Speech Recognition for Edge Devices for Maxim Integrated

By

Dylaan Cornish

Henry Fernandez

Joseph Tomal

Supervisor: Donald Patchell

A Capstone Project

Submitted to the University of Chicago in partial fulfillment
of the requirements for the degree of

Master of Science in Analytics

Division of Physical Sciences

October 2022

Abstract

Maxim Integrated has requested the assistance of University of Chicago MsCA students to integrate Audio Recognition into the company's new chip called the MAX78000 that will be installed into edge devices. This chip has increased power capacities and privacy, and with our applications of Deep Learning we can train the chip to understand key words as well as the recognition of a "wake up" word when spoken to.

Keywords: Deep Learning, Fully Connected Neural Network, Convolutional Neural Network, Auto Encoder, Edge Device, Mel Frequency Cepstral Coefficients, Audio Recognition, Virtual Machine, Cloud Computing

Executive Summary

In this paper, our team worked with Maxim Integrated, a company that designs manufactures, and sells analog and mixed-signal integrated circuits for the automotive, industrial, communications, consumers, and computing markets. Maxim Integrated is developing a state-of-the-art microchip called the MAX78000. The MAX78000 will be more power efficient and secure than past models and competing chips. By pairing the Chip with a complex Deep Learning Neural Network framework and inserting the chip into an edge device, we can perform Low Power & Max Security Speech Recognition.

The requirements Maxim Integrated presented to us are as follows:

1. Convert Audio Waves into Recognizable Speech
2. Reliably Detect the Utterance of a Wake-Up Word
3. Meet the Power and Size Constraints of the Chip
4. Produce less than ⅛ False Alarms per Hour

Table of Contents

List of Figures

Introduction

Founded in 1983 by nine pioneers of the semiconductor industry, Maxim Integrated designs, manufactures, and sells analog and mixed-signal integrated circuits for the automotive, industrial, communications, consumer, and computing markets. The company is headquartered in San Jose, California but has design centers manufacturing facilities worldwide. In August of 2021, the company was acquired by Analog Devices.

In regards to our contribution, Maxim Integrated is developing a new microchip called the MAX78000 that is very power efficient but has limitations. Once the MAX78000 is inserted into an edge device it can control the speech recognition process. It can do so at a higher cost savings and maximum security and privacy. An edge device example could be a tablet, smartphone, router, or modem.

Problem Statement

Voice recognition is a tool used widely in a variety of industries. Most people are familiar with using Alexa, Google Home, and other similar systems that are able to understand audio commands. These speech recognition products work by using the internet to upload speech to the cloud, then use machine learning techniques to determine if a "wake-up word" was spoken. This introduces a large concern about the privacy of our daily conversations. Maxim Integrated wants to take this model and make it work locally, so that no audio data is sent over the internet until a wake-up word is detected.

However, this approach poses a problem. In order to do this locally, audio data must be transformed into features that describe the audio, then used as inputs in a machine learning model. Calculating these audio features is a complex and power consuming process, and as a result, it cannot be done locally. Maxim Integrated has designed a new microchip called the MAX 78000. The chip is able to take advantage of deep learning to make predictions locally. Maxim Integrated envisions a model that can estimate the complex feature extraction as this will be much more power efficient while not sacrificing accuracy too greatly. Once we have estimated features, we can use this data as input to another model that detects which word has been spoken, and will thus be able to determine if a wake-up word was uttered and the device should wait for a query.

Analysis Goals

Maxim Integrated has provided us with pre-set goals for our analysis. They are as follows: 1.) Train a neural network on the chip that meets the power constraints that allows the MAX78000 to work efficiently. 2). Reliably detect the utterance of a wake-up word. A wake-up word is a custom word that will wake up the edge device once the audio is processed. This could be something similar to Apple's “Hey, Siri” or Google's “Ok, Google”. 3.) Classification of up to 20 key-words and background noise. 4. Less than ⅛ false alarms per hour of audio.

Scope

Initially, we will train our models using 20 keywords and 15 rejection words using the Google Speech Commands Dataset. Once this task is done, the scope increases considerably as we will train our models using unlabeled data on the much larger Librispeech audiobook dataset. We can use any audio data to train our model, so while these two datasets are our focus, Maxim Integrated can further develop these models after this work concludes.

