A Comparison of Neural Networks Performance for Sound Source Localization

Arief Ramadhan

18044456

MSc Robotics

University of Bristol
University of the West of England
September, 2019

Abstract

The author compares four neural networks architectures performances in classifying sound location, both in its azimuth and elevation plane. The four architectures are Multi-Layered Perceptron (MLP), Recurrent Neural Network (RNN), Bidirectional RNN, and Convolutional Neural Network (CNN).

The author collects the dataset using a robot head, which is provided with two artificial ears. The robot head is mounted on a pan-tilt mechanism to accommodate movement essential in the dataset acquisition. Fifteen training positions and 27 test positions are contained within the dataset. The raw audio data are then converted into two representations: the Mel-frequency cepstral coefficient (MFCC) and log filter bank energies, before fed into the neural network.

The results show that MLP achieves the best performance for accuracy with less time and memory, and CNN has the highest generalization capability. BRNN requires the most time in training, and RNN has the highest memory cost. In generalizing inputs, the neural networks perform better while using log filterbank energies com than while using MFCC.

Contents

1 Introduction

1.1 Initial Engagement

Machines start to listen. Our smartphones, security systems, and virtual assistant (such as Amazon Alexa and Google Home) now have an audio capability we never thought possible in electronic devices. Audition becomes a necessary feature and could open a wide range of applications in the future, including rescue operation and acoustic mapping (Rascon and Meza, 2017).

There are some advantages of sound over vision. First, unlike vision sound is omnidirectional, means that when a sound source emits energy, a microphone can receive it from all directions (Huang, Ohnishi, and Sugie, 1997). Second, it requires no illumination and less affected by obstacles so that a robot can perceive auditory information in a dark room or from a source behind barriers (Huang, Ohnishi, and Sugie, 1997).

In terms of research popularity, robots audition have only gained increased attention in recent years (Rascon and Meza, 2017). The ultimate goal of the robot audition is to reach a human-like sound understanding or even surpassing it. Human has an advanced audition capability: we can locate and extract the sound we wish to focus on, we can also interpret its meaning (Argentieri, Danes, and Souères, 2015).

In this paper, the author focuses on one aspect of robot audition called Sound Source Localization (SSL). The goal of SSL is to estimate a sound source position via audio data alone (Rascon and Meza, 2017). The author compares four neural network architecture's performance in solving the SSL task. Neural network solves input to output mapping by reducing the need to interpret how the inputs affect the outputs. Hence, it requires fewer assumptions about the environment and the measuring system.

This paper is divided into six chapters. The first includes this introduction, aim, objectives, and project management. A review of the literature on the state of the art of SSL is in the second. The third chapter explains the methodology as well as the tools and equipment used in this study. After that, the author presents the experiment results in chapter four. Following the results, a discussion regarding the neural network performances is covered in chapter five. The report ends with the conclusion and reflection for future projects.

1.2 Aim and Objectives

1.2.1 Aim

The research aims to compare neural network architectures in their performance to locate a sound source in both azimuth and elevation plane, by using a pan-tilt robot head and a pair of artificial ears with microphones.

1.2.2 Objectives

The objectives of the project including details on how to reach them are:

1. Design and build a pan-tilt mechanism for robot head with artificial ears:
    - Review literature of the existing robot for SSL
    - Re-learn Solidworks
    - Design the robot
    - Manufacture and purchase required parts
    - Assemble the robot
2. Learn which Neural Networks architecture have been used in SSL
    - Give a general overview of each neural networks architecture
    - Review literature on NN used in SSL
    - Explore how these architectures can be improved
3. Develop a series of neural networks for SSL based on the findings of the literature review to compare their performance in sound localization
    - Learn Python and Machine Learning frameworks: Tensorflow, Keras, Numpy
    - Build the Neural Network
4. Collect dataset with the robot in a selected environment
    - Choose a room for dataset collection
    - Setup the hardware and recording system
    - Collect audio data
5. Run Neural Networks using the collected audio data as the input

- Test each network
- Tune parameters and select best models for each type of Neural Networks

6. Analyse and compare results
    - Accuracy
    - Generalizaton Ability
    - Time and Memory Cost

1.3 Project Management

Figure 1.1 displays the initial project timeline.

