Deep Scattering and End-to-End Speech Models towards Low Resource Speech Recognition

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Iyalla John Alamina

31 October, 2019

Abstract

This thesis investigates and acknowledges the various limitations of Deep Neural Network (DNN) techniques when applied to low resource speech recognition. Various aspects of developing corpora for speech recognition systems are explored. In particular various recurrent neural network (RNN) techniques were explored to implement end-to-end speech and language models (LM). Gated Recurrent Units (GRU) RNNs were used employed for the language model for a low resourced Wakirike language while bidirectional recurrent neural networks (bi-RNNs) were used to create end-to-end speech recognition model for English language.

Previous systems employed for low resource speech recognition involving deep networks included various knowledge transfer mechanisms including hybrid hidden markov models (HMM) to deep neural networks (HMM-DNN) models and those that are HMM alone-based include subspace Gaussian Mixture Models (GMMs). These models are based on the HMM generative model and N-gram language models. However, the model developed in this thesis makes use of an end-to-end discriminative model using the Bi-RNN acoustic/speech model augmented using speech features from a specialised light weight convolution network-the deep scattering network (DSN). While the light weight DSN helped to reduce the training complexity, at the same time by focusing on end-to-end with Connectionist Temporal Classification (CTC) decoding, the speech model was compressed into a one step process rather than a three-step process requiring an Acoustic Model (AM), Language Model (LM) and phonetic dictionary. The research therefore shows that it is possible to use this compact strategy and improve upon speech features required for speech pattern recognition by deploying deep scattering network features with higher dimensional vectors when compared to traditional speech features.

Dedication

To the praise and glory of our God and of His Christ.

Acknowledgement

Copyright statement

Contents

Chapter 1

Introduction

Automatic Speech Recognition is a subset of Machine Translation that takes a sequence of raw audio information and translates or matches it against the most likely sequence of text as would be interpreted by a human language expert. In this thesis, Automatic Speech Recognition will also be referred to as ASR or speech recognition for short.

It can be argued that while ASR has achieved excellent performance in specific applications, much is left to be desired for general purpose speech recognition. While commercial applications like Google voice search and Apple Siri gives evidence that this gap is closing, there is still yet other areas within this research space that speech recognition task is very much an unsolved problem.

It is estimated that there are close to 7000 human languages in the world (Besacier et al., 2014a) and yet for only a fraction of this number have there been efforts made towards ASR. The level of ASR accuracy that have been so far achieved are based on large quantities of speech data and other linguistic resources used to train models for ASR. These models which depend largely on pattern recognition techniques degrade tremendously when applied to different languages other than the languages that they were trained or designed for (Besacier et al., 2014b, Rosenberg et al., 2017). More specifically, the collection of sufficient amounts of linguistic resources required to create accurate models for ASR are particularly laborious and time consuming sometimes extending to decades (Goldman, 2011, Stan et al., 2016). Research, therefore, geared towards alternative approaches towards developing is ASR systems that are reproducible across languages lacking the resources required

to build robust systems is apt.

1.1   ASR As a Machine Learning problem

Automatic speech recognition can be put into a class of Machine Learning problems described as sequence pattern recognition because an ASR attempts to discriminate a pattern from the sequence of speech utterances.

One immediate problem realised with this definition leads us to discuss statistical speech models that address how to handle the problem described in the following paragraph.

Speech is a complex phenomena that begins as a cognitive process and ends up as a physical process (Becchetti and Ricotti, 1998). The process of automatic speech recognition attempts to reverse engineer steps back from the physical process to the cognitive process giving rise to latent variables or mismatched data or loss of information from interpreting speech information from one physiological layer to the next.

It has been acknowledged in the research community (Deng and Li, 2013, Watanabe and Chien, 2015) that work being done in Machine Learning has enhanced the research of automatic speech recognition. Similarly any progress made in ASR usually constitutes a contribution to enhances made in the Machine Learning field. This also is an attribution to the fact that speech recognition is a sequence pattern recognition problem. Therefore techniques within speech recognition could be applied generally to sequence pattern recognition problems.

The two main approaches to Machine Learning problems historically involve two methods rooted in statistical science. These approaches are generative and discriminative models. From a computing science perspective, the generative approach is a brute-force approach while the discriminative model uses a rather heuristic approach to Machine Learning. This chapter presents the introductory ideas behind these two approaches and establishes the motivation for the proposed models used in this research for low resource speech recognition, as well as introducing the Wakirike language as the motivating language case study.

1.2 Generative-Discriminative Speech Models disambiguation

In the next chapter, the Hidden Markov Model (HMM) is examined as a powerful and major driver behind generative modelling of sequential data like speech. Generative models are data-sensitive models because they are derived from the data by accumulating as many different features which can be seen and make generalisations based on the features seen. The discriminative model, on the other hand, has a heuristic approach to form a classification. Rather than using features of the data directly, the discriminative method attempts to characterise the data into features. It is possible to conclude that the generative approach uses a bottom-to-top strategy starting with the fundamental structures to determine the overall structure, while the discriminative method uses a top-to-bottom approach starting with the big picture and then drilling down to discover the fundamental structures.

Ultimately, generative models for Machine Learning learning can be interpreted mathematically as a joint distribution that produces the highest likelihood of outputs and inputs based on a predefined decision function. The outputs for speech recognition being the sequence of words and the inputs for speech being the audio waveform or equivalent speech sequence.

$$d_y(\mathbf{x}; \lambda) = p(\mathbf{x}, y; \lambda) = p(\mathbf{x}|y; \lambda)p(y; \lambda) \tag{1.1}$$

where $d_y(\mathbf{x}; \lambda)$ is the decision function of y for data labels $\mathbf{x}$. This joint probability expression given as $p(\mathbf{x}|y; \lambda)$ can also be expressed as the conditional probability product in equation (1.1). In this equation, λ predefines the nature of the distribution (Deng and Li, 2013) referred to as model parameters.

Similarly, Machine Learning discriminative models are described mathematically as the conditional probability defined by the generic decision function below:

$$d_y(\mathbf{x}; \lambda) = p(y|\mathbf{x}; \lambda) \tag{1.2}$$

It is clearly seen that the discriminative paradigm follows a much more direct approach to pattern recognition. Although this approach appears cumbersome to

model, this research leans towards this direct approach. However, what the discriminative model gains in discriminative modularity, it loses in the model parameter estimation of (λ) in equation (1.1) and (1.2). As this research investigates, although the generative process is able to generate arbitrary outputs from learned inputs, its major drawback is the direct dependence on the training data from which the model parameters are learned. Specific characteristics of various Machine Learning models are reserved for later chapters, albeit the heuristic nature of the discriminative approach, which means not directly dependent on the training data, gains over the generative approach as discriminative models are able to better compensate for latent variablesGales et al. (2012).

In the case of speech signals, information is lost in training data due to the physiologic transformations of the intended speech message as it moves from one speech mechanism mentioned in section 1.1 to the next. The theme of pattern recognition through arbitrary layers of complexity is reinforced in the notion of deep learning defined in Deng et al. (2014) as an attempt to learn patterns from data at multiple levels of abstraction. Thus while shallow Machine Learning models like Hidden Markov Models (HMMs) define latent variables for fixed layers of abstraction, deep Machine Learning models handle hidden/latent information for arbitrary layers of abstraction determined heuristically. As deep learning mechanisms are typically implemented using deep neural networks, this work applies deep recurrent neural networks as an end-to-end discriminative classifier for speech recognition. This is a so called "end-to-end model" because it adopts the top-to-bottom Machine Learning approach. Unlike the typical generative classifiers that require sub-word acoustic models, the end-to-end models develop algorithms at higher levels of abstraction as well as the lower levels of abstraction. In the case of the deep-speech model (Hannun et al., 2014b) utilised in this research, the levels of abstraction include sentence/phrase, words and character discrimination. A second advantage of the end-to-end model is that because the traditional generative models require various stages of modeling including an acoustic, language and lexicon, the end-to-end discriminating multiple levels of abstractions simultaneously only requires a single stage process, greatly reducing the amount of resources required for speech recognition. From a low resource language perspective this is an attractive feature meaning that the model

can be learned from an acoustic only source without the need of an acoustic model or a phonetic dictionary. In theory this deep learning technique is sufficient in itself without a language model. However, applying a language model was found to serve as a correction factor further improving recognition results (Hannun et al., 2014a).

1.3 Low Resource Languages

A second challenge observed in complex Machine Learning models for both generative as well as discriminative learning models is the data intensive nature required for robust classification models. Saon et al. (2015) recommends around 2000 hours of transcribed speech data for a robust speech recognition system. As is covered in the next chapter, for new languages, which are low in training data such as transcribed speech, there are various strategies devised for low resource speech recognition. Besacier et al. (2014a) outlines various matrices for bench-marking low resource languages. From the generative speech model interest perspective, reference is made to languages having less than ideal data in transcribed speech, phonetic dictionary and a text corpus for language modelling. For end-to-end speech recognition models interests, the data relevant for low resource evaluation is the transcribed speech and a text corpus for language modelling. It is worth noting that it was observed (Besacier et al., 2014a) that speaker-base often does not affect a language resource status of a language and was often observed that large speaker bases could in fact lack language/speech recognition resources and that some languages having small speaker bases did in fact have sufficient language/ speech recognition resources.

Speech recognition methods looked at in this work are motivated by the Wakirike language discussed in the next section, which is a low resource language by definition. Thus this research looked at low research language modelling for the Wakirike language from a corpus of Wakirike text available for analysis. However, due to the insufficiency of transcribed speech for the Wakirike language, English language was substituted and used as a control variable to study low resource effects of a language when exposed to speech models developed in this work.

1.4 The Wakirike Language

The Wakirike municipality is a fishing community comprising 13 districts in the
Niger Delta area of the country of Nigeria in the West African region of the conti-
nent of Africa. The first set of migrants to Wakirike settled at the mainland town
of Okrika between AD860 and AD1515 at the earliest. These early settlers had
migrated from Central and Western regions of the Niger Delta region of Nigeria.
As the next set of migrants also migrated from a similar region, when the second
set of migrants met with the first settlers they exclaimed "we are not different" or
"Wakirike" (S., 2008).

Although the population of the Wakirike community from a 1995 report (Simons
and Fennig, 2018) is about 248,000, the speaker base is significantly less than stip-
ulated. The language is classified as Niger-Congo and Ijoid languages. The writing
orthography is Latin and the language status is 5 (developing) (Simons and Fennig,
2018). This means that although the language is not yet an endangered language,
it still isn't thriving and it is being passed on to the next generation at a limited
rate.

The Wakirike language was the focus for this research. And End-to-end deep
neural network language model was built for the Wakirike language based on the
availability of the new testament bible printed edition that was available for pro-
cessing. The corpus utilized for this thesis work is approximately 9,000 words.

1.5 Research aims and objectives

In this research we develop speech processing models and language models which
deliver robust deep and recurrent neural network implementations towards low re-
source speech recognition. In particular, we develop a language model based on
Gated Recurrent Units (GRU) for the Wakirike language and a bi-directional recur-
rent neural network (Bi-RNN) speech model for the English language.

1.6   Main Contribution to knowledge

This work contributes a neural language model for the low resourced language of
Wakirike. At the same time this work implements a unique combination of end-
to-end deep recurrent neural network models with a pristine and state of the art
audio signal processing mechanism involving a hierarchical scattering network to
engineer features to compete with current acoustic and deep architectures for speech
recognition.

1.7   Thesis outline

The outline of this report follows the development of an end-to-end speech recog-
niser and develops the theory based on the building blocks of the final system.
Chapter two introduces the speech recognition pipeline and the generative speech
model. Chapter two outlines the weaknesses in the generative model and describes
some of the Machine Learning techniques applied to improve speech recognition
performance.

Various Low speech recognition methods are reviewed and the relevance of this
study is also highlighted. Chapter three describes Recurrent Neural Networks
(RNNs). Starting with Multi-Layer Perceptrons (MLPs), we go on to specialised
recurrent neural networks including long short-term memory (LSTM) networks and
the Gated Recurrent Units (GRU) are detailed. These recurrent neural network
units form building blocks of the language model for Wakirike language implemented
in this work.

Chapter Four explains the wavelet theorem as well as the deep scattering spec-
trum. The chapter develops the theory from Fourier transform and details the
significance of using the scattering transform as a feature selection mechanism for
low resource recognition.

Chapters five and six is a description of the models developed by this thesis and
details the experiment setup along with the results obtained. Chapters seven is a
discussion of the result and chapter 8 are the recommendations for further study.

Chapter 2

Low Resource Speech Models, End-to-end models and the scattering network

The speech recogniser developed in this thesis is based on an end-to-end discriminative deep recurrent neural network. Two models were developed. The first model is a Gated-Recurrent-Unit Recurrent Neural network (GRU-RNN) was used to develop a character-based language model, while the second recurrent neural network is a Bi-Directional Recurrent neural Network (BiRNN) used as an end-to-end speech model capable of generating word sequences based on learned character sequence outputs. This chapter describes the transition from generative speech models to these discriminative end-to-end recurrent neural network models. Low speech recognition strategies are also discussed and the contribution to knowledge gained by using character-based discrimination as well as introducing deep scattering features to the bi-RNN speech model is brought to light.

2.1   Speech Recognition Overview

Computer speech recognition takes raw audio speech and converts it into a sequence of symbols. This can be considered as an analog to digital conversion as a continuous signal becomes discretised. The way this conversion is done is by breaking up the audio sequence into very small packets referred to as frames and developing

discriminating parameters or features for each frame. Then using the vector of features as input to the speech recogniser.

A statistical formulation (Young et al., 2002) for the speech recogniser follows given that each discretised output word in the audio speech signal is represented as a vector sequence of frame observations defined in the set $\mathbf{O}$ such that

$$\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T. \tag{2.1}$$

At each discrete time t, we have an observation $\mathbf{o}_t$, which is, in itself is a vector in R^D. From the conditional probability, it can be formulated that certain word sequences from a finite dictionary are most probable given a sequence of observations. That is:

$$arg \max_t \{P(w_i|\mathbf{O})\} \tag{2.2}$$

As we describe in the next section on speech recognition challenges, there is no straightforward analysis of $P(w_i|\mathbf{O})$. The divide and conquer strategy therefore employed uses Bayes formulation to simplify the problem. Accordingly, the argument that maximises the probability of an audio sequence given a particular word multiplied by the probability of that word is equivalent to the original posterior probability required to solve the isolated word recognition problem. This is summarised by the following equation

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})} \tag{2.3}$$

That is, according to Bayes' rule, the posterior probability is obtained by multiplying a certain likelihood probability by a prior probability. The likelihood in this case, $P(\mathbf{O}|w_i)$, is obtained from a Hidden Markov Model (HMM) parametric model such that rather than estimating the observation densities in the likelihood probability, these are obtained by estimating the parameters of the HMM model. The HMM model explained in the next section gives a statistical representation of the latent variables of speech at a mostly acoustic level.

