TOWARDS DEEP LEARNING ON SPEECH RECOGNITION FOR KHMER LANGUAGE

A Thesis
presented to
the Faculty of Graduate School
at the University of Missouri-Columbia

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by

CHANMANN LIM
Dr. Yunxin Zhao, Thesis Supervisor

May 2016

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

TOWARDS DEEP LEARNING ON SPEECH RECOGNITION FOR KHMER LANGUAGE

presented by Chanmann Lim, a candidate for the degree of Master of Science, and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Yunxin Zhao . . . .
Thesis Supervisor
Professor of Computer Science

Professor Jianlin Cheng . . . .
Member, Thesis Committee
Associate Professor of Computer Science

Professor Dominic K.C. Ho . . . .
Member, Thesis Committee
Professor of Electrical & Computer Engineering

Acknowledgments

This piece of work would not have been achievable without the constant support and help from many great people whom I feel so blessed to know and so proud to be surrounded by.

First of all I would like to express my wholehearted thanks to Dr. Yunxin Zhao, my faculty advisor, who not only provides guidances but also encourages and inspires me in pursuing my dream. She spared immeasurable time, energy, wisdom and heart into consulting, teaching and engaging all phases of this work.

I would also like to express my gratitude to Dr. Dominic K.C. Ho and Dr. Jianlin Cheng for spending their precious time serving as my committee members and giving their careful reviews and insightful comments on my thesis.

I am indebted to my colleagues and friends in the Spoken Language and Information Processing Lab at the University of Missouri-Columbia for their valuable ideas, discussions and friendship. Special thanks to Tuo Zhao for his collaborative contributions in the implementation of the project.

I am also very grateful to Fulbright Program for providing financial support during my graduate study at the University of Missouri-Columbia.

Finally, I would like to thank both of my parents and my wife Molinna whose unconditional love and unwavering trust in me have never grown faint over time or distance. I love each and every one of you!

Table of Contents

List of Tables

List of Figures

TOWARDS DEEP LEARNING ON SPEECH RECOGNITION FOR KHMER LANGUAGE

by

Chanmann Lim

Dr. Yunxin Zhao, Thesis Supervisor

Abstract

In order to perform speech recognition well, a huge amount of transcribed speech and textual data in the target language must be available for system training. The high demand for language resources constrains the development of speech recognition systems for new languages. In this thesis the development of a low-resourced isolated-word recognition system for “Khmer” language is investigated. Speech data, collected via mobile phone, containing 194 vocabulary words is used in our experiments. Data pre-processing based on Voice Activity Detection (VAD) is discussed. As by-products of this work, phoneme based pronunciation lexicon and state tying questions set for Khmer speech recognizer are built from scratch. In addition to the conventional statistical acoustic modeling using Gaussian Mixture Model and hidden Markov Model (GMM-HMM), a hybrid acoustic model based on Deep Neural Network (DNN-HMM) trained to predict context-dependent triphone states is evaluated. Dropout is used to improve the robustness of the DNN, and cross-lingual transfer learning that makes use of auxiliary training data in English is also investigated. As the first effort in using DNN-HMM for low-resourced isolated-word recognition for Khmer language, the system currently performs at 93.31% word accuracy in speaker-independent mode on our test set.

Chapter 1

Introduction

For centuries, scientists and engineers have been intrigued by the idea of building conversational speaking machines, ones that are capable of using natural language to fluently communicate with human. Countless science fiction movies and books describe the existence of humanoid robot or machine with miraculous speech understanding capability that can intelligibly converse with people. Automatic Speech Recognition (ASR) is undoubtedly one of the most important technology, which can promise to deliver such characteristic to machine and fulfill this long-awaited desire.

Over the past five decades, a great amount of intensive researches and remarkable achievements in the area of speech recognition and language understanding have taken place[1]. At present we have seen ASR applications that demonstrate impressive performance as a result of many major advances in signal processing, statistical modeling and machine learning algorithms including Linear prediction (LP) analysis, Fourier and Filterbank analysis , hidden Markov models (HMM), Gaussian mixture models (GMM), Phonetic decision tree (PDT), N-gram language models, and Deep neural networks (DNN), just to name a few, and various ASR model training techniques, including maximum likelihood and discriminative training.

In the recent years, speech recognition interface has gained widespread success for many innovative applications, such as virtual assistance on mobile devices, voice search providing realtime information access, video automatic captioning and machine translation, etc. Not only does it provide more user-friendly interface (in term of hands-free and eyes-free capability) over conventional human-machine interaction modalities such as keyboard, mouse or touch, but cross-language communication also becomes possible to overcome the technological and language barriers, from enhancing the human communication and machine interaction experience to enriching the quality of life in today's modern society.

This thesis can be divided into three parts. In the first part (Chapter 1) we introduce the general architecture of an ASR system and the principal components that make up a speech recognizer, including speech feature extraction, acoustic model (AM), pronunciation model, language model (LM) and decoding search. The second part (Chapter 2, 3 and 4) describes the data set we used in this work, conventional acoustic modeling using mixture of Gaussians and hidden Markov models (GMM-HMM), and hybrid acoustic modeling using deep neural network (DNN-HMM), as well as experiments and results for Khmer speech recognition. In the last part (Chapter 5 and 6) we further discuss the implementation issues and experiment outcomes, and then give a conclusion and outline the future work.

1.1 Motivation

Despite the fact that there exist well-known toolkits [2, 3, 4] for building speech recognition system, creating a speech recognizer for a brand new language remains a challenging task for an individual software developer for many reasons. First of all, a decent amount of quality speech data in the target language has to be made available a priori. Secondly, there are many phonetic and phonological aspects in a language to comprehend for accurate transcription of speech. Thirdly, it is considered a separate field of research by itself to handle those intricate speech signals effectively. Finally, a broad range of knowledge in statistical modeling and machine learning algorithms is necessary to make sense of the processes pertaining to speech recognition.

As a speaker of Khmer (ខ្មែរ), the official language of the Kingdom of Cambodia, we are interested in building an automatic speech recognition for our language. For us, the word “Khmer” is not only about the language we speak but it is also referred to the history, culture and identity of the nation and its people.

Today, the top ten languages in the world claim around half of the world’s population and it is controversial to ask “Can language diversity be preserved, or are we on a path to becoming a monolingual species?” [5]. As the globalized economy tends to favor those top languages, we nevertheless hope this work will leave a footprint of Khmer in this modern digital age. Last but not least, the zeal of creating a speech recognition system for our own language is extremely ecstatic.

1.2 Contribution

With respect to speech recognition, in order to perform well a huge amount of transcribed speech and textual data in the target language must be available for system training, which is the case for only a few major languages while other languages are dismissed as “low resource” especially those spoken in developing countries. In this thesis the development of a low-resourced speech recognition system in the case of “Khmer” language is investigated. Precisely, the scope of the work is limited to isolated-word recognition task trained with context-dependent subword model on Khmer keywords dataset [6].

Despite the nature of being somewhat restrictive, isolated-word recognition system is found suitable and efficient in a wide range of scenarios, for example, supporting interactive voice response (IVR) in telephone systems and voice commands, where a task domain’s vocabulary is compact and succinct.

A broadcast news transcription system for Khmer language was introduced in [7] where graphemes were the desired acoustic modeling units. In the current work, we instead model phoneme units which are more closely related to acoustic phenomenon in speech. The phonological question set and the phoneme set of Khmer words used in the task have been developed. In addition, the novelty of this work being the investigation on “Deep Learning” approach to Khmer speech recognition on top of conventional acoustic modeling using mixtures of Gaussians.

This work establishes the foundation for future researchers and software engineers who are interested in building Khmer ASR.

1.3 Speech Recognition Framework

A general framework for speech recognition is associated with the mechanism of speech production and perception of human beings. A message originated in a person's mind is delivered to the listener in the form of sound waves produced by the vibration of vocal cords and the passage of air through vocal tract. The recognition process begins by the listener processing the incoming acoustic signal within his or her inner ear where spectral features are being extracted. The language code, i.e, phoneme, word and sentence, is chosen to interpret those speech features, and eventually message comprehension occurs[8]. Likewise, speech recognition system, the machine listener, first receives and converts audio signal into a sequence of speech feature vectors. It is then fed into recognition hypothesis search, where the likelihood of each word string with respect to the input vectors is scored according to the already trained acoustic model and language model and finally the word sequence with the highest score is selected as the recognition result. Each recognized word in the string is constrained to be from the vocabulary in the pronunciation dictionary. Speech recognition process is illustrated in Figure 1-1.

Well-defined engineering techniques in signal processing take samples of the speech signal and turn them into a sequence of feature vectors X, defined as

$$X = x_1, x_2, \ldots, x_T \qquad ; x_t \in \mathbb{R}^{\mathrm{d}} \tag{1.1}$$

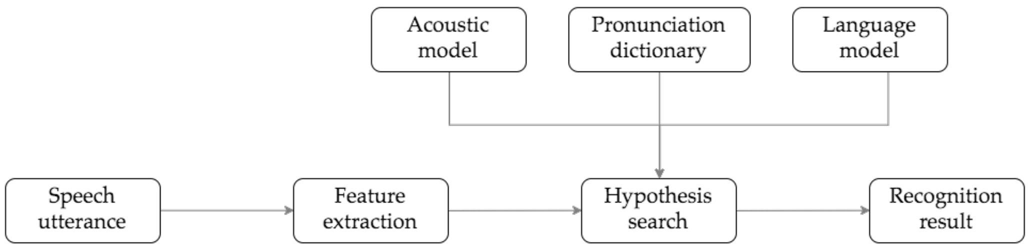

Figure 1-1: Speech recognition process

where x_t denote a speech vector observed at time t and d is the dimension of the x_t. The noisy channel theory in [9] provides mathematical formulation for speech recognition task to find a mapping between X and the underlying word sequence W such that the posterior probability $P(W|X)$ is maximized:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \; P(W|X) \tag{1.2}$$

Using Bayes' Rule, the formulation in (1.2) can be expressed as:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \; \frac{P(X|W)P(W)}{P(X)} \tag{1.3}$$

Having the probability of speech vectors $P(X)$ the same for all Ws does not affect the maximization process and thus the term can then be ignored.

In the log scale we obtained:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \; \{\log P(X|W) + \log P(W)\} \tag{1.4}$$

where likelihood of the speech vectors $P(X|W)$ involves computing score using acoustic model trained with speech data and prior probability of the word sequence $P(W)$ is obtained via language model trained with text data. This produces the best matching word sequence hypothesis $\hat{W}$ for the corresponding speech feature vectors.

1.4 Speech Processing and Feature Extraction

Human speech is produced in a time-varying manner through vocal tract resulting in a non-stationary signal. A microphone is used to capture the speech signal which is converted to data samples by a sampling process usually at the frequency between 8 kHz and 16kHz for speech recognition applications. Although the sampled waveform can preserve the characteristics of the speech signal, those raw data samples cannot immediately be used to recognize speech.

Spectral analysis on short-time windows of speech signal moving at the duration of the time constants of human articulatory apparatus (typically on the order of 10 ms) has to be carried out to form a new representation which is more efficient for recognizing speech. Such analysis involves transforming windows of speech samples from time domain into frequency domain and then feature vectors are extracted. There are several popular feature representations including Filter banks, Mel-frequency cepstral coefficients (MFCC) and Perceptual linear prediction (PLP) described in [8, 10].

Filter banks: use non-uniform bandwidth bins (mel-scale) emulating human ear sensitivity to sound frequency to obtain non-linear spectral analysis of speech signal via Fourier transform.

MFCC: use logarithmic scale on filter bank amplitudes and then apply Discrete cosine transform (DCT).

