

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION

Federal State Autonomous Educational Institution of Higher Education

“South Ural State University (National Research University)”

School of Electrical Engineering and Computer Science

Department of Computer Science

THESIS IS CHECKED

Reviewer

PhD, Associate Professor

_____ N.V. Plotnikova

“ ___ ” _____ 2021

ACCEPTED FOR THE DEFENSE

Head of the department,

PhD, Associate Professor

_____ G.I. Radchenko

“ ___ ” _____ 2021

**DEVELOPMENT OF A SPEECH RECOGNITION LIBRARY FOR
ULTRA-LOW POWER EMBEDDED DEVICES**

GRADUATE QUALIFICATION WORK

SUSU – 09.04.01.2021.308-643.GQW

Supervisor,

PhD, Associate Professor

_____ D.V. Topolsky

Author,

student of the group: KE228

_____ F.K. Chemorion

Normative control

_____ S.V. Syaskov

“ ___ ” _____ 2021 г.

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
Federal State Autonomous Educational Institution of High Education
“South Ural State University (National Research University)”
School of Electrical Engineering and Computer Science
Department of Computer Science

APPROVED
Head of the department,
PhD, Associate Professor
G.I. Radchenko
“ ” 2021

TASK

of the master graduate qualification work

for the student of the group KE-228

Francis Kiptengwer Chemorion
in master direction 09.04.01

“Fundamental Informatics and Information Technologies” (master program
“Internet of Things”)

1. The topic (approved by the order of the rector from 25.05.2021)
Development of a Speech Recognition Library for Ultra-Low Power
Embedded Devices.

2. The deadline for the completion of the work: 05.06.2021.

3. The source data for the work: Speech commands dataset. [Electronic
Resource] URL: <https://arxiv.org/abs/1804.03209>

4. The list of the development issues

4.1. To develop a Library that allows real-time classification of raw audio.

4.2. To develop a library that runs on devices that consume low power and
have low processing capabilities.

4.3. To test the library and give an example of how to implement interfaces.

4.4. To deploy the library in a version control environment.

4.5. To deploy the library in at least one official Library managers for MCU
hardware vendors.

5. Issuance date of the task: 08.02.2021.

Supervisor

PhD, Associate Professor
D. V. Topolsky

The task is taken to perform

Student
F. K Chemorion

Student: Francis Kiptengwer Chemorion

Supervisor: Dmitry Valerianovich Topolsky

Topic: Development of a Speech Recognition Library for Ultra-Low Power Embedded Devices

**The calendar plan
of the execution of master graduate qualifying work (GQW)**

| No | Phase | Duration | Deadline | Report | Actual date of execution | Supervisor's signature |
|----|--|----------|---|--|---------------------------|------------------------|
| 1. | Introduction and literature review | 1 month | February, 25 th | 1. Task of the master graduate qualification work 2. Text of Introduction 3. References | February 25 th | |
| 2. | Development of the model, design of the system | 1 month | March, 15 th | 1. Text of chapter 1 (theoretical part). | March 15 th | |
| 3. | Implementation of a system | 1 month | April, 15 th | 1. Software system 2. Text of chapter 2 (implementation part). | April 15 th | |
| 4. | Testing and debugging of the system, experiments | 2 weeks | May, 1 st | 1. Set of tests 2. Text of chapter 3 (experimental part). | May 1 st | |
| 5. | Full text | 2 weeks | May, 15 th | 1. Full text of GQW | May 15 th | |
| 6. | Validation of the text by supervisor | 1 week | May, 22 nd | 1. Electronic version of the GQW text checked by the supervisor | May 22 nd | |
| 7. | Normative control | 3 days | May, 25 th | 1. Twisted text of GQW signed by student, supervisor and normative controller | May 25 th | |
| 8. | Proposal defense | 1 week | May, 25 th – June, 1 st | 1. Twisted text of GQW signed by student, supervisor and normative controller for the signature of the Head of the Department Head about accepting for the defense 2. A signed review of the supervisor 3. A review of the reviewer, signed and notarized at his place of work 4. Implementation act (if exists) 5. Presentation of the report in PowerPoint | June 17 th | |

Table of Contents

| | |
|--|----|
| Introduction | 6 |
| Structure of the Thesis | 8 |
| Subject Area Analysis | 8 |
| 1.1 Overview of analogues | 9 |
| 1.1.1. Kaldi NL | 9 |
| 1.1.2. Speechmatics..... | 10 |
| 1.1.3. Google Speech API..... | 11 |
| 1.2 Analysis of the main technological solutions | 14 |
| 1.2.1 Speech Commands Dataset..... | 14 |
| 1.2.2 TensorFlow | 14 |
| 1.2.3 GitHub..... | 15 |
| 1.2.4 Google Colaboratory..... | 17 |
| 1.2.5 Programming Technologies | 17 |
| 1.3 Conclusion | 18 |
| 2 Definition of Requirements..... | 19 |
| 2. 1 Functional Requirements | 19 |
| 2.1.1 Speech Recognition | 19 |
| 2.1.2 Continuous Recognition Style | 19 |
| 2.1.3 Internet independence | 19 |
| 2.1.4 Noise | 19 |
| 2.1.5 Processor power | 19 |
| 2.2 Core Requirements..... | 19 |
| 2.3 Speech Requirements..... | 20 |

| | |
|---|--|
| 2.4 Documentation Requirements..... | 20 |
| 2.5 Conclusion | 21 |
| 3 Design and Implementation | 22 |
| 3.1 Architecture of the Proposed Solution..... | 22 |
| 3.2 Description of Data..... | 23 |
| 3.2.1 Data Collection | 23 |
| 3.2.2 Data Usage | 23 |
| 3.2.3 Data Storage..... | 23 |
| 3.2.4 Data Retention | 24 |
| 3.3 Algorithms for solving the problem..... | 25 |
| 3.4 Conclusion | 28 |
| 4 Deployment and Testing | 29 |
| 4.1 Deployment..... | 29 |
| 4.2 Implementation of interfaces | 30 |
| 4.2.1 Arduino Example sketch compiling and running..... | 30 |
| 4.3 Testing the Machine Learning model | 34 |
| 4.4 Testing the library using Arduino Lint | 35 |
| 4.5 Conclusion | 36 |
| 5 Conclusion | 37 |
| 5.1 Optimizing Latency | 38 |
| 5.2 Optimizing Power Usage | 38 |
| 5.3 Optimizing Model and Binary Size | 38 |
| References | Ошибка! Закладка не определена. |

Acknowledgements

I would like to acknowledge the role that God has played in my progress. For giving me this opportunity to further my studies a long way from home. I would also like to thank my family for always being there for me this whole time.

My supervisor Dmitry Topolsky has been very instrumental and resourceful, my department has always been open and available to solving all my educational and non-educational problems. My university has given me a conducive learning environment that I cannot take for granted.

Lastly, I would lastly like to thank the Russian Government for diplomatically facilitating my stay in Russia.

Introduction

Speech recognition, also known as automated speech recognition, machine speech recognition, or speech-to-text, is a feature that allows a program to convert human speech into written text. Although voice recognition and speech recognition are often mixed, speech recognition focuses on the conversion of speech from a spoken format to a text format, whereas voice recognition only attempts to recognize a particular user's voice [1].

As embedded device computing becomes more prevalent, it presents new and creative ways for humans and machines to cooperate, connect, and interact. Though data has received most of the attention in embedded systems, the incorporation of speech into embedded computer applications provides a flexible means of providing human interaction, connectivity, and control. Voice, when implemented properly and with respect for relevance, prices, and human factors, may offer a more versatile user interface at a lower cost than conventional approaches such as touch screens or data entry [2].

There are several advantages to speech-machine interaction:

- Speech as a form of communication, is more intuitive and simpler. Speech recognition is particularly useful when the human's hands or eyes are otherwise occupied. For example, it may very be convenient to use verbal commands when driving or when multitasking.
- Speech telephony is a reliable method of bi-directional speech communication with devices that can listen and respond without the use of complicated commands.
- Speech integration can reduce or eliminate the need for a touch screen on many devices, lowering the cost of the device especially when interacting with apps that are not commonly used and do not need a touchscreen.

- For people with disabilities that limit them from entering data manually, speech enablement is a vital method of interacting with various devices. Speech communication with computers enables the remote control of various functions that would otherwise be manual [3].

Several firms have also made available various speech recognition algorithms, databases, and templates. These services enable developers to add products like a digital personal assistant or even speech-controlled home automation systems. These products are common because they simplify and automate activities that can be time-consuming and repetitive. Consider an apartment where every part of the house is linked to a smart assistant device. It will be as easy as instructing the device to activate night mode, switch off all lights, lock all doors and windows, and activate the security alarm.

The purpose of the research:

To create a software complex to enable running of machine learning algorithms for speech recognition on small devices with limited power and computing capabilities Tasks necessary to achieve the goal:

- Analyzing the market for existing libraries.
- Collect a keyword dataset to use to train a machine learning model for speech recognition.
- Design a model architecture for MBEDSpeech.
- Training the MBEDSpeech Model
- Testing and deploying the model to different embedded device Platforms in a way that can be used by other developers.
- Integrating MCU code with deployed library as an interfacing example
- Testing the integrated model

Structure of the Thesis

This thesis consists of five main chapters: Introduction, definition of requirements, design and Implementation, deployment and testing, conclusion, and references.

