DEEP LEARNING BASED FACIAL EXPRESSION RECOGNITION AND ITS APPLICATIONS

A thesis submitted to **Brunel University London** for the degree of Doctor of Philosophy (Ph.D.)

By

Asim Jan

Supervised by

Dr. Hongying Meng

From the department of

Electronic and Computer Engineering

August, 2017

Abstract

Facial expression recognition (FER) is a research area that consists of classifying the human emotions through the expressions on their face. It can be used in applications such as biometric security, intelligent human-computer interaction, robotics, and clinical medicine for autism, depression, pain and mental health problems. This dissertation investigates the advanced technologies for facial expression analysis and develops the artificial intelligent systems for practical applications.

The first part of this work applies geometric and texture domain feature extractors along with various machine learning techniques to improve FER. Advanced 2D and 3D facial processing techniques such as Edge Oriented Histograms (EOH) and Facial Mesh Distances (FMD) are then fused together using a framework designed to investigate their individual and combined domain performances.

Following these tests, the face is then broken down into facial parts using advanced facial alignment and localising techniques. Deep learning in the form of Convolutional Neural Networks (CNNs) is also explored also FER. A novel approach is used for the deep network architecture design, to learn the facial parts jointly, showing an improvement over using the whole face. Joint Bayesian is also adapted in the form of metric learning, to work with deep feature representations of the facial parts. This provides a further improvement over using the deep network alone.

Dynamic emotion content is explored as a solution to provide richer information than still images. The motion occurring across the content is initially captured using the Motion History Histogram descriptor (MHH) and is critically evaluated. Based on this observation, several improvements are proposed through extensions such as Average Spatial Pooling Multi-scale Motion History Histogram (ASMMHH). This extension adds two modifications, first is to view the content in different spatial dimensions through spatial pooling; influenced by the structure of CNNs. The other modification is to capture motion at different speeds. Combined, they have provided better performance over MHH, and other popular techniques like Local Binary Patterns – Three Orthogonal Planes (LBP-TOP).

Finally, the dynamic emotion content is observed in the feature space, with sequences of images represented as sequences of extracted features. A novel technique called Facial Dynamic History Histogram (FDHH) is developed to capture patterns of variations within the sequence of features; an approach not seen before. FDHH is applied in an end to end framework for applications in Depression analysis and evaluating the induced emotions through a large set of video clips from various movies. With the combination of deep learning techniques and FDHH, state-of-the-art results are achieved for Depression analysis.

Keywords: Facial Expression, Image Processing; Facial Expression Recognition; Convolutional Neural Networks; Machine Learning; Deep Learning; Motion Description, Depression;

Declaration

I hereby declare that this thesis is entirely my work, except where otherwise indicated, describes my research or publications. No part of this thesis has been previously submitted to this or any other university as part of the requirement for a higher degree.

Asim Jan

Acknowledgement

I would like to express my gratitude and appreciation to my supervisor, Dr. Hongying Meng who has helped and supported me throughout my work. Dr. Meng has advised me and gave me many opportunities through challenges and events to extend my knowledge further. I would also like to acknowledge the School of Engineering & Design/Thomas Gerald Gray for their financial support to make this journey possible.

Additionally, I would like to thank my colleagues for their support, teamwork and contribution towards our achievements. Your efforts have encouraged me to go above and beyond. Penultimately, I would also like to thank the group at Ecole Centrale de Lyon for training and contributing towards my knowledge in deep learning.

My biggest thanks go to my wife, Sema Zahid, for her encouragement, support, love and sacrifices she made, alongside the challenges of her Ph.D. that she is facing. She has been the motivation and driving force from the beginning till my completion. To my parents, siblings and family-in-law who have made this journey possible, supporting me throughout and helping me with advice on anything I needed. Thanks to my father-in-law for his guidance and advice in the area. My good friends and colleagues Uthay and Yona, many thanks for going through the entire process of my degree with me from the beginning.

Contents

List of Figures

List of Tables

List of Abbreviations

AAM – Active Appearance Model

ASMMHH – Average Spatial Pooling Multi-scale Motion History Histogram

AUs – Action Units

BDI-II – Beck Depression Inventory II

BMH – Binary Motion History

CCA – Canonical Correlation Analysis

CNN – Convolutional Neural Network

CoST – Corpus of Social Touch

DBN – Deep Belief Networks

DCNN – Deep Convolutional Neural Network

ELBP – Extended Local Binary Pattern

ELMs – Extreme Learning Machines

EOH – Edge Oriented Histogram

F0 – Pitch Frequency

FACS – Facial Action Coding System

FDHH – Feature Dynamic History Histogram

FER – Facial Expression Recognition

FMD – Facial Mesh Distances

GMM – Gaussian Mixture Model

HCI – Human-Computer Interaction

HNR – Harmonics to Noise Ratio

HOF – Histogram of Optical Flow

HOG – Histogram of Oriented Gradients

HOS – Histogram of Shape Index

HSOG – Histogram of Second Order Gradients

iPar-CLR – Incremental Parallel Cascade of Linear Regression

KNN – K-Nearest Neighbour

LBP – Local Binary Pattern

LBP-TOP – Local Binary Pattern – Three Orthogonal Planes

LLD – Low-Level Descriptors

LM83 – 83 Facial Landmarks

LPQ – Local Phase Quantisation

LPQ-TOP - Local Phase Quantization - Three Orthogonal Planes

LR – Linear Regression

MAE – Mean Absolute Error

MBH – Motion Binary Histogram

MBP – Motion Binary Pattern

MDD – Major Depressive Disorder

MEI – Motion Energy Images

meshHOG – Histogram of Mesh Gradients

meshHOS – Histogram of Mesh Shape Index

MFCC – Mel-Frequency Cepstral Coefficients

MHH – Motion History Histogram

MHI – Motion History Image

MMHH – Multi-scale Motion History Histogram

mRMR - Minimum-Redundancy Maximum-Relevancy

MSD – Motion Statistical Distribution

MSE – Mean Squared Error

NLBP – Number Local Binary Pattern

PCA – Principal Components Analysis

PCC – Pearson’s Correlation Coefficient

PHQ 8 & PHQ 9 – People Health Questionnaire

PLS – Partial Least Squares

POV – Probability of Voicing

PSFD – Primitive Surface Feature Distribution

RF – Random Forest

RMSE – Root Mean Squared Error

SD – Statistical Distribution

Seq-CLR – Sequential Cascade of Linear Regression

SMHH – Spatial Motion History Histogram

SMMHH - Spatial Pooling Multi-scale Motion History Histogram

SVM – Support Vector Machine

TSMHH - Time-scaled Motion History Histogram

ULBP – Uniform Local Binary Pattern

ZCR – Zero Crossing Rate

List of Publications

Journal Article (Published)

1. **A. Jan**, H. Meng, Y. F. A. Gaus, and F. Zhang, "Artificial Intelligent System for Automatic Depression Level Analysis through Visual and Vocal Expressions," *IEEE Trans. Cogn. Dev. Syst.*, vol. PP, no. 99, pp. 1-13, 2017.

Conference Articles (Published)

1. **A. Jan**, Y. F. A. Gaus, F. Zhang, H. Meng, and F. Zhang, "BUL in MediaEval 2016 Emotional Impact of Movies task," in *MediaEval 2016 Multimedia Benchmark Workshop Working Notes Proceedings of the MediaEval 2016 Workshop*, vol. 1739, 2016.
2. Y. F. A. Gaus, T. Olugbade, **A. Jan**, R. Qin, J. Liu, F. Zhang, H. Meng, and N. Bianchi-Berthouze, "Social Touch Gesture Recognition using Random Forest and Boosting on Distinct Feature Sets," In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction* (ICMI '15). ACM, New York, NY, USA, pp. 399-406, 2015.
3. Y. F. A. Gaus, H. Meng, **A. Jan**, Fan Zhang, and S. Turabzadeh, "Automatic affective dimension recognition from naturalistic facial expressions based on wavelet filtering and PLS regression," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 05, pp. 19–24, 2015.
4. **A. Jan** and H. Meng, "Automatic 3D facial expression recognition using geometric and textured feature fusion," in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 05, pp. 25–30, 2015.
5. **A. Jan**, H. Meng, Y. F. A. Gaus, F. Zhang, and S. Turabzadeh, "Automatic Depression Scale Prediction using Facial Expression Dynamics and Regression," in *Proceedings of the 4th International Workshop on Audio/Visual Emotion Challenge - AVEC '14*, 2014, pp. 73–80.

Conference Articles (Accepted)

1. **A. Jan**, H. Ding, H. Meng, L. Chen, and H. Li, "Deep Facial Expression Recognition using Localized Facial Parts", in *2018 13th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2018

Chapter 1. Introduction

1.1 Research Content

Human-computer interaction (HCI) is becoming a leading role in everyone's day to day life activities. We are entering a generation where documents are no longer needed to be filled in by hand, where a lot of activities are accomplished from home via PC. The two major areas where HCI research is going towards are on biometric data and emotion detection. There are a substantial number of applications that can benefit from them in many sectors from business to medical and consumers.

Emotions play a critical role in rational decision-making, perception and human interaction. Emotion can be described in several ways. The simplest way is the six basic universal expressions (Anger, Disgust, Fear, Happy, Sad and Surprise) proposed by Ekman et al. [1]. They are called discrete emotion categories. Another way to describe emotions is by using a dimensional space such as arousal, valence, and others. They are called emotion dimensions [2], [3]. Arousal is understood to be the measure of activation, which ranges from high or low. Valence is a measurement of the pleasantness, which ranges from pleasant to unpleasant. The combination of these two measurements can produce a high-dimensional space where any point in that space can represent a type of emotion. This concept is visualised in Figure 1.1, demonstrating a dual axis representation of the possible emotions through combining arousal and valence.

The task for emotion detection is to predict the one of the discrete emotions, or a point in the arousal-valence dimensional emotion space. For artificially intelligent systems, they achieve this by learning various patterns and features from recorded data of different modalities. However, some of these artificial intelligent systems ignore some of the affective states that motivate human actions and communications, as their main objective is to look for specific emotions. Therefore, systems are being developed to understand more emotions than the basic expressions [4], and capturing the affective state of the person can bridge and improve the response of the system.

A humans emotion can be observed through facial expressions, voice, body gesture, touch gesture, and biosensors such as GSR, heart rate, EEG, etc. However, facial expressions are the most significant modality to represent a human's emotion [5]. They are strongly linked to the representations of the emotion due to the expressive behaviour of the face, with the possibility of capturing rich content in multi-dimensional views. Facial expressions provide the cues of communication in which we can interpret the mood, meaning and emotions at the same time. These expressions are used as a tool to interact with others to portray their mood and emotion, and therefore, a vital area of research for HCI. Existing research [6], [7] has exploited the facial model to try and identify these emotions, with more and more active research continually improving FER.

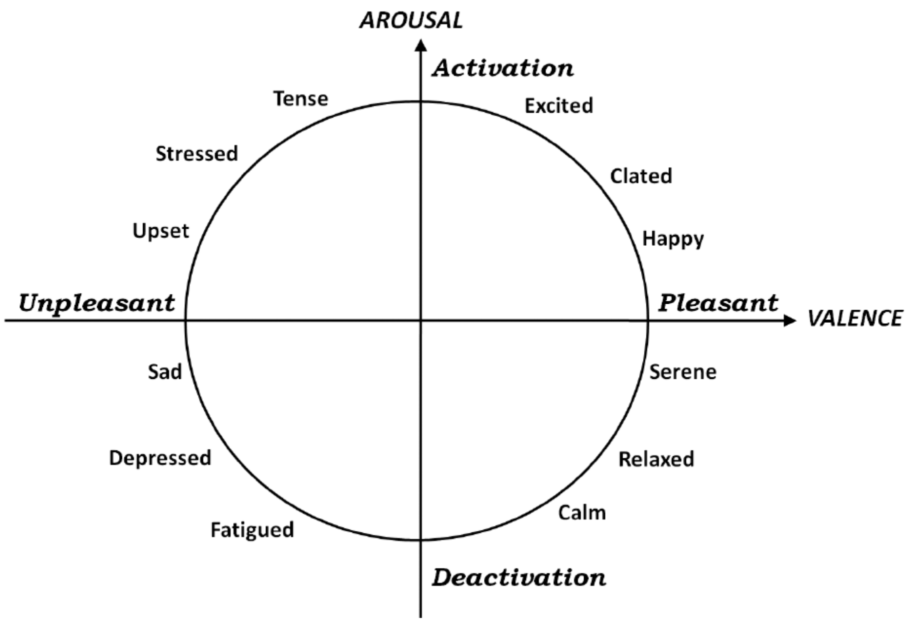


Figure 1.1 - Dimensional representation of Arousal and Valence that can represent many emotions [3].

This research aims to detect and determine emotions from human facial expressions automatically through image and video processing techniques. These emotions are based on various applications that try to understand a human’s emotional state, such as facial expression recognition, Depression analysis etc. Developing advanced artificially intelligent systems will require face image analysis, feature extraction techniques, modelling, and machine learning techniques. It is a research area involving computer science, psychology, etc.

1.2 Current Research and Limitations for FER

The process of facial expression recognition initially requires a professional to observe and classify samples before they are provided to a machine for learning. However, the accuracy of a professional is not always guaranteed. There are only six basic expressions (Happy, Surprise, Fear, Sad, Angry and Disgust) for FER that are chosen as the objectives even though there are thousands more. This demonstrates that humans find it difficult recognise complex emotions like anticipation or remorse, which in-return limits the capabilities of machines.

The issues with current computing environments are that the human-computer interaction designs involve using traditional interface devices such as the keyboard and mouse. They are “*constructed to emphasize the transmission of explicit messages while ignoring implicit information about the user, such as changes in the affective state*” [4]. Existing datasets on facial expressions tend to have their expressions acted out rather than being naturalistic. This is a limitation as the true emotion is not fully conveyed, and therefore, when a system is used in a real environment it may struggle to identify the naturalistic expressions properly. The traditional images used in research for many years have been based on 2D static models of the face. This form of the image also has its limitations that have continuously challenged researchers.

1.2.1 2D Facial Models

Even with the current state of the art techniques, there are problems with 2D imaging that cannot be fixed, making it difficult for face and facial expression recognition. Some of these problems are:

- Pose Variations – they can cause big problems for any image processing that requires the face. Pose variations such as the rotation of the head in 2D imaging would mean that a significant portion of the face detail can be lost. This would make the image processing techniques difficult to benefit from all the facial features, effectively making some algorithms inaccurate.
- Occlusions – these are objects and variations of the facial image causing only part of the face to be visible. This is a big problem especially for face recognition as any slight facial appearance change can make it unrecognisable to a machine. Examples of occlusions can be, glasses, hats, scarfs, facial hair etc.
- Illumination changes – mainly caused by lighting conditions when an image is captured. If there is a change in lighting, images of the face can appear differently. This effect can cause various types of issues in the image like causing shadows, or completely make part of a face black. A demonstration of this effect is visible in Figure 1.2. A lot of light can cause part of the face to be bright and highlighted. Visually for humans, we may still be able to see the face. However, for a computer, shadows on a face will produce higher or lower intensity values compared to the unaffected areas.
- Facial ageing – This occurs naturally for every human in their lifetime. The problems it causes are the facial structure, which changes slightly throughout the early years. In conjunction with ageing, the skin texture becomes rougher with lines mainly on the forehead and beside the eyes.

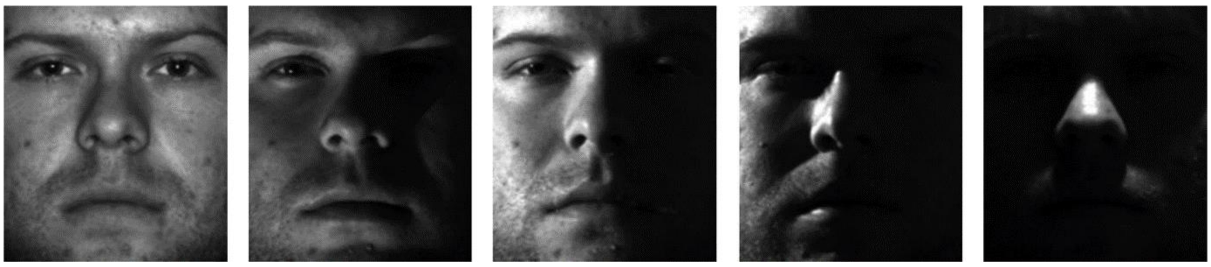


Figure 1.2 - The effects of illumination changes at different angles on a face [8]

1.2.2 Static 3D Facial Models

3D imaging can be computationally expensive to work with compared to 2D images. However, they do provide a more accurate and detailed description of the face. They give the depth parameter of the face, which is important for face and facial expression recognition as the facial structure has muscles that cannot be viewed clearly with 2D images like a concave or convex. For facial expression analysis, the muscular movement of the face can significantly help recognise what expression is portrayed, and these muscular movements can be captured in detail with 3D imaging.

With technology that is evolving facial imaging to other dimensions, 3D imaging can be approached towards amending many of the 2D image processing issues. Illumination changes is not an issue for 3D imaging, as it does not deform the face and the geometric data does not differ based on lighting. The 2D textured data can. However, with most 3D systems, there are multiple cameras that take the pictures from different angles and not all of them would have lighting issues. Therefore, the texture can be reproduced with the unaffected images. Romdhani et al. [9] demonstrate how 3-Dimensional information can improve face recognition, even with the pose and illumination variances.

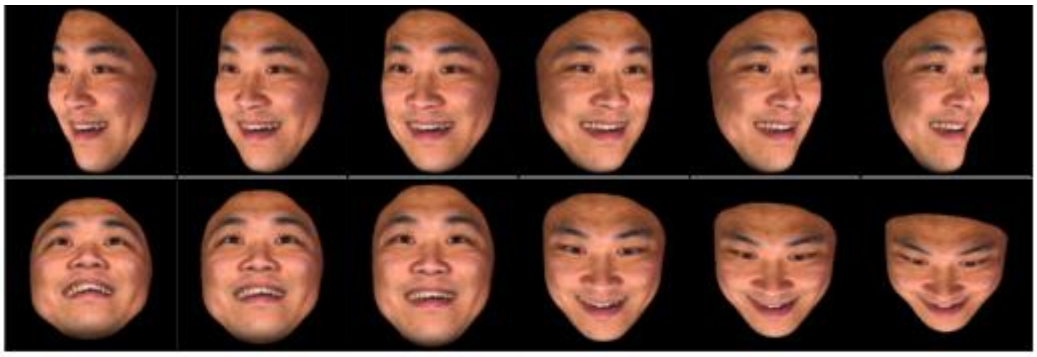


Figure 1.3 - 3D happy facial expression images with different head rotation. Top row is yaw rotation and bottom row is pitch rotation with angles of −40◦, −30◦, −20◦, 20◦, 30◦ and 40◦ going from Left to Right [10].

Pose variations are also eliminated as 3D images are captured to cover the majority of the face and head. Therefore, the face can be rotated back to show the frontal view of the face. Wang et al. show a prime example in [10]; using their method Primitive Surface Feature Distribution (PSFD); of how changing the pitch and yaw rotation of the head (Figure 1.3), can still produce similar results (Figure 1.4) compared to the normal position with no rotation. Figure 1.4 also demonstrates how 2D techniques on the facial texture still get effected.

1.2.3 Temporal Facial Models

Temporal data on facial models can come in 2D and 3D models. The time dimension is added to provide more information for tasks like emotion, gesture and facial expression recognition. The 2D dynamic data provides a different approach for displaying information. This is required for tasks such as expressing a gesture [11] where motion needs to be portrayed, or showing signs of depression which is visible over a period of the subject participating in an activity [12], [13]. Using this data would require a considerably higher amount of computational resources compared to static data.

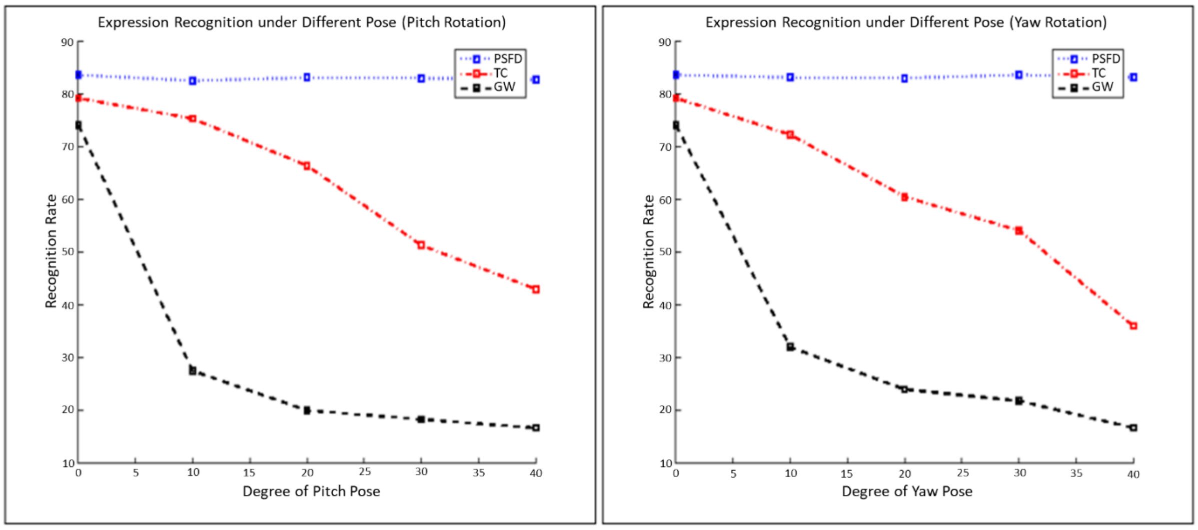


Figure 1.4 - Graphical results of the Recognition Rate achieved by PSFD across different pitch and yaw rotations [10]

3D temporal data is the latest development for researchers to experiment with, it provides a higher depth of information that many applications can benefit from. Facial Expression Recognition is a good example that can benefit from the 3D temporal models, as the data can provide build up to an expression using information about how the facial muscles change throughout the expression.

Binghamton University have produced a database that contain temporal 3D models, which is called Binghamton University – A 3D Dynamic Facial Expression Database (BU-4DFE) [14] and their latest, Binghamton-Pittsburgh 3D Dynamic Spontaneous Facial Expression Database (BP4D-Spontaneous) [15] which is 2.82 Terabytes containing spontaneous 3D temporal expressions made from geometric models with the texture mapped on.

Other forms of emotion can be determined using the Arousal and Valence dimensions. Such applications can include Depression analysis, violence detection and inducing emotions. These tasks have been approached through means of challenges [12], [13], [16]–[18] to get the research community to apply their knowledge into areas of emotion that can be benefitted. These challenges provide large amounts of temporal data that look to capture affective emotions [19], [20] of video clips. The MediaEval workshop in 2015 had shown interesting methods to detect violence and the induced affective emotions by the videos [21]–[24]. In 2016 [17], the challenge of predicting induced emotions [25]–[28] was continued. The AVEC challenges have continued on a yearly basis to allow the state-of-the-art techniques and ideas be adapted for this task. These are all areas in development that will continue to be researched and eventually applied for medical, commercial and personal practises.

1.3 Motivations and the Approach

There has been a lot of work done recently for emotion-based applications such as facial expression recognition, using 2D and 3D images that come in static and temporal forms. Facial pre-processing steps such as face localisation and alignment are also being researched to provide improvements over new and existing systems. There are many existing 3D techniques based on geometric information, as well as the traditional 2D techniques for processing faces. However, there is still room to understand how both areas can complement each other to produce robust and efficient frameworks.

Temporal data is also further explored as it is a richer source of information compared to static data. Many of the tasks for applications such as Depression analysis require a sequence of information as it is hard to estimate such an emotion based on a single image. There are a few motion-based techniques such as Motion binary patterns [29] and Motion history histogram [30] that observe and capture movement across a sequence of frames. This idea can and is explored further to capture richer motion information by viewing the sequence in more than one way. These ideas can also be compared to the spatiotemporal techniques that are based on spatial images but have been moved to the temporal domain, such as LBP-TOP [31] and LPQ-TOP [32]. We also investigate if a sequence of image frames represented instead as a sequence of mathematical representations can improve a system. This approach is achieved by developing an algorithm that can capture feature variations across the sequence of mathematical representations, just as motion is observed across a sequence of image frames.

Deep learning is also a key area that has taken of recently, with the heavy computational requirements now a viable option thanks to the powerful GPUs available. Convolutional Neural Networks introduced a complex black box approach to understand the relations and content within sets of images, making it a big success for tough challenges such as large-scale object detection on the ImageNet database [33], [34]. This approach started off with hand-written digit recognition by LeCun et al. [35] and has expanded to advanced applications including emotion based with recent works in [17], [36]–[38]. Based on these ideas, the motivation is to try and produce advanced techniques to further improve FER, and other emotion-based applications through frameworks that can utilise deep learning techniques. With all the available pre-trained deep networks on large databases, their parameters can be exploited to also benefit these applications.

1.4 Aim and Objectives

The aim of this thesis is to use advanced techniques to improve the current state of human emotion detection using facial expressions. The following objectives are undergone throughout the thesis:

- Investigate and understand existing techniques for FER based on 2D and 3D information
- Determine facial areas that have significant contribution towards each expression
- Use temporal data to produce a dynamic descriptor for emotions that vary across time
- Integrate state-of-the-art deep learning techniques into frameworks that can predict the emotions of a person

1.5 Thesis Contributions

The thesis provides research works with the following contributions:

- Chapter 3 considers 3D + 2D facial expression recognition with the fusion of both domains to classify 6 basic expressions.
- Chapter 4 creates automatically detected facial parts using advanced face pre-processing algorithms, and uses deep learning techniques to learn these facial parts, along with the whole face. Innovative architectures are designed to combine the efforts of the facial parts with a Joint Bayesian approach in the form of metric learning.
- Chapter 5 investigates motion-based descriptors which are modified to extend their capabilities by capturing advanced motion information. Several extensions are proposed and tested against the Motion History Histogram [30] descriptor, demonstrating competing performances on motion intensive and emotion based applications.
- Chapter 6 develops a novel algorithm called Feature Dynamics History Histogram to capture variations in a temporal sequence of feature vectors. This is applied to a Depression analysis application and to capture induced emotions through movies. A constructive framework is designed using CNNs as a pre-trained feature extractor at frame-level on a visual sequence, from which the FDHH algorithm is applied to capture patterns of feature variations. At the time of testing and publication, state-of-the-art results are achieved.

1.6 Thesis Outline

The remaining chapters of this thesis are organised in the following order:

Chapter 2 provides a detailed review of current technology and techniques in the area of facial expression recognition and other emotion-based applications such as Depression analysis and induced emotions. The chapter also contains the details of various hand-crafted descriptors, as well as machine learning techniques and processes.

Chapter 3 contains an investigation into 3D and 2D static image techniques for facial expression recognition. The BU-3DFE and Bosphorus 3D databases are utilised with the extraction of unique texture and geometric features. These descriptors are compared and fused together to demonstrate the effect of using multiple domains in a single framework.

Chapter 4 introduces deep learning techniques to extend the works in Chapter 3. The use of isolated facial parts is also investigated, using a combination of sophisticated geometric techniques for face localisation, alignment and normalisation. The use of facial parts is compared to using the whole face using hand-crafted and deep learning techniques. Joint Bayesian is also investigated in the form of metric learning, which is integrated into the learning process of CNNs.

Chapter 5 considers how motion can be modelled from temporal data. The popular Motion History Histogram descriptor [30] is extended to provide different abilities for better motion capture. These extensions are thoroughly tested and compared on various applications such as human action recognition and Depression analysis.

Chapter 6 applies deep learning techniques on temporal content, demonstrating strategies across emotion based applications. An algorithm is also developed to work with the mathematical representations of each frame in a sequence, by observing patterns of variations throughout the sequence. This technique is applied in a framework that produces state-of-the-art performances for emotion-based applications, using constructive frameworks that exploit deep features in various ways.

Chapter 7 concludes the main contributions achieved by the thesis, and the experimentation conducted for it. The possibility of future works is also detailed.

Chapter 2. Literature Review

In this chapter, various works related to human emotions, especially for human emotion recognition from facial expressions are reviewed. It will describe the different forms of computing tools be used for facial expression recognition. Existing works will be presented to understand what is currently being researched and used as solutions for determining these emotions.

2.1 Human Emotions Representation

Computers play an essential role in life in the 21st century and are everywhere around us. Computers are beginning to gain the ability to recognise emotion and soon enough may be given to "have emotions" [39]. The history of emotions theories reach as far back as the Ancient Greek Stoics, Plato and Aristotle [40]. Aristotle's famous Aristotelian theory of emotions, explores the development of his thought on emotions through defining, explicates, compare, contrast various emotions and characterises emotions themselves. His theory led to remarkable observations: "*Emotions are the things on account of which the ones altered differ with respect to their judgments, and are accompanied by pleasure and pain: such are anger, pity, fear, and all similar emotions and their contraries.*" [41]

Many everyday applications now require the use of biometric data, from logging in to sensitive data using fingerprint scanning technology to being allowed into a country using facial recognition via passport photo. The main purpose is to identify the person more securely than, for example, a password. This technology allows machines to handle requests without the supervision of humans, providing a streamlined process and the prevention of forgery or fraud to occur with stricter security in place. Poli et al. [42] explained the techniques used for biometric identification, which included parts of the human body such as the face, iris, voice, and fingerprints. They stated the benefits of biometric which include:

- No more forgotten passwords, lost cards or stolen pins. You are your own password.
- Positive Identification-It identifies you and not what you have or what you carry.
- The highest level of security, and offers mobility.
- Impossible to forget, serves as a "Key" that cannot be transferred or coerced.
- Non-intrusive, safe & user-friendly.
- Increased security when controlling access to confidential data and IT systems.
- Reduced risk of fraudulent use of identity by employees.

These are all advantages that can improve the current system of passwords and ID cards. Some of the Disadvantages mentioned by Poli et al. are [42]:

- An automatic personal identification system based solely on fingerprints or faces is often not able to meet the system performance requirements.

- In case of face recognition, the face will sometimes change with time or injury, and that poses a problem. Fingerprint verification is reliable but inefficient in database retrieval.
- Some voice recognition systems have some problems since the voice changes with a human's mood and illness and background noise pose some problems.

These are all valid disadvantages. However, they can be overcome as technology and research improve. Systems are being able to process significantly faster every generation. Emotions are linked to another division of HCI called affective computing. The target for behavioural scientists is to understand the social signals we produce from our actions. Computer Vision researchers use this information and try to interpret them in a way that can be fed into machines. Currently, the devices we use do not fully understand how human emotions play a role in decision making. This is an area that is still under development with many researchers striving for a solution to automatically detect affective emotions [5], [43], [44]. Knowing the emotional state of a person can affect a lot of decisions in areas like:

- Detecting pain where vocals cannot be expressed. An example is in an operation theatre for radiotherapy where the doctor cannot be present, the pain can be severe that the patient is left speechless with just a shock expression on his face.
- Detecting depression and aggression of subjects to give early signs that can prevent any incidents occurring. Staff will have a better judgement of the patients' behaviour if they have data of their affective emotions available.
- Using machines to find early signs of autism, and other disabilities that show signs of weakness or differentiation compared to a healthy person.
- Customer Service Satisfaction, if an operator is able to tell the emotional state of the customer, it can prevent questions that will provoke the customer further.
- For daily use, vehicles can display warnings advising drivers to pull over or automatically locate the nearest resting place if it can detect when a person is fatigued or sleepy.

These are just a few examples where being able to understand emotions via HCI can aid the lives of the vulnerable and improve the quality of the medical sector. Further research is needed in this field in order to get to a future where machines can understand how to communicate and handle humans, providing us support on demand wherever needed, but mainly interacting with humans a lot more effectively.

2.1.1 Facial Expressions

Facial Expressions are important tools used to communicate the emotional reaction and/or state of a person during their daily activities. There are many expressions a human can display and behind each emotion are a group of components that control the intensity of the expressions. These are the person's intentions, action tendencies, appraisals, other cognitions, neuromuscular and physiological changes, expressive behaviour, and subjective feelings [45]. These components cause the movement of the facial muscles which in return creates a visual expression for others to see the emotion.

From the vast range of human emotions, there are 6 basic facial expressions noted by Ekman and Friesen [46] which are Happy, Surprise, Fear, Sad, Angry and Disgust as shown in Figure 2.1. There are many other emotions associated with facial expressions, but they are mainly small variations of the basic expressions.

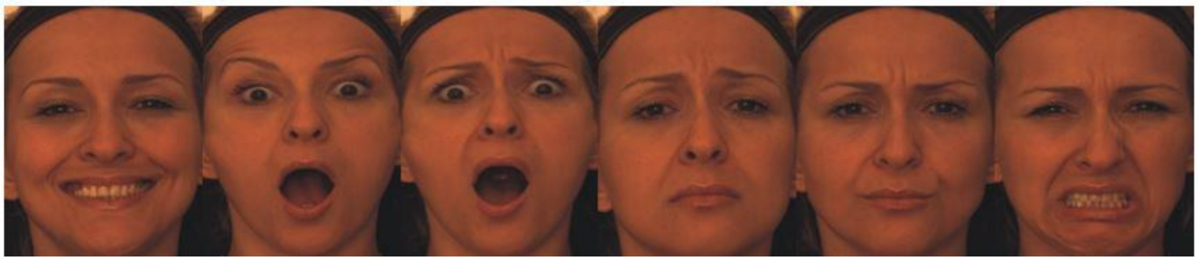


Figure 2.1 – Image of 6 basic expressions taken from the Bosphorus Database

A recent active research area in affective computing has the objective to try and computationally model Major depressive disorder (MDD) through the patient's facial expressions. It is a mental health disability that affects the mind and behaviour of a person, leaving them in a low mood. It is defined as *"a common mental disorder that presents with depressed mood, loss of interest or pleasure, decreased energy, feelings of guilt or low self-worth, disturbed sleep or appetite, and poor concentration."* [47]. The most recent report by the World Health Organisation indicates that MDD is the largest form of disability in the world (over 300M people), affecting 7.5% of people that have a disability [48]. It is estimated that females have a 50% higher rate in suffering from depression than males do [49]. Depression has been a key research area in the affective computing area to try and understand and determine the severity of depression that patients suffer.

Currently, there are a few comprehensive ways to assess the severity of depression in a person that can be translated to aid artificial intelligence. These are in the form of questionnaires such as the BDI-II [50], PHQ-8 [51] and PHQ-9 [52]. Professionals can determine the severity of depression based on the score produced from the questionnaires, and a cut-off point that separates the various levels of depression. These allow for computer vision techniques to map a patient's emotions and facial expressions to the scales that determine their depression level.

2.1.2 Typical FER System

In order to achieve a good FER system, an understanding is required for how a typical system is designed, and what stages are required to go from image to expression. The architectures for these typical systems consist of 3 main building blocks that are universal across many applications including Depression analysis and FER, as shown in Figure 2.2. The following section will describe these blocks in detail.

The traditional design of most systems that work with facial images, or of objects, can be described using 3 main building blocks. The first block directly handles the input which can be raw images, audio or visual sequences. This block applies pre-processing techniques to the data to prepare it for the following feature extraction stage. For facial images, these pre-processing steps can include face detection, alignment, normalisation, augmentation, with many other techniques available and under development. Once all the necessary steps required by the system are complete, the next building block extracts features from the pre-processed samples.

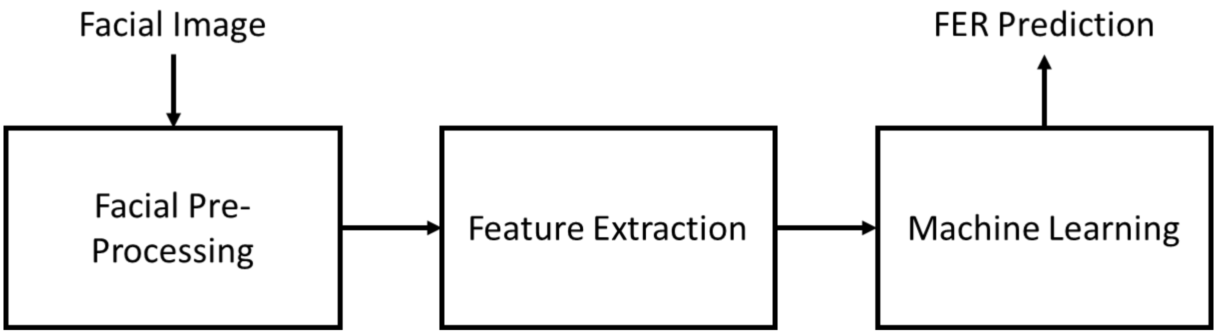


Figure 2.2 - Building blocks for a typical FER system

There are many different techniques and approaches that exist to extract features from the sample images, some of which are detailed later. These approaches can extract features in the form of appearance information, geometric information, temporal and spatiotemporal information. Once these features are obtained, the final building block will try to learn them using machine learning techniques; also mentioned later in this chapter. To get a better understanding of these systems, the following sections will go deeper into the various existing feature extraction techniques used on different modalities of data. This will also include the popular machine learning techniques widely used in diverse applications.

2.2 Feature Extraction Techniques

The upcoming sections will describe and demonstrate feature extraction techniques for image and image sequences. The image techniques use spatial information and are either globally or locally applied. The goal of this section is to understand some of the existing techniques out there that can be used to improve the current state for FER and applications that deal with facial expressions.

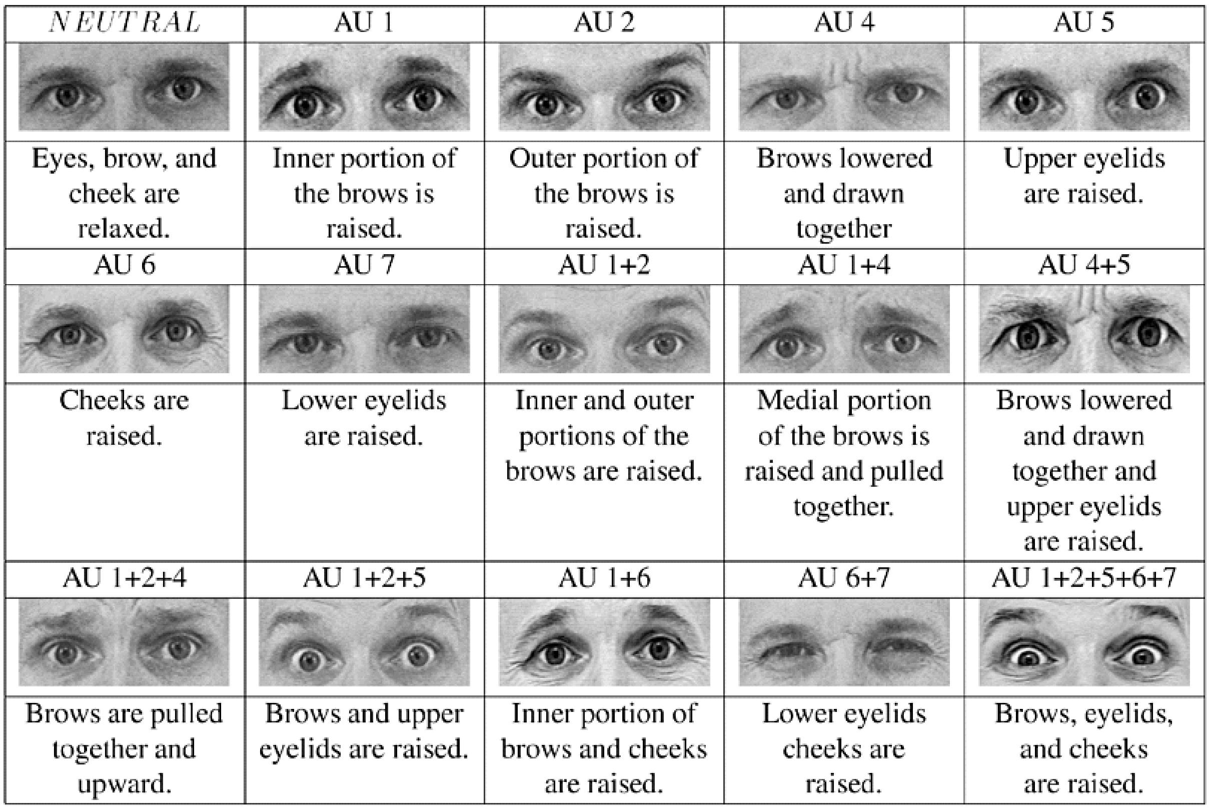
2.2.1 Global Spatial Image Feature Extraction Approaches

The following section will describe some of the techniques of spatial feature extraction which is computed globally across the image. These techniques can be applied for FER or other tasks that work with facial images. The Global approach uses the whole face as a region to work on, rather than breaking the image down into smaller patches.

2.2.1.1 Facial Action Coding System

The Facial Action Coding System (FACS) was first developed by Ekman and Friesen [53] to distinguish between facial expressions. Since then, it is commonly used in facial expression recognition because of how detailed the FACS is and how it can be interpreted by others to suit their tasks. The system works on spotting a single or multiple facial muscle movements called action units (AUs), these are referenced back to a catalogue that indicates what expression it represents. There are 46 different AUs such as raising the left eyebrow or the jaw clenching, the default position is the neutral face [54]. A visual example can be seen in Figure 2.3 of the action units captured from the upper face.

FACS is a reliable way of recognising a large set of expressions. However, a big issue with this system is finding professionals to judge the AU and expression, which can be difficult and expensive. Research has gone into this field to try and overcome the challenge to detect the AUs automatically from subjects, whether it is temporal or spatial-based data. There have been many public challenges out there [55], [56] inviting others to create systems that tackle this problem, showing good results in [32], [57].



NEUTRAL	AU 1	AU 2	AU 4	AU 5
Eyes, brow, and cheek are relaxed.	Inner portion of the brows is raised.	Outer portion of the brows is raised.	Brows lowered and drawn together	Upper eyelids are raised.
AU 6	**AU 7**	**AU 1+2**	**AU 1+4**	**AU 4+5**
Cheeks are raised.	Lower eyelids are raised.	Inner and outer portions of the brows are raised.	Medial portion of the brows is raised and pulled together.	Brows lowered and drawn together and upper eyelids are raised.
AU 1+2+4	**AU 1+2+5**	**AU 1+6**	**AU 6+7**	**AU 1+2+5+6+7**
Brows are pulled together and upward.	Brows and upper eyelids are raised.	Inner portion of brows and cheeks are raised.	Lower eyelids cheeks are raised.	Brows, eyelids, and cheeks are raised.

Figure 2.3 - Facial Action Units example on the upper face, where each action unit is demonstrated, along with the combinations of them [58].

2.2.1.2 Gabor Filters and Wavelets

Gabor filters have been mentioned to be closely linked to the primary visual cortex, they are linear filters that are used mainly for edge detection. It was first introduced by Gabor et al. [59] in 1945 and since then has played a major role in many computer vision applications. The Gabor filter is typically a bandpass filter, multiplying the odd and even filters shown in Equations 2.1 and. 2.2.

$$g_e(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \cos(2\pi\omega_0 x) \tag{2.1}$$

$$g_o(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \sin(2\pi\omega_0 x) \tag{2.2}$$

where ω_0 defines the centre frequency (i.e., the frequency in which the filter yields the greatest response) and σ is the spread of the Gaussian window. These filters have been shown to possess optimal localization properties in both spatial and frequency domain and thus are well suited for texture segmentation problems. Frequency and orientation representations of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave.

2.2.1.3 Primitive Surface Feature Distribution

Primitive Surface Feature Distribution is based on 3D Geometric imaging of faces and structures. It was developed by Wang et al. [10] for 3D FER along with the release of their 3D Facial Database called BU-3DFE Database [60].

The data provided is a point cloud with a triangular mesh, eliminating any illumination and orientation problems that can occur during capture. The data points are 300 microns apart to provide a more detailed structure of the face, a smooth local fitting procedure is applied on the mesh so the minimum λ_1 and maximum λ_2 degrees of bending can be obtained, which is (a) and (b) in Figure 2.4 respectively.

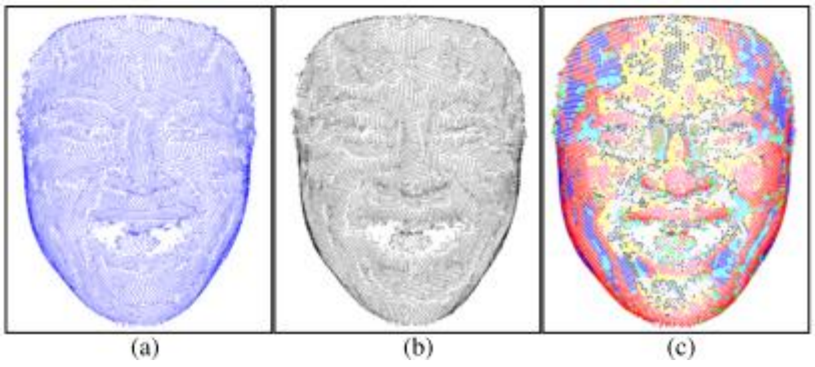


Figure 2.4 - (a) is the minimum curvature direction map, (b) is the maximum curvature direction map and (c) is the primitive label map produced by principal component analysis [10]

Principal Curvature Analysis is used on the newly generated surfaces to split the surface into different regions. Each of these regions is labelled depending on how the feature is. For example, a ridge, ravine, peak, pit, saddle, concave hill, convex hill, etc. This is illustrated as (c) in Figure 2.4. These labels are determined by a set of rules explained in Table 2.1 [10].

Table 2.1 - Classification rules to determine the primitive 3D surface labels [10]*, where λ_1 and λ_2 are minimum and maximum curvature direction maps and T_λ is the threshold to evaluate whether the principal curvatures are insignificant enough to be set as 0.*

λ_1	λ_2	Hillside Label	Non-Hillside Label
$\lvert\lambda_1\rvert < T_\lambda$	$\lvert\lambda_2\rvert < T_\lambda$	Flat	Slope hill
$\lambda_1 < -T_\lambda$	$\lambda_2 < -T_\lambda$	Peak	Convex hill
$\lambda_1 < -T_\lambda$	$\lvert\lambda_2\rvert < T_\lambda$	Ridge	Convex hill
$\lambda_1 < -T_\lambda$	$\lambda_2 > T_\lambda$	Ridge saddle	Convex saddle hill
$\lambda_1 > T_\lambda$	$\lambda_2 < -T_\lambda$	Ravine saddle	Concave hill
$\lambda_1 > T_\lambda$	$\lvert\lambda_2\rvert < T_\lambda$	Ravine	Convex hill
$\lambda_1 > T_\lambda$	$\lambda_2 > T_\lambda$	Pit	
$\lambda_1 > T_\lambda$	$\lambda_2 < -T_\lambda$		Concave saddle hill

2.2.1.4 *Principal Components Analysis*

Principal Components Analysis (PCA) is a well-known technique used to for dimensionality reduction. It decorrelates the high dimensionality data whilst trying to retain high variance within the data. This is achieved by grouping the highly-correlated data together in a lower-dimension subspace that is made from the principal components. PCA is commonly applied to data before the machine learning stage. This has the benefit of speeding up the training process, as there are fewer dimensions to the data. The number of degrees of freedom is also reduced, and for some machine learning techniques, this can prevent overfitting occurring.

PCA essentially projects a set of data into a lower dimensional subspace. These projections are called the Principal Components. To compute PCA, there are a few steps taken to get to the end set of dimensions. First, we start with a data set containing p observations of an n-dimensional vector $\mathbf{x} = [x_1, x_2, \dots, x_n]^{\mathrm{T}}$. This vector requires having a total mean of zero. This is achieved by calculating the mean of each observation p subtracting the mean $\mu_{\mathbf{x}}$ of data X across each component as follows:

$$\mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.3}$$

$$C_{\mathbf{x}} = \sum_{i=1}^{n}(x_i - \mu_{\mathbf{x}})(x_i - \mu_{\mathbf{x}})^T \tag{2.4}$$

This is followed by calculating the covariance matrix C_x, from which the eigenvalues and eigenvectors are obtained from the data. The eigenvectors correspond to the dimensions that contain the highest variance within the dataset. For dimensionality reduction, only some of the highest eigenvectors are kept as the new feature; which is generally balanced between the number of eigenvalues against the total variance kept.

2.2.2 Local Spatial Image Feature Extraction Approaches

The following section will describe the many different techniques for describing local image feature extraction techniques. Most of the techniques have been used widely in the field of computer vision. They have shown to work good in a variety of applications such as face detection [61], facial expression recognition [62], [63], object detection [64], [65], and texture classification [66], [67].

2.2.2.1 Local Binary Pattern

Local Binary Pattern (LBP) is a non-parametric operator that is used to describe the local surroundings of a pixel by producing a pattern from the binary derivatives of the pixel. The algorithm itself is basic when it comes to computational calculations and is robust to monotonic grey change; which is why the operator is usually applied to grey scale images; making it a popular and efficient method for texture analysis.

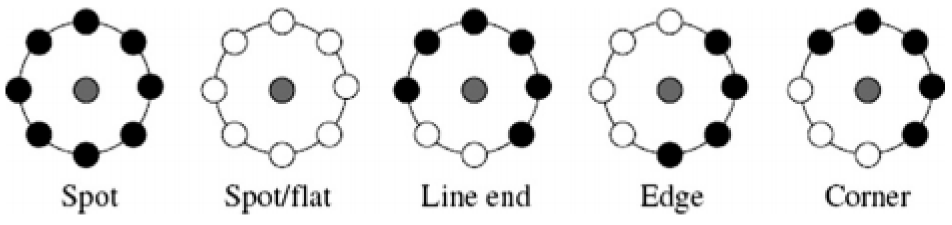
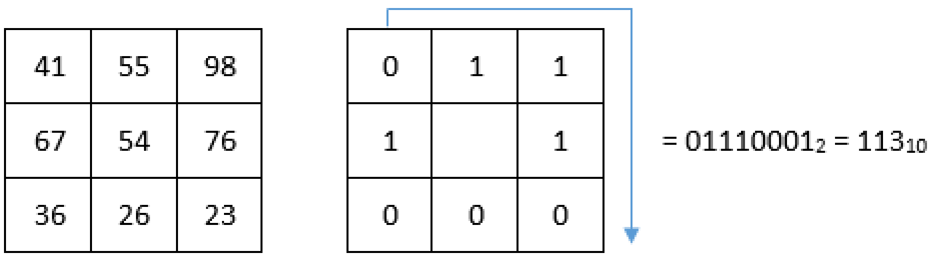


Figure 2.5 – 5 local patterns detected using LBP [68], *each representing a characteristic of a local 9×9 patch.*

Figure 2.5 shows the kind of patterns that can be detected using LBP. With this information, LBP has shown to be useful for face recognition [69] and in facial expression recognition [70]. It has shown tolerance against illumination changes, which for 2D facial models is a major issue.



41	55	98
67	54	76
36	26	23

0	1	1
1		1
0	0	0

$= 01110001_2 = 113_{10}$

Figure 2.6 - LBP Basic 3x3 Window Calculation

Figure 2.6 shows how the binary pattern is calculated for each pixel using its local surrounding pixels, this process can be described using Equation 2.5 [71]. The value m denotes the number of surrounding pixels, the centre pixel m_C, which is 54, is taken as a threshold value and each of the surrounding pixels m_i has the threshold subtracted from them. The function $f(x)$ would determine the output bit assigned to m_i, if x is ≥ 0 then a '1' is assigned, else if x is < 0 then a '0' is assigned. Once all the local surrounding pixels are evaluated then a pattern is created by combining the 8 bits assigned starting from the top left pixel.

$$LBP_i = \sum_{i=0}^{m-1} f(m_i - m_C) \cdot 2^i \qquad f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.5}$$

A drawback with the basic LBP operator is that it is not rotation invariant, making it difficult to use even on slightly rotated images. To fix this, there has been a modification made to prevent rotation causing an issue.

2.2.2.2 Uniform LBP

Uniform LBP (ULBP) extends from LBP by looking at the binary pattern for bit-wise transitions. Given U is the number of changes of binary sequence from '0' to '1' or vice versa, uniform patterns are recorded when a binary number in the sequence changes a maximum two times ($U = 2$). There are 36 rotation invariant patterns which include the 9 uniform patterns, these samples can be seen in Figure 2.7 [72].

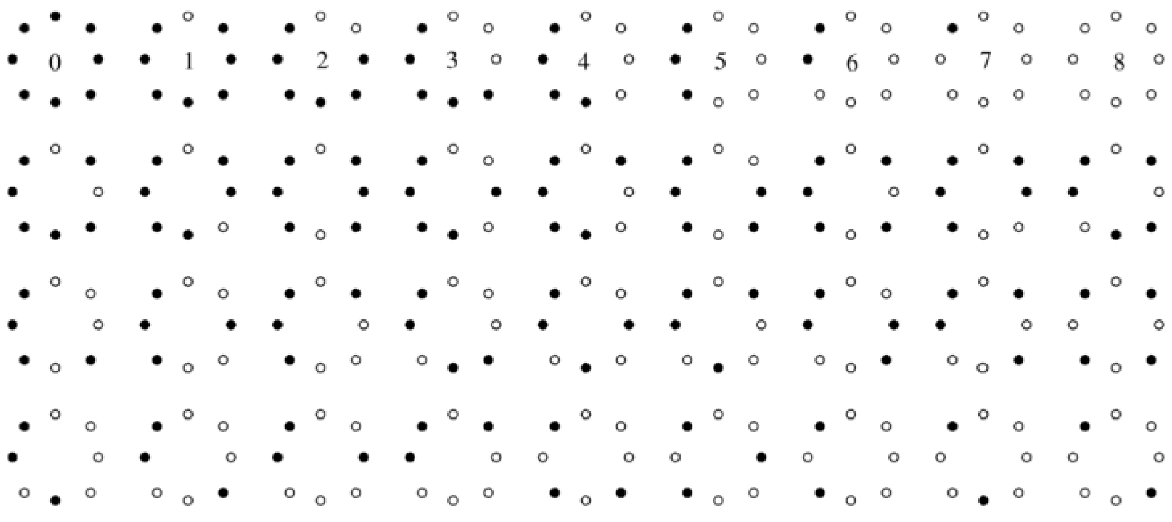


Figure 2.7 - The 36 unique rotation invariant Binary Patterns represented by black and white circles as bit values of 0 and 1

The remaining 27 patterns are when $U \geq 4$, where there are at least 4 bit-wise changes in the pattern. These patterns have been known to have fundamental properties that occur the most when taking LBP on an image.

2.2.2.3 *Extended LBP*

This is an extension of the original LBP code called Extended Local Binary Patterns, which was developed for texture analysis by Zhou et al. [68] in 2008. They demonstrated that a bit of noise can change the uniform patterns into something else that it should not be. To tackle this, the first solution was to assign the closest uniform pattern to every non-uniform pattern. This can be achieved by calculating the minimum sum of the absolute difference between the LBP pattern and all the uniform patterns, the ideal candidate would be the pattern with the most similar pattern and closest distance.

Figure 2.8 illustrates examples of the LBP patterns from which the uniform pattern of 00000111 closely matches. From the selection, the LBP pattern that would benefit this approach the most would be 00000101 as it is the most similar.

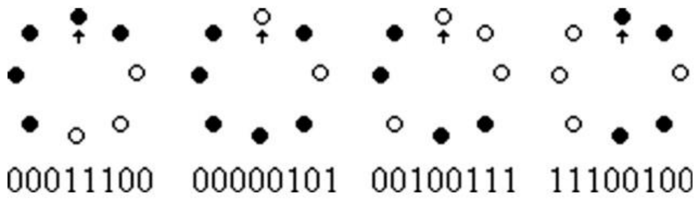


Figure 2.8 - Some local patterns with their most similar uniform pattern of 00000111 [68]

2.2.2.4 *Number LBP*

Number Local Binary Pattern is a variation of LBP created by Yan et al. [71] that provides better texture discrimination when compared to other LBP methods (ULBP and ELBP). According to Yan et al. the ULBPs work well in some cases, but not all because there is no pattern that can describe every texture well enough. To improve on ULBP, they divide the non-uniform patterns further into different groups.

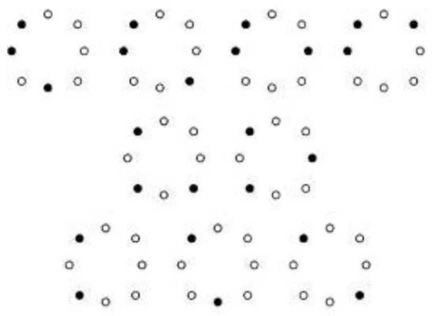


Figure 2.9 - NLBP groups with local similarity

The idea is to group similar looking patterns together as shown in Figure 2.9. This idea transforms closely related patterns to produce 5 more non-uniform groups compared to the single non-uniform group mentioned in ULBP.

2.2.2.5 *Scale Invariant Feature Transform*

Scale Invariant Feature Transform was proposed by Lowe et al. [73], [74] for object recognition. The method will extract features that are invariant to image scaling, translation, and rotation, and partially invariant to illumination changes and affine or 3D projection. The feature itself is a 3D histogram of gradient locations and orientations. To calculate the feature, a grid is placed on the image, and each grid has smaller windows. In each window, the overall gradient approximation is calculated, which is a weight based on the average of all the gradients of each element in the window. Each gradient is based on the centre of the window. All the weights are then made into a histogram.

The orientation of each window is done based on voting, where each orientation is calculated by obtaining the magnitude and the distance of each gradient from the centre of the window. The orientations are also put into a histogram, and all the histograms are normalised and concatenated together to produce a single feature vector. Figure 2.10 shows the process of getting the gradients in each window and producing an approximation of each grid, and then produce a histogram of the different approximations from each grid.

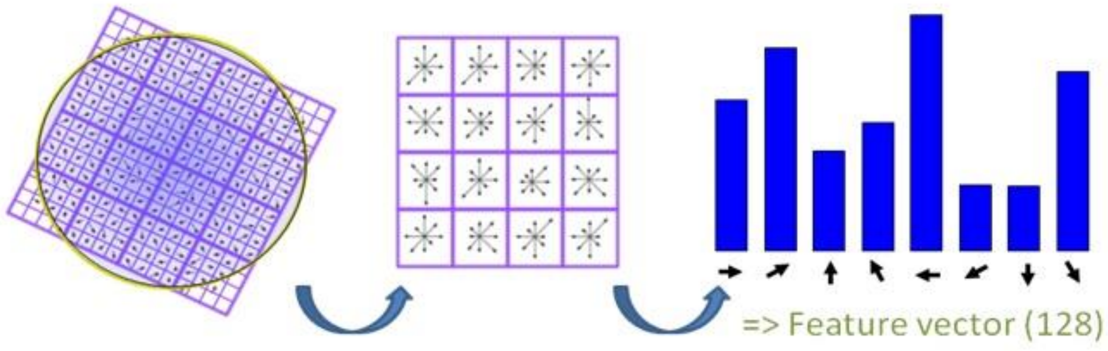


Figure 2.10 - SIFT feature procedure of getting the gradients of each window, making a gradient approximation of each grid and producing a histogram based on them

2.2.2.6 *Edge Orientation Histogram*

Edge Oriented Histogram (EOH) is known as a faster and simpler version of Histogram of Oriented Gradients (HOG), which is a method that produces a feature consisting of gradients across an image. HOG was first used in Hand Gesture Recognition by Freeman and Roth [75], and then for human detection by Dalal et al. [65]. EOH is well known for object and face detection as demonstrated in [76] [77] [74], and is beneficial when considering if computation time and memory is an issue.

The method first converts the original image into grayscale image and is split up into blocks $I(x, y)$. The edges in each block are obtained using Sobel edge operators $\left(K_x \; and \; K_y\right)$ to calculate the horizontal edges (G_x) and vertical edges $\left(G_y\right)$ shown in Equation 2.6.

$$G_x(x, y) = K_x I(x, y) \qquad G_y(x, y) = K_y I(x, y) \tag{2.6}$$

The angle θ and strength $S(x, y)$ of the orientation for the edges are calculated and arranged in a polar coordinate system shown in Equation 2.7 and 2.8. The angles and strengths are then divided into N bins to make the EOH feature.

$$\theta = \arctan\left(\frac{G_y(x, y)}{G_x(x, y)}\right) \tag{2.7}$$

$$S(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)} \tag{2.8}$$

Using blocks with the EOH operator makes it possible to recover the orientation of the image, as the highest bin value will always be at 0 degrees. It is also robust to illumination changes good for detecting borders, as strong gradients moving in different directions will most likely represent a corner of an image.