Our scope of model types are somewhat limited by the inner workings of the MAX 78000. Recurrent neural networks (RNNs) can be used for audio processing as language is sequential and RNNs have memory that allows them to retain information from the past. However, the MAX 78000 is unable to work with any type of RNNs, so we are primarily exploring fully connected networks (FCNs), convolutional neural networks (CNNs), and autoencoders.

Background

Rather than use audio waves directly in speech recognition, a common approach is to calculate a mel-frequency cepstral coefficient (MFCC). This is a compressed representation of audio that functions as the features that describe that audio. Many speech recognition frameworks first calculate MFCCs, then feed them into a machine learning algorithm to classify speech. MFCC calculation requires calculating Fourier transforms, discrete cosine transforms, and applying mel-filter bands. As alluded to above, this MFCC calculation is inefficient and expensive, and Maxim Integrated are looking to replace this using deep learning.

The primary software we will be using is the Pytorch machine learning framework. Pytorch is primarily a machine learning framework based upon the Torch Library,used for applications such as computer vision and natural language processing and in this project for Audio Recognition. The architectures we are creating in Pytorch are primarily Fully Connected Neural Networks (FCNN), Convolutional Neural Networks and Variable Autoencoders. FCNNs are simple, feed forward neural networks in which all nodes in one layer are connected to all nodes in the neighboring layers. Convolutional Neural Networks (CNN) are a class of artificial neural networks, most commonly applied to analyze images. A CNN can take an input image, assign importance to it through learnable weights and biases to various aspects/objects in the image and be able to classify one image to another. Each model architecture type is described in the figures below, respectively.

Figure 1. *Fully Connected Neural Network Architecture*

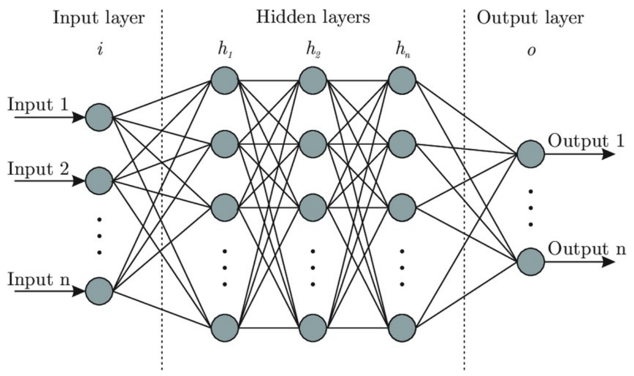


Figure 2. *Convolutional Neural Network Architecture*

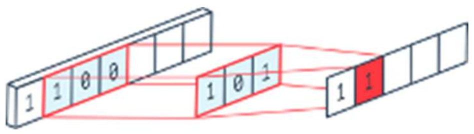
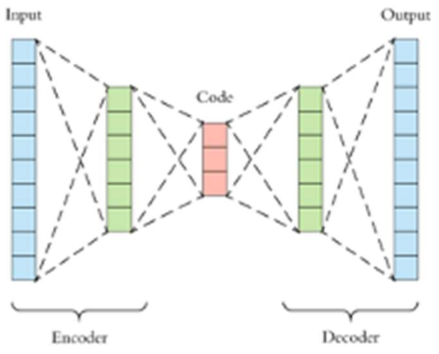


Figure 3. *Auto Encoder Architecture*



Variable Autoencoders are an advanced version of traditional Autoencoders. A traditional Autoencoder is an unsupervised learning technique in which we leverage neural networks for the task of representing data in various forms. By imposing a bottleneck in the network it forces a compressed knowledge representation of the original input. In our project, we use the autoencoder to compress down the audio of one word into a bottleneck encoding to create features of the audio in a smaller form which allows for more efficient training. Variable autoencoders improve our task by enabling the training to be regularized to avoid overfitting and ensure that the latent space has good properties that enable generative processes.

Literature Review

Reviewing the current literature in the field of audio machine learning was crucial to build our understanding of how to practically use audio data in the context of machine learning. Much of the nuance of using audio data comes in the preprocessing step. As audio data are dense with information, it often becomes necessary to window or segment data into smaller and smaller pieces. Especially when working with longer audio clips, they are typically broken into smaller and smaller units called clips, shots, and frames.

In addition to windowing, most research involves extracting features from the dense audio. This can be done using mel-frequency cepstral coefficients as inputs to a neural network, as these features provide a concise way to model audio (Tripathi et al., 2019). Similarly, previous work has been done to use plain spectrograms as input to a CNN to monitor the and determine the state of beehives based on the recorded sound (Kulyukin et al., 2018). Phonetic posteriorgram (PPG) is a way to express class probabilities in a given timeframe (Lin et al., 2021) and has been used in tasks such as accent conversion (Zhao et al., 2018) and even cross-lingual voice conversion between English and Mandarin (Zhou et al., 2019).

