Deep Scattering End-to-End Speech Recognition

# Abstract

# Introduction

This work explores the prospects of deep recurrent end-to-end architectures applied to speech recognition. Complementary aspects of developing speech recognition systems are eliminated by focusing on end-to-end speech units as a two-step process requiring a Connectionist Temporal Classification (CTC)\cite{graves2006connectionist} model and Language Model (LM) rather than a three-step process requiring an Acoustic model(AM), LM and phonetic dictionary. A two-step process rather than a three-step process is particularly desirable for low resource languages as less effort is required developing fewer simplified models.

# Previous Deep Scattering Research

The CTC models currently have been developed using standard MFCC features. The model developed in this work employs a deep scattering features which compared to MFCC posses greater number of features being of a higher dimension (152 compared to 39). These deep scattering vectors have been shown to perform well on music genre classification and TIMIT phone recognition[(Zeghidour et al. 2018)](https://paperpile.com/c/U9RF6T/lX36).

# The Scattering Transform layer

## Mel Filter banks

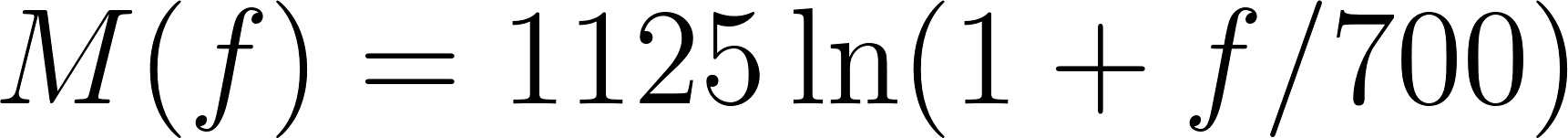
The Fourier and wavelet transform are general means of extracting information from continuous signals using the frequency domain and in the case of the Wavelet transform using both time and frequency domain. The objective in machine learning, however, is to extract patterns from the derived information. In this chapter, in particular, the Mel filter bank and the scatter transform are elaborated on as speech feature extractors. They process high dimensional information obtained from the Fourier and wavelet transform signal processing techniques and reducing the information obtained as lower dimension features. All this aimed towards lossless encoding of speech signals relevant for speech recognition.

The Mel filter banks form the basis of the Mel Frequency Cepstral Coefficients (MFCCs) described by \citep{davis1980comparison}. MFCCs are state-of-the-art speech feature engineering drivers behind automatic speech recognition acoustic models. Other common speech features used in speech recognition include, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs),Perceptual Linear Prediction coefficients (PLP), \citep{mcloughlin2009applied, dines2010measuring}. The following paragraphs describe how the mel filters are derived.

The Mel scale as described by \cite{stevens1937scale} is a perceptual scale which measures sound frequencies as perceived by human subjects equidistant from a sound source as compared to the actual frequency. This scale is nonlinear as the human ear processes sound non-linearly both in frequency as well as amplitude.

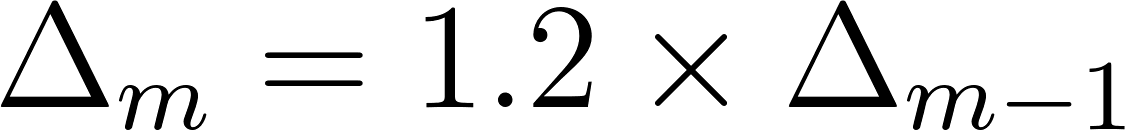
For the case of frequency, the human ear can discriminate lower frequencies more accurately than the higher frequencies. The Mel scale model this behaviour by utilising frequency bins. The frequency bin ranges are narrow at low frequencies and become wide in higher frequencies. In the case of the speech signal amplitude, a similar process is observed, where the ear discriminates softer sounds better than louder sounds. Generally, sound will be required to be 8 times as loud for significant perception by the ear. While the mel scales handle the frequency non-linearity in the speech signal, the signal amplitude is linearised during feature extraction by taking the log of the power spectrum of the signal also known as the cepstral values. Furthermore, using a log scale also allows for a channel normalisation technique that employs cepstral mean subtraction \citep{becchetti1999behaviour}.

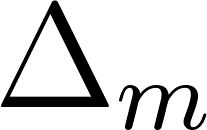
The minimum frequency number of bins used for the Mel scale is 26 bins. In order to determine the frequency ranges we use the following formula to convert between the Mel scale and the regular frequency scale:

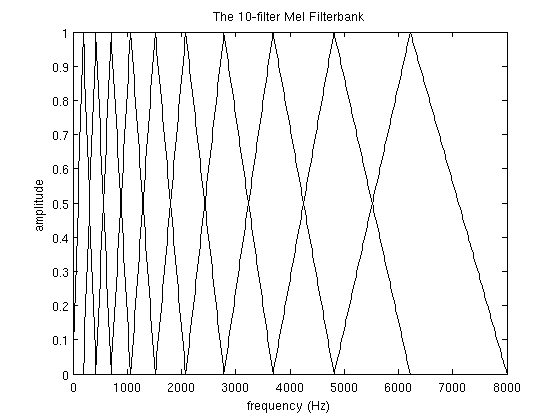
[](https://www.codecogs.com/eqnedit.php?latex=M(f)%3D1125%5Cln(1%2Bf%2F700)%0) - - - (1)

[](https://www.codecogs.com/eqnedit.php?latex=M%5E%7B-1%7D(m)%3D700%5Cexp(m%2F1125)-1%0) - - - (2)

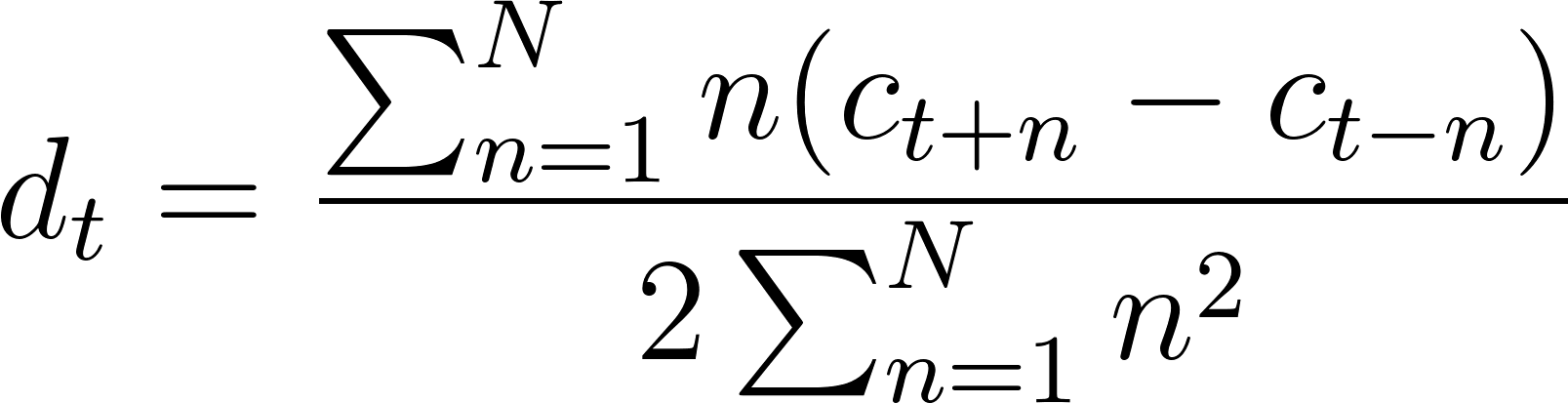
A simple approximation for the Mel scale is obtained by applying linear scale for the first ten filters and for the first 1kHz of the speech frequency range then applying the following formula for the rest \citep{becchetti1999behaviour}:

[](https://www.codecogs.com/eqnedit.php?latex=%5CDelta_m%3D1.2%5Ctimes%20%5CDelta_%7Bm-1%7D%0) - - - (3)

where m is the frequency bin index and [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta_m%0) is the frequency range between the start and end frequencies for the m-th bin. The resulting filters are overlapping filters shown in the figure below.



For speech recognition, we compute a statistical value or coefficient for each Mel frequency bin from the inverse discrete fourier transform (IDFT) of the Mel filters. The coefficients are also concatenated with their delta and delta-delta values. The delta and delta-delta values are determined from the following equation:

[](https://www.codecogs.com/eqnedit.php?latex=d_t%3D%5Cfrac%7B%5Csum_%7Bn%3D1%7D%5ENn(c_%7Bt%2Bn%7D-c_%7Bt-n%7D)%7D%7B2%5Csum_%7Bn%3D1%7D%5ENn%5E2%7D%0) - - - (4)

where [](https://www.codecogs.com/eqnedit.php?latex=c_x%0) is the x-th coefficient and 2n is the delta range which is usually 2-4. The delta values are first order derived coefficients obtained from the original Mel filter coefficients while the delta-delta values are second-order derived coefficients obtained from the first-order derived delta coefficients.

There are two reasons for obtaining the IDFT from the filter banks. The first is that since the bins use overlapping windows, the filter bin outputs tend to be correlated and obtaining the IDFT helps to decorrelate the outputs. Secondly, decorelated signals optimise algorithm computation efficiency involving matrix operations such that rather than using full covariance matrix, it is much simpler to compute the matrix operations from the matrix diagonal. Also note that for cepstral values obtained from taking the log of the power power spectrum, the discrete cosine transform (DCT) is used to obtain the IDFT. This is as a result of the cepstral values being real and symmetric\citep{gales2008application}.

As an attempt for MFCCs to incorporate dynamic frequency changes of the signal, the deltas and the delta-deltas are obtained from the coefficient computation in equation (4). However, it is worthy to note that only the first 13 of the coefficients and the resulting dynamic coefficients are used as speech features as it is observed that higher frequency dynamic coefficients rather degrade ASR performance \citep{gales2008application}.

# CTC Loss Algorithm \label{c3\_ctc}

The term CTC stands for Connectionist Temporal classification. This algorithm was designed to solve the problem of fuzzy alignment between the source input data and the output classification desired from the machine learning system. This type of fuzzy alignment is observed in speech recognition systems since the same speech in either the same individual or different individuals will have different signal forms. This is a many to one relationship between the input signal and the output classification that is also dependent on the style of speaking at the moment when the utterance is said. Unlike hybrid DNN-HMM networks the CTC algorithm deploys an end-to-end framework that models all aspects of the input sequence in a single neural network, therefore discarding the need for an HMM interpretation of the input sequence. In addition, the CTC method does not require pre-segmented training data at the same time output classification is made independent of post-processing.

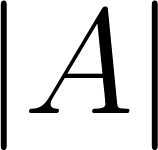
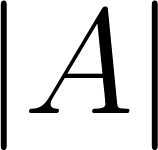
CTC works by making predictions at any point in the input sequence. For the case of speech modelling, CTC makes a character prediction for every time step of the raw audio input speech signal. Although this initially seems counter intuitive, this method models the many to one relationship seen in the fuzzy audio speech to text alignment.

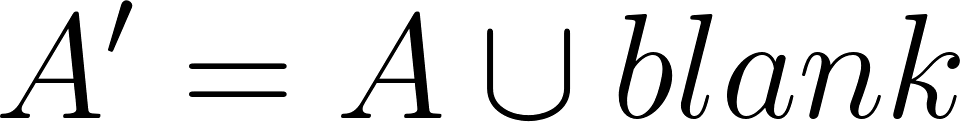
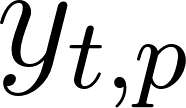
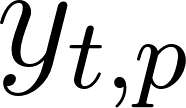
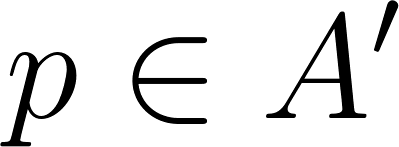
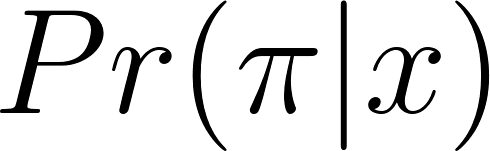
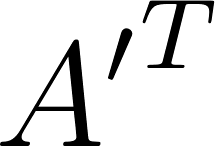
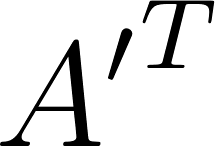
For hybrid DNN-HMM systems, speech or more accurately, acoustic models, require separate training of targets for every time-slice in the input sequence. Secondly, and as a consequence of this, it becomes necessary to segment the audio sequence, in order to provide targets for every time-slice. A third consequence is the limitation of DNNs previously discussed. As the DNN network only outputs local classifications, global aspects such as the likelihood of two consecutive labels appearing together cannot be directly modelled. Without an external model, usually in the form of a language model, the hybrid speech model will significantly degrade performance.

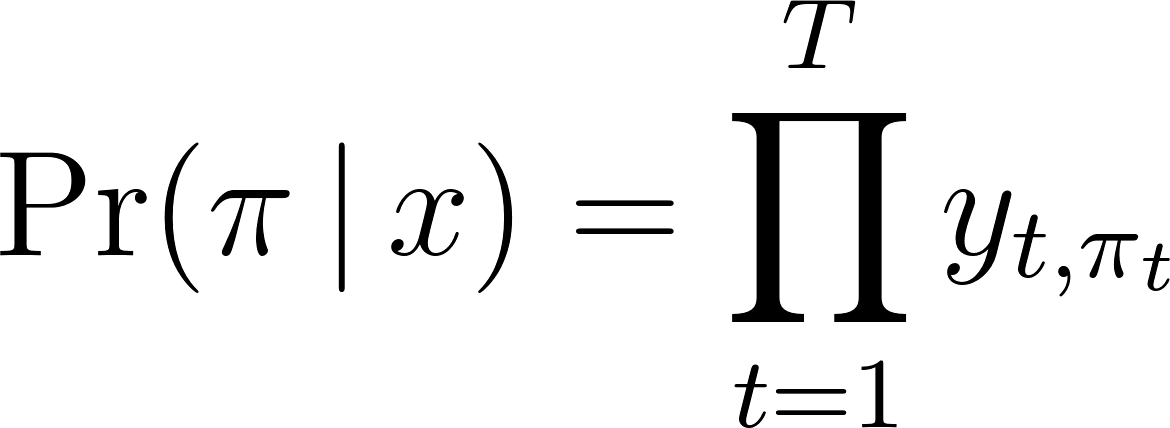
In the CTC case, so long as the overall sequence of labels is correct the network can be optimised to correct the temporal or fuzzy alignments. Since this many to one fuzzy alignment is simultaneously modelled in CTC, then there is no need for pre-segmented data. At the same time, CTC computes probabilities of complete label sequences, hence external post-processing required by hybrid models is eliminated.

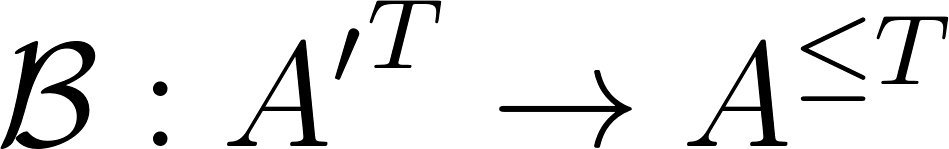
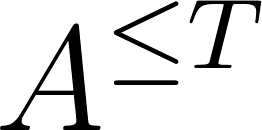
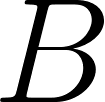
Similar to the HMM sequence model, the CTC algorithm is a sequence model that predicts the next label in a sequence as a cumulative of previous sequences. This section develops the CTC loss function borrowing concepts used in HMM models such as the forward backward algorithm as outlined in \citep{graves2006connectionist}. In the following paragraph we introduce terminology associated with the CTC loss function.

