High Performance Neural Networks for Online Speech Recognizer

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik

des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Thai-Son Nguyen

aus Zweibrücken

Tag der mündlichen Prüfung: 20.11.2020

Erster Gutachter: Prof. Dr. Alexander Waibel

Zweiter Gutachter: Prof. Dr. Shinji Watanabe

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe, sowie dass ich die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des KIT, ehem. Universität Karlsruhe (TH), zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 1. Oktober 2020

Thai-Son Nguyen

Abstract

Automatic speech recognition (ASR) refers to the ability of a machine to identify words and phrases in spoken languages and convert them to a human-readable format. Its application remains an essential ability for human digital life, such as allowing verbal dialog between humans and machines or enabling cross-lingual communication between people speaking different native languages. To fully afford this ability, ASR applications not only need to work with high accuracy but also have to respond quickly enough for their expected interactions with users. This mixture of both constraints opens up the research area of online speech recognition differing from conventional speech recognition, which addresses solely the accuracy problem.

The research on automatic speech recognition have been active for over half of a century. Several patterns and template matching approaches were proposed until the mid-1980 when Hidden Markov Model (HMM) became a breakthrough to solve the speech recognition task. The HMM approach allows a generic framework to statistically decouple and model both temporal and spectral variations in speech. At the latest fashion, an HMM-based recognizer is built up on top of a complex pipeline, composed of several statistical and non-statistical components such as pronunciation dictionaries, HMM topologies, phoneme cluster trees, an acoustic model and a language model. The recent advances of artificial neural networks (ANN) in both acoustic modeling and language modeling have made the hybrid HMM/ANN approach dominant in many types of ASR applications.

In recent years, the introduction of all-neural end-to-end speech recognition, which uses a neural network architecture to approximate the direct mapping from acoustic signals to the textual transcription, has been received significant interest. The advantage of the end-to-end approaches lies in their simplification of training an entire

speech recognition system, thereby hiding the awareness of complicated components as in the HMM-based pipeline. At the same time, the end-to-end ASRs typically require a more substantial amount of training data, and it is more challenging to adapt end-to-end models to perform well on a new task.

This thesis is devoted to the development of high-performance speech recognition systems for the online and streaming scenario. The author achieved this target by a two-stage approach. In the *first stage*, various techniques, applied in both hybrid HMM/ANN and end-to-end paradigms, were proposed to construct high-performance systems in batch mode, i.e., the complete audio data is available when starting processing. In the *second stage*, efficient adaptations were explored to enable the high-performance batch-mode systems to be capable of online and run-on inferences. At the same time, novel algorithms were developed to reduce user-perceived latency, which is the most critical issue of online speech recognizers.

First Stage. The proposed techniques aiming for high-performance achievement are categorized by which stage in the speech recognition pipeline they involved in, which are *feature extraction* and *data augmentation*.

Speech signals, known as the convolution of multiple frequency components in a wide dynamic range, before becoming a digital form, can be changed dramatically with natural factors, such as different speakers, environments, or recording tools. The large variability of speech signals typically causes the mismatch between training and testing, and then may largely degrade recognition performance. We address this mismatch problem by introducing two high-level network-based *feature extraction* approaches. In the first approach, a new feature space with less speaker variance is conducted via a hierarchical combination of bottleneck neural network and speaker adaptation techniques such as maximum likelihood linear regression (MLLR) transformation and speaker identity vector (i-vector) extraction. We showed that a deep neural network (DNN) acoustic model trained on these speaker-adapted features, gains up to 19% relative in word error rate (WER) over the conventional feature extraction. In the second approach, long short-term memory (LSTM) network trained with the connectionist temporal classification criterion (CTC) on phone labels is used as a high-level feature transformation. The combination of the CTC-network derived features and the bottleneck features resulted in an efficient feature space which made a DNN acoustic model outperform a strong CTC-based baseline with a large

margin. Besides, we revealed the use of the standard cepstral mean and variance normalization (CMVN) at low-level feature extraction causes a potential mismatch between offline training and online testing, and proposed a Linear Discriminant Analysis (LDA) based linear transformation for the replacement.

Data augmentation has been used in speech recognition for producing additional training data to increase the quality of the training data. This technique then improves the robustness of the models and avoids overfitting. We pointed out that overfitting is the most critical issue when training end-to-end sequence-to-sequence (S2S) models for the speech recognition task, and proposed two novel on-the-fly data augmentation methods as the solution. The first method, so-called *dynamic time stretching*, obtains the effect of speed perturbation by manipulating the time series of the frequency vectors directly with a real-time interpolation function. In the second method, we proposed an efficient strategy to sub-sample speech sentences on-the-fly, and then enlarge the training data with more variants of original samples. We showed that these methods are very efficient to avoid overfitting, and the combination of them with the *SpecAugument* method in the literature boosted up the performance of the proposed S2S model to be state-of-the-art on the telephone conversation benchmark.

Second Stage. We showed that the proposed high-performance batch-mode ASR systems of both hybrid HMM/ANN and end-to-end paradigms could meet the requirements of online real-world settings with the additional adaptation and inference techniques.

Neither the commonly used *real-time factor* nor *commitment latency* are sufficient to indicate the latency that users perceive. We proposed a novel and efficient method for measuring user-perceived latency in online and streaming setup. We further revealed that to better capture user experience, a run-on hybrid HMM/ANN recognizer needs to be optimized for the latency at either its peak or average. To improve these latency metrics, we introduced a mechanism so-called *hypothesis update*, which allows sending hypothetical transcripts early to the users, then later revising a part of it. Experiments on a real-world setup of the lecture presentation domain showed that this approach largely reduced the word-based latency of our recognizers, i.e., from 2.10 to 1.09 seconds.

Sequence-to-sequence (S2S) attention-based model has become increasingly popular for end-to-end speech recognition. Several advances have been proposed

to the architecture and the optimization of S2S models to achieve state-of-the-art performance on standard benchmarks. However, how to employ S2S models with their batch-mode capacity in online speech recognition still a research question. We approached this problem by analyzing the latency issues that occurred from the regular soft-attention function, bidirectional encoder, and beam-search inference. We addressed all the latency issues as a whole by proposing an additional training loss to control the uncertainty of the attention function on look-ahead frames and novel inference algorithms for providing partial hypotheses. Our experiments on the standard telephone conversation task show that with a delay of 1.5 seconds in all output elements, our streaming recognizer can fully achieve the performance of a batch-mode system of the same configuration. To the best of our knowledge for the first time, a S2S speech recognition model can be used in online conditions without scarifying accuracy.

Zusammenfassung

Automatische Spracherkennung (engl. *automatic speech recognition*, ASR) beschreibt die Fähigkeit einer Maschine, Wörter und Ausdrücke gesprochener Sprache zu identifizieren und diese in ein für Menschen lesbares Format zu konvertieren. Die Anwendungen sind ein maßgeblicher Teil des digitalen Lebens bspw. wird der Dialog zwischen Mensch und Maschine oder ein Dialog zwischen Menschen, die unterschiedliche Muttersprachen sprechen, ermöglicht. Um diese Fähigkeit in vollem Maße zu gewährleisten, müssen ASR-Anwendungen nicht nur mit hoher Genauigkeit, sondern, für eine Interaktion mit einem Benutzer, auch schnell genug, antworten. Dieses Wechselspiel beider Bedingungen eröffnet das Forschungsgebiet der *Online Speech Recognition*, welche sich von der konventionellen Spracherkennung, die sich ausschließlich mit dem Problem der Genauigkeit befasst, unterscheidet.

Schon über ein halbes Jahrhundert wird aktiv in der automatischen Spracherkennung geforscht. Verschiedene Muster- und Template-Matching-Methoden wurden bis Mitte 1980 erforscht, als das Hidden Markov Model (HMM) einen Durchbruch zur Lösung der Spracherkennungsaufgabe ermöglichte. Der HMM-Ansatz schafft ein allgemeines Framework, welches Schwankungen in der Zeit sowie Spektrums-Domäne der Sprache statistisch entkoppelt und modelliert. Ein HMM-basierter Erkenner wird auf eine komplexe Pipeline aufgesetzt, welche aus etlichen statistischen und nicht-statistischen Komponenten, wie bspw. einem Aussprachewörterbuch, HMM-Topologien, Phonem-Cluster-Bäumen, einem akustischen Modell und einem Sprachmodell, besteht. Durch aktuelle Fortschritte bei künstlichen neuronalen Netzen (KNN) für die akustische sowie sprachliche Modellierung dominiert der hybride HMM/KNN-Ansatz in unterschiedlichen ASR-Anwendungen.

In den letzten Jahren hat die Einführung komplett neuronaler Ende-zu-Ende Spracherkennungssystems, welche eine neuronale Netzwerkarchitektur verwenden, um die direkt Abbildung eines akustischen Signals zu einer textuellen Transkription zu approximieren, großes Interesse auf sich gezogen. Die Vorteile des Ende-zu-Ende-Ansatzes liegen in der Einfachheit des Trainings eines kompletten Spracherkennungssystems, wobei die komplexe Struktur einer HMM-basierten Pipeline entfällt. Gleichzeitig benötigt die Ende-zu-Ende ASR oft eine wesentlich größere Trainingsdatenmenge und es ist eine größere Herausforderung ein Ende-zu-Ende Modell so anzupassen, dass es auf einer neuen Aufgabe gut abschneidet.

Diese Dissertation befasst sich mit der Entwicklung eines hoch-performanten Spracherkennungssystems für ein Online- und Streaming-Szenario. Der Autor erreichte dies durch ein Vorgehen in zwei Schritten. Im *ersten Schritt* wurden vielfältige Techniken im HMM-KNN- und Ende-zu-Ende-Paradigma angewandt, um ein hoch-performantes System im Batch-Mode zu bauen. Batch-Mode bedeutet, dass die vollständigen Audiodaten beim Start der Verarbeitung zur Verfügung stehen. Im *zweiten Schritt* wurden effiziente Anpassungen untersucht, die einem hoch-performanten Batch-Mode-System ermöglichen Inferenzen online bzw. fortlaufend durchzuführen. Gleichzeitig wurden neuartige Algorithmen zu Reduktion der wahrgenommenen Latenz, welche das kritischste Problem von online Spracherkennern ist, entwickelt.

Erster Schritt. Die vorgestellte Techniken, die auf hochperformante Ergebnisse abzielen, können anhand deren Position in der Spracherkennungs-Pipeline, wie Merkmalsextraktion und Daten-Augmentierung, kategorisiert werden.

Bevor Sprachsignale eine digitale Form annehmen, sind sie als Ergebnis der Faltung mehrere Frequenzkomponenten in einem großen Dynamikumfang bekannt. Diese Merkmale können drastisch durch natürliche Faktoren, wie bspw. unterschiedliche Sprecher, Umgebungen order Aufnahmegeräte, beeinflusst werden. Die große Varianz der Sprachsignale verursacht typischerweise die Diskrepanz zwischen Training und Test und kann die Erkennungsleistung drastisch verschlechtern. Diese Diskrepanz gehen wir durch zwei high-level Ansätze, welche auf Neuronalen Netzen basieren, in der *Merkmalsextraktion* an. Wir zeigten, dass auf tiefe neuronale Netze (DNN) basierte akustische Modelle, die mittels dieser Sprecher-angepasster Merkmale

trainiert wurden, in Bezug auf die Wortfehlerrate (WER) relativ, bis zu 19% besser abschneiden, als herkömmliche Merkmalsextraktionen. Im zweiten Ansatz wird ein Long short-term memory (LSTM) Netzwerk, das mittels Connectionist Temporal Classification (CTC) Kriterium auf Phon-Labeln trainiert wurde, als High-Level Merkmals-Transformation verwendet. Die Kombination der aus dem CTC-Netzwerk extrahierten Merkmale und der Bottleneck-Merkmale ergab einen effizienten Merkmalsraum, der ein DNN-basiertes akustisches Modell ein starkes CTC-basierendes Baseline Modell mit deutlichem Vorsprung übertreffen ließ. Darüber hinaus zeigten wir, dass die Verwendung einer Standard Cepstral Mean und Varianz Normalisierung (CMVN) als low-level Merkmalsextraktion in einer potenziellen Diskrepanz von Offline Training und Online Test resultiert und schlugen eine Lineare Diskriminaz Analyse (LDA), die auf linearer Transformation basiert, als Ersatz vor.

Daten-Augmentierung wurde in der Spracherkennung verwendet, um zusätzliche Trainingsdaten zu generieren und so die Qualität der Trainingsdaten zu erhöhen. Diese Technik verbessert die Robustheit des Modells und verhindert Overfitting. Wir zeigten, dass Overfitting das kritischste Problem beim Training eines Ende-zu-Ende Sequence-to-sequence (S2S) Modells für die Spracherkennungsaufgabe ist und stellten zwei neuartige on-the-fly Daten-Augmentierungsmethoden als Lösung vor. Die erste Methode (*dynamic time stretching*) simuliert den Effekt von Geschwindigkeitsänderungen durch eine direkte Manipulation der zeitlichen Folge an Frequenzvektoren durch eine Echtzeit-Interpolationsfunktion. In der zweiten Methode zeigten wir eine effiziente Strategie, um gesprochene Sätze on-the-fly zu sub-samplen und so die Trainingsdatenmenge mit mehrere Varianten eines einzelnen Samples zu vergrößern. Wir zeigten, dass diese Methoden sehr effizient sind, um Overfitting zu vermeiden und die Kombination mit der *SpecAugment*-Methode aus der Literatur verbesserte die Leistung des vorgestellten S2S-Modells zu einem State-of-the-Art auf dem Benchmark für Telefongespräche.

Zweiter Schritt. Wir zeigten, dass die vorgestellten Hochleistungs-Batch-Mode ASR Systeme des hybriden HMM/KNN und Ende-zu-Ende Paradigmas die Anforderungen in einer online bzw. realen Situation, durch zusätzliche Anpassungen und Inferenz-Techniken, erfüllen.

Weder der üblicherweise verwendete Echtzeitfaktor, noch die Commitment-Latenz sind ausreichend, um die vom Benutzer wahrgenommene Latenz aufzuzeigen. Wir

stellten eine neuartige und effiziente Methode zur Messung der vom Benutzer wahrgenommenen Latenz in einer Online- und Streaming-Situation vor. Wir zeigten weiter auf, dass ein fortlaufender HMM/KNN Erkenner entweder für den Latenzhöchstwert oder die mittlere Latenz optimiert werden sollte, um das Nutzererlebnis zu verbessern. Um die Latenzmetrik zu optimieren, führten wir einen Mechanismus ein (*Hypothese Update*), welcher erlaubt hypothetische Transkripte früh zum Benutzer zu schicken und diese später teilweise zu korrigieren. In Experimenten in einer realen Situation in der Vorlesungspräsentations-Domäne konnte gezeigt werden, dass dieses Vorgehen die Wort-basierte Latenz unseres Erkenners stark reduziert, d.h. von 2,10 auf 1,09 Sekunden.

Das Sequence-to-sequence (S2S) Attention-basiertes Modell ist für Ende-zu-Ende Spracherkennung zunehmend beliebt geworden. Etliche Vorteile der Architektur und der Optimierung eines S2S-Modells wurde vorgestellt, um State-of-the-Art Ergebnisse auf Standard-Benchmarks zu erreichen. Wie S2S-Modelle mit ihrem Batch-Mode Kapazität aber für eine online Spracherkennung gebraucht werden können, ist dennoch eine offene Forschungsfrage. Wir näherten uns diesem Problem, indem wir die Latenzprobleme, die durch die normale Softmax-Attention Funktion, bidirektionale Encoder und die Inferenz mit Strahlensuche verursacht wurden, analysierten. Wir nahmen uns all dieser Latenzprobleme in einem an, in dem wir einen zusätzlichen Trainings-Loss, um die Unsicherheit der Attention-Funktion auf Frames auf die vorausgeblickt wird, und einen neuartigen Inferenz-Algorithmus, der partielle Hypothesen bestimmt, vorstellen. Unsere Experimente auf dem Datensatz mit Telefongesprächen zeigten, dass unser Stream-Erkenner, mit einer Verzögerung von 1,5 Sekunden für alle Ausgabeelemente, in vollem Umfang die Performanz eines Batch-Mode-Systems derselben Konfiguration erreicht. Nach bestem Wissen ist dies das erste Mal, dass ein S2S-Spracherkennungsmodell in einer online Situation ohne Einbußen in der Genauigkeit genutzt werden kann.

To my family.

In gratitude.

Acknowledgements

I would like to thank every one who has supported and helped me during the work on my thesis. First and foremost, I want to thank Prof. Alex Waibel for making me part of his team and letting me perform the research leading to my thesis, for his advice and support, for our discussions, and for being a rich source of new ideas. Working at the Interactive Systems Laboratories has been a real joy. I would also like to thank Prof. Shinji Watanabe for co-advising this thesis.

A special thank goes to Maximilian Awiszus and Jan Niehues for proof-reading parts of my thesis. My thanks also extend to all the past and present members of the ASR team with whom I have worked over the years: Michael Heck, Matthias Sperber, and Thomas Zenkel, Kevin Kilgour, Markus Müller, Juan Hussain, Tuan Nam Nguyen and Sebastian Stüker. Additionally to the ASR team, I would like to thank the members of the MT and Dialog teams with whom I had many fruitful discussions during my time at the lab: Jan Niehues, Eunah Cho, Thanh-Le Ha, Ngoc Quan Pham, Teresa Herrmann, Mohammed Mediani, Maria Schmidt, Stefan Constantin and Felix Schneider.

My thanks also go out to all secretarial, technical and administrative staff. First and foremost Silke Dannenmaier and Margit Rödder, Bastian Krüger, Franziska Vogel, Virginia Roth and Mirjam Simantzik.

Last but not the least, I would like to give my thanks to my beloved: my wife, Phuong-Nhung, my daughter, Nhu-Van and my son, Nam-Phong for sharing every moments in my life during these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic speech recognition is one of the most challenging AI problems. The task of building an online speech recognition system is even more challenging due to the addition of the low-latency constraint. In this thesis, we present a two-stage approach for developing online streaming speech recognizers. In the first stage, we propose and develop various techniques for the construction of high-performance speech recognition systems in offline condition. In the second stage, we explored efficient adaptations that enable the high-performance batch-mode systems to be capable of streaming processing. We further proposed novel algorithms to reduce the latency of online speech recognizers.

1.1 Contribution

The main contribution of this thesis is a systematic approach and needed techniques to construct and evaluate high-performance neural network for online speech recognizer. While showing the commonly used real-time factor is not a suitable for latency measure, we proposed a novel method to capture better the latency that users perceive in an online speech recognizer. We proposed several techniques to deal with the latency issues occurred with the current approaches to build HMM/ANN and end-to-end ASR systems. We conducted several comprehensive experiments to verify if the proposed online ASR system to match the practical constraints in both latency and accuracy.

1.2 Overview

This section gives an overview of the contents of the individual chapters of this thesis.

Chapter 1 gives an introduction to the topic of this thesis.

Chapter 2 describes the fundamental theory applied in this work. Automatic speech recognition is introduced and fundamentals of ASR systems are given. Machine learning algorithms used throughout this work, such as hidden markov model and neural networks are described. Also, the evaluation metrics we used to measure the performance of the models built in this work are introduced.

Chapter 3 shows the limits of available techniques and the challenges of developing an online speech recognition system.

Chapter 4 provides the descriptions of the experimental datasets as well as toolkit used in the thesis.

Chapter 5 presents the novel techniques for extracting acoustic feature in online condition.

Chapter 6 presents the data augmentation methods which improve the performance as well as increase the robustness of the models on different recording conditions and speaking styles.

Chapter 7 reviews the common methods for measuring latency and describes a novel approach for more suitable user-perceived latency.

Chapter 8 presents the techniques which enable the batch-mode models proposed for both hybrid HMM/ANN and end-to-end paradigms to be used for online recognizers while retaining their high performance.

Chapter 2

Theory and Background

This chapter establishes the background of the thesis. We start by introducing the problem of automatic speech recognition (ASR) and the fundamentals for decomposing and modeling the problem. This is followed the description of the preprocessing and evaluation techniques which are commonly used for the construction of an ASR system. The following chapters concentrate on describing Artificial Neural Networks (ANNs) and the associated techniques that provides methods and tools used for solving the ASR problem in this thesis.

2.1 Automatic Speech Recognition

Automatic speech recognition (ASR) refers to the ability of a machine to identify words and phrases in spoken languages and convert them to a human-readable format. The research on automatic speech recognition have been active for over half of a century. Several patterns and template matching approaches were proposed until the mid-1980 when Hidden Markov Model (HMM) became a breakthrough to solve the speech recognition task. The HMM framework has been dominant for ASR development for several decades. In this section, we first describe the HMM-based ASR approach. We then provide theoretical background for other important parts for the construction of ASR systems including feature extraction, inference and evaluation methods.

2.1.1 Hidden Markov Model

Hidden Markov model (HMM) have been explored in the 1970s for the recognition of continuous speech at IBM [JBM75] and Carnegie Mellon University [Bak75]. The HMM-based ASR approach decomposes the speech recognition task with the inspiration from the noisy channel model [Sha01]:

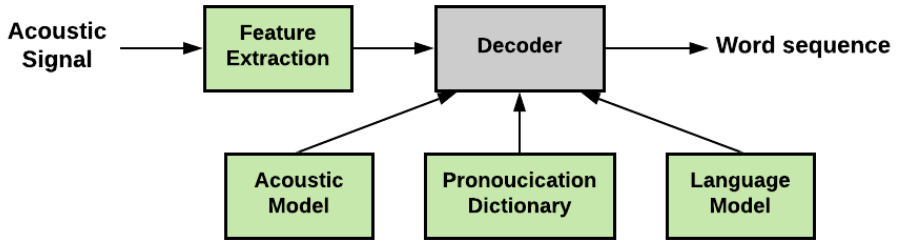
$$\begin{aligned}\hat{W} = argmax_W P(W|X) &= argmax_W \frac{P(X|W).P(W)}{P(X)} \\ &= argmax_W P(X|W).P(W),\end{aligned}$$

where W is a word sequence, X denotes an acoustic observation or speech utterance, and $\hat{W}$ is the best guess for the transcription of the utterance. The noisy channel considers the given utterance to be scrambled by a noisy channel as only a noisy signal is observed. Then, the speech recognition task is to decode the original signal based on the noisy signal. In the derived formula, the term P(W) is a prior which can linguistically be estimated, and P(X|W) is the likelihood of observing a sequence of acoustic vectors given a particular word sequence. The noisy channel offers a convenient approach to separate and combine the prior and the conditional model.
Figure 2.1 illustrates how this approach work in practice. The speech signal is processed to obtain a sequence of feature vectors X and an n-gram language model [Sha01, CG99] is usually applied to P(W). The probability P(X|W) is statistically learned with a Gaussian Mixture Model (GMM) or deep neural network (DNN) acoustic model. Importantly, the language model, acoustic model, and dictionary are estimated independently of each other. Furthermore, because the HMM-based acoustic models work best over phonemes while language models over words, a pronunciation dictionary is necessary to establish a proper mapping. Decoding is performed by employing a beam search over the search space constructed from the HMM states.

