UNIVERSITÉ DE JEAN MONNET

MASTERS 1 THESIS

Automatic Sychronisation of Subtitle Track With Live Audio

Author:
Joshua FENECH

Supervisor:
Dr. Rémi EMONET

A thesis submitted in fulfillment of the requirements
for the degree of Masters in Machine Learning & Data Mining

June 25, 2018

Declaration of Authorship

I, Joshua FENECH, declare that this thesis titled, "Automatic Sychronisation of Subtitle Track With Live Audio" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Joshua Fenech

Date:

UNIVERSITÉ DE JEAN MONNET

Abstract

MLDM

Masters in Machine Learning & Data Mining

Automatic Sychronisation of Subtitle Track With Live Audio

by Joshua FENECH

Subtitle availability is distinctly lacking in cinemas, limiting access to people who are hard of hearing and from minority groups. Whilst technologies are readily available they are rarely implemented, hence there is motivation to devise a technique to enable users to independently access subtitles on their personal devices and synchronise them with minimum difficulty. In order to do so, subtitles were used to create an objective array identifying whether subtitles are present, and this was used to train a convolutional neural network based on the work of [10]. Initial results of 83% after tuning were boosted to 93% by swapping the activation and batch normalisation layers. However, further analysis of results indicates varied results. An array matching method was also proposed in order to provide alignment in real time, but this requires further testing and development.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Deafness is considered to be the most prevalent impairment worldwide, affecting at least 278 million people [14]. However, current provisions for those who may struggle to fully hear and comprehend human speech in commercial settings is distinctly lacking. Cinemas routinely show only 2 or 3 showings a week with subtitles in the UK [13], restricting the range of films and times that can be enjoyed, adding to the sense of exclusion experienced by disabled people. There are technologies available to provide captions for individual users[5], but these rely on each institution having purchased these, and are far from universally available. Even those that do have them only provide them for a small selection of movies, 6/27 at one cinema chain in New Zealand [7]. Therefore, to ensure availability of these it is desirable to find methods that could be applied on the user-side so that subtitles can be made available in whichever context the user desires. Time stamped subtitle files are freely available as SubRip files (.srt file extension) that contain a series of text entries and associated start and stop times, but these rely on knowing the exact start time of a film so that the 2 files are synchronised. This is extremely difficult to achieve in a live setting where the exact commencement of the feature film is unclear, and the introduction of even a small time shift can make subtitle viewing an unwatchable experience. Hence, the demand arises to automatically synchronise a subtitle file using real time audio signals to locate the current position in a video.

1.2 Background

This problem was tackled in a laboratory setting[10], so that a user can synchronise audio and subtitle tracks for future viewing on a personal device. This method extracted and sampled the audio at 16kHz, extracted the features using the common automatic speech recognition (ASR) technique of splitting the signal into 20-40ms frames and extracting the Mel Frequency Cepstral Coefficients for each frame and using these as the predictive features. The subtitle track was then used to create a binary array indicating whether subtitles (and therefore human speech) occur in each corresponding frame to those used for MFCC, providing the target variable. A Recurrent Neural Network (RNN) was initially trained, a logical approach given the chronological nature of the data, but due to the fact that several frames must be used as input, the accuracy was limited to the duration of the number of input frames: if 10*0.05s samples are required, the accuracy is limited to 0.5s which is insufficient in practise. Therefore a Convolutional Neural Network (CNN) was trained, taking as input just 1 sample, with several one-dimensional convolutional layers followed by several dense layers to output an array containing the

probabilities that human speech is present in each frame. Technically, it should be noted that since the presence of subtitles is the target variable, a subtitle- (rather than speech-) detector was built. Minimisation of the log loss function was then used to align the subtitles array with the probability array, the delay calculated and applied to the subtitle track; this loss function was chosen in order to maximise the accuracy by penalising false classifications. Using this process, after training the NN and hyperparameter tuning, the log-loss minimum is clearly defined 1.1. Using dynamic programming to take account of the fact that every step must be evaluated across the signal, this process took 45 seconds for a tv show and 2 minutes for a film, the whole process using CPU's. This approach is promising and formed the basis upon which this study was formed. There were key differences in the aim that require further study in order to implement this in a live setting. Since this was performed on the recorded data rather than a live recording, the signals are pure and contain no significant noise, and so a model that could handle such settings needed to be identified. Furthermore, there would be more uncertainty in the start times of the film which would need to be accounted for – search can commence from the beginning and match the 2 full arrays since the aim is to match the full sound track with the subtitle track (with less concern for excessive processing time), whereas in real time, access to the sound track would occur incrementally as new audio is recorded. The Sabater study was informal, with the results released as a blog post and so lacked a rigorous (or at least reporting of the) approach and testing environment, with no account of the relative complexities and possible different approaches that could be taken.

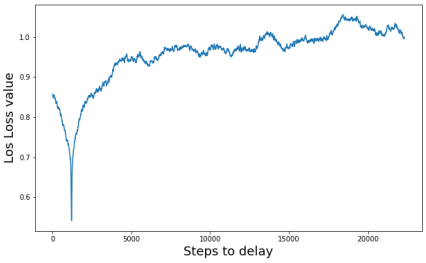

FIGURE 1.1: Log loss plot when matching full audio to full subtitles

A similar approach was adapted in [9] in order to align a draft script with a live presentation which loosely followed this but often presented changes in the script and its chronology (insertions, substitutions and deletions). To do so, 12 order MFCC's were used again, and "normalised energy coefficient augmented by the corresponding delta and delta-delta coefficients information as features"(p2), but an ASR engine formed of "continuous HMM's [Hidden Markov Models] with context dependent acoustic units, where each unit is modelled with a 16 component Gaussian Mixture Model with diagonal covariance matrices" was used. The HMM's enable sequence data to be modelled and are not constrained in the same way as RNN's that must take multiple sequential samples in order to predict a sample, thereby reducing the accuracy achievable. This was a more ambitious objective in that the audio had not been prerecorded and so to compare the audio to the subtitles, the MFCC's were compared to a phoneme network in order to identify utterances and therefore alignment of the tracks. In addition, this system substituted automatically generated subtitles when no script was available due to insertions. This model achieved an accuracy of up to 100% on newscasts that adhered strictly to the script and naturally dropped to 30% when the programme became unscripted. In addition, due to the preprocessed nature of the scripts the latency was deemed to be negligible. Another approach has been implemented [2] In which audio is generated from text by Text To Speech (TTS) systems and this is used to synchronise the text and audio.