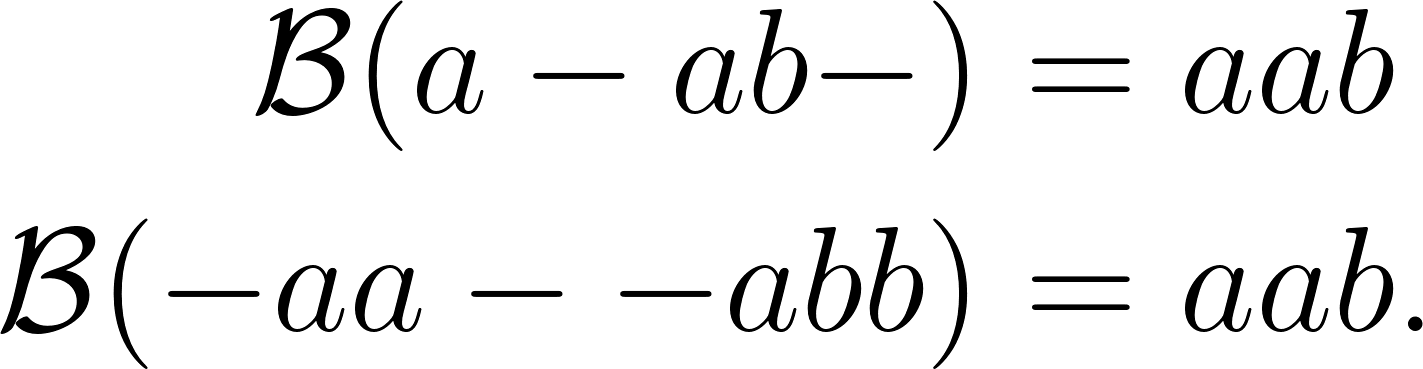
Given two symbols [](https://www.codecogs.com/eqnedit.php?latex=A%0) and [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0) such that [](https://www.codecogs.com/eqnedit.php?latex=A%0) has a many to one relationship with [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0), signifying the temporal nature of the classification. The symbol [](https://www.codecogs.com/eqnedit.php?latex=A%0) represents an alphabet from which a sequence of the output classifications are drawn from. This CTC output consists of a soft-max layer in a BiRNN (bidirectional recurrent neural network).

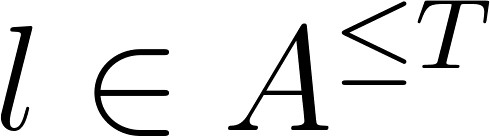
This output models the probability distribution of a complete sequence of arbitrary length [](https://www.codecogs.com/eqnedit.php?latex=%7CA%7C%0) over all possible labels in [](https://www.codecogs.com/eqnedit.php?latex=A%0) from activations within [](https://www.codecogs.com/eqnedit.php?latex=%7CA%7C%0). An extra activation is given to represent the probability of outputting a [](https://www.codecogs.com/eqnedit.php?latex=blank%0), or no label. At each time-step leading up to the final step, the probability distribution estimated as distribution over all possible label sequences of length leading up to that of the input sequence.

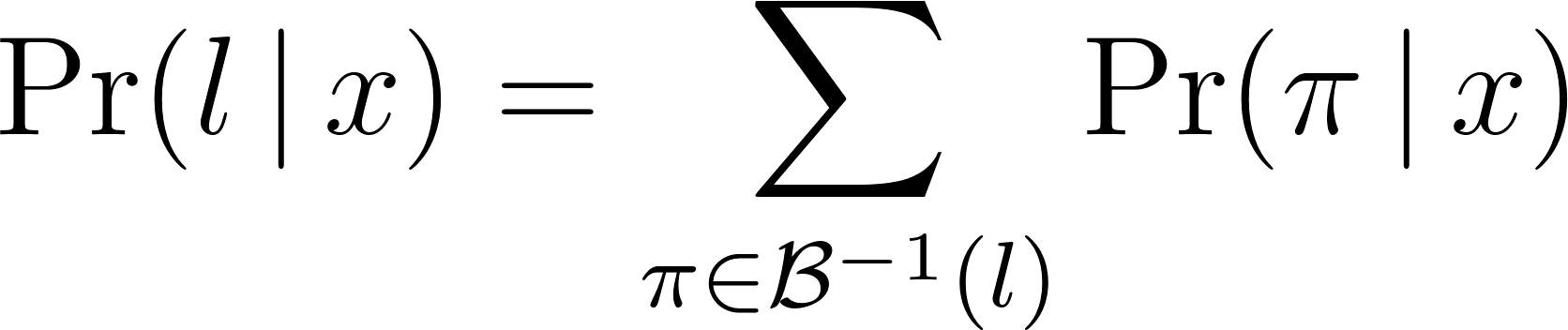
It is now possible to define the extended alphabet [](https://www.codecogs.com/eqnedit.php?latex=A'%20%3D%20A%20%5Ccup%20%7Bblank%7D%0), also, [](https://www.codecogs.com/eqnedit.php?latex=y_%7Bt%2Cp%7D%0) as the activation of network output [](https://www.codecogs.com/eqnedit.php?latex=p%0) at time [](https://www.codecogs.com/eqnedit.php?latex=t%0). Therefore [](https://www.codecogs.com/eqnedit.php?latex=y_%7Bt%2Cp%7D%0) is the probability that the network will output element [](https://www.codecogs.com/eqnedit.php?latex=p%20%5Cin%20A'%0) at time [](https://www.codecogs.com/eqnedit.php?latex=t%0) given that [](https://www.codecogs.com/eqnedit.php?latex=x%0) is the input sequence of length [](https://www.codecogs.com/eqnedit.php?latex=T%0). The distribution sought after [](https://www.codecogs.com/eqnedit.php?latex=Pr(%5Cpi%7Cx)%0), is the conditionally-independent distribution over the subset [](https://www.codecogs.com/eqnedit.php?latex=%7BA'%7D%5ET%0) where[](https://www.codecogs.com/eqnedit.php?latex=%7BA'%7D%5ET%0) is a set of length [](https://www.codecogs.com/eqnedit.php?latex=T%0) comprising a sequence of symbols each of which belong to the set [](https://www.codecogs.com/eqnedit.php?latex=%7BA'%7D%0) such that

[](https://www.codecogs.com/eqnedit.php?latex=%5CPr(%20%5Cpi%20%5C%2C%20%7C%20%5C%2C%20x%20)%20%3D%20%5Cprod_%7Bt%3D1%7D%5E%7BT%7D%20y_%7Bt%2C%5Cpi_t%7D%0) - - - (1)

From equation \ref{eqn\_ch6\_ctc02}, it is now possible to define the many-to-one mapping [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%20%3A%20%7BA'%7D%5ET%20%5Crightarrow%20A%5E%7B%5Cle%20T%7D%0), from the set of paths [](https://www.codecogs.com/eqnedit.php?latex=A'%5ET%0) onto the set [](https://www.codecogs.com/eqnedit.php?latex=A%5E%7B%5Cle%20T%7D%0) of possible labellings of [](https://www.codecogs.com/eqnedit.php?latex=x%0). [](https://www.codecogs.com/eqnedit.php?latex=%5Cmatcal%7BB%7D%0) then becomes a sequence of symbols with length less than or equal to [](https://www.codecogs.com/eqnedit.php?latex=T%0) over [](https://www.codecogs.com/eqnedit.php?latex=A%0). Note that [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0) is a set containing sequential symbols belonging to the set [](https://www.codecogs.com/eqnedit.php?latex=A%0) and not. [](https://www.codecogs.com/eqnedit.php?latex=A'%0) because there is no blank symbol in [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0).This is achieved when first take out all repeated labels and then take out all the blanks from the sequence [](https://www.codecogs.com/eqnedit.php?latex=A'%5ET%0). For example,

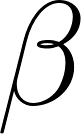
[](https://www.codecogs.com/eqnedit.php?latex=%5Cbegin%7Baligned%7D%5Cmathcal%7BB%7D(a%20-%20ab-)%20%26%3D%20aab%20%5C%5C%20%5Cmathcal%7BB%7D(-aa%20-%20-abb)%20%26%3D%20aab.%5Cend%7Baligned%7D%0) - - - (2)

The mapping obtained by [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0) is equivalent to when the output switches from not predicting a new symbol to predicting a symbol or from predicting one symbol to another symbol assuming this was also possible. Intuitively, the probability of [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0) which is the labelling of [](https://www.codecogs.com/eqnedit.php?latex=l%20%5Cin%20A%5E%7B%5Cle%20T%7D%0) being a many to one of [](https://www.codecogs.com/eqnedit.php?latex=A'%5ET%0) is determined by summing over all the paths in [](https://www.codecogs.com/eqnedit.php?latex=A'%5ET%0) mapped onto it by [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0). Thus:

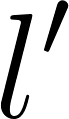
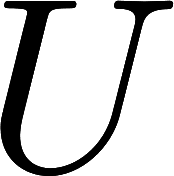
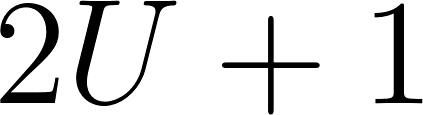
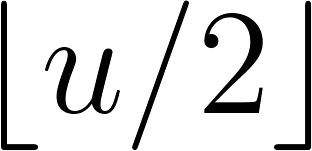
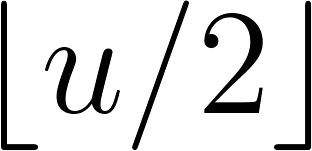
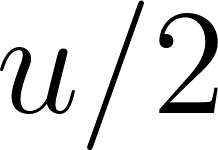
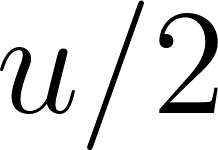
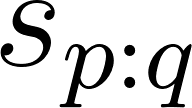
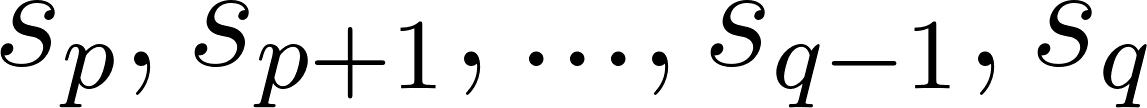
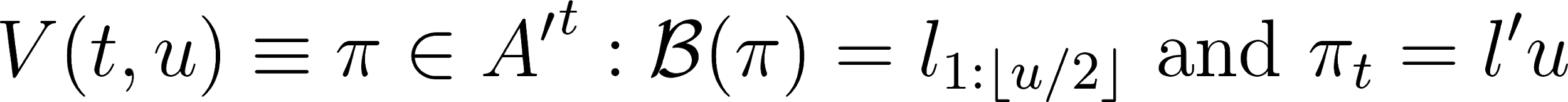
[](https://www.codecogs.com/eqnedit.php?latex=%5CPr(%20l%20%5C%2C%20%7C%20%5C%2C%20x)%20%3D%20%5Csum_%7B%5Cpi%20%5Cin%20%5Cmathcal%7BB%7D%5E%7B-1%7D(l)%7D%20%5CPr(%20%5Cpi%20%5C%2C%20%7C%20%5C%2C%20x)%0) - - - (3)

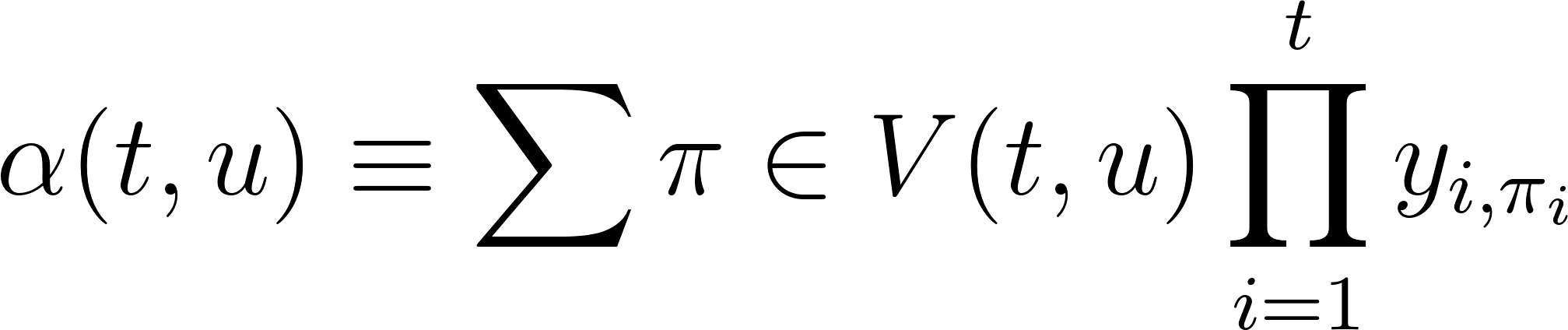
This mapping makes CTC robust to unsegmented data as it predicts all the labels where they occur and later the ‘collapsed’ sequence will be extended over the approximate period where the previous extended sequence occurred thus aligning labels to input sequences on-the-fly without knowing in advance where label to input sequence alignments occur.

## Forward-backward algorithm

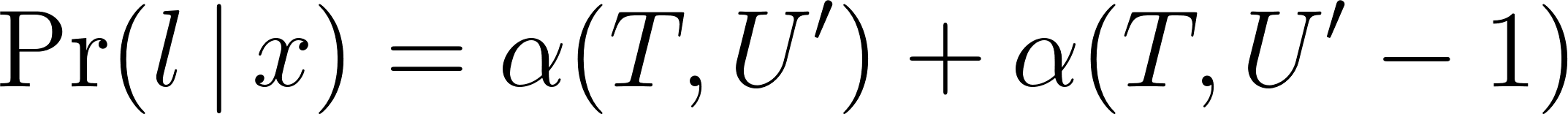
The forward-backward algorithm is used to estimate the probability of a point in the sequence as the product of all point leading up to that point from the initial state, the forward variable ([](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%0)), multiplied by the probability of all the points from that state to the end of the sequence, the backward variable ([](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta%0)).

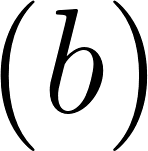
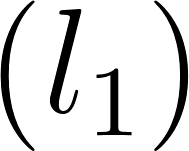
The difference between this estimation and that determined from equation (\ref{eqn\_c3\_ct03}) is the fact that the forward-backward algorithm converts equation (\ref{eqn\_c3\_ct03}) into a form that is both recursive as well as reduces the computational complexity from an otherwise intractable computation to one that is readily computable.

