Autonomous Intelligence System



(Winter Semester: 2020-2021)

**Speech Control of Autonomous Robots**

### Written by:

### Lakshmi Mounika Kolisetty (1324400)

### Raghavendra Yarlagadda (1324413)

### 

Under the Esteemed Guidance of

Prof. **Dr. Peter Nauth**

##### Frankfurt University of Applied Sciences

Table of Contents

[Table of Figures 3](#_bookmark0)

[Abstract 4](#_bookmark1)

[Introduction 5](#_bookmark2)

Automatic Speech Recognition 6

Mel frequency cepstral coefficients 7

Recurrent neural networks 7

Backpropagation through time 9

Long short-term memory (LSTM) 11

ROS middleware and gazebo 12

Implementation 12

Results and discussion 16

Conclusion 17

Acknowledgement 17

References 17

# Table of Figures

##### [Figure 1: Block diagram of MFCC processor 7](#_bookmark4)

##### [Figure 2: RNN vs Feed-Forward Neural Networks 8](#_bookmark6)

##### [Figure 3: Forward and backward propagation in a neural network 10](#_bookmark8)

##### [Figure 4 : RNN with its three gates 11](#_bookmark9)

##### Figure 5: GUI Screenshot 13

##### [Figure 5 : Code Snippet of Extraction of MFCC Vectors](#_bookmark11) 14

##### [Figure 6 : Code Snippet of Model Architecture](#_bookmark12) 15

##### [Figure 7 : Gazebo Map](#_bookmark13) 16

# Abstract

This project goal is to design a moving robot which takes a voice command given by a human and performs the action accordingly. The speech recognition framework refers to a system where a person can talk to a computer through a microphone. The computer converts the words into text or commands to perform computer functions. The intelligent speech recognition system makes spoken instructions understandable by the robot. The speech-recognition system is trained to recognize given commands and the programmed robot navigates through the speech commands based on the instructions. The results prove that the proposed robot is capable of understanding the meaning of speech commands given as a recorded input. This robot simulation has been carried in ROS middleware and the map has been designed in Gazebo.

# Introduction

Speech recognition technology has started to transform the way we live and work and has become one of the primary means for people to communicate with mobile devices (e.g., Siri, Google Now, and Cortana). The arrival of this new trend is indicative to substantial improvement in a number of areas. Today speech technologies are widely available for a limited but fascinating range of tasks. These technologies allow devices to respond precisely and effectively to people's voices and provide relevant and valuable services. It is mainly based on the transformation of sound into text and commands. It is a mechanism by which a device associates an audio signal to some form of abstract speech. A speech recognition system is necessary to perform this task. This recognition technique depends also on many parameters -Speaking Mode, Speaking Style, Speaker Enrolment, Size of the Vocabulary, Language Model, Perplexity, Transducer etc. A lot of research work has been done considering the above-mentioned factors to design a better speech recognition system.

A speech recognition system is necessary to perform this task. Several feature extraction methods and pattern matching techniques are used to make better quality speech recognition systems. To design a good speech recognition system certain factors such as vocabulary size, speaker independence, and processing speed. A lot of research work has been done considering the above-mentioned factors to design a better speech recognition system.

Around 1969, a group of Soviet researchers invented an algorithm called Dynamic Time Warping (DTW) [1]. This algorithm is capable of processing speech by splitting it into short frames. The researchers used this algorithm to create a recognizer which can operate on a 200-word vocabulary. Even though this algorithm carried out to be one of the best algorithms for speech recognition this has certain limitations mainly speaker independence which means this is only specific to a particular speaker. By mid-1980’s IBM’s Fred Jelinek’s team invented a voice activated typewriter called Tangora [2], which could handle a 20,000-word vocabulary. The team’s systematic approach was to understand speech in pursuit of utilizing statistical modelling techniques like HMMs. HMM stands for Hidden Markov Model. DTW has been replaced by HMM to become an assertive speech recognition algorithm.

In the early 2000s, speech recognition was still monopolized by conventional methods such as Hidden Markov Models combined with feedforward artificial neural networks [3]. Many considerations of speech recognition have been taken over by a deep learning method called LSTM (Long Short-Term Memory) [4]. This LSTM RNNs avoid the vanishing gradient problem. When training artificial neural networks with gradient-based learning methods and backpropagation, the vanishing gradient problem is encountered in

machine learning. Around 2007, LSTM trained by Connectionist Temporal Classification (CTC) began to dominate conventional speech recognition in certain applications [5]. In 2009, Geoffrey Hinton and his colleagues at Microsoft Research introduced deep feedforward(non-recurrent) networks for acoustic modelling. This innovation was proved to be as the paradigm shift in accuracy (reduced the rate of word error by 30 per cent) since 1979.

In the colonial past of speech recognition, the recurrent networks of artificial neural networks had been

investigated for many years starting from 1980 until a few years into 2000s. To enhance speech recognition accuracy, numerous deep learning models were used.

