University of Cape Town

Adapting Large-Scale Speaker-Independent Automatic Speech Recognition to Dysarthric Speech

---

Student:
Charles Houston
HSTCHA001

Supervisor:
Mr Stefan S Britz
Co-supervisor:
Dr Ian Durbach

In partial fulfilment of the requirements for the degree of Master of Science at the University of Cape Town

Department of Statistical Sciences

January 5, 2022

Abstract

Despite recent improvements in speaker-independent automatic speech recognition (ASR), the performance of large-scale speech recognition systems is still significantly worse on dysarthric speech than on standard speech. Both the inherent noise of dysarthric speech and the lack of large datasets add to the difficulty of solving this problem. This thesis explores different approaches to improving the performance of Deep Learning ASR systems on dysarthric speech.

The primary goal was to find out whether a model trained on thousands of hours of standard speech could successfully be fine-tuned to dysarthric speech. Deep Speech – an open-source Deep Learning based speech recognition system developed by Mozilla – was used as the baseline model. The UASpeech dataset, composed of utterances from 15 speakers with cerebral palsy, was used as the source of dysarthric speech.

In addition to investigating fine-tuning, layer freezing, data augmentation and re-initialization were also investigated. Data augmentation took the form of time and frequency masking, while layer freezing consisted of fixing the first three feature extraction layers of Deep Speech during fine-tuning. Re-initialization was achieved by randomly initializing the weights of Deep Speech and training from scratch. A separate encoder-decoder recurrent neural network consisting of far fewer parameters was also trained from scratch.

The Deep Speech acoustic model obtained a word error rate (WER) of 141.53% on the UASpeech test set of commands, digits, the radio alphabet, common words, and uncommon words. Once fine-tuned to dysarthric speech, a WER of 70.30% was achieved, thus demonstrating the ability of fine-tuning to improve upon the performance of a model initially trained on standard speech.

While fine-tuning lead to a substantial improvement in performance, the benefit of data augmentation was far more subtle, improving on the fine-tuned model by a mere 1.31%. Freezing the first three layers of Deep Speech and fine-tuning the remaining layers was slightly detrimental, increasing the WER by 0.89%. Finally, both re-initialization of Deep Speech's weights and the encoder-decoder model generated highly inaccurate predictions. The best performing model was Deep Speech fine-tuned to augmented dysarthric speech, which achieved a WER of 60.72% with the inclusion of a language model.

Table of Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence.

ALS Amyotrophic Lateral Sclerosis.

ASR Automatic Speech Recognition.

AWS Amazon Web Services.

BPTT Back-Propagation Through Time.

BRNN Bidirectional Recurrent Neural Network.

CCE Categorical Cross-Entropy.

CER Character Error Rate.

CTC Connectionist Temporal Classification.

DBN Deep Belief Network.

DCT Discrete Cosine Transform.

DFT Discrete Fourier Transform.

DNN-HMM Deep Neural Network Hidden Markov Model.

FFT Fast Fourier Transform.

GPU Graphics Processing Unit.

HMM Hidden Markov Model.

LAS Listen Attend and Spell.

LHUC Learning Hidden Unit Contributions.

LM Language Model.

LSTM Long-Short Term Memory.

MFCC Mel Frequency Cepstral Coefficient.

ReLU Rectified Linear Unit.

RNN Recurrent Neural Network.

RNN-T Recurrent Neural Network Transducer.

SAT Speaker Adaptive Training.

STFT Short-Time Fourier Transform.

WER Word Error Rate.

Chapter 1

Introduction

State-of-the-art speaker-independent[1] automatic speech recognition (ASR) systems have improved to such a degree in recent times that an estimated 46% of US adults make use of a voice assistant such as Apple's Siri or Amazon's Alexa (Pew Research Center, 2017). ASR has been integrated into smart phones, smart watches, smart homes and smart speakers. These systems are developed using a combination of very large labelled datasets and data-driven algorithms, making them effective tools which have found widespread adoption. While these systems work well on speech well-represented in the data, such as typical American English, they fall short when it comes to impaired speech.

De Russis and Corno (2019) investigated the performance of three industrial-grade speech recognition systems on both standard speech and impaired speech. They found that word error rates (WERs)[2] of 15-25% were obtained on standard speech. When applied to severe impaired speech, this error rose to 80-90%. Impaired speech recognition is an exceptionally challenging task – a task that this thesis will seek to explore and improve upon.

The specific type of impaired speech addressed in this thesis is known as dysarthria. This chapter will introduce dysarthria in Section 1.1 and the potential role that speech recognition can play in mitigating its detrimental effects in Section 1.2. Two considerable challenges which stand in the way of effective dysarthric speech recognition – noise and data scarcity – will then be highlighted in Section 1.3. Deep Learning, the modelling paradigm used in many speech recognition systems and in this thesis, will then be introduced at a high level in Section 1.4. Sections 1.5 and 1.6 will discuss the aim and research questions that this thesis poses in some detail, before concluding the chapter by explaining the structure of the remaining chapters in Section 1.7.

1.1 Dysarthria: definition and prevalence

Humans produce most sounds by expelling air from the lungs through the trachea and out through the mouth or nose (Jurafsky and Martin, 2020). The exact manner in which different phonemes[3] are produced is a highly precise operation, whereby multiple vocal organs need to act synchronously to modify the outflow of air appropriately to generate the desired sound. These operations are coordinated by the central and peripheral nervous system (Darley, Aronson, and Brown, 1969).

When the nervous system is damaged, neuromuscular control of the vocal organs can be disrupted. This disruption can cause changes in respiration, phonation, articulation, resonance, and

[1] Speaker-independent speech recognition aims to model speech from a variety of speakers. This is in contrast to speaker-dependent speech recognition which is tailored to a specific individual.

[2] See Section 4.1.5 for a detailed explanation of WERs.

[3] Abstract units of sound that convey a distinction in meaning.

prosody that reduce speaker intelligibility. Dysarthria is a collective name for such neurogenic speech disorders (Darley, Aronson, and Brown, 1969).

Dysarthria comes in six major types, each associated with different speech characteristics. The differing areas of the nervous system that are damaged lead to the following distinct classifications: flaccid, spastic, hypokinetic, hyperkinetic, ataxic, and mixed (Enderby, 2013). The different types of dysarthria are each associated with different speech characteristics. For example, spastic dysarthria is associated with strained, hoarse articulation, while ataxic dysarthria is associated with excess loudness and irregular breakdowns (Enderby, 2013).

Some etiologies of dysarthria are acquired, while others are congenital. Parkinson's disease, stroke, traumatic brain injury, and amyotrophic lateral sclerosis (ALS) are examples of the former, while cerebral palsy is a common example of the latter (Yorkston, 1996).

Although the exact prevalence of dysarthria within specific conditions is often unknown, there is a strong association between diseases characterised by slowly progressive muscle weakness and dysarthria. The prevalence of dysarthria was found to be 46% and 62% in two cohorts of patients with neuromuscular disease, where dysarthria was diagnosed by an experienced speech language pathologist using the Nijmegen Dyarthria Assessment (Knuijt et al., 2013).

Estimates do exist for the prevalence of many conditions that cause neurogenic speech disorders. Between 1980 and 1990, cerebral palsy occurred at a rate of 2.08 per 1000 live births in 13 geographically defined locations in Europe (Johnson, 2002). Worldwide, Parkinson's affects over 6 million people as of 2016 (Dorsey et al., 2018), while ALS is estimated to affect 4.42 people per 100 000 (Xu et al., 2019).

1.2 Role of speech recognition

The conditions which give rise to dysarthria are also often accompanied by physical handicaps that prevent interaction with computers and the environment. This makes automatic speech recognition an especially appealing solution.

A voice-input voice-output communication aid was developed for dysarthric speakers by Hawley et al. (2013). The device harnessed a hidden Markov model (HMM)[4] to convert speech to text, before passing the predictions through a speech synthesizer. While the device worked well in an isolated environment, when applied in a real usage situation its performance dropped significantly. Users of this device reported that it could lead to improved communication if the accuracy could be improved (Hawley et al., 2013).

One study investigated the communication between 15 people with motor neuron disease and their partners over a three year period (Murphy, 2004). Individuals showed a strong preference for using their own voice rather than difficult-to-use augmentative and alternative communication aids[5] such as typewriters. With regards to speech generation, participants were reluctant to accept another voice as their own, particularly because of the inability of assistive devices to replicate intonation and humour. If dysarthric speech recognition could be improved, it would allow users to have the best of both worlds: they could use their own voices while retaining the ability to be understood.

1.3 Challenges associated with dysarthric speech recognition

Two significant challenges standing in the way of improved dysarthric speech recognition are its inherent noise and variability, as well as the limited amount of data available. These problems will briefly be highlighted in more detail.

[4] See Section 2.1.5 for an overview of hidden Markov models.

[5] Any device that assists in communication; encompasses both low-tech and high-tech solutions.

1.3.1 Noise and variability

Speaker-independent speech recognition systems have a number of challenges to overcome. Sources of variability between speakers include: speaker speed, articulation, volume, pronunciation, accent and mannerisms. This can be combined with environmental scenarios such as background noise, distance from the microphone, echoes, and audio quality. The challenge of creating a speech recognition system that is robust to all these factors is considerable, before even introducing dysarthric speech.

State-of-the-art standard speech recognition systems are trained on thousands of hours of standard speech. The vast quantity of data exposes the models to many forms of variability, allowing them to learn robust target functions. This allows them to function effectively, despite the differences in speech characteristics.

In addition to handling speaker variability, these systems can also handle noise, such as the background sounds one might hear at a café or other public place. To represent such data in the training set, the standard approach is to record background noise and to then overlay it on top of the clean audio which contains the relevant speech.

Moore, Venkateswara, and Panchanathan (2018) note that while much work has been done in creating speech recognition systems that are robust to background noise, it does not solve the problem of the noise present in dysarthric speech, which is of a different form. Overlaying background noise on top of standard speech does not translate to improved performance on speakers with hoarse articulation or excess loudness.

1.3.2 Data scarcity

State-of-the-art standard speech recognition systems do not generalize well to dysarthric speech because it is not well enough represented in the data. One solution to this problem would be to simply collect thousands of hours of dysarthric speech and to train a model using that. While this would give the model sufficient opportunity to learn to distinguish the noise from the signal, it is not currently a feasible approach.

The reason this approach is infeasible is that there are myriad ethical and practical difficulties in collecting data from individuals who are often affected by other physical impediments, in addition to that of speech. It is for this reason that large databases of dysarthric speech do not exist. Instead, alternative approaches must be identified and explored in an effort to obtain improved results.

1.4 An overview of Deep Learning

The modelling framework used throughout this thesis is known as Deep Learning[6]. Deep Learning is a subset of machine learning that deals with models that are comprised of multiple layers (hence the "deep" in Deep Learning). The need for multiple layers stems from the notion of a hierarchy of concepts, whereby each concept is built upon a more simple predecessor (Goodfellow, Bengio, and Courville, 2016).

This hierarchical approach allows computers to learn complicated concepts, such as recognizing words or faces, by building them out of simpler concepts. It is an ideal solution to problems which are not easily described by a set of formal mathematical rules: it avoids having to manually specify all of the information that the computer needs and does not require the painstaking stipulation of hand-engineered features (Goodfellow, Bengio, and Courville, 2016). Instead, the model can learn from data in an end-to-end approach.

One of the most notable recent successes of Deep Learning as applied to speech recognition – and a model that has specific relevance to this thesis – is that of Deep Speech 1 (Hannun, Case,

[6]See Section 2.2 for an historical perspective of Deep Learning.

et al., 2014). Deep Speech 1 harnesses large labelled datasets, Deep Learning architectures, and high performance computing to produce a highly effective ASR system. It achieves a WER of 24.01% on an internal test set containing a wide variety of standard speech. When applied to Indian accented speech, Deep Speech 1 acquired a WER of 45.35%. To put this result in context, human transcribers only obtained a WER of 22.15%.

Deep Speech 1 was improved upon by the development of Deep Speech 2 by Amodei et al. (2015). Deep Speech 2 achieves WERs of 13.59% and 22.44% on the internal test set and Indian accented speech, respectively. While Deep Speech 1 performs worse than its successor and is no longer the state-of-the-art, Mozilla has subsequently developed on open-source version that is available to researchers (Mozilla, 2020). For this reason, Deep Speech 1 has been used extensively throughout this thesis to perform experiments on dysarthric speech.

1.5 Aim

The purpose of this dissertation is to investigate ways in which dysarthric speech recognition can be improved. More specifically, the goal is to find out whether a high performing model trained on thousands of hours of standard speech can be successfully fine-tuned[7] to dysarthric speech. The key idea behind using transfer learning is that it could potentially help to circumvent the data scarcity problem associated with dysarthric speech. In addition to investigating fine-tuning, the efficacy of data augmentation, layer freezing and weight re-initialization will be assessed. To this end, a number of different models will be trained, evaluated and compared to one another in an effort to discern which approaches improve performance. The research questions presented in Section 1.6 will guide this process.

1.6 Research questions

It should be noted that the framing of this thesis is exploratory rather than outcomes-oriented. The goal is therefore not to create a model which achieves the very best possible accuracy on dysarthric speech, but rather to see what effect different approaches have on the performance.

1. How does an industrial grade speech recognition model trained on thousands of hours of standard speech perform on dysarthric speech?

To answer this research question, Deep Speech[8] will be evaluated on a test set of dysarthric speech derived from the UASpeech dataset (Section 3.1). Deep Speech is a large-vocabulary continuous-speech recognition system developed by Mozilla and based on a research paper by Baidu (Hannun, Case, et al., 2014). The test set will be made up of 15 dysarthric speakers each uttering commands, digits, the radio alphabet, common words, and uncommon words. Given this baseline result, different models geared towards improving dysarthric speech recognition can subsequently be evaluated on this test set to see if an improvement is observed.

2. Can the performance of this industrial grade model be improved by fine-tuning to dysarthric speech?

The first approach investigated is that of transfer learning. Deep Learning algorithms are extremely data hungry, and large datasets of standard speech are required to train models such as Deep Speech. These models are exposed to a wide variety of speech, and in doing so, the weights of the network are adjusted such that effective feature extraction and prediction strategies are learned.

[7] Transfer learning is an approach in machine learning where a model trained on one distribution is repurposed to another distribution. Fine-tuning of the model's weights is the manner in which the model is repurposed.

[8] Hereafter, Deep Speech refers to the open-source Mozilla Deep Speech, rather than the original models referred to as Deep Speech 1 and Deep Speech 2. See Section 4.6 for a technical description of Mozilla Deep Speech.

This approach is not feasible for dysarthric speech due to the limited amount of data. The idea, then, is to take what was learned in a standard speech model, and adapt it to dysarthric speech. Practically, this question will be answered by fine-tuning the weights of Deep Speech to the UASpeech training data.

3. Does freezing the feature extraction layers during fine-tuning impact performance?

In light of the hierarchy of concepts involved in a Deep Learning model (introduced in Section 1.4), this question aims to identify if freezing the initial layers of Deep Speech will have an impact on performance. It is thought that this approach will lead to an improvement in performance because the number of trainable parameters will be reduced, thus the detrimental impact of overfitting will be lessened. As with the previous question, this question will be answered by fine-tuning Deep Speech to the UASpeech training data, though the weights in the first three layers will be frozen during this process.

4. Does augmenting the impaired speech dataset improve the performance of fine-tuning?

The second approach to improving speech recognition on dysarthric speech is data augmentation. The amount and quality of data available to a machine learning model has a large impact on the ultimate performance of that model, and the recent successes of Deep Learning are only made possible by large, labelled datasets.

Data augmentation is a strategy to increase the amount of training data by creating replicas of the original data which are perturbed in some way. This helps to reduce variance and increase robustness to noise. The approach taken in this thesis is inspired by Park et al. (2019), who showed that time and frequency masks are an effective augmentation strategy in speech recognition.

5. How do models trained only on impaired speech compare to the fine-tuned models?

Firstly, this will be answered by re-initializing Deep Speech and training solely on the dysarthric speech. This will essentially negate anything that was learned when Deep Speech was initially trained on standard speech as all of the weights will once again be randomly initialized – the model will only be able to learn features from the dysarthric speech.

In addition to utilizing Deep Speech, another independent model which makes use of the encoder-decoder recurrent neural network (RNN)[9] architecture will be built and trained solely on dysarthric speech. This model will be separate from Deep Speech and it will be comprised of far fewer weights.

These two models will help to assess whether using transfer learning provides a benefit over training on only dysarthric speech.

1.7 Structure

This chapter has introduced dysarthria and the challenges of dysarthric speech recognition. It has described the aim of this thesis and the research questions it will seek to answer. It has also introduced the modelling framework known as Deep Learning which will be harnessed in this thesis.

Chapter 2 is a literature review which provides context for this research by first addressing the history of speech recognition. This history will start with a brief exploration of the fundamental concepts which have remained relevant since their inception in Section 2.1. This will lead into the Deep Learning era in Section 2.2, where recent advances are of particular importance. Finally,

[9] See Section 4.2.3 for a technical description of the encoder-decoder RNN architecture.

similar work which also tackles the problem of dysarthric speech recognition will be scrutinized in Section 2.3.

Chapter 3 addresses all matters relating to data. In order to investigate dysarthric speech recognition, it is necessary to have a source of dysarthric speech. This source is known as the UASpeech dataset and is described in Section 3.1. In addition to describing and visualizing the UASpeech dataset, this chapter will explain in detail the process of extracting relevant features from the raw audio in Section 3.2. Lastly, Section 3.3 describes the data augmentation strategies applied to certain models.

Chapter 4 will provide all of the technical details necessary to understand the models utilized and trained in Chapter 5. The groundwork for this will be built in Section 4.1, where the basic architecture and training procedure of RNNs is described. Extensions to this setup, such as encoder-decoder RNNs, will be described in Section 4.2. Solutions to the problem of vanishing and exploding gradients will be presented in Section 4.3, while solutions to overfitting can be found in Section 4.4. Strategies to effectively decoding predictions are presented in Section 4.5, and the chapter ends by describing Deep Speech in Section 4.6.

Chapter 5 will first document the implementation procedure in Section 5.1. The observed results will be presented in Section 5.2, where a first approximation at answering the research questions will be provided. This first approximation will be expanded upon in the discussion in Section 5.3, as well as in the manual error analysis presented in Section 5.4.

Chapter 6 will summarize and conclude the dissertation. It will also address limitations and make suggestions for future work.

Chapter 2

Literature Review

Chapter 1 introduced the key challenges and ideas relevant to this thesis. Chapter 2 will build on these ideas by providing historical context through an overview of speech recognition. This overview will be divided into two sections: a pre-Deep Learning historical arc and a Deep Learning historical arc. Each of these will address key discoveries relating to speech recognition. The third and final section of the chapter will review similar work on dysarthric speech recognition. This will focus on work which harnessed either fine-tuning, layer freezing, data augmentation, or a combination of these approaches.

2.1 Speech recognition: pre-Deep Learning historical arc

This pre-Deep Learning historical arc introduces some of the core concepts in speech recognition such as the power spectrum and graph searches. The trajectory of the first two major approaches to speech recognition – template-based approaches and hidden Markov models (HMMs) – will also be discussed.

2.1.1 The power spectrum

Many of the early breakthroughs in automatic speech recognition occurred at AT&T Bell Laboratories. Fletcher (1922) helped to establish the importance of the power spectrum by investigating the relationship it has with sound characteristics and intelligibility. The power spectrum is something that is still of vital importance to speech recognition today, demonstrated by the spectrogram which is typically used as input to acoustic models.[10]

In 1952, the power spectrum was utilized to build a speaker-dependent isolated digit recognition system (Davis, Biddulph, and Balashe, 1952). This device operated by measuring the formant[11] frequencies measured during vowel regions in each respective digit. This isolated digit recognizer was a discrete speech recognition system – it assumed that the input audio was a single digit, thus there was no need to segment the audio to isolate the section which contained the relevant utterance. A phoneme recognizer developed at Kyoto University incorporated the first segmenter which could analyse different sections of the input (Sakai and Doshita, 1962), establishing the first attempt at continuous speech recognition.

2.1.2 Template-based approaches

In the late 1960s, template based approaches began to gain traction, partially due to their ability to circumvent the need for explicit segmentation. To solve the problem of varying speed

[10] An acoustic model takes in audio and produces a probability distribution over characters or words. A language model is a separate entity which helps to rescore those probabilities such that coherent language is produced.

[11] A formant is a concentration in the power spectrum caused by resonance in the vocal tract.

and tempo present in human speech, Vintsyuk (1968) proposed that dynamic programming be used to achieve time alignment between two phrases in order to reasonably assess their similarity. This concept evolved into a template matching approach known as dynamic time warping, which was further built upon by Sakoe and Chiba (1978). Dynamic programming has since been a common theme throughout speech recognition, playing a vital role in HMMs (Section 2.1.5) as well as in certain Deep Learning approaches such as the Connectionist Temporal Classification (CTC) loss (Section 4.1.4).

2.1.3 Graph search

One of the next landmark systems to be built was the Harpy speech recognition system (Lowerre, 1976). The input speech underwent parametric analysis, segmentation, and subsequent template matching. The main contribution, however, was the novel graph search it employed. Taking into account syntactical and word boundary rules, an algorithm based on beam search (Section 4.5.2) could be implemented to identify the sequence with the smallest template-matching distance. This system was the first to make use of a finite state network[12] in its graph search. This can be considered a precursor to HMMs, though efficient optimization of such an approach did not come about until the 1990s.

2.1.4 Acoustic models and language models

During this time, two schools of thought began to emerge. On one hand was AT&T Bell Laboratories, who focused on the acoustic model; on the other was IBM, who concentrated on the language model (Juang and Rabiner, 2004).

AT&T wanted to develop a speaker-independent system that could handle the variability in signal stemming from different speakers. Such a model could be used to automatically route calls at a call centre, for example. The need to create a model robust to acoustic variability lead to research into clustering algorithms for developing sound reference patterns that were applicable to a wider audience (Juang, Levinson, and Sondhi, 1986). Initially these algorithms were template based, though over time the focus shifted to statistical models, essentially paving the way for the development of HMMs.

Meanwhile, IBM were aiming to develop a voice-activated typewriter with the purpose of transcribing words into text (Jelinek, Bahl, and Mercer, 1975). While this system was speaker-dependent, it aimed to accommodate as broad a vocabulary as possible. To correctly model the grammatical and syntactical aspects of language, sequences of words needed to be evaluated probabilistically. To this end, the n-gram language model (Section 4.5.1) was incorporated into their systems. The n-gram language model is an extremely powerful tool capable of capturing the probabilistic nature of grammar (Shannon, 1948) and has been of vital importance to large vocabulary speech recognition since.

2.1.5 Hidden Markov models

The emergence of HMMs marked a shift away from template-based approaches and towards statistical approaches. The concept of the HMM originated at the Institute for Defence Analyses at Princeton (Baum, 1972), though it was not until the 1980s that it became widely used. The publication of the theory by Ferguson (1980) and Levinson, Rabiner, and Sondhi (1983) helped to shift HMMs into the mainstream. This marked a notable landmark in the history of speech recognition as HMMs came to dominate speech recognition technology for decades to come.

An HMM is a doubly stochastic process; it models both the probability of transitioning from one hidden state to another in the Markov Chain, and the probability of emitting a given observation from within a particular state (Rabiner, 1989). This framework is capable of modelling both the

[12] A finite state network is a network made up of a finite number of states where only a single state can be active at any one time. Different states can be reached through transitions.

intrinsic variability in speech through the emission probabilities, and the structure of language through the Markov Chain. Given a training set of labelled utterances, the Baum-Welch Algorithm (Baum, 1972) can be employed to find the optimal set of parameters. Given an unknown utterance, the HMM can then be used to provide the probabilities that the utterance was a given word represented by the model.

Initially, HMMs were successfully applied to discrete observations, as well as to random observations, that were well modeled using log-concave probability density functions (Juang and Rabiner, 2004). The simple log-concave density function proved to be a bottleneck in the performance of HMM speech recognition performance. Bell Laboratories introduced mixture densities (Juang, 1985; Juang, Levinson, and Sondhi, 1986) which rectified this problem and helped facilitate robust large vocabulary speaker-independent speech recognition models.

