Blind Assistant System (Lamba)

Eyuael Bezabeh, Helen Girma, Tomas Beyene, Zerihun H/Michael,
St Mary's University

Abstract

Vision is a fundamental part of our lives. Information is present everywhere in different forms, but most of it is processed by the visual organ, the eyes. In this sense, blind people have great difficulty for getting information from the environment. Especially in today's technologically developed world, how they can cope up with is a great challenge. Locally, almost no product is found that aids blind people in navigating and interacting with technological tools. The main objective of this paper is designing an efficient blind assistance system for the visually impaired people using Artificial Neural Network (ANN) with modern cloud service. ANN is based on a supervised procedure and comprises three layers: input, hidden, and output. The system logically has two phases. The first is Automatic Speech Recognition (ASR). This class of application starts with a clip of spoken command audio in Amharic language and extracts the words that were spoken, as text. Not only do they extract the text but they also interpret and understand the semantic meaning of what was spoken, so that they can respond with answers, or take actions based on the user's commands. This phase is based on Artificial Neural Network (ANN) Model that trains with audio Dataset in order to identify the input audio commands. The second phase, Speech synthesis is where artificially produced human speech delivers the information requested by the blind person. Through the system, blind individuals will peruse basic technological services.

Introduction

Humans have five senses: vision, hearing, touch, smell, and taste. All of these senses are important in our daily lives. However, what do you think which is your most important sense? In other words, which sense would it scare you the most to lose? We asked this question in random survey in St. Mary's University. About 85% of participants said that they are most scared of losing their sense of vision. It is easy to see why we might feel that losing the sense of vision would be worse than losing any other sense, primarily because of the importance of vision in our daily lives. 80 percent of all learning comes through the eyes. Many of the movements we perform to do tasks such as reading books and personal interactions we make rely on vision in some way. The same applies to many of our daily routines. So, it is easy to understand why people are afraid of losing their sense of vision; it would change their lives profoundly.

"In a world built on the ability to see, vision, the most dominant of our senses, is vital at every turn of our lives. The newborn depends on vision to recognize and bond with its mother; the toddler, to master balance and learn to walk; the schoolboy, to walk to school, read and learn; the young woman to participate in the workforce; and the older woman, to maintain her independence." Dr Tedros Adhanom Ghebreyesus, Director-General, World Health Organization

According to World Health Organization (WHO) globally, at least 2.2 billion people have a near or distance vision impairment. In at least 1 billion – or almost half – of these cases, vision impairment could have been prevented or has yet to be addressed. This 1 billion people include those with moderate or severe distance vision impairment or blindness due to unaddressed refractive error (88.4 million), cataract (94 million), glaucoma (7.7 million), corneal opacities (4.2 million), diabetic retinopathy (3.9 million), and trachoma (2 million), as well as near vision impairment caused by unaddressed presbyopia (826 million). In terms of regional differences, the prevalence of distance vision impairment in low- and middle-income regions is estimated to be four times higher than in high income regions. Vision impairment poses an enormous global financial burden. For example, the annual global costs of productivity losses associated with vision impairment from uncorrected myopia and presbyopia alone were estimated to be US$ 244 billion and US$ 25.4 billion, respectively. To reduce such economic loss and improve the contribution of visually impaired people, societies must offer a wide range of support to blind people, so that they can remain active members of their communities. Losing the sense of vision does not normally endanger a person's survival or even that person's ability to function in society. Interestingly, it has been shown that hearing and touch may become more sensitive in blind individuals.

To help blind individuals function and participate in the society, many assistive tools and technologies were developed. These are tools such as canes (blind stick) which help blind individuals to walk; and the Braille to read and write; technologies like optical character reader (OCR); JAWS screen reader; electronic mobility aids; and recently developing area digital assistant system that enables the visually impaired to navigate devices through voice input. Most of these assistant technologies are in foreign languages. Thus, this paper aims at adapting this technology to 1.8 million or 1.6% of the Ethiopian blind and 4.1 million or 3.7% who live with low vision.

Background of the Study

It has long been assumed that in the ancient world the blind enjoyed few opportunities and lived out their days in penury as beggars. Even some societies saw blindness as curse from God. As a result, the blind and the visually impaired didn't get enough support and attention that must be given to them. Despite the disadvantage that is upon them, there were some extraordinary blind individuals that moved the world through their work. They showed the world what they can contribute to the community as any person would do if they get the right support. Not having the contribution of blind and visually impaired community (BVI) has a huge impact on the socio-economic development of a country. Though attention given to them is showing improvement, still they are struggling to get the basic services. There are some systems and devices that are developed by organizations and interested groups such as the Braille for reading, normal canes and Smart Canes which aid in movement and to check obstacles, magnifying technology, Optical Character Reader (OCR), Screen Reader, Voice Assistance etc.

The blind and the visually impaired should be able to use the visual devices that are with us every day which are becoming part of ourselves by the day. Devices such as phones, tablets, personal computers and others are utilized for enjoyment, work, and communication and for other kinds of utility purposes; the list is endless. The visually impaired could benefit a great deal from an assisting feature that compensates the inability to interact visually using the devices. Both input and output are dependent on a visual interstitial, understanding, interaction most of the time.

This presents itself a challenge to the visually impaired as operationalization of the devices is very difficult, leaving out section of society that could benefit just as or, in fact, even more than everyone else. Devices such as mobile phones present even greater advantages to users with disabilities because they bring the world to one's grip. We can look at it this way: the devices are a "plus" to users that do not have any disabilities because users can in a longer road potentially access most of the basic uses the devices are used to achieve. But to disabled users, these devices are extension of a certain impaired ability whether it is mobility, vision, hearing or speaking. To look at it in a revolutionary manner, our team tends to perceive the device as nothing but extensions of ability for the species. Voice Assistance is nothing new, as implemented on almost all operating systems in consideration of people with disability. We have observed that the implemented feature across the numerous platforms can improve to a degree that visually impaired can significantly take advantage of the devices.

There are many kinds of voice assistance systems across platforms and operating systems: Android, Apple, Windows, Google, Amazon. Some of these platforms provide voice assistance for a user. Implementation and use case of the systems might have a tiny difference but all are fundamentally the same.

Statement of the Problem

In today's world, new and modern technologies are booming and revolutionizing the way we live. Most of our daily lives' tasks and activities are aided with technologies either directly or indirectly. Since our world has become technology dependent, everyone is entitled to benefit from it, including disabled persons. This project's scope and focus is to solve problems that

blind/visually impaired people face while dealing with technologies like cellphones and computer devices. A person who is fully or partially visually impaired cannot or struggle to communicate with electronic devices as easily as a non-visual impaired person because most of the technologies have visual outputs. Since visually impaired persons cannot see, they will struggle to control and respond to devices they are interacting with. This results in greater distance between the devices and the visually impaired.

The first and foremost obstacle is visual IO. Visual devices are everywhere if we look around our homes, our workplaces and other locations we frequent. It is likely that there are screens on most kitchen and household appliances, on the fitness equipment at gyms, and on the payment device at your favorite cafe. The list is endless and the issue is that while blind people are able to touch a screen, it is a struggle and there is nothing that tells or directs what they are touching or what happens on the screen when they do. Most visually impaired are able to access the technology but enjoy so little of what it provides. The visually impaired face challenges during commanding, navigating and most importantly getting the most out of the devices, gadgets and tools that the world of technology has been providing in these times of rapid evolution.

The world is not slowing down in its evolution with regard to technology; it is out besting itself and the platforms are ever morphing into more accessible and more convenient form of themselves. The problem is that it is leaving out sections of the society that technology can do the most for, the disabled. Current platforms are mostly visually configured, operated and navigated; the visually impaired and the blind community can't make use of current devices easily or to a higher limit because of a number of reasons. The first problem is language; almost all devices come with foreign languages. This becomes a barrier to use or realize the advantages of the devices.

In order to communicate with the device, one's own language will motivate and be inviting to users; to control and to explore more of the device's features. The next problem is responsive devices as with commanding or instructing the device to perform actions.

The devices, when they are or about to perform actions, are not able to report on the state back to the users real-time by giving a clear definition where in point they are logically and in the digital space. Devices, in most cases are visually configured, operated and also navigated. The way to implement a fully capable assistance system that will equalize the lack and absence of vision in a user is yet to be developed, but our goal is to focus on basic utilities of a device.

General Objective

The general objective of this project is to develop a system software with cloud computing architecture resulting in practical digital assistance system for visually impaired and blind people.

Specific Objectives

To develop a better sophisticated, meaningful blind assistance system we need to:

- Analyze the current assistance technology;
- Develop an effective research question;

- Analyze difficulties faced by the visually impaired and the blind;
- Come up with methodology that solves analyzed problems;
- Design a system that overcomes the problem;
- Implement a practical system that easily integrates with the user's life;
- Implement efficient input system to a device;
- Design Bot for voice dataset collection;
- Implement intelligent recognition;
- Implement a well - trained model;
- Assemble and design micro-controller;
- Implement fluent voice synthesis;
- Design micro services and internal utility services such as call, clock, location…;
- Create a system for blind or visually impaired individuals interact with the devices through audio commands;
- Create a system that the blind person gets response (output) from devices laptop/mobile phones through audio means;
- Enrich the experience of visually impaired people of technology through using technology itself;
- Design a well-defined procedure and schedule to design a flexible system; and
- Interpreting the results and draw conclusions.

Methodology

Lamba is based on deep learning with an artificial intelligence (AI) capability that process audio command input from the user through microphone; the audio passes through a serious of neurons in the neural network model which was previously trained by the audio dataset collected by the Bot from volunteer participants. After the input command passes through the model, not only do they extract the text but they also interpret and understand the semantic meaning of what was spoken, which will be pushed to the micro service requested. Finally, the output of the micro service will be synthesized into speech in order to be understood by the user. To make the process of problem solving and reaching into the solution complete, we group them into consecutive steps: Speech Recognition, Micro-service, Speech Synthesis and finally testing with drawing conclusion.

Speech Recognition (Speech to text (STT))

Speech recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text algorithms, is a capability which enables a program to process human speech into a written format. While it is commonly confused with voice recognition, speech recognition focuses on the translation of speech from a verbal format to a text one, whereas voice recognition just seeks to identify an individual user's voice. This class of applications starts with a clip of spoken audio in Amharic language and extracts the words that were spoken, as text. This stage of the project has further sub steps:

Dataset Collection

One of the hardest problems to solve in deep learning has nothing to do with neural models: it is the problem of getting the *right data* in the *right format*. Getting the right data means

gathering or identifying the data that correlates with the outcomes you want to predict. The right end format for deep learning is generally a tensor, or a multi-dimensional array. So, data pipelines built for deep learning will generally convert all data – be it images, video, sound, voice, text or time series – into vectors and tensors to which linear algebra operations can be applied. That data frequently needs to be normalized, standardized and cleaned to increase its usefulness. Deep learning and machine learning, more generally, need a good training set to work properly. Collecting and constructing the training set – a sizable body of known data – takes time and domain-specific knowledge of where and how to gather relevant information. Machine learning typically works with two data sets: training and test. The first set we use is the **training set**. Running a training set through a neural network teaches the network how to weigh different features, adjusting their coefficients according to their likelihood of minimizing errors in your results. The second set is the **test set**. It functions as a seal of approval, and we don't use it until the end. After we trained and optimized our data, we test our neural model against this final random sampling.

The dataset for our project is voice commands from user that, in turn, initiates the micro-service. To gather such data, the team has built a **telegram bot** to make ease of the collection process. The target audio command is planned to be collected from students and various volunteer individuals.