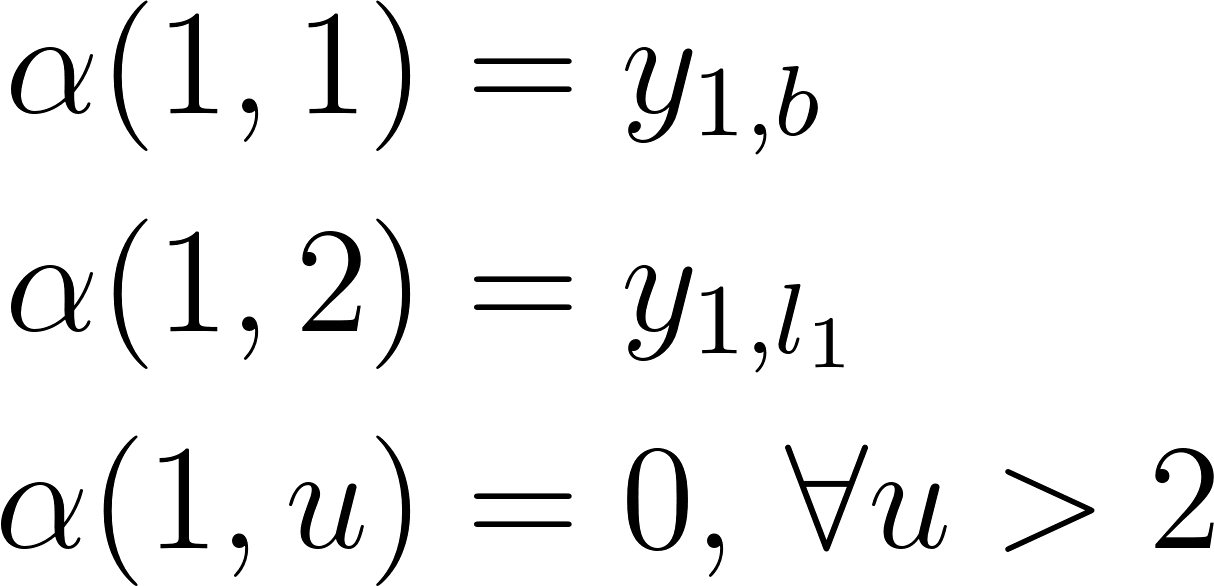
With CTC, consider a modified "label sequence" [](https://www.codecogs.com/eqnedit.php?latex=l'%0), that caters for blank characters in between regular ones [](https://www.codecogs.com/eqnedit.php?latex=l%0), as defined in [](https://www.codecogs.com/eqnedit.php?latex=A'%0). Thus, if [](http://www.texrendr.com/?eqn=U%0) is defined as the length of [](https://www.codecogs.com/eqnedit.php?latex=l%0). Then [](https://www.codecogs.com/eqnedit.php?latex=U'%0) is of length [](https://www.codecogs.com/eqnedit.php?latex=2U%20%2B%201%0). CTC therefore integrates probability distributions of transitions between blank and non-blank labels at the same time CTC calculates those transition occurring between pairs of distinct non-blank labels. The forward variable [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%0) now becomes the summed probability of all length [](https://www.codecogs.com/eqnedit.php?latex=t%0) paths that are mapped by [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BB%7D%0) onto the length [](https://www.codecogs.com/eqnedit.php?latex=%5Cleft%20%5Clfloor%7Bu%2F2%7D%5Cright%20%5Crfloor%0) prefix of [](https://www.codecogs.com/eqnedit.php?latex=l%0). (Note, [](https://www.codecogs.com/eqnedit.php?latex=%5Cleft%20%5Clfloor%7Bu%2F2%7D%5Cright%20%5Crfloor%0) is the floor of [](https://www.codecogs.com/eqnedit.php?latex=u%2F2%0), the greatest integer less than or equal to [](https://www.codecogs.com/eqnedit.php?latex=u%2F2%0).) For some sequence [](https://www.codecogs.com/eqnedit.php?latex=s%0), let [](https://www.codecogs.com/eqnedit.php?latex=s_%7Bp%3Aq%7D%0) denote the sub-sequence [](https://www.codecogs.com/eqnedit.php?latex=s_p%2C%20s_%7Bp%2B1%7D%2C...%2Cs_%7Bq-1%7D%2Cs_q%0), and define the set [](https://www.codecogs.com/eqnedit.php?latex=V(t%2Cu)%20%5Cequiv%20%7B%20%5Cpi%20%5Cin%20%7BA'%7D%5Et%20%3A%20%5Cmathcal%7BB%7D(%5Cpi)%20%3D%20l_%7B1%3A%5Cleft%20%5Clfloor%7Bu%2F2%7D%5Cright%20%5Crfloor%7D%20%5Ctext%7B%20and%20%7D%20%5Cpi_t%20%3D%20l'u%20%7D%0)*.* [**](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%0) *then becomes*

[**](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%20%5Cequiv%20%5Csum%7B%5Cpi%20%5Cin%20V(t%2Cu)%7D%20%5Cprod_%7Bi%3D1%7D%5E%7Bt%7D%20y_%7Bi%2C%5Cpi_i%7D%0) - - - (4)

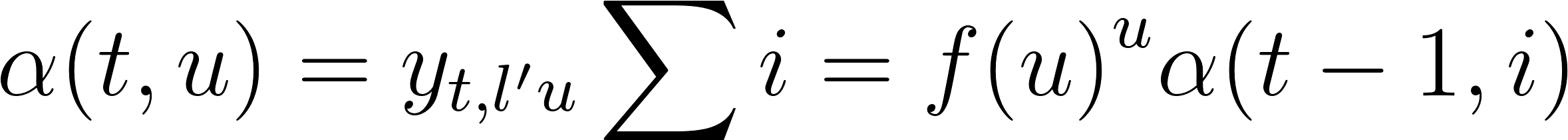
The forward variables at time [](https://www.codecogs.com/eqnedit.php?latex=t%0) can be calculated recursively from those at time [](https://www.codecogs.com/eqnedit.php?latex=t%20-%201%0) and expressed as the sum of the forward variables with and without the final blank at time [](https://www.codecogs.com/eqnedit.php?latex=T%0).

[](https://www.codecogs.com/eqnedit.php?latex=%5CPr(%20l%20%5C%2C%20%7C%20%5C%2C%20x)%20%3D%20%5Calpha(T%2C%20U')%20%2B%20%5Calpha(T%2C%20U'%20-%201)%0) - - - (5)

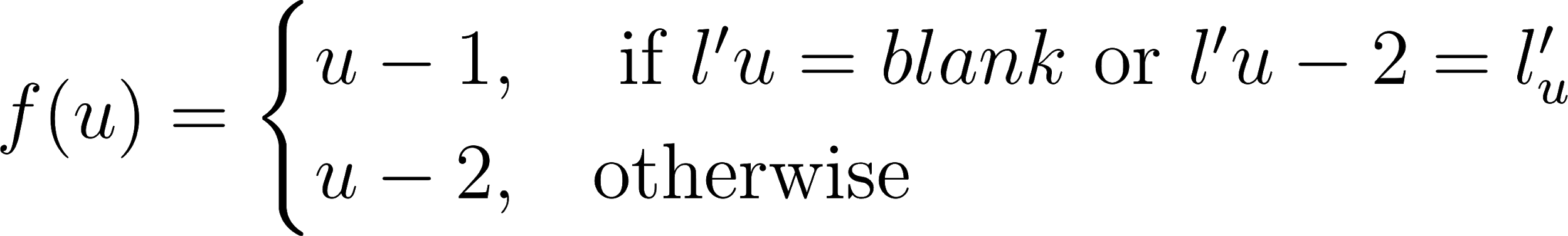
For the initial conditions, correct paths begin with a blank symbol [](https://www.codecogs.com/eqnedit.php?latex=(b)%0) and the first symbol [](https://www.codecogs.com/eqnedit.php?latex=(l_1)%0):

[](https://www.codecogs.com/eqnedit.php?latex=%5Cbegin%7Baligned%7D%5Calpha(1%2C%201)%20%26%3D%20y_%7B1%2Cb%7D%20%5C%5C%20%5Calpha(1%2C%202)%20%26%3D%20y_%7B1%2Cl_1%7D%20%5C%5C%20%5Calpha(1%2C%20u)%20%26%3D%200%2C%20%5C%2C%20%5Cforall%20u%20%3E%202%20%5Cend%7Baligned%7D%0) - - - (6)

The forward variable then takes the following recursive form:

[](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%20%3D%20y_%7Bt%2C%20l'u%7D%20%5Csum%7Bi%20%3D%20f(u)%7D%5E%7Bu%7D%20%5Calpha(t-1%2C%20i)%0) - - - (7)

where

[](https://www.codecogs.com/eqnedit.php?latex=f(u)%20%3D%5Cbegin%7Bcases%7Du-1%2C%20%26%20%5Ctext%7B%20if%20%7D%20l'u%20%3D%20blank%20%5Ctext%7B%20or%20%7D%20l'%7Bu-2%7D%20%3D%20l'_%7Bu%7D%20%5C%5C%20u-2%2C%20%26%20%5Ctext%7Botherwise%7D%5Cend%7Bcases%7D%0) - - - (8)

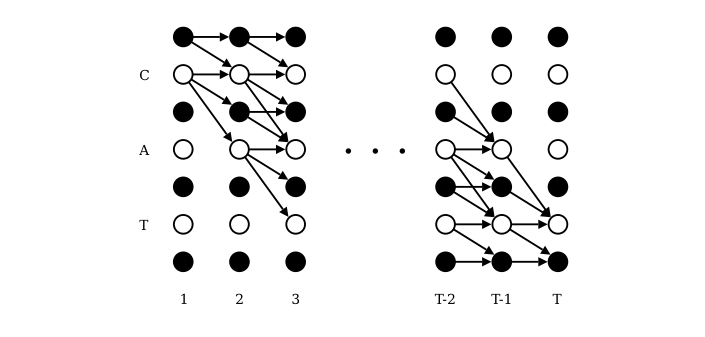
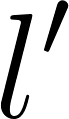
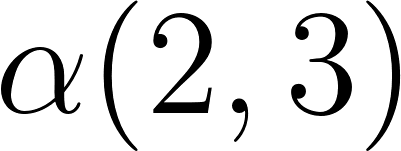
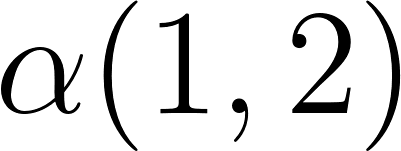
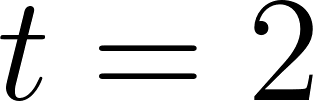
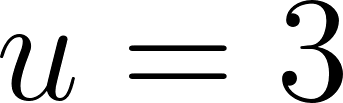
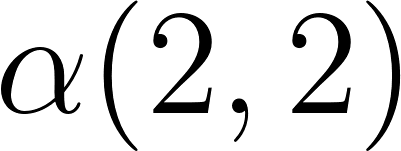
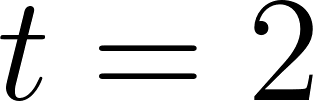
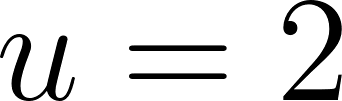
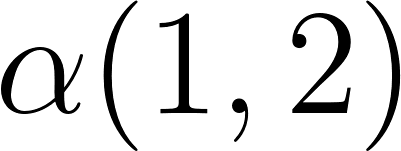
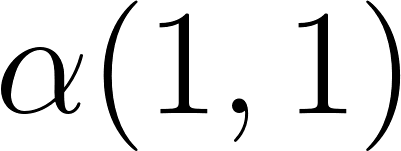
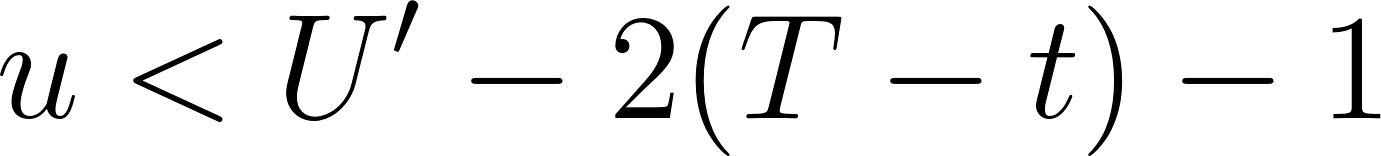
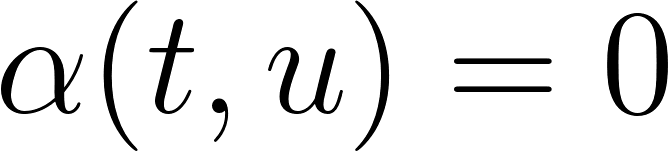
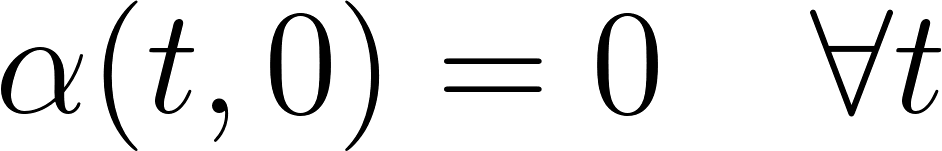
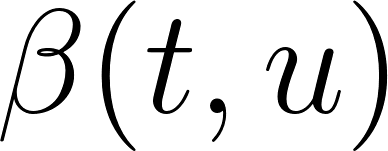
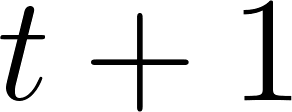
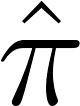
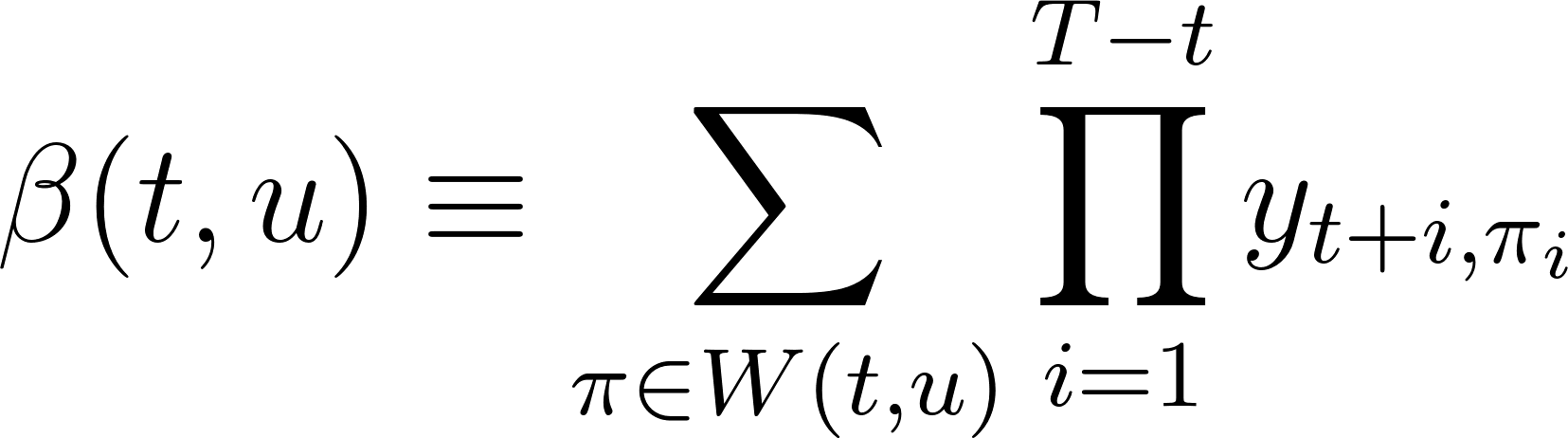


Figure 6.2: Forward-Backward Algorithm Lattice \citep{graves2006}

Figure \ref{fig\_6\_2\_lattice} expresses the recurrence relation for [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2C%20u)%0). While [](https://www.codecogs.com/eqnedit.php?latex=t%0) is expressed on the [](https://www.codecogs.com/eqnedit.php?latex=x%0) axis, [](https://www.codecogs.com/eqnedit.php?latex=u%0) is illustrated on the [](https://www.codecogs.com/eqnedit.php?latex=y%0) axis. The CTC algorithm assumes that outputs of the network potentially alternate between blank symbols indicated as black circles and non-blank elements, the white circles, all in [](https://www.codecogs.com/eqnedit.php?latex=l'%0). The sequential graph constructed from this 2-dimensional matrix show computational dependencies between sequential pairs of the recurrence relation for [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%0). Therefore, the value [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(2%2C3)%0), formed from [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(1%2C2)%0), corresponds to the [](https://www.codecogs.com/eqnedit.php?latex=blank%0) symbol at [](https://www.codecogs.com/eqnedit.php?latex=t%3D2%0) and [](https://www.codecogs.com/eqnedit.php?latex=u%3D3%0), is . Also, [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(2%2C2)%0), equivalent to the symbol [](https://www.codecogs.com/eqnedit.php?latex=c%0) at [](https://www.codecogs.com/eqnedit.php?latex=t%3D2%0) and [](https://www.codecogs.com/eqnedit.php?latex=u%3D2%0), is gotten from [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(1%2C2)%0) and [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(1%2C1)%0). Note that there are not enough time steps when [](https://www.codecogs.com/eqnedit.php?latex=u%20%3C%20U'-2(T-t)-1%0), therefore, [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%3D0%0). Also note the boundary condition