2.2 Speech recognition: Deep Learning historical arc

While much work was done on Deep Learning prior to the rise of HMMs, it has only very recently gained traction and become the state-of-the-art approach to speech recognition. This is due, in part, to the fact that Deep Learning has undergone many so-called "winters", or sustained periods of disinterest and lack of funding. This subsection provides some background to the development of Deep Learning and the key milestones along the way.

2.2.1 Early developments

One of the first steps towards modern day Deep Learning was taken by McCullough and Pitts (1943) with the introduction of the McCulloch-Pitts model. This model was a primitive attempt at modelling the human brain using electrical circuits. Although seen as a precursor to artificial neural networks, its weights needed to be manually tuned rather than updated through a learning process.

Soon after, Donald Hebb proposed what is now known as the Hebbian learning rule (Hebb, 1949). This signalled the first framework for optimizing the weights within a network. This rule stated that the weights between neurons should change proportionally to the product of their activations. Put another way: "cells that fire together, wire together".

Building on both of these discoveries, Rosenblatt (1958) introduced the perceptron, an early stage neuron model that demonstrated the ability to learn. Although a major breakthrough, Minski and Papert (1969) published a book proving that perceptrons could only be applied to a select range of linearly-separable problems and were incapable of learning the simple XOR[13] function. This lead to pessimism concerning artificial intelligence (AI) and the commencement of the first AI winter.

2.2.2 Back-propagation

What the learning models still lacked was a fast and efficient manner in which to calculate the gradient of a network's loss function with respect to its weights (Section 4.1.6). This would in turn facilitate the application of stochastic gradient descent, helping to achieve the goal of substituting hand-engineered features for multi-layer trainable networks (LeCun, Bengio, and Hinton, 2015).

The solution to this problem, which is now called back-propagation, was first discovered by Werbos (1974). The effects of the first AI winter were still being felt at this time, so it was not until the mid 1980s that this idea came to fruition. Interest was once again sparked in Deep Learning, thanks in part to work by Hopfield (1982). Back-propagation itself became widely known when Lecun (1985) and Parker (1985) independently discovered it.

[13] An XOR gate is a Boolean logic gate that returns 1 if the two input bits are different, and 0 if they are the same.

The resolution of this missing link was followed by one of the first examples of the power of back-propagation coupled with neural networks; LeCun, Boser, et al. (1989) developed a convolutional neural network optimized using back-propagation and capable of classifying low-resolution images. This lead to the first widespread application of machine learning in the form of digit recognition at the US Postal Service (Kamath, J. Liu, and Whitaker, 2019).

The reputation of neural networks was further bolstered by Cybenko (1988), who proved that feed-forward neural networks with a single hidden layer and non-linear sigmoid activation functions were "universal approximator functions" (a concept that will be discussed in more detail in Section 4.1.3). This essentially addressed the concerns put forward by Minski and Papert (1969).

2.2.3 Recurrent neural networks

As multi-layer perceptrons do not explicitly incorporate time, Jordan (1986) set about creating an architecture which could perform supervised learning on sequences. Jordan extended the multi-layer perceptron by adding context units which provided a means for information to be passed across timesteps.

Elman (1990) built upon this work by simplifying the Jordan network and demonstrated that XOR logic could be extended to the time domain. Back-propagation needed to be modified in order to work correctly in such temporal-based neural networks, leading to the invention of back-propagation through time (Werbos, 1990).

The development of the Jordan and Elman networks signalled an important milestone in the creation of a machine learning architecture capable of modelling temporal data and set the groundwork for recurrent neural networks (RNNs).

2.2.4 Deep architectures and optimization

What has been discussed thus far in the history of Deep Learning has encompassed only shallow-structured architectures, such as the multilayer perceptron with a single layer. Such architectures have been shown to be highly effective at modelling well-constrained problems, though can be less effective when it comes to more complicated real world problems.

The information processing systems in the human brain, such as are used for vision and hearing, are highly layered systems (Baker et al., 2009); this is suggestive that the corresponding problems in the machine learning realm – image and speech recognition – can also be tackled with deep architectures (Deng and Yu, 2014).

Multilayer perceptrons with many hidden layers can be trained with back-propagation by starting optimization at some random set of weights. Learning is difficult in such a setting due to the presence of local optima and saddle points, making shallow architectures more appealing as a global optimum can be reached at the expense of modelling complexity.

This problem was empirically alleviated by stacking Restricted Boltzmann Machines[14] on top of one another to produce deep belief networks (DBNs) (Hinton, Osindero, and Teh, 2006). The key aspect to these models was an unsupervised pre-training step which, when coupled with a corresponding multilayer perceptron for weight initialization, lead to improved results (Deng and Yu, 2014).

RNNs suffer from a similar problem to deep feed-forward neural networks in so far as gradients can explode or vanish when calculated over many timesteps. The problem of training RNNs is exacerbated by the need to keep track of long-term dependencies in many sequence modelling problems. Bengio, Simard, and Frasconi (1994) showed that it is very difficult for standard RNNs to learn to store information for many timesteps.

[14]Restricted Boltzmann Machines are generative, stochastic two-layer neural networks.

Hochreiter and Schmidhuber (1997) introduced an explicit form of memory into RNNs known as long-short term memory (LSTM), which proved to be much more effective at modelling long-term dependencies and helped to alleviate the problem of vanishing and exploding gradients. They did this by incorporating a memory cell which is connected to itself at each timestep. This memory cell is passed through a gated cell which learns when to clear and when to retain the information.

2.2.5 Widespread adoption

One of the first applications of the DBN pre-training approach was to speech recognition, where neural networks were shown to outperform Gaussian mixture models (Mohamed, Dahl, and Hinton, 2012). This was made possible by the advent of efficient graphics processing units (GPUs) which have allowed researchers to train models 10 to 20 times faster (Raina, Madhavan, and Ng, 2009), and have been a vital factor in facilitating the growth of Deep Learning. While the DBN pre-training step helped to bolster the development of neural networks and end the second AI winter, it turned out that it was only really necessary for small datasets (LeCun, Bengio, and Hinton, 2015).

Of late, Deep Learning has not only achieved good results, it has achieved state-of-the-art results. Deep Learning revolutionized the field of image recognition in 2012 when a convolutional neural network halved the error rates of competing approaches in the ImageNet contest (Krizhevsky, Sutskever, and Hinton, 2012). With regard to speech recognition, in 2018 a team at Google used sequence-to-sequence Deep Learning to improve upon the previous state-of-the-art HMM-LSTM[15] in the challenging voice search task (Chiu et al., 2018). Such positive results have lead large tech companies to invest further into the promise of Deep Learning to improve the products and services they offer (LeCun, Bengio, and Hinton, 2015).

2.2.6 Deep Speech

At the time of its publication, Deep Speech was a hugely successful example of the power of end-to-end Deep Learning in speech recognition. Instead of requiring complex pipelines involving extensive hand engineering, Deep Speech takes advantage of RNNs and large amounts of data to produce a robust-to-noise speaker-independent speech recognition system (Hannun, Case, et al., 2014). Deep Speech requires no concept of phonemes, nor statistical representations obtained from HMMs, though it outperforms such traditional approaches, scoring 16.0% word error rate (WER) on the Switchboard Hub5'00[16] test set.

In addition to demonstrating the power of end-to-end Deep Learning and outperforming traditional approaches, Deep Speech demonstrated the ability of Deep Learning to model noisy data. While the final model was trained on approximately 7380 hours of conversational and read speech, data synthesis was also applied to improve performance on noisy audio (Sapp, Saxena, and Ng, 2008). This was achieved by synthesizing audio recorded in a noisy environment by overlaying background noise on top of clear speech. The Lombard effect[17] (Junqua, 1993) was also captured by collecting data in simulated noisy environments. When compared to the Apple Dictation, Bing Speech, and Google API speech recognition systems, Deep Speech reported substantially better results on noisy speech (Hannun, Case, et al., 2014).

2.3 Similar work

This section covers similar work and provides a body of literature with which to compare the results of this thesis. Work involving fine-tuning, layer freezing, and data augmentation will be discussed, as well as the model that achieves state-of-the-art on the UASpeech dataset.

[15] Long-short term memory (LSTM), details of which can be found in Section 4.3.2.
[16] A dataset comprised of 300 hours of conversational telephone speech from 40 conversations.
[17] The active change of volume, pitch, rate, and duration of syllables that speakers make in noisy environments.

2.3.1 Application of fine-tuning and layer freezing

One of the most extensive attempts at applying Deep Learning to dysarthric speech was made by a team at Google (Shor et al., 2019). They investigated two model architectures: a bidirectional recurrent neural network transducer (RNN-T) and Listen Attend and Spell (LAS) with a total of 49.6 million parameters and 132 million parameters, respectively. These models were pre-trained on standard speech before applying fine-tuning to both dysarthric and accented speech. The standard speech was derived from the 1000-hour Librispeech dataset, while 36.7 hours of dysarthric speech was collected from 67 people with ALS.

The RNN-T model outperformed the LAS model, achieving a WER of 10.8% on mild dysarthria after fine-tuning. This is a substantial improvement over the base model, which was trained only on standard speech and obtained a WER of 33.1% on mild dysarthria. When the base model was fine-tuned to more severe dysarthria, its performance was improved from a WER of 59.7% to 20.9%, an absolute improvement of 39.8%. It should be noted that the authors refer to performance on "severe" dysarthria, though they exclude speakers with an FRS[18] score of 0.

With regards to layer freezing, both the LAS and RNN-T models performed best when the entire model was fine-tuned. The RNN-T achieved 91% of the relative WER improvement by just fine-tuning the joint layer and the first layer (out of a possible five layers) of the encoder. As a final observation, the researchers found that 71% of the relative WER improvement comes from just 5 minutes of dysarthric speech data.

Eberhard and Zesch (2021) used Mozilla Deep Speech to compare the performance of layer freezing schemes in fine-tuning to German and Swiss. While Shor et al. (2019) found that fine-tuning the entire network lead to the best results, Eberhard and Zesch (2021) observed that freezing one or more layers was better than no layer freezing.

While freezing any number of layers in Deep Speech was beneficial, they found that freezing the first two fully-connected layers was optimal. For the German language, fine-tuning all layers produced a WER of 63%, while fine-tuning all but the first two produced a WER of 44%, a total of 19% better. Regarding fine-tuning to Swiss, the benefit was less substantial, though still evident – freezing improved the WER from 76% to 67%, a difference of 9%.

Interestingly, the authors found that freezing as many as five layers (leaving a single trainable layer) still lead to good results, showing that the features learned from English are transferable to German and Swiss.

2.3.2 Application of data augmentation

Augmentation was successfully applied in the form of vocal tract length perturbation, tempo perturbation and speed perturbation by Geng et al. (2020). The modelling was implemented using a deep neural network system. Learning hidden unit contributions (LHUC) based speaker adaptive training (SAT) was further applied to model the variability amongst impaired speakers. The system with augmentation produced a WER of 26.37% on the UASpeech test set, an absolute reduction of 2.92% over the baseline system which did not harness data augmentation.

Tempo and speed augmentation were also harnessed by Vachhani, Bhat, and Kopparapu (2018) to predict 19 command words in the UASpeech dataset. In this context, healthy speech was used to simulate dysarthric speech. An absolute improvement of 4.24% and 2.00% was observed when tempo and speed augmentation were applied, respectively.

Adversarial data augmentation for dysarthric speech was investigated by Jin et al. (2021). Deep convolutional generative adversarial networks were used to convert healthy speech into dysarthric speech, mimicking select dysarthric speakers. When applied on top of tempo and speed augmentation, this generative adversarial approach ensures fine-grained spectro-temporal characteristics

[18] FRS stands for Functional Rating Scale. Scores range from 0 to 4, with zero being incomprehensible and 4 being standard speech.

are emulated. This augmentation approach resulted in an absolute WER reduction of 3.05% over the baseline system which did not harness data augmentation, resulting in a final WER of 25.59% on the UASpeech test set.

2.3.3 State-of-the-art model

The current state-of-the-art on the UASpeech dataset of dysarthric speech is currently held by S. Liu et al. (2021). The UASpeech test set is made up of dysarthric utterances of commands, digits, radio alphabet, common words and uncommon words. The researchers perform a thorough investigation into different Deep Learning architectures, augmentation strategies, speaker adaptation, and cross-domain generation of features.

The lowest WER of 25.21% was achieved on the dysarthric speech test set by a deep neural network system which employed neural architecture search, audio-visual speech recognition technology, data augmentation and LHUC-based SAT. The details of this system can be found in S. Liu et al. (2021).

2.4 Conclusion

This chapter started by presenting a pre-Deep Learning historical arc of speech recognition, starting with the establishment of the power spectrum and building up to hidden Markov model era. This was followed by a Deep Learning historical arc which touched on key points such as the innovation of backpropagation. This historical analysis provides context for the choices made in this thesis, such as the decision to use Deep Learning. The final section in this chapter addressed similar work and provided a body of literature with which the results of this thesis can be compared.

Chapter 3

Data

Training a model to learn features in the data and to make predictions is only possible given a source of training data. In addition, the aim of investigating ways in which dysarthric speech recognition can be improved is only possible given a source of dysarthric speech. Fortunately, Kim et al. (2008) has provided such an opportunity through permitting access to the UASpeech dataset.

The goal of Section 3.1 is to provide insight into the UASpeech dataset. Feature extraction, which is a vital step in any speech recognition system pipeline, will then be discussed in detail in Section 3.2. Finally, Chapter 3 will end by presenting the data augmentation strategies used in certain models in Section 3.3.

3.1 UASpeech

In addition to describing the UASpeech dataset, speaker intelligibility, manual processing and a brief exploratory data analysis will be presented in this section.

3.1.1 Description

The Universal Access Research Speech Database (Kim et al., 2008), or UASpeech for short, is comprised of dysarthric speech contributed by 15 speakers with cerebral palsy. 11 of the speakers have spastic cerebral palsy, two mixed and two unknown. Given that the majority of speakers have spastic dysarthria, the speech is characteristically strained, hoarse, hyper-nasal and slow, with imprecise articulation (Enderby, 2013).

A seven-channel microphone array was used to capture the audio. However, only the seventh channel was used because although some speakers had channels missing, this channel was present across all speakers. The speakers can be identified by an ID which begins with either an M or an F (for Male or Female), and is followed by a two digit number e.g. M01. 11 of the speakers are male and the remaining four are female.

Speech utterances were collected from each speaker in three distinct recording sessions, which will henceforth be referred to as blocks. In all three blocks the following words were uttered, producing three repetitions of each for every speaker: 10 digits (zero to nine), 19 commands (backspace, delete, enter, etc.), 26 radio alphabet (alpha to zulu), and 100 common words (the, of, and, etc.).

While these words were repeated in all three blocks, another section of words known as the uncommon words were distinct across blocks. In block one, 100 uncommon words were uttered by each speaker (e.g. naturalization, faithfulness, etc.). In block two, a different set of 100 uncommon words were uttered by each speaker (e.g. needlepoint, Nuremberg, etc.), and yet another different set in block three. Figure 3.1 depicts this setup diagrammatically.

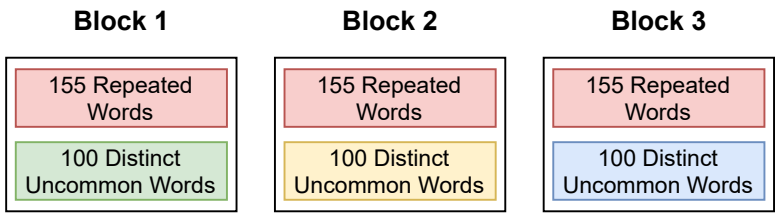

Figure 3.1: Visualization of a single speaker's utterances in the UASpeech dataset. The 155 repeated words are comprised 19 commands, 10 digits, 26 radio alphabet, and 100 common words. The 100 uncommon words are unique to each block, resulting in a total of 300 uncommon words uttered once by each speaker.

In each of these blocks, each speaker utters 255 single word utterances. Multiplied across the three blocks, this results in 765 distinct utterances by each speaker. While 765 distinct utterances were obtained from each speaker, this consisted of only 455 distinct words – 300 uncommon words and 155 repeated words.

All 15 speakers adhered to the exact same configuration depicted in Figure 3.1, uttering the same words in the same order. The total number of distinct utterances across all 15 speakers totals 11436. This is slightly less than 765×15 because speakers F03 and F04 have 25 and 14 utterances missing, respectively.

In addition to the dysarthric speech described above, the UASpeech has an ancillary control dataset made up of standard speech (Kim et al., 2008). A total of 9945 standard speech utterances were obtained from 13 age-matched healthy speakers. The data were recorded in exactly the same manner as depicted in Figure 3.1, with each speaker producing 765 utterances of 455 different words.

3.1.2 Intelligibility

An additional dimension to the UASpeech dataset is the evaluation of intelligibility level for each speaker, the results of which can be viewed in Figure 3.2. Intelligibility was measured by recruiting five unique listeners to transcribe dysarthric speech for each dysarthric speaker.

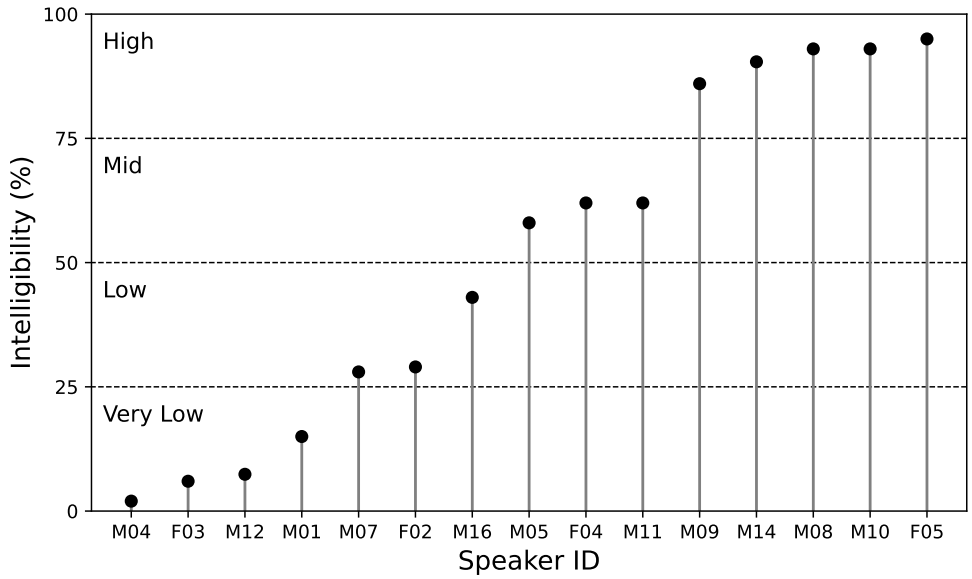

Figure 3.2: Intelligibility scores of the dysarthric speakers sorted in ascending order.

The standard speech listeners were recruited such that they were between 18 and 40 years old and were native American English speakers. It was also required that they have no speech disorder or experience with speech disorders, and no training in phonetic transcription.

The listeners were required to transcribe 225 words taken from block 2 (Kim et al., 2008). A dysarthric speaker's intelligibility was calculated by averaging the correct transcriptions across all five listeners. Each speaker was then classified into one of four categories depending on their intelligibility score: very low (0-25%), low (26-50%), mid (51-75%) and high (76-100%).

3.1.3 Manual processing

It was identified that some audio clips in the dataset contained additional audio not associated with the utterance of the relevant word. This occurred when there was an interaction between the dysarthric speaker and the instructor which was not trimmed prior to release of the dataset. These interactions involved the dysarthric speaker seeking clarity on pronunciation, technical issues, or even external disruptions.

The presence of these interactions could be detrimental to the modelling process because the label would not correspond to the underlying audio clip. It was desirable to identify and remove these interactions while avoiding having to manually process every example in the dataset. The following procedure, which applies to a single speaker, was conceived:

1. Sort the audio clips for the speaker in descending order of clip length. These interactions lead to inflated times such that they are most likely at the top of this sorted list.
2. Starting with the longest clip, listen to the audio and if an interaction is heard, trim it from the audio while retaining the portion pertaining to the word being uttered. Do not trim audio based on stuttering, repetition or silence, only trim based on external speech.
3. Continue until 10 consecutive clips contain only audio from the dysarthric speaker and not the instructor.

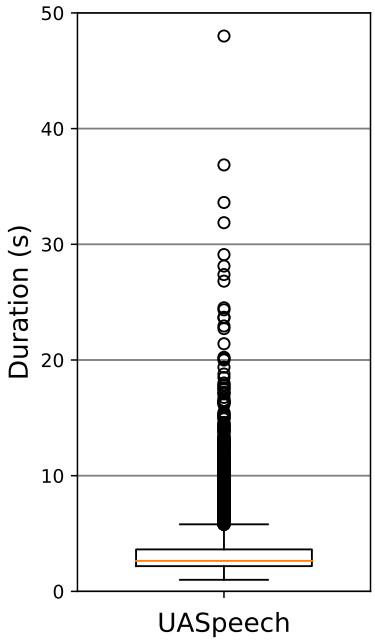

Figure 3.3: Box-plot demonstrating the spread of clip durations in the UASpeech dataset after the application of manual processing.

This procedure was repeated for all 15 speakers and a total of 38 audio clips were trimmed. F05 required the most trimmed audio at 17 clips. The distribution of the processed clips is shown in Figure 3.3.

The median duration occurs at 2.6 seconds, the lower quartile occurs at 2.2 seconds and the upper quartile at 3.6 seconds. It follows that half the non-outlier data fall between 2.2 and 3.6 seconds. It is also evident that a number of outliers exist, with the largest of these occurring at 48 seconds. It should be noted that many of the audio clips do contain silence before, after, or on both sides of the utterance. Therefore, the length of the clips does not necessarily map to the time it takes for a speaker to utter a given word.

3.1.4 Exploratory data analysis

While the word count, structure of the data, and the clip duration have already been discussed, there remains room to address some specific features of the raw waveform. As alluded to in the introduction, impaired speech can differ from standard speech in a multitude of ways.

In their efforts to adapt acoustic models to dysarthric speech, Mengistu and Rudzicz (2011) identified various pronunciation errors committed by dysarthric speakers from the TORGO[19] dataset. These errors are framed in terms of their phonetic transcriptions[20] because phonemes relate directly to pronunciation in a way that letters do not. Two examples of these errors are presented here to provide insight into how impaired speech can differ from standard speech.

The first is an example of what is known as an omission of a word-final stop consonant. The phonetic definition of a consonant is a phoneme that is produced by blocking the airflow in some way (Jurafsky and Martin, 2020). This is in contrast to vowels which generally have less obstruction from vocal organs. More specifically, a stop consonant is produced by completely blocking airflow for a time, before explosively releasing that air.

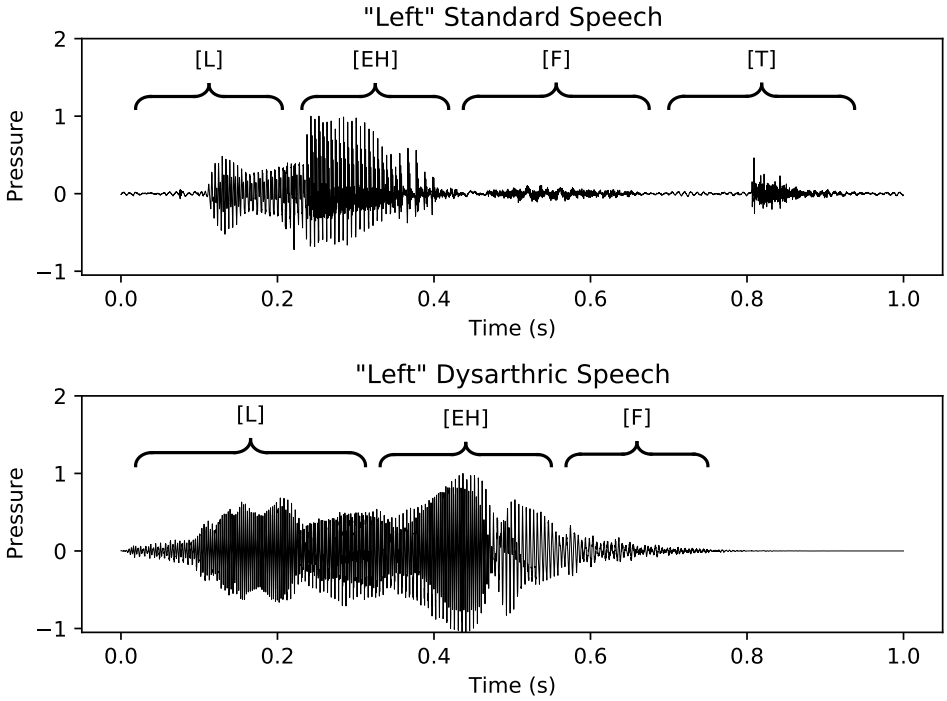

Figure 3.4: Example of the omission of the word-final consonant [T] by speaker F03.