# Automatic speech recognition

Many effective attempts have been made to construct systems that can analyze, categorize and recognize voice signals. Software produced in diverse spheres of activity, such as health care, government and agriculture, have been used for these activities. Speaker recognition is the ability to receive the speech signal from a software or hardware to identify the speaker in a speaking signal and then the speaker to be recognized [6]. Speaker recognition performs a task similar to that performed by the human brain. This begins with an input into the speaker recognition system. In general, the method is performed in three basic steps: acoustic processing, feature extraction and classification/recognition [7]. Before the extraction of the important attributes in the speech [8] and identification, the speech signal must be treated to eliminate noise. The feature extraction aims at illustrating a voice signal through a predefined number of signal components. This is due to the fact that dealing with all of the information in the acoustic signal is too time-consuming, and part of the information is useless to the identification task [9]. Feature extraction takes place by transforming the speech waveform to a type of parametric representation for further data processing and analysis at a reasonably low data rate. This is commonly referred to as front end signal processing [10]. The processed voice signal is transformed into a compact but logical depiction more discriminatory and trustworthy than the actual signal. The quality of the following characteristics (pattern matching and speech modeling) is substantially affected by the quality of the front end, as the first element in the sequence is the front end [11].

Acceptable classification is therefore generated from high quality characteristics. In modern auto speaker recognition systems (ASR), despite changes in the ambient conditions or speakers were generally discovered

in the feature extraction technique to be reasonably dependable for different situations with the same signal, with retention of the component that characterizes the voice signal information.

Feature extraction normally produces a multidimensional feature vector for each signal [12]. The voice signal of the recognition process is accessible in a wide array of different possibilities, including linear perceptive (PLP), linear prediction coding (LPC) and mel-frequency cepstrum coefficients (MFCC). The most famous and popular MFCC [12]. The most essential aspect of speaker recognition is the extraction of features. Speaking characteristics have an important role to play in separating a speaker from others [13]. Feature extraction reduces the size of the speech signal without damaging the speech signal power.

Mel frequency cepstral coefficients (MFCC)

Mel frequency cepstral coefficients (MFCC) were first proposed for recognizing monosyllabic words in continuously spoken sentences, but not for identifying the speaker. MFCC computation replicates the human hearing system in order to apply the ear working principle artificially, assuming that the human ear is a trustworthy speaker recognizer [14]. MFCC characteristics are anchored in the observed discrepancy in the crucial bandwidths of the human ear, with frequent filters linearly separated at low frequencies. The high-frequency logarithm is utilized to maintain phonetically important speech signal attributes. Tones of different frequencies are widely found in speech transmissions, and each tone is calculated at the Mel scale with an actual frequency, f (Hz) and the subjective tone. The Mel-Frequency scale has a linear distance of less than 1000 Hz and a logarithmic distance more than 1000 Hz. The perceptive audible threshold of pitch with 1 kHz and 40 dB is defined as 1000 mels and served as a benchmark. [15]

With the use of a filter bank, MFCC is based on signal disintegration. The MFCC provides a discrete cosine transformation (DCT) with a genuine short-term energy logarithm that is shown on the Mel frequency scale [16]. For safety purposes, MFCC is used to identify airline bookings, telephone numbers and speech recognition systems. Some changes to the base MFCC algorithm have been proposed for greater strength such as increasing the log mel amplitudes to an adequate power (about 2 or 3) before the application of the DCT and minimizing the impact of energy efficiencies.

Figure 1 shows the block diagram of the MFCC processor. It provides a summary of all processes and actions to reach the coefficients needed. MFCC can successfully represent the low frequency area better than the high frequency area and can now calculate formants in the low frequency range and describe resonances of the vocal tract.

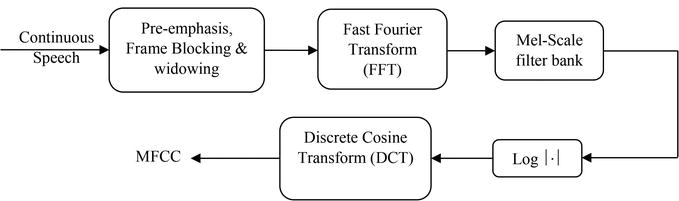


Figure 1 Block diagram of MFCC processor.

Recurrent neural networks

RNNs belong to the most exciting and efficient category of neural network because of the only algorithm with internal memory. RNNs are the most promising. Recurrent neural networks are relatively old, as many deep learning algorithms do. They were first formed in the 1980s, but we saw their true potential only in recent years. RNNs have truly taken precedence with increased computing power along with the huge quantity of data we now need to deal with and the discovery of long-term short-term memory (LSTM) in the 1990s. RNN can recall important facts about the feedback obtained because of their internal memory, which helps them to anticipate very specific things that are coming next. This is why it is the most important algorithm for time series, voice, text, financial data, audio, video, weather, and many more sequential data. Current neural networks can form a much better comprehension than other algorithms for a sequence and its meaning. Current neural networks (RNNs) are a group of neural networks that can be useful for modeling sequence information. RNNs derived from feedback networks have a similar behavior to the functioning of human brains. Put simply: recurrent neural networks generate sequential data predictive results other algorithms cannot do. To properly understand RNNs, people must first have a working knowledge of "natural" feed-forward neural networks and sequential data.

