Open-Vocabulary Audio Keyword Spotting with Low Resource Language Adaptation

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Kirandevraj R
2019701001
kirandevraj.r@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
November 2022

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Open-Vocabulary Audio Keyword Spotting with Low Resource Language Adaptation"** by **Kirandevraj**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. C.V. Jawahar
Date	Adviser: Prof. Vinay P Namboodiri
Date	Adviser: Prof. Vinod K Kurmi

To Maa

Acknowledgments

I would like to express my sincere gratitude to my adviser, Prof. C.V. Jawahar, for his guidance and support. I am grateful to him for providing me with the opportunity to join the lab. I would like to earnestly thank my co-advisor, Prof. Vinay Namboodiri, for helping me build and progress our research project, sharing his wisdom, and guiding me. I would also like to sincerely thank my co-advisor, Prof. Vinod Kurmi, for constantly helping me in my progress, lifting, and guiding me in my research. I would like to thank all my advisors for their guidance.

I am thankful to my friends Deepak, Ashish, Kavin, and senior Avijit for being on my side through my MS journey. I am also thankful to my CVIT seniors, Shyam, Chris, and Jerin, for sharing their knowledge and experience. I am grateful to my colleagues, Harish, Sangeeth, Rudhrabha, Prajwal, Sindhu, Piyush, Siddhant, Shubham, Sashank, and Zeeshan, for their comradeship. I would also like to extend my gratitude to CVIT administrative team, Rohita and Siva.

Finally, I would like to thank my family for their constant love and support.

Abstract

Open-Vocabulary Keyword spotting solves the problem of spotting audio keywords in an utterance. The keyword set can incorporate keywords the system has seen and not seen during training, functioning as zero-shot keyword spotting. The traditional method involves using ASR to transcribe audio to text and search in the text space to spot keywords. Other methods include obtaining the posteriors from a Deep Neural Network (DNN) and using template matching algorithms to find similarities. Keyword spotting does not require transcribing the entire audio and focuses on detecting the specific words of interest.

In this thesis, we aim to explore the usage of the Automatic Speech Recognition (ASR) system for Keyword Spotting. We demonstrate that the intermediate representation of ASR can be used for open vocabulary keyword spotting. With this, we show the effectiveness of using Connectionist Temporal Classification (CTC) loss for learning word embeddings for keyword spotting. We propose a novel method of using the CTC loss function with the traditional triplet loss function to learn word embeddings for keyword spotting on the TIMIT English language audio dataset. We show this method achieves an Average Precision (AP) of 0.843 over 344 words unseen by the model trained on the TIMIT dataset. In contrast, the Multi-View recurrent method that learns jointly on the text and acoustic embeddings achieves only 0.218 for out-of-vocabulary words.

We propose a novel method to generalize our approach to Tamil, Vallader, and Hausa low-resource languages. Here we use transliteration to convert the Tamil language script to English such that the Tamil words sound similar written with English alphabets. The model predicts the transliterated text for input Tamil audio with CTC and triplet loss functions. We show that this method helps transfer the knowledge learned from high resource language English to low resource language Tamil.

We further reduce the model size to make it work in a small footprint scenario like mobile phones. To this extent, we explore various knowledge distillation loss functions such as MSE, KL Divergence, and CosEmbedding loss functions. We observe that small-footprint ASR representation is competitive with knowledge distillation methods for small-footprint keyword spotting.

This methodology makes use of existing ASR networks trained with massive datasets. It converts them into open vocabulary keyword spotting systems that can also be generalized to low-resource language.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The internet can provide a vast amount of information in a reliable manner and has become an integral part of our lives. This information which is stored on huge databases, is searched primarily by text. However, social media constantly produces massive amounts of non-textual data (e.g., audio, video, etc.). Therefore, retrieval algorithms only based on text can give minimal search results. We can use speech to search these online contents as it directly correlates with the audio form.

This problem is approached by spoken term detection, where the user-generated spoken query is used to search through the database and retrieve audio documents containing the query. The traditional method involves using Automatic Speech Recognition (ASR) to transcribe audio to text and search in the text space through string matching to spot keywords of interest. Here, both the spoken keyword query and the audio utterances from the database are converted to text comprising of words, characters, and symbols. Then text-based informational retrieval techniques are applied to detect the keywords. In these approaches, the performance of a spoken term detection system is largely dependent on the accuracy of the underlying ASR system [29, 4]. However, building a good ASR system requires a large quantity of annotated data, and very few languages have such resources. Thus, these approaches become quite challenging when such methods are used to search through low-resource languages.

This motivated us to focus on Keyword Spotting. In speech processing, keyword spotting deals with the identification of keywords in utterances, as depicted in Figure 1.1. Such systems should be able to retrieve test utterances that have audio query samples in them and can be known as query-by-example spoken term detection. The system handles zero-resource tasks as the query keywords can be audio samples the system has not seen during training. Thus this method can be extended to low-resource languages with no constraints on lexicons and accents. This essentially is a problem of finding a similar pattern in audio. Thus the keywords are expressed as articulate sounds with information encoded, and keyword spotting involves detecting such information. We can search multilingual audio archives such as social media and radio broadcasts with such a keyword spotting system. Recent works on query-by-example spoken term detection involve template matching approaches such as DTW for improved performance compared to traditional statistical speech processing methods in zero-resource conditions [35, 42]. The two main steps involved in these approaches are feature extraction and template matching.

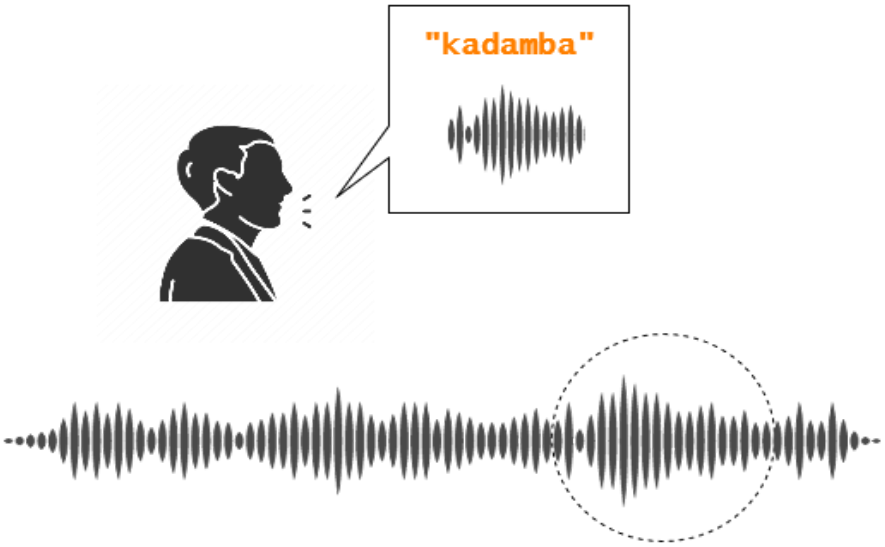


Figure 1.1: Keyword Spotting: Detecting the presence of a query keyword in a test utterance.

The features extracted are the posterior probabilities of a set of phonetic classes. These are estimated outputs of the deep neural network (DNN) and have been successfully used for this purpose [42]. The Dynamic Time Warping algorithm (DTW) [38] compares the query with the test utterance and provides a similarity score. Deep Neural Networks are researched to improve the Keyword Spotting systems' performance and achieve real-life scenarios.

In this thesis, we propose our approach to achieving a low-dimensional subspace representation of speech for QbE-STD. We utilize this approach to improve the state-of-the-art system performance by generating features for Keyword Spotting and generalizing the method to low-resource language systems.

The rest of this chapter is organized as follows. In Section 1.1, we discuss our motivation behind the approach proposed in this thesis. We discuss the application that we built and future potential applications in section 1.2. We share the problem context in Section 1.3. We summarize the contributions of this thesis in Section 1.4. Finally, the chapter-wise outline is presented in Section 1.5

1.1 Motivation

There is a recent trend in the growth of on-device voice-based artificial intelligence machines such as Alexa, Google Home, Siri, and Cortana on wearables and mobile devices. However, to develop such systems, huge training data and resources are required for efficient performance in the real-world scenario. We propose a deep neural network architecture that functions as a high-performance KWS system

for arbitrary keyword sets. We use an ASR trained on large speech corpus as a backbone to build our KWS system. This ASR functions as a transfer network. We conduct experiments using Google Speech Commands Dataset V2 and TIMIT datasets. We adopt this system from a high-resource language setting to a low-resource language scenario and measure its performance using three low-resource languages from the MSWC (Multilingual Spoken Words Corpus) dataset. The proposed network is competitive with state-of-the-art performance in both experiments.

A spoken utterance is a sequence of words expressed to convey information made up of phones. The human speech production process produces speech by the use of the nose, throat, and vocal cord as part of the articulatory mechanism. Thus the generated speech data lies in a non-linear dimensional space [7, 25]. A union of low-dimensional spaces can model the non-linear high-dimensional speech manifolds. Sparse approximation techniques can also model these non-linear speech activities [44, 10]. Large vocabulary and noise robust speech recognition utilize this property to operate on higher dimensional manifolds [44, 10]. This thesis aims to utilize this low-dimensional structure to improve the QbE-STD system.

In this work, we train a Deep Neural Network (DNN) in a high-resource language such as English and other low-resource languages such as Tamil, Vallader, and Hausa to extract features. This work achieves significant improvements over the state-of-the-art baselines for open-vocabulary keyword spotting. The methodology adapts the system from one language to another and has been a motivation to extend the work to a low-resource language scenario. We focus on DNN-based learning due to its strong modeling capability and success in several other problems e.g., speech recognition [18, 12], image classification [47, 15], machine translation [2, 34], etc.

1.2 Applications

We will discuss about two main applications of KWS. Phone call based News search by keyword spotting, and Scam call detection by keyword spotting.

1.2.1 Phone call based News Search

The main motivation behind the project was raised when we wanted to build a voice-based news application through telephone communication. We wanted to build this application to cater to the needs of people who have cell phones without an internet connection or non-smartphone users. Around 18 percent of the Indian population do not have an internet connection while they have access to a mobile.

We built this application such that the user was able to call a mobile number and ask for news topics, e.g. 'cricket scores' or 'election results'. The user is served with news articles related to cricket scores and election results. We built this application using the Automatic Speech Recognition model DeepSpeech2 to convert the input user audio query to text and search news articles in the text space, and convert the relevant new article back to the audio using the Text-to-Speech system Tacotron. The audio

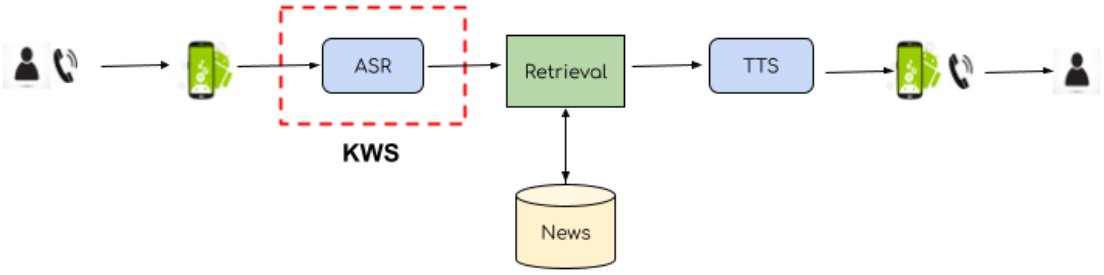


Figure 1.2: Phone call-based News search: The user makes a call to the system; the user speech is recognized using ASR to search relevant news queries; these news queries are converted to audio using TTS and is served back to the user. The ASR system is later replaced by our KWS system.

is served back to the user in the same phone call. We run our news application in a flask server. Twilio API handles data transfer from the telephone network to the web server and vice versa. This is depicted in the Figure 1.2

The ASR model in the pipeline had to be run on GPUs to recognize audio. Hence we wanted to replace the ASR model with a Keyword Spotting model. Such the KWS model should be able to spot keywords from the user's voice query that is relevant to the latest news articles and serve them news based on that. The additional constraint to such a KWS model is that the relevant news keywords keep changing over a week, and hence the model should be able to perform open vocabulary keyword spotting. The ultimate goal from there is to build such a system for low resource Indian language environment.

1.2.2 Scam Call Detection

There is an alarming increase in the number of scam calls people receive these days. In India, as many as 31% of those who continued with these interactions lost money in 2021. The growth of smartphone access to kids and the elderly unaware of such schemes are the primary victims. Though the scammers use the latest technology to conceal their identity, the conversation to do such tasks is almost similar.

We propose a keyword spotting approach where such scenarios can be averted. We begin by finding out the list of keywords that are used in the conversation by the scammers to deceive people. We can then use our keyword spotting model to detect these keywords. When these keywords are detected in the conversation, the user is alerted of a potential scam threat. This helps the elderly and innocent kids to consult and avoid treacherous scenarios. Unlike ASR, the keyword spotting system does not transcribe the entire audio and preserves user privacy.

1.3 Problem Context

There are many problems related to keyword spotting in audio under various constraints, such as fixed-vocabulary or open vocabulary keyword spotting with training pairs and transcripts. We realized that there is less work focused on analyzing Automatic Speech Recognition for open vocabulary Keyword Spotting. Hence, we focus on analyzing the Automatic Speech Recognition model for keyword spotting.

The ASR has not directly been trained to do keyword spotting. Analyzing an ASR and making it usable for a keyword spotting task is challenging. Hence we find a viable way to use the ASR system for KWS and then modify the ASR system by fine-tuning it for keyword spotting tasks. On the other hand, the keyword set is open, and hence the model should be able to work for the keywords the model has never seen.