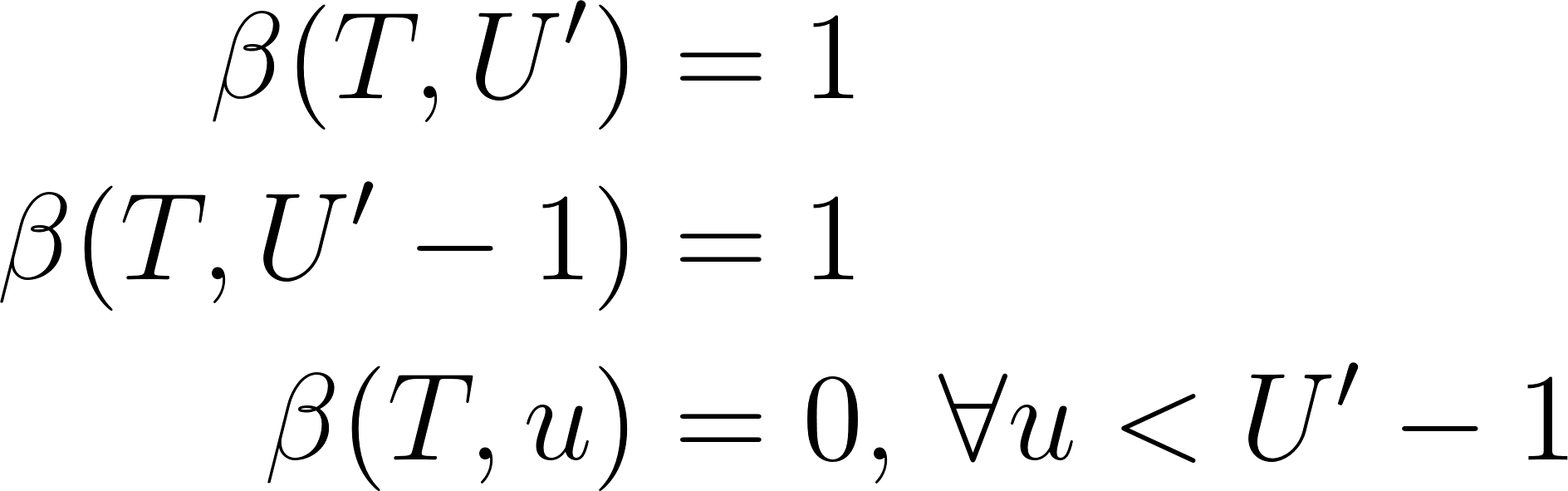
[](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2C%200)%20%3D%200%20%5Cquad%20%5Cforall%20t%0) - - - (9)

The backward variables [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta(t%2Cu)%0) is built similarly to the forward variable. Rather than moving from the start of the sequence to [](https://www.codecogs.com/eqnedit.php?latex=t%0) we define the path starting at [](https://www.codecogs.com/eqnedit.php?latex=t%20%2B%201%0) that completes the sequence at [](https://www.codecogs.com/eqnedit.php?latex=T%0) when appended the path [](https://www.codecogs.com/eqnedit.php?latex=%5Chat%7B%5Cpi%7D%0) that generates [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2Cu)%0). The following is defined [](https://www.codecogs.com/eqnedit.php?latex=W(t%2Cu)%20%5Cequiv%20%7B%20%5Cpi%20%5Cin%20%7BA'%7D%5E%7BT-t%7D%20%3A%20%5Cmathcal%7BB%7D(%5Chat%7B%5Cpi%7D%20%2B%20%5Cpi)%20%3D%20l%20%5C%2C%20%5C%2C%20%5Cforall%20%5Chat%7B%5Cpi%7D%20%5Cin%20V(t%2Cu)%20%7D%0). Then

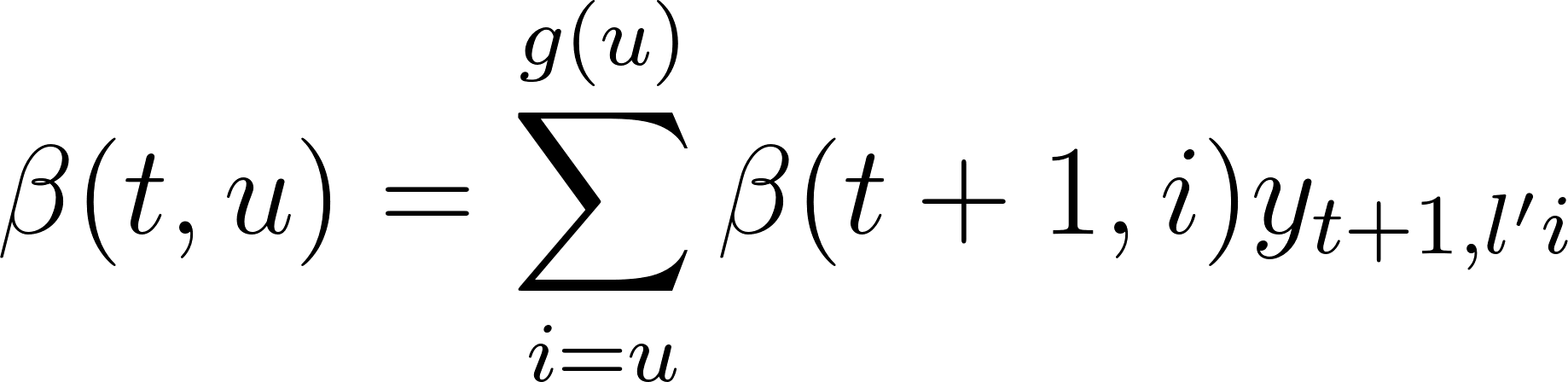
[](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta(t%2Cu)%20%5Cequiv%20%5Csum_%7B%5Cpi%20%5Cin%20W(t%2Cu)%7D%20%5Cprod_%7Bi%3D1%7D%5E%7BT%20-%20t%7D%20y_%7Bt%20%2B%20i%2C%5Cpi_i%7D%0) - - - (10)

The backward variable is therefore equivalently initialised as follows

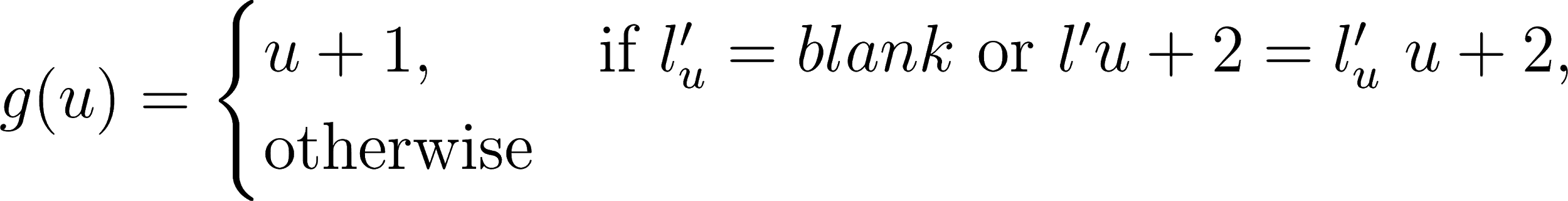
\begin{equation}

[](https://www.codecogs.com/eqnedit.php?latex=%5Cbegin%7Baligned%7D%20%5Cbeta(T%2C%20U')%20%26%3D%201%20%5C%5C%20%5Cbeta(T%2C%20U'%20-%201)%20%26%3D%201%20%5C%5C%20%5Cbeta(T%2C%20u)%20%26%3D%200%2C%20%5C%2C%20%5Cforall%20u%20%3C%20U'%20-%201%20%5Cend%7Baligned%7D%0) - - - (11)

The recursion rule is defined as follows:

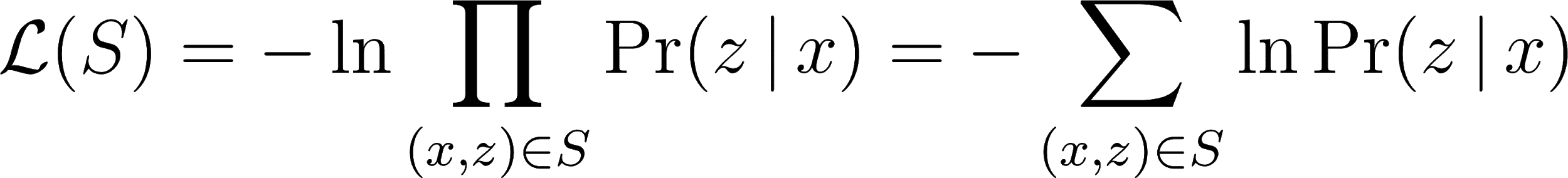
[](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta(t%2C%20u)%20%3D%20%5Csum_%7Bi%20%3D%20u%7D%5E%7Bg(u)%7D%20%5Cbeta(t%2B1%2C%20i)%20y_%7Bt%2B1%2C%20l'i%7D%0) - - - (12)

*similarly,*

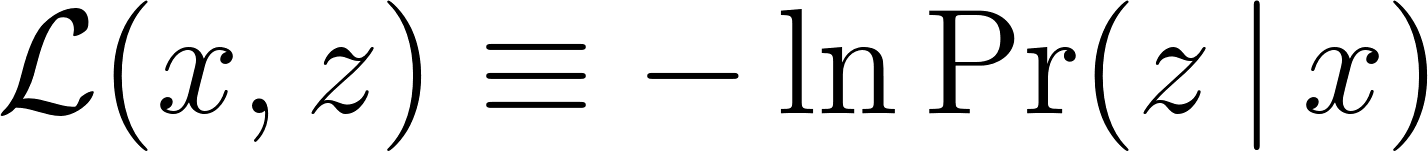
[**](https://www.codecogs.com/eqnedit.php?latex=g(u)%20%3D%20%5Cbegin%7Bcases%7D%20u%20%2B%201%2C%26%20%5Ctext%7Bif%20%7D%20l'_u%20%3D%20blank%20%5Ctext%7B%20or%20%7D%20l'%7Bu%2B2%7D%20%3D%20l'_%7Bu%7D%20%5C%20u%20%2B%202%2C%26%20%5Ctext%7Botherwise%7D%20%5Cend%7Bcases%7D%0) - - - (13)

## Loss Function

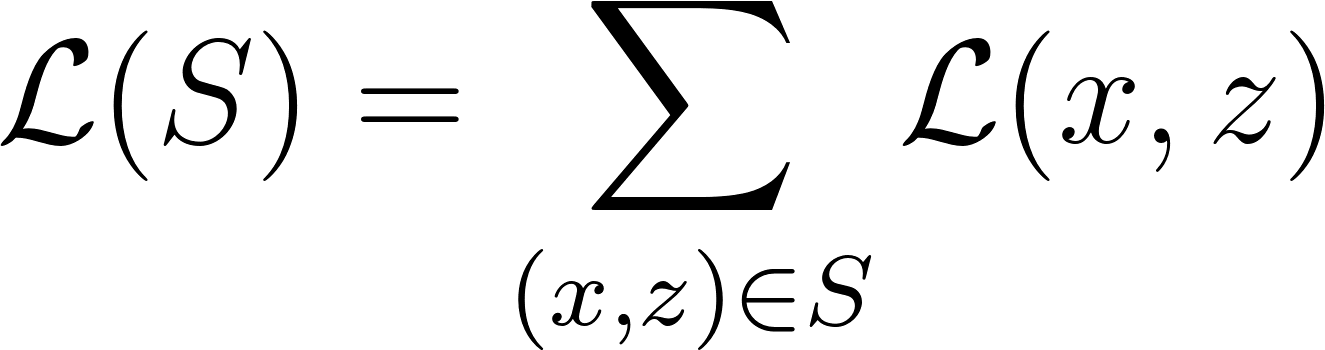
The cross entropy error is a loss function used to measure accuracy of probabilistic measures. It is calculated as the negative log probability of a likelihood measure. The CTC loss function [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(S)%0) uses the cross entropy loss function of and is defined as the cross entropy error of correctly labeling all the training samples in some training set S:

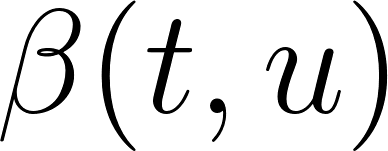
[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(S)%20%3D%20-%20%5Cln%20%5Cprod_%7B(x%2Cz)%20%5Cin%20S%7D%20%5CPr(z%20%5C%2C%20%7C%20%5C%2C%20x)%20%3D%20-%20%5Csum_%7B(x%2Cz)%20%5Cin%20S%7D%20%5Cln%20%5CPr(z%20%5C%2C%20%7C%20%5C%2C%20x)%0) - - - (14)

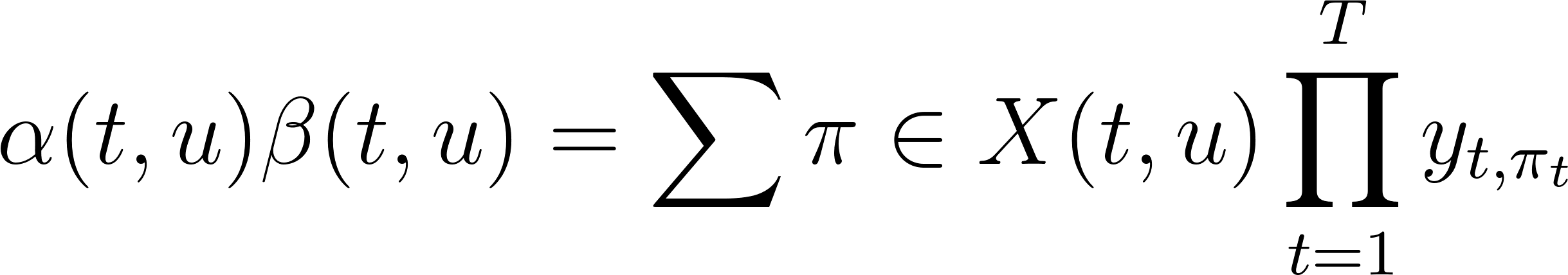
where [](https://www.codecogs.com/eqnedit.php?latex=z%0) is the output label and [](https://www.codecogs.com/eqnedit.php?latex=x%0) is the input sequence. Since [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(S)%0) in equation \ref{eqn\_c3\_ctc11} is differentiable, this loss function can be back propagated to the softmax layer in the BiRNN configuration discussed in section \ref{deepspeech}.

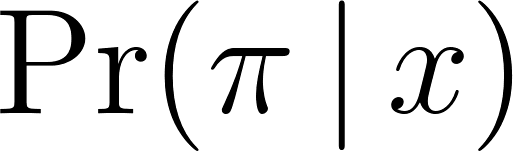
[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(x%2Cz)%20%5Cequiv%20-%20%5Cln%20%5CPr(z%20%5C%2C%20%7C%20%5C%2C%20x)%0) - - - (15)

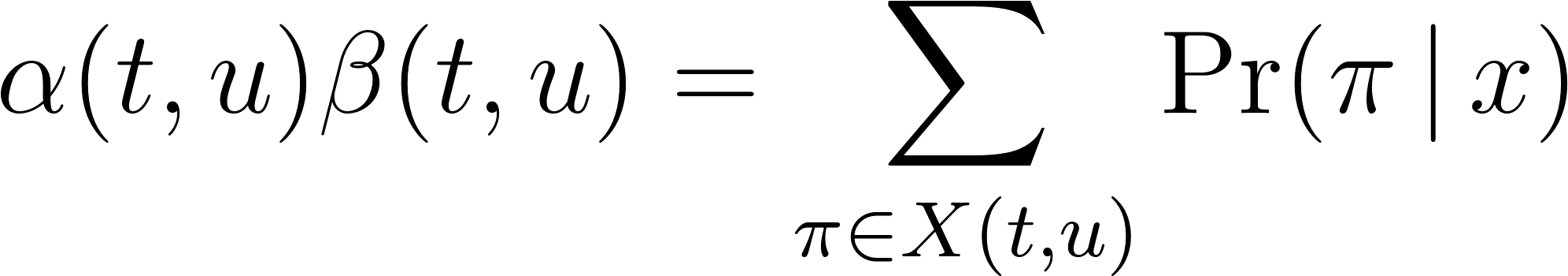
and therefore

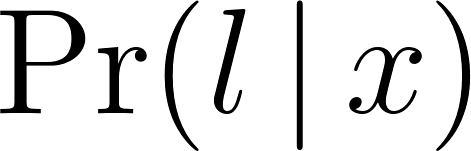
[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(S)%20%3D%20%5Csum_%7B(x%2Cz)%20%5Cin%20S%7D%20%5Cmathcal%7BL%7D(x%2Cz)%0) - - - (16)