2.1.2 Feature Extraction

Acoustic signals are recorded as discrete sequences of samples in a microphone. These signals are digitalized by an A/D converter at intervals of a certain frequency (e.g., 16kHz, 48kHz) and then quantized to have an array of 16-bit integer samples. For a

Figure 2.1: Block diagram of HMM-based speech recognition.



speech utterance of 5 seconds, the sample array can be 90,000 length. This fact results in a very challenging problem for computation and modeling. The raw signal therefore must be transformed into a space of much lower dimension without losing significant information. Based on the observation that humans can acquire the skill of interpreting speech signals in the frequency domain but are very poor at doing the same in the time domain, many prepossessing methods proposed to discrete Fourier transform to convert raw signal into the frequency domain. Further, the Mel scale and a filter-bank can be applied to group the frequencies of fixed ranges into bins as the inspiration from human auditory system. The result is a lower dimension (20-40) feature representation called Mel filterbank (FBANK) features that are by many techniques proposed in this thesis. Other common technique is cepstral coefficients (MFCC) which can be obtained by further applying cosine transformation.

2.1.3 Evaluation Metric

Word-error-rate (WER) is the most common metric for evaluating ASR systems. The WER is based on the edit distance, also referred to as Levenshtein distance. The main idea is to count the minimum number of word-level edits necessary to transform the incorrect output string into the correct reference string. Edits can be substitutions, insertions, and deletions. The WER is then defined as

$$WER = \frac{substitutions + insertions + deletions}{reference\ length} \times 100\%.$$

The minimum number of edits can be computed efficiently through a dynamic

programming algorithm. In languages where no clear word boundaries exist, such as Chinese and Japanese, the character edit rate can be used instead which works exactly the same but operates on character-level instead of word-level.

While WER is meaningful even when computed for individual test sentences, it is usually computed at the corpus level, so that longer sentences in the corpus are given proportionally more weight in the final score than shorter sentences.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) inspired by human brain have been around for over half of a century. In 1957, Rosenblatt formulated the perceptron algorithm [Ros57] in which a perceptron features one or multiple inputs and activates with an activation and bias. As the main criticism of Minsky [MS69], a single perceptron can only be applied to linear separable problems. More complex problems can also be solved, but do require a multilayer perceptron (MLP) and non-linear activation functions. As the increase of number of neurons and layers, estimating for suitable values of the neurons in a neural network is difficult. Error backpropagation [RHW86] had been introduced to determine how the weights should be updated during the learning process and become a very efficient optimization method until now a day.

In recent years, due to the emergence of computing capabilities, several network architectures and optimization methods have been found and made ANNs become a very powerful machine learning method for artificial intelligence. A new trend so-called "Deep Learning" leverages the evolution of ANNs with deeper architectures and a large number of neurons, and efficiently applied it to several problems which were not possible to solve before such as computer vision, nature language processing, machine learning and speech recognition.

2.2.1 Feed-forward Neural Network

MLPs are also called feed forward neural networks (FFNNs). The neurons in such a network are organized in layers. While the neurons are connected between two consecutive layers, there are no connections of neurons within the same layer. The use of FFNN always comes with non-linear activation functions since the layers could

otherwise be can collapsed into one layer big network and the advantage of using multiple layers would vanish.

The connections between neurons in FFNNs can be further modified for better modeling particular inputs. *Time-Delay Neural Networks* (TDNNs), introduced by [WHH+87], and its successor *Convolutional Neural Networks* [LB+95] are a special kind of feedforward neural networks which can effectively deal with variable-length (time-shifted) sequences and able to model long distance dependencies within them. TDNNs therefore have been applied in many machine learning problems, mostly in speech recognition but also widely popular in other areas such as computer vision, robotics, time series prediction and natural language processing.

Instead of using a fully-connected layer, TDNNs run a window from the delayed input D_i to D_{i+N} with its parameters W and computes the weighted sum of its inputs and feed it through a nonlinear function f:

$$h_i = f(W^T \cdot S_{i:i+N})$$

This non-linear transformation in the TDNN architecture are tied across the delays (time steps), thus TDNN layers are forced to learn features shifted within the patterns. Those features are called translation-invariant features, where they do not change over time. Applied in speech recognition, TDNNs are able to detect of time-independent sub-patterns, therefore, one does not need to perform an additional step for time alignment. After that, some sampling operation would be conducted. In TDNN architecture, max pooling is often used. By this way, the most active features can be selected as the inputs for the next TDNN layer. In CNN architecture, the sliding window is called kernels, and in principle, CNNs are TDNNs but they have been widely applied in computer vision and natural language processing.

2.2.2 Recurrent Neural Network

Recurrent neural networks (RNNs) [Elm90] are a variety of neural networks that was designed for modeling temporal sequences. RNNs tries to learn representation of time into internal states of the network. This is done by introducing context units as the way of keeping information at each time step and hidden units as the result of combining current input with the context units. Recurrent connection [Jor97] is used for that

combination. The internal representation is learned via the update on the hidden units which then provides the network with "*memory*". Formally, at time step t, given input x_t and hidden unit h_{t-1} from the previous step, the current hidden state is computed with an the RNN function as:

$$h_t = RNN(x_t, h_{t-1}) = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

The RNNs can be viewed as a feedforward neural networks if we unroll the hidden state over time. By unrolling we simply mean that we write out the network for the complete sequence as shown in Figure 2.2. Note that the learnable parameters are the same in all time step and the hidden units are transfered to the next step, which is the difference with feedforward neural networks. To learn RNNs, the back propagation can be used for updating the parameters as over the unrolled network. That version of backpropagation is called backpropagation through time (BPTT) [RM85, RF87, Wer88].

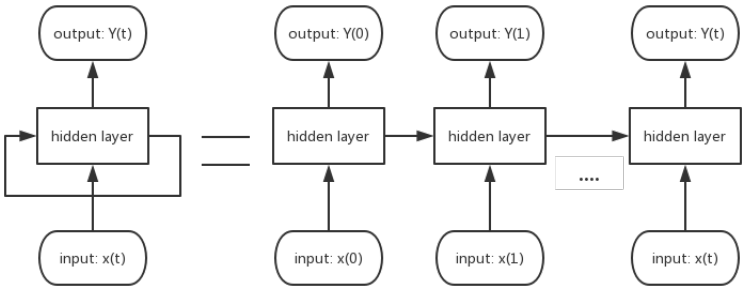


Figure 2.2: An unrolled RNN.

There is a serious problem with the RNNs when training them on a long sequence. On the backward pass, the error derivative w.r.t. to the inputs of simple recurrent hidden units can be extraordinarily large or very small close to zero over the time steps. More specifically, the gradient signal will always be multiplied by the same matrix W_{hh} when it is going back through one time step. So it is basically proportional to W_{hh}^N with N is the number of time steps. If all elements in W_{hh} larger than 1, the gradient becomes very large then the *exploding gradient* occurs. On the other hand if

all elements in W_{hh} smaller than 1, the gradient becomes smaller over time then the *vanishing gradient* occurs.

While RNNs are able to model context, they are limited in capturing long term dependencies due to the vanishing gradient problem [BSF94, HBF+01]. Long short-term memory (LSTM) networks were proposed to mitigate this problem [HS97, Hoc98]. By using internal memory cells, these networks are able to preserve information over longer distances. Each LSTM cell has 3 gates for controlling the hidden state: 1) a forget gate which determines how much information is preserved, 2) an input gate to decide how much new information should be stored and 3) an output gate to control how much of the internal state is output. Figure 2.2 (from [GSS02]) shows the information flow within an LSTM cell, including peephole connections [GSS02]. LSTM networks exist also in a bi-directional variant (BiLSTM).

2.2.3 Attentional Encoder-Decoder Neural Network

Many sequential learning problems can be considered as the task of converting from a source sequence to a target sequence. Such as in machine translation, we have a sentence in English as the source and other sentence in French as the target for translation. Or in speech recognition, we need to convert from a sequence of acoustic vectors to a sequence of words. The attentional encoder-decoder (Attn-EncDec) neural network has invented to deal with these sequence-to-sequence problems. As introduced specifically for machine translation, the attentional encoder-decoder model quickly becomes the dominant approach for the domain [KB13, SVL14, BCB14] and later on was successfully applied in automatic speech recognition [CBS+15, CJLV, PCZ+, NSNW19].

A typical attentional encoder-decoder model includes an encoder network and a decoder network which employs attention mechanism. Given x as the source sequence and y as the target sequence, the Attn-EncDec aims to model the probability $P(y|x)$. At first, the model transforms the source sequence x into a sequence of representation vectors $h = \{h_1, ..., h_T\}$ using the encoder network. Then, the decoder network uses an attention function to produce a probability distribution y_i over the next token given a previous token sequence in auto-regressive manner. A common form of these

functions are:

$$\begin{aligned} h &= EncoderRNN(x) \\ s_i &= DecoderRNN(s_{i-1}, [y_{i-1}; c_{i-1}]) \\ c_i &= Attention(h, s_i) \\ y_i &= Distribution(s_i, c_i) \end{aligned}$$

where s_i is the decoder state at output step i and $Distribution$ is an MLP with softmax output. The $Attention$ function generates a context vector c_i which is a weighted average of all the vectors in h. So far, the most commonly used attention mechanism is originally proposed in [BCB14]. This approach uses trainable matrix W_h, W_s and vectors v, b to compute c_i as follow:

$$\begin{aligned} e_{i,j} &= v^\top tanh(W_h h_j + W_s s_i + b) \\ a_{i,j} &= softmax(e_{i,:})_j \\ c_i &= \sum_{j=1}^{T} a_{i,j} h_j \end{aligned}$$

2.3 Neural Networks Optimizations

How to optimize a neural networks to fit the training task is also the very important aspect beside the designing neural network architectures. In this section, we have reviewed the techniques for improving neural network optimizations. This includes different types of activation and loss functions, parameter initialization approaches as well as regularization techniques.

2.3.1 Activation Functions

Activation functions are a crucial component of neural network. Activation functions introduces nonlinearity, determine the output of a network model and its accuracy. They also have a major effect on the model's ability in converge and convergence speed, or in some cases, good activation functions can prevent the model from getting stack in

local minimums. They also affect to the computational efficiency in training for scaling larger neural networks.

2.3.2 Loss Functions

Loss function is one of the most important part in training neural networks. In one hand, it directly reflects the task that a network model has to learn. In other hand, loss functions hugely influence to the optimization of neural networks.

Cross Entropy

The cross-entropy loss [B+95] is the most common loss function used in training neural network as it is known as the best fit for the classification tasks.

Mean Squared Error

This function produces the mean squared error as the loss between the output and the ground truth label. To satisfy this loss function, the output of the networks and the ground truth need to be real value. This function also works for classification problems with hard targets but is knowns to be not as suitable as cross-entropy.

2.3.3 Parameter Initialization

Using a good set of initial weights of a neural network does not only improve the convergence, but sometime also helps the training to bypass local minimums and then results in better performance. Different network components may need different approaches for good parameters initialization. One approach is to select the parameters randomly, but condition the values based on certain criteria, e.g., based on the used activation function.

2.3.4 Regularizations

Due to the use of much larger number of parameters than other machine learning models, neural networks are typically prone to overfitting. This situation has both advantage and disadvantage. The advantage is to shows that the network is capable enough for the problem, however final result on test set can be poor as the

disadvantage. Regularization techniques are developed to improve this overfitting situation in neural networks. Several methods have been proposed:

L2 Regularization
L2 regularization adds "squared magnitude" of coefficient as penalty term to the loss function. The effect of this regularization is to let the model prefer weights with small values (closer to zeros is the better).

Dropout Training
Dropout [HSK+12] is a method to approximate training a large number of neural network models with different architectures in parallel. During training, some layer outputs node are randomly ignored (or "dropped out"). This results in an effect of making the layer look-like a new layer with a different set of nodes and connectivity to the prior layer. Dropout also has the effect of making the training process noisy, thus prevent network nodes from memorizing and co-adapting the inputs or prior layers.

Gradient Clipping
Another technique is gradient clipping, in which the value of back-propagated gradients are limited to a certain threshold [GBC16]. This reduces the impact of exploding gradients occurred during training.

2.3.5 Learning Schedule

As observed in many practices, the update of network network's parameters is sensitive to good convergence state (e.g. a large update may make the network jump out of the good local minimum), learning schedule needs to be defined carefully to obtain the best possible performance. Multiple learning schedule strategies have been proposed.

Fixed Scheduling
In this basic method, learning rates are predefined and fixed for a number of epochs. The fixed schedule is usually based on expert knowledge and prior experiments. Since

being static, this method does not account for the network's progress.

Exponential Decay

Instead of using fixed intervals, the exponential decay method decreases the learning rate exponentially after each epoch with a factor. This strategy is inspired from two observations. First, the big updates can make the network bypass bad local minimums while the small updates later keep the network go further to the good local minimum. This method can be used together with early stopping in which the training stops after the error on the validation set starts to increase.

Newbob

There exists yet another method, combining static learning rate scheduling with exponential decay, called "newbob" [MB90]. Based on the progress the network shows w.r.t. the error rate on the development set after each epoch, different stages are being selected. The training starts using a fixed learning rate. Once the decrease of the error rate falls below a certain threshold, newbob switches to exponential decay. The training continues in this mode until the observed delta of the error rate drops below the second threshold. After this threshold is met, the training stops. In total, this method introduces two additional parameters: The first threshold to determine the switch from a static learning rate to the exponential decay and the second threshold to stop the training.

Chapter 3

Modern Approaches in ASR

The research on automatic speech recognition has been active for over half of a century. Several patterns and template matching approaches were proposed until the mid-1980 when Hidden Markov Model (HMM) became a breakthrough to solve the speech recognition task. The success of applying HMM framework to ASR lies on the ability to decompose the modeling of speech sentences into two separate components: acoustic model (AM) and language model (LM). With the use of n-gram for language modeling and Gaussian Mixture Model (GMM) for acoustic modeling, HMM-based ASR has been the mainstream for a long time. Until the year 2010s, artificial neural networks (ANN) were successfully applied to solve the ASR task and made several changes to this research field.

This chapter reviews the modern approaches that employ neural networks to solve the speech recognition problem. We first introduce different neural network architectures proposed for modeling ASR and improving feature extraction. To give a systematic view on how neural networks have changed the ASR research, we provide an analysis on the progress of the ASR systems that have been made for the telephone conversation speech task over the decade.

3.1 ASR Modeling

ANNs have played an important role in the construction of high-performance speech recognition systems. At the early stage, ANNs are utilized as a technique for

acoustic modeling, which replace GMMs in HMM-based systems. Later on, ANNs have become a whole system for solving the speech recognition task. This section highlights three different approaches of neural network proposed for the speech recognition. Figure 3.1 gives an overview of these approaches. The left part presents HMM/ANN acoustic modeling in which deep neural network (DNN) is used to classify HMM transition states. The center part presents the neural network model using connectionist temporal classification (CTC) criterion, while the left part shows the overall architecture of a sequence-to-sequence encoder-decoder model.

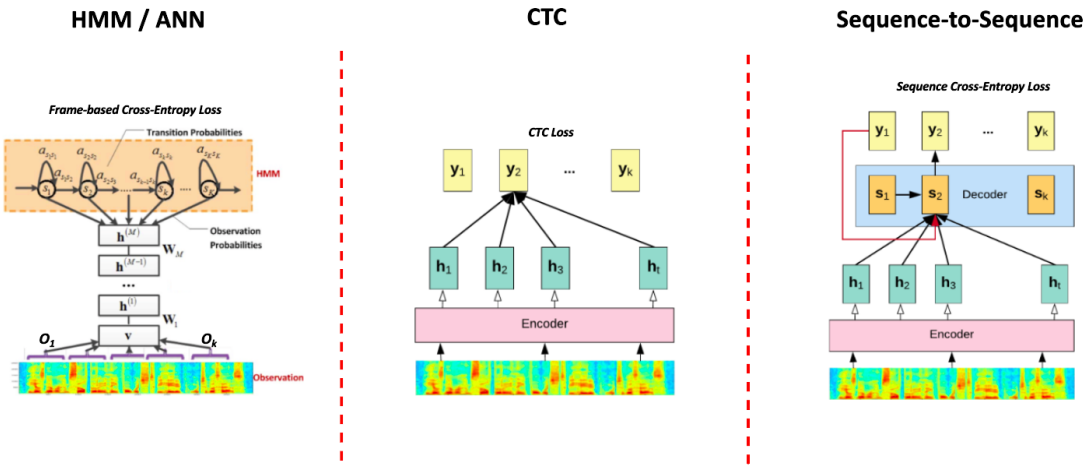


Figure 3.1: Three common approaches for solving ASR.

3.1.1 DNN Acoustic Model

In HMM-based ASR, Gaussian Mixture Model (GMMs) had been used for modeling the phone emission probabilities for a long time. Until the year 2010, [BM12, SLCY11] shown that these probabilities can be efficiently estimated by using several layers of feed forward networks or referred more commonly as deep neural network (DNN) in the literature. Different from GMMs, DNNs are trained in a discrimination manner to classify phones or phone states. The phone posterior probabilities for each frame is then computed by incorporating a prior term. As shown in many practices, this prior

term can be estimated with simply counting from the training dataset. This type of model is also referred as hybrid acoustic model in the literature.

3.1.2 Connectionist Temporal Classification Model

The connectionist temporal classification criterion (CTC) [GFGS06, GJ14] and its associated training methods have received significant interest in speech recognition in recent years. Using recurrent neural networks (typically long short-term memory LSTM), CTC training can efficiently model the long-term dependencies between a small number of units (e.g., phonemes or characters) and speech frames. Utilizing the CTC optimization criterion which handles possible alignments of the units in a sequence label, CTC-based speech recognition systems can be trained in a straight manner, thereby eliminating many complex steps in the conventional hybrid Hidden Markov Model / Artificial Neural Network (HMM/ANN) speech recognition pipeline, such as the definition of an HMM topology, finding context-dependent phonemes and modeling units, and the frame-wise alignment of HMM states and feature vectors. The way CTC was originally introduced motivates the development of such speech recognition systems in end-to-end fashion in which the language model or a vocabulary can also be omitted.

Assume that we use a set of labels L and we can always map the ground-truth transcript of an utterance X into a label sequence $z \in L^*$ (L^* meaning the Kleene closure over the alphabet L). A CTC path π (i.e., a sequence at frame level allowing repeated labels) represents an alignment of z. Denote y_t^π as the posterior probability that a recurrent neural network model assigns to the corresponding label of π at time t. By assuming the independent probabilities of all labels between frames, the CTC objective function solves all possible alignments as:

$$P(z|X) = \sum_{\pi} P(\pi|X) = \sum_{\pi} \prod_{t} y_t^\pi$$

For model optimization, [GFGS06, GJ14] proposed to use the forward-backward algorithm to maximize the likelihood of all the transcripts given the speech utterances

in a training corpus. After training converges, we obtain an optimized model to predict the posteriors y_t of all labels at every time frame t. The inference is then performed via a beam search with the integration of a language model which is similar to the HMM-based ASR except the withdrawal of hierarchical transitions in HMM model.

3.1.3 Attention-based Sequence-to-Sequence Model

To build an HMM-base ASR system, one needs to follow a complicated pipeline to create pronunciation dictionaries, HMM topologies, phoneme cluster trees, an acoustic model and a language model. Some of these components even require language expertises which pose considerable burden developing ASR for new languages. The CTC approach tried to simply the HMM pipeline by employing recurrent neural network (RNN) and CTC loss function. However, the CTC ASR system still requires to integrate an external language model for their inferences. A remedy to this is provided by attention-based sequence-to-sequence model (Seq2Seq) [CBS+15, CJLV15], another HMM-free approach to speech recognition that uses an encoder-decoder architecture.

The main components of the Seq2Seq model include an encoder, which consumes the source sequence and then generates a high-level representation, and a decoder generating the target sequence. The decoder models the data as a conditional language model - the probability of the sequence of discrete tokens is decomposed into an ordered product of distributions conditioned on both the previously generated tokens and the encoder representation. Both encoder and decoder are neural networks components that are able to learn the relationship between the time steps in the input and output sequence. The decoder also requires a mechanism to condition on specific components of the encoder representation.

Note several important differences to the HMM-based approach:

- P (W | X) is modeled directly without any noisy-channel assumption.
- All parameters are trained jointly. This simplify implementation and maintenance. On the other side, using auxiliary data such as monolingual data for language modeling becomes less straight-forward.
- No pronunciation dictionary is required, the creation of which is one of the major burdens in traditional speech recognition.

- This model is more flexible regarding output text normalization, e.g. can be trained to directly produce output that is properly cased, punctuated, and uses properly formatted numbers.

3.2 Extracting Features

3.2.1 Bottleneck Features

In HMM-based ASR, log mel filterbank (FBANK) and mel-frequency cepstrum (MFCC) are the most common techniques to produce frame-based input features from raw audio signal for modeling. Inspired from the representation learning ability of neural networks, [YS11, MKC+11, SKR12] have proposed a so-called bottleneck feature extraction (BNFs) to produce input features for the HMM-based models. BNFs employs several layers of feature forward networks with a very end bottleneck layer (before the final softmax layer). This bottleneck layer typically has a the same size as the number of feature in FBANK or MFCC extraction, and the BNF is trained to classify phone state labels. Several studies have shown that the BNF can transform the original input into a new feature domain by discarding irrelevant information, and easier for modeling with GMMs or DNNs. Using this feature extraction also results in better recognition performance, especially for GMMs.

