MASTER'S PROJECT

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Unsupervised Learning with Audio Visual Data

Author:
Dilan (CID: DC5215)

Date: September 3, 2017

ABSTRACT: WILL BE WRITTEN LATER

Contents

Part I
Introduction

Lip reading is a complex problem and its solution offers direct benefits for the hearing-impaired and provides a way to add information to silent video of a speaker's mouth movement. This may be useful in applications such as surveillance and restoring missing audio. Furthermore tackling the problem of lip reading can lead to further insights about speech production and provide usefull representations of mouth movement.

The challenge of creating a lip-reading model is not trivial given the complexity of the relationship between mouth movement and sounds production. Adding to this challenge, the mapping is many to one, impltying that the same initial mouth movements possibly leading to distinctly different final sounds. Advances in deep learning in recent decades have allowed more complex relationships to be modelled and have been successfully applied to the problem of lip reading. However these models often aim to reproduce text from a speaker's lip movements which requires labelled data and this may often be tedious and expensive to collect.

In this thesis we explore an alternative approach to lip reading using deep learning techniques. Instead of targeting text and aiming to contruct a speaker's sentences as is most commonly done, we look at modelling the sound produced by the speaker directly. Our models target a representation of the audio related to a speaker's video and look to reconstruct a speech signal from the speaker's mouth movements. This approach, if successful, requires no labelling and the 'naturally supervised' nature of this model implies that datasets would be much easier to collect. Only videos with a synchronised audio track would need to be collected. In addition, sentence grammar does not need to be taken into account and interpretability of the language is left to the listener allowing these models to generalise easily accross multiple languages.

Very little research has been done towards audio reconstruction in context of lip reading. In this paper we extend the latest deep learning model applied to this problem by looking at different architectures, features and tests of generalisability accross different speakers. The aim of this work to find a high performance model for the video to audio mapping and provides the basis for further research on this topic.

1 Summary of Contributions

In this work we have expnaded on the work produced by Ephrat and Peleg [15] and contributed the following:

- **Testing different learning parameters and architectures.** We have provided a performance comparison over different key parameters used in the deep learning training process and have experimented with different deep learning architectures for the video to audio mapping. In doing so we have found improved performance under certain settings.

- **The use of recurrent networks.** Deep learning models incorporating different types of recurrent networks have been tested and shown to have a significant impact on improving performance and provides our highest performing model. To the best of our knowledge recurrent networks have not been tested with video to audio models before.

- **Comparison between linear predictive coefficients and Spectograms as Audio Representations.** We tested the effect of two different audio features to be used as our target data and have provided a quantitative and qualitative assesment of the differences.

- **Multiple speaker evaluation.** The effect of evaluating and training our models on different speakers, male and female, was tested.

2 Applications of Video to Audio Lip Reading Models

Possible applications of a working lip reading models are provided below.

- **Surveillance.** Security cameras often do not record audio due to legal reasons. However if the law is broken, law enforcement may have the legal right and need to decode what certain individuals are saying in the video. Lip-reading models in general provide a way to do this.

- **Providing lower rank represenations of video data for lip reading.** Intermediate layers of a deep learning model necessarily encode information of the original input and can be used as inputs to other models. ([37],[41], [30]). This helps reduce training time significantly as models can now work with dimensionally reduced data and the encoded input may store usefull features.

- **Used to contruct sentences** The audio produced from a video to audio lip reading model could be input to a speech recognition system and provide a sentence level lip reader. Given the high accuracy of speech recognition systems ([**?**]) this could lead to a better results than direct mapping. Furthermore its use would be invaluable alongside video to text lip readers as a double

check.

- **Animation and Synchronisation.** Simulated mouth movements could be directly translated into audio signals for the sake of computer generated animations or help synchronise audio with mouth movement by correlating the predicted and actual signals.

Beyond the applications above, video to audio lip reading poses an interesting research question and a successful model could help identify what features and idiosyncrasies are important to the articulatory process and possibly inform human lip readers of better techniques.

Part II
Literature Review

1 Introduction

There is only very recent and limited research on the direct video to audio mapping using deep learning available, much more research has been directed to using deep learning for text synthesizing lip reading models ([9],[12],[7],[28],[40]). However they are often posed as classification problems and have inherently different architectures, having to account for grammar and vocabulary which video to audio models do not need to consider. A more directly related and well studied paradigm in literature is that of articulatory-acoustic mapping. The 'articulatory' here refers to mouth organs in general, including the tongue, mouth cavity, jaw and lips. The lip reading models we consider in this paper are a special subset of these models with the primary difference being that we only have access to a speaker's lip movements and no iniformation on the state of any other articulatory organs.

With the above in mind, we first present literature on the general articulatory-acoustic mapping problem and describe the models used in the past to address this problem. Doing so allows us to highlight the success, drawbacks and limits of past models while highlighting the need for more complex modelling and less invasive techniques which our models provide. As the literature is dealing with targeting acoustic information, the audio representations used and problems faced are informative.

The review will progress chronologically, starting with the earliest work first, moving on to the latest and directly related research on video to audio modelling using deep learning, including a brief discussion of video to text lip reading methods. Closely linked research is also addressed at the end of this chapter for the sake of completeness.

2 Background

The focus of this section will be to provide a brief background of different approaches however when appropriate, a summary of the underlying mathematics of a method will be presented in order to provide intuition and clarity.

2.1 Direct Modelling

When considering how to relate mouth movement to audio signlas, the earliest research focuses on mapping articulatory parameters to an audio spectogram. To

record the state of articulatory organs sensors must be placed on a subject. This has most commonly been done in past literature via an electro-magnetic articulograph (EMA) which monitors the movement of articulatory organs over time as signal feedback ([**?**],[21], [**?**],[**?**]). EMAs require that coil sensors are placed in an individuals mouth and each coil can be detected in an electro-magnetic field to provide a measure of a one dimensional distance. The introduction of more sensors can allow for two and three dimensional representations. With these measurements a researcher can calculate feuatures such as the tongue and lip position. As speech is caused by the movement of articulatory organs, research has often looked at using the trajectories of specific organs as input to be mapped to a representation of audio signal produced over the same time ([**?**],[21]).

Both 3D and 2D input models have been used in the past ([**?**],[**?**]). Earlier methods focussed on deriving functions between an articulatory state and the resulting sound produced ([**?**],[**?**]). These models were used in concordance with a speech synthesizer (eg. [**?**]) based off commonly used theoretical frameworks ([**?**], [**?**],[**?**],[**?**]) , most notably of that due to [**?**] which treats vocal tract as a series of different length pressure tubes. While the theoretical models provide understanding, their performance was naturally poor due to limited number of sounds that could be investigated and little room for variation of common sounds as is found between different speakers.

The simplest way to model the articulatory-acoustic mapping is by using a database of previously recorded articulatory and acoustic signals of a speaker or speakers. With this databse when provided with an input representing an articulatory state, we can match, as close as possible, an articulatory instance in the database to which we look-up the corresponding audio segments for and interpolate. In this way short articulatory state trajectories can be matched with corresponding audio segments and the results can be combined to recreate the full audio signal. This was effectively done in [**?**] with the help of phonetic information (phoneme classes were used to help segregate the database and improve matching ability by first matching the phoneme) and used linear predictive coefficients (these are discussed in detail in Section 1.1, Part III) as their target signal.

The major advantages of the approach above as opposed to other modelling strategies, theoretical or data based, is the simplicity of the model and by matching to actual audio segments, the naturalness of the speech is more likely to be conserved. The method has been shown to have the ability to recreate the audio signal to a good degree of accuracy as can be judged from Figure 1 ([**?**]). However, large databases can be computationally expensive to look through and the method is dependent on a large database being collected with enough variation to capture a speaker's typical articulatory trajectories. It is also heavily dependent on the matching techniques used and does not generalise well.

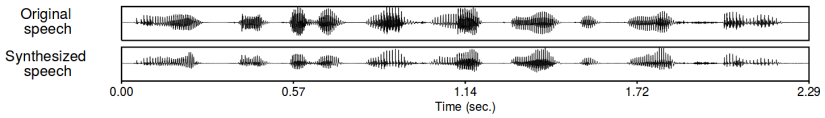


Figure 1: Plot showing a qualitative analysis between original and reconstructed speech using an articulatory-acoustic database. Figure taken from [**?**].

2.2 Statistical Models

Given the compexity of relationship between articulatory parameters and audio, statistical models have been used to model the articulatory-relationship. They are better able to account for the variation in mapping articulatory parameters to audio, as models trying to find an exact relationship (such as the articualtory-acoustic database and theoretical methods) would be unlikely to perform well due to noise and inherent speaker characteristics within both the input and outputs. Furthermore the articulatory-acoustic mapping is known to be many to one [**?**] with multiple visemes producing the same phoneme, justifying statistical and probabilistic approaches found in literature (eg. [21], [**?**]).

2.2.1 Directly Linear Approaches

Linear modelling provides a simple statistical technique and aproaches using linear models have been adopted by [**?**] and [**?**]. Clearly the relationship between articulatory parameters and their corresponding audio is unlikely to be linear in nature. However this assumption may hold *piecewise*. This was done in [**?**] where the problem was reduced to first, clustering the articulatory features and then predicting a spectral features for each of the clusters. The assumption here was that a cluster could represent a typical articulatory state to produce a typical type of sound. While their method did recreate spectral features for each articulatory cluster as best they could, no speech recreations were reported or intelligibilty tests performed so it is hard to gauge the success of the method with only regression errors and no visual reconstructions or intelligibility tests. They did find that, unsurprisingly, linear fits worked better than constant fits for each cluster emphasizing the complexity of the relationship.

2.2.2 Hidden-Markov Models and Gaussain Mixture Models

Following the linear approach, more complex models started to be used in the literature. In the linear approach discussed above clustering was necessarily performed before applying linear formulations. Models which naturally include this type of clustering are Guassian Mixture Models (GMMs) and Hidden Markov Models (HMMs).

GMMs and HMMs have been used to recreate audio in the past ([**?**],[21]). As a

typical case involving the use of a GMMs, we condider the following model based on [**?**]:

$$P(\mathbf{y_t}|\mathbf{x_t}, \lambda) = \sum_{m=1}^{M} P(\mathbf{y_t}|\mathbf{x_t}, \lambda, m) \tag{1}$$

where,

$$P(m|\mathbf{x_t}, \lambda) = \frac{\alpha_m N(x_t|\mu_m, \Sigma_m)}{\sum_{n=1}^{M} \alpha_m N(x_t|\mu_n, \Sigma_n)} \tag{2}$$

$$P(\mathbf{y_t}|\mathbf{x_t}, \lambda, m) = N(\mathbf{y_t}|\mathbf{E_{m,t}}, \mathbf{D_m}) \tag{3}$$

In the above equation, m represents the group, $\mathbf{y_t}$ represents a vector representation of the audio centered at time t (eg. mel-cepstrum coefficients), $\mathbf{x_t}$ represents the vector of input centered at time t (eg. readings from an EMA over an appropriate interval), λ represents the initial prior probabilities for each group α_m.. $\mathbf{E_{m,t}}, \mathbf{D_m}$ are derived from well-known conditional density equations. After the optimum parameters have been found by expectation-maximisation, $E_{m,t}$ from Eq. (3) is used to predict new samples. A qualitative comparison between a targeted and synthesized sample has been shown in Figure 2.

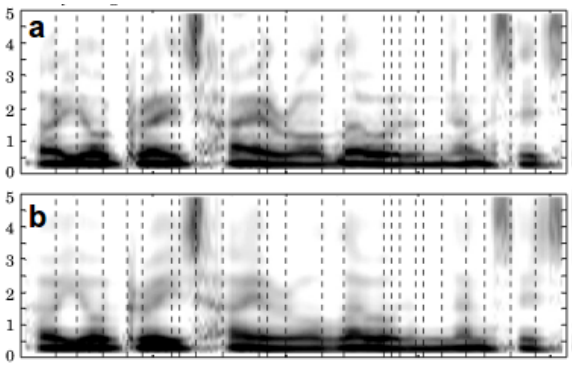


Figure 2: Figure showing a qualitative comparison of a target spectogram (a) and a synthesized spectogram (b). The y-axis represents frequency while the x-axis is a representation of time [**?**]

Figure 2 shows the method has the ability to reconstruct a signal approximately. This method also has the advantage of being phonetic-free implying this model could work well for multiple langauges. However the methods effectiveness relies on tuning the number of groups used in the GMM algorithm and is therefore heavily dependent on the data presented as well as on the subject tested. Furthermore, the assumption that each input vector and output vectors of a group derive from a normal distribution as GMM's inherently assume may not necessarily be justified and may only lead to piecewise approximations of the true underlying mapping, possibly

making it a more innefficient approach.

As mentioned before HMMs have also been studied for the purposes of acoustic production ([**?**], [21]). This has an advantage over GMMs for the sake that they have a natural way of including temporal effects thorugh forward linked hidden variables in their models. A typical model is given by [21] below:

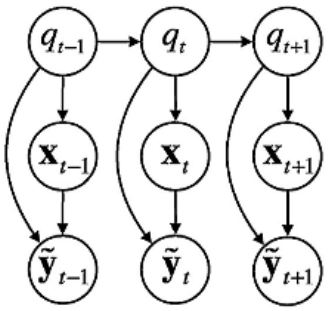


Figure 3: Figure showing the graphical model of a HMM applied to articulatory-acoustic mapping ([21]).

In Figure 3 $\mathbf{x_t}, \mathbf{y_t}$, and q_t are the input vectors, acoustic representation vectors and discrete latent variables centered at time t. The model to be optimised is given below as presented by [21]:

$$p(\mathbf{x}, \mathbf{y}|\Theta) = p(\mathbf{z}|\Theta) = \sum_q P(q|\Theta)P(\mathbf{z}|q, \Theta) \tag{4}$$

with

$$p(q|\Theta) = \pi_q \prod_{t=2}^{T} a_{q_{t-1}q_t}; p(\mathbf{z}|q_t, \Theta) = N(\mu_{q_t}, \Sigma_{q_t}) \tag{5}$$

Effectively from the above, the goal is to find $E[p(\mathbf{y_t}|\mathbf{x_t}, q_t, \Theta)]$ which can be calculated analytically once the optimum parameters Θ have been found though the use of an expectation-maximisation algorithm. This expectation then provides the predicted output given the input signal $\mathbf{x_t}$ and latent variable q_t. In [21], intelligibility tests were carried out on the reuslts of this method by asking participants to identify the vowel or consonant present in the synthesized audio. Accuracy levels of approximately 60 % were recorded. Gueessing the results by chance would lead to 10 % accuracy in their experiment, hence the HMM was able to learn a mapping, proving the concept. However due to the normality assumptions, HMMs suffer from the same downfall as GMMs.

While linear methods,GMMs and HMMs are different approches to the problem of articulatory-acoustic mapping and their derived solutions are different, they ultimately all create linear predictions of sample based on the inputs, as the expecations

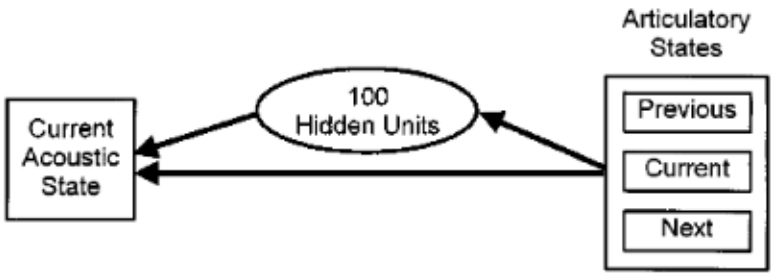


Figure 4: The figure shows the ANN used by [24] with full conectivity. Taken from [24].

of the output of GMMs and HMMs have linear forms, under the assumtions of a normal distribution. This in part has led to investigations into non-linear approaches with the ability to capture more complex relationships. This leads to onto the application of neural netowrks in the next section.

2.3 Use of Neural Networks

The complexity of mapping articulatory features to acoustic features is a complex one and hence it was natural to ask whether neural networks could play a role in the mappings required due to their ability and success in modelling complex functions ([**?**],[**?**]).

To the best of our knowledge there is very little literature involving the use of artificial neural networks (ANNs) for the forward articulatory-acoustic mapping, and most work involved with ANNs has primarily gone to efforts of lip reading in the form of deep learning in which the ouptut is a sequence of text as opposed to an audio signal. ([9],[12],[7],[28],[40],[38]).

One of the earliest uses of neural networks in literature for the purposes of creating audio was provided by [**?**] in which ANNs are used to generate spectral vector distributions from a given trajectory. While this paper was primarily dealing with using articulatory and acoustic information for transcription and not for the purposes of audio reconstruction, it does use ANNs to recreate and audio representation in its process.

Direct work relating to mapping EMA measurements to an audio signal was first provided by [24]. In this study EMA readings over centered 128 m*s* slices were mapped using a neural network (using one hidden layer and direct connection from input to output) to a spectral vector representing the audio over the same time frame. Their model is provided below:

The direct connection from output to input allowed for linear relationships to still be found while the sigmoid activation functions used in the hidden layer allowed non-

linear relationships to be found. The model has the advantage that it does not use any phonetic information and is a direct mapping without a theoretical framework with no need for phonetic labelling or classification. Furthermore full sentence can be reconstructed through this method.

In their experiment, intelligibility tests were conducted using eight participants, who were asked to identify as many words as they could in synthesized sentences from the model. Accuracies of approximately 60 % on test datasets, showing the model does capture articulatory information, though not to a very high degree. On sentences synthesized from the training set, accuracy levels of approximately 80 % were found.

2.4 Deep Learning with Audio-Visual Data

2.4.1 Lip Reading - Video to Audio Mapping

All the models and literature have primarily used electro-magnetic articulogrpaph (EMA) sensors (eg.[24]) with the exception of [21] which, in addition to EMA sensors used ultrasound and lip movement video as input after being dimensionally reduced by principal component analysis (PCA). These are invasive techniques, requiring sensors to be placed inside articulatory organs and placing participants in head stabilisers for accurate readings as seen in [21]. This data is tedious, time-consuming and expensive to collect, given the equipment necessary, as well as payment that may need to be made to the participants. While these methods offer more information than a speaker's mouth movements, given human success at lip reading the question arises as to whether simple video information of a speaker's lips or face could be used to model the corresponding audio. Deep learning has found a large number of applications working with audio-visual data ([9],[12],[7],[28],[40]) and is well suited for extracting features from complex images without needing to resort to simpler methods such as PCA via the use of encoders.

A direct approach to modelling audio from visual data has been provided by the very recent work of Ephrat and Peleg ([15]). To the best of our knowledge, this paper provides the first proof of concept for the work done in this paper, namely creating an audio wave from a silent video of a speaker's face. In their paper a five layered deep convolutional neural network with two fully connected layers were used to map video frames to a representation of the audio sequence which corresponded to the input frames. The video data was split into nine frame batches as input to the convolutional layer network in order to allow for temporal disambiguation and to model brief temporal effects. Linear singular coefficients (LSF) were used as the audio representations due to robustness and ability to be turned back into audio signals. Information is invariably lost when this feature extraction and the audio signal can only ever be recovered to the extent to which the LSF coefficients can be converted back to the original signal. The goal of this paper was to provide a test of concept for the mapping proving it was possible to provide intelligible audio recre-

ations from silent video input. Samples of the audio reconstruction can be heard by following the link http://www.vision.huji.ac.il/vid2speech/ .

Amamzon Mechanical Turk workers (a website to hired help online for the purposes of tedious acitivities such as labelling data) were hired to identify, as best they could, the words spoken in the recounstructed audio allowing for human based classification scores to be obtained. State of the art accuracies of 82.6 % and 79.9 % were found showing the ability of the algorithms to create understandable and meaningful audio reconstructions and is in line with the state of the art intelligibility scores found by the articulatory-acoustic mapping methods described in the previous sections.

They further tested their architecture by assessing how well their models can reacreate audio for words the model has not seen in its training. Their intelligibility results sit around 51.6 % from scores of 93.4 % when tested on sentences using unseen words. It must be noted that this is still siginificantly better than chance at 10 % (they were focussing on the ten uttered digits within each sentence).

While their paper ([15]) primarily focussed on showing a proof of concept, differnent model architecture were not tried, including the natural use of LSTM to model the temporal data. Furthermore there is scope to try mapping to different output features as well as test for generalisability accross different speakers.

2.4.2 Related Work - Video-Text Mapping

While ([15]) appears to be the only paper applying deep learning to synthesize *audio* from silent video, showing the novelty of this line of research, there is a much larger body of work towards the problem of recreating text from silent video ([9],[12],[7],[28],[40],[38]).
These deep learning methods often involve the use of recurrent netowrks to model the temporal effect of the data. In [9] a convolutional network is used in conjuction with gated recurrent units to model visemes and ultimately phonemes leading to sentence reconstruction and [12] uses a deep model primarily relying on LSTM networks (with attention mechanisms) to output sound characters using autoencoded video and audio represetnations. These examples are typical of others in the literature, which generally use mel-frequency cepstrum coefficients (MFCCs) as their audio representation to be input to thier autoencoders ([7],[28],[40],[38]). While these have proved successful (the LipNet algorithm has provided 95.2% on the grid corpus dataset we are working on in this paper [9]) they still ultimately depend on a vocabulary of phonemes or letters which restricts their output representation and still requires knowledge of linguistic features and sentence grammar must be learnt if full sentences are reconstructed.

Ultimately the methods above are closely related to the problem of silent video-audio mapping as they are trying to achieve the same goal: intelligibility of speech result-

ing from a speaker's lip movements. However this goal is achieved by vastly different methods and leads to advantages and disadvantages which we describe below:

KEY ADVANTAGES OF AUDIO SYNTHESIS OVER TEXT PRODUCTION FOR LIP READING

- Is a form of unsupervised learning with 'natural supervision', as a video is simply targeted to it's corresponding synchronised audio track without the need for transcribed sentences reducing the labour involved in collecting transcripts altogether. Due to this a large amount of data becomes easily available.

- Does not need a pre-existing vocabulary or any lingusitic information such as the pronounciation of phonemes or sentence grammar.

