Winter 12-2016

End-to-End Speech Recognition Models

William Chan
Carnegie Mellon University, wchan212@gmail.com

End-to-End Speech Recognition Models

Submitted in partial fullfillment of the requirements for
the degree of

Doctor of Philosophy
in
Department of Electrical and Computer Engineering

William Chan

BASc Computer Engineering, University of Waterloo
MS Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

October 2016

Acknowledgements

I would like to thank my advisor Professor Ian Lane for the excellent mentorship and the research freedom given throughout my PhD career. I also give my thanks to all my committee members. Professor Bhiksha Ramakrishnan for giving me valuable advice and questions. Professor Chris Dyer thanks for all our Skype chats and insightful discussions. Quoc Le for inviting me to work with you at Google Brain and the excellent mentorship. Thank you guys! Without you all, this thesis would definitely have not been possible!

My family, mama Ann Chan, papa Sam Chan and miumiu Joanna Chan thanks for all the love and support! Cousin Alice Chan thanks for just being awesome. Cousin Cassandra for all our funny conversations and visiting me.

To all my CMU classmates, labmates and friends, thank you as well! Aniruddha Basak thanks for all the Pittsburgh fun and homework help! Irina Brinkster thanks for all the fun chats! Akshay Chandrashekaran thanks for all the Kaldi support and beers shared! Guan-Lin Chao thanks for all the fun chats and spiritual mentorship! Eric Chen thanks for all the general chats and advice! David Cohen thanks for all the debates on politics and Costco pizza/smoothies shared! Benjamin Elizalde thanks for all the fun chats and wine shared! Samantha Goldstein for enduring all my academic questions and processing all my visa documents! Chih Hu thanks for all the fun chats and Uber discounts! Jungsuk Kim thanks for all the Kaldi support and BABEL pain shared! Suyoun Kim thanks for driving me to In-N-Out and wine shared! Wonkyum Lee thanks for all the Kaldi support and BABEL

pain shared! Bing Liu thanks for all the research discussions and driving me to badminton! Jason Lohn thanks for taking me on as my initial advisor and giving me research freedom! Shijia Pan thanks for all the fun chats and driving me to badminton/Costco! Erik Reed thanks for all the technical chats and being an awesome course project partner! Nathan Snizaski thanks for answering all my annoying academic questions! Chihro Suga thanks for being an awesome roommate and friend! Chun Hao Tang thanks for all the fun chats! Ryan Yuan Tian thanks for being an awesome roommate and driving me to badminton! Yuan Tian thanks for all the fun chats and driving me to badminton! Guanyu Wang thanks for all the homework help and helping me move! Joy Ying Zhang thanks for teaching me the basics of machine learning and treating me to Facebook noodles! Yingrui Zhang thanks for all the fun chats and spiritual mentorship!

To all my Google coworkers, mentors and friends, thank you as well! Dzmitry Bahdanau thanks for all the fun insightful research discussions! Eugene Brevdo thanks for helping me fight long short term memory! Andrew Dai thanks for trading DistBelief tips and tricks! Jeff Dean thanks for reviewing my change list 2 hours before the end of my last internship! Navdeep Jaitly thanks for being an awesome mentor and solving speech together! Jenny Liu thanks for being an awesome mentor and all the atom/jia/thread lunches shared! Rajat Monga thanks for helping me fight DistBelief! Mohammad Norouzi thanks for all the research chats! Tara Sainath thanks for all the speech help! Ilya Sutskever thanks for all the enthusiastic research chats! Luke Vilnis thanks for all the research chats! Oriol Vinyals thanks for being an awesome mentor and all the research chats! Xuerui Wang thanks for being an awesome mentor and teaching me the basics of machine learning at Google! Yu Zhang thanks for being an awesome collaborator!

To my other friends, thank you as well! Rosanne Borja thanks for driving me to the hospital and spiritual mentorship! Sarah Chan thanks for all the life chats and bringing me to Lord's Grace Christian Church! Sharon Choy thanks for all the life and spiritual mentorship! Terry

Jiang thanks for all the life encouragement and support! Rosemary Ke thanks for being an awesome collaborator and all the general chats! Christine Lee thanks for all the years of friendship from Waterloo to silicon valley! Gilbert Leung thanks for all the Python code reviews at Google and driving me to badminton! Xiaohui Zhang thanks for all the spiritual chats!

I also thank the various organizations and funding agencies which made my research possible including the Electrical and Computer Engineering Department at Carnegie Mellon University, the Intelligence Advanced Research Projects Activity funding agency, Samsung and Google!

Finally, and most importantly, I thank God. 1 Chronicles 16:34: "Oh give thanks to the Lord, for He is good; for His steadfast love endures forever!".

Abstract

For the past few decades, the bane of Automatic Speech Recognition (ASR) systems have been phonemes and Hidden Markov Models (HMMs). HMMs assume conditional independence between observations, and the reliance on explicit phonetic representations requires expensive handcrafted pronunciation dictionaries. Learning is often via detached proxy problems, and there especially exists a disconnect between acoustic model performance and actual speech recognition performance. Connectionist Temporal Classification (CTC) character models were recently proposed attempts to solve some of these issues, namely jointly learning the pronunciation model and acoustic model. However, HMM and CTC models still suffer from conditional independence assumptions and must rely heavily on language models during decoding.

In this thesis, we question the traditional paradigm of ASR and highlight the limitations of HMM and CTC models. We propose a novel approach to ASR with neural attention models and we directly optimize speech transcriptions. Our proposed method is not only an end-to-end trained system but also an end-to-end model. The end-to-end model jointly learns all the traditional components of a speech recognition system: the pronunciation model, acoustic model and language model. Our model can directly emit English/Chinese characters or even word pieces given the audio signal. There is no need for explicit phonetic representations, intermediate heuristic loss functions or conditional independence assumptions. We demonstrate our end-to-end speech recognition model on various ASR tasks. We show competitive

results compared to a state-of-the-art HMM based system on the Google voice search task. We demonstrate an online end-to-end Chinese Mandarin model and show how to jointly optimize the Pinyin transcriptions during training. Finally, we also show state-of-the-art results on the Wall Street Journal ASR task compared to other end-to-end models.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

State-of-the-art Automatic Speech Recognition (ASR) systems are complicated and composed of many individual components: pronunciation models, acoustic models, language models and text normalization [1, 2, 3]. Many components are handcrafted (i.e., pronunciation dictionary) and combined in an ad-hoc manner (i.e., language model weights during decoding). For example, state-of-the-art ASR systems typically weigh the language model much more heavily than the acoustic model [1, 2, 3].

Each component makes various assumptions about the underlying probability distribution they model. Additionally, each component is typically constructed as an independent proxy problem within the speech system [2, 3]. For example, Hidden Markov Models (HMMs) make strong Markovian and independence assumptions between symbols [4]. Deep Neural Network (DNN) acoustic models are optimized towards frame level cross entropy [5, 6] which have no direct connection to Word Error Rate (WER), while pronunciation models are handcrafted [7] and typically never updated.

Various attempts have been made to overcome some of these assumptions and connect the detached proxy problems. Recurrent Neural Network (RNN) language models which do not have Markovian assumptions are used to rescore n-best lists [8]. Sequence training methods

have been developed to bridge the gap between acoustic models and WERs with sequence level objectives [9, 10]. End-to-end Connectionist Temporal Classification (CTC) methods were shown to learn the pronunciation and acoustic model jointly, while directly optimizing [11].

However, all these systems still depend on components that are optimized separately with its own probability distribution assumptions. HMM and CTC systems require a n-gram language model which makes Markovian assumptions and are updated independently or not at all [12, 13, 2, 3]. Additionally, HMM and CTC models still hold the conditional independence assumption between output symbols, which is simply not true in speech. Consequently, HMM and CTC based systems must carry around a (large) n-gram language model to decode [2, 3, 11]. This makes it impractical to deploy on any computational platform with limited memory.

In this thesis, we overcome the aforementioned issues with a new end-to-end speech recognition model. Our model will learn to transcribe an audio sequence signal directly to a word sequence, one character (or word piece) at a time. We will jointly learn all the components of a typical ASR system in one model. We will also not make any conditional independence or Markovian assumptions of the output sequence given the acoustics. Our contribution is not only an end-to-end trained system, but also an end-to-end model.

1.1 Research Contributions

The main contributions of our thesis is as follows:

1. **An offline end-to-end speech ASR model for English** (Chapter 3): We present an end-to-end ASR model that will jointly learn all the components (pronunciation, acoustic, language and text normalization) all within one model. We show competitive results to a state-of-the-art HMM based system on a real world Google voice search ASR task.

2. **An online end-to-end ASR model for Chinese Mandarin** (Chapter 4): We extend on our offline English model and make it online. We also show how to jointly optimize between Mandarin Characters and Mandarin Pinyin. We show the joint Pinyin optimization can improve performance without any increase in inference cost.

3. **An end-to-end ASR model with word pieces** (Chapter 5): We present an end-to-end ASR model which learns to decompose text and emit word pieces. We show state-of-the-art results compared to other end-to-end ASR models on the Wall Street Journal ASR task.

1.1.1 Applications

We believe our end-to-end ASR models are extremely applicable to many real word tasks. Unlike HMM and CTC based systems, our model can learn language directly and not rely on a n-gram language model for decoding. This is especially useful for applications which have limited memory, for example smart phones, smart watches and smart glasses.

We also show our models are competitive to state-of-the-art HMM based systems [14]. Our models also exhibit a very different posterior distribution compared to CTC or HMM based systems (i.e., Markovian and conditional independence assumptions). Our model can be used as a rescoring system to traditional HMM based systems [11, 14].

1.2 Thesis Outline

Chapter 2 will review state-of-the-art HMM based ASR systems. We will review the assumptions and proxy problems a standard HMM based system will make. This will give motivation for our thesis. Chapter 2 will also review Sequence-to-Sequence [15, 16] and Attention [17] methods. We describe their recent success in Machine Translation and give the foundation to our models and applications to ASR.

In Chapter 3, we present an end-to-end ASR model that can jointly learn all the components (acoustics, pronunciation, language and text normalization) all within one model. We show our model to be competitive to a state-of-the-art HMM based system on a real world Google English voice search task. Chapter 4 will extend on Chapter 3 to make the model online and search-free. We will also explore Chinese Mandarin as our target language and show multi-task learning with joint Mandarin characters and Pinyin. Chapter 5 will explore emitting word pieces or sub-word units. We also show state-of-the-art results compared to other end-to-end ASR models under the language model free setting. We will finally close up with a conclusion and future work in Chapter 6.

Chapter 2

Background

This chapter will review state-of-the-art ASR methods [2, 3] and Connectionist Temporal Classification (CTC) end-to-end speech recognition systems [11]. We will discuss their limitations, and give motivation for our thesis on end-to-end speech recognition models. We will also review and give an introduction to the Sequence-to-Sequence (seq2seq) framework [15, 16], which has much higher expressive power compared to HMM and CTC models. We give motivation and intuition to using seq2seq for ASR.

2.1 Hidden Markov Model Systems

This section will give a modern overview of a typical state-of-the-art DNN-HMM speech recognition system [2, 3]. Figure 2.1 gives a visualization.

An acoustic signal is first processed by a signal processing frontend, wherein we extract audio features from the raw audio waveform. We decompose the waveform into a sequence of frames of features. Example of feature representations include: log-mel Filterbanks (Fbanks), Mel-Frequency Cepstral Coefficients (MFCCs) [18] and Perceptual Linear Prediction (PLP) [19]. Recent research has questioned whether this frontend stage is even needed [20] wherein the ASR system can directly model the waveform directly without any feature engineering.

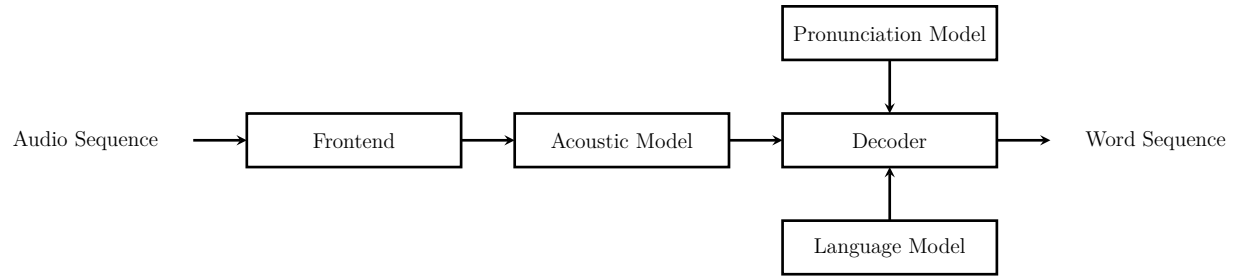

Figure 2.1: A typical speech recognition system consisting of a signal processing frontend, acoustic model, pronunciation model, language model and a decoder.

The audio features are then fed to an acoustic model which classifies a sequence of features into a sequence of phonemes. This is typically done via a DNN [5, 21, 22, 6, 23, 24, 25, 26], Convolutional Neural Network (CNN) [27, 28, 29, 30, 31, 32, 33] or RNN acoustic model [34, 35, 36, 37]. The acoustic model assumes conditional independence and Markovian assumptions, given the acoustic signal (or representation of the acoustic signal), the phoneme predictions are independent in neighbouring frames.

The pronunciation model (also known as the pronunciation dictionary) maps a word (or sequence of words) into a sequence of phones. The pronunciation dictionary is typically handcrafted [7], however it can be also be a learnt statistical model [38]. The pronunciation dictionary is typically fixed and rarely updated (if at all) in the speech pipeline, and the ASR system is only capable of modeling words that exist in the pronunciation dictionary.

The language model is a statistical model giving probability of word sequences independent of the acoustic. N-gram language models are typically used during decoding which hold Markovian assumptions [1]. RNN language models which do not hold Markovian assumptions and are typically more powerful can be used to rescore lattices or n-best hypotheses after decoding [8].

The decoder combines the information from the acoustic model, pronunciation model and language model together to search for the best word sequences given the acoustic signal. Modern decoders are usually implemented as a Weighted Finite State Transducer (WFST)

for efficient searching. The graph requires finite states, and in practice requires Markovian assumptions in its representation. The graph combines the pronunciation dictionary rules, with the acoustic model's HMM transition probabilities and n-gram language model's probabilities all of which hold Markovian assumptions.

2.1.1 DNN-HMM System

We now describe a typical state-of-the-art DNN-HMM speech recognition system in more formal mathematical notation. Given an acoustic signal $\mathbf{x}$ we want to model the word sequence $\mathbf{w}$. We can decompose this probability into two terms using Bayes' rule, an acoustic model $p_{\text{AM}}(\mathbf{x}|\mathbf{w})$ and a language model $p_{\text{LM}}(\mathbf{w})$:

$$p(\mathbf{w}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{w})p(\mathbf{w})}{p(\mathbf{x})} \propto p(\mathbf{x}|\mathbf{w})p(\mathbf{w}) = p_{\text{AM}}(\mathbf{x}|\mathbf{w})p_{\text{LM}}(\mathbf{w}) \tag{2.1}$$

2.1.2 Acoustic Model

The acoustic model $p_{\mathbf{AM}}(\mathbf{x}|\mathbf{w})$ is composed of a DNN-HMM model. First the DNN models an acoustic frame x_t into a phoneme state posterior q_t at time t [6]:

$$p(q_t|x_t) = \text{DNN}_{\text{AM}}(x_t) \tag{2.2}$$

or we can also use a RNN acoustic model [35]:

$$p(\mathbf{q}|\mathbf{x}) = \text{RNN}_{\text{AM}}(\mathbf{x}) \tag{2.3}$$

For each time t, we can get the likelihood probability of the phoneme state by simply dividing

the posterior against the prior $p(q_t)$:

$$p(x_t|q_t) = \frac{p(q_t|x_t)p(x_t)}{p(q_t)} \tag{2.4}$$

and since $p(x_t)$ is independent of the word sequence during the search process, it is typically dropped during inference [21].

The phoneme state observations can be (deterministically) mapped to a HMM triphone state. HMM transition probabilities model the most likely triphone sequence:

$$p(q_{t+1} = u|q_t = v) = \beta(u, v) \tag{2.5}$$

where u and v are the HMM states and β is the transition matrix of the HMM.

Typically the phoneme alignment q_t is generated by a Gaussian Mixture Model (GMM) [6]. A GMM-HMM acoustic model is first trained with Expectation Maximization [39]. The training dataset is then aligned using the GMM and the alignments will be used to train the DNN. There is no theoretical justification for such an approach, but it works well in practice [21, 22, 23, 40]. The DNN acoustic model is at the mercy of the GMM model's ability to generate good alignments (i.e., if the GMM generates garbage alignments, the DNN will learn garbage). Typically to achieve state-of-the-art performance, handcrafted recipes of realignment is necessary [40]. These methods are typically not published since they often relate to proprietary engineering recipes rather than principed methods. In our thesis, we will try and remove some of these ad-hoc training methods.

