Imperial College London

Imperial College London

Department of Computing

Lip Reading in 3D

Author:
Peter Robertson

Supervisor:
Dr Stefanos Zafeiriou

Submitted in partial fulfilment of the requirements for the MSc degree in Computing Science / Machine Learning of Imperial College London

September 2019

Abstract

Lip reading is a skill which is renowned for being challenging for humans, with few individuals being capable of implementing the skill successfully. Recent successes have been made in training Machine Learning models to predict speech from video inputs which have been able to outperform human level accuracy. Currently however, no models exist which have attempted to use 3D data of facial scans for the use of lip reading. In this report lip reading is successfully implemented with the use of 3D facial scans to make predictions on a word level with above 50% mean test accuracy on 500 labels, forming the baseline for future work to build upon. The current lack of datasets suitable for this research is the dominant problem in this field of research. In addition to the task of lip reading from 3D facial scans, solving this secondary problem is also attempted. To resolve this issue, the use of Generative Adversarial Networks is investigated as a means of generating 3D facial data corresponding to an audio input.

Acknowledgements

I would like to thank the support of my supervisor; Dr Stefanos Zafeiriou, for his words of advice and guidance in this project.

Secondly I would like to thank the researchers for assisting with support, resources and advice; Athanasios Papaioannou, Evangelos Ververas and Panagiotis Tzirakis.

Contents

1 Introduction **1**

2 3D Facial Modelling Background **3**
2.1 Data Collection and Representation 3
2.1.1 Blendshapes 4
2.2 Facial Mesh Alignment 6
2.2.1 Procrustes Analysis 6
2.2.2 Procrustes Analysis for Facial Mesh Alignment 9
2.3 Chapter Summary 10

3 Deep Learning Background **11**
3.1 Convolutional Neural Networks 11
3.1.1 Convolutional Layers 11
3.1.2 Layer Output Sizes 12
3.2 Activation Functions 12
3.2.1 Sigmoid 13
3.2.2 Tanh 14
3.2.3 ReLU 14
3.2.4 Softmax 14
3.3 Optimisation Algorithms 15
3.3.1 Stochastic Gradient Descent 16
3.3.2 Momentum 17
3.3.3 RMSprop 17

4 Literature Review **19**
4.1 Current Lip Reading Models and Datasets 19
4.1.1 Datasets 19
4.1.2 Lip Reading Models 23
4.2 Data Generation 24
4.2.1 Audio Driven Data Generation 25
4.2.2 Generative Adversarial Networks 26
4.3 Summary of Related Literature 30

5 Blendshape Classification **31**
5.1 Problem Definition 31
5.2 Generation of a 3D Lip Reading Dataset 32

5.2.1 Blendshape Axes Creation . . . 33
5.2.2 Blendshape Parameter Recovery . . . 34
5.2.3 Dataset Split . . . 35
5.3 Data Preprocessing . . . 35
5.4 Assessment of Model Performance . . . 37
5.4.1 Accuracy . . . 37
5.5 Model Architecture . . . 37
5.5.1 Multiple Towers Classification Model . . . 37
5.5.2 Blendshape Channels Classification Model . . . 39
5.6 Discussion of Results . . . 41
5.7 Conclusions . . . 43

6 Audio Driven Blendshape GAN **45**
6.1 Problem Definition . . . 45
6.2 Model Inputs . . . 46
6.2.1 Mel-frequency Cepstral Coefficients . . . 46
6.2.2 Blendshape Inputs . . . 48
6.2.3 Noise Input . . . 49
6.3 Data Preprocessing . . . 49
6.4 Assessment of Model Performance . . . 49
6.5 Model Architecture . . . 50
6.5.1 Model Structure . . . 50
6.5.2 Generator . . . 52
6.5.3 Critic . . . 53
6.6 Discussion of Results . . . 54
6.7 Conclusions . . . 58

7 Conclusions and Future Work **60**
7.1 Blendshape Classification . . . 60
7.1.1 Future Work . . . 60
7.2 Generative Models . . . 61
7.2.1 Future Work . . . 61

Bibliography **68**

List of Figures

2.1 Point Cloud Representation of FLAME Model [1] 4
2.3 Procrustes Alignment Applied to VOCASET [2] 9

3.1 Activation Functions 13
3.2 Negative Log Likelihood Loss 15
3.3 Learning Rates 16
3.4 SGD with Momentum [3] 17
3.5 Incentive Behind RMSprop [3] 18

4.1 LRW Dataset Pipeline [4] 20
4.2 LRW Face Detection and Tracking [4] 21
4.3 LRW Dataset Sample: 24 frames of a subject saying *'about'*, figure from [4] 21
4.4 VOCASET Example Frames [2] 22
4.5 LRW Multiple Towers Model [4] 23
4.6 Audio Driven Facial Animation Model Proposed by Karras et al. [5] . . 25
4.7 The VOCA Model [2] 26
4.8 Goodfellow's Generator Loss Plots [6] 27

5.1 Audio Sample from LRW [4] "Talking About Fair" 33
5.2 First Blendshape Axis Interpolation 34
5.3 Second Blendshape Axis Interpolation 34
5.4 FLAME Template Mesh [1] 36
5.5 Blendshape Parameter Histograms 36
5.6 Multiple Towers Classification Architecture 39
5.7 Blendshape Channels Classification Architecture 40
5.8 Multiple Towers Training 41
5.9 Blendshape Channels Training 42
5.10 Model Training Comparison 42
5.11 Per Word Test Accuracy 43
5.12 1st Percentile Per Word Test Accuracy 43

6.1 Audio Sample Frames 47
6.2 Mel Filterbank Processing 47
6.3 Discrete Cosine Transformation Processing 48
6.4 MFCC Model Input 48

6.5 GAN Model 51
6.6 Generator Model Architecture 52
6.7 Critic Architecture 53
6.8 Critic Gradient Penalty Training Loss 54
6.9 Critic Total Training Losses 55
6.10 Training Losses from Critic for Real and Fake Data Samples 56
6.11 Generator Training Losses 56
6.12 Classification Class Accuracy for Generated Data 57
6.13 Classification Models Trained on Generated Data 58

7.1 Third Blendshape Axis Interpolation 66
7.2 Fourth Blendshape Axis Interpolation 66
7.3 Generated Sample - “AMERICA” 67

List of Tables

5.1 Multiple Towers Model Architecture 39
5.2 Multiple Towers Model Hyperparameters 39
5.3 Blendshape Channels Model Architecture 40
5.4 Blendshape Channels Model Hyperparameters 41
5.5 Blendshape Classification Models Test Data Accuracy 41

6.1 GAN Hyperparameters 52
6.2 Generator Model Architecture 53
6.3 Critic Model Architecture 54

Chapter 1

Introduction

Visual Speech Recognition (VSR), often referred to as lip reading, is a task which is very difficult for humans. Primarily communication is achieved though the sound of spoken audio rather than the motion of the mouth, while the mouth allows for some cues it requires a large amount of practice to make accurate predictions of the words spoken. However, various methods of addressing the problem exist which focus on training Deep Learning models given 2D temporal data of subjects speaking with either whole video samples labelled to contain a single word [4], or labelled at a character level with the corresponding frames of the video [7, 8, 9].

There currently do not exist any models which attempt to address this problem with the use of 3D temporal datasets made up of head scans of subjects. As all previous models have made use of video data, the subjects are recorded from a single point of view with no depth data, which may be of use to Machine Learning models. Currently there exists a lack in such datasets, this is partly due to the complexity and hardware requirements in capturing such data. With the increasing rise in availability of depth cameras, this situation may change within the short term future, although the use of these devices shall not be discussed in the report.

While there do exist a small number of 3D temporal datasets, LRW-3D [10] and the VOCASET [2], neither of these have been constructed with the intention of lip reading, but for the use of facial animation driven by an audio input. To first assess whether this data representation can be used for lip reading, firstly new datasets which capture 3D temporal models of subjects' heads speaking must be established by the community. Ideally this data would be captured directly, however due to the current barriers to acquiring such data, other areas of synthetic data generation can be explored to see if they can yield results which resemble directly captured data close enough to be used to further explore the limitations and uses of such 3D temporal data. One area of synthetic data generation which has seen recent success is the use of Generative Adversarial Networks (GANs), which allow unseen data samples to be produced from an unknown distribution representing the real data samples.

Without access to an appropriate dataset, this report shall discuss the acquisition of a 3D temporal dataset consisting of data labelled at a word level with the use

of the pretrained VOCA model [2] and how this data is processed with the use of statistical Principal Component Analysis before being input into Machine Learning models. Firstly the use of 3D temporal data as a means of lip reading is successfully investigated through the use of two architecture models, finding that there is in fact a strong correlation between the 3D facial motion of a subject and the words spoken. Secondly, the use of a GAN as a means of generating further 3D temporal data driven by an audio input signal is explored. The aim of this to see if the same facial motion features which can be detected by the lip reading models can be learnt implicitly by a Generative GAN based approach. Finally, the implications of these findings and any future work which can be carried out as an extension of this research are discussed.

The potential applications for the research undertaken in this report are primarily to better the currently existing means of speech detection methods. In the vast majority these are performed on audio speech data, however they perform with reduced accuracy under noisy conditions, one such example would be driving a car. In this environment, the noise within the vehicle is high and cannot be reduced due to the safety concerns of removing audio information which may be of use to the driver. Speech recognition systems would be desirable in such a situation as it would reduce the likelihood of attention being drawn away from the task of driving to alter the instruments of the car. By combining the use of the audio signal with a depth camera placed in front of the driver, the accuracy of speech recognition could be enhanced with the use of sensor fusion from predictions from both the audio recognition and a visual recognition systems. This application can be applied in any scenario in which an audio signal contains significant noise or is unobtainable.

Chapter 2

3D Facial Modelling Background

This chapter shall discuss the preliminary background theory on 3D Facial Modelling and how this data is processed and represented for use with Deep Learning models. As the data captured by 3D facial scans is made of a high dimensional point cloud, each point having an position in the x, y and z dimension, a lower dimensional representation of the data can be found which retains the variance in the data. This can be achieved with the use of Blendshape Axes, found with Principal Component Analysis (PCA). In order to remove unwanted variance in the data from facial movement not correlated with speech, such as head movement, the data samples should first be brought into correspondence (also known as alignment or registration) to remove these undesirable variations in the data. This is achieved with the use of Procrustes Analysis.

2.1 Data Collection and Representation

Unlike 2D video where the subject is captured from a single angle without any depth information, 3D models cannot be captured with a single camera. In most instances, 3D head scans are captured with the use of a multi-camera rig with multiple cameras capturing video simultaneously. The number and type of cameras can vary from system to system, but the principle remains consistent. All cameras must be synchronised to capture footage at the same time, with the same frame rate. This is inherently a more complex system than 2D video capture as is therefore more expensive, limiting the accessibility of such systems. Once the images have been captured, the footage must be combined in order to produce a 3D model of the subject's head [1], often referred to as a facial mesh. A facial mesh is made up of a point cloud of vertices and vertex connections. Depending on the resolution of the system used to capture the data, the number of points which are used to represent the subject can vary, each with x, y and z coordinates in space. Depending on the mesh capture pipeline, the number of vertices may vary, but even a low resolution mesh may contain several thousand vertices. In the example shown in Figure 2.1, the FLAME model [1] uses around 5000 vertex positions, each with an (x, y, z) value, resulting in a total of 15,000 parameters for a single capture in time. While Deep Learning models have been trained to drive

all vertex positions from a starting template mesh [5], a simpler representation can be achieved with the use of Blendshapes.

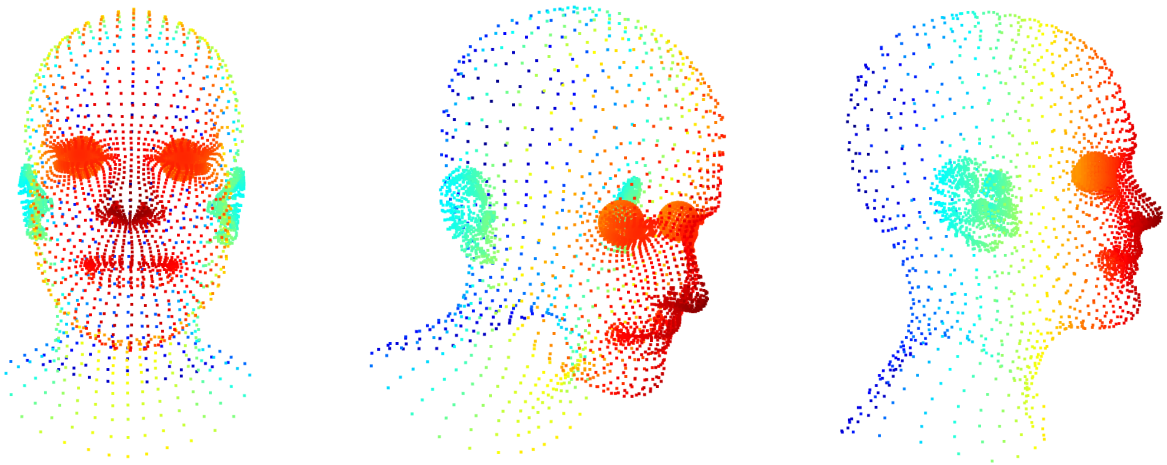


Figure 2.1: Point Cloud Representation of FLAME Model [1]

2.1.1 Blendshapes

Capturing realistic facial motion by modelling the displacements of all of these vertices is impractical when manipulating a model by hand and may hinder the performance of a Machine Learning model due to the high dimensionality of the data. Blendshapes attempt to model aspects of realistic facial motion by finding the relations between these vertices and manipulating them simultaneously [11]; by reducing the number of parameters, the model becomes more manageable to control. One method of producing model Blendshapes is with the use of PCA. By applying PCA on a dataset of facial meshes, the resulting principle components represent the changes in facial motion which capture the largest amount of variation in the set whilst reducing the number of model parameters substantially.

Principal Component Analysis

Given a collection of measured data points in a high dimensional space, it is often desirable to reduce the dimensionality of such data to a lower dimensional latent space, in order to aid processing the data or enable visualisation. In such instances, it is desirable for the mapping to the latent space to maintain as much information from the original data as possible.

PCA aims to reduce the dimensionality of a collection of data points while maximising the variance in the latent space. Alternatively, PCA aims to find a mapping from the original data to a new latent space which allow for the original data to be reconstructed with minimal error. These two aims are in fact equivalent.

Let $\boldsymbol{x_i} \in \mathbb{R}^f$ represent a data point in f dimensional space and $\boldsymbol{y_i} \in \mathbb{R}^d$ represent the point which $\boldsymbol{x_i}$ is transformed to by a linear transformation $\boldsymbol{W}$ equation (2.1), where $d << f$.

$$\boldsymbol{y_i} = \boldsymbol{W}^\top \boldsymbol{x_i} \tag{2.1}$$

where,

$$\boldsymbol{y_i} = \begin{bmatrix} y_{i1} \\ \vdots \\ y_{id} \end{bmatrix}, \quad \boldsymbol{x_i} = \begin{bmatrix} \boldsymbol{x}_{i1} \\ \vdots \\ \boldsymbol{x}_{if} \end{bmatrix}, \quad \boldsymbol{W} = [\boldsymbol{w_1}, \ldots, \boldsymbol{w_d}], \quad \boldsymbol{w_i} = \begin{bmatrix} w_{i1} \\ \vdots \\ w_{if} \end{bmatrix}$$

The optimal transformation $\boldsymbol{W}$ will maximise the variance in the data, where variance is expressed by equation (2.2). This then follows that the optimal solution is found by maximising equation (2.4). To prevent the trivial solution where $\boldsymbol{w}_k = \infty$, constrain a fixed magnitude $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}$.

$$\sigma_y^2 = \frac{1}{N} \sum_{i=1}^{N} (y_{ik} - \mu_k)^2 \tag{2.2}$$

where,

$$\mu_k = \frac{1}{N} \sum_{i=1}^{N} y_{ik}$$

$$\begin{aligned} \boldsymbol{W} &= \arg\max_{\boldsymbol{W}} \frac{1}{N} \sum_{k=1}^{d} \sum_{i=1}^{N} (y_{ik} - \mu_k)^2 \\ &= \arg\max_{\boldsymbol{W}} \frac{1}{N} \sum_{k=1}^{d} \sum_{i=1}^{N} \boldsymbol{w}_k^\top (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top \boldsymbol{w}_k \\ &= \arg\max_{\boldsymbol{W}} \sum_{k=1}^{d} \boldsymbol{w}_k^\top \boldsymbol{S}_t \boldsymbol{w}_k \end{aligned}$$

where $\boldsymbol{S}_t$ is the covariance matrix,

$$\boldsymbol{S}_t = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top \tag{2.3}$$

and,

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i$$

$$\boldsymbol{W} = \arg\max_{\boldsymbol{W}} \operatorname{tr}[\boldsymbol{W}^\top \boldsymbol{S}_t \boldsymbol{W}], \quad \text{subject to} \quad \boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I} \tag{2.4}$$

By constructing the Lagrangian from equation (2.4) and solving, the solution in equation (2.5) can be obtained. From eigendecomposition, $\boldsymbol{W}$ has columns of eigenvectors which correspond to the d largest non-zero eigenvalues of the covariance matrix, $\boldsymbol{S}_t$.

$$L(\boldsymbol{W}, \boldsymbol{\Lambda}) = \operatorname{tr}[\boldsymbol{W}^\top \boldsymbol{S}_t \boldsymbol{W}] - \operatorname{tr}[\Lambda(\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{I})]$$

$$\frac{\partial \operatorname{tr}[\boldsymbol{W}^\top \boldsymbol{S}_t \boldsymbol{W}]}{\partial \boldsymbol{W}} = 2\boldsymbol{S}_t \boldsymbol{W}, \quad \frac{\partial \operatorname{tr}[\Lambda(\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{I})]}{\partial \boldsymbol{W}} = 2\boldsymbol{W}\boldsymbol{\Lambda}$$

$$L(\boldsymbol{W}, \boldsymbol{\Lambda}) = 0$$

$$\boldsymbol{S}_t \boldsymbol{W} = \boldsymbol{W}\boldsymbol{\Lambda} \tag{2.5}$$

2.2 Facial Mesh Alignment

In order to create Blendshapes from 3D facial scans, movement within scans should be limited to facial movement with as little head movement as possible. If head movement remains within the scans, then this will be reflected in the principal axis produced by PCA and as head motion is independent from speech, this is undesirable. Initial data capture can aim to minimise subject head movement, however this cannot be completely eliminated. To remove head motion from the dataset, the scans can be brought into alignment relative to landmark positions on the face which do not move during speech. Such landmarks include the nose, corners of the eyes and cheek bones, as these points will be stationary during speech. By aligning the meshes based on the variation in these points, the head position of the subjects can be constrained while the mouths are not. To achieve this, Procrustes Analysis can be used.

2.2.1 Procrustes Analysis

Procrustes Analysis is a statistical shape analysis method used to analyse the differences between two objects [12]. Given matrices $\boldsymbol{X} \in \mathbb{R}^{(n \times f)}$ and $\boldsymbol{Y} \in \mathbb{R}^{(n \times f)}$, each representing the coordinates of n points in an f-dimensional feature space, a common statistical difference metric is the sum of square differences (2.6).

$$D = \sum_{i=1}^{n} \sum_{j=1}^{f} (x_{ij} - y_{ij})^2 \tag{2.6}$$

However, two objects which are mathematically similar can still have a significant difference when they are scaled differently and are in different positions and orientations in space. In order to compare two objects regardless of these factors, the differences in orientation, scale and spacial position of the objects must be minimised by bringing the objects into optimal alignment.

