Fast head profile estimation using curvature, derivatives and deep learning methods

Gordon Alexander Dickers, MSc (Dist), PGCE(FE), BSc (hons)

Supervised by: Professor John Rees, Dr Kemi Ademoye and Dr Tim Bashford

This research was undertaken under the auspices of
the University of Wales Trinity Saint David

Submitted in partial fulfilment for the award of the degree of Doctor of Philosophy

University of Wales

2021

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed (candidate)

Date: 21st August 2021

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where correction services have been used the extent and nature of the correction is clearly marked in a footnote(s). Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed (candidate)

Date: 21st August 2021

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for deposit in the University's digital repository**.**

Signed (candidate)

Date: 21st August 2021

Abstract

Fast estimation of head profile and posture has applications across many disciplines, for example, it can be used in sleep apnoea screening and orthodontic examination or could support a suitable physiotherapy regime. Consequently, this thesis focuses on the investigation of methods to estimate head profile and posture efficiently and accurately, and results in the development and evaluation of datasets, features and deep learning models that can achieve this. Accordingly, this thesis initially investigated properties of contour curves that could act as effective features to train machine learning models. Features based on curvature and the first and second Gaussian derivatives were evaluated. These outperformed established features used in the literature to train a long short-term memory recurrent neural network and produced a significant speedup in execution time where pre-filtering of a sampled dataset was required. Following on from this, a new dataset of head profile contours was generated and annotated with anthropometric cranio-facial landmarks, and a novel method of automatically improving the accuracy of the landmark positions was developed using ideas based on the curvature of a plane curve. The features identified here were extracted from the new head profile contour dataset and used to train long short-term recurrent neural networks.

The best network, using Gaussian derivatives features achieved an accuracy of 91% and macro F1 score of 91%, an improvement of 51% and 71% respectively when compared with the un-processed contour feature. When using Gaussian derivative features, the network was able to regress landmarks accurately with mean absolute errors ranging from 0 to 5.3 pixels and standard deviations ranging from 0 to 6.9, respectively. End-to-end machine learning approaches, where a deep neural network learns the best features to use from the raw input data, were also investigated. Such an approach, using a one-dimensional temporal convolutional network was able to match previous classifiers in terms of accuracy and macro F1 score, and showed comparable regression abilities. However, this was at the expense of increased training times and increased inference times. This network was an order of magnitude slower when classifying and regressing contours.

Acknowledgments

I would like to thank my supervisors, Professor John Rees, Dr. Kemi Ademoye and Dr. Tim Bashford for their continued support, guidance, and advice throughout the period of my PhD research. Also, I'd like to thank Professor Ian Wells, Dr Glen Jenkins and Dr John Beaton who encouraged, supervised, and advised me in the initial stages of this work.

I am grateful to the University of Wales, Trinity St. David for providing the opportunity to engage in this research and I thank my colleagues in the School of Applied Computing for their advice and encouragement during this time.

To Ceri and Elen, I say thank you for your tolerance and unconditional support.

Thank you.

Publications

The following paper (see Appendix A) has been accepted for publication as a direct result of the research conducted in this thesis:

G. Dickers, J. Rees, T. Bashford, O. Ademoye, "Effectiveness of Curvature and Signal Derivatives in Fast Curve Segmentation", *Conference Proceedings UKSim2021, AIMS2020/21, International Journal of Simulation Systems, Science & Technology,* vol. *22, 202*.
doi: 10.5013/IJSSST.a.22.01.12

Table of Contents

Table of Illustrations

List of Tables

List of Abbreviations

1DTCNN	One-dimensional temporal convolutional neural network
2D	Two dimensional
3D	Three dimensional
2.5D	Two dimensional with depth information
AAM	Active appearance models
AFLW	Annotated Facial Landmarks in-the-Wild
AFW	Annotated Faces in-the-Wild
ANSUR 2012	Anthropometric Survey of Army Personnel of 2012
ASM	Active Shape Models
BPTT	Back propagation through time
CA	Classical manual anthropometric assessments
CAESAR	Civilian American And European Surface Anthropometry Resource
CGI	Computer generated imagery
CLM	Constrained local models
CNN	Convolutional neural network
CVA	Cranio-vertebral angle
DFT	Discrete Fourier transform
DNN	Deeply layered neural networks
DoG	Derivative of the Gaussian
DPM	Deformable parts models
DTW	Dynamic time warping
ECG	Electrocardiogram
exc	Exocanthion
FHP	Forward head posture
FN	False negatives
FFNN	Feed forward neural network
FP	Contains false positives
FSST	Fourier Synchrosqueezed Transform
FWHM	Full width at half maximum
gn	Gnathion
GPU	Graphics programming unit

Abbreviations	**(continued)**
HOG	Histogram of Gradients
LFPW	Labelled Face Parts in the wild
LFW	Labelled Faces in the wild
li	Labiale inferius
LoG	Laplacian of the Gaussian
LSTM	Long short-term memory
ls	Labiale superius
ME	The mean error
MAE	Mean absolute error
ML	Machine learning
MSE	Mean squared error
n/a	Not defined
prn	Pronasale
RGB	Reg, Green, Blue image channels
RGB-D	Reg, Green, Blue and Depth image channels
RNN	Recurrent neural network
RoI	Regions of interest
se	Sellion
sl	Sublabiale
sn	Subnasale
sto	Stomion
TN	True negatives
ToF	Time of Flight
TP	True positives

1 Introduction

Accurately locating and identifying the features of an individual's face is an intuitive ability in humans requiring little thought or effort; so too is classifying gender, age, emotion, pose and other more abstract attributes. It is not, however, a trivial task. These human abilities are the product of millions of years of evolution and, while current machine vision state of the art tools and hardware can match human performance in a narrow range of predefined image recognition activities, in general they cannot compare.

Given the ease with which a human can detect and recognize objects, it is not surprising that the computer vision community turned to the biological sciences for inspiration in developing artificial vision systems capable of competing with or replicating human vision capability. Early computer vision research was guided by existing investigations into the structure of animal neurological systems. This research was informed by the discovery of regions of the brain that appeared to specialize in identifying low level visual primitives (Hubel and Wiesel, 1959). Hierarchies of further structures had also been located in the primate brain (Kruger *et al.*, 2013) that process these visual primitives in order to identify more complex objects and this layered approach to feature detection and segmentation was then co-opted into computer vision research (Fukushima, 1980). More recently, the design of deeply layered convolutional neural networks (CNNs) used in deep learning is also underpinned by this insight (LeCun *et al.*, 1998; Zhang, Zhao and LeCun, 2015).

Whether the activity of locating and identifying features is done by a machine or an animal, the early processing of the image to identify suitable primitives underpins any vision system. Once these features are available, they can then be used by other algorithms to solve problems in classification or regression.

Traditional image processing and computer vision methods attempted to solve the feature recognition problem using the ingenuity and skill of the researcher to create custom made algorithms with some notable early successes in detecting low level features such as corners, edges and contours, and in segmentation and region/object labelling (Abdou and Pratt, 1979; Otsu, 1979; Canny, 1986). Latterly, these have been combined with

optimization methods to identify higher level features, solve alignment problems, model form and so on (Hough, 1962; Lowe, 2004; Bay, Tuytelaars and Gool, 2006; Rosten and Drummond, 2006). The more recent successes of deep learning approaches have surpassed this earlier work in some areas (Krizhevsky, Sutskever and Hinton, 2012), but the effort expended in for example, face detection and recognition is immense, relying on the availability of vast image datasets scraped from the internet and then, in many instances, laboriously hand labelled. Compiling and labelling huge datasets of images is not a trivial task, and the computing time required to train a face detection system using such datasets is also significant, but often worthwhile.

With these thoughts in mind, the focus of this work involves using computer vision and machine learning (ML) ideas together with the mathematical concepts of curvature and curve derivatives to identify salient craniofacial landmarks of profile head images in order to accurately measure head posture at speeds fast enough for use in real-time systems. An important requirement of this process is the development of a suitably labelled head profile dataset that can be used to train suitable classifiers and regressors.

1.1 Why Measure Head Posture?

For this study we refer to head posture in the context of physiological measurement of the head using anthropometric methods. In particular, anthropometric landmarks are identified that could help accurately define the posture, as are specific regions of the face, such as the philtrum, the region between the top of the upper lip (the labiale superius) up to the bottom of the nose (the subnasale). These landmarks and regions that are identifiable from a profile image, together with other landmarks such as the tragus of the ear, can quantify position, that is, define the head's posture. This can be extended to measure forward head posture (FHP), should additional body landmarks be available. For example, landmarks on the upper shoulders and neck.

There are also clear advantages to accurate, real-time, automatic head posture measurement across a range of disciplines. For example, in a clinical setting, a system capable of fast acquisition of facial landmarks can speed up the process of posture measurement and provide near instantaneous feedback to the clinician and patient alike. Moreover, such a tool can make use of established 2D and 3D photogrammetric

technologies which claim to afford highly repeatable and precise measurements on a par with direct anthropometry (Aldridge *et al.*, 2005; Ozsoy *et al.*, 2009; Dindaroğlu *et al.*, 2015).

It has been found that qualitative assessment of head posture by observation can suffer from poor intra-observer and inter-observer reliability and validity (Silva, Punt and Johnson, 2010). Here, the application of a real-time head posture measurement system based on 2D and 3D photogrammetry concepts and emerging deep learning methods could improve the validity and accuracy of qualitative posture assessments.

A real-time posture measurement system could also facilitate posture modification and control in a range of settings. FHP in adults is correlated with neck pain and a limited cervical range of motion (Fawzy Mahmoud *et al.*, 2019) and this has become more common as mobile devices and desktop computers have risen in popularity. Here, immediate feedback from a real-time system could support a suitable physiotherapy regime and in general, real-time posture measurement has potential in all aspects of head and craniofacial anthropometry where accurate location of facial landmarks is crucial.

Additionally, localized craniofacial landmarks may be used directly within a medical setting, prior to calculating posture, for example in sleep apnoea screening (Deberry-Borowiecki, Kukwa and Blanks, 1988; Lam *et al.*, 2005; Eastwood *et al.*, 2020) or in orthodontic examination, harmony assessment and treatment planning (Lippold *et al.*, 2014).

Realtime measurement of head and shoulder posture may also prove useful outside a clinical setting, for example, to support voice control in singing, aspects of dance posture and so on.

The success of the approach relies not only on the development of fast and accurate algorithms that can solve the landmarking problem in this context, but also upon camera technology capable of capturing real-time image data that contains enough useful information and is low in cost.

1.2 Cameras: 2D or not 2D, that is the question

Before a face profile can be landmarked it must first be found within an image. There are several existing computer vision approaches that could be used to detect faces from a 2D image. Popular methods are based upon the Viola-Jones algorithm (Viola and Jones, 2001), histogram of gradients (HOG) (Dalal and Triggs, 2005), active appearance models (AAM) and more recently deeply layered neural networks (DNN), and in particular deeply layered convolutional neural networks, for example RetinaFace (Deng *et al.*, 2019), though most of these methods perform best with frontal or near frontal face poses. Once detected, landmarking prior to pose estimation can begin. Again, recent work on landmarking and shape estimation has focused on AAMs and DNNs, with DNNs currently achieving best performance across several competitions and challenges (Russakovsky *et al.*, 2015; Wang *et al.*, 2015; Deng *et al.*, 2019).

What has this got to do with camera choices? It is apparent that the majority of effort recently has been applied to 2D image land-marking and whilst the results are impressive there is significant processing involved in identifying landmarks. So, could depth information improve the efficiency of landmarking? Depth information can be binarized and used to infer contours of the profile face in controlled environments such as those discussed earlier in this introduction. This information can be used to generate a 2D contour from which curvature can be calculated and, following on from this, salient landmarks identified. Key areas of this thesis will focus upon the use of profile contours extracted in this way and use features related to curvature to identify head landmarks and face regions that in turn can be used to describe head posture.

A 3D camera provides RGB values and an additional depth image, often such an image is referred to as an RGB-D or RGBD image. Several technologies exist that achieve this in different ways. Light sectioning or sheet of light triangulation methods emit a horizontal stripe of light and the reflected light from the object is converted by triangulation into depth information (Minolta, 2001). Stereophotogrammetry where multiple cameras, carefully aligned with matched optical parameters, are available but here the correspondence problem needs to be solved, that is, common points on each image need to be aligned and this takes valuable processing time. They also have the disadvantage of being relatively expensive and less compact than emerging alternatives such as structured light and Time of flight (ToF) cameras.

Structured light (active stereo) cameras illuminate the scene with a specific infrared light pattern and an image of the scene is then captured. Depth is calculated by processing this image. A second RGB camera also captures colour information. An alternative is ToF depth cameras. Here a light pulse is sent out and reflects off objects in the scene. These reflections are received at the camera's infra-red image depth sensor and the depth is inferred from the time it takes for the reflected ray to make the round trip. A second RGB camera is also used here to capture colour information, though the pixel densities of the cameras are typically different.

Recent advances in camera technologies have resulted in widely available, compact and low cost devices, such as Intel's RealSense Depth Camera (Intel, 2018) or, more recently, the Microsoft Azure Kinect that allow high definition colour images with depth information to be captured in real time at up to 30 frames per second (Microsoft, 2019). Additionally, laser and infrared light used for depth measurement is largely unaffected by local changes in the lighting of the scene which results in a more robust depth capturing process. Depth information captured by these devices are generally independent of the background scene and so segmentation of the regions of interest is not a problem. This is a significant advantage when fast processing of the scene information is paramount. ToF cameras are also beginning to be integrated into mobile devices which suggests that this technology will become significantly cheaper and widespread.

1.3 Anthropometry

Accurate and reliable identification of anthropometric landmarks is the foundation of anthropometry. Manual anthropometric measurement made by expert technicians has traditionally been regarded as the gold standard. However, digital anthropometry that uses photogrammetric methods based on 2D camera images and, more recently on 2.5/3D cameras and scanners, has become popular with many studies in the literature comparing the efficacy of both manual and digital approaches and, whilst each has distinct advantages and disadvantages (see section 1.2 above), both are regarded as being capable of generating valid data and outcomes.

By studying craniofacial anthropometry and related photogrammetric methods we can identify candidate landmarks suitable for measuring head posture and use the results of this investigation to inform the design of our algorithms and related models.

Additionally, as is reviewed in section 2.9, there is an abundance of anthropometric data and statistics available from several sources that can also be used to define constraints when searching for landmark parameters.

1.4 Landmarking and posture estimation from RGB-D images

There is already a mature research field in face landmarking for 2D images in controlled and un-controlled ("in the wild") (Wang *et al.*, 2015; Johnston and de Chazal, 2018; Deng *et al.*, 2019) with many datasets of annotated images available and open competitions encouraging research, for example the Menpo 300W faces in the wild challenge (Sagonas *et al.*, 2016). There is, however, no agreed single standard for facial landmark annotation and additionally some profile datasets use their own labelling standards with many points not corresponding to established anthropometric landmarks (Sagonas *et al.*, 2016).

There also appear to be few 3D datasets or 2D datasets with depth information, although a limited number of controlled environment, 3D head profile datasets have been generated (Yan and Bowyer, 2007).

As outlined earlier in this introduction, RGB-D profile images lend themselves to fast contour extraction. Couple this with the observation that many anthropometric landmarks have, necessarily, points of high curvature, then extraction of curvature from the profile contour and the analysis of this to locate landmarks would seem sensible. Indeed curvature has previously been used in analyzing 2D shape (Belongie, Malik and Puzicha, 2001), face recognition (Kakadiaris *et al.*, 2008) and in face gesture recognition (Pantic *et al.*, 2005).

Suitably landmarked training data can be used in training several ML regression algorithms including CNNs. The accuracy of these will depend not just upon the size of the dataset and the sophistication of the model but also upon the accuracy of the landmarking, and manual human landmarking will always be a source of measurement

error. This thesis notes that it may be possible to correct mislabelled landmarks by taking advantage of the underlying curvature properties of a profile. Such an approach is conjectured and investigated in chapter 5. Here, a novel, semi-automatic method is proposed that will allow the approximate placing of landmarks by a skilled operator followed by an automatic correction phase based on local curvature analysis. This annotated dataset may then be used to train selected algorithms as in (Wang *et al.*, 2015).

A curvature dataset parameterized on arc length opens up additional opportunities for time-series analysis. The literature here is mature and practitioners have already investigated the efficacy of time-series analysis methods in silhouette identification (Fawaz *et al.*, 2019).

1.5 Additional requirements

An attractive feature of a system capable of locating landmarks and using these to measure posture is the ability to explicitly follow the process by which a set of pixel values and depth measurements translate into a measured posture, that is, to answer the question "how does it do that?" This is a useful trait as it gives confidence in the reliability of a measurement and of future measurements. A neural network might be able to regress a landmark within an image but it is not yet possible to fully identify what, where or how distinct processes take place within the network and, given the non- linear nature of neural networks, whether points nearby in the input space will map to points nearby in the output space.

The trained network alone might be deterministic but, in isolation, it is not possible to know what it has been trained with. Whilst many would sacrifice deterministic confidence for accurate results most of the time, all other things being equal, it is nice to know how it does what it does, if only because one can replicate its action and thereby gain confidence in its reliability or otherwise.

1.6 Research Question

This introduction identifies the focus for this work and outlines approaches that will be of use in achieving the goal of creating methods for the fast and accurate measurement of head posture and the challenges involved.

Using depth information from a 2.5D camera it is possible to efficiently extract precise 2D geometric information of the shape of the head and shoulders in profile and so, given this assumption, the research question becomes:

"When used as features for deep DNNs, to what extent can curvature and its properties such as the first and second derivatives of a curve be useful in segmenting and regressing points on the occluding head profile?"

From this question aims, objectives and hypotheses can be identified.

1.7 Aims, Objectives and Hypotheses

1.7.1 Aims

The aim of this research is to:

> Explore extensively the suitability of curvature and its properties as features for fast regression and segmentation of parameterized plane curves, and in so doing, examine the effectiveness of these features in training deep neural networks to estimate head profile posture derived from 2.5D images.

1.7.2 Objectives

From this aim the following objectives can be enumerated:

1. To engineer and evaluate features derived from plane curves to train supervised machine learning models capable of efficiently segmenting regions of interest.

2. To develop a dataset of accurately landmarked head profile contours.

3. To develop and evaluate fast ML models capable of estimating head profile by segmenting profile contours into regions of interest and regressing key head profile landmarks.

4. Demonstrate that engineered features, in the context of this thesis, compare well with end-to-end ML approaches.

1.7.3 Hypotheses

From the research question and aim it is hypothesized that curvature and the related first and second derivatives of a curve can be efficiently calculated from a given plane curve and these features will enable the fast and accurate segmentation of a curve when used in conjunction with suitable ML models. A second hypothesis is that the same ML models can also be used to efficiently regress points on a plane curve with high accuracy and precision. A final hypothesis is that, in the context of this, these engineered features can produce results superior to an end-to-end ML approach.

1.8 Ethical Approval and Risks

Ethical approval was sought for the research and experimental work underpinning this thesis and, as such, was considered and approved by the University's ethics committee. A summary of the approved Ethics proposal is included in Appendix E.

1.9 Contributions to the literature

The following contributions are identified and enumerated below:

i) This study provides an evaluation of the effectiveness of curvature and related features used to *efficiently* identify regions of interest in a *uniformly* sampled sequential dataset. This includes the generation of new algorithms, software and tools to evaluate the accuracy of a recurrent neural network (RNN) that uses the features engineered here. Additionally, an investigation of the run-time efficiency of the engineered features is provided.

ii) An existing RGB-D dataset is extended, by extracting head profile contours so creating a *new* database of face profile contour curves. Additionally, a *new* set of manual annotations are generated identifying key anthropometric landmarks on both the RGB images and profile contour curves.

iii) A novel approach is developed in this context to automatically improve the accuracy of the annotations based upon the curvature properties of selected anthropometric landmarks.

iv) The findings of i) above are extended to work with the head profile contour dataset created in this study resulting in a new procedure that can accurately achieve fast face segmentation of head profile images. An evaluation of this procedure documents both the accuracy of the approach and its run-time efficiency when used with two RNNs.

v) The procedure used in iv) is further extended to develop a method to regress landmarks from the segmented profile contour and the accuracy and precision of this method is evaluated and documented.

vi) Finally, the effectiveness of an end-to-end learning approach is investigated using a one-dimensional temporal convolutional neural network (1DTCNN) to achieve the same goals of iv) and v) and the findings of this investigation presented with recommendations.

1.10 Layout of Thesis

Chapter 2 reviews previous work related to head posture estimation from perspectives of anthropometry and machine vision fields. Here, methods of extracting profile contours from 2D and 2.5D image data are reviewed and the theory behind curvature of a sampled plane curve is covered including methods used to efficiently calculate curvature and signal derivatives. Related work that uses curvature as a feature for classification and regression is identified and its relevance to this study assessed. Existing datasets used in both anthropometry and machine vision are also identified and the selection of suitable landmarks discussed. This chapter also includes essential theory related to ML and deep

neural network ideas used in this work and references more detailed discussions in the relevant appendices.

Chapter 3 provides a review of the research methodology and common methods used in the thesis and provides details and justifications for their use.

Chapter 4 applies the ideas of curvature and other related features to segmenting a uniformly sampled time series dataset. A RNN is used in conjunction with a range of engineered features and their run-time efficiency and segmentation efficiency are evaluated using established metrics and conclusions made.

Chapter 5 introduces the Notre Dame J2 dataset and details the methods used to extract profile contour information from the dataset. It includes a description of the manual annotation process and tools developed to generate a new dataset. Here, suitable anthropometric landmarks are selected for annotating the 2D RGB images and methods for extracting curvature and enhancing landmark accuracy are explained and implemented. These methods are evaluated, and their usefulness and limitations discussed. In the second part of the chapter, two RNNs are trained using a variety of features and the effectiveness of the segmentation process is evaluated, as previously for both networks, before settling on a best choice for features, network architecture and training hyper-parameters. Finally, the segmented profile contour is subjected to a further process to regress landmark positions and the accuracy and precision of the estimated landmarks is determined and evaluated using a held-out dataset.

In Chapter 6 a more complex 1DTCNN is used with a subset of features developed in previous chapters. Its accuracy, precision and execution speed are evaluated and compared with the approaches used in chapter 5. This chapter also examines the effectiveness of the end-to-end ML approach when compared with a more traditional methodology that emphasizes feature engineering.

Finally, in Chapters 7 and 8 we summarize our findings, make recommendations and present possible future work.

2 Literature Review

2.1 Introduction

This chapter begins by reviewing the notion of contours and contour curvature followed by an examination of how curvature can be calculated from 2D binary images. Limitations of 2D image capture devices are identified and the necessary image processing operations required prior to curvature calculation are then discussed. An overview of facial landmarking from two perspectives, anthropological and machine vision is provided. Useful datasets generated in both fields that are relevant to this thesis are identified. Classical optimising-based model fitting methods related to time series datasets and shape estimation are reviewed and then the chapter shifts attention to deep learning methods. The aim here is to recognize, review and contextualise selected methods pertinent to this thesis.

2.2 2D Face Landmarking and Feature Detection

2.2.1 Face Landmarking Databases

2D facial landmarking is a relatively mature field with many approaches developed to identify facial landmarks from two dimensional images. Facial landmarking algorithms take an un-labelled image as input and attempt to generate a list of landmarks specific to the algorithm's landmarking scheme. Typically landmarking methodologies rely on the existence of benchmark datasets with high quality annotated sets of landmarks, for example (Belhumeur *et al.*, 2011; Le *et al.*, 2012; Köstinger *et al.*, 2011; Sagonas *et al.*, 2013, 2016; Zhu and Ramanan, 2012) and more recently Deng, *et al.* (2019).

Many large scale face image databases exist that have limited or no landmarking, for example the Labelled Faces in the Wild database (LFW) (Huang *et al.*, 2008) and RGC-V2 (Phillips *et al.*, 2005). More recently the overwhelming success of recent deep learning methods has driven the development of large face and object databases. This began with the success of AlexNet in the 2012 ImageNet (Russakovsky *et al.*, 2015) object detection competition which saw the beginning of the current deep learning approach to face and object detection using CNNs together with multiple layers of feature

extraction and transformation. The success of AlexNet propelled a growth in face recognition DNN architectures such as Facebooks's DeepFace, DeepID, Google's FaceNet and VGGNet for example, which have been best performers in face recognition challenges. All these deep face recognition networks require large databases of images for training and testing. Most commercial face databases are private but the recent public release of the CASIA-Webface database (Wang *et al.*, 2007; Yi *et al.*, 2014) allowed researchers to train, evaluate and compare models. These databases usually have limited annotations, typically a bounding box to identify a face or, in the case of celebrity databases, a person ID too. So, researchers in face landmarking do have choices: use existing face landmarking databases, develop new databases of landmarked annotated images or annotate existing public databases with landmarks. The databases used in deep face recognition that have been identified here are covered in more detail by Wang and Deng (2021) which also provides a good survey of face recognition.

Putting aside databases that have not been annotated with face landmarks the following paragraphs concentrate on existing face landmarking databases and their landmarking schemes.

Sagonas *et al.* (2016) provides a clear overview of popular face landmarking databases categorised by the conditions under which the images were captured, namely controlled conditions databases and un-controlled conditions databases. The controlled conditioned databases are captured in indoor environments, often with controlled lighting conditions, background and camera position. Typically, there are several images per subject where the subject is asked to assume a range of facial expressions and there is no face occlusion, though where near side facing images exist there is, naturally, some degree of self-occlusion. Whilst Köstinger *et al.* (2011) focused on faces-in-the-wild for their AFLW database they also identify constrained databases as well as unconstrained databases in their review of existing databases.

Constrained condition databases identified by Sagonas and Köstinger include: Multi-Pie (Gross *et al.*, 2010), XM2VTS (Messer *et al.*, 1999), FRGC-V2 (Phillips *et al.*, 2005), AR (Martinez and Benavente, 1998) and The BioID Face Database (Jesorsky, Kirchberg and Frischholz, 2001). The unconstrained databases (often referred to as "in the wild") are typically images scraped from the web and manually annotated by local experts.

Popular in-the-wild databases include Annotated Facial Landmarks in-the-Wild (AFLW) (Köstinger *et al.*, 2011), Annotated Faces in-the-Wild (AFW) (Zhu and Ramanan, 2012), HELEN (Le *et al.*, 2012), IBUG (Sagonas *et al.*, 2013, 2016), Labelled Face Parts in-the-wild (LFPW) (Belhumeur *et al.*, 2011) and most recently the Mempo 2D and 3D Facial Landmark Databases (Zafeiriou *et al.*, 2017; Deng *et al.*, 2019).

The majority of these databases concentrate on annotating frontal or near frontal face images, though some such as Multi-Pie and the Mempo 2D annotated databases also include profile annotations using their own landmark configuration scheme. Occlusion is common in these databases.

2.2.2 Landmark Annotation Schemes

The list of face landmarks used when annotating a face is called the annotation scheme. There is no standard annotation scheme but the labels used by popular or successful algorithms appear to be re-used most frequently by subsequent researchers. The landmarks used include those derived from established anthropometrical practice as well as more ad-hoc choices.

Phimoltares, Lursinsap and Chamnongthai (2007), Çeliktutan, Ulukaya and Sankur (2013) and Wu and Ji (2019) also identify a range of commonly used landmarks, suggest suitable groupings and provide criteria for their selection, this is helpful to this study when identifying potential datasets and landmarking schemes for profile based landmarking datasets.

The fundamental characteristic of a good face landmark is its ability to be uniquely identified. This is true whether the chosen landmark is identified by an anthropometric practitioner or by an automatic landmark detecting algorithm. Such landmarks are called primary landmarks in the literature and will have clear features that aid their detection such as points of extremity, gradient information related to areas of high curvature, corners and edges, and texture or other local information. Where primary landmarks are unaffected by changes of expression or orientation they can be used as reliable landmarks or datums for guiding landmark search. Such primary landmarks are often referred to as fiducial landmarks in the literature. Other less obvious landmarks can still be identified but they rely upon their spatial relationship with nearby primary and fiducial landmarks.

These landmarks are often called secondary landmarks. There is some evidence that extending the number of landmarks used can improve accuracy for some methods, for example Milborrow and Nicolls (2008) note improved accuracy of their extended Active Shape Model when additional landmarks were added.

The number of selected landmarks differs significantly across databases, Multi-PIE, XM2VTS and the IBUG (200-W, faces in the wild) use 68 landmarks based on the Muli-PIE scheme; AR annotates 22 landmarks (Ding and Martinez, 2010); AFW uses only 6 (but these are multi-view and used for testing, with training of their method done on Multi-PIE); FRGC-V2 just 5; AFLW has 21; LFPW has 35 and HELEN uses 194.

Sagonas *et al.* (2016) illustrates the position of these landmarks as illustrated in Figure 2-1.

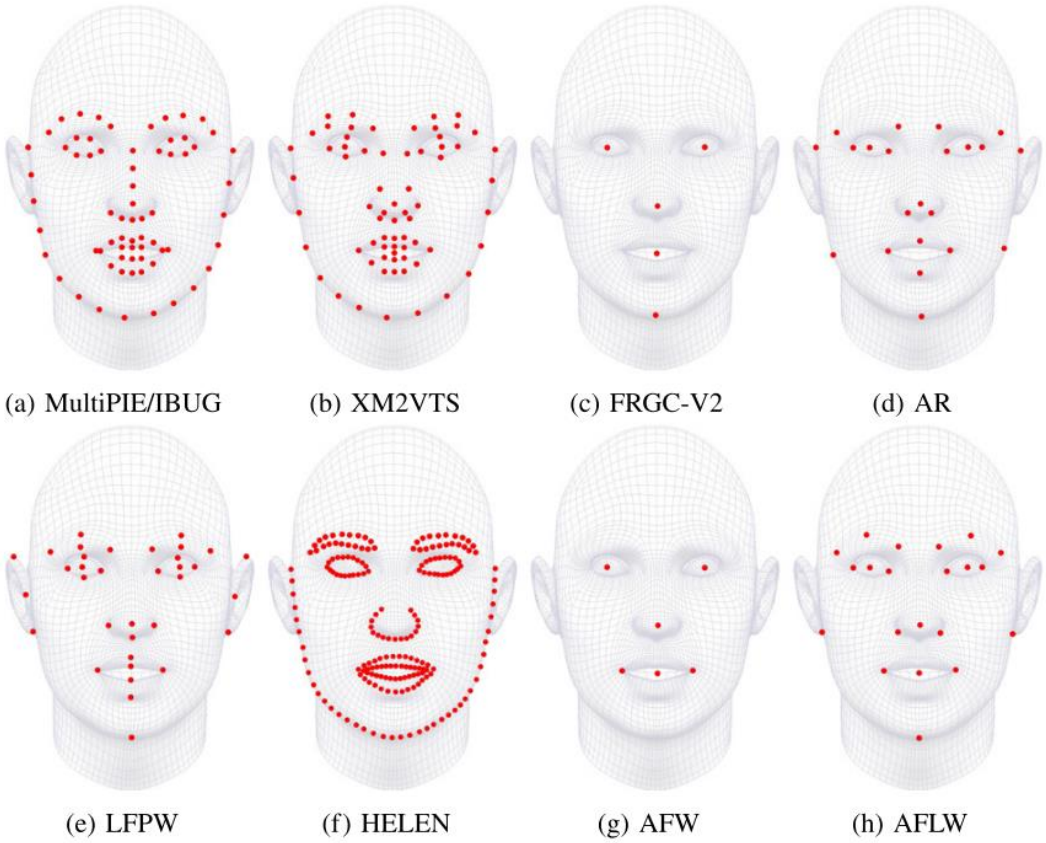


Figure 2-1 Popular landmarking schemes (Sagonas *et al.*, 2016).

Çeliktutan, Ulukaya and Sankur (2013) note 17 primary landmarks, known as the m17 landmarks in some literature. They correspond to the landmarks of the AR database after removal of the face's five outline landmarks.

A goal of this study is to accurately quantify *profiles* using suitable landmarks. The landmarks catalogued in Figure 2-1 identify potential profile landmarks, especially those landmarks intersecting the mid-sagittal plane, that is the vertical midline of the faces above. Ideally these will also be primary landmarks – that is landmarks that, in profile have high curvature and/or are points of extremity. Looking at the landmarking schemes shown in Figure 2-1, there are many landmarks along this vertical midline that may be suitable, and in particular the schemes used by MultiPIE/IBUG, XM2VTS, LFPW and HELEN appear to be good candidates. These databases could also be useful in evaluating the accuracy of the methods used later in this thesis to regress profile landmarks. It's clear that databases that include profile images might be suitable in assessing the performance of methods used in this thesis, assuming of course, the annotation of the databases is both accurate and consistent.

2.2.3 Face Landmarking methods and algorithms

Reviews of face landmarking approaches within the computer vision community have identified various categorizations to better present, classify, understand, and compare methods and techniques used. Phimoltares suggest five categories: geometry-based methods, colour-based approaches, appearance-based methods, motion-based methods and edge-based methods whilst Çeliktutan suggest two basic categories – model-based methods (or shape-based methods) and texture-based methods. Here model-based methods use the whole face image together with facial landmark groupings to help guide the algorithms in regressing landmarks. Typically, such algorithms use a pre-labelled training set of images. In comparison texture-based methods locate facial landmarks independently without the use of global information encoded in an idealised model and as Çeliktutan points out there can be overlap between these categories.

The model based methods would include, for example, those of Cootes, Edwards and Taylor (1998, 2001) and encompass Active Appearance Models (AAM) and Active Shape Models (ASM) (Cootes *et al.*, 1995) which, together with their variants (Milborrow and Nicolls, 2008; Tresadern *et al.*, 2009), have been used with great success since their

introduction in the mid 90's. AAM and ASM both use statistical information generated from the image to modify or morph an ideal model, fitting it to the image by optimizing the model parameters using optimization methods, for example gradient descent. The model based methods would also include neural network models where the model information is implicitly encoded in the network weightings.

The successes demonstrated by recent developments in neural networks have eclipsed AAM methods. For example, Johnston and de Chazal, (2018) note the first two 300W faces in the wild landmarking competition (Sagonas *et al.*, 2016) received few deep neural network submissions whilst the 2017 Menpo Facial Landmark Localisation Challenge (Zafeiriou *et al.*, 2017) comprised wholly of submissions based on deep learning methods and the organizers note that deep learning methods can lead to excellent results when trained with large datasets (Deng *et al.*, 2019).

A third categorization by Wu and Ji (2019) has three categories: holistic (generative) methods, constrained local models (CLM) and regression-based methods. The holistic methods model the appearance of the whole face in the image and include information about the shape of the face as defined by the face landmarks and their inter-relationships and includes AAMs identified above. CLM methods use local patch information and face shape information, and regression-based methods use either or both local patches and/or the whole face image.

Jin and Tan (2016) identify two classifications: holistic (generative) methods as described above and also discriminative methods which include CLMs, deformable parts models (DPM), cascaded regression and deep neural networks.

However one decides to group and classify these approaches, there are limitations to the methods used particularly in real-time scenarios where processor resources are limited. Álvarez Casado and Bordallo López (2021) note such limitations become problematic when dealing with occlusions, extreme poses, and so on.

As the review above indicates, a single, agreed classification standard does not exist. However, these attempts at creating taxonomies of facial landmarking methods can be useful here for several reasons, even though this study concentrates instead on contour

curves of the mid sagittal plane. They help to identify algorithms and methods based on similar approaches and specify characteristics and properties of the image data used by a method or algorithm. This, in turn, helps guide this study in promising directions, searching out methods or algorithms suitable for accurate landmarking given the available data.

The taxonomies described above concentrate on two dimensional methods to locate landmarks and regions of interest. In contrast, the data used in this thesis is a contour curve extracted from an image using depth information. As the taxonomies of facial landmark algorithms and methods discussed above assume two-dimensional, multi-channel, image data (three channels for colour or one for greyscale), then not all the methods identified and evaluated in the reviews are relevant, though some ideas remain pertinent to sequential data such as the need to select regions of interest (RoI) and to apply image smoothing operations prior to applying any landmarking algorithm. Additionally, the principles of adjusting and fitting features to a standard size (Procrustes methods) are relevant when dealing with sequential data in the form of contour curves, for example, as well as two dimensional images.

The excellent results obtained in the last few years using 2D images makes the use of deep neural networks both attractive and relevant to regressing facial landmarks using both two dimensional image information as well one dimensional data. Given the enormous success in 2D landmarking and the focus here on profile contours, section 2.8 further investigates suitable emerging neural network varieties that have been successfully used with sequential data sets.

The next section looks at profile contours and, interpreting this as a data sequence, reviews algorithms that have capability in feature generation for regression and region classification.

2.3 Profile landmarking and curve segmentation

A profile contour and its curvature, once extracted from a binary image, can be used as an input to an algorithm to regress selected facial profile landmarks. Promising methods used to locate landmarks might include classical optimization problems that locate local

maxima and minima within a curve. These can be used with additional information describing the relative ordering of local landmarks to regress co-ordinates on the binary image. Distance based similarity measures such as Euclidian and Mahalanobis distance are regularly used to estimate the similarity of the shape of a candidate curve with that of an architype, either locally on a sub-sequence of the contour, or globally. Such measures can be used here to quantify errors, similarity, and so on.

Interpreting the profile contour curve and its curvature as an abstract sequence of sampled data allows other well studied methods to be investigated. For example, there are several approaches to curve fitting found in the time series analysis literature that have already been used to solve similar problems. Cross-correlation is a simple approach that can work well on similar data sets and is an attractive idea, however it does not perform well when the series to be matched are at different scales and have been stretched or compressed in a non-linear fashion at a local scale. Since this non-linearity is evident in the phenotypic diversity of faces then more sophisticated methods are required here.

Dynamic time warping (DTW) (Sakoe, 1978) attempts to solve this problem. It uses a mapping of two vectors found by minimizing the distance between them using dynamic programming methods and has often been used successfully in time series analysis, for example in speech processing, handwriting and gesture recognition, time series clustering, protein sequencing and, as will be seen soon, in both electrocardiogram analysis and profile recognition.

Hidden Markov models (HMM) and other Bayesian methods are also used in time series analysis in similar areas to DTW. They have been a mainstay of classical speech analysis since the 1950s where they attempt to deal with scaling, stretching and compression of data samples by using the trained model's state transition probabilities to guide the evolution of state space.

More recently RNNs have successfully been applied to sequential data and have been responsible for the impressive advances seen in speech recognition. These methods currently eclipse in performance HMMs and related Bayesian methods (Graves *et al.*, 2006; Graves, Mohamed and Hinton, 2013).

2.4 Object Contours from RGB-D data

The primary advantage of using 2.5 dimensional RGB-D image data is the relative ease of extracting profile contours in a controlled environment. There are several proprietary RGB-D data formats in use and typically they include colour image RGB data stored in a suitable format, for example .png or .bmp file types together with depth information stored separately in a binary or text based format. The depth information may or may not be sampled at the same resolution as the colour image data, however each depth sample typically includes (*x, y, z*) spatial co-ordinate values for each point and may also include binary image data. The binary data consists of a set of binary true or false (1 or 0) values corresponding to image pixel values where a logical true (1) indicates that a valid depth sample has been captured and its related *x, y* and *z* value corresponds to width, height and depth, respectively. A logical false (0) indicates an invalid sampled pixel where no depth range information has been returned by the camera. Typically, this will occur when an object is beyond the range of the camera, when incident light has been attenuated or scattered and no reflected ray has been received by the camera, or when there has been interference between light reflected from multiple surfaces, see Microsoft (2019) for further details.

Binary data can also be created from valid depth pixels by removing all points which are to one side of an arbitrary two-dimensional plane. For example, assuming a right handed co-ordinate frame and a plane with a normal vector, $\underline{\hat{\boldsymbol{n}}}$ = (*0,0,1)* and a point, *p* on the plane, *p=(0, 0, -1)* then all points with a depth value, *z*, less than -1 can be selected and all others culled.

Once a binary image has been captured then its contour can be found. Moore's algorithm (Moore, 1968) as described by Woods and Gonzalez (2017) reliably finds the outer boundary of objects in a binary image where the background pixels are labelled with zeros and the foreground with ones. It is useful to place a boundary frame of logical false (zero) pixel values around the image to prevent errors should the object not be totally contained within the image. The resulting contour is a data structure containing a sequential list of (*x, y*) pixel locations describing the contour of the object.

This algorithm assumes a clockwise orientation and eight-connected pixel neighbours, that is any object pixel has at least one adjacent pixel that is either horizontally adjacent, vertically adjacent, or diagonally adjacent. Figure 2-2 shows an example binary image and the resulting, overlaid contour (in red) produced by an implementation of Moore's algorithm in C++ created for this study.

Figure 2-2: Moore's Boundary algorithm. Left: the binary image with Moore's algorithm applied. Right: A close up of spiral centre showing 8-connectivity.

The resulting contour curve can then be further processed to determine points of interest as discussed in section 2.3. In particular, points of prominence can be located, and areas of high curvature calculated using the ideas discussed next.

2.5 Curvature

In sections 1.4 and 2.3 it was noted that many anthropometric landmarks are placed at points of high magnitude of curvature so this section reviews the idea of curvature and describes how to determine the curvature at any point on a given contour. The calculated curvature at every point can then be used as a feature to describe and identify specific landmarks.

A plane curve is any curve that is contained in a two-dimensional plane. Here an equation for the curvature of a plane curve is developed together with a method to calculate the curvature at sampled points on the contour of a two-dimensional binary image. Sampled points present their own difficulties so an approach is proposed that will reduce the introduction of systemic noise related to the sample space.

The following review of curvature and the development of the curvature formula that follows is based upon established works on differential geometry, for example (Gray, Abbena and Salamon, 2017).

The arc length, *s* of a plane curve is defined as the distance between two points as one moves along the curve and the curvature, κ is given as dØ/ds where Ø is the tangential angle. Figure 2-3 shows this relationship.

$$\frac{\Delta \emptyset}{\Delta s} = \frac{\emptyset_2 - \emptyset_1}{s_2 - s_1}$$

and in the limit as $\Delta s \to 0$

$$\kappa = \frac{d\emptyset}{ds}$$

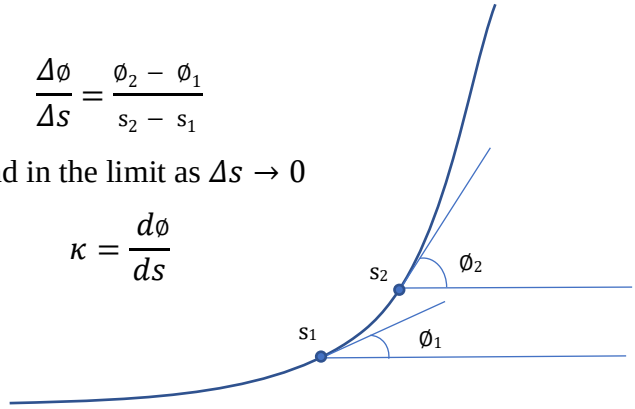


Figure 2-3: Curvature as the derivative of the tangential angle, Ø wrt arc length, s

If the plane curve is given by the Cartesian parametric equations *x=x(t)* and *y=y(t)* then, κ can be represented in terms of the parameter *t* as $\frac{\frac{d\emptyset}{dt}}{\frac{ds}{dt}}$.

Cauchy showed in 1826 (Gray, Abbena and Salamon, 2017; Kline, 1972) that

$$\frac{ds}{dt} = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \quad \text{so,}$$

$$\kappa = \frac{\frac{d\emptyset}{dt}}{\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}}$$

or using the notation,

$x' = \frac{dx}{dt}$ and $y' = \frac{dy}{dt}$ then,

$$\kappa = \frac{\frac{d\emptyset}{dt}}{\sqrt{x'^2 + y'^2}} \tag{2-1}$$

Next, note that the derivative $\frac{d\emptyset}{dt}$ can be represented using the identity $tan(\emptyset) = \frac{dy}{dx}$ and that $tan(\emptyset) = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}$, then,

$$tan(\emptyset) = \frac{y'}{x'} \tag{2-2}$$

Now let $g = tan(\emptyset)$ and differentiating with respect to t using the chain rule,

$\frac{dg}{dt} = \frac{dg}{d\emptyset}\frac{d\emptyset}{dt}$ then, $\frac{d(tan\emptyset)}{dt} = sec^2\emptyset\frac{d\emptyset}{dt}$ and re-arranging this for $\frac{d\emptyset}{dt}$ gives,

$$\frac{d\emptyset}{dt} = \frac{1}{sec^2\emptyset}\frac{d(tan\emptyset)}{dt} \tag{2-3}$$

Differentiating equation (2-2) using the quotient rule gives,

$$\frac{d(tan\emptyset)}{dt} = \frac{x'y'' - y'x''}{x'^2} \tag{2-4}$$

Combining equations (2-3) and (2-4) and noting

$\frac{1}{sec^2\emptyset} = \frac{1}{1+tan^2\emptyset}$ yields,

$\frac{d\emptyset}{dt} = \frac{1}{1+tan^2\emptyset}\frac{x'y'' - y'x''}{x'^2}$,

$= \frac{1}{1+\frac{y'^2}{x'^2}}\frac{x'y'' - y'x''}{x'^2}$ and,

$$\frac{d\emptyset}{dt} = \frac{x' y'' - y' x''}{x'^2 + y'^2} \tag{2-5}$$

Finally, combining equations (2-1) and (2-5) gives the curvature, κ of the curve at point ($x(t)$,$y(t)$).

$$\kappa = \frac{x' y'' - y' x''}{(x'^2 + y'^2)^{\frac{3}{2}}} \tag{2-6}$$

where $x', '', y' \text{ and } y''$ are the first and second derivatives of x and y with respect to the parameter t.

Equivalently and more intuitively the curvature, κ is given by $\kappa = \frac{1}{r}$ where r is the radius of the osculating circle, see Figure 2-4.

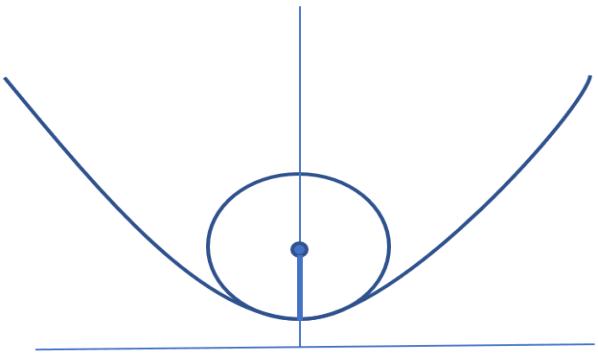


Figure 2-4: Curvature, $\kappa = \frac{1}{r}$ described by the osculating circle of radius, r.

To see this note that if a circle specified as $x = r\cos(t)$ and $y = r\sin(t)$ is tangent to a curve at a given point then

$$x' = -r\,sin(t)\,, x'' = -r\,cos(t)\,,\ y' = r\,cos(t)\ and\ y'' = -r\,sin(t),$$

and so from equation (2-6),

$\kappa = \frac{r^2 \sin^2(t) + r^2 \cos^2(t)}{(r^2 \sin^2(t) + r^2 \cos^2(t))^{\frac{3}{2}}}$, and therefore,

$$\kappa = \frac{1}{r} \tag{2-7}$$

where r is the radius of the osculating circle.

Curvature can also be found numerically by using the radius of curvature by fitting an osculating circle to each point in the curve. Fitting can also be performed using a linear least-squares regression technique (Hopp *et al.*, 2015).

It is worthwhile pointing out that the curvature is dependent on both first and second derivatives and these properties have been used alone by some authors to identify face landmarks with moderate success. Lippold *et al.* (2014) compared first and second derivatives of an outline face profile and reported successful landmark localization using just the second derivative. Pantic, Patras and Rothkruntz (2002) uses first and second derivatives individually to locate landmarks but also uses a priori knowledge of the positions of landmarks and their convexities and concavities. Also, the first and second derivatives of points within a 2D image are used extensively in traditional computer vision and image processing fields for example in filtering and edge detection, in Canny edge detection (Canny, 1986), with the Sobel operator (Szeliski, 2010), etc.

Curvature has been used already to identify landmarks. Efraty *et al.* (2009) calibrate and enhance the manual landmarks using an automatic process that assumes the furthest point, P1 is the point of highest curvature, as shown in Figure 2-5. Here, Pa and Pb form a vector located on the plot of the curvature calculated from the profile. Pa and Pb must lie on either side of P1. Their position is determined by handcrafted rules.

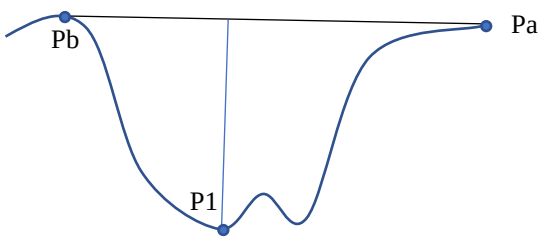


Figure 2-5: Approximating highest point of curvature between two points, after (Efraty *et al.*, 2009).

Bottino and Cumani (2008) find landmarks in profile by using curvature also. Their algorithm also uses a handcrafted approach that identifies landmarks by knowledge of their relative positions, for example the stomion is between the labiale inferius and labiale superius. The algorithm then identifies points of opposite curvature, i.e., concavity and convexity, to locate the landmark position. Their process requires the orientation of the head profile to be first approximated and three reference points, the nasion, pronasale and gnathion, located prior to further landmark localization. This requires additional pre-processing of the profile, for example they need to find the two-dimensional centre of gravity of the profile and uses further hand-crafted rules to identify extremities prior to applying their approach.

Bhanu and Zhou (2004), and Zhou and Bhanu (2005) have also focused on curvature as a means to recognize faces from profiles in both still images and images extracted from video. They use Gaussian functions and convolution to smooth and extract derivatives for curvature calculation of the entire profile before applying dynamic time warping to match a face's profile with a database of image profiles.