The second parameter in the speech model, interpreted from Bayes' formula, is the prior probability of a given word. This aspect of the model is the language model which is reviewed in section 2.2.1.
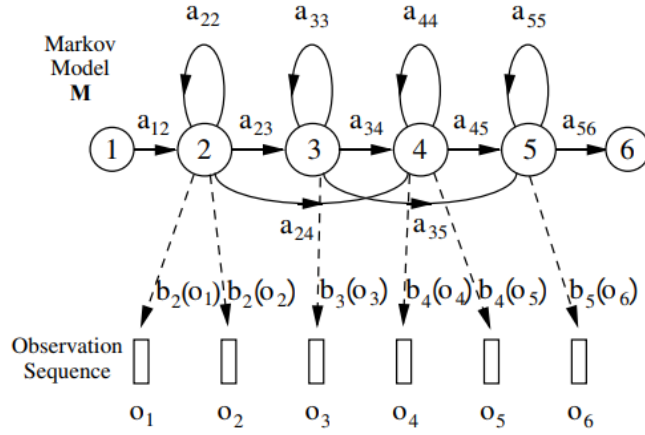
Figure 2.1: HMM Generative Model
Young et al. (2002)

2.1.1 HMM-based Generative speech model

A HMM represents a finite state machine where a process transits a sequence of
states from a set of fixed states (Gales et al., 2008, Young et al., 2002). The overall
sequence of transitions will have a start state, an end state and a finite number of
intermediate states all within the set of finite states. Each state transition emits an
output observation that represents the current internal state of the system.

In an HMM represented in Figure 2.1 there are two important probabilities. The
first is the state transition probability given by a_{ij} this is the probability to move
from state i to state j. The second probability b_j is the probability that an output
observation is emitted when in a particular state.

Where $\mathbf{O}$, are the output observations and M is the HMM. Given that X rep-
resents the sequence of states transitioned by a process, a HMM defines the joint
probability of X and the output probabilities given the HMM in the following rep-
resentation:

$$P(\mathbf{O}|M) = \sum_{X} a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)} \tag{2.4}$$

Generally speaking, the HMM formulation presents 3 distinct challenges. The
first is the likelihood of a sequence of observations given in equation 2.4 above. The
next two, described later, is the inference and the learning problem. While the
inference problem determines the sequence of steps given the emission probabilities,
the learning problem determines the HMM parameters, that is the initial transition
and emission probabilities of the HMM model.

For the case of the inference problem, the sequence of states can be obtained by determining the sequence of states that maximises the probability of the output sequences.

2.1.2 Challenges of Speech Recognition

The realised symbol is assumed to have a one to one mapping with the segmented raw audio speech. However, the difficulty in computer speech recognition is the fact that there is a significant amount of variation in speech that would make it practically intractable to establish a direct mapping from segmented raw speech audio to a sequence of static symbols. The phenomena known as co articulation has it that there are several different symbols having a mapping to a single waveform of speech in addition to several other varying factors including the speaker mood, gender, age, the medium of speech transduction, the room acoustics, et cetera.

Another challenge faced by automated speech recognisers is the fact that the boundaries of the words is not apparent from the raw speech waveform. A third problem that immediately arises from the second is the fact that the words from the speech may not strictly follow the words in the selected vocabulary database. Such occurrence in speech recognition research is referred to as out of vocabulary (OOV) terms. It is reasonable to approach these challenges using a divide and conquer strategy. In this case, the first step would be to create assumption that somehow word boundaries can be determined. This first step in speech recognition is referred to as the isolated word recognition case (Young et al., 2002).

2.1.3 Challenges of low speech recognition

Speech recognition for low resource languages poses another distinct set of challenges. In chapter one, low resource languages were described to be languages lacking in resources required for adequate Machine Learning of models needed for generative speech models. These resources are described basically as a text corpus for language modelling, a phonetic dictionary and transcribed audio speech for acoustic modelling. Figure 2.2, illustrates how resources required for speech recognition are utilised. It is observed that in addition to the three resources identified other processes are required for the speech decoder to function normally. For example,
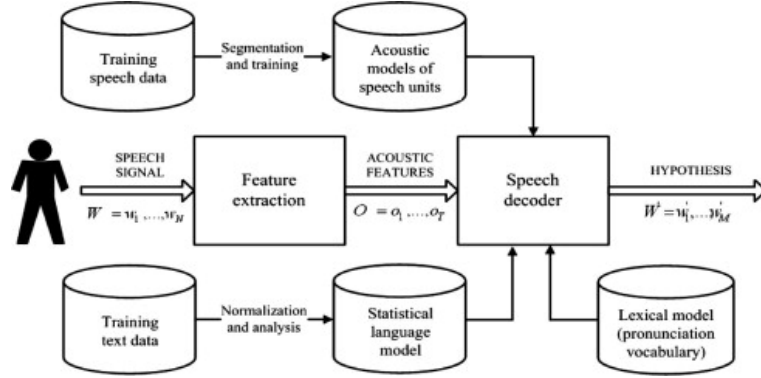
Figure 2.2: Automatic Speech Recognition Pipeline
Besacier et al. (2014a)

aligned speech would also need to be segmented into speech utterances to ensure
that the computer resources are used conservatively.

In terms of data collection processing Besacier et al. (2014a) enumerates the challenges for developing low resource ASR systems to include the fact that phonologies
(or language sound systems) differ across languages, word segmentation problems,
fuzzy grammatical structures, unwritten languages, lack of native speakers having
technical skills and the multidisciplinary nature of ASR constitute impedance to
ASR system building.

2.2 Low Resource Speech Recognition

In this system building speech recognition research, the focus was on the development of a language model and an end-to-end speech model comparable in performance to state of the art speech recognition system consisting of an acoustic
model and a language model. Low resource language and acoustic modelling is now
reviewed keeping in mind that little work has been done on low-resource end-to-
end speech modelling when compared to general end-to-end speech modelling and
general speech recognition as a whole.

From an engineering perspective, a practical means of achieving low resource
speech modelling from a language rich in resources is through various strategies of
the Machine Learning sub-field of transfer learning.

Transfer learning takes the inner representation of knowledge derived from training an algorithm used from one domain and applying this knowledge in a similar

domain having different set of system parameters. Early work of this nature for speech recognition is demonstrated in (Vu and Schultz, 2013) where multi-layer perceptrons were used to train multiple languages rich in linguistic resources. In a later section entitled "speech recognition on a budget", a transfer learning mechanism involving deep neural networks from (Kunze et al., 2017) is described.

2.2.1 Low Resource language modelling

General language modelling is reviewed and then Low resource language modelling is discussed in this section. In section 2.1, recall from equation 2.3, the general speech model influenced by Bayes' theorem.

$$P(w_i|\mathbf{O}) = \frac{P(\mathbf{O}|w_i)P(w_i)}{P(\mathbf{O})} \tag{2.5}$$

The speech recognition model is a product of an acoustic model (likelihood probability),$P(\mathbf{O}|w_i)$ and the language model (prior probability),$P(w_i)$. The development of language models for speech recognition is discussed in Juang and Furui (2000) and Young (1996).

Language modelling formulate rules that predict linguistic events and can be modelled in terms of discrete density $P(W)$, where $W = (w_1, w_2, ..., w_L)$ is a word sequence. The density function $P(W)$ assigns a probability to a particular word sequence W. This value determines how likely the word is to appear in an utterance. A sentence with words appearing in a grammatically correct manner is more likely to be spoken than a sentence with words mixed up in an ungrammatical manner, and, therefore, is assigned a higher probability. The order of words therefore reflect the language structure, rules, and conventions in a probabilistic way. Statistical language modeling therefore, is an estimate for $P(W)$ from a given set of sentences, or corpus.

The prior probability of a word sequence $\mathbf{w} = w_1, \ldots, w_k$ required in equation (2.2) is given by:

$$P(\mathbf{w}) = \prod_{k=1}^{K} P(w_k|w_{k-1}, \ldots, w_1) \tag{2.6}$$

The N-gram model is formed by the conditioning of the word history in equation

2.6. This therefore becomes:

$$P(\mathbf{w}) = \prod_{k=1}^{K} P(w_k | w_{k-1}, w_{k-2}, \ldots, w_{k-N+1}) \tag{2.7}$$

N is typically in the range of 2-4.

N-gram probabilities are estimated from training corpus by counting N-gram occurrences. This is plugged into maximum likelihood (ML) parameter estimate. For example, Given that N=3 then the probability that three words occurred is assuming $C(w_{k-2}w_{k-1}w_k)$ is the number of occurrences of the three words $C(w_{k-2}w_{k-1})$ is the count for $w_{k-2}w_{k-1}w_k$ then

$$P(w_k | w_{k-1}, w_{k-2}) \approx \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})} \tag{2.8}$$

The major problem with maximum likelihood estimation scheme is data sparsity. This can be tackled by a combination of smoothing techniques involving discounting and backing-off. The alternative approach to robust language modelling is the so-called class based models (Brown et al., 1992, **?**) in which data sparsity is not so much an issue. Given that for every word w_k, there is a corresponding class c_k, then,

$$P(\mathbf{w}) \prod_{k=1}^{K} P(w_k | c_k) p(c_k | c_{k-1}, \ldots, c_{k-N+1}) \tag{2.9}$$

In 2003, Bengio et al. (2003) proposed a language model based on neural Multi-Layer Perceptrons (MLPs). These MLP language models resort to a distributed representation of all the words in the vocabulary such that the probability function of the word sequences is expressed in terms of these word-level vector representations. The result of the MLP-based language models was found to be, in cases for models with large parameters, performing better than the traditional n-gram models.

Improvements over the MLPs still using neural networks over the next decade include works of Luong et al. (2013), Mikolov et al. (2011), Sutskever et al. (2014), involved the utilisation of deep neural networks for estimating word probabilities in a language model. While a Multi-Layer Perceptron consists of a single hidden layer, in addition to the input and output layers, a deep network, in addition to having several hidden layers, are characterised by complex structures that render the architecture

beyond the basic feed forward nature. Particularly, for Recurrent Neural Network (RNN) architectures, we also have some feedback neurons in addition to the forward neurons where data flows in the reverse direction, from output to input.

Furthermore, the probability distributions in these deep neural networks were either based upon word or sub-word models, this time having representations which also conveyed some level of syntactic or morphological weights to aid in establishing word relationships. These learned weights are referred to as token or unit embedding (Pennington et al., 2014).

For the neural network implementations so far seen, a large amount of data is required due to the nature of words to have large vocabularies, even for medium-scale speech recognition applications. Kim et al. (2016) on the other hand took a different approach to language modelling taking advantage of the long-term sequence memory of long-short-term memory cell recurrent neural network (LSTM-RNN) to model a language based on characters rather than on words. This greatly reduced the number of parameters involved and therefore the complexity of implementation. This method is forms the basis of the Wakirike language model implementation in this work due to the low resource constraints gains made when using a character-level language model.

Other low resource language modelling strategies employed for the purpose of speech recognition was demonstrated by Xu and Fung (2013). The language model developed in that work was based on phrase-level linguistic mapping from a high resource language to a low resource language using a probabilistic model implemented using a weighted finite state transducer (WFST). This method uses WFST rather than a neural network due to scarcity of training data required to develop a neural network. However, it did not gain from the high non linearity ability of a neural network model to discover hidden patterns in data, being a shallower Machine Learning architecture.

The language model implemented in this thesis report uses a character-based Neural network language model that employs a recurrent neural network similar to that of Kim et al. (2016), however based on Gated Recurrent Units (GRU) RNNs Cho et al. (2014), for the Okrika language which is a low resource language, bearing in mind that the character level network will reduce the number of parameters

required for training, just enough to develop a working language model for the
purpose of speech recognition.

2.2.2 Low Resource Acoustic and speech modelling

Two transfer learning techniques for acoustic modelling investigated by Povey et al.
(2011) and Ghoshal et al. (2013) respectively include the sub-space Gaussian mixture
models (SGMMs) and the use of pretrained hidden layers of a deep neural network
trained multilingually as a means to initialise weights for an unknown language.
This second method of low resource modelling has been informally referred to as the
swap-hat method.

Recall that one of the challenges associated with new languages is that phonetic
systems differ from one language to another. Transfer learning approaches attempt
however to recover patterns common to seemingly disparate systems and model
these patterns.

For phonetic systems, based on the premise that sounds are produced by ap-
proximate movements and positions of articulators comprising the human speech
sound system is common for all humans. It is possible to model dynamic movement
from between various phones as tied state mixture of Gaussians. These dynamic
states modelled using Gaussian mixture models or GMM are also called senones.
Povey et al. (2011) postulated a method to factorize these Gaussian mixtures into
a globally shared set of parameters that are non-dependent individual HMM states.
These factorisations model senones that are not represented in original data and
thought to be a representation of the overall acoustic space. While preserving indi-
vidual HMM states, the decoupling of the shared space and its reuse makes SGMMs
a viable candidate for transfer learning of acoustic models for new languages.

The transfer learning procedure proposed in Ghoshal et al. (2013) employed the
use of deep neural networks, in particular deep belief networks (Bengio et al., 2007).
Deep Belief Networks are pretrained, layer-wise stacked Restricted Boltzmann Ma-
chines (RBMs)(Smolensky, 1986). The output of this network trained on senones
correspond to HMM context dependent states. However, by decoupling hidden lay-
ers from outer and output layers and fine-tuned to a new language, the network is
shown to be insensitive to the choice of languages analogous to global parameters of

SGMMs. The 7-layer, 2000 neuron per layer network used did not utilise a bottle-neck layer corresponding to triphone states trained on MFCC features (Grezl and Fousek, 2008).

2.3 Groundwork for low resource end-to-end speech modelling

The underpinning notion of this work is firstly a departure from the bottom-to-top baggage that comes as a bye-product of the generative process sponsored by the HMM-based speech models so that we can gain from simplifying the speech pipeline from acoustic, language and phonetic model to just a speech model that approximates the same process. Secondly, the model developed seeks to overcome the data intensity barrier and was seen to achieve measurable results for GRU RNN language models. Therefore adopting the same character-based strategy, this research performed experiments using the character-based bi-directional recurrent neural networks (BiRNN). However, BiRNNs researchers have found them as other deep learning algorithms, as being very data intensiveHannun et al. (2014a). The next paragraphs introduce Deep-speech BiRNNs and the two strategies for tackling the data intensity drawback as related with low resource speech recognition.

2.3.1 Deep speech

Up until recently, speech recognition research has been centred on improvements of the HMM-based acoustic models. This has included a departure from generative training of HMM to discriminative training (Woodland and Povey, 2000) and the use of neural network precursors to initialise the HMM parameters (Mohamed et al., 2012). Although these discriminative models brought improvements over generative models, being HMM dependent speech models they lacked the end-to-end nature. This means that they were subject to training of acoustic, language and phonetic models. With the introduction of the Connectionist Temporal Classification (CTC) loss function, Graves and Jaitly (2014) finally found a means to end-to-end speech recognition departing from HMM-based speech recognition.

The architecture of the Deep-speech end-to-end speech recognition model Hannun et al. (2014b) follows an end-to-end Bi-directional Recurrent Neural Network (BiRNN) and CTC loss function (Graves et al., 2006). The CTC loss function uses a modified beam search to sum over all possible input-output sequence alignments thereby maximising the likelihood of the output sequence characters.

