Keyword Spotting on Edge Devices using Neural Networks

Submitted in partial fulfillment of the requirements
of the degree of

B.Tech

by

Meeta Malviya
(Roll No. 18007004)

Supervisors:
Prof. Rajbabu Velmurugan

Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
2021

Contents

List of Tables

Chapter 1

Keyword Spotting on Edge Devices

1.1 Introduction

Keyword spotting (KWS) is a critical component for enabling speech based user interactions on smart devices. It requires real-time response and high accuracy for good user experience. Because of higher accuracy, neural networks have started to become an attractive choice for speech detection. Traditional speech recognition technologies for KWS use Hidden Markov Models (HMMs) and Viterbi decoding, however, they are hard to train and are computationally intensive. Other techniques explored for KWS include deep neural networks (DNN) based models with fully connected layers and rectified linear unit (ReLU) activation functions. These outperform HMM models. The whole setup works on tiny microprocessors and hence have challenges such as limited memory and compute capability. Solving the problem at the edge provides us advantages such as increase in privacy, reduction in latency and network overhead. We have made attempts to use the combined advantages to Deep Learning and Edge devices to classiy the speech commands.

1.2 Motivation

Edge processing with Deep Learning networks refers to data prepossessing done closer to the data source. The current setup uses a centralized facility that collects and processes the data that was captured via devices at data source. The facility uses servers for capturing, detection and transmission of the processed data. The network over head encountered in the process is increased, specially when the data to be processed is huge. Some examples can be home automation systems and video surveillance system. In some cases, such as obstacle detection in a self-driving car if we rely on servers for identification network overhead and latency issues becomes very critical. These applications demand implementations that can be executed on edge devices to leverage the full potential of Artificial Intelligence.

Advantages of using Deep Learning on edge are, lower power consumption, lower data transferred to cloud is decreased and hence does the network overhead. Real time performance improvement provides lower latency. Because the data is not sent to central server, the data remains private. Because of its decentralized nature the system remains robust from cyber attacks and network failure.

KeyWord Spotting (KWS) refers to identifying a set of set of predefined commands (usually a small set) from a stream of input audio, similar to 'Alexa' and 'OK Google'. The power budget is low because the devise has to capture and process spoken words at every instant. Yet these devices have to provide the best accuracy and real time performance for good user experience. These requirements make microcontroller-powered edge devices a good choice for deploying KWS systems.
The major challenges of implementing DL on microcontrollers are :-

1. Limited computational resources
2. Limited memory footprint
3. Limited execution time required- lower latency
4. Energy Constraints - Lower power consumption requirement

Chapter 2

Literature Review

2.1 Related Work

There is an abundance of literature and techniques for keyword spotting. In this line of work, often dataset and complexity of the problem go hand in hand. The earlier attempts for speech recognition technologies used Hidden Markov Models (HMMs) and Viterbi decoding. These methods acquire some resonable ac curacies but on the other hand they are computationally expensive during inference and hard to train.
RNN(recurrent neural networks) outperform HMM models but they have large detection latency. DNN (Deep neural networks) outperforms RNN in terms of accuracy and detection latency in addition to lower size of the model (as it provides huge compression without much reduction in accuracy), but ignores local temporal and spectral correlation in the input speech features. KWS models using deep neural networks (DNN). To overcome these disadvantages CNN (convolutional neural networks) are explored. These models provide higher accuracies and takes the correlation in account. One of drawbacks of CNN is that in the case of modeling time varying signals (e.g. speech) they ignore long term temporal dependencies.

2.2 Feature Extraction

Input speech signal for a given length L is framed into overlapping frames of length l with a stride s, giving a total of $T = \frac{L-l}{s} + 1$ frames. From each frame, F speech features are extracted, generating a total of $T \times F$ features for the entire input speech signal of length L. Mel-frequency cepstral coefficients (MFCC) are computed outputting 13 coefficients. Dimensionality compression of the input signal is achieved as MFCC transforms time domain signal to frequency-domain spectral coefficients. The extracted speech feature matrix is fed into a classifier module, which classifies the output as one of it's classes.