[19] A dysarthric speech dataset consisting of speech from seven individuals with cerebral palsy and ALS (Rudzicz, Namasivayam, and Wolff, 2010).

[20] Appendix A provides an entire phonetic alphabet. Phonemes are placed in square brackets for clarity.

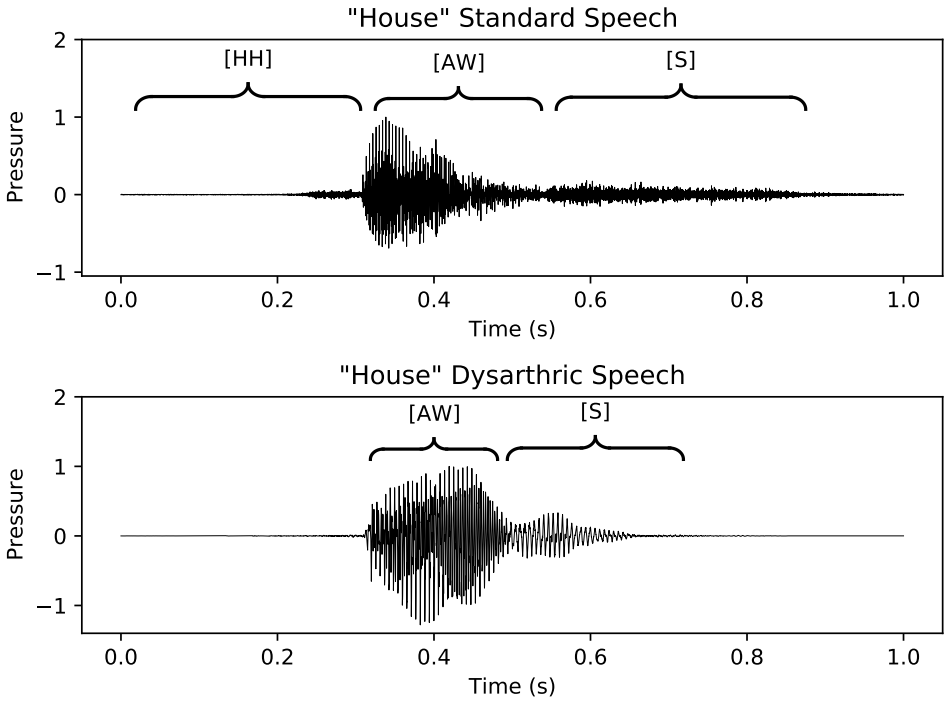

Figure 3.5: Example of initial [HH] deletion by speaker M04.

The top half of Figure 3.4 demonstrates the presence of [T], the word-final stop consonant in the word "left". This is a standard speech utterance derived from the Speech Commands dataset (Warden, 2018). The bottom half provides an example of a dysarthric utterance derived from the UASpeech dataset where the word-final stop consonant is clearly absent.

The second example provided shows the deletion of a voiceless glottal fricative. Voiceless phonemes are produced when the vocal folds are apart and are not vibrating (Jurafsky and Martin, 2020). Fricatives are formed when airflow is restricted, but not completely blocked. This produces turbulent airflow and a characteristic hissing sound. Figure 3.5 compares a standard speech and dysarthric speech utterance of the word "house", where [HH], the voiceless glottal fricative in question, is omitted in the latter.

It should be noted that these examples are cherry-picked for demonstrative purposes. The inter-speaker variability between dysarthric speakers means that some speakers may not make these particular errors at all. Intra-speaker variability must also be considered, as the same speaker may only make certain errors based on their level of tiredness, for example. The main takeaway is that dysarthric speech differs from standard speech and that the omission of certain phonemes can make the task of dysarthric speech recognition much harder as compared to standard speech recognition.

3.2 Feature extraction

Audio is collected via a microphone which converts air pressure to a voltage (Kamath, J. Liu, and Whitaker, 2019). This voltage is converted from analog form to digital form through sampling which occurs in an analog-to-digital converter. The result is a vector of finite length comprised of discrete numbers representing the air pressure over time. The raw waveforms depicted in Figures 3.4 and 3.5 are obtained by plotting this array.

These raw waveforms are high dimensional and can be difficult to model, thus they usually undergo some form of feature extraction before being passed through speech recognition models. An established way in which to achieve this is to convert the raw waveform into its Mel-frequency

cepstral coefficients (MFCCs). The key steps in calculating MFCCs are as follows:

Pre-emphasis: this filter is applied to the input signal and has the effect of amplifying high frequencies. This serves to balance the frequency spectrum, as high frequencies tend to have lower magnitudes compared to low frequencies (Kamath, J. Liu, and Whitaker, 2019).

$$y_t = x_t - \gamma \times x_{t-1} \tag{3.1}$$

Equation 3.1 shows the equation for the filter, where a common value for γ is 0.97. x_t is the original signal at the current timestep, and y_t is the pre-emphasized signal at the current timestep.

Framing: the audio clip is split into a number of overlapping frames. To use a tangible example, the authors of Deep Speech used a frame length of 32 ms and a hop length of 20 ms (Hannun, Case, et al., 2014). Every frame contains 32 ms of the original signal, and the left-most 12 ms of the frame overlaps with the previous frame, while the right-most 12 ms overlaps with the next frame. The purpose of this step is made more clear by the explanation of the Fourier Transform that follows.

Windowing: each frame is scaled by a special function which has the effect of dampening the signal at both the start and end of the frame (Kamath, J. Liu, and Whitaker, 2019). This minimizes the effects of sharp changes in frequency that can occur at the borders of each frame.

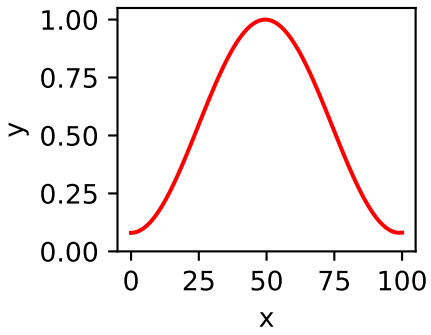

Figure 3.6: Hamming function.

One such framing function, known as the Hamming function, is displayed in Figure 3.6. The centre of the Hamming function has a value of one, which will have no effect on the original signal. At the edges, however, it drops to almost zero, thus dampening the original signal substantially.

Fourier Transform: the Discrete Fourier Transform (DFT) can be understood as a change of basis (Prandoni and Vetterli, 2008). The original basis is that of the time domain, and the target basis is the frequency domain. More specifically, a finite-length signal of length N is decomposed into N sinusoidal components. This decomposition allows one to find the contribution of different frequencies and to discover hidden signal properties that are not evident in the time domain.

The Fourier basis is orthogonal and is described in signal notation by Equation 3.2, which is in complex exponential form.

$$w_k[n] = e^{j\frac{2\pi}{N}nk} \qquad n, k = 0, 1, ..., N-1 \tag{3.2}$$

Where N is the length of the signal, n is the timestep ranging from 0 to $N-1$, and w_k is the k^{th} Fourier basis vector, where k also ranges from 0 to $N-1$. It should be noted that each basis vector is itself a sinusoid with frequency $\frac{2\pi}{N}k$.

To find the coefficients of the Fourier basis vectors, one must simply take the dot product between the Fourier basis vector in question, and the signal. This analysis formula takes the form of Equation 3.3, also represented in signal notation.

$$X[k] = \sum_{n=0}^{N-1} x[n] w_k[n] \qquad k = 0, 1, ..., N-1 \tag{3.3}$$

Where $X[k]$ is the k^{th} Fourier coefficient, w_k is the k^{th} Fourier basis vector, and x is the signal, with timesteps ranging from 0 to $N-1$. The dot product of a complex exponential is itself a complex exponential, thus the resultant coefficient will have both a real and an imaginary part.

While this is the formal definition of the DFT, in practice it can be computationally slow to compute, particularly for large signals. To circumvent this problem, another algorithm known as the Fast Fourier Transform (FFT) is used. The FFT reduces the time complexity of performing the DFT from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ (Prandoni and Vetterli, 2008). This provides a computationally feasible manner in which to transform a signal from the time domain to the frequency domain.

The DFT is applied to every frame obtained from the framing step in what is referred to as the Short-Time Fourier Transform (STFT). The next step is to calculate the magnitude of each coefficient at each timestep by taking the absolute value. This amplitude is usually squared and normalized by the number of points considered, N, ultimately producing what is known as the power, depicted in Equation 3.4. The result of this process is a spectrogram which displays the power as a function of both frequency and time.

$$P = \frac{|X[k]|^2}{N} \tag{3.4}$$

As a final note regarding the Fourier Transform, a trade-off necessarily takes place between time granularity and frequency granularity (Prandoni and Vetterli, 2008). If shorter frames are used (for example 16 ms rather than 32 ms in the case of Deep Speech), then the DFT will be applied to more frames and will have greater temporal resolution. The trade-off is that each timestep will have a lower frequency resolution as N will be smaller, thus fewer Fourier basis vectors can be used. If one opts for longer frames, then the opposite will be true – greater frequency resolution at the cost of temporal resolution. One must find a balance between these two extremes.

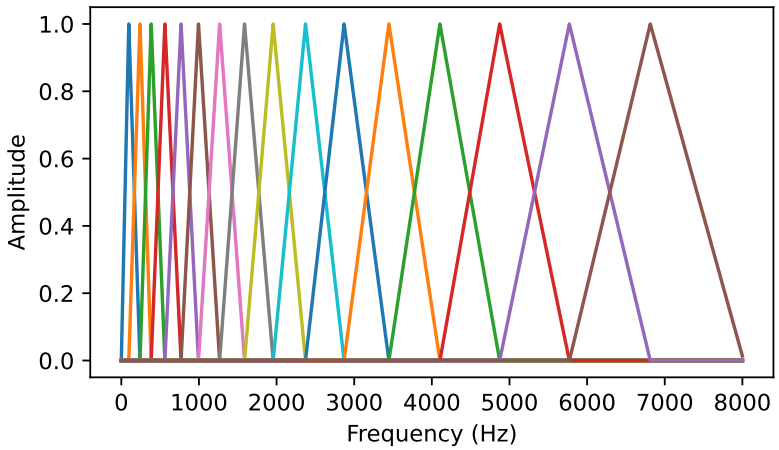

Figure 3.7: Example of 16 Mel filters which could be used to transform a spectrogram to the Mel scale.

Mel filter bank: human auditory perception is non-linear with respect to frequency. Rather, we are more discriminative at lower frequencies than we are at higher frequencies. Therefore, a change from 200 to 400 Hz will seem more significant than a change from 1200 to 1400 Hz, despite the absolute difference being equal.

To better approximate the human auditory system, the Mel scale was introduced (Stevens and Volkmann, 1940). Unlike Hertz, a change in 200 Mels will be perceptually identical no matter where the difference occurs on the Mel scale. Figure 3.7 illustrates an example of Mel filters. Notice how the filters are closer together at lower frequencies and further apart at higher frequencies, mimicking human auditory perception.

The output of this step is a weighted sum of the spectral features that correspond to each filter (Kamath, J. Liu, and Whitaker, 2019). The spectrogram produced from the STFT has thus been transformed to a Mel spectrogram. In cases where decorrelating the filter bank coefficients is not of vital importance, the Mel spectrogram is passed through the log function and is used as the endpoint of pre-processing. Figure 3.8 displays an example of both a raw waveform and the resultant log Mel spectrogram.

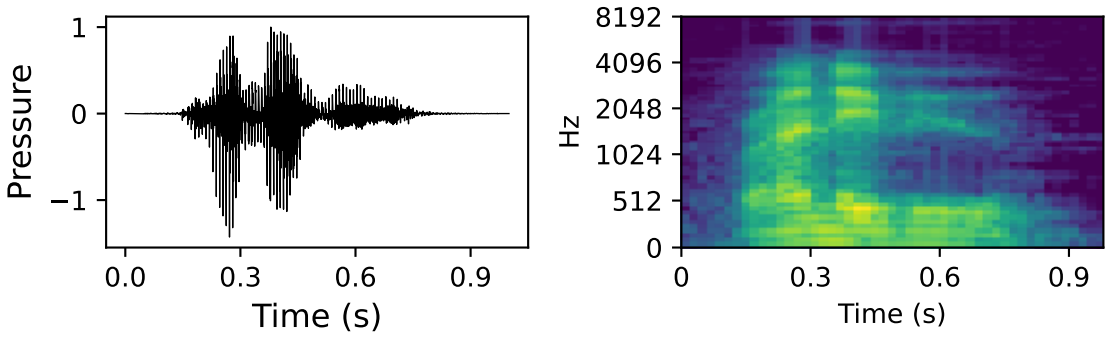

Figure 3.8: Raw waveform and resultant log Mel spectrogram of speaker M10 (high intelligibility) uttering "rendezvous".

Discrete cosine transform: the filter bank coefficients produced in the previous step are often highly correlated with one another. As a final step, the discrete cosine transform (DCT) can optionally be applied to decorrelate these coefficients. The DCT maps the filter bank coefficients into the time domain by compressing the input data into a set of cosine coefficients that describe the oscillations in the function (Kamath, J. Liu, and Whitaker, 2019). This final step produces an MFCC and marks the end of the pre-processing pipeline.

3.3 Data augmentation

When training a model on a small dataset, one problem that is often encountered is that of overfitting. Though overfitting will also be discussed in Section 4.4, it is worth introducing here. Overfitting occurs when the model being trained uses its additional degrees of freedom to fit to idiosyncrasies in the data, such as specific examples and noise, rather than to the underlying target function (Abu-Mostafa, Magdon-Ismail, and Lin, 2012). A key problem in all machine learning applications is ensuring that the model not only performs well on the data it was trained on, but that it generalizes to unseen data.

In addition to using regularization to reduce variance and thus mitigate the problem of overfitting (Section 4.4.1), data augmentation can be harnessed. While data augmentation is commonly associated with image recognition, where horizontal reflections and translations are commonly applied to achieve state-of-the-art models (Krizhevsky, Sutskever, and Hinton, 2012), it is also a viable strategy in speech recognition.

One of the most successful examples of data augmentation in the realm of speech recognition is that of SpecAugment (Park et al., 2019). The augmentation regime applied in SpecAugment consists of three strategies: time masking, frequency masking, and warping, each of which is applied directly to the log Mel spectrogram. Although this strategy is simple, it leads to state-of-the-art results on the Librispeech 960h and Switchboard 300h tasks when applied in conjunction with an LAS model.

Park et al. (2019) found that warping had a very small benefit and recommend that it be the first augmentation strategy to be dropped given any computational limitations, especially since it is the most expensive of the three augmentation strategies to perform. As such, only time masking and frequency masking will be investigated. These two augmentation strategies are explained in more detail below and are depicted visually in Figure 3.9.

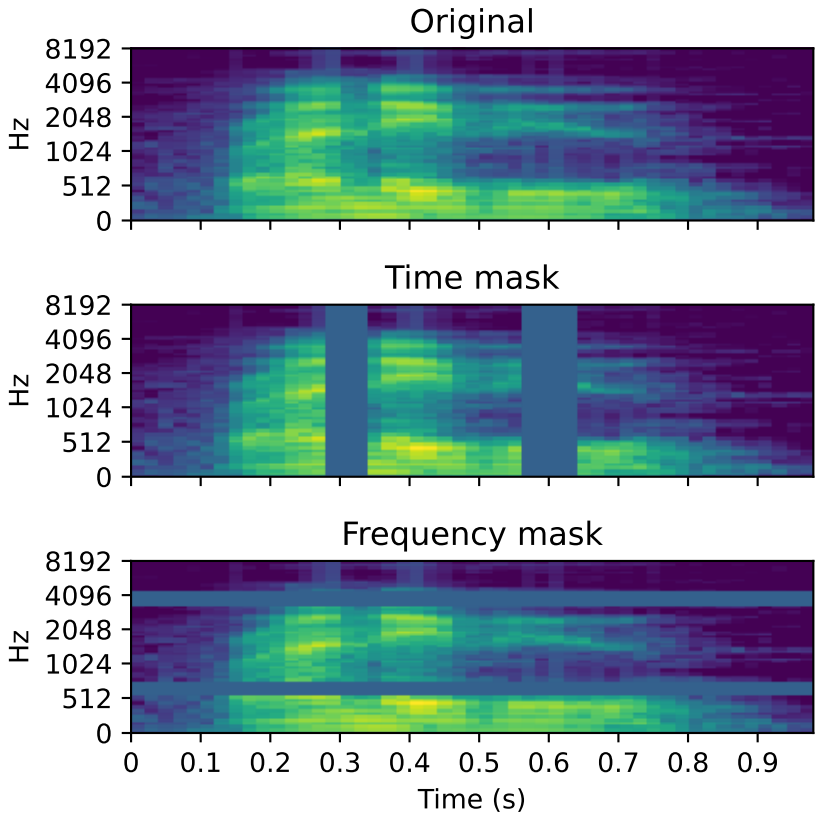

Figure 3.9: Original spectrogram, time masked spectrogram and frequency masked spectrogram of speaker M10 (high intelligibility) uttering "rendezvous". Two masks are applied to each.

The motivation for masking temporal and frequency information stems from a desire to build robust models which can generalize effectively and do not rely on any one feature to make predictions. The log Mel spectrograms are normalized to have a mean of zero, thus any subsequent reference to masking refers to setting values to zero, which corresponds to the mean value of the spectrogram.

Frequency mask: f consecutive frequency bands in the log Mel spectrogram are masked, starting at f_0 and ending at $f_0 + f$. f_0 is chosen from a discrete uniform distribution over $[0, v - f]$, where v is the number of Mel coefficients. f is also chosen from a discrete uniform distribution over $[a, b]$, where a and b are parameters that must be stipulated.

Time mask: operates in the same manner as the frequency mask, but on the time axis of the log Mel spectrogram. t consecutive timesteps in the log Mel spectrogram are masked, starting at t_0 and ending at $t_0 + t$. t_0 is chosen from a discrete uniform distribution over $[0, \tau - t]$, where

τ is the number of timesteps. t is also chosen from a discrete uniform distribution over $[a, b]$. The specific augmentation regime applied in this thesis is addressed in Section 5.1.

3.4 Conclusion

Two pivotal components to any Deep Learning model are data and model architecture. Chapter 3 has established the first of these through providing insight into the UASpeech dataset of dysarthric speech. Both the manual processing and feature extraction strategies applied to this dataset were documented. In addition, the technical detail of the data augmentation methods was provided. The second key component will be addressed in the form of the methodology presented in Chapter 4.

Chapter 4

Methodology

While Chapter 2 introduced the history of Deep Learning and the key ideas relating to speech recognition, this chapter will explore those key ideas from a technical standpoint. The author assumes that the reader has an understanding of linear algebra, calculus and feed-forward neural networks, as these fundamentals will not be covered here. Rather, the emphasis of this chapter will be on recurrent neural networks (RNNs), a model class which builds on feed-forward neural networks, but which is specifically applicable to sequential data.

The chapter will start by addressing the architecture and optimization of RNNs in Section 4.1. This will present the basic formulation, while Section 4.2 will provide certain extensions that can be made. The problem of vanishing and exploding gradients will be addressed in Section 4.3, while Section 4.4 will identify strategies to mitigate overfitting. Approaches to effectively decoding predictions will be discussed in Section 4.5, and the chapter will end by analyzing Mozilla Deep Speech in Section 4.6.

4.1 Recurrent neural networks

This section will start by listing some of the inherent advantages of RNNs. This will be followed by a logical progression through RNNs, starting with forward-propagation in Section 4.1.2, and ending with back-propagation in Section 4.1.6. The sections in between will cover activation functions, loss functions, and error metrics in Sections 4.1.3 through 4.1.5.

4.1.1 Advantages over feed-forward neural networks

RNNs are a variation on fully-connected neural networks which are especially efficient at processing sequential data. More specifically, they are a superset of feed-forward neural networks which incorporate a temporal component through the addition of recurrent edges that span adjacent timesteps Lipton (2015). The sequential data might come in the form of text, speech, video, or even DNA. RNNs can then be harnessed to tackle problems such as machine translation, speech recognition, video activity recognition, and DNA sequence analysis.

RNNs have a number of advantages over fully-connected neural networks. Firstly, fully-connected neural networks are fixed in structure. Therefore, a maximum input length is required in order to process sentences or audio clips. Any examples with shorter length must be padded with zeroes, while any longer examples must be clipped. Alternatively, fully-connected neural networks can be tailored to implicitly model time, such as through sliding window approaches. In contrast, RNNs provide a flexible framework whereby sequences of arbitrary length can be processed and represented in the hidden state (Kamath, J. Liu, and Whitaker, 2019).

Secondly, because fully-connected neural networks process all inputs simultaneously, there is no room for consideration of sequential dependencies – the input features are assumed to be inde-

pendent (Kamath, J. Liu, and Whitaker, 2019). When it comes to sequential data, predictions at the current timestep may be dependent on observations which occurred much earlier in the sequence. Consider the problem of identifying the correct tense of the verb in the following sentence:

Yesterday, in the middle of a dangerous thunderstorm, I ran five kilometres.

The verb "ran" is past tense based on the adverb "yesterday", which occurs much earlier in the sentence.

Thirdly, fully-connected neural networks are not able to share features learned at different positions. If the number of inputs increases then the number of weights must also increase – the number of trainable parameters can quickly grow beyond a computationally feasible level. RNNs can apply the same weight matrices over and over again despite an increasing number of inputs. This ensures that the number of parameters that needs to be trained remains at a manageable level.

4.1.2 Forward-propagation

The base RNN can be described in two equations which are applied at every timestep (Lipton, 2015).

$$h^{(t)} = g_1(W_{hh}h^{(t-1)} + W_{hx}x^{(t)} + b_h) \tag{4.1}$$

$$\hat{y}^{(t)} = g_2(W_{yh}h^{(t)} + b_y) \tag{4.2}$$

Equation 4.1 calculates the hidden state, $h^{(t)}$, while Equation 4.2 uses that result to calculate the prediction, $\hat{y}^{(t)}$. Each RNN cell in Figure 4.1 is an application of Equations 4.1 and 4.2. These equations are "recurrently" applied to every timestep in the input. This takes place until every input has produced a corresponding output, thereby completing the process of forward propagation.

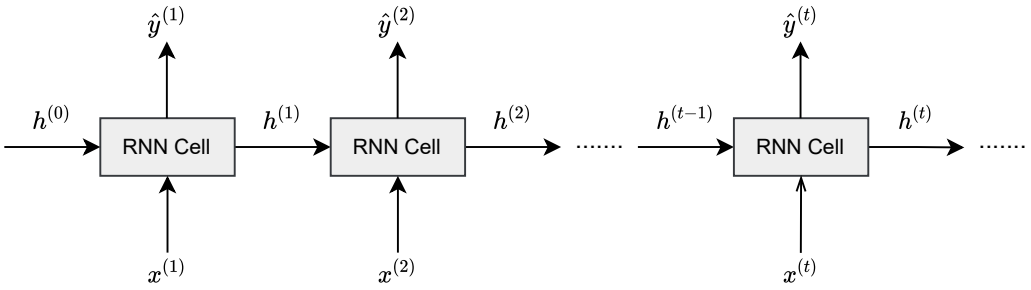

Figure 4.1: Many-to-many RNN.