Sequential data is simply ordered data in which related items follow one another. Financial data or the DNA series are two examples. The most common form of sequential data is probably time series data, which is simply a series of data points described in chronological order.

**RNN vs Feed-Forward Neural Networks**

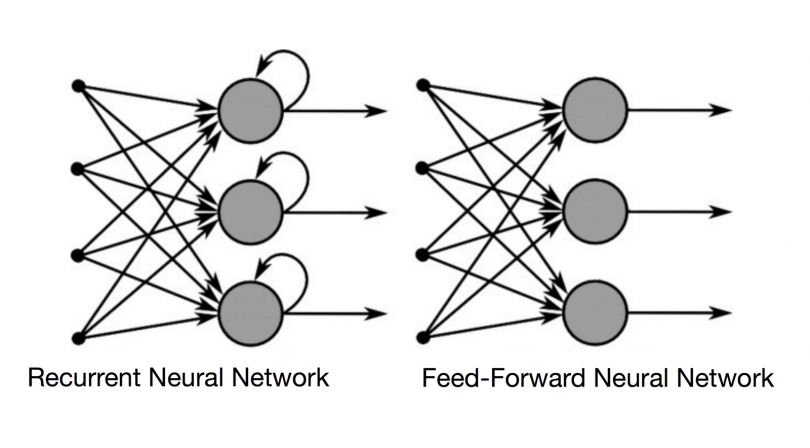


Figure 2 RNN vs Feed-Forward Neural Networks

The names RNNs and feed-forward neural networks come from the way they channel information. In a feed-forward neural network, information only flows in one direction: from the input layer to the output layer through the hidden layers. The data flows directly through the network, never passing through the same node twice. Feed-forward neural networks have no memory of the feedback they receive and are poor predictors. A feed-forward network has no concept of time order because it only considers the current input. It actually cannot recall anything from the past other than its preparation. The above figure 2 shows the difference in information flow between a RNN and a feed-forward neural network.

A traditional RNN has a short-term memory. They have a long-term memory when combined with an LSTM. Consider a standard feed-forward neural network that receives the term "neuron" as input and processes it character by character. By the time it gets to the character "r," it has forgotten about "n," "e," and "u," making it almost impossible for this form of neural network to predict which character will come next. A recurrent neural network, on the other hand, may remember certain characters due to its internal memory. It generates output, copies it, and loops it back into the network. The immediate past is added to the present by recurrent neural networks. As a result, an RNN receives two inputs: the current and the recent past. This is important because the data sequence includes critical knowledge about what is to come, which is why an RNN can do things that other algorithms cannot.

A feed-forward neural network, like all other deep learning algorithms, assigns a weight matrix to its inputs before producing an output. It is worth noting that RNNs apply weights to both the current and previous input. A recurrent neural network can also adjust the weights for both through gradient descent and backpropagation over time (BPTT). RNNs can map one to several, many to several (translation), and several to one input, while feed-forward neural networks can only map one input to one output (classifying a voice).

# Backpropagation through time

# Forward-propagation is used in neural networks to obtain the output of your model and to determine if this output is correct or incorrect in order to obtain the error. Backpropagation is nothing more than traversing the neural network backwards to find the partial derivatives of the error with respect to the weights, allowing you to deduct this value from the weights. Gradient descent, an algorithm that can iteratively reduce a given function, uses these derivatives. The weights are then adjusted up or down depending on which reduces the error. During the training process, a neural network learns exactly like this. We are basically trying to tune the weight during training via back propagation. The following figure 3 shows the notion of forward propagation and back propagation in a neural feed forward network