Open Source ASR models pre-trained on huge datasets and with huge resources can be easily leveraged when these ASRs are directly modified into a KWS system with little modification than building a KWS for open vocabulary grounds up.

1.4 Contribution

Our contributions are as follows:

1. We demonstrate that intermediate representations from ASR when fine-tuned using triplet and Connectionist Temporal Classification (CTC) loss functions on keyword audio segments extracted from the TIMIT [9] dataset can provide generalized keyword spotting abilities.

2. We then further generalize our approach to low-resource language Tamil from the MSWC [32] dataset. To apply CTC loss on low-resource language transcripts, we convert the low-resource language characters to English using transliteration.

3. Finally, we train our model on the Google Speech Commands dataset [50] and compare our results on in-vocabulary classification tasks by using kNN on our acoustic word embeddings and show that they provide comparable in-vocabulary results to the state of the art while having improved generalization abilities.

4. We further work towards reducing the size of the model to make it work in a small footprint scenario by using various knowledge distillation loss functions. We realize that small footprint ASR representation is directly competitive with knowledge distillation methods for keyword spotting.

1.5 Thesis Organisation

We organize the thesis as follows: Chapter 2 presents the overview of related works in the literature to address keyword spotting. We summarize the various methodologies used to build the KWS model. We discuss the different methods to pre-process the speech signals before using them for training. We elaborate on the datasets used in this work. We share details on evaluation methods as well.

In Chapter 3, we present the basics of the Automatic Speech Recognition system and training loss function. We discuss our proposed method of using the triplet loss function in ASR training to obtain representation used for open vocabulary keyword spotting. We discuss about the training details and its performance on fixed vocabulary keyword tasks in addition. We discuss the qualitative results of the system and visualize the results using tSNE plots. We explain how this method can be extended to continuous audio as opposed to isolated audio segments. We provide empirical evidence for all our claims.

Chapter 4, concerns adapting the framework for low-resource languages in the same open vocabulary setting. We explain the effectiveness of transliteration in helping to adapt pre-trained ASR for low-resource languages. We further venture into reducing the size of the model to make it work in a small footprint scenario. We discuss various distillation techniques for the small footprint approach and conclude with the effectiveness of small-footprint ASR representation against distillation techniques.

In Chapter 5, we discuss all the contributions made in our work and the summary of its uses. We share future directions that can build on this work and other potential problems we encountered in the same research direction.

Chapter 2

Background

2.1 Introduction

In this chapter, we provide a brief background on the task of query-by-example spoken term detection (QbE-STD). We summarize several techniques proposed in the literature for solving QbE-STD in Section 2.2. The details of different acoustic feature vectors used in this thesis to represent the speech signal are provided in Section 2.3. Then we describe different monolingual and multilingual databases used to train and evaluate the proposed systems in Section 2.4. Several evaluation metrics are used to thoroughly examine and compare our system with the state-of-the-art, as summarized in Section 2.5. Finally, we present a brief description of our baseline system in Section 2.6.

2.2 Related Work

Initially, we summarize the various approaches to solving open vocabulary query-by-example keyword spotting. The first approach involves spotting the keywords in two steps: (i) feature extraction and (ii) template matching. Feature extraction involves representing the spoken query using Mel Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction (PLP) based on spectral features. Template matching tasks are investigated on these features [45]. Later posterior features from the models trained using supervised and unsupervised methods are explored [14]. This posterior features method has better performance than template matching on features. Given GMM models and a test feature vector, the log-likelihood value of that feature set belonging to a particular GMM is given by the posterior probability. Posterior features have been widely used in template-based systems [55, 37]. Deep Boltzmann Machine (DBM) is used to extract posterior features. These features are trained based on unsupervised as well as semi-supervised manner. The DBM trained in an unsupervised manner captures the hierarchical structural information in the data. In [56], the authors initially train a DBM with unlabeled data and adapt it to a small amount of labeled data by fine-tuning it. The posteriors from GMM are used as labels for DBM training [56]. The posteriors that are obtained from DBM in these cases have improved performance than the posteriors from GMM.

The posterior features obtained from supervised learning methods involve training a DNN with labeled data. The DNN trained with the labeled data containing characters and phones adapts to low-resource, and zero-resource languages [14, 42]. Template matching is performed on the extracted posterior features on these DNNs to do Query-by-Example Keyword spotting. The contents of the speech signals characterize these posteriors irrespective of the underlying language [42]. The DNNs trained with bottleneck layers to perform Keyword Spotting in a multilingual setting with the bottleneck features [3].

We obtain the frame-level distance matrix by comparing the embeddings of the spoken query and the test utterance. A DTW algorithm calculates the similarity between the spoken query and the test utterance from the distance matrix. The query can occur anywhere in the test utterance; hence, the DTW algorithm checks for the entire matrix for the similarity to be high; hence, heavy computation is required. The segmental DTW processes the distance matrix in segments of bands tolerance to temporal distortion [37]. Segmental DTW does not take care of speaking rates in the signals. The slope of the warping path in the distance matrix is penalized while spotting the keyword in the utterance in slope-constrained DTW [55]. Slope-constrained DTW reduces the computations as the query frames map to fewer utterance frames. The sub-sequence DTW [36] enforces the insertion cost as 0 at the start of the query and its end. It allows the warping path to find the query without concern for the query's beginning and end point in the matrix.

The speech signal's acoustic units are discovered and modeled using Hidden Markov Models (HMM). The units used in these models are phones or sub-phones. During Keyword spotting, We convert the spoken query and the test utterance into the HMMs' symbolic units, and the query is retrieved using search techniques on these symbolic units. Most keyword spotting systems use dynamic programming to search for the optimal keyword spotting solution or use DTW to counter the length mismatch of the keyword in the query and the target term [8, 53]. The recent trend involves building a representation space for open vocabulary keyword spotting. The query and utterances are converted to this space and compared [40, 17]. We are approaching this problem in a similar direction.

In recent years, various forms of the neural network have proven to be useful for keyword spotting. A convolutional neural network (CNN) is trained on frame-level similarities between the posteriors of a spoken query and a test utterance in [41]. A Siamese CNN network is used to generate the embedding space in [22]. A multitask objective of learning acoustic word embedding with triplet loss and cross-entropy loss has been explored in [46]. Seq2seq models are used for KWS. The text and audio representation are brought together by the encoder, decoder, and a feed-forward neural network [1]. ASR posteriors are used to search queries in [43]. An encoder network with an attention mechanism is used to learn the representation for the query in [54]. An LSTM is trained to discriminate phones with the CTC criterion, and a substring matching algorithm is used to detect the keyword in [58]. The representations are obtained for fixed vocabulary KWS using triplet loss and are classified using k-Nearest Neighbour (kNN) in [49]. Recurrent Neural Networks combined with convolution are used in KWS

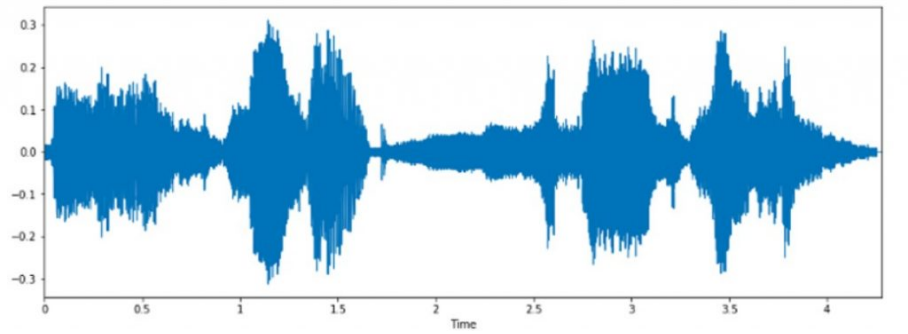
[6, 24, 30]. Few-shot keyword spotting method has been experimented in [31]. Siamese Recurrent Autoencoders[57] have approached a similar task.

While these contributions have been focused on the task of KWS and have made significant contributions to the task, our contribution differs in terms of being focused on generalizing the KWS ability to a set of arbitrary keywords rather than a fixed small command set. The KWS task used here is the same as that in the literature [16], which is similar to an isolated word recognition task, and it is different from detecting keywords from a continuous speech signals [8, 30].

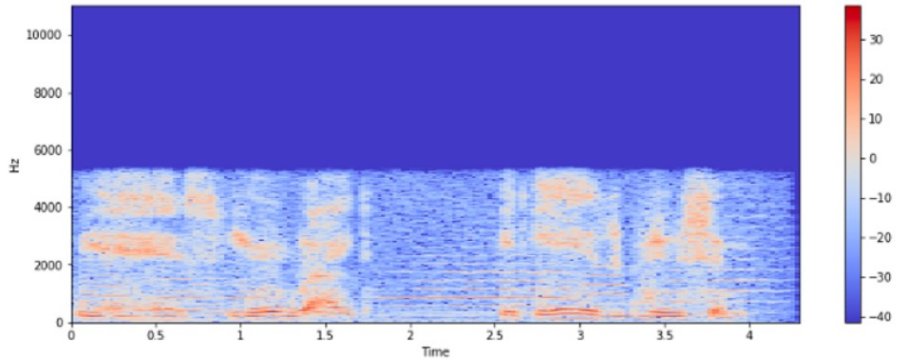
2.3 Feature Vectors

This section describes different feature vectors used in this thesis to represent speech signal.

2.3.1 Spectrogram



(a) Sample Waveform



(b) Example to depict a Spectrogram of the Sample Waveform: 2D image with frequency as x-axis and time as y-axis

Figure 2.1: Example to depict a Spectrogram of the Sample Waveform

Spectrograms are a visual representation of audio. We represent a signal as $s(t)$ with t representing time dimension. We process a small window ω of audio at a time. We calculate the magnitudes of various frequencies on these chunks using Fourier Transform. We place the vectors one after another

as we move the window of the processing audio. We use Short-Time Fourier Transform (STFT) while computing the spectrogram. We take the squared magnitude of STFT to obtain the spectrogram. We can express this as:

$$spectrogram(t, \omega) = |STFT(t, \omega)|^2 \tag{2.1}$$

The spectrogram is a matrix. To visualize it, we can view the matrix as an image with the i, j-th entry in the matrix corresponding to the intensity or color of the i, j-th pixel in the image. Spectrogram of a sample audio is shown in Figure 2.1b

2.3.2 Mel Frequency Cepstral Coefficients

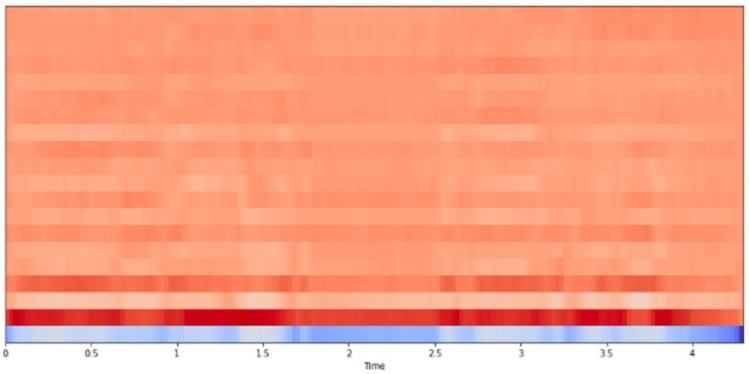


Figure 2.2: Example to depict an MFCC of the Sample Waveform: The frequency bands are equally spaced on the mel scale, approximates the human auditory system, and hence have lesser features than the spectrogram.

In speech processing, Mel Frequency Cepstral Coefficients, commonly known as MFCC feature [5] is a short-term power spectrum based representation of speech signal. These features are widely used in automatic speech recognition and speaker recognition systems, which are extracted by windowing the speech signal into short segments. Generally, a sliding window of 25ms with a shift of 10ms is used. The short segments are assumed to be stationary and are referred to as a frame. Fast Fourier transform (FFT) is applied to the sequence of frames to compute the corresponding power spectrum. A filter-bank analysis follows this step to obtain the energies in various frequency regions. The filter banks are computed by applying triangular filters, typically 40 filters. The filters are linearly spread in the mel domain, which is chosen to mimic the human auditory perception. The final step is to compute the discrete cosine transform of the log of the filter-bank energies. It is required to de-correlate the filter-bank energies. Typically for speech recognition and related tasks, the first 13 coefficients are used. These coefficients are concatenated with their delta (Δ), and double-delta ($\Delta\Delta$) features in order to incorporate the trajectory information of features. Let the MFCC feature vectors for an utterance is represented by, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$. The corresponding delta features are computed using linear

regression over a window of W as follows:

$$\Delta_t = \frac{\sum_{w=1}^{W} w\left(\mathbf{x}_{t+w} - \mathbf{x}_{t-w}\right)}{2\sum_{w=1}^{W} w^2} \tag{2.2}$$

where, Δ_t is the delta feature vector for the t-th frame of an utterance X. The double-delta features can be obtained by successively applying to Equation (2.2). MFCC features of a sample audio is shown in the Figure 2.2 .

2.3.3 Posterior Features

Posterior features are a representation of audio after processing it through a model. The output from a DNN representing a specific task's features can be considered posterior features. These features consist of rich information on the audio corresponding to the DNN task. For example, the posteriors from the ASR consist of character information from the audio. DNNs such as convolutional layers and Recurrent layers can be used to generate these features [14, 42]. Template matching on these features has shown performance improvement. In [42, 39], keyword spotting is performed using phone-based posteriors. The audio features are converted to either spectrogram or MFCC. A DNN is trained to predict phones using these features and can then be used to get posterior features for query audio.