The rest of this section will discuss the limitations and assumptions of the DNN-HMM system. We will also talk about how the DNN-HMM system is combined together during decoding and why the system must depend on a strong language model.

Markovian Assumption

The transition probabilities of the HMM have a Markovian assumption [4]. The transition probabilities between each triphone state is conditioned only on the previous triphone state:

$$p(q_t|q_{<t}) = p(q_t|q_{t-1}) \tag{2.6}$$

The Markovian assumption means the model is incapable of modeling longterm dependencies – the transition of phonemes is only local (which is simply not true). For example, any word that are several syllables long would break this assumption.

Conditional Independence Assumption

The emission probabilities of the HMM have a conditional independence assumption [4]. The emission probabilities between each frame is conditionally independent given the state q_t:

$$p(x_t|x_{<t}, q_{\leq t}) = p(x_t|q_t) \tag{2.7}$$

In a DNN acoustic model, the phoneme posteriors are conditionally independent given the frame x_t:

$$p(q_t|\mathbf{x}, q_{<t}) = p(q_t|x_t) \tag{2.8}$$

In a RNN acoustic model, the phoneme posteriors are conditionally independent given $\mathbf{x}$:

$$p(q_t|\mathbf{x}, q_{<t}) = p(q_t|\mathbf{x}) \tag{2.9}$$

The conditional independence assumptions mean the model is incapable of learning the language of the phonemes – i.e., the model assumes there are no dependencies between

phonemes in speech (which is simply not true). For example, phones often come in pairs (i.e., diphones) and have statistical correlations.

Cross Entropy Error and Sequence Training

The DNN or RNN acoustic model is typically optimized with a proxy criteria, namely the cross entropy error or phoneme classification per frame. Researchers have found there is a disconnect relationship between frame accuracy and WER – namely, improving an acoustic model frame accuracy did not necessarily yield better WERs [31]. Sequence training [10] have been introduced to ameliorate this issue, however sequence training can only improve the acoustic model while the pronunciation model and language model are left untouched.

Pronunciation Dictionary

The acoustic model we described thus far generates a phone sequence $\mathbf{q}$ conditioned on the acoustics $\mathbf{x}$, however we desire a word sequence $\mathbf{w}$. The pronunciation dictionary constrains the model to generate only valid phone sequences, additionally the dictionary maps the phone sequences into word sequences:

$$\mathbf{w} = \text{PronunciationDictionary}(\mathbf{q}) \tag{2.10}$$

The existence and explicit representation of phonemes itself is a very strong assumption. In English, the pronunciation dictionary typically consists of 39 phonemes [7]. The pronunciation dictionary assumes all English words across all different accents and speaking styles can be broken down into these 39 phonemes.

2.1.3 Language Model

The language model $p_{\mathbf{LM}}(\mathbf{w})$ models the most likely sequence of words independent of the acoustics. This is typically accomplished with a n-gram language model [1]. A n-gram lan-

guage model has a Markovian assumption, typically conditioning only on 2-4 word history (i.e., trigram language model) and therefore loses long range context dependency. RNN language models do not have this Markovian assumption and are capable of modeling longterm dependencies. However, RNN language models are too expensive to use during the decoding (and can not fit inside a WFST), and typically only used to rescore lattices n-best lists after decoding [8]. This means during the beam search, we may lose the correct beam before we even get to the language model rescoring stage.

Language models (LMs) are trained independent of the acoustics. Additionally, LMs are typically optimized for perplexity, which have no direct connection to WER [1]. The LMs are almost never optimized end-to-end with the result of the ASR system [2, 3].

2.1.4 Decoding

During decoding, we want to find the best utterance $\mathbf{w}$ given the acoustics:

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} p_{\text{AM}}(\mathbf{x}|\mathbf{w}) p_{\text{LM}}(\mathbf{w}) \tag{2.11}$$

However, in practice Equation 2.11 does not work. Typically the p_{AM} is a very weak model due to the independence assumptions mentioned above [1]. Instead, we put a language model weight β on the LM where $\beta = O(10)$ to give the LM p_{LM} more weight over the acoustic model p_{AM}.

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} p_{\text{AM}}(\mathbf{x}|\mathbf{w}) p_{\text{LM}}(\mathbf{w})^{\beta} \tag{2.12}$$

While this is no longer mathematically sound, this is implemented in virtually all state-of-the-art ASR systems [2, 3]. In practice this means we must carry around a strong language model and rely on it heavily [2, 3]. This is contrary to intuition, state-of-the-art speech recognizers rely more heavily on the language model than the acoustic model.

2.2 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) [41] is a technique to transform a variable sequence $\mathbf{x}$ to another variable sequence $\mathbf{y}$ where the sequence length of the predicted sequence $|\mathbf{y}|$ is less than the original sequence length $|\mathbf{x}|$. We define a special blank token "$-$" to help us construct alignments $\mathbf{a}$. In a more formal mathematical definition:

$$p(\mathbf{a}|\mathbf{x}) = \text{RNN}(\mathbf{x}) \tag{2.13}$$

where the two sequences $\mathbf{x}$ and $\mathbf{a}$ have the same lengths, $|\mathbf{x}| = |\mathbf{a}|$ and RNN is simply some RNN function (e.g., LSTM [42]). We can marginalize the probability and sum over all possible alignments:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} p(\mathbf{a}|\mathbf{x}) \tag{2.14}$$

where the operator $\mathcal{B}$ removes blanks and repeats from a sequence, for example $\mathcal{B}(a-ab-) = \mathcal{B}(-aa--abb) = aab$; and each $\mathbf{a}$ is an alignment of $\mathbf{y}$. The model can be optimized to maximize the likelihood of $p(\mathbf{y}|\mathbf{x})$ by marginalizing over all possible alignments using dynamic programming [41]. CTC can be used to directly model acoustics $\mathbf{x}$ into English characters $\mathbf{y}$ [11]. As opposed to DNN-HMM models, CTC models have been shown to learn the pronunciation model directly, and not rely on an explicit pronunciation dictionary.

CTC models and its variants have been applied successfully to many ASR tasks [11, 43, 44, 45, 46, 47, 48, 49]. However, they tend to rely heavily on an explicit LM since the CTC model assumes conditional independence and can not learn complicated multi-modal language distributions.

Independence Assumption

CTC assumes a conditional independence assumption between each output symbol a_t conditioned on the input sequence $\mathbf{x}$:

$$p(a_t|a_{<t}, \mathbf{x}) = p(a_t|\mathbf{x}) \tag{2.15}$$

Even if there is a deep RNN between $\mathbf{x}$ and $\mathbf{a}$, the model still has strong Markovian assumptions. Each prediction at the frame level is conditionally independent to neighbouring frames given the previous layer's output. Consequently, CTC can use dynamic programming to compute its log likelihood and gradient. However, the CTC model can not learn the language of the output symbol (whether phonemes or English characters). CTC models must rely on a separate model to model the conditional dependency between the output symbols. CTC can jointly learn a pronunciation model and acoustic model, however it is incapable of learning a language model due to its conditional independence assumption. Similar to HMM based systems, CTC models must carry around a strong n-gram language model to decode [11, 44, 46, 48].

2.3 Sequence-to-Sequence

Sequence-to-Sequence (seq2seq) [15, 16] is a general framework that encodes an input sequence $\mathbf{x}$ of any variable length to some latent representation $\mathbf{h}$, and uses $\mathbf{h}$ to decode an output sequence of any variable length $\mathbf{y}$. In the original implementation [15, 16], $\mathbf{h}$ is simply a fixed context vector. An encoder RNN encodes the input sequence $\mathbf{x}$ into a fixed context vector h and a decoder RNN takes h and emits the output sequence $\mathbf{y}$ one token at a time conditioned on all previous emitted tokens. Here, h is simply the last state of encoder RNN and initial state of the decoder RNN. More formally, let input sequence $\mathbf{x}$ be a sequence of length T, output sequence $\mathbf{y}$ be a sequence of length S, $\mathbf{h}$ be the RNN state of the encoder

and $\mathbf{g}$ be the RNN state of the decoder:

$$h_t = \text{EncodeRNN}(\mathbf{x}_t, h_{t-1}) \tag{2.16}$$

$$g_0 = h_T \tag{2.17}$$

$$g_s = \text{DecodeRNN}(\mathbf{y}_{s-1}, g_{s-1}) \tag{2.18}$$

$$p(y_s|\mathbf{x}, \mathbf{y}_{<s}) = \text{TokenDistribution}(g_s) \tag{2.19}$$

where EncodeRNN and DecodeRNN are RNN transfer functions (e.g., LSTM [42]) and TokenDistribution is a Multilayer Perceptron (MLP) network with softmax outputs modelling the output distribution.

The seq2seq framework was originally applied to machine translation [15, 16, 50, 51]. The source sentence (i.e., English words) $\mathbf{x}$ would be encoded into a fixed context vector h. A decoder would take h and decode it into the target sentence $\mathbf{y}$ (i.e., French words). It can even be applied to image captioning where $\mathbf{x}$ is an image and $\mathbf{y}$ is the image caption [52].

Chain Rule

Unlike HMM and CTC models, seq2seq models do not assume conditional independence nor Markovian assumptions. It relies on the chain rule for a left-to-right factorization:

$$p(\mathbf{y}|\mathbf{x}) = \prod_s p(y_s|\mathbf{x}, \mathbf{y}_{<s}) \tag{2.20}$$

where each $p(y_s|\mathbf{x}, \mathbf{y}_{<s})$ is modelled by the TokenDistribution function.

This means seq2seq models can learn non-Markovian distributions and complicated multi-modal distributions, where as HMM and CTC based models can not learn multi-modal non-Markovian distributions even with a deep RNN preceding the output layer. This is especially important in speech, since speech is neither Markovian nor unimodal. For example, is "mister" spelled as "mister", "mr" or "mr."? Each prediction made by a HMM or CTC based

model is not conditioned on the previously emitted tokens (i.e., Markovian assumption), consequently it is unlikely for such a system to generate such varied parses.

The conditional dependency of the seq2seq framework also means it is very easy to overfit the training distribution. We introduce several techniques later in this thesis to overcome this issue.

2.3.1 Attention

Attention [53] is a mechanism to locate and extract information from a memory source. One implementation of attention is content-based attention [17], where there is a content-based query and memory source. We extract information from the memory using the information derived from the query. More formally, let $\mathbf{h}$ be our memory source and q be our content-based query, the content-based attention mechanism 1] computes energies between $\mathbf{h}$ and q, 2] uses the energies to create an alignment distribution, and 3] uses the alignment distribution to create a context vector encapsulating the information of interest:

$$e = \text{DistanceMetric}(\mathbf{h}, q) \tag{2.21}$$

$$\alpha = \text{Normalize}(e) \tag{2.22}$$

$$c = \sum_t \alpha_t h_t \tag{2.23}$$

In the original implementation of seq2seq with attention [17], the DistanceMetric function is a learnable MLP network and Normalize is simply the softmax function. However, DistanceMetric can just as easily be the cosine distance [54] or dot product [55, 14].

In the context of seq2seq learning, the attention mechanism creates an explicit alignment between the input sequence $\mathbf{x}$ and the output sequence $\mathbf{y}$. The content-based attention query is simply the decoder RNN's state $q = g_s$. At each decoder timestep s, we create an α distribution over the encoder sequence $\mathbf{h}$ to extract a single context vector c to make

the y_s prediction. The attention mechanism creates a short circuit between the encoder and the decoder. This allows for much more efficient information flow during the forward propagation and gradient flow during the backward propagation.

The seq2seq framework with attention has been applied successfully to many applications including machine translation [56, 57], conversation modelling [58], parsing [59], image captioning [60], grapheme-to-phoneme [61] and phoneme recognition [62, 63].

In this thesis, we will explore seq2seq with attention for end-to-end speech recognition. The seq2seq framework allows us to model speech directly from acoustics to the output token units without using any handcrafted phonetic representations. We do not make conditional independence assumptions nor Markovian assumptions and allow our model to directly learn and model language (unlike HMM and CTC based models). Chapter 3 will begin with a description of our end-to-end speech recognition model.

Chapter 3

Listen, Attend and Spell

In this chapter, we present Listen, Attend and Spell (LAS): A Neural Network for Large Vocabulary Conversational Speech Recognition [55, 14]. We will present LAS as an end-to-end speech recognition model and show competitive results to a state-of-the-art HMM based system on the Google voice search task. This chapter is based off publications in [55, 14] and portions of the text and figures are (C) 2016 IEEE and reprinted with permission.

3.1 End-to-End Speech Recognition Model

LAS is an end-to-end speech recognition model. LAS learns to transcribe an audio sequence signal to a word sequence, one character at a time, without using explicit language models, pronunciation models, HMMs, etc. LAS does not make any independence assumptions about the nature of the probability distribution of the output character sequence, given the input acoustic sequence. This method is based on the sequence-to-sequence learning framework with attention [15, 16, 17, 62, 63].

It consists of an encoder Recurrent Neural Network (RNN), which is named the *listener*, and a decoder RNN, which is named the *speller*. The listener is a pyramidal RNN that converts speech signals into high level features. The speller is a RNN that transduces these higher

level features into output utterances by specifying a probability distribution over the next character, given all of the acoustics and the previous characters. At each step the RNN uses its internal state to guide an attention mechanism [17, 62, 63] to compute a "context" vector from the high level features of the listener. It uses this context vector, and its internal state to both update its internal state and to predict the next character in the sequence. The entire model is trained jointly, from scratch, by optimizing the probability of the output sequence using a chain rule decomposition. We call this an *end-to-end model* because all the components of a traditional speech recognizer are integrated into its parameters, and optimized together during training, unlike *end-to-end training* of conventional models that attempt to adjust acoustic models to work well with the other fixed components of a speech recognizer [2, 3].

Our model was inspired by [62, 63] that showed how end-to-end recognition could be performed on the TIMIT phone recognition task. We note a recent paper from the same group that describes an application of these ideas to WSJ [64]. Our research was conducted independently, and we explore the challenges associated with the application of these ideas to large scale conversational speech recognition on a Google voice search task. We defer a discussion of the relationship between these and other methods to section 3.4.

3.2 Model

In this section, we formally describe LAS. Let $\mathbf{x} = (x_1, \ldots, x_T)$ be the input sequence of filter bank spectra features and $\mathbf{y} = (\langle\text{sos}\rangle, y_1, \ldots, y_S, \langle\text{eos}\rangle)$ and each $y_i \in \{a, \cdots, z, 0, \cdots, 9, \langle\text{space}\rangle, \langle\text{comma}\rangle, \langle\text{period}\rangle, \langle\text{apostrophe}\rangle, \langle\text{unk}\rangle\}$ be the output sequence of characters. Here $\langle\text{sos}\rangle$ and $\langle\text{eos}\rangle$ are the special start-of-sentence token, and end-of-sentence tokens, respectively, and $\langle\text{unk}\rangle$ are unknown tokens such as accented characters.

LAS models each character output y_i as a conditional distribution over the previous charac-

ters $y_{<i}$ and the input signal $\mathbf{x}$ using the chain rule for probabilities:

$$p(\mathbf{y}|\mathbf{x}) = \prod_i p(y_i|\mathbf{x}, y_{<i}) \tag{3.1}$$

This objective makes the model a discriminative, end-to-end model, because it directly predicts the conditional probability of character sequences, given the acoustic signal.

LAS consists of two sub-modules: the listener and the speller. The listener is an acoustic model encoder that performs an operation called Listen. The Listen operation transforms the original signal $\mathbf{x}$ into a high level representation $\mathbf{h} = (h_1, \ldots, h_U)$ with $U \leq T$. The speller is an attention-based character decoder that performs an operation we call AttendAndSpell. The AttendAndSpell operation consumes $\mathbf{h}$ and produces a probability distribution over character sequences:

$$\mathbf{h} = \text{Listen}(\mathbf{x}) \tag{3.2}$$

$$p(y_i|\mathbf{x}, y_{<i}) = \text{AttendAndSpell}(y_{<i}, \mathbf{h}) \tag{3.3}$$

Figure 3.1 depicts these two components. We provide more details of these components in the following sections.

3.2.1 Listen

The Listen operation uses a Bidirectional Long Short Term Memory RNN (BLSTM) [42, 65, 11] with a pyramidal structure. This modification is required to reduce the length U of $\mathbf{h}$, from T, the length of the input $\mathbf{x}$, because the input speech signals can be hundreds to thousands of frames long. A direct application of BLSTM for the operation Listen converged slowly and produced results inferior to those reported here, even after a month of training time. This is presumably because the operation AttendAndSpell has a hard time extracting the relevant information from a large number of input timesteps.

