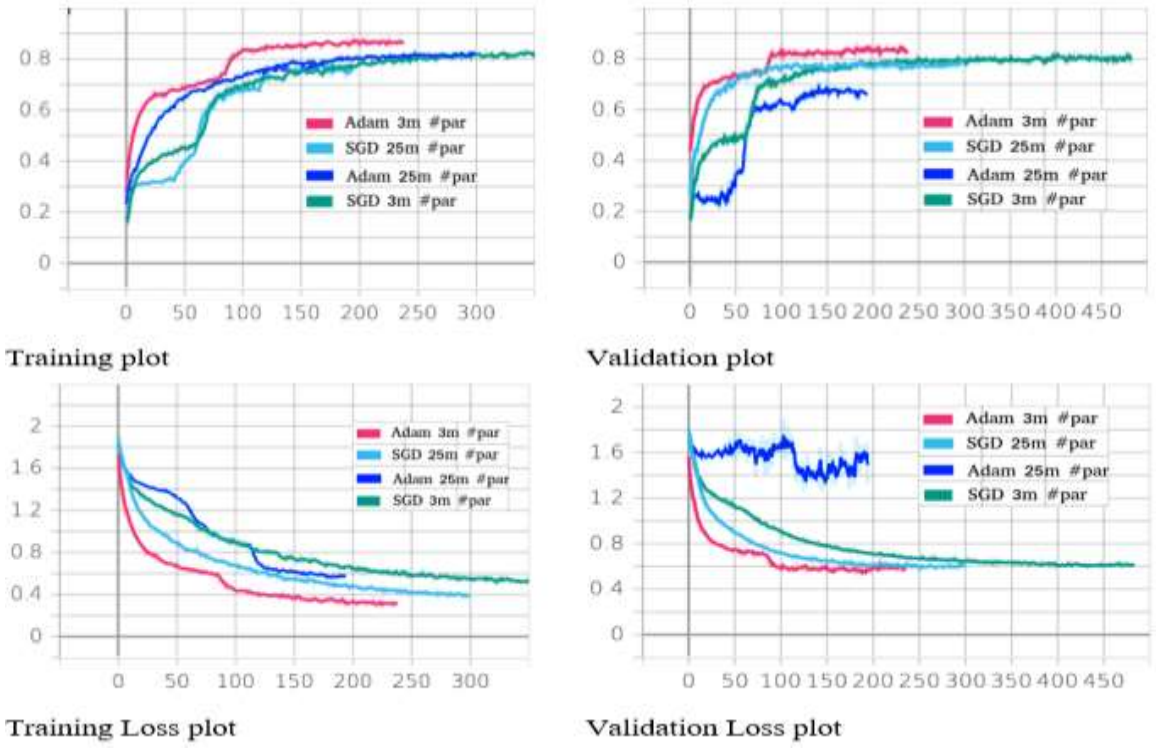


Figure 6. 13: Performance of deep-learning using 7 expression

As the above figures show the performance of the proposed soft-max classifier with a 1e-5 learning rate, 3 million parameters and Adam optimizer are better staring from training accuracy to validation accuracy. some of the lines in the plot stop at the early or middle of the epoch because early stopping method is used to prevent the model from overfitting.

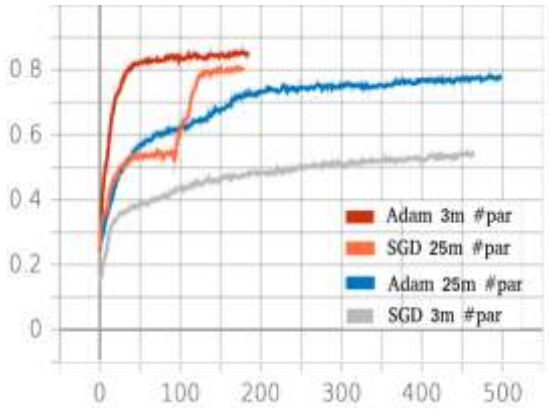
6.1.6 Hand-Craft and Deep-Learning Comparison

To select a model with better accuracy, hand-craft based methods and deep learning-based methods must be compared with each other. In order to identify classifiers with better performance, the following table helps to discuss in detail.

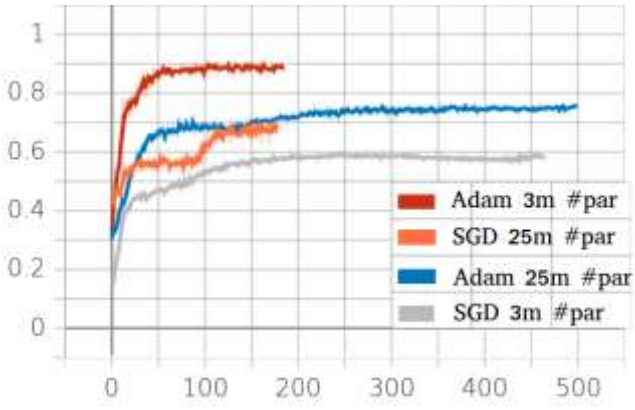
Table 6. 9: Compression of hand-craft techniques and soft-max

No	Classifier	# of Expression	Average Accuracy
1	SVM (Support Vector Machine)	7	85.05%
		6	89.54%
2	LR (Logistic Regression)	7	86.22%
		6	91.36%
3	Soft-Max	7	84.45%
		6	90.90%