PLP: combine Discrete Fourier transform (DFT) and linear prediction (LP) to incorporate physiological factors in human auditory system [10].

In practice, further improvement on speech recognizer's performance is observed when the first-order and second-order time derivatives of the speech feature vectors are included to account for temporal dynamics in speech signal. The final speech vectors or *observations* obtained become the new feature representation.

1.5 Acoustic Modeling

Acoustic models are used to estimate the likelihood of a given sequence of speech feature vectors. Hidden Markov models are the most widely used probabilistic models in dealing with sequential data and thus provide a good facility for fitting speech data [11]. In HMM-based acoustic modeling, a subunit of a word, namely phone, is often used as the modeling unit in a fixed HMM architecture consisting of one entry state, several (typically three) emitting center states and an exit state as shown in Figure 1-2. Subword HMMs can then be concatenated to form the HMM for a word. Similarly, a sentence HMM can be described by the concatenation of a series of word HMMs.

The Left-to-right HMM in Figure 1-2 has its state evolution constrained to be either making self-loop to remain in the same state, or moving forward to next state.

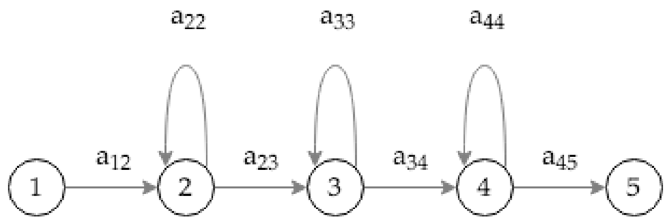

Figure 1-2: HMM in acoustic modeling

The transition probability a_{ij} represents the probability of moving from state i to state j:

$$a_{ij} = P(s(t) = j | s(t-1) = i) \tag{1.5}$$

where $s(t)$ is the state of the HMM at time t. The transition matrix $A = [a_{ij}]$ having all a_{ij}'s as its elements is commonly used to represent the conditional probability table of $P(s_j|s_i)$. Each row of A must be summed up to one except the last row corresponding to the exit state where all elements are zero, also the backward state transitions are not allowed:

$$a_{ij} = 0 \qquad \text{if } j < i \tag{1.6a}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{1.6b}$$

$$a_{Nj} = 0 \tag{1.6c}$$

where N is the number of distinct states of a HMM.

The observation x_t is assumed to be generated by the emission probability distribution of state j at time t, that is $p(x_t|s(t) = j)$ or $b_j(x_t)$ in short, $j = 1, \cdots, N$. The entry state and exit state are non-emitting and do not generate observations, however they can be used as the links connecting with another model. Typically, continuous density function in the form of a Gaussian mixture models (GMM) is widely adopted for computing $b_j(x_t)$:

$$b_j(x_t) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(x_t; \mu_{jm}, \Sigma_{jm}) \tag{1.7}$$

where M is the number of components of GMM, c_{jm} is the weight of the m-th Gaussian component constrained to

$$\sum_{m=1}^{M} c_{jm} = 1 \qquad ; 1 \leq j \leq N \tag{1.8a}$$

$$c_{jm} \geq 0 \qquad ; 1 \leq j \leq N,\ 1 \leq m \leq M \tag{1.8b}$$

and $\mathcal{N}(.; \mu_{jm}, \Sigma_{jm})$ is a multivariate Gaussian density having μ_{jm} and Σ_{jm} as mean vector and covariance, respectively:

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \qquad ; x \in \mathbb{R}^d \tag{1.9}$$

Theoretically, the complete specification of a HMM can be described by λ_{Model}, the parameter set of the model consisting of initial state's probability π, transition probability A and emission probability $B = \{b_j(x_t) \mid 1 \leq j \leq N\}$, i.e.,

$$\lambda_{Model} = (\pi, A, B) \tag{1.10}$$

The celebrated EM (expectation-maximization) algorithm or Baum-Welch in the context of HMM is applied in updating the parameter set λ_{Model} such that the likelihood of the training data given the parameter set of the HMM is locally maximized.

After λ_{Model} has been refined, The joint conditional likelihood of a subword HMM, $P(X, S|Model)$, can then be calculated as follows:

$$P(X, S|Model) = a_{s(0)s(1)} \prod_{t=1}^{T} b_{s(t)}(x_t) a_{s(t)s(t+1)} \tag{1.11}$$

where $s(0)$ and $s(T+1)$ denote the entry state and the exit state of the HMM, respectively.

1.5.1 Pronunciation Dictionary

The pronunciation of a word can be described by a sequence of phonemes, which are the basic modeling units of hidden Markov models. The phoneme based pronunciation lexicon/dictionary defines word-to-phonemes mappings necessitated for converting words into phoneme sequences. The likelihood of an observation sequence is calculated from a product of the likelihood of all phones constituting a word or sentence.

ARPAbet, a popular ASCII representation of international phonetic alphabet (IPA)[1], is widely used to describe word pronunciation in the pronunciation lexicon of English speech recognizer. Nevertheless, it can easily be altered and extended to represent the phonetic characteristics and linguistic differences of other target languages for the recognizer.

The decomposition of words into HMM's modeling units by a pronunciation lexicon makes it feasible for transcribing speech utterances using only a fixed set of predefined phonemes. Therefore, the parameters of each phone HMM can be estimated from a training speech corpus.

1.5.2 Phonetic Decision Tree

The acoustic model trained to characterize each atomic phone unit (monophone) individually so far is often referred to as Context-independent (CI) model, which assumes independence of speech at phonemic level. However, the nature of speech articulation is continuous and the pronunciation of a phone is severely influenced by its adjacent phones. This co-articulation phenomenon is common in speech production and its effect on speech recognition can be captured by incorporating neighboring contexts of the phones being modeled to form Context-dependent (CD) HMM. A triphone, which consists of a center phone, one left and one right phone as context, e.g., [s-aw+n] denoting [aw] as the center phone with [s] and [n] as the left and the right context respectively [2], is the most popular modeling unit for CD-HMM.

[1]https://en.wikipedia.org/wiki/International_Phonetic_Alphabet

[2]Context-dependent triphone notation followed [13]

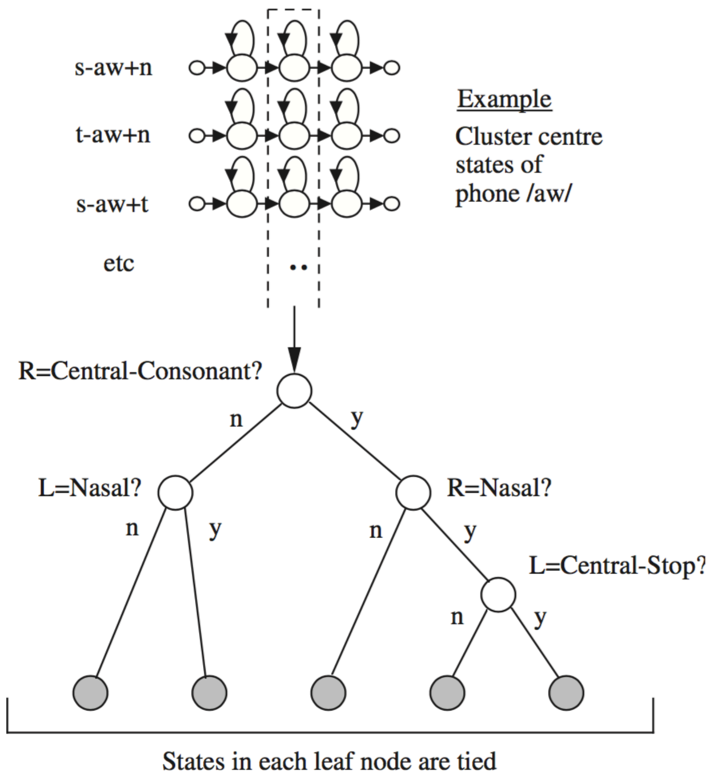

Figure 1-3: Phonetic decision tree [13]

Since there are many different combinations of left and right contexts in triphones, the number of triphone models increases in the cubic order to those of monophone. This leads to the problem of data sparsity since it becomes impracticable to have sufficient data to train each triphone model, and furthermore, there might be some triphones that rarely occur or should never exist in linguistic sense of the target language. In [13], Phonetic decision tree (PDT) clustering is used to reduce the number of model parameters by tying acoustically similar states of the triphone sets which allows more robust estimations of the shared parameters and makes it possible to predict the models of unseen triphones that do not occur during training.

As in Figure 1-3, a phonetic decision tree algorithm begins by asking phonetic questions about the left and right contexts of each triphone state. The questions are chosen such that the likelihood gain from the split is maximized. The splitting process is performed iteratively until the likelihood gain falls below a predefined threshold.

Moreover, the size of the clusters obtained is examined to see whether there are sufficient training samples for reliable parameter estimation. If the cluster occupancy falls below a user-defined minimum state occupancy threshold, the cluster is then merged with its nearest neighbor.

1.5.3 Hybrid Acoustic Model

The quality of an acoustic model is measured by how well the frames of speech spectral vectors can be correctly associated with the hidden states of HMMs. This criterion emphasizes the significance of the emission density function employed in acoustic model training.

The emission density $b_j(x_t)$ taking the form of a mixture of Gaussians has been dominantly used in the past several decades to describe the relationship between HMM states and the acoustic input represented by a series of spectral vectors. However, GMM has also been known to be statistically inefficient when dealing with data that lie on or near a non-linear manifold in the data space due to the growing number of GMMs' components required for modeling the data [14].

The modeling of emission probability of a HMM state using Deep neural network[3], known as *Hybrid DNN-HMM* model, has been proposed as an alternative and has demonstrated superiority over the conventional GMM-HMM models on various large vocabulary speech recognition (LVSR) tasks [14].

In this Hybrid acoustic model, a window of frames of spectral vector is fed in a neural network to produce the HMM states' output posterior probability $P(s(t)|\tilde{X})$ and the state likelihood is obtained via Bayes rule:

$$p(\tilde{X}|s(t)) = \frac{P(s(t)|\tilde{X})p(\tilde{X})}{P(s(t))} \tag{1.12}$$

where $\tilde{X} = x_{t-\Delta} \dots x_t \dots x_{t+\Delta}$ represents a window of spectral vector frames having x_t as the central frame and Δ is the number of neighboring frames with respect to x_t.

[3]Feedforward artificial neural network that has many hidden layers.

1.6 Language Modeling

The job of acoustic model is only to discriminate speech acoustically. However, there remains many phonetically similar word sequences to be further distinguished. Language model is used to assign prior probabilities to word sequences $P(W)$ such that the final prediction will not only be based on acoustic properties of the utterance but also structures of the language as defined in Equation (1.4).

The joint probability of a sequence of M words, $P(w_1, w_2, \ldots, w_M)$, by chain rule of probability, can be written as:

$$P(w_1, w_2, \ldots, w_M) = \prod_{i=1}^{M} P(w_i|w_1, \ldots, w_{i-1}) \tag{1.13}$$

where $w_1, \ldots, w_{i-1}$ is the prefix or *history* of w_i. That is, the probability of a word sequence is the product of the probabilities of the individual word given its *history*.

N-gram language model is commonly used to further simplify the natural language aspect in speech recognition by assuming that meaningful structures of a language can be obtained without the full knowledge of its *history* but a context of N words. In other words, the conditional dependency of w_i is constrained to only $N-1$ immediate words to its left:

$$P(w_1, w_2, \ldots, w_M) \simeq \prod_{i=1}^{M} P(w_i|w_{i-N+1}, \ldots, w_{i-1}) \tag{1.14}$$

Generally, while a smaller N makes a poorer language model due to the fact that language has long-distance dependencies, a larger N requires a vast amount of training data for reliable estimation.