Figure 1: Audio

Lamba's_Bot

Data collection is the core of the project because it takes a good amount of time and cost. The team came up with a bot that aid the process. Lambas_Bot is a telegram bot built to facilitate audio data recording and storing. The bot is user intractable, built with pure GO programing language, and uses telegram API for its functionality. It is deployed in Linux server that is found through sponsorship. To explore further about Lamba's_bot and contribute to the dataset, use the following QR code or use the user's name: ***@Lambas_bot***

Figure 2: Lamaba's Bot Telegram Link QR Code

Prepare Training Data

Include grouping the audio record into directory and prepare metadata for the command word. Then, list the dataset split into training and testing (validation) data. Validation data will be used later after the end of the training process. The training data must have:

- The features (X) are the audio file paths
- The target labels (y) are the class names

	File_name	Relative_path	classID	Class_name
0	10003-2-00.wav	/folder1/10003-2-00.wav	0	lamba
1	1003-2-45.wav	/folder6/1003-2-45.wav	5	felegi
2	10057-2-05.wav	/folder11/10057-2-05.wav	0	deweyi
3	10335-2-75.wav	/folder8/ 10335-2-75.wav	3	dmeri
4	10883-2-11.wav	/folder4/10883-2-11.wav	1	kne

Table 1: Path Structure

Pre-processing Training Data

After audio data is collected the next step is processing and augmenting audio data for deep learning models, since audio file paths cannot be inputted directly into the model. Audio pre-processing will all be done dynamically at runtime by the neural network when we read and load the audio files. The sub-steps include read and load audio from the file, converting mono files into stereo, resampling to standardize and convert all audio to the same sampling rate, resize all the records to the same length, **Data Augmentation** which is the main process as will be discussed later, and lastly, converting the augmented audio to final yield of pre-processing **Mel Spectrograms**.

Data Augmentation

A common technique to increase the diversity of dataset, particularly when we do not have enough data, is to augment data artificially. We do this by modifying the existing data samples in small ways. For instance, with images, we might do things like rotate the image slightly, crop or scale it, modify colors or lighting, or add some noise to the image. Since the semantics of the image have not changed materially, the same target label from the original sample will still apply to the augmented sample.

Spectrogram x cvAugmentation

- Frequency mask — randomly mask out a range of consecutive frequencies by adding horizontal bars on the spectrogram.
- Time mask — similar to frequency masks, except that we randomly block out ranges of time from the spectrogram by using vertical bars.

Raw Audio Augmentation

- Time Shift — shift audio to the left or the right by a random amount.
- Pitch Shift — randomly modify the frequency of parts of the sound.
- Time Stretch — randomly slow down or speed up the sound.
- Add Noise — add some random values to the sound

Mel Spectrogram VS MFCC (Mel Frequency Cepstral Coefficients)

Think of spectrograms as pictures of audio. Spectrograms allow us to visualize audio and the pressure these sound waves create, thus allowing us to see the shape and form of the recorded sound. It is from this visualization that the neural network learns pattern of the audio and predict the word when tested. **Mel Spectrogram** is a spectrogram that is converted to a **Mel scale** and work well for most audio deep learning applications. We use it during neural network model training. It is better than MFCC because it extracts much better features from the audio that are the most relevant in capturing the essential quality of the sound. **MFCC (Mel Frequency Cepstral Coefficients)** is used in the same way as Mel Spectrogram and widely used in automatic speech and speaker recognition. We use this during model testing or validation stage.

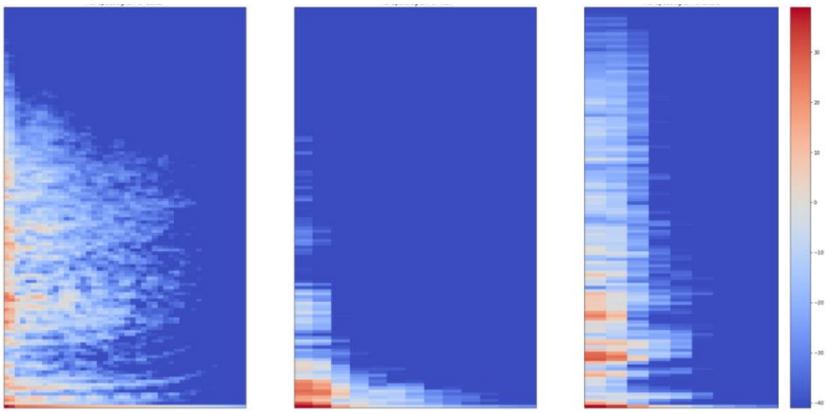


Figure 3: MFCC

Figure Mel Spectrogram of three different sounds, different sound has different Mel Spectrogram

Training Model

Model

Developing Deep Learning model does not really differ that much from any other types of coding. It requires specific background knowledge, but the good coding practices remain the same. For the model we are going to develop, the tool is Keras model.

Keras

Keras is a neural network Application Programming Interface (API) for Python that is tightly integrated with TensorFlow, which is used to build machine learning models. Keras models offer a simple, user-friendly way to define a neural network, which will then be built for you by TensorFlow. The models are used to define TensorFlow neural networks by specifying the attributes, functions, and layers you want. It wraps the efficient numerical computation libraries, ***Theano*** and ***TensorFlow,*** and allows you to define and train neural network models in just a few lines of code. TensorFlow provides a comprehensive machine learning platform that offers both high level and low-level capabilities for building and deploying machine learning models. Keras offers a number of APIs you can use to define your neural network, including:

- **Sequential API**, which lets you create a model layer by layer for most problems. It is straightforward (just a simple list of layers), but it is limited to single-input, single-output stacks of layers.
- **Functional API**, which is a full-featured API that supports arbitrary model architectures. It is more flexible and complex than the sequential API.
- **Model Sub-classing**, which lets you implement everything from scratch. Suitable for research and highly complex use cases, but rarely used in practice.

How to Define a Neural Network with Keras' Functional API

The Keras functional API lets you:

- Define multiple input or output models
- Define models that share layers
- Create an acyclic network graph

Functional API models are defined by creating instances of layers, and connecting them directly to each other in pairs. A model is then defined as that specifies the layers to act as the input and output to the model.

The steps to work with Keras are as follows:

1. Load Data,
2. Define Keras Model,
3. Compile Keras Model,
4. Fit Keras Model,
5. Evaluate Keras Model,
6. Tie It All Together, and
7. Make Predictions.

We create a *Sequential model* and add layers, one at a time, until we are happy with our network architecture. The first thing to get right is to ensure the input layer has the right number of input features. This can be specified when creating the first layer with the **input_dim** argument and setting it to 8 for the 8 input variables

The model includes the following information:

- Layers and their order in the model,
- Output shape (number of elements in each dimension of output data) of each layer,
- Number of parameters (weights) in each layer,
- Total number of parameters in the model.

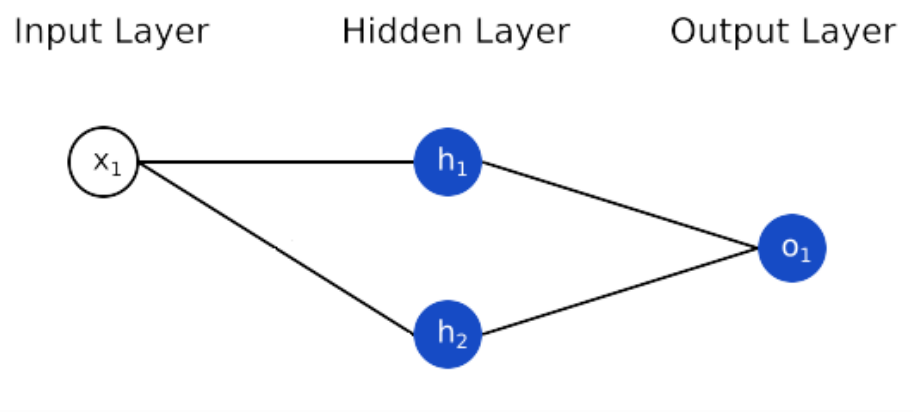


Figure 4: Neural Network

Testing

In this phase, the accuracy of how much speech to text is measured. To help us, we first classify the dataset into training and validation data, so we use validation data here then calculate Word Error Rate (WER).

Word Error Rate (WER): is a common metric for measuring speech-to-text accuracy of automatic speech recognition (ASR) systems. Microsoft claims to have a word error rate of 5.1%. Google boasts a WER of 4.9%. For comparison, human transcriptionists average a word error rate of 4%. The less the WER, the accurate the prediction.

How to Calculate WER: Word Error Rate is a straightforward concept and simple to calculate – it is basically the number of errors divided by the total number of words.

Word Error Rate = (Substitutions + Insertions + Deletions) / Number of Words Spoken
Where errors are:

- Substitution: when a word is replaced.
- Insertion: when a word that was not said is added.
- Deletion: when a word is omitted from the transcript.

Variables that Affect WER

- **Accents and Homophones**
- **Crosstalk**
- **Audio Quality**

Micro-Service

After speech is converted into text, assuming it is the correct one, the next step is to understand the semantic of the text and push it to the right service. Working with micro service has many advantages over writing all the codes in one. It enables to do each service to be written at different time with different language and can be integrated into one system. What makes our micro service different is it uses g**oogle's Remote Procedure Call** (gRPC), a software communication protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details, which is 7 times faster than REST when sending data. The gRPC message through Protocol Buffer

(**protobuf).** The **protobuf** serializes structured data by compiling it to **binary** format which is machine independent and understood by any machine.

Micro services also known as the micro services architecture is an architectural style that structures an application as a collection of services that are:

- Highly maintainable and testable,
- Loosely coupled,
- Independently deployable,
- Organized around business capabilities,
- Owned by a small team.

The micro service architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables technology stack.

The Micro services we are going to implement are:

1. Google search
    - Map Finder
    - Information Finder
2. Social Application Manager
    - Email Inbox Checker
    - Telegram Inbox Checker
3. News Finder
4. Book Finder
5. Weather Report
6. Mobile Command Connector
    - Voice memo
    - Phone call
    - Missed call etc.

Speech Synthesis (Text to speech (TTS)

The result from micro services is delivered to the visually impaired person through speech so that natural word speech must be synthesized. In order to do that, we reverse back the STT. Depending on the schedule and cost, we plan to use one of the following TTS architectures to synthesize speech.

Concatenate — Old School:

A traditional old school technique is that uses a stored speech database where speech is mapped to specific words. While for certain mapped words, you can produce understandable audio, the output speech will not include the natural sounds of voice, “prosody, emotion, etc.”

Tensor Flow TTS:

Tensor Flow TTS provides real time state of the art speech synthesis architectures such as Tacotron-2, Melgan, Multiband-Melgan, FastSpeech, FastSpeech2 based on TensorFlow 2. With Tensorflow 2, we can speed-up training/inference progress, optimize further by

using fake-quantize aware and pruning, make TTS models can be run faster than real-time and be able to deploy on mobile devices or embedded systems.

A hybrid parametric TTS approach relies on a Deep Neural Network consisting of an acoustic model and neural vocoder to approximate the parameters and relationship between input text and the waveform that makes up speech.

Acoustic Model:

- Algorithms are optimized to convert the preprocessed/normalized text into Mel-spectrograms as output.
- For a majority of the algorithms, you need to convert the linguistic features vector to acoustic features to a Mel-Spectrogram. The Spectrogram ensures that we have now accounted for all relevant audio features.

Neural Vocoder:

- The input for the final step is the Mel-Spectrograms that are translated into a waveform via the Neural Vocoder.
- While there are many different types of neural vocoders, the modern ones have a GAN foundation.