Chapter 2

Preprocessing

2.1 Audio Format

It was deemed most appropriate to use the compressed MP3 extracted from the MPEG-4 video file for a number of reasons. Although using the uncompressed wav form would present a more faithful reproduction of the audio, and possibly provide more data to train on, the additional complexity far outweighed any possible gains. The MFCC extraction is relatively expensive DETAIL and testing on the wav file proved prohibitively long. The space requirements were far larger and approached the file size of the compressed video and audio combined. Furthermore, since the aim is to identify human speech, the lossy mpeg compression regime is explicitly designed to mimic the filtering that occurs in human hearing using psychoacoustic modelling, whereby imperceptible frequency ranges, signals with subthreshold amplitudes and frequencies masked by other more dominant frequencies are excluded**??**, among many other filtering techniques. This process is well-suited to feature extraction for ASR as human speech is naturally adapted to be comprehended by human ears, and vice versa.

The spoken voice frequency lies between 300 to 3400 Hertz[11] and so telephones sample at 8kHz in order to satisfy the nyquist criterion which defines the sample rate necessary to avoid aliasing, where the sample rate frequency is insufficient to capture the full details of the waveform, to be: $f_s > 2B$ (sampling frequency greater than twice the bandwidth). The sample rate was chosen to be 16kHz in order to avoid this phenomenon, and since the files are compressed the memory requirements are not excessive.

2.2 Mel Frequency Cepstral Coefficients

In order to extract the features most relevant to speech, Mel Frequency Cepstral Coefficients (MFCC's) were chosen due to their robustness to noise compared to another popular feature extraction technique, Linear Predictive Cepstral Coefficient [1]. Others have found that it performs less well without some preprocessing but is relatively fast to compute [12]. MFCC's are considered the baseline, reliable method and there is a python library (python_speech_features) available for computing these, and so this was the method chosen. This technique also applies pschoacoustic modelling to the signal by applying a series of transformations.

2.2.1 Frame signal into short frames

The signal is framed into overlapping frames in order for the frequency analysis to be performed. This assumes that audio spectra are effectively constant over short

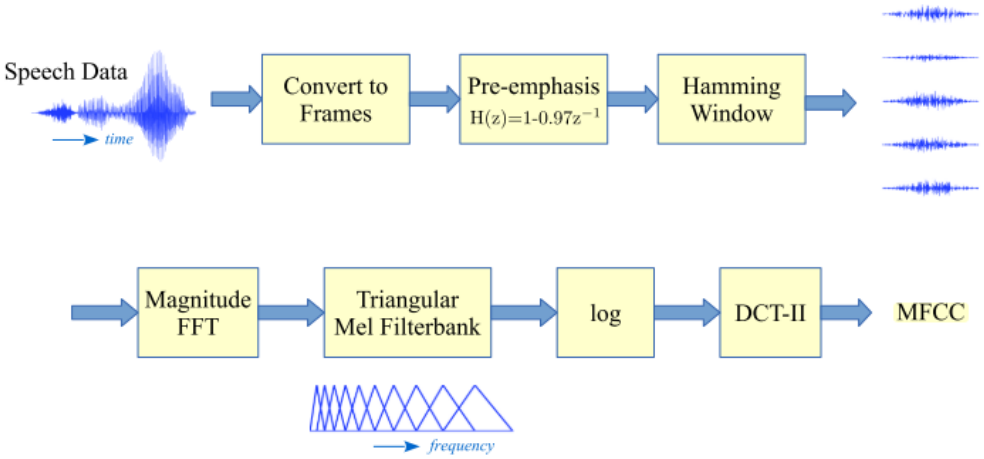

FIGURE 2.1: Steps of MFCC Feature Extraction [8]

time frames (typically 20-30ms). The windows are overlapping so that continual frequency features spanning multiple frames are captured.

2.2.2 Pre-emphasis

There is a preemphasis applied, where for each value in the frequency domain a high-pass filter is applied:

$$s_2(n) = s(n) - a * s(n-1) \tag{2.1}$$

where n is the sample number, s is the signal in the frequency domain and s2 is the transformed signal. This has 3 effects[6]:

- balance the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower frequencies,
- avoid numerical problems during the Fourier transform operation,
- may also improve the Signal-to-Noise Ratio (SNR).

2.2.3 Periodogram

For each frame the periodogram is calculated, which encapsulates the relative energy in different frequency bands of the signal. This is done using the Discrete Fourier Transform (discrete due to the digital nature of the signal):

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \text{for } 1 \leq k \leq K \tag{2.2}$$

where $h(n)$ is an n sample long analysis window (e.g. hamming window), and K is the length of the FFT. For a 16kHz signal in 25ms frames, N equates to 400 samples. The periodogram-based power spectral estimate for the speech frame is then given by:

$$P_i(k) = \frac{1}{N}|S_k|2 \tag{2.3}$$

i.e. we take the absolute value of the complex fourier transform of each coeffecient, and square the result.

2.2.4 Mel Filterbank

This step applies a series of filters (26 commonly) to the periodogram, composed of vectors of length 26 with nonzero values at different sections along the vector. The filters are nonuniform, increasing in width with frequency. This emulates the nature of human hearing to have lower sensitivity to changes in frequency as frequency increases. This is based on the Mel scale which models this nonlinearity and describes exactly how to space the bins. The spectrum $P(f)$ is warped along its frequency axis f (in hertz) into the mel-frequency axis as $P(M)$, where M is the mel-frequency, using:

$$M(f) = 2595 \log((1 + f/700) \tag{2.4}$$

A non-linear frequency scale transformation is applied based on the observation that human hearing does not perceive frequencies linearly but, like amplitude, on a logarithmic scale2.2.