Wherever present, the superscripts in parentheses, $^{(t)}$ and $^{(t-1)}$, represent the current and previous timesteps, respectively. g_1 and g_2 are known as activation functions and will be discussed in more detail in Section 4.1.3. The initial hidden state, $h^{(0)}$, is initialized as a matrix of zeros. By convention, the lengths of the input and output sequences are denoted by T_x and T_y, respectively.

The base RNN is comprised of three weight matrices distinguishable by their subscripts, W_{hh}, W_{hx}, and W_{yh}, as well as two bias vectors, b_h and b_y. These weight matrices and bias vectors are at first randomly initialized, though through the process of gradient descent are updated to produce learned parameters. This process of optimization is the topic of Sections 4.1.6 and 4.2.1.

The three weight matrices that must be learned, W_{hh}, W_{hx}, and W_{yh}, each have different roles. W_{hx} is responsible for incorporating the information from the incoming feature vector, W_{hh} does the same but for the previous hidden state, and W_{yh} learns a transformation to the output.

It is worth noting that the hidden state $h^{(t)}$ is a function of all preceding input vectors, $x^{(1)}$ up to $x^{(t)}$, demonstrating the sequential modelling capabilities of this model class. It also becomes evident that the weights in Equations 4.1 and 4.2 are independent of the number of entities in the input sequence, demonstrating the malleable structure of RNNs and their ability to share parameters across timesteps.

This particular formulation of an RNN, where every timestep produces a prediction, is known as a many-to-many RNN (Karpathy, 2015). It need not be the case that every timestep produces an output. A single output at the final timestep, also referred to as a many-to-one architecture, is also viable and is useful in applications such as sentiment classification. One-to-many architectures are used in applications such as music generation.

4.1.3 Activation functions

A linear model can by definition only represent linear functions (Goodfellow, Bengio, and Courville, 2016). However, most problems faced in the real world are highly non-linear. Activation functions are a means by which non-linearity can be introduced into neural networks.

Feed-forward neural networks with at least a single hidden layer and a "squashing" activation function have been proven to be universal approximator functions (Cybenko, 1989). Therefore, any continuous function can be approximated up to an arbitrary accuracy and a sufficiently large neural network will be capable of learning any function. This does not mean that the training algorithm will be able to find such a function, but it does demonstrate the expressive power of neural networks.

While first proven for feed-forward neural networks, the universal approximator property has subsequently been extended to RNNs (Schafer and Zimmermann, 2007). It stands to reason that activation functions are a vital component in the architecture of neural networks.

The following activation functions satisfy the non-linearity property and are differentiable, thus facilitating the application of gradient-based optimization.

Hyperbolic tangent

An activation function commonly used in RNNs is the hyperbolic tangent, or tanh, depicted in Equation 4.3 and Figure 4.2.

$$g(z) = \tanh z \tag{4.3}$$

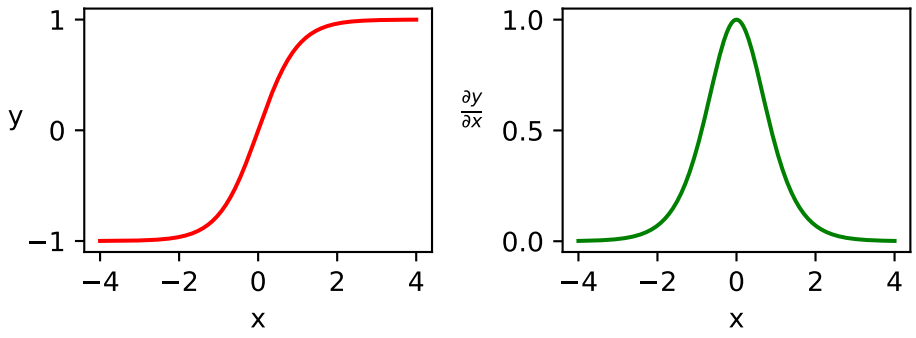

Figure 4.2: tanh activation function and its derivative.

The tanh activation function is a squashing function that transforms an input to the range {-1, 1}. An advantage of the tanh activation function is that it is centred at zero, avoiding the possible occurrence of internal covariate shift[21], something that occurs in another commonly used activation function known as the sigmoid (Kamath, J. Liu, and Whitaker, 2019).

One downside of the tanh activation is that saturation of gradients occurs at both high and low inputs, producing gradients very close to zero (as is evident in Figure 4.2). This is detrimental in backpropagation, where long-term dependencies will not be captured due to vanishing gradients, something that will be discussed further in Section 4.3.

Rectified linear unit

An activation function which helps to alleviate the impact of saturating gradients is the rectified linear unit (ReLU), which can be seen in Equation 4.4 and Figure 4.3.

$$g(z) = max\{0, z\} \tag{4.4}$$

Even as inputs increase in the positive direction, the gradient will still be equal to one. The ReLU activation is renowned for its computational efficiency as the max operation is much faster to compute than the exponential operations required in tanh activations (Goodfellow, Bengio, and Courville, 2016).

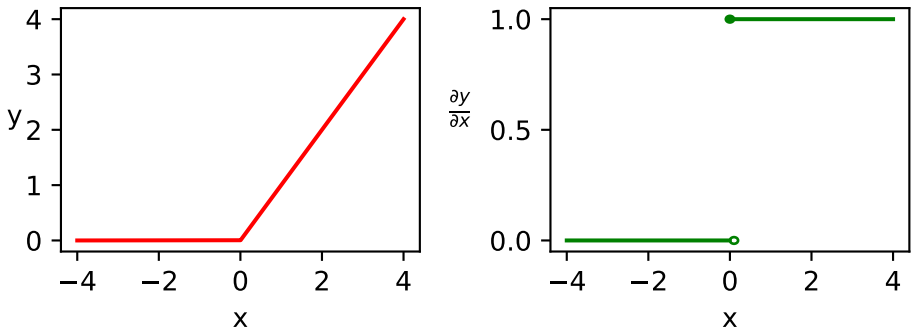

Figure 4.3: ReLU activation function and its derivative.

A downside of the ReLU is that the zero gradient in the negative direction can cause neurons to "die" during training, preventing them from contributing to the network. As many as 40% of neurons can die if the learning rate is set too high (Kamath, J. Liu, and Whitaker, 2019). One way in which to prevent this is to use the leaky ReLU, which introduces a parameter that allows for a small gradient for negative inputs.

Softmax

When the problem at hand is that of classifying observations into different classes, it is desirable to output a probability distribution over classes. While the softmax is not commonly used as an activation function, it is related to activation functions and is used to output a vector of probabilities from a network.

$$g(z) = \frac{e^{z_i}}{\sum_{i=1}^{C} e^{z_i}} \tag{4.5}$$

Equation 4.5 shows the softmax computation (Kamath, J. Liu, and Whitaker, 2019), where C is the number of classes. The function takes in a vector of real numbers, z, and normalizes it

[21] A change in the distribution of a network's activations (Ioffe and Szegedy, 2015).

into a vector of probabilities which sum to one. It does so by applying the exponential function to each input, and then normalizing each by the sum of all such exponentials.

4.1.4 Loss functions

Being able to evaluate the network's performance is the first step to improving it. The loss function is a means by which the performance of a network can be evaluated. If the network is making predominantly incorrect predictions, then one would expect the loss to be very high. On the other hand, if accurate predictions are being made then the loss should be low.

Categorical cross-entropy

Given that the problem of speech recognition is that of classification, a probability distribution over output classes must be produced. This is usually achieved using the softmax activation function at the final layer to output a vector of predicted probabilities, $\hat{y}$. The true probability vector, y, will be one-hot encoded with all indices having values of zero, except for that of the current class which will have a probability of one. The categorical cross-entropy (CCE) loss function, shown in Equation 4.6, takes these two vectors and produces a single value representing the loss (Kamath, J. Liu, and Whitaker, 2019).

$$CCE = -\sum_{i=1}^{C} y_i \log \hat{y}_i \tag{4.6}$$

If $y_i = 1$, but the prediction $\hat{y}_i$ is close to zero, then the loss will be very high because taking the negative log of a small value is a large value. If, on the other hand, the predicted probability $\hat{y}_i = 1$, then the loss will be zero. It follows that the loss will be small when the network makes accurate predictions, but increases as the predicted probabilities diverge from the true probabilities.

Connectionist temporal classification

Another less commonly used loss function is the Connectionist Temporal Classification (CTC) loss. The CTC loss is one way in which to achieve sequence-to-sequence modelling in a machine learning context (Graves et al., 2006). When it comes to speech recognition, the CTC loss can handle variable length inputs and outputs without the need to align input audio features to their corresponding target word, phoneme or character.

The CTC loss is often paired with RNNs, as shown in Figure 4.4, because they conveniently output a probability distribution over classes at each timestep. This probability distribution includes a blank token which plays an important role in calculating the CTC loss. To attain the output sequence, any repeated characters are collapsed (e.g. $AAABBB$ becomes AB), and any characters separated by a blank are also collapsed (AA_AAA becomes AA). The blank token therefore allows for the repetition of characters, and is also used to predict periods of silence in the audio.

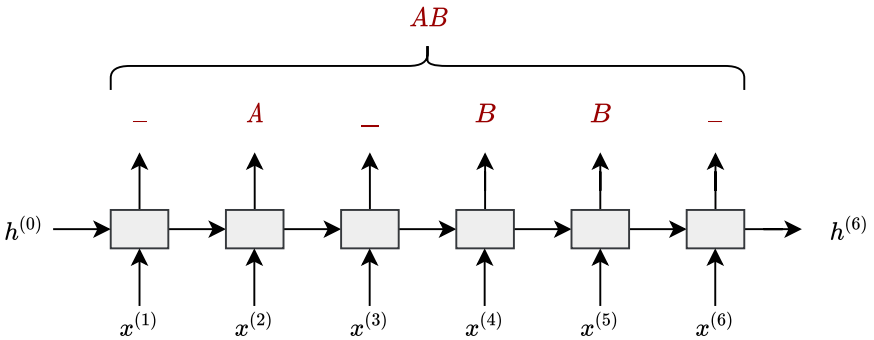

Figure 4.4: Example of CTC loss as applied to an RNN.

There are many ways that these rules could be applied to achieve a desired output. For example, AB can stem from $\_A\_BB\_$, as with Figure 4.4, or from a number of other configurations of these tokens, such as $\_AAA\_B$. The CTC loss sums over the probabilities of all possible alignments that lead to the correct target. A naive brute force approach is computationally infeasible, thus dynamic programming is employed to reduce the time complexity of this algorithm.

Figure 4.5 (inspired by Hannun (2017)) illustrates how this dynamic programming solution operates. The target sequence, Y (AB in this instance), is placed vertically on the left hand side of the figure. It is modified by placing blank tokens between every character, as well as at the beginning and end of the sequence. This new sequence which includes blanks will be referred to as Z. The probability at a given node, $\alpha_{s,t}$, represents the CTC score of the subsequence $Z_{1:s}$ after t timesteps.

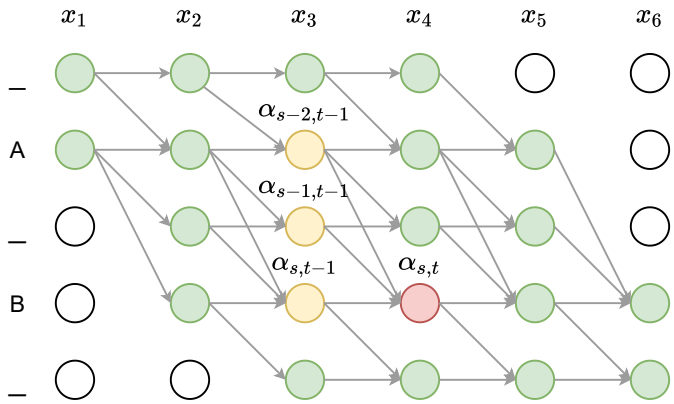

Figure 4.5: Dynamic programming solution to calculating the probability of an output sequence using the CTC loss.

At any given timestep, the prediction can remain at the current token, or move on to the next token. Harnessing dynamic programming, $\alpha_{s,t}$ can be calculated by merging previous sequences which lead to the same target. This can be done in one of two ways, depending on whether the previous token was a character or a blank. The red node in Figure 4.5 can be calculated using Equation 4.7. The top yellow node, $\alpha_{s-2,t-1}$, is included in the equation because the preceding blank token can be skipped without changing the prediction.

$$\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \times \mathbb{P}_t(Z_s|X) \tag{4.7}$$

Nodes where the preceding token is a character cannot include the $\alpha_{s-2,t-1}$ term because it would lead to omission of a desired character in the target sequence. The leads to the use of Equation 4.8, which is the same as Equation 4.7, but with the exclusion of $\alpha_{s-2,t-1}$.

$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \times \mathbb{P}_t(Z_s|X) \tag{4.8}$$

Figure 4.5 provides a route for every possible sequence that produces the target, AB, for an input sequence of length six. The desired probability, $\mathbb{P}(Y|X)$, can be derived by summing the final two nodes at the final timestep. Both nodes are included because the final blank token is optional. Equation 4.9 presents the formal definition of the CTC loss for a single example (Hannun, 2017):

$$\mathbb{P}(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} \mathbb{P}_t(c_t|X) \tag{4.9}$$

Where the sum is over all valid sequences which lead to the desired target, the product is over each timestep and c_t is the output character at timestep t. Given the probability $\mathbb{P}(Y|X)$, the model is trained to minimize the negative log likelihood, as per Equation 4.10 (Hannun, 2017).

$$\mathcal{L} = \sum_{X,Y \in \mathcal{D}} -\log \mathbb{P}(Y|X) \tag{4.10}$$

The CTC loss is calculated by simply summing and multiplying the output probabilities. As such, the weights of the RNN are differentiable with respect to the CTC loss. This permits the application of gradient descent, a topic that will be addressed in Section 4.1.6 and 4.2.1.

4.1.5 Error metrics

While the loss function provides a means by which to evaluate the model during optimization, it cannot easily be used to compare different models and to analyse performance. Error metrics provide a more interpretable means to assess model performance.

Word error rate

The most commonly used error metric in speech recognition is the word error rate (WER). The WER measures the cost of restoring the true output sequence starting from the predicted sequence (Morris, Maier, and Green, 2004). The formula for the WER is shown in Equation 4.11.

$$WER = \frac{I + D + S}{N} \times 100 \tag{4.11}$$

Where I, D and S are the counts of word insertions, deletions and substitutions, and N is the total number of words in the true sequence. The WER is calculated using a dynamic programming algorithm known as the Levenshtein distance (Levenshtein, 1965). It should be noted that the WER can exceed 100% as the number of insertions, substitutions and deletions could be greater than the length of the target sequence.

To provide a tangible example of a WER, consider the following target sentence:

"I'm going to make him an offer he can't refuse"

The value for N is 10 as this is the length of the target sentence. Consider the following predicted sentence.

"I'm goeng to make an offer he can't not refuse"

The predicted sentence has substituted "goeng" for "going", "him" has been deleted, and "not" has been inserted. The WER is therefore 30%.

Character error rate

Equation 4.12 demonstrates that the formula for the character error rate (CER) is identical to that of the WER. The only difference is that the insertions, deletions, substitutions and target sequence length are calculated at the character level, rather than at the word level.

$$CER = \frac{I + D + S}{N} \times 100 \tag{4.12}$$

The CER has the benefit of being more fine-grained than the WER. Consider the word "Californea", a misspelling of "California". While this would result in a WER of 100%, the CER would be much more conservative at 10% (the length of the target is 10, while a single substitution of "e" for "i" has been made). One could argue that most of the meaning is retained in the misspelled version, thus attributing an error of 100% is quite harsh.

4.1.6 Back-propagation through time

Back-propagation through time (BPTT) is the "workhorse" that facilitates the training of RNNs. BPTT provides an inexpensive means by which to calculate the gradients of the weights and biases with respect to the loss. These gradient matrices can then be used to update the weights and biases according to an optimization process known as gradient descent, formulated in Equation 4.13 (Kamath, J. Liu, and Whitaker, 2019):

$$\theta_{t+1} = \theta_t - \lambda \frac{\partial \mathcal{L}}{\partial \theta} \tag{4.13}$$

Equation 4.13 considers the generic case for a matrix of parameters θ, though in practice it will be applied to each of the weight and bias parameters in the RNN: W_{hh}, W_{hx}, W_{yh}, b_h, and b_y.

The λ term is known as the learning rate and governs how large each update step is. The purpose of Equation 4.13 is to change the weights in the opposite direction to that of the gradient in an effort to yield a configuration of weights that produces a low value for the loss. $\frac{\partial \mathcal{L}}{\partial \theta}$ is the key term derived from BPTT.

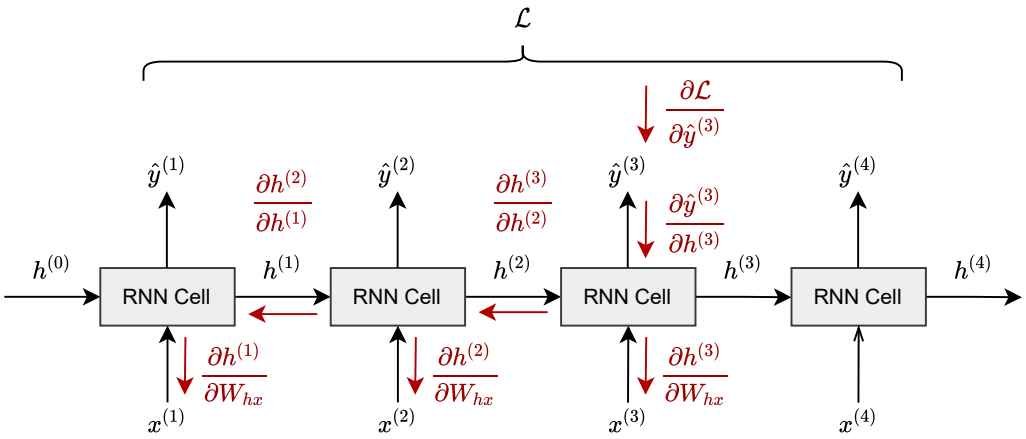

Figure 4.6: Back-propagation through time in an RNN using timestep 3 and weight matrix W_{hx} as an example.

As is evident in Figure 4.6, back-propagation is simply a practical application of the chain rule. Unlike feed-forward neural networks where weights are applied at a single layer, RNNs apply their weights recursively. For this reason, when calculating the gradient of W_{hx}, it is clear that W_{hx} is present not only at timestep three, but at timesteps two and one as well. Back-propagation must therefore be applied "through time".

Continuing the example put forward in Figure 4.6, the process starts by calculating the derivative of the CTC loss, $\mathcal{L}$, with respect to the output probability, $\hat{y}^{(3)}$. Calculating the derivative of the output probability with respect to the hidden state is actually a two step process because of the presence of an activation function:

$$\frac{\partial \hat{y}^{(3)}}{\partial h^{(3)}} = \frac{\partial \hat{y}^{(3)}}{\partial q^{(3)}} \frac{\partial q^{(3)}}{\partial h^{(3)}} \tag{4.14}$$

Where $q^{(3)}$ is the logit prior to the softmax output activation function. The same is true of the derivative of the hidden state, $h^{(3)}$, with respect to the weight matrix, W_{hx}:

$$\frac{\partial h^{(3)}}{\partial W_{hx}} = \frac{\partial h^{(3)}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W_{(hx)}} \tag{4.15}$$

Where $z^{(3)}$ is the matrix prior to the activation function. For simplicity, the derivatives of the activation functions remain implicit in Equation 4.16, which provides an example of how BPTT would be applied to the example in Figure 4.6.

$$\frac{\partial \mathcal{L}}{\partial W_{hx}}^{(3)} = \frac{\partial \mathcal{L}}{\partial \hat{y}^{(3)}} \frac{\partial \hat{y}^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial W_{hx}} + \frac{\partial \mathcal{L}}{\partial \hat{y}^{(3)}} \frac{\partial \hat{y}^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial W_{hx}} + \frac{\partial \mathcal{L}}{\partial \hat{y}^{(3)}} \frac{\partial \hat{y}^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial W_{hx}} \tag{4.16}$$

This calculation has resulted in the gradient of W_{hx} with respect to the loss function for a single timestep. The gradients for each timestep can be accumulated by adding them together, producing $\frac{\partial \mathcal{L}}{\partial W_{hx}}$, thereby completing the process of BPTT. The remaining gradient matrices can be calculated in a similar manner, thus producing $\frac{\partial \mathcal{L}}{\partial W_{hh}}$, $\frac{\partial \mathcal{L}}{\partial W_{yh}}$, $\frac{\partial \mathcal{L}}{\partial b_h}$, and $\frac{\partial \mathcal{L}}{\partial b_y}$. Once all of these gradients have been calculated, gradient descent can be applied as per Equation 4.13.

Minibatch gradient descent

Gradient descent can utilize every example in the dataset to estimate the gradient in what is known as batch gradient descent (Goodfellow, Bengio, and Courville, 2016). This ensures that the estimate is accurate and the optimization is deterministic, but it is usually not feasible due to constraints on placing the entire dataset in memory.

A more commonly used alternative is minibatch gradient descent. Minibatch gradient descent divides the dataset into random groups of examples, each of which is used to estimate the gradient (Goodfellow, Bengio, and Courville, 2016). This both reasonably approximates the gradient and fits into memory, allowing for faster updates. Minibatch sizes are usually a multiple of two due to the run-time improvement this provides on GPUs.

4.2 Extensions to the basic RNN setup

Section 4.1 introduced all of the fundamentals of building and training an RNN. This section will first introduce Adam, a more advanced optimizer that provides some advantages over gradient descent. Insight will then be given regarding two architectural extensions which have not yet been addressed: bidirectional RNNs and encoder-decoder RNNs.

4.2.1 Adam optimizer

Gradient descent, which was introduced in Section 4.1.6, is a means by which to minimize a non-convex[22] error function over a high-dimensional space. While gradient descent is an established way in which to optimize neural networks, certain modifications have been shown to boost its performance.

The main drawback of gradient descent is that it is susceptible to premature convergence. It was previously thought that this was due to the presence of local minima, but Dauphin et al. (2014) argue that saddle points are far more of a problem than local minima. Saddle points are plateaus that occur at a high value for the loss and can cause learning to slow substantially.

To tackle the problem of saddle points, Kingma and Ba (2015) introduced Adam, a computationally efficient modification to gradient descent. Adam effectively amalgamates two other optimizers: momentum and RMSprop. Momentum utilizes exponentially weighted averages to accumulate "momentum", represented by $V_{d\theta}$ in Equation 4.17, such that the rate at which gradient descent can slide off plateaus is increased.

$$V_{d\theta} = \beta_1 V_{d\theta} + (1 - \beta_1) \frac{\partial \mathcal{L}}{\partial \theta} \tag{4.17}$$

[22] A function is non-convex if a line segment between any two points on the graph can lie below the graph.

If the hyper-parameter β_1 is large, then the average will take into account more prior timesteps. In addition to building up momentum in useful directions, momentum also suppresses oscillations and leads to a more direct approach towards minima. Another technique which also seeks to dampen oscillations is known as RMSprop (Equation 4.18).

$$S_{d\theta} = \beta_2 S_{d\theta} + (1 - \beta_2)\frac{\partial \mathcal{L}}{\partial \theta}^2 \tag{4.18}$$

If there are severe oscillations in a certain direction, then $S_{d\theta}$ will be a large value because of the squared term in Equation 4.18. The actual update in that direction will be small, though, because the gradients in Equation 4.19 – which depicts the Adam optimizer – will be divided by a large value.

$$\theta_{t+1} = \theta_t - \lambda \frac{V_{d\theta}}{\sqrt{S_{d\theta}}} \tag{4.19}$$

It is clear that Equation 4.19 is inspired by the standard gradient descent update equation (Equation 4.13), though it includes the terms for momentum and RMSprop which assist in overcoming plateaus faster and more directly.