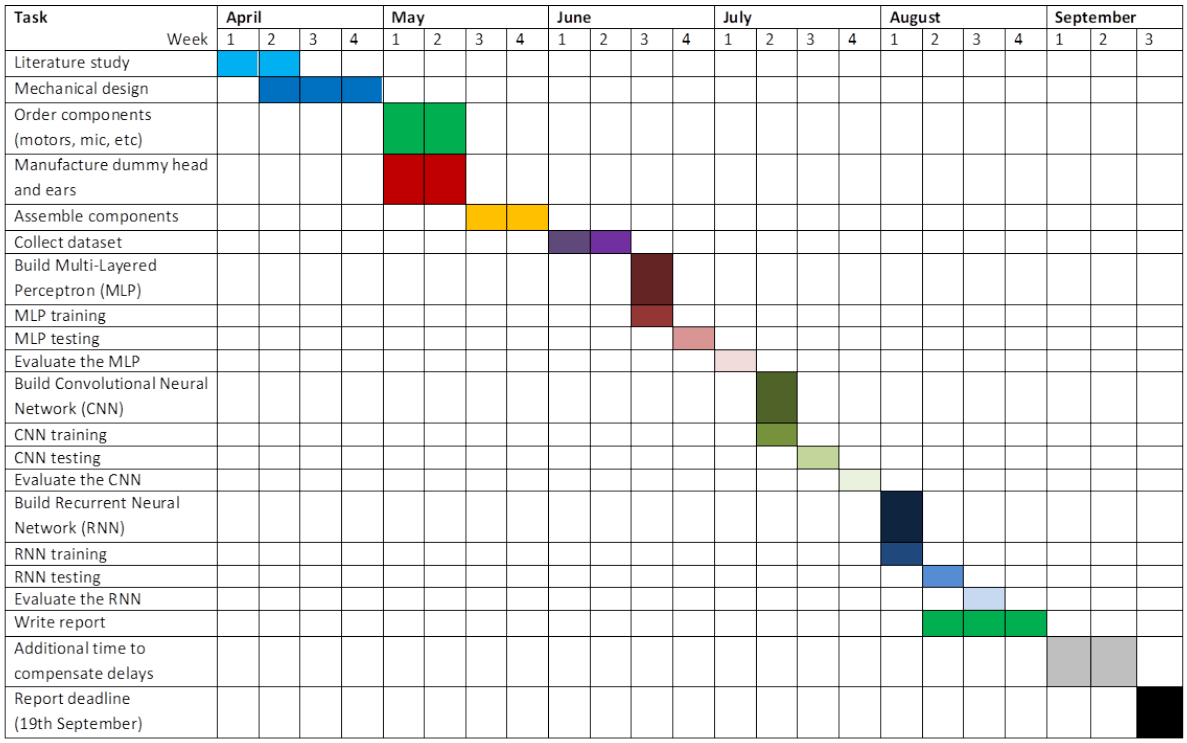

Figure 1.1: Initial Timeline

The project theme was chosen in January. I decided it after one month of thinking and comparing it with other available topics. After that, I did a few literature research on Sound Source Localization (SSL) to get a general view of the current situation. A more extensive literature review was done in April when I wrote the research proposal.

After I finished the research proposal, I started to build the robot. The robot consists of (1) the hardware, and (2) the software part. I began from the hardware.

I looked for robot design that is compatible to be made usinga 3D printer as it is the resource that I had in the lab. I met with my supervisor regularly to discuss which design suited my needs. We decided on the design in May. The inspiration for the platform came from Trossen robotics pan-tilt system, while for the ear and the head were from a paper by (Hwang, Park, and Park, 2011).

After the robot drawing was finished, I ordered and printed the components. It took up to a week to print. However, I misprinted two parts and broke one. I split the printing into two batches; first to test the materials and sizings, and second for the rest of the components. So, manufacturing took a longer time than what was allocated in the timeline.

There was also an extended delivery date for the components. For example, the U2D2 motor interface had a delay in the delivery of about two weeks. I also had to wait for the technician to build housing for my power supply. The power supply construction was not safe, so an enclosure and a fuse that limits the current of the power supply was needed. It took about three weeks to build this.

During these waiting, I started to focus on the software parts. I began by looking at how to record sound using two microphones simultaneously. I used Adobe Audio but then changed to a free alternative, Audacity. After that, I learned to preprocess the audio data using Matlab. However, in the project I used Python Speech Feature library to do the preprocessing.

One of my first obstacle was to control the motor using Raspberry Pi and an octal buffer (74ls241). I followed a lot of tutorials but still couldn't figure out how to do it. I then used Arduino and able to move the motor, but unable to read the servo position. After that, my supervisor and I discussed this issue and decided to use the Dynamixel interface. This interface was easy to use and accommodated different programming languages. I chose the Dynamixel wizard from Roboplus, which I think was the easiest way to control the motors.