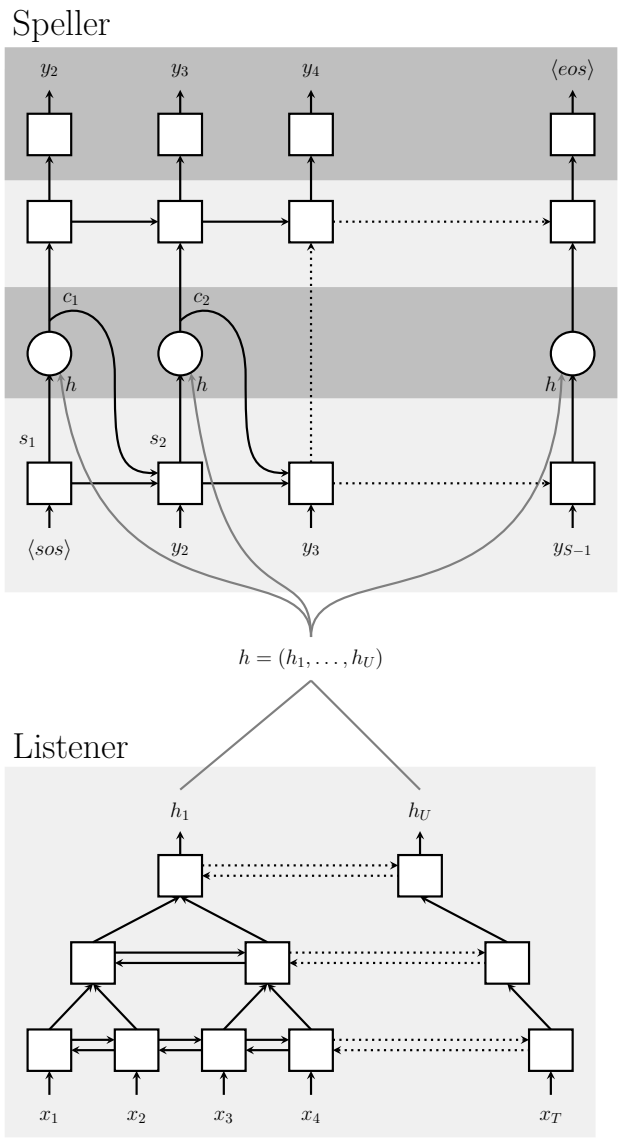

Figure 3.1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence **x** into high level features **h**, the speller is an attention-based decoder generating the **y** characters from **h**.

We circumvent this problem by using a pyramidal BLSTM (pBLSTM). In each successive stacked pBLSTM layer, we reduce the time resolution by a factor of 2. In a typical deep BLSTM architecture, the output at the i-th timestep, from the j-th layer is computed as follows:

$$h_i^j = \text{BLSTM}(h_{i-1}^j, h_i^{j-1}) \tag{3.4}$$

In the pBLSTM model, we concatenate the outputs at consecutive steps of each layer before feeding it to the next layer, i.e.:

$$h_i^j = \text{pBLSTM}(h_{i-1}^j, \left[h_{2i}^{j-1}, h_{2i+1}^{j-1}\right]) \tag{3.5}$$

In our model, we stack 3 pBLSTMs on top of the bottom BLSTM layer to reduce the time resolution $2^3 = 8$ times. This allows the attention model (described in the next section) to extract the relevant information from a smaller number of timesteps. In addition to reducing the resolution, the deep architecture allows the model to learn nonlinear feature representations of the data. See Figure 3.1 for a visualization of the pBLSTM.

The pyramidal structure also reduces the computational complexity. The attention mechanism in the speller has a computational complexity of $O(US)$. Thus, reducing U speeds up learning and inference significantly. Other neural network architectures have been described in literature with similar motivations, including the hierarchical RNN [66], clockwork RNN [67] and CNN [68].

3.2.2 Attend and Spell

The AttendAndSpell function is computed using an attention-based LSTM transducer [17, 63]. At every output step, the transducer produces a probability distribution over the next character conditioned on all the characters seen previously. The distribution for y_i is a function of the decoder state s_i and context c_i. The decoder state s_i is a function of the previous state s_{i-1}, the previously emitted character y_{i-1} and context c_{i-1}. The context

vector c_i is produced by an attention mechanism. Specifically,

$$c_i = \text{AttentionContext}(s_i, \mathbf{h}) \tag{3.6}$$

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \tag{3.7}$$

$$p(y_i|\mathbf{x}, y_{<i}) = \text{CharacterDistribution}(s_i, c_i) \tag{3.8}$$

where CharacterDistribution is an MLP with softmax outputs over characters, and where RNN is a 2 layer LSTM.

At each timestep, i, the attention mechanism, AttentionContext generates a context vector, c_i encapsulating the information in the acoustic signal needed to generate the next character. The attention model is content based - the contents of the decoder state s_i are matched to the contents of h_u representing timestep u of $\mathbf{h}$, to generate an attention vector α_i. The vectors h_u are linearly blended using α_i to create c_i.

Specifically, at each decoder timestep i, the AttentionContext function computes the scalar energy $e_{i,u}$ for each timestep u, using vector $h_u \in \mathbf{h}$ and s_i. The scalar energy $e_{i,u}$ is converted into a probability distribution over timesteps (or attention) α_i using a softmax function. The softmax probabilities are used as mixing weights for blending the listener features h_u to the context vector c_i for output timestep i:

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle \tag{3.9}$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_{u'} \exp(e_{i,u'})} \tag{3.10}$$

$$c_i = \sum_u \alpha_{i,u} h_u \tag{3.11}$$

where ϕ and ψ are MLP networks. After training, the α_i distribution is typically very sharp and focuses on only a few frames of $\mathbf{h}$; c_i can be seen as a continuous bag of weighted features of $\mathbf{h}$. Figure 3.1 shows the LAS architecture.

3.2.3 Optimization

We train the parameters of our model to maximize the log probability of the correct sequences:

$$\theta = \max_{\theta} \sum_{i} \log p(y_i | \mathbf{x}, y_{<i}; \theta) \tag{3.12}$$

however, there is a mismatch between training and inference conditions. During inference (i.e., beam search) we do not have access to the ground truth of the previously emitted characters $\mathbf{y}_{<i}$. Consequently, the model is not robust to conditioning on errors that may happen during beam search.

Additionally, the seq2seq framework (with the conditional dependency) makes it very easy to overfit the training distribution. For example, in our initial experiments, we found it is possible to train a seq2seq model (without attention) to achieve zero training error but without any generalization. During inference, the model would just recall an utterance verbatim from the training distribution.

We can ameliorate this issue by using a technique similar to scheduled sampling [69]. During training, we sample and condition from predictions produced by our own model's posterior. This makes our model much more robust to our own model's error produced during inference. This technique will allow the model to be more robust to substitution errors but not insertion or deletion errors.

$$\tilde{\theta} = \max_{\theta} \sum_{i} \log p(y_i | \mathbf{x}, \tilde{y}_{<i}; \theta) \tag{3.13}$$

where $\tilde{y}_{i-1}$ is the ground truth previous character or a character randomly sampled (with 10% probability) from the model, i.e. CharacterDistribution(s_{i-1}, c_{i-1}).

3.2.4 Decoding and Rescoring

During inference we want to find the most likely character sequence given the input acoustics:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \log p(\mathbf{y}|\mathbf{x}) \tag{3.14}$$

We use a simple left-to-right beam search similar to [15].

We can also apply language models trained on large external text corpora alone, similar to conventional speech systems [3]. We simply rescore our beams with the language model. We find that our model has a small bias for shorter utterances so we normalize our probabilities by the number of characters $|\mathbf{y}|_c$ in the hypothesis and combine it with a language model probability $p_{\text{LM}}(\mathbf{y})$:

$$s(\mathbf{y}|\mathbf{x}) = \frac{\log p(\mathbf{y}|\mathbf{x})}{|\mathbf{y}|_c} + \lambda \log p_{\text{LM}}(\mathbf{y}) \tag{3.15}$$

where λ is our language model weight and can be determined by a held-out validation set.

3.3 Experiments

We used a dataset with three million Google voice search utterances (representing 2000 hours of data) for our experiments. Approximately 10 hours of utterances were randomly selected as a held-out validation set. Data augmentation was performed using a room simulator, adding different types of noise and reverberations; the noise sources were obtained from YouTube and environmental recordings of daily events [37]. This increased the amount of audio data by 20 times. 40-dimensional log-mel filter bank features were computed every 10ms and used as the acoustic inputs to the listener. A separate set of 22K utterances representing approximately 16 hours of data were used as the test data. A noisy test data set was also created using the same corruption strategy that was applied to the training data.

All training sets are anonymized and hand-transcribed, and are representative of Googles speech traffic.

The text was normalized by converting all characters to lower case English alphanumerics (including digits). The punctuations: space, comma, period and apostrophe were kept, while all other tokens were converted to the unknown ⟨unk⟩ token. As mentioned earlier, all utterances were padded with the start-of-sentence ⟨sos⟩ and the end-of-sentence ⟨eos⟩ tokens.

The state-of-the-art model on this dataset is a CLDNN-HMM system that was described in [37]. The CLDNN system achieves a WER of 8.0% on the clean test set and 8.9% on the noisy test set. However, we note that the CLDNN uses unidirectional CLDNNs and would certainly benefit from the use of a bidirectional CLDNN architecture.

For the Listen function we used 3 layers of 512 pBLSTM nodes (i.e., 256 nodes per direction) on top of a BLSTM that operates on the input. This reduced the time resolution by $8 = 2^3$ times. The Spell function used a two layer LSTM with 512 nodes each. The weights were initialized with a uniform distribution $\mathcal{U}(-0.1, 0.1)$.

Asynchronous Stochastic Gradient Descent (ASGD) was used for training our model [70]. A learning rate of 0.2 was used with a geometric decay of 0.98 per 3M utterances (i.e., 1/20-th of an epoch). We used the DistBelief framework [70] with 32 replicas, each with a minibatch of 32 utterances. In order to further speed up training, the sequences were grouped into buckets based on their frame length [15].

The model was trained using groundtruth previous characters until results on the validation set stopped improving. This took approximately two weeks. The model was decoded using beam search with n-best list hypothesis kept where $n = 32$ and achieved 16.2% WER on the clean test set and 19.0% WER on the noisy test set without any dictionary or language model. We found that constraining the beam search with a dictionary had no impact on the WER. Rescoring the top 32 beams with the same n-gram language model that was used by the CLDNN system using a language model weight of $\lambda = 0.008$ improved the results for the

Table 3.1: WER comparison on the clean and noisy Google voice search task. The CLDNN-HMM system is the state-of-the-art system, the Listen, Attend and Spell (LAS) models are decoded with a beam search and $n = 32$. Language Model (LM) rescoring was applied to our beams, and a sampling trick was applied to bridge the gap between training and inference.

Model	**Clean WER**	**Noisy WER**
CLDNN-HMM [37]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

clean and noisy test sets to 12.6% and 14.7% respectively. Note that for convenience, we did not decode with a language model, but rather only rescored the top 32 beams. It is possible that further gains could have been achieved by using the language model during decoding.

As mentioned in Section 3.2.3, there is a mismatch between training and testing. During training the model is conditioned on the correct previous characters but during testing mistakes made by the model corrupt future predictions. We trained another model by sampling from our previous character distribution with a probability of 10% (we did not use a schedule as described in [69]). This improved our results on the clean and noisy test sets to 14.1% and 16.5% WER respectively when no language model rescoring was used. With language model rescoring, we achieved 10.3% and 12.0% WER on the clean and noisy test sets, respectively. Table 3.1 summarizes these results.

On the clean test set, this model is within 2.5% absolute WER of the state-of-the-art CLDNN-HMM system, while on the noisy set it is less than 3.0% absolute WER worse. We suspect that convolutional filters could lead to improved results, as they have been reported to improve performance by 5% relative WER on clean speech and 7% relative on noisy speech compared to non-convolutional architectures [37].

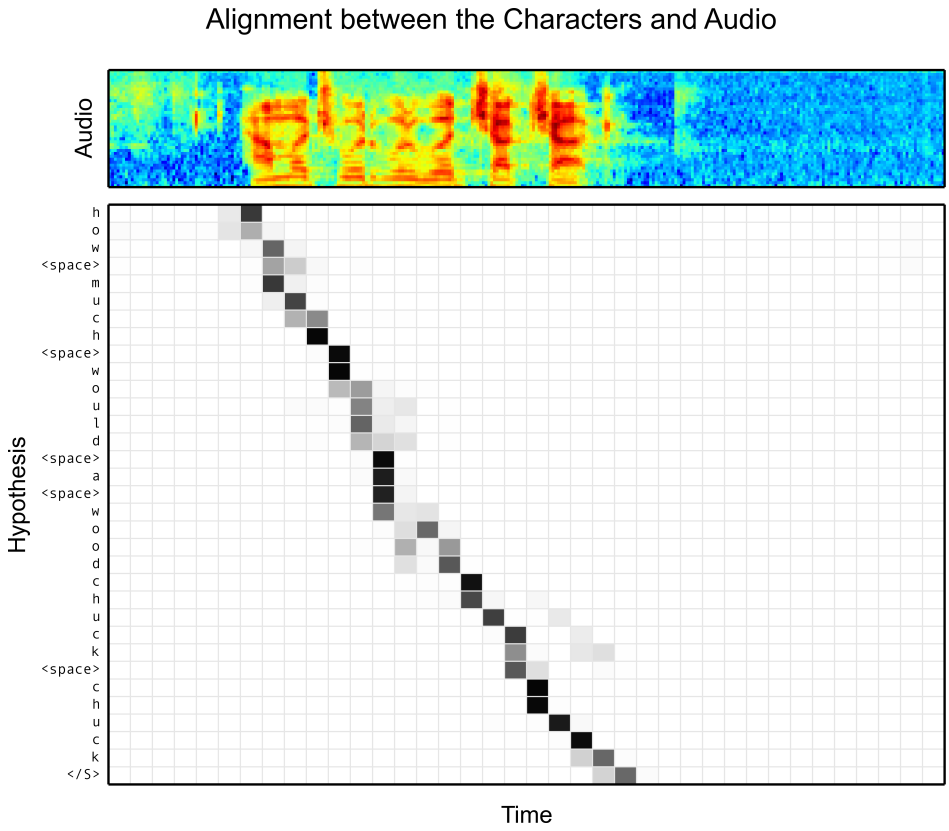

Figure 3.2: Alignments between character outputs and audio signal produced by the Listen, Attend and Spell (LAS) model for the utterance "how much would a woodchuck chuck". The content based attention mechanism was able to identify the start position in the audio sequence for the first character correctly. The alignment produced is generally monotonic without a need for any location based priors.

3.3.1 Attention Visualization

The content-based attention mechanism creates an explicit alignment between the characters and audio signal. We can visualize the attention mechanism by recording the attention distribution on the acoustic sequence at every character output timestep. Figure 3.2 visualizes the attention alignment between the characters and the filterbanks for the utterance "how much would a woodchuck chuck". For this particular utterance, the model learnt a monotonic distribution without any location priors. The words "woodchuck" and "chuck" have acoustic similarities, the attention mechanism was slightly confused when emitting "woodchuck" with a dilution in the distribution. The attention model was also able to identify the start and

end of the utterance properly.

In the following sections, we report results of control experiments that were conducted to understand the effects of n in the beam search, utterance lengths and word frequency on the WER of our model.

3.3.2 Effects of Beam Width

We investigate the correlation between the performance of the model and the n-best list hypothesis size of the beam search, with and without the language model rescoring. Figure 3.3 shows the effect of the decode beam search n, β, on the WER for the clean test set. We see consistent WER improvements by increasing the n up to 16, after which we observe no significant benefits. At a n of 32, the WER is 14.1% and 10.3% after language model rescoring. Rescoring the top 32 beams with an oracle produces a WER of 4.3% on the clean test set and 5.5% on the noisy test set.