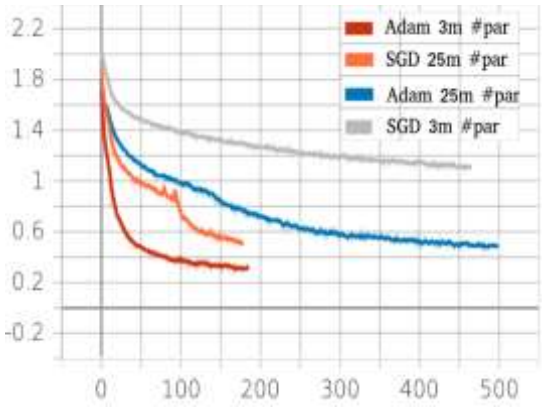
As discussed in this chapter subsection 6.1.4 and 6.1.6 in detail about the implantation and evaluation of proposed solution from the angle of handcraft classifiers and deep learning-based classifiers, logistic regression classifier performed better in case of 90/10 dataset split. In the case of a 10-Fold SVM classifier is performed better than LR. Finally, a soft-max classifier performed in 3rd place on 7 expression types namely known angry, disgust, fear, happy, neutral, sad and surprise. To check the model using 6 expression types namely angry, disgust, fear, happy, sad and surprise, each model is trained using features extracted from 6 expression datasets. the following plots show the performance of the proposed solution using 6 expressions.



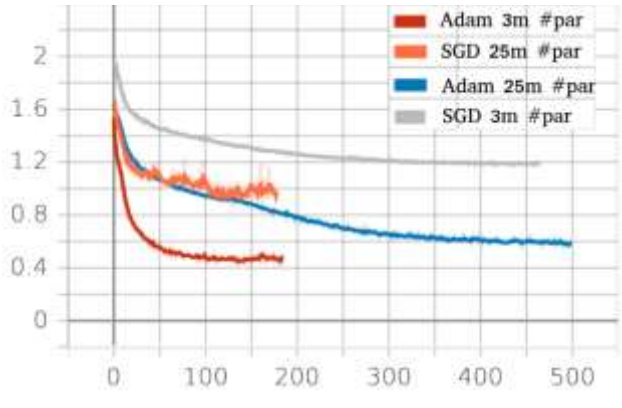
Training Accuracy for 6 expression



Validation Accuracy for 6 expression



Training Loss 6 for expression



Validation Loss for 6 expression

Figure 6. 14: Performance of deep learning using 6 expressions

6.2 Compression with Other Previous Related Works

To check the proposed method is performed better than previously used methods on the same problem. Meanly compare the proposed method with the previous method used 7 expression types rather than 6 expression types and methods which are used facial parts, local feature and global information by combining the features. In addition to the same problem area and number expression type method which used the BU-3DFE 3D dataset for training and testing their works. As the above discussion shows the proposed solution achieved better performance on the 7 and 6 types of expression using SVM, LR, and Soft-max classifiers. In most related work on

the same problems most researchers used 6 expression types, as discussed in previous works, the accuracy obtained by using 7 expression type is 77.3, which used BU-3DFE as training and testing dataset. Other previous work which used Bosporus [26] dataset for experimentation is achieved a 91.3 recognition rate. Finally, most previously experimented studies used 6 types of expressions and achieved an average accuracy of 86%. The following table discusses the comparison of the proposed solution with previously used methods on the fully related and partially related problems, with different parameters mainly parameters like dataset they used, number of expression type, the method used for classification.

Table 6. 10: Comparison of the proposed solution with related works

Previous work	**Dataset**	**# of Expression**	**Average Accuracy**
[17]	BU-3DFE	6	90.17%
[27]	>>	6	83.60%
[6]	>>	7	77.30%
[3]	>>	6	88.54%
[5]	Bosporus	7	91.32%
[26]	BU-3DFE	6	91.30%
[16]	>>	6	87.87%
[4]	>>	6	86.86%
Proposed solution	>>	6	91.36%
		7	86.22%

As shown in the above table 6.1 previous work is better accuracy on 6 expression types, which is used delta face that generated from the 3D model from the same database used in this thesis. [5] is achieved better accuracy using 7 expression types but, its dataset is from another publicly available 3D dataset, so the comparison of this achievement with the proposed solution did not provide better achievement. [6] achieved 77.30% accuracy for 7 types of expression using the same dataset used in the proposed solution. Finally, the performance of the proposed solution is better for 6 and 7 expression types.

6.3 Experiment Result

To evaluate the performance of the proposed solution using a scenario, the following experiments are implemented by the assumption of human and robot interaction application. As human-computer interaction first step is detecting human face from the frame of videos or images. Then to understand the expression, features extracted from the detected face and passed to the classifier to predict the status of the detected face using extracted features. Finally, the robot reacts depending on the response from the model. In order to check the performance of the final output of the proposed solution using the scenario discussed above the following steps are conducted on 7 different facial expression types where each expression types have 4 individuals testing samples.