4.2.2 Bidirectional RNNs

In the discussion of forward propagation in RNNs (Section 4.1.2), it is clear that information propagates in only one direction – forward in time. Even when incorporating more complicated RNN cells such as long-short term memory (LSTM) (Section 4.3.2), the prediction at the current timestep is still only conditional on previous timesteps – all features occurring at future timesteps are not considered. Bidirectional recurrent neural networks (BRNNs) are a modification that allow future context, as well as past context, to be considered in the current timestep's prediction. Figure 4.7 depicts this modification.

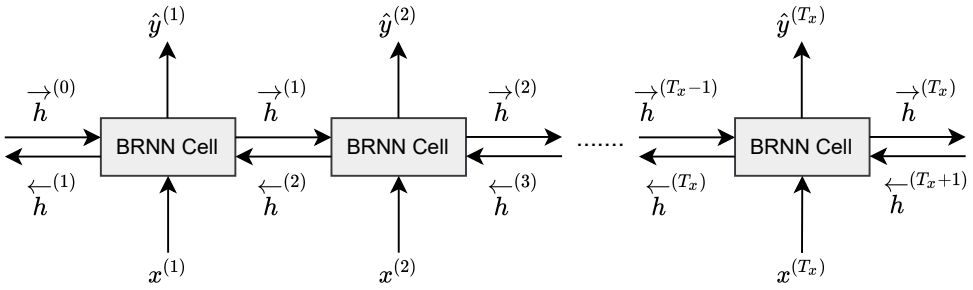

Figure 4.7: Bidirectional RNN.

BRNNs were first introduced by Schuster and Paliwal (1997). They operate by having both a forward pass and a backward pass through the network, running in conjunction with one another. The first pass takes the input feature vectors in chronological order, starting at $x^{(1)}$ and ending at $x^{(T_x)}$, where T_x is the total number of input timesteps. The second pass takes the inputs in reverse order, starting at $x^{(T_x)}$ and ending at $x^{(1)}$. At each timestep two hidden states are produced, $\overrightarrow{h}^{(t)}$ working forward through time, and $\overleftarrow{h}^{(t)}$ working backwards in time. The prediction, $\hat{y}^{(t)}$ is made by concatenating the two relevant hidden states into a single matrix, as depicted in Equation 4.20.

$$\hat{y}^{(t)} = g(W_{yh}[\overrightarrow{h}^{(t-1)}; \overleftarrow{h}^{(t+1)}] + b_y) \tag{4.20}$$

When knowledge of future context is beneficial to a given prediction, BRNNs are often incorporated into the standard RNN architecture. This is the case for tasks such as name-entity

recognition and speech recognition. BRNNs can incorporate both standard RNN blocks, as well as LSTM blocks. One disadvantage of BRNNs is that the full input sequence must be known before predictions can be made, making them unsuitable for real-time applications.

4.2.3 Encoder-decoder RNNs

Section 4.1.4 discussed the use of the CTC loss as a means to perform sequence-to-sequence modelling. The key benefit of CTC is that it allows the input sequence to differ in length to the output sequence, a scenario that is almost always encountered in speech recognition. An alternative approach to using CTC is the encoder-decoder RNN, first introduced by Sutskever, Vinyals, and Le (2014).

Encoder-decoder RNNs function by incorporating two RNNs – an encoder and a decoder. The encoder incorporates the input features and produces a low-dimensional vector representation via the hidden state occurring at the final timestep. The decoder uses this hidden state vector as its initial hidden state and subsequently uses it to produce predictions. More specifically, at each timestep the decoder uses a softmax function to output a probability distribution over characters conditional on the input features through the encoding.

Figure 4.8 illustrates the encoder-decoder setup. The final encoder hidden state, $h_{enc}^{(4)}$, represents the low-dimensional encoding which is passed from the encoder to the decoder. The decoder requires the notion of a start token, $\langle S \rangle$, and an end token, $\langle E \rangle$, which enable the model to define a distribution over sequences of all possible lengths. These tokens are added to the list of characters which can be predicted at any given timestep. The start token initiates the prediction process and is the first input to the decoder in Figure 4.8, $x_{dec}^{(1)}$. The end token is the final prediction, $\hat{y}^{(4)}$, and marks the end of the prediction process.

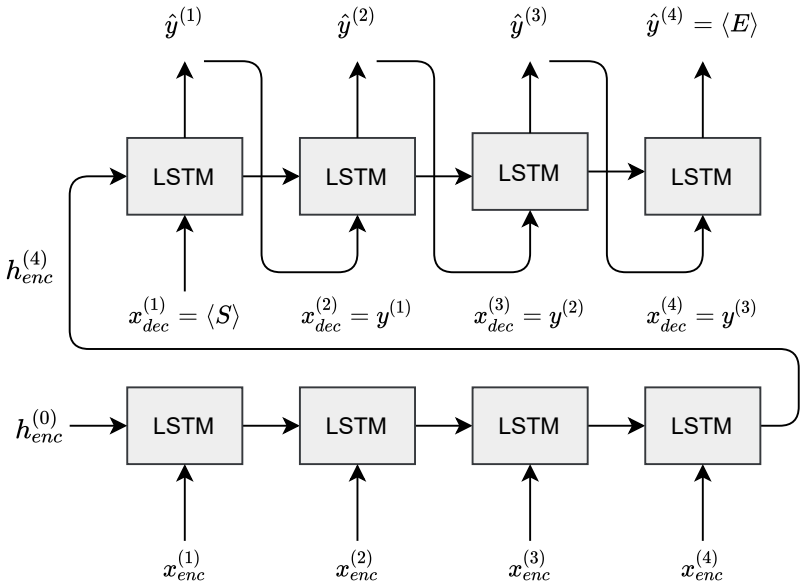

Figure 4.8: Encoder-decoder RNN.

In Figure 4.8, aside from the first input to the decoder, which is the start token, the inputs correspond to the ground truth label from the previous timestep. For example, $x_{dec}^{(2)}$ is equal to $y^{(1)}$, the true label from the previous timestep. This approach to training the encoder-decoder is known as "teacher forcing" and helps the network to learn its own language model (Section 4.5.1). When it comes to inference, the predicted value from the previous timestep is the current timestep's input (rather than the previous timestep's ground truth value). Note that every input to the decoder is one-hot encoded, where the index of the character is 1 and all other indices are 0.

4.3 Vanishing and exploding gradients

As alluded to in the discussion of tanh activation functions in Section 4.1.3, there exists a problem in neural networks known as vanishing and exploding gradients. The saturation of gradients in the tanh activation function can contribute to this, but is only a part of a broader problem.

Three approaches to alleviating the impact of vanishing and exploding gradients will be addressed in the following sections. Section 4.3.1 will address weight initialization, a partial solution to both problems. Long-short term memory, a well-established method to reduce the impact of vanishing gradients and improve the ability of a network to learn long-term dependencies will then be addressed in Section 4.3.2. Finally, a technique to avoid exploding gradients known as gradient clipping will be presented in Section 4.3.3.

4.3.1 Weight initialization

To calculate the gradients of the weights of a deep neural network, back-propagation demands that the weights be multiplied by each other, starting from the final layer and ending with the first layer (Section 4.1.6). If the weights are initialized to very small or very large values, then when multiplied together the gradients will quickly vanish to minute values or explode to enormous values (Bengio, Simard, and Frasconi, 1994). If the gradient of a particular weight is very small, then when passed through the update equation there will be a negligible change in the value of said weight. On the other hand, if the gradient is too large then the updates will oscillate wildly. In both cases, learning becomes infeasible.

Appropriate weight initialization can substantially reduce the impact of vanishing and exploding gradients. One such approach, recommended by Glorot and Bengio (2010), is presented in Equation 4.21.

$$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \tag{4.21}$$

Each weight is drawn from a uniform distribution, with bounds that are determined by the size of the input layer, n. This ensures that the linear combination is close to one, thus when multiplied across layers it neither vanishes nor explodes.

Weight initialization can also determine whether convergence is reached at a high or low loss, and whether convergence is reached at all. It is well established that the initial weights should not be identical values and should rather "break symmetry", allowing each unit to compute a different function (Goodfellow, Bengio, and Courville, 2016). The random initialization of Equation 4.21 achieves this.

4.3.2 Long-short term memory

Although weight initialization can help to minimize the problem of vanishing gradients, the introduction of gates to RNN is a well established approach to further minimizing this problem. Gates allow gradients to propagate backwards through the network, and additionally improve the ability of the network to learn long-term dependencies. The most established approach to incorporating gates is known as long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997).

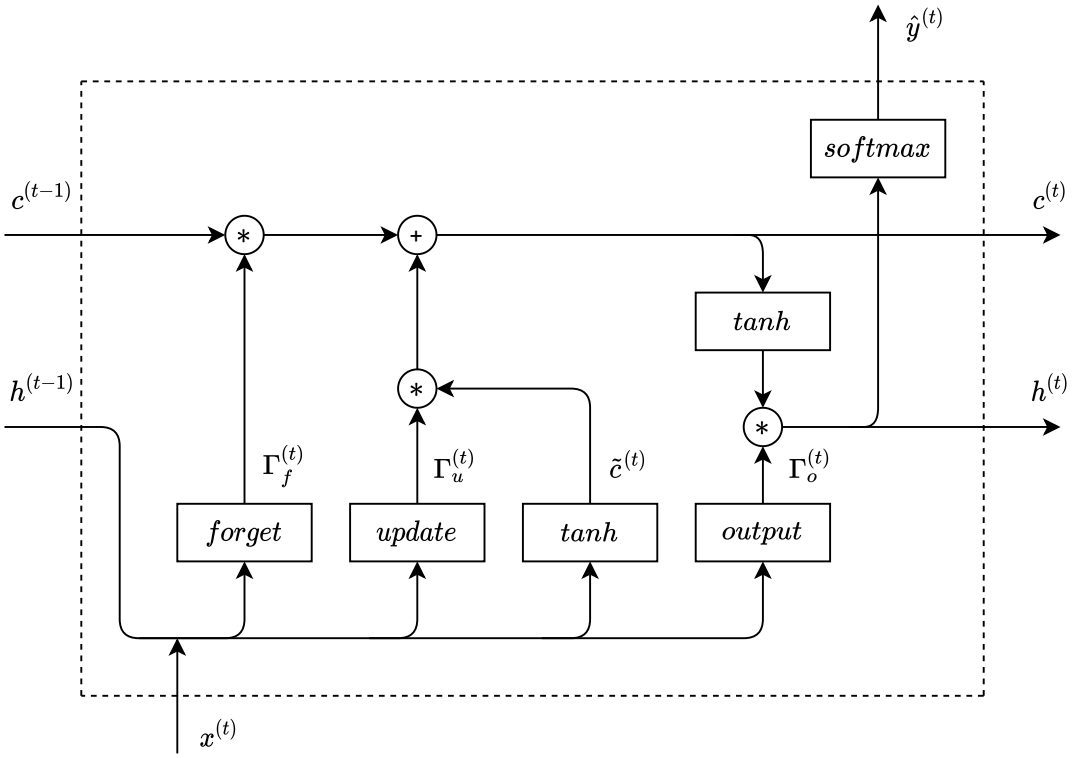

Figure 4.9: Long-short term memory block.

The first step in an LSTM block is to use the current input feature vector, $x^{(t)}$, and the previous hidden state, $h^{(t-1)}$, to calculate three gates: the update gate, forget gate and output gate (Equations 4.22, 4.23, and 4.24).

$$\Gamma_u = \sigma(W_u[h^{(t-1)}; x^{(t)}] + b_u) \tag{4.22}$$

$$\Gamma_f = \sigma(W_f[h^{(t-1)}; x^{(t)}] + b_f) \tag{4.23}$$

$$\Gamma_o = \sigma(W_o[h^{(t-1)}; x^{(t)}] + b_o) \tag{4.24}$$

$[h^{(t-1)}; x^{(t)}]$ is a single matrix obtained by concatenating $x^{(t)}$ and $h^{(t-1)}$. The learnable parameters W_u, W_f, and W_o are used to calculate the update, forget, and output gates. The sigmoid activation, σ, is the activation function typically used to introduce non-linearity. Before these gates can be applied, a candidate cell is calculated as per Equation 4.25 using learnable parameters W_c and b_c.

$$\tilde{c}^{(t)} = \tanh\left(W_c[h^{(t-1)}; x^{(t)}] + b_c\right) \tag{4.25}$$

This candidate cell, $\tilde{c}^{(t)}$, is a candidate at this point because the incorporation of its information into the network is subject to the update gate. If the update gate is close to one then the candidate cell will be extensively incorporated into the cell state. In contrast to the update gate, the forget gate is applied to the previous timestep's cell and has the effect of disregarding information which is no longer necessary to the network. If the forget gate is close to zero then the previous hidden state will largely be discarded from the cell state. The incorporation of information via the update gate and the elimination of information via the forget gate both serve to calculate the current timestep's cell (Equation 4.26).

$$c^{(t)} = \Gamma_u \times \tilde{c}^{(t)} + \Gamma_f \times c^{(t-1)} \tag{4.26}$$

Equation 4.26 is responsible for the improved ability of RNNs to model long-term dependencies and to alleviate vanishing gradients. If the update gate is zero and the forget gate is one, the previous hidden state can be passed through the network indefinitely.

$$h^{(t)} = \Gamma_o \times \tanh\left(c^{(t)}\right) \tag{4.27}$$

The final step in the LSTM block is calculating the hidden state, $h^{(t)}$, using Equation 4.27. The final hidden state is a function of the cell state, $c^{(t)}$, and the output gate, Γ_o. This value can then be passed through a softmax layer to produce probabilistic predictions.

4.3.3 Gradient clipping

Should correct weight initialization still be insufficient to alleviate the affect of exploding gradients, gradient clipping can be employed. If a gradient, ∇, exceeds a given value, then the maximum threshold value, t_{max}, is assigned to that gradient (Kamath, J. Liu, and Whitaker, 2019). If the gradient should be a large negative value, then the minimum value it can assume is t_{min}, where $t_{min} = -t_{max}$.

$$\nabla_{new} = \begin{cases} t_{min} & \text{if } \nabla < t_{min} \\ \nabla & \\ t_{max} & \text{if } \nabla > t_{max} \end{cases} \tag{4.28}$$

4.4 Reducing overfitting

The expressive power of neural networks (Section 4.1.3) is a useful property, though can be detrimental if left unchecked. Overfitting occurs when a model uses its additional degrees of freedom to fit to idiosyncrasies in the data, rather than to the underlying target function (Abu-Mostafa, Magdon-Ismail, and Lin, 2012). Modelling these idiosyncrasies (such as noise) has the effect of leading the model astray and can detrimentally impact its ability to generalize to unseen data.

Using data augmentation to reduce variance, and thus mitigate overfitting, was addressed in Section 3.3. This section introduces two more approaches: regularization and transfer learning.

4.4.1 Regularization

Regularization is a branch of strategies used to counteract overfitting. It serves as a means to inhibit the expressive power of neural networks so that the model might improve its ability to generalize.

Dropout

Dropout, introduced by Hinton, Srivastava, et al. (2012), is one approach to reduce overfitting. It works by randomly omitting connections on each iteration of optimization. Each neuron then learns to detect a feature under a wider variety of internal contexts. This has the effect of preventing complex co-adaptations[23], yielding more robust connections which can generalize more effectively.

[23] When a feature detector is only helpful in the context of several other feature detectors (Hinton, Srivastava, et al., 2012).

Early stopping

Another practical approach to regularization is known as early stopping. Early stopping operates by computing the validation error at the end of every epoch[24]. Rather than training for a predetermined number of epochs, training is halted when the validation error has not improved after a specified number of epochs. The weights that achieved the best performance on the validation set are retained and used during inference (Kamath, J. Liu, and Whitaker, 2019).

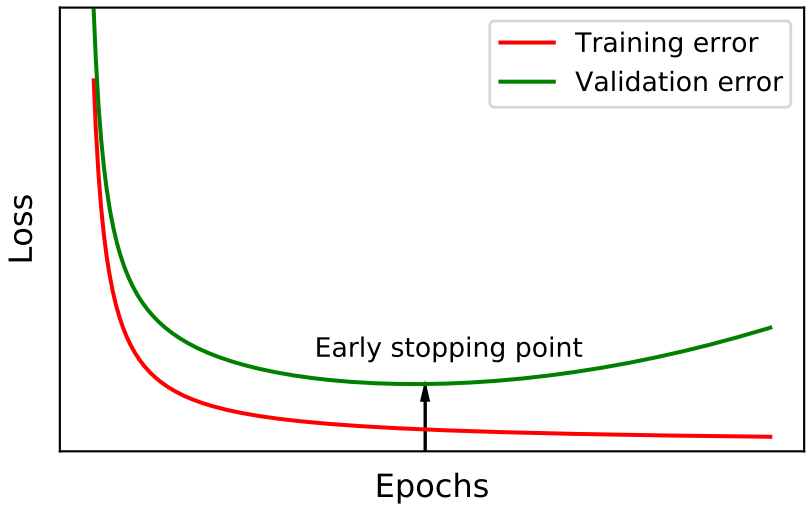

Figure 4.10: Demonstration of early stopping in practice.

The premise is that while the training error will decrease monotonically with the number of epochs, the validation error will eventually plateau and increase due to overfitting. This notion is depicted in Figure 4.10. Early stopping will retain the model which generalizes best before overfitting can have a detrimental impact. It hinges on the assumption that the validation set is of the same distribution as the test set.

4.4.2 Transfer learning

In the field of image recognition, all deep neural networks tend to exhibit similar behaviour: they learn features in the first layer that resemble Gabor filters[25] or colour blobs (Yosinski et al., 2014). This occurs both in different datasets and when applied to different tasks, suggesting that these learned features are general to the problem of image recognition. This is in contrast to specific features that are necessarily dependent on the dataset and task. These specific features occur in the later layers of the network, such as at the output of the network where the classification takes place.

Transfer learning is the process of taking those general features learned on a source dataset, model, and task, and "transferring" them to a target dataset, model, and task (Kamath, J. Liu, and Whitaker, 2019). Harnessing these features can help to improve generalization in the target domain and reduce the computational cost of training a model (Eberhard and Zesch, 2021). It follows that the features that are transferred must be relevant to both the source and the target domains – a model trained on images will transfer to another image recognition task, but not necessarily to a speech recognition task.

Practically, transfer learning is achieved by taking the network weights learned in the source task, and then using them in the target model. These duplicated weights, which hold the learned features, can be used in two ways. They can be frozen and remain unchanged in the

[24] A single pass of the learning algorithm through the entire training set.
[25] A linear filter used for texture analysis.

target model, or they can be fine-tuned (Yosinski et al., 2014). Fine-tuning the weights involves optimizing them using back-propagation, but instead of randomly initializing them, they are initialized using the weights learned from the source dataset.

Transfer learning is often used when the target dataset is much smaller than the source dataset. Doing so allows for the training of large target models without overfitting to the target dataset. The choice of whether to freeze a layer or to fine-tune a layer can come down to the amount of data available (Yosinski et al., 2014). When using large networks to model small datasets, freezing most of the layers may help to avoid overfitting. On the other hand, if a lot of data is available, leaving only the first one or two layers frozen can suffice as the model will be able to learn from the information available in the dataset.

It requires vast computational resources, time and money to develop state-of-the-art models in speech and image recognition. Transfer learning is beneficial in that instead of starting from scratch every time a new problem is faced, information already learned by large, open-source models can be harnessed.

4.5 Decoding predictions

As a reminder, at every output timestep of an RNN a probability distribution over characters and tokens is produced. There still remains the task of choosing the most probable sequence from the outputs. Language models, addressed in Section 4.5.1, can help to rescore the predicted probabilities such that they better reflect English words and sentences. The actual decoding of the predictions through the application of a search algorithm will be discussed in Section 4.5.2.

4.5.1 Language models

Language models provide a means by which to estimate the probability of a sequence of characters or words (Kamath, J. Liu, and Whitaker, 2019). Consider the following two sentences:

"the apple of my I"

"the apple of my eye"

It is clear that the second sentence is the correct form of the idiom. However, in speech recognition systems, an acoustic model might not be able to distinguish this because both "I" and "eye" sound the same. A language model would attribute a much higher probability to the second sentence than the first, thus providing a means by which to choose between the two options.

There are two common approaches to language modelling: n-gram language models and neural language models. Deep Speech makes use of the former, thus n-grams will be explored in more detail here.

Equation 4.29 illustrates how to estimate the probability of a sequence of characters, c, of length L (Kamath, J. Liu, and Whitaker, 2019). The probability of the sequence is obtained by taking the product of every character conditional on the previous n characters, hence the name n-gram language model.

$$\mathbb{P}(c_1, c_2, c_3, ..., c_L) = \prod_{i=1}^{L} \mathbb{P}(c_i | c_{i-n}, ..., c_{i-1}) \tag{4.29}$$

If the current index, i, is less than or equal to the value of n, then the conditional probability will terminate at the first word in the sentence. For example, the probability of the third word using a 4-gram language model will be conditional on the first two words only, $\mathbb{P}(c_3|c_1, c_2)$.

The probability of each character conditional on the previous n characters is estimated by making use of a very large corpus. The number of times the current character, c_i, occurs in the corpus

after the preceding n characters, $c_{i-n}, ..., c_{i-1}$, is divided by the number of times the preceding n characters occur. This is formulated in equation 4.30.

$$\mathbb{P}(c_i|c_{i-n}, ..., c_{i-1}) = \frac{count(c_{i-n}, ..., c_{i-1}, c_i)}{count(c_{i-n}, ..., c_{i-1})} \tag{4.30}$$

For example, using a 4-gram language model one could estimate the probability of "eye" occurring after "the apple of my". This would be achieved by determining the number of times "the apple of my eye" occurs in the corpus, and then dividing that by the number of times "the apple of my" occurs. To estimate the probability of a sentence, this process is applied to each word, and the product is calculated as per Equation 4.29.

4.5.2 Beam search

One approach to finding the best sequence of characters would be to perform a depth-first search and explore every possible combination of output sequences. This would guarantee the arrival at the optimal solution, but it comes at great computational cost. Depth-first search has a time complexity of $\mathcal{O}(b^{T_y})$ (Anari, 2020), where b is the branching factor (which corresponds to the total number of characters in the vocabulary), and T_y is the depth of the graph (corresponding to the length of the output sequence).

On the opposite end of the spectrum, greedy search could be implemented. Greedy search operates by selecting the highest probability at each timestep (or level in the graph), while discarding all other potential pathways. This lead to a favourable time complexity of $\mathcal{O}(T_y b)$, but has the drawback of often arriving at a sub-optimal solution as it ignores most pathways through the graph. Greedy search can be visualized by the tree presented in Figure 4.11

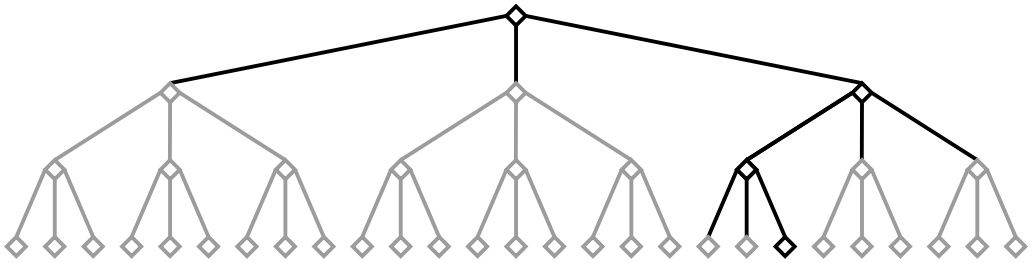

Figure 4.11: Visualization of greedy search.

One approach is to appropriately trade off time complexity and accuracy by using what is known as a beam search (Anari, 2020). Instead of keeping track of only a single pathway, as in greedy search, the K top candidate pathways are stored at any given timestep. K is known as the beam width and it is the parameter which dictates the trade-off. A larger K will be more computationally expensive, but a more optimal solution will likely be obtained.