Lipoščak and Lončarić (1999, p. 245) use the curve and its first and second derivatives to create a scale space interpretation of a face profile using Gaussian functions. Significantly, They note that "at any value of variance (σ^2), the extrema in the n^{th} derivative of the smoothed signal are given by the zero-crossings in the $(n+1)^{th}$ derivative." This means it is possible to locate high points of curvature using both the curvature of a point on a curve and also derivatives of the smoothed curve.

The ideas discussed here are used extensively throughout this thesis where Gaussian filters are used as a means to smooth and differentiate curves simultaneously to create suitable features for supervised machine learning classifiers.

In this study curvature as a feature (amongst others) is used to identify common anthropometric landmarks and so the aim is to match the accuracy of an expert annotator who has the tools to identify a landmark at pixel resolution. Pragmatically it makes sense to use all the information available, so initially in this study, curvature will be calculated at every sampled point on the smoothed contour. Hence it is useful to represent the plane curve by the Cartesian parametric equations $x=x(t)$ and $y=y(t)$ as detailed in the preceding

proof. If follows that, the parameterized curve, *C* will be given by *C(t)* = (*x(t)*,*y(t)*) where $t \in \mathbb{N}: t <= N$ and N is the number of sampled points in the contour. Equation (2-6) can then be used to solve for κ where the parameter *t* represents the sampled arc length from the beginning of the sampled curve. To do this the first and second derivatives of both *x* and *y* co-ordinates with respect to arc length need to be calculated.

Finding derivatives of functional plane curves at given points parameterized by arc length for example, requires a closed form representation or a series of piece-wise fit functions that are continuous, smooth, and which clearly need to be twice differentiable. Unfortunately, the contour datasets used in this thesis, and in the vast majority of image processing systems, are sequences of points sampled from 2 dimensional images, so before considering the calculation of curvature at any point on the curve, the question, "how can we reliably and effectively calculate derivatives on the sampled contour curve" needs to be answered. The answer to this can be found in the relevant literature and the following sections detail several well studied approaches that are used extensively in the fields of numerical methods and image processing.

2.6 Derivatives of a sampled curve

There are several numerical methods that can be used to find the derivatives of planar curves. This section considers three popular methods and outlines their advantages and disadvantages before selecting the most suitable given the problem context.

2.6.1 Finite differences

Finite differences can be used to approximate derivatives numerically. In the limit the derivative of a function *f(x)* is defined as,

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

If *h* is nonzero and small then the derivative is approximated as,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{2-8}$$

and this is known as the *forward difference*. Similarly, the *backward difference* approximates the derivative as,

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \tag{2-9}$$

and the *central difference* approximates the derivative as,

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{2-10}$$

Taylor's theorem can be used to estimate the order of the error of the forward, backward and central difference methods. Appendix B provides details of how this is achieved.

In particular the error in approximating the first derivative using the forward difference method is *O(h)* and so is proportional to the step size, *h*. Here, halving the step size will halve the error associated with finding the derivative. Similarly, for the backward difference method the approximation error is also *O(h)* and here, too, halving the step size also halves the error.

When calculating the first derivative using the central difference method, however, the error is of order $O(h^2)$ so halving the step size will make the error associated with the derivative four times smaller. Therefore, the central difference approach to finding the gradient is more accurate for small step sizes and so is the preferred of these three methods.

The central difference method can be implemented by convolving the sampled image with the kernel [-1/2 0 1/2] but the small kernel size can lead to errors when estimating the shape of a sampled curve. Step changes at the pixel level within a sampled contour lead to errors in calculating the derivative at finer scales and so low pass filtering of a sampled curve is advisable prior to calculating its derivatives (Farid and Simoncelli, 2004). Bouma, et al. (2007) also suggest that the method used to find the derivative should be rotation invariant as well as twice differentiable as otherwise the shape of the pixel grid will be measured instead. This leads to noisy and inaccurate curvature calculations.

Gaussian filtering and Gaussian derivatives are discussed next as they not only have these properties, but they also benefit from other qualities useful for filtering and, as it turns out, for fast curvature estimation. Gaussian derivatives also have the additional benefit of efficiently smoothing the sampled contour as the convolution kernel used encodes both the smoothing and derivative operations as discussed next in 2.6.2. Finally, when represented as a convolution kernel, their size can easily be adapted to suit the image scale.

2.6.2 Gaussian Filters and Derivatives

The Gaussian function has several attractive properties that make it a useful differential operator and filter (Marr and Hildreth, 1980). It is smooth and band-limited in the frequency domain and exhibits spatial localization. It is also smooth and localized in the spatial domain. The Gaussian filter is the only filter capable of this since the Fourier transform of a Gaussian is another Gaussian. There are many other interesting properties of the Gaussian and its derivatives. This chapter covers properties and theory relevant to this thesis and many of the ideas are covered in existing Computer Vision texts. Romeny's text, Front-End Vision and Multi-Scale Image Analysis (Romeny, 2008) acts as a good reference and several of the ideas covered here are to be found in this text.

The Gaussian function is also separable, that is, a bi-variate Gaussian function can be represented as the product of two, uni-variate Gaussians, and the two-dimensional kernel of a bi-variate Gaussian is also separable into two, one-dimensional kernels. This property is regularly used in the image processing and computer vision community when filtering a 2D image as it reduces the time complexity of this operation from $O(n^2)$ to $O(n)$. Since this thesis concentrates on processing contour curves embedded on a two-dimensional plane it is worth while emphasizing that a contour curve is represented in this study as a list of (x, y) co-ordinates and that 1D Gaussian kernels are applied to each co-ordinate separately when calculating derivatives and curvature.

A univariate Gaussian is defined as,

$$G^{\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where σ^2 is the variance, σ the standard deviation (measured in pixels) which controls the scale and μ is the average (expected) value which acts to offset the function along the independent variable axis, x. The Gaussian filter is symmetric and if centred at zero, μ is zero giving,

$$G^{\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}} \tag{2-11}$$

The bi-variate Gaussian is given by

$$G^{\sigma}(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x+y)^2}{2\sigma^2}} \tag{2-12}$$

and the Gaussian filter can be extended into higher dimensions as required.

Filtering is usually performed by creating a kernel of the sampled Gaussian and convolving this Gaussian kernel with the sampled dataset. In this study, a *contour* extracted from a binary image is to be filtered. Recall from section 2.5, the contour is represented as a parameterized curve given by $C(t) = (x(t),y(t))$ where $t \in \mathbb{N}: t <= N$ and N is the number of sampled points on the contour.

Smoothing C is achieved by convolving with a suitable Gaussian kernel, G^{σ}. We convolve $x(t)$ and $y(t)$ with G^{σ} separately to give us $x_s = G^{\sigma} * x(t)$ and $y_s = G^{\sigma} * y(t)$.

To find the first and second derivatives of x_s and y_s we note that differentiation of x_s with respect to t can be written as $\frac{d}{dt}(x_s) = \frac{d}{dt}(G^{\sigma} * x(t))$ and so from the commutative properties of convolution,

$$\frac{d}{dt}(x_s) = x(t) * \frac{d}{dt}(G^{\sigma}) \tag{2-13}$$

and by the same argument,

$$\frac{d}{dt}(y_s) = y(t) * \frac{d}{dt}(G^{\sigma}) \tag{2-14}$$

So, *both* differentiation and smoothing may be achieved with just one convolution on the *unfiltered*, sampled contour curve, *C*. The kernel that represents the Gaussian derivative, $\frac{d}{dt}(G^{\sigma})$ can be constructed efficiently from the coefficients of the Hermite polynomials or by sampling the derivative curve, $\frac{d}{dt}(G^{\sigma})$. We will refer to the Gaussian Derivative as the "Derivative of the Gaussian" and use the abbreviation, *DoG,* to remain consistent with existing image processing literature.

Analytically, the *DoG* is obtained by differentiating the Gaussian,

$$\frac{d}{dt}(G^{\sigma}(x)) = \frac{d}{dt}\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}}\right) \tag{2-15}$$

So,

$$DoG = \frac{x}{-\sigma^2\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}} \tag{2-16}$$

Similarly, the second order derivative of the Gaussian is obtained by a second differentiation of the *DoG*. This is referred to in the literature as the Laplacian of the Gaussian, *LoG*.

$$LoG = \frac{(x^2 - \sigma^2)}{\sigma^4\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}} \tag{2-17}$$

The scale of the Gaussian smoothing operator is given by σ and needs to be chosen carefully to ensure that the range of pixels involved in the smoothing operation is neither too small nor too large. If it is too small, then the derivative kernel will introduce noise based upon the sampling grid structure, too high then the features of interest will be removed. The choice of this value depends very much upon the image sample grid and the regions of interest. Where the scale of the regions of interest differs, σ and hence the smoothness of the Gaussian can be adapted.

The size of the kernel depends upon the choice of σ. Since the tails of the Gaussian are close to zero beyond $\pm 3\sigma$ there is little point in using kernel coefficients beyond this range.

Truncating coefficients beyond this range keeps the calculation computationally efficient and introduces little error. The number of coefficients will naturally be odd to ensure symmetry around the smoothing point and so the value of 3σ should be rounded up accordingly to the nearest odd number. The coefficients themselves will need to sum to unity to prevent any scaling errors. The final kernel will therefore range from -3σ to $+3\sigma$ and will have a size of $6\sigma + 1$.

Choice of σ can be empirically estimated but should the number of pixels of the region of interest be known through prior observation of typical image contours, for instance, then the standard deviation, σ can be estimated using the relationship between standard deviation and the notion of bandwidth. For the Gaussian filter, the bandwidth is often described by the term "the full width at half maximum" (FWHM). The FWHM is $2\sigma\sqrt{2ln2} \approx 2.355\sigma$ pixels.

The scale of the filter is given by σ and so it is helpful when experimenting with the effect of smoothing upon a region of interest to note that the repeated application of consecutive Gaussian kernels can be made more efficient since;

$$G^{\sigma 1} * G^{\sigma 2} = G^{\sqrt{\sigma 1^2 + \sigma 2^2}} \tag{2-18}$$

Here $G^{\sigma 1}\ and\ G^{\sigma 2}$ represent Gaussian functions and the superscripts $\sigma 1$ and $\sigma 2$ represent their standard deviations. Note convolution of the Gaussian functions result in another Gaussian, $G^{\sqrt{\sigma 1^2 + \sigma 2^2}}$ with a standard deviation of $\sqrt{\sigma 1^2 + \sigma 2^2}$. This property can be used to improve calculation efficiency when applying repeated convolutions on an image or other function. To see this imagine the convolution $C_{\sigma 1} = C_0 * G^{\sigma 1}$, with $(\sigma 1 < \sigma 2)$ where C_0 is the curve discussed in the previous section, had already been calculated and next a convolution at a scale $\sigma 2$, was required. Instead

of performing the convolution $C_{\sigma 2} = C_0 * G^{\sigma 2}$, instead $C_{\sigma 2} = C_{\sigma 1} * G^{\sqrt{\sigma 2^2 - \sigma 1^2}}$ is calculated since $G^{\sigma 1} * G^{\sqrt{\sigma 2^2 - \sigma 1^2}} = G^{\sqrt{\sigma 1^2 + \sigma 2^2 - \sigma 1^2}}$, and so,

$$G^{\sigma 1} * G^{\sqrt{\sigma 2^2 - \sigma 1^2}} = G^{\sigma 2}$$

This is more efficient because the kernel size of the Gaussian with $\sigma = \sqrt{\sigma 2^2 - \sigma 1^2}$ is smaller in the number of its elements and so the convolution is faster.

This section has covered methods to efficiently and accurately smooth, differentiate sampled curves and subsequently determine curvature. These methods will be used in this thesis later to develop algorithms to efficiently calculate features used with supervised ML models. The next section covers key ML concepts that are used in this thesis, including metrics used to evaluate a ML model's quality.

2.7 Evaluation metrics in machine learning

Machine learning models can be classified in numerous ways. This thesis aims to develop machine learning models that can segment samples prior to regressing landmarks. This implies the use of a model that can classify a sequence of samples into one of several discrete classes. A contiguous group of like classes is regarded as a region which has been segmented. Additionally, a labelled dataset will be used to train the classifier. Such a training method is called supervised training. Hence the machine learning model used is a classifier and the learning method is supervised training.

In order to assess the quality of the classifier several approaches could be used. Kamiri and Mariga (2021) provide a good review of machine learning methods and identifies several methods and metrics used in evaluating ML models. These methods are used in many disciplines and consequently terminology and presentation of results is not consistent across fields. Appendix D brings together and details methods and metrics used to evaluate machine learning models used for classification in this thesis. It outlines concepts, terms and the choice of formatting and presentation of data as they are used in this thesis, and draws from several sources including the Scikit-learn python project (Pedregosa *et al.*, 2012), TensorFlow (Abadi *et al.*, 2016), Keras documentation (Chollet

and others, 2015), Géron (2019), Russell and Norvig (2020) and Bishop (2006). The following methodology chapter provides details of the choice of metrics and their application in this thesis.

2.8 Machine Learning with Neural Networks

This section provides an overview of related work covering the use of the perceptron, RNNs, CNNs and 1DTCNNs, focusing on their use with sequential data and time series signals.

The introduction to this thesis and the literature review so far has already identified the relevance of deep neural networks and their recent successes in areas such as face recognition and face detection. The successes of the past decade have been attributed to the convergence of existing theoretical ideas with several technological advances:

- The ubiquity of affordable and powerful computing technologies, especially graphics programming units (GPUs) driven by the rise of computer games and computer generated imagery (CGI) in general.
- The Internet and worldwide web;
- Availability of open source, powerful programming languages together with well designed software libraries that enable the development of ML applications and DNNs, for example Python and ML packages such as Keras, Tensorflow, Café, PyTorch and Scikit-learn.

However, the theoretical concepts underpinning deep neural networks had been developed some decades before. In fact, the area of neural networks has a long history within the subject of computer science (Russell and Norvig, 2020) beginning with work done by McCulloch and Pitts (1943); the formalisation of neural networks within the broader field of AI beginning at the seminal 1956, Dartmouth Summer Research Project on Artificial Intelligence (McCarthy *et al.*, 2006); and the introduction of the perceptron by Rosenblatt (1958). However, it was the development of the efficient back propagation algorithm in 1970 by Seppo Linnainmaa (1976), although not focused on neural networks (Schmidhuber, 2015), and its popularisation by Hinton and Rumelhart (Rumelhart *et al.*, 1986) in the 1980's that kicked started the renaissance of neural networks and the growth of deep learning techniques.

The past decade saw many successful applications of deep neural networks in many areas but especially in two domains: sequential data processing and two-dimensional image classification. RNNs, often in the form of long short-term memory (LSTM) RNNs (Gers and Schmidhuber, 2000; Hochreiter and Schmidhuber, 1997), revolutionised sequential data processing in numerous fields from speech translation and recognition to time series prediction and healthcare applications, whilst CNNs (Fukushima, 1980; LeCun *et al.*, 1998; LeCun *et al.*, 1989) have been shown to be capable of impressive image recognition and detection on a par with human ability (Krizhevsky, Sutskever and Hinton, 2012; He *et al.*, 2016). Since their introduction the CNNs have also been used to solve sequential and time series related problems with some success, either in their own right or in combination with RNNs (Shi *et al.*, 2015; Bradbury *et al.*, 2016; Oord *et al.*, 2016; Sejnowski and Rosenberg, 1987). Bai, Kolter and Koltun (2018) provide a detailed comparison of DNNs and CNNs used for sequence modelling and, for instance, describes the architecture of 1DTCNNs suitable for sequence processing.

The following sections briefly review RNNs and 1DTCNNs that are suited for achieving the aims of this study. A more detailed explanation of the underlying theory and operation of these DNNs is provided in appendix C of this thesis.

2.8.1 RNNs for sequential data

Unlike feedforward networks whose connections are constrained to link forward to neurons deeper in the network, RNNs include links feeding back to earlier parts of the network. Interpreting these feedback links as outputs from a previous time step, and chaining these stages together, allows learning across several time steps. Hence the network can learn patterns across time or sequences of data.

Typically, RNNs are used in a number of configurations: vector to sequence; sequence to sequence; sequence to vector or as an encoder-decoder, that is sequence to vector followed by a vector to sequence. These architectures and their uses are detailed in Appendix C and in several texts, (Goodfellow, Bengio and Courville, 2016; Géron, 2019) and so their detailed theoretical underpinning is not repeated here.

Sequence to sequence and sequence to vector RNNs allow an output sequence to be learnt from a labelled dataset of input sequence vectors. The output sequence can be configured to be of a different length to the input sequence or it could be the same length, depending on the problem scenario. Variable lengths can be accommodated by, for example, padding, however the majority of modern neural network libraires include options for variable length sequences (Chollet and Others, 2015; Abadi *et al.*, 2016; Mathworks, 2020a).

An aim of this thesis is the segmentation of a contour curve, and so a sequence-to-sequence network is a candidate worth considering. Here the input and output sequences are the same length, although, depending upon the number of features used the dimension of the input sequence vector may change. For example, a univariate time series or a scalar feature such as curvature would have a dimension of 1, a profile's x and y co-ordinates, on the other hand, would have a dimension of 2. Additionally, the length of each training or test sequence can also be varied.

RNNs can be used for segmentation by adding additional layers to the output as is common in other neural network architectures. Typically, a fully connected layer followed by a softmax layer is used to convert the real valued output state to a finite number of classes. A probability value is associated with each of the multiclass outputs and a decision boundary is chosen (usually a probability of 0.5 or greater is regarded as true). The output sequence may consist of a sequence of labels classifying each input of the sequence. For example, a single input from a long sequence might be a co-ordinate (x,y) pair representing a point on a curve, whilst the corresponding output at that time would be a region label, for example, "upper lip".

RNNs have a limited memory, the literature often states 10 time/sequence steps and attribute this to both vanishing gradients or exploding gradients that cause some oscillation in the gradient magnitudes during back propagation. This led to the development of LSTM RNNs that attempted to address these limitations. LSTMs introduce a memory component that allows internal state to be remembered or forgotten through the use of a gate structure and expands the internal states to include a hidden state h() vector (also referred to as hidden units) and an additional cell state, C() vector. The

hidden state corresponds to the short-term memory and the cell state to the long-term memory.

There are several variants of LSTMs, for example the Gated Recurrent Unit (GRU) (Cho *et al.*, 2014), that aims to simplify the architecture of the gate. This is currently a very popular network choice. Staudemeyer and Morris (2019) suggest GRUs outperform LSTMs, however Shewalkar (2019), for instance, presents empirical evidence that suggests they are comparable in speech applications, though the GRU can be faster to train.

LSTMs, with their ability to learn longer sequences and their previous successes in segmenting sequential data, are a good choice for segmenting profile contours and so their use can contribute toward achieving the aim of this thesis.

The next section reviews the use of CNNs in processing sequential and time series data, focusing on 1DTCNNs.

2.8.2 Sequential data processing using convolutional neural networks

A two-dimensional CNN layer, as used in image detection, typically uses many small kernels or filters that are moved across the input image, performing convolutions at each pixel. Each kernel's convolutions creates a single two-dimensional feature map as it moves across the image. Together, these feature maps form a single layer. The kernels each encode a particular feature that is representative of part of the image to be recognised. As each kernel has effectively scanned the image looking for its feature, the resulting feature map encodes where that feature (or multiple versions of it) lies within the image. Hence the use of the word "map." Multiple scales are accommodated by adding a pooling layer afterwards to downsize or subsample the layer above. Hence a CNN has the ability to localise an object within an image and at multiple scales. The back propagation algorithm is used to train the kernels. More detailed descriptions of CNNs and their variations are provided in Goodfellow, Bengio and Courville (2016) and Géron (2019).

One dimensional CNNs work in a similar manner to the more common two dimensional CNNs. Convolutions become one dimensional in nature and multiple layer subsampling

occurs, and is referred to as dilation. This property is useful where a range of features at differing scales need to be recognised, as is the case in this study, where the aim is to segment the head profile contour.

The wavenet architecture introduced by Oord *et al.* (2016) apply these ideas and additionally includes a temporal feature that prevents looking ahead. This sequential, ordinal property is useful where features to be segmented are constrained by their order, as is the case also for head profile contours. For example, the lower lip is below the upper lip, which is itself below the nose. This, together with the multi-scale property described above also makes this DNN a good candidate for segmenting the head profile contour studied in this thesis.

Bai, Kolter and Koltun (2018) provide a detailed explanation of the operation and architecture of 1DTCNN networks. As with the LSTM, the 1DTCNN performs classification using a fully connected output and softmax layer. Appendix C of this thesis also provides further details of 1DTCNNs along with a description of their architecture.

2.9 Anthropometry and facial landmarking

Section 2.2.2 reviewed face landmark annotation schemes used in the computer vision community. Recall, also, the introduction to the thesis included a brief overview of anthropometry. This section builds on these ideas, focusing on the anthropometric literature related to facial landmarking and identifies potential landmarks, statistics and related datasets that may prove useful in achieving the aim of this thesis related to landmark localisation. With this in mind, this section also reviews the literature related to indirect anthropometric methods used in photogrammetry in order to assess the reliability of those methods.

Anthropometry is the study of the measurements and proportions of the human body and can be divided into two branches, craniofacial anthropometry whose focus is the measurements of the head and face, and somatometry which is the measurement of the body (Salkind, 2007).

As anthropometric measurements rely on the accurate and repeatable identification of landmarks, then it would be sensible to select candidates from the set of established landmarks studied in the anthropometric literature. The observation made in section 1.4 is worth reiterating here, that points of high curvature correlate remarkably well with craniofacial anthropometric landmarks.

To help inform the choices of candidate landmarks used in this study, a very brief history of anthropometry is provided next and is followed by a survey of modern head and upper body anthropometry and the methods and tools used to capture measurements.

Anthropometry has a long history. The techniques used were originally developed by Johann Sigismund Elsholtz in 1654 (Kolar and Salter, 1997), and his ideas were still used extensively in the 19th century to quantify human development and morphology. During this time the work of Paul Broca in the field of physical anthropology and that of the French policeman Alphonse Bertillon, who was searching for a reliable method to identify criminals, helped standardise physical measurements and practices. As the 19th century progressed a separate German school of Anthropometry had emerged which was formalised in the 1882 Frankfurt Convention and it is worth noting that this convention gives its name to the "Frankfort horizontal plane" which is a reference line or plane still in use in modern craniofacial anthropometry (Salkind, 2007, p. 36).

During the early 20th Century anthropometry was linked with pseudo-scientific ideas related to scientific racism and the now discredited eugenics movements and consequently its use in the study of human populations declined, however the practice of anthropometric methods in the study of human growth and development continued and its application in other disciplines increased. Anthropometric tools and techniques are now used in ergonomics, biometrics and security, in several medical disciplines including physiology, dysmorphology, dentistry, surgery, physiotherapy, anthropological medicine, kinanthropometry, forensic science and so on.

An important element of modern anthropometry is quantifying norms of human measurements to facilitate further comparative investigations using appropriate statistical methods. Modern cranio-facial anthropometry has been fuelled by the clinical community's desire for high quality datasets of population norms. Uses for these datasets

include understanding disease, informing plastic surgery and so on. Accordingly, research groups have created datasets of anthropometric data of the head and body (Farkas, 1994; Robinette *et al.*, 2002; Lipira *et al.*, 2010; Gordon *et al.*, 2014; Weinberg *et al.*, 2016) that use a variety of technologies and tools to capture the data. Typically, researchers report summary statistics of standardised measurements of the human face, and some include additional collections of 3D normative craniofacial images. Some published datasets include both craniofacial and somapometric results, the US Army's Anthropometric Survey of Army Personnel of 2012 (ANSUR 2012) (Gordon *et al.*, 2014) and the Civilian American And European Surface Anthropometry Resource (CAESAR) (Robinette *et al.*, 2002) are good examples. As new methods have evolved for capturing anthropometric data researchers have been keen to compare the efficacy of these emerging methods (Farkas, Bryson and Klotz, 1980; Weinberg *et al.*, 2004; Robinette and Daanen, 2006; Ghoddousi *et al.*, 2007; Dindaroğlu *et al.*, 2015). A principal limitation of all these datasets relates to the paucity of ethnographic data. Most datasets concentrate on white Caucasian ethnography which is a problem recognised by researchers in the field and attempts are being made to address it (Weinberg *et al.*, 2016).

For the purpose of this review these methods can be separated into three broad categories: those derived from digital 3D imaging technologies, those derived from 2D images and those derived using direct (manual) anthropometric methods and instruments such as callipers, goniometers and so forth. The first two methods are indirect methods and are forms of 3D photogrammetry and 2D photogrammetry, respectively. The third method is termed direct anthropometry.

2.9.1 Direct and indirect anthropometry

Modern direct craniofacial anthropometry was pioneered by Leslie Farkas (1915-2008) (Naini, 2010) who developed an empirical system of facial measurements and over his career compiled a huge database of craniofacial "norms." He was the first to recognise and subsequently emphasise the importance of the relative proportions of craniofacial measurements. In direct anthropometry, measurements are made using callipers, goniometers, rulers, and tape measures and these rely on trained and experienced practitioners to obtain accurate and repeatable results. This process is time-consuming and requires direct contact and so also the compliance of the subject (Jayaratne and Zwahlen, 2014). Despite these difficulties direct anthropometry remains the touchstone

against which other methods are compared, for example evaluations of 3D stereo-photogrammetry and 2D photogrammetry.

Photogrammetry is the art, science, and technology of obtaining reliable information about physical objects and the environment through processes of recording, measuring, and interpreting photographic images and patterns of recorded radiant electromagnetic energy and other phenomena (McGlone, 2004). Farkas, Bryson and Klotz (1980) studied the reliability of 2D photogrammetry in anthropometry as long ago as 1980 and its use in cranio-facial anthropometry is well documented in the literature (Aldridge *et al.*, 2005; Ozsoy *et al.*, 2009; Aksu, Kaya and Kocadereli, 2010; Dindaroğlu *et al.*, 2015). Fundamental to its use is the correct calibration of the camera.

Software based calibration processes are used to correct distortions generated by the camera's optical system. 2D photogrammetric measurement software, for example, often runs on desk-top machines and uses a range of plug-in cameras, so correct calibration remains an important procedure in setting up such software. Typically, this process uses a standardised object, for example, a printed chessboard pattern whose image is captured in various orientations and transformations on the image are then calculated that remove distortion generated by the camera optics. On the other hand, RGBD cameras and self-contained, proprietary photogrammetric instruments are typically calibrated during manufacture and may need little user calibration, or, where necessary, occasional user based calibration using supplied calibration equipment. The use of 2D and 3D photogrammetric instruments in anthropometry are well studied, and typically these studies compare their results with measurements made using direct anthropometry (Nechala, Mahoney and Farkas, 1999; Aldridge *et al.*, 2005; Weinberg *et al.*, 2006; Ghoddousi *et al.*, 2007; Ozsoy *et al.*, 2009; Dindaroğlu *et al.*, 2015; Kim *et al.*, 2015). All conclude that the accuracy and repeatability of the photogrammetric methods compare favourably with direct anthropometry, a significant finding for this study as it supports the use of indirect 2D and 3D photogrammetry in measuring head posture.

Direct physical contact between the subject and the measuring instruments can lead to underestimates as pressure applied by the instrument will cause indentation of the soft tissue. Use of photogrammetric methods avoids this potential operator error. On the other hand, a significant advantage of direct anthropometry is the opportunity to locate

landmarks using soft tissue palpation. Indirect photogrammetric methods may exclude landmarks if palpation is necessary to locate them or require them to be first located using soft tissue palpation and the skin marked accordingly (Weinberg *et al.*, 2004; Ghoddousi *et al.*, 2007). Where palpation is necessary, crudely marked landmarks may also be a source of measuring error (Farkas, 1994). Consequently, landmarks that require palpation for accurate location may not be the best candidates for use in this thesis.

In direct as well as indirect photogrammetry Ozsoy *et al.* (2009, p. 288) note "..accurate location of landmarks and user skill are important factors to achieve reliable data" and "the period of interaction with the subject is potentially shorter" with indirect methods. Consequently, an automatic, indirect method of identifying landmarks would provide an additional advantage and this itself is a motivating factor in this research.

Whilst photogrammetric methods can produce accurate results, typically equipment used does require some user interaction to define and locate landmark positions, measure distances and angles, and so on. A 2016 study by Kuehnapfel *et al.* (2016) observed that whilst the 3D scanning equipment used produced generally excellent reliability with comparable intra-rater and inter-rater results, overall it was slightly more time consuming to use but was better accepted than classical manual anthropometric assessments (CA). However, many more measurements were obtained in the same amount of time than with CA.

Aldridge *et al.* (2005) reports image acquisition times of 2ms using a 3D imaging system but landmark location and measurement was done manually. Whilst image acquisition is generally very fast, no head anthropometric photogrammetric systems have been identified in this literature review that report accurate, real-time and automatic landmark localization. The review of landmark localization in section 2.2 identified several methods used to localize landmarks using various methods based on AAMs, and more recently, deep learning based approaches. Clearly there is an overlap here and a potential for cross fertilization across fields, however clinical acceptance of these innovations is necessary before these ideas can gain acceptance.

2.9.2 Landmarks used in craniofacial anthropometry

Whilst Farkas described 47 craniofacial landmarks (Farkas, 1994), modern 3D stereo photogrammetry methods typically use a smaller subset, for example FaceBase (Weinberg *et al.*, 2016) uses a subset of 24 of Farkas' original landmarks. The criteria for the selection of these is informed by the nature of 3D methods and their limitations. For example, as noted earlier, in order to reliably locate some traditional direct anthropometric landmarks, the practitioner needs to palpate soft tissue which is not possible using 3D photogrammetry. Previously sections reviewed datasets used for face detection and noted that up to 68 landmarks were used yet only a few of these were established craniofacial anthropometric landmarks.

Next, the established landmarks used in head and face anthropometry are identified, landmarks useful for this study are selected and include two additional landmarks that are not normally regarded as craniofacial landmarks but will be of potential use in future work.

The landmarks selected here are used for three purposes. Their first use is for estimating the relationship between landmarks in terms of their relative distances. Mean values between identified landmark pairs and their variance may be useful in providing constraints and have potential in guiding the design of the identification and selection algorithms used in this thesis. The variance and average distance between recognised landmarks can be obtained from tables of craniofacial norms published by Farkas and others identified earlier in this section.

Their second purpose is to act as reliable reference points to measure face posture defined by angle. This will allow, for example, the measurement of FHP.

Their third use is to act as delineation points when measuring regions of interest associated with head and face profile estimation, for example, the upper lip, nose, chin and so on. Such points of delineation are used later in this thesis to regress landmark positions from segmented head profile contours.

Table 2-1, overleaf, lists common established craniofacial landmarks used in the literature (Kolar and Salter, 1997; Farkas, 1994).

It lists candidate landmarks considered for use in this study and these are highlighted and italicised for this thesis. Those landmarks underlined were ultimately used in chapters 5 and 6 to segment face profiles.

Region	Name	Abr.	Definition
Head	***Glabella***	G	The most prominent midline between eyebrows.
Nose, Columella	Nasion	N	The midpoint on the soft tissue contour of the base of the nasal root at the level of the frontonasal suture.
	Sellion	Se	The most posterior point of the frontonasal soft tissue contour in the midline of the base of the nasal root.
	Pronasale	Prn	The most anterior midpoint of the nasal tip.
	Subnasale	Sn	The midpoint on the nasolabial soft tissue contour between the columella crest and the upper lip.
	Alare	Al	The most lateral point on each alar contour.
	Columella apex	c′	The most anterior, or the highest point on the columella crest at the apex of the nostril
Eye	***Exocanthion***	Ex	The soft tissue point located at the outer commissure of each eye fissure
	Endocanthion	En	The soft tissue point located at the inner commissure of each eye fissure
Philtrum, Lips and mouth	Crista philtra	Cph	The point at each elevated margin of the philtrum just above the vermilion line
	Labiale superius	Ls	The midpoint of the vermilion line of the upper lip
	Cheilion	Ch	The point located at each labial commissure
	Stomion	Sto	The midpoint of the labial fissure when the lips are closed naturally
	Labiale inferius	Li	The midpoint of the lower vermilion line
	Sublabiale	Sl	The midpoint of the Labiomental sulcus
Chin	Pogonion	Pg	The most anterior midpoint of the chin
	Gnathion	Gn	The lowest median landmark on the lower border of the mandible
Ears	Tragion	Tr	The notch at the upper margin of the ***tragus***

Table 2-1: Anthropometric landmarks and regions of the head.

2.9.3 Neck and upper body landmarks

Two axes of rotation related to head posture have been identified in the anthropometric literature. The first is an axis of rotation about the tragus in the sagittal plane and a second axis is located at the point of the C7 vertebra, again, in the sagittal plane. Authors typically construct a line joining the tragus and exocanthion and measure the angle between this construction and the horizontal or the Frankfort line. This angle is referred to as the gaze angle. The second angle is measured by constructing a second line between the tragus and the C7 vertebra and this angle is measured between this line and Frankfort line. This angle is known as the cranio-vertebral angle (CVA). Head posture can then be defined by these two angles (Youssef, 2016).

FHP describes the poor head posture resulting from the hyperextension of the upper cervical vertebrae and forward translation of the cervical vertebrae is significantly correlated with neck pain measures in adults and older adults (Fawzy Mahmoud *et al.*, 2019; Silva, Punt and Johnson, 2010). It is often measured using the CVA, although several authors use the two angles described above to measure the extent of forward head posture during examination as a greater gaze angle indicates a more extended position of upper cervical spine (Youssef, 2016).

Accurately identifying head posture by means of regressing landmarks on the face profile goes some way to measuring these angles, or an equivalent, since the landmarks provide a reliable reference for construction lines. For example, the subnasale can be used in place of the exocanthion in the above description. In order to achieve this, not only does the subnasale landmark need to be located but, in addition, an automatic method of regressing the position of the tragus is required as is the regression of the C7 vertebra position. An alternative to the C7 vertebra is presented here. The suprasternal notch (jugular notch), measured at the point of normal exhalation could be used if the C7 vertebra is not visible.

Relevant anthropometric head profile landmarks identified in this study could therefore be used as a starting point to develop methods to automatically measure forward head posture in real time.

2.10 Summary of gaps in knowledge and contributions

The first two chapters of this thesis have identified several opportunities to advance both knowledge and understanding within this field of research. These are enumerated in Table 2-2.

	Gaps in Knowledge
1	No one has evaluated the effectiveness of curvature and related features that could be used to efficiently identify regions of interest in a uniformly sampled sequential dataset.
2	No-one has investigated the capability of head profile *contours*, derived from fast depth cameras, to localize anthropometric landmarks.
3	A substantial body of work exists that has labelled 2D RGB images of front facing faces, but not with depth information, and few include side profiles. Nor does such work exclusively use anthropometric landmarks.
4	3D anthropometric datasets exist but these use slow, interactive capture methods, and are subject to editing/post-processing.
5	Datasets exist of raw and unprocessed profile head images taken using fast depth cameras, but none have been anthropometrically labelled.
6	There exist no reported publicly available datasets of raw, unprocessed head profile *contours*.
7	Whilst landmark localisation of head profiles from 2D images has been attempted, no evidence was available from the literature of the use of contours, in their raw state, extracted using depth information.
8	Fast depth cameras (30fps) are now commonplace so contours could be extracted in real-time, prior to landmark localisation. This approach has not yet been attempted.
9	A comparison of the effectiveness and run-time efficiency of end-to-end ML DNNs with that of DNNs that use engineered features has not been attempted within the context of head profile contour segmentation and regression.

Table 2-2: Summary of gaps in knowledge.

The gaps in existing knowledge identified here have resulted in the following contributions to knowledge detailed in Table 2-3. These contributions include an evaluation of curvature and curve derivatives as features for uniformly sampled time series datasets as well as more general contour curves sampled from images. They include the generation of a new head profile contour dataset labelled with anthropometric landmarks, as well as investigations of the effectiveness of DNNs in the segmentation of head profile contour and the regression of profile landmarks. Chapters 4,5 and 6 of this thesis detail the investigations carried out that resulted in these contributions.

	Contributions
1	This study provides an evaluation of the effectiveness of curvature and related features used to *efficiently* identify regions of interest in a *uniformly* sampled sequential dataset. This includes the generation of new algorithms, software and tools to evaluate the accuracy of a recurrent neural network (RNN) that uses the features engineered here. Additionally, an investigation of the run-time efficiency of the engineered features is provided (Table 2-2, gap 1).
2	An existing RGB-D dataset is extended, by extracting head profile contours so creating a *new* database of face profile contour curves. Additionally, a *new* set of manual annotations are generated identifying key anthropometric landmarks on both the RGB images and profile contour curves (Table 2-2, research gaps 3,5,6).
3	A novel approach is developed in this context to automatically improve the accuracy of the annotations based upon the curvature properties of selected anthropometric landmarks (Table 2-2, research gaps 3,5,6).
4	The findings of 1, above, are extended to work with the head profile contour dataset created in this study resulting in a new procedure that can accurately achieve fast face segmentation of head profile images. An evaluation of this procedure documents both the accuracy of the approach and its run-time efficiency when used with two RNNs (Table 2-2, research gaps 2,4,7,8).
5	The procedure used in 4 is further extended to develop a method to regress landmarks from the segmented profile contour and the accuracy and precision of this method is evaluated and documented (Table 2-2, research gaps 2,4,7,8).
6	Finally, the effectiveness of an end-to-end learning approach is investigated using a 1DTCNN to achieve the same goals of 4 and 5, above, and the findings of this investigation presented with recommendations (Table 2-2, research gap 9).

Table 2-3: Summary of Contributions cross referenced to gaps in knowledge (see Table 2-2).

2.11 Summary

The importance of landmark localisation was reviewed in the fields of 2D face recognition and landmarking, and in indirect anthropometry and photogrammetry. This identified candidate landmarks that could be applied to head profile contours and their properties. A discussion of their usefulness as fiducial markers in head profiles was also included.

An overview of methods used to localize landmarks in 2D images identified potentially useful approaches that could also be used for segmenting head profile contours. The use of deep neural networks used in 2D face landmarking has had great success recently and shows that the application of DNN concepts will be successful in this thesis. As head profile contours can be regarded as a sequential series then the application of deep learning methods such as RNNs and one dimensional temporal CNNs were reviewed and their effectiveness in this area discussed. Since such methods will need to be evaluated then a brief review of ML metrics was also outlined here.

Additionally, methods useful in extracting profile contours from RGBD images were detailed and the success of curvature and curve derivatives as features in profile recognition and landmark regression were reviewed. The features and the methods identified here were hypothesised to be useful in fast and accurate classification and regression and so efficient approaches to calculating these features were investigated in this review also.

Finally, gaps in knowledge within this field were enumerated and the arising contributions of this thesis were summarised.

The following chapter describes and justifies the research methodology used in the investigations carried out in chapters 4, 5 and 6 of this thesis.

3 Research Methodology

This research explores the suitability of curvature and its properties as features for training deep neural networks to estimate head profile posture. Previous chapters have explored related work and identified opportunities to advance existing knowledge and research arising from this. This chapter will describe the research methods used in the investigative studies that follow. The chapter identifies and justifies the research methods applicable to this research and follows on with a discussion of the chosen methodology, and details the common methods used in the investigative studies of chapters 4, 5 and 6.

3.1 What type of research?

The aim of the research is to "*explore extensively the suitability of curvature and its properties as features for fast regression and segmentation of parameterized plane curves, and in so doing, examine the effectiveness of these features in training deep neural networks to estimate head profile posture derived from 2.5D images."* This aim, derived from the research question, indicates the use of an exploratory investigation to prove or disprove the identified hypotheses.

This research is applied research since it aims to solve a problem, that is, define head posture position. Since the exploration also focuses on evaluating the effectiveness of machine learning algorithms and their time efficiency, then the research is also of a quantitative nature.

3.2 What Research Method?

Given that the research has been identified as exploratory, appropriate research method(s) need to be reviewed that will suit an exploratory study. As the main focus of this study is to explore the suitability of curvature and its properties as effective features for estimating head posture, then ways of measuring this effectiveness need to be identified. Also, any methods used must ensure relationships between selected variables are well defined, quantifiable and that the measurements are repeatable. Additionally, the methods used need to be appropriate for testing hypotheses in controlled conditions to deduce causal

inferences. The experimental research method meets these criteria (Maxion, 2009), and so will be used in this study.

The aim of this thesis also requires the development and evaluation of various DNNs. Typically, DNNs are considered part of the machine learning canon and development of these models follow an established machine learning process which is an empirical procedure and is experimental in nature.

3.3 Experimental Research

There are a range of experimental types used in experimental research. As this thesis aims to evaluate the effectiveness of various features used to train a number of deep neural networks, it is necessary to vary these features in a controlled and systematic way. This, then, is a controlled experiment and is conducted within a setting that is especially created for this investigation.

3.4 Experimental Methodology

In order to achieve the research aim identified in chapter 1, several objectives were identified and enumerated along with corresponding experimental investigations. These investigations are documented in chapters 4, 5 and 6. However, only the general experimental design details and common methods which apply to all these experiments will be discussed in this chapter. Details relevant only to a specific investigative study will be reported in the corresponding chapter.

The experiments and investigations documented in this thesis focus on the development and evaluation of various features as inputs to machine learning algorithms. Whilst the investigation uses an experimental approach, the procedure used naturally follows that of the machine learning process (Kumar and Sharma, 2017). This is described next and accommodates within it opportunities to apply experimental methods during the phases of feature selection, pre-processing, model optimization and model evaluation.

3.5 Machine Learning Process

As part of the investigative studies undertaken in this thesis, several deep learning models were developed. This involved creating datasets, pre-processing existing datasets, engineering features, designing and implementing DNN models, and was followed by their supervised training and evaluation.

Several process standards related to developing machine learning and data science systems have been documented in the literature (Wirth, 2000; Shearer, 2000; Studer *et al.*, 2021; Azevedo and Santos, 2008; Géron, 2019). Broadly they follow the same approach as that shown in Figure 3-1 below.

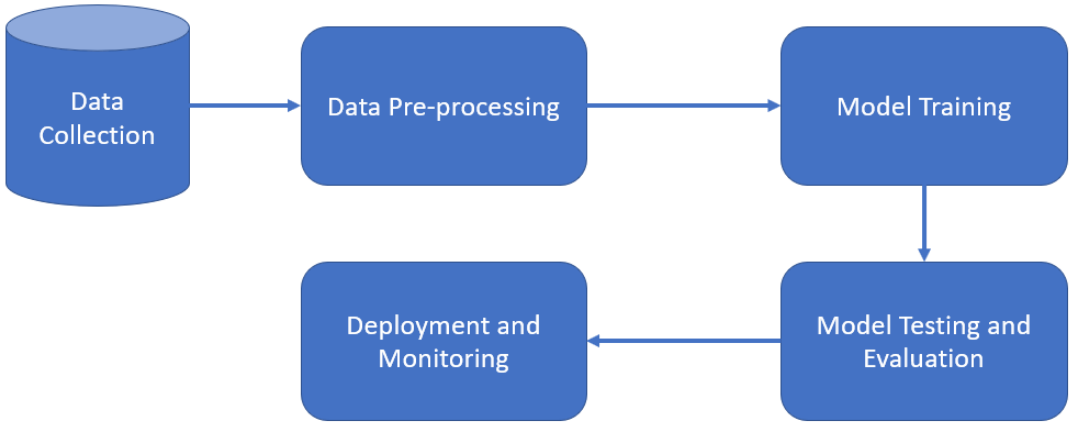


Figure 3-1: Stages involved in the ML process (iterative phases not shown).

Once the dataset has been collected and collated, several candidate features are first identified and transformed during the data pre-processing phase. This is referred to as feature engineering in this study. The experiments relevant here include the evaluation of the features and the transformations used.

In chapter 4 a study of feature transformations is undertaken, and their resulting run-time efficiency is analyzed. During model training further experiments are undertaken to establish the best features to use with the ML algorithms. The experiments of chapter 4 also have a confirmatory aspect since the deep learning model used and its architecture and training parameters were specified in previous work (Mathworks, 2020c). The new

features explored in this experiment can then be evaluated against the previous published results and the previous results can then be verified as part of this experiment.

Chapters 5 and 6 also undertake experiments as part of the ML process. Chapter 5 evaluates the model with newly engineered features (curvature and its derivatives) whilst chapter 6 develops a different DNN ML model to demonstrate the effectiveness of an end-to-end ML approach.

3.6 Apparatus

All experiments in this study were performed on a machine with the following specification and development software.

Processor	Intel core i7-7700 CPU
Memory	32GB RAM
GPU	Nvidia 1080Ti GPU
Operating System	Windows 10
Matlab version	2019b (Chapter 3 and 4); 2020b (Chapter 5).

Table 3-1: Computer hardware specification.

3.7 Evaluation Metrics

Evaluation of the experiments fell broadly into three camps, classifier evaluation, regressor evaluation and run-time efficiency evaluation. The classifiers of chapters 4, 5 and 6 are evaluated using the metrics discussed here.

3.7.1 Classifier Evaluation

Several metrics are used for evaluating classifiers, and whilst a single numerical value that describes the overall accuracy of a classifier is desirable, a more thorough approach is necessary.

Typically recall and precision are reported together with their harmonic mean (F1 score). Overall accuracy and the macro F1 score are also reported. The experimental investigations used in this thesis report per-class F1 scores, recall and precision together with overall accuracy and macro F1 score. These metrics were calculated from multiclass confusion matrices. These values and their significance are outlined in the previous chapter with further details provided in Appendix D.

These metrics were chosen since they are well understood by the ML community and results are normally reported using these metrics (Handelman *et al.*, 2018; Kamiri and Mariga, 2021).

3.7.2 Regressor Evaluation

Where experimental investigations evaluate the accuracy of facial landmark localization, two measures are used, the precision and the accuracy (International Organization for Standardization, 1994). Here, precision is described using the sample standard deviation of the test results from the ground-truth facial landmark.

Accuracy is measured using the mean absolute error (MAE), and was used instead of the mean squared error (MSE) since MSE exaggerates the importance of outliers thus avoiding excessive skewing of results. The mean error (ME) is also reported as it gives additional information (positive or negative values) indicating whether a landmark's estimated position is skewed below or above its true location. These metrics are also well understood by the ML community and regularly used in the literature (International Organization for Standardization, 1994; Handelman *et al.*, 2018; Kamiri and Mariga, 2021).

The procedure followed regarding the inclusion of outliers here is that suggested by typical ML dataset pre-processing methodologies (Géron, 2019). That is, outliers are examined individually, and a decision is made as to whether they are included in the dataset. For example, an image showing no profile at all would not represent a valid test sample and so would not be included in the evaluation.

3.8 Run-time Efficiency Evaluation

A key objective of this study is to investigate candidate features suitable for fast classification and segmentation of planar curves. These results inform the feature engineering processes and algorithms used to train the LSTM time series segmenter of chapter 4, the LSTM profile segmenter of Chapter 5 and the 1DTCNN of chapter 6. Additionally, measuring the processing-time of various algorithms used in the segmentation process is important for this study and, more generally, machine learning researchers are beginning to focus more on run-time evaluations of their algorithms as well as classification or regression performance (Handelman *et al.*, 2018; Kamiri and Mariga, 2021).

3.8.1 Timing procedure

The algorithms were implemented in MATLAB version 2019b and MATLAB version 2020b. MATLAB's timeit() function was used to calculate the run-time of the MATLAB scripts. The algorithms' runtimes are tabulated, and relative comparisons are made. The goal is to compare the speed of the algorithms against each other and not to measure against an absolute reference. Consequently, they were not cross-compiled to C++ nor other optimizations applied beyond MATLAB's default settings.

A comparison of the timeit() function's results with that of MATLAB's built in code profiler was also undertaken. Both methods produced comparable results. The timeit() function was chosen and the procedure described here was used in the experiments detailed chapters 4, 5 and 6.

Current processors are typically multicore which can affect accurate timing of algorithms as operations may be shared across cores. Most modern applications such as MATLAB take advantage of multi-threading techniques which are often out of the control of the script writer. Additionally, modern processors make use of single instruction, multiple data (SIMD) architectures. Here, operations such as multiplication and addition are applied in parallel to a group of data. This can drastically improve execution time. Convolutions and vector additions benefit from this architecture, for example, a convolution involving 10 multiplications will take the same time as one involving 2 multiplications. This further complicates the problem of accurate timing of algorithms.

Consequently, a pragmatic approach was taken in choosing timing methods as described next.

CPU clock cycle timing was considered and would have been preferred if a single core machine without an SIMD architecture was being used. As this could not be guaranteed in this study and would not be representative of the hardware used in a real situation, a variation on a wall clock approach was considered using MATLAB's timeit() function.

The timeit() function measures wall clock time but performs multiple runs of the specified function and returns the median of results. This is done to reduce the effect of errors introduced by multi-tasking operating systems. To summarize, the timeit() method was chosen for the following reasons:

1. It returns the median of multiple timing measurements of a function,
2. times recorded are representative of typical use,
3. a brief review of researcher opinions and papers that report execution times using MATLAB, use wall-time methods such as timeit() and tic/toc, (Mathworks, 2020b).
4. the MATLAB profiler also returned results comparable with the timeit() function.

The procedure used is detailed next:

1. Each feature pre-processing algorithm was encapsulated in a MATLAB function with all requisite one-off initialization of variables completed outside the function and before starting the timing tests.
2. Each algorithm was inserted inside the test function, enclosed in a for-loop and executed 1000 times. The average time taken was calculated from the overall time taken to complete 1000 runs.
3. Within each test function, each algorithm was applied to the same data-subsets.
4. All variables were cleared using the clear all command prior to timing.

Feature generation algorithms require the initialization of several parameters. Where relevant, details of the initialization of parameters specific to run-time experiments are discussed in the relevant chapters.

Regarding the timing of neural network inference times, the same procedure described above was used. Where training times are reported, these were provided by MATLAB's neural network training functions. MATLAB's GPU parallel pool was turned off when measuring training and inference times.

3.9 Common Experimental Methodologies

Several investigative studies are detailed in this thesis, and the experiments undertaken are, in general, specific to each investigation and documented in subsequent chapters. However, there are some aspects of the experimental design and methods which are common, for example the application of the machine learning procedures when training and evaluating the DNN models of chapters 4, 5 and 6. This section describes these common procedures. Details relevant to specific investigative studies will be omitted here and reported in the corresponding chapter.