2.3.4 Bottleneck Features

Auto-encoders encode the information to a lower dimension and then retrieves the information back from these low dimensional representation. The bottleneck layers have lesser dimensions when compared with the other layers [19, 52, 48]. Here the information is made to pass through a hidden bottleneck layer, and the representation obtained from such a layer is called the bottleneck features [19]. For classification tasks, [52] has used these bottleneck features. We can test the performance of such bottleneck features on low-resource languages [48].

The primary difference is that the network has a bottleneck layer, and posterior features are that the bottleneck features come out of the bottleneck layer instead of the output layer.

2.4 Databases

We have used four databases to train and evaluate different systems proposed in this work. We present a brief description of these datasets in the following sections.

2.4.1 Librispeech

The Librispeech is an English speech dataset curated for building speech recognition systems. The dataset is built as part of the LibriVox project. The dataset contains almost 1000 hours of spoken audio

along with their transcripts. The sampling rate of these audio files is 16kHz. The dataset is available for free download. The DNN acoustic models, when trained with the librispeech dataset, gave higher performance on Wall Street Journal (WSJ) dataset than the models trained on WSJ dataset.

The dataset is split into three subsets with an approximate duration of 100, 360, and 500 hours. An automatic procedure is used to find the clean samples from the noisy samples. If the sample is transcribed with a lesser error rate on the test ASR is considered less noisy if not, they are kept in the nonclean set. The clean pool is randomly split into 100 and 360 clean sets. The maximum duration of the audio for each speaker is 25 minutes to avoid data imbalance in the training set. Similarly, for the dev and test set, the maximum duration is 8 minutes. The dev and test set is 5 hours and 20 minutes each.

2.4.2 Google Speech Commands

The Google Speech Commands dataset [50] is a fixed-vocabulary keyword spotting dataset. The dataset is curated for 35 keywords. The keywords have samples ranging from 1000 to 4000. The length of the keywords ranges from 2 to 8. The focus of such a dataset is to build keyword spotting systems capable of performing accurately for these keywords only. The dataset is spoken by 2618 English speakers having different accents and speaking styles. The training, validation, and test set contains 84843, 9981, and 11005 samples. The word choice and the utterance count are given in table 2.1

Table 2.1: List of Words in Google Speech Commands Dataset with the utterance count

Words	Utterances	Words	Utterances	Words	Utterances
Backward	1,664	Happy	2,054	Sheila	2,022
Bed	2,014	House	2,113	Six	3,860
Bird	2,064	Learn	1,575	Stop	3,872
Cat	2,031	Left	3,801	Three	3,727
Dog	2,128	Marvin	2,100	Tree	1,759
Down	3,917	Nine	3,934	Two	3,880
Eight	3,787	No	3,941	Up	3,723
Five	4,052	Off	3,745	Visual	1,592
Follow	1,579	On	3,845	Wow	2,123
Forward	1,557	One	3,890	Yes	4,044
Four	3,728	Right	3,778	Zero	4,052
Go	3,880	Seven	3,998		

2.4.3 TIMIT

The Texas Instruments/Massachusetts Institute of Technology (TIMIT) dataset is used to build Automatic Speech Recognition systems. The dataset is curated to learn the relationship between the audio and the corresponding phones. Hence the text sentences for building the dataset has a diverse set of

phones to capture the different accents and styles. The dataset is built by 630 speakers from 8 major dialects of American English. Each speaker utters 10 phonetically rich sentences. The time alignment of the phonemes in this speech is provided in this dataset. The dataset has a total of 6300 utterances. There are three types of sentences the TIMIT dataset can be divided.

- **Dialect:** The dialect sentences have the capability to expose the variations in the dialect of different speakers. All 630 speakers read these sentences. An example of such a dialect sentence includes ”She had your dark suit in greasy wash water all year.”. The pronunciation of the word ”greasy” is different in different dialect speakers.
- **Compact:** The phonetically compact sentences are hand-designed to compass all pairs of phones. These sentences are compact as well as comprehensive. Since there are extra phones in the sentences than average, these sentences are considered to be difficult. Each speaker read 5 of these sentences, and 7 different speakers spoke each text.
- **Diverse:** In order to add diversity in the sentence types and their phones, the sentences are collected from text sources - the Brown Corpus [27], and stage plays. These texts increased the variety of allophonic contexts in the dataset. Each speaker read 3 of these sentences, with each sentence being read by only a single speaker.

Table 2.2: Number of speakers per sentence and Number of sentences per speaker information of the various sentence types in the TIMIT dataset

Sentence Type	Sentences	Speakers/Sentence	Total	Sentences/Speaker
Dialect	2	630	1260	2
Compact	450	7	3150	5
Diverse	1890	1	1890	3

We use TIMIT [9] dataset to train and test our architecture and the baseline. TIMIT dataset contains approximately five hours of speech. We segment words based on word boundaries instead of voice activity detection in this work. After that, the speech segments are grouped into separate words, and functional words like articles and conjunctions are removed. Finally, words with more than four letters are chosen. The TIMIT dataset has its default train and test split. The train and test splits contain 12261 and 3988 audio segments, consisting of 1645 and 571 unique words. Out of the 571 unique words, 227 words are in vocabulary, and the remaining 344 words are zero-shot out of vocabulary. All the audio files are segmented such that the length of the audio is 1 second.

2.4.4 Multilingual Spoken Words Corpus

Multilingual Spoken Words Corpus (MSWC) [33] is a dataset solely curated for keyword spotting. The dataset contains an isolated audio sample for words from more than 50 languages. The dataset

is collected from audio sentences collectively spoken by around 5 billion people to build research and commercial applications. The total number of samples amounts to 23.4 million, each with a length of 1 second. The total number of keywords from all the languages amounts to 340 K. The keywords are curated from the audio repository of 6000 hours. The dataset has a wide range of applications, such as speech recognition and keyword spotting to call center automation. The keyword timing in the spoken sentences is obtained by forced alignment, and then these keywords are cropped from the audio sentences. All the alignments are provided along with the dataset.

The dataset has audio samples for training and testing for previously unavailable languages. For example, the Italian and Ukrainian languages did not have audio keyword spotting datasets, and now they have around 560K and 64K audio keyword samples. The dataset is curated by processing audio samples from almost 50 languages. The dataset that had less than 10 hours of audio recordings is considered a low-resource language. When the available audio recording is between 10-100 hours, they are referred to as a medium-resource language. If more than 100 hours are available, they are referred to as a high-resource language. Examples of low-resource languages include Tamil and Vallader; examples of medium-resource languages include Czech and Dutch; examples of high-resource languages include English and French. The dataset contains 26 low-resource and 12 high, and 12 medium-resource languages, totaling 50 languages.

1. **Tamil:** The MSWC corpus [32] has spoken word segments for keywords in many languages. We selected the Tamil language to understand how the method generalizes to low-resource languages as it had very low audio samples for training. The split had a total of 190 keywords and a total of 1884 audio samples. We split the dataset into 150 keywords for in-vocabulary training and the remaining keywords for zero-shot out-of-vocabulary testing. We chose 177 samples from 28 in-vocabulary words and all the 294 samples from out-of-vocabulary words for testing.

2. **Vallader:** We choose another low-resource language, Vallader. The total number of words used for training is 233. The in-vocabulary words used for evaluating results are 66, and out-of-vocabulary words used to evaluate the results are 59 The total number of in-vocabulary and out-of-vocabulary pairs are 0.5 million. The number of in-vocabulary pairs is 0.1 million. The number of out-of-vocabulary pairs is 0.1 million.

3. **Hausa:** The final low resource language we chose for the experiment is Hausa. The total number of words used for training is 240. The total number of in-vocabulary and out-of-vocabulary words used to evaluate the performance is 77 and 61 words. The total number of in-vocabulary and out-of-vocabulary pairs is 1.8 million. The number of in-vocabulary pairs is 0.3 million. The number of out-of-vocabulary pairs is 0.5 million.

2.4.5 LRW Dataset

The Lip Reading in the Wild (LRW) dataset is videos of people talking captured in a camera with speech using a microphone. The dataset focuses on recognizing what the user has said just by observing the lip movements without using audio. We focus on a different task; we want to use the audio recorded on these videos for our keyword spotting task. Hence we extract all the audio from these videos and discard the videos for our experiments. From this dataset, we have almost over a million instances of the spoken word corresponding to 500 keywords spoken by over a thousand different speakers.

The authors have used the British television program to build this dataset. They have extracted almost 1000s of hours of talking faces covering a huge vocabulary of 1000. There were almost 1M instances for these keywords spoken by 1000 different speakers.

We have a disjoint set of samples for training, validation, and testing. The keywords list is obtained by selecting the top 500 words from the dataset's list of frequently occurring words. The length of the keywords is between 5 to 10, so they have an average spoken duration of 1 second each. Smaller words are avoided as these words provide contexts when present in sentences and avoid ambiguity. We exploit the presence of a large number of samples(1000) for each of the 500 keywords for evaluating our keyword spotting system.

2.5 Evaluation Metrics

The performance of a QbE-STD system is evaluated using statistical classification theory. We use two different metrics to evaluate our systems: Average Precision and Accuracy.

Average Precision: We have a test set and query set of cropped audio, each set has one sample of the keyword list. We want to retrieve the right sample from the test set for every sample in the query set. We use cosine similarity to compare the similarity between the representation of audio files. The performance is evaluated based on ranking metrics. We sort this set of pairs by ascending distance and compute the average precision (AP) of ranking target over non-target trials. We repeat this for each class and average the per class AP score to produce the reported mean average precision (mAP). For each keyword-clip pair, the match is considered correct if both the audio with the highest similarity contains the same keyword. The formulation is given as:

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

$$\text{precision} = \frac{|\{\text{ relevant documents }\} \cap \{\text{ retrieved documents }\}|}{|\{\text{ retrieved documents }\}|} \tag{2.3}$$

The finite sum is equivalent to

$$\text{AveP} = \frac{\sum_{k=1}^{n} P(k) \times \text{rel}(k)}{\text{number of relevant documents}} \tag{2.4}$$

$P(k)$ is the precision at cut-off k in the list. $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise.

Mean average precision (MAP) for a set of queries is the mean of the average precision scores for each query.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q} \tag{2.5}$$

where Q is the number of queries.

Accuracy: Accuracy is a metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy } = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.6}$$

Chapter 3 uses Average Precision as the evaluation metric in the similar fashion it is used in [16]. Chapter 4 Low Resource Language Adaptation uses the same AP in [16] while the Small-footprint uses the above MAP equation 2.5 evaluation setting.

2.6 Test of Statistical Significance

We draw the Critical Difference (CD) diagram. We use Friedman Test and post-hoc pair-wise Nemenyi test. The Friedman Test captures the treatment across multiple test attempts. For example, If we ask ten vendors to sell ten different items, do any of the items sell well across all the vendors? The items are the various methods tested, and the vendors are the various datasets used. And Friedman Test is a non-parametric test.

The Nemenyi test is a post-hoc test. The focus of the test is to see if any groups of data that differ after a global statistical test has rejected the null hypothesis that the performance of the comparisons on the groups of data is similar. The test makes pair-wise tests of performance.

Chapter 3

Generalized ASR representation for Keyword Spotting

3.1 Introduction

In this chapter, we discuss about the DeepSpeech2 ASR and the representation that has been used from the Deepspeech2 ASR for keyword spotting. We propose our novel method to use this ASR training for keyword spotting alongside the triplet loss function. We explain the training methodology and discuss its results. We present the qualitative and visual analysis of the representations. And finally, we discuss how this method can be extended to continuous audio.

3.2 Automatic Speech Recognition

Automatic Speech Recognition pipelines initially had hand-engineered domain features discovered over decades. Recently, We are training Deep Neural Network architectures in an end-to-end fashion for many speech processing tasks such as ASR, speaker detection, speaker diarization, etc. Such an end-to-end model replaces the hand-engineered pipelines while at the same time providing improved performance on these tasks [13]. The Amazon Mechanical Turk human workers are on-demand workers assigned to tasks such as Speech Recognition, and their performance is considered the human baseline for some tasks. DeepSpeech2 ASR performs better than Amazon Mechanical Turk human workers' performance on several benchmarks. DeepSpeech2 works for multiple languages with little changes in the architecture and can be deployed quickly and hence production ready. The Deep Neural Network-based ASRs are becoming more prominent and are a ubiquitous way for speech recognition. Due to the extensive research in the direction of DNN, the various components of the ASR system, such as collecting the training datasets, and training these models with various architecture designs, have evolved easily.