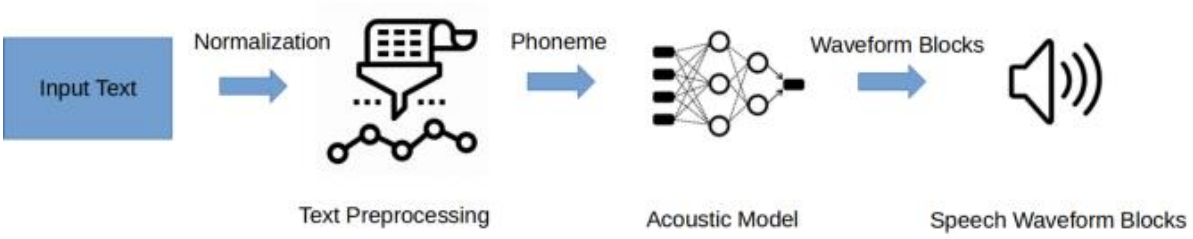


Figure 5: Tensor Flow

Tensor Flow TTS:

- Solely a Text to Speech application optimized for ease.
- Ability to speed-up training/inference progress, optimized further by using quantization aware and pruning leading to near-real-time results
- Built to support real-time speech synthesis.

Tools
Tech Stacks and Tools
Software's Modeling and diagraming tools

- Lucid chart
- Adobe XD

These are the most common modeling tools that are widely used today. The development team will use this modeling tool to develop use cases, class diagrams and other additional models.

Integrated Development Environment (IDE)

VIM - open-source, screen-based text editor program for Unix since some of the team members use UNIX OS.

Visual Studio Code - is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

Postman - is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

Anaconda- is a distribution of the python programming language for scientific computing used mainly for data science.

PyCharm - is an integrated development environment (IDE) used in computer programming, specifically and conventionally used for the Python language. It is developed by the Czech company JetBrains.

GoLand - is an IDE by Jet Brains aimed at providing an ergonomic environment for Go development.

UNIX / Distorts - is a family of multitasking, multiuser computer operating systems. We decided to use any UNIX like OS because of its versatility, security, power, and speed.

Microsoft office 2010 - is a suite of productivity applications that includes Microsoft Word, Microsoft Excel, Microsoft PowerPoint and Microsoft Outlook. We will use this software to make our documentation. The project team will use this tool for preparation of the documentation part of the project.

Microsoft Project Standard - The project team will use this tool for developing the schedule of the project.

Fl studio – is a commercial DAW (Digital Audio Work Station) that is used for producing audio deliverables. It is a tool used for Editing, Recording, Mixing, and Mastering Audio.

Language and Libraries

Libraries

- Python
- Tokenize
- Crawling
- Pyplot
- Keras
- Pickle
- Numpy

Language Stack (What language we'll use)

- Go – for building bot
- Python – to build the model and utilities
- c++ - to build the utilities
- c# - to build the utilities

Version Control - to handle everything from small to very large projects with speed and efficiency

- Git
- gitlab with git

Project Architecture

- Micro service Application Interface (for lamba service) - gRPC with Protobuf
- REST API (for admin panels) - JSON
- we use telegram Web hook API for just a lamba Bots

For test Application Interface (API)

- Postman
- BloomRPC

Google Colab - to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more.

Jupiter Note Books - for interactive computing

TFlit

For load test

- Apache JMeter Metter
- Wrk - has a library of hundreds of bots
- Ghz - to measure computer processing speed

What model we will use for training

- **Keras Model** – for both SST and TTS

For Admin panel we will use

- JWT authentication via REST

Project Structure (How do we manage the project)

- Hexagonal DDD Architecture

Database staff

- Object relational mapping (ORM) – converting data between incompatible type systems using object-oriented programming languages
- Cockroachdb - a distributed database with standard SQL for cloud applications.
- Postgresq - pen-source relational database management system emphasizing extensibility and SQL compliance.

Deployment staff (containerization, orchestration)

- Containerization - Docker
- Orchestration - Docker swarm and kubernetes

Hardware

Micro-controller (Micro-Computer)

- Raspberry Pi Zero 2 W

Recording tools

- Soundcard (M-Audio)
- Condensed Microphone
- Cell phones

Scope

Scope lies in identifying major deliverables, understanding and prioritizing key milestones, identifying major players and valid constraints. Blind assistance system revolves around a visually impaired user and auditory implementations, which is one of the major ways they make sense of the world. The system side implementations also use Machine learning (ML) for Speech Synthesis and Speech Recognition. The major milestone in the progression of the project is voice recognition which the system accepts as command from the user. The next major milestone is Speech Synthesis which the system will be able to respond to the user in auditory format. These can be for utility purposes like telling time, battery level, making Internet searches, reading emails, news, text messages, making phone calls. The features of the system might be updated through time but in a controlled manner so that the project does not go beyond scope and schedule.

We have defined the scope with features and major deliverables as the major deliverables and part of deliverables are the speech recognition and speech synthesis, the system interface. Scope definition is critical in improving the accuracy of cost, time and resource estimates. User will activate and also interact with a digital assistance that will be able to provide numerous services upon demand. The project's main focus is on the improvement as well as implementing newer features.

Limitations

Even though our project is capable of performing much functionality, it has some limitations.

The Perfect Assistance!

No digital systems that are examined and analyzed are perfect; we also are not expecting to implement a perfect assistance system. This is because the technology is not at its best; it takes a great number of team members and resources to collect enough datasets; and the mental power and brainstorming need for a perfect assistance system. We would not lie to ourselves or to anybody of a perfect assistance.

Are you listening?

Training model is what artificial intelligence in most cases relies on. The user need might be to choose places and times to talk to the assistant because trained models always prefer commands in quieter background.

Help me so I can help you?

If the user is not able to articulately pronounce words, sentences, the assistance system will forever find itself unable to reply correctly or be responsive in a way that makes sense. So a user will have to properly articulate and pronounce words, help the digital assistance system, so that, in return, the person can be helped.

What language do you speak?

The system will be able to support language interaction and integration. Language composition, i.e., using one or more languages as a command or instruction will be implemented in Lamba. Supported language is Amharic. The system might struggle to understand commands

composed of the supported language and also use supported language along unsupported language.

Command me if you can!

The user might struggle to come up with a recognized command with supported feature that the system is indeed able to provide.

Significance of the project

When the project was considered, the team has tried to imagine the impact of a system for the visually impaired. It would change the landscape of the technological paradigm where visually impaired users can involve themselves in the digital age alongside everyone else.

- The blind community will be able to use the system to be assisted through operating devices.
- Visually impaired individuals will be able to use the assistance system.
- It produces a system that will further broaden the digital assistance field.
- It increases efficiency of communication with devices including language support.
- It Pioneers a fluent assistant.

Beneficiaries of the project

The system is exceptionally specific and targets no organization, institution, certain society, or boundary. The scope of its use is very specific, yet it benefits users in all margins. It is directly aimed at its first-hand users and beneficiaries of the project.

Blind Users, Primary target users will be able to:

- Interact efficiently with their device.
- Accomplish meaningful tasks through their device.
- Enjoy features previously unavailable like Google searching using voice.
- Language support: The system will be implemented on target user's own language. This in our case is the Ethiopian official language which is Amharic. This will help the blind users to interact with:

Visually Impaired User – through age or through sickness one's vision might decline.

- The system will be able to support a declining vision.
- People with poor vision can also be assisted while enjoying other features they are capable of using.

People with no visual impairment – The system will be able to facilitate the interaction of any user with no disability by minimizing click counts so as the user can do multiple tasks at instants of voicing a command.

Overall:

- Quick navigation is possible;
- Searching will be simplified;
- File handling will be easier;
- It is time saving; and
- It is a modern and intimate way of operating a device.

The faculty - As a research document, this system along with its documentation can serve a great resource to student and teachers for understanding about artificial intelligence and digital assistance systems.

Feasibility Study

Economic Feasibility

The project is economically doable since most of the work is done in a laboratory with basic resource and devices that cost from nothing to a good amount that can be covered by the team's pocket money or by sponsors. So, we are capable of providing the budget for the project. Therefore, we could say that our project is economically feasible.

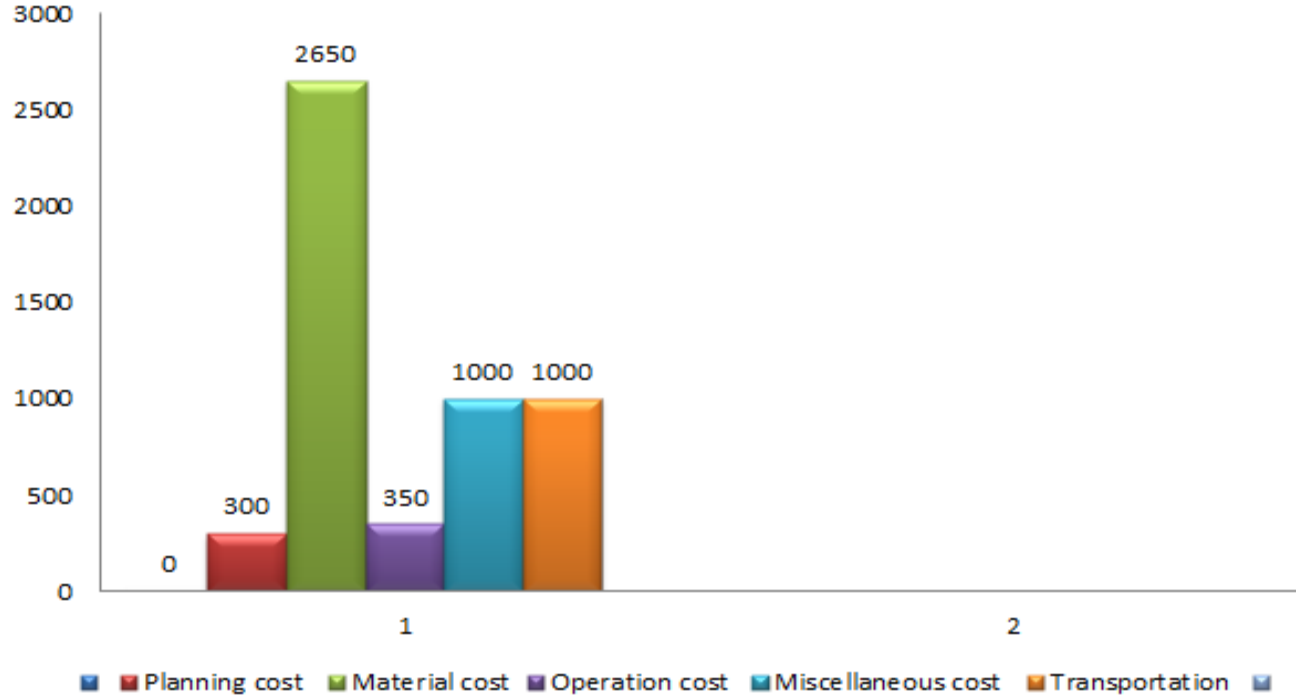


Figure 6: Economic feasibility chart

Table 2: Cost Estimation

Cost Type	Cost details		Total estimated cost in birr
Planning cost	Estimated cost for planning the project		300
Material cost	*Raspberry Pi*	1x	850
	3D Print	1x	1000
	Microphone Module	1x	150
	Chargeable battery	1x	100
	Voice Bonnet	1x	100
	Micro USB cable	1x	50
	Micro SD Card	1x	200
	Mini Speaker	1x	100
	Papers	100	100

Operation cost	Binding and printing	350
Miscellaneous cost	Internet and Server	1000
Transportation	Estimated cost for transportation	1000
Total cost	**The total sum of costs**	**5,300.00 birr**

Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. The group members are well educated undergraduates in Saint Mary's University with potential that is believed to complete this project. The team is composed of individuals with their corresponding role, responsibility and expertise with fair to medium level of programming experience that developed through continuous project undertaking. The project requires computers, Microprocessors, phones, audio recording device, libraries and software that the team can access. The software and hardware we will be using is fairly investigated and studied to fit specifically the project.