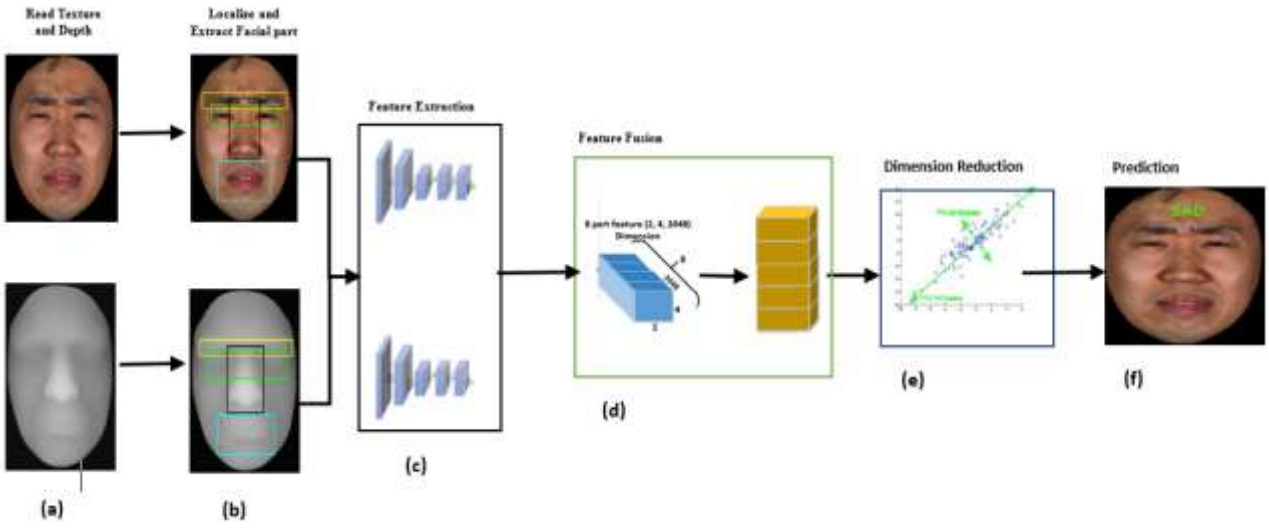


Figure 6. 15:System Scenario of the proposed solution

As a proposed solution uses both texture and depth images to recognize the facial expression, each type of image is loaded using Opencv. The following images are used as the input images for performance evaluation.

Figure 6. 16:Input image for scenario

Before localizing facial part landmarks of each facial part must be detected using the dlib landmark detection method. Localization and extraction of each facial part are implemented by using detected landmarks plus some neighbour pixels depending on the facial part. The localized facial part is resized into 50X100X3 for RGB texture image and depth image. The following image shows a localized and extracted facial part.

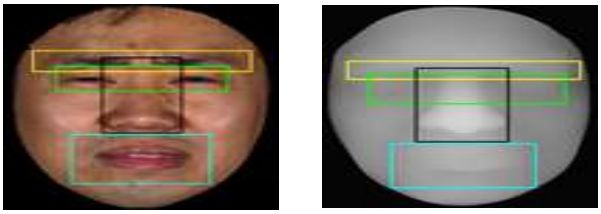


Figure 6. 17:Localized image using scenario

From the facial part, features are extracted using Resnet50 pre-trained model. The facial part that is fed into the resnet50 model is 50X100X3 and the extracted feature has [1x2x4x2048] dimensions for each facial part.

In the fusion layer, the dimension is reshaped to [8x8x2048] after all facial part feature is combined. A new feature with [1, 8192] dimension is obtained after fusion with 4-layer convolution is implemented. In next step dimension of feature is reduced to [1, 1543] by using PCA with 99% of number components. Finally, the predicted expression of the input image using a trained support vector machine model with the predicted label is shown in the following figure.

Figure 6. 18: Predicted image using scenario

After the scenario-based experiment is conducted on 4 individuals testing samples with 7 facial expression types, 50% of Angry facial expression types are recognized correctly where each Disgust, Surprise and Neutral facial expression types are recognized 100% from each testing samples. Each Fear, Happy and Sad facial expression types are recognized 75% of each testing samples. Fear, Happy and Sad facial expression types are recognized correctly. The following diagram shows that different facial parts can represent different facial expression types. For instant, Happy facial expression type must be classified as happy expression type is classified as Disgust facial expression type because the contribution of eye facial part is greater than other facial parts. So, from each experiment, conducted on 4 individuals, disgust facial expression type is best represented by an eye facial part than other facial parts. The relationship between facial expression types and facial parts discussed after the following diagram in detail. The following diagram shows the testing dataset from 4 individuals where individuals are from different continents namely White, East Asian, Indian, Black (African) and predicted value depending on the testing datasets.