3.2.2 Speaker Identity Vector

Speaker identity vector (I-vector) provides a short vector that describes a speaker's identity and are successfully used in speaker verification and speaker recognition tasks. This powerful technique is also useful for speech recognition since i-vectors encapsulate the speaker relevant information in a low-dimensional fixed-length representation [GBM+11]. The i-vector extraction is trained in an unsupervised manner on untranscribed data. Applied to speech recognition, Saon et al. [SSNP13] and Senior et al. [SLM14] augment regular acoustic features with i-vectors as a speaker adaptation for their DNN systems. Their works showed that i-vectors possibly provide additional information allowing for an improving recognition performance. Miao et al. [MZM15] introduced speaker adaptive training for DNN (SAT-DNN) which learns an adaptation neural network to convert i-vectors to speaker-specific linear feature

shifts. The original features (e.g. MFCC) are then speaker-normalized by adding theses shifts.

3.3 ASR Progress

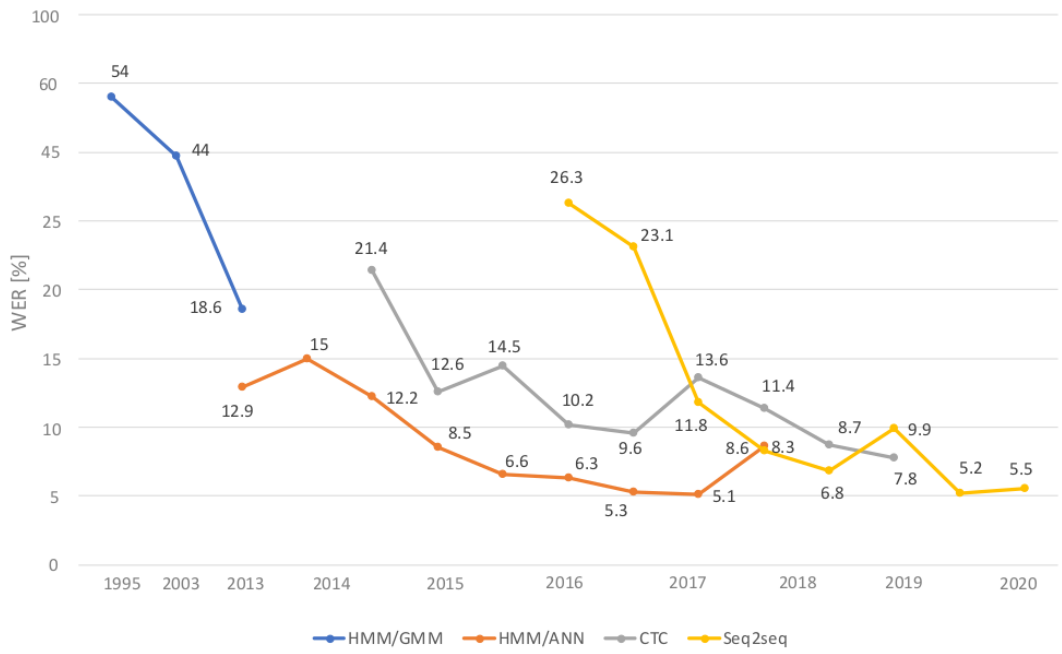


Figure 3.2: WER over years of different ASR approaches.

Since 2010, the research on automatic speech recognition has gained significant interest from various research groups and companies. This fact pushes forward the research development of ASR and resulted in a massive improvement in recognition performance. In many speech recognition benchmarks, machines have already outperformed humans by word error rate accuracy. To give an overview of this success, we review and analyze the progress of ASR research on the Switchboard telephone conversation speech.

Switchboard telephone conversation speech is one of the most popular benchmarks in ASR. This corpus contains spontaneous speech with frequent appearance of disfluency, which is a very challenging recognition task. In Figure 3.2 and Figure 3.3, we show the development of recognition accuracy from several works reported by

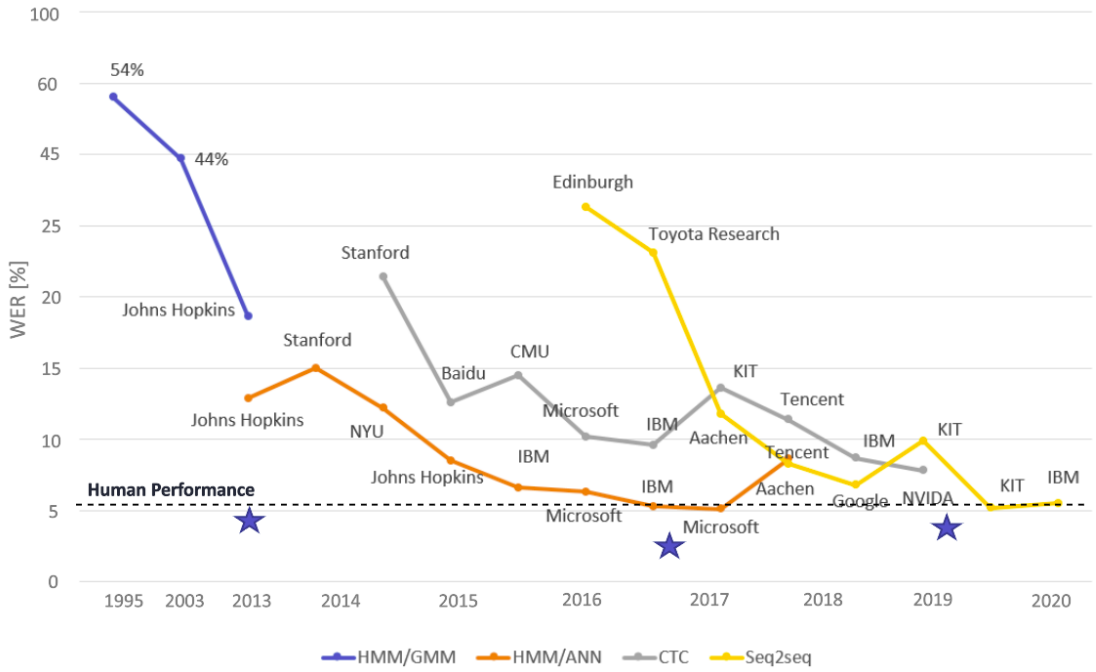


Figure 3.3: Three milestones in ASR progress.

many different research groups and companies. We category the reported results by their modeling approaches, including HMM/GMM, HMM/ANN, CTC-based, and Sequence-to-sequence ASR. When this dataset was first introduced, machines' recognition performance was above 50% and had not changed much over a decade. The significant changes have begun since 2010. We highlighted three milestones in this progress:

- **2012**: *Deep learning* started to show as a potential approach to solve ASR. The HMM-based ASR with the employment of DNN acoustic model outperformed the traditional GMM acoustic model.

- **2016**: Recognition accuracy produced by machines had reached to *human performance* level. Some big companies such as Microsoft and IBM have reported their success in achieving human performance in this benchmark and made a breakthrough in ASR. However, the results are hardly producible since they were produced by several systems and require intensive computing power.

- **2019**: Human performance was achieved again, but with single open-source systems and much simpler training procedure.

Chapter 4

Online and Low-latency ASR

In this chapter, we explain the differences between offline (or batch-mode) and online speech recognition. We further analyze the latency issues that occurred with the components of the current speech recognition architectures and describe a systematic approach for the development of an online speech recognizer.

4.1 Offline vs. Online

Automatic speech recognition is the ability of a machine program to identify words and phrases in spoken sentences and convert them to a human-readable format. In the statistical approach, this program usually refers to a machine learning model that learns a mapping function from audio data to phone, character, or word symbols. Typically, in both training and testing phases, the model is fed with complete audio data of spoken sentences to start its processing. There is also no constraint on when the model needs to produce its output. This setting is referred to as offline or batch-mode processing.

In many real-life applications, the speech recognition program needs to respond quickly to catch up with the verbal interactions. Sometimes, the program must be able to start processing before the users have completed a sentence. This type of application is referred to as online speech recognition, differed from offline in that latency is an additional optimization constraint. An online recognizer may become not applicable if its latency does not satisfy users' needs. The real challenge of building an online

recognizer usually does not lie on the optimizations for the computational power but the latency issues introduced by the components of the speech recognition systems.

4.2 Latency Issues

The ASR studies in the literature has focused on offline (batch-mode) conditions. The batch-mode speech recognition models followed both hybrid HMM/ANN and end-to-end paradigms have their own latency issues when being in online condition. Our analyses on the latency of the common ASR systems are summarized as following:

Feature Extraction So far, most of state-of-the-art speech recognition models work on the extraction of log Mel filter-bank frequency (FBANK) or Mel-frequency cepstral coefficients (MFCC) features. To obtain the optimal configurations, these features are usually further normalized via Ceptral mean and variance normalization (CMVN) or Cepstral mean normalization (CMN) techniques for several samples of the same sentence or many sentences of the same speaker. However, in many online scenarios, there is no guarantee to have enough a certain amount of historical data to be available for the feature normalization. This situation then may leads to a mismatch between training and test, and degrade the recognition accuracy.

Bidirectional Encoding In many end-to-end speech recognition models which employ sequence training criterion such as connectionist temporal classification criterion or sequence-to-sequence encoder-decoder, a neural network architecture is typically used to encode the acoustic input sequence into a high-level representation. To achieve high performance, bidirectional LSTM have been the optimal choice for the encoder of many end-to-end models. However, due to the backward LSTM, bidirectional LSTM are *not* suited to provide partial and low-latency output as needed for streaming recognizers. The addition of acoustic input will affect all of the encoder's hidden states, which then makes all partial inference results unstable. This effect leads to the fact that stable output can be confidently inferred only when the input is complete.

Soft-attention Mechanism The introduce of soft-attention mechanism make a significant change of building end-to-end sequence-to-sequence attention-based models. However, [RLL+17, CR17] pointed out early that the shortcoming of an attention-based S2S model used in online condition lies in its attention mechanism, which must perform a pass over the entire input sequence for every element of the output sequence. [RLL+17, CR17] proposed a so-called monotonic attention mechanism which enforces a monotonic alignment between the input and output sequence.

Beam Search Beam search is the most efficient approach for the inference of both end-to-end or HMM/ANN models. Its basic idea is to maintain a search network in which network paths are extended with new nodes with the highest accumulated scores and to then prune the network only keeping a set of active paths (or hypotheses). Typically, the most probable hypothesis for an utterance X is found and guaranteed when the entire search space constructed from X is supplied to the search. However, needing the complete acoustic signals of X to its very end in order to output the inference result is not efficient for a streaming setup. A streaming recognizer must be able to produce partial output while processing partial input.

4.3 How to Build an Online Recognizer

The difference between offline and online ASR is the appearance of the latency constraint. As the analysis in the previous section, many components in the current ASR approaches require entire acoustic signals of input utterances obtain their optimal performance. When switching to more latency-friendly counterparts, the recognition performance may reduce dramatically (e.g. from bidirectional to unidirectional LSTM). So to build online recognizers, we mostly need to improve the latency of the ASR components while maintain their capacity for the accuracy performance. An ideal online recognizer should perform as good as the offline counterpart of the same setting while being able to serve in real-time and low latency.

We propose a two-stage approach to develop online streaming speech recognizers. In the first stage, we constructed and evaluated different neural network models to build high-performance recognition systems in offline condition. In the second stage,

we explored further techniques that enable the high-performance neural network models for online streaming processing while maintaining their efficiency as in offline. The success of the two-stage approach resulted in a speech recognition system that satisfies the need for both latency and accuracy.

Chapter 5

Experimental Setup

In this chapter, we will outline our experimental setup. Starting with data sets, we will also provide an overview of the tasks used for evaluation, as well as short descriptions of all the toolkits used.

5.1 Datasets

In this work, we used data sets from 3 different projects, covering a wide variety of acoustic conditions. The most challenging data set originates from the BABEL project and consists of telephone recordings. Data from Euronews is of a better acoustic quality, but the available annotations are limited. Data sets used from the BULB project contain only very little data, which is challenging as well.

5.1.1 Switchboard

Switchboard is a collection of about 2,400 two-sided telephone conversations among 543 speakers (302 male, 241 female) from all areas of the United States. A computer-driven robot operator system handled the calls, giving the caller appropriate recorded prompts, selecting and dialing another person (the callee) to take part in a conversation, introducing a topic for discussion and recording the speech from the two subjects into separate channels until the conversation was finished. About 70 topics were provided, of which about 50 were used frequently. Selection of topics and callees

was constrained so that: (1) no two speakers would converse together more than once and (2) no one spoke more than once on a given topic.

5.1.2 Fisher

The Fisher telephone conversation collection protocol was created at LDC to address a critical need of developers trying to build robust automatic speech recognition (ASR) systems. Previous collection protocols, such as CALLFRIEND and Switchboard-II and the resulting corpora, have been adapted for ASR research but were in fact developed for language and speaker identification respectively. Although the CALLHOME protocol and corpora were developed to support ASR technology, they feature small numbers of speakers making telephone calls of relatively long duration with narrow vocabulary across the collection. CALLHOME conversations are challengingly natural and intimate. Under the Fisher protocol, a large number of participants each calls an other participant, whom they typically do not know, for a short short period of time to discuss the assigned topics. This maximizes inter-speaker variation and vocabulary breath while also increasing formality.

5.1.3 Hub4

The 1996 Broadcast News Speech Corpus contains a total of 104 hours of broadcasts from ABC, CNN and CSPAN television networks and NPR and PRI radio networks with corresponding transcripts. The primary motivation for this collection is to provide training data for the DARPA "HUB4" Project on continuous speech recognition in the broadcast domain.

5.1.4 Libri Speech

LibriSpeech is a corpus of approximately 1000 hours of 16kHz read English speech, prepared by Vassil Panayotov with the assistance of Daniel Povey. The data is derived from read audiobooks from the LibriVox project, and has been carefully segmented and aligned. This dataset contains 1000 hours of speech sampled at 16 kHz. We have made the corpus freely available for download, along with separately prepared language-model training data and pre-built language models.

5.1.5 TED Talks

English speech recognition training corpus from 2351 TED talks (452 hours of audio), created by Laboratoire d'Informatique del'Université du Maine (LIUM). A TED talk is a video created from a presentation at the main TED (technology, entertainment, design) conference. This corpus released with a Dictionary with pronunciations (159848 entries) and selected monolingual data for language modeling from WMT12 publicly available corpora.

5.2 Toolkits

Janus Recognition Toolkit Our traditional speech recognition systems were built using the Janus Recognition Toolkit (JRTk) [FGH+97] which features the IBIS decoder. It is being developed at the Carnegie Mellon University (CMU) and at the Karlsruhe Institute of Technology (KIT). In addition to training ASR systems, we also used the audio pre-processing pipeline of JRTk for the extraction of features for all our experiments.

Theano Feed-forward neural networks were trained using a setup based on Theano [BBB+, BLP+12]. Theano being recently discontinued at the time of writing was one of the first toolkits with support for automatic differentiation. It allowed for writing code in Python which would then be compiled to run on either CPUs or GPUs. While this resulted in fast execution, the drawback of this approach is that runtime debugging required special methods.

PyTorch PyTorch [PGM+19] is a novel machine learning library, which provides Python bindings to Torch. It does not require a special compilation step like Theano. Being more recent, it also features a better integration with up-to-date CUDA 1 versions which result in faster processing times. It was used for training the RNN/CTC based ASR systems.

Chapter 6

Feature Extraction

Speech signals, known as the convolution of multiple frequency components in a wide dynamic range, before becoming digital forms, can be changed dramatically with natural factors, such as different speakers, environments, or recording tools. The large variability of speech signals typically causes the mismatch between training and testing conditions, and then may largely degrade the performance of speech recognition models. To address this issue, efficient techniques [Gal98, GMMW13] in feature extraction have been proposed. While [Gal98] used feature-space maximum likelihood linear regression (fMLLR) to reduce speaker variability in feature space, [GMMW13] proposed a bottleneck network architecture for producing a robust feature representation. We advanced this direction by proposing three novel network-based feature extraction approaches.

In the first approach, an efficient architecture used to combine the benefits of both i-vectors and speaker-adaptive feature transformations was introduced. In the second approach, we presented a neural network optimized with connectionist temporal classification criterion can be used a high-level feature extraction for HMM/ANN models. We showed that the common CMVN feature extraction introduces a postental mismatch between offline training and online testing, and proposed a novel approach to overcome it.

6.1 Speaker Adapted Features

In statistical speech recognition, speaker adaptation techniques can fall into two categories: Model adaptation involves modifying the parameters of the acoustic model to fit the actual speech data from a target speaker. Maximum Likelihood Linear Regression (MLLR) [Gal98] and Maximum A Posteriori (MAP) [GL94] are the powerful model adaptation techniques that improve Gaussian Mixture Models (GMMs). However, there is no similar technique for Deep Neural Network (DNN) models which have become prominent in recent years. Due to their many large hidden layers, DNNs have a significantly higher number of parameters. It is therefore hard to adapt DNNs with only a small amount of data. Several studies [Lia13] [SLCY11] have shown that DNN models have greater invariance to speaker variations resulting in model adaptation being less effective than for GMMs. Further, model adaptation usually results in new models for individual speakers, significantly increasing complexity and required storage space.

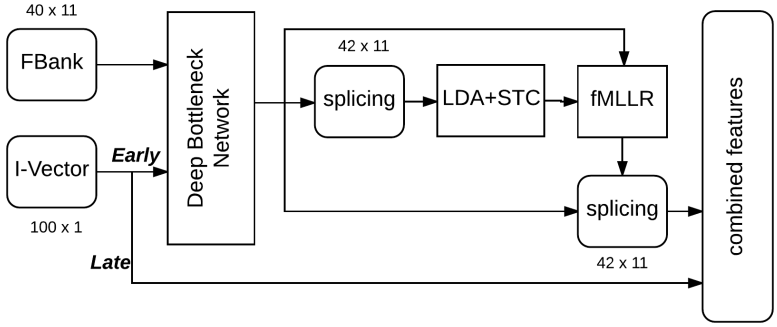
Unlike model adaptation, feature adaptation techniques use regular acoustic features and adaptation data to provide new features which better fit the trained acoustic model, thus improving recognition accuracy without the need to change the model. Feature adaptation is attractive for dealing with the limitations of model adaptation, especially for DNNs. Feature-space MLLR (fMLLR) [Gal98] is a well-known adaptation technique which makes better inputs for GMMs. However, providing good fMLLR features for DNNs is challenging: Due to the huge difference between DNN and GMM models, fMLLR features which are optimized for GMMs are not guaranteed to be better for DNNs than other regular features. Recently, identity vectors (i-vectors) for speaker representation have been introduced [DKD$^+$11], and have been successfully used in speaker verification and speaker recognition. Further research [SSNP13, SLM14] proved that i-vectors can be used in conjunction with regular features to improve DNN performance. A later study [SLM14] showed only small improvements if using a strong DNN baseline.

We examine how i-vectors and fMLLR transformations can be combined in order to improve both GMM and DNN systems. In particular we analyse speaker-adaptive bottleneck features (SA-BNF), where log scale Mel filterbank (FBANK) features are concatenated with i-vectors to form their input features and investigate how both

speaker-adaptive bottleneck features and speaker-independent bottleneck features can be further transformed and augmented before being used as DNN or GMM input features.

6.1.1 Feature Extraction Pipeline

Figure 6.1: Hierarchical combination of bottleneck, fMLLR and i-vector features for either early or late combination.



An overview of our proposed feature extraction process is shown in Figure 6.1. In particular, we employ the DBNF architecture described by Gehring et al. [GMMW13], which is constructed as a stacked denoising auto-encoder, comprised of a bottleneck, a hidden layer and the classification layer. The stacked auto-encoder is first pre-trained layer-wise [VLBM08], then the whole network is fine-tuned to discriminate target phoneme states. After the optimization the last two layers are removed resulting in a network that can transform acoustic features into effective speaker-independent bottleneck features (SI-BNF).

Early I-Vectors Having a similar architecture to DNNs, DBNFs are also capable of modeling high-dimensional correlated input features. We investigate the ability of incorporating acoustic features and i-vectors to train DBNFs. In our approach, regular acoustic features (e.g. FBANK) are spliced for some consecutive frames and then concatenated with i-vector features to be fed into DBNFs. After the training, we are able to build speaker-adapted bottleneck neural networks which can extract speaker-adapted bottleneck features (SA-BNF).

fMLLR fMLLR transformations are typically applied after performing LDA and STC transformations on Mel-frequency cepstral coefficients (MFCC) [RPVČ13] or on bottleneck features [GMMW13]. These transformations are known to make inputs more accurately modeled by GMMs. We observe that LDA and STC techniques can be less effective for DNNs, e.g. MFCC or LDA is usually not good as FBANK. As a further evaluation, we estimate fMLLR transformation directly on SI-BNF or SA-BNF without using LDA and STC transformations.

Late I-Vectors After applying fMLLR transformations, new transformed features are supposed to have less speaker variability. Providing again speaker information with i-vectors can lead to improvement as suggested from [SSNP13]. We also concatenate the transformed SI-BNF or SA-BNF with i-vectors for different combinations.

6.1.2 Experimental Setup

We used a large training dataset of 460 hours from 12000 English talks. This dataset includes the TED-LIUM [RDE14a], Quaero [SKK12a] and Broadcast News [Gra97] corpora. The development set including 8 speakers is taken from TED-LIUM corpus. Since the test set included in TED-LIUM is rather small, we instead used the tst2013 set from the IWSLT evaluation campaign which is much bigger with 27 speakers.

The DBNFs were constructed with 6 hidden layers containing 2000 units and a 42 units bottleneck layer, using input as 11 stacked frames of 40-dimensional mel scale filterbank coefficients with or without concatenating i-vector features. All the DNN models also share the same architecture which has 6 hidden layers with 2000 units per layer. The input of the DNNs is 11 stacked frames of 42-dimensional transformed SI-BNF or SA-BNF, with or without combining i-vector features. We used sigmoid activation for hidden layers and soft-max for output layer.

