Thesis2

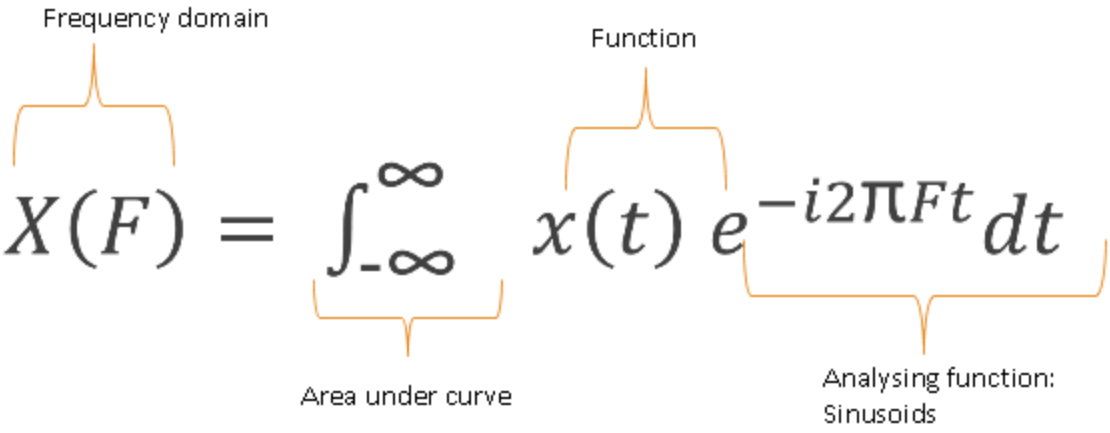
Deep Scattering Network

Curve fitting is a very common theme in pattern recognition. The concept of invariant functions convey mapping functions that approximate a discriminating function when a parent function is reduced from a high dimensional space to a low dimensional space $\cite mallat2016understanding$. In this chapter an invariance function called a scattering transform enables invariance of groups of deformations that could apply to speech signals thereby preserving higher level characterisations useful for classifying speech sounds. Works done by $\cite peddinti2014deep, zeghidour2016deep, anden2011multiscale, sainath2014deep$ have shown that when the scattering spectrum are applied to speech signals and used as input to speech systems have state of the art performance. In particular $\cite sainath2014deep$ shows 4-7% relative improvement in word error rates (WER) over Mel frequences cepstal coefficients (MFCCs) for 50 and 430 hours of English Broadcast News speech corpus. While experiments have been performed with hybrid HMM-DNN systems in the past, this thesis focuses on the use of scatter transforms in end-to-end RNN speech models.

This chapter iterates the use of the Fourier transform as the starting analysis function for building invariant functions and then discusses the Mel filter bank solution and then establishes why the scattering transform through the wavelet modulus operator provides better invariance features over the Mel filters.

Fourier transform

The Fourier transform often referred to as the power spectrum, allows us to discover frequencies contained within a signal. The Fourier transform is a convolution between a signal and a complex sinusoid from $-\infty$ to $+\infty$ (Figure **LaTeX parse error**).



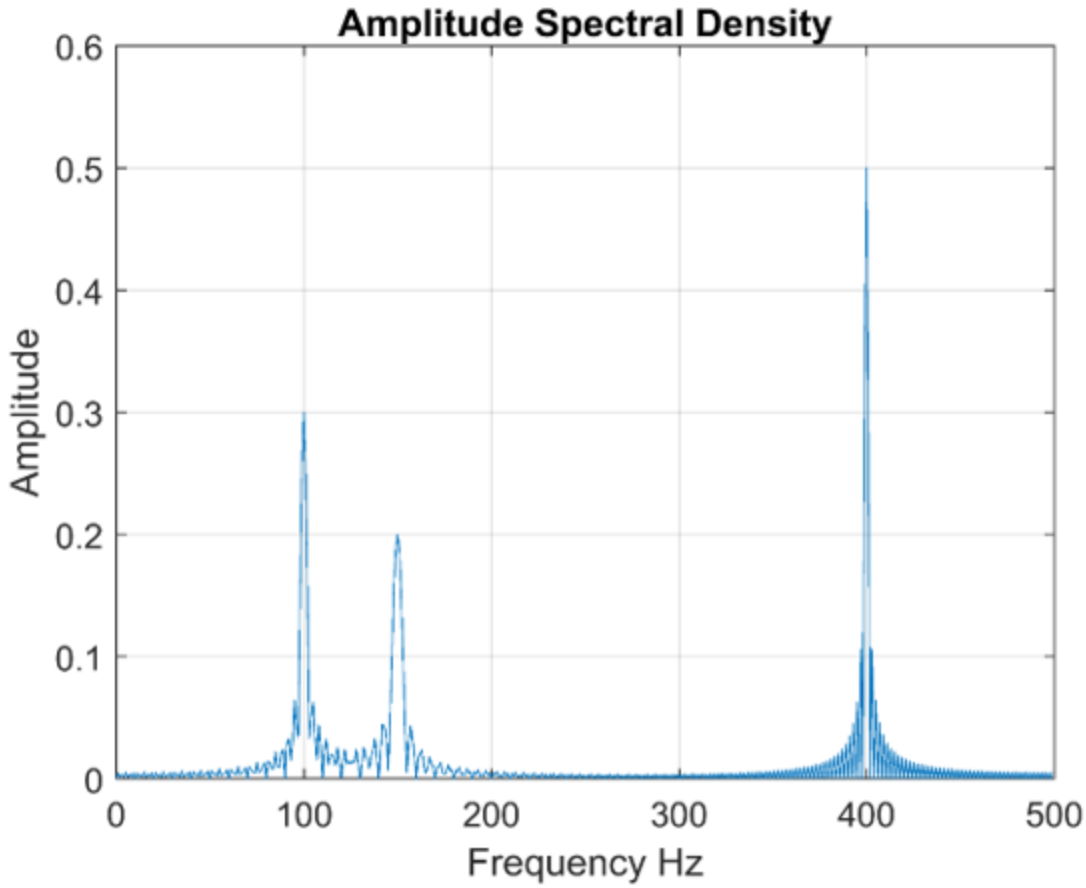
alt text

LaTeX parse error

From the orthogonal property of complex exponential function, two functions are orthogonal if $\int f(x)g(x) = 0$ where $f(x)$ and $g(x)$ are complimentary functions, one being referred to as the analysis equation and the other referred to as the synthesis function.