Operational Feasibility

The user will be able to understand the built model of the interface in the auditory sense. The user will be able to navigate the device in an easier way. User will be able to make use of own commands to perform tasks with devices that are mostly available like mobile phones. The UX (User Experience) is considered thoroughly while operating the system which will simplify the use of the application of the system. The user will be at ease when using the application because a digital assistant will guide that person all the way through.

Schedule Feasibility

Delivering with the planned-out schedule is one of the priority goals in Software Project Management. The ability to visualize and internalize the procedural and/or parallel progression of tasks' alignment with their time of execution and completion is what fine adjustment of the trajectory of a project initiation. We have used the Gantt chart, which we have found to be best fitting for this project. The tasks that are needed to be done can only progress in timely sequence. Despite challenges, we expect to deliver roughly within the timeline we have set now. The schedule for this project is different from the common trend because here implementation begins as soon as the project is initiated. Scheduling and more are explained in this document below.

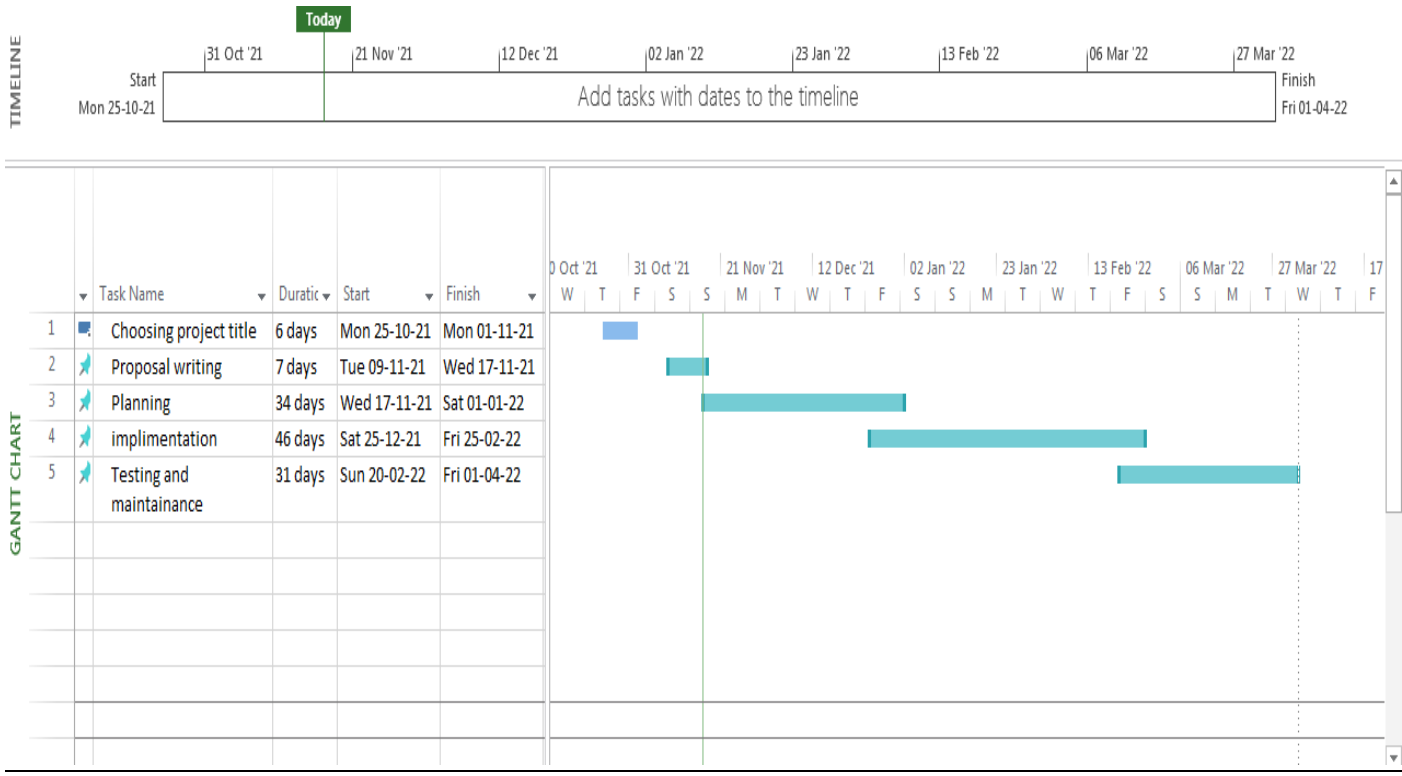


Figure 7: Gantt chart

Functional Requirements

Functional requirement is a description of the service that the system must offer. It describes a software system or its component. These requirements will assure that the system will correctly and reliably perform its intended functionalities. The project provides these functions as a requirement.

Functional requirement 1 (FR1): Register device

The admin will register the device serial number on the system and track down the device performance and activity.

Functional requirement 2 (FR2): Log event

The system will track every event or request from the user.

Functional requirement 3 (FR3): Command audio recognition

The system will listen to and recognize the command voices that are pre-trained. The command words are listed below.

Functional requirement 4 (FR4): Social application access

The system will call an external API or scraper to access Telegram and Email functionality from cell phones.

Functional requirement 5 (FR5): Information access

- Weather forecast
- Google search
- News

This functionality will be requested from the API.

Functional requirement 6 (FR6): Phone service

- Phone call
- SMS message

The system will access this service and process the result to response phase (TTS phase).

Functional requirement 7 (FR7): Utility service

- **Manage reminder and alarm:** set wakeup alarm and reminder.
- **Voice over calculator:** Calculate numbers.
- **Getting Date/Time and events:** Request date and time.

Functional requirement 8 (FR8): Synthesize speech

The system synthesizes voice as response that gets its input from-services.

Non-functional requirement

The non-functional requirement specification for Lamba details the quality attribute, a performance attribute, a security attribute, or a general constraint on the proposed system. They are contrasted with functional requirement that defines specific behaviors or functions. The following are some of non-functional requirements of the proposed system:

- **Availability requirement**: The system is available 24/7 to all users who have access to internet connection.
- **Speed requirement**: The system relatively will have fast communication because it uses JRPC for the communication between micro-services and UDP for the communication between Raspberry pi and server.
- **Performance requirement**: The system can handle high load request from hundreds of different users that is sent from microcontroller. The system should respond to any user request as soon as possible and must handle many concurrent requests simultaneously.
- **Security requirement**: Service communication should be highly encrypted so that non user cannot access user's information.
- **Generate report:** The admin generates report from the log (tracked event).
- **Maintainability:** Each error and malfunction of the device that may occur in the system will be handled accordingly in order to reduce the amount of failure. The system will be best in handling errors and report an appropriate error message.
- **Reliability requirement:** The system should be released after testing thoroughly since it is related to sensitivity and money. Situations in which request and response run unexpectedly should be minimal and the system must generate errors and report back to the administrator.
- **User friendliness: U**ser is allowed to communicate with audio speech in own native language.
- **Ban user:** The admin can ban user temporarily or permanently according to certain situations.

Table 3: Operation Controller

			Operation Controller			
Weak	Connectors	Greetings	Break		Continue & Continual	
ሰላም	ከ	ደህና	አይ	በቃኝ	እሺ	ይደገም
ላምባ	የ	አደርሽ	ተዪው	ይበቃኛል	አዎ	ድገሚ
	በ	ዋልሽ	ይቅር	ቻው	እፈልጋለሁ	ድገሚልኝ
	ለ	አመሸሽ	አልፈልግም		ቀጥይ	ድገሚው
	ኛ	ነሽ				
27						

Table

Table 4: Common Commands

Common Command				
ነው (All)	አዲስ (W/N/R)	ሆነ (Ca/D)	ዘርዝሪሊኝ	መልዕክት (T/E/M)
ላኪልኝ	ያዢልኝ	ቀኑ (W/D)	ዘርዝሪ	ያዢ
ላኪ	አለ (W/N/R)	ላይ (Di/T/E/M)	አንብቢ (N/B/M/E/T)	አስቀምጪ (Co/R/M)
ላኪው	አለኝ (W/N/R)	ተልኮልኛል	ስንት (O/G/Ca)	የዛሬው
አንብቢልኝ (N/B/M/E/T)	ንገሪኝ	ነበረኝ		የዛሬ
24				

Table 5: Calculator, Dates Command

Calculator		Dates		
ሲደመር	ደምሪ	ሰኞ	አርብ	ትናት
ሲቀነስ	ቀንሺ	ማክሰኞ	ቅዳሜ	ዛሬ
ሲካፈል	አካፍዪ	ዕሮብ	እሁድ	
ሲባዛ	አባዢ	ሀሙስ	ነገ	
18				

Table 6: Voice Settings, Contact, Alarm, and Calendar

Voice Settings		Contact	Alarm	Calendar
ድምጽ	ዝቅ	ስልክ	ማንቂያ	ሰአት
ቀንሺ	አርጊ	ቁጥሩ	ደውል	ንገሪኝ
አጥፊ	ጨምሪልኝ	ቁጥር	ሙይልኝ	ቀን
			ሙይ	ቀኑን
16				

Table 7: Numbers

Numbers				
ዜሮ	አምስት	አስር	ሃምሳ	መቶ
አንድ	ስድስት	አስራ	ስልሳ	ሺ
ሁለት	ሰባት	ሃያ	ሰባ	ሚሊየን
ሶስት	ስምንት	ሰላሳ	ሰማንያ	ነጥብ
አራት	ዘጠኝ	አርባ	ዘጠና	
23				

Table 8: Google Search, GPS Location, News, Weather

Google Search				GPS Location		News		Weather	
የት	አይነቶች	መች	ምንድን	ነኝ	ያህል	ምን	ውሎ	አየሩ	አለብኝ
ምን	ክፍሎች	ትርጉም		ያለሁበት	ይርቃል	አዲስ	ዜና	ጃኬት	የአየር
ማን	ዝርዝሮች	ትርጉሙ		ቦታ	ከ-እዚህ	ነገር	ነበር	ጃንጥላ	ሁኔታ
እንዴት	ዋጋ	መቼ		ያለሁት		አለ		መያዝ	መልበስ
ለምን	ያህል	ማለት							
38									

Table 9: Dial, TO DO/Reminder/ Telegram/ Email/ Message

Dial	To Do /Reminder	Telegram	Email	Message
ደውይ	ማስታወሻ	ቴሌግራም	ኢሜል	
ደውይልኝ				
ደውይለት				
ደውይላት				
7				

Table 10: Books/ Missed call / Help

Books	Missed Call	Help
መጽሐፍ	ሚስኮል	ትዕዛዝ
ክፍል		
ታሪክ		
ምዕራፍ		
4		

Total	160

After having carefully planned the project, the team started the project implementation phase. In the implementation phase, we put the project plan into action and work to produce the deliverables. Here the steps undertaken to build the deliverable and its components, the limitation and risk we faced and the final deliverable will be discussed and samples will be shown. To make reading and understanding the implementation, we drafted our own sequence of action as follows:

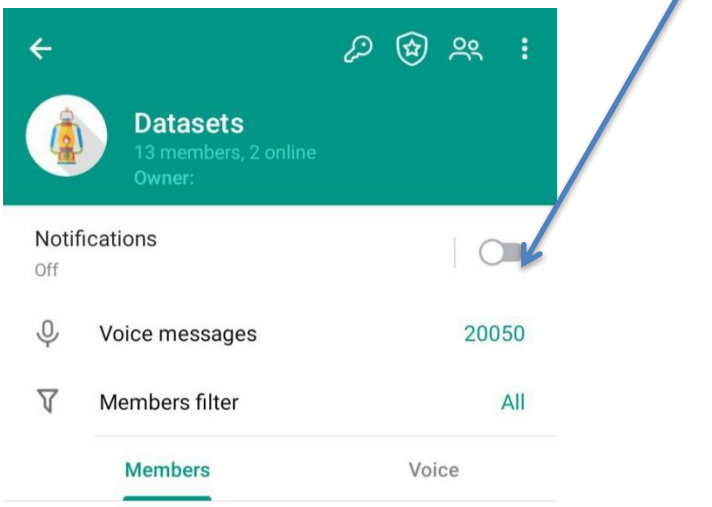
Speech Recognition and Speech to Text

This is the first main phase of implementation where we started to work on and spent most of our time and effort.