DNN and DBNF systems were trained using cross-entropy loss function to predict 8000 context-dependent states. The same training method is applied for all DNNs and DBNFs, which includes pre-training with denoising auto-encoders and followed by fine-tuning with back-propagation. We used an exponential schedule for all of the trainings. The GMM models were trained using incremental splitting of Gaussians (MAS) [KFN98] and followed by optimal space training (OFS) (a variant of STC [Gal99]) if LDA features are used.

To extract i-vectors, a full universal background model (UBM) with 2048 mixtures was trained on the training dataset using 20 Mel-frequency cepstral coefficients with delta and delta-delta features appended. The total variability matrices were estimated for extracting 100-dimensional i-vector which was observed to give the optimal recognition performance in [SSNP13] [SLM14]. The UBM model training and i-vector extraction were performed by using the sre08 module from the Kaldi toolkit [PGB+11a].

The GMMs trained with SI-BNF and SA-BNF were used to compute fMLLR transformations. The process of fMLLR estimation were performed as the traditional approach. During the training, we used the adaptation data of the same speaker and the reference transcriptions to do the alignment, while the same GMMs were used as first-pass systems to generate transcriptions in the testing.

6.1.3 Results with GMM and DNN Systems

We used a DNN system with FBANK features as the speaker independent baseline (SI-DNN). This is a strong baseline since DNNs training with mel scale filterbank is known to outperform other regular features [MHP12]. The other baseline is a speaker-adapted DNN (SA-DNN) using i-vectors. This baseline is similar to the speaker-adapted DNNs presented in [SLM14] except our i-vectors are extracted for speaker-level instead of utterance-level. The results of the baselines on the dev and test set are shown in Table 6.1. In our setup, we are able to reproduce the improvement when using i-vector adaptation for DNN systems in both the dev and test set. The improvement is not large as reported in [SSNP13], but is comparable to [SLM14] since we used a similar baseline setup.

Table 6.1: Word error rate of baseline systems.

Baselines	**tst2013 (dev)**
SI-DNN	16.2 (13.1)
SA-DNN	**15.1** (**12.6**)

GMM systems Table 6.2 presents the results of our evaluated GMM systems. The first three columns show the possible techniques applied to make inputs to the

GMMs. The techniques include *Early I-vector* for extracting speaker-adapted bottleneck features, followed by *LDA+STC* transformation, and *fMLLR* transformation at the last step. The last column presents word error rates (WER) on the both development set and the tst2013 set.

The results of the GMMs using SA-BNF are consistently better than using SI-BNF with identical constructions. The regular bottleneck GMM (with full transformation techniques) is 4-6 % less effective than the adapted bottleneck GMM. This shows that DBNFs can explore the adapted input with the addition of i-vector to provide better discriminative features.

It is worth noting that while the trained GMM systems have good performance, the best speaker-adapted GMM is even better than SA-DNN baseline. This indicates that feeding their input features to DNNs may improve systems due to the better capacity of DNNs in classification task.

Table 6.2: Comparison of word error rate for GMM systems.

Early I-vector	**LDA+STC**	f**MLLR**	**tst2013**
N	N	N	16.7 (13.7)
Y	N	N	15.8 (13.6)
N	N	Y	15.9 (13.4)
Y	N	Y	15.0 (12.8)
N	Y	Y	15.3 (12.9)
Y	Y	Y	**14.4** (**12.4**)

DNN Systems In Table 6.3, we present the results of the examined DNNs in the experiments. Again, the last column shows the results in word error rates, while the other columns indicates the usage of our proposed adaptation techniques.

Applying only *fMLLR* and *Early I-vector* for the DNNs (the first four systems in the table) shows significant improvements in tst2013 set. Improvements with speaker-adapted bottleneck features in the development set are less clear. However, using both *fMLLR* and *Early I-vector* can improve results up to 8% relative. As predicted, the performance of these DNNs largely outperform GMMs of the identical inputs.

When concatenating fMLLR transformed features again with i-vectors, we found the best features combination. The best DNN system with *Late I-vector* and *fMLLR*

Table 6.3: Comparison of word error rate for DNN systems.

Early I-vec	LDA+STC	fMLLR	Late I-vec	tst2013
N	N	N	N	15.2 (12.3)
Y	N	N	N	14.6 (12.4)
N	N	Y	N	14.4 (11.9)
Y	N	Y	N	14.0 (12.0)
N	N	Y	Y	**13.1** (**11.2**)
Y	N	Y	Y	14.1 (12.0)
N	Y	Y	Y	13.6 (11.1)
Y	Y	Y	Y	14.2 (11.3)

gives 15-19 % relative improvement over SI-DNN baseline and 11-13% over SA-DNN baseline. We could not however achieve further improvement with the DNNs by *Late I-vector* together with *Early I-vector* and *fMLLR*. That may be due to either fMLLR transformation not being able to completely remove speaker variability, or our used DNN architecture not being able to exploit this combined structure.

6.2 CTC-Network Derived Features

The connectionist temporal classification criterion (CTC) [GFGS06, GJ14] and its associated training methods have received significant interest in speech recognition in recent years. Using recurrent neural networks (typically long short-term memory LSTM), CTC training can efficiently model the long-term dependencies between a small number of units (e.g., phonemes or characters) and speech frames. Utilizing the CTC optimization criterion which handles possible alignments of the units in a sequence label, CTC-based speech recognition systems can be trained in a straight manner, thereby eliminating many complex steps in the conventional hybrid Hidden Markov Model / Artificial Neural Network (HMM/ANN) speech recognition pipeline, such as the definition of an HMM topology, finding context-dependent phonemes and modeling units, and the frame-wise alignment of HMM states and feature vectors. The way CTC was originally introduced motivates the development of such speech recognition systems in end-to-end fashion in which the language model or a vocabulary can also be omitted. However, to achieve state-of-the-art performance at par with the traditional hybrid HMM/ANN approach, an efficient decoding algorithm

that uses additional knowledge sources (e.g., vocabulary, pronunciation lexicon and language model) is still required to transform the posteriors of the units modeled by the CTC network into word sequences.

Because the training criterion of the CTC model is to maximize the log posterior $P(z|X)$ of the target label z given acoustic features, it does not necessarily optimize the final recognition when decoding with an additional language model. To the best of our knowledge, a decoding with a weighted finite state transducer (WFST) built over a pronunciation lexicon and an n-gram language model applied to CTC posteriors is still the most successful approach with the best word error rate (WER). As observed in [SdCQS+15, KLK17], the performance of a CTC system can be better than that of a hybrid system trained with the cross-entropy criterion but is less than that of a hybrid system optimized with sequence training.

To benefit from the strengths of the CTC network at label discrimination on the one side and the highly optimized decoding stack of conventional hybrid systems on the other side, we investigate the use of CTC posterior probabilities as input features in hybrid HMM/ANN system to boost speech recognition performance.

6.2.1 C-Phone Extraction

Assume that we use a set of labels L and we can always map the ground-truth transcript of an utterance X into a label sequence $z \in L^*$ (L^* meaning the Kleene closure over the alphabet L). A CTC path π (i.e., a sequence at frame level allowing repeated labels) represents an alignment of z. Denote y_t^π as the posterior probability that a recurrent neural network model assigns to the corresponding label of π at time t. By assuming the independent probabilities of all labels between frames, the CTC objective function solves all possible alignments as:

$$P(z|X) = \sum_{\pi} P(\pi|X) = \sum_{\pi} \prod_{t} y_t^\pi$$

For model optimization, [GFGS06, GJ14] proposed to use the forward-backward algorithm to maximize the likelihood of all the transcripts given the speech utterances

in a training corpus. After training converges, we obtain an optimized model to predict the posteriors y_t of all labels at every time frame t.

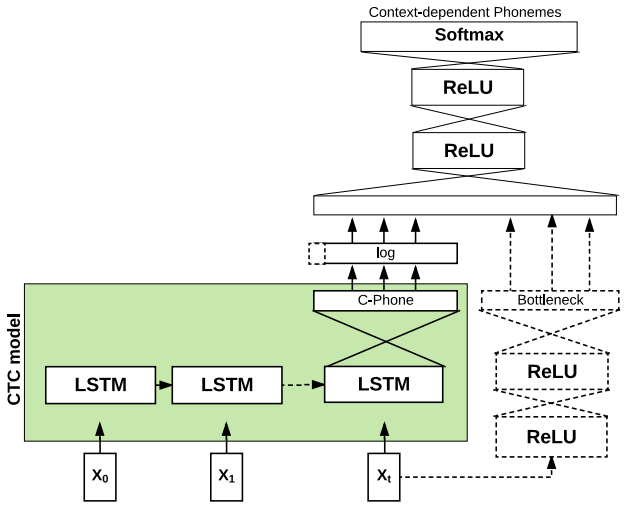
By using independent phones as the set of labels (e.g., 45 English phones), we consider y_t as a phone information vector (so-called *C-Phone*) which indicates the occurrence of the phones in the frames. Since the posterior probabilities extracted from the softmax output usually have sharp distribution [HES00, ZCMS04], we transform them to the log domain for better modelling. During CTC training, the $blank$ label must be introduced to allow the optional occurrences of regular labels in the alignments. The probability of the $blank$ can be eliminated, i.e. removed from the C-Phone, when extracting the C-Phone vector since it does not map to any real acoustic event and has little meaning.

Different from the extracted posterior features [HES00, ZCMS04] or bottleneck features [GKKC07] where the extracting models are trained with fixed Viterbi alignments (e.g., the model exactly learns a feature transformation), C-Phone extraction is trained without any prior alignment (the CTC model needs learns the alignment by itself based on the label sequences). As observed in [GFGS06, SSR+15, ZBSN17], the posteriors produced by the CTC model have peaky behaviors in which $blank$ has the highest probability in almost all frames, except for short peaks where regular labels dominate. This raises the question whether the phone probabilities assigned by the CTC model still correlate to the fixed labels of a traditional Viterbi alignment. In this study, we try to address this question by learning a feed-forward network transformation to bridge between C-Phones and the context-dependent phones labels in the conventional HMM system set-up.

6.2.2 Using C-Phone Features

Figure 6.2 illustrates how we investigated the C-Phone features. We trained a CTC system with an LSTM model for feature extraction as explained in Section 6.2.1. The trained LSTM model is then used to produce posterior vectors for every frame. These posteriors are transformed into the log domain and the probability of the $blank$ label is eliminated to form the final C-Phone feature vectors. These features can be directly fed into a feed-forward network model to build a conventional hybrid HMM/ANN system.

Figure 6.2: Extracting and using C-Phone.



Furthermore, C-Phone features can also be augmented with additional features such as network-based bottleneck features.

6.2.3 Experimental Setup

Our experiments were conducted on the Switchboard-1 Release 2 (LDC97S62) training corpus which contains over 300 hours of speech. The Hub5'00 evaluation data (LDC2002S09) was used as test set. We used a 4-gram language model which was trained on the transcripts of the training data (3M words) and the transcripts (22M words) from the Fisher English Part 1 (LDC2004T19) and Part 2 (LDC2005T19) corpora. We further used the pronunciation dictionary that came with the Switchboard Corpus.

All our systems were trained on the same training data and use the same vocabulary and 4-gram language model. The dictionary used for decoding includes 43 English phonemes and 2 noise models. For the CTC training, $blank$ is used as additional label while for the hybrid HMM/ANN system we use $silence$ instead.

The CTC systems used for C-Phone extraction was trained with Eesen [MGM15]. We used a bi-directional LSTM with 5 layers of 320 units, and a uni-directional LSTM containing 640 units per layer. The training schedule adopts an initial learning rate

of 0.00004 for every training. A decay with a factor of 4 was applied when the cross validation error degraded after 12 epochs.

A FFNN architecture of 7 layers of 1600 units is used for all hybrid HMM/ANN systems. The training of FFNN models uses new bob learning rate schedule with an initial rate of 0.02. Similar to other FFNNs, the bottleneck extraction network is also trained on 11 frames of log mel filter-bank features which are normalized per conversation. The bottleneck layer contains 40 units which is the same as the number of filter-bank coefficients. The extraction network also has 7 layers and the 2 last layers are removed after the training. A feature-space Maximum Likelihood Linear Regression (fMLLR) transformation was estimated from the manual transcripts during the training and from high-confidence decoding transcripts during testing.

6.2.4 Results with C-Phone Features

Table 6.4 compares the results of multiple systems using C-Phone features against conventional hybrid systems with log mel filter-bank (FBank), bottleneck features (BNF) and fMLLR features which are estimated on top of the BNF. For fair comparison, all the systems share the same feed-forward neural network (FFNN) architecture for classifying 8000 context-dependent phonemes on a fixed Viterbi alignment. We use a popular FFNN as the baseline which was trained with the cross-entropy criterion on 11 frames of FBank coefficients. The referenced CTC system is trained using Eesen [MGM15] and also uses Eesen's WFST functionality to decode on the same posteriors as used for C-Phone extraction. The same 4-gram language model is employed in all systems. The results are reported on the full Hub5'00 test set. We noticed in our experiments that our baseline CTC system performs slightly better than a very similar system recently reported in [ARS+17].

We experimented with 3 variants of C-Phone features. The first variant is the direct posterior probabilities (C-Phone-P) while the second variant (C-Phone-L) is obtained after transforming the softmax output to the log domain. The third variant (C-Phone-NB) is the same as C-Phone-L before eliminating the probability of the $blank$ unit. In our setup, the training of the FFNN systems on C-Phone-P features did not converge. However when we switched to C-Phone-L or C-Phone-NB, our training

Table 6.4: Performance in word error rate (WER) of multiple HMM/ANN systems with different input features such as C-Phone, FBank, BNF, and fMLLR-BNF.

Model	Features	Window	Hub5'e (SWB)
FFNN	FBank	11	22.4 (15.8)
CTC	FBank	-	19.9 (14.1)
FFNN	C-Phone-P	-	-
	C-Phone-L	1	19.3 (13.7)
	C-Phone-L	7	19.0 (13.6)
	C-Phone-L	11	**18.9 (13.5)**
	C-Phone-L	15	19.3 (13.8)
	C-Phone-NB	1	19.3 (13.8)
	C-Phone-NB	7	19.0 (13.6)
	C-Phone-NB	11	19.1 (13.6)
	BNF	1	22.7 (16.0)
	BNF	7	21.8 (15.3)
	BNF	11	21.5 (15.1)
	BNF	15	21.5 (15.1)
	fMLLR-BNF	11	21.0 (14.6)
GMM	C-Phone-L	1	20.9 (15.7)
	C-Phone-L	11	20.0 (14.5)
	BNF	11	22.1 (15.7)

converges well for all inputs of different context sizes and without applying further feature normalization techniques.

Even when using only a single C-Phone vector as input, an FFNN can even be trained well. This reveals an additional aspect to the peaky behavior observed in CTC training [GFGS06, SSR+15, ZBSN17], e.g., even for the frames when no (regular) label has its peak probability, the posteriors vector still contains meaningful information for classifying phonemes (or even context-dependent phonemes) labeled in the fixed alignment manner.

Interestingly, the performance of the systems with C-Phone-L and C-Phone-NB are almost identical for the same configurations. This may indicate that the probability of the $blank$ does not carry any useful information for phoneme classification, and thus can be eliminated during decoding. This observation consolidates the identification in [CZQY17].

In terms of word error rate (WER), the FFNN systems trained on C-Phone

outperform FBank by a large margin (15.6% rel.) and clearly improve over the other network-based extracted features such as BNF or fMLLR. The improvement of stacking longer context of C-Phone vectors appears small but is still effective. The extracted C-Phone features also show their usefulness over other features when switching to GMMs instead of FFNNs.

In our experiments, the FFNN systems trained on the C-Phone outperform the referenced CTC system even when using only one feature verctor per frame. It is worth noting that the superior performance of the FFNNs is observed here with cross-entropy training (further improvement is expected when optimizing the FFNNs with sequence training). This result can be explained by either the introduction of context-dependent phonemes, that helps improving the classification, or by the current decoding approach of the CTC system not being as good as that of conventional HMM system.

6.2.5 Results with Feature Stream Combination

As shown previously, C-Phone is a compact vector which contains excellent features for phonemes classification. As a typical approach of feature engineering, one can wonder if the recognition performance can be further improved by combining additional features to C-Phone. In this section, we investigate the combination of C-Phone and FBank, BNF and fMLLR features. Table 6.5 presents the results of different combinations. We report only with C-Phone-L features but other variants have the same results. The *Window* column shows the number of consecutive C-Phone vectors and additional feature vectors (+*Feature*) fed into the FFNNs. Basically, we allow only two different input streams and the center of the context window is always the current frame.

Combining C-Phone with FBank features has almost the same result as using single C-Phone features. This result is different from [ZCMS04] where the combination of PLP features and their derived multiple layer perceptron (MLP) features gave improvements. This indicates that C-Phone does not need the complementary information from the original speech features for phoneme classification.

We found that the other network-based extracted features such as BNF features can supplement C-Phone and result in a better recognition performance (4.2% rel.). This

Table 6.5: Results in WER of different feature combinations.

+Features	Window	Hub5'e (SWB)
FBank	1/1	23.0 (17.7)
	3/3	18.9 (13.6)
	5/5	19.1 (13.7)
	1/5	19.1 (13.7)
BNF	1/1	18.4 (13.1)
	2/2	18.2 (12.9)
	3/3	18.4 (13.1)
	5/5	18.6 (13.3)
	1/5	18.5 (13.1)
fMLLR-BNF	1/1	**18.1 (12.8)**
	2/2	18.2 (13.0)
	3/3	18.2 (13.1)
	5/5	18.3 (13.2)
	1/5	18.2 (12.9)

can be explained by the fact that BNF which is extracted from a wide context window contains additional information for context-dependent phoneme classification.

When transforming BNF into the fMLLR feature space which has less speaker variability, we achieved remarkable result (see Table 6.4). However, the recognition performance stays more or less the same when combining with C-Phone with BNF or fMLLR features. This observation can be explained as the analysis in [ZCMS04] where the extracted posteriors features reduce the variation among speakers, and thus have similar effects as fMLLR.

In many modern speech recognition systems, i-vector [DKD+11] which contains the information about the speaker and environment in a short vector usually helps supplementing the traditional features such as FBank or fMLLR [SSNP13, SLM14] in a speaker adaptation manner. Unfortunately, we were not able to provide results with i-vector adaptation due to our i-vector training setup could not employ Switchboard (or also Fisher) corpus to produce efficient i-vectors. We also found the same observation as reported in [MJZM14].

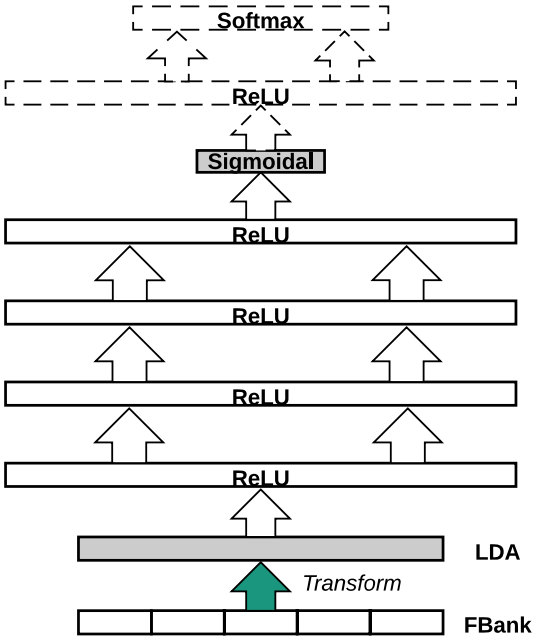
6.3 Real-time Feature Normalization

Ceptral mean and variance normalization (CMVN) [VL98] and other normalization techniques (e.g., Cepstral mean normalization (CMN) [Fur81]) are widely adopted in many neural network speech recognition systems due to several advantages. First, these techniques as shown in [VL98] make the recognizer more robust by canceling out environmental changes. Second, they help reducing the environment mismatch (e.g. background noises or microphones) between training and testing conditions. Last, the acoustic features after normalization have zero mean which is found critical for neural network training [LBOM12].

In offline situations, CMVN is usually applied at the utterance level or more ideally at the speaker level when many utterances of the same speaker are available. However, these approaches are not appropriate for real-time situations, because they require a certain amount of history to be available for the current speaker, and cannot handle unexpected speaker changes. Instead, mean and variance can be continuously computed over a moving window of some hundred frames (e.g., 3 seconds [PMN06, AOKO11]). However, moving windows require the availability of historical data of at least a window-size, so that a delay must be introduced to handle the beginning of a new utterance. A third approach, computing mean and variance globally (e.g., [ZSN16, SKMR13]) for all training and test data, avoids the delay but reduces the recognition performance due to potential data mismatch. CMVN can also be recursively updated in real-time as in [PMN06], but this approach does not handle multiple speakers. Peddinti et al. [PPK15] proposed to use mel-frequency cepstral coefficients (MFCC) without normalization for real-time speech recognition, as currently implemented in the Kaldi toolkit [PGB+11b]. In their approach, i-vectors [DKD+11] which supply the information about the mean offset of the speaker's data are provided to every input so that the network itself can do feature normalization. However, i-vectors still require a certain amount of data of about 6 seconds per speaker.

We investigated and employed the feature extraction methods which exhibit comparable performance to CMVN but do not require speaker historical data and are therefore better suited for real-time situations.

Figure 6.3: Real-time feature extraction.



6.3.1 Feature Extraction Pipeline

LDA Features The most popular acoustic features such as MFCC or FBANK without normalization are problematic input for neural networks to learn. MFCC features usually span a wide range in every dimnension, e.g., [-93, 363] on typical data, while FBANK features only have positive values, e.g. in the range $[0, 11.66]$. We attempt to find a transformed domain such that the transformation can be performed in real-time. Linear Discriminant Analysis (LDA) [DHS00] is usually used for dimensionality reduction, but here we propose to use it only for feature transformation. Using LDA, we compute a $d \times d$ linear transformation matrix which projects d-dimensional *FBANK* into a new domain with the same dimensionality. In this LDA domain, the features maintain the class-discriminatory information and can be mapped with their class-separability magnitudes according to the associated eigenvectors and eigenvalues. When used for dimensionality reduction, LDA is applied by keeping only k (much smaller than d) features with largest magnitudes. We, however, use all d-dimensional features in our in models because we observed better system performance.