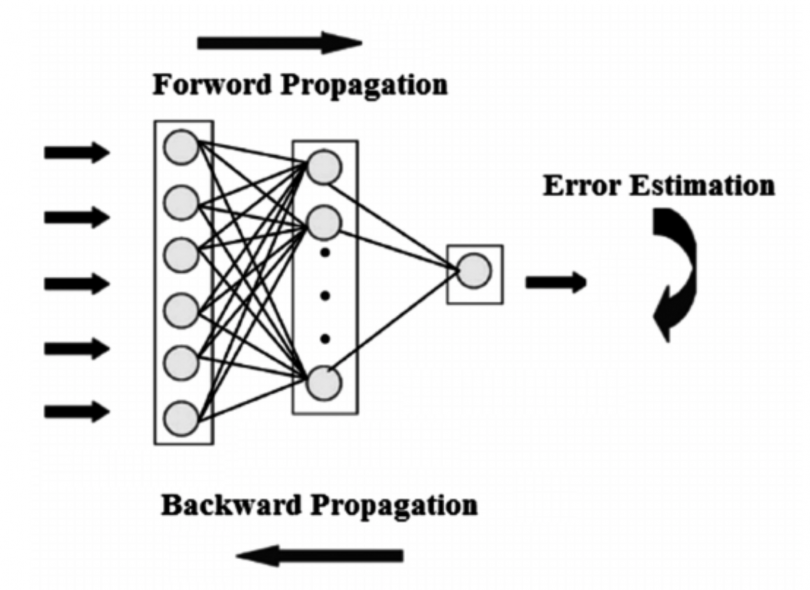


Figure 3 Forward and backward propagation in a neural network

Basically, BPTT is merely a sweet phrase to reproduce an unrolled RNN. Unrolling is a visual and conceptual tool that allows you to comprehend what is happening inside the network. Backpropagation is automatically taken care of, but it does require to be understood how the problem can be solved while developing the recurring neural network in the common programming framework. we can regard an RNN as a sequence of neural networks that we train one after another with backpropagation.

The RNN has two significant hurdles, but first we must know what a gradient is in order to understand them.

A gradient in its inputs is a partial derivative.  if you modify the inputs, a gradient count how much the output of the function changes.

A gradient can also be considered as the path of a function. The higher the pitch, the steeper the pitch and the faster may be the model. But the model stops learning when the path reaches zero. In relation to error variations, a gradient just measures the change in all weights.

EXPLODING GRADIENTS: Exploding gradients occur when the algorithm assigns an absurdly high value to the weights for no apparent reason. However, truncating or squashing the gradients could readily address this problem.

VANISHING GRADIENTS: Gradients are absent when the gradient values are too modest and the model stops learning or takes a long time. In the 19az90s, this was an important topic and considerably more difficult to tackle than explosive gradients. Luckily, Sepp Hochreiter and Jürgen Schmidhuber solved it with the LSTM concept.

Long short-term memory (LSTM): For recurrent neural networks long-term memory networks are an extension, essentially expanding the storage area It is therefore well suited to learn from crucial events with very long intervals. The LSTM units serve as construction units for RNN layers, commonly known as LSTM networks. LSTM allows RNNs to retain inputs for a long time. The reason is that LSTMs contain memory information much like a computer memory. In the LSTM, information can be read, written and removed from the memory.

This memory can be seen as a gated cell, with a gated significance, the cell decides whether to saving or deleting data, based on the value it attaches to the information (i.e. whether or not it opens the gates). Weights, that are also learned via the algorithm, assign importance. This basically implies that over time it learns what information is and what is not important.

In an LSTM, you have three doors: input, forget and output gate. These gates govern whether to allow fresh input (gateway), to delete information, or to allow it to influence the output during the current time period (output gate)

Below is an example of the three gates of the RNN.

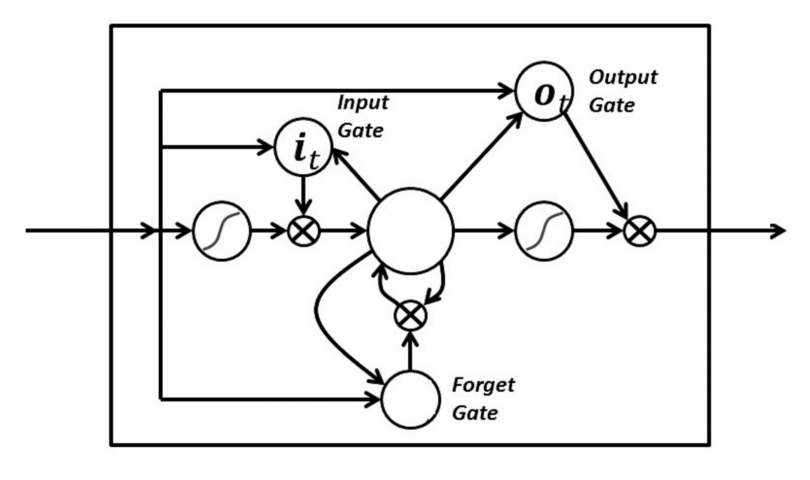


Figure 4 RNN with its three gates

In LSTM, the gates are analogous to sigmoids, which means they range from null to null. They can make backpropagation since they are analogous. The major problems with disappearing gradients are fixed by LSTM since the gradients are kept steep enough to keep their training reasonably brief and their precision excellent.

ROS Middleware and Gazebo

Many environments, such as ROS, Open Robot Control Software (Orocos) and Yet Another Robot Platform (YARP) have been developed in recent years, making it easier to program applications for robots. Since thoroughly validated software pieces are reused, Middleware is particularly valuable to decrease development time and increase robustness in robotics applications. ROS has been selected as a reference middleware for this project. It is the de facto standard and has a bigger community of users and developers than any other middleware. It offers a wide range of drivers, including robotic arms drivers, indoor robots, drones etc.