If the discrete form of the Fourier transform analysis equation is given by
$a_k = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{(-j \frac{2\pi kt}{T})}$
Then, the corresponding synthesis equation is given by
$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{(j \frac{2\pi kt}{T})}$

Recall that $x(t)$ is the original signal while a_k is the Fourier Series coefficient. This coefficient indicates the amplitude and phase of the original signal's higher order harmonics indexed by k such that higher values of k correspond to higher frequency components. In a typical spectrogram (figure \ref{fig_4_2_spectral}), it can be seen that the energy of the signal is concentrated about a central region and then harmonic spikes of energy content exponentially decrease and taper off. Therefore in figure \ref{fig_4_2_spectral}, the energies are concentrated at frequencies of about 100, 150 and 400 hertz.



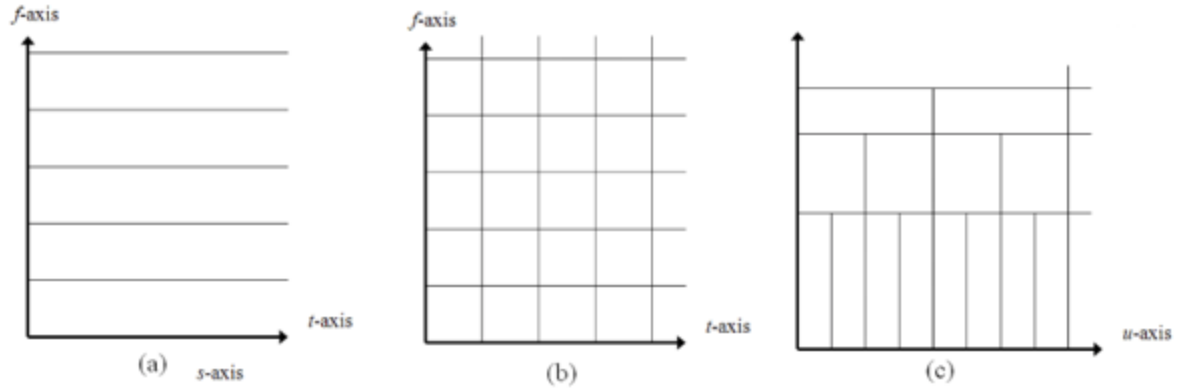
alt text

LaTeX parse error

The Fourier transform discussed in the previous section constitutes a valuable tool for the analysis of the frequency component of a signal. However is not able to determine when in time a frequency occurs hence is not able to analyse time related signal deformations. The Short-time Fourier Transform (STFT) attempts to salvage this by windowing the signal in time signal and performing Fourier transforms over sliding windows sections of the original signal rather than the whole signal. There is however, a resolution trade off that ensues from this operation such that, the higher the resolution in time accuracy, the lower the frequency accuracy and vice versa. In the next section on the continuous wavelet transform, how the wavelet transform improves on the weaknesses of the Fourier Transform and the STFT is reviewed.

Wavelet transform

The continuous wavelet transform can be defined as a signal multiplied by scaled and shifted version of a wavelet function $\psi(t)$ referred to as the mother wavelet. The time-frequency tile-allocation of the three basic transforms examined in the first part of this chapter is illustrated in figure \ref{fig_4_3_tftile}



alt text

LaTeX parse error

It can be seen here that for the Fourier transform there is no time information obtained. In the STFT, as there is no way of telling where in time the frequencies are contained, the STFT makes a blanket range of the resolution of the window and is therefore equally tiled potentially losing information based on this setup. For the case of the wavelet, because it is a scaled and shifted convolution, it takes care of the this problem providing a good resolution in both time and frequency. The fundamental representation of the continuous wavelet function is:

\begin{equation}

$$C(a,b) = \int f(t) \frac{1}{\sqrt{a}} \psi \left(\frac{\overline{t-b}}{a} \right) dt$$

\label{eqn_c4_wavelet01}

\end{equation}

In this equation, a and b respectively represent the scaling and shifting resolution variables of the wavelet function. This is referred to as a mother wavelet. A few other mother wavelet functions discussed later in this chapter. Generally a mother wavelet is identified as being energy spikes in an infinite signal whose accumulative energy sums to zero.

Discrete and Fast wavelet transform

Synthesis and analysis equations (\ref{eqn_c4_fourier02} and \ref{eqn_c4_fourier01}) can be formulated as a linear combination of the basis $\phi_k(t)$ such that the basis, $\phi_k(t) = e^{j2\pi kt}$, and it's conjugate or orthonormal basis, $\widetilde{\phi}_k(t) = e^{-j2\pi kt}$, equations (\ref{eqn_c4_fourier02} and \ref{eqn_c4_fourier01}) now become

\begin{equation}

$$x(t) = \sum_k a_k \phi_k$$
\label{eqn_c4_dwt02}
\end{equation}
\begin{equation}
$$a_k = \int x(t)\widetilde{\phi}_k(t)$$
\label{eqn_c4_dwt01}
\end{equation}