2.2.2.7 Local Phase Quantisation

The Local Phase Quantisation (LPQ) operator was developed by V. Ojansivu and J. Heikkila [67] for blurred texture classification, in which it is shown to perform better than LBP. It has been used frequently for face recognition [78] [79] [80] [81] showing LPQ works in the frequency domain and splits an image into windows. The short-term Fourier transform is then computed across the local $M \times M$ neighbourhoods; similar to LBP; to obtain the Fourier coefficients. This is applied to all pixels in the image, then a scalar quantizer is used to transform the coefficients into an 8-bit binary code and from that, a histogram is produced.

2.2.3 Temporal Feature Extraction Approaches

The following section will describe methods that produce features from temporal data such as videos.

2.2.3.1 Motion Binary Patterns

Motion Binary Pattern (MBP) [29] attempts to capture the characteristics of motion movement by observing the frame level grey values of the video. Each of the frames is divided into cells. Then 3 cells across 3 frames in a fixed XY position are taken and the first frame cell is compared to the second frame cell. For each pixel in each cell, if the grey value is higher, then a 1 is assigned. If not then a 0 is assigned. The third frame cell is also compared to the second frame cell in the same way. The output from both comparisons is combined using an exclusive OR function which then creates the final motion pattern [29]. Unlike the previous methods, MBP tries to use natural occurring motion. However, it lacks the depth of time information as each frame overwrites it's previous using the XOR function.

2.2.3.2 Motion History Histogram

Motion History Histogram (MHH) tracks motion patterns across a visual sequence. It was produced by Meng et al. [30] which was originally proposed for human action recognition [30], has also shown to be effective for tracking subtle emotions caused by depression [82], [83] and has proven to be better than Motion History Image [84].

P_1='010'

P_2='0110'

P_3='01110'

⋮

P_M='01…10'

M Binary Pattern Sequences

Figure 2.11 - Patterns that are generated based on motion movement [30]

MHH captures the motion from a video in its grayscale form, this is done by detecting how much each pixel moves across the frames. Binary patterns are noted for each pixel depending on how much it has moved across the frames if there is a great deal of motion the pattern becomes longer. In Figure 2.11 you can see the patterns produced based on the length of the motion. If a pixel has multiple values for a period of 3 frames then the pattern for that pixel will be P_3 = 01110. MHH is produced by collecting the all the patterns $P_i, (i = 1, 2, \dots, M)$, where M is the highest pattern recorded. Each pattern contains the count for each pixel, indicating how many times that pattern occurred throughout the video for the pixel. Each pattern P_i, MHH(: , : , i) can then be viewed as a frame so that the motion can be visualised.

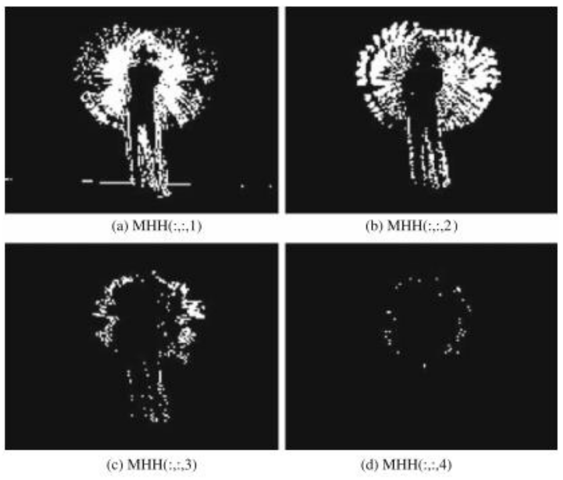


Figure 2.12 - MHH example of a person waving their hands, the resulting patterns are shown from P1 to P4 [30]

Figure 2.12 is an example visualisation of a person waving their hands in the air. MHH is captured where the parameter $M = 4$, each pattern is visualised where P_1 (shown in (a)) shows the small movement that occurs during the gesture. (d) Shows the pattern P_4 occurrences which indicate where there has been significant movement.

2.2.3.3 *Local Binary Patterns – Three Orthogonal Planes*

Local Binary Patterns – Three Orthogonal Planes (LBP-TOP) is an expansion of the original LBP operator for the temporal domain, which can be used on dynamic data like time-based facial expressions. Zhou, Pietikainen et al. [31] introduced their dynamic LBP operator for texture recognition and for facial expressions [31].

LBP-TOP produces small cuboids on the videos, with the 3 axes being X, Y and T (for the time). The radius determines the size of the cuboid, with the default size radius being 1 pixel. It calculates LBP on three Orthogonal Planes XY, XT and YT and then concatenates the three histograms produced from each plane. An example of the process can be seen in Figure 2.13, LBP-TOP being taken from (a) a woman's face, which is split into blocks (b) and the operator is applied on it to produce the histograms for each plane (C).

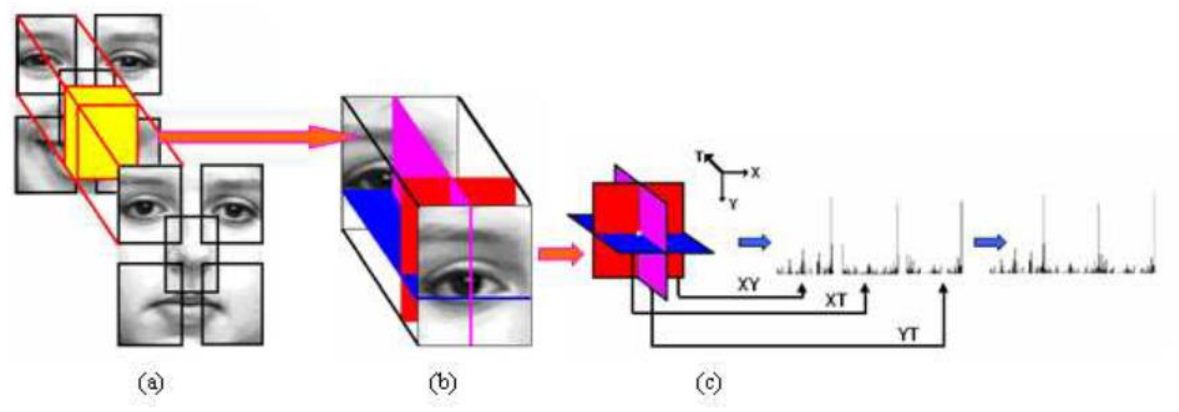


Figure 2.13 - LBP-TOP is being applied across a face, the image is first split into block volumes (a), showing the feature from 3 planes in a single block (b) and their resulting histograms (c) [31]

LBP-TOP has also been used and extended by Mattivi et al. [85] for recognizing human actions such as walking, jogging and boxing. The extension involves using more slices in each of the XY, XT and YT dimensions resulting in 6 slices for each axis. He has called it Extended LBP-TOP. Another modification they have made is to use the LBP operator on gradient images. Claiming that the gradient image contains information about the rapidity of pixel intensity changes along a specific direction. And that it has large magnitude values at edges which can further increase the LBP operator's performances [85]. This method was named as Gradient LBP-TOP.

Almaev et al. and Valstar et al. produced another modification called Local Gabor Binary Patterns from Three Orthogonal Planes (LGBP-TOP) for automatic facial expression recognition [86]. This has been used to detect Action Units based on the Facial action coding system. Their modification introduces Gabor filters that are applied to the subject before LBP-TOP is taken. This has shown to perform better overall than the standard LBP, LBP-TOP and LGBP (Local Gabor Binary Patterns).

2.3 Machine Learning

Machine learning is still an active research area with the goal of creating intelligent machines that can learn from humans. This section will discuss the machine learning techniques used extensively across all areas of computer vision.

2.3.1 Classification Models

This section will briefly describe machine learning techniques that are used for classification tasks.

2.3.1.1 Support Vector Machine

SVM is a robust learning algorithm that is currently the most commonly used machine learning technique by data scientists and industries around the world. It is a supervised learning method that looks for patterns within classes and separates the different classes using hyperplanes. This is achieved by mapping the given features into a high dimensional feature space that can best separate the features from different classes through optimisation of the hyperplane. In the upcoming experiments, a MATLAB toolbox/library called Lib-SVM [87] is used for modelling the SVMs, which specialises in using kernel-based techniques including Polynomial kernel and Radial Basis Function kernel.

LibSVM Setup

This toolbox involves running kernel based SVMs, namely Polynomial and RBF. The hyperparameters are different for each kernel. The RBF kernel $K_{RBF}(u,v)$ can be calculated as shown in Equation 2.9, where g is the gamma hyperparameter in the kernel function, and u, v being the vectors in the input space. g is the only hyperparameter that can be adjusted, which is determined based on the number of feature dimensions *(fd)*. It is calculated as $g = \frac{1}{fd}$, where fd will be based on the reduced number of dimensions after the feature reduction procedure.

$$K_{RBF}(u,v) = e^{-g\|u-v\|^2} \tag{2.9}$$

The Polynomial kernel $K_{Poly}(u,v)$ can be calculated as shown in Equation 2.10. However, it also includes adjustable hyperparameters d for the degree of the kernel, and r being the coefficient in the kernel function. It is hard to try all possible combinations for the upcoming experiments. Therefore, the recommended default values in the toolbox for hyperparameters r and d have been selected. g is the same as the RBF kernel, which is $\frac{1}{fd}$.

$$K_{Poly}(u,v) = \left(gu^{\mathrm{T}}v + r\right)^{d} \tag{2.10}$$

2.3.1.2 *K-Nearest Neighbour*

K-Nearest Neighbour (KNN) is a computationally efficient machine learning technique that looks for a calculated distance between an input sample with a set of training samples. The prediction is based on using a majority voting scheme from the numbers of nearest neighbours (closest distances). The distance used commonly used with KNN is the Euclidean distance d, shown in Equation 2.11, where $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and $\mathbf{y} = (y_1, y_2, \dots, y_n)$ are points in the Euclidean space $\mathbb{R}^n$. A few other distances include Manhattan, Hamming and Minkowski distances.

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.11}$$

2.3.1.3 *Random Forest*

Random Forest is an algorithm that was first developed by Breiman *et al*. [88]. It is based on an ensemble of classifiers that make decision trees, which are made up of individual learners that are combined. N decision trees are created using a training subset, using a greedy procedure [88]. These decision trees are then collectively gathered to produce a prediction based on majority voting, as shown in Equation 2.12, where y_i is the class and $p_n(y_i)$ is the probabilities for each class.

$$R_{(y_i)} = \frac{1}{N}\sum_{n=1}^{N} p_n(y_i) \tag{2.12}$$

2.3.2 Regression Models

This section will briefly describe techniques used to solve regression problems.

2.3.2.1 *Partial Least Squares Regression*

PLS regression attempts to fit multiple response variables in a single model. Because PLS regression models the response variables in a multivariate way, the results can differ significantly from those calculated for the response variables individually. The best practice is to model multiple responses in a single PLS regression model only when they are correlated. The correlation between the feature vector and depression labels is computed in the training set, with the model of PLS as:

$$\begin{aligned}\mathbf{X} &= \mathbf{TP}^{\mathrm{T}} + \mathbf{E}\\ \mathbf{Y} &= \mathbf{UQ}^{\mathrm{T}} + \mathbf{F}\end{aligned} \tag{2.13}$$

where **X** is a $n \times m$ matrix of predictors and **Y** is a $n \times p$ matrix of responses. **T** and **U** are two $n \times l$ matrices that are, projections of **X** (scores, components or the factor matrix) and projections of **Y** (scores); **P**, **Q** are, respectively, $m \times l$ and $p \times l$ orthogonal loading matrices; and matrices **E** and **F** are the error terms, assumed to be independent and identical normal distribution. Decompositions of **X** and **Y** are made to maximize the covariance of **T** and **U**.

2.3.2.2 Linear Regression

Linear regression a fitting technique to model a given set of features with its respective variables using a linear equation. Given a set of variables **X** and the responses/labels **y**, the linear relationship can be defined as:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & \dots & x_1^m \\ \vdots & \ddots & \vdots \\ x_1^m & \dots & x_n^m \end{bmatrix}; \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix};$$

$$\mathbf{y} = \alpha + \beta\mathbf{X} \tag{2.14}$$

Where α is the intercept and β is the slope. These parameters are minimised to best fit the variables.

2.3.3 Deep Learning

In this section, deep learning in the form of Convolutional neural network will be presented and explained in detail.

2.3.3.1 Convolution Neural Networks

CNNs are a form of Neural Networks that recently have been shown to be very effective for solving image-based tasks. The applications can be almost anything image related such as object detection [34], [89] and face recognition [90]. It has provided a revolutionary breakthrough for training deeper networks through supervised learning. The layering idea of the CNN was inspired by the visual cortex of a cat [91], from which LeCun et al. adopted this approach for vision applications such as handwriting recognition [35], [92]. Since then, the idea has evolved into its own active research area in machine learning, becoming a widely used approach in all areas [93]–[95]. In this section, the architecture and building blocks of the CNN are described in detail for a better understanding.

Architecture Layers

Generally, the architecture of a CNN follows a pattern of stacked layers. However, recent works in this area show that have gone beyond stacking and have introduced different ideas [96], [97], which include multiple branches. Generally, CNNs have neurons (parameters as weights) stacked in a 3-dimensional space, measured by height, width and depth. They represent the spatial dimension along with the feature maps of the given input.

Convolution Layer

Convolution layers are used to compute the convolution spatially of a set of input neurons with a learnable filter. Feature maps are produced by doing the dot product of the filter weights with the input neurons, which are then passed to the next layer for further computation. The task of this layer is to produce a response from the input data the best it can by tuning the filters weights to look for generalised features across the data.

The basic parameters for the layer consist of: Kernel Size; Depth; Stride and Padding. Each has their task as described:

• Kernel Size:	Spatial size of the convolution filter.
• Depth:	The 3rd dimension of the filter, that represent the depth of the filters.
• Stride:	The stride parameter represents how much the convolution filter slides across a local region on the given input data.
• Padding:	Determines how much the input should be zero padded and from which edges.

Pooling Layer

The pooling layer works to reduce the spatial dimension of a given input. This can be performed across the whole input using a sliding window on a single depth slice at a time. The pooling operations used in CNNs are to calculate the Average or Max of a spatial location. Equation 2.15 shows the operation for both, where α and β are the dimensions of the spatial window used for the pooling operation, u and v are the spatial location of the input slice k, and $f(u, v, k)$ is used to get the local patch from the previous layer.$\hat{f}_{avg}(k)$ and $\hat{f}_{max}(k)$ represents the pooled pixel at slice k.

$$\begin{gathered}\hat{f}_{avg}(k) = \frac{1}{\alpha \times \beta} \times \sum f(u{:}\, u+\alpha, v{:}\, v+\beta, k) \\ \hat{f}_{max}(k) = \mathrm{MAX}\big(f(u{:}\, u+\alpha, v{:}\, v+\beta, k)\big)\end{gathered} \tag{2.15}$$

Rectified Linear Unit Layer

This layer applies an activation function on neurons to add nonlinearity, generally following a convolution layer. The ReLU layer specifically removes the negative neurons, based on Equation 2.16, by taking the max function of the neuron with a threshold of 0.

$$f(x) = \max(0, x) \tag{2.16}$$

There are other activation functions such as tanh and sigmoid but are not commonly used as they tend to squeeze neurons between 0 and 1. When the output of the previous layer is close to 0, the neuron is killed and when close to 1, the neuron saturates, making them both ineffective. The tanh and sigmoid activation function compared to the max function are slower and more expensive to compute as it involves exponential values. The max function is just simply applying a threshold proving to be far more efficient [34]. The Leaky ReLU is a modification to the ReLU layer where a small negative slope is introduced to prevent all negative neurons from dying (always set to 0). It was developed by He et al. [98] which they have demonstrated how the leaky ReLU can also be parametrised so it can adjust to give its best performance.

Fully Connected Layer

The Fully connected layer is a special form of the convolutional layer, where all the neurons have been reduced to a 1×1 spatial size. The depth of the data connects them all together.

$$y_j = f\left(\sum_{i=1}^{m} z_i \cdot w_{i,j} + b_j\right) \tag{2.17}$$

Equation 2.17 represents the fully connected layer that computes the dot product between each neuron from the input data and the filter weights, where y_j is the output neuron by taking the function $f(z)$ of the given input z_i from the previous layer. The function calculates the sum of all inputs z_i with the dot product of each individual weight j of the fully connected layer plus the bias b_j.

The final fully connected layer size is based on the number of categorical labels for the application it is designed for. If there are 5 classes, then the size will be $1 \times 1 \times 5$ neurons. Each of these neurons will be connected to all the previous layers neurons.

Classification Layer

The Classification Layer is used to measure the performance of the last fully connected layer based on its predicted probabilities. Since applications for Convolutional Neural Networks are highly used for classifying multiple categories, the SoftMax function is the most commonly used for this layer.

$$S_n(z) = \frac{e^{z_n}}{\sum_{i=1}^{N} e^{z_i}} \tag{2.18}$$

Equation 2.18 shows the formulation of the SoftMax function, which takes the output of the CNN as a vector of scores z and squashes them between the values 0 and 1, where the sum totals to 1. The cross-entropy loss is then determined using Equation 2.19.

$$L = -S_{y_i} + \log\left(\sum_{n=1}^{N} e^{S_n}\right) \tag{2.19}$$

2.4 Critical Evaluation

The literature had reviewed techniques that are widely used in many applications, namely FER. The typical structure of an artificial system was detailed, with various feature extraction and machine learning steps. For still images, the local descriptors mentioned in the literature are considered for the upcoming experiments to try and capture changes that occur between different expressions within the local face regions. The LBP descriptor produces a pattern detection feature, good for cataloguing local neighbour patterns. The ULBP variation provides robustness by looking for a set of pre-defined pattern combinations. For facial expressions, ULBP can capture the differences in terms of patterns detected within each expression.

HOG looks to capture the shape of entities within an image, by capturing the different angle of gradients and their strength. This can be useful for capturing the variations of the shape across the facial parts, to determine how they differ across each expression. EOH is a variation of the HOG descriptor that proposes a faster and efficient technique to capture the gradients using Sobel edge detection. Both techniques can capture key attributes of the facial deformation of an expression. LPQ works similarly to LBP but in the frequency domain, making it unique to the others. This technique can be used to investigate whether the combination of descriptors in different domains makes a feature set more robust and reliable.

Another area in which FER can benefit is from integrating deep learning techniques for feature extraction and inferencing. There are existing deep CNNs trained on images such as VGG-Face [90] and AlexNet [34], as well as the opportunity to train a deep CNN from scratch to learn FER images. The FER performances using deep learning and hand-crafted techniques will be compared to investigate whether one is better than the other.

For temporal based feature extractors, LBP-TOP simply extends the LBP feature from an image to an image sequence. This has shown to be effective in applications [31] [99], even though it is based on an idea for static images. However, the basis of MHH has been designed specifically for an image sequence. It presents a good technique to capture motion patterns throughout a sequence. This is a key area to investigate when attempting to model the structural change of the face throughout the expression action. Motion can capture the facial deformations which can provide a reliable description of the expression.

Each technique has been detailed and is considered for the upcoming experiments to obtain their unique set of features. Each of the chapters will have their own related works section, to provide the most appropriate information of other people’s work for comparison.

Chapter 3. Facial Expression Recognition from Still Images

Chapter 2 has reviewed existing feature extraction techniques along with the related works on facial expression recognition. In this chapter, facial expressions from 2D and 3D still images has been studied for human-computer interaction tasks. This is to obtain a better understanding of how machines can best perceive these emotions and expressions in a detailed and efficient way. Various feature extractors are explored to produce a mathematical understanding of facial expressions, which can be modelled with machine learning techniques to understand and predict similar behaviour.

3.1 Introduction

With technology advancing at a fast rate, it is now possible to capture facial expressions in multiple dimensions using many different modalities such as texture and geometric information. There has been significant research done for 2D based facial applications which have shown impressive performance across the spectrum. However, there are still some situations that can cause performance hits purely because the 2D data (static facial images) cannot robustly provide its highest quality representation. Common problems with 2D facial images include facial occlusions, bad lighting, head and pose rotation. This is where 3D facial imaging can eliminate some of these issues and even provide the possibility to bridge the gap by combining both 2D and 3D information in the system. 3D Data can include geometric data (e.g. facial landmarks and 3D facial mesh), time series data, depth information, with more avenues being researched. Computational resources have significantly increased, and still is, making it easier to meet the demands of processing the 3D data.

The focus here is to develop a system to execute facial expression recognition based on 2D and 3D still images. The processes from start to end are explained and experiments are run to focus on evaluating their performances. Various local and global descriptors are captured using different methods, with machine learning techniques trained to classify the descriptors to their appropriate categories. Two databases are selected for the experiments that contain both 3D and 2D facial expression data. These are the Binghamton University 3D Facial Expression and Bosphorus 3D database, both containing 6 basic expressions of Angry, Disgust, Fear, Happy, Sad, Surprise, as well as the Neutral expression. The main experiment will be based on the BU-3DFE database, and the Bosphorus database will be used as a validation to confirm the findings.

3.1.1 Problems & Related Works

Facial expression recognition can be a difficult task to accomplish because of the diversity of each expression across different people. Each person can have their own unique style of portraying these expressions. Capturing people’s emotions using facial expressions can help understand and predict a person’s behaviour. However, the facial images are generally captured based on fixed conditions that made the problem easier to solve, but also unrealistic. These conditions include having the same ideal lighting across the face, which had to be oriented in a frontal profile, and next to no occlusions. Not only this, but the emotions were also acted and sometimes exaggerated to create a clear distinction between each emotion.

Recent challenges include occlusions, varying ethnicities, genders and skin colour. These contribute to making the task more challenging. There has been a lot of research done towards understanding texture images, to extract key features that can determine which expression is portrayed in the image. This is demonstrated by works mentioned in [100], [101], showing a significant progression in using face detection and localisation techniques to improve performance and eliminating some of the challenges.

3D imaging has also been investigated to try solving problems that can occur with 2D imaging. These can be situations where the face is not orientated correctly, the pose varying across samples, or even having issues with the illumination changes. There has been a lot of recent work in this area that demonstrates the performance introducing 3D imaging to FER [62], and how it can eliminate problems that 2D techniques cannot. Considering these benefits, this chapter will focus on combining the efforts of both 2D and 3D techniques to produce a system that is more robust towards these imaging problems.

There are a few 3D FER databases based on static face images, such as BU-3DFE [60], Bosphorus 3D [102], D3DFACS [103], and ND-2006 [104]. The BU-3DFE database is very popular amongst researchers as it provides a consistent set of samples for 6 basic expressions + Neutral, across a variety of ethnicities. They provide both 2D texture image of the frontal face and a 3D scan containing 35,000 vertices. There has been a lot of research applied to this database in the 3D and 2D domain to try and solve the FER task. Early study indicates a good performance using both geometric data [10] [105] [106] [107] [108] [109] and textured data [110] [111]. Recent studies after these experiment show further improvements using tougher protocols [112] [113] [114] [115]. They include a variety of designed feature sets that are based on either 2D texture data, 3D geometric data, or both.

The early 3D techniques started with Wang et al. [10], who have had the initial tests on the database, which involved looking at the Primitive Surface Feature Distribution across the 3D vertices. They create a triangular mesh and estimate a curvature map in which they determined the primitive 3D surface labels based on a set of rules. Linear Discriminant Analysis was adopted for their machine learning approach. Rabiu et al. [105] used the provided 3D facial landmarks to create a set of specific geometric distances and angles, with the idea inspired by FACS. They also apply mRMR to keep the most relevant and non-redundant set of features, paired with a One VS. All SVM approach for machine learning.

Yurtkan et al. [109] looked at facial movements in the 3D space, by generating facial feature point positions. Their idea uses entropy on the changes in face deformation that occurs during an expression change. They use a two-level SVM to classify the 6 basic expressions. Tekguc et al. [108] look at the distances between landmarks to see how they vary throughout the expressions. The 3D Euclidean distance is calculated across all permutations of landmark pairs. Feature selection is applied using a multi-objective genetic algorithm, which is designed to optimise the number of features to use.

Texture-based feature sets were also adopted, with Lemaire et al. [111] creating Differential Mean Curvature Maps to capture the deformations of the 3D mapping. This was based on global and local information. From these texture maps, they applied HOG to capture the orientation gradients within the deformations and used this feature with SVM for classification.

A Recent study also includes using both texture and geometric-based features, with Li et al. [115] capturing the local shape and local texture features and achieving great accuracy. They use modified versions of HOG that include Histogram of Second Order Gradients (HSOG); Histogram of Shape Index (HOS); Histogram of mesh Gradients (meshHOG); Histogram of mesh Shape index (meshHOS); and SIFT features. These are combined using early and late fusion, paired with SVM to produce the state-of-the-art results on the BU-3DFE database using hand-crafted features.

3.1.2 Findings and Research Direction

For this research, we consider how various traditional descriptors from 2D and 3D imaging can complement each other. The chosen 2D techniques are widely used in the computer vision area, to allow various high-quality feature representations of the images. For the 3D features, we will consider techniques that will observe the facial movement of each expression. The 2D and 3D sets of features will be considered for fusion to observe if they can complement each other.

3.2 Framework for 3D Facial Expression Recognition

The framework proposed will comprehensively model various abstract techniques that are applied to sources of different modalities, with the main objective of detecting 6 discrete emotions. The idea is to investigate various hand-crafted techniques applied on the geometric and textured modalities, comparing their individual performances. These features are also fused together at the feature or decision level to determine whether a fusion can provide a performance boost.

The framework can be split into three stages, with the first being the data pre-processing and feature extraction stage. The next stage is the optional feature fusion, followed by dimensionality reduction and normalisation. And finally, there is feature learning stage using various machine learning techniques to try fit the data to its respective class. Figure 3.1 visually demonstrates the data flow and processes of the proposed framework. Each of the features has been extracted based on their respective toolbox recommendation, using the MATLAB environment.

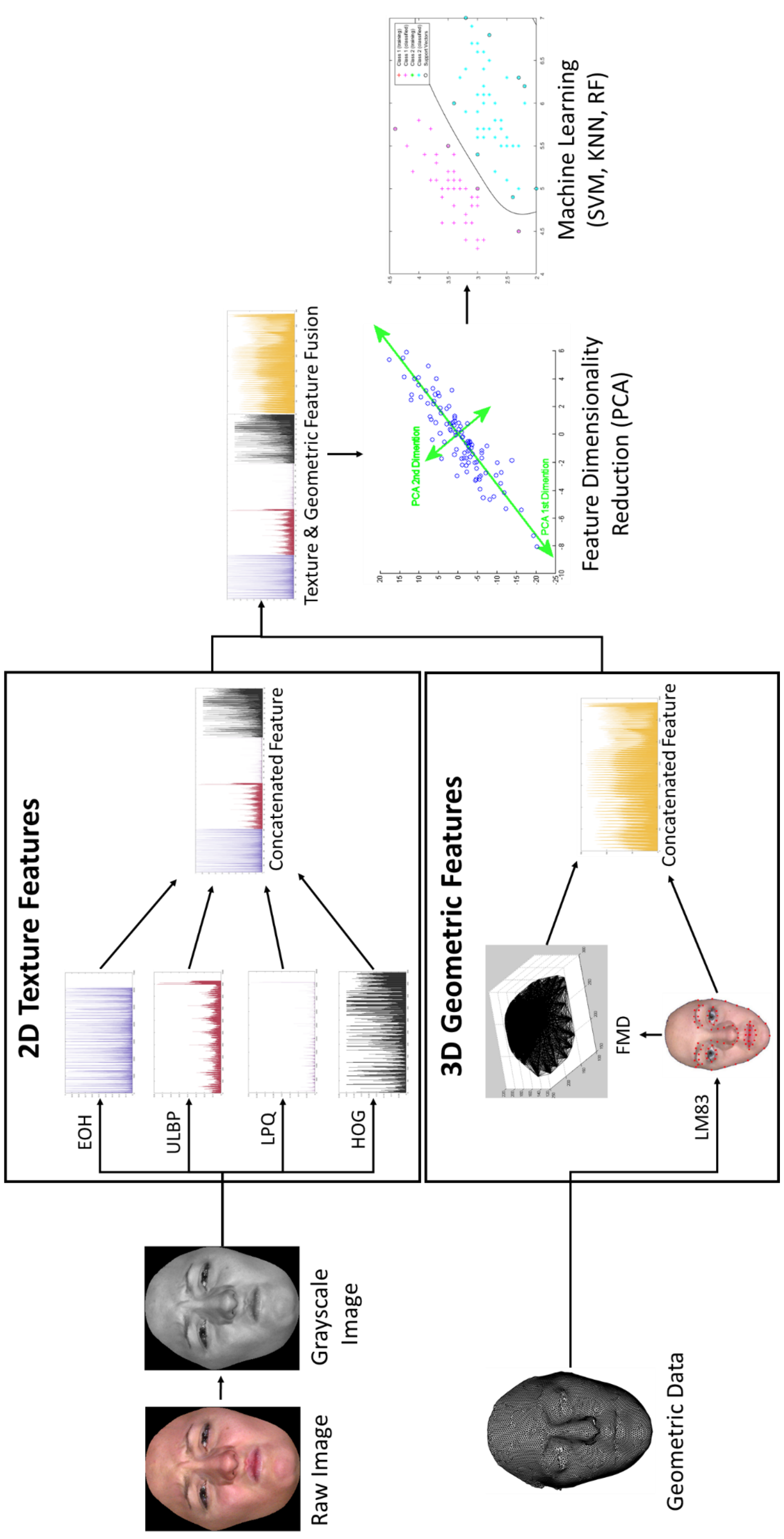


Figure 3.1 - Framework for Fusion based 3D & 2D Facial Expression Recognition using static images. There are two initial branches that start with the texture data and the geometric data. Each of those branches has their respective features extracted, which are fused together using concatenation. Then the two branches are also fused and reduced in dimensionality before being used for machine learning

3.2.1 2D Texture Descriptors

There are many feature descriptors available for facial images to highlight their key features, each with their own benefits and drawbacks. Using descriptors to describe an image is a lot more efficient and accurate to learn and conclude compared to using the raw image. Textured images are a common input source as most applications are based on having visual entities as inputs. These images provide a detailed representation of the entity, in which noise and other information unnecessary to the subject can be found. Hand-crafted texture descriptors can highlight certain characteristics of the image that is best suitable for its application. Therefore, some descriptors perform better than others for certain applications. For the upcoming experiments, a select number of 2D texture descriptors will be extracted for their applications that involve facial images. Before extracting these texture features, each image is also resized to a 128×128 and 64×64 size image. There will then be 3 sets of images to work with, which are the raw images, the resized 128×128 images, and finally the resized 64×64 images. Each hand-crafted descriptor will be applied on each set of images will be experimented to understand what impact the image size makes.

3.2.1.1 ULBP Extraction of Texture Images

The LBP Feature is one of the most popular texture descriptors and is used for the upcoming experiments. LBP is good in looking for local patterns such as edges, corners and lines within an image. For facial images, this would be details such as the ends of the mouth, eyes and the detection of other facial parts. For facial expressions, this technique can further highlight the movement of the facial muscles for each expression. A key factor for determining a certain facial expression is to observe how the facial parts move together, and in which direction.

As there are many variations of this technique, the ULBP descriptor has been chosen as it focuses on capturing specific types of edges and boundaries. With facial expressions, the change that occurs on the face throughout the different expressions is what needs to be captured. Using ULBP can prevent any noisy characteristics being captured from the image, which LBP may capture as it looks for all 255 possible combinations. For the experimentation, each of the images is split up into windows of sizes 8×8, 16×16, 32×32 and 64×64. ULBP is applied on each window, and the resulting histograms from each are concatenated to produce a final ULBP feature vector, to represent the whole face.

An example size of a ULBP feature vector, for say a 128×128 image using window sizes of 16×16, can be determined as follows: $\frac{128\times128}{16\times16} \times 59 = 3{,}776$ components. The largest dimension of all the ULBP variants is the raw image based on 8x8 window sizes. This produces a total of $\frac{320\times256}{8\times8} \times 59 = 75{,}520$ components. This is a very large set of features to handle, which can make the training process very long and tedious. Therefore, having a feature reduction technique is very important for situations like this.

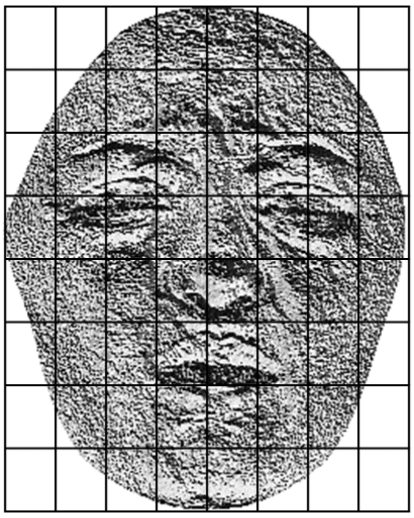


Figure 3.2 - Visualisation of the ULBP operator applied to a Facial image demonstrating an intense Angry expression, where a histogram will be produced on each of the 64 windows to make the ULBP feature

In Figure 3.2, a facial image is split into 64 windows with each window displaying the effect of applying the ULBP operator on it. This is then captured in a histogram to represent the number of occurrences of each pattern.

3.2.1.2 LPQ Extraction of Texture Images

Local Phase Quantisation is another technique like LBP but works in the frequency domain to obtain the Fourier frequency. LPQ is often known to be useful for blurred images [78], [116], which other techniques struggle. LPQ is computationally more expensive to run compared to LBP, as it requires moving to the frequency domain. For facial expressions, LPQ can be used to investigate the differences between expressions that occur in the frequency domain.

When capturing the LPQ feature, each image is broken up into 4×4 windows of equal sizes, each of which LPQ is applied to produce a histogram. Then each histogram from all windows is concatenated together to produce the final feature. This produces a feature vector containing $4 \times 256 = 4{,}096$ components. Throughout the experiments, the radius parameter has been varied to evaluate what is the best radius and how the performance differs across the selected radiuses. The radius values chosen are $r = [2, 4, 8, 16]$.

3.2.1.3 HOG Extraction of Texture Images

Histogram of Oriented Gradients [65] is a feature descriptor that looks at the distribution of the gradients within an image. The histogram is determined by two properties. These are the gradient direction and magnitude. The direction determines which bin to populate, and the magnitude determines how much to populate it by. This is useful for object detection because calculating the gradient around corners and edges of objects produces a large response. For facial expressions, this can determine the mouth position and angle, or the angle around the face and eyebrows. Each expression has a set of characteristics that can be detected with this descriptor.

HOG has a cell size parameter which is configured to be the following sizes: $c = [8, 16, 32, 64]$. The cell size determines how big a window should be to extract the gradients from. A sliding window technique is applied, from which each window produces 31 features containing histogram bins of gradient directions. The 31 features from all windows are later concatenated to produce the HOG feature. The gradient orientations can be observed in Figure 3.3, where the HOG descriptor is applied on a facial image showing the Angry expression. These orientations are further captured in a histogram to produce the final feature vector.

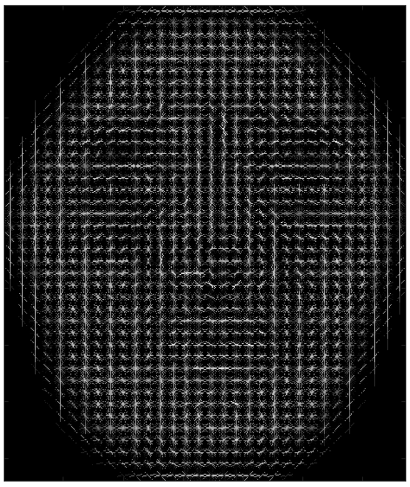


Figure 3.3 - Visualisation of the HOG descriptor applied on a facial image displaying an intense Angry expression

In terms of feature size, this is based on the number of cells and the HOG feature size. For a given image size of 128×128, and a cell size of 16, the total feature size is $\frac{128\times128}{16\times16} \times 31 = 1{,}984$ feature components.

3.2.1.4 EOH Extraction of Texture Images

Edge Oriented Histogram [75] is also based on finding the orientation gradients in an image, but after a Sobel edge detector is applied to the image. It is supposedly a faster and more efficient technique [65]. For facial expressions, EOH can capture information based on edges detailed in the facial structure. This can specifically highlight the facial parts that have moved from their natural position and will be less noise compared to the raw image, as it focuses on just the edges.

In this experiment, EOH is used in a similar style, creating windows on top of the image and then applying the operator to each window. However, the images are only split with 2×2, 4×4 and 8×8 windows as the feature can get very large beyond that point. For a given image of 128×128, and windows of 2×2, the total feature vector size comes to $(2 \times 2) \times 384 = 1536$ components. For the largest window size, the feature size becomes $(8 \times 8) \times 384 = 24{,}576$. The 16×16 window will not be used for this experiment as the feature will become too large and will not be beneficial for the system.

3.2.2 3D Geometric Descriptors

The Geometric descriptors are based on a 3D point cloud data containing $\overrightarrow{XYZ}$ coordinates for each vertex. A useful characteristic of this data is that it can be orientated in any way and keep the same form. This can provide very useful and consistent information for facial expression recognition, especially in events that are not in a controlled environment. In the upcoming experiments, the 83 provided 3D landmarks from the BU3DFE database will be utilised and investigated further.

3.2.2.1 3D Facial Landmarks from Geometric Data

Both the BU-3DFE and Bosphorus 3D database provide 83 and 22 annotated 3D landmarks respectively, of the structure $P_i(x_i, y_i, z_i)$ coordinates. In Figure 3.4, a sample from the BU3DFE dataset can be seen with the 83 facial landmarks annotated onto a cropped face, where the figure is captured in a 2D view.

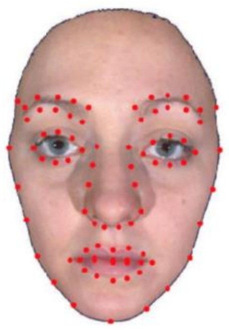


Figure 3.4 - 83 Facial Landmarks annotated on a BU-3DFE sample cropped face

The facial landmarks provide key information to where the expressive facial parts are positioned in the 3D space, and what characteristics can be learnt based on their positions. These include points from the eyebrows, eyes, nose, mouth and around the face. The Bosphorus database also provides similar landmarks, but only 22, so it is not as well annotated as the BU3DFE database which may limit its capabilities.

The landmarks around the face can be an issue because of its nature. They are more likely to represent a personal characteristic of the individual person’s face. This means that the shape of the face can significantly vary between subjects, which in return can be difficult to learn. However, there are some benefits to observing the way the shape changes between expressions. This can be a common characteristic of expressions across all subjects. An example is between the sad and surprise expressions, where the shape stays similar to neutral for sad, yet stretches vertically for the surprise expression.

Before machine learning is applied, the 3D points are pre-processed to be in the best form it can for the machine learning technique. First, each sample (an expression from a subject) is reshaped to become a single vector. Let us denote the feature as LM. For a given sample from BU-3DFE database, each landmark $P_{1:83}$ contains $\mathbb{R}^3$ components. This can be reshaped to a vector which contains $\mathbb{R}^{249}$ components.

Once all samples are reshaped, they are stacked to make the matrix LM. This is then rank normalised between 0 and 1 to provide boundaries to the matrix by having an upper limit of 1 and lower limit 0. This process can be seen in Equation 3.1, where LM_i is the i^{th} component of all samples, α_{min} and α_{max} are the min and max values of LM_i.

$$\widehat{LM}_i = \frac{LM_i - \alpha_{min}}{\alpha_{max} - \alpha_{min}} \tag{3.1}$$

As each landmark is in a local space within a point cloud boundary, it is unlikely that an outlier will be found that can cause the normalisation to deteriorate. $\widehat{LM}$ is the final feature matrix that will be used by the machine learning techniques.

3.2.2.2 3D Facial Mesh Distances from Geometric Data

The facial landmarks can be further exploited to provide greater information than it currently does. A key feature that can be obtained is to capture how the facial parts vary from expression to expression. During each expression, a set of facial muscles tighten or relax which cause them to move from their initial location. These movements are different for each expression, which can be a useful feature to obtain. A way to record these movements is by learning the distance between each of the landmarks.

As each facial part reacts, they will move further or closer to other facial parts. The distance between them can determine how much they have moved and can help understand how some expressions can cause certain movement when compared to others. A metric distance such as Euclidean can provide this information. For example, with the sad expression, the mouth corners drop down on the y-axis based on the Euclidean distance, moving away from the eyes and eyebrows. In the happy expression, the mouth corners are raised on the y-axis and move closer to the eyes and eyebrows, showing an opposite effect on the sad expression. These corresponding distances can help distinguish between the expressions.

$$\delta_{i,j} = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2 + \left(z_j - z_i\right)^2} \tag{3.2}$$

The facial mesh distances feature can be denoted as FMD. The distance is calculated based on the 3D Euclidean distance, as shown in Equation 3.2, where $\delta_{i,j}$ is the distance value between 3D points and

x, y, z are the individual coordinates for each landmark i and j. The idea for this feature is to calculate the distances for all the possible combinations between each of the facial landmarks, similar to [108]. With the BU3DFE database, using the 83 landmarks, there will be a total of 3403 unique distances per sample.

$$\mathbf{d} = \sum_{i}^{83} \sum_{j<i}^{82} \delta_{i,j} \tag{3.3}$$

They are calculated in Equation 3.3, where $\mathbf{d}$ is a vector of all the 3043 distances. These features are also rank normalised to rescale the distances between 0 and 1, providing boundaries and making further processing more efficient.

3.2.3 Feature Dimensionality Reduction

All of the feature vectors extracted from the texture and geometric modality have their own unique characteristics to represent the facial expressions. Some of the feature vectors have high dimensionality due to the descriptor properties. These vectors also contain some highly-correlated feature components that can become redundant within the vector. Using all the feature components can slow down the performance of a system unnecessarily. However, there are feature dimensionality reduction techniques that solve this problem. One of the most popular dimensionality reduction technique used in various research areas is Principal Components Analysis. PCA is used to remove any highly correlated features produced by the descriptors, resulting in a feature set with high variance.

The face has similarities throughout each expression and there will always be features produced by the selected descriptors that are common throughout each expression. Therefore, PCA will be effective in removing the correlated features. Most of the textured descriptors that are used in the experiment produce a large feature vector which mainly consists of noise and highly correlated features. PCA will be set to retain 99% of the variance within them keeping all the contributing features. This applies to both texture and geometric features sets.

3.2.4 Fusion and Feature Learning for 3D Facial Expressions

Fusion is an idea that finds a way to combine the effectiveness of various feature descriptors. This can also be applied during the feature learning process and during the prediction stage, as it is an idea of combining the efforts of multiple entities. For the upcoming experiments, various combinations of feature sets are fused locally by concatenating the components together. With this, we can also evaluate the performance of combining the features from the textured and geometric modalities together, to see if it will produce an improved and robust feature vector. The initial experiment will evaluate the feature vectors individually to compare their performances. Then they will be fused at the feature level by concatenating the various combinations of feature vectors together. PCA will be applied on top of the concatenated feature for dimensionality reduction, to avoid training a significantly large set of fused feature vectors. This feature will then be ranked normalised and ready for the machine learning process.

There are many machine learning techniques available today for all kinds of applications. For these experiments, a few machine learning techniques have been selected based on their popularity and performance. The task for this application is multi-class classification based. SVM has been chosen for its popularity and performance for FER and similar applications. KNN is also chosen for its speed and approach in finding the closest cluster of samples for a given input. For FER, this technique would find the closest matching feature distance of an expression against a dictionary of training expressions. And finally, Random Forest is chosen to give a good comparison and SVM and KNN. All three techniques significantly differ in the way it learns from data, and the upcoming experiments can indicate which performs better under the same controlled settings. These techniques have tuneable hyperparameters that will be optimised throughout the experiment, to obtain the best performance out of the system. For SVM, both the Polynomial and Radial Basis Function kernels will be used in the system. The main hyperparameter for both will be the Cost. KNN has the number of neighbours hyperparameter and RF has the number of trees.

3.3 Experimental Settings and Results

In this section, experiments are carried out on two databases, BU-3DFE and Bosphorus. This is to evaluate the performance of the 2D and 3D feature extractors. The main focus of these experiments is to recognise 6 basic expressions Angry, Disgust, Fear, Happy, Sad and Surprise using various machine learning techniques. The texture and geometric modalities are further evaluated by fusion of their extracted feature vectors. There are 3 experiments that will take place. The first will be the optimisation of the machine learning hyperparameters. The second will run a complete and thorough test of all descriptors on the BU-3DFE database. And finally, the third experiment will run a validation test on the Bosphorus 3D database, to support the findings from the second experiment.

3.3.1 Spatial Datasets

The following section will detail the databases that will be used in the experiments. They are based on texture and geometric data, which involve handling mainly static images or 3D point clouds.

3.3.1.1 BU-3DFE Database

The BU-3DFE database [60] was created by a research group from Binghamton University, who has also developed a temporal equivalent called BU-4DFE and BP-4DFE. The BU-3DFE database contains a total of 2500 samples made up of 100 subjects. For each subject, there are 4 samples for each of the 6 basic expressions that consist of Angry, Disgust, Fear, Happy, Sad and Surprise, along with a single sample of the subject’s neutral face. The 4 samples per expression represent the levels of intensity for that expression, which goes from mild to strong. Each individual sample comes in 4 data formats that are: a cropped front facial 2D texture image; an uncropped side view of the face in a 2D texture image; a 3D geometric point cloud and 83 annotated 3D facial landmarks. The cropped image size is 320×256 pixels, in colour format. The 3D point cloud contains a mesh that has roughly 25K to 35K polygons per face. From the 100 subjects, there are 44 Male and 56 Female that come from a variety of ethnic backgrounds.

3.3.1.2 Bosphorus Database

The Bosphorus database [102] developed by Savran et al. also used for research purposes. The database includes a total of 4,666 scans collected from 105 subjects, of which 61 are male and 44 are female. There are multiple facial expressions included which are represented in 2 ways, first being the basic expressions of Angry, Disgust, Fear, Happy, Neutral, Sad and Surprise. The other are expressions based on Action Units. Ideally, each subject would contain a single frontal face image and 3D landmark file for each expression (except Neutral which contains 4 per subject). This is annotated in Fig. 3. However, the database is not very consistent with some subjects missing certain expressions yet having the others.

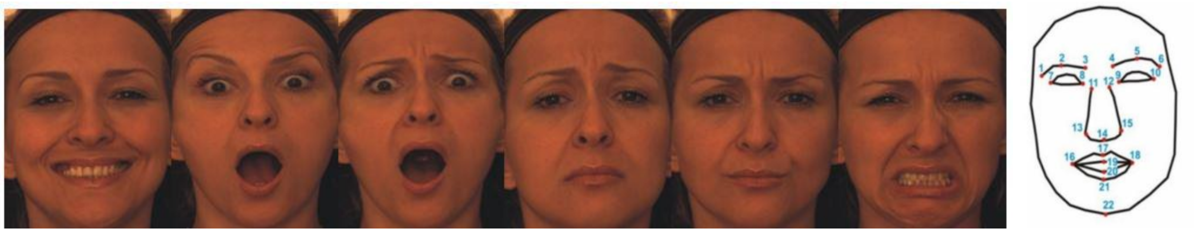


Figure 3.5 - Sample expressions of the Bosphorus database, showing Happiness, Surprise, Fear, Sadness, Angry and Disgust. The rightmost picture shows a 3D annotated landmark sample.

3.3.2 Settings and Protocols

There are 2 protocols that are commonly used for Facial Expression Recognition on the BU-3DFE and Bosphorus 3D datasets. They are both similar to each other, and differ in only one aspect, which is to be subject independent. The protocol is strictly followed as mentioned in [117]. 60 of the 100 subjects are randomly chosen, where 54 are used for training and 6 used for testing. From each subject, the top 2 intensities for each expression are used, making 720 expression samples in total across all subjects. There is a total of 100 tests carried out in a subject independent manner, where no samples from a subject will be found in both the training and testing sets at the same time. In each test, 10-fold cross validation is applied for an extensive and fair result.

Finally, the average result is then taken across all tests to produce a solid recognition rate for each experiment. This protocol is similar to [83] but ensures that images of each subject do not appear in both training and testing data splits, and is now mentioned as Protocol 1. Another protocol that is adopted consists of the same settings as Protocol 1, but it will not be subject independent. This is to achieve a real-world application environment, where each subject can and will most likely reappear several times. This protocol is referred to as Protocol 2. The evaluation metric is based on the recognition rate, measuring the accuracy of the system. This is measured by the number of true class predictions C_T over the total predictions $(C_T + C_N)$ shown in Equation 3.4, where C_N is the false class predictions.

$$RR = \frac{C_T}{C_T + C_N} \tag{3.4}$$

For the Bosphorus database, there are 105 subjects in this dataset with a total of 4666 facial scans. These include Action Unit scans, pose variations and facial expression scans. The upcoming experiment uses just the facial expression scans. These include 1 scan per expression per subject (except Neutral that has 4 samples per subject). However, some of the expressions are missing for subjects, so it is not very consistent. Because of the inconsistency, only Protocol 2 will be applied in the experimental setting for this database. This will be using 60 subjects to predict the 6 basic expressions, along with the Neutral.

3.3.3 Experiment 3A – Hyperparameter Optimisation

This section will go through the procedure and outcome of selecting the best hyperparameter values for the machine learning techniques. All tests from this experiment will be based on the BU-3DFE database because of the data consistency in samples. The feature to be tested is the ULBP texture descriptor and the FMD geometric descriptor, both have PCA applied before machine learning, to keep 99% variance for the texture feature and for the geometric feature. There will be 100 tests using 10-fold cross-validation, taking 60 random subjects from 100 using the Protocol 1 settings.

The initial test will be to select the cost for the SVM RBF and Polynomial (Poly for short) kernels. The hyperparameter that will be optimised is the cost, which will go up in 1, 5, 10, 50, 100, 500, 1000. If X is the given input data, then $\hat{X}$ is the outcome after PCA. The gamma parameter will be set to $\gamma = \frac{1}{\hat{n}}$ for RBF and Polynomial, given that $\hat{n}$ is the number of dimensions of vector $\hat{X}$. The rest will be set to the toolbox default.

Table 3.1 - Cost Hyper-Parameter Optimisation for SVM

Cost	**ULBP 128x128 RBF/Poly**	**FMD RBF/Poly**
1	66.9%/**69.3%**	74.9%/**75.5%**
5	**69.9%**/69.3%	**76.6%**/74.3%
10	69.2%/68.7%	76.5%/73.6%
50	68.1%/68.0%	75.2%/73.2%
100	68.3%/68.1%	75.4%/73.6%
500	68.3%/68.2%	75.4%/75.3%
1000	68.2%/68.3%	75.4%/73.4%

From Table 3.1, the cost for the RBF kernel is best at 5, and for the Polynomial kernel, 1 is best. These will be the choices for the later experiments for the SVM classifier. The next test will be optimising the KNN number of neighbours. The number of neighbours will start from 1 and will reach 49 in steps of 2. An odd number is chosen to prevent equal neighbours of 2 or more different classes. The distance measurement chosen is the default, which is set to '*Euclidean*'.

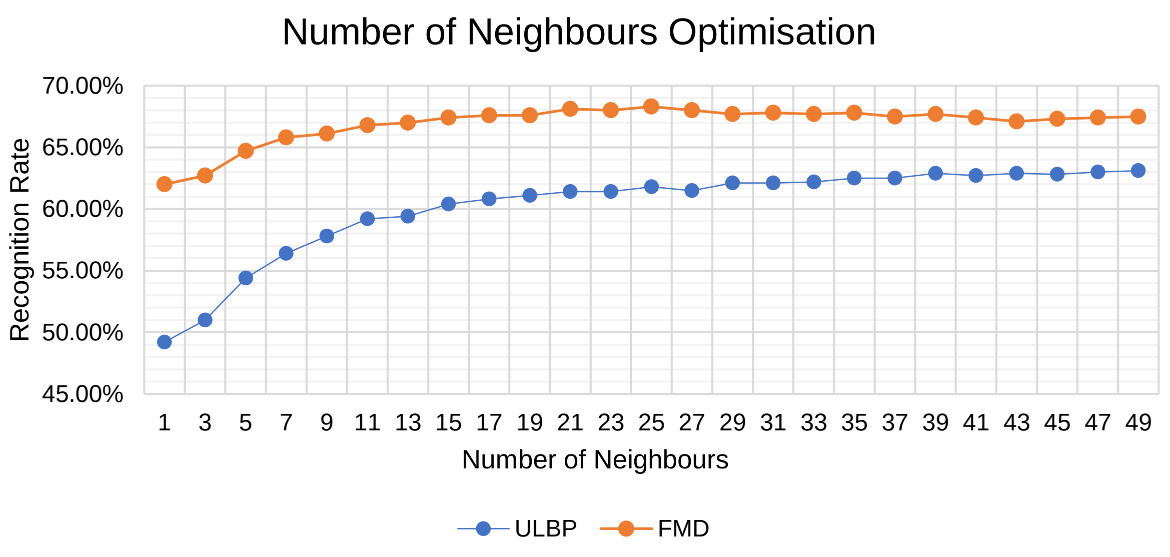


Figure 3.6 - Number of Neighbours optimisation for KNN using the ULBP and FMD feature. Each result is based on 100 tests, and parameter values of 1 to 49 are tested in steps of 2.

Figure 3.6 shows the outcome of optimising the number of neighbours hyperparameter. The chart shows that the number of neighbours beyond 21 generally produce a better performance. ULBP performs best at 49 neighbours whilst FMD is at 25. This indicates that the optimised number of neighbours can be different for each feature.

Random Forest is the final machine learning method that will be used. It also contains a hyperparameter that can be tuned to the data, which is the number of trees to have. The selection of trees is important, as it determines the sample of observations to use. Not having enough trees can mean that some observations of a class may not get selected at all in the training process. For our optimisation, we apply the following values for the number of trees: [10, 25, 50, 100, 150]. Having too many trees significantly increases the computation time and power needed. Therefore, a maximum of 150 trees is selected for optimisation.

Table 3.2 shows the performance of each selection of trees, using the ULBP and FMD features. Having 150 trees does give the best performance of 62.1% for ULBP, and 68.1% for FMD. The results indicate that the higher number of trees do further improve the performance. However, the increase in accuracy gets less and less as more trees are added.

Based on the results from RF, SVM and KNN techniques, SVM has performed the best overall dominating both features using either Poly or RBF kernels. The highest performance from ULBP was 69.9% and FMD was 76.6%, both of which are using SVM with the RBF kernel. KNN has performed ~6-8% worse than SVM using either feature. And RF has ~7-8% margin worse than SVM. Further tests will now include all the texture and geometric descriptors, using the optimised SVM, KNN and RF machine learning techniques.

Table 3.2 - Random Forest optimisation for number of trees to use with the ULBP and FMD features

Trees	ULBP RF 150	FMD RF 150
10	35.9%	54.2%
25	56.7%	66.0%
50	61.3%	67.1%
100	61.8%	67.8%
150	**62.1%**	**68.1%**

3.3.4 Experiment 3B – FER on the BU-3DFE Database

This section will run the main experiment on the BU-3DFE using all feature vectors and their variants, along with the optimised machine learning techniques. Each test will be run 100 times with the average recognition rate taken as the result for that feature vector. Once all the features are tested, a select variant of each descriptor will be tested using further optimisation on the best performing machine learning technique, to give a final defining result for each descriptor.

For this experiment, most of the tests will be based on Protocol 1 as it is a more challenging and tougher protocol to follow. The experiments are broken down into three parts. The first part is to individually test the feature descriptors that are applied to the textured data using each of the optimised machine learning techniques. The second part uses the same process as the first, but on the feature descriptors that are applied to the geometric data. They are also learnt individually using each of the optimised machine learning techniques. Finally, the last part is to apply fusion at the feature level. This is tested thoroughly to determine if fusion makes an impact.

Each texture feature is extracted from the raw image and resized versions. The raw image size is a 320×256 spatial dimension. This is also resized to 128×128 and 64×64, to see if the performance improves or degrades. The benefit of having smaller images are the speedup in computation time for the feature extraction, and the image details also become sharper e.g. long curves can be misinterpreted by ULBP as straight lines in a big image.

When applying the ULBP; LPQ; HOG; and EOH feature descriptors, the image is initially split into windows of $n \times n$, and the descriptor is applied to each window. The final feature is the concatenation of all the features from each window. For the upcoming experiment, we use 4 different window sizes of $n = [8, 16, 32, 64]$ for ULBP and HOG. LPQ and EOH split the image on to a total cell count of $[2, 4, 8, 16]$. Each feature will have their results presented in a table, which will contain results using different window sizes and machine learning methods. Each test is run 100 times, with the same test protocol used in Table 3.1. (Raw, 64,128) represents the result using different image sizes. Raw is the raw image, 64 represents rescaled 64×64 image and 128 is rescaled 128×128 image. These are laid out in the same order for each feature and Machine Learning technique.

The aim of the initial testing is to see how the machine learning techniques perform across the features, along with the variants of each feature descriptor. From this, we can obtain the top 2 machine learning techniques, along with the preferred variant for each texture feature descriptor. Then, a final test will occur to try and achieve the best performance using further optimised machine learning techniques.

3.3.4.1 BU-3DFE Texture Data Performance

Starting with the texture features, the order of testing is ULBP; HOG; EOH; and LPQ. Each table will show all the results from the feature it is representing. The best result from each machine learning technique will be in bold font.

Table 3.3 - 3D FER testing based on the ULBP texture feature. SVM, KNN and RF are applied to each variant of ULBP.

Feature	Image Size	SVM RBF	SVM Poly	KNN 25	KNN 49	RF 150
ULBP_8	Raw	71.4%	68.4%	56.5%	55.7%	58.2%
	128	74.2%	**75.8%**	61.1%	60.6%	**61.5%**
	64	73.9%	73.8%	61.0%	61.6%	60.2%
ULBP_16	Raw	73.2%	74.9%	61.9%	62.0%	58.7%
	128	74.8%	68.7%	**64.2%**	**65.1%**	60.4%
	64	69.7%	74.0%	61.6%	62.7%	55.0%
ULBP_32	Raw	**75.2%**	74.7%	63.0%	64.2%	58.8%
	128	70.4%	68.0%	60.4%	61.4%	55.9%
	64	60.8%	58.1%	52.8%	54.7%	52.0%
ULBP_64	Raw	69.7%	67.5%	59.3%	59.8%	55.2%
	128	59.6%	57.0%	51.3%	51.9%	53.3%
	64	44.1%	42.4%	39.3%	40.2%	42.6%

Table 3.3 contains the results for the ULBP feature. The highest performer is the ULBP_8 feature extracted from the 128×128 size images, using SVM Poly machine learning to get an accuracy of 75.8%. This is closely matched by the ULBP_32 feature extracted from the Raw images using SVM

RBF, with an accuracy of 75.2%. The KNN and RF techniques produced a maximum accuracy of 65.1%, which is significantly less than the SVM techniques.

Table 3.4 is based on the same experiment but using the HOG texture feature. SVM has again dominated in accuracy when compared to KNN and RF. The highest accuracy recorded is 77.1%, using HOG_32 on the Raw images with SVM Poly. SVM RBF with the same feature setup has produced the second highest accuracy of 76.9%. The remaining machine learning techniques perform less than 67%. It is worth noting that the configuration to produce the best performing feature set is HOG_32 on the Raw images, as 4 out of 5 machine learning techniques have this setting yielding their highest accuracy.

Table 3.4 - 3D FER testing based on the HOG texture feature. SVM, KNN and RF are applied to each variant of HOG.

Feature	Image Size	SVM RBF	SVM Poly	KNN 25	KNN 49	RF 150
HOG_8	Raw	66.8%	72.0%	61.7%	63.5%	62.3%
	128	76.1%	76.0%	**64.6%**	64.1%	63.4%
	64	72.4%	72.1%	63.3%	63.0%	58.0%
HOG_16	Raw	76.0%	76.2%	62.2%	65.7%	59.8%
	128	75.1%	74.7%	63.8%	64.0%	60.8%
	64	67.6%	66.7%	60.3%	60.4%	55.6%
HOG_32	Raw	**76.9%**	**77.1%**	62.2%	**66.5%**	**61.3%**
	128	69.8%	69.1%	60.4%	60.6%	58.9%
	64	57.9%	56.7%	55.1%	55.3%	50.4%
HOG_64	Raw	70.8%	70.7%	60.9%	60.9%	59.1%
	128	59.0%	57.0%	55.1%	55.2%	51.3%
	64	41.5%	41.3%	49.2%	49.2%	38.1%

Table 3.5 - 3D FER testing based on the EOH texture feature. SVM, KNN and RF are applied to each variant of EOH.