2.3 Deep Learning based KWS models

2.3.1 Convolutional Recurrent Neural Networks

The research in this field progressed from using HMM models to DNN models and now to CNN models. These models not only have the advantages of the about two but also overcomes their disadvantage ie, ignorance local temporal and spectral correlation in the input speech features. The major disadvantage of using CNN is that it ignores long tern temporal dependencies. Combining the strengths of CNNs and RNNs (Recurrent Neural Networks), CRNN(Convolutional Recurrent NN) models are being explored Similar to DNNs, a potential limitation of RNNs is that the modeling is done on the input features, without learning the structure between successive time and frequency steps.

2.3.2 Depthwise Separable CNN (DS-CNN)

Recent efforts are being made in the process to find alternative to efficient usual 3D convolution. These are being used widely because they have lesser number of parameters to adjust as compared to the standard CNN's, which reduces over-fitting and they are computationally cheaper because of fewer computations which makes them suitable for mobile vision applications. In depth wise operation, convolution is applied to a single channel at a time unlike standard CNN's in which it is done for all the M channels. Each input feature map with a separable 2D filter and uses point wise convolutions to combine the output. It breaks the 3D convolution into 2D and 1D convolutions resulting in efficient computation.

2.3.3 Deep Residual Network (ResNet)

Deeper models like ResNet have shown better performance. The problem with these networks can be that the derivative or slope will get smaller and smaller as we go backward with every layer during backpropagation. The time taken can be too long, or NN can stop completely. This is vanishing gradient problem, this network uses residual block to that introduces skip connections in the network. These blocks skips the training from few layers and connects them directly to the output. The advantage of adding this type of skip connection is because if any layer hurt the performance of architecture then it will be skipped by regularization. Further optimizations in the models are done to the model to reach the *'narrow'* footprint version.

2.3.4 Comparison between number of parameters in models

The below table shows model and parameters comparison-

Model	**Parameters**
SampleCNN	11,627,028
Conv trad	623,500
res15	237,882
res-15-narrow	42,648
res8	110,307
res-8-narrow	19,905

Table 2.1: Size comparison between models and parameters

2.4 Comparison between 8051 and Arm Cortex M7 processor

Property	**8051**	**STM32F769**
CPU bit	8	32
Flash Memory	8Kb	2MB
Timers	16 bit	32 bit
Internal Clock Cycle	12MHz	26 MHz
External Clock Cycle	25MHz	216 MHz
RAM	256Bytes	512KB

Table 2.2: Comparison between 8051 and Arm Cortex M7 processor

All microprocessors can be used for Neural Network processing. Comparison is being made in the above table for different properties of 8051 microprocessor and STM32F769 Cortex M7 Arm processor. If a neural network (of appropriate bit size, and memory footprint) is feed into the 8051, we can expect an output which gives the classification of given input. We faced challenges in trying to fit a model in 2MB ROM size of the discovery board, 8051 provides bigger challenge to us by providing 250 times less memory for network parameters. One of the aim of the project was to provide a fast real time classification for good user experience, the clock oscillator is twice as fast in STM32F769 than 8081 in case of internal clock and about 9 times faster in case of external clock. The RAM is also 2k times more in discovery board. These properties of discovery board makes it easier for us to use neural network in them than in 8051 microcontrollers.

Chapter 3

Methodology

3.1 Neural Networks in micro-controller

This project primarily focus on trying to execute the models on microprocessor and do comparative study between different models performances on board. Major task was to maximize the accuracy with small memory footprint model. Due to limited resources there was a constraint on number of operation that can be done per inference and number of parameters in the model. Hence, we have used parameter efficient algorithms and done compression and quantisation of existing NN.

The models were trained in tensorflow and then converted to tensorflow lite and then in tensorflow micro. We have used edge impulse website to find train the model and convert it into deplyable c++ library.

