Audio-Driven Head Pose with Deep Learning

Nikolay Nikolov

A thesis submitted in fulfilment of the requirements
for the degree of MSc in Computer Graphics, Vision and Imaging

Supervised by:
Professor Lourdes Agapito

University College London, UK

September 2020

I, Nikolay Nikolov, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Automatic synthesis of human body motion is becoming an important component of digital media and communications. Currently, this is still a task requiring specialised software and a laborious process. Using audio speech as the driving factor to produce motion data is a desirable option due to the flexibility and availability of speech data. This project focuses on devising a data-driven system using state-of-the-art *Deep Learning* for mapping audio speech to head pose, made up of a translational and a rotational component.

Data-driven systems require substantial amounts of high-quality curated and pre-processed data to be able to facilitate learning. Thus, a dataset from data *in the wild* was constructed, by initially scraping online footage of a single source actor, then annotating and analysing it with an off-the-shelf facial analysis tool.

Videos of the final results can be seen at the project website[1].

[1] http://nick-nikolov.com/masters-project

Acknowledgements

I would like to express my huge gratitude to Professor Lourdes Agapito for the continuous support and encouragement over the course of my project, especially in the strange and uncertain times of 2020.

I would also like to thank my friends and family, and especially my partner Ellen, for the unlimited supply of patience, support and much-needed positivity.

Table of Contents

List of Figures

List of Tables

Abbreviations

DNN	**D**eep **N**eural **N**etworks
DAE	**D**enoising **A**uto **E**ncoder
MLP	**M**ulti **L**ayer **P**erceptron
PCA	**P**rincipal **C**omponent **A**nalysis
DFC	**D**iscreet **F**ourier **T**ransform
DCT	**D**iscreet **C**osine **T**ransform
CGI	**C**omputer **G**enerated **G**raphics
CNN	**C**onvolutional **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
LSMT	**L**ong **S**hort **T**erm **M**emory
ReLU	**Re**ctified **L**inear **U**unit
GRU	**G**ated **R**ecurrent **U**nit
MFCC	**M**el **F**requency **C**epstral **C**oefficient

Chapter 1

Introduction

1.1 Overview

As our lives become ever more digitised, designers and artists look for new, easier ways to humanise our digital environments. This usually requires mimicking different aspects of human behaviour in the context of social communication, both verbal and non-verbal. With the continuous and steady increase in internet connection bandwidth and speeds as well as the rapidly improving quality of digital displays, media consumption habits have shifted primarily to video or other highly visual formats, such as digital avatars, facial augmentation (also called face filters, as seen in Figure 1.1), augmented reality and so on. This shift also increasingly puts the end-user in a creative and expressive role, blending the creation and consumption aspect of digital media.

Given the importance of the human head and face for non-verbal communication, facial technology is a focal point in this current media landscape. Facial animation technology at top-end production-grade range has seen significant ad-

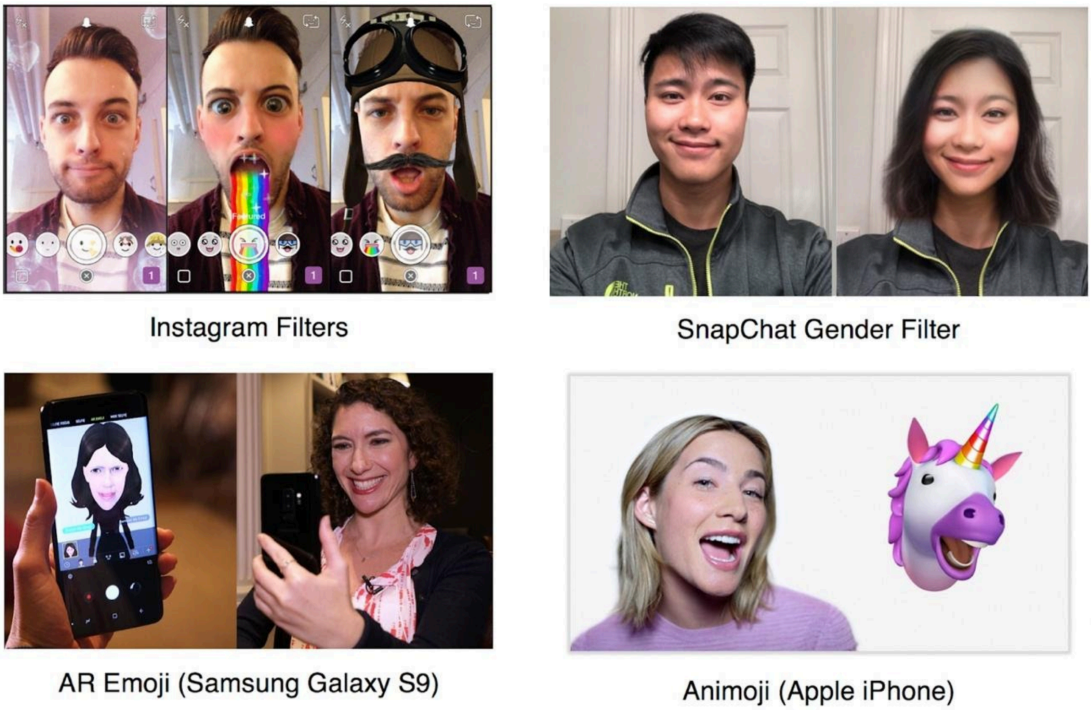


Figure 1.1: "Filtered Digital Self-Representation", term borrowed from Herring et al. [1]

vances. Most recently, acclaimed Hollywood director Martin Scorsese pushed the limits of computer-generated graphics (CGI) to augment the faces of his main cast of actors and make them decades younger [2]. Similarly, it is becoming more common to leverage CGI to create photo-realistic faces of deceased actors [3]. British company Cubic Motion develop technology some of the most sophisticated motion-capture based digital facial animation systems in the industry, closing the gap between the realism of human and digital actor performances [4].

Digital tools that are able to create realistic human-like visuals and animation are today largely still highly-specialised, require labour-intensive work, capturing actors with specialised hardware and generally mostly accessible to professionals and teams with significant budgets. In order to allow artists, designers, indie or amateurs creators to achieve comparable work and introduce synthetic, humanoid and personalised media, new tools and techniques are required. In a re-

cent review of Apple's "Animoji" face-driven avatar animation system, [1] coin the term Filtered Digital Self-Representation in which they try to capture a broader social phenomenon—people with limited technical skills are able to modify and extend their online self-presentation.

New machine learning based techniques have emerged over the last years to offer alternatives to high-end specialised software. In wider culture, some of them have become known as Deepfakes (deep stands for the use of Deep Learning, a subfield of Machine Learning which will be described in Chapter 3). Deepfakes refer to images synthesised with certain types of neural networks (typically Autoenconders or Generative Adversarial Networks, Figure 1.2 shows an example) that can learn to generate an input face and replace another in a target video. While the literature is quickly evolving, examples of creative uses are still somewhat rare but emerging [5]. Computer Vision startup Synthesia specialise in applying these advancements in technology to produce high-fidelity synthesised facial reenactment [6]. Early explorative work shows automated news [7] and weather [8] reports.

The majority of the aforementioned examples are centred around visual data—motion capture, face-to-face reenactment, or more sophisticated physical models targeted for high-end industry professionals. A related topic that has received less attention is using speech audio as the driving factor. The literature on sound-based body and face animation is comparatively less extensive however it can provide many benefits both from a theoretical understanding, as human speech and the corresponding motion are directly connected, as well from a practical standpoint in scenarios where audio is the only data source or it is preferable as the controlling interface.

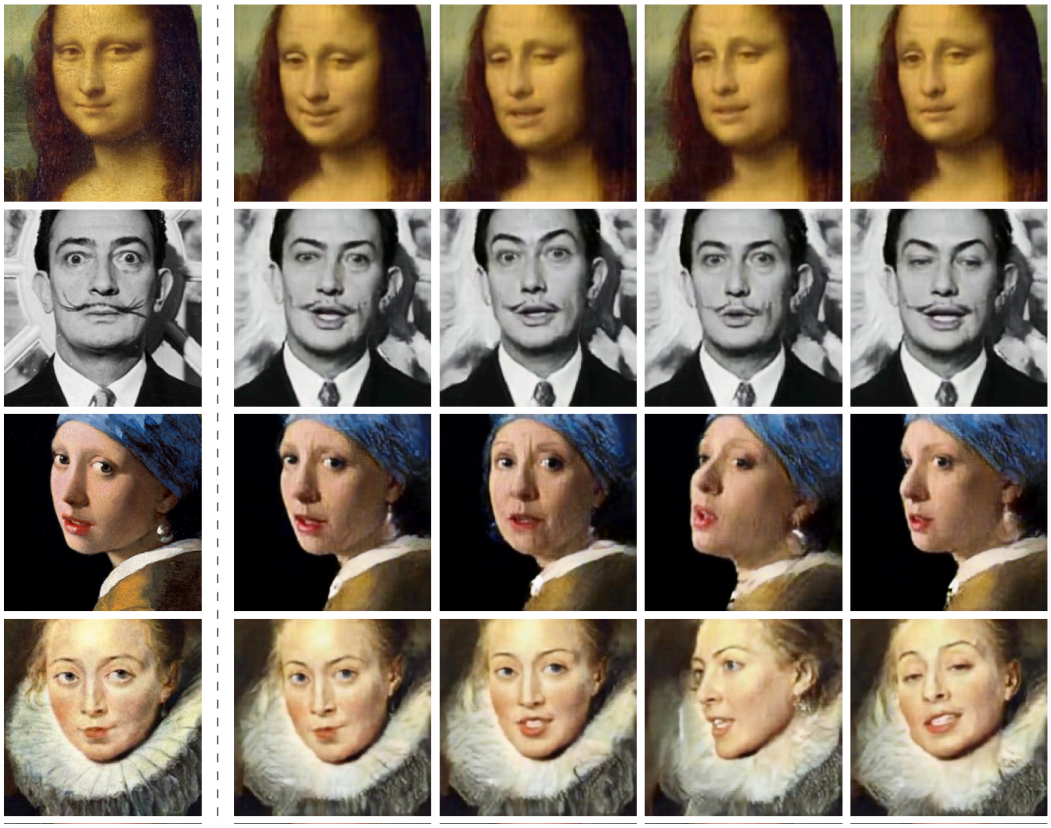


Figure 1.2: "Deep fakes" example — driving face images with video using Deep Learning [9]

1.2 Motivation

Improving our understanding of the relationship between body and facial motion, and the corresponding speech signal will provide valuable insight for the analysis of non-verbal communication. Further practical motivation is provided by the fact that human speech is generally readily available, easily captured in a digital format, requires less bandwidth compared to visual data, and arguably harder to degrade by environmental conditions. Indeed, driving animation from audio yields a solution to problems of missing, ambiguous or noisy visual data (occlusion of actor, bad lighting, etc). Further, it opens many creative possibilities such as creating on-demand animation content for digital personal assistants,

digital help desks, games, education material and so on.

Assuming the speech signal contains features that map to human body motion and pose, presented with sufficient training data and an appropriate machine learning model, a mapping can be *learned*. Once a data-driven approach is chosen, analysing the possible source datasets is itself an interesting problem. While vast amounts of raw footage is generally freely available on the internet, it is a considerably difficult task to curate and pre-process it in a way that it becomes useful for a learning task. Further, movement, gestures and speaking styles vary significantly across individuals and cultures, which makes it very hard to create generically useful models. In that sense, there will be need for tools and techniques for end-users, artists and researchers to be able to extend or create their own purpose-specific datasets, by recording themselves or collecting and processing "in the wild" footage.

1.3 Aims

An overview was given as to why driving human body and facial movements with an audio speech signal is a worthwhile research topic. This project explores the state-of-the-art data-driven audio-based methods to generate human body animation and pose, and implements an extension to a specific Deep Learning architecture to learn to control human head pose from audio. Moreover, the creation of a dataset by scraping online footage and processing it with an off-the-shelf facial analysis tool is investigated.

Chapter 2 will review the current state-of-the-art in the research literature and a starting point for the implementation of the project. Chapter 3 will describe

the theoretical prerequisites in more detail. Chapter 4 lays out the implementation of the proposed system. Chapter 5 reviews the different experiments that were run and evaluates the results qualitatively and quantitatively. Lastly, Chapter 6 forms the conclusion and describes several directions for future work.

Chapter 2

Related Work

2.1 Overview

This project takes a broader look at a few different strands in the research literature unified by the common theme of generating human body pose and motion from audio. This limits the depth of the review to current state-of-the-art papers, especially those based on data-driven Deep Learning methods.

The human body can broken down and categorised into gestures (i.e. the motion of the limbs while talking), head pose and lip motion. Each of this is its own research topic and this categorisation clearly simplifies many aspects of the visual nature of human speech (e.g. gaze direction, brow movement and so forth). Chapter 6 expands on this and proposes ideas and directions for future work. Some methods however do not explicitly differentiate semantically between parts of the human body (particularly those based on Generative Adversarial Networks (GANs), which will be reviewed below).

Another approach to analyse the literature is to consider the representation of the data and the output. Again, three common categories can be traced down—treating the data simply as 3D vertices, using coefficients for parametric models, or using GANs to create models that learn to draw 2D images of human bodies directly. It can be useful to think about those two categorisation principles as the axes of a 2D matrix as most papers reviewed here can be placed inside.

Table 2.1: Audio-driven human motion literature categorisation.

	Gestures	Head Pose	Lip Motion
3D vertices	[10], [11], [12], [13]	[11]	[14]
Parametric Model		[15]	[16], [17]
GAN		[15], [18], [9], [19]	[18], [9]

Lastly, the papers that specifically form the basis of the implementation of this project (Chapter 4) are looked into more detail at the end.

2.2 Rule-based systems vs Deep Learning

Classically, human pose and motion was a problem of interest mainly in the fields of Computer Graphics and Animation, and Robotics ([20], [21], [22]). Animating the human body is a painstakingly detailed process that requires a considerable amount of manual labour to achieve human-like realism. Rule-based systems similarly are very costly to create and are creatively bounded.

A clear alternative to escape the rigid nature of rules or laborious process of manual animation is to leverage the fact that human body motion examples exist in vast amounts and new ones can be captured by using MoCap technologies or generated using computer vision by extracting data from "in the wild" footage. In order to make those data useful, recent advancements in the field of Machine Learning are considered.

2.3 Deep Learning

A relatively recent contribution to the field of Computer Graphics is the newly rejuvenated field of Machine Learning, specifically the family of neural network architectures referred to as deep neural networks, or more commonly Deep Learning. Conceptually, neural networks span back to the origins of modern computers [23] [24]. At a glance, neural networks are biologically-inspired directed, weighted graphs. They can learn by adjusting their weights based on observational data.

The big spike in interest from the research community originated in the early 2010s when several breakthroughs occurred in the space of image recognition. In the 2012 the work Alex Krizchevsky [25], a Convolutional Neural Network (CNN) called AlexNet, achieved an error 15.3% in the object recognition challenge, more than 10.8 percentage lower than the runner up. CNNs are a type of neural network architecture that uses regularisation based on the typical patterns of image and video data, a simple example is shown in Figure 2.1. Krizchevsky's doctoral advisor was Geoffrey Hinton, who together with Yoshua Bengio and Yann LeCun would win the Turing Award in 2019 for their work on Deep Learning.