The traditional ASRs depend on hand-engineered features for each of the components. The acoustic speech signal has numerous information in it. These components involve feature extraction, building acoustic and language models with speaker adaptation, etc. Such a system is difficult to adapt to a new scenario, and since each component depends on one another, failure in one module could harm the

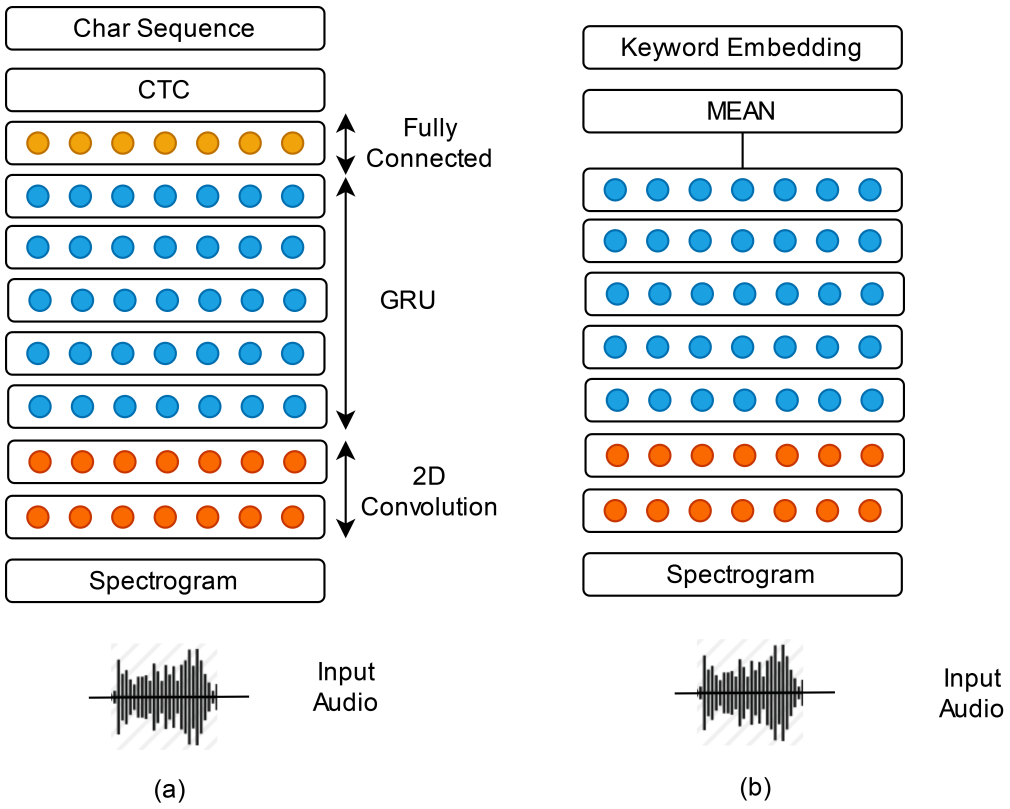


Figure 3.1: (a) DeepSpeech2 ASR Architecture with 2CNN, 5GRU and fully connected layer (b) Keyword Embedding is obtained by taking the average the output of the final GRU layer.

others. Such a system suffers from adapting to new languages and systems. Adapting to new scenarios and applications has become a requirement to provide acceptable accuracy. The method of recognizing speech by humans is totally different. We have the innate ability to understand the words spoken in our childhood through our feedback mechanism. We later adopt these methods to recognize speech in difficult environments. Other tasks like reading and writing help to understand the words spoken much more clearly. Hence we do not explore hand-engineered features but focus on DNN-based methods for our keyword spotting task.

We show that the focus on building ASR models can be an advantage in modifying such systems for recognizing specific keywords. We believe the system we propose to build can handle new keywords efficiently with small changes and can adapt to new languages with minimal intervention. Our keyword spotting system methodology allows us to compete with state-of-the-art systems in several low-resource test languages: Tamil, Hausa, Vallader, and English.

Table 3.1: Intermediate features dimension of the data in DeepSpeech2 layers. The keyword embedding will be of dimension 1600 from final gru5 layer.

Layer No.	DNN Type	Input Features Size	Output Features Size
1	cnn1	160	2560
2	cnn2	2560	1280
3	gru1	1280	1600
4	gru2	1600	1600
5	gru3	1600	1600
6	gru4	1600	1600
7	gru5	1600	1600
8	fc	1600	29

3.3 Methodology

3.3.1 Pretrained ASR with CTC loss

We use DeepSpeech2 ASR for this work. DeepSpeech2 is an end-to-end trained Deep Neural Network that converts audio to text. This model is pretrained by OpenSeq2Seq framework [28]. We use audio spectrograms as the input to this model. A Hanning window of 20 ms and a stride of 10 ms is applied to the audio input while generating the spectrograms. The audio sampling rate is 16kHz, and our input dimension is 160.

The Convolutional Neural Network forms the first two layers of the DeepSpeech2 architecture with an output feature map of 32. The kernels used in these two layers are of the size 41x11 and 21x11, respectively. A convolution kernel of 41x11 has a size of 41 in the time domain and 11 in the frequency domain. The kernel stride is 2 in the time domain on both the convolutional layers, and in the frequency domain, the stride is 2 in the first layer. These convolutional layers output features of dimensions 2560 and 1280 after the first and second layers. Gated Recurrent Units (GRU) follow these convolutional layers. There are 5 bidirectional GRU with a hidden state dimension of 800. A fully connected layer follows the GRUs that map the embeddings to a softmax output of 29. The output classes correspond to the number of characters with a special blank symbol. Batch normalization and a ReLU non-linearity follow each convolutional and recurrent layer. We train this network with the CTC loss function. The ASR Model used in this work is shown in 3.1. The CTC loss function is given as:

$$\mathcal{L}_1 = -\log p(l|x) \tag{3.1}$$

where the probability of a label sequence l given an input sequence x is defined as:

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} \prod_{t=1}^{T} \mathrm{ASR}_t^K(\mathbf{x})\left[\pi_t\right] \tag{3.2}$$

where $\mathcal{B}$ removes the repeated symbols and blanks, $\mathcal{B}^{-1}$ is its inverse image, $\mathcal{T}$ is the length of the label sequence l, and $\mathrm{ASR}_t^K(\mathbf{x})\,[\pi_t]$ is unit j of the model output from the top softmax layer at time t, interpreted as the probability of observing label j at time t.

The acoustic word embedding for the keyword audio segments are obtained from the output of the final GRU layer. The output dimension of the final GRU layer is 1600 for each time instance. We take the average of this representation across time dimension to obtain a fixed representation of size 1600 for variable length audio. This is shown in the figure 3.1(b). These features can also be considered bottleneck features.

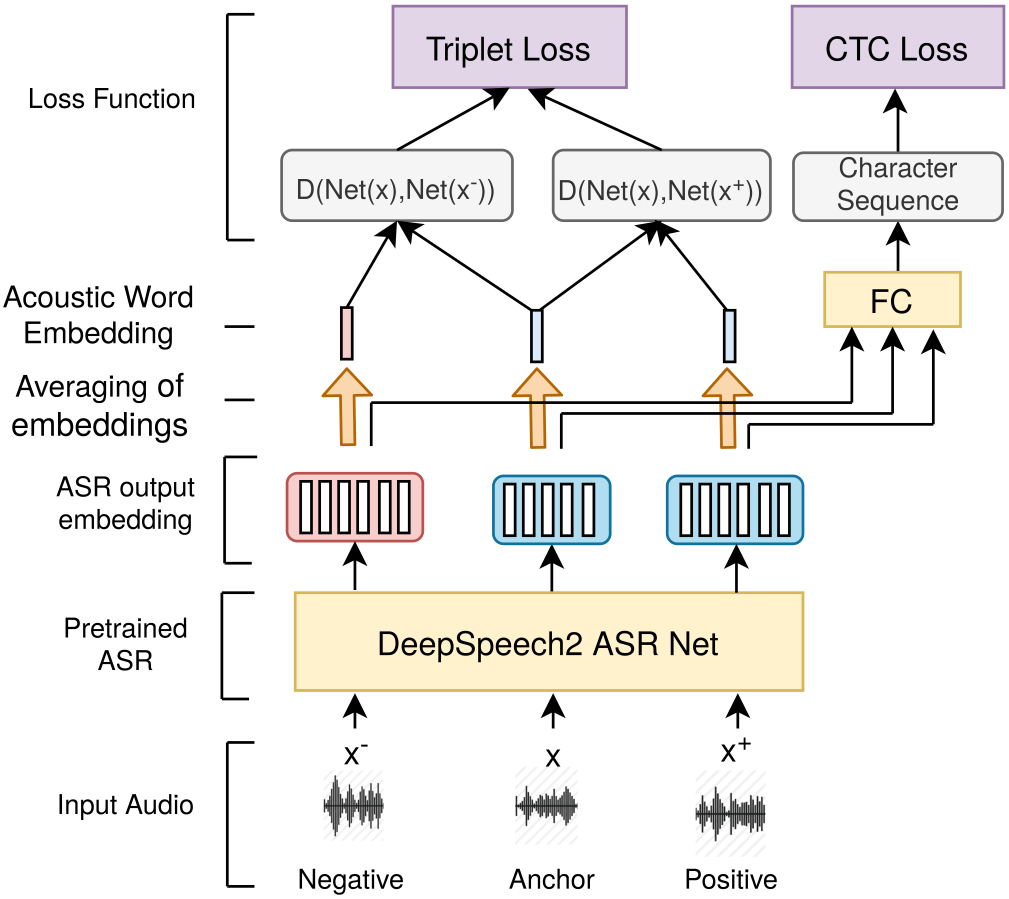


Figure 3.2: Generalized Keyword Spotting Embedding Representation: A pretrained DeepSpeech2 ASR Net is trained with CTC loss and triplet loss for obtaining the acoustic word embeddings used for Keyword Spotting

3.3.2 Adding Triplet loss with CTC loss

A triplet network has three copies of a feed-forward neural network with shared parameters as described in [21]. The triplet network intakes three samples: anchor, positive and negative for each of the

network copy and is denoted as $\mathbf{x}$, $\mathbf{x}^+$ and $\mathbf{x}^-$ respectively. The network intakes these three samples and produces three embeddings for each sample. Then we calculate the distance between the embeddings of the anchor with the embeddings generated for the other two samples. We denote the distance method as D. The triplets are denoted as $\mathbf{x}$, $\mathbf{x}^+$, $\mathbf{x}^-$. We denote $\mathbf{x}$ as the anchor input sample, $\mathbf{x}^+$ represents the positive sample that is the same class as the anchor, and finally $\mathbf{x}^-$ represents the sample from a different class. We denote the representation obtained from the network for each of these audio samples as $\mathrm{Net}(\mathbf{x}), \mathrm{Net}(\mathbf{x}^+), \mathrm{Net}(\mathbf{x}^-)$. We express the output of the triplet network as:

$$\text{TripletNet } \left(\mathbf{x}, \mathbf{x}^-, \mathbf{x}^+\right) = \begin{bmatrix} D\left[\mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^-\right)\right] \\ D\left[\mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^+\right)\right] \end{bmatrix} \in \mathbb{R}_+^2 \tag{3.3}$$

In this scenario, we average the stack of the frame-level embeddings $\mathrm{Net}(\mathbf{x}_t)$ from the output of the embedding model $\mathrm{Net}(\mathbf{x})$. While in [17] the embeddings of the last frame $\mathrm{Net}(\mathbf{x}_T)$ from the embedding models output are used.

The embedding sub-network receives each of the input samples ($\mathbf{x}$, $\mathbf{x}^+$, $\mathbf{x}^-$) independently. The pairwise distance between the positive samples pair $D\left[\mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^+\right)\right]$ is computed. This is computed between the anchor $\mathrm{Net}(\mathbf{x})$ and the positive $\mathrm{Net}\left(\mathbf{x}^+\right)$ sample. Then the pairwise distance between the negative samples pair $D\left[\mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^-\right)\right]$ is calculated. This is computed between the anchor $\mathrm{Net}(\mathbf{x})$ and the negative $\mathrm{Net}\left(\mathbf{x}^-\right)$ sample. The cosine distance is used in [17] as a distance metric. We use the same cosine similarity as the distance metric. We pass these two distances into the comparator. The loss function used to train the triplet network is given as:

$$\begin{aligned} \mathcal{L}_2\left(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-\right) = \max(D[\ \mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^+\right)\right] \\ -D\left[\mathrm{Net}(\mathbf{x}), \mathrm{Net}\left(\mathbf{x}^-\right)\right] + \alpha, 0\Big) \end{aligned} \tag{3.4}$$

where the distance between the anchor and the positive sample is minimized, while the distance between the anchor and the negative sample is maximized, α denotes the margin between the positive and negative sample, and D denotes the distance measurement. We then append the triplet loss with the calculated CTC loss for backpropagation (Figure 3.2). The total loss of the model can be given as follows: (λ_1 and λ_2 are the weights assigned to the CTC and triplet loss)

$$\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2 \tag{3.5}$$

3.4 Training

HyperParameters for TIMIT training: We used the OpenSeq2Seq [28] DeepSpeech2 pretrained model for our proposed KWS model. The raw signal is applied with data augmentation operations such as adding noise and speed perturbation. The model is trained with a polynomial decay learning rate with power 0.5 and uses an initial learning rate of $1 \times e^{-3}$. In addition, an L2 penalty of $1 \times e^{-5}$ is used. We use a batch size of 32 and the Adam optimizer [26]. We train the model up to 500 epochs and

select the last model to report the accuracy on the test set. The value of λ_1 is 1 and λ_2 is 20. We keep the hyperparameters constant for all the experiments in this work. We also train a few other models: with CTC loss and triplet loss separately after loading the pretrained weights, with CTC and triplet loss together without loading pretrained weights. We also measure the performance of the pretrained model without any training.

Triplet Mining: The triplets for the triplet loss are mined online for each batch rather than mining it beforehand. A batch-hard strategy is used where the distance between all the pairs in a batch is calculated, and then for each reference sample, the positive sample, which lies farthest, and the negative sample, which lies closest, are considered for calculating the triplet loss. Each batch has more than one sample for each keyword to help learn the similarity between the positive pairs, and each batch has more than one keyword, with each keyword on default having an equal number of samples.

Baseline Training: We use the Multi-view Recurrent Neural Acoustic Word Embeddings system implemented by [17] as a baseline. They have approached the problem of bringing similar audio keywords into the same embedding space and bringing the audio and its text representation into the same embeddings space. They have shown the latter as better among the two. They have used two LSTM networks; one takes audio as the input while the other takes text as input and tries to bring these two representations closer. They have used the contrastive loss as the training loss, which tries to bring positive pairs closer and negative pairs farther. We train their Multi-View code for 1000 epochs with their best performing objective for baseline comparison.