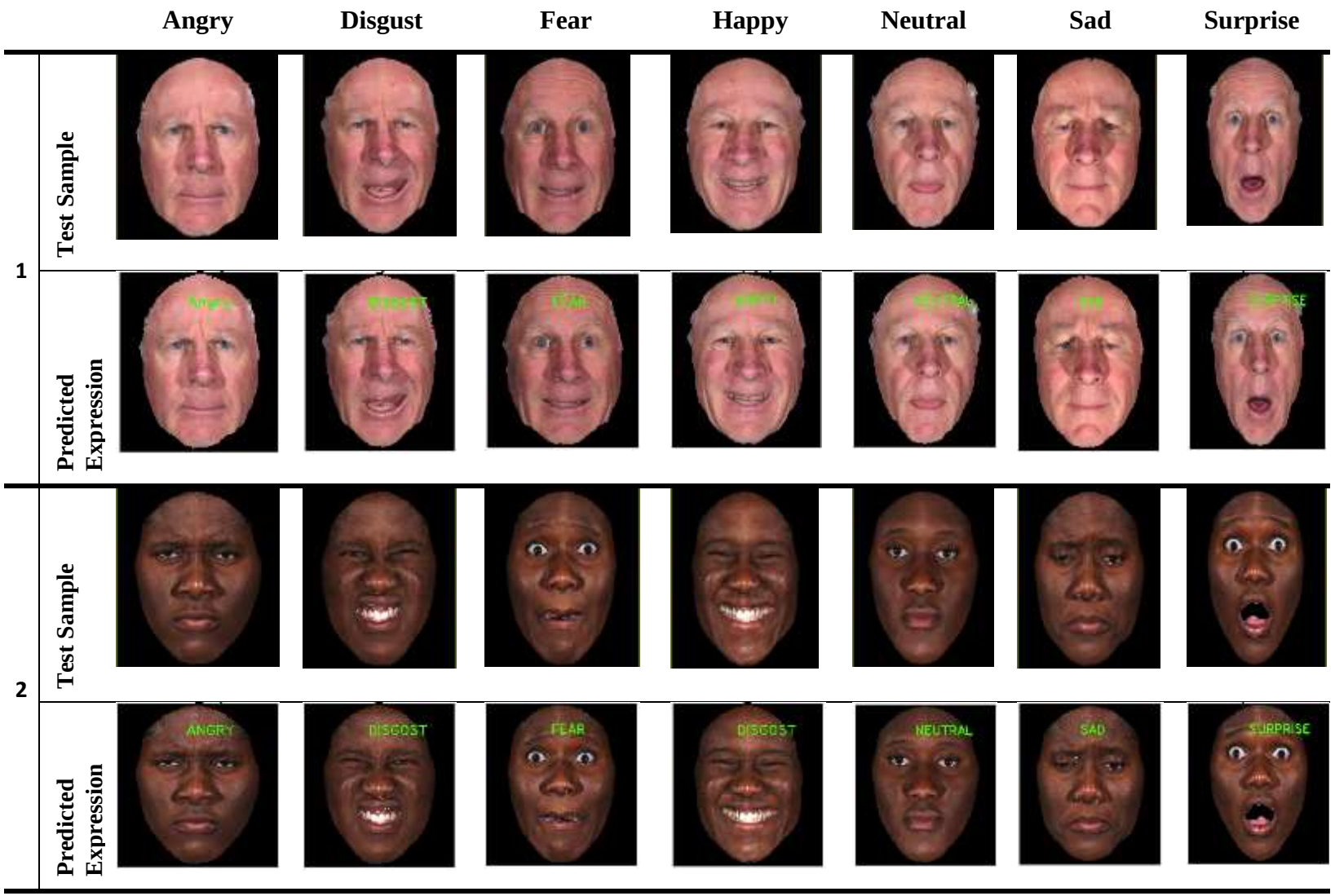

Angry
Disgust
Fear
Happy
Neutral
Sad
Surprise
1
Test Sample
Predicted Expression
2
Test Sample
Predicted Expression
ANGRY
DISGUST
FEAR
DISGUST
NEUTRAL
SAD
SURPRISE

Figure 6. 19:Samples recognized images using 4 individual testing samples.

From the above figures, figure 6.19 shows the output of testing the proposed solution using 7 facial expression types and correct classification and some misclassification obtained by using the final SVM model which achieved 85.38% recognition rate. In addition to the recognition rate, it shows the contribution of facial parts to facial expression types. Mouth facial part has a better contribution to Happy, and Surprise facial expression categories where eyebrow facial part contributed to Angry facial expression type. A combination of eye and eyebrow facial parts are the best representatives to Fear facial expression type. Disgust facial expression type is represented by a combination of eye and mouth facial parts where the combination of eye and mouth represents sad facial expression type. Nose facial parts can represent disgust, fear and angry facial expression types. All facial parts contributed to represent neutral facial expression type in 3D facial expression. The following figure shows the detailed result obtained after the implementation of a scenario-based model evaluation starting from facial part extraction to predicted result.

Figure 6. 20:Sample landmark localization recognized images

Chapter Seven

7 Conclusion, Recommendation, Future work

7.1 Conclusion

Currently, most of the human works are replaced by robots and interaction between humans and robots must be smooth. For a robot to interact with humans smoothly, it needs to understand the behaviour or condition of humans. Even robots for helping elder need to understand their condition by analysing information from there face. Currently, in addition to helping and provide services to humans or other machines, robots are used for medical analysis. In order to implement the above discussion, the robot must read information from an individual face.

To improve the interaction smoothness between humans and robot many researchers' studies on facial expression recognition areas. Most researches on facial expression recognition are depending on 6(six) types of expression some of the researches papers used 7 expression types but, in case of real-world many different expressions are there, so increasing the number of expressions to be recognized increase the smoothness of interaction between human and robot.

In the proposed solution mainly 7 expression types are used and for compression reason 6 expression types also implemented. The dataset for the proposed solution is collected from the University of Binghamton which contained 2500 image samples from 56 females and 44 male individuals. The database contained a 3D model of 7 expressions with textured and manually labelled landmarks for each sample data. From each 3D model, the depth map is generated after pre-processing is implemented. The next step is detecting landmarks from texture image and localize facial parts on both textures and generated depth image. Then extract face part from both images by adding a few neighbour pixels localized face part to prevent information loss. Then using different pre-trained models feature is extracted but, mainly Resnet50 pre-trained model is used for feature extraction.