Translational Alignment

Translational components can be eliminated by having the centroids of the two objects lie at the same position in space. This can be easily achieved by subtracting the mean of each object's points from itself such that the centroid now lies at the origin.

Let $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ and $\bar{y}_j = \frac{1}{n}\sum_{i=1}^{n} y_{ij}$, where $(j = 1, \ldots, f)$ represent the mean of each feature, such that the centroids of the two objects are given by $C_X = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_f)$ and $C_Y = (\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_f)$ respectively. By subtracting the mean of $\boldsymbol{X}$ and $\boldsymbol{Y}$ from themselves, the centroids C_X and C_Y will be in alignment at the origin, minimising the sum of square differences due to translational components.

Scaling Alignment

Similarly, scaling components can be removed by rescaling the objects by the root mean square distance (2.7), such that the root mean square distance will be of unit distance for both.

$$D = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{f} x_{ij}^2} \tag{2.7}$$

Rotational Alignment

In order to align the objects to the same orientation, a rotational matrix which is able to map $\boldsymbol{X}$ to $\boldsymbol{Y}$ as closely as possible must be found. This is described as the Orthogonal Procrustes problem and can be solved with Singular Value Decomposition (SVD). Given matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the orthogonal matrix $\boldsymbol{R}$ is the matrix which matches $\boldsymbol{A}$ to $\boldsymbol{B}$ as closely as possible, as described by equation (2.8), where $\|\cdot\|_F$ is the Fobius norm.

$$\boldsymbol{R} = \arg\min_{\boldsymbol{\Omega}} \|\boldsymbol{\Omega A} - \boldsymbol{B}\|_F \quad \text{subject to} \quad \boldsymbol{\Omega}^\top \boldsymbol{\Omega} = \boldsymbol{I} \tag{2.8}$$

The product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be decomposed (2.10), and then the rotational matrix $\boldsymbol{R}$ can be expressed (2.11).

$$\boldsymbol{M} = \boldsymbol{B}\boldsymbol{A}^\top \tag{2.9}$$

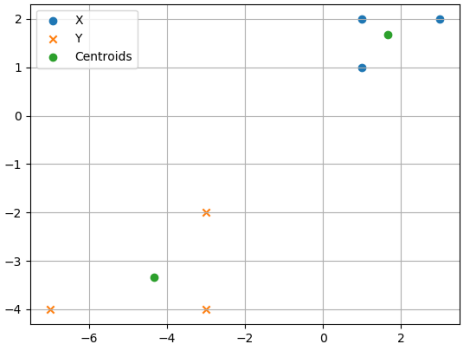
$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{2.10}$$

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^\top \tag{2.11}$$

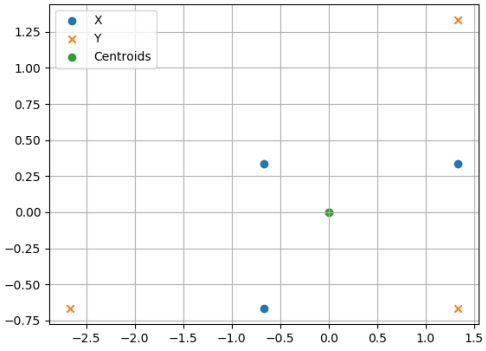
A Simple Application of Procrustes Analysis

This section follows the steps of object alignment using Procrustes Analysis as described above with a simple example. A pair of configurations are given by the matrices below, visualised in Figure 2.2a.

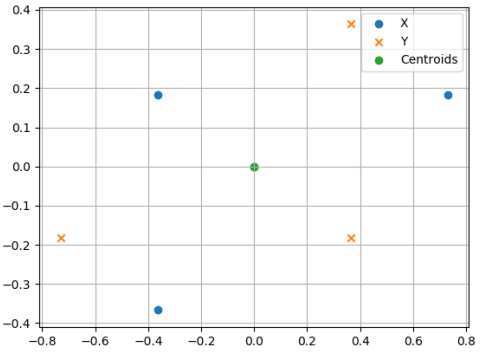
$$\boldsymbol{X} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 3 & 2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{Y} \begin{pmatrix} -3 & -2 \\ -3 & -4 \\ -7 & -4 \end{pmatrix}$$



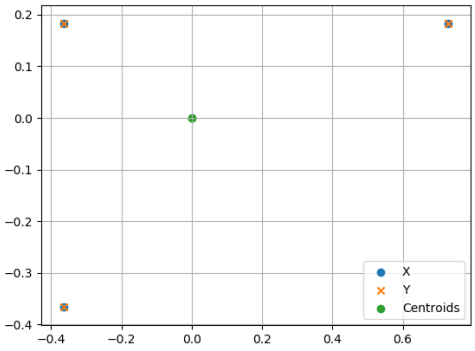
(a) Original coordinates



(b) Translation



(c) Scaling



(d) Rotation

Figure 2.2: Application of Procrustes Analysis

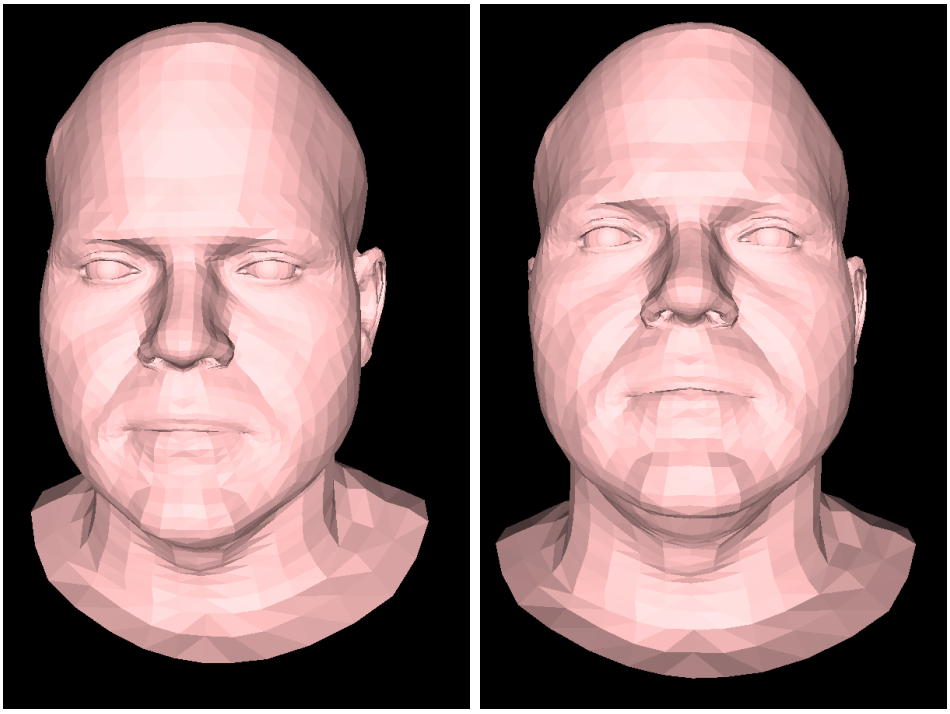
Before applying alignment, the sum of square differences between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is

$$\begin{aligned} D &= (1+3)^2 + (1+2)^2 + (1+3)^2 + (2+4)^2 + (3+7)^2 + (2+4)^2 \\ &= 213 \end{aligned}$$

As described in section (2.2.1), the initial objects are translated so that their centroids lie at the origin (Figure 2.2b). This is then followed by scaling alignment and finally rotational alignment (Figures 2.2c, 2.2d). After alignment, the difference is zero as these objects are both similar triangles.

2.2.2 Procrustes Analysis for Facial Mesh Alignment

The process of Procrustes Analysis described above deals with objects in which all co-ordinate points are used to find the optimum alignment, however this is inappropriate for facial mesh alignment as not only would the position of the head be aligned to, but so would the motion from speech, of which the aim is to maintain variation. However, a subset of points of a pair of objects can be used to align the entire object. By selecting points which do not contain any movement due to speech or facial expression, variation in these points can be assumed to be only from head motion. Such positions include the nose, corners of the eyes and cheekbones. If the facial scans captures 360 degrees, then points on the back of the subject's head can also be used. From this subset of points, the translation, scaling and rotational matrix can be found which aligns these points using the steps described in section 2.2.1 but then applied to the entire object. An example of such an application is shown in Figure 2.3 before and after alignment has been applied.



(a) Before Alignment (b) After Alignment

Figure 2.3: Procrustes Alignment Applied to VOCASET [2]

2.3 Chapter Summary

This chapter has provided an overview of the process of PCA and how the calculated principal axes from this analysis can be used as Blendshape Axes, along which a facial model can be interpolated to describe variation in the dataset. Secondly, an overview of the background mathematics used to align a dataset made up of 3D facial scans can be brought into correspondence such that all data samples are scaled to the same size at the same position and are aligned to a collection of points which should remain fixed throughout the data. This allows for remaining variation in the data to be present in only the desired attributes, such as movement in subjects mouth while removing variation in head movement.

Chapter 3

Deep Learning Background

This chapter shall discuss the Machine Learning methods and techniques used throughout the practical portion of this report. This includes a high level overview of Convolutional Neural Networks, Activation Functions, Loss Functions and Optimisation Algorithms applied.

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a Machine Learning model architecture which uses a filter, or kernel, convolved over the input to extract features relevant to the given kernel from a small area of the input. Multiple kernels are used at each layer of the model to extract multiple features, known as feature maps, from a given input image. The CNN architecture is often paired with a downsampling layer to reduce the number of parameters in the model and prevent the model from growing too deep and becoming difficult to train due to vanishing gradients. An activation function is applied to the output feature map from each convolutional layer to add non-linearities to the model.

3.1.1 Convolutional Layers

Each convolutional layer is made up of a number of learnable kernels which are applied to the entire input image by sliding the kernels across the width and height of the image taking the dot product of the kernel and the current receptive field of the kernel for all channels of the input image. Each kernel is able to capture spatially relevant information about the current receptive field and pass this to the subsequent layers of the model. During training these kernels are optimised to capture the most useful information for the task at hand. The number of kernels used at each layer determines the number of feature maps produced, each of which are stacked as channels. As each kernel is applied to the whole input image, the same learned parameters for each kernel are applied to the whole image, with each kernel extracting specific information.

3.1.2 Layer Output Sizes

An issue with convolutional layers is that the output image is slightly smaller than the input image as the kernel cannot be applied over the edge of the image. For a given kernel size, the output image size can be calculated with equation (3.1), where H_{in} and H_{out} represent the input and output sizes along one dimension, k represents the size of the kernel along that dimension, p is the padding applied to the input image and s is the stride at which the kernel is applied to the image.

$$H_{out} = \frac{H_{in} + 2p - k}{s} + 1 \tag{3.1}$$

Padding can be applied to the image to increase the size of the input such that the output image is of the same size. Commonly zero padding is applied, such that the image is padded with zeros around its border.

Reducing Layer Output Size

In order to prevent the models from becoming too deep and to limit the number of trainable parameters in models, it is desirable to reduce the output feature map size. To achieve this, pooling layers are commonly used, commonly either Max Pooling or Average Pooling.

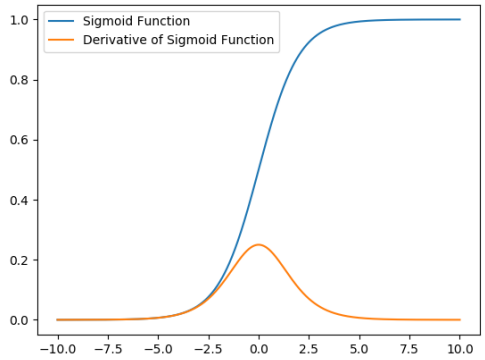
Pooling layers apply a kernel over feature maps in a similar way to a convolutional layer, but apply a fixed function to the inputs rather than using learned parameters. A Max Pooling layer will output the maximum value from each position, and discard the remaining values. An Average Pooling layer will take the mean average of the values at each position.

An alternate method of reducing the size of the output feature maps is by increasing the stride of the convolutional layers, such that the kernel isn't applied at every position, but may skip positions. Provided that the kernel size is larger than the stride of the convolutions, the whole feature map will still be covered. One possible advantage of this is that the kernel values are learnt such that the model will be able to extract the most useful information given that the stride is not one for the layer. By using a layer which learns given the use of strided convolutions, allows the model to learn to downsample feature maps in the manner which retains the most useful information.

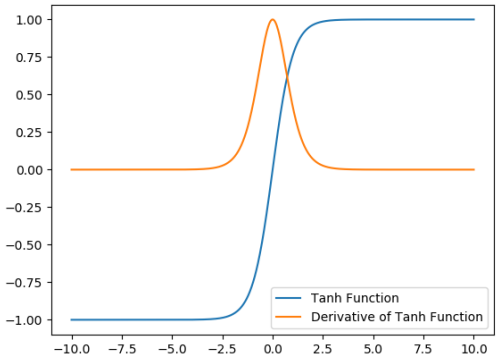
3.2 Activation Functions

In order to add non-linearities to the model, activation functions are applied after each layer of models. This non-linearity enables complex high order functions to be approximated by the network. There exist many activation functions, but this section shall focus on those which have been used in the models proposed in the Chapter

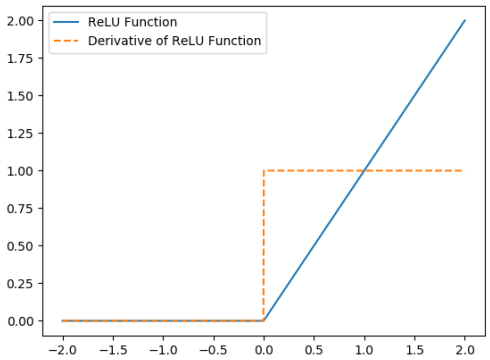
5 and 6. Activation functions are often selected based on the range of their output values, but in cases where this is unimportant it is common to use the Rectified Linear Unit (ReLU) function as it provides strong gradients, which aids in the prevention of vanishing gradients in deep network models, [3].



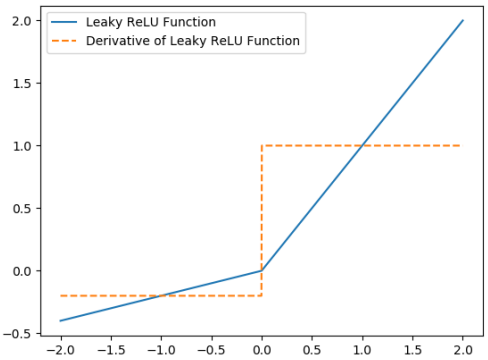
(a) Sigmoid Function



(b) Tanh Function



(c) ReLU Function



(d) LeakyReLU Function

Figure 3.1: Activation Functions

3.2.1 Sigmoid

The Sigmoid function, expressed by equation (3.2) and shown in Figure 3.1a is a logistic function which maps all real values to values within the range of 0 and 1. A potential issue with the Sigmoid function is that the gradients at all points on the curve are small, with a maximum gradient of just 0.25 where $x = 0$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

3.2.2 Tanh

The Tanh function, expressed by equation (3.3) and shown in Figure 3.1b is related to the Sigmoid function by equation (3.4) The Tanh function maps all real values to the range of -1 and 1. While the Tanh function has stronger gradients than the Sigmoid function, the gradients are still close to zero for most input values of x.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.3}$$

$$\tanh(x) = 2\sigma(2x) - 1 \tag{3.4}$$

3.2.3 ReLU

The ReLU function (Figure 3.1c) is a non-linear function with strong gradient values for all input values of x above zero. Unlike the Sigmoid and Tanh function, the ReLU function is not differentiable everywhere due to the function not being continuous. The gradient at the point where $x = 0$ is often treated as zero by library implementations to avoid this issue.

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

While the ReLU function has a positive gradient where $x > 0$, the gradient is 0 where $x < 0$. Another proposed activation function is the LeakyReLU function (Figure 3.1d) described by equation (3.6) where l is a selected parameter. The LeakyReLU function multiplies the input value x by some parameter l, where $l < 1$, such that x is attenuated.

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x > 0. \\ lx, & \text{otherwise.} \end{cases} \tag{3.6}$$

3.2.4 Softmax

The Softmax function is commonly used in classification problems in which a model aims to classify an input value $\boldsymbol{x}$ as one of N labels. The Softmax function, expressed by equation (3.7), converts the input $\boldsymbol{x}$ into a normalised probability distribution over the N labels, the label with the highest likelihood is interpreted as the model's prediction. Such models are commonly trained with the Negative Log Likelihood loss function, by applying the log to the output of the Softmax function.

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j^N e^{x_j}} \tag{3.7}$$

The Negative Log Likelihood loss function can be expressed by equation 3.8 where $\boldsymbol{x}$ is the prediction from the Softmax function.

$$L(\boldsymbol{x}) = -\log(\boldsymbol{x}) \tag{3.8}$$

The Softmax function returns a likelihood value for each label, this can be interpreted as the confidence the model has that the input data values corresponds to each label. The range of the Negative Log Likelihood shown in Figure 3.2 shows that when the model confidence in the correct label is high, the Negative Log Likelihood tends to zero, while when the confidence is low for the correct label, the loss tends to infinity.

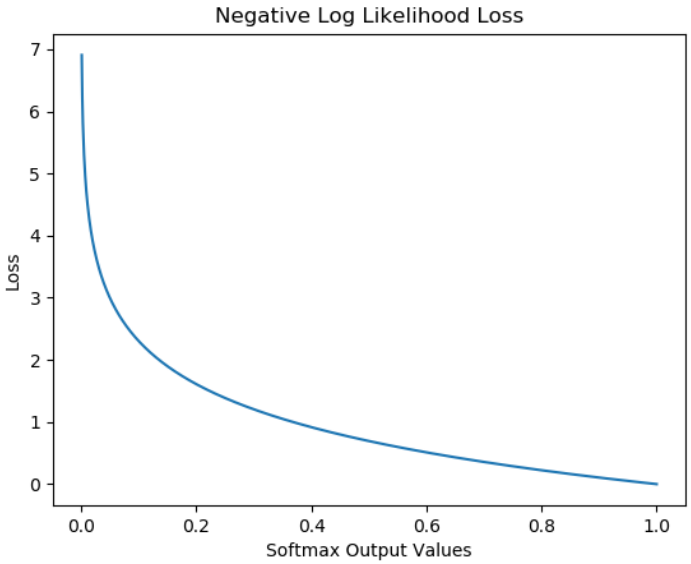


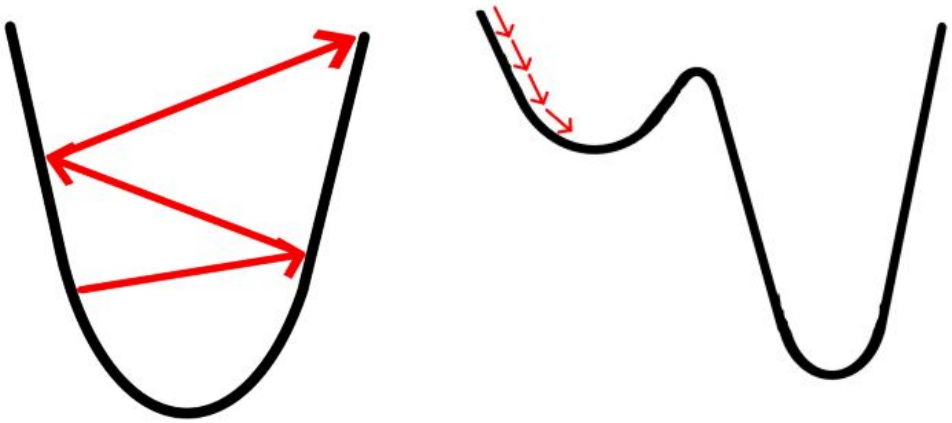
Figure 3.2: Negative Log Likelihood Loss

3.3 Optimisation Algorithms

With the aim of finding the model parameter values which minimise the loss function of the model, Gradient Descent algorithms are commonly used to find such parameters. Given a loss function $L(x)$ which is to be minimised, where x represents the model parameters, the value of x is iteratively updated by calculating the gradient of $L(x)$ for the current parameters and updating the parameters as described by equation (3.9). ϵ is the learning rate for the algorithm which determines the step size made at each update. Too large a step size and the update may overshoot the optimum value (Figure 3.3a), too small a step size may result in a model which takes an impractical length of

time to train or may fail to find a global minimum entirely (Figure 3.3b). It's is often useful in practice to begin with a learning rate which is able to make large steps, then to reduce the learning rate ϵ as the model trains.

$$x' = x - \epsilon \nabla_x L(x) \tag{3.9}$$



(a) Large Learning Rate (b) Small Learning Rate

Figure 3.3: Learning Rates

3.3.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) takes a minibatch of m samples from a data generating independent and identically distributed (i.i.d) distribution, and calculates the average gradient with which the update rule described by equation (3.10) is applied. The use of minibatches results in an approximation of the true gradient as the whole dataset is not evaluated before making an update, just a small subset of the dataset. This can have the affect of adding regularisation to the learning procedure, particularly with smaller batch sizes [3]. The SGD update rule for a loss function $L(.)$ and model $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ can be defined by equation (3.10):

$$\boldsymbol{g} = \frac{1}{m} \nabla_\theta \sum_i^m L(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i)$$

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \epsilon \boldsymbol{g} \tag{3.10}$$

where $\boldsymbol{g}$ represents the approximation of the gradient, $\boldsymbol{\theta}$ represents model parameters and $\boldsymbol{x}_i$ represents m samples making up the minibatch $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ and corresponding target values $\boldsymbol{y}_i$.