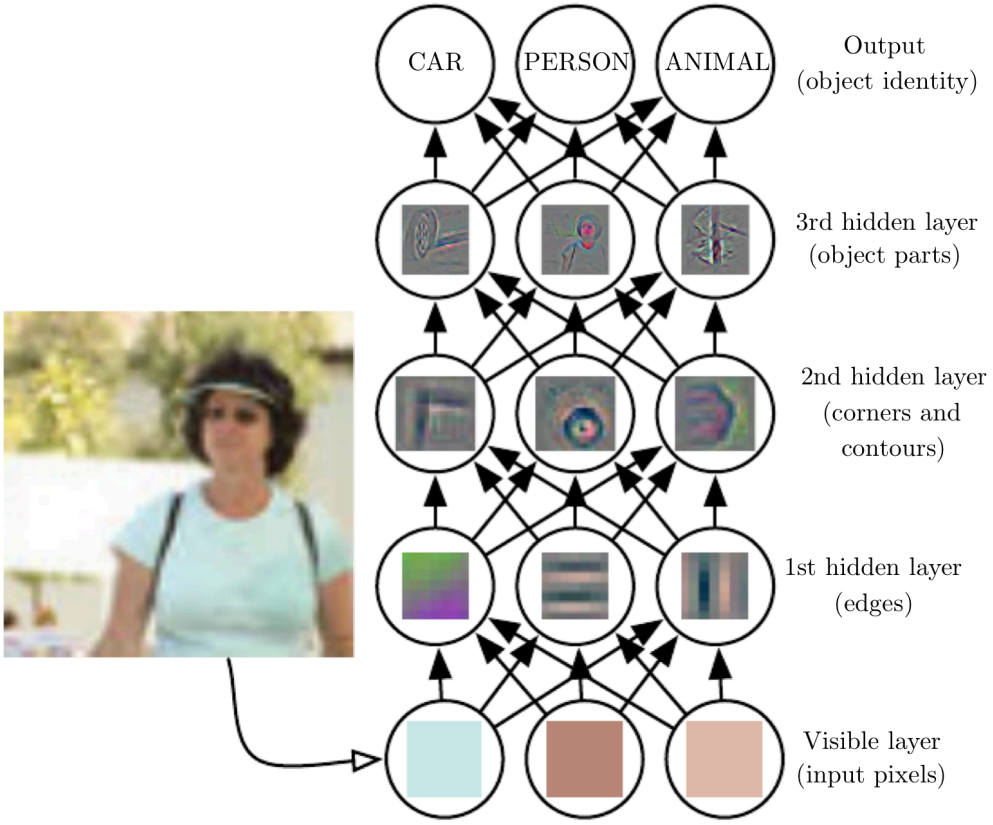


Figure 2.1: Deep Learning Image Classifier [26]

Since 2012 there has been an influx of interest, from both research and industry, and it would be impossible to touch upon all important aspects of the field. Still, several core components can be attributed to its success. Internally, the backbone of virtually all neural network architectures is the class of algorithms called backpropagation. Backpropagation computes the loss function of the network, i.e. how far off the output is from the desired target. Weights in the network are then adjusted based on the mathematical gradient.

Externally, the advent of modern graphical processing units (GPUs) turned out to be a particularly excellent fit for Deep Learning due to the common massively parallel nature of modern computer graphics and neural nets—they both

rely on a substantial amount of simple linear algebra operations. In real-time graphics, this usually means computing matrix operations that yield vertex coordinates, while neural nets can be seen to a large extend as matrix multiplication graphs. Further, the amount of raw data collected and available today has played a crucial role in the success of Deep Learning. The most impressive models largely depend on the size and quality of the datasets.

To tie things back to 3D graphics and human body animations, one can conceptually view Deep Learning as an universal non-linear function approximator. These non-linear models can replace, outperform and generalise better than classical statistical methods. This project specifically looks at modelling human body animations using speech as a non-linear learning task that can leverage a dataset made up of animated 3D models and speech data pairs.

2.3.1 GANs

One of the main contributions of Deep Learning to the research of synthetic imagery is the idea of Generative Adversarial Networks (GAN). As initially proposed by Goodfellow et al. [27] in 2014, the technique is based on two contesting neural networks—a generative and a discriminative network—in a game theoretic setting. The role of the former is try and generate fake samples from a dataset, while the latter tries to correctly detect counterfeits from real samples. As both networks are differentiable (i.e. can be trained using backpropagation), the generator gets better at producing synthetic data while the discriminator gets better at catching fakes.

Since their initial emergence GANs have found number of uses as genera-

tive models—synthesising voices, music and text. However, they have primarily received attention for their effectiveness in producing synthetic imagery. Image generation as a sub-problem in itself has generated plenty of interest from the research community and some of the techniques have found themselves in commercial applications or captured the attention of the media. Most notably, Style Transfer (originally proposed by Gatys et al. [28])—a technique to apply the artistic style of image to another—and various work in the family of photo-realistic human face generation. GANs have also been used to create Deepfakes.

This review will further narrow down on the state-of-the-art literature related to synthetic face generation and animation using GANs. Static face generation has seen steady improvements in fidelity and the state-of-the-art [29] has achieved realism to the extend that synthesised artefacts are nearly undetectable to the human eye [30].

Further GAN extensions have been made to translate input imagery to an output target image. For example the work of Isola et al. [31] on conditional adversarial networks has provided for a practical way to influence (or condition) the output of a GAN opening possibilities for puppetry in the context of face generation. A conditional GAN, or cGAN, as first proposed by Mirza et al. [32] shows that both the generator and discriminator can be conditioned on extra information, such as a label existing in the dataset. In the original image-to-image formulation, the dataset of input and output domain needs to be paired. This limits a lot of the creative uses as large enough paired visual datasets are hard to find or construct.

Zhu et al. [33], propose the CycleGan architecture to allow for unpaired datasets. Since an unpaired domain mapping is heavily under-constrained, Cy-

cleGan trains to separate generator-discriminator pairs, where the image is being transformed to the target domain and back. The loss is then calculated by how much the input deviates after the two transformations (i.e. how much it deformed after a full round trip).

2.4 3D Morphable Models

One way to tackle the seemingly infinite complexity of a human face is approaching it from a statistical perspective and assuming that some facial configurations are more likely than others, and some completely unlikely. If a model is able to constrain facial deformations based on statistical likelihood, correspondence and animation become tractable problems. In a seminal paper Blanz and Vetter [34] proposed such a method–now known as a 3D Morphable Face Models (3DMM).

More broadly, 3DMMs exist in a family of facial expression linear models that are commonly referred to as blendshape models. This project focuses on the statistical nature of 3DMM but other methods exists, such as parametric models, models driven by motion capture and others. In Practice and Theory of Blendshape Facial Models [35], Lewis et al. review the history and common variations.

By preparing a dataset of 3D faces of similar topology and with full correspondence, this method constructs a generative model for face shape and appearance. Linear combinations of these models produce morphologically realistic faces (called morphs). 3DMMs can separate and distinguish the shape, appearance and expression of a face as distinct models.

In the original work, Principal Component Analysis (PCA) is used to derive a statistical shape model. PCA performs a basis transformation to a new coordinate system which axes are ordered by the variation in the data. Since then, the literature has proposed other learning techniques. Shape models can be further separated in global and local models. Local models improves modelling of important localised areas that require more detailed control, thus yielding in higher fidelity results.

Expression models concern themselves with capturing the expression variation of a subject, where expression is distinct from the identity. When decoupled and modelled separately, expressions can be transferred between subjects and interpreted as blendshape coefficients. This plays an important role in facial reenactment as described later. Both facial shape and expression models have been traditionally linear but later work, such as FLAME [36], has proposed non-linear control.

Lastly, appearance models which deal with the albedo and illumination data of a face are described. Similarly to shape and expression models, appearance models can both linear and non-linear and can be derived statistically. As is common in Computer Graphics, appearance information is represented as per-vertex or in a UV texture.

3D Morphable Models is active area of research and this project will aim to explore state-of-the-art solutions. There are many aspects that are out of scope for this project, such as data capture, and won't be addressed here. Please refer to 3D Morphable Face Models - Past, Present and Future [37] for a more thorough review.

2.5 Audio to human body motion

With an overview of the common methodology, several papers considered state-of-the-art as of this writing will be reviewed.

The research literature offers several directions in the problem of mapping audio to human body motion. One sub-problem is generating realistic lip motion. Cudeiro et al. [VOCA2019] propose VOCA, a Deep Learning-based pipeline (Figure 2.2) that, firstly uses a Deep Learning model to extract speech features, then another network that maps those features to the coefficients of a 3DMM.

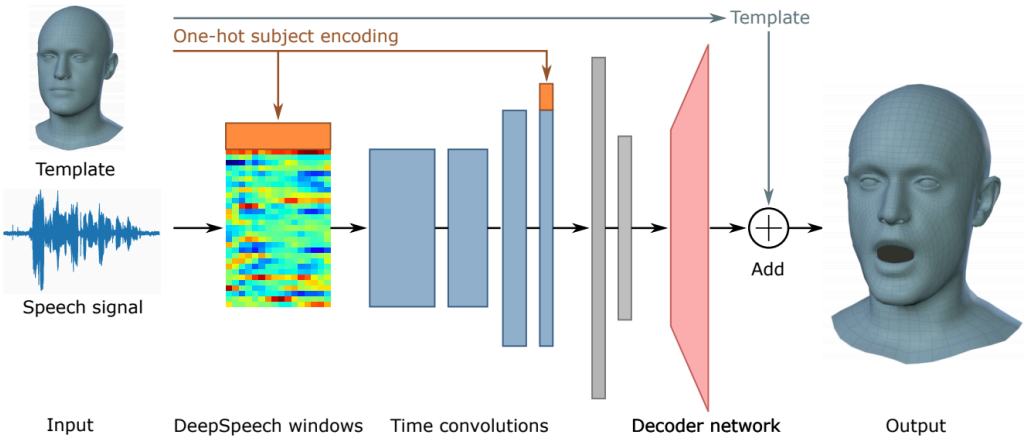


Figure 2: VOCA network architecture.

Figure 2.2: VOCA pipeline [16]

While most speech-to-motion methods use more traditional speech processing techniques like MFCCs, VOCA leverages a pre-trained model in the face of DeepSpeech [38]. DeepSpeech is originally a Recurrent Neural Network-based architecture, however in practice the Long-Short Term Memory (LSTM) neural network architecture is commonly used. It learns speech features from audio data, in this instance using 26 MFCC features instead of the raw spectrogram. The au-

thors claim this allows their model to generalise well over different audio sources (i.e. different persons), as well as being robust to noise, recording artefacts, different languages and accents.

The VOCA models does not work on raw 3D vertices of the lips, it represents the face with 3DMM, and in this instance the FLAME [36] model is the chosen 3DMM implementation.

In a recent paper Yi et al. [15] propose a DNN pipeline that directly produces a talking face video from an audio speech signal of an arbitrary source person.

Their pipeline can be grouped in two stages (see Figure 2.3). In the first stage, a mapping from audio speech to facial expression and head pose is learned. Then, an audio input provided, a 3D face can be reconstructed and the mapping fine tuned to learn a personalised talking behaviour from the input video. In the second stage the 3D face is animated and rendered using the texture and lighting information generated from the input video. The graphics engine can now render frames using his information, which acts merely as a mediator for the final rendering step. Indeed, to achieve photo-realistic results, a memory-augmented GAN is trained to map the 3D faces to refined synthesised video frames of a personalised talking face.

Karras et al. [14] show that different neural network types are can also learn to map audio data to animations, though their work is more involved in terms of processing audio. Filtering and gain normalisation techniques are employed to achieve consistent results.

Alternatively, Taylor et al. [39] show a direct audio approach can be avoided

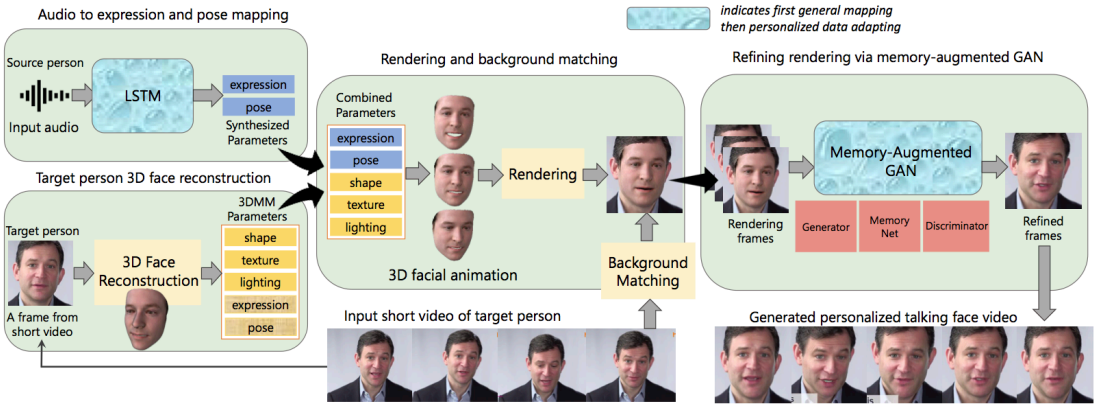


Figure 2.3: Yi et al. pipeline [15]

by transcribing speech to phonemes and training on the phoneme labels. Audio and video data from training footage are paired together. The mouth region is tracked and parametrised using shape models which results in a label for a given frame. The corresponding audio pair is the transcribed phoneme.

Suwajanakorn et al. [19] propose another RNN-based approach in which mouth textures are generated from audio later to be attached to a face in the final composite. The main paper this project will focus on is Thies et al. [17]. Similarly, a DeepSpeech architecture has been used to learn facial expression embeddings.

Lastly, three recent papers ([13], [12], [10]) proposing Deep Learning-based solutions to predict human gestures from audio speech will be reviewed. Specifically, the latest work by Kucherenko et al. [10] forms the basis of implementation of this project which will be described in Chapter 4.

In [13] a direct pipeline that maps speech audio to human gestures is proposed (see Figure 2.4). The audio is extracted into Mel-Frequency Cepstral Coefficients (MFCC) which provide for simpler representation of the signal motivated by the human auditory system, making the input more conducive to learning the speech-to-gesture mapping. The neural architecture is based on the Bi-Directional

l Long Short-Term Memory (LSMT), a particular extension to the Recurrent Neural Network (RNN) that is able to learn sequences with difficult relationships through time (i.e. requiring long-term memory). The gestures are represented as 3D joint positions of the entire body for any point in time, meaning the network the output of the network is also 3D position predictions. The authors use a data set consisting of speech and gesture pairs from an interview-style setting. To achieve further realism, the temporal discontinuities of the output are filtered with a smoothing function.

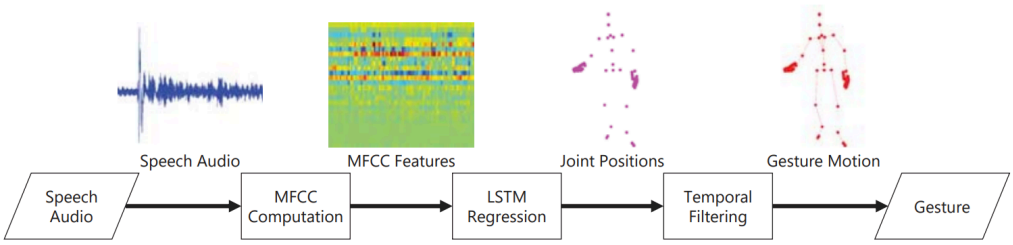


Figure 2.4: Hasegawa et al. pipeline [13]

Kucherenko et al. [10] propose a meaningful extension this architecture by including an additional step involving representation learning for the motion data (Figure 2.5). The proposed strategy is using a Denoising Autoencoder (DAE). That way, a lower dimensional representation of the input data (3D positions of joints) can be learned by decoding it into a latent space and then decoding it by trying to reconstruct the original input as close as possible. The DAE is forced to learn a compact embedding of the dataset. This embedding can be leveraged in the speech-to-gesture mapping. As the network learns to map speech to the latent space of the motion, the final step is to deploy the decoder and output motion in the original space (i.e. 3D positions of all joints).