After all the hardware was ready, I began to assembly the part. I only need one to two hours to did this, other than two weeks as allocated in the timeline. The time required to collect data and build the neural networks was also quicker than expected, as there were a lot of libraries available. These time gaps compromised for the time I lost by waiting for components to arrive, build, or printed.

To run and tune the networks required a lot of time because there was no definite way of optimizing neural network. I took three week to tune the four neural network architectures.

What I didn't add in the timeline was an additional time to collect more data. This additional data collection was done to test the networks on an untrained position. It aims to know how well the neural network generalize the input. Dataset collection took about 5 hours. So a day was enough for that.

I did a literature review throughout the project, especially when I feel confused about what and how to do the next steps. What I learn from this project was to work on the things you could do first other than waiting and do nothing. The actual timeline of the project is displayed in Figure 1.2.

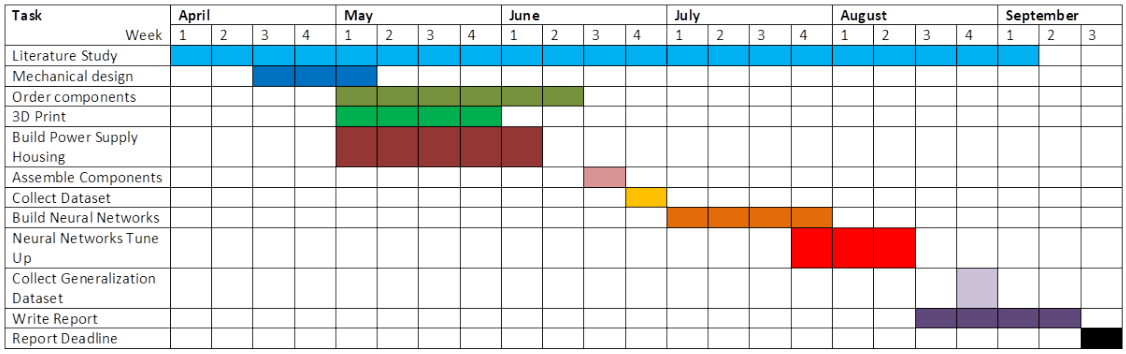

Figure 1.2: Actual Timeline

2 Literature Review

This section reviews works of literature on the state-of-the-art in Sound Source Localization (SSL) research. It contains three main parts: a brief history of SSL, definition of the SSL problems, and reflections. The later summarizes the findings in the literature review. It aims to give an insight into the methods implemented in this study.

2.1 Brief History of SSL

How can the spatial location of a sound source be known, as sound has no dimensions in space(Yost, 2017)? Lord Rayleigh came with an answer in 1876 (Yost, 2017). He explained that displacement of the ears provided a mean for auditory spatial discrimination (Yost, 2017). A sound would be more intense at the ear nearest to the source than the far ear. This loudness difference heard by each ear would later be named as Interaural Level Difference (ILD), a popular feature in the robotic audition for spatial localization in the 21st century (Rascon and Meza, 2017).

Two years after Rayleigh's discovery, Silvanus Thompson delivered a lecture describing that the location of a sound source could be inferred from the relative phases of the sounds at the two ears (Yost, 2017). This phase difference is known as the Interaural Phase Difference (IPD), which is also a commonly used feature in building SSL robots (Rascon and Meza, 2017).

Rayleigh discussed both ILD and IPD in a lecture in 1907 (Yost, 2017). He pointed out that ILD was a possible cue at high frequency sound, while IPD could be a cue at low frequencies (Yost, 2017). The combination of ILD and IPD form the duplex theory of sound localization (Yost, 2017).

Several kinds of audition research took place after Rayleigh and Thompson's era. One of the most distinguish work was by Lloyd Jeffress (Yost, 2017). He presented the "delay line" or coincidence network as a possible physiological process to determine IPD (Yost, 2017).

Until the last decade of the 20th century, SSL research mainly concentrated on its psychoacoustic aspects, and did not gain a lot of attention by robotic researchers. The first SSL robot was in 1989 with a robot named Squirt. Squirt was an autonomous mobile robot which incorporates sensing, actuation, onboard computation, and onboard power supplies (Flynn et al., 1989). This robot acts as a 'bug' hiding in dark corners and venturing out in the direction of last heard noises, only moving after the noises are long gone (Flynn et al., 1989).

The idea of using robot were considered by Japanese researchers in the 1990s (Rascon and Meza, 2017). In 1993, Takanashi et al. studied anthropomorphic auditory system for a robot (Rascon and

Meza, 2017). This research was followed by works on SSL using robots like Chiye, Hadalay, and Jijo-2 (Rascon and Meza, 2017).