3.9.1 Training LSTM and 1DTCNN models.

Chapter 4 investigates the application of various features used to train an LSTM segmenter using an ECG dataset. As this dataset is not used elsewhere in this thesis then it is not discussed in this section. However, chapters 5 and 6 investigate various features used to train DNNs capable of segmenting and regressing profile head contours. Hence, the same head profile contour dataset is used in both chapters. Accordingly, discussions of the dataset design that are common to both investigations are provided here.

Similarly, some common features are engineered from the head profile contour dataset during the pre-processing period and applied to both the LSTM RNN and the 1DTCNN of chapters 5 and 6. Consequently, the following sections summarize: the dataset used and justification of the design, the features engineered from the dataset and applied to the model, and finally, the common procedures used to train and evaluate the networks.

3.9.1.1 The head profile contour dataset

The creation of the head profile contour dataset is described in detail in chapter 5. It is also used for the investigations of chapter 6 and so is outlined here. Figure 3-2 below

illustrates an example instance of a head profile contour that has been labelled with landmarks that delineate the coloured regions.

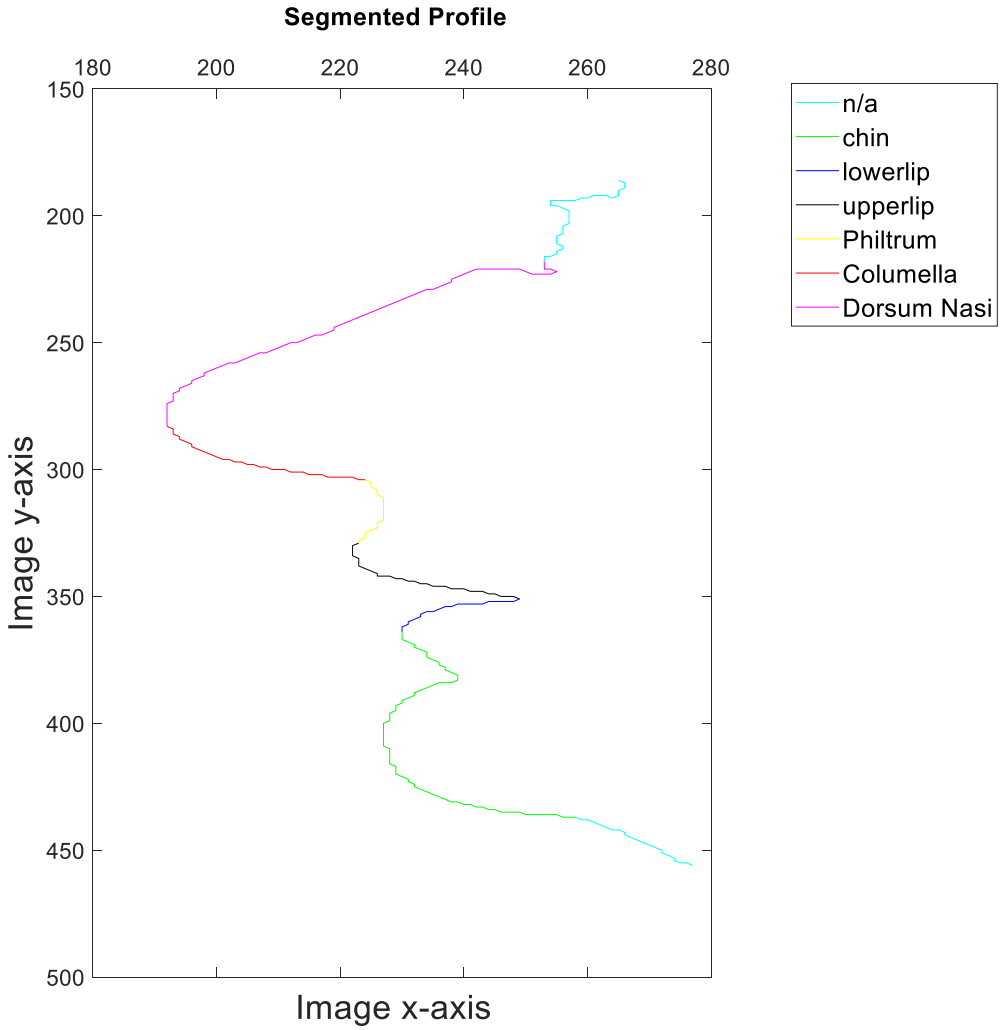


Figure 3-2: Example of a single segmented head profile contour.

The number of contours in the dataset is 648. Segmented profiles consist of seven regions as illustrated in Figure 3-2 and Table 3-2.

Class imbalance is an important consideration in pre-processing datasets. The dataset is mildly imbalanced though not extremely so, with the columella of the nose having the largest support. Class size ratios for philtrum:columella is 2:3, upper lip:columella 2:3 and lower lip:columella 1:2.

Region (label used)	Start Point	End Point
n/a (Not defined)	7 samples before the Gnathion	Gnathion (gn)
Lower lip	Labiale inferius (li)	Stomion (sto)
Upper lip	Stomion (sto)	Labiale superius (ls)
Philtrum	Labiale superius (ls)	Subnasale (sn)
Columella	Subnasale (sn)	Pronasale (prn)
Dorsum nasi	Pronasale (prn)	Sellion (se)
n/a (Not defined)	Sellion (se)	7 samples after the sellion

Table 3-2: Segmented contour profile.

Since the aim of the segmenter is to accurately identify a small number of landmarks that remain invariant under transformation and facial expression, then only a subset of the profile is required. The dataset used was trimmed accordingly, resulting in the number of n/a labels being reduced to 14, 7 before the gnathion and 7 after the sellion. This means the longer "n/a" labelled parts of the profile can be trimmed, thus improving the balance of the data categories.

The chin region is not used in this study as it contains no useful anthropometric landmarks and in comparison to other regions it also adds to data imbalance. Consequently, it was removed from the dataset and the segmented contour profile dataset was adjusted accordingly, thus further reducing dataset imbalance. The resulting dataset consists of contours of varying length, from approximately 14 sample points up to 155. This dataset is used to train the networks of chapters 5 and 6. Further information relating to the generation of the dataset and choices made are specific to chapter 5 and so are not re-produced here.

3.9.1.2 Model training

The ML models used to segment both the ECG dataset and the head profile contour dataset is a supervised classifier trained on labelled datasets. For all the investigations of chapters 4, 5 and 6, training follows the established machine learning approach.

Features are initially engineered from the relevant dataset and then the dataset is spilt into a train:test ratio of 70:30. A validation subset is useful when a parametric optimization of the model is crucial and can also be used during training to prevent over-fitting by using early stopping as described in chapter 2 and appendix D. Since this study was focused on the engineering of features suitable for a DNN segmenter, use of a validation subset was not essential. The DNNs used did not need extensive optimizations since this was not the focus of the study. The investigation of chapter 4 used a predefined experiment which did not use a validation subset and so to ensure repeatability of the existing experiment this constraint was observed also. Additionally, the head profile contour dataset used in chapters 5 and 6 was small and removing a subset for validation would further reduce the dataset size. Finally, overtraining was avoided by monitoring training and stopping once the accuracy of the model plateaued. In summary, all experiments did not use a validation data subset.

The features used to train and test the DNNs are specific to each investigation, however the overall procedure is the same. For each engineered feature set input to a model the following procedure was executed:

- Repeated training runs were undertaken to identify the number of epochs required for each model to plateau.
- Five runs of each training procedure were carried out and the model with the median accuracy was used.

3.9.2 Testing and Evaluation of LSTM and 1DTCNN models

The trained models were tested on their related test dataset. The DNN models investigated in chapters 4, 5 and 6 outputted a sequence of classifications corresponding to each point on a contour. All models used a softmax output layer corresponding to the number of classes for each investigation. The softmax function and its use as an activation function in neural networks is well documented in the literature (Russell and Norvig, 2020) and an overview is presented in Appendix C.

In Chapters 5 and 6 the softmax outputs comprised of 5 classes:

- n/a (not defined),
- lower lip,
- upper lip,
- philtrum,
- columella.

For Chapter 4 the softmax layer had 4 output classes comprised of:

- n/a (not defined),
- P,
- QRS,
- T.

The softmax layer produces a probabilistic result for each class so an argmax operation was applied to identify the class with the highest probability.

Consequently, a sequence of length, *n* applied to the trained models would result in an output sequence of the same length, *n* but containing a sequence of classifications, one for each point of the presented sequence.

3.9.2.1 Classifier evaluation

For a given engineered feature or feature combination, each model's test outputs were stored in MATLAB arrays and evaluated using the metrics and methods described in section 3.7.1 as follows:

- For all test sequences, the labelled ground truth values and the model's predicted class values were used to calculate a multi-class confusion matrix.
- From this confusion matrix, the following metrics were calculated for each class:-
  - The support,
  - The recall,
  - The precision,
  - The F1 score.
- Using these results, the overall accuracy and macro-F1 scores were calculated.

Chapters 4, 5, and 6 present and discuss these results, and where relevant, detail any investigation specific modifications to this method.

3.9.2.2 Regressor Evaluation

This section details the common methods used to evaluate landmark location regression accuracy in both chapter 5 and 6. Chapter 5 provides full details of the approach used to initially label ground truth landmark positions and documents the method used to predict landmark positions from segmented head profile contours.

A predicted landmark position occurs at a transition between regions of a contour. Each point on a contour represents a pixel position and the contour is a list of adjacent pixels. The distance between a predicted and a ground-truth contour position is an integer value, measured in pixels and corresponds to a simple count along the contour line. For example, if the difference between a ground-truth landmark and the model's predicted landmark position is one pixel, then they are adjacent.

The metrics discussed in section 3.7.2 are used to evaluate landmark position. These are MAE, ME and precision. These results are calculated using the full test dataset. The precision is calculated using the sample standard deviation. Results are tabulated and histograms generated to show the distribution of test dataset values. This visualization was included to identify any bi-modal or related artifacts existing within the test results.

3.10 Summary

This chapter presented and considered the experimental research methods adopted in the investigations of the following chapters. It provided details of the common methods used to train and evaluate ML models. The methods identified and detailed here are used to measure the effectiveness and efficiency of both classifiers and regressors.

In the following chapters, several experiments are undertaken that use the ideas considered here to evaluate the effectiveness and efficiency of curvature and curve derivatives as features for a RNN capable of segmenting a uniformly sampled time series dataset. The best of these features are then selected for further investigations in subsequent chapters.

4 Segmenting uniformly sampled datasets with RNNs

This chapter considers approaches used to efficiently calculate 1st and 2nd order derivatives and the plane curvature of a uniform time-series dataset (Mathworks, 2020c). Algorithms are developed that calculate these features efficiently and their effectiveness in classifying a publicly available time series dataset is evaluated and compared with other commonly used alternative features derived from that dataset. All features are used as inputs to a LSTM RNN which is used as the classifier in these experiments.

4.1 Comparison of Gaussian derivatives and central difference methods

In this section the central difference method is compared with the *DoG* methods for calculating signal derivatives on unfiltered univariate times series data and the advantages and disadvantages of both methods are discussed.

The central difference method of calculating derivatives and defined in equation (2-10) is considered first.

This can be implemented by convolving a 1D sample with the kernel [-½ 0 ½]. This approach is fast and requires just 2 multiplications and one addition per sample. However, previously it was noted that sudden step changes as might occur between sampled points, on an image or univariate waveform, can lead to errors in calculating the derivative at finer scales and so low pass filtering of a sampled curve is advisable prior to calculating the derivative (Farid and Simoncelli, 2004; Bouma *et al.*, 2007). This additional filtering step reduces the speed of the process; however, the central difference method is still commonly used to calculate derivatives despite the small support of its kernel and its subsequent sensitivity to noise, even after filtering.

Since an aim of this study is to assess the suitability of derivatives and curvature as features for regression, classification and segmentation, it is still worth considering the central difference method since the limitations of the method may be inconsequential when used as a feature for classification. It may, or may not, be a suitable candidate feature for a classifier when used alone on the raw data, or with pre-filtered data, or when used to calculate curvature, so a further analysis is appropriate.

To determine this initially, a visual comparison is made between the central difference method and the Gaussian derivatives discussed in section 2.6.2. First, both methods are used to calculate curvature then a visual inspection of the resulting curvature graphs is undertaken to confirm whether or not the central difference method produces results comparable to the *DoG.*

To assess the quality of these methods for computing derivatives a sample dataset is required. Here the chosen dataset is taken from the publicly available Research Resource for Complex Physiologic Signals' QT Database ECG dataset (Goldberger *et al.*, 2000) which is used throughout this chapter to evaluate the effectiveness of features used to segment ECG signals and is described in detail in section 4.3.1.

To assess the central difference method of calculating the curvature we load an arbitrary ECG signal comprising of 250 000 samples and first filter this by convolving it with a Gaussian kernel with a standard deviation, $\sigma = 3$. The kernel is a one dimensional array of $6\sigma + 1 = 19$ elements, sampled from the Gaussian function.

The curvature is next calculated using equation (2-6), repeated here for convenience:

$$\kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{\frac{3}{2}}}$$

This equation requires the calculation of the first and second derivatives of the *x* co-ordinate with respect to arc length and the first and second derivatives of the *y* coordinate also with respect to arc length. Since this is a univariate curve, the *x* values increment uniformly by one for each sample. Hence $x' = 1$ and $x'' = 0$ and this equation simplifies to:

$$\kappa = \frac{y''}{(1 + y'^2)^{\frac{3}{2}}}$$

Next, a random section of the smoothed curve is selected then y' and y'' are calculated by using the central difference method and the resulting curvature is plotted and then visually inspected, see Figure 4-1 (lower).

For comparison, the curvature is then calculated using the *DoG* (upper), again, with σ=3. The second derivative is calculated here by applying the *DoG* twice.

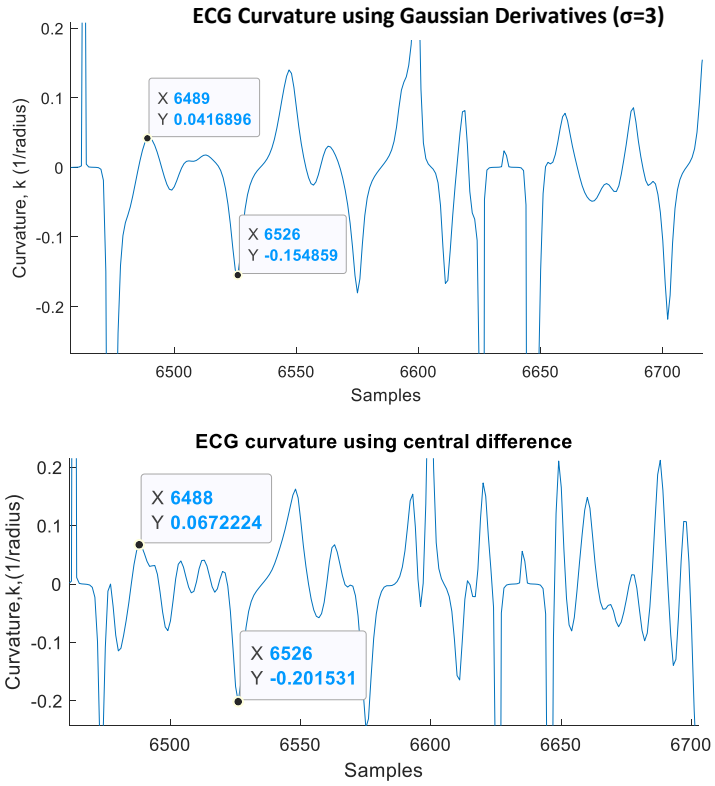


Figure 4-1: Comparison of Curvature Calculations. Upper using Gaussian derivative, lower using central difference method on a Gaussian smoothed (σ=3) signal.

As can be seen, the lower plot is more sensitive to changes and exaggerates curvature, despite the signal being filtered with the Gaussian smoothing kernel (σ=3). However, it does reproduce the curvature characteristics obtained by using Gaussian derivatives and in particular, minima and maxima of curvature occur at the same arc-length sample locations. The randomly selected points shown demonstrate this.

Figure 4-2 visualizes the distribution and range of the curvature calculations for each method using a sample size of 5000. Note the range of the central difference method is approximately twice that of the Gaussian method and exaggerates changes in curvature.

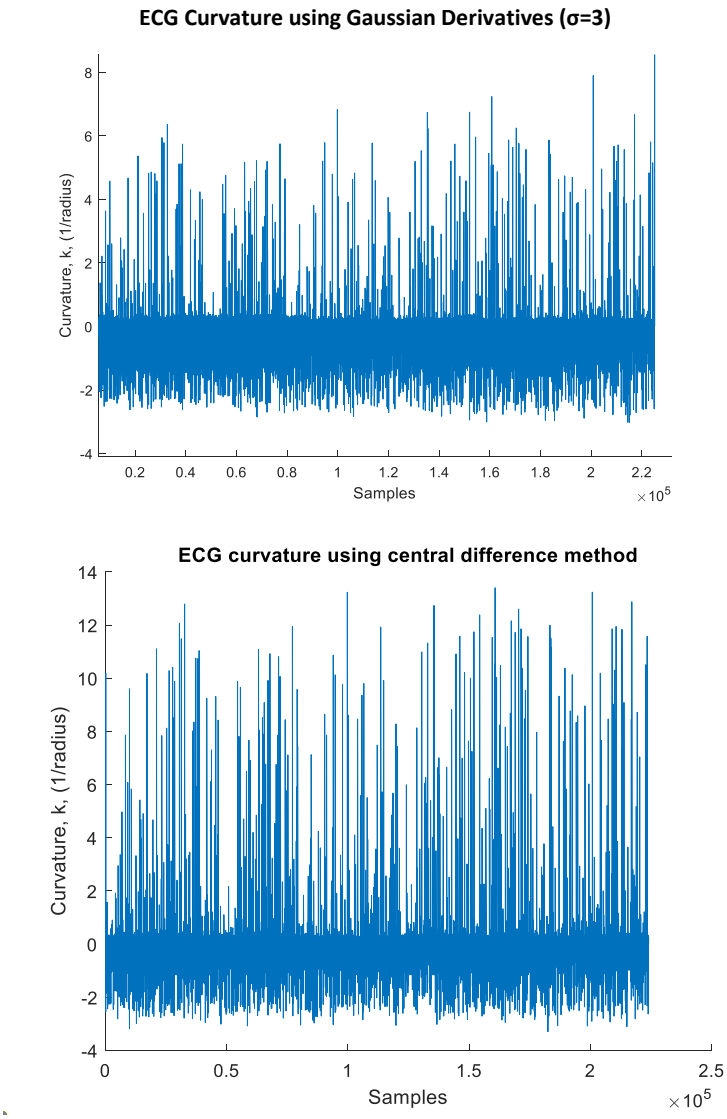


Figure 4-2: Comparison of Curvature Calculations across 250 000 samples. Upper using Gaussian Derivative, lower using central difference method.

The conclusion is, the central difference method, despite being applied to a filtered signal is more susceptible to local changes in curvature due to its small support, compared with a Gaussian derivative. However, it is still capable of locating important curvature features. As a result, the central difference method is not discarded here but is investigated further and its usefulness as a feature for segmenting an ECG signal is evaluated in later sections of this chapter. Its major disadvantage appears to be the need to pre-filter the signal before differentiation. The question of whether pre-filtering is required for the purposes of segmentation is answered in section 4.3.

The next section of this chapter focuses on the efficiencies that can be achieved using the properties of the Gaussian function, in particular its ability to achieve both filtering and derivative calculations in one pass. A procedure to efficiently calculate curvature-based features suitable for training a classifier to segment a time series dataset is then presented.

4.2 Efficient filtering and derivative calculations using Gaussian kernels

Having compared the central difference method with the *DoG* to calculate derivatives in section 4.1, this section now focuses on applying Gaussian smoothing to a one-dimensional time series dataset and the calculation of first and second order derivatives. In particular, it focuses on how these operations can be combined into a single Gaussian derivative kernel. Chapter 2 presented the necessary theory required to achieve this and whilst the focus is on a uniformly sampled time series in this chapter, the procedure is equally valid when the dataset represents any curve on a two-dimensional plane.

Recall from section 2.6.2 the Gaussian function, $G^{\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}}$,

its first order derivative, $DoG = \frac{x}{-\sigma^2\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}}$,

and the second order derivative, $LoG = \frac{(x^2 - \sigma^2)}{\sigma^4\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}}$.

Gaussian smoothing and filtering were achieved in one efficient operation by convolving the *DoG* with the unfiltered curve, *C*. This gives the first derivative of the smoothed (low pass filtered) curve, C'. To obtain the second derivative either convolve the Gaussian second derivative kernel, that is, the *LoG* with *C*, or convolve C' with the *DoG* kernel once more.

To achieve this, first create a Gaussian kernel of the required support by sampling the normalized Gaussian function as previously described. The kernel size is $6\sigma + 1$ and the Gaussian is sampled from x= -3σ to $+3\sigma$ on integer steps. This range is sufficient to ensure the sampled Gaussian is close to zero at $\pm 3\sigma$. The samples are scaled to ensure there is no magnification of the signal, that is the kernel elements sum to one. Note the *DoG* is the Gaussian function scaled by $-x\sigma^{-2}$. As the *DoG* goes to zero a little more slowly than the Gaussian, then kernel size may benefit by extending the support to ± the ceiling of 3.5σ + 1 or higher, depending upon the series length and sample rate.

Determining the appropriate support is explored next, when curvature calculation methods are investigated.

4.2.1 Comparison of numerical and analytical curvature calculations.

Although the curvature equation can be simplified to a univariate function as described in section 4.1 when samples are taken periodically, the curvature is determined by using the *full* curvature equation given by equation (2-6). In this section the first and second derivatives (with respect to arc length) of each data point will be calculated prior to completing the curvature calculation of equation (2-6). As a sanity check, the curvature of a sine wave was determined analytically and then compared with curvature determined using derivatives calculated using *DoG* and *LoG* methods.

The equation for the curvature of a sine wave is derived from the curvature equation (2-6). After differentiating the relevant terms, it is,

$$k = \frac{\sin(x)}{\sqrt{(1+cos^2(x))^3}} \tag{4-1}$$

A plot of the curvature and the sinewave itself is shown in Figure 4-3 overleaf. The left figure shows curvature calculated analytically and the right using *DoG* and *LoG*.

These plots show that the *DoG* method re-produces the analytical plots faithfully. MAE of the normalized range of the analytic and *DoG* calculations is 0.0077 and the Pearson correlation coefficient is 0.9997 showing a very close agreement and provides some confidence that this numerical method gives accurate results.

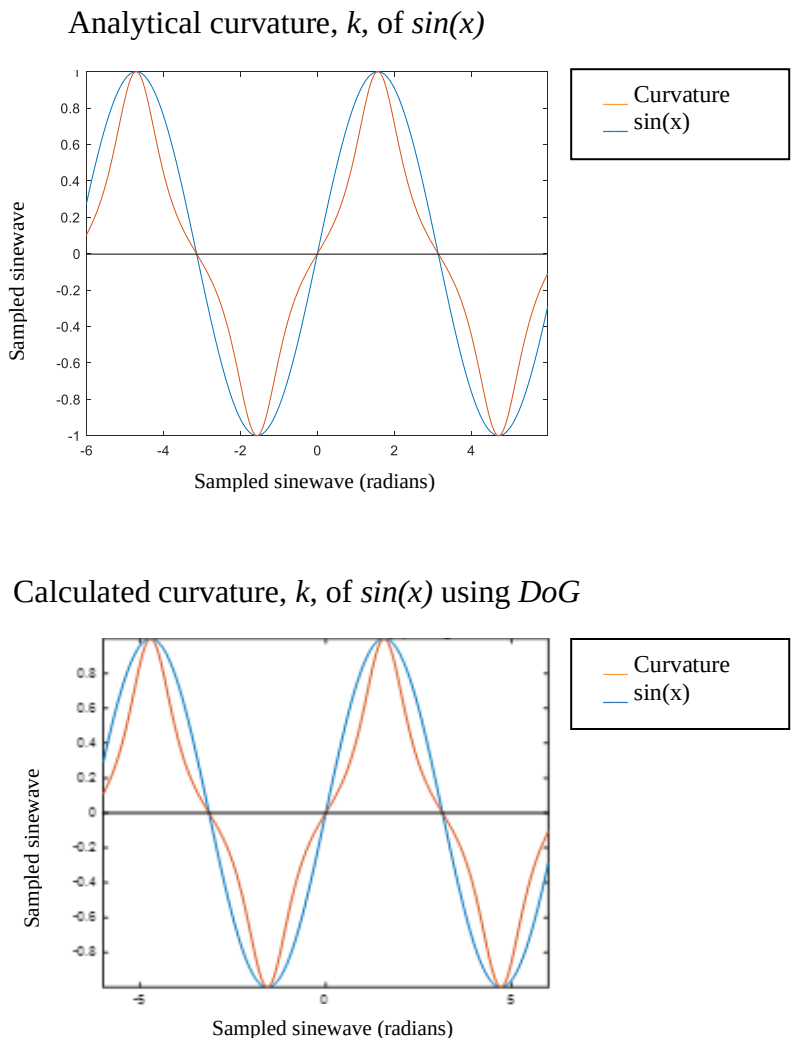
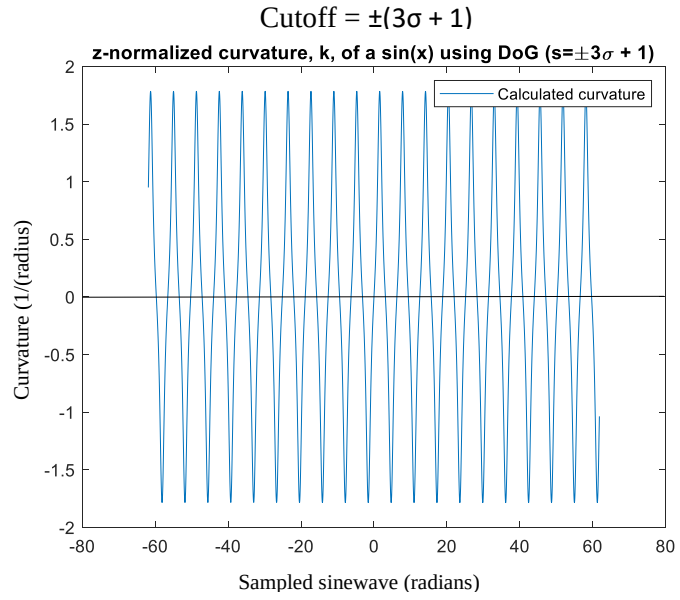


Figure 4-3: Curvature of sine wave: Top- Analytically Calculated, Bottom- DoG used.

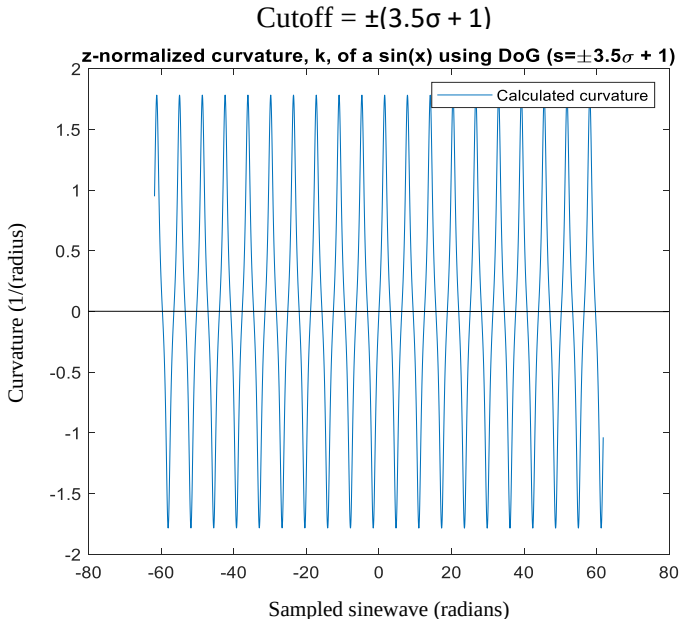
4.2.2 Effect of Gaussian derivative kernel size on long data samples

Directly filtering and calculating the 2nd derivative using the *LoG* is sensitive to the size of the kernel and the support needs to be sufficient to ensure consistent results when convolving with a large number of samples. This assumption is examined and illustrated next. A sampled sinewave of length 5000 is used as this is the size of the feature vectors used with the LSTM RNN investigated in the next sections. *DoG* and *LoG* methods are used to calculate the curvature with a kernel range from – cutoff up to +cutoff where cutoff is defined as the ceiling of ($s\sigma$ + 1). In this experiment the variable, s is set variously to 3, 3.5, 4 and finally 5; σ is set to a constant value of 2. The results are shown in Figure 4-4, Figure 4-5 and Figure 4-6. It was observed that the lower support values of s=3 and 3.5, used with the *LoG* method introduced an error in the calculated curvature

proportional to the magnitude of the *x* co-ordinate – see Figure 4-6. At $\sigma = 2$, an *s* value of 4 is sufficient to remove the error. This gives a kernel size of *2cutoff +1 = 2*9+1= 19*.



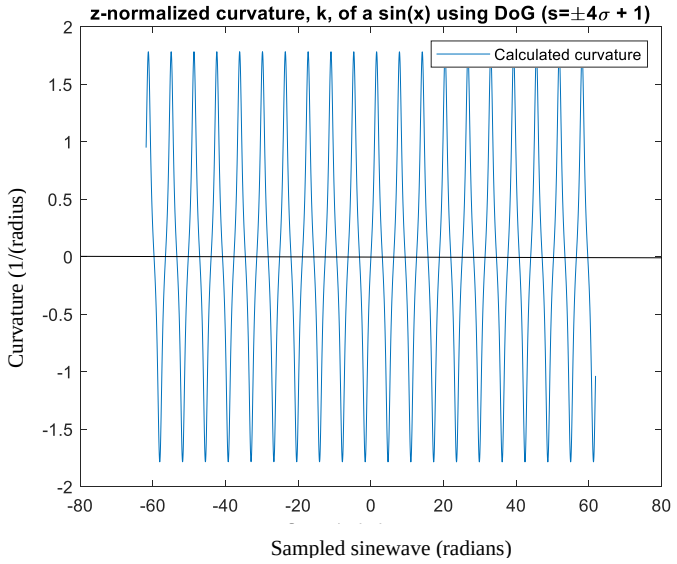
(a)



(b)

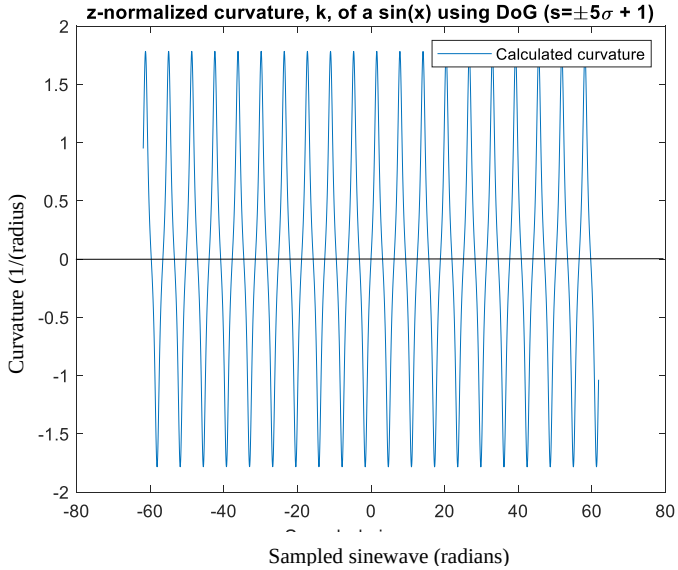
Figure 4-4: Curvature calculated using Derivative of Gaussian, $\sigma = 2$. (a) s = $\pm 3\sigma$ + 1, b) $\pm 3.5\sigma$ + 1.

Cutoff = ±(4σ + 1)



(a)

Cutoff = ±(5σ + 1)



(b)

Figure 4-5: Curvature calculated using Derivative of Gaussian, $\sigma = 2$. (a) $s = \pm 4\sigma + 1$, (b) $s = \pm 5\sigma + 1$.

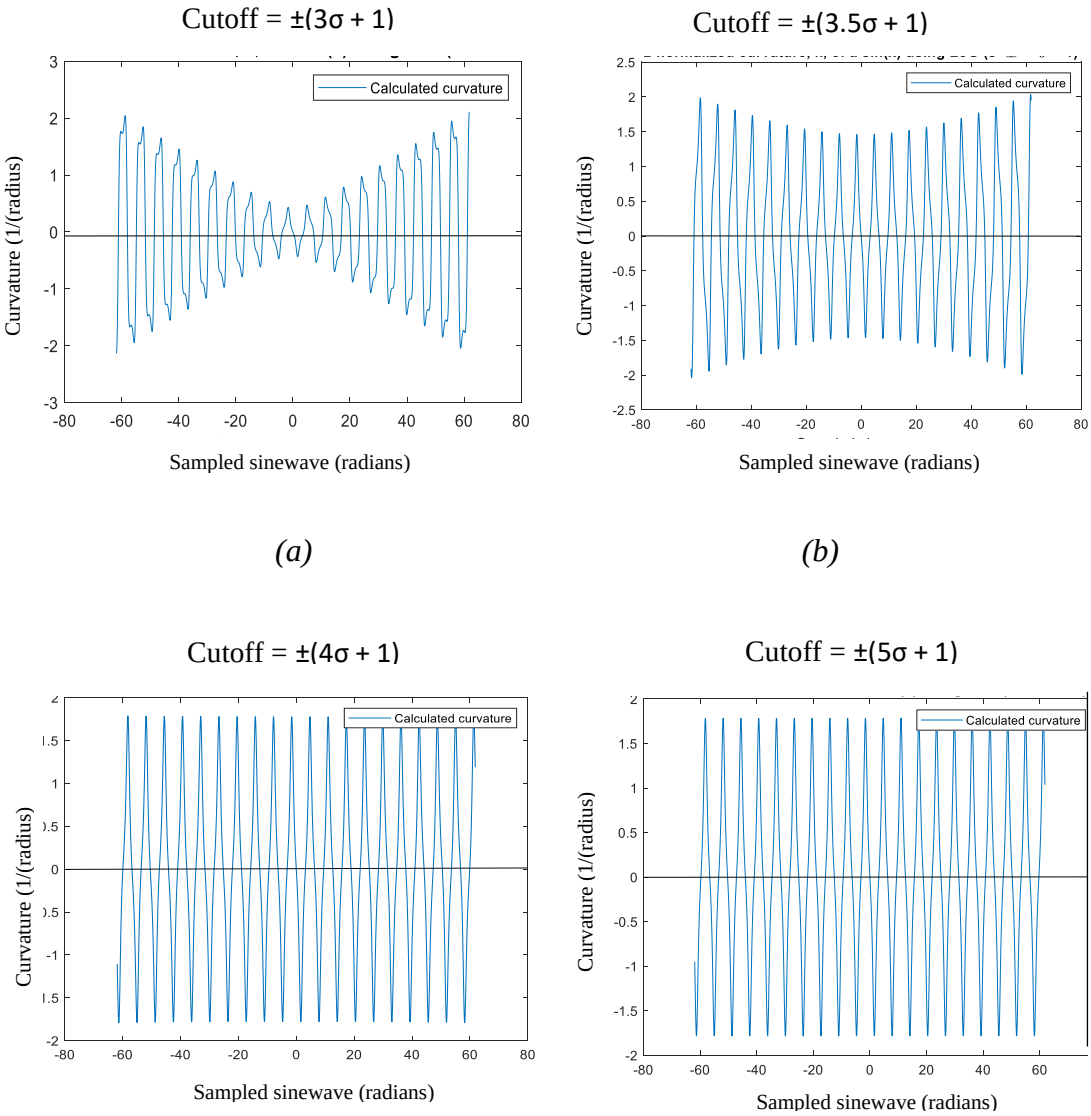


Figure 4-6: Curvature of sin(x) calculated using LoG: $\sigma = 2$: a) $s = \pm 3\sigma + 1$; b) $s = \pm 3.5\sigma + 1$; c) $s = \pm 4\sigma + 1$; d) $s = \pm 5\sigma + 1$.

Figure 4-5 (a) and Figure 4-5 (b) show no changes in amplitude as the magnitude of *x* increases. In comparison, the *LoG* method illustrated in Figure 4-6 (a) and (b) is susceptible to changes in amplitude for large sample sizes when the kernel width is too small. This occurs when the derivatives of the Gaussian taking longer to decay to zero and the independent variable, *x* has a high magnitude.

The findings of section 4.2 are twofold. First, the curvature calculated using *LoG* and *DoG* methods accurately reflect curvature calculated analytically for the example sinewave and second, a kernel size of length *2cutoff +1* where *cutoff* = $4\sigma + 1$ ensures there is no amplification of longer signals when calculating curvature using *LoG* and *DoG*.

These findings inform the design and implementation of the experiments in the following section where an LSTM RNN is trained to segment curves. Its performance is analyzed using a range of engineered features that include *DoG*, *LoG* and curvature.

4.3 Curve segmentation using LSTM RNNs

The review of section 2.8 introduced the LSTM RNN. This kind of neural network was explicitly developed to work with data-series where dependencies between information is distributed across the data, often in a causal manner. This neural network therefore seems an appropriate choice to investigate the efficacy of curvature and derivatives as input features to a classifier. In this case they will be used to segment a times series, though, equally, this approach could be extended to a two-dimensional dataset and this is explored in the following chapter when a head profile contour dataset is segmented.

In this section an LSTM RNN is trained on a range of features and feature combinations and their effectiveness in segmenting electrocardiogram (ECG) signals is evaluated. Section 4.3.1 describes the dataset and 4.3.2 discusses feature choices and combinations in detail. Following on from this in section 4.3.3, combinations of these features are used to train the network. The trained networks are then tested and the efficacy of the features and feature combinations are discussed, compared and evaluated.

4.3.1 Dataset Description

The chosen dataset is based upon the publicly available Research Resource for Complex Physiologic Signals' QT Database ECG dataset (Goldberger *et al.*, 2000). The derived dataset consists of 210 ECG recordings sampled at 250Hz and segmented by an automated expert system (Laguna, Jané and Caminal, 1994). Recordings are taken from

105 separate subjects, and each is of approximately a quarter hour each in length. Figure 4-7 (Yochum, Renaud and Jacquir, 2016) shows a segmented EGC signal delineating the QRS and T regions of an ECG signal. The coloured circles identify the peaks and troughs of the respective waves and the black circles represent the boundaries of the P, QRS and T regions.

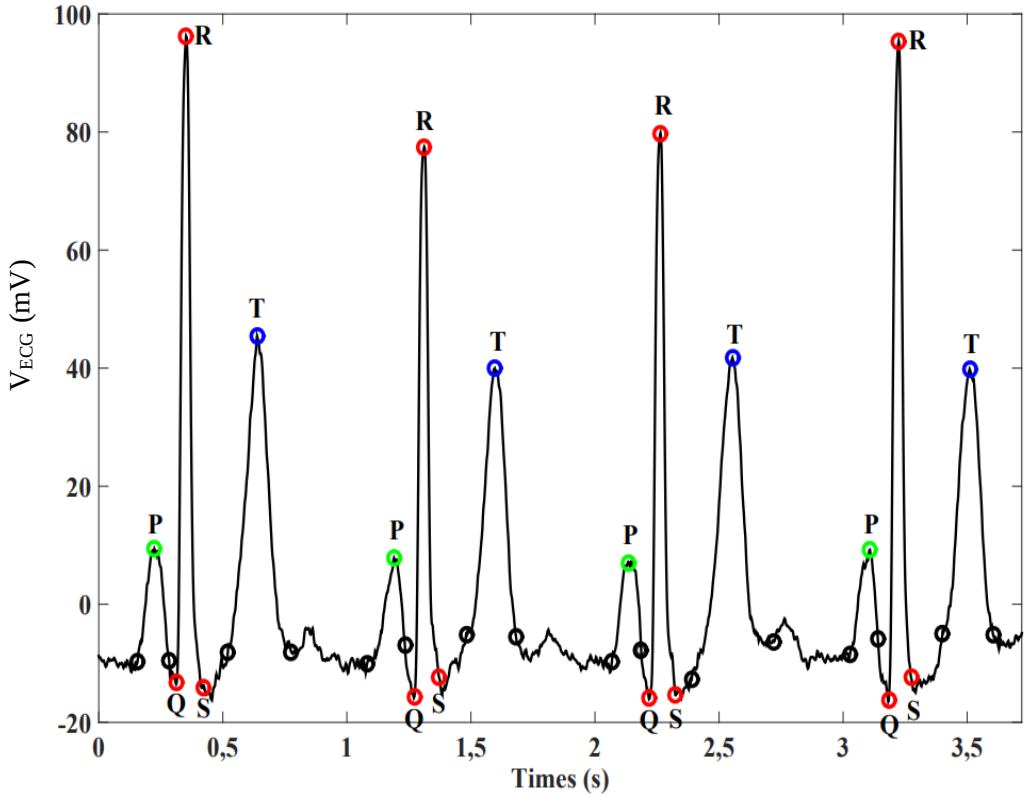


Figure 4-7: P wave, QRS complex and T wave regions of an electrocardiogram (Yochum, Renaud and Jacquir, 2016).

This dataset has properties useful for analysing the effectiveness of curvature and plane curve derivatives. The curvature of these signals also has similar ranges and scales to the contours derived from the two-dimensional head profile image dataset that forms the focus of later chapters in this study.

Focusing on the present ECG dataset, the properties of the dataset that are useful here are:

1. Samples are taken from a wide variety of subjects;
2. ECG signals have recognisable regions- the P wave, QRS complex and the T wave and the aim is, typically, to segment these regions into distinct subsets;
3. these regions are recognisable by their local curvature patterns;

4. local signal derivatives have been used in previous work to isolate the regions (Laguna, Jané and Caminal, 1994);
5. these signals display a wide range of concave and convex curvature.
6. The labelled regions include n/a labels representing a "none-of-the-above" classification.

Another reason for using this dataset is related to previous work where the ECG dataset was used to demonstrate the effectiveness of LSTM RNNs (Mathworks, 2020c). The following discussion documents the reproduction of this work and the features used as inputs to the LSTM RNN and extends it to evaluate the effectiveness of curvature and curve derivatives as input features. In the next sections the selection of features for use with an LSTM recurrent network is discussed, the network architecture and any necessary modifications are defined, and the results of the classification accuracies are presented. In addition, the run-time efficiency of the feature pre-processing and generation algorithms is analysed and compared with the methods used in Mathworks (2020b).

4.3.2 Feature choices

The proposed features used to evaluate the accuracy of the recurrent LSTM network are enumerated in Table 4-1 overleaf, together with a brief outline of the purpose for selecting these features. The replication of the results of previous work is important as it will provide some confidence that the neural network implementation and results are reliable, and it will also provide a benchmark for assessing the features identified in the table.

Additionally, curvature as a feature is investigated at a range of scales; methods of calculating both first and second derivatives as features are investigated as is their effectiveness as input features; and combinations of these features as inputs to the LSTM are also evaluated.

One dimensional input feature vector		
	Feature	**Purpose**
1	Raw ECG signal	Replicate results of previous work (Mathworks, 2020c).
2	Band pass filtered signal	
3	Normalized, curvature of Gaussian filtered signal, σ=3	Assess the effect of standard deviation, σ filter parameter on accuracy of network and evaluate accuracy of curvature as an input feature.
4	Normalized, curvature of Gaussian filtered signal, σ=2	
5	Normalized, curvature of Gaussian filtered signal, σ=1	
6	Central difference of raw data	Evaluate effectiveness of central difference method derivative as a feature and additionally evaluate the effect of pre-filtering signal.
7	Central difference of Gaussian filtered signal, σ=3	
8	First order derivative of Gaussian filtered signal.	Evaluate its effectiveness as an input feature.
9	Second order derivative of Gaussian filtered signal.	
Two dimensional input feature vector		
10	1^{st} + 2^{nd} order derivatives	Evaluate the effectiveness of combinations of 1^{st} order derivative, 2^{nd} order and curvature as input features.
11	Curvature and 1^{st} derivative	
12	Curvature and 2nd derivative	
Three dimensional input feature vector		
13	Curvature, 1^{st}, and 2^{nd} derivatives	Evaluate the effectiveness of curvature, the 1^{st} order derivative and the 2^{nd} order derivative as an input feature.
40 dimensional input feature vector		
14	FSST of raw ECG signal.	Replicate results of previous work (Mathworks, 2020c).

Table 4-1: Proposed features used to evaluate accuracy of a recurrent LSTM Network.

4.3.3 LSTM DNN architecture and training

This section defines the network architecture, the training and testing procedure, and explains the choice of network parameters. Since the LSTM network replicates the experiment of Mathworks (2020c), the LSTM RNN architecture is not adjusted except to alter the number of inputs when combining features.

The network comprises of:

- 200 hidden units;
- a fully connected output layer with 4 outputs corresponding to,
    - the P segment,
    - the QRS complex,
    - the T segment and a neutral, none of the above, classification;
- and a softmax layer.

The network is trained using mini-batches with an adam optimiser and a minibatch size of 45. Shuffle at every epoch is set to true. The initial learning rate is set to 0.01, the learning rate drop period is set to 3 and the gradient threshold is 1. The learning rate schedule is set to 'piecewise.' Training of the network stops after 10 epochs since, for each feature or combination of features used, the testing accuracy has plateaued.

For each feature or feature combination we train and test the LSTM RNN using a 70:30 train:test dataset ratio. The entire dataset comprises of approximately 46 million samples.

4.3.4 Results and Comparisons of Network Accuracy

The network architecture parameters remain fixed. A multi-class confusion matrix is generated for each network/feature combination of a trained LSTM RNN and from this an overall accuracy figure is calculated along with, for each class, its precision, recall and F1 score. The support is also stated for each class's test data. A review of precision, recall, F1 scores, support and accuracy was provided in section 2.7 and Appendix D and the results shown here use these metrics to evaluate the classifiers' success in segmenting the dataset.

The LSTM network architecture remains fixed except for the occasions where the input feature vector changes dimension and then the input layer is modified accordingly.

4.3.4.1 Raw ECG signal

The accuracy of the Raw ECG signal forms the benchmark against which the remaining networks are compared. Table 4-2 summarizes the results. The network has some success in classifying the regions of interest during the segmentation process with a 70.3% overall

accuracy and a macro F1 score of 65.5% indicating that it has moderate success as a classifier. There is some variation between interclass scores and there is some imbalance between class sample size (see the support column of the tables for details). The variation in support is due to the nature of the dataset and its segmentation. For example, the QRS complex is shorter in duration and so has less samples belonging to this class. Nevertheless, its individual shape has resulted in the classifier attaining a higher Precision, Recall and F1 score than the P and T cycles of the ECG.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	39.26	74.56	51.44	1528375
QRS	60.76	79.75	68.97	2039160
T	57.77	78.68	66.62	3686542
n/a	86.97	65.75	74.89	6780923
Overall Accuracy	**70.30%**			

Table 4-2: Evaluation of LSTM Network with Raw ECG signal as input feature.

4.3.4.2 Bandpass Filtered ECG signal

The filter used here is that defined in (Mathworks, 2020c). It is an IIR bandpass elliptic filter with 60dB roll-off, 0.1dB ripple, with a pass band between 0.5Hz and 40Hz. The sample rate of the filtered signal is 250Hz.

Table 4-3 shows an improvement in each class's F1 score and is probably due to the removal of base line, low frequency movement due to breathing and a reduction in sampled noise.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	50.46	65.99	57.19	1528453
QRS	72.74	77.50	75.04	2039082
T	75.30	79.60	77.40	3686542
n/a	80.67	73.27	76.79	6780923
Overall Accuracy	**74.81%**			

Table 4-3 Evaluation of LSTM Network with Bandpass Filtered ECG as input feature.

4.3.4.3 Normalized Curvature (σ=1) Feature

The curvature feature is calculated directly from the raw ECG signal data and improves on the bandpass filtered signal as an input feature with an increase in accuracy of 1.94%. QRS complex and T wave F1 scores differ marginally (about +0.41% and -1.19% for T and QRS wave respectively) though the P and n/a F1 scores improve by +4.5% and +3.03% respectively.

This is an improvement. Curvature is calculated using a kernel of length 19 compared with the bandpass filter that has 46 coefficients. Note that the curvature calculation also includes within it an automatic filtering operation as described in section 4.2.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	53.84	72.22	61.69	1528453
QRS	71.65	76.18	73.85	2038562
T	73.88	82.19	77.81	3687062
n/a	85.01	75.22	79.82	6780923
Overall Accuracy	**76.75%**			

Table 4-4: Evaluation of LSTM Network with Normalized Curvature as input feature (σ=1).

4.3.4.4 Normalized Curvature (σ=2) Feature

When the standard deviation, σ, is changed from 1 to 2 we see a slight decrease in the overall accuracy of about 0.75%, the slight changes in the P,QRS, T and n/a F1 scores reflect this. This is important as it emphasizes the correct choice of σ for a time series signal or, if used in the special domain, the scale of an image.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	54.68	69.06	61.03	1528453
QRS	73.68	74.14	73.91	2038562
T	74.21	79.0	76.53	3687062
n/a	82.42	76.20	79.19	6780923
Overall Accuracy	**75.98%**			

Table 4-5: Evaluation of LSTM Network with Normalized Curvature as input feature (σ=2).

4.3.4.5 First Derivative of Gaussian Feature

Since the 1st derivative has been used in previous work (Laguna, Jané and Caminal, 1994) as part of a hand-crafted expert system used to segment ECG signals, then the *DoG* feature was expected to perform well. It achieved a surprisingly good accuracy of 85.28%. A significant jump from the baseline ECG raw signal accuracy of 70.3% and, also, it improves upon the curvature feature. This result, together with the observation that the *DoG* both filters and calculates the 1st derivative in one pass, is significant as it supports the use of the *DoG* as a feature capable of fast classification of regions of interest on curves. This finding will be further explored in the context of a 2-dimensional head profile curve to be investigated in the following chapters.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	74.15	83.30	78.46	1528453
QRS	86.27	92.60	89.33	2038562
T	82.63	84.87	83.74	3687062
n/a	88.93	83.91	86.34	6780923
Overall Accuracy	**85.28%**			

Table 4-6: Evaluation of LSTM Network with DoG First Derivative as input feature (σ=2).

