

End to End ASR Free Keyword Spotting with Transfer Learning from Speech Synthesis

Koilakuntla Bramhendra

A Thesis Submitted to
Indian Institute of Technology Hyderabad
In Partial Fulfillment of the Requirements for
The Degree of Master of Technology

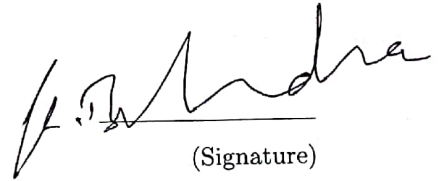


Department of Electrical Engineering

Dec 2019

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

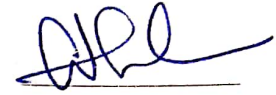

(Signature)

(Koilakuntla Bramhendra)

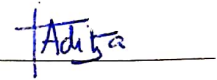
ee17mtech01002
(Roll No.)

Approval Sheet

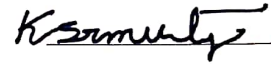
This Thesis entitled End to End ASR Free Keyword Spotting with Transfer Learning from Speech Synthesis by Koilakuntla Bramhendra is approved for the degree of Master of Technology from IIT Hyderabad



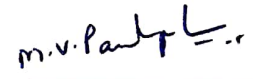
(Dr Sumohana S. Channappayya) Examiner
Dept. Electrical Eng
IITH



(Dr. Aditya Siripuram) Examiner
Dept. Electrical Eng
IITH



(Dr. K Sri Rama Murty) Adviser
Dept. of Electrical Eng
IITH



(Dr M V Pandu Ranga Rao) Chairman
Dept Computer Sci
IITH

Acknowledgements

I would like to thank my prof Dr. K Sri Ram Murty with out his support this wouldn't have been possible. I am grateful to DVLN Dheeraj V Venkatesh and Dr Shekhar , Siva Ganesh , Md Rafi , C Shiva Kumar , Sreekanth, Ramesh, Swetha , Kishor and all my lab mates for their support

Dedication

To my grand parents (K Pichamma and K Rangayya) and to my family

Abstract

Keyword Spotting is an important application in speech. But it requires as much as data of an Automatic Speech Recognition(ASR).But the problem is much specific compare to that of an ASR. Here the work made an effort to reduce the transcribed data dependency while building the ASR. Traditional keyword spotting(KWS) architectures built on top of ASR. Such as Lattice indexing and Keyword filler models are very popular in this approach. Though they give very good accuracy the former suffers being a offline system, and the latter suffer from less accuracy Here we proposed an improvement to an approach called End-to-End ASR free Keyword Spotting. This system has been inspired from traditional keyword spotting architectures consist of three modules namely acoustic encoder and phonetic encoder and keyword neural network. The acoustic encoder process the speech features and gets a fixed length representation, same as phonetic encoder gets fixed length representation both concatenated to form input for keyword neural network. The keyword network predicts whether the keyword exist or not. Here we proposed to retain all the hidden representation to have temporal resolution to identify the location of the query. And also we propose to pretrain the phonetic encoder to make aware of acoustic projection. By doing these changes the performance is improved by 7.1% absolutely. And in addition to that system being end to end gives an advantage of making it easily deploy able.

Contents

Declaration	ii
Approval Sheet	iii
Acknowledgements	iv
Abstract	vi
Nomenclature	viii
1 Introduction	1
1.1 Keyword Spotting	1
1.2 Motivation	2
2 Feature Extraction	4
2.1 Mel Frequency Cepstral Coefficients	4
2.1.1 Pre-Emphasis	4
2.1.2 Windowing	5
2.1.3 Discrete Fourier Transform	5
2.1.4 Mel filter bank and log	5
2.1.5 The Cepstrum	6
2.1.6 Deltas and Energy	6
3 Acoustic Modeling	7
3.1 Forced Alignment	7
3.1.1 Monophone Alignment	9
3.1.2 Tri-phone Alignment	9
3.2 Deep Neural Network based Acoustic modeling	10
3.2.1 Time-Delay Neural Networks	10
3.3 Language Modeling	11
3.4 Decoding Graph Construction	11
4 Keyword Spotting Methods	13
4.1 LVCSR based Keyword Spotting	13
4.2 Discriminative Keyword Spotting	13
4.2.1 Small Foot-Print Keyword Spotting	13
4.3 Lattice indexing	14
4.3.1 Lattice generation	14

4.3.2	Keyword Search	15
4.4	Keyword-Filler based Methods	15
5	Sequential Architectures	16
5.1	Recurrent Neural Networks	16
5.2	Long Short Term Memory	17
5.3	Encoder-Decoder Architecture	18
5.4	Attention Mechanism	19
6	Proposed Approach	20
6.1	End-to-End ASR free Keyword spotting	20
6.1.1	Baseline system	20
6.2	Proposed Architecture	21
6.2.1	Acoustic Encoder	21
6.2.2	Phonetic Encoder	21
6.2.3	Attention	21
6.2.4	KWS Neural Network and Cost Function	21
6.3	Transfer Learning from Speech Synthesizer	22
6.3.1	Duration Modeling	22
6.3.2	Acoustic Modeling	23
7	Experimental Setup	24
7.1	Experimental Setup	24
7.1.1	Database Description	24
7.1.2	End-to-End ASR free Keyword Spotting	24
7.1.3	Results and Discussion	26
7.2	Conclusion	26
	References	26

Chapter 1

Introduction

1.1 Keyword Spotting

Keyword Spotting(KWS) is the task of identifying putative hits of text query in a reference audio file. As we can see unlike traditional text search or audio to audio search here is a domain mismatch between reference and query. Most common way to address this problem is to convert reference audio to text domain. But transcribing entire audio text will require lot of resources both in terms of human effort as well as technical resources. So we general compromise of the text abstraction that we choose to transcribe or the quality of the text that we get. In the first case we chose identify the phonemes in the reference audio content and the query also converted to phoneme level and then perform the search using various search methods. In latter case these keyword spotting systems operate on low quality Automatic Speech Recognition(ASR)[1] hypothesis. As these are of low quality in nature, there must effective search methods to improve the performance. The common practice is do lattice search. Notable works has been in carried out in[2][3]. But operating at word level keyword spotting, the main drawback would be not able to handle out of vocabulary(OOV) words. This problem can be solved up to some extent using phonetic level keyword spotting. Here he search methods can be of lattice indexing or filler keyword models. As we can observe lattice based search methods gives high accuracy but these will be of offline in nature. Because it is important in some applications to detect keywords on the fly for example forensic or call tapping. For this we use keyword-filler based methods. These methods will be fast and have low memory foot print at the cost of accuracy. There are various versions of these keyword-filler models have been proposed[4]. A detailed study of various keyword spotting methods can be found in [5]. The block diagram of generic keyword spotting can be found in 1.1.

With the advent of End-to-End systems every system finding corresponding version of traditional architectures, especially in speech recognition and keyword spotting systems. Though these system performance still need to improve to match that of traditional architectures, they provide unified frame work and easy to train. In the same line of End-to-End keyword spotting systems are proposed. Especially for streaming services [6][7] such as voice assistants End-to-End systems provide low latency. Not just in time as well as in terms memory, the authors in [8] have proposed to use singular value decomposition on time delay neura networks to improve the performance. End-to-End systems performance is significantly low when they operate in low resource scenarios [9]. In

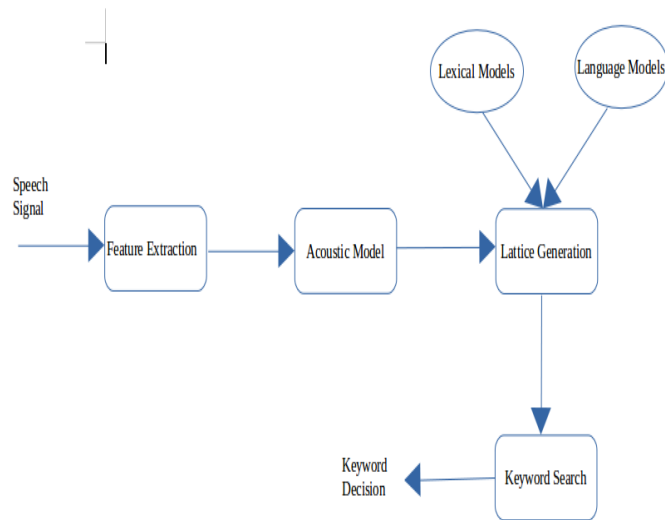


Figure 1.1: Block diagram generic Keyword-Spotting system

[10] the authors show that this due to their peaky nature of posteriors.