Fixed Vocabulary Training: We compare our model's performance for classification tasks on the Speech Commands dataset using kNN. We train our architecture with this dataset and obtain the embeddings for the test set that are later classified using the kNN algorithm. We train our model for 300 epochs for this task. We use [49] to compare the results. They have used a triplet loss-based embedding trained on same dataset and used a variant of kNN to show accuracy performance.

We perform additional end-to-end training with the speech commands dataset. We add additional training layers to do end-to-end training. We attach a fully connected layer to the DeepSpeech2 backbone to reduce the model dimension from 1600 to 35 classes. We then add a softmax layer to the fully connected layer to produce probabilities for each class. We show this in Figure 3.3. The model is trained with a cross-entropy loss function for 300 epochs. We observe the top-1 accuracy results from softmax probability outputs for this experiment. We perform the fixed vocabulary training to compare our method's performance against the fixed vocabulary keyword classification task.

3.5 Results

TIMIT evaluation: For each pair of words from the test data set, the word discrimination system decides whether the pair contains the same or different words. We obtain the embeddings from the trained model for the test data set and decide whether a pair of audio samples are similar or dissimilar. We compare the predicted labels with the actual ground truth labels to estimate average precision. Thus we measure the overall system performance. We calculate the distance between the acoustic embeddings

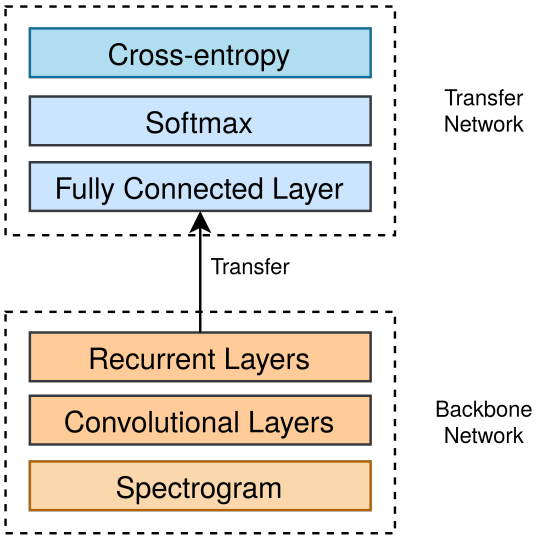


Figure 3.3: Blocks depicting ASR network from pretrained model attached to a Transfer network to achieve fixed-vocabulary KWS training.

Table 3.2: Average Precision evaluation for Generalized Keyword Spotting model and Multi-View model on TIMIT dataset for ALL(both IV and OOV), in-vocabulary(IV) and zero-shot out-of-vocabulary(OOV) test keywords samples.

Model	ALL	IV	OOV
Multi-View [17]	0.328	0.528	0.218
Pretrained ASR	0.848	0.911	0.803
Pretrained + CTC	0.903	0.954	0.750
Pretrained + Triplet	0.936	0.975	0.700
CTC + Triplet	0.965	0.990	0.800
Pretrained + CTC + Triplet	**0.971**	**0.991**	**0.843**

for each pair of words from the test set. A threshold was applied to determine if the pair represents the same or different words. The total number of pairs in the test set was 7.9 million for the TIMIT dataset. This includes both in-vocabulary and out-of-vocabulary words. A precision-recall curve was obtained from which the AP was estimated by sweeping the threshold value. We calculate the AP value for all the trained models on the TIMIT dataset and report the results in Table 3.2. It can be observed that the performance of the model that was trained with loading pretrained weights and CTC and triplet loss has significant performance improvement over Multi-View and other models. We also observed that the model with Triplet + CTC loss was able to converge relatively faster than the model with only triplet loss.

The performance of in-vocabulary and out-of-vocabulary words is calculated separately for both Multi-View and our model and are reported in Table3.2. The total number of pairs used to calculate AP in in-vocabulary is 1.7 million, and out-of-vocabulary is 2.2 million.

Table 3.3: Accuracy evaluation of our model on classification task trained on Google Speech Commands dataset V2. The number of target classes is 35. We use our Pretrained+CTC+Triplet embeddings with kNN for accuracy comparison. We additionally show the performance of end-to-end training on our DeepSpeech2 method.

Model	Accuracy
Pretrained ASR	85.7
Attention RNN [6]	93.9
AST [11]	98.1
Res15 [49]	97.0
Pretrained + FC-Softmax	94.4
Pretrained + CTC + Triplet	96.2

Table 3.4: The Effects of various combinations of CTC, triplet and Cross-entropy loss functions on Fixed Vocabulary Training. The accuracy of the models that has used cross-entropy is evaluated directly from softmax layer while the other models are evaluated through KNN.

S.No.	CTC Loss	Triplet Loss	Cross-Entropy Loss	Accuracy(Softmax)	Accuracy(KNN)
1	Yes	Yes	Yes	96.3	NA
2	Yes	Yes	No	NA	96.2
3	Yes	No	No	NA	95.5
4	No	Yes	No	NA	86.1
5	No	No	Yes	94.4	NA
6	Yes	No	Yes	96.2	NA
7	No	Yes	Yes	**96.5**	NA

Speech Commands Evaluation: We obtain all the embeddings for the train and test set from our model trained on Google Speech Commands dataset V2 (35 classes) with triplet and CTC loss. We then apply kNN to classify the test set to calculate accuracy. We compare our results with [49, 6, 11]. We achieve a classification accuracy of 96.2%. We have tested kNN for several values of k and have found that for the speech commands dataset, the best performing value for k is 7. For the KWS application, larger datasets occupy a lot of memory for the kNN part of the model. Our approach has the potential to achieve competitive results on classification tasks through kNN for small keyword sets. The end-to-end training on the same DeepSpeech2 backbone with Fully Connected and Softmax layer achieves 94.4%. Our method performs slightly better than end-to-end training on the same backbone showing the method is able to generalize to strong cross-entropy loss-based methods. The results are shown in Table 3.3.

3.6 Analysis

We show the nearest neighbors for four sample queries in the test set in Table 3.5. Our model performs better than the pretrained model for the keyword *magnetic* and the keyword *livestock* even though it is unseen during training even, while the triplet model has missed *livestock*. The model improves the

representation of keyword *getting* from the pretrained model after seeing it. The model preserves the representation for the keyword *program* while the triplet model fails to retrieve it. It can be observed that the keywords that lie closer have similar phonemes for our model in all the cases in Table 3.5.

Table 3.5: Keywords retrieved by the Pretrined ASR vs Pretrained + triplet vs Our Pretrained + CTC + Triplet model on OOV and IV keywords. Bold represents query and the closest sample belonging to the same keyword.

Model	Query: Top 3 unique nearest neighbours
Pretrained	*magnetic*: economic, barometric, diagram
Pre + triplet	*magnetic*(oov): **magnetic**, money, nobody
Ours	*magnetic*(oov): **magnetic**, money, barometric
Pretrained	*livestock*: **livestock**, stopwatch, stockings
Pre + triplet	*livestock*(oov): muscular, catastrophic, extra
Ours	*livestock*(oov): **livestock**, extra, electron
Pretrained	*getting*: looking, would, meeting
Pre + triplet	*getting*(iv): began, **getting**, bedroom
Ours	*getting*(iv): **getting**, heating, eating
Pretrained	*program*: **program**, arriving, problem
Pre + triplet	*program*(iv): causeway, cranberry, corduroy
Ours	*program*(iv): **program**, problem, popular

3.6.1 Statistical Significance Testing

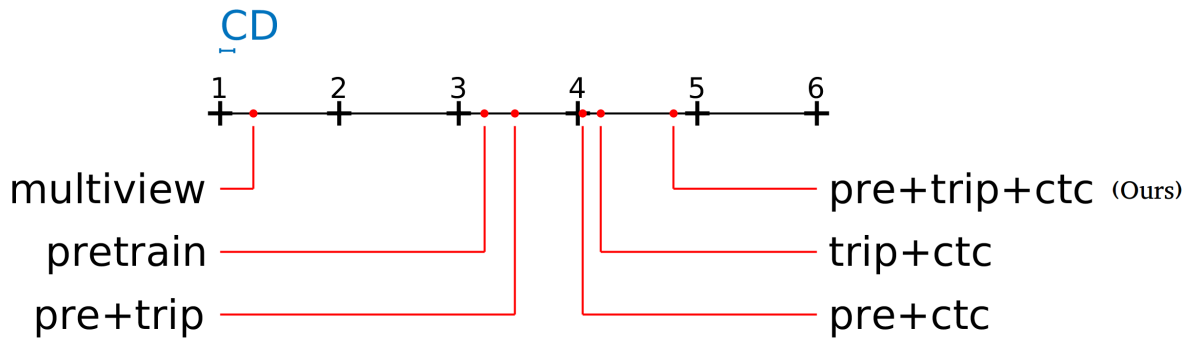


Figure 3.4: Statistical Significance Test depicting the efficient performance of our model. Our method with Pretrained + Triplet + CTC on the far right denotes that the method performs better compared to others.

We take the outcomes of multiple treatments of the data over multiple observations and create the Critical Difference (CD) diagram. For example, we compare the performance of multiple methodologies with different datasets to see the effect of each method. The treatments across multiple test attempts are

tested with the Friedman test. We then perform a post-hoc pairwise Neymani test to find the data groups that differ. The x-axis denotes the average rank, and each tested methodology lies on this 1-D graph. The null hypothesis is that there is no difference in the proposed method. If a horizontal line below the x-axis connects two methods, they are not statistically significant. If they are not connected, then they are statistically significant. Our Pretrained + CTC + Triplet method's rank is higher than all other methods. Hence our method is statistically significant over all other methods. The CD is 0.09.

3.6.2 Distance Distribution between embeddings of Positive and Negative Pairs

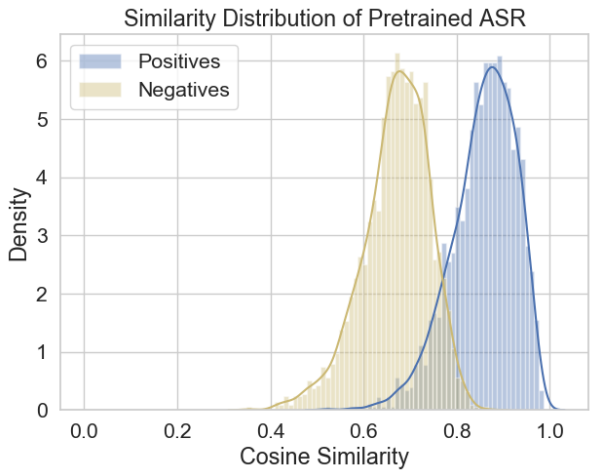


Figure 3.5: Graph showing overlapping distributions of positive and Negative pairs that lie close to each other, causing non-clear boundaries to do keyword spotting.

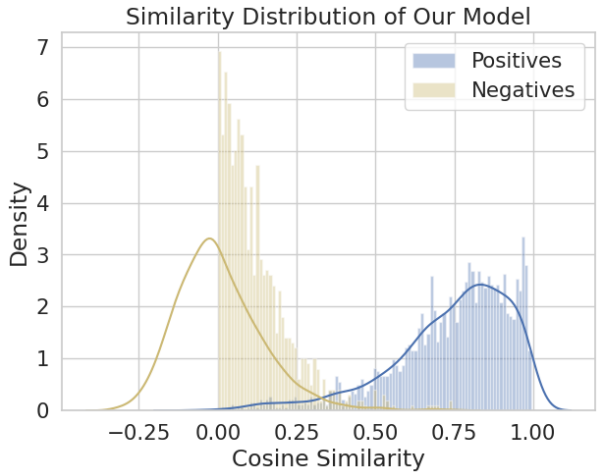


Figure 3.6: Graph showing separated distributions of positive and Negative pairs that enable to spot keywords with higher confidence.

We take 1000 triplets by choosing 1000 positive and negative pairs. We calculate the distances between the AWE of these samples. All the models are tested on the same pair of audio samples to show uniformity during evaluation. The results are shown on the TIMIT dataset. The similarity distribution of positive and negative pairs of the pretrained model is shown in Figure 3.5. It can be observed that both the curves lie close to one other. The similarity distribution of samples of our model is shown in Figure 3.6. Here it can be observed that the positive distribution and the negative distribution are well separated. From Figure 3.6, it can be observed that the optimal similarity threshold for a pair of audio samples to be of the same class is where both the positive and negative samples' boundaries meet which is around 0.3.