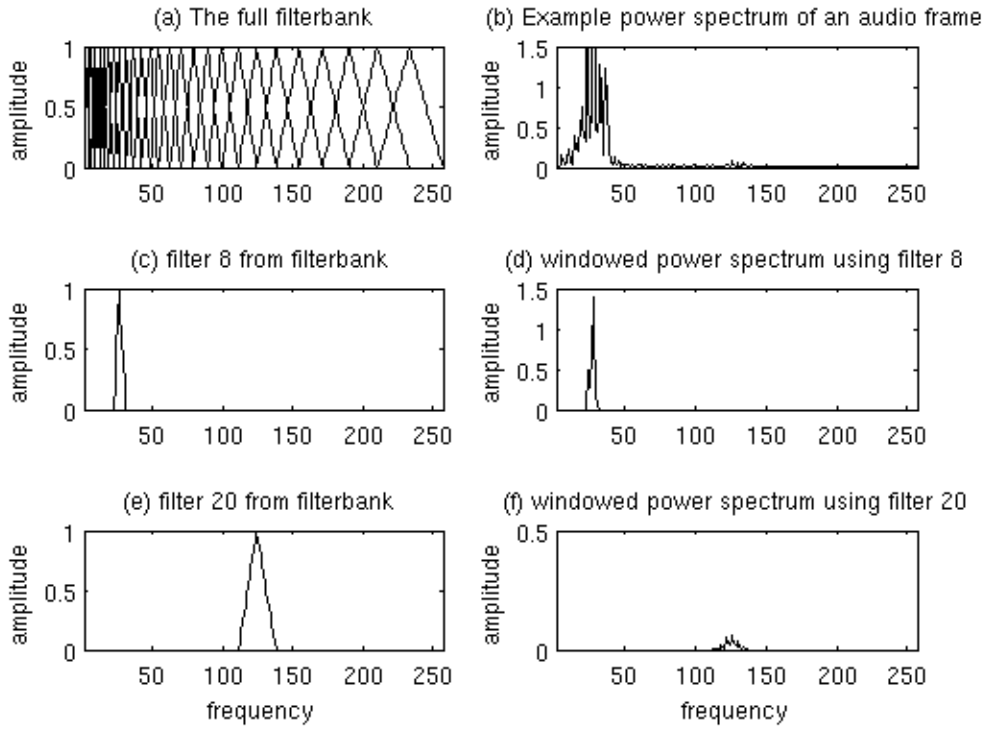

FIGURE 2.2: Filtering with Mel Filterbanks [4]

2.2.5 Log and DCT

The log of the energies in each filterbank is taken, and a Discrete Cosine Transformation (DCT) is applied to extract 13 new coefficients of this new spectrum. The DCT has several benefits over other transformations, such as the Fourier transform, in that it does not interpret components as infinite waves, and so is better suited where signals are more likely to be interrupted, as the framing process does. Furthermore,

only real numbers are output which is more readily dealt with by learners such as CNN's.
Generally the largest 13 coefficients are taken that represent the most important information. https://dsp.stackexchange.com/questions/31/how-do-i-interpret-the-dct-step-in-the-mfcc-extraction-process probably need better source...

2.2.6 Output

MFCC extraction therefore function as a feature selection technique that reduces the 400 samples each with 256 values (8 bits) of amplitude representation to this lower feature space specifically designed to emulate how humans hear, which are themselves well placed to interpret speech signals. However, Figure 2.3 indicates that the relation between MFCC's and speech presence is not immediately obvious. MFCC2 appears to have the most distinctive correlation with the coefficients doubling in size from ∼30 to ∼60 in the absence of speech, but other features have similar values, and there is significant intraclass variation.

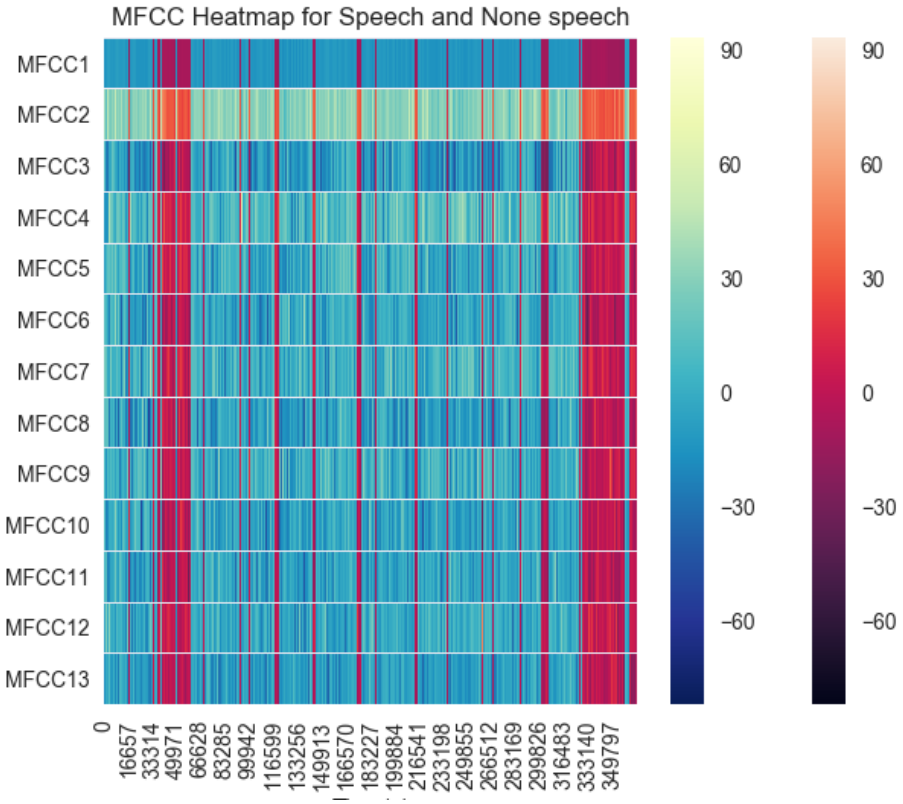

FIGURE 2.3: The MFCC's extracted from a Game of Thrones episode. Bluish shades indicate subtitles are absent, pinkish shades indicate they are present. Feature window index is used as a proxy for time.

2.2.7 Processing Time

The MFCC extraction was one of the more costly processes in this method, but not prohibitively so. To generate the coefficients for approximately 47 minutes of audio took on average 45.4 seconds, equating to around 1 second per minute of audio.