4.3.4.6 First Derivative Using Central Difference Method

Replacing the *DoG* derivative with the central difference method to calculate the first derivative gives a good result, though it does not perform as well as the *DoG* with a difference of 3.64% in overall accuracy. This is an important finding for this study since, for this dataset at least, the *DoG* method improves upon the central difference method. This could be related to the lack of smoothing. This hypothesis is investigated in the next section by prefiltering the signal using a Gaussian kernel before the central difference method is applied.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	61.47	78.35	68.89	1528453
QRS	82.33	86.90	84.56	2038562
T	82.27	82.54	82.41	3687062
n/a	85.64	80.33	82.90	6780923
Overall Accuracy	**81.64%**			

Table 4-7: Evaluation of LSTM Network with Central Difference First Derivative as input feature.

4.3.4.7 First Derivative of Pre-smoothed Signal Using Central Difference Method

Pre-filtering the signal before applying the central difference derivative feature has further improved the accuracy of the feature and it is now almost identical to that of the *DoG* method (see next section). Note there is an additional step required to smooth the signal prior to application of the central difference derivative. Here the smoothing function was achieved by first convolving the signal with a Gaussian kernel.

Once more this is a significant finding as the central difference method is worth considering but only if the dataset used has already been pre-filtered. If it is to be used in a fast or real-time application and the signal needs to be filtered, then its use should be avoided and the *DoG* method considered instead.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	74.25	83.06	78.49	1528453
QRS	86.83	92.71	89.68	2038562
T	83.77	85.63	84.69	3687062
n/a	89.43	84.77	87.04	6780923
Overall Accuracy	**85.91%**			

Table 4-8: Evaluation of LSTM Network with Filtered Central Difference Derivative as input feature.

4.3.4.8 The Laplacian of Gaussian Second Derivative Feature

The *LoG* feature performs equally as well as the first derivative in terms of its overall accuracy. This is an interesting result as the second derivative describes the concavity/convexity of curvature whilst the first derivative encodes the degree of curvature at any particular point on the curve. The implication here is that both first and

second derivates, when used together improve the accuracy of the segmentation process. The next section confirms this hypothesis.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	78.20	82.71	80.39	1528453
QRS	88.06	93.28	90.60	2038562
T	80.24	83.92	82.04	3687062
n/a	88.62	84.17	86.34	6780923
Overall Accuracy	**85.20%**			

Table 4-9: Evaluation of LSTM Network with LoG Second Derivative as input feature (σ=2).

4.3.4.9 Combined DoG and LoG Derivative Features

Using the *LoG* and *DoG* as a 2 dimensional input vector to the network produces the overall best result, improving upon the FSST's overall accuracy by 2% (see Table 4-12). This combination of features also outperforms the FSST feature vector in three out of the four, per class F1 scores. The FSST improves on the P class F1 score, alone, by 0.69%. This result demonstrates that it is possible to match and improve upon methods that use frequency domain information as features for classification and that, as demonstrated later, a significant, order of magnitude speedup in feature generation is possible.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	79.23	84.96	81.99	1528453
QRS	89.99	94.01	91.95	2038562
T	85.24	85.02	85.13	3687062
n/a	89.19	86.87	88.01	6780923
Overall Accuracy	**87.18%**			

Table 4-10: Evaluation of LSTM Network with First and Second Derivative as input feature (σ=2).

4.3.4.10 Combined Curvature, DoG and LoG Derivative Features

The results of this feature combination are similar to those of the previous section's *DoG* and *LoG* feature combination, though the accuracy and F1 scores are slightly less across all classes. This is surprising since adding an additional feature (curvature) was expected to improve the results. This may be due to the nature of the dataset, however, it is more likely

to be due to the curvature calculation combining both the first and second derivatives into a single scalar feature and therefore losing some salient information. Once again, this result is important to this study as curvature was originally hypothesized to be a good choice as a feature for classification.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	76.77	85.30	80.81	1528453
QRS	89.38	94.14	91.67	2038562
T	84.57	85.24	84.90	3687062
n/a	89.82	86.21	87.98	6780923
Overall Accuracy	86.96			

Table 4-11: Evaluation of LSTM Network with First and Second Derivative, and curvature as input feature ($\sigma=2$).

4.3.4.11 40 dimensional FSST Vector Feature

Frequency domain features are also used in time-series classification and image segmentation. For example, frequencies corresponding to the first 5 peaks in amplitude of the discrete Fourier transform (DFT) have been used as a multi-dimensional feature for classifying human activity (Altun, Barshan and Tunçel, 2010). Additionally, extracting time-frequency features allows a classifier to use local time and frequency information together. The Fourier Synchrosqueezed Transform (FSST) (Auger *et al.*, 2013) used here achieves this, but this additional information comes at a cost as, for each sample, a 40 dimensional vector encoding local time and frequency information needs to be calculated. The FSST used here produces some excellent results with an overall accuracy of 85.48%, second only to the *DoG* and *LoG* feature combination.

ECG Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
P	82.21	83.16	82.68	1528453
QRS	90.45	91.90	91.17	2039082
T	82.09	84.43	83.25	3686542
n/a	86.57	84.67	85.06	6780923
Overall Accuracy	**85.48%**			

Table 4-12: Evaluation of LSTM Network with 40 dimensional FSST vector as input feature.

4.3.5 Summary of overall accuracy and F1 scores

Table 4-13 summarizes the results of section 4.3 and the plot in Figure 4-8 presents a visual summary of these results.

	Feature	Accuracy	Macro-F1 score
1	Raw ECG signal	70.30%	65.48%
2	Band pass filtered signal	74.81%	71.61%
3	Normalized, curvature of Gaussian filtered signal, σ=1	76.75%	73.29%
4	Normalized, curvature of Gaussian filtered signal, σ=2	75.98%	72.67%
5	First order derivative of Gaussian filtered signal.	85.28%	84.47%
6	Central difference of raw data	81.64%	79.69%
7	Central difference of Gaussian filtered signal, σ=3	85.91%	84.95%
8	Second order derivative of Gaussian filtered signal.	85.20%	84.84%
9	1^{st} + 2^{nd} order derivatives	87.18%	86.77%
10	Curvature, 1^{st}, and 2^{nd} derivatives	86.96%	86.35%
11	FSST of raw ECG signal.	85.48%	85.54%

Table 4-13: Summary table of accuracy and macro-F1 scores.

The overall accuracy is presented but, in addition, the macro F1 score for each class is also presented. The macro F1 score is used in this summary chart as it represents an average of all four class F1 scores without weighting, by sample size, for each class. Thus, it treats each of the multi-class F1 scores equally ignoring the support for each class and avoiding biases due to unequal class sizes.

It is found that features that include the *DoG* or the *LoG*, either alone or in combination, result in a significant improvement in the classifier's overall accuracy and macro F1 score.

Additionally, the curvature, which also uses the *DoG* and *LoG*, first and second derivatives as part of its calculation does not perform as well. This is probably due to the

curvature combining both the first and second derivatives into a single scalar feature. This becomes clear when the *DoG* and *LoG* features are interpreted as a two-dimensional vector. Reducing this vector to a scalar value loses the directional information encoded by the vector.

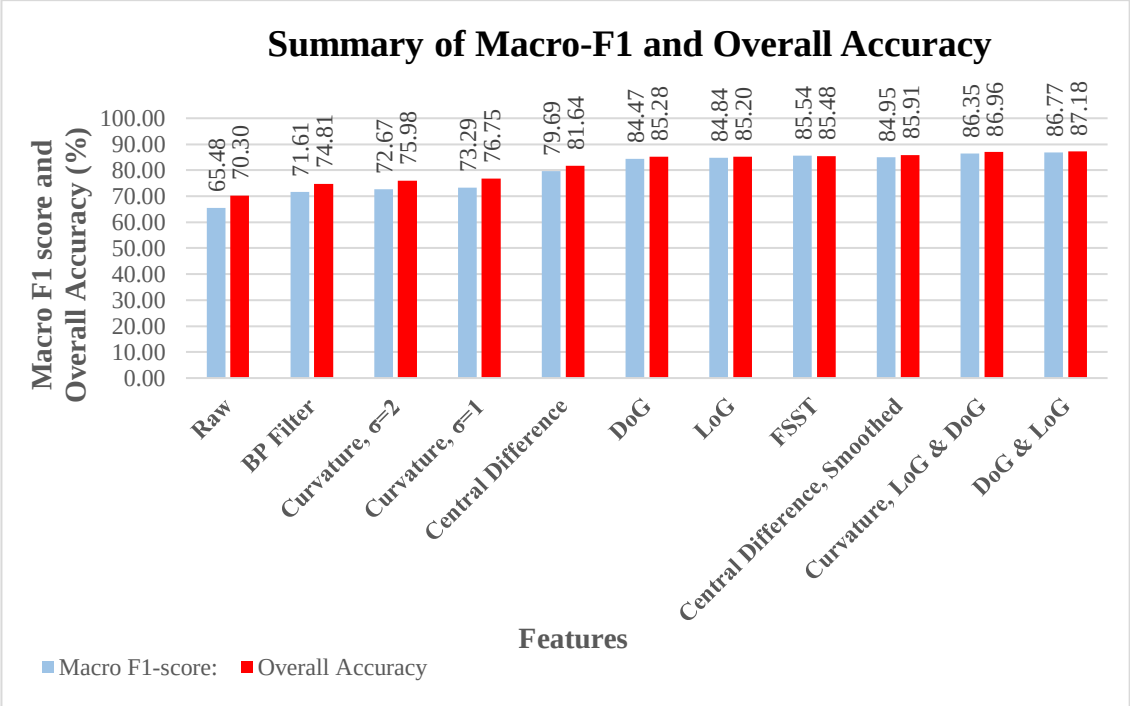


Figure 4-8: Summary of Macro F1 and Overall Accuracy.

The choice of classifier would change the results and no-doubt, further adjustment of the architecture and tuning of the LSTM RNN attributes will also improve the accuracy of the network for all the pre-processed features. Bear in mind, the purpose here is to assess the suitability of the various features as inputs to a given classifier so no further modification of the classifier was attempted. Additionally, this study is based on just one existing dataset. Consequently, more work using a range of datasets is needed to assess the effectiveness of curvature, *DoG* and *LoG* used in time series segmentation.

4.3.6 Runtime Results and Comparisons

Having evaluated the accuracy of the segmentation network, the focus of this section now becomes the evaluation of the execution time of the pre-processing algorithms used to generate the input features for the LSTM network.

4.3.6.1 Timing procedure

The timing procedures used here are detailed in the methodology chapter, section 3.8. The features evaluated in this section had their parameters initialized as follows.
For these timing tests σ was set to 3, both the 1st and 2nd derivative's kernel size was set to 27 (cutoff =4*σ+1, kernel length = 2*cutoff +1), there were 48 filter coefficients used in the bandpass filter and the datasets used were of size 112 000, 56 000 and 28 000 samples. The results of these tests are shown Table 4-14.

Dataset Size, N	**112 000**	**56 000**	**28 000**
1st Derivative (DoG)	188.4µs	103.6µs	57.8µs
2nd Derivative (LoG)	186.0µs	100.7µs	57.1µs
1st Derivative (Central Difference method)	284.9µs	124.5 µs	67.3 µs
Filter	2.12ms	1.08ms	559.8µs
Curvature	3.72ms	1.92ms	973.6µs
FSST	1.37s	684.7ms	340.9ms

Table 4-14: Algorithm execution times to process dataset of size N.

These results are then further combined together where necessary. For example, the pre-filtered, central difference derivative feature's execution time is obtained by adding the filter's execution time to the central difference's execution time (284.9µs + 2.12ms = 2.4049ms), and so on.

From the table and the discussion of these algorithms the time complexity of the 1st and 2nd derivative features is *O(nm)* where *n* is the size of the dataset and *m* is the kernel size, but since n>>m then the complexity is *O(n)*. The curvature algorithm calculates both the first and second derivatives and uses both square roots and 2nd and 3rd powers in the calculation which accounts for a slower run-time. The timings show an approximate complexity of *O(n)* for the derivatives and curvature.

Note MATLAB's gradient() function was originally used to calculate the derivative using the central difference algorithm, however this function appeared to run very slowly in comparison the *DoG* and *LoG* algorithms. Consequently, the central difference algorithm

was re-coded and the times recorded in Table 4-14 reflect this. Nevertheless, given the central difference method's smaller kernel size, the result here appears to be slower than expected (although approximately the same as the *DoG* and *LoG* methods). The central difference method coded here could be optimized further.

4.4 Estimating the effectiveness of the feature pre-processing

Choosing a feature engineering algorithm for fast and accurate classification is often a tradeoff. Ideally the fastest algorithm is not the most accurate and vice versa. To help guide the engineer we summarize the timing and overall accuracy results here as a plot in Figure 4-9.

The top right of the plot is where one would find the ideal feature pre-processor. The higher the point is located, the better the accuracy and the further to the right the faster the algorithm runs. To achieve this, accuracy was plotted against the normalized reciprocal of the run-time.

Note the FSST algorithm is not shown as its runtime is several orders of magnitude greater than the other algorithms.

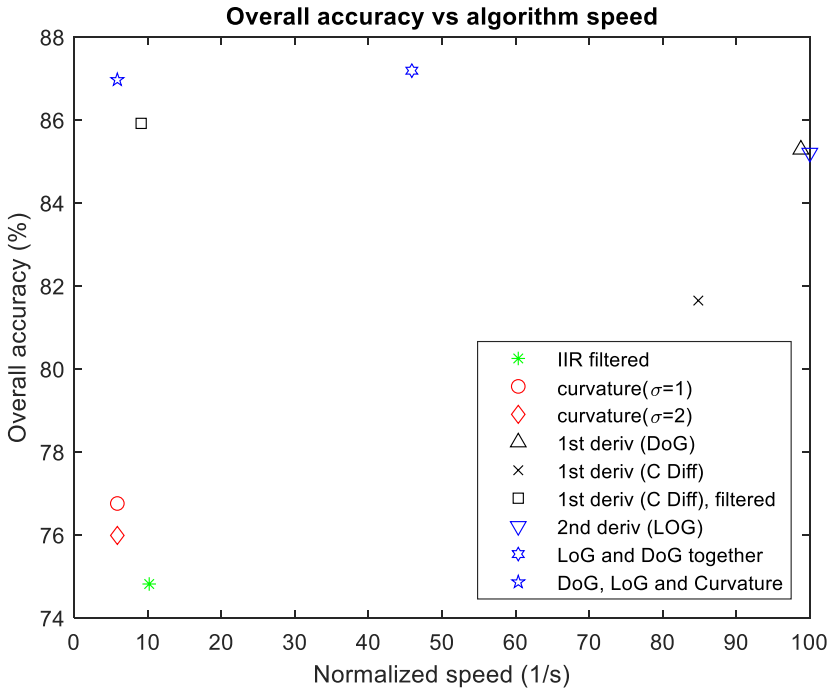


Figure 4-9: Overall accuracy of the feature pre-processing algorithms vs their run-time speed measured as reciprocal time.

A measure of the algorithm's quality would then be the Euclidian distance, given by the L2 norm and called here, Q_{L2}.

$$Q_{L2} = \sqrt{accuracy^2 + NormSpeed^2} \tag{4-2}$$

Table 4-15 shows the Q_{L2} for the feature preprocessors of Figure 4-9. Ultimately the choice depends upon the application.

Ranking	Q_{L2}	**Feature**
1	131.37	2nd deriv (LOG)
2	130.51	1st deriv (DoG)
3	117.74	1st deriv (C Diff)
4	98.52	LoG and DoG together
5	87.15	DoG, LoG and Curvature
6	86.39	1st deriv (C Diff) filtered
7	76.97	curvature(σ = 1)
8	76.20	curvature(σ = 2)
9	75.50	IIR filtered

Table 4-15: Measure of each feature pre-processing's Q_{L2} plotted in Figure 4-9.

We could improve upon this quality estimate by including an additional parameter to encode the perceived importance of accuracy or speed, let us call this parameter the relevance, r. It acts to generalize the quality estimate of the algorithm used. We can modify the Q_{L2} to incorporate this. Equation (4-3) shows how we might do this.

$$adjusted\ Q_{L2} = \sqrt{(2(1-r)s)^2 + (2ar)^2} \tag{4-3}$$

Here, s is the execution speed, *a* is accuracy and *r* is the relevance parameter that ranges from 0 to unity. Table 4-16, Table 4-17 and Table 4-18 show the sorted *adjusted* Q_{L2} for $r = 1$, $r = 0$ and $r = 0.5$. When $r = 0$ the accuracy is ignored, and the execution speed is used to rank the algorithms.

Ranking	Adjusted Q_{L2}	Algorithm Relevance = 0.5
1	1	2nd deriv (LOG)
2	0.993399	1st deriv (DoG)
3	0.896251	1st deriv (C Diff)
4	0.749961	LoG and DoG together
5	0.663436	DoG, LoG and Curvature
6	0.657599	1st deriv (C Diff) filtered
7	0.585915	curvature(σ = 1)
8	0.580071	curvature(σ =2)
9	0.574713	IIR filtered

Table 4-16: Ranking Algorithms as a function of the relevance factor, r = 0.5.

Ranking	Adjusted Q_{L2}	Algorithm Relevance = 0
1	1	2nd deriv (LOG)
2	0.987889	1st deriv (DoG)
3	0.84844	1st deriv (C Diff)
4	0.459003	LoG and DoG together
5	0.102001	IIR filtered
6	0.091054	1st deriv (C Diff) filtered
7	0.058648	curvature(σ = 1)
8	0.058648	curvature(σ = 2)
9	0.058648	DoG, LoG and Curvature

Table 4-17: Ranking Algorithms as a function of the relevance factor, r = 0.

Ranking	Adjusted Q_{L2}	Algorithm Relevance = 1
1	1	LoG and DoG together
2	0.997482	DoG, LoG and Curvature
3	0.985432	1st deriv (C Diff) filtered
4	0.978206	1st deriv (DoG)
5	0.977288	2nd deriv (LOG)
6	0.936453	1st deriv (C Diff)
7	0.880362	curvature(σ = 1)
8	0.87153	curvature(σ = 2)
9	0.85811	IIR filtered

Table 4-18: Ranking Algorithms as a function of the relevance factor, r=1.

Notice the algorithms are ordered by execution speed alone. When $r = 0.5$ both accuracy and execution speed have equal relevance and the rankings match those shown in Table 4-15. When $r = 1$ the execution speed is ignored and the accuracy is used exclusively to rank the algorithms. The results shown are normalized. Using the adjusted quality distance estimate outlined here we can see that the calculated distance when r is 0 has a wide range which reflects the wide range of execution speeds and with r = 1 we see the range now reflects that of the accuracies of the pre-processors. Note these results occupy the top three quarters of the accuracy range, that is, they all have accuracies around 75% and above.

Since this study is investigating approaches to fast and accurate segmentation, these results show that *DoG* and *LoG* based features are ranked highly even when there are changes to the application requirements, for example if hardware and processor choices limit the executions speed, then one should still be considering *LoG* and *DoG* based features. As an aim of this study is to identify and develop fast features that are also accurate classifiers then, *r* is set to 0.5 to reflect their equal relevance.

This measure can also be useful in specifying a system's requirements as weighting a requirement's importance in a project is often a qualitative decision so using a quantitative method to rank requirements based on measurable attributes and perceived relevance can simplify decision making.

4.5 Discussion and Conclusions

The accuracy of the network depends very much upon the chosen features. The raw ECG signal alone shows an accuracy of 70.3%, but pre-processing of the raw ECG signal has been shown to be worthwhile. Using the filtered signal as an input feature to the LSTM RNN provides an overall accuracy increase of (74.81 -70.3)/70.3 = 4.51%, however use of the curvature feature improves on this again, with an increased accuracy of (75.98 - 70.3)/70.3 = 5.68%. For this additional 1.17% improvement there is a time penalty, as the curvature takes approximately 1.75 times as long to calculate as the filtered signal. Note the curvature algorithm contains within it an inherent smoothing due to the *LoG* and *DoG* operations.

Using the derivatives of the ECG signal as input features to the LSTM network provides the best performance and efficiency improvements. Using the *LoG* to calculate the filtered second derivative provides an overall accuracy of 85.28%, an impressive improvement of 85.28-70.3/(70.3)=21.31% over the use of the raw signal as an input feature. Notably, this algorithm runs over 9 times faster than the filter pre-processor and 17 times faster than the curvature calculation. The second derivative performs equally as well as the first derivative in terms of speed and overall accuracy. The filtered first and second derivatives of the signal match the overall accuracy of the FSST of the raw signal input feature. Finally, when combined together the *LoG* and *DoG* features provide an improvement in accuracy of (87.18 – 70.4)/70.4 = 23.87%. This outperforms the FSST by 2%.

The FSST 40 dimensional vector does show some good results. As discussed, it is a close second to the combined *LoG* and *DoG* input feature, but the cost in run-time performance is excessive. The FSST feature extractor is about two orders of magnitude slower than all the other candidate feature pre-processing algorithms analysed here.

Looking at the individual classes, the FSST, *LoG* and *DoG* features have similar F1 scores with little separating them. In general, for all features used as input to the LSTM RNN, the P segment remains the most difficult to classify. In this regard, the FSST feature performs marginally better, however the input feature is a 40-dimensional vector and takes considerably more time to calculate than any of the other input features.

The curvature, whilst performing better than both the raw and filtered features, was out-performed by the first and second derivative features either alone or together. This is hypothesized to be due to loss of relevant information when both first and second derivatives are combined into a single scalar value.

Figure 4-9 allows the reader to visually compare the algorithms by their attributes (accuracy and execution speed), however, where a simple quantitative estimate is required the Q_{L2} and adjusted Q_{L2} formulae of (4-2) and (4-3), respectively, may be helpful. The example rankings shown in Table 4-16 to Table 4-18 give a simple quantitative label that reflects the pre-processors' suitability for the task, given the selected relevance, r, indicating the importance of the feature's accuracy and speed attributes.

In summary, the derivatives of ECG signals are good indicators of the onset of the P/QRS/T complexes, accurately segment these regions and are fast to calculate. Using both the first and second derivatives as input features together in an LSTM RNN segmenter-classifier produces the best classification accuracy results of all the input features evaluated here and outperforms the second-best classifier's execution time by two orders of magnitude. Should either accuracy or execution speed relevance change due to project considerations, then *DoG* and *LoG* based features are still excellent choices.

The choice of classifier would change the results and adjusting the architecture and tuning the LSTM RNN attributes will also improve the accuracy of the network for all the pre-processed features. Bear in mind, the purpose here is to assess the suitability of the various features as inputs to a given classifier, not to optimize the classifier alone.

Additionally, this study is based on just one existing dataset. Consequently, more work using a range of datasets is needed to assess the effectiveness of curvature, *DoG* and *LoG* derivatives used in time series segmentation.

In the next chapter we consider segmenting and regressing head profile contours extracted from a 2D binary image using the ideas developed here for the uniformly sampled, univariate ECG dataset.

5 Segmenting face profile contours with RNNs

This thesis proposes novel methods to segment and regress co-ordinates on face contour profiles. In the previous chapter fast and effective feature processing algorithms were developed that could segment a univariate time series dataset. In this chapter these methods are developed further with the aim of designing, implementing and evaluating a more sophisticated process capable of segmenting face profiles and regressing landmarks accurately, given an unseen image. To achieve this, chapter 4's feature processors and underlying algorithms are extended and applied to a plane curve, consisting of a list of two-dimensional co-ordinates that represent sampled points on a face profile's contour. These contours must first be extracted from a 2.5D image dataset. This chapter also includes a description of the profile image dataset used and the creation of a new dataset of labelled, profile contours.

The outline of this chapter is as follows. An overview of the method and procedures involved is outlined first. The image dataset used in this study is introduced in section 5.2. Section 5.3 describes the method and apparatus used to pre-process and label the dataset. Section 5.4 details the process of extracting the profile contours. Sections 5.5 and 5.6 explains how the manual landmarking accuracy can be automatically improved. Section 5.7 details the method used to segment the dataset and section 5.8 details the design, implementation, training, and testing of a suitable classifier used to segment profile images. 5.9 investigates how changes to the LSTM network can improve accuracy, 5.10 demonstrates how landmark positions can be regressed using the segmented dataset and 5.11 evaluates the runtime efficiencies of both LSTM networks. Finally, section 5.12 discusses results and presents conclusions of this chapter.

5.1 Procedure and toolchain

This section identifies the process required to achieve the aim of segmenting and locating landmarks on a face profile contour. Initially, a suitable profile image dataset was identified, reviewed and pre-processed to remove unusable images (see section 5.2).

Next, a software tool was developed to automatically extract profile contours from images. The curves extracted from the profiles were no longer a set of uniformly sampled

univariate time series as was considered in the previous chapter, but instead represented a plane curve as sampled points with each point represented by a pair of co-ordinate values. Following on from this, the RGB image was manually annotated by an expert to produce a list of profile landmarks associated with each image.

At this point the dataset now comprised of:

1. a set of profile images,
2. a set of corresponding landmarks,
3. a set of corresponding profile contour curves.

Next, a second software tool was created that could automatically adjust landmark co-ordinates on the profile contours with the aim of reducing positioning errors. To achieve this, a novel process used the curvature of the profile contours to guide the positioning of the landmarks. At this point the extensions to the dataset consisted of a set of 2-dimensional vectors of x and y co-ordinates describing contour curves of face profiles together with labelled anthropometric landmarks. This dataset now stands alone and can be used to both train and test a suitable classifier or segmentation process in its current form.

The previous chapter described the successful application of an LSTM RNN to segment an ECG dataset. Since this method was able to segment data with good accuracy then it would make sense to implement a similar experiment using the labelled profile dataset. Consequently, a final pre-processing step was required to modify the contour profile dataset. This step involved segmenting the contour profiles into regions of interest such as upper and lower lips, chin, and so on. In order to achieve this, a further bespoke application was developed that used a profile contour and its associated landmarks to segment the regions of interest. From here the extended and processed dataset could be used to train and test a suitable classifier to segment face profiles and regress landmark position co-ordinates.

The final stage of the procedure to automatically segment face contour profiles is the training and testing of a suitable LSTM RNN. This is detailed in sections 5.8 and 5.9 of this chapter. Regression of landmarks is detailed in section 5.10.

In summary, the method and software tools outlined above extend the original dataset and so create a new stand-alone dataset. The extensions comprise of:

1. an anthropometrically labelled set of 2D RGB images,
2. an anthropometrically labelled set of face profile contours,
3. an anthropometrically labelled set of segmented face profile contours.

This dataset is then used to train and test an LSTM RNN. These results are then analyzed and discussed.

5.2 The Notre Dame J2 Dataset

In order to segment a profile and regress landmark co-ordinates a suitable classifier needs to be trained on a dataset of labelled face profile contours. A review of publicly available datasets failed to identify any useable, existing datasets; however, one candidate dataset was identified that could be extended with sufficient effort. The criteria used here to select an appropriate database was twofold:

1. It had to be a "real-world" dataset, that is, the images captured were not pre-processed and any flaws or "holes" in the images must not have been removed. The raw un-processed nature of such a dataset is an advantage for two reasons. First, it would be beneficial during the supervised ML training process used in this study, improving the generalization capability of the models created. Additionally, such a dataset would be representative of images captured in a practical, real-world scenario.

2. Both RGB image and corresponding depth image data was required. This was essential for two reasons. First the manual landmarking procedure required access to the 2D images. For example, locating the labiale superius requires identification of the vermilion of the upper lip, where the red of the lip tissue meets the philtrum. A secondary requirement related to this is the potential to identify the tragus and exocanthion. This would be useful in future work as an alternative method to quantify head posture angle.

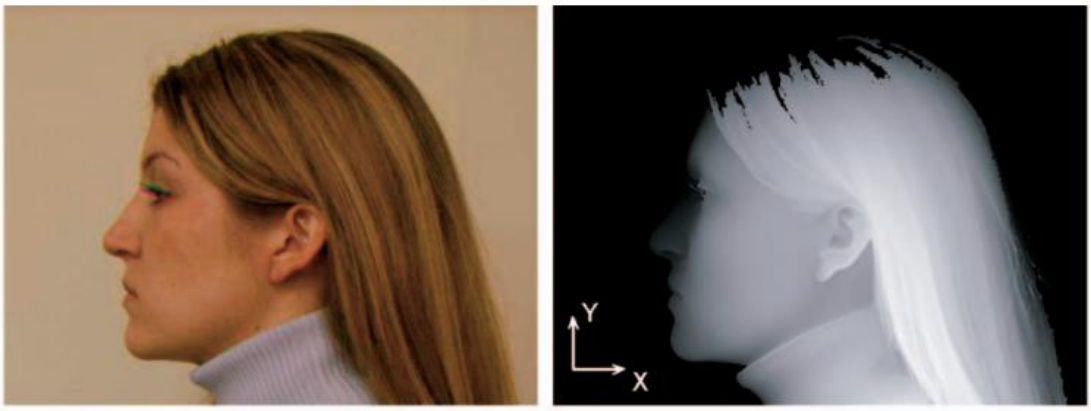
The dataset selected was the Notre Dame University ND-Collection J2 Ear profile dataset (Yan and Bowyer, 2007), consisting of 2413 RGBD (with corresponding two dimensional

RGB) head profile images with the ear visible. Each scanned head profile image comprises of one colour, RGB image of size 640x480 pixels and one 2.5D scan of size 640x480 pixels. Image data was acquired with a Minolta Vivid 910 range scanner (Minolta, 2001).

The Minolta Vivid 910 is a 3D scanner with camera that uses light sectioning triangulation to acquire depth information and has been used in several other 3D head and face detection and recognition studies (Phillips *et al.*, 2005; Liang *et al.*, 2008). Examples of the raw data are shown in Figure 5-1. The left image, (a) is a two-dimensional left profile image and the right, (b) is the corresponding three-dimensional depth image. The subjects are positioned approximately 1.5 metres from the camera.

The first half of the dataset was used in this study. It composed of 985 images (both 3D scans and 2D colour images) of 127 individuals. All were left profile images taken at different times and with slight variations in composition, such as minor changes to the profile pose or translations in camera framing.

The right depth image of Figure 5-1 (b) shows depth coded as a grey-scale (range 0 to 255) with pixels a lighter shade of grey indicating objects closer to the camera and darker pixels, objects further from the camera. Note where hair has scattered or attenuated the light beam towards the top of this image resulting in no reflected rays being detected by the sensor. There is also evidence of scattering or attenuation near the upper eyelid. On the left image, near the eye, there is evidence of separation of the RGB colour components. Scanning of an object using this device can take over two seconds so subjects are expected to avoid any movement during the capturing process. Failure to do this can result in a misalignment of the 2D and 3D images. During the landmarking process, it was observed that images with excess colour separation tended to correlate with landmarks offset from the extracted contour. As the contour was derived from the 3D image, it is assumed movement of the subject during image capture was to blame.



(a) (b)

Figure 5-1: Example images of the ND- N2 Ear collection dataset (Yan and Bowyer, 2007). *(a) 2D colour image; (b) 3D depth image.*

The majority of the images are of useable quality, however several are poorly framed with, for example, parts of the profile cropped, poorly oriented head pose or an arm occluding part of the profile. Facial hair may also occlude part of the profile in some images and, as shown above, scattering or attenuation artifacts often corrupt the 3D image. This limited the usable contour range for this study to be from the gnathion up to the sellion.

The dataset stored RGB images and 3D depth information in separate folders with corresponding images linked by a common filename but different file name extension. Unfortunately, a significant minority of these files were mis-named which reduced the number of useable images in the study. Section 5.8.1 provides details of the final profile contour dataset derived from the Notre Dame N2 dataset.

5.3 Labelling landmarks

An important goal of this study is the accurate estimation of head posture. In order to achieve this using profile contouring it is sufficient to efficiently and accurately identify a small number of landmarks to act as reference points for measurement. Anthropometric landmarks are an attractive option since they are, by design, easy to identify and have obvious characteristics, particularly high curvature. As previously noted, they are also well studied and there are significant databases of anthropometric measurements of the face that include the mean and variance of relative distances between landmarks. These

statistics may be helpful to identify landmarks once the profile has been segmented. Several landmarks are concentrated in a small part of the head profile, between eye level and the chin. Not all are needed to achieve the goal of measuring head posture, it is sufficient to select three or four of these. They could then be used together with the tragus to measure the rotation of the head about the tragus. Alternatively, they could be used with segmented regions of interest to describe the whole face pose.

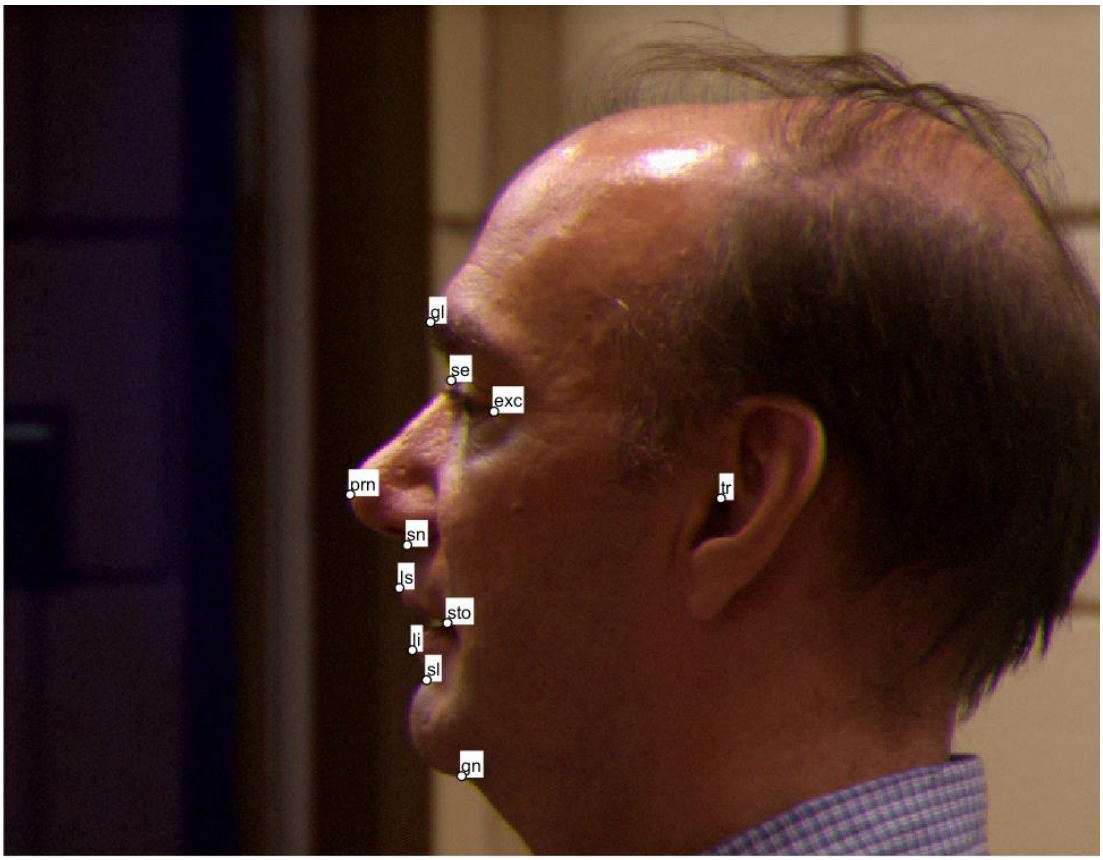
Selecting stable landmarks for measuring head posture ensures measurements are repeatable and accurate. Stable here means that a landmark will remain unchanged when the facial expression changes or when the head rotates in the frontal axis about the tragus in the sagittal plane. Three landmarks that have these properties are the sellion (where the eyebrows meet), the pro-nasale (tip of the nose) and the sub-nasale (where the cartilaginous lowest point of the nose meets the flesh of the upper lip).

When a neutral, unchanging expression is adopted then other landmarks can be used as reference points too. For example, the stomion (contact point where closed lips meet), the labiale inferius (in profile this corresponds to lower vermilion lip), the labiale superius (in profile this corresponds to upper vermilion lip) and the sublabiale (the midpoint of the labiomental groove). Where a landmark occurs near fleshy areas such as the gnathion at the bottom of the chin, there may be movement of that landmark under gravity so measurements may be unreliable if the subject is measured in different positions, for example in the upright then supine positions. For this reason such landmarks were excluded.

5.3.1 Chosen landmarks

The review of craniofacial anthropometric regions and landmarks in section 2.9.2 together with a visual inspection of the images in the Notre Dame dataset indicates the profile below the eye including the pronasale, the subnasale, the labiale superius, the stomion and the labiale inferius are good canditates for localisation since they have the characteristics discussed above and are typically unoccluded. Additionally, as already observed, the three-dimensional scans of this dataset have few deleterious artifacts such as scattering and attenuation in this region. All ears of this dataset are clearly visible so the tragus can be included as a landmark to identify too, since this will feature in future

work and acts as a point of rotation as discussed in section 2.9.3. Figure 5-2 shows these anthropometric landmarks applied to an example profile image from the dataset.



gl: glabella, between the eyebrows; **se**: sellion, deepest point between nose and forehead but not the nasion; **exc**: exocanthion, outer canthus where eyelids meet (termination of the white part of the eye); **prn**: pronasale, tip of nose; **sn**: subnasale, where nose joins the lips;	**ls**: labiale superius, in profile this corresponds to upper vermilion lip; **sto**: stomion, contact point where closed lips meet; **li**: labiale inferius, in profile this corresponds to lower vermilion lip; **sl**: sublabiale, the midpoint of the labiomental groove; **gn**: gnathion, the bottom of the chin; **tr**: tragus or tragion.

Figure 5-2: Profile image together with labelled anthropometric landmarks.

5.3.2 Landmark capturing software

Landmarking was done in two phases. First images were manually landmarked. After all images had been landmarked and checked by a second expert a further process was

applied to automatically improve the accuracy of the landmark locations. This second process is detailed in section 5.5. The following discussion refers to the initial, manual landmarking process.

In order to successfully annotate the images a suite of software tools was developed. These tools were to be used to landmark images. The essential requirements for this software are identified in Table 5-1, together with justification for each requirement.

Requirement Number	Requirement Description	Justification
1	View the dataset.	Each image requires initial screening by the land-marker prior to landmarking.
2	Manually annotate the RGB image with the relevant landmark label selected from a list. Edit and adjust incorrect landmark positions should the need arise.	Once selected, each image needs labelling. The annotator needs to have the option to adjust landmark positions until satisfied.
3	Save: i) a list of landmark positions as a list of pairs of co-ordinates corresponding to pixel positions within an image, ii) the landmark names, iii) a reference to the image file name.	Saved annotations should be kept in a separate database to protect the original image and ensure the annotation database can be stored and distributed independently.
4	Review annotated images and/or adjust incorrect landmark positions should the need arise. The option to update the database must be provided.	For quality purposes, the annotator must be able to revisit saved image annotations and data to both view and edit landmark positions until the correct position has been located.
5.	No images should be changed. Instead, the facility to overlay the landmarks on the corresponding RGB image should be provided.	In order to adhere to the licence agreement, the original dataset must not be corrupted or altered.

Table 5-1: Landmarking software requirements.

Landmarking was undertaken by a local trained expert over a period of several months. Images were randomly chosen from the first half of the Notre Dame dataset. Overall, 985 images were labelled.

Labelling images manually is both time consuming and error prone. In order to minimise errors, the annotator was trained prior to beginning the process. A subset of images and landmarks were then re-checked and any necessary adjustments were made to incorrectly positioned or missing landmarks. Despite this additional step in quality control, the landmarking process cannot produce perfect results. There will always be some variance, although, as reviewed in section 2.9, anthropometric measurements using direct and indirect methods such as photogrammetry are reliable. A sample result of the landmarking process is shown in Figure 5-2.

5.4 Extracting Profile Contours

There are several approaches to extracting contours within an image. Previous chapters identified classic computer vision algorithms for segmentation and edge detection such as Canny edge detection, watershed algorithms and so on (Canny, 1986; Meyer and Beucher, 1990; Woods and Gonzalez, 2017). The advantage of the Notre Dame dataset is that it includes a 3D depth image which can be used to extract relevant parts of the profile efficiently. Using a 3D depth image to extract a contour has other advantages too. Camouflaging or skin colour has no effect on the method, neither does changes in lighting shade, contrast, or backlighting.

Section 2.4 explained how a 3D image can be sectioned along a plane using a plane equation and plane-point distance tests. The resulting curve intersecting this plane then represents the head profile contour. For the images used in this study a simple camera to point distance depth test can be employed. The plane of interest here would have its normal parallel with the *z*-axis and pointing out of screen space towards the viewer. Although this is sufficient for this study, the 3D dataset used here can also be used to extract contours from profiles using planes that are not parallel with the camera's viewing plane. This could be useful should a head profile be slightly rotated about the vertical *y*-axis. In this study, this is not investigated further, leaving it as future work.

Often, and as is the case with the Minolta scanner, a binary mask is encoded as part of the file standard used. This can be used to extract the profile and determine the contour with no further pre-processing necessary. This is the method used here with the Notre Dame dataset. The 3D depth ASCII files were read, the masks converted to a suitable binary image using bespoke C++ and MATLAB scripts and stored in a convenient format, a .png file here.

These binary files were examined manually to determine the quality of the profile, focusing on the region between the eye and chin. Of the 2436 NDJ2D Dataset images converted, 67 images had unusable contours and some poses were not usable, for example part of the head profile was cropped. A few hundred images were incorrectly named, hence the correspondence between an RGB camera image and its associated scan was incorrect, reducing the available useable image samples. Figure 5-3 shows an example profile image of the subject shown in Figure 5-2 above. Note the artifacts due to attenuation and scattering. Once the 3D images had been converted to a 2D binarized format, the next phase was to find the profile contour.

Figure 5-3: Binarized Image profile of previous image.

The algorithm selected is Moore's algorithm described in section 2.4. Images were opened and read, an outline border was placed around the image and a starting point was found by scanning from left to right/top-to bottom until an outline pixel was detected. The algorithm assumes pixels are 8-connected and walks the contour in a clockwise

direction. The framing prevents the contour moving beyond the image extremities. The result of this process is illustrated in Figure 5-4.

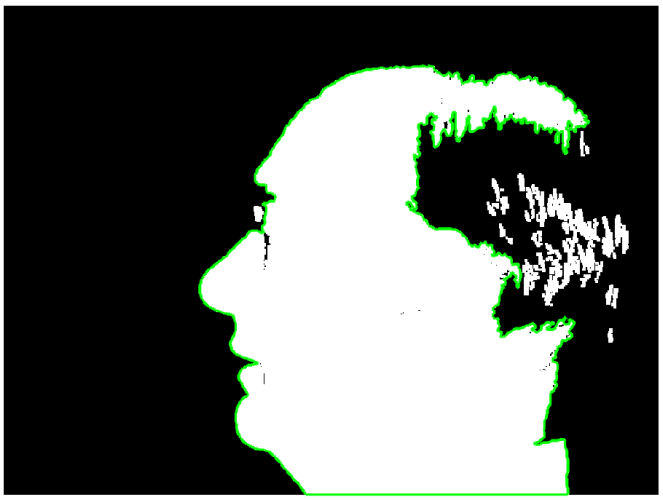


Figure 5-4: Illustration of the contour found using Moore's algorithm applied to the binary image of Figure 5-3.

As the contour algorithm progresses, the contour is stored as a vector of (x, y) co-ordinates representing the whole contour. Only a subset of this contour is required. Extracting this region is achieved initially by empirically selecting a sub-set of co-ordinate values and later, once the dataset was labelled, by selecting a start and end point identified as the nearest co-ordinate to the gnathion and sellion, respectively. This contour subset, together with the manually labelled dataset further extends the Notre Dame dataset. Once the subset of the contour has been identified and cropped, it is saved along with the file name of the processed image. This file name then acts to uniquely identify the contour and links to the generating image. This final dataset can now be input to the procedure to automatically adjust the manually labelled dataset.

5.5 Adjusting Landmarks

A significant finding of this study is that it is possible to automatically improve the accuracy of the landmarking process by adding a further procedure. Other researchers have also attempted to use various approaches based on the concavity and convexity of curvature to either identify landmarks directly or improve the accuracy of manually labelled landmarks (Efraty *et al.*, 2009). These ideas were reviewed in section 2.5. The approach used here differs from these in that the curvature is calculated at all points and

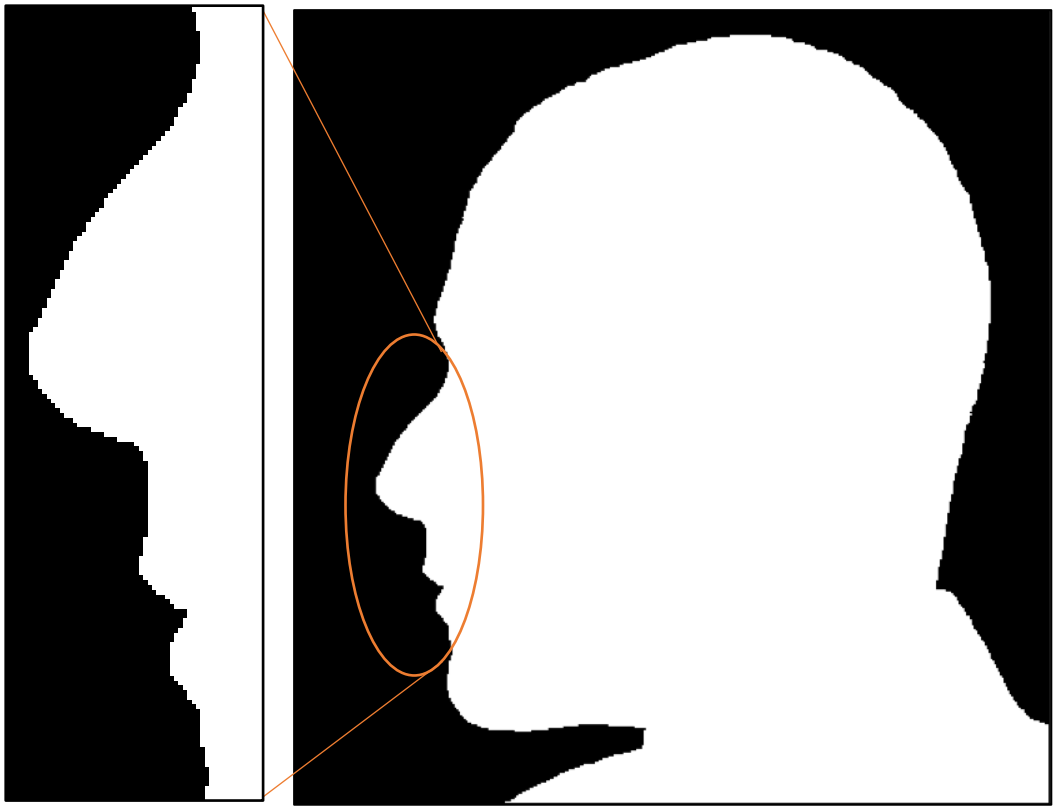
relevant maxima and minima are identified based upon their locality to the manually labelled landmark.

The approach is based on first observing that the3D dataset has additional 3D information that has not yet been used. In particular, the contour profile corresponding to each RGB 2D image is available for further processing. Next, we recognise that the landmarks identified above have useful properties related to the information contained within the contour. In particular, these landmarks are placed at points of high curvature, and this means that two additional and automatic steps can be added to the landmarking process to produce a semi-automatic method. First, any manually labelled landmark that does not sit exactly on the head profile contour can be automatically placed on the profile by selecting the nearest contour point.

A second enhancement would be to then calculate the curvature of the contour and move the point to the nearest maxima or minima of curvature. Whether the point is moved to a maxima or minima depends upon the particular landmark being moved. If the landmark's curvature is convex, then the point is moved to the nearest significant maxima and if concave it is moved to the nearest significant minima. The remainder of this section details this automatic landmark adjustment process, beginning with the extraction of the profile contours from the image datasets.

The image size is 480x640. To place this in context of a typical profile from the dataset, the number of sampled pixels required to represent the profile contour from the sellion down to the gnathion is about 270 to 300 pixels depending upon the composition of the scene. This results in a rather jagged contour as shown in Figure 5-5a which is a close up taken from the synthetic profile head mockup shown in Figure 5-5b.

Applying derivative and curvature operations directly to this contour can lead to errors as described in section 2.6 as step changes at the pixel level within a sampled contour lead to errors in calculating the derivative at finer scales and so low pass filtering of a sampled curve is advisable prior to calculating its derivatives (Farid and Simoncelli, 2004).



(a) (b)

Figure 5-5: Synthetic profile head mockup (b) and a close up (a).

The previous chapter had demonstrated that the Gaussian function has the necessary properties to both smooth and find derivatives efficiently in one or two dimensions. A further attractive feature of the Gaussian kernel is that of separability. This can be taken advantage of to efficiently smooth the contour. Also, when calculating derivatives, as will be needed when calculating curvature, both smoothing and differentiation can be achieved in one pass as described and used in chapters 2 and 4.

To illustrate the effect of filtering the profile contour, a Gaussian kernel was applied separately to each co-ordinate of a profile contour extracted from a binary image. A Gaussian kernel was applied to the *x* co-ordinates of the contour, then separately to the *y* co-ordinates. Figure 5-6 shows the resulting close-up of the smoothed contour superimposed upon the raw, unfiltered profile contour. The standard deviation, $\sigma = 2.83$ pixels here.