Feature	Image Size	SVM RBF	SVM Poly	KNN 25	KNN 49	RF 150
EOH_2	Raw	72.8%	71.4%	61.7%	61.2%	57.8%
	128	**73.3%**	**72.5%**	**62.6%**	**63.1%**	**59.7%**
	64	71.7%	71.7%	62.5%	62.9%	58.8%
EOH_4	Raw	71.1%	71.3%	58.8%	58.3%	53.0%
	128	71.2%	71.5%	59.6%	59.2%	52.8%
	64	70.1%	70.5%	58.3%	58.3%	51.6%
EOH_8	Raw	66.6%	68.6%	55.6%	54.9%	50.5%
	128	63.5%	67.4%	53.6%	53.4%	49.4%
	64	62.4%	66.2%	53.1%	53.0%	48.1%

Table 3.5 shows the results when using the EOH feature. The best performer is EOH_2 on 128x128 images, using SVM RBF. The accuracy is 73.3%, with SVM Poly on the same feature producing 72.5% accuracy. The results indicate that the best performing feature variant for all of the machine learning technique is the same, which is EOH_2 on 128×128 images. Table 3.6 is based on the LPQ feature. It has achieved the best accuracy of 74.7% using LPQ_8 on the Raw images and using SVM RBF. LPQ_8 on the Raw images produce the best accuracy for each machine learning technique.

Table 3.6 - 3D FER testing based on the LPQ texture feature. SVM, KNN and RF are applied to each variant of LPQ.

Feature	Image Size	SVM RBF	SVM Poly	KNN 25	KNN 49	RF 150
LPQ_2	Raw	70.9%	70.5%	58.4%	59.1%	52.2%
	128	72.9%	73.2%	58.1%	58.2%	54.1%
	64	71.6%	71.5%	55.0%	56.5%	53.7%
LPQ_4	Raw	74.1%	73.9%	62.4%	62.5%	55.7%
	128	73.2%	73.1%	61.3%	62.2%	55.8%
	64	70.2%	69.8%	56.8%	57.0%	51.9%
LPQ_8	Raw	**74.7%**	**74.2%**	**62.9%**	**63.6%**	**56.0%**
	128	71.0%	70.3%	58.0%	57.4%	52.8%
	64	66.8%	65.9%	56.25	55.9%	49.0%
LPQ_16	Raw	72.9%	72.2%	59.5%	59.8%	55.3%
	128	66.9%	66.1%	55.8%	56.0%	49.5%
	64	55.1%	54.3%	46.7%	47.1%	41.0%

From all the tests above, it is very clear that the SVM techniques outperform both KNN and RF by a good margin. SVM RBF and SVM Poly both perform very similar, making either of them a worthwhile selection. For ULBP, using the ULBP_8 variant on 128×128 images has produced the best result. HOG_32 on the Raw images is the best variant for the HOG feature. EOH_2 on 128×128 images is best for the EOH feature and finally, LPQ_8 on the Raw images is the best for LPQ. 64×64 images have proven to be inefficient, with the idea that the faces become too small and lose their detailed information. To achieve a better performance from each texture feature, a test will occur that considers using the best variant of each feature, along with the SVM techniques. The several Cost hyperparameter values will also be used to see if this can improve the result for each further.

Figure 3.7 demonstrates the performance that can be achieved by the best-selected texture feature variants with further optimised SVM technique. The best feature in the texture domain is HOG_32, which is applied to the Raw image. This feature produced 77.3% accuracy. All features produce a result within a ~3% range.

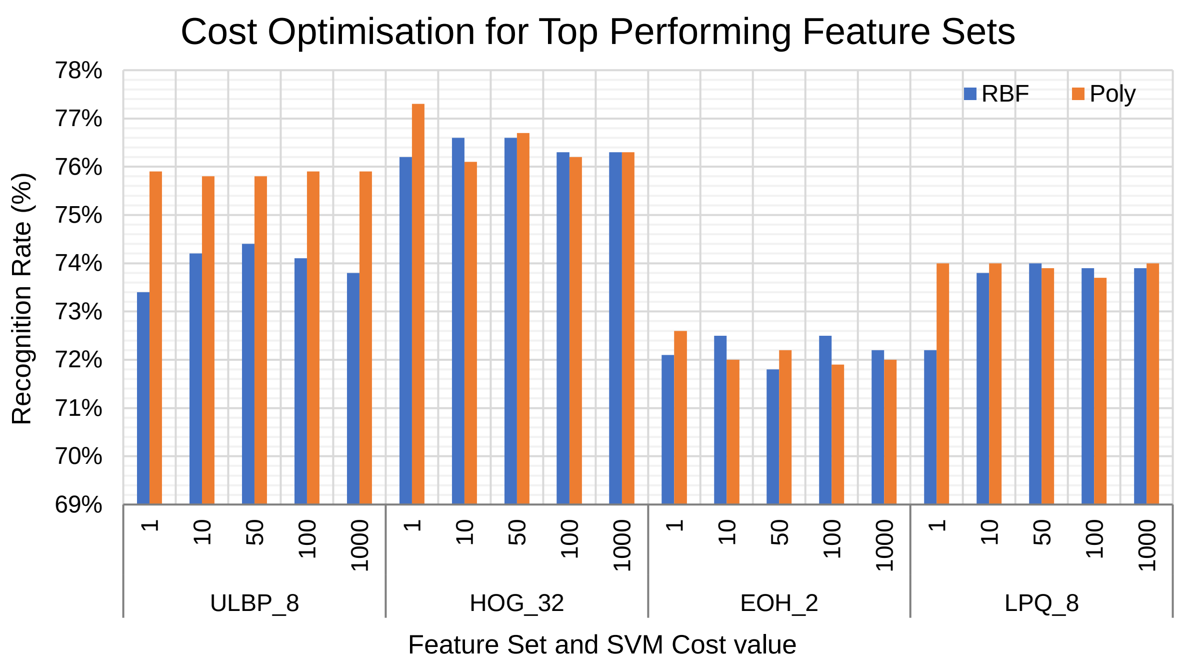


Figure 3.7 - Performance of each optimised texture feature using SVM that is further optimised.

3.3.4.2 BU-3DFE Geometric Data Performance

For the Geometric testing, as there are only 2 features (FMD and LM83), the testing using each machine learning technique will be thorough. These features will not be resized like the images for the texture descriptors, as it will not make any difference. This means that several values of the cost hyperparameter for the SVM techniques will be tested. The number of neighbour's selection for KNN, and the number of trees optimisation for the RF technique are also thoroughly tested.

The LM83 feature vector contains very few feature components (only 249). Therefore, there is no need to apply PCA to this, and the vector will be used as it is. The FMD descriptor contains 3403 components, and PCA will be applied to keep 99% variance. Both features will still be rank normalised before the machine learning technique is applied. For the SVM RBF and Poly tests, the Cost hyperparameter will vary as $[1, 10, 50, 100]$. The number of neighbour's selection will vary as $[9, 25, 37, 49]$. And finally, for RF, the number of trees will vary as $[10, 25, 50, 100, 150]$.

Table 3.7 contains the results of the geometric feature testing. The best machine learning technique is once again SVM, with the Poly kernel just edging over the RBF kernel. The FMD feature shows a higher accuracy than any other texture feature used in the experiments, reaching 77.5% when combined with SVM Poly. This indicates that using the geometric modality can be advantageous and just as useful as the texture information. LM83 seems to struggle with the KNN and RF machine learning techniques, producing less than 50% accuracy in all tests using them. However, with SVM, LM83 comes in as the 3rd best feature out of the texture and geometric features when the Cost $\alpha = 1000$. SVM Poly has produced 1% better than SVM RBF, with SVM Poly now coming in the highest against all individual features.

Table 3.7 - Optimised 3D FER testing based on geometric features FMD and LM83 using Protocol 1.

Feature	**SVM α**	**SVM RBF**	**SVM Poly**	**KNN α**	**KNN**	**RF α**	**RF**
FMD	1	74.1%	76.4%	9	66.1%	10	45.4%
	10	**77.1%**	**77.5%**	25	68.0%	25	54.8%
	50	76.6%	76.3%	37	67.4%	50	57.9%
	100	76.6%	76.2%	49	67.2%	100	60.0%
	1000	76.7%	76.2%	65	67.2%	150	60.8%
LM83	1	20.7%	30.1%	9	24.1%	10	34.1%
	10	46.6%	59.1%	25	23.0%	25	38.7%
	50	65.4%	69.5%	37	22.5%	50	40.7%
	100	69.4%	72.2%	49	22.4%	100	42.3%
	1000	**74.9%**	**75.9%**	65	22.3%	150	42.4%

3.3.4.3 BU-3DFE Feature Fusion Performance

For the fusion, the first part looks at the feature level. In this, the texture descriptors are fused together first, to evaluate the modality performance. This is followed by the geometric features, and then they are both fused together. Each of the tests that will take place will have the same protocol used in the previous tests so that they can be compared fairly. Only SVM will be used for the machine learning stage, using both kernels of RBF and Poly.

Firstly, the combination of the texture features is applied by concatenating the various features together. These combinations include the groupings of ULBP with LPQ; EOH with HOG; and finally, all of them together. The feature vector will become very large when they are combined. Therefore, PCA is applied on top of the concatenated feature to reduce the dimensionality whilst retaining 99% variance within the data. The results presented in Figure 3.8 indicates the performance based on fusing various sets of features from the texture and geometric domains. The combination of ULBP and LPQ have shown an increase for either feature when using SVM Poly as the machine learning technique. EOH and HOG shows a very slight increase (0.2%) over the HOG feature and a significant increase of 4.9% increase over the EOH feature. With all texture features combined, the performance increases to a highest of 78.7% using SVM Poly. This shows a positive sign for feature fusion.

The geometric features FMD and LM83 are also fused together and shows a decrease in performance than having FMD alone. This may be due to the unstable results of the LM83 individual feature, which looks like it is having a negative influence. Finally, the texture and geometric domain features are fused together, showing a further improvement over the previous highest (all texture features fused) by 0.23%.

Not a great margin, however, when the LM83 feature is not included, the performance goes up an additional 0.3%.

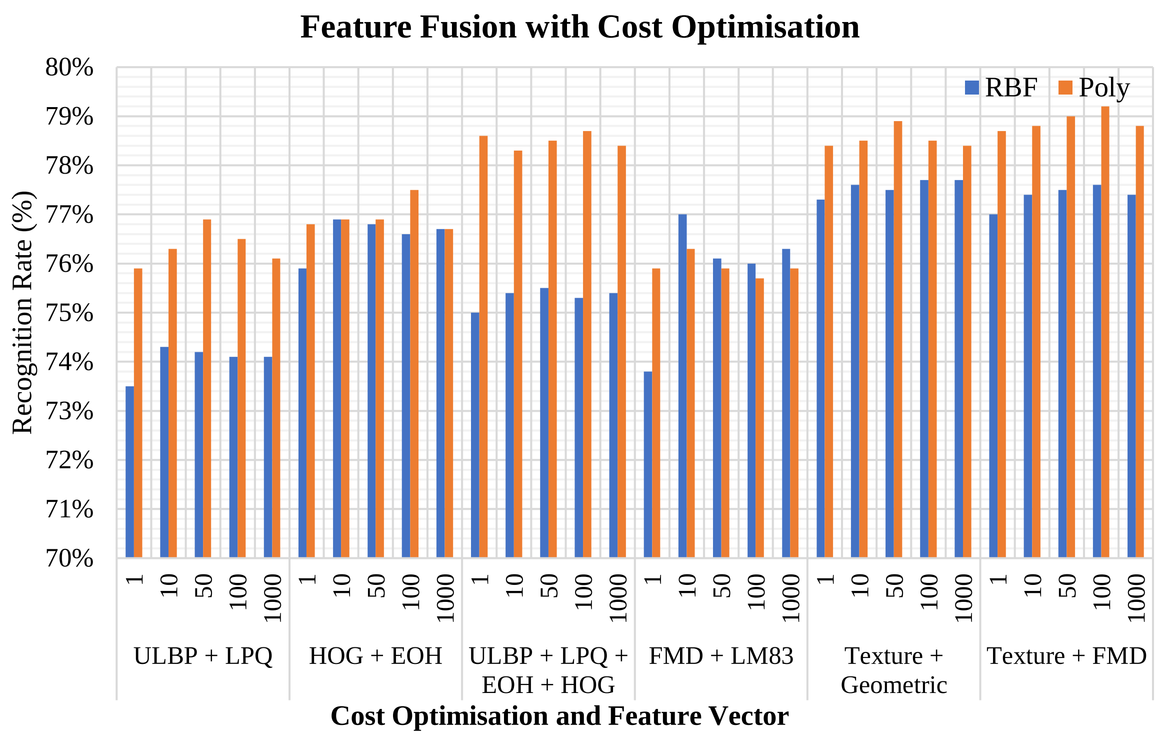


Figure 3.8 - Experiment results based on the fusion of features using Protocol 1, taken from both texture and geometric domains. Accuracy is measured based on the SVM technique using RBF and Poly kernels, with a varied Cost parameter.

3.3.4.4 Experiment 3B Highlights

This experiment has been an attempt to investigate how available texture-based descriptors and geometric descriptors can be used to interpret the human face to determine their expression. The descriptors that were captured from the textured image data are ULBP; EOH; HOG; and LPQ. For the 3D data, the facial landmarks annotations (LM83 and LM22) are taken. Based on these landmarks, a set of distances (FMD) are recorded in the 3D space, calculating the 3D Euclidean distance between each of the landmarks. Initially, a set of machine learning techniques had been adopted in the system. These included SVM with Poly and RBF kernels, KNN and RF.

Table 3.8 is a summary of findings from the experiment, showing the best performances from individual and fused feature sets. The first clear indication from this table is that the SVM Poly kernel has outperformed all other machine learning techniques, including the SVM RBF kernel. Secondly, in most cases, combining the various feature sets together increases the accuracy of the system. From the Lowest individual accuracy of 72.6% to the highest fused feature accuracy of 79.3%. HOG has shown to be the best individual texture feature, and FMD the best geometric. Between them, there is only a 0.2% difference, indicating that both domains can provide high-quality features of similar calibre. Feature fusion has shown a positive sign in most cases, except for when the LM83 feature is included, as it shows a negative impact on the accuracy.

Table 3.8 – A summary table showing the best performance for each individual feature and the fusion combinations tested in the experiment. Each test is based on SVM RBF and SVM Poly machine learning techniques. Those highlighted in bold indicate the best result out of the SVM kernels.

Feature	Domain	Protocol 1	Protocol 2
ULBP	Texture	75.9%	83.5%
HOG	**Texture**	**77.3%**	**83.2%**
EOH	Texture	72.6%	75.7%
LPQ	Texture	74.0%	82.0%
FMD	**Geometric**	**77.5%**	**81.1%**
LM83	Geometric	75.9%	83.3%
ULBP + LPQ	Texture	76.9%	-
HOG + EOH	Texture	77.5%	-
ULBP + LPQ + EOH + HOG	**Texture**	**78.7%**	**86.5%**
FMD + LM83	Geometric	76.3%	81.2%
Geometric + Texture	Both	78.7%	90.0%
Texture + FMD	**Both**	**79.3%**	**90.2%**

Table 3.9 is a confusion matrix based on 100 tests of the fused texture + FMD feature, which indicate some interesting findings. There seems to be a lot of confusion between the Sad and Angry expressions, with Angry predicted as Sad 18.78% of the time, and Sad predicted as Angry 13.15% of the time. Both expressions do have similar characteristics, with them being negative expressions, and the ends of the mouth going downwards. The best distinguishable expression is Surprise, with a 92.46% accuracy, followed by Happy with 90.15% accuracy. Happy and Surprise also have similar characteristics but show signs of clear differences. Surprise causes the mouth to close in and create a round shape. Happy creates a smile which stretches the mouth instead.

Table 3.9 - Confusion matrix of the Texture + FMD test, detailing accuracy between each expression

		Predicted Label					
Actual Labels	**Accuracy (%)**	**Angry**	**Disgust**	**Fear**	**Happy**	**Sad**	**Surprise**
	Angry	**72.30%**	8.16%	5.18%	0.98%	13.15%	0.21%
	Disgust	7.70%	**80.16%**	4.30%	1.67%	3.26%	2.88%
	Fear	5.00%	8.42%	**70.15%**	6.64%	6.35%	3.43%
	Happy	0.04%	2.04%	7.29%	**90.15%**	0.00%	0.47%
	Sad	18.78%	2.65%	8.01%	0.00%	**70.34%**	0.20%
	Surprise	0.02%	2.10%	4.57%	0.80%	0.02%	**92.46%**

Table 3.10 highlights the comparison of our work with performances by others using Protocols 1 & 2. Our result for Protocol 2 indicates a very strong performance against others, above 90%. The highest

recorded performance is 92.2% using a set of defined distances and angles produced using the provided 3D landmarks. For Protocol 1, the best performance was produced by Li et al. [115] using a fusion of 3D and 2D descriptors. They used the SIFT + HSOG 2D feature sets and meshHOG + meshHOS 3D features along with late fusion to produce 86.3%. Our performance was not as strong as Protocol 2 but shows improvement over the works that were done before this experimentation. Li et al. [115] have also used both texture and geometric domains, but in a different manner to ours. They first apply techniques on the geometric data, and then treat that as an image for the texture descriptors. Whereas we apply unique descriptors separately on the data and fuse their resulting feature vectors. From the findings, it is clear that there is a benefit towards incorporating both modalities into a single system.

Table 3.10 - Comparison of our fusion performance with other state-of-the-art techniques

Performance Comparison			
Feature	**Domain**	**Protocol 1**	**Protocol 2**
Wang et al. [10]	Geometric	61.7%	83.6%
Rabiu et al. [105]	Geometric	-	**92.2%**
Soyel et al. [106]	Geometric	67.5%	91.3%
Li et al. [107]	Geometric	-	90.2%
Tekguc et al. [108]	Geometric	-	88.1%
Yun et al. [110]	Texture	-	85.3%
Lemaire et al. [111]	Texture	75.7%	78.1%
Yurtkan et al. [109]	Geometric	-	88.2%
Tang et al.[1] [112]	Geometric	75.7%	87.1%
Zhen et al.[1] [113]	Geometric	84.5%	-
Yang et al.[1] [114]	Geometric	84.8%	-
Li et al.[1] [115]	**Both**	**86.3%**	-
Ours [117]	**Both**	**79.2%**	**90.2%**

3.3.5 Experiment 3C – FER on the Bosphorus 3D Database

Experiment 3C is a validation test, to ensure that what occurs with the BU-3DFE is similar if applied to another database, which in this case is the Bosphorus database. We have used a similar protocol as the BU-3DFE database, as mentioned in [117], but due to the lack of consistency between samples for each expression, they cannot be followed completely. The textured features produced for this experiment are ULBP; LPQ; and EOH. These are extracted from the frontal face image. The landmark annotations are provided by the database but only consist of 22 to 25 landmarks, compared to 83 of the BU-3DFE database. From this set, there are 22 landmarks that are consistent with all samples that are used in the experiment. These are denoted as LM22. The FMD feature is created from these 22 landmarks providing a 231 Facial Mesh Distances. These features are reduced in dimensionality using PCA and classified with SVM using the same parameters as the previous experiment.

[1] The referenced experiments are run after in 2015, after our experiments had taken place

The tests in the experiment consist of a shortened but similar style as the previous experiment. First, each feature individually tested from both domains. Secondly, the fusion of features is applied, starting with using the textured features separately, then the geometric features separately, and finally the fusion of both. As the subjects contain inconsistent samples for each expression, we have decided to combine all the relevant expressions from every subject and to randomly shuffle them around throughout the 100 tests per experiment. Therefore, the tests will not be subject independent. All tests will use the SVM machine learning technique, with the polynomial kernel.

3.3.5.1 Experiment 3C Highlights

The purpose of this experiment is to demonstrate and validate facial expression recognition using texture-based spatial descriptors and 3D geometric features. In addition, the effect of fusing descriptive features taken from spatial and 3D data. Protocol 2 was adopted from Experiment 3B, but to classify the 6 basic expressions and the Neutral face.

Table 3.11 - Performance of each individual and fused texture and geometric domain features from the experiment. All tests are an average of 100 tests, using the SVM Poly kernel.

Feature	Domain	SVM Poly
ULBP	Texture	72.3%
LPQ	**Texture**	**73.0%**
EOH	Texture	71.7%
LM22	Geometric	72.1%
FMD	**Geometric**	**73.3%**
ULBP + LPQ + EOH	Texture	75.6%
LM22 + FMD	Geometric	74.4%
Texture + Geometric	**Both**	**79.4%**

Table 3.11 shows the performances of all the tests taken place in this experiment. The initial 5 tests indicate the individual performances. From these 5 tests, the FMD feature has the highest accuracy of 73.3%, with LPQ (73.0%) being the highest for the texture domain. EOH has performed the worst, with a difference of 1.2% compared to LPQ. When the features are fused, in all cases, the performance has increased compared to the individual tests. Fusing the texture features sets has shown an increase of 2.6% over the LPQ feature, and the fusion of geometric features has shown an increase of 1.3% over the FMD feature. When both domains are fused, there is a significant increase in performance reaching 79.4% accuracy, a 3.7% increase over the fused texture features, and a 6.1% increase over the best individual feature.

3.4 Evaluation and Discussion

The chapter investigated different avenues in how 2D and 3D data can be exploited for FER. The descriptors that were captured from the textured image data are ULBP, EOH, HOG, and LPQ. For the 3D data, the facial landmarks annotations (LM83 and LM22) are taken. Based on these landmarks, a set of distances (FMD) are recorded in the 3D space, based on the 3D Euclidean distance between each of the landmarks. A set of machine learning techniques had been adopted in the system to learn the features. These included SVM with Poly and RBF kernels, KNN and RF.

The first experiment 3A shows how the geometric and texture descriptors FMD and ULBP perform alongside various machine learning techniques. The goal was to optimise the techniques hyperparameters. Each descriptor went through the same pre-processing stages as each other, with the deciding hyperparameter values for each technique based on the highest accuracy. Another optimisation from experiment 3B was based on the facial image size and what effect it can have on the descriptor performance. The image size can play a crucial role in the system, that can improve or deteriorate the performance of it. From the findings, the Raw image size of 320x256 seemed to perform the best using majority of the texture features. The 64x64 images achieved accuracies far worse the other image sizes. From this, it can be understood that having a bigger image size can provide more information for the descriptors, and therefore, create a more detailed feature. The downside to this is that the processing overhead and computation time is increased.

One of the major findings from these tests indicates that SVM has been the most dominant technique. This is evidenced by the machine learning optimisation process in experiment 3A, and the FER tests in experiment 3B. Both Poly and RBF kernels performed roughly as good as each other, with Poly marginally performing better. Experiment 3C was solely based on the SVM Poly machine learning, as this was the best performer from experiment 3B. From the experiments on fusing various feature descriptors, it shows a great demonstration that these descriptors can perform better together. Fusion was adopted at the feature level, where the descriptors are concatenated together to produce a higher quality feature vector for each sample. The idea from this is to combine the characteristics from each descriptor, to produce a super feature descriptor that can pool the benefits from each of them to improve the robustness of the system. The experiments 3B and 3C tried fusion at various levels. First being within domain level feature sets, and second of fusing multi-domain level feature sets. The results for fusing domain level feature sets shows an improvement over any individual feature set. This shows the capability of characteristics complementing each other for a better solution. However, the increase in accuracy was not a large amount for this database (1.35% for texture domain), which can then be debatable whether it is worth the extra computation required. Though, the accuracy does increase further when the geometric feature FMD and the texture features are fused together, to reach 79.25%. There is also an indication that some feature sets can have a negative influence on the fused features, such as the LM83 feature.

The validation experiment on the Bosphorus 3D database also indicates how fusing features from the same or different domains can improve the performance of a system. The outcome showed a progressive increase as more feature sets were fused together, and a significant increase when features from the texture and geometric domain were fused together. Though the Bosphorus 3D database lacked consistent structure and images per expression, there was still a clear indication from the findings that follow a similar trend to the conclusions from the BU-3DFE database. However, the performance increase compared to the BU-3DFE was a lot higher, making it worthwhile to extract various descriptors and fuse them together. In terms of the textured data VS. geometric data, when each domain level features are fused together, the textured features edge over the geometric features. But the individual performance of the geometric features is higher than any of the texture features. Both findings are by a small margin, and the texture domain does contain two additional feature sets. An extension on this could be to look at how the face can be better interpreted before applying the various descriptors. As in most images, there can be a lot of unwanted noise. Examples can be of objects in the background of an image, or for facial expressions, as similarities of the face that occur within all expressions. These similarities across the expressions do not increase the variance throughout the samples. This can only make it more challenging to learn as the picture is not as clear as it could be. In Chapter 4, this idea is explored to find a solution to remove the noise where possible before the feature descriptors are applied.

3.5 Summary

In this chapter, the classification of facial expressions has been practised using the textured and geometric modalities as the sources of information. From this, we looked for diverse ways to understand the data, using various feature descriptors. These descriptors give a mathematical representation of the characteristics of each facial expression, whether it be from the facial image or their face mapping in a 3D space. The BU-3DFE database had been the main test bench for the experiments, with the Bosphorus 3D database acting as a validation to the findings from BU-3DFE tests. Based on the outcomes of all the experiments, the following can be understood:

- The fusion of unique hand-crafted descriptors improves the system performance for FER, in both Texture and Geometric domains.
- Combining both domains also provides a boost to the performance, which was demonstrated on 2 databases.
- In terms of comparing geometric and texture features, geometric features provided a better performance, showing better robustness for FER.
- The SVM technique has achieved the highest accuracy in all of the experiments when compared to KNN and RF.

Chapter 4. Deeply Understanding Expressions via Facial Parts Isolation

In the previous chapter, existing feature extraction techniques had been applied on 2D and 3D facial expression data based on the facial structure. This chapter will introduce deep learning as a feature extractor and learning technique for FER. The face will also be broken down into facial parts to perceive a better understanding of how the face changes throughout the expressions.

Deep learning has made a major impact in all sorts of applications since it had taken off in the ImageNet competition [34], with a huge improvement over other techniques submitted showing a possible breakthrough in artificial intelligence. A deep learning approach, namely Convolutional Neural Networks (CNNs), are applied and manipulated for Facial Expression Recognition. Furthermore, a processing stage is developed on the input facial image to break down and capture the key parts of the face that can better represent an expression. The experiments will include the comparison to traditional hand-crafted techniques used in Chapter 3.

4.1 Introduction

Earlier, in Chapter 3, 3D Facial Expression information was investigated using the texture and geometric domains of the face. It was understood that there can be more done to improve the performance based on the findings from the experiments. What we can understand is that each facial expression has its own combination of facial muscles that move to produce the expression. This can be interpreted using FACS [53]. However, if the whole face is closely observed, there are in-fact parts of the face that are stationary across all the expressions. This could potentially give little to no meaningful contribution to humans and/or machines for distinguishing between the expressions.

Visually, the most moving parts of the face are the eyes; mouth; eyebrows; and nose. This is concluded by observing the facial parts mentioned in the FACS Table 4.1. Thus, it can be assumed that these facial parts can provide valuable information about facial expressions. The rest of the face is assumed providing a little contribution, or even be perceived as noise. Based on this idea, we consider extracting only the mentioned facial parts from the face. Advanced face processing techniques can be adopted to accurately extract facial parts, using generated geometric information to process the textured data. Hand-crafted techniques and deep learning can be applied to the facial parts to understand the level of discrimination each part contributes towards FER, and how well they complement each other.

During the experiment section, facial parts will initially be assessed as separate entities to try and understand what role each can play. It could possibly be that some facial parts are better at distinguishing certain expressions than others. This is followed by combining the efforts of each facial part, to see what combination performs the best for each expression. All of this will be compared to the performance of the whole face as a sample, with a thorough analysis and discussion to follow the experiments.

Table 4.1 - Facial action coding system table consisting the details of all the action units that represent a movement of facial muscles that occur on the face [53]

Facial Action Code	**FACS Name**	**Facial Action Code**	**FACS Name**
AU0	Neutral face	AU24	Lip Pressor
AU1	Inner Brow Raiser	AU25	Lips Part
AU2	Outer Brow Raiser	AU26	Jaw Drop
AU4	Brow Lowerer	AU27	Mouth Stretch
AU5	Upper Lid Raiser	AU28	Lip Suck
AU6	Cheek Raiser	AU29	Jaw Thrust
AU7	Lid Tightener	AU30	Jaw Sideways
AU8	Lips Toward Each Other	AU31	Jaw Clencher
AU9	Nose Wrinkler	AU32	[Lip] Bite
AU10	Upper Lip Raiser	AU33	[Cheek] Blow
AU11	Nasolabial Deepener	AU34	[Cheek] Puff
AU12	Lip Corner Puller	AU35	[Cheek] Suck
AU13	Sharp Lip Puller	AU36	[Tongue] Bulge
AU14	Dimpler	AU37	Lip Wipe
AU15	Lip Corner Depressor	AU38	Nostril Dilator
AU16	Lower Lip Depressor	AU39	Nostril Compressor
AU17	Chin Raiser	AU41	Glabella Lowerer
AU18	Lip Pucker	AU42	Inner Eyebrow Lowerer
AU19	Tongue Show	AU43	Eyes Closed
AU20	Lip Stretcher	AU44	Eyebrow Gatherer
AU21	Neck Tightener	AU45	Blink
AU22	Lip Funneler	AU46	Wink
AU23	Lip Tightener		

Deep learning is also thoroughly investigated in this chapter, to see how it can be applied to facial recognition. This will be applied on facial parts that are extracted using the proposed extraction process, as well as directly on the whole face for comparison purposes. Different deep network architectures will be designed with experimental architectures that combine branches for each facial part to learn them all simultaneously. The deep learning approach will also extend as a feature extractor, from which the probability of expression similarity will be observed, and a metric distance computed to suggest the popular expressions.

4.2 Deep Learning & Facial Localisation for FER

A lot of systems for face recognition, facial expression recognition and similar applications apply common pre-processing techniques that are well known to improve the robustness and stability of their system. Deep learning is becoming an emerging solution in which different forms on networks are utilised for certain tasks or sub-tasks to aid FER. A fast representation of Deep Belief Networks (DBNs) was developed by Hinton et al. [118], using a greedy algorithm to quickly learn a generative model one layer at a time. This idea has been utilised for emotion detection and recognition applications in [38], [119], [120] to try and learn and understand the face. Lv et al. [38] proposed a framework that has multiple learning stages to try and understand the face in detail. They use a DBN as a hierarchical detector that starts by detecting the face using the HOG feature. This is achieved by using HOG on different patches of the face using a sliding window technique. The patches that contain the detected face are followed up with the same process to try and detect parts of the face and continues until smaller individual parts are detected. Gabor features are then captured from the eyes and mouth and are fed to a stacked auto-encoder for classification.

Liu et al. [119] proposed a Boosted Deep Belief Network (BDBN) for FER that contains a 3 stage loopy framework. They apply supervised and unsupervised learning techniques to iteratively learn patches of the face, followed by feature selection and then classifying the 6 basic facial expressions using AdaBoost. They ran experiments on the CK+ and JAFFE database with 80 BDBNs trained from which each is specific to a patch of the face.

Huynh et al. [36] explored the use of CNNs for classifying the 6 basic facial expressions. Two CNNs are trained on the BU-3DFE database, with the first based on the facial appearance, and the second based on the 3D face shape. Both networks consist of convolution layers, ReLU, Pooling, Local Response Normalisation layers and Drop Out layers. The fully connected layers from the networks are concatenated and used for prediction, with the SoftMax function to evaluate the loss. They adopted a protocol to use all 100 subjects for training and testing instead of 60 and achieved good results, with the best performance of 92% coming from the joint effort of the 2D and 3D data.

Zhong et al. [121] split the face into small non-overlapping patches from which they try to categorise groups of patches. These categories are based on the relation between the different expressions, which include the *common facial patches*, *specific facial patches*, and the *rest*. They found that the only having the highly discriminative patches improves performance and that having too many patches causes a volatile decrease in performance. Essentially, including too many patches introduces noise that makes the task more challenging.

They propose ideas that mention looking within the whole face, analysing patches instead for better discrimination of expressions. The mentioned systems have interesting ideas that can be exploited and improved upon using recently advanced face detection, alignment and localisation technologies [61], [122]–[124]. They can be utilised to provide a consistent and less noisy approach. Facial parts such as the Mouth, Eyes, Nose and Eyebrows can be accurately captured and extracted. They can provide the samples to a framework that can analyse what facial parts work better and for which expression.

Having small patches as proposed by Zhong et al. [121] lacks the ability to automatically localise and determine what part is in effect for an expression. It can also have trouble distinguishing patches when applied to faces of different gender and ethnicity. Having a learned detector for the face and parts as proposed by Lv et al. [38] cannot guarantee the robustness as much as face detection, alignment and localisation techniques do, especially when there are peculiar samples provided. Both proposals used the JAFFE and CK+ database, which are not very diverse when it comes to their subject's ethnicities. In their experiments, they adopt a cross-validation approach, in which they do not ensure subject independence. This can result in a high performance which can be inconsistent if a new subject is ever tested.

This chapter looks to create a robust face alignment and localisation procedure to accurately crop out facial parts for further processing. A way to capture a facial part is to first locate it in the image, find the boundaries in which it consists in, and then crop the image based on this bounding box. The quality of the facial part extraction mainly relies on good detection of the boundaries. As the facial parts move with each expression, the bounding box cannot be determined at the subject level. It will have to be done separately for each expression by each subject.

There are many techniques that exist to locate a face in an image [61], [125], and other techniques that can produce the facial landmarks of that face [122], [124], [126], [127]. The combination of face detection and facial landmark localisation can provide all the information needed to create a bounding box for each part, only if the landmarks produced are sufficient for each facial part. The BU-3DFE and BU-4DFE databases both provide manually annotated 3D facial landmarks and frontal face images, but no direct mapping of the 3D landmarks on the 2D images. Therefore, they would require re-localising to obtain 2D landmarks to map directly on the texture face image.

4.3 Facial Parts Extraction using Geometric Information

To extract the facial parts from a sample, an approach using geometric information is investigated. This is an attempt to use the given 3D landmarks and map it on to the 2D frontal images of the subjects faces. This process is broken down into 4 stages, to get from frontal face image to facial parts processed and ready for the CNNs and hand-crafted algorithms.

For the first stage, the 3D landmarks are mapped directly onto the 2D frontal face image. The second stage is to apply pre-processing steps to make sure each sample of the frontal face is aligned and corrected before the facial parts are extracted. The third stage involves using the mapped and aligned 2D landmarks on the facial image to be able to crop out the different facial parts. The parts are then cropped using a bounding box that is generated from the landmarks. Finally, each of the cropped facial parts has post-processing steps applied to it, to make sure that they are of the same size across each subject. Once all the stages are complete, each facial part can be used for experimentation using hand-crafted and deep learning techniques.

Before the bounding box can be created for each facial part, the facial image should be adjusted in a way that can narrow the differences amongst each other. These include the size of the face, its rotation and facial parts localisation. These differences can be reduced by observing some of the key locations on a face that tend not to move with facial expressions. They can aid the image adjustment process, providing a reference point in which all the faces can be corrected to match.

For example, the inner eye locations will always remain in their position on a face. Using this information, the Euclidean distance can be calculated and used as a reference distance that other faces can match. This especially is a very effective method for resizing faces from the same subject that is portraying different expressions. There are also other techniques and the combination of locations that can aid the normalisation of the faces.

4.3.1 Spatial Mapping of 3D Landmarks

The facial images provided in the databases contain 2D texture information. However, the 3D landmarks are based on a 3D point system making direct the mapping of the 2D image close to impossible. Therefore, a protocol called Incremental Parallel Cascade of Linear Regression (iPar-CLR) [122] has been adopted to localise each 2D projected facial image. This technique is based on the Supervised Descent Method [128], [129], which was updated to be parallelised and to allow incremental updates when given new samples. The outcome of this technique is 49 landmarks for each facial image, that cover the inner parts of the face.

The iPar-CLR technique aims to reduce the variance of the perturbations at each level in a Monto-Carlo setting, training and updating all the levels of the regression functions independently using only the statistics of the previous level. This eliminates the need for propagation through all the previous iterations of samples. It becomes highly parallelisable in all the regression functions which means they can be trained independently and without any loss in the alignment accuracy.

iPar-CLR can be understood as the following [122]: let a set of M images $\mathcal{L} = \{I_i\}_{i=1}^{M}$ and the set of ground-truth shapes $\mathrm{S} = \{\mathbf{s}_i^*\}_{i=1}^{M}$ with $\mathbf{s}^0 \in \mathbb{R}^{n\times 1}$. Also, let a feature function $\mathbf{h}(\mathbf{I}, s) \in \mathbb{R}^{1\times h}$, where, h is the dimensionality of the feature. This function $\mathbf{h}$ returns the SIFT or HOG features around each landmark of shape $\mathbf{s}$ [128], [129] from an image $\mathbf{I}$. $\mathcal{P}^* = \{\mathbf{p}_i^*\}_{i=1}^{M}$ is a set of ground-truth shape parameters, with the goal of learning a function from an initial estimate of $\mathbf{p}$ that takes us to the ground-truth shape parameters $\mathbf{p}^*$, where, both $\mathbf{p}^*$ and $\mathbf{p} \in \mathbb{R}^{1\times l}$. l is the total number of shape parameters [122]. A linear rule is learnt from the perturbed parameters $\mathbf{p}^{(1)}$ of image $\mathbf{I}$ such as that of Equation 4.1 [122]:

$$\begin{aligned}\mathbf{p}^* &= \mathbf{p}^{(1)} + \mathbf{h}\left(\mathbf{I}, \mathrm{s}\left(\mathbf{p}^{(1)}\right)\right)\mathbf{W} + \mathrm{b} \\ \mathbf{p}^* &= \mathbf{p}^{(1)} + \left[\mathbf{h}\left(\mathbf{I}, \mathrm{s}\left(\mathbf{p}^{(1)}\right)\right) 1\right]\widetilde{\mathbf{W}} \\ \mathbf{p}^* &= \mathbf{p}^{(1)} + \tilde{\mathbf{h}}\left(\mathbf{I}, \mathbf{p}^{(1)}\right)\widetilde{\mathbf{W}}\end{aligned} \tag{4.1}$$

where $\widetilde{\mathbf{W}} = [\mathbf{W}; \mathrm{b}]$ is the learning objective, $\tilde{\mathbf{h}}\left(\mathbf{I}, \mathbf{s}\left(\mathbf{p}^{(1)}\right)\right)$ is a feature extraction function of the SIFT descriptor, $\mathbf{I}$ is an image from a set of M images, $\mathbf{p}$ is a set of perturbed shape parameters, with $\mathbf{p}^*$ as the ground-truth shape parameters. The initial $\widetilde{\mathbf{W}}^{(1)}$ regression problem is solved sequentially (known as Seq-CLR) as the following in Equation 4.2 [122]:

$$\underset{\mathbf{W}^{(1)}, \mathrm{b}^{(1)}}{\arg\min} \sum_{i=1}^{M} \sum_{j} \left\| \Delta\mathbf{p}_{ij}^{(1)} - \mathbf{h}\left(\mathbf{I}_i, \mathbf{p}_{ij}^{(1)}\right)\widetilde{\mathbf{W}}^{(1)} \right\|^2 \tag{4.2}$$

where the updated perturbed shape parameter is: $\Delta\mathbf{p}_{ij}^{(1)} = \mathbf{p}^* - \mathbf{p}_{ij}^{(1)}$, j is the number of perturbations, $\mathbf{I}_i$ is the ith image from a set of M images. The following perturbed shape parameter is calculated as shown in Equation 4.3 [122]:

$$\mathbf{p}_{ij}^{(2)} = \mathbf{p}_{ij}^{(1)} + \mathbf{h}\left(\mathbf{I}_i, \mathbf{p}_{ij}^{(1)}\right)\widetilde{\mathbf{W}}^{(1)} \tag{4.3}$$

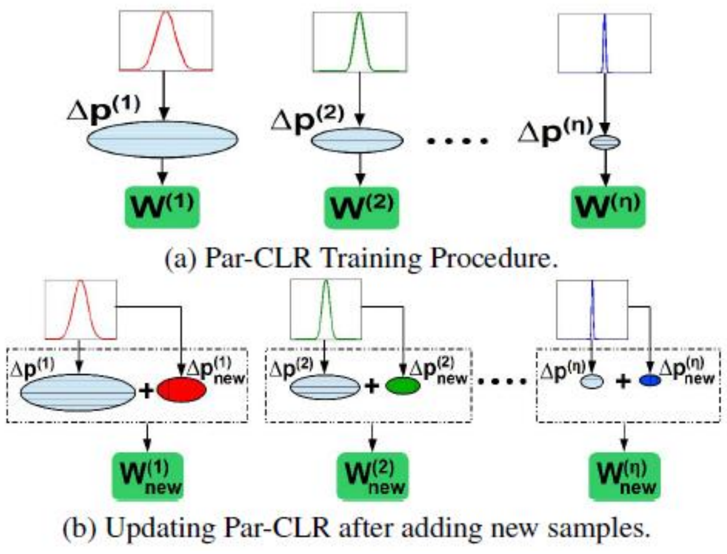

$\Delta\mathbf{p}^{(1)}$
$\Delta\mathbf{p}^{(2)}$
$\Delta\mathbf{p}^{(\eta)}$
$\mathbf{W}^{(1)}$
$\mathbf{W}^{(2)}$
$\mathbf{W}^{(\eta)}$
(a) Par-CLR Training Procedure.
$\Delta p^{(1)}$ + $\Delta p^{(1)}_{new}$
$\Delta p^{(2)}$ + $\Delta p^{(2)}_{new}$
$\Delta p^{(\eta)}$ + $\Delta p^{(\eta)}_{new}$
$W^{(1)}_{new}$
$W^{(2)}_{new}$
$W^{(\eta)}_{new}$
(b) Updating Par-CLR after adding new samples.

Figure 4.1 - Incremental update addition to the Par-CLR training procedure [122].

The problem with this is that it requires the previous perturbed shape parameter of $\mathbf{p}_{ij}^{(1)}$ which means it can't be run in parallel. And when new samples are added to the trained model, all the shape parameters must be retrained from scratch. Par-CLR fixes this issue by using the statistics of the previous levels (variance) instead, removing the need for relying on the propagated perturbed shape parameter from the previous iterations. This consequently allows all the perturbed shape parameters to be learnt in parallel.

Figure 4.1 shows a visualisation of the parallelised training procedure for Par-CLR (a), along with the incremental update in (b) [122]. iPar-CLR is applied similarly to Par-CLR, with the addition of allowing perturbed shape parameter to be updated by using new samples. This technique is applied to facial expression images to learn the 49 2D facial landmarks similarly in [115], which are used in the following stages of the framework to extract and isolate facial parts from a given facial image.

4.3.2 Facial Pre-processing Steps

Once the 49 2D landmarks $P_1, \dots, P_{49}$ are obtained as shown in Figure 4.2, the following step is to align each frontal face image so that they are normalised spatially. This can be achieved by using the landmarks to rotate and resize the facial images. Each image is rotated firstly to make sure the face is completely straight. This is to prevent any misalignment of hand-crafted descriptors. Subsequently, the face is resized within the image based on a given parameter, which will be the same across all samples. This is to make sure that all faces are roughly the same size within the image frame so that the facial parts can roughly be of a similar size across the different subjects and expressions.

4.3.2.1 Face Rotation Correction

To calculate the angle of the face, two locations are needed: a central origin point, and a line that goes through the upper face centrally. These can be obtained by observing the nose and mouth. As most of the face is symmetrical across a vertical plane, the landmark that is the centre of the upper lip (P_{35} in Figure 4.2) can be selected as a reference point. It is roughly central horizontally across the lower face and varies little in its position across the different expressions. A straight line can be produced along the nose using point P_{11} as $x_{11}y_{11}$ and the upper lip P_{35} as $x_{35}y_{35}$. The angle can then be calculated between this line and the norm. After observing the data, the consistency for P_{11} to be central proved to be an issue with the possibility that the subject may have suffered from a broken nose or any equivalent incidences that changes its natural shape. Therefore, to address this issue, a new landmark is generated to replace P_{11} that will be less likely to deviate much from each subject.

$$P_{50} = (x_{50}, y_{50}) = \left(\frac{(x_6 - x_5)}{2}, \frac{(y_6 - y_5)}{2} \right) \tag{4.4}$$

$$d_{50,35} = \sqrt{(x_{50} - x_{35})^2 + (y_{50} - y_{35})^2} \tag{4.5}$$

This landmark is generated near the centre of the forehead (P_{50}), created using the inner end of each eyebrow, which are landmarks P_5 and P_6. It is difficult for a person to move these inner eyebrow landmarks throughout the expressions, and because the face is symmetrical, the centre of the inner eyebrows is roughly at the centre of the face horizontally. This centre point makes a robust location to be used as the higher end reference point, replacing the nose point P_{11}. With the new point P_{50} generated using equation 4.4, the next step is to calculate the distance from that point and P_{35}. This is achieved by calculating the 2D Euclidean distance between them shown in Equation 4.5. The rotating angle S can be calculated as followed:

$$V = (v_1, v_2) = \left((x_{50} - x_{32}), (y_{50} - y_{32})\right) \tag{4.6}$$

$$Z = (0, v_2) \tag{4.7}$$

$$S = \cos^{-1}\left(\frac{VZ^T}{\|V\| \times \|Z\|}\right) \tag{4.8}$$

With the angle S, the face can be rotated at point P_{35} until this becomes 0°. Doing so will mean that the angle of the face is also 0°. This alignment can prevent straight lines being mistaken for curves or angled slopes across samples, which can have a negative influence on algorithms such as LBP and HOG by them producing incorrect feature histograms. Therefore, it is vital to keep all samples at the same angle. Figure 4.2 demonstrates the facial landmarks being annotated on a sample face, with the new point face rotated to 0°. The landmarks have also gone through the same transformation to ensure that they are properly aligned to the rotated face.

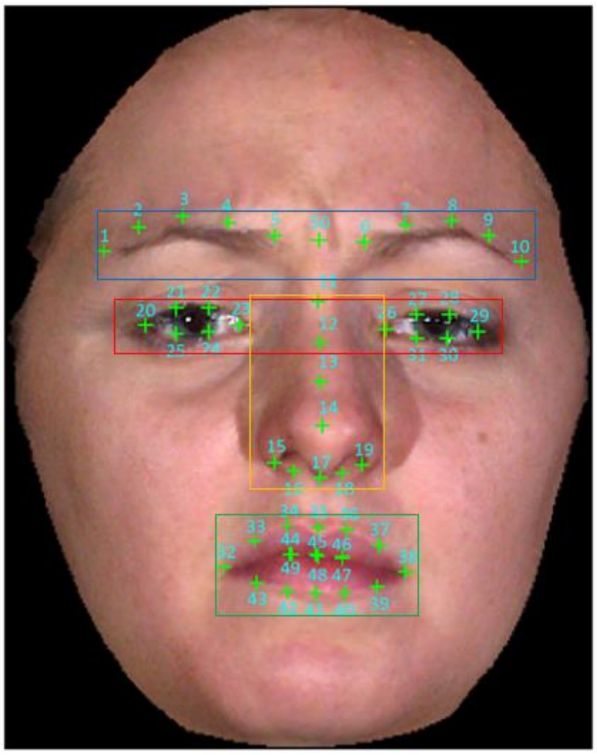


Figure 4.2 - Sample face with rotation correction. Facial landmarks annotated on the face as a green cross, with the number representing the order of landmark detected. Bounding boxes are outlined for the 4 main facial parts that will be extracted.

4.3.2.2 Face Resizing

Once all the faces are rotated to the same angle, the next step is to resize them to approximately the same size within the images spatial dimensions. This is a crucial step as it makes sure that the facial parts across the expressions are of the exact same size. This issue can occur during the capturing process of the faces. If the subject moves closer or further from the camera, the face can get bigger or smaller respectively. The uneven sizing can cause the feature descriptors to produce a different feature of the same image (that has smaller or bigger face).

Resizing the face to match the same size across the dataset can be tricky, and sometimes cannot be guaranteed between different subjects as their face shape may vary. However, it should be possible for each expression by a subject. This can be achieved by measuring a distance between two points on the face that does not vary between expressions. Then, if all the following samples are resized to produce the same distance between the same points, the face should ideally become the same size.

Two points that can be used are the inner eye corners P_{23} and P_{26}. This is preferred over the inner eyebrows points P_5 and P_6 as the eyes are more towards the middle of the face. The distance $d_{26,23}$ between these points do not vary throughout the expressions, which makes it a suitable candidate to be the reference distance. It is calculated using Equation 4.9, the Euclidean distance between both inner eye points.

$$d_{26,23} = \sqrt{(x_{26} - x_{23})^2 + (y_{26} - y_{23})^2} \tag{4.9}$$

Initially, a reference size for the inner eye distance is required that can be used on all the samples. This can be determined by taking the mean value of a random batch of faces. Each sample can then have the face resized evenly until the distance matches the reference.

4.3.3 Facial Parts Extraction

Once the face is aligned and resized correctly, the facial parts can be extracted using the new aligned and resized 2D landmarks. Four key facial parts are extracted for the upcoming experimentation. These are the Eyebrows; Eyes; Mouth; and Nose. This is achieved by creating a bounding box around each facial part using the relevant landmarks. The size of the edges for each bounding box is increased by approximately 5-10 pixels, to ensure that each facial part is fully inside each bounding box. Then, each image is cropped with its respective bounding boxes to separate the facial parts, producing 4 new images per sample.

The bounding box is created using 4 points across the corners of the facial parts. To determine the corners of each facial part, a few steps are required to prepare for this process. The first step is to isolate all the landmarks relevant to each facial part. They can be determined by visually inspecting the landmark positions in Figure 4.2. The Mouth consists of the points from P_{32}: P_{49}, Eyes are points

P_{20}: P_{31}, Eyebrows are points P_1: P_{10} and finally Nose are points P_{11}: P_{19}. To determine the corners of each facial part, the Min and Max x and y points are required within the points set. From this, 4 combinations are created to representing each corner.

Given the set of all 50 landmark points as P:

$$\begin{gathered} \mathrm{P} = \{P_1, P_2, \cdots, P_{50}\} \\ P_i = \left(v_1^i, v_2^i\right) \\ \widehat{\mathrm{P}} = \left\{\hat{P}_1, \cdots, \hat{P}_k\right\} \end{gathered} \tag{4.10}$$

Where P_i is the point i, which is a subset from $i \in [1{,}50]$, and $\widehat{\mathrm{P}}$ is a subset of points for a specific facial part. The bounding box is created using 4 points representing each corner as:

$$\begin{gathered} x_{min} = \min\left(v_1^1, \cdots, v_1^k\right) \\ x_{max} = \max\left(v_1^1, \cdots, v_1^k\right) \\ y_{min} = \min\left(v_2^1, \cdots, v_2^k\right) \\ y_{max} = \max\left(v_2^1, \cdots, v_2^k\right) \end{gathered} \tag{4.11}$$

$$\begin{gathered} P_{Tl} = (x_{min}, y_{min}) \\ P_{Tr} = (x_{max}, y_{min}) \\ P_{Bl} = (x_{min}, y_{max}) \\ P_{Br} = (x_{max}, y_{max}) \end{gathered} \tag{4.12}$$

Where P_{Tl} is the top left corner, P_{Tr} is the top right corner, P_{Bl} is the bottom left corner and P_{Br} is the bottom right corner. With all the corner points of the bounding box calculated for the requested facial part, the next step is to simply crop the image with the bounding box. This step will isolate the requested facial part of the whole face.

4.3.4 Facial Parts Post-Processing Steps

Not all facial parts are the same in size across all the samples, therefore another processing step is taken to resize all the parts into 64x64x3 images. This is to provide consistency in size for the hand-crafted and deep learning algorithms. Resizing of the image should not alter the facial parts differently, as the initial face resizing that has taken place is done on the face within the image. Therefore, each part should still be normalised according to the initial resizing. Now the facial parts of each sample are ready to be used in the upcoming experiments. Figure 4.3 shows sample images of the Mouth after all the processing techniques. The Happy, Neutral and Fear expressions are presented on the same subject.

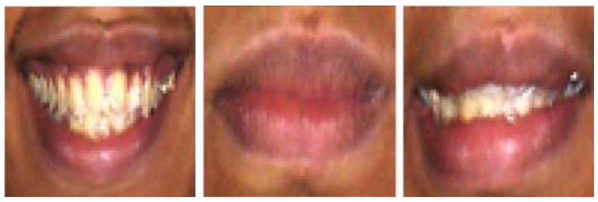


Figure 4.3 - Samples of a subjects Mouth portraying the Happy, Neutral and Fear expressions. This is the resulting images with all the processing steps before and after extraction.

4.4 Deep Network Designs for Understanding Facial Parts

In this section, the use of Convolutional Neural Networks designed for facial expressions is described in detail, for both facial image and parts. Several CNN architectures are designed and tested on the data, to see how their performance varies based on their structural differences. There are many different building blocks to a CNN, making the design process very tricky to know what the right or wrong thing to try is. Therefore, the architectures that are designed, are inspired by the existing networks that have shown their effectiveness towards their application.

With CNNs, there are two key factors to consider when creating a network. Firstly, how much data is available for training. The second is how deep does the CNN really need to be. The general rule is that the quantity of training data helps a deep network perform better. With this in mind, networks are designed deeper if there is a lot of data to play with, and usually show better performance with the cost of longer training time and extra computational and memory resources required. However, the depth of the network also depends on the difficulty of the task. For instance, if the task is to classify 1000 different objects, then this would require a very deep network to create all the associations necessary for each object. However, if there are only 10 objects to classify, then the task becomes significantly easier making a very deep network a waste of time and resources to train, as the same performance can be possibly achieved using a smaller network.

To test this theory out, one of the experiments will focus on comparing the performance of assorted size network architectures using the same data and parameters. This will give an indication if there is any improvement of using a deeper network with a small size database.

4.4.1 Data Preparation and Pre-processing Steps

Preparing the data for any system that goes through data mining tasks can have a really large impact and contribution on how well the system performs [130]. Training a CNN directly on raw images may not be the most efficient way to do so. Preparation can include tasks such as data acquisition; cleaning; format and structure; storing and handling. Some of the data pre-processing steps mentioned can be applied to speed up training and even increase the performance of a CNN. Initially, all the samples must be of the same spatial dimensions. Then, each image can be rank normalised between 0 and 1, which is also known as rescaling. This can be achieved easily with images due to the known pixel value range of 0 to 255. Each pixel from all the images can be divided by 255 to achieve this.

Another step is to standardise the images, which is essentially calculating the zero mean unit variance. This is done by calculating the mean of all images and subtracting this value with all the images so that the new mean of all images is now equal to zero. The images are then divided by the standard deviation of all images. The formula is shown in 4.13, where μ is the mean of vector **x**, and σ is the standard deviation of **x**.

$$\bar{\bar{x}}_i = \frac{x_i - \mu}{\sigma} \tag{4.13}$$

Another concept that can be applied to achieve better performance from CNNs is to augment the data. This can provide a form of regularisation and effectively to increase the quantity of data. This can be advantageous when training the network as is gives more depth and variation of the data, which in result can also reduce or even prevent overfitting occurring.

Some of the common data augmentations are:

- Rotation – An image is rotated uniformly from the centre point with an angle of 0 to 360 degrees.
- Image Flipping – An image is flipped vertically or horizontally to produce an alternate sample.
- Image Rescaling – An Image can be uniformly rescaled within the spatial dimensions, to produce a distinct size of the image within its initial dimensions.
- Image Stretching – Where an image can be stretched vertically or horizontally.
- PCA reduction – This is a step that requires calculating the zero-mean beforehand. Then, the covariance matrix is obtained to understand the correlation of the data. This data is then decorrelated by calculating the eigenvectors using singular vector decomposition, and multiplying it with the zero-centred images. The dimensionality can be reduced by multiplying the zero-centred images with only a subset of the top eigenvectors.
- PCA-Whitening – This is applied directly to the decorrelated data, to normalise the scales of every dimension. This is achieved by diving the decorrelated data with the eigenvalues.

For the experiments, data augmentation is also considered, with the possibility to expand on the total number of samples. From the lists above, the following techniques will be adopted: image normalisation between 0 and 1, calculating the zero mean with unit variance, and image flipping is used for augmentation. The images will be flipped horizontally as opposed to vertically, or both. Doing so vertically is an impractical solution as there will rarely be any realistic situations where the mouth, or other parts, are positioned upside down.

4.4.2 Pre-Training and Fine-Tuning Networks

The idea of pre-training a network is to learn from existing data before the network is applied to the data from the active task. Once this network is trained, it becomes a pre-trained network. There are 2 useful applications of a pre-trained network. The first application is to use this network to propagate new data forwards only. The prediction can then be taken from the network, or even the features produced from the previous layers (pre-trained feature extraction). The second is to further train the deep network with the new data, which has the benefit of starting with optimised weights from its previous application. This process is also known as fine-tuning a network.

Razavian et al. [131] had considered how an off-the-shelf OverFeat model [132] worked for other applications. The pre-trained model they used was successfully applied on a variety of datasets for recognition tasks such as object image classification, scene recognition, fine-grained recognition, attribute detection and image retrieval. They simply extracted CNN features coupled with a linear Support Vector Machine for recognition and managed to achieve state-of-the-art performances at the time on those datasets. Using this idea, we plan to investigate if pre-trained networks can be exploited successfully for emotion-based applications such as Facial Expression Recognition.

There are a variety of publicly available pre-trained networks that have been trained on existing data. These include applications such as object recognition [34], [96], [97], face recognition [133], [134] and semantic segmentation [135], [136]. These networks can be utilised for extracting pre-trained features or fine-tuning from the facial expression images. To get the best response from the pre-trained network, there usually is a given set of pre-processing commands used on the images before propagating them through the network. This is to make sure they are in the same form before they are propagated (zero mean, normalised, rescaled).

For the upcoming experiments, the use of a pre-trained network for feature extraction will be considered for FER. The following pre-trained networks are considered: AlexNet [34] and VGG-Face [133]. AlexNet is suited for object detection, but it will be interesting how the performance will differ to VGG-Face (trained on faces). A network will also be trained from scratch based on the BU-4DFE database, using a set of sample frames of each sequence as the training data. The network architecture will be based on combining facial parts, so all of the training occurs in one session.

4.4.2.1 AlexNet Pre-Trained Deep Network

AlexNet is a popular deep network that was known for its successful demonstration in the ImageNet challenge [33]. It was designed by Alex Krichevsky, which at the time was not attempted for such challenges. The results achieved in 2012 had surprised the research area after it achieved 16% error, which was at 25% in 2011. It had produced a significant increase in performance compared to other hand-crafted techniques, and the error decrease trend was then followed yearly in the ImageNet challenges by other deep networks.

The AlexNet pre-trained CNN contains a 21-layer architecture that has 8 parameter layers, 5 of which are convolutional layers and 3 fully connected layers. The remaining are a mix of Pooling, Rectified Linear activation function and Normalisation layers with a single SoftMax layer at the end. As this network is used as a pre-trained feature extractor, the model will not be retrained. The main focus is to know which layers to extract the feature from when propagating facial expression samples through the network.

The areas of interest are the Fully Connected layers FC1 and FC2, as they contain information that connects all the previous feature maps. They are also the smallest form of features as the spatial dimensions are 1×1. The initial convolution layers are not considered to be used as the extracting feature as the parameter and memory count is drastically higher than the fully connected layers. With AlexNet, there were around 70K parameters VS. 4096 when comparing the initial convolution layer VS. the fully connected layers. For VGG-Face, this number increases significantly to 802K VS. 4096. These are too large to be used directly as a feature vector. Therefore, the targeted layers for this experiment are 16 and 18, which are represented as FC1 and FC2 respectively. This network is designed for recognizing up to 1000 various objects, which may result in unsuitable features when applied with facial images.

4.4.2.2 VGG-Face Pre-Trained Deep Network

VGG, known as Visual Geometry Group, is a group from Oxford, UK, that have designed and trained deep network models. Their works include several popular networks based on object detection. These are the VGG-S, VGG-F, VGG-M, VGG-D and VGG-E networks [137], [138] networks trained on the ImageNet dataset [33]. All the networks vary in speed and layer size, with the very deep networks VGG-E and VGG-D (16 and 19 convolution layers) performing a lot better than the VGG slow, fast and medium (all containing 5 convolution layers) network. This can be a demonstration of how the depth of the network can help in performance when there are millions of data samples.