2.3 Subtitles

The MFCC's provide the descriptive variables upon which to make predictions, and the objective variable is defined to be the presence of subtitles. This setup was chosen to enable the model to provide synchronisation using solely information contained in the srt file: since this subtitle synchroniser is aimed principally at cinemagoers, it is assumed that the original audio/video is unavailable, but it is assumed that an srt file is accessible. These are readily available on the internet for most films and are small (essentially text) files and so networking and device requirements would be minimal. They contain a series of entries made up of a start time, stop time, and a string to be presented on the screen at these times.

2.3.1 Subtitle Array

In order to generate the subtitle array, which has length equal to the number of MFCC frames and is composed of binary values representing subtitles or no subtiles present:

- Times are converted into seconds from start.
- The time at which the last subtitle disappears from the screen is used to determine the length in seconds that subtitles are present. Subtitles need only be synchronised when there are actually subtitles present.
- The number of frames required is determined by dividing the length in seconds by the frame step (10ms) (the frame length is 25ms, but these are overlapping).
- An array of zeros, pb_array, is generated with columns for an index, start time and binary subtitle value to indicate presence of subtitles. It is of equal length to the number of frames required.
- The 2 arrays, pb_array and subs_array, are parsed concurrently, with the frame times of pb_array checked to see if they lie within a times entry, and the value changed to 1 from 0 if true.

2.3.2 Pseudocode

Algorithm 1 pb_array_fill

```
1:  procedure
2:      i ← 0
3:      j ← 0
4:      m ← pb_array_length
5:      n ← subs_array_length
6:      while True do
7:          if i > m then break
8:          if j > n then break
9:          if pb_array[i] start time ≥ subs[j] start time then
10:             if pb_array[i] end time < subs[j] end time then
11:                 pb_array[i] ← 1
12:                 i ← i + 1
13:         else
14:             j ← j + 1
15:
16:
```

This method was evaluated with 5 trial runs on 59:18 minutes of audio, equating to 355,830 MFCC samples and took on average 12.03 seconds. This is a little longer than would be desired in practice but is not excessively long. In addition it is efficient in that it parses over the arrays only once. The output is a vector with length equal to number of MFCC frames, filled with binary values indicating presence of subtitles. This provides the objective feature to predict.

Chapter 3

Learning

3.1 Learner Architecture

The network architecture implemented by Sabater was used as a starting point. This used convolutions in 1d with kernel size 3, where the filter is initially applied over the MFCC features. This process is identical to the more traditional 2d convolutions in computer vision, except the kernel is a vector rather than an array. 3 filters were used.

This architecture associates adjacent MFCC's with each other in a hierarchical fashion. It could be hypothesised that during speech the changes in pitch/frequency occur in a consecutive fashion, with sudden changes from different parts of the spectrum unlikely, and so this information can be compressed with the network learning the best means of doing so. No padding was used, and although this means central coefficients are over represented in relation to those at the edges, it could also be hypothesised that these occur less often and, being at the edges of the frequency spectrum, are less critical to comprehension, thus features can be reduced further. As in 2d convolutions, the exact nature of the filters are left to be learned during the optimisation (back propogation?) process. 12 filters in 2 convolution layers gives 24 parameters to be learned. The 1D convolutions output tensors of shape $(V-(F-1), N_F)$, where V is the feature vector size, F is the filter size and N_F is number of filters. To produce the correct shape for the dense layer, the tensors are flattened out to feature vectors again. The ReLu activation function is applied at this point, before applying batch normalisation and dropout. These layers are then repeated once before the output, which is activated with a sigmoid activation function.

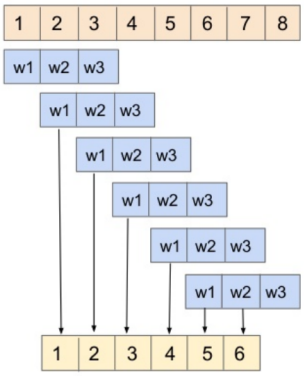

FIGURE 3.1: Convolutions in 1d [3]

3.1.1 Activation Functions

The activation functions provide the mathematical function that emulates the non-linear, biphasal response in the brain. For all the internal layers, the ReLU function was used as this is less computationally expensive than the sigmoid where all neurons are guaranteed to fire at all times, and the use of a dropout layer removes any dying neurons to give a sparser output. The output activation function was a sigmoid, which is well suited for classification and ensures that all outputs from the final dense layer will be evaluated.

3.1.2 Batch Normalisation

Although normalisation of the features is associated with increased performance and reduced training times, the features were not normalised directly since this relies on having access to the full dataset and when deployed it would not be possible to do this since we are only granted access incrementally. However, this was compensated for by application of the batch normalisation layer. In order to learn, it is most effective when the training and test data have the same distribution. However, at each layer the outputs retain this relation only weakly due to the non-linearity of the activation functions, a process known as internal covariance shift, and so at each layer the new distribution must be relearnt. In addition, due to the flattening of the sigmoid function at larger values, the optimisation process slows as the gradient varies less over fixed learning rates. This can be compensated for by renormalising after every layer, but in order to do so the mean and variance must be computed across the whole training set: this is expensive and "not differentiable everywhere". Instead, batchnorm stochastically computes the mean and variance over much smaller overlapping batches and the outputs then normalised to a gaussian with zero mean and unit variance across each of the dimensions, output $\hat{x}$. In order to ensure that the activation functions are thus not constrained to this gaussian distribution, a further transformation is made: $y = \gamma\hat{x} + \beta$, and these new parameters are incorporated into the learning process to be identified via backpropagation, hence different distributions can be inferred directly. This can be seen as having a regularising effect as the outputs through each layer are now dependent on what other examples were present in that batch via the mean and variation with which it is normalised rather than being processed independently. Since this process can be applied at each layer of the network, a single example can be related to many other examples. At test time, the learned features of the gaussian are fixed and applied to the data to give deterministic responses.