In practice, bi-gram ($N = 2$) and tri-gram ($N = 3$) language models are the most popular and often work well. Besides, $P(w_i|w_{i-N+1}, \ldots, w_{i-1})$ in (1.14) is computed from maximum likelihood estimation by simply counting the events of $w_{i-N+1}, \ldots, w_i$ among those of $w_{i-N+1}, \ldots, w_{i-1}$ that occur in the training text corpus:

$$P(w_i|w_{i-N+1}, \ldots, w_{i-1}) = \frac{Count(w_{i-N+1}, \ldots, w_i)}{Count(w_{i-N+1}, \ldots, w_{i-1})} \tag{1.15}$$

Despite the fact that N-gram language model provides a simple procedure to obtain the prior probabilities of word sequences, there are two major issues that need to be addressed. One, there exist zero probabilities for unseen word sequences thus an effective smoothing method is necessary to drag some probability mass to fill the zero probabilities while retaining the distribution of the data. Another problem of N-gram is the domain-specific tendency leaning toward the training set, rendering a N-gram model useless for speech recognition tasks of different domains, and this requires careful selection of the training text corpus to mitigate the impacts incurred.

1.7 Hypothesis search

Hypothesis search or decoding in speech recognition makes use of the log probability scores produced by acoustic model and language model to find the most probable sentence corresponding to the given sequence of speech feature vectors of arbitrary length. In the context of HMM, Viterbi algorithm [15] is a commonly used algorithm for finding the best sequence of latent states given the observations.

$$S^* = \underset{S}{\operatorname{argmax}} \; P(S|X) \tag{1.16}$$

where S^* is the most probable state sequence.

Viterbi algorithm can be viewed as an application of dynamic programming [16] in which a complex problem is broken down into a series of independent subproblems and by solving each subproblems in a bottom-up manner recursively, the solution to the original problem can be obtained. Decoding using Viterbi algorithm is accomplished in two steps: first, forward extension step, and second, back-tracking step [8].

In the first step, an auxiliary term $\psi_j(t)$ defined as the probability of the most likely state sequence ending in state j at time t is computed from time 1 to T for the observation vectors of length T.

$$\psi_j(t) = \max_{s(1),\ldots,s(t-1)} P(X_{1:t}, s(1), \ldots, s(t-1), s(t) = j) \qquad j = 1, \cdots, N \tag{1.17}$$

By using chain rule, it can easily be shown that the recursion formulation of (1.17) takes the form:

$$\psi_j(t) = b_j(x_t) \max_i \{a_{ij} \psi_i(t-1)\} \tag{1.18}$$

and

$$\psi_j(1) = \pi_1 b_j(x_1) \tag{1.19}$$

where $\pi_1 = 1$ is the initial probability of an entry state.

In fact, logarithm of $\psi_j(t)$ is used in place of (1.18) since a long sequence of probability multiplications can cause an underflow error on floating point numbers very quickly:

$$\psi_j(t) = \log b_j(x_t) + \max_i \{\log(a_{ij}) + \psi_i(t-1)\} \tag{1.20}$$

In addition, the best previous state leading to the state j at time t is recorded in $\delta_j(t)$ whose role is to keep record of the trajectory of the state evolution as depicted in Figure 1-4:

$$\delta_j(t) = \operatorname*{argmax}_i \, \{\log(a_{ij}) + \psi_i(t-1)\} \tag{1.21}$$

In the second step of Viterbi algorithm, the path with the highest ending score $\psi_j(t)$ at last time step T is selected, and then the recursive lookup in backward order can be performed on $\delta_j(.)$ to reproduce the most likely state sequence.

While the search space in the forward extension step grows exponentially with respect to the progression of time index, the number of state sequences having high

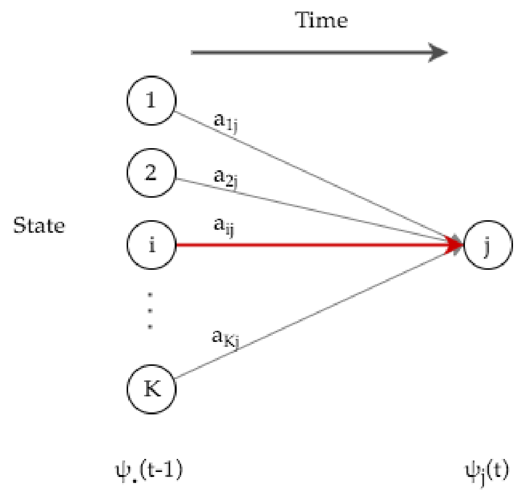

Figure 1-4: Viterbi algorithm chooses the state transition yielding the maximum probability, also known as the most probable path [17].

probability to be the final winner[4] is only a few. Of cause, a lot of low probability candidates can be deactivated during decoding search by various heuristic pruning methods in order to reduce the number of possible search paths and thus to speed up the search process.

When the search traverses through the state lattice, the vocabulary in the dictionary determines whether a state sequence up to the current time step would make up any possible word sequences for carrying out (1.4) to find the most probable sentence. In practice, a *language model weight* "α_{LM}" and a *word insertion penalty* "WP" are often included in the implementation of (1.4) to adjust the significance of language model and to penalize long-time span words (if the *word insertion penalty* is negative it encourages long-time span words instead) respectively:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \{\log P(X|W) + \alpha_{LM} \log P(W) - M * WP\} \tag{1.22}$$

where M is the number of words in the word sequence W.

When dealing with large vocabulary continuous speech recognition (LVCSR), a generic Viterbi decoding search mentioned above is usually not sufficient. Hence, a more complex variant of Viterbi algorithm involving top-down time-synchronous

[4]the final recognition result

beam searching is often used [2]. Typically, a more advanced decoding search such as Weighted finite-state transducers (WFST) decoder is preferable for LVCSR yet it exceeds the scope of this work and the technical details of method are skipped. Nevertheless, a good description and analysis of WFST can be found in [18].

Chapter 2

Preparing for Khmer ASR

Automatic speech recognition technology has undergone intensive research and development for many decades, and yet the supported languages of speech recognizers in today's market remain only a few. For the languages used in developing countries, there is little or no language resources (annotated speech, dictionary, and text data) readily available for system training.

In regard to Khmer speech recognition, the authors in [7] recorded broadcast news in Khmer language from several radio stations and trained a grapheme based acoustic model for a broadcast news transcription system for Khmer language. In the experiments of the thesis, we instead used the dataset in [6], which is more suitable for word recognition task, and we manually constructed a phoneme based dictionary for labeling the pronunciation of Khmer words (194 words) speech data prior to acoustic model training.

In this chapter, we first present the dataset used. Next, relevant data preparation steps such as data preprocessing and choosing the test set are described. Lastly, the process of building the pronunciation dictionary for Khmer ASR is discussed.

2.1 Dataset

"Khmer keywords" database [6], created by the Institute of Technology of Cambodia, was intended to be used in an Interactive Voice Response (IVR) telephone system.

It initially consists of 194 commonly used vocabulary words, i.e, province names, numerical counts, month names, weekdays, yes/no answers, common disease names and essential daily commodities. The speakers recorded were 15 university students (9 males and 6 females) aged between 19 and 23, reading words in the vocabulary with a short silence between each pair of words using *Standard Khmer*, which is the official spoken dialect taught in Cambodian schools. Recording condition was chosen to be in a low to semi-noisy environment and was done via mobile phones. The sampling rate of 8kHz were used to produce audio files in WAVE format [1]. These setups were to mimic the conditions of telephone speech during IVR transactions.

The original dataset contains a total of 15 long wave files (one for each speaker) and their transcriptions. Each audio file is approximately 11 minutes and 30 seconds in duration and contains about 194 uttered words. Since the objective of an isolated word recognition task is to identify which single word was being spoken, it is desirable to put each word in a separate file for both system training and testing phases. This requires the original audio wave files to be further processed.

2.1.1 Data Preprocessing

Each wave file in the dataset, normally consisting of 194 to 196 uttered words [2], is to be segmented into smaller files, each of which contains a single word. If word boundaries in time are known, an audio file can be separated into those of words automatically. When word boundaries are unknown, the short silence between two words can be used as a word boundary and finding these silence points in the whole wave file can also be considered as a simplified problem of Voice activity detection(VAD).

[21] suggested the design of a typical VAD procedure comprising of the following three stages: *1)* feature extraction stage, *2)* detection stage and *3)* decision smoothing.

In the feature extraction stage in Figure 2-2, we compute the energy profile generated from a non-overlapping moving window of 10 ms on a speech signal as

[1]https://en.wikipedia.org/wiki/WAV
[2]Some words were read more than one time.

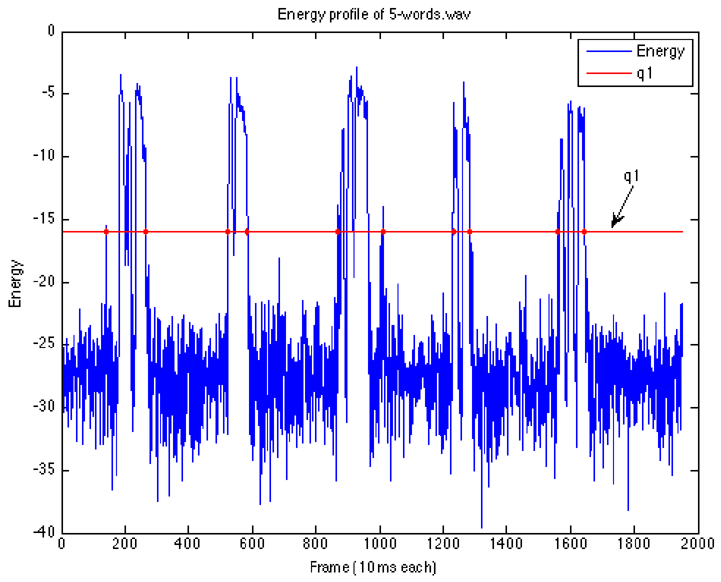

Figure 2-1: Energy profiling in VAD

$$E = 10 \log_{10} \sum_{n=1}^{80} s_n^2 \tag{2.1}$$

where E is the log energy of a single frame of the speech signal, and s_n's are the samples within the window. Since the audio files have the sampling rate of 8kHz, the number of samples in each 10 ms window is equal to 80.

A simplified decision rule based on tunable *energy* and *word spanning* thresholds are then used to determine speech versus non-speech regions from the energy features. Figure 2-1 shows a log energy profile on an audio file which contains five words. The high energy regions, having values above the energy threshold ($q1$) are considered as the word regions whereas the regions with low energy (below the threshold) are regarded as silence. The high energy regions are scanned to see if the distance between two adjacent high energy frames exceeds the word spanning threshold ($q2$), and if so, then those two frames are considered as belonging to two different word regions. For example, we chose energy threshold ($q1$) to be (-16) and the word spanning threshold ($q2$) to be 100 frames in Figure 2-1. The first two red dots on the $q1$ line are viewed as in the same word region because the distances between their adjacent high energy

```
Algorithm 1: VAD algorithm
  Input   : $s$ : Audio file containing multiple words
  Input   : $q1$: Energy threshold
  Input   : $q2$: Word spanning threshold
  Input   : $duration$: Duration threshold
  Output: $words \leftarrow []$: Word boundaries
  $energy \leftarrow$ ExtractFeature($s$);
  $words \leftarrow$ EnergyThresholding($energy$, $q1$, $q2$);
  for $i \leftarrow 1$ to LENGTH($words$) do
      if $words[i].frames < duration$ then
          REMOVE($words[i]$);
      end
      SilPadding($words[i]$);
  end
  return $words$;
```

```
Function EnergyThresholding($energy$, $q1$, $q2$)
  Output: $words \leftarrow []$: words boundaries
  $highEnergy \leftarrow []$;
  $i \leftarrow 1$;
  for $t \leftarrow 1$ to LENGTH($energy$) do
      if $energy[t] > q1$ then
          $highEnergy[i] \leftarrow t$;
          $i \leftarrow i + 1$;
      end
  end
  $j \leftarrow 1$;
  $words[j].start \leftarrow highEnergy[1]$;
  for $i \leftarrow 2$ to LENGTH($highEnergy$) do
      if $highEnergy[i-1] + q2 < highEnergy[i]$ then
          $words[j].end \leftarrow highEnergy[i-1]$;
          $j \leftarrow j + 1$;
          $words[j].start \leftarrow highEnergy[i]$;
      end
  end
  $words[j].end \leftarrow highEnergy[i]$;
  return $words$;
```

Figure 2-2: Voice activity detection algorithm

frame pairs are within the range of 100. On the contrary, the distance between the second and the third red dots is larger than the word spanning threshold yet there is no high energy frame between them; therefore they are in two different word regions.