Real platform experience is academically highly useful, even if it has certain practical disadvantages. The software design provides for a balance with virtual robots and physical robots between practical sessions. It enables a physical robot or its simulated counterpart with just a few changes in configuration to be connected to the same academic application. Many simulators are available: V-REP, webots and so forth. The GazeBo Simulator was chosen as the reference for the instruction of intelligent robotics in this project. It is high-quality free software, supported by the open source robotics foundation (OSRF) and has great acceptance in the international robotics community.

## Implementation

This chapter explains the implementation of the project. The project implementation can be separated into four steps.

1. GUI Creation
2. Preprocessing of data
3. Design of Deep Learning Model
4. Gazebo Map Design
5. GUI Creation: This section gives a brief overview over the architecture and implementation of the Graphical User Interface (GUI). We have designed the GUI as shown in the Figure that allow the user to allow a prerecorded audio file and the user can actually see the text form of the prerecorded command by clicking on a button. And finally, the user can move the robot to the desired location by clicking on a button according to the voice command loaded. GUI is built by using the Tkinter framework. It is a cross- platform framework that’s built into the Python standard library. The main window of the GUI is given in the Figure

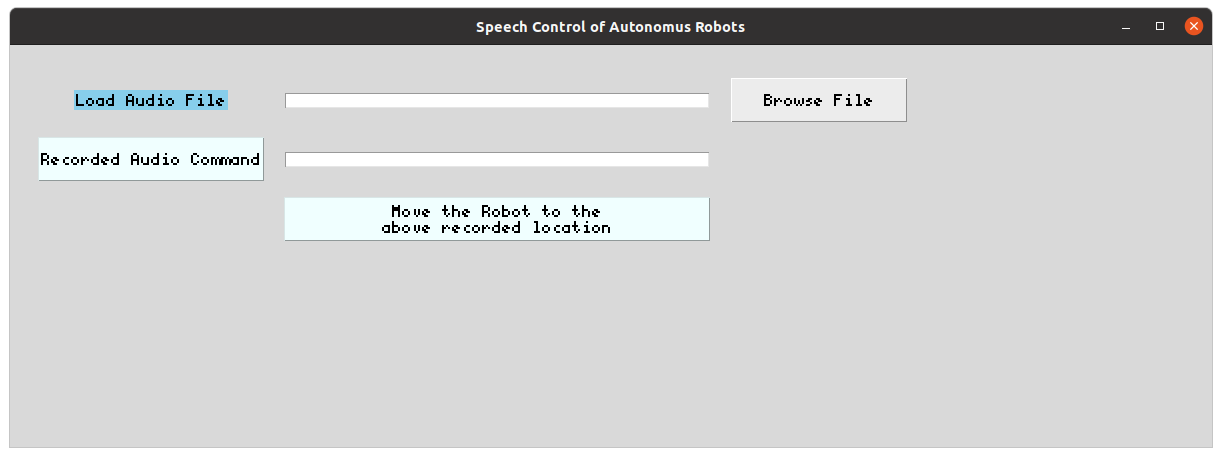


Figure GUI Screenshot

1. Preprocessing of data: The first phase in the speech recognition process in the development of an ASR system is to distinguish the voiced or unvoiced signal and generate feature vectors. Pre-processing adapts or alters the speech signal to make it acceptable for the analysis of feature extraction. In speech signal processing, the signal should mainly be examined if it is corrupted by the background or ambient noise. MFCCs are the Mel Frequency Cepstral Coefficients. MFCC takes into account human perception for sensitivity at appropriate frequencies by converting the conventional frequency to Mel Scale, and are thus suitable for speech recognition tasks quite well (as they are suitable for understanding humans and the frequency at which humans speak/utter).

MFCCs vector for each audio file is extracted by a predefined library called librosa. The code snippet is given in Figure 5. The gist of the code is to loop through the whole dataset and take a voice signal in each iteration and by predefining the sample rate we can drop the audio signals which does not have the predecided number of samples. Next step is to extract the MFCC vector by passing the number of coefficients to extract, interval to apply FFT, hoplength which is slicing window for FFT. The extracted MFCC vectors are all saved into a Json file. Each audio file is mapped with a MFCC vector so that it is easy to train the deep learning model to train a digital data rather than analog data. Next step is to design a model and train the model by splitting the dataset.