- The model is langauage independent as it only looks to reproduce the sounds produced by lip movements as opposed to the recognition of key words.

- A speech signal may contain information about emphasis and emotion which text cannot capture.

KEY DISDVANTAGES OF AUDIO SYNTHESIS OVER TEXT PRODUCTION

- The target function is now a regression problem towards an audio representation and in general more complex and harder to fit than a softmax layer representing letters. This is due to the fact that the maximum of the softmax function is taken as a prediction allowing leeway for non-maximal elements, whereas in regression problems the output vector must be modelled precisely if we do not wish to lose accuracy.

- Audio signals are too complex to fit directly, so dimensionally reduced audio representations of a signal must be used, losing information when the audio is reconstructed.

- There are no easily interpretable classification percentage scores and instead intellibility tests must be carried out by human listeners for objective accuracy.

3 Indirectly Related Work - Unsupervised Learning

While most unsupervised deep learning research has gone towards the classification of static images ([19], [20], [18], [14]), less has been done in the field of video data

given the extra challenges posed including different video time lengths, less obvious representations and capturing dynamics between images as well as their features. Most research using unsupervised learning with temporal data has focussed on mapping the data to targets within itself as in [37]and [41] which maps a set of sequential video frames to the future video frames to find a usefull representation of video data using Long-Short-Term Memory (LSTM) networks. Adversarial networks have also been used for this purpose [**?**].

Our work in this paper work with similar ideas to those presented in the previous paragraph, except that we target audio. Hence our models could in theory also be used to find usefull representations for video for the purposes of lip reading by extracting appropriate hidden layers within our deep framework.

4 Indirectly Related Work - The Inversion Mapping

Work has also been done on the inverted problem, i.e. mapping acoustic to articulatory movements as opposed to articulatory movements to an acousitc signal ([**?**],[**?**],[**?**]) using codebooks to map acoustic sounds to typical articualtory trajectories. Studies like these have helped produce insight into speech production and help to highlight important features of co-articulation such as the many to one articulatory-acoustic mapping with multiple articulatory trajectories ([17]).

5 Discussion

Early work on audio mapping has shown to depend on invasive techniques such as EMAs to obtain information from patients. If speech signals can be reconstructed to good degree of accuracy through the sole use of deep learning networks and video data, very significant savings can be made in terms of time, cost and the intrusiveness of data collection.

The complexity of articulatory-acoustic mapping, of which lip reading is a subset has been shown through the increasingly more complex models which have been applied to the problem, from simple database searches to statistical models and culminating in deep learning approaches. Given the variety of morphology in human speakers with different mouth shapes and mouth dilation during speech, even for the same sounds, very flexible models are required for success. Given the range of possible architectures and their ability to capture non-trivial relationships deep learning models are well suited for this task and represent the latest research endeavors on this problem.

The audio representations used in the literature primarily consist of linear predictive coefficients (LPC), spectograms and mel-frequency cepstral coeficients (MFCC). In

our models we need a way to re-synthesize an audio signal from its respective audio coefficients. Linear predictive coefficients and spectograms provide straightforward means of doing this, however there are no obvious techniques to convert MFCCs back to an audio signal. The use of MFCCs are pervasive in literature, mentioned in the previous section) but are primarily used in audio visual classification and video to text lip reading algorithms to provide usefull features from speech signals as opposed to a reconvertible representation. For this reason our audio representations are limited to that of LPCs and spectograms in this paper.

The advantages and disadvantges of the video to audio lip reading as opposed to video to text has been disceussed and the real power of the deep learning approach to video to audio lip reading is in its unsupervised nature as synchronised audio immediately provides 'natural supervision'. Deep learning model's performance are highly sensitive to amount of training data available and without the need to provide labels, data becomes much simpler and easier to collect with less labour.

A successful video to audio deep learning algorithm would have a large impact on the field of lip reading and provide an important alternative approach. Furthermore, if the work in this paper can be extended to the wild, then data collection would no longer be a limiting factor, with youtube videos presenting an easy source of data.

Part III
Methodology

In this chapter we look at summaries of the key techniques used in our models and provide a background for the diffferent methods implemented. Diagrams will be used where appropriate and the mathematics will be presented only to the extent it adds clarity.

The models and training algorithms used will be discussed, as well as the drawbacks and beenfits of the audio representations used in this paper.

1 Audio Representation

1.1 Linear Predictive Coefficients

Linear predictive coefficients (LPC) provide a compact representation of audio based on a speech production model and encode audio through a small set of coeffients. The original signal can be recreated from the coefficients but not perfectly due to the fact that the speech production model it is based is only an approximation to the truth and assumes the same general model for all speakers.

In order to describe the LPC method the speech model LPC is based on must first be depicted. The following explanation and equations are presented following [**?**].

The speech production model LPC's are based on is provided by Figure 5 below:

In this model speech is looked at as a model of two composites: voiced sounds, sounds such as vowel sounds 'o','i','e' and 'u', and unvoiced sounds such as 'sh' and 's'. The difference between these sounds is that voiced sounds are assumed to be created by glottal pulses (the glottis is a vocal tract organ which can expand and contract and is located in a human's throat near the layrnx) formed from the movement of vocal folds in a human's trachea while unvoiced sounds are assumed to be the result of air being pushed through a person's vocal tract and do not stimulate any vocal folds (effectively white noise). The signal used to produce a sound, whether from glotal pulses or passing air, is known as the *source* and represent the effects of the glottis. The left hand diagrams of **A** in Figure 5 shows an example of these glottal pulses for a voiced sound.

Once a source produces an initial signal (glotal pulses or white noise), this signal then passes through the mouth cavity and is radiated from the speaker's lips. The effect of this is to create a resonance chamber in which certain frequencies are amplified of the source signal are amplified. Exactly which frequencies are amplified depends on the precise shape of a person's mouth cavity. The effect of a person's

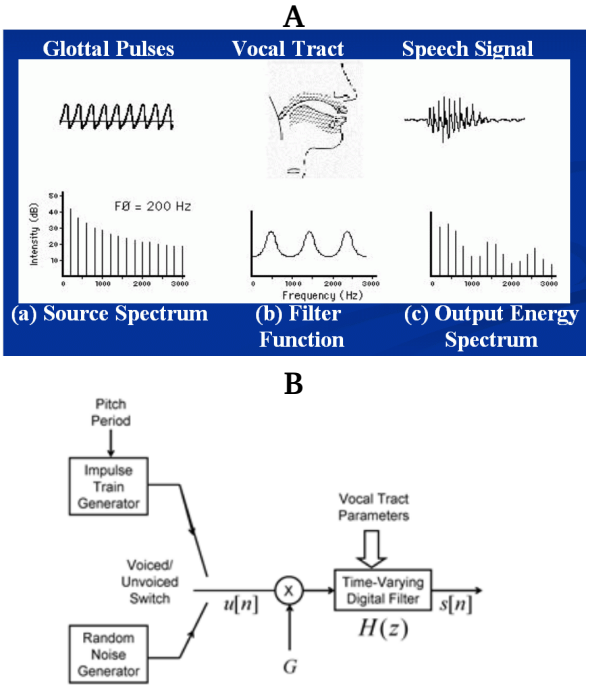


Figure 5: The figure shows a simplified model of speech production, based off an approximation of the human vocal tract ([**?**]).

mouth cavity on the sound that passes through it can be summarized as a filter in the frequency domain as represented by the bottom middle figure of **A** in Figure 5. The filter can be applied by simple multiplication of the source and filter spectrum and the final resulting speech is created.

Unvoiced sounds work by the same process as depicted in **A** of Figure 5 but instead of pulsating signal as a source, white noise is considered as the source, resulting from turbulent airlfow passing through a person's vocal tract.

The production of a signal from a source and its consequent filtering in the mouth cavity fully defines the *source-filter* model LPC are based on and an example of reproduced signal as represented by the upper right hand diagram of **A** in Figure 5.

The information in the preceding paragraphs can be summarized by algorithm represented by **B** in Figure 5. We first decide on a pitch period, a reasonable value which represents a normal frequency of a typical glotal pulse (eg. 150 Hz [**?**]). Once the voiced/unvoiced nature of the sound is determined a 'switch' is then used to change the source appropriately from a glotal pulse to white noise or vice versa depending on whether the sound is voiced or unvoiced. This is then multiplied by G called the gain. This represents the energy of the signal (qualitatively this represents a measure of the amplitude or 'loudness' of the sound produced). There is an inherent assumption here that the energy of the signal is constant throughout the sound frame and that the signal is *stationary* (its mean and variance do not change over time), assumptions which will be discussed later. This signal is then passed through

a filter $H(z)$ to produce the final signal, $s[n]$ representing the signal sample at time n. While this filter is assumed a constant for the signal we are processing, it is in reallity not constant over time as the shape of the mouth continually changes, leading to a change in which frequencies are amplified and consequently changing the shape of the filter spectrum.

The LPC model described by **A** in Figure 5 assumes the filter spectrum shape,power, mean and variance of the signal remains constant, however the shape of the mouth changes over time to produce sounds this is not the case. In order to allow this assumption to hold true (at least approximately), a full audio signal is generally split up into very short, seqential window segments and LPC analysis applied. In order to help avoid discontinuties in inference, overlapping windows are taken with a hamming function applied [15] to each window.

From the decription of the LPC speech model, if we determine the typical pitch of a glotal pulse and the energy or amplitude of the source signal, the only thing that needs to be determined to encode an audio signal is the shape of the filter spectrum at each point in time, representing the effects of the changing mouth cavity. Using the pitch and energy information, glotal pulses or white noise can be synthetically generated and if a filter is available, this can be used to recreate the original signal. In LPC analysis coefficients are derived which effectively capture the shape of the filter spectrum. This is what makes the LPC representation so effective, as the theoretical model encodes most of the information needed to reproduce a signal as prior knowledge within its model and allows for very efficient representations of the signal analysed.

LPC analysis works by assuming each next sample of an audio sequence can be predicted as a linear combination of a finite number of previous samples. This is expressed by the equation below:

$$s[n] = \sum_{i=1}^{p} \alpha_i s[n-i] + Gx[n] \tag{6}$$

In the above equation $s[n]$ is the signal sample at time n, α_i is the i^{th} LPC coefficient, p is the order of the LPC model, G represents the gain (the energy of the frame) and $x[n]$ which represents the effects of the source signal at time n, i.e. which will be periodic if the signal is voiced, due to glotal pulses or not if the signal is unvoiced. In this case $Gx[n]$ represents the error of prediction. Our goal is therefore to find the coefficients a_i and the gain G as $x[n]$ can be synthetically generated. It can be seen that $s[n] = \sum_{i=1}^{p} \alpha_i s[n-i]$ summarises the effects of the mouth cavity filter which, as mentioned before, is precisely what we need to know in order complete the speech production model presented by **B** in Figure 5.

A smoothed version of the signal, provided by a tenth-order LPC model, in the frequency domain has been shown in Figure 6. The main outcome of the smoothed

signal is that the general shape is able to capture the *formants* of the signal, i.e. the peaks of the spectrum in Figure 6. These represent the frequencies the mouth cavity resonates with the original signal. If we were to remove and detrend the smooth red line from Figure 6 we would find a pulsating signal or white noise. Accurate determination of the true formants of the mouth cavity is imperative and if the order too high or too low, the signal will be overfitted/underfitted with smoothed line following the original signal too closely or not at all respectively. This may mean the true formants will not be found, affecting speech quality.

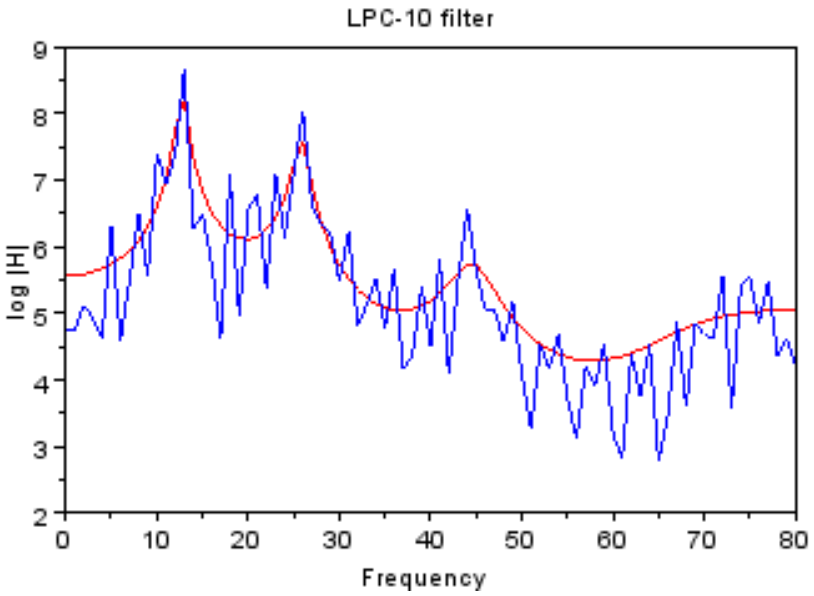


Figure 6: Figure showing an example of a smoothed spectrum. [**?**]

In practice once we have found the coefficients (the procedure is described in the next section), to regenerate speech, glotal pulses or white noise is synthetically generated depending whether the audio frame in question is voiced or unvoiced, relating to $Gx[n]$ and then Equation 6 is applied recursively to reproduce the signal.

1.2 Determining the Coefficients

In order to determine the coefficients, a_k for LPC we look at LPC as an autoregressive procedure. The following equations and descriptions will follow [**?**].

To find the LPC coefficients we minimise the squared error, where N is the total number of samples, P is the order of the LPC analysis, $s[n]$ and $e[n]$ are the signal and error at time n respectively. (We assume that the signals have been zero-padded appropriately).

$$E = \sum_{n=-\infty}^{\infty} e^2[n] = \sum_{n=-\infty}^{\infty} (s[n] - \sum_{k=1}^{p} \alpha_k s[n-i])^2 \tag{7}$$

$$\frac{\partial E}{\partial a_k} = 0 \quad \Rightarrow \quad \sum_{n=\infty}^{\infty} s[n-i]s[n] = \sum_{n=-\infty}^{\infty} \sum_{k=1}^{P} a_k \left[s(n-i)s(n-k)\right] \quad \forall k \tag{8}$$

The above can be rewritten as an autocorrelation sequence $R(i) = \sum_{n=i}^{N-1} [s(n-i)s(n-k)]$ to give:

$$R(i) = \sum_{k=1}^{P} a_k R(i-k) \quad \forall k \tag{9}$$

This can be written in matrix form as:

$$\begin{pmatrix} R(1) & R(2) & ... & R(P) \\ R(2) & R(3) & ... & R(1) \\ ... & ... & ... & ... \\ R(P) & R(P-1) & ... & R(1) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ ... \\ a_P \end{pmatrix} = - \begin{pmatrix} R(1) \\ R(2) \\ ... \\ R(P) \end{pmatrix} \tag{10}$$

The left hand matrix of Equation 10 is a *toeplitz* matrix and can be inverted to find the required coefficients $(a_1, a_2, ..., a_P)$.

In order to find the gain G we could theoretically use:

$$Gx[n] = s[n] - \sum_{i=1}^{p} \alpha_i s[n-i] \tag{11}$$

However exact determination the α_i coefficients would be needed which highly unlikley. As a result the energy matching criterion below is used:

$$G^2 \sum_{n=0}^{N-1+p} x^2[n] = \sum_{n=0}^{N-1+p} e^2[n] \tag{12}$$

The error of LPC analysis at time n is represented by $e[n]$ in the above equation. Note that that as $e[n]$ is required, once the coefficeints from LPC analysis have been found they are used to detrend the original signal by means of inverse filtering producing $e[n]$. In order to find a solution for G voiced and unvoiced information about the signal is required and their properties are used to find an estimate of G via a similar method to the autocorellation scheme above.

1.3 Discussion

The LPC model allows for very low dimensional representations of an audio signal as it only requires us to store information about the filter and energy of a signal to reproduce it. For this reason it was often used to transmit signal information over phone lines where bandwidth was limited ([**?**]).

However, while an efficient representation is produced the simplified model of speech production on which the LPC model is based often means the natural sounding quality of the reproduced sound is reduced and informal listening tests sound 'grainy', possibly also arising from the effect of necessary audio segmenation.

The biggest drawback of this method in our paradigm is the neccessity of voiced/unvoiced information to calculate the gain. This means that we would have to provide labels for our audio data, defeating the major advantage of our video to audio lip reading models as 'naturally supervised' methods if these coefficients were used as a target for our models. However, while the quality of reconstructions is reduced by not having access to voiced/unvoiced information, the signals can still produce intelligible audio when assuming all frames are assumed to be voiced. Informal listening tests show that while the naturalness of the voice is not perfectly replicated, all the words in reconstructed sentences can be made out.

LPC analysis is known to be unstable, changing with small changes to the input, hence in this paper line spectral pairs ([22]) are used as a robust representation of the coefficients derived. From here on throughout this paper our references to LP coeficients implies that we are using them in their line spectral form.

Our implementation of LPC analysis follows that provided by the code of Ephrat and Peleg (`https://github.com/arielephrat/vid2speech`) and follows their preset parameters and architecture decisions (such as pitch of sound and choice of glotal pulse simulator).

2 Short Time Fourier Transform

?? This is one the most commonly used algorithms for audio encoding by means reinterpreting a signal in terms of a basis of sine and cosine functions.

In order to use this method a signal is first split into sequential segments of specific window length w. These segments may overlap by a fraction h (e.g. half overlapping windows) and if the number of samples present in a real signal x is N then this leads to $n = (N/w) * (1/h)$ segments to be analysed.

Each window of samples is then anlysed using the following function to provide the required 2D matrix representing the spectogram $S[n,t]$:

$$S[n,t] = \sum_{k=0}^{N-1} e^{-i2\pi(kn/N)} w(k)x(k+wht) \tag{13}$$

In the above equation n is the frequency index while t is the time index. Note that the resulting values are complex coefficients and the value of $S[n,f]$ encodes the magnitude (provided by the magnitude of the complex number) and phase shift (provided by the angle of the complex vector from the positive real line) of a sine wave of frequency proportional to $(2\pi\frac{kn}{N})$ that can be used in linear combination with other sinusoidal waves represented by the vector $S[:,t]$ to produce the signal presented by the window at time t. The inverse of the STFT process is thus provided by the following where s represents the synthesis window:

$$\hat{x}[l] = \sum^{t} s[l-thw] \sum\nolimits_{n} S[n,t] e^{i2\pi n(l-thw)/N} \tag{14}$$

For real valued signals the complex coefficients provided byt the STFT are found to conjugate symmetrical, implying that only half of the coefficients need to be stored to capture the frequency information. This is known as a 'one sided' spectogram.

In many applications only the magnitude of the STFT representation is taken while the phase information is discarded ([**?**],[**?**],[**?**]). This is done as human perception is found to be less affected by phase and due to the fact that phase information has often been found harder to model ([**?**],[**?**]). In our own experiments, our models did not converge when phase information was inculded. However phase information is necessary to reconstruct the original signal.

In order to overcome the loss of information through the loss of phase, methods have been developed which look to reconstruct the phase information and improve signal quality from signals reconstructed from magnitude only spectra. The most notable of these is a recursive algorithm applied by Griffin and Lim ([**?**]). This algorithm is presented in Figure 7 ([**?**]).

In Figure 7 $\tilde{x_i}(t)$ represents a real signal which we wish to improve at iteration i. A STFT is first taken, which importantly has phase information even if the original signal $\tilde{x_0}(t)$ was taken from a signal recreated from magnitude only spectra. This is because of overlapping frames meaning when each window has its signal reproduced by an inverted STFT operation, its signal is combined with signals produced by neighbouring windows to produce the final signal. In this way a STFT on an inverted phase free spectogram does not produce zero phase.

At each iteration i the spectogram is then scaled to match the magnitude of the original spectrum $S_0(n,m)$ and the resulting spectogram is inverted to form the next

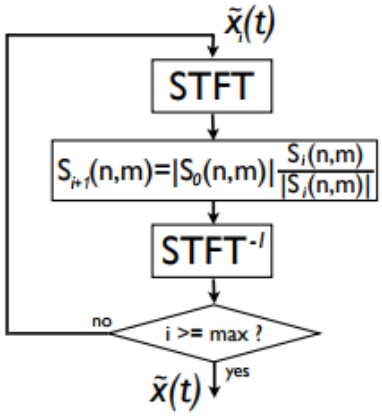


Figure 7: Figure showing an implementation of the typical Griffin and Lim algorithm. Taken from [**?**].

iteration $x_{i+1}\tilde{(t)}$. As the inverted spectorgram includes phase information, phase information is gradually added to the iterated signal and the phase is reconstructed. The proof of this method is beyond the scope of this paper but is provided by [**?**] and while the algorithm converges it will not necessarily converge to original phase.

2.1 Discussion

STFTs provide a method of completely reconstructing the original signal if phase information is captured. This is in stark contrast to the LPC method which can never reconstruct the original signal due its simplified model of speech production. As a result we test our algorithms using STFTs in Part III with the hope of finding better and more natural sounding reconstructions.

However the drawback of STFTs is the number of frequency bins needed to represent each signal window and the number of frequencies sampled is often set to half of or the same as the number of samples present in the window. In our case this was 322 coefficients as opposed to the 18 coefficients we needed under the LPC method. This has consequences for model training as the dimensionality of the targets has significantly increased increasing training times and small reression errors over a large number of coefficients can lead to high aggregated error.

3 Neural Networks

The founding blocks of deep learning methods are provided by neural networks. Our goal in this section is to provide a brief intuition behind the workings of neural networks.

