# Speech Commands in Embedded Device as a Smart-home Device

1st Fadlullah Raji
*department of Computer Science)*
*Middle Tennessee State University*
Murfreesboro, Tennessee
far2i@mtmail.mtsu.edu

2nd Austin Fine
*department of Computer Science*
*Middle Tennessee State University*
Murfreesboro, Tennessee
adf5f@mtmail.mtsu.edu

3rd Mena Bushra
*department of Computer Science*
*Middle Tennessee State University*
Murfreesboro, Tennessee
Mnb4u@mtmail.mtsu.edu

4th Mohamed Yusuf
*department of Computer Science*
*Middle Tennessee State University*
Murfreesboro, Tennessee

***Abstract*—The world in the 21st century has moved drastically towards automaton and smart devices. We have seen the Manufacturing industry, healthcare, education and several professional spaces expanded rapidly with the introduction of automation and smart device. We have also seen how our hendheld devices is being smart enough to regonize simple commands. What we have yet to see is a complete smartness of our home. This project intoduces a smart home devices that can help turn on appliances with only our speech commands and apply commands locally without internet or cloud access. We build a model that simply recognizes 5 words ”on”, ”off”, ”light”, ”heater” and ”fan” and was deployed on an arduino device for operation. Results were great and at 90 percent of the time, we were able to turn these appliances on and off with the recognized words.**

***Index Terms*—Machine Learning, Embedded Systems, Neural Network, Signal Processing**

## I. Introduction

From generation to generation, the people around the globe have evolved and adapted their life around technology. Whether it is smart devices, televisions, or even simple appliances like your everyday coffee machine, we continue to grow in our understanding of how we can develop and manipulate technology to assist in our everyday lives.

Often times these modern technologies are intelligently controlled and interconnected with each other. The Amazon Echo, for example, is a device that responds to one's voice demands by answering users' questions, streaming music, and interacting with other smart home devices [1]. The global market for smart home devices is rapidly expanding and most of these devices support some form of speech recognition. However, most speech recognition software utilizes cloud-based solutions using deep Neural Networks to recognize and process user's speech commands [1]. With cloud-based server, data is transferred back and forth between the device and the cloud. This sometimes introduces issues like privacy of data. A typical example is a hidden Microphone reported by users of Google's home alarm system [2]. Performing this computation and process on the device though would eliminate some of the challenges and issues associated with cloud computing such as data security and privacy [3].

All this being considered, this research aims to develop a machine learning model as a typical smart home device that understands and performs an action upon sensing its command from a microphone and the model will be run on an embedded system where users won't be worried about their privacy or the other challenges and issues associated with cloud computing.

## II. Background

In the field of neural network technologies, speech recognition has fallen behind compared to other areas in the tech industry such as language processing or image recognition [4]. In this research, we plan to build a Convolutional Neural Network (CNN) model that will add insight to the field of neural networks and improve some of the challenges currently faced in the industry such as those that are associated with cloud computing.

Similar model has been designed by [5] for a limited vocabulary words with no specific application. Another model designed by [6] uses a streaming audio samples that introduces a new methodology of training speech recognition model. This project is different from existing model because we have considered a specific application towards home appliances. We will be modelling a speech recognition device that helps users turn on and turn off their light, fan and heater.

Conventionally, Kaldi has been used for speech recognition model [7]. However, differently, we have decided to use Convolutional Neural Network (CNN) to recognize speech commands because a CNN can reduce the spectral variation, model spectral correlations that exist within an audio signal, and is relatively simple to implement [8].