1.2 Motivation

Though all these systems give very good performance these systems , require sufficient amount of data to build the keyword spotting system. The sufficiency could be varying from tens of hours data for traditional architectures and it will be 100s of hours data for End-to-End to architectures. In addition the problem of domain mismatch is always addressed by projecting reference audio into text domain but not the other way around. If we project text query in to audio domain, by the nature speech synthesis systems it wont be having speaker variability and still we need to be measuring the similarity of text content. But one advantage we get by projecting query text in to audio domain is that it wont require as much as data of speech recognition. The main challenge will be to obtain speaker variability and get a feature representation that is rich in text contents to compare the similarity. In this line of thought authors in [11] have proposed End-to-End asr free keyword spotting. This method does not require any transcribed data expect the information that whether the keyword is exist or not. With the minimal supervision it gives quite encouraging result to pursue research in that direction. Our approach similar to this approach in fact its a modified version of it. In this asr-free keyword spotting there are three components, Acoustic Encoder, Phonetic Encoder and Keyword Network. Acoustic encoder and phonetic encoder initially trained as auto encoders with the corresponding input and then the encoder parts of both encoders will be taken apart and joined with keyword encoder to form one single architecture. The acoustic encoder will be fed speech input to get a fixed length representation and the phonetic encoder will fed a keyword character as input to get another fixed length representation both will be concatenated to form one single input. These input will be forwarded to keyword network to classify whether the keyword exist or not.

The main draw of backs this system is that if get a fixed length representation for acoustic

features the timing information will be lost. In keyword spotting it is equally important to locate the query as well. And another draw back is that we try to project acoustic features and phonetic features in to same shared space with just by classifying whether the keyword exist or not i.e the cross entropy loss. This one arduous task to achieve without any further supervision. To overcome these draw backs we propose to two main changes. One is that instead training for both encoder to be auto encoder and getting a fixed length representation we keep all the hidden states in acoustic encoder and take the last state of phonetic encoder refereed as keyword embedding. From these we compute attention between acoustic hidden representation and keyword embedding. This gives the idea of where the query is located. And another change is we propose to pre-train the phonetic encoder with speech synthesis. This is to simplify the task of projecting acoustic and phonetic encoder in to same shared space. Given these two changes the system will perform better and gives significant improvement over the existing system. The further details of the system has been enclosed in remaining chapters

Chapter 2

Feature Extraction

In general speech signal contains lot of information. While a person is speaking it records speech contents, speaker information, gender information, emotion information, etc...irrespective of all these we should be able to perform the keyword search. Operating at raw samples will be very difficult. As our problem is text content dependent the most suitable will be Mel Frequency Ceptral Coefficients.

2.1 Mel Frequency Ceptral Coefficients

The general block diagram of MFCC feture extraction has been represented in 2.2. The content has been inspired from[12]

2.1.1 Pre-Emphasis

MFCC feature extraction will start with the step of pre-emphasis. This is to enhance the contents of higher frequencies. The spectrum of voiced segments like vowels more focused on lower frequencies. It makes higher frequencies irrelavant. To access higher frequencies information we will do pre-emphahsis as a first step. The pre-emphasis enhanced frequencies has depicted in 2.1

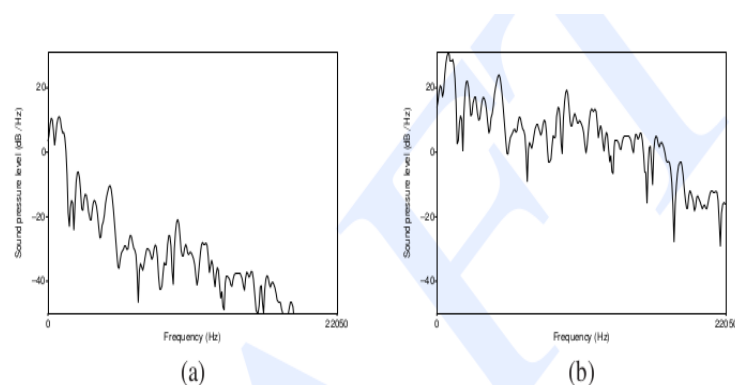


Figure 2.1: Pre-emphasis of Vowel [aa][12]

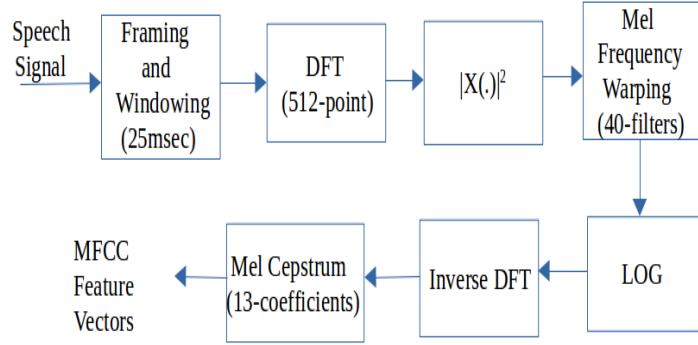


Figure 2.2: MFCC Feature Exatraction Steps

2.1.2 Windowing

As we initially said we want to extract features for phonetic contents of speech signal and thats why we wont chose to extract feature from utterance level. In addition to that speech is highly non stationary signal. The spectral characteristics and distributional changes will happen across time. For this we break the speech signal into chunks where we can assume stationarity. In general this duration in speech is 25 milliseconds. We apply a window either rectangular or hamming window with a shift of 10 milliseconds to avoid the windowing effects.

$$\begin{aligned}
 \text{rectangular}w[n] &= \begin{cases} 1 & 0 \leq n \leq L - 1 \\ 0 & \text{otherwise} \end{cases} \\
 \text{hamming}w[n] &= \begin{cases} 0.54 - 0.46\cos(\frac{2}{L}n) & 0 \leq n \leq L - 1 \\ 0 & \text{otherwise} \end{cases}
 \end{aligned} \tag{2.1}$$

2.1.3 Discrete Fourier Transform

The next step in MFCC feature extraction is to analyse the spectral contents in speech signal. This done by using Fourier transform. Fourier transform gives energy in each frequency. Because each phonetic content has characterized by different spectral contents. lets say $x[1], x[2]..x[m]$ as the input signal..

$$X[k] = \sum_0^N -1x[n] \exp \frac{-j2\pi kn}{N} \tag{2.2}$$

2.1.4 Mel filter bank and log

DFT will be energy distribution in frequency domain. But human ear is not equally sensitive all frequencies. Human ear is more sensitive to frequencies lower than 1000Hz and lesser sensitive to higher frequencies. Its would improve speech recognition performance if we wrap the frequencies. Such wrapping is provided mel scale. mel is a unit of pitch.

$$\text{mel}(f) = 1127 \ln(1 + \frac{f}{700}) \tag{2.3}$$

This wrapping is created at the time of filter bank computation which collects energy from each frequency band. And then we apply logarithmic for frequency values.