In the first chapter, we will have the subject area analysis briefly then have an overview of analogues and the main technological solutions that I will use will be featured. All the different software platforms to be used will be adequately described.

In the second chapter, there is a description of both functional and non-functional requirements as well as a description of how the various software components will interact with each other. We then finish by describing how documentation will be done for the device.

In the third chapter we describe the design and implementation of the software and how the different components will interact with each other as well as the algorithms for tackling the problem and a description of the data.

In the fourth chapter we will do deployment and testing for both the software accuracy and the hardware performance and interfacing with a real MCU. For this practice we will use an Arduino MCU.

Finally in the fifth chapter we will have a conclusion for the thesis, with future improvements to the solution being discussed as well as opportunities.

Subject Area Analysis

An ultra-low-power embedded device is one that is inexpensive, runs on a few hundred kilobytes of RAM, has similar amounts of flash memory for persistent program and data storage, has a clock speed of just tens of megahertz, does not run a full operating system, and avoids using dynamic memory allocation functions like `new` or `malloc()` because they're designed to be reliable and long-running, and it's extremely difficult to ensure that if you have a heap that can be fragmented [4].

To take advantage of ultra-low-power embedded devices' low power consumption, it is imperative that we merge the embedded software development with machine learning since data is the most important by-product of any system. One of the most anticipated problems that this combination of disciplines can solve through predicting things based on past observations is speech recognition [4].

To do speech recognition using deep learning, a programmer should feed data into learning algorithm that discovers the rules in the data, which then builds a model based on the data provided through a process called training and finally data is then run through this model to make predictions, a process called inference [4].

The past few years has seen products provide a voice user interface (UI) designed to give instant access to information without the need for a screen or keyboard. All these applications use speech recognition libraries in their development as will be discussed below. In most cases, speech recognition is done in the cloud, on powerful servers running large ML models bringing up privacy issues, efficiency, and speed due to latency and high-power consumptions sending a constant stream of data consumes a lot of energy [4].

This thesis will therefore focus on training a tiny model that listens for a wake words specified in the Speech Commands dataset and run it on a low-powered chip and does offline inference as well as give access to other embedded software developers to use it in various applications.

1.1 Overview of analogues

1.1.1. Kaldi NL

Kaldi is a speech recognition toolkit written in C++ intended for use by researchers whose main goal is to have modern and flexible code, written in C++, that is easy to modify and extend. Important features include:

- It can be used at code-level for integration as a library.
- It has extensive support for linear algebra.

- Kaldi provides algorithms in a way that can easily be extended in the most generic form possible.
- Code for Kaldi is provided using an unrestrictive license: Apache 2.0 permitting and encouraging modifications and re-release.
- Kaldi's documentation is complete and highly accessible to everyone [5].

1.1.2. Speechmatics

Speechmatics is speech to text recognition software powered by machine learning with high accuracy. The software is available on both cloud and on-premise for users; it can also be embedded in devices. It uses a custom language build substructure i.e., automatic linguist, which lets the software learn new languages at a high pace [6].

Speechmatics boasts a high accuracy level that is further boosted by the custom dictionary feature. You can add new words to a language quickly. Speechmatic takes a new approach to recognize the English language, and it provides accurate speech recognition despite the user's accent. Features of Speechmatics include:

- It gives a very accurate speech to text conversion.
- It constantly delivers a low word error rate across all the languages it is supporting. Speechmatic provides frequent testing of the languages they are offering to keep a check on word error rate.
- It is an industry major in language coverage. Speechmatic is getting updated with new vocabulary to meet business-relevant needs.
- It can be installed with cloud services to avail its speech-to-text technology in real-time or, one can avail a pre-recorded (batch) files for on-premise installation. Speechmatics provides adjustable deployment.
- It allows its users to add context-specific words to its dictionary. It enhances your transcription accuracy. This feature enables you to define the context

of a conversation in advance. You can input variables like name, accents, abbreviations, acronyms, special, or industry-specific language, et cetera.

- It gives advanced punctuation, that is built over 2.5 billion words and holds an industry-leading set of supported punctuation marks. This optimizes the pace and ease of reading a transcript for human users.
- The software identifies a change of speaker within the user's transcript. It adds a token automatically when the change in the speaker is noticed. This helps in easing the transcript modification for readability [7].

1.1.3. Google Speech API

Google Cloud Speech API is a programmatic interface to Google Cloud Platform services that allows a user to send speech recognition requests to Speech-to-Text in any programming language using the Google Cloud Client Libraries. It is a key part of Google Cloud Platform, allowing you to easily add the power of speech to everything from computing to networking to storage to machine-learning-based data analysis to your applications [8].

Speech-to-Text can process up to 1 minute of speech audio data sent in a synchronous request. After Speech-to-Text processes and recognizes all of the audio, it returns a response. A synchronous request is blocking, meaning that Speech-to-Text must return a response before processing the next request.

Speech-to-Text has three main methods to perform speech recognition. These are listed below:

- Synchronous Recognition which sends audio data to the Speech-to-Text API, performs recognition on that data, and returns results after all audio has been processed.
- Asynchronous Recognition which sends audio data to the Speech-to-Text API and initiates a Long Running Operation which can periodically be polled for recognition results.

- Streaming Recognition which performs recognition on audio data provided within a gRPC bi-directional stream. Streaming requests are designed for real-time recognition purposes, such as capturing live audio from a microphone. Streaming recognition provides interim results while audio is being captured, allowing result to appear, for example, while a user is still speaking [9].

After comparing the different transcribers Google Cloud Speech-to-Text, Speechmatics and Kaldi on Word Error Rate and hourly cost to see how they compare to each other, I found out what are their weaknesses are and it helped me to build up on my proposed solution.

First, when on comparing Google Cloud Speech-to-Text and Speechmatics, both these services are quite similar in their offering: cloud-based speech-to-text for many different languages with high performance. Both Google and Speechmatics continually update their language models to increase accuracy and introduce new words where applicable. This continuous development is a strong point for both, especially the introduction of new words which can help with new company names and other terms. Both services have a cost per minute of audio transcribed: Google uses a fixed price, the price for Speechmatics goes down as more minutes are purchased up front. Google has a few different transcription models available more than Speechmatics, however [21].

On the other hand, Kaldi which is an opensource speech recognition toolkit developed and maintained mainly by Daniel Povey with the help of about 70 other contributors so far has a lot of flexibility, especially since it's open source and can be extended or improved by anyone who dares understand it. It does carry a lot of complexity however, requiring a lot of time and effort to fully learn the quirks.

Using Word Error Rate, which compares a reference with a hypothesis and is given by $(\text{Substitutions} + \text{Insertions} + \text{Deletions}) / \text{Number of words}$, where a substitution is when a word is replaced, an insertion is when a word is added that

wasn't said and a deletion is a word that is omitted, Google Speech-to-text scored 4.9% [10], Speechmatics 14.7% [11] and Kaldi 8.01% [12].

I also compared them in terms of the cost of running for instance on an Amazon EC2 instance by dividing costs of the EC2 instance by the amount of audio it can transcribe per hour, for a c5.2xlarge instance it costs \$0.384 per hour. There is an additional monthly fee for storage usage as well, however this does not significantly increase hourly costs.

With the Amazon instance running I also did some testing to determine how much audio could be transcribed per hour and determined that we could transcribe at a rate of 5 to 10% of the duration of the audio. I determined that it cost about \$0.038 per hour of transcribed audio for Kaldi NL but \$ 2.4 for Speechmatics and \$ 1.4 for the Google Speech API. However, this does not include the cost of setup and maintenance!

| Feature | Kaldi NL | Speechmatics | Google Speech API |
|---------------------|--|----------------------------|------------------------------------|
| Language support | 60 Languages | 21 languages | 80 languages |
| Cost / min | \$ 0.038 | \$ 2.4 | \$ 1.4 |
| Speaker detection | English (8Khz) | No | No |
| Audio Formats | FLAC, Siren, WAV, OGG, NULAW, Siren SR | FLAC, PCM, WAV, OGG, NULAW | FLAC, Linear16, MULAW, ARM, AMR_W8 |
| Noise Friendly | Yes | No | Yes |
| Word Hints | Yes | No | No |
| Internet dependency | Yes | Yes | Yes |

Table 1 Comparison of analogues

1.2 Analysis of the main technological solutions

1.2.1 Speech Commands Dataset

The Speech Commands Dataset was created by the TensorFlow and AIY teams to showcase the speech recognition example using the TensorFlow API. The dataset has 65,000 clips of one-second-long duration. Each clip contains one of the 30 different words spoken by thousands of different subjects [13]. This dataset will be combined with the Microsoft scalable Noisy Speech Dataset, which has a collection a variety of environmental noise files in .wav format sampled at 16khz. For the unknown category, I will use audio files with a collection of other words that are not considered in the dataset classes.

It has limited vocabulary but is still have enough variety for models trained on the data to potentially be useful for some applications. The dataset's top ten common words as the core of our vocabulary that would be useful as commands in embedded systems or robotics applications; "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", and "Go" [14]. I will be focusing my thesis on these.

In the second version of the dataset, there are four more command words; "Backward", "Forward", "Follow", and "Learn". One of the most challenging problems for keyword recognition is ignoring speech that does not contain triggers, so I also needed a set of words that could act as tests of that ability in the dataset. Some of these, such as "Tree", were picked because they sound like target words and would be good tests of a model's discernment. Others were chosen arbitrarily as short words that covered a lot of different phonemes. The final list was "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow". These extra words will be added in a future release of the library.