3.3.2 Momentum

In order to the accelerate the training of models and find the global minimum faster, momentum is often used with optimisation algorithms. In addition to the gradient term, a velocity term is also calculated as an accumulation of decaying moving average values from the previous gradient approximation [3]. The result of this is that the optimisation will continue to move in the direction in which it has been moving, aiding in the prevention of the model becoming stuck in local minimum. Figure 3.4 demonstrates how momentum may accelerate the optimisation process. Here the black arrows represent the gradient evaluated at each point, while the red line shows the path taken with momentum.

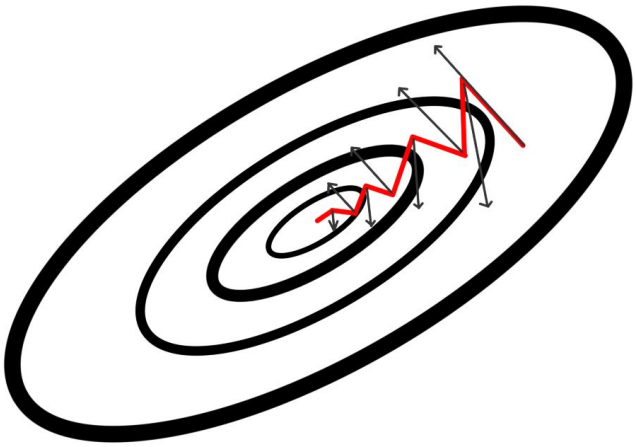


Figure 3.4: SGD with Momentum [3]

Adaptive Learning Algorithms

Such algorithms as RMSprop and Adam exist which are built upon the principles of the SGD algorithm with the addition of momentum but adapt their learning rates or momentum values based on a history of previous gradient values. These aid in finding the global minimum for the loss function at hand.

3.3.3 RMSprop

While momentum can be used to help reduce oscillations while training, RMSprop goes further by updating the learning rate dependent on each dimension in the loss function. Given a pair of weight parameters b and w, during training these are improved to find the optimal solution. During training the contour plot in Figure 3.5 can be seen to oscillating a large amount in the b axis.

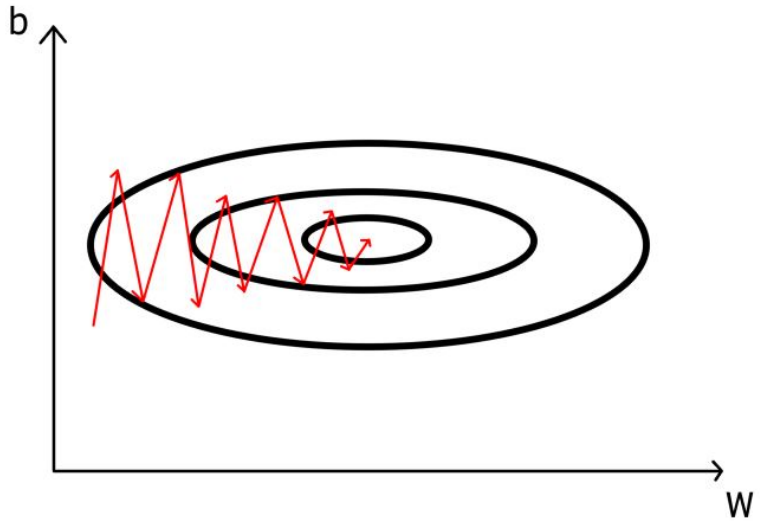


Figure 3.5: Incentive Behind RMSprop [3]

To prevent this oscillation the learning rate could be reduced, but it can also be seen that it would be desirable for the learning rate for the w parameter to be increased as the model would converge faster. RMSprop is an adaptive learning rate optimisation algorithm which aims to reduce the learning rate for parameters with large gradients and increase the learning rate for those with small gradients [3].

This is achieved by keeping an sum of weighted squares of the derivatives of each parameter, where β is a tunable hyperparameter:

$$\begin{aligned} S_{\delta w} &= \beta S_{\delta w} + (1-\beta)\delta w^2 \\ S_{\delta b} &= \beta S_{\delta b} + (1-\beta)\delta b^2 \end{aligned}$$

The weight parameters are then updated with the root mean square of the weighted sum of the parameter derivative, where α is a tunable hyperparameter.

$$\begin{aligned} w &= w - \alpha \frac{\delta w}{\sqrt{S_{\delta w}}} \\ b &= b - \alpha \frac{\delta b}{\sqrt{S_{\delta b}}} \end{aligned} \tag{3.11}$$

When the gradient of the weight parameter is large, the learning rate for that values is small, while weight parameters with small gradient values are updated with a larger learning rate, as expressed by equation (3.11).

Chapter 4

Literature Review

This section shall discuss the currently existing datasets and models relating to VSR. This shall be explored in both the space of two dimensional images and three dimensional scans of speaking subjects, the current successes in these fields and the current issues which are faced in progressing these areas of research forwards.

4.1 Current Lip Reading Models and Datasets

In recent years the problem of VSR, also know as lip reading, has seen huge advances due to the availability of new datasets and the use of Deep Learning models [4, 8, 9]. This section shall discuss the progresses which has already been made in this field and the datasets publicly available on which to train such models. Current lip reading models, to the best of the author's knowledge, all make use of 2D temporal data such as videos by evaluating the frames of the video, using this information to make a prediction of the word, or words, which were spoken in the sequence.

However in reality, humans who are able to lip read also have access to three dimensional data due to our depth perception, which is not well represented in 2D video frames; thus is can be hypothesised that some information captured by 3D temporal scans of speaking subjects can also be used for speech prediction. Unlike 2D temporal data, 3D temporal data (3D video), requires multiple cameras to record simultaneously, which in turn requires synchronisation between the cameras, increasing the complexity of the system beyond simply having multiple cameras. The data must then be processed to produce the final product, whether this is in the form of video with depth information or more complex 3D scans [1].

4.1.1 Datasets

There currently exists multiple labelled video datasets which can be used for traditional VSR; here 'traditional' refers to using 2D temporal data. Datasets such as GRID [13],

LRW [4], LRS [8] and LSVSR [9] have been constructed by compiling the video data from various sources, each with an increasing vocabulary and dataset size.

Controlled Conditions Datasets

The GRID dataset was released in 2006 [13] and contains 34,000 samples from 34 speakers, each with 1000 sentences. Based on previous work using the coordinate response measure (CRM) [14], the corpus uses sentences with a fixed grammar: <command:4>, <colour:4>, <preposition:4>, <letter:25>, <number:10>, <adverb:4>, with a total vocabulary of 51 words. Aside from the restricted vocabulary size in comparison to more recent datasets, the primary limiting factor of the GRID dataset is that all data was captured directly for the use of the dataset, and so models trained on this dataset have learnt under controlled conditions, something not found in the desired wider applications of VSR.

In The Wild Datasets

To build larger datasets more suited for deep learning models, the following are commonly built with "*in the wild*" data, meaning that the original videos have not been captured with the intention of being used for this dataset, whilst the data collected for the datasets is preprocessed such that it is suited for the task. Variations in lighting, angles, speakers and a wide vocabulary are common, this does however make the datasets more challenging to learn from, but the models are no longer as biased to controlled conditions.

LRW and LRS are both comprised of content from BBC broadcasts [4, 8] allowing for a far larger corpus size of 500 words for LRW and over 6000 words for LRS. Both datasets are captured and processed with the same pipeline (Figure 4.1) summarised as follows.

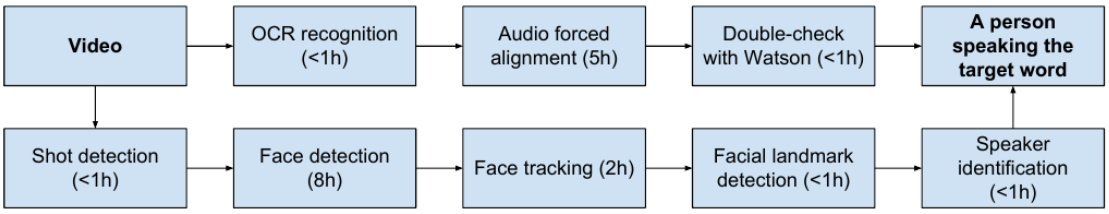


Figure 4.1: LRW Dataset Pipeline [4]

Firstly, as the text transcripts are broadcast as bitmaps, optical character recognition (OCR) [15] is used to obtain the text spoken in the video, the audio and text are then aligned per frame using the HTK toolkit [16]. The quality of these predictions are validated with the use of the commercial IBM Watson Speech to Text converter.

Secondly, face detection and tracking are used so that the frame can be cropped to feature the subject's mouth at the centre. To achieve this, a histogram of oriented

gradients based (HOG) detection algorithm [17] is used on all video frames to detect faces, see left hand side of Figure 4.2. A Kanade-Lucas-Tomasi (KLT) feature tracker is also applied to the frames, such that where HOG and KLT overlap (see centre of Figure 4.2), it is assumed to be tracking the face correctly. To identify mouth position, facial landmarks are then used using an ensemble of regression trees method from [18], see the right of Figure 4.2.

Figure 4.2: LRW Face Detection and Tracking [4]

As multiple faces can appear at any one time, it is assumed that the speaker will be the only subject with a moving mouth. Assuming that the lip movements fall within a frequency range, the Fourier transform is applied to the openness of the mouth defined by the distance between top and bottom lip from the facial landmarks. A support vector machine (SVM) is then trained on the frequency spectrum to classify who is speaking in each frame. The frames are then cropped around the relevant speaker, an example of a sample from the dataset is shown in Figure 4.3.

Figure 4.3: LRW Dataset Sample: 24 frames of a subject saying *'about'*, figure from [4]

LSVSR is a dataset published by DeepMind and Google which makes use of the huge amount of videos on YouTube [9], resulting in a total length of 3886 hours of training data. Unlike LRW or LRS, LSVSR aligns phonemes to frames, as opposed to words or characters. The pre-processing steps are similar as in LRW and LRS, although the alignment is performed with the algorithm laid out in previous work by DeepMind [19].

3D Datasets

To the best of the author's knowledge, there are currently only two datasets with 3D temporal data which could be appropriate for training lip reading models, neither of which have been captured with the intention of being used for VSR.

The first of which is LRW-3D [10] which has been captured from four subjects, two native English speakers and two non-native to increase variability in the dataset. The subjects have been captured speaking the corpus used in the LRW dataset [4], a vocabulary of 500 words. The resulting dataset comprises of 660 seconds of 3D meshes and audio per subject. The dataset is not comprised of full sentences but would be appropriate for word-level lip reading, however a total duration of 660 seconds is likely to be too short for a deep learning model to train effectively. The dataset had also not been officially published at the time of this report.

The second is the VOCASET [2], an unlabelled dataset captured from 6 male and 6 female subjects. Each subject was recorded speaking 40 sequences, each ranging from 3 to 5 seconds, resulting in a total time of 30 minutes. The recorded 3D meshes are registered to the FLAME model [1], a statistical 3D facial mesh with around 5000 vertices, an example sample from the dataset can be seen in Figure 4.4. Unlike the LRW-3D dataset, the sequences are grammatically correct sentences, chosen to maximise phonetic diversity. This makes the VOCASET appropriate for creating a model for sentence-level lip reading. Similarly to LRW-3D, 30 minutes of training data is likely to be a limiting factor when training a deep learning model.

Figure 4.4: VOCASET Example Frames [2]

It should be noted that neither of these datasets were captured for the purpose of lip reading, but for synthesising realistic statistical facial models driven from an audio input, and thus there are no transcriptions for VOCASET. The transcriptions could be obtained with an Audio Speech Recognition (ASR) method such as DeepSpeech [20] and then alignment would have to be performed to the datasets before using the data to train lip reading models.

4.1.2 Lip Reading Models

In [4] Chung et al. used a multiple towers convolutional model based on the VGG-M architecture for multi-label classification to classify a video clip to one of the five hundred labels in the LRW dataset. The model architecture (Figure 4.5) first processes each frame separately with a shared convolutional layer to extract features from each frame. These features are down-sampled with a pooling layer before being concatenated into a single high channel feature map. A one dimensional convolutional layer then reduces the channels of this feature map before being processed by subsequent convolutional and pooling layers. The final layer of the model is passed through a softmax activation function to predict one of the 500 word labels, see section 3.2.4.

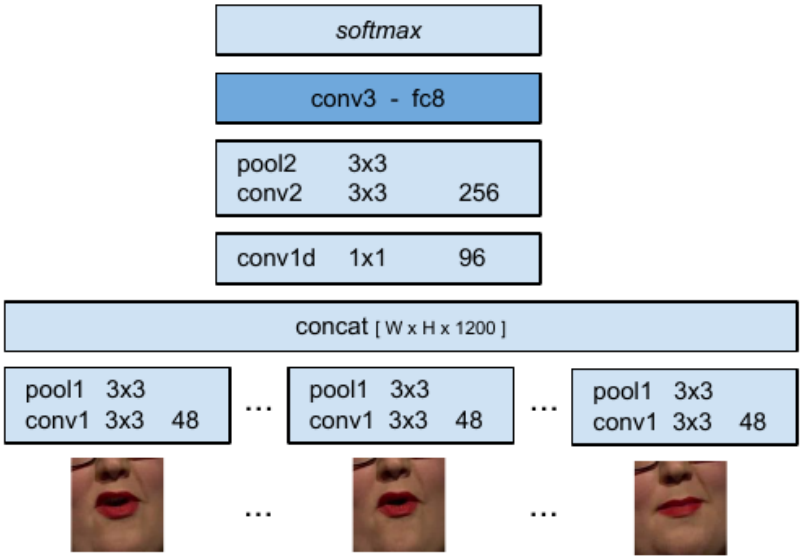


Figure 4.5: LRW Multiple Towers Model [4]

Chung et al. suggest that the multiple towers model is effective due to the delay in time domain operations until after the first convolutional layer which allows for tolerance to registration errors in the dataset.

As well as measuring the performance of the model on total prediction accuracy, [4] also uses character-level edit distance, measured as the minimum number of characters required to be changed in the predicted label to obtain the ground truth.

The LipNet model [7] was the first model produced to be able to predict end-to-end sentence-level lip reading of varied length, while previous work predicted on a word level. The model used spatiotemporal convolutions to process multiple frames of video at once followed by a recurrent layer using Gated Recurrent Units (GRUs) [21] and made character level predictions. The model used the GRID dataset [13] due to it being the largest sentence level dataset at the time, on which LipNet achieved the state of the art performance.

As the LRW dataset could not be used to train a model such as LipNet for sentence-level lip reading, but the GRID dataset had limitations in the number of subjects and

vocabulary, Chung et al. created the LRS dataset [8]. The model presented in [8] was trained on both audio and video and made capable of taking either or both as the model inputs. To prevent the model from being dependent on a single input source, the inputs are systematically distorted or removed. The video input is passed through convolutional layers, followed by LSTM (Long Short-Term Memory) layers [22], while the audio is converted to Mel-frequency cepstral coefficients (MFCC), then input to LSTM layers. The two are combined with the attention mechanism and further LSTM layers and an output fully connected layer with softmax activation for character level predictions. The model also makes use of curriculum learning by initially training the model on short sequences of single words, before increasing the length of training sequences during training. It is stated by [8] that this accelerates training and reduces overfitting.

In 2018 DeepMind published their V2P (Video to Phonemes) model [9] along with the LSVSR dataset built from content from YouTube. LSVSR greatly surpassed all previous datasets in size and variation, containing 3886 hours of training data. The model follows a similar architecture to LipNet [7] using spatiotemporal convolutional layers followed by recurrent layers, however it used an increased number of convolutional layers and used LSTM layers in place of GRUs. It should be commented that the size of the model and dataset dictated the use of 64 GPUs for training to allow a batch size of 128. Unlike previous models, the V2P model predicts phonemes as opposed to characters, these phonemes are then processed by a language model for word prediction as with previous models.

To the best of the author's knowledge, there do not currently exist any papers which have explored the use of 3D temporal datasets for use with lip reading models. This is likely due to the shortage of 3D temporal data due to the difficulties in obtaining such data, on which such models could be trained on.

4.2 Data Generation

In order to construct a deep learning lip reading model capable of being trained on 3D temporal data, appropriate datasets must first be established. Existing datasets [10, 2] discussed in section 4.1.1 have been captured directly with the use of multi-camera capture rigs under controlled conditions. The total duration of both of these datasets is very short in comparison to the video datasets such as LRW, LRS and LSVSR, which is a limiting factor to the models which could be trained using them.

As the models also use different mesh models to represent the data that has been captured this also prevents the two datasets being joined directly. Unlike 2D video data, there currently lacks a large body of 3D video data which is publicly available, limiting the construction of 3D datasets to directly capturing more 3D scans with multi-camera capture rigs, similar to those used in [10, 2] and generating synthetic training data.

4.2.1 Audio Driven Data Generation

Karras et al. proposed a method for generating 3D facial animation from audio with the use of a CNN architecture [5], see Figure 4.6. The model is actor specific, but restricting this allowed for the model to be trained on just 3-5 minutes of data per actor. Firstly, a fixed function auto-correlation layer is used to extract a time-varying sequence of speech features from the audio input, named the formant analysis layer. Then five convolutional layers are used to extract short-term animation features from the formant analysis network layer. This is then followed by five convolutional layers, with each input concatenated with a learned emotional state representation. Finally two linear layers are used to drive vertex positions from a starting mesh of the given actor.