With respect to scaling and shifting variables of continuous wavelet transforms in equation (\ref{eqn_c4_wavelet01}), a similar linear combination transformation can be applied by constructing orthonormal bases parameters, referred to as scaling (ϕ) and translating (ψ) functions. For example, a simple Haar mother wavelet transform associated with a delta function, it is seen that:
\begin{equation}
$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k)$$
\label{eqn_c4_dwt03}
\end{equation}
\begin{equation}
$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$$
\label{eqn_c4_dwt04}
\end{equation}
where j is associated with the dilation (scaling) parameter and k is associated with the position (shifting) parameter. If the haar coefficients $h_{(\cdot)}[n] = 1/\sqrt{2}, 1/\sqrt{2}$ are extracted we have the following dilation and position parameters.
\begin{equation}
$$\phi(t) = h_\phi[n]\sqrt{2}\phi(2t - n)$$
\label{eqn_c4_dwt05}
\end{equation}
\begin{equation}
$$\psi(t) = h_\phi[n]\sqrt{2}\psi(2t - n)$$
\label{eqn_c4_dwt06}
\end{equation}

For any signal, a discrete wavelet transform in $l^2(Z)^1$ can be approximated by
\begin{equation}
$$f[n] = \frac{1}{\sqrt{M}} \sum_k W_\phi[j_0, k]\phi_{j^0,k}[n] + \frac{1}{\sqrt{M}} \sum_{j=j^0}^{\infty} \sum_k W_\psi[j, k]\psi_{j,k}[n]$$
\label{eqn_c4_dwt07}

\end{equation}

Here $f[n]$, $\phi_{j^0,k}[n]$ and $\psi_{j,k}[n]$ are discrete functions defined in [0,M - 1], having a total of M points. Because the sets $\phi_{j^0,k}[n]_{k\in\mathbf{Z}}$ and $\psi_{(j,k)\in\mathbf{Z}^2,j\geq j^0}$ are orthogonal to each other. We can simply take the inner product to obtain the wavelet coefficients.
\begin{equation}
$W_\phi[j_0,k] = \frac{1}{\sqrt{M}}\sum_n f[n]\phi_{j^0,k}[n]$
\label{eqn_c4_dwt08}
\end{equation}
\begin{equation}
$W_\psi[j,k] = \frac{1}{\sqrt{M}}\sum_n f[n]\psi_{j,k}[n] \quad j \geq j_0$
\label{eqn_c4_dwt09}
\end{equation}
Equation (\ref{eqn_c4_dwt08}) is called approximation coefficient while (\ref{eqn_c4_dwt09}) is called detailed coefficients.
These two components show that the approximation coefficient, $W_\phi[j_0,k]$, models a low pass filter and the detailed coefficient,$W_\psi[j_0,k]$, models a high pass filter. It is possible to determine the approximation and detailed coefficients without the scaling and dilating parameters. The resulting coefficients, called the fast wavelet transform, are a convolution between the wavelet coefficients and a downsampled version of the next order coefficients. The fast wavelet transform was first postulated in \citep{mallat1989theory}.
$W_\phi[j,k] = h_\phi[-n] * W_\phi[j+1,n]\big|_{n=2k,k\geq 0}$
\label{eqn_c4_dwt10}
\end{equation}
\begin{equation}
$W\psi[j_0,k] = h_\psi[-n] * W_\phi[j+1,n]\big|_{n=2k,k\geq 0}$
\label{eqn_c4_dwt11}
\end{equation}
For analysis of the Haar wavelet and the derivation of equations (\ref{eqn_c4_dwt10} and \ref{eqn_c4_dwt11}) see appendix \ref{app01}.

Mel filter banks

The Fourier and wavelet transform are general means of extracting information from continuous signals using the frequency domain and in the case of the Wavelet transform using both time and frequency domain. The objective in machine learning

how ever is to extract patterns from the information derived. In this chapter in particular the Mel filter bank and the scatter transform are used as speech processing means of representing high dimension information from these signal processing techniques and representing them as lower dimension information without loss of information relevant for speech recognition.

Mel Frequency Cepstral Coefficients (MFCCs) are described in the works of \citep{davis1980comparison}. MFCCs have since then been the state-of-the-art feature engineering mechanism behind speech recognition research. Other speech features used in speech recognition include, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs),Perceptual Linear Prediction coefficients (PLP), \cite{mcloughlin2009applied, dines2010measuring}. The following paragraphs describe the six-stage processing pipeline used by MFCC in pattern recognition while attempting to model the physiologic process of hearing. The initial process is similar to the Short Time Fourier Transform (STFT).

After the initial sampling of the original continuous signal, the discrete samples are further windowed similar to what the STFT does. This is done so to minimise the dynamic temporal effect of sound that is constantly changing from one sound to the other. The strategy is to select a window where the sound is most stable before it transits to the next sound. This is found to be in the range of 20-40ms window. These time windows are also called frames. Frames shorter than this window usually do not contain sufficient samples for reliable sound estimation. At the same time frames having larger samples over longer time periods than this window contain overlapping sounds which also render poor estimation of the sounds contained.

The STFT on the windowed frames are then determined thus modelling the physiologic hearing process of the human ear. As the cochlea organ within the human ear receives sound by vibrating different areas for different sound frequencies, the STFT determines the various frequencies contained in the sound window. However, because the frequency resolution of the ear is lower than that of the STFT, in order for the resolution of the STFT frequencies match that of the ear similar frequencies are grouped in to bins called Mel filterbanks.

Not only does the ear have a low frequency resolution but this resolution seems to further decline and thus tapers off as the frequency increases. As a result, the first Mel filter is very narrow and estimates how the energy contained in the frequencies

near 0 Hertz. As the frequencies get higher however, our filters band becomes wider. Therefore the filter banks are not uniform in range and tend to increase in size until the cutoff frequency. The Mel filterbanks sum up periodogram bins energies for each bank. The Mel scale used in speech recognition specifies the band width of the filter banks and what frequencies to begin with.