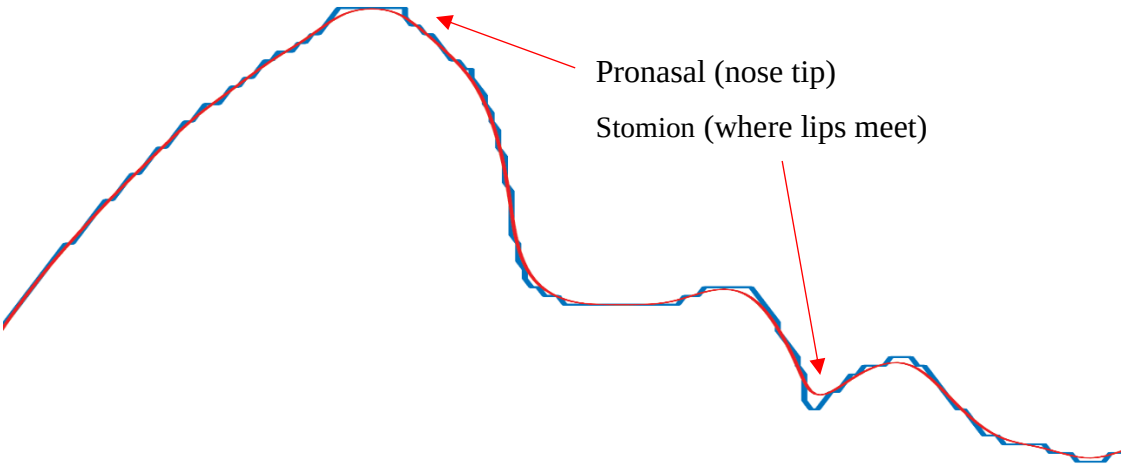


Figure 5-6: Smoothed contour (red) overlayed on raw sampled image (blue).

Although this demonstrates the separability of the Gaussian kernel well and shows it is possible to effectively smooth the raw sampled image, it is important to emphasize that the image contours were not pre-smoothed. Instead, the ideas developed in Chapter 2 are applied, as they were in Chapter 4, to make the calculation of curvature more efficient by combining both smoothing and derivative calculations into one convolution using either the *DoG* or *LoG*.

Here, a recapitulation of the *DoG* and *LoG* process to calculate first and second derivatives, and curvature is offered in the context of a contour curve comprising of a list of (x,y) co-ordinates. The procedure used is shown in the following pseudocode:

function *CalcCurvature(σ,contour)* **returns** *contour_curvature*

DoGKernelWidth ← $8\sigma + 1$

DoG ← *InitializeDoGKernel(DoGKernelWidth)*

LoGKernelWidth ← $9\sigma + 1$

LoG ← *InitializeLoGKernel(LoGKernelWidth)*

contour_dx ← *Convolve(DoG, contourXcoords)*

contour_d2x ← *Convolve(LoG, contourXcoords)*

contour_dy ← *Convolve(DoG, contourYcoords)*

contour_d2y ← *Convolve(LoG, contourYcoords)*

contour_curvature ← *Curvature(Contour_dx, Contour_d2x, Contour_dy, Contour_d2x)*

return *contour_curvature*

Next, a hand-crafted system was developed to finalize the location of the labelled landmarks based on local maxima and minima. This approach was then used to finalize the location of the regressed landmarks on all profile contours. This procedure requires as input: a landmark co-ordinate, ***L;*** a list of head profile contour points, ***P*** and a corresponding profile curvature list, ***C***, calculated from ***P*** using the curvature algorithm above.

L is automatically adjusted as follows:

For each ***L***:

1. Search the list ***P*** for the contour point nearest ***L*** using the Euclidian distance measure (L2-norm). Let this point on the contour be ***P[i]*** where,
   $\boldsymbol{i} = arg\,\min_{x} |\boldsymbol{L} - \boldsymbol{P}[x]|.$
2. Starting at ***C[i]***, search ***C*** for the nearest significant local minimum or maximum of curvature (depending upon the convex or concave nature of the given landmark – see table below). Let ***j*** be the index to this located list element. Therefore ***C[j]*** will contain the corresponding maximum/minimum curvature value.
3. Let ***L = P[j]***.

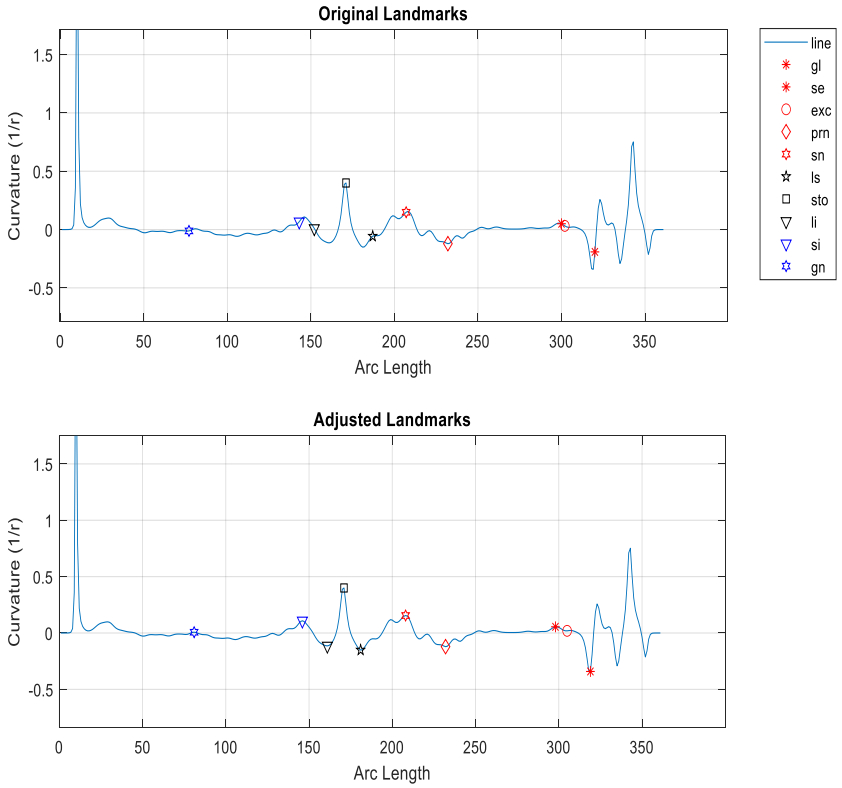
With regard to the convexity or concavity of the curvature around specific landmarks, Table 5-2 lists the landmarks indicating whether each is convex or concave. Note the notion of convexity and concavity here is arbitrary and depends upon the initial presentation of the curve to the algorithm.

Number	**Landmark**	**Concavity/Convexity**
1	Sellion (se)	Concave
2	Exocanthion (exc)	Not on contour
3	Pronasale (prn)	Convex
4	Subnasale (sn)	Concave
5	Labiale superius (ls)	Convex
6	Stomion (sto)	Concave
7	Labiale inferius (li)	Convex
8	Sublabiale (si)	Concave
10	Gnathion (gn)	Concave

Table 5-2: Concavity or convexity of face profile landmarks.

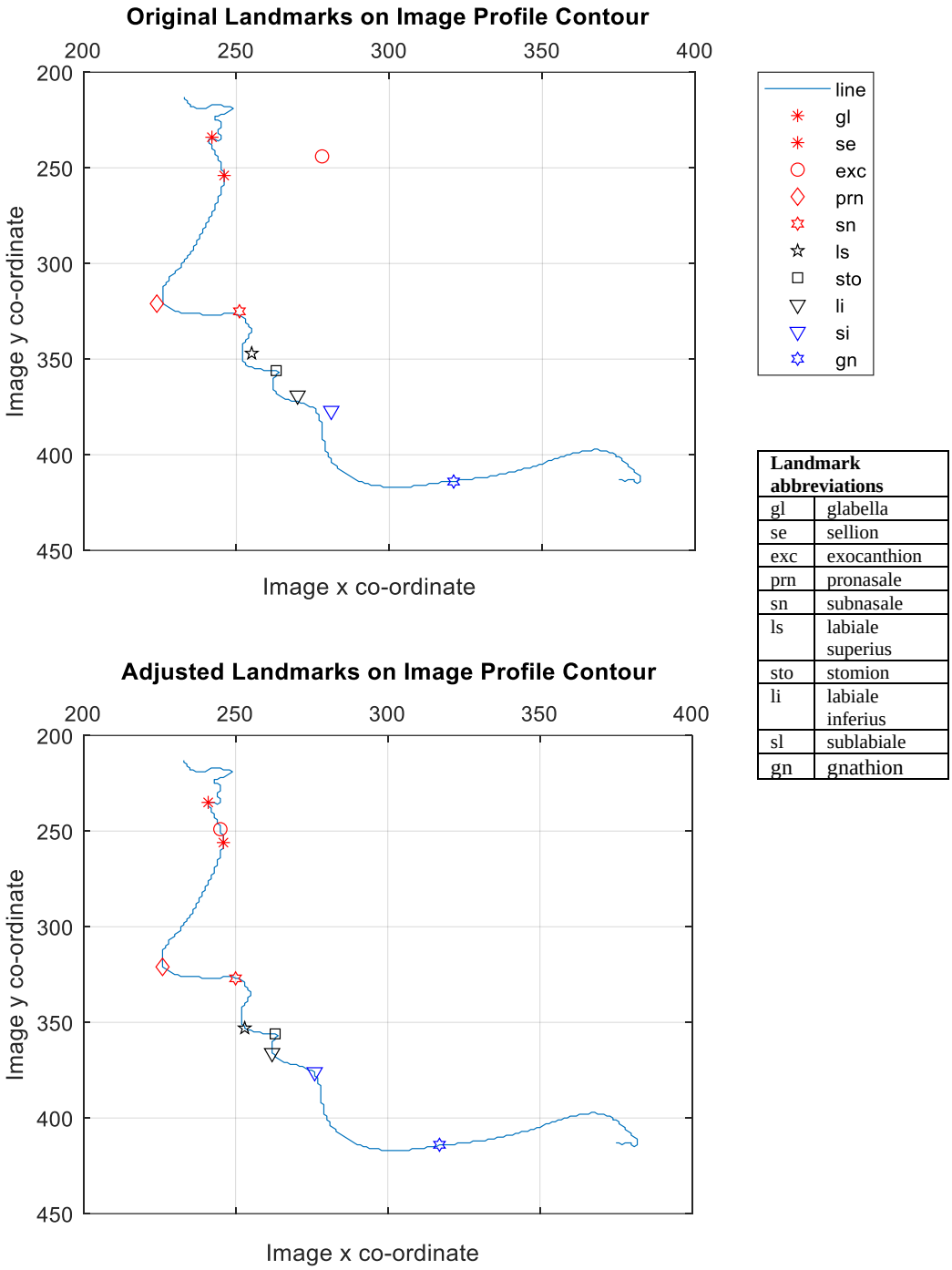
5.6 Adjustment Algorithm Results and Discussion

Figure 5-7 and Figure 5-8 demonstrate the effect of applying the automatic adjustment method described above. Notice that the landmarks have successfully been moved onto the contour and then further adjusted, when necessary to locate the highest point of curvature that relates to the landmark's true location.



Landmark abbreviations	
gl	glabella
se	sellion
exc	exocanthion
prn	pronasale
sn	subnasale
ls	labiale superius
sto	stomion
li	labiale inferius
sl	sublabiale
gn	gnathion

Figure 5-7: Extracted profile curve curvature overlayed with manually labelled landmarks (top). Extracted profile curvature and landmarks after adjustment using the automatic process (bottom).



Landmark abbreviations	
gl	glabella
se	sellion
exc	exocanthion
prn	pronasale
sn	subnasale
ls	labiale superius
sto	stomion
li	labiale inferius
sl	sublabiale
gn	gnathion

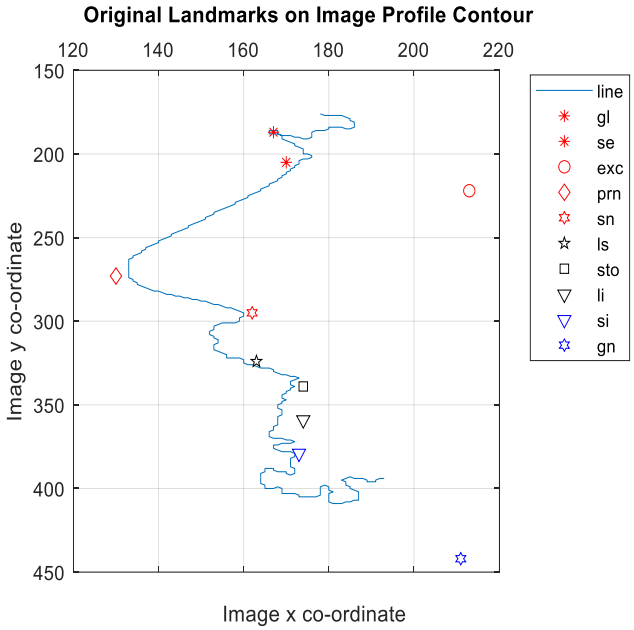
Figure 5-8: Extracted profile contour overlayed with manually labelled landmarks (top). Extracted profile contour and landmarks after adjustment using the automatic process developed here (bottom).

Whilst the automatic landmark adjustment algorithm laid out here works well with this dataset, there are limitations to its use as a method for accurately positioning landmarks around the chin, lips and nose. In particular, facial hair such as moustaches and beards may cause problems if they occlude a landmark. Naturally, this is also a problem when manually annotating images of this kind and sometimes only an estimate can be made by the expert annotator. Figure 5-9 illustrates how the adjustment algorithm works with a pre-labelled image of a bearded face profile.

Scanning of hair often results in aberrations due to attenuation and scattering of the scanner's incident light. This is apparent in the figure. Nevertheless, the algorithm performs well, locating the nearest points on the contour and searching for significant points of curvature to locate landmarks.

The smoothing process acts to reduce spurious curvature and results in some reasonable adjustments to the landmarks particularly around the nose, upper and lower lip areas.

The Minolta camera/scanner used with this dataset can take up to 2 seconds to complete the 3D scan. As already noted, any head movement can cause problems with accurate sampling. The profile of Figure 5-10 show an offset which illustrates this idea. The adjustment algorithm deals with these kinds of error effectively and relocates landmarks onto the contour accurately.



Landmark abbreviations	
gl	glabella
se	sellion
exc	exocanthion
prn	pronasale
sn	subnasale
ls	labiale superius
sto	stomion
li	labiale inferius
sl	sublabiale
gn	gnathion

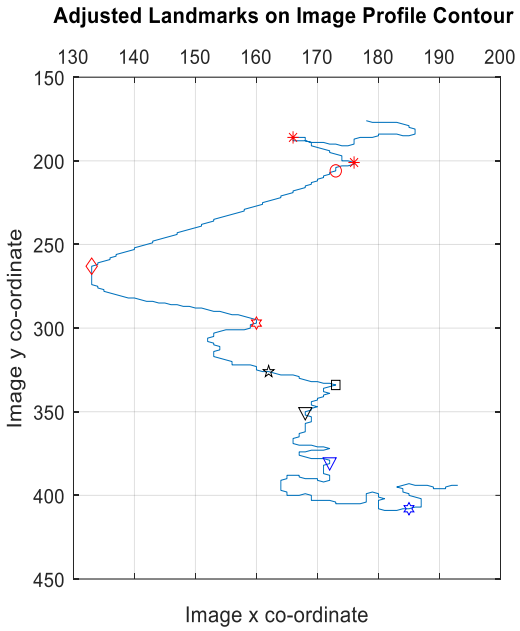
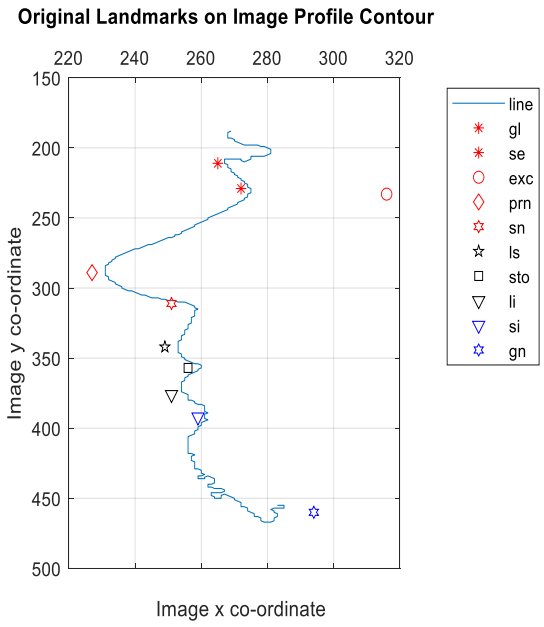


Figure 5-9:Application of the adjustment algorithm upon a bearded profile image. Top: before adjustment; bottom: after application of automatic adjustment algorithm.



Landmark abbreviations	
gl	glabella
se	sellion
exc	exocanthion
prn	pronasale
sn	subnasale
ls	labiale superius
sto	stomion
li	labiale inferius
sl	sublabiale
gn	gnathion

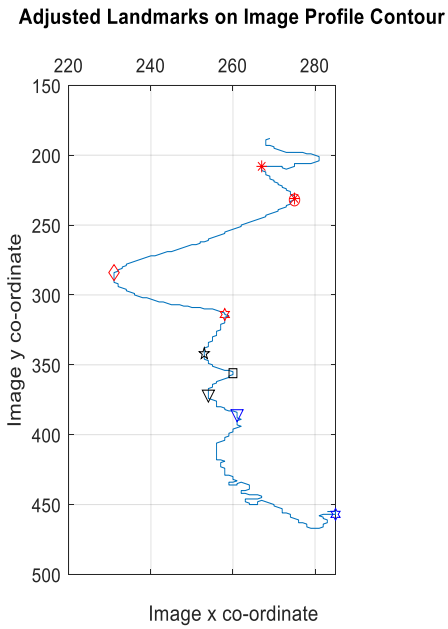


Figure 5-10: Extracted profile contour overlayed with manually labelled landmarks (top). Note the scanning inaccuracy leading to contour/RGB image overlay mismatch. Note the adjustment algorithm deals with this well (bottom), relocating the landmarks correctly on the contour at the expected points.

A final, important consideration is the effect of the automatic adjustment algorithm on the labiale superius. In the anthropometric literature this landmark represents the border of the vermillion line with the upper lip. This also corresponds to a point of high convex curvature which is used by the algorithm to place this landmark. In many individuals this corresponds to the true labial superius, however in many other individuals there

commonly appears a second, higher curvature point located slightly below the labiale superius on the upper lip itself but above the stomion which the adjustment algorithm gravitates towards. This is illustrated in Figure 5-11.

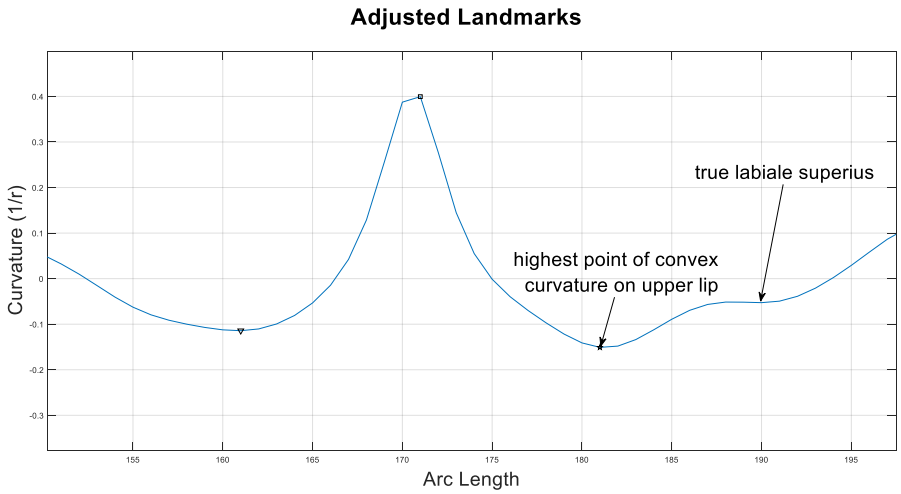


Figure 5-11: Adjusted landmarks, illustrating points of curvature located on the upper lip.

This point of curvature is stable and the algorithm repeatably locates it when it exists in an individual. However, it is not the true labiale superius. In order to select the true labiale superius during the adjustment process the algorithm was modified to provide a choice when selecting the point of convex curvature.

Either point can be used. Traditionally the labiale superius has been used as it is easily located by an expert viewing a colour image. This study has the advantage of being able to calculate profile curvature for every point of the profile contour and so curvature as a feature has been used to guide the final positioning of the landmarks. For this study, the highest point of curvature on the upper lip is used as a landmark.

This is sometimes the true labiale superius and sometimes the highest point of curvature on the upper lip itself. To avoid confusion, in this thesis the highest point of curvature is referred to as the labiale superius.

In order to provide some flexibility in the choice of local minima or maxima, the adjustment algorithm is extended to include an upper and lower maximum range at which

point the algorithm will stop searching for better local maxima or minima. This range is termed the *extent* of the search and is set empirically to ± 9 pixels. Finally, for this algorithm a range of σ values between 1 and 3 were used to smooth the curve. A value of $\sigma = 3$ gave the best results for use with the automatic adjustment algorithm. This was probably due to the reduction in local minima near a target landmark.

At this point the extended dataset comprises of:

1. A dataset of semi-automatically adjusted landmark co-ordinates related to the original Notre Dame dataset images,
2. a profile contour curve subset extending from around the sellion to the gnathion. (again the co-ordinates of each point correspond the Notre Dame image dataset),
3. the corresponding curvature values for each point on the profile contour curve.

Having identified the best choices for the parameters used in the automatic landmark adjustment algorithm, and having completed the creation this new dataset, the dataset can now be further processed to generate candidate features that can be used to train a classifier to segment face profile contours and regress local landmarks. The following section discusses the methods used to achieve this and describes the implementation of a suitable classifier used to segment the profile. A selection of features informed by the experiments of chapter 4 are used here. For each feature, or combination of features, the accuracy of the resulting classifier/segmenter is subsequently analyzed and evaluated.

5.7 Segmenting profiles

Once the semi-automatic process of labelling key landmarks has been completed then the dataset can be used to engineer features and train suitable classifiers. The labels themselves can be used as inputs in combination with the contour and engineered features, or the dataset could be further segmented into specific regions which would increase the amount of useful information available to guide the training of a classifier. This latter approach was used in the previous chapter to good effect with a uniformly sampled dataset and it is repeated here.

To achieve this the dataset is further extended by defining regions of interest delimited by the landmarks. Regions between these labels are segmented resulting in all points

between labels being classified as belonging to its accorded region. The result of this process is an additional vector of categorical variables equal in length to the profile contours with each point labelled with its relevant categorical variable. The following discussion details the relevant data structures used in the segmentation process.

Each profile contour consists of a vector of contour co-ordinates, each co-ordinate represents a pixel on the image profile. The size of the contour could vary but is set to a length of $n=361$ in order to capture all relevant landmarks. It is noted that the positioning of all subjects in the dataset is relatively constant and so profile sizes, whilst they do vary, approximately occupy a similar space. The vector is illustrated in Figure 5-12. Note each co-ordinate is a two dimensional vector and is referenced by the contour element number, which corresponds approximately to the parameterized arc-length of the curve.

1		2		3		4		5						n-2		n-1		n	
x	y	x	y	x	y	x	y	x	y					x	y	x	y	x	y

Figure 5-12: Contour vector containing screen co-ordinates of the two-dimensional image.

The labelled landmarks are referenced by the contour element that corresponds to the landmark co-ordinates. The regions identified for segmentation are listed in Table 5-3 and are delimited by the labelled landmarks. For example, the Philtrum extends from the labiale superius to the subnasale.

Region (label used)	**Start Point**	**End Point**
n/a (Not defined)	Contour beginning	Gnathion (gn)
Chin (chin up to lower lip)	Gnathion (gn)	Labiale inferius (li)
Lower lip	Labiale inferius (li)	Stomion (sto)
Upper lip	Stomion (sto)	Labiale superius (ls)
Philtrum	Labiale superius (ls)	Subnasale (sn)
Columella	Subnasale (sn)	Pronasale (prn)
Dorsum nasi	Pronasale (prn)	Sellion (se)
n/a (Not defined)	Sellion (se)	Contour end

Table 5-3: Definition of regions for profile segmentation.

Using the labelled landmarks and the contour vector described here, a further vector mask was generated labelling each point on the contour with its designated region as shown in Figure 5-13. The resulting mask was of equal length to the contour. Figure 5-14 illustrates a profile image segmented using the regions defined in Table 5-3 above. Note that the chin region label incorporates the region between the gnathion and labiale inferius. This was done in order to simplify the labelling of the region.

Element	1		2		3		4		5								n-2		n-1		n	
Label Mask	n/a		n/a		n/a		gn		gn								se		se		se	
Co-ordinates	*x*	*y*	*x*	*y*	*x*	*y*	*x*	*y*	*x*	*y*							*x*	*y*	*x*	*y*	*x*	*y*

Figure 5-13: Example segmentation mask generated from contour and region labels.

Both the regions of interest identified here and the landmarks describe the profile posture. The landmarks useful here are those that remain unchanged with posture, whether the subject be supine or prone and no matter what facial expression or pose is held by the subject. However, all subjects will be expected to have a neutral expression during the inference stage when classifying the head posture of a new, never before seen, subject.

At this point the dataset is ready to be used to train and analyze a new classifier. In the next section we design, implement and analyze a profile segmenter using a LSTM RNN using this dataset of segmented contours.

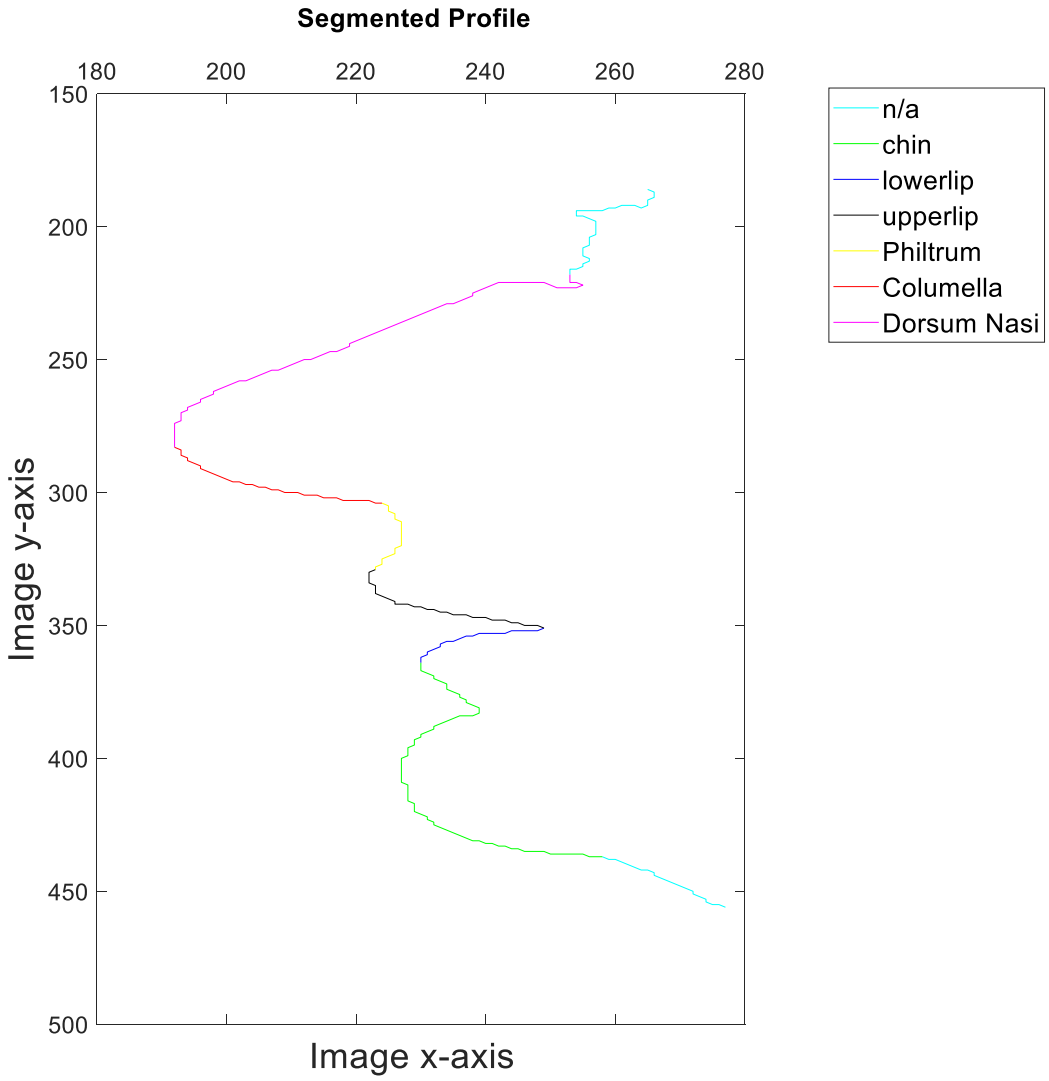


Figure 5-14: An example profile segmented into regions.

5.8 Curve Segmentation using LSTM Neural Network

Due to the relatively small profile contour dataset available it was deemed essential to engineer suitable, effective features to train the network since it was assumed that development of an end-to-end classifier that could learn the required features from the raw dataset would rely upon a large training dataset. Additionally, as one aim of this study is to identify fast methods for posture estimation and measurement it would also make sense to develop an efficient classifier capable of fast inference. For example, a deep neural network that is capable of learning both the necessary features and locations of landmarks given only the raw input contour, would require not just a large training

dataset but also a more complex architecture which would require additional calculations during inference.

As a simpler network would be the more desirable option, the emphasis is on engineering suitable features for training and inference. This section explores this approach, using both the ideas documented in chapter 4 and the region segmentation methods developed in this chapter to inform the engineering of features and assess their effectiveness in segmenting and regressing landmarks in an LSTM network.

5.8.1 Dataset Description

The dataset of segmented and labelled profile contours created previously in this chapter was used to generate suitable features to train the LSTM network. The dataset initially consisted of 693 segmented and labelled contours comprising of 86832 categorically labelled points. These were generated from the Notre Dame dataset and includes a minority of images that have occlusions, missing regions or are poorly framed profiles. These were included to improve the generalization properties of the LSTM network. Errors in file naming of corresponding 2D and 2.5D depth files further reduced the number of images used to 648. Of these, the test set comprised of 194 images and the training set 454. This remained unchanged during the investigation detailed in this study.

Segmented profiles consist of seven regions as illustrated in Figure 5-14. These were listed in Table 5-3.

Class imbalance is an important consideration in pre-processing datasets. The dataset is mildly imbalanced, with the columella of the nose having the largest support. Class size ratios for philtrum:columella is 2:3, upper lip:columella 2:3 and lower lip:columella 1:2. This imbalance is by no means extreme and the results of this section show the models generated still produced good results.

Since the aim of the segmenter is to accurately identify a small number of landmarks that remain invariant under transformation and facial expression, then only a subset of the profile is required. This means the longer “n/a” labelled parts of the profile can be trimmed improving the balance of the data categories. Additionally, the chin region is superfluous since it contains no useful landmarks and in comparison to other regions it

also adds to data imbalance. Consequently, the segmented contour profile dataset has been adjusted. Table 5-4 shows the regions used after adjusting the segmented profile to take into account these changes.

Region (label used)	**Start Point**	**End Point**
n/a (Not defined)	7 samples before the Gnathion	Gnathion (gn)
Lower lip	Labiale inferius (li)	Stomion (sto)
Upper lip	Stomion (sto)	Labiale superius (ls)
Philtrum	Labiale superius (ls)	Subnasale (sn)
Columella	Subnasale (sn)	Pronasale (prn)
Dorsum nasi	Pronasale (prn)	Sellion (se)
n/a (Not defined)	Sellion (se)	7 samples after the sellion

Table 5-4: Adjusted segmented contour profile. Compare with Table 5-3 above.

The resulting dataset now consists of contours of varying length, from approximately 14 sample points up to 155.

5.8.2 Feature Choices

The features generated here are informed by those used in Chapter 4 to segment the univariate ECG signal. The proposed features used to evaluate accuracy of the segmenting LSTM network are enumerated Table 5-5 overleaf.

The features are grouped by the dimensionality of the feature vectors and a brief outline of the purpose for selecting these features is also included.

One and two dimensional input feature vector		
	Feature	**Purpose**
1	Raw profile contour curve	Provides baseline for comparison.
2	Normalized, curvature of Gaussian filtered signal, σ=3	Assess the effect of standard deviation, σ filter parameter on accuracy of network and evaluate accuracy of curvature as an input feature.
3	Normalized, curvature of Gaussian filtered signal, σ=2	
4	Normalized, curvature of Gaussian filtered signal, σ=1	
5	First order derivative of Gaussian filtered signal.	Evaluate its effectiveness as an input feature.
6	Second order derivative of Gaussian filtered signal.	
Three and four dimensional input feature vector		
7	1^{st} + 2^{nd} order derivatives	Evaluate the effectiveness of combinations of 1^{st} order derivative, 2^{nd} order and curvature as input features.
8	Curvature and 1^{st} derivative	
9	Curvature and 2nd derivative	
Five dimensional input feature vector		
10	Curvature, 1^{st}, and 2^{nd} derivatives	Evaluate the effectiveness of curvature, the 1^{st} order derivative and the 2^{nd} order derivative as an input feature.

Table 5-5: Proposed features used to evaluate accuracy of a recurrent LSTM Network in segmenting face profiles.

5.8.3 LSTM DNN architecture and training

The LSTM RNN architecture is that used in chapter 4. Only the input layer is adjusted to alter the number of inputs to take into account the two dimensional (x, y) co-ordinates of the curve and combinations of input features.

The network comprises of:

- 150 hidden units;
- a fully connected output layer with 5 outputs corresponding to,
    - n/a (not defined),
    - lower lip,
    - upper lip,
    - philtrum,
    - columella,
- a softmax layer.

The network is trained using mini-batches with an adam optimiser and a minibatch size of 45. Training of the network stops after 15 epochs since, for each feature or combination of features used, the testing accuracy has plateaued.

For each feature or feature combination we train and test the LSTM RNN using a 70:30 train:test dataset ratio. The dataset size was 648 landmarked contours. Of these, the test set comprised of 194 images and the training set 454.

All experiments were performed on machine with an Intel core i7-7700 CPU with 32GB RAM and an Nvidia 1080Ti GPU.

5.8.4 Results and Comparisons of Network Accuracy

This section follows the experimental methods used in section 4.3.4 and discussed in chapter 3. Here, the network architecture parameters remain fixed as defined in section 5.8.3. In each experiment a multi-class confusion matrix is generated for each network/feature combination of a trained LSTM RNN and from this an overall accuracy figure is calculated along with, for each class, its precision, recall and F1 score. The support is also stated for each class's test data.

The LSTM network architecture remains fixed except for the occasions where the input feature vector changes dimension. Then the input layer is modified accordingly. The following section details the results obtained for each of the features and feature

combinations identified in Table 5-5. Following this an analysis and discussion of the results is provided.

5.8.4.1 Raw profile contour

The raw face contour profile curve is a two dimensional vector of sampled (x, y) image pixels and the accuracy of the network trained on this feature was to be the benchmark against which the remaining networks are compared. Table 5-6 summarizes these results. The network has very little success in classifying the regions of interest during the segmentation process with a 51.33% overall accuracy and a macro F1 score of 37.5% indicating that its performance is better than random guessing from 5 classes but is still not good. It fails to correctly identify any points within the philtrum region. Since this feature is not effective as an input to the classifier the following curvature features form an initial basis for comparison in this section.

Profile Class	**Recall (%)** TP/(TP+FN)	**Precision (%)** TP/(TP+FP)	**F1 Score (%)**	**Support**
Columella	94.73	50.41	65.80	6112
Philtrum	0.00	0.00	0.00	3859
LowerLip	64.99	54.56	59.32	3028
n/a	5.15	50.36	9.35	2716
UpperLip	55.23	51.24	53.16	4298
Overall Accuracy (%)	**51.33**	**Macro F1 Score (%)**	**37.53**	**20013**

Table 5-6: Evaluation of LSTM network with raw profile contour curve as input feature.

5.8.4.2 Normalized curvature (σ=3) feature

Curvature is a one-dimensional vector describing the curvature of the contour profile curve. It is calculated using the full curvature equation of (2-6). The overall accuracy of the classifier has improved significantly to 76.42% with a macro F1 score of 76.47%.

The architecture of the classifier remained unchanged, and the size of the feature vector has halved in comparison with the raw curve feature, yet its accuracy at the class level

and overall macro-F1 score has improved significantly when compared with the raw curve input feature.

Profile Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
Columella	82.41	76.99	79.61	6112
Philtrum	63.75	65.93	64.82	3859
LowerLip	84.28	90.05	87.07	3028
n/a	76.14	75.53	75.83	2716
UpperLip	73.89	76.20	75.03	4298
Overall Accuracy (%)	**76.42**	**Macro F1 Score (%)**	**76.47**	**20013**

Table 5-7: Evaluation of LSTM network using normalized curvature with σ=3 as input feature.

5.8.4.3 Normalized Curvature (σ=2) Feature

When the standard deviation is changed from 3 to 2 we see a decrease in the overall accuracy of about 1.5% in Table 5-8. Interestingly the classifier has poor recall. The confusion matrix related to this table reveals that it confuses the "nose tip" and the "none of the above" (n/a) labelled points with nose tip incorrectly predicted 1354 times and the n/a class correctly predicted 1362 times. A poor result. The columella class's recall has increased by 7.3% at the expense of precision. Otherwise, though slightly lower, the recall and precision follow similar patterns to the previous classifier using the σ=3 curvature feature. This is important as it emphasizes the correct choice of σ can be significant given the scale of an image.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	89.79	70.13	78.75	6112
Philtrum	61.05	63.52	62.26	3859
LowerLip	86.06	90.33	88.14	3028
n/a	50.15	99.71	66.73	2716
UpperLip	74.06	75.30	74.67	4298
Overall Accuracy (%)	**74.93**	**Macro F1 Score (%)**	**74.11**	**20013**

Table 5-8: Evaluation of LSTM network with Normalized Curvature as input feature (σ=2).

5.8.4.4 Normalized Curvature (σ=1) Feature

Using curvature as a feature with σ=1 provides the best classification of this group of curvature features with an overall accuracy of 79% and a macro F1 score of 78%. Table 5-9 shows the upper lip's F1 score has increased by over 6%, the philtrum's by 6.5% and the lower lip's by 5.5% The n/a class still has poor recall as in the previous section's results.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	91.59	72.91	81.19	6112
Philtrum	68.93	68.50	68.72	3859
LowerLip	93.10	93.62	93.36	3028
n/a	51.29	95.67	66.78	2716
UpperLip	78.08	84.22	81.03	4298
Overall Accuracy (%)	**79.08**	**Macro F1 Score (%)**	**78.22**	**20013**

Table 5-9: Evaluation of LSTM network with Normalized Curvature as input feature (σ=1).

5.8.4.5 First Derivative of Gaussian Feature

Building on the work of the previous chapter, the first derivative of the profile curve was investigated next. The previous chapter required only a one-dimensional derivative vector as the ECG data used there was sampled uniformly. Here, a two-dimensional vector is used as the derivatives were taken with respect to the parameterized arc length of both the *x* and *y* sampled positions which made up the profile contour curve.

Table 5-10 shows it achieved an accuracy of 87.57% and an F1 score of 87.89%. This is a significant improvement of approximately 8% on the best curvature feature.

Profile Class	**Recall (%)** TP/(TP+FN)	**Precision (%)** TP/(TP+FP)	**F1 Score (%)**	**Support**
Columella	94.93	89.08	91.91	6112
Philtrum	78.28	75.68	76.96	3859
LowerLip	95.64	92.64	94.12	3028
n/a	90.65	100.00	95.09	2716
UpperLip	77.80	85.31	81.38	4298
Overall Accuracy (%)	**87.57**	**Macro F1 Score (%)**	**87.89**	**20013**

Table 5-10: Evaluation of LSTM network with DoG First Derivative as input feature (σ=3).

As also noted in the equivalent experiment in the previous chapter, this result is also significant as, together with the observation that the *DoG* both filters and calculates the 1st derivative in one pass, it also demonstrates that the *DoG* is a feature capable of fast classification of regions of interest on any contour curves extracted from a two dimensional image.

5.8.4.6 The Laplacian of Gaussian Second Derivative Feature

The *LoG* feature does not perform as well as the first derivative, *DoG*. In fact the *DoG* feature's accuracy and macro-F1 scores are approximately 8% better than this feature. However, it is on a par with the curvature feature in performance.

The dataset used here appears to rely more on first derivatives than second derivatives for landmark classification and segmentation. The second derivative indicates the

concavity/convexity of curvature and where it is located, whilst the first derivative encodes the degree of curvature at any particular point on the curve.

The philtrum's recall figure of 57.35% is below the rest, however the n/a recall figure is high, indicating that further combinations of features is warranted. Indeed, the following experiment shows the results of combining both first and second derivatives, since, when used together, these improve the accuracy of the segmentation process as was discovered in the previous chapter.

Profile Class	**Recall (%)** TP/(TP+FN)	**Precision (%)** TP/(TP+FP)	**F1 Score (%)**	**Support**
Columella	90.76	81.25	85.74	6112
Philtrum	57.35	61.08	59.16	3859
LowerLip	83.16	89.32	86.13	3028
n/a	94.85	100.00	97.35	2716
UpperLip	70.92	73.13	72.01	4298
Overall Accuracy (%)	**79.46**	**Macro F1 Score (%)**	**80.08**	**20013**

Table 5-11: Evaluation of LSTM network with LoG Second Derivative as input feature ($\sigma=2$).

5.8.4.7 Combined DoG and LoG Derivative Features

Using the *LoG* and *DoG* as a 2-dimensional input vector to the network produces a good result and is only slightly less accurate than the overall best feature identified in this series of experiments (see the next sub-section for details of this). It improves upon the *DoG*'s overall accuracy by 1.6% and its macro-F1 score by 1.8% (see Table 5-12).

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	94.45	93.26	93.85	6112
Philtrum	79.89	74.67	77.19	3859
LowerLip	95.74	94.71	95.22	3028
n/a	99.52	100.00	99.76	2716
UpperLip	78.87	86.26	82.40	4298
Overall Accuracy (%)	**89.18**	**Macro F1 Score (%)**	**89.69**	**20013**

Table 5-12: Evaluation of LSTM network with First and Second Derivative as input feature (σ=2).

5.8.4.8 Combined Curvature, DoG and LoG Derivative Features

The results of this feature combination are slightly better than those of the previous section's *DoG* and *LoG* feature combination and are detailed in Table 5-13 below. In the previous chapter, adding an additional feature (curvature) did not improve the results. There is additional information in a useable form that can help the classification process in this experiment. This is due to the curvature calculation improving the generalization of the classifier. Once again, this result is important to this study as curvature was originally hypothesized to be a good choice as a feature for classification.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	95.50	92.71	94.08	6112
Philtrum	78.80	76.03	77.39	3859
LowerLip	96.43	94.99	95.71	3028
n/a	99.41	99.82	99.61	2716
UpperLip	79.20	86.44	82.66	4298
Overall Accuracy (%)	**89.45**	**Macro F1 Score (%)**	**89.89**	**20013**

Table 5-13: Evaluation of LSTM network with First and Second Derivative, and curvature as input feature (σ=3).

5.8.4.9 *Curvature with either first or second derivatives*

The previous experiment combined curvature with both first and second derivatives of the contour curve. Two further investigations were carried out to evaluate the effect of curvature alone with *DoG* and then curvature alone with *LoG*. Both combinations provided poorer results, although the *DoG* in combination with the curvature feature performs well when compared with the curvature, *DoG* and *LoG* features together.

The overall accuracy and macro F1 scores for both combinations are shown in Table 5-14. These results indicate the first derivative, *DoG* feature contains significant information to guide the training and inference of the classifier.

	Overall Accuracy	**Macro-F1 Score**
Curvature & *DoG*	**88.68%**	**89.05%**
Curvature & *LoG*	**83.75%**	**84.19%**

Table 5-14: Overall Accuracy and macro-F1 score for Curvature with DoG and curvature with LoG.

5.8.5 Summary of overall accuracy and F1 scores

Table 5-15 summarizes the results of section 5.8.4 and the graph of Figure 5-15 presents a visual summary. The overall accuracy is presented along with the macro F1 score for each class. As discussed in section 2.7 and Appendix D, the macro-F1 score treats each of the multi-class F1 scores equally ignoring the support for each class and avoiding biases due to unequal sample sizes. It is found that features that include the *DoG* and, to a lesser extent, the *LoG*, either alone or in combination, result in an improvement in the classifier's overall accuracy and macro F1 score. The *DoG* appears to be the most powerful feature and the strategy of including *LoG* and curvature with *DoG* improves the overall performance of the classifier by 1.5% to 2%.

The raw curve alone has very little predictive power when used with this classifier and dataset. The relevant information is contained within it (i.e., the *DoG*, *LoG* and curvature) but it appears that with this classifier, and the relatively small dataset used, it has been

unable to extract the necessary information for learning the mappings between the inputs and the classifications.

	Feature	Accuracy	Macro-F1 score
1	Raw profile	37.53%	51.33%
2	Normalized, curvature of Gaussian filtered signal, σ=1	79.08%	78.22%
3	Normalized, curvature of Gaussian filtered signal, σ=2	74.93%	74.11%
4	Normalized, curvature of Gaussian filtered signal, σ=3	76.42%	76.47%
5	First order derivative of Gaussian filtered signal (*DoG*).	87.57%	87.89%
6	Second order derivative of Gaussian filtered signal (*LoG*).	79.46%	80.08%
7	1st + 2nd order derivatives (*DoG* and *LoG*).	89.18%	89.69%
8	Curvature, 1st, and 2nd derivatives (*DoG* and *LoG*).	89.45%	89.89%
9	Curvature and 1st derivative (*DoG*).	88.68%	89.05%
10	Curvature and 2nd derivative (*LoG*).	83.75%	84.19%

Table 5-15: Summary of Accuracy and macro-F1 scores.

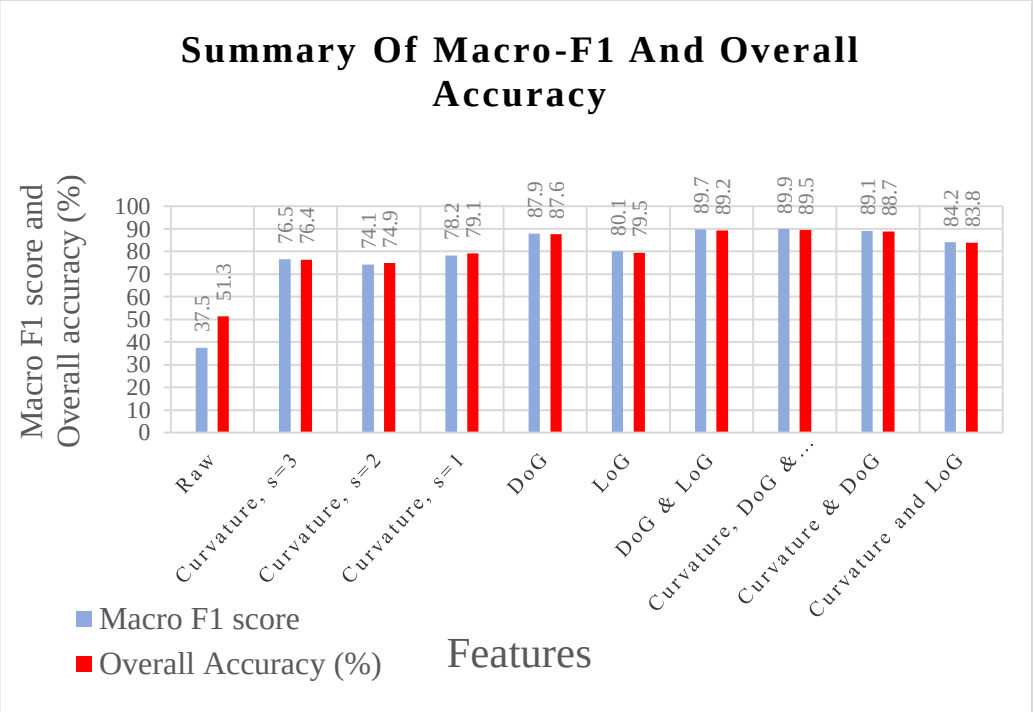


Figure 5-15: Summary of Macro-F1 and Overall Accuracy.

Two datasets have been used so far in this study, the ECG dataset of the previous chapter and the profile contour dataset generated and described in this chapter. Whilst this dataset set is very small in comparison to the ECG dataset, it has been able to generate results that are, in general, equally good. This supports the argument that a well-engineered feature set enables a classifier to learn effectively on smaller datasets. If the features can be calculated efficiently, as has been demonstrated previously, then this is an additional benefit too.

The choice of classifier would change the results as would the adjustment of the hyper-parameters and architecture of the classifier. To ensure a fair comparison between the ECG segmenter and the profile contour segmenter, no modification to the architecture of the classifier or hyper-parameters was attempted in this section. In the next section some adjustment of the hyperparameters was attempted to fine tune the classifier.

5.9 Effect of parameter adjustment on the LSTM network

5.9.1 Changes to the LSTM RNN architecture and training

Here, the effect of changes to training hyper-parameters and network architecture were investigated.

The network architecture used in the previous section was the starting point. As before, the input layer was adjusted to alter the number of inputs to take into account the two dimensional (x, y) co-ordinates of the curve and combinations of input features, and the network comprised of:

- 150 hidden units;
- a fully connected output layer with 5 outputs corresponding to,
    - n/a (Not defined),
    - lower lip,
    - upper lip,
    - philtrum,
    - columella,
- a softmax layer.

For each feature or feature combination we train and test the LSTM RNN using a 70:30 train:test dataset ratio. The dataset size was 648 images. Of these, the test set comprised of 194 images and the training set 454.

