Visual Speech Synthesis using Dynamic Visemes and Deep Learning Architectures

Ausdang Thangthai

A thesis submitted for the degree of

Doctor of Philosophy

University of East Anglia

School of Computing Sciences

April 2018

Abstract

The aim of this work is to improve the naturalness of visual speech synthesis produced automatically from a linguistic input over existing methods. Firstly, the most important contribution is on the investigation of the most suitable speech units for the visual speech synthesis. We propose the use of dynamic visemes instead of phonemes or static visemes and found that dynamic visemes can generate better visual speech than either phone or static viseme units. Moreover, best performance is obtained by a combined phoneme-dynamic viseme system. Secondly, we examine the most appropriate model between hidden Markov model (HMM) and different deep learning models that include feedforward and recurrent structures consisting of one-to-one, many-to-one and many-to-many architectures. Results suggested that that frame-by-frame synthesis from deep learning approach outperforms state-based synthesis from HMM approaches and an encoder-decoder many-to-many architecture is better than the one-to-one and many-to-one architectures. Thirdly, we explore the importance of contextual features that include information at varying linguistic levels, from frame level up to the utterance level. Our findings found that frame level information is the most valuable feature, as it is able to avoid discontinuities in the visual feature sequence and produces a smooth and realistic animation output. Fourthly, we find that the two most common objective measures of correlation and root mean square error are not able to indicate realism and naturalness of human perceived quality. We introduce an alternative objective measure and show that the global variance is a better indicator of human perception of quality. Finally, we propose a novel method to convert a given text input and phoneme transcription into a dynamic viseme transcription in the case when a reference dynamic viseme sequence is not available. Subjective preference tests confirmed that our proposed method is able to produce animation, that are statistically indistinguishable from animation produced using reference data.

Acknowledgements

First and foremost I would like to thank my supervisory team, Dr. Ben Milner, Dr. Sarah Taylor and Dr. Barry-John Theobald who is my former supervisor. Without your excellent guidance, advice, support and encouragement, this thesis would not have been possible.

I would also like to thank my external examiner, Dr. Hiroshi Shimodaira, and my internal examiner, Prof. Richard Harvey for your comments and suggestions which aim to improve the quality of this thesis.

I am grateful to the office of National Science and Technology Development Agency (NSTDA) and the Royal Thai government for funding and allowing me to study master and PhD at the university of East Anglia during 2012-2018. Thanks also go to members and interns of Disney research who helping made my life in Pittsburgh magical and wonderful for three months at the beginning of my PhD.

Last but not least, I would sincerely like to thank my family, Somsak Thangthai, Nongluck Thangthai, and my wife, Kwanchiva Thangthai, as well as my brother, Adisorn Thangthai, and my sister, Suluksana Thangthai, for their continued support and unconditional love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Visual speech synthesis or speech animation is the computer generation of a person speaking and is widely used in the entertainment industry for animated characters in games and films. In practice the process of creating a high quality speech animation is time-consuming and costly to obtain, because this requires expert artists and expensive equipment. For example, Hollywood films use motion capture technology with camera arrays capturing an actor's movement via sensors attached on their body and face. On the other hands, automated systems have not been widely used for industry applications. There are many reasons why automated systems with low cost equipments are still not used in this area. The high expectation of the quality for industry applications is one of the major reasons. Hence, the improvement of automated speech animation systems is the main goal of this work.

The progress of visual speech synthesis is less evolved than audio speech synthesis. Audio speech synthesis has been an active area of research in the last 20 years and has become commonplace in various real life applications, for example, car navigation and smartphones. Several techniques proposed for audio speech synthesis and for visual speech synthesis are the same or similar. Existing automatic visual speech synthesis systems can be classified as blendshape [16, 33], sample-based [10, 21, 35, 71, 72, 103, 105] and statistical-based [5, 56, 68, 89, 96, 99, 101, 102, 115, 122]. Blendshape approach is the simplest method for animating visual speech animation that first defined a different static facial blendshape in each block of speech units, for example one lip-shape in each viseme onset. Then, a various interpolation techniques is applied to make a facial movement between two consecutive visemes. One of the limitations of this approach is the ability to describe the complexity of real facial movement. Sample-based approach involves the concatenation of pre-recorded speech data and this is similar to the unit selection method of audio speech synthesis. The major benefit of this approach is the naturalness of the output quality. While, the major limitations of this approach are the output will look the same as the actors face in the database and the smoothness problem generally arises between any two consecutive units. Statistical-based method is used to overcome smoothing problem that learns a model of visual speech parameters instead of storing the original speech as in sample-based approaches and this is similar to the statistical method of audio speech synthesis. The benefit of this method is the potential to generate the smooth output and the ability to generate the flexible output with the compact model that do not appear in the training data. For this reason, we take the statistical-based approach in this thesis.

There are mainly two different types of input that can be used to drive visual speech synthesis; i) speech-driven animation system and ii) text-driven animation system. The former systems use an audio speech signal as an input and then generates a corresponding visual speech animation. The latter systems use text or a sequence of speech units as an input and then predict the visual speech output. In this thesis, we focus on the text-driven visual speech synthesis and examine suitable units of speech from three types of basic units: phonemes, traditional (static) visemes and dynamic visemes.

This thesis describes a method that can be considered a novel unit, called dynamic viseme units [98], and incorporated into statistical-based approach which is based on text-driven animation system. This thesis assume that natural speech is available and needed because we would like to focus on visual only speech synthesis system. Hence, the timing information of real speech in this thesis is derived by automatic speech recognisers or humans. Overall, we aim to answer the main research question that 'can we replace the conventional (audio) speech units (phonemes or (static) visemes) and find the suitable (visual) unit by incorporating the dynamic viseme unit into statistical-based approach to improve visual speech synthesiser?'.

1.1 Goal and Objectives

Our goal is to generate a facial animation of a person speaking and to improve the naturalness of the synthesiser from an input sequence of text over existing method. The objective of this thesis can be summarised as follows:

- To investigate the suitable basic building block of speech units for the visual speech synthesis.
- To incorporate dynamic viseme units of speech into hidden Markov model (HMM)-based visual speech synthesis.
- To combine dynamic viseme and phoneme units of speech into deep neural network (DNN)-based visual speech synthesis.
- To improve recurrent neural network long short term memory (RNN-LSTM)-based visual speech synthesis by combining dynamic visemes and phonemes.
- To use frame level information to avoid discontinuities and to produce a smooth and realistic visual speech output.
- To improve the method of a mapping phoneme sequence to a dynamic viseme sequence.
- To measure the performance using both objective and subjective tests.
- To use global variance as an objective measure to predict the outcome of subjective tests measuring realism and naturalness.

1.2 Contributions

This thesis is set to contribute to the visual speech synthesis community by achieving the objectives that described in Section 1.1. Additionally, the following is a list of publications arising out of this work by the author and have given contributions to the research area:

1. **A. Thangthai** and B. Theobald. "HMM-based visual speech synthesis using dynamic visemes". In Auditory-Visual Speech Processing, AVSP 2015, Vienna, Austria, September 11-13, 2015, pages 88–92, 2015 [101].

2. **A. Thangthai**, B. Milner and S. Taylor. "Visual speech synthesis using dynamic visemes, contextual features and DNNs". In Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016, pages 2458-2462, 2016 [102].

3. **A. Thangthai**, B. Milner and S. Taylor. "Synthesising visual speech using dynamic visemes and deep learning architectures". Computer Speech & Language. (Submitted).

1.3 Thesis Outline

We therefore organise the remainder of the thesis as follows:

Chapter 2, Overview of Visual Speech Synthesis: presents an overview of audio and visual text to speech (TTS) and then reviews work and techniques related to visual TTS. We also describe units of speech in a visual context.

Chapter 3, Data Corpora: presents our audiovisual speech dataset, namely KB-2k, that is used for training and testing the models in this thesis. We also describe the process of data preparation and data representation.

Chapter 4, Visual Speech Synthesis based on Hidden Markov Models: presents an overview of the Gaussian mixture model-hidden Markov model (GMM-HMM). We then introduce hidden semi-Markov model (HSMM) from a synthesis perspective. For the experimental section, we examine the most suitable unit of speech for visual speech synthesis using HMMs. We also explore the effect of the video frame rate, dynamic features, number of HMM states and visual feature normalisation on the synthesised visual output.

Chapter 5, Visual Speech Synthesis based on Feedforward Networks: presents an overview of feedforward networks and continues with a deep neural network (DNN)-based approach for predicting visual lip motion parameters from a text input. We combine both phoneme and dynamic viseme units and use their as units of speech. We consider using more low-level (frame-based) contextual information in the feature vector applied to the DNN which is derived from the speech unit annotations, with the aim of producing a more realistic and smooth visual feature trajectory.

Chapter 6, Visual Speech Synthesis based on LSTM-RNN: presents an overview of recurrent neural networks (RNNs) in terms of structure, architecture, training methods, and then considers long short term memory (LSTM). We describe how to develop a visual speech synthesis system using an encoder-decoder architecture with a bidirectional LSTM. In experiments, both objective and subjective tests are carried out to evaluate the effectiveness of various configurations.

Chapter 7, Phoneme-to-dynamic viseme visual speech synthesis: identifies the practical problem of obtaining a dynamic viseme sequence from input text. A traditional phoneme to dynamic viseme conversion is explored first and then a new conversion method is proposed. We evaluate both objective and preference tests of the synthesiser.

Chapter 8, Conclusion: presents a summary and comments on the research questions related to the contributions. We also discuss further work.

Chapter 2

Overview of visual speech synthesis

2.1 Introduction

An audio speech synthesis system (audio TTS) is the automatic generation of a synthetic voice from a given text, while a visual speech synthesis system (visual TTS) is the computer generation of a person speaking using animated graphics. Audio TTS is well evolved and has been around for many years, while visual TTS is less evolved and draws on techniques in audio TTS. It is obvious that there is a relationship between audio TTS and visual TTS in many view points. The existing techniques for synthesising a sound can be broadly categorised into model-based approaches, unit selection approaches and statistical-based approaches. Visual synthesis also categorises into blend-shape approaches and the same three approaches as audio synthesis systems. Additionally, the basic architecture is shared leading to sharing with the same front-end text processing module, as discussed in the following section.

The approach that is used in commercial applications is one of the major differences between audio TTS and visual TTS. With the development in audio TTS, most commercial applications are based on the unit selection approach (e.g. Real-

Speak [18]), statistical approach (e.g. Acapela[1]) or hybrid approach (e.g. Google [45], Siri [12]). Interestingly, the model-based approach (e.g. formant or articulatory synthesis) is not in favour as the result is generally lower quality as the models are not sufficiently accurate. Commercial applications in visual TTS, on the other hand, are based-on all approaches depending on the particular application. This is not surprising because model-based approaches in audio TTS cannot produce the acceptable speech quality, while the model synthesis approach in visual TTS can produce high quality graphics.

The following sections give an overview of the field as a whole, including input and output, speech units of a synthesiser, and a review of the various techniques relating to visual speech synthesis. The discussion begins by exploring the basic architecture for audio and visual synthesis. The individual components of the synthesiser are then discussed with a focus on visual synthesis.

2.2 Overview of text-to-speech (TTS) synthesis systems

This section aims to explore the basic work flow diagram for either audio TTS or visual TTS. The general block diagram consists of text processing as a front-end module and speech synthesis (audio or visual) as a back-end module, as shown in Figure 2.1. The front-end text processing module aims to convert a given text into linguistic features along with durations. This module roughly contains three steps; the conversion of non standard words to standard words (e.g. 7 to seven, dr. to doctor), the conversion of written form to sound form (e.g. yes to /y eh s/) and the prosodic analysis (e.g. phoneme duration prediction, phrase-break prediction). The back-end speech synthesis module aims to convert these linguistic features (e.g. a sequence of phonemes with its durations) into acoustic speech features for audio TTS or visual speech features for visual TTS.

[1]http://www.acapela-group.com

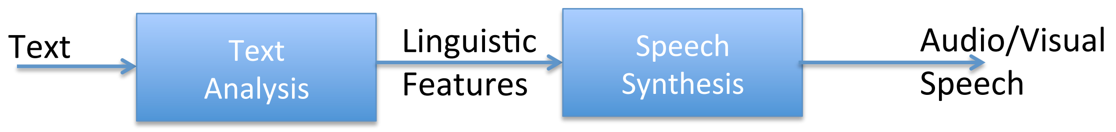


Figure 2.1: A basic speech synthesis framework for either audio TTS or visual TTS.

2.3 Input to the visual speech synthesiser

Text is typically used as an input property to the audio TTS synthesiser, but text is not the only possible input property to visual TTS. There are different categories of input used for generating synthetic visual speech. For example, Theobald and Matthews [104] classified the input of a synthesiser into unit-driven synthesis and feature-driven synthesis where units are a phonetic transcription and features are acoustic speech features. Moreover, Mattheyses and Verhelst [70] described three possible input types based on text, audio, and the image of a new target speaker. This section uses input information based on Gerard Bailly and Maxime Berar and Frederic Elisei and M. Odisio [42] that used two types of input to drive a synthesiser, specifically text input and audio input. More details are given in the following sections.

2.3.1 Text input

Text-driven visual TTS systems attempt to synthesise a visual speech signal from a text input, as can be seen in Figure 2.1. Written text is taken as an input and is transformed into visual speech in the form of a sequence of parameters as an output. The given text is not directly used because it is hard to find relationships between a written form and a sound or visual form. Some languages (e.g. English language) contains lots of ambiguity, for example, the same word can be pronounced in different ways depending on its part of speech (e.g. live to /liv/ or /laiv/). Moreover, the English word is not an appropriate unit because it is difficult to collect or create rules for all words. To construct a visual speech synthesis system from the input text, this raw input has to be converted to a sequence of lower level, sub-word, units instead (e.g. phonemes, visemes or syllables units).

It should be noted that several works in visual TTS borrow the text processing module from text to audio speech synthesis (TTS) systems, for example the MikeTalk facial model [32] used the Festival speech synthesis system [97]. This system converts the given text input into a sequence of phoneme units and timing information. Moreover, most visual synthesisers convert a sequence of phonemes to visemes because visemes (visual phonemes) are more appropriate for visual speech [37]. There have been many studies that try and identify a set of visemes from a set of phonemes, for example a many-to-one mapping from phoneme to static lip shape to produce the facial animation or many-to-many phoneme-to-viseme mapping. Unfortunately, there is still no standard set of visemes and visemes are also less well defined than phonemes. The standard visual speech units will be discussed later in Section 2.5. However, one phoneme or viseme with a static lip-shape was found to be insufficient to model natural lip animations. One of the reasons is that animation requires more contextual information to address visual coarticulation problems, where the current visual speech unit depends on the neighbour visual speech units. It should be noted that some languages are less complex than other languages. In the Japanese language, Tamura et al. [96] used 119 syllables as visual speech synthesis units rather than phonemes or visemes units. They found that the bigger units (e.g. syllables) were able to reduce discontinuities and produce more natural output animation.

Due to the increasing size of audiovisual speech databases, recently, the conversion of phonemes/visemes sequence and their timing information can be included in the linguistic and prosodic information input representations as used in TTS systems. [36] reported that this richer information would reduce the effect of visual coarticulation and improve the animation. From the emergence of machine learning techniques, various methods have been proposed to train a statistical model between this richer information and their corresponding visual features. Examples of learning machines are hidden Markov models (HMM) and deep neural networks (DNN). These are given more details in Section 2.6.

2.3.2 Audio speech input

Audio speech-driven visual TTS systems attempt to synthesise visual speech from an acoustic speech input. Overall, these take audio speech as an input and transform it into a set of visual speech parameters as an output. The general idea of this approach consists of two steps; speech feature extraction and learning of the mapping. As for the speech feature extraction step, various types of speech features have been used. For example, Mel-frequency Cepstrum Coefficients (MFCC) were employed by Tamura et al. [96], by Deena et al. [23], and by Theobald et al. [106]. It is interesting to note that the number of speech features in each work is not exactly the same number, and depends on training data and the task. Moreover, other well-known speech features include line spectral pairs (LSPs, [57]), line spectral frequencies (LSFs), [27], linear predictive coding (LPC), [27], and filter-bank coefficients [50]. As for learning of the mapping step, several classification techniques have been proposed, such as GMMs [14], HMMs [9], switching linear dynamical systems [30], switching shared Gaussian process dynamical models [23] and DNNs [99].

It should be noted that some researchers proposed a hybrid approach between speech-driven and text-driven. For example, [100] first converted the given audio speech signal into a sequence of phonemes using automatic speech recognition software (e.g. the Penn Phonetics Lab Forced Aligner [123]). Secondly, a text-driven approach was used to predict the visual speech signal from the corresponding phoneme label. An advantage of this framework is that it can generate the animation output from any language and any speaker, although is reliant on the accuracy of ASR.

2.4 Output modality of the visual speech synthesiser

This section presents the workflow of audiovisual speech synthesis systems (AVTTS) based on a two-phase and single approach.

2.4.1 Audiovisual synthesis system - based on a two-phase approach

Most traditional audiovisual TTS is a two-phase synthesis approach. The two-phase system consists of an audio TTS system and a visual TTS system. In the first step, a given text input is converted to a phoneme sequence and its duration and used to synthesise the acoustic speech signal with audio TTS. Afterwards, the phoneme sequence and its durations are used to predict the accompanying audio and video sequence with an audio speech TTS and visual speech TTS system, respectively. The final visual speech animation is produced by the combining the synthesised audio and visual speech, as shown in Figure 2.2.

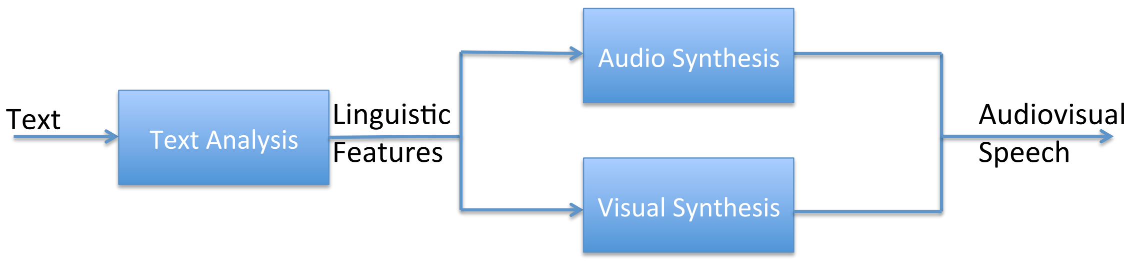


Figure 2.2: An audiovisual speech synthesis framework of the two-phase approach.

One of the benefits of this framework is that it is easy to identify the errors in each synthesiser. However, one drawback of using such a system is the processing time due to the time it takes for each synthesiser to operate. Moreover, the combination of two synthesised sequences that originate from different synthesisers can cause the McGurk effect [74], which may degrade intelligibility because of the mismatched between audio and visual.

In most cases, natural speech is available and needed, for example in the case of that applications require a high quality and/or expressive audio speech signal, as shown in Figure 2.3. It can be seen that this framework does not require a synthesised speech signal. In this case, the phonemes or viseme sequence and their timing information (labels) that corresponds to the natural speech is also required. The reason is that phoneme durations of natural speech are needed to generate a synchronous visual speech signal. Without these durations it may generate audio and visual sequences that are out of sync which will introduce sound delay problems. Afterwards, the phonemes sequence and its durations are used to predict a visual speech signal and are then combined with natural speech forming lip-sync speech animation.

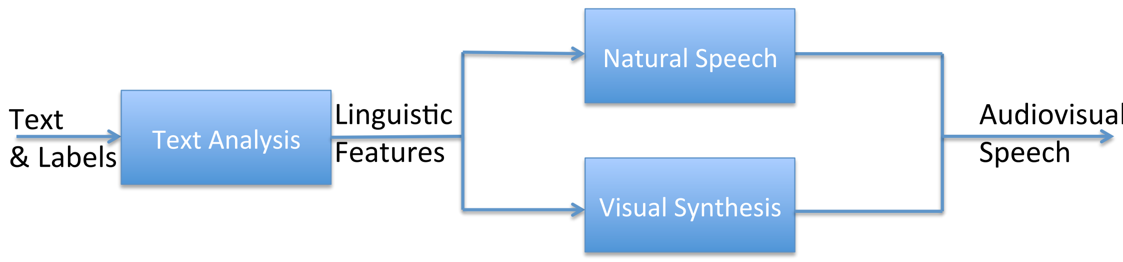


Figure 2.3: An audiovisual speech synthesis framework of the two-phase approach in the use of natural speech.

2.4.2 Audiovisual synthesis system - based on a single-phase approach

Recently, research has been concerned with single-phase synthesis to generate the highest possible coherence between the audio and visual modalities [72, 78]. In this framework, a given text input is used to simultaneously synthesise acoustic speech and visual speech with an audiovisual TTS system, as shown in Figure 2.4.

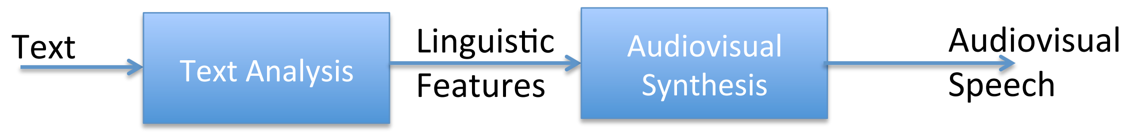


Figure 2.4: An audiovisual speech synthesis framework of single-phase approach.

It is believed that a joint model is able to improve the perceived quality of the synthesised audiovisual speech. Schabus et al. [89] also compared the results between the use of separate HMM-based audio TTS and HMM-based visual TTS. They concluded that the single-phase synthesiser is better than the two-phase synthesiser. However, the major drawback of combining the two systems is the difficulty in maximising the quality of the synthesiser. The reason for this is that the single-phase synthesis aims to generate coherence between two modalities rather than maximising the quality of individual modalities. Therefore, Mattheyses and Verhelst [70] pointed out that future research should focus on how to combine the high quality outputs of acoustic-only and visual-only synthesis.

2.5 Speech units for visual speech synthesis

In audio TTS, phonemes, or their derivatives (e.g. diphone and triphone), are used as the speech unit. However, for visual TTS, various speech units have been applied including phonemes, visemes and dynamic visemes units [98]. The following sections overview these three units.

2.5.1 Phoneme units

The smallest unit of speech in a language is a phoneme which is used to distinguish one sound from another. A sequence of phonemes forms words such as a sequence of /s iy/ refers to “see” in the English language (using the CMU phonetic dictionary). Moreover, the words “see” and “fee” are made different by changing the first phoneme from /s/ to /f/. The particular number of phonemes varies depending on the language and lexicon. For example, the CMU lexicon [82]comprises 39 phones while the Unisyn lexicon [38] comprises 56 phonemes. In this thesis, we use the Carnegie Mellon University pronunciation dictionary in the form of the ARPAbet phoneme set. The IPA and ASCII symbol for the consonants, vowels and diphthongs used in the ARPAbet phoneme set consists of 41 phonemes including silence, as shown in Table 2.1.

Clearly, the same phoneme in any word represents the same phonetic meaning and produces a similar sound. Hence, phonemes are used as the standard unit for various audio speech and visual speech applications such as automatic speech recognition and speech synthesis system.

Table 2.1: IPA symbols for the consonants, vowels, and diphthongs used in ARPA-bet, including the equivalent ASCII symbol and sample pronunciations from the CMU pronunciation dictionary.

IPA	ASCII	Example	Translation	IPA	ASCII	Example	Translation
a	aa	**o**dd	aa d	k	k	**k**ey	k iy
ae	ae	**a**t	ae t	l	l	**l**ee	l iy
ʌ	ah	h**u**t	hh ah t	m	m	**m**e	m iy
ɔ	ao	**ou**ght	ao t	n	n	k**n**ee	n iy
aʊ	aw	c**ow**	k aw	ŋ	ng	pi**ng**	p ih ng
ə	ax	disc**u**ss	d ih s k ax s	oʊ	ow	**oa**t	ow t
aɪ	ay	h**i**de	hh ay d	ɔɪ	oy	t**o**y	t oy
b	b	**b**e	b iy	p	p	**p**ee	p iy
tʃ	ch	**ch**eese	ch iy z	r	r	**r**ead	r iy d
d	d	**d**ee	d iy	s	s	**s**ea	s iy
ð	dh	**th**ee	dh iy	ʃ	sh	**sh**e	sh iy
ɛə	eh	**E**d	eh d	t	t	**t**ea	t iy
ər	er	h**ur**t	hh er t	ɵ	th	**th**eta	th ey t ah
e	ey	**a**te	ey t	ʊ	uh	h**oo**d	hh uh d
f	f	**f**ee	f iy	ᴜ	uw	tw**o**	t uw
g	g	**g**reen	g r iy n	v	v	**v**ee	v iy
h	hh	**h**e	hh iy	w	w	**w**e	w iy
ɪ	ih	**i**t	ih t	j	y	**y**ield	y iy l d
i	iy	**ea**t	iy t	z	z	**z**ee	z iy
ʤ	jh	**g**ee	jh iy	ʒ	zh	sei**z**ure	s iy zh er

2.5.2 Static viseme units

The smallest unit of visual speech in a language is a viseme which is used to group similar visual appearances, and was first introduced by [37]. From a visual point of view, when a person speaks, the lips, teeth and tongue are the only visible articulators, while the other places of articulation such as soft and hard palates cannot be seen. That means some phonemes can be grouped together as they are visually similar. For examples, the phones /p/, /b/ and /m/ are visually similar as they have the same place of articulation but differ in the manner of articulation. Hence, they are grouped as the same viseme.

Early approaches to visual speech animation grouped all phonemes into 10 visemes or static mouth shapes, as shown in Figure 2.5 and Table 2.2. For example, the first static lip shape, A, is the group of phonemes that refers to an open wide mouth with tongue and teeth visible. The process of creating speech animation begins with mapping any sequence of phonemes to the ten visemes, such as /s iy/ to /J A/. In this process each viseme is represented by one mouth shape called a keyframe. Finally, a sequence of mouth movements are keyframed to synchronise with the audio speech.

Table 2.2: Many-to-one phoneme-to-viseme mapping based on the ASCII symbol in Table 2.1.

Viseme	**Phoneme**
A	a, ae, ah, ao, aw, ay, iy
B	ax
C	eh, er, ey
D	f, v
E	l
F	m, b, p
G	ow, oy
H	uh, uw
I	w
J	ch,d, dh, g, hh, jh, k, ng, r, s, sh, t, th, y, z, zh

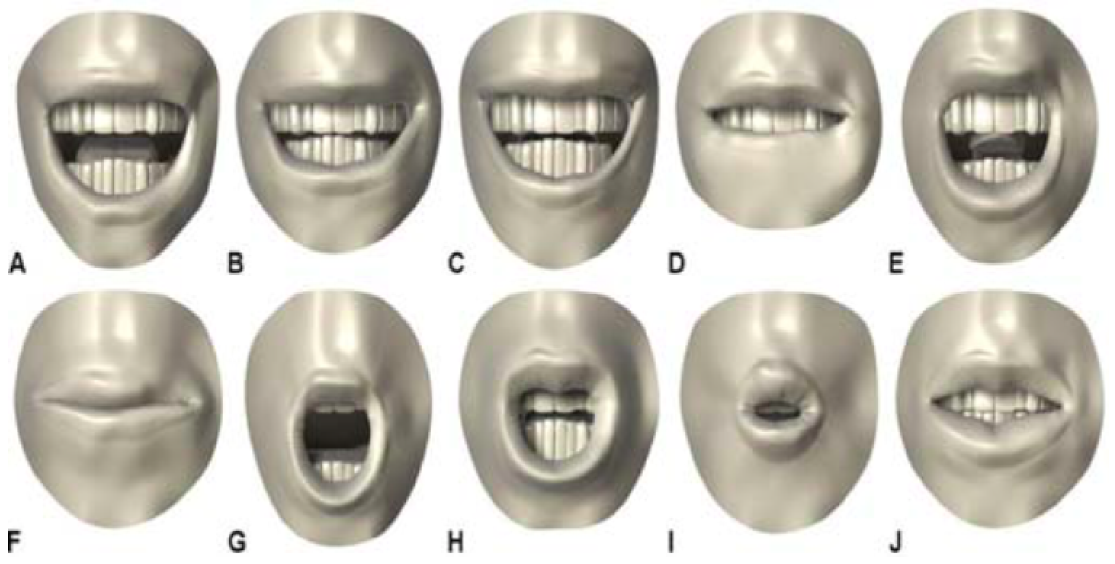


Figure 2.5: Examples of a lip shape in each viseme [113].

It can be seen that the mapping function from phonemes to visemes is many-to-one. Even now there is still no standard viseme set unlike the well defined phoneme set. Instead, several studies have created their own set of visemes [9, 26, 112], as shown in Table 2.3. These examples are specifically the consonant phoneme-to-viseme mapping and vowel phoneme-to-viseme mapping. For example, 27 consonant phonemes are mapped to 9 viseme groups and 17 vowel phonemes are mapped to 5 viseme groups using a decision tree in [3].

Table 2.3: Examples of a many-to-one phonemes to visemes mapping.

Classification	Viseme phoneme sets

Fisher [37]	{p,b,m}{f,v}{t,d,s,z,th,dh}{w,r}{ch,sh,jh,zh} {k,g,n,l.hh,ng,y}{sil,sp}
	{eh,ey,ae,aw,er,ea}{ah,ax,ay}{aa}{er,oh}{ao,ow,oy,ua}{uh,uw}{iy,ih,ia}

Dongmei [26]	{p,b,m}{f,v}{th,dh,t,d,s,z}{w,r}{ch,sh,jh,zh}{k,g,n,l,ng,h,y}{jj}
	{i,ii}{e,a}{aa,o}{uh,@}{oo},{u,uu}{w-au}{o-ou}

MPEG-4 [3]	{p,b,m}{f,v}{th,dh,t,d,s,z}{k,g}{ch,sh,jh,zh}{s,z}{n,l}{r}
	{ae,aw,ah,ax,ay,aa,axr}{eh,ey,er}{iy,ih}{ao,ow,oy}{uh,uw}

Bregler [10]	{p,b,m}{f,v}{th,dh}{t,d,s,z}{k,g,n,l}{ch,sh,jh,zh}{w,r}{hh}{y}{ng}

Bozkurt [9]	{p,b,m}{f,v}{th,dh}{t,d,l,n,en,el}{s,z}{ch,sh,jh,zh}{g,hh,k,ng}
	{ay,ah}{ey,eh,ae}{er}{ix,iy,ih,ax,axr,y}{uw,uh,w}{ao,aa,oy,ow}{aw}

Many factors contribute to the ambiguity of phoneme to viseme mapping; for instance, the variation of speakers, the language complexity, and the grouping criteria [103]. With coarticulation, the speech articulatory movements of the current speech sound depends on the neighboring phonemes. Hence, the use of viseme independent are replaced by context viseme dependent. Martino et al. [67] presented the use of phonetic context of each phoneme when grouping visemes, for example the phonetic contexts of /pu/, /upI/ and /upU/ are grouped into P2 and those of /ilU/ and /ulU/ are grouped into L4. Another interestingly from [70], they have an assumption that one sound can be represented by numerous lip shapes and tongue movements. For this reason, the previous ideas that aimed to group the same sound or its sound with phone contexts into the same group of visemes cannot be trusted. Hence, [70] proposed the use of many-to-many phoneme-to-viseme mapping instead of the mapping of many-to-one. Their experiment results also confirmed that many-to-many phoneme-to-viseme is shown as effective solutions compared with many-to-one mapping.

2.5.3 Dynamic viseme

Taylor et al [98] reported that the set of static mouth shapes from traditional visemes (e.g. many-to-one or many-to-many phoneme-to-viseme mapping) does not suit visual speech processing for two reasons. Firstly, the acoustic boundaries (e.g. phonemes or visemes) do not align with the visual boundaries. The reason is that the visible articulators tend to move before or after the inner articulators. For example, Figure 2.6 shows that for the sound /r/ the lips move before the sound is made. Hence, it is wrong to represent acoustic and visual speech with the same boundaries as phonemes and visemes.

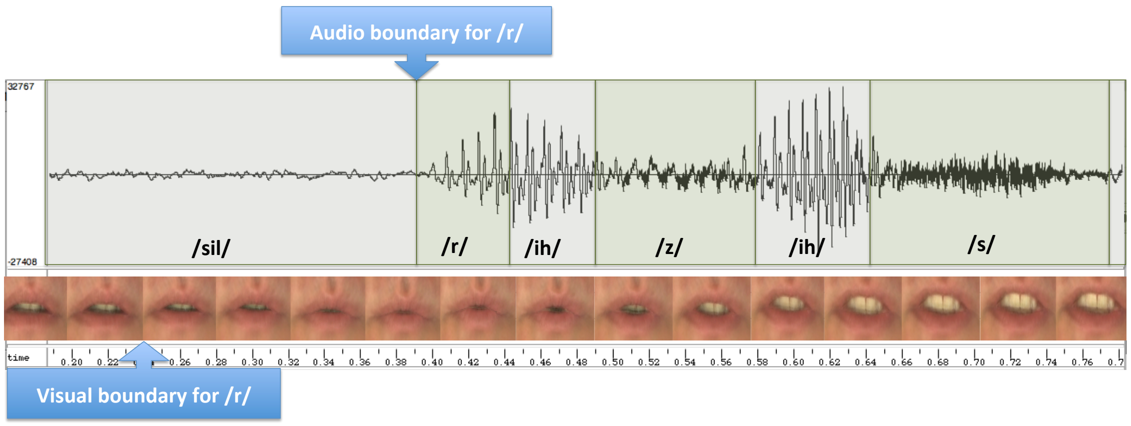


Figure 2.6: The difference between the beginning of audio boundary and visual boundary for sound "/r/".

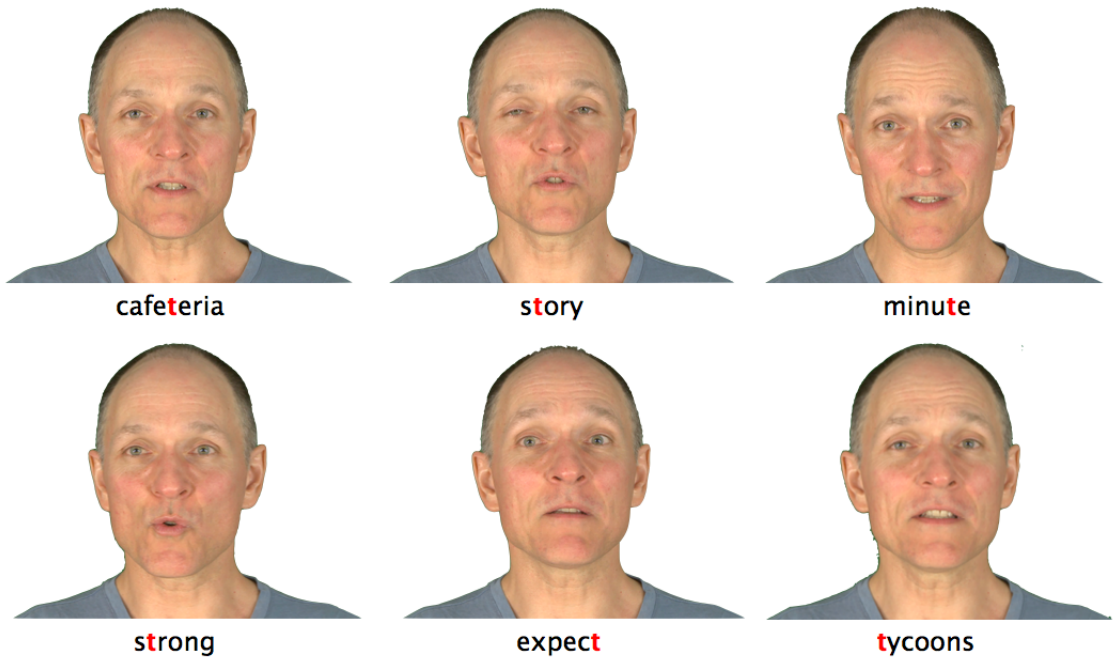


Figure 2.7: Different mouth shapes at the onset of phoneme /t/ in different word [98].

Secondly, they found that the representation of a single mouth shape for a viseme is not true in the real speech. They also pointed out that the same phoneme can be produced from many different lip-shapes. For example, Figure 2.7 presents different lip-shapes at the onset of phoneme /t/ in different words. This is because phonemes are learnt from the audio signal and the same abstract phoneme unit cannot distinguish one word from another but not necessary the same mouth shape.

To overcome these two limitations of traditional viseme and phoneme units, Taylor et al. [98] defined groups of similar lip movements (gestures), called dynamic viseme units, instead of groups of static lip shapes. These units can be considered as a basic unit for visual speech because a set of dynamic visemes is learned automatically by clustering visual speech parameters from a database. Additionally, the identification and clustering of the similar lip-motions are given in Section 4.4. Clearly, the boundaries of dynamic visemes are different to phoneme and viseme boundaries and the same abstract dynamic viseme has the same lip movements, as shown in Figure 2.8.

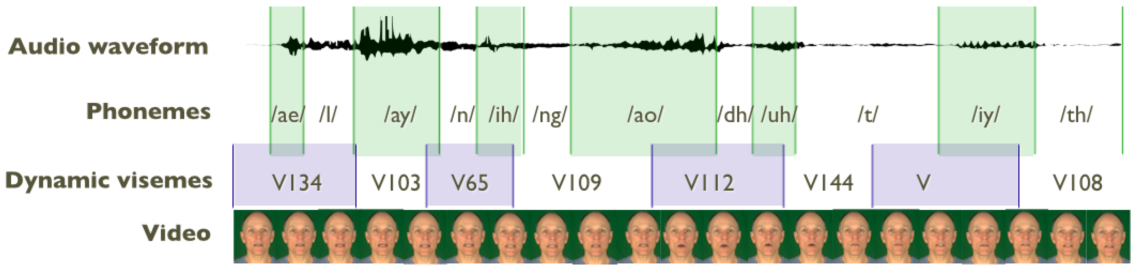

Audio waveform
Phonemes
/ae/ /l/ /ay/ /n/ /ih/ /ng/ /ao/ /dh/ /uh/ /t/ /iy/ /th/
Dynamic visemes
V134 V103 V65 V109 V112 V144 V V108
Video

Figure 2.8: The difference between phoneme and dynamic viseme units [98].

2.6 Approaches to visual speech synthesis

This section gives a review of the main techniques used for visual speech synthesis. These methods can be categorised into performance-based, blendshape-based, model-based, unit selection-based and statistical-based.

2.6.1 Performance-based approach

The performance-based approach is the commonly used in real applications, from mid-budget to large-budget productions. For the large budget productions, motion capture or motion tracking is used, with camera arrays recording an actor's movements via sensors that are attached on the face and/or body of the actor. These movements are then used to retarget visual animations by professional animators. It can be seen that this technique can generate high quality animations but it requires expensive hardware. Additionally, this is a time-consuming process, as the actor has to perform and the retargeted hand-crafted animation process needs artists controlling and generating an animation in every key-frame.

For mid budget productions, however, they use depth cameras (e.g. Microsoft Kinect) recording the facial movements signal instead of using motion capture. Then, the hand crafted animation is used to generate 2D or 3D facial speech animations from its signal. It clearly shows that this technique is able to produce animations without expensive equipment. Moreover, this approach is possible to do automatically using various techniques that will be discussed in the following sections, without any editing from experts to control and refine animations. However, one of the major limitations of the performance-based approach is that we need the actor back for any new animation.

2.6.2 Blendshape approaches

A blendshape approach is the most common and simplest method for animating 3D visual speech animation. This approach is inspired from the traditional hand-drawn cartoon animations, which first draw a few important frames (keyframe) by the primary (senior) artist. Then, the secondary (junior) artist would draw the remaining in-between frames. In the computer animation process, after creating a 2D or 3D character model, most systems first define a different facial blendshape in each viseme (e.g. opening mouth, round mouth) or each facial expression (e.g. neutral, angry), for instance the 10 static blendshapes in Figure 2.5. Note that, a blendshape can be referred to as a keyframe in animation. Then, the process of creating speech animation begins with mapping a sequence of visemes to its corresponding blendshape. To make a facial movement appear realistic, the in-between blendshapes are automatically determined by a variety of interpolation technique such as a linear and non-linear combinations. Figure 2.9. [91] illustrates a simple linear combination between two blendshapes.

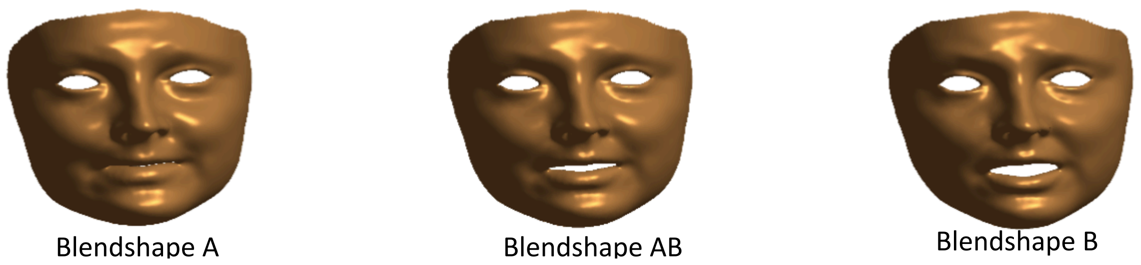


Figure 2.9: An example of interpolation technique using a simple linear combination [91].

In the early stages, it was found that the 10 static blendshapes in Figure 2.5 could not reliably describe the complexity of real facial movements. Several approaches have been proposed to increase the number of blendshapes. The most popular and widely used in both research and industry (e.g. game and movies [85]) is the facial action coding system (FACS). The FACS was first published in 1978 by Paul Ekman and Wallace Friesen [28] and was updated in 2002 [29]. It consists of 64 action units (AUs) to describe facial motion for example action unit 6 and 12 for cheek raiser and lip corner puller, respectively. The combination of these action units can form the emotion facial action coding system (EMFACS), for instance happiness is the combination of action unit 6 and 12. More interactive examples can be found in a visual guidebook website[2].

2.6.3 Model-based

In audio TTS, model-based approaches are referred to as formant-based auditory speech synthesisers, which are based on a parametric representation of speech, including both parallel and cascade resonators. In the past when the computers were underpowered, most commercial products were based on this technique. The DECTalk system is one example [52]. In terms of facial animation, the general ideas of model-based approaches consists of three tasks; (i) using predefined rules to determine appropriate sets of 2D or 3D facial parameters in each keyframe, (ii) generating in-between parameters using blendshape approach and reconstructing

[2]https://imotions.com/blog/facial-action-coding-system

images (iii) combining a sequence of images with synthetic audio generated by a separate audio text-to-speech system, e.g. MITalk, Festival, and Euler/MBROLA[3]. The first parameterised facial model was proposed by Parke [80], and also created the first 3D geometric model of the human face. Parke's facial model is hand-crafted by constructing the geometry of polygonal surfaces and is controlled by parameters that link to the important facial surfaces. For example, there are a set of parameters describing the size of the eyeball and the rotation of the jaw, which determines the mouth opening, the length of the nose and the scaling of the face. In terms of producing images, they applied a set of rules to deform these parameters. The major drawback of complex rules is the final facial animation is far from the movement of a real person. There are many descendants of Parke's facial model that try to improve the realism of the transitions between keyframes, such as Baldi by Cohen and Massaro [16], MASSY by Fagel and Clemens [35], and Kattis by Beskow [8], as shown in Figure 2.10.

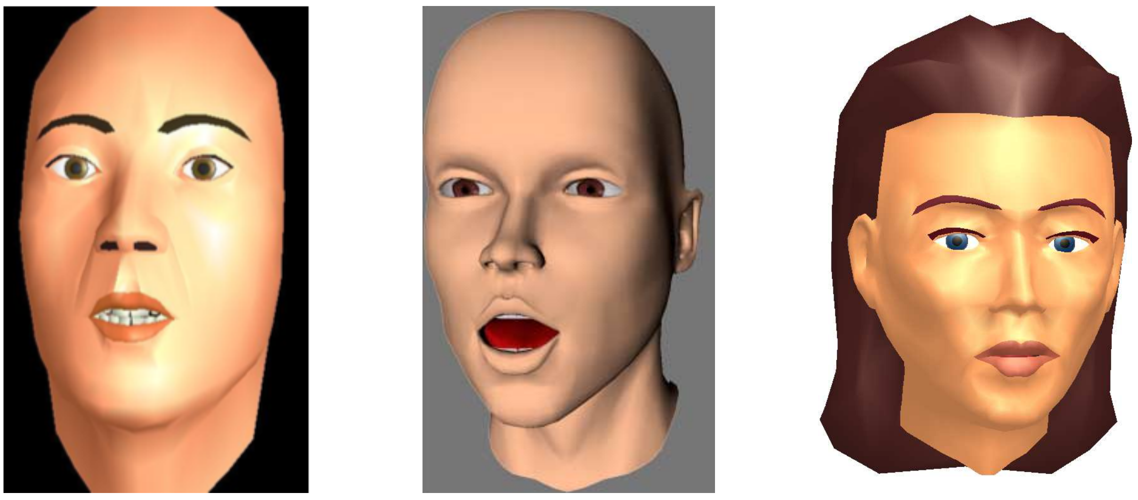


Figure 2.10: Descendants of Parke's facial model. From left to right: Baldi [16], MASSY [35] and Kattis [8]

Baldi is an adapted form of Lofqvist's articulatory gesture model [66]. The coarticulation model represents coarticulation by overlapping of raising and falling negative exponential functions, called dominance functions. Baldi has been imple-

[3]http://tcts.fpms.ac.be/synthesis/mbrola.html

mented in 20 languages[4], such as Arabic, called as Badr [77]. RULSYS is a system for rule-based audiovisual speech synthesis and its 3D talking head can be animated based on Parke's model and synchronised with a formant-based auditory speech synthesiser. This synthesiser controls both audio and visual modalities using rules which transform the given input text into 40 parameters of auditory speech and 50 visual parameters. These parameters are then used to improve the in-between keyframe. The modular audiovisual speech synthesiser (MASSY) is a 3D parametric audiovisual speech synthesiser. MASSY was originally implemented for German [35] and has been extended to new languages, such as English [34] and Estonian [75]. Unlike Baldi's work, the facial control parameters are extracted from articulatory motion data of a human speaker by using a Carstens AG 100 electromagnetic articulograph (EMA). From the EMA data, six articulation parameters are used in the model, including the width of the lips, the height of the lower jaw, the height of the lip opening independent from the height of the lower jaw, the retraction of the lower lip, the height of the tongue tip, and the height of the tongue back. The real target values of each facial control parameter were determined from EMA and video recordings. After which, the dominance model (adapted from Cohen and Massaro [16]) of six facial control parameters was generated by combining with the real target values for each viseme.

2.6.4 Sample-based approach

Sample-based synthesis involves the concatenation of pre-recorded speech data that corresponds to speech units. This approach generates animation for an unseen input by selecting appropriate speech units from an existing database. In audio TTS, the general speech data is the speech waveform, while either image or a visual feature representation is the general speech data for visual TTS. From existing work, size and type of speech units can be vary from small to large, for instance phonemes, di-phone or other units (e.g. word, phrase). The major benefit of this approach is the naturalness of the output quality. While, the limitations are that this system

[4]http://mambo.ucsc.edu/psl/international.html

requires a large database of visual speech and the facial output animation will look the same as the actor's face in the training database.

Bregler el al. [10], Video Rewrite, used context dependent triphone as a basic speech unit. The mouth region and background region in each image are extracted from an audiovisual database (depicted in the top row of Figure 2.11). EigenPoints were used to specify the contour of the mouth and jaw in each image. These lip and jaw contours are used to find the distance in adjacent frames regarding to the lip width, lip height, inner lip height and teeth height, referred to join cost function. Phoneme and viseme context distances are used to compute the target cost function between the candidate and target triphone. In the synthesis stage, a given audio speech input is transcribed to a sequence of phonemes using HMMs. After converting phonemes to triphones, the appropriate mouth and jaw images are retrieved from the video modelling database using a combination of target and join cost distance. Finally, the final output video is synthesised by stitching the lips and jaw onto the background image sequences. Their resulting synthesised output frames are shown in the bottom row of Figure 2.11.

This "Video Rewrite" system was developed further by the AT&T Lab Research [21]. They segmented each face image into several facial regions in Figure 2.12 including forehead, eyes, mouth, upper-teeth, and chin with the lower-teeth if visible. Each part learned the target cost between phoneme context dependent and its corresponding face region separately. The final facial animation is generated by simple image overlay.

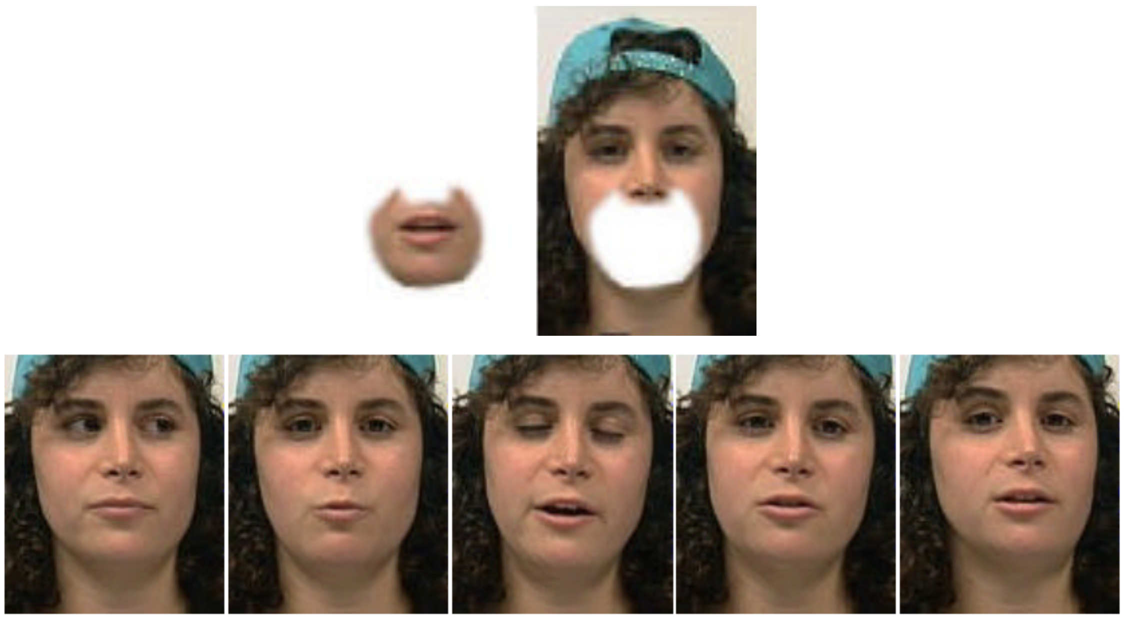


Figure 2.11: An example of the mouth and jaw facial regions for a concatenative visual speech synthesis (top row) and examples of synthesised output frames (bottom row) [10].

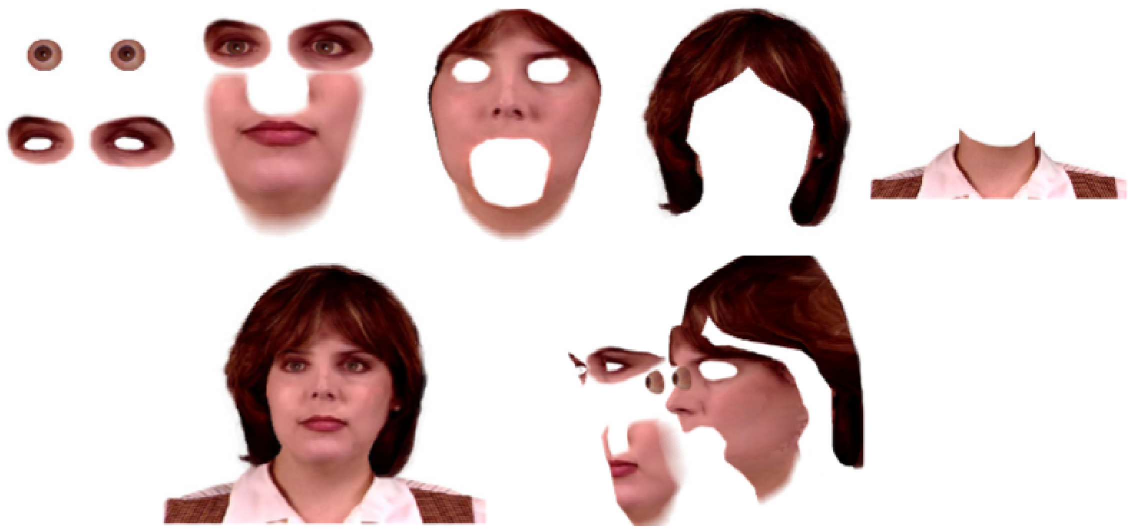


Figure 2.12: The segmentation in different facial regions for a concatenative visual speech synthesis [21].

Instead of using the geometric lip and jaw position as visual features, Theobald [103] used shape and appearance models to generate a photo-realistic facial animation from text. Each image is described by a set of landmark points, called shape parameters, and a set of appearance parameters. For training, a simple lookup table between phonemes and shape and appearance parameters was created. For testing, a given text was converted to a phoneme sequence and its durations, and for each phoneme, the lookup module returns shape and appearance parameters of the closest phoneme and durations in the table. Note that the final facial animation was created by applying a smoothing spline to the set of parameters in each frame, to ensure that the output is smooth. An additional benefit of using visual features instead of raw image is the ability to modify the output, for example Theobald extend their 2D-based visual TTS to 2.5D-based visual TTS [105].

Another important question for visual speech synthesis is how to avoid the lack of audio and visual synchrony. Instead of modelling audio and visual features separately, some methods consider the coherence between the audio and visual streams by joining these two features into single modality (described in Section 2.4.2) and the audio and visual segments jointly selected from the same corpus. A first preliminary work on joint audiovisual synthesis was proposed by Fagel [35]. He selected and concatenated various speech units (chunks) from an audiovisual database in German, beginning with a sequence of phonemes that is then split into all possible length speech units that vary from diphones to the whole utterance. During selection, the maximum length of a chunk was limited to 12 phones. After that, the system searched every chunk from the database and sorted the chunk sequences by joint cost functions. Two costs are considered between two chunks; the audio join cost and visual join cost. The audio join cost is calculated as the percentage difference between the F0 value of the final frame of first chunk and the F0 value of the first frame of the next chunk. The visual join cost is calculated as the percentage difference between the RGB values of the final frame of first chunk and the first frame of next chunk.

Mattheyses et al. [71] also confirmed that mismatches between audio and visual speech can be avoided when selecting the same segment for both modalities from the same units in the database. Their synthesiser is extended from unit-selection audio TTS system and uses a cost function to select the segment sequences from the LIPS2008 audiovisual speech database [106]. The cost function consists of a target cost function and join cost function. In the target cost function, longer units are found based on their symbolic features. Examples of symbolic features are phonetic context, part of speech, lexical stress, and the position in the phrase [62]. The join cost functions indicate how smoothly two segments can be concatenated. Audio and video join costs are used to calculate the cost between two segments. The audio join cost calculates the percentage difference in log F0, energy, and spectrum values between the two sides of join. The visual join cost calculates the percentage difference in shape and appearance parameters of an AAM model, and histogram information between two sides of join. In terms of synchronisation, they aim to find the video join position as close as possible to the audio join position. The results show that there is no significant difference between a separate model and a joint model. In [71], the join cost function was improved by optimal coupling; that either determines the cut-points based on the audio approach or determines the cut-points based on the video approach. Subjective tests show that 12 viewers prefer the audio approach rather than the video approach.

In 2011, Mattheyses et al., [72] improved the target cost function by increasing the number of candidate units. They used visemes as a basic unit of synthesis instead of phonemes. In a many-to-one phoneme-to-viseme mapping, the video data were represented using an active appearance model (AAM). Then, AAM vectors at the middle frame of each phoneme were gathered and the the mean visual representation of each phone calculated. The mean vectors were clustered using hierarchical clustering to determine which phonemes are visually similar to others. Unfortunately, the highest performance was achieved by the phoneme-based synthesiser in both objective tests and subjective tests. There are two major reasons. First, a many-to-one phoneme-to-viseme mapping does not sufficiently describe coarticulation effects. They suggested that a many-to-many phoneme-to-viseme mapping should be used. Secondly, the LIP2008 database does not contain enough data to cover all possible phonetic contexts.

Instead of using phoneme or viseme context distance in target cost function, Wang and Soong [114] compute the target cost by measuring the Euclidean distance between trajectory-guided and candidate PCA vectors. They proposed a hybrid approach between HMM synthesis and unit-selection synthesis which aims to combine the best properties of HMMs and sample-based approach. Generally, this approach used a statistical HMM-based visual speech synthesis predicting visual PCA parameters and then these features are used as trajectory-guided for the sample-based approach in the next step. Then, these trajectory-guided PCA vectors were used to find the best lip sequence from lips candidate from the prepared database in sample-based approach, as shown in Figure 2.13. From their experiments, it clearly see that these two step approaches can be generate the high quality output with the improvement of the output at the boundaries.

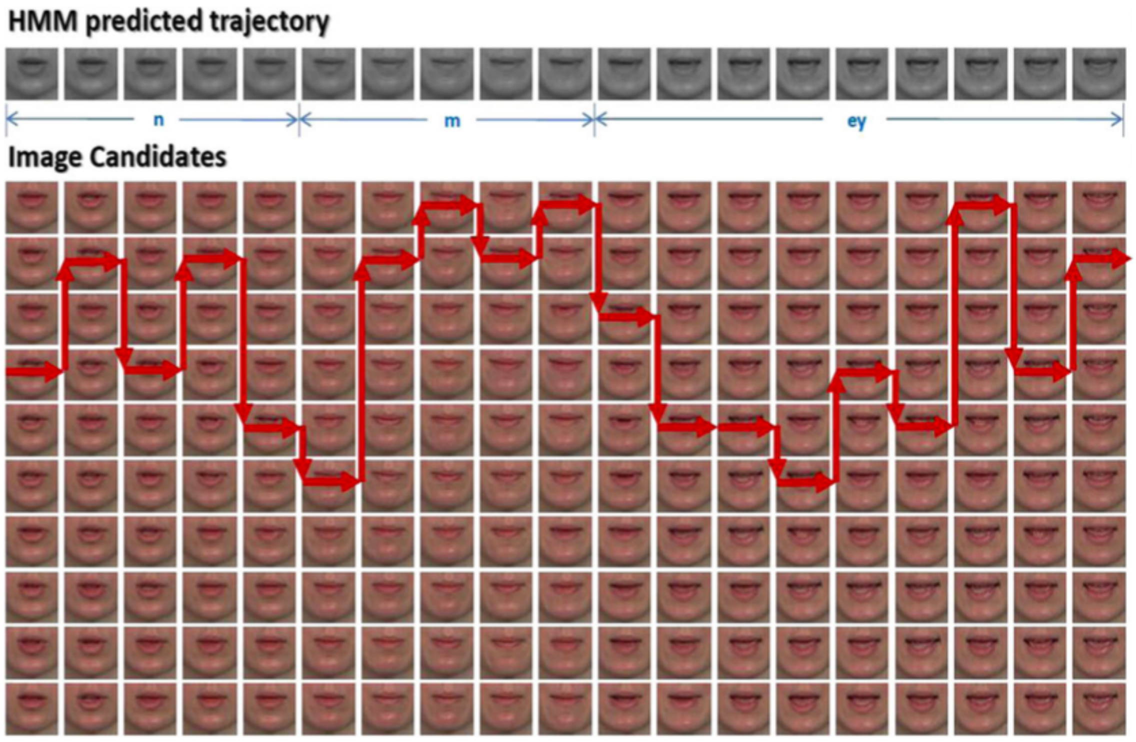


Figure 2.13: A hybrid approach between HMM-based and unit selection approach where the HMM-based is used to guide the search in unit selection database approach. [114].

The size of the corpus has been found to directly affect the naturalness and the choice of speech units. Mattheyses et al. [73] investigated further the effect of phoneme and viseme units on a larger audiovisual database of Dutch speech (approximately 140 minutes). Their viseme units are based on a many-to-many phoneme-to-viseme mapping. An AAM was trained to represent the mouth-region of the video frames in terms of shape and appearance parameters. Then, each phoneme was represented by three vectors of combined AAM parameters at 25%, 50%, and 75% of the duration of the phoneme. Multi-dimensional regression trees were used to build a binary decision-tree based on context clustering. Each node, except leaf nodes, had a context related question, such as R-Voiced? (Is the next phone voiced or not?) and C-visible teeth? (Are teeth visible in the current phone?), for which each node contains a yes or no answer. This leads to the tree-based clustering which must be able to model the audio/visual coarticulation effects with the same phoneme being mapped to different visemes based on its context. The results indicate that the viseme-based approach outperforms the phoneme-based approach when using a large database. Secondly, the many-to-many viseme mapping outperforms a many-to-one viseme mapping in terms of a subjective test using a 5-point comparative MOS scale.

In a recent approach, [98] argue that the phoneme and static viseme units are not suited as units for visual TTS because boundaries of the audio and visual speech should not be placed at the same time. This implies that it does not matter how visemes are defined from neccesary be phonemes (e.g. many-to-many, many-to-one) as they are still not good enough. Hence, new speech units that describe similar lip motions are introduced in [98]. This leads to a new set of boundaries meaning that dynamic viseme boundaries not the same as phoneme/viseme boundaries. Additionally, all visual gestures in each dynamic viseme class represent for the same lip motions. Hence, the median visual gesture is used to animating the speech animation in the case of dynamic visemes are available. One of the drawbacks of this unit is the mapping process of phoneme to dynamic viseme in the case of real setting. They found that a little relationship between phoneme and dynamic viseme leads to incorrect lip motions in a sentence.

2.6.5 Statistical-based

The statistical-based approach is generally called a statistical parametric model because the model does not store the original speech as in sample-based approaches but it learns a model of speech parameters instead. These speech parameters are described by statistical (means and variances of probability density function (PDF)), which capture the distribution of some sets of similar speech units during the training phase. This means this approach can generate an unseen output. The major advantages of this approach is the potential to modify the output speech (e.g. emotions) and the ability to generate the output with the compact model. There are various frameworks in the literature, for instance GMMs, a Gaussian Process Dynamical model (GPDM) for each variable length Markov model (VLMM). Two common approaches are to use hidden Markov models (HMMs) and deep neural network (DNN) approaches.

2.6.5.1 HMM-based

In the last 20 or so years, statistical parametric speech synthesis based on HMMs has become the state of the art for automatic speech recognition (ASR). HMMs have subsequently gained popularity for speech synthesis after the introduction of temporal derivatives as constraints with the maximum likelihood solution in 1995 [110, 111], which produces a more smooth and realistic output.

In 1990 before the emergence of HMM synthesis, Cox and Simons [1990] proposed an early HMM based visual speech synthesiser for producing a talking head over telephone lines (videophone). With the limitation of the telephone bandwidth, this system was designed to transmit a sequence of action units (AUs) of the Facial Action Coding System (FACS). Hence, in training, each image frame is labelled with the AU as visual features and the corresponding audio is represented with quantised Mel frequency cepstral coefficients (MFCCs). Note that, 16 AUs were used to describe the variation of mouth movements in the database. Afterwards, an ergodic HMM topology was used to learn the mapping between mouth shapes from AUs and audio speech from MFCCs, meaning that each HMM state represents one of the 16 AU codes. For simplicity, Figure 2.14 shows an example of an 8-state ergodic HMM topology. In synthesis, given an audio input, a sequence of image codes are generated using the Viterbi algorithm. Then, mouth shape sequences are produced using a simple lookup table. Two problems with this approach are a discontinuous speech output (c.f. the smoothing problem) and the possibility of the wrong state being selected.

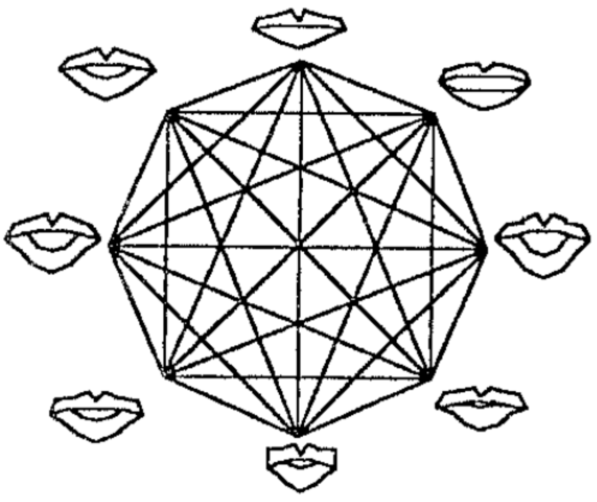


Figure 2.14: An 8-state ergodic HMM topology to represent mouthshapes [21].