The second consideration made the MFCC processors, is that sound energy is perceived by the human ear in a non-linear fashion. This means that variations is the loudness of a sound may not differ much when perceived if the sound is loud to begin with. Generally sound will be required to be 8 times as loud for a significant perception. The MFCC therefore takes the logarithm of energies contained in the Mel filter banks to estimate energy content of the Mel filters. Using a log scale also allows for a channel normalisation technique that employs cepstral mean subtraction.

The final step is to compute the DCT of the log filterbank energies. There are 2 main reasons this is performed. Because our filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal co-variance matrices can be used to model the features in e.g. a HMM classifier. But notice that only 12 of the 26 DCT coefficients are kept. This is because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade ASR performance, so we get a small improvement by dropping them.

Deep scattering spectrum

In the first two sections of this chapter, the Fourier Transform and the Wavelet transform were described. It was seen that while the Fourier transform estimates only the frequency component of the signal, the wavelet transform in addition approximates both frequency and time information in a finer manner than the advanced time-compensating forms of the Fourier Transform such as the Short Time Fourier transform (STFT). While the Mel filter bank is seen to create speech features based on the Mel filter bank which in turn is a form of the STFT, the scattering transform described in this section uses the wavelet transform as the basis for feature engineering of the sound signal for speech recognition. The following paragraphs introduce the wavelet modulus operator as the basic operation

used to derive the deep scattering spectrum described by \citep{anden2011multiscale, anden2014deep, zeghidour2016deep}.

Given a signal x, the wavelet sub-transforms Wx is determined as a convolution of the signal with a low-pass filter ϕ, and the signal with higher order complex frequency wavelets ψ^{λ_1}:

\begin{equation}
$$Wx = (x \star \phi(t), x \star \psi^{\lambda_1}(t))_{t \in \mathrm{R}, \lambda_1 \in \Lambda_1}$$
\label{eqn_c4_dss01}
\end{equation}

A modulus operator is then applied to the wavelet coefficients obtained from this initial convolution. This has an effect of eliminating complex phases as well as extracting envelopes for different resolutions

\begin{equation}
$$|W|x = (x \star \phi(t), |x \star \psi^{\lambda_1}(t)|)\, t \in \mathrm{R}, \lambda_1 \in \Lambda_1$$
label{eqn_c4_dss02}
end{equation}

It is seen that due to the time averaging filter $\phi(t)$, the sub-transform $S_0 x = x \star \phi(t)$ *is invariant to translation. This time-averaging filter attenuates the high frequency information, at the same time information lost when the high frequencies were filtered is recovered by the wavelet modulus coefficients* $|x \star \psi^{\lambda_1}|$. The end product of the wavelet modulus operation is the sub-transform $S_0 x = x \star \phi(t)$*, which is invariant to translation and the wavelet modulus* coefficients *which are not invariant to translation.*

It is possible to now derive the first layer of scattering coefficients by subjecting the wavelet modulus coefficients to another time averaging filter in an attempt to obtain invariant features from the high frequency coefficients.

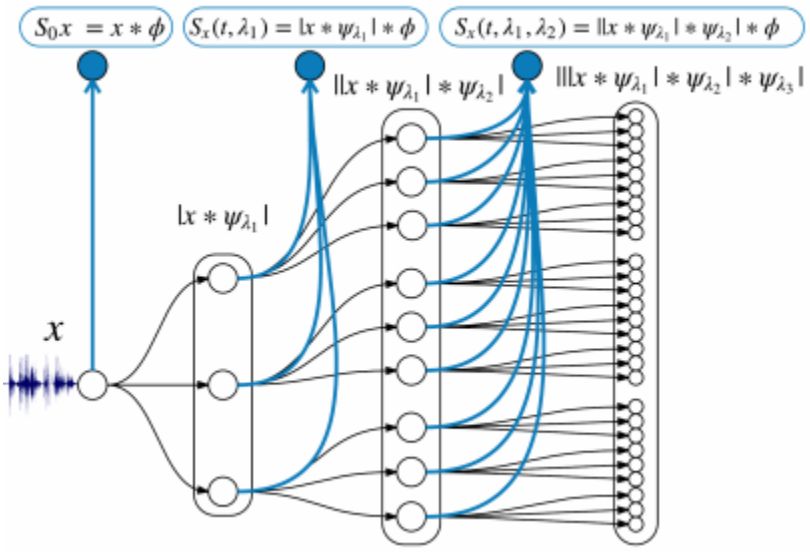
\begin{equation}
$$S_1 x(t, \lambda_1) = |x \star \psi^{\lambda_1}| \star \phi(t)$$
\label{eqn_c4_dss03}
\end{equation}

\citep{anden2014deep} demonstrates that the frequency resolution of the wavelets ψ^{λ_1} can be tuned to approximate the standard Mel-filter coefficients. By continuing this process of convolution through a higher frequency bank of wavelets ψ^{λ_2}, the

scattering transform is able to recover more invariant information lost when using traditional MFCCs.

\begin{equation}
$|W2||x \star \phi^{\lambda 1}| = (|x \star \psi^{\lambda 1}| \star \phi, ||x \star \psi^{\lambda 1}| \star \psi^{\lambda 2}|)_{\lambda 2 \in \Lambda 2}$
\label{eqn_c4_dss04})
\end{equation}

To complete the cycle and derive the next scattering layer, add a low pas filter ϕ *to average the wavelet modulus coefficients and rendering them invariant to translation.*
\begin{equation}
$S2x(t, \lambda 1, \lambda 2) = ||x \star \psi^{\lambda 1}| \star \psi^{\lambda 2}| \star \phi(t)$
\label{eqn_c4_dss05})
\end{equation}



Scattering network - 2 layers deep
\cite{zeghidour2016deep}

LaTeX parse error

The scattering transform obtained by successive computation of invariant features and retrieval of lost information is shown in the figure \ref{fig_4_3_scatter}. \citep{zeghidour2016deep, anden2011multiscale} confirm that for speech signals,

features can be entirely characterized by the first two layers of the spectrum. Therefore, this research employs the first two layers spectrum for speech representation. The scatter spectrum speech representation for these first two layers has the property of being invariant to translations and stable to deformations, and simultaneously retaining better information than MFCC features.