2.1.5 The Cepstrum

Though it would be possible to use filter bank as feature representation but spectrum has some problem to use as it is for phone recognition. As we pointed out earlier speech production is modeled using source filter model. Where the source is corresponding to speaker characteristics and the filter is responsible for phones. For this we would like to extract the filter characteristics to make it more speaker agnostic. As we applied log on frequency domain the source and the filter will get separated in addition. From here we take IDFT(Inverse Discrete Fourier Transform). So that would be not exactly time domain so we call it Cepstrum.

2.1.6 Deltas and Energy

In general we take first 13 coefficients to represent the speech frame. The first coefficient gives energy of the samples present in the frame. Another important thing is that while calculating feature representation it is important to characterize speaking variations in speech signal. This is done by using velocity and acceleration features. The first derivative corresponds to velocity(delta) and the second derivative corresponds to acceleration(double delta) features

$$d(t) = \frac{c(t+1) - c(t-1)}{2} \quad (2.4)$$

The summary of the mfcc feature extraction can be found in block diagram depicted in figure2.

Chapter 3

Acoustic Modeling

Acoustic modeling is to model the distribution of phone sounds. The production of phones is a random process, in order to model the phones we first know where each phone starts and it ends. But marking those boundaries will be very tough, where are marking at word level and sentence level information will be relatively simple. Being said that its very difficult to model the words distribution as the number of words in any language runs into millions. So phones are the universal sounds that can generalise the entire language and will be in limited number. We will be knowing the how each word is pronounced, given this information and having labels marked at word or sentence level we can still model the phonetic distribution. For that we use forced alignment algorithm.

3.1 Forced Alignment

The time varying distribution of phone sounds will be characterized by Hidden Markov Models(HMMs). The observation probabilities are modeled by Guassian Mixture Models(GMMs)But as we mentioned earlier we dont have the boundaries for each phone. So we simultaneously models the phones to obtain the boundaries. Modeling HMMs implies to find the $[A,B,\pi]$ parameters and observation probabilities. Usally phone production mechanism will have three stages, transition burst and closer regions. Inspired by this we use left right HMMs with 3 to 5 states to model the phones. This has been depicted in figure 1. While modeling the HMMs we initially segment the audio file into equal segments as many as the number of phones in the audio file. This will be done across the training data. The examples for each phone will collected and trains a HMMs for it. With this HMM the boundaries will be estimated again. With the refined boundaries the the HMMs will trained again. This process will repeat until there no longer change in the boundary. This gives the boundaries of each phone and the a model that characterize its distribution. Gaussian Mixture Models

$$p(x_n) = \sum_{k=1}^K \pi_k N(x_n | \mu_k, \Sigma_k) \quad (3.1)$$

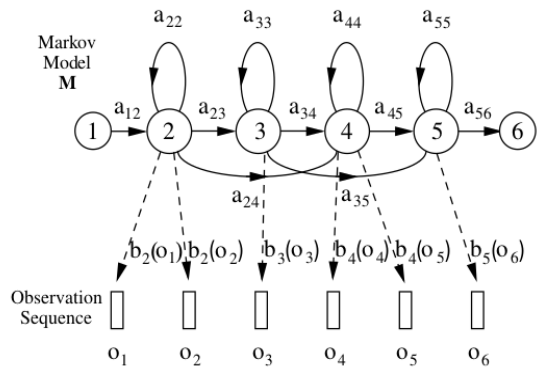


Figure 3.1: Hidden Markov Models[12]

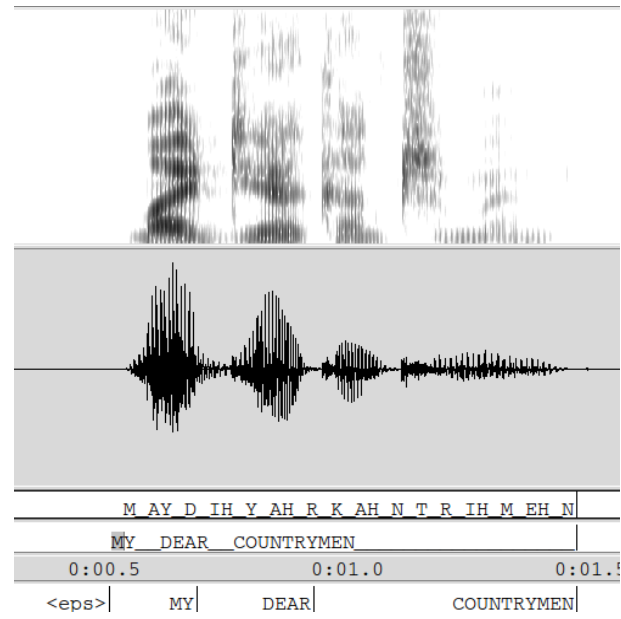


Figure 3.2: Forced alignment example

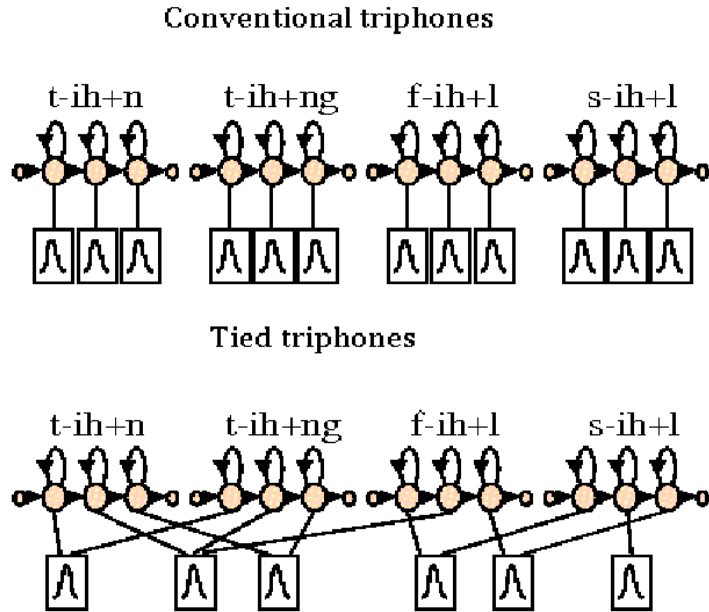


Figure 3.3: State tying of tri-phones

3.1.1 Monophone Alignment

Monophone alignment is to alignment with out considering any contextual information. Each phone is treated independent of other. The phone are modeled at state level. That means each phone will be modeled with a HMM, so each state will have different distribution. The word level modeled are just concatenation of phone level models. This gives good performance but there is some more knowledge that we can use to improve the alignments,i.e contextual information

3.1.2 Tri-phone Alignment

As there are co-articulation variations involved when we pronounce words its better to incorporate it. But it would be difficult to consider longer contexts. For example if consider three phones together and there are 40 language, we will end up in estimating 1600000 parameters. This is impractical and each one of them will not be appearing in the language. Studies show that only 10% are frequently occurring out of all the possible phone. To choose this we form a decision tree, at each end will be asking a question to decide which triphone to keep and which or not. Now we use this as fundamental units like we did with mono phones. Even with 10% of triphones some times it is difficult to find enough data to estimate the parameters for. For this we tie some of the state of each triphone HMM. For example p-a-t and b-a-t have similar type of ending, so the corresponding to the HMMs will be added to form one single state. While getting back to the phone we keep enough information so that the inversion is unique. This has been depicted in 6.1

Architecture of a DNN-HMM hybrid system

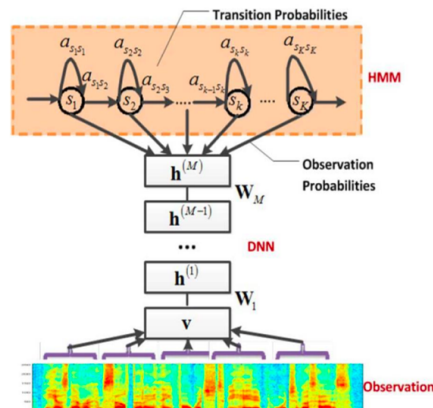


Figure 3.4: Deep Neural Network Based Acoustic Modeling [1]

3.2 Deep Neural Network based Acoustic modeling

Even though HMM-GMM frame work itself can be seen as model for speech recognition. But it is generative frame work, and a discriminative model will most suits for the given problem. So we chose to model Deep Neural Networks(DNNs) for acoustic modeling. After getting the boundaries from forced alignment these boundaries are used as labels to train the DNNs. The input features are MFCC. Instead of feeding one frame at time, given that speech signal has time dependency we concatenate previous and and future frames to current fram to from one context vector. In general these frame context can be vary from 9 to 3 frames, depending one the problem. Here we predict the triphone HMMs state observation probabilities instead of directly classifying the phone. Then these HMMs states are mapped to phones and then to words. The DNNs acoustic modeling diagram has been depicted in figure3. DNNs are memory less systems, Ideally we will be requiring longer memory to model speech. So there are better deep architectures that can model acoustic model better. For example TDNNs and RNNs are widely used in the context of acoustic modeling.