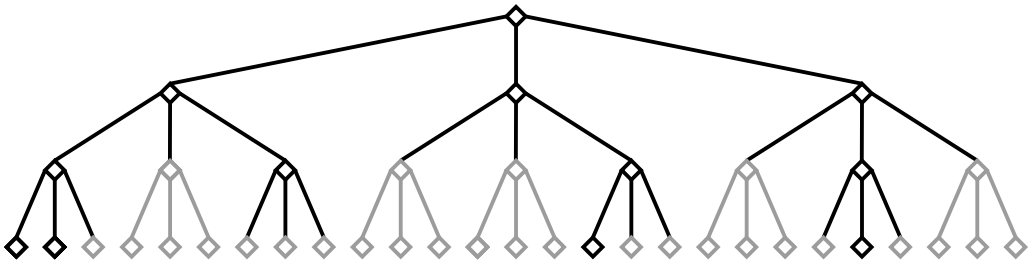

Figure 4.12: Visualization of beam search with $K = 4$, $T_y = 4$, and $b = 3$.

Figure 4.12 visually presents an example of the beam search algorithm. At any given level, $K = 4$ nodes are under consideration, represented by the dark shading. The nodes and edges in

grey depict pathways that are outside of the beam width. Beam search has a time-complexity of $\mathcal{O}(T_y bK log K)$ (Anari, 2020), which is considerably more favourable than the exponential growth of depth-first search.

4.6 Deep Speech

While the original Deep Speech model was developed by Hannun, Case, et al. (2014) on behalf of Baidu, Mozilla developed an open source version which can be used freely for research and other industrial applications. Although inspired by the original paper, Mozilla Deep Speech is different in a few subtle ways. Mozilla Deep Speech will will be discussed in more detail here as it is the version used in the implementation in Chapter 5.

Deep Speech is comprised of five layers, the first three of which are non-recurrent fully-connected layers. The input at each timestep is made up of the spectrogram frame, $x^{(t)}$, and the nine frames on either side. These first layers are responsible for extracting features from the input spectrogram and are computed according to Equation 4.31, which is the equation for a feed-forward neural network.

$$h_t^{(l)} = g(W^{(l)} h_t^{(l-1)} + b^{(l)}) \tag{4.31}$$

Where l is the index of the current layer, W is the weight matrix, and b the bias vector. The fourth layer is a bi-directional recurrent layer (Section 4.2.2) which makes use of LSTM cells (Section 4.3.2). The forward and backward recurrences at each timestep are summed together before being passed to the fifth layer, illustrated by Equation 4.32.

$$h_t^{(4)} = \overrightarrow{h}_t + \overleftarrow{h}_t \tag{4.32}$$

The fifth layer is another non-recurrent feed-forward layer. This layer's output is fed to a softmax layer, producing a probability distribution over characters {a, b, c, ..., z, apostrophe, space, blank} at each timestep. The Deep Speech architecture is presented visually in Figure 4.13.

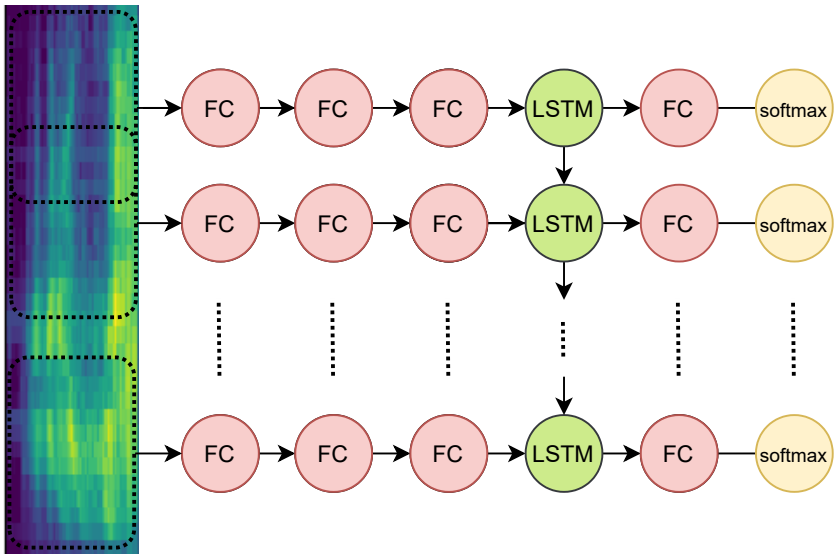

Figure 4.13: Deep Speech architecture. FC stands for fully-connected layer, LSTM stands for long-short term memory cell, and softmax represents the softmax output layer.

Each layer in the network is made up of 2048 neurons. The architecture uses ReLU activation functions in the fully-connected layers, rather than sigmoid or tanh activations, due to the

speed advantages they offer. To handle the alignment problem between input audio and output transcripts, the CTC loss is employed (Section 4.1.4). Regularization is applied during training in the form of dropout (Section 4.4.1) to the feed-forward layers at a rate of 5-10%. The model is optimized using the Adam optimizer (Section 4.2.1).

The architecture described above was trained using open source datasets such TED-LIUM (450 hours) and Librispeech (1000 hours), as well as paid datasets such as Fischer (2000 hours) and Switchboard (300 hours). In an effort to improve performance even further, Mozilla collected over 2000 hours of English speech. This has subsequently been integrated into a dataset known as Common Voice (Mozilla, 2020). All told, it is estimated that the Mozilla Deep Speech training data consists of over 5000 hours of audio.

A key property of Deep Speech is its choice of loss function. One notable characteristic of the CTC loss is that it assumes every output is independent of every other output, conditional on the inputs (Hannun, 2017). Models trained using CTC loss are less efficient at learning language models than alternatives such as encoder-decoder RNNs, which decode outputs dependently. Therefore, an additional language model must be incorporated to fill this role.

In the case of Deep Speech, a 5-gram language model trained on Librispeech normalized training text (Open Speech and Language Resources, 2014) is combined with the acoustic model when decoding. Equation 4.33 shows how the output of the acoustic model, $\mathbb{P}(c|x)$, is combined with the probability of the sentence obtained from the language model, $\mathbb{P}_{lm}(c)$, as well as a penalty term for the length of the sentence.

$$Q(c) = \log\left(\mathbb{P}(c|x)\right) + \alpha \log\left(\mathbb{P}_{lm}(c)\right) + \beta\, word\_count(c) \tag{4.33}$$

α and β are tunable parameters which control the weighting of the language model and the penalty for lengthy predictions, respectively. A beam search (Section 4.5.2) with a beam width of 1024 is employed to identify the most probable output sequence.

4.7 Conclusion

This chapter provided technical descriptions of the methods used in this thesis. The architecture and training procedure of RNNs was addressed in detail, before considering extensions to the basic formulation such as BRNNs. The problems of vanishing and exploding gradients and overfitting were each addressed in their own sections. It was then explained that accurate predictions can be obtained through incorporating language models and applying beam search. The chapter ended with a description of Deep Speech, a model which will be of vital importance in Chapter 5 – the application and results.

Chapter 5

Application and Results

Chapter 3 established the source of dysarthric data and Chapter 4 the technical information necessary to harness Deep Learning. This has provided a platform from which to explore different approaches to improving dysarthric speech recognition. The strategy employed here is to train and compare different models, and in doing so, observe the effect of different factors.

The exact implementation of each model is described in Section 5.1. Section 5.2 presents the results, with a focus on simply noting what was observed. The reasons and possible explanations for these observations are addressed in Section 5.3, while Section 5.4 contextualizes the results further by providing a manual error analysis.

5.1 Implementation

This section will first address practical matters such as the hardware and software used, the approach to splitting the data, and the feature extraction strategy. This will be followed by descriptions of the models, fine-tuning regime, data augmentation regime, and approach to tuning the language model weighting. The section will end by describing the error metrics used to report the results. Appendix C provides a link to the GitHub repository which contains the code used to carry out this implementation.

Hardware and software

The experiments were conducted using an Amazon Web Services (AWS) EC2 p2.xlarge instance. The instance used a Deep Learning Base Amazon Machine Image running Ubuntu 18.04 as its operating system. A Tesla K80 GPU was harnessed in order to speed up training and evaluation. In conjunction with this GPU, CUDA version 10.0 and CUDNN version 7.6.5 were used.

Deep Speech, built upon the TensorFlow programming framework, was used in most of the experiments. The Deep Speech documentation provides directions as to the dependencies that need to be installed in order to train or fine-tune a model (Mozilla, 2020).

Data split

The UASpeech dataset, described in Section 3.1, was the source of dysarthric speech used to train and fine-tune models. As a reminder, each of the 15 speakers contributed speech in three blocks, with each block containing utterances of commands, digits, the radio alphabet, common words and 100 unique uncommon words.

As is the convention with the UASpeech dataset, block 2 of the possible three blocks was used as the test set. This comprised of 3821 distinct utterances. The remaining two blocks of speech were randomly split into training and validation sets, with 85% going into the training set and 15% into the validation set. This lead to a total of 6472 and 1143 training and validation examples, respectively.

Feature extraction

All raw waveforms were transformed to Mel-frequency cepstral coefficients (MFCCs), as per Section 3.2. The selected hyper-parameters are displayed in Table 5.1 and correspond to those used in the original Deep Speech model. The frame length and hop length provide a reasonable trade-off between time granularity and frequency granularity.

Table 5.1: Feature extraction hyper-parameters.

Hyper-parameter	Value
Frame length	32 ms
Hop length	20 ms
MFCC coefficients	26

Model descriptions

Deep Speech and five other models based on Deep Speech were investigated. These models are described below, where each is given an identifying name which will henceforth be used to refer to them.

- `deep-speech`: Mozilla Deep Speech, described in detail in Section 4.6, was used as the industrial-grade speech recognition technology in the experiments that follow. This decision is justified in that it is both open-source and, at the time of its inception, state-of-the-art.

  Version 0.9.3 of Deep Speech was used. This model consists of 2048 neurons in each layer, leading to tens of millions of weights in the entire network. Deep Speech is optimized for standard speech and no alteration of the weights and biases takes place.

  Decoding is performed using the recommended values for α and β – 0.93 and 1.18. All subsequent models undergo fine-tuning of the language model weighting, α, to dysarthric speech. A beam width of 1024 is applied when searching for the best prediction.

- `fine-tune`: this model undergoes fine-tuning of Deep Speech to the dysarthric speech training set. The entire graph is fine-tuned and all weights and biases are available for adaptation to dysarthric speech.

- `fine-tune-aug`: the same as `fine-tune`, except that the dysarthric training dataset is augmented according to the augmentation regime described later in this section.

- `freeze`: the first three fully-connected layers in the network are frozen, while the recurrent layer and all subsequent layers are available for adaptation to dysarthric speech. This roughly corresponds to fixing the feature extraction layers during the fine-tuning process.

- `freeze-aug`: the same as `freeze`, except that the dysarthric training dataset is augmented.

- `re-init`: all weights in the model are randomly initialized and training commences from scratch. Anything learned from the original Deep Speech model is disregarded, yet the same architecture is used.

In addition to these models based on Deep Speech, another custom model was investigated:

- `encoder-decoder`: an encoder-decoder recurrent neural network (RNN) as described in Section 4.2.3. Teacher forcing is used during training and greedy search (Section 4.5.2) is used during inference. Both the encoder and decoder have a hidden dimension of 128, leading to a total of 164 000 weights in the network.

  This model provides an opportunity for a different architecture with much fewer weights to be represented in the experiments.

Data augmentation regime

Time and frequency masking, the augmentation strategies described in Section 3.3, were applied to `fine-tune-aug` and `freeze-aug`. Both types of augmentation were applied to every example with a probability of 1.0. It was decided that a single mask of each be applied to every example. This choice stems from the original SpecAugment paper, where the basic augmentation strategy adhered to this policy (Park et al., 2019). Therefore, the total number of utterances in the training set was increased threefold.

The frequency masks were applied such that between three and five frequency bands were masked. The time masks were applied in a similar manner, with masks ranging from between 40 to 80 ms. The choice of these ranges was inspired by the suggested values in the Deep Speech documentation (Mozilla, 2020). The specific sizes of the masks applied to an example are drawn from a discrete uniform distributions over the stipulated ranges.

Fine-tuning regime

Due to the cost of renting an AWS instance, a large grid search over different hyper-parameters could not be employed. Such a grid-search would require the training of each model dozens of times to determine the best configuration of hyper-parameters. This would require many days of training and would accrue substantial fees. Instead, a standard set of hyper-parameters was selected and used in the optimization of all models. These hyper-parameters are shown in Table 5.2.

Table 5.2: Choice of model hyper-parameters.

Hyper-parameter	Value
Initial learning rate	0.00001
Delta threshold	0.05
Reduce learning rate epochs	10
Plateau reduction factor	0.5
Early stopping epochs	20
Train batch size	32
Validation batch size	16
Dropout	0.05

An initial learning rate of 0.00001 was used. At this learning rate the training loss decreases monotonically, though at larger values large oscillations were observed. In an effort to decrease the loss at later stages of training, reduction of the learning rate on plateaus was enforced. The learning rate was reduced by half every 10 consecutive epochs whereby the Connectionist Temporal Classification (CTC) loss did not decrease more than the absolute delta threshold of 0.05.

It was observed that a training batch size of 32 and validation batch size of 16 were capable of fitting into memory without triggering errors. Dropout (Section 4.4.1) was used at a rate of 0.05 to help improve robustness and reduce co-adaptation of neurons (Hinton, Srivastava, et al., 2012). In an effort to reduce overfitting, early stopping (Section 4.4.1) was triggered after a total of 20 epochs without a reduction in the loss by the absolute delta threshold. The best validating model was saved.

Tuning the language model

Given that the fine-tuning process alters the acoustic model to better predict dysarthric speech, the language model does not necessarily require the same weighting as with standard speech. To account for this, a small grid search of language model weightings was performed for each model.

The trained acoustic models were combined with the Deep Speech language model and evaluated

on the validation set. Values of α investigated range from 0 to 1 by increments of 0.25. The weighting with the best validation word error rate (WER) was used in the final model.

`encoder-decoder` was the only model not to incorporate a language model in the analysis. A key property of the CTC loss is that every output is conditionally independent of every other output given the input (Hannun, 2017). In contrast, encoder-decoder models decode predictions conditional on previous predictions. This provides an opportunity to compare the inherent language modelling abilities of encoder-decoder models with the conditionally independent CTC predictions.

While the language model weighting (α) was tuned, the word count weighting (β) was not. The fine-tuning process was shown to adapt the predictions made by Deep Speech to an isolated word recognition task, thus it was not deemed necessary to fine-tune this hyper-parameter. This observation is discussed in more detail in section 5.4.

Error metrics

The WER (defined in Section 4.1.5) will be the primary means of comparing the performance of different models within this thesis. The use of the WER will also facilitate comparisons with other models in the literature as it is the most commonly used error metric in speech recognition. The character error rate (CER) was also reported as it can provide additional insight into the results due to its increased granularity.

It is desirable for the WER and CER to be low as it indicates that the predicted transcription is close to that of the true transcription. A high WER or CER indicates that the prediction contains many insertions, substitutions and deletions relative to the true transcription. As a reminder, the WER can exceed 100% when the number of insertions, deletions and substitutions is greater than the length of the target sequence.

5.2 Results

This section will begin by assessing the convergence of the models and will briefly present the results of the grid search performed over the language model weighting. Section 5.2.2 will then report on the performance of each model. The section will conclude by presenting on additional insights in Section 5.2.3.

5.2.1 Assessing convergence and tuning the language model

Figure 5.1 shows the training and validation loss for each of the Deep Speech based models, excluding `deep-speech` as it did not undergo any training or fine-tuning. In all cases, acceptable convergence of the validation loss is observed.

It is notable that the validation loss for `fine-tune`, `fine-tune-aug`, `freeze` and `freeze-aug` are almost indistinguishable. This similarity is also observed in the final performance of these models, as will become evident in Section 5.2.2. The CTC loss of `re-init` is considerably higher and much more erratic, something that will also be reflected in the results.

Finally, Figure 5.2 provides the training and validation losses of `encoder-decoder` on a single plot. While the training loss converges to zero, the validation loss improves only marginally from its initialized state.

Figure 5.3 shows the results of the language model fine-tuning process. For `fine-tune`, `fine-tune-aug` and `freeze`, the optimal language model weighting occurred at 0.50, while for the remaining two models 0.25 was the best value.

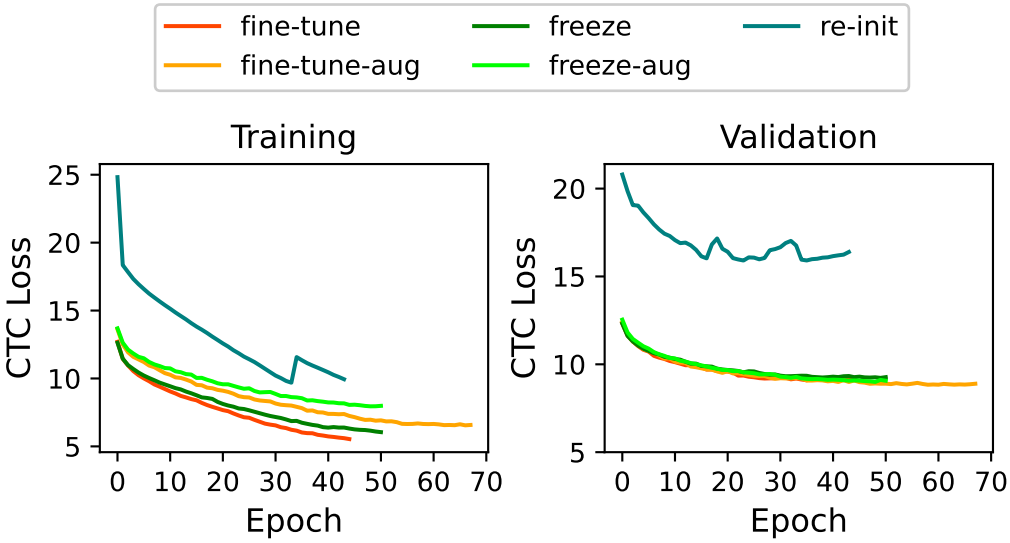

Figure 5.1: Training and validation CTC loss as a function of epoch for the Deep Speech based models.

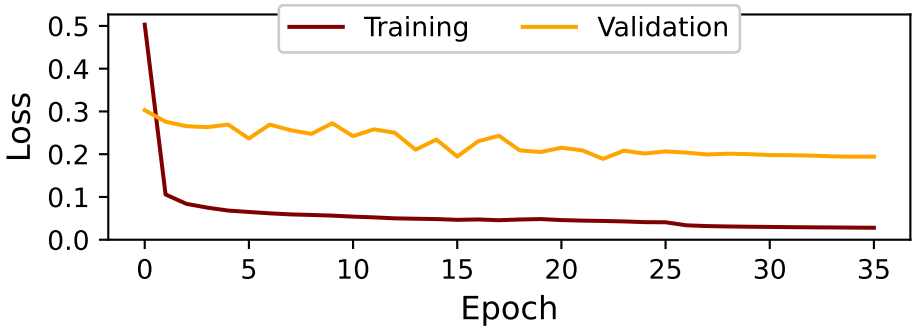

Figure 5.2: Training and validation loss for `encoder-decoder`.

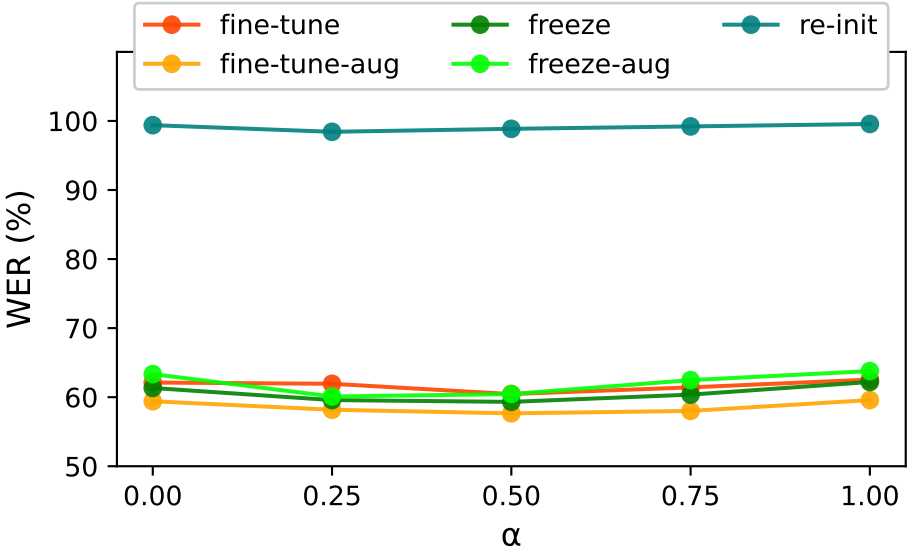

Figure 5.3: Fine-tuning the language model weighting (α) to each model.

5.2.2 Research questions

Table 5.3 forms the crux of the results of this thesis as it provides answers to the five research questions posed in Section 1.6. Those research questions will be reiterated here in order to provide clarity in the analysis.

Table 5.3: Final model performance on the UASpeech test set as measured by the WER and CER, with and without a language model (LM).

Model	No LM		With LM	
	WER (%)	CER (%)	WER (%)	CER (%)
`deep-speech`	141.53	75.27	110.36	68.67
`fine-tune`	70.30	47.52	63.02	48.50
`fine-tune-aug`	68.99	46.21	60.72	46.58
`freeze`	71.19	48.68	63.81	49.27
`freeze-aug`	71.00	48.59	63.57	48.26
`re-init`	99.16	85.13	98.40	88.71
`encoder-decoder`	93.56	89.13	-	-

When referring to the performance of a model in the following discussion, it is assumed that the acoustic model (No LM) is being referred to. It will be explicitly mentioned when the performance with a language model (With LM) is being considered.

Research Question 1: *How does an industrial grade speech recognition model trained on thousands of hours of standard speech perform on dysarthric speech?*

Table 5.3 shows that `deep-speech` attains a WER of 141.53%, which decreases to 110.36% when a language model is incorporated. This result is comparable to what was observed by De Russis and Corno (2019), where three industrial speech recognition systems attained WERs between 80-90% on severe dysarthria.

To ascertain the effect of data mismatch between standard speech and dysarthric speech, `deep-speech` was also evaluated on block 2 of the control data. The WER roughly halves when applied to standard speech in both the case when a language model is included, and when it is excluded. In the latter case a reduction from 141.53% to 68.27% is observed, while in the former it drops from 110.36% to 56.71%.

Research Question 2: *Can the performance of this industrial grade model be improved by fine-tuning to dysarthric speech?*

`fine-tune` improves upon, and more than halves, the WER of `deep-speech`. A reduction from 141.53% to 70.30% is observed, which is comparable to the performance of `deep-speech` on standard speech. This shows a difference in WER of 71.23% and demonstrates fine-tuning can improve performance on dysarthric speech for a model initially trained on standard speech.

Research Question 3: *Does freezing the feature extraction layers during fine-tuning impact performance?*

There is some indication that freezing layers leads to a slightly worse performance than fine-tuning all weights in the network, though the difference is minor: the WER of `freeze` is 0.89% greater than that of `fine-tune`, and the WER of `freeze-aug` is 2.01% greater than that of `fine-tune-aug`.

Research Question 4: *Does augmenting the impaired speech dataset improve the performance of fine-tuning?*

While the improvement observed from fine-tuning is significant, the benefit of data augmentation is far more subtle: `fine-tune-aug` obtains a WER of 1.31% lower than `fine-tune`. The effect

of data augmentation was even smaller on the models which contained frozen layers, where `freeze-aug` improves on `freeze` by a mere 0.19%.

Research Question 5: *How do models trained only on impaired speech compare to the fine-tuned models?*

`re-init` and `encoder-decoder` obtained WERs of 99.16% and 93.56%. These errors are worse than what was seen for the fine-tuned models, though they are better than that of `deep-speech`. However, on closer inspection, the CERs that these two models scored – 85.13% and 89.13% – are higher than the 75.27% achieved by `deep-speech`. This demonstrates that when viewed from the character level, these two models are highly inaccurate. This observation will become more clear in the manual error analysis presented in Section 5.4.