Decision smoothing can also be applied to provide further fine-tunings on the word boundaries produced from detection stage. In our implementation, a duration threshold of 100 ms is used to filter out abnormal high-energy regions such as impulse noise which only lasts for a few frames (less than 100 ms). In addition, silence frame padding is also used to extend the word boundaries to allow a more robust detection of un-voiced sound in Khmer language, e.g, រ, ស and ហ letters which have characteristics similar to that of silence sound.

We applied the VAD algorithm on the entire dataset using the same energy, word spanning and duration thresholds and observed that the highest segmentation error rate was $6/194 \approx 3.1\%$ when the most noisy file was excluded[3].

The segmentation error is defined as:

$$Error = \sum_{f \,\in\, \text{Segmented files}} err(f) \tag{2.2a}$$

$$err(f) = \begin{cases} 0 & \text{if WordCount}(f) = 1 \\ 1 & \text{if WordCount}(f) = 0 \\ \text{WordCount}(f) & \text{otherwise} \end{cases} \tag{2.2b}$$

The segmentation error rate is the ratio of the total segmentation errors divided by the total number of words in the original audio file.

All segmentation errors were manually corrected and we ended up with a total of 2711 audio files (one per word) from 14 speakers, 8 of which are males and the others 6 are females as shown in Table 2.1.

Speaker	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Gender	f	f	m	m	f	m	f	m	m	m	f	f	m	m
# of files	194	191	194	194	194	193	194	194	193	194	195	195	194	192

Table 2.1: Word dataset

2.1.2 Choosing the Test set

Among all the utterances from the 14 speakers, we selected those of 4 speakers (2 males and 2 females), i.e., speaker 1, 4, 8 and 11, to be the test set for our model evaluation. There is a total of 777 utterances in the test set and the remaining 1934

[3]It is admissible to exclude the single most noisy file in which the microphone was placed too far away from the speaker. The inclusion of this file in system training will negatively affect acoustic model as well.

files are used as the training set as shown in Table 2.2. The ratio of the sizes of the test set to the training set is about 2 to 5, which is considered as a decent partition to make a stable test set for recognition performance assessment.

The four test speakers are chosen based on our perception and linguistic knowledge as a native speaker of Khmer language to favor average quality of speech measured by speaker's vocal tract, accent, and speaking rate.

	Speaker	**Gender**	**# utterances**
Training set	2, 3, 5, 6, 7, 9, 10, 12, 13 and 14	6 males, 4 females	1934
Test set	1, 4, 8 and 11	2 males, 2 females	777

Table 2.2: Training and test sets

2.2 Pronunciation modeling

The transcription of the dataset is available in Khmer unicode format. However, for subword model based speech recognition, the pronunciation of words is also required to represent words in term of ARPAbet characters each of which denotes a distinct sound in the target language.

A phonetic analysis of Khmer language had been studied in [22] and the text-to-sound mapping tables illustrated in the study is useful for constructing Khmer phonetic inventory for speech recognition. In Khmer pronunciation, consonants are divided into two groups: ɑ-group which inherits /ɑ/ sound and ɔ-group having /ɔ/ sound (The list of consonants' sounds can be found in Table A.1). A consonant can either be followed by another consonant in the form of a subscript to make a consonant cluster, or by a vowel[4]. Normally, the vowel will take /ɑ/ sound if the immediate consonant it follows is in ɑ-group, and it will take /ɔ/ sound otherwise.

The list of consonant and dependent vowel sound mappings can be found in Tables A.1 and A.2, respectively. The sound mappings relying only on the sound group of preceding consonant allow the pronunciation dictionary to be constructed simply by substituting Khmer unicode characters with ARPAbet symbols via table lookup.

[4]Here we refer to a dependent vowel only as Khmer also has independent vowels which do not follow a consonant.

One problem with this approach is that a subtle pronunciation of a word containing consonant clusters or diacritics might not be captured due to linguistic complexity of Khmer language and such a case has been handled manually.

Chapter 3

GMM-HMM Acoustic Modeling

Acoustic modeling is an essential component in building a speech recognizer. One common modeling approach is to use a Gaussian mixture model based hidden Markov models, or GMM-HMM for fitting the MFCC feature sequences of speech data. The success and popularity of GMM-HMM in speech recognition are mainly due to the effectiveness of GMM in modeling the distribution of spectral vectors, the use of HMM to represent temporal speech pattern, and more importantly the highly efficient Baum-Welch [23] re-estimation in which the parameters of the HMMs are trained to maximize the likelihood of the training data.

In this chapter, we first describe the classical Baum-Welch parameter re-estimation, which is one of the most important algorithm in training hidden Markov model parameters, we then describe a flat start procedure for model initialization and the main uses of forced alignment. We also cover context-dependent models, state tying and mixture splitting. Finally, experimental results on GMM-HMMs are presented.

3.1 Parameter Re-Estimation

The goal of parameter re-estimation of a model is to iteratively adjust the set of parameters $\lambda = (\pi, A, B)$ of the HMM to maximize the likelihood of observation sequences given the model. Because the parameter estimates of the model cannot be obtained explicitly, Baum-Welch algorithm is used instead to iteratively maximize

Baum's auxiliary function $Q(\lambda, \bar{\lambda})$, which has been proven to increase the likelihood of the training data as described in [11]:

$$Q(\lambda, \bar{\lambda}) = \sum_{S} P(S|X, \lambda) \log[P(X, S|\bar{\lambda})] \tag{3.1}$$

where $\bar{\lambda} = (\bar{\pi}, \bar{A}, \bar{B})$ is model parameters to be re-estimated.

The Baum-Welch re-estimation procedure defines $\xi_t(i, j)$ as the probability of being in state i at time t and being in state j at time $t + 1$ given the observation sequence and the model:

$$\xi_t(i, j) = P(s(t) = i, s(t + 1) = j|X, \lambda) \tag{3.2}$$

By using a forward variable $\alpha_t(i)$, i.e., the probability of the partial observation sequence $x_1, \dots, x_t$ and state i at time t, and a backward variable $\beta_t(i)$, i.e., the conditional probability of the partial observation sequence $x_{t+1}, \dots, x_T$ given that the state at time t equals to i:

$$\alpha_t(i) = P(x_1, \dots, x_t, s(t) = i) \tag{3.3a}$$

$$\beta_t(i) = P(x_{t+1}, \dots, x_T|s(t) = i) \tag{3.3b}$$

we can re-write (3.2) in the following form:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(x_{t+1}) \beta_{t+1}(j)}{P(X|\lambda)} \tag{3.4}$$

Defining $\gamma_t(i) = P(s(t) = i|X, \lambda)$, we can relate $\gamma_t(i)$ to $\xi_t(i, j)$ by summing over j:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j) \tag{3.5}$$

where N is the number of distinct states. In addition, interesting quantities can be obtained by summing $\gamma_t(i)$ and $\xi_t(i, j)$ over time:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } i. \tag{3.6}$$

and

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from state } i \text{ to state } j. \tag{3.7}$$

Using these quantities, the re-estimation formula of an HMM can then be expressed as the following:

$$\bar{\pi}_i = \text{expected frequency (number of times) in state } i \text{ at time } 1 \tag{3.8a}$$

$$= \gamma_1(i) \tag{3.8b}$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \tag{3.9a}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{3.9b}$$

and the GMM parameter updates are:

$$\hat{\mu}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{tm}(j) x_t}{\sum_{t=1}^{T} \gamma_{tm}(j)} \tag{3.10}$$

$$\hat{\Sigma}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{tm}(j)(x_t - \hat{\mu}_{jm})(x_t - \hat{\mu}_{jm})^T}{\sum_{t=1}^{T} \gamma_{tm}(j)} \tag{3.11}$$

$$\hat{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_{tm}(j)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_{tk}(j)} \tag{3.12}$$

where $\gamma_{tm}(j)$ is the probability of being in state j at time t with the m-th mixture component accounting for the output observation x_t:

$$\gamma_{tm}(j) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(j) a_{ij} c_{jm} b_{jm}(x_t) \beta_t(j)}{P(X|\lambda)} \tag{3.13}$$

The above Baum-Welch re-estimation procedure is also applicable for a word-level HMM since it is merely a concatenation of a sequence of its phone HMMs. Parameter re-estimation of a set of HMMs from multiple speech utterances can also be achieved with minor modification on the re-estimation formula as described in [8, 24].

3.1.1 Flat start initialization

Prior to parameter re-estimation, an initial set of phone HMMs has to be established. To initialize HMMs, the so-called *flat start* procedure is often used since it does not require phonetic level transcription to be readily available, which is also our case.

A flat start training initializes mean and variance of each phone HMM to be the global mean and variances of the whole training utterances. The transition probability of the initial models can be any fixed structure of transition matrix constrained to (1.6). The associated word transcription of an utterance is first converted into a sequence of phones by a pronunciation dictionary. A composite HMM is then constructed according to the pronunciation order of the phone labels and during the first cycle of parameter re-estimation, each training utterance is uniformly segmented based on the number of phone states in the utterance [2].

3.1.2 Forced alignment

The phone models previously trained with the Baum-Welch algorithm can be used to realign the training transcriptions to include time boundaries. This process is often referred to as *forced alignment.* Forced alignment uses Viterbi algorithm to find the best matching phone sequence and boundaries according to the acoustic evidences embedded in an utterance. It is particularly useful for words with multiple pronunciations since the acoustic realizations of each phone will determine the actual pronunciation of the words. Table 3.1 shows the words with multiple pronunciation in the task vocabulary.

Khmer word	IPA transcription	English translation
០	sɔːon ʒeː ro	Zero
៧	pram piː pram pil	Seven
១៧	ɗɑɓ pram piː ɗɑɓ pram pil	Seventeen

Table 3.1: Khmer words with more than one pronunciations in the vocabulary

In this work, however we treat each pronunciation as a separate word since they appear independently in the training data and thus forced alignment is not being used for this scenario.

Another use case of forced alignment is to produce state level transcription which is necessary for DNN-HMM training described in Chapter 4.

3.2 Context-Dependent Models

The use of triphones as HMM modeling units is usually desirable in speech recognition since context-dependent triphones can better capture the co-articulation phenomenon in continuous speech, as described in Section 1.5.2. As proposed in [13], a triphone HMM set can be initially constructed by cloning the parameters of the corresponding monophone HMMs.

3.2.1 Tied-state Triphone HMMs

There are a total of 60 monophones in our task. Normally, the number of triphones is in the cubic order of that of the monophones, which produces more than 200k possible triphones. In this scenario, phonetic decision tree was used to tie similar acoustic states of the triphones of each phone state in order to ensure a robust estimation of all state distributions.