1.2.2 TensorFlow

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and

inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources [15].

TensorFlow is the best library of all because it is built to be accessible for everyone. TensorFlow library incorporates different API to build at scale deep learning architecture like CNN or RNN. TensorFlow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensor board. This tool is helpful to debug the program. Finally, TensorFlow is built to be deployed at scale. It runs on CPU and GPU. To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword to the search bar, Google provides a recommendation about what could be the next word [16].

Some supported TensorFlow algorithms include:

- Linear regression
- Classification
- Deep learning classification
- Deep learning wipe and deep
- Booster tree regression
- Boosted tree classification [17].

1.2.3 GitHub

GitHub provides combines the distributed version control and source code management features of Git with access control, issue tracking, feature requests, task management, continuous integration, and wikis for every project [18]. These services are free of charge for individuals; however, the more complex professional and corporate services are commercial. It provides limitless private repositories to all plans, including free accounts, but only allows up to three contributors per repository. The following are some of the benefits of utilizing git:

- Git performs very strongly and reliably when compared to other version control systems. New code changes can be easily committed, version branches can be effortlessly compared and merged, and code can also be optimized to perform better. Algorithms used in developing Git take the full advantage of the deep knowledge stored within, with regards to the attributes used to create real source code file trees, how files are modified over time and what kind of file access patterns are used to recall code files as and when needed by developers.
- Git is designed specially to maintain the integrity of source code. File contents as well as the relationship between file and directories, tags, commits, versions etc. are secured cryptographically using an algorithm called SHA1 which protects the code and change history against accidental as well as malicious damage. You can be sure to have an authentic content history for your source code with Git.
- Git offers support several kinds of nonlinear development workflows and its efficiency in handling both small scale and large scale projects as well as protocols. It is uniquely designed to support tagging and branching operations and store each and every activity carried out by the user as an integral part of “change” history. Not all VCSs support this feature.
- Git offers the type of performance, functionality, security, and flexibility that most developers and teams need to develop their projects. When compared to other VCS Git is the most widely accepted system owing to its universally accepted usability and performance standards.
- Git is a widely supported open-source project with over ten years of operational history. People maintaining the project are very well matured and

possess a long-term vision to meet the long-term needs of users by releasing staged upgrades at regular intervals of time to improve functionality as well as usability. Quality of open-source software made available on Git is heavily scrutinized a countless number of times and businesses today depend heavily on Git code quality [19].

1.2.4 Google Colaboratory

Colaboratory, is a product from Google Research that allows anybody to write and execute arbitrary python code through the browser and is well suited to machine learning and data science. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs [20].

I will be using Colab to:

- Write and execute code in Python
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from and to GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Utilize free Cloud service with free GPU

1.2.5 Programming Technologies

For this project we will use Python, C and C++ as our main programming languages for development. Development that tools are required to develop and test/debug the code include:

- Compiler.
- Debugger.

1.3 Conclusion

The following components are being used for this project. It is urgent and under development right now:

1. Google Speech Dataset
2. TensorFlow
3. Keras
4. GitHub
5. Google Colab

2 Definition of Requirements

2.1 Functional Requirements

2.1.1 Speech Recognition

The MBEDSpeech recognition library allows 1 second word length of a 10 word vocabulary. It stores the "trained" word patterns used for recognition in internal memory. The main board has a charging port that can be hooked up to a battery to power the static ram when the main circuit is turned off. This keeps all the trained words safely stored in memory (SRAM) so the circuit does not have to be retrained every time it is turned on.

2.1.2 Continuous Recognition Style

It is the natural conversational speech people are accustomed to in everyday life. It is extremely difficult for a recognizer to shift through the text as the words tend to merge together. Isolated speech recognition system is another feature, of the IC that is used by MBEDSpeech.

2.1.3 Internet independence

The machine learning model used by MBEDSpeech should run on the MCU offline.

2.1.4 Noise

MBEDSpeech should be able to work well in noisy environments by automatically recognizing and filtering out the noise.

2.1.5 Processor power

MBEDSpeech should run on an edge device with a 32-bit ARM Cortex-M4F microprocessor running at 64MHz with 1MB of program memory and 256KB RAM

2.2 Core Requirements

2.2.1. MBEDSpeech shall be able to classify audio in Realtime.

2.2.2 MBEDSpeech shall be compatible with edge Devices from Arduino, STM Electronics and NVIDIA

2.2.3 MBEDSpeech shall be easy to use by developers through bootstrapping of all the required code and shall have comments

2.2.4 MBEDSpeech prototype used for this thesis will be limited to 4 words to assess how effective the approach taken is, then be updated, through GitHub to include all the words in the Google Speech Commands Dataset for completeness

2.2.5 MBEDSpeech shall provide binary files for testing on the devices targeted

2.3 Speech Requirements

The following guidelines are specific to speech:

2.3.1 MBEDSpeech SHALL use only approved keywords words found in the Google Speech Commands Dataset.

2.3.2 MBEDSpeech SHALL provide serial output to indicate when the Microphone Off.

2.3.3 MBEDSpeech shall recognize words in English only for the scope of this thesis

2.3.4 The audio used for MBEDSpeech training shall be 16KHz

2.4 Documentation Requirements

Documentation of this library will be delivered in various formats including pdf, ppt, html and docx and will be English language. This documentation will be the user manual for “MBEDSpeech” outlining its various features and how to use the Library for microcontroller development. It will contain all the options for deploying the service on various MCUs.

Error codes will also be explained and several solutions or how to avoid or solve them will also be outlined.

2.5 Conclusion

In this chapter, we defined the various core and speech requirements for MBEDSpeech. We also described how the various software components within it will interact with each other. Lastly, we gave the documentation requirements that described among other things the user manual and its contents.

3 Design and Implementation

3.1 Architecture of the Proposed Solution

The MBEDSpeech Library is composed of various components, as shown in the image below. First, the microphone receives audio input then using MFCC (Mel frequency cepstral coefficients) extracts features and at the same time reducing the magnitude of the speech signal devoid of causing any damage to the power of speech signal. The inference is then run on the features outputting class probabilities as shown in the diagram below.

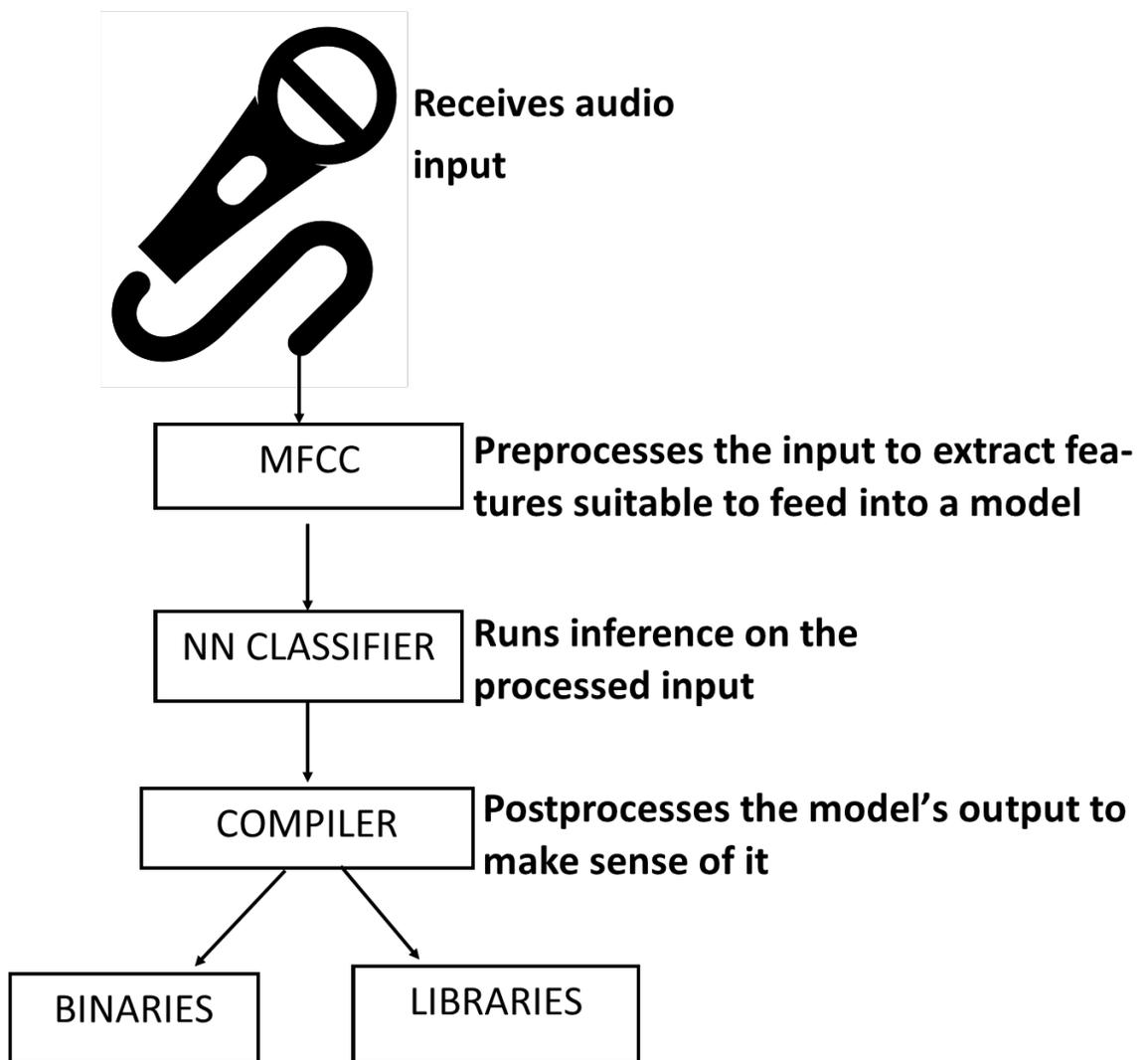


Figure 1 MBEDSpeech Architecture

The model used in this thesis is trained to recognize the words "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", noise and unknown speech, but the list will be updated after a first successful deployment. The model takes in one second worth of data at a time and outputs the probability scores. This data is consumed by the model in terms of spectrograms which are two-dimensional arrays that are made up of slices of frequency information, each taken from a different time window.