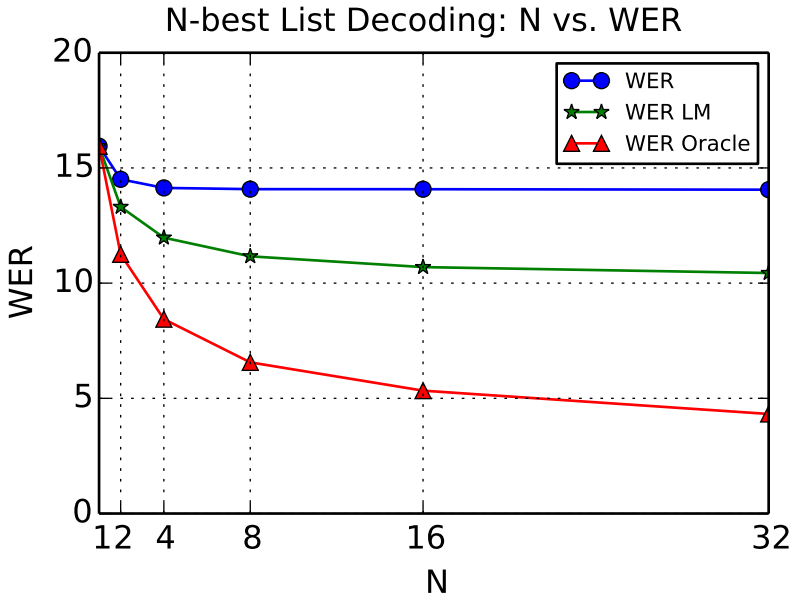

Figure 3.3: The effect of the decode n-best hypothesis on WER for the clean Google voice search task. The reported WERs are without a dictionary or language model, with language model rescoring and the oracle WER for different beam search n. The figure shows that good results can be obtained even with a relatively small beam size.

3.3.3 Effects of Utterance Length

We measure the performance of our model as a function of the number of words in the utterance. We expect the model to do poorly on longer utterances due to limited number of long training utterances in our distribution. Hence it is not surprising that longer utterances have a larger error rate. The deletions dominate the error for long utterances, suggesting we may be missing out on words. It is surprising that short utterances (e.g., 2 words or less) perform quite poorly. Here, the substitutions and insertions are the main sources of errors, suggesting the model may split words apart.

Figure 3.4 also suggests that our model struggles to generalize to long utterances when trained on a distribution of shorter utterances. It is possible location-based priors may help in these situations as reported by [63].

3.3.4 Word Frequency

We study the performance of our model on rare words. We use the recall metric to indicate whether a word appears in the utterance regardless of position (higher is better). Figure 3.5 reports the recall of each word in the test distribution as a function of the word frequency in the training distribution. Rare words have higher variance and lower recall while more frequent words typically have higher recall. The word "and" occurs 85k times in the training set, however it has a recall of only 80% even after language model rescoring. The word "and" is frequently mis-transcribed as "in" (which has 95% recall). This suggests improvements are needed in the language model. By contrast, the word "walkerville" occurs just once in the training set but it has a recall of 100%. This suggests that the recall for a word depends both on its frequency in the training set and its acoustic uniqueness.

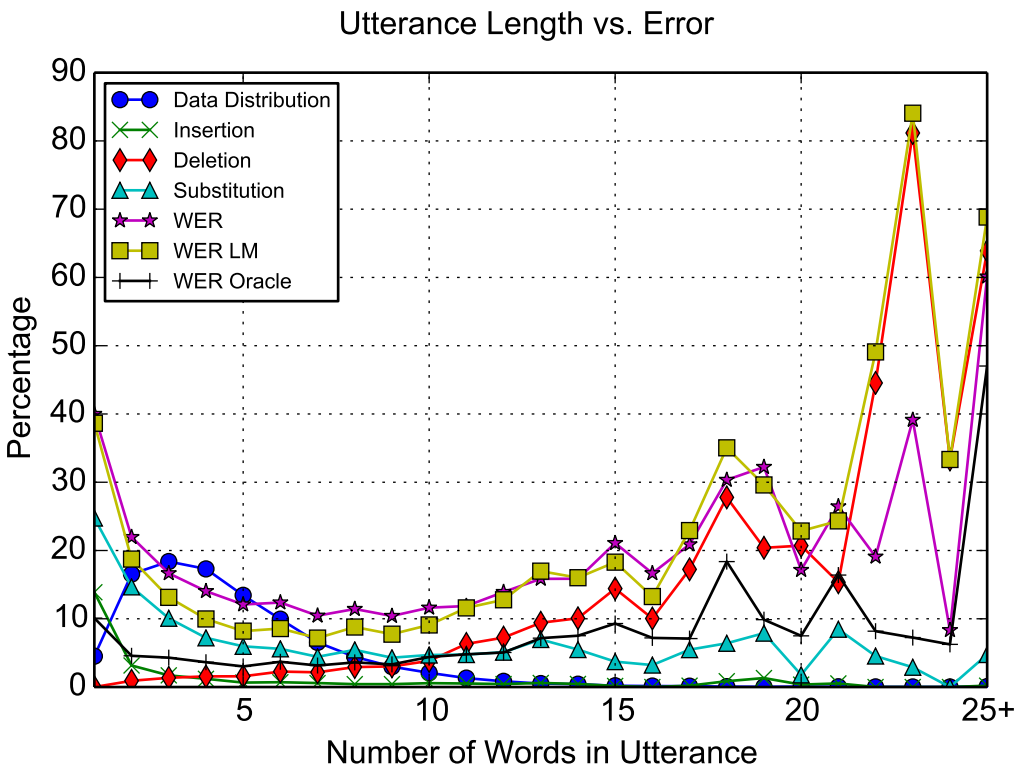

Figure 3.4: The correlation between error rates (insertion, deletion, substitution and WER) and the number of words in an utterance. The WER is reported without a dictionary or language model, with language model rescoring and the oracle WER for the clean Google voice search task. The data distribution with respect to the number of words in an utterance is overlaid in the figure. LAS performs poorly with short utterances despite an abundance of data. LAS also fails to generalize well on longer utterances when trained on a distribution of shorter utterances. Insertions and substitutions are the main sources of errors for short utterances, while deletions dominate the error for long utterances.

3.3.5 Qualitative Analysis

In this section, we show the outputs of the model on several utterances to demonstrate and understand the capabilities of LAS. All the results in this section are decoded without a dictionary or a language model.

During our experiments, we observed that LAS can learn multiple spelling variants given the same acoustics. Table 3.2 shows top hypotheses for the utterance that includes "triple a". As can be seen, the model produces both "triple a" and "aaa" within the top four

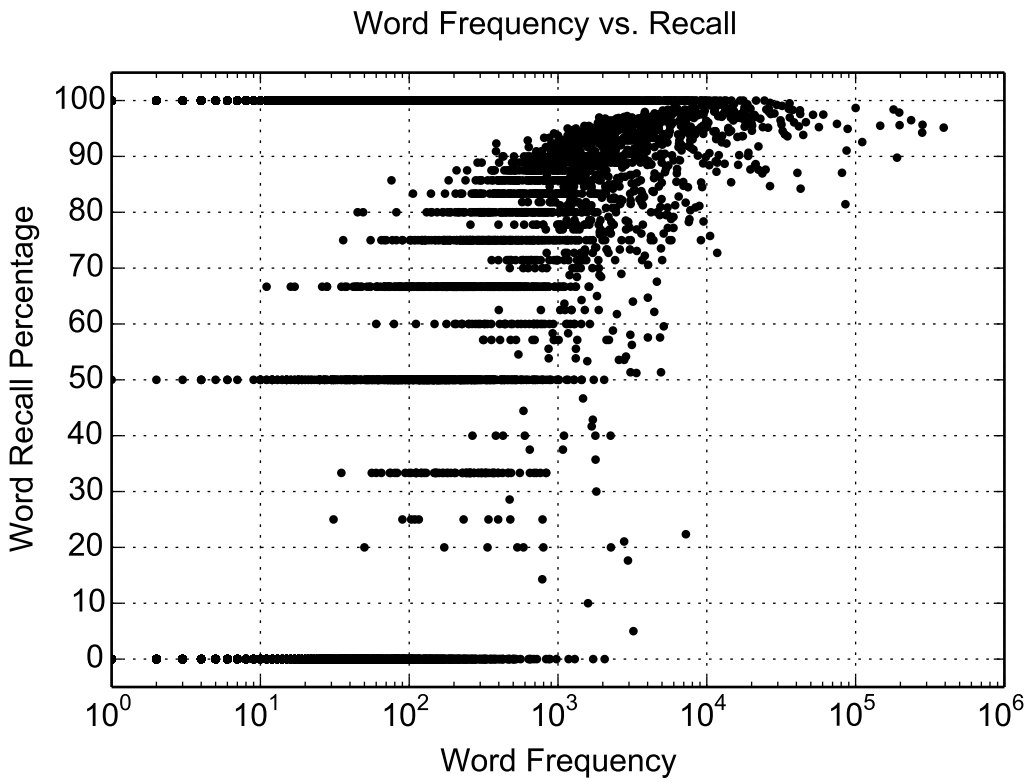

Figure 3.5: The correlation between word frequency in the training distribution and recall in the test distribution. In general, rare words report worse recall compared to more frequent words.

hypotheses. The decoder is able to generate such varied parses, because the next step prediction model makes no assumptions on the probability distribution by using the chain rule decomposition. It would be difficult to produce such differing transcripts using CTC due to the conditional independence assumptions, where $p(y_i|x)$ is conditionally independent of $p(y_{i+1}|x)$. Conventional DNN-HMM systems would require both spellings to be in the pronunciation dictionary to generate both spelling permutations.

It can also be seen that the model produced "xxx" even though acoustically "x" is very different from "a" - this is presumably because the language model overpowers the acoustic signal in this case. In the training corpus "xxx" is a very common phrase and we suspect the language model implicit in the speller learns to associate "triple" with "xxx". We note that "triple a" occurs 4 times in the training distribution and "aaa" (when pronounced "triple a"

Table 3.2: Example 1: “triple a” vs. “aaa” spelling variants.

Beam	**Text**	**Log Probability**	**WER %**
Truth	call aaa roadside assistance	-	-
1	call aaa roadside assistance	-0.5740	0.00
2	call triple a roadside assistance	-1.5399	50.00
3	call trip way roadside assistance	-3.5012	50.00
4	call xxx roadside assistance	-4.4375	25.00
5	call chip away roadside assistance	-4.4898	50.00
6	call aaa woodside assistance	-5.1151	25.00
7	call trip way rhode side assistance	-5.1471	100.00
8	call aaa rhode side assistance	-5.3313	50.00
9	call trip away roadside assistance	-5.6718	50.00
10	call aaa road side assistance	-5.8003	50.00
11	call chip way roadside assistance	-5.8925	50.00
12	call triple a rhode side assistance	-5.9156	100.00
13	call tripoli roadside assistance	-6.3010	25.00
14	call xxx woodside assistance	-6.3451	50.00
15	call triple a road side assistance	-6.6978	100.00
16	call aaa roadside	-7.5878	25.00

rather than “a”-“a”-“a”) occurs only once in the training distribution.

We are also surprised that the model is capable of handling utterances with repeated words despite the fact that it uses content-based attention. Table 3.3 shows an example of an utterance with a repeated word. Since LAS implements content-based attention, it is expected to “lose its attention” during the decoding steps and produce a word more or less times than the number of times the word was spoken. As can be seen from this example, even though “seven” is repeated three times, the model successfully outputs “seven” three times. This hints that location-based priors (e.g., location based attention or location based regularization) may not be needed for repeated contents.

Table 3.4 gives the top hypothesis for the utterance “st mary’s animal clinic”. Note the different parses for “saint” vs “st” and “mary’s” vs “marys”. Once again, this is due to the conditional dependency of our model and the ability of the seq2seq framework to produce multimodal distributions. Since the reference transcript expected “st” instead of “saint”

Table 3.3: Example 2: Repeated "seven"s.

Beam	Text	Log Probability	WER %
Truth	eight nine four minus seven seven seven	-	-
1	eight nine four minus seven seven seven	-0.2145	0.00
2	eight nine four nine seven seven seven	-1.9071	14.29
3	eight nine four minus seven seventy seven	-4.7316	14.29
4	eight nine four nine s seven seven seven	-5.1252	28.57
5	eight nine four ninety seven seven seven	-6.0537	14.29
6	eight nine four nine zero seven seven seven	-6.5026	28.57
7	eight nine four seven seven seven	-6.9038	14.29
8	eight nine four nine a seven seven seven	-7.0844	28.57
9	eight nine four hundred seven seven seven	-7.1271	14.29
10	eight nine four hundred minus seven seven seven	-7.4677	14.29
11	eight nine five minus seven seven seven	-7.4842	14.29
12	eight nine four minos seven seven seven	-7.6443	14.29
13	eight nine four ninez seven seven seven	-7.7647	14.29
14	eight nine four minus seven seven	-7.8402	14.29
15	eight nine four nine seven seventy seven	-7.8692	28.57
16	eight nine four five seven seven seven	-7.8732	14.29

(which in our opinion is equally valid), our model scored 1 substitution error or 25% WER for the utterance. The model learns the text normalization distribution as presented in the training distribution. We note that DNN-HMM systems will typically rely on a handcrafted text normalization rules based model to fix these problems.

3.4 Related Work

CTC has been shown to be an end-to-end speech recognition models going directly from acoustics to text [11]. However, CTC models are limited by the conditional independence and Markovian assumptions in their output tokens (see Chapter 2). Consequently, CTC models must rely on a strong language model during decoding [71, 44, 72, 46, 47, 48, 49]. Even with a deep RNN preceeding layer, CTC models can not learn multimodal distributions that our LAS model can learn (i.e., see Section 3.3.5 above). Each CTC frame output prediction is independent of neighbour prediction.

Table 3.4: Example 2: “saint” vs. “st”, and apostrophe spelling variations.

Beam	**Decoded Text**	**Log Probability**	**WER %**
Truth	st mary’s animal clinic	-	-
1	saint mary’s animal clinic	-0.4844	25.00
2	st mary’s animal clinic	-1.0439	0.00
3	st marys animal clinic	-4.0087	25.00
4	saint marys animal clinic	-4.7760	50.00
5	st maries animal clinic	-8.1787	25.00
6	say mary’s animal clinic	-8.3186	25.00
7	saint mery animal clinic	-8.5515	50.00
8	saint berry’s animal clinic	-9.1439	50.00
9	saint mary animal clinic	-9.1710	50.00
10	st mary’s	-9.2880	50.00
11	st mary’s animal	-9.3984	25.00
12	st marries animal clinic	-9.4590	25.00
13	saint mary’s animal	-9.5272	50.00
14	st mary animal clinic	-9.6920	25.00
15	st marry animal clinic	-9.7798	25.00
16	saint mary is animal clinic	-10.0433	75.00

In practice, CTC models suffer the same problems as HMM based models, we must carry around a n-gram language model. The LAS model we presented in this chapter does not suffer the same assumptions. While we do benefit from the usage of a language model for rescoring our hypothesis, our model still produces competitive results without any language model which is not at all possible with CTC and HMM based models. This is due to the intrinsic language model in the transducer, each output token is conditioned on previously emitted tokens.

The model proposed here was inspired by previous work of [62, 63]. However, this work was only done on TIMIT (which predicts phonemes) rather than characters as expected from an end-to-end speech recognition model. Our model is a true end-to-end speech recognition model as we go directly from acoustics to character outputs without needing to rely on another model to convert the phonemes to words.

We also point the reader to [73, 64]. Both of these models use seq2seq with attention-based

transducer as an end-to-end speech recognition model. Our work and [73, 64] were performed concurrently and independently. Compared to [73], our work involved using a pyramidal RNN in our encoder, which we showed was critical to achieving good model performance. [64] used a hierarchical RNN with skip connections which is similar in principle to our pyramidal RNN. We also used a sampling technique during training to improve generalization which [64] did not use. However, compared to both [73, 64], we were able to show competitive results to a state-of-the-art HMM based model while [73, 64] are still an order of magnitude behind. We attribute this to the much large dataset we used (with 3 million utterances), and consequently our model is able to learn a much stronger intrinsic language model.

Finally, multiple extensions of our LAS model have been recently published following our work. This included online variants [74, 75], hard alignments using REINFORCE [76] and joint CTC-LAS models [49].

3.5 Summary

We have presented Listen, Attend and Spell (LAS), a neural speech recognizer that can transcribe acoustic signals to characters directly without using any of the traditional components of a speech recognition system, such as HMMs, language models and pronunciation dictionaries. We submit that it is not only an *end-to-end trained* system, but an *end-to-end model*. LAS accomplishes this goal by making no conditional independence assumptions about the output sequence using the sequence-to-sequence framework. This distinguishes it from models like CTC, DNN-HMM and other models that can be *trained end-to-end* but make various conditional independence assumptions to accomplish this. We showed how this model learns an implicit language model that can generate multiple spelling variants given the same acoustics. We also showed how an external language model, trained on additional text, can be used to re-rank the top hypotheses. We demonstrated that such an end-to-end model can be trained and be competitive with state-of-the-art CLDNN-HMM systems. We

achieve 14.1% WER without the usage of any language model and 10.3% WER using only language model rescoring, which compares to a well tuned CLDNN-HMM system of 8.0% WER.