The decision tree for triphone state clustering is simply a binary tree with a yes/no phonological question at each node asking about the left and right contexts of a triphone. The tree is built up iteratively and the question at each node is chosen to maximize the likelihood gain due to the node split, where at each node, the training data is modeled by a single Gaussian distribution[1]. When the likelihood gain obtained from a node split in the decision tree falls below a threshold, the node split stops and the nodes that do not have enough data get merged with their neighbors. Finally, the triphone state clusters at each leaf node becomes the tied-states of triphone HMMs.

To use PDT based state clustering for Khmer, phonological questions for Khmer language need to be designed from scratch. Yet, a question set (**QS**) is available for clustering the English phonemes that can be adapted to create a new **QS** by performing a manual sound mapping from English to Khmer phones and then replacing English phonemes with that of Khmer. The procedure for converting English **QS** to Khmer **QS** consists of the following four steps:

1. Create a map of phonetically similar English and Khmer phones [2].
2. Remove the phones that do not exist in the above map from the **QS**.
3. Remove any question in the **QS** if it contains no phone after step 2.
4. In the **QS**, replace English phones with Khmer phones according to the mapping created in step 1.

Table 3.2 shows a portion of the English to Khmer sound mappings used for creating Khmer **QS**. Each row in the table contains an English sound, the matching Khmer sounds and the corresponding characters producing those sounds. The complete set of the sound mappings can be found in Tables A.3 and A.4.

3.3 Mixture Models

To use GMM for those tied-state triphone models, conversion from single Gaussian to multiple mixture components HMMs is required, and the process called *mixture splitting* [2] is used to accomplished that.

[1]HTK supports decision tree-based clustering for single Gaussian only

[2]One English phone can have more than one similar Khmer phones since there are two groups of consonants in Khmer that take the same basic sound (Table A.1).

[3]The complete English to Khmer sound mappings table can be found in Tables A.3 and A.4.

English sound	Khmer sounds	Khmer characters
/P/	/P/, /PH/	/ប៉, ព/, /ផ, ភ/
/G/	/K/	/ក, គ/
/F/	N/A	N/A
/NG/	/NG/, /GN/	/ង៉, ង/, /ញ៉, ញ/
/EH/	/EH/, /EO/, /OEH/	/ ិ /, / ឹ /, /េះ /
...	...	...

Table 3.2: English to Khmer sound mappings in ARPAbet format[3].

Mixture splitting starts by selecting the mixture component with the 'heaviest' weight for splitting into two mixture components each having half the parent mixture weight, and relative to the parent's mean vector, the mean vectors of the child mixture components are then increase by 0.2 standard deviation for one and decrease by the same amount for the other. After each split, parameter re-estimation using Baum-Welch algorithm is performed. The mixture splitting process repeats until the desired number of mixture components is reached.

3.4 Experiments and results

In our experiments, we first built monophone GMM-HMM and evaluated the word accuracy performance on the test set that consists of 777 utterances with one word per utterance. To form a feature vector of 39 dimensions, 13-dimension static MFCC augmented with its first and second order time derivatives were used. Table 3.3 shows the word accuracy performance of monophone and 468-tied-state triphone models with different number of mixture components. The monophone acoustic models trained with single Gaussian performed poorly at the accuracy of 68.85%. However, by just using two mixture components in a GMM, the accuracy jumped straight up to 80.31% which is an absolute improvement of 11.46%. The performance of monophone GMM-HMM peaked at 90.86% required 14 mixture components as shown in Table 3.3.

# mixtures	1	2	4	6	8	10	12	14	16
Monophone	68.86	80.31	87.39	89.58	90.48	89.96	90.35	**90.86**	90.22
Triphone(468 tied states)	90.86	94.34	96.53	**97.17**	96.65	95.75	94.72	94.34	94.34

Table 3.3: Performance of GMM-HMM acoustic models in term of word accuracy (%)

For the case of 468-tied-state triphone HMMs produced from the PDT with the Khmer question set, we observed that the word accuracy of 90.86% was achieved by the triphone models trained with just a single Gaussian, which puts itself at the same position of the best monophone models. The best result was 97.17% word accuracy when using context-dependent 468-tied-state triphone GMM-HMMs and the number of mixture components was 6.

In PDT state clustering, we fixed state occupancy threshold to 30 and used various likelihood gain thresholds denoted as **TB** to create different numbers of tied-states triphone models. The performances of the tied-state GMM-HMMs are presented in Figure 3-1.

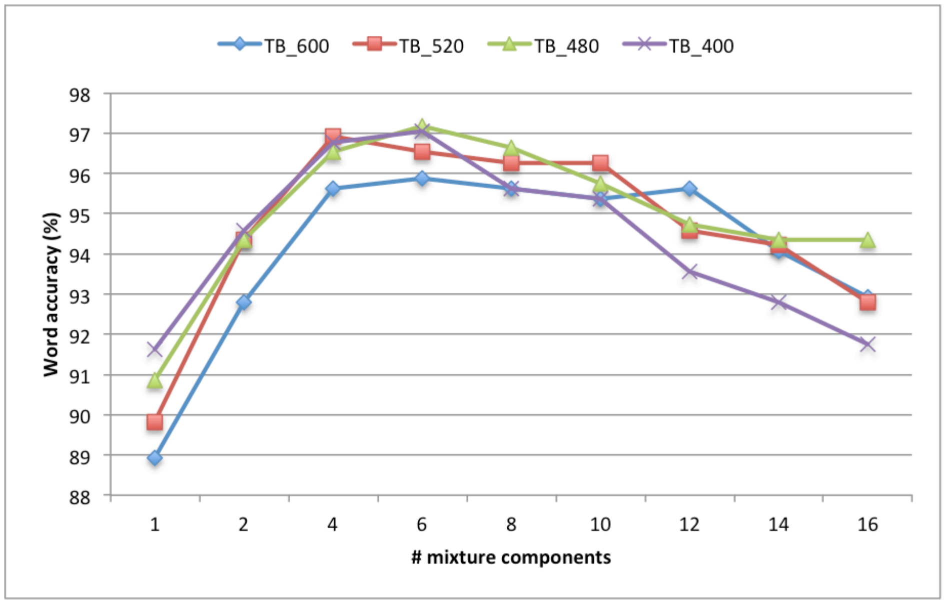

Figure 3-1: Word accuracies of GMM-HMMs produced from decision tree with likelihood gain thresholds (TB) of 600, 520, 480 and 400.

There are performance gains in all case of **TB**s when more than 1 mixture component are used. TB_480 (the triphone models produced by setting likelihood gain threshold to 480) using 6 mixture components is the top performer achieving 97.17% word accuracy whereas TB_520 having the same number of mixture components fails to improve. Onward, the improvement diminishes for all triphone models that use 8 or more mixture components.

The **TB**s of {600, 520, 480 and 400} produced {331, 395, 428 and 542} physical triphone HMMs and {393, 437, 468 and 539} tied-states, respectively. The number of mixture components that work best for each case, its corresponding word accuracy and the number of parameters of the models are shown in Table 3.4. It can be seen that 468-tied-state triphone GMM-HMM with 6 mixture components having the best performance requires 221,832 parameters.

TB	**# physical models**	**# tied-states**	**# mixtures**	**Word accuracy**	**# parameters**
600	331	393	6	95.88	186,282
520	395	437	4	96.91	138,092
480	428	468	6	**97.17**	221,832
400	542	539	6	97.04	255,486

Table 3.4: The best results of tied-state triphone GMM-HMMs for different **TB**s

Chapter 4

DNN-HMM Acoustic Modeling

The idea of using artificial neural networks (ANN) in combination with hidden Markov models for speech recognition had been proposed since early 90's [25]. In ANN-HMM hybrid model, a multilayer perceptron or MLP, which estimates the posterior probability $P(s|x_t)$ of each phone state given an acoustic observation at frame t, is used in place of Gaussian mixture models. The state posterior probability is then converted to the emission likelihood $p(x_t|s)$ via Bayes rule:

$$p(x_t|s) = \frac{P(s|x_t)p(x_t)}{P(s)} \tag{4.1}$$

where $p(x_t)$, the evidence of acoustic observation, is independent of HMM state and can be ignored, and $P(s)$ is the state prior probability that can be estimated from the training data by counting.

A deep neural network (DNN) is simply a MLP with many hidden layers. The recently proposed DNN-HMM differs from ANN-HMM in the 90's in that: *1)* ANN[1] is replaced by DNN which uses a pre-training procedure to make training of many parameters more reliable, *2)* in speech recognition, the output layer of DNN is used to model tied-state triphones directly [26, 27].

[1]Refer to neural network with single layer of hidden units.

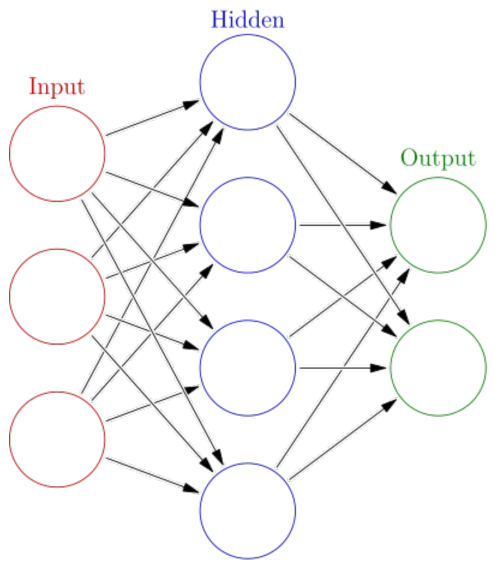

Figure 4-1: 3 layer neural network[2].

4.1 Neural Network Architecture

A neural network generally consists of a layered structure of “neurons” each of which is connected with all neurons from its lower layer as seen in Figure 4-1. For a neural network with $L+1$ layers, we denote the input layer as layer 0 and the output layer as layer L. The value of a neuron in a layer l is defined as

$$\mathbf{y}^{(l)} = f(\mathbf{z}^{(l)}) = f\left(\mathbf{W}^{(l)}\mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{4.2}$$

where $\mathbf{z}^{(l)}$, $\mathbf{y}^{(l)}$, $\mathbf{W}^{(l)}$, and $\mathbf{b}^{(l)}$ are the l layer's excitation vector, activation vector, weight matrix and bias vector, respectively. The value of the input layer, which is the observation, is represented by $\mathbf{y}^{(0)} = x$, and $f(\cdot)$ is the activation function that maps $\mathbf{z}^{(l)}$ element-wise into $\mathbf{y}^{(l)}$.

Usually, the activation function takes the form of a sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{4.3}$$

[2]https://en.wikipedia.org/wiki/File:Colored_neural_network.svg

where the output value of $\sigma(z)$ is between 0 and 1 inclusively. Yet, rectified linear unit (ReLU) function is often used in hidden layer since it can lead to faster convergence[2].

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

In DNN-HMM, the activation function of the output layer of DNN is typically a softmax function. Given the model parameter $\{\mathbf{W}, \mathbf{b}\} = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)} \mid 0 < l < L\}$, the posterior probability of the DNN at the i-th element is

$$y_i^L = \text{softmax}(\mathbf{z}^L) = \frac{e^{z_i^{(L)}}}{\sum_{j=1}^{C} e^{z_j^{(L)}}} \tag{4.5}$$

where C is the number of HMM states and i is the i-th element of the DNN's output layer. The operation of computing the output of a neural network in this manner is known as *feed-forward* propagation.