2.3.2 Speech Recognition on a low budget

In this section, a recent transfer learning speech model (Kunze et al., 2017) that has some characteristics similar to the speech model developed in this thesis is reviewed. The end-to-end speech model described by Kunze et al. (2017) is based on that developed by Collobert et al. (2016) and is based on deep convolutional neural networks rather than the Bi-RNN structure proposed by this work. In addition it uses a loss function based on the AutoSegCriterion which is claimed to work competitively with raw audio waveform without any preprocessing. The main strategy for low resource management in their system was the freezing of some layers within the convolutional network layer. The low resource mechanisms used in this work includes the use of a unique scattering network being used as input features for the BiRNN model. The fascinating similarity between the end-to-end BiRNN speech model developed in this work and the transfer learning model in Kunze et al. (2017) is the fact that the scattering network input are equivalent to the output of a light-weight convolutional neural network Hannun et al. (2014b). Therefore the proposed system then approximates a combination of a recurrent neural network as well as a convolution neural network without the overhead of actually training a convolutional neural network (CNN).

Introduction of the unique scattering network is discussed in the next section. It is worthy to note however that Kunze et al. (2017) uses a CNN network only while (Amodei et al., 2016) uses both RNN and CNN network. The speech model in this thesis uses a BiRNN model and combines an RNN model with the scattering layer which represents a light-weight low resource friendly pseudo enhanced CNN backing. What is meant by pseudo enhanced CNN backing is reserved for the next section, however, therefore, the proposed speech model in this thesis stands to gain from an enhanced but lightweight CNN combined with RNN learning.

2.3.3 Adding a Scattering layer

In Machine Learning, training accuracy is greatly improved through a process described as feature engineering. In feature engineering, discriminating characteristics of the data is enhanced at the same time non-distinguishing features constituting noise is removed or attenuated to a barest minimum. A lot of the components signal speech signal are due to noise in the environment as well as signal channel distortions such as losses due to conversion from audio signals to electrical signal in the recording system.

In Figure 2.2, feature engineering is done at the feature extraction stage of the ASR pipeline. It has been shown that a common technique using Mel-frequency cepstral coefficients (MFCCs) (Davis and Mermelstein, 1980) can represent speech in a stable fashion that approximate how the working of the human auditory speech processing and is able to filter useful components in the speech signal required for human speech hearing. Similar feature processing schemes have been developed include Perceptual Linear Prediction (PLP) (Hermansky, 1990) and RASTA (Hermansky and Morgan, 1994).

The scattering spectrum defines a locally translation invariant representation of a signal resistant to signal deformation over extended periods of time spanning seconds of the signal (Andén and Mallat, 2014). While Mel-frequency cepstral coefficients (MFCCs) are cosine transforms of Mel-frequency spectral coefficients (MFSCs), the scattering operator consists of a composite wavelet and modulus operation on input signals.

Over a fixed time, MFSCs measure signal energy having constant Q bandwidth Mel-frequency intervals. This procedure is susceptible to time-warping signal distortions since these information often reside in the high frequency regions discarded by Mel-frequency intervals. As time-warping distortions is not explicit classifier objective when developing these filters, there is no way to recover such information using current techniques.

In addition, short time windows of about 20 ms are used in these feature extraction techniques since at this resolution speech signal is mostly locally stationary. Again, this resolution adds to the loss of dynamic speech discriminating information on signal structures that are non-stationary at this time interval. To minimize

this loss Delta-MFCC and Delta-Delta-MFCCs (Furui, 1986) are some of the means developed to capture dynamic audio signal characterisation over larger time scales.

By computing multi-scale co-occurrence coefficients from a wavelet-modulus operation Andén and Mallat (2011), show that non-stationary behaviour lost by MFSC coefficients is captured by the scattering transform multi scale co-occurrence coefficients and the scattering representation includes MFSC-like measurements. Together with higher-order co-occurrence coefficients, deep scattering spectrum coefficients represents audio signals similar to models based on cascades of constant-Q filter banks and rectifiers. In particular, second-order co-occurrence coefficients carry important signal information capable of discriminating dynamic information lost to the MFCC analog over several seconds and therefore a more efficient discriminant than the MFCC representation. Second-order co-occurrence coefficients calculated by cascading wavelet filter banks and rectified using modulus operators have been evaluated as equivalent to a light-weight convolutional neural networks whose output posteriors are computed at each layer instead of only at the output layer Mallat (2016).

The premise for this work is that low speech recognition can be achieved by having higher resolution features for discrimination as well as using an end-to-end framework to replace some of the cumbersome and time-consuming hand-engineered domain knowledge required in the standard ASR pipeline. In addition, this research work makes contributions to the requirements for the two tracks specified in the Zero Resource challenge of 2015 (Versteegh et al., 2015). The first requirement is sub-word modelling satisfied with using deep scattering network and the second that of spoken term discovery criteria being satisfied with the end-to-end speech model supplemented with a language model.

2.4 Methodology

System building methodology (Nunamaker Jr et al., 1990) for speech recognition systems require models to be evaluated against speech recognition Machine Learning metrics. For language models, perplexity metric was used for evaluation. Bleu has also been used as a metric for evaluating language models.

Perplexity measures the complexity of a language that the language model is designed to represent (Jelinek, 1976). In practice, the entropy of a language with an N-gram language model $P_N(W)$ is measured from a set of sentences and is defined as

$$H = \sum_{\mathbf{W} \in \Omega} P_N(\mathbf{W}) \tag{2.10}$$

where Ω is a set of sentences of the language. The perplexity, which is interpreted as the average word-branching factor, is defined as

$$PP(W) = 2^H \tag{2.11}$$

where H is the average entropy of the system or the average log probability defined as

$$H = -\frac{1}{N} \sum_{i=1}^{N} [log_2 P(w_1, w_2 \dots w_N)] \tag{2.12}$$

For a bi gram model therefore, equation (2.12) becomes

$$PP(W) = 2^H = 2^{-\frac{1}{N} \sum_{i=1}^{N} [log_2 P(w_1, w_2 \dots w_N)]} \tag{2.13}$$

After simplifying we have

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \tag{2.14}$$

Full speech recognition pipelines are usually evaluated against the Word Error Rate (WER). WER is computed as follows:

$$WER = \frac{I + D + R}{WC} \times 100 \tag{2.15}$$

Here I, D, and R are wrong insertions, deletions and replacements respectively and WC is the word count.

Metrics used for low speech recognition in the zero speech challenge (Versteegh et al., 2015) includes the ABX metric. Other common speech recognition error metrics following a similar definition as the Word Error Rate (WER) are Character Error Rate (CER), Phoneme Error Rate (PER) and Syllabic Error Rate (SyER)

and sentence error rate (SER).

Chapter 3

Recurrent Neural Networks in Speech Recognition

The HMM model described in the Chapter 2 uses a divide and conquer strategy which has also been described as a generative Machine Learning algorithm in which we use the smaller components represented by the HMM to learn the entire speech process. In previous chapters, this was referred to as the bottom-top strategy. The discriminative method however uses the opposite mechanism. Instead of using the building blocks of speech to determine speech parameters of a HMM, the discriminative strategy determines the posterior probability directly using the joint probability distribution of the parameters involved in the discriminative process. The discriminative approach, discussed in this chapter focuses in on Neural network architectures.

3.1   Neural network architecture

The building block of a neural network simulates a combination of two consecutive linear and non-linear operations having many inputs interconnected with the linear portion of the network. This rudimentary structure is described by McCullough and Pitts (1942) in Cowan (1990) as the Perceptron in Figure 3.1

The linear operation is the sum of the products of the input feature and a weight vector set. This vector sum of products is referred to as an affine transformation or operation. The non linear operation is given by any one of a selection of nonlinear
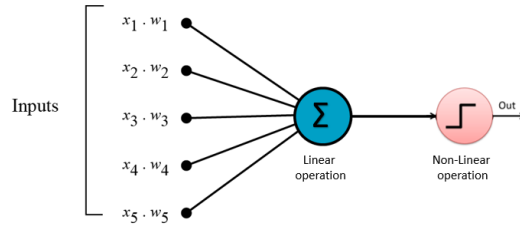
Figure 3.1: Perceptron
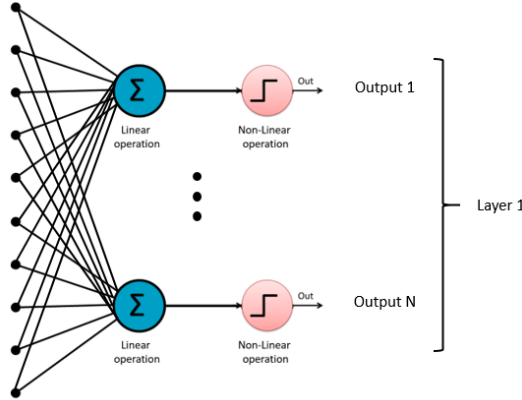
Figure 3.2: Neural network

functions. In Figure 3.2 this is shown as a step function. The step function is
activated (becomes 1) whenever the output of the linear function is above a certain
threshold, otherwise remains at 0. A simple neural network of perceptrons is formed
by stacking the perceptrons into an interconnected layer as shown in the Figure 3.2
:

In this regime each combination of linear operation followed by a non linear
operation is called a neuron and the total number of neurons in the layer formed is
termed as M-number of neurons in the layer.

3.1.1 Multi-layer Perceptron (MLP)

The multilayer Perceptron or MLP extends the basic Perceptron structure by adding
one or more hidden layers. These hidden layers comprise the outputs of one layer
becoming the input of the next layer. In the simplest case having one hidden layer,
the output of layer 1 becomes the input of the final output layer. In comparison, the
Perceptron is a one dimensional structure having one or more linear and non linear
combination outputs, while the multilayer Perceptron is a 2-dimensional structure
having one or more hidden layers of N linear and non-linear combination outputs.

Mathematically speaking the output of each layer of an MLP having N inputs and M neurons is given by

$$z_j = h(b_j) = \frac{1}{1 + e^{-b_j}}$$

(3.1)

is the non-linear function while is the linear function given by:

$$b_j = \sum_{i=0}^{N} w_{ji}^{(1)} \qquad j = 1, 2, \ldots, M$$

(3.2)

For each layer in the MLP, the zeroth input value x_0 is 1 indicating a bias term. This bias term is used in the neural network to ensure regularised and expected behaviour of the neural network. In this example the non-linear step function is given by a more complex exponential. In the next section the nonlinear functions for a multilayer Perceptron is derived.

3.1.2   Sigmoid and soft-max Activation Function

The combination of the linear function and the non linear function in the neural network could be said to be transformation of an algebraic problem to a probabilistic function. In this case the "step" function is a squashing sigmoid-shaped function that converts the inputs into a Naive Bayes function evaluating the probability that an output belongs to any of the output classes (C_y) given the data $(\mathbf{x})$.

$$p(C_1|\mathbf{x}) = f(a) = f(\mathbf{w}^\top \mathbf{x} + w_0)$$

(3.3)

In a two class problem with classes C_1 and C_2, the posterior probability of class C_1 is expressed using Bayes's theorem

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(\mathbf{x}|C_2)p(C_2)}$$

(3.4)

Dividing through by $p(\mathbf{x}|C_1)p(C_1)$ gives us

$$p(C_1|(x) = \frac{1}{1 + \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)}}$$

(3.5)

If we define the ratio of the log posterior probabilities as

$$a = \ln \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} \tag{3.6}$$

If we substitute back into (4) we have:

$$p(C_1|\mathbf{x}) = f(a) = \frac{1}{1 + e^{-a}} \tag{3.7}$$

Here $a = \mathbf{w}^\top \mathbf{x} = w_0$. Thus the activation for the non-linear function is driven by the probability of the data to give the output class. The probabilistic function here is called a sigmoid function due to the s-shaped graph that is plotted by the function.

Rather than using the sigmoid function for multi-class classification a similar soft max function is derived by using the log probability of classes. If $a_k = \ln(p(\mathbf{x}|C_k)p(C_k))$ then:

$$y_k = p(C_k|\mathbf{x}) = \frac{e^{a_k}}{\Sigma_{\ell=1}^{K} e^{a_\ell}} \tag{3.8}$$

$$a_k = \sum_{i=0}^{d} w_{ki} x_i \tag{3.9}$$

Recall that in the generative classification method the problem is divided into sub problems by using the conditional probability, while in the discriminative approach the joint probability is determined by looking at the data directly. This is what $p(C_k|\mathbf{x})$ represents. However also, recall that we still need to determine the correct probability distribution represented by the data. This is achieved by determining the values of the weights of the linear operation. In the next section a method known as back propagation is discussed. Back propagation is the training algorithm used to determine the weight vector of all the layers in the neural network. Back propagation is an extension of the Gradient descent algorithm.

3.1.3   Back propagation algorithm (backprop)

In the previous section, the neural network architecture has been described as having N inputs M neurons and L layers. Each layer comprises M neurons of a maximum of N inputs times M neurons interconnections which embodies the inner product

of the inputs and unknown set of weights. The output of this inner product is then passed to a logistic squashing function that results in the output probabilities. The discriminative process is used here to determine the correct combination of weight vectors that accurately describe the training data. For neural networks, the weight vectors at each layer are determined through propagating the errors back through each preceding layer and adjusting the weights according to the errors propagated each time a batch of the data is processed. This process of continuously adjusting weights from back propagation continues until all the data is processed and a steady state has been reached. The steady state refers to the fact that the error has reached a steady and/or acceptable negligible value. This is often referred to in Machine Learning as convergence (Boden, 2002).

3.1.4 Gradient Descent

The last section ended stating that the back-propagation algorithm is an extension of the gradient descent algorithm. It has also been seen that back propagation works by propagating the error and making adjustments on the weights. In this section, the Gradient Descent algorithm is reviewed and how it is used in back propagation is examined.

The concept behind the Gradient descent algorithm is the fact that a function is optimized when the gradient of the function is equal to 0. Gradient descent algorithm is significant in Machine Learning applications because a cost function is easily defined for a particular Machine Learning application that is able to determine the error between the predicted value and the actual value. Then, the parameters of the problem can be adjusted until the derivative of the cost function using gradient descent is zero. Thus the Machine Learning algorithm adjusts its parameters until the error is minimised or removed.

A common error function or cost function for neural networks is the sum-of-squares error cost function. This is obtained by summing the difference between the actual value and the Machine Learning model value over the training set N.

$$E^n = \frac{1}{2} \sum_{k=1}^{K} (y_k^n - t_k^n)^2 \qquad (3.10)$$

In a neural network having a weight matrix $\mathbf{W}$ of M neurons times N inputs, the resulting gradient is a vector of partial derivatives of E with respect to each element.

$$\nabla_{\mathbf{W}} E = \left(\frac{\partial E}{\partial w_{10}}, \ldots, \frac{\partial E}{\partial w_{ki}}, \ldots, \frac{\partial E}{\partial w_{Kd}} \right) \tag{3.11}$$

The adjustment on each weight therefore on each iteration is:

$$w_{kj}^{\tau+1} = w_{kj}^{\tau} - \eta \frac{\partial E}{\partial w_{kj}} \tag{3.12}$$

Where τ is the iteration and η is a constant learning rate which is a factor to speed up or slow down the rate of learning of the Machine Learning algorithm which in this case is the neural network.