VGG-Face is the network they trained for face recognition amongst 2622 celebrity faces. This had shown great performance on many of the top face databases, achieving state-of-the-art performance [133] on the YouTube Faces Dataset [139] at the time. This network contains 37 layers in total. These consist of 13 convolution layers; 3 fully connected layers; 15 ReLU Activation layers; 5 pooling layers; and a final SoftMax layer at the end to calculate the loss. Each of the convolution layers has a kernel size of 3×3, padding of 1 and stride of 1. This is the style of convolutions that VGG networks are known for, as the input image becomes the same in spatial dimension as the output feature maps produced by the filter. The condensed form of the network architecture can be seen in Figure 4.4, where each visual convolution layer from Conv1 to Conv4 represents a set of 3 convolutions. All layers apart from Input Image and SM1 have a ReLU Activation layer after it, and Conv1 to Conv4 have a max pooling layer after the ReLU layers, reducing the image spatial dimensions by 2.

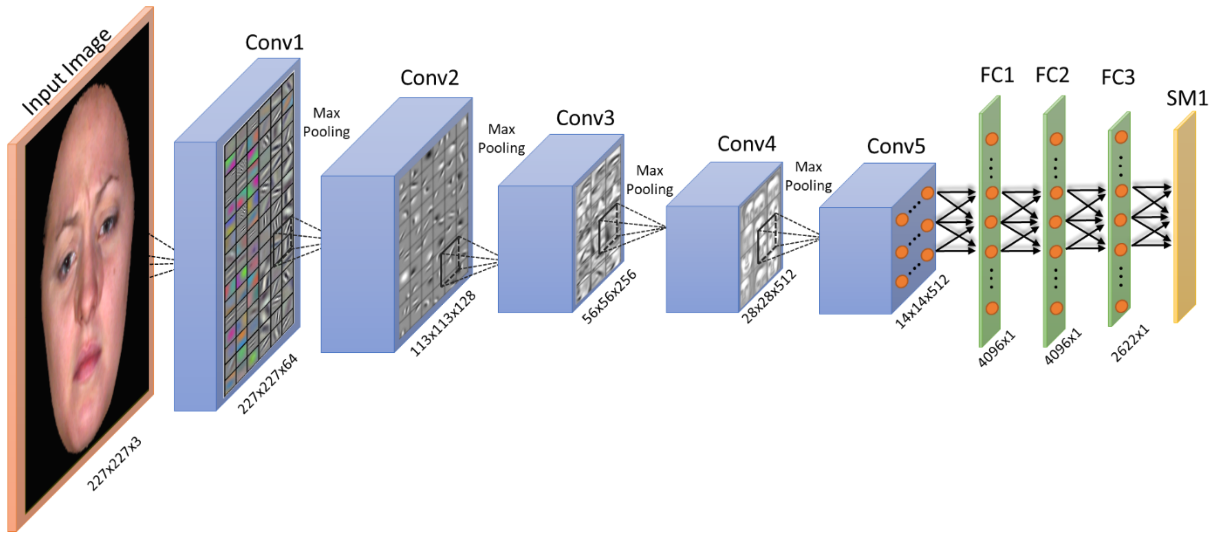


Figure 4.4 - Visual process of the VGG-Face Network Architecture

The VGG-Face network had been designed and trained to recognise faces from a list of 2622 famous people. Compared to AlexNet, this network is trained solely on faces and will, in theory, be more beneficial for FER. The early convolutional layers contain details in the form of edges, blobs, facial parts, but do not have any interconnectivity between them. This is handled by the fully connected layers, linking the blobs and facial parts that produce a response.

$$y_j = f\left(\sum_{i=1}^{m} x_i \cdot w_{i,j} + b_j\right) \tag{4.14}$$

The interconnectivity can be interpreted in Equation 4.14, where y_j is the fully connected output feature by taking the function $f(x)$of the given input x_i from the previous layer. The function calculates the sum of all inputs x_i multiplied by each individual weight $(j = 1{:}\,4096)$ of the fully connected layer plus the bias b_j. These layers will also be the focus of the upcoming experiments, with FC1 and FC2 extracted as 4096-dimensional feature vectors used as the deep representations of the given samples.

4.4.3 Network Architecture Designs

There will be 4 forms of network designs in total. The first is to focus on learning an image of the frontal face, which can also be adopted for each individual facial part. Here a single branch network is designed to only learn 1 image at a time. The facial parts network contains 4 branches that gradually concatenate to 1, where each branch learns a specific facial part. This is to try and get a deep understanding of each facial part and fuse the relations between them via parameter concatenation and the use of fully connected layers.

A network is also pre-trained using the BU-4DFE database, from which fine-tuning will occur to see if external data of a similar nature will benefit FER. The pre-training will be based on the facial parts using the same procedure as on the BU-3DFE database, and the fine-tuning will be to simply apply BU-3DFE samples for light re-training and prediction. Finally, a Joint Bayesian technique will be adopted on top of the trained networks. These networks will be treated as pre-trained networks for feature extraction, from which JB is applied to find similarities amongst the expressions.

4.4.3.1 Comparison of Small, Medium and Large Deep Network Architectures for FER

Having substantial amounts of data is a key ingredient to get a deep network to perform the best it can. The databases used here for facial expression recognition are very small (up to 3000) when compared to the ImageNet database [33] that has millions of images. The ImageNet database has shown how deep networks can learn a lot from enormous quantities of data. Training a small deep network on a small database will perform similarly to training a large deep network as the lack of data is what will cause the large network to overfit the data. Part of the experimentation will investigate how much difference there is when comparing small, medium and large network architectures, under controlled settings.

The networks are designed using a similar style to the VGG-Face network [133] as it has a consistent architecture structure that can be manipulated to be small or large. The VGG-Face network architecture features a design that has a combination of convolutional and ReLU layers that do not change the spatial dimensions of its the input. This allows the pooling layers to have full control over reducing the spatial dimensions and allows to easily make the network deeper if needed.

The training iterations will be set based on when the objective converges to a steady value. This may not be the same for the large and small networks as the large network may take longer to converge. Therefore, sufficient training will be needed to allow for fair comparison. The networks will be trained on frontal face images of spatial resolution 128×128, and with RGB colour. For facial parts, the spatial resolution of them will be 64×64.

4.4.3.2 Network Architectures

A network is designed to learn from facial images and facial part individually. It will come in three variants, a small, medium and large version. The small version will be created to be a fast and efficient network. The large will be a slower network with more memory usage but will be able to learn data at a higher depth. The medium version sits in between to provide a deeper network but with fewer memory requirements than the large version.

Some standard design rules are followed.

- Batch Normalisation is applied immediately after every convolution layer, they are paired together.

- Each convolution layer (apart from prediction) + Batch Normalisation is followed by a ReLU activation layer.
- When a pooling layer is defined, Max pooling is always chosen, apart from the final pooling layer to create 1x1 feature maps, where average pooling is applied instead.
- Activation of all parameterised layers is done the same for all created networks, which is using the Xavier improved technique proposed in [89].

The networks are based on similar principals as the VGG-Face network but modified to accommodate the image spatial dimensions. The Convolutional layers will always produce an output of the same input spatial dimensions and will always be followed by a Batch Normalisation layer and a ReLU layer. The Pooling layers will shrink the spatial dimensions throughout the network. The deeper network will contain more convolutional layers before a Pooling layer is applied. This will be considered the differentiation point between small, medium and large Deep networks for this experiment.

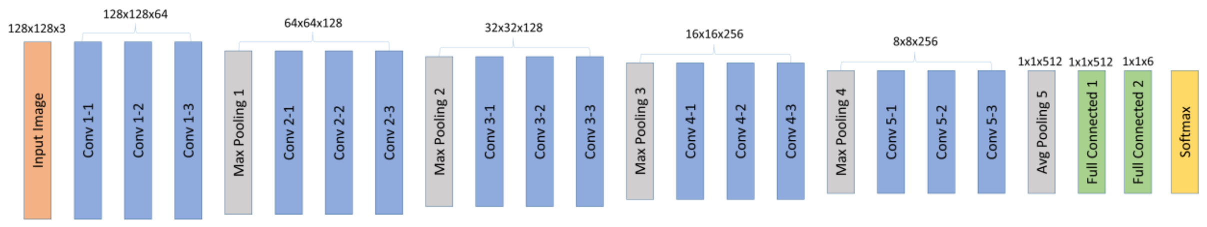


Figure 4.5 - Architecture of the Large Size Deep Network

Figure 4.5 shows the architecture of the large deep network. It starts with the input image, followed by 3 sets of convolutional layers. These convolutional layers will produce feature maps of the same spatial dimensions, with a depth of 64. They are followed by a max pooling layer to reduce the spatial dimensions of the feature maps to 64×64. This sequence is then repeated 4 more times, decreasing the spatial dimensions and increasing the feature maps. After layer Conv 5-3 an average pooling layer is applied reducing the 8×8 feature maps to 1×1. This is then followed by 2 fully connected layers and a SoftMax layer.

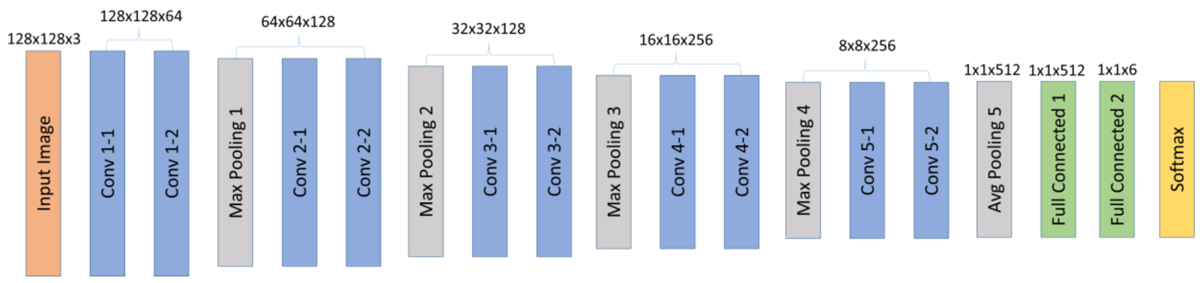


Figure 4.6 - Architecture of the Medium Size Deep Network

In Figure 4.6, the medium size network is shown with the only difference of having 2 convolutional layers per sequence. Figure 4.7 is the small architecture containing just 1 convolutional layer per sequence. Each Conv layer will have a kernel size of 3×3, with a stride of 1 and a padding of 1. The

Max pooling layers will have a stride of 2 and a kernel of 2×2. The average pooling layer will have a stride of 1 and a kernel of 8×8, and finally, the Fully Connected layers will have a stride of 1 and kernel of 1×1.

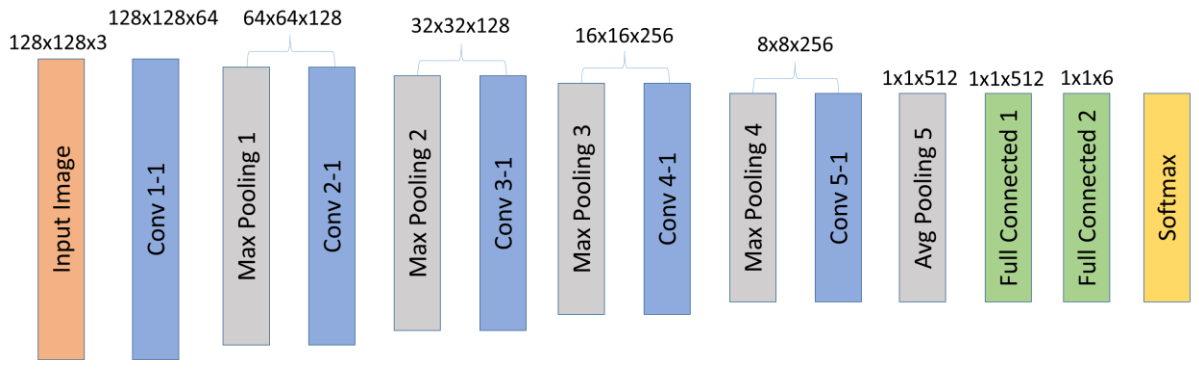


Figure 4.7 - Architecture of the Small Size Deep Network

For the facial parts, the initial spatial dimension will be 64×64, and the architecture is adjusted appropriately to accommodate this factor. The architecture can be seen in Figure 4.8, with the proposed solution of removing the Conv 5 sequence. This will allow keeping the same style architecture used for the frontal face images. This architecture is repeated for the Large and Medium variants for comparison using individual parts. The remaining hyper-parameters will be set the same to allow a fair comparison.

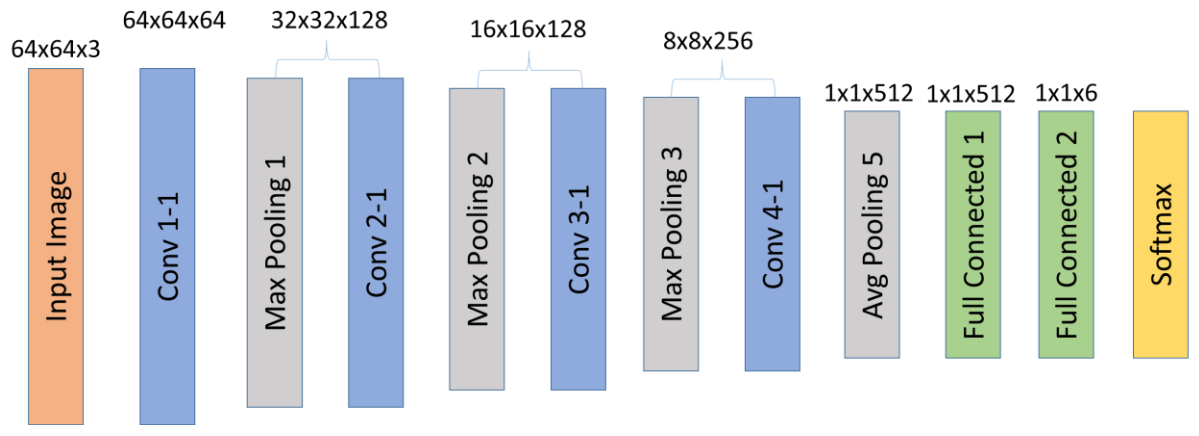


Figure 4.8 - Small Size Deep Network architecture for facial parts

4.4.4 Dynamic Joint Bayesian Approach from Feature Learning

An interesting approach is to look at how the deep networks can learn the differences between each expression. The fully connected parameters contain information about all previously activated neurons and will be a good place to investigate.

Techniques like KNN often use distances such as the Euclidean distance directly between the feature vectors. There are some major downfalls to this idea, as the data can be very correlated within its dimensional space. Metric learning, namely Mahalanobis distance, looks to learn a new metric that can make two classes more separable by removing the covariance between the vector dimensions. This can be achieved by decorrelating the data similar to PCA, and calculating the Euclidean distance within the eigenvector space.

Joint Bayesian is a technique that has been successful for detecting face similarity [140], [141]. It tries to find the probability of a sample belonging a class, with the aid of learning the shortest Mahalanobis metric distance. Chen et al. [140] looked to find the similarity of a given face compared to 2000+ other possible faces. They represent a sample face as $x = \mu + \varepsilon$, where μ is the identity of the face and ε is the variation of the face that can consist of lighting and expression changes. $\mu \sim N(0, S_\mu)$ and $\varepsilon \sim N(0, S_\varepsilon)$ are the Gaussian distributions from which S_μ and S_ε are covariance matrixes. The joint likely-hood ratio can be assumed as follows [140]:

$$r(x_1, x_2) = \log \frac{P(x_1, x_2 | H_I)}{P(x_1, x_2 | H_E)} \tag{4.15}$$

where H_I is the hypothesis that the identity of μ_1 and μ_2 are the same, and H_E is the hypothesis that intra-person variations ε_1 and ε_2 are not the same. This can also be represented as Equation 4.16 [140]:

$$\log \frac{P(x_1, x_2 | H_I)}{P(x_1, x_2 | H_E)} = {x_1}^T A x_1 + {x_2}^T A x_2 - 2{x_1}^T G x_2 \tag{4.16}$$

where A and G are negative semi-definite matrices. The covariance matrices S_μ and S_ε are therefore required to get the joint likely-hood ratio, from which they can be learnt and described by Chen et al. [140], using their proposed EM-like algorithm to jointly estimate the matrices.

This idea can also be manipulated for facial expression recognition, by determining the highest similarity of a given expression to each of the possible expressions. However, it may be that the whole face can provide some noise (parts of the face that do not relate to any expression), especially when comparing different genders, ethnicities and skin colour. This can cause groupings of these noise similarities that can be unrelated to how the expression changes. Therefore, using facial parts can seem like a positive move towards removing the biases that go towards the facial anatomy.

Metric learning is adopted to handle the issue of dealing with more than two classes. It shares similar properties as the Joint Bayesian face [140], that can be exploited to provide a solution for classifying expressions. Metric learning can be understood by the following equation [140]:

$$\mathrm{Md} = (x_1 - x_2)^T M (x_1 - x_2) \tag{4.17}$$

where M is a positive definite matrix that defines the Mahalanobis distance Md of samples x_1 and x_2. This essentially presents a solution to learn the best distance for two given samples by observing the features of each when they are decorrelated. This results in the distance of the variability between them.

Chen et al. [140] proposed reformulating the metric to better preserve the separability, which is reformulated to:

$$Md = (x_1 - x_2)^T A(x_1 - x_2) + 2{x_1}^T(A - G)x_2 \tag{4.18}$$

which in return can be rewritten as [140]:

$$Md = {x_1}^T Ax_1 + {x_2}^T Ax_2 - 2{x_1}^T Gx_2 \tag{4.19}$$

This equation is broken down into 3 terms of ${x_1}^T Ax_1$, ${x_2}^T Ax_2$ and $2{x_1}^T Gx_2$. The protocol used for learning the facial expression samples is to initially compute the A and G term on a subset of training samples containing all types of expressions. ${x_1}^T Ax_1$ and $2{x_1}^T Gx_2$ will be based on the test samples, and are now referred to as T_1 and T_3 respectively, with the term ${x_2}^T Ax_2$ that will be a pre-computed dictionary for the training samples, referred to now as T_2. Each new test sample has T_1 and T_3 computed, and the distances of the test sample compared to all the training dictionary are computed as $Y = T_1 + (T_2 - T_3)$. To find the best matching expression, Y is sorted in descending order, and a K-nearest neighbour approach is applied to determine the predicted class.

The Joint Bayesian principal can be applied on the CNN Fully connected layer features. The CNN feature will continually learn the facial parts, and what makes them unique to each expression at every training iteration. This is whilst the Joint Bayesian technique calculates the most relevant distance that can find the best similarity matches between the features. PCA can be applied to the CNN feature to reduce the dimensionality and focus in only keeping the variance.

The upcoming experiments will train CNNs on the facial parts, from which this proposed Joint Bayesian approach will run in parallel. This will be applied to the individual facial parts, the whole face and the combined network using all the facial parts.

4.5 Related Datasets

The works from this chapter is a continuation of Chapter 3, but with the introduction of deep learning protocols, and a different viewpoint to observe the face. The main database remains the same, to give a thorough comparison of deep features against hand-crafted techniques. The details for each database have been mentioned in the previous chapters. Therefore, this section will mainly just highlight how each dataset will be used for the experiments.

The main experiments will be run on the BU-3DFE. From these, the textured and geometric data will be utilised. The BU-4DFE database will also be used but for pre-training a network which will then be retrained using the BU-3DFE database. The initial training is achieved by collecting key frames from each temporal sequence. The intensity of each expression in a sample starts from neutral and hits the peak roughly in the middle of the sequence, which then returns to the neutral state by the end of the sequence [14]. Based on this, several frames can be taken for near the centre of each sample to capture each expression. These frames will then be used as samples for pre-training a network that will be used with the BU-3DFE database.

The Binghamton University 4-Dimension Facial Expression (BU-4DFE) Database [14] is similar to the BU-3DFE database, but with an added time dimension. There are 101 subjects included, each with 6 emotions (Angry, Disgust, Fear, Happy, Sad and Surprise). From this, 58 are female and 43 are male, with a variety of racial backgrounds/ethnicities, including Asian; Black; Hispanic/Latino; and White.

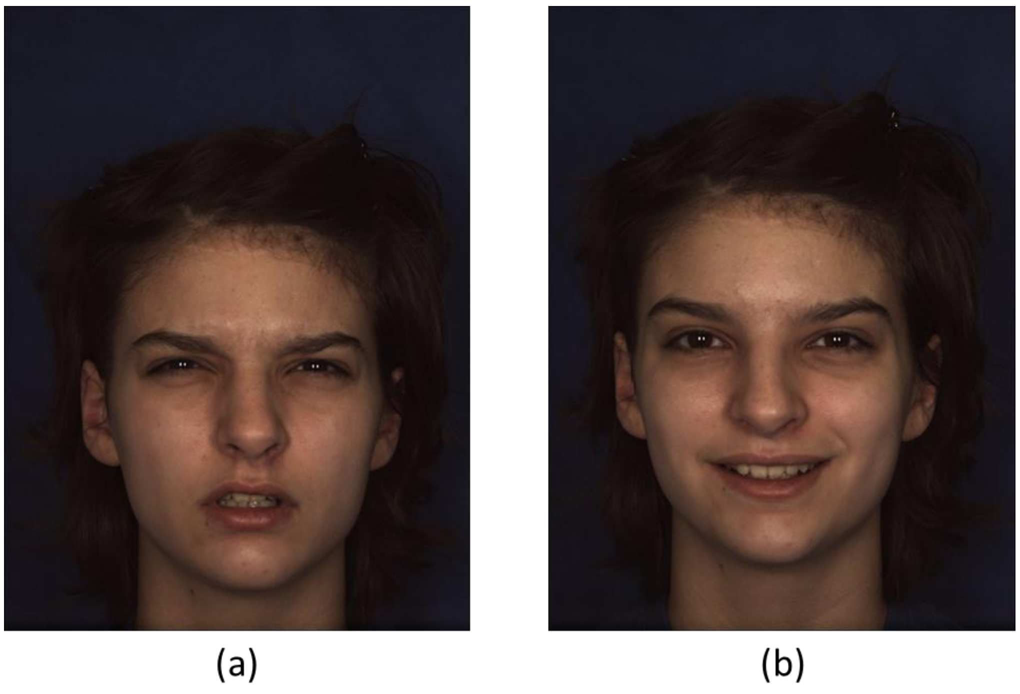


Figure 4.9 - Samples from the BU-4DFE database showing (a) Angry expression and (b) Happy expression

Each emotion for each subject has a 3D facial expression sequence; captured and generated using a Di3D system. This includes a 3D model sequence and a 2D texture video. There is a total of 606 samples at 25 frames per second. In each sample, the subjects face starts from a neutral state, goes to the acted emotion and then back to the neutral state. The 2D texture videos have a spatial resolution of 1040x1329 pixels per frame which contains the subjects cropped face along with a navy background, as demonstrated in Figure 4.9.

4.6 Experimental Setup & Results

The following experiments will be based on understanding how facial parts can be utilised for better discrimination of facial expressions. This idea will be compared with the experiments in Chapter 3, which was applying hand-crafted techniques on the whole face for facial expression recognition. Deep learning techniques will also be implemented to learn from the face and the extracted facial parts. This is to study and apply recent technologies and advancements of machine learning, demonstrating how they can process facial expressions.

There will be four experiments taking place. The first will focus on applying hand-crafted techniques on the facial parts using the same protocol and settings from 3.3. This will demonstrate if using facial parts instead of the whole face is beneficial for FER. The second experiment will focus on applying deep learning techniques to the frontal face images and facial parts. A small, medium and a large-sized deep network will be tested and compared to see if there is a performance difference. The third experiment will apply pre-trained deep network approaches on the whole face and facial parts. This is to investigate if using existing networks trained on similar and different image content will make a difference. Finally, the fourth experiment will apply a Joint Bayesian approach at the decision level to see if there are improvements when finding the best metric distance between each expression.

4.6.1 Settings and Protocols

The main experimentation will be applied on the BU-3DFE database, as it is consistent in samples per expression. The experiments will be extensive and the settings for each will be mentioned at the start of each experiment. The BU-4DFE database is also utilised for one of the experiments, with its data used to pre-train a network for further experimentation on the BU-3DFE database.

For the experiments that do not require training a deep network, there will be one protocol adopted for them. This protocol is the same as Protocol 1 from Chapter 3. That is to use 60 of the 100 subjects in the tests that are randomly chosen. 54 are used for training and 6 used for testing. From each subject, the top 2 intensities for each expression are used, making 720 expression samples in total across all subjects. There is a total of 100 tests carried out in a subject independent manner, where no samples from a subject will be found in both the training and testing sets at the same time. In each test for each facial part/face, 10-fold cross validation is applied for an extensive and fair result. The average result of taken from the 100 tests which will be used as the final recognition rate for that approach.

The experiments specific to training deep networks will be implemented on a publicly available deep learning toolbox called MatConvNet [142]. This will be used in the MATLAB 2017a environment, where the whole framework is designed and simulated. The tests will follow a similar Protocol of running 10-fold cross-validation, using 60 out of 100 subjects. However, for the networks that require training or re-training, they will only be tested once with 10-fold cross-validation as it is a lengthy procedure. Some general setting for the training of all deep convolutional neural networks include the optimiser set to Stochastic Gradient Descent, Nesterov's Momentum update [143] used to improve the convergence, and Xavier's improved method [89] which is used to initialise all the network parameters. The remaining hyper-parameters will be defined in each of the experiments as they may vary between them.

4.6.2 Experiment 4A – Facial Parts Analysis using Hand-Crafted Techniques

This experiment is based on applying several hand-crafted techniques on the separated facial parts images. The framework for this experiment will be largely based on the previous experiment, but with the addition of facial part fusion, alongside feature fusion. Initially, each facial part is tested on separately, with and without feature level fusion. This is to judge the contribution of each facial part. After, the facial parts are fused with the different possible combinations. From this, we can evaluate if a facial part has contributed any improvement towards FER or not.

Each sub-test will be thoroughly tested to give an accurate result, running 100 times with each based on 10-fold cross-validation. The parameters will be the same as used in the previous chapter. The feature dimensionality for each descriptor is as follows: $ULBP = 3{,}712$, $LPQ = 4{,}096$, $HOG = 1{,}984$, $EOH = 1{,}536$, and finally, $Fused\ Texture = 3{,}712 + 4{,}096 + 1{,}984 + 1{,}536 = 11{,}328$. PCA is used in all tests for dimensionality reduction, retaining 99% of the feature variance.

4.6.2.1 Individual Facial Parts Test

The first test will give an indication of how using facial parts individually compares to the whole face. Using the face has already shown to be an effective tool for FER from the previous experiments in Chapter 3. This test aims to break down how much each part of the face contributes towards the performance and to try find areas of improvement.

Table 4.2 - Performance of each individual facial part using hand-crafted descriptors ULBP; LPQ; HOG and EOH.

Facial Part	Feature	SVM RBF Accuracy	SVM Poly Accuracy
Eyebrows	**ULBP**	**42.15%**	**43.70%**
	LPQ	40.64%	41.75%
	HOG	41.64%	43.61%
	EOH	40.93%	41.78%
Eyes	**ULBP**	**46.93%**	**47.76%**
	LPQ	45.13%	47.39%
	EOH	42.81%	43.65%
	HOG	46.39%	47.53%
Mouth	**ULBP**	**72.78%**	**73.34%**
	LPQ	72.43%	73.08%
	HOG	72.56%	73.04%
	EOH	72.12%	73.09%
Nose	**ULBP**	**48.80%**	**50.69%**
	LPQ	45.53%	47.42%
	HOG	48.16%	49.41%
	EOH	48.45%	48.81%

Currently, only the Eyebrows, Eyes, Mouth and Nose are extracted from the face, which are considered the parts that stand out the most from a face. It will be interesting to find out if parts of the face show better performance for certain expressions over others. An example can be the mouth, which visually contributes a lot to the surprise expression as it deforms significantly from its neutral state. This section will include the individual performance for each facial part using the feature descriptors ULBP, LPQ, HOG and EOH. SVM will be the only machine learning technique used, with both RBF and Poly kernels tested. The hyper-parameter selection for both feature extraction and machine learning have been optimised based on the previous experiments in Chapter 3.

Table 4.2 has provided some interesting findings, with a clear demonstration that the mouth contains the most expressive information for each expression. It has in-fact performed closely to using the whole face. The Nose and Eyes performed similarly, with the nose hitting just above 50%. The Eyebrows have performed the worst, 4.06% behind the Eyes performance, and 29.64% behind the best performance produced by the Mouth.

For all of the facial parts, the ULBP feature has performed the best amongst the rest. For the Mouth facial part, the feature descriptors have performed similarly with a 0.3% difference between the best and the worst. For the other facial parts, the differences are slightly bigger.

Table 4.3 - Performance of each facial part using fused texture descriptors

Facial Part	Feature	SVM RBF Accuracy	SVM Poly Accuracy
Eyebrows	**Fused Texture**	44.59%	44.86%
Eyes	**Fused Texture**	48.85%	50.02%
Mouth	**Fused Texture**	**73.86%**	**74.52%**
Nose	**Fused Texture**	50.22%	52.75%

Table 4.3 shows the performances of each facial part when all the texture features are fused together. They show a similar trend in the ranking of the best facial part, but there is an improvement for each facial part when the texture features are fused. The biggest improvement is achieved by the Eyes facial part, increasing by 2.26%. Table 4.4 shows the confusion matrix of the Mouth facial part. The Surprise expression has the best accuracy of 90.2%, followed closely by Happy with 87.9%. Both expressions have a heavy emphasis on the mouth, with unique signatures of the movement that can easily be distinguished amongst other expressions. The most confused expressions were amongst Sad and Angry. This is understandable as they both have similar mouth movements. The Fear expression has performed the worst out of the lot. It is an expression that can vary based how the person reacts and interprets fear. It has been confused a lot with the Angry, Disgust and Happy expressions, that demonstrates how much the expression can change per subject. All the expressions apart from fear have one other expression where a lot of confusion occurs.

Table 4.4 - Confusion Matrix of the Mouth Facial Part

Actual Labels \ Predicted Label: Accuracy (%)	Angry	Disgust	Fear	Happy	Sad	Surprise
Angry	**72.9%**	5.4%	6.4%	0.5%	14.2%	0.5%
Disgust	3.9%	**68.2%**	12.7%	2.7%	3.9%	8.4%
Fear	11.0%	14.0%	**54.0%**	13.3%	6.4%	1.0%
Happy	0.6%	2.5%	7.9%	**87.9%**	0.0%	1.0%
Sad	17.5%	3.0%	5.1%	0.2%	**74.0%**	0.0%
Surprise	0.5%	4.5%	2.6%	1.2%	0.7%	**90.2%**

The confusion matrix for the eyebrows in Table 4.5 shows Fear also as the lowest recognised expression. However, the Sad expression has failed to be recognised as easily, demonstrating that the mouth has more changes occurred during the Sad expression than the eyebrows. Happy and Surprise have produced the highest accuracies, demonstrating that the change of the eyebrows form is significant enough to distinguish between other expressions. Both of those expressions cause the eyebrows to raise, which could be the movement that causes the high accuracy.

Table 4.5 - Confusion Matrix for the Eyebrows Facial Part

Actual Labels \ Predicted Label: Accuracy (%)	Angry	Disgust	Fear	Happy	Sad	Surprise
Angry	**39.3%**	33.1%	6.1%	5.8%	13.6%	2.1%
Disgust	25.1%	**49.3%**	7.6%	6.3%	10.5%	1.3%
Fear	11.8%	7.5%	**25.4%**	14.4%	17.9%	23.0%
Happy	4.0%	5.5%	7.9%	**63.0%**	8.7%	10.9%
Sad	19.8%	9.8%	18.4%	15.8%	**31.3%**	4.9%
Surprise	2.1%	1.2%	12.4%	18.0%	3.7%	**62.7%**

4.6.2.2 Combined Facial Parts Test

This section will investigate combining the efforts of the facial parts in a variety of combinations. The feature to represent each facial part will be the fusion of all the texture features. Each facial part will have a reference number, with Eyebrows as 1, Eyes as 2, Mouth as 3 and Nose as 4. The combinations will include testing the upper face, which is facial parts 1 and 2, and then the lower face 3 and 4. The best 3 features from the individual test will be combined, and finally, all of them.

The results for the facial Parts combination are presented in Table 4.6. The best performing parts combination was using the Eyes, Mouth and Nose, producing an accuracy of 80.30%. However, it was only 0.25% better than using all facial parts. Such a small amount can be produced by the randomness factor. What this really indicates is that the Eyebrows do not contribute any extra when using all the other facial parts. Using just the Eyes and the Eyebrows does increase the performance to 53.74% from 50.02%. The lower face combination also shows an improvement over the Mouth alone, increasing by 4.06%.

Table 4.6 - Facial Parts combination performances, using fused texture feature sets

Facial Parts Combination Eyebrows (1), Eyes (2), Mouth (3), Nose (4)	**Feature**	**SVM Poly Accuracy**
Upper Face (1+2)	**Fused Texture**	53.74%
Lower Face (3+4)	**Fused Texture**	78.60%
Best 3 (2+3+4)	**Fused Texture**	**80.30%**
All (1+2+3+4)	**Fused Texture**	80.05%

4.6.2.3 Experiment 4A Highlights

This experiment consisted of using the geometric information to map and crop out facial parts from the facial image. Each of the parts was subsequently tested using the same protocol (Protocol 1) from Chapter 3. The tests have demonstrated that reducing the facial image to individual facial parts does improve the performance for FER. When you compare the best performance based on the fused texture descriptors from the previous experiment in Chapter 3, of 78.7%, there is a 1.6% increase using the combinations of the parts. The increase is not big, but the main indication is that using parts of the face rather than the whole face performs better.

In terms of contribution, the Mouth has provided the most in these tests, and for every test, fusion of the texture descriptors also provides a boost to the performance. The joint effort of the Mouth, Eyes and Nose provide the best performance on 80.30%, an increase of 5.78% over the best individual facial part.

4.6.3 Experiment 4B – Introducing Deep Network Architectures for FER

The initial experiment using deep networks will be used on the whole face to determine the optimal network for the FER. This includes investigating the depth of the network and what parameter combination works best. The initial tests will include running a small, medium and large deep network on the face and all the facial parts using the same hyperparameter settings. Each test will use 10-fold cross-validation, with each fold training a network for up to 50 epochs until the network has converged.

Each cross-fold contains 80% training data, 10% validation data and 10% testing data. The hyperparameters are tuned as follows:

- 50 training epochs per cross-fold;
- Batch Size = 12;
- Learning rate = 0.0002 which slowly decreases to 0.00002 over 50 epochs;
- Momentum = 0.9;
- Weight Decay (regularisation) = 0.00001;
- Nesterov's Momentum Update [143] is used;
- Stochastic Gradient Descent is used as the learning optimiser;

With the following pre-processing techniques applied:

- Mean image is subtracted from all samples;
- Data is augmented to flip each image horizontally;
- Network weights are randomly initialised using Xavier's improved method [89].

Table 4.7 shows the average performance of the Face and each facial part using the small, medium and large network sizes. There are some interesting results produced, with the Mouth performing better than the Face, and Large networks performing better on the Test data in most cases, apart from the Mouth. In all cases, the Validation data has a better accuracy than the Testing data. The results on the Validation data has achieved a better accuracy using small-medium networks.

Table 4.7 - Average Recognition Rate for Deep Network performances of the Face and individual face parts using 10-fold cross-validation.

Face/Facial Part	Network Size	Average Validation Accuracy	Average Test Accuracy
Face	**Small**	73.81%	66.38%
	Medium	**76.73%**	68.05%
	Large	74.86%	**71.80%**
Eyes	**Small**	**49.44%**	40.00%
	Medium	46.18%	38.33%
	Large	45.48%	**44.44%**
Mouth	**Small**	72.91%	71.25%
	Medium	**75.32%**	**73.05%**
	Large	73.95%	70.55%
Nose	**Small**	**49.37%**	**43.88%**
	Medium	47.56%	42.63%
	Large	47.70%	40.83%
Eyebrows	**Small**	47.43%	38.05%
	Medium	49.02%	37.08%
	Large	**50.48%**	**40.00%**

Figure 4.10 displays the progress of how the system performs on the Validation and Testing data during the training stage. Based on this information, there is a visible trend that the accuracy varies a lot every epoch, which can make it difficult to understand exactly what size network is better. Figure 4.11 is a plot on how the objective function for the Training and Validation data changes after each epoch. It is based on the same test in Figure 4.10. From both plots, there is an indication that beyond ~10 epochs the network starts to overfit the Training data, as there is not much improvement on the Validation and Testing data. There is also signs of instability fluctuating rapidly between 65-75% accuracy. These symptoms are a result of not having enough data, where the networks cannot learn any more useful information that can benefit any new test samples.

The combination of facial parts can also be applied using a multi-branch approach. This will incorporate all the facial parts to initially be learnt individually. After a few convolutions, the feature maps for each the facial parts are merged through multiple concatenations and the network continues as a single branch. Figure 4.12 shows the desired architecture to allow the contribution of each facial part within the system. The medium size layout is used, which involves 2 convolution stages before every max pooling stage.

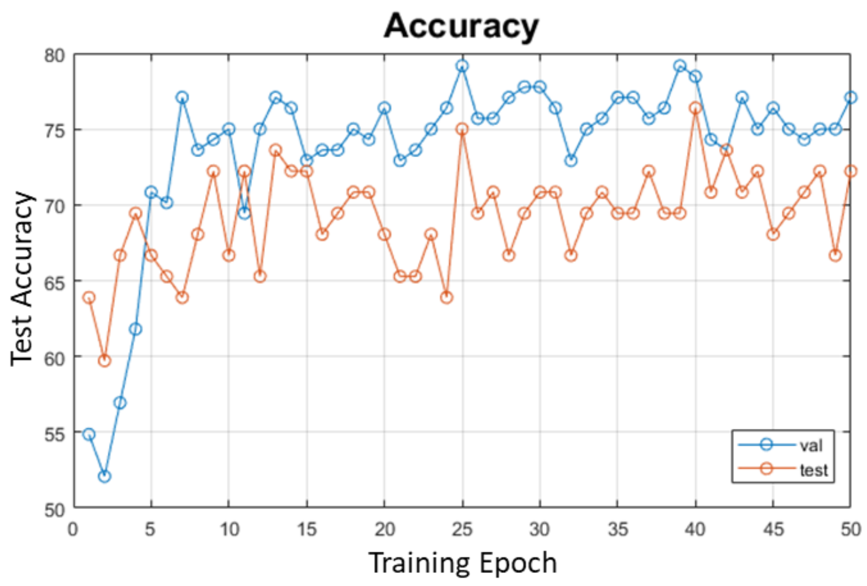


Figure 4.10 - Example of the Training process for the Medium size Deep Network on the Mouth. The Validation and Test accuracies are plotted throughout the 50 epochs of training.

The desired outcome of this test is to see if the features of the facial parts can be integrated and learnt together to improve the performance over using an individual facial part. There are a few decisions needed to make the architecture, with the main being when to join the branches together to share their feature maps. Another is how to concatenate the feature maps, should they be joint in the spatial dimension or should they be stacked on top of each other. And finally, how long to extend the joint branch after concatenation. We have opted to make sure that each facial part has enough convolutions before concatenation so that the quality of features for each facial part is good enough as they will be learnt better.

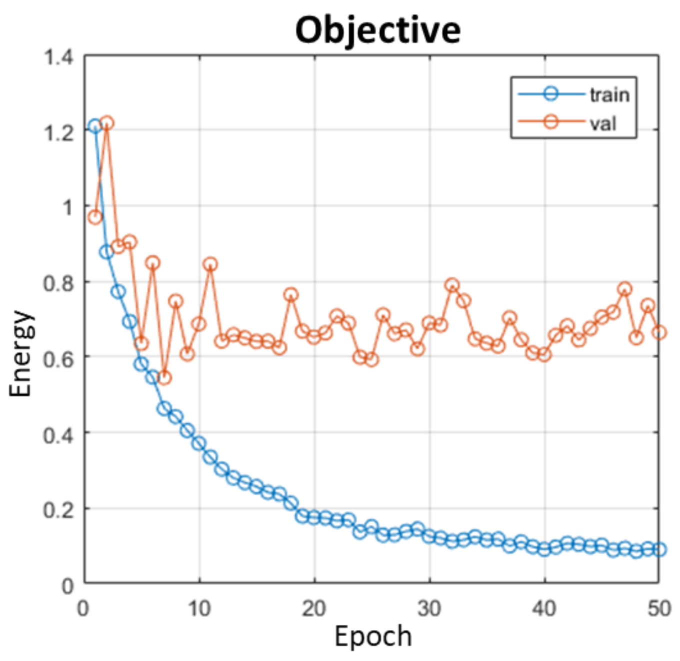


Figure 4.11 – The objective function of the training process for the Medium size Deep Network of the Mouth.

For the concatenation layer, each of the feature maps for each facial part has been joint spatially. This was decided as the facial parts have different characteristics, and they may not be compatible with each other if they are stacked. Joining them spatially allows the chance for further convolutions to find any characteristics that can be common amongst all facial parts that can be represented in future feature maps. To allow these characteristics to be found, there is an extra convolution sequence (Conv 5-1) applied straight after the concatenation, which is followed be one last convolution sequence (Conv 6) before all the neurons are fully connected.

Table 4.8 - Performance of all facial parts combined using the Joint Architecture that learns each facial part individually and jointly throughout. The test is based on 10-fold cross-validation which is run once. Results of the previous test on the face and the best performing individual facial part is included for comparison, both using the same protocol.

Feature	**Average Validation Accuracy**	**Average Test Accuracy**
All Facial Parts	**83.33%**	**77.63%**
Best Individual Part (Mouth)	75.32%	73.05%
Face	74.86%	71.80%

The parameter size for this architecture is significantly larger than the other proposed architectures. However, this accomplishes more as a network using all facial parts than using them individually. It makes the whole system framework more efficient by training only one deep network that takes care of all the facial parts and their fusion.

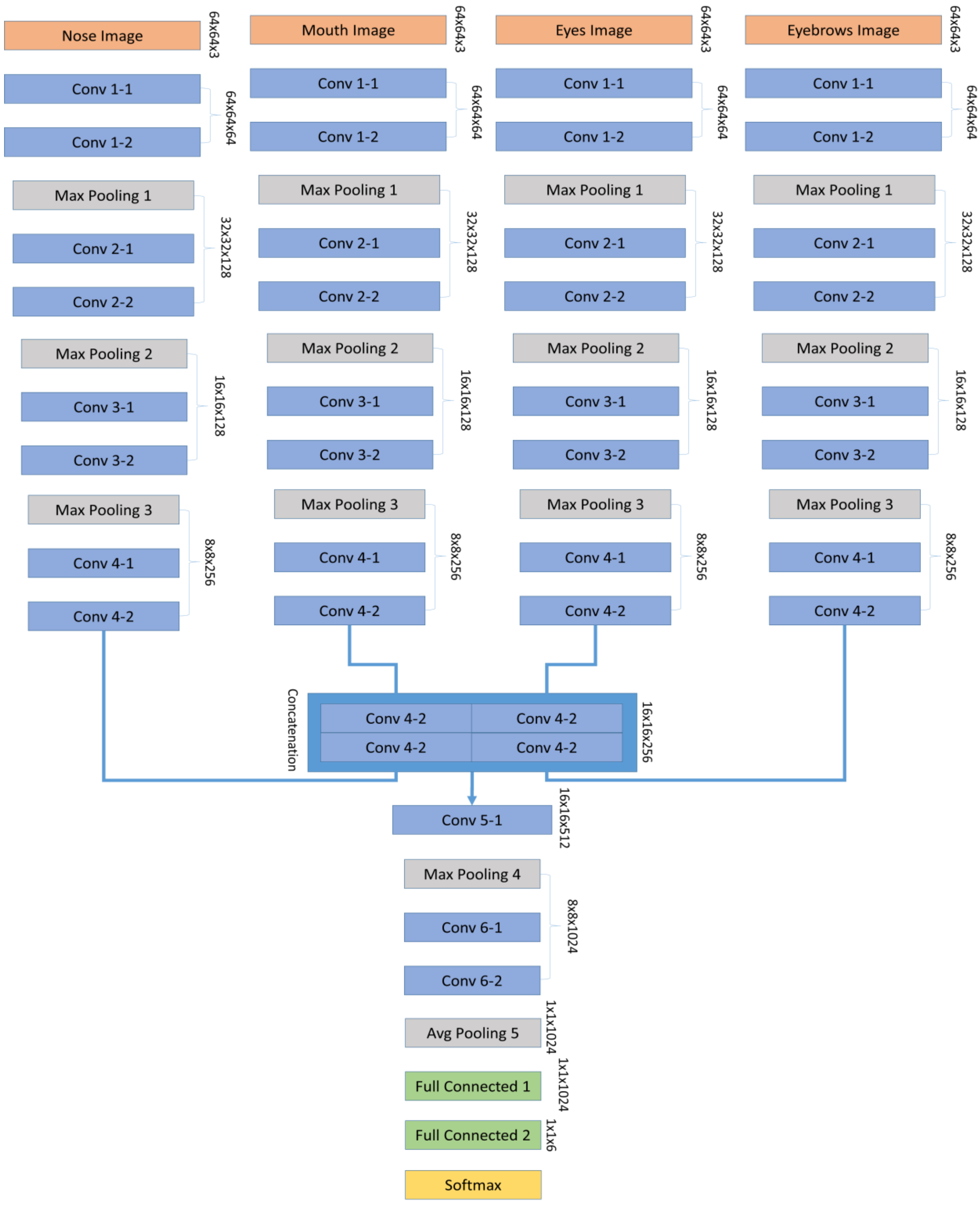


Figure 4.12 - Joint Architecture of Deep Network containing all 4 facial parts, starting with 4 individual branches that are spatially concatenated together in the later stage of the network. The following convolutions connect and share parameters of each facial part.

The tests on combining all facial parts using the Joint Architecture showed great performance in Table 4.8. The results show a superior increase in validation accuracy against using the whole face or the individual Mouth facial part. The average test accuracy also has a significant improvement of 5.83% over the whole face and 4.58% improvement over the individual mouth facial part.

4.6.3.1 Experiment 4B Highlights

Experiment 4B consisted of thorough tests to introduce deep learning in the form of CNNs for FER. This was applied to the whole face and the facial parts that were tested individually and combined. Several architectures were proposed consisting of different depths to investigate if overfitting occurs, and if there is a performance difference between them. A joint architecture was also proposed that strategically combined information from each facial part by individually learning them and then sharing their parameters through a concatenation layer.

A few outcomes are understood from this experiment, which is summarised as the following:

- The network architecture sizes do not contribute much towards a difference in performance, using this database. This is most likely due to the lack of training data to learn from, that can even cause smaller network architectures to overfit.
- The Training data is learnt very quickly, with the network converging in a matter of a few epochs. This is also the probable cause of not having enough data. Batch normalisation also speeds up the convergence of the network, contributing to overfitting the data.
- The Mouth facial part performed better than the whole face, which did not occur in the previous experiment. This may be due to the spatial increase in its size when tested individually, paired with the fact that the mouth is detected as the most expressive facial part.
- There is a significant increase in performance using the Joint Architecture compared to the single branch architecture used for the face. This demonstrates that multiple spatial entities can be included in a single architecture. The facial parts combined once again demonstrates a solid improvement, indicating that the face does contain noise that should be avoided.

4.6.4 Experiment 4C – Facial Expression Recognition using Pre-Trained Networks

This experiment will be split into two sections, with the first focusing on using existing pre-trained networks as a feature extraction tool, in which the Protocol and framework will follow Experiment 4A. The texture features are essentially replaced with the extracted Deep features. The second section will focus on pre-training a network based on the BU-4DFE database samples, and then applying a fine-tuning process with the BU-3DFE samples.

4.6.4.1 Pre-Trained Feature Extraction Test

This section will test the VGG-Face pre-trained network layers FC1 and FC2, along with the equivalent for AlexNet to compare their performances. The full face is tested along with each facial part and their combined efforts. Each of the following tests is run 100 times with SVM Poly as the machine learning technique. PCA is also applied to reduce the dimensionality, keeping 99% variance. Each face is pre-processed according to the requirements by the VGGFace and AlexNet networks. For VGGFace, this includes resizing the images to a spatial dimension of 227x227 and subtracting the provided mean for each colour channel. For AlexNet, the images are resized to 224x224 spatial dimensions, and a provided pixel level average image is subtracted from each image.

Table 4.9 - Performance of extracted Pre-trained features from the VGGFace and AlexNet networks. Two fully connected layers are extracted from each network, with SVM Poly used as the machine learning technique.

Image	**VGGFace FC1**	**VGGFace FC2**	**AlexNet FC1**	**AlexNet FC2**
Face	69.01%	63.71%	77.74%	76.22%
Eyebrows	38.95%	34.97%	43.14%	42.48%
Eyes	44.16%	36.74%	47.63%	45.59%
Mouth	71.79%	66.17%	76.71%	75.63%
Nose	47.78%	37.74%	47.66%	46.65%
All Parts	**76.98%**	**73.55%**	**80.86%**	**80.51%**
Best 3 Parts	76.18%	73.38%	79.70%	79.47%

Table 4.9 shows a few interesting outcomes from the experiment. Firstly, the AlexNet features have performed far better (80.86%) than the VGGFace features (76.98%). This is an unexpected outcome as the training for VGGFace is more relevant. Secondly, the initial fully connected layer FC1 for both networks have performed better in every test. The third outcome to take is that the parts combined perform significantly better than using the whole face for VGGFace (~8%), and a slightly smaller jump with AlexNet (~3.1%). In-fact, for VGGFace, the Mouth facial part alone performed better than the whole face. The performances of the facial parts follow the trend from the previous test in Table 4.3, with Eyebrows being the weakest, followed by Eyes, Nose then Mouth. However, for both pre-trained networks, using all the facial parts combined has produced the best accuracies.

4.6.4.2 BU-4DFE Fine-tuning Test

For this test, a deep network based on the architecture in Figure 4.12 is trained using samples from the BU-4DFE database. The BU-4DFE database contained visual sequences of the 6 basic expressions, where the facial expression starts in the neutral state and then transforms quickly into the assigned expression which then gradually reverts to the neutral face. Based on the observation of random samples, the peak intensity is determined to be portrayed after roughly a third of the duration time. A static subset is then created taking the frame of every sample sequence that occurs at that time.

Using this new subset of images, the same techniques for facial parts extraction are used to extract all the facial parts from each face. These parts are used together during training, with the pre-processing steps of taking the mean image value applied. The network is trained once using 90% of the data, with the remaining 10% used as validation. There are 200 full epochs of training with all the parameters saved and ready for the fine-tuning process.

The number of expressions and the training objective is the same for the BU-3DFE database. Therefore, there is no modification needed to the architecture, and the BU-3DFE samples can immediately start training 10-fold Cross-validation on the pre-trained network. Majority of the settings are set the same as Experiment 4B using the combined facial parts. The only differences are:

- Training is applied for 50 Epochs, with the weights are already pre-trained for the desired task.
- As a result of pre-training, the learning rate is significantly lower to prevent drastic changes to the already tuned parameters. The learning rate starts at 0.0000002 and decreases slowly.
- The mean value subtracted from the images of the fine-tuning task is based on the BU-3DFE samples. This is because the pre-training task is the same as the fine-tuning task. Therefore, it is better to subtract the mean of a new batch of images on those samples because if there is any lighting difference, then the mean value for the BU-4DFE database may alter the new samples more than it should.

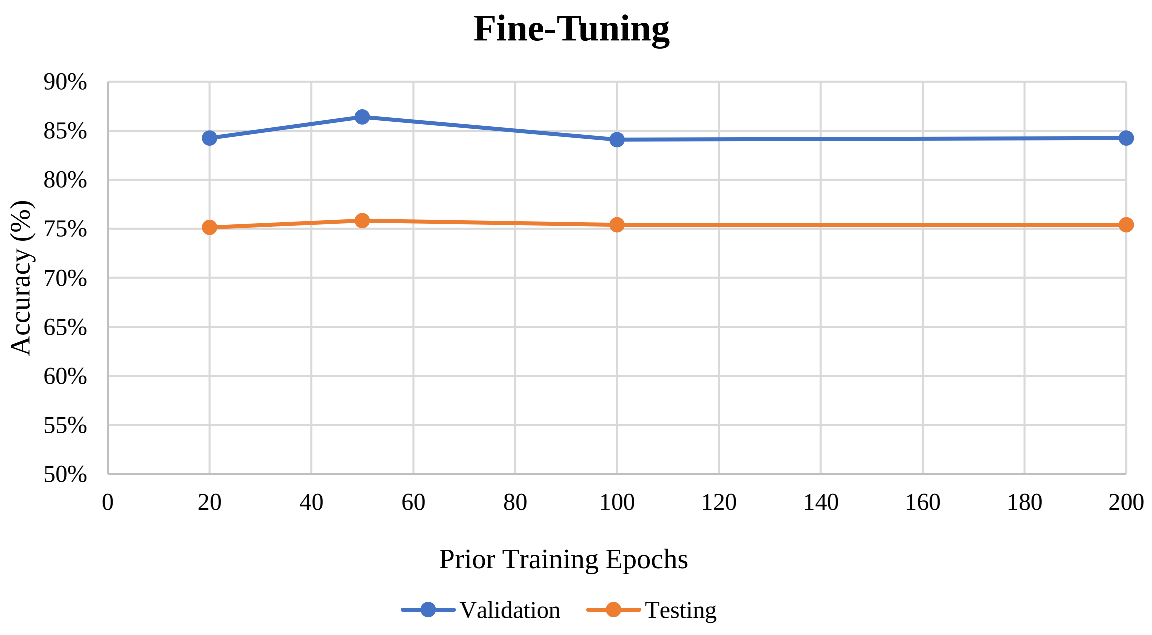


Figure 4.13 - Fine-Tuning performance for FER using a pre-trained network on the BU-4DFE database

The rest of the protocol is set the same as in Experiment 4B, for combining facial parts. This is to allow a fair comparison. This experiment includes only one test using 10-fold cross-validation, as 100 tests will require training 100 networks using 10-fold cross-validation which will be highly impractical.

Figure 4.13 shows the performance of the fine-tuning test. The best accuracy is **75.83%**, which is not better than training all the facial parts from scratch. However, it still performs better than using the face alone. An improvement over training from scratch is that the networks hit high accuracies almost immediately when retraining, showing that the pre-training has made an impact.

4.6.4.3 Experiment 4C Highlights

This experiment had 2 main tasks included, first using pre-trained networks as feature extractors for the whole face and the individual facial parts. The second task was to fine-tune a pre-trained network to see if retraining on the new data can improve the performance. Both tasks have shown promising improvements over the work in Chapter 3 through the process of using facial parts and creating applying deep learning technology.

There were a lot of interesting findings to take from the first task. Each will be broken down and understood as to why:

- AlexNet VS. VGGFace: The results indicated that the AlexNet pre-trained extracted features performed a lot better than those from VGGFace. This is interesting because the AlexNet network is trained on objects rather than faces. There are differences between the networks that can justify the performance difference:
    - VGGFace is trained specifically to identify a face, essentially from one it learnt during the training process. This means that for the given facial expression samples of the same subject, the network may just consider them to be the same person and consequently producing a similar feature for each expression. This can have a negative impact as it may consider all expressions to be the same. There will also be a lot of variation in the produced feature for an angry expression of a male subject VS. a female subject.
    - AlexNet is trained on more data, and it is trained on detecting objects in images. This may give an advantage for facial expressions as the differences between the expressions are related to how the facial parts change. With objects trained being a variety of shapes, sizes, and angles, this may aid the process of highlighting these changes across the facial structure. This may be considered a better approach to distinguish changes of the face when compared to the VGGFace network, that looks at the whole face.
- The mouth facial part produced better performance than the whole face for VGGFace: This result backs up the point made about VGGFace producing similar features for the different

expressions of the same subject. The mouth has performed better alone because it has been separated from the whole face, making it harder to identify if it is the same subject. However, the performance increase is not substantial enough to completely overcome the problem. It still requires retraining the network to the facial parts, which would help towards separating facial parts from the same subject.

  - The face has performed better than the mouth when using AlexNet, which may be because it doesn't treat the image as a face. The case may be that multiple filters respond as it may detect more variety of objects in the face than the mouth, producing a more comprehensive feature vector than the mouth.

- Facial parts combined performed a lot better than using the whole face: The idea of using facial parts instead of the whole face has proven again to be a better option. This indicates that there is a lot of noise in the face and taking the key parts of the face can contribute better towards FER. To humans, the mouth is visually the biggest indicator in the face of how the person is feeling. This has also shown to be true for artificial intelligence, meaning that there is a higher variance in the mouth structure across expressions than other parts.
- The FC1 Layers have produced better performances than the FC2 Layers: This can be expected because they are both fully connected, except that there is a ReLU layer in between them. What this means is that all the negative magnitude responses from the FC1 layer will essentially be killed to 0. This could result in significant loss of valuable information when used as a pre-trained feature extractor.

The second task involved an attempt to pre-train a network based on similar data, using the BU-4DFE database. This was later fine-tuned with the BU-3DFE database to investigate if prior training will improve the training process with the new data. The results indicated that there was not an improvement in the test accuracy, but there was on the validation accuracy. The quality of the network parameters after pre-training are significantly improved over random initialisation, with minimal effort required for convergence. The pre-training duration of 200 epochs did not provide any benefits, as the 50 iterations of pre-training provided the best fine-tuning performance. This is probably caused by the lack of training samples by both BU-3DFE and BU-4DFE databases.

4.6.5 Experiment 4D – Joint Bayesian for Similarity

This experiment takes a different approach to the others, by trying to incorporate a Joint Bayesian approach that includes learning the Mahalanobis distance of expressions to find the closest match. It is based on exploiting an advanced technique proposed by Chen et al. [140] for face verification, into a multi-class application of FER. The idea also investigates how the CNN features develop throughout the training process. The fully connected layer of the CNN is extracted from every sample after every training epoch and split into a training set and testing set. These sets contain the same samples used for training, validating and testing the CNN.

Table 4.10 - Results of applying the Joint Bayesian technique on the face and facial parts.

Feature	Average Validation Accuracy	Average Test Accuracy	Improvement in Test Accuracy over Deep Learning
Face	76.04%	70.41%	2.36%
Eyebrows	47.77%	39.72%	**2.63%**
Eyes	43.68%	40.13%	1.80%
Mouth	73.95%	75.41%	2.36%
Nose	45.97%	43.05%	0.41%
All Parts	**80.69%**	**78.33%**	0.69%

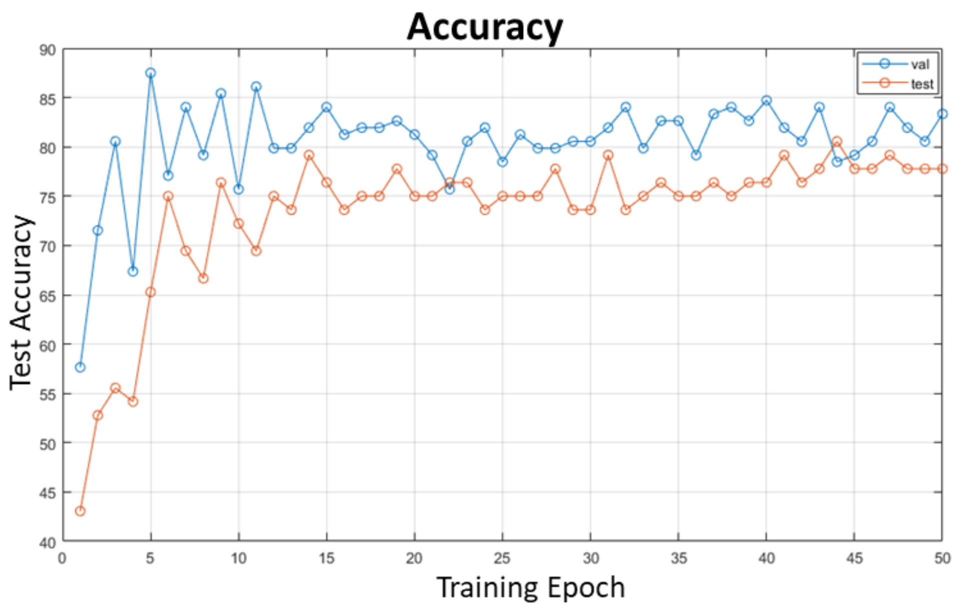


Figure 4.14 - Plot showing the performance of the Joint Architecture network learning all facial parts. Accuracy is based on the CNN performance, from which the Test and Validation accuracy is measured at every epoch.