3.2 Description of Data

3.2.1 Data Collection

Audio is continuously sampled at 16 KHz using a microphone. The data is then temporarily stored in the device before being passed to the MFCC which converts raw audio data into spectrograms that will be run through the inference. Users' exposure to malicious actions by other users by discarding the data completely.

3.2.2 Data Usage

After the raw data is run through inference, probabilities for the various classes of data is then calculated and displayed on the serial monitor for every second of data. At this point a command recognizer could be programmed to undertake a certain command which the command responder will execute.

3.2.3 Data Storage

Different metadata is generated and stored by the MBEDSpeech system for the purposes of device diagnostics and service improvements. Audio inputs being the core piece of MBEDSpeech data, is however not stored because of the size of devices that MBEDSpeech is targeting. Speech is processed through the neural

network and then to extract the user's intent. These system uses machine learning techniques to continuously improve itself with each input.

3.2.4 Data Retention

Data is not in any way stored through the MBEDSpeech Library, but if the developer of the device decides to store data, they are free to do so. As for the metadata, access is granted via specific, audited permissions and access to customer data requires review and approval by the responsible managers. Additionally, the permissions to access this metadata are reviewed and positively confirmed by management at least quarterly and access is audited.

Some system level data is also stored in log files, for either service troubleshooting purposes, or security incident resolution. Troubleshooting logs contain information necessary for developers to troubleshoot the MBEDSpeech Library, but do not contain customer speech recordings or data derived from customer speech recordings. Access to these logs is restricted to teams needing access to this data to perform their business functions. Troubleshooting logs are encrypted and their access is audited.

We apply retention policies to data to minimize the (meta)data we retain. Data is retained when it serves a business purpose (including providing the service to customers and improving our systems) or as necessary to comply with law. We also offer debug interfaces like SWD and we also have disabled code readout on Arm platforms. Even though these measures are not perfect, they will raise the cost of an attack

The speech recognition and natural language understanding in the MBEDSpeech system are based on machine learning (ML) algorithms. Data sets from real use cases are fed into the various ML systems to build new algorithms and improve existing algorithms.

3.3 Algorithms for solving the problem

3.3.1 Audio as data

The first step to do speech recognition for our dataset will be to extract features. This is the identifying of components of the audio signal that are good for identifying the content while eliminating background noise. Mel Frequency Cepstral Coefficients (MFCCs), Introduced by Davis and Mermelstein are widely used for automatic speech recognition [21].

First the signal needs to be framed into short frames, then for each frame we need to calculate the periodogram estimate of the power spectrum. After this we apply the mel filterbank to the power spectra and summing the energy in each filter. The fourth step will be to take the logarithm of all filterbank energies, then taking the DCT of the log filterbank energies before finally keeping the DCT coefficients 2 – 13 and discarding the rest of them [22].

The MFCC process transforms audio into a table of data where each row represents a range of frequencies and each column represents a span of time. The value contained in each cell represents the amplitude of its associated range of frequencies during that span of time. The spectrogram shows each cell as a colored block, the intensity of which varies by amplitude [23]. Examples of the spectrograms on some of the keywords are as shown below:

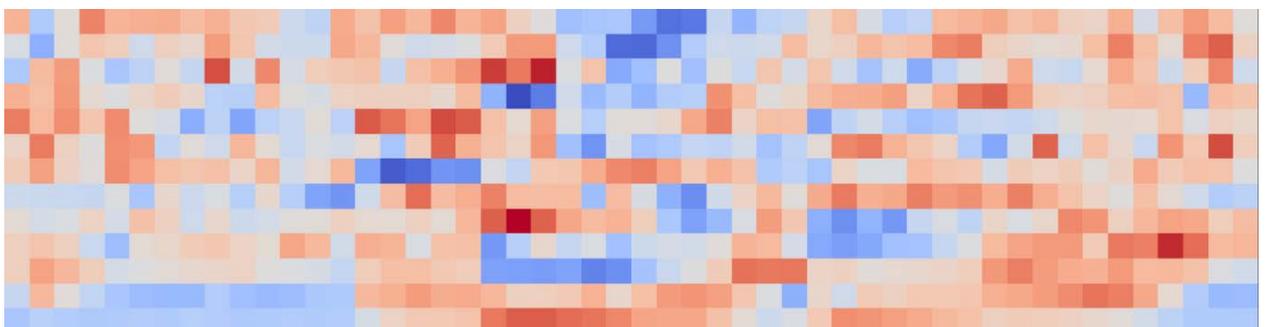


Figure 2 Spectrogram of unknown speech

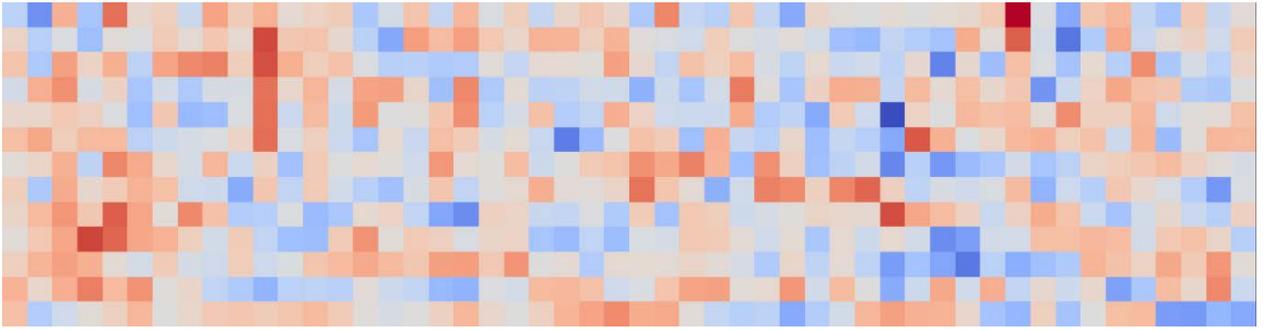


Figure 3 Spectrogram of background noise

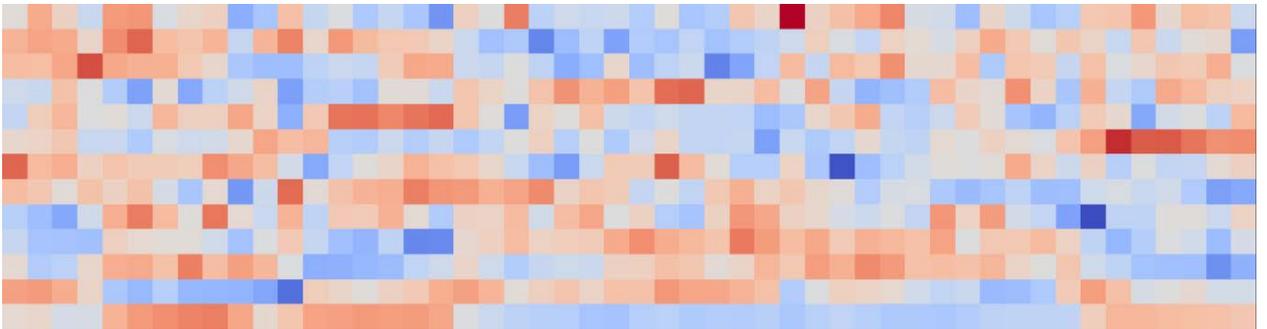


Figure 4 Spectrogram of "stop"

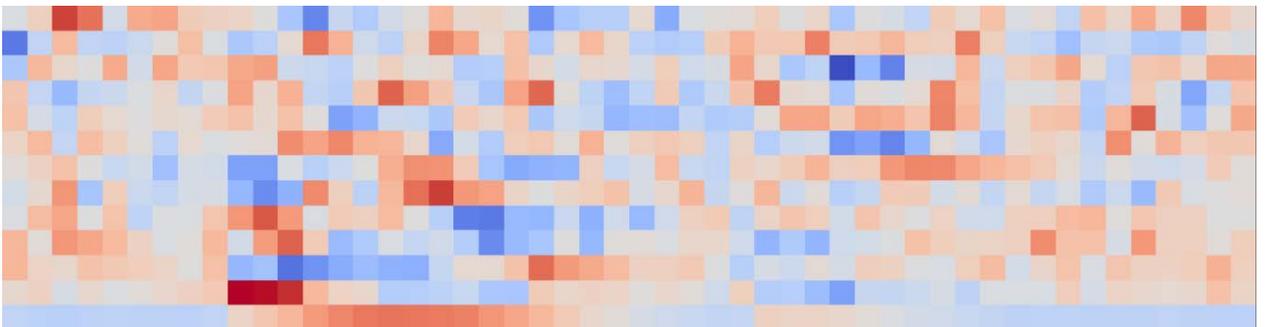


Figure 5 Spectrogram of "right"

3.3.2 Convolutional neural networks

Convolutional neural networks (CNN) are Deep Learning algorithms that take image inputs then assign weights and biases to various parts of the image to be able

to differentiate it from another image. For the Convolutional Neural Network in this thesis, we will use the spectrograms of the different words we are trying to recognize as input. It will then reduce the images into a form that is easier to process without losing features critical for good prediction [24].