To overcome the smoothing problem, the framework for the HMM-based visual-only speech synthesiser was adapted from the framework for a HMM-based audio text-to-speech synthesis system, which replaces audio feature vectors with visual feature vectors [68]. In the training phase, audio and visual features are extracted from an audiovisual speech corpus consisting of 216 phonetically balanced Japanese words. MFCCs and their dynamic features are used as speech features and inner lip position parameters (shown in Figure 2.15(a)) and their dynamic features are used as visual features. Next, 3-state HMMs with a left-to-right, no-skip topology with a single diagonal Gaussian output distribution are used to build syllable models. Note that Japanese syllables are categorised into 42 visual categories. In the synthesis phase, the text processing of the text-to-speech synthesis system is used to convert the given text into a phonetic transcription. Then a sentence HMM is constructed by concatenating the syllable HMMs corresponding to the sequence of phonemes. Finally, a sequence of visual speech parameters is generated from the sentence HMM and converted into lip animations. Results show that the trajectories of the synthetic visual speech generated with dynamic features are smoother and closer to real parameters in terms of both objective and subjective tests than visual speech generated without using dynamic features.

A speech-driven approach was proposed by Tamura et al. [96] and used ASR to extract a sequence of syllables from a given audio signal. These syllable sequence are then used as an input for the Masuko's system (described above). See Figure 2.15(b) for an example of final lip animation in both text-drive and speech-driven synthesis. Based on their example videos, we can observe that the mouth movement is not in synchrony with audio. This unacceptable result may originate from many factors, such as the syllable units not being suitable for visual TTS, the over smoothing problem because of the HMM training criterion, or the the database is not large enough.

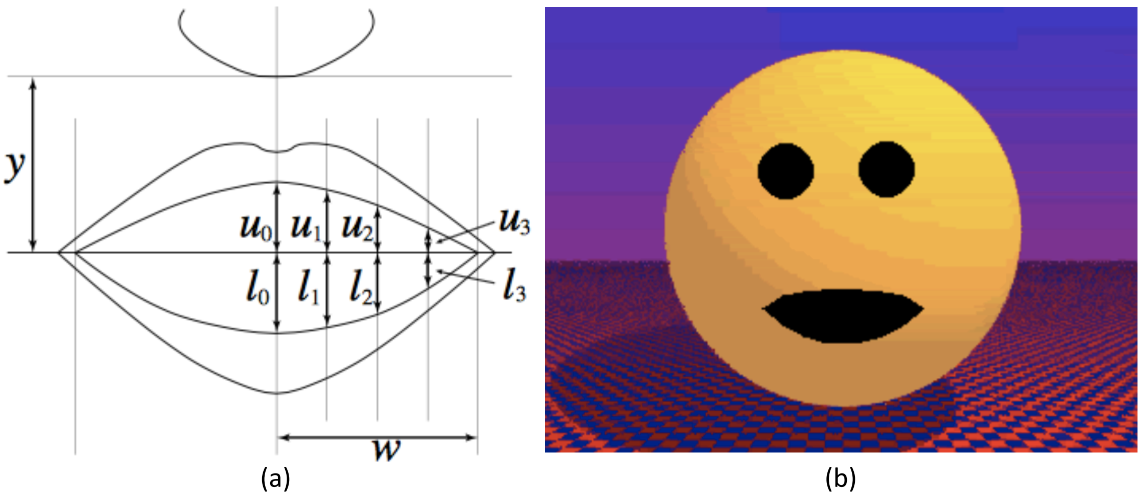


Figure 2.15: (a) Inner lip position parameters and (b) an example lip motion of a single frame extracted from [96].

Another similar HMM system based on maximum likelihood (ML) criterion using different geometric features was proposed by Hofer et al. [56]. With the geometric features, they used two lip distances including the distance between the left and right corner of the mouth and the distance between upper and lower lip to control shape of the lips. Note that, these four landmarks were recorded from tracked markers during captured database. Their system also consists of two steps approach as Tamura et al. [96]; recognition and synthesis step. In the recognition step, a sequence of viseme were produced and used as the basic speech units. For the synthesis step, a sequence of viseme context dependent from the previous step was used to predict two smooth trajectories. Finally, these two trajectories were used to drive the spread and open of the lip shape. Their findings also found that viseme sets can make the difference in terms of performance, hence the selected viseme sets is essential. Additionally, they also confirmed that viseme units gave the highest objective scores comparing with phoneme units. Conversely, [101] found that objective scores and inspection AAM trajectories on HMM-based system using phoneme units and viseme units were nearly the same in the case of text-driven synthesis. This work also examined suitable speech units (e.g. phonemes, visemes and dynamic visemes units) for visual TTS and found that dynamic visemes units (groups of similar speech movements) was the best speech unit for visual TTS. More details are given in Chapter 5.

HMM synthesis is able to solve the smoothing problem by introducing dynamic features (described above), but the maximum likelihood (ML) criterion in training tend to generate over-smoothed mouth movements. Wang et al. [115] proposed the use of a minimum generated trajectory error (MGE) method after initialising the model with ML. The aim of MGE is to refine the means of the visual HMM using the probabilistic descent (PD) algorithm. An objective evaluation shows that MGE is able to improve visual speech trajectories which led to better mean opinion scores in subjective test compared with ML-based.

Wang et al. [116] extended their system from 2D to 3D photo-realistic facial animation. In fact, this is called 2.5D because there is no depth information. Their 2D-to-3D conversion method allows the system to get the benefits from both 2D and 3D based models. In 2D-based the teeth and tongue can be seen when they are available, but it is difficult to modify the expression. Conversely, it is easier to deform one expression to another in a 3D-based system, but it is difficult to render some details on inner mouth area (e.g. teeth and tongue). In data preparation, two types of visual features are extracted from each frontal image: i) 3D face geometry and ii) 2D face texture. As their audiovisual database does not have the 3D geometry information, they reconstruct this feature from 2D face shape alignment by applying the method in [58]. In training, multistream HMM synthesis with MGE is used to train the 2D-to-3D facial model. The first stream is a 2D face texture and the second stream is a 3D face geometry. In testing, the phone transcription is recognised from a given audio input. Then, the context-dependent HMMs are used to predict 2D face texture and 3D face geometry. Subsequently, the 2D face texture is mapped

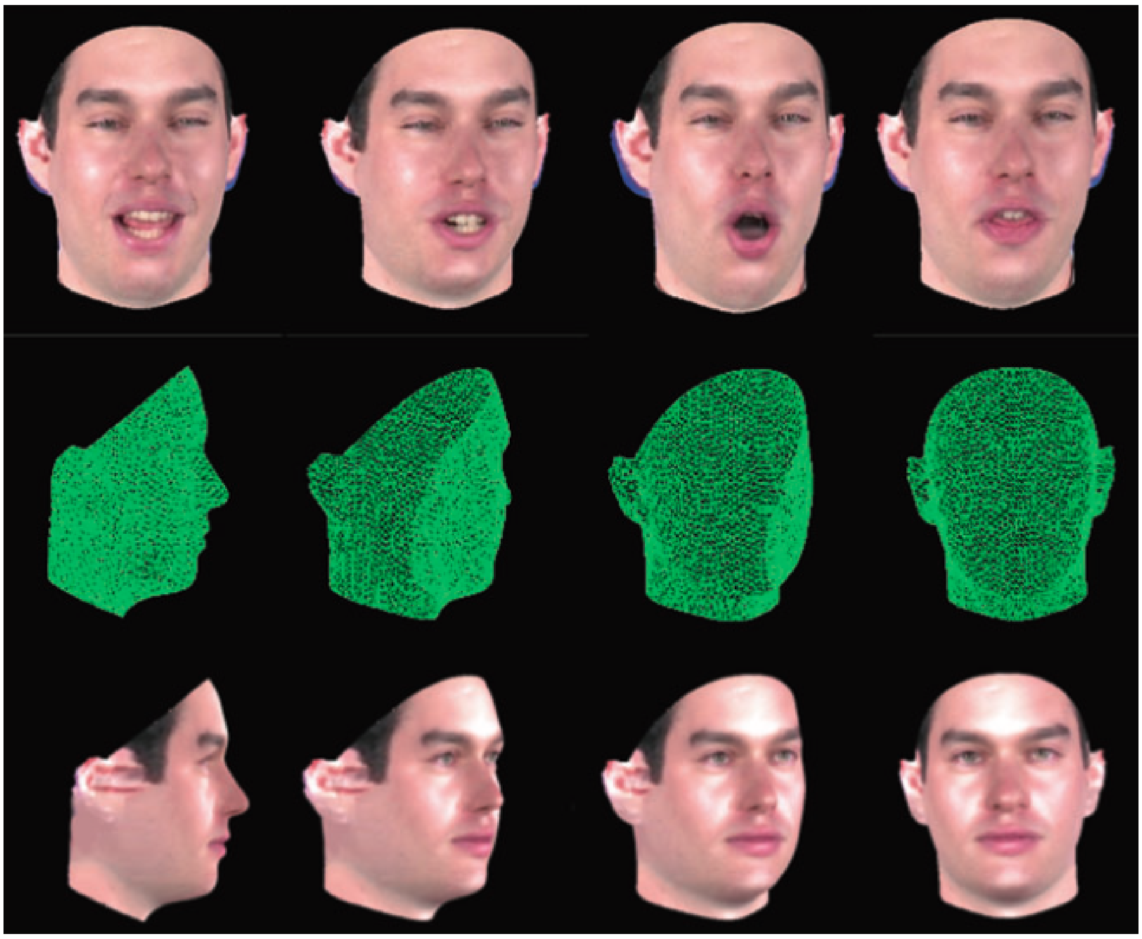


Figure 2.16: Examples of 2.5D face model with appearance (top). Examples of 2.5D face model as wire-frame (middle). Example frames with different angles (bottom). (modified from [116])

onto the 3D face geometry forming a 3D face model, as shown in Figure 2.16.

Another visual features, namely articulatory features, provide various kinds of visible (e.g. lips) and non-visible (e.g. tongue) articulator information. There are several technologies have been used to capture the vocal tract shape and movement. ElectroMagnetic articulography (EMA) and motion capture, for example, are the process of recording the movements of objects interested (e.g. mouth and body movements) via sensors or markers that attached on objects. This recorded information provides highly accurate data with three degree of freedom for each sensor. These data can be used to animate 3D facial animation directly from the input without 2D-to-3D conversion methods. For instance [122] used Poser Pro software[5] to animate 3D virtual avatar for speech-driven head and lip motions. Moreover, one of interesting findings found that the use of HMM synthesis with these articulatory features are effective than prosodic features for the head motion synthesis task because of the highly accurate of the features from EMA.

Based on the single phase approach discussed previously, Schabus et al. [89] proposed an audiovisual speech synthesis strategy based on HMMs. They made the assumption that both the audio and visual signals are the result of the underlying articulation process, and should be treated as one modality. Based on this they were able to ignore audio-visual synchronisation because the joint audiovisual model aims for a maximal level of coherence between these two modalities. Their results also confirm that joint modelling outperforms separate audio and visual modelling. One of their interesting results concerns visual features. Their results showed that visual features (6-classes of viseme questions) are useful for visual modelling because these appear often in the context-based clustering tree for the visual stream. For example, they also pointed out two limitations of the synthesiser. First, the output quality of the synthesiser, especially for the auditory modality, is not acceptable. They found that some phonemes are generated incorrectly and belong to different phoneme classes. The main reason is the limited size of the database because only 223 utterances (approximately 11 minutes) of German speech was used in training the model. Secondly, the audiovisual time-lag problem was found in the separated models. They showed that the phoneme boundaries are different to the viseme boundaries. This is the main reason why the animation results of separated models are unnatural compared with that of joint audiovisual model.

To overcome time-lag problems, Govokhina et al. [47] proposed a phasing model to predict the time lag between acoustic boundaries and gestural boundaries. This idea is similar to the time lag between note timing and voice timing for an HMM-based singing voice synthesis system [86], as shown in Figure 2.17. They proposed to add decision tree context clustering dependent time-lag models. This time-lag

[5]http://poser.smithmicro.com

modelling is used to predict time-lags and state-durations after predicting the duration of each musical note from the given musical score input. It follows that the voice timing can be determined according to the note timing and can be obtained automatically by forced alignment.

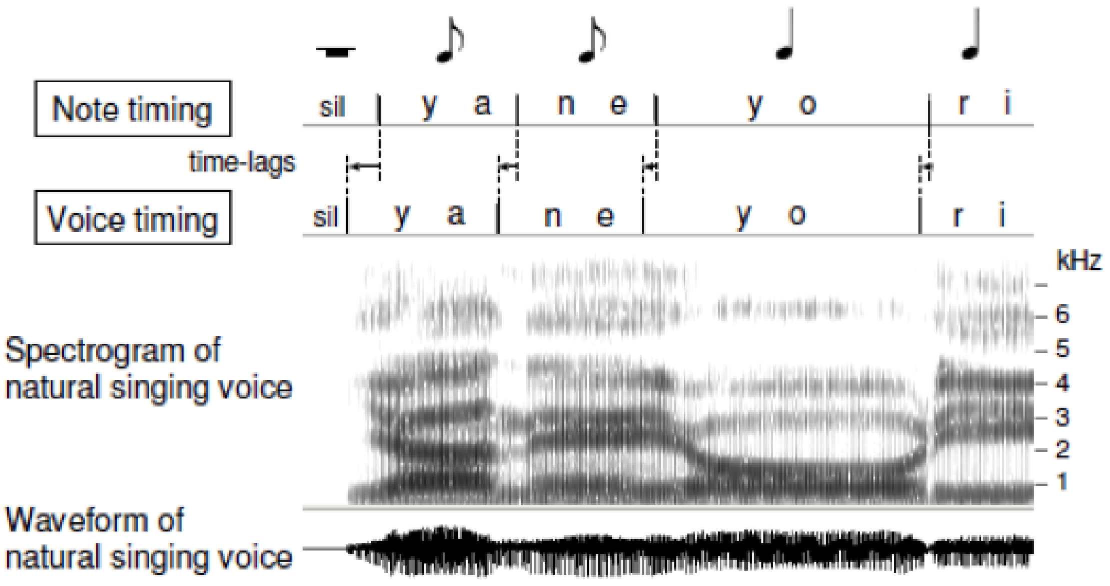


Figure 2.17: An example of time-lag in singing voice speech synthesis [86].

To apply the HMM synthesis to expressive visual speech data, Cluster Adaptive Training (CAT) was used to model six different speech expressions consisting neutral, angry, happy, tender, sad and fearful [5]. CAT is an extension of HMM synthesis and each expression has its own decision tree. Figure 2.18 shows 2.5D facial image results for the six different expressions. In this system, a new expression can formed by any combination of input expressive weights.

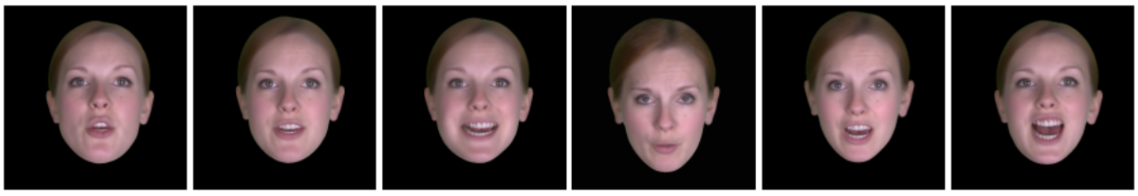


Figure 2.18: Examples of 2.5D facial image results in six different expressions[5].

2.6.5.2 DNN-based

HMMs have been state of the art in (visual) speech synthesis for the past decade and typically employ decision tree clustered context-dependent models, although a drawback has been an over smoothed output [31]. For example, a similar problem exits in audio speech synthesis [128] which can lead to a muffled sound being produced. The equipvalent in visual speech synthesis from HMMs tends to produce under-articulated lip movements [101]. Deep neural network (DNN) approaches have more recently been proposed to address these limitations and are able to learn a better model using multiple levels of non-linear activation functions such as the use of multi-layer perceptrons with many hidden layers and numbers of units [6].

Taylor et al. [99] developed a DNN based visual synthesiser which learns a mapping function from audio to visual speech. A large audiovisual database of 2,542 sentences was used in this work. The audio data was converted to MFCC acoustic features and visual image represented by 104 AAM parameters. Due to coarticulation effects and the smoothing problem, both input and output features were represented with the sliding window approach adopted by [61]. Their findings found that 330ms (5 frames preceding and 5 frames ahead) was the best width for the input window and is long enough to avoid the coarticulation effects. Additionally, an output window width of 165 ms (2 frames preceding and 2 frames ahead) was best for visual the output and is not too small (smoothing problem) or too large (over-smooth problem). Both objective and subjective results showed that the deep neural network model outperformed a baseline HMM approach.

Using the same audiovisual database, [102] developed a DNN based visual synthesiser which learnt the mapping function from text to visual speech. This work explored three aspects to improve visual speech synthesis; i) speech units, ii) contextual features and iii) learning algorithm. For the speech units, the work explored two speech units; phoneme and dynamic viseme units. They found that both units gave similar results meaning that each unit can provide different unique information to the model. This is confirmed by the improvement in performance after combining these two units. With contextual linguistic features, an analysis of various information from the frame level to the utterance level found that frame level context is very important for reducing discontinuities in the output. In terms of the learning algorithm, the work found that a DNN-based system outperforms a HMM-based system. (more details can be found in Chapter 6)

Recurrent neural networks (RNNs) are another type of neural network designed to process sequential data, for example, text, speech, image, etc. These were first introduced in [83] and have been shown to outperform feedforward network in a range of applications, such as machine translation, automatic speech recognition. Fan et al. [36] also applied bidirectional long short term memory (LSTM) networks to visual TTS, where the LSTM is within an RNN architecture. A given text input was converted to a sequence of tri-phone labels, but suggested that rich information (e.g. stress, part-of-speech) should be used if a large database is available. A given audio input was also used to obtain phoneme state levels as this feature was found to allow a better prediction to be made. The output animation focuses on the lower face and represented by AAM features. Preference tests showed clearly that the bidirectional-LSTM speech animation is significantly better showed that the baseline HMM-based.

Some groups have also built language and speaker independent visual TTS in different kinds of frameworks [100, 129]. [100] attempt to learn a non-linear mapping from a sequence of phoneme labels to mouth movements using deep neural networks. Both phoneme input labels and AAM output features were applied using a sliding window approach (described in [99]). Their results suggested that a feed forward approach significantly outperforms other learning approach including dynamic visemes-based [98], HMM-based [101], LSTM-based [36] and decision tree regression [61]. Their approach was trained with a single speaker, but is able to generate different characters of speech animation, as shown in Figure 2.19. It is also able to predict the AAM visual output from any content and speaking style as long as automatic speech recognition is available. Note that the quality of the speech animation depends on the accuracy of the recognition system. The main limitation is that the lip animation tends to be under articulated and require post processing to deal with this problem (scaling up the signal for example).

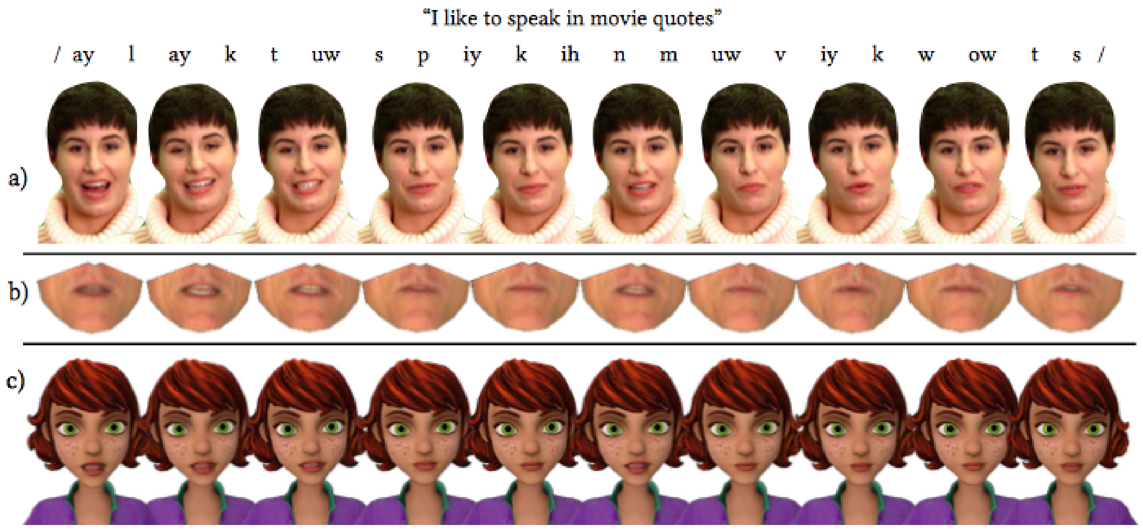

"I like to speak in movie quotes"
/ ay l ay k t uw s p iy k ih n m uw v iy k w ow t s /
a)
b)
c)

Figure 2.19: Example of three different characters of speech animation (a) the original video (b) the predicted lip animation from AAM (c) the facial rigs that retargated from the shape model [100].

Sato et al. [88] proposed a hybrid approach between HMM synthesis and DNN synthesis. The goals of this paper are developing a small footprint synthesiser, using auto labelling of visual features, generating photo-realistic image for the entire face. From these goals, the two commons statistical parametric approaches were used instead of sample-based approach as we know that sample-based approach is costly in terms of computational and storage. The main reason of using hybrid HMM-DNN approach because they expect to get the best performance from both properties of HMMs and DNNs. In the first step, they used HMMs synthesis to predict visual features from a sequence of triphone context dependent. After that, these visual features are used to predict output pixel images from DNN models. It can be seen that the dimension of output is very high and takes up a huge of CPU time and memory in the case of using HMM-based. That is the main reason they attempt to apply DNN-based in the second step. Note that, 16 action units (AUs) were extracted automatically using Microsoft Kinect v2 and used as visual features in this work. Their preference test showed that viewers significantly preferred the hybrid approach more than conventional HMM-based using PCA visual features [87]. However, the image quality of the proposed technique still need to improve because they can spot some unclear image as the same as found in PCA approach.

Chapter 3

Technical background

3.1 Introduction

This chapter aims to overview three most popular statistical parametric approaches for both audio and visual speech synthesis including hidden Markov models (HMMs), deep neural networks (DNNs) and recurrent nerual network long short term memory (RNN-LSTM). This starts with an general overview of the Gaussian mixture model-hidden Markov model (GMM-HMM). It then follows with an introduction of a variety of HMMs from a synthesis point of view, as known as the hidden semi-Markov model (HSMM). After that, an overview of feedforward networks is described. Then, we explain how feedforward networks learn their representations in the network using the backpropagation algorithm. Finally, an overview of recurrent neural networks (RNNs) in terms of structure, architecture, training methods, and long short term memory (LSTM) are described.

3.2 Overview of HMM synthesis

This section aims to provide a brief introduction of HMM synthesis, of which more details can be found in [110, 111, 120, 121]. This starts by introducing the concept of the use of a Gaussian mixture model (GMM) to explain a data distribution. After that, the HMM is used to overcome the limitation of modelling temporal features in a GMM. It also shows how the basic structure of HMMs use a transition probabilities matrix to model duration in each HMM state, This, however, does not have the ability to control phoneme durations from explicit parameters. To overcome this problem, new models have been proposed, called hidden semi-Markov models (HSMM), which use explicit duration modelling to model the duration of the HMM states instead of the transition probability matrix. Finally, we describe how HMM synthesis treats a series of step outputs using the parameter generation algorithm with dynamic features (delta and delta-delta) to produce a more smooth output.

3.2.1 Gaussian mixture model (GMM)

A Gaussian mixture model can be used to represent random variables when its distribution or the probability density function (PDF) is unknown. A Gaussian distribution is normal and has the shape of a bell curve, as shown in Figure 3.1. This Figure presents an example of a standard normal distribution with a mean, μ, of zero and standard deviation, σ, of one.

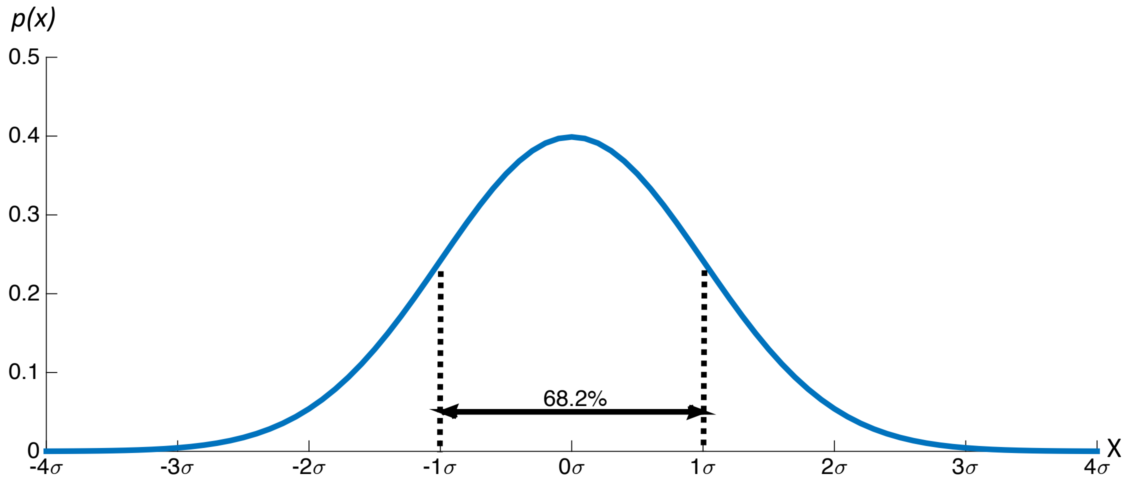


Figure 3.1: An example of standard normal distribution with zero mean and unit variance.

Both scalar and vector random variables are able to be represented with a single or univariate Gaussian. Firstly, the PDF of a scalar random variable is defined by

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \tag{3.1}$$

where $p(x)$ denotes the PDF of the random variable x with mean, μ, and variance, σ^2. Note that, the standard deviation is denoted σ on the square root of the variance. An equivalent notation can be specified as

$$X \sim \mathcal{N}(\mu,\, \sigma^2). \tag{3.2}$$

Secondly, the PDF of random vector, known as multivariate Gaussian distribution, is specified by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\boldsymbol{T}} \boldsymbol{\Sigma}^{\boldsymbol{-1}} (\boldsymbol{x}-\boldsymbol{\mu})\}, \tag{3.3}$$

where $p(\mathbf{x})$ denotes the PDF of the random variable $\mathbf{x}$ with mean vector, $\boldsymbol{\mu}$, and covariance matrix, $\boldsymbol{\Sigma}$. An equivalent notation can be specified as

$$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu},\, \boldsymbol{\Sigma}), \tag{3.4}$$

As mentioned earlier, a single Gaussian can model only a normal distribution. However, the distribution of much real-world data takes on different types of distributions and is far more complex than a normal distribution. Hence, the Gaussian mixture model (GMM) is introduced to represent multiple Gaussian distributions as depicted in Figure 3.2. The concept of a GMM is that it represents a random variable with a set of weighted normal distribution components. Then, the linear combination is applied to all the components as follows;

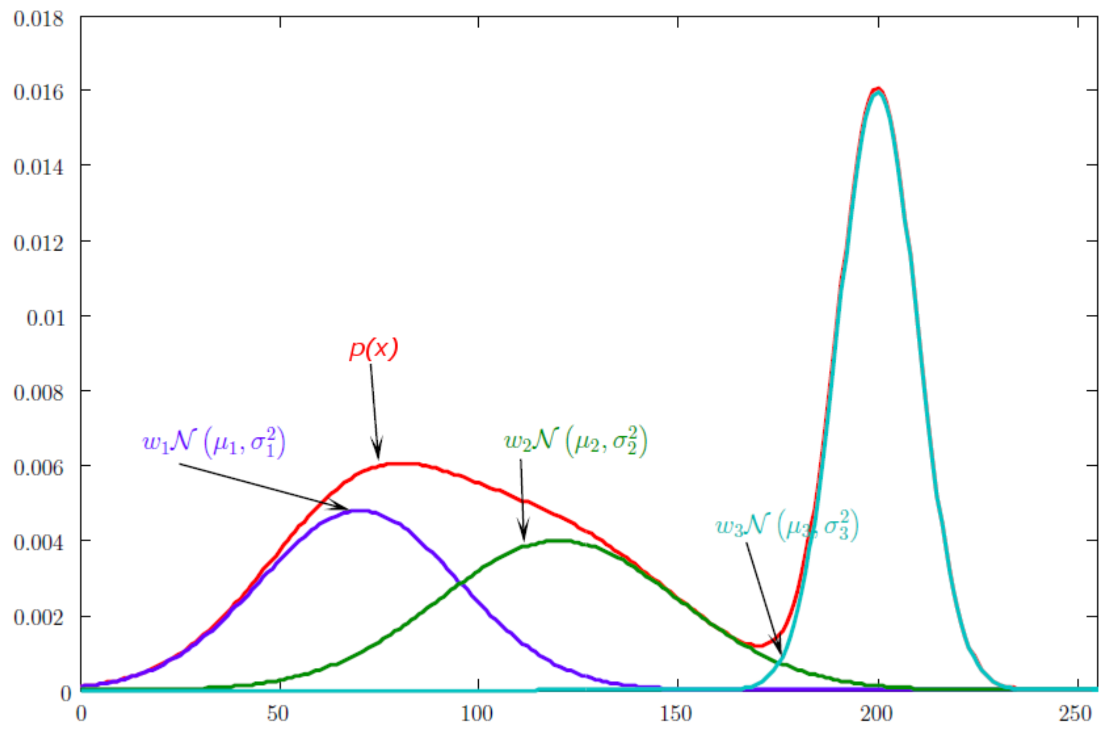


Figure 3.2: An example of the linear combination of a set of Gaussian models, as called Gaussian mixture model. (modified from [2])

$$p(x) = \sum_{m=1}^{M} w_m \mathcal{N}(x; \mu_m, \sigma_m^2), \tag{3.5}$$

where M is the number of mixture components, w_m, μ_m and σ_m denote the mixture weights, means, and variances in each Gaussian, respectively. Note that, w_m is used to ensure that the total probability is equal 1. In the literature, the use of Gaussian mixture models is the key factor of the success of various speech applications such as speaker recognition [81]. However, some speech applications, for example speech synthesis and speech recognition systems, do not directly use a GMM to represent the acoustic features. The main reason is that the GMM shows a good fit when the time information is excluded but it is not a good model when temporal features or time series data are included as speech is highly time-varying. That is the main reason to use an HMM instead of a GMM. More details are given in next Section.

3.2.2 Hidden Markov model (HMM)

The hidden Markov model (HMM) is an effective framework that can model time-varying features, such as speech, part-of-speech tagging, and handwriting. The HMM is an extension of a Markov chain which is a stochastic process and uses transition probabilities, a_{ij}, to describe a change of state i to state j in the system. Within a state the signal is assumed to be stationary, but the linkage of states allows a time-varying signal to be modeled. Generally, an ergodic HMM topology allows transition from any state to any other state. However, to model speech, a specific topology is used, namely a left-to-right HMM as shown in Figure 3.3, which is constrained to move from left to right, or stay in the same state.

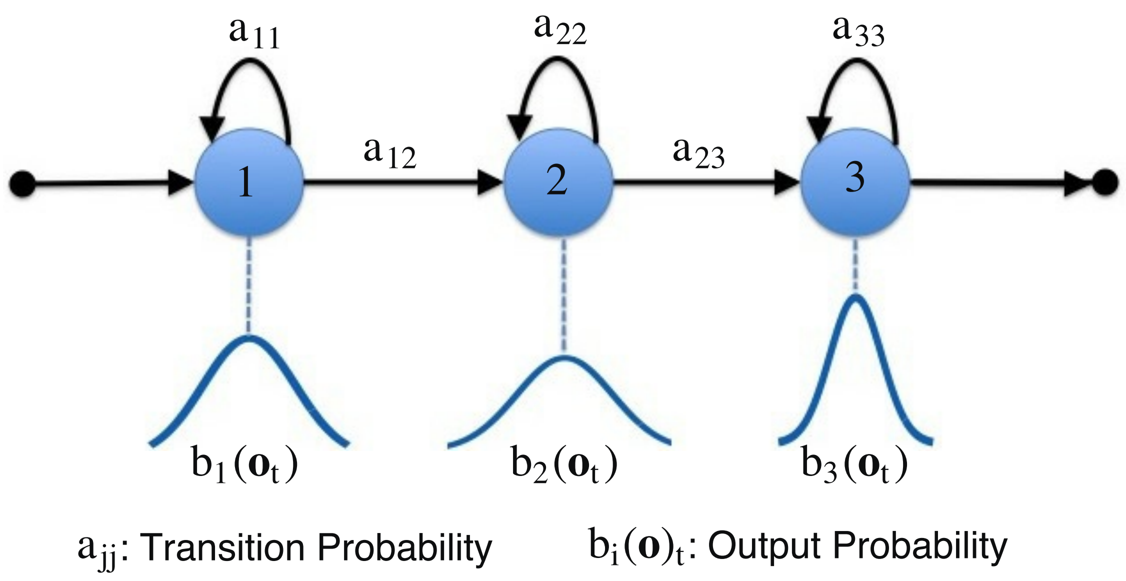


Figure 3.3: An HMM topology of a left-to-right three state with no skip and a single Gaussian output probability.

Moreover, each state of a chain, or HMM state, is characterised by the observation data using a univariate/multivariate Gaussian mixture model, $b_i(\mathbf{o}_t)$, (as detailed in Section 3.2.1). Clearly, an HMM is able to overcome the GMM's limitation to model the temporally changing features. This is somewhat true because a GMM can be referred as a single state HMM. Therefore, an increasing number of states can be thought of as increasing the number of GMMs in the state emission PDFs which leads to a better model.

From this left-to-right HMM topology, the duration probability of i-th state, $p_i(d_i)$, is controlled by the self-transition probabilities in each HMM state for d times, which can be considered as implicit state duration modelling, as follows:

$$p_i(d_i) = (a_{ii})^{d_i - 1}(1 - a_{ii}) \tag{3.6}$$

where d_i is the number of durations (frames) in i-th state. However, the distribution of this implicit modelling becomes an exponential distribution, as shown in Figure 3.4. This concept, however, works well for speech recognition but it is not appropriate for synthesis point of view because the distribution of phone durations is generally normal distribution. Moreover, in the case of maximisation problem, the duration in each state becomes one, $d_i = 1$. In synthesis applications, therefore, it is preferable to control the duration in each state with an explicit model, which is Gaussian distribution for example.

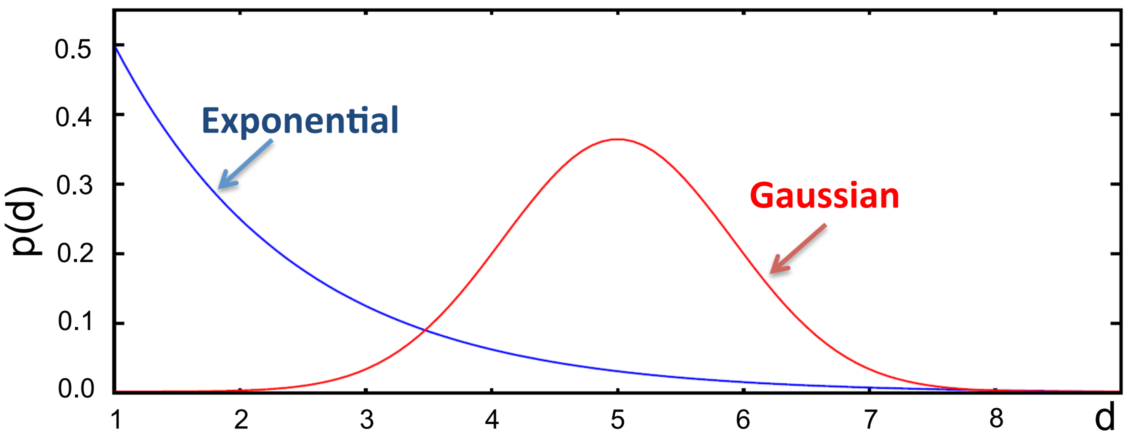


Figure 3.4: The exponential distribution of the implicit duration modelling and the Gaussian distribution of the explicit duration modelling.

Another significant problem is that HMMs with the maximum-likelihood criterion generate step-wise outputs which are produced a sequence of flat-line from their mean values in each state. The next two following subsection overcome these two problems, firstly, the incorporation of explicit duration modelling within HMMs was proposed by [120, 127], also known as hidden semi-Markov models (more details in Section 3.2.3). While, secondly, the use of temporal derivative constraints with the maximum likelihood solution was proposed by [110], called parameter generation algorithm (described in Section 3.2.4).

3.2.3 Hidden semi-Markov model (HSMM)

In a conventional HMM, the number of frames in each state is determined by the transition probabilities, also known as implicit state duration modelling. However, the general concept of the implicit duration modelling is not suitable for a synthesis point of view. For example, the exponential distribution from the implicit models is not appropriate for modelling the duration of phonemes as they are generally normal distribution. Hence, in synthesis point of view, [120, 127] introduced explicit duration modelling into the HMM framework which is known as the hidden semi-Markov model (HSMM). In this model, the transition probabilities are replaced by a model of explicit state duration probabilities, $p_i(d)$, which is usually modelled with Gaussian distributions, as illustrated in Figure 3.5.

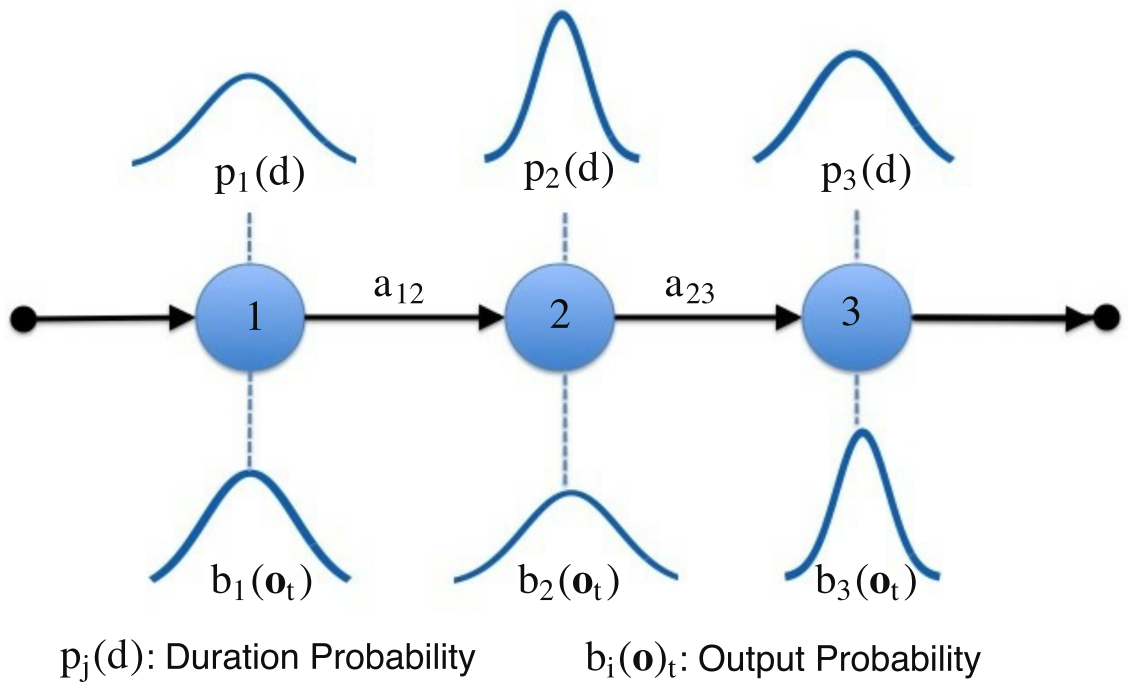


Figure 3.5: An HSMM topology of a left-to-right three state with no skip, a single Gaussian duration probability and output probability.

In this HSMM topology, an explicit state duration is usually modeled by the duration PDFs, $p_i(d)$, such as a single univariate Gaussian distributions specified by mean μ_i^D and variance σ_i^{D2} as:

$$p_i(d_i) = \mathcal{N}(\mu_i^D, \sigma_i^{D2}), \tag{3.7}$$

where d_i denotes the duration in state i which is equal the mean of Gaussians in the case of the maximum solution problem.

$$d_i = \mu_i^D. \tag{3.8}$$

Note that, there were many approaches proposed the improvement of the accuracy of duration prediction. For example, the use of decision tree for unseen contexts, a combination of HMM and MLP, and so on.

While a state output can be modeled by the output PDFs, $b_i(\mathbf{o}_t)$, such as single multivariate Gaussian distributions specified by mean vector $\boldsymbol{\mu_i}$ and diagonal covariance matrix $\boldsymbol{\Sigma_i}$ as:

$$b_i(\mathbf{o}) = \mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}), \tag{3.9}$$

where $\mathbf{o}$ is N-dimensional of observation vector or 30-AAM output feature vector in present work.

3.2.4 Maximum likelihood parameter generation (MLPG)

A major drawback of HMMs and HSMMs is the stepwise output because this is based on a maximum likelihood criterion and becomes a sequence of mean vectors, as shown in Figure 3.6. This is not the output we desire because natural speech is smooth. So, [110] proposed the smoothing post processing module which uses the relationships between static feature (e.g. AAM parameters) and dynamic features (e.g., delta, delta-delta of AAMs) to generate a smoother output, namely the maximum likelihood parameter generation algorithm.

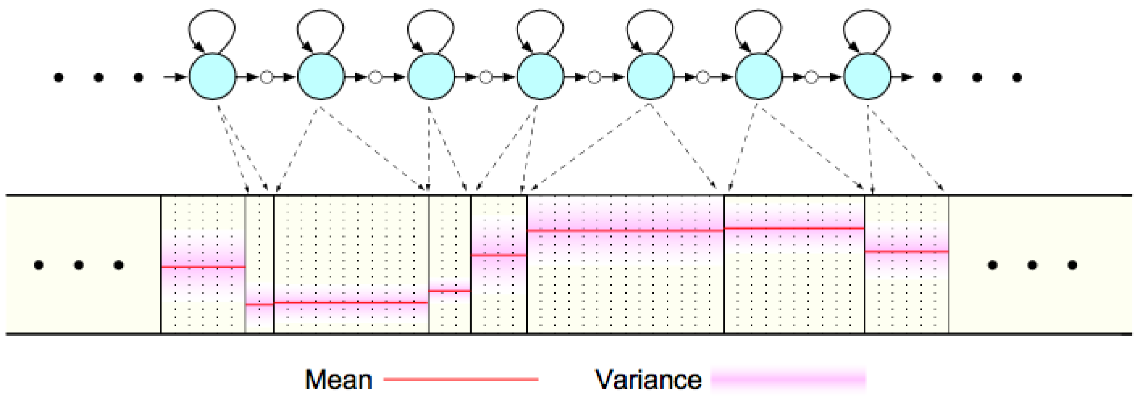


Figure 3.6: Parameter generation with stepwise issue from HSMM [109].

Dynamic features are a simple and effective method used in various speech applications, to model temporal changes in the signal. For example, [40] was first proposed to append dynamic speech spectrum onto static speech spectrum for the propose of improving automatic speech recognition accuracy. In the case of synthesis, [110] proposed the inclusion of dynamic features into the observation vector which are captured from the rate of change of its adjacent frames. A feature vector output at frame t, $\mathbf{o}_t$, comprises the N-dimensional static vector, $\mathbf{c}_t$ (e.g., AAM parameters), and dynamic feature vectors, $\Delta\mathbf{c}_t$, $\Delta^2\mathbf{c}_t$ (e.g., delta and delta-delta AAM parameters, respectively), as follows:

$$\mathbf{o}_t = [\mathbf{c}_t, \Delta\mathbf{c}_t, \Delta^2\mathbf{c}_t], \tag{3.10}$$

$$\mathbf{O} = [\boldsymbol{o_1}, \ldots, \boldsymbol{o_t}, \ldots, \boldsymbol{o_T}], \tag{3.11}$$

$$\mathbf{c}_t = [c_t^1, c_t^2, \ldots, c_t^N], \tag{3.12}$$

where T is the length of the sequence (e.g., number of frames), t is the frame index, and N is the length of the static vector (e.g., number of AAM parameters). The delta, $\Delta\mathbf{c}_t$, and delta-delta parameter vector, $\Delta^2\mathbf{c}_t$, can be calculated as a weighted sum over a left and right window of size 2L + 1 frames:

$$\Delta\mathbf{c}_t^i = \sum_{\tau=-L}^{L} \mathbf{w}^{(1)}(\tau)\mathbf{c}_{t+\tau}^i, \tag{3.13}$$

$$\Delta^2\mathbf{c}_t^i = \sum_{\tau=-L}^{L} \mathbf{w}^{(2)}(\tau)\mathbf{c}_{t+\tau}^i, \tag{3.14}$$

where $-L$ is the width of the left window and L is the width of the right window. With the configurations of standard HMM synthesisis, L is set to 1 which will produce a 3-frame window for computing both delta and delta-delta features. $\mathbf{w}^{(1)}(\tau)$ and $\mathbf{w}^{(2)}(\tau)$ are the window coefficients used to compute delta and delta-delta parameters respectively, can be calculated as:

$$w^0(i) = 1 \qquad for\, i = -1, 0, 1 \tag{3.15}$$

$$w^1(i) = \frac{i}{2} \qquad for\, i = -1, 0, 1 \tag{3.16}$$

$$w^2(i) = 3i^2 - 2 \qquad for\, i = -1, 0, 1 \tag{3.17}$$

The relationship between static feature vector sequence, $\mathbf{c}$, the window properties of the adjacent frames, $\mathbf{W}$, and output feature vector sequence, $\mathbf{O}$, can be arranged in a matrix form as follows, $\mathbf{O} = \mathbf{Wc}$:

$$\begin{bmatrix} c_1 \\ \triangle c_1 \\ \triangle^2 c_1 \\ c_2 \\ \triangle c_2 \\ \triangle^2 c_2 \\ \vdots \\ c_T \\ \triangle c_T \\ \triangle^2 c_T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0.5 & 0 & 0 & \cdots & \cdots & 0 \\ 2 & -1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & \cdots & 0 \\ -0.5 & 0 & 0.5 & 0 & \cdots & \cdots & 0 \\ 1 & 2 & -1 & 0 & \cdots & \cdots & 0 \\ & & \ddots & & & & \\ 0 & \cdots & \cdots & 0 & 0 & 0 & 1 \\ 0 & \cdots & \cdots & 0 & 0 & -0.5 & 0 \\ 0 & \cdots & \cdots & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_T \end{bmatrix} \tag{3.18}$$

As mentioned, the generation of the output parameter sequence becomes a sequence of mean vector of the Gaussian distribution when the HMMs are trained without the conditions (3.13) and (3.14).

$$\hat{\mathbf{O}} = argmax\{\mathcal{N}(\mathbf{O}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\} \tag{3.19}$$

On the other hand, the use of these conditions to form dynamic features and train the models is able to overcome the problem of step-wise output and produce a curved or smoothed trajectory [110]. Equations (3.13) and (3.14) show that $\mathbf{O}$ is a linear transform of **c**, therefore, Equation (3.19) can be rewritten with respect to **c** as

$$\hat{\mathbf{c}} = argmax\{\mathcal{N}(\mathbf{Wc}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\} \tag{3.20}$$

By setting the derivative of log normal distribution with respect to **c** under the conditions (3.13) and (3.14) equals zero, as

$$\frac{\partial log\mathcal{N}(\mathbf{Wc}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \mathbf{c}} = 0 \tag{3.21}$$

We can obtain a set of linear equations to determine c as;

$$\hat{\mathbf{c}} = (\mathbf{W}^{\boldsymbol{T}}\boldsymbol{\Sigma}^{-1}\mathbf{W})^{-1}(\mathbf{W}^{\boldsymbol{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}) \tag{3.22}$$

The illustration in Figure 3.7 also confirmed that MLPG with dynamic features is able to overcome the problem of step-wise output and produce a curved or smoothed trajectory, **c**, reflecting both means and covariance as derived in Equation (3.22). More details can be found in [110].

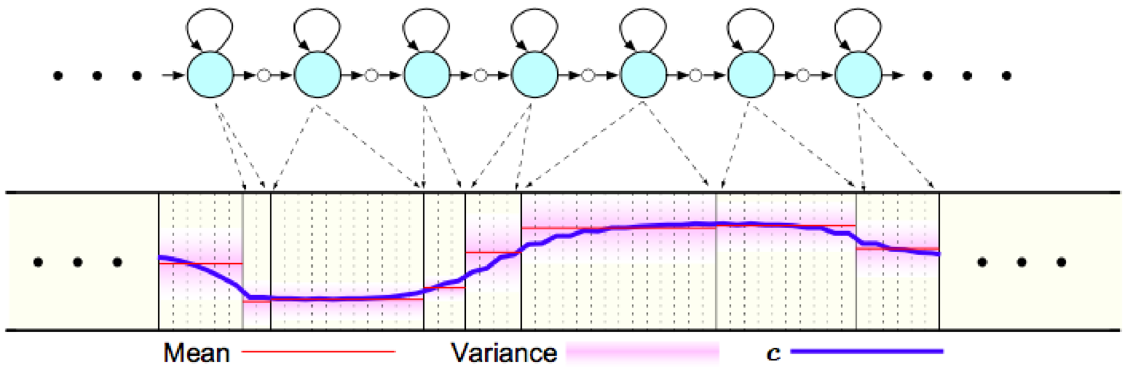


Figure 3.7: Parameter generation with dynamic features to overcome stepwise issue from HSMM [109].

3.3 Overview of feedforward neural networks

Artificial neural networks (ANNs) have been inspired by the human brain. They have been applied to a wide range of problems such as speech synthesis [128] and image understanding [46]. Figure 3.8 depicts three common ANN architectures including; (i) single-layer feedforward neural network, (ii) multi-layer feedforward neural network, and (iii) recurrent neural network. More specific details about recurrent neural networks can be found in Chapter 7. This section discusses how to represent the feedforward networks and how they learn.

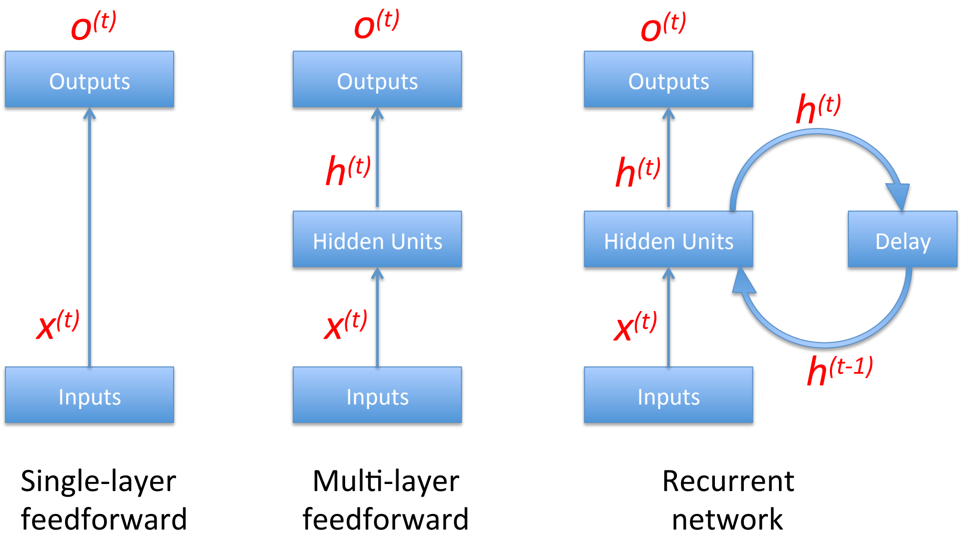


Figure 3.8: The single-layer and multi-layer feedforward neural networks compared with recurrent neural networks.

A single layer perceptron (SLP) is the main processing unit of a single-layer feedforward network. A single-layer perceptron comprises one input layer and one output layer. An example of a neuron in the single-layer perceptron is shown in Figure 3.9, which consists of an input vector $\mathbf{x}$ with D dimensions, $\mathbf{x} = [x_1, x_2, ..., x_D]$, from the input layer and a single scalar output y.

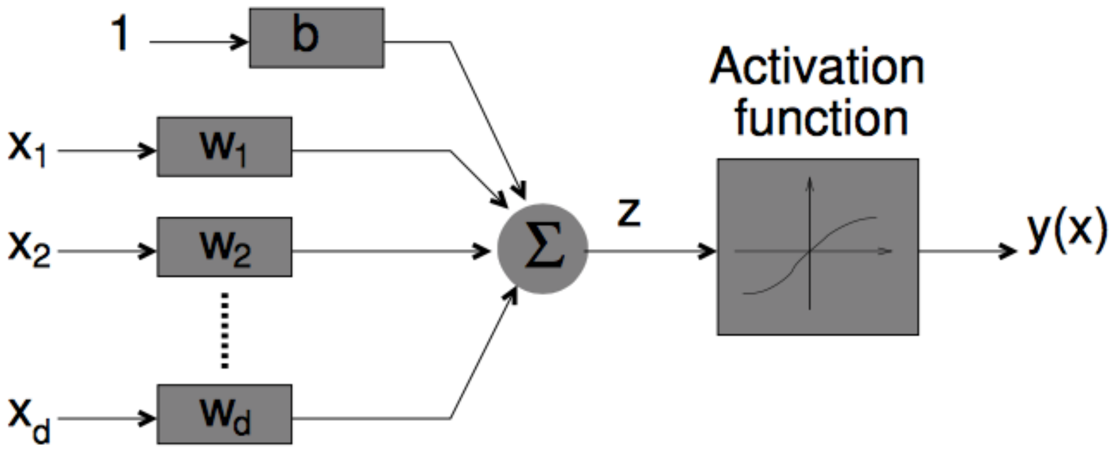


Figure 3.9: The form of single layer perceptron.

A weight vector, $\mathbf{w}$, $\mathbf{w} = [w_1, w_2, ..., w_D]$ is then combined with $\mathbf{x}$ produced output z as follow;

$$z = wx^T + b, \tag{3.23}$$

where b denote a bias which is used to specify the position of a hyperplane decision boundary. The output from the perceptron, a, is calculated by the activation function, f;

$$a = f(z), \tag{3.24}$$

where f is used for limiting the amplitude of the neuron output. This is possibly denoted as either a linear or nonlinear activation function. Examples of the linear activation function are calculated as for example, a binary step function:

$$f(z) = \begin{cases} 0 & \text{if} \quad z < 0 \\ 1 & \text{if} \quad z \geq 0 \end{cases} \tag{3.25}$$

or, a linear function:

$$f(z) = z. \tag{3.26}$$

The activations of the non-linear functions can be written as, for example, a hyperbolic tangent function (tanh),

$$f(z) = \frac{2}{1 + e^{-2z}} - 1. \tag{3.27}$$

or, a rectified linear function (ReLU):

$$f(z) = \begin{cases} 0 & \text{if} \quad z < 0 \\ z & \text{if} \quad z \geq 0 \end{cases} \tag{3.28}$$

Taking an example of the use of single-layer perceptron with a linear function on separating positive and negative data is shown in Figure 3.10. This shows examples of the possible hyperplane boundaries (dotted line) and the optimal hyperplane (solid line) of the linear classifier that aims to separate positive and negative classes. From this figure, it can be seen that the single layer perceptron is able to handle the simple problem in the case of linearly separable data.

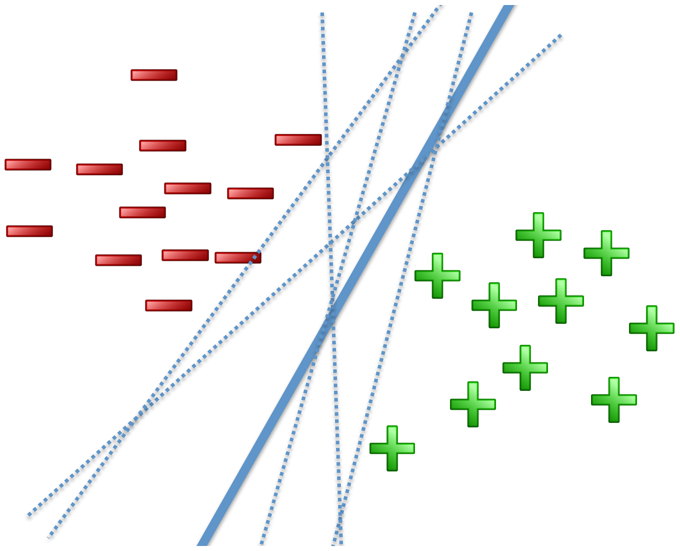


Figure 3.10: Examples of possible hyperplanes and optimal hyperplane of linear classifier.

So far has been discussed the single layer perceptron. Next, we show how a feedforward neural network implements a multi-layer perceptron from a set of single perceptrons. The main reason for the use of a multi-layer perceptron is that the single layer perceptron is limited because it is unable to handle a more complex problems. It is found that a line from single-layer perceptron is not able to classify a non-linearly seperable problem such as the example of "XOR" problem in Figure 3.11.

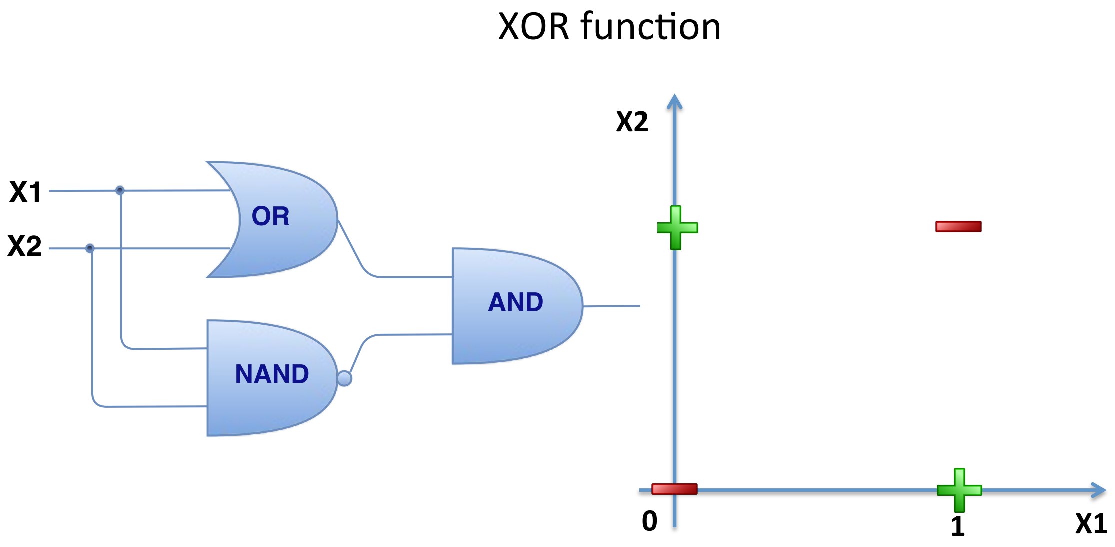


Figure 3.11: Examples of XOR problem that single layer perceptron cannot handle.

The XOR problem in Figure 3.11 has two binary input units, x_1 and x_2, that have either 0 or 1 and returns one binary output with values in 0 and 1, depicted with negative and positive symbols, respectively. We then also consider to model these data with single perceptron but it cannot separate these data with one line, as shown in Figure 3.12 (A and B). To overcome this problem, it requires to use two lines from two single-layer perceptrons, as called multi-layer perceptron, as depicted in Figure 3.12 (C).

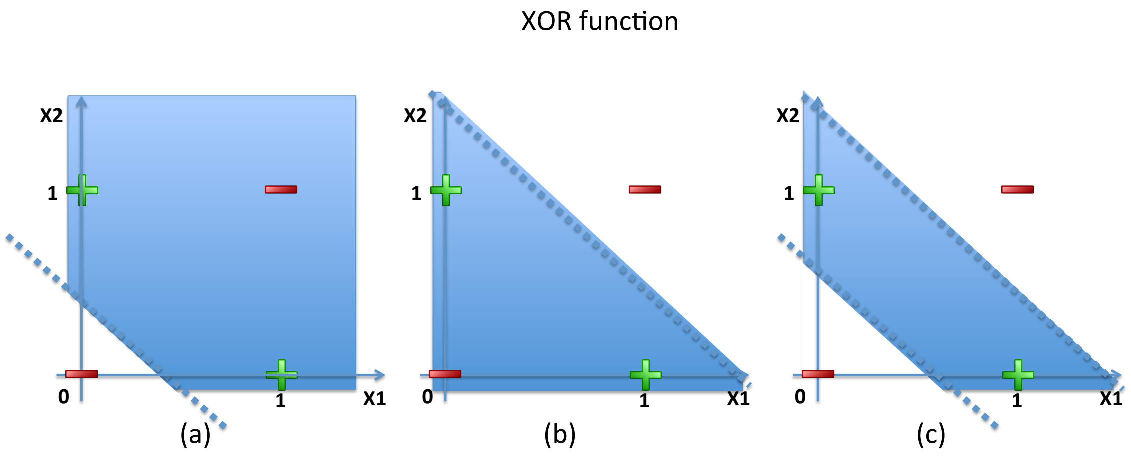


Figure 3.12: Examples of XOR problem that use multi layer perceptron with a tanh activation function.

The use of multi-layer perceptron for the XOR problem comprises two perceptrons in the hidden layer and one perceptron at the output layer, as shown in Figure 3.13. From this example, both hidden and output layer use step functions as the activation function because this is sufficient to find the hyperplane boundary between "1" (positive symbol) and "0" (negative symbol) output data. After training the network, all weight parameters in each layer are learned automatically using the backpropagation algorithm (more details in Section 3.3.1). These weight and bias parameters show the power of neural networks that each perceptron can learn its appropriate function from training data. This is likely to be correct because one of the XOR expressions may be depicted in Figure 3.11 (left) and written (in boolean notation) as;

$$y = (x_1 + x_2) \cdot (\bar{x_1} \cdot \bar{x_2}). \tag{3.29}$$

Clearly, the three perceptron networks in Figure 3.13 can solve the XOR problem, as the first perceptron makes the first decision boundary (shown as Figure 3.12 (a)) and acts as "OR" function. The second perceptron makes the second decision boundary (shown as Figure 3.12 (b)) and act as "NAND" function. Then, the third perceptron combines the first and second perceptron together with "AND" function (shown as Figure 3.12 (c)).

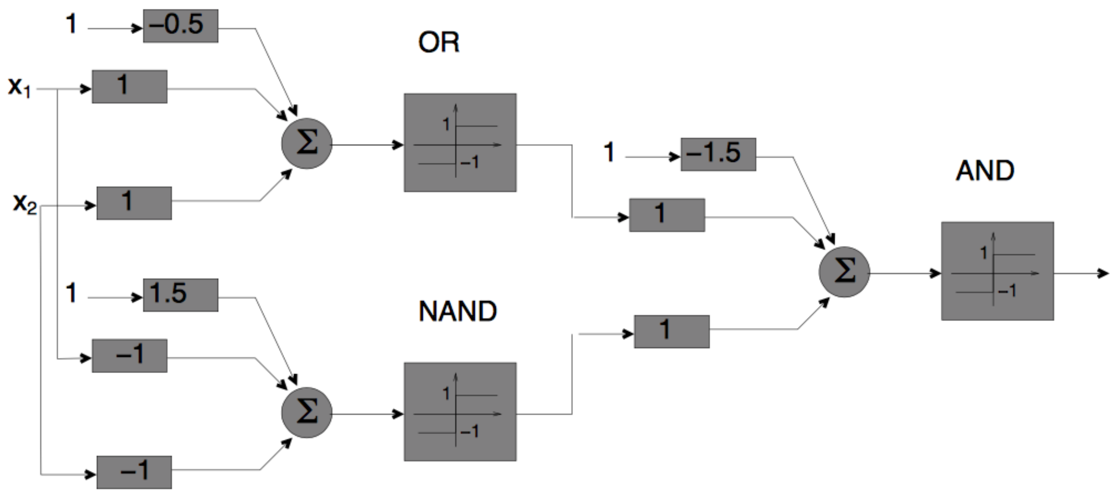


Figure 3.13: Multi-layer perceptron of the XOR problem.

However, if we would like to handle more complex problems that consist of several positive and negative examples in Figure 3.14, this clearly requires the linear activation function of the hidden layer to be replaced by a non-linear activation function. The reason is that the MLP with linear activation function is not able to find the optimal boundary that separate these data. Hence, the MLP with non-linear activation function often uses a feedforward network for finding a non-linear division of boundaries. A common non-linear activation function is the rectified linear unit (ReLU) [44, 63]. Additionally, it has been reported that the ReLU function outperforms the traditional logistic sigmoid and hyperbolic tangent (tanh) functions on various applications such as image recognition [44] and speech recognition [124]. Nair and Hinton [76] reported that there are two major benefits for that; (i) sparsity, (ii) reducing vanishing gradients.