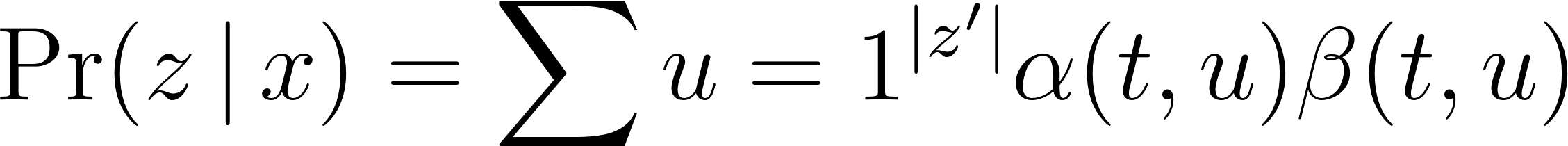
From the definition of the forward and backward variables [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2C%20u)%0) and [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta(t%2C%20u)%0), we also establish that [](https://www.codecogs.com/eqnedit.php?latex=X(t%2Cu)%20%5Cequiv%20%5C%7B%20%5Cpi%20%5Cin%20%7BA'%7D%5ET%20%3A%20%5Cmathcal%7BB%7D(%5Cpi)%20%3D%20z%2C%20%5Cpi_t%20%3D%20%7Bz'%7Du%20%5C%7D%0)*, such that*

[**](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2C%20u)%20%5Cbeta(t%2C%20u)%20%3D%20%5Csum%7B%5Cpi%20%5Cin%20X(t%2Cu)%7D%20%5Cprod_%7Bt%20%3D%201%7D%5E%7BT%7D%20y_%7Bt%2C%20%5Cpi_t%7D%0) - - - (17)

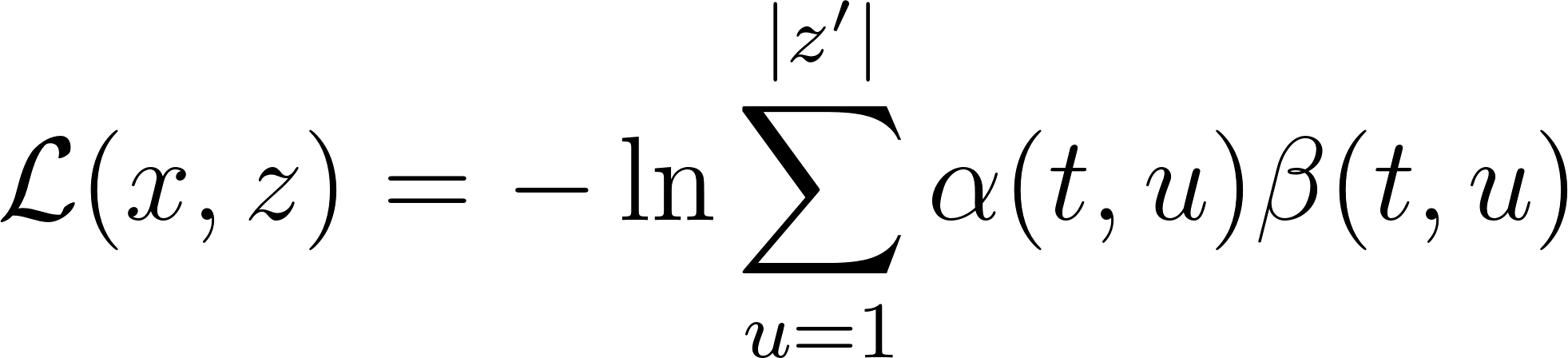
then substituting [](https://www.codecogs.com/eqnedit.php?latex=%5CPr(%5Cpi%20%5C%2C%20%7C%20%5C%2C%20x)%0) from the expression in equation \ref{eqn\_c3\_ctc01}, we have

[](https://www.codecogs.com/eqnedit.php?latex=%5Calpha(t%2C%20u)%20%5Cbeta(t%2C%20u)%20%3D%20%5Csum_%7B%5Cpi%20%5Cin%20X(t%2Cu)%7D%20%5CPr(%5Cpi%20%5C%2C%20%7C%20%5C%2C%20x)%0) - - - (18)

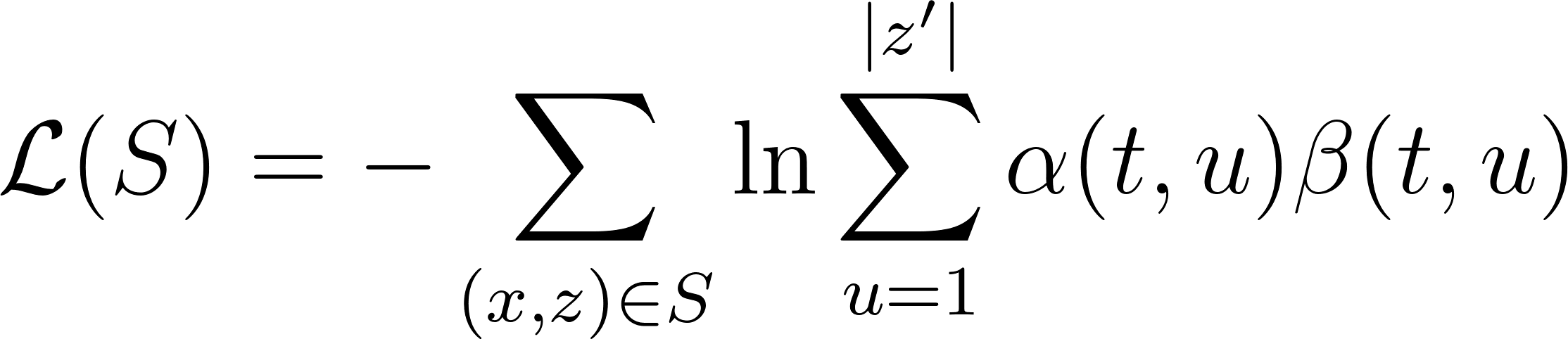
Also observe that [](https://www.codecogs.com/eqnedit.php?latex=%5CPr(l%20%5C%2C%20%7C%20%5C%2C%20x)%0) is equivalent to the total probability [](https://www.codecogs.com/eqnedit.php?latex=%5CPr(z%20%5C%2C%20%7C%20%5C%2C%20x)%0). Paths going through [](https://www.codecogs.com/eqnedit.php?latex=z'u%0) *at time* [**](https://www.codecogs.com/eqnedit.php?latex=t%0) *can be obtained as summed over all* [**](https://www.codecogs.com/eqnedit.php?latex=u%0) *to get*

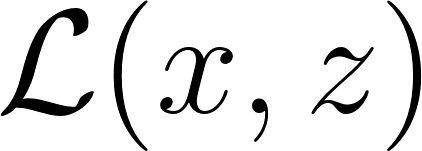
[**](https://www.codecogs.com/eqnedit.php?latex=%5CPr(z%20%5C%2C%20%7C%20%5C%2C%20x)%20%3D%20%5Csum%7Bu%20%3D%201%7D%5E%7B%7Cz'%7C%7D%20%5Calpha(t%2C%20u)%20%5Cbeta(t%2C%20u)%0) *- - -* (19)

Thus a sample loss is determined by

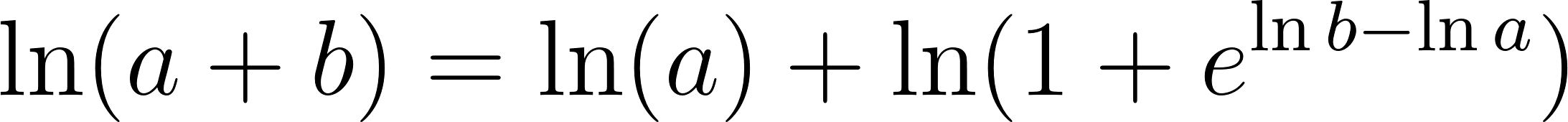
[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(x%2C%20z)%20%3D%20-%20%5Cln%20%5Csum_%7Bu%20%3D%201%7D%5E%7B%7Cz'%7C%7D%20%5Calpha(t%2C%20u)%20%5Cbeta(t%2C%20u)%0) - - - (20)

and therefore the overall loss is given by

[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(S)%20%3D%20-%5Csum_%7B(x%2Cz)%20%5Cin%20S%7D%20%5Cln%20%5Csum_%7Bu%20%3D%201%7D%5E%7B%7Cz'%7C%7D%20%5Calpha(t%2C%20u)%20%5Cbeta(t%2C%20u)%0) - - - (21)

In the model described in this work, the gradient [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BL%7D(x%2C%20z)%0) is computed using TensorFlow's automatic differentiation capabilities. In practice, computations soon lead to underflow. However, the log scale, being used in the above loss function calculations avoids this situation and another useful equation in this context is

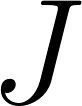
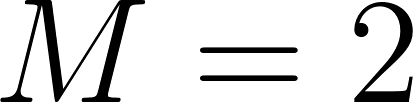
\begin{equation}

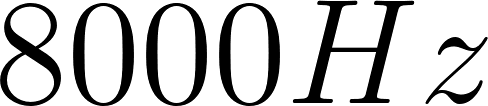
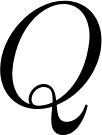
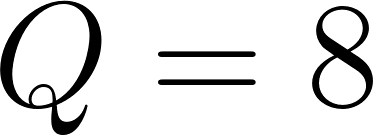
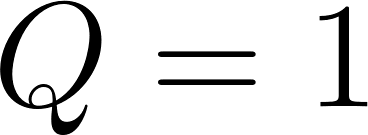
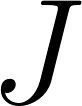
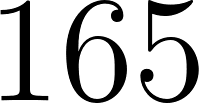
[](https://www.codecogs.com/eqnedit.php?latex=%5Cln(a%20%2B%20b)%20%3D%20%5Cln(a)%20%2B%20%5Cln(1%20%2B%20e%5E%7B%5Cln%20b%20-%20%5Cln%20a%7D)%0) - - - (22)

# The CTC Speech Model

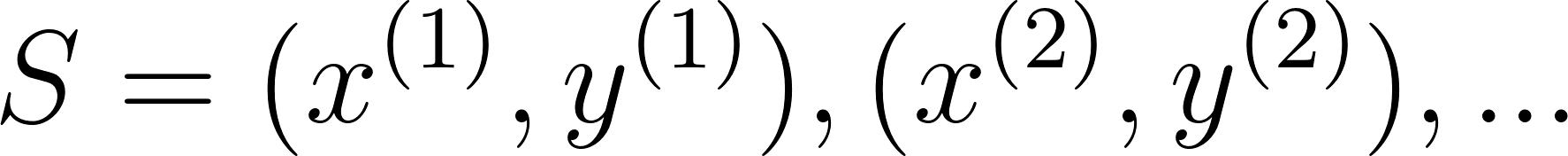
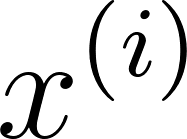
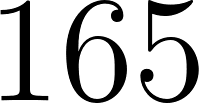
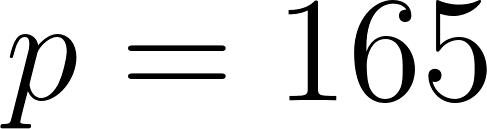
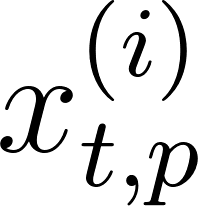
Earlier in chapter one, deep learning was defined as a type of representational learning whereby different levels of complexity are captured in internal layer-wise encapsulations. It has also been noted that layer-wise stacking of neural and neural network type architectures such as the Restricted Boltzmann Machine (RBM) deep belief networks (DBMs) were 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 are described and the decoding algorithm is detailed.

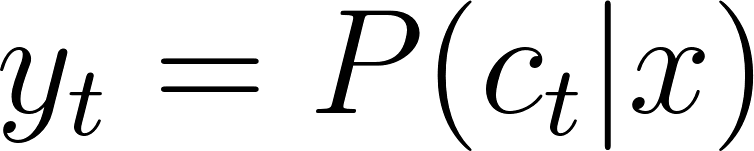
## Deep Scattering Features

The fast wavelet transform 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 [](https://www.codecogs.com/eqnedit.php?latex=2%0) 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, [](https://www.codecogs.com/eqnedit.php?latex=T%0); The Q-band value for the band pass filter and the number of wavelets [](https://www.codecogs.com/eqnedit.php?latex=J%0) at each scattering layer for the total number of layers, [](https://www.codecogs.com/eqnedit.php?latex=M%3D2%0).

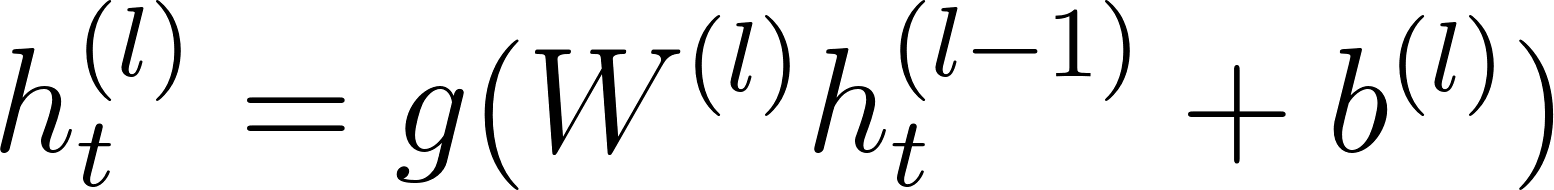
The matlab scatnet toolbox \citep{anden2014scatnet}, used to determine the scatter coefficient features for this research, provides optimal values for hyper parameters for audio signal processing into scatter features. In this regime the value for the hyper parameter [](https://www.codecogs.com/eqnedit.php?latex=T%0)= [](https://www.codecogs.com/eqnedit.php?latex=512%0) samples per window. This corresponds a window of 50 milliseconds for the audio signals sampled at [](https://www.codecogs.com/eqnedit.php?latex=8000%20Hz%0). For the first scattering layer the [](https://www.codecogs.com/eqnedit.php?latex=Q%0)-band parameter was [](https://www.codecogs.com/eqnedit.php?latex=Q%3D8%0) and the second scattering layer took the value [](https://www.codecogs.com/eqnedit.php?latex=Q%3D1%0). Finally [](https://www.codecogs.com/eqnedit.php?latex=J%0) is pre-calculated based on the value of [](https://www.codecogs.com/eqnedit.php?latex=T%0). These after Scat-Net processing produce a feature-vector having [](https://www.codecogs.com/eqnedit.php?latex=165%0) dimensions. These feature vectors in turn are used as inputs to the bi-direction neural network model whose architecture is described in succeeding sections.

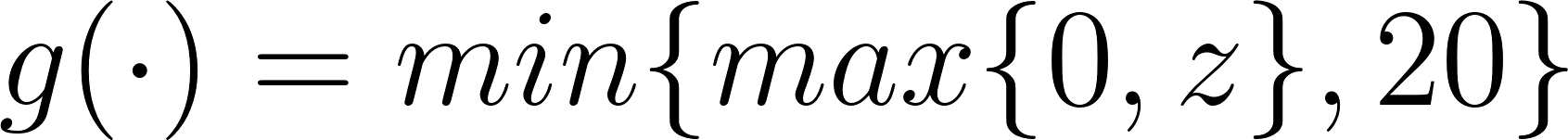
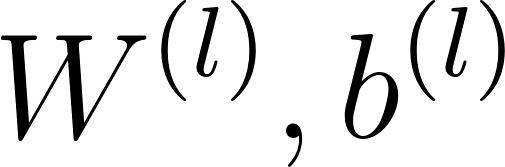
## CTC-BiRNN Architecture

The core of the system is a bidirectional recurrent neural network (BiRNN) trained to ingest scatter coefficients described in the previous section, in order to generate English text transcriptions. An end-to-end system therefore specifies that utterances [](https://www.codecogs.com/eqnedit.php?latex=x%0) and the corresponding label [](https://www.codecogs.com/eqnedit.php?latex=y%0) be sampled from a training set such that the sample [](https://www.codecogs.com/eqnedit.php?latex=S%20%3D%20%7B(x%5E%7B(1)%7D%2C%20y%5E%7B(1)%7D)%2C%20(x%5E%7B(2)%7D%2C%20y%5E%7B(2)%7D)%2C%20.%20.%20.%7D%0) In our end-to-end model, each utterance, [](https://www.codecogs.com/eqnedit.php?latex=x%5E%7B(i)%7D%0) is a processed feature vector consisting of [](https://www.codecogs.com/eqnedit.php?latex=165%0) dimensions. Recall, every window passes through a scattering transform to yield an input of vector of [](https://www.codecogs.com/eqnedit.php?latex=p%3D165%0) features; consequently, [](https://www.codecogs.com/eqnedit.php?latex=x%5E%7B(i)%7D_%7Bt%2Cp%7D%0) denotes the p-th feature in a scatter transform at time [](https://www.codecogs.com/eqnedit.php?latex=t%0).