3.1.3 Dropout

Dropout is one method of limiting overtraining and the interdependence of neurons within layers. Nodes are defined to be active in a stochastic function as determined by a probability parameter. Hence when training, every time a sample is propagated through the network a random subset, in proportion to the probability parameter, of the neurons will not produce any output. Hence, later layers cannot depend on the presence of all inputs and this reduces the tendency of neurons to co-adapt, activating in similar ways to other neurons and providing no additional information. In this way, a much larger set of network configurations is randomly sampled, and hence can be viewed as operating as an ensemble method for 1 hidden layer. All architectures share weights – whenever we use a hidden unit its got the same weights as it has in other architectures. Extreme form of bagging – very large number of architectures trained on just 1 sample. Sharing of weights means that every model is very strongly regularised – the weights learned from 1 architecture are stored before the architecture is sampled again. A dropout value of 0.2 was applied.

3.2 Tuning

Once the general neural network architecture was decided, there were still a range of hyperparameters to tune. In order to evaluate the effect of different optimisers,

batch size and number of epochs, the hyperas package was used. This enables models with different parameters to be evaluated and the optimal parameters identified using grid search cross validation and an optimisation algorithm, Tree of Parzen Estimators (TPE) was used here. http://hyperopt.github.io/hyperopt/. The range of parameters tested was: batch_size=[32, 64, 128], epochs=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 50, 100], optimizer=['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam'], In so doing, the best performing parameters were identified to be: optimiser = Nadam, batch size = 32. [However, due to the random initialisation process this produced different results every time...]

3.3 Training/Test Set Size

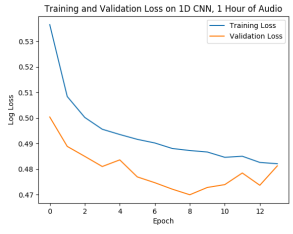

FIGURE 3.3: Validated on a separate episode of The Walking Dead

In order to identify the size of training set required, the optimised learner was run on a separate validation set and the loss evaluated with epoch to see how training accuracy would improve with more data. What was thought to be a small dataset consisting of 1 hour of audio from The Walking Dead in fact was enough for the log-loss plot to indicate that the best results achievable had already been done so, and no more data would be required to replicate the results on data with a similar distribution. Interestingly the validation loss is less than the training loss.. The validation results gave [0.81, 0.50] [accuracy, loss], which emulated closely the training results. This indicated that an hour of audio should be sufficient, but to ensure success, a full feature film was used for training.

3.4 Test Time

3.4.1 Clean Data

Prior to tuning, the model achieved respectable results Validation Loss: 0.47 Validation Accuracy: 0.76. Time: $\sim$ 103 seconds

After tuning, these results were improved slightly: Validation Loss: 0.48, Validation Accuracy: 0.81

The model was then evaluated on a full feature film, Goldstone and produced similar results: Validation Loss: 0.49, Validation Accuracy: 0.84 Time: $\sim$ 259 seconds

However, there is some debate over the position of the batch normalisation layer and whether it should be placed before or after the activation [sources here], and so the tests were rerun with this swap implemented. In doing so, the training time and accuracy were improved significantly.

Validation Loss: 0.50, validation Accuracy: 0.93, time: $\sim$ 216 seconds, suggesting better generalisation at test time.

3.4.2 Noisy Data

A live recording of a Game of Thrones episode was made in order to produce data that that would more closely emulate the real-world setting. To do so, an episode was played and an mp3 recording made at the same from the same laptop, with care taken to ensure that background noise was present, including speech. The episode

recorded was one on which the learner had been evaluated to provide direct comparison between noisy and clean data. One problem encountered was that it was difficult to identify exactly where the film began in the recording, with an approximate uncertainty of $\pm$ 1s, and so it was not obvious whether the predicted array and true array were aligned. Although this is to be expected in deployment, in testing it would be preferential to know that the value being tested against for accuracy corresponds to the actual predictors. Nonetheless, an accuracy of 76% was achieved on this noisy set.

3.4.3 Confusion matrices

Unfortunately, when we examine the confusion matrices for the different datasets we see that the accuracies are less impressive than first appears, possibly due to the unbalanced datasets.