After feature extracted from each facial part, combining extracted feature is the next step then reshape combined feature and apply fusion to reshaped feature. The fusion layer has 4 convolutional layers with dropout and batch normalization to prevent overfitting. The feature dimension is changed to 8192 vectors after the feature fusion layer implemented. Then using the PCA dimension reduction method 99% of the principal component is reduced.

Finally, using reduced features train SVM, LR, Soft-max classifiers with different parameters and obtained an accuracy 86.22 for 7 expression types and 91.36 for 6. Then evaluate the trained model and compare the performance of the proposed solution with previous works on the same problem as shown in table 6.10.

7.2 Recommendation

There is a shortage of 3D datasets in the facial expression recognition area and that makes it hard for researchers to improve the recognition performance of facial expression. As one of the researchers in this area, I recommend constructing efficient 3D models for expression recognition that may lead to the use of deep-learning rather than hand-craft methods to achieve better accuracy. Using a fusion of different forms of a dataset is better than using a single form.

7.3 Future works

Still, the facial expression recognition rate is needed to be improved. A feature vector with a 4096 dimension represented a single expression after a new flatten layer is added to current fusion CNN. Train and test classification methods used in the proposed solution by using a new feature vector which is obtained after the new flatten layer is added to current fusion CNN. Other best face representation system is delta face, so train and test the current classification methods by combining depth, texture, and delta face features. As third future work combining 83 manually labelled landmark with a current feature vector, train and test method like KNN, Random forest in addition to proposed classification methods.

As recommended in recommendation developing the local 3D model and use the developed model of facial expression and combine model used in this proposed solution with the developed model and check the performance of the combination on datasets.

Finally, if a resource like GPU is available to construct a 3D model on the execution time of the model testing stage and use the constructed model as a testing image. in other words, implement the proposed solution directly to the real-world problems and check the performance of the proposed solution.

References

[1] D. Acharya, Z. Huang, D. P. Paudel, and L. Van Gool, "Covariance pooling for facial expression recognition," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2018-June, pp. 480–487, 2018.

[2] S. Shojaeilangari, W. Y. Yau, K. Nandakumar, J. Li, and E. K. Teoh, "Robust Representation and Recognition of Facial Emotions Using Extreme Sparse Learning," *IEEE Trans. Image Process.*, vol. 24, no. 7, pp. 2140–2152, 2015.

[3] A. Jan, H. Ding, H. Meng, L. Chen, and H. Li, "Accurate facial parts localization and deep learning for 3D facial expression recognition," *Proc. - 13th IEEE Int. Conf. Autom. Face Gesture Recognition, FG 2018*, pp. 466–472, 2018.

[4] H. Li, J. Sun, Z. Xu, and L. Chen, "Multimodal 2D+3D Facial Expression Recognition with Deep Fusion Convolutional Neural Network," *IEEE Trans. Multimed.*, vol. 19, no. 12, pp. 2816–2831, 2017.

[5] W. Wang, W. Zheng, and Y. Ma, "3D facial expression recognition based on combination of local features and globe information," *Proc. - 2014 6th Int. Conf. Intell. Human-Machine Syst. Cybern. IHMSC 2014*, vol. 2, pp. 20–25, 2014.

[6] D. Han and Y. Ming, "Facial expression recognition with LBP and SLPP combined method," *Int. Conf. Signal Process. Proceedings, ICSP*, vol. 2015-Janua, no. October, pp. 1418–1422, 2014.

[7] M. Lennon, G. Mercier, M. C. Mouchot, and L. Hubert-Moy, "Independent component analysis as a tool for the dimensionality reduction and the representation of hyperspectral images," *Int. Geosci. Remote Sens. Symp.*, vol. 6, no. 1, pp. 2893–2895, 2001.

[8] K. Keerthi Vasan and B. Surendiran, "Dimensionality reduction using Principal Component Analysis for network intrusion detection," *Perspect. Sci.*, vol. 8, no. July, pp. 510–512, 2016.

[9] Y. Wu, T. Hassner, K. Kim, G. Medioni, and P. Natarajan, "Facial Landmark Detection

with Tweaked Convolutional Neural Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3067–3074, 2018.

[10] Z. H. Feng and J. Kittler, "Advances in facial landmark detection," *Biometric Technol. Today*, vol. 2018, no. 3, pp. 8–11, 2018.

[11] D. Sudha and M. Ramakrishna, "Comparative Study of Features Fusion Techniques," *Proc. - 2017 Int. Conf. Recent Adv. Electron. Commun. Technol. ICRAECT 2017*, pp. 235–239, 2017.

[12] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "Supplementary material for : How transferable are features in deep neural networks ?," *Nips*, pp. 1–5, 2014.

[13] Z. Huang, Z. Pan, and B. Lei, "Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data," *Remote Sens.*, vol. 9, no. 9, pp. 1–21, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016.

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1800–1807, 2017.

[16] P. Zarbakhsh and H. Demirel, "Fuzzy SVM for 3D facial expression classification using sequential forward feature selection," *Proc. - 9th Int. Conf. Comput. Intell. Commun. Networks, CICN 2017*, vol. 2018-Janua, pp. 131–134, 2018.