Kucherenko et al. continue their work in a separate paper [10]. A new

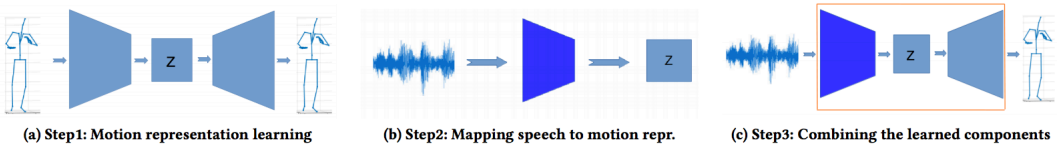


Figure 2.5: Kucherentko et al. pipeline [12]

neural architecture is proposed, using the Gated Recurrent Unit (GRU), a slightly simpler sibling of the LSTM. The authors explored some alternatives to MFCCs, by training on raw spectrograms and prosodic features separately. Prosodic feature entail specific signifying qualities that exist in human speech. In this instance, the energy of the speech and the time derivative of the energy signal, as well as the logarithm of the F0 pitch contour and its time derivative. The pitch contains information about the speech intonation.

In summary, the literature reviewed in this chapter tends to have a few themes in common. Human motion consists of a very intricate parameter space, whatever representation is chosen. To avoid laborious manual methods of generating realistic animations or rigid and constrained rule-based methods, a data-driven approach is desired. However, a naive approach will not be able to model the complexity of the data. To this end, deep neural networks are employed to learn a representation of recorded human motion and map it to a corresponding input signal, in this case audio speech.

This should give sufficient motivation for the choice of methodology. The following Chapter will provide a more detailed theoretical overview of the commonly used techniques in the reviewed papers.

Chapter 3

Preliminaries

This chapter will introduce all underlying concepts before describing the implementation in Chapter 4.

3.1 Head Pose

This project involves finding test subjects "in the wild" and generating a dataset of head pose and speech audio pairs using video footage. To that end, an off-the-shelf Facial Behaviour Analysis toolkit (specifically OpenFace 2.0 [40]) was used. OpenFace provides an automatic modern face analysis pipeline (see Figure 3.1) consisting of facial landmark location, head pose, eye gaze and facial expressions. This project focuses on the head pose component which compared to other systems is generated at a reasonable quality and uses less computationally demanding. See [40] for a full comparison.

OpenFace extracts the head pose as two components, translation and ori-

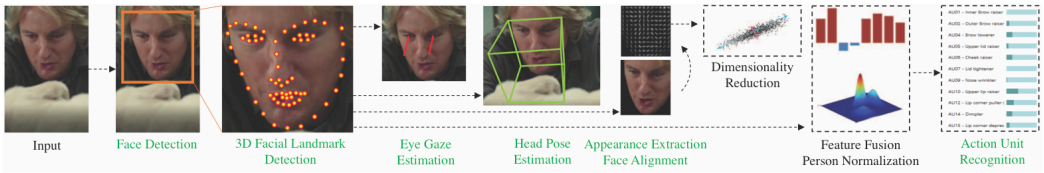


Figure 3.1: OpenFace 2.0 analysis pipeline [40]}

entation, from the 3D representation of the facial landmarks. Those are being projected using an orthographic camera model which allows for an accurate estimate of the head pose by solving an n point in perspective problem. In Figure 3.2, an example can be seen of the head pose estimate given by OpenFace, on the right superimposed on top of the original footage, and on the left using the head pose information to translate and rotate a 3D mesh.

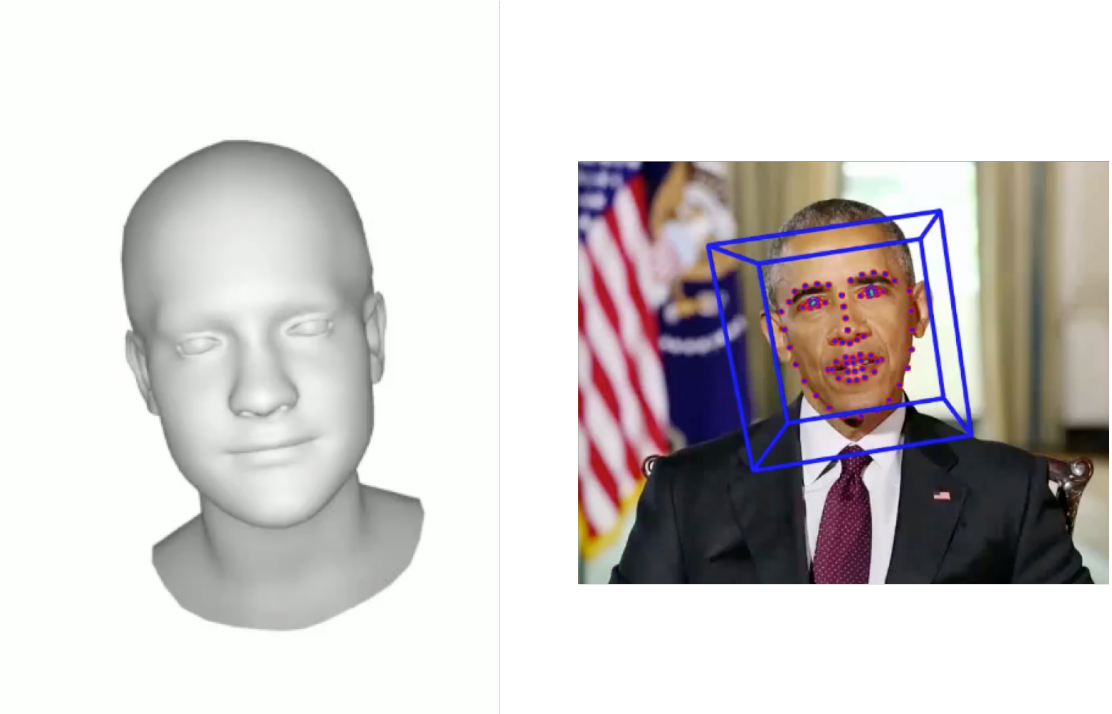


Figure 3.2: Head pose on a custom dataset

3.2 Deep Neural Networks

Chapter 1 has given some motivation and context as to why Deep Neural Networks (DNN) are appropriate to tackle the problem of modelling human motion data. Chapter 2 reviewed several key papers that has shown successful results in deploying DNNs and related techniques. The section will introduce the concepts related to DNNs for the proposed method in the next chapter.

DNNs extend Feed-Forward Neural Networks, also called Multi-Layer Perceptrons (MLPs) introduced by Rosenblatt [24]. A neural network, while referring to the human brain in name and being originally inspired by neurons, is simply a function f that maps an input x to an output y and learns specific parameters, most commonly a bias b parameter and a set of weights $\mathbf{b}$. Importantly, neural networks usually incorporate some kind of non-linear activation function, say the sigmoid function σ, that allow the function to model non-linear manifolds.

$$f(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b)) \tag{3.1}$$

Both Deep Learning, DNNs and related nomenclature refer to the concept of neural networks with many hidden layers and all accompanying mathematical and computational techniques that allow for deep architectures to learn parameters over big datasets.

3.2.1 Autoencoder

Encoder-decoder architectures, loosely, are DNN architectures in which an encoder function maps an input into a latent space, and then a decoder func-

tion maps it once again to a desired output space. Autoencoders are a type of encoder-decoder network that aims to reconstruct the input as close as possible given a metric. Usually, the information passes through a lower dimensions space bottleneck before getting reconstructed back. In a sense, Autoencoders perform lossy compression, where the neural network tries to learn features in the data that allow for the most optimal compression. Another way to think about, Autoencoders perform a similar role to Principal Component Analysis (PCA), where the Autoencoder learns to fit a non-linear manifold compared to the linear fitting PCA performs. A simple example of this is show in Figure 3.3.

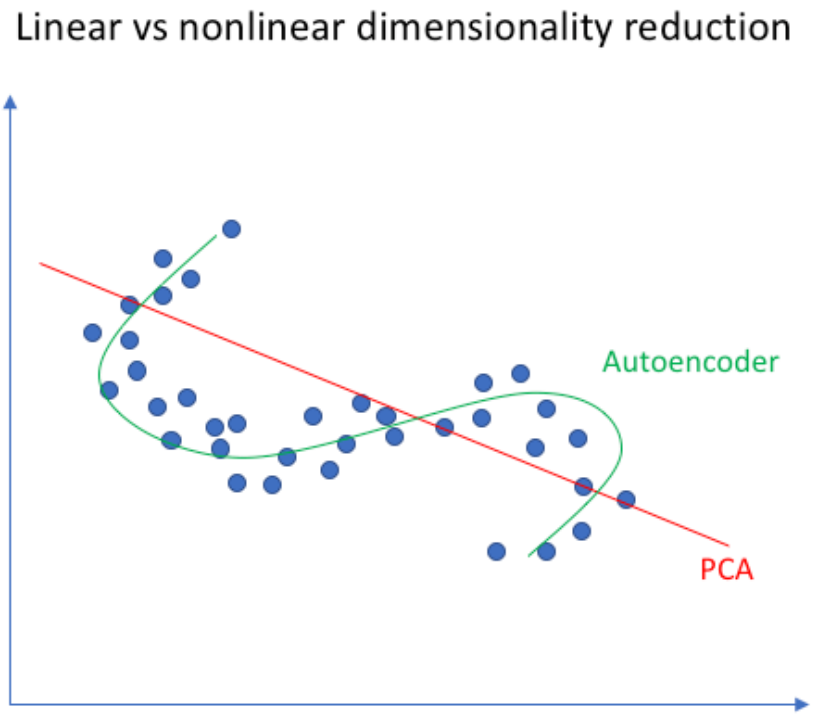


Figure 3.3: Figure from [41]

While Autoencoders do not outperform traditional compression algorithms, they tend to have interesting applications as applied to noise reduction and dimensionality reduction, the latter being mostly used for data visualisation purposes of highly dimensional data. From the perspective of the problem of human motion, some Autoencoder architectures present interesting qualities of learning lower-dimension representation of highly-dimensional data, usually referred to as Representation Learning (for example [12]).

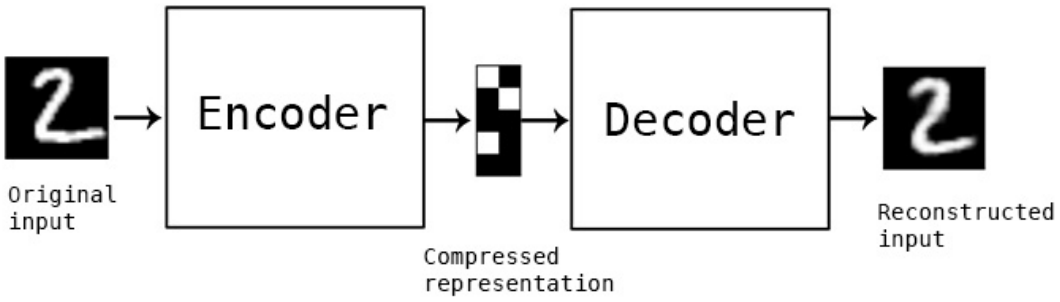


Figure 3.4: Simplified Autoencoder diagram [42]

Vincent et al. [43] propose an architecture (Figure 3.5) named a Denoising Autoencoder (DAE) that achieves a good representation by defining a denoising training and reconstruction criterion and corrupting the input data. While the goal is not denoising specifically, the task of denoising tends to force the Autoencoder to learn a useful structure in the distribution of the data. Chapter 4 will show that incidentally, this offers a double benefit as compared to high-fidelity studio-captured datasets, "in the wild" tends to be noisy anyway. Even though artificially corrupting the input data is generally done when using DAEs, the dataset for this project benefits from the denoising quality at the same time a representation is learned.

As per [43], to train a DAE the initial input $\mathbf{x}$ is corrupted into $\bar{\mathbf{x}}$ by means of stochastic mapping

$$\bar{\mathbf{x}} \approx q_D(\bar{\mathbf{x}}|\mathbf{x}) \tag{3.2}$$

Next, the corrupted input $\bar{\mathbf{x}}$ is mapped to a hidden representation with the same process as a traditional Autoencoder

$$\mathbf{h} = f_\theta(\bar{\mathbf{x}} + \mathbf{b}) \tag{3.3}$$

Finally, the model reconstructs the input data from the hidden representation

$$\mathbf{z} = g_{\theta'}(\mathbf{h}) \tag{3.4}$$

θ and θ' are parameters of the models that are learned by minimising the average reconstruction error over the training data. The corruption strategy can be of any kind, though for motion data, Gaussian noise is generally used.

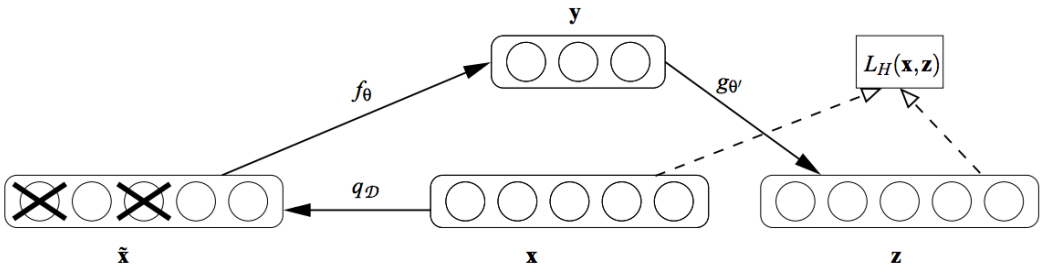


Figure 3.5: Denoising Autoencoder [43]. $\mathbf{x}$ is an example that is stochastically corrupted via q_D into $\hat{\mathbf{x}}$, further mapped to $\mathbf{y}$ via the encoder f_θ and finally reconstructed back via the decoder $g_{\theta'}$ where $L_h(\mathbf{x}, \mathbf{z})$ is the loss function.

3.2.2 Learning Sequences

While Representation Learning is helpful to provide for a compact embedding of human motion data, this project is concerned with trying to generate realistic predictions from speech signals. This means that a neural network architecture needs to be chosen that lends itself well to learning and predicting sequential data. Human motions tend to have a specific range of motion and can move in a limited range of velocity and acceleration, so it is important for the network to be able to learn these properties well.

The family of neural networks that has shown to be able to successfully learn sequences is called Recurrent Neural Networks, or RNNs for short. RNNs expand on MLPs by introducing loops in the architecture and allowing for information persistence.

Consider a sequence of values $\mathbf{x}_1, ..., \mathbf{x}_n$, and time step t. For each cycle the network looks at $\mathbf{x}_t$ and outputs the result into a so-called hidden state $\mathbf{h}_t$.

Elman [44], considered one of the simplest RNN architectures, defines the hidden state $\mathbf{h}_t$ and output $\mathbf{y}_t$ as such

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{3.5}$$

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_h) \tag{3.6}$$

with $\mathbf{W}$ and $\mathbf{U}$ being weight matrices.

This allows for unrolling the graph (Figure 3.6) for any number of time steps, allowing learning of arbitrary length of sequences and producing sequential

predictions that the network has not seen. In practice however, classical RNNs are very hard to train due to their susceptibility to a vanishing gradient [45]—a common numerical problem in very deep networks in which the calculated gradient in a given training step is too small due (i.e. vanishes) to the multiplication of too many small weight matrices. When vanishing gradients appear, the network's weights do not change either and learning fails to occur.