Size comparison of all is shown below:-

Frame work	**size**
Tensorflow	100 – 300 MB
Tensorflow Lite	1 – 2 MB
Tensorflow Lite for Microcontrollers/ Edge Impulse	$<$ 300 kB

Table 3.1: Comparison of library size and frameworks

3.1.1 CMSIS-NN library

The CMSIS-NN library is a collection of efficient neural network kernels that maximizes the performance and minimizes the memory footprint of neural networks on Cortex-M processor cores. It supports 8b/16b/32b fixed point computations for various NN layers.

3.1.2 TensorFlow Lite

We generate Tensorflow model and train with the given dataset in python, later using quantisation methods we convert it into tensorflow lite and later into tensorflow micro. Tensorflow lite is a machine learning framework designed to run on micro controllers. It is written in C++11. It can be used for wide variety of ARM Cortex-M Series architecture. Major limitations to the library is that it supports limited subset of TensorFlow operations and on device training is not possible.

3.1.3 Edge Impulse

It is an online program that helps in the end-to-end development and deployment of neural networks to micro-controllers. The platform can be used to generate C++ code with source libraries that can be used in micro controllers It supports 8b quantization of the NN mode.

3.2 Data Set

The dataset that we are using is *'Google speech commands dataset'*. The dataset consists of 1,05,829 utterances of 35 words namely “Backward”, “Bed”, “Bird”, “Cat”, “Dog”, “Down”, “Eight”, “Five”, “Follow”, “Forward”, “Four”, “Go”, “Happy”, “House”, “Learn”, “Left”, “Marvin”, “Nine”, “No”, “Off”, “On”, “One”, “Right”, “Seven”, “Sheila”, “Six”, “Stop”, “Three”, “Tree”, “Two”, “Up”, “Visual”, “Wow”, “Yes”, “Zero”.
We have trained our model on 7h 44m 51s worth data for 10 output classes namely “Yes”, “No”, “Up”, “Down”, “Left”, “Right”, “On”, “Off”, “Stop”, and “Go”. Each wav file was of 1s in length and the sample data is encoded as linear 16b single channel PCM values at 16k Hz.

3.3 Pre-Processing

3.3.1 Mel-Frequency Cepstrum Coefficients MFCC

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.
Mel-frequency cepstral coefficients (MFCCs) are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The frequency bands are equally spaced on the mel scale(hence are non linearly spaced in frequency space, lower frequencies filters have smaller bandwidth than higher frequencies), which approximates the human auditory system’s response more closely than the linearly-spaced frequency bands used in the normal spectrum. To calculate MFCC-

1. Take Fourier transform of a signal

2. take triangular overlapping windows(having increasing bandwidth in subsequent filters)

3. take logs of the powers at each of the mel frequencies

4. take discrete cosine transform of the mel frequencies

5. MFCCs are amplitude of resulting spectrum

Chapter 4

Implementation and Results

4.1 Reduction in sampling rate

One of the factors that favored the choice of the ResNet based model is the better tunability of the model by controlling the depth (number of network layers) as well as width (number of features per layer) of the model, without largely impacting its memory footprint.
The models namely DSCNN and res-8, were trained for 8k sampling frequency and 16k sampling frequency. A comparative study can be done from the below tables. We observe that as we quantize the models from tensor flow to tensorflow lite and then to Tensor flow lite quantized huge size reduction occurs without hampering the accuracy of the model. Reduction in sampling rate ie. down sampling of the dataset reduced the feature extractions time considerably without compromising on accuracy. The below shows the summary of the results,

DS-CNN	**8k SampRate**		**16k SampRate**	
Frame work	Size(bytes)	Accuracy	Size(bytes)	Accuracy
Tensor Flow	245493	90.7651	245493	86.9474
Tensor flow lite	94848	90.7651	94848	86.9474
Tensor flow lite quantized	44984	76.2909	45616	71.3729

Table 4.1: accuracy and size comparison between 8k and 16k sampling rate in DSCNN model