[17] S. An and Q. Ruan, "3D facial expression recognition algorithm using local threshold binary pattern and histogram of oriented gradient," *Int. Conf. Signal Process. Proceedings, ICSP*, pp. 265–270, 2017.

[18] A. Jan and Hongying Meng, "Automatic 3D facial expression recognition using geometric and textured feature fusion," *IEEE Int. Conf. Work. Autom. Face Gesture Recognit.*, pp. 1–6, 2015.

[19] H. Li *et al.*, "An efficient multimodal 2D + 3D feature-based approach to automatic facial expression recognition," *Comput. Vis. Image Underst.*, vol. 140, pp. 83–92, 2015.

[20] X. Li, Q. Ruan, G. An, and Y. Jin, "Analysis of range images used in 3D facial expression recognition systems," *Comput. Informatics*, vol. 35, no. 1, pp. 203–221, 2016.

[21] H. Yang and L. Yin, "CNN based 3D facial expression recognition using masking and landmark features," *2017 7th Int. Conf. Affect. Comput. Intell. Interact. ACII 2017*, vol. 2018-Janua, pp. 556–560, 2018.

[22] Q. Li, G. An, and Q. Ruan, "3D Facial expression recognition using orthogonal tensor marginal fisher analysis on geometric maps," *Int. Conf. Wavelet Anal. Pattern Recognit.*, vol. 1, pp. 65–71, 2017.

[23] W. Hariri, H. Tabia, N. Farah, A. Benouareth, and D. Declercq, "3D facial expression recognition using kernel methods on Riemannian manifold," *Eng. Appl. Artif. Intell.*, vol. 64, no. May, pp. 25–32, 2017.

[24] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, "A 3D facial expression database for facial behavior research," *FGR 2006 Proc. 7th Int. Conf. Autom. Face Gesture Recognit.*, vol. 2006, no. August, pp. 211–216, 2006.

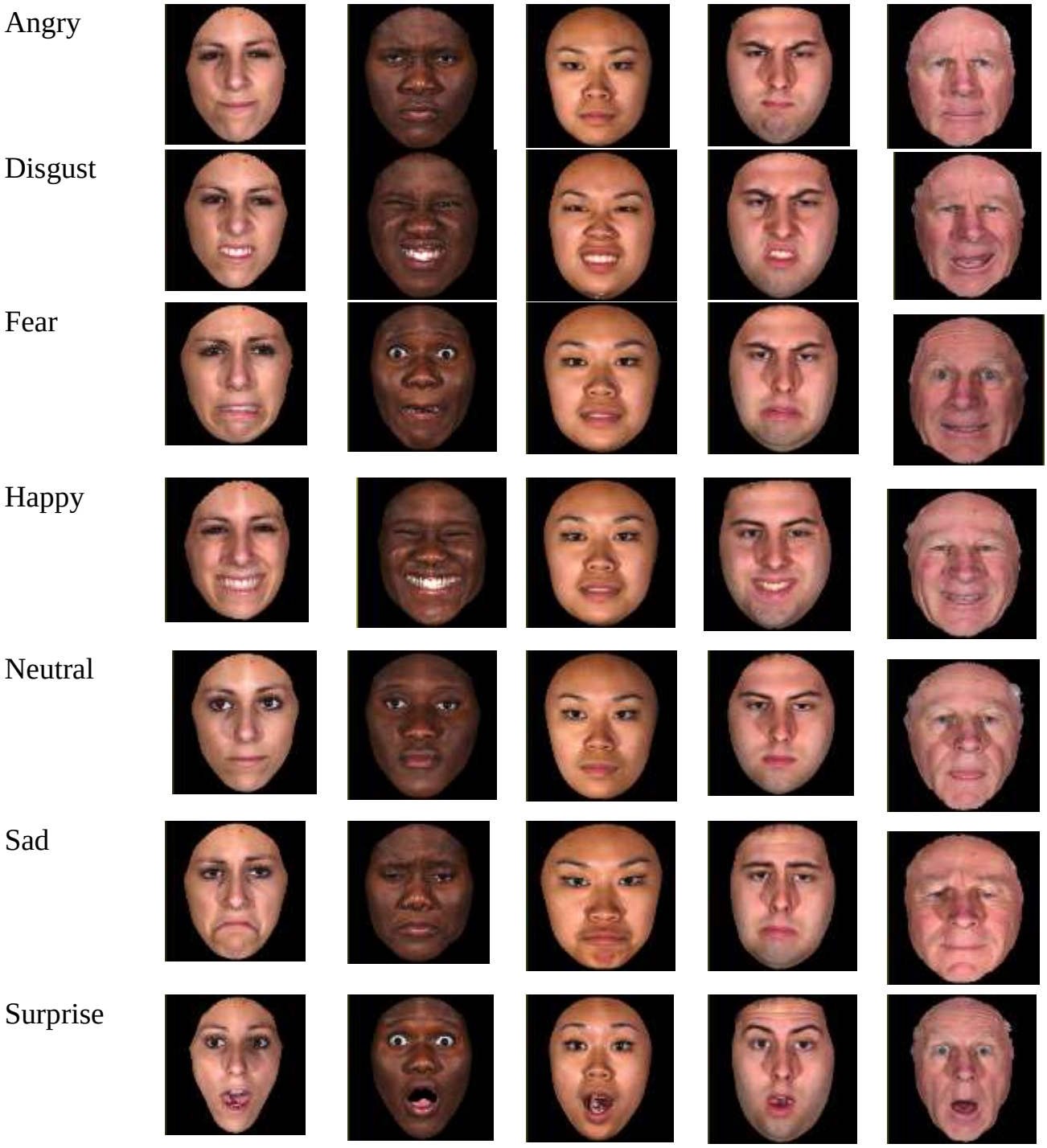
[25] X. Dong, Y. Yan, W. Ouyang, and Y. Yang, "Style Aggregated Network for Facial Landmark Detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 379–388, 2018.