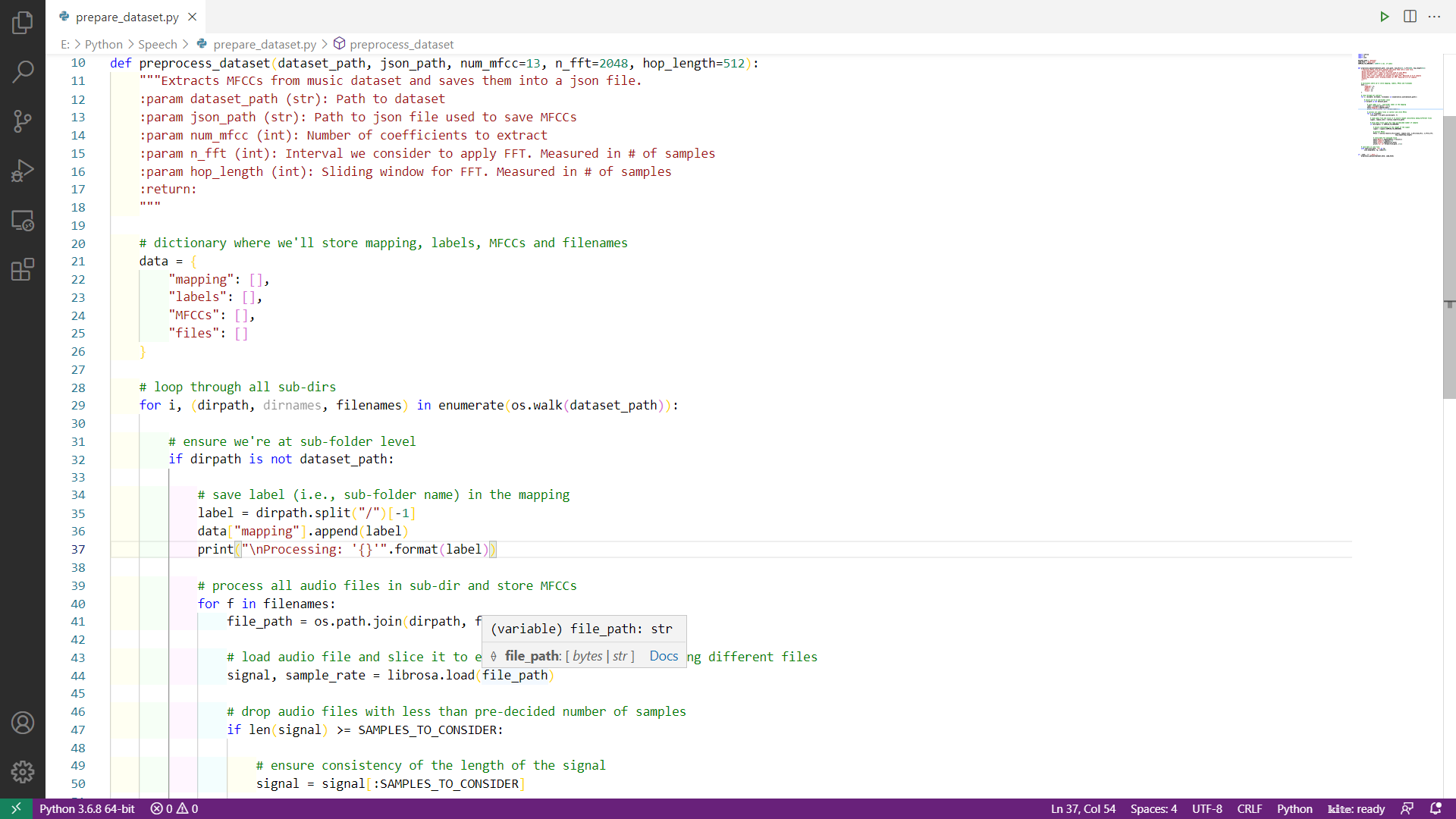


Figure 6 Code Snippet of Extraction of MFCC Vectors

1. Design of Deep Learning Model: The deep learning model is build using TensorFlow which is a predefined library. The training data is loaded as a json file which we have saved in the preprocessing step. The Json file consists of labels and MFCC coefficients of each audio signal. The loaded dataset is split by using a train\_test\_split function from sklearn. The dataset is divided into train, test, validation data. The test and validation are of each 20% of whole dataset and the remaining goes into training data. The code snippet of the model architecture is given in the below figure. The architecture involves three convolutional layers, one dense layer and one softmax output layer. The loss function is chosen as sparse categorical cross entropy and the learning rate is fixed at 0.001.

Now the model is to be trained. This can be done by fit function. This fit function takes the argument of X and Y labels of dataset and number of epochs, batch size, and the X, Y labels of validation dataset. The trained model is saved as a .h5 file. This saved model file is loaded into the prediction method and the test data will be predicted with the help of saved model file. This saved model file contains all the information about the model architecture and the information about the training data. We can save lot of time during the time of prediction by saving a model otherwise it will take huge amount of time to train the model and then use it for prediction. The model architecture is given in Figure 6.