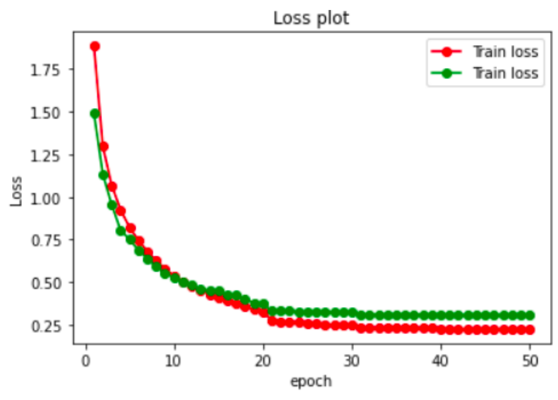

Figure 4.1: Training and validation loss at 8k samp rate for dscnn

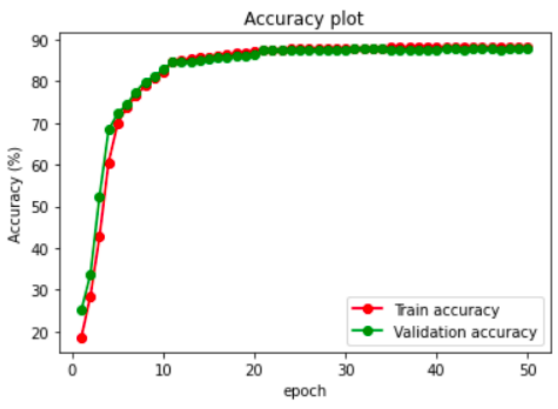

Figure 4.2: Training and validation accuracy at 8k samp rate for dscnn

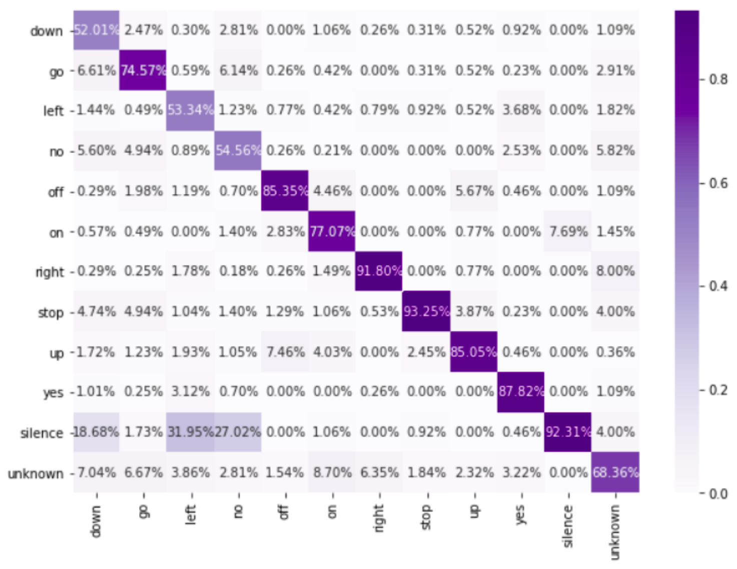

Figure 4.3: Confusion matrix for TF lite model with 8b quantization at 8k samp rate for dscnn

res-8	**8k SampRate**		**16k SampRate**	
Frame work	Size(bytes)	Accuracy	Size(bytes)	Accuracy
Tensor Flow	198515	90.2921	208305	90.6954
Tensor flow lite	92276	90.2921	92588	90.6954
Tensor flow lite quantized	30808	85.9278	31016	81.7892 9

Table 4.2: accuracy and size comparison between 8k and 16k sampling rate in res-8 model