3.3.2.1 Convolution Layer

The convolution operation extracts high level features from the input image. Convolutional neural networks can have multiple convolution layers with the first being responsible for capturing low-level features and adapting to high-level with added layers, giving a network with a wholesome understanding of images in the dataset [25].

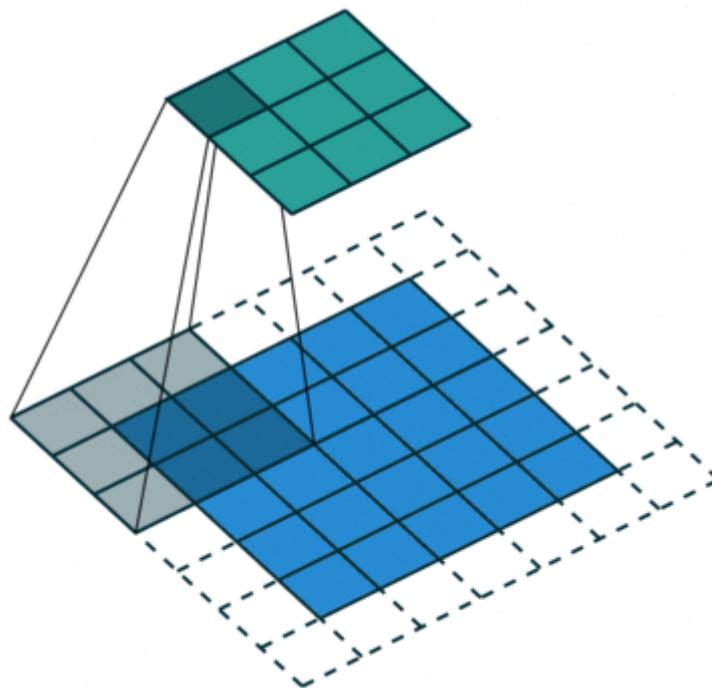


Figure 6 Convolutional Operation

3.3.2.2 Pooling Layer

By reducing the spatial size of the convolved feature, the pooling layer decreases the computational power required to process the data through

dimensionality reduction. The Convolutional layer and the pooling layer together form the i -th layer of a convolutional neural network, with the number of such layers dependent on the image complexity. This process enables the model to understand the features [26].

3.3.2.3 Classification

This phase flattens the image into a column vector then fed to a feed-forward neural network and back propagation applied to every iteration of training since the input image has already been converted into a suitable form for our multi-level perceptron. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classifies them using the Softmax classification technique [27].

3.4 Conclusion

The model trained in this chapter will then be converted into a TensorFlow Lite model which can now run on microcontrollers. This TensorFlow Lite model will be the main inferencing component of the EMBEDSpeech Library interface for different vendors, the focus in this thesis being Arduino. This Library will deliver a seamless and efficient keyword spotting service for the Google Speech Command Dataset on MCUs. The user will be able to use this library to perform machine learning on their Low Power MCU for real-time speech recognition.

4 Deployment and Testing

4.1 Deployment

The MBEDSpeech library has been added to the official Arduino Library Manager as shown in the list below of all the repos that are included in the Library Manager.

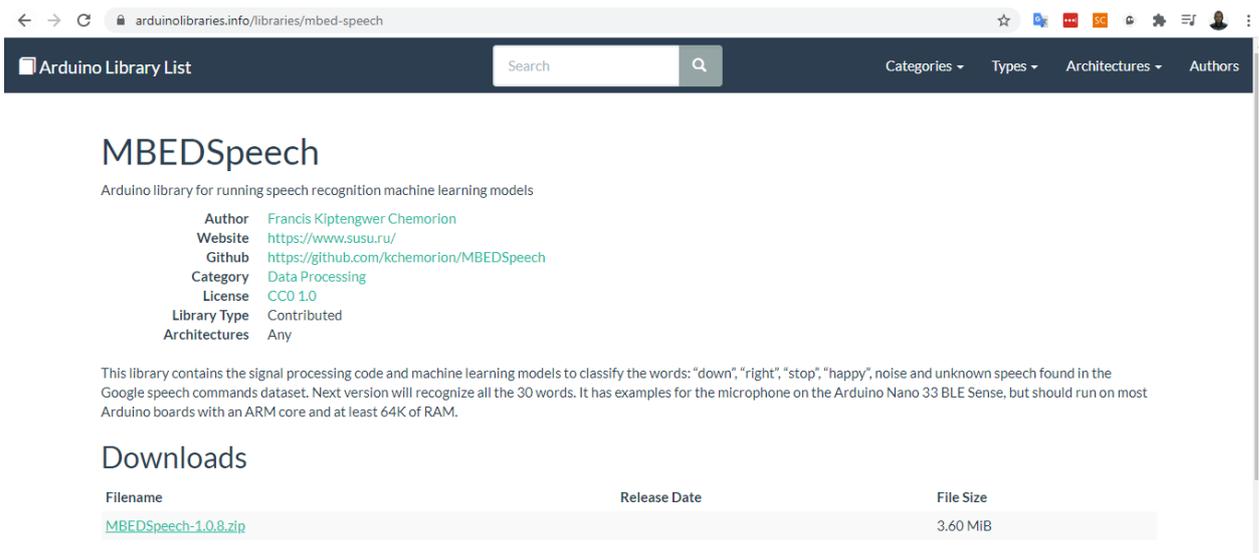


```

{
  "category": "Device Control",
  "architectures": [
    ""
  ],
  "types": [
    "Contributed"
  ],
  "repository": "https://github.com/siktec-lab/SIK-Rotary-Encoder.git",
  "url": "https://downloads.arduino.cc/libraries/github.com/siktec-lab/SIKTEC_Rotary-1.0.2.zip",
  "archiveFileName": "SIKTEC_Rotary-1.0.2.zip",
  "size": 8413,
  "checksum": "SHA-256:1423701a4e80309dba85e9ed837642b4d6b1c53ec094fa0185d57785d27def29"
}
}
{
  "name": "MBEDSpeech",
  "version": "1.0.8",
  "author": "Francis Kiptengwer Chemorion \u003ckchemorion@gmail.com\u003e",
  "maintainer": "Francis Kiptengwer Chemorion \u003ckchemorion@gmail.com\u003e",
  "sentence": "Arduino library for running speech recognition machine learning models",
  "paragraph": "This library contains the signal processing code and machine learning models to classify the words: \"down\", \"right\", \"stop\", \"happy\", noise and unknown speech found in the Google speech commands dataset. Next version will recognize all the 30 words. It has examples for the microphone on the Arduino Nano 33 BLE Sense, but should run on most Arduino boards with an ARM core and at least 64K of RAM.",
  "website": "https://www.susu.ru",
  "category": "Data Processing",
  "architectures": [
    ""
  ],
  "types": [
    "Contributed"
  ],
  "repository": "https://github.com/kchemorion/MBEDSpeech.git",
  "dependencies": [
    {
      "name": "Arduino_LSM9DS1"
    }
  ],
  "url": "https://downloads.arduino.cc/libraries/github.com/kchemorion/MBEDSpeech-1.0.8.zip",
  "archiveFileName": "MBEDSpeech-1.0.8.zip",
  "size": 3771413,
  "checksum": "SHA-256:4dd62f3fafb12df45cd4b773249e57ea586419a8c3bac7d653bbecad0f314b48"
}
}
}

```

Figure 7 EMBEDSpeech entry in the Arduino Library Index



MBEDSpeech
Arduino library for running speech recognition machine learning models

| | |
|---------------|---|
| Author | Francis Kiptengwer Chemorion |
| Website | https://www.susu.ru/ |
| Github | https://github.com/kchemorion/MBEDSpeech |
| Category | Data Processing |
| License | CC0 1.0 |
| Library Type | Contributed |
| Architectures | Any |

This library contains the signal processing code and machine learning models to classify the words: "down", "right", "stop", "happy", noise and unknown speech found in the Google speech commands dataset. Next version will recognize all the 30 words. It has examples for the microphone on the Arduino Nano 33 BLE Sense, but should run on most Arduino boards with an ARM core and at least 64K of RAM.