4.2 DNN Training

We have so far assumed that the model parameter $\{\mathbf{W}, \mathbf{b}\}$ are known. However, they have to be estimated from the training data based on some training criteria that would lead to minimizing a cost function. Cross entropy (CE) loss function is commonly used when softmax is used as the output activation function.

$$J_{CE}(\mathbf{W}, \mathbf{b}; \mathbf{x}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} \hat{y}_i \log y_i^{(L)} \tag{4.6}$$

where $\mathbf{x}$ is the observation vector, $\hat{\mathbf{y}}$ is the training label in one-hot representation and $\hat{y}_i$ is the i-th element of $\hat{\mathbf{y}}$. Then the DNN training can be done via error backpropagation (BP) algorithm which uses the first-order gradient information for updating the model parameters:

$$\mathbf{W}^{(l)}(t+1) \leftarrow \mathbf{W}^{(l)}(t) - \eta \Delta \mathbf{W}^{(l)}(t) \tag{4.7}$$

and

$$\mathbf{b}^{(l)}(t+1) \leftarrow \mathbf{b}^{(l)}(t) - \eta \Delta \mathbf{b}^{(l)}(t) \tag{4.8}$$

where t is the update iteration and η is the learning rate. Let $\nabla_{\mathbf{a}} J$ be the gradient of J with respect to $\mathbf{a}$, and let

$$\Delta \mathbf{W}^{(l)}(t) = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{\mathbf{W}^{(l)}(t)} J(\mathbf{W}, \mathbf{b}; \mathbf{x}^{(m)}, \hat{\mathbf{y}}^{(m)}) \tag{4.9}$$

and

$$\Delta \mathbf{b}^{(l)}(t) = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{\mathbf{b}^{(l)}(t)} J(\mathbf{W}, \mathbf{b}; \mathbf{x}^{(m)}, \hat{\mathbf{y}}^{(m)}) \tag{4.10}$$

be the average weight matrix gradient and the average bias vector gradient at iteration t and estimated from the training batch of size M_b. Usually the training batch with M_b samples is called a *mini-batch* and the gradient applied on a sequence of mini-batch is often referred as *Stochastic gradient descent* (SGD). With some derivations using the chain rule, the gradients of the weight matrix and bias vector are:

$$\nabla_{\mathbf{W}^{(l)}(t)} J(\mathbf{W}, \mathbf{b}; \mathbf{x}, \hat{\mathbf{y}}) = [f'(\mathbf{z}^{(l)}(t)) \bullet \mathbf{e}^{(l)}(t)](\mathbf{y}^{(l-1)}(t))^T \tag{4.11}$$

and

$$\nabla_{\mathbf{b}^{(l)}(t)} J(\mathbf{W}, \mathbf{b}; \mathbf{x}, \hat{\mathbf{y}}) = f'(\mathbf{z}^{(l)}(t)) \bullet \mathbf{e}^{(l)}(t) \tag{4.12}$$

where $\mathbf{e}^{(l)}(t)$ is the error signal at layer l and the iteration t, $\bullet$ is the element-wise product operator and $f'(\mathbf{z}^{(l)}(t))$ is the element-wise derivative of the activation function of $\mathbf{z}^{(l)}(t)$. The formula for $\mathbf{e}^{(l)}(t)$ and the detailed derivations can be found in[1].

4.3 Gradient Refinement

The DNN gradient computed using SGD can sometimes be either too small, leading to gradient vanishing, or too big, known as gradient "explosion". Several techniques have been successfully used to alleviate these issues [2]. *Gradient clipping* is used to prevent gradient explosion by clipping the gradient once its absolute value exceeds a given positive threshold f:

$$g_w^{\text{clip}}(t) = \begin{cases} f & \text{if } g_w(t) > f \\ -f & \text{if } g_w(t) < -f \\ g_w(t) & \text{otherwise} \end{cases} \tag{4.13}$$

where $g_w(t)$ is the gradient of a single weight w at the iteration t. *Weight decay* is another widely used regularization that penalizes the objective function J_{CE} by adding a scaled l_2 norm. The gradient with weight decay becomes:

$$g_w^{\text{decay}}(t) = g_w(t) + \lambda w(t) \tag{4.14}$$

where λ is a scaling factor of the regularization term. Another important technique to speed up the convergence is the use of *momentum* with a momentum factor α:

$$g_w^{\text{momen.}}(t) = \alpha g_w(t-1) - \eta \frac{\partial J}{\partial w}(t) \tag{4.15}$$

Max-norm can also be used to set the maximum l_2-norm constraint or bound to the weight vector $\mathbf{w}$ and scale the weight if the l_2-norm exceeds the bound c:

$$w^{\text{max-norm}}(t) = \begin{cases} w(t) \cdot \dfrac{c}{\|\mathbf{w}(t)\|_2} & \text{if } \|\mathbf{w}(t)\|_2 > c \\ w(t) & \text{otherwise} \end{cases} \tag{4.16}$$

4.4 Dropout

A deep neural network with multiple non-linear hidden layers can learn a very complicated function. However, with limited training data, this instead leads to *overfitting* since standard backpropagation learning builds up co-adaptations among hidden units too much, which makes it work well with the training data but do not generalize to unseen data [28]. *Dropout* is a powerful technique that can prevent overfitting in large

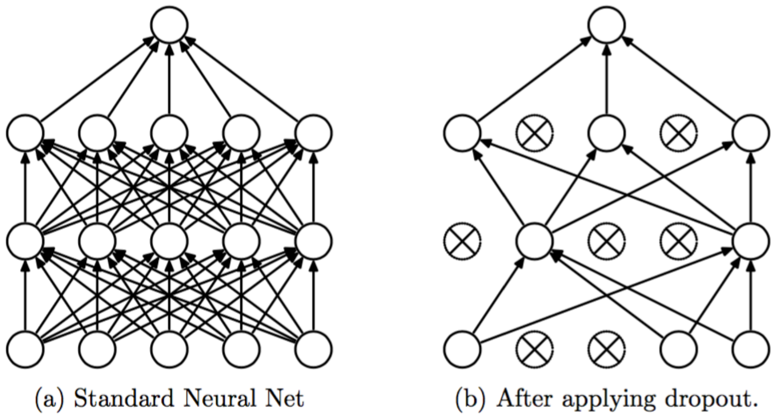

Figure 4-2: Dropout applied on the standard neural network in (a) results in a dropout net in (b). [28]

neural networks by randomly omitting a certain percentage of hidden units from the networks as shown in Figure 4-2.

With a dropout probability r during training, the feed-forward operation in (4.2) becomes (4.17c)

$$\mathbf{m} \sim \text{Bernoulli}(1-r) \tag{4.17a}$$

$$\tilde{\mathbf{y}}^{(l-1)} = \mathbf{m} * \mathbf{y}^{(l-1)} \tag{4.17b}$$

$$\mathbf{y}^{(l)} = f(\mathbf{W}^{(l)}\tilde{\mathbf{y}}^{(l-1)} + \mathbf{b}^{(l)}) \tag{4.17c}$$

where $\mathbf{m}$ is a random binary mask with each element following Bernoulli and with the probability of $(1-r)$ for having a value 1, $*$ is the element-wise product operator and $\tilde{\mathbf{y}}$ can be regarded as a thinned network sampled from $\mathbf{y}$ according to $\mathbf{m}$.

4.5 Cross-lingual Transfer Learning

The demand for a large amount of labeled speech data to train a DNN-based acoustic model is often an unavoidable bottleneck for resource-scarce languages. *Transfer learning* has emerged as a learning framework intended to address this problem by efficiently retaining and leveraging the knowledge learned from one or more similar tasks or domains [1]. The idea of cross-lingual transfer learning is to make use of auxiliary data from a different language for compensating the data sparsity problem in the target language of the recognizer.

In a DNN, the combination of all hidden layers can be considered as a feature learning module and only the output layer is in charge of the classification task. Therefore, this feature learning module can be shared across different languages or transfer to a new language by extracting the hidden layers of the DNN model trained with the auxiliary data and stack a new softmax layer corresponding to the tied-state of the target language [30].

4.6 Experiments and Results

In our experiments, we selected at random one speaker from the training set to be used as the validation data for DNNs training. MLPs with 1 to 8 hidden layers were constructed using 3 different numbers of hidden neurons. i.e., 512, 1024 and 2048 resulting in 24 different networks to model 468 tied-states of the triphones directly. The activation function for all hidden layers was ReLU and softmax was used as the output layer's activation. We used 15 frames (7-1-7) of MFCCs, which were normalized to have zero mean and unit variance, as the input features of the DNNs. A learning rate began at 0.0001 and the Newbob scheduler[3] would decay the learning rate by a half at each epoch if a decrease in validation set's performance was observed. The minibatch was 200 samples trained for at least 24 epoches and the max-norm upper bound was set to 1. We performed discriminative layer-wise pretraining with a single epoch SGD and 0.9 momentum parameter α to initialize the network weight parameters. In the fine-tuning stage, the momentum parameter α was then increase to 0.99. These hyperparameters were primarily tuned for single hidden layer network with 512 hidden units then applied to all other networks with the same number of neurons per hidden layer. We believe that a better performance might be obtained by a more exhaustive hyperparameter searching strategy. Table 4.1 shows the word

[3]Newbob scheduler details can be found in [2]

accuracy performance on the test set for DNNs with 512 hidden units and the best result obtained is 90.99% accuracy for a 4-hidden-layer network.

# hidden layers	1	2	3	4	5	6	7	8
Word accuracy	87.90	90.22	90.86	**90.99**	90.60	89.06	88.55	87.13

Table 4.1: Word accuracy (%) of eight 512-hidden-unit DNNs.

4.6.1 Applying Dropout

The best network[4] from the above was then chosen to be re-trained with the dropout method in order to improve its performance. We observed that when the input and the hidden dropout rates were 0.5 and 0.02, respectively, the word accuracy was 92.54% which is an absolute improvement of 1.55%. The input dropout rate of 0.5 and the hidden dropout rate of 0.02 combined with the already tuned hyperparameters were then used to re-train all 24 DNNs. The results of all DNNs trained with the dropout method are shown in Figure 4-3.

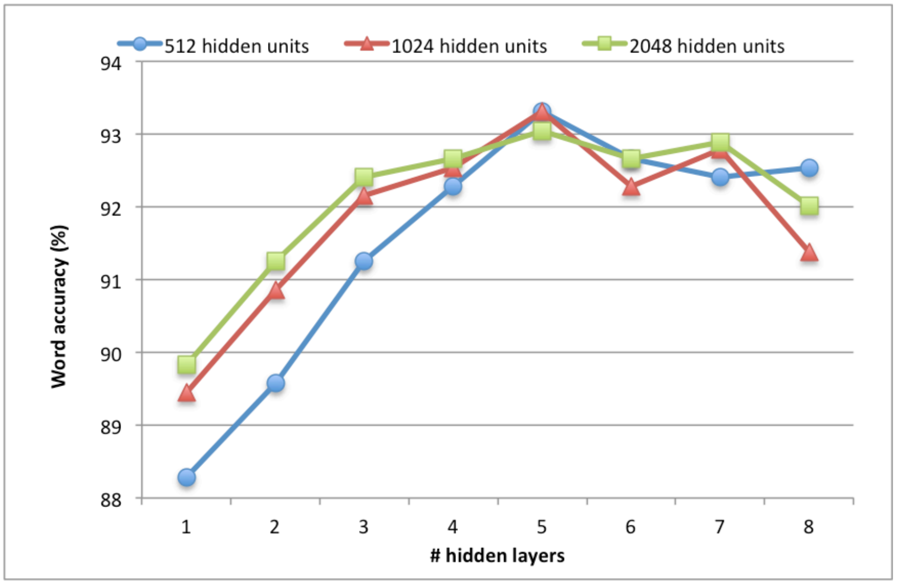

Figure 4-3: Word accuracies of all DNNs trained with dropout.