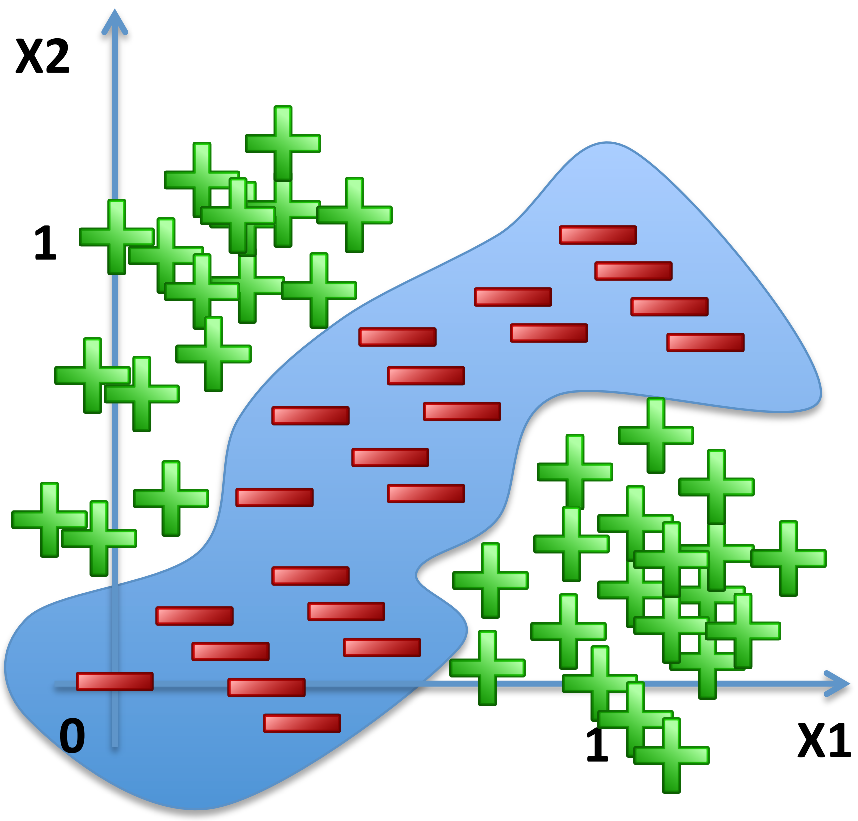


Figure 3.14: Examples of XOR problem that use multi layer perceptron.

From the examples earlier, we use only one hidden layer but we tend to use more than one hidden layer in practical applications, known as deep feedforward neural networks (DNNs). So far we have shown that the multi-layer neural network can learn automatically its appropriate function from training data using the backpropagation algorithm, but have not explained how the backpropagation works. Hence, the next section presents how the algorithm learns their weights and biases from the training examples in the context of general DNNs.

3.3.1 Training feedforward neural network using backpropagation

In the previous section we described how to represent deep feedforward neural networks from single layer perceptrons. We also discussed the limitations of the single-layer networks and showed the benefits of a multi-layer perceptron in feedforward networks. It showed that each neuron uses weights and biases to represent their expression for the XOR problem but it did not show how to learn the weight values. Hence, this section explains how feedforward networks learn their weights and biases in the network using the backpropagation algorithm.

The basic idea of backpropagation was introduced in 1988 by [84], and aims to compute the gradient of the objective or cost function with respect to the weight of all layers, $\nabla_W C$. This function shows how the cost is changed when adjusting the weight parameters to make the cost smaller. The procedure of the backpropagation algorithm comprises three procedures; (i) forward propagation, (ii) backward propagation, and (iii) weight update.

3.3.1.1 Forward Propagation Pass

This step aims to compute all of the activation functions of the network, $h_j^{(l)}$, for $l = 1, 2, 3, ..., L$. Where L denotes the number of layers and j refers to the j^{th} unit in the l^{th} layer. For example in Figure 3.15, there are 4 layers, $L = 4$ for an example of feedforward neural networks. The first layer in the network is called input layer, and the neurons in this layer are called input neurons, x_m, where m refers to the number of input neurons. The last layer is called output layer, and the neurons in this layer are called output neurons, o_n, where n refers to the number of output neurons. The middle layers between input and output layer are called hidden layers, and the neurons within this layer are neither input or output neurons, h_j^l, where j refers to the specific hidden neuron. For example, h_2^3 in Figure 3.15 denotes the activation function of the 2^{nd} unit in layer 3.

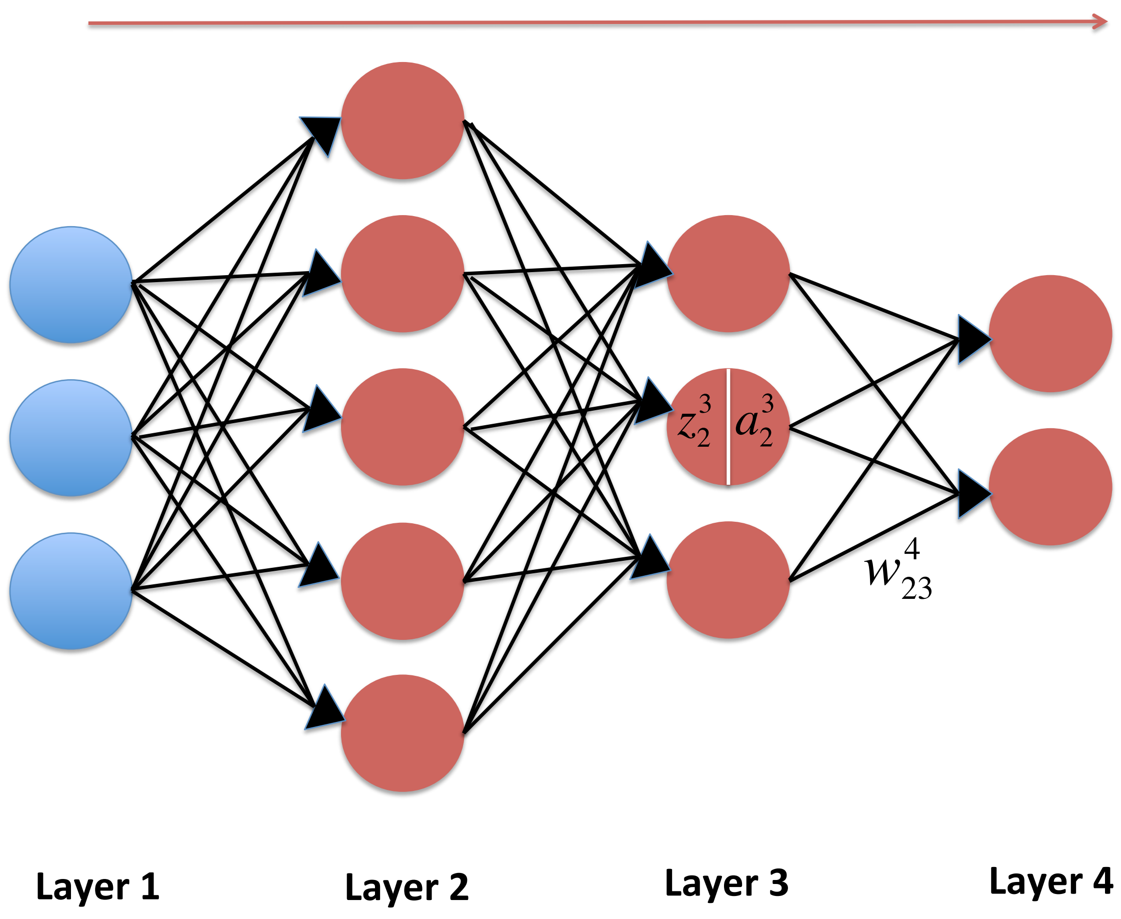


Figure 3.15: An illustration of forward pass.

The calculation of the forward propagation pass starts from the input layer to the output layer, where the output from previous layer is used as the input in next layer.

At input layer, $l = 1$, a given input, $\mathbf{x}$, is fed as an input to the network. Note that, the linear combination, z_j^1, and activation output, h_j^1, in each input neuron refer to the input, x_j, where j is the specific input units $(1 \leq j \leq M)$, as follows;

$$\mathbf{x} = \{x_1, x_2, \ldots, x_M\}, \tag{3.30}$$

$$\mathbf{h}^1 = \{h_1^1, h_2^1, \ldots, h_M^1\}, \tag{3.31}$$

$$h_j^1 = z_j^1 = x_j \tag{3.32}$$

At hidden layers, $l = 2, ..., L-1$, the activation outputs in each neuron are calculated using the weight matrix, $\mathbf{w}^l$, and the activation output from previous layer, $\mathbf{h}^{l-1}$. Note that, the bias parameters are omitted for simplicity. The linear combination in Equation (3.23) can be rewritten as;

$$\mathbf{w}^l = \{w_1^l, w_2^l, \ldots, w_M^l\}, \tag{3.33}$$

$$z^l = w^l h^{(l-1)}, \tag{3.34}$$

where w_{jk}^l denotes to the weight parameters of the k^{th} unit in the previous layer, $l-1^{th}$, to the j^{th} unit in the current layer, l^{th}. $h_k^{(l-1)}$ denotes to the activation output of the k^{th} unit in the previous layer.

Then a non-linear function, f, in Equation (3.28) is used to compute the activation function, h_j^l, and can be rewritten as;

$$h_j^l = f(z_j^l), \tag{3.35}$$

In the case of using a rectified linear unit (ReLU) as a non-linear function, it can be written as;

$$h_j^l = max(0, z_j^l), \tag{3.36}$$

where the hidden neuron output, h_j^l, becomes zero when the input is smaller than zero, $z_j^l < 0$, and then linear with slope 1 when the input is greater than zero, $z_j^l > 0$.

At output layer, $l = L$, the linear combination can be computed as the same as in the hidden layer. The use of activation function depends on the problem. In the case of a classification problem, a softmax function is often used as the activation function. With regression problem, a linear function often uses as the activation function. So that the output unit, o_j^l, can be written as follows;

$$z_j^L = \sum_k w_{jk}^L h_k^{(L-1)}, \tag{3.37}$$

$$o_j = h_j^L = z_j^L, \tag{3.38}$$

where the output neuron in each unit can be denoted as o_j. Then, this output will be used in the backward propagation pass in the next Section.

3.3.1.2 Backward Propagation Pass

This step aims to compute the partial derivative of the cost function, E, with respect to a weight parameter of each unit, w_{jk}^l, in the network, as denoted in $\frac{\partial E}{\partial w_{jk}^l}$. The backward pass propagates the partial derivative from the L^{th} layer down to the 2^{nd} layer using the chain rule, as shown in Figure 3.16.

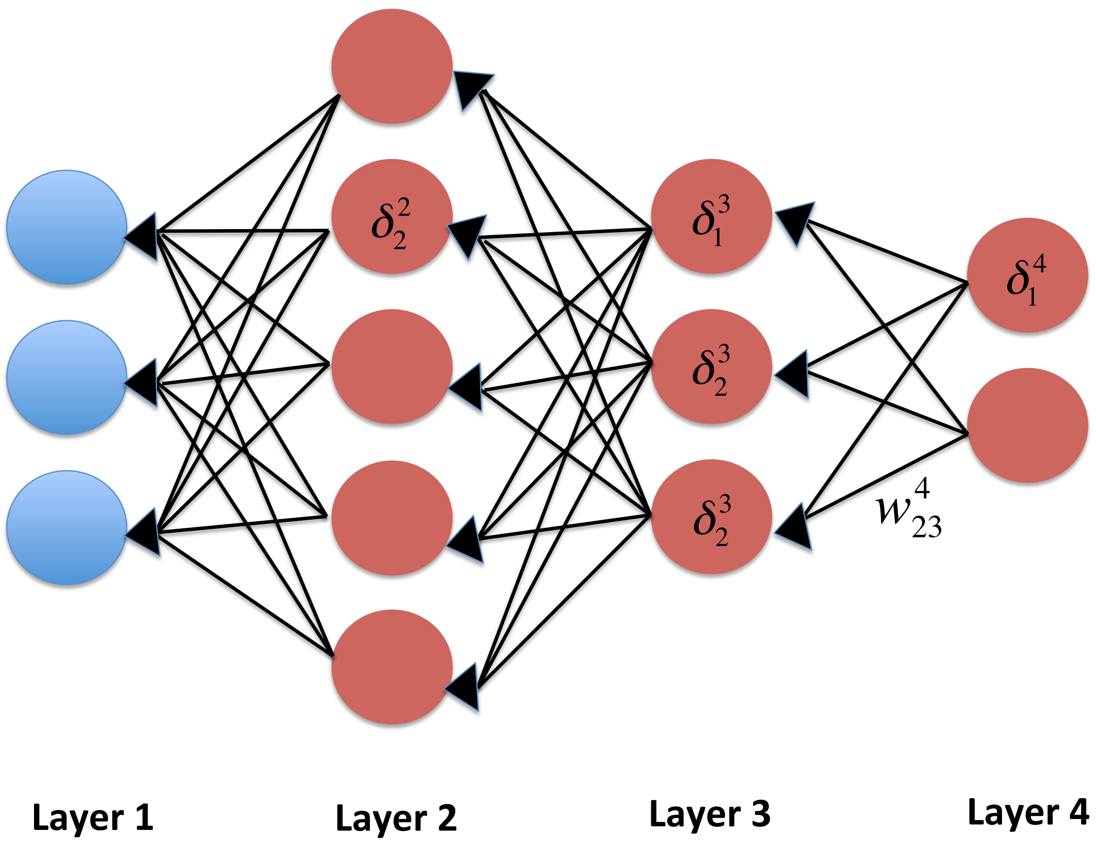


Figure 3.16: An illustration of backward pass.

At the output layer, the cost function computes the error between the reference vector answer, $\mathbf{y}$, and the activation output from the forward propagation step, $\mathbf{o}$, as

$$\mathbf{y} = \{y_1, y_2, \ldots, y_n\}, \tag{3.39}$$

$$\mathbf{o} = \{o_1, o_2, \ldots, o_n\}. \tag{3.40}$$

where n is the number of output neurons, as $n = 2$ in Figure 3.16. The common standard cost or error function for the regression problem is the least square error, E, as follows;

$$E = \frac{1}{2}(\| \mathbf{y} - \mathbf{o} \|)^2 = \frac{1}{2}\sum_j (y_j - o_j)^2, \tag{3.41}$$

where o_j is the activation output at the output layer at j^{th} unit, and y_j is the reference output of the j^{th} unit. This expression will be used to compute the output error in each output neuron, δ_j^L. As each neuron consists of a summation component and an activation component, the delta error in each neuron can be defined by;

$$\delta_j^L = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial z_j^L}, \tag{3.42}$$

where the second term on the right, $\frac{\partial o_j}{\partial z_j^L}$, is cancelled in the case of using linear output as mentioned in Equation (3.38). The first term on the right, $\frac{\partial E}{\partial o_j}$, measures the change in E with respect to the output in each output unit. Note that, the factor of $\frac{1}{2}$ is cancelled when applying differentiation as follow;

$$\frac{\partial E_j^l}{\partial o_j} = o_j - y_j. \tag{3.43}$$

Once the output error at the output layer, δ_j^L, is known. The gradient of the cost function with respect to the weight parameters of the output layer can be defined by;

$$\frac{\partial E}{\partial w_{jk}^L} = \frac{\partial z^L}{\partial w_{jk}^L} \cdot \delta_j^L, \tag{3.44}$$

where the gradient of the linear combination of output layer, as shown in Equation (3.37), can be derived by

$$\frac{\partial z^L}{\partial w_{jk}^L} = h_k^{(L-1)}, \tag{3.45}$$

so that,

$$\frac{\partial E}{\partial w_{jk}^L} = h_k^{(L-1)} \cdot \delta_j^L, \tag{3.46}$$

At hidden layer, $l = L-1, ..., 2$, the output error in each hidden unit, δ_j^l, can be defined as

$$\delta_j^l = \frac{\partial E}{\partial h_j^l}\frac{\partial h_j^l}{\partial z_j^l}, \tag{3.47}$$

where

$$\frac{\partial E}{\partial h_j^l} = \sum_k w_{jk}^{(l+1)} \delta_k^{(l+1)}. \tag{3.48}$$

Once the output error at the hidden layer l in each unit j, δ_j^l, is known, the gradient of the cost function with respect to the weight parameters of the hidden layer can be given by

$$\frac{\partial E}{\partial w_{jk}^l} = \frac{\partial z^l}{\partial w_{jk}^l} \cdot \delta_j^l, \tag{3.49}$$

where,

$$\frac{\partial z^l}{\partial w^l_{jk}} = h_k^{(l-1)}, \tag{3.50}$$

so that,

$$\frac{\partial E}{\partial w^l_{jk}} = h_k^{(l-1)} \cdot \delta^l_j. \tag{3.51}$$

Note that the partial derivative of the cost function with respect to all weight parameters, $\frac{\partial E}{\partial \mathbf{W}}$, will be used to optimise the new weight parameters in the next step.

3.3.1.3 Optimisation: Weight Update

This procedure aims to minimise a cost function by changing the weight parameters in the case of knowing the gradient error in each neuron in the network from the backward propagation pass. This minimisation of E is used in an iterative process of gradient descent defined by

$$\frac{\partial E}{\partial \mathbf{W}} = (\frac{\partial E}{\partial \mathbf{w}^2} \cdots \frac{\partial E}{\partial \mathbf{w}^L}). \tag{3.52}$$

There are many gradient descent optimisation that can be used to minimise the objective function such as stochastic gradient descent (SGD), root mean square propagation (RMSProp), adaptive moment estimation (Adam), and so on [46]. This chapter focuses on SGD with the network weights updated as follows:

$$\widehat{\mathbf{W}} = \mathbf{W} - \propto * \frac{\partial E}{\partial \mathbf{W}}, \tag{3.53}$$

where $\widehat{\mathbf{W}}$ is the updated tensor matrix, and $\propto$ is the learning rate parameter ($0 \leq \propto \leq 1$), which is used to control how quickly the network learns. The learning can be too slow if $\propto$ is set too low, and the weights can oscillate if $\propto$ is set too high. One technique that can help the SGD to learn quicker and move to the right direction with global minima is called SGD with momentum. The momentum is given by,

$$\widehat{\mathbf{V}} = \gamma \mathbf{V} + \propto * \frac{\partial E}{\partial \mathbf{W}}, \tag{3.54}$$

$$\widehat{\mathbf{W}} = \mathbf{W} - \widehat{\mathbf{V}}, \tag{3.55}$$

where $\widehat{V}$ is the updated velocity matrix, which is the same size of weight parameters and initials with all zero, with γ the momentum value. Note that, if γ is set to zero, this means the new weight parameters are updated with SGD as described in Equation (3.53).

In batch mode, the weights are updated after all training examples have been used to compute gradient errors. In practice, the computation time for all training examples is likely to be prohibitively long. Hence, the use of mini-batches is used to estimate the gradient errors instead. The number of mini-batches, mb, refers to the number of samples in each batch, which are (not) randomly selected from the training data. Hence, the following algorithm applies a gradient descent learning step based on that mini-batch:

$$\widehat{\mathbf{W}} = \mathbf{W} - \frac{\propto}{mb} * \sum_{x} \frac{\partial E^{x}}{\partial \mathbf{W}^{x}}, \tag{3.56}$$

3.4 Overview of recurrent neural networks

Recurrent neural networks (RNNs) are another type of neural network designed to process sequential data, for example, text, speech, image, etc. They were first introduced in [83] and have been shown to outperform feedforward network in a range of applications, such as machine translation, automatic speech recognition, and handwritten character recognition.

Figure 3.17 (a) shows the conventional feedforward neural network architecture. This is compared to the RNN archtecture in Figure 3.17b with the introduction of a feedback connection. It can be seen that the difference between the architectures is that the RNN structure consists of the feedforward structure with cycle or recurrence connections as a time-delay between nodes. From these recurrent structures, RNNs have the advantage that contextual information from previous input vectors can be memorised over timesteps, which means RNNs are can be considered as being stateful. Conversely, traditional feedforward neural networks are stateless, as the output at timestep t depends only on the current input at time t.

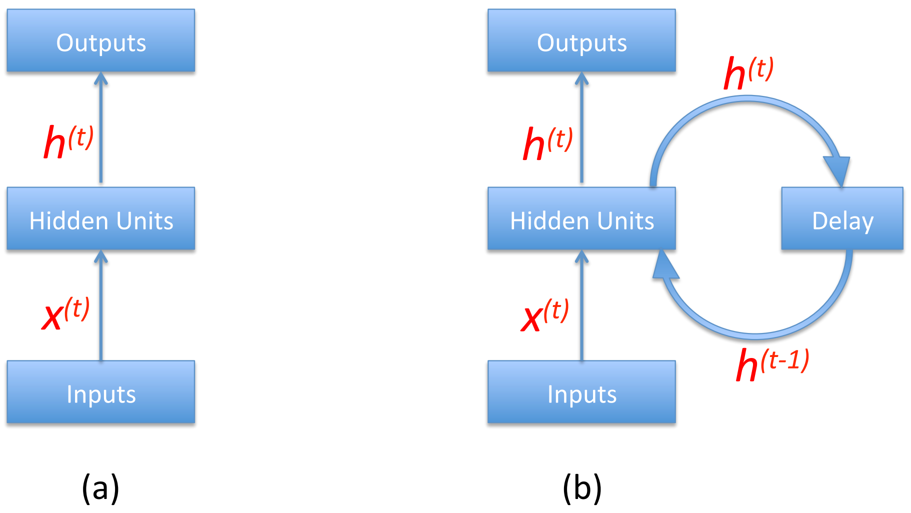


Figure 3.17: Simple feedforward and RNN architectures.

The goal of recurrent networks is to learn a mapping function that maps input sequences to output sequences. For example, in the regression problem, we define inputs and outputs as a sequence of data with time steps ranging from 1 to τ, where τ is the length of the sequence. For such basic RNNs, we assume that both inputs, $\mathbf{X}$, and outputs, $\mathbf{O}$, have the same length, as follows:

$$\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^\tau\}, \tag{3.57}$$

$$\mathbf{O} = \{\mathbf{o}^1, \mathbf{o}^2, \ldots, \mathbf{o}^\tau\}. \tag{3.58}$$

where each time step or frame consists of m input features, $\mathbf{x}^t \in \mathbb{R}^m$, and n output features, $\mathbf{o}^t \in \mathbb{R}^n$.

$$\mathbf{x}^t = \langle x_1^t, x_2^t, \ldots, x_m^t \rangle, \tag{3.59}$$

$$\mathbf{o}^t = \langle o_1^t, o_2^t, \ldots, o_n^t \rangle, \tag{3.60}$$

The use of time-delayed recurrence can be found in the forward propagation of RNNs as follows; where the hidden state vector, $\mathbf{h}^t$ at time t is calculated

$$\mathbf{h}^{(t)} = f(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b}), \tag{3.61}$$

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)} + \mathbf{c}, \tag{3.62}$$

where $t = 1, \ldots, \tau$; $\mathbf{h}^{(t)} \in \mathbb{R}^k$, k is the number of hidden units. Matrices $\mathbf{W}, \mathbf{U}$, and $\mathbf{V}$; are the weight parameters with; $\mathbf{U} \in \mathbb{R}^{k \times m}$ the weight matrix for input-to-hidden connection, $\mathbf{W} \in \mathbb{R}^{k \times k}$ the parameter for hidden-to-hidden connection and $\mathbf{V} \in \mathbb{R}^{n \times k}$ the hidden-to-output weight connection. RNNs share the same weight parameter in every time step which benefits the RNNs in the case of training with different sequence lengths. Vectors $\mathbf{b}$ and $\mathbf{c}$ are the hidden and output bias; and f denotes the nonlinear activation function for hidden nodes. This generally uses a logistic sigmoid function or a hyperbolic tangent function [46].

Figure 3.17 (b) and Equation (3.61) show that the new hidden state or internal state, $\mathbf{h}$, operates not only based on the input time at step t, $\mathbf{x}^{(t)}$, but also on the previous hidden state, $\mathbf{h}^{(t-1)}$. That means the internal state has a relationship to the whole past sequence of inputs up to t. In the first time step of each sequence, the initial values of the internal state, $\mathbf{h}^{(0)}$, are set to zero. After that, the output of the RNN at time t, $\mathbf{o}^{(t)}$, can be computed by equation (3.62). Given weight parameter, $\mathbf{V}$, and bias, $\mathbf{c}$, the output at time t, $\mathbf{o}^{(t)}$, depends only on the state of the internal state at time t, $\mathbf{h}^{(t)}$, much like a feedforward neural network. As mentioned earlier the RNN structure is built around feedforward networks, a key feature that distinguishes feedforward from RNNs is feedback hidden state. Hence, we can easily transform RNNs to a feedforward structure by setting $\mathbf{W}$ to zero.

3.4.1 Unidirectional RNN architectures

The neural network we just described in the above section is based on the assumption that the length of input and output sequences are equal. In practice, different tasks may have different input and output lengths. For example, a part-of-speech (POS) tagger requires the same length of input and output because the goal of POS tagger is to find the POS in each single word [117]. However, some tasks represent inputs and outputs with different lengths such as machine translation showing a different number of words for a source and a target language [95]. Therefore, this section will explore how RNNs handle sequential data of arbitrary length in four aspects, namely: (i) synchronised many-to-many, (ii) many-to-one, (iii) one-to-many, and (iv) encoder-decoder many-to-many.

3.4.1.1 Synchronised many-to-many

This recurrent network's structure is called synchronised many-to-many because it has the same length of input sequence and output sequence, which is the same approach as described in the previous section. Figure 3.18 illustrates another common graphical diagram, known as the unfolded or unrolled RNN, which is used to convert a cyclic recurrent neural network representation into a multilayer feedforward neural network by unfolding over time.

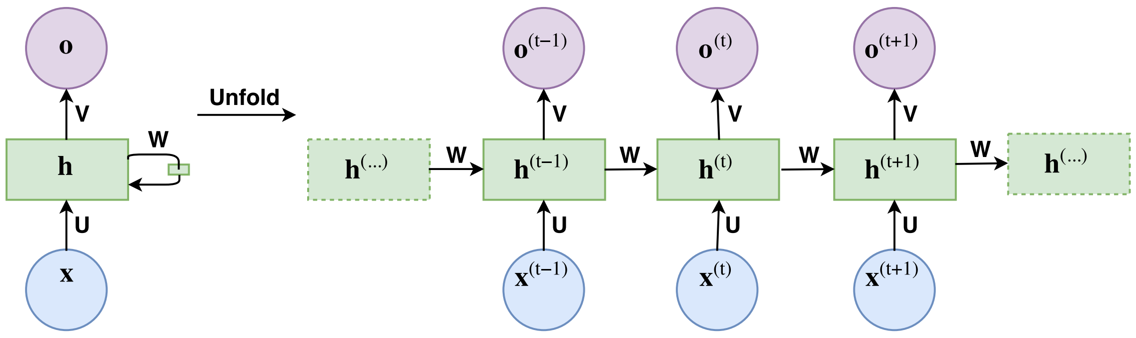


Figure 3.18: Synchronised many-to-many RNN architecture (adapted from [46]).

The time-unfolded RNN shows clearly that the full network takes an input frame at each time step and produces an output frame at the same time step input. With feedback mechanism, networks have the same weight matrices, $\mathbf{U}, \mathbf{W}, \mathbf{V}$ at every time step. The activation function of the previously hidden state and the current input act as an internal memory of the whole past sequence of inputs up to frame t. Then, the output layer uses Equation (3.62) to compute the output values.

There have been several extensions to this RNN architecture. For example, [59] proposed to use output-to-hidden recurrent connections instead of hidden-to-hidden recurrent connections. They have the feedback connections from the output at one time step to the hidden layer at the next time step. The recurrent hidden layer of the basic RNN in Equation (3.61) is updated as follows,

$$\mathbf{h}^{(t)} = f(\mathbf{R}\mathbf{o}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b}), \tag{3.63}$$

where $\mathbf{o}^{(0)} = 0, \mathbf{R}$ corresponds to the weight for output-to-hidden recurrence connection. The other parameters use the same definition as Equation (3.61). The advantage of using output-to-hidden recurrence in training is that there is no need to compute the previous time step first, because we can use the correct output, from the training set, known as teacher forcing [46]. Figure 3.19 shows an example of using reference output from the previous time step, $\mathbf{y}^{(t-1)}$, when calculating the hidden layer at time t, $\mathbf{h}^{(t)}$. However, we still need to compute and feed the predicted output, $\mathbf{o}^{(t-1)}$, back to the model in the testing step.

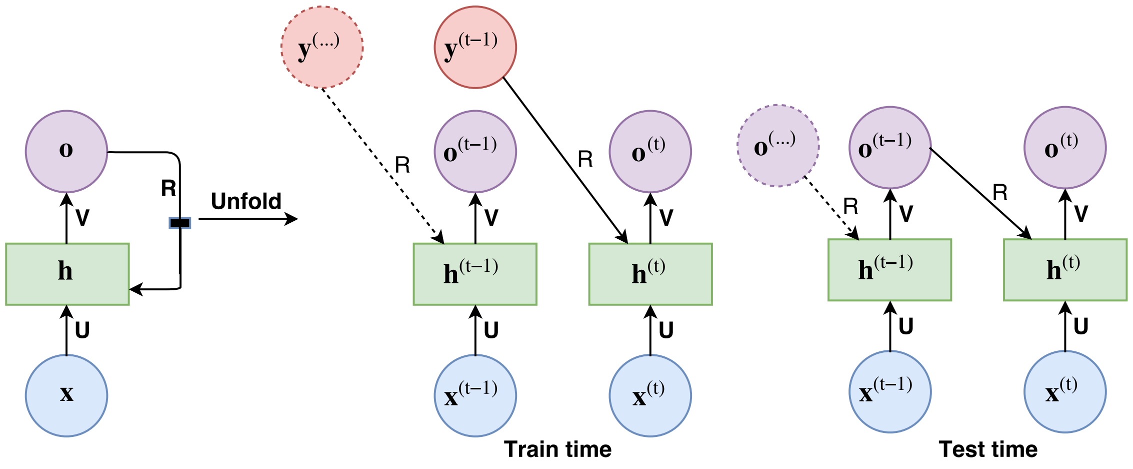


Figure 3.19: Synchronised many-to-many RNN architecture with teacher forcing (adapted from [46]).

In [125], the authors proposed a new output-to-output recurrence connection to the basic hidden-to-hidden recurrence connection. That means this architecture has two recurrences connections. The first recurrent connection, parametrised by a new weight parameter, $\mathbf{R}$, from the previous output to the current output state. The second recurrent connection, parameterised by the same weight parameter, $\mathbf{W}$, from the past hidden to the present hidden state. Figure 3.20 shows that the forward propagation at the hidden layer is still the same as the basic RNN in Equation (3.62). However, the new output-to-output recurrence terms will be added to the output layer in each time step as follow;

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)} + \mathbf{R}\mathbf{o}^{(t-1)} + \mathbf{c}, \tag{3.64}$$

where $\mathbf{o}^{(0)} = 0, \mathbf{R} \in \mathbb{R}^{n \times n}$ corresponds to the weight for output-to-output recurrence connection, n is the number of output units. The other parameters use the same definition as Equation (3.62). This architecture has been applied to text-to-speech synthesis, which found that the new output recurrence state improved the naturalness of the synthesised speech [125], as the recurrent output layer was ask to create smooth transitions at the frame boundaries.

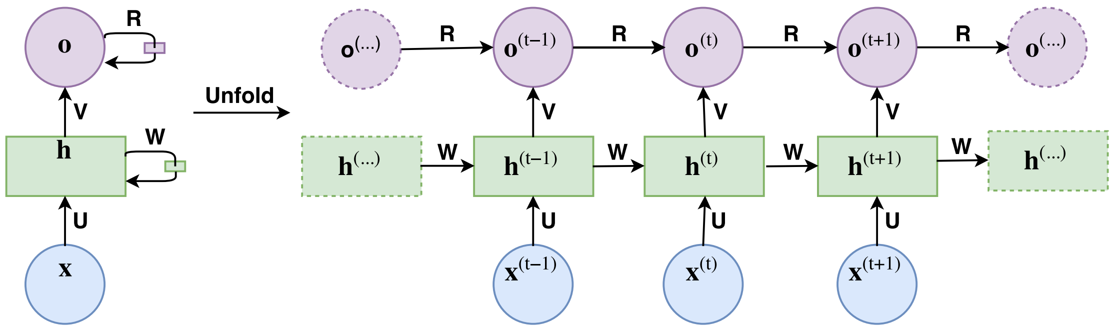


Figure 3.20: Synchronised many-to-many RNN architecture with two recurrent connections (adapted from [46]).

3.4.1.2 Many-to-one

This recurrent network aims to learn a sequence input and produce single time step output or fixed-size output vector. The general idea of this architecture is that it allows the network to see and learn the whole input information in different time steps. Then, the networks predicts a one time step output, as depicted in Figure 3.21.

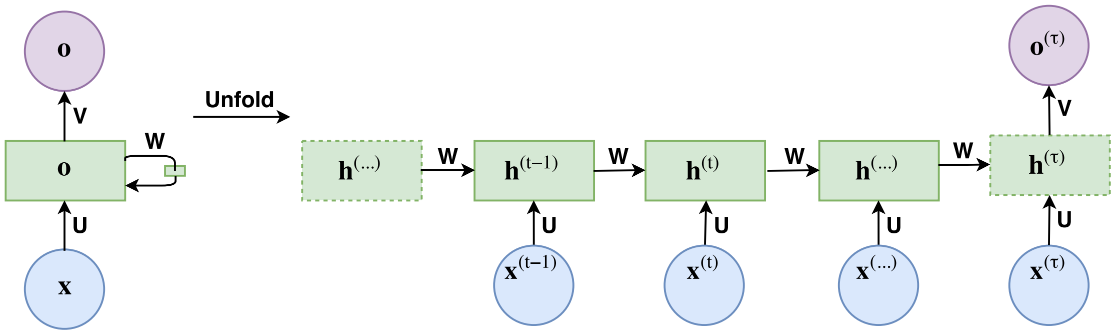


Figure 3.21: Many-to-one RNN architecture (adapted from [46]).

There are many ways to prepare the training data for this architecture. In fact, this architecture is commonly used in NLP applications, for example sentiment analysis and text generation. For the sentiment analysis task [65], RNNs can learn the sentiment (such as positive or negative) from the paragraph or sentence. Then, the learned networks are used to predict the sentiment of a paragraph or sentence. For the text generation task [48], this architecture predicts the next word or character from the past word sequences. The most common approach splits the book into blocks with a fixed-length window. Then, the next block is created by sliding this window one character at a time. From this way, if the input sequence is set to length N that means the RNN learns the next character (i.e. $N + 1$ character) from the N preceding characters.

3.4.1.3 One-to-many

Many applications have a fixed-length input and require an output sequence. For example, an image captioning task takes a single image as an input, $\mathbf{X}$. Then, the output, $\mathbf{O}$, will be generated as a sequence of words describing the image. Figure 3.22 shows the common one-to-many architecture. Compared with the many-to-many architecture, it seems that both architectures pass the input through a hidden layer at every time step. The main difference is the one-to-many approach uses the same input vector at every time step while the synchronised many-to-many uses different inputs depending on the time step order. With a single hidden feedback connection, applying the same input will make no difference at the output at each time step. To overcome this problem, an extra input at each time step is added to the network. Hence, the output at the current time step is fed to the next hidden state. That means this architecture has two recurrent connections. The first recurrent connection is parametrised by a new weight parameter, $\mathbf{U}$, from the previous output, $\mathbf{o}^{(t-1)}$, to the current hidden state, $\mathbf{h}^{(t)}$. The second recurrent connection is parameterised by a weight parameter, $\mathbf{W}$, from the previous hidden state, $\mathbf{h}^{(t-1)}$, to the current hidden state, $\mathbf{h}^{(t)}$.

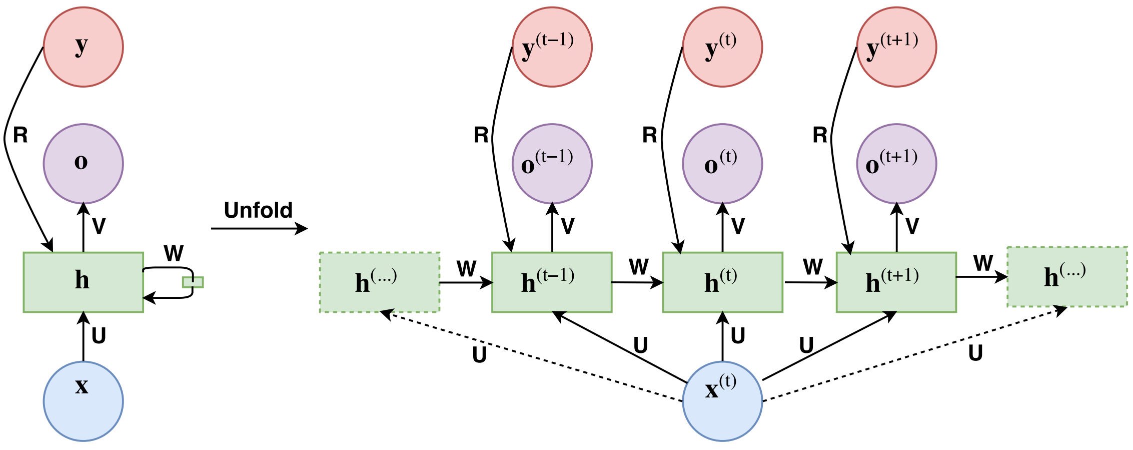


Figure 3.22: One-to-many RNN architecture (adapted from [46]).

3.4.1.4 Encoder-decoder many-to-many

This architecture aims to map an input sequence into an output sequence. Comparing with synchronised many-to-many, this RNN allows the networks to learn the different length of the input and output time steps. This architecture is useful for many tasks, for example, machine translation where the text from the source and target language are probably not a one-to-one mapping relationship [95]. That is the main reason of varying input-output length.

This architecture comprises the combination of the many-to-one and one-to-many architectures, as shown in Figure 3.23. The first part of the network is often called an encoder. As the input sequence is mapped to a fixed-sized vector represented by a context vector, **c**. The idea is to use one RNN for the encoder that aims to summarise the input sequence to a fixed-sized vector. After that, the second part of the network uses another RNN to generate an output sequence back from the context vector, **c**, called a decoder.

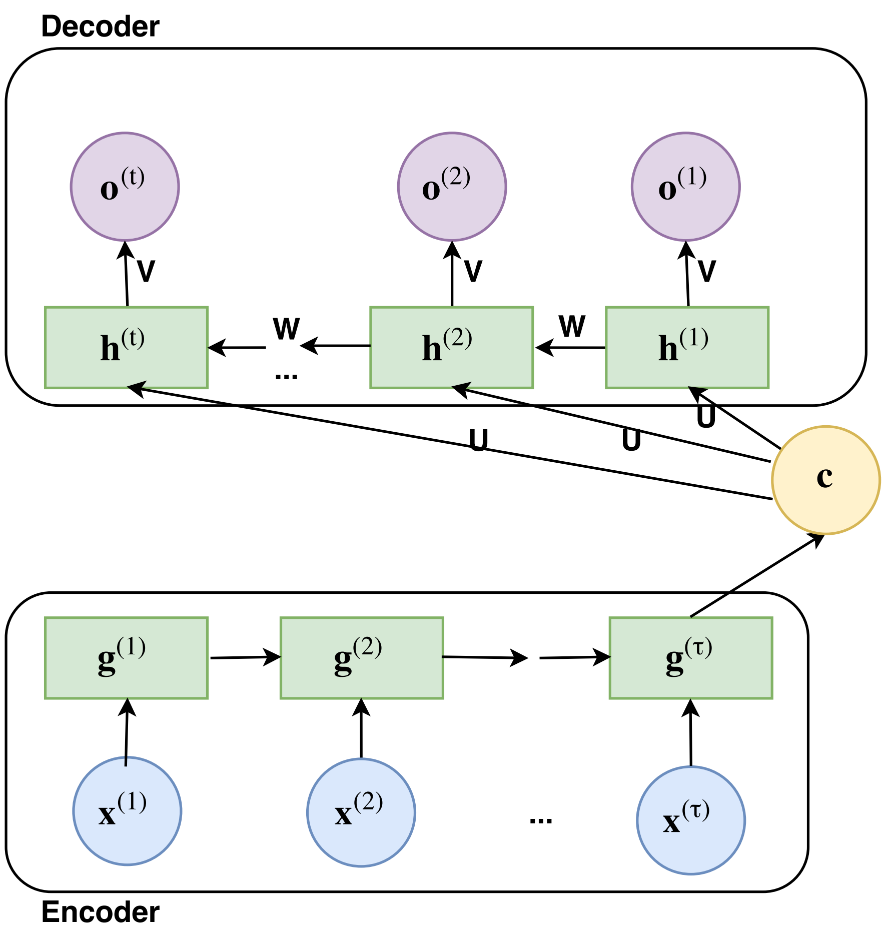


Figure 3.23: Basic encoder-decoder many-to-many RNN architecture (adapted from [46]).

Figure 3.23 represents the full network of the basic encoder-decoder many-to-many approach. The many-to-one approach is used to encode the input sequence to the context vector representation, $\mathbf{c}$. Where $\mathbf{c}$ is calculated by the last time step of the RNN encoder. Then, the one-to-many approach is used to decode the context vector, $\mathbf{c}$, to the output sequence, $\mathbf{y}$. For the RNN decoder the context vector, $\mathbf{c}$, is used as an initial hidden state, $\mathbf{h}^{(0)} = \mathbf{c}$. Also, the hidden state of the decoder at time t is computed by;

$$\mathbf{h}^{(t)} = f(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{c} + \mathbf{b}), \tag{3.65}$$

where $\mathbf{U}$ and $\mathbf{W}$ are the weight parameter of input-hidden and hidden-hidden connection, respectively.

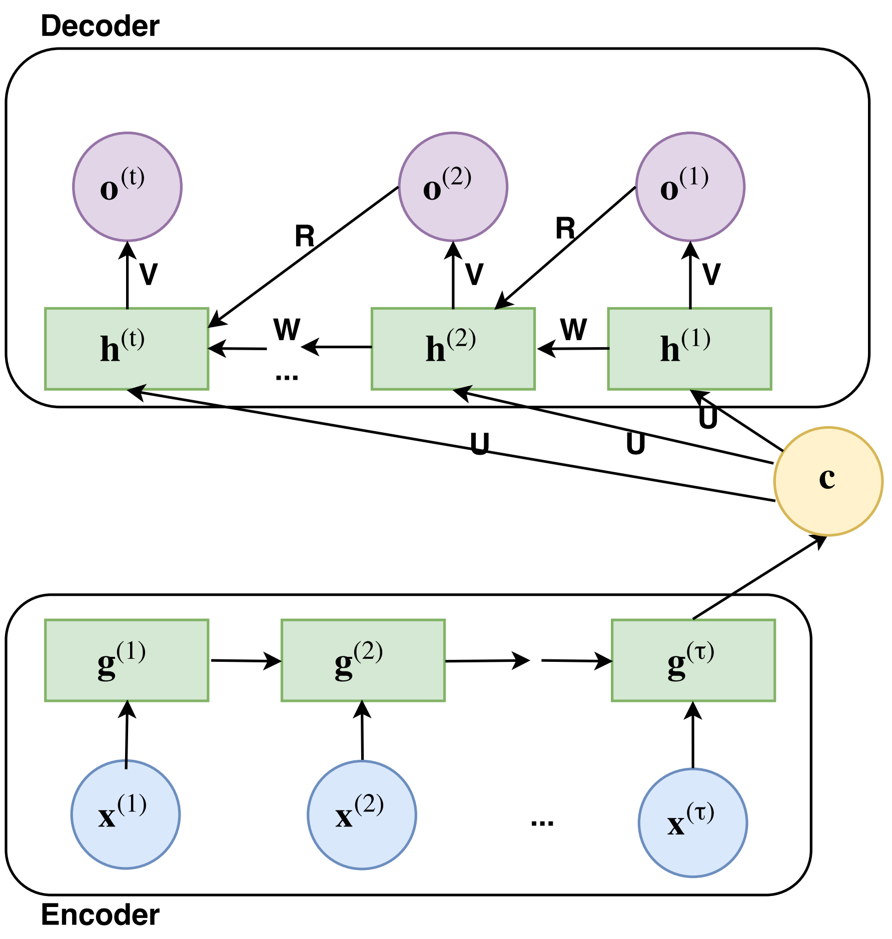


Figure 3.24: Encoder-decoder with feedback (adapted from [46]).

Many variation of encoder-decoder RNNs have been proposed in the literature. For example, [95] proposed an encoder-decoder with additional output-to-hidden feedback connections, as illustrated in Figure 3.24. It shows that the encoder part is still the same as in the basic encoder-decoder approach. The major difference can be found in decoder part. Firstly, $\mathbf{c}$, is used as initial input, $\mathbf{y}^{(0)} = \mathbf{c}$, and initial hidden state, $\mathbf{h}^{(0)} = \mathbf{c}$. Also, the output of the decoder at each time step also becomes the input to the decoder at the next time step, which is called output-hidden recurrence. Hence, the hidden state of the decoder at time t is computed by;

$$\mathbf{h}^{(t)} = f(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{R}\mathbf{y}^{(t-1)} + \mathbf{b}), \tag{3.66}$$

where $\mathbf{R}$ and $\mathbf{W}$ are the weight parameter of output-hidden and hidden-hidden connection, respectively.

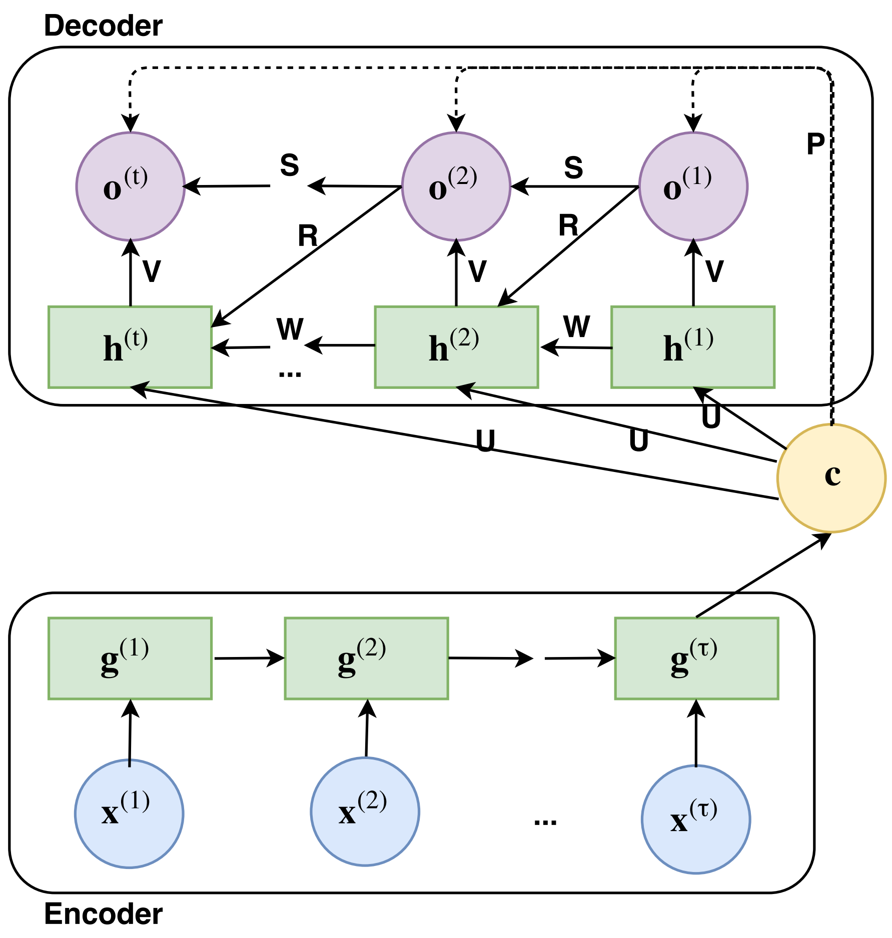


Figure 3.25: Encoder-decoder with peek (adapted from [46]).

The next example of encoder-decoder models, proposed by [15], is called encoder-decoder with peek. Clearly, the encoder RNN is still the same, but the main modification is on the decoder part. It consists of three recurrence connections including: i) hidden-to-hidden, ii) output-to-output, and iii) output-to-hidden. Hence, the hidden state and output state at time t can be computed as follows,

$$\mathbf{h}^{(t)} = f(\mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{R}\mathbf{y}^{(t-1)} + \mathbf{b}), \tag{3.67}$$

$$\mathbf{y}^{(t)} = f(\mathbf{V}\mathbf{h}^{(t)} + \mathbf{S}\mathbf{y}^{(t-1)} + \mathbf{P}\mathbf{c} + \mathbf{c}), \tag{3.68}$$

where $\mathbf{R}$, $\mathbf{W}$, $\mathbf{V}$, $\mathbf{S}$, and $\mathbf{P}$ are the weight parameter of output-hidden, hidden-hidden, hidden-output, output-output, and input-output connection, respectively. The context vector, $\mathbf{c}$, is used as the initial input and the initial hidden state, $\mathbf{h}^{(0)} = \mathbf{y}^{(0)} = \mathbf{c}$.

3.4.2 Bidirectional RNN architectures

With standard or unidirectional RNNs, future information cannot be reached from the current state. RNNs consider only the past context, as shown in equation (3.61). This indicates that the hidden representation at time step t is computed based on current input information at time t and the previous hidden state, $\mathbf{h}^{(t-1)}$. Unidirectional RNNs are well-suited for some tasks, for example text generation because we do not know the next words and we see only the previous words. On the contrary, many tasks are able to access and require both the past and future contexts, such as grapheme-to-phoneme conversion, speech recognition, and text-to-speech. To overcome these problems, bidirectional recurrent neural networks (BRNN) were introduced by Schuster and Paliwal [90] which aim to consider both the past context and future context.

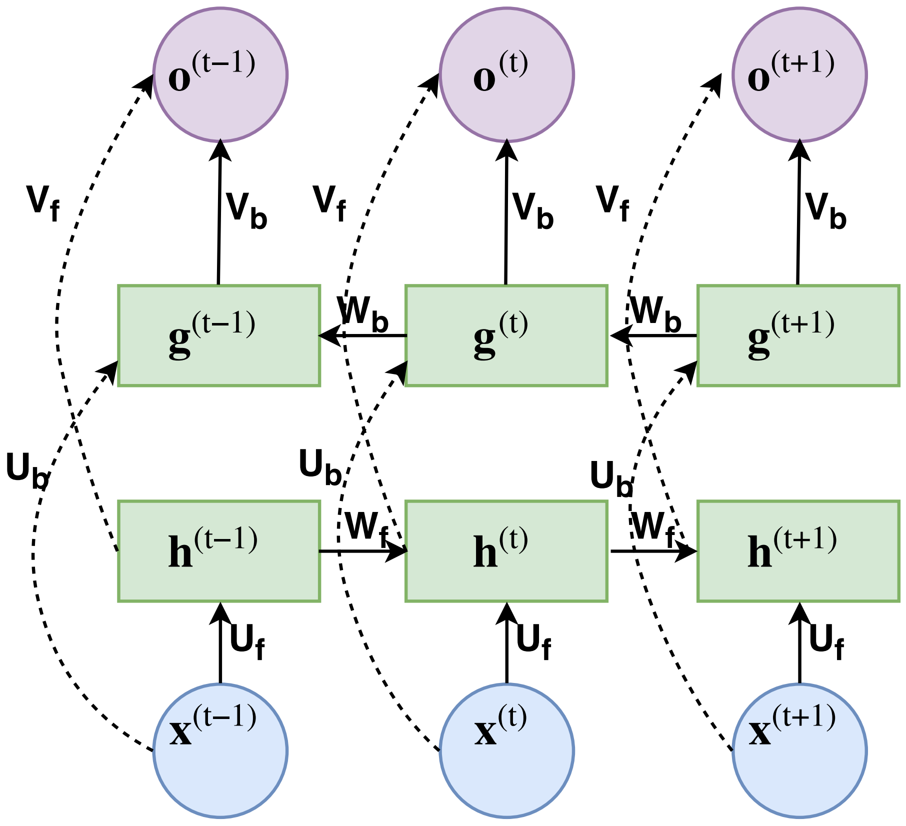


Figure 3.26: Bidirectional RNN (adapted from [46]).

The general structure of BRNNs consists of two parallel hidden layers propagating in two directions to the same output, as shown in Figure 3.26. The first layer, a forward pass, processes the original time step input sequence. The second layer, a backward pass, processes the reverse input sequence time step by time step. Since there are two RNN layers, each RNN shares their parameters $\mathbf{U}_f, \mathbf{W}_f, \mathbf{V}_f$ for the forward pass and $\mathbf{U}_b, \mathbf{W}_b, \mathbf{V}_b$ for the backward pass. The output vector can created using by various options. For example, the summation of the forward and the backwards hidden layer, $\mathbf{o}^{(t)} = [\mathbf{o}_f^{(t)} + \mathbf{o}_b^{(t)}]$, the concatenation of these two layers, $\mathbf{o}^{(t)} = [\mathbf{o}_f^{(t)}, \mathbf{o}_b^{(t)}]$, and so on.

3.4.3 Long short term memory (LSTM)

The RNNs are designed to memorise previous information to predict the present output, as mentioned in section 3.4.1. In theory, standard RNNs should capture such long-term dependencies. However, it has been reported that they may not had the ability to learn these long-term dependencies in real applications [7]. For example, language models have been successful in the case of short-term dependencies, but can not predict a correct word in the case of a relevant word that happened several steps apart from the current word. Hochreiter and Schmidhuber [55] and Bengio et al. [7] found that the difficulty of learning long-term dependencies happens during the gradient backpropagation through time. This issue is also known as the vanishing gradients problem and the exploding gradients problem.

In order to deal with the unstable gradients problem, long short term memory networks (LSTMs) were introduced by Hochreiter and Schmidhuber[55]. They have been widely used in major technology companies including Microsoft [117], Baidu [4] and Google [128] and have been found to be extremely successful in various kinds of applications, such as winning the ICDR handwriting recognition competition in 2009 [43]. The LSTM is a recurrent neural network that uses memory blocks instead of classical neurones. Each LSTM block contains gates that it uses to manage the information into and out of its block. They called gates because each gate uses a sigmoid function, which is in the range of 0-1, to make decisions about storing, reading and writing cell memory. There are three types of gates within the LSTM, namely: input, $\mathbf{i}$, forget, $\mathbf{f}$, and output gate, $\mathbf{o}$. Notice that, each gate has the same equation (as shown in equation 3.69 - 3.71), just with different input-hidden weights,

U, and recurrent hidden-hidden weights, **W**, into the cell.

$$\mathbf{i} = \sigma(\mathbf{W}^i\mathbf{h}^{(t-1)} + \mathbf{U}^i\mathbf{x}^{(t)}) \tag{3.69}$$

$$\mathbf{f} = \sigma(\mathbf{W}^f\mathbf{h}^{(t-1)} + \mathbf{U}^f\mathbf{x}^{(t)}) \tag{3.70}$$

$$\mathbf{o} = \sigma(\mathbf{W}^o\mathbf{h}^{(t-1)} + \mathbf{U}^o\mathbf{x}^{(t)}) \tag{3.71}$$

$$\mathbf{g} = tanh(\mathbf{W}^g\mathbf{h}^{(t-1)} + \mathbf{U}^g\mathbf{x}^{(t)}) \tag{3.72}$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \circ \mathbf{f} + \mathbf{g} \circ \mathbf{i} \tag{3.73}$$

$$\mathbf{h}_t = tanh(\mathbf{c}_t) \circ \mathbf{o} \tag{3.74}$$

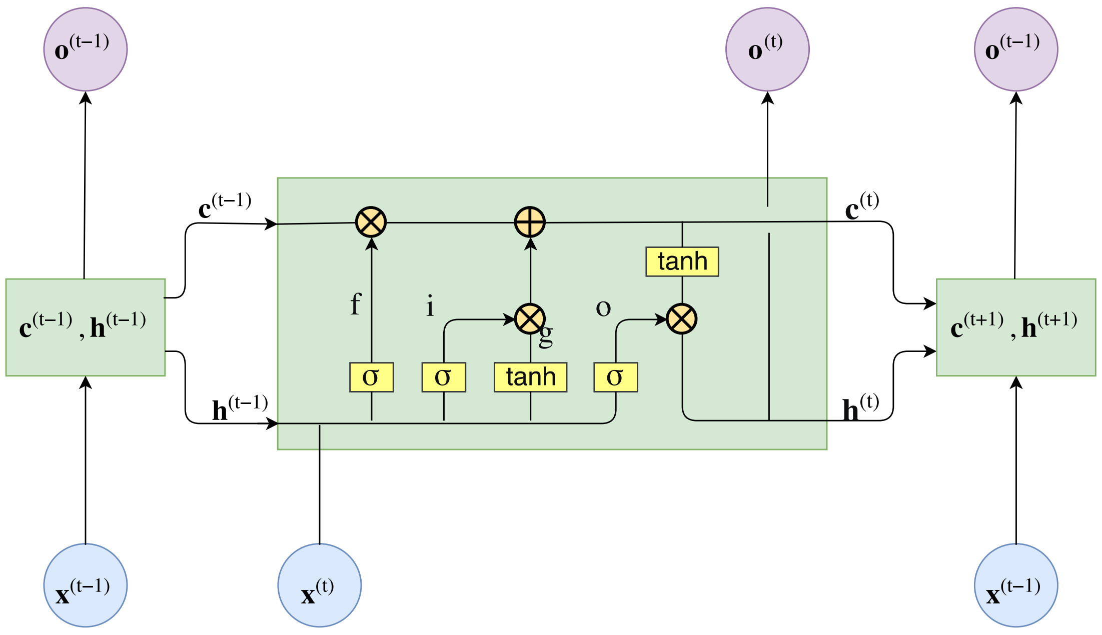


Figure 3.27: Synchronised many-to-many RNN architecture using LSTM Memory Cell.

3.4.4 Training LSTM-RNN using backpropagation through time (BPTT)

Backpropagation is an algorithm for minimising the cost function, which aims to find the optimal weight and bias parameters in the network. We discussed standard backpropagation in Section 3.3.1.2, but that technique is designed for feedforward networks. Hence, this section will give an overview of the backpropagation through time (BPTT) training method and explain how it differs from the standard backpropagation algorithm that uses in a feedforward network. A BPTT is a common technique of training recurrent neural networks, while a backpropagation uses to train feedforward neural network. Both algorithms have the same goal, which is to optimise the weight parameters in the network. The key difference is that BPTT begins with an unfolding process, which converts the recurrent neural network diagram into a multilayer feedforward neural network over t time steps, which depends on the sequence length. After that, the backpropagation algorithm is applied to the unfolded RNNs, which is known as backpropagation through time.

BPTT consists of the two steps of propagation and weight update. The propagation process comprising of two steps including forward propagation and backward propagation. In the forward process of backpropagation, this aims to compute the output values in each time step t given an input sequence, the weight parameters and biases. The calculation of the predicted output at time t, $\mathbf{o}^t$, is defined as Equation (3.62). In the backward process of backpropagation, this aims to calculate the gradients of the error with respect to the parameters. To begin with, the predicted output at time t , $\mathbf{o}^t$, and the actual output at time t, $\mathbf{y}^t$, are used to calculate a loss function or error function, L^t, shown in Equation (3.75):

$$L^t = \frac{1}{2}(\parallel \mathbf{y}^t - \mathbf{o}^t \parallel)^2. \tag{3.75}$$

The total error, L, is the calculated by the summation of error at each time step, L^t, as follow;

$$L = \frac{1}{T}\sum_{t=1}^{T} L^t. \tag{3.76}$$

After getting the error in each output time step and the total error, we can now process the gradients of the error with respect to the parameters. We can then use the chain rule of differentiation as used in feedforward networks. The main difference is that the BPTT sums up the gradient errors at each time step, while the feedforward network is not required to do this because it has no connection between time step.

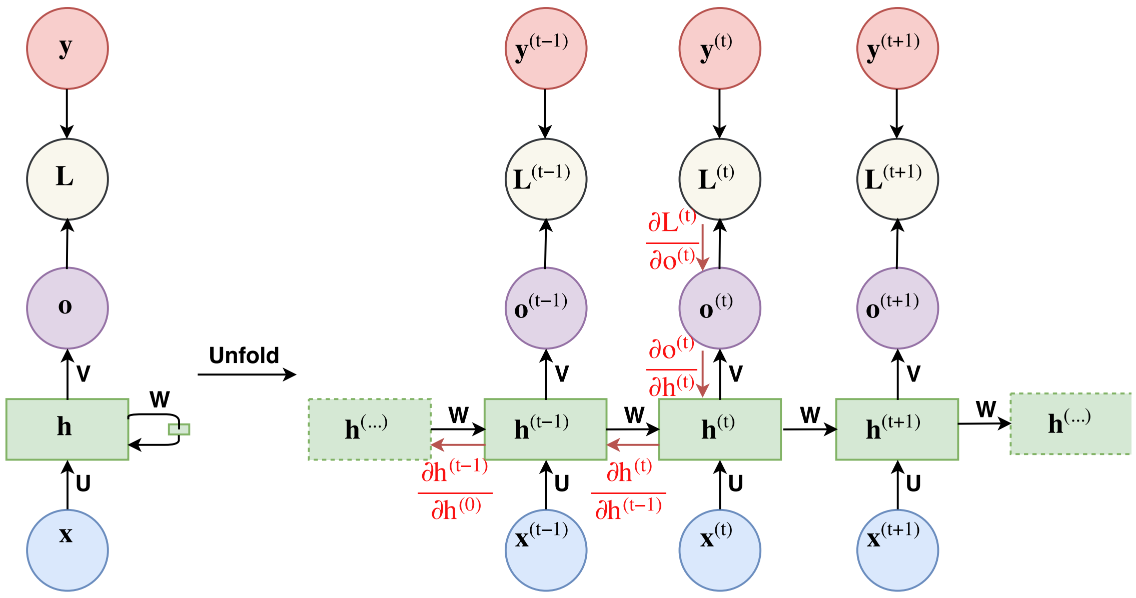


Figure 3.28: An example of how backpropagation throughtime method uses chain rule and applies on a loss funtion at time t (adapted from [46]).

Figure 3.28 shows an example of how the BPTT method using the chain rule algorithm finds a change in $L^{(t)}$ with respect to weight parameter $\mathbf{W}$, called the partial derivative of $L^{(t)}$ with respect to $\mathbf{W}$ can be shown as follow:

$$\frac{\partial L^t}{\partial \mathbf{W}} = \frac{\partial L^{(t)}}{\partial \mathbf{o}^{(t)}} \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{W}}. \tag{3.77}$$

As we know that $\mathbf{h}^{(t)}$ depends on $\mathbf{W}$ and $\mathbf{h}^{(t-1)}$, so chain rule is applied again all the way down to $t = 0$.

$$\frac{\partial L^t}{\partial \mathbf{W}} = \sum_{k=t}^{0} \frac{\partial L^{(t)}}{\partial \mathbf{o}^{(t)}} \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}}. \tag{3.78}$$

From this gradient, it can be seen that the partial derivative of L^t with respect to $\mathbf{W}$, $\frac{\partial L^t}{\partial \mathbf{W}}$, does not depend only on the parameters at the time t, but also depends on the parameters from the previous time step from t to 0. We can then apply this chain rule to every time step and then sum up over all time steps as done for the total error in Equation (3.76).

After a complete forward step and backward step in the propagation process, we move on to the updating weight process. In this step, the gradient errors from the previous step are used by the optimisation method, which aims to minimise the loss function by:

$$\widehat{\mathbf{W}} = \mathbf{W} - \propto * \frac{\partial E}{\partial \mathbf{W}}, \tag{3.79}$$

where $\widehat{\mathbf{W}}$ is the updated weights, $0 \leq \propto \leq 1$ is the learning rate parameter, which is used to control how quickly the network learns. The learning can be too slow if the learning rate is set too low, and the weights can oscillate if learning rate is set too high.

3.4.4.1 Epochwise and truncated backpropagation through time

There are two different backpropagation through time (BPTT) approaches related to the calculation of gradient errors and updating weights: epochwise BPTT and truncated BPTT. The main difference between these two approaches is that epochwise back-propagation through time computes the gradient error at all time steps in each sentence from the start time to the end time of the sequence, denoted as $L^{(t_0, t_\tau)}$, where τ is the sequence lenght. Figure 3.29 shows how epochwise BPTT computes the gradient errors of each time step.

