Voice Recognition on Simple Microcontrollers

Callie Y. Kim
University of Maryland,
College Park
ckim1123@cs.umd.edu

Diana Chou
University of Maryland,
College Park
dchou@cs.umd.edu

Geng Liu
University of Maryland,
College Park
leoliu@cs.umd.edu

ABSTRACT

Advancements in voice and speech recognition in consumer equipment have been possible with the development of signal processing, machine learning, and low-cost embedded processors such as microcontrollers. In this paper we review steps towards developing a system for small vocabulary recognition on low-cost microcontrollers such as Arduino. To do so, the method of speech recognition must be limited in its complexity to be programmed onto a standard microcontroller device, in order to accommodate for the limitations of memory and computing power in the hardware components. In this work we review approaches towards voice and speech recognition, present our system design and evaluation, and discuss limitations and potential for future work.

1. INTRODUCTION

Automatic speech recognition, also known simply as speech recognition, is one of the most adapted techniques in communication and speech processing areas, where a computer is trained to understand a user's speech and convert it into a series of words, thus creating a network of interactions between humans and machines. Speech recognition can also be defined by a signal of frequencies emitted as a function of time. Speech processing techniques such as speech synthesis and processing [8] as well as speaker identification and verification [4] have made it possible to develop voice interfaces or perform voice interaction with devices. Voice recognition can be applied in several applications such as: voice services (smartphone automation, home appliances), health (prosthetic arms, wheelchairs, disability services), data quality control (data, training), vocal dictation (speech-to-text software), and other innovative technologies (robotics, AI).

The development of computational tools with an objective for automation in industrial and domestic products have led to innovation in powerful embedded microprocessors. In contrast, a microcontroller is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. Most programmable microcontrollers that are used today are embedded in other consumer products or machinery including phones, hand-held devices, automobiles and household appliances for computer systems. When designing a system to be used primarily on a microcontroller there are several advantages and disadvantages to consider. One major advantage is the low cost and small size of components. Since microcontrollers are fully integrated onto one chip, they are cheap to manufacture. Microcontrollers typically have much lower specs than even a low-power consumer grade general CPU and has a generally standardized architecture, making them even more easy to mass produce. Also, microcontrollers are very flexible, due to their programmable nature and integration for additional RAM, ROM and I/O ports. While there are several advantages to using microcontrollers, the technology can be limited for what developers want to achieve too. The amount of memory available for data and program manipulation limits microcontrollers' performance, in that it can only understand a limited number of commands, and these commands are specific to the functions the device is designed to handle. Furthermore, microcontrollers cannot interface high power devices directly and have a more complex structure compared to microprocessors.

Our objective is to be able to implement a voice recognition system on simple microcontrollers. The use of low-cost microcontrollers causes a reduction in system complexity and limits the selection of variables and methods.

In summary, this paper makes the following contributions:

- We explore various approaches to implement speech recognition.
- We analyze and discuss the limitations of speech recognition model in TensorFlow lite
- We experimentally deploy CMUSphinx to resource-constrained microcontrollers and address issues that occurred.

2. A PRIMER ON THE PREREQUISITES

2.1 Phonemes, Hidden Markov Models, and Gaussian Mixture Models

Speech recognition can be performed at three different levels : signal level, phoneme level and word level. The process can use complex modeling systems, involving mathematical functions and probabilistic models to determine the intended word or phrase. One such technique is the Hidden Markov Model, which is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (i.e. hidden) states. In speech recognition, a phoneme is treated as a link in a chain, and the completed chain represents a word. A phoneme is any of the abstract units of the phonetic system of a language that correspond to a set of similar speech sounds which are perceived to be a single distinctive sound in the language. To determine the next phoneme, the chain forms branches of different sounds that can come next, a probability score is given to each branched off phoneme based on the built in dictionary. Thus, the complete word is finally determined.

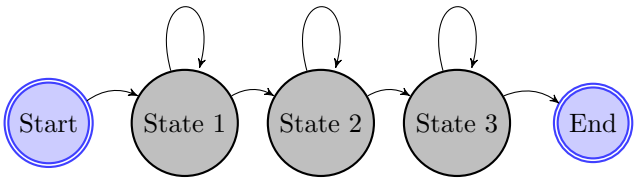


Figure 1: Hidden Markov Model for a Phoneme h.