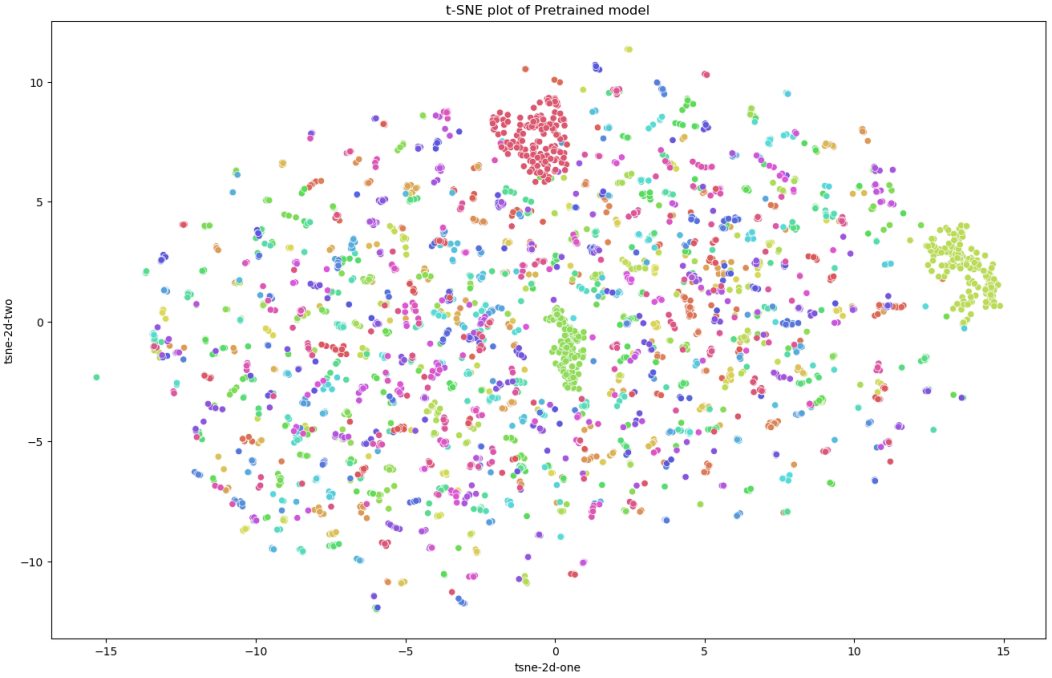
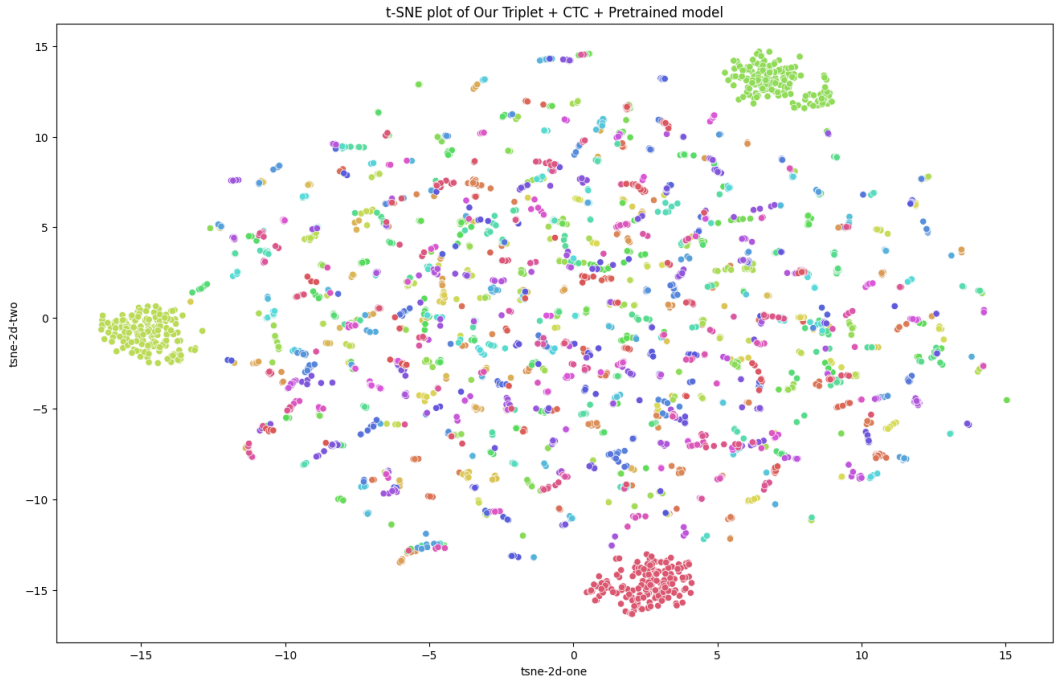


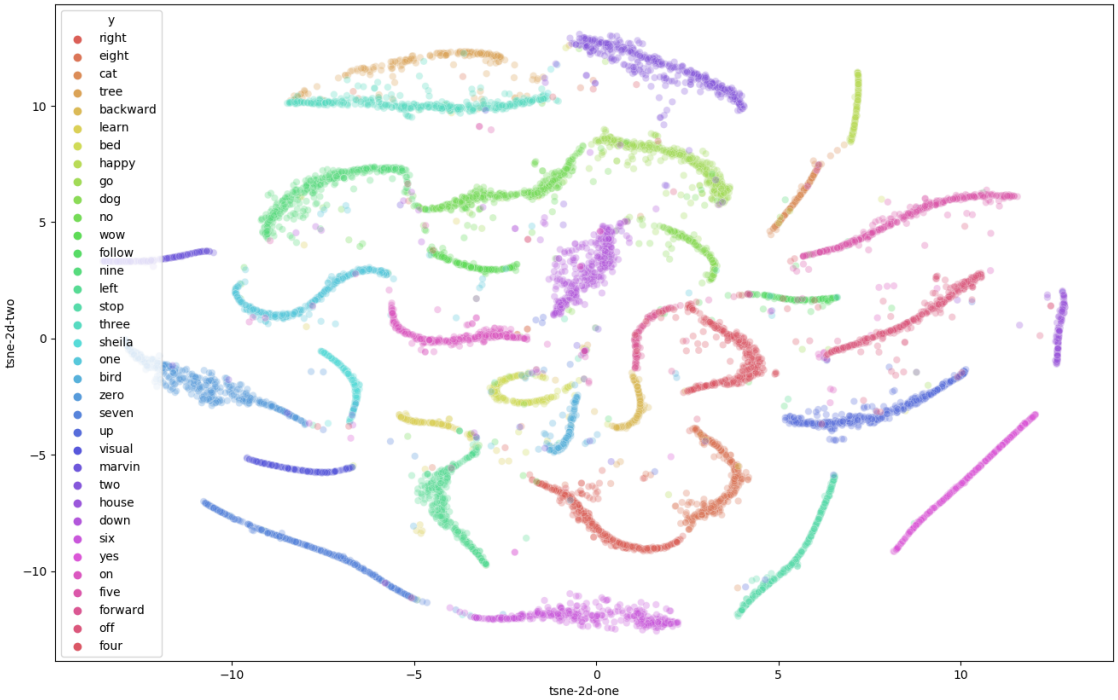
Figure 3.7: tSNE Plot of TIMIT keywords from Pretrained Model.

3.6.3 t-SNE Plots

We use the tSNE plot to visualize the learned embedding space. We show the tSNE plots for the pretrained ASR model and our model trained on TIMIT and google speech commands dataset. The resulting plots are presented in Fig. 3.7 3.8a, 3.8b, respectively. We observe from the diagram that samples from similar classes lie closer and samples from different classes lie farther. The number of training samples is very few for many training classes. The classes with many training samples have more efficiently clustered from other classes. The distance between these clusters indicates the phonetic similarity between these keywords.



(a) tSNE Plot of TIMIT keywords from Our (Pretrained + CTC + Triplet) model. Comparison between the plot depicts the better embedding representation of our model over the pretrained model as the intra-distance is lower and inter-distance is higher between keywords in our model.



(b) tSNE Plot of Google Speech commands keywords from Our(Pretrained + CTC + Triplet) Model depicting keywords being grouped together, denoting the efficiency of the representation.

Figure 3.8: tSNE plots of Our Model on TIMIT and GSC dataset

We need to analyze further to find the phonetic similarity between the keywords in the tSNE plot. The amount of training data to their corresponding plots can give insights into adapting the setting to low resource languages. The classes are better clustered in Google Speech Commands (GSC) dataset when compared to TIMIT dataset and is because the number of classes is lesser in the GSC dataset than in the TIMIT dataset.

3.7 Extension to continuous audio

When we want to spot keywords in continuous audio, we use a sliding window approach. Here continuous audio represents an utterance that has multiple spoken words in it. This is illustrated in Fig. 3.9.

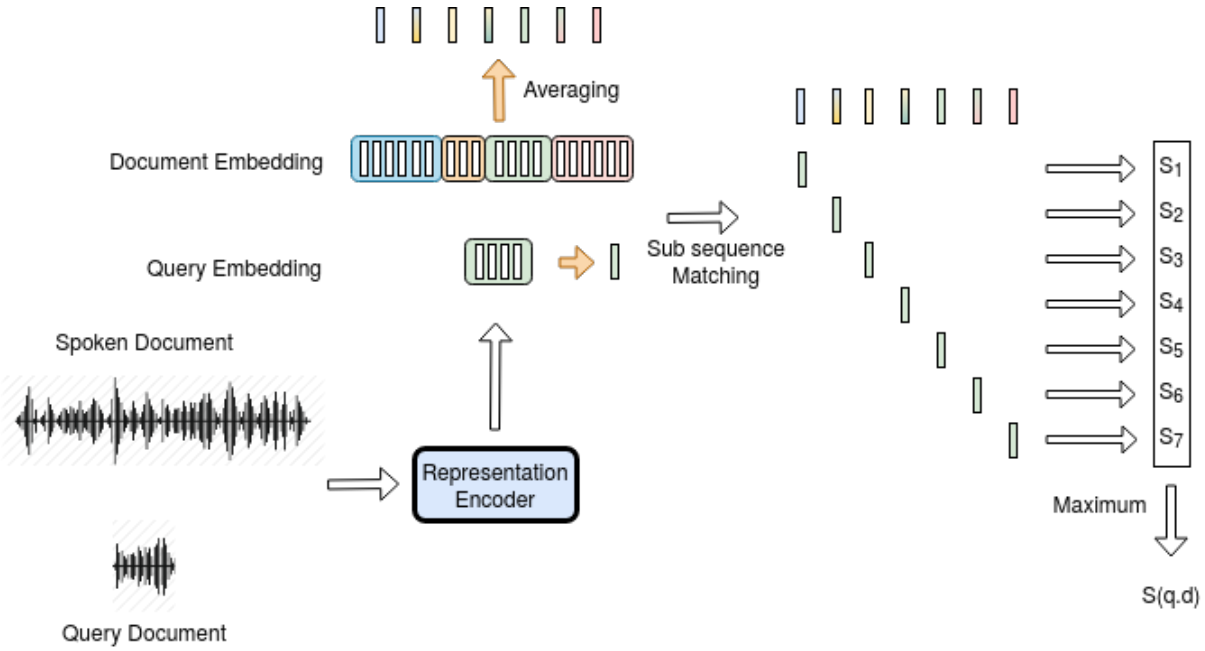


Figure 3.9: Schematic Diagram of methodology to extend our kws approach to continuous audio: A sliding window approach is used to encode embeddings at regular interval which are then compare with keywrod embedding. The maximum similarity value among the computed similarities is acknowledged as the similarity score between the audio Embedding and the query.

Given frame sequences of a spoken query and a spoken document, Representation Encoder from our model can represent these sequences as embeddings. These embeddings are then averaged for every window of 0.5 second to give the fixed embedding d_n with a window shift of 0.25 second to give d=d_1,d_2,...,d_n

The query word embedding is matched with embedding obtained by averaged at each window of the document embedding sequentially and their similarity score is obtained. This is the sub sequence matching.

$$S(\mathbf{q}, \mathbf{d}) = \max\left(S_1, S_2, \ldots, S_n\right) \tag{3.6}$$

$$S_n = \mathrm{Sim}\left(\mathbf{q}, \mathbf{d}_n\right) \tag{3.7}$$

Cosine Similarity is used in the similarity measure. The relevance score $(\mathbf{q}, \mathbf{d})$ between the query and the document is then the maximum out of all $\mathbf{d}_n$'s obtained in the Equation. The optimal window size for KWS detection varies per utterance based on speech rate, noise, and adjacent utterance. This is shown in Fig. 3.9

Chapter 4

Low Resource Language Adaptation and Small-Footprint KWS

In this chapter, we discuss the transliteration used to convert low-resource language script to English. Transliteration is used to convert low-resource language scripts to English. We share the details of the low-resource language transliteration and training procedure. We evaluate and compare our results of various methods. We discuss results on three low-resource languages. Then we move towards small footprint keyword spotting using distillation, whose focus is on reducing the size of the model. We discuss the effect of various distillation loss functions on keyword spotting tasks. Finally, we evaluate the results of small footprint ASR against distillation methods.

4.1 Transliteration

Transliteration refers to the conversion of characters from one language to another. The focus of transliteration is not the sound or pronunciation but the most accurate way of representing characters from one language with the characters from another. Transliteration is different from translation. Translation tries to replace words and sentences from one language to another without concern for pronunciations, and the resulting character sets are different for source and different. On the other hand, transliteration tries to convert characters from one language to another with minimal disturbance to pronunciation. For example, In the Hindi language, the transliteration of ”namaste” is नमस्ते.

It is normal for transliteration to have more than one representation in target language when converting from the source language. For example, the Hindi text नमस्ते can be both converted to ”namaste” or ”nemaste” due to the fuzzy nature of how the word is pronounced by each individuals.

We created a dataset of transliteration text for all the Tamil keywords in the MSWC dataset. This transliteration is created by a Tamil language expert whose native language is Tamil. All Tamil language keyword scripts are transliterated to English such that keywords sound as similar as possible to ASR's English character vocabulary. Though few phonemes in the Tamil language don't exist in English and several Tamil phonemes sound differently, these characters are converted to the closest sounding English

characters. We also share our transliteration dataset of Tamil keywords in the MSWC dataset here[1]. We are sharing few examples of the transliteration below.

4.2 Zero Shot Out-Of-Vocabulary Generalization

This method is easily generalized to Zero-Shot Out Of Vocabulary keywords. This is possible because we operate at the embedding space directly. We obtain the acoustic word embedding for any audio segment from our model as shown in Figure 3.2. The audio sample can be from keywords the model has or has not seen. During prediction, we convert all the sample audio keyword files into this embeddings space. Then, for every test audio sample, we first obtain the word embedding by passing it through the model. Then we compare the test audio embedding with all the keyword embeddings using cosine similarity. When the similarity is above a certain fixed threshold, we say the keyword and the test audio is a match.