Neural networks are algorithms which mimic the workings of human neurons [6]. The general model can be represented by the picture below in Figure 8 ([**?**]):

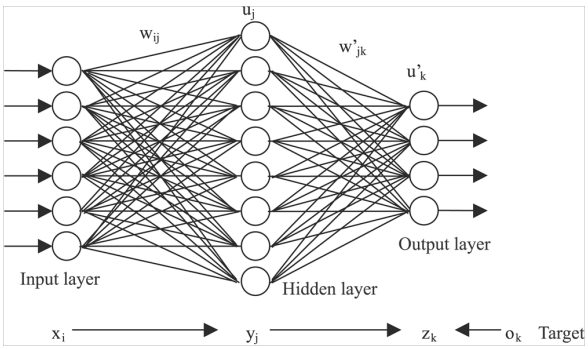


Figure 8: An image of a typical neural network ([**?**]).

Every layer in the model is composed of nodes as seen in Figure 8. Let the i^{th} layer of nodes be represented by the vector of numbers $\mathbf{x}^i$. In Figure **??** the Input, Hidden and Output layer would each be represented by $\mathbf{x}^1, \mathbf{x}^2$ and $\mathbf{x}^3$ respectively. Any layer,k,(other than the first which is equivalent to the input layer) can be represented by:

$$\mathbf{x}^i = act - func(\mathbf{W} * \mathbf{x}^{i-1} + \mathbf{b}) \tag{15}$$

In the above equation $\mathbf{W}$, represents a matrix of weights, one for each connection of the bipartite graph that connects adjacent layers as seen in Figure 8 and $\mathbf{b}$ represents a bias vector. The $act - func$ is a function (such as inverse tan) which is applied to every element of the vector. Thus each node of a layer is a weighted combination of the nodes of it's previous layer, after having a function applied to it.

The final layer of the network seen in Figure 8 is the predicted output of the model while the first layer is the same as the input vector. For regression problems this final layer is left as is and compared with the target output eg. in our case the features exctracted from the audio signal. A softmax layer can also be applied to this layer and the results can also be used for classification [6]. In this paper only regression is used and the classification case will not be discussed further.

Ultimately it is the weights which determine the neural network model and it's output. These weights are found through a method called backpropogation as explained in the next section.

3.1 Backpropogation

Backpropogation is the process in which the parameters of a neural network are fitted and is the most commonly used technique today [**?**]. After random initial

weights are assigned, a neural network takes training inputs $\mathbf{x}_i$ and produces outputs $\mathbf{o}_i$. Given an appropriate loss function L we can characterise the error of the model for N inputs, E as:

$$E = \sum_{i}^{N} L(y_i - o_i) \tag{16}$$

In the above equation y_i is the true target sample that we wish to predict. In backpropogation we aim to find $\delta w_{ij} = \frac{\partial E}{\partial w_{ij}}$ where w_{ij} represents the j^{th} weight in the i^{th} layer. This can be solved analytically. This is done for every weight as well as for any other parameters in the model. Once this is calculated the problem is the treated as a gradient descent problem. In practice, due to computational issues and speed stochastic gradient descent is used with batches [**?**]. A simple algorithm has been provided below for illustration purposes:

SIMPLE BACKPROPOGATION ALGORITHM

```
do:
    for each batch in batched input:
        temp_err = 0
        for each input in batch:
            output = model(input)
            temp_err = temp_err + output
        for each parameter p of model:
            delta(p) = gradient(temp_err w.r.t p)
            p = p - learning_rate*delta(p)
    until:
        stopping criterion reached
return:
    parameters
```

The gradients required in the backpropogation algorithm are calculated recursively. First the model takes in inputs and the activations and final error are calculated. This is known as the 'forward pass'. The final error, as described in Equation 16, depends on the output layer weights and biases so the gradient with respect to these are calculated first. The chain rule is then used to find the gradients for the preceding layer. These gradients are then used to calculate the gradients of its preceding layer and this is then continued until the input layer is reached. For this reason the method is called 'backpropogation'.

The process described above has many adaptations ([**?**]) and the learning rate is discussed further in a later section. There are also more efficient matrix representations of the algorithm described above which will not be discussed here.

3.2 Motivation and Discussion

Neural networks were originally inspired and designed to mimic the working of human synapses, with many signals being recieved and combined at a synapse which then has its end activated, sending its output signal to numerous other synapses [**?**]. Though it is a very simple model as compared to the human nervous system, it has helped produce state of the art results in recent decades and formed the basis for more complex models.

Neural networks, in general, can be be made much deeper with many more layers of varying size being incorporated into the model than shown in Figure 8. Due to this they have the ability match complex target functions which is especially usefull when dealing with audio-visual data where the relationships are often too hard to model statistically or theoretically and have achieved state of the art performance in certain tasks ([9],[12],[11]).

However the optimum weights of the model offer no immediate interpretation as statistical models do and improving the model generally relies on better parameter estimation. Furthermore the models are trained with gradient descent their outcomes are subject to data availabity and finely tuned parameters such as the learning rate to achieve good performance. Due to the number of weights involved with fully connected layers training.

4 Convolutional Networks

Convolutional networks are neural network models which have special hidden layers known as convolutional layers. These act on 3D inputs as opposed to traditional neural networks which act on flattened vectors of inputs. Take an input of dimension W (width) x H (height) x D (depth). A convolution filter is a block of weights, ω of dimensions (A, B, D) where $A \leq W$ and $B \leq H$. This filter is then is then convolved with the input to produce a new layer. Qualitatively this means that to create an element of output, the the filter overlaps a region of the input and the corresponding input and filter elements are multiplied and summed before having an activation function applied. The filter is then shifted according to a pre-chosen stride length over the input to produce the next ouput. This continues until all possible shifts have been accounted for. Note that this means that if the stride is less than dimensions of a window, the inputs will be correlated. Mathematically, an ouput of a convolution filter is given by:

$$x_{ij}^{\ell} = act - func(\sum_{a=0}^{A-1}\sum_{b=0}^{B-1} \omega_{ab} x_{(i+a)(j+b)\cdot}^{\ell-1}) \tag{17}$$

where x_{ij}^{ℓ} is the i^{th} row j^{th} element of layer ℓ and $act - func$ is an activation funciton

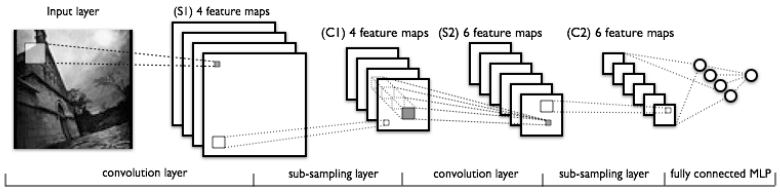


Figure 9: A picture showing a standard convolutional neural network ending with a Multi -Layered Perceptron (MLP) to provide the final output. [**?**]

of the users choice. Note that more than one convolutional filter can be used on the input and if this is done the output will have a depth corresponding to the number of filters used.

Convolutions are typically used hand in hand with max pooling layers ([9],[12],[11]). These are layers in which the input is subsampled to create an output of lower dimension. They take a pxp region of the input and return the maximum element of that region. The pxp regions do not intersect, resulting in a lower dimensional output . If the input has a depth, the process is repeated for each layer of depth so that the depth of the final ouput is conserved while the first 2D dimensions will be reduced. This is primarily done to help reduce the dimension of the inputs, helping to autoencode information.

A picture of a typical convolutional neural network has been provided in Figure 9 [**?**].

4.1 Motivation and Discussion

The primary reasons why convolutional neural networks are used is for their advantages involving sparse interactions, parameter sharing and equivariant representations [**?**]. The nature of these advatages often means convolutional networks are useful in decoding images and they are discussed below in turn.

Sparse interactions refers to the fact that usefull if not the only information necessary can be obtained by comparing regions of the input as opposed to the entire input. An example of this is edge detection (which can be performed with a convolution) in images, which may be enough for classification algorithms. Furthermore it leads to better statistical efficiencly with a lower rank approximation of the input.

Parameter sharing refers to the fact that instead of full connectivity with the input as in typical neural networks, only convolutional filters need to be stored which are much smaller in size, in general. This offers advantages for computational memory and speed as well as a more robust model with the presence of fewer parameters.

Equivariance refers to the fact that if a function f, such as translation is applied to an input, then $convolution(f(input)) = f(convolution(input))$. This means that for

instance, if a pattern in time series data is delayed, the ouptut will still show the same representation for that pattern, just at a later time. This helps to ensure the same patterns are produced for the same objects helping to justify parameter sharing.

Convolutions together with max pooling also help to form translational invariance, that is the ability to detect an objject or feature within an input regardless of where it is within the input [43]. The equivariance described in the previous paragraph helps with this as well as the fact that max pooling 'summarises' the data and lowers the resolution of the output, hence convolutions and max pooling are often used in tandem.

5 Recurrent Networks

Convolutional networks and traditional neural networks have no explicit way of accounting for time series data and if applied to time series will often simply take the depth of the input as time frames or flatten a number of time frames into one vector respectively. In this section we look at networks designed to take account of time and help identify patterns and relationships accross an input or multiple input dimensions.

5.1 Basic Recurrent Neural Networks

A typical Recurrent Neural Network (RNN) is effectively neural network with a memory. This is due to the fact that it uses a cyclical connection with itself, feeding its output back to itself as input. A simple RNN can be seen in Figure 10 [**?**].

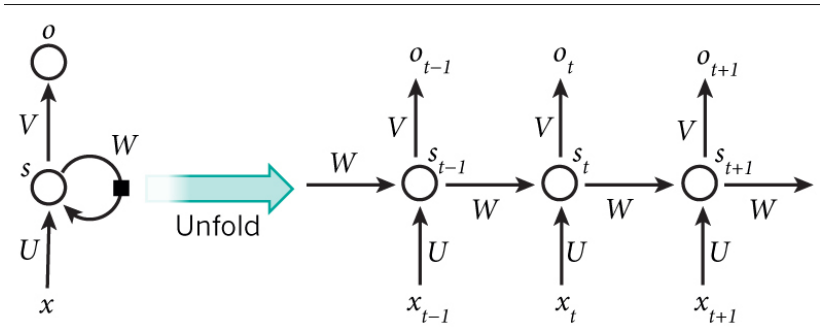


Figure 10: Picture showing a typical RNN structure. [**?**]

Following the description provided by [**?**], in the model described by Figure 10, x_t represents the input at time t and subsequent data points are fed into the model to produce the output o_t. However at each subsequent input is also input with another form of output s_t. The s_t state is defined by the following:

$$s_t = act - func(Ux_t + Ws_{t-1}) \tag{18}$$

where U and W are trainable weight matrices and $act - func$ is pre-determined (generally non-linear) function. The output o_t is then produced by the following:

$$o_t = act - func(Vs_t) \tag{19}$$

The above equations ulitmately define a basic RNN to which there are more variations as described in the follwoing sections.

5.2 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) networks are a variation of recurrent networks designed to address the 'vanishing or exploding gradient problem' [**?**] of simpe RNNs. This will be discussed in more detail in Section 5.4. To explain how LSTM's work, the following equations and descriptions will follow a summary of [**?**] alongside Figure 11, [**?**].

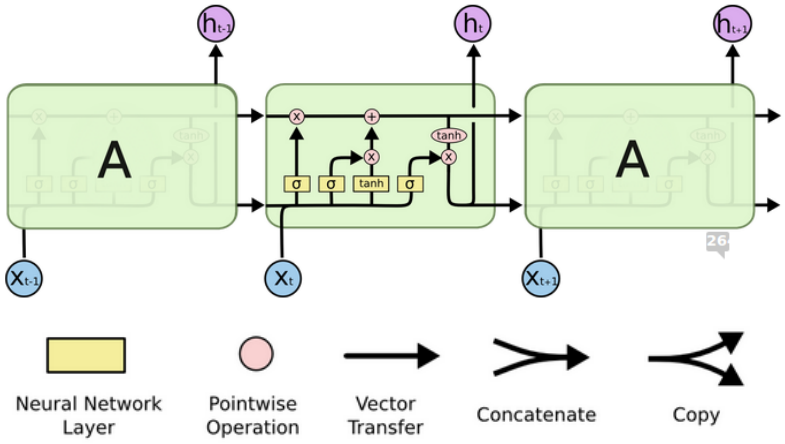


Figure 11: Diagram showing the workings of a LSTM [**?**].

In Figure 11, inputs h_{t-1}, the output of a cell at timestep $t-1$ and x_t, the input data at time $t-1$ of a time series, are fed into a LSTM cell. In the cell we first compute the following:

$$f_t = \sigma(W_f(h_{t-1}, x_t) + b_f) \tag{20}$$

The sigmoid function of the above normalizes the matrix muliplcation and translation of its inputs inside the range of (0,1). This is then multiplied by C_{t-1}, the cell state at the previous timestep. This represents the memory of the cell and could store

information such as the width of the mouth in the last ten timesteps for a lipreading problem. As it is multiplied by f_t, f_t effectively acts as a 'forget gate' as when it is a vector of zeros, no information about the previously held memory C_{t-1} is carried forward and all past information is forgotten, while if f_t is a vector of ones, then all the past information is kept.In this way f_t controls how much and which parts of the previous memory is usefull. The dependence of f_t on the currently known state x_t allows the forget gate to be conditional according to what it is observing a that time. In the case of lip reading, words at the beginning of a sentence may have not as much effect in decoding words in the middle of the sentence as the words immediately preceding it.

To incorporate new information we create a temporary new memory, $\hat{C}_t$ defined below:

$$\hat{C}_t = tanh(W_c(h_{t-1}, x_t + b_c)) \tag{21}$$

$$i_t = \sigma(W_i(h_{t-1}, x_t) + b_i)) \tag{22}$$

The inverse tan function in Equation 21 allows a way for the current information to be stored and normalised between -1 and 1. Furthermore, i_t is then multiplied with $\hat{C}_t$ and hence works in precisely the same way as f_t, acting as a gate to decide which parts of the temporary new memory should be kept and with what strength. The final new memory, C_t is then produced by adding the kept parts of the old and new memory as defined by the following equation:

$$C_t = i_t * C_{t-1} + f_t * \hat{C}_t \tag{23}$$

To produce the final output of the cell the following equation is then used:

$$h_t = \sigma(W_o(h_{t-1}, x_t) + b_o)) \tag{24}$$

In this way we end up with the information we had at the cell's start, namely the output h_t and the cell state C_t, and the process can begin again for the next timestep. The final output of a LSTM is considered to be the sequence of vectors $(h_1, h_2, ..., h_C)$ where C is the number of cells the LSTM is composed of.

LSTMS are the most commonly used form of recurrent networks and have been used extensively for many different purposes as well as lip reading ([9],[12],[7],[28],[40]).

5.2.1 Bi-Directional LSTM

This is a variant of a traditional LSTM which allows for more infromation to be captured by the LSTM and first introduced by [**?**]. It works in precisely the same way an LSTM except that there are effectively two sets of cells, the first set taking the input in the forward timestep direction while the other set takes the input in the *reverse* order. The corresponding outputs are then concatenated to from the final output of the bidirectional LSTM. The cells here work precisely in the same manner as the cells described in Section 5.2. The main reason for this, is to allow future information to be captured at each timestepped output as well as previous information. This is in contrast to normal LSTMs which assumes that outputs depend only on information at previous timesteps.

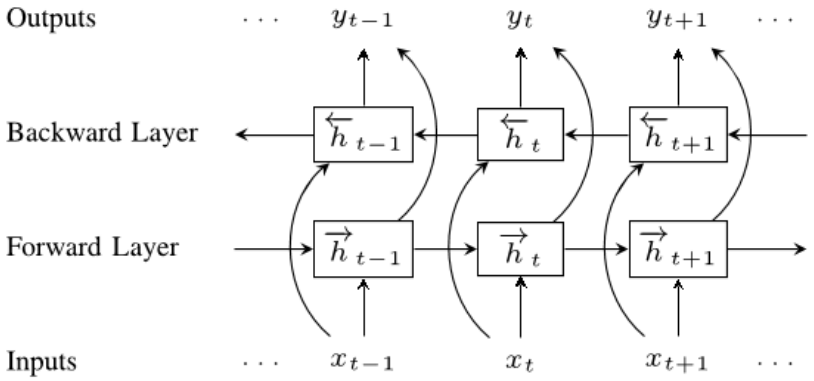


Figure 12: Picture showing typical structure of a Bi-Directional LSTM [**?**]

Figure 12 shows the typical architecture of a bi-directional LSTM where x_t, h_t and y_t represent the input, cell output (left and right arrow hats determine in Figure 12 determine which timeflow, forward or backwards, the cell refers to) and the concatenated output at timestep t respectively.

By allowing the LSTM to have access to future information at each output, more insightful and better predictions can be obtained. This is especially true for lip-reading in which many mouth postions may start with the same movement but end differently. However this architecture comes at the cost of introducing more sets of weights given the number of cells have been doubled to account for the two time directions, making the models harder and more time consuming to train.

5.2.2 Stateful vs Non-Stateful)

The LSTM architecture takes a finite number of timesteps and tries to find a pattern and relation between those timesteps to produce ouput. For example, if a LSTM was given a sequences of time-stepped input such as (2,3,4) with target output 35 and (4,5,6) with target ouput 46 it may be able to discern that the ouput is simply

a concatenation of the first and last timestepped inputs it has been provided with. However what the minimum number of timesteps that is needed to deduce a pattern is greater than the number of timesteps we provide to an LSTM? What if we are looking for very long term patterns? Would an LSTM still be able to provide a solution? The answer depends on whether we use a *stateful* or *non-stateful* LSTM.

In non-stateful LSTMs the first cell has no preceding cell hence it's previous cell inputs h_0, C_0 are considered to be zero or some constant state for every new sequence of time-stepped inputs fed to the LSTM. For the example presented in the previous paragraph, this means the same initial state is used whether we input (2,3,4) or (4,5,6) to the LSTM. No information is passed from one sequence of inputs to another, hence any pattern to be found between timesteps must ultimately be able to be decoded within the sequence provided by any one input. If the number of timesteps is large enough then non-stateful LSTMs may be all that is needed.

In statefull LSTMs, the last output and cell state of the last sequence is passed on to the next sequence. In this way it able to learn from arbitrarily long patterns.

Though statefull LSTMs have the ability to learn longer term patterns, this does not necessarily mean that they will provide better performance than stateless LSTMs due to the fact that the pattern may be fully realized within the the timesteps given to the lstm with every sequence and longer term dependencies may only increase unwanted information.

5.3 Gated Recurrent Units

Gated Recurrent Units (GRUs) are a variation of LSTMs but have important differences. In this model, the input and forget gates are combined into one gate and rather than have an explicit memory state C_t, at timestep t, they incorporate that information into h_t directly. These were introduced by [**?**]. A diagram of this model and it's corresponding equations are shown in Figure 13, [**?**] .

Due to less states being present, GRUs are simpler models than LSTMs and if they are able to fit the complexity of the data required, are more statistically efficient with less parameters. They have also been used in literature to a common degree ([**?**],[**?**],[**?**]).

5.4 Motivation and Discussion

As mentioned RNNs were created to help model temporal effects and work with time series. The s_t connection in the RNNs is what makes them useful for this purpose. Without this connection the RNN would effectively be a sequence of *different* neural network, one for each timestep.The connection is what provides the 'memory' of the system and links the input of past samples to each other through time. In this way

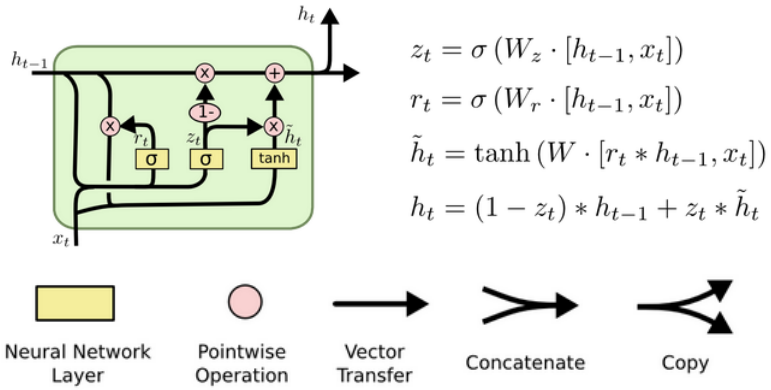


Figure 13: Diagram of typical GRU structure. [**?**]

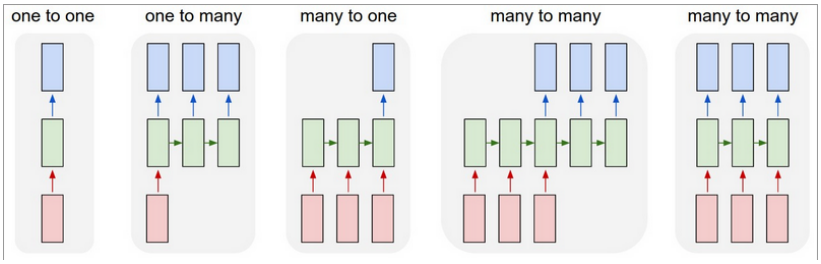


Figure 14: Picture showing possible architectures for a RNN [**?**]

if there is a pattern in a sequence of time stamped, regardless of how long, a RNN could theoretically learn it due to the state information it continually passes on at each input.

Note that there are many variations to the structure presented by Figure 10 as it may not be necessary for all timesteps to be input and the number of inputs or outputs could exceed or be less than the number of inputs. Possible alternative structures are shown in Figure 14 [**?**]

A simple example where a one to many RNN may be needed are for predicting sequences of the type (2,3,4, ..), (5,6,7,..), and (1,2,3,..) i.e. sequences which simply add one to the inital digit. In these cases only the first digit is needed to complete the entire sequence.

Given that the weights, U and W in Equation 18 represent the same weights over time there is a large memory saving compared to stacking neural networks on top of each other to model each time step. Due to this the weights of a RNN are effectively capture a representation of the dynamics of a system.