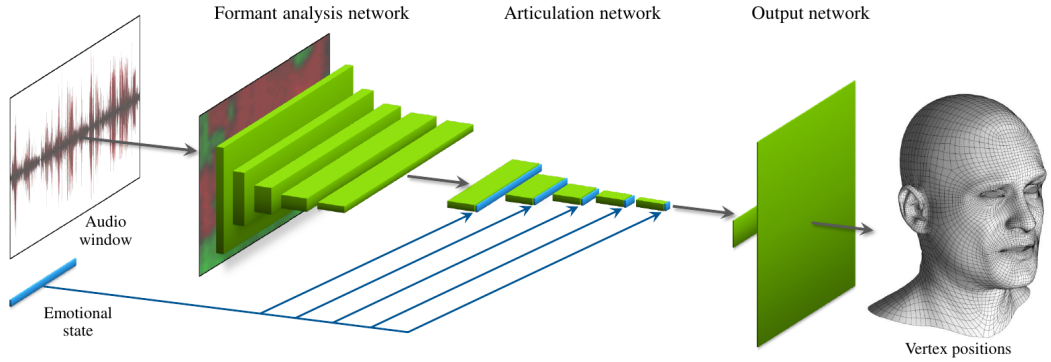


Figure 4.6: Audio Driven Facial Animation Model Proposed by Karras et al. [5]

However, as the model by Karras et al. is not independent of the actor it cannot generalise to new subjects. Tzirakis et al. propose a model which is independent of speaker and capture rig [10]. As discussed in section 4.1.1, a dataset was constructed of 3D speaking faces using the LRW dataset [4] for the corpus. This allowed Tzirakis et al. to create a model which can synthesise facial motion from audio from the LRW dataset. The model used is similar to that used in [5], firstly extracting short term temporal features from the input audio with a convolutional network followed by another convolutional network to analyse the extracted features. Unlike the model used in [5], the model used in [10] is trained to drive learnt Blendshapes as opposed to vertex positions. This reduces the number of output parameters of the model substantially in comparison to that used by Karras et al. whilst reducing the potential expressions possible to be produced by the model.

Similar to [10], the VOCA model [2] synthesises video sequences of 3D models speaking given an audio input, see Figure 4.7. The VOCASET dataset discussed in section 4.1.1, was captured with the intention of training this model to be independent of the speaker, hence a large range of speaker's are used within the dataset. The model is comprised of three sections: audio feature extraction, a feature encoder and a decoder to drive a template facial mesh from the FLAME model [1]. The audio feature extraction makes use of the pre-trained Mozilla implementation of the DeepSpeech model, based on the paper by Hannun et al. [20]. The DeepSpeech model takes audio as an input

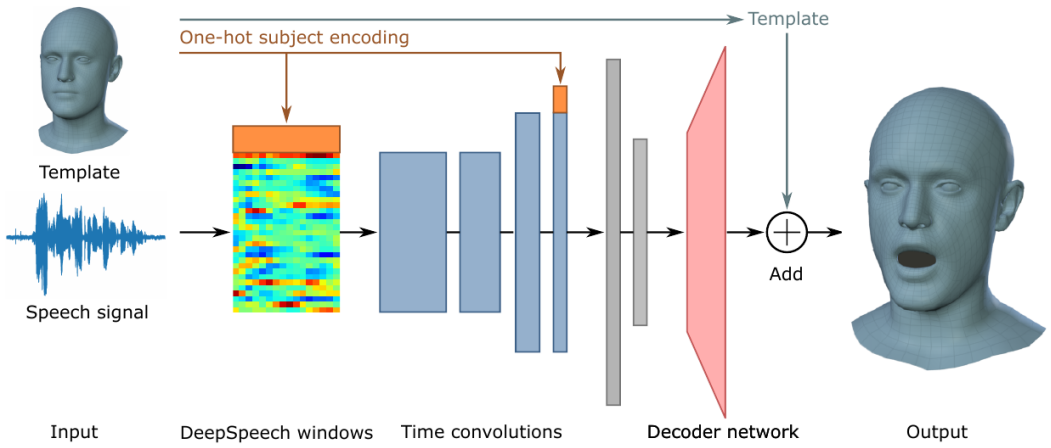


Figure 4.7: The VOCA Model [2]

and returns the unnormalised log-probabilities for an alphabet of the 26 standard characters, a space, apostrophe and blank character for time slices in the audio input. The encoder is a convolutional network which is conditioned on the speakers identity, such that the latent space of speaker styles can later be explored on new audio inputs. Finally, the decoder is made up of a fully connected layer with a linear activation function used to output the displacements of the 5023 vertices in the template face. However, as the model is conditioned with the label of which speaker the input data is from, the model does not truly learn to generalise to multiple speakers, but learns each speaker separately, thus not being independent of the speaker.

4.2.2 Generative Adversarial Networks

A recent development in generating synthetic data samples is the use of generative adversarial networks (GANs) [6]. The concept behind GANs is to have two machine learning models; a generator and a discriminator. The task of the generator is to produce samples from an unknown high order probability distribution which correctly resemble samples from the distribution defined by training data. The generator achieves this by transforming a random sample from a known probability distribution, such as a Gaussian distribution as used in [6], to a sample from the unknown distribution. This is achieved by finding the function which maps between the two distributions. The discriminator however, attempts to correctly learn to discriminate between the real and the generated samples.

The original loss function (4.1) proposed by Ian Goodfellow [6] forms a min-max game, where the loss of the generator is attempting to be minimised by having the discriminator label all the generated samples as real, while the loss of the discriminator is maximised by correctly classifying real and fake samples. Here $\boldsymbol{x}$ represents the real data samples from an unknown probability distribution p_{data} and $\boldsymbol{z}$ represents a random noise sample from a known distribution. The two networks are trained in an alternating fashion until the discriminator achieves an accuracy of 50% on real and

generated samples, effectively making binary guesses between the two.

$$\min_G \max_D V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{4.1}$$

GANs however, are difficult to train for two main reasons. Firstly, the equation (4.1) is challenging as it provides small gradients while generated samples are poor as discussed in [6]. Progress in developing new loss functions is discussed in section 4.2.2 Secondly, early networks must also be balanced with a similar model capacity to prevent one from getting too much better than the other, preventing the other from improving. Various architectural changes have improved this issue [23, 24], although it seems to be closely tied to the loss function being used [25].

Stability Improvements of GANs

There have been a large number of papers presenting new techniques with differing levels of success and training stability, a small handful of key papers which provide large advances in the generative adversarial training model shall be discussed here. A primary research focus around GANs has been in finding new loss functions on which to train the model to improve stability and performance. The original loss function proposed in the original paper [6] identifies an issue with equation (4.1) in that the gradient back propagated to the generator when the generated samples are poor, is very low as shown in figure 4.8.

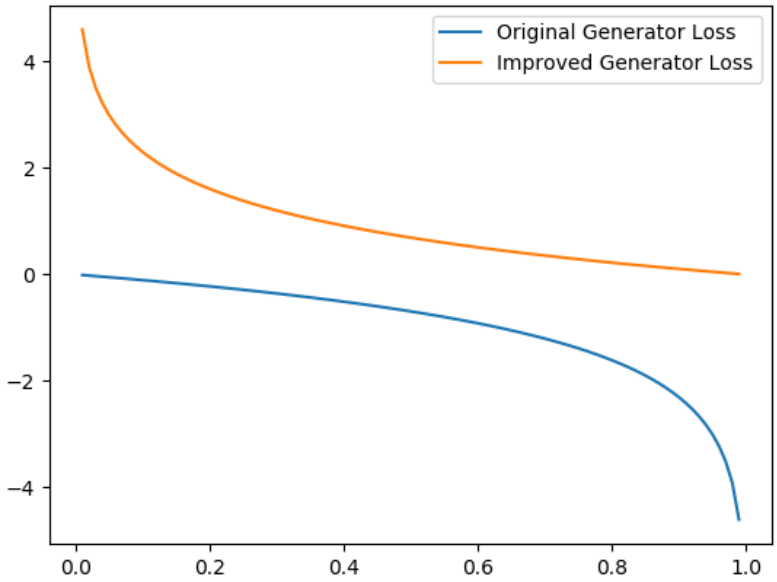


Figure 4.8: Goodfellow's Generator Loss Plots [6]

This in turn makes it very challenging to improve the performance of the generator. The suggested improvement made in [6] is rather than to maximise the number of generated samples which the discriminator incorrectly classifies, instead to minimise the number of generator samples the discriminator correctly classifies, as described in equation (4.2). Figure 4.8 shows how this results in the gradient propagated back to the generator is far larger when the generated samples are poor.

$$\min_G \max_D V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))] \tag{4.2}$$

Further stability improvements were proposed in [23] which allowed deep convolutional generative adversarial networks (DCGANs) to be successfully trained for the first time. Radford et al. proposed three main contributions.

- Replace deterministic pooling layers with strided convolutions in both the generator and discriminator networks, allowing the networks to learn their own spatial upsampling and downsampling.
- Remove all fully connected layers used on top of convolutional layers, resulting in a fully convolutional model.
- Apply batch normalisation [26] before the input of each layer. This normalises the input to each model layer to zero mean and unit variance. Batch normalisation assists with training difficulties due to poor weight initialisation and allows gradients to flow through deeper networks more easily.

Radford et al. state these to be critical improvements to allow generator networks to begin learning by preventing all samples from collapsing to a single point.

In an attempt to stabilize the training of GAN models, the Wasserstein or 'Earth Mover Distance' loss function was proposed by Arjovsky et al. [27] shown in equation (4.3) where $\mathcal{D}$ is a set of 1-Lipschtiz functions.

$$\min_G \max_{D \in \mathcal{D}} W(\mathbb{P}_r, \mathbb{P}_g) = \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})] - \mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\hat{\boldsymbol{x}})] \tag{4.3}$$

The Wasserstein GAN (WGAN) model uses a critic as opposed to a discriminator, this is due to the fact that the discriminator is no longer a binary classifier, but being used to critique the real and generated samples. As the Wasserstein function is continuous and differentiable, the critic can be trained until optimality, and [27] argues that it should be. As the critic is trained to optimality it does not saturate, but converges to a linear function. This provides the generator with a gradient which is more reliable, resulting in the generator learning consistently. The Wasserstein loss function has been shown to greatly improve training stability by providing consistent gradients throughout training and no longer requires that the two networks have a balanced model capacity.

In order to use the Wasserstein distance as the loss for WGAN, the Lipschtiz condition must be enforced. In the WGAN the model weights are clipped to enforce this constraint [27], however this is stated to be a non-ideal method of achieving the Lipschtiz constraint and calls for future work to investigate more effective methods. The use of gradient penalty is proposed in [25] in order to satisfy this condition more elegantly by minimising the Critic's loss function as described in equation (4.4).

$$L = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\| - 1)^2] \tag{4.4}$$

Using the Wasserstein loss function with gradient penalty enforces the Lipschtiz condition and allows highly complex architectures to be trained successfully, including those with residual units [25].

Architectural Developments

Mirza et al. showed that GANs could be trained with conditional inputs to the generator network in addition to the noise component [28]. This has facilitated other uses of GANs such as image to image translation for style transfer [29]. Vougioukas et al. used a temporal model to generate a sequence of video frames given an image of a subject and an audio sequence of spoken text to synthesise the subject speaking [30]. The model uses two discriminators, one to determine if individual frames are realistic images of the subject's face and a second which evaluates the sequence of video frames to determine if it is realistic. The model uses temporal components to examine if the frames of video are consistent in time, preventing sudden jumps in facial position.

Other novel architectures include the progressively growing GAN model [31] which is able to generate highly realistic images of faces to a high resolution. This is achieved by initially training a shallow model to produce 4x4 pixel images before increasing the depth of the network and training further at a higher resolution. By forcing the model to firstly produce and examine low resolution images, the model has to be able to synthesise simple low level features effectively, such as facial shape which is common to all samples. Once the model can produce these features a higher resolution is used, allowing it to learn more complex features, such as hair and eyes.

The attention mechanism [32] has been shown to be useful when applied to image data [33] in order to capture relationships between spatially distant points. As such points are further apart in the image, previously deeper convolutional models were required to allow a large enough receptive field to capture information on these two points. Zhang et al. point out that this is a common issue with DCGANs [23], where generated samples fail to produce structural patterns, while they do exceedingly well at local textural patterns. This is often seen in generated images of animals with realistic fur, but oddly shaped or an incorrect number of limbs. The attention mechanism is applied to GANs [24] to allow for long range dependencies in the image to be modelled by convolutional models more effectively.

To the best of the author's knowledge, there currently exists no generative adversarial networks which aim to generate 3D facial models with speech audio as a conditional input.

4.3 Summary of Related Literature

Visual speech recognition has seen large improvements in recent years, progressing from word level prediction on a relatively small vocabulary set [4], to variable length sequences predicting a sequence at a sentence level with the use of recurrent networks making character level predictions to construct the completed sentence [9]. These advances have come in tandem with the increase in the size and variation of the datasets on which to train such models. While there currently exist no datasets made up of 3D temporal data which have been constructed with the intention of lip reading, there has been progress in the area of 3D model generation from an audio input. A proposed solution to this problem would be to generate synthetic 3D temporal data by using a generative model, allowing a VSR classifier to be trained on the resulting dataset. The model by Karras et al. is currently not publicly available due to being researched in collaboration with Nvidia. The VOCA model however, is publicly available, although how independent the model truly is of the speakers it was trained on is unclear. The VOCA model also has additional dependencies on the DeepSpeech model and thus is not trained fully end-to-end, potentially losing some information which might be of use for driving facial animation. A currently unexplored area of research is the use of GANs driven by audio to generate 3D temporal data of speaking subjects.

Chapter 5

Blendshape Classification

This chapter shall discuss the problem of word level classification from Blendshape Parameters obtained from a dataset created with the VOCA model [2] processing audio from the LRW dataset [4]. Classification is performed with two Convolutional Neural Network architectures on 500 labels. The two model architectures, how performance is assessed and how the models have been tuned to maximise performance on a held out validation set is also discussed. The final performance of the models is evaluated on the test set and conclusion as to the effectiveness of using Blendshape Parameters as a means of lip reading is evaluated.

5.1 Problem Definition

As with traditional lip reading problems, the end goal of VSR models using 3D temporal data is to correctly classify variable length input sequences at a sentence level. However, before this problem can be tackled, the simpler problem of word level classification should be first approached to ensure that this is possible, before moving on to more advanced problems. If word level classification cannot be achieved, it may be concluded that there is not a sufficient amount of information encoded in 3D temporal models for more complex problems to be solved with the current set of assumptions.

These assumptions include:

1. Provided an appropriate dataset of 3D temporal facial scans of subjects speaking, a set of Blendshape Parameters which realistically represent the motions of speech can be found.

2. The Blendshape Parameters represent an encoding from which a classification model is able to make word level predictions.

5.2 Generation of a 3D Lip Reading Dataset

In order to train any model, an appropriate dataset must exist upon which the model can be trained. An appropriate dataset for word level prediction from Blendshape Parameters would consist of:

- A large enough vocabulary that single instances of words with unique phonemes do not exist, as this would allow classification of these words to be too simple.
- Facial mesh recordings for all samples in the dataset, all of which brought into alignment to remove movement not related to speech.
- Blendshape Axes which can be shown to contain useful interpolation for speech.
- The Blendshape Parameters corresponding to each sample for the given Blendshape axis.

As discussed in section 4.1.1, currently there exists no public datasets which have been created for the purpose of lip reading from 3D temporal data. In addition to this, the datasets which have not been created with the purpose of lip reading which do contain 3D temporal data of speaking facial scans, are either not public in the case of Karras's model [5] or are unlabelled and too small to train a model effectively as is the case with the VOCASET [2]. Both of these datasets also contain a large vocabulary, with too few samples of each word. The VOCASET dataset could be labelled with the use of an automatic speech recognition model such as DeepSpeech as suggested in section 4.1.1, however as the dataset was constructed to contain as much variation in speech sounds, a degree of word repetition would offer little benefit, hence the lack of consistent word repetition.

In order to tackle the issue of an appropriate dataset not existing, two solutions exist. The ideal solution would be to capture subjects speaking words from a set corpus using a 3D capture equipment, this solution was not a viable option for this project due to the specialised equipment required and the data processing required after data capture had been performed. Another means of constructing the dataset would be to generate the data with an existing model which can convert an appropriate audio dataset to a series of facial meshes from which Blendshape axis can be constructed and parameters recovered. This can be achieved by processing audio from the LRW dataset [4] though the VOCA model [2]. The VOCA model claims to be able to generate realistic 3D facial motion from audio, as lip reading is notoriously difficult for humans, this is difficult to validate. Assuming that the VOCA model does in fact generate realistic facial motion, the LRW dataset allows for a 3D dataset to be generated with a set corpus of the 500 words contained in the LRW dataset.

55,000 audio samples are taken from the LRW dataset with a uniform distribution of all labels, resulting in 110 samples for each label. Audio samples from the LRW dataset are from "in the wild" scenarios [4], meaning that there is background noise and audio in the clip. A sample of the word "*ABOUT*" will contain the word "*ABOUT*" within the audio sample, but will contain other speech either side of the clip. Figure 5.1 shows such a sample labelled as "*ABOUT*" in which the phrase "talking about fair" is spoken.

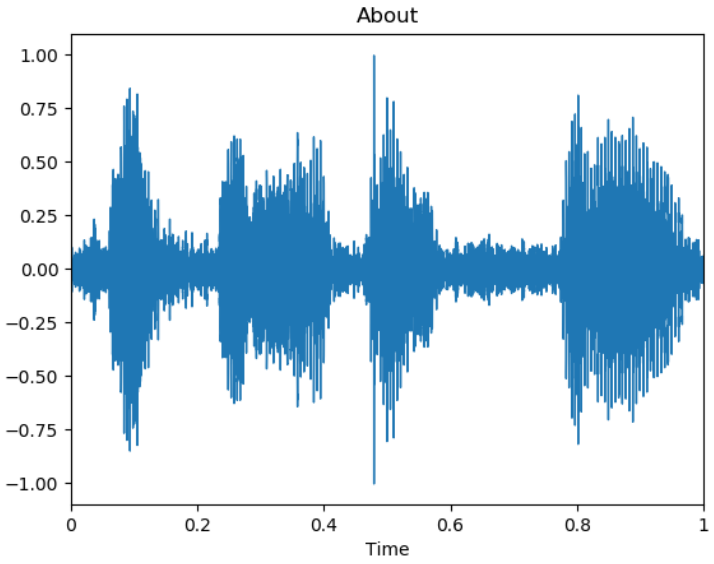


Figure 5.1: Audio Sample from LRW [4] "Talking About Fair"

The full LRW dataset contains 1000 training samples for each label, however due to the size of the VOCA model, processing 55000 samples (11% of the full dataset) took roughly 8 days with the GPU acceleration of an Nvidia Tesla M60. This is due to the size of the VOCA model, which incorporates the DeepSpeech model as a means of audio feature extraction. In parallel to the generation of the 3D temporal data from the VOCA model, Blendshape Parameters were recovered as the facial scans were produced. No significant decrease in the speed of data generation was observed running these tasks in parallel.