Chapter 4

Chinese Mandarin

In the previous chapter, we presented an end-to-end ASR model for offline English Google voice search. The model is offline due to the usage of bidirectional RNNs and the attention mechanism needing to see the entire input sequence first. In this chapter, we focus on making the model online and working with Chinese Mandarin. This chapter is based off the work done in [75].

4.1 Model

We will describe our online attention model in this section. Our model closely relates to end-to-end neural speech recognizer proposed in the previous chapter [14, 77]. The model consists of two components, an encoder and an attention-based decoder, we will first describe the encoder. Let $\mathbf{x} = (x_0, \ldots, x_T)$ be our input audio sequence (e.g., sequence of filter bank spectra). We process the input signal $\mathbf{x}$ into higher level features $\mathbf{h}$ with an unidirectional encoder RNN:

$$\mathbf{h} = \text{RNNEncoder}(\mathbf{x}) \tag{4.1}$$

where RNNEncoder is an unidirectional RNN network. We use a deep GRU [16] network with hierarchical subsampling [66, 14, 77] for the RNNEncoder function. The hierarchical subsampling allows the attention model (described below) to attend to fewer timesteps and help prevent the attention alignment from being diluted. It also helps speed up the training and inference computation since a single frame of h can span over many frames of x. See Figure 4.1 for visualization. The main difference between the encoder of this chapter and the previous chapter is the usage of unidirectional RNNs (rather than bidirectional), and the usage of skip connections rather than pyramidal structure to create the hierarchical RNN (for a more detailed empirical study on the different hierarchical subsampling mechanisms, we defer the reader to [78]). We also used GRUs instead of LSTMs in this chapter (we also defer the reader to [79] for a comparison of different RNN cells).

The decoder network generates a character sequence $\mathbf{y} = (y_0, \ldots, y_U)$ with an attention-based RNN:

$$s_j = \text{RNNDecoder}(y_{j-1}, s_{j-1}, c_{j-1}) \tag{4.2}$$

where s_j is the state of the decoder RNN and RNNDecoder is a GRU network in our experiments. The context c_j is generated by an AttentionContext mechanism from the RNN decoder state s_j and some encoder features $\mathbf{w}_j$:

$$c_j = \text{AttentionContext}(s_j, \mathbf{w}_j) \tag{4.3}$$

For the AttentionContext function, we use a MLP attention mechanism [17]. First, we compute the energies $e_{i,j}$, then the normalized alignments $\alpha_{i,j}$ between the decoder state s_j and encoder features $\mathbf{w}_j$. The context c_j is the weighted bag of features over $\mathbf{w}_j$ using the

alignments α as the weights:

$$e_{j,i} = \langle v, \tanh(\phi(w_i) + \psi(s_j) + b) \rangle \tag{4.4}$$

$$\alpha_{j,i} = \frac{\exp(e_{j,i})}{\sum_i \exp(e_{j,i})} \tag{4.5}$$

$$c_j = \sum_i \alpha_{j,i} w_i \tag{4.6}$$

where ϕ and ψ are MLP networks and v, b are weight vectors. The main difference between this attention mechanism and the previous chapter is that we use a MLP network to generate the attention energies as opposed to using the dot product (for a more detailed empirical study on the different attention energy mechanisms, we defer the reader to [74]).

If $\mathbf{w}_j = \mathbf{h}$ [17, 14], then we have our standard attention-based seq2seq model as described in the previous chapter, and the model is not decodable online as we need to wait for the entire input acoustic signal to be seen before we can begin decoding. We follow an approach similar to [77] and use a sliding window, we use the median of the previous alignment α_{j-1} to create a sliding window $\mathbf{w}_j$:

$$m_j = \text{median}(\alpha_{j-1}) \tag{4.7}$$

$$\mathbf{w}_j = \{h_{m_j-p}, \ldots, h_{m_j+q}\} \tag{4.8}$$

where m_j is the median of the previous alignment, and p, q are the hyperparameters to our window size. In our experiments we set $p = 100$ and $q = 10$. This means, we can start decoding (and continue decoding) as long as the we have up till $m_j + q$ frames of h signal available (since the RNNEncoder function is unidirectional).

The model produces a conditional distribution as a function over all previously emitted

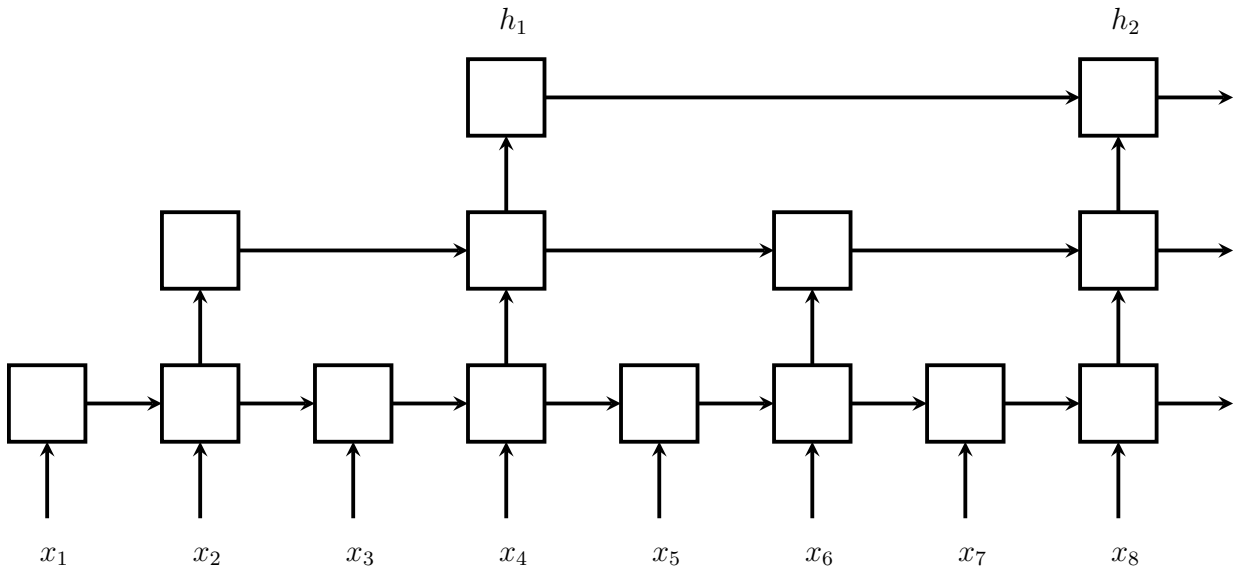

Figure 4.1: Hierarchical recurrent neural network: we subsample the inputs $\mathbf{x}$ to reduce the time dimension into higher level features $\mathbf{h}$.

characters and the sliding window of acoustic features $\mathbf{w}_j$:

$$p(y_j|\mathbf{w}_j, y_{<j}) = \text{TokenDistribution}(\varphi(s_j, c_j)) \tag{4.9}$$

where φ is a MLP. We note that our model will likely have difficulties predicting the next token if there is a large gap or silence between characters, for example if there was silence for more than q frames. We however found this to not be a problem with the datasets we experiment with. We also note the model in Equation 4.9 is non-Markovian, meaning we can learn the language directly (which is much harder for CTC systems due to its conditional independence assumption). However (as noted by [14]), the implicit language model learnt is limited by the number of transcribed transcripts unlike n-gram or RNN LMs which can leverage on any text data. See Figure 4.2 for visualization of our sliding window model.

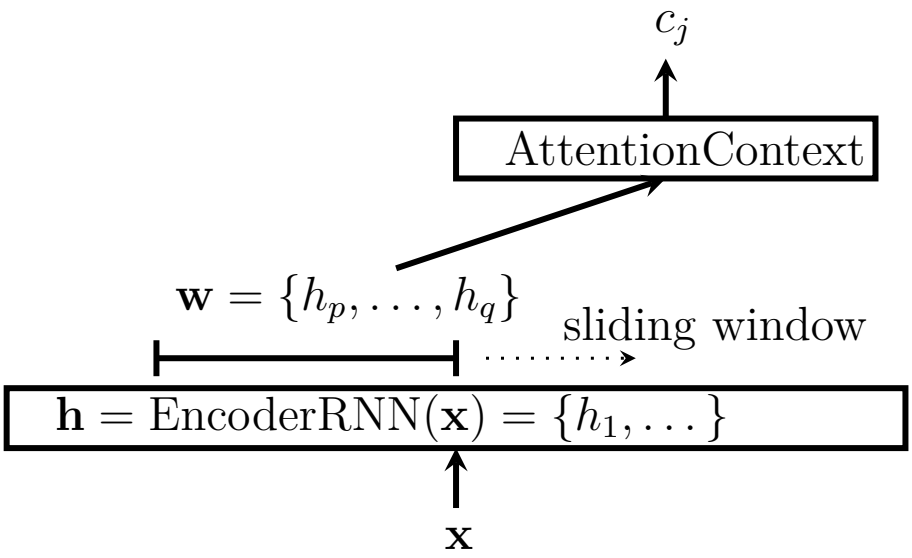

Figure 4.2: Online Attention: The acoustic signal $\mathbf{x}$ is processed online by an unidirectional RNN into features $\mathbf{h}$. A sliding window $\mathbf{w}_j$ as a function of the previous alignment moves across $\mathbf{h}$ to be processed by the attention AttentionContext mechanism for the next alignment.

4.2 Optimization

We follow the procedure described in the previous chapter. We optimize the parameters of our model to maximize the log probability of the correct sequences:

$$\theta = \max_{\theta} \sum_{j} \log p(y_j | \mathbf{w}_j, \tilde{y}_{<j}; \theta) \tag{4.10}$$

where $\tilde{y}_{i-1}$ is the ground truth previous character or a character randomly sampled (with 10% probability) from the model using the sampling procedure as described in the previous chapter. We found the sampling technique important to help reduce overfitting.

4.3 Decoding

Unlike CTC systems, neural attention models can learn the language of our output tokens due to the conditional dependency of previously emitted symbols $y_{<j}$ in Equation 4.9. As

observed in the previous chapter and in [77], while LMs and beam search can improve results, the improvement is small relative to CTC and HMM based systems. Thus, in this chapter we focus on experiments that do not use any LMs or searching. We simply take the greedy path of our conditional model in Equation 4.9:

$$\hat{y}_j = \arg\max_{y_j} p(y_j|\mathbf{w}_j, y_{<j}) \tag{4.11}$$

We believe many applications will benefit from models that do not use searching, for example on embedded platforms without GPUs and limited computation and memory capacity.

4.4 Joint Mandarin Character-Pinyin Model

We found the attention model difficult to converge with Mandarin data. The attention mechanism has a difficult time learning the alignment and/or the decoder overfits to the transcripts before the acoustic model encoder converges. We suspect this is due to Mandarin using a logographic orthography and the Chinese characters give limited information on the sounds of the spoken language. Additionally, the large vocabulary and the conditional dependency of our model (unlike CTC which is conditionally independent [45, 80]) makes learning a generalized model more difficult.

Mandarin characters can be readily transcribed to Pinyin, which is the romanization and a (rough) phonetic representation with English characters using a Mandarin Pinyin dictionary. The cost to transform the Mandarin characters to Pinyin is minimal and only consists of a lookup. While we could build our ASR system to directly output Pinyin, such a system is non-ideal as we would require another system to convert the Pinyin (with possible misspellings) back into the Mandarin characters. We want our model to leverage the phonetic Pinyin information readily available while still emitting Mandarin characters as our output.

We adapt our network architecture to jointly learn the Character and Pinyin transcriptions,

and discard the Pinyin paths during inference (to be explicit, our model still only conditions on the Mandarin characters and never conditions on the Pinyin). We extract the Pinyin transcriptions using a dictionary [81] (without the tones). Each Mandarin character has a phonetic Pinyin with a maximum Pinyin character length of 7. We pad each Pinyin representation to be exactly length 7. Thus, for each Mandarin character y_j, we can lookup its Pinyin transcription $\mathbf{z}_j = \text{PinyinDictionary}(y_j)$ with each vector $\mathbf{z}_j = (z_1, \ldots, z_7)$. For Out-of-Vocabulary (OOV) words in the Pinyin dictionary, we use the original character $z_{j,1} = y_j$. The Pinyin dictionary [81] models bigram of Mandarin characters, we thus greedily use these transcriptions if available. Additionally, the dictionary may give multiple Pinyin transcriptions for a Mandarin character, in this case we simply use the first transcription in the dictionary.

We model the Pinyin characters with additional MLP networks as a function of the RNNDecoder and attention states:

$$\log p(\mathbf{z}_j|\mathbf{w}_j, y_{<j}) = \sum_k \log \text{TokenDistribution}(\varphi_k(s_j, c_j)) \tag{4.12}$$

where y_j are the Mandarin characters and φ_k are MLPs modelling the Pinyin characters.

We jointly optimized the model to learn the Mandarin character and the Pinyin representation. We do not condition on the Pinyin characters, thus we only use the Pinyin information during training. During inference we discard the Pinyin outputs and only read off the Mandarin softmax outputs. The joint Mandarin Character-Pinyin model optimization objective is the joint probability of the Mandarin characters y_j and Pinyin $\mathbf{z}_j$:

$$\max_\theta \sum_i \log p(y_j|\mathbf{w}_j, \tilde{y}_{<i}; \theta) + \log p(\mathbf{z}_j|\mathbf{w}, \tilde{y}_{<i}; \theta) \tag{4.13}$$

The motivation of providing the Pinyin information is not to give the model perfect pronunciation information, but rather to give it some (possibly noisy) information on how each

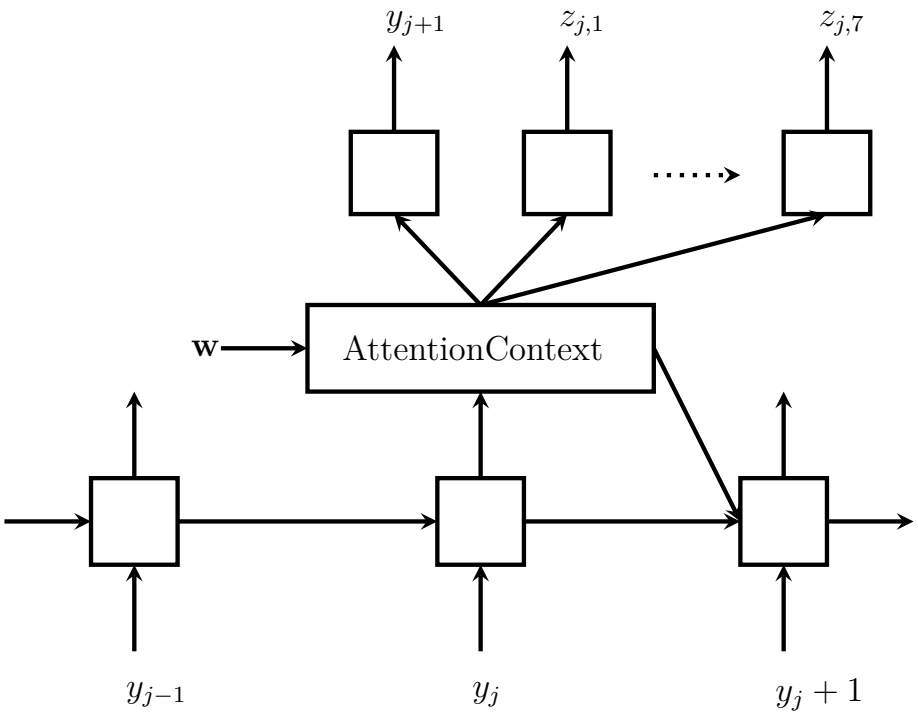

Figure 4.3: Joint Mandarin Character-Pinyin decoder: we model the Mandarin character y_j and Pinyin $\mathbf{z}_j = (z_{j,1}, \ldots, z_{j,7})$ together in the decoder RNN. The sliding window $\mathbf{w}$ of encoder features is from Figure 4.2. The AttentionContext function consumes the window $\mathbf{w}$ to produce a context to emit the character or Pinyin outputs and update the RNN state.

Mandarin character is pronounced. We hypothesize that this additional pronunciation information can be backpropagated to the encoder and learn a more generalized model. Without the additional Pinyin information, we hypothesize the decoder would very easily overfit to the training transcripts without using the acoustics. We leverage on the explicit alignment (as opposed to implicit alignment of CTC) to inject this additional pronunciation information into the model. We also hypothesize that if we had sufficient data, we would not need this technique [43, 14].

4.5 Experiments

We experimented with English Wall Street Journal (WSJ) and GALE Mandarin datasets. We tried to use similar hyperparameters and models between the two datasets, however there are some minor differences as described below.