## III. Methods

### A. Overview

In order to successfully implement a Machine learning model, several architecture and preprocessing steps has to take place. At the "big picture" level, these processes include; compiling the datasets to be used for training; preprocess; validating; and testing the model. After the model has achieved theoretical accuracy on the testing split of the dataset, we then convert and migrate this model to the embedded device. The choice of device used in this project is the Arduino NANO-BLE due to its compact and mobility. We trained the model using Python3 and TensorFlow on a desktop computer equipped with dual RTX 2070 super CUDA enabled graphics cards working in scalable link interface (SLI) and 32 gigabytes of memory, and the compiled data sets consisted of a total of 107,365 audio samples of 38 unique labels. However, because we needed to implement this on an embedded system with much lower hardware capabilities, we trimmed our model into a Tiny-CNN (learning a very small number of parameters) and convert model into TensorFlow Lite-Micro model. We also reduced the size of our dataset to include only the audio labels we intended to test for and all other labels goes into the unknown category Those intended labels were "on", "off", "heater", "light", "fan", "silence", and "unknown". By doing this we effectively decreased the size of our CNN model and made it easier to deploy to the embedded device.

### B. Dataset

The datasets used in this project were TensorFlow's built-in Speech Commands dataset, combined with recorded audio samples for all of the major labels that we needed since they are not available publicly. To speed up the process of recording our audio samples we used the TinyML speech recording web-page and pete warden's [5] tool to trim only 10 milli-seconds of the loudest part of every audio samples. This software allowed us to record multiple audio samples quickly, verify the general quality of our recordings, and download them as Ogg files. Afterwards we compiled all of these recordings into one large data set and converted them from Ogg files to Wav files to match the same audio format from the TensorFlow Speech Commands dataset. After these conversions were complete, we combined TensorFlow's Speech Command dataset and our dataset into one large dataset. Finally, we split the combined data sets into three separate sets: a training set consisting of 75,098 audio samples, a validation set consisting of 22,529 samples, and a testing set consisting of the remaining 9,656 audio samples.

### C. Preprocessing

To preprocess an input data from a sensor, we have to watch out for several nuances that happens for a time series data. The sensor microphones collects the Sound wave and converts them into electrical signal in the form of voltage distortion which will be, in the case of a Microcontroller sampled at 16KHZ (i.e 16,000 data points per second). This data is required to be fed into our neural network as fast as possible. The preprocesing of the input will be looking at how our data input can be differentiated from each other. We first convert from a time domain into a frequecny domain where the signals can then be transformed into a human differentiable images called the spectrogram. Neural Networks itself mimics how the human brain works to solve a problem. To mimic this on the neural netwrok architecture, we have to be fed with an human like input for proper differentiation of the audio samples.

For a long time, Mel-Frequency Cepstral Coefficients (MFCCs) were a common feature; however, filter banks have recently gained popularity. The signal from the microphone is pre-emphasised, then sliced into (overlapping) frames and a window function added to each frame; after that, we perform a Fourier transform on each frame (or, more precisely, a Short-Time Fourier Transform) and measure the power spectrum; and finally, we compute the filter banks. A Discrete Cosine Transform (DCT) is applied to the filter banks to obtain MFCCs, with a subset of the coefficients retained and the remainder discarded. Mean normalization is a final step in both cases. The mathematical computation is discussed in [9].

As a Neural Network software, Tensorflow has an Api for running through this filter banks. We employed tensorflow micro processing for audio preprocessing and runs through the dataset to prepare the audio datasets for training. we provided window strides of 480 and window size of 320 with a sampling frequency of 40. The output from the preprocessing is is 40 X 49 2-dimensional array ready to be fed into the network for training and prediction.

### D. Model Architecture

The general model architecture is a Convolutional Layer with Maxpool layer and a softmax output layer to predict input into 38 number of classification. To simulate the input from the arduino microphone, we flattened the preprocesed data in training into a 1960 1-dimension data input. The first layer of the model is then a reshape layer to put it back into a 2d spectrogram of image (40X49). The output from the reshape layer is then fed into the a conv2D layer with 128 number of filters and striding window size is 1 with Rectified linear activation function(RELU). The first convolutional layer feeds into a dropout layer which helps prevent the model of overfitting the training dataset. This dropout layer is then connected to a max pool layer that will help prevent the model from becoming too sensitive to the locations of detected features within the spectrogram and will help improve the model's generalization. The training step uses a Gradient Descent approach with 256 batch size for training at a time. we used a scheduler for the learning rate to start with an high value and a very low value in the last episodes of learning.