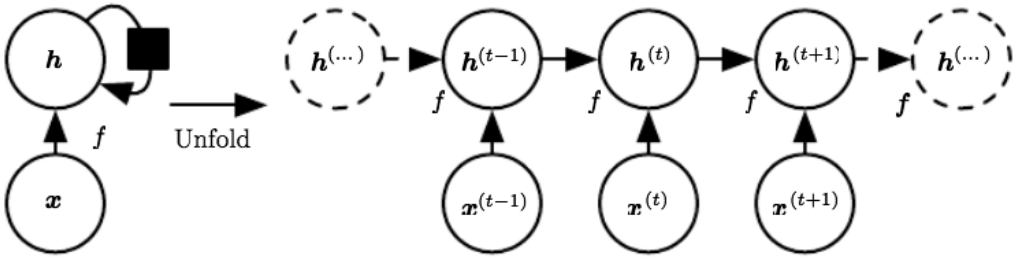


Figure 3.6: RNN unrolled [26]

A successor to RNNs, the Long Short Term Memory (LSTM), first proposed by Hochreiter and Schmidhuber [46], address this problems by extending the architecture with a Cell C that aims to model the relationship of the current time step with the past. The Cell in turn is controlled by three gates

- f_t - usually called the forget gate. The network uses it to decide which information should be discarded (weighted from 0 to 1) from the cell state. The decision is based on the previous hidden state and the input.
- i_t - usually called the input gate. Decides which values will be updates by looking at the previous hidden state and the input.
- o_t - this gate is the last step, filters out the output based on the already calculated current cell state

Lastly, $\hat{c}$ consists of the new values the network is going to add to the cell

state. Combined with the input gate which decides how much of those will be used when constructing c_t.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{3.7}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{3.8}$$

$$\hat{\mathbf{C}}_t = tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{3.9}$$

$$\mathbf{C}_t = \mathbf{i}_t \times \hat{\mathbf{C}} + \mathbf{f}_t + \mathbf{C}_{t-1} \tag{3.10}$$

$$\mathbf{0}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{3.11}$$

$$\mathbf{h}_t = o_t \times tanh(C_t) \tag{3.12}$$

Another variation of the LSTM architecture is the Gated Recurrent Unit (GRU) proposed by Cho et al [47] which similarly uses a gating mechanism. GRUs are simpler than LSTMs since they have fewer parameters, however in some scenarios show comparable performance.

Canonically, GRUs can be defined as

$$\mathbf{z}_t = \sigma_g(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{3.13}$$

$$\mathbf{r}_t = \sigma_g(\mathbf{W}_r \mathbf{x}_w + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_k) \tag{3.14}$$

$$\hat{\mathbf{h}}_t = \phi_h(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_k) \tag{3.15}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \hat{\mathbf{h}}_t \tag{3.16}$$

where

- $\mathbf{x}_t$ - input vector
- $\mathbf{h}_t$ - output vector
- $\hat{\mathbf{h}}_t$ - candidate activation vector
- $\mathbf{z}_t$ - update gate vector
- $\mathbf{r}_t$ - reset gate vector
- $\mathbf{W}, \mathbf{U}, \mathbf{b}$ - parameters

and the activations functions

- σ_g - sigmoid
- ϕ_h - hyperbolic tangent

though other variations exist.

3.3 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral (MFC) is a representation of the short-term power spectrum of the sound that is widely used in the fields of sound processing, automatic speech recognition, music information retrieval and so on. Mel Frequency Cepstral Coefficients (MFCC) are the coefficients that the MFC is made of. Introduced by Davis and Mermelstein [48], they superseded Linear Prediction Coefficients (LPC) and Linear Prediction Cepstral Coefficients (LPCC) as the state-of-the-art technique for representing speech features. Only recently have Deep Learning-based techniques made progress in presenting compelling alternatives [38]. All of these methods are motivated by the human auditory and speech systems.

Specifically for MFCCs, the periodogram (power spectrum) that is calculated for each frame by taking the absolute Discreet Fourier Transform (DFT) is based on frequency analyses done on the human cochlea.

At a glance, the algorithm to extract MFCCs from an audio signal consists of the following steps

- frame the signal into frames, usually 25ms
- for each frame, the magnitude of the DFT is calculated
- the mel filter bank is convolved with the periodogram
- the logarithm of the filter bank energies is taken
- apply the Discreet Cosine Transform (DCT) to the resulting logarithm
- keep a number of coefficients, discard the rest

The parameters used for calculating the MFCCs are described in Chapter 5.

Chapter 4

Audio-To-Headpose Architecture

4.1 Overview

This chapter will provide a detailed technical overview of the proposed method and its implementation. The pipeline and architecture of the network is largely an extension of [10] with the exception of the initial step of collecting data in the wild and constructing a dataset for head pose.

The DAE, which will be referred to as the HeadposeEncoder and HeadposeDecoder networks for its individual components, and the GRU-based architecture, which will be referred to as the SpeechToDecoded network, were implemented using TensorFlow [49]. TensorFlow is an open-source Deep Learning software package based on data-flow, differentiable programming, maintained by Google.

Figure 4.1 summarises the complete system, consisting of two main components, the training and the inference. The training itself consists of three different parts. Firstly, after dataset is processed and prepared, the DAE is trained on

the head pose data, aiming to reconstruct the data as close as possible using an embedded space as the bottleneck. In the second step the HeadposeEncoder component of the DAE is used to encode the complete dataset. Finally, in the third step the encoded head pose data is paired with the audio speech signal and fed to the SpeechToEncoded network, aiming to learn a relationship between the pairings. In the second phase, the pipeline is ready to be used for inference. To do that, the input audio signal is transformed into an MFCC vector, fed to the SpeechToEncoded, and lastly decoded using the HeadposeEncoder component of the DAE, which results in the desired head pose motion data.

4.2 Constructing a dataset for speech and head pose

The main difference between rule-based and machine learning systems is that the latter usually require a substantial amount of high-quality data. While it is common for most body motion research papers to record or use off-the-shelf production-grade studio captured datasets (i.e. using motion capture), estimating the head pose simplifies the problem somewhat as it can be represented by only two components, translation and rotation. Further, state-of-the-art face tracking systems can estimate head pose from video to a reasonably accurate degree, at least to the extend of using it for a learning task.

Given the vast amounts of video material online, a pipeline that automatically scrapes the Obama White House archive website and downloads his weekly presidential addresses[1] for a given date range was scripted. Once a raw video archive was collected, the videos were processed and cleaned by hand and

[1] https://obamawhitehouse.archives.gov/briefing-room/weekly-address

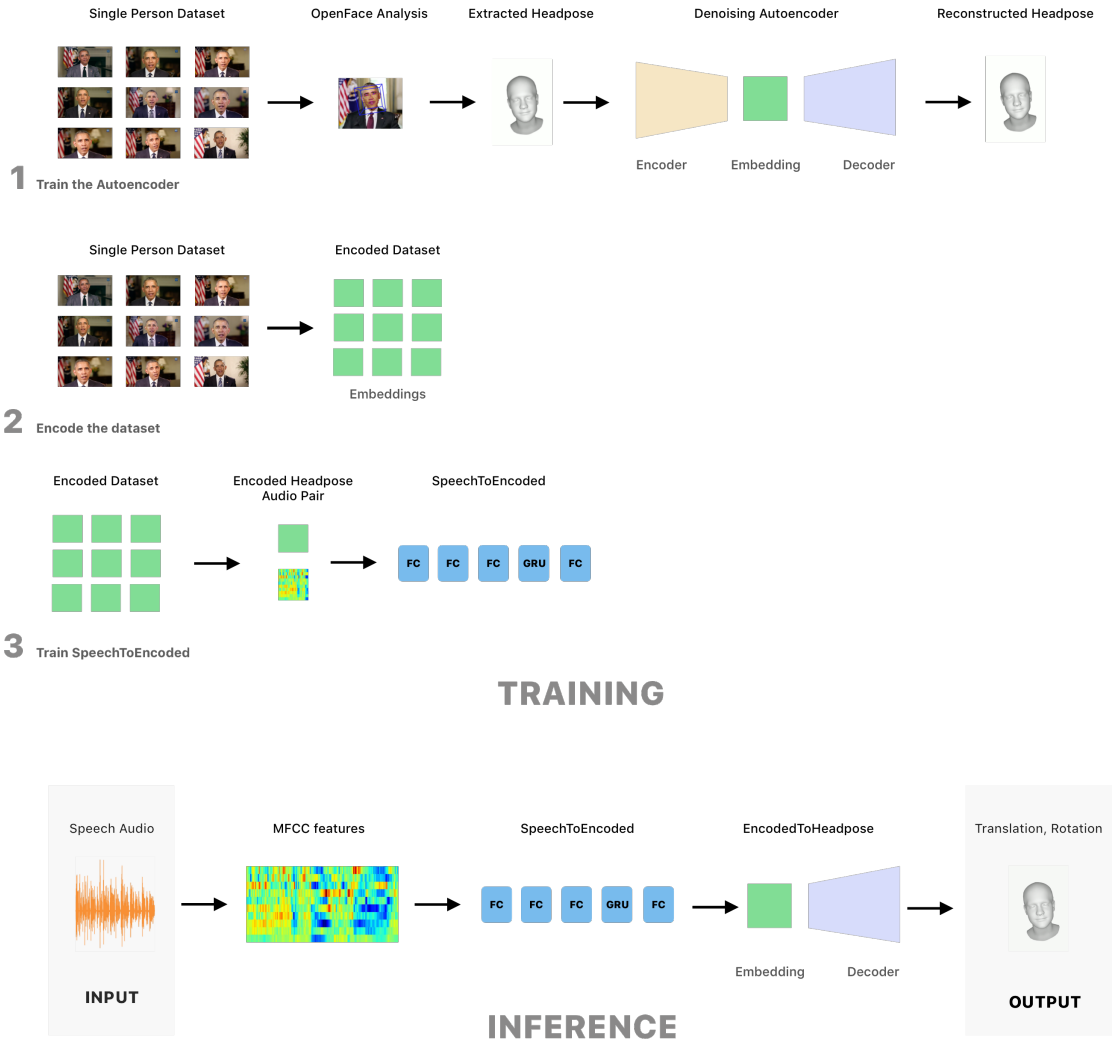


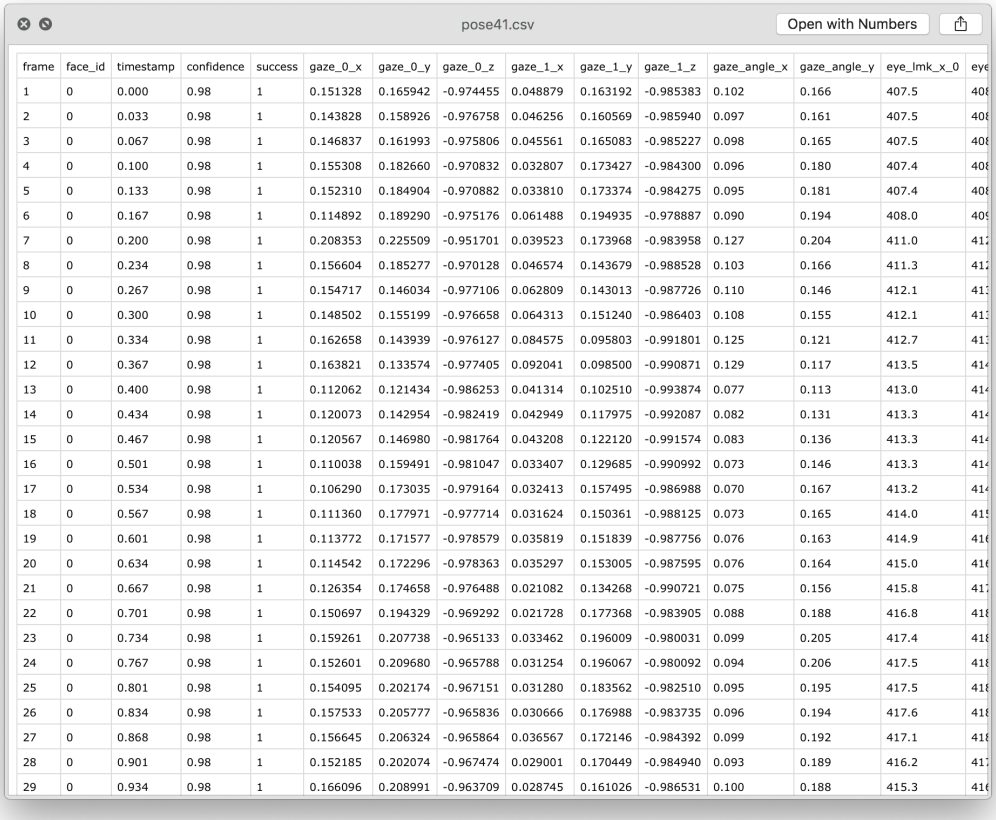
Figure 4.1: Audio-To-Headpose Architecture. On the top, the training process is visualised in three different steps and on the bottom the inference pipeline

timestamps of all sections in the video with a static camera angle and a visible front-facing talking head were extracted. Figure 4.2 shows some examples of the general content of the videos. These extracted video speech chunks could now be passed through OpenFace which generated the head pose translation and rotation components for each video frame. Lastly, by extracting the audio from the video snippets a complete audio-to-head pose dataset was constructed for one person (Obama). Due to different camera angles for each snippet, the mean translation for each example was calculated and subtracted in order to centre all the data.

Figure 4.2: Example snapshots from different examples in the dataset

The next step was to process the dataset with OpenFace. For ease of use, the files were transformed onto a Google Colab instance, which executed the necessary scripts. As a result, for each video a CSV text file was generated containing per-frame facial information, an example is shown in Figure 4.3.

To train a machine learning model, the data usually has to be organised into three separate groups—training, validation and testing. The split used for producing the final results contains overall 45 examples, 38 used for training with overall length 33 minutes and 44 seconds. For validation, 4 examples were used 6 minutes and 33 seconds in length. For validation, 3 examples were used 2 minutes and 51 seconds in length.



pose41.csv — Open with Numbers

frame	face_id	timestamp	confidence	success	gaze_0_x	gaze_0_y	gaze_0_z	gaze_1_x	gaze_1_y	gaze_1_z	gaze_angle_x	gaze_angle_y	eye_lmk_x_0	eye
1	0	0.000	0.98	1	0.151328	0.165942	-0.974455	0.048879	0.163192	-0.985383	0.102	0.166	407.5	408
2	0	0.033	0.98	1	0.143828	0.158926	-0.976758	0.046256	0.160569	-0.985940	0.097	0.161	407.5	408
3	0	0.067	0.98	1	0.146837	0.161993	-0.975806	0.045561	0.165083	-0.985227	0.098	0.165	407.5	408
4	0	0.100	0.98	1	0.155308	0.182660	-0.970832	0.032807	0.173427	-0.984300	0.096	0.180	407.4	408
5	0	0.133	0.98	1	0.152310	0.184904	-0.970882	0.033810	0.173374	-0.984275	0.095	0.181	407.4	408
6	0	0.167	0.98	1	0.114892	0.189290	-0.975176	0.061488	0.194935	-0.978887	0.090	0.194	408.0	409
7	0	0.200	0.98	1	0.208353	0.225509	-0.951701	0.039523	0.173968	-0.983958	0.127	0.204	411.0	412
8	0	0.234	0.98	1	0.156604	0.185277	-0.970128	0.046574	0.143679	-0.988528	0.103	0.166	411.3	412
9	0	0.267	0.98	1	0.154717	0.146034	-0.977106	0.062809	0.143013	-0.987726	0.110	0.146	412.1	413
10	0	0.300	0.98	1	0.148502	0.155199	-0.976658	0.064313	0.151240	-0.986403	0.108	0.155	412.1	413
11	0	0.334	0.98	1	0.162658	0.143939	-0.976127	0.084575	0.095803	-0.991801	0.125	0.121	412.7	413
12	0	0.367	0.98	1	0.163821	0.133574	-0.977405	0.092041	0.098500	-0.990871	0.129	0.117	413.5	414
13	0	0.400	0.98	1	0.112062	0.121434	-0.986253	0.041314	0.102510	-0.993874	0.077	0.113	413.0	414
14	0	0.434	0.98	1	0.120073	0.142954	-0.982419	0.042949	0.117975	-0.992087	0.082	0.131	413.3	414
15	0	0.467	0.98	1	0.120567	0.146980	-0.981764	0.043208	0.122120	-0.991574	0.083	0.136	413.3	414
16	0	0.501	0.98	1	0.110038	0.159491	-0.981047	0.033407	0.129685	-0.990992	0.073	0.146	413.3	414
17	0	0.534	0.98	1	0.106290	0.173035	-0.979164	0.032413	0.157495	-0.986988	0.070	0.167	413.2	414
18	0	0.567	0.98	1	0.111360	0.177971	-0.977714	0.031624	0.150361	-0.988125	0.073	0.165	414.0	415
19	0	0.601	0.98	1	0.113772	0.171577	-0.978579	0.035819	0.151839	-0.987756	0.076	0.163	414.9	416
20	0	0.634	0.98	1	0.114542	0.172296	-0.978363	0.035297	0.153005	-0.987595	0.076	0.164	415.0	416
21	0	0.667	0.98	1	0.126354	0.174658	-0.976488	0.021082	0.134268	-0.990721	0.075	0.156	415.8	417
22	0	0.701	0.98	1	0.150697	0.194329	-0.969292	0.021728	0.177368	-0.983905	0.088	0.188	416.8	418
23	0	0.734	0.98	1	0.159261	0.207738	-0.965133	0.033462	0.196009	-0.980031	0.099	0.205	417.4	418
24	0	0.767	0.98	1	0.152601	0.209680	-0.965788	0.031254	0.196067	-0.980092	0.094	0.206	417.5	418
25	0	0.801	0.98	1	0.154095	0.202174	-0.967151	0.031280	0.183562	-0.982510	0.095	0.195	417.5	418
26	0	0.834	0.98	1	0.157533	0.205777	-0.965836	0.030666	0.176988	-0.983735	0.096	0.194	417.6	418
27	0	0.868	0.98	1	0.156645	0.206324	-0.965864	0.036567	0.172146	-0.984392	0.099	0.192	417.1	418
28	0	0.901	0.98	1	0.152185	0.202074	-0.967474	0.029001	0.170449	-0.984940	0.093	0.189	416.2	417
29	0	0.934	0.98	1	0.166096	0.208991	-0.963709	0.028745	0.161026	-0.986531	0.100	0.188	415.3	416