Research in SSL robotics started to generate interest after SIG was presented as an experimental platform for the RoboCup Humanoid Challenge 2000 (Rascon and Meza, 2017). SIG was built to promote audition as an essential skill for robots (Rascon and Meza, 2017).

A critical rift occurred in SSL techniques in recent years. Initially, contributions to robot audition were rooted in the binaural paradigm (Argentieri, Danes, and Souères, 2015). Binaural audition cemented itself by the motivation to imitate nature (Rascon and Meza, 2017). On the other hand, there was a motivation to increase performance, which pushed for the use of more microphones (Rascon and Meza, 2017).

The rise of Artificial Intelligence (AI) also challenges conventional calculation methods. AI method such as deep learning is considered as a way of removing the complexity of calculation that was innate in conventional. The increase in computational power made deep learning a promising method (He, Motlicek, and Odobez, 2018).

Research involving deep learning methodology for SSL purposes includes SNN by (Wall et al., 2012), MLP and CNN models by (He, Motlicek, and Odobez, 2018), and CNN by (Vera-Diaz, Pizarro, and Macias-Guarasa, 2018). So far, the author only finds a single work (He, Motlicek, and Odobez, 2018) on the comparison of neural network performance in solving SSL task.

The central goal for robots with an SSL system has been to support interaction with humans (Rascon and Meza, 2017). As technology advances, a further realization of SSL usage may be understood. For example, SSL can improve surveillance, rescue operation, and navigation.

2.2 Definition of the SSL Problem

To solve the SSL task, we must first divide it into components. Divide and conquer as the old saying goes. This section gives a detailed account on each component of SSL – from the simplest to the more advanced ones.

2.2.1 Position Components

There are two components of source position: (1) direction of arrival and (2) distance (Rascon and Meza, 2017). The direction can be divided into two parts: azimuth and elevation angles (Rascon and Meza, 2017).

Azimuth covers the left-right direction or rotational in the vertical axis. Elevation covers the front-back direction or rotational in the horizontal axis. Compare to direction, distance is harder to predict as it is more affected by factors like timbre, loudness, and room acoustics (Rascon and Meza, 2017).Currently, most research in SSL focus on estimating the direction of arrival (Rascon and Meza, 2017).

2.2.2 Features

The initial thing to decide in designing an SSL robot is the type of sound feature to use. Humans mainly depend on interaural differences and spectral modification to locate a sound (Hwang, Park, and Park, 2011). The author explains these features in this section.

Inter-Aural Differences

As mentioned in the introduction, initial contributions to robot audition were rooted in the binaural paradigm (Argentieri, Danes, and Souères, 2015). This paradigm relied on cues such as the inter-aural differences. These cues show the difference of value obtained in each microphone or ear. There are four common types of inter-aural differences:

- Inter-aural Time Difference (ITD) is the time difference in which sound enters the microphones (Rascon and Meza, 2017)
- Inter-aural Phase Difference (IPD) is the phase difference of sound wave entering the microphones(Rascon and Meza, 2017). This value can be calculated when both the ITD and sound frequency are known.
- Inter-aural Intensity Difference (IID) is the volume difference the sound enters the microphones (Rascon and Meza, 2017).
- Inter-aural Level Difference (ILD) is the frequency-domain version of IID (Rascon and Meza, 2017). It covers the difference in loudness and frequency distribution of sound wave captured by the microphones (Hwang, Park, and Park, 2011).

Estimating both azimuth and elevation cannot be done with inter-aural differences alone because a sound source might have the same inter-aural value in several locations (Rascon and Meza, 2017). This problem is called the *cone of confusion* and illustrated in figure **??**. The figure showed the location where the sound has the same inter-aural value.

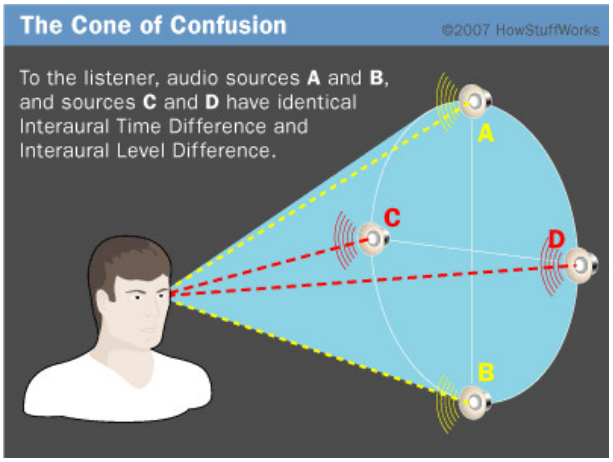

Figure 2.1: Cone of Confusion

Spectral Modification

Human utilizes inter-aural differences along with spectral modification in locating a sound source. The spectral modification describes how the shape of the head and ear modifies the sound before it reaches the human's eardrum (Murray and Erwin, 2011) (or microphone in this case). The shape has a preference that it might amplify or attenuate certain frequencies (Murray and Erwin, 2011).