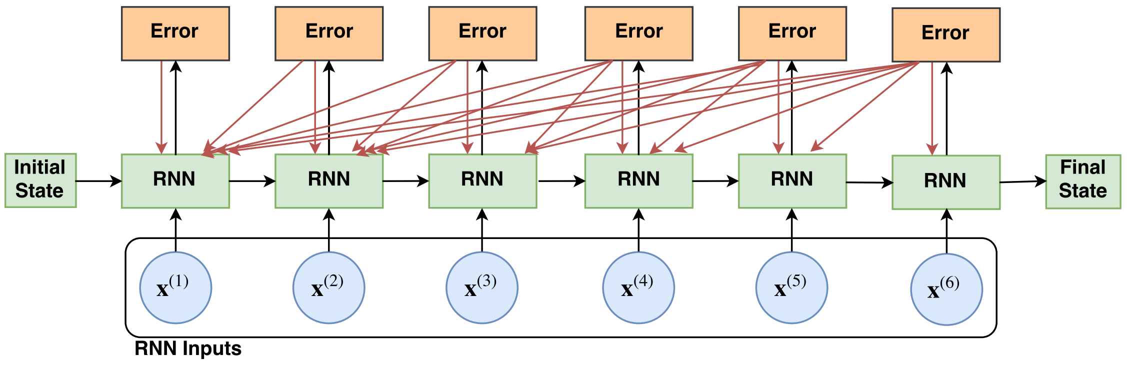


Figure 3.29: An example of how epochwise backpropagation looks on a sequence of length 6.

To do this, epochwise BPTT requires more memory because it has to save the entire input sequence to the network, the network state, the target output and also the gradient error of the interval $[t^0, t^\tau]$. To overcome this problem, truncated BPTT uses a fixed length t^{trunc} to alleviate this problem. The truncated BPTT instead computes the error from the start time to the fixed time instead, denoted as

$L^{(t^0, t^{trunc})}$. Noted that $t^{trunc} < t^{\tau}$. One additional process of truncated BPTT is it has to propagate the network state to the next block if the current sequence has not finished, otherwise the state is set to zero. Figure 3.30 shows how truncated BPTT computes the same sequence of Figure 3.29 with fixed truncation to 3-time steps.

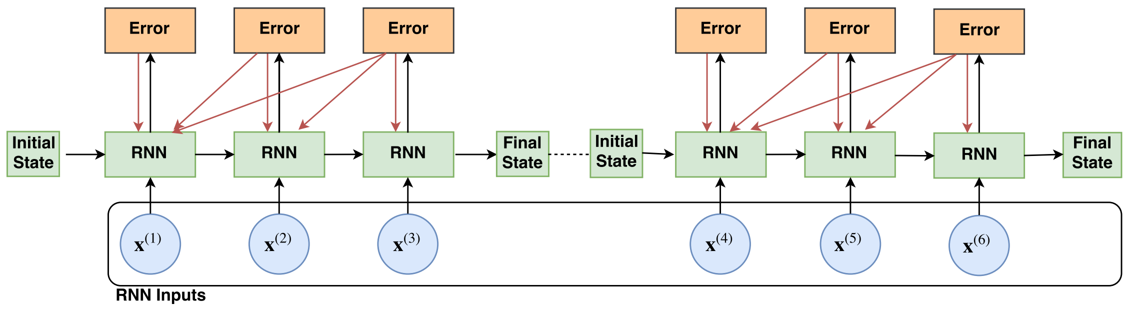


Figure 3.30: An example of how truncated backpropagation looks on a sequence of length 6 with 3 truncation steps.

It can be seen that truncated BPTT is more practical than epochwise BPTT in the case of a long sequence. However, one of the limitations of truncated BPTT is its ability to get the information from both the past and future from the bidirectional networks. As shown in Figure 3.30, the first truncation block cannot see the information from the second block. Additionally, both approaches are based on global sequence dependency. This makes the mini-batch training less efficient because it is required to see and process the whole sequence in order.

Chapter 4

Data corpora

4.1 Introduction

This chapter describes the audiovisual speech dataset that was used for training and testing the models in this work, namely KB-2k. We used the KB-2k dataset because it is a large audiovisual speech dataset and is designed for visual speech synthesis. There are many publicly large audiovisual dataset such as GRID [17], TCD_TIMIT[53], but they are designed for audiovisual speech recognition. The major limitations of audiovisual speech recognition datasets are limited-domain (i.e. for digit recognition) and/or they contain a lot of speech from different speakers but not many examples for each speaker. Conversely, for visual speech synthesis, it is important to have a lot of recordings from a single person. Additionally, we found that most publicly available audiovisual datasets for visual speech synthesis have limited vocabulary and are too small for modern machine learning approaches. For example, LIPS [106] corpus contains only 279 sentences, whereas our dataset contains 2,542 sentences. This chapter begins with the overview of the KB-2k dataset in terms of data collection, recording conditions, acoustic data, and acoustic (phoneme) units. Then, we describe the visual feature extraction that we use to parameterise the facial motion in KB-2k. Finally, the three additional modules regarding the new basic units and data pre-processing are described including: (i) a novel visual units named dynamic viseme units, (ii) visual features normalisation, and (iii) syllabification of sound form.

4.2 Overview of KB-2k dataset

A set of 2,542 phonetically balanced TIMIT sentences of audiovisual speech dataset from Disney Research named KB-2k was used in this work [98]. The KB-2k dataset contains speech from a professional male actor with clear articulation and standard US accent. The recordings were spoken at a normal speaking rate with no emotion. The dataset contains both frontal and side views of the actor, but only the frontal view was used in this work. The full high definition video was captured on a Sony DMW-EX3 professional video camera at 29.97 frames per second (fps) with 1080p resolution according to the progressive scan mode. The recordings of the dataset were made in the same lighting conditions. The total video length is approximately 8 hours and contains about 260,000 frames of speech. The average sentence length is about 5 seconds. The following sentences show the shortest and longest sentences: "they were shattered" and "to date the one meat showing favorable results at sterilization doses is pork". Figure 4.1 shows example images from the KB-2k dataset.

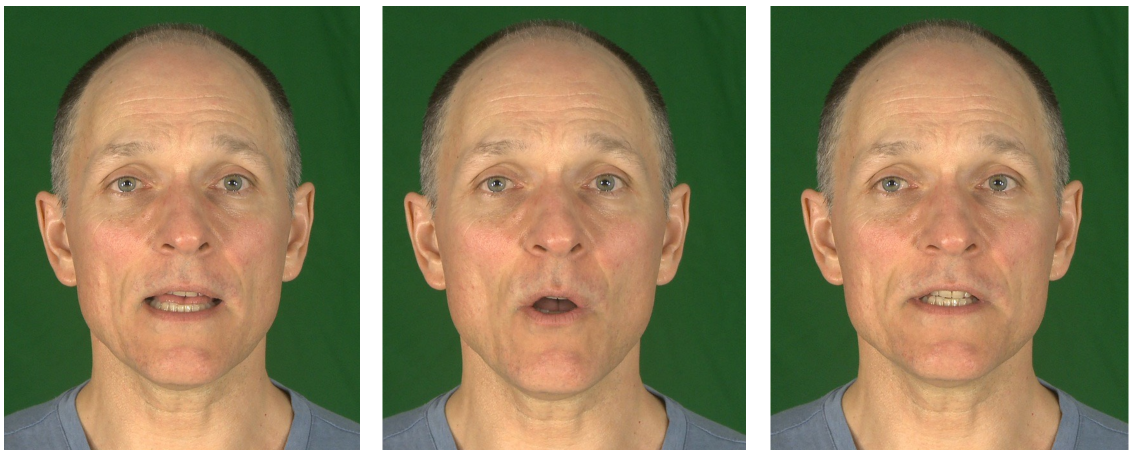


Figure 4.1: Example images from the KB-2k dataset.

The acoustic speech signal was recorded at a sample rate of 48 kHz with 16 bits per sample in the form of WAV files. Although, the natural speech is used only for combination with the visual speech in this work. Each word was phonetically transcribed using the Carnegie Mellon University pronunciation dictionary into the ARPAbet phoneme set. The IPA and ASCII symbol for the consonants, vowels and diphthongs used in the ARPAbet phoneme set are shown in Table 2.1. Manual annotation was carried out to obtain phonetic label boundaries after applying word to phoneme conversion. Figure 4.2 shows an example of the waveform and label file from the KB-2k dataset. The second and third panes show the phoneme (PH) and word boundaries (WRD), respectively.

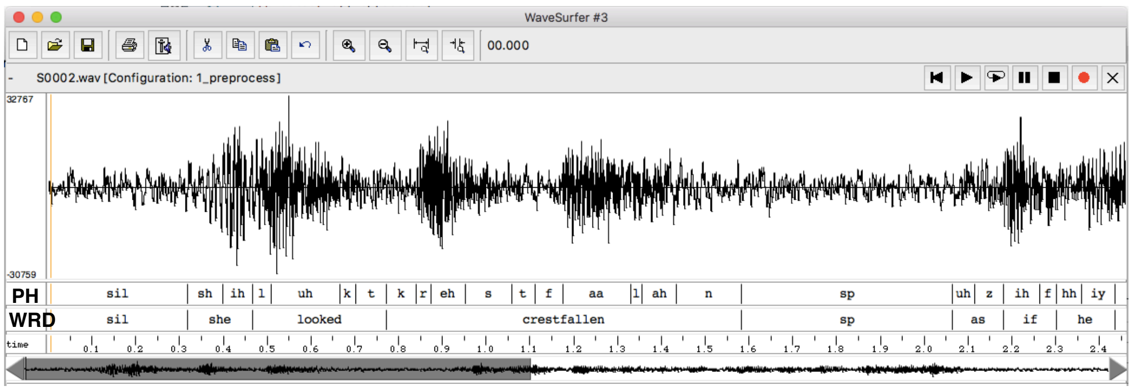

WaveSurfer #3
00.000
S0002.wav [Configuration: 1_preprocess]
32767
-30759
PH
sil sh ih l uh k t k r eh s t f aa l ah n sp uh z ih f hh iy
WRD
sil she looked crestfallen sp as if he
time
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4

Figure 4.2: Example of the waveform and label file from the KB-2k dataset. The PH pane denotes for phoneme boundary and WRD pane denotes for word boundary.

4.3 Visual processing of KB-2k dataset

The visual speech parameters were extracted from the the visible speech articulators (e.g. face) in each frame using an active appearance model (AAM) [20, 69]. Active appearance models (AAMs) provide a low dimensional representation of the shape and appearance of the speech articulators and they are also generative and so can re-synthesise near photo-realistic images of faces [20, 69]. In this work, the lower face including jaw and lips was used instead of the whole face because the other parts, such as nose, eyes, eyebrows, can sometimes introduce noise to the AAM models. Also, we make the assumption that they are not speech related.

The shape component, $\mathbf{s}$, of the AAM is a set of facial images with the two-dimensional vertices of a mesh. For each labelled image, the shape vector is represented by concatenating the (x, y) coordinates of each landmark.

$$\mathbf{s} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n, \}, \tag{4.1}$$

Where (x_i, y_i) are the x and y coordinates of the landmark i and n denotes the number of landmarks. This KB-2k database used 34 landmarks, $n = 34$, to capture the contour of jaws, inner and outer lips, as shown in Figure 4.3.

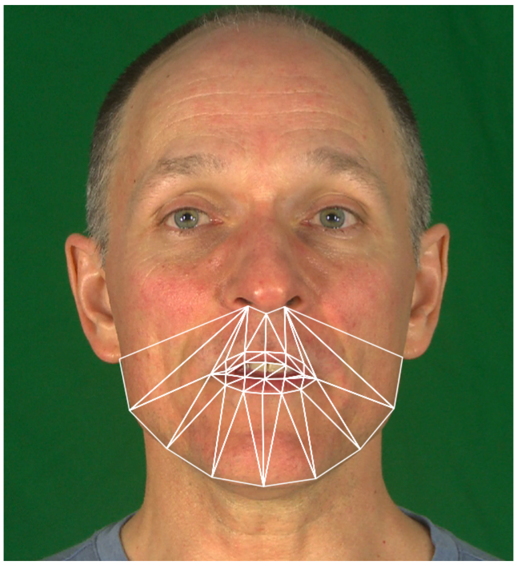


Figure 4.3: An example of a hand-labelled lower face image of a shape component [98]

.

Each training image was then alingned to the same translation, scale and rotation to ensure that they represent in the same coordinate system in all images using Procrustes analysis [22]. An illustration of unaligned and aligned shape images is shown in Figure 4.4.

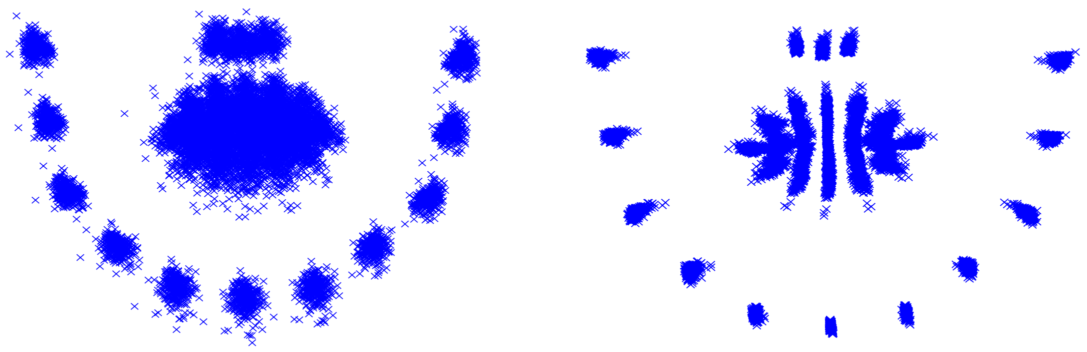


Figure 4.4: A comparison of an unaligned (left) and aligned (right) shape landmarks. Each cross represents the x and y coordinates of each landmark.

These aligned shape vector then model the variation of the shape using a linear model of the form:

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{P}_s \mathbf{b}_s, \tag{4.2}$$

where $\mathbf{s}$ is a vector of (x, y) coordinates of the shape vertices. $\bar{\mathbf{s}}$ is the mean shape from the training data, $\mathbf{P}_s$ are the modes of shape variation, and $\mathbf{b}_s$ a vector of shape parameters that encode the shape. For KB-2k, Five modes, $m = 5$, were used to capture 95% of the shape variation.

The appearance component, $\mathbf{a}$, of an AAM is the pixel intensities of the image inside the shape boundary. PCA is then applied on the shape normalised pixel intensities, giving a compact, linear model of appearance variation of the form:

$$\mathbf{a} = \bar{\mathbf{a}} + \mathbf{P}_a \mathbf{b}_a, \tag{4.3}$$

where $\mathbf{a}$ is the resulting image. $\bar{\mathbf{a}}$ is the mean appearance image, $\mathbf{P}_a$ are the modes of appearance variation containing n eigenvectors, and $\mathbf{b}_a$ a vector of appearance parameters. To more compactly represent the face and to better model the non-linear relationship between facial regions, multi-segment AAMs can be used rather than the single-segment AAMs [107]. This technique aims to model independent appearance components. From the KB-2k dataset, two-segment AAMs were used with the first region containing the lower face without inner mouth, $\mathbf{a}^1$, and the second region containing the inner mouth pixels, $\mathbf{a}^2$, as shown in Figure 4.5. Note that, $\mathbf{P}_a^1$ and $\mathbf{P}_a^2$ contain $n1 = 46$ and $n2 = 10$ modes to describe 95% of the lower face without inner mouth and inner mouth, respectively.

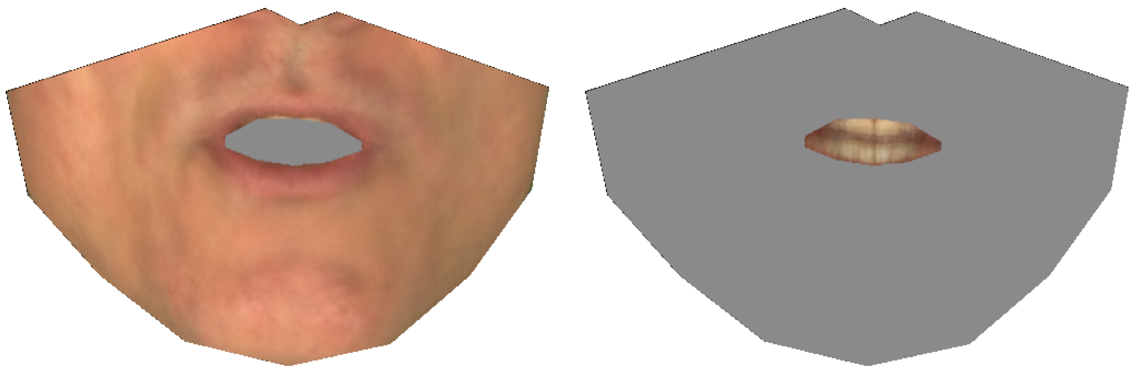


Figure 4.5: An example of two-segment AAMs.

The original shape and appearance parameters, in both regions are concatenated and normalised into the same unit scale between appearance intensity and shape parameters as follows,

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{W}_a \mathbf{b}_a^1 \\ \mathbf{b}_a^2 \end{pmatrix}, \tag{4.4}$$

where $\mathbf{b}_s$ is the vector of shape parameters that encodes the shape and $\mathbf{b}_a^1$ and $\mathbf{b}_a^2$ are vectors of appearance parameters from the two multi-segment model (e.g. inner mouth and the rest). $\mathbf{W}_s \in \mathbb{R}^{m \times m}$ is a diagonal weight matrix to correct the difference in units between $\mathbf{b}_s$ and $\mathbf{b}_a^2$ which has the same entry on the main diagonal and zero elsewhere. $\mathbf{W}_a \in \mathbb{R}^{n_1 \times n_1}$ denotes a diagonal weight matrix to correct the difference in units between $\mathbf{b}_a^1$ and $\mathbf{b}_a^2$ which has the same entry on the main diagonal and zero elsewhere. These two weight are calculated as follows, where $\sigma^2_{b_{s_i}}$, $\sigma^2_{b^1_{a_i}}$ and $\sigma^2_{b^2_{a_i}}$ represent the variance in each dimension of the shape and appearance models,

$$\mathbf{W}_s = \sqrt{\frac{\sum_{i=1}^{n_2} \sigma^2_{b^2_{a_i}}}{\sum_{i=1}^{m} \sigma^2_{b_{s_i}}}}, \mathbf{W}_a = \sqrt{\frac{\sum_{i=1}^{n_2} \sigma^2_{b^2_{a_i}}}{\sum_{i=1}^{n_1} \sigma^2_{b^1_{a_i}}}}. \tag{4.5}$$

To remove redundancy in the shape and appearance components of the model, $\mathbf{b}$, a third PCA to give a combined model of the shape and appearance variation, as follows;

$$\mathbf{b} = \mathbf{Q}\mathbf{y}, \tag{4.6}$$

where $\mathbf{Q}$ are the eigenvectors, and $\mathbf{y}$ is a vector of AAM parameters controlling the combined shape and appearance parameters. Note that, there is no mean of combined parameters in this expression because the shape and appearance parameters have zero mean. For KB-2k, the final model requires 30 modes to account of 98% of the total variance, as shown in Table 4.1. The first three modes of variation for a combined shape and two appearance models built from the speaker in the KB-2k corpus are shown in Figure 4.6. It can be seen that the first mode represents the opening and closing mouth with spreading lips, while the second mode seems to represent the same thing with neutral lips.

Table 4.1: *A summary of AAM visual features of KB-2k corpus.*

Features	Values
Video data rate (fps)	29.97
Number of face pixels	2073600
Number of AAM ($\mathbf{y}$)	30
Number of shape ($\mathbf{b}_s$)	5
Number of appearance-outer ($\mathbf{b}_a^1$)	46
Number of appearance-inner ($\mathbf{b}_a^2$)	10

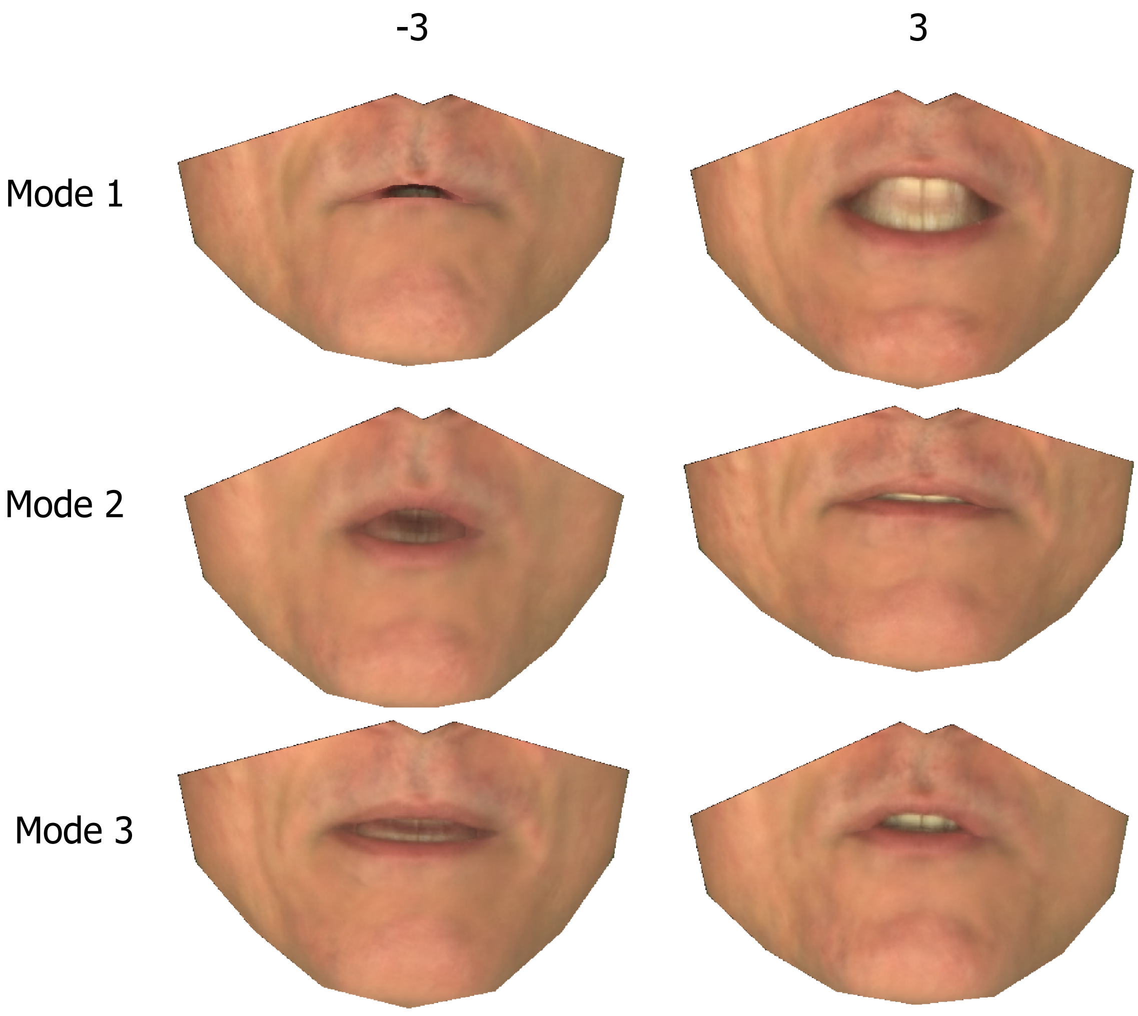


Figure 4.6: The first three modes of a combined shape and appearance model at 3 standard deviations (right) and -3 standard deviations (left) from the mean.

To reconstruct an image from a given set of AAM parameters, $\mathbf{y}$, it can be calculated as follows,

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{P}_s \mathbf{W}_s^{-1} \mathbf{Q}_s \mathbf{y}, \tag{4.7}$$

$$\mathbf{a}^1 = \bar{\mathbf{a}}^1 + \mathbf{P}_a^1 \mathbf{W}_a^{-1} \mathbf{Q}_a^1 \mathbf{y}, \tag{4.8}$$

$$\mathbf{a}^2 = \bar{\mathbf{a}}^2 + \mathbf{P}_a^2 \mathbf{Q}_a^2 \mathbf{y}, \tag{4.9}$$

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_s \\ \mathbf{Q}_a^1 \\ \mathbf{Q}_a^2 \end{pmatrix}, \tag{4.10}$$

where $\mathbf{Q}_s$ is a matrix of the first m largest eigenvectors corresponding to the shape model, $\mathbf{Q}_a^1$ is the next $n1$ eigenvectors corresponding to the first appearance model, and $\mathbf{Q}_a^2$ is the next $n2$ eigenvectors corresponding to the second appearance model. The final image is then computed by a forward piecewise affine warping of the two appearances to the shape landmarks.

4.4 Dynamic viseme units of KB-2k dataset

Dynamic visemes were recently introduced as units of visual speech, equivalent to the phonemes of acoustic speech. They represent groups of similar **lip-motions** instead of groups of similar acoustic sounds for phonemes or a **static mouth shape** for traditional visemes. A set of dynamic visemes is learned automatically by clustering visual speech parameters in a database. This section will describe our replication of dynamic viseme units pioneered in Taylor et al. [98]. The framework of dynamic viseme classifier system can be divided into two parts: training and testing. In the training part, dynamic viseme units are learned automatically based on visual speech parameters. In the testing part, a sequence of dynamic viseme classes is predicted from AAMs sequences using the trained models.

4.4.1 Training part

This section describes the training part of how to build dynamic viseme units. We focus on how to identify and cluster gesture from an audiovisual speech database, namely KB-2K, as shown in Figure 4.7.

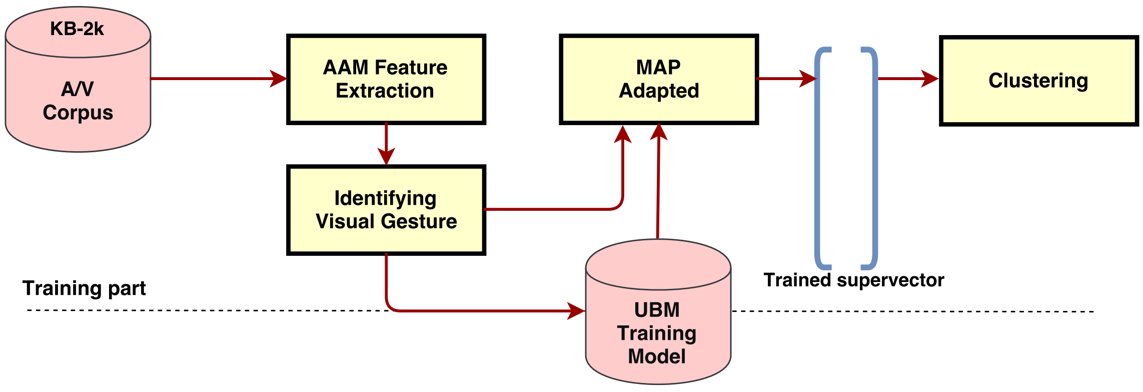


Figure 4.7: The framework of dynamic viseme training system.

4.4.1.1 Identifying Visual Gestures

This section aims to identify a movement of the visible articulators in the video, aka a visual gesture sequence. To segment gestures, the image sequences from the training video are first projected onto an AAM to give a low-dimensional representation of the visual features. KB-2k is parameterised by a 30 dimensional AAM as described in Section 4.3. After that, it is assumed that when a person is speaking, the articulators slow down to reach an articulatory target and begin to accelerate towards the next target. Hence, gesture boundaries, g, can be defined as the position where articulators change direction. The velocity (delta) of the AAM parameters in each dimensional are computed to find the rate of change as follow:

$$\Delta\mathbf{y}_t = \frac{\mathbf{y}_{t+1} - \mathbf{y}_{t-1}}{2}, \tag{4.11}$$

where $\Delta\mathbf{y}_t$ is a velocity vector at frame t, $\mathbf{y}_t$ represents a vector of 30 AAM parameters at frame t, and $1 \leqslant t < T$, T is a total number of frames in a sequence of $\mathbf{y}$. Then, the magnitude of the velocity coefficient is computed to find the speed of change as follow:

$$dm_t = \sqrt{\sum_{i=1}^{M=30} (\Delta\mathbf{y}_t^i)^2}, \tag{4.12}$$

$$\Delta dm_t = \frac{dm_{t+1} - dm_{t-1}}{2}, \tag{4.13}$$

where dm_t is the magnitude (speed) of velocity at time t, M is number of AAM parameters, and y^i is the i-th coefficient of the velocity at frame t. As an example, Figure 4.8, shows the derivative of **dm**, $\boldsymbol{\Delta}$**dm**, on a blue line, for the utterance "resistance thermometers". Where $\boldsymbol{\Delta}$**dm** crosses from negative to positive, this indicates a change of articulation and hence a change of dynamic viseme. These are illustrated with red dots, and provide dynamic viseme (gesture) boudaries. The lower tier with green boxes shows the phoneme boundaries and the upper tier with the purple boxes mark the gesture boundaries. It can be seen that the gesture boundaries (dynamic viseme) do not align with acoustic (phonemes) and all visual gestures denote as g.

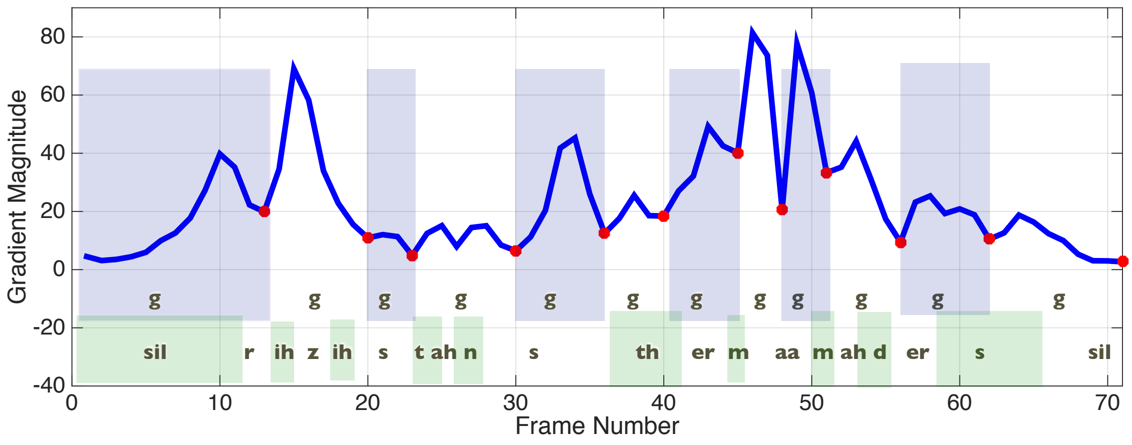


Figure 4.8: An illustration of the gradient magnitude in AAM parameter space (blue line) corresponds to phoneme and dynamic viseme boundaries from the sequence "resistance thermometers". Where the red dotted lines with purple boxes reflect dynamic viseme boundaries and the bottom green boxes display phonemes and its boundaries.

4.4.1.2 Clustering Visual Gestures

The segmented visual gestures from the previous section are clustered into visually similar classes and the number of dynamic viseme classes is determined in advance. Note that one cannot directly apply traditional clustering technique because the sequence lengths of the gestures are different. The initialisation process is an attempt to represent different gestures or varying length in a fixed length vector. There are many approaches to convert varying length representations into a fixed length representation. For example, Hilder et al. [54] measured the similarity between different length gestures using a dynamic time warp (DTW). In addition, hidden Markov model (HMM) supervectors are also applied to non-equal lengths of gesture [98]. They noted that the HMM supervector technique is better than the DTW approach because HMM supervectors tend to generate better visual clusters.

To generate HMM supervectors [11], a three-state HMM universal background model (UBM) is learned from all of the gestures, $\mathbf{g}$, in the corpus. Each state j is represented as a multivariate Gaussian mixture model (GMM), $\boldsymbol{\zeta}_j(\boldsymbol{g})$, as follow:

$$\boldsymbol{\zeta}_j(\mathbf{g}) = \sum_{k=1}^{K} w_{jk} N(\mathbf{g}; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}), \tag{4.14}$$

where j is the HMM state, K is the number of modes, w_{jk} is the weight of the k-th modes in the j-th HMM state. $N(\mathbf{g}; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$ represents as a multivariate Gaussian with mean, $\boldsymbol{\mu}_{jk}$, and covariance, $\boldsymbol{\Sigma}_{jk}$. For a given sequence of AAM vectors for gesture $\boldsymbol{g}$, there are decoded into a state sequence in the UBM. According to the state sequence, the state of the UBM are adapted by the AAM vectors that have been allocated. Then, the variable length gestures are transformed into a fixed-length maximum a posteriori (MAP) adapted mean vector by updating the UBM using MAP adaptation [64];

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{N_{jk}}{N_{jk} + \tau} \bar{\boldsymbol{\mu}}_{jk} + \frac{\tau}{N_{jk} + \tau} \boldsymbol{\mu}_{jk}, \tag{4.15}$$

where $\hat{\boldsymbol{\mu}}_{jk}$ represents the MAP updated mean of the j-th state and k-th mode, $\boldsymbol{\mu}_{jk}$ denotes the mean of the j-th state and k-th mixture in the UBM, and $\bar{\boldsymbol{\mu}}_{jk}$ is the mean of the AAM vector allocated to state j and mode k. τ is a weight that controls the influence of the adapted data where τ is set to 10 for the MAP estimation in this work, N_{jk} is the occupation likelihood of the adaptation data in state j and mode k.

Next, the HMM supervector for each gesture, $\mathbf{G}_s$, is computed as the difference between the UBM mean vectors, $\boldsymbol{\mu}_j$, and the MAP adapted mean vectors, $\hat{\boldsymbol{\mu}}_j$ in each mixture k model:

$$\mathbf{G}_s = \begin{bmatrix} \mu_{11-}\hat{\mu}_{11} \\ \mu_{22-}\hat{\mu}_{22} \\ \cdots \\ \mu_{jk-}\hat{\mu}_{jk} \end{bmatrix}, \tag{4.16}$$

where $\mathbf{G}_s \in \mathbb{R}^{J \times K \times D}$, J is the number of HMM state, K is the number of mixture and D is the number of AAM features. In this work, we created three-state hidden Markov models with each state modeled by a single Gaussian mixture model in a left-to-right model with no skip. Note that, these training conditions were the same as we found in [98].

A graph-patitioning-based clustering algorithm is used to cluster all of the HMM supervectors [60]. This continues to split the graph into two groups until it reaches the specific number of k-clusters using a min-cut graph partitioning algorithm. It can be computed using a traditional single-link criterion function. Taylor et al. [98] noted that graph-based clustering is able to group similar gestures better than k-means clustering. An example of selected dynamic visemes from the graph-based clustering is depicted in Figure 4.9. It can be seen that "V29" represents lip funnel, "V36" denotes mouth stretch. More interactive examples can be found in http://oaom.openservice.in.th/DV/index.html.

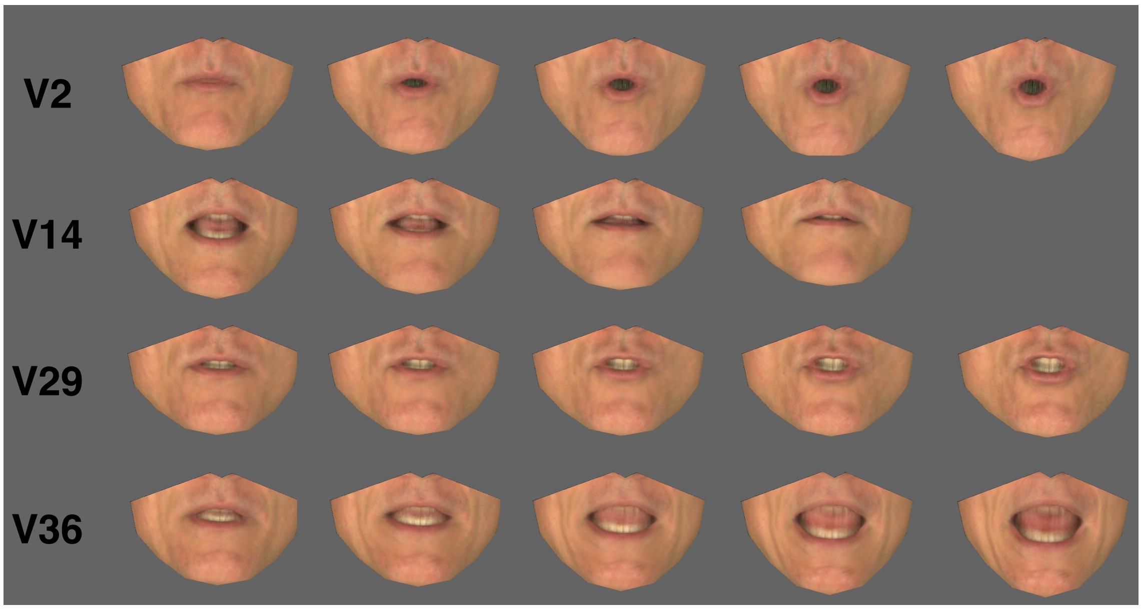


Figure 4.9: An example of selected dynamic visemes from the graph-based clustering. Each viseme represents a different visual function. For example "V29" represents lip funnel and "V36" denotes mouth stretch.

To determine how many dynamic visemes classes are required, we then do this manually by looking at the compactness captured by the different classes of dynamic visemes. Cluster compactness is used as a measure of clustering quality, where the compactness is the sum of the distances of the supervectors assigned to a cluster to the respective cluster centroid (median). Figure 4.10 shows the cluster compactness as a function of the number of clusters between 5 and 600. It can be seen that it is difficult to conclude which is a reasonable number of dynamic visemes. In this work, hence, we will consider the analysis-synthesis approach. This approach determines this empirically by testing synthesisers trained using a different number of dynamic visemes classes, and then select the number that achieves the highest performance on a validation set. More details are given in Section 5.3.1.4. It can be concluded that a reasonable number of dynamic visemes is approximately 160 viseme classes .

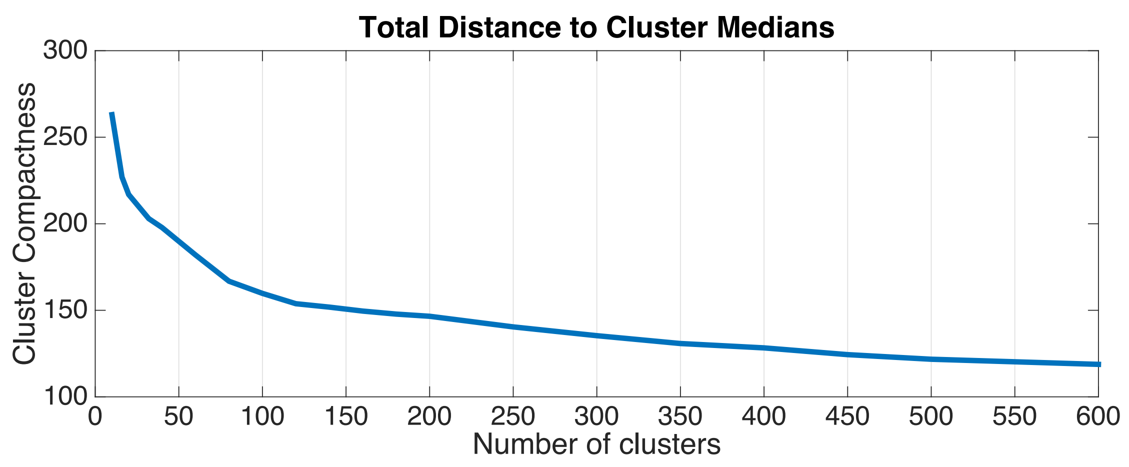


Figure 4.10: The cluster compactness as a function of the number of clusters computed from the HMM supervectors extracted from the KB2K corpus.

4.4.2 Testing part

This section describes the testing part of how to classify dynamic viseme units. We focus on how to determine the dynamic viseme class from a sequence of AAM parameters in the testing set, as shown in Figure 4.11.

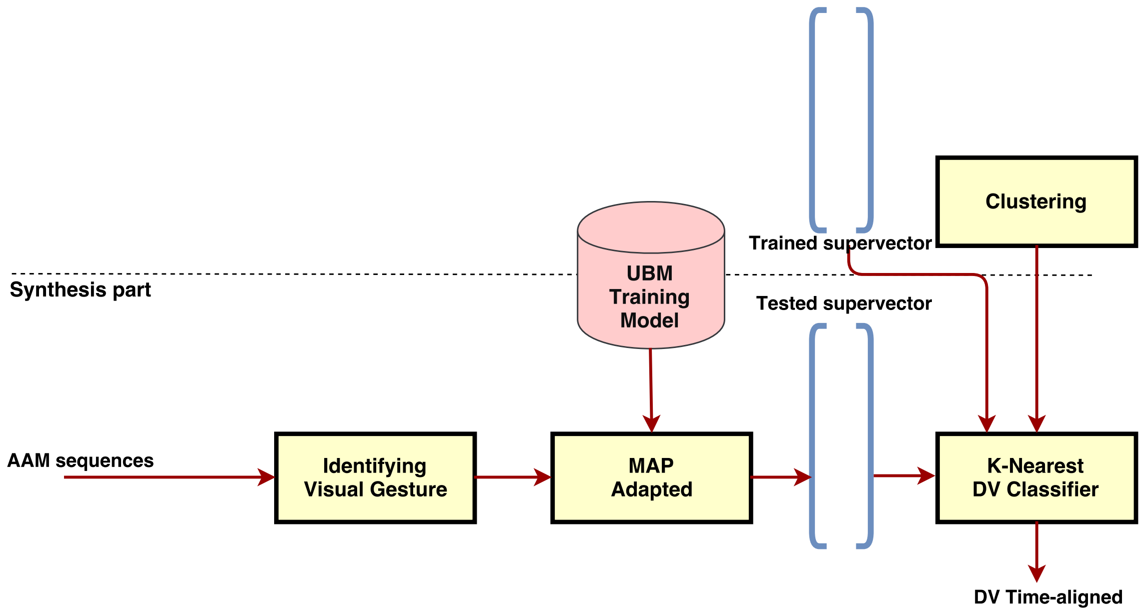


Figure 4.11: The framework of dynamic viseme classifier system.

4.4.2.1 Determining Visual Gestures

This section describes how to predict dynamic viseme classes from a sequence of AAM parameters in the testing set. In the first step, a sequence of AAM parameters are segmented into gesture sequences, and then converted to supervectors using the same segmentation and conversion modules, as described in the training process. Then, the k nearest neighbour algorithm is used to classify each unlabelled test supervector into a dynamic viseme class. Generally, we can use various distance metrics to find the distance between a test supervector, $\hat{\mathbf{g}}$, and the supervector in the training data, $\mathbf{g}_j$ ($1 \leqslant j \leqslant M$ where M is number of training supervectors). Note that the Euclidean distance is used in this work, as follows:

$$d\left(\hat{\mathbf{g}}, \mathbf{g}_j\right) = \sum_{d=1}^{D} \parallel \hat{g}_d - g_{jd} \parallel_2, \tag{4.17}$$

$$j^* = \underset{j}{argmin}\ d\left(\hat{\mathbf{g}}, \mathbf{g}_j\right) \tag{4.18}$$

where D denotes the number of element in the supervector and $d(\hat{\mathbf{g}}, \mathbf{g}_j)$ is the Euclidean distance between each training supervector, $\mathbf{g}_j$, and the test supervector, $\hat{\mathbf{g}}$. We then select the j^*-th training supervector with the shortest distance. Finally the dynamic viseme class of $\hat{\mathbf{g}}$ is simply assigned from the class of g_{j^*}.

4.5 Data preparation of KB-2k dataset

This section describes two additional modules of KB-2K's preprocessing. Firstly, feature normalisation is used to scale the AAM features into the same range. Secondly, syllable segmentation is used to find a syllable's boundaries in the case where the pronunciation of the words is available.

4.5.1 Feature normalisation

Since the range of each dimension of features varies widely, this would make some methods not work efficiently. For example, it reported that a large difference in the variance of feature dimensions is a reason of slow training and getting stuck in local optima during gradient descent. Therefore, the range of all features must be scaled or normalised before training the models.

This is an important concern for our AAM features on KB-2k dataset because the scale of each dimension is very different. Figure 4.12 shows the examples of AAM feature order 1, 10, 20, and 30. Each box denotes the first quartile (Q1) to the third quartile (Q3). It can be seen that, the medians of each AAM parameter (identified by the red line within the box) are all at the similar level but they represent with different distributions. In the case of non outliers (identified by red signs), for example, the range of 1st parameter is approximately between -150 and 200, whilst the 30th AAM feature is between -12 and 12.

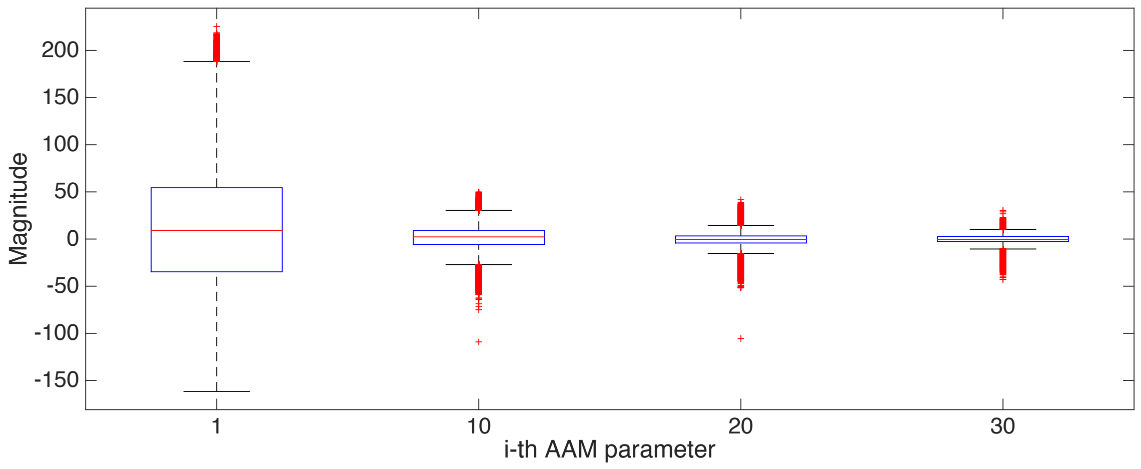


Figure 4.12: Examples of the range of AAM features order 1, 10, 20, and 30.

There are two common techniques used to adjust the range of parameters: feature scaling and standardisation. Feature scaling or min-max scaling aims to scale the values of original data to the fixed range of $[0, 1]$ or $[-1, 1]$ with smaller standard deviations. Feature standardisation or z-score normalisation aims to standardising the features into z-score. The new data is centered around 0 with a standard deviation to 1. Equations (4.19) and (4.20) show how to calculate feature scaling, $\mathbf{z}_m$, and standardisation, $\mathbf{z}_s$, respectively.

$$\mathbf{z}_m = \frac{\boldsymbol{y} - \boldsymbol{y}_{min}}{\boldsymbol{y}_{max} - \boldsymbol{y}_{min}}, \tag{4.19}$$

$$\mathbf{z}_s = \frac{\boldsymbol{y} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \tag{4.20}$$

where $\mathbf{y}$ is the original features, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ is the mean and standard deviation of the original features, respectively.

For our visual speech synthesis application, we prefer z-score normalisation over min-max scaling because we would like to maximise the variance rather than limiting the output data between maximum and minimum values in the min-max scaling method.

4.5.2 Syllable segmentation

The syllable is one of the basic units of speech, and can be used to describe a sequence of speech sounds. Generally, every syllable has one vowel sound and the number of vowel in a sequence equals the number of syllables. For example, "/let.ər/" has two vowels (/e/ and /ə/) and "/kæt/" has one vowel (/æ/). The syllable structure, σ, comprises of two components: onset, O, and rime, R, as shown in Figure 4.13. The onset is usually a consonant or cluster consonant. The right branch of the syllable structure is rime, which consists of nucleus, N, and coda, C. Nucleus is usually vowel and optional coda refers to the consonant that following the nucleus.

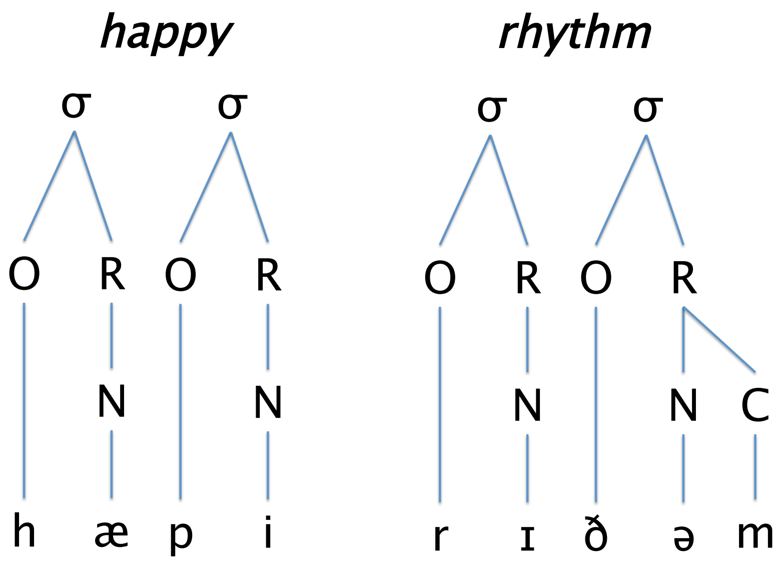


Figure 4.13: Syllable structure for happy and rhythm.

Syllabification is the process of syllable segmentation, which comprises of two types: syllabification of letters and syllabification of phonemes. For example, a syllabification of a word "happy" is "hap-py" which normally uses a hyphen to separate each syllable, and a syllabification of a sequence of phonemes "/hæpi/" is "/hæ.pi/" which normally uses a dot to divide each syllable. Figure 4.13 depicts the syllable structure for the word "happy" and "rhythm".

This work focuses only on the syllabification of phonemes and describes how to identify the syllable boundaries in sound form. These syllable boundaries will be used to represent the input features in our speech animation model (Section 7.2.1), as we have made an assumption that syllable information, such as number of syllables in a word and the number of phonemes in a syllable, affects the way people speak. For the syllable marker, we use simple syllabification rules including;

- finding the vowels and diphthongs (nucleus) by scanning from left to right and form a syllable node from them.
- For each syllable node, link onsets (consonants) to the left of the nucleus
- For each syllable node, link codas (consonants) to the right of the nucleus
- For the ambiguous consonants that represent as both onset and coda;
  - adding syllable marker after coda in the case of stressed nucleus and short vowel, for example "/lev.əl/" instead of "/le.vəl/"
  - adding syllable marker after nucleus in the case of unstressed nucleus and short vowel, for example "/bɪ.liv/" instead of "/bɪl.iv/"
  - adding syllable marker after nucleus in the case of stressed nucleus and long vowel, for example "/pli.zɪŋ/" instead of "/pliz.ɪŋ/"

4.5.3 Training/validation/testing dataset

All subsequent experiments in this thesis are performed on the KB-2k audiovisual speech dataset which contains 2,542 phonetically balanced utterances from TIMIT totalling around 8 hours. We randomly partition this dataset into four parts: a training set (80%), a validation set (10%), a testing set (10%) and a special testing set (2%). Note that the special testing set is selected as the same 50 held-out utterances which were used in Taylor's work [98] and are a subset of the testing set. Table 4.2 summaries key characteristics of the database. We know that having mismatched training and test sets makes it harder to improve the performance of the synthesiser. Hence, we show the phoneme distribution of each set in Figure 4.14. It can be seen that each set especially the development and test sets come from the similar distributions.

Table 4.2: *A summary of KB-2k corpus.*

Features	Training	Validation	Testing	SpecialTesting
Total number of utterances	2,042	250	250	50
Total number of unique PH	41	41	41	41
Total number of unique SV	14	14	14	14
Total number of unique DV	160	160	160	160
Average number of PH per syllables	2.55	2.57	2.55	2.57
Average number of PH per word	3.94	3.94	3.97	3.97
Average number of syllables per word	1.54	1.53	1.55	1.54
Average number of PH per phrase	31.56	30.84	31.64	30.94
Average number of DV per phrase	15.24	15.25	14.99	14.48
Average number of syllables per phrase	12.38	11.97	12.34	12.02
Average number of words per phrase	7.97	7.84	7.95	7.74
Average number of PH per utt	35.96	36.42	34.99	33.32
Average number of syllables per utt	14.51	14.64	14.01	13.16
Average number of words per utt	9.58	9.77	9.16	8.62
Average number of phrases per utt	1.26	1.31	1.2	1.14
Average number of PH per DV	2.68	2.66	2.68	2.72
Average number of syllables per DV	1.42	1.42	1.41	1.4
Average number of words per DV	1.28	1.28	1.27	1.28

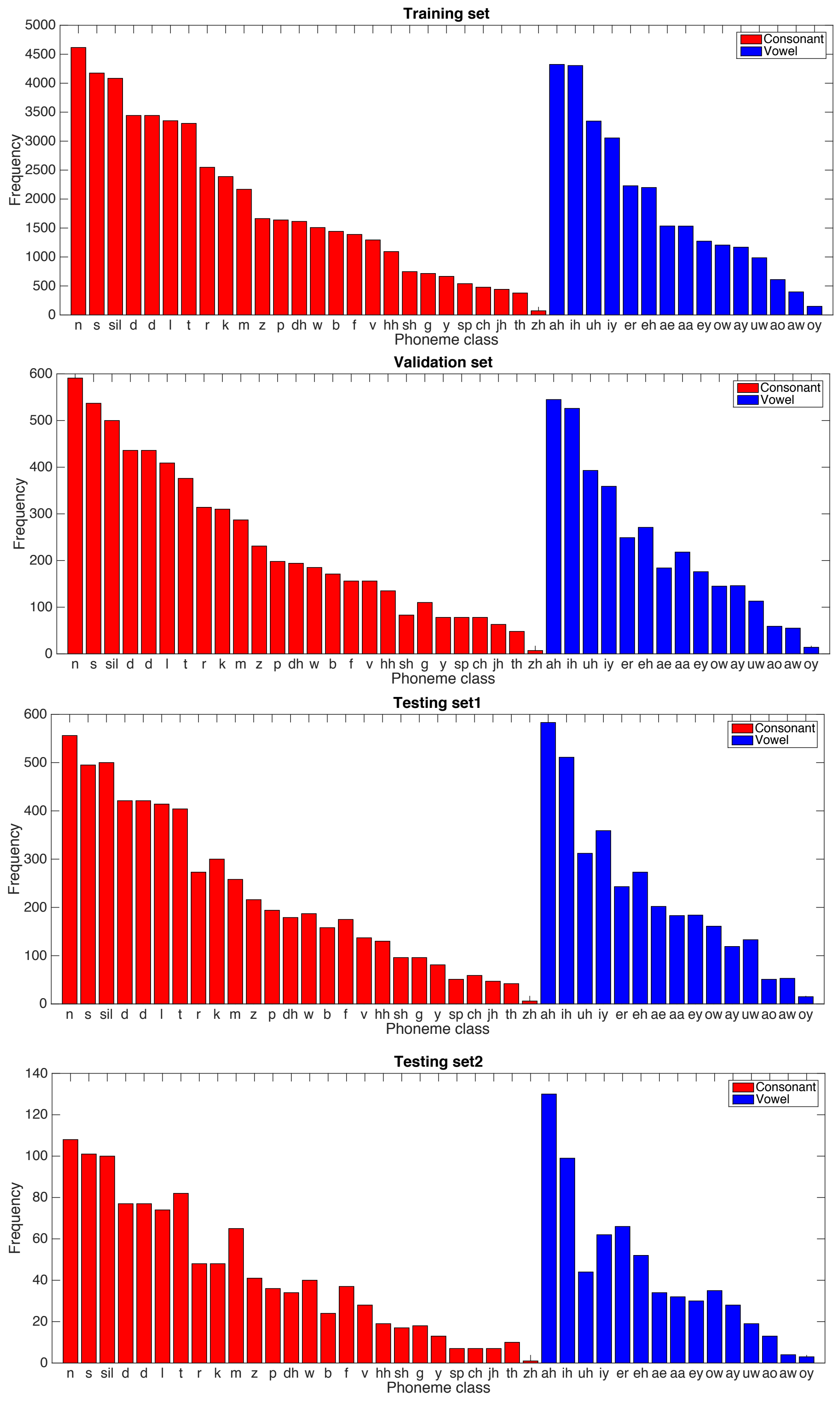


Figure 4.14: Phoneme distributions in training, development, testing and special testing sets.

Chapter 5

Visual speech synthesis based on hidden Markov models

5.1 Introduction

Until recently, in the last 20 or so years, most automatic speech recognition systems that have been developed are based on hidden Markov models (HMMs) [41, 51]. At the start of that period, text to speech systems (TTS) tended to use unit selection techniques. At this time, research into using HMMs for synthesis led to a step-wise output of parameters which produced low quality speech. This problem is originated from a maximum likelihood solution which always outputs the same mean values in each state, and a step change output at state boundaries. A major improvement was the introduction of using temporal derivative as constraints with the maximum likelihood solution, which produces a more smooth and realistic output [110, 111]. Subsequently HMMs gained popularity for synthesis and a series of publications and toolkits (e.g., an open source system HTS [126]) for developing speech synthesis system have been produced.

This chapter starts with the basic structure of HMM synthesis in the visual domain is described, for both the training and synthesis parts. Finally, an experimental section focuses on the effect of video data rate, dynamic features, number of HMM states and visual feature normalisation in terms of the synthesised visual output. We also examine three types of basic units: phonemes, traditional (static) visemes, and dynamic visemes, and aim to find the most suitable unit for visual speech synthesis using HMMs.

5.2 HMM-based visual speech synthesis

This section describes how context-dependent labels are used to train context-dependent HMMs and generate visual speech parameters using phoneme, static viseme, or dynamic viseme units. Figure 5.1 shows a framework of the proposed HMM-based visual speech synthesis system, that aims to generate a sequence of lip shapes from a given text. Our framework can be divided into two parts: training and testing. In the training part, context dependent HMMs are trained based on an audiovisual speech database. In the synthesis part, the visual parameters are generated from input text using the trained HMMs. Note that, this work was implemented by HTS-toolkit [126] and it takes about three days to train on UEA's High Performance Compute Cluster (GRACE) [1].

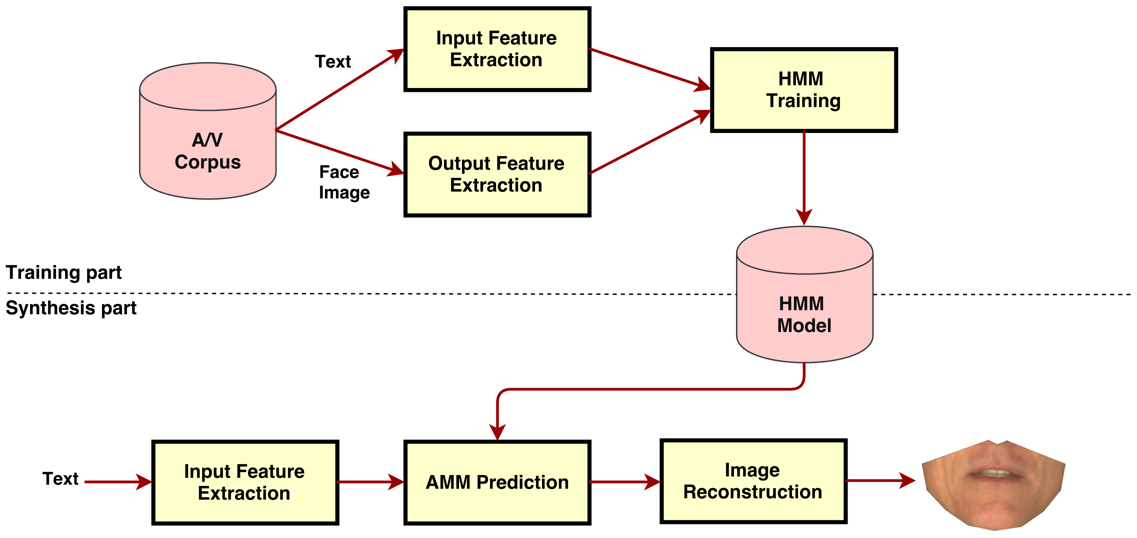


Figure 5.1: An overview of the HMM-based visual speech synthesis system.

[1]https://rscs.uea.ac.uk/high-performance-computing

5.2.1 Training part

The training process of a HMM-based visual speech synthesiser is similar to the training of a HMM-based audio speech synthesiser. The main difference is the observation feature vectors for visual speech synthesisers are typically based on visual features, for example AAM parameters, 3D facial points, or lip-shape points from face images. However, audio TTS systems are typically based on audio features, for example mel-generalised cepstrum or filter-bank features.

For training, the given text and face image from an audiovisual speech database are converted to the input features, **X**, and output features, **O**, using input feature extraction and output feature extraction. Then, both contextual input and contextual output features are used to train the context-dependent HMMs. More specific details of each module are described as follows.

5.2.1.1 Input feature extraction: contextual input

The input of this module is a sequence of words that will subsequently generating the lip movement animation. Generally, most systems, except the latest end-to-end speech synthesis systems convert each word (written form) to a sequence of phonemes (sound form). The main reason for the use of written form to sound form conversion is that the written form does not indicate how to pronounce each word but we are able to know how to make up its sound from the sound form. Moreover, this work will incorporate not only the sound form but also the visual form. We then consider two additional types of the visual form: static viseme units and dynamic viseme units.

For simplicity, we begin with the conversion of a sequence of word, w, to a sequence of phoneme with linguistic features, l, also known as contextual labels. This process can define as follow

$$\hat{l} = argmax_{\ell} P(l|w) \tag{5.1}$$

where w is the given text input and l is the contextual labeling. Ideally, this process aims to get useful information that affects the way people speak from the text input. In speech recognition, triphones are the standard contextual unit. Conversely, quinphones with prosodic and linuguistic information are the standard contextual information for audio and visual speech synthesis systems and are known as full contextual features [25]. These contextual factors can be divided into 5 levels of (visual) speech unit including segment (e.g. phoneme, static viseme, dynamic viseme), syllable, word, phrase, and utterance levels. The relationship between each level can be represented in a tree as shown in Figure 5.2. Bigraphs are used to link items in each level together.

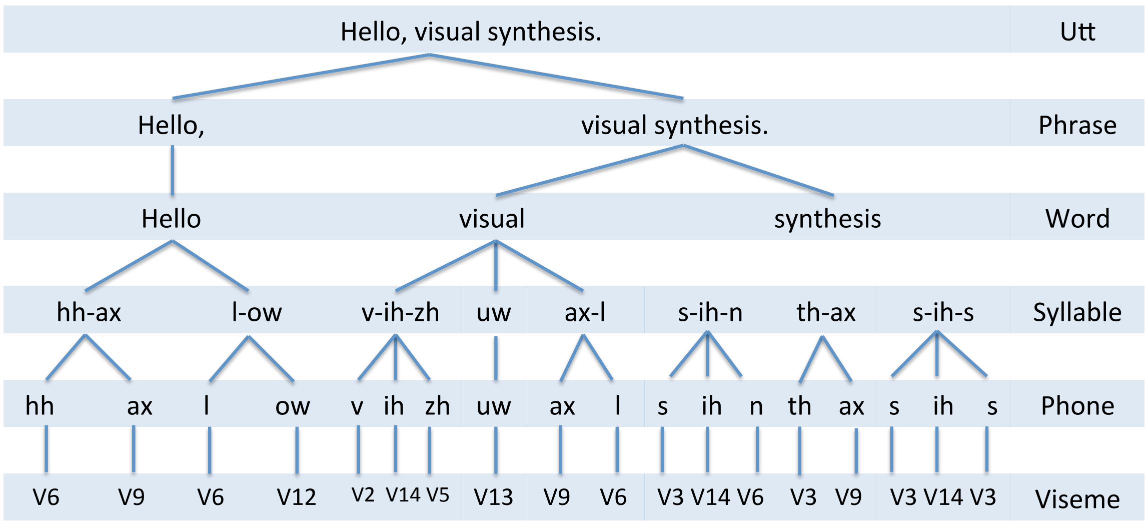


Figure 5.2: An example of feature representation structure in utterance, phrase, word, syllable, phone, and static viseme levels.

From this Figure, it shows that this utterance, “Hello, visual synthesis”, has a breath break after “Hello”. This example comprises three words: “hello”, “visual”, and “synthesis”. The total number of syllables is 8 and we are also able to find out the number of syllables in each word, such as the word “synthesis” has 3 syllables (described in Section 4.5.2). This structure also shows the one-to-one mapping of two types of segment units: phoneme and static viseme units. As mentioned, we aim to extract the useful information from raw text input, so the full set of features considered is summarised in Table 5.1 which shows those for phonetic units (PH), static viseme units (SV), and for dynamic viseme units (DV).