Originally the mini-batch size had been set to 45 as this was the value used in the original published network used to train the large ECG dataset. Le-Cun cites Masters and Luschi (2018), observing that better performance is obtained by reducing the mini-batch size to as low a level as possible. This observation is documented too by Wilson and Martinez (2003). A consequence of using a smaller mini-batch size in this work is a significant increase in the training time, however as the profile contour dataset is reasonably small it was possible to reduce the mini-batch size down to single figures without unduly increasing the time allotted to complete the analysis of the networks investigated in this study. A mini-batch size of 1 was selected. That is, on-line training was used. This had the greatest, positive effect on the testing dataset's overall accuracy and F1 scores.

The effect of changing the architecture of the LSTM network was investigated next. The number of LSTM units in the hidden layer was adjusted. The network performance increased slightly and peaked at approximately 500 units and began to fall off marginally after this. Consequently, the number of hidden units was set to 500. This would reduce the inference speed. As noted in section 4.4, and is also the case here, there is always a trade-off between speed and accuracy. Here we concentrate on the accuracy of the classifier.

The final adjusted architecture and hyper-parameters are shown below.

Input Layer	Adjusted as required to accommodate the dimensions of the input features.
Hidden units	500.
Minibatch size	1.
Output layers	Fully connected with 5 outputs corresponding to: n/a (Not defined), lower lip, upper lip, philtrum and columella with a final softmax layer.

Table 5-16: Adjusted LSTM RNN architecture.

5.9.2 Results and Comparisons of Modified Network Accuracy

This section also follows the experimental methods detailed in chapter 3 and in section 4.3.4. Here, the network architecture parameters remain fixed as defined in the previous section and each experiment generates a multi-class confusion matrix from which overall accuracy figure is calculated along with, for each class, its precision, recall and F1 score.

The support is also stated for each class's test data. The following section details the results obtained for each of the features and feature combinations identified Table 5-17 below. The features identified here are a subset of those used in Table 5-5. Following this an analysis and discussion of the results is provided.

One and two dimensional input feature vector		
	Feature	**Purpose**
1	Raw profile contour curve.	Establishes whether the modified network is able to learn important features.
2	Normalized, curvature of Gaussian filtered signal, σ=1.	Acts as a comparison for curvature. Does the modified network significantly improve upon the previous architecture? The best performing σ is chosen (σ=1).
3	First order derivative of Gaussian filtered signal.	Acts as a comparison. Does the modified network significantly improve upon the previous architecture?
4	Second order derivative of Gaussian filtered signal.	
Four dimensional input feature vector		
5	1^{st} + 2^{nd} order derivatives.	Acts as a comparison. Does the modified network significantly improve upon the previous architecture?
Five dimensional input feature vector		
6	Curvature, 1^{st}, and 2^{nd} derivatives.	Acts as a comparison. Does the modified network significantly improve upon the previous architecture?

Table 5-17: Proposed features used to evaluate accuracy of a modified recurrent LSTM network in segmenting face profiles.

5.9.2.1 Raw profile contour

The network trained on the raw profile data was not successful. It mis-classified to the extent that two classes contained no true positive results. As previously, the following curvature features form an initial basis for comparison in this section.

Profile Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
Columella	86.44	45.41	59.54	6112
Philtrum	0	0	0	3859
LowerLip	0	0	0	3028
n/a	9.72	96.35	17.66	2716
UpperLip	55.91	29.65	38.75	4298
Overall Accuracy (%)	**39.72**	**Macro F1 score (%)**	**19.66**	**20013**

Table 5-18: Evaluation of LSTM network with raw profile contour curve as input feature.

5.9.2.2 Normalized curvature (σ=1) feature

Curvature is a one-dimensional vector describing the curvature of the contour profile curve. It is calculated using the full curvature equation of (2-6). The overall accuracy of the classifier has improved significantly on the previous architecture with accuracy increasing from 76.42% to 87.31% and the macro F1 score increasing also by approximately 10% to 87.68%.

Profile Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
Columella	93.18	90.24	91.68	6112
Philtrum	78.47	72.72	75.48	3859
LowerLip	95.74	93.40	94.55	3028
n/a	93.15	96.20	94.65	2716
UpperLip	77.29	87.33	82.00	4298
Overall Accuracy (%)	**87.31**	**Macro F1 Score (%)**	**87.68**	**20013**

Table 5-19: Evaluation of LSTM network using normalized curvature with σ=1 as input feature.

5.9.2.3 First Derivative of Gaussian Feature

Table 5-20 shows the modified architecture achieved an improvement in accuracy of 2.5% on the previous architecture. The macro-F1 score also increased from 87.89% to 90.57%. These improvements are not as impressive as the curvature feature's above but nevertheless 2.5% is significant given the starting point is already quite high at 87%. As also noted in the equivalent experiment in the previous chapter, this result is also significant as, together with the observation that the *DoG* both filters and calculates the 1st derivative in one pass, it also demonstrates that this feature is capable of fast classification of regions on any contour curves extracted from a two-dimensional image.

Profile Class	**Recall (%)** TP/(TP+FN)	**Precision (%)** TP/(TP+FP)	**F1 Score (%)**	**Support**
Columella	95.86	94.58	95.21	6112
Philtrum	80.59	76.53	78.51	3859
LowerLip	96.07	96.04	96.05	3028
n/a	99.96	99.85	99.91	2716
UpperLip	80.34	86.20	83.16	4298
Overall Accuracy (%)	**90.17**	**Macro F1 Score (%)**	**90.57**	**20013**

Table 5-20: Evaluation of LSTM network with DoG First Derivative as input feature (σ=2).

5.9.2.4 The Laplacian of Gaussian Second Derivative Feature

Here the *LoG* performs significantly better than previously with an improvement of around 10% on both the overall accuracy and marcro-F1 scores as shown in Table 5-21. The *LoG* feature still does not perform as well as the first derivative, *DoG*. The following shows the results of combining both first and second derivatives.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	94.78	93.38	94.07	6112
Philtrum	78.34	75.03	76.65	3859
LowerLip	96.07	94.76	95.41	3028
n/a	99.71	99.93	99.82	2716
UpperLip	79.60	85.53	82.45	4298
Overall Accuracy (%)	**89.21**	**Macro F1 Score (%)**	**89.68**	**20013**

Table 5-21: Evaluation of LSTM network with LoG Second Derivative as input feature (σ=2).

5.9.2.5 Combined DoG and LoG Derivative Features

The adjusted network again improves on the previous architecture's results with an overall accuracy of 90.83% and macro F1 score of 91.26%, an improvement of about 1.5%. This combination of features, architecture, and hyper-parameter selection results in the overall best performance as a segmenter and so this model was used to analyze the capability of the segmenter as a landmark regressor. This process and analysis is detailed in section 5.10.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	96.01	95.17	95.59	6112
Philtrum	81.76	77.35	79.49	3859
LowerLip	96.30	97.79	97.04	3028
n/a	99.93	100.00	99.96	2716
UpperLip	82.01	86.57	84.23	4298
Overall Accuracy (%)	**90.83**	**Macro F1 Score (%)**	**91.26**	**20013**

Table 5-22: Evaluation of LSTM network with First and Second Derivative as input feature (σ=2).

5.9.2.6 Combined Curvature, DoG and LoG Derivative Features

The results of this feature combination are slightly worse than those of the previous section's *DoG* and *LoG* feature combination, although there is very little difference.

Profile Class	Recall (%) TP/(TP+FN)	Precision (%) TP/(TP+FP)	F1 Score (%)	Support
Columella	96.70	94.00	95.33	6112
Philtrum	81.21	77.10	79.10	3859
LowerLip	96.07	98.24	97.14	3028
n/a	100.00	100.00	100.00	2716
UpperLip	80.83	87.20	83.89	4298
Overall Accuracy (%)	**90.66**	**Macro F1 Score (%)**	**91.09**	**20013**

Table 5-23: Evaluation of LSTM network with First and Second Derivative, and curvature as input feature (σ=3).

5.9.3 Summary of overall accuracy and F1 scores

Table 5-24 summarizes the results of section 5.9.2 and the plot in Figure 5-16 presents a visual summary of these results. The overall accuracy is presented with the macro F1 score for each class investigated in this section.

	Feature	Accuracy	Macro-F1 score
1	Raw profile	39.72%	19.66%
2	Normalized, curvature of Gaussian filtered signal, σ=1	87.89%	87.68%
3	First order derivative of Gaussian filtered signal (*DoG*).	90.17%	90.57%
4	Second order derivative of Gaussian filtered signal (*LoG*).	89.21%	89.68%
5	1^{st} + 2^{nd} order derivatives (*DoG* and *LoG*).	90.83%	91.26%
6	Curvature, 1^{st}, and 2^{nd} derivatives (*DoG* and *LoG*).	90.66%	91.09%

Table 5-24: Summary of overall accuracy and macro-F1score.

It is found that features that include the *DoG* and, to a lesser extent, the *LoG*, either alone or in combination, result in an improvement in the classifier's overall accuracy and macro F1 score. This mirrors the finding of section 5.8.4. The *DoG* appears to be the most powerful feature, however, the strategy of including *LoG* and curvature with *DoG* does not improve the overall performance of the classifier. The overall best classifier, albeit marginally, is the adjusted network with *DoG* and *LoG* feature inputs with the *DoG*, *LoG* and curvature a very close second. As before, the raw curve alone has no useful predictive power when used with this classifier and dataset.

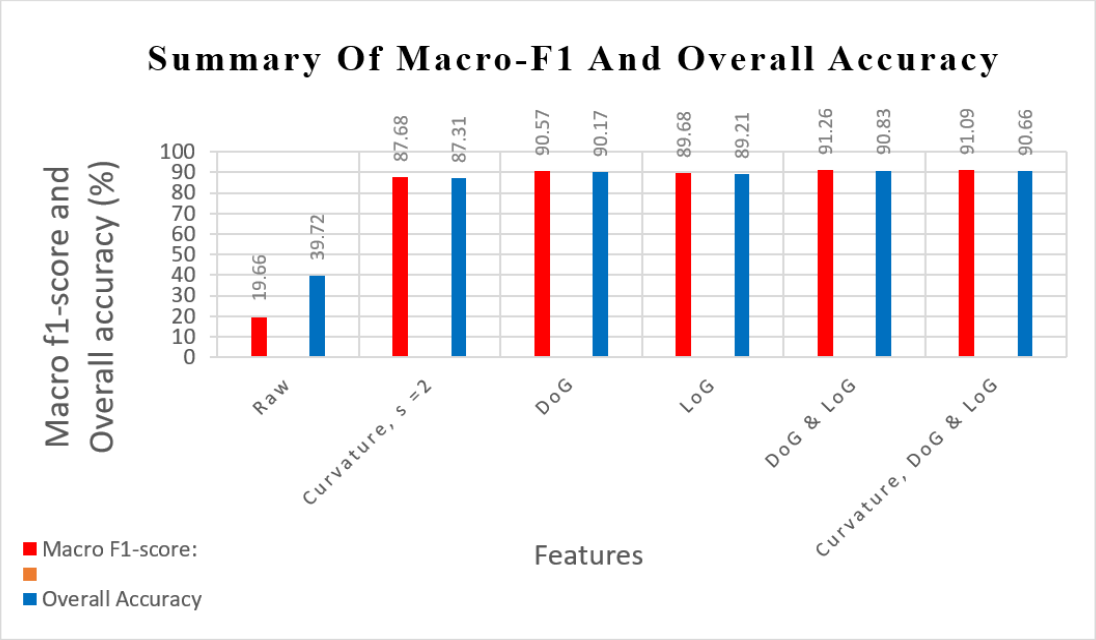


Figure 5-16: Summary of Macro-F1 and overall accuracy for adjusted network.

Finally, an overall comparison between the performance of the two networks is summarized in the plot shown in Figure 5-17.

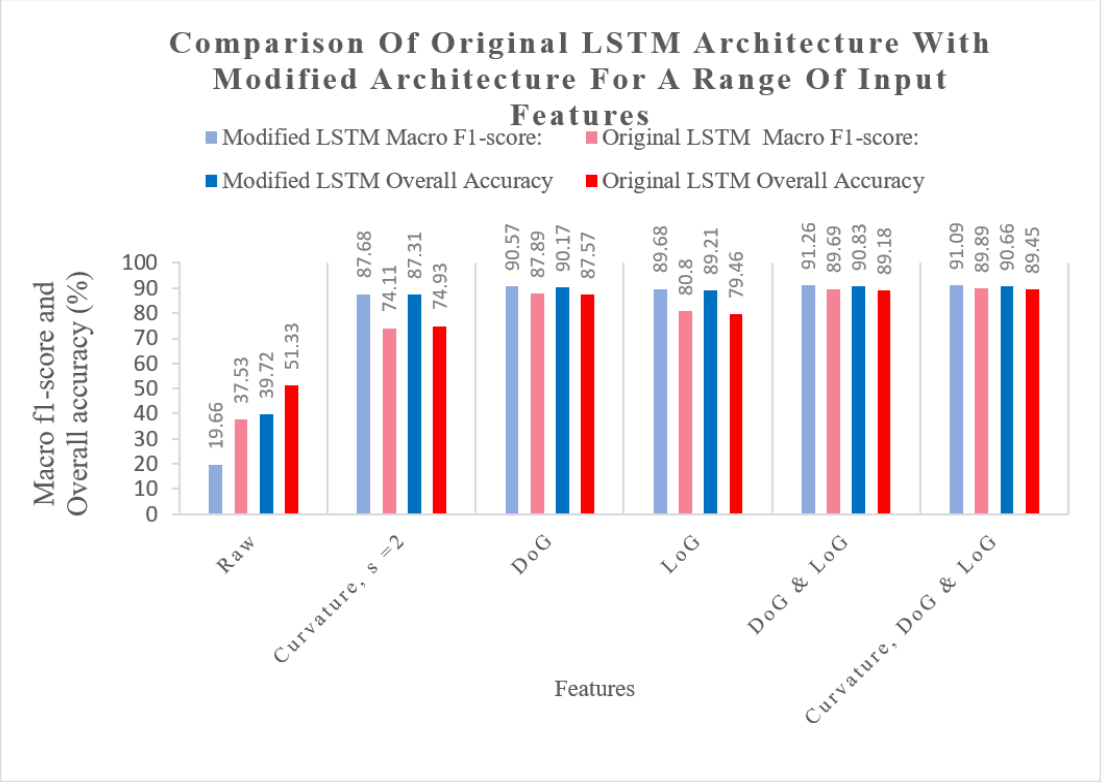


Figure 5-17: Comparison of the original LSTM network and the modified LSTM network.

This figure provides a visual comparison of the original and modified LSTM networks' macro-F1 and overall accuracy scores. In both networks use of the *DoG* and *LoG* features improve accuracy and F1 scores dramatically compared to the raw dataset alone. Curvature as a feature is interesting. The modified LSTM network with curvature as an input feature clearly performs significantly better than the original network did with the same curvature input feature. The reason is not clear but could be related to the increased complexity of the network. Overall, the best classifiers benefit from the additional changes, but at what cost? Training took approximately 8 to 10 times longer and the inference time of the modified network will be increased due to the result of increasing the complexity of the architecture's hidden layer to 500 from the original 150 units. For an increase in performance of 1 to 1.5%, its worth will depend upon the context of the problem space.

Considering the class F1 scores of the best performing network (the modified LSTM with combined *LoG* and *DoG* features) individually, the n/a region classes perform best. This can be attributed to the design of the experiment. The contours were "bookended" on

both sides by a fixed number of n/a labels and the result was the network was able to easily learn this simple rule. A better approach would be to randomize the number of n/a labels or use the untrimmed contour. This would require a significantly larger dataset with which to train the network.

Having a fixed amount of "n/a" points also impacted on the adjacent regions. The columella, for example had few incorrectly predicted points where it joins the n/a region. Similarly, the lower lip benefited from this too. Nevertheless, the philtrum and upper lip segmentation performed well with only the precision of the classifier falling below 80% for the philtrum. What effect does this have on the segmenter's ability to regress landmarks? This is investigated next.

5.10 From segmentation to regression

Accurate profile estimation is an aim of this study. So far, the face profile has been segmented, and this can be used to describe the various regions of the face and hence the posture of the head with respect to the camera's reference frame, or with respect to a common axis of rotation, eg the tragus if available, or to see a change in posture between images in a video for example. However, the precision and accuracy of the transition between regions of interest has not been analysed yet. The question, "how well does the model locate landmarks on the contour?" has yet to be answered and so this section attempts this.

5.10.1 Locating region transitions

As there is an ordinal relationship between landmarks, the segmenter quickly learned to differentiate between regions and, after manual review, the prediction errors were seen to occur at the transition between regions.

To identify the landmarks delimiting the regions of the segmented contours predicted during the testing of the segmenter, each contour was programmatically analysed and the transition between regions noted. This was achieved by searching for transitions and marking the contour position at which the transitions occurred. References to the original file image were included to examine outliers later during analysis of the results.

5.10.2 Evaluation of predicted landmark accuracy

The error was evaluated by determining the difference between the predicted value and the adjusted expert, ground truth value for all landmarks used by the segmenter. This was repeated for all image profile contours in the dataset. Both the ME and MAE were calculated. These errors were calculated on results generated using the segmenter trained with the first and second derivatives as input features.

MAE was used instead of the mean squared error (MSE) since MSE exaggerates the importance of outliers. The majority of outliers examined included badly framed profiles; badly positioned head posture, for example looking away or toward the camera and not at 90 degrees to it; and occlusion. These issues should have been resolved at the point of image capture so the test dataset should ideally have such anomalies removed – in fact, arguably, they should not be present in the first place. It should also be noted that during the training process imperfect images as described above were purposefully left in the dataset to aid the model's generalisation properties. One or two images had high reflectance which interfered with the contour capturing process. Where an outlying RGB 2D image showed no obvious problems that should have been corrected at the time of image capture, it was not removed from the test dataset. Additionally, the precision of the measurements was used as a second metric to describe the quality of the model. The standard deviation of the errors in pixels was used to measure precision. Table 5-25 summarizes the accuracy and precision of the landmark position derived from the segmenter.

Landmark Label	Mean Absolute Error	Mean Error	Precision (as described by standard deviation)
Labiale inferius	0	0	0
Stomion	0.542	0.045	2.333
Labiale superius	5.277	0.791	6.898
Subnasale	1.548	0	3.587
Pronasale	0.011	-0.011	0.0106

Table 5-25: Summary of Errors and precision for best segmentation model (measurements in pixels).

Histograms of the measurement error for the stomion, labiale superius and subnasale are provided below in Figure 5-18, Figure 5-19 and Figure 5-20.

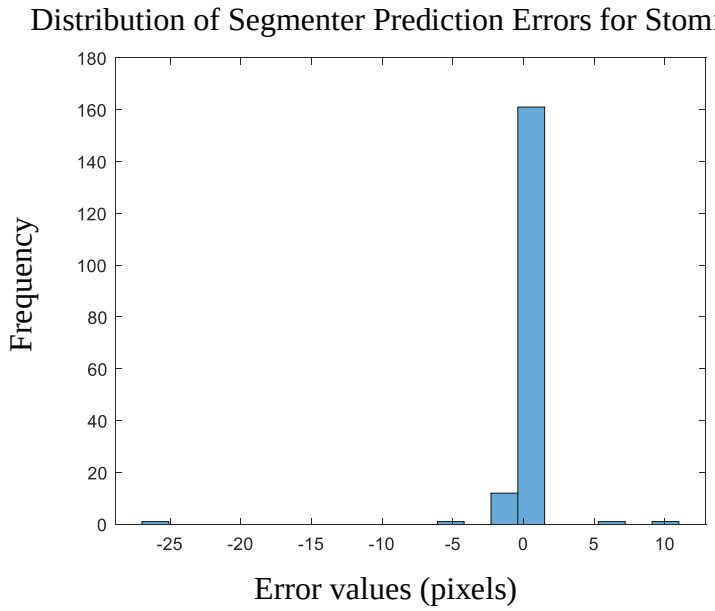


Figure 5-18: Histogram showing distribution of errors in landmark prediction against expert annotator for the stomion landmark.

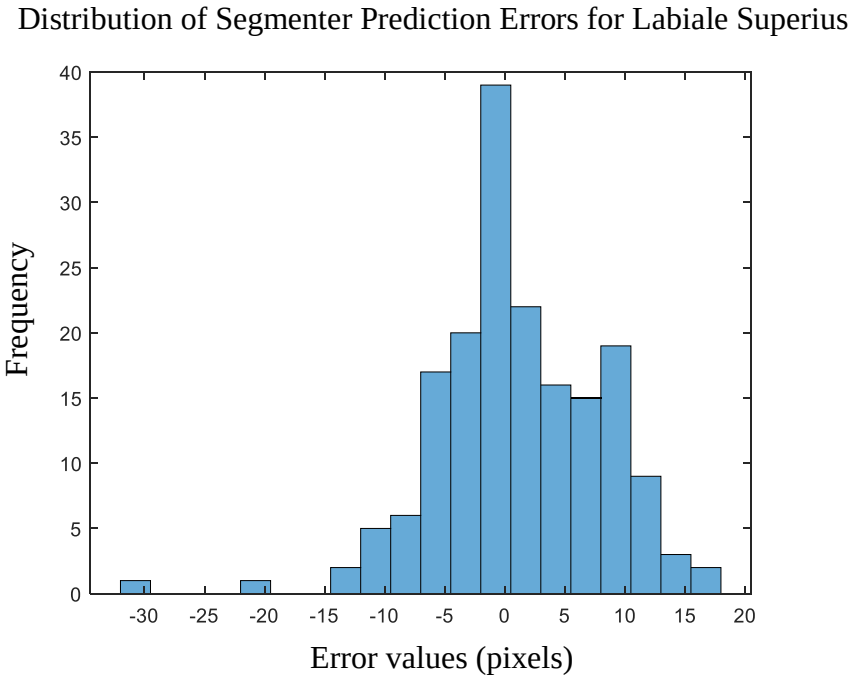


Figure 5-19: Histogram showing distribution of errors in landmark prediction against expert annotator for the labiale superius landmark.

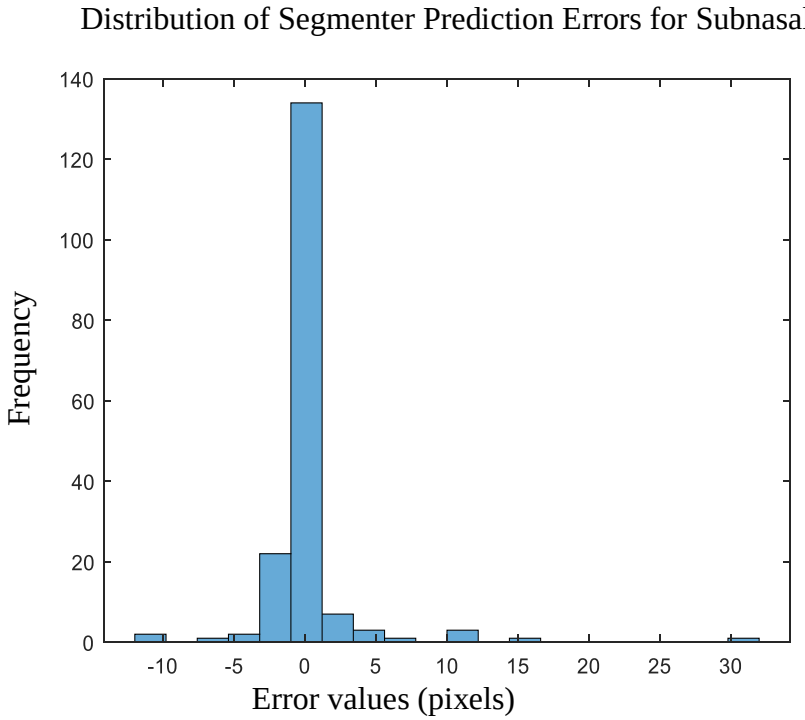


Figure 5-20: Histogram showing distribution of errors in landmark prediction against expert annotator for the subnasale landmark.

The pronasale and labiale inferius are located with high accuracy and precision and so are not plotted here. This is due to the fixed amount of n/a labels surrounding the contour. Nevertheless, the stomion is located with sub-pixel resolution (bear in mind the image is 480 pixels in height and 640 pixels in width), and the subnasale has an MAE of less than 2 pixels and shows good precision.

The classifier has a lower accuracy and precision when predicting the location of the labiale superius. This may be related to this issue described in an earlier section discussing the automatic adjustment of the labiale superius landmark in section 5.5. The algorithm uses the highest point of curvature at the vermillion of the upper lip, and this varies depending upon the anatomy of the individual. This assumption will need further investigation. Additionally, the training and testing datasets do include some individuals wearing beards and moustaches. This could have affected the training and also the inference accuracy of the classifier.

Based upon this discussion it appears that the subnasale and stomion are good candidate landmarks to use in estimating head profile posture. Of these two landmarks, the subnasale has the advantage that it moves very little during changes of expression when compared with the stomion. However, if a neutral expression is posed then either or both can be used.

5.11 LSTM Classification Runtime Results and Comparisons

This section investigates the inference speed of both chosen network designs using the input features identified in the previous sections. The timing procedure described in the methodology chapter, section 3.8 was used here. As before, all variables were cleared before timing took place and Matlab's parallel pool, which makes use of the GPU, was not used.

Four tests were carried out for each feature input combination:

1. Original network with 150 hidden units and minibatch hyper-parameter set to 45.
2. Original network with 150 hidden units and minibatch hyper-parameter set to 1.
3. Adjusted network with 500 hidden units and minibatch hyper-parameter set to 45.
4. Adjusted network with 500 hidden units and minibatch hyper-parameter set to 1.

Each test used all 194 test data contour samples.

Table 5-26, overleaf, shows the results for each test. The rows represent the hyper-parameters and the columns the number of hidden units used. Times are in milliseconds and give the time to classify all images in the test dataset.

A further test using just the best performing combination of input features was undertaken. This involved classifying just one sample image's contour. The aim here was to assess the time required to initialize the classifier using the chosen network before inference of the single contour.

		Hidden units	
		150	500
Mini batches	1	375.4	581.1
	45	30.6	44.8

DoG and *LoG*

(4 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	376.3	596.1
	45	29.3	46.2

DoG

(2 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	373.8	576.2
	45	29.5	44.9

Raw

(2 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	360.9	577.4
	45	30.9	44.0

Curvature

(1 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	378.3	595.8
	45	30.31	45.3

LoG

(2 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	374.3	590.5
	45	29.3	45.4

Curvature and DoG

(3 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	358.0	577.2
	45	30.8	45.0

Curvature, DoG & *LoG*

(5 dimensional vector)

		Hidden units	
		150	500
Mini batches	1	375.8	578.1
	45	29.5	46.2

Curvature and *LoG*

(3 dimensional vector)

Table 5-26: Inference times for the original and modified LSTM network. All times are in milliseconds.

The results show that the timings were very similar across each feature combination used and so independent of the number of features used as inputs to the classifiers. With regard to batch classification, decreasing the mini-batch hyperparameter to 1 causes an approximate 12-fold increase in the classification time since the LSTM classifier needs to know the min-batch size used for training. Changing the number of hidden units from 150 to 500 causes an increase in inference time by a factor of approximately 1.5.

For the best performing modified network (the *LoG* and *DoG* combined using 500 hidden units and with the mini-batch hyperparameter set to 1) as identified in the previous section, the inference time for *one* sample contour based on the batch classification of the whole dataset was 3ms on average, including the one-off initialization of the network. Run-time speed vs classification accuracy is always a tradeoff as discussed in section 4.4, but sacrificing 1-2% of accuracy in the network by selecting the original, unmodified LSTM network with 150 hidden units and a minibatch size of 45 yields an improved inference time of 150μs per sample.

Focusing on the modified network, the time required to classify one contour alone, including the initialization of the classifier, increased to approximately 5.8ms and for the original network it was still 3.7ms. This is assumed to be due to the overhead in initializing the classifier in order to classify one sample. In a well-designed piece of software this point will always need to be addressed in order to ensure fast inference if images are processed online. This could be achieved by initializing the classifier just once and reusing it whenever a new image is presented for processing.

5.12 Discussions and Conclusions

This section summarizes the findings of this chapter, compares the results and draws conclusions from these.

The experiments undertaken here have all relied upon the same dataset, in this case the Notre Dame J2 dataset. The Notre Dame dataset had some attributes that are advantageous. It contains 2.5D images (RGB images with additional depth information), and it is the nature of this additional depth data that has been useful in this study. In particular the depth information has not been pre-processed and so any flaws due to reflections and attenuations have been kept. This ensured that any algorithms or ML based models would be forced to deal with flawed depth information.

Several relevant observations were gathered from this:

1. Hair can cause unwanted attenuation and reflections causing errors in depth measurements. Lighting of some subjects caused skin reflections that also affected depth measurements.

2. Consequently, there were no complete head profile depth images. Only a segment of a profile was useable. This limited the useful depth information to the face region, between the sellion at the top of the nose down to the gnathion at the chin.
3. The useable dataset contained flawed images, for example missing subjects, occluded faces, badly framed subjects, etc.
4. Many images that had been labelled manually could not be used as the dataset provided had errors in file names that linked 2D RGB images to their respective depth information files.
5. As a consequence of points 3 and 4 above, the original portion of the dataset used was reduced from 985, then 693 and after discovering the issue enumerated in point 4, then down to just 648 images. Of these, the test set comprised of 194 images and the training set 454.
6. On a few images, subject head movement caused a mismatch in the captured depth profile information and the RGB 2D image.
7. The automatic landmark adjustment process was able to solve the problem associated with point 6.
8. The automatic adjustment process also improved (based on visual investigation) the accuracy of the expert manual annotation process.
9. Despite the flawed nature of the depth information data, the segmentation and regression models, together with the adjustment algorithms, showed good results. The modified model using *DoG* and *LoG* as input features was used to regress landmarks. It had an overall accuracy and macro-F1 score of around 91%.

Interestingly, the raw data did not make a good feature for either LSTM network. Both networks were unable to extract useful information directly from the raw input. However, use of the *LoG* and *DoG* operators together with curvature significantly improved the performance of both the original LSTM network and the modified version. The modified network worked best when the *LoG* and *DoG* operations, with their inherent smoothing, were used as feature inputs.

Putting aside the networks that used just the raw signal feature, and reviewing the inter-class precision and recall for both the original and modified LSTM network, it is clear that there was a consistency in the way all network variations performed. The n/a class was classified very well (often 100% recall and precision). This was due to the way the

contours were pre-processed - adding a fixed amount of n/a labelled points to both ends of the profile contour curve. The classes adjacent to the n/a signals, the columella and the lower lip, benefited from this approach to a certain extent, and still performed well with F1 scores in the mid to high 90's. The philtrum and upper lip regions were nevertheless segmented well with F1 scores of 79% and 84% respectively.

The curvature signal alone performed better in the modified network with an increase in performance of 12% when compared with the original network. This was due, perhaps, to the additional complexity of this network. As was the case with the ECG signal, networks using curvature alone as a feature were out-performed by networks using the first and second derivative features either alone or together. This is, again, hypothesized to be due to loss of relevant information when both first and second derivatives are combined into a single scalar value.

Regression of landmark locations produced good results. The accuracy and precision of these results, as measured using MAE and standard deviation, also reflected the network segmentation results. The labiale inferius and pronasale landmarks had near perfect accuracy and precision due to their positioning at the extremities of the contour curve. The stomion accuracy was at a sub-pixel resolution with high precision. This was probably due to its very recognizable curvature shape. Regression of the subnasale performed well too, however regression of the labiale superius performed worst with a wider standard deviation and larger MAE caused, it is hypothesized, by two competing local minima of curvature. These minima are related to an individual's anatomy and the adjustment algorithm would need further modifications, perhaps, to improve this.

The inference times investigated in the previous section show that there is a trade off between accuracy and speed, but a reduction in accuracy by 1-2% yields an order of magnitude improvement in runtime performance. Classifier initialization is important and should be optimized where possible.

In summary, the derivatives of the contour profile are good input features capable of accurately segmenting these regions when used with the LSTM classifiers developed here. They are also fast to calculate. Using both the first and second derivatives as input features together in an LSTM RNN segmenter-classifier produces the best classification

accuracy. *DoG*, *LoG* and curvature based features are still worth considering depending upon a project's requirements.

Curvature alone was hypothesized in this study to be a good feature to use for profile segmentation, and whilst the experimental results indicate it has capability in this regard, it is still nevertheless outperformed by segmenters using the *LoG* and *DoG* features.

The best performing network used *LoG* and *DoG* features, had 500 hidden units and a mini-batch size of one. Minimizing the mini-batch hyper-parameter improved network classification performance by two percentage points but at the expense of an increased training time.

In the next chapter we consider end-to-end ML approaches to segmenting and regressing contours using a CNN. A 1DTCNN is developed and its performance at segmentation and regression is investigated and its performance with respect to the recurrent LSTM networks used previously is analyzed.

6 Segmenting face profile contours with 1DTCNN Networks

In this chapter the suitability of a 1DTCNN network is investigated to solve the contour profile segmentation and regression problem studied in previous chapters. This kind of network was selected for a number of reasons:

1. It uses convolutional dilation layers to provide feature recognition at multiple scales.
2. It can deal with sequences of varying length.
3. It can output sequences of equal length to the input sequence and,
4. It is an example of a more complex neural network that can act as a suitable choice to explore the power of an end-to-end learning approach and contrast it with the LSTM network, developed in chapter 5, that was used to segment and regress the profile contours dataset.

Here, the effectiveness of a 1DTCNN network on segmenting and regressing the head profile dataset is explored and its accuracy and runtime efficiency is compared with the LSTM RNNs used in previous chapters. In particular the 1DTCNN network is first trained with the raw feature set (i.e. the contour curve co-ordinates) and subsequently with the curvature feature and finally with the best-performing engineered feature combination identified in the previous chapter. The segmentation and regression ability of these networks is assessed, and the resources required to train this network and the time to classify profiles is compared with the equivalent models developed in chapter 5. Conclusions are drawn from these results.

6.1 1DTCNN architecture and training

The 1DTCNN architecture is based upon that described in Bai, which in turn follows the work of Oord *et al.* (2016) and it is implemented using Matlab. The network is made up of three cascaded residual blocks using the ideas outlined in section 2.8 and Appendix C. The filter size, k=3, and the dilation factor, d at each level is increased exponentially, that is, at level 1, $d=1$, at level 2, $d=2$ and at level 3 $d=4$. Scaling could also be adjusted by altering the size of the convolution filter, k, but it is left constant here.

The network is trained using mini-batches of size one for 15 epochs. Training of the network stops after 15 epochs since, for each feature or combination of features used, the loss has leveled off. In order to make the comparisons between networks as balanced as possible, the values chosen for these hyper-parameters also match those used by the LSTM network investigated in chapter 5. This network randomizes the order of the samples during the training process so 3 iterations were carried out and the median selected. During the testing period there were no outlying results, that is, no model generated performed exceptionally better or worse than those in its group.

For each feature or feature combination we train and test the 1DTCNN using a 70:30 train:test dataset ratio. As previously, the test set comprises of 194 images and the training set 454 images.

All experiments were performed on machine with an Intel core i7-7700 CPU with 32GB RAM and an Nvidia 1080Ti GPU.

6.2 Results and Comparisons of Network Accuracy

As before, this section follows the experimental methods used in section 4.3.4 but focuses on three features, the raw curvature data co-ordinates, curvature and the best performing feature combination of chapter 5, that is the first and second derivatives of the curve. The network architecture parameters defined in the previous section remain fixed for both these experiments. In these experiments, as before, a multi-class confusion matrix is generated and from this an overall accuracy figure is calculated along with, for each class, its precision, recall and F1 score. The support is also stated for each class' test data.

A change to the architecture's input layer is required for each feature investigated as the input feature vector changes dimension for each experiment. The raw curve co-ordinates are 2-dimensional, representing the pixel x and y co-ordinates; the first and second derivative combination feature is a 4-dimensional vector comprising of the first derivatives of the x and y co-ordinates with respect to the arc length, and the second derivatives of the x and y co-ordinates, again, with respect to the arc length. Curvature is

simply a one-dimensional vector. The following section details the results obtained for each of these feature sets.

6.2.1 Raw profile curvature

Table 6-1 shows results obtained when using the raw profile co-ordinates alone. The network achieves a high overall accuracy (91.07%) and macro-F1 score (91.41%) when classifying the regions of interest during the segmentation process and contrast starkly with the corresponding results for the raw feature used with the LSTM network of Chapter 5. The LSTM network had an overall accuracy of 51.33% and an overall macro-F1 score of 37.53%. This is due to the higher complexity of the 1DTCNN network giving it the ability to learn the requisite features as discussed in section 2.8. This comes at the expense of a lower classification speed and a longer training time.

Profile Class	**Recall (%)**	**Precision (%)**	**F1 Score (%)**	**Support**
	TP/(TP+FN)	**TP/(TP+FP)**		
Columella	97.17	95.21	96.18	6112
Philtrum	81.47	78.90	80.16	3859
LowerLip	96.70	96.83	96.76	3028
n/a	99.71	99.71	99.71	2716
UpperLip	81.92	86.94	84.36	4298
Overall Accuracy (%)	91.14	**Macro F1 Score (%)**	91.43	20013

Table 6-1: Evaluation of 1DTCNN with raw profile contour curve as input feature.

6.2.2 Combined *DoG* and *LoG* Derivative Features

Table 6-2 summarizes the results when using the first and second derivatives of the profile contour curve. The network achieves a slightly better overall accuracy (91.58%) and macro-F1 score(91.9%) when compared with the raw feature used with the 1DTCNN network. As indicated already, this ability to learn the features required for good segmentation comes at a cost. Inference is slower, as is training. This is covered in section 6.3.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	97.57	95.78	96.66	6112
Philtrum	83.31	79.86	81.55	3859
LowerLip	96.20	98.45	97.31	3028
n/a	100	100	100	2716
UpperLip	81.92	86.17	83.99	4298
Overall Accuracy (%)	91.58	**Macro F1 Score (%)**	91.90	20013

Table 6-2: Evaluation of 1DTCNN with First (DoG) and Second (LoG) Derivative, and curvature as input feature (σ=3).

6.2.3 Normalized curvature (σ=1)

Table 6-3 summarizes the results when using the curvature of the profile contour curve as an input feature with σ=1. The network achieves a slightly poorer overall accuracy (90.53%) and macro-F1 score (90.80%) when compared with the raw feature used with the 1DTCNN. The recall figure for the Philtrum is 76.5% which is 5% lower than that achieved with the raw dataset and 7% lower than that achieved with the combined *DoG* and *LoG* features. The curvature feature is a one-dimensional sequence compared with the other features. With this is in mind, it represents a surprisingly good result.

Profile Class	Recall (%)	Precision (%)	F1 Score (%)	Support
	TP/(TP+FN)	TP/(TP+FP)		
Columella	96.96	93.56	95.23	6112.00
Philtrum	76.50	80.79	78.58	3859.00
LowerLip	97.09	97.13	97.11	3028.00
n/a	98.12	99.07	98.59	2716.00
Upperlip	84.57	84.38	84.48	4298.00
Overall Accuracy (%)	90.53	**Macro F1 Score (%)**	90.80	20013.00

Table 6-3 Evaluation of 1DTCNN with normalized curvature as input feature (σ=1).

6.2.4 Summary of overall accuracy and F1 scores

Table 6-4 summarizes the results of section 6.2 and Figure 6-1 provides a visualization of the overall accuracy and macro-F1 scores for the 1DTCNN.

	Feature	Accuracy	Macro-F1 score
1	Raw profile	91.14%	91.43%
2	1^{st} + 2^{nd} order derivatives (*DoG* and *LoG*)	91.58%	91.90%
3	Normalized, curvature of Gaussian filtered signal, σ=1	90.53%	90.80%

Table 6-4: Summary of accuracy and macro-F1 scores.

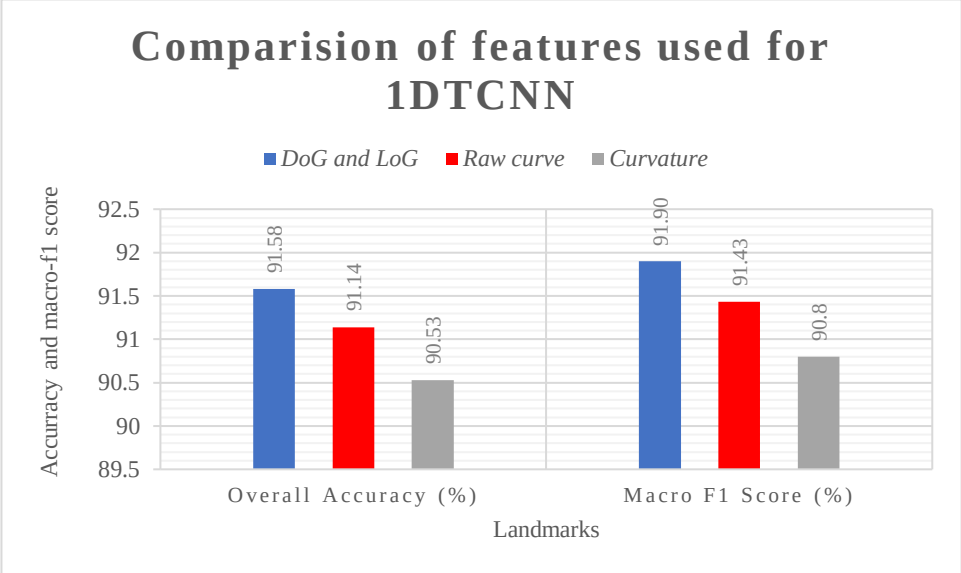


Figure 6-1: Comparison of Macro-F1 and overall accuracy for 1DTCNN.

It's clear from this comparison that the features used as input to the 1DTCNN network compare well with a range of about 1.5 percent. This indicates the network is good at learning the underlying information necessary for good classification and, when compared with the use of raw features as input, it demonstrates well the end-to-end learning capability of the 1DTCNN network, however the *DoG* and *LoG* features do confer an advantage. Curvature as a feature performs least well, however it is interesting to see that this feature is a simple scalar value, and the 1DTCNN network is capable of extracting a lot of useful information from this simple feature.

6.3 Evaluation of runtime results and training times

Here, the run-time inference speed of the 1DTCNN is evaluated together with the training times. The measurements are made for networks trained on the raw profile contour co-ordinates, curvature and with the combined first and second derivative (*DoG* and *LoG*) feature. As used with the best performing LSTM RNN, the 1DTCNN network mini-batch hyper-parameter was set to a size of 1 and the number of epochs was set to 15.

Training of this network took significantly longer that the LSTN DNN investigated in previous chapters. For the LSTN DNN network with a mini-batch size of 1 and trained for 15 epochs, training times were 2.5 minutes. For the 1DTCNN network training times took approximately 12 minutes.

The inference time for the 1DTCNN network was an order of magnitude slower that the LSTM DNN networks used previously. For the network trained with the raw profile curve co-ordinates, the median 1DTCNN network inference time was 4.32 seconds for all 194 profile contours, including a one-off median classifier initialisation period of 17ms. Inference times using the curvature feature were similar with results of 4.66 seconds for all 194 contours.

Similar results were achieved when using the DoG and LoG features as inputs. Here the median inference time of the classifier was 4.85 seconds with an initialisation time of 33ms. The timings were conducted without MATLAB's parallel pool, GPU support.

6.4 From Segmentation to Regression

The same method as that of section 5.10 was used to regress landmark positions on the contour curve, that is, the arc length was traversed to find a transition point between regions and this point then represented the corresponding landmark that defines the transition. For example, the stomion delineated the labiale inferius and labiale superius regions. As before the accuracy and precision were measured by calculating the MAE and the standard deviation. The ME was also included. The error is defined here as the difference between the ground truth test value and the estimated value determined at inference.

6.4.1 Evaluation of predicted landmark accuracy

Results are presented for a network trained with the raw profile curve co-ordinates, with curvature and then the combined first and second derivative features (*DoG* and *LoG)*.

6.4.1.1 Raw profile contour

Histograms are included here to visualise the distribution error. Table 6-5 summarizes the results and Figure 6-2 to Figure 6-4 below and overleaf, show the distributions. Note where there is little or no variance in the measurements, then the histogram is not included.

Landmark Label	**Mean Absolute Error (pixels)**	**Mean Error (pixels)**	**Precision (as described by standard deviation) (pixels)**
Labiale inferius	0	0	0
Stomion	0.52	0.37	3.61
Labiale superius	5.09	0.19	8.64
Subnasale	1.74	0.34	6.77
Pronasale	0.045	-0.02	0.58

Table 6-5: Summary of Errors and precision for 1DTCNN, Raw contour input feature (measurements in pixels).

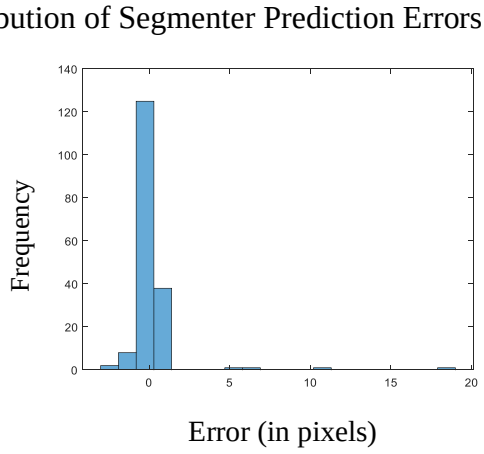


Figure 6-2: Distribution of errors in 1DTCNN landmark prediction for stomion landmark.

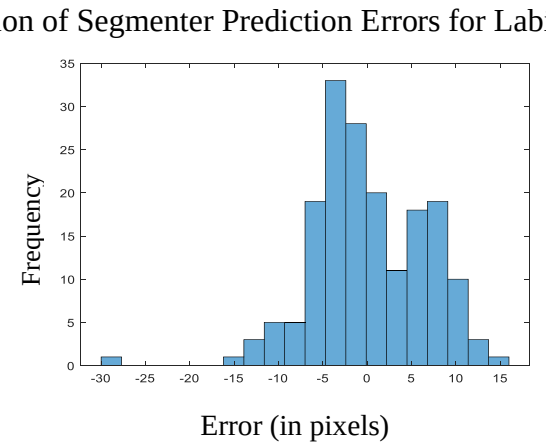


Figure 6-3: Distribution of errors in 1DTCNN landmark prediction for labiale superius landmark.

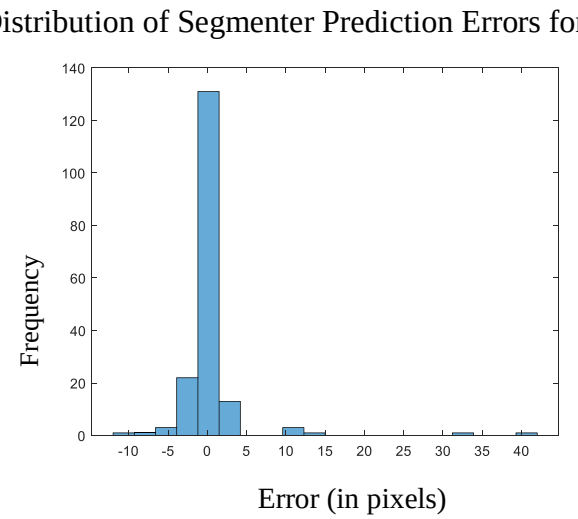


Figure 6-4: Distribution of errors in 1DTCNN landmark prediction for subnasale landmark.

The labiale superius landmark shows a lower precision and accuracy than the other landmarks. This was also the case when using the best performing LSTM RNN of Chapter 5 which showed a similar performance in regressing this landmark. The histogram shows a significant number of measurements offset from the ground truth values. This, as before, is conjectured to be a result of the adjustment algorithm and the nature of the anatomy variances amongst individuals, or due to the lower support of the region classes on either side of this landmark (upper lip and philtrum). The stomion and subnasale, on the other hand, show higher accuracy and precision.

6.4.1.2 Normalized Curvature feature (σ=1)

Histograms are included here also, and Table 6-6 summarizes these results. Figure 6-5 to Figure 6-7 show the distributions. Note where there is little or no variance in the measurements, then the histogram is not included. These results are similar to those of the raw profile contour input feature results, but they do have an improved precision for the labiale superius and subnasale.

Landmark Label	Mean Absolute Error (pixels)	Mean Error (pixels)	Precision (as described by standard deviation) (pixels)
Labiale inferius	0	0	0
Stomion	0.32	0.08	1.02
Labiale superius	4.8	0.10	6.28
Subnasale	1.79	0.84	6.00
Pronasale	0.35	-0.16	0.60

Table 6-6: Summary of Errors and precision for 1DTCNN, curvature feature (σ=1), (measurements in pixels).

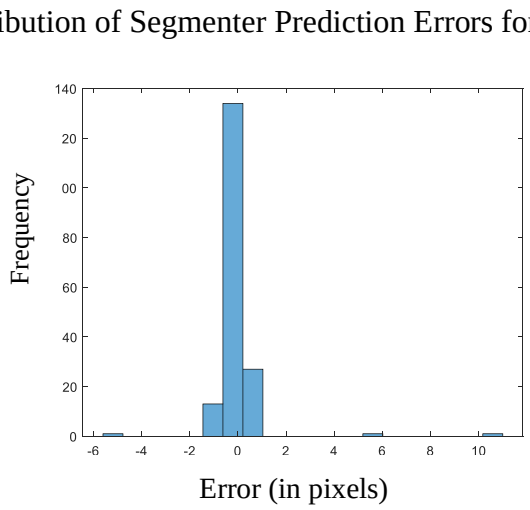


Figure 6-5: Distribution of errors in 1DTCNN landmark prediction for stomion landmark.

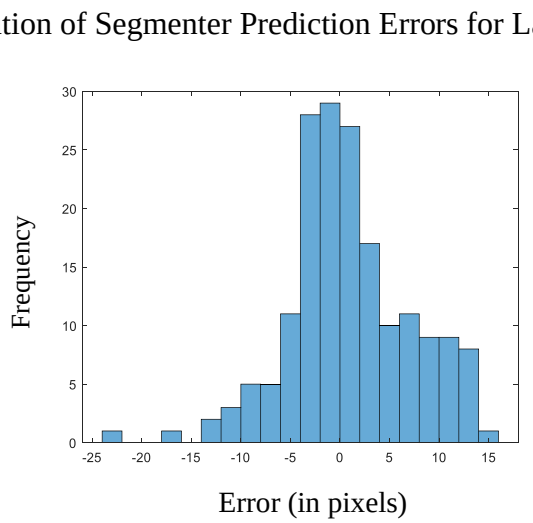


Figure 6-6: Distribution of errors in 1DTCNN landmark prediction for labiale superius.