5.2.1 Blendshape Axes Creation

Ideally, all 55000 samples would be collected and the PCA would be performed on this to construct the Blendshape Axis. However, due to disk space limitations a complete dataset of 55000 facial scans could not be collected at once due to the size of the generated facial meshes as this would total 2.365 million generated facial mesh data samples, each of which occupying around 185KB, requiring a total of 440GB of disc space. To perform PCA on the generated facial meshes for Blendshape Axis (see section 2.1.1), the first 5000 facial mesh sequences were generated totalling 83 minutes of data and 215000 individual facial mesh data samples. Once PCA has been applied to a dataset of facial scans, the transformation matrix $\boldsymbol{W}$ is known. $\boldsymbol{W}$ has columns of eigenvectors corresponding to eigenvalues, each of these eigenvectors represents the principal axis upon which interpolated can be performed. These axes are the Blendshapes Axes and by interpolating along these axes by given amounts, motions of the facial mesh can be expressed in a low dimension. The first Blendshape Axis corresponds to the largest eigenvalue, as that axis contains the largest amount of variation amongst the dataset. The number of Blendshapes which can be created will

be limited by the number of positive non-zero eigenvalues found from PCA, however as each successive Blendshape Axis corresponds to a decreasingly small eigenvalue, the amount of variation in each successive axis decreases. This results in realistic reconstruction being able to be achieved with a relatively small number of Blendshapes containing the majority of the variation within the dataset. Examples of the First and Second Blendshape Axis from the first 5000 samples are shown in Figures 5.2 and 5.3 respectively, further examples can be viewed in the Appendix: Figure 7.1, 7.2. These are constructed by interpolating along each axis individually by a fixed step amount.

Figure 5.2: First Blendshape Axis Interpolation

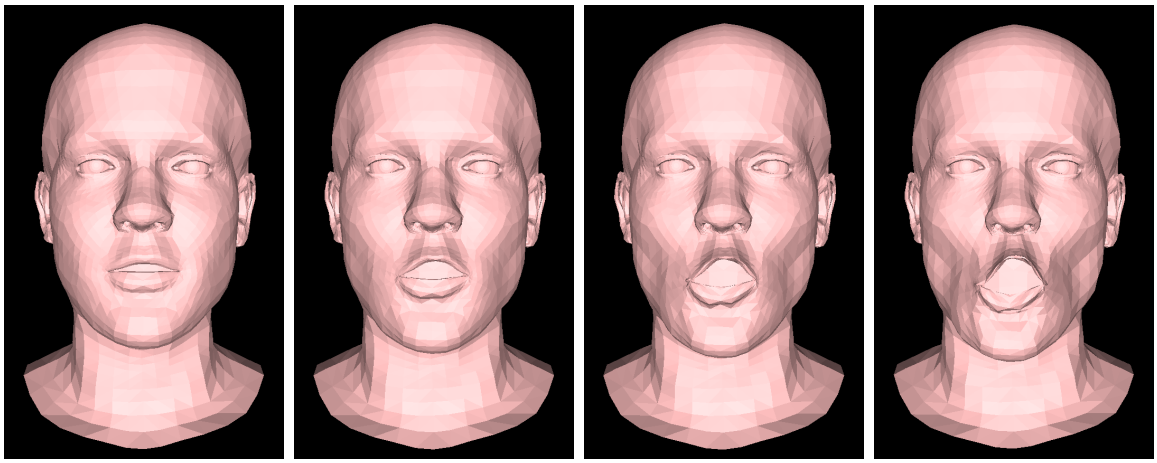


Figure 5.3: Second Blendshape Axis Interpolation

5.2.2 Blendshape Parameter Recovery

Given a template mesh $\boldsymbol{M} \in \mathbb{R}^{(N\times 1)}$ upon which to interpolate, where $N = 3n$ and n represents the number of vertex points in the facial mesh, each of which have an x, y and z coordinate value and Blendshape Axis $\boldsymbol{W} \in \mathbb{R}^{(N\times d)}$ as described in section 2.1.1, where the d first principal axes have been kept, representing the axis along which the template mesh $\boldsymbol{M}$ can be interpolated.

$$\boldsymbol{M} = \begin{bmatrix} x_1, & y_1, & z_1, & \ldots, & x_n, & y_n, & z_n \end{bmatrix}^\top, \quad \boldsymbol{M} \in \mathbb{R}^{N \times 1}, \quad N = 3n,$$

$$\boldsymbol{W} = [\boldsymbol{w_1}, \ldots, \boldsymbol{w_d}], \quad \boldsymbol{W} \in \mathbb{R}^{(N \times d)}$$

$$\boldsymbol{w_i} = \begin{bmatrix} x_1, & y_1, & z_1, & \ldots, & x_n, & y_n, & z_n \end{bmatrix}^\top$$

Given subsequent frames $\boldsymbol{M'}$ which deviate the facial mesh from $\boldsymbol{M}$, $\boldsymbol{M'}$ can be expressed by equation 5.1, where $\boldsymbol{\Lambda}$ is a vector containing Blendshape Parameters.

$$\boldsymbol{M'} = \boldsymbol{M} + \boldsymbol{W\Lambda} \tag{5.1}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1, & \ldots, & \lambda_d \end{bmatrix}^\top$$

The Blendshape Parameters λ_i can be found by rearranging equation (5.1) and solving for $\boldsymbol{\Lambda}$ as shown in equation (5.2). The solution to this can be found by finding the pseudo-inverse for the Blendshape Axes $\boldsymbol{W}$.

$$\boldsymbol{\Lambda} = \boldsymbol{W}^{-1}(\boldsymbol{M'} - \boldsymbol{M}) \tag{5.2}$$

The template mesh used throughout this report is from the FLAME model [1] shown in Figure 5.4, while visualisation is achieved with the visualisation scripts from the VOCA model [2] which render the 3D model into a series of images, such as those shown in Figure 5.2.

5.2.3 Dataset Split

The dataset consists of 55000 samples. This is split into training, validation and test data with 20000/20000/15000 data points respectively. When training the model the training data will be shown to the model, the model will then be evaluated on the validation data which allows model hyperparameters to be adjusted to improve performance on this dataset. To ensure that the model has not been optimised just for the validation data and can generalise to unseen data, the final performance is reported on the test data, which is never evaluated on until the model has been tuned to optimum performance on the validation data.

5.3 Data Preprocessing

To aid training, the dataset is normalised to a range of [0,1] using equation (5.3) where x_{min} and x_{max} are the smallest and largest values in the dataset respectively (Figure 5.5b). While each Blendshape Axis is separate from one another, normalisation should

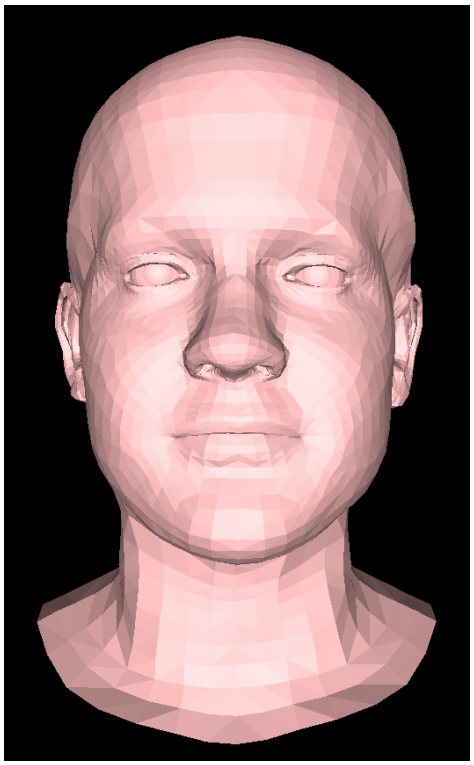
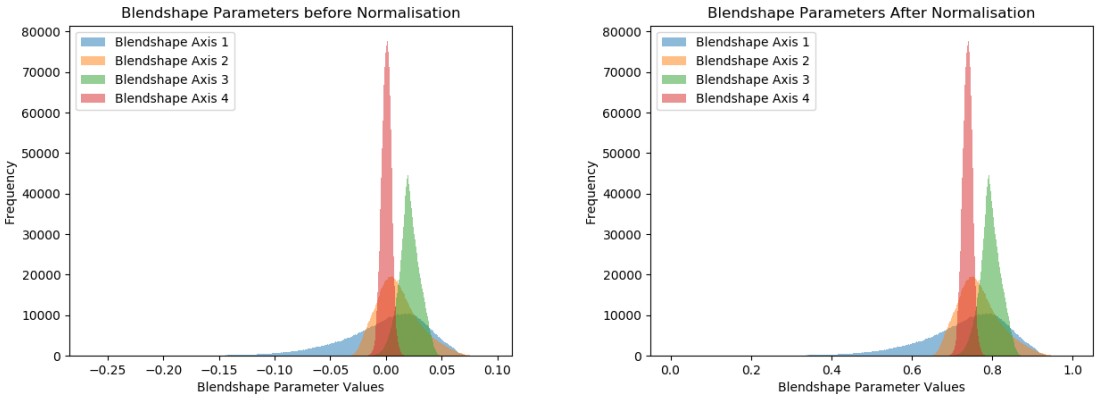


Figure 5.4: FLAME Template Mesh [1]

be performed on all parameters as a whole to preserve information on the variation found in each Blendshape Axis relative to one another. As described by PCA in section 2.1.1, the variation in each subsequent principal axis decreases which can be seen in Figure 5.5b.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.3}$$



(a) Unnormalised

(b) Normalised

Figure 5.5: Blendshape Parameter Histograms

5.4 Assessment of Model Performance

In order to assess the performance of the models in the task of word level classification, several metrics exist which could be used. The chosen metrics are the model accuracy and the Levenshtein Distance. The accuracy gives a measure of how frequently the model correctly predicts the word while the Levenshtein Distance represents how far the incorrect predictions are from the correct words.

5.4.1 Accuracy

The accuracy of a model is the ratio between the number of predicted labels made by the model which are correct and the total number of predictions. Accuracy is an appropriate metric in the instance when the dataset is balanced, that is to say that there are an equal number of all labels in the dataset.

If the dataset is unbalanced, the accuracy of a model is less informative. For example, in the scenario where there is a dataset with two labels A and B with the data being unbalanced such that label A occurs 9 times more frequently than label B. A model which achieves 90% accuracy on this dataset may be correctly labelling all data 90% of the time, or it may be labelling all data samples as label A without learning anything about the dataset and achieving the same accuracy.

For unbalanced data the F1-Score metric is more useful as this will remain low when a single label class is frequently being incorrectly predicted. However, as the dataset this problem is concerned with is balanced, this shall not be discussed.

5.5 Model Architecture

5.5.1 Multiple Towers Classification Model

The first model is based on the Multiple Towers architecture used in [4] used for lip reading on 2D temporal data with the LRW dataset described in Section 4.1.2. This is a CNN architecture which firstly applies a convolutional layer to the input image frames separately before concatenating the output feature maps and proceeding to convolve over the time domain. The original Multiple Towers architecture can be seen in Figure 4.5 while the architecture adapted from this design for prediction from Blendshape Parameters is shown in Figure 5.6

Model Input

The model inputs are in the form of Blendshape Parameter values for each frame in the motion sequence. These represent the interpolation from the template mesh as described in section 5.2.2 for each time frame. For a duration of one second there are

43 frames, 4 Blendshape Parameters per frame and a single channel dimension, such that the input to the model is $\boldsymbol{x} \in \mathbb{R}^{1\times4\times43}$

Loss Function

The model is trained with the Negative Log Likelihood (NLL) on 500 class labels for each of the labels in the dataset. The model makes word level predictions with the LogSoftmax activation function applied to the final layer of the model.

Model Architecture

Similar to the architecture proposed by Chung et al. [4], the first convolutional layers do not convolve over the time domain by using 3x1 kernels. These two layers reduce the input shape from [1×4×43] to [512×1×43], encoding the 4 Blendshape Parameters into a single value across multiple channels, allowing the single dimension to be collapsed and 1D convolutional layers to be used for subsequent layers.

The Multiple Towers architecture used by Chung et al. featured max pooling layers to reduce the size of feature maps. As a linear layer is to be used at the final layer of the model to facilitate label prediction with a LogSoftmax activation function, it is desirable to reduce the size of the feature maps to prevent the linear layer having too many parameters as this becomes difficult to train and can lead to overfitting. Max pooling layers reduce the size of feature maps in the network, only retaining the information which is of the most use to the model.

In order to aid in the training of the model, batch normalisation is applied after convolutional layers in all cases except for the final convolutional layer. Similarly, dropout is applied to all but the final convolution layer in order to assist the model from overfitting the training data. During training, the model suffered from significant overfitting to the training data, dropout was applied and assist in the prevention of this. In addition to dropout, weight decay was also used to improve performance on the validation data. Early stopping was used to prevent the model learning to overfit further on the training data. This was achieved by keeping a running average of the loss on validation data of the previous 10 epochs, training was stopped once the validation loss stopped decreasing (see Figure 5.8a).

A detailed breakdown of the model specification can be seen in Table 5.1. The hyper-parameters which provided the optimum results on the validation data are shown in Table 5.2

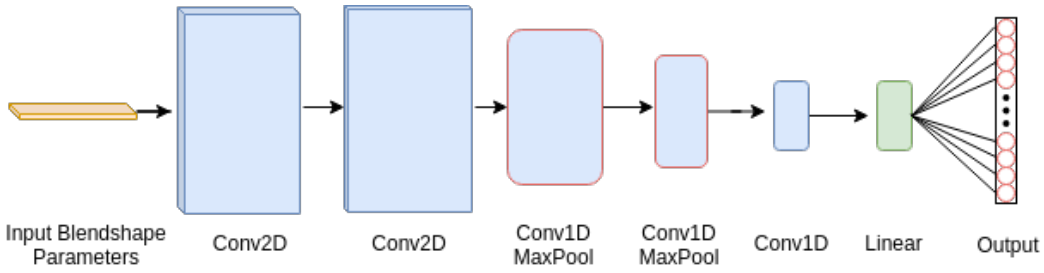


Figure 5.6: Multiple Towers Classification Architecture

Layer	Output	Kernel	Stride	Batch Norm	Dropout	Activation
Conv2D	512x2x43	3x1	1x1	True	0.2	ReLU
Conv2D	512x1x43	3x1	1x1	True	0.2	ReLU
Conv1D	256x41	3	1	True	0.2	ReLU
MaxPool	128x20	3	2	-	-	-
Conv1D	64x18	3	1	True	0.2	ReLU
MaxPool	64x9	3	2	-	-	-
Conv1D	64x7	3	1	False	0.0	ReLU
Linear	500	-	-	False	0.0	LogSoftmax

Table 5.1: Multiple Towers Model Architecture

Optimisation	Learning Rate	Weight Decay	Batch Size
RMSprop	0.0002	0.001	128

Table 5.2: Multiple Towers Model Hyperparameters

5.5.2 Blendshape Channels Classification Model

The second model applied to this task is not directly inspired by any existing model architecture. Unlike the Multiple Towers Model, this model convolves over the time domain throughout the whole depth of the model and treats each Blendshape Axis as a separate channel to the input of the model, thus the model shall be referred to as the Blendshape Channels Model. As with the previous classification model, NLL is used as a loss function.

Model Input

The model inputs, as before are in the form of Blendshape Parameter values corresponding to frames in the motion sequence. Unlike the previous model, each Blendshape Axis is treated as a separate input channel, such that the input data is in the form $\boldsymbol{x} \in \mathbb{R}^{4\times43}$. Treating the Blendshape Axes as separate channels allows the model to extract information from them separately while considering the time domain for each.

Model Architecture

The Blendshape Channel model is a CNN architecture convolving over the time domain of the input data throughout the whole depth of the model. Reducing the length of the feature maps as the model deepens, while increasing the number of channels in the feature maps to allow for more fine details to be extracted from the data.

In order to reduce the length of the feature maps in the model, max pooling layers are applied after the 3^{rd} and 5^{th} convolutional layers. As with the Multiple Towers Model, this forces the model to discard information which is of less importance, while reducing the size of the feature maps.

The final convolutional layer reduces the number of channels in the output feature map in order to reduce the number of parameters in the output linear layer for word level predictions with the LogSoftmax activation function. Full model specification is shown in Table 5.5.2. Similar to the Multiple Towers Model, batch normalisation is applied to aid in stabilising the training of the model, while dropout did not improve the performance on the validation data. The model was tuned on the validation data with the hyperparameters shown in Table 5.5.2.

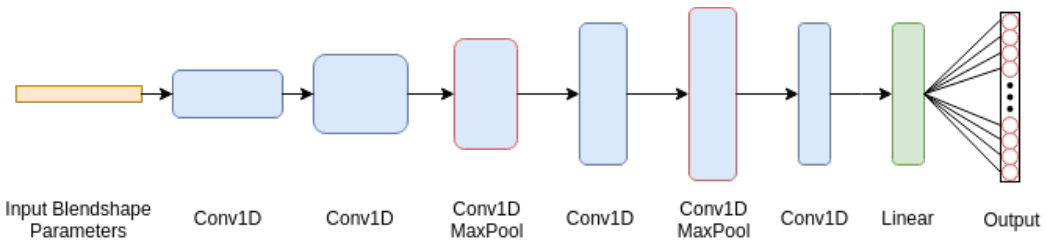


Figure 5.7: Blendshape Channels Classification Architecture

Layer	Output	Kernel	Stride	Batch Norm	Dropout	Activation
Conv1D	4x43	3	1	True	0.0	ReLU
Conv1D	32x41	3	1	True	0.0	ReLU
Conv1D	64x39	3	1	True	0.0	ReLU
MaxPool	128x18	3	2	-	-	-
Conv1D	256x16	3	1	True	0.0	ReLU
Conv1D	512x14	3	1	True	0.0	ReLU
MaxPool	512x7	2	2	-	-	-
Conv1D	256x5	3	1	False	0.0	ReLU
Linear	500	-	-	False	0.0	LogSoftmax

Table 5.3: Blendshape Channels Model Architecture

Optimisation	Learning Rate	Weight Decay	Batch Size
RMSprop	0.00005	0.005	128

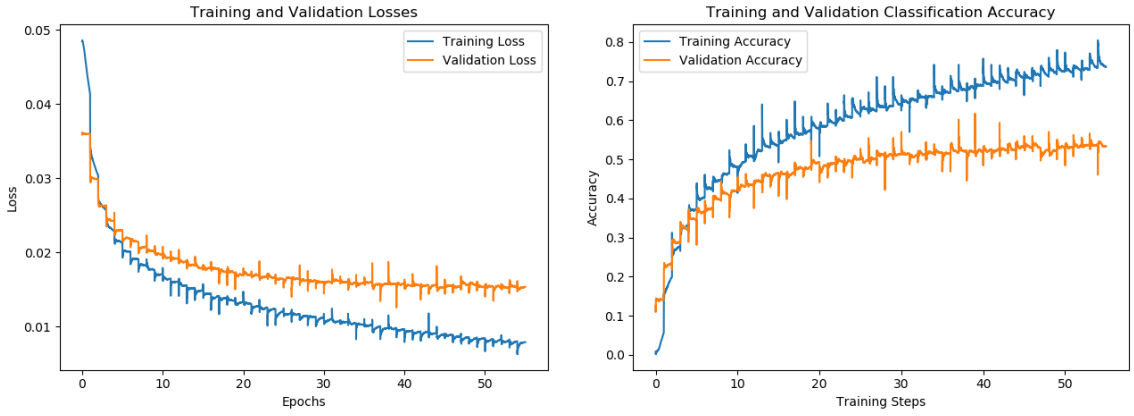
Table 5.4: Blendshape Channels Model Hyperparameters

5.6 Discussion of Results

Both models achieve an average test accuracy above 50% shown in Table 5.6. It is clear however that both models remain to overfit to the training data as can be seen in the training losses on the training and validation data as shown in Figures 5.8, 5.9. The model was tuned on the validation set to optimise the performance and reduce overfitting, this was achieved with weight decay parameters and dropout were applied to both models, however reductions in the deviation between training and validation losses resulted in an overall worse performance on the validation data. As mentioned in section 5.5.1, early stopping was applied to both models to halt training once the validation loss had stopped decreasing after ten epochs. The optimum performance on the validation data resulted in classification models which remain to overfit to the training data.