First, we used AdaDelta [82], we found it converges much more quickly than Stochastic Gradient Descent (SGD) even with acceleration [83]. We used AdaDelta with $\rho = 0.95$ and $\epsilon = 1e-8$ to optimize our model, however we may lower the value of ϵ as training progresses and described in the later subsections below. Despite AdaDelta's decay, we found resetting (i.e., zero out) the AdaDelta accumulators $\mathbb{E}[g^2]$ and $\mathbb{E}[\Delta x^t]$ after each epoch to help with the optimization. We clipped the gradients to have a maximum norm of 1.

We initialized all the weighted connection matrices of our model with uniform distribution $\mathcal{U}(-0.1, 0.1)$, and for our embedding matrix we used a uniform distribution $\mathcal{U}(-\sqrt{3}, \sqrt{3})$. We initialized the GRU update and reset gate bias to one and all other biases to zero. We imposed a max column norm of 1 for all our weight matrices after each update [84].

If our EncoderRNN is a hierarchical RNN, then our model may overfit towards the subsampling permutations of our architecture. We follow an approach typically found in computer vision [85], we randomly add up to f frames of input delay to our acoustic signal, where f is the subsampling factor of the EncoderRNN (e.g., $f = 4$ for English WSJ).

4.5.1 Wall Street Journal

We first ran experiments on Wall Street Journal (WSJ) (available as LDC93S6A and LDC94S13A). We used the si284 as the training set, dev93 as the validation set and eval92 as the test set. We observe the WER of the validation set after epoch and stop training when the validation WER no longer improves. We used 40 dimensional filterbanks with energies, delta and delta-delta coefficients (total 123 dimensional features). We follow [11] in text normalization, our token output space consists of the English alphabets, space and punctuations.

We used 384 GRU cells for the encoder, with 3 layers, and the last two layers being hierarchical subsampling layers with a factor of two (i.e., in total the encoder reduces the frame rate by $4 = 2^2$). The decoder is a 2 layer attention-based RNN with 256 GRU cells. We trained the model for around 30 epochs, after 10 epochs we manually lowered the AdaDelta

Table 4.1: Wall Street Journal WERs: We train end-to-end WSJ models without LMs or searching. Compared to online CTC, our attention model has a 14% relative reduction in WER.

Model	WER %
Hybrid-HMM Models	
DNN-HMM [3]	3.8
RNN-HMM [14]	3.5
Offline Character Models	
CTC (Graves et al., 2014) [11]	30.1
CTC (Hannun et al., 2014) [86]	35.8
seq2seq + Attention + Conv (Bahdanau et al., 2016) [77, 87]	21.3
seq2seq + Attention + TLE (Bahdanau et al., 2015) [87]	18.8
seq2seq + Attention + Noise (Chan et al., 2016) [88]	14.8
Online Character Models	
CTC (Hwang et al., 2016) [72]	38.4
Online Attention (this work)	33.0

ϵ several times to a final value of $\epsilon = 1e - 15$.

Table 4.1 gives the WER of our online attention model as well as a comparison to several other models. First, our online attention model achieves a 33.0% WER without LMs or searching. This compares to 21.3% WER without LMs or searching trained on cross entropy or 18.8% WER trained with Task Loss Estimation (TLE), both of which are offline models [87]. It is clear the offline networks perform much better for this task, we also note that [77, 87] has a convolutional location-based attention mechanism which our model does not have. The offline seq2seq model with attention achieves 14.8% WER is with variational noise which we did not have in this set of experiments [88]. Our online attention model however does outperform online CTC models [72] which achieved 38.4% WER or a 5.4% absolute improvement.

However, compared to the state-of-the-art HMM models [14] (with trigram LMs), the online end-to-end ASR models are still significantly behind. We believe more data would ameliorate this issue, as the attention-based end-to-end models are much more powerful and overfit much

more easily [14].

4.5.2 GALE Mandarin

We also experimented with the GALE Phase 2 Chinese Broadcast News Speech (LDC2013S08 and LDC2013T20). We used the exact same train (about 104 hrs) and test (about 6 hrs) split as in the Kaldi s5 recipe [3]. We used 40 dimensional filterbanks with energies as our input features. For the encoder, we used 384 GRU cells with 2 layers. We used hierarchical subsampling with a factor of 2 for the second layer only. We also used "super-frames" [89] stacking 16 frames with a stride of 4. The combination of the subsampling and superframes greatly decreased our encoder feature dimensions and consequently our training time as well. The decoder is a 1 layer attention-based RNN with 256 GRU cells. We trained the model for approximately 20 epochs until convergence.

Table 4.2 gives an overview of our experimental results in CER. We achieved a 72% CER in our online end-to-end Mandarin model, which compares to 18.5% CER for a Kaldi DNN-HMM system. We spent considerable effort in hyperparameter tuning including optimization and regularization hyperparmeters, however we were unable to improve the results of the Mandarin only model. We suspect the large vocabulary (around 4000 Chinese characters) with relatively limited training data [14] made the mode difficult to learn and generalize well.

We trained a joint Character-Pinyin model and the benefits of adding the Pinyin information into the training process was obvious. We achieved a CER of 59.3% for our joint Mandarin Character-Pinyin or 12.7% absolute improvement over the character only model. Mandarin characters generally have limited phonetic information (i.e., the strokes), especially since our input is Chinese characters (as opposed to the strokes or pixels of the characters) whereas the Pinyin transcription more closely matches the phonetic representation of Mandarin speech. The neural network model is able to leverage this additional phonetic information. While

Table 4.2: GALE Mandarin CERs: We train end-to-end Mandarin models without LMs or searching. The Pinyin model uses the Pinyin phonetic information during training, but discards it during inference.

Model	CER %
DNN-HMM [90]	18.5
Online Attention Character	72.0
Online Attention Character + Pinyin	59.3

we do use the additional Pinyin information during training, our model does not condition on the Pinyin and thus has the exact same model architecture (and inference cost) as the Mandarin Character only model.

Finally, without any attempt on minimizing the model size (e.g., SVD, projection layers or quantization [91, 92]) we note that both English and Mandarin models are less than 64 MiB in size. We believe future research in this area can lead to extremely compact ASR models applicable for embedded devices.

4.6 Related Work

Recently there has been much interest in end-to-end ASR models for Chinese Mandarin [45, 46]. However, other work has focused on CTC based methods. CTC may actually be more ideal for Chinese Mandarin compared to English. The mapping between acoustics to Chinese characters is many-to-one, while the mapping from acoustics to English characters is many-to-many. This property yields less advantage of the reordering abilities of the seq2seq model (as shown in the previous chapter). Additionally, Chinese may not benefit as much from a language model, as the language has much less long term dependencies compared to English characters [80]. Future work should involve a detailed study of the two models on the same dataset, as most publications have used different Chinese speech datasets [45, 80, 46].

It should be noted our joint Character-Pinyin training methodology can not be applied to CTC directly. While it is possible to optimize for both sequences using separate CTC layers, it is intractable to learn the Pinyin per Chinese character. This is due to the implicit alignment generated by the CTC model is latent, an approximate inference algorithm would be needed. This could be seen as a benefit to an explicit alignment model such as seq2seq.

4.7 Summary

In this chapter, we presented an online end-to-end speech recognition model based on attention-based neural networks. Our models do not rely on any pronunciation dictionaries, language models or searching during inference. On English, our model achieves a WER of 33.0% on WSJ, or an 5.4% absolute improvement over an online end-to-end CTC system [72]. On Chinese Mandarin models, we show how to build a joint Character-Pinyin model, combining the available Pinyin transcriptions to our end-to-end character model. The joint model does not condition on the extra Pinyin information and is only used during training. For the GALE Phase 2 Chinese Broadcast News Speech evaluation, we achieved a CER of 72.0% for our character only model. With our joint Character-Pinyin model, we achieve a CER of 59.3% or a 12.7% CER improvement over the character model only.

Chapter 5

Latent Sequence Decompositions

In the previous chapters, we assumed a fixed deterministic decomposition for each output sequence. For example, in Chapter 3 our output sequence was a sequence of English characters, and in Chapter 4 our output sequence was a sequence of Chinese Mandarin characters. In this chapter, we emit word pieces rather (as opposed to English characters), and we learn multiple decompositions for the same sequence. We will show state-of-the-art results on the Wall Street Journal task. This chapter is based off of publications in [88, 93].

5.1 Motivation

Previous seq2seq work has assumed a fixed deterministic decomposition for each output sequence. The output representation could be a fixed sequence of words [15, 16], phonemes [63], characters [14, 64] or even a mixture of characters and words [57]. However, in all these cases, the models are trained towards one fixed decomposition for each output sequence.

We argue against using fixed deterministic decompositions of a sequence defined a priori. Word segmented models [50, 51] often have to deal with large softmax sizes, rare words and Out-of-Vocabulary (OOV) words. Character models [14, 64] overcome the OOV problem by modelling the smallest output unit, however this typically results in long decoder lengths and

expensive inference runtimes. And even with mixed (but fixed) character-word models [57], it is unclear whether such a predefined segmentation is optimal. In all these examples, the output decomposition is only a function of the output sequence, but shouldn't the output decomposition be a function of the input sequence as well?

We want our model to have the capacity and flexibility to learn a distribution of sequence decompositions. Additionally, the decomposition should be a sequence of variable lengthed chunks as deemed most probable. For example, language may be more naturally represented as word pieces [94, 95] rather than individual characters. In many speech and language tasks, it is probably more efficient to model "qu" as one output unit rather than "q" + "u" as separate output units (since in English, "q" is almost always followed by "u"). Word piece models also naturally solve rare word and OOV problems similar to character models. Evidence has been shown in [95] that word piece decompositions may be better than character or word decompositions in Machine Translation (MT).

The output sequence decomposition should be a function of both the input sequence and the output sequence (rather than output sequence alone). For example, in speech, the choice of emitting "ing" as one word piece or as separate tokens of "i" + "n" + "g" should be a function of the current output word as well as the audio signal (i.e., speaking style).

We present the Latent Sequence Decompositions (LSD) framework. LSD does not assume a fixed decomposition for an output sequence, but rather we learn to decompose sequences as function of both the input and output sequence. Each output sequence can be decomposed to a set of latent sequence decompositions using variable lengthed units. The LSD framework produces a distribution over the latent sequence decomposition space and marginalizes over them during training. During test inference, we can do approximate inference and a beam search decoding algorithm is presented.

5.2 Learning the Decompositions

In this section, we will describe and formalize the LSD framework. Let $\mathbf{x}$ be our input sequence, $\mathbf{y}$ be our output sequence and $\mathbf{z}$ be some latent sequence decomposition of $\mathbf{y}$. The latent sequence decomposition $\mathbf{z}$ consists of a sequence of $z_i \in \mathcal{Z}$ where $\mathcal{Z}$ is the constructed token space. Each token z_i need not be the same length, but rather in our framework, we expect the tokens to have different lengths; for example, $\mathcal{Z} \subseteq \mathcal{C}^n$ where $\mathcal{C}$ is the set of smallest output unit and n is the cartesian power. In ASR , $\mathcal{C}$ would typically be the set of English characters, while $\mathcal{Z}$ would be word pieces (i.e., n-grams of characters). Consequently, a latent sequence decomposition length $|\mathbf{z}_a|$ need not to be the same length as $|\mathbf{y}|$, nor with another latent sequence decomposition $|\mathbf{z}_b|$. We also define a collapsing function $\mathbf{y} = \text{collapse}(\mathbf{z})$ when some decomposition $\mathbf{z}$ collapses to $\mathbf{y}$. For any non-trivial $\mathbf{y}$ and $\mathcal{Z}$, the set of valid decompositions $\{\mathbf{z} : \text{collapse}(\mathbf{z}) = \mathbf{y}\}$ has an exponential number of permutations.

If we knew the best possible segmentation $\mathbf{z}^*$ for a given $(\mathbf{x}, \mathbf{y}^*)$ pair, we could just train towards the fixed deterministic decomposition $\mathbf{z}^*$ and be done. However, in most problems, we don't know the best possible $\mathbf{z}^*$ decomposition. For example, in end-to-end ASR we typically use characters as the output unit of choice [14, 64]. However, word pieces might be better units as they more closely align to the acoustics, yet we usually don't know the best $\mathbf{z}^*$ decomposition of word pieces for a given $(\mathbf{x}, \mathbf{y}^*)$ pair. Given a particular $\mathbf{y}^*$, the best $\mathbf{z}^*$ could even change depending on the input sequence $\mathbf{x}$ (i.e., speaking style).

We want to learn a probabilistic segmentation mapping from $\mathbf{x} \rightarrow \mathbf{z} \rightarrow \mathbf{y}$. The LSD framework produces a distribution of decompositions $\mathbf{z}$ given an input sequence $\mathbf{x}$, and the objective is to maximize the log-likelihood of the ground truth sequence $\mathbf{y}^*$. We can accomplish this by factorizing and marginalizing over all possible $\mathbf{z}$ latent sequence decompositions under

our model $p(\mathbf{z}|\mathbf{x};\theta)$ with parameters θ:

$$\log p(\mathbf{y}^*|\mathbf{x};\theta) = \log \sum_{\mathbf{z}} p(\mathbf{y}^*, \mathbf{z}|\mathbf{x};\theta) \tag{5.1}$$

$$= \log \sum_{\mathbf{z}} p(\mathbf{y}^*|\mathbf{z},\mathbf{x}) p(\mathbf{z}|\mathbf{x};\theta) \tag{5.2}$$

$$= \log \sum_{\mathbf{z}} p(\mathbf{y}^*|\mathbf{z}) p(\mathbf{z}|\mathbf{x};\theta) \tag{5.3}$$

where $p(\mathbf{y}^*|\mathbf{z}) = \mathbb{1}(\text{collapse}(\mathbf{z}) = \mathbf{y}^*)$ captures path decompositions $\mathbf{z}$ that collapses to $\mathbf{y}^*$. Due to the exponential number of decompositions of $\mathbf{y}$, exact inference and search is intractable for any non-trivial $\mathcal{Z}$ output spaces and $|\mathbf{z}|$ sequence lengths. We describe a beam search algorithm to do approximate inference decoding in Section 3.2.4.

Similarly, computing the exact gradient is intractable. However, we can derive a gradient estimator by differentiating w.r.t. to θ and taking its expectation:

$$\frac{\partial}{\partial\theta} \log p(\mathbf{y}^*|\mathbf{x};\theta) = \frac{1}{p(\mathbf{y}^*|\mathbf{x};\theta)} \frac{\partial}{\partial\theta} \sum_{z} p(\mathbf{y}^*|\mathbf{z}) p(\mathbf{z}|\mathbf{x};\theta) \tag{5.4}$$

$$= \frac{1}{p(\mathbf{y}^*|\mathbf{x};\theta)} \sum_{z} p(\mathbf{y}^*|\mathbf{z}) \nabla_\theta p(\mathbf{z}|\mathbf{x};\theta) \tag{5.5}$$

$$= \frac{1}{p(\mathbf{y}^*|\mathbf{x};\theta)} \sum_{z} p(\mathbf{y}^*|\mathbf{z}) p(\mathbf{z}|\mathbf{x};\theta) \nabla_\theta \log p(\mathbf{z}|\mathbf{x};\theta) \tag{5.6}$$

$$= \frac{1}{p(\mathbf{y}^*|\mathbf{x};\theta)} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x};\theta)} \left[p(\mathbf{y}^*|\mathbf{z}) \nabla_\theta \log p(\mathbf{z}|\mathbf{x};\theta) \right] \tag{5.7}$$

Equation 5.6 uses the identity $\nabla_\theta f_\theta(x) = f_\theta(x) \nabla_\theta \log f_\theta(x)$ assuming $f_\theta(x) \neq 0 \; \forall \; x$. Also, noting that $p(\mathbf{y}^*|\mathbf{z}) = \mathbb{1}(\text{collapse}(\mathbf{z}) = \mathbf{y}^*)$, consequently we can change our expectation in Equation 5.7 to sample only valid $\mathbf{z}$ decompositions and scale the gradient accordingly by the ratio of correct decompositions $p(\mathbf{y}^*|\mathbf{x};\theta)$:

$$= \frac{1}{p(\mathbf{y}^*|\mathbf{x};\theta)} p(\mathbf{y}^*|\mathbf{x};\theta) \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x};\theta)|\mathbf{z}\in\{\mathbf{z}':p(\mathbf{y}^*|\mathbf{z}')=1\}} \left[\nabla_\theta \log p(\mathbf{z}|\mathbf{x};\theta) \right] \tag{5.8}$$

$$= \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x};\theta)|\mathbf{z}\in\{\mathbf{z}':\text{collapse}(\mathbf{z}')=\mathbf{y}^*\}} \left[\nabla_\theta \log p(\mathbf{z}|\mathbf{x};\theta) \right] \tag{5.9}$$

Equation 5.9 gives us an unbiased estimator of our gradient. It tells us to sample some latent sequence decomposition $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x};\theta)$ under our model's posterior such that $\mathbf{z} \in \{\mathbf{z}' : \text{collapse}(\mathbf{z}') = \mathbf{y}^*\}$ where $\mathbf{z}$ is a valid sequence that collapses to $\mathbf{y}^*$ and backprop its derivative $\nabla_\theta \log p(\mathbf{z}|\mathbf{x};\theta)$. In our implementation, the model $p(\mathbf{z}|\mathbf{x})$ can be decomposed with the chain rule (i.e., seq2seq models). We use ancestral sampling in a left-to-right fashion costing $O(n)$ runtime at each output step, where n is the length of the longest token in $\mathcal{Z}$. This left-to-right sampling procedure is computationally efficient but biased towards longer decompositions.