A Gaussian Mixture Model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. It is used to represent the various modalities in speech recognition. It could also be used to create a unique voice print for each speaker. Most phonemes occupy separate or partly separate sub-spaces as shown in [12], and the consonants show more separation than vowels, revealing Gaussian like distribution. The paper discusses an improved training procedure for speech recognition, in which phonemes are manually extracted from the training sequence and used to form Gaussians individually.

2.2 Mel-frequency cepstrum(MFC)

We need to pre-process the audio signal in order to produce an accurate representation of the phoneme spoken such as extracting features to identify the key components of the audio signal and reduce background noise. The Cepstrum is a sequence of numbers that characterise a frame of speech which is a result of running Inverse Fourier Transform(IFT) on logarithm of the estimated signal spectrum. Mel-frequency cepstrum, especially Mel-frequency cepstral coefficients (MFCCs) are commonly used as features in speech recognition systems. The reason why MFC is commonly used in speech recognition is because the frequency bands are scaled on the Mel scale. Mel scale is a perceptual scale, constructed that sounds with equal distance to each other in the Mel scale implies that those would be heard the same in distance to the listener. Thus, MFC is appropriate to extract features by mirroring the human auditory systems.

2.3 Linear Predictive Coding(LPC)

Linear Predictive Coding is also another useful method for encoding quality speech at a low bit rate by analysing speech signals and estimating formats. The basic idea of the LPC model is that a speech sample can be approximated as a linear combination of the past speech samples, as a difference equation and called a linear predictor. Thus, while MFC extract features by using a perceptual scale that approximates a human auditory system, LPC predicts future samples based on past speech samples. Due to its linear nature, LPC is an appropriate method to compress signals such as speech coding.

2.4 Speaker Identification

So far, we have covered approaches towards speaker-independent speech recognition systems, which are designed to recognize anyone's voice without additional training. In addition, there are also speaker-dependent systems which work by learning the characteristics of a single person's voice, in order to identify that speaker from other users. Speaker identification is a complicated factor that falls into speaker-dependent voice recognition systems. Characteristics such as speech generation (source of the voice) and the envelope behavior (vocal and nasal tract) affect the production of speech that occurs. According to [4], all speaker recognition systems contain two main phases: feature extraction and recognition. During the first phase, a training vector is generated from the speech signal of the word spoken by the user, which are stored in a database for subsequent use in the recognition phase. During the recognition phase, the system tries to identify the unknown speaker by comparing extracted features from password with the ones from a set of known speakers. In their speaker recognition system, a Welch algorithm was applied to a speech signal to estimate the power spectrum density, and then a cosine distance was used to measure similarity between vectors in order to apply a matching algorithm between the signal and the speaker. In our system, we will be focusing on speaker-independent voice recognition.

3. INTUITION AND FEASIBILITY STUDY

Training and testing on phoneme recognition could improve speech recognition accuracy and [8] gives a strong insight to this. The authors used a 3-state HMM model for phoneme recognition and argues that they were able to achieve up to 100% accuracy in

recognizing the phonemes, thus being a cornerstone approach for implementing complex speech recognition systems. Hence, CMUSphinx[1] implements speech recognition using HMM and recently added support for phoneme recognition to PocketSphinx decoder. Thus, we chose PocketSphinx as an approach to implement speech recognition on microcontrollers.

Our project aims to combine the dynamic technological developments of voice and speech recognition with low-cost, single chip, self-contained computer systems such as microcontrollers. These boards are embedded inside countless everyday items, such as wearables, drones, toys, household appliances, and more. There is high practicality in our project implementation, and the area has been researched in recent developments such as [7], [11], and [3]. In order to develop the system, we needed to acquire a few technical components, such as the Adafruit EdgeBadge and Arduino microcontroller. We also needed a considerable amount of time to design, develop, train, and evaluate out models over a 2-month time period.