Training a RNN requires a modified version of the backpropogation algorithm known as back propogation through time (BPTT) [**?**]. However this algorithm is subject to a major drawback in terms of 'vanishing' gradients [**?**]. This is when backpropogated gradients too small as they are transmitted through many layers of a network due

to reasons studied in depth by [**?**]. Briefly speaking this arises due to gradients being multiplied repeatedly by fractions (arising from the derivatives of activation functions), as the chain rule for the gradients is propogated backwards through the layers. With the algorithm presented in Section 3.1, this implies that training does not take place effectively and long term dependencies may never actually be learned, as inputs further away in time, have many more cycled layers between them (refer to the unrolled network presented in Figure **??** for an intuition of this).
LSTM's and GRUs were designed to overcome the problem of 'vanishing' gradients by using a different architecture. The primary reason why they are able to do this is because of the additive function which combine s old memory with new memory and their use of gates. Both of these help to ensure that the derivative form calculated with reference to the cell's memory is less likely to be of an exponentially decaying form as it is with a simple RNN thus getting around the problem.

A natural question arises as to which form of RNN is the best. Studies by [**?**] and [**?**] suggest that no one form is conclusively better and it may ultimately depend on the data at hand.

6 Training Strategies

Netowrks required the tuning of numerous parameters and are often sensetive to the training strategy used in term with regards to the learning rate, batch size, and regularization. These will be disussed below:

6.1 Learning Rate Strategies

Given the large number of parameters present in neural network models, stochastic gradient descent is used to train deep learning models (eg.[9],[12],[11]). The backpropogation algorithm described in Section 3.1 is an example of this method. An important part of this process is determining the step size with which to change the parameters at each step, as too large steps may lead to parameters which miss the true minimum of the objective function, while too small steps increase the time for convergence and are more likely to get stuck in local optima ([**?**]). A few key strategies to this problem will be discussed below briefly.

The most straightforward learrning strategy is provided through stochastic gradient descent with momentum and decay. This follows the update equation:

$$p_t = p_t - \mu \nabla p_t + m\delta p_{t-1} \tag{25}$$

In the above equation p_t represents a general paramter at iteration t of the training

process, μ_t represents the iteration-varying learning learning rate, ∇p_t represents the gradient of the parameter with respect to the loss function, m represents a new momentum parameter and δp_{t-1} represents the most recent change to the parameter p.

The addition of $m\delta p_{t-1}$ in Equation 25 allows a part of previous updates to a parameter to be included in it's next update. In this way if a parameter was changing rapidly (when it is likely to be far away from the true optimum on a continuous surface) the updates will continue changing the parameter to faster extent whereas if previous updates were small (when it is likely to be near a local optimum, hopefully the global optimum) the paramter will change to a lesser degree. The parameter m controls exactly how strong this effect is.

The learning rate m_t in Equation 25 also changes as training progresses according to a decay equation. Different decay equaitons can be used but the one tested in this paper is provided by the following equation:

$$\mu = \mu * (1/(1 + k * t)) \tag{26}$$

The equation above ultimately means that the learning rate slowly reduces over increasing iterations with an appropriate choice of k, the decay rate. As the training iterations proceed we are more likely to be closer to a local optimum (hopefully the global optimum) and would need to reduce the learning rate so we do not 'step over' the local minimum.

Adagrad ([**?**]) is another learning rate adaptation which is an improvement to the typical stochastic gradient descent algorithm. It uses gemoetrical information from previous updates to provide better updates moving forward and has worked well for sparse data ([**?**]). Closely related to this are RMSprop and Adadelta ([**?**] ,[**?**]). These algorithms are based off Adagrad, but ensures that the learning rate does not diminish so rapidly allowing for quicker learning and have both been used successfully in the past ([**?**],[**?**]).

One of the more common algorithms in use today has been provided by Adam ([25]). This works similarly to RMSprop and Adagrad but also uses but also uses a decaying average of past gradients to provide an effect similar to that of momentum as described before with stochastic gradient descent.

To date there is no conclusively better strategy and may often depend on the data at hand ([**?**]). In our paper we test different learning strategies on a subset of our data to decide an optimal strategy to use.

6.1.1 Regularization Techniques

Deep learning methods are subject to overfitting. Overfitting is when a model fits the trainning data well but does not generalise well when tested against new data. In order to prevent this a number of techiniques can be used but only dropout, early stopping and batch normalisaiton are discussed here.

Dropout is the random selection of parameters within a layer to not update during the training process ([**?**]). This helps prevent connections which mirror each other too much and ensure the effective number of parameters during training are reduced, reducing the liklihood of overfitting by creating a thinner version of the model with less complexity. It has proven to be very effective and is used in most deep learning networks to date ([9],[12],[7],[28],[40]).

When training, after each epoch (i.e. a single pass through all of the training data present), we can monitor *validation loss*. This loss represents the loss found when the current model is tested against a set of inputs not seen in the training dataset. Early stopping refers to the monitoring validation loss at each epoch and stopping the training process when there is no significant improvement. In this way the network is not able to fully tune itself to the training data and increases it's liklihood of generalisation.

6.1.2 Batch Normalisation

Batch normalisation is the normalising of output produced by a layer in a deep neural network when a batch of input is processed by the model ([**?**]). This is done as parameter updates in earlier layers implies that later layers must adapt to agree with those changes as the model is dependent on the distribution of the outputs at each layer. As a simplified example suppose that the ouptut batch of vectors of the first layer of a deep network is increased by one and multiplied by three due to a parameter changes. All subsequent layers must now accomodate to this change by changing their weights, slowing down the learning process if they would like to predict the same output. However this problem could be fixed if we simply normalised the first layer, suntracting the mean and dividing by it's standard deviation, that way any shifts or scaling of first layers output would never need to be adjusted for and the distribution would remain the same. This is what batch normalisation helps to avoid.

In addition to normalising the batches of an output layer, batch normalisation offers the flexibility of deciding exactly how we wish to scale and shift the data. Suppose we have normalised a batch of ouptut from a layer, x, (it's mean has been subtracted and the batch has been divided by it's sample standard deviation) then batch normalisation serves the y as defined by the equation below as the input for the next layer:

$$y = \gamma x + \beta \tag{27}$$

In the equation above γ and β represent trainable scaling factors respectively that are trained through backpropogation in precisely the same way as the any other parameter in the network. This means that if the network benefits from leaving the output batch unnormalised it would have the freedom to do so by training the scaling and shift parameter approriately and increases the chances of finding a more optimal model.

Batch normalisation can be used at reapeatedly after any layer and can increase the speed of convergence. It can also act as a regulizer as the values provided for a training example within a batch are no longer deterministic due to different bathc means and variances used for normalisation [**?**]. When testing on a new sample, population estimates from the entire training set are estimated and used.

7 Measures of Evaluation

Comparing reconstructed and original audio signals can be done in a variety of ways and in this paper we discuss the four methods used.

The most straightforward quantitative way of comparing a target to a predicted signal is by looking at the average regression error between the predicted and target coefficients used to represent the audio in question. While this measure provides a means of guaging the performance of one model to another, it only provides a comparitive measure and does not represent anything fundamental the quality of the signal produced. Only the mean squared error loss has been used to train our models in this paper due to it's robustness when compared to other comparison methods presented below.

The correlation coefficient between one signal and another or their spectograms provides a measure of association and its range lies in [-1,1] with 1 representing perfect match, 0 no association and -1 opposite association. Given its finite range, its score is in part, informative of the quality of how well the signal mimics the original. However this is still a sensitive measure especially when applied to raw waveforms which vary rapidly and in which short time displacements affect results significantly. Due to this when applied to raw signals, the correlation between all possible diplacements of one signal with respect to another are taken and the maximum recorded. So long as the predicted signal shares some similarity to the original this provides a more robust measure.

The signal to noise ratio is another measure and is provided by the following equation where **y** is the original signal and **x** is the predicted signal:

$$e_{sigtonoise} = 10\log_{10}\left[\frac{|\mathbf{y}|^2}{|\mathbf{y}-\mathbf{x}|^2}\right] \tag{28}$$

The equation above represents a measure of the true information, the signal, to the error, the noise. Given the dynamic range of this measure, the ratio is scaled to the logarithmic decibel scale. This measure provides a quantitative view of how much of the true signal is captured within the predicted signal, with higher values representing a better quality signal $\mathbf{x}$. However given the range of this measure is the entire real, objective quality is hard discern but can still be compared accross research papers unlike the mean regression error which depends on what features which audio representations were used. Furthermore this is still a sensitive one given it acts on raw waveforms and can be negative due to the presence of high noise.

The final way we have looked at comparing predicted and constructed speech is qualtiatively by examining plots of the signals and thier spectra. While this does not provide an objective measure it helps provide confidence in our results by ensuring the audio recontructions do indeed match their original counterparts.

8 Chapter Discussion

The full details of all the methods used in this paper are extensive and cannot be covered in one chapter. the primary purpose of this chapter has been to provide a brief understanding of the key methods used while explaining our motivations for using them clearly.

The audio representations used in this paper each have there flaws and while work has been done into extracting features from audio data less has been done into the synthesis of speech from dimensionally reduced representations. This stands as a major limitation of our models as even if our models were to predict perfectly, it does not necessarily mean they will reconstruct the original signal perfectly. Future research into this problem is required if these models are to be used in practice.

The deep learning methods described have many variations and the models in general have many parameters which require tuning or must be chosen from the start. Our experiments in this paper thus start with a tried and tested deep learning model provided by [15] and then progress to more complicated architectures.

A major limitation of our methodology lies in our evaluation measures. Ultimately, the true test of the quality of an audio reconstruction can only be provided by intelligibility tests by human observers asked to identify as many words as possible in the reconstructed sentences. In this way clasification scores can be obtained with percentage accuracy providing objective as opposed to comparitive evaluation. However due to limited resources, these evaluations have not been carried out and only

informal listening tests have been conducted.

All the deep learning methods were implemented using *Keras*, based on *Tensorflow*. Their libraries and default settings for their implementations can be found at `https://keras.io/`.

Part IV
Data Preprocessing and Preparation

This chapter provides the details of the dataset that was used and the pre-processing perfomed on the dataset.

1 The GRID Dataset

The GRID dataset used in this paper consists of 34 talkers (18 male, 16 female), labelled as *s1* to *s34*, speaking directly towards a single camera while saying sentences composed of 51 possible words with digits included. For each speaker, there are one thousand sentences. Links to the publicly available dataset have been provided by [**?**]. Examples of a spoken sentences are 'Put red at G9 now' and 'Bin blue at E9 now' and the sentences do not necessarily abide by grammar. Low and high qualtiy videos have been provided by the GRID corpus dataset and in this paper we deal with the latter, with 720 x 576 pixel resolution and three colour channels. Each clip is three seconds long with a frame rate of 25 frames per second. The corresponding high qualtiy audio tracks have a frequency of 50 kHz and are single source signals.

An example of typical speakers and their orientation towards the camera have been provided in Figure 15.

Figure 15: Stilled frames from speakers s2 and s4 GRID corpus videos

While high quality videos are provided by this dataset with speakers speaking directly to a camera with constant unmoving backgraounds, it is limited in its vocabularly and further research on larger vocabulary datasets is not pursued in this paper.

2 Face and Lip Cropping

In order to crop the face of the subject, key markers were such as the corners of the eyes, center of the face, nose tip, inner and outer lips were tracked using certain software (???) throughout the 75 frames present in each video clip. An example of this can be seen in Figure 16.

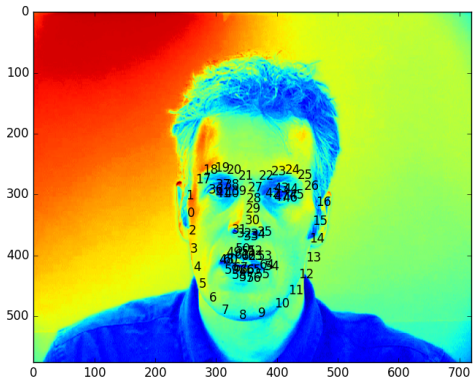


Figure 16: Picture showing the points tracked on a speaker's face

During a speaker's video, head tilt may occur and the speaker may move their head forward or backwards slightly changing the exact position of various facial features within the frames of the video. In order to account for this and ensure consistency accross the frames, the target positions of certain markers (e.g. the sides of the eyes and center of the mouth) were defined and the actual postions of those markers in a frame were then mapped onto the target positions by an affine transformation. This transforms all the marker positions of the original frame as well. Then the face can be cropped from the image by adding/subtracting a margin from the highest lowest values accross each dimension of the transformed markers creating a rectangluar crop which encapsulates the face (This is assuming the face is now non-tilted with an appropriate choice of target positions). By using the same set of target markers accross all the frames, the faces accross all the frames of a video are consequently aligned. The algorithm used is provided below in brief detail:

FACE ALIGNMENT ALGORITHM

```
moi = define which marker positions you wish to align
target_pos = define target position for each marker in moi
stack = empty array
for each frame in video:
    acutal_pos = position of moi in frame
    transform = calculate transformation from actual_pos to target_pos
    trans_face = apply transform to entire frame
```

```
trans_markers = apply transfrom to all markers in frame
face = trans_face[(min_x_trans_markers-MARGIN,max_x_trans_markers+MARGIN), ...
                  (min_y_trans_markers - MARGIN, max_y_trans_markers + MARGIN)]
save face to stack
```

Our margin used was 30 pixels, found by trial and error. Lip cropping is virtually identical to the algorithm above except that we assume we only have the markers for the lips available and ignore the rest. In Ephrat and Peleg's work [15] the face in each frame was cropped but without an affine transformation and their algorithm simply found an upright rectangular bounding box that encompases the face in each. Without the affine transformation, the cropped faces are still subject to inconsistencies due to head tilt changes in depth. A comparison of Epharat and Peleg's algorithm and our algorithm as applied to faces and lips are provided in Figure 17.

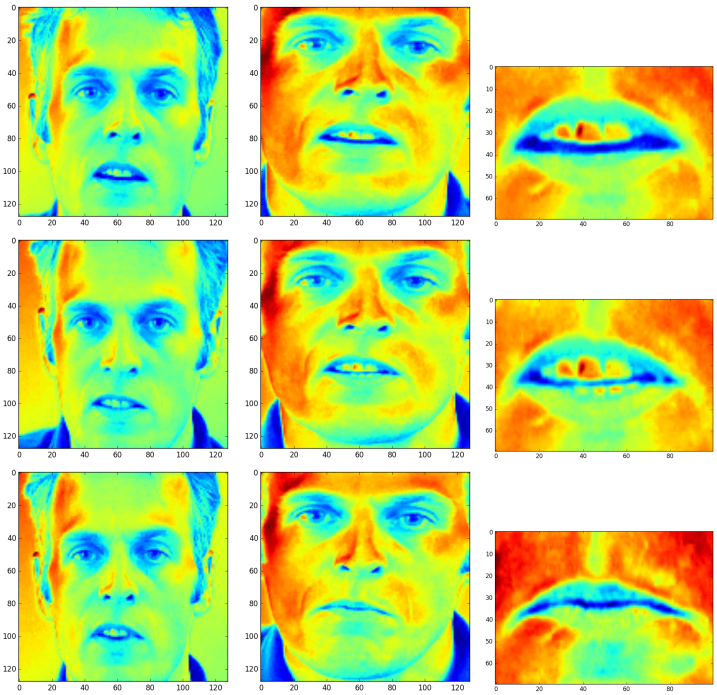


Figure 17: Three randomly sampled images (normalised and greyscaled) showing the results of the original cropping algorithm used in [15], affine transform cropping of face and affine transform cropping of lips from left to right respectively. The positional variation of facial features in left column show the importance of accurate registration.

The presence and absence of the speaker's collar and the different hair outline shapes in the three left images show a presence of slight depth and orientation change. The primary importance of accurate registration is to allow deep learning networks to learn better and faster by having to account for less variation in which nodes are activated for a certain feature.

Cropped faces were sampled to 128 x 128 pixels while cropped lip regions were left at their original dimensionality of 70 x 100. All input images were greyscaled, centered and normalised to the range [0,1] before they were input into our deep

learning models.

3 Audio Pre-processing and AudioVisual Alignment

Each speaker's video has 75 frames over three seconds implying each frame takes place over 0.04 seconds of time. To correspond with this the audio signal was first sampled at 8 kHz form 50 kHz and then broken up into segments each 0.04 seconds in length with a half overlap leading to 150 segments. Each of these segments, consisting 320 samples, was then analysed either via LPC analysis resulting in LPC coefficients or via a one window Fourier transform to provide a representation of the audio. The 8 kHz sampling was done to help reduce dimensionality while still ensuring intelligible speech can be heard. Note that every one frame is then matched with *two* concatenated sets of audio representation due to the fact that the overlap in the audio semgments means two segments represent part of the same information. In reality a single frame, which represents a snapshot in time cannot be expected to provide information about a time varying signal, hence instead of mapping a single frame to an audio representation, multiple frames are taken to the left and right of the singled frame, creating a window of frames as input, to add context over time. A window centered at time t is then matched with audio representation which takes place during the time interval $[t-0.02, t+0.02]$. This leads to our dataset *(X,Y)* and significantly increases the number of input, output pairs, as well as memory requirements (as each frame is now mapped with frames to the left and right) from the original number of videos. In our experiments one thousand video sets were available for each speaker resulting in approximately 75000 input pairs (in reality this is slightly less given window frames at the beginning and end of videos require a margin).

All audio representations were centered and normalised by diving by their standard deviation. The mean and standard deviations were stored and then used to convert the predicted model output back to the correct scale. While this adds variation in terms of the variation in estimating the mean and standard deviation, this was necessary to ensure fast convergence and given that the error can be unbounded in regression problems scaling is essential to ensure the scale of the output targets are within the range of typical model output.

A visual representation of the alignment between a video frame and it's 8^{th} order LPC representation is provided by Figure 18, as taken from Ephrat and Peleg ([15]).

4 Missing Data

There were two main sources of missing data. The first was in the form of absent tracking points to provide the markers shown in Figure 16 and the second was in

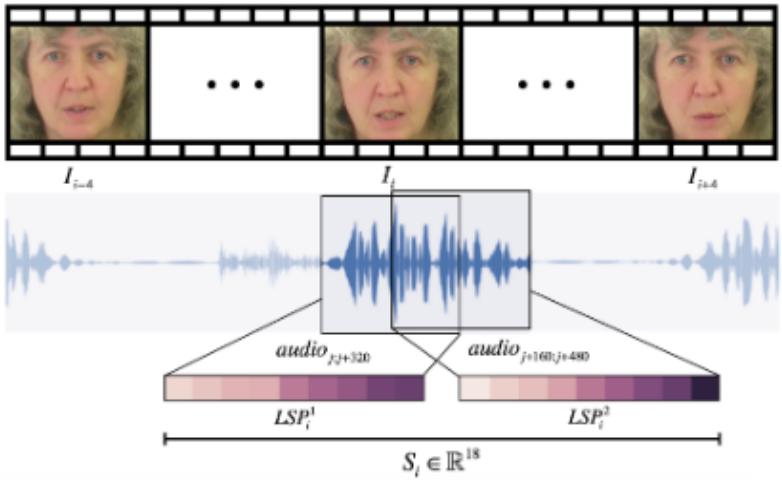


Figure 18: Picture showing the audio-visual alignment of a frame of video and it's corresponding audio representation [15].

missing frames of videos towards the end of a speaker's video.

When tracking points were not available, they were estimated by taking the mean of all the tracking points throughout the entire video sequence. This will only be accurate if a speaker does not have large variation in their facial movement and by using markers of interest such as the center of the mouth and sides of the face (as opposed to lip or chin markers which vary more), accurate registration can still be achieved. An alternative would be to ignore the audio and video data for which we do not have tracking points but this would reduce the information present and lead to more complications if a video sequence was broken, so this option was not taken.

When missing frames were found, they were found at the end of the video sequence and in all cases, when the speaker had finished speaking. Rather than simply remove the audio corresponding to the missing frames and effectively make the video-audio data of shorter length, the last video frame present was repeated for the missing frames present. This allowed the model to reinforce it's learning that a sequence of closed lips produces no sound and mimics the state that would have been seen if all the video frames were actually present.

5 Chapter Discussion

The GRID dataset provides an easy to use database but is limited in vocabulary. Furthermore the speakers are well placed and the videos show little variation in lighting and with speakers facing straight ahead, hence it is unclear and unlikely that our models would perform well if the speakers were observed in the wild but nevertheless, this dataset provides a good starting point for experimentation and was used to good effect by [15].

Part V
Experiments

1 Introduction

In this chapter we first look at finding a well performing model for the problem of video to audio lip reading using deep learning models. Given the large range of possible architectures and model choices we start with implementing the latest state of the art deep learning model applied to this problem presented by Ephrat and Peleg ([15]). This is then fine tuned and tested using different key parameters. Different architectures are tested and our key contribution is the addition of recurrent networks to their architecture in order to provide a better way of modelling temporal information. Our final model is then compared with the original model provided by Ephrat and Peleg ([15]).

We then proceed to test our best model using STFT features, considering with and without phase reconstructions and compare these with our results using LP coefficients. Finally we look at how well our models generalise to multiple subjects.

The experiments were conducted on single Titan GPU with 8GB of memory. On average, smaller models took approximately 5 hours to run while larger models took around 30 hours.

2 LPC Video to Audio Lip Reading Model

In this section we look at creating an optimised deep learning framework for video to audio lip reading using LP coeffficients following the work of Ephrat and Peleg ([15]). Their architecture is fully defined in the Appendix (page 7) and is represented pictorially below:

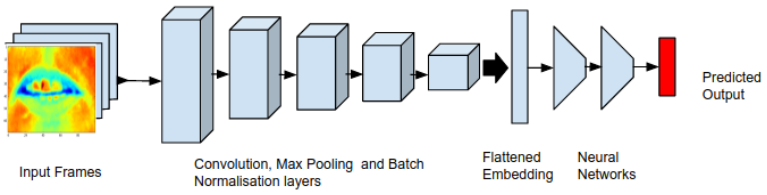


Figure 19: Figure showing the Ephrat and Peleg architecture.

All the models in this section were tested with a with an 80-10-10 % train-val-test split. Mean squared error between the predicted and actual LP coefficients was used

as the loss function and test errors reported refer to this quantity. 8^{th} order LPC analysis was used as per [15] with half overlapping frames. This results in 9 coefficients per audio window (8 linear coefficients and one gain parameter) and an 18 coefficient target vector per input (as the results of overlaping frames are concatenated).