In practice, we find that the biased gradient without any entropy regularization or exploration techniques, the model will quickly collapse and never escape from a local minima. In our ASR experiments, we found the model to collapse to the character distribution and almost never emit any ($n > 2$) sized word pieces (where n is the n-gram of characters). However, one notable exception is the word piece "qu" ("u" is almost always followed by "q" in English). The model does learn to consistently emit "qu" as one token and never produce "q" + "u" as separate tokens. We ameliorate this issue by adding an ϵ-greedy exploration strategy commonly found in reinforcement learning literature [96].

5.3 Model

In this work, we model the latent sequence decompositions $p(\mathbf{z}|\mathbf{x})$ with an attention-based seq2seq model [17]. Each output token z_i is modelled as a conditional distribution over all previously emitted tokens $\mathbf{z}_{<i}$ and the input sequence $\mathbf{x}$ using the chain rule:

$$p(\mathbf{z}|\mathbf{x};\theta) = \prod_i p(z_i|\mathbf{x}, \mathbf{z}_{<i}) \tag{5.10}$$

The input sequence $\mathbf{x}$ is processed through an EncodeRNN network. The EncodeRNN function transforms the features $\mathbf{x}$ into some higher level representation $\mathbf{h}$. In our experimental implementation EncodeRNN is a stacked Bidirectional LSTM (BLSTM) [97, 65] with hier-

archical subsampling [66, 67]:

$$\mathbf{h} = \text{EncodeRNN}(\mathbf{x}) \tag{5.11}$$

The output sequence $\mathbf{z}$ is generated with an attention-based transducer [17] one z_i token at a time:

$$s_i = \text{DecodeRNN}([z_{i-1}, c_{i-1}], s_{i-1}) \tag{5.12}$$

$$c_i = \text{AttentionContext}(s_i, \mathbf{h}) \tag{5.13}$$

$$p(z_i|\mathbf{x}, \mathbf{z}_{<i}) = \text{TokenDistribution}(s_i, c_i) \tag{5.14}$$

The DecodeRNN produces a transducer state s_i as a function of the previously emitted token z_{i-1}, previous attention context c_{i-1} and previous transducer state s_{i-1}. In our implementation, DecodeRNN is a LSTM [42] function without peephole connections.

The AttentionContext function generates c_i with a content-based MLP attention network [17]. Energies e_i are computed as a function of the encoder features $\mathbf{h}$ and current transducer state s_i. The energies are normalized into an attention distribution α_i. The attention context c_i is created as a α_i weighted linear sum over $\mathbf{h}$:

$$e_{i,j} = \langle v, \tanh(\phi(s_i, h_j)) \rangle \tag{5.15}$$

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j'} \exp(e_{i,j'})} \tag{5.16}$$

$$c_i = \sum_j \alpha_{i,j} h_j \tag{5.17}$$

where ϕ is linear transform function. TokenDistribution is a MLP function with softmax outputs modelling the distribution $p(z_i|\mathbf{x}, \mathbf{z}_{<i})$.

5.4 Decoding

During inference we want to find the most likely word sequence given the input acoustics:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \sum_{\mathbf{z}} \log p(\mathbf{y}|\mathbf{z}) p(\mathbf{z}|\mathbf{x}) \tag{5.18}$$

however this is obviously intractable for any non-trivial token space and sequence lengths. We simply approximate this by decoding for the best word piece sequence $\hat{\mathbf{z}}$ and then collapsing it to its corresponding word sequence $\hat{\mathbf{y}}$:

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} \log p(\mathbf{z}|\mathbf{x}) \tag{5.19}$$

$$\hat{\mathbf{y}} = \text{collapse}(\hat{\mathbf{z}}) \tag{5.20}$$

We approximate for the best $\hat{\mathbf{z}}$ sequence by doing a left-to-right beam search similar to Chapter 3 and Chapter 4. We found this crude approximation to work. However, we believe future work should improve on this search process.

5.5 Experiments

We experimented with the Wall Street Journal (WSJ) ASR task. We used the standard configuration of train si284 dataset for training, dev93 for validation and eval92 for test evaluation. Our input features were 80 dimensional filterbanks computed every 10ms with delta and delta-delta acceleration normalized with per speaker mean and variance as generated by Kaldi [3]. The EncodeRNN function is a 3 layer BLSTM with 256 LSTM units per-direction (or 512 total) and $4 = 2^2$ time factor reduction. The DecodeRNN is a 1 layer LSTM with 256 LSTM units. All the weight matrices were initialized with a uniform distribution $\mathcal{U}(-0.075, 0.075)$ and bias vectors to 0. Gradient norm clipping of 1 was used, Gaussian weight noise $\mathcal{N}(0, 0.075)$ and L2 weight decay 1e$-$5 [98]. We used ADAM with the

Table 5.1: Wall Street Journal test eval92 Word Error Rate (WER) varying the n sized word piece vocabulary without any dictionary or language model. We compare Latent Sequence Decompositions (LSD) versus the Maximum Extension (MaxExt) decomposition. The LSD models all learn better decompositions compared to the baseline character model, while the MaxExt decomposition appears to be sub-optimal.

n	**LSD WER %**	**MaxExt WER %**
Baseline	14.76	
2	13.15	15.56
3	13.08	15.61
4	**12.88**	14.96
5	13.52	15.03

default hyperparameters described in [99], however we decayed the learning rate from 1e−3 to 1e−4. We used 8 GPU workers for asynchronous SGD under the TensorFlow framework [100]. We monitor the dev93 Word Error Rate (WER) until convergence and report the corresponding eval92 WER. The models took $O(5)$ days to converge.

We created our token vocabulary $\mathcal{Z}$ by looking at the n-gram character counts of the training dataset. We explored $n \in \{2, 3, 4, 5\}$ and took the top $\{256, 512, 1024\}$ tokens based on their count frequencies (since taking the full n-cartesian exponent of the unigrams would result in an intractable number of tokens for $n > 2$). We found very minor differences in WER based on the vocabulary size, for our $n = \{2, 3\}$ word piece experiments we used a vocabulary size of 256 while our $n = \{4, 5\}$ word piece experiments used a vocabulary size of 512. Additionally, we restrict $\langle$space$\rangle$ to be a unigram token and not included in any other word pieces, this forces the decompositions to break on word boundaries. We decoded with the left-to-right beam search described in Section 5.4. We kept $n = 16$ best hypothesis in the search process.

Table 5.1 compares the effect of varying the n sized word piece vocabulary. The Latent Sequence Decompositions (LSD) models were trained with the framework described in Section 5.2 and the Maximum Extension (MaxExt) models were trained using a greedy left-to-right longest word piece decomposition. The MaxExt decomposition is not the shortest $|\mathbf{z}|$ pos-

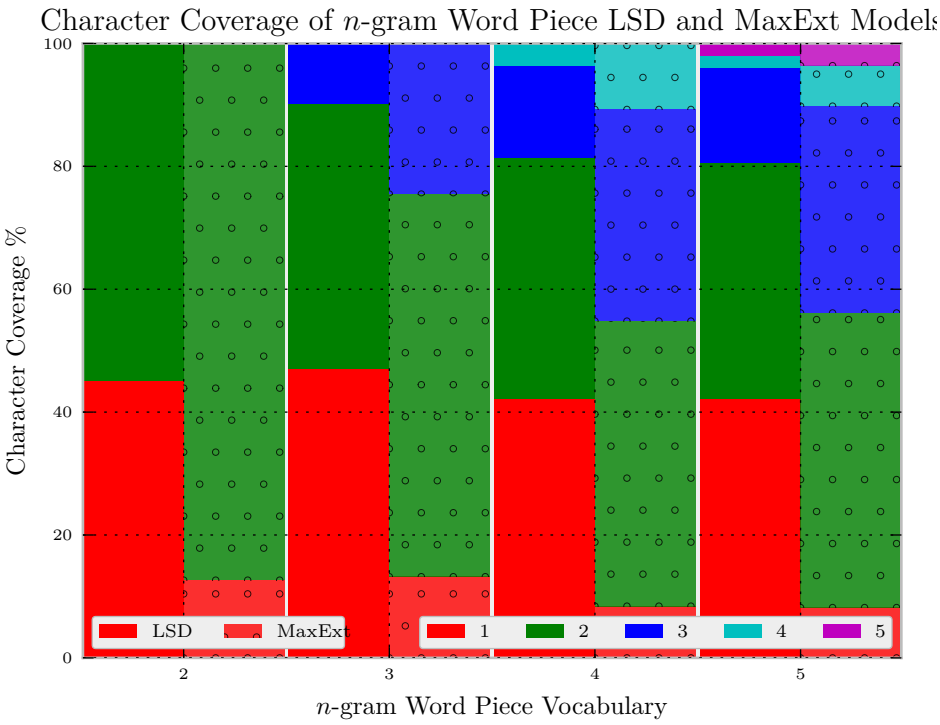

Figure 5.1: Character coverage of the word piece models. We train Latent Sequence Decompositions (LSD) and Maximum Extension (MaxExt) models with $n \in \{2, 3, 4, 5\}$ sized word piece vocabulary and measure the character coverage of the word pieces. Both the LSD and MaxExt models prefer to use $n \geq 2$ sized word pieces to cover the majority of the characters. The MaxExt models prefers longer word pieces to cover characters compared to the LSD models.

sible sequence, however it is a deterministic decomposition that can be easily generated in linear time on-the-fly. We decoded these models with simple n-best list beam search without any external dictionary or language model.

The baseline model is simply the unigram or character model and achieves 14.76% WER. We find the LSD $n = 4$ word piece vocabulary model to perform the best at 12.88% WER or yielding a 12.7% relative improvement over the baseline character model. None of our MaxExt models beat our character model baseline, suggesting the maximum extension decomposition to be a poor decomposition choice. However, all our LSD models perform better than the baseline suggesting the LSD framework is able to learn a decomposition better than the baseline character decomposition.

We also look at the word character coverage based on the word piece lengths during inference across different n sized word piece vocabulary used in training. We define word character coverage as the percentage of characters covered by the set of word pieces with the same length across the test set, and we exclude $\langle$space$\rangle$ in this statistic. Figure 5.1 plots the distribution of the $\{1, 2, 3, 4, 5\}$-ngram word pieces the model decides to use to decompose the sequences. When the model is trained to use the bigram word piece vocabulary, we found the model to prefer bigrams (55% of the characters emitted) over characters (45% of the characters emitted) in the LSD decomposition. This suggest that a character only vocabulary may not be the best vocabulary to learn from. Our best model, LSD with $n = 4$ word piece vocabulary, covered the word characters with 42.16%, 39.35%, 14.83% and 3.66% of the time using 1, 2, 3, 4 sized word pieces respectively. In the $n = 5$ word piece vocabulary model, the LSD model uses the $n = 5$ sized word pieces to cover approximately 2% of the characters. We suspect if we used a larger dataset, we could extend the vocabulary to cover even larger $n \geq 5$.

The MaxExt models were trained to greedily emit the longest possible word piece, consequently this prior meant the model will prefer to emit long word pieces over characters. While this decomposition results in the shorter $|\mathbf{z}|$ length, the WER is slightly worse than the character baseline. This suggests the much shorter decompositions generated by the MaxExt prior may not be best decomposition. This falls onto the principle that the best $\mathbf{z}^*$ decomposition is not only a function of $\mathbf{y}^*$ but as a function of $(\mathbf{x}, \mathbf{y}^*)$. In the case of ASR, the segmentation is a function of the acoustics as well as the text.

Table 5.2 compares our WSJ results with other published end-to-end models. The best CTC model achieved 27.3% WER with REINFORCE optimization on WER [11]. The previously best reported basic seq2seq model on WSJ WER achieved 18.0% WER [87] with Task Loss Estimation (TLE). Our baseline, also a seq2seq model, achieved 14.8% WER. Main differences between our models are that we did not use convolutional locational-based priors

Table 5.2: Wall Street Journal test eval92 Word Error Rate (WER) results across Connectionist Temporal Classification (CTC) and Sequence-to-sequence (seq2seq) models. The Latent Sequence Decomposition (LSD) models use a $n = 4$ word piece vocabulary. The Convolutional Neural Network (CNN) model is with deep residual connections, batch normalization and convolutions. The best end-to-end model seq2seq + LSD + CNN at 10.6% WER.

Model	**WER %**
Graves et al., 2014 [11]	
CTC	30.1
CTC + WER	27.3
Hannun et al., 2014 [86]	
CTC	35.8
Bahdanau et al., 2015a [64]	
seq2seq	18.6
Bahdanau et al., 2015b [87]	
seq2seq + TLE	18.0
Yu et al., 2016 [93]	
seq2seq + CNN	11.8
Our Work	
seq2seq	14.76
seq2seq + LSD	12.88
seq2seq + LSD + CNN	10.6

and we used weight noise during training. The deep CNN model with residual connections, batch normalization and convolutions achieved a WER of 11.8% [93].

Our LSD model using a $n = 4$ word piece vocabulary achieves a WER of 12.9% or 12.7% relatively better over the baseline seq2seq model. If we combine our LSD model with the CNN [93] model, we achieve a combined WER of 10.6% WER or 28.4% relatively better over the baseline seq2seq model. These numbers are all reported without the use of any language model.

Decompositions

Here in Table 5.3, we give the top 8 hypothesis generated by a baseline seq2seq character model, a Latent Sequence Decompositions (LSD) word piece model and a Maximum Extension (MaxExt) word piece model. We note that "shamrock's" is an out-of-vocabulary word while "shamrock" is in-vocabulary. The ground truth is "shamrock's pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said". Note how the LSD model generates multiple decompositions for the same word sequence, this does not happen with the MaxExt model.

5.6 Related Work

Recently, there has been work in MT to use a mixture of words and characters [57], or even a mixture of words and word pieces [101] to handle Out-of-Vocabulary (OOV) words and rare words. [95] showed in MT that using word pieces derived from a language model [94] can yield state-of-the-art results compared to character and word models. However, in all these work, the models assume a fixed deterministic decomposition of the word sequence – for example the decomposition are prescribed by a word piece language model [94]. In our work, we do not assume a fixed deterministic decomposition, but rather we marginalize over all possible decompositions. The decompositions we generate are also a function of both the input sequence and output sequence (as opposed to output sequence alone).

[102] used latent codes to generate text, and assume conditional independence and thus leverage on dynamic programming for exact inference. Such models can not learn the output language if the language distribution is complex and multimodal. Our model makes no such Markovian assumptions and can learn multimodal output distributions. [103] used seq2seq to outputs sets, the output sequence is unordered and used fixed length output units; in our decompositions we maintain ordering and use variable lengthed output units.

5.7 Summary

We presented the Latent Sequence Decompositions (LSD) framework. LSD allows us to learn decompositions of sequences. LSD learns a distribution of decompositions, the decompositions are a function of both the input and output sequence. We presented a training algorithm based on sampling valid extensions and an approximate decoding algorithm. On the Wall Street Journal speech recognition task, the sequence-to-sequence character model achieves 14.8% WER while the LSD model achieves 12.9%. If we combine our LSD model with a deep convolutional neural network on the encoder, we achieve 10.6% WER.