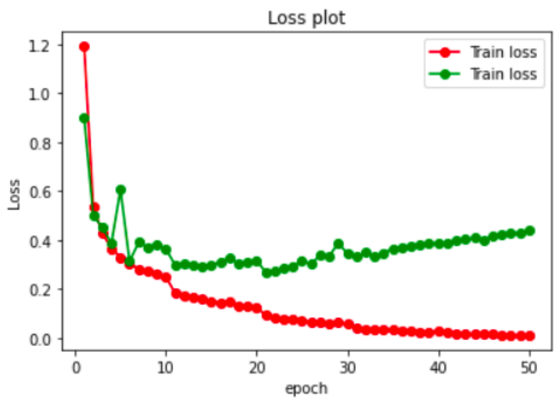

Figure 4.4: Training and validation loss at 8k samp rate for res8

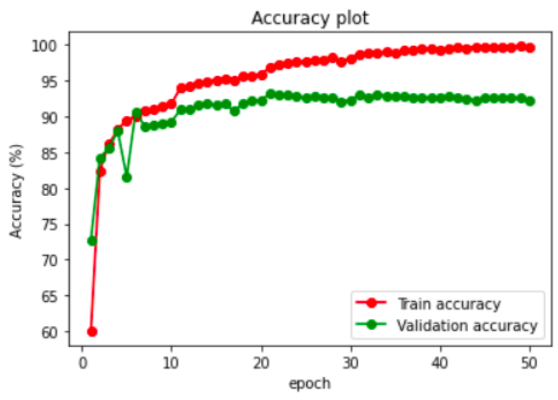

Figure 4.5: Training and validation accuracy at 8k samp rate for res8

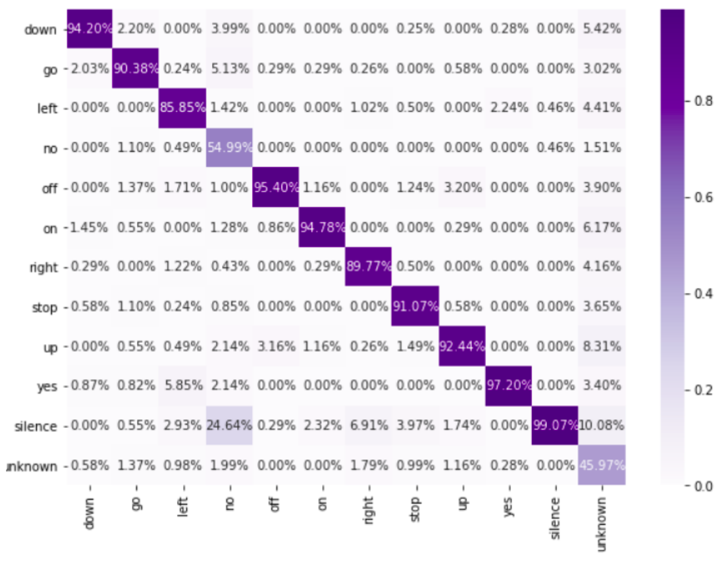

Figure 4.6: Confusion matrix for TF lite model with 8b quantization at 8k samp rate for res8

4.2 Changes in MFCC computations

The platform of Edge Impluse was utilized for training the neural networks. It is an online program that helps in the end-to-end development and deployment of neural networks to micro-controllers. The platform can be used to generate C++ code with source libraries that can be used in micro controllers It supports 8b quantization of the NN mode.

Parameter	Value
Sampling Rate	16k Hz
Frame Length	30ms
Frame Stride	10ms
FFT length	256
Ram Usage(int8)*	35.8K
Ram Usage(float32)*	109.3K
Latency(int8)*	94ms
Latency(float32)*	439ms

Parameter	Value
Flash Usage(int8)*	58.8K
Flash Usage(float32)*	115.1K
Training Accuracy	80.4%
Accuracy(int8)	73.69%
Accuracy(float32)	77.36%
Peak Ram Usage*	22.9KB
Processing Time*	224ms
MFCC	13

Table 4.3: Edge Impluse training for reduced window size

* refers to on device performance predictions made via edgeimpluse platform

Parameter	Value
Sampling Rate	16k Hz
Frame Length	40ms
Frame Stride	20ms
FFT length	256
Ram Usage(int8)*	22.9K
Ram Usage(float32)*	57.5K
Latency(int8)*	49ms
Latency(float32)*	228ms

Parameter	Value
Flash Usage(int8)*	58.7K
Flash Usage(float32)*	115.0K
Training Accuracy	82.1%
Accuracy(int8)	75.45%
Accuracy(float32)	75.52%
Peak Ram Usage*	22.9KB
Processing Time*	119ms
MFCC	13

In the previous models we have used a 40 ms window length, stride length of 20ms and FFT size of 512. 40 ms (window size) is a bit on the high side for speech signals (it is suitable for other kinds of audio including music). We have tried a lower window size, namely of 30ms length. The hop is ideally 10 ms but can be increased to 20ms, in the interest of achieving computation efficiency.
After training the model for the changes in MFCC, we can see that, in case of frame length of 30ms and stride length of 10ms, as number of computations to be done are more because of more windows to be processed, the RAM usage, processing time and Latency is more. And the accuracy achieved also does not have major difference than that in the case of 40ms window and 20ms shift. Hence, it is better to use bigger window for classification.