2.1 Parameter Tuning

Here we consider our choice of learning strategy, batch size and dropout rates used in our deep learning models presented throughout the next experiments. While there are many other parameters that could be focussed upon these are the only ones considered here as they represent key factors in controlling time till convergence and the final test loss observed.

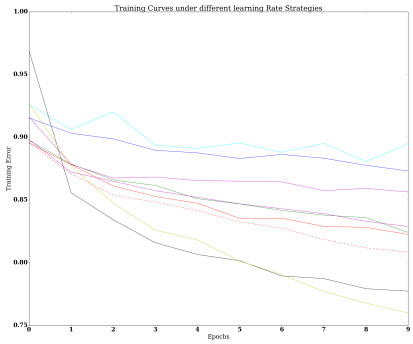
2.1.1 Choice of Learning Strategy

To identify an appropriate learning strategy, a selection of those explained in Section 6.1, Part III (page 37) were chosen and tested on a single speaker, *s2*, of the GRID corpus dataset using a deep learning model.

The deep learning model used, was that of Ephrat and Peleg [15] and is fully defined in the Appendix (page 68). This model takes in 9 frames of greyscaled, normalised video input centered at time t and maps this onto a set of 18 normalised LPC coefficients representing the audio at that time. The loss function used is a mean squared error loss function between the actual and predicted coeffients.

The model was run for ten epochs on ten percent of the dataset and all the models were started from the same set of inital weights to ensure consistency over their starting points. The results are shown in Figure 20. The discrepancies in the y-intercept for the losses arises because they represent the loss *after* one epoch at which point multiple steps have been made accross th optimisation surface and in different directions.

More experiments were conducted with the Adam optimiser in Figure **??** as it had already found success in Ephrat and Peleg's algorithm [15]. Furthermore, all these optimisers were implemented using *Keras* and the inital learning rate as well as any undefined parameters for each of the optimisers, were set around their recommended values or left to their default values respectively, as per the Keras support site (`https://keras.io/optimizers/`). The results in Figure **??** show that RMSprop with an initial learning rate of 0.01 and the Adadelta algorithm with a learning rate of 1 showed the steepest decline in error rate. As RMSprop produced the lowest traning loss after ten epochs and still appears to be decreasing, this is the learning rate strategy we continue with for the rest of our deep learning models.



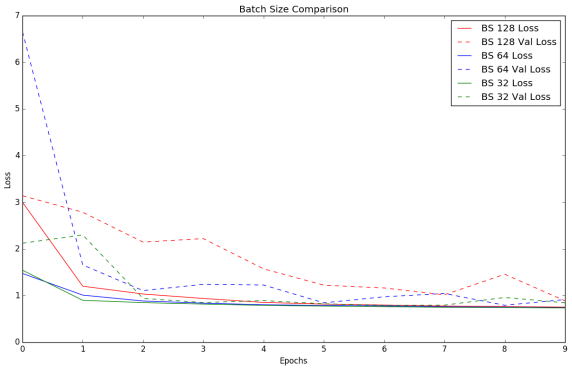
L.R. Strategy	Colour	Final Training Loss
SGD (LR,0.01)	–	0.89
SGD (LR,0.01) (MOM, 0.9)	–	0.92
Adam (LR,0.001)	–	0.84
Adam (LR,0.01)	–	0.84
Adam (LR,0.1)	–	0.87
Adagrad (LR,0.01)	- -	0.85
Adagrad (LR,0.001)	–	0.78
RMSprop (LR,0.001)	–	0.76
Adadelta (LR,1)	- -	0.83

Figure 20: Figure showing the effect of different learning strategies on training loss. LR stands for the intial learning rate and MOM stands for momentum.

We are treating the deep learning model used here as a proxy for all the other deep learning models we will go on to test and while it is possible a different learning strategy would provide better results with different models. Furthermore these experiments have only been conducted with 10 epochs and single speaker's videos. Due to this these results do not provide a conlcusive strategy but rather a heuristic and allow for a more informed decision to be made as opposed to arbitrary selection. RMSprop likely provides better results as it is known to do well with sparse data (only 1000 videos are available for a speaker) and provides an adaptive learning rate, as detailed in Section 6.1, Part III (page 6.1).

2.1.2 Choice of Batch Size

In the same procedure decribed for the learning rate strategy, three sets of batch sizes were also tested using the RMSprop learning strategy found in the previous section. The model was trained on 8% of the data and validated on 2 %. The results are shown in Figure **??**.



Batch Size	Final Val.Loss.
128	1.08
64	1.09
32	1.08

Figure 21: Plot of the effect of different batch sizes.

Figure **??** shows that the final validation loss after 10 epochs are virtually equivalent but that with a batch size of 32, the training error and validation loss are consistently lower throughout the training epochs. This is most likely due to the fact that smaller batch sizes provide noisier estimates of the gradient and are less likely to approach the closest local optima. Furthermore if the solution surface is a complex one with a large number of local optima and curves, then accurate determination of gradients may be more misleading. As a result smaller batch sizes can add regularization as supported the consistenly lower validation curves presented by a batch size of 32 and for this reason, this is the batch size we proceed with in this paper.

As before, these results only provide an approximation to the true optimum batch size due to the small sample size and limited number of epochs.

2.1.3 Dropout

Dropout is used extensively in Ephrat and Peleg's model (page 7 in the Appendix) and there are numerous ways they could be implemented. In all our models we follow a dropout scheme as close as possible to that of Ephrat and Peleg's model due to it's success in their paper. Furthermore, when validation losses are checked during training, none have found to be rising suggesting that the dropout scheme is effective and no more regularization is required.

2.2 Testing Different Inputs

In this section we test Ephrat and Peleg's architecture, with different processed inputs. The first (labeled *Original*), being one using a simple direct face cropping algorithm (as done by Ephrat and Peleg [15]) to create the input frames. The second (labelled *Segmented Face*) and third (labelled *Segmented Face 2*) inputs were created by using an affine transformation before cropping the face (as described in Section 2, Part IV, page 44) but with different target markers. The second input used the sides of the head, center of the head and center of the mouth as target markers to be aligned with their respective mean face positions while the third used the center of the eyes, tip of the nose and mouth center as a target markers. A fourth input (labelled *Segmented Lips*) was also used in which the lips were cropped after using an affine transformations with the target markers being the center and sides of the mouth. A window size of 9 frames was used.

The model was tested with the inputs as descibed above with along an audio representation provided by an 8^th order LPC analysis using *s2* speaker data. The results have been shown in Figure **??**.

In Figure **??** we can see that the inputs which use affine transformations provide more accurate predictions than the Original algorithm which uses no transformations. The test loss is quoted here is the mean squared error between the actual and

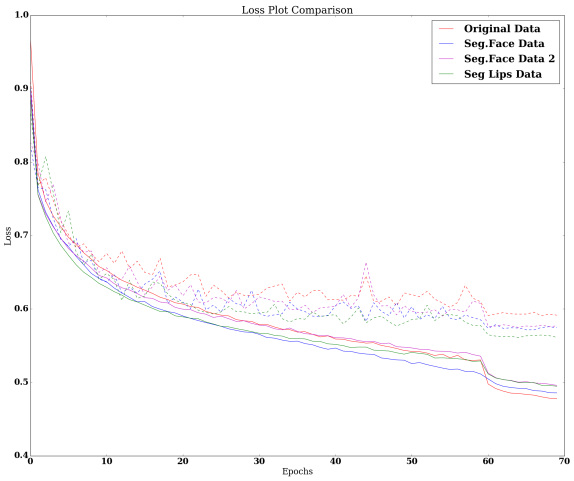


Figure 22: Loss curves for different datasets. Dashed lines represent the validation error while the solid lines, the training error for each epoch during training.

Dataset	Test Loss
Original	0.5905
Segmented Face	0.5714
Segmented Face 2	0.5748
Segmented Lips	0.5610

predicted LPC coefficients. This is in line with expectations as the affine transformations reduce the variability of the inputs and the nodes which activate with specific facial features are more consistent.

We can also see that the lip segmentation provides the best result. As mouth movement is directly correlated with the audio signal, limiting the input to the mouth region reduces the noise and increases the relevant information within the input. Though the result is in line with intuition, Ephrat and Peleg [15] informally reported finding better results with the full face as input. This may have been due to poor cropping of the mouth region, without any affine registration. Furthermore, the restriction to mouth movement also allows for quicker learning as seen by the fact that the Segmented Lips curve is consistently lower than than the others in Figure **??**. Due to this we continue testing with lip segemnted data from here on.

2.3 Different Window Sizes

In this section we test the effect of different window lengths, i.e. the number of frames taken before and after the frame of interest at time t. Note that as the Ephrat and Peleg model is a convolutional model with no recurrent network, the addition of surrounding frames to the central frame corresponding time t is the only way temporal information is added to the model. This allows the pattern of mouth movement over time to be introduced into the model. If too large a window is taken there may be frames of video that no longer correspond to the audio segment it is paired with but if too small a window size is taken, then there may not be enough frames to correlate a pattern with a specific sound.

Window sizes of 5,7,9,11 were tested with the Ephrat and Peleg architecture on *s2* speaker data. The results have been plotted in Figure 34 using both the *Segmented Face* input and *Segmented Lip* input. Though the *Segmented Lip* dataset was found to produce greater accuracy in the previous section, the *Segmented Face* dataset is still tested to see if facial expressions over a different number of frames, expressing em-

phasis or emotion, could lead to better accuracy. The test errors have been provided below:

DIFFERENT WINDOW SIZES

SEGMENTED FACE DATA		SEGMENTED LIPS DATA	
Number of Frames	**Test Loss**	**Number of Frames**	**Test Loss**
5	0.5709	5	0.5667
7	0.5764	7	**0.5597**
9	0.5714	9	0.5610
11	**0.5675**	11	0.5600

The information above shows Segmented Lip Data does consistently produce better Test Loss strongly suggesting features outside of the mouth region have little or no impact on accuracy. While for face data the number of channels increases perfromance, for lip data this is not necessarily true with 7 frame channels giving rise to the best results. However the difference between 7 and 11 frames is marginal and offer no significant improvement when reconstructions are listened to. As the 11 frame model captures more information, we continue using a window length of 11 in our subseqeuent models.

It is possible that using a larger window length could result in a better prediction however due to the limited memory on our machine, this was not possible.

2.4 Different Architectures

In this section we explore the effect of deep learning architectures different to that of Ephrat and Peleg's model. We do this in two ways: the first, by keeping the convolutional structure but simply adding more or less convolutional layers to create different sized models. Second by the addition of recurrent networks to the model allowing temporal information to be accounted for explicilty.

2.4.1 Convolutional Deep Learning Networks of Different Depths

We try changing the depth of the Ephrat and Peleg architecture to create four different models labelled *Smaller*, *Medium*, *Ephrat and Peleg* and *Larger Architecture*. All these architectures are defined in full in the Appendix (pages 68 - 69). The results have been presented below:

DIFFERENT SIZED CONVOLUTION MODELS

Model	**Test Loss**
Smaller Architecture	0.5664
Medium Architecture	0.5630
Ephrat and Peleg Architecture	0.5667
Larger Architecture	**0.5629**

The Medium and Larger architecture appear to perform the best but only slightly. The best perfromance is provide by the Larger Architecture which suggests the complexity of the mapping between audio and video is not fully captured by smaller models. However unexpectedly the Original Architecture performs relatively poorly despite being larger than the Medium Architecture. This may be due to an artefact of the learning process or something inherent and due to this we continue with the Larger Architecture as it is shown to have the ability to provide equivalent or better accuracy than all the other models which are effectively its subsets. Furthermore it contains more free parameters, thus allowing more complex relationships to be learned.

2.4.2 Recurrent Network Layers

We experiment with the use of recurrent neural networks. To do this we took the Larger Architecture as described in the previous section and added a recurrent layer to the architecture. The typical model used is defined by the Appendix (page 69) and is pictorialy represented by the Figure 23 below. Due to the size of model, pretraining was also carried out to improve the model performance.

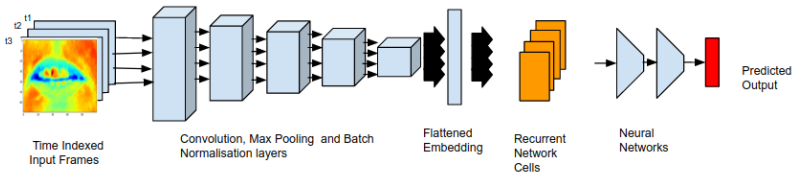


Figure 23: Figure showing the typical recurrent architecture used. The multiple arrows highlight the fact that each frame is now processed individually by the convoltional layers and the multiple time stepped representations are then fed into the recurrent network.

RECURRENT NETWORKS

Model	**Test Loss**
LSTM Stateless	0.5490
Bidirectional LSTM	**0.5401**
LSTM Stateful	0.5930
Gated Recurrent Nnets	0.5730

The results above show that a Bi-LSTM helps produce the best results. As each cell output of a Bi-LSTM has access to both past and future information, better predictions can be learnt. Different sounds may be produced by the same inital mouth movements and the access to future information at each output cell may help in disambiguation leading to it's superior performance.

Given the ability of recurrent networks to take in temporal information directly the LSTM stateless model was retested except this time, all 75 frames of a video were treated as a single input and mapped to a 18*75 vector representing the LP coefficients for an entire video. While the training error reduces during training, its final test score sits aroun 0.8540, significantly worse than the other recurrent net models presented. The poor performance highlights the usefulness of the convolutional encoder inherently present in Figure 23 in extracting features and benefits of segmenting the data to provide more inputs and better learning.

3 Analysis of Final LPC Model

Our final model consists of a Bi-LSTM using 11 frame window inputs of lip-segmented data targeted to 18 LP coefficient vectors supplied by 8^th order LPC analysis. The results of this model are presented in this section. We compare the results of our model (labelled *BiLSTM*) to those created by Ephrat and Peleg's model (labelled as *EP*).

The mean square error of predeiction for the *EP* and *BiLSTM* model were found to be 0.5905 and 0.5401 respectively. We show this difference qualitatively through Figure 24. We can see that there is significant improvement in prediction of the coefficients with that of the *BiLSTM* model over the *EP*. The predictions of *BiLSTM* follow the true coeffients closely while those of *EP* correlate with the true coeffients but do not predict them accurately.

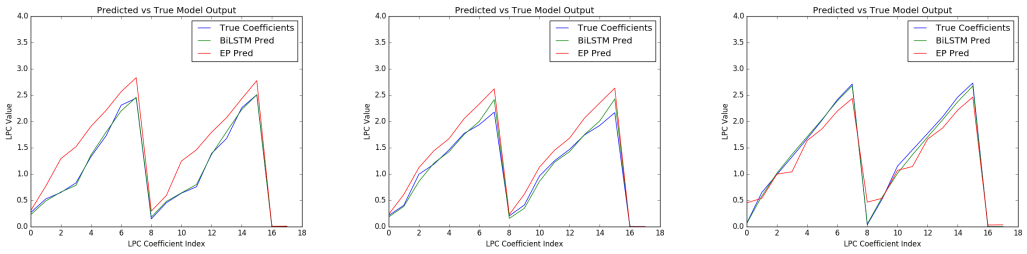


Figure 24: Figure showing three randomly sampled predicted ouputs as compared to the true output for both the Ephrat and Peleg and Bi-LSTM models.

While *BiLSTM* creates much more accurate predictions of the coefficients, it is ultimately thethe signal we are interested in predicting and increase accuracy in coefficient representation does not necessarily translate to a proportional in accuracy of signal production. With this in mind the synthesized waveforms produced from the coefficients have been provided by the Figure 25.
Figure 25 shows the reconstruction results for four randomly chosen sentences. The 'Target Signal' presented in the signal plots is the audio reconstruction that would be observed if the LP coefficients were modelled perfectly. They have been shown to shown to help seperate the effects of the quality lost due to the LPC analysis and the

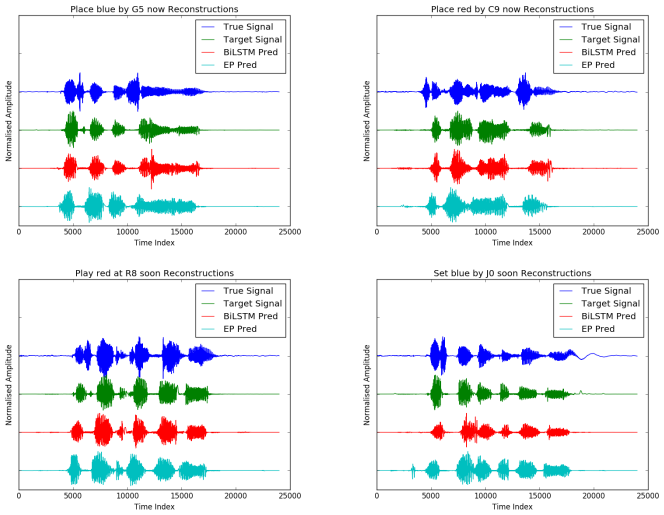


Figure 25: Audio reconstructions for four randomly sampled videos produced by the *EP* and *BiLSTM* models. They are compared with their true and target signals. The target signal is the signal that would be generated if the LP coefficients were matched perfectly.

quality lost due to the imperfect prediction of the LP coefficients. We can see that in general the results are significantly more accurate for the *BiLSTM* model than the *EP* model, with the *BiLSTM* predictions sharing more similarity to their respective target signals, while the *EP* can 'halucinate' clumps of signal as shown by the 'Set blue by J0 soon' plot.

The magnitude spectograms presented in Figure 26 show that both the *BiLSTM* and *EP* plots lose the fine structure of the true spectograms and effectively 'smooth' out the spectral information. This is primarily due to the LPC representation which is generally unable to capture all the harmonics and resonant frequencies present in real speech due to its simplied model of speech production. However we can that *EP* spectogram reconstructions are slightly 'grainier' representing the higher innacuracy. An interesting feature to note is that 'Target Signal' reconstructions show that the LPC methods loses information on higher frequencies as shown by the missing higher frequncy index red bands present in the true signal spectograms and remains a limiting factor for our reconstructions.

The superiority of the *BiLSTM* to that of the *EP* model is confirmed quantitatively by Table 1, in which the true signals were evaluated against the predictions produced by the models over the test set and then averaged. All measures indicate better performance from the *BiLSTM* model. Note that the Signal to Noise ratio (SNR) is negative, indicating a high presence of noise in the predicted signals.

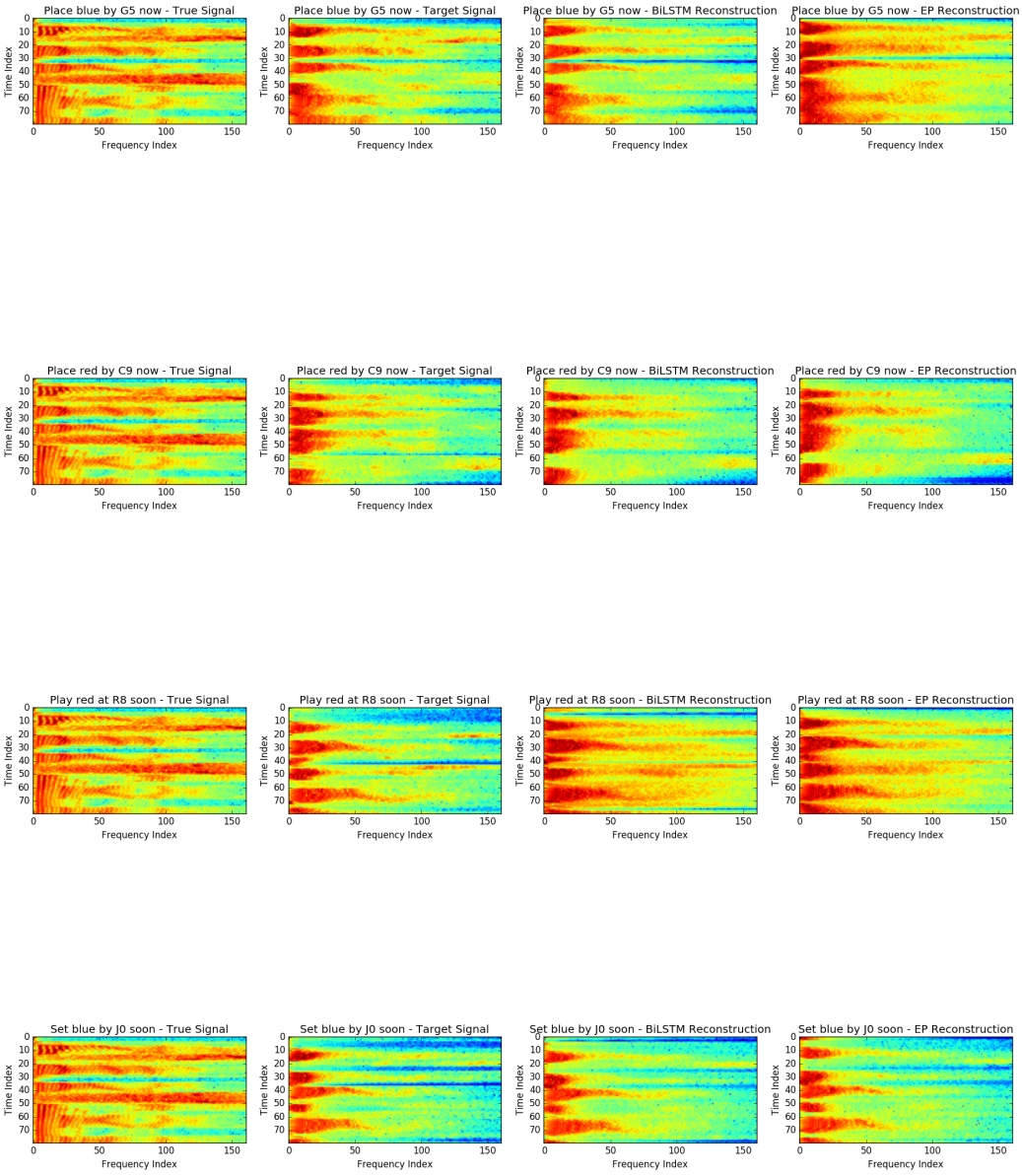


Figure 26: Magnitude spectograms for four randomly sampled waveforms produced by the *EP* and *BiLSTM* models. They compared with their true and target spectograms. The spectograms have been plotted on the log scale for clarity.