[26] X. Li, Q. Ruan, and G. An, "3D Facial Expression Recognition Using Delta Faces," *5th IET Int. Conf. Wireless, Mob. Multimed. Networks (ICWMMN 2013)*, vol. 1, pp. 234–239, 2013.

[27] J. Wang, L. Yin, X. Wei, and Y. Sun, "3D facial expression recognition based on primitive surface feature distribution," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, pp. 1399–1406, 2006.

Appendixes

Appendix A: Sample Dataset



Appendix B: Sample Python Codes

Feature Fusion Code

```
# import the necessary packages
from keras.layers import Conv2D, InputLayer, Flatten, concatenate,
Input, Dropout, BatchNormalization, Activation
from keras.models import Sequential, Model
import numpy as np
```

```
# Create model to read texture image of facial part
mouth_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                    name='mouth_input')
eye_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                  name='eye_input')
eye_brow_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                       name='eye_brow_input')
nose_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                   name='nose_input')

# Create model to read Depth image of facial part
depth_mouth_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                          name='depth_mouth_input')
depth_eye_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                        name='depth_eye_input')
depth_eye_brow_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                             name='depth_eye_brow_input')
depth_nose_input = Input(shape=(2 , 4 , 2048), dtype='float32',
                         name='depth_nose_input')
```

```
# Combine all facial part model
ALL_Method = concatenate([mouth_input, eye_input,
                          eye_brow_input, nose_input,
                         depth_mouth_input, depth_eye_input,
                          depth_eye_brow_input, depth_nose_input],
                         name='all_models')

# Display the dimension of combined model
ALL_Method.shape
# Crate new model using combine model
fusion = Model(inputs = [mouth_input, eye_input,
                         eye_brow_input, nose_input,
                         depth_mouth_input, depth_eye_input,
                         depth_eye_brow_input, depth_nose_input],
               outputs = ALL_Method)
```

```
# Extract feature using new combined model
new_feature = fusion.predict([np.array(mouth_predicted),
                              np.array(eye_predicted),
                            np.array(eyeb_predicted),
                              np.array(nose_predicted),
                           np.array(debth_mouth_predicted),
                              np.array(debth_eye_predicted),
                              np.array(debth_eyeb_predicted),
                              np.array(depth_nose_predicted)])
# Display the dimension of newly extracted feature
new_feature.shape
# Reshape newly extracted feature
new_feature = np.reshape(predict, (1,8,8,2048))
# Display reshaped feature dimension
new_feature.shape
# Create final fusion model
# Create input layer that takes reshaped new feature
part_input = Input(shape=(8, 8, 2048), dtype='float32',
                   name='part_input')
# Create convolutional, Batchnormalization, Dropout and Flatten layers
```

```
# Convolution 1
conv_1 = Conv2D(1024, (4,3), padding="same", activation='relu',
                name = 'final_conv1')(part_input)
conv_1 = Dropout(0.25)(conv_1)
conv_1 = BatchNormalization()(conv_1)
# Convolution 2
conv_2 = Conv2D(512, (1,1), padding="same", activation='relu',
                name = 'final_conv2')(conv_1)
conv_2 = Dropout(0.25)(conv_2)
conv_2 = BatchNormalization()(conv_2)
# Convolution 3
conv_3 = Conv2D(256, (1,1), padding="same", activation='relu',
                name = 'final_conv3')(conv_2)
conv_3 = Dropout(0.25)(conv_3)
conv_3 = BatchNormalization()(conv_3)
# Convolution 4
conv_4 = Conv2D(128, (1,1), padding="same", activation='relu',
                name = 'final_conv4')(conv_3)
conv_4 = Dropout(0.25)(conv_4)
conv_4 = BatchNormalization()(conv_4)
```

```
# Convolution 4
conv_4 = Conv2D(128, (1,1), padding="same", activation='relu',
                name = 'final_conv4')(conv_3)
conv_4 = Dropout(0.25)(conv_4)
conv_4 = BatchNormalization()(conv_4)
# flatten layer
flat = Flatten()(conv_4)
# Creaate model using final fusion model
last_fusion = Model(part_input, flat)
# Extract feature using lasf fusion
final_feature = last_model.predict(predict_new)
# Display final_feature dimansion
final_feature.shape
```

Training Code

In [2]:
```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split
(TrainX, TrainY, test_size = 0.10)
```

In [3]:
```
svm_clf = SVC(kernel='rbf', C=15, gamma='auto')
lr_model = LogisticRegression(solver='lbfgs',
        multi_class='multinomial', max_iter=1000)
```