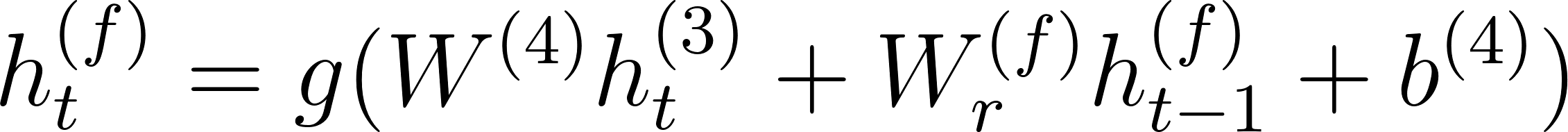
GPU training of the speech model architecture developed above was conducted using Mozilla deepspeech \cite{mozilla/deepspeech\_2019} CTC bi-directional RNN implementation along with the accompanying Mozilla Common voice dataset \cite{common voice by mozilla\_2019}. The Common Voice Dataset project consists of voice samples in short recordings approximately 4 seconds each. The complete dataset is about 250 hours of recording divided into training, test and development subsets. The BiRNN, given the input sequence, [](https://www.codecogs.com/eqnedit.php?latex=x%0), outputs a sequence of probabilities [](https://www.codecogs.com/eqnedit.php?latex=%5Cyat%7By%7D_t%3D%5CmathBB%7BP%7D(c_t%7Cx)%0), where [](https://www.codecogs.com/eqnedit.php?latex=c_t%20%5Cin%20%7Ba%2Cb%2Cc%2C%20.%20.%20.%20%2C%20z%2C%20space%2C%20apostrophe%2C%20blank%7D%0).

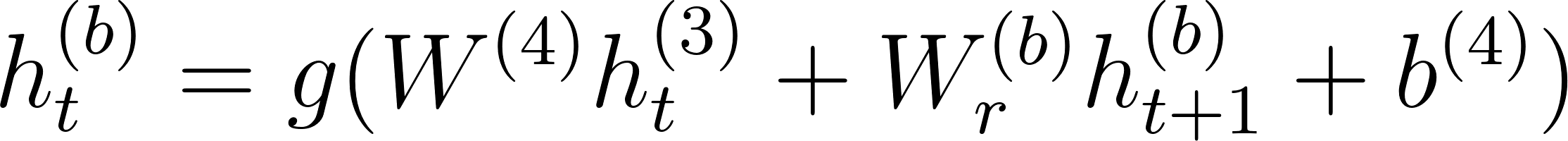
The actual architecture of our core Bi-RNN is similar to the deepspeech system described in \cite{hannun2014deep}. This structure consists of a 5 hidden layers and one output layer. The first three layers are regular DNNs followed by a bi-directional recurrent layer. As such, the output of the first three layers are computed by:

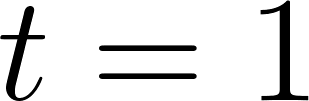
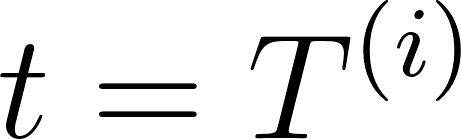
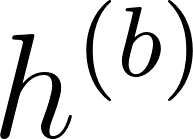
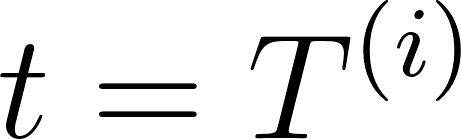
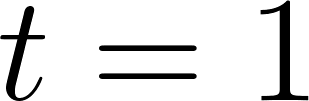
[](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(l)%7D_t%20%3D%20g(W%5E%7B(l)%7D%20h%5E%7B(l-1)%7D_t%20%2B%20b%5E%7B(l)%7D)%0)

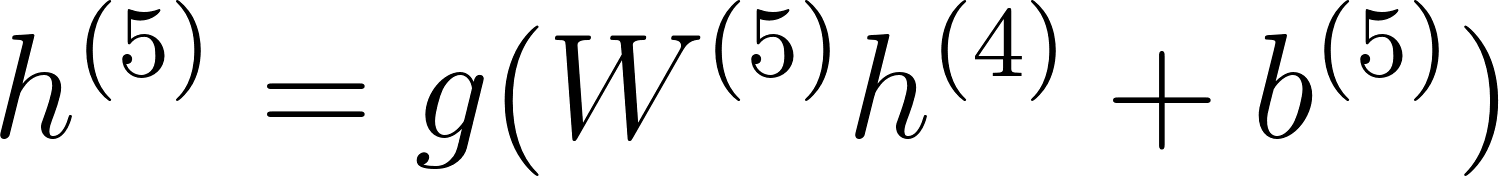
[](https://www.codecogs.com/eqnedit.php?latex=g(%5Ccdot)%20%3D%20min%5C%7Bmax%5C%7B0%2Cz%5C%7D%2C20%5C%7D%0) is the clipped rectified linear unit and [](https://www.codecogs.com/eqnedit.php?latex=W%5E%7B(l)%7D%2Cb%5E%7B(l)%7D%0) are weight matrix and bias parameters for layer [](https://www.codecogs.com/eqnedit.php?latex=l%0) as described in sections \ref{dnn} and \ref{deepspeech} respectively.

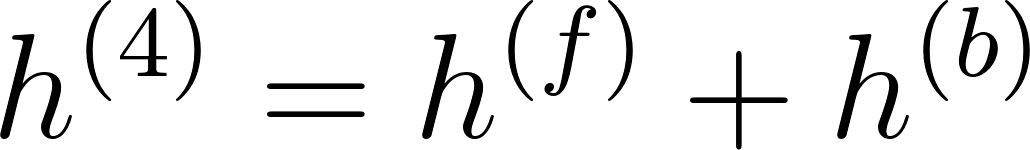
It was shown in chapter \ref{ch3RNN} the recurrent layer comprise a forward and backward RNNs whose equations are repeated here for reference

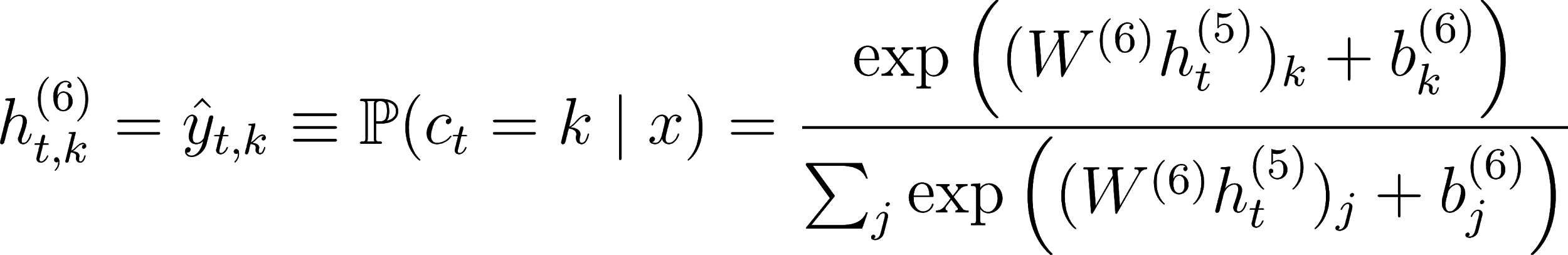
[](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(f)%7D_t%20%3D%20g(W%5E%7B(4)%7D%20h%5E%7B(3)%7D_t%20%2B%20W%5E%7B(f)%7D_r%20h%5E%7B(f)%7D_%7Bt-1%7D%20%2B%20b%5E%7B(4)%7D)%0)

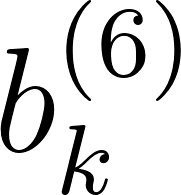
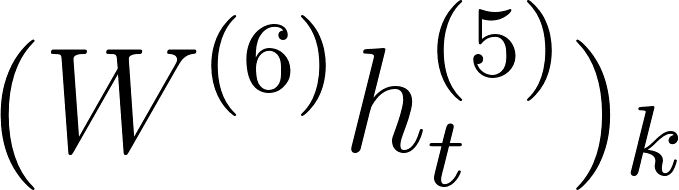
[](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(b)%7D_t%20%3D%20g(W%5E%7B(4)%7D%20h%5E%7B(3)%7D_t%20%2B%20W%5E%7B(b)%7D_r%20h%5E%7B(b)%7D_%7Bt%2B1%7D%20%2B%20b%5E%7B(4)%7D)%0)

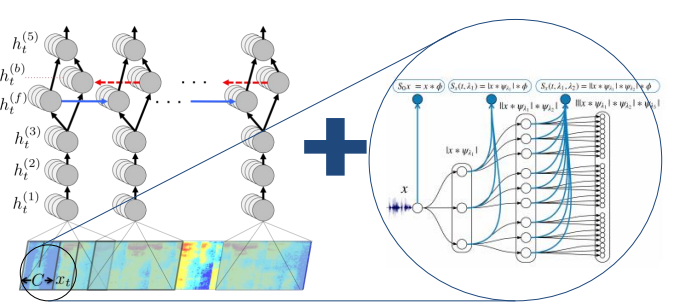
Consequently, [](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(f)%7D%0) is the sequential computation from [](https://www.codecogs.com/eqnedit.php?latex=t%3D1%0) to [](https://www.codecogs.com/eqnedit.php?latex=t%3DT%5E%7B(i)%7D%0) for the [](https://www.codecogs.com/eqnedit.php?latex=i%0)-th utterance and [](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(b)%7D%0) is the reverse computation from [](https://www.codecogs.com/eqnedit.php?latex=t%3DT%5E%7B(i)%7D%0) to [](https://www.codecogs.com/eqnedit.php?latex=t%3D1%0). In addition the output from layer five is summarily given as the combined outputs from the recurrent layer:

[](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(5)%7D%20%3D%20g(W%5E%7B(5)%7D%20h%5E%7B(4)%7D%20%2B%20b%5E%7B(5)%7D)%0)

where [](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(4)%7D%20%3D%20h%5E%7B(f)%7D%20%2B%20h%5E%7B(b)%7D%0). The output of the Bi-RNN on layer 6 is a standard soft-max function that outputs a predicted character over probabilities for each time slice [](https://www.codecogs.com/eqnedit.php?latex=t%0) and character [](https://www.codecogs.com/eqnedit.php?latex=k%0) in the alphabet:

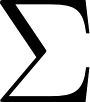
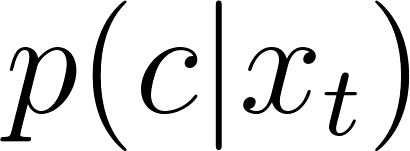
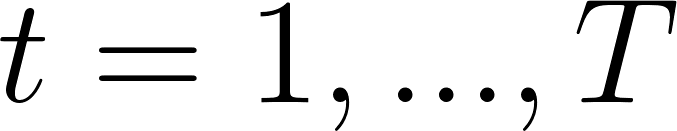
[](https://www.codecogs.com/eqnedit.php?latex=h%5E%7B(6)%7D_%7Bt%2Ck%7D%20%3D%20%5Chat%7By%7D_%7Bt%2Ck%7D%20%5Cequiv%20%5Cmathbb%7BP%7D(c_t%20%3D%20k%20%5Cmid%20x)%20%3D%20%5Cfrac%7B%5Cexp%7B%20%5Cleft(%20(W%5E%7B(6)%7D%20h%5E%7B(5)%7D_t)_k%20%2B%20b%5E%7B(6)%7D_k%20%5Cright)%7D%7D%7B%5Csum_j%20%5Cexp%7B%5Cleft(%20(W%5E%7B(6)%7D%20h%5E%7B(5)%7D_t)_j%20%2B%20b%5E%7B(6)%7D_j%20%5Cright)%7D%7D%0)

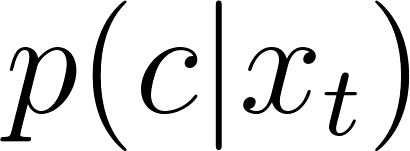
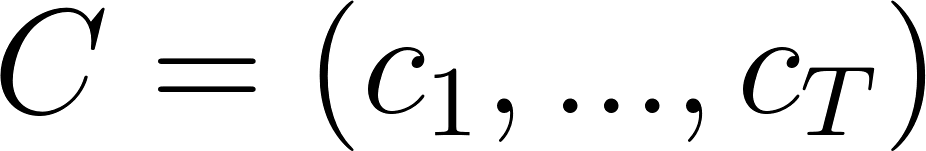
[](https://www.codecogs.com/eqnedit.php?latex=b%5E%7B(6)%7D_k%0) takes on the [](https://www.codecogs.com/eqnedit.php?latex=k%0)-th bias and [](https://www.codecogs.com/eqnedit.php?latex=(W%5E%7B(6)%7D%20h%5E%7B(5)%7D_t)_k%0) is the matrix product of the [](https://www.codecogs.com/eqnedit.php?latex=k%0)-th element. The error of the outputs are then computed using the CTC loss function \cite{graves\_2014} described in chapter \ref{ch3DNN}. A summary of our model is illustrated in figure \ref{ch06\_01\_ctc\_scatter}.

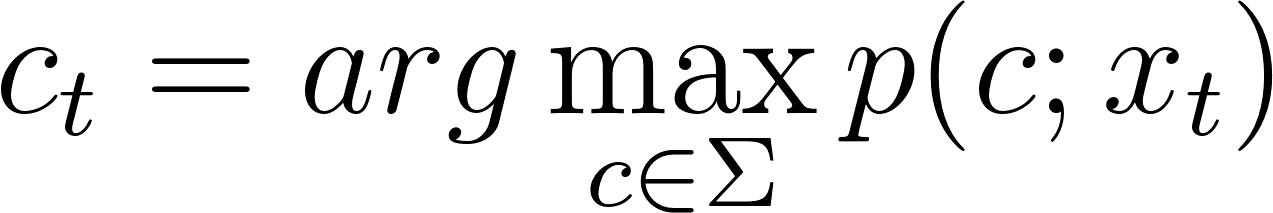


## CTC decoding

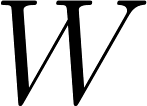
In chapter three the CTC loss function algorithm was established as being able to maximise the probability of two cases. The first case of transiting to a blank and the second case of transiting to a non blank. In this section, this concept is used to enable decoding of the network output from posterior distribution output to character sequences which can be measured against a reference transcription using either character error rate (CER) or word error rate (WER).

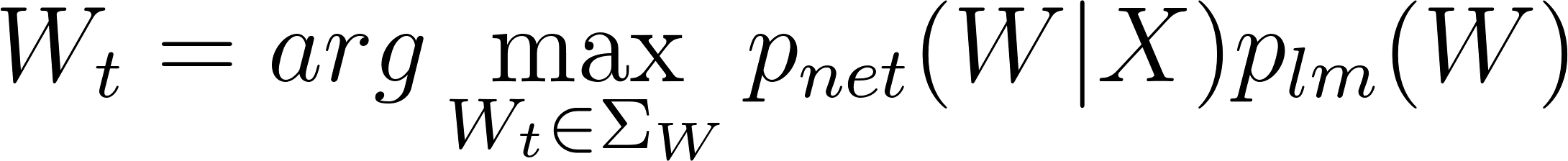
Recall, all the output symbols are in the alphabet [](https://www.codecogs.com/eqnedit.php?latex=%5CSigma%0) and augmented with the blank symbol. The posterior output of the CTC network is the probability of the symbol given the speech feature input [](https://www.codecogs.com/eqnedit.php?latex=p(c%7Cx_t)%0) at time [](https://www.codecogs.com/eqnedit.php?latex=t%0) for [](https://www.codecogs.com/eqnedit.php?latex=t%3D1%2C...%2C%20T%0) and [](https://www.codecogs.com/eqnedit.php?latex=T%0) is the length of the input sequence. Also recall two further sets of probabilities also being maintained by the model are the probability of a blank character [](https://www.codecogs.com/eqnedit.php?latex=p_b%0) and that of a non blank character [](https://www.codecogs.com/eqnedit.php?latex=p_%7Bnb%7D%0).