With segment level features, **Quin-phone context** considers the preceding two and succeeding two phonemes, **Quin-static viseme context** considers the preced-

Table 5.1: *Contextual features for phonemes (PH), static visemes (SV) and dynamic visemes (DV) units at varying levels.*

Level	Symbol	Feature	PH	SV	DV
Segment	(U1)	Quin-phone context	x		
	(U2)	Quin-static viseme context		x	
	(U3)	Quin-dynamic viseme context			x
	(U4)	Number of phonemes in dynamic viseme			x
	(U5)	Phonemes in dynamic viseme			x
Syllable	(S1)	Position and number of phonemes in syllable	x		
	(S2)	Position and number of static visemes in syllable		x	
	(S3)	Position and number of dynamic visemes in syllable			x
Word	(W1)	Position and number of syllables in word	x	x	x
Phrase	(P1)	Position and number of syllables in phrase	x	x	x
	(P2)	Position and number of words in phrase	x	x	x
Utterance	(U1)	Position of syllable, word and phrase in utterance	x	x	x

ing two and succeeding two static visemes, and **Quin-dynamic viseme context** considers the preceding two and succeeding two dynamic visemes. Figure 5.3 illustrates examples of these three type of units regarding the context features.

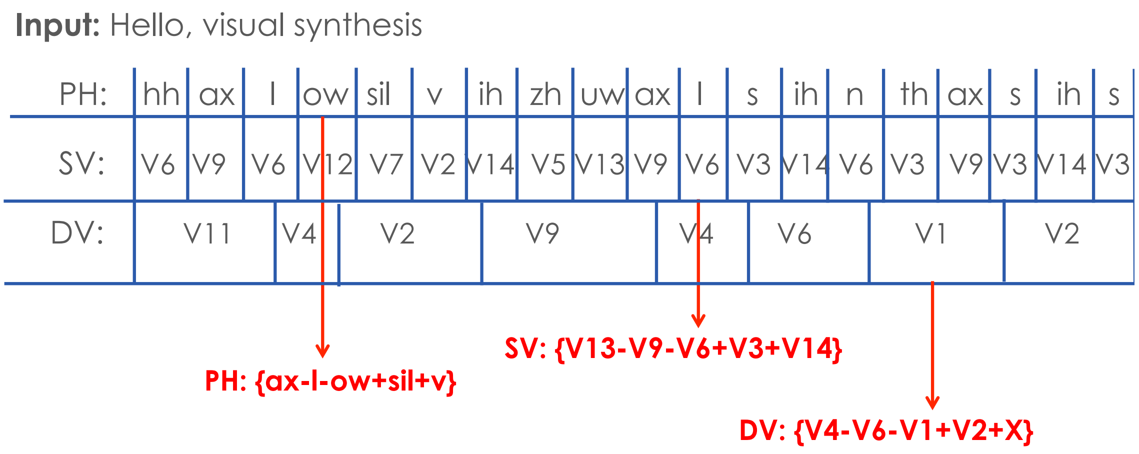

Input: Hello, visual synthesis
PH: hh ax l ow sil v ih zh uw ax l s ih n th ax s ih s
SV: V6 V9 V6 V12 V7 V2 V14 V5 V13 V9 V6 V3 V14 V6 V3 V9 V3 V14 V3
DV: V11 V4 V2 V9 V4 V6 V1 V2
PH: {ax-l-ow+sil+v}
SV: {V13-V9-V6+V3+V14}
DV: {V4-V6-V1+V2+X}

Figure 5.3: Examples of Quin-phone, Quin-static viseme, and Quin-dynamic viseme context features are shown in red.

Note that a dynamic viseme relates to a cluster of visual gestures, and each visual gesture is a specific instance of a dynamic viseme (c.f. phonemes and phones). With However, there is an issue when using dynamic viseme units in terms of how the number of phones associated with a visual unit is defined. This issue arises because the boundaries of the visual gestures generally do not align with the boundaries of phonemes and each visual gesture can span multiple phones. In the KB-2k corpus we find that approximately 80% of the gestures span two to four phones, as can be seen in Figure 5.4.

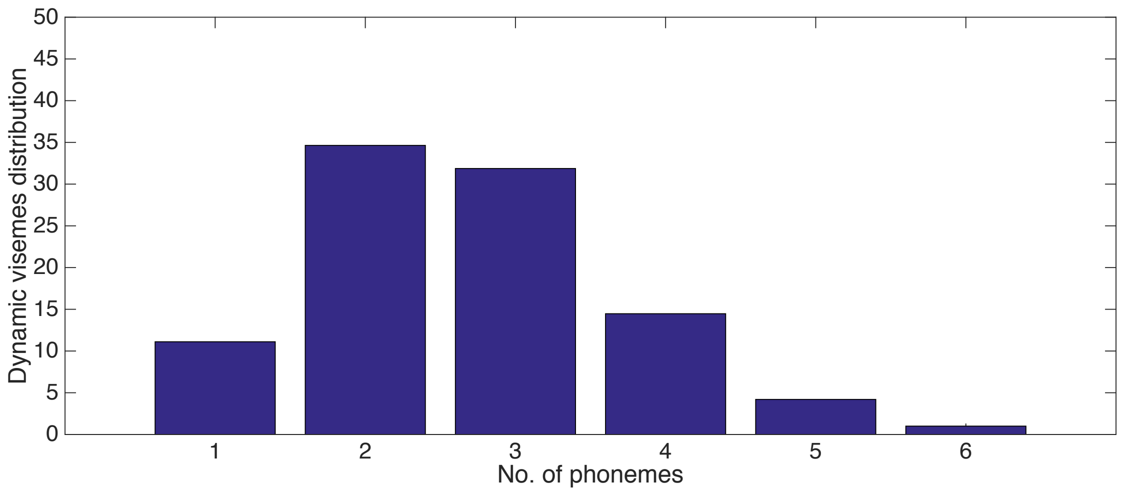


Figure 5.4: Distribution of the number of phones spanning over in each dynamic viseme.

To allow for partial coverage of a phone by a dynamic viseme, we introduce contextual labels that mark the presence of a dynamic viseme within the phone. Specifically, if a phone is completely spanned by a dynamic viseme the customary label is used (e.g. /p/). The following contextual labels are then added if a dynamic viseme is present only at the beginning (/@p/) or end (/p@/) of the phone. Occasionally there are very short gestures identified completely within a phone, and so the gesture covers only the mid-portion of the phone. In this instance /@p@/ is used. Figure 5.5 shows examples of the phonemes in dynamic viseme "V4" and "V9".

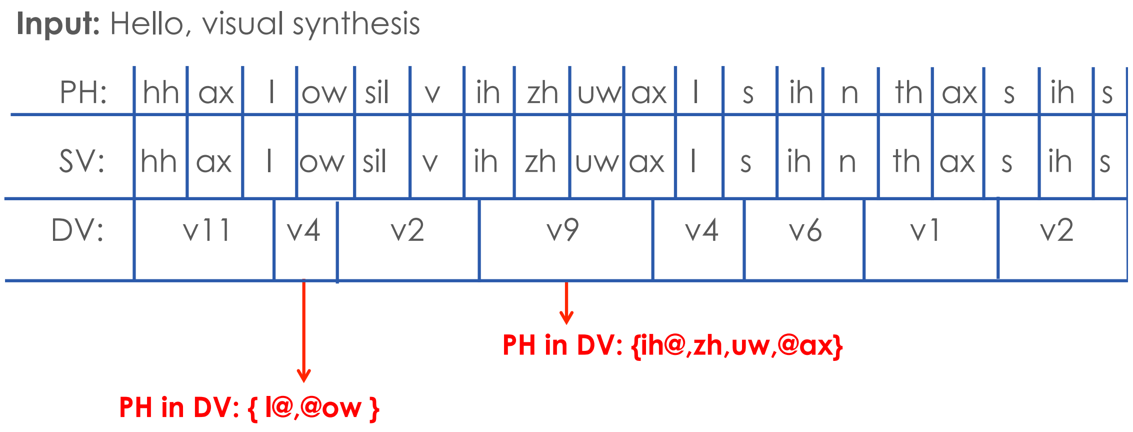

Input: Hello, visual synthesis
PH: hh ax l ow sil v ih zh uw ax l s ih n th ax s ih s
SV: hh ax l ow sil v ih zh uw ax l s ih n th ax s ih s
DV: v11 v4 v2 v9 v4 v6 v1 v2
PH in DV: {ih@,zh,uw,@ax}
PH in DV: { l@,@ow }

Figure 5.5: Examples of the extraction of phoneme in dynamic visemes are shown in red.

With the other levels of syllable, word, phrase and utterance, all features describe the number and position (*start*, *middle*, *end*, and *single*) of one level in another level, such as number of phonemes in syllables. According to the example in Figures 5.2 and 5.5, the linguistic features of phoneme "ih" for the word "synthesis" are illustrated in Table 5.2.

Table 5.2: *Contextual linguistic features of phoneme /ih/ in Figure 5.2 for phonetic (PH), static viseme (SV), and dynamic visemes (DV) units at syllable, word, phrase, and utterance levels.*

Level	Feature	Representation
Syllable	Position of phonemes in syllable	start
	Number of phonemes in syllable	3
	Position of dynamic visemes in syllable	start
	Number of dynamic visemes in syllable	2
Word	Position of syllables in word	start
	Number of syllables in word	3
Phrase	Position of syllables in phrase	middle
	Number of syllables in phrase	8
	Position of words in phrase	end
	Number of words in phrase	2
Utterance	Position of syllables in utterance	end
	Position of words in utterance	end
	Position of phrase in utterance	start

5.2.1.2 Output feature extraction: contextual output

This module aims to extract a sequence AAMs of face images into output features. Regarding the face image, this work focuses on only the lower face movements because these part are directly effect with speech movements and other parts, such as eyes, eyebrows, can sometimes introduce noise to the models. Subsequently, active appearance models (AAMs) is used to describe the shape and appearance of its lower face (see in Section 4.3). From this algorithm, it is able to reduce a high dimensional,

1920×1080 block of pixels, into a low M-dimensional AAM parameter vector where, for example $M = 30$. These AAM parameters are also used to reconstruct a near photo-realistic animation of the face. Eventually, we use not only the static features, but also include the dynamic features forming the contextual output before training the HMMs synthesis system. More specific details can be found in Section 3.2.4.

5.2.1.3 HMM training: context dependent HMMs

After extracting full contextual input and contextual output features, training uses both sets of information to train the decision tree-based context dependent HMMs. In the case of using phoneme as a basic speech unit, firstly, a context independent phoneme-HMM (or monophone HMM) is used to model each phoneme. Dines et al. [25] suggests that the standard number of HMMs usually consists of 5 emitting states with single Gaussian probability distribution function (PDF) per emitting state. There are seven states in total because of an additional non-emitting state at the beginning and end of each model.

There have been several studies that suggested that using monophone HMMs is not effective. Instead, context independent HMMs are replaced by context dependent HMMs. For example [92] were able to improve recognition accuracy by using triphone models compared with monophone models. However, one of the major problems of triphone models is sparsity of data, in that it is not possible to prepare training data to include examples of every possible triphone context. For the KB-2K corpus, this would require 68,921 (e.g., 41^3 phonemes) triphone HMMs. To deal with this problem, many tied-parameter techniques have been proposed to cluster HMM states that have a similar statistical distribution.

In (visual) speech synthesis, it is not only triphone models but also many contextual features in cluding quinphone models with linguistic information, that are used. Based on the increasing of these contextual features combinations, binary decision tree-based context clustering is used to control the model complexity by clustering HMM states and sharing the distributions (pdfs) of the visual speech features among states in each cluster. Each node, except the leaf nodes, has a context related label sequence, called a question. This question sets are prepared and related to input contextual information. For example, *C-Silence* represents "Is the current phone silence or not" and relates to "Quin-phone context", *Num_syl-in-word==5* represents "Does the current word consist of five syllables" and relates to "W1" in Table 5.1. As a clustering scheme, this ensures that all contexts, including unseen contexts, can be found by traversing the decision tree. An example part of binary decision trees of dynamic viseme HMM models is shown in Figure 5.6.

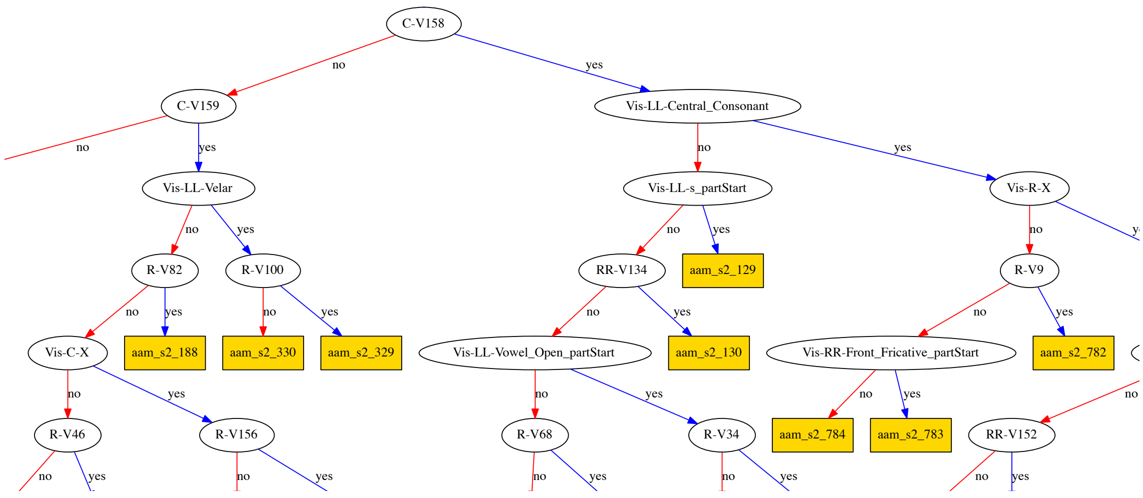


Figure 5.6: An example part of context-dependent binary decision tree that generated from HMM-based using dynamic viseme units.

Speech recognition uses Minimum Description length (MDL) criterion to control the tree growth [93]. Note that, the MDL was applied to audio TTS and incorporated into HTS toolkit by [119]. In the process of decision tree construction, the MDL is used to make a decision whether splitting or stopping the current node. As mentioned, the context-dependent decision trees are constructed by each individual states. The statistical of the generated models is shown in Table 5.3. Overall, it can be seen that the depths of the decision tree from dynamic viseme units are deeper that that from phoneme and static viseme units. This is not surprisingly result because number of dynamic visemes is greater than that of other two units. Moreover, the most distribution of dynamic viseme units was originated from state 5. This means more than 95 percent of the total number of 39,534 full-context models can be grouped into 1839 clusters.

Table 5.3: *Statistics of the models generated after training HMMs using MDL.*

Features		PH	SV	DV
Number of contexts in training	Including silence	69,391	68,638	39,534
	Excluding silence	68,184	67,767	37,444
Number of leaf nodes	State 1	543	462	1,129
	State 2	723	542	831
	State 3	984	944	926
	State 4	533	487	961
	State 5	438	410	1,839
	Total	3,217	2,895	5,686

5.2.2 Synthesis part

This section describes how to generate the lip movement animation from the input text. In a real-world application, this part comprises two main systems including the text-to-speech (TTS) system and a text-to-visual speech system. Firstly, the TTS system is used to convert the written form to the acoustic form and then predict the phone durations forming the mono phone labels. After that, the synthesised speech is generated based on these labels. Secondly, the visual text to speech system also uses the same labels to predict the visual features and then reconstructs a sequence of lip images. Finally, the lip movement animation is created by combining lip image synchrony sequences and synthesised speech from the TTS system.

However, this work focuses on visual-only synthesis. It is assumed that mono phone labels are available and natural speech is used instead of synthesised speech in the process of generating the lip movement animation. From the synthesis parts in Figure 5.1, the input text and mono phone labels are converted into full contextual input features (as described in Section 5.2.1.1). Then, the stepwise AAM parameters of static and delta and delta-delta derivative are predicted from the trained decision tree based context HMMs. Next, the parameter generation algorithm, as described in Section 3.2.4, is used to generate a sequence of smoothed AAM outputs. Finally, these outputs are reconstructed into the lip images and combined with the natural speech to form the lip movement animation.

5.3 Experiment results

This section aims to perform an evaluation of the synthesised visual speech considering three different kinds of objective scores including correlation, normalised root means square error, and global variance. Experiments are performed on the KB-2k audiovisual speech dataset, as described in Chapter 4. To make a fair comparison, then, we adopt the same 2,042 training sentences and synthesise lip animation from the 50 held-out sentences (Training and SpecialTesting data set as described in Section 4.5.3), which were used in the previous work such as the unit-selection system in [99].

5.3.1 Objective tests

In objective evaluation, the error between the predicted visual AAM features and the groundtruth AAM parameters is measured using three objective metrics: (i) correlation, (ii) normalised root mean square error (NRMSE), and (iii) global variance (GV). Note that, most visual speech synthesisers commonly measure error objectively by correlation and RMSE. A positive or negative correlation in terms of visual speech synthesis relates to audio and visual synchronisation. For example, the same AAM visual trajectory with shift in time (small correlation) leads to the lack of audio and visual synchrony. Most of the case RMSE requires to consider with correlation because it does not guarantee that the low RMSE is better than the high RMSE. For example, the work from Theobald [103] found that the high RMSE with positive correlation from smoothed data showed better animation compared with the small RMSE with negative correlation. Additionally, we consider using normalised-RMSE instead of RMSE to confirm that the entire vector has the same unit and scale of the features [104].

We also propose the global variance (GV) of the visual features as one of our objective tests. GV could be used to guide and select the best model in the case of correlation and normalised-RMSE methods report the similar scores. This metric is inspired by work in the speech synthesis community that found the dynamic range of the acoustic features degraded the sound quality. From this concept, we assume that the range is possibly linked to the under and over articulation of lip movement animations and this is the main reason to use raw data instead of normalised data as used in RMSE metric. Table 5.4 shows the GV of the training, validation and testing set of the original data.

Table 5.4: *The global variance of the training, validation and testing set of the original KB-2k data.*

	Training set	Validation set	Testing set
GV	1461.03(±384.35)	1477.94(±457.65)	1477.82(±378.94)

Figure 5.7 illustrates an example of the first AAM parameter between ground-truth (normal dynamic range in black) and generated (small dynamic range in red) trajectories for the utterance: "students choices of ideal educational goals are not arbitrary or whimsical". After that, we also investigate the effect of normal and low dynamic range into the lip movement animation for the word "chooses", as depicted in Figure 5.8. It can be seen that the image quality from the small dynamic range trajectory (bottom) is getting more blured and less flexible in terms of open and closed mouth shaped compared with the normal dynamic range (top). That means the global variance of the AAM features is related to the naturalness of the synthesised lip animation in terms of over- and/or under-articulation. The animation results of normal and small dynamic range can be seen in supplementary video.

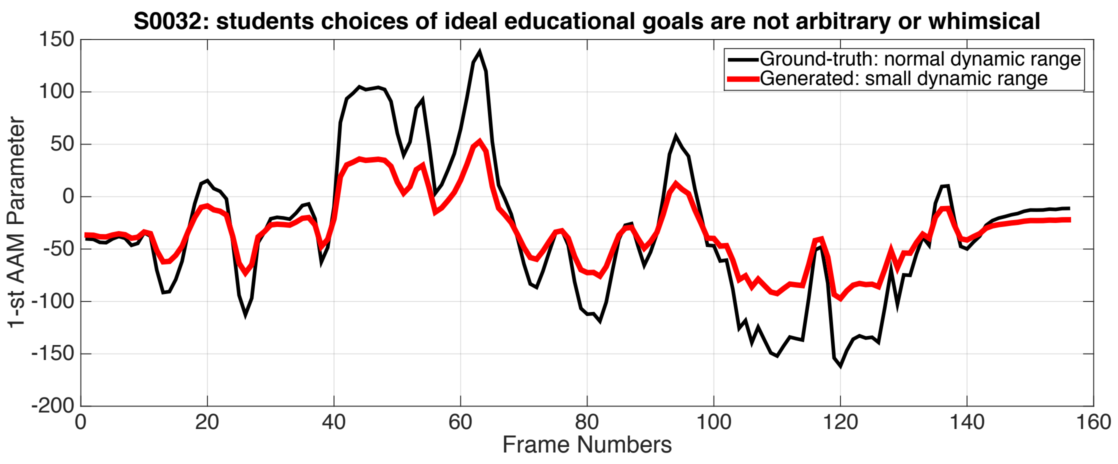


Figure 5.7: Two different dynamic ranges that originated from the ground-truth and generated AAM trajectories.

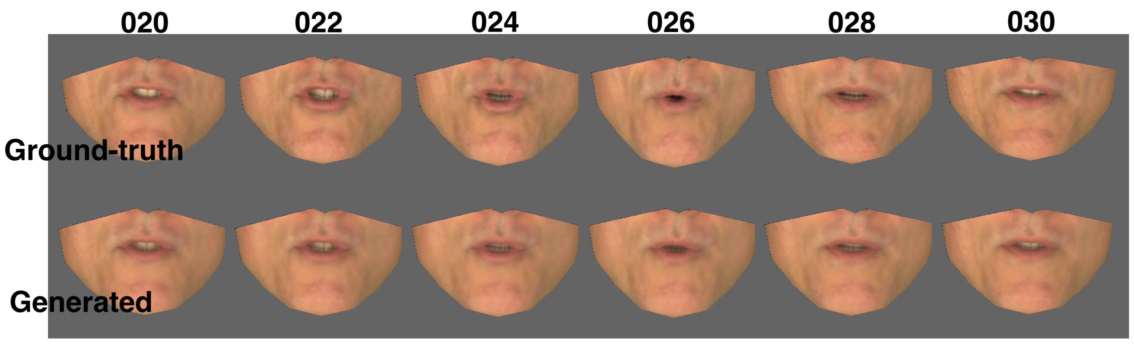


Figure 5.8: Frames 20-30 from the ground-truth trajectories (top) and generated trajectories (bottom), corresponding to the the first syllable of the word choices. Both examples tend to present similar lip shapes but the image quality from the small dynamic range trajectory (bottom) is getting more blur and less flexible in terms of open and closed mouth shapes than the normal dynamic range (top).

The correlation, NRMSE, and GV measures are calculated as follows:

$$Correlation = \frac{1}{D \times N} \sum_{d=1}^{D} \sum_{j=1}^{N} \frac{(y_d^j - \mu_d)(o_d^j - \hat{\mu}_d)}{\sigma_d \hat{\sigma}_d}, \tag{5.2}$$

$$NRMSE = \frac{1}{D} \sum_{d=1}^{D} \frac{\sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_d^j - o_d^j)^2}}{y_{max,d} - y_{min,d}}, \tag{5.3}$$

$$GV = \frac{1}{D \times N} \sum_{d=1}^{D} \sum_{j=1}^{N} (o_d^j - \hat{\mu}_d)^2, \tag{5.4}$$

where $\mathbf{y}_j$ is the D-dimensional ground-truth AAM vector, $\mathbf{o}_j$ is the D-dimensional synthesised AAM vector, N is the number of vectors in each sentence. μ_d and $\hat{\mu}_d$ are the respective mean of the d^{th} dimension of y and o across the frames from 1 to N, and μ_d and $\hat{\mu}_d$ are the respective variance of the d^{th} dimension of y and o across the frames form 1 to N. $y_{max,d}$ and $y_{min,d}$ are the maximum and minimum value from the D-dimensional ground-truth AAM vectors, respectively. Cootes et al. [19] and Theobald [103] reported that it is not necessary to capture 95% of the total variation when building speech animation. Theobald found that the first few parameters at 70-85% of the total variation are able to retain the most important variation in terms of synthesis. Note that, for evaluation proposed this work sets the number of D to 5 which retains approximately 80% of the variation that they explain, as shown in Figure 5.9.

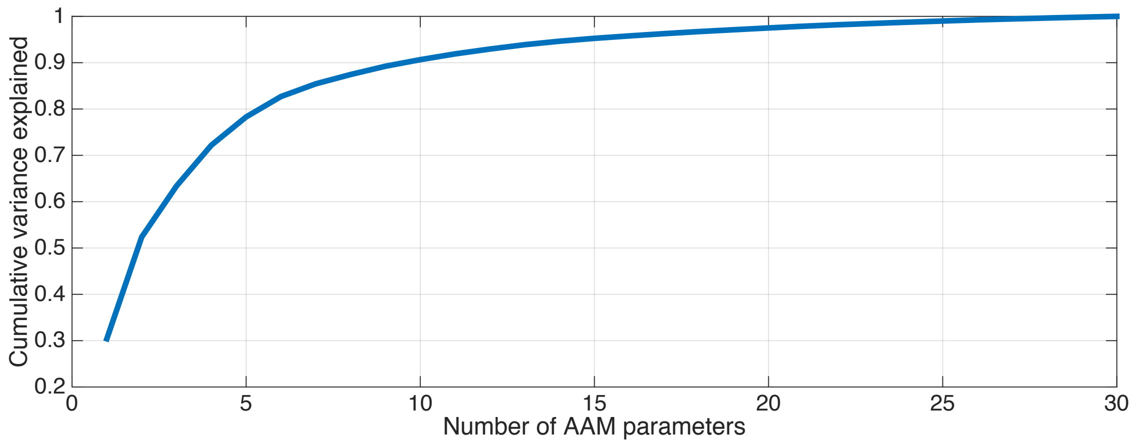


Figure 5.9: Cumulative variance explained by AAMs.

5.3.1.1 Effect of frame rate

This experiment aims to investigate the effect of the AAM's frame rate to find out a suitable value for HMM-based visual speech synthesis. Most audiovisual speech applications (e.g., audiovisual speech recognition) upsample the visual data to match the audio data frame rate. For example, the video rate at 50fps was upsampled to 200fps in audiovisual speech synthesis [23]. It is, however, interesting to know which data rate is the best number for visual speech synthesis using HMMs. The training conditions in this experiment follow the standard HMM-based speech synthesis from the HTS toolkit [126] with five emitting states, no skips and single component Gaussians with a diagonal covariance matrix used to model context dependent phoneme HMMs.

Our preliminary work found that the original video rate of 29.97fps is insufficient for training the HMMs because it does not have enough vectors for some short-duration phonemes. We found that the duration in each phoneme must be longer than approximately 150ms for the five emitting state HMMs. Unfortunately, the average of phoneme duration of KB-2k is 95 ms and nearly 85% of phonemes have a duration less than 150ms. To overcome this problem, the AAM parameters are first upsampled from 30fps (original video rate) to 100fps or 200fps using cubic spline interpolation. Note that, linear interpolation function is not recommended because this may cause zero 2nd-order derivative which we do not learn anything about the slope of the features.

Table 5.5 presents the averaged objective scores of data rate at 100fps and 200fps. It shows that the data rate at 100fps outperforms the data rate at 200fps in two-third of the objective scores including correlation and normalised RMSE. Moreover, with the inspecting the prediction videos, we found that the lip animation from the frame rate of 100fps appear natural and better than that of 200fps. Hence, it can be conclude that 100fps is a suitable frame rate for HMM visual speech synthesis and this rate will be used for the remaining experiments in this chapter.

Table 5.5: *The averaged scores of phoneme HMM-based visual speech synthesis in different data rate.*

Data rate	Correlation	NRMSE	GV
30fps	-	-	-
100fps	0.77(±0.09)	9.93(±2.24)	794.53(±168.01)
200fps	0.75(±0.08)	10.31(±2.07)	889.96(±177.95)

5.3.1.2 Effect of dynamic feature

This experiment aims to investigate the effect of the width of the window used while computing the dynamic features in each frame (described in Section 5.2.1.2).

The hypothesis is that the longer window might be beneficial in a different way for visual speech. Based on the standard HMM-based speech synthesis from HTS [126], most systems use three-frames window to calculate delta and delta-delta (e.g., $K^L = K^R = 1$ in Equation (3.13)).

To explore the benefits of different window widths for dynamic coefficients for HMM-based visual speech synthesis, we varied the window width from 1 to 9 via K^L and K^R settings. HMMs with 5-states, left-to-right with no-skip topology with a single diagonal Gaussian output distribution were used to build all phone context dependent models. The averaged results (mean standard deviation) of correlation, normalised-RMSE, and GV in terms of AAM parameters are shown in Table 5.6.

Table 5.6: *The averaged objective scores of phoneme HMM-based visual speech synthesis in different number of window length for computing dynamic feature.*

K^L	K^R	Total Width	Correlation	NRMSE	GV
0	0	1-frame	0.70(±0.08)	11.12(±2.06)	987.97(±158.97)
1	1	3-frame	**0.77(±0.09)**	**9.93(±2.24)**	794.53(±168.01)
2	2	5-frame	0.76(±0.09)	10.20(±2.22)	866.30(±151.39)
3	3	7-frame	0.74(±0.09)	10.35(±2.16)	867.97(±153.94)
4	4	9-frame	0.74(±0.08)	10.47(±2.20)	866.87(±147.78)

From this table, the first configuration "(1-frame)" sets K^L and K^R to 0 that yields to the use only static features. It shows that this setting ($K^L = K^R = 0$) reports the worst scores compared with the others settings in terms of correlation and NRMSE scores. Note that, a large GV does not use to indicate the best setting in this case because this score corresponds to random signals. Additionally, it can be confirmed by the inspection of the time-varying trajectory of the predicted AAM (blue) compared with the ground-truth AAM (black) in Figure 5.10. Note that, the five piecewise trajectories in each phone (blue) are not appropriate for visual animations because these will lead to get the same lip image in each HMM state. However, this is not surprisingly results because this appears to have suffered from one of the HMM synthesis drawbacks where the same HMM state is produced the same outputs. It clearly shows that the use of dynamic features ($K^L = K^R = 1$) is able to reduce the discontinuities problem (red).

Figure 5.11 illustrates the inspection of the time-varying trajectory of the dynamic features with 5 window length ($K^L = K^R = 2$) and 7 window length ($K^L = K^R = 3$). It shows that both results are jerky and not able to generate continuous trajectories. It clearly shows that the use of dynamic features with the window length greater or equal five is unable to produce the smooth trajectory and propose a jerky problem.

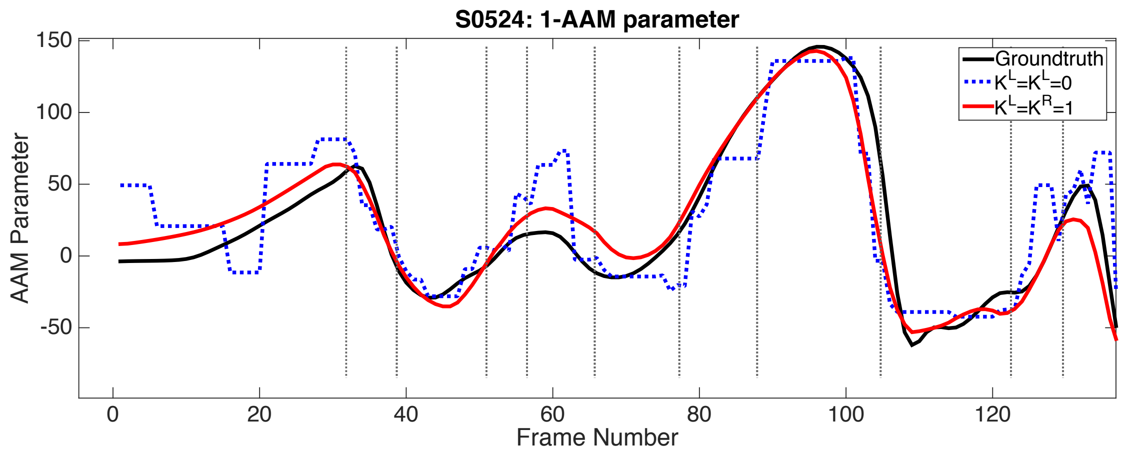


Figure 5.10: A ground-truth (black) AAM parameter trajectory compared with a synthetic AAM parameter trajectory from static features only (blue) and dynamic features with 3 window length ($K^L = K^R = 1$). Vertical lines show phoneme boundaries. This result shows static features, ($K^L = K^R = 0$), are not able to generate continuous trajectories. This can only produce around five different lip images from five emitting states in each phoneme. It clearly shows that the use of dynamic features is able to reduce the discontinuities problem (red).

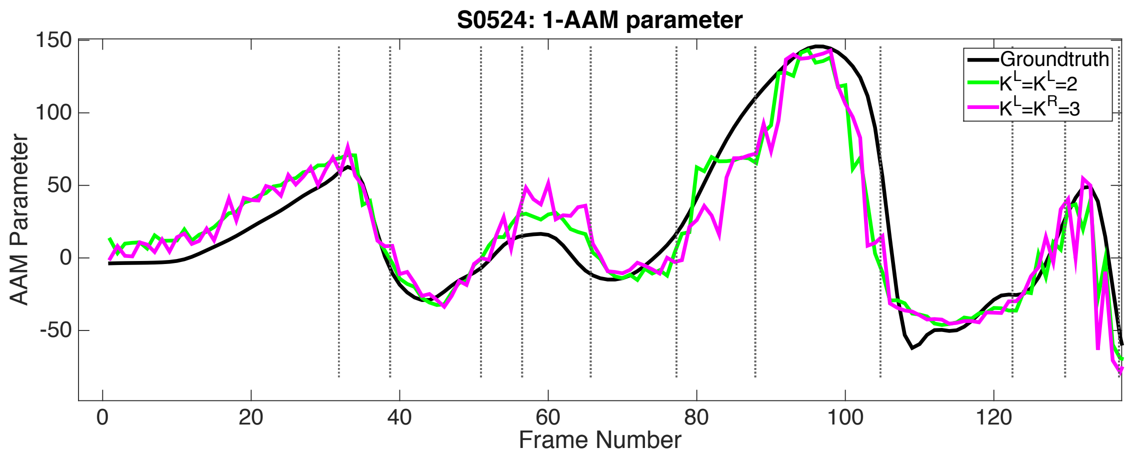


Figure 5.11: A ground-truth (black) AAM parameter trajectory compared with a synthetic AAM parameter trajectory from dynamic features dynamic features with 5 window length ($K^L = K^R = 2$) and 7 window length ($K^L = K^R = 3$). Vertical lines show phoneme boundaries. This result shows these results are jerky and not able to generate continuous trajectories. It clearly shows that the use of dynamic features with the window length greater or equal five is unable to produce the smooth trajectory and propose a jerky problem.

5.3.1.3 Effect of static viseme classes

Many existing visual synthesisers use (static) visemes as the visual speech unit. Visemes or visual phonemes are the basic unit of mouth movement [37]. From a visual point of view, some phonemes cannot be visually distinguished because they have the same place of articulation. For example, the phonemes /p, b, m/ are visually similar and represent the same viseme class.

There are many viseme classifications in the literature, because there is still no standard set of visemes and that visemes are less well defined than phonemes. Traditionally, most researchers define visemes as groups of phonemes and they are mapped using either a many-to-one phoneme-to-viseme mapping or a many-to-many phoneme-to-viseme mapping. The differences in each classification are the choice of the different groups of phonemes in each viseme. For example, the most common viseme classification are: MPEG-4 [79] mapped 43 phonemes to 14 visemes groups, or 5 consonant visemes groups from Fisher [37], .

In this experiment, we wish to determine if a HMM-based synthesiser based on visual phonemes (called static visemes) is able to synthesise visual speech better than equivalent synthesisers trained using phoneme units. Hence, our experiment uses the most common viseme groups from Fisher [37]; specifically, (i) consonant phoneme-to-viseme mapping and vowel phoneme-to-viseme mapping. In particular, They mapped 26 consonant phonemes to 7 visemes groups and mapped 17 vowel phonemes to 7 visemes groups using a decision tree, as shown in Table 5.7.

Table 5.7: A many-to-one phonemes to visemes mapping [37].

Visemes	Consonant phones	Visemes	Vowel phones
V1	/p,b,m/	V8	/eh,ey,ae,aw,er/
V2	/f,v/	V9	/ah,ax,ay/
V3	/t,d,s,z,th,dh/	V10	/aa/
V4	/w,r/	V11	/er/
V5	/ch,sh,zh,jh/	V12	/ao,ow,oy/
V6	/k,g,n,l,hh,ng,y/	V13	/uh,uw/
V7	/sil,sp/	V14	/iy,ih/

Three components including input features, output features, and HMM configurations are prepared to train the static viseme HMM-based system for visual speech synthesis. With input features, the full contextual input of static viseme units (SV) is summarised in Table 5.1 which includes features for static viseme (segment), syllable, word, phrase, and utterance. For output features, 30 AAM features and their delta and delta-delta at 29.97 fps (original frame rate) are upsampled to 100 fps, as shown from the promising results from the previous experiment. From the standard HTS training conditions [45], five emitting states, no skips with single component Gaussians with a diagonal covariance matrix are considered in this experiment.

The first three different HMM synthesis systems were used to determine the effect of static viseme classes: (System A) HMM-based using 7 groups of vowel viseme classes and 26 consonant phonemes, (System B) HMM-based using 7 group of consonant viseme classes and 17 vowel phonemes, and (System C) HMM-based using both 7 groups of consonant and 7 vowel viseme classes.

Table 5.8: *The averaged objective scores of static viseme HMM-based visual speech synthesis in different static viseme classes.*

System	Vowel units	Consonant units	Correlation	NRMSE	GV
A	7 visemes	26 phonemes	0.76(±0.08)	10.26(±2.23)	747.23(±134.95)
B	17 phonemes	7 visemes	0.76(±0.09)	10.09(±2.26)	803.51(±173.60)
C	7 visemes	7 visemes	0.74(±0.09)	10.49(±2.21)	764.99(±161.17)

From the results in Table 5.8, it can be seen that there are not significant difference among the three systems. Based on the average results, however, the better performance is indicated by high correlation and GV values and low NRMSE. Hence, the performance of the group of consonant visemes (System B) is slightly better than that of the group of vowel visemes (System A) because System B has the low NRMSE and high GV comparing with System A. This suggest that the individual vowel phonemes is very important for the synthesisers because System A grouped 17 vowel into 7 classes while System B used all of the vowel phonemes. This is also confirmed by the worst results in System C which used only the 14 viseme classees. These results suggest that the phonetic information of both consonant and vowel from phonemes is important.

Table 5.9: *The averaged objective scores of the different combination of static viseme and phoneme units for HMM-based visual speech synthesis system.*

System	Correlation	NRMSE	GV
PH HMM	0.77(±0.09)	9.93(±2.24)	794.53(±168.01)
System D	0.77(±0.09)	10.01(±2.25)	812.38(±170.06)
System E	0.77(±0.09)	9.98(±2.21)	804.51(±170.46)
System F	0.77(±0.09)	10.04(±2.25)	812.10(±171.60)

Hence, we are interested in incorporating static viseme units (SV) into phoneme units (PH) in three different systems: (System D) uses phoneme units and vowel viseme units, (System E) uses phoneme units and consonant viseme units, and (System F) uses phoneme units and a combination of consonant and vowel viseme units. From these three systems we are able to use the phonetic information from phonemes and use viseme classes as additional visual information. As expected, the combined PH+SV units (System D-F) achieved better objective scores than SV units (System A-C). Unfortunately, the results of combined PH+SV systems did not bring in the improvement compared with baseline PH HMM system. This indicates that the group of consonant and vowel phonemes (static visemes) did not utilise the additional visual information. Thus, this is the main reason to introduce the use of dynamic viseme units in the next experiment.

5.3.1.4 Effect of dynamic viseme classes

The previous experiment showed that the use of static viseme units for HMM-based visual synthesis does not bring in the benefits of visual information from the groups of visually similar phonemes. This is not surprising as traditional static visemes are not visually similar. Hence, these are the main reason to use dynamic visemes that are learnt independently of acoustic speech, and provide a library of visible articulator movements [98]. A set of dynamic visemes is the first thing that we are concerned with when using dynamic viseme units.

This experiment aims to find a suitable number of dynamic viseme classes for HMM-based visual speech synthesis. As mentioned in Section 4.4.1.2, the cluster compactness is used to measure clustering quality and to determine the required number of dynamic visemes. However, this measurement does not directly tell us about how good the synthesised speech is. Hence, the number of dynamic visemes is varied between 10 and 250 to represent a basic unit for HMM-based visual speech synthesis. In a training step, the full contextual input of dynamic viseme units (DV) is extracted from each dynamic viseme (segment), syllable, word, phrase, and utterance features in Table 5.1. For output features, the AAM static and dynamic features, with an original frame rate of 29.97 fps, are upsampled to 100 fps. From the standard HTS training conditions [45], five emitting states, no skips with a single component Gaussian with a diagonal covariance matrix are considered in this experiment.

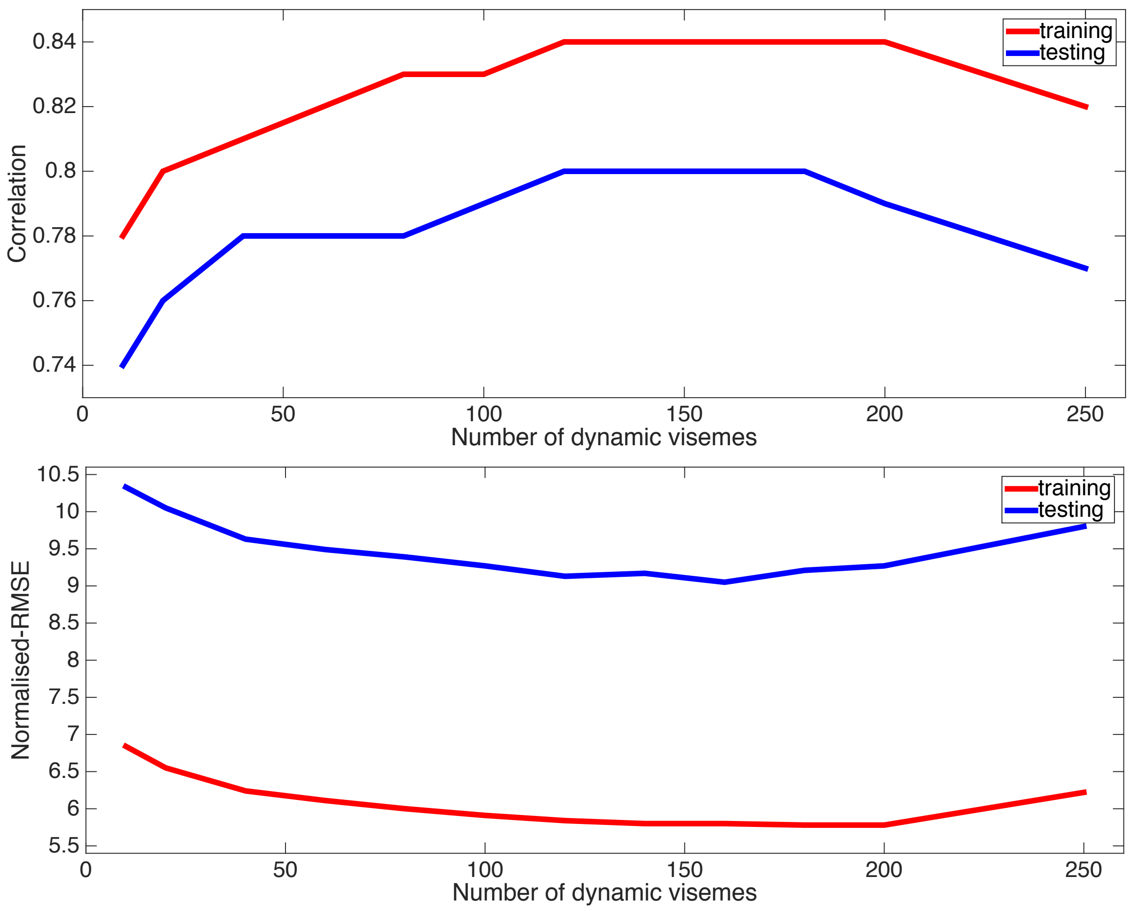


Figure 5.12: The averaged correlation (top) and normaliased-RMSE (bottom) using dynamic viseme classes from 10 to 250.

From this figure, both training and testing sets have the same trend of results which the predicted AAM features improve when increasing the number of dynamic visemes. However, after 180 classes the correlation and normalised RMSE begin to deteriorate. Eventually, the appropriate number of visemes is about 140-180, which is close to the number as determined by Taylor et al. [98] at 150 classes.

5.3.1.5 Effect of output normalisation

As we know, the process of normalising input and output features in each dimension into the same range is necessary for neural networks [63]. With the HMM-based visual speech synthesis framework, most systems have used the raw data with no normalisation. Hence, the purpose of this experiment is to examine a suitable data encoding. We are not interested to find out the effect of this transforming process to the input data because the input values for HMM-based framework do not affect to the performance. For instance, one system specifies the stress value with "1" and another system specifies a stressed value with "1.5". In the process of building full context model, it does not matter how they denote the stress value. Both systems are still able to refer to the same model and get the same mean and variance.

There are various options of transforming the dynamic range of the output data, such as scaling and standardisation (described in Section 4.5.1). This work applied the zero normalisation (z-score normalisation) method to the AAM output parameters. Then, the transformed data in each dimension could represent the data into a common scale with a mean of zero and unit standard deviation. After that, these data were used to train the phoneme HMM-based and dynamic viseme HMM-based for visual speech synthesis systems.

Table 5.10: *The averaged objective scores of the importance of standardisation method on AAM visual output for phoneme HMM system and dynamic viseme HMM system.*

		Correlation	NRMSE	GV
PH	norm	0.77(±0.09)	10.12(±2.33)	744.86(±159.13)
	unnorm	**0.77(±0.09)**	**9.93(±2.24)**	**794.53(±168.01)**
DV	norm	0.80(±0.08)	9.34(±1.94)	881.69(±171.07)
	unnorm	**0.80(±0.07)**	**9.05(±1.86)**	**942.62(±197.33)**

The averaged results in both phoneme (PH) and dynamic viseme (DV) system are shown in Table 5.10. Where "unnorm" denotes the raw AAM data and "norm" denotes for z-score normalised AAM data. Clearly, the unnorm data has slightly better results than the norm data in both systems. The difference in results might be originated in the process of re-transforming the normalised data back to the raw data space. Overall, it can be concluded that standardisation had no effect on performance for HMM-based visual speech synthesis framework.

5.3.1.6 Effect of number of HMM states

The aim of this experiment is to find out a suitable number of HMM states in both phoneme HMM-based and dynamic viseme HMM-based systems. The training conditions of this experiment followed the standard settings from HTS (as described in 5.2.1.3) except the number of HMM emitting states which was varied from 1 to 9.

Table 5.11: *The averaged objective scores of phoneme HMM-based visual speech synthesis in different number HMM states.*

Unit	# of HMM states	Correlation	NRMSE	GV
PH	1	0.72(±0.10)	10.79(±2.33)	672.78(±160.96)
	3	**0.78(±0.08)**	**9.92(±2.24)**	**819.18(±179.62)**
	5	0.77(±0.09)	9.93(±2.24)	794.53(±168.01)
	7	0.68(±0.13)	11.28(±2.27)	645.01(±148.72)
	9	0.34(±0.19)	14.21(±2.11)	191.78(±58.51)

Table 5.12: *The averaged objective scores of dynamic viseme HMM-based visual speech synthesis in different number HMM states.*

Unit	# of HMM states	Correlation	NRMSE	GV
DV	1	0.65(±0.09)	11.32(±1.83)	628.22(±177.94)
	3	0.78(±0.07)	9.45(±1.85)	897.98(±192.06)
	5	**0.80(±0.07)**	**9.05(±1.86)**	**942.62(±197.33)**
	7	0.80(±0.07)	9.19(±1.80)	936.48(±204.13)
	9	0.79(±0.08)	9.36(±1.95)	910.07(±204.60)

Generally, the longer unit may lead to the complex trajectories, then the more HMM states is essential to fit the complex trajectories. The averaged objective score results in Table 5.11 and 5.12 indicate that the smallest number of HMM states, *nState*=1, is not able to represent the complex visual trajectories in both phoneme and dynamic viseme. An interesting result shows that the appropriate number of HMM states in both units is different. The optimal number of HMM states is 3 for phoneme units and 5 for dynamic viseme units. We found that the averaged phoneme durations ($\approx 95ms$) is shorter than the averaged dynamic viseme durations ($\approx 180ms$). Hence, this is the important reason that the number of HMM states for dynamic visemes units is larger than that for phoneme units.

5.3.1.7 Comparing phoneme, static viseme, and dynamic viseme Units

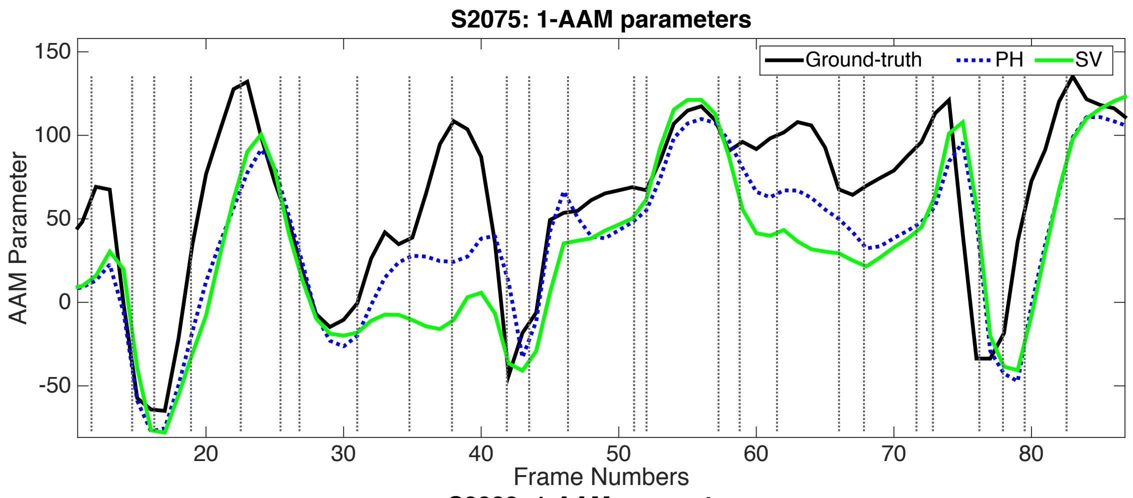
This experiment aims to find the best unit for HMM-based visual speech synthesis. We compare three approaches for synthesising AAM visual speech parameters: one that uses phone units as a basic unit, one that uses consonant viseme (static viseme) units as a basic unit, and another one uses dynamic viseme as a basic unit. Three HMM-based synthesisers using phones, static visemes, and dynamic viseme units were trained under the same training conditions and tested under the same testing set and measured in the same metrics. For the visual parameterisation, 30-dimensional AAM features were used for the KB2K audiovisual speech corpus which their delta and delta-delta coefficients were appended. These static and dynamic visual parameters are then upsampled from 29.97 fps to 100 fps. With phone and static viseme units, they are modelled using a left-to-right five state(three emitting), no skip HMM with a single component Gaussian with a diagonal covariance matrix. With dynamic viseme units, while, is modelled using a left-to-right seven states (five emitting), no skip HMM with a single component Gaussian with a diagonal covariance matrix. The full context label representation in Table 5.2 and the prepared question set are used to construct the decision tree based context clustering.

For testing, the ground-truth of the phone, static viseme, and dynamic-viseme transcriptions are used in this experiment because we would like to observe the best performance of our synthesiser from the perfect input. we measure the correlation, normalised root mean square error (NRMSE), and global variance (GV) between the generated AAM feature vectors and the corresponding original features.

The objective results are summarised in Table 5.13 as the mean (and standard deviation). We first compared our HMM-based approach in three different units with the baseline system from [98] that using unit selection approach with dynamic viseme units. It clearly shows that our HMM systems outperform baseline system especially correlation results. These might be originated from an insufficient dynamic viseme library that represent one lip-motion in each dynamic viseme. Regarding to HMM-based approach with three different units, the results suggest that the use of dynamic viseme units is better than the use of phone and/or static viseme units. In Figure 5.13, it demonstrates time-varying parameter sequences generated using the phoneme and static viseme HMMs compared with the ground-truth equivalent measured directly from the video. Clearly, both results (blue and green) were nearly the same and still not close to the ground-truth trajectories (black). While, the comparison between phoneme HMM and dynamic viseme HMM were shown in Figure 5.14. It can be seen that dynamic-viseme trajectories were closer to original than the phone unit. This can be concluded that dynamic viseme is the best unit for visual speech synthesis HMM systems as same as the previous application of dynamic visemes to synthesis used a sample-based approach by [98]. We refer to the supplementary video for animation results.

Table 5.13: The mean (±standard deviation) scores averaged for HMM-based approach using phonemes, traditional visemes, and triphones in KB2K corpus.

Systems	Unit	Correlation	NRMSE	GV
Unit-sel[98]	DV	0.63(±0.07)	10.56(±1.42)	1781.70(±354.53)
HMM	PH	0.78(±0.08)	9.92(±2.24)	819.18(±179.62)
HMM	SV	0.76(±0.08)	10.10(±2.23)	828.27(±190.63)
HMM	DV	**0.80(±0.07)**	**9.05(±1.86)**	**942.62(±197.33)**



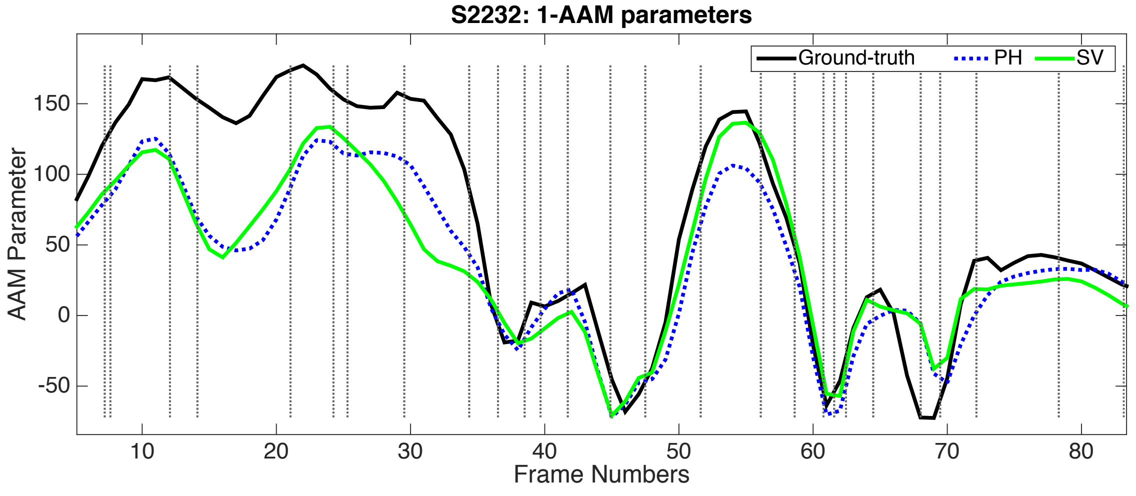


Figure 5.13: The time varying of the first AAM shape parameter as measured from video (black), synthesised using phones (blue) and static visemes (green). The vertical dashed lines mark the beginning and end of each phoneme.

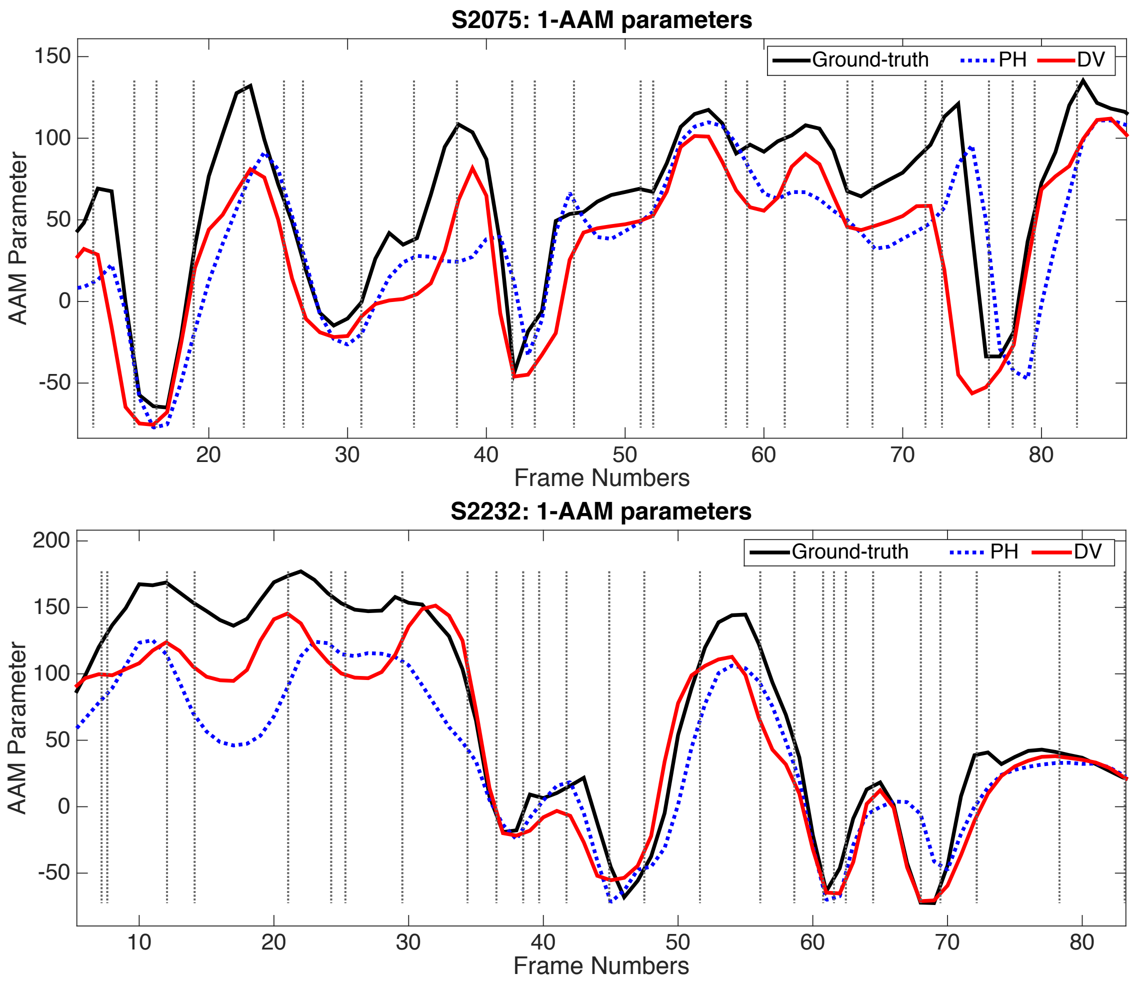


Figure 5.14: The time varying of the first AAM shape parameter as measured from video (black), synthesised using phones (blue) and dynamic visemes (red). The vertical dashed lines mark the beginning and end of each phoneme.

From the inspection of lip movements videos, the dynamic viseme sequences tended to have better articulation (in terms of mouth opening and audio/visual synchronisation for example) than phoneme and static viseme models. This is illustrated in Figure 5.14, where the overall trajectory of the AAM parameter exhibits greater variance, as shown in Table 5.13 and similar shape comparing with the ground-truth (this parameter largely controls mouth opening/closing). These results also confirm that dynamic viseme units are an effective unit compared with phoneme and static visemes in terms of representation accuracy and feasibility of mapping, as described in Section 4.4. With representation accuracy, each dynamic viseme can ensure that a movement of the visible articulators starts from the beginning of one articulation towards the next target. Whilst the beginning and ending of phoneme and static viseme units are generally random because they start and end with the boundaries of similar acoustic sounds. With the feasibility of mapping in the case of training and synthesising, it is easy to model and map a dynamic viseme to a sequence of AAMs because the same dynamic viseme represents similar lip-motion.

5.3.1.8 Analysis of contextual features

We aim to analyse the importance of each input contextual feature in Table 5.1. The context-dependent binary decision trees that were generated in the HMM training process are used to explore in this experiment. An example of binary decision trees of dynamic viseme HMMs is shown in Figure 5.15.

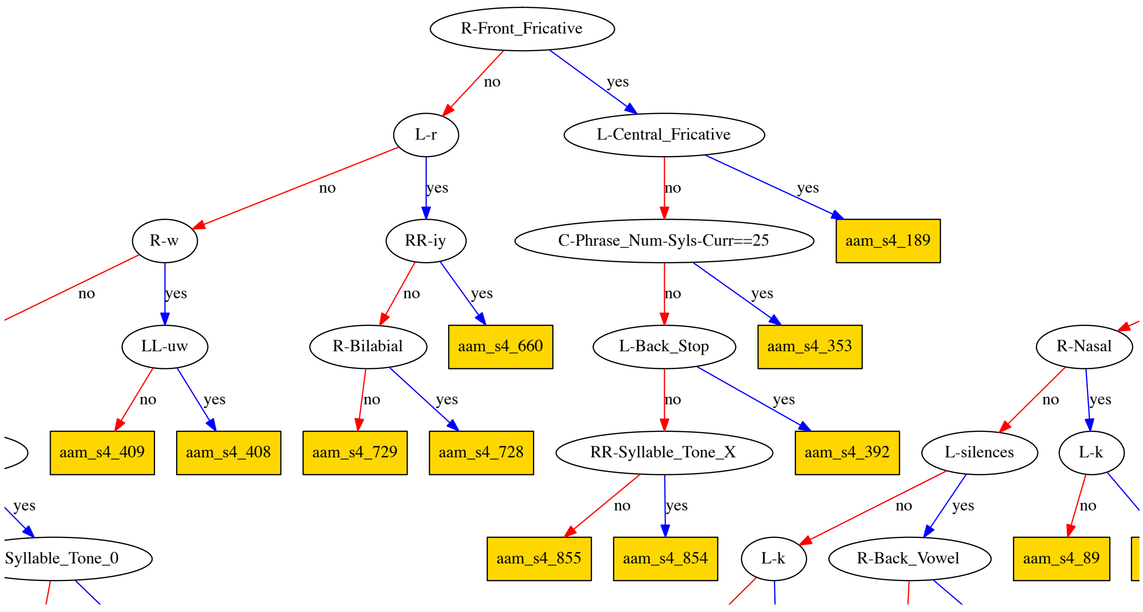


Figure 5.15: An example part of context-dependent binary decision tree that was generated from HMM-based phoneme units.

There are two assumptions used in this experiments. In the first assumption, we assume that the questions in the decision trees are essential. Hence, we analyse the contribution of each input contextual features by counting the number of the question that appears in the binary decision trees. Note that we found that each state has the same proportion, so we report only the total results for all states. Figure

5.16 shows top three highest proportions of all states for HMM-based phoneme, static viseme and dynamic viseme units. It can be seen that the most important input feature in each unit is quin-phone context (U1), quin-static viseme context (U2) or quin-dynamic viseme context (U3). Position and number of syllables in phrase (P1) and Position and number of phonemes,static viseme in syllable are the second and third highest proportions of tree occupancy in phonemes and static viseme units, respectively. Another interesting result in the case of using dynamic visemes as speech units, found that phonemes in dynamic viseme (U5) information is the second highest proportion. This concludes that phonetic information is still important even in dynamic visemes speech units. This additional information is the reason why dynamic visemes are the best unit and leads to the best system in Table 5.13.