3.2.1 Time-Delay Neural Networks

The draw of DNNs is that they wont be having any memory, but speech signal have dependency on previous frames. To incorporate longer temporal contexts time-delay-neural networks(TDNNs)[13] have been proposed. TDNNs can process longer contexts which can improve the accuracy of acoustic modeling. The fundamental idea is that instead of processing entire input at once it process along the network. Initially smaller receptive field is fed across the network as the depth increases this receptive will have more temporal context. By doing this we can process more temporal context. The lower layers learn transforms insensitive feature in variance. In addition to this authors in[] have also included sub-sampling which improved the performance of acoustic modeling.

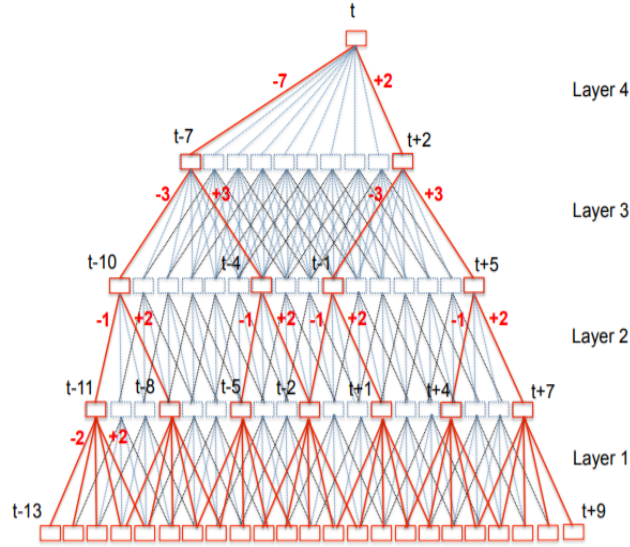


Figure 3.5: TDNN block diagram with sub sampling[13]

3.3 Language Modeling

The temporal context for speech is longer. Even though we have context maximum up to phones. But each language will context in terms of words. It is not that every word sequence is possible. To incorporate language structure we use language model. Its statistical model to estimate the probability of the sequence of words. The longer the word sequence the better the model is. As we never able to match the original distribution because it is not possible to have every word sequence. But the curse of dimensional prevents us from doing this. Lets say N is the word is number of word sequence we will be estimating, then we have to estimate N^V parameters, where V is the vocabulary size. As the N increases this number becomes beyond our reach-ability. Usually in practice this number is in between three to five. The best numbers are obtained with three in literature. The parameters are empirically estimated from the text we used for training. The equations for N -gram are as follows

From chain rule

$$P(W_1, W_2 \dots W_N) = P(W_1)P(W_2|W_1)P(W_3|W_1, W_2) \dots P(W_n|W_1, W_{n1})$$

$$P(W_i|W_{i1}) = \frac{C(W_{i1}W_i)}{C(W_{(i-1)})}$$

We apply N -gram markov approximation to this

$$P(W) = \pi_{i1}^n P(W_i|W_{in+1}) \tag{3.2}$$

3.4 Decoding Graph Construction

After building an acoustic model, the out puts are not human readable because they are triphones. We need to map them words to make understandable. For this we use a generic and unified frame

Chapter 4

Keyword Spotting Methods

Once we transcribe the audio into phoneme or word level then we perform the keyword search. There are different ways of doing it.

4.1 LVCSR based Keyword Spotting

On the face it one can intuitively think of performing keyword spotting by transcribing audio in to text. We build lattice of word sequence and then perform the keyword search. When we transcribe at word level the immediate problem will be out of vocabulary(oov). It is very difficult to recognise oov words and especially proper names such company names and unique people names. The other problem is that choosing the right language model. Because it is very important to have good language model other wise the performance will be poor.

4.2 Discriminative Keyword Spotting

Any keyword or word can be represented in terms of phonetic sequence[14]. The goal is to identify whether the given phone sequence is present in the speech signal or not. lets say the speech signal is represented as $x_1, x_2 \dots x_T$ for $t \in T$ in R^d space. The phonetic sequence is represented as $p_1, p_2 \dots p_L$ where p is phoneme domain set. our goal is \mathbf{x}, \mathbf{p} are matches are not. In addition to that we need find the alignment as well. The goal is optimize

$$f(\mathbf{x}, \mathbf{p}) = \mathbf{w} \cdot \max_{s, w} \phi(\mathbf{x}, \mathbf{p}, \mathbf{s}) \quad (4.1)$$

Where \mathbf{s} is the alignment. This is learned over an iterative algorithm.

4.2.1 Small Foot-Print Keyword Spotting

This is to detect whether a keyword is exist in a audio or not[15]. The keyword here known as wake up words. These kind of algorithms are used in voice assistants. The given speech signal is represented in terms of MFCC features and the label for each speech signal is given either one or zero based on the keyword existence. The network is trained with cross entropy loss. AS it is predicted for each frame at the end posterior smoothing will be done to make the decision.

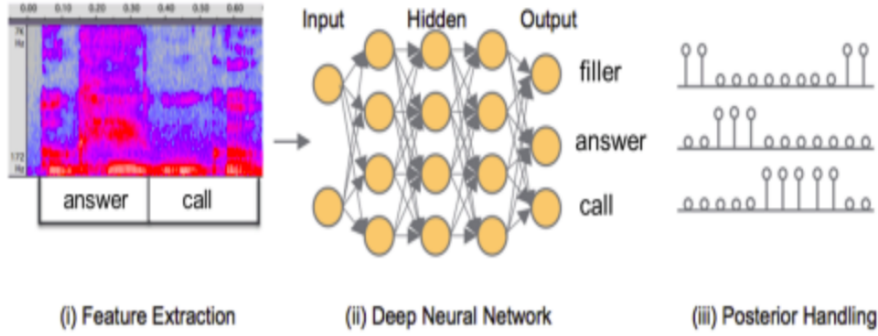


Figure 4.1: DNN based wake word detection

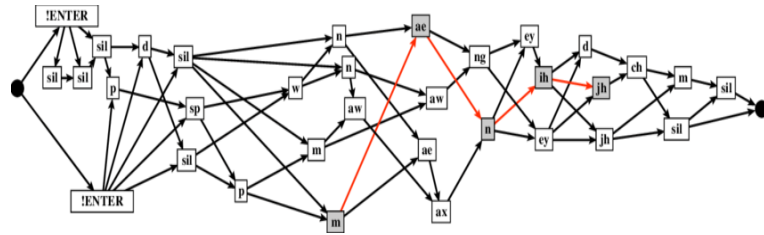


Figure 4.2: Phone lattice for lattice indexing[15]

4.3 Lattice indexing

4.3.1 Lattice generation

Usually the definition of the lattice is not exactly defined[16]. There are several variants of definitions that have been followed. The decoding graph HCLG.fst will be combined with the acoustic model to form the search graph. Let's say our acoustic model is represented as U and the decoding graph as HCLG. We form the search graph by taking the composition of both graphs

$$S = U \circ HCLG \quad (4.2)$$

From this given T frame duration acoustic signal, we encode the N best possible path information in this graph. When it comes to the definition of the lattice, it is loosely defined as a weighted directed acyclic graph. In some of the definitions, it includes timing information and phone-level alignments are also included. It is very difficult to traverse all the paths in the lattice, so we do α level pruning such that the following properties are preserved:

- The lattice will contain every path below α threshold
- The scores are very accurate in the lattice as in the original lattice
- There won't be any duplicate paths in the lattice

More details of generating the lattice can be found in [2].