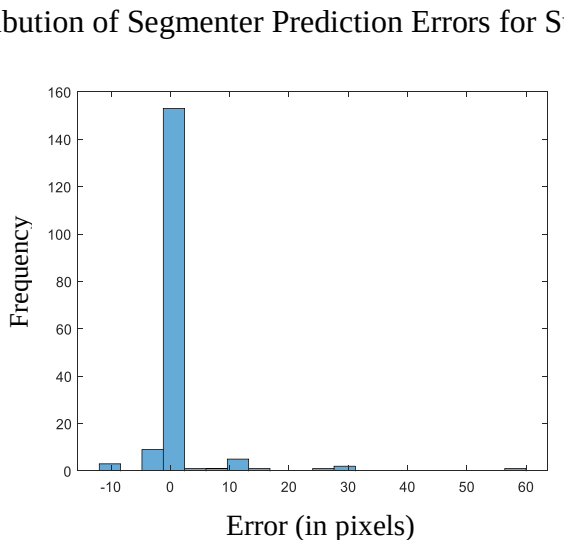


Figure 6-7: Distribution of errors in 1DTCNN landmark prediction for subnasale.

6.4.1.3 Combined DoG and LoG derivative features

Histograms are included here also, and Table 6-7 summarizes the results. Figures overleaf show the distributions. Note where there is little or no variance in the measurements, then the histogram is not included. These results are similar to those of the raw profile contour input feature results and further demonstrate the end-to-end learning capability of the 1DTCNN network when using the raw curvature co-ordinates. These results show a small increase in precision.

Landmark Label	Mean Absolute Error (pixels)	Mean Error (pixels)	Precision (as described by standard deviation) (pixels)
Labiale inferius	0	0	0
Stomion	0.33	0.08	1.52
Labiale superius	4.89	1.81	6.40
Subnasale	1.39	-0.05	3.81
Pronasale	0	0	0

Table 6-7: Summary of Errors and precision for 1DTCNN (measurements in pixels).

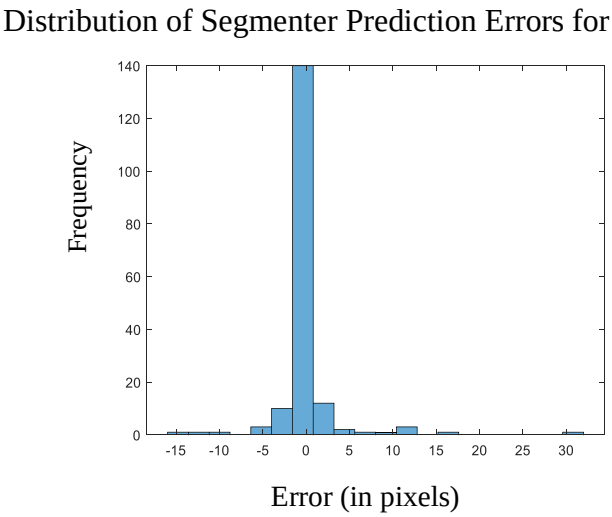


Figure 6-8: Distribution of errors in 1DTCNN landmark prediction for stomion.

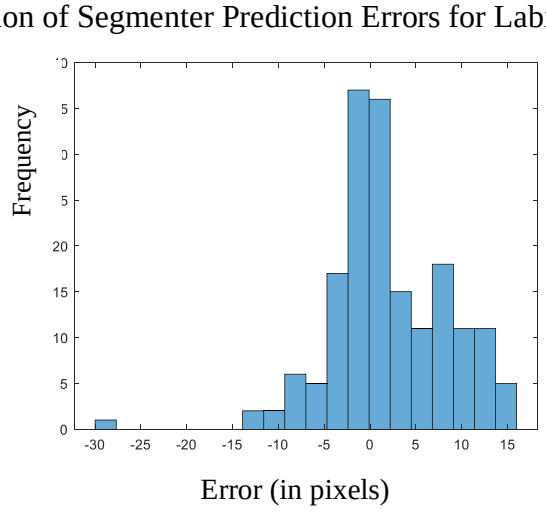


Figure 6-9: Distribution of errors in 1DTCNN landmark prediction for labiale superius.

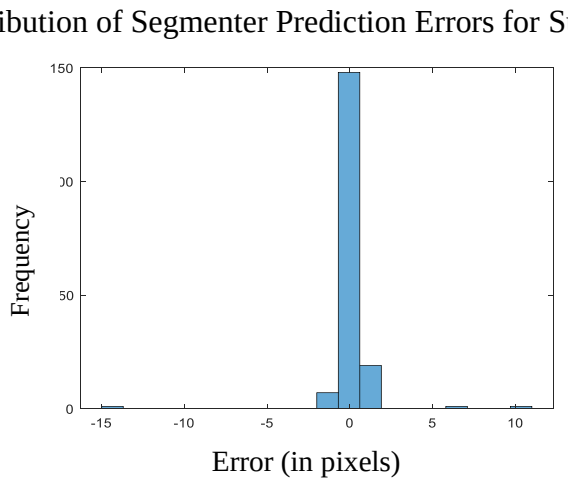


Figure 6-10: Distribution of errors in 1DTCNN landmark prediction for subnasale.

As in the previous sub-section, the labiale superius landmark shows an improved precision and accuracy than the raw landmarks and, in particular, the subnasale precision is the best result from all three features used.

6.5 Regression results and comparisons

This section discusses the accuracy and precision of regressing landmarks using the 1DTCNN network and the three features investigated in the previous section. Figure 6-11 compares the MAE of the three 1DTCNN networks, that is the raw profile curve co-ordinates as an input feature, the combined feature consisting of the first derivatives (*DoG)* and second derivatives (*LoG)* of the profile contour curve, and finally the curvature feature alone. Similarly, Figure 6-12 shows comparisons for the precision. The labiale inferius is not included here since it has zero error and perfect precision.

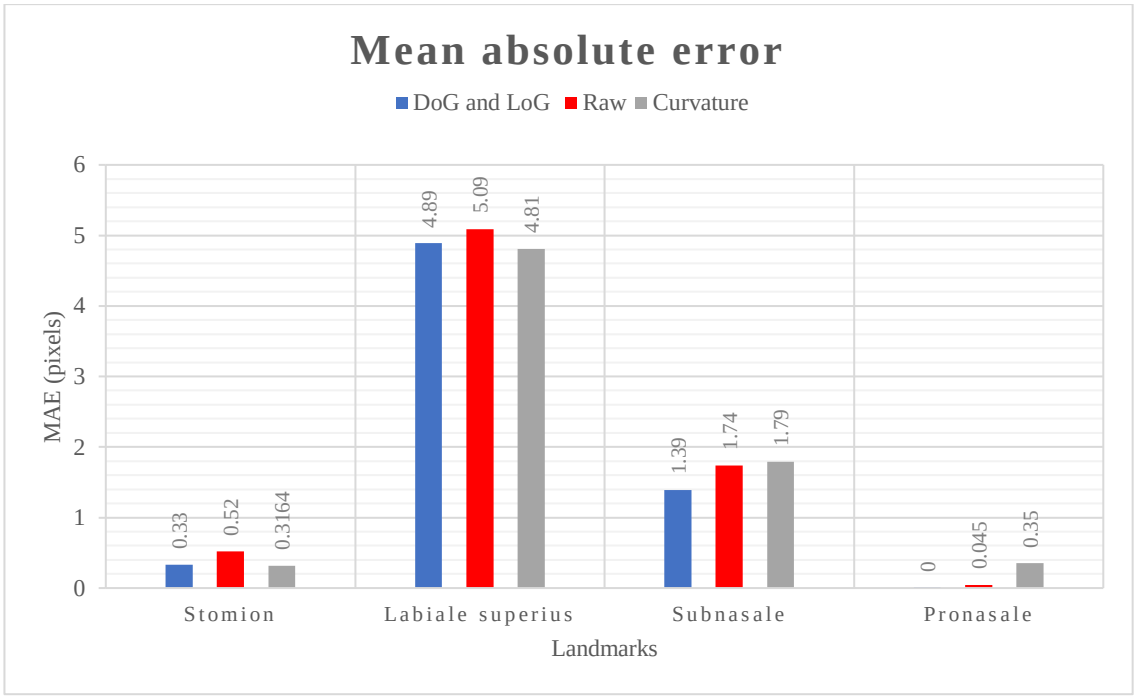


Figure 6-11: Comparison of MAE of 1DTCNN using raw input, curvature and combined LoG and DoG.

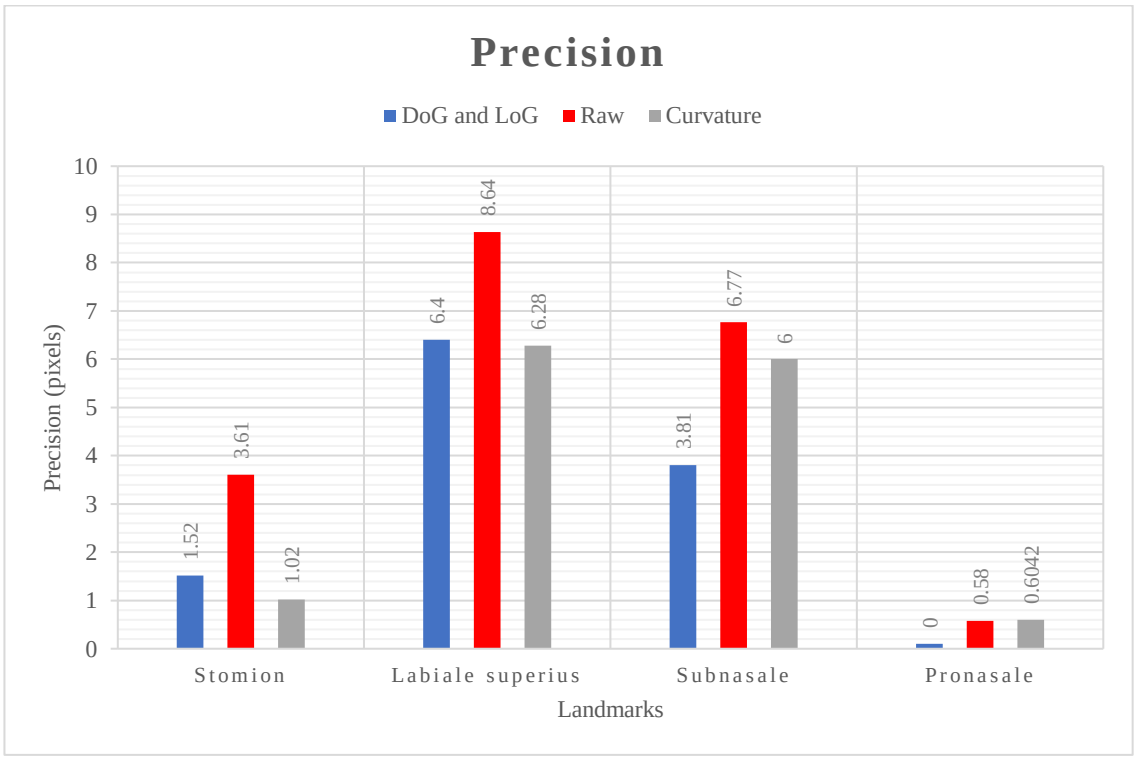


Figure 6-12: Comparison of precision of 1DTCNN using raw inputs, curvature and combined LoG and DoG.

The network trained on the raw data performs well and is comparable with the results obtained for the curvature, and *DoG* and *LoG* feature combination. The pronasale has zero MAE and almost perfect precision. Results are comparably low for the network trained on the *DoG* and *LoG* combination as well as the curvature. These results are expected given that the contour labels are bookended with a fixed amount of n/a labels. As choices for landmarks to use in head posture measurement, these should be avoided as the network learns the number of n/a labels at the start and end of the contour. Nevertheless, other landmarks within the contour are localised well. As before, the localisation of the labiale superius performs least well.

The greatest improvement is achieved when training with the *DoG* and *LoG* combination. Here, the precision of key landmarks is significantly improved. When compared to the raw input feature, the standard deviation that is used to quantify the precision has been more than halved for the stomion and is over 44% less for the subnasale. The labilale superius' precision remains quite high but has been improved by a factor of 26% when also compared with the raw input feature.

Interestingly, the curvature has performed very well given it is only a one-dimensional input vector, compared with the 4 dimensions used to represent the *DoG* and *LoG*. The curvature feature outperforms the *DoG* and *LoG* features for regressing the stomion and labiale superius, though the key subnasale landmark is still regressed best by the network using the *DoG* and *LoG* feature input combinations.

6.6 Comparison of LSTM Results with 1DTNN

The plots of Figure 6-13 and Figure 6-14 compare the precision and accuracy of the best performing LSTM RNN investigated in Chapters 5 and 6 with the 1DTCNN network. The 1DTCNN network does performs better than the best LSTM RNN when the hand-crafted *DoG* and *LoG* features are used for both networks.

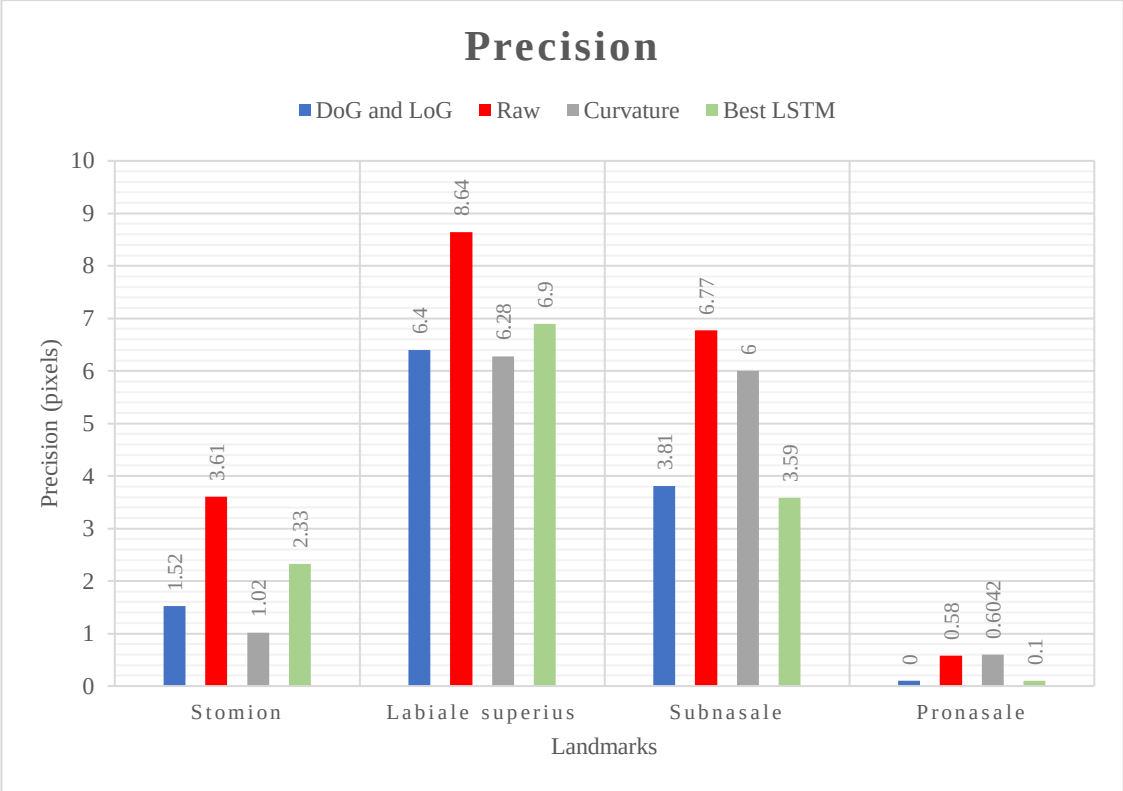


Figure 6-13: Comparison of the precision (in standard deviations) of the best LSTM with the 1DTCNN.

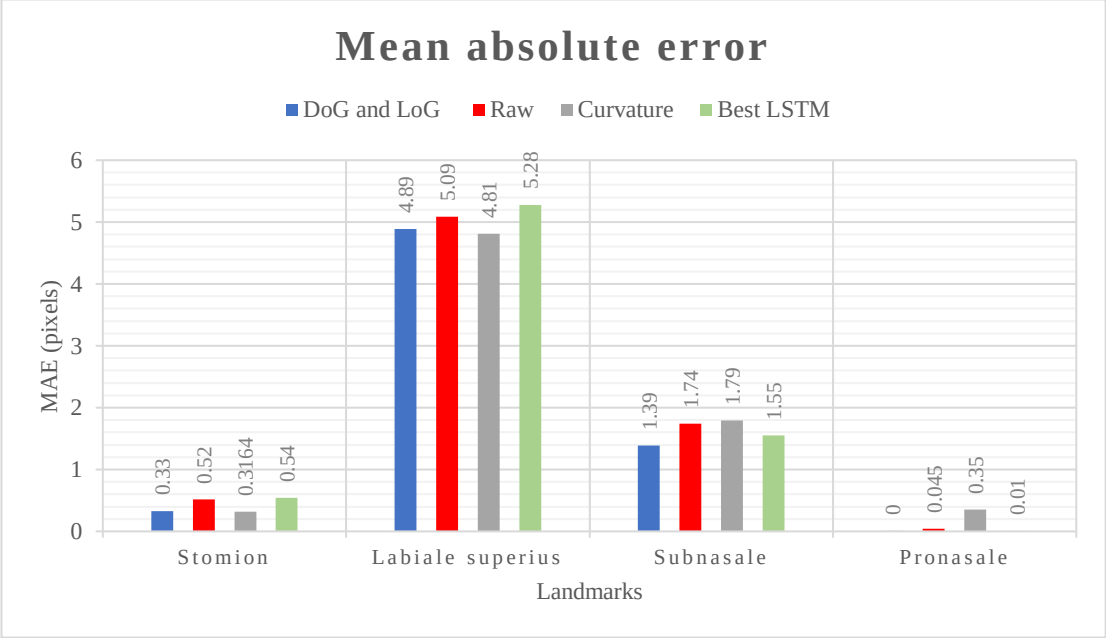


Figure 6-14: Comparison of the accuracy (measured using MAE) of the best LSTM with the 1DTCNN.

Figure 6-15 shows there is also little difference between the performance of the raw input feature 1DTCNN network and the best performing LSTM network with regard to classification accuracy and macro F1 scores. Both score around 91% in each category. The LSTM network that uses the engineered features of *LoG* and *DoG* compares favourably with the results produced using an end-to-end approach of the 1DTCNN. Additionally, the LSTM network has the advantage of faster training and inference times when compared with the 1DTCNN network.

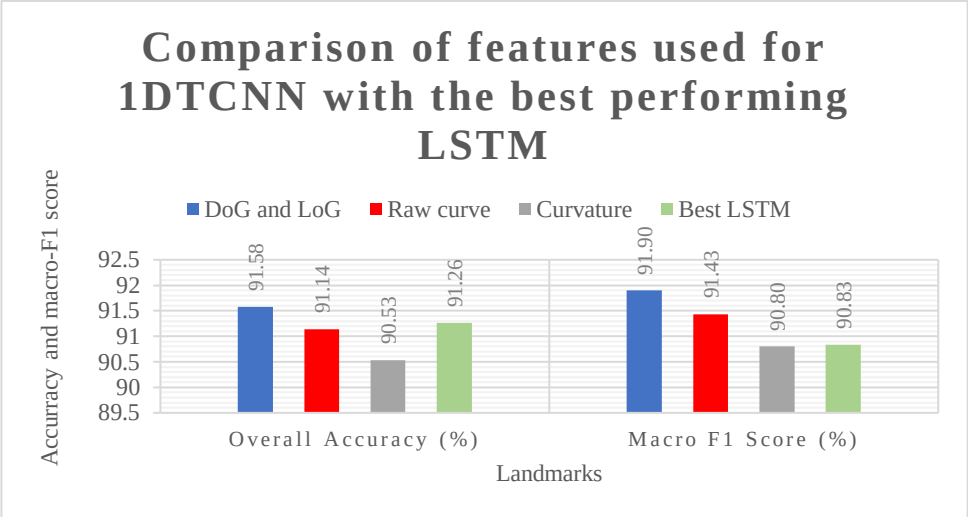


Figure 6-15: Comparison of accuracy and macro-F1 scores for features used with the 1DTCNN network and with the best performing LSTM.

Regarding the regression accuracy calculated using MAE, for the stomion and labial superius there is a difference between the 1DTCNN network and the LSTM network, with the 1DTCNN network improving on the LSTM's MAE score by about 4% in both cases. Conversely, the LSTM network improves upon the 1DTCNN network by 10% when regressing the subnasale landmark. Regarding the pronasale landmark, both networks have a very low, sub-pixel resolution MAE, so a percentage comparison does not help here. The 1DTCNN network has an MAE of 0.001 pixels and the LSTM an MAE of 0.045 pixels.

6.7 Discussions and conclusions

Looking at the results obtained in the experiments of this chapter a clear conclusion is that the 1DTCNN network is capable of good end-to-end learning using just the raw contour profile co-ordinates. However, engineering features prior to training still confers an advantage, especially with regard to improved precision.

The downside of using this network to learn the underlying features from the raw data is the additional time required both to train the network and to regress landmark locations. Regressing landmarks required an order of magnitude increase in inference time when compared to the LSTM DNN used in chapter 5. As concluded in previous investigations in the study, there is a clear trade-off between execution speed and accuracy of classification and regression. The choice depends upon the problem context, for example, where landmarks need to be regressed in real-time then it would be appropriate to choose faster feature engineering and regression whilst sacrificing some accuracy. It is better to have results that meet some baseline accuracies in real time rather than no results at all.

Regressing the location of the first and last landmarks (the labiale inferius and the pronasale) is trivial for these networks given the fixed number of n/a labels adjacent to these. This can be attributed to the design of the experiments and the model, and needs to be addressed in future work. However, as remarked upon previously, the remaining landmarks have been successfully localised using this network and whilst the landmarks are ordered, their precise locations are dependent upon the anatomy of each individual.

Consequently, the conclusion here, as in Chapter 5, is that the best choice of landmark for estimating posture, should head posture estimation be the aim, is the subnasale since it is the most stable, unchanging, landmark when head position or when face expression changes and the 1DTCNN using the *DoG* and *LoG* input feature combination locates it well with a MAE of 1.29 pixels and a precision of 3.81 pixels.

There is an issue here with some outlying results. Further investigation is required in order to locate the reason for the regression errors. It may be due to mislabeling, the result of poorly hand-labelled, ground truth images or the test set still including images that are flawed in some other way.

Regarding run-time efficiency, the 1DTCNN network does not perform well when compared with the LSTM DNN network. The 1DTCNN network takes between 4.6 and 4.9 seconds to segment all 197 contours. That corresponds to about 25ms to segment one contour plus a one-off initialization time of 33ms recorded for the combined *DoG* and *LoG* feature. Overall, the 1DTCNN network is approximately 10 times slower than the best LSTM DNN network.

These results have been achieved with a reasonably small dataset of 454 image profile contours. The more sophisticated 1DTCNN network will certainly perform better with a larger dataset. To achieve this, the contours could be augmented by creating a contour profile derived from a combination of two randomly selected contours and applying established methods used in the graphics field, for example, morphing (Beier and Neely, 1992) and in particular a 2D process based upon the idea of "tweening" (Hill Jr. and Kelley, 2006). Since there are 454 available training images then by selecting two images with replacement, it is possible to choose over a hundred thousand combinations and so generate an additional one hundred thousand images.

The following chapter concludes the thesis, bringing together key findings and conclusions from both this chapter as well as chapters 5 and 6. It identifies the contributions of this thesis to the field as well identifying some limitations of the work.

7 Conclusions and Recommendations

This chapter draws together conclusions and considers the contributions of this thesis and its limitations. Contributions to knowledge are detailed in section 7.1 and thesis conclusions are presented in Section 7.2.

7.1 Contributions

A number of novel ideas were used that contributed to the overall aim of this thesis. An evaluation of the effectiveness of curvature and derivatives was produced that demonstrated the utility of the proposed features in the fast segmenting of head profile contours and the regressing of landmarks (see the published paper of Appendix A). Additionally, a new dataset of labelled face profile contours was created for use in evaluating the features created. In order to automatically improve the accuracy of the annotations, a novel approach was developed in this context based upon the curvature properties of selected anthropometric landmarks and the head profile contour curve itself.

Extending the work of chapter 4 and using the head profile contour dataset created in this study resulted in a new procedure and model that can accurately achieve fast face segmentation of head profile images. An evaluation of this procedure documented both the accuracy of the approach and its run-time efficiency when used with two LSTM RNNs. Additionally, this method was extended once more to develop a method to regress landmarks from the segmented profile contour with good accuracy and precision. To the best of the author's knowledge this is the first use of RNNs to segment and regress head profile contours extracted from real world, un-processed, RGB-D images and has not been done elsewhere with the dataset generated as part of this thesis.

Finally, a 1DTCNN network was applied to the problem of head profile contour segmentation and regression and demonstrated the power of end-to-end ML methods, showing they are effective but at the cost of a significantly slower inference time. To the best of the author's knowledge a 1DTCNN network has not been used to segment and regress real-world, un-processed head profile contours before and not with the dataset generated as part of this thesis.

7.2 Conclusions

Chapters 4, 5 and 6 discussed results and drew conclusions within the context of the experiments undertaken. In this section these are considered from both the context of the aims, objectives and hypotheses of this thesis and also from a broader perspective. The aims, objectives and hypotheses are restated here for the convenience of the reader and the extent to which these have been achieved is then discussed.

The aim of this study was to:
Explore extensively the suitability of curvature and its properties as features for fast regression and segmentation of parameterized plane curves, and in so doing, examine the effectiveness of these features in training deep neural networks to estimate head profile posture derived from 2.5D images.

From this aim the following objectives were enumerated:

1. To engineer and evaluate features derived from plane curves to train supervised machine learning models capable of efficiently segmenting regions of interest.

2. To develop a dataset of accurately landmarked head profile contours.

3. To develop and evaluate fast ML models capable of estimating head profile by segmenting profile contours into regions of interest and regressing key head profile landmarks.

4. Demonstrate that engineered features, in the context of this thesis, compare well with end-to-end ML approaches.

It was hypothesized that:

1. Curvature and the related first and second derivatives of a curve can be efficiently calculated from a given plane curve and these features will enable the fast and accurate segmentation of a curve when used in conjunction with suitable ML models.

2. A second hypothesis is that the same ML models can also be used to efficiently regress points on a plane curve with high accuracy and precision.

3. A final hypothesis is that, in the context of this thesis, these engineered features can produce results superior to an end-to-end ML approach.

7.2.1 Review of Objectives

The first objective of this thesis has been achieved. Chapter 4 demonstrated that the engineered features developed and evaluated here were capable of accurately and efficiently segmenting a uniformly sampled time series dataset and provided justification for further investigations with overall F1 score and accuracy both achieving levels of 87% on the benchmark LSTM RNN. These results surpassed those achieved using short time Fourier methods by approximately 1.5% and exhibited a run-time speed orders of magnitude faster.

The second objective has also been achieved by annotating the Notre Dame dataset introduced in section 5.2 with anthropometric landmarks and extracting a dataset of 648 useable head profile contours from the RGBD images. The usable regions were limited to the face, and this was sufficient to accurately segment regions of interest and regress key landmarks. A larger dataset can now be developed by landmarking further images from the Notre Dame dataset or augmenting it by creating artificial contours using the tweening concept discussed in chapter 6.

Using these features as inputs to an LSTM DNN to segment regions of interest on a head contour and regress landmarks proved successful, too, with a further improvement of 4% percent achieved. However, the context of the problem is important and section 4.4 shows the effective quality of a selected feature needs to be evaluated by weighting the importance of speed and accuracy. The weighted L2 norm distance measure described here provided a numerical score of effectiveness and demonstrates that the best features to use depend both upon execution speed and desired accuracy.

The network used to segment the ECG dataset in chapter 4 is, by necessity, identical to the architecture used in (Mathworks, 2020c) in order to ensure a valid comparison between the features used in that experiment and those engineered and used here. Whilst

the results demonstrated an improvement, the author notes that different classifiers and network architectures, together with further tuning of hyper-parameters will improve upon the accuracy of these results. For example, Moskalenko, Zolotykh and Osipov (2019) propose a superior UNet like convolutional deep neural network that claims F1 scores of 97% and above on an ECG dataset.

Chapters 5 and 6 follow from the third objective of this thesis. They demonstrate that the selected features evaluated in chapter 4 could accurately segment regions and regress landmarks in a more generalised environment, that is, in any plane curve. Here, head profile contours extracted from RGBD images were segmented and regressed and the results obtained demonstrate that good segmentation of regions is possible (the best performing feature combinations achieved a macro-F1 score of 91% and an overall accuracy of 91%) without extensive parametric investigation and architectural modification of the chosen network.

The discussions of section 6.7 point out that the inference time for one contour using an LSTM network is 10 times faster than those of the 1DTCNN network, hence any decisions regarding the suitability of a given network still needs to consider both the desired accuracy as well as the classification times.

In general, all networks and feature combinations perform least well when regressing the labiale superius. The author concludes that this is due to the phenotypic diversity of the anatomy of the upper lip and philtrum of subjects within the dataset, together with the assumptions made in the automatic regression algorithm. In contrast the pronasale and labiale inferius are located with near perfect accuracy and precision. This can be attributed to the design of the experiments. Placing a fixed number of "n/a" labels at the start and end of the contour was an imperfect design choice since the networks easily learned the "n/a" label positions and hence could accurately locate the first and last landmarks. This, however, did not impact on the accuracy and precision of regressing the remaining landmarks.

Extracting the head profile contour from a 2D image using classic computer vision algorithms adds additional pre-processing steps to the segmentation process. Chapters 2 and 5 note that RGB images with depth information allow fast head profile contour

extraction and so avoids these pre-processing steps. Current technology also allows RGBD images to be captured at 30fps.

Overall, the third objective has been achieved. Accurate segmentation is possible as is fast classification of regions of interest. Accurate regression of key landmarks is also possible, but the experimental results introduce additional complications indicating that further work is required. This is discussed in the following chapter on future work.

The fourth and final objective, demonstrating the effectiveness of engineered features when compared to the end-to-end machine learning approach, has been achieved and some additional results have emerged from this. The end-to-end ML approach initially used the raw contour data as an input feature to the 1DTCNN network of chapter 6 and was left to learn the relevant feature information. Overall, it proved to learn features well from the raw data although at some cost. The best LSTM (using *LoG* and *DoG* features) significantly outperformed the 1DTCNN network in terms of inference time (the LSTM was 10 times faster). Additionally, the regression precision of the 1DTCNN network when using just the raw data as input could not match that of the LSTM network.

From the perspective of measuring accuracy and macro F1 scores alone, the 1DTCNN network produces slightly better results overall when engineered features are used, but the time required to train and classify is, again, significantly greater than the LSTM network. Additionally, the use of curvature as a feature for the 1DTCNN network produced good results and, interestingly, used only a one-dimensional data structure, compared with a four dimensional one of the derivatives.

7.2.2 Review of Hypotheses

The hypotheses re-iterated at the start of this section are reviewed next. Whilst it was hypothesised that curvature would be a good feature to use with ML models and methods, it seems that, for the LSTM DNN networks used in chapter 4, the first and second derivatives, which themselves are used to calculate curvature, consistently produced results better than curvature alone. They are also far quicker to compute than curvature. Hence it is concluded that the first hypothesis has been met, but *DoG* and *LoG* features are superior to curvature alone in terms of their predictive properties and speed of calculation when used with LSTM network. The modified LSTM network showed an

improvement when curvature was used alone as an input feature indicating it contained useful information and this was further confirmed by the experiments using the 1DTCNN network. The 1DTCNN network was able to use curvature alone to produce comparable macro F1 scores (90.83% for the LSTM network and 90.8% for the 1DTCNN network) and accuracy (91.26% for the LSTM network and 90.53% for the 1DTCNN network), which is a noteworthy result since it is the only scalar (one-dimensional) input feature.

The second hypothesis focused on the regression capabilities of the deep learning models used and their features. Analysis of the figures above show this hypothesis to be proven for the models used, however both regressors demonstrate poorer accuracy and precision for the labiale superius landmark. The conclusion here is that the dataset needs to be enlarged and the automatic landmark adjustment algorithm needs further work. Additionally, the experiments could benefit from modifying the length of the contours by adding a random amount of "n/a" labels at the beginning and end of the contours.

With regard to the third hypothesis, it has been proven overall. The LSTM network with engineered features produced a faster segmenter and regressor with accuracy comparable to that of an end-to-end ML approach. Nevertheless, the results of chapter 6 demonstrated the power and potential usefulness of the end-to-end ML approach. Equally it shows the value of engineering good features, as a significant speed up in classification was demonstrated in this thesis. The author notes that these results are dependent upon the processing hardware used.

Next, the potential for future work is considered.

8 Future Work

This chapter presents future work that could both improve and extend this thesis or that could become a focus for a new research project.

8.1 A real-time profile landmarking application

Primarily, the next step is to create a single application that can estimate head posture in real-time. This will combine the methods and ideas demonstrated here into a single application that can take real-time, 2D colour images, including depth information, to locate key landmarks and segment profiles as demonstrated in chapter 5. The technology to produce real-time, 2D images with depth information is cheap and already available, for example Microsoft's Azure Kinect device can work in real time and can produce high resolution 2D images with depth information at 30fps. Measuring head posture angle would be an obvious goal. This would require further work to regress the tragus of the ear and the C7 vertebra, or an alternative reference could be used instead, perhaps using markers if no other method was available. Such an application would have uses in several fields as identified already in chapter 2.

8.2 Investigate alternative networks

The investigations of chapter 4 compared the engineered features developed in this thesis with those of a previously published experiment. No attempt to investigate alternative models was undertaken since the purpose was to compare the features engineered here against an existing model, dataset and experiment. Future work could continue this investigation by using alternative networks, architectures and hyperparameters.

8.3 Augment the developed dataset

The head profile contour dataset is not large but could easily be augmented using the "tweening" concept discussed in chapter 6 and then investigating the effect of a larger dataset on the networks developed here already. For example, a further study to evaluate any improvement in the accuracy of the segmentation process and regression capability of both the LSTM RNN and 1DTCNN network models would be of interest.

8.4 Creation of a new dataset

Whilst the Notre Dame dataset was invaluable for this thesis, the creation of a labelled, higher resolution 2D head profile image dataset with depth information would be an important goal as image capturing technologies have significantly evolved since the creation of the Notre Dame dataset and the resulting, new data set would be likely to have fewer flawed images. Extending this approach into real-time video would allow the use of additional methods such as motion tracking, for example, including Kalman filter (Stratonovich, 1959) and particle filter methods (Gordon, Salmond and Smith, 1993).

8.5 Extend and refine the landmarking algorithm

Whilst the method used to regress the landmark locations on the profile produces some reasonable results, it could be improved. Currently the method searches for the first transition and assumes the ordinal nature of the sequence has been learned by the model. This has not been proven here. Hence a statistical approach might benefit the regression process, for example, by looking at the landmarks around the transition zone and choosing an average or median value.

8.6 Investigate multi-scale input features

It is observed that the 1DTCNN network makes explicit use of dilation layers to capture scale dependent features. The LSTM RNN models used features that had a fixed scale determined by the standard deviation, σ, used in the curvature and derivative calculations. Using, for example, three *DoG* features with scales of σ=1, 2 and 3 at the same time, would, it is hypothesised, improve the accuracies of the classes segmented.

8.7 Investigate the energy efficiency of approaches used in this research

Finally, with a wide range of hardware becoming available such as GPUs, dedicated camera systems, multi-core microprocessors and so on, it will be worthwhile to study the efficiency of the approaches used in this thesis from the context of power consumption.

This is pertinent given the increased use of mobile based hardware and, more generally, the limited resources available to power these devices.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2016) 'TensorFlow: A system for large-scale machine learning', *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pp. 265–283. doi:10.5555/3026877.3026899.

Abdou, I.E. and Pratt, W.K. (1979) 'Quantitative design and evaluation of enhancement/thresholding edge detectors', *Proceedings of the IEEE*, 67(5), pp. 753–763. doi:10.1109/PROC.1979.11325.

Aksu, M., Kaya, D. and Kocadereli, I. (2010) 'Reliability of reference distances used in photogrammetry', *The Angle Orthodontist*, 80(4), pp. 670–677. Available at: http://www.angle.org/doi/abs/10.2319/070309-372.1.

Aldridge, K., Boyadjiev, S.A., Capone, G.T., DeLeon, V.B. and Richtsmeier, J.T. (2005) 'Precision and error of three-dimensional phenotypic measures acquired from 3dMD photogrammetric images', *American Journal of Medical Genetics Part A*, 138A(3), pp. 247–253. doi:10.1002/ajmg.a.30959.

Altun, K., Barshan, B. and Tunçel, O. (2010) 'Comparative study on classifying human activities with miniature inertial and magnetic sensors', *Pattern Recognition*, 43(10), pp. 3605–3620. doi:10.1016/j.patcog.2010.04.019.

Álvarez Casado, C. and Bordallo López, M. (2021) 'Real-time face alignment: evaluation methods, training strategies and implementation optimization', *Journal of Real-Time Image Processing*, 18(6), pp. 2239–2267. doi:10.1007/S11554-021-01107-W/FIGURES/15.

Auger, F., Flandrin, P., Lin, Y.T. and McLaughlin, S. (2013) ‘Time-frequency reassignment and synchrosqueezing: An overview’, *IEEE Signal Processing Magazine*, 30(6), pp. 32–41. doi:10.1109/MSP.2013.2265316.

Azevedo, A. and Santos, M.F. (2008) ‘KDD, SEMMA and CRISP-DM: A Parallel Overview’, *ISCAP - Sistemas de Informação - Comunicações em eventos científicos* [Preprint].

Bai, S., Kolter, J.Z. and Koltun, V. (2018) ‘An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling’.

Bay, H., Tuytelaars, T. and Gool, L. Van (2006) ‘SURF: Speeded Up Robust Features’, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS, pp. 404–417. doi:10.1007/11744023_32.

Beier, T. and Neely, S. (1992) ‘Feature-based image metamorphosis’, *Computer Graphics*, 26(2), pp. 35–42. doi:10.1145/133994.134003.

Belhumeur, P., Jacobs, D., Kriegman, D. and Kumar, N. (2011) ‘Localizing parts of faces using a consensus of exemplars’, in *Proc. of Conf. on Computer Vision and Pattern Recognition*.

Belongie, S., Malik, J. and Puzicha, J. (2001) ‘Matching shapes’, *Proceedings of the IEEE International Conference on Computer Vision*, 1, pp. 454–461. doi:10.1109/ICCV.2001.937552.

Bhanu, B. and Zhou, X. (2004) ‘Face recognition from face profile using dynamic time warping’, *Proceedings - International Conference on Pattern Recognition*, 4, pp. 499–502. doi:10.1109/ICPR.2004.1333820.

Bishop, C.M. (2006) *Pattern recognition and machine learning*. 1st edn. Cambridge: Springer Verlag.

Bottino, A. and Cumani, S. (2008) ‘A fast and robust method for the identification of face landmarks in profile images’, *WSEAS, Transactions on Computers*, 7(8), pp. 1250–1259.

Bouma, H., Vilanova, A., Bescós, J., Ter Haar Romeny, B. and Gerritsen, F. (2007) 'Fast and accurate Gaussian derivatives based on B-splines', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Verlag, pp. 406–417. doi:10.1007/978-3-540-72823-8_35.

Bradbury, J., Merity, S., Xiong, C. and Socher, R. (2016) 'Quasi-Recurrent Neural Networks', *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* [Preprint].

Canny, J. (1986) 'A Computational Approach to Edge Detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), pp. 679–698. doi:10.1109/TPAMI.1986.4767851.

Çeliktutan, O., Ulukaya, S. and Sankur, B. (2013) 'A comparative study of face landmarking techniques', *EURASIP Journal on Image and Video Processing*, 2013(1), p. 13. doi:10.1186/1687-5281-2013-13.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014) 'Learning phrase representations using RNN encoder-decoder for statistical machine translation', in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*. Association for Computational Linguistics (ACL), pp. 1724–1734. doi:10.3115/v1/d14-1179.

Chollet, F. and Others (2015) *Keras*. Available at: https://keras.io.

Cootes, T., Taylor, C.J., Cooper, D.H. and Grahama, J. (1995) 'Active shape models: their training and application', *Comput. Vis. Image Understand*, 91. doi:10.1006/cviu.1995.1004.

Cootes, T.F., Edwards, G.J. and Taylor, C.J. (1998) 'Active appearance models', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1407, pp. 484–498. doi:10.1007/BFB0054760.

Cootes, T.F., Edwards, G.J. and Taylor, C.J. (2001) ‘Active appearance models’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), p. 681. doi:10.1109/34.927467.

Dalal, N. and Triggs, B. (2005) ‘Histograms of oriented gradients for human detection’, in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, pp. 886–893. doi:10.1109/CVPR.2005.177.

Deberry-Borowiecki, B., Kukwa, A. and Blanks, R.H.I. (1988) ‘Cephalometric Analysis for Diagnosis and Treatment of Obstructive Sleep Apnea’, *The Laryngoscope*, 98(2), pp. 226–234. doi:10.1288/00005537-198802000-00021.

Deng, J., Guo, J., Zhang, D., Deng, Y., Lu, X. and Shi, S. (2019) ‘Lightweight face recognition challenge’, *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 2638–2646. doi:10.1109/ICCVW.2019.00322.

Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I. and Zafeiriou, S. (2019) ‘RetinaFace: Single-stage Dense Face Localisation in the Wild’, *arXiv preprint arXiv 1905.00641* [Preprint].

Deng, J., Roussos, A., Chrysos, G., Ververas, E., Kotsia, I., Shen, J. and Zafeiriou, S. (2019) ‘The Menpo Benchmark for Multi-pose 2D and 3D Facial Landmark Localisation and Tracking’, *International Journal of Computer Vision*, 127(6–7), pp. 599–624. doi:10.1007/s11263-018-1134-y.

Dickers, G., Rees, J., Bashford, T. and Ademoye, O. (2021) ‘Effectiveness of Curvature and Signal Derivatives in Fast Curve Segmentation’, in *Conference Proceedings UKSim2021, AIMS2020/21*. International Journal of Simulation Systems, Science & Technology, p. vol 22. doi:10.5013/IJSSST.a.22.01.12.

Dindaroğlu, F., Kutlu, P., Duran, G.S., Görgülü, S. and Aslan, E. (2015) ‘Accuracy and reliability of 3D stereophotogrammetry: A comparison to direct anthropometry and 2D photogrammetry’, *The Angle Orthodontist*, 86(3), pp. 487–494. doi:10.2319/041415-244.1.

Ding, L. and Martinez, A.M. (2010) ‘Features versus context: an approach for precise and detailed detection and delineation of faces and facial features’, *IEEE Trans. Pattern Anal. Mach. Intell*, 32. doi:10.1109/TPAMI.2010.28.

Eastwood, P., Gilani, S.Z., McArdle, N., Hillman, D., Walsh, J., Maddison, K., Goonewardene, M. and Mian, A. (2020) ‘Predicting sleep apnea from 3-dimensional face photography’, *Journal of Clinical Sleep Medicine* [Preprint]. doi:10.5664/jcsm.8246.

Efraty, B.A., Ismailov, E., Shah, S. and Kakadiaris, I.A. (2009) *Towards 3D-aided Profile-Based Face Recognition, IEEE 3rd International Conference on Biometrics: Theory, Applications and Systems, BTAS 2009*. doi:10.1109/BTAS.2009.5339078.

Farid, H. and Simoncelli, E.P. (2004) ‘Differentiation of Discrete Multidimensional Signals’, *IEEE Transactions on Image Processing*, 13(4). doi:10.1109/TIP.2004.823819.

Farkas, L.G. (1994) ‘Anthropometry of the head and face (ed 2)’, *Journal of Oral and Maxillofacial Surgery*, 53(6), p. 733. doi:10.1016/0278-2391(95)90208-2.

Farkas, L.G., Bryson, W. and Klotz, J. (1980) ‘Is Photogrammetry of the Face Reliable?.’, *Plastic and Reconstructive surgery*, 66(3), pp. 346–355.

Fawaz, H.I., Forestier, · Germain, Weber, J., Lhassane Idoumghar, · and Muller, P.-A. (2019) ‘Deep learning for time series classification: a review’, *Data Mining and Knowledge Discovery*, 33, pp. 917–963. doi:10.1007/s10618-019-00619-1.

Fawzy Mahmoud, N., Hassan, K.A., Abdelmajeed, S.F., Moustafa, I.M. and Silva, A.G. (2019) ‘The Relationship Between Forward Head Posture and Neck Pain: a Systematic Review and Meta-Analysis’, *Current reviews in musculoskeletal medicine*, 12(4), pp. 562–577. doi:10.1007/s12178-019-09594-y.

Fukushima, K. (1980) ‘Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position’, *Biol. Cybernetics*, 36, pp. 193–202. doi:10.1016/0893-6080(88)90014-7.

Géron, A. (2019) *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*. 2nd edn. O'Reilly Media.

Gers, F.A. and Schmidhuber, J. (2000) 'Recurrent nets that time and count', *Proceedings of the International Joint Conference on Neural Networks*, 3, pp. 189–194. doi:10.1109/IJCNN.2000.861302.

Ghoddousi, H., Edler, R., Haers, P., Wertheim, D. and Greenhill, D. (2007) 'Comparison of three methods of facial measurement', *International Journal of Oral and Maxillofacial Surgery*, 36(3), pp. 250–258. doi:10.1016/j.ijom.2006.10.001.

Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K. and Stanley, H.E. (2000) 'PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.', *Circulation*, 101(23). doi:10.1161/01.cir.101.23.e215.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press.

Gordon, C.C., Blackwell, C.L., Bradtmiller, B., Parham, J.L., Barrientos, P., Paquette, S.P., Corner, B.D., Carson, J.M., Venezia, J.C. and Rockwell, B.M. (2014) *2012 Anthropometric Survey of US Army Personnel: Methods and Summary Statistics*. Army Natick Soldier Research Development and Engineering Center MA.

Gordon, N.J., Salmond, D.J. and Smith, A.F.M. (1993) 'Novel approach to nonlinear/non-gaussian Bayesian state estimation', *IEE Proceedings, Part F: Radar and Signal Processing*, 140(2), pp. 107–113. doi:10.1049/IP-F-2.1993.0015.

Graves, A., Ch, A., Fernández, S., Gomez, F., Schmidhuber, J. and Ch, J. (2006) 'Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks', in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376.

Graves, A., Mohamed, A.-R. and Hinton, G. (2013) 'Speech Recognition with Deep Recurrent Neural Networks', in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 369–376.

Gray, A., Abbena, E. and Salamon, S. (2017) *Modern Differential Geometry of Curves and Surfaces with Mathematica®*. 3rd edn. Chapman and Hall/CRC.

Gross, R., Matthews, I., Cohn, J., Kanade, T. and Baker, S. (2010) 'Multi-PIE', *Image Vis. Comput*, 28. doi:10.1016/j.imavis.2009.08.002.

Handelman, G.S., Kok, H.K., Chandra, R. V., Razavi, A.H., Huang, S., Brooks, M., Lee, M.J. and Asadi, H. (2018) 'Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods', *https://doi.org/10.2214/AJR.18.20224*, 212(1), pp. 38–43. doi:10.2214/AJR.18.20224.

He, K., Zhang, X., Ren, S. and Sun, J. (2016) 'Deep residual learning for image recognition', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, pp. 770–778. doi:10.1109/CVPR.2016.90.

Hill Jr., F.S. and Kelley, S.M. (2006) 'Computer Graphics Using OpenGL (3rd Edition)'.

Hochreiter, S. and Schmidhuber, J. (1997) 'Long Short-Term Memory', *Neural Computation*, 9(8), pp. 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Hopp, C.S., Chiou, K., Ragheb, D.R.T., Salman, A.M., Khan, S.M., Liu, A.J. and Sinnis, P. (2015) 'Longitudinal analysis of plasmodium sporozoite motility in the dermis reveals component of blood vessel recognition', *eLife*, 4(AUGUST2015). doi:10.7554/eLife.07789.

Hough, P. (1962) 'Method and means for recognizing complex patterns'. US Patent, 3(6).

Huang, G.B., Mattar, M., Berg, T., Learned-Miller, Eric and Learned-Miller, Erik (2008) 'Labeled faces in the wild: A database forstudying face recognition in unconstrained environments', *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition.* [Preprint].

Hubel, D.H. and Wiesel, T.N. (1959) ‘Receptive fields of single neurones in the cat’s striate cortex’, *The Journal of Physiology*, 148(3), p. 574. doi:10.1113/JPHYSIOL.1959.SP006308.

Intel (2018) *Depth Camera D435 – Intel® RealSense™ Depth and Tracking Cameras*. Available at: https://www.intelrealsense.com/depth-camera-d435/ (Accessed: 30 July 2021).

International Organization for Standardization (1994) *ISO 5725-1:1994(en) Accuracy (trueness and precision) of measurement methods and results — Part 1: General principles and definitions*.

Jayaratne, Y.S.N. and Zwahlen, R.A. (2014) ‘Application of Digital Anthropometry for Craniofacial Assessment’, *Craniomaxillofacial Trauma & Reconstruction*, 7(2), pp. 101–107. doi:10.1055/s-0034-1371540.

Jesorsky, O., Kirchberg, K.J. and Frischholz, R.W. (2001) ‘Robust face detection using the Hausdorff distance’, in *International Conference on Audio- and Video-Based Biometric Person Authentication*. Springer Berlin Heidelberg, pp. 90–95. doi:10.1007/3-540-45344-X_14.