Normalized Bottleneck Features As will be experimentally shown, optimizing single network models on unnormalized data can be hard. Dealing with this situation, our idea is to train a first network model for extracting *length-normalized features*. Later we can use a second network to perform the real classification task. Figure 6.3 illustrates our proposed feature extraction architecture. The input of the network can be unnormalized FBANK or LDA-transformed features. We employ some rectifier [ZRM+13] layers on top of the input layer, followed by a narrow (bottleneck) layer of 42 sigmoidal units. Two last layers which will be discarded after the training include one rectifier and the final softmax. Since the training of this feature extraction optimizes phonemes classification, the extracted features at the bottleneck layer are supposed to be significant for class-discrimination. When using a sigmoidal activiation function, we can obtain bottleneck features that are normalized to be in a small range which can be easier handled by the second network. We experimented with sigmoidal functions, the logistic function which has range if [0,1] and the hyperbolic tangent which produces features in range [-1,1].

Different from [GMMW13, YS11], the proposed feature extraction is able to handle both normalized and unnormalized inputs. It does not suffer from vanishing gradients and does not need pre-training which significantly reduces the training time. Applying this feature extraction in real-time can be considered as adding more hidden units to the classification network, which linearly increases the computation time (i.e. 25% in our experiments).

6.3.2 Experimental Setup

The dataset is the result of combining TED-LIUM [RDE14a], Quaero [SKK12a] and Broadcast News [Gra97] corpora. Our three evaluation sets include TED-LIUM test, tst2013 from the IWSLT evaluation campaign [CNS+13] and the English set from the MSLT corpus [FL16] which contains conversations over Skype.

The volume perturbations were done as suggested by [PPK15] where each recording was scaled with a random variable using *sox*. We set the random variable within the range [0.2,2] for all recordings in the training data set. Then they were added to the original training set to form the augmented dataset. To investigate the

robustness against volume mismatch, we used the ranges [0.2, 0.6] and [1.6, 2.0] for the all recordings of the tst2013 set to create a perturbed test set.

All the network models used roughtly same number of input features (i.e, 440 FBANK and 462 LDA or bottleneck features) and were trained using the cross-entropy loss function to predict 8,000 context-dependent phonemes. Rectifier networks were constructed of 6 hidden layers with 1,600 units per layer. For sigmoidal networks, we used 5 hidden layers of 2,000 units and performed pre-training with denoising auto-encoders [VLBM08]. For our convolution neural network (CNN), we used the best architecture from [SMKR13] which includes two convolutional layers of 256 hidden units with filter size 9 and a max pool size of 3, followed by 4 fully connected layers with 1,024 units. However, we did not use delta and delta-delta features for consistent comparisons between models.

6.3.3 Results with Unnormalized Features

In Table 6.6, we compare the systems using different CMVN methods against various systems trained on unnormalized FBANK features. Using our training data, CMVN systems performance depends on the amount of available speaker historical data. Normalization at speaker level yielded the best performance, followed by utterance level normalization and normalizations with windows 300 frames in length. The results on the perturbed test set show an interesting fact that these normalizations produce robust features to the changes of audio volume. Global CMVN is less optimal than other normalizations (7.1% rel. increase in WER compared to speaker level). However, real-time system may have to adopt this method, in order to achieve acceptable latency.

For the normalized features, the gap between sigmoidal and rectifier [ZRM+13] networks appears small. However, when using the features without normalization which have only positive values in a large range [0, 11.66], optimizing sigmoidal networks for good convergence becomes difficult. We had to reduce the initial learning rate by a factor of ten compared to normalized features. The training then converged at a poor local minimum and caused worse classification performance. The situation changed with the rectifier network. We were able to keep the same learning rate and the training converged with the same pattern. However, it suffers from a 7.3% rel.

increase in WER compared to global CMVN. Switching to a CNN network gave a further improvements, however its result is still not good as that of the CMVN systems. These results demonstrate the difficulties when training single network models on unnormalized FBANK features.

The increase in WER of the systems using unnormalized features and global normalized features on the perturbed test set indicates that they may be sensitive to volume mismatch between training and test data.

Table 6.6: Word error rates of various systems using 40 log mel-filter bank features with and without CMVN

CMVN	Network Type	tst2013	tst2013-vp
Speaker	sigmoid	15.5	15.5
Utterance	sigmoid	15.8	15.8
Window	sigmoid	16.2	16.4
Global	sigmoid	16.6	17.3
Global	rectifier	16.5	17.1
none	sigmoid	22.3	23.2
none	rectifier	17.7	18.0
none	rectifier (CNN)	17.1	17.6

6.3.4 Results with LDA Features

Table 6.7 compares the efficiency of different LDA transformations applied to unnormalized features. Such a conventional approach (e.g. [RPVČ13]) which reduces dimensionality of 440 features of 11 consecutive frames down to 42 and then stacks again for 11 frames, does not show clear improvements. When transforming 40 FBANK features without reduction and stacking 11 adjacent frames of LDA features as the network input, the systems improved. Further improvement was achieved when transforming 440 features of 11 consecutive frames via LDA and using them as network input. Interestingly, the transformed features which are in the range [-14.95, 14.50] without zero-mean are better than FBANK with global CMVN. When applying global mean and variance normalization again on these LDA features, the performance even got worse showing that the normalization is unnecessary for this training data.

The large degradation (5.4% rel. in WER) of the performance on the perturbed test set presents the need of a method for improving LDA features against possible

environment mismatch.

Table 6.7: Results of the systems using LDA features.

LDA Feature	CMVN	DNN	tst2013	tst2013-vp
Reduction	none	rectifier	17.5	17.8
Full-40	none	rectifier	16.8	17.4
Full-440	none	rectifier	**16.2**	**17.0**
Full-440	Global	rectifier	16.5	17.2
Full-440	none	sigmoid	16.8	17.7

6.3.5 Results with Normalized Bottleneck Features

The proposed bottleneck feature extraction shows its advantages when applied to both unnormalized FBANK and LDA features and produces improved features. The same networks trained on the bottleneck features showed relative reduction of 7.4% and 4.9% as shown in Table 6.8. The extracted bottleneck features are in a normalized range [0, 1] or [-1, 1], so a sigmoid network can be trained well showing again that we do not need to apply mean normalization.

When evaluating against the mismatch test set, we found that the extracted features are more stable to speech variations indicating the normalized bottleneck network may be automatically forced to learn robust features.

Table 6.8: Results of the systems using normalized bottleneck (BN) features.

Feature	BN Type	DNN	tst2013	tst2013-vp
FBANK	*sigmoid*	rectifier	16.4	16.6
FBANK	*sigmoid*	sigmoid	16.5	16.8
LDA	*sigmoid*	rectifier	**15.5**	**15.8**
LDA	*tanh*	rectifier	15.5	15.8

Chapter 7

Data Augmentation

In automatic speech recognition, data augmentation has been used for producing additional training data in order to increase the quality of the training data, i.e. their amount and variety. This then improves the robustness of the models and avoids overfitting.

As in [KTO, RKRG], both unsupervised and artificial training data has been augmented to improve HMM/ANN ASR model training in low-resource conditions. The addition of training data with perturbation of the vocal tract length [JH13] or audio speed [KPPK15] helps models to be robust to speaker variations. Simulated far-field speech [KPP$^+$] and noisy speech [HCC$^+$14] have been used to supplement clean close-talk training data.

Sequence-to-sequence attention-based models [CBS$^+$15, CJLV15] were introduced as a promising approach for end-to-end speech recognition. Several advances [CSW$^+$, ZISN, WCW$^+$] have been proposed for improving the performance of S2S models. While many works focus on designing better network architectures, the authors in [PCZ$^+$] have recently pointed out that overfitting is the most critical issue when training their sequence-to-sequence model on popular benchmarks. By proposing a data augmentation method together with a long training schedule to reduce overfitting, they have achieved a large gain in performance superior to many modifications in network architecture.

In this chapter, we show that our on-the-fly data augmentation methods could help two latest architectures of sequence-to-sequence models to achieve state-of-the-art

performance on telephone conversation benchmark. We further revealed the difference in behavior between HMM/ANN and end-to-end models on learning from a speech dataset of multiple domains. This result leads to a direction for finding and providing useful data for the improvement of HMM/ANN and end-to-end models.

7.1 On-the-fly Data Augmentation

Overfitting was found to be the most critical issue when training sequence-to-sequence models. We investigated three data augmentation methods for improving the performance of sequence-to-sequence encoder-decoder models. The first two modify the input sequences from different inspirations and aim to improve the generalization of the log-mel spectrogram encoder. The third approach improves the decoder by adding sub-samples of target sequences. All of the proposed methods are computationally cheap and can be performed on-the-fly and can be optimized together with the model.

7.1.1 Dynamic Time Stretching

Many successful S2S models adopt log-mel frequency features as input. In the frequency domain, one major difficulty for the recognition models is to recognize temporal patterns which occur with varying duration. To make the models more robust to temporal variations, the addition of audio data with speed perturbation in the time domain such as in [KPPK15] has been shown to be effective. In contrast, in our work we manipulate directly the time series of the frequency vectors which are the features of our S2S models, in order to achieve the effect of speed perturbation. Specifically, given a sequence of consecutive feature vectors seq, we stretch every window of w feature vectors by a factor of s obtained from an uniform distribution of range $[low, high]$, resulting in a new window of size $w * s$. There are different approaches to perform window stretching, in this work we adopt *nearest-neighbor interpolation* for its speed, as it is fast enough to augment many speech utterances on a CPU while model training for other utterances is being performed on a GPU. The dynamic time stretching algorithm is implemented by the following python code:

```
def time_stretch(seq,w,low=0.8,high=1.25):
        ids = None; time_len = len(seq)
        for i in range(time_len // w + 1):
                s = random.uniform(low, high)
                e = min(time_len, w*(i+1))
                r = numpy.arange(w*i, e-1, s)
                r = numpy.round(r).astype(int)
        ids = numpy.concatenate((ids, r))
return seq[ids]
```

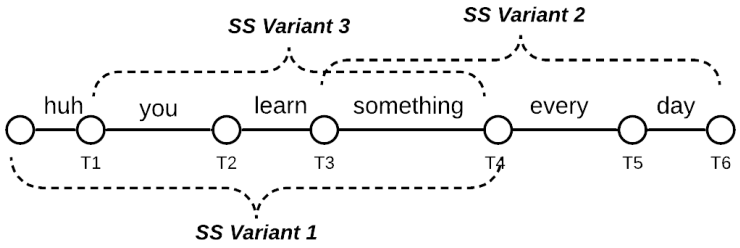
7.1.2 SpecAugment

Recently [PCZ+] found that LSTM-based S2S models tend to overfit easily to the training data, even when regularization methods such as Dropout [SHK+14] are applied. Inspired by the data augmentation from computer vision, [PCZ+] proposed to deform the spectrogram input with three cheap operations such as time warping, frequency and time masking before feeding it to their sequence-to-sequence models. Time warping shifts a random point in the spectrogram input with a random distance, while frequency and time masking apply zero masks to some consecutive lines in both the frequency and the time dimensions. In this work, we study the two most effective operations which are the frequency and time masking. Experimenting on the same dataset, we benefit from optimized configurations in [PCZ+]. Specifically, we consider $T \in [1, 2, 3]$ – the number of times that both, frequency and time masking, are applied. For each time, f consecutive frequency channels and t consecutive time steps are masked where f and t are randomly chosen from $[0, 70]$ and $[0, 7]$. When $T = 2$, we obtain a similar setting for 40 log-mel features as the SWB mild (SM) configuration in [PCZ+]. We experimentally find T for different model architectures in our experiments.

7.1.3 Sub-sequence Sampling

Different from other S2S problems, the input-output of speech recognition models are the sequences of speech feature vectors and label transcripts which are monotonically aligned. The alignment can be also estimated automatically via the traditional

Figure 7.1: Sub-sequence Sampling.



force-alignment process. Taking advantage of this property, we experiment with the ability to sub-sample training utterances to have more variants of target sequences. Since the approach of generating sub-sequences with arbitrary lengths does not work, we propose a constraint sampling depicted in Figure 7.1. Basically, given an utterance, we allow three different variants of sub-sequences with equal distributions. The first and second variants constraint sub-sequences to having either the same start or end as the original sequence while the third variant needs to have their start and end point within the utterance. All sub-sequences need to have at least half the size of the original sequence. During training, we randomly select a training sample with probability $alpha$ and replace it with one of the sampled sub-sequence variants. We also allow $static$ mode in which only one fixed instance of sub-sequence per utterance per variant is generated. This mode is equivalent to statically adding three sets of sub-sequences to the original training set.

7.1.4 Models

To date, there have been different sequence-to-sequence encoder-decoder models [PCZ$^+$, PNN$^+$] reporting superior performance over the HMM hybrid models on standard ASR benchmarks. While [PCZ$^+$] uses Long Short-Term Memory (LSTM) networks, for both encoder and decoder, [PNN$^+$] employs self-attention layers to construct the whole S2S network. We use two different S2S models to investigate the on-the-fly data augmentation methods. In the first model, we use LSTMs and a new approach for building the decoder network. For the second model, we follow the work in [PNN$^+$] to replace LSTMs with deep self-attention layers in both the encoder and decoder.

LSTM-based S2S Before the LSTM layers in the encoder, we place a two-layer Convolutional Neural Network (CNN) with 32 channels and a time stride of two to down-sample the input spectrogram by a factor of four. In the decoder, we adopt two layers of unidirectional LSTMs as language modeling for the sequence of sub-word units and the approach of Scaled Dot-Product (SDP) Attention [VSP+17] to generate context vectors from the hidden states of the two LSTM networks. Specifically, our implementation for LSTM-based S2S works as follows:

$$
\begin{aligned}
enc &= LSTM(CNN(spectrogram)) \\
emb &= Embedding(subwords) \\
dec &= LSTM(emb) \\
context &= SDPAttention(dec, enc, enc) \\
y &= Distribution(context + dec)
\end{aligned}
$$

Different from previous works [CJLV, CSW+, ZISN, WCW+], we adopt a simpler recurrent function in the decoder (i.e. without Input-feeding [WCW+]), and a more complicated attention module. The adopted attention function learns an additional linear transformation for each input parameter (known as query, key and value) and use the multi-head mechanism together with Dropout and LayerNorm for efficiently learning content-based attention [VSP+17]. In fact, the implementation of the attention function is shared with the deep self-attention network from Section **??**. In addition to that, we share the parameters between $Embedding$ and $Distribution$ to improve the word embedding. Because this implementation does not require us to customize LSTM cells (which is needed by Input-feeding), we can achieve high parallelization [1] to speed up training.

Self-Attention S2S We follow [PNN+] to build an encoder-decoder model with deep self-attention layers. Specifically, we use many stochastic self-attention layers (e.g., 36 and 12) for the encoder and the decoder for better generalization of the deep architecture. Instead of using a CNN for down-sampling the input spectrogram, we stack four consecutive feature vectors after applying the augmentation methods.

[1] Highly optimized LSTM implementation offered by cuDNN library

Compared to [PNN$^+$], we use BPE sub-word units instead of characters for target sequences. For more details refer to [PNN$^+$].

7.1.5 Experimental Setup

Our experiments were conducted on the Switchboard (300 hours) and the Fisher+Switchboard (2000h) corpora. The Hub5'00 evaluation data was used as the test set. For input features, we use 40 dimensional log-mel filterbanks normalized per conversation. For labels, SentencePiece was used for generating 4,000 BPE sub-word units from all the transcripts. We use Adam [KB15] with an adaptive learning rate schedule defined by (*lr, warm-up, decay*) in which the learning rate *lr* increases for the first *warm-up* steps and then decreases linearly. We adopted the approach in [PNN$^+$] for the exact calculation of the learning rate at every step. In addition to that, we further decay the learning rate exponentially with a factor of 0.8 after every *decay* step. We save the model parameters of 5 best epochs according to the cross-validation sets and average them at the end.

7.1.6 Baseline Results

Table 7.1: Baseline models using Switchboard 300h.

Model	Size	SWB	CH	Hub5'00
LSTM	4x512	12.9	24.1	18.5
	6x1024	12.1	22.7	17.4
	6x1024 (*SP*)	10.7	20.5	15.6
Transformer	8x4	13.2	24.7	19.0
	36x12	11.1	21.1	16.1
	36x12 (*SP*)	10.2	19.4	14.8

Using the SWB material and an unique label set of 4k sub-words, we trained both of the proposed S2S models for 50 epochs. We adopt a mini-batch size of 8,000 label tokens which contains about 350 utterances. In our experiments, the LSTM-based models tend to overfit after 12k updates (i.e. perplexity increases on the cross-validation set) while the self-attention models converge slower and saturate at 40k updates. We were able to increase the size of the LSTM-based as well as the depth

of the self-attention models for performance improvement. We stop at six layers of 1,024 units for the encoder of the LSTM-based and 36-12 encoder-decoder layers of self-attention models, and then use them as baselines for further experiments. Table 7.1 shows the WER of the baselines. We also include the results of the baseline models when trained on the speed-perturbed dataset [KPPK15].

7.1.7 Results with Time Stretching and SpecAugment

Table 7.2: The performance of the models trained with *TimeStretch* and *SpecAugment* augmentation.

TimeStretch w	**SpecAugment** T	**LSTM** Hub5'00	**Transformer** Hub5'00
50	-	16.1	15.5
100	-	15.9	14.9
200	-	16.0	14.9
∞	-	16.1	15.0
-	1	14.7	14.3
-	2	14.1	14.5
-	3	14.3	14.4
100	1	14.2	13.8
∞	1	13.9	13.6
100	2	13.7	13.9
∞	2	13.6	13.7

Both *Time Stretching* and *SpecAugment* are augmentation methods which modify the input sequences aiming to improve the generalization of the encoder network. We trained several models for evaluating the effects of these methods individually as well as the combinations as shown in Table 7.2.

For *Time Stretching*, WER slightly changed when using different window sizes. However the 8.6% and 12.4% rel. improvement over the baseline performance of the LSTM-based and self-attention models clearly shows its effectiveness. With a window size of 100ms, the models can nearly achieve the performance of the static speed perturbation augmentation.

As shown in [PCZ+], *SpecAugment* is a very effective method for avoiding overfitting on the LAS model. Using this method, we can also achieve a large WER

improvement for our LSTM-based models. However, our observation is slightly different from [PCZ+], as *SpecAugment* slows down the convergence of the training on the training set and significantly reduces the loss on the validation set (as for *Time Stretching*) but does not change from overfitting to underfitting. The losses of the final model and the baseline model computed on the original training set are similar.

SpecAugment is also effective for our self-attention models. However, the improvements are not as large as for the LSTM-based models. This might be due to the self-attention models not suffering from the overfitting problem as much as the LSTM-based models. It is worth noting that for the self-attention models, we use not only Dropout but also *Stochastic Layer* [PNN+] to prevent overfitting. When tuning T for both models, we observed different behaviours. The LSTM-based models work best when $T = 2$, but for self-attention, different values of T produce quite similar results. This might be due to the fact that the self-attention encoder has direct connections to all input elements of different time steps while the LSTM encoder uses recurrent connections.

When combining two augmentation methods within a single training (i.e. applying *Time Stretching* first and then *SpecAugment* for input sequences), we can achieve further improvements for both models. This result indicates that both methods help the models to generalize across different aspects and can supplement each other. We keep using the optimized settings ($T = 2$ and $w = \infty$ for LSTM-based and $T = 1$ for self-attention) for the rest of the experiments.

7.1.8 Results with Sub-sequence Sampling

Table 7.3: The performance of the models trained with *Sub-sequence* augmentation.

Sub-sequence *alpha*	**SpecAugment** & **TimeStretch**	**LSTM Hub5'00**	**Transformer Hub5'00**
0.3	N	18.6	15.6
0.5	N	18.6	15.4
0.7	N	18.8	15.3
0.7 (static)	N	15.4	15.1
0.7	Y	13.5	13.4
0.7 (static)	Y	13.0	13.2

Table 7.3 presents the models' performance when we applied *Sub-sequence* augmentation with different $alpha$ values. We observe contrary results for different models: improving the self-attention but downgrading the performance of the LSTM-based models. These observations are indeed consistent with the overfitting problems observed with the two models. The LSTM-based models even overfit more quickly to the dataset with sub-sequence samples while self-attention models do not, so that they can benefit from *Sub-sequence*. However, when using a static set of sub-sequences, we obtained clear improvement for LSTM-based models but had comparable performance for self-attention models. This reveals an interesting observation for the differences between self-attention and LSTM when interpreting them as language models in the decoder. The static approach is also better when combined with other augmentation methods.

7.1.9 Results on Larger Training Set

We report the final performance of our models trained on the 2,000h in Table 7.4. Slightly different from 300h, we used a larger mini-batch size of 12k tokens and do not use the exponential decay of the learning rate. We also increased the model size by a factor of 1.5 while keeping the same depth. We need 7 hours to finish one epoch for the LSTM-based models, 3 hours for the self-attention models. With the bigger training set, the LSTM-based models saturate after 100k updates while the self-attention models need 250k updates. Even with the large increase in training samples, the proposed augmentation is still effective since we observe clear gaps between the models with and without augmentation. For the final performance, we found that the ensemble of the LSTM-based and self-attention models are very efficient for the reduction of WER. Our best performance on this benchmark is competitive compared to the best performance reported in the literature so far, and it is notable that we did not employ any additional text data, e.g., for language modeling.

Table 7.4: Final performance on Switchboard 300h and Fisher 2000h training sets.