We can see that all 5-hidden-layer networks worked well in general and they perform best at 93.31% accuracy when having 512 or 1024 hidden nodes. Figure 4-4

[4]The network that had the highest word accuracy

depicts the validation set cross entropy for all 5-hidden-layer networks, and the 512-hidden-unit network has the lowest cross entropy score starting from epoch 19.

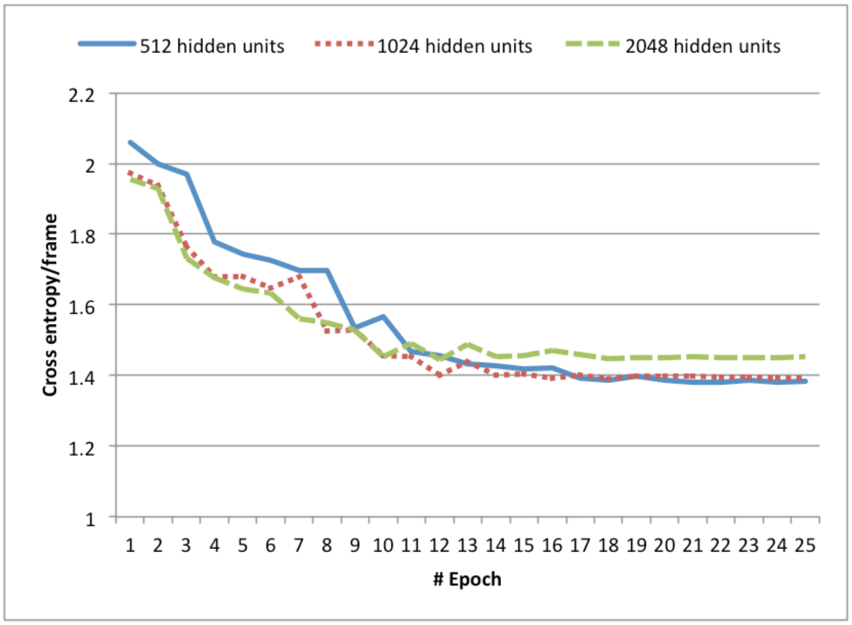

Figure 4-4: Cross entropy values on a validation set for the 5-hidden-layer networks with different numbers of hidden units for 25 epoches.

4.6.2 Cross-lingual Model Transfer from English

We further investigated using cross-lingual transfer learning from English data by training a 5-hidden-layer DNN on TIMIT continuous speech corpus [29]. The network was trained with the same hyperparameters from the above settings except no dropout had been used. We then extracted the weights in the hidden layers of this network to use in the DNNs of Khmer task for fine-tuning. Table 4.2 shows that the performances of the target DNNs were slightly decreased when the networks were initialized with the extracted hidden layers from the DNN trained on English data.

	# hidden units		
	512	**1024**	**2048**
no__transfer	93.31	93.31	93.05
TIMIT__transfer	92.60	93.18	92.15
Performance lose	0.71	0.13	0.9

Table 4.2: Comparison of various 5-hidden-layer DNNs with and without weight transfer from English task.

Chapter 5

Discussion

The resurgence of artificial neural networks, often referred to as *Deep Learning*, has attracted a great deal of attention and interest in the field of machine learning and speech recognition research. Speech recognizers using deep learning algorithms reported start-of-the-art performance on many large vocabulary continuous speech recognition tasks [14]. These success stories among many other show cases of deep learning have made *Deep Learning* a buzzword over the years. Nevertheless, the availability of data and computing power remains strong driving forces behind the development of these advanced algorithms and techniques.

Unlike training acoustic models on existing well-studied speech corpora, developing a speech recognizer for Khmer language from a newly collected speech data set bears several challenges.

Firstly, the data is not well-formatted. i.e., long audio files have to be segmented into short files. This leads us to the study of automatic data pre-processing tool based on voice activity detection, which is crucial in the early stage of developing Khmer ASR in this thesis work.

Secondly, the pronunciation dictionary for building Khmer ASR is not available and it has to be prepared from scratch.

Finally, the amount of training data is still very limited given that an advanced acoustic model such as DNN-HMM, which could contain more than 1 million parameters[1], was employed in this work. For 1934 speech utterances from 10 training speakers, there is a total of 254,458 training vectors. However, by modeling 468 tied triphone states, the 5-hidden-layer DNN with 512 nodes per hidden layer contains about 1.5 million parameters, which is more than 6 times the number of parameters of the GMM-HMM with 6 mixture components[2]. Dropout training is remarkably useful in dealing with overfitting of our DNN-HMM acoustic model.

At our first effort, currently the Khmer isolated-word recognition system using DNN-HMM performs at 93.31% word accuracy on the test set, which is lower than that of GMM-HMM that can perform at 97.17% accuracy. This tells us that either our DNNs training recipe is suboptimal since the strategy used in searching for the best combination of hyperparameters is too greedy and naive, or the data at hand indeed severely constrains the training of large neural networks to be more effective than that of GMM-HMM.

Cross-lingual transfer learning was also investigated to try to leverage auxiliary data from English speech corpus [29] but did not help. This might be due to the discriminative training procedure in our English DNN that have made the DNN weights too specialized to the English phoneme set to be generalized for Khmer phones.

Nevertheless, deep learning remains a promising technology for speech recognition and other related fields of research. Exploring more deep learning techniques can only benefit Khmer ASR research in general and far into the future as the current GMM-HMM framework has become less appealing when dealing with continuous speech recognition problem.

5.1 Error Analysis

A glance at our experimental results suggests that GMM-HMM performs better than DNN-HMM in term of word accuracy (%) on the current test set. However, a closer observation into prediction errors committed by each type of acoustic models reveals

[1]5-hidden-layer DNN with 512 hidden nodes used in the experiment contains 1,587,712 weight parameters

[2]GMM-HMM with 6 mixture components has 221,832 parameters

different learning behaviors of those models. GMM-HMM gives fault predictions that are at variance with actual pronunciations of words as shown in Table A.5. On the other hand, many of the prediction errors as in Table A.6, which are generated by DNN-HMM tend to conform with the words' actual pronunciations, i.e., containing similar sound (vowel or consonant). Finally, there are still a few vocabulary words with similar pronunciation or short spanning duration that could be easily mistaken in the present of noise and thus they challenge both GMM-HMM and DNN-HMM. A list of shared prediction errors by GMM-HMM and DNN-HMM can be found in Table A.7.

Chapter 6

Conclusion and Future Work

Building a Khmer speech recognition system using deep learning algorithms has been the ultimate goal of our study. As far as we know, this work marked the first Khmer ASR that uses deep neural networks for acoustic modeling, yet it is merely the beginning. As the use of deep learning in current speech recognition research becomes more common, Khmer ASR that embraces this technology is expected to be better at harnessing new findings in the field.

In this work, we have derived Khmer pronunciation dictionary and phonetic question set which are useful for building context-dependent phone unit based Khmer ASR. A GMM-HMM acoustic model for isolated-word speech recognition system for the task was created as the basis for comparison. The preliminary investigation of DNN-HMM was also conducted to observe the behavior of DNN-HMM on low-resourced, isolated-word speech recognition for Khmer.

Since the performance of the DNN depends crucially on both the amount of data available for system training and how well the hyperparameters are chosen, we will continue to examine unsupervised pre-training, which allows un-transcribed speech data to be used for model training, and to explore different combinations of the hyperparameters for our DNN training. Different types of DNN. i.e., Recurrent neural network (RNN) and Convolutional neural network (CNN) might also be potential candidates for future investigation. After all, we wish to extend our knowledges and lessons learned in this work to tackle Khmer continuous speech recognition using deep learning algorithms.

Appendix A

Tables

ɑ-group	subscript	ɔ-group	subscript	IPA sound	ARPAbet
ក	្ក	គ	្គ	k	K
ខ	្ខ	ឃ	្ឃ	k^h	KH
ង៉		ង	្ង	ŋ	NG
ច	្ច	ជ	្ជ	c	C
ឆ	្ឆ	ឈ	្ឈ	c^h	CH
ញ៉		ញ	្ញ	ɲ	GN
ដ	្ដ	ឌ	្ឌ	ɗ	D
ឋ,ថ	្ឋ្ថ	ឍ,ធ	្ឍ្ធ	t^h	TH
ណ	្ណ	ន	្ន	n	N
ត	្ត	ទ	្ទ	t	T
ប	្ប	ប៊		ɓ	B
ផ	្ផ	ភ	្ភ	p^h	PH
ប៉		ព	្ព	p	P
ម៉		ម	្ម	m	M
យ៉		យ	្យ	j	Y
រ៉		រ	្រ	r	R
ឡ	+ ឡ	ល	្ល	l	L
វ៉		វ	្វ	w	W
ស	្ស	ស៊		s	S
ហ	្ហ	ហ៊		h	HH
អ	្អ	អ៊		ʔ	AA

Table A.1: Consonants sound mapping

Letter	**/ɑ/ sound**	**ARPAbet**	**/ɔ/ sound**	**ARPAbet**
Inherent vowel	ɑː	AA	ɔː	OA
ា	aː	AH	iːə	EA
ិ	e	EH	i	IH
ី	ej	EY	iː	IY
ឹ	ə	OE	ɨ	EO
ឺ	œː	ER	ɨː	EU
ុ	o	OH	u	UH
ូ	ɔːo	OW	uː	UW
ួ	uːə	UE	uːə	UE
ើ	aːə	AER	əː	EER
ឿ	ɨːə	EUR	ɨːə	EUR
ៀ	iːə	EA	iːə	EA
េ	eː	IE	eː	IE
ែ	aːɛ	AE	ɛː	AE
ៃ	aj	AY	ej	EY
ោ	aːo	AW	oː	OW
ៅ	aw	AOW	əw	AUW
ុំ	om	OUM	um	UM
ំ	ɑm	OM	um	UM
ាំ	am	AM	oam	AOM
ះ	ah	EHX	ɛah	AHX
ិះ	eh	EEH	ih	IH
ឹះ	əh	ERH	ɨh	EOH
ុះ	oh	OUH	uh	UUH
េះ	eh	OEH	ih	IYH
ោះ	ɑh	AOH	uəh	UEH

Table A.2: Dependent vowels sound mapping

English sound	Khmer sounds	Khmer characters
/AA/	/AA/, /OA/	/inherent vowel/, /inherent vowel/
/AE/	/AE/, /AER/, /AIE/, /IE/	/ែ/, /ើ /, /េ , /ែ/
/AH/	/AH/, /OUH/	/ ា /, / ុះ /
/AO/	/AOH/, /OH/	/ោះ /, / ុ /
/AW/	/AW/, /AOW/, /AUW/	/ោ /, /ៅ /, /ៅ /
/AX/	/AAK/	/ៈ /
/AXH/	/AHX/	/ះ /
/AXR/	N/A	N/A
/AY/	/AY/	/ៃ /
/B/	/B/	/ប, ប៊/
/BCL/	N/A	N/A
/CH/	/CH/	/ឆ, ឈ/
/D/	/D/	/ដ, ឌ/
/DCL/	N/A	N/A
/DH/	N/A	N/A
/DX/	N/A	N/A
/EH/	/EH/, /EO/, /OEH/	/ ិ /, / ឹ /, /េះ /
/EL/	N/A	N/A
/EM/	/AM/, /OM/, /OUM/, /UM/	/ ាំ /, / ំ /, / ុំ /, / ំ /
/EN/	N/A	N/A
/ENX/	N/A	N/A
/EPI/	N/A	N/A
/ER/	/ER/, /EER/, /EU/, /OE/	/ ឺ /, /ើ /, / ឺ /, / ឹ /
/EY/	/EY/	/ៃ /
/F/	N/A	N/A
/G/	/K/	/ក, គ/
/GCL/	N/A	N/A
/HH/	/HH/	/ហ, ហ៊/
/HV/	N/A	N/A
/IH/	/IH/	/ ិះ /
/IX/	N/A	N/A