3.2 RNN, LSTM and GRU Networks

Neural networks have become increasingly popular due to their ability to model non-linear system dynamics. Since their inception, there have been many modifications made to the original design of having linear affine transformations terminated with a nonlinear functions as the means to capture both linear and non-linear features of the target system. In particular, one of such neural network modifications, namely the recurrent neural network, has been shown to overcome the limitation of varying lengths in the inputs and outputs of the classic feed-forward neural network. In addition the RNN is not only able to learn non-linear features of a system but has also been shown to be effective at capturing the patterns in sequential data. This section develops recurrent neural networks (RNNs) from a specialised multi-layer Perceptron (MLP) or the deep neural network (DNN).

3.2.1 Deep Neural Networks (DNNs)

Deep neural networks have been accepted to be networks having multiple layers and capable of hierarchical knowledge representation (Yu and Deng, 2016). This will therefore include multi-layer Perceptrons (MLPs) having more than one hidden layer (Dahl et al., 2012) as well as deep belief networks (DBNs)(Mohamed et al., 2009, Yu et al., 2010) having a similar structure. Therefore, following the MLP architecture,

A DNN uses multiple hidden layers and generates distribution function, $p(c|x_t)$ on the output layer when an input vector $\mathbf{x}_t$ is applied. At the first hidden layer, activations are vectors evaluated using

$$\mathbf{h}^{(1)} = \sigma(\mathbf{W}^{(1)T}\mathbf{x}_t + \mathbf{b}^{(1)}) \tag{3.13}$$

The matrix $\mathbf{W}^{(1)}$ is the weight matrix and vector $b^{(1)}$, the bias vector for the layer. The function $\sigma(\cdot)$ is the point-wise non-linear function. DNNs activations $h^{(i)}$ at layer i, at arbitrarily many hidden layers after the first hidden layer, are subsequently hidden activations are determined from

$$\mathbf{h}^{(1)} = \sigma(\mathbf{W}^{(1)T}\mathbf{h}^{(i-1)} + \mathbf{b}^{(1)}) \tag{3.14}$$

The distribution over all the possible set of characters c is obtained in the final layer of the network in the exact way of a multi-layer Perceptron, that is, using soft max activation at the output layer of the form,

$$p(c = c_k|x_t) = \frac{exp(-(\mathbf{W}_k^{(s)T}h^{(i-1)} + b_k^{(1)}))}{\sum_j exp(-(\mathbf{W}_k^{(s)T}h^{(i-1)} + b_k^{(1)}))} \tag{3.15}$$

$W_k^{(s)}$ and $b_k^{(k)}$ respectively are the output weight matrix and the scalar bias term of the k-th neuron. Accordingly, sub gradients for all parameters in the DNN are utilised to back propagate errors in weights during training for gradient-based optimisation techniques. In DNN-HMM speech models, DNNs are trained to predict probability distributions over senones. However, in the model neural network described in section 3.3.1, of this thesis, predicts per character conditional distributions. Combining equations (3.12, 3.13, 3.14 and 3.15) the following simplified algorithm ensues

Result: Optimal weights

initialise weights randomly;

while *error is significant or epochs less than maximum* **do**

    forward computation in equation (3.13 and 3.14);

    determine layer wise error for weights and biases $\Delta_{\mathbf{W}}E$ and $\Delta_{\mathbf{b}}E$;

    update weights and biases according to gradient descent. Equation (3.12);

end

<div align="center">

Algorithm 1: DNN training algorithm

</div>

3.2.2   Recurrent Neural Networks

One of the two advantages RNNs have over regular DNNs is the ability to capture varying lengths of outputs to inputs. That is for tasks such as language translation where there is no one to one correspondence of number of words in a sentence for example from the source language to the output destination language. At the same time the sentence length appearing at the input and that appearing at the output differ for different sentences. This is the first problem of varying lengths for input and output sequences.

The second issue that RNNs effectively contain as opposed to DNNs is capturing temporal relationships between the input sequences. As was realised for hidden Markov models, it was seen that the HMM modeled not just observation likelihoods but also transition state likelihoods which were latent or hidden variables. By tying the output of previous neuron activations to present neuron activations, a DNN inherits a cyclic architecture becoming a recurrent neural network (RNN). As a result, an RNN is to able capture previous hidden states and in the process derive memory-like capabilities (Yu and Deng, 2016).

In speech processing, it is observed that for a given utterance, there are various temporal dependencies which may not be sufficiently captured by DNN-based systems because DNN systems ignore previous hidden representations and output distributions at each time step t. The DNN derives its output using only the feature inputs x_t. The architecture of RNN to enable better modelling of temporal dependencies present in a speech is given in (Hannun et al., 2014b, Yu and Deng, 2016).

$$h_t^{(j)} = \sigma(\mathbf{W}^{(j)T} h_t^{(i-1)} + \mathbf{W}_k^{(j)T} h_{t-1}^{(j)} + b^{(j)})) \tag{3.16}$$

It can be seen in equation (3.16) above that given a selected RNN hidden layer j, a temporally recurrent weight matrix $W^{(f)}$ is computed for output activations $h_{t-1}^{(j)}$ for the hidden activation vector of layer j at time step $t-1$ such that the output contributes to the standard DNN output of $\mathbf{W}^{(j)T} h_t^{(i-1)}$. It can also be seen from equation (3.16) that the temporal recurrent weight matrix computation is a modified version of the standard DNN weight matrix computation and that the overall output is a superposition of the two.

Since computations for a RNN are the same as those described in standard DNN evaluations, it is possible to compute the sub gradient for RNN architecture using the back propagation algorithm. The modified algorithm appropriately called back propagation through time (BPTT) (Boden, 2002, Jaeger, 2002) is derived in section 3.2.3 below.

3.2.3 Back propagation through time (BPTT) algorithm

First we define an arbitrary but carefully chosen number of time steps $t = 1, 2, \ldots, T$ such that at each time step the states of the neuron activations $j = 1, 2, \ldots, J$ are captured. Using the sum-squared error as the cost function

$$E = c \sum_{t=1}^{T} ||\mathbf{l}_t - \mathbf{y}_t||^2 = c \sum_{t=1}^{T} \sum_{j=1}^{L} (l_t(j) - y_t(j))^2 \tag{3.17}$$

Where c is a gradient descent convenience factor, equation (3.17). $||\mathbf{l}_t - \mathbf{y}_t||$ is the modulus of the difference between the actual output $\mathbf{y}_t$ and the label vector $\mathbf{y}_t$ at time t. The two-step BPTT algorithm described in Yu and Deng (2016) is involves the recursive computation of the cost function and updating of the network weights.

For each of these steps recall from equation (3.16) the activation of a hidden layer is a result of the composition of the regular DNN activation and an activation generated from weights from the previous time step.

The error term at final time t=T is

$$\delta_T^y(j) = -\frac{\delta E}{\delta y_T(j)}\frac{\delta y_T(j)}{\delta v_T(j)} = (l_T(j) - y_T(j))g'(v_T(j)) \text{ for } j = 1, 2, \ldots, L \qquad (3.18)$$

or

$$\delta_T^y = (\mathbf{l}_T - \mathbf{y}_T) \bullet g'(\mathbf{v}_T) \qquad (3.19)$$

The error at the hidden layer is given as

$$\delta_T^h(j) = -\left(\sum_{i=1}^{L} \frac{\partial E}{\partial v_T(i)}\frac{\partial v_T(i)}{\partial h_T(j)}\frac{\partial h_T(j)}{\partial u_t(j)}\right) = \sum_{i=1}^{L} \delta_T^y(i)w_{hy}(i,j)f'(u_T(j)) \text{ for } j = 1, 2, ..., N$$

$$(3.20)$$

or $\delta_T^h = \mathbf{W}_{hy}^T \delta_T^y \bullet f'(\mathbf{u}_T)$ where $\bullet$ is element-wise multiplication.

The recursive component for other time frames, $t = T - 1, T - 2, \ldots, 1$, the error term is determined as

$$\delta_t^y(j) = (l_t(j) - y_t(j))g'(v_t(j)) \text{ for } j = 1, 2, \ldots, L \qquad (3.21)$$

or

$$\delta_t^y = (\mathbf{l}_t - \mathbf{y}_t) \bullet g'(\mathbf{v}_t) \qquad (3.22)$$

Therefore the output units are

$$\delta_t^h(j) = -\left[\sum_{i=1}^{N} \frac{\partial E}{\partial \mathbf{u}_{t+1}(i)}\frac{\partial \mathbf{u}_{t+1}(i)}{\partial h_t(j)} + \sum_{i=1}^{L} \frac{\partial E}{\partial v_t(i)}\frac{\partial v_t(i)}{\partial h_t(j)}\right]\frac{\partial h_t(j)}{\partial u_t(j)}$$

$$= \left[\sum_{i=1}^{N} \delta_{t+1}^h(i)w_{hh}(i,j) + \sum_{i=1}^{L} \delta_t^y(i)w_{hy}(i,j)\right]f'(u_t(j)) \text{ for } j = 1, \ldots, N$$

or $\delta_t^h = \left[\mathbf{W}_{hh}^{\top}\delta_{t+1}^h + \mathbf{W}_{hy}^{\top}\delta_t^y\right] \bullet f'(\mathbf{u}_t)$

$$(3.23)$$

Note that the error terms are propagated back from hidden layer at time frame $t + 1$ to the output at time frame t.

Update of RNN Weights

The weights are updated using the error terms determined in the previous section.
For the output weight matrices, we have

$$
\begin{aligned}
w_{hy}^{new}(i,j) &= w_{hy}(i,j) - \gamma \sum_{t=1}^{T} \frac{\partial E}{\partial v_t(i)} \frac{\partial v_t(i)}{\partial w_{hy}(i,j)} = w_{hy}(i,j) - \gamma \sum_{i=1}^{T} \delta_t^y(i) h_t(j) \\
&\text{or } \mathbf{W}_{hy}^{new} = \mathbf{W}_{hy} + \gamma \sum_{t=1}^{T} \delta_y^t \mathbf{h}_t^\top
\end{aligned}
\tag{3.24}
$$

For the input weight matrices, we get

$$
w_{xh}^{new}(i,j) = w_{xh}(i,j) - \gamma \sum_{t=1}^{T} \frac{\partial E}{\partial u_t(i)} \frac{\partial u_t(i)}{\partial w_{xh}(i,j)} = w_{xh}(i,j) - \gamma \sum_{t=1}^{T} \delta_t^h(i) x_t(j)
\tag{3.25}
$$

or

$$
\mathbf{W}_{xh}^{new} = \mathbf{W}_{xh} + \gamma \sum_{t=1}^{T} \delta_h^t \mathbf{x}_t^\top
\tag{3.26}
$$

For the recurrent weight matrices we have

$$
\begin{aligned}
w_{hh}^{new}(i,j) &= w_{hh}(i,j) - \gamma \sum_{t=1}^{T} \frac{\partial E}{\partial u_t(i)} \frac{\partial u_t(i)}{\partial w_{hh}(i,j)} \\
&= w_{hh}(i,j) - \gamma \sum_{t=1}^{T} \delta_h^t(i) h_{t-1}(j) \\
&\text{or } = \mathbf{W}_{hh}^{new} = \mathbf{W}_{hh} + \gamma \sum_{t=1}^{T} \delta_h^t \mathbf{h}_{t-1}^\top
\end{aligned}
\tag{3.27}
$$

In the BPTT algorithm the sub gradients are summed over all time frames. The
algorithm is summarised below:

Result: Optimal weights

initialise weights randomly;

for *error is significant or epochs less than maximum* **do**

> forward computation ;
>
> determine layer-wise error for weights and biases $\Delta_\mathbf{W} E$ and $\Delta_\mathbf{b} E$;
>
> update weights and biases according to gradient descent;

end

Algorithm 2: RNN training algorithm

3.2.4 LSTMs and GRUs

A special implementation of the RNN called the Long Short Term Memory (LSTM) has been designed to capture patterns over particularly long sequences of data and thus is an ideal candidate for generating character sequences while preserving syntactic language rules learned from the training data.

The internal structure and working of the LSTM cell is documented by its creators in Sak et al. (2014). The ability to recall information over extended sequences results from the internal gated structure which performs a series of element wise multiplications on the inputs and internal state of the LSTM cell at each time step. In addition to the output neurons which in this text we refer to as the write gate and denote as the current cell state, $\mathbf{c}_t$, three additional gates (comprising a neural network sub-layer) located within the LSTM cell are the input gate, the forget gate and the output gate. Together with the initial current state cell, these gates along with the current-state cell itself enable the LSTM cell architecture to store information, forward information, delete information and receive information. Generally however, the LSTM cell looks like a regular feed-forward network having a set of neurons capped with a nonlinear function. The recurrent nature of the network arises, however due to the fact that the internal state of the RNN cell is rerouted back as an input to the RNN cell or input to the next cell in the time-series giving rise to sequence memory within the LSTM architecture. Mathematically, these gates are formulated as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}^{(xi)}\mathbf{x}_t + \mathbf{W}^{(hi)}\mathbf{h}_{t-1} + \mathbf{W}^{(ci)}\mathbf{c}_{t-1} + \mathbf{b}^{(i)}) \tag{3.28}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^{(xf)}\mathbf{x}_t + \mathbf{W}^{(hf)}\mathbf{h}_{t-1} + \mathbf{W}^{(cf)}\mathbf{c}_{t-1} + \mathbf{b}^{(f)}) \tag{3.29}$$

$$\mathbf{c}_t = \mathbf{f}_t \bullet \mathbf{c}_{t-1} + \mathbf{i}_t \bullet \tanh(\mathbf{W}^{(xc)}\mathbf{x}_t + \mathbf{W}^{(hc)}\mathbf{h}_{t-1} + \mathbf{b}^{(c)}) \tag{3.30}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^{(xo)}\mathbf{x}_t + \mathbf{W}^{(ho)}\mathbf{h}_{t-1} + \mathbf{W}^{(co)}\mathbf{c}_{t-1} + \mathbf{b}^{(o)}) \tag{3.31}$$

$$\mathbf{h}_t = \mathbf{o}_t \bullet \tanh(\mathbf{c}_t) \tag{3.32}$$

The gates in the above formula are illustrated in Figure 3.3. $\mathbf{i}_t$ represents the input gate, $\mathbf{f}_t$ is the forget gate and $\mathbf{o}_t$ represents the output gate. At each of these gates therefore, the inputs consisting of hidden states in addition to the regular
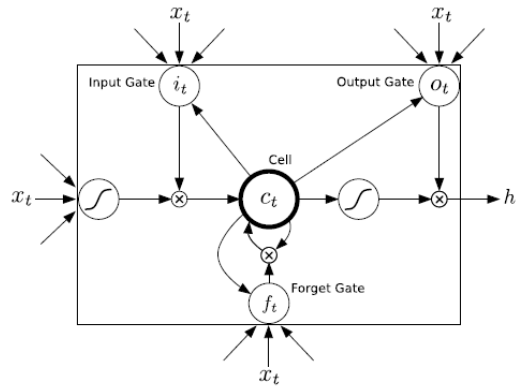
Figure 3.3: An LSTM Cell (Graves et al., 2013)

inputs are multiplied by a set of weights and passed through a soft-max function.
These weights during training learn whether the gate will, during inference, open
or not. In summary, the input gate tells the LSTM whether or not to receive new
information, the forget gate determines whether the current information it already
has from the previous step should be kept or dropped and the output gate determines
what should be forwarded to the next LSTM cell. Note also that the LSTM has two
sigmoid ($tanh$) activation functions utilised at the input and output of the current
cell $\mathbf{c}_t$.