Scattering Theory

Lax and Phillips introduced the scattering theory in 1967 (Lax and Phillips, 1990). The theory compares the asymptotic behavior of an evolving system before and after being perturbed (Lax and Phillips, 1990).

A paper by Nakadai, Matsuura, Okuno, and Kitano explained how scattering theory could be used for SSL purposes (Nakadai et al., 2003). The scattering theory is employed to take into consideration the diffraction of sounds around a robot's head for a better approximation of IID and IPD. They claim that both azimuth and elevation angle can be estimated by using scattering theory.

2.2.3 Transfer Function

The information on spectral modification and sound scattering are contained within the system's transfer function (Hwang, Park, and Park, 2011). There are several types of transfer functions, including:

- Head-Related Transfer Function (HRTF): HRTF is the ratio of sound pressure at each eardrum to that measured at the head center with the head absent (Hwang, Park, and Park, 2011)
- Pinnae-Related Transfer Function (PRTF): PRTF is the ratio of sound pressure at each eardrum to that measured just outside the ear (Murray and Erwin, 2011)

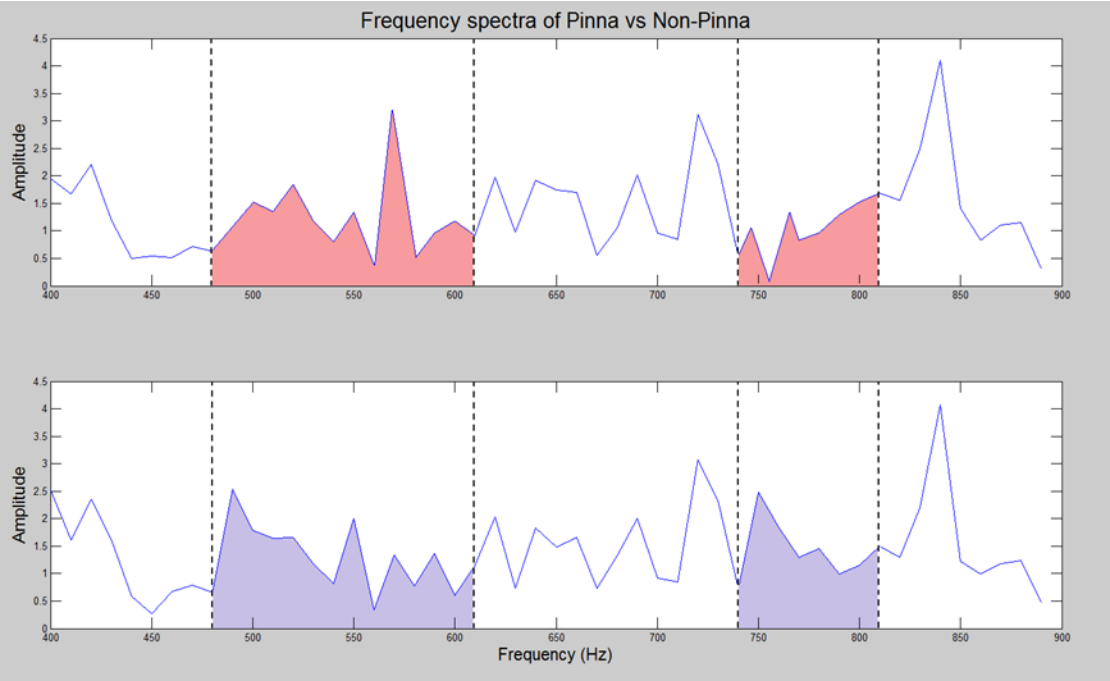

Figure 2.2: Pinnae Related Transfer Function (Murray and Erwin, 2011)

- Inter-aural Transfer Function (ITF): ITF is the ratio of sound pressure at one eardrum to another (Hwang, Park, and Park, 2011)

The transfer function is obtained by first transforming the recorded sound wave from the time domain into the frequency domain (Potisk, 2015). After that, the frequency domain features are compared according to which type of transfer function is employed (Potisk, 2015). An example of the PRTF comparison is shown in Figure 2.