The extracted layer from every trained network is the Fully Connected 1 layer. This layer contains the interconnectivity of all the neurons from the previous layer, which includes the combination of each facial part information using the Joint Architecture. Each of the tests is run in parallel to the tests in Experiment 4B, which allowed for exact comparison using the same initialisation of parameters and randomisation in training and testing. The upcoming results are based on the medium sized network architecture, which includes tests on the Face, on each individual facial part and on the combined parts.

The average validation and test accuracies are recorded from the 10-fold cross-validation, along with the performance difference compared to the CNN.

Table 4.10 contains the results achieved by using Joint Bayesian on the face and facial parts. The Combined architecture performed best yet again, followed by the Mouth, then the Face. In all tests, there was some increase in the test accuracy, with the best improvement of 2.63% on the Eyebrows. The Mouth had a significant increase of 2.36%, but the parts combined only went up by 0.69%, reaching an overall highest of 78.33% using the protocol for training a CNN.

Figure 4.14 is a plot based on the performance throughout the training stage of the CNN. The results after of the Validation and Test data that is propagated and predicted through the network after each epoch is trained. The accuracy of both Validation and Test data starts off low and increases to the hit its steady peak after ~15 epochs. The test performance slowly improves and becomes more stable after 40 epochs.

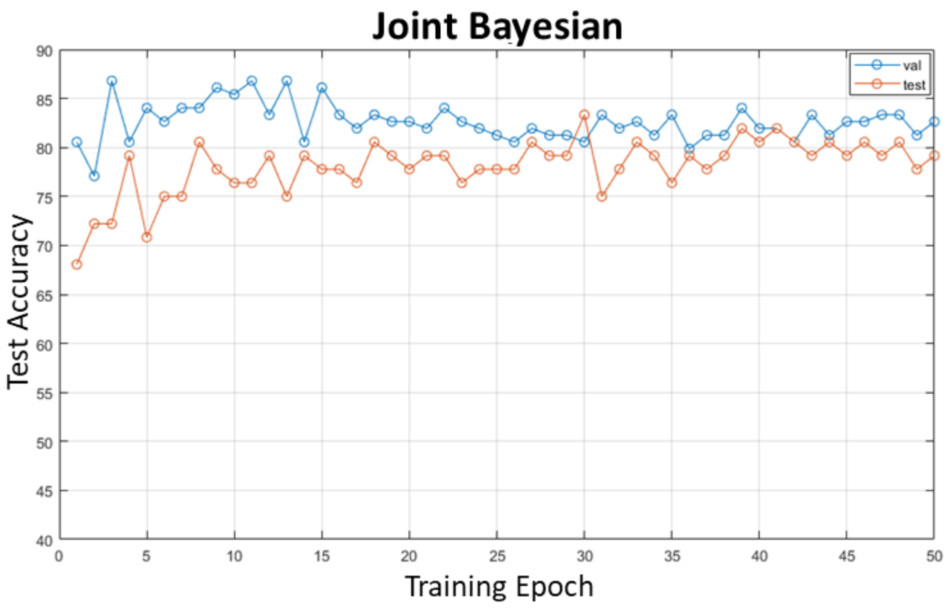


Figure 4.15 - Plot showing the performance of the Joint Architecture network learning all facial parts. Accuracy is based on the Joint Bayesian technique that is applied using the fully connected features. The Test and Validation accuracy is measured of the feature extracted at every epoch.

Figure 4.15 shows the equivalent plot based on the Joint Bayesian performance for the Validation and Test data. There is a noticeable difference in how the starting epochs perform. Joint Bayesian immediately jumps above 65% for the test accuracy and above 80% for validation. The performance does stabilise the more the CNN is trained, with the test accuracy fluctuating between 78% and 82%. Compared to Joint Bayesian, the CNN performance is slower to hit its peak performance, with the Test data taking 6 epochs. The performance of both techniques is similar towards the end of the training.

4.7 Evaluation and Discussion

The general outcome of the experiments has proven that it is beneficial to use cropped facial parts instead of the whole face. In all experiments, using the proposed facial parts produced a better result, and in some cases, the mouth part alone performed better than the whole face. The introduction of deep learning has provided a better understanding of facial parts. Better results were achieved than using the hand-crafted techniques in Experiment 4A. Different avenues were explored to give a thorough insight into how deep learning can be adopted for FER.

Experiment 4A extracted the facial parts and applied the same framework adopted in Chapter 3, using the textured features only. This was applied to each individual facial part, and then combined using feature fusion. The findings demonstrated an improvement in performance of 1.6% over the equivalent feature for the whole face, that was recorded in Chapter 3 using Protocol 1. This was achieved when the facial parts were combined at the feature level.

Table 4.11 - Performance of all techniques, where the bold font results represent the highest accuracy in each protocol.

Performance Comparison				
Feature	**Domain**	**Protocol 1**	**Protocol 2**	**Protocol 3**
Wang et al. [10]	Geometric	61.7%	83.6%	-
Rabiu et al. [105]	**Geometric**	-	**92.2%**	**-**
Soyel et al. [106]	Geometric	67.5%	91.3%	-
Xioli et al. [107]	Geometric	-	90.2%	-
Tekguc et al. [108]	Geometric	-	88.1%	-
T. Yun et al. [110]	Texture	-	85.3%	-
Lemaire et al. [111]	Texture	75.7%	78.1%	-
Yurtkan et al. [109]	Geometric	-	88.2%	-
Tang et al. [112]	Geometric	75.7%	87.1%	-
Zhen et al. [113]	Geometric	84.5%	-	-
Yang et al. [114]	Geometric	84.8%	-	-
Li et al. [115]	**Both**	**86.3%**	-	-
Chapter 3 [117]	Both	79.2%	90.2%	**-**
Fine-Tuning	Facial Parts	-	-	75.8%
CNN	Facial Parts	-	-	77.6%
Joint Bayesian	**Facial Parts**	**-**	**-**	**78.3%**
Hand-Crafted	Facial Parts	80.3%	-	-
Pre-Trained Feature	Facial Parts	80.8%	**-**	**-**

Experiment 4B introduced the deep learning techniques into the mix. The initial tests involved different deep network architectures designed and tested on the whole face and the facial parts. This was followed by an elaborate deep network design to incorporate all of the facial parts within the same network. This was achieved by initially creating multiple branches that learnt each facial part separately, which were all combined in the later convolutions using concatenation layer. This allowed each facial part to have their own unique and shared parameters. For the individual tests, there were three network architectures designed which were inspired by the VGGFace network. The idea was to understand if having different depths of convolutional neural networks training on a small size database provided any benefits. Based on this objective, small, medium and large deep network architectures were designed and tested on the face and facial parts.

All of the tests were conducted using a very similar protocol to Experiment 4A, with the exception of running the test 1 time instead of 100 times, as 100 tests are impractical in time and resource consumption. The results from the individual tests showed a similar trend in terms of facial parts. But interestingly, the mouth facial part had performed better than the whole face by 1.25% on the Testing data. The combination of facial part using the Joint Architecture produced the best result of 77.63%, which is 5.83% better than using the face. In terms of the small, medium and large depth of network architecture, it seemed like there was no major impact by any. The best result was produced using the medium-sized architecture and was therefore adopted in the Joint Architecture. This outcome is likely to be caused by the short supply of data samples.

In Experiment 4C, the idea of using pre-trained networks as a feature extractor was explored along with pre-training a network using the BU-4DFE network, and applying a fine-tuning process using the BU-3DFE database. The pre-trained feature extractor tests induced some strange results which suggested that a pre-trained network of objects for object detection performs better than a pre-trained network on faces for face recognition. The conclusion of the results was that the VGGFace network performed badly because it was trained specifically to learn similar faces. This can be problematic in the case when different expressions are produced from the same face, as the pre-trained network will respond similarly to the expressions.

The pre-training + fine-tuning test was an attempt to see if prior training on the same task using pretty much the same form of data would provide a boost to the task. This was an attempt to extend the idea of a pre-trained network by retraining them specifically for facial expression recognition. An advantage of using existing networks is that there is already a set of trained parameters provided that are learnt from other data sources. For the experiment, the BU-4DFE database was used for pre-training a network that used all the facial parts with the Joint Architecture in Figure 4.12.

Unfortunately, there was no improvement in terms of test accuracy. However, in other areas there were improvements such as the shorter time required for convergence. Along with the improvement in validation accuracy and compatibility of trained parameters. One area in which the pre-training stage could be further improved is to use a larger data source, as there were only 101 subjects with 6 expressions from each that were utilised.

Experiment 4D introduced a Bayesian concept for attempting to create a metric distance that is derived from the deep feature representation of each expression. It was an attempt to understand a probabilistic view of how an expression is closely related to another. The Mahalanobis distance was adopted to separate deep representations extracted from the fully connected layer of each trained network, for each facial expression.

The experiment produced an improvement over the CNN networks trained in Experiment 4B, increasing by up to 2.63% on the individual networks, and 0.69% on the combined facial parts test. Another finding was that the performance hit high accuracies almost immediately when the CNN started training. This benefit can possibly be adopted in networks that take a much longer time to converge, as an early insight into the potential of the network.

4.8 Summary

This chapter involved investigating 3 main areas of innovation for facial expression recognition. The first is to use the geometric information to identify and separate facial parts, in order to capture the most contributing factors of the face for FER. The second is to investigate and apply a variety of deep learning techniques on these facial parts, and the face for comparison. And the third is to apply a Bayesian approach to find similar and dissimilar expressive facial parts across facial expressions, integrated into a framework that uses deep learning. The outcome from all experiments can be understood as follows:

- Using facial parts provides a significant boost over using the whole face. The Mouth area contributes towards the majority of the performance but improves when combined with the Nose, Eyes and Eyebrows.
- Using popular well trained Pre-Trained networks as feature extractors can provide high-quality features, even if it is based on object detection.
- The Joint Bayesian with Metric learning adaptation for FER has shown improvement over using a CNN alone. There is room for improvement for Joint Bayesian + Metric learning to be further integrated into the learning process.
- Based on the outcomes of the experiments, using very deep networks performs similarly to shallower deep networks when training on a small database. This portrays that there is overfitting occurring in the deeper networks, and that it is unnecessary as a shallower network can achieve the same performance.

Chapter 5. Facial Motion Dynamics Modelling

In the previous two chapters, the focus had been in using static data for facial expression recognition. This can be considered a limiting factor for facial expressions as the build up to each expression is not considered. Dynamic data is noticeably growing to be an emerging solution for HCI applications because of its rich information. Motion information that occurs on the face whilst a facial expression is in action can be considered a valuable tool for FER. In order to produce a better system for capturing emotions, the motion dynamics in a video recording data is investigated.

In this chapter, motion dynamics are studied to understand how this technology can be used for video-based action recognition. Research in this area spans across all sorts of applications, from body gesture recognition to surveillance-based applications for object or human identification. This chapter focuses specifically on how human interaction can be observed and processed using various motion-based feature extractors that can model and capture its dynamics. In order to create an effective motion descriptor, it is designed and tested for applications that contain large amounts of movement in the dynamic data, such as gesture recognition and human action recognition. This is paired with machine learning techniques to understand the dynamic model and predict similar behaviour.

There are two parts to this chapter. The first part is to develop a set of motion descriptors based on extending MHH descriptor using dynamic data that contains significant movement. These features are generated by evaluating motion patterns at the pixel level within a temporal sequence. The second part is to investigate how these descriptors can then be applied towards human emotions, and how well it performs.

5.1 Introduction

In image processing and understanding, the use of dynamic content has becoming increasingly common to help solve complex problems because of its extensive information pool. Videos are a common source of dynamic content providing temporal data, and video dynamic descriptors extracted from the frames to provide an efficient description of the occurring dynamic actions. This video descriptor can be helpful for automatic systems to understand and analyse the video content. It can be further applied in the applications of human-computer interaction, robotics and automatic multimedia content analysis for complex big data.

In computer vision, visual recognition on temporal and spatiotemporal domains using appearance-based methods has a very active area of research. A key component of temporal data is Motion. Motion can represent a lot of information about an entity, whether it is an individual or group of humans, animals or objects. It is the fundamental description of an action represented by a sequence of data.

Majority of the early research had been based on still images, which at the time was possible to apply image processing techniques at a good speed. But with still images, for certain applications, the real full picture sometimes cannot be seen. Videos, a form of temporal data, gives the possibility to view what exactly is happening in an event from start to finish. The depth and quality of the information surpasses still images, but at the cost of computation and storage. The amount of resources required to constantly process video data is a lot compared to a still image. A lot of research is ongoing to try and understand this data and seek information only relevant to the task.

Motion that can be captured is used to summarise a video into a smaller manageable subset of data, by looking for a unique signature that can be related to actions and movements. This motion can be used recognising human actions [144]; detecting unnatural activities through surveillance videos for that environment [145] and other human-computer interaction tasks. Some techniques look at the frame level and observe all the changes from one frame to the next. These are appearance-based methods, where motion can be extracted from the pixel values of the frames.

Optical flow is another technique that looks at the distribution of motion [146] by observing the velocities of objects between the frames of a video. This has been used in applications for action recognition [147], with recent works in semantic segmentation [148], [149] and also in dynamic facial expression recognition [150]. There are different algorithms developed to determine the optical flow, which include the differential techniques of estimating the flow as the Buxton–Buxton method; the Lucas–Kanade method; the Horn–Schunck method; Phase correlation, as well as block-based and discrete optimisation methods. Some of these are compared and tested thoroughly by Baker et al. [151] They concluded that although some methods demonstrate good performance on certain tasks, there is not any that has shown significant improvement across a range of different tasks.

This chapter applies appearance-based techniques, namely MHH with a few proposed extensions for it, for applications that include human action recognition, gesture recognition and Depression analysis. The variety of applications can demonstrate how robust and well the motion descriptors can interpret input sequences that are based on various content, which also have different forms of motion.

5.2 Related Works using Temporal Techniques

For appearance-based methods, there has been a continuous effort in the computer vision society. Early works have been on methods like Motion Energy Images (MEI) and Motion History Image (MHI) to capture the movement of a motion. Bobick and Davis [152] created MEI to construct temporal templates with a view specific representation of motion over time. The temporal templates are constructed by vector images that can be matched stored representations of known actions. A sequence would sweep out a particular region of the image where the motion is, which can be used to suggest the current action and the viewing angle. MHI [152] is used to represent the direction of motion in an image sequence.

Each pixel has the temporal history of its motion displayed by its assigned intensity value. All the pixels with a scalar value can be represented as an image of recent motion. The feature itself lacks the depth information as it is solely based on the energy differences between two frames at a time. For a visual sequence based on an action, it would create an equivalent binary visual sequence based on the energy and decay of motion occurred.

A more effective approach would be to summarize the energy across a whole sequence as a single vector, as some actions require longer sequences before they can be recognized, such as a jump that would require the subject running beforehand. An attempt to address this issue was followed by Weinland et al. [153] as they extended the 2D motion templates described by Bobick and Davis to create Motion History Volumes. They claim that a 3D representation of motion templates is a more robust and natural way to fuse information from multiple images, producing view-invariant features. To do this, they use the occupancy function $D(x;\ y;\ z;\ t)$, considering voxels instead of pixels, expressing motion in a cylindrical coordinate system. With the captured motion, they apply a Fast Fourier Transform on the cylindrical coordinates to produce Fourier magnitudes which they represent as their feature.

There are also techniques that look for patterns and edges within the temporal space. LBP-TOP is a method that works well, which is based on the popular spatial method Local Binary Patterns [72]. A variant on LBP-TOP has been developed recently by Almaev and Valstar [86] called Local Gabor Binary Patterns - Three Orthogonal Planes, for the purpose of Automatic Facial Expression Recognition. They extended the spatial feature extractor method Local Gabor Binary Patterns produced by Senechal et al. [154], which applies multiple Gabor filters to get the magnitude response of the images, and then an LBP operator is applied on top of those images. LGBP-TOP combines the ideas from LBP-TOP and LGBP using the three orthogonal planes XY, XT and YT, to split the temporal data into blocks that are processed through a bank of Gabor filters. After this, the LBP operator is applied.

Local Phase Quantization - Three Orthogonal Planes is another descriptor for temporal data, similar to LBP-TOP, by capturing LPQ features across the XY, XT and YT dimensions. This is developed by Jiang et al. [32] in which they applied it to facial expression sequences for detecting Action Units. LPQ captures local phase information by obtaining the coefficients produced by using a short-term Fourier transform across a grid of $M \times M$ blocks [78]. LPQ-TOP uses LPQ across the three planes and concatenates the resulting histograms to produce the temporal feature.

Histogram of Oriented Gradients 3D [155] created by Kläser, Marszałek and Schmid is a descriptor based on histograms of oriented spatiotemporal gradients. Based on HOG [65], [75], they extend HOG to calculate the gradients and their orientations across the X,Y and time dimensions of a visual sequence. LBP-TOP, LPQ-TOP, LGBP-TOP and HOG3D are recent methods that look for patterns using traditional hand-crafted techniques that are designed for spatial applications. These techniques are being adapted into the temporal domain using a straightforward method. This idea lacks the naturalistic concept for temporal motion depiction as it is primarily based on spatial applications. This can then propagate inefficiencies such as computational speed, especially when you consider the complex calculations required for HOG or LPQ on a single frame. Motion Binary Pattern [29] attempts to capture the characteristics of motion movement by observing the frame level gray values of the video. Each of the frames is divided into cells. Then 3 cells across 3 frames in a fixed XY position are taken and the first frame cell is compared to the second frame cell. For each pixel in the cells, if the gray value is higher, then a 1 is assigned, if not, than a 0. The third frame cell is then compared to the second frame cell in the same way. The output from both comparisons are combined using an exclusive XOR function which then creates the final motion pattern [29]. Unlike the previous methods, MBP tries to capture natural occurring motion. However, it lacks the depth of time information as each frame overwrites its previous using the XOR function. To solve this issue, Motion History Histogram was developed by Meng et al. [30] that records the motion history.

5.3 Motion History Histogram & Extensions

In this section, Motion History Histogram is firstly explained and understood in detail, with the use-case of gesture recognition. Secondly, it is modified to produce multiple extensions that enhance the basic principle of MHH. This includes the addition of other adjustable dimensions such as time and spatial area. A discussion is made based on the merits and downfalls of each extension. During the discussion, the visual samples provided will be based on the KTH dataset [144] for the demonstration of each extension. This dataset is based on human actions and is chosen because there is a lot of visual movement produced by the actions.

5.3.1 The Motion History Histogram Descriptor

As previously detailed in Chapter 2.2.3.2, MHH is a descriptor that captures the patterns of motion across a visual sequence. In this section, there will be an in-depth review of the MHH descriptor, explaining how it works with all the technical details. There will also be a discussion on the drawbacks of the descriptor, and how it can be improved.

The MHH descriptor looks at temporal data to produce motion that has visually occurred in a sequence of frames. This means that there are no motion sensor data required, which is a major benefit as video data is far more available and accessible to consumers than data from motion sensors. The visual motion can be produced by detecting the change of intensity in each pixel as the entity moves within the frame via a sequence of frames. From this, binary patterns are noted that occur in each pixel depending on the duration of the change occurrences. In simple terms, the length of the pattern P_m determines how long a sequence of motion must occur before the histogram in the MHH feature; for pattern P_m of length m; has its pixel bin incremented. And for motion to be detected, the absolute difference value of pixel intensities between two frames must be greater or equal to a threshold value.

The MHH feature can be visually seen in Figure 5.1, containing the motion patterns of a person waving their hands up and down. Patterns up to $M = 5$ are recorded, with each bigger pattern showing less and less motion occurring for longer periods. This is because the action has motion occurring for shorter intervals in a fixed spatial location. But in fact, the motion occurs continuously up and down, which is why there are streaks of the subjects' hand that makes a circle shape. Hence the word "History" is included, and it records the history of the object/subjects' path, including the overlaps in spatial location.

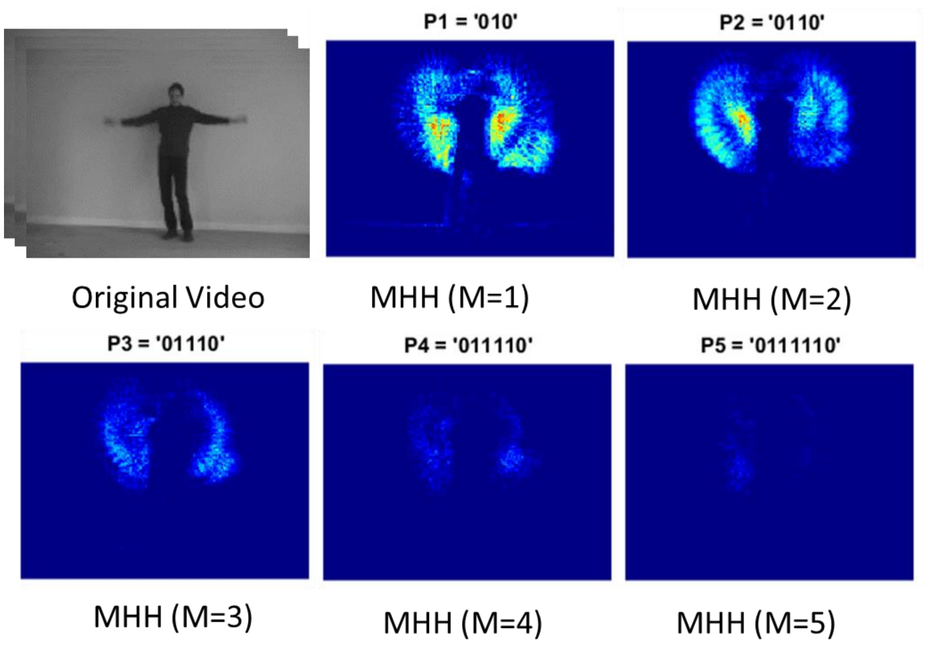


Figure 5.1 - MHH feature captured of a person waving their hands. Visually showing patterns up to M=5, along with the original visual sample

The algorithm for MHH from [30] can be understood as follows: Let $f(u = 1{:}U, v = 1{:}V, k = 1{:}K)$ be a visual sequence, where k is the frame number and (u, v) is a spatial location on a frame, representing the row and column. $D(u = 1{:}U, v = 1{:}V, k = 1{:}K - 1)$ is defined as a binary sequence of the operation in Equation 5.1. Where T is the threshold hyperparameter to determine the amount of motion required before accepting it has occurred. α is the frame difference between two pixels as calculated in Equation 5.2.

$$D_{(u,v,k)} = \begin{cases} 1, \alpha \geq T \\ 0, \ else \end{cases} \tag{5.1}$$

$$\alpha = \left| f_{(u,v,k+1)} - f_{(u,v,k)} \right| \tag{5.2}$$

$$\text{MHH}_{(u,v,m)} = \begin{cases} \text{MHH}_{(u,v,m)} + 1, & if\{P_m is\ found\} \\ \text{MHH}_{(u,v,m)} \quad , & else \end{cases} \tag{5.3}$$

Once D is computed, the next step is to check D for patterns P_m, where $m = 1{:}M$. The feature histogram MHH is also defined and initialised as $\text{MHH}(1{:}U, 1{:}V, 1{:}M) = 0$. A frame index $I_{(u,v)}$ keeps track of the number of the starting frame of a new pattern on the pixel (u, v). For each pattern pixel in $I_{(u,v)}$, pattern P_i will be observed which can be seen in Figure 2.11. Here, the number of consecutive '1' determines the duration of the motion at location (u, v).

The matrix I is initialised as $I(1{:}U, 1{:}V) = 1$. That means a new pattern starts from frame 1 for every pixel. $I_{(u,v)}$ will then be updated to $I_{(u,v)} = k$, while $\{D(u, v, I(u, v)), \cdots, D(u, v, k)\}$ builds one of the patterns from $P_m (1 \leq m \leq M)$. Each current pattern P_m must start and end with a $'0'$, to understand when the detected motion is complete. When a pattern P_m is produced, the feature histogram $MHH_{(u,v,m)}$ increases by 1, as shown in Equation 5.3. This process is repeated until $I_{(u,v)} = K$, for all pixel location. The full pseudo code for the algorithm is shown in Figure 5.2:

```
Algorithm (MHH)
Input: Video clip $f(u,v,k)$, $u=1,…,U$, $v=1,…,V$ frame $k=1,…,K$
Initialisation: Patterns $i=1,…,M$,
                MHH$(1:U,1:V,1:M) = 0$,
                Pattern Starting Index $I(1:U,1:V) = 1$
For $k=2$ to $K$ (For K)
    Compute $D(:,:,k,s)$ (0 or 1 based on frame difference)
    For $u=1$ to $U$ (For U)
        For $v= 1$ to $V$ (For V)
            If ($D(u,v,k)$=0)   (If 1)
                If {$D(u,v,I(u,v)),…,D(u,v,k)$} is pattern $i$  (If 2)
                    Update: MHH$(u,v,i)$=MHH$(u,v,i)+1$
                End (If 2)
                Update: $I$(u,v)=k
            End (If 1)
        End (For V)
    End (For U)
End (For K)
Output: MHH$(1:U,1:V,1:M)$
```

Figure 5.2 – Pseudocode for the MHH Descriptor Algorithm

Concretely, the MHH feature summarises the motion representation from a sequence of frames into a single frame per pattern. The feature itself can, therefore, be treated as a still image for motion as visualised in Figure 5.1. As a set of still images, the MHH feature can be processed further to provide information based on local feature descriptor techniques like those used in Chapter 3.2.1. As the dimensions of the MHH feature can be large when flattened, using these local feature descriptors can be viewed as a descriptive form of dimensionality reduction.

With the principals of the technique understood, there are some possible situations that can occur which can have a negative impact on the descriptor, or that the descriptor lacks the expertise needed for these situations. In the following section, these points are noted and discussed in detail.

5.3.2 Motion History Histogram inefficiencies

MHH has shown good performances in a variety of tasks that have some form of motion that can be interpreted. However, there are some areas in which it can be improved to be more efficient and robust. Here are some key factors of MHH that are missing or can be improved:

- MHH Lacks the ability to capture motion at different speeds, which can cause issues when motion varies throughout a sequence. Currently, the algorithm is designed to capture the fastest possible motion occurring as it only looks at two consecutive frames at a time.
    - A slow-paced motion may not be detectable by MHH if the input video has a characteristic of motion occurring across long time periods. MHH only utilises the following frame, which for slow motion, could look very similar to the previous frame.
    - Fast motion has the opposite effect, where consecutive frames provide big motion jumps within the frames. In this case, the MHH descriptor might not recognise them as the same source of motion because of the pixel distance.
- Multiple motion entities are combined and not detectable separately within the algorithm. This shows a limitation to global motion capture and not local.
- A lot of noise can be captured if the video source suffers from small stuttering or similar effects. This can be a potential problem caused by background objects in the scenes that are irrelevant to the task.
- The spatial size of the motion captured can be visually sparse if too large. This can cause hand-crafted techniques, such as LBP (if applied on top of the MHH feature), to miss gradually bending corners and other deviations.
- The motion captured is pixel based, not object/person based. This means that multiple entities can contribute to the motion capture of what is supposed to be of a single entity if there is any overlap between them or their paths.
- The Motion is only captured in grayscale. Converting a frame to grayscale could mean the loss of valuable information that could only be visible in a colour space.

These are some of the recognised inefficiencies of the MHH descriptor. There may be more, but of course, it is not easy to develop an algorithm that can take care of every situation for its application. However, it is clear that there is room for improvement for the MHH descriptor. The upcoming ideas will try to solve some of the issues mentioned above, as well as providing different views to the existing algorithm.

5.3.3 Extension 1 - Multi-Scale Motion History Histogram

The first proposed extension attempts to capture motion at different speeds. The issue of MHH, as mentioned previously, that relates to this extension is that the descriptor is fixed in how it looks at temporal data. This can make the feature very limited, and not dynamic enough to find all forms of motion. This extension, called Multi-Scale Motion History Histogram (MMHH), proposes to add more dynamics, providing motions that cannot be captured by the regular MHH descriptor.

Motion across visual sequences can indeed be examined at different scales, not necessarily from a frame with its following frame. Working frame by frame can limit the algorithms descriptiveness. This extension can help provide more information about slow and fast paced motion. This is achieved by adding a scale dimension that can compare each frame with various other following frames. The number of following frames to compare can be defined by the user via a hyper-parameter S. This can allow the tunability for different speeds of motion based on its intended application. By looking at various subsets of frames, hidden motion, or those submerged in noise can be detected and made clearer. These would otherwise be lost or covered up by the way MHH prioritises fast motion.

The new hyperparameter S is introduced to define the scales in which to calculate MMHH. This range can be defined as $s = 1, \cdots, S$. This will determine how many frames to compare pixel variation beyond frame k too. Each of these scales will also have M patterns recorded producing a feature vector of $\mathrm{MMHH}(U, V, M, S)$ dimensions.

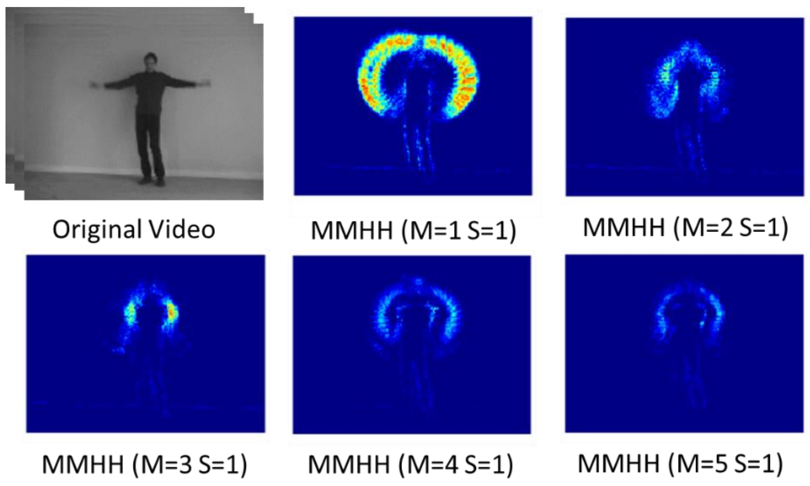


Figure 5.3 - MMHH feature captured of a person waving their hands. Visually showing patterns up to M=5, with the scale set to S=1, to demonstrate the difference of motion captured across each pattern when introducing S. The original visual sample is included

Figure 5.3 demonstrates the effect of adding the Scale dimension to MHH. The scale has only been set to $S = 1$ so it can be visually compared with Figure 2.12, which uses the same sample and shows the same number of patterns $M = 5$. When compared with the MHH visual pattern in Figure 2.12, the visual motion is more concentrated on the hands and where they move, showing a higher motion count for $M = 1$.

The MHH visualisation in Figure 5.1. Figure 5.3 shows its equivalent motion concentration, where the arms move closer to the body for $M = 1$. Also, for $M = 5$, MMHH shows higher motion count. It can be said that the faster motions are captured and are more related to MMHH at $S = 1$. This is based on how the hands are the fastest and furthest moving body part in this action, and how higher patterns $M > 2$ have been detected more. The motion is effectively amplified by the visual differences across the shorter time range.

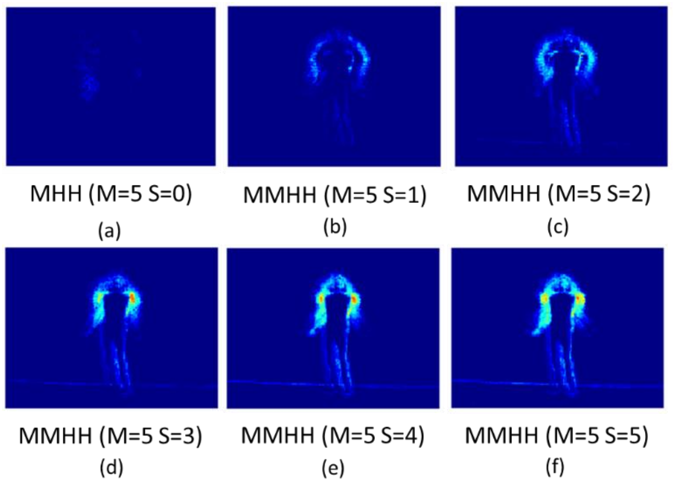


Figure 5.4 - MMHH feature captured of a person waving their hands. Visually showing patterns of M=5, with the scale increasing from S=0:5. S=0 represents the MHH equivalent for M=5 as there is no scale included.

Figure 5.4 is another visualisation of MMHH, where the scale is varied from $S = 0{:}\,5$ and showing only pattern $M = 5$. This is to demonstrate how the adjusting the scale can detect the long duration motion ($M = 5$) at varied speeds ($S = 0{:}\,5$). From the figure, the increase in scale shows how the speed of the motion is captured, with $S = 1$ demonstrating the fast hand movement, to $S = 5$ that shows the slower motion produced inner arm resulting from waving the hands. When taking the existing MHH parameters into account, the MMHH algorithm can be understood as follows:

$$\alpha = \left|f_{(u,v,k+s)} - f_{(u,v,k)}\right| \tag{5.4}$$

$$D_{(u,v,k,s)} = \begin{cases} 1, \alpha \geq T \\ 0, \ else \end{cases} \tag{5.5}$$

$$\text{MMHH}_{(u,v,m,s)} = \begin{cases} \text{MMHH}_{(u,v,m,s)} + 1, & if\{P_m is\ found\} \\ \text{MMHH}_{(u,v,m,s)} \quad , & else \end{cases} \tag{5.6}$$

The frame difference α is redefined as shown in Equation 5.4, along with the reformulation of the MHH descriptor in Equations 5.5 & 5.6 to produce the MMHH descriptor. Where the scale s is incorporated in the calculation of the frame difference, essentially defining how many frames to skip. $D_{(u,v,k,s)}$ represents the binary matrix that now includes the scale dimension. It is calculated by checking if α is $\geq$ the threshold T. The full algorithm is shown in Figure 5.5.

5.3.3.1 Time-Scaled Variation

The dynamics of the MMHH feature has increased compared to MHH. However, upon using the MMHH algorithm, there has been situations where the algorithm would not differ much from MHH. With thorough investigation, a vulnerability of MMHH was found. This was that the variability in framerate of a video can produce different outcomes even though the content is the same. i.e. if MMHH is applied to a video that is recorded at 30 fps, it will not produce the same feature if that same video was recorded at 60 fps instead.

```
Algorithm (MMHH)
Input: Video clip $f(u,v,k)$, $u=1,…,U$, $v=1,…,V$ frame $k=1,…,K$
Initialisation: Patterns $i=1,…,M$,
                Scales $s=1,…,S$,
                MMHH$(1:U,1:V,1:M,1:S)$=0,
                Pattern Starting Index $I(1:U,1:V)=1$
For $s=1$ to $S$ (For S)
    For $k=2$ to $K$ (For K)
        Compute $D(:,:,k,s)$ (0 or 1 based on frame difference)
        For $u=1$ to $U$ (For U)
            For $v=1$ to $V$ (For V)
                If ($D(u,v,k,s)=0$) (If 1)
                    If $\{D(u,v,I(u,v),s),…,D(u,v,k,s)\}$ is pattern $i$ (If 2)
                        Update: MMHH$(u,v,i,s)$= MMHH$(u,v,i,s)+1$
                    End (If 2)
                    Update: $I(u,v)$=k
                End (If 1)
            End (For V)
        End (For U)
    End (For K)
End (For S)
Output: MMHH$(1:U,1:V,1:M,1:S)$
```

Figure 5.5 - Algorithm of the MMHH extension

This would make sense as the scale dimension is linear in the way it is defined. This means that for visual sequences with different framerates, special attention will always be required to adjust the scales accordingly. If the framerate is high, this will cause the consecutive scales to be very similar. This can be inefficient as the feature will become unnecessarily larger. This would require the total number of scales S to increase in order to obtain the motion at the intended speed. This drawback does also play true to the MHH algorithm, as the quality is also affected by higher framerates.

A solution to this problem is to introduce the time dimension, where the measurement of movement is not based on the frame scale. Now it can be based on the timeline of a sequence, which can account for any variation in the framerate by selecting a time interval instead of a certain frame to calculate the motion. This provides a meaningful approach to the motions at various speeds. This solution will be called Time-Scaled Motion History Histogram (TSMHH).

The scale hyperparameter S from the MMHH extension has been changed to represent time instead, that can be denoted as τ. This parameter can be used to determine at what gap of time should two frames be observed. For instance, if the motion is to be calculated every half a second at a given framerate of $Fr = 30fps$, then $\tau = 0.5 \times Fr$. This means that each frame $f(u, v, k)$ will be compared with $f(u, v, k + \tau)$. There can also be Q instances of time-scaled motion captured, by making τ a vector of $\mathbb{R}^Q$. The benefit of having the parameter in terms of time can help the user judge what to set it to, by considering the application it is used for. An example of this use can be to detect very slow movements from emotions, such as signs of depression. The time parameter Fr can be set to capture the slow occuring motion.

5.3.4 Extension 2 - Spatial Motion History Histogram

Spatial Motion History Histogram (SMHH) aims to look at a different viewpoint within the spatial domain. The idea comes from Convolutional Neural Networks, in the way the network pools an image after applying a convolution filter. Here, the image from a sequence is pooled several times, and motion is captured on each new sequence.

By introducing pooling on the image sequences, the image size is reduced making the valuable information such as corners, edges and boundaries sharper. This can have the effect of highlighting areas of motion that provide useful information for situations like when a slow and long turning corner can be mistaken for a straight line. For the algorithm, there are three new tuneable hyperparameters when compared to MHH. The first is to decide the kind of pooling that is desired, with average pooling shown in Equation 5.7, max pooling in Equation 5.8, and the operation function in Equation 5.9.

$$Pool_{Avg} = \frac{1}{\beta^2 \cdot n} \times \sum f\big(u{:}\big(u + (\beta \cdot n)\big), v{:}\big(v + (\beta \cdot n)\big), k\big) \tag{5.7}$$

$$Pool_{Max} = \mathrm{MAX}\Big(f\big(u{:}\big(u + (\beta \cdot n)\big), v{:}\big(v + (\beta \cdot n)\big), k\big)\Big) \tag{5.8}$$

$$\bar{f_n} = Pool(f, n) \tag{5.9}$$

where $Pool$ can either be the Avg or Max depending on the user. $n = \{0{:}N\}$ is the number of pooling operations that will occur, with $n = 0$ representing no pooling operation, and continues to increment till N. β is the spatial pooling size, which will be a $\beta \times \beta$ window. $\bar{f_n}$ is the outcome of the frame sequence f when pooling is applied with pooling operation n. The rest of the algorithm is as follows:

$$\mathrm{SMHH}_{(n)} = \mathrm{MHH}(\bar{f_n}) \tag{5.10}$$

$$\mathrm{SMHH} = \left\{\mathrm{MHH}\left(1{:}\frac{U}{\beta^0}, 1{:}\frac{V}{\beta^0}, 1{:}M\right), \cdots, \mathrm{MHH}\left(1{:}\frac{U}{\beta^N}, 1{:}\frac{V}{\beta^N}, 1{:}M\right)\right\} \tag{5.11}$$

Equations 5.10 & 5.11 shows how the SMHH feature is created based on the collective MHH operation on the raw and pooled visual sequences. Equation 5.10 represents MHH being applied on a pooled sequence $\bar{f_n}$. This is repeated for all $n = 0{:}N$, with the 4th dimension to the SMHH feature becoming each MHH feature that is produced.

5.3.5 Extension 3 - Spatial Multi-Scale Motion History Histogram

Both extensions MMHH and SMHH can work effectively together to produce a comprehensive feature detailing various motion speeds with varied shape and size. This can ultimately be the feature descriptor that can uncover motion undetected by the MHH algorithm, giving a significantly higher amount and quality of information. However, this is at the cost of having a much larger feature size.

The algorithm Spatial Pooling Multi-Scale Motion History Histogram (SMMHH) is based on applying the MMHH algorithm on the spatially pooled sets of visual sequences, including the original set. The entire process is visually depicted in Figure 5.6, using an action sample of a person waving their hands. The total feature size can be computed as:

$$dim = \sum_{i=0}^{N} \left(\frac{1}{\beta^2 \cdot i} (U \cdot V \cdot M \cdot S) \right) \tag{5.12}$$

Two variants are based on the Average and Max operator for the spatial pooling. These are denoted as ASMMHH for average pooling, and SMMHH for max pooling.

5.4 Dynamic/Temporal Datasets

Five public databases have been utilised in the upcoming experiments, that are based on a variety of different applications. This is to demonstrate the performances of MHH and the extensions across different forms of motion occurrence.

5.4.1 CoST Database

The Corpus of Social Touch (CoST) dataset is based on touch gestures with human interaction [156]. It contains 14 types touch gestures which are: grab, hit, massage, pat, pinch, poke, press, rub, scratch, slap, stroke, squeeze, tap and tickle. There are also 3 variations of each gesture (normal, gentle and rough). There is a total of 31 subjects producing each of the 14 gestures 6 times.

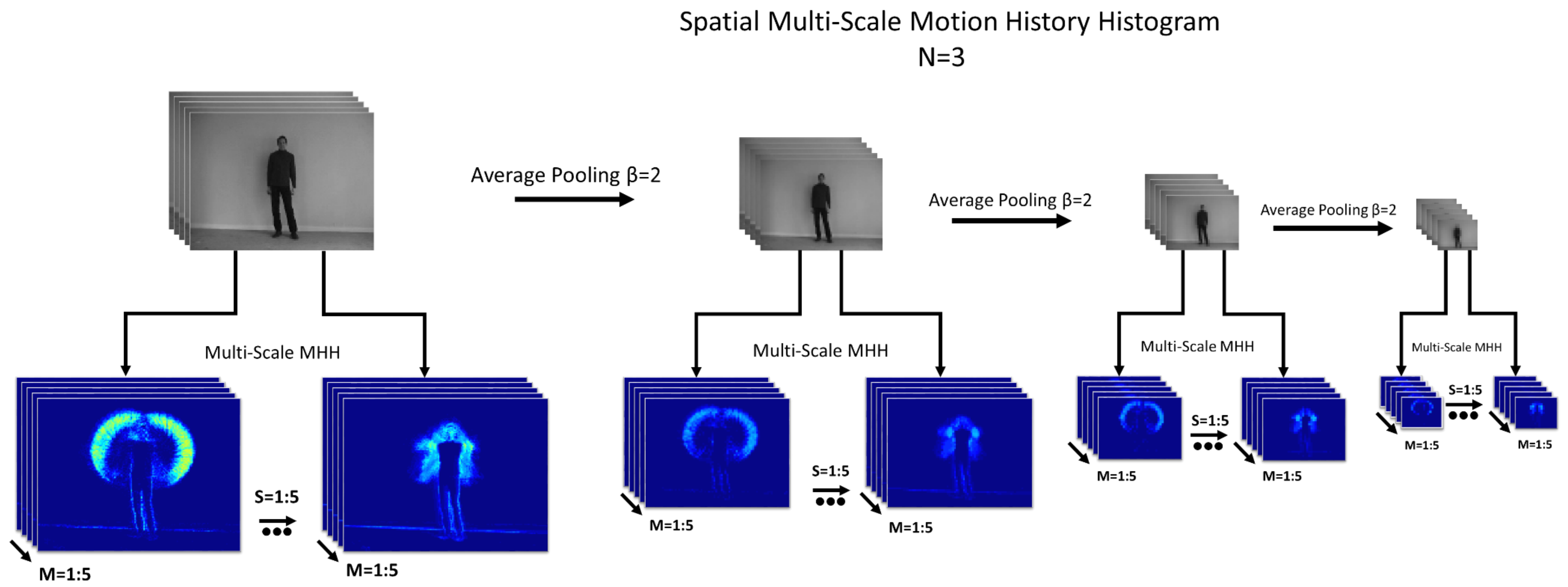


Figure 5.6 - Overview of Spatial Multi-Scale Motion History Histogram being applied to a person waving their hands. Average pooling is applied N = 3 times on the video data, each with β= 2. Multi-scale Motion History Histogram is used to capture the motion data on the original and pooled videos, using the Multi-scale of S = 1: 5 with all patterns from M = 1: 5

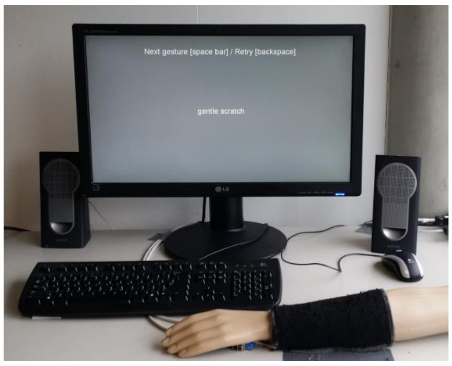


Figure 5.7 - Placement of pressure sensors (black fabric) on a mannequin arm

The device used to record the gestures is an 8×8 grid of pressure sensors. They had been wrapped around a mannequin arm, as shown in Figure 5.7, which was chosen as a neutral body location to place the sensors. The pressure intensity is the measuring unit for the sensors and are recorded at 135 times per second for the duration of the gesture. Each sensor channel produces a 10-bit integer value between 0 and 1023. The detailed description of how each of the 14 gestures is acted is mentioned in Table 5.1 [156].

Table 5.1 - CoST gesture labels and definitions [156]

Gesture Label	**Gesture Definition**
Grab	Grasp or seize the arm suddenly and roughly
Hit	Deliver a forcible blow to the arm with either a closed fist or the side or back of your hand.
Massage	Rub or knead the arm with your hands.
Pat	Gently and quickly touch the arm with the flat of your hand.
Pinch	Tightly and sharply grip the arm between your fingers and thumb.
Poke	Jab or prod the arm with your finger
Press	Exert a steady force on the arm with your flattened fingers or hand.
Rub	Move your hand repeatedly back and forth on the arm with firm pressure.
Scratch	Rub the arm with your fingernails
Slap	Quickly and sharply strike the arm with your open hand.
Squeeze	Firmly press the arm between your fingers or both hands.
Stroke	Move your hand with gentle pressure over the arm, often repeatedly.
Tap	Strike the arm with a quick light blow or blows using one or more fingers
Tickle	Touch the arm with light finger movements.

The HAART database [157] is based on expressing emotion towards a “haptic creature”, with the intent to understand and observe human emotion via touch gestures. A physical prototype is created in the form of a furry animal, with pressure and conductive sensors attached to the skeleton and fur. The sensors are in the form of a 10×10 grid, with each sensor the size of 1 square inch wide. An 8×8 grid is provided to match the CoST data format. This device can be seen in Figure 5.8, with the complete prototype on the left, the Styrofoam and skeleton on the top right, and the skeleton with the fabric pressure sensor on the bottom right [157].

5.4.2 HAART Database

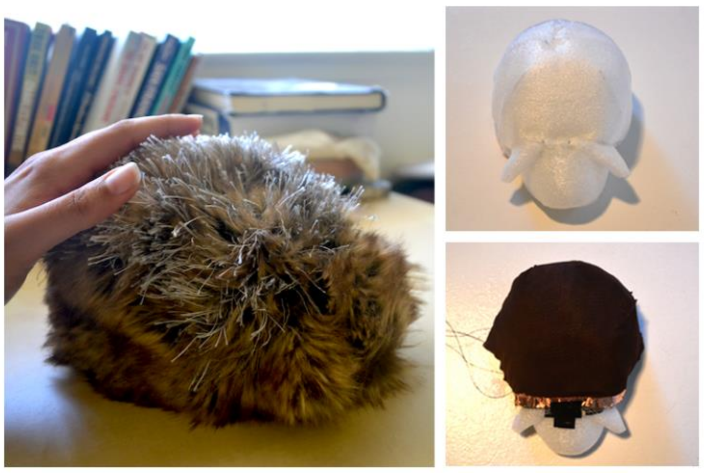


Figure 5.8 - An affective touch-sensing zoomorphic prototype with pressure and conductive sensors attached to the device, inner Styrofoam and skeleton [157].

The data collected from the prototype contains 7 touch gestures that include: constant, no touch, pat, rub, tickle, scratch, and stroke. These gestures are derived from the touch dictionary provided by Yohanan et al. [158], which communicate emotions in human-animal interactions. The gestures are performed by 10 subjects, with each gesture being 10 seconds long. There are 3 substrate conditions (firm and flat, foam and flat, foam and curve), with 4 cover conditions (none, short minkee, long minkee, synthetic fur) for each gesture. This produces a total of 840 gestures. The sampling rate is set to 54 times per second, with each sensor channel producing a 10-bit integer value between 0 and 1023.

5.4.3 KTH Database

The KTH is a database on human actions recorded as a visual sequence. This database has been thoroughly experimented on by many researchers as it contains 6 general actions that are performed clearly which are: walking, jogging, running, boxing, hand waving, and hand clapping. There are 25 subjects performing the actions in 4 different settings. These are outdoor actions *s1*, outdoor actions with scale variation *s2*, outdoor actions with different clothes *s3* and indoor actions *s4*, each of which has a homogeneous background. Samples of these actions with different scenarios can be seen in Figure 5.9. This makes a total of 600 action samples, however, with one sample missing. Each sample is recorded at 25 frames per second, with on average 4 seconds per sample, and a spatial resolution that is down-sampled to 160×120 pixels. Each sample sequence contains 4 sub-sequences of the same action, which can be split to produce a total of 2391 sequences [144].

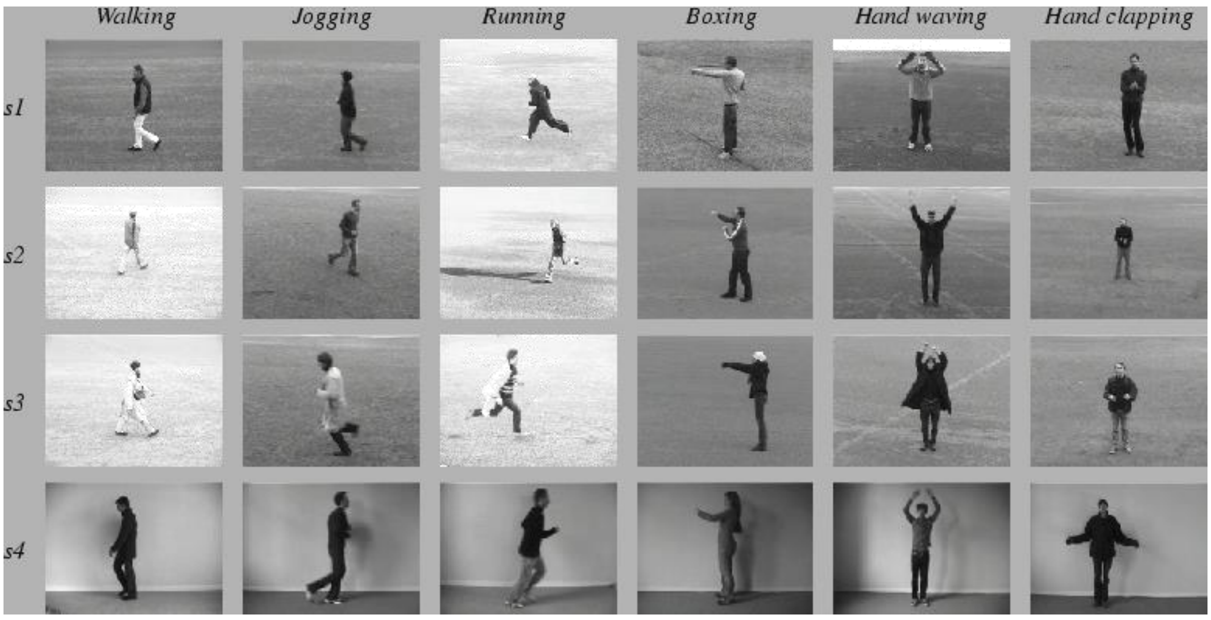


Figure 5.9 - Samples of the KTH database taken from [144] *demonstrating the 6 different actions based on 4 scenarios*

5.4.4 AVEC2014 Dataset

The AVEC2014 Dataset [13] is a subset of the audio-visual depressive language corpus (AViD-Corpus) [12]. The dataset contains a total of 300 video clips from 84 subjects performing Human-Computer Interaction tasks whilst being recorded by a webcam and a microphone in several quiet settings. Out of all the tasks, two have been selected based on being the most completed tasks. These tasks are the "Northwind" and "Freeform", which require the following [13]:

- Northwind - Participants read aloud an excerpt of the fable "Die Sonne und der Wind" (The North Wind and the Sun), spoken in the German language
- Freeform - Participants respond to one of many questions such as: "What is your favourite dish?"; "What was your best gift, and why?"; "Discuss a sad childhood memory", again in the German language

There is only one person in each clip and some subjects feature in more than one clip. All the participants are recorded between one and four times, with an interval of two weeks. 18 subjects appear in three recordings, 31 in 2, and 34 in only 1 recording. The length of the clips for the two tasks is between 6 seconds to 4 minutes 8 seconds. The mean age of subjects is 31.5 years, with a standard deviation of 12.3 years and a range of 18 to 63 years. The depression is measured using the Beck Depression Inventory II, which ranges from 0 to 63. The value ranges can be interpreted as follows: 0-10 is considered normal as ups and downs, 11-16 is mild mood disturbance, 17-20 is borderline clinical depression, 21-30 is moderate depression, 31-40 is severe depression and over 40 is extreme depression.

In terms of the data format, the audio is produced with a variable sampling rate and resampled to a fixed 128kbps using the AAC codec. The visual content has been recorded using a variety of codecs and frame rates. This was also resampled using the H.264 codec, to be at a uniform 30 frames per second with a spatial resolution of 640×480 pixels. This was stored in an mp4 container. Samples of the recordings that are allowed to be published are shown in Figure 5.10.

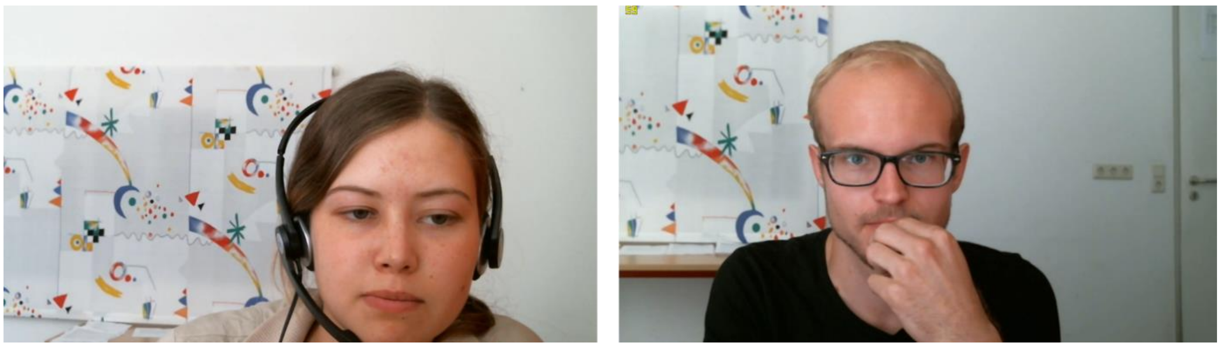


Figure 5.10 - Controlled sample frames provided by AVEC2014 that can be published, showing 2 people completing their task

The data is split into 3 partitions, which are for training, development, and testing. The purpose of these partitions is to try and build a system that is trained using the training partition, which can then be tuned to perform well on the development partition. Once the system is tuned and validated, the test partition can be used to give the final score of the system. This way there is no biases as the system is only tuned for the development data. For the majority of the global challenges, the testing labels are not released, and the predictions generally should be submitted to the hosts, so they can evaluate the performance.

5.5 Frameworks for Action/Gesture Recognition and Emotion Recognition

The framework for action and gesture recognition will be based on making the most of MHH and its extensions. To allow fair testing of MHH and each extension, the framework around each feature will remain the same. The common hyperparameters will also be kept the same so that only the contributions from each extension will be highlighted from the differences in performance. The initial frameworks will be based on the human action recognition and gesture recognition applications. These are followed by the Emotion recognition frameworks, which will be based on Depression analysis. These emotions portray slower motion throughout the sequences. Therefore, there will be a separate set of parameter values for them to make the best of the descriptors.

5.5.1 Framework for Touch Gesture Recognition

The CoST and HAART databases contain similar data, and each will host the same framework shown in Figure 5.11. The data format for both is a sequence of an 8×8 grid, which represent the pressure sensor readings. The values range from 0 to 972 for HAART, and 0 to 1023 for CoST. For this experiment, the sequence of pressure sensitivity is treated as a sequence of 8×8 pixels with a range of

10-bits, which can be visualised in the framework. For this experiment, the following features are tested: MHH, MMHH, SMHH, SMMHH (Average and Max), and the LBPTOP feature for comparison.

The MHH and extension algorithms are applied on top of the sequence to try and model the motion of the gesture. Each algorithm has the same set of values for the Threshold parameter T, and the number of patterns M. The threshold has a range of values tested from 0.75 till 30. These are directly linked to the database as if depends on values of the sequences. The remaining parameters are defined as stay the same for each experiment.

Starting with MHH, each sequence will have several versions captured, mainly adjusting the threshold. The number of patterns M is set to 5 for all features. For the MMHH feature, the scale parameter S is set to 5. This is to make sure various speeds of motion will be captured for each gesture. The threshold will vary the same as the MHH feature, with the pattern count also set to $M = 5$. This will give a total of features size of $8 \times 8 \times 5 \times 5 = 1600$ components.

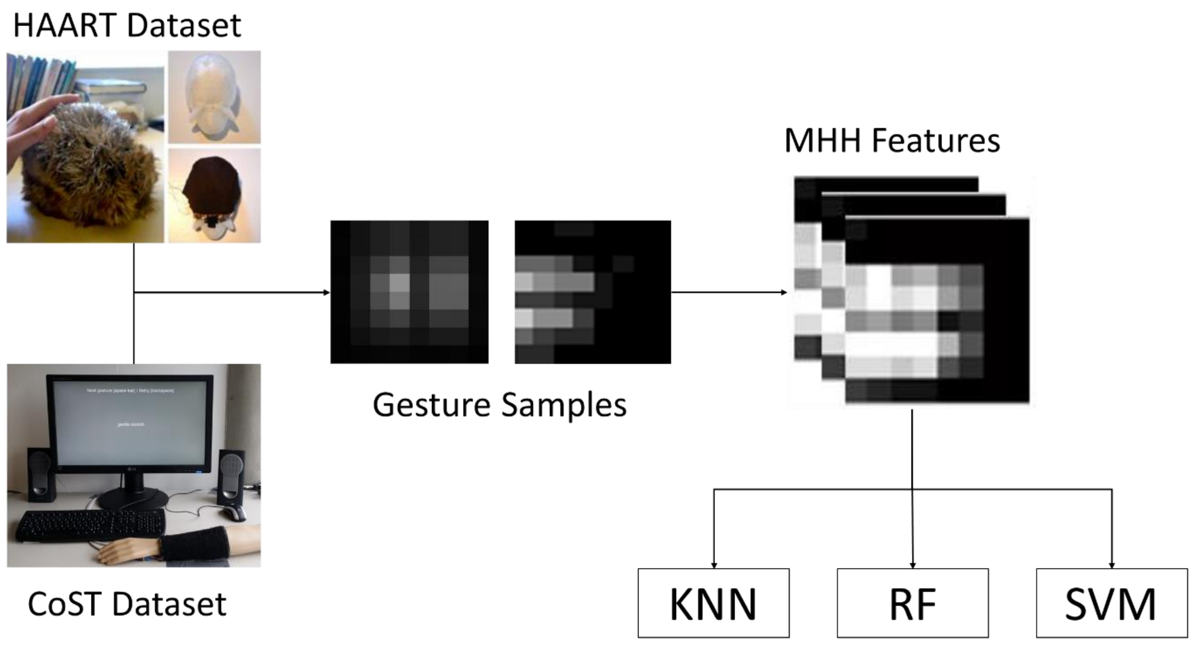


Figure 5.11 - Framework for HAART and CoST dataset, where the gesture samples are treated as videos, and MHH + extensions are captured off. KNN, SVM and RF are used for the feature learning process to predict the gesture.

When applying SMMHH to the framework, β is set to 2, and N is set to 3. This is so that the sequences can successfully be pooled, making 4 (including original) total sets of sequences. For each gesture, this becomes an 8×8, 4×4, 2×2 and 1×1 resolution sequence. Each set of sequences will have scales of up to $S = 5$ and patterns of $M = 5$.