5.2.3 Additional insights

According to Table 5.3, incorporating the language model leads to a marked improvement in performance. The largest improvement is observed for `deep-speech` which sees a reduction in WER by 31.17%. This is similar to the improvement of 21.7% observed by Hannun, Maas, et al. (2014) when incorporating a bigram language model on top of their CTC based acoustic model. The gains from incorporating a language model into the fine-tuned models is less substantial, but still reduces the WER by between 7.28% and 8.27%.

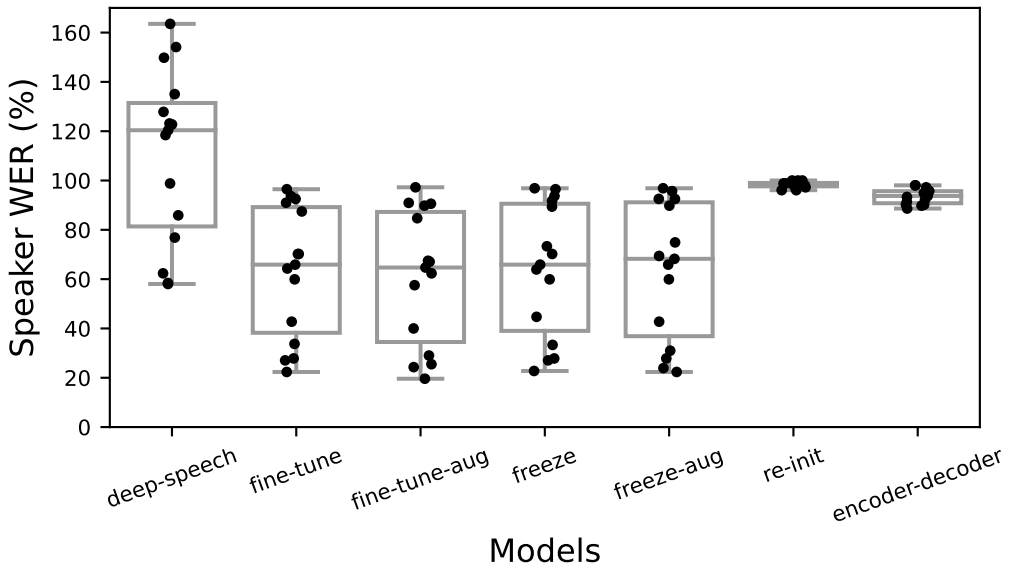

Figure 5.4: Box-plots showing the distribution of speaker WERs for each model. Results are reported with the use of a language model. The individual performance of each dysarthric speaker is represented by the black markers.

Figure 5.4 shows the spread of WERs obtained by the 15 speakers on each model. `deep-speech` ranges from 58% to 164%, a total difference of 106% between the best and worst performing speakers. The range is reduced to 74% for `fine-tune`, with the lowest and highest values occurring at 22% to 96%, respectively. The spread of the three other fine-tuned models are similar to that of `fine-tune`. In contrast, `re-init` and `encoder-decoder` are both clustered at very high WERs, close to 100%.

As shown in Figure 5.5, `deep-speech` performs fairly poorly across the lower three intelligibility levels, obtaining 132%, 137% and 124% WERs, respectively. An improvement is observed at high intelligibility, where a WER of 68% is attained – close to the performance of `deep-speech` on standard speech.

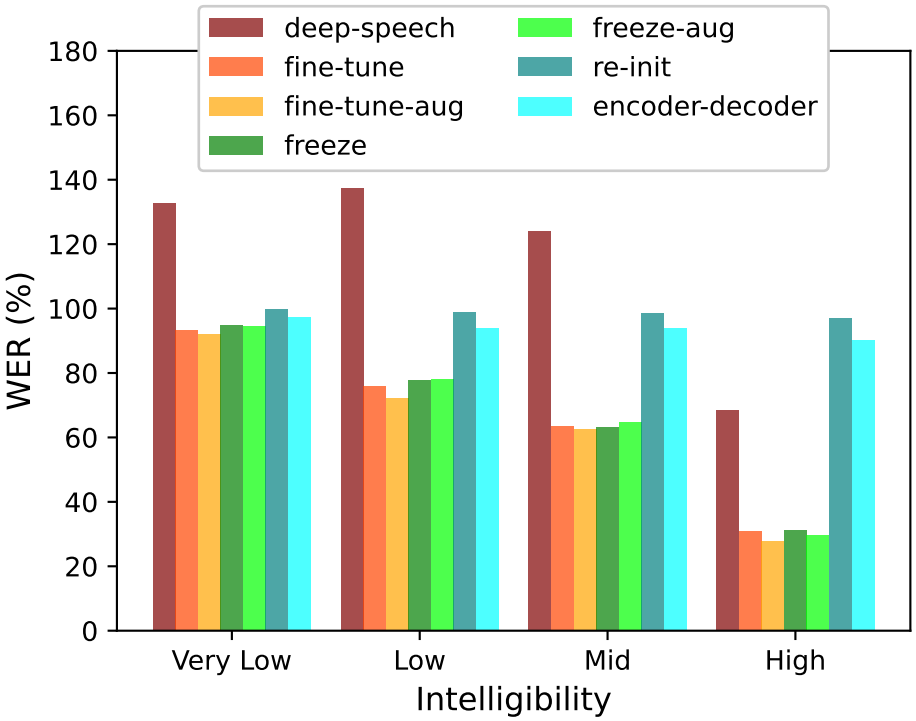

Figure 5.5: WERs grouped by intelligibility for each model. Results are reported with the use of a language model.

The four fine-tuned models outperform `deep-speech` across all levels of intelligibility, and the WERs monotonically decrease with increasing intelligibility. Using `fine-tune` as a reference point, WERs of 93%, 76%, 63% and 31% are attained for very low, low, mid, and high intelligibility. As with Figure 5.4, the performance of `re-init` is close to 100% across all levels of intelligibility, with `encoder-decoder` performing only slightly better.

5.3 Discussion

This section seeks to build on the results presented in Section 5.2 by providing both context and possible explanations for what was observed.

Contextualizing the WERs

While a single number evaluation metric can be useful to compare models, it does not necessarily tell the full story regarding a model's performance. This is particularly true regarding the WER, which is the evaluation metric of choice throughout the majority of speech recognition literature.

Consider that `fine-tune-aug` predicts "saentence" for the underlying target "sentence" in Table 5.4 (which will be scrutinized in Section 5.4). Although the prediction is very close to the true word, it results in a WER of 100% for that particular example. Intuitively, this is punitive as even though it is incorrect, much of the meaning is still conveyed.

Another consideration is that while the goal is to reach a WER of close to zero, this is beyond the capabilities of current speech recognition technologies. It is an immense challenge to create a model which can generalize to different speakers and environments, let alone with the additional challenges of modelling dysarthric speech discussed in Section 1.3.

To provide context for what constitutes a good WER, consider Deep Speech. At the time of its publication, Deep Speech achieved state-of-the-art performance on the Switchboard Hub5'00 test set by scoring a WER of 16.0% with a language model (Hannun, Case, et al., 2014). In

another paper, a bi-directional recurrent deep neural network trained with the CTC loss achieved a WER of 14.1% when evaluated on a standard speech test set with a language model (Hannun, Maas, et al., 2014). This increased to 35.8% when only the acoustic model was used.

Therefore, while the WERs obtained are high, they must be viewed with the appropriate context. Speech recognition is a challenging task where good WERs for standard speech commonly range from 10-20%. In addition, predictions can still convey meaning even if high WERs are obtained.

Performance of Deep Speech on control data

As already mentioned, Deep Speech scored a WER of 16.0% on the Switchboard Hub5'00 test set. However, when applied to the UASpeech control data comprised of standard speech, a WER of 56.71% is obtained. The discrepancy of approximately 40% begs the question – why does Deep Speech perform so much worse than expected?

One reason that could contribute to this discrepancy is that Deep Speech is a large-vocabulary continuous speech recognition system, yet it is being applied to an isolated word recognition problem. For example, it is often easier to discern a word when it is in the context of a sentence. Consider "two" vs "to", where in an isolated context it can be difficult to discern one from another as they sound the same. When these words are inserted into a sentence, the problem becomes much easier as the context in the sentence can help to identify whether a numeral or a preposition is appropriate.

Another potential reason that could contribute to the drop in performance is the manner of pronunciation of an isolated word as compared to a sentence of words. Deep Speech is trained on both read and conversational speech to provide an array of different speaking styles in the training data. However, it could be that speakers pronounce isolated words differently to a string of words in a sentence, thus a data mismatch problem would arise.

Contextualising the differences between models

Because optimizing the weights of a neural network is a non-convex optimization problem, there is no guarantee that minibatch gradient descent will converge on the global minimum. In high-dimensional spaces, this is further complicated by the problem of saddle points which can slow learning dramatically and prevent further improvement of the model (Dauphin et al., 2014). These complications mean that small differences in performance between models should not be given too much attention – it could be that the optimization process is responsible for such minor discrepancies.

The fact that a set of standard hyper-parameters was used could also lead to minor differences in model performance. The purpose of a hyper-parameter search is to explore the space of hyper-parameters and to give each model the best possible opportunity to perform at its best. While the chosen hyper-parameters were forgiving enough to allow for the validation errors to plateau, it could be that it slightly favours one model over another.

Factors which improved dysarthric speech recognition

Following on from the previous two paragraphs, consider the effect of data augmentation in the results, where `fine-tune-aug` improved upon the WER of `fine-tune` by 1.31%. Although this provides some evidence that this data augmentation regime can improve generalization, it is not a very strong signal. Indeed, this difference was reduced to 0.19% between `freeze-aug` and `freeze`.

The improvements observed by `fine-tune-aug` and `freeze-aug` are less than what was seen using other augmentation strategies on the UASpeech dataset. Geng et al. (2020) found that vocal tract length perturbation, tempo perturbation and speed perturbation lead to an improvement over their baseline model by 2.92%. Adversarial data augmentation implemented by Jin et al. (2021) improved performance by 3.05%. This is suggestive that the particular augmentation regime used here is inferior to other approaches.

While data augmentation lead to a minor improvement, fine-tuning, on the other hand, had a substantial impact. `fine-tune` produced a WER that was 71.23% better than that of `deep-speech`. This substantial improvement was also observed for the other three fine-tuned models. This provides convincing evidence that fine-tuning can radically improve performance on dysarthric speech for a model which was originally trained on standard speech.

When Shor et al. (2019) fine-tuned their base model trained on standard speech to dysarthric speech, an improvement of 39.8% was observed. Though this improvement is less than that seen by `fine-tune`, the initial and final WERs of 59.7% and 20.9% were both lower. This comparison is not perfect though as the standard and impaired speech datasets are both different to those used here.

While fine-tuning from the original Deep Speech model was beneficial, very poor results were achieved by `re-init` and `encoder-decoder`, whose weights were randomly initialized. The scarcity of training data and its inherent noise prevented these models from building up a general framework for recognizing dysarthric speech. This demonstrates that the knowledge learned from standard speech was indeed transferable to dysarthric speech and helped to improve generalization.

Eberhard and Zesch (2021) found that freezing any number of layers lead to an improvement in performance when fine-tuning Deep Speech to both German and Swiss. The largest improvement was observed when the first two layers were frozen while fine-tuning to German, which improved the WER by 19% relative to the model which fine-tuned the entire network.

This phenomenon was not observed when fine-tuning to dysarthric speech. The impact of freezing the first three layers had a slightly detrimental impact on performance, as shown by `freeze`, which performs 0.89% worse than `fine-tune`. As with data augmentation, this difference is evident, but not substantial. This result also corresponds to what was observed by Shor et al. (2019), who found that fine-tuning all layers in the network was the best approach.

Possible explanation for layer freezing having no benefit

Given that the UASpeech dataset is relatively small, one might think that reducing the number of trainable parameters might help to prevent overfitting and improve generalization. The reason that freezing the feature extraction layers failed to improve performance might be that the feature extraction process needed to be modified.

The assumption made when using transfer learning in an image recognition context is that the input distributions are similar between problems, and it is the output distributions that are different. Images are all generally made up of similar input features such as edges and colours, but the goal might be to predict different objects. In the context of image recognition, it therefore makes sense to freeze the early layers and modify the later layers.

This assumption does not hold for transfer learning from standard speech to dysarthric speech as the input distributions are palpably different, with dysarthric speech committing errors such the omission of word-final consonants (Figure 3.4). It follows that freezing the early feature extraction layers might not be beneficial because the features themselves are different between standard speech and dysarthric speech. This could explain why `freeze` performs slightly worse than `fine-tune`.

Comparison with the state-of-the-art

The best performing model, `fine-tune-aug`, harnessed fine-tuning and data augmentation to reduce the WER on the UASpeech test set to 60.72%. Despite the promise of this model, and fine-tuning in general, it is still substantially worse than the state-of-the-art on the UASpeech test set.

The model created by S. Liu et al. (2021) is the best performing to date and achieves a WER of 25.21% on the UASpeech test set, such that the performance of `fine-tune-aug` is worse by

35.51%. In addition to the state-of-the-art model, Geng et al. (2020) and Jin et al. (2021) achieve near state-of-the-art with their systems, achieving WERs of 26.37% and 25.89%, respectively.

While these models each harness techniques that can improve performance by a few percent, such as neural architecture search in the case of S. Liu et al. (2021), and adversarial data augmentation in the case of Jin et al. (2021), they all share a common core architecture. Each of these models uses a hybrid deep neural network hidden Markov model (DNN-HMM) system that harnesses learning hidden unit contributions (LHUC) based speaker adaptive training (SAT). It is thought that this core setup is responsible for the majority of the discrepancy between `fine-tune-aug`, with other peripheral improvement strategies making up the minority of the difference.

Although fine-tuning Deep Speech to dysarthric speech has provided insights into the problem of dysarthric speech recognition, particularly from a Deep Learning point of view, it is not comparable in performance to the state-of-the-art DNN-HMM LHUC-based SAT approach.

5.4 Manual error analysis

To gain deeper insight into the performance of the different models, Table 5.4 shows some of the predictions made by `deep-speech`, `fine-tune-aug`, `re-init` and `encoder-decoder`. Five examples from each category are provided, though more predictions can be viewed in Appendix B[26].

Length of predictions

It is clear from Table 5.4 that while the true underlying word is always of length one, `deep-speech` tends to make multi-word predictions. This is particularly evident for words which have multiple syllables, such as the commands, radio alphabet and uncommon words. Consider, the uncommon word "sharpshooter", which is comprised of three syllables and which `deep-speech` predicts as being three separate words.

The reason that multiple words are predicted is thought to stem from the nature of the dysarthric speech. Impaired speech is more likely to be slower than standard speech and can often contain pauses between syllables. `deep-speech`, which is optimized for standard speech, recognizes those

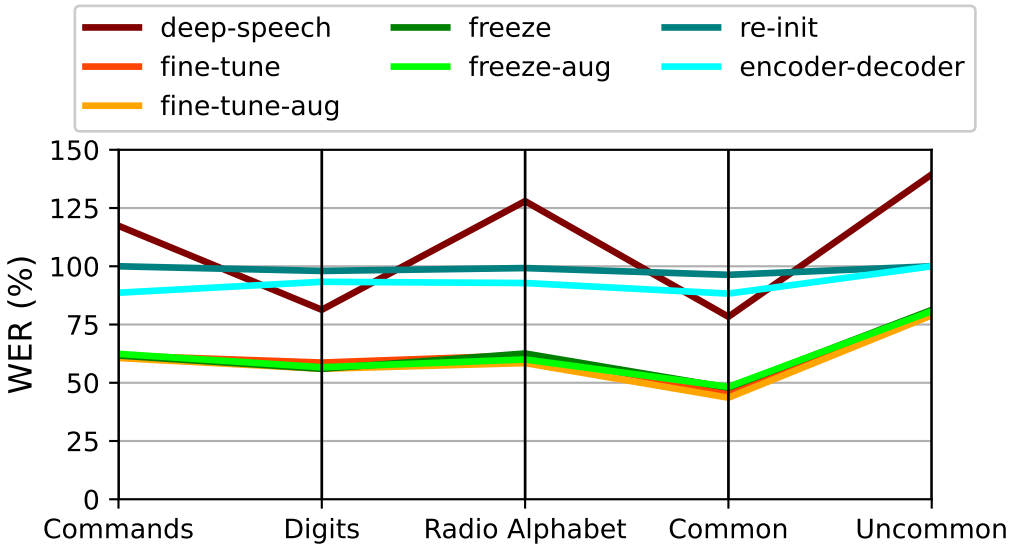

Figure 5.6: WERs grouped by category of word for each model. Results are reported with the use of a language model.

[26] All of the predictions of commands, digits and the radio alphabet for speaker M05 are evident in Table B.1, as well as half the predictions of common and uncommon words in Tables B.2 and B.3, respectively.

pauses as separation between distinct words, rather than as silence within a single word. This same phenomenon was also observed by Moore, Venkateswara, and Panchanathan (2018), where bloated WERs were obtained on dysarthric speech due to excessive word insertions.

Word insertions are less of a problem for the digits and common words, which are generally comprised of a single syllable and thus produce single word predictions. This observation is bolstered by Figure 5.6, which stratifies the results by word category. `deep-speech` performs noticeably better on the digits and common words than it does on the commands, radio alphabet and uncommon words.

The fine-tuned models are effective at learning to ignore such pauses, as illustrated by the predictions of `fine-tune-aug`. In addition to reducing the number of word insertions, `fine-tune-aug` makes predictions that are closer to the true target. Even without a language model, `fine-tune-aug` gets some words correct, such as "charlie" and "zero". It also gets close in other cases, such as "notanber" for "november", or "rigt" for "right". This somewhat subjective improvement in predictions is a manifestation of what was observed in Table 5.3, where the

Table 5.4: Model predictions for speaker M05, a mid intelligibility speaker. All models exclude a language model except for `fine-tune-aug (LM)`.

word	deep-speech	fine-tune-aug	fine-tune-aug (LM)	re-init	encoder-decoder
commands					
command	cal a man	tomman	conan	m	there
right	i	rigt	right	t	was
sentence	sin firs	saentence	sentence	set	saven
downward	thanward	townwat	downward	-	have
alt	aah	alt	at	-	all
digits					
zero	therwo	zero	zero	-	soothing
two	two	to	to	-	to
four	fomer	for	for	-	for
six	ler	sit	sit	s	shout
eight	ana	at	at	-	and
radio alphabet					
whiskey	whes ki	wiskqey	whiskey	st	bashapdieg
charlie	tarly	charlie	charlie	o	copy
tango	hanged tangil	handtango	panago	e	tango
foxtrot	fogs to ar	foxto	fox	-	first
november	no gan blah	notanber	november	me	nowhere
common words					
the	za	tha	the	-	so
of	a oh	of	of	o	out
and	and	and	and	-	than
a	ah	a	a	-	alp
to	two	to	to	o	command
uncommon words					
sharpshooter	sob sad er	sobsupe	soupe	o	southeasterly
advantageous	ia do in haid shis	adurontaegous	advantageous	dis	ashamed
orange	e hn baroh wench	awent	went	t	onver
endowments	leand do ments	andowlments	endowments	no	enter
deluge	the newge	geluce	deuce	h	thee

fine-tuned models outperformed `deep-speech` across the board.

As a final note regarding Figure 5.6, the fine-tuned models perform similarly across the commands, digits, and radio alphabet, yet perform slightly better on common words and slightly worse on uncommon words. The reason for this is likely the nature of these groupings: the complexity of common words is lowest, while uncommon words are more convoluted, thereby providing more opportunity for error.

Effect of the language model

Incorporating the language model into `fine-tune-aug` shows a marked alteration in predictions. While the acoustic model predictions approximate the true word, there are often spelling errors and incorrectly inserted letters. For the most part, all of the predictions made with the inclusion of a language model are correct English words. The acoustic model predictions are rescored such that even if the transcription is incorrect, a valid word is produced.

The language model appears to be most effective when the acoustic model is already almost correct. For example, "andowlments" gets correctly rescored by the language model to "endowments". However, when the acoustic model is far from the correct word, the language model's use is limited. Consider the target word "orange", which the acoustic model predicted to be "awent". This was modified to "went" by the language model, rather than making the substantial leap to "orange".

Although predominantly beneficial, in some instances the addition of the language model is detrimental. Because the language model rescores the acoustic model predictions to better approximate English, it tends to favour common words which are likely to occur. This was observed for the command "alt", which was correctly transcribed by `fine-tune-aug`, but incorrectly modified to "at" when the language model was included.

Re-initialized models: CTC vs encoder-decoder

The predictions made by the `re-init` model are generally either blank, or consist of only a few letters. Looking back at Section 4.1.4, the CTC loss is calculated by taking the target sequence and placing blank labels between every character, as well as at the beginning and the end of the sequence. Therefore, one in two labels in the dynamic programming setup (Figure 4.5) are blank labels. It is thought that the abundance of blank labels in the target leads the model to favour the prediction of blanks over characters.

Bluche et al. (2015) found that handwriting recognition models trained with the CTC loss favour the prediction of blank labels heavily in the early stages of training. Only later in training does the probability of predicting characters increase. While `re-init` failed to successfully move beyond the point of predicting blank tokens, this is not necessarily because training was stopped early. The fact that `encoder-decoder`, the only other randomly initialized model, also performed poorly suggests that the state of the data also works against either model generalizing effectively.

The predictions made by `encoder-decoder` are predominantly real words, and those that are real words can be found in the training set. Despite the benefit of the inherent language modelling capabilities of the encoder-decoder model, it rarely predicts the correct word for a particular target. Consider how `encoder-decoder` predicts "there" for "command", but subsequently predicts "command" when the true label is "to". It is capable of spelling words, but not of predicting them at the appropriate moment.

The inability of `encoder-decoder` to generalize to unseen data is reflected in the fact that the training loss converged on zero, while the validation loss improved only slightly before reaching a plateau (Figure 5.2). The scarcity of training data, as well as its inherent noise, inhibits the ability of data hungry Deep Learning models to build up a robust and generalizable framework from scratch.

5.5 Conclusion

This chapter saw the foundation built by the literature review, data, and methodology chapters come to fruition. The different models selected and trained served to identify differences caused by fine-tuning, layer freezing, and data augmentation. The results and how they relate to the research questions were presented, and a deeper discussion of what was observed took place. A manual error analysis helped to provide more context and highlighted key differences between models.

Chapter 6

Conclusion

6.1 Summary and conclusions

The key idea behind this thesis was to take Deep Speech, a model trained on thousands of hours of standard speech, and to fine-tune it to dysarthric speech in the form of the UASpeech dataset. Deep Learning algorithms are data hungry, hence it was hypothesized that this approach would help to fill the void created by the chronic shortage of dysarthric speech.

The results indicated that fine-tuning to dysarthric speech is effective at improving the performance of a model initially trained on standard speech. While Deep Speech scored a word error rate (WER) of 141.53% on the UASpeech test set without a language model, when fine-tuned it scored 70.30%. This is close to the 68.27% that Deep Speech obtained on the control data comprised only of standard speech.

In addition to investigating fine-tuning, layer freezing and data augmentation were also considered. Freezing the first three fully-connected layers of Deep Speech had a slightly detrimental impact, increasing the WER by 0.89% relative to the fine-tuned model. Data augmentation, applied in the form of time and frequency masking, had a slightly positive impact, decreasing the WER by 1.31%. The small differences in WER show that the effect of these factors is minor.

When the weights of Deep Speech were randomly initialized and training on dysarthric speech commenced from scratch, a WER of 99.16% was obtained. Another model architecture – the encoder-decoder RNN – was also trained from scratch and scored a WER of 93.56%. These poor results are indicative of insufficient training data available to build up a generalizable model.

When a language model was incorporated on top of the base acoustic model, performance consistently improved. Deep Speech received the largest gain of 31.17%, though the fine-tuned models also improved between 7.28% and 8.27%.

A manual error analysis further demonstrated the benefit of incorporating the language model, which is most useful when the prediction that the acoustic model makes is close the true target. It is effective at rescoring the predictions such that minor mistakes are fixed, though is less effective when the acoustic model is far from the correct target.