Model	LM	SWB	CH
300h Switchboard			
Zeyer et al. 2018 [ZISN]	LSTM	8.3	17.3
Yu et al. 2018 [ZISN]	LSTM	11.4	20.8
Pham et al. 2019 [PNN+]	-	9.9	17.7
Park et al. 2019 [PCZ+]	LSTM	7.1	14.0
Kurata et al. 2019 [KA]	-	11.7	20.2
LSTM-based	-	8.8	17.2
Transformer	-	9.0	17.5
ensemble	-	7.5	15.3
2000h Switchboard+Fisher			
Povey et al. 2016 [PPG+]	n-gram	8.5	15.3
Saon et al. 2017 [SKS+]	LSTM	5.5	10.3
Han et al. 2018 [HCKL17]	LSTM	5.0	9.1
Weng et al. 2018 [WCW+]	-	8.3	15.5
Audhkhasi et al. 2018 [AKR+]	-	8.8	13.9
LSTM-based (*no augment.*)	-	7.2	13.9
Transformer (*no augment.*)	-	7.3	13.5
LSTM-based	-	5.5	11.4
Transformer	-	6.2	11.9
ensemble	-	5.2	10.2

7.2 Augmentation with Multi-domain Data

The studies on multi-domain and domain-invariant speech recognition can be roughly divided into two general approaches. The first approach focuses on exploiting additional speech data to train acoustic models which then become invariant to specific acoustic conditions, resulting in domain-invariant speech recognition systems. E.g., [YSL+13, KMC+17] used simulated noisy utterances together with clean training data to achieve invariance to background noise. Similar to that, the authors in [YSL+13] used a mixed bandwidth training dataset to help the acoustic model generalize to multiple sampling rates. In [PMW+16], far-field speech recognition is significantly improved by exploiting large-scale simulated data for training deep neural networks. [NMS+18] build a multi-domain speech recognition system by pooling a huge amount of training data from several sources and simulated conditions like background noise,

codecs and sample rates.

Previous research have focused on hybrid acoustic models to build cross-domain and domain-invariant speech recognition systems. We empirically examine the difference in behavior between hybrid acoustic models and neural end-to-end systems when mixing acoustic training data from several domains. For these experiments we composed a multi-domain dataset from public sources, with the different domains in the corpus covering a wide variety of topics and acoustic conditions such as telephone conversations, lectures, read speech and broadcast news. We show that for the hybrid models, supplying additional training data from other domains with mismatched acoustic conditions does not increase the performance on specific domains. However, our end-to-end models optimized with sequence-based criterion generalize better than the hybrid models on diverse domains.

7.2.1 Multi-domain Dataset

Several public speech corpora have been released for speech recognition research. However, to the best of our knowledge, there has not been such a common corpus for the study of multi-domain speech recognition. We composed a multi-domain training dataset which consists of four well-known corpora: Switchboard [GHM92], TED-LIUM [HNG+18], Libri Speech [PCPK15] and Hub4 (LDC97S44 & LDC98S71). The statistics of the multi-domain set are described in Table 7.5. Basically, it includes 1,282 hours of speech data coming from different domains such as *telephone conversations*, *talks and lectures*, *read speech* and *broadcast news*. Compared to an internal multi-domain set reported in [NMS+18], our composed multi-domain set is fairly distributed in which there is no disproportionally large sub-set.

To evaluate the performance of our models, we use the Hub5'00 test set (LDC2002S09), the TED-LIUM test set, the Libri test-clean set and the Hub4 eval set (LDC97S66) as the evaluation sets for the corresponding domains.

7.2.2 Hybrid Models

From a bootstrap system built on a part of *Multi-Set*, we used a common cluster-tree of 8,000 context-dependent phonemes and the same forced-alignment system to provide frame-based labels for all domain-specific and multi-domain training sets. We then

Table 7.5: The training and test datasets for cross-domain speech recognition.

Dataset	Domain	Hours	#Utt
Switchboard	Telephone Conversation	318	263K
TED-LIUM	Lecture Presentation	453	268K
Libri	Read Speech	363	104K
Hub4	Broadcast News	148	125K
Multi-Set	Multiple	1282	760K
Hub5-2000 (Hub5'00)	Telephone Conversation	3.79	4458
TED-LIUM test (TED)	Lecture Presentation	2.61	1155
Libri test (Libri)	Read Speech	5.4	2620
Hub4 eval (Eval98)	Broadcast News	2.81	825

trained hybrid acoustic models using both a feed-forward neural network (FFNN) and a long short-term memory (LSTM) network. The FFNN models consist of 7 layers of 2,000 units while bidirectional LSTM models have 5 layers with 320 units each. 40 log mel filter-bank features which are mean and variance normalized per utterance are used for all models. For FFNNs, we used a window of 15 consecutive frames as the input, while for LSTMs, we generated sub-sequences of 50 frames with a moving step of 25 frames from the training utterances.

The hybrid acoustic models were decoded with domain-adapted language models (LM) for individual test sets. Specifically, the LM for *Hub5'00* was built from the transcripts of the Switchboard and Fisher corpora, while the standard LM for *Libri* is described in [PCPK15]. We used the same Cantab LM [WPM+15] for *TED* and *Hub4*. To investigate the influence of the domain-adapted language models on recognition performance, we used an additional LM which was built from the transcripts of the *Multi-Set* set.

7.2.3 Results with Hybrid Models

In the first 4 rows of Table 7.6, we present the WER performance of the hybrid acoustic models trained on individual domain-specific training sets and evaluated with all the test sets. As can be observed, the domain-specific models only perform well on the test sets that match the training domain conditions, and can be very poor on out-of-domain test sets. On *Eval98* the recognizer trained on an in-domain dataset with much smaller

size still outperforms other training sets. At the worst case of mismatching, the recognition performance hugely drops shown on *TED*. These observations substantiate the importance of in-domain data in building hybrid speech recognition. The cross comparisons also reveal the similarities between the individual training sets. For example, *Switchboard* and *Libri* are very different from the others while *TED-LIUM* and *Hub4* are closer corpora. These results are consistent with the analysis in Section **??**.

We evaluated the multi-domain model with all the test sets as in the last three rows of Table 7.6. The WER performance of the multi-domain model shows two interesting facts for the combination of speech corpora of different domains. First, when two training corpora are close enough (e.g. *TED-LIUM* and *Hub4*), they can supplement each other so that the hybrid acoustic model can benefit from the mixed data training. Second, for the case of *Switchboard* and *Libri*, the recognition performance is hardly improved when the additional training datasets are diverse.

These results also reveal the abilities as well as the limitations of the hybrid speech recognition approach. On one hand, the hybrid models are capable of modeling short windows of frames from a mixed domain dataset and actually produce no performance loss in comparison to specific domain modeling. However, on the other hand, when acoustic conditions are too diverse, the hybrid models cannot generalize well which show their limitations in exploiting multi-domain speech data.

The other limitation of a conventional hybrid model is that it always requires a domain-adapted language model for inference. We investigated the influence of domain-adapted language models by decoding the multi-domain model with two different LMs. As can be seen, the performance of the multi-domain model clearly degrades when switching to the *Multi-Set* LM which partly includes in-domain data, and largely drops for the Cantab LM which does not match the domains of the test sets.

7.2.4 End-to-end Models

The acoustic-to-word (A2W) model based on the Connectionist temporal classification (CTC) [GFGS06] criterion was first introduced in [SSRB15] as a natural end-to-end model directly targeting words as output. In [SLS16], the authors have successfully built a direct A2W system that achieves state-of-the-art speech recognition

Table 7.6: The WER performance of hybrid systems with FFNN and LSTM (in brackets) acoustic models. The columns of the table indicate the different test sets while the rows show the used training sets.

	Hub5'00	**TED**	**Libri**	**Eval98**
Switchboard	**23.3 (18.3)**	65.7	22.5	63.1
TED-LIUM	54.3	**12.0 (10.5)**	8.1	17.6
Libri	61.6	18.8	**6.5 (5.9)**	22.0
Hub4	37.1	15.0	9.7	**14.5 (12.8)**
Multi-Set	**23.2 (18.3)**	**11.1 (9.6)**	**6.4 (5.8)**	**13.6 (11.6)**
+Multi-Set LM	24.2 (19.2)	12.3 (11.1)	10.1 (8.4)	14.0 (11.8)
+Cantab LM	27.5 (22.5)	-	10.7 (8.6)	-

performance by leveraging 125,000 hours of training data collected from Youtube videos. Later on, [ARS$^+$17, AKR$^+$] proposed training optimization to train A2W models on the standard Switchboard 300 hours training set which results in competitive performance with other end-to-end approaches.

One of the major difficulties of training A2W system is the data sparsity problem. While [SLS16] has alleviated this problem by using exceptionally large training data, [ARS$^+$17, AKR$^+$] have used pre-trained CTC-phone models and and used GloVe word embeddings [PSM14] to initialize acoustic-to-word models on a moderately sized training data. We use the multi-task training approach proposed in [TSN19] to directly train A2W models on the domain-specific and cross-domain data sets.

Specifically, to build A2W models we use 5 LSTM layers of 320 units. We also keep the same feature extraction as for the hybrid acoustic models. For each training set, we find words appearing more than 5 times in the transcripts to build target units for the corresponding A2W model. The second task of the multi-task network is always the framewise classification of 8000 context-dependent phonemes. We adopted a down-sampling on acoustic features performed by stacking two consecutive frames followed by the drop of one frame. Stochastic gradient descent (SGD) with New-bob training schedule are used for model optimization. Initial learning rates are set as high as possible for individual training, and then is decayed by a factor of 0.8 after 12 epochs.

Sequence-to-sequence attention-based speech recognition models [CBS$^+$15, BCS$^+$16, CJLV] use a single neural network that consists of an encoder recurrent neural network (RNN) and a decoder RNN, and uses an attention

Table 7.7: The performance of acoustic-to-word and sequence-to-sequence models trained on domain-specific and multi-domain data sets.

	Hub5'00	**TED**	**Libri**	**Eval98**
Char Seq2Seq Model				
Switchboard	**22.9**	45.4	35.5	*60.2
TED-LIUM	60.1	**13.0**	17.0	*29.0
Libri	72.7	34.3	**10.3**	*52.1
Hub4	42.2	25.7	23.8	***25.5**
Multi-Set	**18.2**	**10.6**	**7.6**	***20.8**
Word Seq2Seq Model				
Domain-Specific	22.4	12.8	11.2	23.9
Multi-Set	**18.5**	**10.6**	**8.5**	**11.9**
Acoustic-to-word Model				
Domain-Specific	23.8	14.2	12.2	19.4
Multi-Set	**19.4**	**11.3**	**8.9**	**11.9**

mechanism to connect between them. The decoder is analogous to a language model due to attention-based model being trained to provide a probability distribution over sequences of labels (words or characters). The encoder converting low level acoustic features into higher level representation is analogous to the RNN of an CTC model.

We follow the approach in [TSN19] in which we took the pre-trained LSTM layers from the A2W network (trained with the same data set) to initialize the encoder of attention-based models. For the decoder, we used only one uni-directional LSTM layer. Adam [KB15] and New-bob schedules are used to optimize the attention-based models. We experimented with sequence-to-sequence models using characters and words as target units. For both label units, we use a beam search with the beam size of 12 for decoding.

7.2.5 Results with End-to-end Models

As done for the hybrid acoustic systems, we trained individual end-to-end models for different domain-specific and multi-domain training sets and evaluated them with the proposed test sets. For character-based models, we used an unified label set of 52 characters while the word-based sequence-to-sequence and acoustic-to-word models share the same vocabularies for individual training sets. During inference, we observed that the character-based sequence-to-sequence models have confusion when

decoding with very long utterances (e.g. 60-120 seconds) so that it performs worse for the *Eval98* test set. The word-based models do not have this issue, however it is theoretically encountered with out-of-vocabulary words.

As shown in Table 7.7, the end-to-end models trained on the domain-specific sets are also very poor at handling the domain mismatches between training and testing conditions. However, when switching to the multi-domain dataset, all of the end-to-end models behave differently from the hybrid models. As can be seen, the performance of the multi-domain models outperform all the domain-specific models with clear margins. The improvements on the *Hub5'00* and *Libri* test sets clearly indicate that the end-to-end models can exploit the additional training data which come from different domains. This observation also reveals the advantage of the end-to-end approaches over the hybrid approach in multi-domain speech recognition.

In our multi-domain setup, the performance of the multi-domain end-to-end systems are still lacking behind the multi-domain hybrid systems using domain-adapted LMs, but it already surpasses the hybrid systems when the LMs do not match the target test sets, and is at par with the hybrid domain-specific systems.

Chapter 8

Measure of Latency

Although latency in general refers to the response time of the recognition systems, it has been viewed in different ways in the past. For instance, in dialogue systems such as Google Voice [SBB+10], latency is defined as the time from when the user finishes speaking until the search results appear. In other related work on speech recognition for broadcast news [SRBG02], latency measurement has included the time for the input to be completed. In this chapter, we review the methods for measuring latency used in the literature. We further analyze the shortcomings of these methods when being for streaming speech recognition and propose novel and useful approaches to replace them.

8.1 Common Methods

Real-time factor (**RTF**), is calculated as the ratio between the utterance duration and its required decoding time. RTF is a common measure to evaluate the speed of a speech recognition system. Although distinct from the concept of latency, reducing the RTF can lead to a reduced latency in recognition systems, especially when the decoding starts after the input is completed.

Commitment latency is the difference between the end time of an audio segment or portion and when its transcription is available at the display component. This is equivalent to the latency measure used in [SRBG02].

8.2 User-perceived Latency

Latency is one of the most important factors that decide the usability of a user-based online ASR system. Latency measure needs to reflect the actual delay that users perceive so that the improvement of latency can lead to better usability. Strictly, the latency that the users observe for a word is the time difference between when the word was uttered and when its transcript appears to the users.

Neither the commonly used real-time factor nor commitment latency are sufficient to measure user-perceived latency for a streaming recognizer. For example, the transcript outputs for an 11-seconds sentence can appear 10 seconds later than a 1-second sentence, but the RTFs measured in two cases can be similar.

8.2.1 Decomposition

We formulate the user-perceived latency as follows. Assume that a recognizer can *confidently* infers the word w at the time C_w while D_w is the *delay* required for the inference process. If U_w is the uttered time of w, then the user-perceived with regard to w is calculated as:

$$Latency_w = C_w + D_w + T_w - U_w$$

where T_w presents the transmitting time for audio and text data. T_w is usually small and can be taken out.

For a speech utterance S consisting of N words w_1, w_2,.. w_n, we are interested in an average latency:

$$\begin{aligned} Latency_S &= \sum_i^N (D_{w_i} + C_{w_i} - U_{w_i})/N \\ &= \sum_i^N D_{w_i}/N + \sum_i^N C_{w_i}/N - \sum_i^N U_{w_i}/N \\ &= \sum_i^N D_{w_i}/N + \sum_i^N C_{w_i}/N - \sum_i^N (U_{w_i} - \Delta)/N + \Delta \\ &= D_{avg} + C_{avg} - U_{avg-\Delta} + \Delta \end{aligned}$$

In the final equation, the first term presents the computational delay. If we normalize this term with length of the utterance, then we have real-time factor (RTF). The second term indicates how much acoustic evidence the model needs to confidently decide its output. This latency term represents the difference between offline and online processing. In offline, it is always a constant for a specific test set, since all the offline transcripts are outputted at the end of utterances.

To estimate the third term, we usually need to use an external time alignment system. It is inconvenient to re-run the time alignment for every new transcripts. To cope with this issue, we use a fixed delay Δ for all the outputs, and proposed to compute in advance a set of $U_{avg-\Delta}$ with different values of Δ as in Section 8.2.2. Later on, we only need to compute C_{avg} and compare it with the pre-computed set to find the corresponding delay Δ.

The latency improvement requires to optimize both D_{avg} and C_{avg} which we refer as *computation latency* and *confidence latency*. While *computation latency* can be improved by faster hardware or more optimized implementation, *confidence latency* depends on the recognition model. Thus, we usually need different strategies for the improvement of them.

8.2.2 Confidence Latency

Assume that a recognizer processes a sentence S of T seconds in streaming fashion and it outputs N token s_1, s_2,.. s_n at different timestamps t_1, t_2,.. t_n. And assume the timestamp t_i is when the recognizer is confident of producing s_i. The Confidence latency of recognizing S with regard to the transcript s_1, s_2,.. s_n, is calculated as the average of all token timestamps $t_i \in [1, n]$ normalized by the duration of S: $\sum t_i/(n*T)$. For short, the proposed method produces a normalized and absolute latency for a sentence. With this measure, the confidence latency of an offline system is always 1 – as the offline system is only confident for all transcripts until end-of-sentence. In the same way, we simulate the latency of an *instant* recognizer by using a forced-alignment to find t_i for s_i.

Chapter 9

Online Capacity and Streaming Inference

Online speech recognition differs from offline recognition in that latency is a crucial issue. In many online scenarios, the speech recognition component needs to output transcript fast enough, e.g., within a delay constraint unless it becomes applicable. Building an online ASR system which satisfies a latency constraint and produces comparative accuracy with its offline counterpart, is not a straightforward task. The real challenge usually does not lie on the optimizations for computing power but the latency issues introduced by the sub-modules of the system. For example, the feature normalization technique, e.g., CMVN, which is commonly operated at the utterance level or more ideally at the speaker level, is not appropriate for low-latency conditions. Or the use of the attention function and bidirectional encoders in the sequence-to-sequence model, which require the entire input sequence, is an obstacle to build an end-to-end online recognizer.

In this chapter, we present the techniques which enable the batch-mode models proposed for both hybrid HMM/ANN and end-to-end paradigms to be used for online recognizers while retaining their high performance. In Section 9.1, we analyze the latency constraint needed for a real-world simultaneous translation system and then present a set of adaptation to achieve this requirement. In Section 9.2, we discuss the latency issues introduced by the sub-components of an end-to-end sequence-to-sequence model and propose effective techniques to overcome.

9.1 Streaming HMM/ANN System

Since 2012, a system for the simultaneous translation of lectures [FWK07, FÖ8] has been operating in several of KIT's lecture halls on a regular basis. In order for students to be able to follow a lecture by using the system's automatic translation and transcription, the system's output needs to be as much in sync with the lecturer and his presentation as possible. Thus, the speech and translation components of the systems do not only need to run in real-time, but must produce output with as low a latency as possible. The high importance of a low latency is also the result of a user study and test conducted with the lecture translation system during real-world operation [MFSW16].

In this section, we present two approaches to address the problem of reducing the latency of the HMM/ANN speech transcription component while maintaining its accuracy. The first approach uses a real-time recognition system, utilizing an incremental decoding framework to decode continuous audio streams, in combination with a traceback of stable partial hypotheses. This approach is used to output whole portions, i.e. several words in sequence, of the final hypothesis as soon as possible. The second approach enhances the first one in combination with the display components by allowing to output partial hypotheses not only when they are stable, i.e., when it can be guaranteed that they will not change anymore in the future, but at any time as soon as they are available. In unison with the display and translation components the recognition system is then allowed to correct itself later on, i.e. to revise the most recent history of its output, when a different word sequence has become more probable. Since very often the system will not need to correct itself, but the early output turns out to be the stable one, even though this could not have been guaranteed at the time it was passed on to the translation and display components, the latency of the system is reduced further this way. Both approaches are shown to reduce the latency of the speech transcription component significantly from 18.1s to 1.1s without any loss in recognition performance.

9.1.1 Run-on Recognition

Run-on recognition overlaps the decoding process with the audio recording process in order to reduce the latency. Our system used an adapted version of the run-on decoding described in [FÖ8].

As in [FÖ8] our system uses an audio segmenter in a separate process for pre-processing which writes the incoming audio stream into a shared memory with the decoder, while at the same time filtering out stretches of silence. This reduces the system load by stopping the recogniser from decoding long stretches of silence. We refer to everything between two stretches of silence as a *segment*, which are usually several minutes long but could be as long as a lecture.

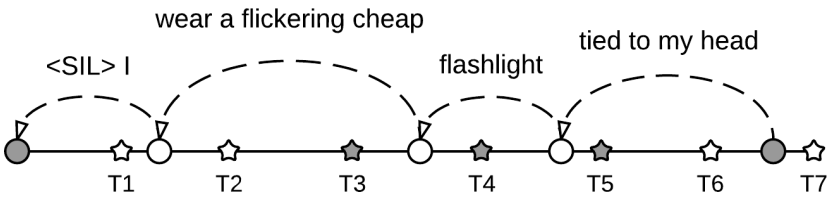
Our recognition system's search is re-initialised before processing a new segment and reads the segment's audio data from the shared memory while the audio segmenter continues to write to it. The audio data read in *chunks* consisting of a fixed number of frames and incrementally decoded. The system therefore only has to wait until a chunk of audio is available in contrast to batch processing which requires complete segments before decoding.

9.1.2 Stable Hypothesis Portion

The decoder tries the to find the most probable hypothesis. At any given frame there are usually many competing hypotheses and only once the final frame has been processed can the best one be decided upon. However, because waiting for an end-of-segment detected by the segmenter leads to a high latency, we use a partial trace-back [FÖ8] for finding stable portions of the hypothesis early. As a partial hypothesis we define the part of the current hypothesis, i.e. the most probable word sequence when not all audio data has been processed yet, that will not change anymore even when more audio data becomes available. We know that a part of a hypothesis will not change anymore when during Viterbi decoding all paths that do not contain this word sequence as a prefix have been pruned away. In our design, we detect partial hypotheses right after a chunk has been processed. Whenever a partial hypothesis is detected, its stable portion is extracted and delivered. The end of the portion will be tracked for the next detection. Note that the final hypothesis concatenated from all stable portions is the most probable hypothesis as obtained when batch decoding the whole segment.

Figure **??** presents the process of finding stable portions for a segment in TED talk 1541. The dark circles indicate the start time and end time of the segment while at the stars, the decoding was performed incrementally. The dark stars indicate when the

Figure 9.1: Finding stable portions.



system found a partial hypothesis and the borders of the stable portions are marked by white circles. The latency is improved by delivering the portions early (at T3, T4 and T5) rather than delivering all of them at the end of the segment.