Wakirike Language Models

A language model for the Wakirike language is developed in this chapter. This model draws upon the premise that the grammar of a language is expressed in the character sequence pattern which is ultimately expressed in words and therefore the abstract grammar rules can be extracted and learned by a character-based RNN neural network.

Dataset Preparation

The Wakirike new testament bible served as the source of data for the deep neural network training. As there wasn't a soft or on-line copy of the Wakirike new testament bible readily available for use, the four gospels of the Wakirike new testament bible were quickly typed and duplicated once giving a complete corpus word size of about 165,180 words. This gracefully yielded a character count of about 1,113,820 characters void of punctuation characters. The dataset was then divided 1 in 10 parts for testing and the remaining 9 parts were used for training.

During data preparation, the dataset was first striped off all punctuation marks such that only characters and spaces are selected. Next, each character in the dataset was substituted with its equivalent Unicode numeric representation. Finally the numeric values were one-hot encoded deriving a sparse array of values having unique indexes set to one for each unique Unicode value and zero every where else. One-hot encoded array therefore, for each character input.

GRU Training

GRUs have been shown to give similar performance to regular LSTMs with a lighter system resource consumption foot print \cite{cho2014learning}. The internal network size of the GRU was 256 nodes and the number of GRUs representing each

time step in the recurrent input sequence was 30 GRUs; one GRU per time step. In addition, each unrolled sequence was layered 3 times. Therefore the unrolled 30-GRU-sequence long network was also 3-layers deep. Due to the multi-layered high-dimensional depth of this neural network, there was a tendency for the network to over fit the data, hence, a small learning rate of 0.001 was used. To further reduce the risk of over fitting the popular and effective dropout method for regularising deep neural networks kept at 80\% of activations while deactivating the rest.

GRU Output Language Generation
Once training of the neural network as described above is completed after several epochs or training cycles to an acceptable margin of error. It is possible to seed the network with an input character and the model samples from the top-N most likely candidates. We thus have instructed the language model developed to immanently construct its own sentences. The output language should therefore be intelligible similar to the training data.

In this work, the output language generated was used to generate a corpus that measured the perplexity and compared the result with the perplexity based on an n-gram model applied to the original training data. The results discussed below showed that the LSTM model generated a superior model compared to the n-gram model that better matched the training data.

Deep Learning Speech Models

Earlier in chapter one, deep learning was defined as a type of representational learning whereby different levels of complexity are captured in internal layer-wise encapsulations. It has also been noted that layer-wise stacking of neural and neural network type architectures such as the Restricted Boltzmann Machine (RBM) deep belief networks (DBMs) are used to implement such representations. In this chapter, the end-to-end Bi-directional Recurrent Neural Network model is described. Here, the development of the features using the deep scattering convolution network is first elaborated on. The model parameters and architecture is described and the decoding algorithm explained.

Deep Scattering Features

The fast wavelet transformed is derived in Chapter 4 from a low pass filter and a high pass filter. The speech features used in this research using a deep scattering network 2 layers deep was created using the wavelet modulus operator comprising a low pass filter and a band pass filter. Hyper parameters of the system included the window period for each sampled sub section, T; The Q-band value for the band pass filter and the number of wavelets J at each scattering layer for the total number of layers, M=2.

The matlab scatnet toolbox \citep{anden2014scatnet}, used to determine the scatter coefficient features for this research, provides optimal values for hyper parameters for audio signal processing into scatter features. In this regime the value for the hyper parameter T, the number of samples per window, = 512 samples per window. This approximates a window of 50 milliseconds for the audio signals sampled at 8000 Hz. For the first scattering layer the parameter, Q=8 and for the second scattering layer, the Q=1. Finally J is precalculated based on the value of T. These after scatnet processing, eventually produce a feature-vector 165 coefficients long. These feature vectors in turn are used as inputs to the bidirection neural network model whose architecture is explained in the succeeding sections.

Deep speech model

The core of the system is a bidirectional recurrent neural network (BRNN) trained to ingest scatter coefficients described in the previous section, in order to generate English text transcriptions. Let a single utterance x and label y be sampled from a training set $S = (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots$. Each utterance, $x^{(i)}$ is a time-series of length $T^{(i)}$ where every time-slice is a vector of audio features, $x^{(i)}t$ *where* $t = 1, \ldots, T^{(i)}$. *We use spectrograms as our features; so* $x^{(i)}t, p$ denotes the power of the p-th frequency bin in the audio frame at time t. The goal of our BRNN is to convert an input sequence x into a sequence of character probabilities for the transcription y, with $\hat{y}_t = \mathrm{P}(c_t \mid x)$, where $c_t \in a, b, c, \ldots, z, space, apostrophe, blank$. (The significance of $blank$ will be explained below.)