Dataset Collection

For Speech Recognition

To collect dataset for the recognition we built LAMBAS_BOT
LAMBAS_BOT is a telegram bot built to facilitate audio data recording and storing. The bot is user intractable, built with pure GO programing language and uses telegram API for its functionality. It is deployed in Linux server that we found from sponsorship outside Ethiopia. We have collected the voice in sub word standard from volunteers in St. Mary University; friends and relatives about **356** people participated. The collection began from Jan/4/2022 till may/28/2022 for about 6 months. We included **159** sub words that are very important in commanding and initiating the right micro services. Through the bot we have collected about: **20050** voices



@LAMBAS_BOT

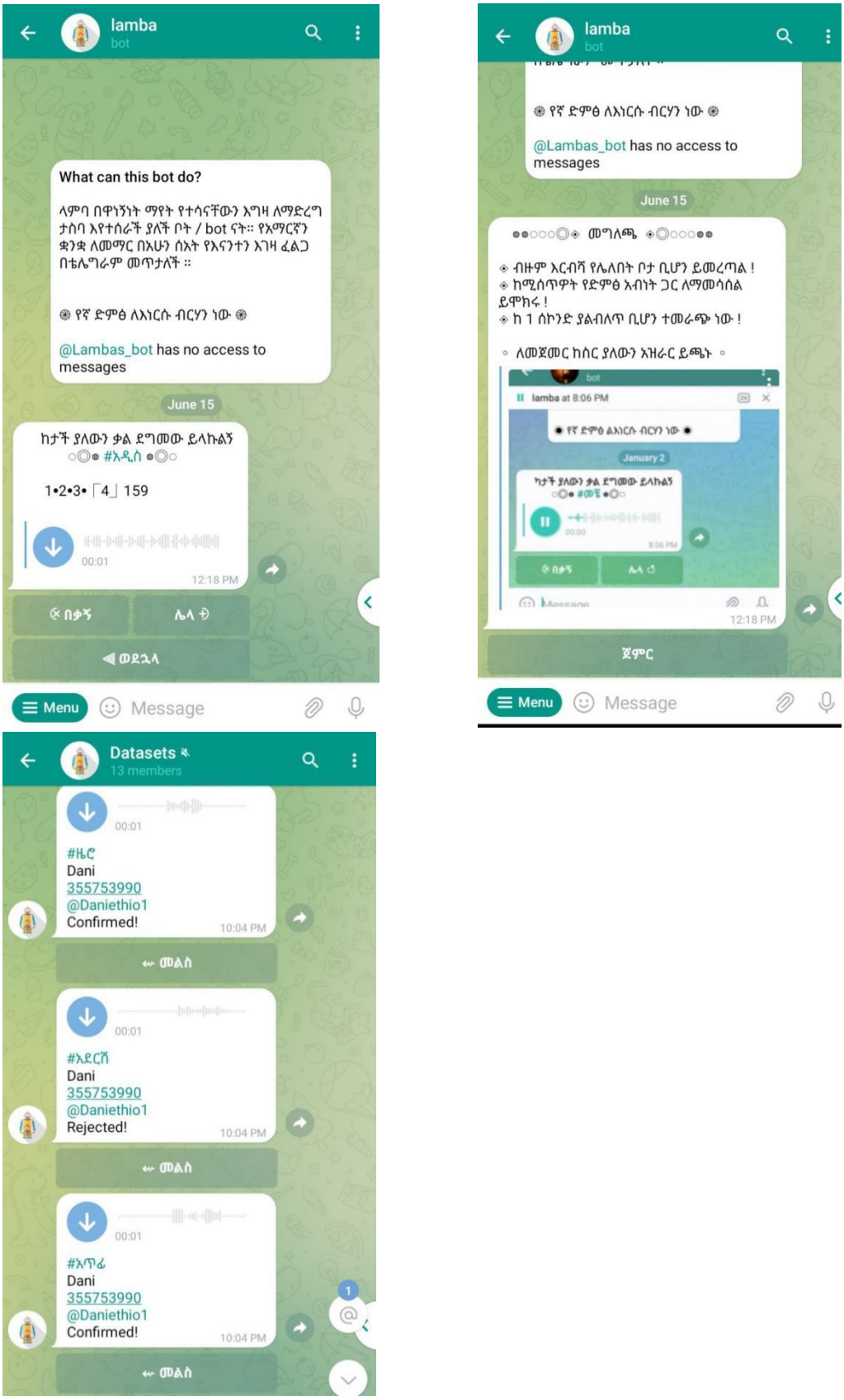


Figure 8: User interface of LAMBAS bot

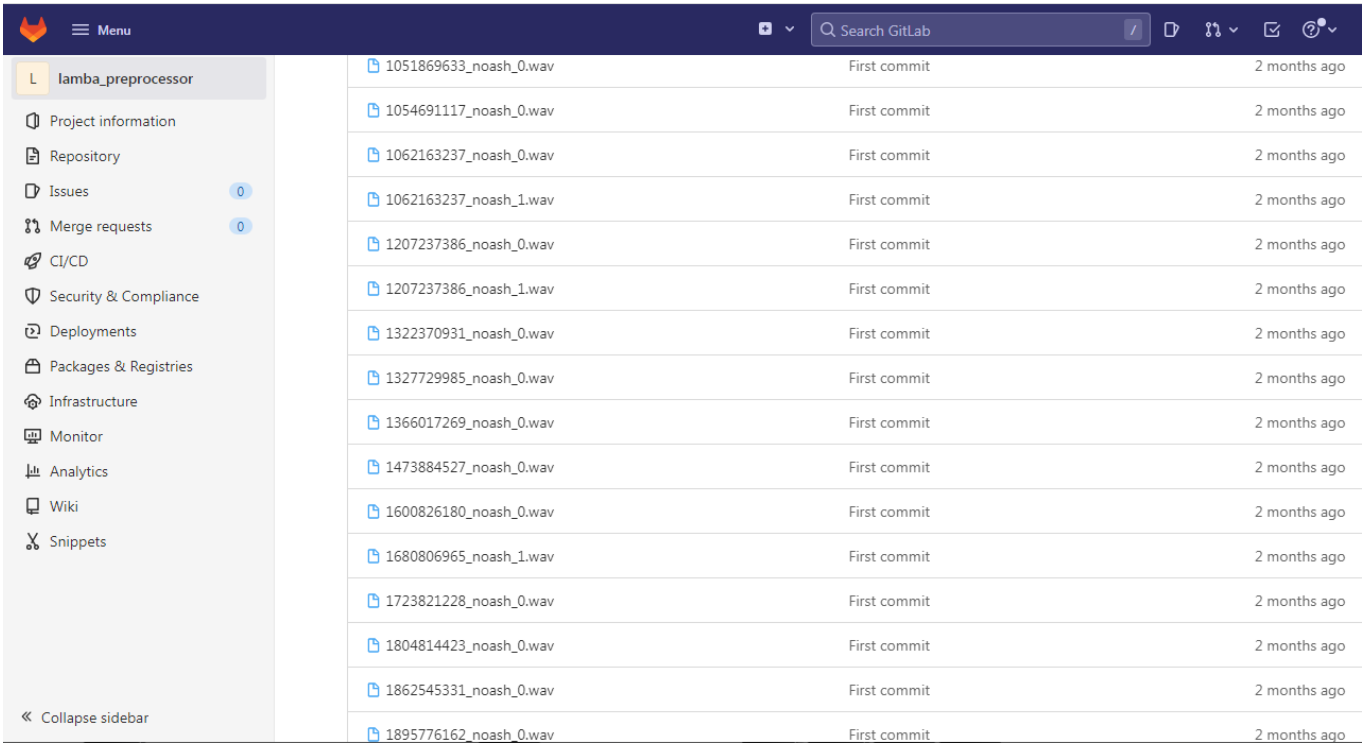


Figure 9: LAMBA ASR audio dataset for Gitlab

For Speech to Text

Collecting dataset for speech synthesis is relatively straightforward since Amharic text data can be found written from different sources and the audio translation is made by the voice of one of our own members.

We prepared about 1800 line of statement and its audio translation.
Tools used at this stage are sound card and condensed microphone.

Our source of Amharic text data

- BBC News Amharic (https://www.bbc.com/amharic)
- The Reporter (https://www.ethiopianreporterjobs.com)
- Amharic Bible (articles)