In [10]:
```
start = time.time()
svm_clf.fit(np.array(x_train),np.array(y_train))
#lr_model.fit(np.array(x_train),np.array(y_train))
end = time.time()
print('\n\n time spend to train SVM: ', (end-start)/60,
      'minutes \n\n')
```

```
C:\Users\given\Anaconda3\lib\site-packages\sklearn\utils\
when a 1d array was expected. Please change the shape of
  y = column_or_1d(y, warn=True)
```

```
 time spend to train SVM:  13.146777447064718 minutes
```

In [17]:
```
# Save the model as pickle file
svm_model = 'SVM_Model.sav'
pickle.dump(svm_clf, open(svm_model, 'wb'))
```

In [12]:
```
start = time.time()
svm_y_pred = svm_clf.predict(x_test)
end = time.time()
print('\n\n time spend to test SVM model: ',
      (end-start)/60, 'minutes \n\n')
```

```
 time spend to test SVM model:  1.139573629697164 minutes
```

In [13]:
```
start = time.time()
lr_y_pred = lr_model.predict(x_test)
end = time.time()
print('\n\n time spend to test LR model: ',
      (end-start)/60, 'minutes \n\n')
```

```
 time spend to test LR model:  0.004992290337880453 minutes
```

In [14]:
```
target_class = ['Angry','Disgoust','Fear','Happy',
                'Neutral','Sad','Surprise']
print(confusion_matrix(y_test, svm_y_pred))
print(classification_report(y_test, svm_y_pred,
                            target_names = target_class))
print(confusion_matrix(y_test, lr_y_pred, labels=[0,1,2,3,4,5,6]))
print(classification_report(y_test, lr_y_pred, target_names = target_class))
```

Dimension Reduction code

In [73]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [74]:
```
scalar = StandardScaler()
scalar.fit(x_train)
```

Out[74]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [75]:
```
scalar_train = scalar.transform(x_train)
scalar_test = scalar.transform(x_test)
```

In [76]:
```
pca = PCA(.99)
pca.fit(scalar_train)
print('PCA components = :', pca.n_components)
```

```
PCA components = : 0.99
```

In [77]:
```
pca_train = pca.transform(scalar_train)
pca_test = pca.transform(scalar_test)
```

In [78]:
```
print(pca_train.shape)
print(pca_test.shape)
```

```
(1680, 1543)
(720, 1543)
```

Facial Part Extraction Code

```
# import the necessary packages
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2
```

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
    help="path to facial landmark predictor")
ap.add_argument("-i", "--image", required=True,
    help="path to input image")
ap.add_argument("-i", "--depth", required=True,
    help="path to depth image")
args = vars(ap.parse_args())
```

```
# detect faces in the grayscale image
rects = detector(gray, 1)

# loop over the face detections
for (i, rect) in enumerate(rects):
    # determine the facial landmarks for the face region, then
    # convert the landmark (x, y)-coordinates to a NumPy array
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
    # loop over the face parts individually
    for (name, (i, j)) in face_utils.FACIAL_LANDMARKS_IDXS.items():
        # clone the original image so we can draw on it, then
        # display the name of the face part on the image
        print('star', i)
        print('end', j)
        clone = image.copy()
        cv2.putText(clone, name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
                    0.7, (0, 0, 255), 2)
```

```
# loop over the subset of facial landmarks, drawing the
# specific face part
for (x, y) in shape[i:j]:
    cv2.circle(clone, (x, y), 3, (255, 255, 255), -1)
classes = 3
# extract the ROI of the face region as a separate image
if i == 17:
    j = 27
    (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
    roi = image[y:y + h, x:x + w]
    depth_roi = depth[y:y + h, x:x + w]
    roi = imutils.resize(roi, width=250, height=250,
                         inter=cv2.INTER_CUBIC)
    cv2.imwrite('eye_brow_part1'+ str(classes) + str(im_num)
                + '.jpg', roi)
    cv2.imwrite('depth_eye_brow_part1'+ str(classes)  +
                str(im_num) + '.jpg', depth_roi)
elif i == 36:
    j = 48
    (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
    roi = image[y:y + h, x:x + w]
    depth_roi = depth[y:y + h, x:x + w]
```

```
    roi = imutils.resize(roi, width=250, height=250, inter=cv2.INTER_CUBIC)
    cv2.imwrite('eye_part1'+ str(classes) + str(im_num) + '.jpg', roi)
    cv2.imwrite('depth_eye_part1'+ str(classes) + str(im_num) + '.jpg', depth_roi)
elif i == 48:
    (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
    roi = image[y:y + h, x:x + w]
    depth_roi = depth[y:y + h, x:x + w]
    roi = imutils.resize(roi, width=250, height=250, inter=cv2.INTER_CUBIC)
    cv2.imwrite('mouth_part1'+ str(classes) + str(im_num) + '.jpg', roi)
    cv2.imwrite('depth_mouth_part1'+ str(classes) + str(im_num) + '.jpg', depth_roi)
elif i == 27:
    (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
    roi = image[y:y + h, x:x + w]
    depth_roi = depth[y:y + h, x:x + w]
    roi = imutils.resize(roi, width=250, height=250)
    cv2.imwrite('nose_part1' + str(classes) + str(im_num) + '.jpg', roi)
    cv2.imwrite('depth_nose_part1' + str(classes) + str(im_num) + '.jpg', depth_roi)
im_num = im_num + 1
cv2.imshow("ROI", roi)
cv2.imshow("Image", clone)
cv2.waitKey(0)
```

Appendix C: Sample Facial Expression Recognition