The feature learning will be based on a classification task and will involve 3 commonly used classifiers. These are KNN, RF, and SVM. Multiple classifiers are adopted to observe the performance difference using different techniques. They can also validate the performance of each other, making it visible if there is an anomaly result. The KNN and RF classifiers have their hyperparameter adjusted to find the best for this task. For KNN, this is to select the number of neighbours to use, and for RF, the number of trees to have in the classifier.

5.5.2 Framework for Human Action Recognition & Dynamic Facial Expression Recognition

The framework for the KTH database and AVEC2014 database contain similar format data, in the form of an image sequence. The framework for them will vary from the HAART and CoST database as they are based on actual video content and not a grid of sensors. Because of this, the frame sizes are a lot bigger and will make the resulting feature very large. To overcome this, the motion histograms will be reduced by applying a local feature descriptor on each motion frame, as each motion histogram has the same spatial dimensions of the samples frame size. This will not only reduce the dimensionality but will also provide a mathematical representation of the motion captured.

The main goal of the experiment is to evaluate the motion descriptors. Therefore, it is not necessary to use many local descriptors on top of the motion features frames. As the output from the proposed methods can be large, each of the resulting motion histograms of 120×160 (for Action Recognition) for all methods will have EOH applied to it, reducing the feature size from $19{,}200$ components per frame down to 384. The local descriptor will be used on each frame for all the motion features, in the same way to allow for fairness. Each of the EOH feature produced from each frame of a motion feature will be concatenated together to produce a single feature vector. This vector will represent each sample.

Starting with the original MHH feature, this has been captured across all of the videos with patterns up-to $M = 5$ along with a threshold of $T = 8$. The result would be a feature of 5 frames, each equivalent to the frame size of the original video (120×160). Each of these frames will have EOH extracted from them and concatenated together. The final MHH feature will now consist of $M \times 384 = 1920$ components, which is denoted as MHH_EOH. There are 2 Spatial Motion History Histogram with EOH features captured, first is using max pooling, which is denoted as SMHH_EOH and the other is using Average pooling, which is denoted as ASMHH_EOH. The spatial pooling is applied $N = 3$ times, each with a scaling factor of $\alpha = 2$. This will produce 4 videos: the original video, and the 3 pooled videos. MHH is applied to each video capturing patterns up-to $M = 5$; all with a threshold set of $T = [5, 8, 10, 12, 15]$. EOH is taken of all the patterns produced from each scale on each pooled video, making a total feature size of $(N + 1) \times M \times 384 = 7680$ components.

The next feature is Multi-Scale Motion History Histogram with EOH, denoted as MMHH_EOH, and the Time-Scaled Motion History Histogram denoted as TSMHH_EOH. The MMHH_EOH feature introduces a scale factor to obtain a dynamic feature that is affected by the movement speed. The Scale value chosen goes from $S = 1{:}5$, with the patterns, produce up-to $M = 5$, and a threshold set of $T =$

$[5, 8, 10, 12, 15]$, like MHH. For each scale, there will be 5 frames produced which represent each of the patterns P_m. EOH is taken of all the patterns produced from each scale, making a total feature size of $S \times M \times 384 = 9600$ components. TSMHH slightly differs by replacing S with τ, where τ is set to $\tau = [0.5, 0.75, 1, 2, Fr]$. When calculating the number of frames to skip, the algorithm simply applies $\frac{Fr}{\tau}$. E.g. if $Fr = 25$, then the resulting scales will be $S = [50, 33.3, 25, 12.5, 1]$.

Spatial pooling and Multi-scale have been combined and tested for this framework. There are two features captured with this method, using both Max pooling and Average pooling denoted as SMMHH_EOH and ASMMHH_EOH respectively, with parameters $N = 3$ and a scaling factor of $\alpha = 2$. MMHH is then taken on each of the videos with a Scale of $S = 1{:}5$; patterns produced up-to $M = 5$; with a threshold of $T = [5, 8, 10, 12, 15]$. EOH is taken of all the patterns produced from each scale on each pooled video, making a total feature size of $(N + 1) \times S \times M \times 384 = 38{,}400$ components.

5.5.3 Framework for Depression Analysis

For Depression analysis, the framework will differ from the others as it is based on a regression task, rather than classification. Unlike the other databases, the AVEC dataset contains audio and visual information that can both be utilised in the upcoming experiment.

The idea of this framework is to capture the temporal motion using the algorithms (MHH, MMHH, SMHH, SMMHH and TSMHH) and apply local hand-crafted descriptors to further extract information from it. The framework is shown in Figure 5.12, which can be understood in 4 stages. The first stage converts the raw data into grayscale and then applies the MHH algorithm or its extensions to extract motion data. Stage 2 then uses these motion frames as samples for a set of three hand-crafted descriptors of EOH, LBP and LPQ. These features are fused together by concatenating each descriptor histogram from each motion frame to make one big feature vector. Then, stage 3 applies PCA to reduce the number of dimensions from the fused feature vector. And finally, stage 4 applies multiple regression techniques to predict the depression score. The efforts from these predictors are further fused at decision level to give a joint prediction of a score between 0 and 63.

The evaluation metric is based on the Mean Absolute Error (MAE), and the Root Mean Squared Error (RMSE). MAE is defined in Equation 5.13 and RMSE in Equation 5.14, where n is the number of labels, y is a set of ground truth labels, and $\hat{y}$ is the set of prediction labels by the system.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5.13}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5.14}$$

The regression techniques that are used for the Depression analysis are Partial Least Squares regression and Linear Regression. Both techniques are applied individually for each feature set, as well as jointly using decision level fusion.

$$\hat{y_i} = \sum_{i=1}^{N} \sum_{j=1}^{R} w_j \cdot y_{(i,j)} \tag{5.15}$$

The predictions from these techniques are combined using decision level fusion based on the weighted sum rule. This is shown in Equation 5.15, where R is the number of regression techniques used, N is the total number of samples, $y_{(i,j)}$ is the prediction value from the regression method j, of sample i, w_j is the weight assigned to regression method j. The objective is to fuse the confidence measure of each regression technique to get one set of predictions $\hat{y}$.

The efforts from both are linearly fused together using a weighted sum rule to aggregate their predictions. The weights are optimized by observing the performance of PLS and LR on the development partition. Initially, they both start with a weight of 0.5, with a step size of 0.05 whilst being optimised. The first change in weight is based on the individual performance for each technique, with the best performer getting an increase in weighting, and the worst receiving a loss the equivalent amount. Whilst the performance gets better, the weights continue to increase/decrease in the same direction. Once the performance increase halts, the optimal solution is decided based on the lowest error.

5.6 Experimental Settings and Results

In this section, four experiments will take place based on four databases. Each of which will be to demonstrate the performance of MHH and the proposed extensions. The first 2 experiments will be to evaluate the performance of the MHH and extensions on the KTH, CoST and HAART database. This is to provide an indication of how they perform on applications that contain a lot of motion. Then the last experiment will focus on applying these techniques into emotion recognition and detection, using the AVEC14 database. This will be in the form of Depression analysis.

5.6.1 Settings and Protocols for Action/Gesture Recognition

The CoST and HAART tasks have the guidelines followed by the Social Touch and Gesture Challenge that took place in 2015 [159], by using the training and testing sets they suggested. During the challenge, the test partition labels were not made available to the public. Therefore, the training set has been split into a ~70% training subset and ~30% validation subset. For the HAART training subset, there are 412 samples, and 166 samples for the validation subset. For the CoST training set, the training data is split into 2685 samples for the training subset and 839 samples for the validation subset.

Initially, to get a better understanding of the parameter selection for the machine learning methods, 10-fold cross-validation is applied on the training subset. Once they are defined, the training and validation subset are used to determine the performance of each feature method. The LBP-TOP feature is also computed on the sequences for each sample as a reference to compare the MHH and its extensions performances.

For the KTH experiment, protocols are followed based on the experiments from [144] and [30], where the subjects are split so that 16 are used for training and 9 are used for testing. This will give a subject independent view of the experiment, which is a bonus for robustness. The proposed methods are applied to the first 300 frames of the 599 video clips, as done in [85].

5.6.2 Experiment 5A – Human Action Recognition based on Temporal Changes

The first experiment will be based on detecting human actions using the KTH database. The following MHH features and its extensions will be tested: MHH, MMHH, TSMHH, SMHH, ASMHH, SMMHH, and ASMMHH. Each feature will be tested on the dataset using the same protocol, to provide fair testing using controlled settings. There will be a range of thresholds used for each feature, which can help determine what the best is for this application.

The machine learning techniques that will be adopted for this experiment are SVM, KNN and RF. SVM will consist of both RBF and Poly kernels, as used in the previous experiment. The cost parameter C will be fixed to 1000, and $\gamma = \frac{1}{\hat{n}}$. For KNN, the number of neighbours is set to 5. And finally, for RF, the number of trees is set to 500. There will be no cross-validation procedure for the testing, as mentioned in the experimental protocol.

The resulting feature vectors after the extraction process can become very large and take a long time to train. However, it will not be necessary to apply PCA to reduce the dimensionality and speeding up the training process, because each test will only be run once. Therefore, the full feature can be tested to produce the best performance it can.

5.6.2.1 Performance Evaluation of Motion Based Descriptors

Starting with the original MHH feature, Table 5.2 shows its performance based on 5 different thresholds, using 4 machine learning techniques. This feature can be used as a baseline, to evaluate the performance of the proposed extensions against it. The highest result has come from the MHH_15 feature using SVM with RBF kernel, producing 87.50%. It is hard to determine the best threshold for this application, as it can depend on the machine learning technique. The machine learning techniques show that SVM does come on top in most cases for each threshold, but it is closely followed by RF. KNN looks like the weakest techniques with all thresholds of MHH.

Table 5.2 - Performance of the MHH descriptor on the KTH database, with 5 patterns captured using various thresholds. These are tested based on a 70/30 split of the data, using SVM RBF and Poly, RF and KNN for machine learning.

Accuracy	SVM RBF	SVM Poly	RF 500	KNN 5
MHH_5	79.62%	76.85%	81.02%	74.07%
MHH_8	86.11%	85.64%	**86.11%**	80.09%
MHH_10	86.57%	**87.03%**	83.33%	78.70%
MHH_12	87.04%	85.64%	84.25%	79.62%
MHH_15	**87.50%**	86.54%	85.65%	**82.41%**

Table 5.3 shows the performance of the MMHH and TSMHH extensions, with some interesting findings. Firstly, both features have shown an improvement over the MHH feature, with an increase of 1.85%. This may not be a large amount, mainly because it is limited by the number of samples in the dataset. Between MMHH and TSMHH, in most cases, TSMHH has produced a better recognition rate across all machine learning techniques.

Table 5.3 - Results based on the first set of MHH extensions: MMHH and TSMHH.

Accuracy	SVM RBF	SVM Poly	RF 500	KNN 5
MMHH_5	82.41%	83.33%	79.63%	77.22%
MMHH_8	87.50%	86.11%	81.94%	81.48%
MMHH_10	87.96%	**86.57%**	83.80%	82.40%
MMHH_12	88.88%	86.11%	**87.96%**	85.18%
MMHH_15	**89.35%**	**86.57%**	85.65%	**85.65%**
TSMHH_5	85.64%	85.18%	81.48%	78.24%
TSMHH_8	86.11%	87.03%	82.87%	82.40%
TSMHH_10	90.27%	88.89%	83.79%	82.40%
TSMHH_12	**90.74%**	**89.35%**	84.25%	**86.11%**
TSMHH_15	89.35%	88.89%	**85.18%**	85.18%

This can be an indication that capturing motion based on the time scale can provide more insightful information. However, it is interesting to know if each application will have its desired speed of motion to capture the best possible motion features for it. Multi-scale has also shown a positive increase in performance throughout, indicating that using more information and capturing more scales of motion can improve performance.

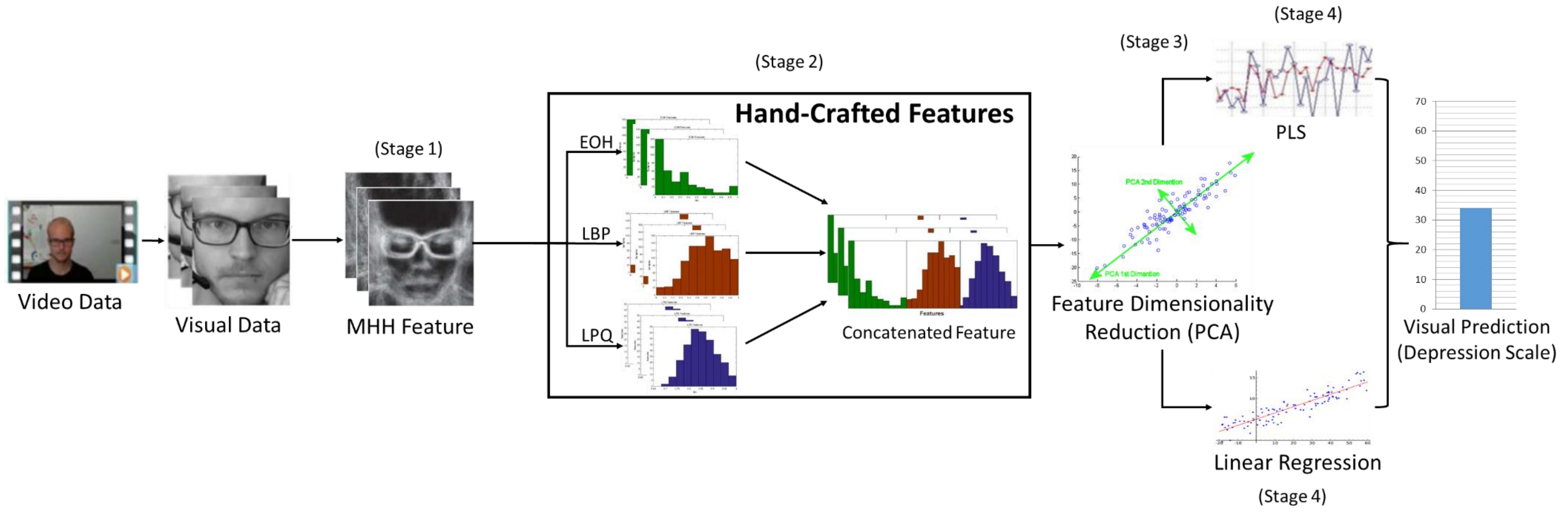


Figure 5.12 - Depression Framework based on the visual data for depression scale prediction, showing the 4 stages from start to end. The visual sequences have the MHH descriptor and its extensions extracted, from which hand-crafted descriptors are captured and concatenated together. The feature is reduced in dimensionality using PCA and regression techniques are applied to predict the depression scales.

Table 5.4 portrays the introduction of spatial pooling, with both average pooling and max pooling applied. Here, the best performer is the ASMHH feature, producing 88.89% using a threshold of 8. However, the SMHH feature has not performed well compared to the baseline, hitting only a max accuracy of 83.79% using SVM with both kernels. This is an accuracy decline of 3.71%.

Table 5.4 - Results based on the SMHH and ASMHH features

Accuracy	SVM RBF	SVM Poly	RF 500	KNN 5
SMHH_5	80.09%	80.55%	79.69%	74.07%
SMHH_8	82.40%	**83.79%**	79.63%	73.14%
SMHH_10	**83.79%**	82.09%	73.15%	**79.17%**
SMHH_12	81.94%	81.48%	**80.55%**	72.22%
SMHH_15	82.87%	82.87%	79.62%	67.12%
ASMHH_5	79.62%	79.62%	74.07%	79.62%
ASMHH_8	**88.89%**	86.57%	**87.50%**	**84.72%**
ASMHH_10	88.42%	**87.96%**	84.72%	78.24%
ASMHH_12	79.16%	79.16%	79.16%	72.22%
ASMHH_15	80.55%	81.48%	80.09%	67.12%

Table 5.5 - Performance of the SMMHH and ASMMHH features

Accuracy	SVM RBF	SVM Poly	RF 500	KNN 5
SMMHH_5	86.11	87.03	82.87	79.62
SMMHH_8	86.57	87.96	82.87	77.78
SMMHH_10	87.50	87.96	83.33	81.48
SMMHH_12	**88.89**	**88.89**	**85.19**	79.63
SMMHH_15	87.04	87.50	84.26	**83.33**
ASMMHH_5	84.72	85.64	81.01	74.53
ASMMHH_8	**92.13**	**91.20**	87.04	88.89
ASMMHH_10	90.27	89.35	87.50	**89.35**
ASMMHH_12	89.88	87.96	**88.89**	87.03
ASMMHH_15	86.57	88.89	85.18	87.50

Table 5.5 shows the performance of the combined efforts of the spatial pooling and multi-scale extensions, which are the SMMHH and ASMMHH features. ASMMHH_8 has performed significantly well compared to the baseline, achieving 92.13%, which is a 4.63% increase in accuracy. This is also the best performing feature compared to MHH and all the extensions. SMMHH has also performed

well, beating the baseline, but not as well as ASMMHH. The threshold produces performances that vary with each machine learning technique, as well as each feature. Therefore, determining a threshold is not as straightforward. However, the performance does not significantly degrade from thresholds ranging from 8 and 12.

5.6.2.2 Experiment 5A Highlights

This experiment has tested the proposed MHH extensions for human action recognition against the baseline MHH feature, to demonstrate the boost in performance each has over the original feature. 3 extensions have been proposed, each with a variant of its own, making 6 feature sets in total. The experiment has thoroughly tested each extension using a range of threshold values, along with various machine learning techniques. A summary of the performances for each motion feature set, including external descriptors by other people, are presented in Table 5.6. Mattivi et al. [85] have used SVM to classify their features LBP-TOP, Extended LBP-TOP and Extended Gradient LBP-TOP. They have also tested Laptev's method [160] of HOG-HOF feature with the same experimental setup. Kläser et al. [155] have used the HOG3D feature for their experiment using the same protocol. The best-performing feature out of all is ASMMHH with SVM RBF, narrowly beating the external descriptor HOG3D and Extended Gradient LBP-TOP. This demonstrates that the MHH extension has credibility and can compete with the state-of-the-art.

In all tests, apart from the MHH feature, the best performances are recorded with a threshold range between 8 and 12. This demonstrates that there may not be a single stable threshold that can work best for an application. It can vary based on the type of motion feature used, the machine learning technique adopted, and the speed of motion within the data.

Time-Scale MHH has shown to be a lot more effective than MMHH, reaching 90.74% to become the second-best performer out of MHH and the extensions. The difference in performance can be due to the wide range of motion capture speeds. Multi-scale only looks at the consecutive frames for motion difference. However, when the frame rate is high, this will not be very effective as each scale feature will have captured similar motion to the previous and following. TSMHH ensures that the motion speed desired will always be captured, as it is frame-rate invariant, providing robustness.

In terms of including spatial pooling, there is a benefit in using it, but only when set to average pooling. Max pooling does not seem to provide any benefits, according to the SMHH and SMMHH feature performances. The best performance was achieved using average pooling with multi-scale. This experiment has also demonstrated the capabilities of SVM, as previously demonstrated in the experiments from Chapter 3. Both kernels have once again trumped in performance when compared to RF and KNN. However, this time there is a small gap between them and a bit more competitive. The RBF kernel has shown to be the better out of the kernels, slightly outperforming the Poly kernel.

Table 5.6 - Comparison of all motion feature sets, using SVM, RF and KNN. The best result is taken from the tested set of threshold values, to represent each feature vector. Other feature descriptors are included for comparison.

Accuracy	**SVM RBF**	**SVM Poly**
MHH	87.50%	87.03%
MMHH	89.35%	86.57%
TSMHH	90.74%	89.35%
SMHH	87.39%	87.39%
ASMHH	88.89%	87.96%
SMMHH	88.89%	88.89%
ASMMHH	**92.13%**	**91.25%**
LBP-TOP	86.25%	**-**
Extended LBP-TOP	88.19%	**-**
Extended Gradient LBP-TOP	91.25%	**-**
HOF-HOG	89.88%	**-**
HOG3D	91.40%	**-**

5.6.3 Experiment 5B – Social Touch and Gesture Recognition

This experiment will apply the proposed MHH extensions in a similar style to experiment 5A, but for touch gesture applications. The main purpose is to see how effective the motion descriptors are across different applications that can benefit from this. From experiment 5A, it seemed like having ASMMHH and SMMHH produced the same of better performance as ASMHH and SMHH respectively. This is understandable as there are more scales of the same kind of motion information captured. Therefore, the ASMHH and SMHH feature will be omitted from this experiment as it will not be necessary.

For this experiment, we will consider how MHH and all proposed extensions will perform on the HAART and CoST datasets. Tests will be run on both datasets, using the same motion feature sets, and same protocol. The LBP-TOP will also be tested in the same protocol for comparison purposes. The machine learning techniques adopted are KNN, SVM and RF, with the hyperparameter settings based on experiment 5A. The experiment on both datasets had taken place during this challenge, with the results and findings published in [161]. The protocol for thoroughly testing MHH and extensions is not based on the global challenge in [159], as the test labels had not been released at the time of this experiment. For the protocol, a 10-fold cross-validation approach has been adopted on the available training samples for each dataset. All the samples are randomly shuffled before the data is split, to ensure that each test is different. The average from 50 tests is taken for each machine learning and feature set, to give a stable result. The performance measurement for both datasets are based on recognition rate out of 100%, to determine how accurately the system can predict gestures.

5.6.3.1 *Human-Animal Interaction Gesture Recognition based on HAART dataset*

Starting with the HAART dataset, there are 578 samples in total for the training set. The test set contains 251 samples, but the labels are unknown due to the hosts not releasing them. The 10-fold cross-validation will split the data into 520 for training and 58 for validation for each fold. Like the previous experiment, we start with the MHH feature. The data itself is not originated as an image, but an 8×8 grid of pressure sensor values. Therefore, the threshold values chosen are set low to allow more changes in the pressure activity to be picked up. These are the following: $T = [0.5, 1, 1.5, 2, 3]$, which is applied to each feature. Although the dimensions of these features are not too large, PCA is still applied to remove any highly-correlated features, by keeping 99% of the variance. This is because the larger feature vectors produced by MMHH and ASMMHH can contain a lot of no motion features throughout all the samples.

Table 5.7 - HAART test based on the MHH descriptor, using SVM, RF and KNN

Accuracy	SVM RBF	SVM Poly	RF 250	KNN 5
MHH_0.5	50.20%	49.61%	42.36%	39.41%
MHH_1	63.14%	61.43%	55.98%	56.05%
MHH_1.5	**63.19%**	**61.57%**	56.77%	**55.99%**
MHH_2	59.68%	59.02%	58.10%	55.67%
MHH_3	58.95%	56.41%	**58.78%**	54.77%

Table 5.8 - HAART test based on the extensions MMHH and TSMHH, using SVM, RF and KNN

Accuracy	SVM RBF	SVM Poly	RF 250	KNN 5
MMHH_0.5	65.16%	66.79%	53.02%	55.94%
MMHH_1	70.34%	**72.19%**	57.11%	58.04%
MMHH_1.5	**70.62%**	71.88%	56.35%	57.94%
MMHH_2	69.40%	70.58%	60.01%	57.88%
MMHH_3	68.80%	69.65%	**63.12%**	**60.18%**
TSMHH_0.5	67.39%	67.70%	55.49%	**59.37%**
TSMHH_1	**72.02%**	72.92%	55.18%	54.36%
TSMHH_1.5	71.80%	72.91%	55.50%	54.45%
TSMHH_2	71.67%	**73.50%**	55.73%	53.06%
TSMHH_3	69.73%	71.71%	**58.09%**	58.59%

Table 5.7 shows the performance of the MHH feature. The SVM techniques produce a higher accuracy than the KNN and RF techniques. The best accuracy of 63.19% was achieved using SVM RBF and

MHH with a threshold of 1.5, with SVM Poly 1.62% less using the same MHH feature. Table 5.8 demonstrates the performance of the MMHH and TSMHH extensions. For MMHH, the best accuracy of 72.19% is achieved using a threshold of $T = 1.5$, along with the SVM Poly technique. Both MMHH and TSMHH have performed better than MHH, with TSMHH performing slightly better than MMHH (1.31%). Table 5.9 contains the results for the spatially pooled features SMMHH and ASMMHH. ASMMHH with a threshold of 1, using SVM Poly has achieved a performance of 76.52%. SMMHH has produced a best of 75.57%, a difference of 0.95%. ASMMHH has the best performance of all the extensions and MHH. The summarising results of this experiment can be seen in Table 5.10. There is a clear performance gap between MHH and the extensions, with ASMMHH using SVM Poly having a 12.85% increase over MHH. LBP-TOP is also tested for comparison, which performed better than MHH, MMHH and TSMHH. However, the introduction of spatial pooling has improved the performance enough to go 3.6% above LBP-TOP.

Table 5.9 - HAART test based on the SMMHH and ASMMHH descriptor, using SVM, RF and KNN

Accuracy	SVM RBF	SVM Poly	RF 250	KNN 5
SMMHH_0.5	70.39%	70.68%	56.40%	58.59%
SMMHH_1	73.70%	**75.57%**	59.47%	60.90%
SMMHH_1.5	73.61%	75.50%	59.53%	60.54%
SMMHH_2	**74.41%**	75.10%	63.50%	61.25%
SMMHH_3	72.84%	74.31%	**67.12%**	**61.93%**
ASMMHH_0.5	**75.48%**	76.02%	63.36%	62.64%
ASMMHH_1	75.39%	**76.52%**	62.35%	62.30%
ASMMHH_1.5	75.04%	76.21%	62.18%	**62.83%**
ASMMHH_2	74.21%	74.60%	64.90%	61.21%
ASMMHH_3	71.56%	73.31%	**66.84%**	62.27%

Table 5.10 - Summary of the best results for MHH and the proposed extensions. LBP-TOP is included for comparison.

Accuracy	SVM RBF	SVM Poly
MHH	63.19%	61.57%
MMHH	70.62%	71.88%
TSMHH	72.02%	72.92%
SMMHH	74.41%	75.10%
ASMMHH	**75.48%**	**76.02%**
LBP-TOP	72.29%	72.42%

5.6.3.2 Social Touch Gesture Recognition based on CoST dataset

The CoST dataset is similar to HAART but has more than double the gestures to classify making the task a lot more challenging. It is also based on an 8×8 grid of pressure sensors, containing a similar data format. The framework for the system is the same as for HAART, making the task straightforward. The CoST dataset contains 3524 training samples based on 14 gestures. The considerable number of classes and data can cause a significant increase in the training time for a system. Therefore, this experiment will adopt a different testing protocol, where the data is split 70% for training and 30% for testing. This makes 2685 training samples and 839 test samples. The test is only run once, and the split will remain the same for each feature.

The order of feature testing will remain the same as the HAART tests, which is MHH, MMHH, TSMHH, SMMHH, and ASMMHH. The threshold values will range from $T = [10, 15, 20, 25, 35]$. They are higher values than those used in the HAART because of the characteristics of the sensors, gesture sensitivity and the recording settings. The results for each feature below will represent the best produced from all the thresholds. Both databases contain similar gesture classes, with CoST having 7 additional gestures. This would significantly increase the training time required at the machine learning stage. The LBP-TOP feature is also extracted and tested for comparison.

Table 5.11 - Results table of the CoST dataset, using the MHH descriptor and its extensions.

Accuracy	**SVM RBF**	**SVM Poly**	**RF 500**	**KNN 5**
MHH	43.50%	33.49%	41.59%	37.90%
MMHH	50.65%	41.00%	45.17%	41.83%
TSMHH	43.86%	38.14%	41.95%	40.16%
SMMHH	**52.44%**	42.90%	**47.19%**	43.14%
ASMMHH	51.72%	**43.14%**	46.96%	**44.45%**
LBP-TOP	41.95%	41.95%	37.42%	24.55%

Table 5.11 shows the results of all the descriptors including LBP-TOP. SMMHH has produced the best performance of 52.44% using SVM RBF, with the confusion matrix shown in Table 5.12. The LBP-TOP feature did not produce satisfactory results. In fact, it was the worse feature of the bunch. Both Spatial Pooling extensions show an impressive performance, followed by MMHH, then TSMHH and MHH.

The confusion matrix in Table 5.12 indicates that Grab was the easiest gesture to recognise (88.3%), closely followed by slap (81.6%). The worst gesture was Tap (23.3%) which had been mistaken as Pat 31.6%, Hit 13.3% and Slap 10.0% of the time. This is understandable as the gestures are of similar nature, that has short durations with fast actions.

5.6.3.3 Experiment 5B Highlights

The experiments on both CoST and HAART databases have shown that the MHH extensions have shown improvement over the MHH feature in all cases. SVM has once again outperformed KNN and RF for the machine learning technique. However, RF had been better than SVM Poly on the CoST database, by ~4%. SVM with RBF kernel performed well on the CoST dataset, whereas the Poly kernel performed well on the HAART dataset. This may indicate that the Poly kernel does not work well with more classes and that the RBF kernel does better.

The Spatial pooling extensions have proven to be the best descriptor for touch gesture recognition. ASMMHH proved to be the best on the HAART tests, and SMMHH for the CoST tests. Both features performed better than the popular LBP-TOP feature on both databases, demonstrating the benefits of motion detection for touch gesture applications. The Challenge itself also demonstrated the performance of SMMHH in [161], showing the best individual performance amongst a set of other descriptors that include: Statistical Distribution (SD) of the surface; Binary Motion History (BMH); Motion Statistical Distribution (MSD); and LBP-TOP. The machine learning approach used was Random Forest and Boosting algorithms.

Table 5.12 - Confusion Matrix of the SMMHH feature producing 52.44% using SVM with RBF kernel.

		Predicted Label													
	Accuracy (%)	**Grab**	**Hit**	**Massage**	**Pat**	**Pinch**	**Poke**	**Press**	**Rub**	**Scratch**	**Slap**	**Squeeze**	**Stroke**	**Tap**	**Tickle**
Actual Labels	**Grab**	**88.3**	0	1.6	0	0	0	0	0	0	0	10.0	0	0	0
	Hit	0	**58.3**	0	6.6	0	3.3	3.3	0	0	25.0	0	0	3.3	0
	Massage	3.3	0	**65.0**	0	0	0	1.6	5.0	5.0	0	16.6	3.3	0	0
	Pat	0	3.3	0	**45.0**	0	1.6	0	5.0	5.0	21.6	0	5.0	3.3	10.0
	Pinch	5	0	6.6	3.3	**40.0**	16.6	3.3	1.6	0	0	23.3	0	0	0
	Poke	0	10.0	0	8.3	13.3	**55.5**	6.6	0	0	0	0	0	5.0	1.6
	Press	10.0	3.3	0	3.3	6.6	3.3	**45.0**	0	3.3	1.6	21.6	0	0	1.6
	Rub	1.6	1.6	13.3	5.0	1.6	0	0	**30.0**	16.6	3.3	3.3	20.0	0	3.3
	Scratch	0	0	1.6	0	1.6	0	0	10.0	**58.3**	1.6	1.6	13.3	0	11.6
	Slap	1.6	15.0	0	1.6	0	0	0	0	0	**81.6**	0	0	0	0
	Squeeze	46.6	0	8.3	0	5	0	0	0	0	0	**40.0**	0	0	0
	Stroke	0	0	3.3	0	0	0	0	31.6	1.6	3.3	0	**58.3**	0	0
	Tap	0	13.3	0	31.6	6.6	6.6	3.3	0	1.6	10.0	0	0	**23.3**	3.3
	Tickle	0	0	3.3	0	3.3	0	0	11.6	26.6	0	0	10.0	0	**45.0**

Table 5.13 shows the results based on a slightly different protocol to Table 5.10 but includes and compares the SMMHH descriptor with the others. There is a 10.84% increase in accuracy over MSD, the next highest accuracy recorded using RF. The combined performance is an aggregation of the statistical feature sets to provide a further jump in the accuracy of 10.99% using RF. This shows how different mathematical models can complement each other to provide further improvement, even if BMH has only 22.67% accuracy. The result of the combined model on the testing partition shows a slight decrease over the training partition but still shows a high accuracy overall.

Table 5.14 is the equivalent experiment for CoST in the Challenge. It presents a similar trend to Table 5.13, with SMMHH having the best individual performance, and the combined model increasing the accuracy further. The testing partition also shows a decrease in accuracy but provides a higher accuracy than any individual feature in the training partition. An interesting find is how the confusion matrix as Table 9 [161] closely matches the confusion matrix in Table 5.12. There are similarities in confusion between the Stroke, Slap, Tap and Tickle gestures that indicates an area that can be further worked on to distinguish between the gestures.

Table 5.13 - Published Challenge results on the HAART database of a group of feature sets including SMMHH, the best individual performer.

Data Set	Feature Set	Random Forest	Boosting
Training	SD	36.17%	33.50%
	BMH	22.67%	22.34%
	MSD	54.82%	53.61%
	SMMHH	**65.66%**	**60.84%**
	LBP-TOP	53.01%	54.82%
	Combined	**76.65%**	**77.67%**
Testing	Combined	66.53%	65.54%

Table 5.14 - Published Challenge results on the CoST database of a group of feature sets including SMMHH, the best individual performer.

Data Set	Feature Set	Random Forest	Boosting
Training	SD	41.24%	41.31%
	BMH	27.55%	28.88%
	MSD	44.82%	44.93%
	SMMHH	**52.68%**	**52.56%**
	LBP-TOP	45.65%	46.36%
	Combined	**64.52%**	**64.44%**
Testing	Combined	58.67%	58.19%

The outcome of this experiment was successful and contributed to the Challenge [159] proposed for the databases. Once again, the combination of Spatial pooling and Multi-scale extensions produced the best performance for both datasets. Both Average pooling and Max pooling performed closely to each other, with ASMMHH the best on HAART, and SMMHH for CoST. When compared to the MHH descriptor, there was a 14.45% increase on the HAART database and an 8.94% increase on the CoST database. These are considerably high jumps in accuracy, especially for a task with 14 classes. When compared to other techniques such as LBP-TOP; MSD; BMH; and SD, the increase is a more on the HAART database, and significantly more on the CoST database, showing a higher efficiency for a more complicated task.

Feature fusion of these feature sets further increases the accuracy by a big margin [161]. This idea was observed from the findings in Chapter 3 and applied successfully, demonstrating the capabilities of fusing feature sets that are products of different modalities. Machine learning techniques can obtain the valuable information from each of these feature sets to give an aggregated prediction. These will most often be better than individual predictions by the same set of features.

5.6.4 Settings and Protocols for Emotion Application

So far, the proposed extensions have shown impressive performances for motion intensive applications. This idea was then moved towards emotional data, to see if motion information can be encoded from dynamic emotional data effectively. The application area which was considered for this was Depression analysis. It was chosen for the varying speeds of motion, specifically the slow changing expressions from people suffering from depression.

The AVEC2014 guidelines are followed from [13], based on the global challenge that had taken place in 2014. In this, the data had been split into 3 partitions, which are training, development and testing. Each contained 50 video samples from the "Northwind" and the corresponding "Freeform" tasks, totalling to 100 video clips per partition. The labels provided are based on 1 real value for each pair of tasks, with the idea that the depression scale should be the same for both, as each subject has conducted both tasks with the same mental state. The evaluation metric consists of the RMSE and MAE.

Each system is trained with the training partition data, and then evaluated on the development partition data. Once the results are good enough, the Testing partition is evaluated to give a final judgement of the system. This experiment was run after the challenge occurred due to the limitation in submissions, it would not be possible to get a result for each MHH extension. Therefore, the results are not taken from the works in [83].

At the time of the global challenge, the testing labels were not made available to the public, showing the initial experiments [83] limited, as there were only 5 submissions allowed. Since the challenge has finished, the test labels have been made available to the public, in which further optimisation can be allowed beyond 5 attempts. This has allowed for testing the MHH extensions in detail with the development partition to thoroughly evaluate their performance.

5.6.5 Experiment 5C – Depression Analysis based on Temporal Data

The proposed MHH extensions will be tested on a depression database to further demonstrate the techniques on an uncommon task, that also happens to be a regression problem. The framework is slightly different to the previous experiments in the machine learning stage. The audio data is not used in this experiment as the focus is mainly on the MHH extensions performances. Each extension is applied to the raw video to produce its respective motion frame histograms. Local descriptors will be used to extract information from these motion frames to produce a single vector per visual sample. Partial Least squares and Linear Regression are then tested individually and fused together to give a combined prediction of depression scale. The MAE and RMSE are used to evaluate the performance of the system for all sessions. The initial tests will be run on the development partition, and the best configuration for each descriptor will be run on the test partition.

Each frame dimension is 640×480 pixels, which will be the motion frames for each descriptor. The local features are extracted from each motion descriptor and are denoted as their respective feature name, followed an underscore and by the extracted local descriptor. E.g. for MHH, the denotation will be MHH_LBP, MHH_EOH and MHH_LPQ. The threshold values for each test are $T = [0.5, 1, 2]$, with the pattern size set to $M = 5$. These are chosen by observing the MHH feature captured from random samples to see the depth of the motion captured.

Table 5.15 - List of feature size for MHH and each extension, before and after applying local descriptors

Feature	Resulting Feature Size	After EOH	After LBP	After LPQ
MHH	$640 \times 480 \times 5$	$384 \times 5 = 1{,}920$	$256 \times 5 = 4720$	$256 \times 5 = 1{,}280$
MMHH	$640 \times 480 \times 5 \times 5$	$384 \times 25 = 9{,}600$	$256 \times 25 = 6{,}400$	$256 \times 25 = 6{,}400$
TSMHH	$640 \times 480 \times 5 \times 5$	$384 \times 25 = 9{,}600$	$256 \times 25 = 6{,}400$	$256 \times 25 = 6{,}400$
SMMHH	$\sum_{i=0}^{4}\left(\frac{1}{2^2 \cdot i}(640 \times 480 \times 6 \times 5)\right)$	$384 \times 120 = 46{,}080$	$256 \times 120 = 30{,}720$	$256 \times 120 = 30{,}720$
ASMMHH	$\sum_{i=0}^{4}\left(\frac{1}{2^2 \cdot i}(640 \times 480 \times 6 \times 5)\right)$	$384 \times 120 = 46{,}080$	$256 \times 120 = 30{,}720$	$256 \times 120 = 30{,}720$

The threshold values are set low compared to experiment 5A because there are very subtle movements in these videos as they are based on a person talking and showing slow emotions. For the MMHH feature, the scale parameter is set to $S = 5$, capturing motions from scales 1 to 5. The TSMHH feature is adjusted to capture motion based on 5 time scales, which are $[0.5, 0.75, 1, 2, Fr]$. Finally, the SMMHH feature is captured using both mean and max pooling. The number of pooling stages is set to

4. Each of the resulting pattern histograms will have the 3 local descriptors of EOH, LBP, and LPQ extracted from them, which will reduce the dimensionality significantly. The dimension size for each feature vector can be seen in Table 5.15. It is determined by the number of motion frames times the local descriptor size. Once the features are extracted, PCA is applied to reduce the feature dimensionality to 35 components, as PLS regression requires the number of dimensions to be less than the number of samples. These features are rank normalised between 0 and 1, where PLS and LR are then used to predict the depression scale for a given sample.

Table 5.16 - Performance on the development partition using MHH and the proposed extensions, that have LBP applied on the motion frames. All 3 thresholds are tested for each feature to find the best combination that produces the lowest error.

LBP	**PLS**		**LR**		**PLS + LR**	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
MHH_0.5	**9.83**	**7.70**	10.07	7.88	**9.91**	7.74
MHH_1	10.14	7.85	**10.01**	**7.71**	10.01	**7.72**
MHH_2	11.82	9.32	11.41	8.84	11.60	9.04
MMHH_0.5	10.36	**7.44**	10.29	7.64	10.34	7.66
MMHH_1	**10.03**	7.61	**9.86**	**7.49**	**9.91**	**7.54**
MMHH_2	11.37	9.07	11.12	8.65	11.20	8.82
TSMHH_0.5	10.09	**7.81**	9.97	7.84	9.96	7.84
TSMHH_1	**9.88**	7.82	**9.67**	**7.53**	**9.76**	**7.64**
TSMHH_2	10.49	8.58	10.30	8.21	10.28	8.16
SMMHH_0.5	9.85	**7.65**	9.79	**7.56**	9.86	**7.58**
SMMHH_1	**9.63**	7.78	**9.57**	7.73	**9.61**	7.82
SMMHH_2	10.13	8.31	10.12	8.19	10.13	8.20
ASMMHH_0.5	9.86	**7.62**	10.02	**7.65**	9.93	**7.56**
ASMMHH_1	**9.63**	7.82	**9.63**	7.86	**9.65**	7.92
ASMMHH_2	10.12	8.40	10.16	8.36	10.16	8.40

5.6.5.1 Threshold Test

The initial test will be to determine what is the best threshold to use for each motion feature. This is a crucial step as it gives an indication of which speed of motion and sensitivity is best observed with each descriptor. The LBP descriptor will be used for this test, along with the MHH descriptor and the proposed extensions. This test will follow the framework and the performance will be based on the development partition, with the training partition used to train the system. PLS and LR will be tested individually, and then fused together using linear weighted fusion.

The initial tests results are shown in Table 5.16, which indicated that the threshold values 0.5 & 1 produce the best performances, and threshold value 2 the worst in every test. SMMHH_LBP with a threshold of 1 has produced the lowest RMSE value of 9.57, with MMHH_LBP with a threshold of 0.5 has produced the lowest MAE of 7.44. The following tests will have both thresholds of 0.5 & 1 tested, and the best out of both will be chosen.

Table 5.17 - Performance on the Development partition using MHH and the proposed extensions, that have EOH applied on the motion frames. Thresholds 0.5 and 1 are tested for each feature, with the lowest error of the 2 chosen as the performance of the feature.

EOH	**PLS**		**LR**		**PLS + LR**	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
MHH	10.67	8.62	10.72	8.73	10.89	8.88
MMHH	10.68	8.73	10.78	8.87	10.87	8.98
TSMHH	**9.79**	**7.79**	**9.70**	**7.62**	**9.81**	**7.68**
SMMHH	10.71	8.84	10.80	9.15	10.84	9.16
ASMMHH	10.55	8.70	10.58	8.77	10.65	8.82

Table 5.18 - Performance of LPQ descriptor on the Development partition, captured from the motion frames from each MHH descriptor and its Extensions.

LPQ	**PLS**		**LR**		**PLS + LR**	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
MHH	10.34	8.23	10.39	8.17	10.39	8.12
MMHH	10.65	7.83	10.25	**7.71**	10.29	**7.64**
TSMHH	10.52	8.55	10.64	8.54	10.59	8.48
SMMHH	9.79	7.72	9.92	7.74	9.87	7.78
ASMMHH	**9.77**	**7.72**	**9.79**	7.79	**9.71**	7.70

Table 5.17 & Table 5.18 demonstrate the performance when using the EOH and LPQ local descriptor on the motion frames. The EOH descriptor has shown to work best with the TSMMHH motion descriptor, using LR to produce the lowest RMSE of 9.90 and MAE of 7.62. LPQ is best paired with the ASMMHH motion descriptor, producing the lowest RMSE of 9.71 with MAE of 7.70 using the fusion of PLS and LR.

Between all 3 local descriptors, SMMHH_LBP has produced the lowest RMSE and MAE score. EOH has produced the second lowest with a difference of 0.13 RMSE and 0.09 MAE, and ASMMHH_LPQ the third lowest with a difference of 0.01 RMSE and 0.08 MAE against TSMMHH_EOH. The next step is to fuse the 3 feature descriptors together to see their performance.

Table 5.19 - Combined feature performances on the Development partition, using the LBP, LPQ and EOH descriptor that are applied on the motion frames of MHH and its extensions.

Comb	**PLS**		**LR**		**PLS + LR**	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
MHH	9.93	7.92	10.05	7.95	9.97	7.90
MMHH	10.16	8.53	9.98	8.41	10.15	8.52
TSMHH	**9.59**	**7.39**	**9.55**	**7.49**	**9.52**	**7.37**
SMMHH	9.82	7.80	10.12	8.22	9.97	8.00
ASMMHH	9.76	7.77	10.01	8.18	9.69	8.08

Table 5.19 applies feature fusion in terms of the concatenation of all 3 local descriptors extracted from each motion frame. The error has dropped to the lowest recorded using the TSMHH_Comb descriptor and the fusion of PLS with LR, with an RMSE of 9.52 and MAE of 7.37. This is a difference of 0.07 RMSE and 0.36 MAE against the best individual feature SMMHH_LBP using LR.

Table 5.20 - Performance of the Combined features using MHH and all the extensions on the Test partition.

Test	**PLS**		**LR**		**PLS + LR**	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
MHH	9.83	8.00	10.05	8.21	9.73	7.88
MMHH	9.80	7.67	10.07	8.08	9.82	7.68
TSMHH	**9.39**	7.80	**9.40**	**7.70**	**9.32**	**7.62**
SMMHH	9.96	**7.63**	9.91	7.99	9.91	7.84
ASMMHH	9.97	7.75	9.92	8.01	9.89	7.74

Table 5.20 shows the performance of the Combined local features from MHH and all the extensions on the Test partition. The performance is similar to the development partition performance, but with a better RMSE of 9.32, and a slightly worse MAE of 7.62 using the TSMHH feature with PLS and LR fusion. This indicates that the feature is robust and not biased towards the Development partition data.

5.6.5.2 *Experiment 5C Highlights*

This experiment has shown how the Motion descriptors paired with hand-crafted descriptors perform for Depression analysis that is based on a regression task. There are promising results that indicate that the extensions provide a better motion description than the original MHH. The baseline feature performance is based on the LGBP-TOP feature, which has had a worse performance than the all of our tested motion descriptors.

Table 5.21 compares the TSMHH performance against others that participated in the AVEC14 challenge. The main objective here is to observe the performances that only use the visual data, for a fair comparison. Only Dhall et al. [162] managed to get a performance better on the Test partition using only visual data, with an improvement of 0.29 for RMSE and 0.54 MAE. They first used face processing techniques to detect and track the face along the sequence. Then they extracted Block-wise LBP-TOP on sub-sequences and used Fisher Vector encoding with Bag of Words. Williamson et al. [163] who has the state-of-the-art performance uses vocal tract timing from both visual and audio data to achieve a low RMSE and MAE of 8.12 and 6.31.

Table 5.21 - Comparison of TSMHH against the baseline performance, other techniques using only the Visual modality (image frames). Additionally, the state-of-the-art that uses the audio and visual modalities. Performance is based on the Test partition data.

Methods	Modality	RMSE	MAE
Perez et al. [164]	Visual	11.91	9.35
Baseline [13]	Visual	10.86	8.86
Jain et al. [165]	Visual	10.24	8.39
Kaya et al. [166]	Visual	9.61	7.69
TSMHH	**Visual**	**9.32**	**7.62**
Dhall et al. [162]	Visual	8.91	7.08
Williamson et al. [163]	**Audio + Visual**	**8.12**	**6.31**

TSMHH performs better than the rest of the works from the Challenge, demonstrating the ability of the descriptor. A probable reason for its superior performance over the other MHH extensions could be because the visual data contains emotions that change at a slow pace. TSMHH can take advantage of this situation by capturing motions at faster speeds that can interpret the relatively slow-paced emotion to fast-paced emotion.

5.7 Evaluation and Discussion

The initial experiment (5A) on human action recognition showed competitive performance against other spatiotemporal techniques like LBP-TOP and HOG3D. The highlight performance was by the ASMMHH descriptor, that using the pooling and multi-scale approach to provide a descriptive and robust feature. The pooling concept for this application showed to be more effective because human actions are very visually descriptive with a lot of motion. Pooling reduces the noisy information by reducing the spatial dimensions, making the most moving visual parts stand out more. This could be in form of having sharper curves of the body parts, as they can be identified easier in a smaller resolution being less sparse, making the motion more relative to that body part. In this case, average pooling had performed better. This may be because the detailed information is retained better as max pooling would saturate the pixel values.

The next experiment (5B) consisted of touch gesture recognition, which was based on human interaction with animals (HAART), and social interaction with humans (CoST). This experiment differed from experiment 5A in the form of data, as it was based on pressure sensors rather than visual data. However, this information was treated as a sequence of 8×8 images. The descriptors do not need to specifically have 8-bit pixel values. Therefore, this was not an issue if a pressure sensor contained a value beyond 255. This data format was chosen to demonstrate the various kinds of motion the MHH extensions can capture, not necessarily just based on the sequential image form.

So far, the proposed extensions have shown impressive performances for motion intensive applications. This idea was then moved towards emotional data, to see if motion information can be encoded from dynamic emotional data effectively. The application that was considered for this was Depression analysis. This was chosen for its varying speeds of motion, with the slow changing expressions from people suffering from depression.

Experiment 5D looked at a different form of emotional data, based on Depression analysis from slow-moving emotions. This task seems challenging at first because the visual data is just based on a person talking, which does not contain as much motion as the previous experiments. However, there are subtle expressions that can be captured by looking at the slow motion that occurs on the face. The expressions for depression will generally have low arousal as the subject’s mood will be sad, showing lack of energetic movement. Considering this, the expressions will be slow paced and consistent throughout the sequence. It is no surprise that the extension that performs the best is TSMHH, as that is specifically designed to capture different speeds of motion. Consequently, it can be adjusted to pick up the slow and sad emotions portrayed by the subjects.

Performance wise, TSMHH has a decrease of 0.41 for RMSE and 0.62 for MAE against MHH. Both features benefitted from using decision level fusion of PLS and LR. The other extensions also performed better than MHH, but by a small amount. This may be because of the application type, and quantity of motion that makes the resulting features from the extensions (apart from TSMHH) similar to the MHH feature. When compared to other frameworks, ours had provided better RMSE and MAE than all the others that only used visual modality, apart from Dhall et al. [162]. This includes the baseline technique that used the LGBP-TOP variation of LBP-TOP. This is a good indication that motion capture can be a useful tool for emotion analysis based on dynamic content.

5.8 Summary

In this chapter, extensions were proposed on the MHH descriptor and thoroughly tested across a variety of applications. These extensions were proposed after analysing the inefficiencies of the MHH descriptor, where each extension tackled a specific issue to make a more robust motion capturing algorithm that can be applied to applications with different forms of motion. The first part of the chapter introduced the proposed extensions to the MHH algorithm, which were demonstrated on human action recognition and touch gesture recognition applications as they contain a lot of motion in different forms. This was followed by demonstrating how the extensions would work on emotion data, through the application of Depression analysis. From the experiments, the following was understood:

- All the proposed extensions provide a benefit over the MHH descriptor, which is observed by the performances across the various applications.
- The Spatial Pooling combined with Multi-scale extension provided the best performance for the motion intensive applications.
- Time-Scale MHH worked best with Depression analysis. This is concluded by the ability to control the speed of motion captured. As the emotions display with depression can be slowly changing, TSMHH can be adjusted to capture them effectively.
- Competitive performances were achieved across all applications against popular techniques and frameworks.

Chapter 6. Deep Learning for Dynamic Facial Expression Recognition and its Applications

In Chapter 5, extensions on the MHH motion descriptor were proposed and demonstrated good performances on various applications. Those ideas of capturing motion from the image sequences are limited to the visual changes. To try and remove this limitation, this chapter investigates the motion that occurs in the deep feature space of those raw images.

A new algorithm is developed called Feature Dynamic History Histogram (FDHH) and is utilised on top of the deep features to capture the dynamics across the deep feature space. The idea is to show ways of combining efforts from deep learning with the dynamics found across a feature sequence. State-of-the-art results are achieved on the AVEC2014 database, demonstrating the performance of the proposed FDHH algorithm.

Experiments are carried out to observe the affective state of people under certain conditions. This includes observing continuous emotions and induced emotions, which is achieved by modelling temporal data to the emotions. The data is based on Depression analysis using the AVEC2014 dataset and predicting emotional impact of movies through induced emotions using the LIRIS-ACCEDE database. Both AVEC2014 and LIRIS-ACCEDE tasks are based on a regression, to predict the BDI-II scale for depression, and the Arousal and Valence of the induced emotion from movies. These are also demonstrated in the publications of [167], [168].

6.1 Introduction

Discrete and continuous emotions are explored from temporal data for an understanding of how the dynamic information can be utilised to capture these emotions effectively. The data contains richer information than static images. For emotions, capturing the early state of an expression in a still image may be interpreted as a completely different expression. This would be easier to determine with temporal data, as the expression would unfold over time.

A lot of research has moved towards temporal data such as video because the basic PCs performance has continuously increased to become more powerful. Because of this, a higher quality of data is available allowing for research and applications to go beyond the scope of still images. The data and media content generated has become so large that processing each sample manually has become inefficient and tedious. Tasks can be made automated to speed up this process. However, some tasks require intelligence to succeed automatically, such as an example to automatically group video clips in terms of content/genre.

Emotions have also been considered in this area, with a recent task requiring the suggestion of induced emotion on a person based on viewing certain clips of videos. This is different to the task of determining a person's emotional state in a video, as it is the person viewing the video of whom the emotional state is required. This was proposed in the MediaEval16 challenge that had 2 main tasks. First was to suggest the induced emotion on a global level, which was to suggest a single value of Arousal and Valence for over 9000 short samples of sequences (< 30s). The second was to suggest a continuous Arousal and Valence value of induced emotion per frame of a longer sequence (> 60s). Depression is another form of emotion that has an affective state which can be modelled. It can also be determined by the affective dimensions of Arousal and Valence. These emotions come in different dimensional forms and can be interpreted in many ways. Finding the affective state of a person has been studied significantly, to provide a detailed and accurate measurement compared to discrete emotions. Affective dimensions mainly include Arousal and Valence. When presented together, it can measure the many different emotions across a 2-dimensional space.

Emotion recognition and all its applications can benefit from the capabilities of deep learning. The concept of deep learning has been compared to a black box idea, as it gives the ability to adapt to any situation the best it can, where the internals adjust themselves to the data given. Long-short term memory (LSTM) is a form of deep learning that specifically works with time series data, to try and create a connection between the past and current features. However, using image frames as a direct input can be too large for the network. Therefore, it is common to use some form of descriptive algorithm to summarise the image into a bunch of features.

Convolutional Neural Networks are designed to learn spatial data such as images, which can be paired with LSTMs to create a 2-stage framework that handles temporal information at the frame level. There have been some attempts to try and create 3D CNNs [169] to work directly on temporal data, by training on cubes of data from consecutive frames. This has also been recently attempted for dynamic facial expression recognition [170], where it has shown to be affective in capturing changes of facial parts throughout a sequence. However, with this technique, is hard to understand how it can model temporal information as it works just works on small spatial dimensions. It does not consider the enough of the sequence as one instance to fully understand events that can occur over time. CNNs generally look at whole images at a time to sweep across it and try learning any local characteristics. This concept is lost at the local level as it requires several iterations to process a whole sequence.

Spatial algorithms can also be exploited for temporal content, by looking at the data at frame level and stacking the feature vectors for each into a feature matrix. This is known as a spatiotemporal approach, which gives the benefit of allowing a large amount of research to be compatible with this kind of data structure. However, sometimes it is not enough to just apply a spatial feature extractor on all the data frames, as in most cases, this results in a large vector for one data sample. Therefore, it is almost required to run a second technique on top of the extracted spatial features to describe them all into one feature vector. Common techniques include simply as taking the mean of all the frames or other statistical approaches. This may solve the problem of a large feature, but it will not do efficiently as a lot of valuable information is lost throughout the sequence.

Our approach is to try and understand a temporal sequence by treating it as a feature vector sequence. Most of the research that works with temporal data generally apply temporal algorithms directly on the raw images. We adopt a different approach by looking at the mathematical representations of images from a sequence and then apply algorithms that work with the whole sequence of representations.

The objectives of this chapter are to design a complete framework that can handle dynamic content at the frame level. This should produce a solution that incorporates deep learning techniques to produce high-quality feature representations. A descriptor will be developed to work on these feature representations, that can observe variations occurring dynamically across the feature sequence. Advanced regression techniques will be investigated to improve the system analysis and prediction.

6.2 Related Works and Problems

Depression can be expressed through body language, emotions and expressions. The early work for Depression analysis started with Cohn et al. [171] and McIntyre et al. [172]. Cohn et al. used face analysis techniques to try and understand the movement that occurs across the face. They used Active Appearance Model along with FACS to track the facial movements. From their findings, they concluded that it is possible to develop a system for Depression analysis. Since then, there have been several challenges to encourage further research for depression through the AVEC series [12], [13], [18].

The AVEC challenges have gained a lot of attention for their efforts to detect signs of MDD, as this disorder continually increases around the world. The depression recognition sub-challenge of AVEC2013 [12] and AVEC2014 [13] has had many participants contribute to finding a solution. Some of the proposed methods [83], [99], [163]–[166], [173], [174] demonstrate a strong link of depression with the visual and vocal cues of patients. AVEC2016 [18] is the most recent which also contributed to further research ideas in [175]–[177], but it has a slight change in its depression task. The data provided in the challenge are pre-computed sets of features that consist of: HOG features on the aligned face; 3D head poses position and orientation; gaze direction estimate for both eyes; 2D and 3D facial landmarks; and emotion-based measures. The Audio features they provide are also pre-computed. Unfortunately, the raw data is not made available for ethical reasons, which is why the database will not be utilised in the upcoming experimentation. The techniques for AVEC2014 will be briefly mentioned and used as a benchmark for the upcoming experiments.

Williamson et al. [163] were the most successful for Depression analysis. They won the depression sub-challenge for AVEC2013 and AVEC2014. They focused on the timing of movement in the vocal tract in 2013 and included the coordination between the vocal tract and facial gestures in 2014 to determine the level of depression. Recently [163], they extracted low-level features from the voice, speech and facial data information. With the voice data, they look for a set of features like Harmonics-to-noise ratio and Cepstral peak prominence. For the speech system, they look for Dormant frequencies and Mel frequency cepstral coefficients. Facial action units are captured across the face to identify the changes in facial expression. The timing for all the low-level features is coordinated and characterised to correlate them and create high-level features. For the feature learning, they design a novel Gaussian mixture model (GMM)-based multivariate regression scheme that is referred to as Gaussian Staircase Regression. Extreme learning machines (ELMs) are also used to predict the depression level.

The works by Meng et al. [82] is based on the AVEC2013 dataset, which involved making a framework to model the visual and vocal cues. These cues are branched off to be processed separately, and then later joined at the decision level. For the visual data, the dynamic motion is captured using the MHH algorithm on the visual data. Furthermore, the MHH features are treated as image frames, where the LBP and EOH local descriptors are applied to produce the two features. These features are concatenated together to produce the final visual feature.

For the vocal branch, the baseline Low-Level Descriptors (LLD) feature provided by the host is utilised. They consist of 2268 features, composed of 32 energy and spectral related with 42 functionals; 6 voicing related LLD with 32 functionals; 32 delta coefficients of the energy/spectral LLD with19 functionals; 6 delta coefficients of the voicing related LLD with 19 functionals; and 10 voiced/unvoiced durational features. These features are then considered as a sequence in which MHH can be applied on to. For each branch, PLS regression is adopted to predict the depression scales. Decision level fusion is applied to combine the predictions of both branches.

Gupta et al. [99] add on to the visual and vocal modalities, by including the linguistic modality, from which they generate lexicon from the speech transcripts. For the visual features, they use the baseline LGBP-TOP features provided by the organisers. In addition, LBP features are computed at the frame level, and LBP-TOP on the temporal sequence. They also consider motion features, via optical flow. Key points are detected using a corner detection algorithm. The motion is computed by summing how much each key point has moved across the frames. Facial Landmarks are their final visual feature, applied to the frame level, generating 66 landmarks of the facial parts.

The vocal features used by Gupta et al. [99] consist of the baseline features, to produce an acoustic feature representation by combining the speech streams. These streams include spectral shape; spectro-temporal modulations; periodicity structure due to the presence of pitch harmonics; and the long-term spectral variability profile. These are stacked together and PCA is applied to reduce the dimensionality whilst keeping 90% of the variance. Perez et al. [164] designed 3 models that are based on audio data, speech from video and the silence from the video. The silence was segmented using the Voice Activity Detection information provided by the organisers. The visual information was extracted based on the voice and silence segments. They used the WEKA tool for applying the Relief feature selection to select the best features from each modality and used a meta-model for fusion.

Kaya et al.[166] used the baseline features of LGBP-TOP and LPQ features on the visual data. They observed the inner face with the LPQ descriptor and the facial regions with the LGBP-TOP descriptor. Then they focused on regression-based machine learning of each feature using Canonical Correlation Analysis (CCA), creating an ensemble for their depression score prediction. Jain et al. [165] extracted LBP-TOP features on the video samples and captured Dense Trajectories of motion captured by optical flow. These trajectories have Histogram of Oriented Gradients, Histogram of Optical Flow (HOF) and Motion Binary Histogram (MBH) extracted from them. LLD Audio features are also used in the system and all of the features are encoded using Fisher Vectors. Linear SVR is used to learn the concatenated feature vector and predict the depression scales.