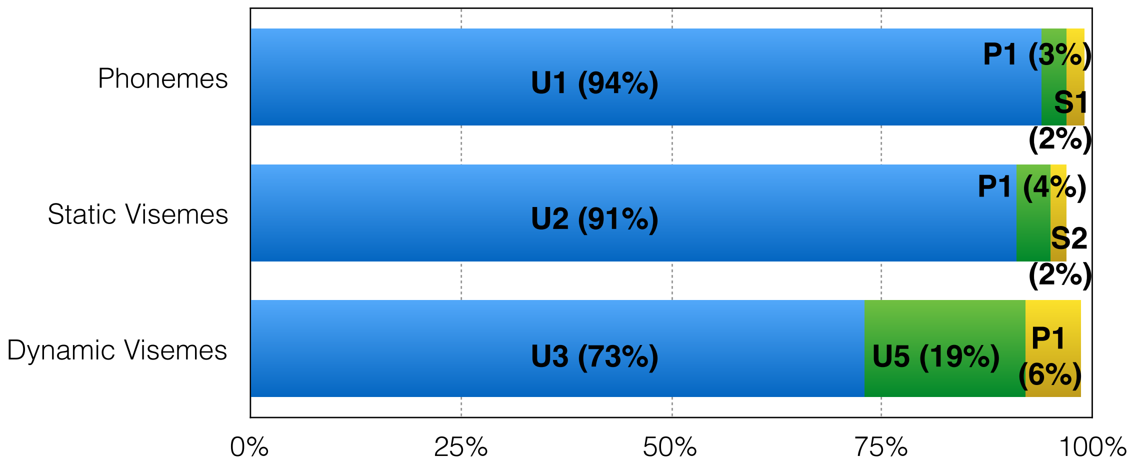


Figure 5.16: The percentage of top three highest proportions of tree occupancy for phoneme, static viseme and dynamic viseme units. Where U1, U2 and U3 denote for Quin-phone, Quin-static viseme and Quin-dynamic viseme context, respectively. The other input features can be found in Table 5.1. (first assumption)

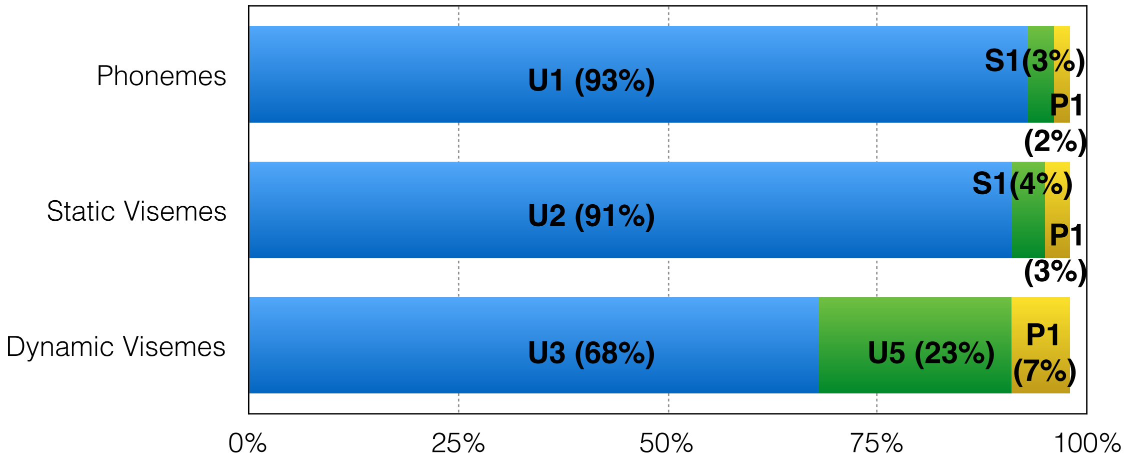


Figure 5.17: The percentage of top three highest proportions of dominance scores for phoneme, static viseme and dynamic viseme units. Where U1, U2 and U3 denote for Quin-phone, Quin-static viseme and Quin-dynamic viseme context, respectively. The other input features can be found in Table 5.1. (second assumption)

In the second assumption, we know that the decision tree is constructed from the root node to the leaf node and each node splits based on the minimal description length criterion. That means the depth of questions relates to the importance of the features because the question that appears near the root node is more important than the one further away. Hence, we compute a dominance score based on the reciprocal distance from the root node to the question node, as shown in Figure 5.17. It can be seen that the most important input level feature is quin-phone, static viseme or dynamic viseme feature. The second proportions of dominance scores in dynamic viseme units is phonemes in dynamic viseme (U5), which is the same as the result in previous assumption. This also indicates that both information is important for visual speech synthesis. It can be concluded that dynamic viseme information from U3 relates to visual information and phoneme in dynamic viseme information from U5 relates to audio information. Overall, these two assumptions result in the same trend and segment level is the most important for predicting lip movements from the full contextual input.

Chapter 6

Visual Speech Synthesis based on Feedforward Networks

6.1 Introduction

HMMs have been state of the art in (visual) speech synthesis for the past decade and typically employ decision tree clustered context-dependent models, although a drawback has been an oversmoothed output [31]. For example, a similar problem exits in audio speech synthesis [128] which can lead to a muffled sound being produced. The equipvalent in visual speech synthesis from HMMs tends to produce under-articulated lip movements [101]. Deep neural network (DNN) approaches have more recently been proposed to address these limitations in audio speech synthesiser and are able to learn a better model from multiple levels of non-linear transfer functions such as the use of multi-layer perceptrons with many hidden layers and numbers of units [6].

This chapter starts with an overview of a DNN-based approach for predicting visual lip motion parameters from a text input and aims to improve the resulting naturalness of the animation over the HMM-based system [102]. Firstly, from the text input, two kinds of speech units are considered. The first decomposes the input text into phonetic units. Although phonemes have been used widely in speech processing they have been shown to be suboptimal as visual speech units [73]. Instead, dynamic visemes (described in Section 4.4) are proposed as speech units and their performance compared to using phonetic units. In a third system, we combine both phoneme and dynamic viseme units. Secondly, we consider using more low-level (frame-based) contextual information in the feature vector applied to the DNN which is derived from the speech unit annotations, with the aim of producing a more realistic and smooth visual feature trajectory. These include the phonetic and dynamic viseme window context, the position and number of frames in phoneme and dynamic viseme, the forward phoneme span and the question of acoustic class (described in Section 6.2.1).

6.2 DNN-based Visual Speech Synthesis

This section describes how to incorporate DNNs into a visual speech synthesis system. Our proposed method focuses only on transforming text to visual features suitable for visual speech synthesis, and is based on a feed-forward neural network with a number of hidden layers. Our framework can be divided into two parts: training and synthesis, as illustrated in Figure 6.1. In the training part, the DNNs are trained based on an audiovisual-speech database. In the synthesis part, the visual active appearance model (AAM) parameters are generated from the models. Note that, this work was implemented by Keras [1] and it takes about 6 hours to train on UEA's High Performance Compute Cluster (GRACE) [2].

[1]https://keras.io

[2]https://rscs.uea.ac.uk/high-performance-computing

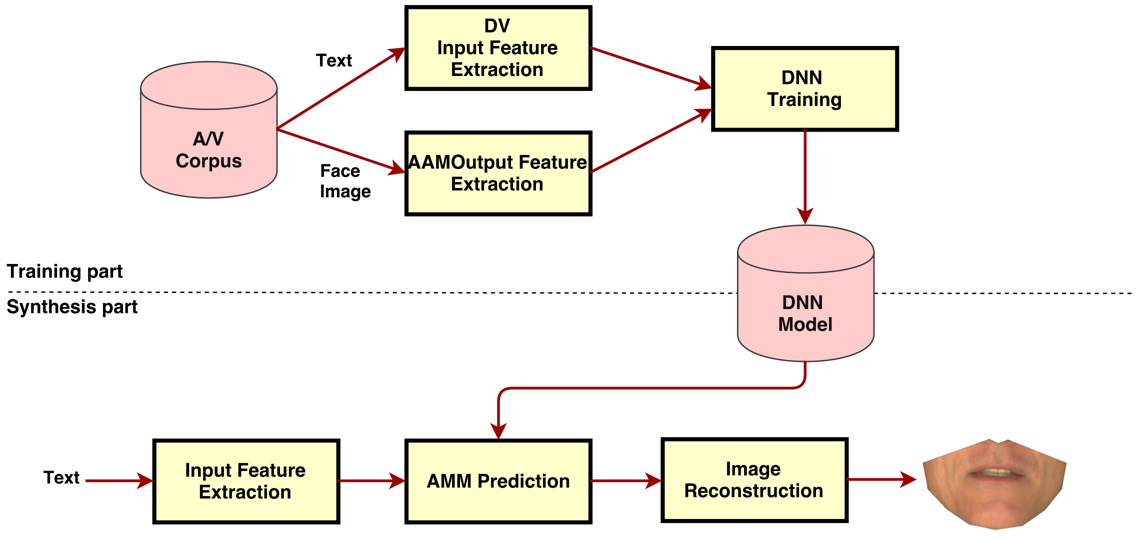


Figure 6.1: An overview of the DNN-based visual speech synthesis system.

For the training part, the given text and face image are converted to input features, **x**, and output features, **y**, using input feature extraction and output feature extraction (as discussed in the section below). The input and output features are time-aligned frame-by-frame. Then, in each hidden and output unit in the DNN training, a nonlinear activation function including a sigmoid, a hyperbolic tangent (tanh) and rectifier (ReLU) function is used to map all inputs from the previous layer to the next layer. Commonly, the activation function is controlled by connection weights and biases which are initialised by a uniform distribution function or a pre-training algorithm. The goal of training is to find an optimal set of weight parameters using the backpropagation algorithm, as described in Section 3.3.1. For neural network regression purposes, a nonlinear activation function is used for hidden layers while a linear activation function is adopted in the output layer.

In DNN synthesis, the input text is first converted into a sequence of features and the output sequence of visual active appearance model (AAM) features are predicted using forward propagation from the set of trained weights and biases. To avoid the over smoothing problem, DNN synthesis generates a sequence of visual parameters by using an arithmetic mean of its sequences instead of using parameter generation algorithm with dynamic features as in HMM-based synthesis. The output AAM features are then used to reconstruct the lip images, as described in Section 4.3.

Table 6.1: *Contextual features for phonetic (PH) and dynamic visemes (DV) units at varying levels.*

Level	Feature	PH	DV
Frame	Phonetic window context	x	
	Position and number of frames in phoneme	x	
	Forward phoneme span	x	
	Acoustic class	x	
	Dynamic viseme window context		x
	Position and number of frames in dynamic viseme		x
	Forward dynamic viseme span		x
Segment	Quin-phone context	x	
	Quin-dynamic viseme context		x
	Number of phonemes in dynamic viseme		x
Syllable	Position and number of phonemes in syllable	x	
	Position and number of dynamic visemes in syllable		x
Word	Position and number of syllables in word	x	x
Phrase	Position and number of syllables in phrase	x	x
	Position and number of words in phrase	x	x
Utterance	Position of syllable, word and phrase in utterance	x	x

6.2.1 Input Feature Representation

Feature extraction begins with the input text in both the training and synthesis parts. In a real world application, the input text is converted into either a time-aligned phoneme or dynamic viseme label sequence. The former can be generated automatically from, for example, automatic speech recognition system or from human annotation or from text to speech system and the latter can be learned automatically from the video as described in Section 4.4 or converted from phoneme sequences [99]. In this present work, we assume that we know the phoneme transcription and dynamic viseme transcription because we would like to proof the concept of the use of perfect input and determine the upper bound performance of the visual synthesis system. Hence the text-to-speech system and phoneme-to-dynamic viseme conversion are not required. To transform the sequence of phonemes and/or dynamic viseme units into a time sequence of visual features as required for visual synthesis, contextual labels at the phonetic/dynamic viseme and linguistic levels are extracted and used to create a suitable input feature vector. For HMM synthesis of visual vectors this level of contextual labelling is sufficient but for input into a DNN, to create smooth trajectories, it is necessary to include frame-level features. In practice many contextual factors affect the way people speak which includes the number of syllables in the current word, the phoneme/dynamic viseme context and the part-of-speech [73, 128]. We consider a number of such factors in our features and extract information at the frame, segment, syllable, word, phrase and utterance level. The importance of these is examined in Section 6.3.1.4 in terms of their effect on the synthesised visual features. The full set of features considered is summarised in Table 6.1 which shows those for phonetic units (PH) and for dynamic viseme units (DV).

6.2.1.1 Frame Level Features

This frame level features are essential and required for DNN because DNN is based on frame by frame synthesis framework. We consider a number of such factors in our features as shown in Table 6.1. Considering the **Phonetic window context** in Table 6.1, frame level phonetic context is included for the t frames preceding and ahead of the current phoneme which form the $2t + 1$ (K) dimensional per frame. After that, the conversion of phoneme symbol to binary format is applied using one-hot representation. This is known as one-hot because this is a zero vector except for a "1" at each phoneme individually. The size of the one-hot vector is the number of phonemes, $p = 41$. Hence, the total dimensionality of phonetic window context features is the $K\times$ 41 dimensional binary features. To simplify, Figure 6.2 shows how to extract the phonetic window context for the 2 frames preceding and ahead of the current phoneme, i.e. $K = 5$, of the word "Interspeech" at frame 15 and 18. From this Figure, a one-hot of /t/ and /er/ is "$0, ..., 1, ..., 0$" and "$0, ..., 0, ..., 1$",respectively. The total dimensionality of phonetic window context features in each frame is the $5 \times 41 = 205$ dimensional binary features.

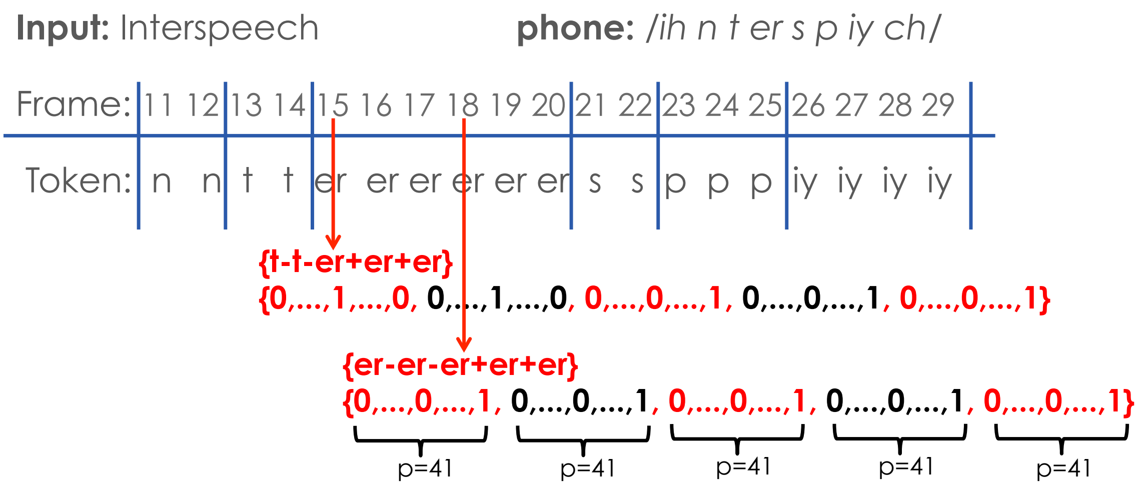


Figure 6.2: An example of phonetic window context feature extraction.

The **position** feature has three binary elements that correspond to whether the centre frame is at the *start (1,0,0)*, *middle (0,1,0)* or *end (0,0,1)* of the current phoneme, while **number** indicates how many frames are in the phoneme [128]. The **forward phoneme span** indicates for how many frames the current phoneme is present before changing to another phoneme [61]. An example of these three features: position, number of frames in phoneme and forward phoneme span are shown in Figure 6.3. At frame 24, the current frame is at the middle of the phoneme /p/ and there are three frames in this phoneme and it has one frame left before changing to another phoneme.

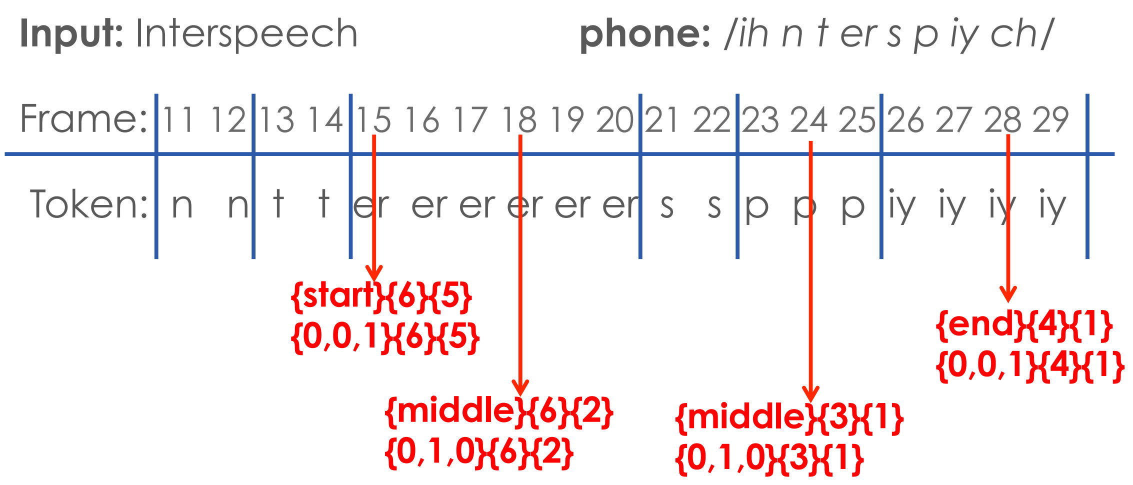

Input: Interspeech
phone: /ih n t er s p iy ch/
Frame: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Token: n n t t er er er er er er s s p p p iy iy iy iy
{start}{6}{5}
{0,0,1}{6}{5}
{middle}{6}{2}
{0,1,0}{6}{2}
{middle}{3}{1}
{0,1,0}{3}{1}
{end}{4}{1}
{0,0,1}{4}{1}

Figure 6.3: An example of position and number of frames in phoneme and forward phoneme span feature extraction.

Acoustic class is represented by a 57-D binary feature where each element is a response to 57 questions such as '*Is the phoneme voiced?*' or '*Is the phoneme nasalised?*', which are taken from the contextual questions in HTS [126]. All prepared questions are given in Appendix A. The acoustic class vector is a zero vector, except the position that correspond to the question and phoneme. For example, a "1" is assigned to the position $12, 17, 20, 31, 34, 40$ of acoustic class vector because these indices correspond to the question with phoneme /t/.

The final column of Table 6.1 shows a similar set of features defined for dynamic viseme units. These form longer binary features given that 160 dynamic visemes are used as opposed to 41 phonemes and no equivalent acoustic class feature exists as the units are visually-derived.

6.2.1.2 Segment Level Features

We define a segment as being five phonemes or five dynamic visemes in duration, centred about the middle unit, as following the standard setup from HMM system. Segments typically have an average duration about 95 ms and 183 ms for a phone and a dynamic viseme, respectively. The five phonemes in the segment are represented by the same one-hot method as used in the frame level features, which form a 41×5 dimensional vector. **Quin-phoneme context** binary features indicate the current, two preceding and two following phonemes. Similarly, **Quin-dynamic viseme context** is a 160 × 5 dimensional feature that indicates the five DVs in the segment. **Phonemes in DV** is a numeric feature representing the number of phonemes in the dynamic viseme. An example of segment level feature extraction is depicted in Figure 6.4 including "Quin-phoneme context","Quin-dynamic viseme context", and "Phonemes in DV" .

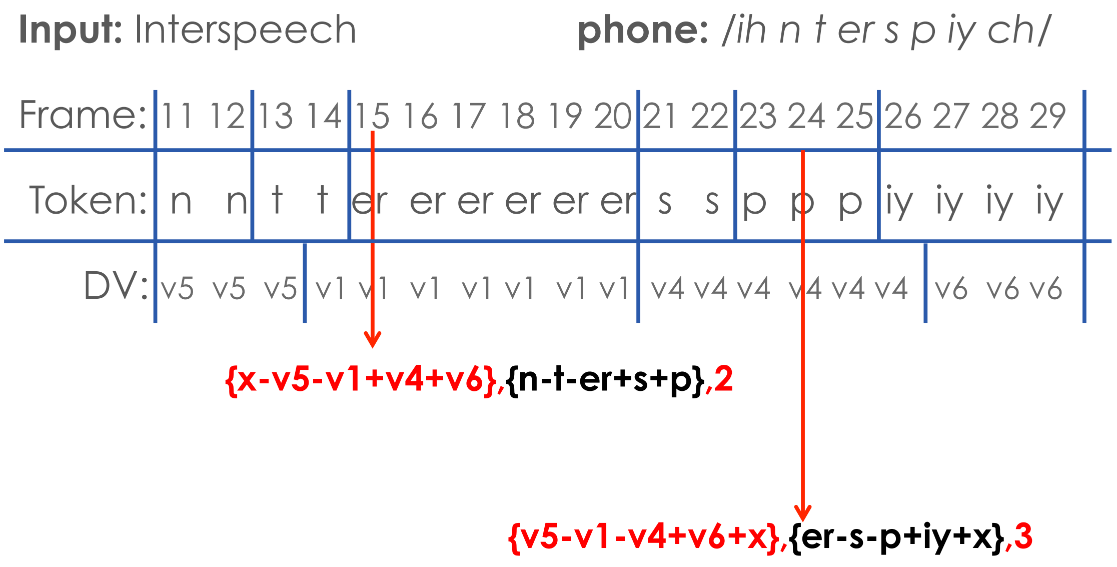

Input: Interspeech
phone: /ih n t er s p iy ch/
Frame: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Token: n n t t er er er er er er s s p p p iy iy iy iy
DV: v5 v5 v5 v1 v1 v1 v1 v1 v1 v1 v4 v4 v4 v4 v4 v4 v6 v6 v6
{x-v5-v1+v4+v6},{n-t-er+s+p},2
{v5-v1-v4+v6+x},{er-s-p+iy+x},3

Figure 6.4: An example of segment level feature extraction.

6.2.1.3 Syllable, Word, Phrase and Utterance Features

The syllable level features of **number** and **position** indicate how many phonemes or DVs are in the current syllable and the current position (*start*, *middle*, *end*, and *single*) within the syllable. At the word, phrase and utterance levels the number and position features indicate similar information but are no longer unique to phonemes or DVs. According to Figure 6.5, this shows the utterance structure of "Hello, world". This structure represents a relationship between a set of items such as a syllable, word, phrase, while these relationship between each items is used to extract the syllable, word, phrase and utterance features, for example, at frame 4 the features are as Table 6.2.

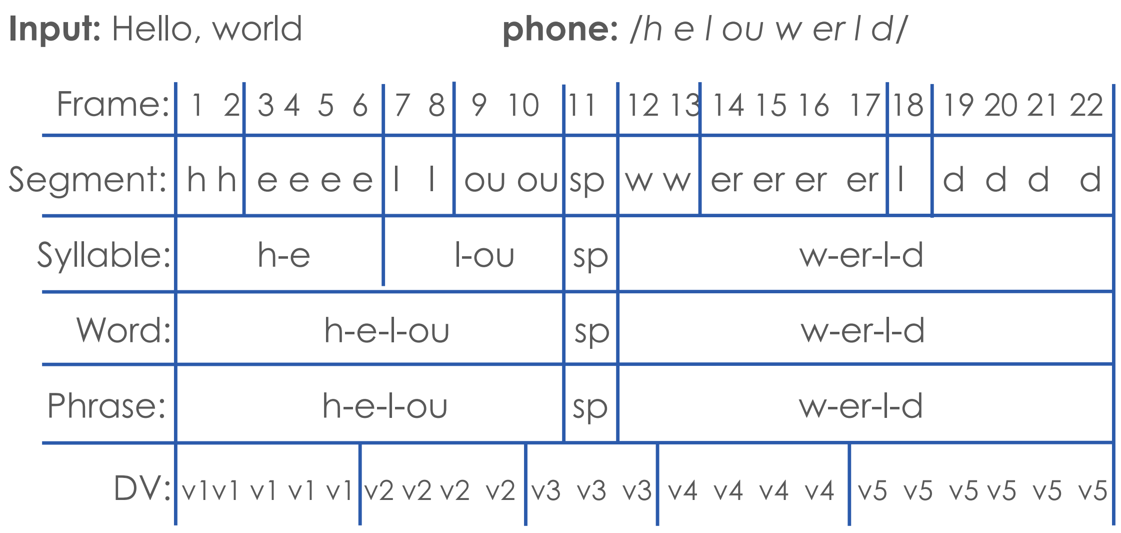

Input: Hello, world
phone: /h e l ou w er l d/
Frame: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Segment: h h e e e e l l ou ou sp w w er er er er l d d d d
Syllable: h-e l-ou sp w-er-l-d
Word: h-e-l-ou sp w-er-l-d
Phrase: h-e-l-ou sp w-er-l-d
DV: v1 v1 v1 v1 v1 v2 v2 v2 v2 v3 v3 v3 v4 v4 v4 v4 v5 v5 v5 v5 v5 v5

Figure 6.5: An example of segment, syllable, word, and phrase level feature extraction.

Table 6.2: *Contextual linguistic features of frame 4 in Figure 6.5 for phonetic (PH) and dynamic visemes (DV) units at syllable, word, phrase, and utterance levels.*

Level	**Feature**	**Representation**
Syllable	Position of phonemes in syllable	middle
	Number of phonemes in syllable	4
	Position of dynamic visemes in syllable	end
	Number of dynamic visemes in syllable	5
Word	Position of syllables in word	start
	Number of syllables in word	2
Phrase	Position of syllables in phrase	start
	Number of syllables in phrase	2
	Position of words in phrase	start
	Number of words in phrase	1
Utterance	Position of syllables in utterance	start
	Position of words in utterance	start
	Position of phrase in utterance	start

6.2.2 Output Feature Representation

As described in Section 4.3, a high dimensional, 1920×1080 block of pixels is transformed into a compact dimensional using active appearance model (AAM). In this work, AAM is used to track and parameterise the lower facial region in each frame of the video [20, 69]. From a set of 34 2-D vertices that define a mesh demarcating the contours of the lips, jaw and nostrils a 30-D AAM vector, $\mathbf{O}$, is extracted. Moreover, our experiments in the HMM-based system (Section 5.3.1.1) found that visual frame rates at 100 fps obtained the highest accuracy and the original frame rates of 29.97fps is insufficient for training. In this chapter, we examined the suitable number of visual frame rate for DNN-based system between 29.97fps and 100fps, as shown in Table 6.3. Note that, the input window of 367ms gives an equivalent window width of $K = 11$ and $K = 37$ frames when this window is applied to the frame rate at 29.97fps and 100fps, respectively. It can be seen that 29.97fps is a suitable frame rate for DNN visual speech synthesis and this frame rate will be used for all subsequent testing in this chapter. We refer to the supplementary video for animation results.

Table 6.3: *The averaged scores of phoneme DNN-based visual speech synthesis in different data rate.*

Data rate	Correlation	NRMSE	GV
29.97fps	0.87(0.05)	5.42(1.29)	1268.73(224.00)
100fps	0.86(0.05)	5.84(1.29)	1047.52(198.12)

6.2.3 Network Structure

This section describes our feedforward neural network structure. With DNN training, the scaling method was applied to numerical input features and output features as described in Section 4.5.1. The input features were then normalised to the range [0.01 0.99] using the min-max scaling method. Then, we applied min-max scaling and mean-variance normalisation to the output visual features to find the best scaling method for AAM parameters. A suitable network topology was chosen by using combinations of hyper-parameter as shown in Table 6.4. The different numbers of hidden layers (1-6) and numbers of nodes (128-4000) were used in our preliminary work on the validation data. The two common activation functions, a rectified linear unit activation function (ReLU) and hyperbolic tangent activation function (tanh), were also used to examine the best non-linear activation function for the hidden layers. A linear function was used for the output layer. The stochastic gradient descent (SGD) algorithm with mini-batch and momentum (as described in Section 3.3.1.3) was used to optimise the weights of the network. To avoid the problem of overfitting, dropout was varied between 0.1-0.9 and applied to hidden layers [94]. The maximum number of epochs was set to 150. However, the training process is then stopped early in the case the validation loss does not improve.

Table 6.4: *The combination of hyper-parameters for our preliminary tests.*

Hyper-parameters	**value**
Number of hidden layers	1,2,3,4,5,6
Number of units	128,256,512,1024,2048,3000,4000
Activation function for hidden layer	ReLU, tanh
Activation function for output layer	linear
Number of learning rate	0.000001-0.1
Number of momentum	0.5,0.7,0.9
Number of epochs	150
Input scaling method	min-max scailing
Output scaling method	min-max scailing, z-score normalisation
Optimisation	SGD with minibatch
Number of minibatch	100

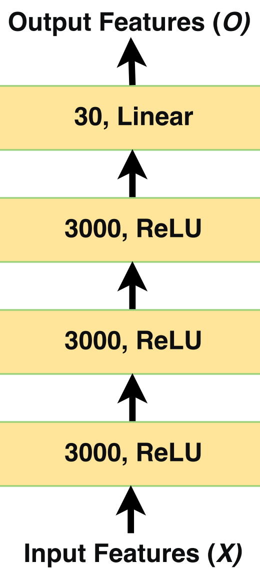


Figure 6.6: Finalised feedforward neural network architecture for visual speech synthesis.

The preliminary work on the validation set found that the best model comes from the feedforward network with three hidden layers each consisting of 3000 units. A rectified linear unit activation function was used for hidden layers and a linear activation function was employed at the output layer, as shown in Figure 6.6. Tests also found that a rectified linear unit activation function is slightly better than a hyperbolic tangent activation function. Mini-batch stochastic gradient descent was used and the size of mini-batch set to 100; learning rate and momentum were fixed to 0.1 and 0.9, respectively. The weights of the DNN were initialized randomly with no pretraining using Glorot uniform function [44]. The maximum number of epochs was set to 150. 50% dropout and early stopping were also applied to avoid overfitting on hidden units. Figure 6.7 illustrates the loss learning curve of training and validation sets during the training process that shows 74 is the maximum number of epochs for this setting.

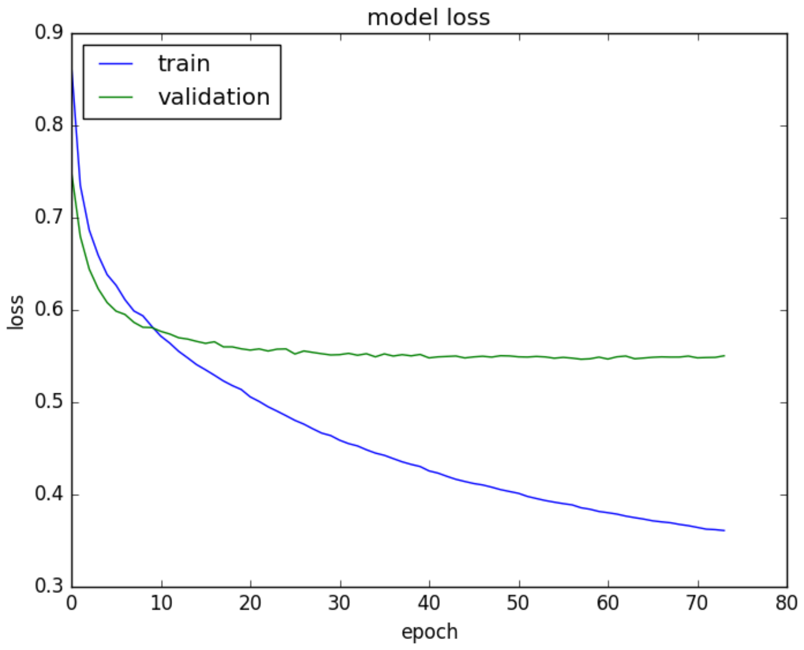


Figure 6.7: The loss learning curve of training and validation sets during the training process.

6.3 Experiment Results

This section aims to perform an analysis of the synthesised visual speech considering both objective evaluation and subjective evaluation. Experiments are performed on the KB-2k audiovisual speech dataset, as described in Chapter 4. We then partition this dataset into: a training set (80%), a validation set (10%), a testing set (10%), as described in Section 4.5.3. Table 6.5 summaries significant features of the dataset. As there is no standard unit for visual speech processing, we aim to find a suitable unit for visual speech synthesis using hidden Markov models (HMMs). Two kinds of speech units are considered in our experiments and include phoneme units and dynamic visemes units. We also consider the low-level (frame-based) contextual information in the input and output features vector, with the aim of producing a more realistic and smooth visual feature trajectory. Moreover, we also analyse the frame level features, with the aim of finding the most influential features.

Table 6.5: A summary of KB2K corpus for DNN-based.

Features	**Training**	**Validation**	**Testing**
AAM dimensionality	30	30	30
Number of unique phones	41	41	41
Number of unique dynamic visemes	160	160	160
Number of utterances	2,042	250	250
Number of frames	214,151	26,553	25,394

6.3.1 Objective Tests

For objective evaluations, we compare the first five coefficients of the predicted visual AAM features with the groundtruth AAM parameters using the same three objective metrics: (i) correlation, (ii) normalised-RMSE (NRMSE), and (iii) global variance (GV), as described in Section 5.3.1. Note that correlation and RMSE are common objective evaluation techniques for visual speech synthesis. In this work, we also propose a global variance measure because the variance of the AAM features is related to the naturalness of the synthesised lip animation in terms of over- and/or under-articulation.

6.3.1.1 Effect of contextual input

To include contextual information, and improve the resulting predicted visual contour, a sliding window is used so that frame level information preceding and ahead of the current frame is included. The width of the input window needs to be wide enough to capture a useful context information and avoid the effect of visual coarticulation problem. [61] reported that a satisfactory window width of K = 11 frames applied to 29.97 frame-per-second (fps) data which equates to a width of 367 ms. Our findings in Table 6.6 observed the performance over the validation set for various number of the input window between 100ms and 567ms. Clearly, the correlation results in each configuration are nearly the same and the NRMSE results have decreased when increasing the width of the window, and GV have gone up and gone down after setting the window more than 367 ms. From the inspection of lip animation, it can be observed that the small NRMSE from the larger context window tends to generate under articulated lip animation. Hence, it can be concluded that a input window of 367ms is the satisfactory value which corresponds to the highest correlation and GV with an acceptable NRMSE.

Table 6.6: *Objective scores for finding the suitable number of input window width computed on the validation set.*

Window length	Correlation	NRMSE	GV
100ms	0.84(±0.05)	6.14(±1.56)	1127.61(±206.19)
167ms	0.85(±0.05)	5.98(±1.44)	1184.78(±209.54)
234ms	0.86(±0.05)	5.82(±1.39)	1159.01(±209.75)
300ms	0.86(±0.05)	5.76(±1.30)	1242.53(±224.21)
367ms	0.86(±0.05)	5.72(±1.29)	1255.02(±214.73)
434ms	0.86(±0.05)	5.70(±1.29)	1185.67(±207.27)
500ms	0.86(±0.05)	5.68(±1.25)	1164.43(±206.14)
567ms	0.86(±0.05)	5.64(±1.24)	1152.21(±204.93)

The input window of 367ms gives an equivalent window width of $K = 11$ frames (5 preceding and 5 ahead) and $K = 37$ frames (18 preceding and 18 ahead) when this window is applied to the orignal video rate at 29.97fps and the upsampled video rate at 100fps, respectively. In our preliminary work, we observe that the appropriate number of data rate for DNN-based system is 29.97fps which is used for all subsequent testing.

6.3.1.2 Effect of contextual output

The motivation for this experiment is intended to determine the suitable number of output window as we observed the number of input window in Section 6.2.1. Ideally, the output window must large enough to avoid discontinuous and over-smoothing of features. The input window of 360ms (described in Section 6.2.1.1) is used to capture the input context and then used to find out the best number of output window width, as shown in Table 6.7. We observe that an output window of 100ms is the appropriate number that use to capture the smoothness of the output according to the highest correlation and GV. Note that, the output window of 100ms gives an equivalent window width of $K_{out} = 3$ frames when applied to the orignal video rate at 29.97fps. This means the same t^{th} frame may be prepared and predicted K_{out} times, which form a vector of AAM parameters (30 AAMs per facial image) times the output window length, $30 \times 3 = 90$ dimensional vector. In the synthesis part, the final output in each frame could be averaged using the frame-wise mean.

Table 6.7: *Objective scores for finding the suitable number of output window width computed on the validation set.*

Window length	Correlation	NRMSE	GV
33ms	0.86(±0.05)	5.72(±1.29)	1255.02(±214.73)
100ms	0.87(±0.05)	5.42(±1.29)	1268.73(±224.00)
167ms	0.88(±0.04)	5.32(±1.25)	1234.91(±219.14)
234ms	0.88(±0.05)	5.28(±1.27)	1218.99(±220.73)
300ms	0.88(±0.05)	5.29(±1.24)	1193.93(±217.31)

6.3.1.3 Effect of Frame Level Feature

Since the HMM-based system, as described in Section 5, makes state predictions, the frame level features are not necessary. However the DNN system makes frame-level predictions, and so frame level features should be of benefit to the predicted AAM parameters.

To investigate the effect of frame level feature on phone units (PH), we compared two settings regarding the frame level feature and the features from the other levels, as defined in Table 6.1. The first setting, System $F1$, represents the input features without frame level features, but includes feature for segment, syllable, word, phrase, and utterance (more details in Table 6.1). The second setting, System $F2$, uses all features which combines the features in the first setting with frame level feature.

The blue line in Figure 6.8 shows one limitation of "$F1$ System", that represents the input features without frame level features. It shows that "$F1$" cannot generate smooth AAM parameter for the utterance "If dark came they would lose her". In this experiment the model generates one output per phoneme, resulting in the step trajectory at each phoneme boundary and becomes an unrealistic output. Two conventional ways to overcome this problem in the (visual) speech synthesis system exist. Firstly, the smoothing process is considered using the speech parameter generation algorithm (MLPG) with dynamic features [128]. Secondly, more advanced neural network architectures were used to smooth predicted visual speech output, for example unidirectional LSTM with RNN output [125] and bidirectional LSTM-RNN [36]. Later work in Chapter 7, considers this approach.

Our preliminary work found that the smoothing MLPG with dynamic features is able to avoid discontinuous visual lip trajectories, but introduces under smoothing trajectories. Hence, we investigate an alternative way of avoiding discontinuous AAM features by including frame level information as one of the inputs into the DNN. The red line in Figure 6.8 (System $F2$) indicates the importance of the frame level information for DNN-based systems which is now able to avoid discontinuities in the visual feature sequence and produce a smooth and more realistic output. It can be claimed that the use of frame level features overcomes the step trajectories problem. The animation results of System F1 and System F2 can be found in supplementary video.

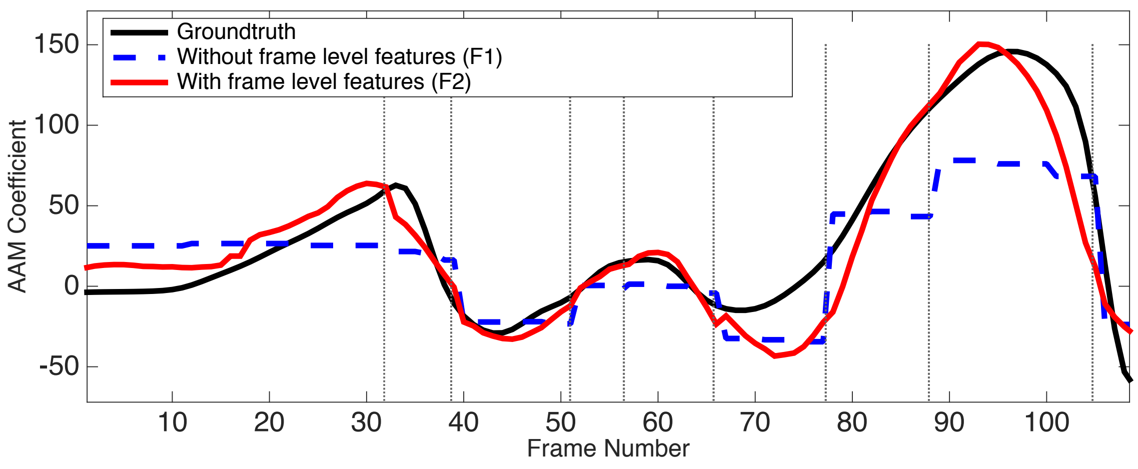


Figure 6.8: A ground-truth (black) AAM parameter trajectory compared with two synthetic AAM parameter trajectories from F1 (blue) and F2 (red). "F1 System" represents the input features without frame level features. While, "F2 system" includes input features into the input representation. Vertical lines show phoneme boundaries.

6.3.1.4 Optimisation of Frame Features

The previous experiment has shown that the frame level features are very important for DNN-based systems. However, we do not know the contribution of each feature in the frame level. This next experiment analyses the effect on prediction accuracy of the four different frame level features. Seven frame level feature combinations with features for segment, syllable, word, phrase and utterance (as defined in Table 6.1) are defined and summarised in Table 6.8.

System A-C were used to find the most influencial feature in the frame level feature. System A is inspired by [36] and [128]. In [36], the phoneme state is obtained from forced alignment of audio speech, then used its state combining with the number of frames in its corresponding phoneme to distinguish the input features in each frame. In our experiment, however, we consider classifying this feature into three categories: start, middle, and end [128]. This based on their coarse position within the current frame in the current phoneme instead of using phoneme state HMM. Moreover, we also include the number of current frames into the input features. System A uses only the the frame position and number features in each phoneme. System B uses only the phonetic window context and system C uses only the acoustic class of the phonetic window context. System D combines System B and C and adds the forward span phoneme feature. System E represents all features excluding the acoustic feature. System F combines the features in System A and B and adds the acoustic feature. System G includes all features.

Table 6.8: *Frame level feature combinations.*

Input feature for phone units (PH)	A	B	C	D	E	F	G
Position and number of frames in phoneme	x				x	x	x
Phonetic window context of k frame		x		x	x	x	x
Acoustic class of frame level feature			x	x		x	x
Forward phoneme span				x	x		x
Segment, syllable, word, phrase, utterance feature level	x	x	x	x	x	x	x

Table 6.9: *Correlation, NRMSE, and GV performance of phonetic frame level feature combinations on the validation set (brackets show ±standard deviation).*

	Correlation	NRMSE	GV
System A	0.73(±0.07)	10.54(±2.14)	802.89(±162.42)
System B	0.80(±0.07)	9.53(±2.19)	920.22(±174.20)
System C	0.73(±0.08)	10.61(±2.16)	840.36(±174.40)
System D	0.81(±0.07)	9.45(±2.12)	922.10(±189.92)
System E	0.79(±0.07)	9.65(±2.24)	955.86(±190.35)
System F	0.81(±0.07)	9.31(±2.22)	952.40(±183.90)
System G	0.81(±0.07)	9.26(±2.22)	964.55(±200.50)

Table 6.9 confirmed that the phonetic window context feature in System B makes the most significant contribution to the continuous AAM output. Hence, we believe that the combination of System B with other frame level features (e.g. System D-G) will give the better results. Table 6.8 presents correlation, RMSE, and GV results of the seven systems which shows that including all frame level features gives the

best performance (System G). The difference between Systems A and B and System B and C are shown to be large and is confirmed when looking at AAM coefficients as discussed earlier. The objective scores of Systems D, E, and F shows a slight improvement when comparing with the best (System G. To conclude, the results from System A-I indicate that the most influential feature is the phonetic context feature and the most second influential feature is acoustic class. The acoustic class is the most second influential feature because the objective scores for System E dropped while System D and F were nearly the same as using all features (System G). Moreover, the combination of all features is useful and needed, and results in the best scores when all features are considered (System C).

Firstly, the use of the specific number of current frames and position of the frame in the phoneme (System A) does not help the model to distinguish the differences between each frame. The features are not able to generate smooth AAM trajectories and produces an unrealistic lip animation, as shown in Figure 6.9. The difference between System A and the previous experiment is that System A produces approximately three different lip images per phoneme depending on the current position of frame, while the previous experiment predicts a single output in each phoneme.

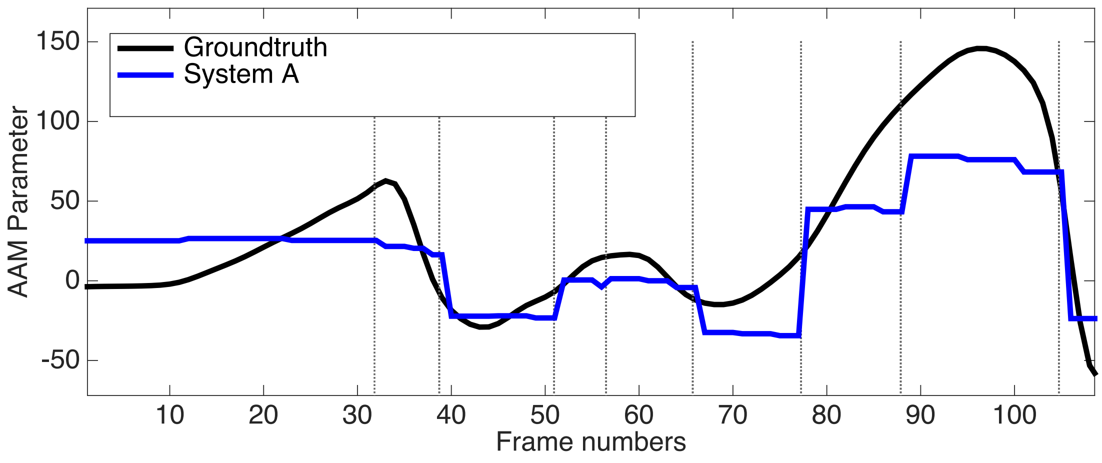


Figure 6.9: A ground-truth (black) AAM parameter trajectory compared with a synthetic AAM parameter trajectory from System A (blue). System A represents the input features with position and number of frames in phoneme. Vertical lines show phoneme boundaries.

System B is inspired by [61], in which they include contexual frame level information into the input features instead of using just knowledge of the central phoneme. A sliding window is used so that frame level information preceding and ahead of the current frame is included. In this experiment, we set the width of the window, K, to 330 ms, which reported as a satisfactory value from [61]. In this work the visual frame rate is 100 fps which gives an equivalent window width of $K = 33$ frames (16 preceding and 16 ahead). Figure 6.10 illustrates an example AAM trajectory between ground-truth and System B compared to System A, it shows that the AAM trajectory contour changed from being discontinuous into continuous by including phonetic window context. This can be confirmed by viewing the selected frames 26-32 from sequence S0524 in the KB-2k dataset that the lip sequences of System B were closely related to that of the groundtruth AAM parameters, as illustrated in Figure 6.11.

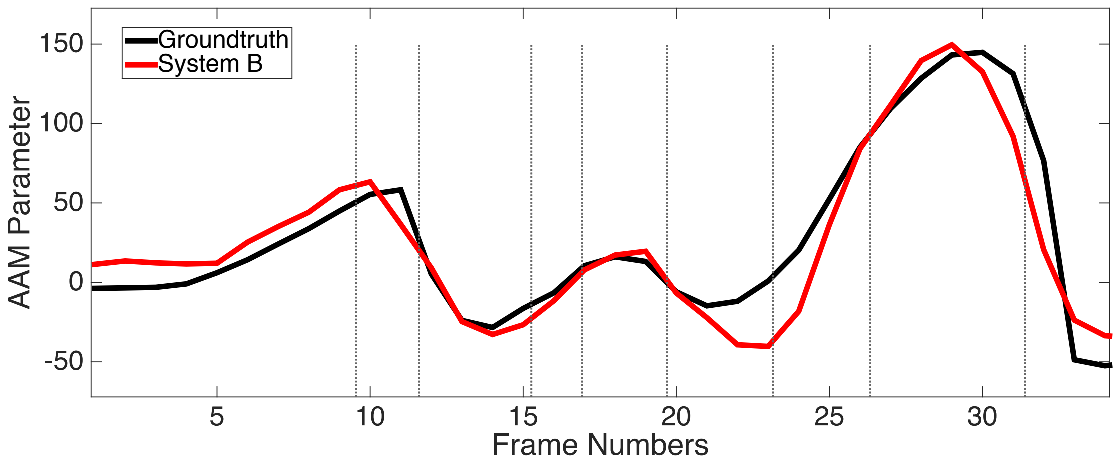


Figure 6.10: A ground-truth (black) AAM parameter trajectory compared with a synthetic AAM parameter trajectory from System B (red). System B represents the input features with phonetic window context features. Vertical lines show phoneme boundaries.

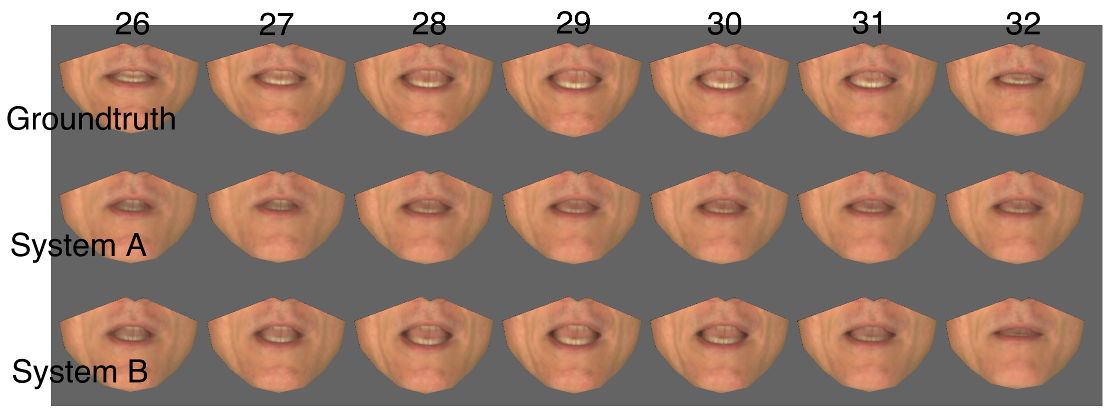


Figure 6.11: Selected frames 26-32 from the sequence (S0524) corresponding to the word "came". Each row shows an equivalent lip shape video that reconstructed from AAM parameters. Row 1 correspond to the groundtruth parmeters. Rows 2 and 3 correspond to System A and System B. The lip shape sequences of System A are nearly the same images in each frame, resulting in an unrealistic lip animation. While the synthesised lip motion of System B are closely to the groundtruth sequences.

6.3.1.5 Comparing Phoneme and Dynamic Viseme Units

An investigation is now made into the effect of using either phonetic units or dynamic viseme units to find the most suitable unit for visual speech synthesis. For the phoneme based system all PH features shown in Table 6.1 are included while for the dynamic viseme system all DV features are included. A third configuration was also tested which combines the phonetic and dynamic viseme unit features and includes all features shown in Table 6.1. For comparison, a baseline HMM synthesis system was created which used five-state hidden semi-Markov models with each state modeled by a single Gaussian with diagonal covariance. Quinphone HMMs were created using decision tree clustering that considered phoneme, syllable, word, phrase and utterance level questions.

Table 6.10: *Correlation, NRMSE, and GV performance of HMM and DNN approaches using phonemes and dynamic viseme units.*

	Correlation	NRMSE	GV
Phoneme HMM (±Baseline)	0.75(±0.08)	10.31(±2.07)	889.96(±177.95)
Phoneme DNN	0.81(±0.07)	9.26(±2.22)	964.55(±200.50)
Dynamic-viseme DNN	0.80(±0.06)	8.82(±1.26)	1093.12(±232.51)
Phoneme + DV DNN	0.87(±0.05)	7.35(±1.31)	1209.38(±233.24)

Table 6.10 shows correlation, NRMSE, and GV for the phoneme, DV and combined phomeme-DV systems using DNNs and the phoneme-based HMM system. The results show good improvement over the baseline HMM system regarding a phoneme DNN system. Moreover, both the phoneme DNN and DV DNN systems outperform the HMM synthesis approach. A more realistic and smooth visual feature trajectory can be found when combining phoneme and DV features which we attribute to their complementary information, one relating to acoustics and the other to visual information. This leads to an improvement of the resulting naturalness of the lip animation. Figure 6.12 illustrates an example inspection video which shows that the combination of phoneme and dynamic viseme units in frame-by-frame synthesis is able to reduce the lack of audio/video synchrony. It can be seen that the reconstructed lip shape video from the DNN system and groundtruth have the same lip shape at each time step, while the lip shape from HMM system happens later than it should. For example, the word "chicken" starts at frame 25 for the groudtruth video, but pronunciation of this word starts at frame 27 in HMM system. With the under-articulation problem, the lip motion of HMM system appear correct, but is somewhat under articulated. We refer to the supplementary video for animation results.

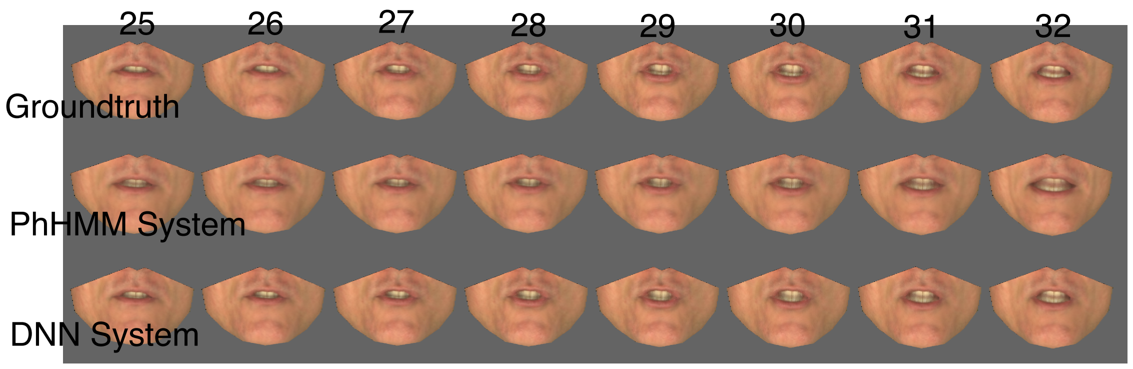


Figure 6.12: Selected frames 25-32 from the sequence (S0980) correspond to the word “chicken”. Each row shows an equivalent lip shape video that reconstructed from AAM parameters. Row 1 correspond to the groundtruth parmeters. Row 2 and 3 correspond to the phoneme HMM system and PhDV DNN system.

Chapter 7

Visual speech synthesis based on LSTM-RNN

7.1 Introduction

The aims of this chapter are to overview and develop a sequence to sequence method of statistical parametric visual speech synthesis using a bidirectional LSTM-RNN. It begins with an overview of visual speech synthesis system using an encoder-decoder architecture with a bidirectional LSTM. In experiments, both objective and subjective tests are carried out to evaluate the effectiveness of various configurations. Our experiments focus on how the overall quality of the synthesised visual speech is improved by comparing with a baseline HMM-based system (described in Chapter 5) and DNN-based system (described in Chapter 6).

7.2 Encoder-decoder LSTM-RNN visual speech synthesis

The proposed method of transforming text to lip animation, which is based on an encoder-decoder RNN structure using an LSTM with a number of hidden layers is shown in Figure 7.1. A text input is first converted to a sequence of contextual features which comprises a combination of binary features for categorical contexts (e.g. phonetic labels) and numerical features to represent values (e.g. a number of phonemes in a syllable). Specific details of the input features are given in Section 7.2.1. The output features are visual features (specifically active appearance model (AAM) features), and more details can be found in Section 7.2.2. Note that, this work was implemented by Keras and it takes about 16 hours to train on UEA's High Performance Compute Cluster (GRACE).

For training the neural network model part, the input and output sequences are first converted to context-truncated blocks, more details can be found in Section 7.2.3.1. After that, the encoder-decoder RNN architecture is used with the LSTM-RNNs to train the neural network model from all blocks from the encoder layer to the decoder layer. The goal of training the neural network (NN) model is to find an optimal set of weight and bias parameters, and this uses the context-truncated backpropagation through time algorithm. For neural network regression purposes as we aim to predict the AAM parameters as an output, a linear activation function is adopted in the output layer.

For the lip animation synthesis part, the input text is first converted into a sequence of features and the output sequence of visual features computed using forward propagation from the set of trained weights and biases. The output features comprise a sequence of AAM visual features corresponding to each frame. Finally, a rendering module re-synthesises a lip animation using the smoothed static AAM parameters [69].

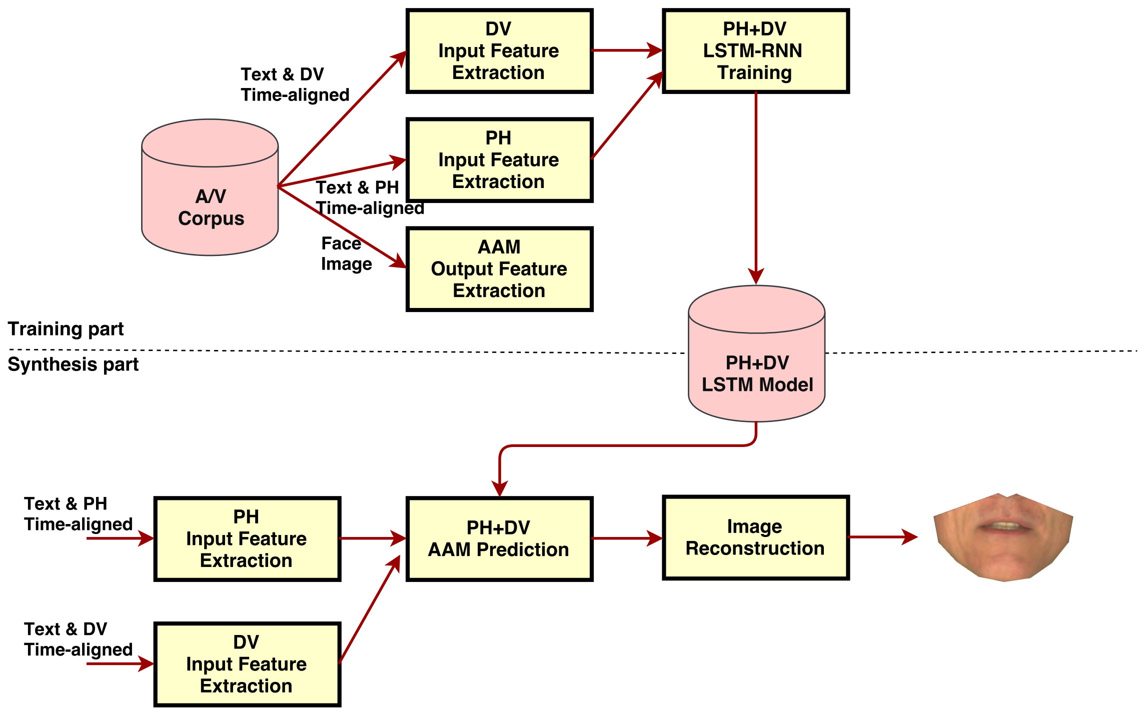


Figure 7.1: An overview of our visual speech synthesis system.

The next sections explain three major modules in Figure 7.1 including input feature extraction, output feature extraction, and LSTM-RNN trainig and predicting modules.

7.2.1 Input features representation

This section describes how input features in each time step are represented. Feature extraction begins with either a time-aligned phoneme sequence or a dynamic viseme sequence that can be generated automatically from, for example, HMM decoding or from human annotation. To transform this sequence of speech units into a time sequence of visual features as required for visual synthesis, contextual labels at the phonetic/dynamic viseme and linguistic levels are extracted and used to create a suitable feature vector. For HMM synthesis of visual vectors this level of contextual labelling is sufficient but for input into a neural network (NN), to create smooth trajectories, it is necessary to include frame-level features. In practice, many contextual factors affect the way people speak and include the number of syllables in the current word, the phoneme/dynamic viseme context and the part-of-speech. We consider a number of such factors in our features and extract information at the frame, segment, syllable, word, phrase and utterance level. The importance of these on the synthesised visual features is examined in Section 7.2.3.1. The full set of features considered is summarised in Table 7.1 which shows those for phonetic units (PH) and for dynamic viseme units (DV).

7.2.1.1 Frame level: input features

As we know so far from the feedforward structure, the use of sliding window frame level information preceding and ahead of the current frame significantly improves the performance of visual speech synthesis. However, one of the limitations of feed-forward networks is that the output at time t depends only on the input at that time step. That is the main reason for including contextual sliding frame information into the input for each frame. In this chapter, the contextual information is not needed in the input features because our structure is based on RNNs, which have the ability to access information from the previous time steps input. Therefore, we use only the current phoneme and dynamic viseme in each frame.

Considering the frame level features in Table 7.1, the **centre phoneme** feature is a 41-D binary feature that indicates the phonetic class of the current frame. The **position** feature has three binary elements that correspond to whether the current frame is at the *start*, *middle* or *end* of the current phoneme, while **number** indicates how many frames are in the phoneme [128]. The **forward phoneme span** indicates how many frames of the current phoneme are present before changing to another phoneme [61]. **Acoustic class** is represented by a 57-D binary feature where each element is a response to questions such as '*Is the current phoneme voiced?*' or '*Is the current phoneme nasalised?*', which are taken from the contextual questions in HTS [126], more details can be found in Appendix A. The final column of Table 7.1 shows a similar set of features defined for dynamic viseme units. These form longer binary features given that 160 dynamic visemes are used as opposed to 41 phonemes and no equivalent acoustic class feature exists as the units are visually-derived.

Table 7.1: *Contextual features for phonetic (PH) and dynamic visemes (DV) units at varying levels.*

Level	Feature	PH	DV
Frame	Current phoneme	x	
	Position and number of frames in phoneme	x	
	Forward phoneme span	x	
	Acoustic class	x	
	Current dynamic viseme		x
	Position and number of frames in dynamic viseme		x
	Forward dynamic viseme span		x
Segment	Quin-phone context	x	
	Quin-dynamic viseme context		x
	Number of phonemes in dynamic viseme		x
Syllable	Position and number of phonemes in syllable	x	
	Position and number of dynamic visemes in syllable		x
Word	Position and number of syllables in word	x	x
Phrase	Position and number of syllables in phrase	x	x
	Position and number of words in phrase	x	x
Utterance	Position of syllable, word and phrase in utterance	x	x

7.2.1.2 Segment level: input features

We define a segment as being five phonemes or five dynamic visemes in duration, centred about the middle unit, as preliminary tests found this to give best performance. The five phonemes in the segment are represented by the 41×5 dimensional **Quin-phone context** binary feature that indicates the current, two preceding and two following phonemes. Similarly, **Quin-dynamic viseme context** is a 160×5 dimensional feature that indicates the five DVs in the segment. **Phonemes in DV** is a numeric feature representing the number of phonemes in the dynamic viseme.

7.2.1.3 Syllable, word, phrase and utterance: input features

The syllable level features of **number** and **position** indicate how many phonemes or DVs are in the current syllable and the current position (*start*, *middle* or *end*) within the syllable. At the word, phrase and utterance levels the number and position features indicate similar information but are no longer unique to phonemes or DVs.

7.2.2 Visual: output features extraction

An active appearance model (AAM) is used to track and parameterise the facial region in each frame of the video [20, 69]. From a set of 34 2-D vertices that define a mesh demarcating the contours of the lips, jaw and nostrils a 30-D AAM vector, $\mathbf{y}$, is extracted. More details are given in Section 4.3. The preliminary tests in previous chapter about feedforward neural network examined visual frame rates of between 30 fps and 100 fps and established highest accuracy was with 30 fps which is used for all subsequent testing. Specific details are given in Section 4.4.

7.2.3 Network structure

As described earlier in Section 3.4.1, there are four different types of RNN architectures that can be applied to sequence to sequence tasks. We also pointed out the drawbacks of the basic BPTT training method in Section 3.4.4.1. Therefore, this section examines the effect of the type of RNN architecture and the BPTT training method that we propose in this work.

7.2.3.1 Context-truncated BPTT

This section explains the benefits of a context-truncated BPTT algorithm over two traditional RNN training algorithms: epochwise BPTT and truncated BPTT. As mentioned earlier, the limitation of the two RNN training methods is that they require utterance level processing (described in subsection 3.4.4.1). With epochwise BPTT, back-propagation error and weights are updated after seeing the whole sequence. The resulting latency is a problem because RNNs require a training sample sequence from the start to the end. That means this method cannot be processed in parallel and requires large amount of memory for long sequences. To avoid memory problems, the truncated BPPT uses a fixed number of time steps to reduce the storage aspect. Setting the amount of truncation is difficult because if set too short the performance will likely be worse than the epochwise BPTT. Additionally, the parallel issue of the truncated BPTT method still exists because the next truncated examples need to wait for the memory output from the previous state in the case of an unfinished sequence.

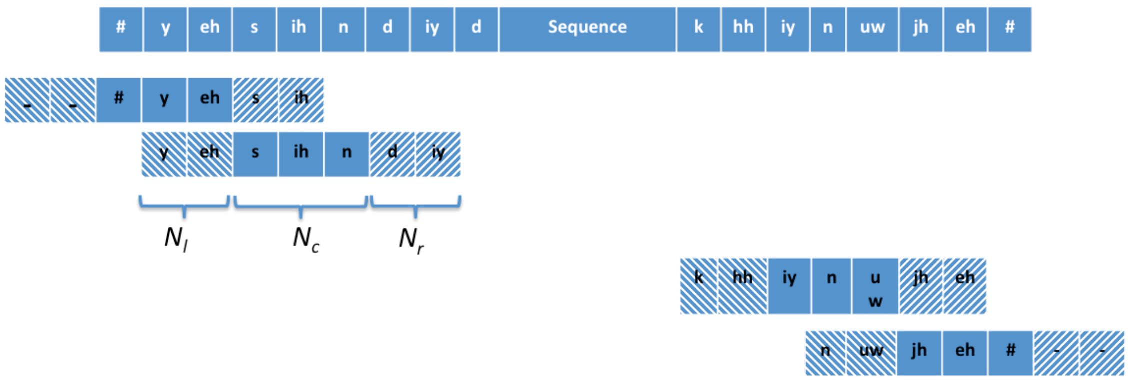


Figure 7.2: An example of context-truncated block with a single frame overlap. The sequences are first partition to blocks and each block comprises N_c length. Then, left, N_l , and right, N_r, contexts are appended.

Our approach to this problem is inspired by Chen et al. [13], as shown in Figure 7.2. For the sake of simplicity, this figure assumes that there is one phoneme symbol as an input feature per frame. The general idea of this approach combines aspects of epochwise BPTT with truncated BPTT. The first idea is to process local dependency instead of global dependency. Hence, each sequence is split into fixed of length, N_c, called a truncated block. Then, if each block acts as the local dependency, the epochwise BPTT can then be applied to each block. With this approach, the performance is not acceptable because the information at the beginning and ending of each block is lost. Due to the lost or incomplete information in each block, more information is added to the beginning and end of the truncated blocks, which we now refer to as a context-truncated block. The left contextual frame has length, N_l, and the right contextual frame has length, N_r. These are appended to each truncated block. The appended frame is used to avoid the incomplete information, hence there is no output. Moreover, empty frames are appended to the first truncated block and last truncated block for blocks shorter than N_c. From this appended idea, each block has no need to preserve the network state to the next block as with the truncated BPTT method and can be made as efficient as in mini-batch training as classical feedforward networks.