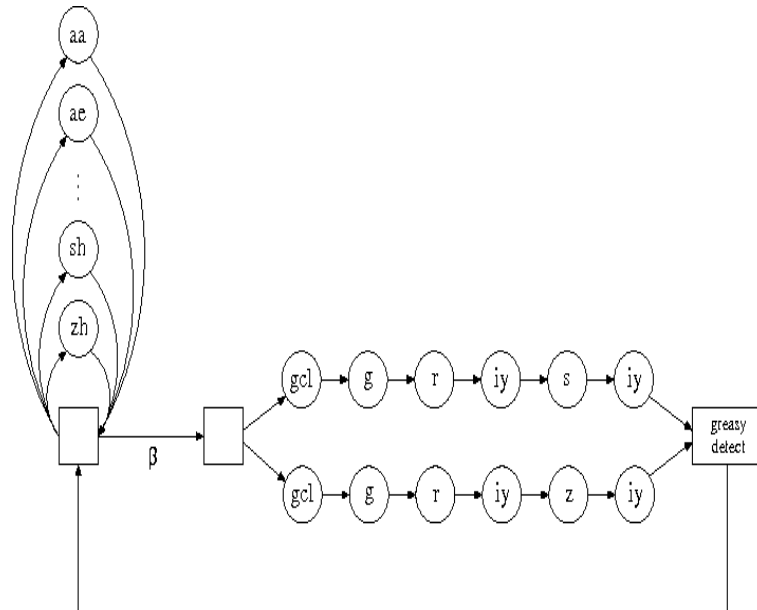


Figure 4.3: Keyword-Filler Model

4.3.2 Keyword Search

Once the lattice is generated if its word lattice, then it will be grep on lattice arcs. The main drawback of this would be not to able to identify out of vocabulary words. If its a phone lattice we convert out query in to phone fst. A linear mapping its pronunciation will be indexed[] in the lattice. Its matches the arcs with query pronunciation, if its below certain threshold then we say its a hit, other wise miss. Example lattice has been shown in figure 3.

4.4 Keyword-Filler based Methods

As lattice indexing is offline method, it requires the keyword spotting method to be online for some applications. For this instead of building a entire lattice over phones, we build linear pronunciation for keywords. All the pronunciation variations are also included. The other words which are not keywords are modeled by using phonetic loops. If the likelihood being a keyword more than its being a any other word then we say its hit other wise. The problem is every word can be modeled by using phonetic loop, so we favor keywords with a penalty. This makes when ever a phone sequence is close to keyword then it chooses keyword path than the phonetic loop. This method is very fast in real time. But it may not provide as much as accuracy of that lattice indexing. This has been illustrated in figure

Chapter 5

Sequential Architectures

This chapter is a preliminary for the next chapter. All the networks are used in the proposed approach are sequential architectures. So it would be good to look through the architectures one. The whole chapter is inspired from[17]. Here we will discuss briefly about recurrent neural networks(RNN) and long short term networks(LSTM) and bi directional architectures and at the end we will go through encoder-decoder architectures.

5.1 Recurrent Neural Networks

RNNs are proposed to process sequential data. As speech sequential data, they are ubiquitous in every application of speech. In DNNs, it is not possible to have memory, each example is treated independently. Which results in sub-optimal performance. TDNNs have limited memory, beyond their context they also treat each example independent. Here in RNNs is to treat each example dependent on the previous examples. One way to incorporate this information to have feed from the previous examples. RNNs exactly do the same thing, they have feed back from previous hidden units as input. This keeps the information flowing from one example to next. Mathematically RNNs can be represented as

$$\begin{aligned} a^{(t)} &= b + W.h^{(t-1)} + Ux^{(t)} \\ h^{(t)} &= \tanh(a^{(t)}) \\ o^{(t)} &= c + V(h^t) \\ \hat{y}^{(t)} &= \text{softmax}(o^{(t)}) \end{aligned} \tag{5.1}$$

The loss function is defined as follows, each output at time t is conditioned on inputs till t th time

$$\begin{aligned} &= Lx^{(1)}, \dots, x^{(t)}, y^{(1)}, \dots, y^{(t)} \\ &= \sum_t L^{(t)} \\ &= - \sum_t \log p_{model}(y^{(t)} | x^{(1)}, \dots, x^{(t)}) \end{aligned} \tag{5.2}$$

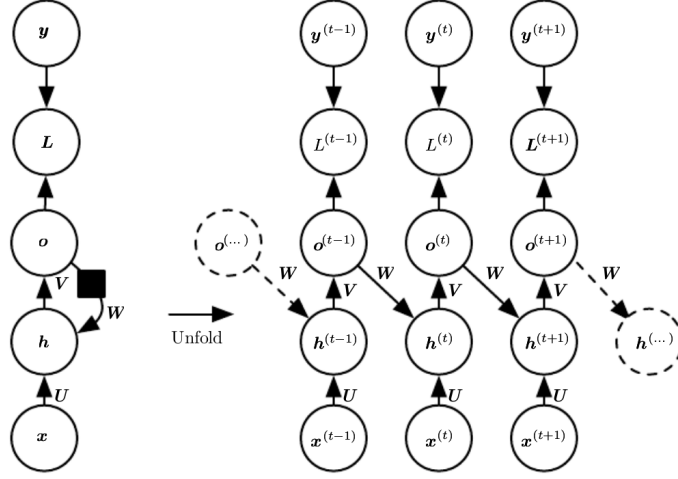


Figure 5.1: RNN block diagram[17]

5.2 Long Short Term Memory

The drawback of RNNs is that the gradient vanishes or explodes quickly. This is due to long depth in the network. This problem is more profound in addition to the network spacial depth it has depth in time. When we apply back propagation through time the error get multiplies and depends on the range it either vanishes or explodes. To combat this problem the authors[] in have introduced LSTMs. The fundamental is idea to keep the error constant, and this is done by using constant error carousal(CEC). The CEC will have linear network but it wont be able to learn all the variations or memory simultaneously. To have non-linearity's gates has been introduced, namely Forget gate, Output gates, and Input gate. Depending on the application there are changes made to these gates. The mathematical formulation of LSTM is as follows

f indicates for Forget gate and i indicates input gate and o indicates output gate in following equations

$$\begin{aligned}
 f_i^{(t)} &= \sigma(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}) \\
 s_i^{(t)} &= f_i^{(t)} s_{i-1}^{(t-1)} + g_i^{(t)} + (\sigma(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)})) \\
 g_i^{(t)} &= \sigma(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}) \\
 h_i^{(t)} &= \tanh(s_i^{(t)}) q_i^{(t)} \\
 q_i^{(t)} &= \sigma(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)})
 \end{aligned} \tag{5.3}$$