The first drawback of using transfer function for SSL purpose is the need to design a dummy head and ears for the measurement (Rascon and Meza, 2017). A simple design that shows significant spectral modification is desired for ease of manufacture and measurement (Hwang, Park, and Park, 2011). Like a fingerprint, each head and ear shapes form a unique spectral modifier.

The second drawback is that the SSL system needs to be retrained or re-calibrated if it is installed in a different room. Some experiments, such as done by (Hwang, Park, and Park, 2011) also suggest the system to be trained in an anechoic chamber – to reduce noise and reverberations – if the transfer function representation is included as the input feature.

2.3 Ear Design

The physical structures of a listener, such as head, pinnae, shoulder, and torso, transform the sound waves when they reach the listener's eardrum. HRTF characterizes this physical transformation of the sound wave. Humans mainly depend on the interaural differences and spectral modification to perceive

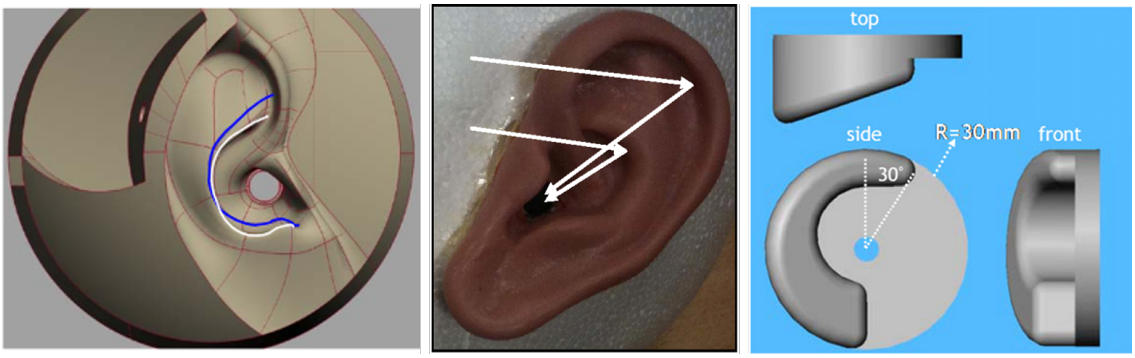

Figure 2.3: Three Ear Designs (Murray and Erwin, 2011) (Rodemann et al., 2008) (Hwang, Park, and Park, 2011)

the lateral and vertical angle of a sound source. The inter-aural differences and spectral modification informations are both contained within the HRTF. (Hwang, Park, and Park, 2011)

The author's robot design needs to be equipped with ears that can provide proper sound cues as human pinnae do. If the ear is adequately designed, it is expected that sound direction in 3-D space can be estimated from two microphone signals only (Hwang, Park, and Park, 2011).

Figure 2.3 displays several ear models used in previous SSL study. The left one is ear by Rodemann, which is modeled after the human outer ears. The blue line indicates the concha of the left ear, while the white corresponds to the right ear's concha outline. The middle one was used by Murray to locate the elevation angle of a sound source. He uses an anthropomorphic design made out of silicon.

The right design is by Hwang, Park, and Park. They created a simple geometry that relies on a specific part called posterior concha wall (Hwang, Park, and Park, 2011). They pointed out that the concha wall is a crucial part in creating spectral modification that is required for sound localization (Hwang, Park, and Park, 2011).

2.4 Dataset

There are lots of datasets that can be used for SSL Purposes. The author identified six available datasets:

- Camil dataset: The Camil dataset is a unique set of audio recordings made with the robot Popeye. Popeye was a realistic dummy head mounted on a pan-tilt actuating system. The dataset was gathered to investigate audio-motor contingencies from a computational point of view.

- Kemar dataset: This dataset consists of recordings using a Kemar dummy head with two microphones in MIT's anechoic chamber (Gardner, Martin, et al., 1994). There are 710 different positions–elevations from -40 to +90 degrees and azimuth from 0 to 360 degrees–in this dataset (Gardner, Martin, et al., 1994).

- Avasm dataset: Similar to Camil, the Avasm is completed using the dummy head POPEYE. Avasm consists of a set of audio-visual recordings for localization in 2D.

- Rwcp database: This dataset is one of the first collected for scene understanding (Rascon and Meza, 2017). The data includes non-speech sounds recorded in an anechoic room, reconstructed signals in various rooms, impulse responses for a microphone array, speech data recorded with the same array, and recordings of background noises (Nakamura et al., 2000).