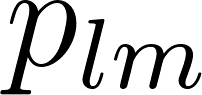
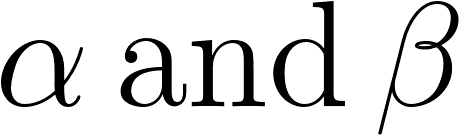
Several strategies have been employed to obtain a translation string from the output of the deep neural network. The prefix beam search employed by the CTC decoder of this research is derived from an initial greedy approximation, where at each time step it determines the argument that maximises the probability [](https://www.codecogs.com/eqnedit.php?latex=p(c%7Cx_t)%0) at each time step. Let [](https://www.codecogs.com/eqnedit.php?latex=C%3D(c_1%2C%20...%2C%20c_T)%0) be the character string then, the greedy approach has

[](https://www.codecogs.com/eqnedit.php?latex=c_t%3Darg%5Cmax_%7Bc%5Cin%5CSigma%7Dp(c%3Bx_t)%0) - - - 23b

However, this simple approximation is unable to collapse repeating sequences and remove blank symbols. In addition, the approximation is unable to include the constraint of a lexicon or language model.

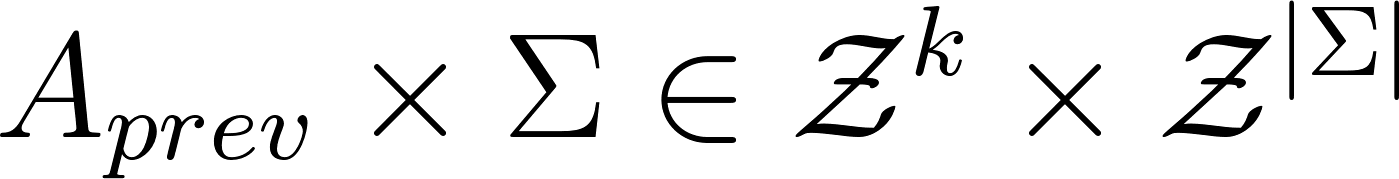
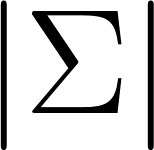
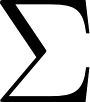
The prefix beam search algorithm \cite{hannun2014first} adopted in this work incorporates a language model derived from a lexicon in addition to keeping track of the various likelihoods used for decoding. For the language model constraint, the transcription [](https://www.codecogs.com/eqnedit.php?latex=W%0) is recovered from acoustic input [](https://www.codecogs.com/eqnedit.php?latex=X%0) at time [](https://www.codecogs.com/eqnedit.php?latex=t%0) by choosing the word which maximising the posterior probability:

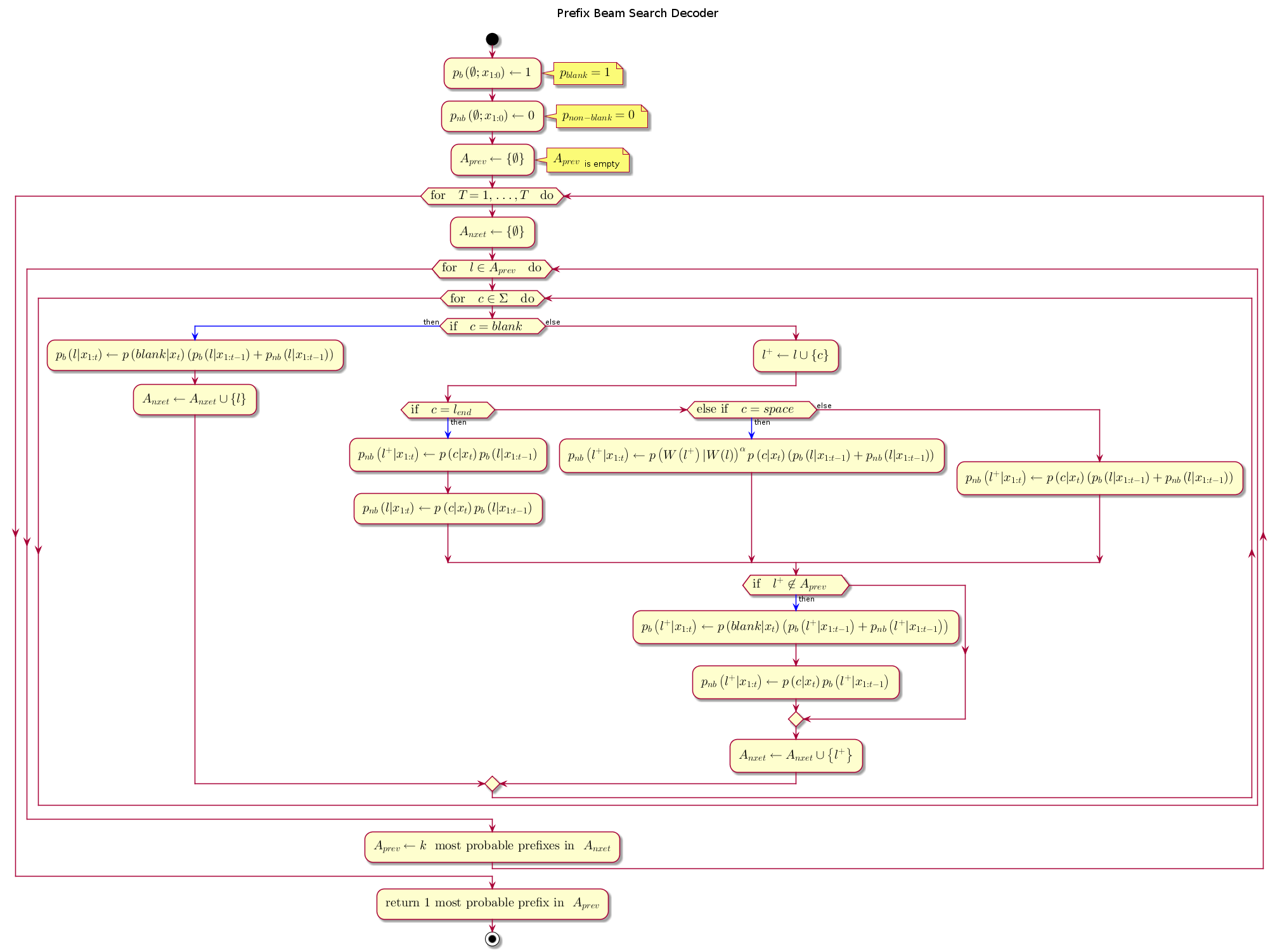
[](https://www.codecogs.com/eqnedit.php?latex=W_t%3Darg%5Cmax_%7BW_t%20%5Cin%20%5CSigma_W%7D%20p_%7Bnet%7D(W%7CX)p_%7Blm%7D(W)%0) - - - (24)

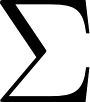
In equation \ref{eqn\_c6\_decoder01}, the Bayes product of language model prior [](https://www.codecogs.com/eqnedit.php?latex=p_%7Blm%7D%0) and the network output [](https://www.codecogs.com/eqnedit.php?latex=p_%7Bnet%7D%0) are utilised to maximise the probability of a particular character-word sequence in the lexicon given by [](https://www.codecogs.com/eqnedit.php?latex=%5CSigma_W%0). The overall calculation used to derive the final posterior distribution includes word insertion factors ([](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%20%5C%2C%20%5Ctext%7Band%7D%20%5C%2C%20%5Cbeta%0)) used to balance the highly constrained n-gram language-model.

[](https://www.codecogs.com/eqnedit.php?latex=W_t%3Darg%5Cmax_%7BW_t%20%5Cin%20%5CSigma_W%7D%20p_%7Bnet%7D(W%7CX)p_%7Blm%7D(W)%5E%5Calpha%7CW%7C%5E%5Cbeta%0) - - - (25)

The second strategy adopted by the prefix beam search which improves the decoding algorithm is the beam search strategy. With this approach, the search maintains all possible paths; however, it retains only [](https://www.codecogs.com/eqnedit.php?latex=k%0) number paths which maximise the output sequence probability. Improvements gained with this method are seen when certain maximal paths are made obsolete owing to new information derived from the multiple paths in being maintained in memory.

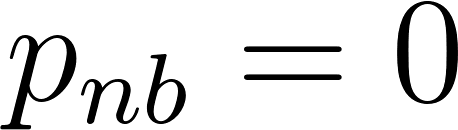
The recursive prefix beam search algorithm illustrated in figure \ref{fig\_c6\_decoder01} attempts to find the string formulated in equation \ref{eqn\_c6\_decoder02}. Two sets prefixes [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0) and [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bnxet%7D%0) are initialised, such that at [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bnxet%7D%0) maintains the prefixes in the current timestep while [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0) maintains only [](https://www.codecogs.com/eqnedit.php?latex=k%0)-prefixes from the previous timestep. Note that at the end of each time step [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0) is updated with only [](https://www.codecogs.com/eqnedit.php?latex=k%0)-most probable prefixes from [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bnxet%7D%0). Therefore while, [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bnxet%7D%0) contains all the possible new paths from based on [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0) as a cartesian product of [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%20%5Ctimes%20%5CSigma%20%5Cin%20%5Cmathcal%7BZ%7D%5Ek%20%5Ctimes%20%5Cmathcal%7BZ%7D%5E%7B%7C%5CSigma%7C%7D%0) where [](https://www.codecogs.com/eqnedit.php?latex=%7C%5CSigma%7C%0) is the length of [](https://www.codecogs.com/eqnedit.php?latex=%5CSigma%0). The probabilities of each prefix obtained at each time step are the sum of the probability of non-blank plus the probability of a blank symbol



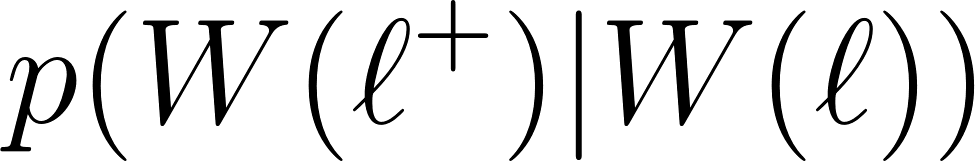
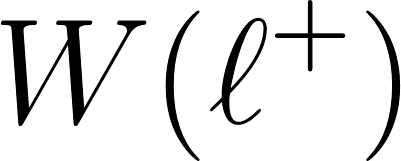
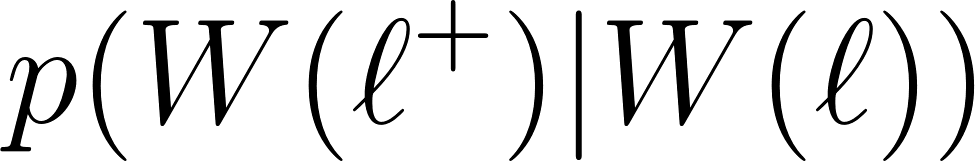
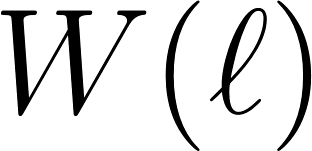
At every time step and for every prefix [](https://www.codecogs.com/eqnedit.php?latex=%5Cell%0) currently in [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0), a symbol from the alphabet [](https://www.codecogs.com/eqnedit.php?latex=%5CSigma%0) is presented to the prefix. The prefix is only extended only when the presented symbol is not a blank or a space. [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0) and [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bnext%7D%0) maintain a list of active prefixes at the previous time step and proposed prefixes at the next time step respectively, The prefix probability is given by multiplying the word insertion term by the sum of the blank and non-blank symbol probabilities.

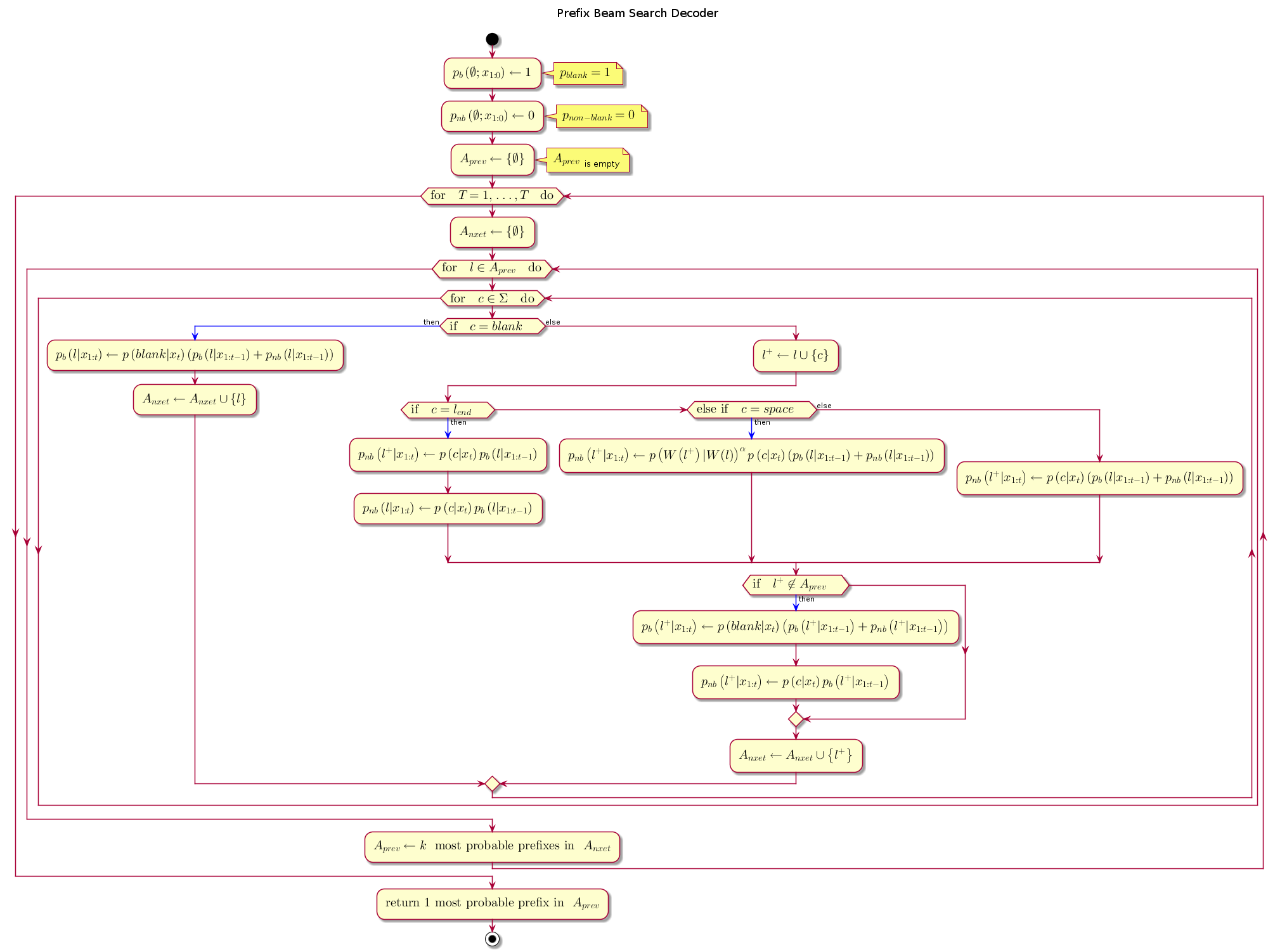
[](https://www.codecogs.com/eqnedit.php?latex=p(%5Cell%7Cx_%7B1%3At%7D)%3D(p_%7Bnb%7D(%5Cell%7Cx_%7B1%3At%7D)%2Bp_b(%5Cell%7Cx_%7B1%3At%7D))%7CW(%5Cell)%7C%5E%5Cbeta%0) - - - (26)

[](https://www.codecogs.com/eqnedit.php?latex=W(%5Ccdot)%0) is obtained by segmenting all the characters in the sequence with the space-character symbol and truncating any characters trailing the set of words in the sequence . The prefix distribution however varies slightly depending on network output character being presented.