Model Architecture

Our BRNN model is composed of 5 layers of hidden units. For an input x, the hidden units at layer l are denoted $h^{(l)}$ with the convention that $h^{(0)}$ is the input. The first three layers are not recurrent. For the first layer, at each time t, the output depends on the spectrogram frame x_t along with a context of C frames on each side. (We

typically use $C \in 5, 7, 9$ for our experiments.) The remaining non-recurrent layers operate on independent data for each time step. Thus, for each time t, the first 3 layers are computed by:

\begin{equation}
$h_t^{(l)} = g(W^{(l)} h_t^{(l-1)} + b^{(l)})$
\label{eqn_c6_brnn01}
\end{equation}

where $g(z) = \min \max 0, z, 20$ is the clipped rectified-linear (ReLu) activation function and $W^{(l)}$, $b^{(l)}$ are the weight matrix and bias parameters for layer l. The fourth layer is a bidirectional recurrent layer[1]. This layer includes two sets of hidden units: a set with forward recurrence, $h^{(f)}$, and a set with backward recurrence $h^{(b)}$:

\begin{equation}
$h^{(f)} t = g(W^{(4)} h_t^{(3)} + W r^{(f)} h^{(f)} t - 1 + b^{(4)}$
\label{eqn_c6_brnn02}
\end{equation}

\begin{equation}
$h^{(b)} t = g(W^{(4)} h_t^{(3)} + W r^{(b)} h^{(b)} t + 1 + b^{(4)})$
\label{eqn_c6_brnn03}
\end{equation}

Note that $h^{(f)}$ must be computed sequentially from $t = 1$ to $t = T^{(i)}$ for the i-th utterance, while
the units $h^{(b)}$ must be computed sequentially in reverse from $t = T^{(i)}$ to $t = 1$.
The fifth (non-recurrent) layer takes both the forward and backward units as inputs

\begin{equation}
$h^{(5)} = g(W^{(5)} h^{(4)} + b^{(5)})$
\label{eqn_c6_brnn04}
\end{equation}

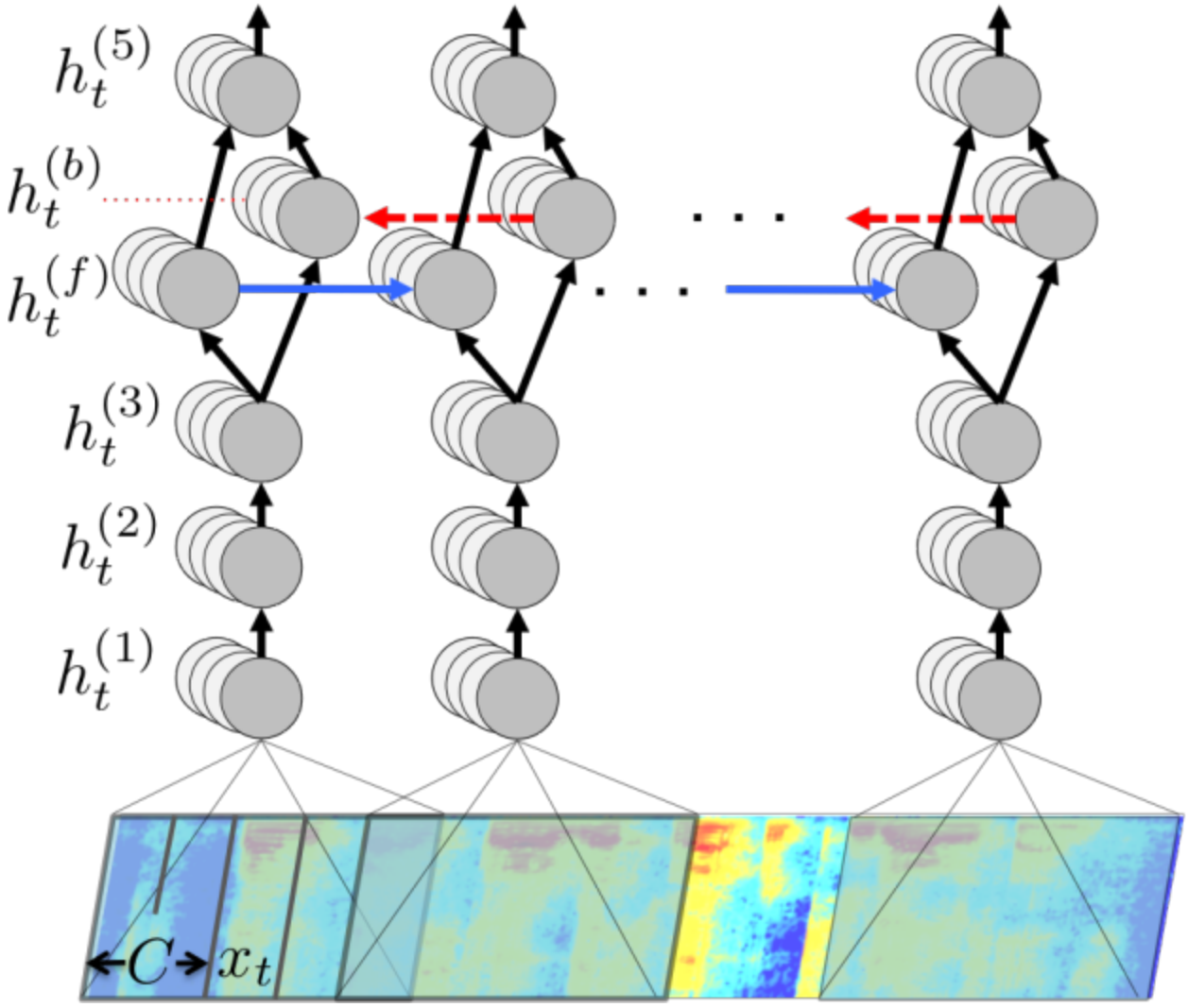
where $h^{(4)} = h^{(f)} + h^{(b)}$. The output layer is a standard softmax function that yields the predicted character probabilities for each time slice t and character k in the alphabet:

\begin{equation}
$h^{(6)} t, k = \hat{y} t, k \equiv \mathrm{P}(c_t = k \mid x) = \frac{\exp\left((W^{(6)} h_t^{(5)})_k + b_k^{(6)}\right)}{\sum_j \exp\left((W^{(6)} h_t^{(5)})_j + b_j^{(6)}\right)}$
\label{eqn_c6_brnn05}
\end{equation}

Here $b_k^{(6)}$ denotes the k-th bias and $(W^{(6)} h_t^{(5)})_k$ the k-th element of the matrix product.