While the cosine similarity comparison method has been the key, the other generalization methods use a modified version with one or few train samples for the keywords. In the work [23], the model requires configuration with a sample keyword audio and respective threshold values for prediction. While our method does not require configuration for each keyword and the threshold is constant. In the work [31], a keyword-specific classification layer is trained with five training samples and uses this classifier for predicting those words. In contrast, our method does not require any sample for out of vocabulary zero-shot prediction. Other works have focused on fixed vocabulary keyword spotting and require retraining to add new classes.

4.3 Low resource Language Training and Evaluation

Low resource Language Training: The model is trained on MSWC Tamil split for 1000 epochs on the training set. We train three models: with only triplet loss, triplet and CTC loss with pretrained weights, and without pretrained weights. The performance is measured on in-vocabulary and zero-shot out-of-vocabulary words from the test split from the same dataset. The CTC loss is applied to the transliterated text during training.

Tamil Language Evaluation: We calculate the AP value for the generalized keyword spotting models for low resource language on the Tamil MSWC test dataset, and the results are reported in Table 4.1. It can be observed that pretraining and fine-tuning with CTC and triplet loss has helped the model in achieving better performance in the Tamil language. Although pretraining and fine-tuning with only triplet loss has not helped in generalization. The total number of pairs in the test set was 110K. The total number of pairs used to calculate AP in in-vocabulary is 15K, and out-of-vocab is 43K. We hereby share this as the initial benchmark for low-resource zero-shot out of vocabulary keyword spotting.

[1] https://github.com/Kirandevraj/GeneralizedKWS

அஞ்சாத-anjadha	அண்ண-anna
அரச-arasa	ஆதரிக்க-aadharikka
அழுத-aludha	ஆழ்ந்த-aalndha
இங்குள்ள-ingulla	இணையற்ற-inayatra
இந்திய-indhiya	இருந்த-irundha
இல்லாத-illadha	இழந்த-ilandha
ஈடற்ற-edattra	உடுத்த-udutha
உண்மையான-unmayana	உளமுற்ற-ulamutra
என்ற-endra	எண்ணிய-enniya
கல்யாண-kalyana	கட்டிய-kattiya
கிடந்திட்ட-kidandhitta	கற்க-karka
கேட்க-kaetka	காங்கேய-kaangeya
கொரகொர-korakora	காத-kaatha
சென்ற-sendra	தவறான-thavraana
நன்றாக-nandraaga	நாடிய-naadiya
நிச்சயமாக-nichayamaga	நீங்க-neenga
படிக்க-padikka	மறைவாக-maraivaga
பிறந்த-pirandha	ரொம்ப-romba
பொல்லாத-polladha	வாய்த்த-vaaitha
மிந்த-mindha	வைக்க-vaikka

Figure 4.1: Few samples from the Tamil Transliteration dataset

Table 4.1: Average Precision evaluation for Generalized Keyword Spotting Model for Low Resource Language trained on Tamil split of MSWC dataset. The columns indicate ALL(both IV and OOV), in-vocabulary(IV) and zero-shot out-of-vocabulary(OOV) test keywords samples.

Model	ALL	IV	OOV
Pretrained + Triplet	0.030	0.061	0.057
CTC + Triplet	0.161	0.300	0.253
Pretrained + CTC + Triplet	**0.295**	**0.587**	**0.321**

Vallader Language Evaluation: The total number of words used for training is 233. The IV words used for evaluating results is 66 and OOV words used to evaluate the results is 59 The total number of IV and OOV pairs is 0.5 million. The number of IV pairs is 0.1 million. The number of OOV pairs is 0.1 million.

Table 4.2: Average Precision evaluation for Generalized Keyword Spotting Model for Low Resource Language trained on Vallader split of MSWC dataset. The columns indicate ALL(both IV and OOV), in-vocabulary(IV), and zero-shot out-of-vocabulary(OOV) test keywords samples.

Model	ALL	IV	OOV
Pretrained + Triplet	0.025	0.039	0.048
CTC + Triplet	0.313	0.450	0.536
Pretrained + CTC + Triplet	**0.383**	**0.545**	**0.566**

Hausa Language Evaluation: The total number of words used for training is 240. The total number of IV and OOV used to evaluate the performance is 77 and 61 words. The total number of IV and OOV pairs is 1.8 million. The number of IV pairs is 0.3 million. The number of OOV pairs is 0.5 million.

Table 4.3: Average Precision evaluation for Generalized Keyword Spotting Model for Low Resource Language trained on Hausa split of MSWC dataset. The columns indicate ALL(both IV and OOV), in-vocabulary(IV) and zero-shot out-of-vocabulary(OOV) test keywords samples.

Model	ALL	IV	OOV
Pretrained + Triplet	0.082	0.135	0.187
CTC + Triplet	0.408	0.552	0.604
Pretrained + CTC + Triplet	**0.428**	**0.554**	**0.634**

It can be observed that our method performs efficiently when compared with other methods in all the three low resource languages both in in-vocabulary and out-of-vocabulary scenario. Hence we can understand that the transliteration has contributed at a very high scale when compared to triplet loss functions in low resource scenario.

4.4 Small Footprint KWS through Knowledge Distillation

We focus on building a small footprint KWS to be able to run this keyword spotting application on a smaller device. We approach the small footprint task by knowledge distillation in the form of teacher-student network training to reduce the size of the model while retaining the performance. The performance of a neural network increases with an increase in the number of trainable parameters for complex tasks. However, in such scenarios, there is an additional constraint of increased computing resources and the training data required to train such networks efficiently. To overcome this issue, we focus on building a network with lesser parameters on low computational resources and lesser training data using Knowledge Distillation (KD) approaches. Knowledge distillation involves transferring knowledge from a larger teacher network to a smaller student network. KD helps the student network to learn with lesser computational resources and training data while training faster. The primary techniques to perform KD involve either transferring class probabilities or the hidden representation between the teacher and the student networks.

For classification tasks, the final layer of the network is usually softmax. The softmax layer assigns the probability values to each class for a particular input. The sum of these probability values is 1. These probability values are usually preferred in KD compared to one hot encoding since the probabilities have additional information about other classes that are close and farther. Hinton et al. [20] use the KL Divergence loss to perform KD between the softmax layers of teacher and student. KD in the Speech Recognition system happens at the softmax layer predicting character labels at each input frame other than the one-hot encoding of the character labels. However, the hidden representations hold the knowledge used in the KD process to train a student's hidden layer representation.

We show the schematic diagram of knowledge distillation in Figure 4.2. The representation between the teacher and student is compared to calculate the loss. This is denoted as $\mathcal{L}_{RKD}$. We explore three main knowledge distillation methods to reduce the size of the student model, namely Mean Squared Error loss, KL Divergence Loss, and CosEmbedding loss. We have discussed these loss functions in the following sections.

4.4.1 MSE Loss

It has been observed that MSE loss helps in ASR distillations [51]. We explore the same for Keyword Spotting.

Let us denote the hidden representation of the i-th and j-th layer of the teacher and student network as $y^{(i)}$ and $\hat{y}^{(j)}$. Here we assume that both the models have similar architecture. The teacher model in the speech recognition system trained with a CTC framework accepts the speech signal x as input. The hidden layer output is represented as $y^{(i)}(x) \in R^{T \times D_t}$, where T represents the total number of frames with the hidden layer dimension D_t. Similarly for the student model, the hidden layer is represented as $\hat{y}^{(i)}(x) \in R^{T \times D_s}$, where D_s denotes the hidden layer width. The hidden layer dimension D_t and

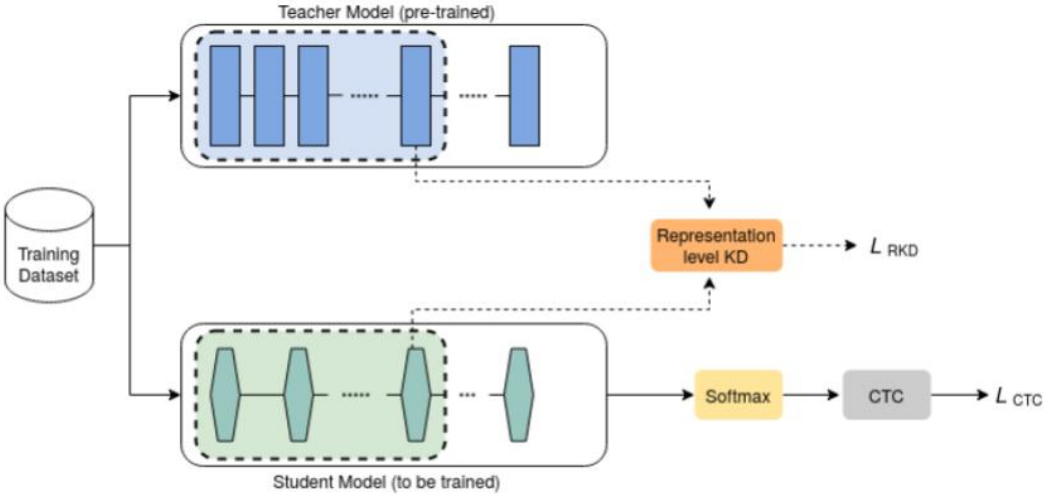


Figure 4.2: Schematic diagram of Knowledge Distillation for Small footprint KWS. We experiment with MSE, KL Divergence and CosEmbedding Loss function at the representation level KD to obtain small footprint keyword spotting model.

D_s from the teacher and student network are similar. We apply the MSE loss between the teacher and student representation as :

$$\mathcal{L}_{RKD} = \sum_{x \in Z} \sum_{(i,j) \in \mathcal{I}} \left\| \left(y^{(i)}(x) - \left(\hat{y}^{(j)}(x) \right) \right) \right\|_2^2 \tag{4.1}$$

We apply the MSE loss for distillation purposes and observe the performance of the student embeddings for keyword spotting tasks.

4.4.2 KL Divergence Loss

We apply the KL divergence loss as the objective function of KD on the probability distribution between the teacher and student network with a scaling parameter. Kullback-Leibler divergence measures the difference in the probability distribution of two networks and is a distance measure but not a metric. We estimate an approximate distribution from a true distribution in Bayesian theory. In the KL divergence measure, we try to measure the approximate distribution $\hat{y}$ from the true distribution y. We use the same KL Divergence loss to build a light weight keyword spotting model. The expression of KL divergence loss can be given as:

$$\mathcal{L}_{RKD}(\hat{y}||y) = \sum_{c=1}^{M} \hat{y}_c \log \frac{\hat{y}_c}{y_c} \tag{4.2}$$

Here $\hat{y}$ is the student models predictions, and y is the teacher models predictions. M is the number of classes, $\hat{y}_c$ is the logit value of class c from the student network, and $\hat{y}_c$ is the logit value of class c from the teacher network. Here a softmax function is applied on the embeddings output and then KL Divergence loss is applied on these softmax intermediate layers output and the c corresponds to the size of the intermediate dimensional vectors.

4.4.3 Cosine Embedding Loss

This is similar to the triplet loss used in the previous section, where cosine distance is used to measure the distance between the teacher and the students' intermediate layers. The cosine distance between two vectors is given as :

$$cos(y, \hat{y}) = \frac{y \cdot \hat{y}}{|y||\hat{y}|} \tag{4.3}$$

The Cosine Embedding Loss is given as:

$$\text{loss}(y, \hat{y}, l) = \begin{cases} 1 - \cos(y, \hat{y}), & \text{if } l = 1 \\ \max(0, \cos(y, \hat{y}) - \text{margin}), & \text{if } l = -1 \end{cases} \tag{4.4}$$

The cos embedding loss tells how close the embeddings should be when the label l is 1 and how far the embeddings should be when the label l is -1. Hence cosine similarity works on embeddings that are nonlinear and embeddings of semi-supervised models.

For our student teacher training, we consider the y as the teachers' intermediate representation and $\hat{y}$ as the students' intermediate representation, and the margin helps in penalizing the similarity between the similar samples and dissimilar samples representation based on the label l.

4.4.4 Knowledge Distillation Results

MSE for Distillation: The training dataset used for the experiment librispeech. The Teacher model used for the experiment is DeepSpeech2 architecture with 56 million parameters, and the student model size is 1.3 million parameters. The student model has one CNN and GRU layer then followed by a fully connected layer. We conducted experiments with the combinations of MSE loss and CTC loss, with distillation applied to more than one layer in the network. The CTC loss is applied to the experiments where it is listed in the loss function column in the table 4.4. Only the parameters of the student network are updated during training.

Results Discussion: It can be observed from the table 4.4 that MSE loss on its own has failed to contribute to the distillation of knowledge, while the distilled model performs well when trained with the combination of CTC loss. At the same time, we can see a slight improvement in distillation applied

Table 4.4: Effect of MSE loss function on Distillation for small footprint KWS on both TIMIT and LRW dataset

S.No.	Loss Function	Distillation Layers	TIMIT	LRW
1	CTC + MSE	RNN	0.569	0.355
2	Only MSE	RNN	0.088	0.018
3	Only MSE	RNN and CNN	0.043	0.018
4	CTC + MSE	RNN and CNN	0.687	0.549

to more than one location. Distillation applied on both RNN and CNN layers performs slightly better than distillation applied only on the RNN layer.

Comparison of MSE, KL Div and CosEmbedding loss for distillation: We conduct experiments using all the three loss functions for reducing the size of the keyword spotting model to do small footprint keyword spotting. We measure the performance of the student model with reduced size for the open vocabulary keyword spotting task after training with MSE, KL Div, and CosEmbedding loss functions individually and report the results in the table 4.5. The student network for these experiments is constant and has one CNN layer and one RNN layer totaling 2.5 million parameters.