One particular variant of the original LSTM model is the GRU cell. Though
simpler than an LSTM cell the GRU cell performs equally efficiently. The GRU
cell is a subset implementation of the LSTM cell. Rather than using the output
gate of the LSTM, this gate is omitted in the GRU and the output result of the
other internal gates are always forwarded. The second simplification is a merge
of the internal gate state vectors into a single vector $\mathbf{h}_{(t)}$. This merged gate here
referred to as $\mathbf{z}(t)$, controls both the forget gate an the input gate and acts as follows.
Whenever a value is retained by the cell the previous value is erased first. That is,
if the gate controller outputs a 1, in the LSTM this corresponds to the input gate
is open and the forget gate is closed. Therefore if $\mathbf{z}(t)$ it outputs a 0, the reverse
happens for the input gate and the forget gate in the LSTM. There is, however, a
new gate controller, $\mathbf{r}(t)$, which determines which portion of the previous state will
be shown at the output (Cho et al., 2014).

The architecture of a GRU is formulated as follows:

$$\mathbf{z}_{(t)} = \sigma(\mathbf{W}_{xz}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hz}^T \cdot \mathbf{x}_{(t-1)}) \tag{3.33}$$

$$\mathbf{r}_{(t)} = \sigma(\mathbf{W}_{xr}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hr}^T \cdot \mathbf{x}_{(t-1)}) \tag{3.34}$$

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)})) \tag{3.35}$$

$$\mathbf{h}_{(t)} = (1 - \mathbf{z}_{(t)}) \otimes (\mathbf{h}_{(t-1)}) + \mathbf{z}_{(t)} \otimes \mathbf{g}_t \tag{3.36}$$

Due to the light-weight nature of the GRU cell, it is common practice to use GRU cells in place of LSTM cells. This precedence achieves the much desired lighter computation load on the actual hardware performing the RNN training. As each of the gates required in an LSTM cell comprises high density matrix multiplication operations in themselves, the condensation of two gates into one and the omission of the output gate within GRU cells pushes towards halving the architectural complexity and coupled with the equally efficient performance of the GRU when compared to the LSTM cell ultimately serves as an overall improvement on the LSTM architecture. For these reasons, GRUs have highly appealing features when compared to LSTMs and was the RNN cell of choice used for the study in this report.

3.3 Deep speech architecture

This work makes use of an enhanced RNN architecture called the Bi-directional Recurrent Neural Network (BiRNN). While Hannun et al. (2014b) assert that forward recurrent connections does reflect the sequential relationships of an audio waveform, perhaps the BiRNN model achieves a more robust sequence model.

The BiRNN is a preferred end to end mechanism due to the length of sequence over which temporal relationships can be captured. This implies that BiRNNs will be suited for capturing temporal relationships over much longer sequences than a forward only RNN, because hidden state information is preserved in both forwards and backwards direction.

In addition, such a model has a notion of complete sentence or utterance integration, having information over the entire temporal extent of the input features

when making each prediction.

The formulation of the BiRNN is derived by starting off with the basic RNN architecture which is referred to as the forward architecture. From the forward architecture we derive the backward architecture. If we choose a temporally recurrent layer j, the BiRNN forward and backward intermediate hidden representation $h_t^{(f)}$ and $h_t^{(b)}$ is given as.

$$h_t^{(f)} = \sigma(\mathbf{W}^{(j)T}h_t^{(i-1)} + \mathbf{W}_k^{(f)T}h_{t-1}^{(j)} + b^{(j)})) \qquad (3.37)$$

$$h_t^{(b)} = \sigma(\mathbf{W}^{(j)T}h_t^{(i-1)} + \mathbf{W}_k^{(b)T}h_{t+1}^{(b)} + b^{(j)})) \qquad (3.38)$$

Temporal weight matrices $W^{(f)}$ and $W^{(b)}$ propagate $h_t^{(f)}$ and $h_t^{(b)}$ forward and backward in time respectively.

Hannun et al. (2014b) points out that the recurrent forward and backward components are evaluated entirely independent of each other and for optimal training, a modified non linearity function $\sigma(z) = min(max(z,0),20)$ is recommended.

The final BiRNN representation $h_t^{(j)}$ for the layer is now the superposition of the two RNN components,

$$h_t^{(j)} = h_t^{(f)} + h_t^{(b)} \qquad (3.39)$$

Also note that back propagation through time (BPTT) sub gradient evaluations are computed from the combined BiRNN structure directly during training.

3.3.1 Connectionist Temporal Classification (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 (Graves et al., 2006). In the following paragraph we introduce terminology associated with the CTC loss function.

Given two symbols A and $\mathcal{B}$ such that A has a many to one relationship with $\mathcal{B}$, signifying the temporal nature of the classification. The symbol A 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 $|A|$ over all possible labels in A from activations within $|A|$. An extra activation is given to represent the probability of outputting a *blank*, 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 $A' = A \cup \{blank\}$, also, $y_{t,p}$ as the activation of network output p at time t. Therefore $y_{t,p}$ is the probability that the network will output element $p \in A'$ at time t given that x is the input sequence of length T. The distribution sought after $Pr(\pi|x)$, is the conditionally-independent distribution over the subset A'^T where A'^T denotes the set of length T sequences in A'.

$$\Pr(\pi \,|\, x) = \prod_{t=1}^{T} y_{t,\pi_t} \tag{3.40}$$

From the above, it is now possible to define the many-to-one mapping $\mathcal{B} : A'^T \to A^{\leq T}$, from the set of paths onto the set $A^{\leq T}$ of possible labellings of x (i.e. the set of sequences of length less than or equal to T over A). We do this by removing first the repeated labels and then the blanks from the paths. For example,

$$\mathcal{B}(a - ab-) = aab$$
$$\mathcal{B}(-aa - -abb) = aab. \tag{3.41}$$

Intuitively, this corresponds to outputting a new label when the network either switches from predicting no label to predicting a label, or from predicting one label to another. As $\mathcal{B}$ is many-to-one, the probability of some labelling $l \in A^{\leq T}$ can be calculated by summing the probabilities of all the paths mapped onto it by $\mathcal{B}$:

$$\Pr(l \,|\, x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} \Pr(\pi \,|\, x) \tag{3.42}$$

This 'collapsing together' of different paths onto the same labelling is what makes it possible for CTC to use unsegmented data, because it allows the network to predict the labels without knowing in advance where they occur. In theory, it also makes CTC networks unsuitable for tasks where the location of the labels must be determined. However in practice CTC tends to output labels close to where they

occur in the input sequence.

3.3.2 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 (α), multiplied by the probability of all the points from that state to the end of the sequence, the backward variable (β).

The difference between this estimation and that determined from equation (3.42) is the fact that the forward-backward algorithm converts equation (3.42) 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" l', that caters for blank characters in between regular ones l, as defined in A. Thus, if U is defined as the length of l. Then U' is of length $2U + 1$. 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, $\alpha(t, u)$ now becomes the summed probability of all length t paths that are mapped by $\mathcal{B}$ onto the length $\lfloor u/2 \rfloor$ prefix of l. (Note, $\lfloor u/2 \rfloor$ is the floor of $u/2$, the greatest integer less than or equal to $u/2$.) For some sequence s, let $s_{p:q}$ denote the sub-sequence $s_p, s_{p+1}, \ldots, s_{q-1}, s_q$, and define the set $V(t, u) \equiv \{\pi \in A'^t : \mathcal{B}(\pi) = l_{1:\lfloor u/2 \rfloor} \text{ and } \pi_t = l'_u\}$. $\alpha(t, u)$ then becomes

$$\alpha(t, u) \equiv \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} y_{i,\pi_i} \tag{3.43}$$

The forward variables at time t can be calculated recursively from those at time $t - 1$ and expressed as the sum of the forward variables with and without the final blank at time T.

$$\Pr(l \mid x) = \alpha(T, U') + \alpha(T, U' - 1) \tag{3.44}$$

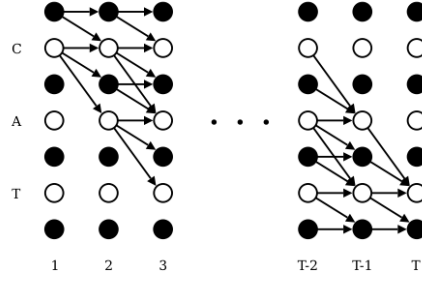All correct paths must start with either a blank (b) or the first symbol in l (l_1),

Figure 3.4: Beam Search Lattice Structure (Graves et al., 2006)

yielding the following initial conditions:

$$\alpha(1,1) = y_{1,b}$$
$$\alpha(1,2) = y_{1,l_1} \quad (3.45)$$
$$\alpha(1,u) = 0, \ \forall u > 2$$

Thereafter the variables can be calculated recursively:

$$\alpha(t,u) = y_{t,l'_u} \sum_{i=f(u)}^{u} \alpha(t-1,i) \quad (3.46)$$

where

$$f(u) = \begin{cases} u-1, & \text{if } l'_u = blank \text{ or } l'_{u-2} = l'_u \\ u-2, & \text{otherwise} \end{cases} \quad (3.47)$$

Graphically we can express the recurrence relation for $\alpha(t,u)$ as follows.

where t runs along the x axis and u runs along the y axis. The black circles of the diagram represent *blank* elements of l' while the white circles represent non-*blank* elements of l'. The arrows represent computational dependencies derived from our recursion relation for $\alpha(t,u)$. So, for example, the value of $\alpha(2,3)$, corresponding to the *blank* at $t = 2$ and $u = 3$, is derived from $\alpha(1,2)$. Similarly, the value of $\alpha(2,2)$, corresponding to the letter c at $t = 2$ and $u = 2$, is derived from $\alpha(1,2)$ and $\alpha(1,1)$.

$$\alpha(t,u) = 0 \quad \forall u < U' - 2(T-t) - 1 \quad (3.48)$$

because these variables correspond to states for which there are not enough time-

steps left to complete the sequence. We also impose the boundary condition

$$\alpha(t, 0) = 0 \quad \forall t \tag{3.49}$$

The backward variables $\beta(t, u)$ are defined as the summed probabilities of all paths starting at $t + 1$ that "complete" l when appended to any path $\hat{\pi}$ contributing to $\alpha(t, u)$. Define $W(t, u) \equiv \{\pi \in A'^{T-t} : \mathcal{B}(\hat{\pi} + \pi) = l \ \forall \hat{\pi} \in V(t, u)\}$. Then

$$\beta(t, u) \equiv \sum_{\pi \in W(t,u)} \prod_{i=1}^{T-t} y_{t+i, \pi_i} \tag{3.50}$$

The rules for initialisation of the backward variables are as follows

$$\beta(T, U') = 1$$
$$\beta(T, U' - 1) = 1 \tag{3.51}$$
$$\beta(T, u) = 0, \ \forall u < U' - 1$$

The rules for recursion are as follows:

$$\beta(t, u) = \sum_{i=u}^{g(u)} \beta(t + 1, i) y_{t+1, l'_i} \tag{3.52}$$

where

$$g(u) = \begin{cases} u + 1, & \text{if } l'_u = blank \text{ or } l'_{u+2} = l'_u \\ u + 2, & \text{otherwise} \end{cases} \tag{3.53}$$

3.3.3   CTC 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 $\mathcal{L}(S)$ uses the cross entropy loss function of and is defined as the cross entropy error of correctly labelling all the training samples in some training set S:

$$\mathcal{L}(S) = -\ln \prod_{(x,z) \in S} \Pr(z \,|\, x) = -\sum_{(x,z) \in S} \ln \Pr(z \,|\, x) \tag{3.54}$$

where z is the output label and x is the input sequence. Since $\mathcal{L}(S)$ in equation 3.54 is differentiable, this loss function can be back propagated to the softmax layer in the BiRNN configuration discussed in section 3.3.

$$\mathcal{L}(x, z) \equiv -\ln \Pr(z \mid x) \tag{3.55}$$

and therefore

$$\mathcal{L}(S) = \sum_{(x,z) \in S} \mathcal{L}(x, z) \tag{3.56}$$

From the definition of the forward and backward variables ($\alpha(t, u)$ and $\beta(t, u)$), we also establish that $X(t, u) \equiv \{\pi \in A'^T : \mathcal{B}(\pi) = z, \ \pi_t = z'_u\}$, such that

$$\alpha(t, u)\beta(t, u) = \sum_{\pi \in X(t,u)} \prod_{t=1}^{T} y_{t, \pi_t} \tag{3.57}$$

then substituting $\Pr(\pi \mid x)$ from the expression in equation 3.40, we have

$$\alpha(t, u)\beta(t, u) = \sum_{\pi \in X(t,u)} \Pr(\pi \mid x) \tag{3.58}$$

Also observe that $\Pr(l \mid x)$ is equivalent to the total probability $\Pr(z \mid x)$. Paths going through z'_u at time t can be obtained as summed over all u to get

$$\Pr(z \mid x) = \sum_{u=1}^{|z'|} \alpha(t, u)\beta(t, u) \tag{3.59}$$

Thus a sample loss is determined by

$$\mathcal{L}(x, z) = -\ln \sum_{u=1}^{|z'|} \alpha(t, u)\beta(t, u) \tag{3.60}$$

and therefore the overall loss is given by

$$\mathcal{L}(S) = -\sum_{(x,z) \in S} \ln \sum_{u=1}^{|z'|} \alpha(t, u)\beta(t, u) \tag{3.61}$$

In the model described in this work, the gradient $\mathcal{L}(x, z)$ 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 calcula-
tions avoids this situation and another useful equation in this context is

$$\ln(a + b) = \ln(a) + \ln(1 + e^{\ln b - \ln a}) \tag{3.62}$$

Chapter 4

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 Mallat (2016). 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 (Andén and Mallat, 2011, Peddinti et al., 2014, Sainath et al., 2014, Zeghidour et al., 2016) 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 Sainath et al. (2014) shows 4-7% relative improvement in word error rates (WER) over Mel frequencies cepstral 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.

Figure 4.1: Fourier Equation

4.1 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 4.1).

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 complementary 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

$$a_k = \frac{1}{T} \int_{-T/2}^{T/2} x(t)e^{\left(-j\frac{2\pi kt}{T}\right)} \tag{4.1}$$

Then, the corresponding synthesis equation is given by

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{\left(j\frac{2\pi kt}{T}\right)} \tag{4.2}$$

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 4.2), 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 4.2, the energies are concentrated at frequencies of about 100, 150 and 400 hertz.