Once we have computed a prediction for $\mathrm{P}(c_t = k \mid x)$, we compute the CTC loss[2] $\mathrm{L}(\hat{y}, y)$ to measure the error in prediction. During training, we can evaluate the gradient $\nabla \mathrm{L}(\hat{y}, y)$ with respect to the network outputs given the ground-truth character sequence y. From this point, computing the gradient with respect to all of the model parameters may be done via back-propagation through the rest of the network. We use the Adam method for training[3].

The complete BRNN model is illustrated in the figure below.



DeepSpeech BRNN

5 hidden layers 1824 hidden units, 20.9M free parameters

Weights are initialised from a uniform random distribution scaled by the weight matrix input and output layer size (Glorot et al 2011) Nesterov accelerated gradient

optimisation algorithm as described in Sutskever et al. (2013) with initial learning rate 10-5, and maximum momentum 0.95.

After each full pass through the training set we divide the learning rate by 1.2 to ensure the overall learning rate decreases over time. We train the network for a total of 20 passes over the training set, which takes about 96 hours using our Python GPU implementation. For decoding with prefix search we use a beam size of 200 and cross-validate with a held-out set to find a good setting of the parameters α and β. Table 1 shows word and character error rates for multiple approaches to decoding with this trained BRDNN

CTC decoder

Assuming an input of length T, the output of the neural network will be $p(c; x_t)$ for $t = 1, ..., T$. Again $p(c; x_t)$ is a distribution over possible characters in alphabet Σ, which includes the blank symbol, given audio input x_t. In order to recover a character string from the output of the neural network, as a first approximation, we take the argmax at each time step. Let $S = (s_1, ..., s_T)$ be the character sequence where $s_t = arg\max_{c \in \Sigma} p(c; x_t)$. The sequence S is mapped to a transcription by collapsing repeat characters and removing blanks. This gives a sequence which can be scored against reference transcription using both CER and WER.

The first approximation lacks the ability to include the constraint of either a lexicon or language model. We propose a generic algorithm which is capable of incorporating such constraints. Taking X to be acoustic input of time T, we seek a transcription W which maximises the probability.

\begin{equation}
$p_{net}(W; X)p_{lm}(W)$
\label{eqn_c6_brnn06}
\end{equation}

Here the overall probability of the transcription is modeled as the product of two factors: pnet given by the network and plm given by the language model prior. In practice the prior plm(W), when given by an n-gram language model, is too constraining and thus we down-weight it and include a word insertion penalty or bonus as

\begin{equation}
$p_{net}(W; X)p_{lm}(W)^{\alpha} \mid W \mid^{\beta}$

\label{eqn_c6_brnn07}
\end{equation}

Algorithm 1 attempts to find a word string W which maximizes equation 8. The algorithm maintains two separate probabilities for each prefix, $p_b(\ell; x_{1:t})$ and $p_{nb}(\ell; x_{1:t})$. Respectively, these are the probability of the prefix ending in blank or not ending in blank given the first t time steps of the audio input X.

Algorithm 1 Prefix Beam Search: The algorithm initializes the previous set of prefixes A_{prev} to the empty string. For each time step and every prefix ℓ currently in A_{prev}, we propose adding a character from the alphabet Σ to the prefix. If the character is a blank, we do not extend the prefix. If the character is a space, we incorporate the language model constraint. Otherwise we extend the prefix and incorporate the output of the network. All new active prefixes are added to A_{next}. We then set A_{prev} to include only the k most probable prefixes of A_{next}. The output is the 1 most probable transcript, although the this can easily be extended to return an n-best list.

The sets A_{prev} and A_{next} maintain a list of active prefixes at the previous time step and a proposed prefixes at the next time step respectively. Note that the size of A_{prev} is never larger than the beam width k. The overall probability of a prefix is the product of a word insertion term and the sum of the blank and non-black ending probabilities.
\begin{equation}
$p(\ell; x_{1:t}) = (p_b(\ell; x_{1:6}) + p_{nb}(\ell; x_{1:t}))\,|\,W(\ell)\,|^{\beta}$
\label{eqn_c6_brnn08}
\end{equation}

Where $W(\ell)$ is a set of words in the sequence . When taking the k most probable prefixes of A_{next}, we sort each prefix using the probability given in equation (\ref{eqn_c6_brnn08}).
The variable ℓ_{end} is the last character in the label sequence ℓ. The function $W(\cdot)$, which converts ℓ into a string of words, segments the sequence ℓ at each space character and truncates any characters trailing the last space.

We incorporate a lexicon or language model constraint by including the probability $p(W(\ell^{+})\,|\,W(\ell))$ whenever the algorithm proposes appending a space character to ℓ. By setting $p(W(\ell^{+})\,|\,W(\ell))$ to 1 if the last word of $W(\ell^{+})$ is in the lexicon and 0 otherwise, the probability acts a a constraint forcing all character strings to consists

of words only in the lexicon. Furthermore, $p(W(\ell^{+}) \mid W(\ell))$ can represent an n-gram language model by considering only the last n-1 words in $W(\ell)$.

Conclusion and Discussion of Results

Future Direction

The advancement of machine learning has a direct impact on the development of more efficient speech recognition algorithms and at the same time the advancement of speech recognition helps with the improvement of machine learning algorithms, as in general, the methods used in machine learning usually are directly transferable to speech processing and vice-versa. This mutual relationship implies that speech recognition is a blossoming research field because there is a tremendous amount of work being done in the machine learning community. Particularly in the area of deep learning and neural networks, there is quite a vast array of neural network solutions that have been applied or are yet to be applied to speech recognition. Two models worthy of mentioning are Generative Adversarial Networks (GANs) and Attention-based models.

Generative adversarial networks (GAN)

GANs consists of two Networks working as adversaries to one another. The first being a generative network generates content. The second network is a discriminative network to determine the accuracy of the first generative network. Hence the generative network is generating output less distinguishable for the discriminator while the discriminator uses output from the generator to improve it's ability to discriminate output from the generator with the original data.