4. SYSTEM DESIGN

4.1 TensorFlow Lite

As a first step to develop speech recognition on microcontrollers, we utilized TensorFlow lite[2] which is an open source platform for machine learning on microcontrollers. Performing machine learning on microcontrollers has an advantage such as low latency since an Internet connection is not required therefore there is no round-trip to a server. Figure 2 is the model architecture we used in TensorFlow lite.

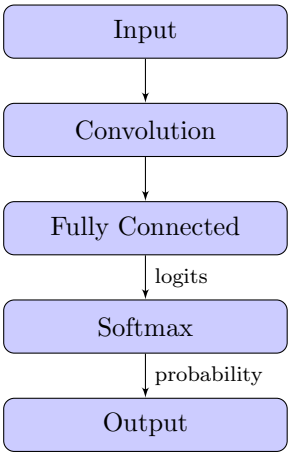


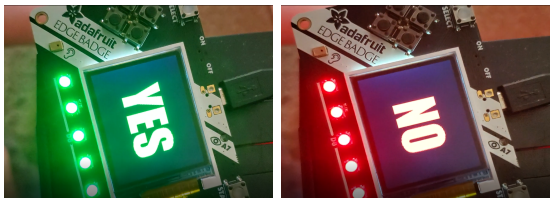
Figure 2: Model Architecture

The model comprises of a convolutional 2D layer, a fully connected layer and a softmax layer where the outputs are probabilities given raw audio sample data as an input.The model was derived from [6] and it aims to be used as a basic pipeline that could run on resource-constrained microcontrollers. The model uses a multiclass classification to recognize keywords from the speaker.

Since there is a CNN, the model transforms raw audio sample data to spectrograms which are two dimensional arrays. The transformation is performed by slicing the audio sample data each taken from a different time window. The window length is 30ms and FFT is performed when creating each frequency slice. Each result of FFT has 256 entries and 6 entries are grouped together, creating 6 frequency buckets. Weights and biases are added in the convolutional layer and fully connected layer where the activation function was Rectified Linear Unit(ReLU) as it is the simplest non-Linear function appropriate for resource-constrained microcontrollers.

There 15,000 training loops in total and the learning rate was 0.001 for the first 12,000, and 0.0001 for the final 3000. After the training was finished, the model size was 50.74kB. This is larger compared to the model that was trained for 2 words since it uses 20kB of RAM. We converted the TensorFlow Lite model into a C source file that can be loaded by for the microcontroller.

We deployed the model both on Adafruit EdgeBadge[3] which has 512KB of flash and 192KB of RAM and Arduino[4] but the model is also deployable to SparkFun Edge, ARC EM SDP, etc. The device will wait until the speaker says the 2 keywords trained for the model which are "Yes" and "No". The device will print out the results on the screen and turn on the NeoPixel LEDs if it recognizes the keywords as you can see in Figure 3. After confirming that the model actually performs speech recognition, we trained the model again and extended the vocabulary to recognize 10 keywords instead of 2 : "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go". The data we used to train the model was obtained from a dataset in this paper [9], which consists of over 105,000 WAVE audio files of people speaking 30 different words.



(a) Yes recognized (b) No recognized

Figure 3: Result of Speech Recognition

[1]CMUSphinx is an open source speech recognition toolkit https://cmusphinx.github.io/

[2]https://www.tensorflow.org/lite/microcontrollers

[3]https://www.adafruit.com/product/4400

[4]https://store.arduino.cc/usa/nano-33-ble-sense

4.2 CMUSphinx

In initialization, CMUSphinx[5] takes input from command line arguments and initialize all the fields based on provided arguments or DEFAULT values. During data processing, CMUSphinx takes audio inputs, convert the inputs to frames and then applies Mel Scale and computes Mel-Frequency Cepstral Coefficients (MFCC). MFCC is used to represent the feature of the speech. As soon as there are some frames computed, typically around 5, they are used to feed into N-gram search which uses HMM to find the best matched word and its score.

Their library works on Unix (including Raspberry Pi), IOS, Windows, and Android systems. We were focusing on pocketsphinx[6], which is a light weighted library that uses CMUSphinx for speech recognition. As moving pocketsphinx to Arduino, we need to unwrap all the command line argument part to reduce the amount of code. Arduino does not have a file system, we need to remove all the file-IO and replace it with some data structure in C.