	Train	Validation	Test
Multiple Towers	0.7911	0.5333	0.5276
Blendshape Channels	0.8544	0.5504	0.5461

Table 5.5: Blendshape Classification Models Test Data Accuracy

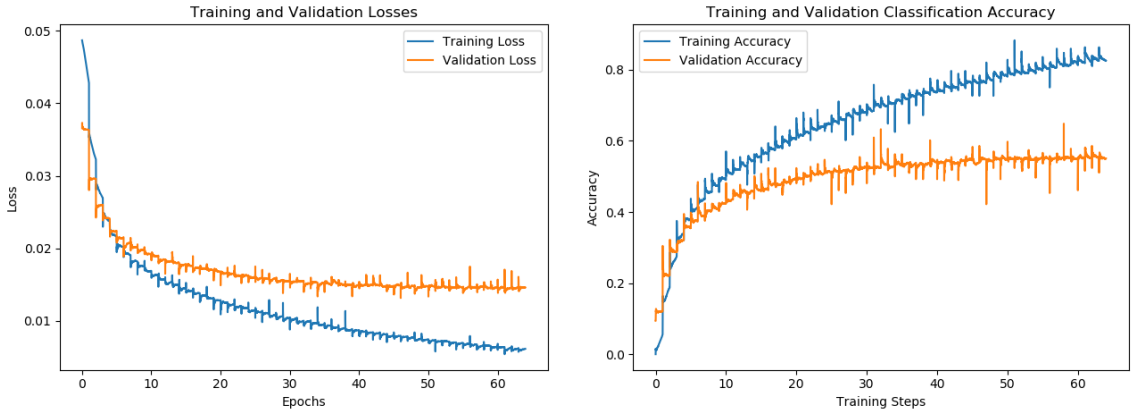


(a) Losses

(b) Accuracy

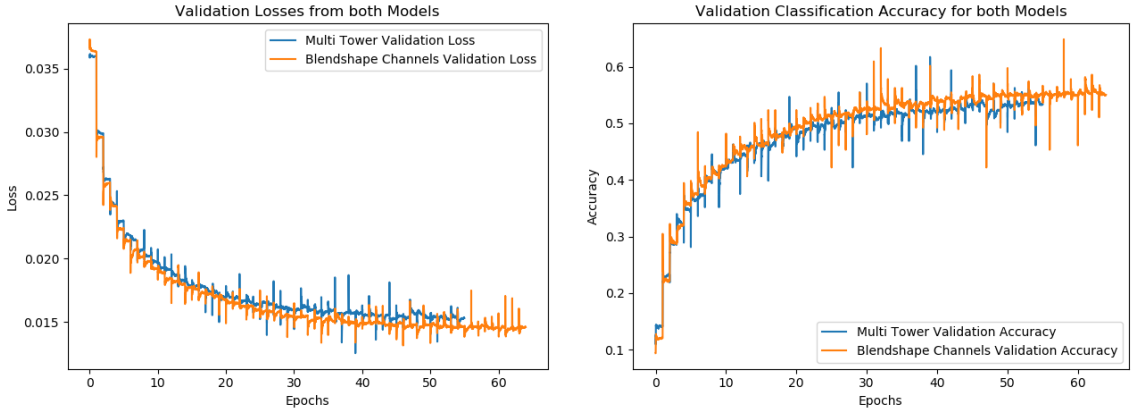
Figure 5.8: Multiple Towers Training

A direct comparison of the models on the validation data shows that the Blendshape Channel architecture slightly outperforms the Multiple Towers architecture throughout training (Figure 5.10). The final model accuracy values for each dataset split is shown in Table 5.6. Evaluation on the test data concludes that the Blendshape Channels model consistently outperforms, while the margin between the two models remains small.



(a) Losses (b) Accuracy

Figure 5.9: Blendshape Channels Training

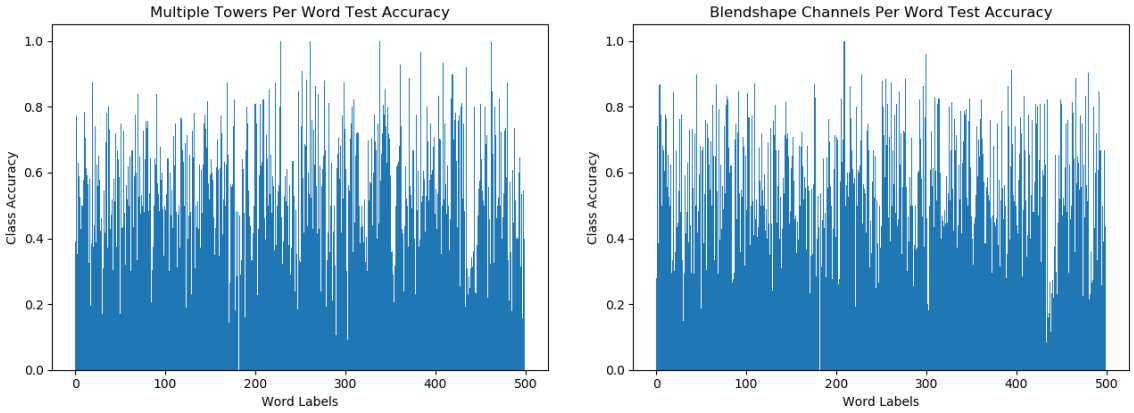


(a) Losses (b) Accuracy

Figure 5.10: Model Training Comparison

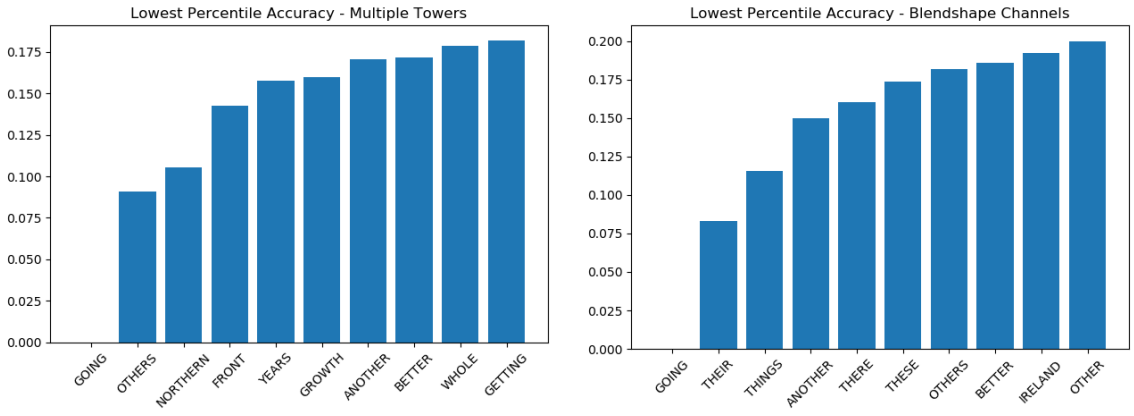
Examining word level classification accuracy on the test dataset (Figure 5.11) shows that both models perform similarly across all labels. Commonly for a classification task a confusion matrix would be displayed showing how the model is predicting individual labels. This highlights any labels which are commonly mislabelled. However as there are 500 labels, this confusion matrix is far too large to be useful for visualisation. The main labels of interest are those which are least frequently predicted, thus the 1st percentile of the test dataset is plotted instead, shown in Figure 5.12 This highlights that the most poorly predicted label was the same for both models: (*'GOING'*). While the second worse predicted label in both cases achieved 9.09% and 8.33% for the Multiple Towers and Blendshape Channels models respectively, far higher than the 0.2% accuracy which would be achieved by a model which makes a purely random guess from 500 labels.

The two models, while overfitting to the training data, perform well on the validation and test data, achieving over 50% mean accuracy on test data. Given that there are 500



(a) Multiple Towers (b) Blendshape Channels

Figure 5.11: Per Word Test Accuracy



(a) Multiple Towers (b) Blendshape Channels

Figure 5.12: 1st Percentile Per Word Test Accuracy

labels upon which to predict, this is a strong correlation between the input Blendshape Parameters and the words predicted. The results from the two models are very similar, such that neither can be determined as the superior architecture for this task.

5.7 Conclusions

This chapter has discussed the use of Blendshape Parameters as a means of word level prediction. Two machine learning models were evaluated, one which reduced the parameter values into a single dimension encoding before convolving over time while the other treated each Blendshape Parameter as a separate input channel and convolved over the time domain throughout the whole depth of the network. While neither architecture can be said to be superior as both achieved similar results, it can be concluded that there exists a very strong correlation between the Blendshape

Parameters obtained from a 3D temporal model of a speaking subject and the models spoken, such that word level prediction was possible.

Chapter 6

Audio Driven Blendshape GAN

This chapter shall discuss the use of GANs for audio driven facial animation. Unlike previous generative models which solve a regression problem to correctly match an audio input to a sequence of parameters which describe facial motion, GANs also include an noise input resulting in variations in the output from the same conditional input. The model outputs will then be assessed on the classification models discussed in Chapter 5. The GAN is comprised of two neural networks, a Generator and a Critic, which are trained together using the outputs of each network to train the other.

6.1 Problem Definition

Currently, there is a lack of 3D temporal datasets which sync speech audio with corresponding facial motion with the purpose of training a model for VSR. Such datasets are difficult to capture directly due to the equipment and time required. Similar datasets exist which have enabled for the creation of audio driven models but these datasets are limited in size and variation and have led to models which are specific to subjects either though direct subject specific training data as with the model by Karras et al. [5] or through character encoding, as is the case with the VOCA model [2]. Through the use of these models, a synthetic dataset of 3D temporal facial motion was generated as described in Section 5.2. This data however is subject to all biases and restrictions the VOCA model is subject to. An appropriate dataset would contain a small vocabulary, similar to the LRW dataset of 500 words, all spoken by multiple subjects to capture different speaking styles for the same words captured from '*in the wild*' scenarios.

An appropriate dataset would meet the following conditions:

- A small vocabulary of a similar size to the LRW dataset.
- Multiple subjects speaking the same vocabulary to capture different speaking styles for the same words.
- "*In the wild*" audio conditions.

As a means to increase the variation in speaking style in existing audio drive generative models such as VOCA, a GAN model is to be trained on the data generated by the VOCA model and the original LRW audio from which the VOCA generated dataset was driven from. The resulting generative model should be able to replicate the results of the VOCA model, but with increased variation in speaking styles due to the noise input component.

6.2 Model Inputs

The generative model is to be driven from an audio input of audio samples from the LRW dataset which were used as inputs to the VOCA model to generate the dataset of Blendshape Parameters. Audio data in the time domain has a high number of samples per second to allow all frequencies observable by human hearing to be captured. Human hearing can commonly hear up to 20kHz, which results in a sample rate of at least 40kHz to meet the Nyquist sampling rate, which states that the sample rate must be at least twice the desired maximum observable frequency to accurately represent the signal at this frequency. 40,000 samples for a single second is however, is a large number of samples and an inefficient data representation. Commonly audio data is converted to the frequency domain which allows for a far more efficient means of data representation.

6.2.1 Mel-frequency Cepstral Coefficients

Common implementations of machine learning models which use audio as in input, focused on speech, have used Mel-frequency cepstral coefficients (MFCCs) to represent audio, [34, 35, 36]. MFCCs were inspired by human anatomy and speech, invented by Davis and Mermelstein in 1980 [37]. Human speech is naturally filtered by the shape of the mouth and vocal tract. This envelope can be well represented by the short time power spectrum. A large amount of information is contained within an audio sample in the time domain, much of which does not contain meaningful information. By filtering the audio sample in the frequency domain to this envelope, information useful to speech detection is maintained while other information is discarded.

The first assumption is that over a short time frame, the audio signal does not statistically vary greatly. This assumption allows the signal to be split into short frames which can be processed separately. An example of the audio frames is show in Figure 6.1.

For each frame the power spectrum is calculated using the periodogram estimation. The incentive behind calculating the power spectrum is derived from human anatomy, specifically from the cochlea. The cochlea is a bone which resides within the ear and different parts of the bone vibrate depending on the frequency of the incoming audio signal. The periodogram estimation of the power spectrum performs a similar operation

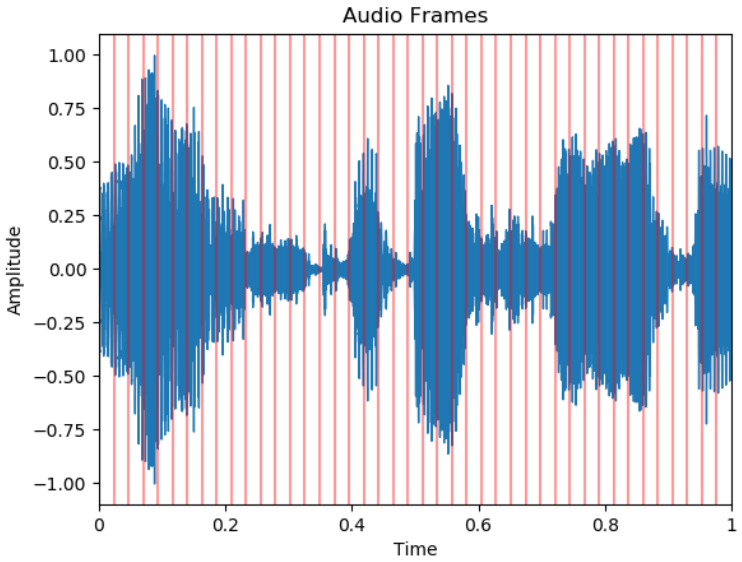
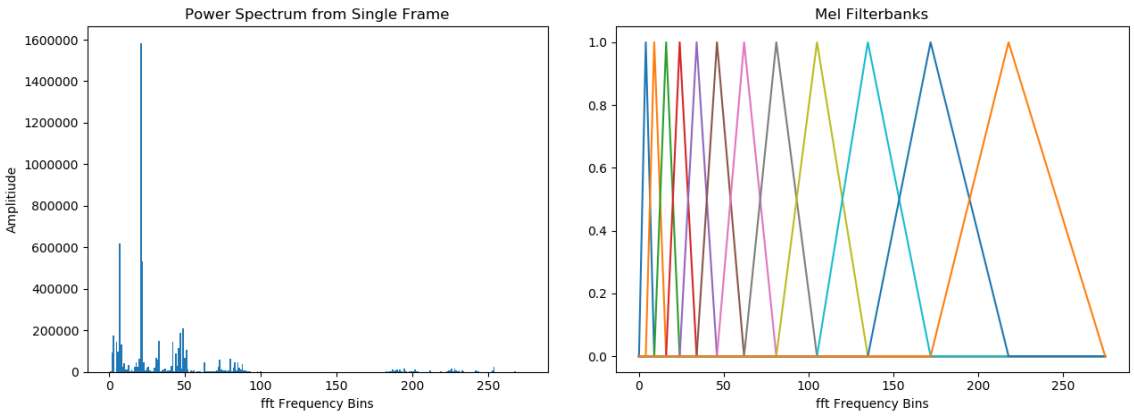


Figure 6.1: Audio Sample Frames

by identifying which frequencies are present within an audio signal. This representation still retains information which is not useful for speech recognition, for example the cochlea cannot distinguish between closely spaced frequency values, this is more apparent at higher frequencies. The power spectrum for a single audio frame is shown in Figure 6.2a. For this reason a series of Mel filterbanks are applied to the power spectrum and the resulting bin for each filter is summed, this gives an approximation of the energy present in each frequency region. The Mel filterbanks are visualised in Figure 6.2b. The log of each summation is then taken as humans do not perceive loudness on a linear scale, so the log compression matches the features more closely to what humans hear, shown in Figure 6.3a.



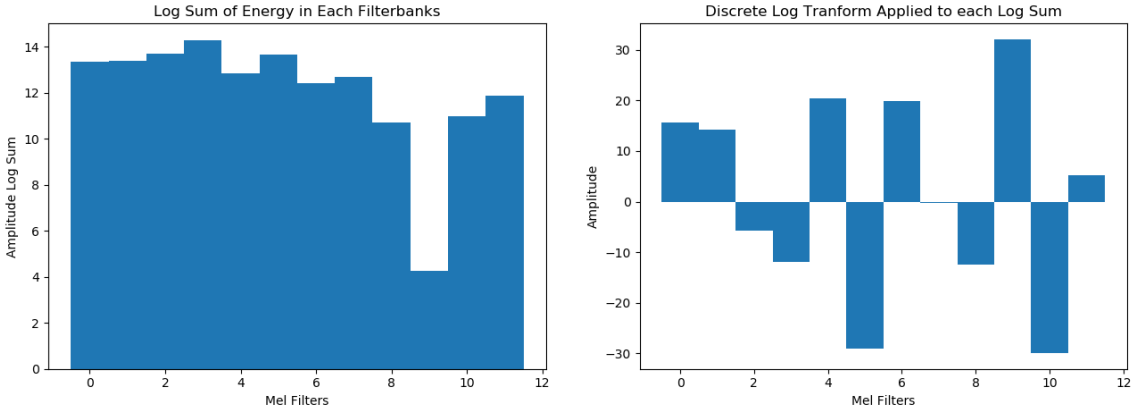
(a) Audio Frame Power Spectrum

(b) Mel Filterbanks

Figure 6.2: Mel Filterbank Processing

Lastly, the Discrete Cosine Transformation is applied to each filterbank energy, this

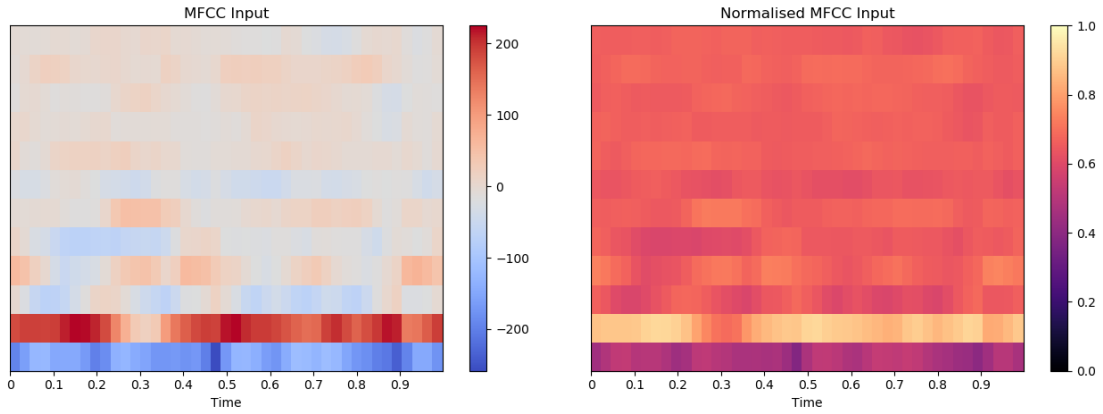
is performed to decorrelate the filterbanks as they are highly correlated due to the overlap in the Mel filters, the resulting representation is shown in Figure 6.3b. This represents the MFCC values for a single audio frame for 8 Mel filterbanks.



(a) Log Sum (b) Discrete Cosine Transformation

Figure 6.3: Discrete Cosine Transformation Processing

The model inputs use a total of 12 filterbanks over the span of one second audio clips. This results in a model input MFCC $\boldsymbol{x} \in \mathbb{R}^{12\times43}$, visualised in Figure 6.4.



(a) MFCC Model Input (b) Normalised MFCC Model Input

Figure 6.4: MFCC Model Input

6.2.2 Blendshape Inputs

The Generator aims to produce realistic Blendshape Parameters for the input audio signal, thus the input to the Critic must be the true Blendshape Parameter values and the 'fake' generated Blendshape Parameters. These values are the same as were input to the Blendshape Classification models discussed in Section 5.5.1: $\boldsymbol{y} \in \mathbb{R}^{4\times43}$. The reader is directed to this section for further details.