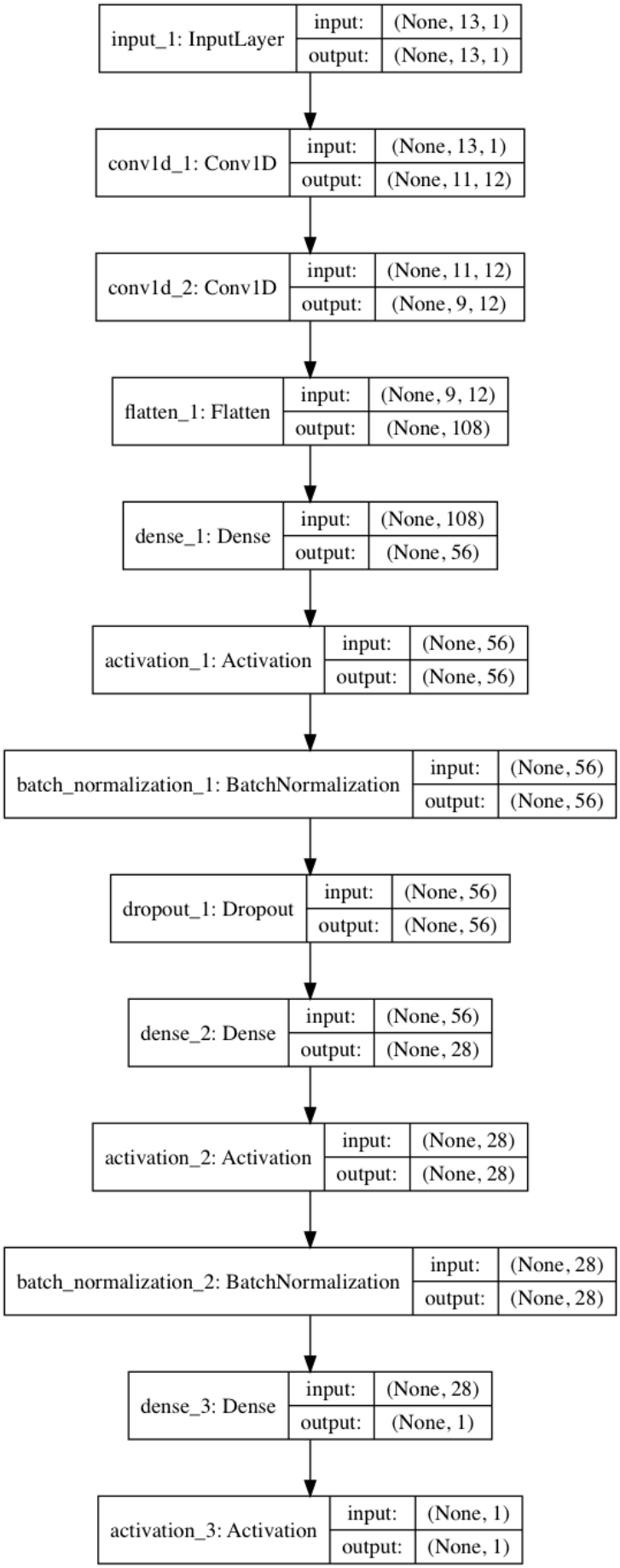

FIGURE 3.2: Initial Model Architecture

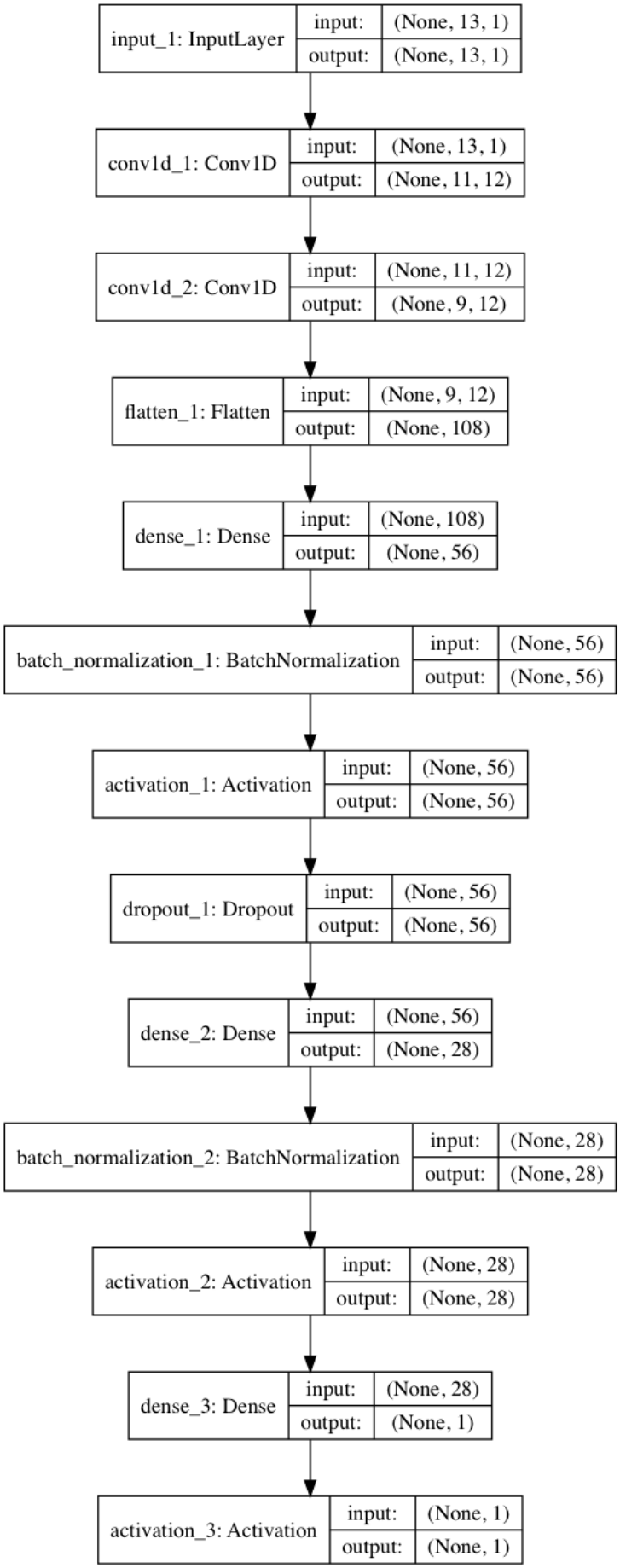

FIGURE 3.4: Model Architecture with batch normalisation and activation layers swapped.

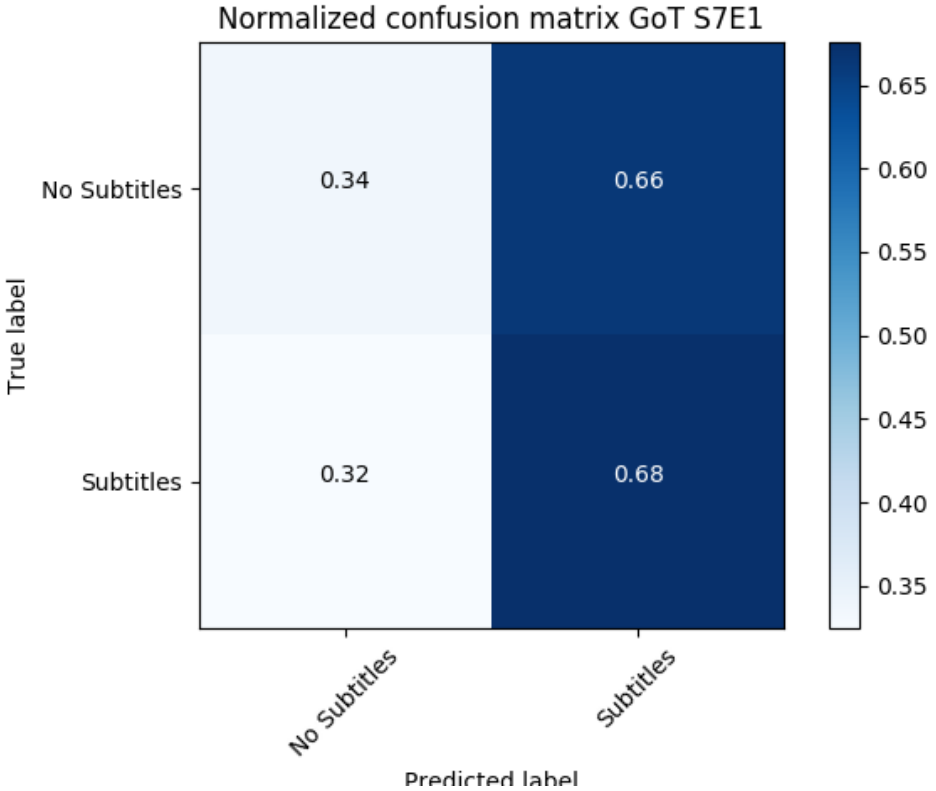

FIGURE 3.5: The equal proportion with which subtitles are predicted across both true labels suggests a fixed ratio is being applied to the data.