Many types of models have been trained to handle speech data. Traditional ML algorithm have previously been the standard for audio classification tasks. Hasegawa-Johnson et al. (2006) show that support vector machines (SVM) and Guassian mixture model-hidden Markov models (GMM-HMMs) are able to classify some dysarthric speech patterns, but only in certain cases.

The current standard is to use some form of artificial neural network (ANN) as these have shown experimentally to perform better than traditional machine learning algorithms. CNNs paired with various types of audio features are a common approach as noted above.

Data

Data Sources

The first dataset we use to train our models is Google Speech Commands (GSC). GSC contains over 100,000 one-second long utterances of 35 words. 20 words are considered keywords, and the other 15 words constitute a rejection class. GSC separates the files with 9981 clips designated for validation, and 11005 for testing, leaving nearly 85,000 training audio samples. All audio clips are sampled at 16kHz. This dataset represents the foundation of how we train our models. They will learn to process the audio into features, then use those features to predict which word is being spoken.

The second datasource we utilize is Librispeech. Librispeech contains a corpus of approximately 1000 hours of 16kHz read English speech. An in depth description can be found in the paper by Panayotov et al (2015), who prepared the dataset. The speech comes from audiobooks from the LibriVox project, which includes many hundreds of thousands of words to train our models with. The audio data has been aligned and segmented by the authors, as well as split into smaller subsets of 100, 360, and 500 hours to be more manageable to work with.

Descriptive Analysis

The Google Speech Commands dataset does not require much exploration. It is straightforward audio data, all one second long clips and all sampled at 16 kHz. One caveat to mention is that the distribution of clips is not equal; some words have more audio clips than others. However, all words have a substantial number of clips that is more than enough to train a machine learning model to recognize each word.

Methodology

We used the University of Chicago Midway supercomputer as our first computing option. Eventually, we required more computing resources and tried using Google's Colab Pro, but ultimately ended up using Microsoft Azure as our cloud computing platform. We trained our models using GPUs in parallel to speed up the training process.

Feature Engineering

Training a feature extractor required using audio waveforms as input and calculated MFCCs as output. We use the *librosa* Python library to calculate the MFCC using 256 fast Fourier transforms and a hop length of 128. As a result, MFCCs are calculated with overlap. We calculated 20 coefficients for each audio, resulting in a matrix of size (20, 126). For the input audio, we slide along with overlap to calculate the MFCCs so that each audio sample of length 256 has a corresponding MFCC frame of length 20 that the neural network will learn to predict.

For the keyword classification, our inputs are the MFCC matrices and we are predicting the class label of each word in the Google Speech Commands dataset. We already have the calculated MFCCs from the feature extractor. For the labels, we simply need to use scikit-learn's *LabelEncoder* class to encode them as numbers that a neural network can use.

We also experimented using quantized audio. To quantize the audio, *n* equally spaced values are selected and the audio is put through a step function ensuring that every value is equal to one of the discrete values that were chosen. As quantization will likely happen on the chip, we need to see if limiting the values the audio waveform could take would lead to a sizeable drop in accuracy, which we discuss in the Findings section.

Modeling Frameworks

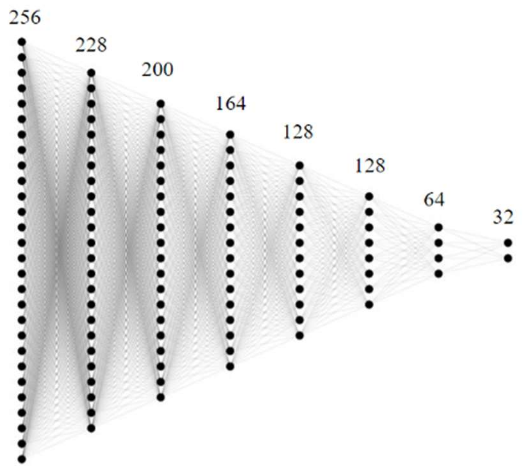
In this section we describe modeling frameworks for both the feature extractor and the keyword classifier as different approaches were used.