6.2.3 Noise Input

In addition to audio represented in the form of MFCCs, the Generator is also input with a noise component consisting of random samples taken from a uniform probability distribution within the range of 0 and 1, expressed by equation (6.1).

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \leq x \leq b \\ 0, & \text{elsewhere} \end{cases} \tag{6.1}$$

6.3 Data Preprocessing

As with the classification models, data inputs to the model are normalised within the range [0,1], the normalised MFCC input is shown in Figure 6.4b. This normalisation makes training both the critic and the generator easier as the range of values which the model will observe is reduced while preserving the statistical properties of the data. As the generated Blendshape Parameters are within the range [0,1] the Sigmoid activation function can be used at the output layer of the model, rather than a linear output with an infinite range.

6.4 Assessment of Model Performance

Assessing the performance of a GAN is less straightforward than other machine learning models. For example, in a supervised learning scenario, a classification model can be assessed on the accuracy of predictions or f-score on a test set in addition to the model loss on these predictions, as discussed in Section 5.4. As GANs describe an unsupervised learning scenario, assessment metrics are less clear and problem specific. In this problem, the aim of the Generator is to find a mapping from an audio signal represented by a series of MFCCs, to a series of Blendshape Parameters which the Critic considers to be a correct match. The Critic's goal is to be able to judge pairings of MFCCs and Blendshape Parameters as a realistic match.

A realistic pairing of audio and Blendshape Parameters would meet the following conditions:

- Movement would appear realistic to a human observer.
- Movement would appear correlated to the speech in the audio signal to a human observer.
- Speech would be correlated, such that a pretrained Blendshape classifier would be able to make some degree of successful predictions from the parameters.

The first two conditions are difficult to evaluate in a quantitative manner as this is a matter of opinion from the observer. A subjective evaluation can be conducted on

a collection of participants, however finding subjects who can accurately lip read is challenging due to the difficulty of the skill, especially in this scenario where there is no context for the spoken phrases. Subjects who cannot lip read could be used, however this is unreliable as in many cases a video clip which has been dubbed over with an audio signal, which has been roughly matched with a speaking subject may seem plausible. With the absence of audio, the 'realism' of the animation movement is also difficult to determine. While aspects of animation which easily make generated data identifiable are sudden movements and shaking in the animation which would not be apparent in real speech, a series of parameter values which produce a smooth animation may easily fool observers.

A quantitative metric which can be evaluated is the performance of generated samples on the pretrained Blendshape classifiers, the architecture and performance of which is discussed in Chapter 5. As the Generator aims to find the mapping between the input audio signal and Blendshape Parameters, if successful, the generated Blendshape Parameters should have properties similar to those of the original data which would allow for correct classification.

Given that the success of the Generator is dependent upon producing Blendshape Parameters which the critic cannot distinguish from real samples, the primary metric which shall be used is the losses on the validation set from both the Critic and the Generator during training. Training of the Generator will be complete when the Critic can no longer distinguish between real and fake samples. When this occurs the Critic losses have settled at 0.

6.5 Model Architecture

The GAN is composed of a Generator and a Critic, trained with the Wasserstein Loss with Gradient Penalty to ensure the Lipschtiz condition, described by equations (4.3, 4.4) respectively for the two models. More details on the loss function is discussed are Section 4.2.2.

6.5.1 Model Structure

The GAN model is constructed of two CNN architectures; a Generator and a Critic. A high level diagram of the GAN model is shown by Figure 6.5.

The Critic is trained on a corresponding pair of MFCC values and Blendshape Parameters. The goal of the Critic is to be able to identify a correct pairing from the dataset and a false pairing consisting of MFCC values and a fake series of Blendshape Parameters generated by the Generator. The Critic returns a score value within the range of [-1, 1], where data inputs which it regards as realistic are larger, while pairings which it regards as fake are smaller. For each minibatch in a training epoch the Critic is shown a batch of real data values $\boldsymbol{x}$ and the Generator creates a batch of fake data values $\tilde{\boldsymbol{x}}$. The Critic evaluates this data and returns a score for both the real and fake

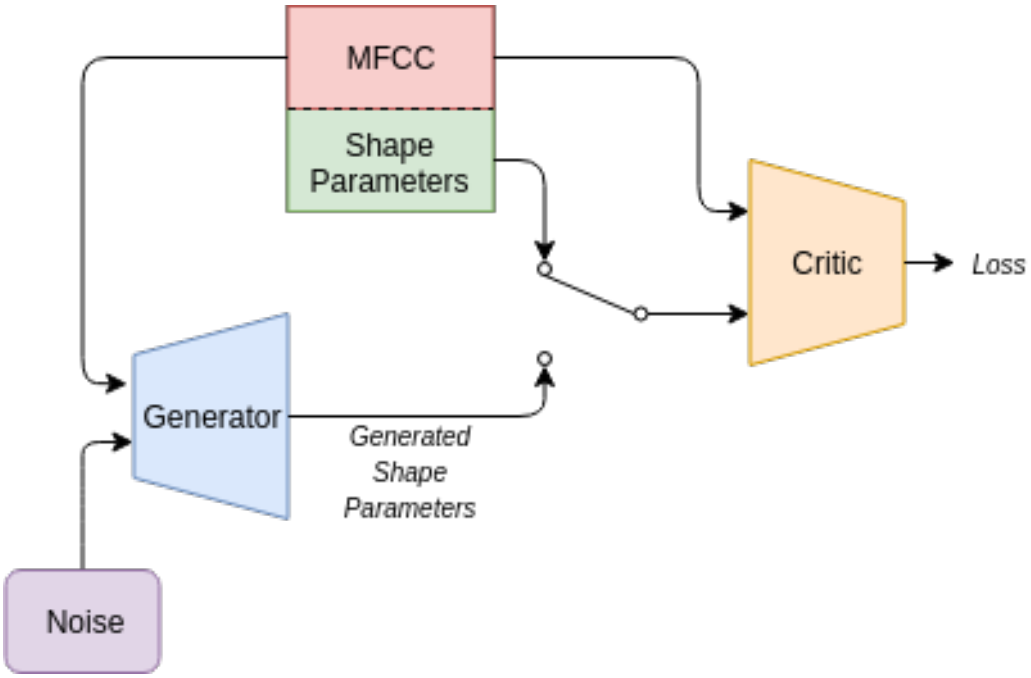


Figure 6.5: GAN Model

data. In addition to the loss from the Critic score on real and fake data samples, the gradient penalty term is calculated, this term is to be minimised to meet the Lipschtiz condition, [25]. This is found by evaluating the score of the Critic on a new data sample taken at random along the interpolation between the real and fake Blendshape Parameters, $\hat{\boldsymbol{x}}$. The gradient penalty term $\boldsymbol{k}$ can then be found with equation (6.2) with the interpolated value, where λ is a tunable hyperparameter.

$$\boldsymbol{k} = \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\| - 1)^2] \tag{6.2}$$

The Generator takes inputs of MFCC values and noise from a uniform distribution, these are concatenated together and processed as a single input. The Generator then attempts to map this input into Blendshape Parameter values which the Critic evaluated to a low loss value. During training, the samples produced by the Generator are assessed by the Critic. The Generator aims to have the samples it produces labelled as realistic, such that the Critic returns a high value.

At any point for the Generator to improve the quality of the samples it can produce, the Critic must initially be able to distinguish between the real and fake samples, else the Generator has no incentive to improve further. For this reason, the Critic is trained more frequently then the Generator, in this case the Generator is trained one in ten batches, while the Critic is trained on every batch, the data is shuffled so that the Generator is still exposed to the whole training dataset.

	Generator	Critic
Optimisation Algorithm	RMSprop	RMSprop
Learning Rate	0.00001	0.00001
Scheduler	0.999	0.999
Batch Size	128	128
Training Ratio	1:10	1:1
Gradient Penalty	-	5

Table 6.1: GAN Hyperparameters

6.5.2 Generator

The Generator aims to transform noise from a uniform distribution into realistic Blendshape Parameters given the conditional input of MFCC values for a given audio sample. Five channels of noise are sampled with the same shape as in MFCC input and the two are concatenated along the channel dimension, such that the input data is $\boldsymbol{z} \in \mathbb{R}^{6\times12\times43}$. The Generator model architecture is a fully convolutional network which takes the input values $\boldsymbol{z}$ and transforms this into Blendshape Parameters $\tilde{\boldsymbol{x}} \in \mathbb{R}^{4\times43}$.

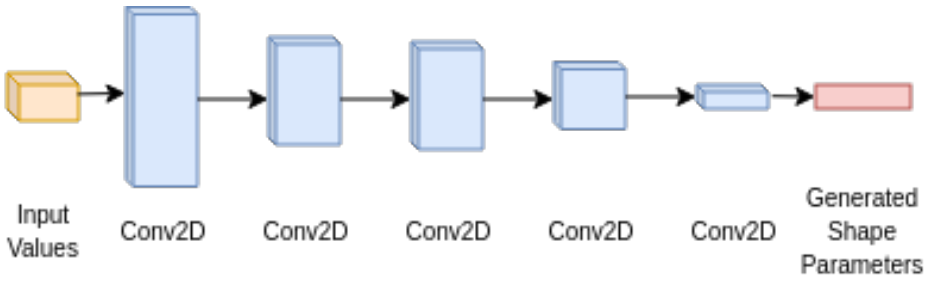


Figure 6.6: Generator Model Architecture

The model (shown in Figure 6.6) consists of five convolutional layers, the detailed specification of which is shown in Table 6.2. The first pads the input along the time domain as the output dimension along the time domain is to be the same as in input, corresponding to separate frames of Blendshape Parameters, without padding this dimension would be reduced due to convolution operations. This is a significant amount of padding to be applied to a single layer, however when a padding values of one was applied to all layers this resulted artifacts at the beginning and end of the generated sequence. The first layer has a high number of filter channels to extract the largest possible amount of information from the input data, following layers reduce the number of channels to compress this information into the final output layer which has 4 channels, one for each Blendshape Parameter.

The model uses ReLU activate functions for all hidden layers and the Sigmoid activation function at the output layer to achieve Blendshape Parameters in the range [0, 1]. The output layer also uses a larger kernel size than the hidden layers, which all use a kernel size of 3. When a small kernel was used at the output layer there existed a large amount of jitter in the output parameters, increasing the output kernel seems to eliminate this and produce animation with smoother motion. The role of this output

layer is not to extract useful information for subsequent layers to use, but to generate the output values. A larger kernel increases the receptive field of this layer on the feature maps from the previous layer, resulting in reduced variation between subsequent frames and less jitter.

Layer	Output	Kernel	Padding	Activation
Conv2D	256x10x53	(3,3)	(0,6)	ReLU
Conv2D	128x8x51	(3,3)	(0,0)	ReLU
Conv2D	128x6x49	(3,3)	(0,0)	ReLU
Conv2D	64x4x47	(3,3)	(0,0)	ReLU
Conv2D	4x1x43	(4,5)	(0,0)	Sigmoid

Table 6.2: Generator Model Architecture

6.5.3 Critic

The Critic aims to distinguish real data samples from the training dataset from data samples from the Generator. The high level overview of the model architecture is shown in Figure 6.7 and listed in detail in Table 6.3. Inputs to the Critic are comprised of MFCC values and Blendshape Parameters, both of which have different dimension sizes, but share a common time dimension. There are 12 MFCC filter values and 4 Blendshape Parameter per time instance. In order to concatenate these values together, the Blendshape Parameter values are treated as a separate channel and duplicated 12 times, such that the Blendshape Parameters are expressed as $\boldsymbol{b} \in \mathbb{R}^{4\times12\times43}$. This is then concatenated with the MFCC values $\boldsymbol{m} \in \mathbb{R}^{1\times12\times43}$, into the input values $\boldsymbol{x} \in \mathbb{R}^{5\times12\times43}$.

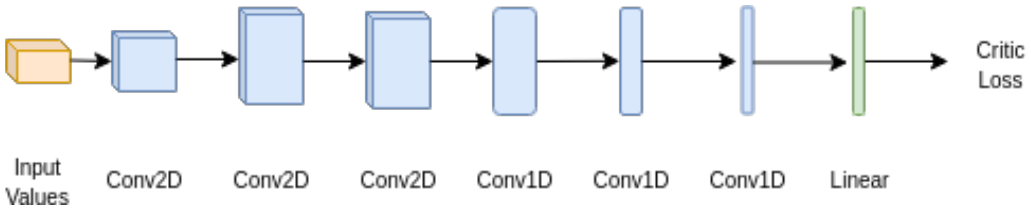


Figure 6.7: Critic Architecture

The Critic model consists of three sections, the first two are built of convolutional layers using the LeakyReLU activation function while the last output layer, a linear layer using the Tanh activation function. The first three layers are two dimensional convolutional layers which do not convolve over the time domain, just the height dimension. This allows the model to find an efficient encoding of the input audio and Blendshape Parameters. The number of filters is increased throughout this encoding stage as opposed to extracting a large number of filters initially as was the case with the Generator. The reasoning behind this is that the initial input contains a great deal of repetition due to the Blendshape Parameters being duplicated into the same shape as the MFCC values.

The second section aims to convolve over the time domain, as the singleton dimension can now be collapsed, one dimension convolutional layers can now be used to achieve this. The section consists of three convolutional layers, each of which have a stride of two and a decreasing kernel size across the three layers. The first layer uses a kernel size of 5 in order to capture higher level features while the subsequent layers have kernel sizes of 4 and 3 respectively to capture more fine detailed features. The final layer is a fully connected layer, which provides the critic score for the input data sample. The layer uses the Tanh activation function to output a score in the range of [-1, 1].

Layer	**Output**	**Kernel**	**Stride**	**Activation**
Conv2D	64x5x43	(4,1)	(1,1)	LeakyReLU
Conv2D	128x3x43	(3,1)	(1,1)	LeakyReLU
Conv2D	128x1x43	(3,1)	(1,1)	LeakyReLU
Conv1D	128x20	5	2	LeakyReLU
Conv1D	128x9	4	2	LeakyReLU
Conv1D	64x4	3	2	LeakyReLU
Linear	16	-	-	Tanh

Table 6.3: Critic Model Architecture

6.6 Discussion of Results

Examining the training losses of the Generator and Critic, the performance of the model can be deduced. The first condition of the Critic is that the Lipschtiz condition is met, enforced by the gradient penalty term, by examining Figure 6.8 this can be seen to reduce rapidly and remain close to zero throughout training.

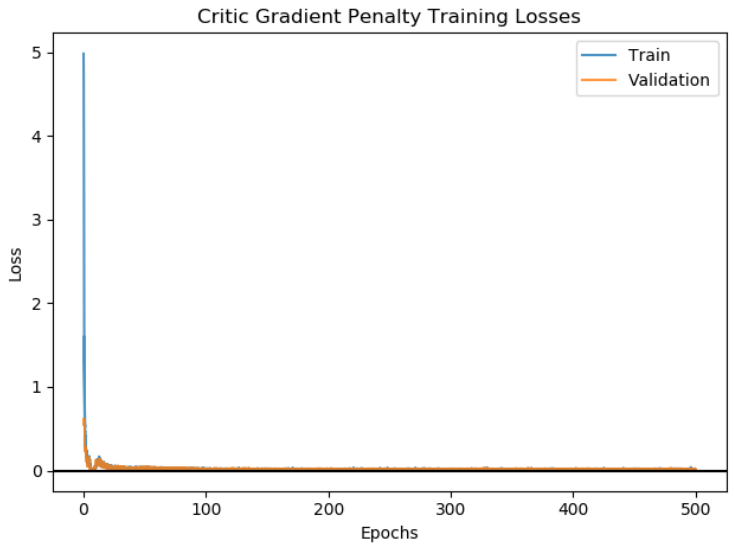


Figure 6.8: Critic Gradient Penalty Training Loss

Secondly, the total losses of the Critic as described by equation (4.4) are shown in Figure 6.9. Due to the high initial value of the gradient penalty term, the initial terms have not been plotted for this or subsequent plots to aid visualisation. The Generator and Critic both aim to have as low a loss as possible, a negative Generator loss is an indication that the fake samples are being scored incorrectly by the Critic as it considers the fake Blendshape Parameters as a correct pairing for the incoming MFCC values, as described by equation (4.3). A negative values for the total loss of the Critic is an indication that on average the real data samples are being given a higher score than the fake samples. However, as the loss from the Critic is a summation of multiple terms, this plot alone does not display all information about the model. From Figure 6.9 it can be seen that throughout the whole training process the Critic correctly scores real and fake input data.

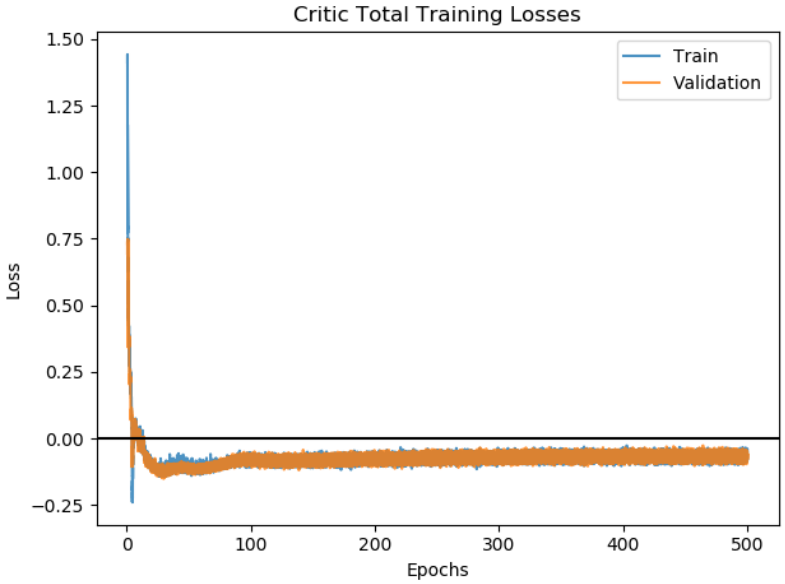
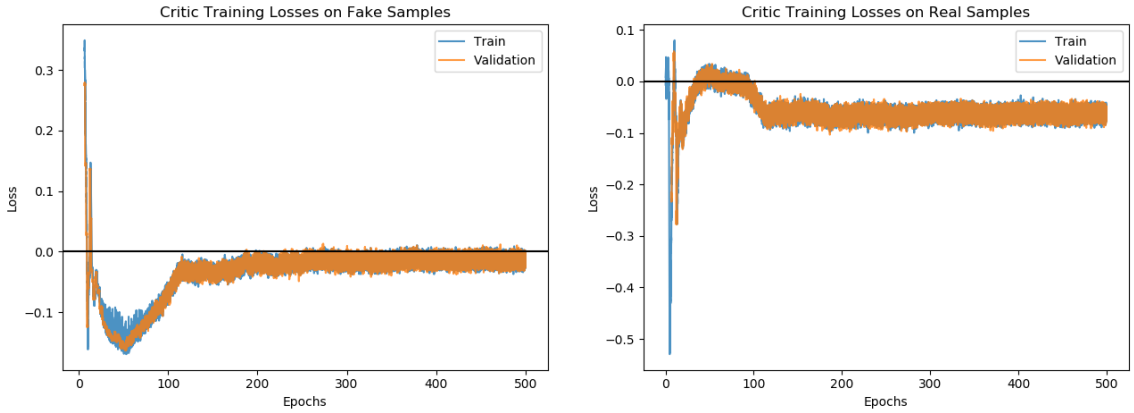


Figure 6.9: Critic Total Training Losses

In order to gain a better insight of the Critic and Generator's training losses, the loss on the real and fake data samples can be plotted during the training process. The Critic aims to score real samples positively and fake samples negatively, such that the total loss will be negative. From equation (4.4) the first term $\mathbb{E}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})]$ describes how the Critic evaluates fake data samples produced by the Generator, this is plotted in Figure 6.10a. The Generator training losses are also plotted in Figure 6.11 and can be seen to be a mirror of the Critic's evaluation of generated samples. From the two plots the Generator can be seen to be gradually improving throughout training, although the the losses never completely fool the Critic. The Critic settles on a consistent loss for real data samples shown in Figure 6.10b as described by the second term in equation (4.4): $-\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})]$.