9.1.3 Adaptive Pruning

Although our recognition system runs, on average, significantly faster than real time, we frequently encounter individual chunks which are processed much slower than real-time. This happens when encountering chunks that are difficult to decode, e.g. speech with background music, in which case the beam might fail to prune away competing paths effectively. This problem results in an unstable response time and introduces latency peaks. To overcome the problem, we use an adaptive pruning scheme. When an audio chunk is processed slower than real-time, we will narrow the beam to reduce the processing time of the following chunks. Once the recognition system has caught up again with the live audio the beam size is set back to its normal size.

9.1.4 Hypothesis Update

Next, we introduced another method that dramatically reduces the latency. We output probable parts of the unstable hypothesis and present them to the user. Later, the recognizer can revise its decision and overwrite the previous output if necessary. In this way, the recognition component does not need to wait until a stable portion or end-of-segment, instead it finds the most probable hypothesis every iteration of the incremental decoding, detects and sends the update portions to the display component.

Figure 9.2 illustrates how this works in detail. In the example, the incremental decoding was performed 7 times and each time the most probable hypothesis was

Figure 9.2: An example of hypothesis update.

generated. The updated parts (italic text) were detected each time and sent to the display component. At T4, T6 and T7 the system detected the stable portions (underlined text). These had however already appeared as part of the unstable hypothesis at much earlier times. At T5 and T7, the hypotheses had new start times as described in Section 9.1.2.

Ignoring the words that are later replaced this algorithm can be seen as inducing a partition of stable hypothesis resulting in an improved latency without any accuracy loss. For example, only at T6 the system was sure about the stable hypothesis portion "wear a flickering cheap", but the parts of it were already sent, "wear" at T1, "a" at T2, "flickering" at T3 and "cheap" at T4. So the latency is again improved.

9.1.5 Experiments

We evaluate two baseline systems and three variants using the techniques described above for reducing latency. The first baseline which demonstrates batch processing, waits for completed segments before performing the whole decoding. The segments are generated by our integrated energy based segmenter. In the second baseline, we replace the batch processing with the run-on decoding described in Section **??**. The decoded results are still produced for whole segments. Run-on decoding is employed in all three of the examined experimental systems.

The first advanced system, labelled *Portion*, uses the algorithm from Section 9.1.2 for finding stable hypothesis portions. The second experimental system, called *Update*, applies the update protocol explained in Section 9.1.4. Both of these utilise adaptive pruning which could result in a loss of accuracy. The third variant, named *Update-NA*,

Table 9.1: System Summary (AP = Adaptive Pruning, PH = Partial Hypothesis).

System	Run-on	AP	PH	Update
Baseline-1	N	N	N	N
Baseline-2	Y	Y	N	N
Portion	Y	Y	Y	N
Update	Y	Y	Y	Y
Update-NA	Y	N	Y	Y

applies the update protocol but without adaptive pruning. A chunk size of 40 frames is used in all run-on systems. Table 9.1 shows the summary of the applied techniques.

All systems share the same basic setup. The setup is based on the off-line systems used in the IWSLT 2015 evaluation [MNS+15]. It uses a hybrid DNN/HMM acoustic model with log-Mel features. The acoustic model uses a context dependent phoneme setup with three states per polyphone. The DNN has an input window of +-7 frames, followed by 4 layers of 1,600 neurons and a classification output layer containing just over 8,000 neurons. The acoustic model was trained on the TEDLIUM [RDE14b] and Quaero data [SKK12b]. We used a 4-gram language model with more than 150 thousand words.

Evaluations were conducted using an Intel Xeon E5-2697 server with 512 GB memory. Experiments were performed using the Janus Recognition Toolkit (JRTk) [FGH+97] which was developed jointly by Karlsruhe Institute of Technology and Carnegie Mellon University.

First we evaluate the overall performance by averaging the measurements of WER, RTF, commitment latency and word latency over all talks in the test set. RTF is measured by the ratio between the processing time and the length of the processed audio segment. Commitment latency and word latency are measured as defined in Section **??**. Secondly, we measure the peaks in latency. We use both commitment latency and word latency for this analysis.

The test data used for the evaluation includes 8 TED talks from the development set of the IWSLT 2015 evaluation campaign. These talks are about different domains and presented in English by different speakers.

9.1.6 Accuracy and Latency

Table 9.2 shows the performance of all systems on all test talks in terms of overall WER as well as average RTF, commitment latency and word latency.

Table 9.2: Overall performance.

System	WER	RTF	Commit. Latency	Word Latency
Baseline-1	18.6	0.51	7.02	18.1
Baseline-2	18.4	0.68	0.92	10.2
Portion	18.5	0.68	1.72	2.10
Update	18.5	0.68	0.83	1.09
Update-NA	18.5	0.71	1.03	1.23

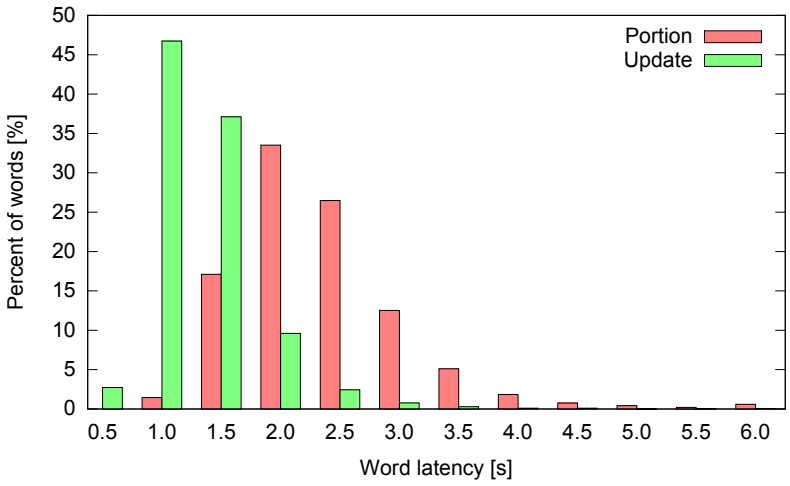
All the systems have similar WER performance. This confirms that our implemented algorithms did not change the accuracy. The batch processing *Baseline-1* achieves a lower RTF than the other systems that employ run-on processing. This is because it is less efficient for the DNN acoustic model to process multiple smaller chunks than a few large chunks.

Despite its low RTF *Baseline-1* has a large commitment latency since in the batch processing this latency mostly reflects the processing time of the segments. *Portion* has a larger commitment latency than *Baseline-2* and *Update* since it needs to wait until the output can be guaranteed to be stable. *Baseline-2* demonstrates that we can significantly reduce the commitment latency by following the run-on design. Note also that commitment latency, word latency, and RTF are only loosely correlated, indicating that commitment latency and RTF are not sufficient for evaluating the latency of the continuous recognition systems, and justifying our introduction of word latency.

Baseline-2 is especially interesting in this regard, because it has a low commitment latency but a very high word latency. This demonstrates the need for committing recognition results as quickly as possible in order to achieve a low latency. In this sense, *Portion* and *Update* perform better than the others.

As a more detailed analysis, we provide the statistics in Figure 9.3. It shows the latency distribution of all uttered words in the test set. We only focus on *Portion* and *Update*. According to the diagram, most spoken words are recognised within 2 seconds in *Update*, and 3.5 seconds in *Portion*.

Figure 9.3: Word latency distribution (the rightmost column also includes the words with latency larger than 6s).



9.2 Streaming End-to-End System

[RLL+17, CR17] pointed out early that the shortcoming of an attention-based S2S model used in online condition lies in its attention mechanism, which must perform a pass over the entire input sequence for every element of the output sequence. They proposed a so-called monotonic attention mechanism which enforces a monotonic alignment between the input and output sequence. Later on, [FZC+19a, MCZ+19, TKKW19] have addressed the latency issue of bidirectional encoders which is also an obstacle for online speech recognition. In these studies, unidirectional and chunk-based encoder architectures replace the fully-bidirectional approach to control the latency.

In this work, we analyze the alignment behavior of the attention function of a high-performance S2S model and propose an additional constraint loss to make it capable of streaming inference. By discussing the problems that occurred when adapting a S2S model to be used for a streaming recognizer, we additionally show that the standard beam-search has no guarantee for low-latency inference results, and needs to be modified for providing partial hypotheses.

In contrast to earlier research in the literature, our experimental results prove that a bidirectional encoder can be combined with suitable inference methods to produce high accuracy and low latency speech recognition output. With a delay of 1.5 seconds

in all output elements, our streaming recognizer can fully achieve the performance of an offline system of the same configuration. To the best of our knowledge for the first time, a S2S speech recognition model can be used in online conditions without scarifying accuracy.

9.2.1 LSTM-based Model

Our model can be decomposed using a set of neural network functions as follows:

$$\begin{aligned} enc &= LSTM(CNN(spectrogram)) \\ emb &= LSTM(Embedding(subwords)) \\ ctx, attn &= SoftAttention(emb, enc, enc) \\ y &= Distribution(ctx + emb) \end{aligned}$$

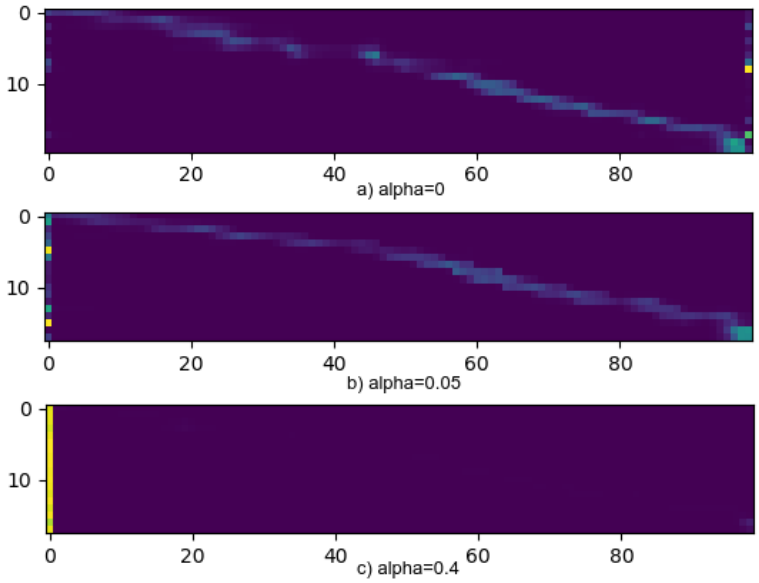
In principle, the functions are designed to map a sequence of acoustic vectors to a sequence of sub-words and can be grouped into two parts: encoder and decoder. In the encoder, acoustic vectors are down-sampled with two convolutional layers and then fed into several bidirectional LSTM layers to generate the encoder's hidden states enc. In the decoder, two unidirectional LSTM layers are used to embed a sub-word unit into a latent representation emb. The *soft-attention* function proposed in [VSP+17] is used to model the relationship between enc and emb, which results in a context vector ctx. All the functions are jointly trained via the sequence cross-entropy loss by plugging a softmax distribution on top of ctx and emb.

As shown in [NSNW19], this S2S model can achieve highly-competetitive offline performance on the Switchboard speech recognition task. However, the model encounters latency issues when being used in online conditions since both, the attention function and the bidirectional encoder network, require the entire input sequence to achieve their optimal performance.

9.2.2 Discouraging Look-ahead Attention

The core of S2S models is the mechanism that autoregessively generates a context vector *ctx* for the prediction of the next token. For this model as described in Section **??**, *ctx* is computed as a sum of all the encoder's hidden states weighted by the *attention*

Figure 9.4: Attention-based alignments provided by a) the regular attention function and b) c) the attention function trained with the constraint loss during the inference of an utterance of 4-seconds length (down-sampling of 4 frames after encoder's layers). The alignments for the tokens 3, 8, 9, 16 are dominated by both start and end frames in a), and dominated by start frames only in b).



scores which are calculated by the attention function. The attention scores calculated for a specific token typically reveals which of the positions of the encoder (or spectral frames) correspond to the token. So, the attention function can be considered as an *alignment model*. However, this unsupervised alignment does not pursue what traditional forced-alignments (or human alignments) do for the speech recognition task. As illustrated in Figure 9.4a, during the inference of an utterance, the attention scores produced for many tokens (e.g., #3, 8, 9, 16) are dominated by the start and end frames, which are not the proper alignments. In this case, the inference still produces the correct transcript, and so the attention function works as it is expected. The mismatch between the attention-based alignment and regular alignment reveals uncertainty that the attention function may have while being optimized with the sequence training likelihood. Although this uncertainty may not lead to inference errors, the attention function always employs all the encoder's hidden states, which hinders the model from being used in streaming inference. It is preferable for streaming inference that for the prediction of a token S, the attention function only

considers past frames until a particular time S_t (the endpoint) and disregards all future frames.

To build such a S2S model for streaming, we investigated the incorporation of an additional loss which discourages the attention function from using future frames during training. Specifically, given token S which belongs to word W in label sequence L, we find a region $R_s = (W_t, \infty)$ in which W_t is the end time of W provided by a Viterbi alignment. The attention-based constraint loss is computed as the sum of all attention scores within the region R_s for all S in L:

$$\mathcal{L}_{attn} = \alpha \sum_{S} \sum_{x}^{|R_s|} Ascore_S^x$$

The tuneable parameter α adjusts the influence of the constraint loss to the maximum likelihood loss of the label sequence during training. By minimizing the total of both losses, we expect that the attention function learns to produce *close-to-zero* scores for the constraint regions for all label tokens while still minimizing the main loss.

9.2.3 Inference for Partial Stable Hypothesis

Beam search is the most efficient approach for the inference of S2S models. Its idea is to maintain a search network in which network paths are extended with new nodes with the highest accumulated scores and to then prune the network only keeping a set of active paths (or hypotheses). Typically, the most probable hypothesis for an utterance X is found and guaranteed when the entire search space constructed from X is supplied to the search. However, needing the complete acoustic signals of X to its very end in order to output the inference result is not efficient for a streaming setup. A streaming recognizer must be able to produce partial output while processing partial input. In this section, we describe our search algorithm applied to the proposed S2S model to produce partial output while retaining high accuracy.

Assume that in a streaming setup, at time t we use the proposed S2S model to perform inference for t audio frames. Given a context sequence C, the attention function is used to generate t attention scores for the prediction of the next token. We find a time $t_c <= t$ such that the sum of all attention scores from the *covering* window

$w = [0, t_c]$ is equal to a constant $\theta = \sum_x^{|t_c|} Ascore^x$. When $\theta = 0.95$, w covers all dominant attention scores and the context vector generated from w is almost the same as from $[0, t]$. If t_c is observed to be unchanged when t keeps growing, then we consider t_c as the endpoint of C. During stream processing, we use a term Δ to determine if endpoint t_c finally gets fixed as $t_c < t - \Delta$.

We then incorporate the information of endpoints into the beam search to find a partial stable hypothesis. Assume that our beam search can always perform in real-time for t audio frames to produce N considered hypotheses. If all N hypotheses share the same prefix sequence C and the endpoint of C is determined, then we consider C to be an *immortal* part that will not change anymore in the future. When more audio frames are available in the stream, C will be used as the prefix for all search hypotheses, and we repeat this step to find a longer stable hypothesis. Except the condition on endpoints, the idea of finding *immortal prefix* is similar to the partial trace-back [BSHB82, SAHW11] used in HMM-based speech recognizers.

In addition to the immortal prefix, we also investigated a more straightforward method in which we only consider *best-ranked* hypothesis and decide on a stable part C based solely on the term Δ. This approach is inspired from the incremental speech recognition proposed in [WFS98].

9.2.4 Bidirectional Encoder

To achieve high performance, bidirectional LSTM have been the optimal choice for the encoder of LSTM-based S2S models. However, due to the backward LSTM, bidirectional LSTM are *not* suited to provide partial and low-latency output as needed for streaming recognizers. The addition of acoustic input will affect all of the encoder's hidden states, which then makes all partial inference results unstable. This effect leads to the fact that stable output can be confidently inferred only when the input is complete. Therefore, earlier works [RLL+17, HSP+19, NPC+19] switched to unidirectional LSTM in their online models.

In this work, we try to utilize bidirectional LSTM for high-performance speech recognition in a streaming scenario. In the first setting, we investigated the use of the S2S model with a fully bidirectional encoder. First, we train the S2S model with optimal settings found for an offline setup, and with the attention-based constraint

loss proposed in Section 9.2.2. Then, during inference, we updated the encoder's hidden states from all available acoustic input before performing the search approaches in Section 9.2.3 to find stable hypotheses. As will be shown later, the use of a bidirectional LSTM as this way is possible since the proposed inference methods rely on the determination of endpoints, and the update of encoder's hidden states leads to stabilizing this determination.

In addition to fully bidirectional LSTM, we also experimented with a chunk-based BLSTM approach. During training, we divide input sequences into many non-overlapping blocks of a fixed size of K, and then use a BLSTM to compute each block sequentially. To benefit from long-context learning, we initialize the forward LSTM with its last hidden states after processing the previous chunk. The initialization of the backward LSTM can either be a constant or from the previous chunk. By doing that, the encoder's hidden states can be computed incrementally and efficiently as for unidirectional LSTM. This chunk-based approach is different from [AST+19] and the latency-controlled BLSTM [FZC+19b, XY17] that adopt constant initialization of both directions.

9.2.5 Experiments

Our experiments were conducted on the Fisher+Switchboard corpus consisting of 2,000 hours of telephone conversation speech. The Hub5'00 evaluation data was used as the test set. All the experimental models use the same input features of 40 dimensional log-mel filterbanks to predict 4,000 BPE sub-word units generated with the SentencePiece toolkit from all the training transcripts. The models with bidirectional encoder employ six layers of 1024 units while it is 1536 for the unidirectional encoders. We used only 1-head for the attention function in all setups. All models were trained with a dropout of 0.3. We further used the combination of two data augmentation methods *Dynamic Time Stretching* and *SpecAugment* proposed in [NSNW19] to reduce model overfitting. We use Adam [KB15] with an adaptive learning rate schedule to perform 12,000 updates during training. The model parameters of the 5 best epochs according to the perplexity on the cross-validation set are averaged to produce the final model.

For beam search, we use neither length normalization nor a language model. With a beam size of 8, the experimental models typically achieve their optimal accuracy.

9.2.6 Effect of the Constraint Loss

Table 9.3: WER performance of the S2S model with bidirectional encoder trained with different scales of the constraint loss.

Model	α	SWB	CH	Hub5'00
6x1024 BLSTM		5.9	11.8	8.9
	0.40	6.2	12.2	9.2
	0.20	6.1	12.1	9.1
	0.05	5.8	12.0	8.9

In this section, we evaluate the influence of the constraint loss proposed in Section 9.2.2 on the training of the S2S model. We started by using a high value for α and exponentially decreased it to train several systems for comparison. As observed during training, the constraint loss gets small quickly to a stable value depended on α. Joint training slows down the convergence of the main loss but does not have a significant impact on the final performance. As shown in Table 9.3, WERs are slightly worse with high α and can be similar to the regular training when α is small (e.g., 0.05). Different from that, the constraint loss may largely change the behavior of the attention function. For example, in Figure 9.4b, the attention function moves the high scores of the mismatched alignment to start frames, instead of start and end frames as in the regular training. We also found an extreme case when $\alpha = 0.4$. The attention-based alignment does not correspond at all to the proper alignment as illustrated Figure 9.4c.

Using the model trained with $\alpha = 0.05$, we follow the approach in Section 9.2.3 to extract the endpoints for all prefixes found during the inference of the evaluation set. We could verify that the extracted endpoints in all sentences match the expectation for streaming inference described in Section 9.2.2. So we keep this model for further experiments.

Table 9.4: Latency and accuracy of the S2S model with bidirectional encoder on Hub5'00 test set.

Method	Beam Size	Δ	WER	Latency
Force-Alignment	8		8.9	0.60
	4		9.1	0.60
	2		9.3	0.60
Immortal Prefix	8	20	**8.9**	**0.93**
	8	30	8.9	0.93
	4	20	9.2	0.86
	4	30	**9.1**	**0.87**
	2	20	12.6	0.74
	2	40	10.1	0.79
	2	60	9.5	0.83
	2	80	9.3	0.86
1st-Ranked Prefix	8	30	11.2	0.75
	8	50	9.6	0.80
	8	70	9.3	0.84
	4	30	11.3	0.75
	4	50	9.6	0.80
	4	70	9.3	0.84
	2	10	25.8	0.62
	2	30	11.4	0.75
	2	50	9.7	0.80
	2	70	9.3	0.84
Combination	8	20-70	9.2	0.83
	4	30-70	**9.4**	**0.81**
	2	60-70	9.5	0.83

9.2.7 Latency on Various Conditions

Using the S2S model with a bidirectional encoder trained with the constraint loss scale $\alpha = 0.05$, we performed several experiments with the inference approaches described in Section 9.2.3. In the experiments, the streaming scenario is simulated by repeatedly feeding an additional audio chunk of 250 ms to the experimental systems for incremental inferences. All the inferences were performed on a single Nvidia Titan RTX GPU, which produced an average RTF of 0.065 with a beam size of 8. The RTF result shows that real-time capacity is not a bottleneck problem in this setup. So we focus on the latency measure proposed in Section **??**.

For baselines, we computed the offline WER performance with the beam sizes 8, 4, 2, and then used a force-alignment system to produce the *ideal* latency from the offline transcripts. The ideal latency is always 0.6. If we shift the time alignment of the transcripts with 250 ms (i.e., all the outputs have a delay of 250 ms), 500 ms, 1 second, and 1.5 seconds, then we obtained a latency of 0.71, 0.78, 0.86 and 0.91 respectively.

Table 9.4 presents the accuracy and the latency we achieved when using the *immortal prefix* and *1st-ranked prefix* inference methods with several settings of Δ. Overall, the two methods are consistent with the observations in the HMM-based systems [BSHB82, SAHW11, WFS98]. Using the *immortal prefix* condition, the final accuracy can be guaranteed as for the offline inference for large beam sizes, e.g., 8 and 4. For a smaller beam size, this condition is not strong enough to deal with unstable partial results – probably due to the changes of the encoder's hidden states. In the *1st-ranked prefix* approach, increasing Δ allows for a flexible trade-off between the accuracy and the latency. The offline accuracy can also be achieved if a very large Δ is applied. These results consolidate our findings in two aspects. First, the integration of Δ is reliable and crucial for the streaming inferences to work efficiently. And second, the use of the bidirectional LSTM for the encoder is possible and results in high accuracy.