The truncated block can be split into context-truncated with or without overlap. We believe that the overlapping method might benefit the training process due to increasing data augmentation. With testing, if overlapping is applied, the same t^{th} frame may be generated N_t times, as denoted by $\{\mathbf{y}_1^t, .., \mathbf{y}_{N_t}^t\}$. Hence, the final output could be averaged by an arithmetic mean (eq. 7.1) as follow:

$$\mathbf{y}^t = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathbf{y}_i^t \tag{7.1}$$

7.2.3.2 Encoder-decoder LSTM-RNNs

This work first proposes an encoder-decoder many-to-many bidirectional LSTM-RNN architecture for visual speech synthesis, which aims to generate lip movements from a text input. This encoder-decoder architecture is suited to the context-truncated BPTT because of the mismatched input and output time steps that originate from appended left and right contextual frame into the truncated block.

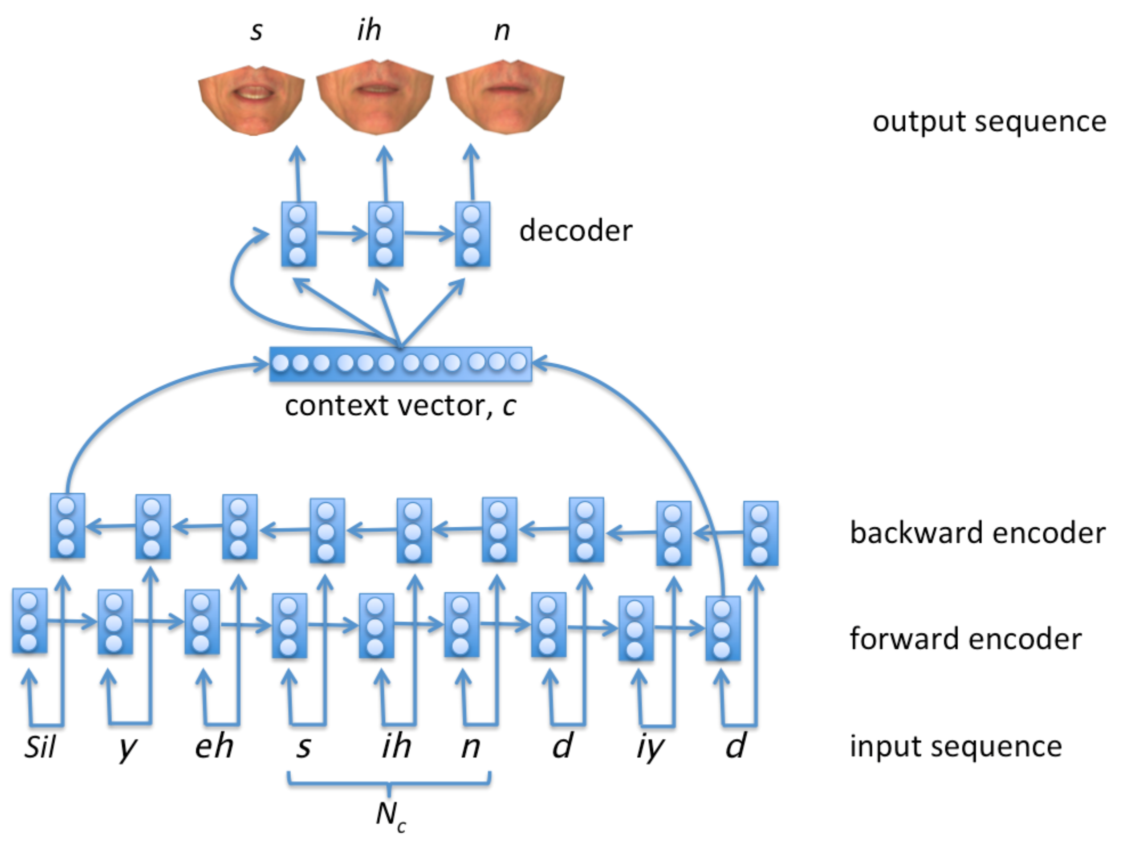


Figure 7.3: Our encoder-decoder many-to-many bidirectional LSTM-RNN architecture for visual speech synthesis.

Our network structure consists of the combination of two many-to-one architectures (encoder) and a one-to-many architecture (decoder), as illustrated in Figure 7.3. We aim to use one encoder to summarise the forward part of the input sequence, and another encoder to summarise the backward part of the input sequence. This is similar to the bidirectional technique that is described in Section 3.4.2. The main difference is this technique can access all time steps in both the forward and backward encoders, but the bidirectional technique in Section 3.4.2 is able to access the whole sequence in forward encoder and only one time step in the backward encoder. After getting the summarised context vector, **c**, for the cascade of the forward and backward encoder, we then use the LSTM-RNN many-to-one to generate the output sequence. Our work uses the basic encoder to provide the context vector as an initial hidden state and input for all time steps of the decoder part. The number of output time steps depends on the number of truncated blocks, N_c, which is shown as three-time steps in the example of Figure 7.3.

7.3 Experimental results

This section aims to evaluate the overall quality of the synthesised visual speech considering both objective tests and subjective tests. We first investigate the effectiveness of various configurations of LSTM-RNN system, for example the effect of the phone unit and dynamic viseme units. Then, we also investigate the best model for visual speech synthesis by comparing with a baseline HMM-based system and a DNN-based system. Finally, we also analyse where does the performance come from and what LSTM actually learn.

7.3.1 Experimental measurement

This section describes how the quality of a visual speech synthesiser is evaluated. Generally, the evaluation of visual speech synthesis can be classified into two types; objective tests and subjective tests. Both tests will be considered in this work.

7.3.1.1 Objective tests

The goal of objective tests for a visual speech synthesiser is to measure the distance between the synthesised and the ground-truth AAM visual features. Compared to subjective tests, objective testing methods are less time consuming and easy to perform because these can be computed automatically and repeatedly. Note that objective tests only indicate how far synthesised visual speech is from the real visual speech. They cannot indicate the realism, naturalness or intelligibility of a lip animation. Most visual speech synthesisers uses root mean squared error (RMSE) or correlation as an objective score. This work will use another additional objective measure, namely the global variance (GV). More specific details can be found in Section 5.3.1.

7.3.1.2 Subjective tests

Subjective tests measure a viewer's opinion of quality. Subjective tests can be divided into three types: intelligibility, naturalness and realism of the lip animations [24]. The task of the naturalness and realism tests are generally used for subjective evaluation. These tests ask viewers to compare the quality of synthesised lip animations in each system and focus on the how much the synthesised video is similar to the natural video. On the other hand, the task of the intelligibility test uses humans to lip-read the synthesised video and many studies reported that the performance of this approach has not been effective. So that, the tests in this work focus on two common approaches in visual speech synthesis testing; namely, (i) Turing test (realism) and (ii) A/B test (naturalness). With Turing test, this will ask viewers to watch video and choose whether they consider the lip motion real or synthesised. With A/B test, this will ask viewers to watch pairs of lip animations played side-by-side and to select the sequence that they found most natural.

7.3.2 Experimental training conditions

Three types of neural networks including one-to-one (feedforward), many-to-one using LSTM, and our proposed encoder-decoder many-to-many approaches are used in this experiment. For the feedforward network, the model contains three hidden layers and 3000 units per layer, which came from the best configuration from chapter 6. A rectified linear unit (ReLU) activation function is used for hidden layers and a linear activation function is employed at the output layer. Mini-batch stochastic gradient descent is used as the optimizer in the feedforward network since it achieves the best performance compared to other optimization methods. The size of mini-batch is set to 128; a learning rate and momentum were fixed to 0.3 and 0.9, respectively. The maximum number of epochs was set to 140. 55% dropout is also applied to avoid overfitting on both hidden and recurrent units.

For the many-to-one architecture using long short term memory (LSTM), the model contains three hidden layers and 256 units per layer. A linear activation function is employed at the output layer. RMSprop [22], which generates its parameter updates using a momentum on the rescaled gradient, is used as the optimizer in our experiment since it achieves the best performance compared to other optimization methods. We utilized early stopping with patience 15 to avoid over-fitting. The size of mini-batch is set to 128; a learning rate and momentum were fixed to 0.0003 and 0.9, respectively. The maximum number of epochs was set to 140. 55% dropout is also applied to avoid overfitting on hidden units.

For our encoder-decoder many-to-many architecture using LSTM, the model contains three hidden layers in both the encoder and decoder parts, and 256 units per layer. The other settings are set to the same configurations as the many-to-one approach.

7.3.3 Analysis of framing method

This experiment aims to explore the appropriate framing method of the input features for the RNN structure. In the feedforward network, a fixed stack of input features of the prior and next time steps is used as the contextual input features in each time step, namely the window method. The key is that there is no connection between each time step for the feedforward network structure. In the RNN structure, on the other hand, there is a connection between current time step and prior time step. The context information from the neighbouring frames would be excluded from the current frame, but would be provided in the form of recurrent structure instead, called time step method.

Three types of RNN architectures using two different framing methods were conducted in this experiment; (i) one-to-one, (ii) one-to-many, and (iii) encoder-decoder many-to-many, as shown in Table 7.2. The training conditions can be found in section 7.3.2. There is no result for one-to-one architecture using the time step method because of structure's limitation. To make a fair comparison, both window and time step methods must be trained with the same features. In the case of window method, the input features in each frame are append from the neighbour frames. Note that, this window method suits for one-to-one architecture because input representation of the current frame in this architecture can not access the information from the neighbour frames, as shown in Figure 3.17(a). In the case of timestep method, however, the input features in each frame are represented by the features of the current frame. Moreover, this timestep method suits for recurrent structure (for example many-to-one and many-to-many) because this structure is able to access the previous information from the recurrent step, as shown in Figure 3.17(b). Note that the width of the contextual information for both methods has to be determined in advanced and needs to be wide enough to include articulation movements but short enough to avoid over-smoothing of features. Related studies have reported a window width of 340ms which is a satisfactory value [99]. This time corresponds to an 11-frame window when applied to 30 frame-per-second (fps) data, and comprises 5 preceding, 5 ahead, and the current frame.

Table 7.2: Comparison between window method and timestep method.

		Correlation	RMSE	GV
One-to-one: Feedforward	Window	0.87(±0.05)	6.54(±1.34)	987.29(±202.74)
	Timestep	-	-	-
Many-to-one: LSTM	Window	0.86(±0.04)	6.42(±1.19)	1203.18(±241.03)
	Timestep	0.86(±0.05)	6.32(±1.24)	1262.35(±245.07)
Many-to-many: LSTM	Window	0.86(±0.04)	6.45(±1.21)	1463.36(±279.33)
	Timestep	0.87(±0.05)	6.23(±1.21)	1493.52(±284.12)

From Table 7.2, it would suggest that the use of time step method outperforms the use of window method in both many-to-one and many-to-many architecture using LSTM-RNN. It can be seen that a large lift in performance originates from the improvement of GV and RMSE. Hence, the following experiments will use time step method to represent the input features vectors through out this chapter. An important reason of making the better NN model when using time step method is that the structure of input data makes a lot more sense which represents one features vector based on their time step. This could make the network easy to desire about keeping, deleting, and outputting the useful information in each time step.

7.3.4 Analysis of speech units

So far, the combination of phoneme and dynamic viseme units is the most effective unit for visual speech synthesis using feedforward network. This experiment aims to ensure that the combination of phoneme and dynamic viseme units is still the most appropriate units for visual speech synthesis using an encoder-decoder LSTM-RNNs architecture. An investigation is now made into the effect of using either phonetic units or dynamic viseme units. For the phoneme based system all PH features shown in Table 7.1 are included while for the dynamic viseme system all DV features are included. A third configuration, PhDV, was also explored which combines the phonetic and dynamic viseme unit features and includes all features shown in Table 7.1.

Table 7.3: Analysis of speech units.

System	Correlation	RMSE	GV
Phone-only (Ph LSTM)	0.82(±0.07)	7.56(±1.89)	1243.10(±247.11)
Dynamic viseme-only (DV LSTM)	0.83(±0.05)	6.93(±1.24)	1386.63(±293.73)
Phoneme-DV (PhDV LSTM)	0.87(±0.05)	6.23(±1.21)	1493.52(±284.12)

Table 7.3 shows the mean (±standard deviation) of the correlation, RMSE, and GV for the phoneme, DV and combined phomeme-DV systems using the encoder-decoder many-to-many LSTM approach. We refer to the supplementary video for animation results. First of all, Table 7.3 shows that the use of dynamic viseme units to the models outperforms phone models in all metrics. It can be concluded that the well defined visual units, formed by dynamic viseme, is the key point of improving the resultant. Furthermore, the PhDV system in the last row of Table 7.3 shows the big improvement in terms of better correlated, low rmse, and large GV comparing with Ph- or Dv-only system. This is very interesting result because it is confirmed that both information is useful for generating people's lip movement. The advantage of combining phoneme and DV is likely to getting acoustic information from phone units and providing extra information about visual from dynamic viseme units in the same model. That is the reason to use both information representing as the unit of visual speech synthesis instead of forcing to choose only one unit as we have done

in HMM system.

As mentioned earlier, one of the limitations of using phoneme units is that the visual movements sometimes happen before or after the sound. This leads to the perceived quality because humans are very sensitive to the coherence between audio and visual signals. Figure 7.4 shows example time-varying of the first AAM parameter sequences generated using the phone-based (blue) and DV-based (red) compared with the ground-truth (black) equivalent measured directly from the video. Frames 38-50 correspond to the phrase “the outer door”. It can be seen that phone trajectory and dynamic-viseme trajectory have the similar shape but different timing. This leads to an audiovisual lack of synchrony. It can be confirmed by the inspection of the side-by-side video outputs, which shows that the lip shape from phone units starts and ends before it should.

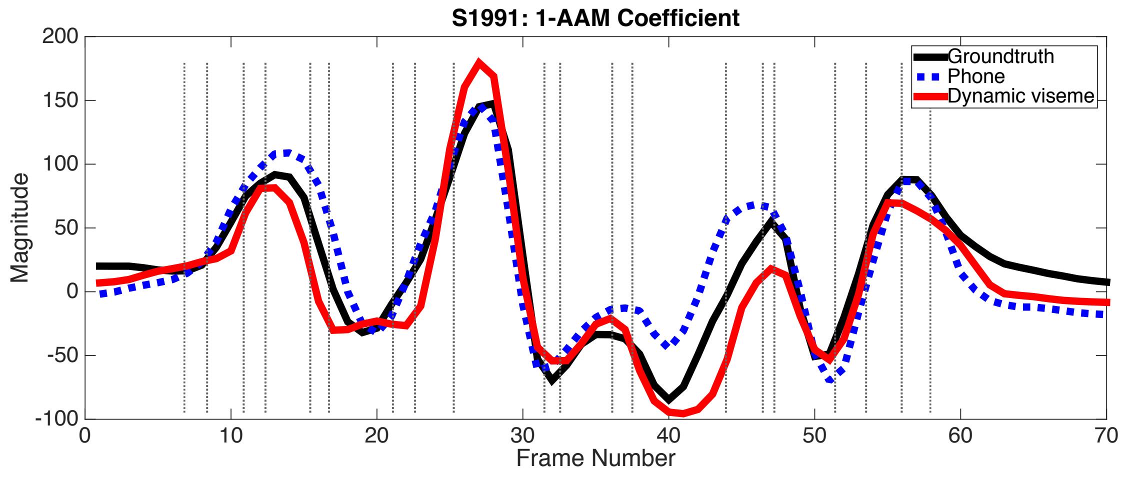


Figure 7.4: The time varying trajectory of the first AAM parameter as measured from video (black), synthesised using dynamic visemes (red), and phones (blue).

Even though dynamic viseme units are able to reduce the audiovisual lack of synchrony, they sometimes introduce over-smoothing. From this problem, the visual video output sometimes produces a too smooth or wrong lip motion that can appear to under or over articulate. To alleviate this problem, combining phoneme and DV features further improves performance which we attribute to their complementary information, one relating to acoustics and the other to visual information. To illustrate this, Figure 7.5 shows the first AAM parameter for ground-truth, dynamic-viseme, and combination of phone and dynamic-viseme sequence. This corresponds to the phrase “then he heard the outer door closing”.

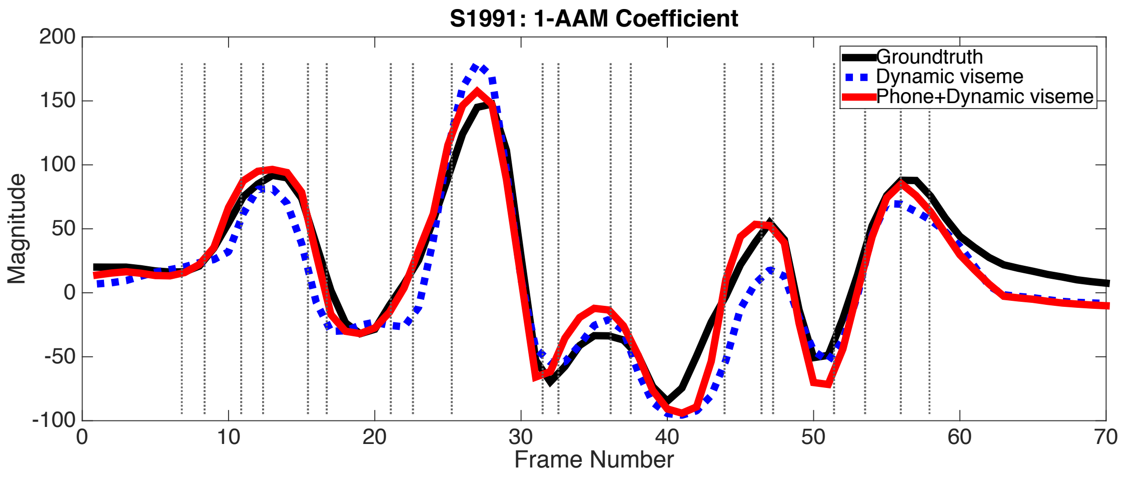


Figure 7.5: The time varying trajectory of the first AAM parameter as measured from video (black), synthesised using combination of dynamic visemes and phones (red), and dynamic visemes (blue).

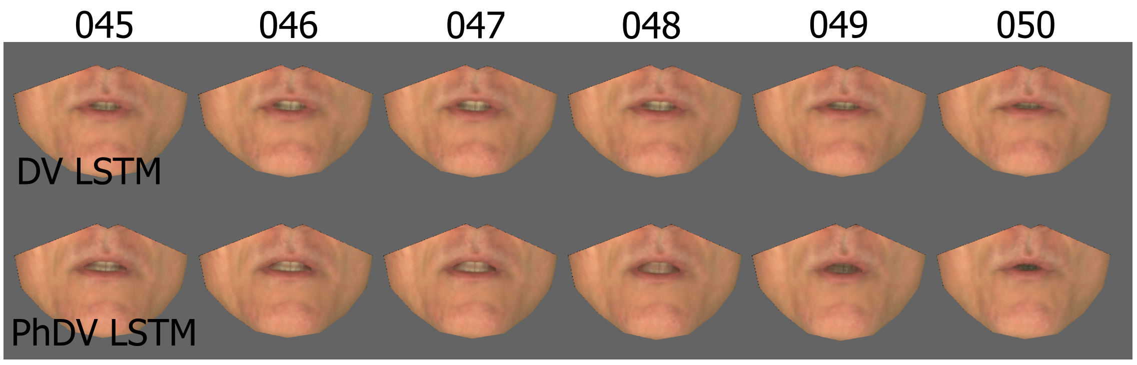


Figure 7.6: Selected frames 45-50 from the sequence (S1991) correspond to the word “door”. Each row shows an equivalent lip shape video that reconstructed from AAM parameters. Row 1 and 2 correspond to the LSTM-based visual speech synthesis using dynamic viseme units (DV LSTM) and the combination of phone and dynamic viseme units (PhDV LSTM), respectively. The lip shape sequences are correct in both system, but is somewhat under articulated for the use of dynamic viseme units.

Frames 45-50 from Figure 7.5 correspond to the word “door”. Analysis showed that the shape of dynamic-viseme output (blue) is similar to the shape of combination of phone and dynamic-viseme output (red) with somewhat different magnitude. From this contour, the inspection of the visual difference between DV-based (top) and PhDV-based (bottom) synthesiser can be found in Figure 7.6. The reconstructed images from the similar contours but with different magnitudes showed similar lip shapes. However, the major difference is that the generated mouth shapes from dynamic-viseme (top) are not flexible as the mouth does not open much as it should, resulting in the teeth not being clearly seen, which called under articulation problem.

7.3.5 Effect of the truncated context length (N_l and N_r)

This test investigates the effect of the length of the left, N_r and right context-truncated block, N_r. According to this, we assume that contextual information is very important for visual speech synthesis because it is well-known that the movement of the articulators does not depend only on the current sound but also on the neighbouring sounds. Hence, all speech and visual speech applications, e.g. speech synthesis and visual speech synthesis, consider their adjacent context when predicting the current sound.

A preliminary study in Table 7.4 was carried out to examine how contextual information affects the performance of visual speech synthesis using the LSTM encoder-decoder. Each configuration in Table 7.4 represents a truncated of length N_c with N_l left context and N_r context frames as "$N_l - N_c + N_r$". When excluding contextual information, N_l and N_r are set to 0. The meaning of $0 - 3 + 0$ is that this block comprises three input and three output frames without left or right context frames. As expected, the objective scores when including contexts, $6 - 1 + 6$, is better than the others that exclude context configurations. These imply that the output lip shape in each frame is influenced by the input information before and after it. The next experiment explores a suitable number of left, N_l, and right contextual frames, N_r, of the context-truncated block.

Table 7.4: The mean (±standard deviation) scores averaged for different truncation lengths with no context information in BLSTM encoder-decoder many-to-many.

$N_l - N_c + N_r$	Correlation	RMSE	GV
$(0-1+0)$	0.86(±0.05)	6.61(±1.37)	1411.66(±254.00)
$(0-3+0)$	0.86(±0.05)	6.51(±1.37)	1420.80(±258.18)
$(0-7+0)$	0.87(±0.05)	6.29(±1.26)	1375.71(±250.64)
$(0-15+0)$	0.87(±0.04)	6.29(±1.23)	1383.79(±259.10)
$(0-23+0)$	0.87(±0.04)	6.20(±1.24)	1389.16(±272.07)
(6-1+6)	**0.88(±0.04)**	**6.18(±1.21)**	**1482.47(±285.58)**

In Table 7.5, we assume that the truncated block, N_c, is fixed to 1, and the size of the contextual frames was varied from 0 to 11. Table 7.5 shows the effect of contextual configurations in BLSTM many-to-many and encoder-decoder BLSTM. An example of (5-1+5) configuration can be interpreted that we aim to predict one time step output after looking the previous five time steps and the next five time steps. It is seen that the small number of appending contexts from 1 to 4 cannot capture the salient coarticulation effects and causes an unacceptable degradation to the synthesised visual speech. This implies that this information is not enough. Also, it is observed in the correlation results that it cannot distinguish the performance in each configuration, as reported the same amount of errors. The best GV and RMSE error is obtained by "$7 - 1 + 7$" and "$11 - 1 + 11$" in terms of averaged results, respectively. From these results, the window duration is about 495-695ms, and is longer than [61, 99] (340ms) and [39] (183ms). This is one of the benefits of the LSTM-RNN structure in terms of the ability of capturing such long-term dependencies.

Table 7.5: The mean (±standard deviation) scores averaged for different left and right context length with a single truncation length in BLSTM encoder-decoder many-to-many.

$N_l - N_c + N_r$	Correlation	RMSE	GV
33ms $(0-1+0)$	0.86(±0.05)	6.61(±1.37)	1411.66(±254.00)
100ms $(1-1+1)$	0.86(±0.05)	6.50(±1.36)	1438.84(±272.17)
165ms $(2-1+2)$	0.87(±0.05)	6.41(±1.28)	1478.10(±264.68)
230ms $(3-1+3)$	0.87(±0.05)	6.32(±1.31)	1451.99(±267.69)
300ms $(4-1+4)$	0.87(±0.05)	6.30(±1.31)	1489.16(±278.96)
360ms $(5-1+5)$	0.87(±0.04)	6.19(±1.24)	1483.60(±286.63)
430ms $(6-1+6)$	0.88(±0.04)	6.18(±1.21)	1482.47(±285.58)
495ms $(7-1+7)$	0.88(±0.04)	6.15(±1.22)	1503.54(±293.04)
560ms $(8-1+8)$	0.88(±0.04)	6.13(±1.20)	1494.30(±286.20)
630ms $(9-1+9)$	0.88(±0.04)	6.15(±1.18)	1492.28(±278.62)
695ms $(10-1+10)$	0.88(±0.04)	6.09(±1.15)	1483.14(±288.42)
760ms $(11-1+11)$	0.88(±0.04)	6.08(±1.15)	1491.07(±290.55)

In fact, it is difficult to claim that the “$7-1+7$” configuration is better than the “$11-1+11$” or other configurations. Analysis in Figure 7.7 shows that the observed trajectory of “$7-1+7$” (blue) is likely to be close to the ground-truth trajectory (black) and the trajectory of “$11-1+11$” (red) of the word ”purists” at frames 32-37 has a different shape when comparing with the ground-truth. From this difference, it interesting to note that the overall lip shapes are still the same, as shown the reconstructed lip images in Figure 7.8. They tend to open the mouth in the same degree as it should be. The difference in lip shape can be seen at frame 35, where the teeth are not visible when using 11-1+11 setting (top). This shows that the small errors in a single frame from this settings lead to the similar output video sequence. However, we would choose “7-1+7” as the best setting and using 7 as the number of left and right context length in the next experiment, because we found that the higher GV tends to create the better animation (as described in Section 5.3.1).

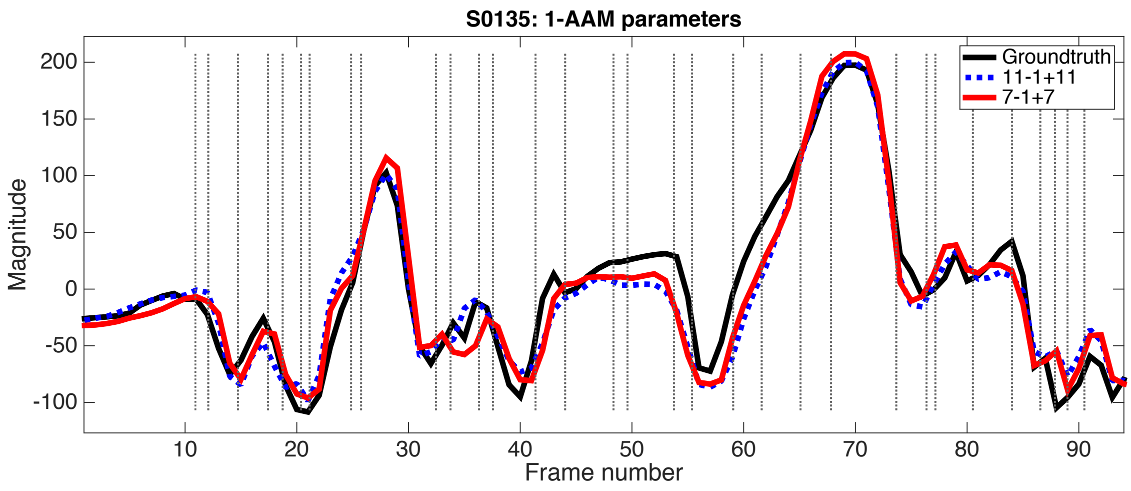


Figure 7.7: The time varying trajectory of the first AAM parameter as measured from video (black), synthesised using "7-1+7" (red), and "11-1+11" context-truncated block setting (blue).

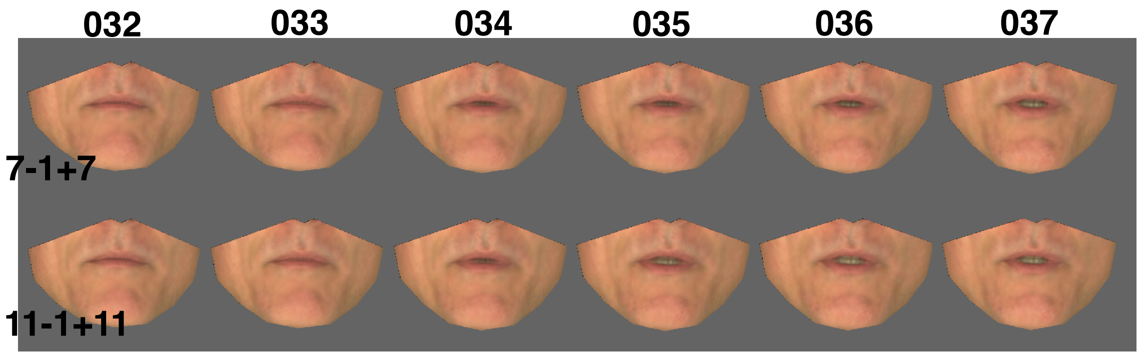


Figure 7.8: Frames 32-37 from a "$7-1+7$" configuration (top) and a "$11-1+11$" configuration (bottom), corresponding to the first syllable of the word "purists". Every frame shows the similar lip shape in both systems.

7.3.6 Effect of the truncated context decoding (N_c)

Based on the previous experiment, we know that "7" is the best number of left, N_l, and right, N_r, contextual frame for a truncated block. In this experiment, we examine the length of a truncated block, N_c, that is varied from 1 to 9, as shown in Table 7.6. It observed that there is no difference in terms of correlation results as we found in previous experiment. However, the RMSE error and GV are able to reflect difference in the performance of each configuration. From these results, the wider truncation block (i.e. larger N_c) tends to generate the lower RMSE and GV. Generally, we expect to get a smaller RMSE, but the RMSE scores in Table 7.6 does not correspond to better animation, because this always leads to under articulated problem.

Table 7.6: Analysis of the length of truncated with 256 length of context vector, **c**.

$N_l - N_c + N_r$	Correlation	RMSE	GV
$7-1+7$	0.88(±0.04)	6.15(±1.22)	1503.54(±293.04)
$7-3+7$	0.88(±0.05)	6.07(±1.25)	1424.44(±266.16)
$7-5+7$	0.88(±0.05)	6.06(±1.23)	1410.25(±266.51)
$7-7+7$	0.88(±0.05)	6.04(±1.23)	1408.24(±272.96)
$7-9+7$	0.88(±0.04)	6.01(±1.19)	1409.22(±270.52)

As we mentioned in Section 7.2.3.2, the limitation of the basic encoder-decoder architecture is the size of context vector, **c**, as this architecture aim to encode all input information to a fixed-sized vector, then generate or decode the output based on that vector. We can conclude that the length of our context vector is too small for the larger context outputs. Therefore, the size of context vector is increased from 256 to 1024. As expected, Table 7.7 shows the improvement in terms of both RMSE and GV. We found that we should not choose N_c larger than 3 because the larger N_c causes smoother output from the averaged. Again, it is difficult to choose the best configuration between "7-1+7" and "7-3+7" because their output is very close. However,inspection of side-by-side videos and visual trajectories we found that "7-3+7" is a better configuration because the output is smoother than "7-1+7" with no under articulation problem, as shown in Figure 7.9.

Table 7.7: Analysis of the length of truncated with 1024 length of context vector, **c**.

$N_l - N_c + N_r$	Correlation	RMSE	GV
$7-1+7$	0.88(±0.04)	6.13(±1.22)	1496.54(±280.77)
$7-3+7$	0.88(±0.04)	6.02(±1.24)	1457.32(±272.54)
$7-5+7$	0.88(±0.04)	5.99(±1.23)	1420.47(±266.46)
$7-7+7$	0.88(±0.04)	6.06(±1.24)	1422.13(±271.64)
$7-9+7$	0.88(±0.04)	5.98(±1.22)	1443.91(±267.73)

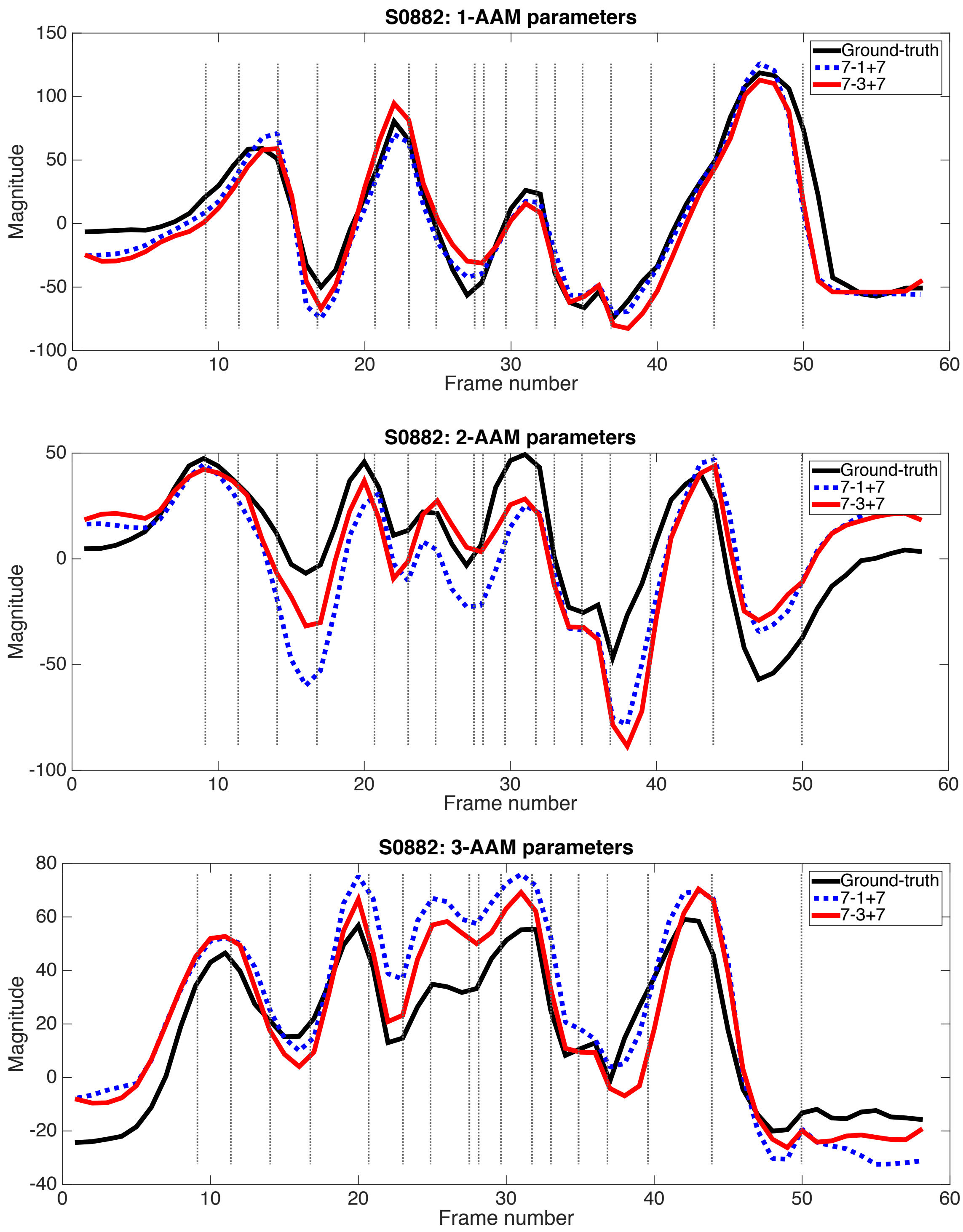


Figure 7.9: The time varying trajectory of the first-third AAM parameters as measured from video (black), synthesised using the two-best configurations of our NN model including "7-1+7" (blue) and "7-3+7" (red). The predicted AAM parameter of the first dimension in both configurations is nearly the same as ground-truth parameter. The shape of second and third parameters is still similar, but somewhat different magnitude.

7.3.7 Analysis of the RNN architecture on visual speech synthesis

This experiment summarises how the overall performance of the synthesised visual speech is improved by our LSTM-based approach, which uses an encoder-decoder many-to-many architecture (described in Section 7.2.3.2). There are two common problems that are concerned with our visual speech synthesis system; visual speech modeling and over smoothed articulation. With the visual model, this problem originated from the misshaped and mistimed estimation of the visual speech parameter trajectories. For under articulation, this problem exits when the model predicts the correct shape but the magnitude is too small. Hence, the ultimate goal of our proposed LSTM is to learn the dynamic visual model and synthesise a flexible output.

For comparison, two systems including a baseline phoneme HMM synthesis system (HMM; described in Chapter 5) and a feedforward neural network synthesis system (DNN; described in Chapter 6) were compared with the proposed LSTM synthesis system (LSTM). In our LSTM system, the left and right context, N_l and N_r, were set to 7 and the truncation block, N_c, was set to 3, due to its best performance in previous experiments. The training configurations for the HMM, DNN and LSTM system are described in Section 5.2.1.3, 6.2.3, and 7.3.2, respectively.

The averaged results for all three systems with different numbers of training utterances, which were varied as; 100, 300, 500, 800, 1000, 1200, 1500, 1800, 2000 and 2042, are presented in Figure 7.10. Note that, we evaluate all appraoches on the KB-2k 250 held out test utterances. We first consider the effect of the amount of training data for each approach. It can be seen that the performance is increased as the number of training utterances is increased all three approaches. These results also confirm that the number of training utterances is very important and more training data leads to a better predictor. Moreover, acceptable results are originated from the training data with at least 1000-1200 utterances. This amount of data relates to the same number of speech synthesis data such as the 1200 phonetically balanced utterances in CMU-ARCTIC. We refer to the supplementary video for animation results.

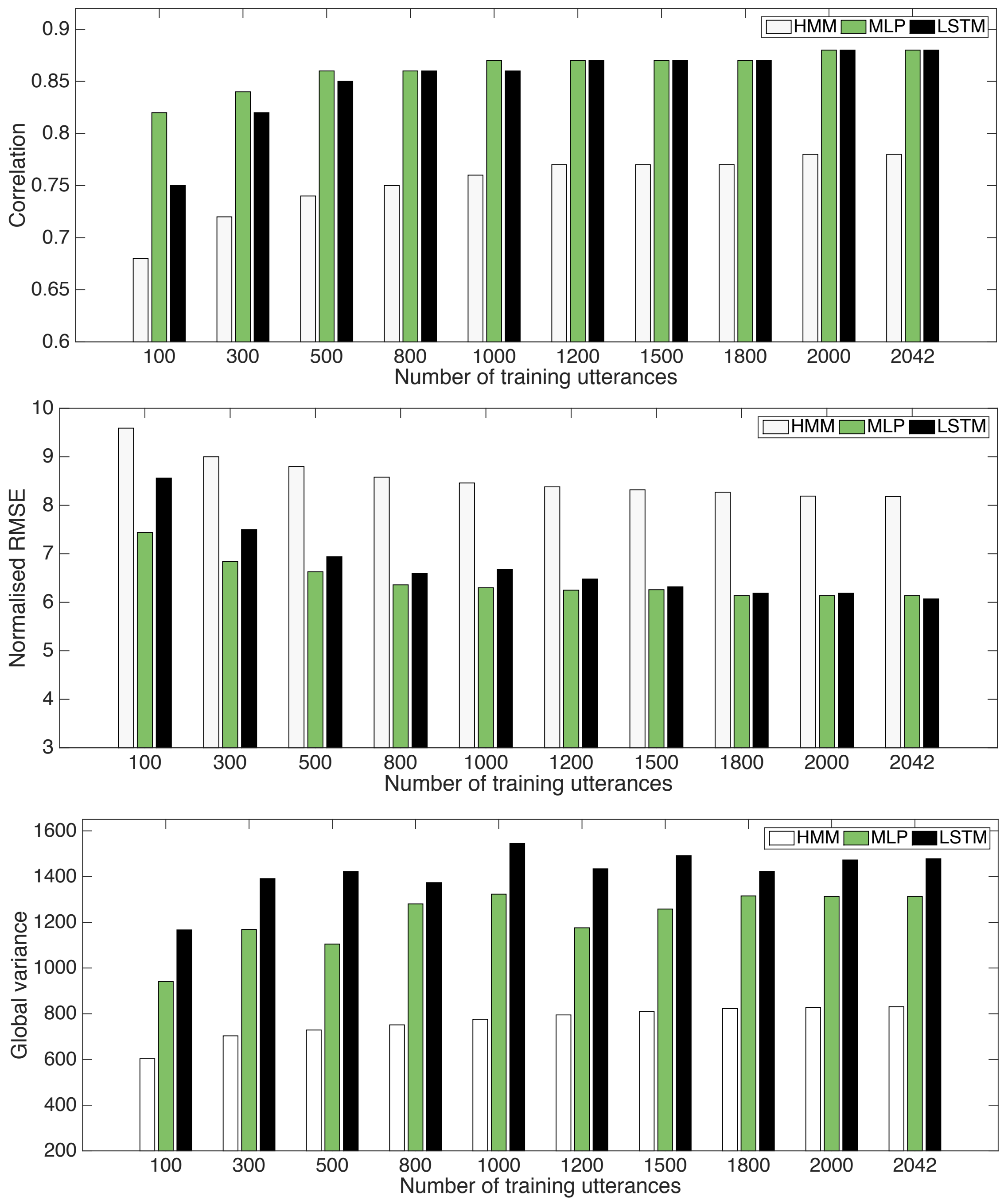


Figure 7.10: The objective scores averaged for three different systems: HMM, DNN and LSTM.

Clearly, the deep neural network approach in both DNN and LSTM approach consistently outperforms the state of the art HMM system in terms of correlation, RMSE, and GV. Note that using only correlation and normalised-RMSE measurements achieve almost the same performance on DNN and LSTM system. However, the GV measure gives an indication of which approach produces the best predictor in terms of articulation. Findings suggest our proposed LSTM system is the most preferred overall as shown the larger gap on GV scores. From these results there are several reasons that the performance of our proposed encoder-decoder using LSTM is better than the baseline HMM-based and DNN-based synthesis system. Firstly, the framing time step method has the benefit of using one set of features in its time step that makes learning the method easier. Compared with the HMM and DNN, both systems incorporate contextual frames in one set of features, which possibly makes it difficult to model the input features. Specific detail were discussed in Section 7.3.3. Secondly, the combination of the audio unit (phones) and visual unit (dynamic visemes) is the key of improving the synchronisation of audio and visual. The investigation in Section 7.3.4 showed that the synthesised lip motion from this combination can overcome the under articulation problem. Thirdly, the ability of learning long range dependencies from the LSTM structure is one of the reasons for the improved synthesis. Specific details are given in Section 7.3.5. Fourthly, another important problem for visual speech synthesis is under articulation of synthesised lip animation. For this problem, the model is able to predict the correct lip movements, but they lack sufficient articulation. Error from correlation and RMSE objective scores were not able to reflect this problem, which become apparent with subjective tests. In this work, we believe that the GV object metric is able to reflect the articulation of animation. Figure 7.11 illustrates the articulation of lip animation from LSTM-based compared with original, HMM-based, and DNN-based. We refer to the supplementary video for animation results.

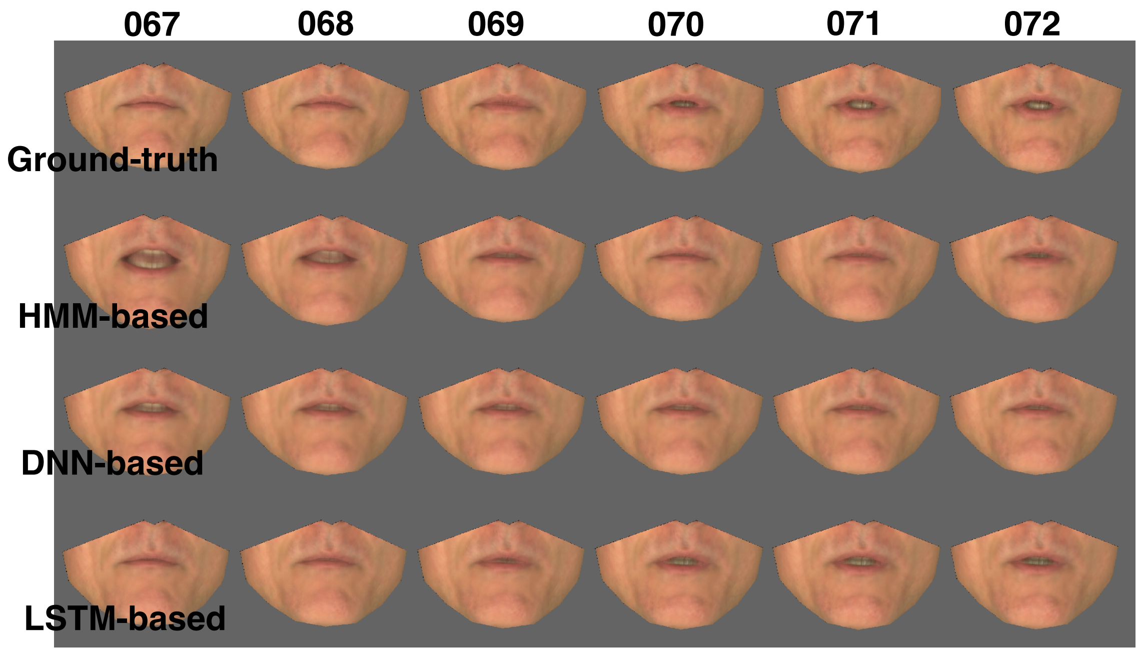


Figure 7.11: Selected frames 67-72 from the sequence "those who are not purists use canned vegetables when making stew". The frames correspond to the word "canned" and each row shows an equivalent lip shape video that reconstructed from AAM parameters. Row 1 correspond for ground-truth AAM parameters. Row 2,3, and 4 correspond to the HMM-based, DNN-based, and LSTM-based visual speech synthesis using the combination of phone and dynamic viseme units, respectively.

7.3.8 Subjective tests

All previous experiments evaluated the quantity of the visual speech synthesis system using objective tests. This section measures the quality of each system using subjective tests. These tests are conducted by humans who were asked to evaluate the quality of the lip animations. Compared to subjective tests, objective testing methods are less time consuming and can be done automatically, but they are not able to confirm the quality of the lip animation. They also have no guarantee that better objective scores will generate the better videos. For example, [103] reported that the high error scores from a smoothing method showed better animation compared with no-smoothing method. These experiments aim to evaluate the quality of lip animations in terms of realism and naturalness. Two subjective tests were conducted in a sequential sequence; a Turing test to measure realism followed by a preference test to compare naturalness. Regarding to the subjects who evaluate in this test, we have an assumption that this application should not only specific to US English speakers because this application should use by everyone who understand English language. From this concept, a total of 30 UK English speakers and non-native viewers who study abroad took part in the tests. There were recruited online and took the test over the Internet using a web browser (http://oaom.openservice.in.th).

To evaluate the statistical significance of the realism and preference tests, a binomial test at the 1% significance level, $\alpha = 0.01$, was conducted [49]. Note that, the binomial test is used in all subsequent subjective tests because there are only two possible answers in each test. For example, the outcome of Turing test corresponds to "real" or "synthesised" and the outcome of preference test corresponds to "system A" or "System B". With the binomial process, we first define null hypothesis (H_0) and alternative hypothesis (H_1) as follows:

$H_0 : p = \frac{1}{2}$ ("The judgements are not biased to either an A or B category")

$H_1 : p > \frac{1}{2}$ ("The judgements are more likely to favour an A category")

If, for example, a subject watches $N = 15$ animations and they prefer "A" 13 times ($C = 13$). It is likely that humans are more likely to favour "A". So in this case we then calculate the probability (p-value) that the number of humans choosing "A" 13 times or more than 13 times out of 15 judgements, as follows:

$$p - value = P(X = 13) + P(X = 14) + P(X = 15). \tag{7.2}$$

where,

$$P(X = C) = \left(\begin{array}{c} N \\ C \end{array} \right) \times p^C \times (1 - p)^{N-C} \tag{7.3}$$

$$\left(\begin{array}{c} N \\ C \end{array} \right) = \frac{N!}{C!(N - C)!} \quad where \quad N! = 1 \times 2 \times \ldots \times N \tag{7.4}$$

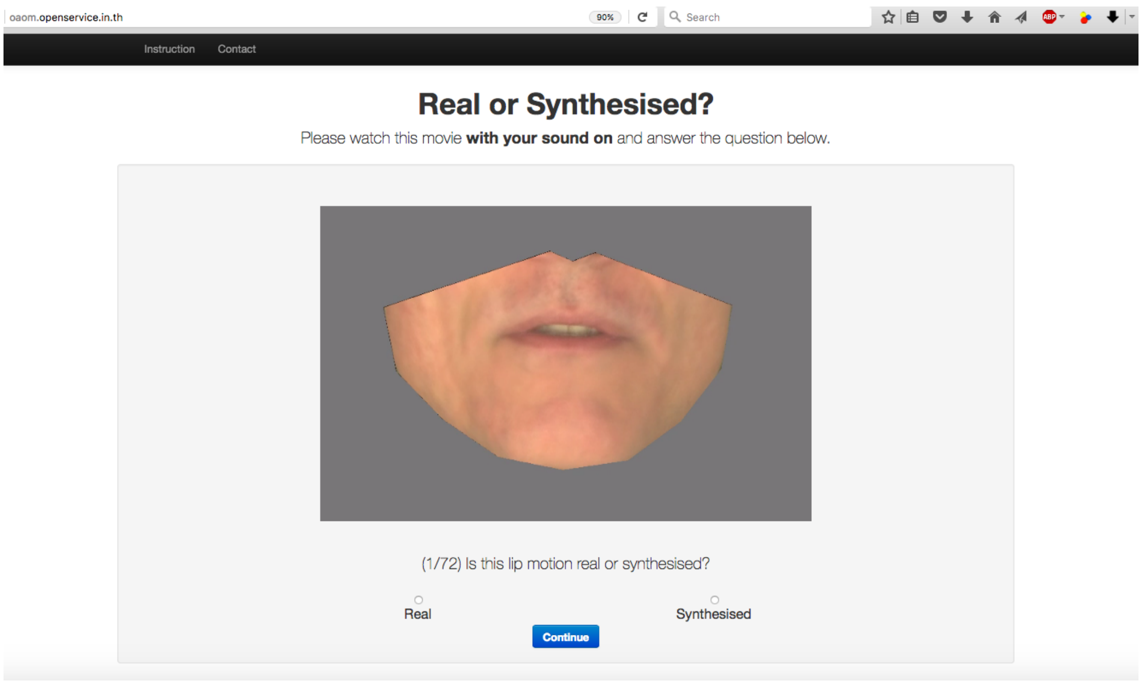
so that,

$$p - value = 0.0037 \tag{7.5}$$

With the calculation of the p-value=0.0037 with our significance level at 0.01, we reject the null hypothesis and we conclude that subjects prefer significantly the animation from the "A" category.

7.3.8.1 Turing tests

The Turing test or realism test aims to measure how realistic the synthesised lip animation is. In this test, each video was shown to viewers and then they were asked to rate with a simple question “Is this lip motion real or synthesised?”, as shown in Figure 7.12. Viewers were told to ignore image and audio quality and make a choice based on only the mouth movements and the audiovisual coherence. They could watch the videos as many times as they required before assigning their feedback that reflects the synchronisation between acoustic speech and lip movement and whether they think the lip motion is real or synthesised.



Real or Synthesised?

Please watch this movie **with your sound on** and answer the question below.

(1/72) Is this lip motion real or synthesised?

Figure 7.12: Visual speech synthesis experiment - Turing test.

Four systems were involved in this test; HMM_PH, DNN_PH_DV, LSTM_PH_DV and REAL. The first system was a baseline phoneme HMM synthesis system (HMM_PH), as described in Chapter 5. The second system was a feedforward system (DNN_PH_DV), as described in Chapter 6. The third system was a bidirectional-LSTM encoder-decoder system (LSTM_PH_DV), as described in Chapter 7. Finally, the real video (REAL) means the video was generated from the ground-truth AAM parameters.

The videos were randomly chosen from the 250 testing utterances that were not included in the training data. These videos are shown to viewers in a random order and a single-video at a time. Each participant evaluates 10 sentences from the four different systems, giving 40 videos in total. The first four answers from each participant were not counted for familiarisation reasons. From this omission, the remaining 36 answers from each participant were used to count the number of real and synthesised rated videos for each system.

We measure the statistical significance of the test result using a binomial test at the 1% significance level, $\alpha = 0.01$ (as described in Section 7.3.8). To test the hypothesis in each system we first define the null hypothesis (H_0) that humans' choice is not biased towards "Real" or "Synthesised" category ($p = \frac{1}{2}$). We then compute the probability and compare with our significance level to confirm it H_0 is true. Notice that H_0 is accepted if the probability is greater than the significance level.

Results of the realism tests, with the percentages were shown in brackets, are illustrated in Figure 7.13. It shows that subjects perceived that the lip motion videos from REAL is the best system in terms of realism at 68.66%, with a significance level of p-level $= 8.5e^{-10}$. More than 50% of the time, participants perceived the animations as real in DNN and LSTM neural networks structures. This suggests these two approaches are generally considered acceptable for by subjects. However, we find that only the lip motion videos from our LSTM_PH_DV system are significantly real with a p-level $= 9.8e^{-8}$ and show more realistic animation than from the HMM_PH_DV and DNN_PH_DV systems. It is interesting to note that the synthesised videos from the LSTM_PH_DV system achieve nearly the same performance as the ground-truth videos from the REAL system which was perceived as real 69% of the time as opposed to 66% of the LSTM_PH_DV system. These results mean that our LSTM_PH_DV system is the most competitive method and even humans in this test confuse the synthetic videos from this approach with the REAL system.

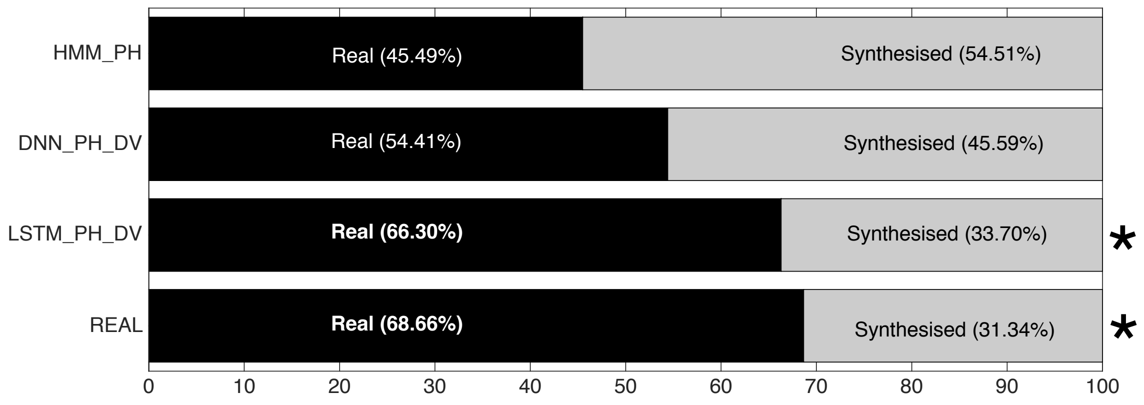


Figure 7.13: Subjective results for realism tests with four different systems: REAL, HMM_PH, DNN_PH_DV and LSTM_PH_DV.

7.3.8.2 Preference tests

The preference test or AB test aims to compare the naturalness of two systems at one time. Two system videos were played simultaneously as side-by-side videos. They were shown to viewers and asked a simple question "Tell us which on looks more natural". Subjects were free to replay the sequences as many times as required. Then, viewers were forced to choose one answer from a binary choice: "Left" or "Right", as shown in Figure 7.14.

In this test the animations were generated by AAM sequences from 4 different systems the same as in the Turing tests: REAL, HMM_PH, DNN_PH_DV, LSTM_PH_DV. From these four systems, 6 different combinations can be made:

1. HMM_PH vs. DNN_PH_DV
2. HMM_PH vs. LSTM_PH_DV
3. HMM_PH_DV vs. REAL
4. DNN_PH_DV vs. LSTM_PH_DV
5. DNN_PH_DV vs. REAL
6. LSTM_PH_DV vs. REAL

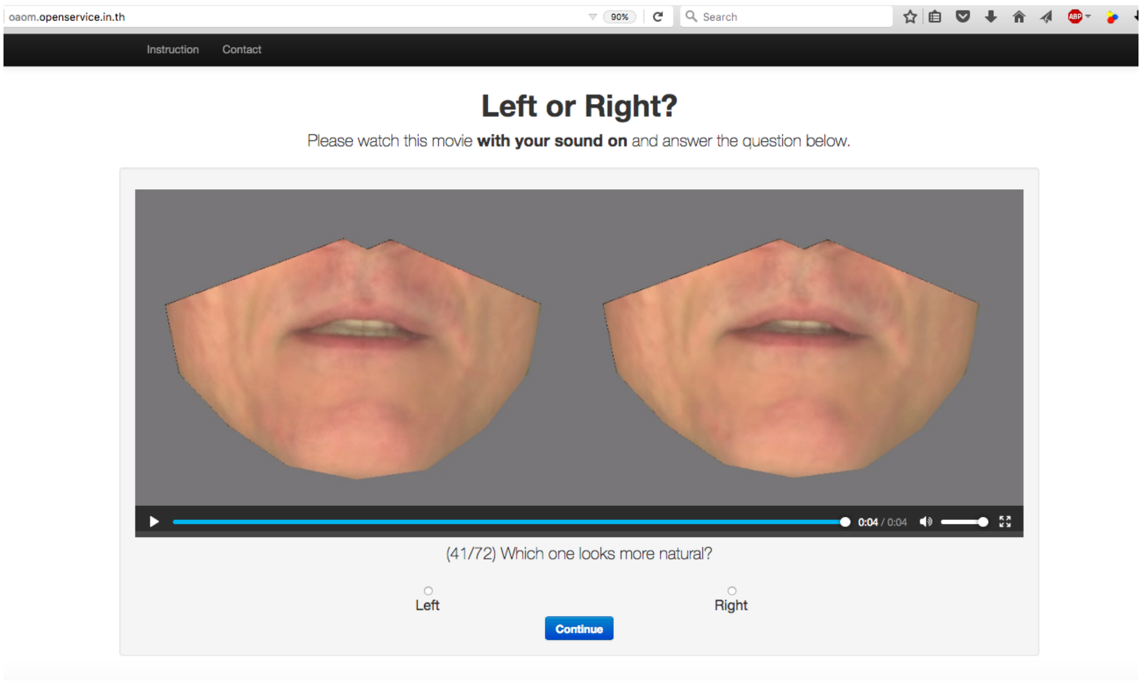

oaom.openservice.in.th
Instruction
Contact
Left or Right?
Please watch this movie **with your sound on** and answer the question below.
0:04 / 0:04
(41/72) Which one looks more natural?
Left
Right
Continue

Figure 7.14: Visual speech synthesis experiment - Preference test.

For the test, each combination uses four sentences that were randomly selected from the 250 sentence testing data. That means each viewer watches 24 side-by-side videos in total. The videos were played also simultaneously with the left-right order of the presentation randomized and the order that the sequences were played randomized.

Note that this test is shown to participants after they finished the Turing test, hence, the same 30 viewers take part in this forced-choice test. The preference scores of AB tests are shown in Table 7.8, along with the statistical significance using a binomial test at the 1% significance level, $\alpha = 0.01$. To test the hypothesis, we first assume that the videos from any paired-system are comparable, ($p = \frac{1}{2}$). Hence, a probability of the binomial test is used to find whether or not there is a significant difference between system A and system B in each paired-system. More details can be found in Section 7.3.8.

Table 7.8: Subjective preference scores (%) of the six combinations from four different systems: REAL, HMM_PH, DNN_PH_DV and LSTM_PH_DV Systems. Based on a binomial test (p-value) bold font is used to indicate whether differences are statistically significant.

System A	System B	Percentage	p-value
REAL	HMM_PH	**83.47/16.53**	2.03e-14
REAL	DNN_PH_DV	**75.63/24.37**	1.33e-8
REAL	LSTM_PH_DV	52.99/47.01	1.1297
HMM_PH	DNN_PH_DV	**34.71/65.29**	9.67e-4
HMM_PH	LSTM_PH_DV	**20.49/79.51**	2.20e-11
DNN_PH_DV	LSTM_PH_DV	**29.75/70.25**	8.15e-6

The percentage results of AB tests are shown in Table 7.8 with their corresponding p-values. Each p-value is used to determine if two systems are statistically different, where numbers with bold font indicate a significant difference between a pair of systems. Unsurprisingly, it can be seen that the REAL system is the best approach and significantly more natural than the DNN_PH_DV and HMM_PH but not significantly more natural than LSTM_PH_DV Systems. We found that there is no significant preference between REAL System and LSTM_PH_DV. This means that the animated lip movements from "LSTM_PH_DV System" is comparable to that of a ground-truth lip animation. Comparing the basic deep learning approach (DNN_PH_DV) with the basic Hidden Markov model approach (HMM_PH System), we see that 65% of participants found the DNN-based animations significantly more natural than HMM-based animations. These results also confirmed the findings in the realism tests that the LSTM_PH_DV significantly outperforms the phoneme HMM-based system and the combination of phoneme and dynamic viseme units on the DNN-based system. We refer to the supplementary video for animation results.

Several viewers reported that some of the animations (in the LSTM_PH_DV system) have realistic articulation in terms of open and closed mouth animation, nearly the same as REAL system does. That is a reason why they found that it was difficult to make a decision in some paired-video tests (the LSTM_PH_DV and REAL systems). They also pointed that in some of the paired-videos it was easy to spot the differences between them. That is because one of them (presumably the HMM_PH system) tended to lack audio-visual synchrony and to be under articulated in terms of mouth opening, as discussed in Section 7.3.7. Another feedback was that some of the videos were under-smoothed (presumably the DNN_PH_DV system) but they are still far from the flexible dynamic lip motion videos, as indicated global variance in this point in Section 7.3.3.

Chapter 8

Phoneme-to-dynamic viseme visual speech synthesis

8.1 Introduction

The work in the previous chapter demonstrated that a sequence to sequence method of visual speech synthesis using an encoder-decoder bidirectional LSTM-RNN is the best statistical parametric algorithm compared with HMM-based and DNN-based. Furthermore, we also found that using dynamic viseme information is better than phoneme and static viseme systems in all three statistical parametric algorithms including; HMM-based in Chapter 5, DNN-based in Chapter 6 and LSTM-based in Chapter 7. However, we have assumed that the dynamic viseme information in these three previous chapters was available because we aim to find the upper-bound of the system.

In a real setting there would be no ground-truth dynamic viseme transcription available, and consequently an input phoneme string would need to be mapped into a corresponding dynamic viseme input sequence. We initially used an existing phoneme-to-dynamic-viseme mapping method but found results to be unacceptable. We therefore developed a new phoneme-to-dynamic-viseme mapping method.

The rest of this chapter is structed as follows; firstly we overview and discuss why the traditional phoneme-to-dynamic viseme conversion method by Taylor et al. [98] does not produce reasonable results. We then propose a new dynamic viseme conversion in Section 8.3, followed by experiments that examine the number of dynamic viseme classes. Finally, we present the results of a subjective preference test and discuss the significance of this result.

8.2 Traditional phonemes to dynamic visemes conversion

A ground-truth dynamic viseme transcription was assumed in all previous experiments since the goal was to find the most suitable units for visual speech synthesis. It can be found that dynamic visemes units are the reasonable units as they are true visual unit of speech. However, in a real setting there would be no ground-truth dynamic viseme transcription available, and consequently an input phoneme string would need to be mapped to the corresponding dynamic viseme input. The following section reviews the existing method and our proposed method for phoneme-to-dynamic viseme mapping.

From the traditional approach by Taylor et al. [98], a sequence of dynamic visemes is mapped from a given input sequence of phonemes. To find this mapping, a dynamic viseme look-up table is built which contains a unique string of phonemes and their possible dynamic visemes candidates, for example /w-er/ $\in$ {V30, V12, V20, V66}, /w/ $\in$ {V23, V66} and /w-er-d/ $\in$ {V98, V20, V12}. This look-up table is used to generate all possible sequences of dynamic visemes from given phoneme sequences. For example, a given phoneme string /w-er-d/ for the word “word” can be generated all possible combination of phonemes, as illustrated in Figure 8.1, where black circled indicate dynamic viseme nodes.