Figure 4.3: Data from OpenFace

4.3 Pre-processing step

Now that the raw dataset is ready to use, a few pre-processing steps are required to prepare the data before feeding it to the Autoencoder and the Speech-ToDecoded network.

4.3.1 Audio Input data

The input audio is stored in a lossless format at 48000 Hz and 16 bits per sample. The motivation for using MFCCs was described in the previous chapter.

To calculate the MFCCs on the prepared dataset several steps are undertaken.

1. All input audio is transformed from stereo to mono by taking a simple average of the left and right channel.
2. The moving analysis window is set to 0.01 seconds which results in 100 frames per second.
3. We average every 5 frames to drop the FPS to 20.

4.3.2 Label data

The label data (i.e. the head pose) is stored originally as 30 fps which is what OpenFace produces. The data is read and downsampled to 20 FPS by dropping every third element.

Since the loss functions of the Autoencoder is based on distance, in order to capture the rotation information all all axes (i.e. pitch, yaw, roll), the rotation is encoded into three unit vectors. The unit vectors are based on OpenCVs coordinate system since this is what OpenFace uses. Pitch, yaw and roll respectively represent the rotation in radians around the x, y and z axes.

If a matrix of unit vectors $\mathbf{x}, \mathbf{y}$ and $\mathbf{x}$ is constructed for

$$\mathbf{x} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \mathbf{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.1}$$

then we have

$$\hat{\mathbf{M}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

The rotation vectors from OpenFace thus can be constructed into a rotation matrix. The rotation matrices around each axis in 3D is defined as such:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{4.3}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{4.4}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Multiplying the above three will result in a rotation matrix for all axes.

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) =$$

$$\begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{4.6}$$

Finally, the unit vector matrix can be multiplied by R and yield the rotation representation M that will be passed to the Autoencoder.

$$M_{center} = \hat{M_{center}} R \tag{4.7}$$

After running experiments, it was necessary to add the translation component for each frame to the respective rotation to improve prediction results. Thus the final form becomes

$$M = M_{center} + T \tag{4.8}$$

The translation component can be passed as is.

In addition to the translation and rotation, the velocities of those are calculated and included in the input vector of the Autoencoder. This is done simply by taking the positional difference between each consecutive frame. For example, to calculate the velocity between the 2nd and 3rd frame for the translation along the x axis

$$V_2 = T_{x_2} - T_{x_1} \tag{4.9}$$

The first and last velocities are assumed to be 0. This is done for the translation vector and the three rotation unit vectors.

In conclusion, the input vector for the Autoencoder consists of

$$[T, M, V_t, V_r] \tag{4.10}$$

with a shape $N \times 24$. Specifically, vector for translation, 3 vectors for rotation (1×12) and all of their velocities (1×12).

4.3.3 Striding

Lastly, both vectors are being aligned by cutting the longer one and ensuring the same length.

In order to provide enough surrounding context for each frame, the audio data is padded with MFCC frames calculated on silence and then reshaped in strides.

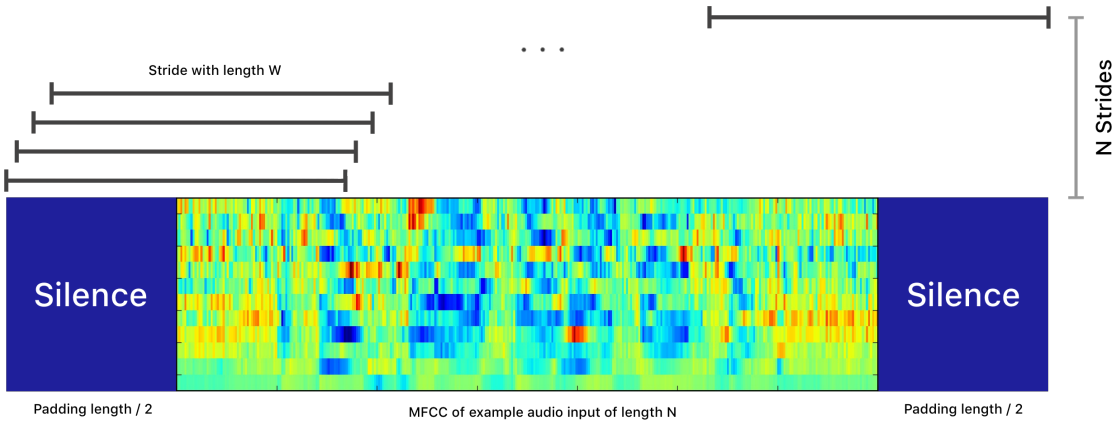


Figure 4.4: Striding strategy

Figure 4.4 shows the process for a single input example. A chunk of silence with a given number of frames, say N, is used to calculate MFCCs. Those are split in half and prepended and appended to the original MFCC vector, in essence adding silence at the beginning and end of the audio snippet. For each frame in the original signal, a context window with length $N+1$ is taken (the frame + two surrounding $N/2$ chunks).

4.4 Denoising Autoencoder

This section will describe the technical details for the Representation Learning step using the Denoising Autoencoder as applied to this project's specific dataset (a visual overview is seen in Figure 4.5).

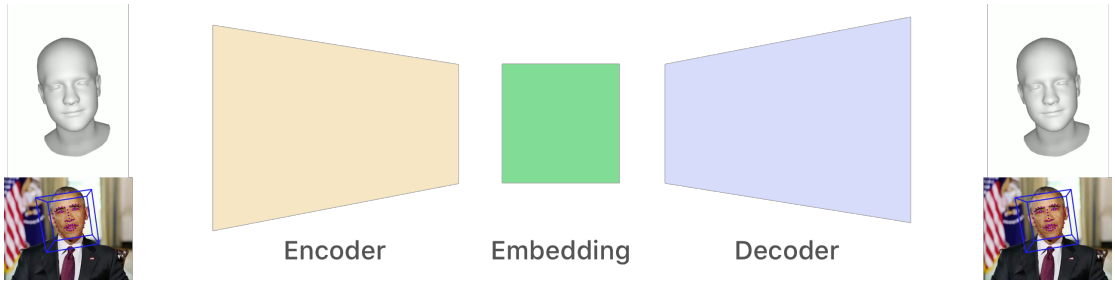


Figure 4.5: Headpose motion Autoencoder

Several pre-processing steps were described earlier in order to prepare the data to be fed to the Autoencoder. There are additional augmentation steps that can be performed to facilitate the process of learning the reconstruction of the input. Initially, each group (training, validation, and testing) is centred around the mean of the whole group

$$\mathbf{X'}^{(n)}_{train} = \mathbf{X}^{(n)}_{train} - \bar{\mathbf{X}}_{train} \tag{4.11}$$

$$\mathbf{X'}^{(n)}_{validate} = \mathbf{X}^{(n)}_{validate} - \bar{\mathbf{X}}_{validate} \tag{4.12}$$

$$\mathbf{X'}^{(n)}_{test} = \mathbf{X}^{(n)}_{test} - \bar{\mathbf{X}}_{test} \tag{4.13}$$

Then each group is normalised in the range of -1 to 1 by dividing the whole group by the maximum value in each.

By design the Denoising Autoencoder corrupts the input data, in this case with *Gaussian Noise*.

$$\hat{\mathbf{x}}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \sigma^2, I) \tag{4.14}$$

The DAE learns to reconstruct the input head pose data as closely as possible by minimising the mean squared error (MSE) loss function

$$MSE(\mathbf{m}, \hat{\mathbf{m}}) = ||\hat{\mathbf{m}} - \mathbf{m}||_2^2 \tag{4.15}$$

Finally, the system is ready to learn. Chapter 5 will describe the specific parameters and quantitative results of the training phase.

4.5 SpeechToDecoded network

After the DAE is trained, the whole dataset is encoded and the encodings are stored. The next task is to learn the mapping between the input speech data and the compact embedding of the head pose that the decoded data represents.

The network uses the Adam [50] optimiser, a widely-used stochastic gradient decent method that is based on adaptive estimation of the first and second-order movements. Each layer hidden layer uses the Rectified Linear Unit (ReLU) [51] activation function (Figure 4.7). Lastly, a Dropout layer is applied after each activation. The Dropout technique [52] is a widely used regularisation technique to prevent the network to overfit, by simply randomly turning off a specified number of nodes in the graph.

4.6 Temporal smoothing

The final results will be described in Chapter 5, however a common problem is temporal inconsistencies and jerking motion that tends to look unnatural. This has been a consistent problem with this approach in the papers and results of Kucherenko et al. [12], [10] and Hasegawa et al. [13]. In fact, in this project exacerbates the problem with employing a much noisier data set than the original work. Even though having to post-process the data after the neural network is generally a sign that the architecture is not optimal and a better end-to-end learning task is possible, a simple temporal filter can be applied to improve visual aspect of the results. Both filtered and unfiltered results are presented Chapter 5.

4.7 Rendering

The rendering is done using PyRender, an OpenGL-based rendering 3D rendering library for Python. A crucial difference between OpenCV and OpenGL is their coordinate system conventions, namely the y and z axis are flipped so in order to visualise the results correctly, this transformation was done (see Figure 4.8).

4.8 Conclusion

Once both networks are trained, it is possible to use the network to predict head pose motion on new, unseen by the model, speech audio. To achieve this, the audio example has to be processed using the same pipeline as the training

data. Then, it can be passed to the SpeechToDecoded network, which will yield predictions in the embedded space of the HeadPoseDecoder network. Finally, that decoder is used to bring the predictions in the desired space, namely per-frame 3D positions and rotations. The network also produces the velocities it was trained on, however those can be discarded.