To achieve 8.9% WER (the offline accuracy), the system needs to delay outputs with an average duration of about 1.5 seconds. To obtain a lower latency of 1 second, the WER increases to 9.2%, e.g., by using the *immortal prefix* method with $\Delta = 20$ and $beamsize = 4$. The *combination* of both methods is efficient if we want to reach a latency of 0.81, which is closer to the average delay of 0.5 seconds.

9.2.8 Performance of Different Encoders

The shortcoming of the bidirectional encoder lies on the re-computation of the entire encoder's hidden states for every addition of input signal in the stream. In this section, we investigate two additional network architectures, *unidirectional* LSTM and *chunk-based* BLSTM described in Section 9.2.4, that improve the computational efficiency of the encoder. For chunk-based, we experimented with $K = 80$ and $K = 200$, as the chunk sizes of 800 ms and 2 seconds. We constantly found that initializing the backward LSTM from the last hidden state of the previous chunk is

Table 9.5: Latency and accuracy of the S2S models with unidirectional and chunk-based encoders using immortal prefix.

Encoder	Beam Size	Δ	WER	Latency
Unidirectional	8	30	12.7	0.94
	8	∞	12.6	1.00
	2	20	13.6	0.82
	2	30	13.2	0.85
	2	∞	13.1	1.00
Chunk-based *K=80*	8	30	10.5	0.91
	8	∞	10.4	1.00
	2	20	11.1	0.80
	2	30	10.9	0.82
	2	∞	10.8	1.00
Chunk-based *K=200*	8	30	10.3	0.89
	8	∞	10.0	1.00
	2	30	11.3	0.79
	2	60	10.8	0.85
	2	∞	10.7	1.00

better than a constant, so we only present the results of this approach. We evaluated two types of encoders in two categories: the best accuracy and the accuracy that the systems retain when maintaining an average delay of 1 second. To do so, we use the same *immortal prefix* inference and experiment with different settings of beam size and Δ.

As shown in Table 9.5, there is a big gap between the best WER of the unidirectional and bidirectional encoders (12.6% vs. 8.9%). The chunk-based encoder closes the gap and moves closer to the performance of the bidirectional encoder when a large chunk size is used. As the encoder's states are fixed early, the inferences are already stable when $\Delta = 30$ for all beam sizes. To achieve 1-second delay, all the approaches need to trade-off for an accuracy reduction of 5% relatively. In term of latency, the chunk-based approach with $K = 80$ and $beamsize = 2$ and $\Delta = 30$ is the best setting in this setup.

Chapter 10

Conclusion

In this thesis, we tackled the problem of online streaming speech recognition. Online streaming speech recognition is different from conventional speech recognition in which both latency and accuracy play an equally important role and need to be addressed together. We approached this problem with a two-stage approach. In the first stage, we constructed and evaluated different neural network models to build high-performance recognition systems in offline condition. In the second stage, we explored further techniques that enable the high-performance neural network models for online streaming processing while maintaining their efficiency as in offline. The success of the two-stage approach resulted in a speech recognition system that satisfies the need for both latency and accuracy.

We started by investigating all the current modeling approaches in automatic speech recognition. While the traditional Hidden Markov Model / Artificial Neural Network (HMM/ANN) model has been the mainstream for a long time, the newly introduced end-to-end models with connectionist temporal classification (CTC) and sequence-to-sequence (seq2seq) learning are attractive approaches as they only use single neural networks for directly mapping from acoustic signals to textual transcription. So far, several advances in architectural modifications and optimization have been proposed to improve the training of end-to-end models. However, their performance is still behind highly engineered traditional HMM/ANN models unless a large amount of training data (several thousand hours) employed.

We argued that the seq2seq learning approach is more likely the better approach

for performance improvement as it allows us to learn and optimize both acoustic and language modeling in a single training. Pursuing in this direction, we revealed that overfitting is the most critical issue when training seq2seq models and has not been solved efficiently with available techniques. We proposed to use the combination of three on-the-fly data augmentation methods to overcome this problem. *Dynamic Time Stretching* and *SpecAugment* modify the input sequences from different inspirations and aim to improve the generalization of the log-mel spectrogram encoder. *Sub-sequence Sampling* improves the decoder by adding sub-samples of target sequences. All of these methods can be optimized together with the model, which remains the simplification of training seq2seq speech recognition. On the telephone conversation benchmark, our seq2seq model optimized with the proposed data augmentation achieved a WER of 5.2% on the Switchboard test set, which is so far the state-of-the-art and on par with human performance.

While the proposed seq2seq model has shown to achieve appealing offline performance, it encounters latency issues when used in online conditions as both the attention mechanism and the bidirectional encoder network, require an entire input sequence to achieve the optimal performance. Tackling these latency issues, we first proposed an additional training loss, which prevents the attention mechanism from using unnecessary future frames. We then proposed modifications to the beam search inference to provide partial stable output and to cope with the bidirectional encoder's incremental update. We also argued that the common real-time factor is not a proper choice for measuring the user-perceived latency in online and streaming setup and proposed a novel and suitable technique for the replacement.

In contrast to the earlier works in the literature, our experimental results proved that a bidirectional encoder could be combined with suitable inference methods to produce high accuracy and low latency speech recognition output. With a delay of 1.5 seconds in all output elements, our streaming recognizer can achieve an offline system's ideal performance with the same configuration. To the best of our knowledge for the first time, a S2S speech recognition model can be used in online conditions without scarifying accuracy.

Bibliography

[AKR+] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny. Building competitive direct acoustics-to-word models for english conversational speech recognition. In *ICASSP 2018*. 60, 64

[AOKO11] Md Jahangir Alam, Pierre Ouellet, Patrick Kenny, and Douglas O'Shaughnessy. Comparative evaluation of feature normalization techniques for speaker verification. In *International Conference on Nonlinear Speech Processing*, pages 246–253. Springer, 2011. 45

[ARS+17] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo. Direct acoustics-to-word models for english conversational speech recognition. *arXiv preprint arXiv:1703.07754*, 2017. 41, 64

[AST+19] Kartik Audhkhasi, George Saon, Zoltán Tüske, Brian Kingsbury, and Michael Picheny. Forget a bit to learn better: Soft forgetting for ctc-based automatic speech recognition. *Proc. Interspeech 2019*, pages 2618–2622, 2019. 83

[B+95] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995. 11

[Bak75] James Baker. The dragon system–an overview. *IEEE Transactions on Acoustics, speech, and signal Processing*, 23(1):24–29, 1975. 4

[BBB+] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David

Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. 29

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 9, 10

[BCS+16] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016. 64

[BLP+12] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*, 2012. 29

[BM12] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012. 16

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 9

[BSHB82] P Brown, J Spohrer, P Hochschild, and J Baker. Partial traceback and dynamic programming. In *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 1629–1632. IEEE, 1982. 82, 86

[CBS+15] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, 2015. 9, 18, 51, 64

[CG99] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999. 4

[CJLV] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP 2016*. 9, 55, 64

[CJLV15] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015. 18, 51

[CNS+13] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Frederico. Report on the 10th IWSLT evaluation campaign. In *The International Workshop on Spoken Language Translation (IWSLT) 2013*, 2013. 47

[CR17] Chung-Cheng Chiu and Colin Raffel. Monotonic chunkwise attention. *arXiv preprint arXiv:1712.05382*, 2017. 25, 78

[CSW+] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *ICASSP 2018*. 51, 55

[CZQY17] Zhehuai Chen, Yimeng Zhuang, Yanmin Qian, and Kai Yu. Phone synchronous speech recognition with ctc lattices. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25:90–101, 2017. 42

[DHS00] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification. *Wiley-Interscience*, 2000. 46

[DKD+11] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011. 32, 44, 45

[Elm90] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 7

[FÖ8] Christian Fügen. *A system for simultaneous translation of lectures and speeches*. PhD thesis, Universität Karlsruhe (TH), 2008. 72, 73

[FGH+97] Michael Finke, Petra Geutner, Hermann Hild, Thomas Kemp, Klaus Ri es, and Martin Westphal. The karlsruhe VERBMOBIL speech recognition engine. In *Proc. of ICASSP*, 1997. 29, 76

[FL16] Christian Federmann and William D Lewis. Microsoft speech language translation (MSLT) corpus: The IWSLT 2016 release for english, french and german. In *The International Workshop on Spoken Language Translation (IWSLT) 2016*, 2016. 47

[Fur81] Sadaoki Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(2):254–272, 1981. 45

[FWK07] Christian Fügen, Alex Waibel, and Muntsin Kolss. Simultaneous translation of lectures and speeches. *Machine Translation*, 21:209–252, 2007. 72

[FZC+19a] Ruchao Fan, Pan Zhou, Wei Chen, Jia Jia, and Gang Liu. An online attention-based model for speech recognition. *Proc. Interspeech 2019*, pages 4390–4394, 2019. 78

[FZC+19b] Ruchao Fan, Pan Zhou, Wei Chen, Jia Jia, and Gang Liu. An online attention-based model for speech recognition. *Proc. Interspeech 2019*, 2019. 83

[Gal98] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998. 31, 32

[Gal99] Mark JF Gales. Semi-tied covariance matrices for hidden markov models. *IEEE Transactions on speech and audio processing*, 7(3):272–281, 1999. 34

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 12

[GBM+11] Ondřej Glembek, Lukáš Burget, Pavel Matějka, Martin Karafiát, and Patrick Kenny. Simplification and optimization of i-vector extraction.

In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4516–4519, 2011. 19

[GFGS06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. 17, 37, 38, 39, 42, 63

[GHM92] John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In [*Proceedings*] *ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE, 1992. 61

[GJ14] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772, 2014. 17, 37, 38

[GKKC07] Frantisek Grézl, Martin Karafiát, Stanislav Kontár, and Jan Cernocky. Probabilistic and bottle-neck features for lvcsr of meetings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–757. IEEE, 2007. 39

[GL94] J-L Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298, 1994. 32

[GMMW13] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3377–3381. IEEE, 2013. 31, 33, 34, 47

[Gra97] David Graff. The 1996 broadcast news speech and language-model corpus. In *Proceedings of the DARPA Workshop on Spoken Language technology.*, 1997. 34, 47

[GSS02] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002. 9

[HBF+01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. 9

[HCC+14] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014. 51

[HCKL17] Kyu J Han, Akshay Chandrashekaran, Jungsuk Kim, and Ian Lane. The capio 2017 conversational speech recognition system. *arXiv preprint arXiv:1801.00059*, 2017. 60

[HES00] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1635–1638. IEEE, 2000. 39

[HNG+18] François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia Tomashenko, and Yannick Estève. Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International Conference on Speech and Computer*, pages 198–208. Springer, 2018. 61

[Hoc98] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998. 9

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 9

[HSK+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 12

[HSP+19] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385. IEEE, 2019. 82

[JBM75] Frederick Jelinek, Lalit Bahl, and Robert Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21(3):250–256, 1975. 4

[JH13] Navdeep Jaitly and Geoffrey E Hinton. Vocal tract length perturbation (VTLP) improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, 2013. 51

[Jor97] Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997. 7

[KA] Gakuto Kurata and Kartik Audhkhasi. Guiding ctc posterior spike timings for improved posterior fusion and knowledge distillation. *Proc. Interspeech 2019*. 60

[KB13] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013. 9

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. of ICLR*, 2015. arXiv:1412.6980. 56, 65, 83

[KFN98] Timo Kaukoranta, Pasi Franti, and Olli Nevalainen. A new iterative algorithm for vq codebook generation. In *Image Processing, 1998. ICIP*

98. Proceedings. 1998 International Conference on, volume 2, pages 589–593, 1998. 34

[KLK17] Naoyuki Kanda, Xugang Lu, and Hisashi Kawai. Minimum bayes risk training of ctc acoustic models in maximum a posteriori based decoding framework. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4855–4859. IEEE, 2017. 38

[KMC+17] Chanwoo Kim, Ananya Misra, Kean Chin, Thad Hughes, Arun Narayanan, Tara N Sainath, and Michiel Bacchiani. Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home. *Proc. Interspeech 2017*, pages 379–383, 2017. 60

[KPP+] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. A study on data augmentation of reverberant speech for robust speech recognition. In *ICASSP 2017*. 51

[KPPK15] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *INTERSPEECH*, pages 3586–3589, 2015. 51, 52, 57

[KTO] Naoyuki Kanda, Ryu Takeda, and Yasunari Obuchi. Elastic spectral distortion for low resource speech recognition with deep neural networks. In *ASRU 2013*. 51

[LB+95] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 7

[LBOM12] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012. 45

[Lia13] Hank Liao. Speaker adaptation of context dependent deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7947–7951. IEEE, 2013. 32

[MB90] Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in neural information processing systems*, pages 630–637, 1990. 13

[MCZ+19] Haoran Miao, Gaofeng Cheng, Pengyuan Zhang, Ta Li, and Yonghong Yan. Online hybrid ctc/attention architecture for end-to-end speech recognition. *Proc. of Interspeech 2019*, pages 2623–2627, 2019. 78

[MFSW16] Markus Müller, Sarah Fünfer, Sebastian Stüker, and Alex Waibel. Evaluation of the kit lecture translation system. In *Proc. of LREC*, 2016. 72

[MGM15] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 167–174. IEEE, 2015. 40, 41

[MHP12] Abdel-rahman Mohamed, Geoffrey Hinton, and Gerald Penn. Understanding how deep belief networks perform acoustic modelling. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4273–4276, 2012. 35

[MJZM14] Yajie Miao, Lu Jiang, Hao Zhang, and Florian Metze. Improvements to speaker adaptive training of deep neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 165–170. IEEE, 2014. 44

[MKC+11] Lidia Mangu, Hong-Kwang Kuo, Stephen Chu, Brian Kingsbury, George Saon, Hagen Soltau, and Fadi Biadsy. The ibm 2011 gale arabic speech transcription system. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 272–277. IEEE, 2011. 19

[MNS+15] Markus Müller, Thai-Son Nguyen, Matthias Sperber, Kevin Kilgour, Sebastian Stüker, and Alex Waibel. The 2015 KIT IWSLT speech-to-text systems for English and German. In *Proc. of IWSLT*, 2015. 76

[MS69] Minsky Marvin and A Papert Seymour. Perceptrons, 1969. 6

[MZM15] Yajie Miao, Hao Zhang, and Florian Metze. Speaker adaptive training of deep neural network acoustic models using i-vectors. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(11):1938–1949, 2015. 19

[NMS+18] Arun Narayanan, Ananya Misra, Khe Chai Sim, Golan Pundak, Anshuman Tripathi, Mohamed Elfeky, Parisa Haghani, Trevor Strohman, and Michiel Bacchiani. Toward domain-invariant speech recognition via large scale training. *arXiv preprint arXiv:1808.05312*, 2018. 60, 61

[NPC+19] Arun Narayanan, Rohit Prabhavalkar, Chung-Cheng Chiu, David Rybach, Tara N Sainath, and Trevor Strohman. Recognizing long-form speech using streaming end-to-end models. *arXiv preprint arXiv:1910.11455*, 2019. 82

[NSNW19] Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. *arXiv preprint arXiv:1910.13296*, 2019. 9, 79, 83

[PCPK15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015. 61, 62

[PCZ+] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *Proc. of Interspeech 2019*. 9, 51, 53, 54, 57, 58, 60

[PGB+11a] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584, 2011. 35

[PGB+11b] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. 45

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 29

[PMN06] Pere Pujol, Dusan Macho, and Climent Nadeu. On real-time mean-and-variance normalization of speech recognition features. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006. 45

[PMW+16] Vijayaditya Peddinti, Vimal Manohar, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur. Far-field asr without parallel data. In *INTERSPEECH*, pages 1996–2000, 2016. 60

[PNN+] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Muller, and Alex Waibel. Very deep self-attention networks for end-to-end speech recognition. *Proc. of Interspeech 2019*. 54, 55, 56, 58, 60

[PPG+] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. *Interspeech 2016*. 60

[PPK15] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*, pages 3214–3218, 2015. 45, 47

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 64

[RDE14a] Anthony Rousseau, Paul Deléglise, and Yannick Estève. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *LREC*, 2014. 34, 47

[RDE14b] Anthony Rousseau, Paul Deléglise, and Yannick Estève. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *Proc. of LREC*, 2014. 76

[RF87] AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering, 1987. 8

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 6

[RKRG] A Ragni, KM Knill, SP Rath, and MJF Gales. Data augmentation for low resource languages. In *Proc. of Interspeech 2014*. 51

[RLL+17] Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2837–2846. JMLR. org, 2017. 25, 78, 82

[RM85] David E Rumelhart and James L McClelland. On learning the past tenses of english verbs. Technical report, CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1985. 8

[Ros57] Frank Rosenblatt. The perceptron, a perceiving and recognizing automaton project para. In *Cornell Aeronautical Laboratory*, 1957. 6

[RPVČ13] P. Shakti Rath, Daniel Povey, Karel Veselý, and Jan Černocký. Improved feature processing for deep neural networks. In *Proceedings of Interspeech 2013.*, number 8, pages 109–113, 2013. 34, 49

[SAHW11] Ethan O Selfridge, Iker Arizmendi, Peter A Heeman, and Jason D Williams. Stability and accuracy in incremental speech recognition. In *Proceedings of the SIGDIAL 2011 Conference*, pages 110–119. Association for Computational Linguistics, 2011. 82, 86

[SBB+10] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Garret, and Brian Strope. "Your Word is my Command": Google search by voice: A case study. In *Advances in Speech Recognition*, pages 61–90. Springer, 2010. 67

[SdCQS+15] Haşim Sak, Félix de Chaumont Quitry, Tara Sainath, Kanishka Rao, et al. Acoustic modelling with cd-ctc-smbr lstm rnns. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 604–609. IEEE, 2015. 38

[Sha01] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001. 4

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 2014. 53

[SKK12a] Sebastian Stüker, Kevin Kilgour, and Florian Kraft. Quaero 2010 speech-to-text evaluation systems. In *High Performance Computing in Science and Engineering'11*, pages 607–618. Springer, 2012. 34, 47

[SKK12b] Sebastian Stüker, Kevin Kilgour, and Florian Kraft. Quaero 2010 speech-to-text evaluation systems. In *High Performance Computing in Science and Engineering'11*, pages 607–618. Springer, 2012. 76

[SKMR13] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. Learning filter banks within a deep neural network framework. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 297–302. IEEE, 2013. 45

[SKR12] Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4153–4156. IEEE, 2012. 19

[SKS+] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *Proc. Interspeech 2017*. 60

[SLCY11] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 24–29. IEEE, 2011. 16, 32

[SLM14] Andrew Senior and Ignacio Lopez-Moreno. Improving dnn speaker independence with i-vector inputs. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014. 19, 32, 35, 44

[SLS16] Hagen Soltau, Hank Liao, and Hasim Sak. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*, 2016. 63, 64

[SMKR13] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*, pages 8614–8618. IEEE, 2013. 48

[SRBG02] Murat Saraclar, Michael Riley, Enrico Bocchieri, and Vincent Goffin. Towards automatic closed captioning: low latency real time broadcast news transcription. In *Proc. of INTERSPEECH*, 2002. 67

[SSNP13] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *ASRU*, pages 55–59, 2013. 19, 32, 34, 35, 44

[SSR+15] Haşim Sak, Andrew Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk. Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4280–4284. IEEE, 2015. 39, 42

[SSRB15] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. *arXiv preprint arXiv:1507.06947*, 2015. 63

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 9

[TKKW19] Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. Towards online end-to-end transformer automatic speech recognition, 2019. 78

[TSN19] Alex Waibel Thai-Son Nguyen, Sebastian Stueker. Learning shared encoding representation for end-to-end speech recognition models. *arXiv preprint arXiv:1904.02147*, 2019. 64, 65

[VL98] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1):133–147, 1998. 45

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *The 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008. 33, 48

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention

is all you need. In *Advances in neural information processing systems*, 2017. 55, 79

[WCW+] Chao Weng, Jia Cui, Guangsen Wang, Jun Wang, Chengzhu Yu, Dan Su, and Dong Yu. Improving attention based sequence-to-sequence models for end-to-end english conversational speech recognition. *Proc. Interspeech 2018*. 51, 55, 60

[Wer88] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988. 8

[WFS98] Sven Wachsmuth, Gernot A Fink, and Gerhard Sagerer. Integration of parsing and incremental speech recognition. In *9th European Signal Processing Conference (EUSIPCO 1998)*, pages 1–4. IEEE, 1998. 82, 86

[WHH+87] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. In *Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE)*, pages SP87–100. IEICE, 1987. 7

[WPM+15] Will Williams, Niranjani Prasad, David Mrva, Tom Ash, and Tony Robinson. Scaling recurrent neural network language models. pages 5391–5395. IEEE, 2015. 62

[XY17] Shaofei Xue and Zhijie Yan. Improving latency-controlled blstm acoustic models for online speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5340–5344. IEEE, 2017. 83

[YS11] Dong Yu and Michael L Seltzer. Improved bottleneck features using pretrained deep neural networks. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011. 19, 47

[YSL+13] Dong Yu, Michael L Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide. Feature learning in deep neural networks - studies on speech recognition tasks. *arXiv preprint arXiv:1301.3605*, 2013. 60

[ZBSN17] Albert Zeyer, Eugen Beck, Ralf Schlüter, and Hermann Ney. Ctc in the context of generalized full-sum hmm training. 2017. 39, 42

[ZCMS04] Qifeng Zhu, Barry Chen, Nelson Morgan, and Andreas Stolcke. On using mlp features in lvcsr. In *Eighth International Conference on Spoken Language Processing*, 2004. 39, 43, 44

[ZISN] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney. Improved training of end-to-end attention models for speech recognition. *Proc. Interspeech 2018*. 51, 55, 60

[ZRM+13] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013. 47, 48

[ZSN16] Albert Zeyer, Ralf Schlüter, and Hermann Ney. Towards online-recognition with deep bidirectional LSTM acoustic models. *Interspeech 2016*, pages 3424–3428, 2016. 45