Table A.3: English to Khmer sound mappings in ARPAbet format

English sound	Khmer sounds	Khmer characters
/IY/	/IY/, /IYH/, /EA/	/ ី /, /េះ /, /ៀ ,ា /
/JH/	/C/	/ច, ជ/
/K/	/KH/	/ខ, ឃ/
/KCL/	N/A	N/A
/L/	/L/	/ឡ, ល/
/M/	/M/	/ម៉, ម/
/N/	/N/	/ណ, ន/
/NG/	/NG/, /GN/	/ង៉, ង/, /ញ៉, ញ/
/NX/	N/A	N/A
/OW/	/OW/	/ោ /
/OY/	N/A	N/A
/P/	/P/, /PH/	/ប៉, ព/, /ផ, ភ/
/PAU/	N/A	N/A
/PCL/	N/A	N/A
/Q/	N/A	N/A
/R/	/R/	/រ៉, រ/
/S/	/S/	/ស, ស៊/
/SH/	N/A	N/A
/T/	/T/, /TH/	/ត, ទ/, /ឋ,ថ,ឍ,ធ/
/TCL/	N/A	N/A
/TH/	N/A	N/A
/UH/	/UH/, /UE/, /UEH/, /UEK/	/ ុ /, / ួ /, /ោះ /, /ក់/
/UW/	/UW/	/ ូ /
/UX/	N/A	N/A
/V/	N/A	N/A
/W/	/W/	/វ៉, វ/
/Y/	/Y/	/យ៉, យ/
/Z/	N/A	N/A
/ZH/	N/A	N/A

Table A.4: English to Khmer sound mappings in ARPAbet format (Cont.)

Actual Predicted	IPA	English
ខែមករា	/kɛː.mɑkaːraː/	January
ខែមិថុនា	/kɛː.mi.t^{h}o.naː/	June
ថ្ងៃអាទិត្យ	/thŋaj.ʔaː.tit/	Sunday
តាកែវ	/taː.kaːɛw/	Takeo[1]
១៥	/ɗɑɓ.pram/	Fifteen
អត់ព្រម	/ʔɑːt.prɔːm/	Disagree
៩	/pram.ɓuːən/	Nine
៤	/ɓuːən/	Four
មាន់	/mʔɑːn/	Chicken
៩	/pram.ɓuːən/	Nine
ថ្ងៃ	/t^{h}ŋaj/	Day
ហៅ	/haw/	Call
អាទិត្យ	/ʔaː.tit/	Week
ខេត្តកែប	/kaːɛt.kaːɛɓ/	Keb province
ថ្ងៃនេះ	/t^{h}ŋaj.nih/	Today
ថ្ងៃសៅរ៍	/t^{h}ŋaj.saw/	Saturday
អំពៅ	/ʔɑm.pəw/	Cane
គ្រុនក្តៅ	/krun.kɗaw/	Fever

Table A.5: Recognition errors committed by GMM-HMM but not DNN-HMM

[1] Takeo, Keb and Svayriang are province names.

Actual Predicted	IPA	English
កៅស៊ូ	/kaw.suː/	Rubber
ថ្ងៃសុក្រ	/t^{h}ŋaj.sok/	Friday
ថ្ងៃអាទិត្យ	/t^{h}ŋaj.ʔaː.tit/	Sunday
អាទិត្យ	/ʔaː.tit/	Week
ទឹកនោមផ្អែម	/tɨk.noːm.p^{h}ʔaːɛm/	Diabetes
ទឹកនោមប្រៃ	/tɨk.noːm.ɓraj/	Nephrosis
ស្រូវ	/srɔːow/	Paddy
ស្វាយ	/swaːj/	Svay[2]
ព្រម	/prɔːm/	Agree
ប្រុស	/ɓros/	Male
ថ្ងៃព្រហស្បតិ៍	/t^{h}ŋaj.prɔː.hɑːs/	Thursday
ភ្នែកក្រហម	/p^{h}nɛːk.krɑː.kɑːm/	Red eye
រលាកថ្លើមអា	/rɔː.liːək.t^{h}laːəm.ʔaː/	Hepatitis A
រលាកថ្លើមបេ	/rɔː.liːək.t^{h}laːəm.ʔaː/	Hepatitis B
តាកែវ	/taː.kaːɛw/	Takeo[1]
ក្រចៅ	/krɑː.caw/	Jute
ខែសីហា	/kɛː.sej.haː/	Auguest
ខេត្តព្រៃវែង	/prej.waːɛŋ/	Preyveng province
ម្សិលម្ងៃ	/kɛː.sej.haː/	Before yesterday
ម្សិលមិញ	/prej.waːɛŋ/	Yesterday
ស្វាយរៀង	/swaːj.riːəŋ/	Svayriang
ស្វាយ	/swaːj/	Svay[2]
ប្រកាំង	/ɓrɑː.kamŋ/	Migraine
តាកែវ	/taː.kaːɛw/	Takeo[1]
អំពៅ	/ʔɑm.pəw/	Cane
កំពត	/kɑm.pɔt/	Fever

Table A.6: Recognition errors committed by DNN-HMM

[2]ស្វាយ and ស្អុច are disease names and English translations are just the sound transliteration from Khmer.

Actual Predicted	IPA	English
ខែ	/kɛː/	Month
ចា	/caː /	Yes (female speaker)
អេដ៍	/ʔeːd/	HIV
កែប	/kaːɛɓ/	Keb[1]
ទេ	/teː/	No
កែប	/kaːɛɓ/	Keb[1]
ស្អុច	/sʔɑːoc/	Saoch[2]
ពោត	/poːt/	Corn
អង្ករ	/ʔŋ.kɑː/	Rice
បន្ត	/ɓɑːn.tɑː/	Continue
ហៅ	/haw/	Call
ស្រូវ	/srɔːow/	Paddy
១៧_ដប់ប្រាំពិល	/ɗɑɓ.pram.pil/	Seventeen
៧_ប្រាំពិល	/pram.pil/	Seven
១៨	/ɗɑɓ.pram.ɓej/	Eighteen
៨	/pram.ɓej/	Eight
១៩	/ɗɑɓ.pram.ɓuːən/	Nineteen
១៤	/ɗɑɓ.ɓuːən/	Fourteen

Table A.7: Recognition errors committed both GMM-HMM and DNN-HMM

Bibliography

[1] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach.* Springer-Verlag London, 2015.

[2] Cambridge University Engineering Department. *The Hidden Markov Model Toolkit (HTK).* http://htk.eng.cam.ac.uk, 2015.

[3] Carnegie Mellon University. *CMU Sphinx.* http://cmusphinx.sourceforge.net, 2015.

[4] Daniel Povey. *Kaldi project.* http://kaldi-asr.org, 2015.

[5] Rachel Nuwer. *Why we must save dying languages.* BBC - http://www.bbc.com/future/story/20140606-why-we-must-save-dying-languages, 2014.

[6] Department of Computer Science, Institute of Technology of Cambodia. *Khmer Keywords dataset [unpublished].* PO Box 86, Russian Conf. Blvd. Phnom Penh, Cambodia. 2014.

[7] Sopheap Seng, Sethserey Sam, Viet-Bac Le, Brigitte Bigi and Laurent Besacier. *Which Units For Acoustic And Language Modeling For Khmer Automatic Speech Recognition?.* LIG Laboratory, UMR 5217. BP 53, 38041 Grenoble Cedex 9, FRANCE. 2007.

[8] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* PTR Prentice Hall, Englewood Cliffs, New Jersey 07632, 1993.

[9] Daniel Jurafsky and James H. Martin. *Speech and Language Processing, (2nd Edition)* Pearson Prentice Hall, Upper saddle river, New Jersey 07458, 2009.

[10] Hynek Hermansky. *Perceptual linear predictive (PLP) analysis of speech.* Speech Technology Laboratory, Division of Panasonic Technologies, Inc., 3888 State Street, Santa Barbara, California 93105, 1990.

[11] Lawrence R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceeding of the IEEE, Vol, 77, No.2, February 1989.

[12] Mirjam Killer, Sebastian Stüker and Tanja Schultz. *Grapheme Based Speech Recognition.* Eurospeech, Geneva, 2003.

[13] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Anton Ragni, Valtcho Valtchev, Phil Woodland and Chao Zhang. *The HTK Book (for HTK Version 3.5, documentation alpha version).* Cambridge University Engineering Department, 2015.

[14] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. *Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of four research groups.* IEEE Signal Processing Magazine, pp. 1053-5888, 2012.

[15] A. J. Viterbi. *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.* IEEE Transactions on Information Theory, 1967.

[16] Hermann Ney and Stefan Ortmanns. *Dynamic Programming Search for Continuous Speech Recognition.* IEEE Signal Processing Magazine, vol. 16, issues 5, pp. 64-83, 1999.

[17] Jianlin Cheng. *Hidden Markov Models Lecture Slide.* Department of Computer Science, University of Missouri-Columbia, 2015.

[18] Mehryar Mohri, Fernando Pereira and Michael Riley. *Weighted Finite-State Transducers in Speech Recognition.* AT&T Labs – Research and Computer and Information Science Dept., University of Pennsylvania, 2001.

[19] Central Intelligence Agency Library. *The World FactBook - Cambodia.* https://www.cia.gov/library/publications/the-world-factbook/geos/cb.html, 2015.

[20] Kimchhoy Phong and Javier Solá. *Research Study: Mobile Phones in Cambodia 2014.* https://www.cia.gov/library/publications/the-world-factbook/geos/cb.html, 2015.

[21] J. Ramírez, J. M. Górriz and J. C. Segura. *Voice Activity Detection. Fundamentals and Speech Recognition System Robustness.* University of Granada, Spain, 2007.

[22] Annanda th. Rath, Long S. Meng, Heng. Samedi, Long Nipaul, Sok K. heng. *Complexity of Letter to Sound Conversion (LTS) in Khmer Language: under the context of Khmer Text-to-Speech (TTS).* NLP lab, Department of Computer and Communication Engineering, Institute of Technology of Cambodia, Cambodia, PAN10 and IDRC Canada.

[23] Leonard E. Baum and Ted Petrie. *Statistical Inference for Probabilistic Functions of Finite State Markov Chains.* Ann. Math. Statist. vol. 37, issue 6, pp. 1554-1563, 1966.

[24] Veton Këpuska. *Search and Decoding in Speech Recognition: Automatic Speech Recognition Lecture Slide.* Electrical and Computer Engineering, Florida Institute of Technology.

[25] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach.* The Kluwer International Series in Engineering and Computer Science; v. 247, Boston: Kluwer Academic Publishers, 1994.

[26] Frank Seide, Gang Li and Dong Yu. *Conversational Speech Transcription Using Context-Dependent Deep Neural Networks.* INTERSPEECH, 2011.

[27] George E. Dahl, Dong Yu, Li Deng and Alex Acero. *Large Vocabulary Continuous Speech Recognition with Context-Dependent DBN-HMMs.* University of Toronto, Department of Computer Science, Toronto, ON, Canada and Speech Research Group, Microsoft Research, Redmond, WA, USA.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* Journal of Machine Learning Research. vol. 15, pp. 1929-1958, 2014.

[29] John S. Garofolo, Lori F. Lamel, William M. Fisher, Jonathon G. Fiscus, David S. Pallett, Nancy L. Dahlgren. *The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM.* National Institute of Standards and Technology, 1990.

[30] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng and Yifan Gong. *Crosslanguage Knowledge Transfer using Multilingual Deep Neural Networks with Shared Hidden Layers.* ICASSP, 2013.