- Aira corpus: Aira corpus is made up of recordings in different environments that vary in terms of noise energy, reverberation, amount of microphones used, and mobility of sources. The recordings were made using a 3-microphone array.

- Ravel: Ravel contains typical scenarios useful for human-robot interaction (HRI) systems. The data contains video and audio recordings using binaural head.

2.5 Neural Networks

Neural network is a set of algorithms that are designed to recognize patterns (Skymind). It is inspired by the human brain and went on to become one of the most powerful methods in the field of artificial intelligence.

The main advantage of neural network is that it eliminates the need to interpret how the inputs affect the outputs, as this is learned autonomously during training. In the following section, we discuss three types of neural network and how they relate to sound localization.

Neural networks can perform either classification and regression task. Classification involves identifying a class membership, or in other words is when the network predicts to which class does an input belongs to. Regression involves predicting a value or a response.

2.5.1 Multi-Layered Perceptron

The most basic form of neural network is called Feedforward Neural Network or Multi-Layered Perceptron (MLP). In this network, information flows through the network only in the forward direction. There are no feedback connections that channel the output back into itself.

Figure **??** displays the structure of MLP networks. MLP contains three kinds of layers: input, hidden, and output. The input layer is where the initial data brought into the system for processing. The hidden layer is where the processing or computation occurs. The output layer is the last layer in the network where the input has been interpreted.

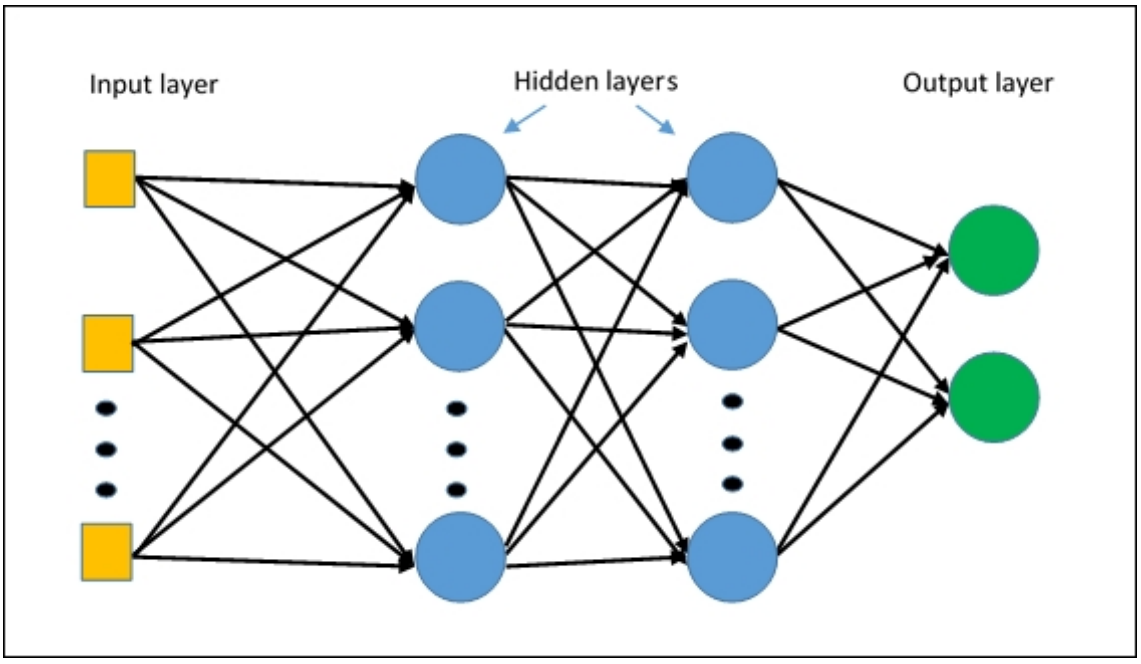

Figure 2.4: MLP Networks

In supervised learning, a collection of input-output pairs are used by the MLP network for training. The goal is to approximate some functions (f) that best maps an input (x) to its output (y) (Goodfellow, Bengio, and Courville, 2016). The approximations happen in the hidden layers by tuning the function's parameter through an iterative process known as gradient-based learning (Goodfellow, Bengio, and Courville, 2016). Section 2.3 gives a detailed explanation about gradient-based learning.

MLP in SSL

Murray and Erwin developed an MLP network to predict the elevation angle of a sound source (Murray and Erwin, 2011). They utilized the spectral modification of artificial pinnae as the input for their network [5]. Weipeng He, Motlicek, and Odobez developed an MLP network for estimating both azimuth and elevation angle based on the generalized cross-correlation with phase transform (GCC-PHAT) input (He, Motlicek, and Odobez, 2018). The GCC-PHAT function is the most popular method for estimating ITD (He, Motlicek, and Odobez, 2018). Their networks were able to predict with an average error of fewer than 5 degrees (He, Motlicek, and Odobez, 2018).