1100.wav,ዘላለም መንግሥቱ / ዕዝራ ስነ ጽሑፍ፣ አርምኮ የነዳጅ አምራች የሚሳኤል ጥቃት ደረሰበት።
1101.wav,በጅዳ ከተማ ሚገኘው አርምኮ የተሰኘው የሳዑዲ አረቢያ ነዳጅ አምራች ድርጅት ዛሬ በሚሳኤል ጥቃት ተፈፀመበት።
1102.wav,ዓለም አቀፍ የሆኑ የስፖርት ውድድሮች በሱፐር ስፖርት በአማርኛ ቋንቋ መቅረብ ጀመሩ።
1103.wav,በመድረኩ የአማርኛ ይዘት ከሚቀጥለው ዓመት ጀምሮ የግርጌ የትርጉም ጹሑፍን ታክሎባቸው ለዕይታ ይበቃሉ ተብሏል።
1104.wav,ሩሲያ የዩክሬንን የአየር ክልል ለመዝጋት ከሚሞክሩ ኃይል ጋር ጦርነት እንደምትከፍት ገለፀች።
1105.wav,ሀገሪቱ ተጠናቆ ባለባቸው የመጀመሪያ ምዕራፍ ያስቀመጠችው ግቦች እንደተሳኩ የገለጸች ሲሆን ብዙ የዩክሬን ግዛት መቆጣጠሯን አሳውቃለ
1106.wav,በተጨማሪም ኬሮስና ዛፒሮስ የተባለ የዩክሬን ግዛቶች አሁን ላይ በእጇ ላይ መውደቁን አሳውቃለች።
1107.wav,በዚህ ሳምንት የልቦና ውቅር ነጻ የሃሳብ መድረኮች መድረካችንን ስለ ስሜት ብስለት እንነጋገራለን።
1108.wav,ስሜትን የሚረብሹ አስቸጋሪ ሁኔታዎች እንዴት በብስለት ማለፍ ይቻላል የሚለው ላይ ይበልጥ እናተኩራለን።
1109.wav,ቅዳሜ መጋቢት 17 ከቀኑ 9 ሰአት ላይ በኢቲቪ ስቱዲዮ በመገኘት እንድትሳተፉ በአክብሮት ተጋብዛችኋል።
1110.wav,ኢትዮጵያዊቷ በየዓመቱ የሚካሄደው ዓለም አቀፍ ፒአይቲኤች አዲስ የስራ ፈጠራ ውድድር በትላንት እለት ማሸነፍዋ ይፋ ሆኗል።
1111.wav,ማክ ዲጂታል ቤተ መጻሕፍት ያለውና በውስጡ አሜሪካ ተኮር ጽሑፎችን፣ መጻሕፍት አዳዲስ ቴክኖሎጂዎችን መያዙ ታውቋል።
1112.wav,ከዩኒቨርሲቲ ማኅበራዊ ገጽ የተገኘው መረጃ እንደሚጠቁመው ማዕከሉ ከተመረቀበት ዕለት ጀምሮ ለማኅበረሰቡ አገልግሎት ክፍት ይሆናል።
1113.wav,ዩኒቨርሲቲ የምድር ሳተላይት መቆጣጠሪያ ጣቢያ ከመገንባት ጀምሮ በራሱዋ ተማሪዎች መምህራን የሳተላይት ግንባታ እያከናወነ መሆኑ
1114.wav,ባለሙያዎቹ እንደሚሉት ዊንዶ 7 ግለሰባዊ የኮምፒዩተር አጠቃቀም ሥርዓት ሲሆን «ዊንዶ» የተባለው የከወና ሥርዓት አካል ነው።
1115.wav,ኢትዮጵያ ሳተላይት ለማልማትና ለማምጠቅ በ2009 ዓ.ም. ከቻይና መንግሥት ጋር ከስምምነት መድረሷ ይታወቃል።
1116.wav,ሁሉንም ያስተሳሰረችው በፀሐይ ውስጥ ለተገኙ ከዋክብትና ፕላኔቶች ስያሜ መዋላቸው ነው።
1117.wav,ከነዚህም መካከል በአሜሪካ ደቡባዊ ምዕራብ የሚገኘው አላስካ ወንዝ ከተመረጡት አንዱ ነው ሆኗል።
1118.wav,የዓመቱ 14 ወራት በእንግሊዝኛ ጥር፡ የካቲት፡ መጋቢት፡ ኤፕሪል፡ ግንቦት፡ ሰኔ፡ ሐምሌ፡ ነሐሴ፡ መስከረም፡ ጥቅምት፡ ኅዳር፡ ታኅሣሥ።
1119.wav,ነጭ-ሰማያዊ-ቀይ በአግድም ባለ መስመር ያለው የሩስያ ባንዲራ በ17ኛው ክፍለ ዘመን ሲሆን በኔዘርላንድ ባለሶስት ቀለም ተመስሏል።
1120.wav,በሩሲያ ሞዴል ላይ ልዩነት ያላቸው ባንዲራዎች የክሮኤሺያ እና የስርቢያ ባንዲራዎች ያካትታሉ, ሁሉቱም ቀይ-ሰማያዊ-ነጭ አግድም የገ
1121.wav,ባላቸው ከፍተኛ በይነ ዲሲፕሊናዊ አወቃቀራቸው ምክንያት፣ የተፈጥሮ ሳይንስ ዲግሪያቸውን በተለይ ሳይንስ ለመማር ለሚፈልጉ ተማሪዎች፣ ለማድረግ ላይ ይገኛል።
1122.wav,የተፈጥሮ ሳይንስ ምሩቃን ሰፋ ያለ የአጠቃላይ ሳይንሳዊ እውቀት፣ እንዲሁም በመገናኛ፣ በቁጥር እና በኢንፎርሜሽን ቴክኖሎጂ ችሎታዎች
1123.wav,አንዳንድ የተፈጥሮ ሳይንሶች መርሃ ግብሮች ለጤና ህዝብ አስተዋፅዖ ለማድረግ አለምአቀፍ እና አካባቢያዊ ጉዳዮችን በመፍታት ላይ ያተኩ
1124.wav,ይደውሉልን : 0911156257 ከታች ያለው አገናኝ በመጫን የቴሌግራም አባል ይሁኑ ቦሌ ወሎሰፈር ህንፃ 2ኛ
1125.wav,የምሽት ሰማይ በአስደናቂ የስነ ፈለክ ክስተቶች የተሞላ ነው።
1126.wav,በአንዳንድ የኤፒአይ ምላሾች ውስጥ የተወሰነ የጊዜ እና የቀን መረጃ ብዙ አሃዞችን ያሏቸው ረጅም ቁጥሮችን ሆነው ይታያሉ።

Figure 10: Amharic text dataset for TTS in CSV format

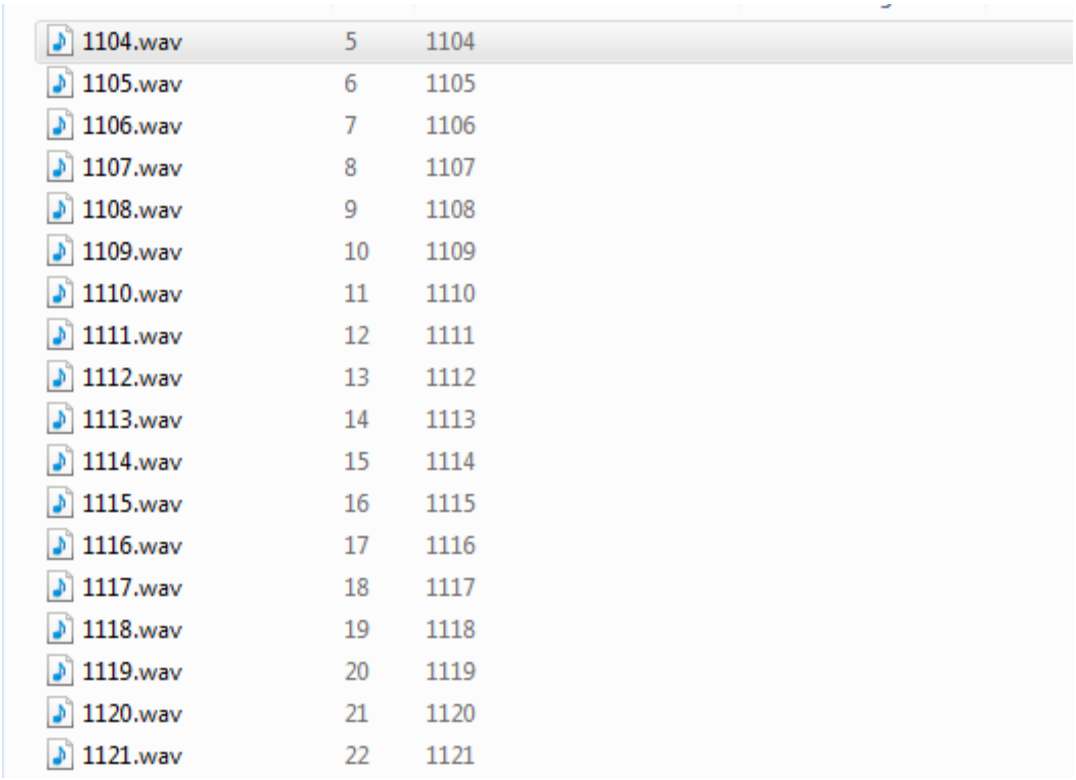


Figure 11: Amharic text datasets voice translation in .WAV format

Manually Prepare and Pre-process the Data

After the textual and audio dataset are collected, we had to repair the raw data to make it suitable for a building and training Machine Learning model. We transformed the raw data into an understandable and readable format.

Manually processing ASR and TTS:-

- Prepared CSV file for the textual data for ASR
- Prepared .WAV file translation for the textual data for ASR
- Prepared CSV file for the textual data for TTS
- Manually cut the TTS voice translation into full length voice statement

Augmenting ASR voice with code

Sample rate

- Convert all the voice dataset to **16 kHz** (audio can be in different sample rate when recorded 8 kHz, 16 kHz, 22 kHz and 44 kHz).

No of channel

- Make the number of channels for all voice =1(mono)

Sample width

- Make sample width 2 Byte to store the framed WAV file

Processing TTS with code

Sample rate – must be **22 kHz**

Tools used: -

- Python script
- FL Studio
- Audacity

```python
import collections
import contextlib
import sys
import wave
import webrtcvad

def read_wave(path):
    """Reads a .wav file.
    Takes the path, and returns (PCM audio data, sample rate).
    """
    try:
        with contextlib.closing(wave.open(path, 'rb')) as wf:
            num_channels = wf.getnchannels()
            assert num_channels == 1
            sample_width = wf.getsampwidth()
            assert sample_width == 2
            sample_rate = wf.getframerate()
            assert sample_rate in (8000,16000, 32000, 48000)
            pcm_data = wf.readframes(wf.getnframes())
            return pcm_data, sample_rate
    except:
        print(path," file couldn't augmented")
        return 0,0


def write_wave(path, audio, sample_rate):
    """Writes a .wav file.
    Takes path, PCM audio data, and sample rate.
    """
    with contextlib.closing(wave.open(path, 'wb')) as wf:
        wf.setnchannels(1)
        wf.setsampwidth(2)
        wf.setframerate(sample_rate)
        wf.writeframes(audio)


class Frame(object):
    """Represents a "frame" of audio data."""
    def __init__(self, bytes, timestamp, duration):
        self.bytes = bytes
        self.timestamp = timestamp
        self.duration = duration
```

Figure 12: Augmentation sample code

(https://gitlab.com/Lambas/lamba_preprocessor/-/blob/main/tools/augmenter.py)

Training

Setup our environment

Hardware environment

MacBook Air
HP

Software environment

Google Collaboratory (Colab)
Jupiter Notebook
Anaconda

Getting started with ASR

Import necessary modules and dependencies

```
import os
import pathlib

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf
from tensorflow import lite

from tensorflow.keras import layers
from tensorflow.keras import models
from IPython import display

# Set the seed value for experiment reproducibility.
seed = 42
tf.random.set_seed(seed)
np.random.seed(seed)
model_filename = 'lamba_model.h5'

# Parameters// behuala modelu ena tf lit bezi yekemet lemalet new
keras_model_filename = 'lamba_model.h5'
tflite_filename = 'lamba_model.tflite'
```

Import the Lamba Commands dataset:

Download and extract the Lamba_commands.zip file containing the smaller Speech Commands datasets

```
DATASET_PATH = 'data/mini_speech_commands'
data_dir = pathlib.Path(DATASET_PATH)
if not data_dir.exists():
  tf.keras.utils.get_file(
      'mini_speech_commands.zip',
      origin="http://storage.googleapis.com/download.tensorflow.org/data/mini_speech_commands.zip",
      extract=True,
      cache_dir='.', cache_subdir='data')
```

Extract the audio clips into a list called filenames, and shuffle it:

```
filenames = tf.io.gfile.glob(str(data_dir) + '/*/*')
filenames = tf.random.shuffle(filenames)
num_samples = len(filenames)
```

Split filenames into training, validation and test sets using a 80:10:10 ratio, respectively:

```
train_files = filenames[:879]
val_files = filenames[879: 879 + 110]
```

```
test_files = filenames[-110:]
```

Define a function that preprocesses the dataset's raw WAV audio files into audio tensors:

```
def decode_audio(audio_binary):
  audio, _ = tf.audio.decode_wav(contents=audio_binary)
  return tf.squeeze(audio, axis=-1)
```

Define a function that preprocesses the dataset's raw WAV audio files into audio tensors:

```
def decode_audio(audio_binary):
  audio, _ = tf.audio.decode_wav(contents=audio_binary)
  return tf.squeeze(audio, axis=-1)
```

Define a function that creates labels using the parent directories for each file:

- Split the file paths into tf.RaggedTensors (tensors with ragged dimensions—with slices that may have different lengths).

```
def get_label(file_path):
  parts = tf.strings.split(
      input=file_path,
      sep=os.path.sep)
  return parts[-2]
```

Define another helper function—get_waveform_and_label—that puts it all together:

- The input is the WAV audio filename.
- The output is a tuple containing the audio and label tensors ready for supervised learning.

```
def get_waveform_and_label(file_path):
  label = get_label(file_path)
  audio_binary = tf.io.read_file(file_path)
  waveform = decode_audio(audio_binary)
  return waveform, label
```

Build the training set to extract the audio-label pairs:

Create a tf.data.Dataset with Dataset.from_tensor_slices and Dataset.map,
using get_waveform_and_label defined earlier.

```
AUTOTUNE = tf.data.AUTOTUNE
files_ds = tf.data.Dataset.from_tensor_slices(train_files)
waveform_ds = files_ds.map(
    map_func=get_waveform_and_label,
    num_parallel_calls=AUTOTUNE)
```

Convert waveforms to spectrograms

The waveforms in the dataset are represented in the time domain. Next, transform the waveforms from the time-domain signals into the time-frequency-domain signals by

computing the short-time Fourier transform (STFT) to convert the waveforms as spectrograms, which show frequency changes over time and can be represented as 2D images. Then feed the spectrogram images into your neural network to train the model.

```
def get_spectrogram(waveform):
  input_len = 16000
  waveform = waveform[:input_len]
  zero_padding = tf.zeros( [16000] - tf.shape(waveform),dtype=tf.float32)
  waveform = tf.cast(waveform, dtype=tf.float32)
  equal_length = tf.concat([waveform, zero_padding], 0)
            spectrogram = tf.signal.stft(equal_length, frame_length=255, frame_step=128)
  spectrogram = tf.abs(spectrogram)
  spectrogram = spectrogram[..., tf.newaxis]
   return spectrogram
```

Start exploring the data. Print the shapes of one example's tensorized waveform and the corresponding spectrogram, and play the original audio:

```
for waveform, label in waveform_ds.take(1):
  label = label.numpy().decode('utf-8')
  spectrogram = get_spectrogram(waveform)

print('Label:', label)
print('Waveform shape:', waveform.shape)
print('Spectrogram shape:', spectrogram.shape)
print('Audio playback')
display.display(display.Audio(waveform, rate=16000))
```

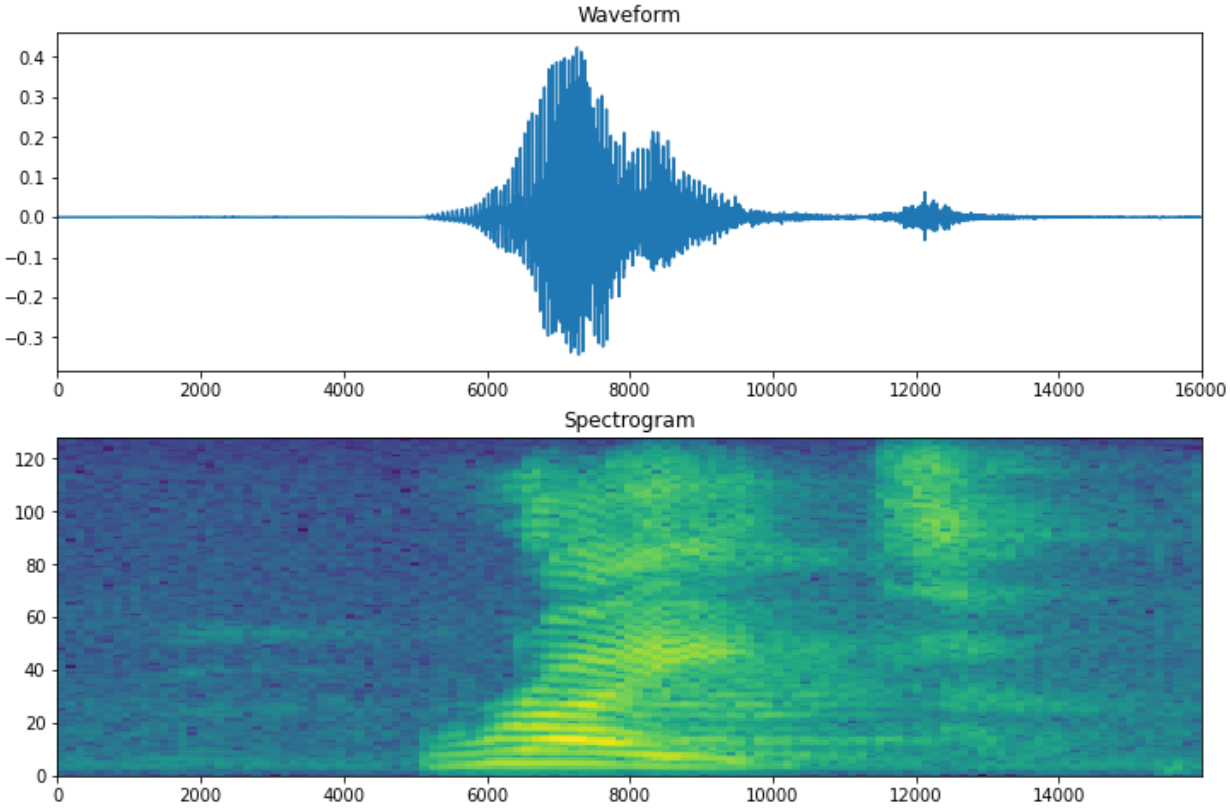
```
============================================================
Label: lamba
Waveform shape: (16000,)
Spectrogram shape: (124, 129, 1)
Audio playback
```

Define a function for displaying a spectrogram:

```
def plot_spectrogram(spectrogram, ax):
  if len(spectrogram.shape) > 2:
    assert len(spectrogram.shape) == 3
    spectrogram = np.squeeze(spectrogram, axis=-1)
      log_spec = np.log(spectrogram.T + np.finfo(float).eps)
  height = log_spec.shape[0]
  width = log_spec.shape[1]
  X = np.linspace(0, np.size(spectrogram), num=width, dtype=int)
  Y = range(height)
  ax.pcolormesh(X, Y, log_spec)
```

Plot the example's waveform over time and the corresponding spectrogram (frequencies over time):

```
fig, axes = plt.subplots(2, figsize=(12, 8))
timescale = np.arange(waveform.shape[0])
axes[0].plot(timescale, waveform.numpy())
axes[0].set_title('Waveform')
axes[0].set_xlim([0, 16000])
plot_spectrogram(spectrogram.numpy(), axes[1])
axes[1].set_title('Spectrogram')
plt.show()
```



Now, define a function that transforms the waveform dataset into spectrograms and their corresponding labels as integer IDs:

```
def get_spectrogram_and_label_id(audio, label):
  spectrogram = get_spectrogram(audio)
  label_id = tf.argmax(label == commands)
    return spectrogram, label_id
```

Map get_spectrogram_and_label_id across the dataset's elements with Dataset.map:

```
spectrogram_ds = waveform_ds.map(
  map_func=get_spectrogram_and_label_id,
  num_parallel_calls=AUTOTUNE)
```

Build and train the model

We repeated the training set preprocessing on the validation and test sets

```
def preprocess_dataset(files):
  files_ds = tf.data.Dataset.from_tensor_slices(files)
```

```
  output_ds = files_ds.map(
      map_func=get_waveform_and_label,
      num_parallel_calls=AUTOTUNE)
  output_ds = output_ds.map(
      map_func=get_spectrogram_and_label_id,
      num_parallel_calls=AUTOTUNE)
    return output_ds
train_ds = spectrogram_ds
val_ds = preprocess_dataset(val_files)
test_ds = preprocess_dataset(test_files)
```

Batch the training and validation sets for model training:

```
batch_size = 64  # 1098/64=17.15
train_ds = train_ds.batch(batch_size)
val_ds = val_ds.batch(batch_size)
train_ds = train_ds.cache().prefetch(AUTOTUNE)
val_ds = val_ds.cache().prefetch(AUTOTUNE)
```

tf.keras.Sequential model uses the following Keras preprocessing layers:

- tf.keras.layers.Resizing: to downsample the input to enable the model to train faster.
- tf.keras.layers.Normalization: to normalize each pixel in the image based on its mean and standard deviation.

```
for spectrogram, _ in spectrogram_ds.take(1):
  input_shape = spectrogram.shape
print('Input shape:', input_shape)
num_labels = len(commands)
norm_layer = layers.Normalization()
norm_layer.adapt(data=spectrogram_ds.map(map_func=lambda spec, label: spec))

model = models.Sequential([
    layers.Input(shape=input_shape),
    layers.Resizing(32, 32),
    norm_layer,
    layers.Conv2D(32, 3, activation='relu'),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_labels), ])
```

Configure the Keras model with the Adam optimizer and the cross-entropy loss:

```
model.compile
    optimizer=tf.keras.optimizers.Adam(),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=['accuracy'],
```

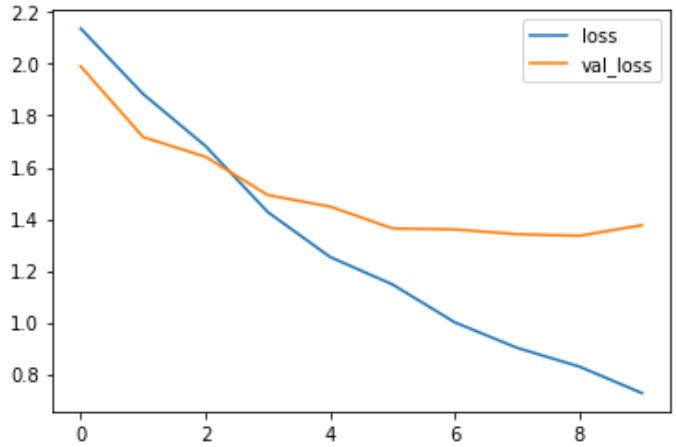
Train the model over 10 epochs for demonstration purposes:

```
EPOCHS = 10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=2),
```

```
Epoch 1/10
14/14 [==============================] - 6s 342ms/step - loss: 2.1369 - accuracy: 0.2207 - val_loss: 1.9905 - val_accuracy: 0.3636
Epoch 2/10
14/14 [==============================] - 3s 195ms/step - loss: 1.8836 - accuracy: 0.3879 - val_loss: 1.7170 - val_accuracy: 0.4909
Epoch 3/10
14/14 [==============================] - 3s 196ms/step - loss: 1.6826 - accuracy: 0.4357 - val_loss: 1.6411 - val_accuracy: 0.4273
Epoch 4/10
14/14 [==============================] - 3s 196ms/step - loss: 1.4269 - accuracy: 0.5188 - val_loss: 1.4928 - val_accuracy: 0.4364
Epoch 5/10
14/14 [==============================] - 3s 194ms/step - loss: 1.2532 - accuracy: 0.5688 - val_loss: 1.4480 - val_accuracy: 0.4545
Epoch 6/10
14/14 [==============================] - 3s 194ms/step - loss: 1.1471 - accuracy: 0.5950 - val_loss: 1.3639 - val_accuracy: 0.5091
Epoch 7/10
14/14 [==============================] - 3s 195ms/step - loss: 1.0004 - accuracy: 0.6576 - val_loss: 1.3600 - val_accuracy: 0.5364
Epoch 8/10
14/14 [==============================] - 3s 193ms/step - loss: 0.9019 - accuracy: 0.6894 - val_loss: 1.3414 - val_accuracy: 0.5182
Epoch 9/10
14/14 [==============================] - 3s 194ms/step - loss: 0.8293 - accuracy: 0.7122 - val_loss: 1.3355 - val_accuracy: 0.5455
Epoch 10/10
14/14 [==============================] - 3s 194ms/step - loss: 0.7271 - accuracy: 0.7600 - val_loss: 1.3764 - val_accuracy: 0.5000
```

Let's plot the training and validation loss curves to check how the model has improved during training:

```
metrics = history.history
plt.plot(history.epoch, metrics['loss'], metrics['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.show()
```