Above **b,U,W** bias and input and recurrent weights for each gate

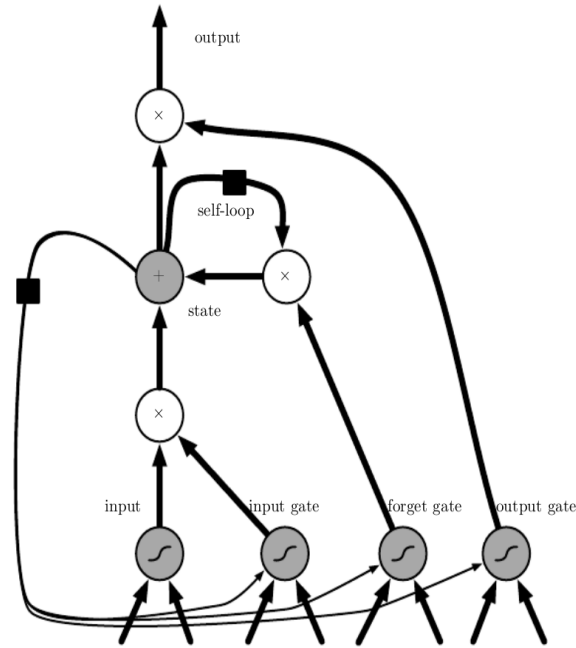


Figure 5.2: LSTM block diagram[17]

5.3 Encoder-Decoder Architecture

The encoder decoder[18] contains two networks. Encoder network process the entire input and gets a fixed length representation, and the decoder network takes the fixed length representation process the fixed length input to produce the out put. The fixed length representation is obtained from the final state of the encoder architecture. Both of these architectures are one of the sequential architectures. The main advantage is that it process variable length inputs to get fixed length representation. This allows to process variable length examples. The encoder decoder diagram has depicted in figure.

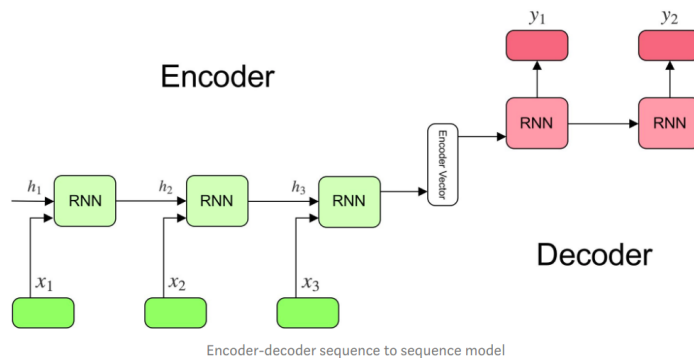


Figure 5.3: Encoder Decoder architecture[19]

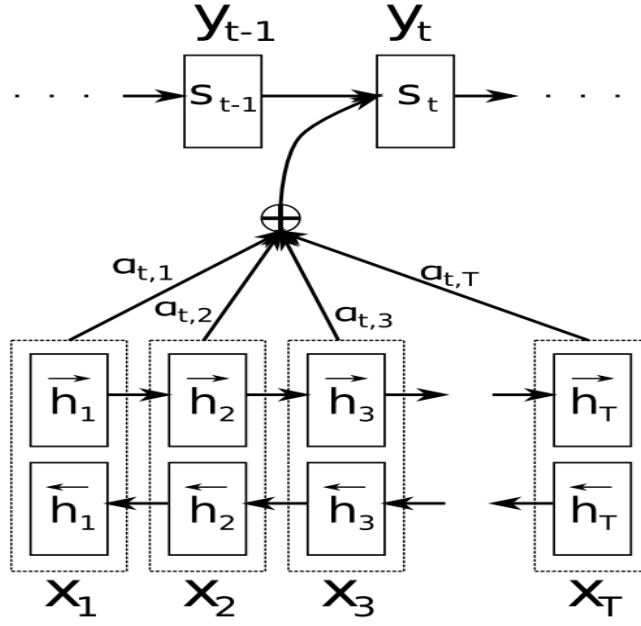


Figure 5.4: Attention Block diagram

5.4 Attention Mechanism

The problem with encoder-decoder architecture is that the fixed length representation it get is only of final state. The information in the input can be distributed across time. Though the network is sequential if the time is more then the information loss will be more. To avoid this the authors[20] have introduced attention mechanism to encoder-decoder network. The attention mechanism identifies which part of the input is relevant to the output and weights them accordingly. There several ways to compute the attention, some of them listed here. The block diagram with attention mechanism is shown in figure2. Lets say \mathbf{x}, \mathbf{y} as input and output targets and \mathbf{h} as the hidden representations,

$$\begin{aligned}
 \mathbf{x} &= [x_1, x_2, \dots, x_n] \\
 \mathbf{y} &= [y_1, y_2, \dots, y_m] \\
 \mathbf{h}_i &= [\vec{\mathbf{h}}_i^T; \overleftarrow{\mathbf{h}}_i^T]^T, i = 1, \dots, n
 \end{aligned} \tag{5.4}$$

Attention is given as

$$\begin{aligned}
 \mathbf{c}_t &= \sum_{i=1}^n \alpha_{t,i} \mathbf{h}_i \\
 \alpha_{t,i} &= \text{align}(y_t, x_i) \\
 &= \frac{\exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_i))}{\sum_{i'=1}^n \exp(\text{score}(\mathbf{s}_{t-1}, \mathbf{h}_{i'}))} \\
 \text{score}(\mathbf{s}_t, \mathbf{h}_i) &= \mathbf{v}_a^T \tanh(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i])
 \end{aligned} \tag{5.5}$$

Chapter 6

Proposed Approach

The main draw back of the traditional approach is that they require sufficient amount of transcribe data. Which may not available for all the time especially in low resource scenario. To overcome that problem End-to-End ASR free keyword spotting has been proposed.

6.1 End-to-End ASR free Keyword spotting

6.1.1 Baseline system

The baseline system considered here is [11].The baseline system we considered here is end-to-end ASR free keyword spotting from speech proposed in [18]. It consists of three modules an Acoustic encoder and a Phonetic encoder and a Keyword Neural network. It has been inspired from traditional ASR keyword spotting where it contains an acoustic model and a language model and keyword search algorithm. As we can see, one can find one to one corresponding between Acoustic Encoder and Acoustic model like wise. The acoustic encoder is an auto encoder, it takes speech features as input and projectes into fixed length representation and then try to reconstruct the signal back. As speech is a time dependency signal the best to realise this auto encoder would be a sequential architecture. In a similar means Phonetic encoder or Char RNNLM will take the keyword phone sequence and gets fixed length embedding. The acoustic auto-encodere and phonetic auto-encoder is trained with mean square and cross entropy loss respectively. Once these auto-encoders get trained we remove the decoder parts to use in the keyword spotting. From these we get corresponding representation for speech and keyword, then these two representation or concatenated to form input of KWS(neural network). The KWS neu- ral network is a feed-forward neural network. Now it considered as one joint network and trained for keyword occurrence with cross entropy loss. The draw back of this approach is that it wont consider temporal information present in speech signal as it is lost when it gets to fixed length representation. So we wont be knowing where the keyword is occurred. The other problem is that it try to project the phonetic features in to acoustic space with keyword occurrence loss. Which is very difficult to achieve. In thesis these are the problem that we try to addressed. We follow the same architecture as of the base line system.

6.2 Proposed Architecture

6.2.1 Acoustic Encoder

Here unlike the baseline system the acoustic encoder is not trained to be auto encoder. Only the encoder part is used in the network. Motivated by [20], its sequential network with gated recurrent units(GRU) as hidden units and tanh activation function. Lets say x_1, x_2, \dots, x_T as speech features, these features are fed into the network and it produces h_1, h_2, \dots, h_T as the hidden representations. These entire representations are used as each this will contain information speech signal and one final representation is may not sufficient to represent the reference audio. By doing this temporal information remains intact. And further these encoder representations are used to compute the attention with keyword embedding.

6.2.2 Phonetic Encoder

The phonetic encoder is a bidirectional sequential network with GRU layers. Each keyword will be represented in terms of its phonetic pronunciation. These phone sequence will be fed in to phonetic encoder as an embedding. These embeddings are obtained by using a mapping matrix. The final hidden representation will be used as the representation for keyword. This is used to compute attention along with acoustic hidden representation.