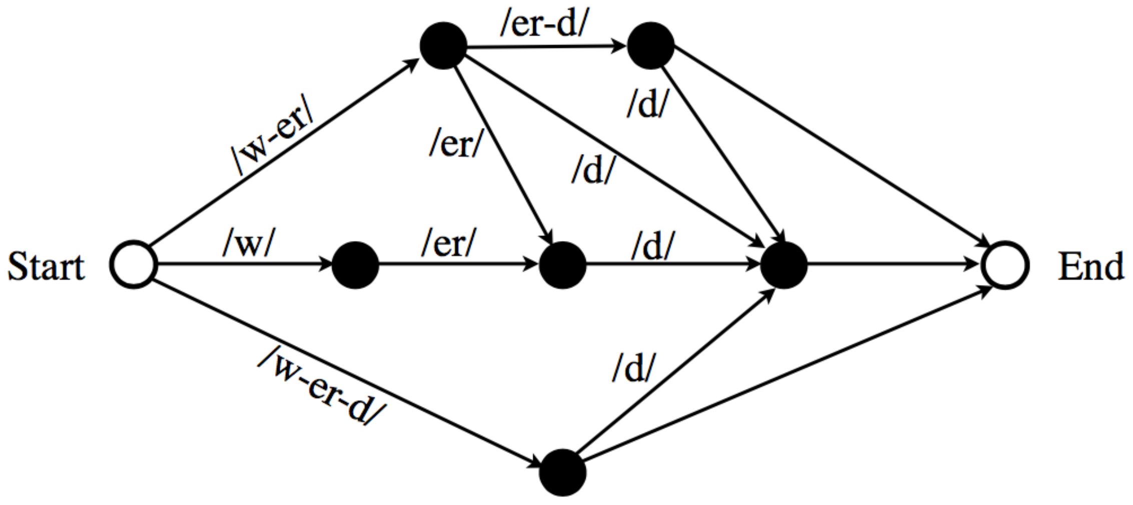


Figure 8.1: All possible combination of phonemes paths for a sequence of phonemes /w-er-d/ to dynamic visemes (black nodes).

To find the best dynamic viseme sequences, all possible combinations of dynamic viseme candidates that could have been generated from the phoneme string need to be considered. A cost function, c_i, is used to compute the quality of each dynamic viseme mapping, as follows;

$$c_i = \alpha(-Pr(V_i|PH)) + \beta(t_s(V_i, PH)) + \gamma(d(V_i)), \tag{8.1}$$

where V_i denotes the i^{th} candidate viseme. The first term in Equation (8.1) represents the probability of viseme, V_i, given the phoneme string, PH. The second term, t_s, represents the closest relationship between V_i and PH in terms of durations. The final term, γ, is used to control the smoothness at the boundaries in AAM space. α, β, γ are used to control the weights in each term. The total costs of each part through the trellis are the calculated and that with lowest cost is selected as the dynanmic viseme sequence. This is shown a red path in Figure 8.2. Regarding dynamic viseme time alignment, phoneme durations are used to assign a time to each viseme in the sequence. This consists of two cases; i) when the viseme and phoneme boundaries are exactly the same place, and ii) when the viseme boundary is half of its phoneme boundary. Figure 8.2 presents an example of dynamic viseme segmentation.

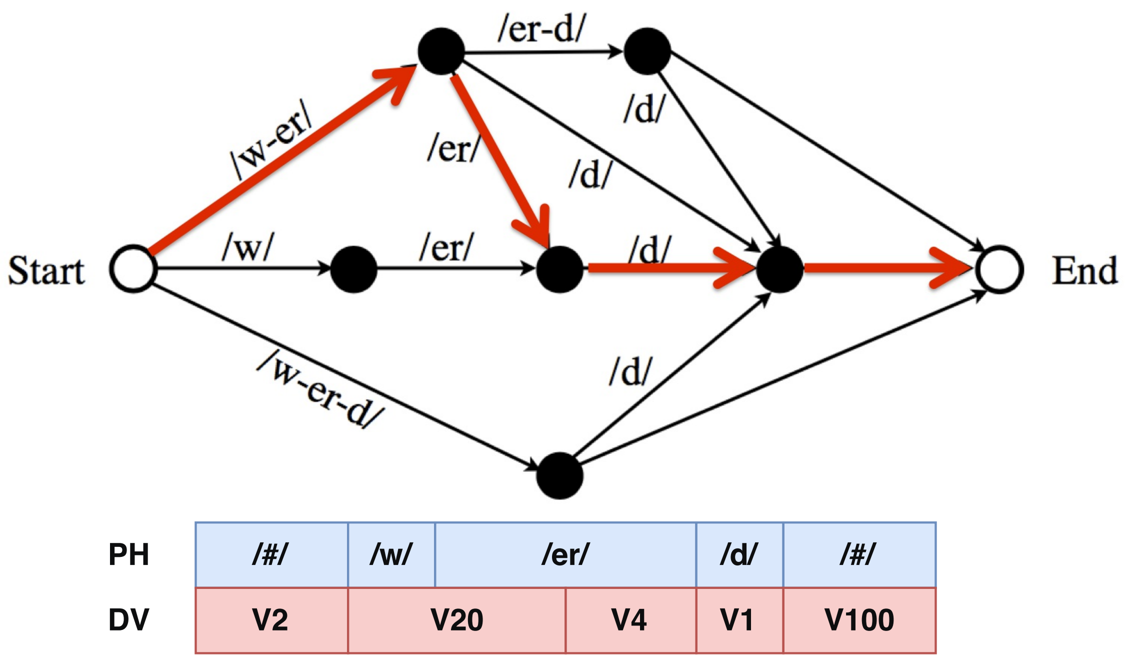


Figure 8.2: All possible combination of phonemes paths for a sequence of phonemes /w-er-d/ to visemes (black nodes).

From this mapping technique, there are two major drawbacks. Firstly, it is difficult to find the right sequence of dynamic visemes, as there is a many-to-many phoneme-to-dynamic viseme mapping. In particular, the same sequence of phonemes can map to different sequences of dynamic visemes. Secondly, dynamic viseme boundaries depend on phoneme boundaries. This is not true and would lead to audio-visual synchronisation problems as discuss in Section 2.5.3. These two major drawbacks are then confirmed by the unacceptable objective scores results of the baseline system using sample-based approach as shown in Table 8.1.

Table 8.1: The objective scores results of the sample-based approach that predicting dynamic viseme sequences from the traditional phoneme to dynamic viseme conversion [98].

System	Unit	Correlation	NRMSE	GV
Unit-sel[98]	DV_PRED	0.44(±0.13)	14.64(±2.07)	1053.85(±246.83)

8.3 Proposed phonemes to dynamic visemes conversion

To improve the quality of the dynamic viseme sequence and to avoid problems from the traditional approach, we propose that the mapping process should be done in the same manner as the process of creating dynamic visemes. In this section we propose a novel method that maps a sequence of dynamic visemes from AAM parameters rather than phonemes, as illustrated in Figure 8.3.

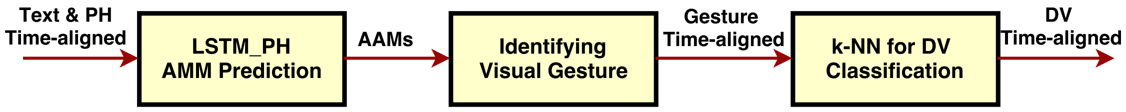


Figure 8.3: The digram of our proposed phonemes to dynamic visemes conversion.

To find a sequence of AAM parameters from a given text and phoneme transcription in a real setting, our proposed method in Figure 8.3 is introduced LSTM_PH AAM predictor into the system and used this predictor to predict the AAM parameters. Note that, we adopted an encoder-decoder bidirectional LSTM based on phoneme units (PH-LSTM System) because this predictor has been shown to be very effective at sequence modelling, as confirmed by the results in Table 8.2. It can be seen that the correlation and NRMSE of the DNN_PH and LSTM_PH Systems are no significant difference. In this case, we choose LSTM_PH System over DNN_PH System because of the major difference in GV results.

Table 8.2: The objective score results of three different statistical parametric system including HMM, DNN and LSTM System based on phoneme units.

System	Unit	Correlation	NRMSE	GV
HMM	PH	0.78(±0.08)	8.17(±1.88)	849.22(±165.91)
DNN	PH	0.83(±0.06)	7.43(±1.84)	992.09(±196.22)
LSTM	PH	0.82(±0.07)	7.53(±1.90)	1240.97(±245.56)

To find a sequence of dynamic visemes from AAMs, we use the same manner as the process of creating dynamic visemes, as described in Section 4.4.2.1. Firstly, AAMs are segmented into visual gestures using "Identifying Visual Gesture" (as illustrated at "G tier" in Figure 8.4). Secondly, each visual gesture, g, is mapped to supervectors and subsequently the dynamic viseme class using the 1-nearest neighbour between the supervector of the predicted and trained dynamic viseme (as illustrated at "DV tier" in Figure 8.4). An example of phoneme, visual gesture and dynamic viseme labels for the word "word" is shown in Figure 8.4. It can be seen that our approach gives dynamic viseme boundaries that are not necessary aligned to phoneme boundaries. This leads to our approach overcome the boundary problem of the traditional approach.

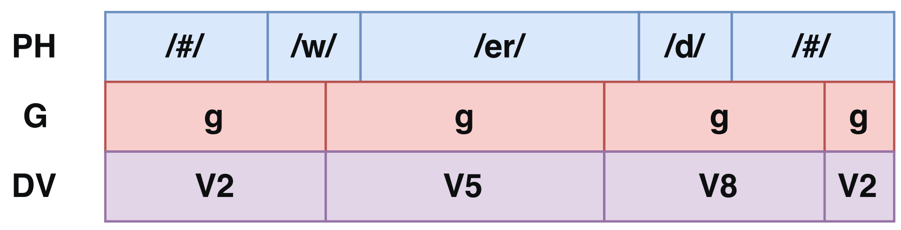


Figure 8.4: An example of phoneme, visual gesture and dynamic viseme transcriptions for the word "word".

So far, dynamic viseme labels have been available. Using the direct mapping approach it is possible to predict a sequence of AAMs from DVs. However this was not considered as we found that a simple one dynamic viseme to one lip motion results in poor animations. More importantly, we now use the model that we proposed in the previous chapter.

8.3.1 Full pipeline visual speech synthesis

This section describes our full pipeline diagram of visual speech synthesis in the case of reference dynamic viseme sequence is not available, as depicted in Figure 8.5. We proposed a novel method that maps a given text input and phoneme transcription into dynamic transcription, as described in the previous section. Consequently, these predicted dynamic viseme transcriptions are used to extract input features in the DV input feature extraction module and then the main predictor, LSTM_PH_DV model, is used to generate a sequence of AAMs as an output using both phoneme information and predicted dynamic viseme information. Finally, the AAMs in each frame are reconstructed and formed lip motions.

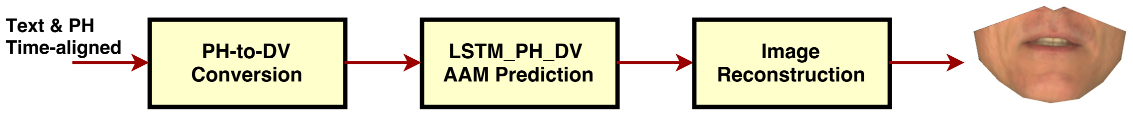


Figure 8.5: The framework of full pipeline synthesis.

The objective score results in Table 8.3 show that the errors from the predicted dynamic viseme sequences are slightly higher in terms of RMSE and lower in terms of correlation comparing with the ground-truth dynamic viseme sequences. Surprisingly, with the inspection of lip motion outputs, the synthesiser is still able to do a reasonable job at reproducing AAM parameters using predicted dynamic viseme units and groud-truth phoneme units. We refer to the supplementary video for animation results.

Table 8.3: The objective scores results of LSTM System with and without ground-truth dynamic visemes labels.

System	Unit	Correlation	NRMSE	GV
LSTM	PH	0.82(±0.07)	7.53(±1.90)	1240.97(±245.56)
LSTM	PH_DV_PRED	0.80(±0.07)	7.98(±1.94)	1425.82(±255.84)
LSTM	PH_DV	0.88(±0.05)	6.07(±1.27)	1478.21(±268.81)

To explore the effect of predicting dynamic viseme sequences, Figure 8.6 depicted AAM tracks generated using the reference (LSTM_PH_DV) and predicted (LSTM_PH_DV_PRED) dynamic viseme sequences along with the original track for the phrase “collects rare and novel”. The AAM track from the reference dynamic viseme sequence to the original shows it to follow more closely than using the predicted sequence although the underlying structure is clearly captured. Comparing the inspection animations in Figure 8.7, using the reference and predicted dynamic visemes look more similar although it is interesting to observe the slight over-articulation of the word “rare”. More experiments about preference tests can be found in Section 8.4.

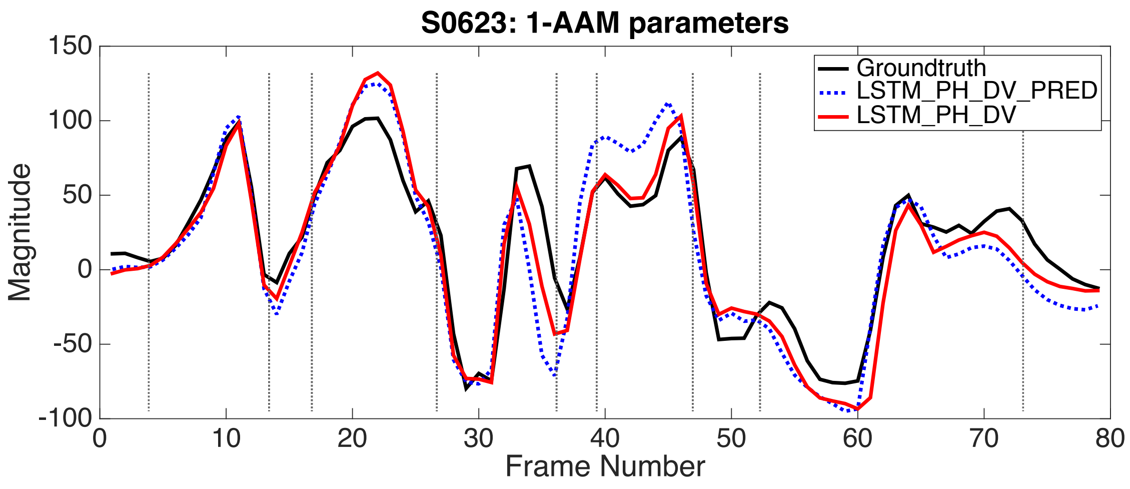


Figure 8.6: AAM trajectories generated using reference (LSTM_PH_DV) and predicted dynamic viseme sequences (LSTM_PH_DV_PRED), and for comparison the reference AAM track (Groundtruth), for the phrase “collects rare and novel”.

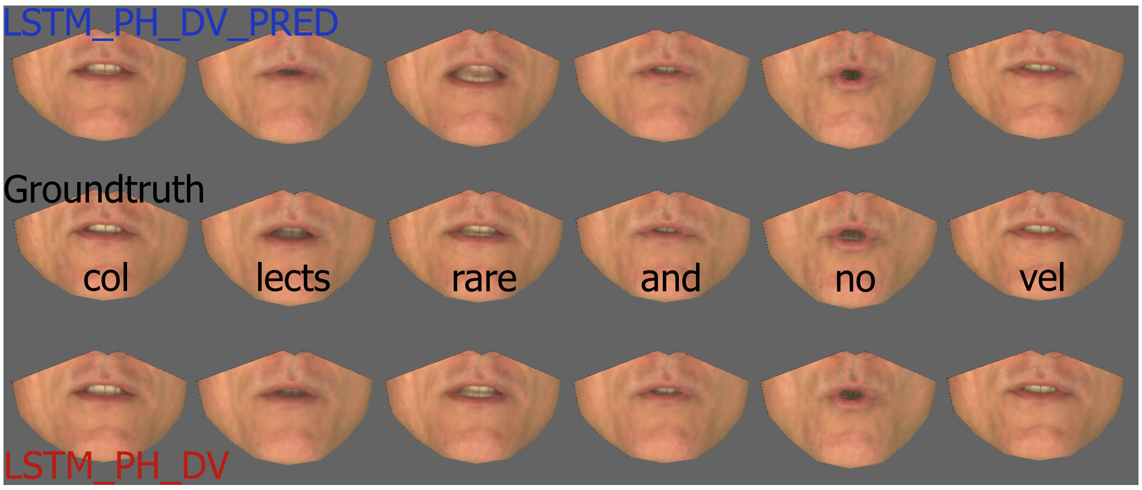


Figure 8.7: Sequence of lip images taken from animations produced using reference (LSTM_PH_DV) and predicted dynamic viseme sequences (LSTM_PH_DV_PRED), and for comparison using the reference AAM track (Groundtruth), for the phrase “collects rare and novel”.

8.4 Preference tests

The aim of the preference tests is to determine the best approach if no reference DV sequence is available. Two comparisons are made, the first comparison a system using predicted DV sequence to a phoneme LSTM system (LSTM_PH vs. LSTM_PH_DV_PRED). The second test compares using the predicted DV sequence against the reference to see the different (LSTM_PH_DV vs. LSTM_PH_DV_PRED). Side-by-side videos were shown to viewers and asked a simple question "Tell us which on looks more natural". They were allowed to watch the sequences as many times as needed. Then, viewers made their decision based on a binary choice: "Left" or "Right".

In this test the animations were generated by AAM sequences from three different systems: LSTM_PH, LSTM_PH_DV and LSTM_PH_DV_PRED. Our goal was to compare the systems when ground-truth DV labels are available (LSTM_PH_DV) and not available (LSTM_PH_DV_PRED). Moreover, we also compared the LSTM_PH_DV System to an LSTM system using only phoneme-based speech units (LSTM_PH) without the DV speech units. Each combination comprised four sentences that were randomly selected from the 250 sentences testing data. Also, each pair could evaluated by at least 10 subjects. Note that, this test is evaluated under a binomial test with $\alpha = 0.01$ by the same 30 participants that completed the Turing and preference tests in Section 7.3.8.

Table 8.4: Result of preference test on full pipeline synthesis system.

System A	System B	Percentage	p-value
LSTM_PH	LSTM_PH_DV_PRED	**31.67/68.33**	6.43e-05
LSTM_PH_DV	LSTM_PH_DV_PRED	56.30/43.70	0.3079

Table 8.4 shows that the combined phoneme units and predicted dynamic viseme LSTM (LSTM_PH_DV_PRED) system is significantly preferred to the LSTM system using only Phonemes units (LSTM_PH), as participants chose LSTM_PH_DV_PRED at about 68% with p-values=6.43e-05. This is somewhat interesting result because the objective scores especially correlation and NRMSE of LSTM_PH are better than LSTM_PH_DV_PRED System. This result suggests that GV result can be used to indicate the quality of the synthesiser, and the similar result was reported in the work from Toda and Tokuda [108]. They found that an increasing dynamic range is able to enhance quality of synthesised speech. Another interesting result is a pair-

wise comparison between LSTM System with and without ground-truth dynamic visemes labels. It shows that there is no clear preference of LSTM_PH_DV System over LSTM_PH_DV_PRED System with p-values=0.3079. Therefore, it can be concluded that our dynamic viseme conversion approach is able to produce reasonable animation results. We refer to the supplementary video for animation results.

Even though acceptable results have been achieved, this system still has two major limitations which are audiovisual intonation and photo-realism. The first limitation is related to the mismatch between audio and visual intonation, which means lip animation outputs do not sync with rhythms and melodies (intensity, pitch accent, expression) of acoustic speech. We found that the use of textual information only is not able to produce animations related to the right rhythm and melody of acoustic speech because the same sentence can be uttered in different patterns. Typically, pitch accent can be put in different positions in the sentence for example "i **LOVE** you", "**I** love you", "i love **YOU**". Then, the different pitch accent position from the acoustic speech will be produce the difference lip animations. Figure 8.8 shows a sequence of lip images taken from animations produced using this system (LSTM_PH_DV_PRED) compared with the use of reference AAM track (Groundtruth). It can be seen that the animation from this system still generates wrong lip sequences or is not flexible in terms of open/close mouth which are originated from the mismatch between audio and visual intonation as discussed earlier. To overcome this problem, the inclusion of acoustic information as one of the input features could solve this problem, as we can include the right intonation for both acoustic speech and visual speech into the output animations.

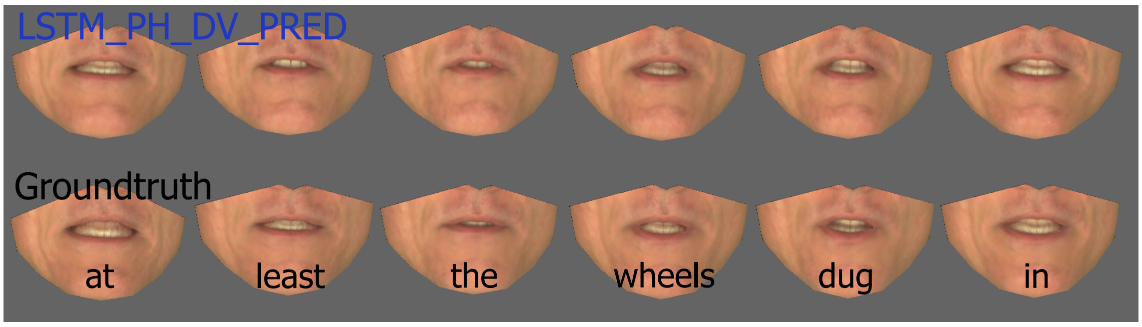


Figure 8.8: Sequence of lip images taken from animations produced using predicted dynamic viseme sequences (LSTM_PH_DV_PRED), and for comparison using the reference AAM track (Groundtruth), for the phrase "At least the wheels dug in". The frames correspond to the word at least shows that the animation from this system (LSTM_PH_DV_PRED) still generate wrong lip sequences and not flexible in terms of open/close mouth.

The second limitation is related to the output quality. Some viewers comment that the quality of the output animations, especially information inside the mouth (teeth and tongue), is degraded. This suggests that the dimensionality reduction of AAMs cannot generate a high quality output. The hybrid approach between statistical-based and sample-based approach [114] is one of the interesting solutions for this problem. Additionally, we also found that the degradation inside the mouth still happens even in the original video. Figure 8.9 shows yellow balls when captured at different video frame rates from low to high. It shows that the yellow ball of the low video frame rate between 15-60fps have more motion blur, while the high video frame rate at 96fps is able to reduce motion blur. This means the common video frame rate of the audiovisual database (for example 29.97fps for KB-2k [98], 50fps for LIPS [106]) might not good enough for visual speech synthesis systems. We suggest that the use of high video frame rate at 100 or 200fps is one of the possible solutions, as we found that there is no degradation of images when capturing videos with the high video frame rate.

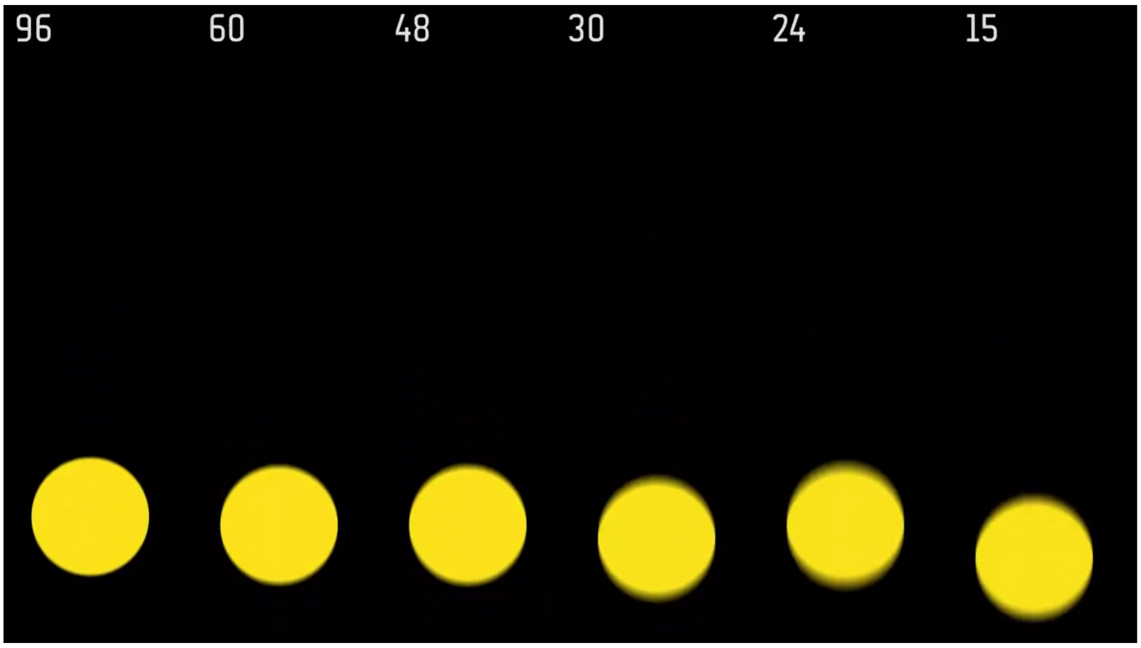


Figure 8.9: Examples of yellow ball when capturing in different video frame rates between 15fps and 96fps [1].

Chapter 9

Conclusion and Future work

9.1 Conclusion

This thesis has described three basic statistical parametric visual speech synthesis approaches including HMM-based, DNN-based, encoder-decoder LSTM-based synthesis to generate lip animations from a textual input. We concluded this work by considering seven research questions that have arisen.

9.1.1 What is the most appropriate unit of speech?

We have investigated what the most appropriate basic building block units of speech are. As mentioned in Section 2.5.1, phonemes are the basic units of speech for various acoustic speech applications (e.g. audio TTS and ASR system). However, there is no standard basic building block of visual speech compared with that of acoustic speech. A many-to-one phoneme-to-viseme mapping, namely static visemes, is assumed to be the basis of visual speech units, but they are not a good unit for many reasons as we described in Section 2.5.3, primaily that the same phoneme can have different lip-shapes. Hence, we incorporated visual speech units from [98], namely dynamic visemes units, which represent a group of similar lip movements instead of a static lip shapes. The experiments based on HMM synthesis in Section 5.3.1.7, a DNN synthesis in Section 6.3.1.5 and an LSTM synthesis in Section 7.3.4 confirmed that the use of phonemes or static visemes is less suitable for visual speech synthesis than dynamic visemes. Using dynamic visemes as the basic building block units of visual speech is able to bring the benefits of visual information from the groups of similar visual appearances and shows a big improvement in terms of objective scores.

9.1.2 How can discontinuity be reduced?

The next issue we addressed was the frame-wise problem and we focused on how can discontinuities be avoided to produce a smooth and realistic visual speech signal. For HMM-based synthesis, we have used the same framework as audio HMM synthesis which used the relationships between static and dynamic features to generate a smoother output with the maximum likelihood parameter generation algorithm [111] (described in Section 3.2.4). For DNN-based and encoder-decoder LSTM-based, our preliminary work from Section 6.3.1.3 and the findings from [118] reported that the post-processing MLPG algorithm is able to avoid step-wise trajectories, but this technique introduces the over smoothing problem. Hence, we proposed an alternative way to deal with the frame-wise problem for DNN and LSTM architecture. The use of frame context features have been proposed in Section 6.3.1.4 and we show that these features are able to solve the step-wise problem and also do not introduce over smoothing as much as MLPG algorithm does.

9.1.3 What are the best input and output feature representations?

For this issue, we further analysed how much does the performance improve according to the input and output feature representations. So far we know that the dynamic viseme is the most suitable unit of speech for visual speech synthesis. Related to this, we further examine other useful linguistic features that we divide into five levels of hierachy including segment, syllable, word, phrase and utterance. We found that all features make some contribution to the model. Interestingly, the findings in Section 6.3.1.5 suggest that the combining two building block of speech units: phonemes and dynamic visemes units gives a substantial increase in performance. It can be concluded that information relating to acoustics can be extracted from phonemes and the information of visual speech can be extracted from dynamic visemes. Moreover, our findings also show that the improvement of the model does not only come from the input representation but also output representation. The traditional output representation for ASR and audio TTS often used dynamic features to learn temporal changes in speech signals [40]. In Section 6.2.2 and 7.3.6, we found that the sliding window of the raw output for the neural network approach also improves smoothness with a window length of three.

9.1.4 What is the most appropriate model?

Next, we have evaluated how much does the performance increase according to three common machine learning techniques: HMM-based, DNN-based and encoder-decoder LSTM-based synthesis systems. Note that these three systems all use the same input and output representations. The objective scores have shown that the encoder-decoder LSTM architecture is the best learning technique (as reported in Section 7.3.7) and the second best is DNN-based architecture. Similar findings from [36, 118] and the subjective results in Section 7.3.8 also confirmed that our encoder-decoder LSTM architecture is significantly close to original as the viewers cannot identify the difference between the reconstructed lip images from the AAM ground-truth parameters and that from the predicted AMM parameters (described in 7.3.8.1). It can be concluded that frame-by-frame synthesis outperforms state-based synthesis and the advanced encoder-decoder LSTM architecture is better that the normal feedforward neural network architecture.

9.1.5 What relation in there between objective and subjective measure?

We are also interested into how to use objective measures to indicate the perceived quality of the animation from the human subjects. As we know the objective measures can be computed automatically and repeated, but they only estimate the quantity from the distance between each point of the ground-truth and synthesised. While the subjective measures are time consuming and require a human's opinion, they do indicate the quality of the lip animations. This work included global variance (GV) as an additional objective measure along with two common objective measures: correlation and RMSE. Interestingly, we found that the GV results are better predictor of realism and naturalness of humans' perceived quality, as shown in Section 7.3.8.

9.1.6 How much training data is needed?

Another interesting issue is how much training data is required to train a realistic visual TTS based on statistical parametric approaches. This issue is a common question in various applications. In the case of our visual TTS application, one of the limitations is the size of audiovisual database. As mentioned earlier, most publicly available audiovisual databases for visual speech synthesis have limited vocabulary and are too small. In our experiments in Section 7.3.7, we first found the common results that the more training data gives to better performance. Another interesting result shows that the modern neural network learning approaches can learn a better model compared with HMM-based approach when trained with the same amount of training data. It clearly shows that the satisfactory number of training data is about 1000-1200 utterances. This finding is similar to the standard training data of audio TTS (e.g CMUARCTIC).

9.1.7 Can visual TTS be made practical?

The final issue is focused on how much the accuracy decrease according to the absence of dynamic viseme sequences. The earlier results are assumed that dynamic viseme transcriptions are available. We also proposed a novel technique to map dynamic visemes from phoneme sequences, as described in Section 8.3. Our preference test reflects that our proposed dynamic viseme conversion method is able to produce the better animations compared with a baseline phoneme LSTM system, and found no significant difference between animations generated by the ground-truth and predicted dynamic visemes sequences.

9.2 Future Work

So far, this thesis has shown that it is possible to create smooth visual parameter outputs and to improve co-articulation for neutral speech. We are interested in three areas in the future. Firstly, we are interested on creating a new facial animation by adding new audiovisual database. As mentioned earlier, the output of this thesis is based on male single speaker but our proposed framework is able to extend to the new speaker by using the same method that explained in this thesis, as follows;

- Extracting AAM visual parameters as described in Section 4.3.
- Learning dynamic viseme units as described in Section 4.4.
- Representing input and output features for encoder-decoder LSTM-RNN architecture as described in Section 7.2.
- Modelling the network for encoder-decoder LSTM-RNN architecture as described in Section 7.2.3.2.
- Tuning the hyperparameters of the network including a learning rate, momentum, dropout, minibatch size as described in Section 7.3.2.
- Training LSTM_PH and LSTM_PH_DV system as described in Section 7.3.4.
- Building the final system as described in Section 8.3.1.

Secondly, we would like to focus on expressive speech with various emotion expressions. There have been various successful scenarios for these, for example, we have to prepare a new audiovisual speech database that contains these kinds of emotion expressions and apply model adaptation or model interpolation techniques. However, we are interested to take the concept from Shaw [91] which takes neutral speech and a target type of emotion as an input and generates expressive speech as an output.

Thirdly, we are also interested on advanced deep learning architectures. Another idea we aim to combine all separate modules including text-to-neutral visual speech and neutral visual speech to expressive visual speech into a single end-to-end model. This will benefit from new advanced techniques from deep learning methods.

Appendices

Appendix A

Appendix A

The following table is a set of phonetic questions that uses in this work. These questions are modified from the questions of the acoustic-only TTS system and selected only the phonetic question from the original set of contextual questions in HTS [126].

ID	Question	Descriptions
1	Is the phoneme vowel?	/eh,ih,uw,ah,ax,ao,ey,ay,ow,ea,ua,iy, ae,uh,oh,er,aa,oy,aw,ia/
2	Is the phoneme consonant?	/p,k,th,sh,d,d,v,z,m,w,y,jh,hh,t,f, s,b,g,dh,zh,n,r,l,ch,ny/
3	Is the phoneme plosive?	/b,d,g,k,p,t/
4	Is the phoneme affricative?	/ch,jh/
5	Is the phoneme nasal?	/m,n,ng/
6	Is the phoneme fricative?	/f,v,th,dh,s,z,sh,zh,hh/
7	Is the phoneme approximant?	/w,r,y/
8	Is the phoneme lateral?	/l/
9	Is the phoneme bilabial?	/p,b,m/
10	Is the phoneme labiodental?	/f,v/
11	Is the phoneme dental?	/th,dh/
12	Is the phoneme alveolar?	/t,d,n,s,z,l/
13	Is the phoneme postalveolar?	/ch,jh,sh,zh,r/

14	Is the phoneme palatal?	/y/
15	Is the phoneme velar?	/k,g,ng,w/
16	Is the phoneme glottal?	/hh/
17	Is the phoneme unvoiced consonant?	/p,f,th,t,s,ch,sh,k,hh/
18	Is the phoneme voiced consonant?	/b,m,v,dh,d,n,z,l,jh,zh,r,y,g,ng,w/
19	Is the phoneme voiced plosive?	/b,d,g/
20	Is the phoneme unvoiced plosive?	/p,t,k/
21	Is the phoneme voiced fricative?	/v,dh,z,zh/
22	Is the phoneme unvoiced fricative?	/f,th,s,sh,hh/
23	Is the phoneme semi consonant?	/y,w/
24	Is the phoneme sibilant consonant?	/ch,jh,s,z,sh,zh/
25	Is the phoneme sibilant affricate?	/ch,jh/
26	Is the phoneme sibilant fricative?	/s,z,sh,zh/
27	Is the phoneme front vowel?	/iy,ih,en,ae/
28	Is the phoneme central vowel?	/er,ax,ah/
29	Is the phoneme back vowel?	/uw,uh,ao,aa,oh/
30	Is the phoneme front consonant?	/b,f,m,p,v,w/
31	Is the phoneme central consonant?	/d,dh,dx,l,n,r,s,t,th,z,zh/
32	Is the phoneme back consonant?	/ch,g,hh,jh,k,ng,sh,y/
33	Is the phoneme front stop?	/b,p/
34	Is the phoneme central stop?	/d,t/
35	Is the phoneme back stop?	/g,k/
36	Is the phoneme front fricative?	/f,v/
37	Is the phoneme central fricative?	/dh,s,th,z/
38	Is the phoneme back fricative?	/ch,jh,sh,zh/
39	Is the phoneme front?	/b,f,m,p,v,w,iy,ih,en,ae/
40	Is the phoneme central?	/d,dh,dx,l,n,r,s,t,th,z,zh,er,ax,ah/
41	Is the phoneme back?	/ch,g,hh,jh,k,ng,sh,y,uw,uh,ao,aa,oh/
42	Is the phoneme long vowel?	/iy,er,uw,ao,aa/
43	Is the phoneme short vowel?	/ih,eh,ae,ax,ah,uh,oh/

44	Is the phoneme vowel close?	/iy,ih,uw,uh/
45	Is the phoneme vowel mid?	/eh,er,ax,ao/
46	Is the phoneme vowel open?	/ae,ah,aa,oh/
47	Is the phoneme vowel front?	/iy,ih,eh,ae/
48	Is the phoneme vowel central?	/er,ax,ah/
49	Is the phoneme vowel back?	/uw,uh,ao,aa,oh/
50	Is the phoneme dipthong vowel?	/ey,ay,oy,ow,aw,ia,ua,ea/
51	Is the phoneme dipthong closing?	/ey,ay,oy,ow,aw/
52	Is the phoneme dipthong centring?	/ia,ua,ea/
53	Is the phoneme IVowel?	/ih,iy,ia/
54	Is the phoneme EVowel?	/eh,ey,er/
55	Is the phoneme AVowel?	/ay,ae,aa,aw,ao/
56	Is the phoneme OVowel?	/ao,ow,oy,oh/
57	Is the phoneme UVowel?	/ah,ax,ua,uh,uw/

Table A.1: A summary of frame feature extraction.

Bibliography

[1] 96 - 60 - 48 - 30 - 24 - 15 FPS Comparison. `https://www.reddit.com/r/pcmasterrace/comments/3728sz/96_60_48_30_24_15_fps_comparison/`. Accessed: 2018-02-14.

[2] Expectation Maximization, EM. `http://sens.tistory.com/304`. Accessed: 2018-01-1.

[3] Gabriel Antunes Abrantes and Fernando Pereira. MPEG-4 facial animation technology: survey, implementation, and results. *Circuits and Systems for Video Technology, IEEE Transactions on*, 9(2):290–305, mar 1999. ISSN 1051-8215. doi: 10.1109/76.752096.

[4] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015. URL `http://arxiv.org/abs/1512.02595`.

[5] Robert Anderson, Bjorn Stenger, Vincent Wan, and Roberto Cipolla. Expressive visual text-to-speech using active appearance models. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3382–3389, June 2013. doi: 10.1109/CVPR.2013.434.

[6] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009. ISSN 1935-8237. doi: 10.1561/2200000006. URL `http://dx.doi.org/10.1561/2200000006`.

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2): 157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL `http://dx.doi.org/10.1109/72.279181`.

[8] Jonas Beskow. Talking heads communication, articulation and animation. *TMH-QPSR*, 37(2):053–056, 1996.

[9] Elif Bozkurt, Cigdem Eroglu Erdem, Engin Erzin, Tanju Erdem, and Mehmet Ozkan. Comparison of phoneme and viseme based acoustic units for speech driven realistic lip animation. In *2007 3DTV Conference*, pages 1–4, May 2007. doi: 10.1109/3DTV.2007.4379417.

[10] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 353–360, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: 10.1145/258734.258880. URL `http://dx.doi.org/10.1145/258734.258880`.

[11] W. M. Campbell, D. E. Sturim, and D. A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE Signal Processing Letters*, 13(5):308–311, May 2006. ISSN 1070-9908. doi: 10.1109/LSP.2006.870086.

[12] Tim Capes, Paul Coles, Alistair Conkie, Ladan Golipour, Abie Hadjitarkhani, Qiong Hu, Nancy Huddleston, Melvyn Hunt, Jiangchuan Li, Matthias Neeracher, Kishore Prahallad, Tuomo Raitio, Ramya Rasipuram, Greg Townsend, Becci Williamson, David Winarsky, Zhizheng Wu, and Hepeng Zhang. Siri on-device deep learning-guided unit selection text-to-speech system. In *Proc. In-*

terspeech 2017, pages 4011–4015, 2017. doi: 10.21437/Interspeech.2017-1798. URL `http://dx.doi.org/10.21437/Interspeech.2017-1798`.

[13] Kai Chen, Zhi-Jie Yan, and Qiang Huo. Training deep bidirectional LSTM acoustic model for LVCSR by a context-sensitive-chunk BPTT approach. September 2015. URL `https://www.microsoft.com/en-us/research/publication/training-deep-bidirectional-lstm-acoustic-model-lvcsr-context-sensitive-ch`

[14] Tsuhan Chen. Audiovisual speech processing. *IEEE Signal Processing Magazine*, 18(1):9–21, Jan 2001. ISSN 1053-5888. doi: 10.1109/79.911195.

[15] 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. URL `http://arxiv.org/abs/1406.1078`.

[16] Michael M. Cohen and Dominic W. Massaro. Modeling coarticulation in synthetic visual speech. In *Models and Techniques in Computer Animation*, pages 139–156. Springer-Verlag, 1993.

[17] Martin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao. An audio-visual corpus for speech perception and automatic speech recognition. *Acoustical Society of America Journal*, 120:2421, 2006. doi: 10.1121/1.2229005.

[18] Geert Coorman, Justin Fackrell, Peter Rutten, and Bert van Coile. Segment selection in the l & h realspeak laboratory tts system. In *In Proceedings of the Intl. Conf. on Spoken Language Processing*, pages 395–398, 2000.

[19] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. In *Proc. Fifth European Conf. Computer Vision.*, volume 2, pages 484–498, 1998.

[20] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):681–685,

Jun 2001. ISSN 0162-8828. doi: 10.1109/34.927467. URL `http://dx.doi.org/10.1109/34.927467`.

[21] Eric Cosatto. *Sample-Based Talking-Head Synthesis.* PhD thesis, Signal Processing Laboratory, Swiss Federal Institute of Technology, 2002.

[22] Michael A. A. Cox and Trevor F. Cox. *Multidimensional Scaling*, pages 315–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-33037-0. doi: 10.1007/978-3-540-33037-0_14. URL `https://doi.org/10.1007/978-3-540-33037-0_14`.

[23] Salil Deena, Shaobo Hou, and Aphrodite Galata. Visual speech synthesis by modelling coarticulation dynamics using a non-parametric switching state-space model. In *ICMI-MLMI*, page 29, 2010.

[24] Salil Prashant Deena. *Visual speech synthesis by learning joint probabilistic models of audio and video.* PhD thesis, School of Computing Sciences, The University of Manchester, 2012.

[25] John Dines, Junichi Yamagishi, and Simon King. Measuring the gap between HMM-based ASR and TTS. *Selected Topics in Signal Processing, IEEE Journal of*, 4(6):1046–1058, Dec 2010. ISSN 1932-4553. doi: 10.1109/JSTSP.2010.2079315.

[26] Jiang Dongmei, Xie Lei, Zhao Rongchun, Werner Verhelst, Ilse Ravyse, and Hichem Sahli. Acoustic viseme modelling for speech driven animation: a case study. In *IEEE Benelux Workshop on MPC*, December 2002.

[27] Yangzhou Du and Xueyin Lin. Realistic mouth synthesis based on shape appearance dependence mapping. *Pattern Recognition Letters*, 23(14):1875–1885, 2002.

[28] Paul Ekman and Wallace V. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Consulting Psychologists Press, Palo Alto, 1978.

[29] Paul Ekman, Wallace V. Friesen, and Joseph C. Hager. *Facial Action Coding System: The Manual on CD ROM.* A Human Face, Salt Lake City, Palo Alto, 2002.

[30] Gwenn Englebienne, Tim Cootes, and Magnus Rattray. A probabilistic model for generating realistic lip movements from speech. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 401–408. Curran Associates, Inc., 2008. URL `http://papers.nips.cc/paper/3287-a-probabilistic-model-for-generating-realistic-lip-movements-from-spe` `pdf`.

[31] Saher Esmeir, Shaul Markovitch, and Claude Sammut. Anytime learning of decision trees. *Journal of Machine Learning Research*, 8:2007.

[32] Tony Ezzat and Tomaso Poggio. Miketalk: A talking facial display based on morphing visemes. In *Proceedings of the Computer Animation Conference*, pages 96–102, 1998.

[33] Tony Ezzat, Gadi Geiger, and Tomaso Poggio. Trainable videorealistic speech animation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 388–398, New York, NY, USA, 2002. ACM. ISBN 1-58113-521-1. doi: 10.1145/566570.566594. URL `http://doi.acm.org/10.1145/566570.566594`.

[34] Sascha Fagel. MASSY speaks english: adaptation and evaluation of a talking head. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association, Brisbane, Australia, September 22-26, 2008*, page 2324, 2008. URL `http://www.isca-speech.org/archive/interspeech_2008/i08_2324.html`.

[35] Sascha Fagel and Caroline Clemens. An articulation model for audiovisual speech synthesisdetermination, adjustment, evaluation. *Speech Communication*, 44(14):141 – 154, 2004. ISSN 0167-6393. doi: http://dx.doi.org/10.1016/j.specom.2004.10.006. URL `http://www.sciencedirect.com/`

science/article/pii/S0167639304001128. Special Issue on Audio Visual speech processing.

[36] Bo Fan, Lei Xie, Shan Yang, Lijuan Wang, and Frank K. Soong. A deep bidirectional LSTM approach for video-realistic talking head. *Multimedia Tools and Applications*, 75(9):5287–5309, 2016. ISSN 1573-7721. doi: 10.1007/s11042-015-2944-3. URL http://dx.doi.org/10.1007/s11042-015-2944-3.

[37] Cletus G. Fisher. Confusions among visually perceived consonants. *Journal of Speech and Hearing Research*, 11:796–804, 1968.

[38] Susan Fitt and Stephen Isard. Synthesis of regional english using a keyword lexicon. *International Speech Communication Association*, 1999.

[39] Shengli Fu, Ricardo Gutierrez-Osuna, Anna Esposito, Praveen K. Kakumanu, and Oscar N. Garcia. Audio/visual mapping with cross-modal hidden Markov models. *IEEE Transactions on Multimedia*, 7(2):243–252, April 2005. ISSN 1520-9210. doi: 10.1109/TMM.2005.843341.

[40] Sadaoki Furui. Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 1991–1994, Apr 1986. doi: 10.1109/ICASSP.1986.1168654.

[41] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, January 2007. ISSN 1932-8346. doi: 10.1561/2000000004. URL http://dx.doi.org/10.1561/2000000004.

[42] Gerard Bailly and Maxime Berar and Frederic Elisei and M. Odisio. Audiovisual speech synthesis. *International Journal of Speech Technology*, 6: 331–346, 2003. ISSN 1381-2416. doi: 10.1023/A:1025700715107. URL http://dx.doi.org/10.1023/A%3A1025700715107.

[43] Felix A. Gers and Jurgen Schmidhuber. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, Nov 2001. ISSN 1045-9227. doi: 10.1109/72.963769.

[44] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011. URL `http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf`.

[45] Xavi Gonzalvo, Siamak Tazari, Chun an Chan, Markus Becker, Alexander Gutkin, and Hanna Silen. Recent advances in google real-time HMM-driven unit selection synthesizer. In *Interspeech 2016*, pages 2238–2242, 2016. doi: 10.21437/Interspeech.2016-264. URL `http://dx.doi.org/10.21437/Interspeech.2016-264`.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[47] Oxana Govokhina, Gérard Bailly, Gaspard Breton, and Paul Bagshaw. TDA: A new trainable trajectory formation system for facial animation. In *Interspeech*, pages 2474–2477, Pittsburgh, United States, September 2006. URL `https://hal.archives-ouvertes.fr/hal-00366489`.

[48] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`.

[49] F.J. Gravetter and L.B. Wallnau. *Statistics for the Behavioral Sciences*. Available Titles Aplia Series. Wadsworth, 2009. ISBN 9780495602200. URL `https://books.google.co.uk/books?id=wWFmkwxSUfUC`.

[50] R. Gutierrez-Osuna, P. K. Kakumanu, A. Esposito, O. N. Garcia, A. Bojorquez, J. L. Castillo, and I. Rudomin. Speech-driven facial animation with

realistic dynamics. *Trans. Multi.*, 7(1):33–42, February 2005. ISSN 1520-9210. doi: 10.1109/TMM.2004.840611. URL `http://dx.doi.org/10.1109/TMM.2004.840611`.

[51] Thomas Hain, Philip C. Woodland, Gunnar Evermann, Mark J. F. Gales, Xunying Liu, Gareth L. Moore, Dan Povey, and Lan Wang. Automatic transcription of conversational telephone speech. *IEEE Transactions on Speech and Audio Processing*, 13(6):1173–1185, Nov 2005. ISSN 1063-6676. doi: 10.1109/TSA.2005.852999.

[52] William I. Hallahan. Dectalk software: Text-to-speech technology and implementation. *Digital Technical Journal*, 7, 1995.

[53] Naomi Harte and Eoin Gillen. TCD-TIMIT: An audio-visual corpus of continuous speech. *IEEE Transactions on Multimedia*, 17(5):603–615, May 2015. ISSN 1520-9210. doi: 10.1109/TMM.2015.2407694.

[54] Sarah Hilder, Barry-John Theobald, and Richard Harvey. In pursuit of visemes. In *AVSP*, pages 8–2, 2010.

[55] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[56] Gregor Hofer, Junichi Yamagishi, and Hiroshi Shimodaira. Speech-driven lip motion generation with a trajectory HMM. In *Interspeech-2008*, pages 2314–2317, 2008.

[57] Chao-Kuei Hsieh and Yung-Chang Chen. Partial linear regression for audio-driven talking head application. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 281 –284, july 2005. doi: 10.1109/ICME.2005.1521415.

[58] Yuxiao Hu, Dalong Jiang, Shuicheng Yan, Lei Zhang, and Hongjiang Zhang. Automatic 3d reconstruction for face recognition. In *In Proceedings of the*

IEEE International Conference on Automatic Face and Gesture Recognition, pages 843–848, 2004.

[59] Michael I. Jordan. Chapter 25 - serial order: A parallel distributed processing approach. In John W. Donahoe and Vivian Packard Dorsel, editors, *Neural-Network Models of CognitionBiobehavioral Foundations*, volume 121 of *Advances in Psychology*, pages 471 – 495. North-Holland, 1997. doi: http://doi.org/10.1016/S0166-4115(97)80111-2. URL `http://www.sciencedirect.com/science/article/pii/S0166411597801112`.

[60] George Karypis. CLUTO - A Clustering Toolkit. In *Technical Report*. University of Minnesota, Department of Computer Science, Min- neapolis, 2002.

[61] Taehwan Kim, Yisong Yue, Sarah Taylor, and Iain Matthews. A decision tree framework for spatiotemporal sequence prediction. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 577–586, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783356. URL `http://doi.acm.org/10.1145/2783258.2783356`.

[62] Lukas Latacz, Yuk On Kong, Wesley Mattheyses, and Werner Verhelst. An overview of the vub entry for the 2008 blizzard challenge. In *Proc. Blizzard Challenge 2008*, 2008.

[63] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49430-0. doi: 10.1007/3-540-49430-8_2. URL `http://dx.doi.org/10.1007/3-540-49430-8_2`.

[64] Howard Lei and Nikki Mirghafori. Word-conditioned HMM supervectors for speaker recognition. In *in Proc. of Interspeech*, 2007.

[65] Dan Li and Jiang Qian. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, pages 471–475, Oct 2016. doi: 10.1109/CCI.2016.7778967.

[66] Anders Lofqvist. Speech as audible gestures. *Speech Production and Speech Modelling*, 55:289–322, 1990. doi: 10.1007/978-94-009-2037-8_12. URL `http://dx.doi.org/10.1007/978-94-009-2037-8_12`.

[67] Jos Mario De Martino, Lo Pini Magalhes, and Fbio Violaro. Facial animation based on context-dependent visemes. *Computers & Graphics*, 30 (6):971 – 980, 2006. ISSN 0097-8493. doi: http://dx.doi.org/10.1016/j.cag.2006.08.017. URL `http://www.sciencedirect.com/science/article/pii/S0097849306001518`.

[68] Takashi Masuko, Takao Kobayashi, Masatsune Tamurat, Jun Masubuchi, and Keiichi Tokuda. Text-to-visual speech synthesis based on parameter generation from HMM. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 6, pages 3745–3748 vol.6, May 1998. doi: 10.1109/ICASSP.1998.679698.

[69] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135 – 164, November 2004.

[70] Wesley Mattheyses and Werner Verhelst. Audiovisual speech synthesis: An overview of the state-of-the-art. *Speech Commun.*, 66(C):182–217, February 2015. ISSN 0167-6393. doi: 10.1016/j.specom.2014.11.001. URL `http://dx.doi.org/10.1016/j.specom.2014.11.001`.

[71] Wesley Mattheyses, Lukas Latacz, and Werner Verhelst. On the importance of audiovisual coherence for the perceived quality of synthesized visual speech. *EURASIP J. Audio Speech Music Process.*, 2009:1:1–1:12, January 2009. ISSN 1687-4714. doi: 10.1155/2009/169819. URL `http://dx.doi.org/10.1155/2009/169819`.

[72] Wesley Mattheyses, Lukas Latacz, and Werner Verhelst. Automatic viseme clustering for audiovisual speech synthesis. In *INTERSPEECH*, pages 2173–2176, 2011.

[73] Wesley Mattheyses, Lukas Latacz, and Werner Verhelst. Comprehensive

many-to-many phoneme-to-viseme mapping and its application for concatenative visual speech synthesis. *Speech Communication*, 55(78):857 – 876, 2013. ISSN 0167-6393. doi: http://dx.doi.org/10.1016/j.specom.2013.02.005. URL `http://www.sciencedirect.com/science/article/pii/S0167639313000319`.

[74] Harry McGurck and John MacDonald. Hearing lips and seeing voices. *Nature*, 264(246-248), 1976. URL `http://www.nature.com/nature/journal/v264/n5588/abs/264746a0.html;jsessionid=18B8F4F4E09E98C54D5410A41B4F3D4A`.

[75] Einar Meister, Rainer Metsvahi, and Sascha Fagel. Evaluation of the estonian audiovisual speech synthesis. In *Human Language Technologies - The Baltic Perspective - Proceedings of the Sixth International Conference Baltic HLT 2014, Kaunas, Lithuania, September 26-27, 2014*, pages 11–18, 2014. doi: 10.3233/978-1-61499-442-8-11. URL `https://doi.org/10.3233/978-1-61499-442-8-11`.

[76] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Frnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010. URL `http://www.icml2010.org/papers/432.pdf`.

[77] Slim Ouni, Michael M. Cohen, and Dominic W. Massaro. Training baldi to be multilingual: A case study for an arabic badr. *Speech Communication*, 45(2): 115 – 137, 2005. ISSN 0167-6393. doi: http://dx.doi.org/10.1016/j.specom.2004.11.008. URL `http://www.sciencedirect.com/science/article/pii/S0167639304001463`.

[78] Slim Ouni, Vincent Colotte, Utpala Musti, Asterios Toutios, Brigitte Wrobel-Dautcourt, Marie-Odile Berger, and Caroline Lavecchia. Acoustic-visual synthesis technique using bimodal unit-selection. *EURASIP Journal on Au-*

dio, Speech, and Music Processing, (2013:16), June 2013. doi: 10.1186/1687-4722-2013-16. URL `https://hal.inria.fr/hal-00835854`.

[79] Igor S. Pandzic and Robert Forchheimer, editors. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2003. ISBN 0470854626.

[80] Frederick I. Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference - Volume 1*, ACM '72, pages 451–457, 1972. doi: 10.1145/800193.569955. URL `http://doi.acm.org/10.1145/800193.569955`.

[81] Douglas A. Reynolds and Richard C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, Jan 1995. ISSN 1063-6676. doi: 10.1109/89.365379.

[82] Alex Rudnicky. The CMU pronunciation dictionary, release 0.7 a, 2007. *URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict*, 2007.

[83] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL `http://dl.acm.org/citation.cfm?id=104279.104293`.

[84] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL `http://dl.acm.org/citation.cfm?id=65669.104451`.

[85] Mark Sagar. Facial performance capture and expressive translation for king kong. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: 10.1145/1179849.1179882. URL `http://doi.acm.org/10.1145/1179849.1179882`.

[86] Keijiro Saino, Heiga Zen, Yoshihiko Nankaku, Akinobu Lee, and Keiichi Tokuda. An HMM-based singing voice synthesis system. In *Ninth International Conference on Spoken Language Processing*, 2006.

[87] Shinji Sako, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. HMM-based text-to-audio-visual speech synthesis. In *International Conference on Spoken Language Processing (ICSLP 2000)*, 2000.

[88] Kazuki Sato, Takashi Nose, and Akinori Ito. HMM-based photo-realistic talking face synthesis using facial expression parameter mapping with deep neural networks. *Journal of Computer and Communications*, 5(10):50 – 65, 2017. doi: https://doi.org/10.4236/jcc.2017.510006. URL `www.scirp.org/journal/PaperInformation.aspx?PaperID=78666`.

[89] Dietmar Schabus, Michael Pucher, and Gregor Hofer. Joint audiovisual hidden semi-markov model-based speech synthesis. *Selected Topics in Signal Processing, IEEE Journal of*, 8(2):336–347, 2014.

[90] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov 1997. ISSN 1053-587X. doi: 10.1109/78.650093.

[91] Felix Shaw. *Expressive Modulation of Neutral Visual Speech*. PhD thesis, School of Computing Sciences, University of East Anglia, 2015.

[92] Koichi Shinoda and Takao Watanabe. Acoustic modeling based on the mdl principle for speech recognition. In *EUROSPEECH*, pages 99–102, 1997.

[93] Koichi Shinoda and Takao Watanabe. Mdl-based context-dependent subword modeling for speech recognition. *Acoustical Science and Technology*, 21(2): 79–86, 2000. doi: 10.1250/ast.21.79.

[94] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[95] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2969033.2969173`.

[96] Masatsune Tamura, Takashi Masuko, Takao Kobayashi, and Keiichi Tokuday. Visual speech synthesis based on parameter generation from HMM: Speech-driven and text-and-speech-driven approaches. In *ICASSP-98*, pages 3745–3748, 1998.

[97] Paul Taylor, Alan W Black, and Richard Caley. The architecture of the festival speech synthesis system. In *3rd ESCA Workshop in Speech Synthesis*, pages 147–151, 1998.

[98] Sarah Taylor, Moshe Mahler, Barry-John Theobald, and Iain Matthews. Dynamic units of visual speech. In *Proceedings of the Symposium on Computer Animation*, pages 275–284, 2012.

[99] Sarah Taylor, Akihiro Kato, Iain Matthews, and Ben Milner. Audio-to-visual speech conversion using deep neural networks. In *Interspeech 2016*, pages 1482–1486, 2016. doi: 10.21437/Interspeech.2016-483. URL `http://dx.doi.org/10.21437/Interspeech.2016-483`.

[100] Sarah Taylor, Taehwan Kim, Yisong Yue, Moshe Mahler, James Krahe, Anastasio Garcia Rodriguez, Jessica Hodgins, and Iain Matthews. A deep learning approach for generalized speech animation. *ACM Trans. Graph.*, 36(4): 93:1–93:11, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073699. URL `http://doi.acm.org/10.1145/3072959.3073699`.

[101] Ausdang Thangthai and Barry-John Theobald. HMM-based visual speech synthesis using dynamic visemes. In *Auditory-Visual Speech Processing, AVSP 2015, Vienna, Austria, September 11-13, 2015*, pages 88–92, 2015. URL `http://www.isca-speech.org/archive/avsp15/av15_088.html`.

[102] Ausdang Thangthai, Ben Milner, and Sarah Taylor. Visual speech synthesis using dynamic visemes, contextual features and dnns. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 2458–2462, 2016. doi: 10.21437/Interspeech.2016-1084. URL `http://dx.doi.org/10.21437/Interspeech.2016-1084`.

[103] Barry-John Theobald. *Visual speech synthesis using shape and appearance models*. PhD thesis, School of Computing Sciences, University of East Anglia, 2003.

[104] Barry-John Theobald and Iain Matthews. Relating objective and subjective performance measures for aam-based visual speech synthesis. *IEEE Transactions on Audio, Speech and Language Processing*, 20(8):2378–2387, oct. 2012. ISSN 1558-7916. doi: 10.1109/TASL.2012.2202651.

[105] Barry-John Theobald, Andrew Bangham, Iain Matthews, and Gavin C Cawley. Near-videorealistic synthetic talking faces: implementation and evaluation. *Speech Communication*, 44(14):127–140, 2004. ISSN 0167-6393. doi: 10.1016/j.specom.2004.07.002. URL `http://www.sciencedirect.com/science/article/pii/S0167639304000822`.

[106] Barry-John Theobald, Sascha Fagel, Gérard Bailly, and Frédéric Elisei. LIPS2008: visual speech synthesis challenge. In *INTERSPEECH*, pages 2310–2313, 2008.

[107] Barry-John Theobald, Iain Matthews, Michael Mangini, Jeffrey R. Spies, Timothy R. Brick, Jeffrey F. Cohn, and Steven M. Boker. Mapping and manipulating facial expression. *Language and Speech*, 52(2-3):369–386, 2009. doi: 10.1177/0023830909103181. URL `http://las.sagepub.com/content/52/2-3/369.abstract`.

[108] Tomoki Toda and Keiichi Tokuda. A speech parameter generation algorithm considering global variance for HMM-based speech synthesis. *IEICE - Trans. Inf. Syst*, pages 816–824, 2007.

[109] Keiichi Tokuda and Heiga Zen. Fundamentals and recent advances in HMM-based speech synthesis. In *INTERSPEECH*, 2009.

[110] Keiichi Tokuda, Takao Kobayashi, and Satoshi Imai. Speech parameter generation from HMM using dynamic features. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 660–663 vol.1, May 1995. doi: 10.1109/ICASSP.1995.479684.

[111] Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 3, pages 1315–1318 vol.3, 2000. doi: 10.1109/ICASSP.2000.861820.

[112] Ashish Venna, Nitendra Rajput, and L Venkata Subramaniam. Using viseme based acoustic models for speech driven lip synthesis. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 3, pages III–533–6 vol.3, July 2003. doi: 10.1109/ICME.2003.1221366.

[113] Johan Verwey and Edwin H. Blake. The influence of lip animation on the perception of speech in virtual environments. In *Proceedings of the 8th Annual International Workshop on Presense, University College London*, pages 163–170, 2005.

[114] Lijuan Wang and Frank K. Soong. HMM trajectory-guided sample selection for photo-realistic talking head. *Multimedia Tools and Applications*, 74(22): 9849–9869, Nov 2015. ISSN 1573-7721. doi: 10.1007/s11042-014-2118-8. URL `https://doi.org/10.1007/s11042-014-2118-8`.

[115] Lijuan Wang, Yi-Jian Wu, Xiaodan Zhuang, and Frank K. Soong. Synthesizing visual speech trajectory with minimum generation error. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4583, May 2011. doi: 10.1109/ICASSP.2011.5947374.

[116] Lijuan Wang, Wei Han, and Frank Soong. High quality lip-sync animation for 3d photo-realistic talking head. In *2012 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pages 4529–4532, March 2012. doi: 10.1109/ICASSP.2012.6288925.

[117] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *CoRR*, abs/1510.06168, 2015. URL `http://arxiv.org/abs/1510.06168`.

[118] Oliver Watts, Gustav Eje Henter, Thomas Merritt, Zhizheng Wu, and Simon King. From HMMs to DNNs: where do the improvements come from? In *Proc. ICASSP*, volume 41, pages 5505–5509, Shanghai, China, March 2016. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7472730`.

[119] Takayoshi Yoshimura. *Simultaneous modeling of phonetic and prosodic parameters, and characteristic conversion for HMM-based text-to-speech systems.* PhD thesis, Department of Electrical and Computer Engineering Nagoya Institute of Technology, 2002.

[120] Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Duration modeling for HMM-based speech synthesis. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP)*, pages 29–32, 1998.

[121] Takayoshi Yoshimura, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In *Sixth European Conference on Speech Communication and Technology*, pages 2347–2350, 1999.

[122] Atef Ben Youssef, Hiroshi Shimodaira, and David A. Braude. Speech driven talking head from estimated articulatory features. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 4573–4577, 2014. doi: 10.1109/ICASSP.2014.6854468. URL `https://doi.org/10.1109/ICASSP.2014.6854468`.

[123] Jiahong Yuan and Mark Liberman. Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America*, 123(5):3878, 2008.

[124] Matthew D. Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Mark Z. Mao, K. Yang, Quoc V. Le, Patrick Nguyen, Andrew W. Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffrey E. Hinton. On rectified linear units for speech processing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521, May 2013. doi: 10.1109/ICASSP.2013.6638312.

[125] Heiga Zen and Hasim Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4470–4474, 2015.

[126] Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan W. Black, and Keiichi Tokuda. The HMM-based speech synthesis system version 2.0. In *Proc. of ISCA SSW6*, pages 294–299, 2007.

[127] Heiga Zen, Keiichi Tokuda, Takashi Masuko, Takao Kobayasih, and Tadashi Kitamura. A hidden semi-markov model-based speech synthesis system. *IEICE - Trans. Inf. Syst.*, E90-D(5):825–834, May 2007. ISSN 0916-8532. doi: 10.1093/ietisy/e90-d.5.825. URL `http://dx.doi.org/10.1093/ietisy/e90-d.5.825`.

[128] Heiga Zen, Andrew Senior, and Mike Schuster. Statistical parametric speech synthesis using deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7962–7966, 2013.

[129] Xinjian Zhang, Lijuan Wang, Gang Li, Frank Seide, and Frank K. Soong. A new language independent, photo-realistic talking head driven by voice only. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 2743–2747, 2013. URL `http://www.isca-speech.org/archive/interspeech_2013/i13_2743.html`.