GAN networks can have application where the generative network consists of a speech synthesis network and the discriminating network is a speech recogniser. However successive training of these two networks from a data-resource perspective would require an immense amount of data resources for expected performances.

Attention-based Models

The objective of attention-based networks highlighted by \cite{vaswani2017attention} is to reduce sequential computation while attaining hidden representation across arbitrary lengths of sequential input. Mechanisms which have been deployed to achieve this includes a combination of convolutional and recurrent schemes \cite{kaiser2016can,kalchbrenner2016neural, gehring2017convolutional}. \cite{vaswani2017attention} introduces a transduction model known as a Transformer based on self attention network with the ability to compute long term dependencies while eliminating sequence aligned RNN and convolutional architectures.

Self attention is a network that intrinsically reduces the need for intensive resource training. \cite{vaswani2017attention} reports that state of the art BLEU score of 41.0 having used a small fraction of training resources. While GANs might not be attractive for low resource speech recognition. They still remain an important tool for verification of the output of other networks at the same time self attention networks can help to the resource needs of GANs when applied to a GAN.

As a further to this thesis, these networks are likely candidates for network training using scatter features as input discriminatory functions. Attention based networks as a means reduce training resources required while GANs can be used as a means to generate training data.

Appendices

Appendix 1 Mother Wavelet examples

Haar Wavelets

Daubechies

Based on these equations, Daubechies [9], designed a type of wavelet for a given vanishing moment p and found the minimum size discrete filter. The conclusion is that if we want the wavelet function with p vanishing moments, the minimum filter is 2p. The derivation starts from (5.17), rewrite as

- - (5.19)
- The absolute-square of this function is
- - (5.20)

- The last step makes $$P\left(sin^2\frac{\omega}{2}\right)=|R(e^{j\omega})|^2$$. Recall (5.6), we can determine the form of P(x). Let $$y=sin^2\frac{\omega}{2}$$, we have
- - (5.21)
- A theorem in algebra, called Bezout theorem, can solve this equation. The unique solution is
- -- (5.22)
- The polynomial P(y) is the minimum degree polynomial satisfying equation (5.21). Once we have P(y), the polynomial $$R(e^{j\omega})$$ can be derived. First we decompose $$R(e^{j\omega})$$ according to its roots
- - (5.23)
- Let , the relation between P and R is
- - (5.24)
- By solving the roots of $$P\left(\frac{2-z-z^{-1}}{4}\right)=0$$, we have the roots of $$R, {a_k,1/a_k}_{k=0,1,\dots,m}$$ and $$r_0=2^{p-1}$$ Usually, we choose $$a_k$$ lies in the unit circle to have minimum phase filter.

Taking p=2 for an example, the obtained polynomial P(y) is

- - (5.25)
- - (5.26)

The roots are $$2+\sqrt{3}$$ and $$2-\sqrt{3}$$. After factorisation, we have the lowpass filter to be

(5.27)

The discrete-time domain representation is

(5.28)

The result is the minimum size filter with 2 vanishing moments and the corresponding filter size is 4. Recall the conclusion mentioned above, the filter size is two times the vanishing moment. Higher order Daubechies wavelets are derived at similar way.

Symlets

Take a look at the discrete filters and the scaling/wavelet functions of Daubechies wavelets. These functions are far from symmetry. That's because Daubechies wavelets select the minimum phase square root such that the energy concentrates near the starting point of their support. Symmlets select other set of roots to have closer symmetry but with linear complex phase.

Coiflets

For an application in numerical analysis, Coifman asked Daubechies [9] to construct a family of waveletsψthat have p vanishing moments, minimum-size support and

- - (5.29)

- - (5.30)
- The equation above can be taken as some requirement about vanishing moments of the scaling function. The resulting coiflets has a support of size3 p–1.

Appendix 2: Matlab Code to generate scattering features

```
function scatter()
    %UNTITLED Summary of this function goes here
    %   Detailed explanation goes here
    T = readtable('data/cv-valid-dev.xlsx','ReadRowNames',tru
e);
    F=table2cell(T(:,{'scatterc','wav_filename'}));
    all_files=size(F,1);
    ofm='hh:mm:ss';
    ifm='dd-mmm-yy HH:MM:SS.FFF';
    tic;
    for i = 1:all_files
      wav_file=strjoin(F(i,2));
      dss_file=strjoin(F(i,1));
      if exist(wav_file,'file')>0
          if exist(dss_file,'file')==0
            st = transpose(scatter_audio(wav_file));
            csvwrite(dss_file,st);
          end
      else
          fprintf('\nNot found:%s',wav_file);
      end

      pg=i/all_files*100;
      ts=datestr(now,ifm);
      tv=toc;
```

```
      d=duration(seconds(tv),'Format',ofm);
      pc=(all_files/i*tv)-tv;
      eta=duration(seconds(pc),'Format',ofm);

      if mod(i,500)==0 || i==1 || i==10 || i==100
          fileID = fopen('log/dss180625.log','w+');
          s=sprintf('\n%s: processing file  %s',ts,wav_file);
          fprintf(fileID,'%s',s);
          fprintf('%s',s);
          s=sprintf('\n%s : processing %d of %d files  %3.2f%%
complete.. time elapsed = %s, eta = %s',ts,i,all_files,pg,d,et
a);
          fprintf(fileID,'%s',s);
          fprintf('%s',s);
          fclose(fileID);
      end
    end
end



function st= scatter_audio(inputArg1)
    y=audioread(inputArg1);
    N=length(y);
    T=2^9;
    filt_opt=default_filter_options('audio',T);
    Wop=wavelet_factory_1d(N,filt_opt);
    S=scat(y,Wop);
    S=renorm_scat(S);
    S=log_scat(S);
    st=format_scat(S);
```

```
end
```