Jin, X. and Tan, X. (2016) ‘Face Alignment In-the-Wild: A Survey’, *Computer Vision and Image Understanding*, 162, pp. 1–22. doi:10.1016/j.cviu.2017.08.008.

Johnston, B. and de Chazal, P. (2018) ‘A review of image-based automatic facial landmark identification techniques’, *EURASIP Journal on Image and Video Processing*, 2018, p. 86. doi:10.1186/s13640-018-0324-4.

Kakadiaris, I.A., Abdelmunim, H., Yang, W. and Theoharis, T. (2008) ‘Profile-based face recognition’, *2008 8th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2008* [Preprint], (May 2014). doi:10.1109/AFGR.2008.4813370.

Kamiri, J. and Mariga, G. (2021) ‘Research Methods in Machine Learning: A Content Analysis’, *International Journal of Computer and Information Technology(2279-0764)*, 10(2), pp. 2279–0764. doi:10.24203/IJCIT.V10I2.79.

Kim, S.-H., Jung, W.-Y., Seo, Y.-J., Kim, K.-A., Park, K.-H. and Park, Y.-G. (2015) 'Accuracy and precision of integumental linear dimensions in a three-dimensional facial imaging system', *Korean Journal of Orthodontics*, 45(3), pp. 105–112. doi:10.4041/kjod.2015.45.3.105.

Kline, M. (1972) *Mathematical thought from ancient to modern times Volume 2 (vol. 2)*. Oxford University Press.

Kolar, J.C. and Salter, E.M. (1997) *Craniofacial Anthropometry: Practical Measurement of the Head and Face for Clinical, Surgical, and Research Use*. C.C. Thomas, Springfield.

Köstinger, M., Wohlhart, P., Roth, P.M. and Bischof, H. (2011) 'Annotated facial landmarks in the wild: a large-scale, real-world database for facial landmark localization', in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on Benchmarking Facial Image Analysis Technologies,*. Spain: IEEE, pp. 2144–2151. doi:10.1109/ICCVW.2011.6130513.

Krizhevsky, A., Sutskever, I. and Hinton, G. (2012) 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems*, 25, p. pp 1097–1105. doi:10.1145/3065386.

Kruger, N., Janssen, P., Kalkan, S., Lappe, M., Leonardis, A., Piater, J., Rodriguez-Sanchez, A.J. and Wiskott, L. (2013) 'Deep hierarchies in the primate visual cortex: What can we learn for computer vision?', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), pp. 1847–1871. doi:10.1109/TPAMI.2012.272.

Kuehnapfel, A., Ahnert, P., Loeffler, M., Broda, A. and Scholz, M. (2016) 'Reliability of 3D laser-based anthropometry and comparison with classical anthropometry OPEN', *Nature Publishing Group* [Preprint]. doi:10.1038/srep26672.

Kumar, N. and Sharma, D. (2017) 'A Review on Machine Learning Algorithms, Tasks and Applications', *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 6(10), pp. 2278–1323.

Laguna, P., Jané, R. and Caminal, P. (1994) ‘Automatic detection of wave boundaries in multilead ECG signals: Validation with the CSE database’, *Computers and Biomedical Research*, 27(1), pp. 45–60. doi:10.1006/cbmr.1994.1006.

Lam, B., Ip, M.S.M., Tench, E. and Ryan, C.F. (2005) ‘Craniofacial profile in Asian and white subjects with obstructive sleep apnoea’, *Thorax*, 60(6), p. 504. doi:10.1136/thx.2004.031591.

Le, V., Brandt, J., Lin, Z., Bourdev, L. and Huang, T.S. (2012) ‘Interactive facial feature localization’, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer, Berlin, Heidelberg, pp. 679–692. doi:10.1007/978-3-642-33712-3_49.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D. (1989) ‘Backpropagation Applied to Handwritten Zip Code Recognition’, *Neural Computation*, 1(4), pp. 541–551. doi:10.1162/NECO.1989.1.4.541.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998) ‘Gradient-based learning applied to document recognition’, *Proceedings of the IEEE*, 86(11), pp. 2278–2323. doi:10.1109/5.726791.

Liang, L., Xiao, R., Wen, F. and Sun, J. (2008) ‘Face alignment via component-based discriminative search’, in *European Conf. on Computer Vision*. Springer, Berlin, Heidleberg, pp. 72–85. doi:10.1007/978-3-540-88688-4_6.

Linnainmaa, S. (1976) ‘Taylor expansion of the accumulated rounding error’, *BIT Numerical Mathematics 1976 16:2*, 16(2), pp. 146–160. doi:10.1007/BF01931367.

Lipira, A.B., Sachanandani, N.S., Govier, D., Payne, A., Wyas, S., Kleeschulte, W. and Kane, A.A. (2010) ‘Craniobank: an online collection of three-dimensional normative craniofacial images’, *Plastic and reconstructive surgery*, 126(2), pp. 70e-72e. doi:10.1097/PRS.0b013e3181de23e5.

Lipoščak, Z. and Lončarić, S. (1999) *A scale-space approach to face recognition from profiles, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springe, Berlin, Heidelberg. doi:10.1007/3-540-48375-6_30.

Lippold, C., Liu, X., Wangdo, K., Drerup, B., Schreiber, K., Kirschneck, C., Moiseenko, T. and Danesh, G. (2014) 'Facial landmark localization by curvature maps and profile analysis', *Head and Face Medicine*, 10(1), pp. 1–7. doi:10.1186/1746-160X-10-54.

Lowe, D.G. (2004) 'Distinctive image features from scale-invariant keypoints', *Int. J. Comput. Vis*, 60. doi:10.1023/B:VISI.0000029664.99615.94.

Marr, D. and Hildreth, E. (1980) 'Theory of edge detection', *Proceedings of the Royal Society of London - Biological Sciences*, 207(1167), pp. 187–217. doi:10.1098/rspb.1980.0020.

Martinez, A. and Benavente, R. (1998) *The AR face database: CVC Technical Report, 24*.

Masters, D. and Luschi, C. (2018) 'Revisiting Small Batch Training for Deep Neural Networks', *arXiv preprint arXiv:1804.07612* [Preprint].

Mathworks (2020a) *Deep Learning Toolbox Documentation - MathWorks United Kingdom*. Available at: https://uk.mathworks.com/help/deeplearning/ (Accessed: 14 August 2021).

Mathworks (2020b) *Measure the Performance of Your Code - MATLAB & Simulink - MathWorks Benelux*. Available at: https://uk.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html (Accessed: 20 July 2021).

Mathworks (2020c) *Waveform Segmentation Using Deep Learning - MATLAB & Simulink - MathWorks United Kingdom*. Available at: https://uk.mathworks.com/help/signal/ug/waveform-segmentation-using-deep-learning.html#mw_rtc_WaveformSegmentationUsingDeepLearningExample_BFEA95F4 (Accessed: 13 May 2021).

Maxion, R. (2009) 'Experimental Methods for Computer Science Research', pp. 136–136. doi:10.1109/LADC.2009.29.

McCarthy, J., Minsky, M.L., Rochester, N. and Shannon, C.E. (2006) ‘A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955’, *AI Magazine*, 27(4), pp. 12–12. doi:10.1609/AIMAG.V27I4.1904.

McCulloch, W.S. and Pitts, W. (1943) ‘A logical calculus of the ideas immanent in nervous activity’, *The bulletin of mathematical biophysics 1943 5:4*, 5(4), pp. 115–133. doi:10.1007/BF02478259.

McGlone, J. (2004) *Manual of photogrammetry*. 5th ed. Bethesda: American Society for Photogrammetry and Remote Sensing.

Messer, K., Matas, J., Kittler, J., Luettin, J. and Maitre, G. (1999) ‘XM2VTSDB: the extended M2VTS database’, *Second international conference on audio and video-based biometric person authentication*, 964, pp. 965–966.

Meyer, F. and Beucher, S. (1990) ‘Morphological segmentation’, *Journal of Visual Communication and Image Representation*, 1(1), pp. 21–46. doi:10.1016/1047-3203(90)90014-M.

Microsoft (2019) *Azure Kinect DK*. Available at: https://azure.microsoft.com/en-gb/services/kinect-dk/#overview (Accessed: 30 July 2021).

Milborrow, S. and Nicolls, F. (2008) ‘Locating Facial Features with an Extended Active Shape Model’, in *European conference on computer vision*. Springer, Berlin, Heidelberg, pp. 504–513. doi:10.1007/978-3-540-88693-8_37.

Minolta (2001) *Non-contact 3D Digitizer Vivid 910/VI-910 Instruction Manual (hardware)*.

Moore, G.A. (1968) ‘Automatic scanning and computer processes for the quantitative analysis of micrographs and equivalent subjects’, in *Pictorial pattern recognition*, pp. 275–362.

Moskalenko, V., Zolotykh, N. and Osipov, G. (2019) ‘Deep Learning for ECG Segmentation’, *Studies in Computational Intelligence*, 856, pp. 246–254. doi:10.1007/978-3-030-30425-6_29.

Naini, F.B. (2010) ‘Leslie G. Farkas: pioneer of modern craniofacial anthropometry’, *Archives of facial plastic surgery*, 12(3), pp. 141–142.

Nechala, P., Mahoney, J. and Farkas, L.G. (1999) ‘Digital two-dimensional photogrammetry: a comparison of three techniques of obtaining digital photographs.’, *Plastic and reconstructive surgery*, 103(7), pp. 1819–1825. doi:10.1097/00006534-199906000-00002.

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K. (2016) ‘WaveNet: A Generative Model for Raw Audio’, *arXiv preprint arXiv:1609.03499* [Preprint].

Otsu, N. (1979) ‘A threshold selection method from gray-level histograms.’, *IEEE Trans Syst Man Cybern*, SMC-9(1), pp. 62–66. doi:10.1109/TSMC.1979.4310076.

Ozsoy, U., Demirel, B.M., Yildirim, F.B., Tosun, O. and Sarikcioglu, L. (2009) ‘Method selection in craniofacial measurements: advantages and disadvantages of 3D digitization method.’, *Journal of cranio-maxillo-facial surgery : official publication of the European Association for Cranio-Maxillo-Facial Surgery*, 37(5), pp. 285–290. doi:10.1016/j.jcms.2008.12.005.

Pantic, M., Patras, I. and Rothkruntz, L. (2002) ‘Facial genture recognition in face profile image sequences’, *Proceedings - 2002 IEEE International Conference on Multimedia and Expo, ICME 2002*, 1(March), pp. 37–40. doi:10.1109/ICME.2002.1035712.

Pantic, M., Valstar, M., Rademaker, R. and Maat, L. (2005) ‘Web-based database for facial expression analysis’, in *2005 IEEE proceedings of international conference on Multimedia and Expo*, p. 5. doi:10.1109/ICME.2005.1521424.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A. and Cournapeau, D. (2012) ‘Scikit-learn: Machine Learning in Python’, *Journal of Machine Learning Research*, 12, pp. 2825–2830.

Phillips, P.J., Flynn, P.J., Scruggs, T., Bowyer, K.W., Chang, J., Hoffman, K., Marques, J., Min, J. and Worek, W. (2005) 'Overview of the face recognition grand challenge', in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. IEEE, pp. 947–954. doi:10.1109/CVPR.2005.268.

Phimoltares, S., Lursinsap, C. and Chamnongthai, K. (2007) 'Face detection and facial feature localization without considering the appearance of image context', *Image Vis. Comput*, 25. doi:10.1016/j.imavis.2006.05.017.

Robinette, K.M., Blackwell, S., Daanen, H., Boehmer, M. and Fleming, S. (2002) *Civilian American and European Surface Anthropometry Resource (CAESAR), Final Report. Volume 1. Summary*. SYTRONICS INC DAYTON OH.

Robinette, K.M. and Daanen, H.A.M. (2006) 'Precision of the CAESAR scan-extracted measurements', *Applied Ergonomics*, 37(3), pp. 259–265. doi:10.1016/j.apergo.2005.07.009.

Romeny, B.M.H. (2008) *Front-End Vision and Multi-Scale Image Analysis: multi-scale computer vision theory and applications, written in mathematica (Vol. 27)*. Dordrecht: Springer Science & Business Media. doi:10.1007/978-1-4020-8840-7.

Rosenblatt, F. (1958) 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review*, 65(6), pp. 386–408. doi:10.1037/H0042519.

Rosten, E. and Drummond, T. (2006) 'Machine Learning for High-Speed Corner Detection', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS, pp. 430–443. doi:10.1007/11744023_34.

Ruder, S. (2016) 'An overview of gradient descent optimization algorithms'. doi:10.48550/arxiv.1609.04747.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) 'Learning representations by back-propagating errors', *Nature*, 323(6088), pp. 533–536. doi:10.1038/323533A0.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L. (2015) 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision*, 115(3), pp. 211–252. doi:10.1007/s11263-015-0816-y.

Russell, S.J. and Norvig, P. (2020) *Artificial Intelligence: A Modern Approach*. 4th edn. Pearson Education.

Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S. and Pantic, M. (2016) '300 Faces In-The-Wild Challenge: database and results', *Image and Vision Computing*, 47, pp. 3–18. doi:10.1016/j.imavis.2016.01.002.

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S. and Pantic, M. (2013) '300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge':, *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 397–403. doi:10.1109/ICCVW.2013.59.

Sakoe, H. (1978) 'Dynamic Programming Algorithm Optimization for Spoken Word Recognition', *IEEE transactions on acoustics, speech and signal processing*, 26(1), pp. 43–49. doi:10.1109/TASSP.1978.1163055.

Salkind, N.J. (2007) *Encyclopedia of Measurement and Statistics*. SAGE Publications.

Schmidhuber, J. (2015) 'Deep Learning in neural networks: An overview', *Neural Networks*, 61, pp. 85–117. doi:10.1016/j.neunet.2014.09.003.

Sejnowski, T. and Rosenberg, C.R. (1987) 'Parallel Networks that Learn to Pronounce English Text', *Complex Systems*, 1(1), pp. 165–168.

Shearer, C. (2000) 'The CRISP-DM Model: The New Blueprint for Data Mining', *Journal of Data Mining*, 5, pp. 13–22.

Sherstinsky, A. (2020) 'Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network', *Physica D: Nonlinear Phenomena*, 404. doi:10.1016/J.PHYSD.2019.132306.

Shewalkar, A. (2019) ‘Performance evaluation of deep neural networks applied to speech recognition : RNN, LSTM and GRU’, *Journal of Artificial Intelligence and Soft Computing Research*, 9(Vol. 9, No. 4), pp. 235--245. doi:10.2478/JAISCR-2019-0006.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W. and Woo, W. (2015) ‘Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting’, *Advances in neural information processing systems*, pp. 802–810. doi:10.5555/2969239.2969240.

Silva, A.G., Punt, T.D. and Johnson, M.I. (2010) ‘Reliability and validity of head posture assessment by observation and a four-category scale’, *Manual therapy*, 15(5), pp. 490–495. doi:10.1016/j.math.2010.05.002.

Staudemeyer, R.C. and Morris, E.R. (2019) ‘Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks’, *arXiv preprint arXiv:1909.09586*. [Preprint].

Stratonovich, R.L. (1959) ‘Optimum nonlinear systems which bring about a separation of a signal with constant parameters from noise.’, *Radiofizika*, 2:6, pp. 892–901.

Studer, S., Bui, T.B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S. and Müller, K.-R. (2021) ‘Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology’, *Machine Learning and Knowledge Extraction 2021, Vol. 3, Pages 392-413*, 3(2), pp. 392–413. doi:10.3390/MAKE3020020.

Szeliski, R. (2010) *Computer Vision: Algorithms and Applications*. 2nd edn. Springer Science & Business Media.

Tresadern, P.A., Bhaskar, H., Adeshina, S.A., Taylor, C.J. and Cootes, T.F. (2009) ‘Combining local and global shape models for deformable object matching’, in *Proceedings of British Machine Vision Conference*, pp. 1–12. doi:10.5244/C.23.95.

Viola, P. and Jones, M. (2001) ‘Robust real-time object detection’, *International Journal of Computer Vision*, 4(34–47), p. 4. doi:10.1023/B:VISI.0000013087.49260.fb.

Wang, C.W., Huang, C.T., Hsieh, M.C., Li, C.H., Chang, S.W., Li, W.C., Vandaele, R., Marée, R., Jodogne, S., Geurts, P., Chen, C. and Zheng, G. (2015) 'Evaluation and Comparison of Anatomical Landmark Detection Methods for Cephalometric X-Ray Images: A Grand Challenge', *IEEE Transactions on Medical Imaging*, 34(9), pp. 1890–1900. doi:10.1109/TMI.2015.2412951.

Wang, M. and Deng, W. (2021) 'Deep Face Recognition: A Survey', *Neurocomputing*, 429, pp. 215–244. doi:DOI10.1016/j.neucom.2020.10.081.

Wang, S.L., Lau, W.H., Liew, A.W.C. and Leung, S.H. (2007) 'Robust lip region segmentation for lip images with complex background', *Pattern Recogn*, 40. doi:10.1016/j.patcog.2007.03.016.

Weinberg, S.M., Naidoo, S., Govier, D.P., Martin, R.A., Kane, A.A. and Marazita, M.L. (2006) 'Anthropometric precision and accuracy of digital three-dimensional photogrammetry: comparing the Genex and 3dMD imaging systems with one another and with direct anthropometry', *Journal of Craniofacial Surgery*, 17(3), pp. 477–483. doi:10.1097/00001665-200605000-00015.

Weinberg, S.M., Raffensperger, Z.D., Kesterke, M.J., Heike, C.L., Cunningham, M.L., Hecht, J.T., Kau, C.H., Murray, J.C., Wehby, G.L., Moreno, L.M. and Marazita, M.L. (2016) 'The 3D Facial Norms Database: Part 1. A Web-Based Craniofacial Anthropometric and Image Repository for the Clinical and Research Community', *The Cleft palate-craniofacial journal : official publication of the American Cleft Palate-Craniofacial Association*, 53(6), pp. e185–e197. doi:10.1597/15-199.

Weinberg, S.M., Scott, N.M., Neiswanger, K., Brandon, C.A. and Marazita, M.L. (2004) 'Digital three-dimensional photogrammetry: evaluation of anthropometric precision and accuracy using a Genex 3D camera system', *The Cleft palate-craniofacial journal*, 41(5), pp. 507–518. doi:10.1597/03-066.1.

Williams, C.K.I. (2021) 'The Effect of Class Imbalance on Precision-Recall Curves', *Neural Computation*, 33(4), pp. 853–857. doi:10.1162/neco_a_01362.

Wilson, D.R. and Martinez, T.R. (2003) ‘The general inefficiency of batch training for gradient descent learning’, *Neural Networks*, 16(10), pp. 1429–1451. doi:10.1016/S0893-6080(03)00138-2.

Wirth, R. (2000) ‘CRISP-DM: Towards a standard process model for data mining’, *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, pp. 29--39.

Woods, R.E. and Gonzalez, R.C. (2017) *Digital Image Processing*. 4th edn. Pearson.

Wu, Y. and Ji, Q. (2019) ‘Facial Landmark Detection: a Literature Survey’, *International Journal on Computer Vision*, 127(2), pp. 115–142. doi:10.1007/s11263-018-1097-z.

Yan, P. and Bowyer, K.W. (2007) ‘Biometric recognition using 3D ear shape’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8), pp. 1297–1308. doi:10.1109/TPAMI.2007.1067.

Yi, D., Lei, Z., Liao, S. and Li, S.Z. (2014) ‘Learning Face Representation from Scratch’, *arXiv preprint arXiv:1411.7923* [Preprint].

Yochum, M., Renaud, C. and Jacquir, S. (2016) ‘Automatic detection of P, QRS and T patterns in 12 leads ECG signal based on CWT’, *Biomedical signal processing and control*, (25), pp. 6–52. doi:10.1016/j.bspc.2015.10.011ï.

Youssef, A.R. (2016) ‘Photogrammetric quantification of forward head posture is side dependent in healthy participants and patients with mechanical neck pain’, *International Journal of Physiotherapy*, 3(3), pp. 326–331. doi:10.15621/IJPHY/2016/V3I3/100838.

Zafeiriou, S., Chrysos, G.G., Roussos, A., Ververas, E., Deng, J. and Trigeorgis, G. (2017) ‘The 3D Menpo Facial Landmark Tracking Challenge’, in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2503–2511.

Zafeiriou, S., Trigeorgis, G., Chrysos, G., Deng, J. and Shen, J. (2017) ‘The Menpo Facial Landmark Localisation Challenge: A step towards the solution’, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 170–179.

Zhang, X., Zhao, J. and LeCun, Y. (2015) ‘Character-level Convolutional Networks for Text Classification’, *Advances in Neural Information Processing Systems*, 28.

Zhou, X. and Bhanu, B. (2005) ‘Human Recognition Based on Face Profiles in Video’, in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05) - Workshops*. IEEE, pp. 15–15. doi:10.1109/CVPR.2005.471.

Zhu, X. and Ramanan, D. (2012) ‘Face detection, pose estimation, and landmark localization in the wild’, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2879–2886. doi:10.1109/CVPR.2012.6248014.

Appendix A: Publications

The following paper (Dickers *et al.*, 2021) has been accepted for publication as a direct result of the research discussed in this thesis:

Copyright
image

Copyright
image

Copyright
image

Appendix B: Estimating derivative errors with Taylor's theorem

Equation (B1) shows Taylor's theorem.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{B-1}$$

f is $(n + 1)$ times differentiable and $f^{(n)}(a)$ is the nth derivative of f with respect to x, evaluated at x=a.

If x_i is a point on the curve then Taylor's theorem can approximate $f(x_{i+1})$, where x_{i+1} is another point on the curve separated by a small distance $h = x_{i+1} - x_i$. Substituting into (B-1), letting $x = x_{i+1}$, $a = x_i$, noting that $x_{i+1} = x_i + h$ and then expanding the series gives,

$$f(x_i + h) = f(x_i) + f'(x_i)h + \frac{1}{2}f''(x_i)h^2 + \frac{1}{6}f'''(x_i)h^3 + \frac{1}{n!}f^{(n)}(x_i)h^n + R_n \tag{B-2}$$

where the truncation error, $R_n = \frac{1}{n+1!}f^{(n+1)}(\xi)h^n$ and ξ is a number on the open interval between x_i and x_{i+1}. Truncating the Taylor series above after the 1st derivative gives $f(x_i + h) = f(x_i) + f'(x_i)h + R_1$, where $R_1 = \frac{1}{2}f''(\xi)h^2$

Rearranging and dividing throughout by h gives the exact forward difference, $f'(x_i)h$

$$f'(x_i) = \frac{f(x_i + h) - f(x_i)}{h} - \frac{R_1}{h}$$

Or

$$f'(x_i) = \frac{f(x_i + h) - f(x_i)}{h} + O(h) \tag{B-3}$$

Where *O(h)* is the complexity of the 1st derivative approximation and so is proportional to the step size, h . *Here, halving the step size will halve the error associated with the derivative.*

Similarly the backward difference can be found by noting $h= x_i - x_{i-1}$, $f(x_{i-1})= f(x_i - h)$ and using Taylor's theorem to find $f(x_i - h)$.

$$f(x_i - h) = f(x_i) - f'(x_i)h + \frac{1}{2}f''(x_i)h^2 - \frac{1}{6}f'''(x_i)h^3 + \frac{1}{n!}f^{(n)}(x_i)h^n + R_1 \quad \text{(B-4)}$$

and,

$$f'(x_i) = \frac{f(x_i) - f(x_i - h)}{h} + O(h) \quad \text{(B-5)}$$

Again, the truncation error is also of order *O(h).*

The central difference 1st order derivative is found by and subtracting (B-2) and (B-4) $f(x_i + h) - f(x_i - h) = 2f'(x_i)h + R_2$ and the local truncation error, $R_2 = \frac{1}{6}f'''(\xi_3')h^3$, where ξ_3' is found from the truncation errors of (B-2) and (B-4) using the intermediate value theorem. Dividing by *2h* gives,

$$\frac{f(x_i+h) - f(x_i - h)}{2h} = f'(x_i) + R_2/2h$$

The truncation error, $R_2/2h$ is of the order $O(h^2)$ so halving the step size will make the error associated with the derivative four times smaller.

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad \text{(B-6)}$$

Therefore, the central difference approach to finding the error is more accurate for small step sizes.

Appendix C: Theory underpinning Sequential DNNs

Recurrent Neural Networks

Unlike feedforward networks whose connections are constrained to link forward to neurons deeper in the network, recurrent networks include links feeding back to earlier parts of the network. Interpreting these feedback links as outputs from a previous time step, and chaining these stages together, allows learning across several time steps. Hence the network can learn patterns across time or sequences of data. Figure C-1 shows the overall structure of a single recurrent neuron, typically referred to as a cell. Several of these cells can be combined together to form a more complex cell.

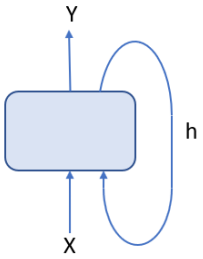


Figure C-1: Structure of a recurrent neural network cell.

Training of a RNN can be achieved by using the idea of "unrolling through time" (Staudemeyer and Morris, 2019; Sherstinsky, 2020) as shown in figure C-2. When seen like this, the back propagation algorithm can be applied in the normal way to calculate gradients through the network and is referred to as back propagation through time (BPTT) in the literature. Hence, from the perspective of the BPTT algorithm, there are two sets of inputs and their associated weights that correspond to both the normal input vector weights and also the weights for the previous time step's output state, typically referred to as h where h refers to the word "hidden" since this state is contained within the network. Y is the output of the cell.

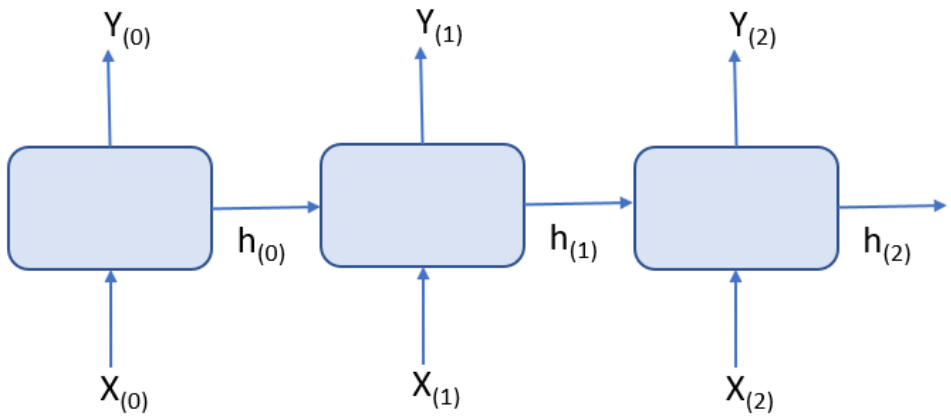


Figure C-2: A recurrent neural network cell unrolled through time.

Typically, RNNs are used in a number of configurations: vector to sequence; sequence to sequence; sequence to vector or as an encoder-decoder, that is sequence to vector followed by a vector to sequence. These architectures and their uses are detailed in several texts (Goodfellow, Ian, Bengio, Yoshua, and Courville, 2016; Géron, 2019) and so their detailed theoretical underpinning is not repeated here, though figure C-2 is an example of a sequence-to-sequence network with the output sequence ($Y_{(0)}$-$Y_{(2)}$) being the same length as the input sequence ($X_{(0)}$-$X_{(2)}$).

Sequence to sequence and sequence to vector RNNs allow an output sequence to be learnt from a labelled dataset of input sequence vectors. The output sequence can be configured to be of a different length to the input sequence or it could be the same length, depending on the problem scenario. Variable lengths can be accommodated by, for example, padding, however the majority of modern neural network libraires include options for variable length sequences (Chollet and Others, 2015; Abadi *et al.*, 2016; Mathworks, 2020a). An aim of this thesis is the segmentation of a contour curve, hence the focus here is on the essential concepts related to a sequence-to-sequence network where the input and output sequences are the same length, although, depending upon the features used the dimension of the input sequence vector may change. For example, a univariate time series or a scalar feature such as curvature would have a dimension of 1, a profile's x and y co-ordinates, on the other hand, would have a dimension of 2. Additionally, the length of each training or test sequence may also vary.

RNNs can be used for segmentation by adding additional layers to the output as is common in other neural network architectures. Typically a fully connected layer

followed by a softmax layer is used to convert the real valued output state to a finite number of classes. The purpose of the softmax layer is to calculate and associate a probability value between zero and one for each of the multiclass outputs. A true/false decision boundary is chosen, usually a probability of 0.5 or greater is regarded as true. It normalises the probability distribution over the output classes, hence the total class probabilities sum to one (Russell and Norvig, 2020).

The output sequence may consist of a sequence of labels classifying each input of the sequence. For example, a single input from a long sequence might be a co-ordinate (*x*,*y*) pair representing a point on a curve, whilst the corresponding output at that time would be a region label, for example, “upper lip”.

RNNs have a limited memory, the literature often states 10 time/sequence steps and attribute this to both vanishing gradients or exploding gradients that cause some oscillation in the gradient magnitudes during back propagation. This led to the development of LSTM RNNs that attempted to address these limitations. LSTMs introduce a memory component that allows internal state to be remembered or forgotten through the use of a gate structure and expands the internal states to include the hidden state h() vector (also referred to as hidden units) and an additional cell state, C() vector. The hidden state corresponds to the short-term memory and the cell state to the long term memory. An LSTM cell is shown in figure C-3.

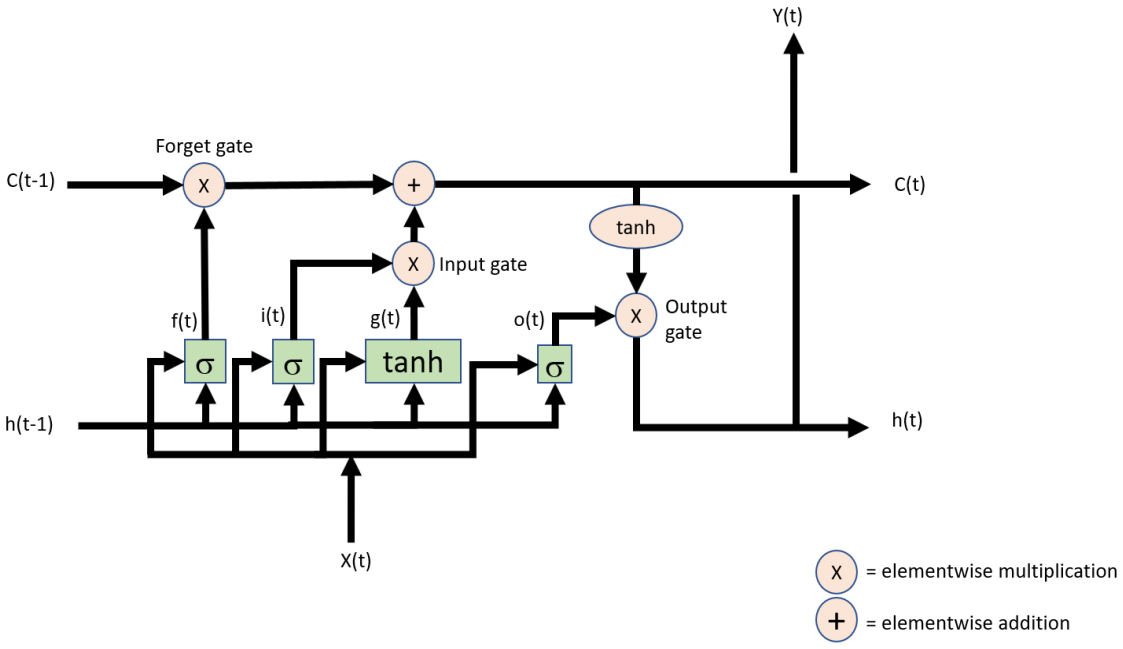


Figure C-3: LSTM Cell.

The functions *i(t), f(t), o(t), g(t), c(t) and y(t)* correspond to the matrix equations (C-1) to (C-6) respectively. Note that both *x(t)* and *y(t)* are weighted and activation functions *σ()* and *tanh()* are applied as in a normal feed forward neural network (FFNN). *σ()* corresponds to the sigmoid activation function and *tanh() to* the hyperbolic activation function. These have the effect of "squashing" the output ranges to ± 1. The $\otimes$ operator defines element-wise vector multiplication (also known as the Hadamard product).

$$\boldsymbol{i}(t) = \sigma(\mathbf{W}_{\boldsymbol{xi}} \cdot \mathbf{x}(t) \; + \; \mathbf{W}_{\boldsymbol{hi}} \cdot \mathbf{h}(t-1) \; + \; \mathbf{b}_i) \tag{C-1}$$

$$\boldsymbol{f}(t) = \sigma(\mathbf{W}_{\boldsymbol{xf}} \cdot \mathbf{x}(t) \; + \; \mathbf{W}_{\boldsymbol{hf}} \cdot \mathbf{h}(t-1) \; + \; \mathbf{b}_f) \tag{C-2}$$

$$\boldsymbol{o}(t) = \sigma(\mathbf{W}_{\boldsymbol{oi}} \cdot \mathbf{x}(t) \; + \; \mathbf{W}_{\boldsymbol{oi}} \cdot \mathbf{h}(t-1) \; + \; \mathbf{b}_o) \tag{C-3}$$

$$\boldsymbol{g}(t) = tanh(\mathbf{W}_{\boldsymbol{xg}} \cdot \mathbf{x}(t) \; + \; \mathbf{W}_{\boldsymbol{hg}} \cdot \mathbf{h}(t-1) \; + \; \mathbf{b}_g) \tag{C-4}$$

$$\boldsymbol{c}(t) = \boldsymbol{f}(t) \otimes \mathbf{c}(t-1) \; + \; \boldsymbol{i}(t) \otimes \; \boldsymbol{g}(t) \tag{C-5}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(t) = \boldsymbol{o}(t) \otimes tanh(\mathbf{c}(t)) \tag{C-6}$$

As can be seen the hidden state, h() corresponds to the output of one cell. Look at equation (C-3) and (C-6) the dimensions of these states depend upon the design choices and represent the cell's output at one point in time or, equivalently, just one sampled sequence value. *i(t), f(t), o(t) and g(t)* are in a form where the backpropagation algorithm can now be applied, and the weight matrices $\mathbf{W}_{**}$ elements learnt in the usual manner. Typically an LSTM can learn relationships spread across hundreds of time steps (Hochreiter and Schmidhuber, 1997) so any design will need to take this into account.

There are several variants of LSTMs, for example the Gated Recurrent Unit (GRU) (Cho *et al.*, 2014), that aims to simplify the architecture of the gate. This is currently a very popular network choice. Staudemeyer and Morris (2019) suggest GRUs outperform LSTMs, however Shewalkar (2019), for instance, presents empirical evidence that indicates they are comparable in speech applications, though the GRU is faster to train.

The next section reviews the use of CNNs in processing sequential and time series data, focusing on 1DTCNNs.

Convolutional neural networks

A two-dimensional CNN layer, as used in image detection, typically uses many small kernels or filters that are moved across the input image, performing convolutions at each pixel. Each kernel's convolutions creates a single two-dimensional feature map as it moves across the image. Together, these feature maps form a single layer. The kernels each encode a particular feature that is representative of part of the image to be recognised. As each kernel has effectively scanned the image looking for its feature, the resulting feature map encodes where that feature (or multiple versions of it) lie within the image. Hence the use of the word "map." Multiple scales are accommodated by adding a pooling layer afterwards to downsize or subsample the layer above. Hence a CNN has the ability to localise an object within an image and at multiple scales. The back propagation algorithm is used to train the kernels. More detailed descriptions of CNNs and their variations are provided in Goodfellow, Ian, Bengio, Yoshua, and Courville (2016) and Géron (2019).

One dimensional neural networks work in a similar manner to the more common 2D CNN. Convolutions are one dimensional in nature and multiple layer subsampling is referred to as dilation, where layers have a dilation rate, for example a dilation factor of 2, samples every other input of the layer above. This is efficient since n dilations allows features of up to length 2^n to be recognised in n layers, should a constant dilation rate of 2 be used, as is common.

The wavenet architecture introduced by Oord *et al.* (2016) applies these ideas and includes a temporal feature that prevents looking ahead. The authors grouped 10 convolutional layers with dilation rates of 1, 2, 4, …,256 and 512. They created three such groups and stacked them on top of each other. They justified this by pointing out that each group represented an efficient implementation of a convolutional layer with kernel size of 1024. Figure C-4(a) (Bai, Kolter and Koltun, 2018) shows the architecture of a 1DTCNN network. Figure C-4(a) shows a dilated causal convolution with dilation factors, d = 1, 2 and 4 and kernel filter size k = 3 (blue lines). Figure C-4(b) shows a residual block that includes two of the dilated causal convolutions of figure C-4(a). A 1x1 convolution is added to ensure the residual input and outputs have the same dimensions.

Figure C-4(c) shows an example residual connection where the dilation factor is one and the kernel filter has a size of three (see blue lines). Classification is achieved with the use of a fully connected output and softmax layer.

Figure C-4(a): Architectural elements of a 1DTCNN network (Bai, Kolter and Koltun, 2018).

Figure C-4(b): Residual block that includes two of the dilated causal convolutions of Figure C-4(a).

Figure C-4(c): An example residual connection where the dilation factor is 1 and the kernel filter has size 3 (see blue lines).

Appendix D: Evaluation metrics in machine learning classification

This appendix provides a review of evaluation metrics commonly used in the evaluation of machine learning classifiers. Typically, a dataset used is split into two parts – a training set and a smaller test set. The test set it placed to one side and will not be used during the training process. The training set may be further split into a training set and a smaller validation set. The training set is used to train the classifier whilst the validation set is used to assess the classifier as training takes place. Use of the validation set can help prevent over-training of the data using the idea of early stopping, a regularisation method that Geoffrey Hinton referred to as a "beautiful free lunch" (Ruder, 2016). Over-training occurs when the model has been trained for too long on the training set and has effectively memorised it. It is then unable to generalise when new data is presented for inference (classifying). The validation set can then detect when overtraining is likely by testing the model on the validation set as training goes on. Note the test dataset is never used for this purpose. Only when the model is ready to be used is the test data applied and evaluation takes place. Without a validation set (for example, in the case where there is a small dataset), manual inspection of the training loss as training progresses is necessary to prevent overtraining.

Once trained the classifier needs to be evaluated. The following sections discuss the most commonly used approaches.

The confusion matrix

A confusion matrix is used to present the results of the performance of a binary classifier but it can also be extended to present results of a multiclass classifier. There is no set standard for organizing the layout of the matrix so care should be taken when interpreting a confusion matrix.

In the confusion matrix of figure D-1, the rows contain instances of the correct expected results. The top row contains instances that should have been classified as true. The total number of the instances in this row is denoted by P, for positive. The top left cell contains a value representing the number of instances that were correctly predicted by the classifier, and these are known as the true positives (TP). The top right cell contains a

value representing the number of actual, true instances that were misclassified as false by the classifier. They are called false negatives (FN). Hence P =TP + FN.

Similarly, the second row also contains totals of actual observations that should have been classified as negative. The total number of these in this row is denoted by N for negative. The bottom left cell contains false positives (FP), these should have been classified as negative but were wrongly classified as positive. The bottom right contains a value representing the number of instances that were correctly classified as negative and are referred to as true negatives (TN). Hence N= TN + FP.

Note the true positives and true negatives are always found in the leading diagonal of the matrix.

	Classifier Predicted: YES	**Classifier Predicted: NO**	$\boldsymbol{FP\ rate = \frac{FP}{N}}$	$\boldsymbol{TP\ rate = \frac{TP}{P}}$
Actual: YES (observation)	**True Positives (TP) (correct result)**	**False Negatives (FN) (unexpected result)**	$\boldsymbol{Precision = \frac{TP}{TP + FP}}$	$\boldsymbol{Recall = \frac{TP}{TP + FN}}$
Actual: NO (observation)	**False Positives (FP) (unexpected result)**	**True Negatives (TN) (correct result)**	$\boldsymbol{Accuracy = \frac{TP + TN}{P + N}}$	$\boldsymbol{F1 - score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}}$

Figure D-1: Binary classifier confusion matrix (see following sections for definitions of equations).

The terms "positive" and "negative" refer to the classifier's prediction. The terms "true" and "false" refer to whether the prediction corresponds to the actual observation- what it really is.

The total number of actual, expected results for a class is called the support. In figure D-1 the Yes class' (top row) support is given by P= TP+FN. The No class's support is given by N = TN+FP. Some useful metrics can now be calculated from the confusion matrix.

Accuracy

The overall accuracy of the classifier can be determined by first finding the total number of instances correctly classified by the classifier and then dividing by the total number of instances in the dataset, both true and false. That is,

$$Accuracy = \frac{TP + TN}{P + N} \tag{D-1}$$

Accuracy is often used to give an overview of a classifier's performance but can be misleading. For example, if a dataset was heavily skewed (misbalanced) with 95 true instances and 5 false then a bad classifier that classifies all results as a positive class would achieve 95% accuracy. Precision, recall (also known as the true positive rate), and the false positive rate are metrics that aim to avoid this problem.

False positive rate

The false positive rate is defined in equation (D-2). In the previous example the classifier will be revealed as a poor one when the false positive rate is calculated. This figure gives an idea of often the classifier incorrectly classifies a true instance – 100% of the time in our previous example.

$$False\ Positive\ Rate = \frac{FP}{TN + FP} \tag{D-2}$$

Recall

Recall is defined by equation (D-3). It is also known as the true positive rate and is the complement of the false positive rate. It gives a sense of how often the classifier was right when classifying genuinely positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{D-3}$$

Precision

Precision is defined in equation (D-4). It aims to explain accurately how often the classifier correctly classifies genuinely true instances, that is how many of the positive predictions are correct.

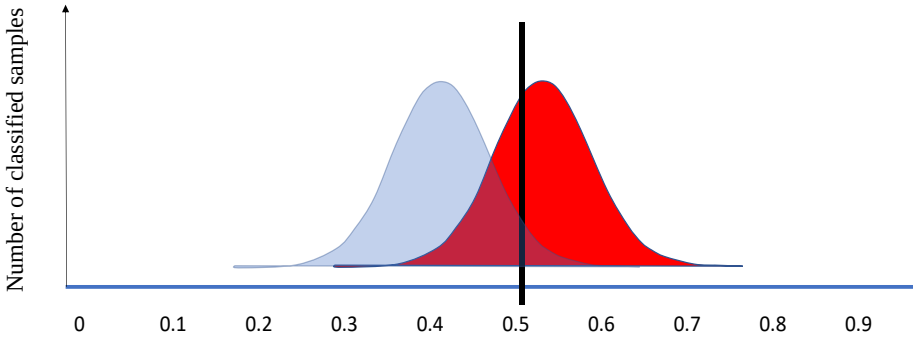
$$Precision = \frac{TP}{TP + FP} \tag{D-4}$$

Using any of the above metrics alone can be misleading. Often precision and recall are used together as an increase in precision can cause a decrease in recall and vice versa. To see this, the diagram in figure D-2 below shows the distribution of a classifier's positive and negative predictions together with a plane (here a vertical line) representing a decision boundary.

The classifier's positive predictions are represented by the red distribution and its negative predictions by the blue distribution. Blue instances to the left of the decision boundary are correctly classified, they are true negatives (TN), whilst the remaining blue instances to the right of the boundary are incorrectly classified. These are false positives (FP). Similarly, the red instances (positive predictions) to the right of the decision boundary are correctly classified, they are true positives (TP), whilst the remaining red instances to the left of the boundary are incorrectly classified. These are false negatives (FN).

Moving the decision boundary to the left will increase the number of TP but decrease the number of FN. That is, the recall will increase. However, moving the boundary like this will also increase the number of FP and reduce the precision. It would be better to improve the classifier, if possible, by instead reducing the overlap of distributions. With a clear gap between distributions the classifier would have a 100% precision and recall.

Distribution of positive and negative instances predicted by a classifier



Key: *Blue distribution = true negatives.*
Red distribution = true positives.
Vertical line = decision boundary.

Figure D-2: Distribution of positive and negative instances predicted by a binary classifier.

The F1 Score

The F1 score combines the recall and precision values into one average value. The average is the harmonic mean which is used when averaging values representing ratios such as speed for example and, in this case precision and recall. The harmonic mean of the two values of recall and precision is given in equation (D-5). Williams (2021) notes that precision and recall, and hence F1 scores are dependent upon class balance and hence where comparisons are made across different scenarios or datasets then class ratios should be maintained.

$$F1\ score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{D-5}$$

Multi-class confusion matrix

The multi-class confusion matrix extends the binary confusion matrix to work with several classes. TP, TN, FP and FN values exist for each class as before. Given a class, for example, class A, shown in figure D-3, the metrics described above are calculated by labelling the TP,TN,FP and FN as shown in the confusion matrix. So here, classes that are not class A are all considered to be the complement of A, that is, not A. The support for each class is calculated by summing the entries horizontally for each row. Williams (2021) notes that use of the F1 score to compare different models with varying inter-model class support can be problematic. When using F1 scores ratios of class support size should be consistent between models (as is the case with the dataset used in this thesis).

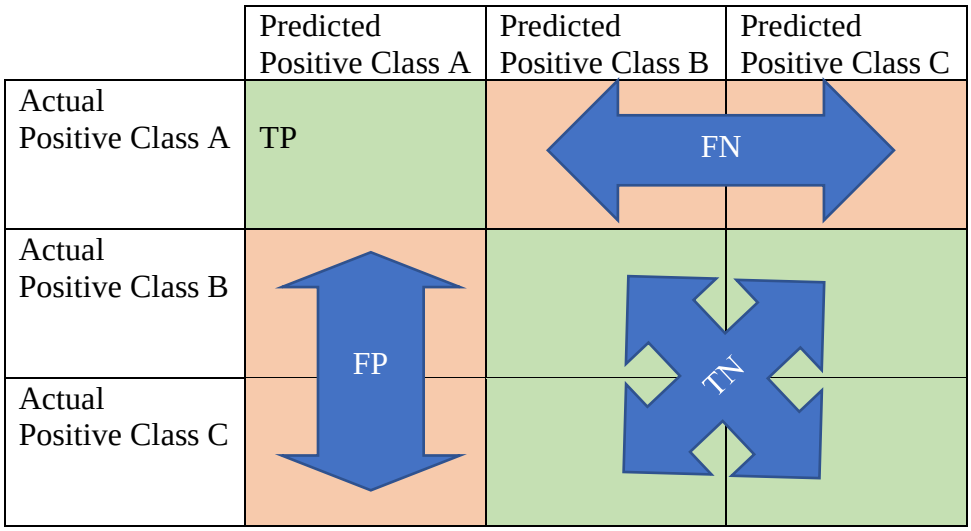


Figure D-3: Multiclass confusion matrix.

Macro F1 score

F1 scores for each class of a multiclass classifier can be calculated in the manner described above, but it is also possible with multi-class models to assign an overall F1 score for the classifier by calculating the average of the F1 scores. Here all classes are given equal weighting. This is useful if the classes are imbalanced as each is given equal representation. A weighted average F1 score could also be calculated to provide additional weighting to a given class based upon its size.

Appendix E: Application for Ethical Approval Form Summary

Ethical approval for this research has been granted by the University's Ethics Committee. Summary sections of the approved application are documented on the following pages with personal information redacted.

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

PG2 / E1 FORM

APPLICATION FOR ETHICAL APPROVAL

In order for research to result in benefit and minimise risk of harm, it must be conducted ethically. A researcher may not be covered by the University's insurance if ethical approval has not been obtained prior to commencement.

The University follows the OECD Frascati manual definition of **research activity**: "creative work undertaken on a systematic basis in order to increase the stock of knowledge, including knowledge of man, culture and society, and the use of this stock of knowledge to devise new applications". As such this covers activities undertaken by members of staff, postgraduate research students, and both taught postgraduate and undergraduate students working on dissertations/projects.

The individual undertaking the research activity is known as the "principal researcher".

Ethical approval is not required for routine audits, performance reviews, quality assurance studies, testing within normal educational requirements, and literary or artistic criticism.

Please read the notes for guidance before completing ALL sections of the form.

This form must be completed and approved prior to undertaking any research activity. Please see Checklist for details of process for different categories of application.

Delete the Guidance Notes at the end of the form BEFORE submitting your application

SECTION A: About You (Principal Researcher)

Full Name:		Gordon Alexander Dickers			
Tick all boxes which apply:		Member of staff:	☒	Honorary research fellow:	☐
Undergraduate Student	☐	Taught Postgraduate Student	☐	Postgraduate Research Student	☒

Institute/Academic Discipline/Centre:	WISA
Campus:	Swansea, SA1
E-mail address:	Gordon.dickers@uwtsd.ac.uk
Contact Telephone Number:	[redacted]
For students:	
Student Number:	[redacted]
Programme of Study:	PhD, Research
Director of Studies/Supervisor:	Professor John Rees

SECTION B: Approval for Research Activity

Has the research activity received approval in principle? (please check the Guidance Notes as to the appropriate approval process for different levels of research by different categories of individual)		YES	☒	NO ☐
				Date
If Yes, please indicate source of approval (and date where known): ***Approval in principle must be obtained from the relevant source prior to seeking ethical approval***	Research Degrees Committee		☒	
	Institute Research Committee		☐	
	Other (write in)		☐	

SECTION K: Declaration

The information which I have provided is correct and complete to the best of my knowledge. I have attempted to identify any risks and issues related to the research activity and acknowledge my obligations and the rights of the participants. In submitting this application I hereby confirm that I undertake to ensure that the above named research activity will meet the University's Research Ethics and Integrity Code of Practice which is published on the website: https://www.uwtsd.ac.uk/research/research-ethics/		
Signature of applicant:	*Gordon Dickens*	**Date:** **20/4/2020**

For STUDENT Submissions:

Director of Studies/Supervisor:		**Date:**
Signature:		

For STAFF Submissions:

Academic Director/ Assistant Dean:	Kapilan Radhakrishnan	**Date:** **20/4/2020**
Signature:		