All of the mentioned techniques consider visual information based on hand-crafted features and algorithms designed to specifically look for certain changes, edges, surface changes and texture information. This is an area that can go further by introducing deep learning to try and capture intelligent characteristics from the texture information that can be directly related to emotions and facial expressions. Not having fixed algorithms provides opportunities to obtain features that may not be considered when designing a hand-crafted technique. With deep learning, there are still some decisions required to design the network architecture and how it learns from data. However, to control what specific characteristics to find in an image is not possible as it has elements of randomness in the process.

Recently, deep learning techniques have been considered for Depression analysis. Chao et al. [174] look to obtain features from three modalities, which are the Audio information, Face appearance information and Face shape information. From the Audio information, a set of audio features are extracted using the YAAFE toolbox [178], some of which are: MFCC, LPC and ZCR. With the Face appearance information, they use a pre-trained network to extract the features at the ReLU hidden layer. The pre-trained network they use has been trained by them on the FER2013 database [179] that contains 35,886 48x48 grayscale images of facial expressions taken from the wild. The Facial shape information is based on 49 facial landmarks across the sequence to capture the head movement and pose.

All of the features are reduced with PCA, and an LSTM neural network is then designed on top of the reduced features, with the first layer of the LSTM being a multimodal fusion layer. They adopt a multi-task learning approach to learn related tasks jointly. They achieved a better RMSE and MAE compared to most of the approaches mentioned earlier. However, the performance is still not close to the state-of-the-art approach produced by Williamson et al. in [163], indicating that there is still room for improvement. Some problems with their approach include:

- Small images used for training the pre-trained network, as this will lose a lot of spatial information that deep networks can learn from.
    - The expressions that are associated with depression are very subtle, as the emotions are slow paced, resulting in small changes across the face. If the dimensions of the image are reduced a lot, these small changes are lost with it.
- Lack of colour images can reduce the quality of the image for a deep network to learn from. The distribution of colours can contribute to a better feature.
- The number of LSTM cells are hard to determine for this application, as the speed of emotion change can be slow.

The most recent work by Zhu et al. [37] also looks into using deep CNNs for modelling facial appearance and facial dynamics and combining both efforts using a pair of joint tuning layers. For their facial appearance CNN, they pre-trained a model based on the GoogleNet [180] architecture, using the public CASIA WebFace database [181] for face recognition. The facial dynamics CNN is trained from scratch, based on motion changes between each frame and the following 10 using optical flow. They propose a good approach for creating a well-trained system for Depression analysis. A few limitations on their idea is that the motion dynamics they capture may not be sufficient for the slow-changing emotion found in depression. The motion would be better captured across the whole sample as each subject's behaviour may be different and moves at different speeds.

When considering emotions induced by video content, there has been a series of challenges by MediaEval [16], [17] that look to capture affective emotions based on a large-scale database [19], [20] of video clips. The challenge in 2015 had shown interesting methods to detect violence and the induced affect by the videos [21]–[24]. The MediaEval 2016 [17] challenge of predicting induced emotions in the following year [25]–[28], which included our contribution in [167].

6.3 Feature Dynamic History Histogram

With the idea of MHH [30] to capture motion across temporal data, the FDHH algorithm has been developed to apply this principle on to the feature space. The feature space is a dimension that is a mathematical representation of a source of data. For humans, it is easier to visualise an image, than a descriptive feature vector of that same image in the feature space. Our minds are not capable of computing the feature descriptor on the image, as well as understanding what we are seeing. However, for computers, this may be a preferred space to get a better understanding of the original image.

It seems that majority of the temporal techniques work directly on the visual data, whether frame by frame sequentially or all at once. Here, we introduce a motion based temporal algorithm that instead of capturing motion across visual space, it captures the variation in the feature dynamics across the feature space. This feature space contains information that has previously been extracted from the raw frame level data, generally of higher quality information, using hand-crafted techniques such as LBP, LPQ and EOH.

Currently, there are several techniques proposed to move these hand-crafted algorithms into the temporal domain, such as LBP-TOP [31], LPQ-TOP [32], HOG3D [155] and LGBP-TOP [86]. They show reliable performance and introduce a way to move the spatial algorithms into the temporal domain. However, these techniques simply apply the hand-crafted techniques in three spatial directions, which is effectively just applying the spatial algorithm multiple times. Therefore, this extends the techniques spatially in different directions rather than dynamically taking the time domain into account.

Instead of looking for a way to extend these algorithms, we propose a solution to use the benefits and qualities from any spatial algorithm with a temporal ideology, by looking for the variation of feature dynamics in a feature sequence, like how motion is captured across an image sequence. For example, if an image sequence contains a person walking, then its LBP equivalent feature sequence will show the mathematical representation of a person walking. The variation across the feature sequence of the person walking can be captured by observing the differences between each feature sequence. From this, pattern occurrences are observed in these variations from which histograms can be created to summarise them.

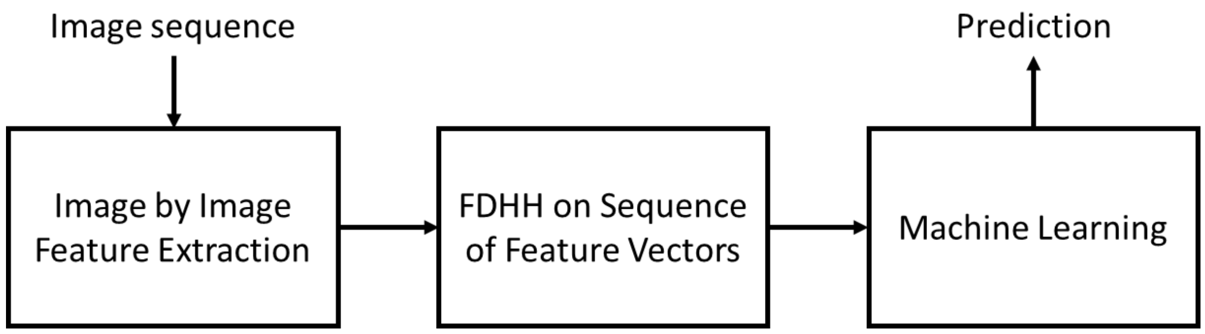


Image sequence
Image by Image Feature Extraction
FDHH on Sequence of Feature Vectors
Machine Learning
Prediction

Figure 6.1 - Block diagram of a given FDHH use-case. Starting with a sequence of images, each image has features extracted to produce a feature vector. These are combined to become a sequence of feature vectors. The FDHH algorithm is then applied onto the sequence of feature vectors to capture the variation within it and is used with a machine learning technique for training and prediction.

Figure 6.1 demonstrated a generic use-case on how the FDHH algorithm can be integrated into a framework designed to learn and predict from temporal data. The FDHH algorithm can be broken down into 3 stages. The initial stage requires the feature vector to be rank normalised. The second stage calculates the difference between each frame sample, and the final stage locates and creates a histogram on all the variation patterns.

The algorithm contains two hyperparameters: Threshold T and number of patterns M. The threshold T controls what intensity of variation needs to occur before it is noted. The patterns M determines what length of variations that occur should be recorded. The input feature matrix $\mathbf{V} \in \mathbb{R}^{C \times N}$, where N is the total number of frame samples and C is the total number of feature components for each frame. The algorithm can be broken down into 3 stages. The first stage prepares the feature matrix by rank normalising the rows and producing a new matrix that contains the absolute difference of one frame sample to its adjacent. Given $\mathbf{x} = \mathbf{V}_{i,1:N}^{\mathrm{T}}$ where $i = 1, \dots, C$, the normalisation of $\mathbf{V}$ is achieved using $f(\mathbf{x})$ in equation 6.1 on each row, where α_{min} and α_{max} are the minimum and maximum of the vector $\mathbf{x}$. The normalised matrix is represented as the $\widehat{\mathbf{V}}$. Normalisation will make determining a threshold value universally the same across various kinds of data. E.g. setting threshold $T = 0.1$ is equivalent to only looking for variations that exceed 10% change of value in either direction within the normalised vector.

$$f(\mathbf{x}) = \frac{\mathbf{x} - \alpha_{min}}{\alpha_{max} - \alpha_{min}} \tag{6.1}$$

$$\widehat{\mathbf{V}} = \begin{bmatrix} f\left(\mathbf{V}_{1,1:N}^{\mathrm{T}}\right)^{\mathrm{T}} \\ \vdots \\ f\left(\mathbf{V}_{C,1:N}^{\mathrm{T}}\right)^{\mathrm{T}} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \left|\widehat{\mathbf{V}}_{1,2} - \widehat{\mathbf{V}}_{1,1}\right| & \cdots & \left|\widehat{\mathbf{V}}_{1,N} - \widehat{\mathbf{V}}_{1,N-1}\right| \\ \vdots & \ddots & \vdots \\ \left|\widehat{\mathbf{V}}_{C,2} - \widehat{\mathbf{V}}_{C,1}\right| & \cdots & \left|\widehat{\mathbf{V}}_{C,N} - \widehat{\mathbf{V}}_{C,N-1}\right| \end{bmatrix} = \begin{bmatrix} b_{1,1} & \cdots & b_{1,j} \\ \vdots & \ddots & \vdots \\ b_{i,1} & \cdots & b_{i,j} \end{bmatrix} \tag{6.2}$$

$$\widehat{\mathbf{B}} = \begin{bmatrix} b_{1,1} \geq T & \cdots & b_{1,j} \geq T \\ \vdots & \ddots & \vdots \\ b_{i,1} \geq T & \cdots & b_{i,j} \geq T \end{bmatrix} \tag{6.3}$$

After the normalisation, the absolute difference is taken between each column and its following. This is represented as $\mathbf{B}$ in equation 6.2, where $i = 1, \dots, C$, and $j = 1, \dots, N-1$. The second stage checks to see if each component of $\mathbf{B}$ is greater than or equal to the threshold T, producing a logical matrix $\widehat{\mathbf{B}}$. These steps are also represented visually in Figure 6.2. Once $\widehat{\mathbf{B}}$ is generated, the next stage is to loop through each row of $\widehat{\mathbf{B}}$ to find patterns of sequential '1's of up to $m = 1, \dots, M$. Given $\mathbf{q}_i = \widehat{\mathbf{B}}_{i,1:N-1}$ is the ith row, where $i = 1\!:\!C$, the patterns $m = 1, \dots, M$ are detected using equation 6.4.

$$z = h(\mathbf{q}_i) \tag{6.4}$$

$$\mathbf{FDHH}_{i,1:M} = z \tag{6.5}$$

where the counter function $h(\mathbf{q}_i)$ that produces the total number of each pattern $(1\!:\!M)$ detected as $z \in \mathbb{R}^{1\times M}$. Once the counter for the row has completed detecting the patterns, the feature histogram row is updated in equation 6.5, where $\mathbf{FDHH} \in \mathbb{R}^{C\times M}$ represents the final feature produced by the algorithm, where all the histogram bins are initialised to '0'. The final stage of the process can be visualised in Figure 6.3, demonstrating how pattern sequences are detected and how the final feature is updated.

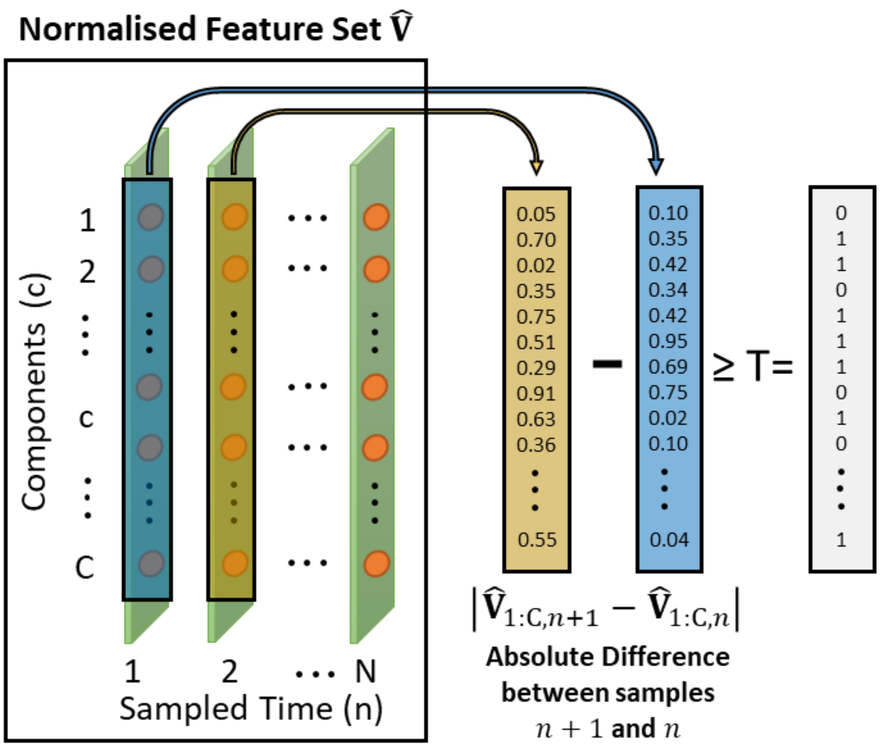


Figure 6.2 - The second stage of the FDHH algorithm, calculating the absolute difference between each sample and its following sample, where the result is compared to a threshold to produce a logical representation of the absolute difference

There are two special cases that can occur during this process. The first is if a detected pattern exceeds M, then the histogram for pattern M is increased, which is z_M. The other case is where consecutive '0's occur. For this case, none of the histograms is increased. In the end, there are M histograms that can be represented as a $C \times M$ Matrix **FDHH**. The full algorithm pseudo code is presented in Figure 6.4, where r represents an internal counter to count the consecutive '1's.

6.4 Proposed Frameworks for Dynamic Emotion Modelling

In this section, frameworks are proposed for Depression analysis and determining Induced emotions from movie clips. The tasks split into two categories, first on Depression analysis and the other on induced emotions, both are using dynamic data. For Depression analysis, deep learning is applied in the form of obtaining pre-trained feature sets for each frame. From these features, dynamic pattern variations are recorded using the FDHH feature descriptor. This is followed by dimensionality reduction and then decision level fusion for prediction.

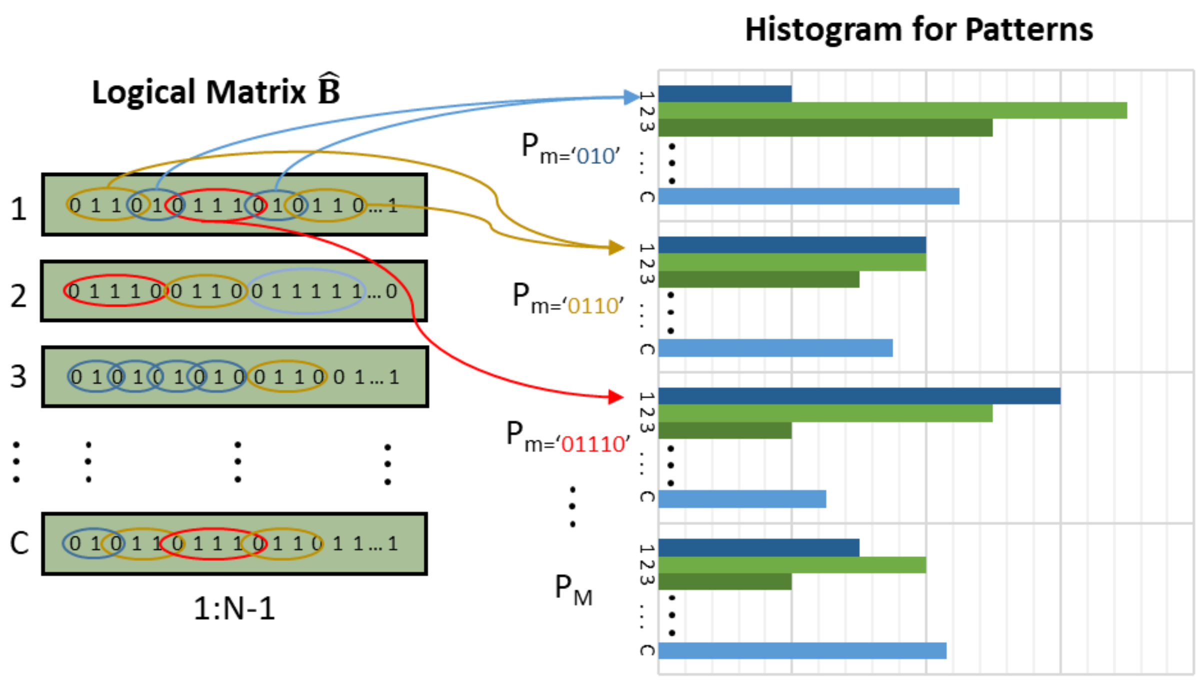


Figure 6.3 - The final stage of the FDHH algorithm, showing the patterns created from the output of the initial stage

The objective of determining induced emotions from movie clips has a different view of Depression analysis. Being that the data source is from movie clips already introduces a non-controlled environment factor compared to Depression analysis. This will consequently require a different approach to handle the objective. There are a lot of noisy samples in the form of inconsistent background and scenery. To tackle this form of data, a deep auto-encoder will be introduced to try and learn consistent parts across the movie clips. This can include the face and body parts throughout the various samples. The following procedure has a goal to reproduce the movie clips based on what the Deep Auto-Encoder has learnt, and then capture the feature variations in terms of patterns across the new representations using the FDHH algorithm. This will be followed by a deep convolutional neural network (DCNN) to get a deep understanding of what can be learnt from all the patterns.

6.4.1 Framework for Depression Analysis

The upcoming experiment on Depression analysis will be based on the AVEC2014 database. The emotions expressed by people suffering from MDD can vary from the healthy people, as the person is in a different mental state. This includes variation in the person's speech and facial expressions. This experiment is an attempt to map these expressions to the BDI-II scale, by combining dynamic descriptions within naturalistic facial and vocal expressions of the person. The AVEC2014 database contains dynamic visual and vocal data, both of which will be utilised.

Experiments on the AVEC2014 dataset will be extensive, covering a novel approach based on the FDHH feature, with the fusion of modalities and regression techniques. An alternative approach, similar to [82], will also be adopted to demonstrate the performance of MHH and the proposed extensions from

Chapter 5 on the AVEC2014 dataset. The alternative approach will also provide a means of comparison between the effectiveness of feature variation against the image motion. This can provide an indication whether the feature space is more valuable for detecting motion than the raw images.

```
Algorithm (FDHH)
Input: Normalised Feature Matrix $\hat{\mathbf{V}}(c,n)$, $c=1,\dots,C$, $n=1,\dots,N$
Initialisation: Patterns $m=1,\dots,M$,
                FDHH$(1{:}C,1{:}M) = 0$,
                $\hat{\mathbf{B}}(1{:}C,1{:}N\text{-}1) = 0$,
                $z(1{:}M)=0$,
                $r=0$
For $n$=1 to $N$-1    (For N1)
        Compute $\hat{\mathbf{B}}(1{:}C,n) = |\hat{\mathbf{V}}(c,n+1) - \hat{\mathbf{V}}(c,n)| \geq T$ (Binary Output)
End    (For N1)
For $c$=1 to $C$    (For C)
        For $n$=1 to $N$-1    (For N2)
                If ($\hat{\mathbf{B}}(c,n+1)$==1)    (If 1)
                        Update: $r=r+1$
                ElseIf ($r>0$ & $r\leq M$)
                        Update: $z(r)+1$
                        Update: $r=0$
                ElseIf ($r>M$)
                        Update: $z(M)+1$
                        Update: $R=0$
                End    (If 1)
        End    (For N2)
```

Figure 6.4 - Pseudocode for the FDHH algorithm, detailing all of the algorithm steps

The framework for the FDHH approach can be broken down into 4 main stages, with a separate branch for the visual and vocal data. Figure 6.5 & Figure 6.6 illustrate the entire system framework, from the raw data, till the predicted depression scales. The visual and vocal data are processed separately through different branches to investigate the use of multiple modalities. Combining these modalities will involve fusing the two branches at decision level, to give a joint prediction from the visual and vocal modalities.

6.4.1.1 Framework Details for Visual Data Branch

Starting with Figure 6.5, the visual data is addressed at the frame level in 4 stages. The first stage looks at each frame, using a selection of local feature descriptors to capture the characteristics of the subject's facial expression. The descriptors in the framework consist of LBP, EOH, LPQ that have been used in the previous chapter, along with deep features. The features are extracted from all the frames to make a feature matrix consisting of the (total frames $\times$ feature size).

The second stage applies the FDHH algorithm to each feature matrix. This is to try and capture the dynamic patterns of variations that occur within the sequence of the facial expression characteristics. The number of patterns M is defined by observing the variation depth that occurs through a random set of samples. If after M patters, the histogram indicates that the count is significantly less, or none, then the $M-1$ patterns are chosen for the framework. Once the FDHH features are produced for each local descriptor, they are concatenated together to produce a single feature vector per visual sequence. The third stage looks to reduce the dimensionality of the FDHH features, as they can be very large with the number of pattern histograms and local descriptors used. The PCA technique is applied to reduce the dimensionality down to 49 components, with the aim to keep at least up-to $\sim 94\%$ of the variance.

Finally, two regression techniques are adopted to predict the scale of depression, based on the FDHH feature. PLS regression and LR have been chosen again for their reliable performance in the Chapter 5 experiments. This will also allow direct comparison of the features performances from both experiments. This method reduces the predictors to a smaller set of uncorrelated components and performs least squares regression on these components, instead of on the original data. PLS regression is especially useful when the predictors are highly collinear, or when you have more predictors than observations and ordinary least-squares regression either produces coefficients with high standard errors or fails completely. Hence why PCA is applied to reduce the dimensionality below the sample size. Linear Regression is the other machine learning technique that is adopted for Depression analysis.

$$\hat{y}_i = \sum_{i=1}^{N} \sum_{j=1}^{R} w_j \cdot y_{(i,j)} \tag{6.6}$$

The decision level fusion technique is the same as that adopted for Depression analysis in Chapter 5, which is based on linear weighted fusion as shown in Equation 6.6. where R is the number of regression techniques used, N is the total number of samples, $y_{(i,j)}$ is the prediction value from the regression method j, of sample i, w_j is the weight assigned to regression method j. The objective is to fuse the confidence measure of each regression technique to get one set of predictions $\hat{y}$. The optimisation remains the same, with the efforts of both techniques linearly fused together using a weighted sum rule to aggregate their predictions.

6.4.1.2 Framework Details for Vocal Data Branch

Figure 6.6 shows the branch for the vocal data. It starts with taking short segments of the speech of 3 seconds at a time, with a 1-second shift between each segment. Each of these segments has a set descriptors extracted based on the openSMILE [182] toolbox. These extracted descriptors are shown in Table 6.1 and Table 6.2 [13], which consist of 2268 total baseline features that are provided by the host. They are composed of 32 energies and spectral related with 42 functionals; 6 voicing related LLD with 32 functionals; 32 delta coefficients of the energy/spectral LLD with 19 functionals; 6 delta coefficients of the voicing related LLD with 19 functionals; and 10 voiced/unvoiced durational features.

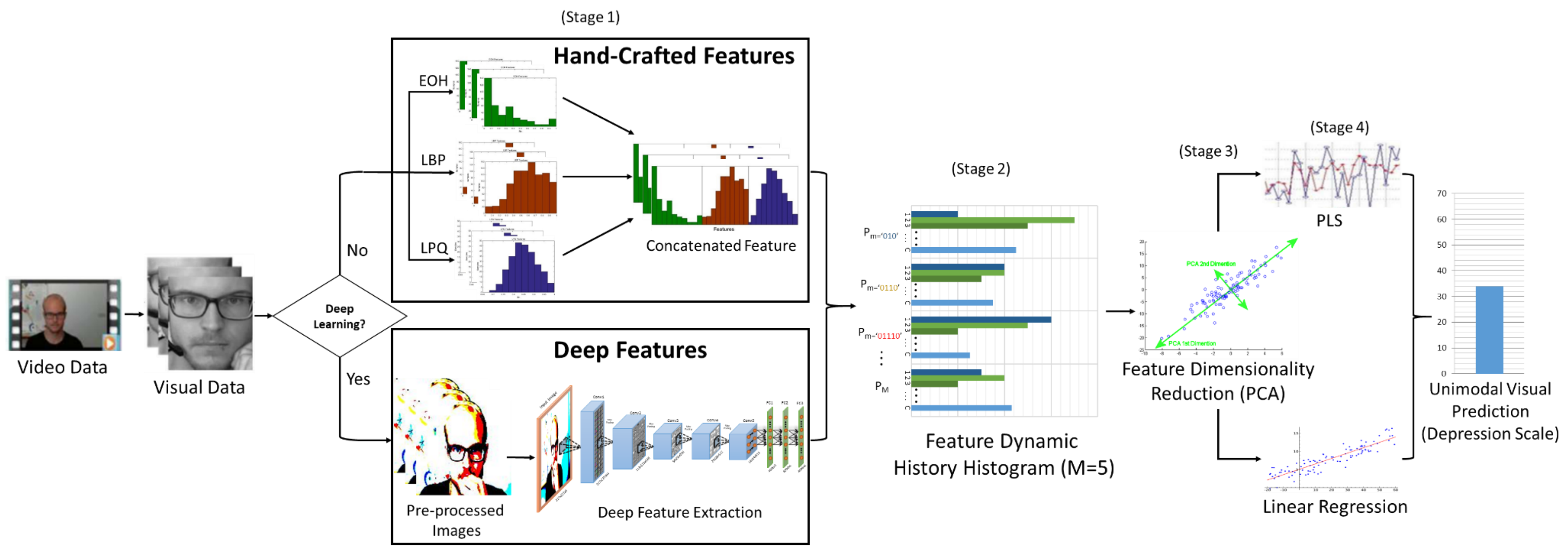


Figure 6.5 - Visual branch of the framework for depression scale prediction, using only visual data. 4 main stages are visualised from start to end. Stage 1 determines the feature extraction process, whether to use deep learning or hand-crafted techniques. Stage 2 applies FDHH on the feature vector sequence, Stage 3 reduces the feature dimensions using PCA and finally, Stage 4 applies decision level fusion using PLS and LR.

From this set of descriptors, some are selected to be used within the framework. The selection process involves first testing each individual feature vector with the development data set and then taking the top 8 performing descriptors. These are the following: Flatness, Energy in Bands 1k-4kHz (Band1000), Entropy, MFCC, Probability of Voicing (POV), PSY Sharpness, Shimmer and Zero Crossing Rate (ZCR).

Table 6.1 - Low-Level Descriptors from the openSMILE toolbox provided by the host [13]

Energy and spectral (32)
Loudness (auditory model based), zero crossing rate, energy bands from 250 – 650Hz, 1kHz – 4kHz, 25%, 50%, 75%, and 90% spectral roll-off points, spectral flux, entropy, variance, skewness, kurtosis, psychoacoustic sharpness, harmonicity, flatness, MFCC 1 – 16
Voicing related (6)
F_0 (sub-harmonic summation, followed by Viterbi smoothing), probability of voicing, jitter, shimmer (local), jitter (delta: “jitter of jitter”), logarithmic Harmonics-to-Noise Ratio (logHNR)

Next, all the features are tested in all combinations together, to determine the best set of descriptors that work well together. They are tested also tested on the development set, and from this, the best combination includes Flatness, Band1000, PSY Sharpness, POV, Shimmer and ZCR. These descriptors are concatenated to make the vocal feature vector for stage 2 of the vocal branch.

The third stage is similar to the visual branch, where the vocal feature vector is reduced in dimensionality down to 49 using PCA, with the aim to keep at least up-to $\sim$94% of the variance. The Final stage is also the same as the visual branch, where the PLS and LR techniques are fused using a weighted sum rule to predict the depression scales. To compare this framework fairly with the framework proposed in Chapter 5, majority of it will remain the same, with just the visual branch changing in how the data is processed. The only differences are in stage 1 and 2 for the visual branch, which is effectively reversed (hand-crafted features are extracted in stage 2), and the FDHH algorithm is replaced with MHH and its extensions (stage 1).

The third and fourth stage is the same, where the dimensionality is reduced using PCA to keep 49 components. Regression techniques PLS and LR use the reduced features to predict the depression scales, where the fusion is also based on the weighted sum rule. The predictions from the vocal branch are also fused with the visual branch to give a multi-modality prediction.

Table 6.2 - Set of all 42 functionals [13]. *The functionals below with* ^{1} *are not applied to delta coefficient contours.* ^{2} *are for delta coefficients the mean of only positive values is applied, otherwise the arithmetic mean is applied.* ^{3} *are not applied to voicing related LLD.*

Statistical functionals (23)
(positive^2) arithmetic mean, root quadratic mean, standard deviation, flatness, skewness, kurtosis, quartiles, inter-quartile ranges, 1%, 99% percentile, percentile range 1% - 99%, percentage of frames contour is above: minimum + 25%, 50%, and 90% of the range, percentage of frames contour is rising, maximum, mean, minimum segment length^{1,3}, standard deviation of segment length^{1,3}
Regression functionals^1 (4)
Linear regression slope, and corresponding approximation error (linear), Quadratic regression coefficient α, and approximation error (linear)
Local minima/maxima related functionals^1 (9)
mean and standard deviation of rising and falling slopes (minimum to maximum), mean and standard deviation of inter maxima distances, amplitude mean of maxima, amplitude range of minima, amplitude range of maxima
Other^{1,3} (6)
LP gain, LPC 1 – 5

6.4.2 Framework for Induced Emotion Detection

Induced emotion from content is a trickier task compared to observing content containing emotion. A specific emotion can be induced in a person from two video samples of completely different scenes and nature, whereas the content of 2 people demonstrating the same emotion contains similarities that can be observed. The task of the application is to predict the induced emotion via scales of Arousal and Valence.

The proposed framework will contain 3 stages designed to process, observe and learn the data. The amount of data is large and the content is from the wild (taken from movies). This entails random background scenes with many different people, containing objects, occlusions, different camera angles and poses etc. The first stage of the framework will be to try and understand and find similarities within the wide range of visual samples. This would be a form of pre-processing the data so that the later stages can work with the relevant information. This process will be achieved using a deep auto-encoder, to try and learn parameters that can be used to reproduce the contents of an image found frequently amongst the samples.

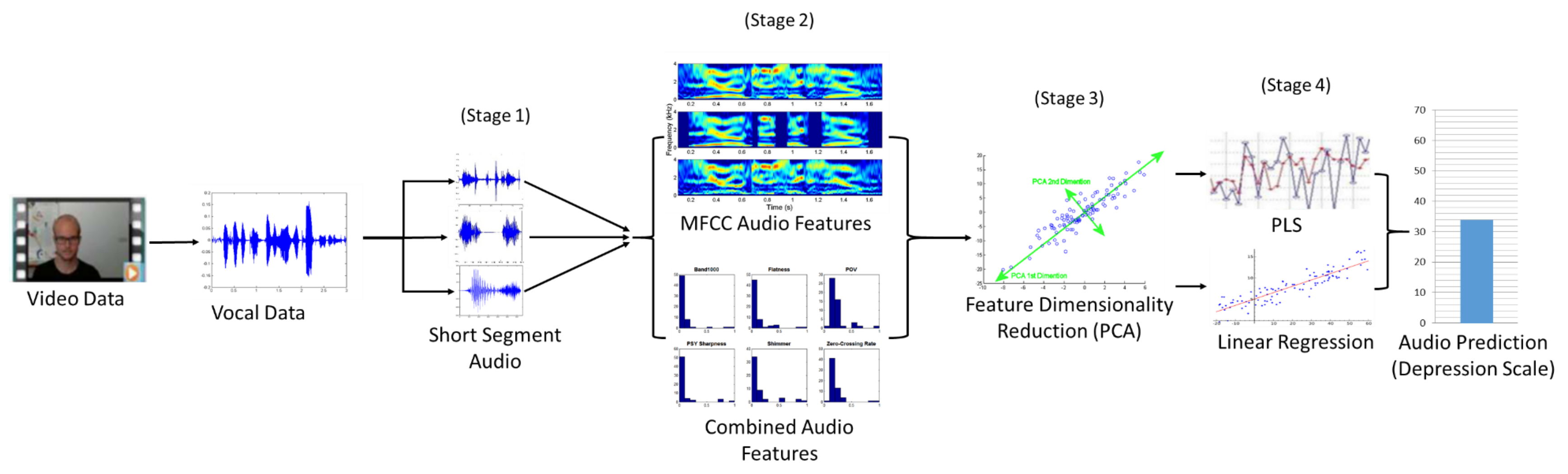


Figure 6.6 - Vocal branch of the framework for depression scale prediction, using only vocal data. 4 main stages are visualised from start to end. Starting with Stage 1, which extracts the short segments from the audio data. Stage 2 extracts audio features from the short segments, with the option of using only the MFCC feature, or a combination of 6 LLDs. Stage 3 reduces the feature dimensions using PCA and finally, Stage 4 applies decision level fusion using PLS and LR.

A deep auto-encoder is designed to process random image frames from a given video sample and reproduce them in a form that highlights main components within it. The architecture, that is inspired by the works in [183], contains 4 convolutional layers followed by 4 deconvolutional layers. Each convolutional layer shrinks the spatial dimensions, creating feature maps that represent various characteristics of the input. This will slowly transform the image into smaller encoded representations. The following 4 deconvolutional layers will use these representations to try and decode them and reproduce the original image. Each deconvolutional layer enlarges and combines the representations from the previous layer to slowly reconstruct the original image. Each of the convolutional and deconvolutional layers is followed by a Rectified Linear Unit activation layer.

Figure 6.7 visually demonstrates the network architecture of the Auto-encoder stage using a sample image from the LIRIS-ACCEDE database. The architecture reduces the spatial dimensions of an input image as it propagates through the network. What this essentially does is encode whatever it can learn from the initial input across a set of parameters with smaller spatial dimensions. However, these parameters do have a higher depth to store different variations/representations of the initial input. By reducing the spatial dimensions, this ensures that the network will train to encode the key details from the input and not just set all the parameter values to 1, to essentially just pass the whole image through.

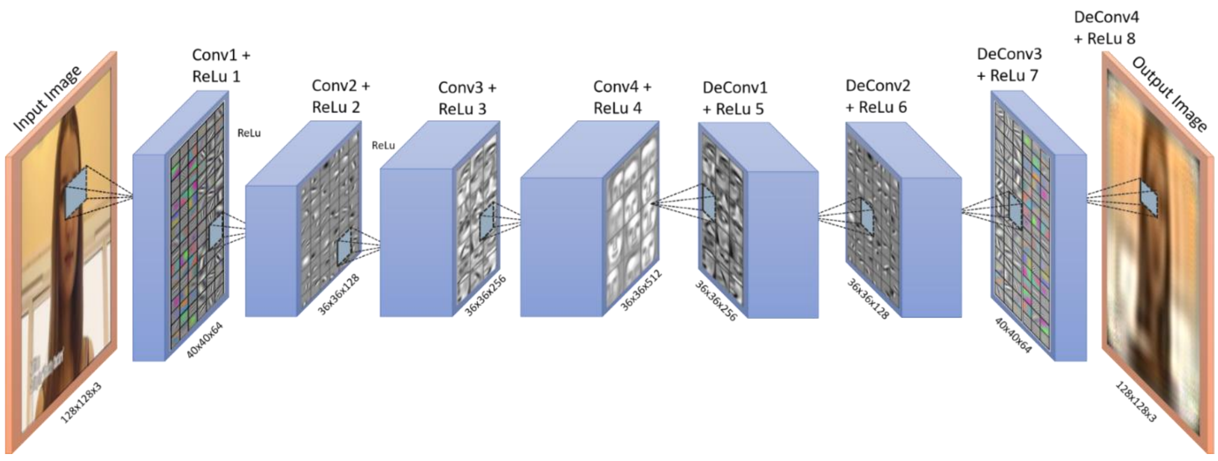


Figure 6.7 - First Stage of the framework, using a Deep Auto-Encoder architecture to learn common components that are found in the database

The deconvolutional layer (also known as Convolution Transpose) attempts to decode an input image using the learnt parameters, and the feature maps of the input image produced by the previous layer. A symmetrical design is used for the convolutional and deconvolutional layers. During the training process, the final output is compared to the original image using a loss function of either MSE or Euclidean loss. The MSE and Euclidean loss equations are calculated as follows in Equation 6.7 & Equation 6.8:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{6.7}$$

$$Euc = \sqrt{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{6.8}$$

Where Y is a vector of n ground-truth labels, $\hat{Y}$ is the vector of predictions produced by the network. The loss for the auto-encoder is based on the sum of every output neuron compared to each of its respective pixel from the original image. This is applied to both MSE and Euclidean loss functions.

The second stage occurs once the deep auto-encoder from the first stage is trained. Each sample video will then have all its image frames propagated through the deep auto-encoder to produce its refined version. Once the outputs are gathered, FDHH is applied on the sequence to capture the variations that occur throughout by searching for a set number of patterns. Figure 6.8 shows a sample of FDHH patterns, reshape as images, that are captured from a refined sequence produced from stage 1.

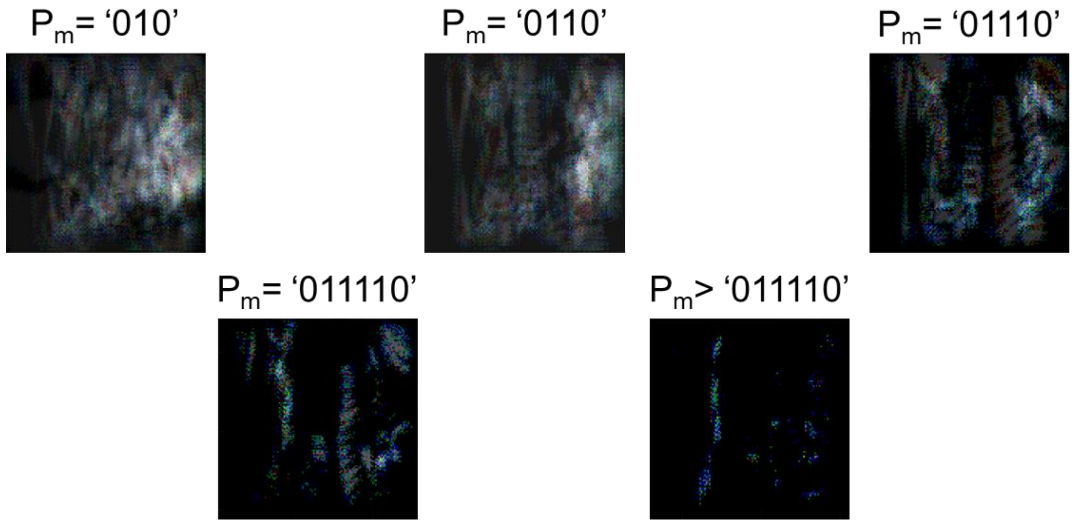


Figure 6.8 - Stage 2 of framework, capturing FDHH features of the output sequence from the Auto-Encoder

The final stage is to learn and map FDHH patterns to predict the Arousal and Valence scales used to describe the induced emotion. For this stage, a deep CNN is used to learn the FDHH features for each sample. The input feature itself is reshaped to have a 128×128 spatial resolution with a depth of 15 channels representing the patterns. The following layer is designed to reduce the spatial dimension a lot as there are a lot of parameters required to cope with the 15 channels of information. The structure goes as follows:

Conv1→ Pooling 1→ Conv2→ Pooling 2→ Conv3→ Conv4→ Conv5→ FC1→ FC2→ FC3→ Output

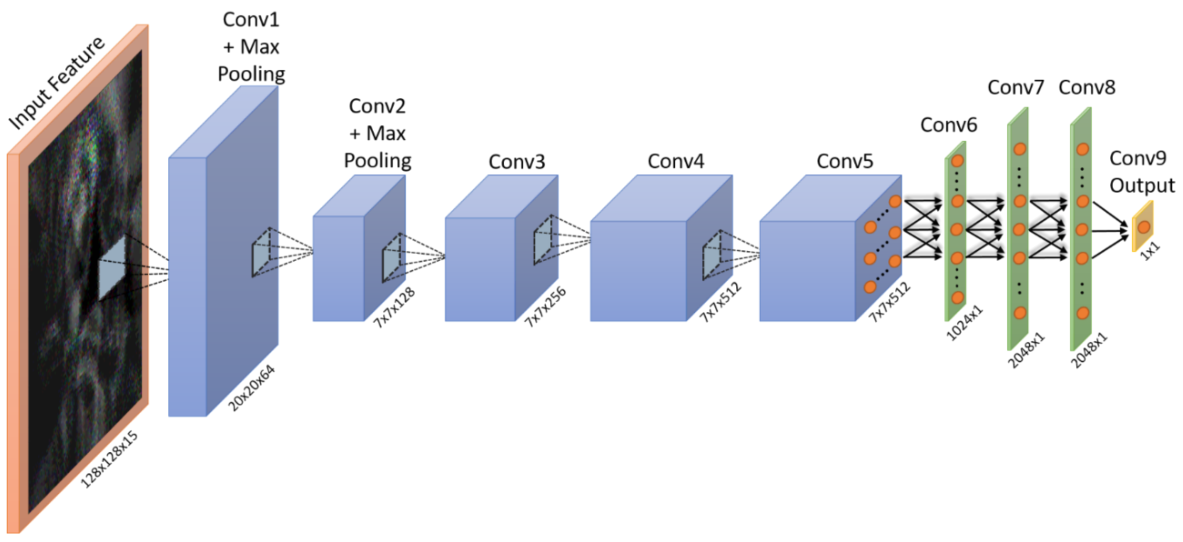


Figure 6.9 - Third Stage of the framework, using a deep network architecture with regression loss, to learn the feature variations captured by the FDHH algorithm

Each convolutional layer is followed by a ReLU layer for neuron activation. The prediction stage for this network is based on a regression model, with the loss function used for training as MSE and Euclidean. The loss functions for the third stage network will only deal with a single output neuron based on regression, unlike a whole image size as the first stage.

Two networks are trained in parallel, as the Arousal and Valence ground-truth labels are learnt separately. The audio information is also considered in the framework, for which there is a special case. The Audio features are produced using the openSMILE toolbox [182]. They are applied to the whole audio provided per sample. Using the toolbox, there are 16 low-level descriptors extracted which include the root mean square (RMS) of the frame energy; the zero-crossing rate (ZCR); the harmonics-to-noise ratio (HNR); pitch frequency (F0); Mel-frequency cepstral coefficients (MFCC) 1-12. Once these features are produced, further statistics are computed that include the 12 functionals mean, standard deviation, kurtosis, skewness, minimum and maximum value, relative position, and range as well as two linear regression coefficients with their mean square error (MSE).

When the audio descriptors are used in the framework, the deep network in stage 3 is no longer used as a predictor. Instead, it is used as a pre-trained network to extract the learnt features from the visual data. These features are then fused with the audio features and PLS regression is applied to predict the Arousal and Valence.

6.5 Related Datasets

For the upcoming experiments, the datasets chosen are related to capturing emotions using real values. They include the previously used AVEC2014 database for Depression analysis and a new database called LIRIS-ACCEDE that is based on predicting induced emotions.

6.5.1 AVEC2014

The AVEC dataset is visited again to compare a deep learning approach against the previously tested hand-crafted techniques from Chapter 5. The data usage remains the same, with the training, development and testing partitions playing the same role. Unlike the previous AVEC2014 experiment, the upcoming experiment will consider utilising the audio data within the framework. This will be in the form of the provided audio segments and feature sets for each.

6.5.2 LIRIS-ACCEDE

The LIRIS-ACCEDE Dataset [20] contains a large subset of videos, with the intention of inducing emotions on the viewer. The content is from several publicly available movies with the Creative Commons license. This dataset was selected as it shows a different view of how emotions are affected, by displaying what kind of content causes certain reactions to a person's mental state. There are two sets of data within the database, for two different tasks. The first is based on predicting continuous induced emotion across long videos. The other is predicting global induced emotion on small video clips of ~10 seconds. The induced emotion is based on the Arousal and Valence dimensions, that is 2 real values per frame for the continuous task, and 2 real values of each video clip for the global task. The upcoming experiment is based on the Emotional Impact of Movies Task that took place in the MediaEval16 workshop [184]. The global task was chosen to demonstrate the performance of the FDHH feature across whole sequences.

There are 9,800 excerpts that are extracted from 160 different movies for the global task, which are all under the Creative Commons licenses. The total time of all excerpts is 26 hours, 57 minutes and 8 seconds. The annotation of the database samples was produced workers using the crowdsourcing software CrowdFlower, and a quicksort algorithm to compare the annotations and rank the excerpts accordingly. The induced valence value ranging from +1 (Negative) to +5 (Positive), and the induced arousal value ranging from +1 (Passive) to +5 (Active).

6.6 Experimental Setup & Results

The following experiments will be designed to demonstrate the use of deep learning techniques along with the FDHH algorithm. The first experiment will be to revisit the Depression analysis task from Chapter 5, using a the proposed framework to handle it this time. A set of feature descriptors will be tested that include using deep features and hand-crafted features along with the FDHH algorithm. All of the results will be compared to the state-of-the-art result and others.

The second experiment will be based on a challenge that was partaken, to determine the emotional impact of movies. Specifically, the task of identifying induced emotions from a large-scale database of over 9,800 samples. Only 5 tests are run that is based on the challenge settings from the MediaEval16 workshop. These are compared to the other participants during that challenge.

6.6.1 Settings and Protocols

The AVEC2014 guidelines are followed from [13], based on the global challenge that had taken place in 2014. In this, the data had been split into 3 partitions, which are training, development and testing. Each contained 50 video samples from the “Northwind” and the corresponding “Freeform” tasks, totalling to 100 video clips per partition. The labels provided are based on 1 real value for each pair of tasks, with the idea that the depression scale should be the same for both, as each subject has conducted both tasks with the same mental state. The evaluation metric consists of the RMSE and MAE. At the time of the global challenge, the testing labels were not made available to the public, showing the initial experiments in [83] limited, as there were only 5 submissions allowed. Since the challenge has finished, the test labels have been made available to the public, in which further optimisation can be allowed beyond 5 attempts.

The LBP feature extraction for this experiment is based finding uniform patterns across 4x4 windows of a frame. Each pattern consists of 59 different combinations, which in total will become $4 \times 4 \times 59 = 944$ components per frame. The EOH and LPQ descriptors are captured across the whole frame. With EOH producing a 384-component vector, and LPQ producing a 256-component vector. When using the FDHH algorithm, the threshold is set to find 1% variation change, due to the very slow movement in facial expressions in the feature space. Based on this, patterns are found up to $M = 3$, which is selected by observing, from random samples, the depth of count in the resulting histogram using the 1% threshold.

When going for the deep learning approach, both AlexNet and VGGFace produce 4096-dimensional feature vectors at the FC1 and FC2 Layers. Before these are extracted, each frame has is gone through the required pre-processing steps before propagated through the network. These steps include the resizing of frames to $224 \times 224 \times 3$ for AlexNet and $227 \times 227 \times 3$ for VGGFace, and subtracting the mean image provided by the network, with AlexNet providing a pixel level average and VGGFace providing a colour channel level average.

The next stage was to propagate each pre-processed frame through the networks and obtain the features produced by the filter responses at the desired layers. Both features will be utilised separately to investigate which layer provides better features for Depression analysis. The Threshold for FDHH is also set to 1% for variation detection, with the patterns set to $M = 5$. These values have been chosen after observing a random sample with FDHH applied using a high pattern count of $M = 10$. These 10 patterns are then observed to determine at what point the pattern stops being detected, which in this case was at $P_m =' 01111110'$. Based on this, the pattern size was set to the last detected pattern which is $M = 5$.

For the Induced Emotion challenge using the MediaEval16 database, there are also 3 partitions of the data. The 9,800 provided samples are used for the training and validation of the system. They are split according to the details provided by the challenge hosts. The final test partition is based on 1,200 new samples that are released by the challenge host (without the ground-truth labels) approximately halfway through the challenge duration. The evaluation metrics are the MSE and Pearson's Correlation Coefficient (PCC). The MSE can be calculated as shown in Equation 6.7. PCC can be calculated by the following in Equation 6.9:

$$PCC = \frac{cov\left(Y - \hat{Y}\right)}{\sigma_Y \times \sigma_{\hat{Y}}} = \frac{\sum\left(\left(\hat{Y} - \mu_{\hat{Y}}\right)\left(Y - \mu_Y\right)\right)}{\sigma_{\hat{Y}} \times \sigma_Y} \tag{6.9}$$

Where Y is the set of ground-truth labels, $\hat{Y}$ is the set of predicted labels by the network, μ_Y and $\mu_{\hat{Y}}$ are the respected means for the labels, σ_Y and $\sigma_{\hat{Y}}$ are the respected standard deviations of the labels.

The initial process of the framework is to resize all the images the spatial dimensions of $128 \times 128 \times 3$ before the auto-encoder learns from the images. This is to have a consistent size across all samples which are not too big to run out of memory, or too small to lose important spatial information. During the auto-encoder training, the following hyperparameters are set for the network:

- 100 training epochs;
- Batch Size = 25;
- Learning rate = 0.000001;
- Momentum = 0.9;
- Weight Decay (regularisation) = 0.00001;
- Stochastic Gradient Descent is used as the learning optimiser;
- Network weights are randomly initialised using Xaviers improved method [89];
- Data is split 50% training 50% validation as directed by the challenge host;
- Each batch contains a single image from 25 visual sequences.

Once the auto-encoder is trained, every visual sample has all its frames propagated through the network to produce the new feature sequence ready for stage 2, with the dimensions remaining the same of $128 \times 128 \times 3$ per frame. Next, the FDHH algorithm is applied on all the new feature sequences, with the threshold set to 5% movement, that results in 5 patterns captured. This results in a new FDHH feature size on $128 \times 128 \times 3 \times 5 = 128 \times 128 \times 15$ per sample. Stage 3 uses the FDHH features for each sample to train two DCNNs, one to predict Arousal and the other to predict Valence. Both DCNNs are identical in parameter, size and hyperparameter settings. They consist of the same settings as those used for the auto-encoder, except for the following changes:

- Batch size is decreased to 10, due to the large FDHH image;

- Learning rate is set to 0.00001.

The two variants of the training processes are based on the change of loss function in stages 1 & 3. They are either set to MSE loss or Euclidean loss. The denotations are **DCNN_MSE** for the MSE loss and **DCNN_EUC** for Euclidean loss, with the audio features denoted as **Audio**.

6.6.1.1 Feature Sets for Depression Analysis

In the Depression Framework for the Visual Data branch, shown in Figure 6.5, the 3 local descriptors of EOH, LBP and LPQ are extracted to produce 3 separate sets of feature vectors from each frame of a sample video, as depicted in stage 1. For thorough testing, FDHH is initially applied on each local feature (EOH, LBP and LPQ) vector separately. This is to evaluate the performance of each individual descriptor with the FDHH feature extracted. These are denoted as **EOH_FDHH**, **LBP_FDHH** and **LPQ_FDHH** respectively. The final feature is the concatenation of the three local feature vectors, to produce a feature size of $384 + 944 + 256 = 1{,}584$ components. FDHH is applied on top of this feature vector to produce the feature denoted as **MIX_FDHH**. This feature is reshaped to contain $M \times 1{,}584 = 4{,}752$ components, which is for each Northwind and Freeform video. When producing the final feature for the next stage, the Northwind and Freeform feature vectors are concatenated to produce a $2 \times 4{,}752 = 9{,}510$ component vector.

For the deep learning route, pre-trained features using the AlexNet network are extracted at layers 16 & 18, which are FC1 & FC2. The same is done for VGGFace, extracting layers 32 & 34 which are the FC1 & FC2 layers. Each of these has FDHH applied to it, to produce vectors of $5 \times 4{,}096 = 20{,}480$ dimensions. With both Northwind and Freeform feature sets combined, this becomes a total of $40{,}960$ dimensions. Each of which is denoted as **A16_FDHH** & **A18_FDHH** for AlexNet, and **V32_FDHH** & **V34_FDHH** for VGGFace.

The feature dimensions for both approaches become large when combined. To help reduce the dimensions, PCA is applied producing a total of 49 components. This is the maximum principal components that can be used as the full MIX_FDHH matrix, as the training set is $50 \times 9{,}510$; which is 50 samples containing $9{,}510$ feature components. This produces a final feature of 50×49 components for the training set. The same process is applied to each of the Deep features, as they are individually tested. The development and testing sets are also reduced to 49 components, using the covariance matrix obtained from the training set.

Before the machine learning is applied, all of the feature sets are rank normalised between 0 and 1. This is to give boundaries to the feature values which helps the speed and performance of the regression method. This normalisation is achieved by looking at a component at a time across all samples from the training set. Equation 6.10 shows the Rank normalisation calculation, where X_i the i_{th} column of the feature matrix, α_{min} is the minimum value of the i_{th} column, α_{max} is the maximum of the i_{th} column, and $\hat{X}_i$ is the normalised column.

$$\hat{X}_i = \frac{X_i - \alpha_{min}}{\alpha_{max} - \alpha_{min}} \tag{6.10}$$

Once the features are normalised, PLS regression and Linear regression are applied separately to the feature sets to observe their individual performances. The fusion of the regression predictions is applied after, using the weighted sum rule. This technique is optimised by looking at how well the regression technique performs on the development partition.

The framework for the vocal branch, as shown in Figure 6.6, involves extracting the selected combination of audio features from the provided set of features. Initially, the short segment features are used, that have 2268 total features that comprise of those mentioned in Table 6.1 and Table 6.2. This feature is denoted as (Audio) and will also be tested as a whole feature. From these set of features, the combined audio features of Flatness, Band1000, POV, PSY Sharpness, Shimmer and ZCR are used, comprising of 285 from the 2268 features. This is denoted as (**Comb**). The final set of the audio features is the MFCC. This is used alone to see how it performs compared to the rest. This is denoted as (**MFCC**). Once the selected features are used, they are reduced in dimensionality using PCA. This has the same setting as the visual branch, producing 49 principal components that will be used as the reduced feature. And then the audio features are learnt through PLS and LR, from which decision level fusion occurs and the scales of depressions are predicted.

6.6.2 Experiment 6A – Deep Depression Analysis

The experiment will be split into 3 parts. The first part is to test the FDHH algorithm on hand-crafted and deep features individually using the development partition. The second part is to test the Audio feature sets on both the development and testing partitions. Then finally, the visual and audio feature sets are combined and tested on the test partition to produce the final evaluation for the features. Individual performances of the best visual features are also tested on the test partition to compare the best unimodal feature. In each of the tests, each individual feature is first tested using PLS regression, then followed by Linear regression, and then the fusion of both. The Audio fusion takes place after the individual feature tests. This is to allow a fair comparison with the framework shown in Figure 5.12.

The initial tests in Table 6.3 demonstrate the performance of the hand-crafted features and deep features combined with FDHH. One of the clear indications is that the deep features performed better than all the combinations of the hand-crafted features. LPQ_FDHH has performed the best compared to EOH and LBP, and the fusion of all three performing better when decision level fusion is applied. For the deep features, A16_FDHH and V32_FDHH have the lowest errors in terms of their pre-trained networks, indicating that the FC1 layer is once again better to extract than the FC2 Layer. The AlexNet network has performed better than the VGGFace network by 0.30 for RMSE and 0.26 for MAE. When compared to MIX_FDHH, AlexNet has 0.46 lower RMSE and 0.78 for MAE.

Table 6.3 - Performances of hand-crafted and Deep features taken from the visual sequence, with FDHH applied to capture feature variations. The systems are trained on the Training partition and tested on the Development partition. RMSE and MAE are used to evaluate the performance.

Visual Feature	PLS		LR		PLS + LR	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
EOH_FDHH	11.09	8.87	12.39	9.92	11.39	9.14
LBP_FDHH	11.16	9.34	12.68	9.86	11.15	9.18
LPQ_FDHH	10.88	8.79	11.49	9.73	10.63	8.70
MIX_FDHH	9.68	7.72	10.05	7.52	9.48	7.44
A16_FDHH	**9.02**	**6.66**	9.52	6.96	**9.05**	**6.68**
A18_FDHH	9.36	7.19	9.43	7.23	9.39	7.16
V32_FDHH	9.52	7.25	**9.32**	**6.90**	9.34	6.91
V34_FDHH	9.52	7.08	9.53	7.09	9.55	7.04

Table 6.4 - Performances of the Audio feature sets that are trained on the Training partition and tested on the Development partition, as well as the Testing partition.

Audio Feature + Partition	PLS		LR	
	RMSE	MAE	RMSE	MAE
Audio (Develop)	**10.08**	**8.25**	**10.21**	**8.39**
Comb (Develop)	11.52	9.31	11.64	9.42
MFCC (Develop)	10.70	8.86	10.92	8.86
Audio (Test)	10.46	8.42	10.73	8.45
Comb (Test)	10.49	8.52	10.33	12.99
MFCC (Test)	**10.42**	**8.04**	**10.28**	**8.07**

The results for the Audio features are presented in Table 6.4, demonstrating their performances on both development and test partitions. From the tests on the development partition, the Audio descriptor has performed the best, followed by the MFCC features. However, the test set puts MFCC as the best audio feature set, which is followed by Audio, then Comb. The difference between the audio and visual features is significant, with a difference of 1.06 for RMSE and 1.59 for MAE when comparing Audio and A18_FDHH on the development set.

Table 6.5 - Final test that evaluates the best set of combinations from the visual and audio sets of features. Each combined feature is tested on the

Feature	PLS		LR		PLS + LR	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
MIX_FDHH+MFCC	9.15	7.28	8.98	7.11	8.95	7.10
A16_FDHH+MFCC	8.19	6.58	8.45	6.87	8.27	6.60
A18_FDHH+MFCC	7.96	6.44	8.57	7.10	8.07	6.50
V32_FDHH+MFCC	**7.44**	**6.14**	**7.59**	**6.31**	**7.43**	**6.14**
V34_FDHH+MFCC	7.56	6.14	7.80	6.51	7.55	6.18
A18_FDHH	9.12	7.50	9.45	7.89	9.25	7.66
V32_FDHH	**8.01**	**6.68**	**8.23**	**6.90**	**8.04**	**6.68**

The proposed framework also included the fusion of the visual and audio modalities, to investigate is having sets of features extracted from various sources can improve performance. In Table 6.5, there is a clear indication that going for a bimodal approach does improve the system of a unimodal approach. In this test, the MIX_FDHH and all deep feature sets are fused with the MFCC audio descriptor, which is tested on the test partition for final evaluation. The MIX_FDHH feature did not perform as well as the deep feature combinations, with a decrease in its performance compared to the performance achieved in the development partition. The route of taking deep features rather than hand-crafted has proven to be a clear winner when using the FDHH algorithm. In terms of using FC1 or FC2 for the pre-trained networks, both performed very closely, with FC1 for VGGFace predicting the best.

The V32_FDHH feature has performed the best as an individual feature with an RMSE of 8.01 and MAE of 6.68, outperforming all the features in Table 5.21. When both visual (V32_FDHH) and audio (MFCC) features are fused, the error is dropped even lower to 7.43 RMSE and 6.14 MAE. Figure 6.10 shows how the output prediction looks like of the Test partition using the V32_FDHH + MFCC feature. It follows the general trend of the ground-truth labels, showing similar ups and downs.

6.6.2.1 *Experiment 6A Highlights*

This experiment has demonstrated a framework for Depression analysis based on temporal content. The framework captures features using advanced deep networks as feature extractors. These features are further processed using the developed and proposed FDHH algorithm, to observe patterns of variations that occur in the feature space. Multiple modalities are explored and fused together to produce a competing performance against the proposed approaches by others in Table 6.6. The table has two sets of results to demonstrate performances based on using single or multiple modalities, sorted in terms of best RMSE performance. The first part compares single modality techniques, including the approach proposed in Chapter 5 using the TSMHH descriptor.