Feature Extractor

We experimented with three models for the feature extractor: convolutional neural networks (CNN), autoencoders, and fully connected neural networks (FCNN). Due to the size of the dataset and training time required, we started training with only two words to experiment with various architectures within each of the three models. We trained our models using the Adam optimizer and trained for at least 200 epochs each time. We used mean squared error (MSE) as our loss metric, and once we reached an acceptable MSE with a given architecture, we moved on to training it with data for five words. We repeated this process until we reached training on all words and continually updated architectures and hyperparameters until we discovered the best MSE through training. On the full dataset, the model yielding the lowest MSE was the FCNN, and this is our proposed solution as an MFCC feature extractor. The model is fairly simple, comprising seven dense layers and ReLU activation functions

Below details a figure of our fully connected neural network with the amount of neurons per layer.

Figure 4. *Fully Connected Neural Network Final Architecture*

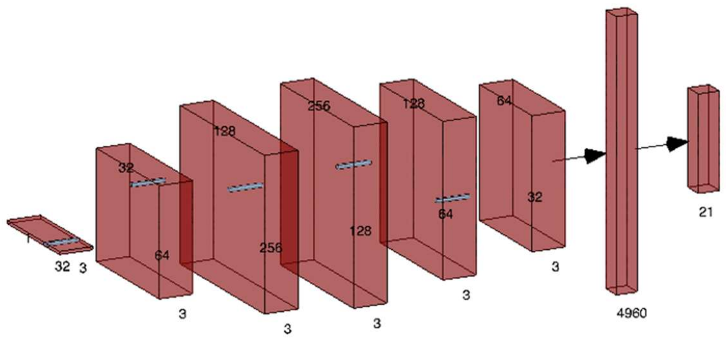


Training was split between local and cloud services, making use of GPUs as we included more data.

Keyword Classifier

The keyword classifier takes MFCC matrices as input and outputs one of 21 class labels, 20 for the keywords and one for a general rejection class. We experimented with several different CNN architectures to try to minimize cross entropy loss. Training was primarily done using cloud GPUs to minimize training time. Our final CNN framework has seven convolutional layers and one fully connected layer at the end. After each convolution is a dropout to avoid overfitting and a ReLU activation function. The convolutional layers follow a reverse butterfly structure, first increasing in size through the initial four layers, then decreasing in size for the last three layers. The model contains over 879,000 parameters and is trained for 100 epochs. Below details the final model of the CNN classifier with number of layers and sizes of each convolution.

Figure 5. *Convolutional Neural Network Classifier*



Findings

For the MFCC estimator, we used raw audio waves as input and tried to predict calculated MFCCs. After trying numerous different architectures, our best performing FCNN ended with an MSE of 14.35. The best autoencoder had an MSE of 20.37, and the final CNN had a much higher 104.96 MSE. MSE is used as it shows us how close to a perfect MFCC estimate our model can get, with an MSE of zero being a perfect reproduction.

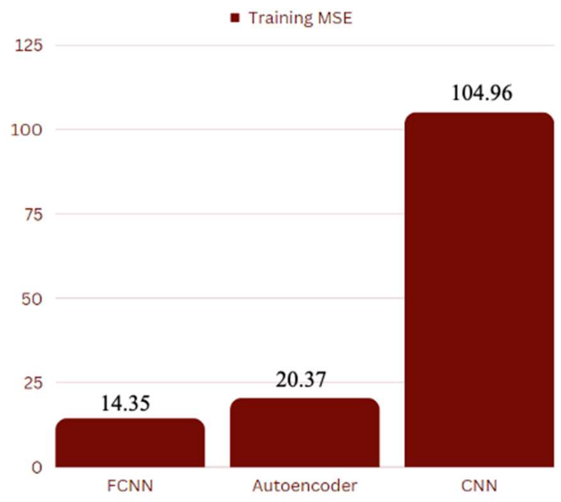
The second part of our framework is the keyword classifier. We fit all the Google Speech Commands data using the CNN described in the Methodology section. This model ultimately achieved 92.5% accuracy classifying words from their calculated MFCCs.. We also trained the model using the MFCCs from the quantized audio data, and this resulted in a very similar 91.5% classification accuracy.

When combining the two models, we are able to estimate MFCCs from audio, then use the estimated MFCCs as input to the classifier to get an accuracy for the entire process. Using the

FCNN for MFCC estimation provided the highest accuracy for the full process, reaching 92.0% accuracy on the full dataset. The CNN only achieved 80% accuracy, and the autoencoder similarly underperformed, only reaching 81% accuracy.