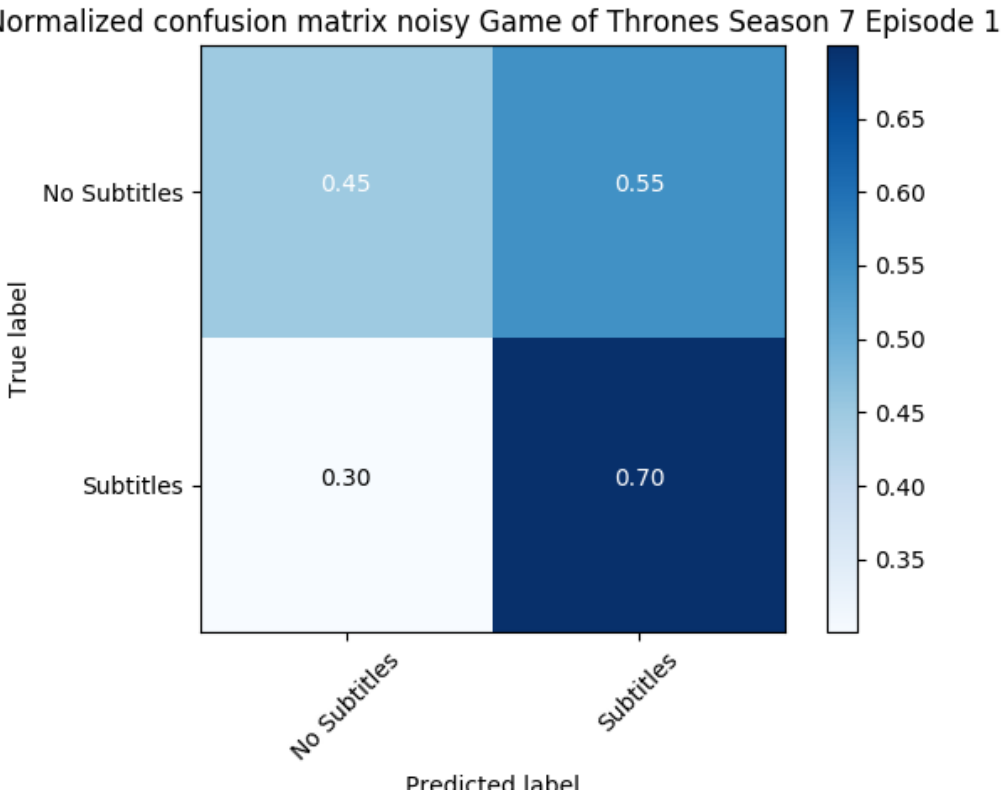

FIGURE 3.6: Interestingly, the noisy dataset performs better at test time, with greater accuracy in predicting the features in both classes.

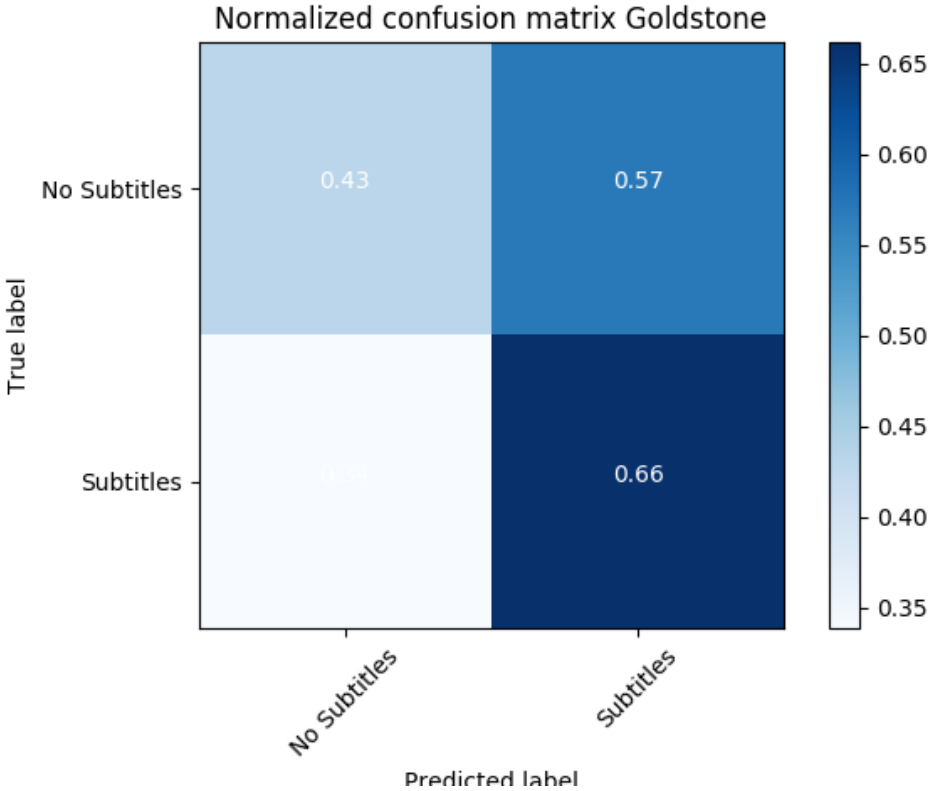

FIGURE 3.7: This shows better results than on the clean Games of Thrones episode but is still less than the noisy dataset.