4.3 Deployment on Board

STM32F769 Discovery Board is being used for the purpose of measuring performance of the model at edge. We have used the output of Edge Impulse website as a C++ library for classification of the input.
The code for giving input(time series) into the neural network via code memory works perfectly on the discovery board. I have made attempts to combine the code for collecting the data from the microphone and using that as input for the classification task, unfortunately, the code has errors and some work is required.

Chapter 5

Conclusion and future scope

An end-to-end deep learning-based keyword spotting system was implemented on the STM32F769I-DISC1 development board. We have explored various models that can be used for the purpose of KWS. We observed that as we go on making complex models the memory footprint increases. There is a trade off between accuracy and memory footprint. We also observed that Neural Networks outperform traditional methods of keyword spotting algorithms
Future scope of this project includes:-

1. The work can be extended and implemented for quantization of weights from floating point numbers to smaller 8, 4 or even 2 bit integers to make them smaller. The challenge is to maintain their accuracy while reducing their size by great amount. These will let us train more deeper models with more classes in the traning set.

2. The work can be extended to recognizing keywords in a sentence and work out more meaningful commands. We can train models for specific command such as "play 'abc' artist song."

3. The response to the live audio stream can be improved.

4. The model can be made robust to variations in speech signals due to background noises.

References

[1] Y. Zhang, N. Suda, L. Lai, and V. Chandra, “Hello edge: Keyword spotting on microcontrollers,” *arXiv preprint arXiv:1711.07128, 2017*

[2] T. N. Sainath and C. Parada, “Convolutional neural networks for small-footprint keyword spotting,” in Sixteenth Annual Conference of the International Speech Communication Association, 2015 P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” *arXiv preprint arXiv:1804.03209, 2018.*

[3] R. Tang and J. Lin, “Deep residual learning for small-footprint keyword spotting,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 5484–5488.*

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

[5] Alaa Saade, Alice Coucke, Alexandre Caulier, Joseph Dureau, Adrien Ball, Theodore Bluche, ´ David Leroy, Clement Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Mael Primet "Spoken Language Understanding on the Edge" IEEE,2019

[6] 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.

[7] S. O. Arik, M. Kliegl, R. Child, J. Hestness, A. Gibiansky, C. Fougner, R. Prenger, and A. Coates, “Convolutional recurrent neural networks for smallfootprint keyword spotting,” arXiv preprint arXiv:1703.05390, 2017.

[8] Edge Impulse, `https://docs.edgeimpulse.com/docs`

[9] TensorFlow Lite for Microcontrollers, `https://www.tensorflow.org/lite/microcontrollers`

[10] T. Kim, J. Lee, and J. Nam, “Comparison and Analysis of SampleCNN Architectures for Audio Classification,” IEEE Journal of Selected Topics in Signal Processing, vol. 13, no. 2, pp. 285–297, 2019.

[11] T. N. Sainath and C. Parada, “Convolutional neural networks for small-footprint keyword spotting,” in Sixteenth Annual Conference of the International Speech Communication Association, 2015.

Acknowledgments

I want to extend my sincere thanks to Professor Rajbabu Velmurugan for giving me the opportunity and mentoring me and helping me to pursue this research topic. I also want to thank him for his constant support, encouragement and inspiration throughout the project work. Without his invaluable guidance, this work would never have been a successful one. I want to thank Ms. Shyama P, my student mentor, who has helped me understand challenging topics.

Meeta Malviya
Indian Institute of Technology Bombay
January 21, 2022