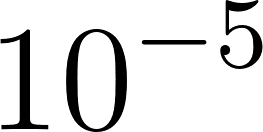
[](https://www.codecogs.com/eqnedit.php?latex=%5Cell_%7Bend%7D%0) is the variable representing the last symbol in the [](https://www.codecogs.com/eqnedit.php?latex=A_%7Bprev%7D%0). If the symbol presented is the same as [](https://www.codecogs.com/eqnedit.php?latex=%5Cell_%7Bend%7D%0) then the probability of a non-blank symbol, [](https://www.codecogs.com/eqnedit.php?latex=p_%7Bnb%7D%3D0%0). If the symbol being presented is blank then we do not extend the prefix. Finally, if the symbol being presented is a space then we invoke the language model as follows

[](https://www.codecogs.com/eqnedit.php?latex=p(%5Cell%5E%2B%7Cx_%7B1%3At%7D)%3Dp(W(%5Cell%5E%2B)%7CW(%5Cell))%5E%5Calpha(p_%7Bnb%7D(%5Cell%7Cx_%7B1%3At%7D)%2Bp_b(%5Cell%7Cx_%7B1%3At%7D))%7CW(%5Cell)%7C%5E%5Cbeta%0) - - - (27)

Note that [](https://www.codecogs.com/eqnedit.php?latex=p(W(%5Cell%5E%2B)%7CW(%5Cell))%0) is set to [](https://www.codecogs.com/eqnedit.php?latex=0%0) if the current word [](https://www.codecogs.com/eqnedit.php?latex=W(%5Cell%5E%2B)%0) is not in the lexicon. This becomes a constraint to enforce all character strings to consist only of words in the lexicon. Furthermore, [](https://www.codecogs.com/eqnedit.php?latex=p(W(%5Cell%5E%2B)%7CW(%5Cell))%0) is extended to include all the character sequences representing number of words considered by the n-gram language model by constituting the last [](https://www.codecogs.com/eqnedit.php?latex=n-1%0) words in character sequence [](https://www.codecogs.com/eqnedit.php?latex=W(%5Cell)%0).

[](http://www.plantuml.com/plantuml/img/hLPTRvim57tthxZINa1jil3QD5KtxGTCQgKzp6jam1IhndZ6h9ae_tt6XCyIGjVoG8XzVSwznuThRZDDbCuJZf1ccbFuhcZ43l2LaWJkA5513Dzea8PK0HO8sN24hguJekCRkVSTJ1Ef_sPKm_QmBxohJUbYJYCJfz9dyAu_syWj4gcceDXZh14qgmkV4_5KxfeeH4K8LM17K0Y_FCEncUSGgLWTUJQdUBxi2wdedxA7YekM5PSp3CtIOmYm3EowZkCvPaPOfmx7cXvq4QMgnBzp4iBzphjShzUNzr3_h-V3jC5p4K2Nf3XGVJv9W3bAp04p0Mt99sddOI2mCFYEFIO4vb2gYqL361OrA3kmctD4Q1Q2Zga0fxft802hdvzydjDVgnleZNPcCLxXBzOYUc0HwLWICwLTft9j4xNoIlVYwBB-cDiJSqGpjGtjM91BNF31NjGNvPbHihhtwsb1-SD53uzNE4LGJYGnBoNV5yPsPItc6Is7UeucXk-VoMvQrS0gEXPqCkrNcEz6hCGSYs5bxYkIIHBGPURw245-L5FkY_bntGTCk8n9MzH_6--qUzwuKU_9G8GiMkR3oikcyOrxpCUeFrVa-1rlumQb3aONUdM1_mUm2oLyIqyneavqbHRIFDbkZNgFfm-tfsEhXYJDD4YL-iGtdLxQmvzcOFQkRFXRqZ5k3Q6epfK0RnAg0UgVa5lp4P5AX6uDdldm-0S0)  
[PNG](http://www.plantuml.com/plantuml/img/hLPTRvim57tthxZINa1jil3QD5KtxGTCQgKzp6jam1IhndZ6h9ae_tt6XCyIGjVoG8XzVSwznuThRZDDbCuJZf1ccbFuhcZ43l2LaWJkA5513Dzea8PK0HO8sN24hguJekCRkVSTJ1Ef_sPKm_QmBxohJUbYJYCJfz9dyAu_syWj4gcceDXZh14qgmkV4_5KxfeeH4K8LM17K0Y_FCEncUSGgLWTUJQdUBxi2wdedxA7YekM5PSp3CtIOmYm3EowZkCvPaPOfmx7cXvq4QMgnBzp4iBzphjShzUNzr3_h-V3jC5p4K2Nf3XGVJv9W3bAp04p0Mt99sddOI2mCFYEFIO4vb2gYqL361OrA3kmctD4Q1Q2Zga0fxft802hdvzydjDVgnleZNPcCLxXBzOYUc0HwLWICwLTft9j4xNoIlVYwBB-cDiJSqGpjGtjM91BNF31NjGNvPbHihhtwsb1-SD53uzNE4LGJYGnBoNV5yPsPItc6Is7UeucXk-VoMvQrS0gEXPqCkrNcEz6hCGSYs5bxYkIIHBGPURw245-L5FkY_bntGTCk8n9MzH_6--qUzwuKU_9G8GiMkR3oikcyOrxpCUeFrVa-1rlumQb3aONUdM1_mUm2oLyIqyneavqbHRIFDbkZNgFfm-tfsEhXYJDD4YL-iGtdLxQmvzcOFQkRFXRqZ5k3Q6epfK0RnAg0UgVa5lp4P5AX6uDdldm-0S0) | [SVG](http://www.plantuml.com/plantuml/svg/hLPTRvim57tthxZINa1jil3QD5KtxGTCQgKzp6jam1IhndZ6h9ae_tt6XCyIGjVoG8XzVSwznuThRZDDbCuJZf1ccbFuhcZ43l2LaWJkA5513Dzea8PK0HO8sN24hguJekCRkVSTJ1Ef_sPKm_QmBxohJUbYJYCJfz9dyAu_syWj4gcceDXZh14qgmkV4_5KxfeeH4K8LM17K0Y_FCEncUSGgLWTUJQdUBxi2wdedxA7YekM5PSp3CtIOmYm3EowZkCvPaPOfmx7cXvq4QMgnBzp4iBzphjShzUNzr3_h-V3jC5p4K2Nf3XGVJv9W3bAp04p0Mt99sddOI2mCFYEFIO4vb2gYqL361OrA3kmctD4Q1Q2Zga0fxft802hdvzydjDVgnleZNPcCLxXBzOYUc0HwLWICwLTft9j4xNoIlVYwBB-cDiJSqGpjGtjM91BNF31NjGNvPbHihhtwsb1-SD53uzNE4LGJYGnBoNV5yPsPItc6Is7UeucXk-VoMvQrS0gEXPqCkrNcEz6hCGSYs5bxYkIIHBGPURw245-L5FkY_bntGTCk8n9MzH_6--qUzwuKU_9G8GiMkR3oikcyOrxpCUeFrVa-1rlumQb3aONUdM1_mUm2oLyIqyneavqbHRIFDbkZNgFfm-tfsEhXYJDD4YL-iGtdLxQmvzcOFQkRFXRqZ5k3Q6epfK0RnAg0UgVa5lp4P5AX6uDdldm-0S0) | [TXT](http://www.plantuml.com/plantuml/txt/hLPTRvim57tthxZINa1jil3QD5KtxGTCQgKzp6jam1IhndZ6h9ae_tt6XCyIGjVoG8XzVSwznuThRZDDbCuJZf1ccbFuhcZ43l2LaWJkA5513Dzea8PK0HO8sN24hguJekCRkVSTJ1Ef_sPKm_QmBxohJUbYJYCJfz9dyAu_syWj4gcceDXZh14qgmkV4_5KxfeeH4K8LM17K0Y_FCEncUSGgLWTUJQdUBxi2wdedxA7YekM5PSp3CtIOmYm3EowZkCvPaPOfmx7cXvq4QMgnBzp4iBzphjShzUNzr3_h-V3jC5p4K2Nf3XGVJv9W3bAp04p0Mt99sddOI2mCFYEFIO4vb2gYqL361OrA3kmctD4Q1Q2Zga0fxft802hdvzydjDVgnleZNPcCLxXBzOYUc0HwLWICwLTft9j4xNoIlVYwBB-cDiJSqGpjGtjM91BNF31NjGNvPbHihhtwsb1-SD53uzNE4LGJYGnBoNV5yPsPItc6Is7UeucXk-VoMvQrS0gEXPqCkrNcEz6hCGSYs5bxYkIIHBGPURw245-L5FkY_bntGTCk8n9MzH_6--qUzwuKU_9G8GiMkR3oikcyOrxpCUeFrVa-1rlumQb3aONUdM1_mUm2oLyIqyneavqbHRIFDbkZNgFfm-tfsEhXYJDD4YL-iGtdLxQmvzcOFQkRFXRqZ5k3Q6epfK0RnAg0UgVa5lp4P5AX6uDdldm-0S0) | [Edit](http://www.planttext.com/planttext?text=hLPTRvim57tthxZINa1jil3QD5KtxGTCQgKzp6jam1IhndZ6h9ae_tt6XCyIGjVoG8XzVSwznuThRZDDbCuJZf1ccbFuhcZ43l2LaWJkA5513Dzea8PK0HO8sN24hguJekCRkVSTJ1Ef_sPKm_QmBxohJUbYJYCJfz9dyAu_syWj4gcceDXZh14qgmkV4_5KxfeeH4K8LM17K0Y_FCEncUSGgLWTUJQdUBxi2wdedxA7YekM5PSp3CtIOmYm3EowZkCvPaPOfmx7cXvq4QMgnBzp4iBzphjShzUNzr3_h-V3jC5p4K2Nf3XGVJv9W3bAp04p0Mt99sddOI2mCFYEFIO4vb2gYqL361OrA3kmctD4Q1Q2Zga0fxft802hdvzydjDVgnleZNPcCLxXBzOYUc0HwLWICwLTft9j4xNoIlVYwBB-cDiJSqGpjGtjM91BNF31NjGNvPbHihhtwsb1-SD53uzNE4LGJYGnBoNV5yPsPItc6Is7UeucXk-VoMvQrS0gEXPqCkrNcEz6hCGSYs5bxYkIIHBGPURw245-L5FkY_bntGTCk8n9MzH_6--qUzwuKU_9G8GiMkR3oikcyOrxpCUeFrVa-1rlumQb3aONUdM1_mUm2oLyIqyneavqbHRIFDbkZNgFfm-tfsEhXYJDD4YL-iGtdLxQmvzcOFQkRFXRqZ5k3Q6epfK0RnAg0UgVa5lp4P5AX6uDdldm-0S0)

## Model Hyper parameters

The hidden layer matrix for each layer comprised 1024 hidden units (6.6M free parameters). The weights are initialised from a uniform random distribution having a standard deviation of 0.046875. The Adam optimisation algorithm \citep{kingma2014adam} was used with initial learning rate of [](https://www.codecogs.com/eqnedit.php?latex=10%5E%7B-5%7D%0), and a momentum of 0.95 was deployed to optimise the learning rate.

The network was trained for a total of five to fifty epochs over the training set for experiments conducted. The training time for Python GPU implementation is shown in Table \ref{tab\_c6\_01\_training}. For decoding with prefix search we use a beam size of 200 and cross-validated with a held-out set to find a good setting of the parameters [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%0) and β. Figure \ref{fig\_6\_3\_wer} shows word error rates for various GPU configurations and audio dataset sizes.

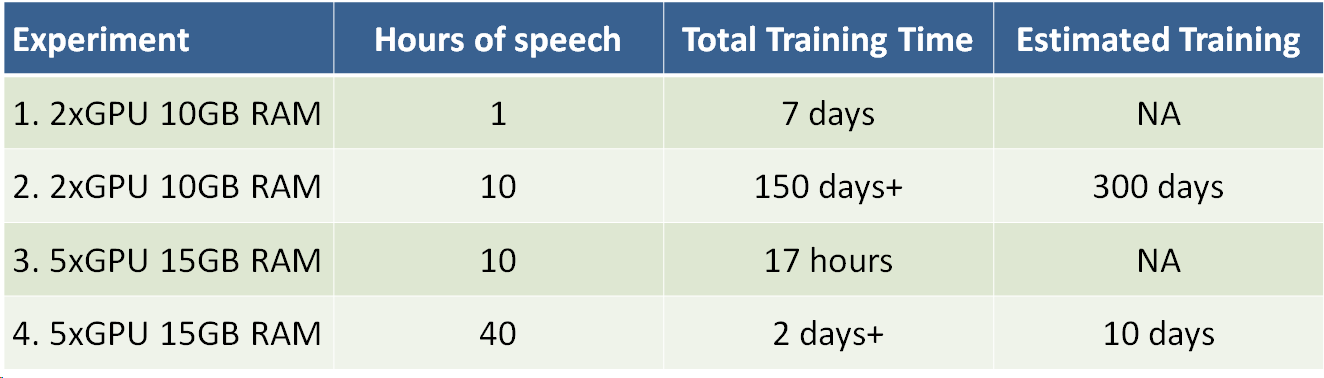
## Model baseline

The study by \cite{hannun2014first} reported successful character error rate (CER) using deep neural network (DNN), recurrent deep neural network with only forward temporal connections (RDNN), and also bi-directional recurrent neural networks (BRDNN). The models used in this their study had 5 hidden layers having either 1,824 or 2,048 hidden units in each hidden layer. For a baseline, we use a model having a similar architecture of 5 hidden units and 2048 hidden units trained on the Librespeech corpus \citep{panayotov2015librispeech, mozilla/deepspeech\_2019}.

Word Error Rates by this model were optimised after 75 epochs, learning rate of 0.0001 and a dropout rate of 15%. In addition, the language model hyper parameters for alpha and beta were 0.75 and 1.85 respectively. This achieved 8% WER. This was done using MFCC features of the training corpus.

# [Results](https://github.com/mozilla/DeepSpeech/releases)

A total of four experiments were carried out on two different GPU configurations. A set of experiments was performed a GPU configuration consisting of 2 GPUs having a total of 10 gigabytes of memory. The second set of experiments was carried out on a GPU configuration comprising 5 GPUs having a total of 15 gigabytes of memory. For each configuration two experiments were carried out on a small subset of the dataset then on a larger subset of the common voice dataset being used. The various GPU configurations along with the training times are shown in Table 1.



The output of the training produced mostly gibberish when trained in both configurations using only just one hour of training data. Training loss reduced significantly once the data was increased to ten hours of training. However word error rates (WER) only showed improvement on the 40 hours dataset.

# Discussion/Summary

The results showed that the training of the model was moving towards a very slow convergence as indicated by the slow decrements in training loss. However, we speculate that on the complete data-set, the model will not only converge but show improvements in word error rates.

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.

In this chapter the details of the novel structure having the end-to-end deep bi-RNN architecture and deep scattering features were elaborated on. The architecture which follows a five-layer structure consisting of a feedforward neural network in the first three layers and the last two consisting of recurrent structures flowing in two different directions. The network is then fed in with a 165-dimension feature vector containing deep-scattering encoding derived from a sampled raw audio file.

# Future study

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 the ability to discriminate output from the generator with the original source data.

GAN networks can have applications 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 \citep{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 reduce the resource requirements of GANs when used within the context of a GAN.

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

## Conclusion

End-to-end discriminative neural network speech models have now become a well established method in Automatic Speech Recognition.

Our Bi-directional Recurrent neural network (Bi-RNN) end-to-end system, is augmented by features derived from a deep scattering network as opposed to the standard Mel Frequency Cepstral Coefficients(MFCC) features used in state of the art acoustic models. These specialised deep scattering features, consumed by the Bi-RNN, model a light-weight convolution network. This work shows that it is possible to build a speech model from a combination of deep scattering features and a Bi-RNN. There has been no record of deep scattering features being used in end-to-end bi-RNN speech models as far as we are aware.

We show in this work that Deep Scattering features derived from wavelet filter operations on audio data produce viable candidates for end-to-end training of Automatic speech recognition models. Generative adversarial networks (GAN)Attention-based Models

References

Lyons, J. (2012). Mel Frequency Cepstral Coefficient (MFCC) tutorial. Retrieved from http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

# 