Sig. Comparison	**Sig. Correlation**	**Spec. Correlation**	**SNR** (dB)
Target Signal/True Signal	0.011	0.580	-2.74
BiLSTM/True Signal	0.012	0.492	-2.64
BiLSTM/Target Signal	0.190	0.754	-1.98
EP/True Signal	0.005	0.096	-3.49
EP/Target Signal	0.010	0.153	-3.75

Table 1: Evaluation Measures on LPC Reconstructions

The better performance of the *BiLSTM* model over the *EP* model, is not just due to the presence of a recurrent network but a culmination of improvements from lip segmentation, choice of frame size and convolutional architecture. Comparisons of the target signal to the true signal and that of the predictions to their target signals also prove informative. The close comparison between the 'Target Signal/True Signal' and '*BiLSTM*/Target Signal' suggest that the *BiLSTM* model is producing LP coeffients on par with actual LP coefficents and furthermore highlights that the limiting factor to this models progress is the infromation lost in the audio representation as opposed to predictive capability.

When informal listening tests were conducted, the reconstructions from the *BiLSTM* model sounds 'crackly' and jarred though most words could still be identified. The loss of a natural sound is primarily due to the loss of information when encoding using LPC coefficients but al so due to the imperfect match between between the predicted and true LPC coefficients. While mean-squared errors allow for a relative quantitative comparison, while signal plots and spectograms lead to a more qualtivative one, intelligibility tests are required by human listeners to ultimately decide the usefulness of the reconstructions. Due to limited resources, these have not been carried out here and limits our ability to compare the effectiveness of our reconstructions to those presented in previous research.

4 Changing the Target Features

The LPC model is simplified model of the human vocal system and as a result does not produce perfect reconstructions of the original signals it encodes. Furthermore it tends to produce unnatural sounding reconstructions. For this sake, we look to spectograms as an alternative set of target features.

4.1 Comment on changing the order of LPC plots

One parameter that can be changed in the LPC model is the order of the LPC analysis. In our experiments we use 8^th order LPC analysis following the work of Ephrat and Peleg [15] however a higher or slightly lower order cold affect the quality of the speech through more accureate determination of formants. To test this qualtiatively, LPC analysis of different orders was applied to a spoken sentences and the audio

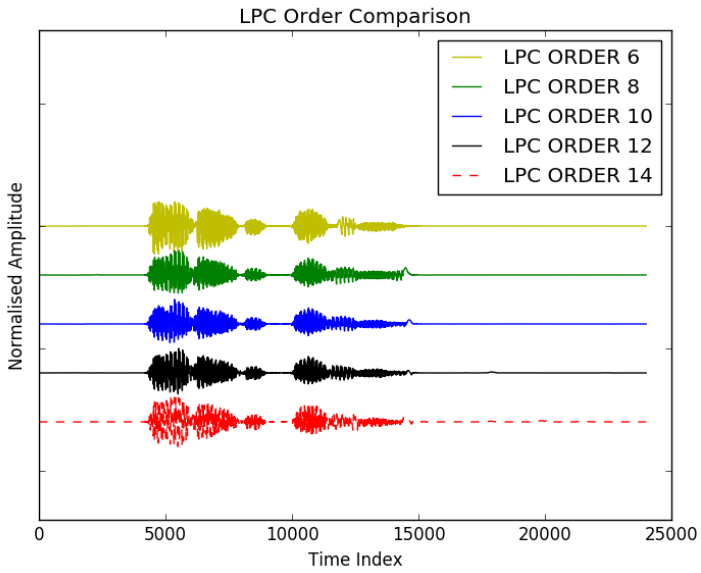


Figure 27: Figure showing the audio for reconstuctions for the spoken sentence 'bin blue at f1 now' after LPC analysis of different orders have been applied.

signal recontructed. Figure 27 show the result for one of these sentences. This result is typical of other sentences and ultimately changing the order of the LPC analysis did not have any significant effect on the reconstruction results and does not improve the quality of the sound from the 8^th order representation, as observed when informal listening tests were conducted. However higher orders were observed to degrade the signal possibly due to overfitting as explained in Section 1.1, Part III (page 1.1).

For the reasons above the effect of order of the LPC analysis has not been studied, as there is no reason to believe the final reconstruction results would improve even if the coefficients were perfectly predicted.

4.2 STFTS

To use the results of a STFT as our target features, the 8 kHz audio signals were split into windows of 320 samples (0.04 seconds) , with half overlap as done for the LPC method and a one sided fourier transform was applied, to find corresponding spectral information for each window. This led to 161 fourier coefficients being produced, implying a 322 length target vector (as overlapping spectra are concatenated).

To test the effect of using a STFT representation, the *BiLSTM* model was run under its optimal settings from the previous section with a slight modification to the final layer of the model to allow it to target 322 length vectors instead of the previous 18 length vectors under the LPC method. This was done by simply adding more nodes

to final neural network layer present in the *BiLSTM* model.

Only the magitude of the STFT features have been used as targets for our deep learning model. Our attempts to model the phase proved unsuccessful and our algorithms did not converge. Modelling the phase has been noted to be a harder task in literature ([**?**]). As a result phase information was not modelled but the true phases were still kept for the purposes of signal reproduction. The results of our experiments have been provided by Table 2, Figure 28 and Figure 29.

Predicted/True Sig. Comparison	**Sig. Correlation**	**Spec. Correlation**	**SNR**
BiLSTM without phase reconstruction (magnitude only)	-0.02	0.765	-1.612
BiLSTM with phase reconstruction	0.061	0.682	-1.602
BiLSTM with orignal phase	0.783	*	3.901
Test Loss - 0.549			

Table 2: Evaluation Measures on STFT Reconstructions

*These are not reported as the phase addition, true or reconstructed, does not attempt to change the magnitude of the spectograms and would provide repeated information.

The reconstructions provided by both models for four randomly selected sentences have been shown below:

A sample of predicted waveforms has been provided by Figure 28. The plots in Figure 28 that the method can produce reasonable approximations to the true signal. When no phase information is taken to account, the signal produced from an inverse STFT operation on the magnitude STFT predicted is sparse and segregated as seen by the '*BiLSTM* - No Phase' plots. While the audio produced does follow the undulations of the actual sentence spoken with some words almost discernable, most of the reconstructed waveform is unintelligible and jarring to listen to. This shows the importance and non-negligibility of phase in intelligibility.

Following the above, the signal is seen to be greatly improved by the addition of phase presented by the '*BiLSTM* - True Phase' and '*BiLSTM* - Reconstructed Phase' plots in Figure 28. The former plot added back the original phase component of the predicted magnitude spectrum while the latter attempts to reconstruct the phase information through the use of the Griffin and Lim algorithm ([**?**] using 10,000 iterations, see Section **??**, Part III). Though the original phase information is generally not available to us when predicting on new inputs, its addition highlights the effect of phase on the signal and provides a comparison for our phase reconstructed signal. We can see that the signals created with reconstructed phase, mirror the shape of the signal produced with the true phase, providing an important step to producing an intelligible waveform. However this reconstruction comes at the cost of more noise due to the increased presence of non-continuous 'spikes' within the '*BiLSTM* -

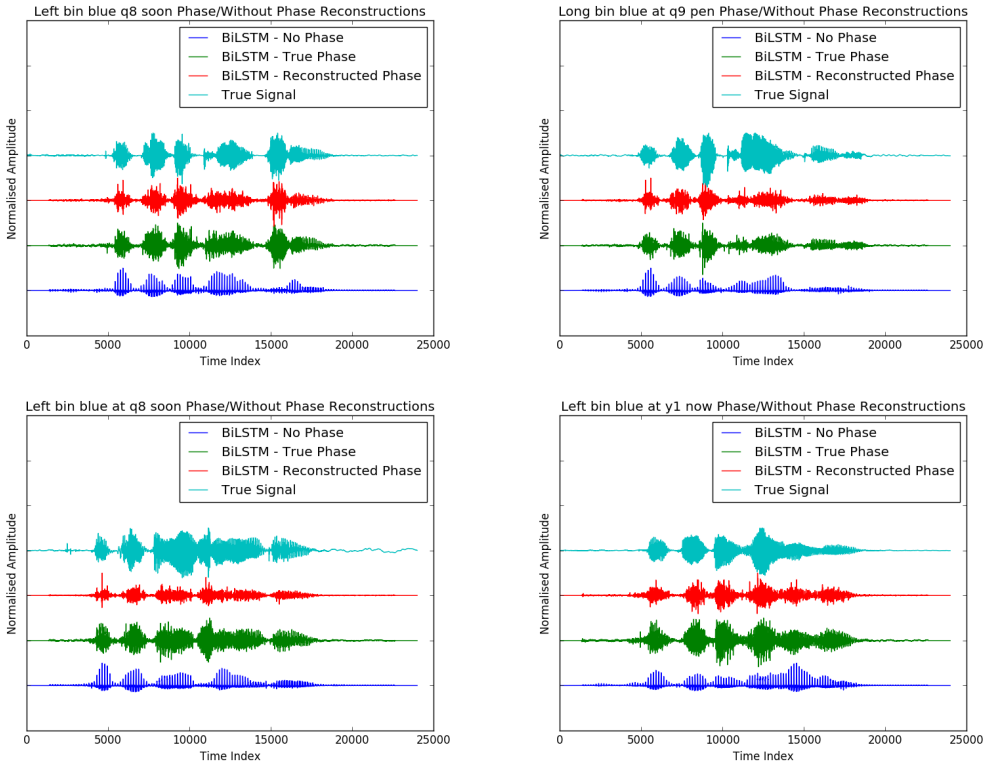


Figure 28: Signal plots of four randomly reconsrutructed sentences for both the BiLSTM model and Ephrat and Peleg model.

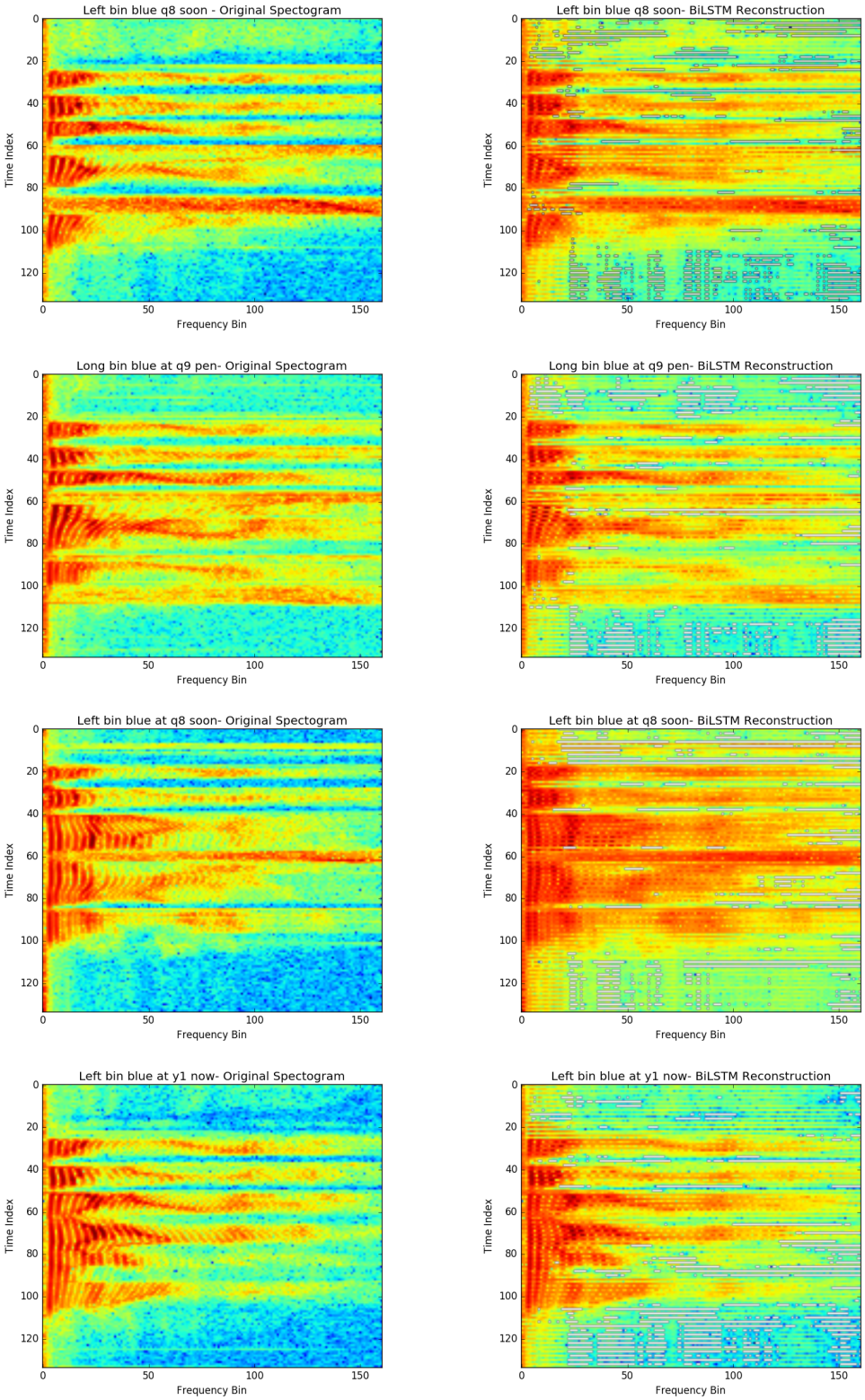


Figure 29: Spectogram plots of four randomly selected sentences comparing the original spectogram with that of the predicted spectogram produced by the BiLSTM. The white spaces in the BiLSTM reconstructed spectograms represent `nan` values produced when the logarithm of values too close to zero were taken.

Reconstructed Phase' signals which reduce the quality of the sound listened to.

Figure 29 shows a sample of the predicted magnitude spectra. We can observe that the predicted spectra follow that of the original spectrum closely and though some fine detail is lost, impressively the predicted spectra manage, in part, to capture the harmonics of natural speech presented by the vertical formant striations in the original spectorgrams. This is important for developing natural sounding speech however this method is still limited the requirement of phase reconstruction.

The evaluation measures presented in Table 2 provide a quantitative view of the results. The evaluation measures were averaged over the results onthe test data set. The test loss, representing the mean squared error between the predicted and true STFT coefficients, sits at an average 0.549 similar to the test loss presented under the LPC algorithms however while the increased number of coeffients could mean more aggrgated error in the final reconstructions, the reproduced signals still provide reasonable approximations. A high spectogram correlation is present in line with the results produced by Figure 29. The highest signal correlation is found with true phase reconstructions while the poor correlation of the reconstructed phase coefficients is due to the extra noise produced by the phase reconstruction algorithm though the algorithm. When listened to we found there is a star difference in the quality of the true and reconstructed phase signals with the former producing perfectly intelligible speech while the latter sounds very noisy though some words are still able to be heard. While the no phase and recontructed phase measures share similar values, zero phase recontructions do not produce any intelligble speech whereas the reconstructed phase signals can.

Accurate phase reconstruction is the major limitation of this the STFT procedure and while we have employed a simple phase reconstruction technique, more accurate and advance modelling of the phase is required if perfectly normal sounding speech is to be found.

5 Discussion and Comparison of LPC and STFT as target features

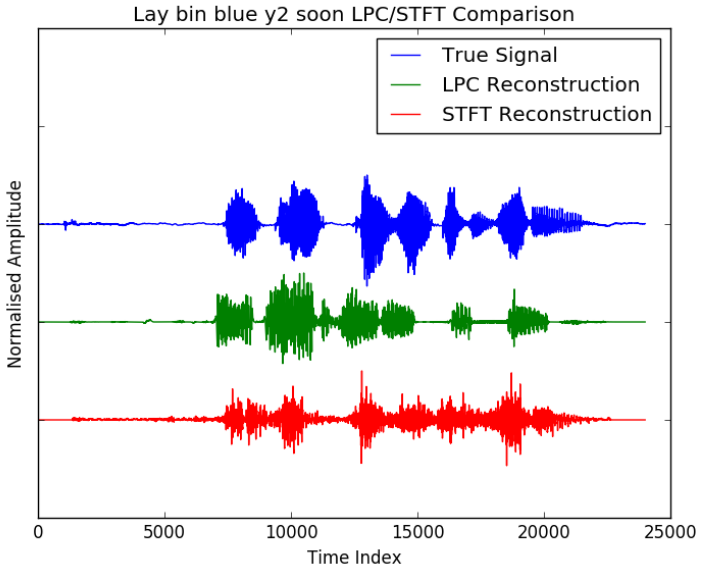
The reconstructions provided by the STFT and LPC implentations of the *BiLSTM* architecture provide two alternative ways to produce an audio signal. We qualitatively compare the differences here and note the limitations of each. For the STFT implemention we compare, using the phase reconstructed signals as in practice the phase information of predicted signals will not be known. Though there are many variations and possibly better implementations of LP and STFT coefficient target based models via deep learning networks, a comparison between the effects of the different features with our model is still infomative.

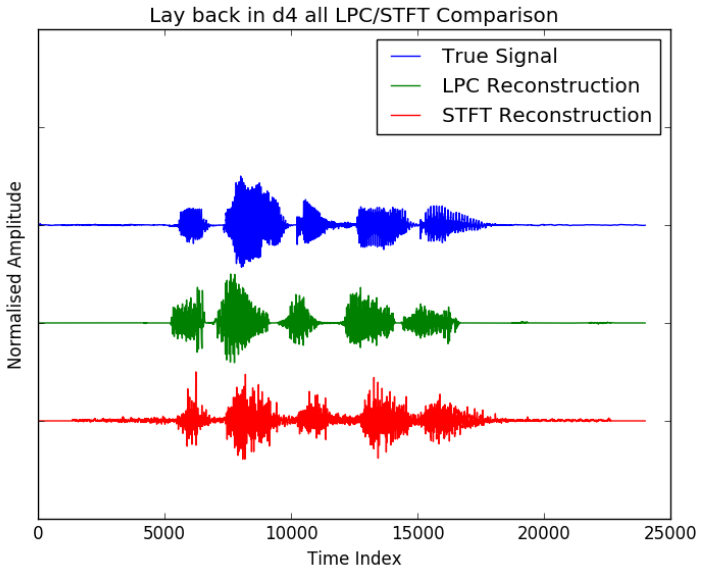
The signal plots and magnitude spectograms of two samples have been plotted in Figure 30. The signal plots of the LPC model follows the true signal more closely while those of the STFT model show much more variation in their signal and present much more noise. This is confirmed by the low SNR score provided by reconstructed phase signals in Table 2. The reconstructed phase signals typically show random 'juttering' in their signal while the LPC signals are much more compact and consistent.

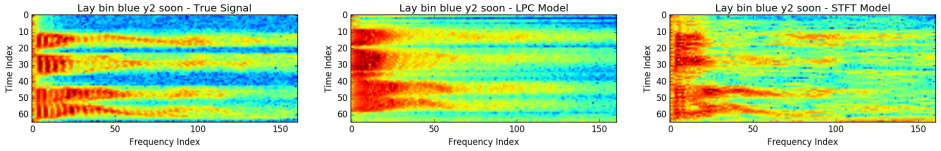
The conclusions above are also suggested by examination of the spectograms in Figure 30, in which there is much greater 'speckling' in the reconstructed phase spectograms than that of the LPC model recontructions, though they contain finer detail. While the magnitude of the spectograms was shown to be mapped relatively accurately (as shown by Figure 29 and its high correlation score in Table 2), phase reconstruction was still found to be necessary to reproduce intelligible speech. The Griffin and Lim phase reconstruction algorithm ([**?**]) attempts to preserve the magnitude of the spectrum while adding phase (see Section **??**), but it loses some of the spectogram structure and the reconstructed phase signals spectograms are poorer than those of the original predictions.