The Fourier transform discussed in the preceding paragraph 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
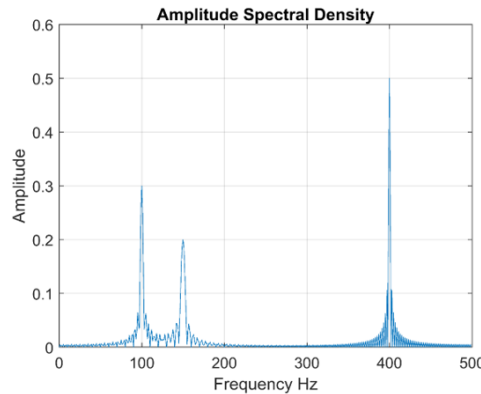
Figure 4.2: Sample Spectrogram
cwt (2015)

related signal deformations. The Short-time Fourier Transform (STFT) attempts to
salvage this by windowing the signal in time signal and performing Fourier trans-
forms 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.

4.2 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 4.3

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
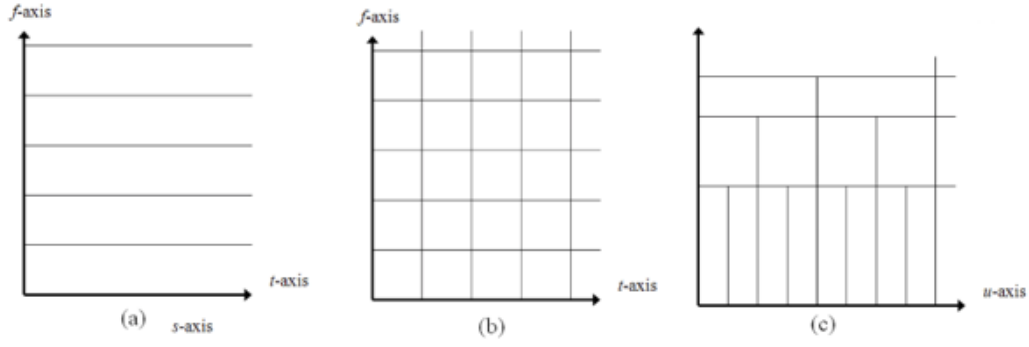
Figure 4.3: Time frequency tiling for (a) Fourier Transform (b) Short-time Fourier
Transform (STFT) (c) Wavelet transform

fundamental representation of the continuous wavelet function is:

$$C(a, b) = \int f(t) \frac{1}{\sqrt{a}} \psi \left(\frac{t - b}{a} \right) dt \tag{4.3}$$

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.

4.3   Discrete and Fast wavelet transform

Synthesis and analysis equations (4.2 and 4.1) can be formulated as a linear com-
bination of the basis $\phi_k(t)$ such that the basis, $\phi_k(t) = e^{j2\pi kt}$, and it's conjugate or
orthonormal basis, $\tilde{\phi}_k(t) = e^{-j2\pi kt}$, equations (4.2 and 4.1) now become

$$x(t) = \sum_k a_k \phi_k \tag{4.4}$$

$$a_k = \int x(t) \tilde{\phi}_k(t) \tag{4.5}$$

With respect to scaling and shifting variables of continuous wavelet transforms
in equation (4.3), 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:

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k) \tag{4.6}$$

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k) \tag{4.7}$$

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.

$$\phi(t) = h_{\phi}[n]\sqrt{2}\phi(2t - n) \tag{4.8}$$

$$\psi(t) = h_{\phi}[n]\sqrt{2}\psi(2t - n) \tag{4.9}$$

For any signal, a discrete wavelet transform in $l^2(Z)^1$ can be approximated by

$$f[n] = \frac{1}{\sqrt{M}}\sum_{k} W_{\phi}[j_0, k]\phi_{j_0,k}[n] + \frac{1}{\sqrt{M}}\sum_{j=j_0}^{\infty}\sum_{k} W_{\psi}[j, k]\psi_{j,k}[n] \tag{4.10}$$

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.

$$W_{\phi}[j_0, k] = \frac{1}{\sqrt{M}}\sum_{n} f[n]\phi_{j_0,k}[n] \tag{4.11}$$

$$W_{\psi}[j, k] = \frac{1}{\sqrt{M}}\sum_{n} f[n]\psi_{j,k}[n] \quad j \geq j_0 \tag{4.12}$$

Equation (4.11) is called approximation coefficient while (4.12) 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 down-sampled

version of the next order coefficients. The fast wavelet transform was first postulated in (Mallat, 1989).

$$W_\phi[j, k] = h_\phi[-n] * W_\phi[j + 1, n]|_{n=2k, k \geq 0} \tag{4.13}$$

$$W_\psi[j_0, k] = h_\psi[-n] * W_\phi[j + 1, n]|_{n=2k, k \geq 0} \tag{4.14}$$

For analysis of the Haar wavelet and the derivation of equations (4.13 and 4.14) see appendix **??**.

4.4 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 loss-less encoding of speech signals relevant for speech recognition.

The Mel filter banks form the basis of the Mel Frequency Cepstral Coefficients (MFCCs) described by (Davis and Mermelstein, 1980). 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), (Dines et al., 2010, McLoughlin, 2009). The following paragraphs describe how the Mel filters are derived.

The Mel scale as described by Stevens et al. (1937) 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 non-linear 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. (Becchetti, 1999).

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:

$$M(f) = 1125 \ln(1 + f/700) \tag{4.15}$$

$$M^1(m) = 700 \exp(m/1125)1 \tag{4.16}$$

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 (Becchetti, 1999)(Becchetti, 1999):

$$\Delta_m = 1.2 \times \Delta_{m-1} \tag{4.17}$$

where m is the frequency bin index and Δ_m is the frequency range between the start and end frequencies for the m-th bin. The resulting filters are overlapping filters shown in Figure 4.4. 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:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} c_{tn})}{2 \sum_{n=1}^{N} n^2} \tag{4.18}$$
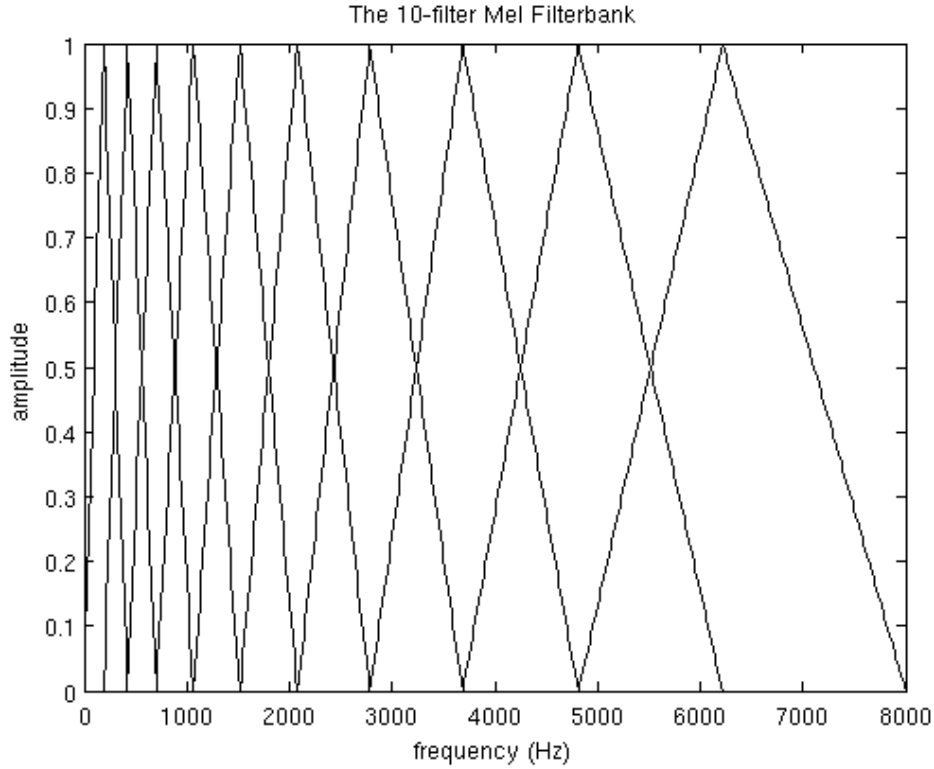
Figure 4.4: Mel filter plot (Lyons, 2012)

where c_x 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, decorrelated signals optimise algorithm computation efficiency involving matrix operations such that rather than using full co variance 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(Gales et al., 2008).

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 4.18. 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 (Gales
et al., 2008).

4.5   Deep scattering spectrum

In this section reference is made to (Andén and Mallat, 2011, 2014, Zeghidour et al.,
2016). For a signal x we define the following transform W_x as a convolution with a
low-pass filter ϕ and higher frequency complex analytic wavelets ψ_{λ_1}:

$$Wx = (x \star \phi(t), x \star \psi_{\lambda_1}(t))_{t \in R, \lambda_1 \in \Lambda_1} \tag{4.19}$$

We apply a modulus operator to the wavelet coefficients to remove complex phase
and extract envelopes at different resolutions

$$|W|x = (x \star \phi(t), |x \star \psi_{\lambda_1}(t)|)_{t \in R, \lambda_1 \in \Lambda_1} \tag{4.20}$$

$S_0 x = x \star \phi(t)$ is locally invariant to translation thanks to the time averaging ϕ. This
time-averaging loses the high frequency information, which is retrieved in the wavelet
modulus coefficients $|x \star \psi_{\lambda_1}|$. However, these wavelet modulus coefficients are not
invariant to translation, and as for S_0, a local translation invariance is obtained by
a time averaging which defines the first layer of scattering coefficients

$$S_1 x(t, \psi_{\lambda_1}) = |x \star \psi_{\lambda_1}| \star \phi(t)) \tag{4.21}$$

It is shown in Andén and Mallat (2014) that if the wavelets ψ_{λ_1} have the same fre-
quency resolution as the standard Mel-filters, then the $S_1 x$ coefficients approximate
the Mel-filter coefficients. Unlike the Mel-filter banks however, there is a strategy
to recover the lost information, by passing the wavelet modulus coefficients $|x \star \phi_{\lambda_1}|$
through a bank of higher frequency wavelets ψ_{λ_2}:

$$|W_2||x \star \phi_{\lambda_1}| = (|x \star \psi_{\lambda_1}| \star \phi, ||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|)_{\lambda_2 \in \Lambda_2}) \tag{4.22}$$

This second layer of wavelet modulus coefficients is still not invariant to translation,
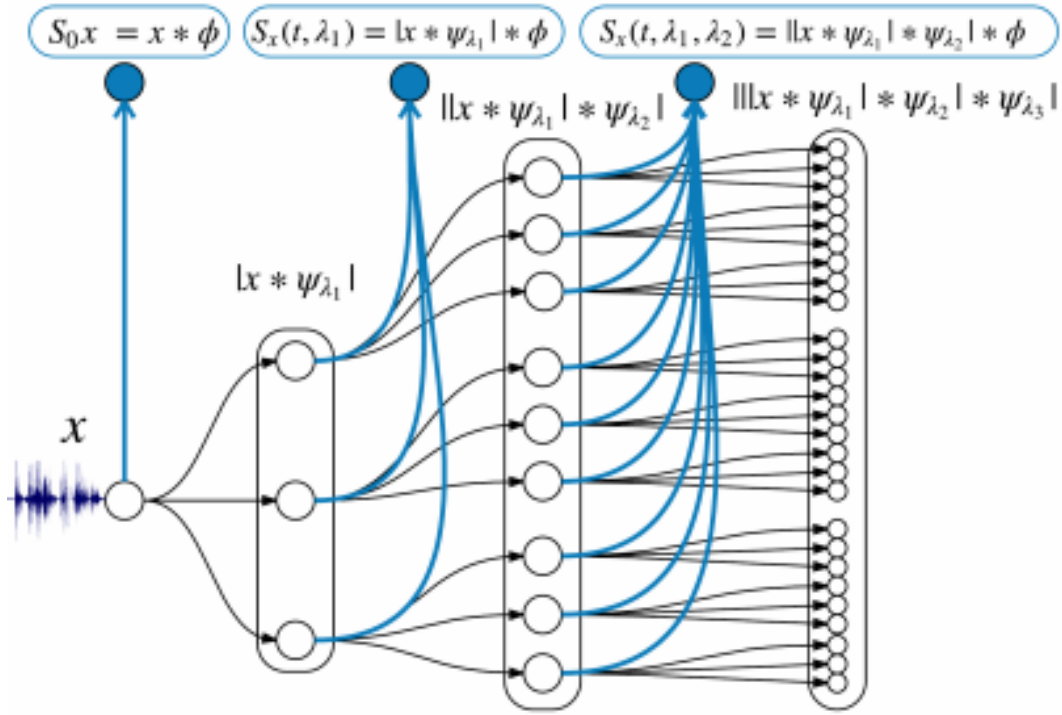
Figure 4.5: Scattering network - 2 layers deep
Zeghidour et al. (2016)

hence we average these coefficients with a low-pass filter ϕ to derive a second layer of of scattering coefficients.

$$|W_2||x \star \phi_{\lambda_1}| = (|x \star \psi_{\lambda_1}| \star \phi, ||x \star \psi_{\lambda_1}| \star \psi_{\lambda_2}|)_{\lambda_2 \in \Lambda_2}) \qquad (4.23)$$

Repeating these successive steps of computing invariant features and retrieving lost information leads to the scattering spectrum, as seen in Fig. 1, however speech signals are almost entirely characterized by the first two layers of the spectrum, that is why a two layers spectrum is typically used for speech representation. It is shown in [6] that this representation is invariant to translations and stable to deformations, while keeping more information than the Mel-filter banks coefficients

Chapter 5

Wakirike Language Model

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.

5.1   Data Preparation

The Wakirike New Testament Bible served as the source of data for the deep neural network training. As there wasn't a readily available soft or on-line copy of the Wakirike new testament bible for use, this was typed, giving rise to a complete corpus word size of 668,522 words and a character count of 6,539,176 characters. The data set was then divided into 11 parts. Two parts dedicated for testing and validation and the remaining 9 parts were used for training.

The Unicode representations of the character set consisting of letters and punctuation marks are one-hot encoded and batched for sequential input, each batch having a character sequence length of 30 characters.

5.2   GRU Training

The modified LSTM RNN known as the Gated Recurrent Unit (GRU) is employed for the neural network model in order to optimise network performance in terms of

resource conservation. GRUs have been shown to give similar performance to regular LSTMs however, with a lighter system resource footprint (Cho et al., 2014). The GRU RNN used to train the Wakirike text corpus comprised an internal network size of 512 nodes for each layer and was 3 layers deep. Externally, 30 GRUs represented the number of recurrent connections each connection representing a time step bearing contextual for the recurrent input sequence.

To mitigate for over-fitting, due to the multi-layered high-dimensional depth of this neural network, a small learning rate of 0.001 was used. To further marginalise over-fitting the popular and effective dropout (Srivastava et al., 2014) method for regularising deep neural networks kept at 20% such that only 80% activations were propagated from one layer to the next and the remaining 20% were zeroed out.