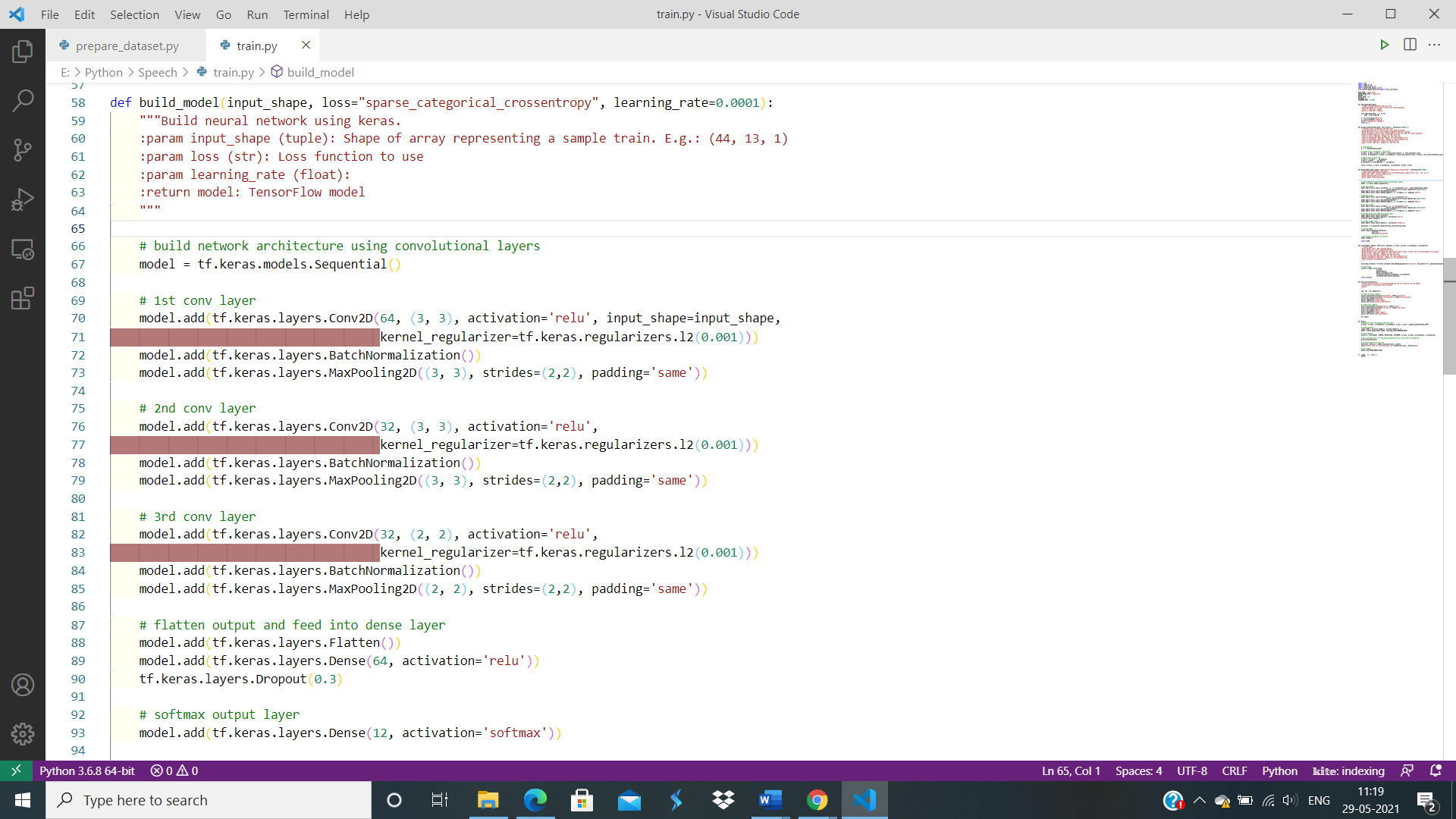


Figure 7 Code Snippet of Model Architecture

1. Gazebo Map Design: The Gazebo simulator materializes the behavior of the physical robot in a virtual scenario. In order to start a world simulation for a specific exercise, a configuration file that determines the scenario, the robots involved, etc. is created and used. Gazebo provides a local viewer to observe the evolution of the simulated world, including the behavior of the robot programmed by the user and also allowing interaction with the scene by adding certain element at runtime, stopping, re-launching the simulation, etc. We designed a gazebo world file which involves a house with different rooms in which a robot can move from one room to another room with a recorded input. The rooms and the commands to each are given in the table are given below.

|  |  |
| --- | --- |
| **Room Name** | **Command to the Room** |
| Room One | Move to Room One/ Go to Room One |
| Room Two | Move to Room Two/Go to Room Two |
| Origin / Home Position | Move to Origin/Go to Origin/Move to Home Position/Go to Home Position |
| Living Room | Move to Living Room/Go to Living Room |
| Kitchen | Move to Kitchen/Go to Kitchen |



Figure 8 Gazebo Map

#### As shown above in the map we have created four rooms and an origin position. The turtlebot is placed at the origin position initially and it can be moved from one room to another room by a recorded voice command. The code for this is written in Python Programming Language. As mentioned in the GUI section to move the robot from one room to another room it is recommended to first load the audio file and then click the Move the Robot button to the desired location. Navigation and Simulation: After performing the map generation in Gazebo, we used the Odometry package, euler\_from\_quaternion, quaternion\_from\_euler, Twist Package, P controller for the controlled rotation of turtlebot for navigation inside the generated map.