5. EVALUATION

The result of training model with TensorFlow shows in Table 1 with various number of keywords. The accuracy decreases as number of keywords increases. Since we never have a complete system of CMUSphinx implemented on Arduino, we could not evaluate its accuracy.

KeyWord Number	Accuracy	Test set size
2	91.7%	1236
10	75.3%	4726
30	63.5%	10229

Table 1: Training result with TensorFlow.

6. RELATED WORK

Keyword spotting systems have a small memory footprint and low computational power thus making it possible to run on mobile devices. Due to this nature, neural networks have become an attractive choice for KWS architecture. In [6], the authors explore a small-footprint keyword spotting task by using Convolutional Nerual Networks(CNN). They compare CNNs to DNNs by limiting number of multiplies or parameters since previous research such as [1] showed that using CNN for KWS improves performance and reduced model size when convolution is applied along the frequency axis. For the CNN, the model has one convolutional layer and the output of this convolutional layer is passed to a linear low-rank layer and then 2 DNN layers. The reason why the model has only one convolutional layer instead of two is because the device is power-constrained. Multiple convolutional layers could exacerbate number of multiplies across time, frequency and feature maps which is not suitable for microcontrollers. As one of our approach to explore various methods to implement speech recognition on microcontrollers, we train a TensorFlow lite model roughly derived from this architecture as a ground-work for our study.

Continuous research has been done to improve KWS. In the paper [2], the new architecture improves the accuracy by 3.4% with 60% fewer parameters and 50% fewer operations compared to the current state of the art for KWS applications. The architecture is a compact binary architecture using binary orthogonal codes to analyse speech features from a voice command. The paper uses the same dataset we used to train the TensorFlow lite model and their model size was lower than 30kB, thus showing an opportunity to deploy speech recognition even with microcontrollers with less memory then our approach. One of the key difference that made this possible was the network architecture. Instead of learning convolutional filters directly it learns it indirectly by combining weighting coefficients of deterministic binary basis linearly. For further work, we could extend the current TensorFlow lite model by adopting their network architecture, mitigating memory constrained problems when using microcontrollers for machine learning.

7. LIMITATIONS AND DISCUSSION

TensorFlow Lite model we trained has limitations because it uses multiclass classification to recognize words and results with slow computation, low accuracy when the number of words increase. We were not able to complete a full system that can run on Arduino based on CMUSphinx since we turned our focus from TensorFlow lite to CMUSphinx mid-way during the semester. CMUSphinx is a complex and complicated library, relying heavily on file IO such as using memory-mapped files. However, all of those are supported by Linux kernel but not on Arduino. Arduino is not capable to run a kernel since it can not process file IO. Thus, to convert the platform the code needs to run we need to restructure components that require file IO in CMUSphinx.

In [10], the authors introduce a noise suppression filter to enhance speech recognition on microcontrollers. Background noise can introduce recognition errors thus developing a noise suppression filter is important to improve the accuracy of speech recognition. Especially, microcontrollers have limited power and resource to implement speech recognition thus recognition errors are more likely to occur when the environment is noisy. Thus, for future work we could extend our study by implementing a noise suppression filter for microcon-

[5] `https://cmusphinx.github.io/`

[6] `https://github.com/cmusphinx/pocketsphinx`

trollers to improve the accuracy of the TensorFlow lite model.

There has been prior research on how to utilize speech recognition on microcontrollers and one study was an application of a voice-activated powered wheel chair control [11]. The research of how to improve voice controlled wheel chair was continued recently [3]. The goal was to implement a voice controlled wheel chair prototype with low cost to assist people with disabilities. Conventional wheel chairs use a joystick input to control direction and movement which limits access for disabled people, especially people who have difficulty controlling their hands.Hence, in the case of disabled users speech is an input worth exploring even for microcontrollers as it could be the only remaining medium to communicate such as patients with severe spinal cord injuries. Also, paper [7] introduces an application of speech recognition deployed on microcontrollers by controlling movement of a mobile robot through limited voice commands. Speech recognition could facilitate communication with robots, an important feature to consider in Human Robot Interaction(HRI). The area where voice command control using microcontrollers can be even extended to everyday lives such as electrical household appliances [5].