Training settles on the Critic being able to distinguish real samples from fake, although the confidence in these predictions remains low throughout training while the confidence in identifying the fake samples gradually decreases throughout training. While



(a) Fake Data Samples (b) Real Data Samples

Figure 6.10: Training Losses from Critic for Real and Fake Data Samples

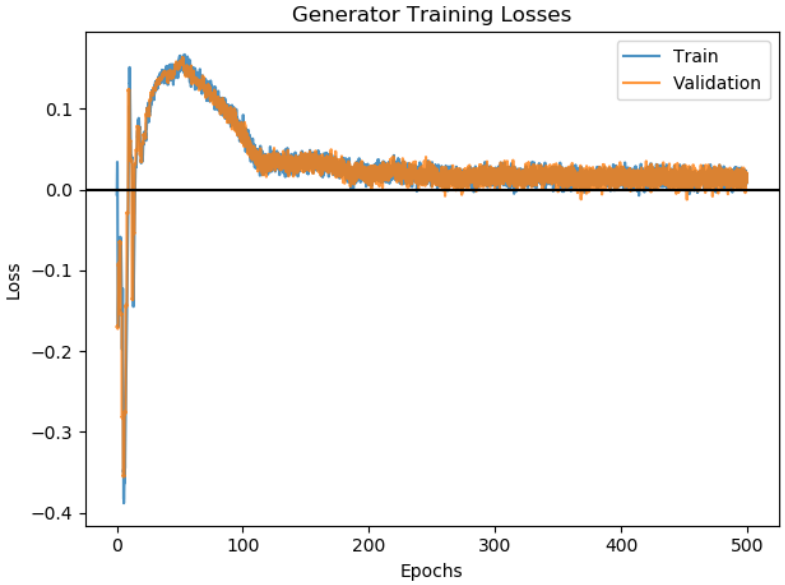


Figure 6.11: Generator Training Losses

the Generator remains unable to produce samples which completely fool the Critic, the generated results may still exhibit the desired qualities as set out by the assessment conditions in Section 6.4.

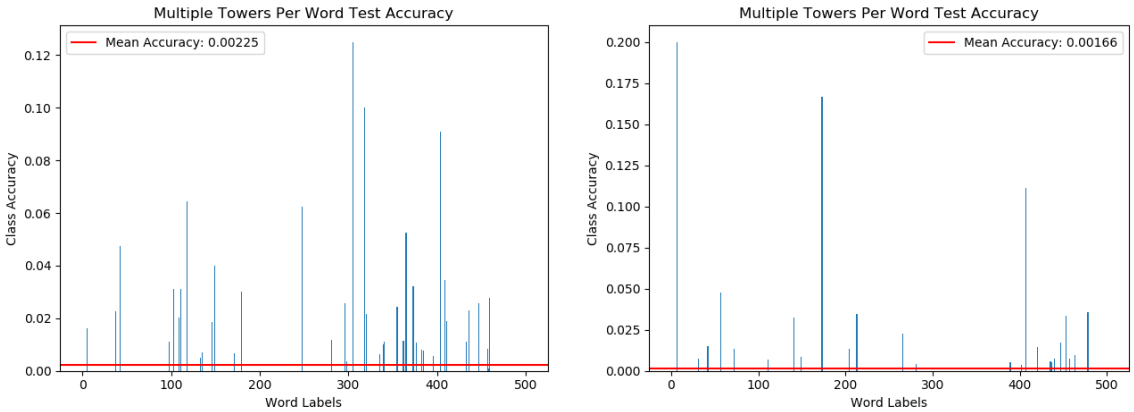
Given that the samples generated by the model are video, this is difficult to represent in a report. Odd frames from a single generated sample labelled as "*America*" are shown in the Appendix 7.3, but this is a poor means of evaluating the first two assessment conditions stated in Section 6.4. In light of this, a selection of generated samples is hosted on YouTube under an account held by the Author. Such samples can be found at: `https://www.youtube.com/watch?v=zoboRQyqt5Q&t=5s`.

These video samples allow for the evaluation of the first two assessment conditions. As stated previously, both of these conditions are open to interpretation by the viewer,

however the movement appears to be well correlated with the speech to a human observer. The generated samples arguably do not appear to be realistic, falling into the uncanny valley, although the original data from the VOCA model was also criticised for this [2], the lack of facial motion around the eyes is a strong indicator. This is a quality from which the Generator could not learn due to the lack of movement in this part of the face in the dataset. Eye movements, while being correlated with speech expression, do not contain information as to the words spoken, but rather enhance the meaning and expression, however it was not the intention of this work to capture such detail.

While it can be argued that the first two conditions are met by the Generator, if the lack of overall facial motion is ignored and the 'realism' is focused on the animation of the mouth, humans find lip reading to be a highly difficult task. Facial motion which is strongly correlated with audio without being correlated with the specific words spoken may easily fool viewers that the two sources are a matching pair. A quantitative means to measure if there exists a correlation between the facial motion and the spoken words can be evaluated by the use of the pretrained classification models described in Chapter 5. If the Generator has learnt to correlate speech data encoded by MFCC values to facial motion, the classification models should be able to recognise these features and provide a classification accuracy above the performance of a random assignment of a label; 1/500.

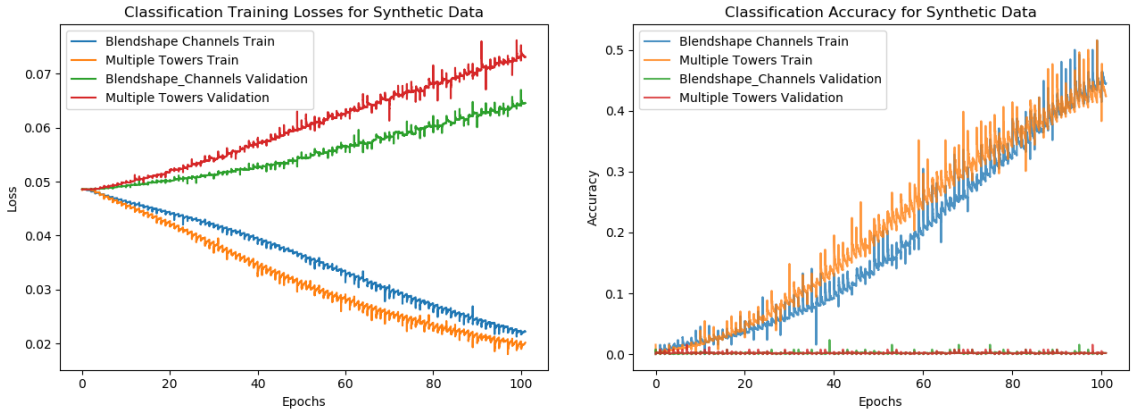
Evaluating the class accuracy on the Multiple Towers Model (Section 5.5.1) and the Blendshape Channel model (Section 5.5.2), highlights that there there seems to be very little correlation between the generated samples and the speech motion features identified by the two classifiers (Figure 6.12). The mean accuracy of both models is close enough to 1/500 to suggest that the Generator has not broadly learnt to produce facial motion correlated to the words spoken. Although both the models contain a small handful of word labels which are predicted with an above random accuracy, given that this behaviour is not consistent across many labels, this does not show any strong correlation.



(a) Multiple Towers (b) Blendshape Channels

Figure 6.12: Classification Class Accuracy for Generated Data

While the pretrained classification models did not identify any speech features corresponding to the word labels in the generated samples, this does not definitively mean that the data is completely uncorrelated to the speech, just that the features which the classifiers have learnt to identify are not present. To further investigate if there exists some correlation in the data, the two classification models have been retrained on the synthetic data. The results however, confirm that there seems to be no correlation across the synthetically generated data. Figure 6.13a demonstrates that both the Blendshape Channels and Multiple Towers models fail to generalise to unseen data completely, only able to learn features related to the training data. The prediction accuracy further highlights this in Figure 6.13b, as accuracy on the validation data remains at 0.002 throughout training, equivalent to random guessing.



(a) Training Losses

(b) Training Accuracy

Figure 6.13: Classification Models Trained on Generated Data

6.7 Conclusions

This chapter has discussed the use of a GAN model to generate synthetic Blendshape Parameters given an audio input upon which to drive a model. The audio was represented by the use of MFCC values, which was combined with noise as an input to the Generator model. The Critic was input with audio in the same MFCC representation on which the Generator was trained and corresponding Blendshape Parameter values from the original dataset and generated Blendshape Parameters from the Generator. The Critic learnt to distinguish the real pairing of Blendshape Parameters and audio from the fake successfully, while the Generator attempted to learn to produce realistic Blendshape Parameters to fool the Critic.

While the generated samples appeared to be correlated with the audio to a human observer, evaluation of the pretrained classification models from Chapter 5 highlighted that there was no correlation between the generated Blendshape Parameters and the words spoken in the audio samples. This further highlights the difficulties of lip reading

as a task for humans while the use of Blendshape Parameters for prediction is possible for a machine learning model.

Chapter 7

Conclusions and Future Work

This chapter shall discuss the successes and failures of the project as a whole, both in terms of the successes of the individual models trained and the implications of these findings. In light of this, the steps which can be taken in future work shall be explored in both the areas of lip reading with blendshape parameters and generative models to produce 3D temporal data for the purpose of lip reading tasks.

7.1 Blendshape Classification

This report aimed to investigate as to whether 3D facial data includes information which is of use for word level speech prediction. To achieve this, firstly a lower dimension representation of a 3D facial mesh was constructed with the use of Principal Component Analysis, referred to as the Blendshape Axes. Each facial mesh could then be expressed as an interpolation along each Blendshape Axis from a template facial mesh, expressed by the Blendshape Parameters. This representation of the 3D facial meshes allows for word level prediction for the 500 labels found in the Lip Reading Words Dataset with a mean test accuracy above 50% for all model architectures investigated. This demonstrates that firstly, there exists a strong correlation between the facial motion and the words spoken and secondly that this low dimension representation of the data effectively captures this data.

7.1.1 Future Work

The classification models are currently limited to word level prediction of a fixed sequence length and are unable to predict unknown word labels. A future extension of this work would be to build a model which can perform character level prediction of variable sequence length. This would allow for the prediction of entire sentences end to end, of variable sequence length. In order to achieve this, a new dataset would have to be constructed containing sequence frames labelled on a character level. This could be

achieved with the existing dataset by processing the audio data with a speech recognition system capable of character level prediction per frame, such as the DeepSpeech model.

Secondly, it should be noted that as a large amount of human lip reading relies on the context of the word in the current sentence, incorporating this context, either at a word or character level would likely prove beneficial to model performance.

Given that the classification models have successfully shown that there does exist a correlation between facial motion and speech, a practical application for these findings would be to apply visual speech prediction in a noisy environment where traditional audio speech recognition may perform poorly. A suggested implementation of this would be to combine predictions from both audio speech recognition and visual speech recognition with the means of a depth camera. The fusion of these two predictions produces an aggregate model, increasing the likelihood of correct predictions.

7.2 Generative Models

In addition to investigating the use of 3D facial meshes as a means of speech prediction, this report also investigated the use of a Generative Adversarial Network (GAN) to generate Blendshape Parameters driven by an audio input. The primary incentives behind this were that the acquisition of 3D temporal data is currently difficult which resulted in the use of existing audio driven facial animation models to construct the dataset on which all models were trained. This data however is limited in the variations of facial animation for given audio inputs. The intention of this model was to be able to produce realistic facial animation from an audio input which contained larger variations in facial motion from or original dataset. This would allow the dataset to be augmented with generated samples such that classification models may learn to generalise better to unseen data. The model however proved unsuccessful in achieving this goal. Generated samples appeared to be correlated with audio samples to human observers, but failed to contain features which are correlated to the words spoken within the audio itself when evaluated with the classification models trained beforehand. Regardless of the failure of this generative model, generating synthetic data remains an area which should be further pursued while data acquisition of 3D facial scans remains a large barrier to entry.

7.2.1 Future Work

The generative and classification models in this report were trained on data not directly captured due to the difficulties of obtaining such data for the purpose of lip reading. A primary area of work would be to directly obtain such a dataset, this would allow for a better assessment of the use of GANs as a means of generating synthetic data.

The GAN trained in this report was trained with a Critic used to distinguish between real and fake data samples given an audio input and the real or fake generated Blend-

shape Parameters. An extension of this would be to train the Generator on both the loss from the Critic and a pretrained classification model. This may allow for the Generator to learn the features which are present in the data which are correlated to the speech in the audio. However, this risks the Generator learning to trick the classification model without learning to generalise to unseen audio samples. Given that the current mean test accuracy of the classification models trained in this report is around 50%, this may not be an effective means to force the Generator to learn the features which it has currently failed to learn.

An alternate method of training the GAN would be to drive the generation from a character level conditional input, as opposed to audio driven conditional input. This would require the model to be able to be trained on variable length input sequences, which in turn would require a dataset labelled on a character level as discussed in Section 7.1. This could be trained either in combination with an audio input on without.

Ethical Considerations

As with any piece of work, there exists the potential for the work to be used with malicious intentions which were not the intentions of the authors. The efforts of improving visual speech recognition allow for the attainment of conversations between individuals who may be believed to be engaging in private discussion. The scenario in which lip reading accuracy reaches a high reduction accuracy and the tools for making such predictions become accessible to both the public and commercial entities, an environment in which you can be seen but not heard may no longer exist. This could be widely considered a breach of Article 8 of the European Convention on Human Rights [38] by undertaking "intrusive surveillance" against an individual.

The models developed in this report however currently require full head scans requiring a multiple camera set-up, followed by considerable processing to construct the facial meshes. However, the principles investigated here could be implemented with a single depth camera, although the quality of the results cannot be commented on without further work. Due to the symmetrical nature of faces, it may be found that a capture from a single side of a face may allow for the inference of the unseen face, enabling Blendshape Parameters to be estimated and speech predicted from this. As the public availability of these cameras becomes more widespread, this may pose a greater threat to the public.

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓

Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent / criminal / terrorist abuse?	✓	
Does your project involve information on/or the use of biological, chemical, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓

Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Appendix

Blendshape Axis Interpolation

Figure 7.1: Third Blendshape Axis Interpolation

Figure 7.2: Fourth Blendshape Axis Interpolation

GAN Generated Samples

Figure 7.3: Generated Sample - “AMERICA”

Bibliography

[1] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), 2017. pages vii, 3, 4, 19, 22, 25, 35, 36

[2] Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael Black. Capture, learning, and synthesis of 3D speaking styles. *Computer Vision and Pattern Recognition (CVPR)*, 2019. pages vii, 1, 2, 9, 22, 24, 25, 26, 31, 32, 35, 45, 57

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. pages vii, 13, 16, 17, 18

[4] Joon Son Chung and Andrew Zisserman. Lip reading in the wild. In Shang-Hong Lai, Vincent Lepetit, Ko Nishino, and Yoichi Sato, editors, *Computer Vision - ACCV 2016 - 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II*, volume 10112 of *Lecture Notes in Computer Science*, pages 87–103, 2016. pages vii, 1, 19, 20, 21, 22, 23, 25, 30, 31, 32, 33, 37, 38

[5] Tero Karras, Timo Aila, Samuli Laine, Antti Herva, and Jaakko Lehtinen. Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Trans. Graph.*, 36(4):94:1–94:12, 2017. pages vii, 4, 25, 32, 45

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. pages vii, 26, 27, 28

[7] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. Lipnet: Sentence-level lipreading. *CoRR*, abs/1611.01599, 2016. pages 1, 23, 24

[8] Joon Son Chung, Andrew W. Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3444–3453. IEEE Computer Society, 2017. pages 1, 19, 20, 24

[9] Brendan Shillingford, Yannis M. Assael, Matthew W. Hoffman, Thomas Paine, Cían Hughes, Utsav Prabhu, Hank Liao, Hasim Sak, Kanishka Rao, Lorrayne Bennett, Marie Mulville, Ben Coppin, Ben Laurie, Andrew W. Senior, and Nando de Freitas. Large-scale visual speech recognition. *CoRR*, abs/1807.05162, 2018. pages 1, 19, 20, 21, 24, 30

[10] Panagiotis Tzirakis, Athanasios Papaioannou, Alexander Lattas, Michail Tarasiou, Björn W. Schuller, and Stefanos Zafeiriou. Synthesising 3d facial motion from "in-the-wild" speech. *CoRR*, abs/1904.07002, 2019. pages 1, 22, 24, 25

[11] J. P. Lewis and K. Anjyo. Direct manipulation blendshapes. *IEEE Computer Graphics and Applications*, 30(4):42–50, July 2010. pages 4

[12] W. J Krzanowski. *Principles of multivariate analysis : a user's perspective*. Oxford statistical science series. Clarendon Press, Oxford, revised edition. edition, 2000. pages 6

[13] Martin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao. An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America.*, 120(5):2421–2424, 2006. pages 19, 20, 23

[14] Robert S Bolia, W Todd Nelson, Mark A Ericson, and Brian D Simpson. A speech corpus for multitalker communications research. *The Journal of the Acoustical Society of America*, 107(2):1065–1066, 2000. pages 20

[15] Patrick Buehler, Andrew Zisserman, and Mark Everingham. Learning sign language by watching TV (using weakly aligned subtitles). In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 2961–2968. IEEE Computer Society, 2009. pages 20

[16] Philip C. Woodland, C. J. Leggetter, Julian Odell, Valtcho Valtchev, and Steve J. Young. The 1994 HTK large vocabulary speech recognition system. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995*, pages 73–76. IEEE Computer Society, 1995. pages 20

[17] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 07 2009. pages 21

[18] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1867–1874. IEEE Computer Society, 2014. pages 21

[19] Hank Liao, Erik McDermott, and Andrew W. Senior. Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 368–373. IEEE, 2013. pages 21

[20] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. pages 22, 25

[21] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. pages 23

[22] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561. The Association for Computational Linguistics, 2016. pages 24

[23] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. pages 27, 28, 29

[24] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018. pages 27, 29

[25] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. pages 27, 29, 51

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. pages 28

[27] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. pages 28, 29

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. pages 29

[29] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. pages 29

[30] K. Vougioukas, S. Petridis, and M. Pantic. End-to-End Speech-Driven Facial Animation with Temporal GANs. *arXiv e-prints*, May 2018. pages 29

[31] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. pages 29

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010, 2017. pages 29

[33] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org, 2015. pages 29

[34] M. Holmberg, D. Gelbart, and W. Hemmert. Automatic speech recognition with an adaptation model motivated by auditory processing. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):43–49, Jan 2006. pages 46

[35] B. Milner and X. Shao. Prediction of fundamental frequency and voicing from mel-frequency cepstral coefficients for unconstrained speech reconstruction. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):24–33, 2007. pages 46

[36] K. S. R. Murty and B. Yegnanarayana. Combining evidence from residual phase and mfcc features for speaker recognition. *IEEE Signal Processing Letters*, 13(1):52–55, Jan 2006. pages 46

[37] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, August 1980. pages 46

[38] *European Convention on Human Rights*. Council of Europe. pages 63