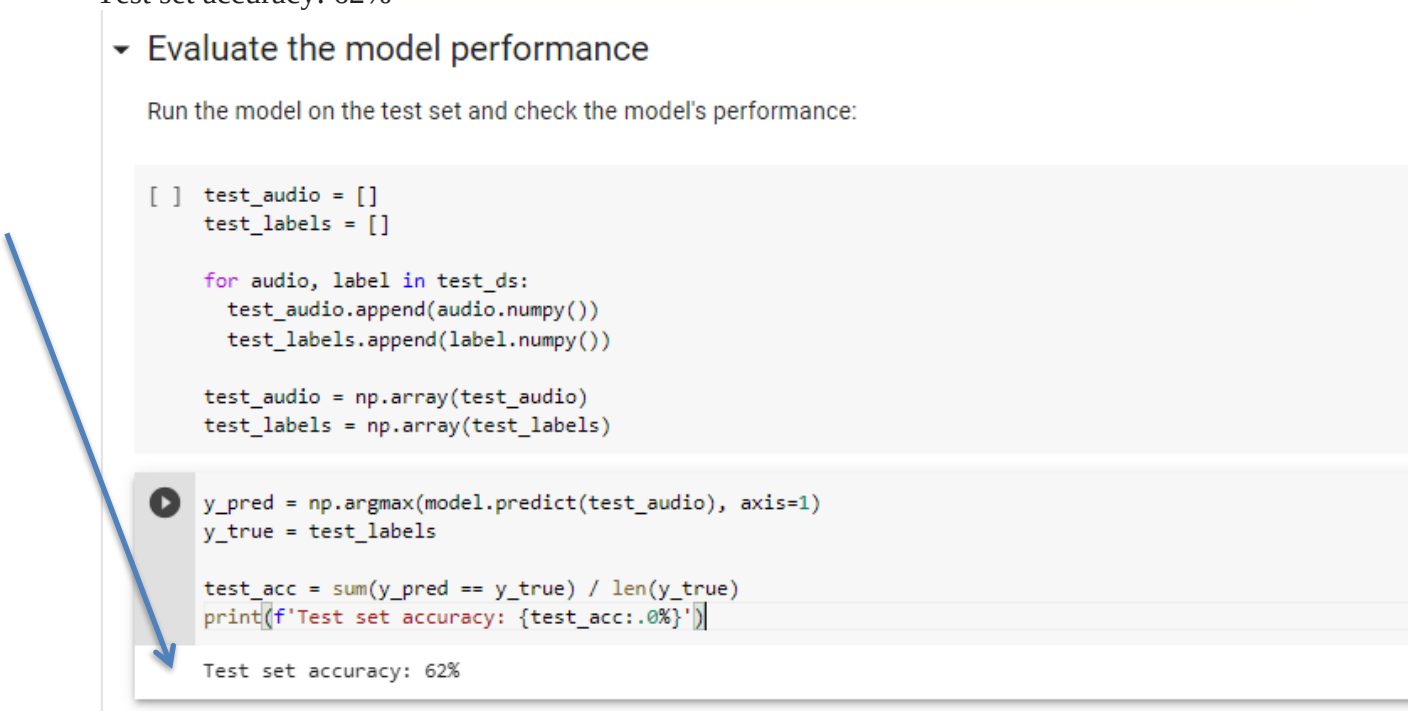
Evaluating the model performance

Run the model on the test set and check the model's performance:

```
test_audio = []
test_labels = []

for audio, label in test_ds:
  test_audio.append(audio.numpy())
  test_labels.append(label.numpy())
```

```
test_audio = np.array(test_audio)
test_labels = np.array(test_labels)


y_pred = np.argmax(model.predict(test_audio), axis=1)
y_true = test_labels

test_acc = sum(y_pred == y_true) / len(y_true)
print(f'Test set accuracy: {test_acc:.0%}')
```

Test set accuracy: 62%



Save the model as a File

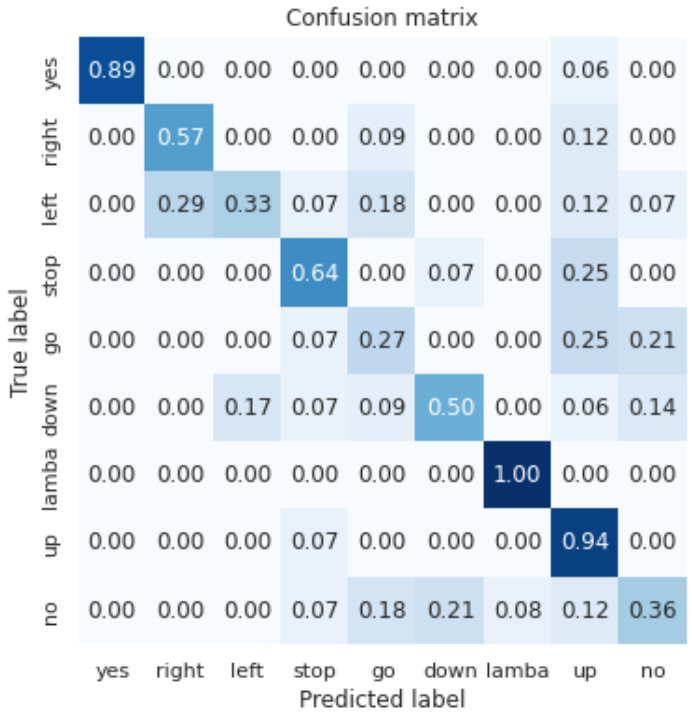
models.save_model(model, model_filename)

Display a confusion matrix

Use a confusion matrix to check how well the model did classifying each of the commands in the test set:

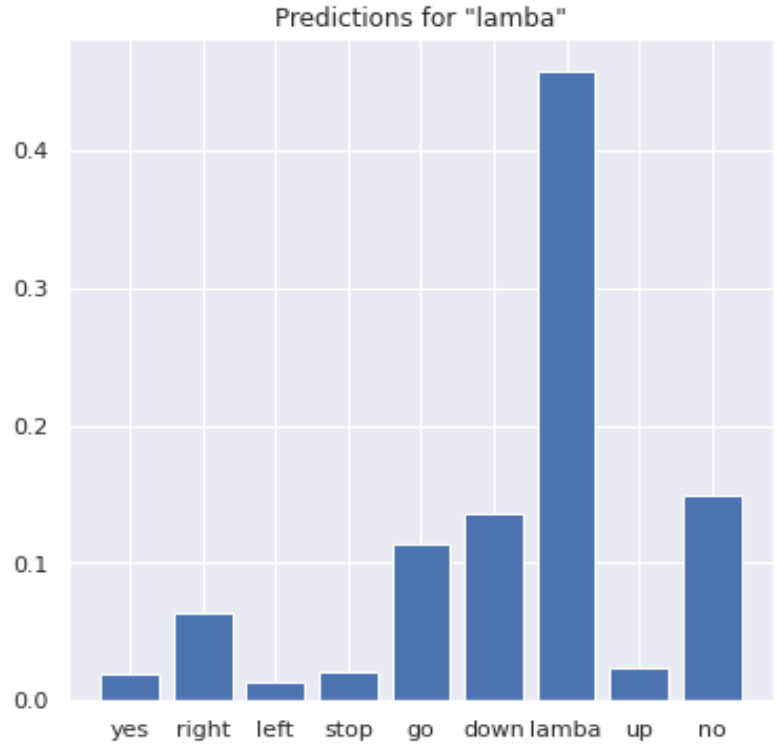
```
def show_confusion_matrix(confusion, test_labels):
  """Compute confusion matrix and normalize."""
  confusion_normalized = confusion.astype("float") / confusion.sum(axis=1)
  sns.set(rc = {'figure.figsize':(6,6)})
  sns.heatmap(
    confusion_normalized, xticklabels=test_labels, yticklabels=test_labels,
    cmap='Blues', annot=True, fmt='.2f', square=True, cbar=False)
  plt.title("Confusion matrix")
  plt.ylabel("True label")
  plt.xlabel("Predicted label")
```

```
confusion_matrix = tf.math.confusion_matrix(y_true, y_pred)
show_confusion_matrix(confusion_matrix.numpy(), commands)
```



Run inference on an audio file

```
sample_file = data_dir/'lamba/356289679827_noash_0.wav'

sample_ds = preprocess_dataset([str(sample_file)])

for spectrogram, label in sample_ds.batch(1):
  prediction = model(spectrogram)
  plt.bar(commands, tf.nn.softmax(prediction[0]))
  plt.title(f'Predictions for "{commands[label[0]]}"')
  plt.show()
```



Convert model to TF Lite model

```
model = models.load_model(keras_model_filename)
converter = lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
open(tflite_filename, 'wb').write(tflite_model)
```

```
def cnn_tflite_compatible_model():

    waveform = layers.Input(batch_shape=(1, params.SAMPLE_RATE))
    logMelgram = features_tflite_lib.waveform_to_log_mel_spectrogram(
        waveform, params)

    net = layers.Reshape(
        (params.FRAMES, params.MEL_BINS, 1))(logMelgram)

    net = layers.Conv2D(16, 3, activation='relu',
                        kernel_initializer='he_normal')(net)
    net = layers.Conv2D(32, 2, activation='relu',
                        kernel_initializer='he_normal')(net)
    net = layers.MaxPooling2D()(net)
    net = layers.Conv2D(64, 2, activation='relu',
                        kernel_initializer='he_normal')(net)

    net = layers.MaxPooling2D()(net)
    net = layers.Flatten()(net)
    net = layers.Dense(128, activation='relu',
                       kernel_initializer='he_normal')(net)
    net = layers.Dropout(params.DROPOUT)(net)
    net = layers.Dense(len(params.WORDS), activation='softmax')(net)
```

```
    model = Model(name="TFLITE_COMPATIBLE_CNN",
                  inputs=waveform, outputs=net)
    return model
```

Load the model and weights

```
model = cnn_tflite_compatible_model()
model.load_weights(args.input)
```

Convert the model

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.experimental_new_converter = True
tflite_model = converter.convert()
open("{}.tflite".format(args.output), 'wb').write(tflite_model)
```

Conclusion

The system performed well and better than what the team expected when we used pre trained model to further capability of recognition. We optimized the system code so it performed within less time duration.

Recommendations4

- Concentrative– implementation of Real-time TTS: Lamba uses concatenative approach for speech synthesis but must change to more flexible and natural way of synthesis - the TTS
- ASR – Complete ASR from scratch with Character based. Our speech recognition is sub-word level and it must get into character level
- User real time training, user voice authentication
- Hardware - Raspberry pi with camera module, object detection, Smart cane with sonic sensor system for localization
- Real-time learning and continuous development
- Web-API - to be supported for various developments, for commercial product 127

References

Programming Collective Intelligence: Building Smart Web 2.0 Applications Collective

Intelligence Author – Toby Segaran Edition – First

Tom M Mitchell, Machine learning(Latest Edition)

Software Engineering (2009), ―A Practitioner‘s Approach‖, Seventh Edition

Grady Booch, Ivar Jacobson and James Rumbaugh, User Guide for UML

https://www.techtarget.com/searchapparchitecture/tip/Pros-and-cons-of-monolithic-vsmicroservices-architecture

https://www.qsstechnosoft.com/how-does-amazon-alexa-works

https://www.youtube.com/watch?v=5J72iMSQmy8

https://github.com/TensorSpeech/TensorFlowTTS

https://youtu.be/0fn7pj7Dutc

https://github.com/TensorSpeech/TensorFlowASR#corpus-sources-and-pretrained-models

https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition

Alan, S. (1988). "The People Factor in Competitiveness" Address Presented at the University of Club of Chicago.