Thus, even though microcontrollers have limitations in memory and computing power, it is worth to continue the exploration of implementing voice recognition on microcontrollers in various areas.

8. REPRODUCIBILITY INFORMATION

All of code we have used can be found at Google Drive[7] License are included either in the directory or on top of each file.

Upload the code under /Speechrecognition/-TensorFlow/micro_speech/main.cc to Arduino nano BLE 33 Sense[8] board, it should be able to recognize "yes" - green light, "no" - red light, and unknown - blue light.

Training code is under TensorFlow/micro_speech/train, the data set is under same directory in a folder named dataset. We have run the training code with Jupyter Lab with Anaconda. All the required python packages are included inside requirements file.

Install sphinxbase[9] to install sphinxbase library, then install pocketsphinx [10]. Then follow the instructions under CMUSphinx tutorials site[11], /Speechrecongnition/CMUSphinx/example/hello_ps.c should be able to compile and run, which recognize the words in goforawrd.raw file containing speech "go forward ten meters"

Folder /Speechrecognition/example/Voice_with_Sphinx contains the code that transformed from CMUSphinx and pocketSphinx library with modification to run on Arduino, it currently does not compile and still working in progress.

[7] https://drive.google.com/drive/folders/1wnlHxrm3tkQFrOikMfsx8THPAhi73ZCY?usp=sharing
[8] https://store.arduino.cc/usa/nano-33-ble-sense
[9] https://github.com/cmusphinx/sphinxbase
[10] https://github.com/cmusphinx/pocketsphinx
[11] https://cmusphinx.github.io/wiki/tutorialpocketsphinx/

9. REFERENCES

[1] Abdel-Hamid, O., Mohamed, A., Jiang, H., and Penn, G. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012), pp. 4277–4280.

[2] Fernández-Marqués, J., Tseng, V. W.-S., Bhattachara, S., and Lane, N. D. On-the-fly deterministic binary filters for memory efficient keyword spotting applications on embedded devices. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning* (2018), pp. 13–18.

[3] Hou, T. K., Yagasena, and Chelladurai. Arduino based voice controlled wheelchair. *Journal of Physics: Conference Series 1432* (jan 2020), 012064.

[4] Maazouzi, A., Aqili, N., Raji, M., and Hammouch, A. A speaker recognition system using power spectrum density and similarity measurements. In *2015 Third World Conference on Complex Systems (WCCS)* (2015), pp. 1–5.

[5] Novani, N. P., Hersyah, M. H., and Hamdanu, R. Electrical household appliances control using voice command based on microcontroller. In *2020 International Conference on Information Technology Systems and Innovation (ICITSI)* (2020), pp. 288–293.

[6] Sainath, T. N., and Parada, C. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association* (2015).

[7] Thiang, D. W. Limited speech recognition for controlling movement of mobile robot implemented on atmega162 microcontroller. In *2009 International Conference on Computer and Automation Engineering* (2009), pp. 347–350.

[8] Veeravalli, A. G., Pan, W. D., Adhami, R., and Cox, P. G. A tutorial on using hidden markov models for phoneme recognition. In *Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 2005. SSST '05.* (2005), pp. 154–157.

[9] Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition, 2018.

[10] Yan Chan, K., Nordholm, S., Yiu, K. F. C., and Togneri, R. Speech enhancement strategy for speech recognition microcontroller under noisy environments. *Neurocomput. 118* (Oct. 2013), 279–288.

[11] Yi, J. Z., Tan, Y. K., Ang, Z. R., and Panda, S. K. Microcontroller based voice-activated powered wheelchair control. In *Proceedings of the 1st International Convention on Rehabilitation Engineering Assistive Technology: In Conjunction with 1st Tan Tock Seng Hospital Neurorehabilitation Meeting* (New York, NY, USA, 2007), i-CREATe '07, Association for Computing Machinery, p. 67–72.

[12] Zhang, Y., Alder, M., and Togneri, R. Using gaussian mixture modeling in speech recognition. In *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing* (1994), vol. i, pp. I/613–I/616 vol.1.