6.2.3 Attention

The attention [20] is used to identify the location of the query. Its computed between acoustic hidden features and keyword embedding. Lets say the acoustic hidden features as h_1, h_2, \dots, h_T and keyword embedding as w_N the attention is computed as follows

$$\begin{aligned} e_i &= w_N^T h_i \\ \alpha_i &= \frac{\exp(e_i)}{\sum_j \exp(e_j)} \end{aligned} \tag{6.1}$$

From the attention weights we compute context vector from acoustic representations

$$C = \sum_i \alpha_i * h_i \tag{6.2}$$

This context vector and phonetic representation are concatenated together to form input for KWS neural network.

6.2.4 KWS Neural Network and Cost Function

KWS neural network identifies whether the keyword is there in the audio reference or not. Here we used the same architecture as in the baseline. The KWS network is a classifier, which takes the input after the attention module which is concatenation acoustic context vector and keyword embedding vector. For each audio utterance and given keyword, the target is annotated as either yes or no. And finally the network is trained jointly with acoustic encoder and phonetic encoder with cross entropy.

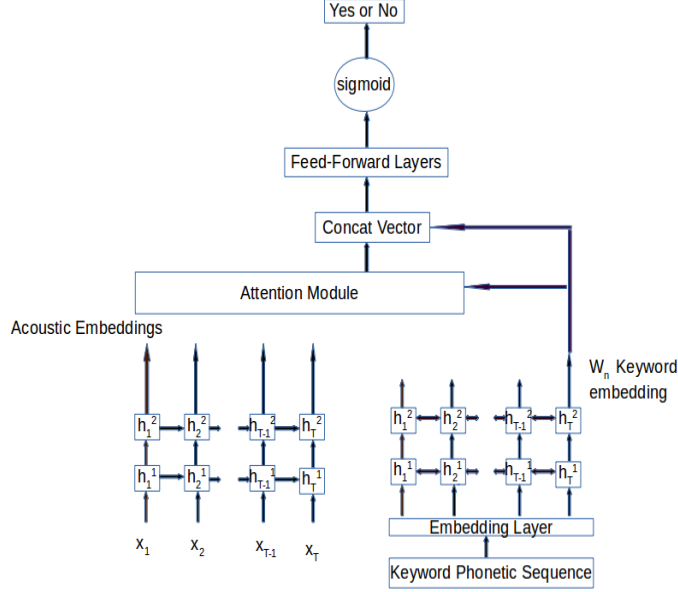


Figure 6.1: End to End ASR free Keyword Spotting Architecture

In summary, acoustic features are fed into the acoustic encoder, keyword phonetic pronunciation phonetic encoder, and attention between these two representations will be computed, to get a context vector. The context vector and keyword embedding will be concatenated together and gives it to the KWS neural network for the classification. The entire KWS network is depicted in figure 1. But still acoustic representations and keyword embedding are very much far apart, that is evident from training loss in shown in figure 2. To address this we propose to pre-train phonetic encoder with speech synthesizer.

6.3 Transfer Learning from Speech Synthesizer

In previous methods the audio reference has always been converted to text to make queries and references into the same domain. But it requires a sufficient amount of transcribed data. While converting query into audio would not required as much of a data as ASR. Besides, most of the languages have more or less a similar number of phonetic sounds. So we propose to pre-train the phonetic encoder with speech synthesizer[21] this will make the phonetic encoder aware of acoustic projection, which makes the comparison easier. The actual speech synthesizer system contains several components, and mainly it contains duration modeling and acoustic modeling.

6.3.1 Duration Modeling

The duration model is a regression between input phone and the duration of the phone. To get the duration information, first we will align it through HMM-GMM forced alignment. This gives the information of where each phone starts and where it ends. Here duration modeling is done by using a bidirectional recurrent neural network(RNN)[22] after getting duration information. The network takes one hot encoded phones as input and duration in frames as output. The network is trained with mean square error(MSE) loss function.

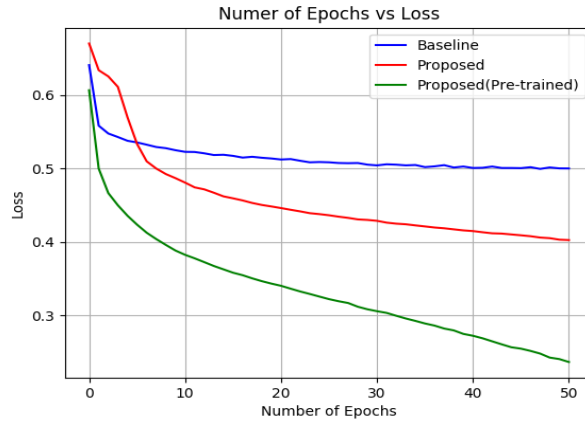


Figure 6.2: Trend in training loss for end-to-end free asr KWS network

6.3.2 Acoustic Modeling

Acoustic model learns the distribution of each phone in terms of acoustic features. For acoustic modeling, along with the phone, the duration will be concatenated at input level to learn the Mel Frequency Cepstral Coefficients(MFCC). In addition, to retain the speaker characteristics band aperiodicity and fundamental frequency will be used in the network. Same as the duration modeling, a bidirectional RNN[22] is used to train the acoustic model. Each phone with left right context and duration will be given as input and at the output MFCC,band aperiodicity and fundamental frequency is taken. MSE loss is used to train the network.

We can see that acoustic modeling is more appropriate for our application. In this acoustic modeling network, we remove the input layer and output layer and transfers the hidden layers to the phonetic encoder in end-to-end ASR keyword spotting. The text query is not completely converted to audio domain, its the knowledge transfer from acoustic modeling that helps the phonetic encoder to learn acoustic characteristics and to converge faster. The same thing is observed in training and shown in figure 2. The loss significantly comes down. Usually speech synthesizer trains for a single speaker application which may limit transfer-learning but with keyword occurrence classification, we can expect the network to learn speaker-independent characteristics. By following this, we require relatively less amount of transcribed data, and we also trained it with a different language to avoid same language transcribed data dependency.

Chapter 7

Experimental Setup

7.1 Experimental Setup

7.1.1 Database Description

The database used in this experiment is Telugu data of 40 hours. This data was part of the Microsoft India language challenge[23]. It's consists of clean and read speech data, with 44k utterances and 48k lexical words. From this data, we prepared 56k positive examples and 56k negative examples for 10k keywords, which occurred more than five times in the training data. Each example will have a utterance and a keyword associated with it. If the keyword is present in the utterance the label will be 1 otherwise 0. The number of keywords can be extended to any number of keywords and made vocabulary independent. The test has 3k utterances from this we have made 5.6k positive and 5.6k negative examples. Though we need to create more negative examples because of the nature of the problem, we can show that performance won't affect much, even after increasing the number of examples. For feature engineering, we have used Mel frequency ceptral coefficients(MFCC). We have extracted 40-dimensional MFCC features for each 25 milliseconds window with 10 milliseconds frame shift. The features are generated from the Kaldi tool kit [24]. All the experiments are developed in pytorch[25].