The following chapter will present both the quantitative and qualitative results of this process.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_1 (TimeDist (None, 61, 256)           6912
_________________________________________________________________
batch_normalization_1 (Batch (None, 61, 256)           1024
_________________________________________________________________
activation_1 (Activation)    (None, 61, 256)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 61, 256)           0
_________________________________________________________________
time_distributed_2 (TimeDist (None, 61, 256)           65792
_________________________________________________________________
batch_normalization_2 (Batch (None, 61, 256)           1024
_________________________________________________________________
activation_2 (Activation)    (None, 61, 256)           0
_________________________________________________________________
dropout_2 (Dropout)          (None, 61, 256)           0
_________________________________________________________________
time_distributed_3 (TimeDist (None, 61, 256)           65792
_________________________________________________________________
batch_normalization_3 (Batch (None, 61, 256)           1024
_________________________________________________________________
activation_3 (Activation)    (None, 61, 256)           0
_________________________________________________________________
dropout_3 (Dropout)          (None, 61, 256)           0
_________________________________________________________________
gru_1 (GRU)                  (None, 256)               393984
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 24)                6168
_________________________________________________________________
activation_5 (Activation)    (None, 24)                0
=================================================================
Total params: 542,744
Trainable params: 540,696
Non-trainable params: 2,048
_________________________________________________________________
```

Figure 4.6: SpeechToDecoded

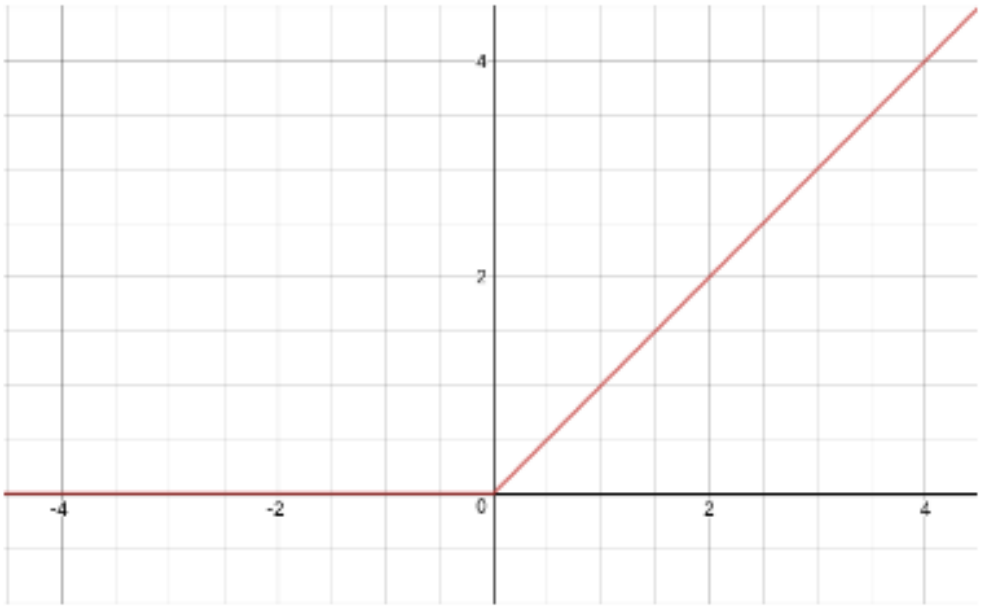


Figure 4.7: The ReLU activation function [51]

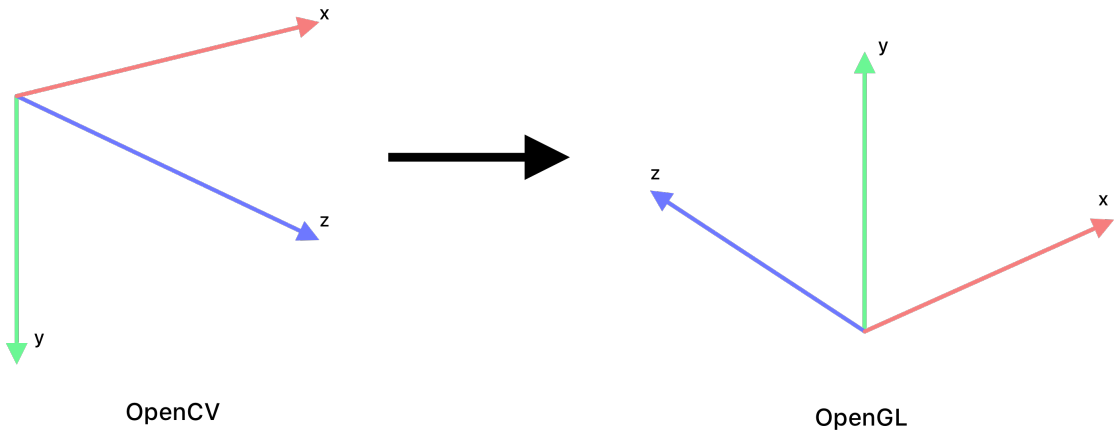


Figure 4.8: Coordinate system change

Chapter 5

Evaluation

5.1 Pre-processing

The audio was processed into strides with a window of 60, meaning the length was 60 + 1, given the original frame in the middle. The input stereo audio is transformed into mono using a simple average of the two channels. Table 5.1 summarises the split and length of the dataset which was pre-processed and trained on. The original sample-rate of the audio files is 48,000 Hz, at 16 bits per sample. The hop length of the MFCC is 0.01 seconds, meaning 100 times a second, or at 100 FPS. An average is applied merging every 5 frames and dropping the FPS to 20. The analysis window size is 0.02 seconds. The number of filters in the filterbank is set to 26, the FFT size is 512. After processing, the first 26 cepstrum coefficients are taken.

Table 5.1: Dataset details

	videos	minutes	frames
training	38	33:44	40438
validation	4	6:33	3432
testing	3	2:51	7871

5.2 Head Pose Embedding

The Denoising Autoencoder architecture was trained on the custom head pose dataset. That means the input and the output of the Autoencoder correspond to the shape of the data—namely layers with 24 width. The Adam optimiser was configured with:

- learning rate: 0.0001
- 1st momentum exponential decay: 0.9
- 2nd momentum exponential decay: 0.999

The network converges around 80 epochs (Figure 5.1). In the first experiment, the embedding dimension is the same as the input dimension.

A crucial trade off was observed when running tests on the Autoencoder. Namely, while the input was being successfully compress, denoised and decoded back, some of the expressivity, such as sharp movements and rotations, were being lost. Three examples of the reconstruction performance are shown, Figures 5.2, 5.3, 5.4.

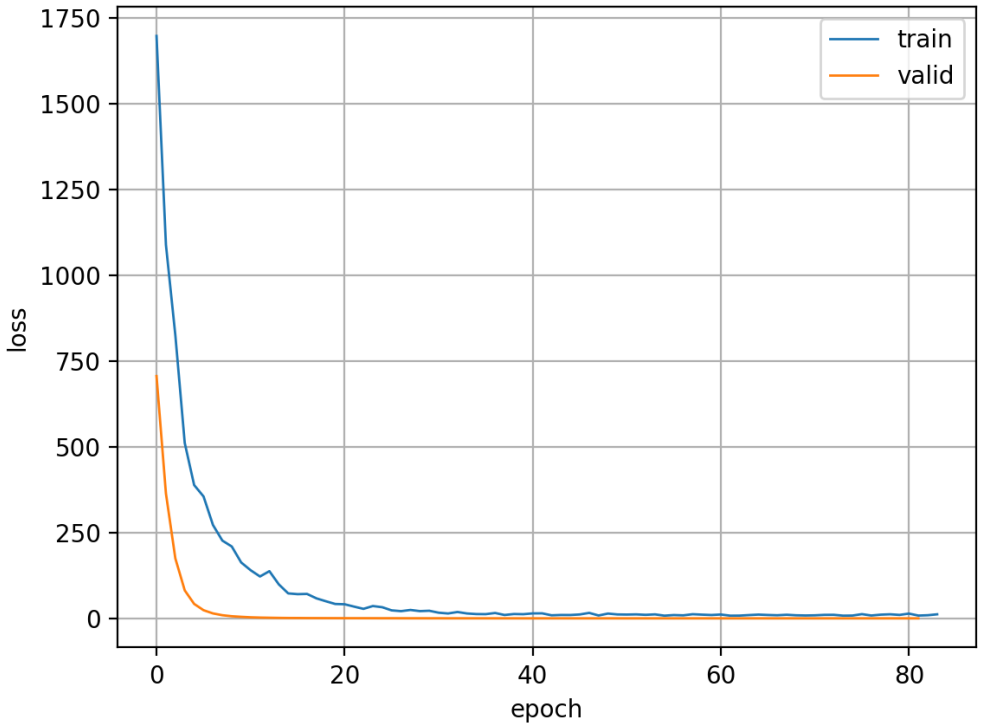


Figure 5.1: Convergence of the Motion-Encoder network

A frame-per-frame comparison (Figure 5.5) can be seen in the provided link.

5.3 Speech-To-Encoded

The architecture for the Speech-To-Encoded network was shown in Figure 4.6. The Adam optimiser was configured with:

- learning rate: 0.001
- 1st momentum exponential decay: 0.9
- 2nd momentum exponential decay: 0.999 Dropout is applied at every layer

Figure 5.2: Quantative comparison between the original head pose data and the decoded signal from the Autoencoder for test example 41.

with a 10% chances for the neurons to turn off.

The SpeechToEncoded network converges around 100 epochs (Figure 5.6).

Three different test examples are be shown below (Figures 5.10, 5.11, 5.12), both filtered and unfiltered. While the network has not seen test data (i.e. data the network has not trained on) the voice is from a common source actor (the Obama weekly addresses). Further, the examples are the same as the ones used to test the reconstruction performance of the HeadPoseDecoder network, so it is useful to compare.

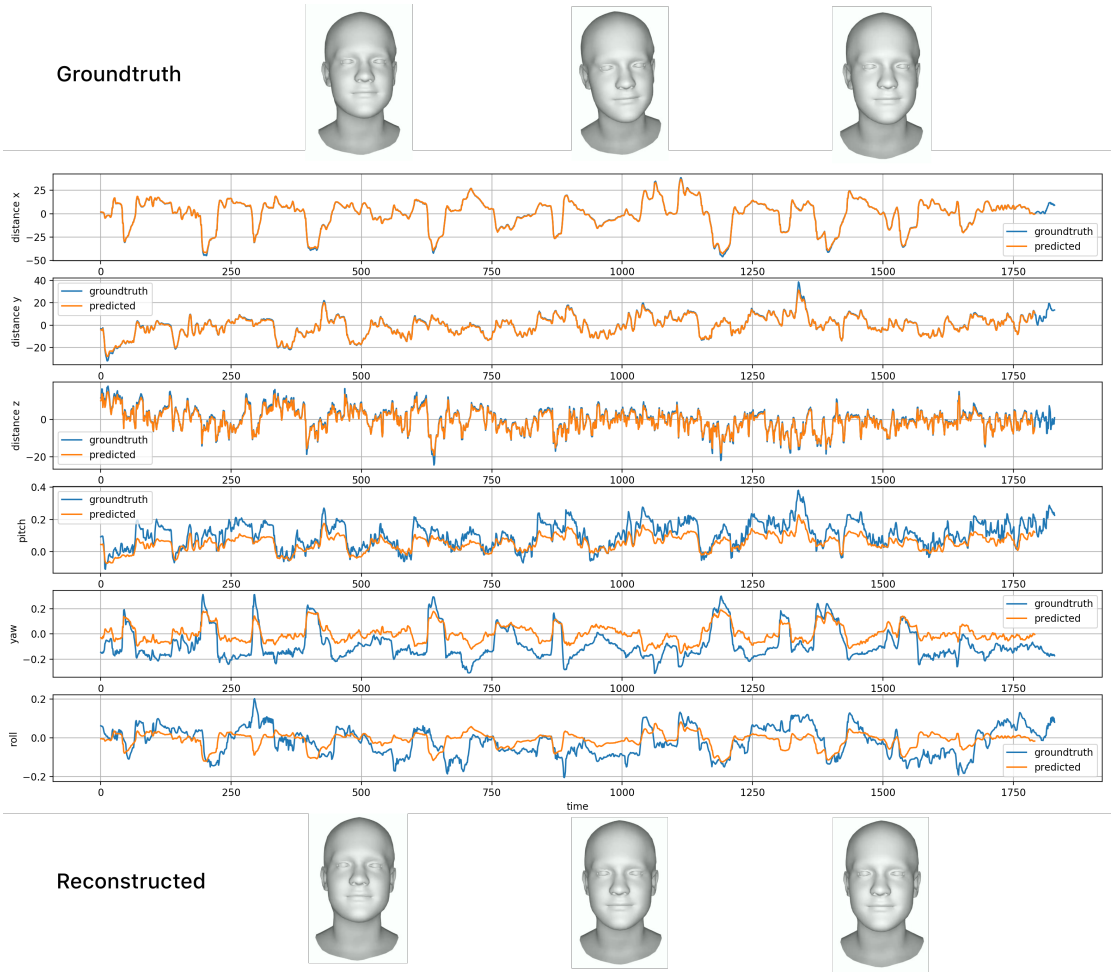


Figure 5.3: Quantative comparison between the original head pose data and the decoded signal from the Autoencoder for test example 43.

The final visualisations were combined with the model of Cudeiro et al. [16] to generate lip motion on the 3D face mesh from the same input speech audio that generated the the head pose. For full videos, please visit the online project page.[1]

Finally, a few arbitrary visual examples are shown that compare the results with the groundtruth data qualitatively (Figures 5.13, 5.14 and 5.15).

[1]http://nick-nikolov.com/masters-project

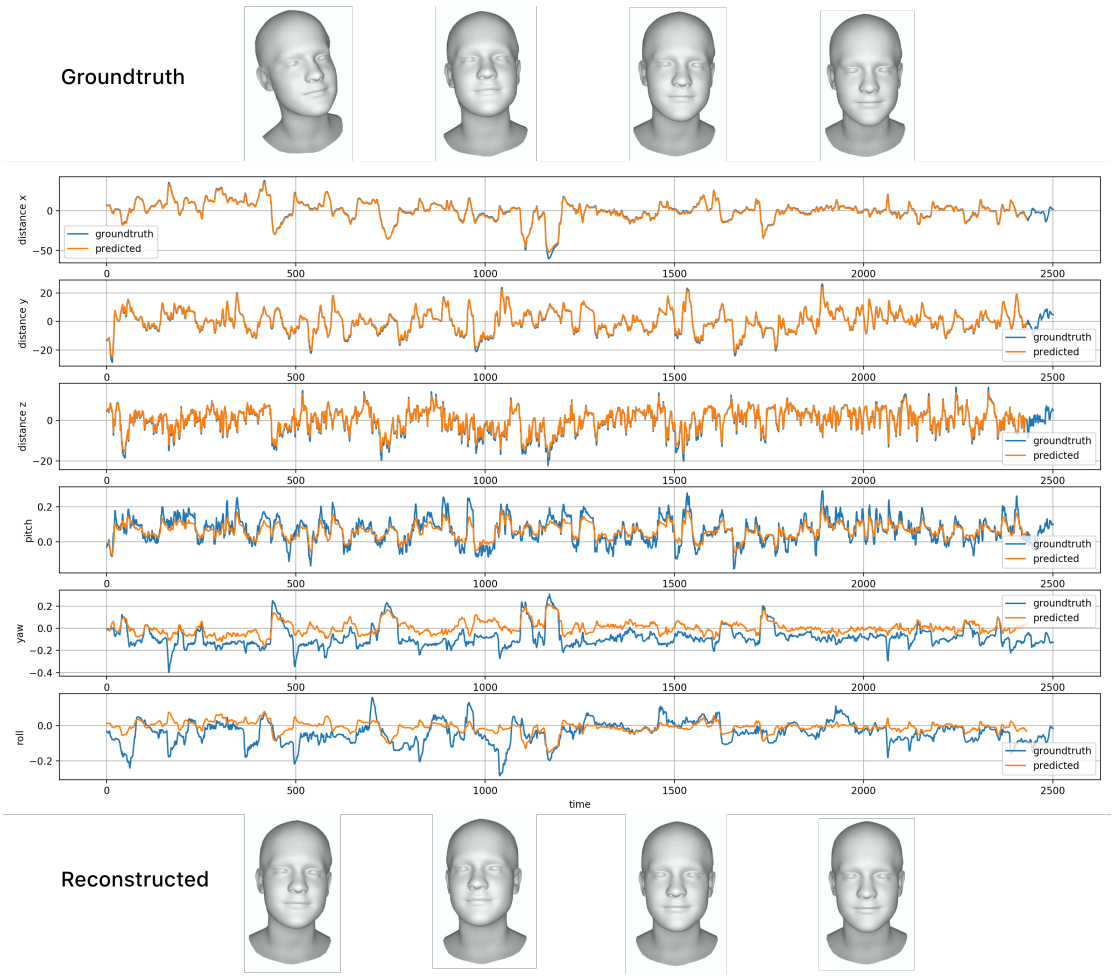


Figure 5.4: Quantative comparison between the original head pose data and the decoded signal from the Autoencoder for test example 45.

5.4 Discussion

As seen in Figures 5.13, 5.14 and 5.15, the network clearly learns some realistic and useful aspects of the source actor's head behaviour whilst speaking. In fact, it successfully captures the overall behaviour and range of motion. However, one major weakness is the tendency to slow down, avoid sharp, quick, expressive gestures. The reason for this can be one hand rooted in a poor of choice of parameters or architecture, or on the other an insufficiently strong relationship between the speech signal and the gesture. The obvious focus of the system here

Figure 5.5: Frame-per-frame comparison of arbitrary frames of original data and the reconstructed version by the Autoencoder. On the right, the original tends to be more expressive than the decoded (denoised) version on the left.

is the acoustic aspect of speech, leaving the obvious question of what the meaning of the content contributes to the motion of the actor unanswered.