Downloads

| Filename | Release Date | File Size |
|--------------------------------------|--------------|-----------|
| MBEDSpeech-1.0.8.zip | | 3.60 MiB |

Figure 8 EMBEDSpeech Arduino Library description on arduinolibraries.info

4.2 Implementation of interfaces

4.2.1 Arduino Example sketch compiling and running

To add the Library to Arduino IDE, the user has 2 options. The first option is by using the Arduino Library Manager. This can be found in the Tools > Manage Libraries menu or using the shortcut ctrl + shift + I in the resulting window, search for “MBEDSpeech”, then click on install when the Library is found as shown below:

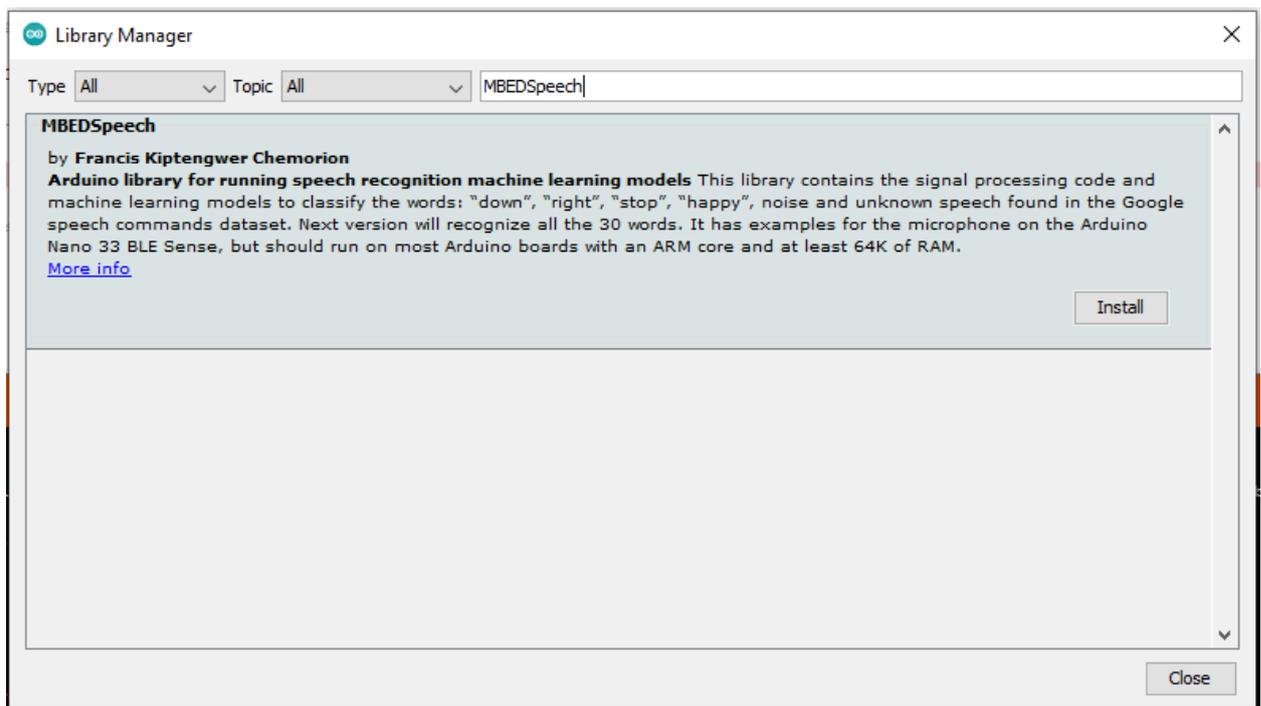


Figure 9 EMBEDSpeech Library in Arduino Library Manager

Alternatively, the user can go to the MBEDSpeech git repository at the link:

<https://github.com/kchemorion/MBEDSpeech> and download the zip either from the latest development branch or from the certified releases:

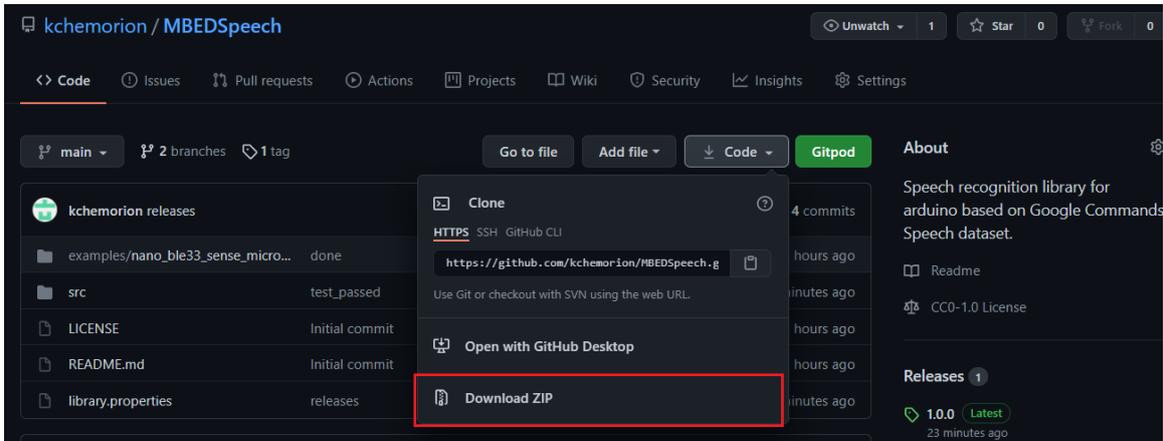


Figure 10 EMBEDSpeech Github

Or

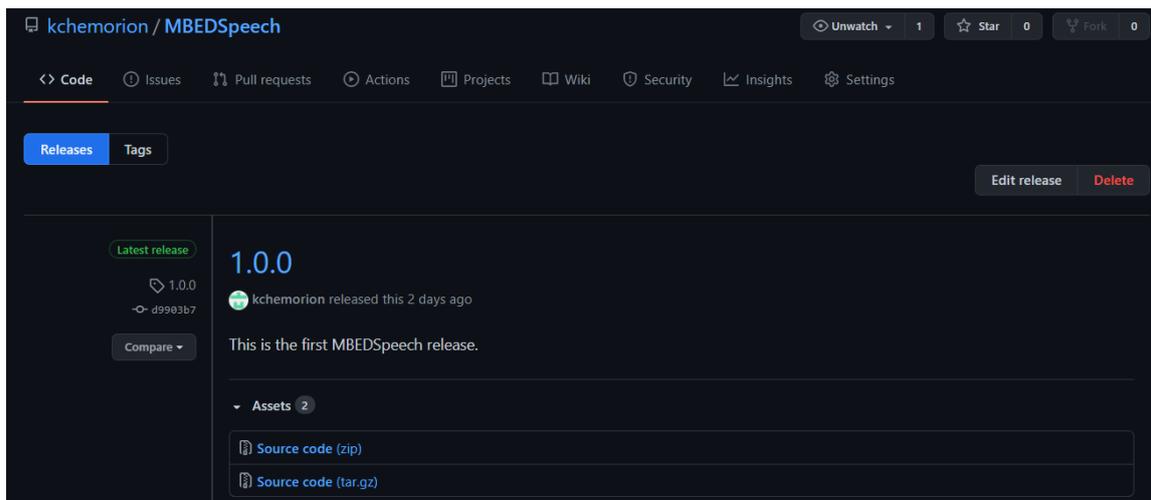


Figure 11 EMBEDSpeech Github Release

Then add the zip file manually through the sketch menu:

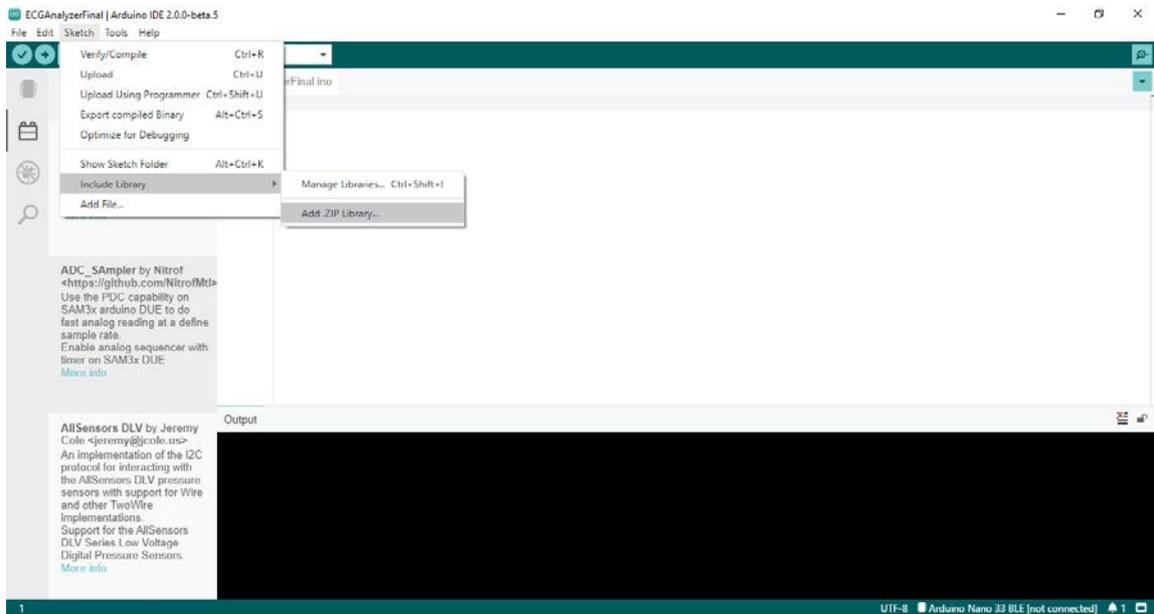


Figure 12 EMBEDSpeech Library Manual Addition

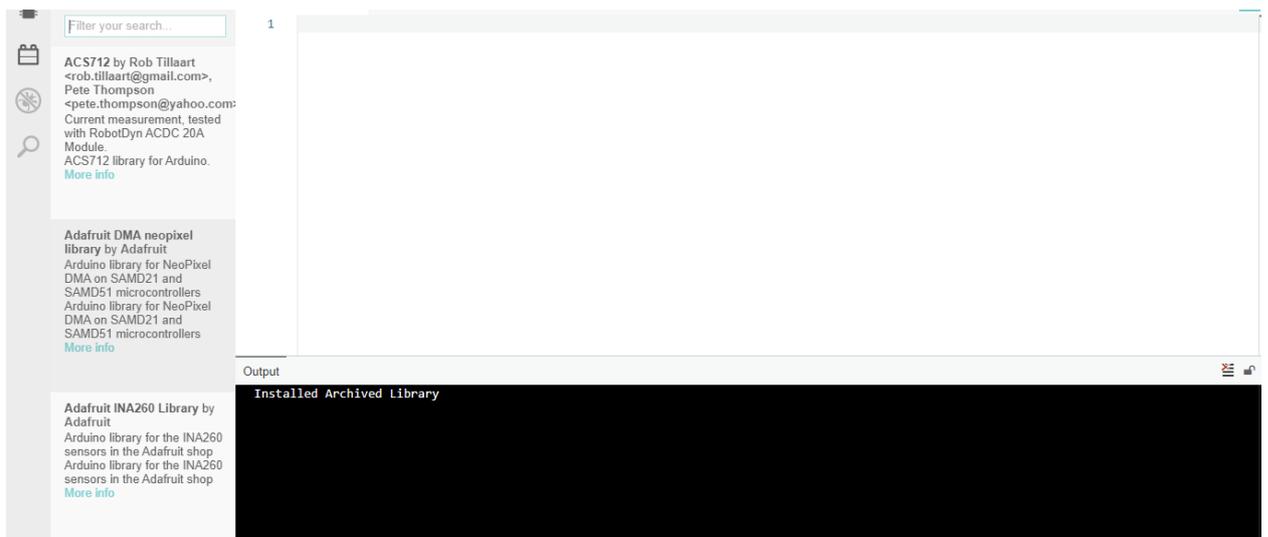


Figure 13 EMBEDSpeech Library Externally added

And runs successfully giving correct predictions when words are said through the microphone:

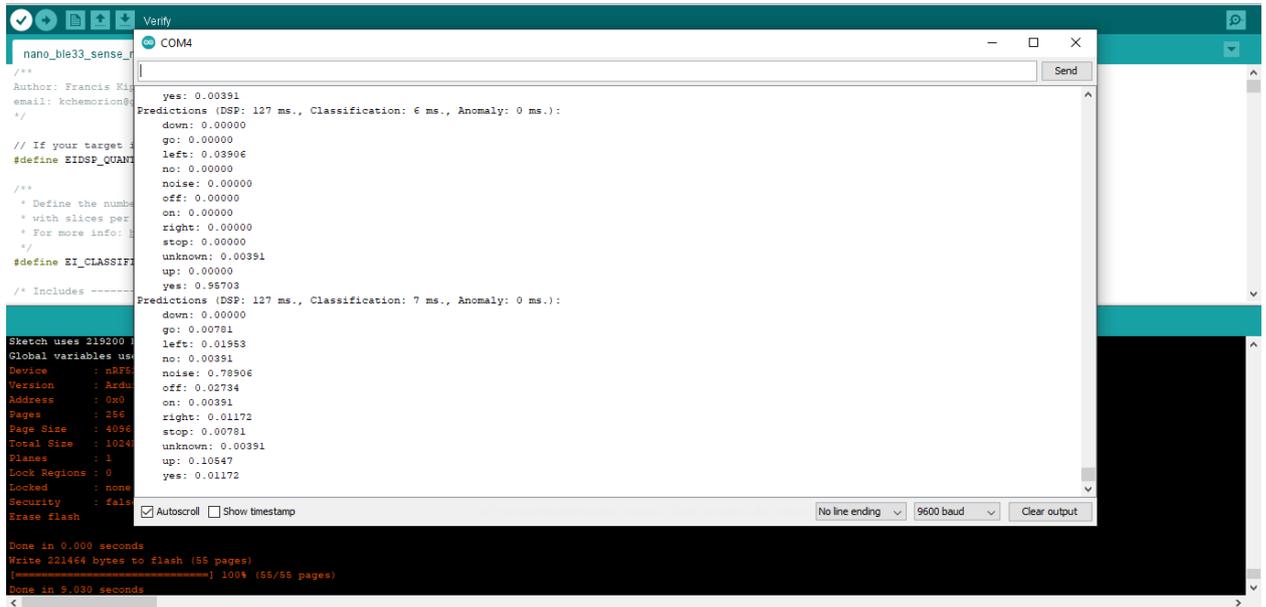


Figure 16 EMBEDSpeech Inference running

4.3 Testing the Machine Learning model

For testing the neural network, 20% of the data used for training and 80% sample data that the CNN had not interacted with was used to evaluate how the model is performing. Accuracy of the model was measured as the percentage of windows of audio that were correctly classified

A Confusion matrix is a table showing the balance of correctly versus incorrectly classified windows. This is by comparing the values in each row, the On-device performance region shows statistics about how the model is likely to run on-device, inferencing time is an estimate of how long the model will take to analyze one second of data on a typical microcontroller, peak is how much RAM will be required to run the model on-device.

For EMBEDSpeech, the following were the results of testing the model: The accuracy is 87.6%, an inferencing time of 4ms, peak ram usage of 4.3k and ROM usage of 47.3k. The confusion matrix is as shown below:

| | DOWN | GO | LEFT | NO | NOISE | OFF | ON | RIGHT | STOP | UNKNOWN | UP | YES |
|----------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| DOWN | 73.30% | 13.10% | 0.30% | 2.10% | 3.60% | 0.60% | 0.90% | 0% | 1.50% | 1.20% | 1.80% | 1.80% |
| GO | 8.40% | 74.00% | 0.60% | 0.60% | 4.30% | 3.10% | 1.20% | 0.30% | 1.90% | 0.90% | 3.40% | 1.20% |
| LEFT | 0.60% | 0.60% | 79.60% | 0.30% | 5.20% | 1.20% | 0% | 4.30% | 0.60% | 0.30% | 0.60% | 6.70% |
| NO | 4.00% | 18.00% | 2.10% | 67.00% | 3.10% | 0.60% | 0.90% | 0.60% | 0% | 0.60% | 1.80% | 1.20% |
| NOISE | 0% | 0% | 0.20% | 0% | 96.60% | 0% | 0.20% | 1.00% | 0.20% | 0.20% | 0.70% | 0.70% |
| OFF | 0% | 0.90% | 0.30% | 0% | 4.90% | 78.70% | 0.90% | 0% | 0.90% | 0% | 13.00% | 0.30% |
| ON | 0.90% | 0% | 0.30% | 0% | 5.10% | 5.70% | 84.10% | 0.90% | 0% | 0.60% | 2.40% | 0% |
| RIGHT | 0% | 0% | 2.50% | 0% | 3.90% | 0.60% | 1.10% | 91.70% | 0% | 0% | 0.30% | 0% |
| STOP | 0.30% | 3.40% | 0% | 0% | 15.80% | 2.10% | 0% | 0% | 68.50% | 0.30% | 9.60% | 0% |
| UNKNOWN | 2.00% | 10.70% | 6.10% | 3.10% | 11.20% | 4.10% | 18.90% | 14.80% | 4.10% | 18.90% | 4.10% | 2.00% |
| UP | 0.30% | 2.10% | 0.60% | 0% | 14.10% | 6.50% | 0.90% | 0% | 0.30% | 0.30% | 75.10% | 0% |
| YES | 0% | 0.30% | 4.70% | 0.30% | 4.00% | 0% | 0% | 0.30% | 0% | 0.30% | 0% | 90.10% |
| F1 SCORE | 0.78 | 0.67 | 0.82 | 0.78 | 0.77 | 0.78 | 0.83 | 0.89 | 0.77 | 0.3 | 0.72 | 0.88 |

Figure 17EMBEDSpeech Confusion Matrix

4.4 Testing the library using Arduino Lint

The Arduino team created a tool to check Arduino projects for common problems. Arduino Lint runs over 175 checks on your sketches, libraries, and boards platforms which cover specification compliance, Library Manager submission requirements, and best practices.