## Results and discussion

## Using the simulation control software, we performed experiments of control of virtual robots under ROS. During the experiments there was no difference in robots behaviour compared to the simulations. However, due to differences in the interprocess communication speed and calculation speeds there was need to tune some parameter of the Navigation stack node. Robot movement using a simulation control software under ROS cause very high load on CPU’s. The laptop configuration used in this project is Intel® CoreTM i5-8200@1.6Ghz equipped with 8GB RAM. During the experiments the robot speed was set to 0.4 [m/s] and the whole map building and navigation run smoothly. The ROS and Gazebo software run under 64-bit Ubuntu 20.04 OS. Even though there exist several software platforms for simulation and robot control, ROS allows building of reliable robot control and navigation software and Gazebo simulation together with ROS’s RVis library helps to create simulation, which results can be directly deployed to the real robot hardware. During the robot navigation it is very much important to perform precise localization and position correction of the mobile robot.

## Conclusion

#### The purpose of this study is to develop a reliable environment for simulation and control of mobile robots using the ROS and Gazebo software. It was shown that after designing properly the models of the robot platforms and their working environments the software used in the simulations can be directly used to control the real robots. Simulations and experimental results in environments prove the usability of the models. The main contribution of this paper is that the well-done combination of ROS packages allowed virtual movement of the turtlebot. The paper describes in details which software packages were employed and we hope that the results reported here will be useful at least for part of the roboticists community.

## Acknowledgement

## We are profoundly grateful to **Prof. Peter Nauth and Mr. Sudeep Sharan** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. We would also like to express our deepest appreciation towards Information Technology Department whose invaluable guidance and constructive criticism supported us in completing this project. At last, we must express our sincere heartfelt gratitude to all the authors who helped us indirectly during this course of work.

## References

## J. Benesty, M. M. Sondhi and Y. Huang, Springer Handbook of Speech Processing, Springer Science & Business Media. ISBN 978-3540491255., 2008.

## "Pioneering Speech Recognition," IBM, 7 March 2012. [Online]. Available: https://www.ibm.com/ibm/history/ibm100/us/en/icons/speechreco/

## H. B. a. N. Morgan, Connectionist Speech Recognition:A Hybrid Approach, The Kluwer International Series in Engineering and Computer Science, v.247, Boston: Kluwer Academic Publishers, 1994

## S. Hochreiter and J. Schmidhuber, ""Long Short-Term Memory"," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

## A. G. a. J. S. Santiago Fernandez, "An application of recurrent neural networks to discriminative keyword spotting," Proceedings of ICANN pp 220-229, 2007.

## Kurzekar PK, Deshmukh RR, Waghmare VB, Shrishrimal PP. A comparative study of feature extraction techniques for speech recognition system. International Journal of Innovative Research in Science, Engineering and Technology. 2014;3(12):18006-18016

## Ahmad AM, Ismail S, Samaon DF. Recurrent neural network with backpropagation through time for speech recognition. In: IEEE International Symposium on Communications and Information Technology (ISCIT 2004). Vol. 1. Sapporo, Japan: IEEE; 2004. pp. 98-102

## Shaneh M, Taheri A. Voice command recognition system based on MFCC and VQ algorithms. World academy of science. Engineering and Technology. 2009;57:534-538

## Yousefian N, Analoui M. Using radial basis probabilistic neural network for speech recognition. In: Proceeding of 3rd International Conference on Information and Knowledge (IKT07), Mashhad, Iran. 2007

## Cornaz C, Hunkeler U, Velisavljevic V. An Automatic Speaker Recognition System. Switzerland: Lausanne; 2003. Retrieved from: http://read.pudn.com/downloads60/sourcecode/multimedia/audio/209082/asr

## Shah SAA, ul Asar A, Shaukat SF. Neural network solution for secure interactive voice response. World Applied Sciences Journal. 2009;6(9):1264-1269

## Ravikumar KM, Rajagopal R, Nagaraj HC. An approach for objective assessment of stuttered speech using MFCC features. ICGST International Journal on Digital Signal Processing, DSP. 2009;9(1):19-24

## Kumar R, Ranjan R, Singh SK, Kala R, Shukla A, Tiwari R. Multilingual speaker recognition using neural network. In: Proceedings of the Frontiers of Research on Speech and Music, FRSM. 2009. pp. 1-8

## Chakroborty S, Roy A, Saha G. Fusion of a complementary feature set with MFCC for improved closed set text-independent speaker identification. In: IEEE International Conference on Industrial Technology, 2006. ICIT 2006. pp. 387-390

## de Lara JRC. A method of automatic speaker recognition using cepstral features and vectorial quantization. In: Iberoamerican Congress on Pattern Recognition. Berlin, Heidelberg: Springer; 2005. pp. 146-153

## Ravikumar KM, Reddy BA, Rajagopal R, Nagaraj HC. Automatic detection of syllable repetition in read speech for objective assessment of stuttered Disfluencies. In: Proceedings of World Academy Science, Engineering and Technology. 2008. pp. 270-273