6 Multiple Speaker

err is 0.82


Lay bin blue y2 soon LPC/STFT Comparison
True Signal
LPC Reconstruction
STFT Reconstruction
Normalised Amplitude
0
5000
10000
15000
20000
25000
Time Index


Lay back in d4 all LPC/STFT Comparison
True Signal
LPC Reconstruction
STFT Reconstruction
Normalised Amplitude
0
5000
10000
15000
20000
25000
Time Index


Lay bin blue y2 soon - True Signal
Lay bin blue y2 soon - LPC Model
Lay bin blue y2 soon - STFT Model
Time Index
Frequency Index

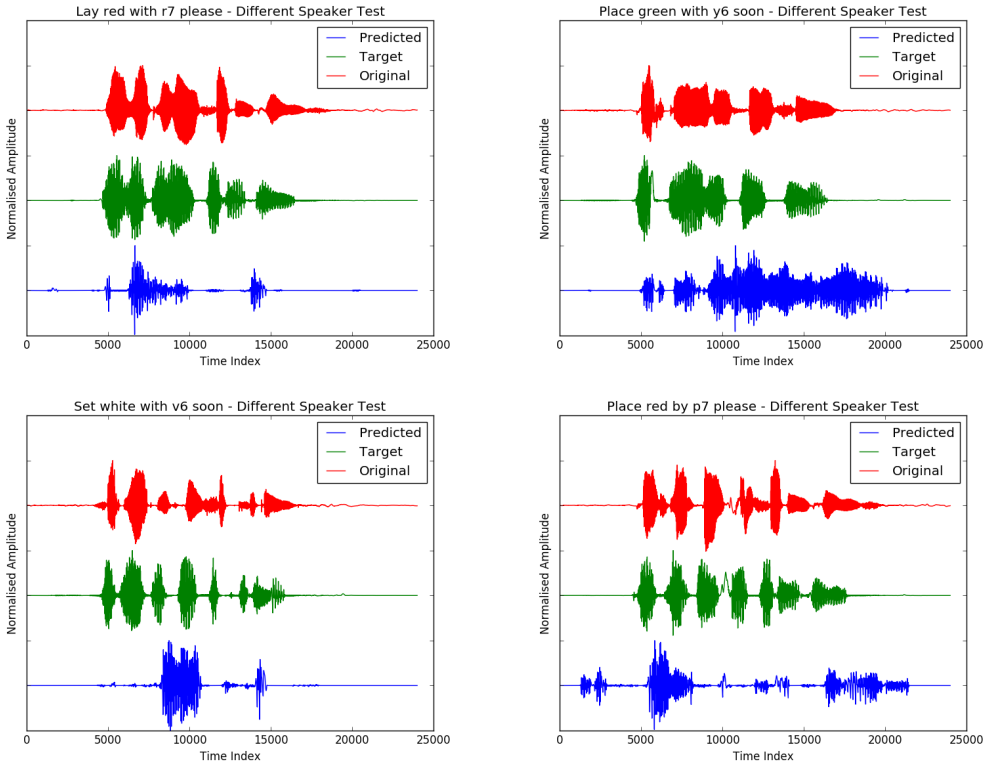

Lay red with r7 please - Different Speaker Test
Place green with y6 soon - Different Speaker Test
Set white with v6 soon - Different Speaker Test
Place red by p7 please - Different Speaker Test
Predicted
Target
Original
Normalised Amplitude
Time Index
0
5000
10000
15000
20000
25000

Figure 31

Appendix

7 Ephrat and Peleg Architecture

Order of Layer	Type of Layer	Associated Parameters
1	Conv. Layer**	3 by 3 convolution, 32 Kernels, Zero Padding*
2	Max Pooling	2 by 2 Window
3	Conv. Layer**	3 by 3 convoluiton, 32 Kernels, Zero Padding*
4	Conv. Layer**	3 by 3 convoluiton, 32 Kernels, Zero Padding*
5	Max Pooling	2 by 2 Window
6	Dropout	0.25% Probability
7	Conv. Layer**	3 by 3 convoluiton, 64 Kernels, Zero Padding*
8	Conv. Layer**	3 by 3 convoluiton, 64 Kernels, Zero Padding*
9	Max Pooling	2 by 2 Window
10	Dropout	0.5% Probability
11	Conv. Layer**	3 by 3 convoluiton, 128 Kernels, Zero Padding*
12	Conv. Layer**	3 by 3 convoluiton, 128 Kernels, Zero Padding*
13	Max Pooling	2 by 2 Window
14	Dropout	0.5% probability
15	Conv. Layer**	3 by 3 convoluiton, 128 Kernels, Zero Padding*
16	Conv. Layer**	3 by 3 convoluiton, 128 Kernels, Zero Padding*
17	Max Pooling	2 by 2 Window
18	Dropout	0.5% probability
19	Dense Layer***	512 outputs
20	Dropout	0.5% probability
21	Dense Layer***	0.5% probability
22	Dense Layer	18 outputs to match target/ no activation.

8 Smaller Architecture

FILL IN

9 Medium Architecture

FILL IN

10 Larger Architecture

FILL IN

11 Architecture with Recurrent Layer

FILL IN

NOTES ON THE MODELS

*The Zero padding here allows the same border size to be maintained throughout the convolutions.
** The convolutional layers were followed by batch normalisation and Leaky ReLu activation functions.
*** The dense layers were followed by batch normalisation and a tanh activation function.

Model was run for 70 epochs using mean squared error loss.

The optimiser used was the RMSprop Optimiser with base learning rate = 0.01 with no decay and learning rate was reduced by a factor of 10 for the last remaining epochs to help fintune the parameters.

The model was trained on 80% of the data and tested on the remaining 20%.

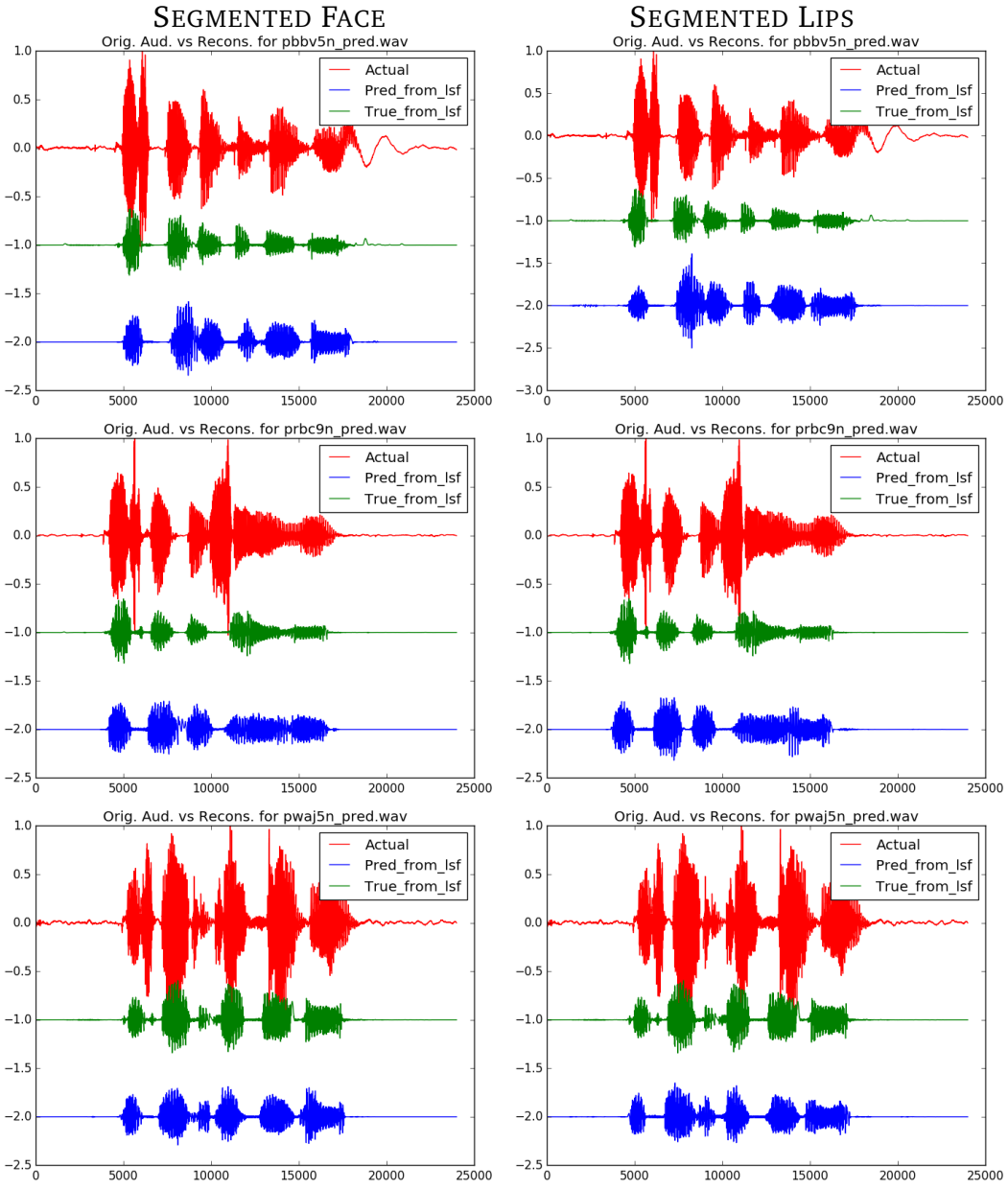
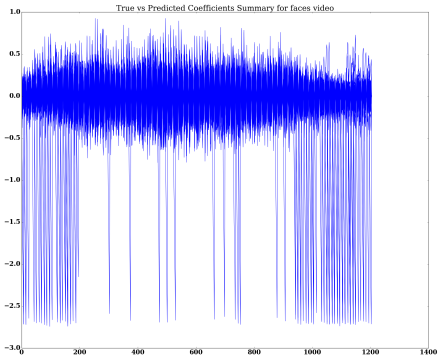

SEGMENTED FACE
SEGMENTED LIPS
Orig. Aud. vs Recons. for pbbv5n_pred.wav
Orig. Aud. vs Recons. for prbc9n_pred.wav
Orig. Aud. vs Recons. for pwaj5n_pred.wav
Actual
Pred_from_lsf
True_from_lsf

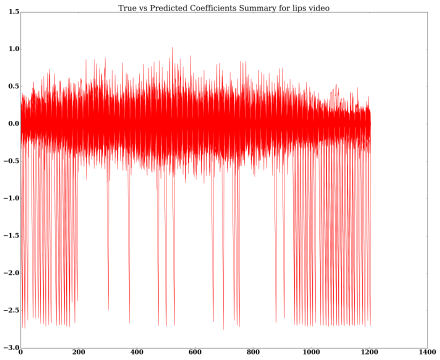
Figure 32

SEGMENTED FACE



('standard deviation of TEST1 error', 0.19184594)

SEGMENTED LIPS



('standard deviation of TEST2 error', 0.22171974)

Figure 33: Coefficient Differences taken over the test set for both types of data input

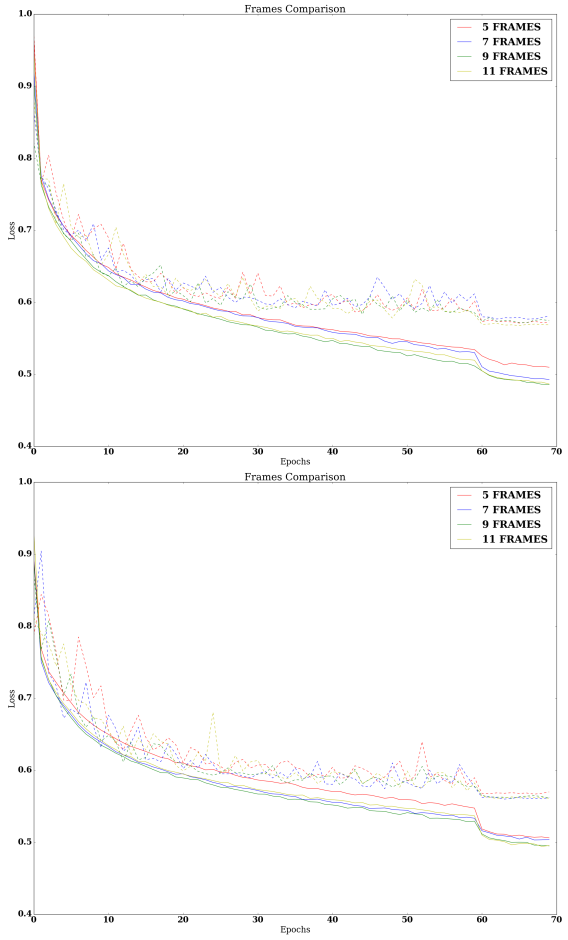


Figure 34

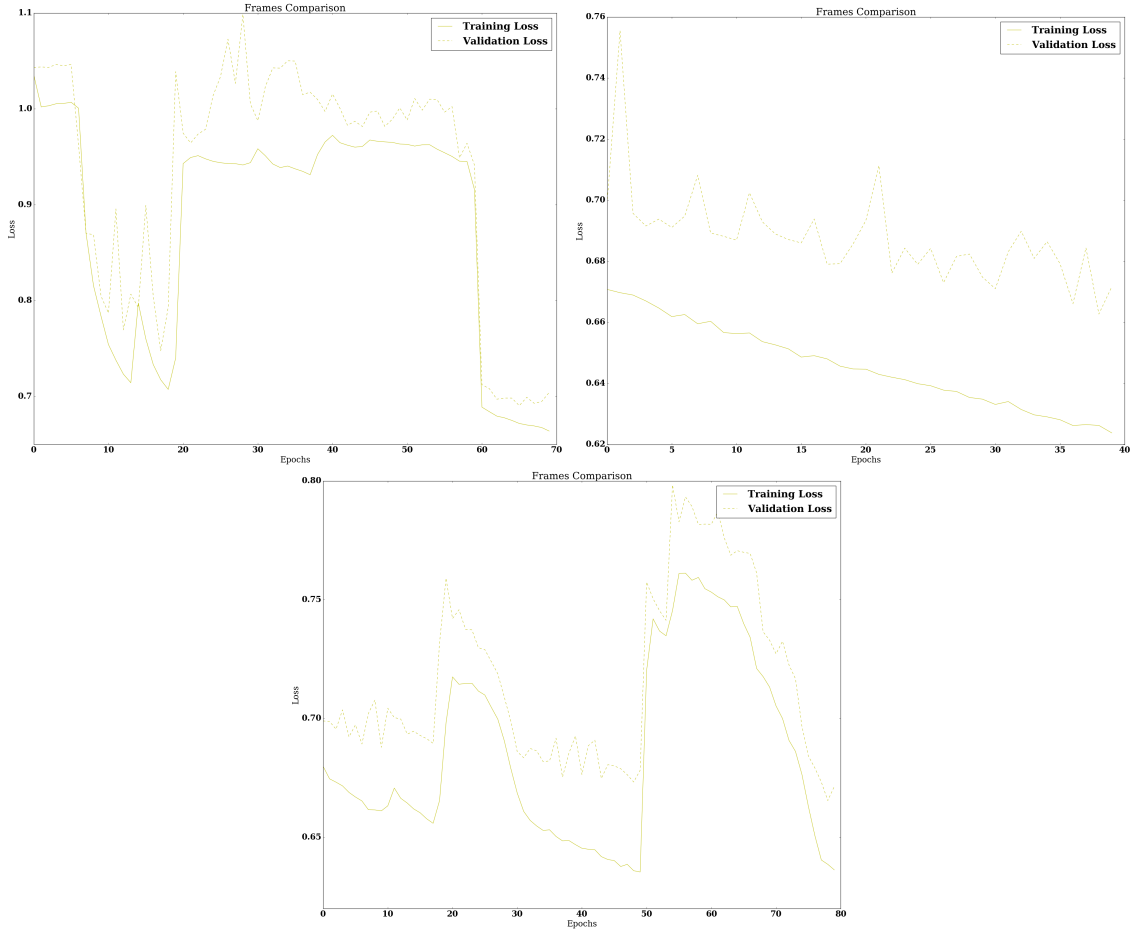


Figure 35

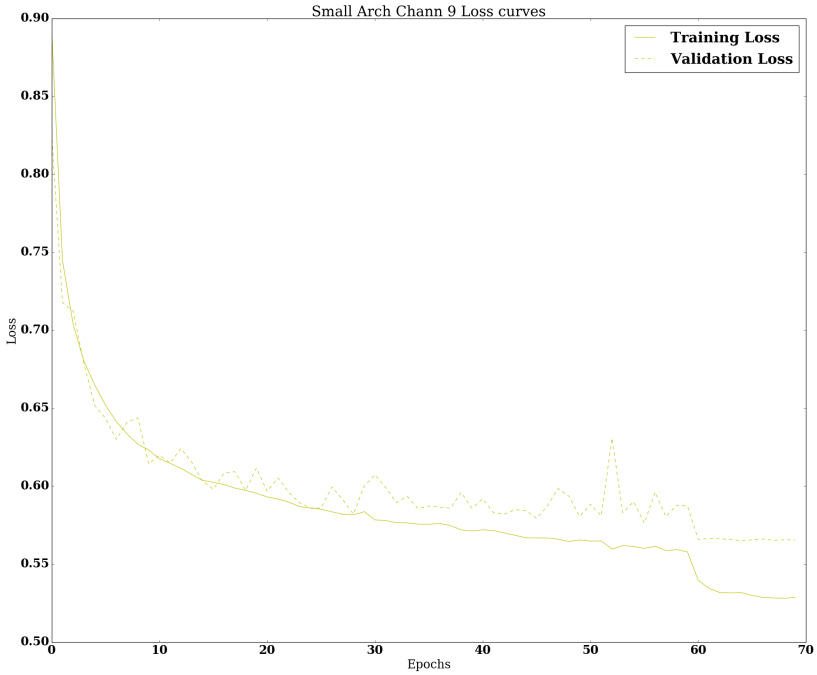


Figure 36

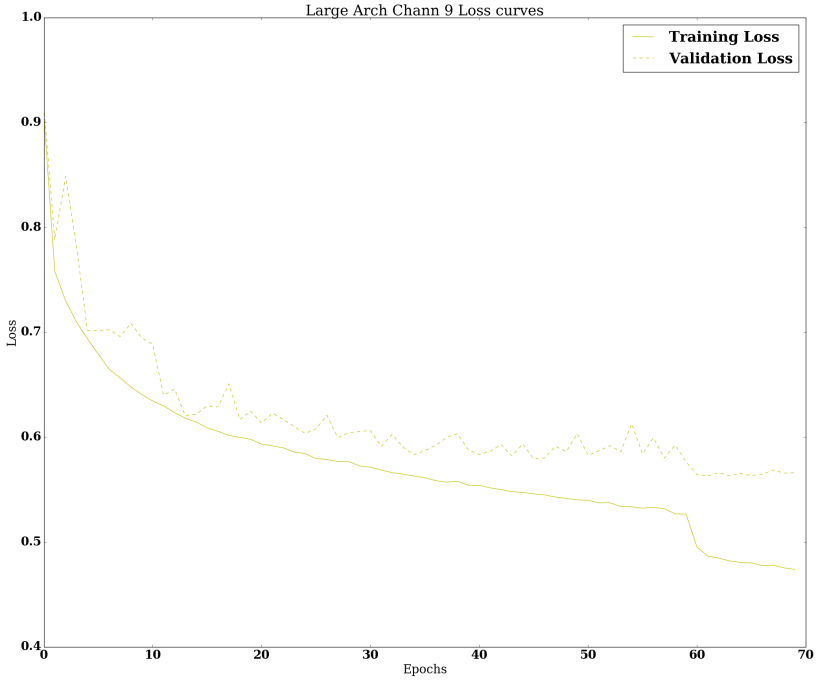


Figure 37

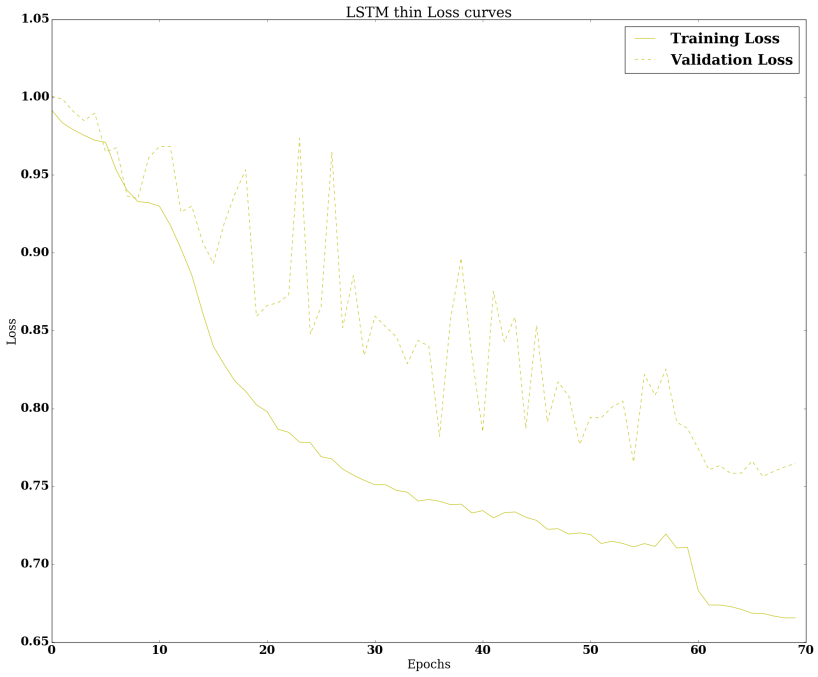


Figure 38

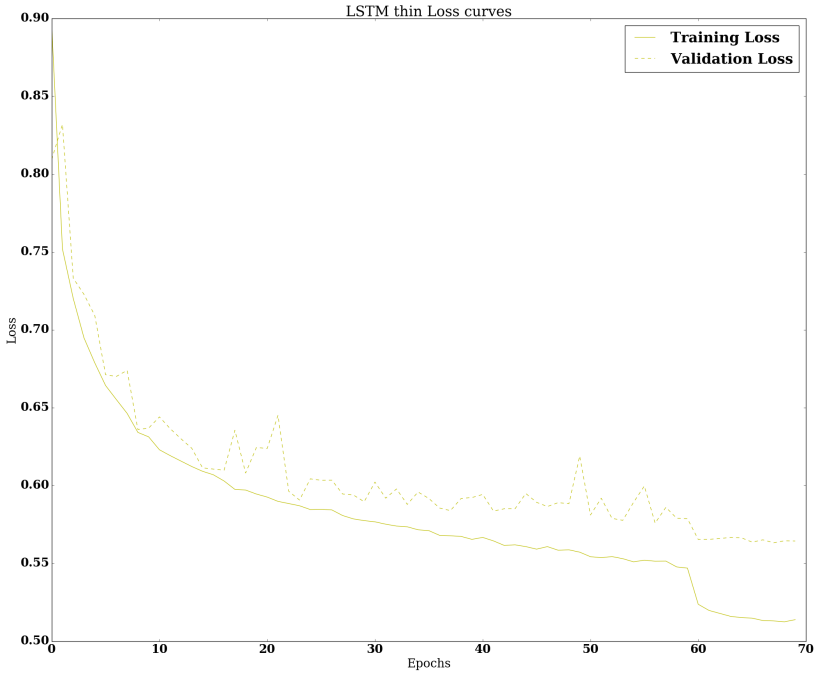
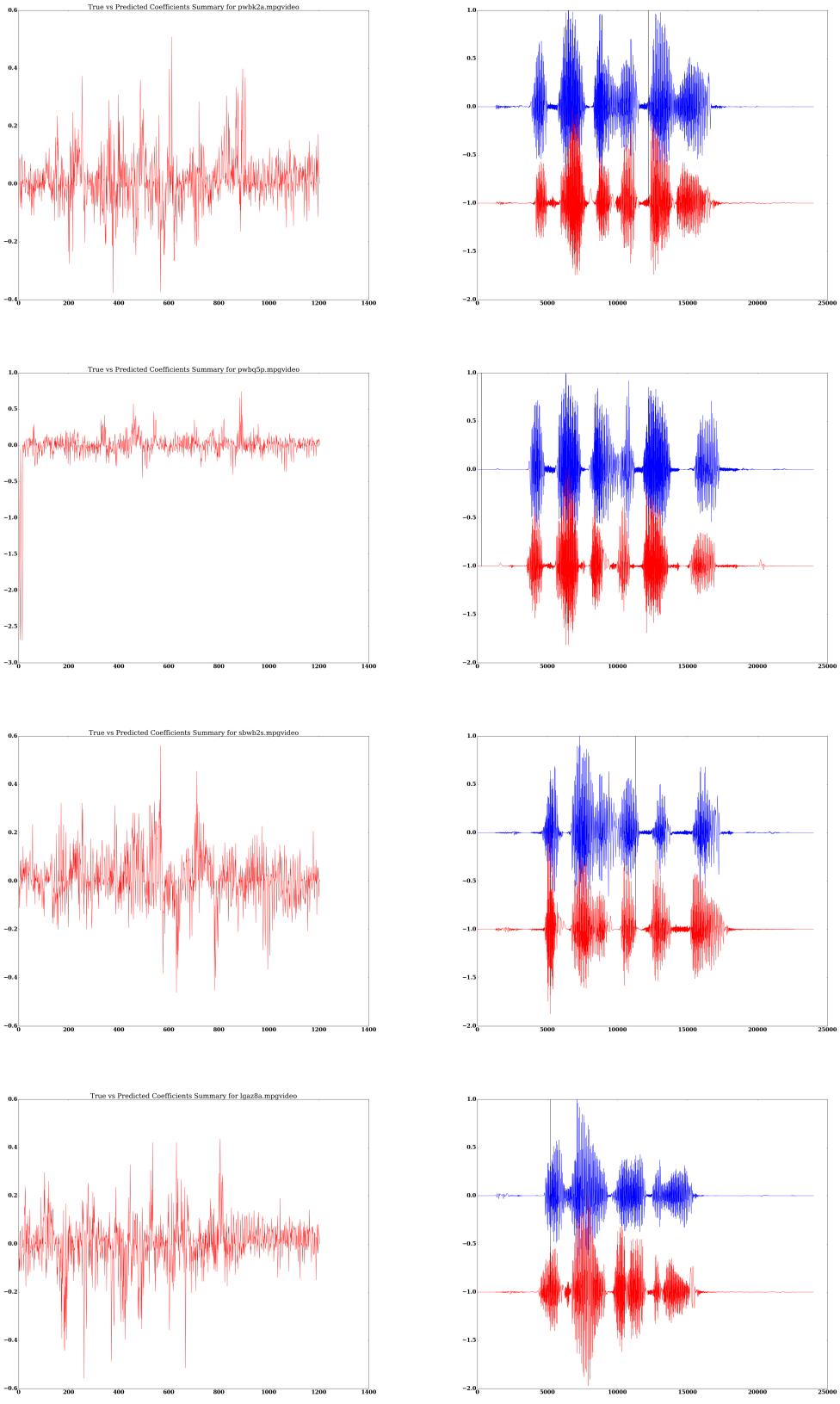