Table 4.5: Performance of MSE, CosEmbedding and KL Divergence loss functions on distillation for small footprint KWS on TIMIT dataset

S.No.	Loss Function	TIMIT
1	MSE Loss	0.538
2	Cosine Embeddings Loss	0.569
3	KL Divergence Loss	0.534

Results Discussion: It can be observed that CosEmbedding loss function helps in improving the performance of the student model in distillation from a Deepspeech2 teacher model. However, the model performance is insufficient to apply to real-world small footprint keyword spotting scenarios.

4.5 Distillation vs Small Footprint ASR

We sensed that the distillation loss functions could hinder the CTC loss function's contribution to learning representation for keyword spotting. We focused on building a small footprint ASR and extracting the small footprint KWS model from a small footprint ASR without any distillation involved. Hence we have to train an ASR with varying parameters to analyze the keyword spotting results.

4.5.1 Small Footprint ASR Training

The ASR training is similar to the one described in the previous chapter with the CTC loss function. Modern ASRs take the speech processed in the form of either spectrogram, MFCC, fbank, or some

other features as input rather than raw audio. The raw audio has many data points for a single second of audio and cannot be input to the ASR. These are a sequence of vectors with a dimension D representing a fraction of the input audio. The ASR outputs the probability scores for each possible character for each input frame. The CTC loss converts this output by replacing repeating characters not separated by blank characters into a single character. In this manner, ASR handles the task of transforming variable length inputs into variable length outputs. The number of parameters used for such training is always huge as the input space is very high dimensional, and we want to capture the spoken words from it. However, our task is keyword spotting which doesn't require the audio to be transcribed but brings the representations of similar audio keywords closer. Hence we try to build a small footprint ASR whose focus will be on building an efficient representation of keyword audio rather than an efficient ASR.

However, the number of parameters used in this ASR is much smaller than the one used in the previous models. The results of KWS Model evaluated on small footprint ASR is shown in table 4.6. The training data used for this experiment is Librispeech and is evaluated in TIMIT and LRW datasets. The model parameters is varied by varying the number of RNN layers and the hidden dimension of the RNN layers.

Table 4.6: Effect of Model Parameters on small footprint keyword spotting performance.

Model Architecture	Embedding Dimension	Model Parameters	Input Features	TIMIT	LRW
2CNN 3BiLSTM	1024	18.1M	logfbank	0.916	0.819
2CNN 5BiLSTM	96	1.8M	spectrogram	0.752	0.607
2CNN 3LSTM	64	316K	MFCC	0.473	0.197
2CNN 7BiLSTM	1024	56M[Pretrained]	spectrogram	0.857	0.802

Results Discussion: From table 4.6, it can be observed that reducing the size of the model from 56M to 18.1 has helped in improving the KWS results. But while reducing the model further, the model's performance has reduced. It can be observed that training a small footprint ASR has better performance in keyword spotting 4.6 than the distillation techniques observed in Table 4.5. This is also due to the fact that the ASR training has been periodically checked for keyword spotting and the training is stopped irrespective of the Word Error Rate of the ASR model. With further training, the ASR model WER decreases but the keyword spotting performance decreases. This is the reason for the ASR model to perform better when trained from scratch and involves early stopping while focusing on keyword spotting performance. It can be observed that the small footprint ASRs are quite competitive with distillation techniques for small footprint Keyword Spotting. This also shows that Knowledge transfer through distillation in Speech Processing tasks is quite challenging when compared to building the solution from scratch.

Chapter 5

Conclusion and Future Directions

In this thesis, we propose a novel method of learning acoustic word embeddings by transferring pretrained ASR architecture for KWS with triplet network and CTC. We show that this method efficiently detects keywords the model has not seen during training. By comparing it with other benchmarks, we show that this method is competitive on open-vocabulary audio keyword spotting tasks. Hence the system can be extended to detect any audio keywords while having the opportunity to fine-tune the keywords to improve performance. This method utilizes the knowledge learned during ASR training for the speech recognition task.

We show that the transliteration can be used to utilize the system trained on the English language to be directly used to low resource language by transliterating the low resource language to English character. We show initial zero-shot generalization results for low-resource languages and share the text transliteration for Tamil language keywords in the MSWC dataset. It is also observed that pretraining in one language helps in KWS on a different low-resource target language. We also confirm the findings on other low-resource languages Vallader and Hausa.

Finally, we focus on reducing the model parameters by Knowledge Distillation techniques to make the KWS model work in a small footprint scenario. We observe that the CTC loss function plays a significant role in training a small footprint KWS model compared to the distillation loss functions. Hence we understand that representation learned from a small footprint ASR can produce a better KWS model even when the Word Error Rates of such small footprint ASR models are high.

5.1 Future Directions

The transliteration helps adapt a pretrained ASR for a low-resource language. It would be interesting to see how a pretrained English ASR recognizes a low-resource language when fine-tuned with limited training data. The predicted ASR's English alphabets can be converted to the low-resource language script using transliteration. There are not many efficient transliteration packages available for many low-resource languages. Using an ASR to perform transliteration would be an interesting area to explore.

The keyword spotting system can be used to detect the intention of the audio when specific keywords are matched. This helps in building an alert system based on the contents of the audio. Such systems are possible with keyword spotting systems as the entire audio is not transcribed but by looking for specific keywords.

Related Publications

Conference:

1. **Kirandevraj R**, Vinod K Kurmi, Vinay P Namboodiri, CV Jawahar. **Generalized KWS using ASR embeddings.** *INTERSPEECH , 2022*

Bibliography

[1] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury. End-to-end asr-free keyword search from speech. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1351–1359, 2017.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[3] H. Chen, C. C. Leung, L. Xie, B. Ma, and H. Li. Multitask feature learning for low-resource query-by-example spoken term detection. *IEEE Journal of Selected Topics in Signal Processing*, 11:1329–1339, 2017.

[4] T. K. Chia, K. C. Sim, H. Li, and H. T. Ng. A lattice-based approach to query-by-example spoken document retrieval. In *SIGIR '08*, 2008.

[5] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken se. 1980.

[6] D. C. de Andrade, S. Leo, M. Viana, and C. Bernkopf. A neural attention model for speech command recognition. *ArXiv*, abs/1808.08929, 2018.

[7] L. Deng. Switching dynamic system models for speech articulation and acoustics. In *Mathematical Foundations of Speech and Language Processing*, pages 115–133. Springer, 2004.

[8] T. Fuchs and J. Keshet. Spoken term detection automatically adjusted for a given threshold. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1310–1317, 2017.

[9] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

[10] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen. Exemplar-based sparse representations for noise robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19:2067–2080, 2011.

[11] Y. Gong, Y.-A. Chung, and J. R. Glass. Ast: Audio spectrogram transformer. *ArXiv*, abs/2104.01778, 2021.

[12] A. Graves, A. rahman Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[13] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. F. Diamos, E. Elsen, R. J. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng. Deep speech: Scaling up end-to-end speech recognition. *ArXiv*, abs/1412.5567, 2014.

[14] T. J. Hazen, W. Shen, and C. M. White. Query-by-example spoken term detection using phonetic posteriorgram templates. *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 421–426, 2009.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] W. He, W. Wang, and K. Livescu. Multi-view recurrent neural acoustic word embeddings. *arXiv preprint arXiv:1611.04496*, 2016.

[17] W. He, W. Wang, and K. Livescu. Multi-view recurrent neural acoustic word embeddings. In *Proc. ICLR*, 2017.

[18] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. rahman Mohamed, N. Jaitly, A. W. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29:82–97, 2012.

[19] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504 – 507, 2006.

[20] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[21] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015.

[22] H. Kamper, W. Wang, and K. Livescu. Deep convolutional acoustic word embeddings using word-pair side information. In *2016 IEEE ICASSP*, pages 4950–4954. IEEE, 2016.

[23] B. Kim, M. Lee, J. Lee, Y. Kim, and K. Hwang. Query-by-example on-device keyword spotting. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 532–538, 2019.

[24] T. Kim and J. Nam. Temporal feedback convolutional recurrent neural networks for keyword spotting. *ArXiv*, abs/1911.01803, 2019.

[25] S. King, J. Frankel, K. Livescu, E. McDermott, K. Richmond, and M. Wester. Speech production knowledge in automatic speech recognition. *The Journal of the Acoustical Society of America*, 121 2:723–42, 2007.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] H. Kucera, W. N. Francis, W. F. Twaddell, M. L. Marckworth, L. M. Bell, and J. B. Carroll. Computational analysis of present-day american english. 1967.

[28] O. Kuchaiev, B. Ginsburg, I. Gitman, V. Lavrukhin, J. Li, H. Nguyen, C. Case, and P. Micikevicius. Mixed-precision training for nlp and speech recognition with openseq2seq. *arXiv preprint arXiv:1805.10387*, 2018.

[29] L.-S. Lee and Y.-C. Pan. Voice-based information retrieval — how far are we from the text-based information retrieval ? *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 26–43, 2009.

[30] C. T. Lengerich and A. Y. Hannun. An end-to-end architecture for keyword spotting and voice activity detection. *ArXiv*, abs/1611.09405, 2016.

[31] M. Mazumder, C. R. Banbury, J. Meyer, P. Warden, and V. J. Reddi. Few-shot keyword spotting in any language. In *Interspeech*, 2021.

[32] M. Mazumder, S. Chitlangia, C. Banbury, Y. Kang, J. M. Ciro, K. Achorn, D. Galvez, M. Sabini, P. Mattson, D. Kanter, et al. Multilingual spoken words corpus. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[33] M. Mazumder, S. Chitlangia, C. R. Banbury, Y. Kang, J. Ciro, K. Achorn, D. Galvez, M. Sabini, P. Mattson, D. Kanter, G. F. Diamos, P. Warden, J. Meyer, and V. J. Reddi. Multilingual spoken words corpus. In *NeurIPS Datasets and Benchmarks*, 2021.

[34] L. Miculicich, D. Ram, N. Pappas, and J. Henderson. Document-level neural machine translation with hierarchical attention networks. *ArXiv*, abs/1809.01576, 2018.

[35] X. A. Miró, L. J. Rodriguez-Fuentes, I. Szöke, A. Buzo, F. Metze, and M. Peñagarikano. Query-by-example spoken term detection evaluation on low-resource languages. In *SLTU*, 2014.

[36] M. Müller. Information retrieval for music and motion. 2007.

[37] A. Park and J. R. Glass. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 16:186–197, 2008.

[38] L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *Journal of the Acoustical Society of America*, 63, 1978.

[39] D. Ram, A. Asaei, and H. Bourlard. Subspace detection of dnn posterior probabilities via sparse representation for query by example spoken term detection. In *INTERSPEECH*, 2016.

[40] D. Ram, L. Miculicich, and H. Bourlard. Multilingual bottleneck features for query by example spoken term detection. In *2019 IEEE ASRU*, pages 621–628, 2019.

[41] D. Ram, L. M. Werlen, and H. Bourlard. Cnn based query by example spoken term detection. In *Interspeech*, pages 92–96, 2018.

[42] L. J. Rodriguez-Fuentes, A. Varona, M. Peñagarikano, G. Bordel, and M. Díez. High-performance query-by-example spoken term detection on the sws 2013 evaluation. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7819–7823, 2014.

[43] A. Rosenberg, K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny. End-to-end speech recognition and keyword search on low-resource languages. In *2017 ICASSP*, pages 5280–5284. IEEE, 2017.

[44] T. N. Sainath, B. Ramabhadran, M. Picheny, D. Nahamoo, and D. Kanevsky. Exemplar-based sparse representation features: From timit to lvcsr. *IEEE Transactions on Audio, Speech, and Language Processing*, 19:2598–2613, 2011.

[45] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[46] D. Shitov, E. Pirogova, T. A. Wysocki, and M. Lech. Learning acoustic word embeddings with dynamic time warping triplet networks. *IEEE Access*, 8:103327–103338, 2020.

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[48] K. Veselý, M. Karafiát, F. Grézl, M. Janda, and E. Egorova. The language-independent bottleneck features. *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 336–341, 2012.

[49] R. Vygon and N. Mikhaylovskiy. Learning efficient representations for keyword spotting with triplet loss. In *SPECOM*, 2021.

[50] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[51] J. Yoon, H. Lee, H. Y. Kim, W. I. Cho, and N. S. Kim. Tutornet: Towards flexible knowledge distillation for end-to-end speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1626–1638, 2021.

[52] D. Yu and M. L. Seltzer. Improved bottleneck features using pretrained deep neural networks. In *INTERSPEECH*, 2011.

[53] B. Yusuf and M. Saraclar. An empirical evaluation of dtw subsampling methods for keyword search. In *INTERSPEECH*, 2019.

[54] H. Zhang, J. Zhang, and Y. Wang. Sequence-to-sequence models for small-footprint keyword spotting. *arXiv preprint arXiv:1811.00348*, 2018.

[55] Y. Zhang and J. R. Glass. Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams. *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 398–403, 2009.

[56] Y. Zhang, R. Salakhutdinov, H.-A. Chang, and J. R. Glass. Resource configurable spoken query detection using deep boltzmann machines. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5161–5164, 2012.

[57] Z. Zhu, Z. Wu, R. Li, H. M. Meng, and L. Cai. Siamese recurrent auto-encoder representation for query-by-example spoken term detection. In *INTERSPEECH*, 2018.

[58] Y. Zhuang, X. Chang, Y. Qian, and K. Yu. Unrestricted vocabulary keyword spotting using lstm-ctc. In *Interspeech*, pages 938–942, 2016.