7.1.2 End-to-End ASR free Keyword Spotting

Baseline System The original baseline system is proposed for Georgian language of Babel project. Here the experimented language is Telugu, but given both databases of same of size, and parameters from the original paper, we have quoted best obtained results for the experimented database. The acoustic auto encoder is a uni-directional GRU with 300 hidden neurons layer. The network is trained with adam optimiser of initial learning rate $1 * 10^{-3}$, and whenever validation loss is not decreased the learning rate is halved. The character RNNLM consist of convolutional encoder of 300 mask with size of $50*3$. The decoder is uni directional GRU followed by soft-max layer of 60 units. The same criterion for learning rate is followed for character RNNLM. The KWS neural network trained with 256 hidden layer with input of 600, resulted from concatenating representations from both acoustic auto encoder and character RNNLM. The whole network is also trained with same learning rate criterion. **Proposed Approach** For the acoustic encoder, we have used a fully

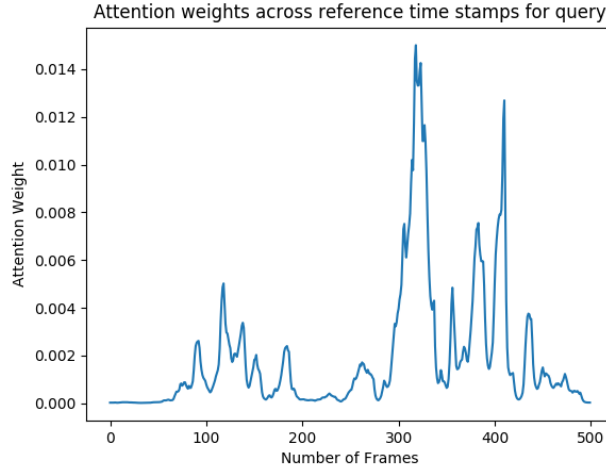


Figure 7.1: Attention weights vs Reference time. High attention for query actual location

connected 3 layer uni-directional GRU network. Each layer has 256 GRU hidden units. Input is a 40-dimensional MFCC feature vectors. The longest utterance of length 2200 frames, so all the utterances are padded to make it equal length. As a consequence, we have unrolled then network for 2200 time steps.

The phonetic encoder is a bidirectional GRU network. It has 2 layers of each 256 hidden units. Each keyword is converted to its phonetic pronunciation. The most extended keyword has length of 25 phones. These phones are commonly occurring phones across all Indian languages. An embedding layer of 60*100 dimension is used to get the continuous representations. The same phonetic encoder architecture used is pre-trained with speech synthesizer acoustic modeling. For that, we have used open-source CMU arctic data[26], which has around 1.5-hour English and consist of 1150 phonetically balanced utterances.

Attention between acoustic embedding vectors and keyword embedding is used to get the context vector. The context vector and keyword phonetic embedding vector will be projected to 200 dimension each and concatenated to make a single vector. The KWS neural network has 2 successive hidden layers of 256 each, and the final single dimension project layer to classify the keyword occurrence. The entire system is jointly trained with SGD optimizer of learning rate of 0.01 and trained for 40 epochs. For the supervised keyword spotting, we have built a time delay neural network(TDNN)[13] based speech recognition and took the best path for locating the query. It would be unreasonable to compare with lattice indexing, which is a very sophisticated algorithm. Nevertheless, we have considered the best possible acoustic model to quote the results.

For OOV words, We considered 550 words apart from 10k words from the training. From these we have considered 10% in magnitude of test size and obtained 534 positive example and 534 negative examples.

Table 7.1: Performance evaluation of systems. All number are quoted in-terms of accuracy

Method	In Vocabulary	OOV	T@3(In Vocab)
ASR based	92.96	NA	NA
Baseline	68.52	49.34	NA
Proposed	64.90	51.59	51.79
Proposed(Pre-trained)	75.54	53.37	55.93

7.1.3 Results and Discussion

The proposed pre-trained network converges faster and to a better loss as shown in Figure 2. With pre-trained model there is an absolute improvement of 7% compared to the baseline system in terms of accuracy. For OOV words also the improvement is consistent. We can observe that there is 4% absolute improvement over the baseline. While computing locations, instead of just considering maximum attention value, we have considered top 3 maximum attention locations and check whether they coincide with the actual location. Because acoustic encoder is a sequential network, every frame will have the essence of previous frames. There by attention will not be centered around one particular area. The same thing can be observed in figure 3. High attention is obtained where query is actually located, and the next best values are happened after the query location. We have referred it as T@3 in the results table. While computing second best we excluded 20 frames right and left of the peak, because of the smooth nature of attention function. We can see that almost 56% of them matching the location of the query with the best system. In terms of accuracy without pre-training with speech synthesizer is closer to the baseline, and with pre-training, it brings closer to the supervised system.

7.2 Conclusion

In the thesis, we have proposed to use transfer learning from speech synthesizer and an attention module for ASR free end-to-end keyword spotting. Experimental studies suggest that this direction of keyword spotting can be done with minimal supervision, one can achieve reasonable performance. The time to get labeled data also significantly less and no level of expertise is needed to get the labeled data. And also this method reduces memory footprint and can be trained efficiently. There is lot of room for improvement and especially in confining attention to one particular time instance. If we achieve this, it will be good alternative for traditional keyword spotting architectures with less supervision.

References

- [1] D. Yu and L. Deng, *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.
- [2] D. Can and M. Saraclar, “Lattice indexing for spoken term detection,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, pp. 2338–2347, Nov 2011.
- [3] J. Foote, S. J. Young, G. J. Jones, and K. S. Jones, “Unconstrained keyword spotting using phone lattices with application to spoken document retrieval,” *Computer Speech & Language*, vol. 11, no. 3, pp. 207–224, 1997.
- [4] R. C. Rose and D. B. Paul, “A hidden markov model based keyword recognition system,” in *International Conference on Acoustics, Speech, and Signal Processing*, April 1990, pp. 129–132 vol.1.
- [5] I. Szoke, P. Schwarz, P. Matejka, L. Burget, M. Karafiát, M. Fapso, and J. Cernocky, “Comparison of keyword spotting approaches for informal continuous speech,” in *Ninth European conference on speech communication and technology*, 2005.
- [6] Y. He, R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw, “Streaming small-footprint keyword spotting using sequence-to-sequence models,” in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 474–481.
- [7] C. Shan, J. Zhang, Y. Wang, and L. Xie, “Attention-based end-to-end models for small-footprint keyword spotting,” *arXiv preprint arXiv:1803.10916*, 2018.
- [8] H. Zhang, J. Zhang, and Y. Wang, “Sequence-to-sequence models for small-footprint keyword spotting,” *arXiv preprint arXiv:1811.00348*, 2018.
- [9] M. Sun, D. Snyder, Y. Gao, V. K. Nagaraja, M. Rodehorst, S. Panchapagesan, N. Strom, S. Matsoukas, and S. Vitaladevuni, “Compressed time delay neural network for small-footprint keyword spotting,” in *INTERSPEECH*, 2017, pp. 3607–3611.
- [10] 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 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 5280–5284.
- [11] 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*, vol. 11, no. 8, pp. 1351–1359, 2017.

- [12] D. Jurafsky, *Speech & language processing*. Pearson Education India, 2000.
- [13] V. Peddinti, D. Povey, and S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [14] J. Keshet, D. Grangier, and S. Bengio, “Discriminative keyword spotting,” *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.
- [15] G. Chen, C. Parada, and G. Heigold, “Small-footprint keyword spotting using deep neural networks,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4087–4091.
- [16] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian *et al.*, “Generating exact lattices in the wfst framework,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4213–4216.
- [17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [18] I. Sutskever, O. Vinyals, and Q. Le, “Sequence to sequence learning with neural networks,” *Advances in NIPS*, 2014.
- [19] “MS Windows NT kernel description,” <https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346>, accessed: 2010-09-30.
- [20] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [21] H. Zen, K. Tokuda, and A. W. Black, “Statistical parametric speech synthesis,” *speech communication*, vol. 51, no. 11, pp. 1039–1064, 2009.
- [22] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, “Tts synthesis with bidirectional lstm based recurrent neural networks,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.
- [23] B. M. L. Srivastava, S. Sitaram, R. K. Mehta, K. D. Mohan, P. Matani, S. Satpal, K. Bali, R. Srikanth, and N. Nayak, “Interspeech 2018 low resource automatic speech recognition challenge for indian languages.” in *SLTU*, 2018, pp. 11–14.
- [24] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motliceck, Y. Qian, P. Schwarz *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.
- [25] B. Steiner, Z. DeVito, S. Chintala, S. Gross, A. Paszke, F. Massa, A. Lerer, G. Chanan, Z. Lin, E. Yang *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [26] J. Kominek, A. W. Black, and V. Ver, “Cmu arctic databases for speech synthesis,” 2003.