5.3   Output Language Generation

The neural network is trained for 10 epochs and achieves a prediction accuracy of 85% on held-out data. With the GRU model it is possible to seed this network with an input character and select from the top-N candidates thus causing the Neural network to generate its own sentences. In this scenario, the network is said to perform language generation by immanently constructing its own sentences. The generated language was found to be intelligibly similar to that of the training data.

A clever use of this new corpus generated by the GRU language model of this work was to determine a word-based perplexity metric for the GRU neural language model. In this work, the word-based perplexity metric was achieved from the output language generated by first estimating the word based perplexity on the training data. The same perplexity calculation was then used on the generated neural language model corpus. The corpus size of the neural language model was made to be equivalent to that of the training data, that is containing 6,539,176 characters. The perplexity calculation was based on a modified Kneser-Key 5-gram model with smoothing (Heafield et al., 2013). 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.

The evaluation of the GRU language model of the Wakirike language was per-

formed using a perplexity measurement metric. The Perplexity metric applies the language model to a test dataset and measures how probable the test dataset is. Perplexity is a relative measure given by the formula:

$$PP(W) = P(w_1, w_2 \dots w_N)^{\frac{1}{N}} \tag{5.1}$$

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \tag{5.2}$$

Where $w_1, \dots, w_N$ are the sequence of words. The language model with the lower relative perplexity score is therefore expected to yield better approximation of the data when applied to unseen data generally. The result of the training of the GRU-Cell Recurrent Neural Network on low-resourced Wakirike Language gave impressive and intelligible results and showed better results when measured with standard n-gram language models. The results showed that it is indeed possible to use an LSTM on a low resource character sequence corpus to produce an Wakirike language model.

A character based perplexity metric is possible using the negative log likelihood of the character sequence.

$$PP(X) = exp \left\{ \frac{\sum_{t=1}^{T} \log P(x_t|x_{1:t1})}{T} \right\} \tag{5.3}$$

However, our base-line language model is a 5-gram word-based language model. Therefore, comparing a word based model to a character based model requires a conversion step. In this work, the conversion step involved using the GRU language model generated a corpus which was rescored by re-estimating with a 5-gram word-based language model

Table 5.1 shows the Results of the Perplexity model of the LSTM Wakirike Language model and an equivalent 5-gram Language model with interpolation and Keysner smoothing (Heafield et al., 2013).

5.4 Summary

It can be inferred that the GRU character-model developed has an improved language model and because it is based on a character-model, which is fine-grained

Table 5.1: Perplexity Calculation results

Language Model	Perplexity	
Held-out data size (characters)	998	99
LSTM RNN	1.6398	1.7622
3-gram with Keysner Soothing and interpolation	1.8046	1.9461

when compared to a word model, it is likely to generalise data better when used in practice is and less biased than a word-based model. This can be observed from the fact that the output corpus produced a larger vocabulary size.

Chapter 6

Deep Learning Speech Model

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 Character Classification (CTC)(Graves et al., 2006) 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.

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 is detailed.

6.1 Deep Scattering Features

The fast wavelet transform is derived in Chapter 4.5 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 (Andén et al., 2014), 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 $T = 512$ samples per window. This corresponds to a window of $50 milliseconds$ for the audio signals sampled at $8000 Hz$. For the first scattering layer the Q-band parameter was $Q = 8$ and the second scattering layer took the value $Q = 1$. Finally J is pre-calculated based on the value of T. These after Scat-Net processing produce a feature-vector having 165 dimensions. These feature vectors in turn are used as inputs to the bi-direction neural network model whose architecture is described in succeeding sections.

6.2   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 x and the corresponding label y be sampled from a training set such that the sample $S = (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ...$ In our end-to-end model, each utterance, $x^{(i)}$ is a processed feature vector consisting of 165 dimensions. Recall, every window passes through a scattering transform to yield an input of vector of $p = 165$ features; consequently, $x_{t,p}^{(i)}$ denotes the p-th feature in a scatter transform at time t.

GPU training of the speech model architecture developed above was done using Mozilla deepspeech (moz, 2019a) CTC bi-directional RNN implementation along with the accompanying Mozilla Common voice dataset (moz, 2019b). 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 in-

put sequence, x, outputs a sequence of probabilities $y_t = P(c_t|x)$, where $c_t \in a, b, c, \ldots, z, space, apostrophe, blank$.

The actual architecture of our core Bi-RNN is similar to the deepspeech system described in Hannun et al. (2014a). This structure constitutes 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:

$$h_t^{(l)} = g(W^{(l)}h_t^{(l1)} + b^{(l)}) \tag{6.1}$$

$g(\cdot) = min\{max\{0, z\}, 20\}$ is the clipped rectified linear unit and $W^{(l)}, b^{(l)}$ are weight matrix and bias parameters for layer as described in sections 3.2.1 and 3.3 respectively.

It was shown in chapter 3 the recurrent layer comprise a forward and backward RNNs whose equations are repeated here for reference

$$h_t^{(f)} = g(W^{(4)}h_t^{(3)} + W_r^{(f)}h_{t1}^{(f)} + b^{(4)}) \tag{6.2}$$

$$h_t^{(b)} = g(W^{(4)}h_t^{(3)} + W_r^{(b)}h_{t+1}^{(b)} + b^{(4)}) \tag{6.3}$$

Consequently, $h^{(f)}$ is the sequential computation from $t = 1$ to $t = T^{(i)}$ for the i-th utterance and $h^{(b)}$ is the reverse computation from $t = T^{(i)}$ to $t = 1$. In addition the output from layer five is summarily given as the combined outputs from the recurrent layer:

$$h^{(5)} = g(W^{(5)}h^{(4)} + b^{(5)}) \tag{6.4}$$

where $h^{(4)} = h^{(f)} + h^{(b)}$. The output of the Bi-RNN on layer 6 is a standard soft-max layer that outputs a predicted character over probabilities for each time slice t and character k in the alphabet:

$$h_{t,k}^{(6)} = \hat{y}_{t,k} \equiv P(c_t = k \mid x) = \frac{\exp\left((W^{(6)}h_t^{(5)})_k + b_k^{(6)}\right)}{\sum_j \exp\left((W^{(6)}h_t^{(5)})_j + b_j^{(6)}\right)}) \tag{6.5}$$

$b_k^{(6)}$ takes on the -th bias and $(W^{(6)}h_t^{(5)})_k$ is the matrix product of the k-th element. The error of the outputs are then computed using the CTC loss function Graves (2014) as described in chapter 3. A summary of our model is illustrated in
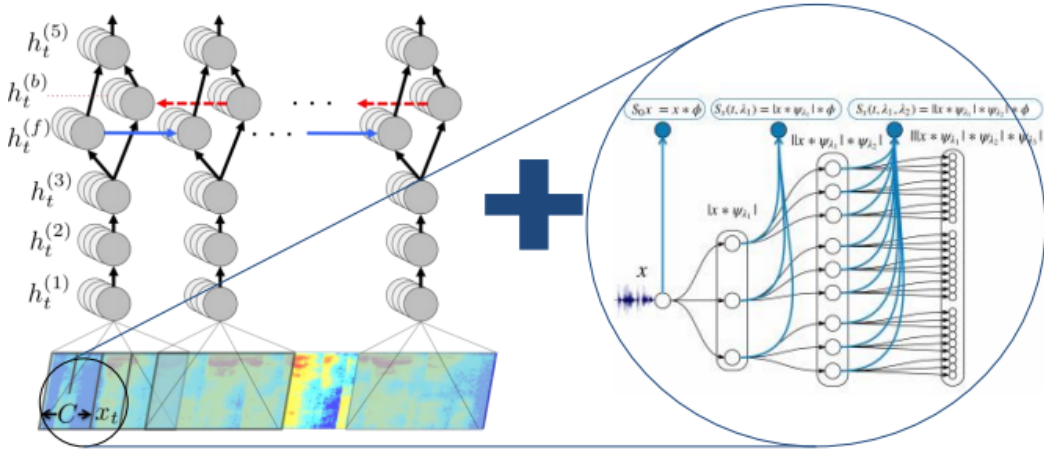
Figure 6.1: Deep scattering Bi-RNN Model

Figure 6.1.

6.3 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 symbols in the alphabet Σ and augmented with the blank symbol.
The posterior output of the CTC network is probability of the symbol given the
speech feature input $p(c|x_t)$ at time t for $t = 1, \ldots, T$ and T is the length of the
input sequence. Also recall two further sets of probabilities also being maintained
by the model being the probability of a blank character p_b and that of a non blank
character p_{nb}.

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 determine the argument that maximises the probability $p(c|x_t)$ at
each time step. Let $C = (c_1, \ldots, c_T)$ be the character string then, the greedy approach

has

$$c_t = arg \max_{c \in \Sigma} p(c|x_t) \tag{6.6}$$

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.

Prefix beam search algorithm Hannun et al. (2014b) 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 W is recovered from acoustic input X at time t by choosing the word which maximising the posterior probability:

$$W_i = arg \max_{W_i \in \Sigma_W} p_{net}(W; X) p_{lm}(W) \tag{6.7}$$

In equation 6.7, the Bayes product of language model prior p_{lm} and the network output p_{net} are utilised to maximise the probability of a particular character-word sequence in the lexicon given by Σ_W. The overall calculation used to derive the final posterior distribution includes word insertion factors (α and β) used to balance the highly constrained n-gram language model.

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 retains only k 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 6.2 attempts to find the string formulated in equation 6.7. Two sets prefixes A_{prev} and A_{nxet} are initialised, such that at A_{nxet} maintains the prefixes in the current time-step while A_{prev} maintains only k-prefixes from the previous time-step. Note that at the end of each time step A_{prev} is updated with only -most probable prefixes from A_{nxet}. Therefore while, A_{nxet} contains all the possible new paths from based on A_{prev} as a Cartesian product of $A_{prev} \times \Sigma \in \mathcal{Z}^k \times \mathcal{Z}^{|\Sigma|}$ where $|\Sigma|$ is the length of Σ. The probabilities of each prefix obtained at each time step is the sum of the probability

of non-blank plus the probability of a blank symbol.

At every time step and for every prefix ℓ currently in A_{prev}, a character from the alphabet Σ is presented to the prefix. The prefix is only extended only when the presented symbol is not a blank or a space. A_{nxet} and A_{prev} 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.

$$p(\ell|x_{1:t}) = (p_{nb}(\ell|x_{1:t}) + p_b(\ell|x_{1:t}))|W(\ell)|^{\beta} \qquad (6.8)$$

$W(\cdot)$ 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.

ℓ_{end} is the variable representing the last symbol in the prefix sequence in A_{prev}. If the symbol presented is the same as ℓ_{end} then the probability of a non-blank symbol, $p_{nb} = 0$. 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

$$p(\ell^+|x_{1:t}) = p(W(\ell^+)|W(\ell))^{\alpha}(p_{nb}(\ell|x_{1:t}) + p_b(\ell|x_{1:t}))|W(\ell)|^{\beta} \qquad (6.9)$$

Note that $p(W(\ell^+)|W(\ell))$ is set to 0 if the current word $W(\ell^+)$ is not in the lexicon. This becomes a constraint to enforce all character strings to constitute of words only in the lexicon. Furthermore, $p(W(\ell^+)|W(\ell))$ is extended to include all the character sequences representing number of words considered by the n-gram language model by constituting the last $n - 1$ words in character sequence $W(\ell)$.

6.4 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. Adam optimisation algorithm (Kingma and Ba,

Table 6.1: GPU Experiments

Experiment	Hours of speech	Total training time	Estimated training
1. 2xGPU 10GB RAM	1	7 days	Completed
2. 2xGPU 10GB RAM	10	150+ days	300 days
3. 5xGPU 15GB RAM	10	17 hours	Completed
4. 5xGPU 15GB RAM	40	5+ days	10 days

2014) was used with initial learning rate of , and a momentum of 0.95 was deployed
to optimise the learning rate.

The network for 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 6.1. For decoding with prefix search we use a
beam size of 200 and cross-validated with a held-out set to find optimal settings
for the parameters α and β. Figure 6.4 shows word error rates for various GPU
configurations and audio data-set sizes.

6.5   Model Baseline

he study by Hannun et al. (2014b) 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 wither
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 (moz, 2019a, Panayotov et al., 2015).

Word Error Rates by this model was optimised after 75 epoch, 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.

6.6   Results

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 is 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.