Table 5.3: Top hypothesis comparison between seq2seq character model, LSD word piece model and MaxExt word piece model.

n	**Hypothesis**	**LogProb**
Reference		
-	shamrock's pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said	-
Character seq2seq		
1	chamrock's pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said	-1.373868
2	chamrox pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said	-2.253581
3	chamrocks pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said	-3.482713
4	chamrock's pretax profit from the sale was one hundred twenty five million dollars of spokeswoman said	-3.493957
5	chamrod's pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said	-3.885185
6	chamrox pretax profit from the sale was one hundred twenty five million dollars of spokeswoman said	-4.373687
6	chamroc's pretax profit from the sale was one hundred twenty five million dollars a spokeswoman said	-5.148484
8	chamrocks pretax profit from the sale was one hundred twenty five million dollars of spokeswoman said	-5.602793
Word Piece Model Maximum Extension		
1	sh|am|ro|ck|'s| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-1.155203
2	sh|am|ro|x| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-3.031330
3	sh|ar|ro|x| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-3.074762
4	sh|e| |m| |ro|x| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-3.815662
5	sh|e| |mar|x| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-3.880760
6	sh|ar|ro|ck|s| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-4.083274
7	sh|e| |m| |ro|ck|ed| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-4.878025
8	sh|e| |m| |ro|ck|s| |pre|ta|x| |pro|fi|t| |from| |the| |sa|le| |was| |one| |hu|nd|red| |tw|ent|y| |five| |mil|lion| |doll|ars| |a| |sp|ok|es|wo|man| |said	-5.121490
Word Piece Model Latent Sequence Decompositions		
1	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|lio|n| |doll|a|r|s| |a| |sp|ok|e|s|wo|ma|n| |said	-28.111485
2	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|li|o|n| |doll|ar|s| |a| |sp|ok|e|s|wo|ma|n| |said	-28.172878
3	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|lio|n| |doll|a|r|s| |a| |sp|ok|e|s|w|om|a|n| |said	-28.453381
4	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|li|o|n| |doll|a|r|s| |a| |sp|ok|e|s|w|om|a|n| |said	-29.103184
5	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|lio|n| |doll|a|r|s| |a| |sp|ok|e|s|w|om|a|n| |sa|id	-29.159660
6	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|lio|n| |doll|a|r|s| |a| |sp|o|k|e|s|w|o|ma|n| |said	-29.164141
7	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|li|o|n| |doll|a|r|s| |a| |sp|ok|e|s|w|om|a|n| |sai|d	-29.169310
8	sh|a|m|ro|c|k|'s| |pre|ta|x| |pro|fi|t| |fro|m| |t|h|e| |sa|l|e| |was| |on|e| |hu|n|dr|e|d| |t|we|nt|y| |fiv|e| |mil|li|o|n| |doll|a|r|s| |a| |sp|ok|e|s|w|om|a|n| |sa|id	-29.809937

Chapter 6

Conclusion

In this thesis we have presented end-to-end speech recognition models. Our models jointly learn all the traditional components of a speech recognizer (acoustic, pronunciation, language, text normalization). We submit our model is not only an *end-to-end trained system*, but also an *end-to-end model*. We accomplish this by making no conditional independence or Markovian assumptions about the distribution of our output sequence. This distinguishes from CTC and HMM based models that can be trained end-to-end but is not a true end-to-end model as they must rely heavily on an external language model.

On the Google voice search task, we show our model can be competitive to state-of-the-art HMM based systems (Chapter 3). We show our model to be able to learn multimodal output distributions given the acoustics. For example, our model can generate "triple a" vs "aaa" in our top 2 hypotheses due to the conditional dependency of our model. This is likely not possible in a CTC or HMM based system without special handcrafted dictionaries or a very strong language model. We achieve 14.1% WER without the usage of any language model and 10.3% WER using only language model rescoring, which compares to a well tuned CLDNN-HMM system of 8.0% WER.

We also show our model can be made online, search-free and applied to Chinese Mandarin

(Chapter 4). We show how to jointly optimize the Mandarin Character output sequence with the Mandarin Pinyin sequence. We show how this can improve the final CER results. On the GALE Mandarin evaluation, we achieve 72.0% CER with an online end-to-end model without any searching, and with the joint Pinyin training we achieve 59.3% CER.

Finally, we show how to train our models to emit word pieces and learn multiple decompositions of the output sequence as a function of the input sequence and output sequence (Chapter 5). We show state-of-the-art results on the Wall Street Journal task when compared to other end-to-end models without using an explicit language model. We achieve 12.88% WER with our latent sequence decomposition model, which compares to a sequence-to-sequence baseline of 14.76% WER and a CTC models achieve 27.3% WER.

We believe the research in our thesis will spur further research into end-to-end speech recognition models. The engineering simplicity in training and inference is very beneficial for many tasks – especially in embedded platforms where we can not carry around n-gram language models. Ultimately, we believe end-to-end speech recognition models will outperform CTC and HMM based systems. We already show competitive results on the Google voice search task, and we believe one of the key missing ingredients is simply more data.

6.1 Future Work

Research is never ending process. In this final section of our thesis, we highlight a few promising areas to advance end-to-end speech recognition models:

1. **More Data.** In Chapter 3 we showed competitive results on the Google voice search task (with approximately 2000 hours of training data) compared to state-of-the-art HMM based models. However, in Chapter 5, despite a more powerful latent sequence decomposition model, the performance gap between our end-to-end model and HMM based systems grew on the Wall Street Journal task (with approximately 100 hours of training data). Our hypothesis is that our model is simply too powerful, and too

easy to overfit (i.e., unlike CTC and HMM based systems, we do not make conditional independence assumptions). We believe our model will simply scale better on larger datasets.

2. **Task Loss Optimization.** In our work, we mostly only optimized for log probability which is not the true metric we care about. The log probability objective assumes we condition on the ground truth prefix and does not optimize for the relative rankings of the hypothesis. While our error sampling method alleviates some of these assumptions, they only fix for substitution errors and not insertion or deletion errors. Additionally, our optimization loss does not account for the beam search that happens in inference. Future work should optimize directly for WER. Evidence has already been shown that optimizing directly for the task loss with reinforcement learning may yield better models [104, 87].

3. **Learning the Network Latent Structure.** In all our experiments, we used a fixed computational network graph. Our neural network is a fixed hyperparameter. The best neural network structure may not be static and may be a function of the input sequence, similar to our latent sequence decomposition model. Future work should make the network structure a function of the input sequence and the parameters of the model. Evidence has been shown for some language tasks that this may be beneficial to learning a structure [105].

4. **Learning from Acoustic Data.** We should leverage on the virtually unlimited amount of acoustic-only data we have. The encoder should be able to improve with acoustic-only data to learn a better encoder. Possible methods include pre-training with a generative model such as variational autoencoders [106], generative adversarial networks [107] or autoregressive networks [108]. Evidence has already been shown we can build a very good generative acoustic models [109].

5. **Learning from Text Data**. We should leverage on the virtually unlimited amount

of text-only data we have. The decoder should be improved with text-only data to learn a stronger language model which is available in HMM based systems. Evidence has already been shown in machine translation that this may be a promising avenue of research [110].

6. **Learning the Word Pieces**. In Chapter 5, our model emitted word pieces from a vocabulary created based off of counts. Future work should learn this vocabulary and be able to dynamically grow and shrink the word piece vocabulary as needed. With a better word piece vocabulary, the model may learn better decompositions. Reinforcement learning methods would probably be needed [96].

Proverbs 19:21: "Many are the plans in the mind of a man, but it is the purpose of the Lord that will stand."

References

[1] C. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[2] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The htk book", in *Cambridge University Engineering Department*, 2002.

[3] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit", in *Automatic Speech Recognition and Understanding Workshop*, 2011.

[4] L. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains", *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 6 1966.

[5] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden markov models", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1990.

[6] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups", *IEEE Signal Processing Magazine*, Nov. 2012.

[7] CMUdict, "Cmu pronouncing dictionary", in *Http://www.speech.cs.cmu.edu/cgi-bin/cmudict*, 2014.

[8] T. Mikolov, K. Martin, B. Luka, E. Jan, and K. Sanjeev, "Recurrent neural network based language model", in *INTERSPEECH*, 2010.

[9] D. Povey, "Discriminative training for large vocabulary speech recognition", in *PhD Dissertation, Cambridge University Engineering Department*, 2003.

[10] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks", in *INTERSPEECH*, 2013.

[11] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks", in *International Conference on Machine Learning*, 2014.

[12] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recogniser", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 35, 3 1987.

[13] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1995.

[14] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: a neural network for large vocabulary conversational speech recognition", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2016.

[15] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks", in *Neural Information Processing Systems*, 2014.

[16] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwen, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation", in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", in *International Conference on Learning Representations*, 2015.

[18] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, Aug. 1980.

[19] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech", *Journal of the Acoustical Society of America*, vol. 87, 4 1990.

[20] T. Sainath, R. Weiss, A. Senior, K. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns", in *INTERSPEECH*, 2015.

[21] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition", *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 30–42, 1 Jan. 2012.

[22] N. Jaitly, P. Nguyen, A. W. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition", in *INTERSPEECH*, 2012.

[23] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft", May 2013.

[24] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[25] W. Chan, N. R. Ke, and I. Lane, "Transferring knowledge from a rnn to a dnn", in *INTERSPEECH*, 2015.

[26] G.-L. Chao, W. Chan, and I. Lane, "Speaker-targeted audio-visual models for speech recognition in cocktail-party environments", in *INTERSPEECH*, 2016.

[27] O. Abdel-Hamid, A. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

[28] T. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[29] T. Sainath, B. Kingsbury, A. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for lvcsr", in *Automatic Speech Recognition and Understanding Workshop*, 2013.

[30] William Chan and Ian Lane, "Distributed asynchronous optimization of convolutional neural networks", in *INTERSPEECH*, 2014.

[31] W. Chan and I. Lane, "Deep convolutional neural networks for acoustic modeling in low resource languages", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2015.

[32] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual convolutional neural networks for lvcsr", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2016.

[33] T. Sercu and V. Goel, "Advances in very deep convolutional neural networks for lvcsr", in *INTERSPEECH*, 2016.

[34] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[35] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling", in *INTERSPEECH*, 2014.

[36] W. Chan and I. Lane, "Deep recurrent neural networks for acoustic modelling", in *ArXiv:1504.01482*, 2015.

[37] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[38] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[39] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, pp. 257–286, 2 Feb. 1989.

[40] A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "Gmm-free dnn training", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014.

[41] A. Graves, S. Fernandez, F. Gomez, and J. Schmiduber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks", in *International Conference on Machine Learning*, 2006.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[43] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng, "Deep speech: scaling up end-to-end speech recognition", in *ArXiv:1412.5567*, 2014.

[44] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: end-to-end speech recognition using deep rnn models and wfst-based decoding", in *Automatic Speech Recognition and Understanding Workshop*, 2015.

[45] J. Li, H. Zhang, X. Cai, and B. Xu, "Towards end-to-end speech recognition for chinese mandarin using long short-term memory recurrent neural networks", in *INTERSPEECH*, 2015.

[46] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, "Deep speech 2: end-to-end speech recognition in english and mandarin", in *International Conference on Machine Learning*, 2016.

[47] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system", in *ArXiv:1609.03193*, 2016.

[48] G. Zweig, C. Yu, J. Droppo, and A. Stolcke, "Advances in all-neural speech recognition", in *ArXiv:1609.05935*, 2016.

[49] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning", in *ArXiv:1609.06773*, 2016.

[50] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation", in *Association for Computational Linguistics*, 2015.

[51] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation", in *Association for Computational Linguistics*, 2015.

[52] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: a neural image caption generator", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[53] A. Graves, "Generating sequences with recurrent neural networks", in *ArXiv:1308.0850*, 2013.

[54] Alex Graves and Greg Wayne and Ivo Danihelka, "Neural turing machines", in *ArXiv:1410.5401*, 2014.

[55] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell", in *ArXiv:1508.01211*, 2015.

[56] W. Ling, I. Trancoso, C. Dyer, and A. Black, "Character-based neural machine translation", in *ArXiv:1511.04586*, 2015.

[57] M.-T. Luong and C. Manning, "Achieving open vocabulary neural machine translation with hybrid word-character models", in *Association for Computational Linguistics*, 2016.

[58] O. Vinyals and Q. V. Le, "A neural conversational model", in *International Conference on Machine Learning: Deep Learning Workshop*, 2015.

[59] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language", in *Neural Information Processing Systems*, 2015.

[60] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: neural image caption generation with visual attention", in *International Conference on Machine Learning*, 2015.

[61] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion", in *INTERSPEECH*, 2015.

[62] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: first results", in *Neural Information Processing Systems: Workshop Deep Learning and Representation Learning Workshop*, 2014.

[63] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition", in *Neural Information Processing Systems*, 2015.

[64] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2016.

[65] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with bidirectional lstm", in *Automatic Speech Recognition and Understanding Workshop*, 2013.

[66] S. Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies", in *Neural Information Processing Systems*, 1996.

[67] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn", in *International Conference on Machine Learning*, 2014.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 11 Nov. 1998.

[69] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks", in *Neural Information Processing Systems*, 2015.

[70] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks", in *Neural Information Processing Systems*, 2012.

[71] A. Maas, Z. Xie, D. Jurafsky, and A. Ng, "Lexicon-free conversational speech recognition with neural networks", in *North American Chapter of the Association for Computational Linguistics*, 2015.

[72] H. Hwang and W. Sung, "Character-level incremental speech recognition with recurrent neural networks", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016.

[73] L. Lu, X. Zhang, K. H. Cho, and S. Renals, "A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition", in *INTERSPEECH*, 2015.

[74] N. Jaitly, D. Sussillo, Q. Le, O. Vinyals, I. Sutskever, and S. Bengio, "A neural transducer", in *Neural Information Processing Systems*, 2016.

[75] W. Chan and I. Lane, "On online attention-based speech recognition and joint mandarin character-pinyin training", in *INTERSPEECH*, 2016.

[76] Y. Luo, C.-C. Chiu, N. Jaitly, and I. Sutskever, "Learning online alignments with continuous rewards policy gradient", in *BooktitlearXiv:1608.01281*, 2016.

[77] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016.

[78] L. Lu, L. Kong, C. Dyer, N. Smith, and S. Renals, "Segmental recurrent neural networks for end-to-end speech recognition", in *INTERSPEECH*, 2016.

[79] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", in *Neural Information Processing Systems: Workshop Deep Learning and Representation Learning Workshop*, 2014.

[80] Y. Miao, M. Gowayyed, X. Na, T. Ko, F. Metze, and A. Waibel, "An emprical exploration of ctc acoustic models", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016.

[81] *Cc-cedict*, cc-cedict.org, 2016.

[82] M. Zeiler, "Adadelta: an adaptive learning rate method", in *ArXiv:1212.5701*, 2012.

[83] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning", in *International Conference on Machine Learning*, 2013.

[84] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", in *ArXiv*, 2012.

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Neural Information Processing Systems*, 2012.

[86] A. Hannun, A. Maas, D. Jurafsky, and A. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns", in *ArXiv*, 2014.

[87] D. Bahdanau, D. Serdyuk, P. Brakel, N. R. Ke, J. Chorowski, A. Courville, and Y. Bengio, "Task loss estimation for sequence prediction", in *International Conference on Learning Representations Workshop*, 2016.

[88] W. Chan, Y. Zhang, Q. Le, and N. Jaitly, "Latent sequence decompositions", in *ArXiv*, 2016.

[89] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition", in *INTERSPEECH*, 2015.

[90] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition", in *INTERSPEECH*, 2015.

[91] V. Vanhoucke and A. Senior, "Improving the speed of neural networks on cpus", in *Neural Information Processing Systems: Deep Learning and Unsupervised Feature Learning Workshop*, 2011.

[92] Y. Wang, J. Li, and Y. Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-vq", in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[93] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition", in *ArXiv*, 2016.

[94] M. Schuster and K. Nakajima, "Japanese and korean voice search", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2012.

[95] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: bridging the gap between human and machine translation", in *ArXiv:1609.08144*, 2016.

[96] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[97] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks", *IEEE Transactions on Signal Processing*, vol. 45, no. 11, 1997.

[98] A. Graves, "Practical variational inference for neural networks", in *Neural Information Processing Systems*, 2011.

[99] D. Kingma and J. Ba, "Adam: a method for stochastic optimization", in *International Conference on Learning Representations*, 2015.

[100] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `http://tensorflow.org/`.

[101] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units", in *Association for Computational Linguistics*, 2016.

[102] W. Ling, E. Grefenstette, K. M. Hermann, T. Kocisky, A. Senior, F. Wang, and P. Blunsom, "Latent predictor networks for code generation", in *Association for Computational Linguistics*, 2016.

[103] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: sequence to sequence for sets", in *International Conference on Learning Representations*, 2016.

[104] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks", in *International Conference on Learning Representations*, 2016.

[105] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks", in *ArXiv:1609.01704*, 2016.

[106] D. Kingma and M. Welling, "Auto-encoding variational bayes", in *International Conference on Learning Representations*, 2014.

[107] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks", in *Neural Information Processing Systems*, 2014.

[108] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks", in *International Conference on Machine Learning*, 2014.

[109] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: a generative model for raw audio", in *BooktitlearXiv:1609.03499*, 2016.

[110] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation", in *BooktitlearXiv:1503.03535*, 2015.