When I run Arduino Lint on my Library, all checks are passed as seen in the outputs below:

```
C:\Users\kiptengwer\Downloads\MBEDspeech-arduino-1.0.0>arduino-lint --library-manager submit
Linting library in C:\Users\kiptengwer\Downloads\MBEDspeech-arduino-1.0.0

Finished linting project. Results:
Warning count: 0
Error count: 0
Rules passed: true

-----

Linting sketch in C:\Users\kiptengwer\Downloads\MBEDspeech-arduino-1.0.0\examples\nano_ble33_sense_microphone_continuous

Finished linting project. Results:
Warning count: 0
Error count: 0
Rules passed: true

-----

Finished linting projects. Results:
Warning count: 0
Error count: 0
Rules passed: true

C:\Users\kiptengwer\Downloads\MBEDspeech-arduino-1.0.0>
```

Figure 18 Arduino Lint Test Results

```

C:\Users\kiptengwer\Downloads\MBEDspeech-arduino-1.0.0>arduino-lint --library-manager submit --format json
{
  "configuration": {
    "paths": [
      "C:\\Users\\kiptengwer\\Downloads\\MBEDspeech-arduino-1.0.0"
    ],
    "projectType": "all",
    "recursive": false
  },
  "projects": [
    {
      "path": "C:\\Users\\kiptengwer\\Downloads\\MBEDspeech-arduino-1.0.0",
      "projectType": "library",
      "configuration": {
        "compliance": "specification",
        "libraryManager": "submit",
        "official": false
      },
      "rules": [],
      "summary": {
        "pass": true,
        "warningCount": 0,
        "errorCount": 0
      }
    },
    {
      "path": "C:\\Users\\kiptengwer\\Downloads\\MBEDspeech-arduino-1.0.0\\examples\\nano_ble33_sense_microphone_continuous",
      "projectType": "sketch",
      "configuration": {
        "compliance": "specification",
        "libraryManager": "submit",
        "official": false
      },
      "rules": [],
      "summary": {
        "pass": true,
        "warningCount": 0,
        "errorCount": 0
      }
    }
  ],
  "summary": {
    "pass": true,
    "warningCount": 0,
    "errorCount": 0
  }
}
C:\Users\kiptengwer\Downloads\MBEDspeech-arduino-1.0.0>_

```

Figure 19 Arduino Lint test results json

4.5 Conclusion

In this chapter we have looked at the deployment of EMBEDSpeech to the Arduino library manager and have also conducted testing for the machine learning model which returned an accuracy of 86% and tested the Arduino library using Arduino lint. The library is now ready for use by other developers.

5 Conclusion

This project undertakes a viable solution for the need of machine learning at the very basic level, that is, in low powered embedded devices. The project will enable us to bring device with a microcontroller under the control of our voice without having connect to any speech recognition cloud services. It basically uses the commands in the speech command dataset, which is carefully chosen and ideal for robotic applications.

The tasks solved in this thesis include

- a) Development of a Library that allows real-time classification of raw audio that can run in Arduino microcontrollers. This is by developing a machine learning model that can identify different words in the Speech commands dataset.
- b) Development of a library that runs on devices that consume low power and have low processing capabilities by converting the machine learning model into a TensorFlow Lite model that can run in very small devices.
- c) Testing of the library has been done using Arduino Lint and an example of how to implement interfaces is added into the Library and can be accessed using the Arduino IDE.
- d) Deployment of the library in a version control environment (GitHub) has been done. The link to the repository is: <https://github.com/kchemorion/MBEDSpeech.git> .
- e) This Library has been deployed into the Arduino library manager for all MCU architectures. Link to information about the Library is : <https://www.arduinolibraries.info/libraries/mbed-speech> .

Due to the successful running of the initial library release, the next steps will be to add the rest of the keywords present in the speech dataset with the 1.0.2 release.

Optimizations as well will be done to tune the performance of the library as shown below:

5.1 Optimizing Latency

Designing model architectures is difficult and time-consuming, but there have recently been some advances in automating the process, such as MnasNet, using approaches like genetic algorithms to improve network designs. These are still not at the point of entirely replacing humans.

I am therefore looking forward to using ready services like AutoML that allow users to avoid many of the gritty details of training, be able to design the best possible model for your data and efficiency trade-offs solving latency issues.

5.2 Optimizing Power Usage

For this I will try to estimate how much power the model uses on different devices by measuring the latency for running one inference, and then multiplying the average power usage of the system for that period to get the energy usage. After knowing how many arithmetic operations a model requires, and roughly how many operations per second a processor can perform, I can roughly estimate the time that model will take to execute. I intend to get these device power usage numbers at a particular frequency and voltage from datasheets.

5.3 Optimizing Model and Binary Size

Currently during training, weights are usually stored as floating-point values, taking up 4-bytes each in memory. Because space is such a constraint for embedded devices, I will use the compression utility in TensorFlow Lite to reduce those values down to a single byte in a process called quantization. It works by keeping track of the minimum and maximum values stored in a float array, and then converting all the values linearly to the closest of 256 values equally spaced within that range. These codes are each stored in a byte, and arithmetic operations can be performed on them with a minimal loss of accuracy.

References

1. «Speech Recognition,» IBM, 2 9 2020. [В Интернетe]. Available: <https://www.ibm.com/cloud/learn/speech-recognition>.
2. H. S. X. L. P. W. Wenhong Lv, «Application of speech recognition technology».
3. A. Brown, «The Role of Voice in IoT Applications,» London, 2015.
4. D. S. Pete Warden, «TinyML,» California, Published by O'Reilly Media, Inc., 2020.
5. «About the Kaldi project,» [В Интернетe]. Available: <http://kaldi-asr.org/doc/about.html>.
6. «Speechmatics API,» [В Интернетe]. Available: <https://app.speechmatics.com/api-details>.
7. «Speechmatics Software Overview,» [В Интернетe]. Available: <https://www.techjockey.com/detail/speechmatics#:~:text=What%20is%20Speechmatics%3F,also%20be%20embedded%20in%20devices..>
8. «Speech-to-Text,» [В Интернетe]. Available: <https://cloud.google.com/speech-to-text>.
9. «Speech-to-Text basics,» Google, [В Интернетe]. Available: <https://cloud.google.com/speech-to-text/docs/basics>.
10. H. Chen, «Does Word Error Rate Matter?,» 28 1 2021. [В Интернетe]. Available: [https://www.smartaction.ai/blog/does-word-error-rate-matter/#:~:text=Word%20Error%20Rate%20\(WER\)%20is,word%20error%20rate%20of%204%25..](https://www.smartaction.ai/blog/does-word-error-rate-matter/#:~:text=Word%20Error%20Rate%20(WER)%20is,word%20error%20rate%20of%204%25..)
11. «What is WER? What Does Word Error Rate Mean?,» 2021. [В Интернетe]. Available: <https://www.rev.com/blog/resources/what-is-wer-what-does-word-error-rate-mean>.

12. 2021. [B Интернетe]. Available:
https://github.com/syhw/wer_are_we.
13. «Google Speech Commands,» [B Интернетe]. Available:
https://pyroomacoustics.readthedocs.io/en/pypi-release/pyroomacoustics.datasets.google_speech_commands.html#:~:text=Google%20Speech%20Commands-,Google's%20Speech%20Commands%20Dataset,Creative%20Commons%20BY%204.0%20license.&text=Create%20the%20sound%20obj.
14. P. Warden, «Launching the Speech Commands Dataset,» Google, 24 08 2017. [B Интернетe]. Available:
<https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html>.
15. «TensorFlow is an end-to-end open source platform for machine learning,» [B Интернетe]. Available:
<https://www.tensorflow.org/overview>.
16. S. Yegulalp, «What is TensorFlow? The machine learning library explained,» 18 06 2019. [B Интернетe]. Available:
<https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html#:~:text=TensorFlow%20is%20a%20Python%20friendly,machine%20learning%20faster%20and%20easier&text=Machine%20learning%20is%20a%20complex%20discipline.&te>.
17. «Custom Federated Algorithms,» Tensorflow, [B Интернетe]. Available:
https://www.tensorflow.org/federated/tutorials/custom_federated_algorithms_1.

18. «Understanding the GitHub flow,» 04 07 2020. [В Интернетe]. Available: <https://guides.github.com/introduction/flow/>.
19. J. Clancy, «The Advantages and Disadvantages of Using GitHub,» 19 11 2020. [В Интернетe]. Available: <https://www.codeclouds.com/blog/advantages-disadvantages-using-github/>.
20. «About Colaboratory,» [В Интернетe]. Available: <https://colab.research.google.com/notebooks/welcome.ipynb?hl=ru>.
21. V. Passricha и R. K. Aggarwal, «Convolutional Neural Networks for Raw Speech Recognition,» 2018.
22. O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn и D. Yu, «Convolutional Neural Networks,» 2014.
23. J. Le, «The 3 Deep Learning Frameworks For End-to-End Speech Recognition That Power Your Devices,» 2019.
24. D. Palaz, M. Magimai.-Doss и R. Collobert, «Convolutional Neural Networks-based continuous speech recognition using raw speech signal,» 2015.
25. S. K. Gouda, S. Kanetkar, V. Harrison и M. K. Warmuth, «Speech Recognition: Key Word Spotting through Image,» 2020.
26. N. Dimmita и P. Siddaiah, «Speech Recognition Using Convolutional Neural Networks,» 2018.
27. O. Abdel-Hamid, A.-r. Mohamed, H. Jiang и G. Penn, «APPLYING CONVOLUTIONAL NEURAL NETWORKS CONCEPTS TO HYBRID NN-HMM,» 2019.
28. «Mel Frequency Cepstral Coefficient,» [В Интернетe]. Available: <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>.

29. «Implementing production-ready live audio transcription using Speech-to-Text,» Google, [В Интернете]. Available: <https://cloud.google.com/speech-to-text/docs/tutorials>.