The results showed that the training of the model was heading towards a very slow convergence as indicated by the slow decrements in training loss. However, we perceive that given the complete dataset to train the model will not only converge but show improvements in word error rates.
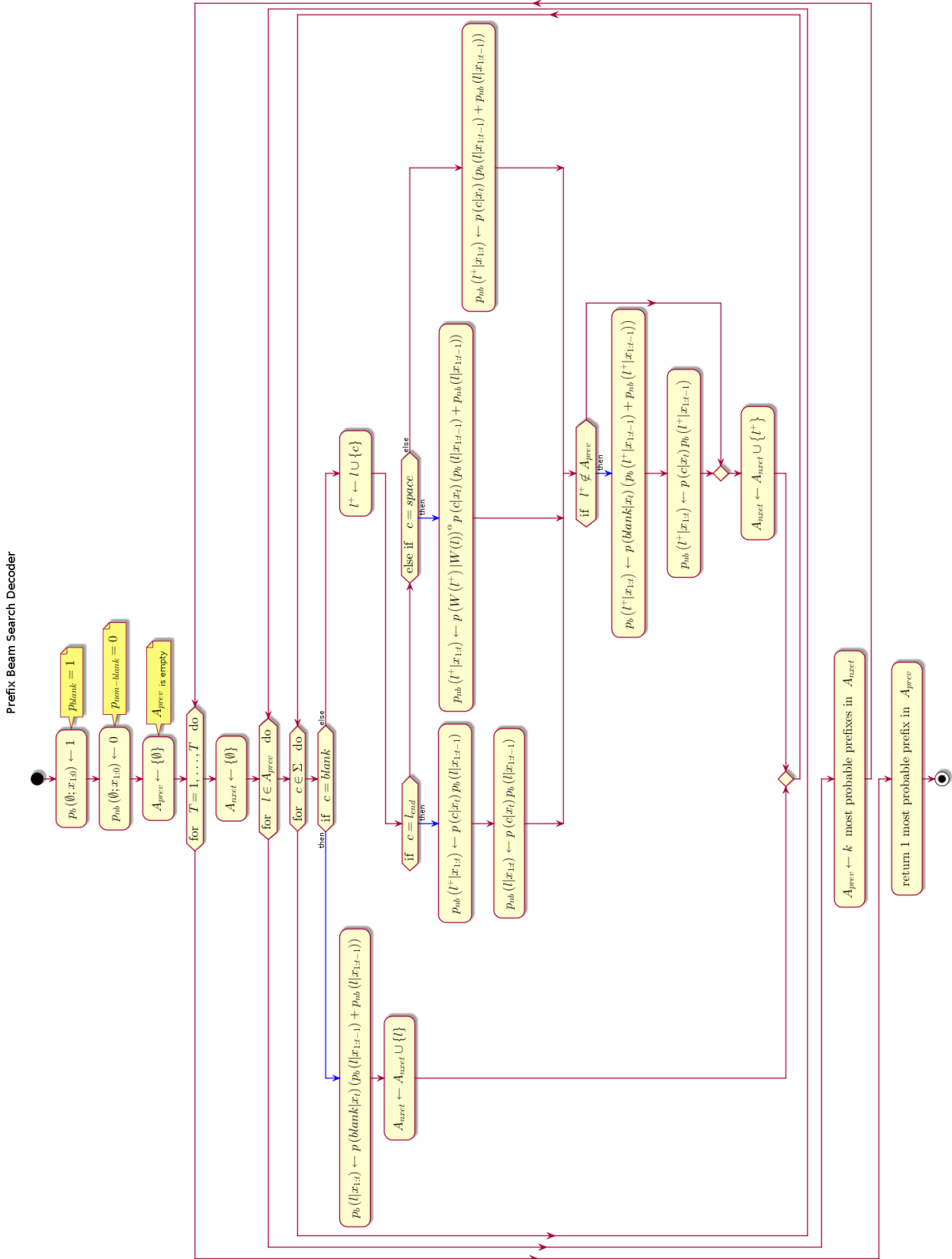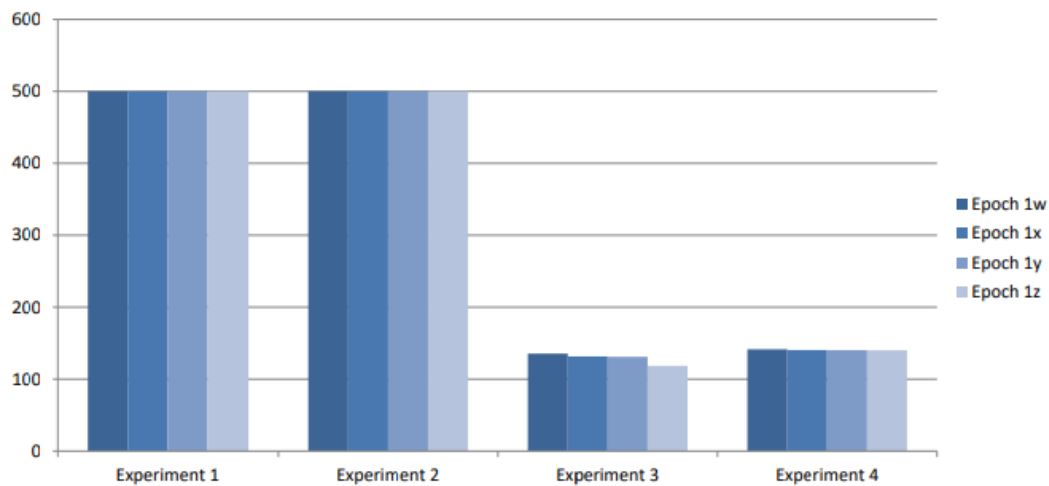
Figure 6.2: Prefix beam search algorithm

Figure 6.3: Training Loss, where $w < x < y < z$ are taken arbitrarily across the total number of epochs
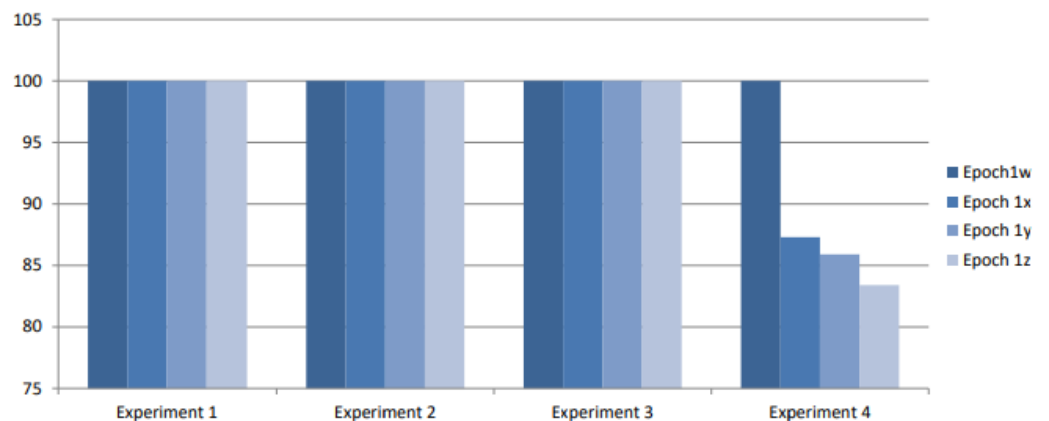

Figure 6.4: WER, where w¡x¡y¡z are taken arbitrarily across the total number of epochs

Chapter 7

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.

7.1 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.

7.2 Attention-based Models

The objective of attention-based networks highlighted by Vaswani et al. (2017) is
to reduce sequential computation while attaining hidden representation across arbi-
trary lengths of sequential input. Mechanisms which have been deployed to achieve
this includes a combination of convolutional and recurrent schemes Gehring et al.
(2017), Kaiser and Bengio (2016), Kalchbrenner et al. (2016). Vaswani et al. (2017)
introduces a transduction model known as a Transformer based on self attention net-
work 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 re-
source training. Vaswani et al. (2017) 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.

Bibliography

Continuous wavelet transform, 2015.

2019a. URL `https://github.com/mozilla/DeepSpeechcommon-voice-training-data`.

Mozilla deepspeech, 2019b. URL `https://voice.mozilla.org/en`.

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

J Andén, L Sifre, S Mallat, M Kapoko, V Lostanlen, and E Oyallon. Scatnet (v0. 2). *Computer Software. Available: http://www.di.ens.fr/data/software/scatnet/. [Accessed: December 10, 2013]*, 0.2, 2014.

Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. In *ISMIR*, pages 657–662. Miami, FL, 2011.

Joakim Andén and Stéphane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.

Claudio Becchetti and Lucio P. Ricotti. *Speech recognition: theory and C++ implementation*. Wiley, New York, 1998. ISBN 0471977306;9780471977308;.

L Becchetti. The behaviour of financial time series: stylised features, theoretical interpretations and proposals for hidden markov model applications. *Speech recognition. Theory and C++ implementation*, 1999.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, 56:85–100, 2014a.

Laurent Besacier, Etienne Barnard, Alexey Karpov, and Tanja Schultz. Introduction to the special issue on processing under-resourced languages. 2014b.

Mikael Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2002.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.9919rep=rep1type=pdf`

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

Jack D Cowan. Discussion: Mcculloch-pitts and related neural nets from 1943 to 1989. *Bulletin of mathematical biology*, 52(1-2):73–97, 1990.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2012.

Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

Li Deng and Xiao Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, 2013.

Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

John Dines, Junichi Yamagishi, and Simon King. Measuring the gap between hmm-based asr and tts. *IEEE Journal of Selected Topics in Signal Processing*, 4(6):1046–1058, 2010.

Sadaoki Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(1):52–59, 1986.

Mark Gales, Steve Young, et al. The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3):195–304, 2008.

Mark John Francis Gales, Shinji Watanabe, and Eric Fosler-Lussier. Structured discriminative models for speech recognition: An overview. *IEEE Signal Processing Magazine*, 29(6):70–81, 2012.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR.org, 2017.

Arnab Ghoshal, Pawel Swietojanski, and Steve Renals. Multilingual training of
deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP),
2013 IEEE International Conference on*, pages 7319–7323. IEEE, 2013.

Jean-Philippe Goldman. Easyalign: an automatic phonetic alignment tool under
praat. 2011.

Alex Graves. *Supervised sequence labelling with recurrent neural networks*. Springer,
2014.

Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recur-
rent neural networks. In *International Conference on Machine Learning*, pages
1764–1772, 2014.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Con-
nectionist temporal classification: labelling unsegmented sequence data with re-
current neural networks. In *Proceedings of the 23rd international conference on
Machine learning*, pages 369–376. ACM, 2006.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recog-
nition with deep bidirectional lstm. In *Automatic Speech Recognition and Under-
standing (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

Frantisek Grezl and Petr Fousek. Optimizing bottle-neck features for lvcsr. In
ICASSP, volume 8, pages 4729–4732, 2008.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich
Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates,
et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint
arXiv:1412.5567*, 2014a.

Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng. First-
pass large vocabulary continuous speech recognition using bi-directional recurrent
dnns. *arXiv preprint arXiv:1408.2873*, 2014b.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp
Koehn. Scalable modified Kneser-Ney language model estimation. In
*Proceedings of the 51st Annual Meeting of the Association for Computa-
tional Linguistics*, pages 690–696, Sofia, Bulgaria, August 2013. URL
https://kheafield.com/papers/edinburgh/estimate_paper.pdf.

Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *The Journal
of the Acoustical Society of America*, 87(4):1738–1752, 1990.

Hynek Hermansky and Nelson Morgan. Rasta processing of speech. *IEEE transac-
tions on speech and audio processing*, 2(4):578–589, 1994.

Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT,
RTRL, EKF and the" echo state network" approach*, volume 5. GMD-
Forschungszentrum Informationstechnik Bonn, 2002.

F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the
IEEE*, 64(4):532–556, 1976. doi: 10.1109/PROC.1976.10159.

Bing-Hwang Juang and S. Furui. Automatic recognition and understanding of
spoken language - a first step toward natural human-machine communication.
Proceedings of the IEEE, 88(8):1142–1165, 2000. doi: 10.1109/5.880077. URL
`http://ieeexplore.ieee.org/document/880077`.

Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances
in Neural Information Processing Systems*, pages 3781–3789, 2016.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex
Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv
preprint arXiv:1610.10099*, 2016.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware
neural language models. In *AAAI*, pages 2741–2749, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980, 2014.

Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and
Sebastian Stober. Transfer learning for speech recognition on a budget. *arXiv
preprint arXiv:1706.00290*, 2017.

Thang Luong, Richard Socher, and Christopher D Manning. Better word represen-
tations with recursive neural networks for morphology. In *CoNLL*, pages 104–113,
2013.

J Lyons. Mel frequency cepstral coefficient (mfcc) tutorial, 2012. URL
`http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-freque`

Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc.
A*, 374(2065):20150203, 2016.

Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet
representation. *IEEE transactions on pattern analysis and machine intelligence*,
11(7):674–693, 1989.

Ian McLoughlin. *Applied speech and audio processing: with Matlab examples*. Cam-
bridge University Press, 2009.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černocký.
Empirical evaluation and combination of advanced language modeling techniques.
In *Twelfth Annual Conference of the International Speech Communication Asso-
ciation*, 2011.

Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep belief networks
for phone recognition. In *Nips workshop on deep learning for speech recognition
and related applications*, volume 1, page 39. Vancouver, Canada, 2009.

Abdel-rahman Mohamed, George E Dahl, Geoffrey Hinton, et al. Acoustic modeling
using deep belief networks. *IEEE Trans. Audio, Speech & Language Processing*,
20(1):14–22, 2012.

Jay F Nunamaker Jr, Minder Chen, and Titus DM Purdin. Systems development
in information systems research. *Journal of management information systems*, 7
(3):89–106, 1990.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Lib-
rispeech: an asr corpus based on public domain audio books. In *2015 IEEE
International Conference on Acoustics, Speech and Signal Processing (ICASSP)*,
pages 5206–5210. IEEE, 2015.

Vijayaditya Peddinti, TaraN Sainath, Shay Maymon, Bhuvana Ramabhadran, David
Nahamoo, and Vaibhava Goel. Deep scattering spectrum with deep neural net-
works. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE Inter-
national Conference on*, pages 210–214. IEEE, 2014.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vec-
tors for word representation. In *Proceedings of the 2014 Conference on Em-
pirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543,
Doha, Qatar, October 2014. Association for Computational Linguistics. doi:
10.3115/v1/D14-1162.

Daniel Povey, Lukáš Burget, Mohit Agarwal, Pinar Akyazi, Feng Kai, Arnab
Ghoshal, Ondřej Glembek, Nagendra Goel, Martin Karafiát, Ariya Rastrow, et al.
The subspace gaussian mixture model—a structured model for speech recognition.
Computer Speech & Language, 25(2):404–439, 2011.

A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny. End-
to-end speech recognition and keyword search on low-resource languages. In
*2017 IEEE International Conference on Acoustics, Speech and Signal Processing
(ICASSP)*, pages 5280–5284, March 2017. doi: 10.1109/ICASSP.2017.7953164.

Charles Ogan D. S. *Okrika: A kingdom of the Niger Delta*. Onyoma Research
Publications, Port Harcourt, Rivers State, Nigeria, 1 edition, 2008.

Tara N Sainath, Vijayaditya Peddinti, Brian Kingsbury, Petr Fousek, Bhuvana Ram-
abhadran, and David Nahamoo. Deep scattering spectra with deep neural net-
works for lvcsr tasks. In *Fifteenth Annual Conference of the International Speech
Communication Association*, 2014.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory
based recurrent neural network architectures for large vocabulary speech recogni-
tion. *CoRR*, abs/1402.1128, 2014. URL http://arxiv.org/abs/1402.1128.

George Saon, Hong-Kwang J Kuo, Steven Rennie, and Michael Picheny. The ibm
2015 english conversational telephone speech recognition system. *arXiv preprint
arXiv:1505.05899*, 2015.

Gary F. Simons and Charles D. Fennig. Ethnologue: Languages of the world, twenty-
first edition., 2018. URL http://www.ethnologue.com.

Paul Smolensky. Information processing in dynamical systems: Foundations of har-
mony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF
COMPUTER SCIENCE, 1986.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Adriana Stan, Yoshitaka Mamiya, Junichi Yamagishi, Peter Bell, Oliver Watts, Robert AJ Clark, and Simon King. Alisa: An automatic lightly supervised speech segmentation and alignment tool. *Computer Speech & Language*, 35:116–133, 2016.

Stanley Smith Stevens, John Volkmann, and Edwin B Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Maarten Versteegh, Roland Thiolliere, Thomas Schatz, Xuan Nga Cao, Xavier Anguera, Aren Jansen, and Emmanuel Dupoux. The zero resource speech challenge 2015. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Ngoc Thang Vu and Tanja Schultz. Multilingual multilayer perceptron for rapid language adaptation between and across language families. In *Interspeech*, pages 515–519, 2013.

Shinji (. e. Watanabe and Jen-Tzung Chien. *Bayesian speech and language processing*. Cambridge University Press, Cambridge, 2015. ISBN 1107055571;9781107055575;.

PC Woodland and Daniel Povey. Large scale discriminative training for speech recognition. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*, 2000.

Ping Xu and Pascale Fung. Cross-lingual language modeling for low-resource speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 21 (6):1134–1144, 2013.

Steve Young. A review of large-vocabulary continuous-speech. *IEEE Signal Processing Magazine*, 13(5):45, 1996. doi: 10.1109/79.536824.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. *Cambridge university engineering department*, 3:175, 2002.

Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION.* Springer, 2016.

Dong Yu, Li Deng, and George Dahl. Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

Neil Zeghidour, Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. A deep scattering spectrum—deep siamese network pipeline for unsupervised acoustic modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4965–4969. IEEE, 2016.