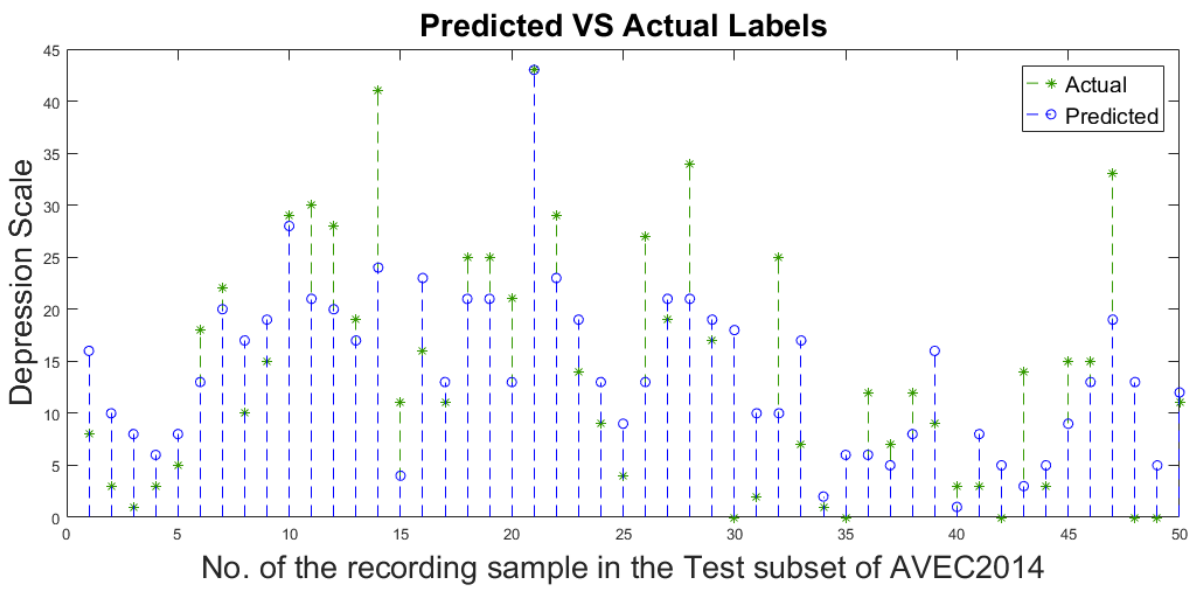


Figure 6.10 - Predictions of the Test partition produced by the V32_FDHH + MFCC feature

The comparison of single modality shows our proposed descriptor V32_FDHH produce the lowest RMSE and MAE of 8.01 and 6.68 respectively. The next best performance is produced by Dhall et al. [162], who achieved an RMSE of 8.91 and MAE of 7.08. V32_FDHH produces a significant improvement for single modality techniques of 0.90 for RMSE and 0.40 for MAE. The following best result is achieved by the TSMHH descriptor proposed in our earlier works.

For the multi-modality comparison, Williamson et al. [163] had previously held the top performance. This Chapter included the audio modality as a separate branch in the framework, which has been fused with the visual branch in the later stages. The MFCC feature on its own doesn't perform as well as some of the techniques from the visual modality. However, when it is combined with a visual technique, it provides a boost in performance. For the V32_FDHH feature, there was a boost of 0.58 for RMSE and 0.54 for MAE. This was enough of a boost to surpass the lowest error produced by Williamson et al. [163] quite significantly to become the new state-of-the-art in this area.

This experiment has also demonstrated how deep features can provide better feature descriptions than some of the popular hand-crafted techniques, for the application of Depression analysis. The Mix_FDHH + MFCC feature produced good results, better than most of the other approaches apart from Williamson et al. [163], but fell behind the both AlexNet and VGGFace pre-trained deep features.

The proposed framework also demonstrated a better use of deep learning techniques when compared to Chao et al. [174] and Zhu et al. [37] Chao et al. [174] adopted an idea that may work well for visually apparent tasks such as Action recognition and Facial expressions recognition. But for Depression analysis, their approach discounts many factors such as spatial resolution, that can help understand the slight changes in facial structure caused by the affected emotions in depressed subjects. Improvements can be made if they adopt a better deep network approach such as the VGGFace network, or others trained on faces.

Table 6.6 - Comparison of performances against other approaches applied on the Test partition, including the previous state-of-the-art results and performance achieved from Figure 5.12. Table is split into two halves, first showing performances using only a single modality, and the other showing performances using more than one modality.

Methods	**Modality**	**RMSE**	**MAE**
Ours (V32_FDHH)	**Visual**	**8.01**	**6.68**
Dhall et al. [162]	Visual	8.91	7.08
Ours (TSMHH)	**Visual**	**9.32**	**7.62**
Zhu et al. [37]	Visual	9.55	7.47
Kaya et al. [166]	Visual	9.61	7.69
Jain et al. [165]	Visual	10.24	8.39
MFCC (Test)	**Audio**	**10.28**	**8.07**
Baseline [13]	Visual	10.86	8.86
Mitra [185]	Audio	11.10	8.83
Perez et al. [164]	Visual	11.91	9.35
Ours (V32_FDHH + MFCC)	**Audio + Visual**	**7.43**	**6.14**
Williamson et al. [163]	Audio + Visual	8.12	6.31
Ours (MIX_FDHH)	**Audio + Visual**	**8.95**	**7.10**
Kaya et al. [166]	Audio + Visual	9.61	7.69
Kachele et al. [186]	Multimodal	9.70	7.28
Chao et al. [174]	Audio + Visual	9.98	7.91
Gupta et al.	Multimodal	10.33	-
Senoussaoui et al. [173]	Audio + Visual	10.43	8.33
Perez et al. [164]	Audio + Visual	10.82	8.99

Zhu et al. [37] achieve reliable performance with their system, but they decided to train a new network from scratch based on a face recognition database with ~495K images. They also trained a deep network to learn from optical flow images generated from the AVEC14 samples. They may have achieved better facial features by using existing networks that are trained on millions of faces, and that have been trained thoroughly for weeks. Another possible downfall could be using a deep network to learn the optical flow images from the AVEC14 samples. There is a limited amount of data for the deep network to train from and using such a large network architecture can result in severe overfitting.

6.6.3 Experiment 6B – Emotional Impact of Movies

This experiment is based on the challenge of the Emotional impact of movies that took place at the MediaEval16 workshop. Therefore, the rules and guidelines were followed based on the challenge. Only 5 submissions were allowed, which made it difficult to fully explore the framework proposed for the task, making this experiment short and concise.

There are 5 test runs submitted to the challenge, 2 of which are based on deep learning with FDHH, another 2 with deep learning using FDHH and audio, and 1 final submission based on just the audio.

Table 6.7 - Results of the Emotional Impact of Movies challenge, based on 5 submissions to predict the Arousal and Valence of induced emotion. The results are based on the Test data partition provided by the challenge host.

Submission	**Arousal**		**Valence**	
	MSE	**PCC**	**MSE**	**PCC**
DCNN_EUC	1.443	0.248	0.231	0.143
DCNN_MSE	**1.431**	0.263	**0.231**	**0.149**
Audio	1.525	0.143	0.236	0.125
DCNN_EUC + Audio	1.462	0.251	0.236	0.143
DCNN_MSE + Audio	1.441	**0.271**	0.237	0.144

Table 6.7 shows the results of each submission, based on their evaluation of MSE and PCC score. The goal is to have a low MSE and a high PCC. The best performance based on both Arousal and Valence was by the DCNN with MSE loss function during the stage 1 & 3 training process. The worse performance was using the Audio features alone, which didn't include any deep learning or use the FDHH algorithm. When both DCNN_MSE and audio features are combined, only the PCC for the Arousal is improved. This demonstrates that in this case, the visual features perform better than the audio and that the audio features only provide a little boost in PCC to the performance when combined with the visual features.

Table 6.8 - Comparison of Emotional Impact of Movies task with other participants in the challenge, performance is measured in MSE and PCC.

Submission	**Arousal**		**Valence**	
	MSE	**PCC**	**MSE**	**PCC**
Ours (DCNN_MSE)	**1.431**	**0.263**	**0.231**	**0.149**
Yang et al. [28]	**1.185**	0.174	0.378	-
Chen et al. [26]	1.479	**0.467**	0.218	**0.312**
Ma et al. [25]	1.531	0.266	**0.214**	0.295

Due to the large data size, it was difficult for many to participate in the challenge. Therefore, there are not many other techniques to compare the performance too. Table 6.8 shows the performance comparison with the remaining participants of the challenge. They show a similar performance, with Yang et al. [28] having the lowest MSE for Arousal. However, their performance for PCC is the worst, along with the MSE for the Valence label. Our performance for Arousal in terms of the MSE was the next best, followed by Chen et al. [26] and then Ma et al. [25]. For Valence, Ma et al. [25] produce the lowest MSE but Chen et al. had the highest PCC. Our MSE for Valence came in very close to both of theirs. From these results, it is difficult to get a clear image of what works best as all the evaluation metrics produced different responses with each method. Both Chen et al. and Ma et al. also incorporated deep learning techniques in their system, whilst Yang et al. projected high dimensional features into a low dimensional space, whilst keeping similar values from Arousal and Valence together and separating the dissimilar values far away from each other.

6.6.3.1 Experiment 6B Highlights

The experiment was brief yet demonstrated how deep learning and the FDHH algorithm can be utilised in a creative framework to tackle a large temporal database. The system was based on a 3-stage framework, with each stage playing a key role in the system. The first stage was a deep auto-encoder designed to process the raw data and remove any unwanted artefacts. The second stage applied the FDHH algorithm to the improved data from the first stage. The last stage was a deep network designed specifically to learn the FDHH feature histograms and map them separately to the Arousal and Valence ground-truth labels. Audio features were also utilised to see if there was any performance gain using the popular toolbox openSMILE. For 2 of the submissions, the audio features were fused with the Deep features and PLS regression was used for prediction.

The outcome from all 5 submissions indicated that using just the DCNNs provided the best performance for both Arousal and Valence predictions. Audio had performed the worst and the combined efforts of Audio and DCNN fell short behind the DCNN alone. Another finding from the experiment was that using the MSE loss for training the deep auto-encoder and deep convolutional network in stages 1 & 3 respectively, was better than using Euclidean loss. When our performance is compared to others, it is in a similar range, not being the best nor worst technique.

6.7 Evaluation and Discussion

In this Chapter, we investigated the use of deep learning techniques on dynamic emotion applications. The FDHH algorithm was also introduced to observe and captures variations that occur across the feature space along a sequence of feature vectors. Both ideas were combined and proposed in a set of frameworks to handle the tasks of Depression analysis and predicting Induced emotions.

For Depression analysis, state-of-the-art performances were achieved with the proposed framework. This demonstrates the capability of using the FDHH feature and deep features together. In Chapter 4, we investigated the use of pre-trained deep networks (namely VGGFace and AlexNet) for facial expression recognition. Here we observed some interesting results that indicated that a network trained on millions of objects performs better at FER than a network trained specifically on faces. The conclusion from this was that the VGGFace network struggled because it is trained to learn if faces are the same, which can result in producing similar features for a subject portraying an angry facial expression and the same subject portraying a surprise emotion. This is because the facial structure is off the same person, from which the network is trained to detect.

However, for Depression analysis, the VGGFace network has the best performance. This can be expected because the network is more suited to this task as we specifically aim to understand the face detected in each frame. In our framework, we do not use the network to specifically determine between different facial expressions, but to detect the facial parts and structure of the face across each frame. The FDHH algorithm has been designed and developed to detect the changes in the facial parts and structure that occur across the sequence of pre-trained features. The AlexNet pre-trained network has shown once again to be powerful for a job it is not trained for. However, it did not overcome the VGGFace network but gave an impressive performance that challenged many other approaches and techniques. This demonstrates that a well-trained deep network on a lot of images can capture useful information no matter the data type.

If Depression analysis is to become a service that can be used in real-time, then the speed of running the framework from start to end is a crucial factor for artificial intelligence systems. To get a better understanding of how long the framework takes to run, a table is compiled showing the time each stage of the system takes to execute. The time is based on a 6-second segment of a video sample for all tests to allow a fair comparison. The deep learning and hand-crafted approaches are observed, where the system has already been trained, and a new sample is being fed in for analysis.

The tests are run on the Windows 10 operating system using MATLAB 2017a. The PC has an i7-6700K processor @ 4.3GHz, with a Titan X (Pascal) GPU using the MatConvNet toolbox [142]. The tests are based on how the experiments were run, as a baseline performance. Further optimisations can take place with the code and tools to achieve faster speeds.

Table 6.9 shows the deep feature framework when using a GPU having a total time of less than the sample time (6s). This means that it could potentially process the information in real-time, which can be done by creating video segments. The hand-crafted features perform fast, except for LPQ. This feature also affects the Combined performance, but if the LPQ code is efficiently optimised further, it should reach under 6s. The pre-trained networks training time is unknown. However, as we did not retrain it, it does not need to be included in the timing. The regression and PCA stages don't apply the training process as this is only done once in a system. This is because in a real-time situation, the regression techniques will already be trained, and each new test sample will be processed individually.

For systems that require updating the trained information, a form of batch processing can be applied to take care of this process are certain intervals that do not impact process on the live stream of data.

Table 6.9 - Computation time compilation for running the proposed Depression analysis framework based on testing new sample of 6 seconds. Each main computational step is computed with the total time taken for it. The deep learning and hand-crafted approaches are demonstrated.

Video clip with 180 frames (6 second)	**Feature Extraction**	**FDHH**	**PCA**	**Regression PLS+LR**	**Total Time**
VGGFace with GPU	**3.265s**	**0.0356s**	**0.0027s**	**0.0034s**	**3.3067s**
VGGFace no GPU	34.07s	0.0356s	0.0027s	0.0034s	34.1117s
LBP	1.080s	0.0119s	0.0004s	0.0033s	1.0956s
EOH	1.146s	0.0104s	0.0006s	0.0041s	1.1611s
LPQ	6.830s	0.0084s	0.0005s	0.0053s	6.8442s
Combined (EOH+LBP+LPQ)	9.056s	0.0146s	0.0012s	0.0045s	9.0763s

When understanding the nature of inducing emotions, there are many ways possible. Media is a source capable of inducing all sorts of emotions from happy, joy, surprise to fear and anger. It is all linked to the content, personality and subjective assessments of the viewer. To try and capture this behaviour and model it is a very challenging task, as there is also a big dependency on the viewer. The emotional impact of movies task takes on the challenge to try and model this behaviour. They use a large set of short movie scenes and a crowdsourcing concept to get feedback from the public on the induced emotions they feel viewing them. We look to try and understand what patterns can be found from the data and the provided feedback that has been constructed in the form of Arousal and Valence.

The experiment on induced emotion demonstrated a framework designed to learn about the data itself before learning the main objective. The idea was to see how big data can be managed and understood to work more efficiently and be relatable to the task. The new representations were used with the FDHH algorithm to observe the occurrences of 5 variation patterns. This was to essentially summarise the movement of features across each sequence to a set of 5 histograms. These movements were learnt using a deep convolutional neural network in the final stage of the system. This network was trained from scratch to try and understand feature movement. The option for using a pre-trained deep network as a feature extractor, as done in Experiment 6A, was not investigated. This decision was made because the data is not in the form of images and has more than 3 channels. The final predictions were produced by two deep networks, one trained specifically for Arousal, and the other for Valence.

The results indicate that the proposed framework shows it is a capable idea that performs well, with room for improvement. Deep learning on the large quantity of visual data containing non-controlled scenes has proven to be better than using the audio features used in the experiment and even performs better when used alone. The FDHH feature demonstrated its usefulness in both experiments, justifying the nature of how it works as a high-quality descriptor.

The deep auto-encoder design was chosen for the task of understanding the content from the mixture of data. This was applied as a pre-processing tool to make the task easier for the rest of the system. The idea was to learn the common components found in the scenes for each sample, with the hopes that it will disregard the anomalies. When reconstructing an image sample, the deep auto-encoder, after being trained using MSE loss, had the effect of blurring out and even removing parts of the image that did not respond to the learnt parameters. This showed signs that the auto-encoder performed how it was designed too.

Experiment 6A on Depression analysis had some limitations that can influence the system if exploited. Firstly, the BDI-II self-report system may not be the best available measurement for depression, as it may not be able to understand the complete level of depression the user has. It is essentially limited by the questions asked, from which the responses may not accurately portray the true depressed feelings of the patient. There have been attempts by the same host to use other approaches in their later challenges (AVEC2016 [18]) where they adopt a PHQ-8 questionnaire (The PHQ-8 as a measure of current depression in the general population [51]). However, this dataset was not chosen as the host does not provide any raw visual content, which is what the proposed framework heavily relies on.

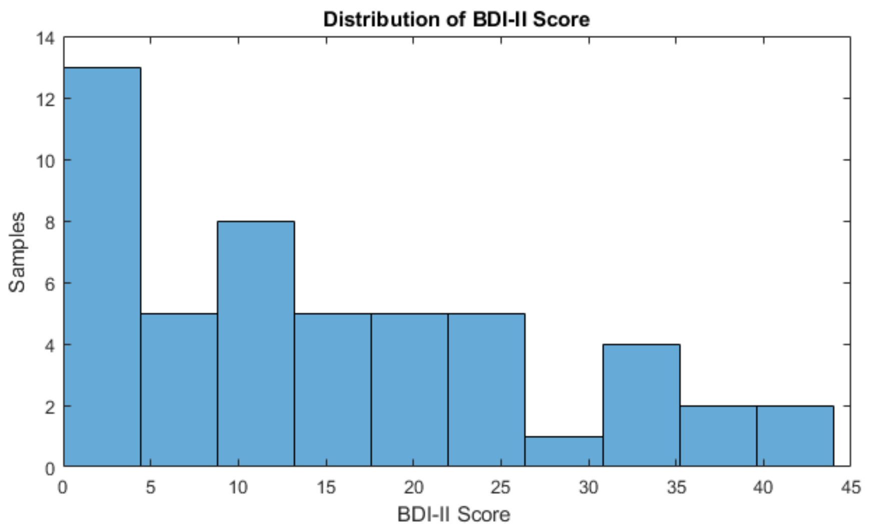


Figure 6.11 - Distribution of BDI-II score across the Training ground-truth labels, with a range of 0 to 44.

The BDI-II score is also limited in the samples provided, as it reaches a maximum of 44 in the training set, and 45 in the remaining development and testing. The distribution of scores for the training labels can be seen in Figure 6.11, and for all 3 partitions combined in Figure 6.12. These plots show a larger number of samples for the lower BDI-II scores. This could be an issue for some systems that get a new sample that has a BDI-II score higher than 45 and may not understand what it fully means.

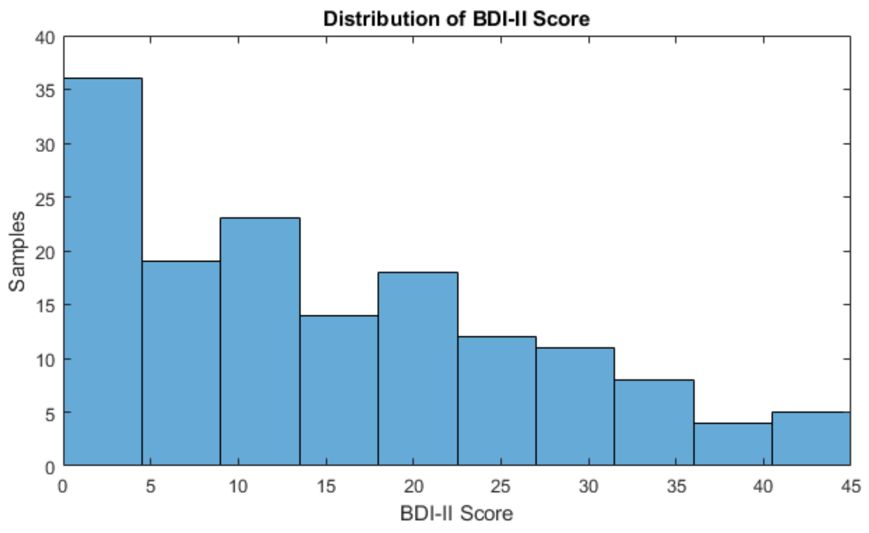


Figure 6.12 - Distribution of BDI-II score across all the ground-truth labels, with a range of 0 to 45.

Another limitation of the AVEC2014 dataset is that it is based on German patients that all seem to be of a Caucasian race. This could potentially affect the system performance when other ethnicities are provided as samples. However, the proposed features do mainly work on the grayscale image, and with the deep pre-trained networks, they have been trained on a variety of ethnicities beforehand. Also, the features are based on edges and patterns on the face, so it should not impact the performance that much.

For the Induced emotions task, there is a lot in that area that can be improved to make the task easier and produce systems that can learn the objective more. The labels for Arousal and Valence are based on a crowdsourcing concept, which is not always a reliable source compared to using professional psychologists. The judgement for Arousal and Valence will swing more accurately towards those that understand exactly what each means. If someone was to observe a sample that will induce a lot of emotion and rated it highly, to then receive a new sample that induced, even more, emotion than the scale may not extend far enough to cover this.

6.8 Summary

This chapter has presented novel techniques and frameworks to tackle emotion-based applications. Deep learning was explored in the form of a pre-trained feature extractor for Depression analysis, and as a deep auto-encoder combined with a trained classifier for determining induced emotions from movies. The experiments showed that the combination of FDHH and deep learning can challenge the best of frameworks out there. The following was achieved from this chapter:

- A novel algorithm called Feature Dynamic History Histogram was developed to observe the variations that occur in the temporal feature space. The technique looks for motion but in the higher quality mathematical representations of the image frames.

- The applications of Depression analysis and Induced emotions through movies addressed with novel frameworks that incorporate the FDHH algorithm along with deep learning techniques.
- The state-of-the-art performance was achieved for Depression analysis, using a pre-trained deep CNN as a feature extractor on each frame from a sequence. This is paired with the FDHH descriptor to capture the variations across the sequence of features, to produce a feature vector to predict the depression scales.
- Real-time performance can be achieved for Depression analysis
- An auto-encoder approach was used in the induced emotion task, to try and filter common properties across a large set of video sequences. FDHH was captured from the responses to produce feature vectors that are trained by another CNN to predict the induced emotions.
    - The results indicated a robust performance against others that participated in the same challenge.

Chapter 7. Conclusions and Future Works

The overall goal of this thesis was to understand and improve emotion-based studies using a host of technologies and frameworks. Deep learning was also introduced and applied to improve the area of analysing and recognising emotions. Several frameworks were proposed, along with a novel algorithm, and several extensions for an existing temporal motion descriptor. State-of-the-art result have been published for Depression analysis, along with other published work to support the contributions made.

7.1 Conclusion of Contributions

For facial expression recognition, there has been a lot of research done to attempt and improve the area. This includes:

- In Chapter 3, a thorough test of FER was implemented of various features captured from the 2D texture and 3D geometric information. From the experiments, the combination of both modalities through feature fusion provided a significant increase in performance.
- Chapter 4 observed and obtained key facial parts using a framework of various facial pre-processing techniques, along with deep learning techniques for learning.
- Chapter 5 extended towards temporal data and proposed extensions for the MHH descriptor. They have achieved an improvement over MHH and competing results against state-of-the-art techniques such as LBP-TOP.
- Chapter 6 attempted to capture movement within temporal data at the feature level, and a novel algorithm (FDHH) was developed for this. FDHH was integrated into a framework consisting of deep learning and produced state-of-the-art results.

7.2 Future Works

Amongst all the works produced, there are areas to progress and improve further. These can address problems found with the proposals for each task.

7.2.1 Extensions of Motion Descriptor

The proposed motion extensions produced some interesting results from the experiments in Chapter 5. There is a clear indication that the proposed extensions give an improvement over the original MHH descriptor. Some extensions benefit certain applications more than others, but generally, always better than the MHH descriptors. However, there are still some issues with the proposed extensions that can be worked on to improve the algorithms further. These are:

- It is hard to determine some of the hyper-parameters to select without using searching for the best using trial and error

- The descriptor can become very large, especially for large frames where there are areas that contain no motion
- The descriptor can vary significantly for applications that have sample sequences with different durations

Further improvements can be implemented to make a more efficient and all-around descriptor for a broader range of applications. The TSMHH descriptor can have spatial pooling applied, along with some statistical modelling to minimise the number of pattern histograms to capture. This can help optimise the descriptor as it can be difficult to know what to set the hyperparameter values for certain applications. An indicator that can determine which motion speeds are better to capture can improve the quality of the resulting motion histograms and make it easier to determine the right hyperparameter values to choose. Removing sparsity from the histograms can make descriptor a lot more efficient in memory and speed. In most cases, majority of the motion histograms contain empty bins that contribute towards a feature vector large.

A further processing stage can be developed to highlight the hotspots in which there is significant motion captured, to provide a better summarisation of the total motion. For our framework, we proposed to use local descriptors like LBP, LPQ and EOH. This may not be the best option as they are not designed specifically to work with motion information. Therefore, something more specific to motion can provide higher quality motion descriptors, potentially yielding better results. All of these improvements can be further work to this descriptor.

7.2.2 Improvement for Facial Expression Recognition

In this thesis, there were many works tested for facial expression recognition using texture and geometric data. Deep learning was integrated to replace the hand-crafted descriptors to investigate if there is an improvement. Based on the outcomes, the following can be considered to further improve the works.

- The deep learning techniques lack sufficient data to be the most effective it can. Therefore, it may be good to pre-train a deep CNN on many other databases prior and apply a fine-tuning process.
- Geometric information can be included in the training process that can consist of landmarks and depth maps of the face and facial parts.
- Future tests can include applying the framework, including facial pre-processing, and tougher facial images that are obtained in the wild. This can highlight any weakness to the framework in a harsh environment.

This works using the Joint Bayesian approach was a start in the area for FER and deep learning. Further works in this area can produce a more effective and robust system. These can include:

- Providing further integration to the CNN by incorporating a loss function that can contribute towards the training process of the CNN. This can produce a better feature at the fully connected layer that is tailored to the metric learning stage.
- Investigating a binary verification process for each expression, to determine how similar a new sample is to 6 separately trained dictionaries. A OneVSAll approach can be used for training CNNs separately.
- Investigate what layer of the network will produce the features to achieve the best Mahalanobis distance metric. Facial parts can also be trained individually and combined using Joint Bayesian, rather than parameter sharing.

7.2.3 Dynamic Framework with FDHH

Chapter 6 demonstrated a novel dynamic descriptor as FDHH, along with the innovative frameworks for applications based on emotions. For both tasks, improvements can be made that may help towards the performance of the task. With the framework for Depression analysis, a face detection procedure can be used to isolate the face across the frames. The generated features are likely to contain less noise and perform better. For lengthier episodes of samples and maybe for live streaming, the video can be equally segmented throughout the sequence, and a continuous reading between segments can be produced. Having short segments can be a way to deal with sudden reactions and emotional stimuli as it can be isolated in the segment, and therefore, not affect the whole sequence.

The FDHH feature can capture better information and consolidate the pattern information better. There are other areas of improvement that can make the descriptor more effective for all tasks it is used for. These can be applied to:

- Apply a pre-processing step to ensure that the sequence duration does not create biases towards the pattern histograms.
- Preservation of the original feature in some form may improve the descriptor quality, as currently only the changes are kept.
- Creating a manifold space that can mould the features and show how they change over time.
- Automatically determine the number of histograms required based analysing the patterns as they are detected.

All the mentioned improvements can be set as future work towards updating the algorithm and its goal. By making the process of the frameworks generic, they can be designed to work with applications that meet the data format. There definitely is room for improvement here, along with the following stages of the framework.

The scale for inducing emotions can have a different approach by having discrete emotions for this task. This can provide a common understanding that most of the public is familiar with, making their emotions more accurate for them to portray. The proposed framework can also be improved in some aspects of its goal. The auto-encoder idea can be improved by applying more research in this area. Some suggestions include:

- To pre-train the auto-encoder on specific entities so that the initial parameters already have an idea of what to learn.
- Use a more constructive loss function that can give feedback on how much data is retained within that image.
- Investigate different architectures and the depths of the convolutional filters to see the detail of information kept.
- Investigate if multiple stacked auto-encoders is better than a large single auto-encoder when dealing with a large database.

It may also be better to consider using the convolutional layer for the next stage rather than the output. The convolutional layer will contain feature maps of different responses rather than one feature map combining all responses like the output does. The FDHH algorithm can then be used on certain feature maps with the highest responses to capture variations that are more critical to the objective. All of these can be future works to improve individual parts of the system that can be used in many applications, and not just specific to the previously demonstrated applications.

Bibliography

[1] P. Ekman, "Universal Facial Expressions of Emotion," *Calif. Ment. Heal. Res. Dig.*, vol. 8, no. 4, pp. 151–158, 1970.

[2] A. Mehrabian, "Framework for a comprehensive description and measurement of emotional states.," *Genet. Soc. Gen. Psychol. Monogr.*, vol. 121, no. 1978, pp. 339–361, 1995.

[3] G. Valenza, A. Lanata, and E. P. Scilingo, "The role of nonlinear dynamics in affective valence and arousal recognition," *IEEE Trans. Affect. Comput.*, vol. 3, no. 2, pp. 237–249, 2012.

[4] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 1, pp. 39–58, 2009.

[5] L. F. Barrett, "Discrete Emotions or Dimensions? The Role of Valence Focus and Arousal Focus," *Cogn. Emot.*, vol. 12, no. 4, pp. 579–599, 1998.

[6] G. Sandbach, S. Zafeiriou, M. Pantic, and L. Yin, "Static and dynamic 3D facial expression recognition: A comprehensive survey," *Image Vis. Comput.*, vol. 30, no. 10, pp. 683–697, 2012.

[7] T. Wu, S. Fu, and G. Yang, "Survey of the facial expression recognition research," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7366 LNAI, pp. 392–402, 2012.

[8] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, 2001.

[9] S. Romdhani, J. Ho, T. Vetter, and D. J. Kriegman, "Face recognition using 3-D models: Pose and illumination," *Proc. IEEE*, vol. 94, no. 11, pp. 1977–1998, 2006.

[10] 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.

[11] S. Escalera *et al.*, "Chalearn looking at people challenge 2014: Dataset and results," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8925, pp. 459–473, 2015.

[12] M. Valstar *et al.*, "AVEC 2013: The Continuous Audio/Visual Emotion and Depression Recognition Challenge," *Proc. 3rd ACM Int. Work. Audio/Visual Emot. Chall.*, pp. 3–10, 2013.

[13] M. Valstar *et al.*, "AVEC 2014 – 3D Dimensional Affect and Depression Recognition Challenge," *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC '14*, pp. 3–10, 2014.

[14] L. Yin, X. Chen, Y. Sun, T. Worm, and M. Reale, "A high-resolution 3D dynamic facial expression database," *2008 8th IEEE Int. Conf. Autom. Face Gesture Recognit.*, no. 1, pp. 1–6, Sep. 2008.

[15] X. Zhang *et al.*, "BP4D-Spontaneous: a high-resolution spontaneous 3D dynamic facial expression database," *Image Vis. Comput.*, vol. 32, no. 10, pp. 692–706, Oct. 2014.

[16] M. Sjöberg *et al.*, "The MediaEval 2015 Affective Impact of Movies Task," *Proc. Mediaev. 2015 Work.*, pp. 14–16, 2015.

[17] E. Dellandréa, L. Chen, Y. Baveye, M. V. Sjöberg, C. Chamaret, and others, "The mediaeval 2016 emotional impact of movies task," *Mediaev. 2016 Multimed. Benchmark Work. Work. Notes Proc. Mediaev. 2016 Work.*, vol. 1739, 2016.

[18] M. Valstar *et al.*, "AVEC 2016: Depression, Mood, and Emotion Recognition Workshop and

Challenge," *Proc. 6th Int. Work. Audio/Visual Emot. Chall. - AVEC '16*, pp. 3–10, May 2016.

[19] Y. Baveye, J. N. Bettinelli, E. Dellandréa, L. Chen, and C. Chamaret, "A large video data base for computational models of induced emotion," *Proc. - 2013 Hum. Assoc. Conf. Affect. Comput. Intell. Interact. ACII 2013*, pp. 13–18, Sep. 2013.

[20] Y. Baveye, E. Dellandrea, C. Chamaret, and L. Chen, "Liris-accede: A video database for affective content analysis," *IEEE Trans. Affect. Comput.*, vol. 6, no. 1, pp. 43–55, 2015.

[21] O. Seddati, E. Kulah, G. Pironkov, S. Dupont, S. Mahmoudi, and T. Dutoit, "UMons at MediaEval 2015 affective impact of movies task including violent scenes detection," *CEUR Workshop Proc.*, vol. 1436, 2015.

[22] M. P. Vlastelica, S. Hayrapetyan, M. Tapaswi, and R. Stiefelhagen, "Kit at MediaEval 2015 - Evaluating visual cues for affective impact of movies task," *CEUR Workshop Proc.*, vol. 1436, 2015.

[23] Q. Dai *et al.*, "Fudan-Huawei at MediaEval 2015: Detecting violent scenes and affective impact in movies with deep learning," *CEUR Workshop Proc.*, vol. 1436, 2015.

[24] V. Lam, S. Phan, D. D. Le, S. Satoh, and D. A. Duong, "NII-UIT at mediaeval 2015 affective impact of movies task," *CEUR Workshop Proc.*, vol. 1436, 2015.

[25] Y. Ma, Z. Ye, and M. Xu, "THU-HCSI at MediaEval 2016: Emotional Impact of Movies Task.," *Mediaev. 2016 Multimed. Benchmark Work. Work. Notes Proc. Mediaev. 2016 Work.*, vol. 1739, 2016.

[26] S. Chen and Q. Jin, "RUC at MediaEval 2016 Emotional Impact of Movies Task: Fusion of Multimodal Features.," *Mediaev. 2016 Multimed. Benchmark Work. Work. Notes Proc. Mediaev. 2016 Work.*, vol. 1739, 2016.

[27] T. Anastasia and H. Leontios, "AUTH-SGP in MediaEval 2016 Emotional Impact of Movies Task," *Mediaev. 2016 Multimed. Benchmark Work. Work. Notes Proc. Mediaev. 2016 Work.*, vol. 1739, 2016.

[28] Y. Liu, Z. Gu, Y. Zhang, and Y. Liu, "Mining Emotional Features of Movies.," *Mediaev. 2016 Multimed. Benchmark Work. Work. Notes Proc. Mediaev. 2016 Work.*, vol. 1739, 2016.

[29] F. Baumann, J. Lao, A. Ehlers, and B. Rosenhahn, "Motion Binary Patterns for Action Recognition," *Proc. 3rd Int. Conf. Pattern Recognit. Appl. Methods*, pp. 385–392, 2014.

[30] H. Meng, N. Pears, M. Freeman, and C. Bailey, "Motion History Histograms for Human Action Recognition," *Embed. Comput. Vis.*, pp. 139–162, 2009.

[31] G. Zhao and M. Pietikäinen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 915–928, 2007.

[32] B. Jiang, M. F. Valstar, and M. Pantic, "Action unit detection using sparse appearance descriptors in space-time video volumes," *2011 IEEE Int. Conf. Autom. Face Gesture Recognit. Work. FG 2011*, pp. 314–321, 2011.

[33] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 248–255, Jun. 2009.

[34] A. Krizhevsky, I. Sutskever, and H. Geoffrey E., "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst. 25*, pp. 1–9, 2012.

[35] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *Handb. brain theory neural networks*, vol. 3361, pp. 255–258, 1995.

[36] X.-P. Huynh, T.-D. Tran, and Y.-G. Kim, "Convolutional Neural Network Models for Facial Expression Recognition Using BU-3DFE Database," *Inf. Sci. Appl. 2016*, pp. 441–450, 2016.

[37] Y. Zhu, Y. Shang, Z. Shao, and G. Guo, "Automated Depression Diagnosis based on Deep

Networks to Encode Facial Appearance and Dynamics," *IEEE Trans. Affect. Comput.*, vol. PP, no. 99, pp. 1–1, 2017.

[38] Y. Lv, Z. Feng, and C. Xu, "Facial expression recognition via deep learning," *2014 Int. Conf. Smart Comput.*, pp. 303–308, Nov. 2014.

[39] R. W. Picard, "Affective Computing," no. 321, pp. 1–16, 1995.

[40] E. S. Belfiore, "The Emotions of the Ancient Greeks: Studies in Aristotle and Classical Literature (review)," *Classical World*, vol. 101, no. 1. pp. 106–107, 2007.

[41] S. R. Leighton, "Aristotle and the Emotions," *Phronesis*, vol. 27, no. 2, pp. 144–174, 1982.

[42] V. Poli, "Evaluation of Biometrics," *Int. J. Comput. Sci. Netw. Secur.*, vol. 9, no. 9, pp. 261--269, 2009.

[43] T. Alashkar, B. Ben Amor, S. Berretti, and M. Daoudi, "Analyzing trajectories on Grassmann manifold for early emotion detection from depth videos," *2015 11th IEEE Int. Conf. Work. Autom. Face Gesture Recognit.*, pp. 1–6, May 2015.

[44] H. Meng and N. Bianchi-Berthouze, "Naturalistic Affective Expression Classification by a Multi-stage Approach Based on Hidden Markov Models," *Affect. Comput. Intell. Interact. Lect. Notes Comput. Sci.*, vol. 6975, 2011.

[45] J. Cohn, "Foundations of human-centered computing: Facial expression and emotion.," *Int. Jt. Conf. Artif. Intell.*, pp. 233–238, 2007.

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

[47] M. Marcus, M. T. Yasamy, M. van Ommeren, and D. Chisholm, "Depression, a global public health concern," *WHO Department of Mental Health and Substance Abuse*, 2012. [Online]. Available: http://www.who.int/mental_health/management/depression/who_paper_depression_wfmh_2012.pdf.

[48] World Health Organization, "Depression and other common mental disorders: global health estimates," 2017.

[49] World Health Organization, "The Global Burden of Disease: 2004 update," *World Heal. Organ. 2004 Updat.*, p. 146, 2008.

[50] R. A. Steer, R. Ball, W. F. Ranieri, and A. T. Beck, "Dimensions of the Beck Depression Inventory-II in clinically depressed outpatients," *J. Clin. Psychol.*, vol. 55, no. 1, pp. 117–128, 1999.

[51] K. Kroenke, T. W. Strine, R. L. Spitzer, J. B. W. Williams, J. T. Berry, and A. H. Mokdad, "The PHQ-8 as a measure of current depression in the general population," *J. Affect. Disord.*, vol. 114, no. 1–3, pp. 163–173, 2009.

[52] K. Kroenke, R. L. Spitzer, and J. B. W. Williams, "The PHQ-9," *J. Gen. Intern. Med.*, vol. 16, no. 9, pp. 606–613, 2001.

[53] P. Ekman and W. V Friesen, "Facial action coding system: A technique for the measurement of facial movement.," *CA Consult. Psychol. Press. Ellsworth, PC, Smith, CA (1988). From Apprais. to Emot. Differ. among unpleasant Feel. Motiv. Emot.*, vol. 12, pp. 271–302, 1978.

[54] P. Ekman, W. V. Friesen, and J. C. Hager, "Facial Action Coding System: The Manual," *Salt Lake City A Hum. Face*, pp. 1–197, 2002.

[55] M. F. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. Scherer, "The first facial expression recognition and analysis challenge," *Face Gesture 2011*, pp. 921–926, 2011.

[56] M. F. Valstar *et al.*, "FERA 2015 - Second Facial Expression Recognition and Analysis Challenge," 2015.

[57] A. Lonare and S. V Jain, "A Survey on Facial Expression Analysis for Emotion Recognition,"

Int. J. Adv. Res. Comput. Commun. Eng., vol. 2, no. 12, pp. 4647–4650, 2013.

[58] Y. L. Tian, T. Kanade, and J. F. Conn, "Recognizing action units for facial expression analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 97–115, 2001.

[59] D. Gabor, "Theory of communication. Part 1: The analysis of information," *Electrical Engineers-Part III: Radio and Communication Engineering, Journal of the Institution of*, vol. 93, no. 26. pp. 429–441, 1946.

[60] 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, pp. 211–216, 2006.

[61] C. Zhang and Z. Zhang, "A Survey of Recent Advances in Face Detection," *Microsoft Res.*, no. June, p. 17, 2010.

[62] G. Sandbach, S. Zafeiriou, M. Pantic, and L. Yin, "Static and dynamic 3D facial expression recognition: A comprehensive survey," *Image Vis. Comput.*, vol. 30, no. 10, pp. 683–697, 2012.

[63] B. Fasel and J. Luettin, "Automatic Facial Expression Analysis: A Survey," *Pattern Recognit.*, vol. 36, no. 1, pp. 259–275, 2002.

[64] A. K. Jain, N. K. Ratha, and S. Lakshmanan, "Object detection using Gabor filters," *Pattern Recognit.*, vol. 30, no. 2, pp. 295–309, 1997.

[65] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. - 2005 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, CVPR 2005*, vol. I, pp. 886–893, 2005.

[66] S. Arivazhagan, L. Ganesan, and S. P. Priyal, "Texture classification using Gabor wavelets based rotation invariant features," *Pattern Recognit. Lett.*, vol. 27, no. 16, pp. 1976–1982, 2006.

[67] V. Ojansivu and J. Heikkilä, "Blur insensitive texture classification using local phase quantization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5099 LNCS, pp. 236–243, 2008.

[68] H. Zhou, R. Wang, and C. Wang, "A novel extended local-binary-pattern operator for texture analysis," *Inf. Sci. (Ny).*, vol. 178, no. 22, pp. 4314–4325, 2008.

[69] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[70] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on Local Binary Patterns: A comprehensive study," *Image Vis. Comput.*, vol. 27, no. 6, pp. 803–816, May 2009.

[71] Y. Ma, "Number local binary pattern: An extended local binary pattern," *Wavelet Anal. Pattern Recognit.*, pp. 10–13, 2011.

[72] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002.

[73] D. G. Lowe, "Object recognition from local scale-invariant features," *Proc. Seventh IEEE Int. Conf. Comput. Vis.*, vol. 2, pp. 1150–1157, 1999.

[74] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–118, 2004.

[75] W. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," *Int. Work. Autom. Face Gesture Recognit.*, vol. 12, pp. 296–301, 1995.

[76] K. Levi and Y. Weiss, "Learning object detection from a small number of examples: the importance of good features," *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. 53–60, 2004.

[77] C. Yang, R. Duraiswami, and L. Davis, "Fast multiple object tracking via a hierarchical particle

filter," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. I, pp. 212–219, 2005.

[78] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkila, "Recognition of blurred faces using Local Phase Quantization," *2008 19th Int. Conf. Pattern Recognit.*, pp. 6–9, Dec. 2008.

[79] Z. Lei and S. Z. Li, "Fast multi-scale local phase quantization histogram for face recognition," *Pattern Recognit. Lett.*, vol. 33, no. 13, pp. 1761–1767, Oct. 2012.

[80] S. Dobrišek, "Face recognition in the wild with the Probabilistic Gabor-Fisher Classifier," *Int. Conf. Work. Autom. Face Gesture Recognit.*, vol. 2, pp. 1–6, 2015.

[81] C. H. Chan, M. A. Tahir, J. Kittler, and M. Pietikäinen, "Multiscale local phase quantization for robust component-based face recognition using kernel fusion of multiple descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1164–1177, 2013.

[82] H. Meng, D. Huang, H. Wang, H. Yang, M. AI-Shuraifi, and Y. Wang, "Depression recognition based on dynamic facial and vocal expression features using partial least square regression," *Proc. 3rd ACM Int. Work. Audio/visual Emot. Chall. - AVEC '13*, pp. 21–30, 2013.

[83] A. Jan, H. Meng, Y. F. A. Gaus, F. Zhang, and S. Turabzadeh, "Automatic Depression Scale Prediction using Facial Expression Dynamics and Regression," *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC '14*, pp. 73–80, 2014.

[84] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 3, pp. 257–267, 2001.

[85] R. Mattivi and L. Shao, "Human action recognition using LBP-TOP as sparse Spatio-temporal feature descriptor," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5702 LNCS, pp. 740–747, 2009.

[86] T. R. Almaev and M. F. Valstar, "Local gabor binary patterns from three orthogonal planes for automatic facial expression recognition," *Proc. - 2013 Hum. Assoc. Conf. Affect. Comput. Intell. Interact. ACII 2013*, pp. 356–361, Sep. 2013.

[87] C.-C. Chang and C.-J. Lin, "Libsvm," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 1–27, 2011.

[88] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1026–1034, 2016.

[90] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," *Procedings Br. Mach. Vis. Conf. 2015*, vol. 1, no. 3, pp. 6–17, 2015.

[91] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J. Physiol.*, vol. 195, no. 1, pp. 215–243, 1968.

[92] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[93] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," *ISCAS 2010 - 2010 IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.*, pp. 253–256, 2010.

[94] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]," *IEEE Comput. Intell. Mag.*, vol. 5, no. 4, pp. 13–18, 2010.

[95] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, 2015.

[96] C. Szegedy *et al.*, "Very Deep Convolutional Networks for Large-Scale Image Recoginition," *2015 IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 313, no. 5786, pp. 2018–2025, 2015.

[97] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016*

IEEE Conf. Comput. Vis. Pattern Recognit., vol. 7, no. 3, pp. 770–778, Jun. 2016.

[98] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[99] R. Gupta *et al.*, "Multimodal Prediction of Affective Dimensions and Depression in Human-Computer Interactions Categories and Subject Descriptors," *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC '14*, pp. 33–40, 2014.

[100] T. Wu, S. Fu, and G. Yang, "Survey of the facial expression recognition research," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7366 LNAI, pp. 392–402, 2012.

[101] C. A. Corneanu, M. O. Simón, J. F. Cohn, and S. E. Guerrero, "Survey on RGB, 3D, Thermal, and Multimodal Approaches for Facial Expression Recognition: History, Trends, and Affect-Related Applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1548–1568, 2016.

[102] A. Savran *et al.*, "Bosphorus database for 3D face analysis," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5372 LNCS, pp. 47–56, 2008.

[103] D. Cosker, E. Krumhuber, and A. Hilton, "A FACS valid 3D dynamic action unit database with applications to 3D dynamic morphable facial modeling," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2296–2303, 2011.

[104] T. C. Faltemier, K. W. Bowyer, and P. J. Flynn, "Using multi-instance enrollment to improve performance of 3D face recognition," *Comput. Vis. Image Underst.*, vol. 112, no. 2, pp. 114–125, 2008.

[105] H. Rabiu, M. I. Saripan, S. Mashohor, and M. H. Marhaban, "3D facial expression recognition using maximum relevance minimum redundancy geometrical features," *EURASIP J. Adv. Signal Process.*, vol. 2012, no. 1, p. 213, Dec. 2012.

[106] H. Soyel and H. Demirel, "3D facial expression recognition with geometrically localized facial features," *2008 23rd Int. Symp. Comput. Inf. Sci.*, pp. 1–4, Oct. 2008.

[107] X. Li, Q. Ruan, and Y. Ming, "3D Facial expression recognition based on basic geometric features," *IEEE 10th Int. Conf. SIGNAL Process. Proc.*, pp. 1366–1369, Oct. 2010.

[108] U. Tekguc, H. Soyel, and H. Demirel, "Feature selection for person-independent 3D facial expression recognition using NSGA-II," *2009 24th Int. Symp. Comput. Inf. Sci.*, pp. 35–38, Sep. 2009.

[109] K. Yurtkan and H. Demirel, "Feature selection for improved 3D facial expression recognition," *Pattern Recognit. Lett.*, vol. 38, no. 1, pp. 26–33, 2014.

[110] T. Yun and L. Guan, "Human emotional state recognition using real 3D visual features from Gabor library," *Pattern Recognit.*, vol. 46, no. 2, pp. 529–538, 2013.

[111] P. Lemaire, M. Ardabilian, L. Chen, and M. Daoudi, "Fully automatic 3D facial expression recognition using differential mean curvature maps and histograms of oriented gradients," *Autom. Face Gesture Recognit. (FG), 2013 10th IEEE Int. Conf. Work.*, pp. 1–7, Apr. 2013.

[112] Hao Tang and T. S. Huang, "3D facial expression recognition based on automatically selected features," *2008 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 1–8, Jun. 2008.

[113] Q. Zhen, D. Huang, Y. Wang, and L. Chen, "Muscular Movement Model Based Automatic 3D Facial Expression Recognition," *Multimed. Model. 21st Int. Conf. MMM 2015, Sydney,NSW, Aust. January 5-7, 2015, Proceedings, Part I*, pp. 522–533, 2015.

[114] X. Yang, D. Huang, Y. Wang, and L. Chen, "Automatic 3D facial expression recognition using geometric scattering representation," *2015 11th IEEE Int. Conf. Work. Autom. Face Gesture Recognit.*, pp. 1–6, May 2015.

[115] 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, Nov. 2015.

[116] Jun Li, Shasha Li, Jiani Hu, and Weihong Deng, “Adaptive LPQ: An efficient descriptor for blurred face recognition,” *2015 11th IEEE Int. Conf. Work. Autom. Face Gesture Recognit.*, no. 1, pp. 1–6, May 2015.

[117] A. Jan and H. Meng, “Automatic 3D Facial Expression Recognition using Geometric and Textured Feature Fusion,” in *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2015, pp. 15–20.

[118] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[119] P. Liu, S. Han, Z. Meng, and Y. Tong, “Facial Expression Recognition via a Boosted Deep Belief Network,” *2014 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1805–1812, 2014.

[120] L. Chao, J. Tao, M. Yang, Y. Li, and Z. Wen, “Multi-scale Temporal Modeling for Dimensional Emotion Recognition in Video,” *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC ’14*, pp. 11–18, 2014.

[121] L. Zhong, Q. Liu, P. Yang, J. Huang, and D. N. Metaxas, “Learning Multiscale Active Facial Patches for Expression Analysis,” *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1499–1510, 2015.

[122] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, “Incremental face alignment in the wild,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1859–1866, 2014.

[123] S. Ren, X. Cao, Y. Wei, and J. Sun, “Face Alignment at 3000 FPS via Regressing Local Binary Features,” *Cvpr*, 2014.

[124] X. Yu, J. Huang, S. Zhang, and D. N. Metaxas, “Face landmark fitting via optimized part mixtures and cascaded deformable model,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2212–2226, 2016.

[125] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[126] P. Nair and A. Cavallaro, “3-D Face Detection, Landmark Localization, and Registration Using a Point Distribution Model,” *Multimedia, IEEE Trans.*, vol. 11, no. 4, pp. 611–623, 2009.

[127] X. Zhu and D. Ramanan, “Face detection, pose estimation, and landmark localization in the wild,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2879–2886, 2012.

[128] X. Xiong and F. De La Torre, “Supervised descent method and its applications to face alignment,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 532–539, 2013.

[129] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic, “Robust discriminative response map fitting with constrained local models,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3444–3451, 2013.

[130] S. Zhang, C. Zhang, and Q. Yang, “Data preparation for data mining,” *Appl. Artif. Intell.*, vol. 17, p. 2003, 2010.

[131] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 512–519, 2014.

[132] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” *Int. Conf. Learn. Represent.*, pp. 1–15, 2014.

[133] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep Face Recognition,” *Procedings Br. Mach. Vis. Conf. 2015*, no. Section 3, p. 41.1-41.12, 2015.

[134] Y. Sun, D. Liang, X. Wang, and X. Tang, “DeepID3: Face Recognition with Very Deep Neural Networks,” *arXiv Prepr. arXiv1502.00873*, pp. 2–6, Feb. 2015.

[135] S. Zheng *et al.*, “Conditional Random Fields as Recurrent Neural Networks,” *2015 IEEE Int. Conf. Comput. Vis.*, pp. 1529–1537, Dec. 2015.

[136] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 3431–3440, 2015.

[137] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the Devil in the Details: Delving Deep into Convolutional Nets,” *Proc. Br. Mach. Vis. Conf.*, pp. 1–11, 2014.

[138] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *CoRR*, pp. 1–14, Sep. 2014.

[139] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” *Univ. Massachusetts Amherst Tech. Rep.*, vol. 1, pp. 07–49, 2007.

[140] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, “Bayesian face revisited: A joint formulation,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7574 LNCS, no. PART 3, pp. 566–579, 2012.

[141] B. Moghaddam, T. Jebara, and A. Pentland, “Bayesian modeling of facial similarity,” *Adv. Neural Inf. Process. Syst.*, pp. 910–916, 1999.

[142] A. Vedaldi and K. Lenc, “MatConvNet - Convolutional Neural Networks for MATLAB,” *Arxiv*, pp. 1–15, 2014.

[143] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, “On the importance of initialization and momentum in deep learning,” *Int. Conf. Mach. Learn.*, vol. 28, pp. 1139–1147, 2013.

[144] C. Schüldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local SVM approach,” *Proc. - Int. Conf. Pattern Recognit.*, vol. 3, pp. 32–36, 2004.

[145] S. Oh *et al.*, “AVSS 2011 demo session: A large-scale benchmark dataset for event recognition in surveillance video,” *2011 8th IEEE Int. Conf. Adv. Video Signal Based Surveillance, AVSS 2011*, pp. 527–528, 2011.

[146] S. S. Beauchemin and J. L. Barron, “The computation of optical flow,” *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, 1995.

[147] I. Laptev and T. Lindeberg, “Local descriptors for spatio-temporal recognition,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3667 LNCS, pp. 91–103, 2006.

[148] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black, “Optical Flow with Semantic Segmentation and Localized Layers,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 3889–3898, 2016.

[149] Deqing Sun, E. B. Sudderth, and M. J. Black, “Layered segmentation and optical flow estimation over time,” *2012 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1768–1775, 2012.

[150] R. Niese, A. Al-Hamadi, A. Farag, H. Neumann, and B. Michaelis, “Facial expression recognition based on geometric and optical flow features in colour image sequences,” *IET Comput. Vis.*, vol. 6, no. 2, p. 79, 2012.

[151] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, “A database and evaluation methodology for optical flow,” *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.

[152] J. W. Davis and A. F. Bobick, “The representation and recognition of human movement using temporal templates,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 23, no. 402, pp. 928–934, 1997.

[153] D. Weinland, R. Ronfard, and E. Boyer, “Free viewpoint action recognition using motion history volumes,” *Comput. Vis. Image Underst.*, vol. 104, no. 2–3 SPEC. ISS., pp. 249–257, 2006.

[154] T. Senechal, V. Rapp, H. Salam, R. Seguier, K. Bailly, and L. Prevost, “Facial action recognition

combining heterogeneous features via multikernel learning," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 4, pp. 993–1005, 2012.

[155] A. Klaeser, M. Marszalek, and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," *Procedings Br. Mach. Vis. Conf. 2008,* p. 99.1-99.10, 2008.

[156] M. Jung, R. Poppe, M. Poel, and D. K. J. Heylen, "Touching the Void -- Introducing CoST," *Proc. 16th Int. Conf. Multimodal Interact. - ICMI '14,* pp. 120–127, 2014.

[157] A. Flagg and K. MacLean, "Affective touch gesture recognition for a furry zoomorphic machine," *Proc. 7th Int. Conf. Tangible, Embed. Embodied Interact. - TEI '13,* no. August, p. 25, 2013.

[158] S. Yohanan and K. E. MacLean, "The Role of Affective Touch in Human-Robot Interaction: Human Intent and Expectations in Touching the Haptic Creature," *Int. J. Soc. Robot.*, vol. 4, no. 2, pp. 163–180, 2012.

[159] M. M. Jung, X. L. Cang, M. Poel, and K. E. MacLean, "Touch Challenge '15," *Proc. 2015 ACM Int. Conf. Multimodal Interact. - ICMI '15,* pp. 387–390, 2015.

[160] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR,* 2008.

[161] Y. F. A. Gaus *et al.*, "Social Touch Gesture Recognition using Random Forest and Boosting on Distinct Feature Sets," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction - ICMI '15,* 2015, pp. 399–406.

[162] A. Dhall and R. Goecke, "A temporally piece-wise fisher vector approach for depression analysis," *2015 Int. Conf. Affect. Comput. Intell. Interact. ACII 2015,* pp. 255–259, 2015.

[163] J. R. Williamson, T. F. Quatieri, B. S. Helfer, G. Ciccarelli, and D. D. Mehta, "Vocal biomarkers of depression based on motor incoordination and timing," *Proc. 4th ACM Int. Work. Audio/Visual Emot. Chall. (AVEC '14),* pp. 41–47, 2014.

[164] H. Pérez Espinosa, H. J. Escalante, L. Villaseñor-Pineda, M. Montes-y-Gómez, D. Pinto-Avedaño, and V. Reyez-Meza, "Fusing Affective Dimensions and Audio-Visual Features from Segmented Video for Depression Recognition," *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC '14,* pp. 49–55, 2014.

[165] V. Jain, J. L. Crowley, A. K. Dey, and A. Lux, "Depression Estimation Using Audiovisual Features and Fisher Vector Encoding," *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC '14,* no. 3, pp. 87–91, 2014.

[166] H. Kaya, F. Çilli, and A. A. Salah, "Ensemble CCA for Continuous Emotion Prediction," *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC '14,* pp. 19–26, 2014.

[167] A. Jan, Y. F. A. B. A. Gaus, F. Zhang, H. Meng, and F. Zhang, "BUL in MediaEval 2016 Emotional Impact of Movies task," *Mediaev. 2016 Multimed. Benchmark Work. Work. Notes Proc. Mediaev. 2016 Work.*, vol. 1739, 2016.

[168] A. Jan, H. Meng, Y. F. A. Gaus, and F. Zhang, "Artificial Intelligent System for Automatic Depression Level Analysis through Visual and Vocal Expressions," *IEEE Trans. Cogn. Dev. Syst.,* vol. PP, no. 99, pp. 1–13, 2017.

[169] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 35, no. 1, pp. 221–231, 2013.

[170] M. Liu, S. Li, S. Shan, R. Wang, and X. Chen, "Deeply learning deformable facial action parts model for dynamic expression analysis," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics),* vol. 9006, pp. 143–157, 2015.

[171] J. F. Cohn *et al.*, "Detecting depression from facial actions and vocal prosody," *Proc. - 2009 3rd Int. Conf. Affect. Comput. Intell. Interact. Work. ACII 2009,* 2009.

[172] G. McIntyre, R. Gocke, M. Hyett, M. Green, and M. Breakspear, "An approach for automatically measuring facial activity in depressed subjects," *2009 3rd Int. Conf. Affect. Comput. Intell.*

Interact. Work., pp. 1–8, Sep. 2009.

[173] M. Senoussaoui, M. Sarria-Paja, J. F. Santos, and T. H. Falk, “Model Fusion for Multimodal Depression Classification and Level Detection,” *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC ’14*, pp. 57–63, 2014.

[174] L. Chao, J. Tao, M. Yang, and Y. Li, “Multi task sequence learning for depression scale prediction from video,” *2015 Int. Conf. Affect. Comput. Intell. Interact. ACII 2015*, pp. 526–531, 2015.

[175] M. Nasir, A. Jati, P. G. Shivakumar, S. Nallan Chakravarthula, and P. Georgiou, “Multimodal and Multiresolution Depression Detection from Speech and Facial Landmark Features,” *Proc. 6th Int. Work. Audio/Visual Emot. Chall. - AVEC ’16*, pp. 43–50, 2016.

[176] X. Ma, H. Yang, Q. Chen, D. Huang, and Y. Wang, “DepAudioNet,” *Proc. 6th Int. Work. Audio/Visual Emot. Chall. - AVEC ’16*, pp. 35–42, 2016.

[177] R. Weber, V. Barrielle, C. Soladié, and R. Séguier, “High-Level Geometry-based Features of Video Modality for Emotion Prediction,” *Proc. 6th Int. Work. Audio/Visual Emot. Chall. - AVEC ’16*, pp. 51–58, 2016.

[178] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, “YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software,” *Proc. 11th Int. Soc. Music Inf. Retr. Conf. (ISMIR 2010)*, no. Ismir, pp. 441–446, 2010.

[179] I. J. Goodfellow *et al.*, “Challenges in representation learning: A report on three machine learning contests,” *Neural Networks*, vol. 64, pp. 59–63, 2015.

[180] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1–9, 2015.

[181] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning Face Representation from Scratch,” *arXiv Prepr. arXiv1411.7923*, Nov. 2014.

[182] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile,” *Proc. Int. Conf. Multimed. - MM ’10*, p. 1459, 2010.

[183] Y. Zhou, D. Arpit, I. Nwogu, and V. Govindaraju, “Is Joint Training Better for Deep Auto-Encoders?,” *ArXiv e-prints*, May 2014.

[184] M. Larson, M. Soleymani, G. Gravier, B. Ionescu, and G. J. F. Jones, “The Benchmarking Initiative for Multimedia Evaluation: MediaEval 2016,” *IEEE Multimed.*, vol. 24, no. 1, pp. 93–96, Jan. 2017.

[185] V. Mitra *et al.*, “The SRI AVEC-2014 Evaluation System,” *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC ’14*, pp. 93–101, 2014.

[186] M. Kächele, M. Schels, and F. Schwenker, “Inferring Depression and Affect from Application Dependent Meta Knowledge,” *Proc. 4th Int. Work. Audio/Visual Emot. Chall. - AVEC ’14*, pp. 41–48, 2014.