Chapter 4

Synchronising

4.1 Array Matching

With a sufficiently accurate subtitle predictor, the next challenge was to match the predictions to the true array. Sabater initially took the whole prediction array and computed the log loss over different positions of the subtitles array in order to find the minimum which would equate to the best match. However, this approach assumes full access to both arrays: in order to match subtitles in real time, the probability array would be accessed incrementally as new audio is recorded and features extracted.

4.1.1 Fixed Recording Duration

Initially, it was modelled that audio of fixed duration would be recorded in order to match array (50 frames initially, equivalent to 500ms), whilst noting the time for the algorithm to run, and applying a delay to the subtitles once the match had been found. The length of the audio was to be optimised, and it was assumed that live audio data would commence acquisition after the beginning of the film so that the algorithm does not attempt to match audio features generated before the film has actually begun, perhaps due to background noise or advertisments. To test this independently of the subtitles predictor, the array match was given a sample of the subtitles array N steps after the beginning of the full subtitles array in order to evaluate the behaviour. Interestingly, this approach consistently returned index 0 as the best position. On investigation, it was revealed that since there were no subtitles at the beginning of the film the sample array was made up of entirely zeros, and so the function returned this as the optimal spot: there were in fact multiple optimum positions (generally all at the start) where no subtitles were present and it simply returned the first position which fulfilled the minimisation condition.

4.1.2 Live Recording

Whilst this seemed like an obstacle initially, it was realised that the duration of time in which no subtitles occurs can be considered a distinctive feature in itself, especially when combined with the assumption that synchronisation has commenced not long after the beginning of the film: it is unlikely that viewing would commence after the majority of the film has already finished. In addition, new data can be continuously acquired by simply appending new predictions based on new features extracted from the incoming audio. This foreknowledge, a product of the specific nature of this problem, can be utilised to shrink the search space significantly by defining the first match to be obtimal, whilst continuously expanding the number of

examples to match against the base truth. In practice this is implemented by recording data and extracting the features, and if no speech is found, continue to acquire new data until a subtitle is predicted. This subtitle would correspond to the first subtitle present in the film and would provide the distinctive feature to search for when matching. This could be done in 2 ways, either the array generated, of a long sequence of zeros followed by a 1, is matched using the log loss, or in an even greater simplification, the subtitles are simply commenced as soon as speech is detected. Alternatively, if speech commences from the beginning, the recorded features should provide enough variation that a match can be found when searching from the beginning.

Chapter 5

Conclusions

5.0.1 Preprocessing

An efficient means of extracting features from subtitle files has been combined with traditional speech centred feature extraction in order to generate data for training easily.

5.0.2 Learner

A learner has been trained that can identify speech with approximately 80% accuracy in a range of conditions. This is less than commercial speech detectors which approach 100% accuracy, but the conditions generally involve much less noise where the speaker is emphasised. The results could possible be improved by appending the log energy to the feature vector, using padding when taking convolutions to ensure all features have equal representation, using noise reduction techniques. Alternative learning frameworks could be evaluated to ensure the highest accuracy possible is attained.

5.0.3 Array Matching

Given a sufficiently accurate model, a means of matching predictions to a true array in real time has been proposed. This attempts to utilise additional information peculiar to the setting of the cinema to restrict the search space. This is effective assuming insignificant speech noise from the surroundings, but requires additional testing in order to identify the most effective sample length to match with when speech is present in the film.

Bibliography

[1] Utpal Bhattacharjee. "A comparative study of LPCC and MFCC features for the recognition of Assamese phonemes". In: *International Journal of Engineering Research & Technology* 2.3 (2013), pp. 1–6. ISSN: 2153-1234. DOI: 10.4236/jis.2012.34041. URL: https://pdfs.semanticscholar.org/7c8f/8a9d5ba85788b569bc04ca9f07d6ce689pdf.

[2] N Campbell. "Autolabelling Japanese ToBI". In: *ICSLP 96. Fourth International Congress on Conference on Language Processing Proceedings* 4 (1996), pp. 2399 – 2402. DOI: 10.1109/ICSLP.1996.607292. URL: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=607292.

[3] Universitat Politècnica de Catalunya. *Recurrent Neural Networks II (D2L3 Deep Learning for Speech and Langu...* 2017. URL: https://www.slideshare.net/xavigiro/recurrent-neural-networks-2-d2l3-deep-learning-for-speech-and-language-upc-2017.

[4] *Crypto*. URL: http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[5] *Dolby CaptiView*. URL: https://www.dolby.com/us/en/professional/cinema/products/captiview.html.

[6] Haytham Fayek. 2016. URL: http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html.

[7] *HOYTS Cinemas*. URL: https://www.hoyts.co.nz/experiences/cc.

[8] D. S. Pavan Kumar. "Feature Normalisation for Robust Speech Recognition". In: *CoRR* abs/1507.04019 (2015). arXiv: 1507.04019. URL: http://arxiv.org/abs/1507.04019.

[9] Alfonso Ortega et al. "Real-time live broadcast news subtitling system for Spanish". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (2009), pp. 2095–2098. ISSN: 19909772. URL: http://vivolab.es/demos/subtitle/subtitling.pdf.

[10] Alberto Sabater. *Automatic Subtitle Synchronization – Machine Learnings*. 2017. URL: https://machinelearnings.co/automatic-subtitle-synchronization-e188a9275617.

[11] R Sandanalakshmi et al. "Speaker Independent Continuous Speech to Text Converter for Mobile Application". In: (), pp. 1–7.

[12] Urmila Shrawankar and V M Thakare. "Techniques for Feature Extraction In Speech Recognition System : A Comparative Study". In: *International Journal Of Computer Applications In Engineering, Technology and Sciences (IJCAETS),ISSN 0974-3596* (2013), pp. 412–418. arXiv: 1305.1145. URL: http://arxiv.org/abs/1305.1145.

[13] *Subtitled Cinema Showings in London*. URL: http://yourlocalcinema.com/london.central.html.

[14] Debara L. Tucci, Michael H. Merson, and Blake S. Wilson. "A summary of the literature on global hearing impairment: Current status and priorities for action". In: *Otology and Neurotology* (2010). ISSN: 15317129. DOI: 10.1097/MAO.0b013e3181c0eaec.