Below a figure details the Mean Squared Error between the three model types.

Figure 6. *Model Training Mean Squared Error*



Discussion

As mentioned in Results, our best MFCC estimator was the fully connected neural network. This came as a surprise as we were expecting the CNN to perform best. One possible explanation for the lower performance of the CNN could be that the CNN took much longer to train than the other models. Even using cloud computing resources, the CNN trained about an order of magnitude slower than the other models. This meant that in our time constraints, we were unable to experiment with as many architectures and tune hyperparameters to the same extent. There may be a CNN architecture that outperforms the FCNN, but we have not found it yet and this is an area that is worth continuing research.

For the keyword classifier, a 92% classification accuracy is in line with what we expected to achieve as well as what Maxim Integrated outlined. This is within the range of other state of the art speech recognition models. In addition, we see that quantizing the audio degrades the classifier by a very small amount, just one percent. The MAX 78000 will likely quantize the audio when our model is embedded into the chip, and our result leaves us optimistic that our model will reliably classify keywords even with quantized audio.

Conclusion

In conclusion, our locally calculated speech recognition solution ended with a 92% accuracy rate on 20 words and a rejection class of 15 words. The final model was the two part architecture of a fully-connected neural network that used raw audio as input and outputs estimated Mel-Frequency Cepstral Coefficients, combined with a convolutional neural network that inputs the estimated MFCCs and outputs the correctly classified word.

The next steps will be a proper documentation and file handoff to Maxim Integrated that contains all items pertaining to the speech recognition solution. These files include but are not limited to code we have written that shows model architectures and how we trained them, files that contain our pretrained models, as well as word documents that detail the processes needed to get our code up and running. This will most likely be done in the Github environment which enables seamless code and file sharing.

Now that this algorithm has been proven to work with high accuracy, it will be handed off to Maxim Integrated to be productionalized onto the MAX78000 chips. Maxim Integrated will then train our speech recognition algorithm on an expanded dataset that captures the necessary commands for each MAX78000 microchip depending on each mico-chips use case.

Likely, a MAX78000 being sold to a Smart Hearing Aid device company will contain a model that has been trained on a dataset containing the command words that are associated with the hearing-aids functionality. Similarly, the model on the MAX78000 sold to a smart thermometer device company will be trained on a subset of words pertaining to the functionality of a thermometer. Ultimately, the takeaway is that Maxim Integrated will have the capability to train our model on larger, more specific datasets to fit their products needs.

Once the MAX78000 is in production it will be the most efficient microchip on the market that does not upload your data over the internet. Companies will take advantage of the low power consumption of the chip as well as its minimal latency and consumers will take advantage of the speech privacy. Our speech recognition solution paired with the MAX78000 microp-chip has the potential to revolutionize the smart-device industry.

References

Kulyukin V, Mukherjee S, Amlathe P. Toward Audio Beehive Monitoring: Deep Learning vs. Standard Machine Learning in Classifying Beehive Audio Samples. Applied Sciences. 2018; 8(9):1573. https://doi.org/10.3390/app8091573

Lin Y-Y, Zheng W-Z, Chu WC, Han J-Y, Hung Y-H, Ho G-M, Chang C-Y, Lai Y-H. A Speech Command Control-Based Recognition System for Dysarthric Patients Based on Deep Learning Technology. Applied Sciences. 2021; 11(6):2477. https://doi.org/10.3390/app11062477

Hasegawa-Johnson, M., Gunderson, J., Perlman, A., &amp; Huang, T. (2006). HMM-based and SVM-based recognition of the speech of talkers with spastic dysarthria. 2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings. https://doi.org/10.1109/icassp.2006.1660840

Tripathi, S., Kumar, A., Ramesh, A., Singh, C., & Yenigalla, P. (2019). Deep learning based emotion recognition system using speech features and transcriptions. arXiv preprint arXiv:1906.05681.

Zhao, G.; Sonsaat, S.; Levis, J.; Chukharev-Hudilainen, E.; Gutierrez-Osuna, R. Accent Conversion Using Phonetic Posteriorgrams. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; IEEE: New York, NY, USA, 2018; pp. 5314–5318.

Zhou, Y.; Tian, X.; Xu, H.; Das, R.K.; Li, H. Cross-lingual Voice Conversion with Bilingual Phonetic Posteriorgram and Average Modeling. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; IEEE: New York, NY, USA, 2019; pp. 6790–6794.