LSTM thin Loss curves
Training Loss
Validation Loss
Loss
Epochs
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
10
20
30
40
50
60
70

Figure 39

10 RANDOM SAMPLES



FURTHER RANDOM SAMPLES

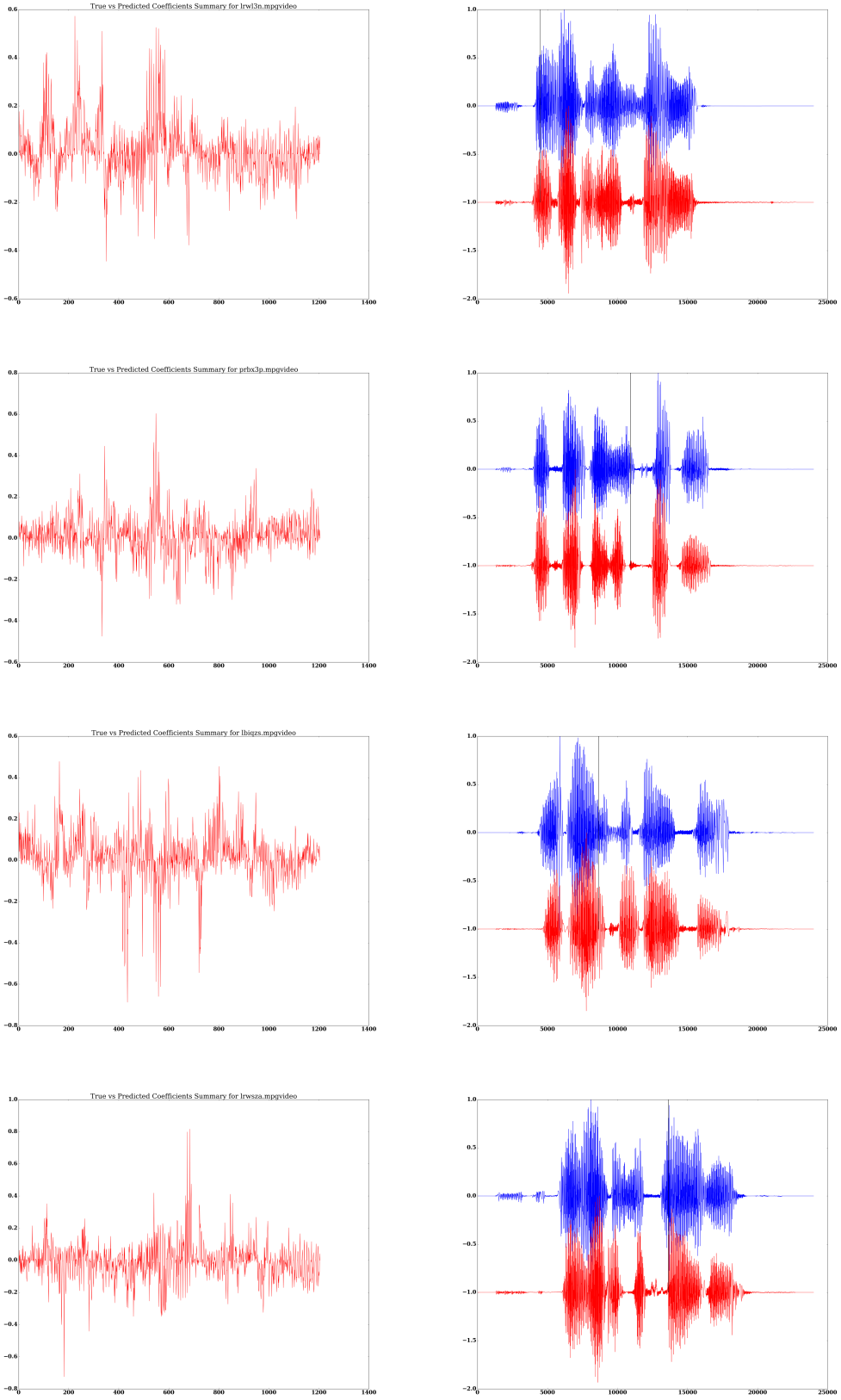
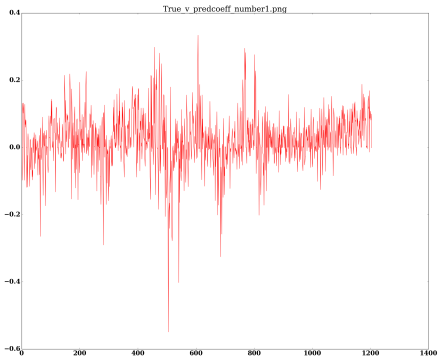
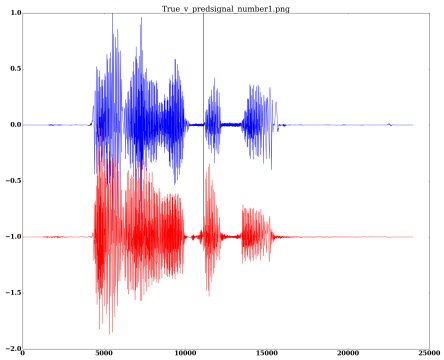
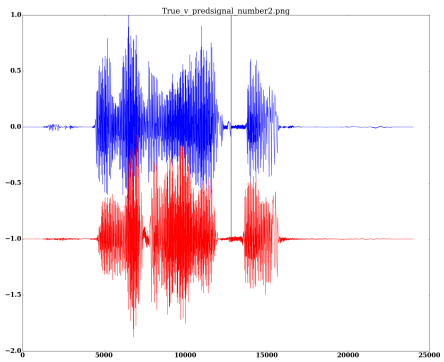


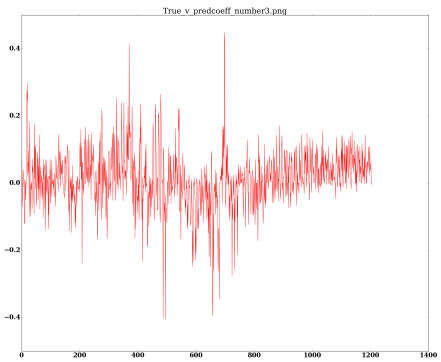
Figure 40: dfa

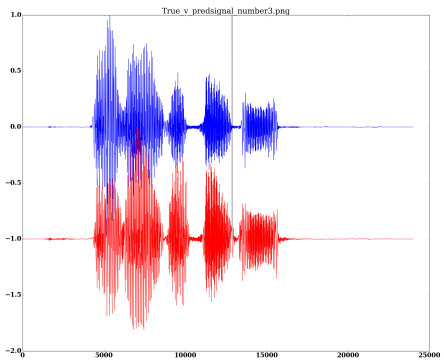
BEST SAMPLES [’bbil2s.mpg’, ’lrwr8s.mpg’, ’bgaa5n.mpg’, ’pbbi7n.mpg’, ’lwwm1p.mpg’]

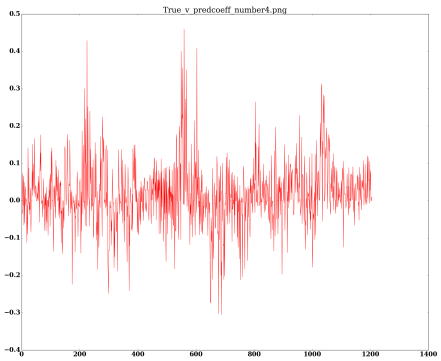


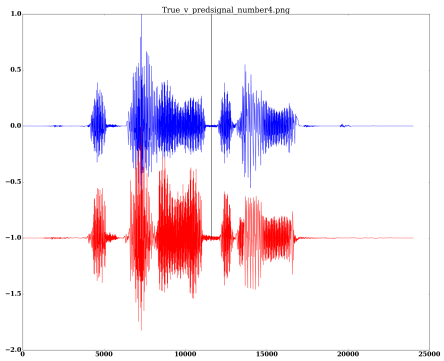




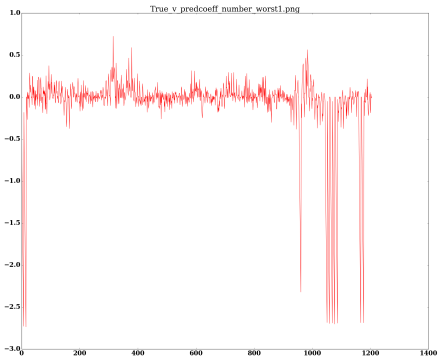


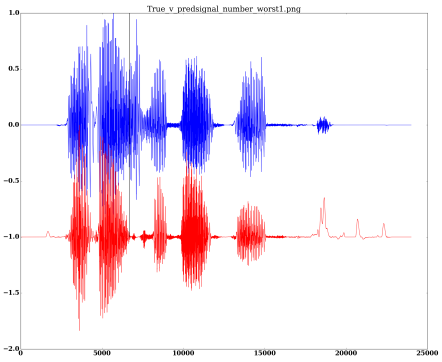


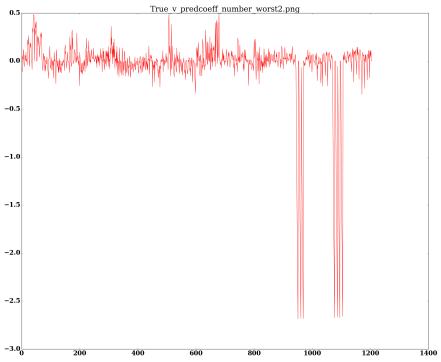


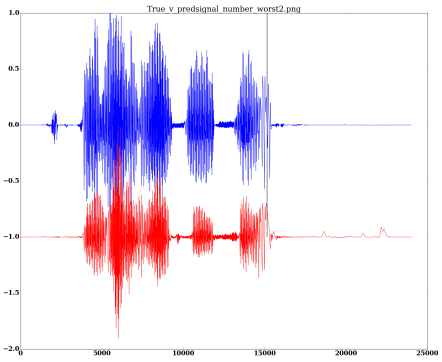


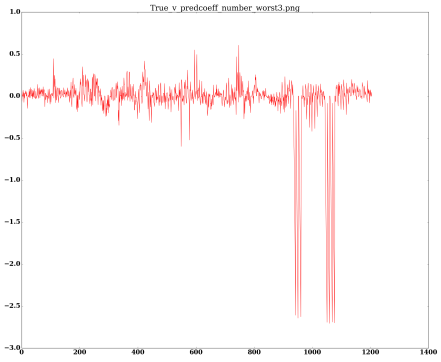
WORST SAMPLES [’sbwh8a.mpg’, ’lbaq5p.mpg’, ’lray4s.mpg’, ’pbbv5n.mpg’, ’lbbrza.mpg’]

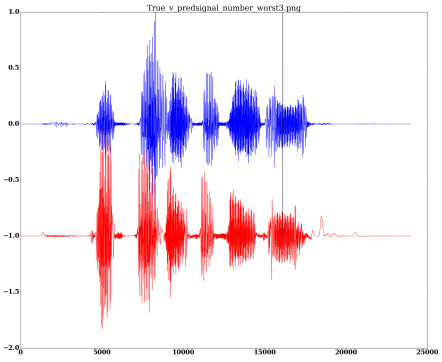


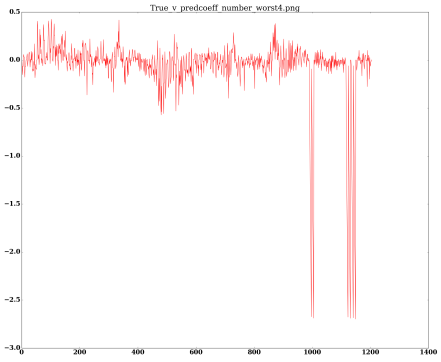


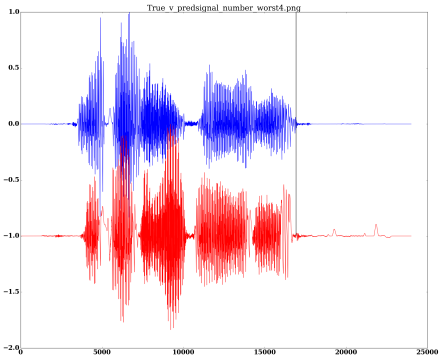












References

[1] The GRID audiovisual sentence corpus. http://spandh.dcs.shef.ac.uk/gridcorpus/#credits. pages

[2] Lipreading using convolutional neural network. https://www.researchgate.net/publication/288107194_Lipreading_using_convolutional_neural_network. pages

[3] Long Short-term Memory (PDF Download Available). https://www.researchgate.net/publication/13853244_Long_Short-term_Memory. pages

[4] Picture of a neural network - Google Search. https://www.google.co.uk/search?q=picture+of+a+neural+network&client=ubuntu&hs=d6O&channel=fs&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwiwi9_7k7HUAhXDWxoKHVfCBwAQ7AkIQg&biw=1855&bih=621#imgrc=As0100NAEqdn-M:. pages

[5] Line spectrum representation of linear predictor coefficients of speech signals. *The Journal of the Acoustical Society of America*, 57(S1):S35–S35, April 1975. pages

[6] S Agatonovic-Kustrin and R Beresford. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5):717–727, June 2000. pages 27

[7] I. Almajai, S. Cox, R. Harvey, and Y. Lan. Improved speaker independent lip reading using speaker adaptive training and deep neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2722–2726, March 2016. pages 8, 13, 14, 15, 33, 39

[8] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding Deep Neural Networks with Rectified Linear Units. *arXiv:1611.01491 [cond-mat, stat]*, November 2016. pages

[9] Yannis M. Assael, Brendan Shillingford, Shimon Whiteson, and Nando de Freitas. LipNet: End-to-End Sentence-level Lipreading. *arXiv:1611.01599 [cs]*, November 2016. pages 8, 13, 14, 15, 29, 30, 33, 37, 39

[10] Joan Bruna, Pablo Sprechmann, and Yann LeCun. Super-Resolution with Deep Convolutional Sufficient Statistics. *arXiv:1511.05666 [cs]*, November 2015. pages

[11] Joon Son Chung, Amir Jamaludin, and Andrew Zisserman. You said that? *arXiv:1705.02966 [cs]*, May 2017. pages 29, 30, 37

[12] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip Reading Sentences in the Wild. *arXiv:1611.05358 [cs]*, November 2016. pages 8, 13, 14, 15, 29, 30, 33, 37, 39

[13] Fred DeLand. *The Story of Lip-Reading: Its Genesis and Development*. The Volta Bureau, 1931. Google-Books-ID: pnnaAAAAMAAJ. pages

[14] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009. pages 16

[15] Ariel Ephrat and Shmuel Peleg. Vid2speech: Speech Reconstruction from Silent Video. *arXiv:1701.00495 [cs]*, January 2017. pages 5, 14, 15, 21, 41, 45, 46, 47, 49, 50, 52, 53, 59

[16] Gunnar Fant. *Acoustic Theory of Speech Production with Calculations Based on X-Ray Studies of Russian Articulations*. Mouton, 1970. Google-Books-ID: hZ-cLAQAAIAAJ. pages

[17] C. G. Fisher. Confusions among visually perceived consonants. *Journal of Speech and Hearing Research*, 11(4):796–804, December 1968. pages 17

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. pages 580–587, 2014. pages 16

[19] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, November 1973. pages 16

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. pages 1026–1034, 2015. pages 16

[21] Thomas Hueber and Gérard Bailly. Statistical conversion of silent articulation into audible speech using full-covariance HMM. *Computer Speech & Language*, 36:274–293, March 2016. pages 9, 10, 12, 14

[22] F. Itakura. Line spectrum representation of linear predictor coefficients of speech signals. *The Journal of the Acoustical Society of America*, 57(S1):S35–S35, April 1975. pages 24

[23] Rafal Kapela, Aleksandra Świetlicka, Andrzej Rybarczyk, Krzysztof Kolanowski, and Noel E. OConnor. Real-time event classification in field sport videos. *Signal Processing: Image Communication*, 35:35–45, July 2015. pages

[24] Christopher T. Kello and David C. Plaut. A neural network model of the articulatory-acoustic forward mapping trained on recordings of articulatory parameters. *The Journal of the Acoustical Society of America*, 116(4):2354–2364, October 2004. pages 13, 14

[25] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. pages 38

[26] Volodymyr Kuleshov, S Enam, and Stefano Ermon. Audio Super-Resolution using Neural Nets. *Workshop track - ICLR 2017*. pages

[27] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. *arXiv:1112.6209 [cs]*, December 2011. pages

[28] Y. Li, Y. Takashima, T. Takiguchi, and Y. Ariki. Lip reading using a dynamic feature of lip images and convolutional neural networks. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6, June 2016. pages 8, 13, 14, 15, 33, 39

[29] Justin London, Birgitta Burger, Marc Thompson, and Petri Toiviainen. Speed on the dance floor: Auditory and visual cues for musical tempo. *Acta Psychologica*, 164:70–80, February 2016. pages

[30] William Lotter, Gabriel Kreiman, and David Cox. Unsupervised Learning of Visual Structure using Predictive Generative Networks. *arXiv:1511.06380 [cs, q-bio]*, November 2015. pages 6

[31] Kuniaki Noda, Yuki Yamaguchi, Kazuhiro Nakadai, Hiroshi G. Okuno, and Tetsuya Ogata. Audio-visual speech recognition using deep learning. *Applied Intelligence*, 42(4):722–737, June 2015. pages

[32] S. Petridis and M. Pantic. Prediction-Based Audiovisual Fusion for Classification of Non-Linguistic Vocalisations. *IEEE Transactions on Affective Computing*, 7(1):45–58, January 2016. pages

[33] Stavros Petridis, Zuwei Li, and Maja Pantic. End-To-End Visual Speech Recognition With LSTMs. *arXiv:1701.05847 [cs]*, January 2017. pages

[34] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: A baseline for generative models of natural videos. *arXiv:1412.6604 [cs]*, December 2014. pages

[35] Jongju Shin, Jin Lee, and Daijin Kim. Real-time lip reading system for isolated Korean word recognition. *Pattern Recognition*, 44(3):559–571, March 2011. pages

[36] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, September 2014. pages

[37] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. *arXiv:1502.04681 [cs]*, February 2015. pages 6, 17

[38] Themos Stafylakis and Georgios Tzimiropoulos. Combining Residual Networks with LSTMs for Lipreading. *arXiv:1703.04105 [cs]*, March 2017. pages 13, 15

[39] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*, September 2014. pages

[40] Michael Wand, Jan Koutník, and Jürgen Schmidhuber. Lipreading with Long Short-Term Memory. *arXiv:1601.08188 [cs]*, January 2016. pages 8, 13, 14, 15, 33, 39

[41] Xiaolong Wang and Abhinav Gupta. Unsupervised Learning of Visual Representations using Videos. *arXiv:1505.00687 [cs]*, May 2015. pages 6, 17

[42] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. *arXiv:1409.2329 [cs]*, September 2014. pages

[43] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *arXiv:1311.2901 [cs]*, November 2013. pages 31