Fundamentally, it is unclear if there is sufficient correlation between the speech signal and the head pose. One can imagine different phrases with a similar acoustic character that result in different gestural movements, even for the same person. Further, some gestures tend to persist over larger windows of time than this architecture is based on, then quickly changing as the meaning of the contents of the speech changes, which is a type of behaviour that will be difficult to capture with this type of system.

The outlined problems are all inherently solvable by attaining more data

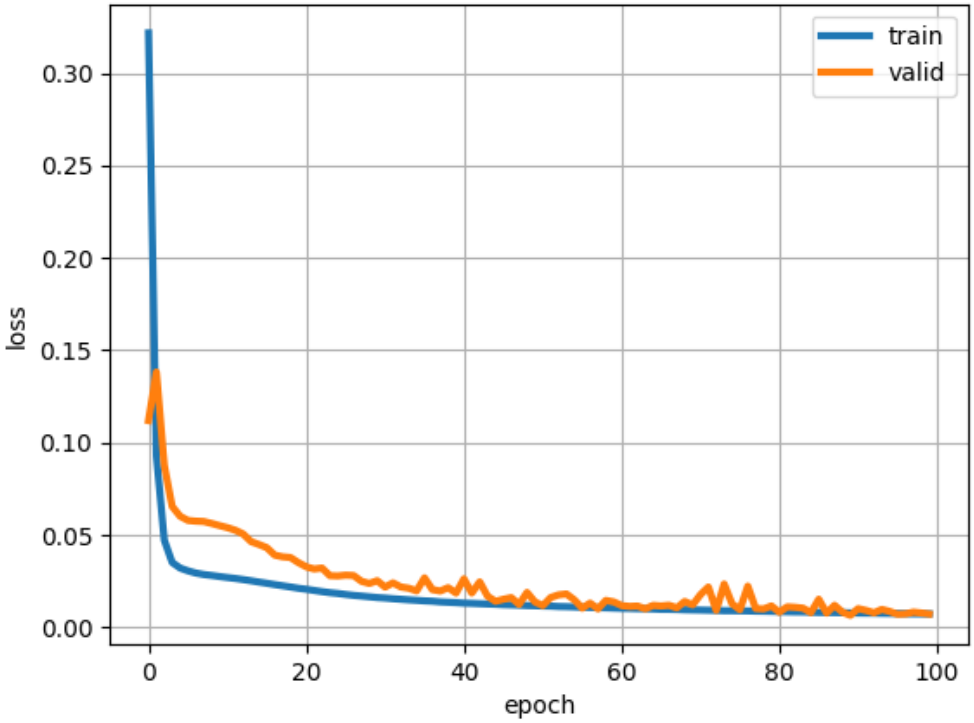


Figure 5.6: SpeechToEncoded convergence.

and improving on the architecture of the learning task. Chapter 6 will expand on some of the possible avenues of research on this topic.

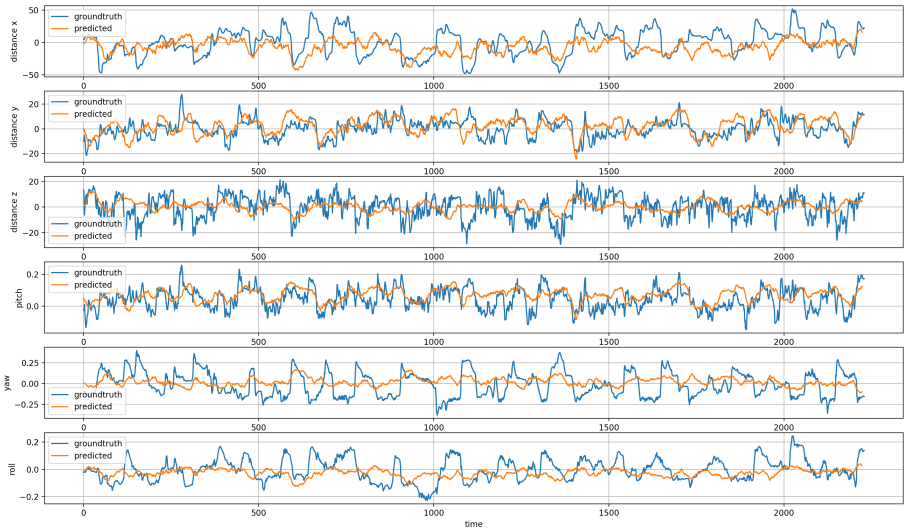


Figure 5.7: Quantitive comparison between the original head pose data and the decoded and **unfiltered** prediction from test example 41.

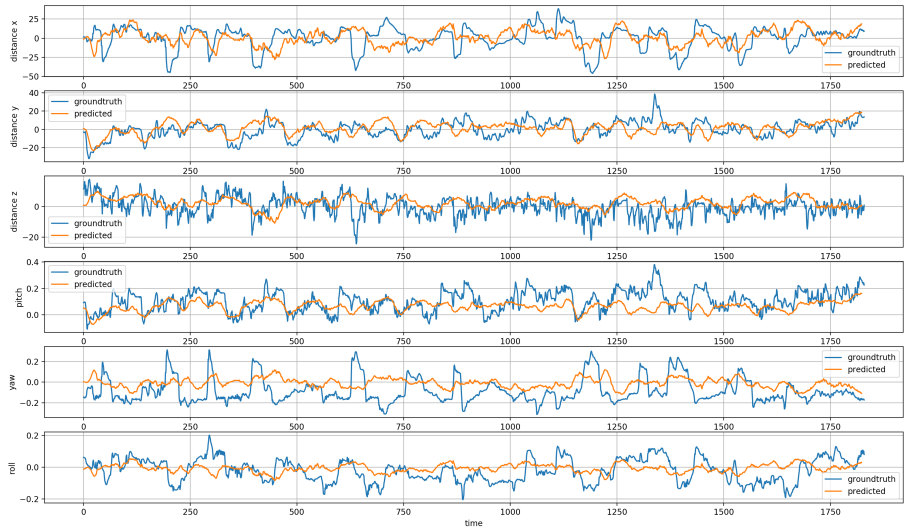


Figure 5.8: Quantitive comparison between the original head pose data and the decoded and **unfiltered** prediction from test example 43.

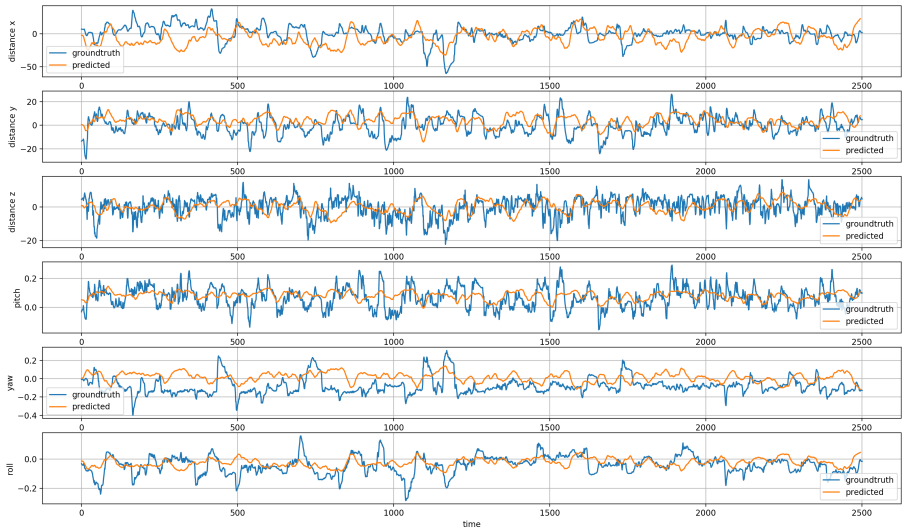


Figure 5.9: Quantitive comparison between the original head pose data and the decoded and **unfiltered** prediction from test example 45.

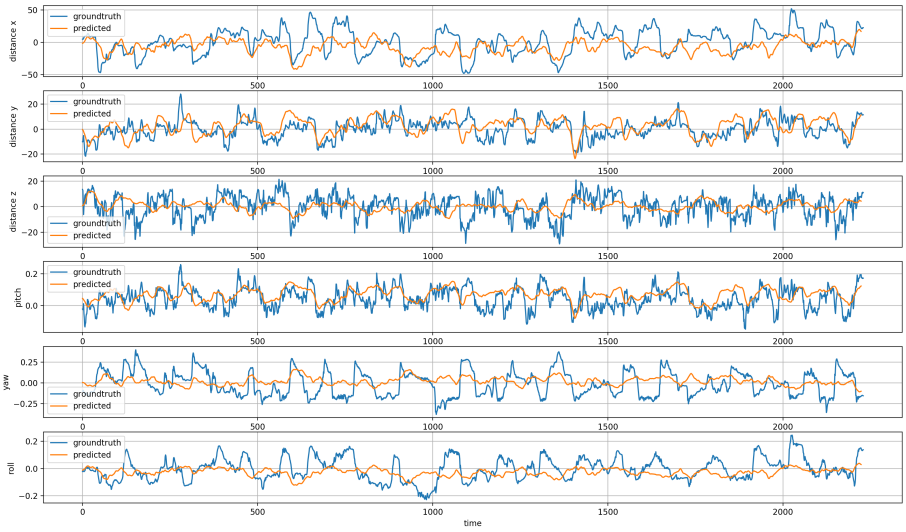


Figure 5.10: Quantitive comparison between the original head pose data and the decoded and **filtered** prediction from test example 41.

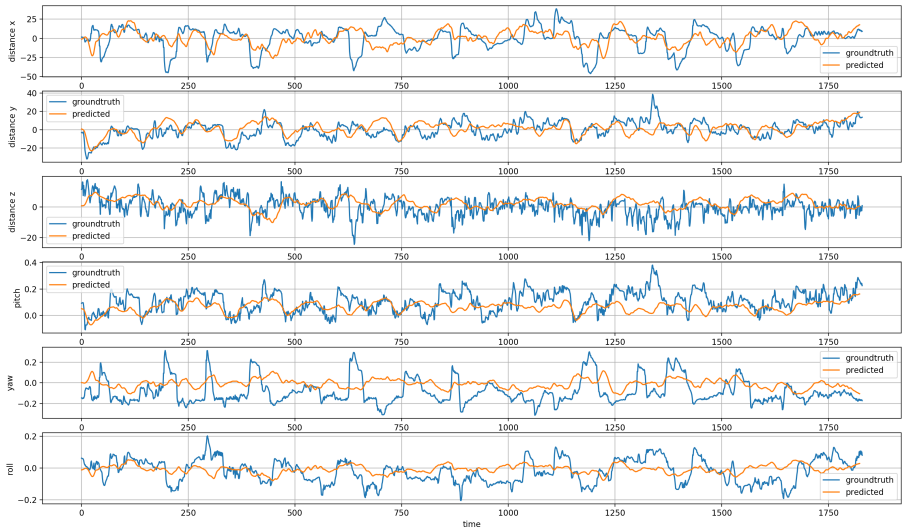


Figure 5.11: Quantitive comparison between the original head pose data and the decoded and **filtered** prediction from test example 43.

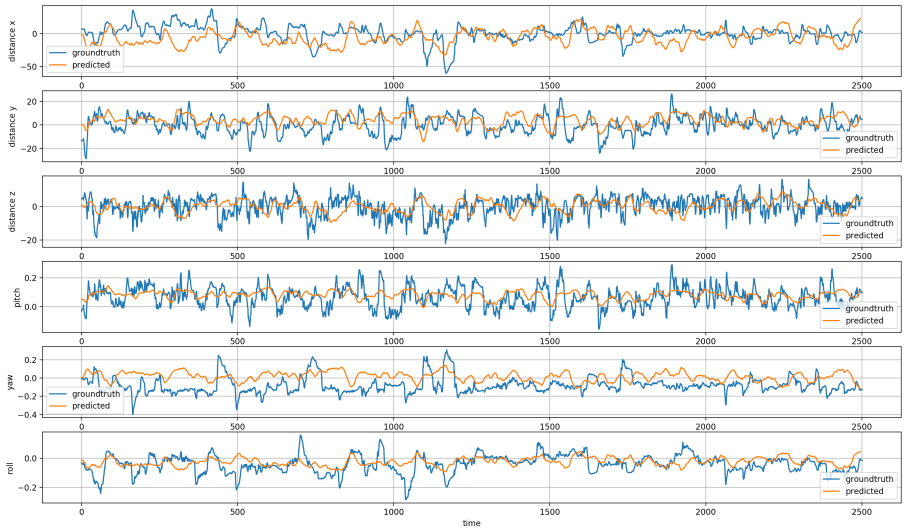


Figure 5.12: Quantitive comparison between the original head pose data and the decoded and **filtered** prediction from test example 45.

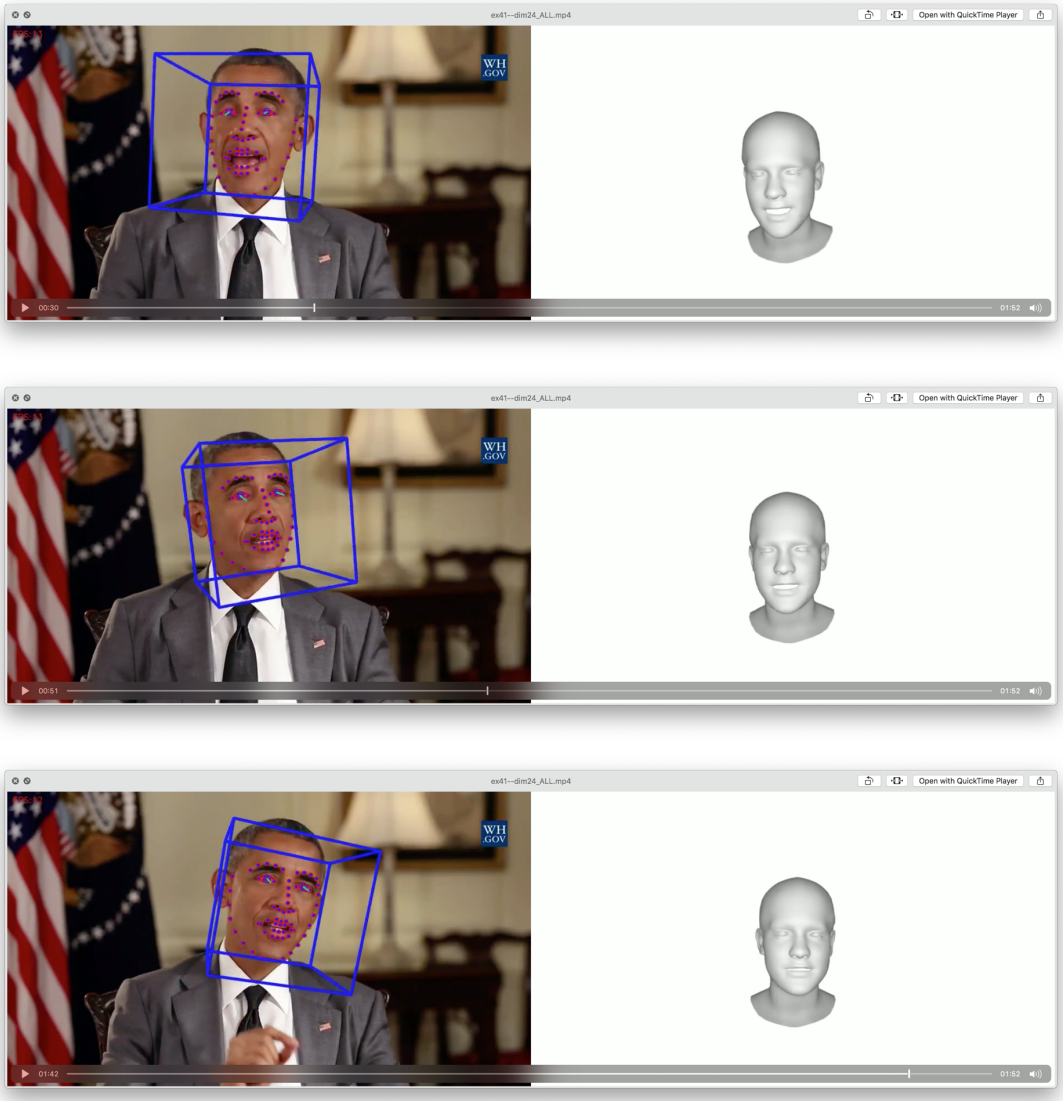


Figure 5.13: Example 41: Frame-by-frame comparison of SpeechToEncoded prediction (right) and groundtruth data (left).

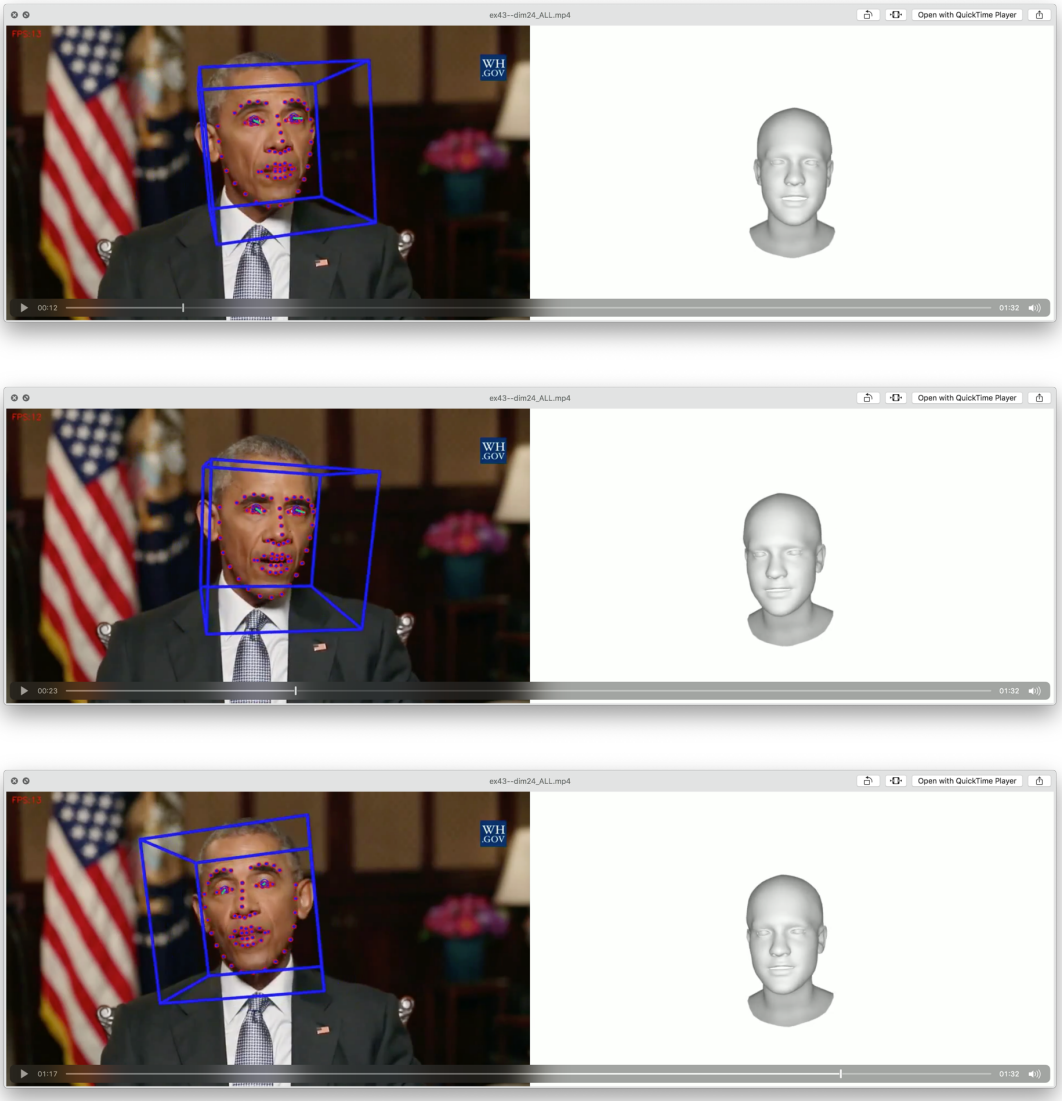


Figure 5.14: Example 43: Frame-by-frame comparison of SpeechToEncoded prediction (right) and groundtruth data (left).

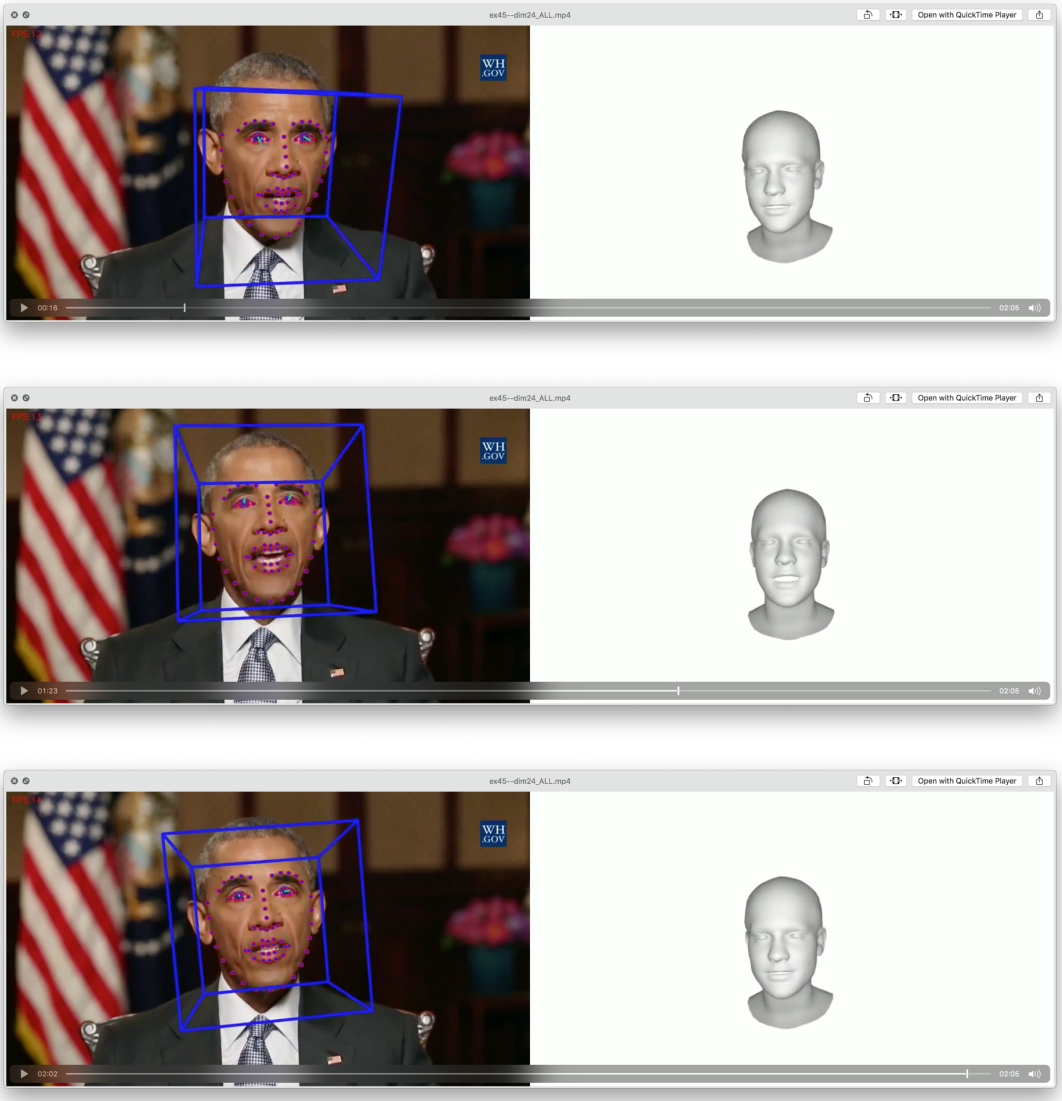


Figure 5.15: Example 45: Frame-by-frame comparison of SpeechToEncoded prediction (right) and groundtruth data (left).

Chapter 6

Conclusion and Future Work

This thesis has provided a theoretical background and experimental results as to the possibility of mapping a speech signal to deterministically determine the head pose, made up of a translation and rotation component, of a single speaking person. A data-driven machine-learning approach was leveraged and successfully shown that this process can be automated. Further, instead of using hard to acquire, studio-quality motion capture datasets, online footage was collected, analysed and pre-processed to be made applicable for the task.

Still, many constraints were imposed in order to achieve these results. The dataset contains only a single person, meaning the speaking style and movement that the system generates will inherently mimic the style of the source actor in the dataset. The environmental setting was formal (a weekly presidential addressing) and facial analysis was made easier by the fact the source person's head was looking into the camera for the whole duration of the speech. Audio quality was consistent, and each utterance was loud and clear. The same applies to lighting conditions and camera movement. Clearly, these are close to ideal factors that

will not apply for most "in the wild" footage. Still, it is the case that the proposed techniques can be indeed extended for many types of similar footage and different actors, producing models for different contexts and styles, or perhaps growing the existing dataset and generalising the model for multiple persons.

One of the main limitations of the proposed system in this thesis is the deterministic nature of the data generation process. A natural extension would to be look for ways to model the latent space probabilistically, opening up many creative possibilities. In a very recent paper, Alexanderson et al. [11] address this problem and propose a probabilistic system with impressive results.

In learning lip movements, Cudeiro et al. [16] leverage a pre-trained Deep Learning that has learned speech feature. This is an interesting alternative to the MFCC-based one proposed in this paper and a potential direction for improvement.

Another obvious line of work is improving the performance of such a system. The history of Computer Graphics is a history of non-real-time systems becoming real-time so it is only natural to expect Deep Learning synthesis to follow suit. One can imagine the desirability of real-time performance in tasks such as mobile communications, online presentations and so on. Further, creative controls of motion using an interface in real-time would be much preferable to artists over the tedious cycle of waiting for a render to finish, changing parameters and rerunning the process from scratch.

In terms of rendering, the literature has shown the flexibility and impressive realism that GANs can achieve. In that sense, the pipeline of this project can be extended and be fed to another GAN rendering system, similar to what Yi et al. [15] propose.

While the literature is mostly focused on the more obvious body movements, such as the gesturing of the hands, lip motion or, what this project focused on, head pose, there are other interesting non-verbal communication cues that can be explored. For example, mapping gaze direction or eyebrow movement to speech will further increase the realism of generative systems. With the sophistication and complexity of the generated data, rendering will too become a more difficult task.

Lastly, with the advent of voice-based personal assistants it will be desirable to test and extend motion synthesis systems to work well with synthetic voices. Indeed, while it would be a more natural engineering approach to synthesise the motion directly from the system that has generated the voice, it is plausible that using the audio speech as the source of the motion would increase realism and naturalness.

References

[1] S. C. Herring, A. R. Dainas, H. Lopez Long, and Y. Tang, "Animoji performances: 'Cuz i can be a sexy poop'," *Language@Internet*, vol. 18, no. 1, 2020.

[2] "How the irishman's groundbreaking vfx took anti-aging to the next level." https://youtu.be/OF-lElIlZM0 (accessed Apr. 27, 2020).

[3] E. Winick, "How acting as carrie fisher's puppet made a career for rogue one's princess leia." https://bit.ly/33f9qOD (accessed Apr. 27, 2020).