The manual error analysis also revealed that Deep Speech tends to predict pauses within words as spaces between words, leading to inflated WERs due to an increased number of insertions. The fine-tuned models picked up on this and primarily made only single word predictions. The inherent language modelling capability of the encoder-decoder architecture was also demonstrated in the manual error analysis, especially in contrast to the re-initialized Deep Speech model which favoured the prediction of blank labels.

The best performing model was `fine-tune-aug (LM)`, which scored a WER of 60.72% on the UASpeech test set. This model incorporated a language model and was obtained by fine-tuning Deep Speech to an augmented UASpeech training dataset. Although transfer learning in particular showed promise, the best performing model is considerably distant from the state-of-the-art model currently held by S. Liu et al. (2021), which scores a WER of 25.21%.

This is indicative that taking a Deep Learning model designed for standard speech and transferring it to impaired speech is less effective than certain approaches which focus solely on doing well on impaired speech. It was observed that the state-of-the-art model, as well as two other models close to the state-of-the-art (Geng et al., 2020; Jin et al., 2021), all harness deep neural network hidden Markov model (DNN-HMM) architectures with learning hidden unit contributions (LHUC) based speaker adaptive training (SAT). It is thought that this core architecture is responsible for the majority of the discrepancy between the state-of-the-art and the best model in this thesis.

To the best of the author's knowledge, this work marks the first investigation into fine-tuning Deep Speech to dysarthric speech. Although this is the primary contribution, it has also added insight into the effects of layer freezing and data augmentation to the body of literature. Finally, it demonstrated the difficulty in training Deep Learning based dysarthric speech recognition models from scratch with limited data.

This thesis has attempted to make a small contribution to improving dysarthric speech recognition. There is still much work to be done in creating speech recognition systems that work for all people, including impaired and accented speech. This thesis will conclude by making some suggestions as to where future work could be directed, as well as listing some of the limitations of this work.

6.2 Limitations and future work

The computational limitations placed on training the models meant that a standard set of hyper-parameters was used. While it is not expected that the performance will be substantially improved, it could be worthwhile trying to quantify how much improvement can be gained by performing a large hyper-parameter search as compared to using a standard set of hyper-parameters.

This analysis was limited to the case of isolated words, rather than continuous speech, as this is the nature of the UASpeech dataset. It would be instructive to apply this approach to dysarthric speech in the form of sentences and conversations. Certainly, the effect of the language model is expected to improve given the increased context that words have when in a sentence.

It is thought that due to the high inter-speaker variability in dysarthric speech, speaker-dependent systems could provide performance advantages over speaker-independent systems. This comes at the cost of needing to obtain data from a new speaker before the system can be used. One interesting question that arises is how much data one would need to collect from a new speaker for the speaker-dependent system to be useful.

To answer this question, one would need to collect many hours of speech from a single dysarthric speaker. One could then take a speech recognition system and fine-tune it to that single speaker using a varying amount of data. It is expected that there would be decreasing gains with increasing amounts of data, though the exact rate of decrease is not known.

Following on from this thought, it would also be interesting to see if fine-tuning models to types of dysarthria, rather than to individual speakers, would be viable. Given a new speaker, one could first run the speech through a classifier which would determine the type of dysarthria, before passing the speech through the appropriate model. Theoretically, this could provide a balance between the usability of speaker-independent systems and the performance of speaker-

dependent systems. This does, however, hinge on the assumption that there is low intra-type variability and high inter-type variability.

This thesis took particular care to contextualize the resultant WERs through discussion and manual error analysis. This was necessary as the WER measures the number of insertions, substitutions, and deletions, and not the amount of information a transcription communicates. Although the WER is a useful metric broadly used throughout the literature, future work could investigate other metrics that better capture the information conveyed in a transcription.

As a final suggestion, it could be beneficial to take some of the models that perform best on the UASpeech test set and to quantify how much each component adds to the final result. This thesis tried to estimate how much benefit fine-tuning, data augmentation and layer freezing provide. In a similar vein, it would be useful to know where the biggest gain stems from in the state-of-the-art model, whether it is the DNN-HMM architecture, LHUC-based SAT, the augmentation regime, neural architecture search, or some other component.

Appendix A

Phoneme Dictionary

Table A.1: Phoneme dictionary derived from Bird (2021).

phoneme	example	translation	phoneme	example	translation
AA	odd	AA D	AE	at	AE T
AH	hut	HH AH T	AO	ought	AO T
AW	cow	K AW	AY	hide	HH AY D
B	be	B IY	CH	cheese	CH IY Z
D	dee	D IY	DH	thee	DH IY
EH	Ed	EH D	ER	hurt	HH ER T
EY	ate	EY T	F	fee	F IY
G	green	G R IY N	HH	he	HH IY
IH	it	IH T	IY	eat	IY T
JH	gee	JH IY	K	key	K IY
L	lee	L IY	M	me	M IY
N	knee	N IY	NG	ping	P IH NG
OW	oat	OW T	OY	toy	T OY
P	pee	P IY	R	read	R IY D
S	sea	S IY	SH	she	SH IY
T	tea	T IY	TH	theta	TH EY T AH
UH	hood	HH UH D	UW	two	T UW
V	vee	V IY	W	we	W IY
Y	yield	Y IY L D	Z	zee	Z IY
ZH	seizure	S IY ZH ER	-	-	-

Appendix B

Model Predictions

Table B.1: `fine-tune-aug` predictions of commands, digits and the radio alphabet for speaker M05 (mid intelligibility) with and without a language model.

word	`fine-tune-aug`	`fine-tune-aug (LM)`	**word**	`fine-tune-aug`	`fine-tune-aug (LM)`
commands					
command	tomman	conan	paragraph	powaglac	powala
backspace	foxfaithe	fox faith	sentence	saentence	sentence
delete	belide	believe	paste	paste	paste
enter	enter	enter	cut	cot	to
tab	pob	pop	copy	topcote	toppy
escape	scape	cape	upward	outward	outward
alt	alt	at	downward	townwat	downward
control	control	control	left	left	left
shift	seft	set	right	rigt	right
line	line	mine	-	-	-
digits					
zero	zero	zero	five	five	five
one	wone	one	six	sit	sit
two	to	to	seven	saven	seven
three	suman	some	eight	at	at
four	for	for	nine	line	nine
radio alphabet					
alpha	alpha	alpha	november	notanber	november
bravo	talvo	bravo	oscar	asto	as
charlie	charlie	charlie	papa	pabpa	papa
delta	denter	dante	quebec	qupat	at
echo	exho	to	romeo	romeo	romeo
foxtrot	foxto	fox	sierra	siora	sir
golf	gof	go	tango	handtango	panago
hotel	hotel	hotel	uniform	iniform	uniform
india	inindia	in india	victor	victor	victor
juliet	toliete	juliet	whisky	wiskqey	whiskey
kilo	kalo	lo	xray	xray	cray
lima	nabor	never	yankee	inkkey	ikey
mike	mike	mike	zulu	u	i

Table B.2: `fine-tune-aug` predictions of common words for speaker M05 (mid intelligibility) with and without a language model.

word	`fine-tune-aug`	`fine-tune-aug (LM)`	**word**	`fine-tune-aug`	`fine-tune-aug (LM)`
common words					
the	tha	the	or	or	or
of	of	of	had	had	had
and	and	and	by	butby	by
a	a	a	word	wold	would
to	to	to	but	but	but
in	in	in	not	not	not
is	has	as	what	wat	what
you	you	you	all	ar	a
that	that	that	were	war	we
it	it	it	we	wre	we
he	he	he	when	when	when
was	was	was	your	yar	your
for	far	for	can	coman	conan
on	on	on	said	sied	sit
are	ar	a	there	there	there
as	as	as	use	use	use
with	whith	with	an	and	and
his	has	his	each	each	each
they	they	they	which	wich	which
i	i	i	she	see	she
at	at	at	do	de	de
be	day	day	how	how	how
this	this	this	their	there	there
have	have	have	if	it	it
from	foun	from	will	qel	well

Table B.3: `fine-tune-aug` predictions of uncommon words for speaker M05 (mid intelligibility) with and without a language model.

word	`fine-tune-aug`	`fine-tune-aug (LM)`	**word**	`fine-tune-aug`	`fine-tune-aug (LM)`
uncommon words					
mouth	moute	mouth	although	altho	also
needlepoint	nantopined	antoine	anxieties	anxity	anxiety
nuremberg	noennomboult	noanoa	anybody	anybity	anybody
pennyworth	annywers	anyway	anything	anyting	anything
reunited	wenited	reunited	apothecary	upupscaly	pascal
roof	wofe	of	appreciable	ubpasact	otasite
schoolyard	scolad	scold	apprehend	absrhand	abraham
sharpshooter	sobsupe	soupe	approach	upfot	pot
skyward	cywald	coward	astounded	astandic	astandin
sugar	sirga	serge	atrocious	afosess	exposes
supreme	supeme	supreme	authoritative	asoitap	aetat
toothache	tosate	to sate	aversion	overser	overton
unusual	unuselo	anusia	bachelor	tatler	tatler
voyage	viliete	vile	bathe	bash	bath
abbreviated	upeviated	opiated	baths	fass	fast
ablutions	oparnusence	arsene	battlefield	fidafimd	fateful
absolve	altol	also	battlements	totevent	botherment
absorb	atol	athol	battleship	thaterther	tatter
adhesion	oethen	then	beef	baye	day
adjacent	adeslec	adele	beleaguering	beenein	been
advantageous	adurontaegous	advantageous	bengal	bengo	ben
agricultural	aeagocuto	doctor	bequeath	peperase	pease
allure	ano	at	betook	fattat	fate
aloft	alot	alas	betroth	dattroth	betroth
aloof	adu	odu	blithe	live	live

Appendix C

Code

The code for this thesis can be found at the following GitHub repository:
https://github.com/CharlesRHouston/dysarthria-project

Bibliography

Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin (2012). *Learning From Data*. Pasadena, CA. AMLbook.com.

Amodei, Dario et al. (2015). *Deep Speech 2: End-to-End Speech Recognition in English and Mandarin*. arXiv: 1512.02595 [cs.CL].

Anari, Nima (2020). *CPs: Beam Search*. https://stanford-cs221.github.io/autumn2020-extra/modules/csps/beam-search.pdf. [Online; accessed 21-October-2021].

Baker, J. et al. (2009). "Research developments and directions in speech recognition and understanding". In: *IEEE Signal Processing Magazine* 26.3, pp. 75–80.

Baum, L. E. (1972). "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes". In: *Inequalities* 3, pp. 1–8.

Bengio, Yoshua, P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166.

Bird, Steven (2021). *Natural Language Toolkit: Carnegie Mellon Pronouncing Dictionary Corpus Reader*. URL: https://www.nltk.org/_modules/nltk/corpus/reader/cmudict.html (visited on 11/18/2021).

Bluche, Théodore et al. (2015). "Framewise and CTC training of Neural Networks for handwriting recognition". In: *13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 81–85. DOI: 10.1109/ICDAR.2015.7333730.

Chiu, Chung-Cheng et al. (2018). *State-of-the-art Speech Recognition With Sequence-to-Sequence Models*. arXiv: 1712.01769 [cs.CL].

Cybenko, George (1988). *Continuous valued neural networks with two hidden layers are sufficient*. Tech. rep. University of Illinois at Urbana-Champaign. Center for Supercomputing Research and Development.

– (Dec. 1989). "Approximation by superpositions of a sigmoidal function". In: *Journal Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314.

Darley, Frederic L., Arnold E. Aronson, and Joe R. Brown (1969). "Differential Diagnostic Patterns of Dysarthria". In: *Journal of Speech and Hearing Research* 12, pp. 246–269.

Dauphin, Yann et al. (2014). arXiv: 1406.2572 [cs.LG].

Davis, K. H., R. Biddulph, and S. Balashe (1952). "Automatic Recognition of Spoken Digits". In: *The Journal of the Acoustical Society of America* 24.6, pp. 637–642.

De Russis, L. and F. Corno (2019). "On the impact of dysarthric speech on contemporary ASR cloud platforms". In: *Journal of Reliable Intelligent Environments* 5, pp. 163–172. DOI: https://doi.org/10.1007/s40860-019-00085-y.

Deng, Li and Dong Yu (2014). *Foundations and Trends in Signal Processing*. Vol. 7. 3-4. now.

Dorsey, E. Ray et al. (2018). "The Emerging Evidence of the Parkinson Pandemic". In: *Journal of Parkinson's Disease* 8.s1, S3–S8. DOI: 10.3233/JPD-181474.

Eberhard, Onno and Torsten Zesch (Sept. 2021). "Effects of Layer Freezing on Transferring a Speech Recognition System to Under-resourced Languages". In: *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*. Düsseldorf, Germany: KONVENS 2021 Organizers, pp. 208–212. URL: https://aclanthology.org/2021.konvens-1.19.

Elman, Jeffrey L. (1990). "Finding Structure in Time". In: *Cognitive Science* 14.2, pp. 179–211.

Enderby, Pam (2013). "Disorders of communication: dysarthria". In: *Handbook of Clinical Neurology* 110.3, pp. 273–281.

Ferguson, J. D. (1980). *Hidden Markov Analysis: An Introduction, in Hidden Markov Models for Speech*. Tech. rep. Institute for Defense Analyses.

Fletcher, Harvey (1922). "The Nature of Speech and its Interpretation". In: *Journal of the Franklin Institute*.

Geng, Mengzhe et al. (2020). "Investigation of Data Augmentation Techniques for Disordered Speech Recognition". In: *Proc. Interspeech 2020*, pp. 696–700. DOI: 10.21437/Interspeech.2020-1161.

Glorot, Xavier and Yoshua Bengio (Jan. 2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research - Proceedings Track* 9, pp. 249–256.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Graves, A. et al. (2006). "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets". In: *ICML'06: Proceedings of the 23rd International Conference on Machine Learning*, pp. 369–376.

Hannun, Awni (2017). "Sequence Modeling with CTC". In: *Distill*. https://distill.pub/2017/ctc. DOI: 10.23915/distill.00008.

Hannun, Awni, Carl Case, et al. (2014). *Deep Speech: Scaling up end-to-end speech recognition*. arXiv: 1412.5567 [cs.CL].

Hannun, Awni, Andrew L. Maas, et al. (2014). *First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs*. arXiv: 1408.2873 [cs.CL].

Hawley, Mark S. et al. (2013). "A Voice-Input Voice-Output Communication Aid for People With Severe Speech Impairment". In: *IEEE Transactions on Neural Systems and Rehabilitation Enginering* 21.1, pp. 23–31.

Hebb, Donald Olding (1949). *The organization of behavior: A neuropsychological theory*. Wiley.

Hinton, Geoffrey E., S. Osindero, and Y.-W. Teh (2006). "A fast learning algorithm for deep belief nets". In: *Neural Computation* 18.7, pp. 1527–1554.

Hinton, Geoffrey E., Nitish Srivastava, et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580. arXiv: 1207.0580. URL: http://arxiv.org/abs/1207.0580.

Hochreiter, Sepp and Jurgen Schmidhuber (1997). "Long short-term memory". In: *Neural Computation* 9.8, pp. 1735–1780.

Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities". In: *Proc. of the National Academy of Sciences* 79.8, pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554..

Ioffe, Sergey and Christian Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv: 1502.03167 [cs.LG].

Jelinek, F., L. R. Bahl, and R. L. Mercer (1975). "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech". In: *IEEE Trans. On Information Theory* IT-21, pp. 250–256.

Jin, Zengrui et al. (2021). *Adversarial Data Augmentation for Disordered Speech Recognition*. arXiv: 2108.00899 [eess.AS].

Johnson, Ann (2002). "Prevalence and characteristics of children with cerebral palsy in Europe". In: *Developmental Medicine & Child Neurology* 44, pp. 633–640.

Jordan, M.I. (1986). *Serial order: A parallel distributed processing approach*. Tech. rep. 8604. San Diego: University of California, Institute for Cognitive Science.

Juang, B. H. (1985). "Maximum Likelihood Estimation for Mixture Multivariate Stochastic Observations of Markov Chains". In: *AT&T Tech. J.* 64.6, pp. 1235–1249.

Juang, B. H., S. E. Levinson, and M. M. Sondhi (1986). "Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains". In: *IEEE Trans. Information Theory* It-32.2, pp. 307–309.

Juang, B. H. and Lawrence R. Rabiner (2004). *Automatic Speech Recognition – A Brief History of the Technology Development*. Tech. rep. Rutgers University and the University of California.

Junqua, J.-C. (1993). "The Lombard reflex and its role on human listeners and automatic speech recognizers". In: *Journal of the Acoustical Society of America* 93, pp. 510–524. DOI: https://doi.org/10.1121/1.405631.

Jurafsky, Dan and James Martin (2020). *Speech and Language Processing*. 3rd. Prentice Hall.

Kamath, Uday, John Liu, and James Whitaker (2019). *Deep Learning for NLP and Speech Recognition*. 1st. Gewerbestrasse 11, 6330 Cham, Switzerland: Springer.

Karpathy, Andrej (May 2015). *The Unreasonable Effectiveness of Recurrent Neural Networks*. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Kim, Heejin et al. (2008). "Dysarthric speech database for universal access research". English (US). In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 1741–1744. ISSN: 2308-457X.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980.

Knuijt, Simone et al. (2013). "Dysarthria and dysphagia are highly prevalent among various types of neuromuscular diseases". In: *Disability and Rehabilitation* 36.15, pp. 1285–1289.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* 25.

Lecun, Yann (1985). "Une procedure d'apprentissage pour reseau a seuil asymmetrique (A learning scheme for asymmetric threshold networks)". English (US). In: *Proceedings of Cognitiva 85, Paris, France*, pp. 599–604.

LeCun, Yann, Yoshua Bengio, and Geoffrey E. Hinton (2015). "Deep learning". In: *Nature* 521, pp. 436–444. DOI: doi:10.1038/nature14539.

LeCun, Yann, B. Boser, et al. (1989). "Back-propagation applied to handwritten zip code recognition". In: *Neural Computation* 1.4, pp. 541–551.

Levenshtein, Vladimir I. (1965). "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics. Doklady* 10, pp. 707–710.

Levinson, S. E., Lawrence R. Rabiner, and M. M. Sondhi (1983). "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition". In: *Bell Syst. Tech. J* 62.4, pp. 1035–1074.

Lipton, Zachary Chase (2015). "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019. URL: http://arxiv.org/abs/1506.00019.

Liu, Shansong et al. (2021). "Recent Progress in the CUHK Dysarthric Speech Recognition System". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29, pp. 2267–2281. DOI: 10.1109/TASLP.2021.3091805.

Lowerre, Bruce (1976). "The HARPY Speech Recognition System". PhD thesis. Carnegie-Mellon University.

McCullough, W. S. and W. H. Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *Bull. Math Biophysics* 5, pp. 115–133.

Mengistu, Kinfe and Frank Rudzicz (May 2011). "Adapting acoustic and lexical models to dysarthric speech". In: pp. 4924–4927. DOI: 10.1109/ICASSP.2011.5947460.

Minski, Marvin L. and Seymour A. Papert (1969). *Perceptrons: an introduction to computational geometry*. MIT Press.

Mohamed, A.-R., G. E. Dahl, and Geoffrey E. Hinton (2012). "Acoustic modeling using deep belief networks". In: *IEEE Trans. Audio Speech Lang. Process.* 20, pp. 14–22.

Moore, Meredith, Hemanth Venkateswara, and Sethuraman Panchanathan (2018). "Whistle-blowing ASRs: Evaluating the Need for More Inclusive Speech Recognition Systems". In: *Interspeech*, pp. 466–470. DOI: 10.21437/Interspeech.2018-2391.

Morris, Andrew, Viktoria Maier, and Phil Green (Oct. 2004). "From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition". In: DOI: 10.21437/Interspeech.2004-668.

Mozilla (2020). *Training Your Own Model*. URL: https://deepspeech.readthedocs.io/en/r0.9/TRAINING.html (visited on 10/28/2021).

Murphy, Joan (2004). "'I Prefer Contact This Close': Perceptions of AAC by People with Motor Neurone Disease and their Communication Partners". In: *Augmentative and Alternative Communication* 20.4, pp. 259–271.

Open Speech and Language Resources (2014). *LibriSpeech language models, vocabulary and G2P models*. [Online; accessed 24-November-2021].

Park, Daniel S. et al. (Sept. 2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019*. DOI: 10.21437/interspeech.2019-2680. URL: http://dx.doi.org/10.21437/Interspeech.2019-2680.

Parker, D. B. (1985). *Learning-logic*. Tech. rep. TR-47. Center for Comp. Research in Economics and Management Sci., MIT.

Pew Research Center (2017). *Nearly half of Americans use digital voice assistants, mostly on their smartphones*. URL: https://www.pewresearch.org/fact-tank/2017/12/12/nearly-half-of-americans-use-digital-voice-assistants-mostly-on-their-smartphones/ (visited on 2021).

Prandoni, Paolo and Martin Vetterli (2008). *Signal Processing for Communications*. 1st ed. EPFL Press.

Rabiner, Lawrence R. (1989). "A Tutorial on Hidden Markov Models". In: *IEEE* 77.2, pp. 257–286.

Raina, R., A. Madhavan, and A. Y. Ng (2009). "Large-scale deep unsupervised learning using graphics processors". In: *26th Annual International Conference on Machine Learning*, pp. 873–880.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain". In: *Psychological review* 65.6, p. 386.

Rudzicz, Frank, Aravind Namasivayam, and Talya Wolff (Jan. 2010). "The TORGO database of acoustic and articulatory speech from speakers with dysarthria". In: *Language Resources and Evaluation* 46, pp. 1–19. DOI: 10.1007/s10579-011-9145-0.

Sakai, Toshiyuki and Shuji Doshita (1962). "An Automatic Recognition System of Speech Sounds". In: *Studia Phonologica* 2, pp. 83–95.

Sakoe, Hiroaki and Seiba Chiba (1978). "Dynamic Programming Algorithm Optimization for Spoken Word Recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1, pp. 43–49.

Sapp, B., A. Saxena, and A. Y. Ng (2008). "A fast data collection and augmentation procedure for object recognition". In: *AAAI Twenty-Third Conference on Artificial Intelligence*.

Schafer, Anton Maximilian and Hans-Georg Zimmermann (2007). "Recurrent Neural Networks are Universal Approximators". In: *International Journal of Neural Systems* 17.4, pp. 253–263.

Schuster, M. and K.K. Paliwal (1997). "Bidirectional Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681. DOI: https://doi.org/10.1109/78.650093.

Shannon, C. (1948). "A mathematical theory of communication". In: *Bell System Technical Journal* 27, 379-423 and 623–656.

Shor, Joel et al. (2019). "Personalizing ASR for Dysarthric and Accented Speech with Limited Data". In: *Interspeech 2019*. DOI: 10.21437/interspeech.2019-1427. URL: http://dx.doi.org/10.21437/Interspeech.2019-1427.

Stevens, S. and J. Volkmann (1940). "The Relation of Pitch to Frequency: A Revised Scale". In: *The American Journal of Psychology* 8.3, pp. 329–353. DOI: https://doi.org/10.2307/1417526.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc.

Vachhani, Bhavik, Chitralekha Bhat, and Sunil Kumar Kopparapu (2018). "Data Augmentation using Healthy Speech for Dysarthric Speech Recognition". In: *Interspeech*, pp. 471–475. DOI: 10.21437/Interspeech.2018-1751.

Vintsyuk, T. K. (1968). “Speech Discrimination by Dynamic Programming”. In: *Kibernetika* 4.2, pp. 81–88.

Warden, Pete (2018). *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition.* arXiv: 1804.03209 [cs.CL].

Werbos, Paul (1974). “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.” PhD thesis. Harvard University.

– (1990). “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10, pp. 1550–1560.

Xu, Lu et al. (2019). “Global variation in prevalence and incidence of amyotrophic lateral sclerosis: a systematic review and meta-analysis”. In: *Journal of Neurology* 267.4, pp. 944–953. DOI: 10.1007/s00415-019-09652-y.

Yorkston, Kathryn M. (1996). “Treatment Efficacy: Dysarthria”. In: *Journal of Speech and Hearing Research* 39, S46–S57.

Yosinski, Jason et al. (2014). *How transferable are features in deep neural networks?* arXiv: 1411.1792 [cs.LG].