After training, the model is required to be converted into a format compatible with the Arduino since ardunio cannot be programmed in python3. The RAM and Memory on the arduino is only limited. We employed Tensorflow Method with the TfLite and TfLite-Micro for conversion into a compatible model for the arduino device. However, this process requires quantization - a process of reducing a float values of the weigths into an int8 to reduce their memory usage

## IV. Results

Model was trained for 15000 epochs, and achived 87 percent accuracy on the validation at 15000 epochs. However, the validation accuracy was 89.80 percent and the training accuracy was 94.50 percent at 13,000 epochs as depicted in "Fig. 1". we reload The testing samples performed similarly with a validation accuracy of 87.70 percent and with 87.57 percent model accuracy. After training, the model is required to be converted into a format compatible with the Arduino since ardunio cannot be programmed in python3. The RAM and Memory on the arduino is only limited. We employed Tensorflow Method with the TfLite and TfLite-Micro for conversion into a compatible model for the arduino device. However, this process requires quantization - a process of reducing a float values of the weigths into an int8 to reduce their memory usage. TFLite resulted in 93.98

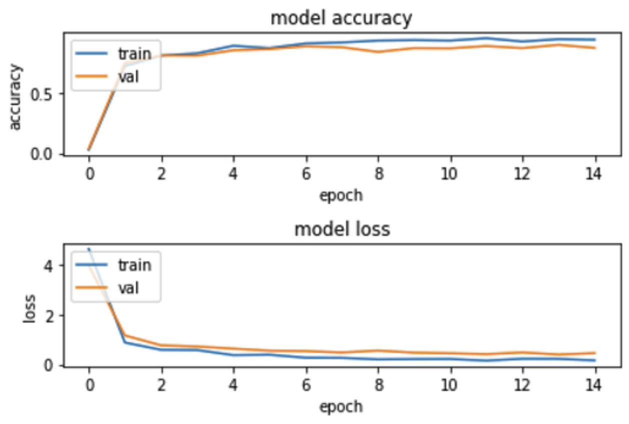

Fig. 1. Validation/Training Accuracy and Loss vs Epochs (x1000).

At the time of writing the results, the ON/OFF commands were successful in powering the three LEDs. Two out of the three LEDs performed properly and heeded to their corresponding commands. These two LEDs were LIGHT and HEATER. The third LED for FAN didnt respond very well to its command and we plan to increase the audio samples in the FAN label as there are few number of audio samples. This being said, the final configuration and testing of the product are currently in progress and we anticipate promising results that show the success of the implemented speech recognition software with all 3 LEDs.

## V. Discussion and Conclusion

We developed a model that was tested to help turn on and off basic appliances in a residential with publicly available datasets and included a recorded limited number of audio samples for our specific dataset. We tested the model under a validation files and testing files and we achieved interesting results. In the future, we hope that researchers could expand this commands to several other appliances found in a typical residential homes so that we can just not only not achieved an Industry 4.0 ( a typical generational name for the current technological revolution) but also automate our home activities

## References

[1] M. Wang, T. Sirlapu, A. Kwasniewska, M. Szankin, M. Bartscherer, and R. Nicolas, "Speaker recognition using convolutional neural network with minimal training data for smart home solutions," pp. 139–145, 07 2018.

[2] CNNET, "Google calls Nest's hidden microphone an 'error' kernel description."

[3] A. Mondal, S. Paul, R. Goswami, and S. Nath, "Cloud computing security issues challenges: A review," pp. 1–5, 01 2020.

[4] Z. Song, "English speech recognition based on deep learning with multiple features," *Computing*, vol. 102, pp. 1–20, 03 2020.

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

[6] K. Kim, K. Lee, D. Gowda, J. Park, S. Kim, S. Jin, Y.-Y. Lee, J. Yeo, D. Kim, S. Jung, J. Lee, M. Han, and C. Kim, "Attention based on-device streaming speech recognition with large speech corpus," 2020.

[7] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit," 2019.

[8] A. Hernandez, F. Reyes Jr, and A. Fajardo, "Convolutional neural network for automatic speech recognition of filipino language," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, pp. 34–40, 03 2020.

[9] H. Fayek, "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between kernel description."