[4] "Cubic motion." https://cubicmotion.com/persona/ (accessed Apr. 27, 2020).

[5] J. Vincent, "Deepfakes helped charli xcx imitate the spice girls in her latest music video." https://bit.ly/2GDtAu8 (accessed Apr. 27, 2020).

[6] "The startup behind that deep-fake david beckham video just raised 3 million." https://tcrn.ch/35mtLEn (accessed Apr. 27, 2020).

[7] "Reuters and synthesia unveil ai prototype for automated video reports." https://reut.rs/2Fc1Nkc (accessed Apr. 27, 2020).

[8] "Synthetic weather reports." https://bbc.github.io/synthetic-weather-demo/ (accessed Apr. 27, 2020).

[9] E. Zakharov, A. Shysheya, E. Burkov, and V. Lempitsky, "Few-shot adversarial learning of realistic neural talking head models," in *2019 ieee/cvf international conference on computer vision (iccv)*, 2019, pp. 9458–9467.

[10] T. Kucherenko, D. Hasegawa, G. Henter, N. Kaneko, and H. Kjellström, "Analyzing input

and output representations for speech-driven gesture generation," *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*, 2019.

[11] S. Alexanderson, G. Henter, T. Kucherenko, and J. Beskow, "Style-Controllable speech-Driven gesture synthesis using normalising flows," *Computer Graphics Forum*, vol. 39, pp. 487–496, May 2020, doi: 10.1111/cgf.13946.

[12] T. Kucherenko, D. Hasegawa, N. Kaneko, G. E. Henter, and H. Kjellström, "On the importance of representations for speech-driven gesture generation," in *Proceedings of the 18th international conference on autonomous agents and multiagent systems*, 2019, pp. 2072–2074.

[13] D. Hasegawa, N. Kaneko, S. Shirakawa, H. Sakuta, and K. Sumi, "Evaluation of speech-to-gesture generation using bi-directional lstm network," in *Proceedings of the 18th international conference on intelligent virtual agents*, 2018, pp. 79–86.

[14] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen, "Audio-driven facial animation by joint end-to-end learning of pose and emotion," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017.

[15] R. Yi, Z. Ye, J. Zhang, H. Bao, and Y.-J. Liu, "Audio-driven talking face video generation with learning-based personalized head pose," Feb. 2020.

[16] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. J. Black, "Capture, learning, and synthesis of 3D speaking styles," in *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)*, 2019, pp. 10093–10103.

[17] J. Thies, M. Elgharib, A. Tewari, C. Theobalt, and M. Nießner, "Neural voice puppetry: Audio-driven facial reenactment," *arXiv preprint arXiv:1912.05566*, 2019.

[18] O. Wiles, A. Sophia Koepke, and A. Zisserman, "X2face: A network for controlling face generation using images, audio, and pose codes," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 670–686.

[19] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: Learning lip sync from audio," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017.

[20] J. Cassell, H. H. Vilhjálmsson, and T. Bickmore, "BEAT: The behavior expression animation toolkit," in *Proceedings of the 28th annual conference on computer graphics and interactive*

techniques, Aug. 2001, pp. 477–486.

[21] C. Huang and B. Mutlu, "Robot behavior toolkit: Generating effective social behaviors for robots," in *2012 7th ACM/IEEE international conference on Human-Robot interaction (HRI)*, Mar. 2012, pp. 25–32.

[22] V. Ng-Thow-Hing, Pengcheng Luo, and S. Okita, "Synchronized gesture and speech production for humanoid robots," in *2010 IEEE/RSJ international conference on intelligent robots and systems*, Oct. 2010, pp. 4617–4624.

[23] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[24] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[27] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proceedings of the 27th international conference on neural information processing systems - volume 2*, Dec. 2014, pp. 2672–2680.

[28] L. A. Gatys, A. S. Ecker, M. Bethge, and C. Sep, "A neural algorithm of artistic style. ArXiv 2015," *arXiv preprint arXiv:1508.06576*, 2015.

[29] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," 2018.

[30] "This person does not exist." https://thispersondoesnotexist.com/ (accessed Apr. 27, 2020).

[31] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *2017 ieee conference on computer vision and pattern recognition (cvpr)*, 2017, pp. 5967–5976.

[32] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *ArXiv*, vol. abs/1411.1784, 2014.

[33] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 ieee international conference on computer vision (iccv)*, 2017, pp. 2242–2251.

[34] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of the 26th annual conference on computer graphics and interactive techniques*, 1999, pp. 187–194.

[35] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, "Practice and Theory of Blendshape Facial Models," in *Eurographics 2014 - state of the art reports*, 2014.

[36] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero, "Learning a model of facial shape and expression from 4D scans," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 194:1–194:17, Nov. 2017.

[37] B. Egger *et al.*, "3D morphable face models—past, present, and future," *ACM Trans. Graph.*, vol. 39, no. 5, Jun. 2020.

[38] A. Y. Hannun *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *ArXiv*, vol. abs/1412.5567, 2014.

[39] S. Taylor *et al.*, "A deep learning approach for generalized speech animation," Jul. 2017.

[40] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L. Morency, "OpenFace 2.0: Facial behavior analysis toolkit," in *2018 13th IEEE international conference on automatic face gesture recognition (FG 2018)*, May 2018, pp. 59–66.

[41] H. Nugroho, "Fully convolutional variational autoencoder for feature extraction of fire detection system," *Jurnal Ilmu Komputer dan Informasi*, vol. 13, Mar. 2020.

[42] "Building autoencoders in keras." https://blog.keras.io/building-autoencoders-in-keras.html (accessed Aug. 30, 2020).

[43] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[44] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[45] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, 1994.

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[47] K. Cho *et al.*, "Learning phrase representations using RNN encoder–Decoder for statistical machine translation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Oct. 2014, pp. 1724–1734.

[48] 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*, vol. 28, no. 4, pp. 357–366, 1980.

[49] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems." 2015, [Online]. Available: http://tensorflow.org/.

[50] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014.

[51] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Journal of Machine Learning Research*, Jan. 2010, vol. 15.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.