2.5.2 Gradient Based Learning

One way to understand MLP is to begin with linear models (Goodfellow, Bengio, and Courville, 2016). Assume we have a function that transform the input (x) into the output ($\hat{y}$). Equation 2.1 displays this relation. The letter "w" in the equation symbolizes the network's weight, while the letter "b" means the

bias term.

$$\hat{y} = wx + b \tag{2.1}$$

The output ($\hat{y}$) is the prediction made by the function using the input (x). We aim to create a function that has a minimum error between the actual and predicted value. A loss function measures the error. There are many forms of the loss function, for example the Mean Squared Error (MSE):

$$L = \frac{\sum_i^n (y_i - \hat{y}_i)^2}{n} \tag{2.2}$$

The letter "L" symbolized the loss function, in this case we use MSE. The letter n indicates the number features in a single training example. The regular y is the actual value, while ($\hat{y}$) is the predicted value. To calculate the total of loss function across all training examples, we use cost function. Equation shows the formula to calculate cost function.

$$J(w,b) = \frac{1}{m} \sum_i^m (y_i - \hat{y}_i)^2 \tag{2.3}$$

The letter J indicates the cost function and m indicates the number of training examples.The cost function is used by neural networks to update the network parameters – weight(w) and bias(b). The weight and bias is updated through a process known as gradient descent. Gradient descent is possible because the networks perform back propagation. Back propagation allows the information of cost function flow backward through the network in order to compute gradient (Goodfellow, Bengio, and Courville, 2016).

2.5.3 Bias and Variance

There is a dilemma in building a neural network. It is caused by the diversity of network architecture and hyper parameter to choose from and select which suits our need. If we go to a deeper level to figure out why the dilemma arises, we find that it is because we have to do a trade off between bias and variance.

Bias is the accuracy of predictions (Guanga, 2011). It shows the difference between the predictions and actual values of the function (**goodfellow**). Variance relates to the ability to generalize new data. It describes the deviation from the expected estimator value of the function or parameter (**goodfellow**). Figure 2.5 visually explains both bias and variance.

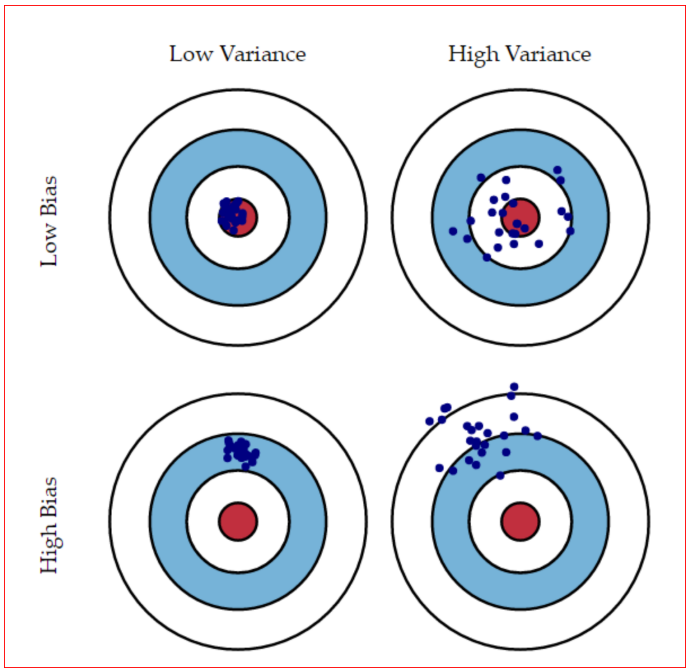

Figure 2.5: Bias and Variance

The ideal would be to have low bias and low variance. However, to reduce the amount of variance in a prediction, we must add bias, while to decrease bias, we need to increase variance.

There are several things that can minimize bias: change the network architectures, train a bigger network, train longer, use a larger dataset, and use a better optimization strategy. While to reduce variance what we can do are: adding more diverse data to the training set, performing regularization, and changing the networks architecture.

2.5.4 Regularization

Regularization trades an increased bias for a reduced variance (Goodfellow, Bengio, and Courville, 2016). It works by eliminating some portions of training sets (Goodfellow, Bengio, and Courville, 2016). Technically, regularizer decreases the effect of the activation function. Therefore, a less complex function will be fit to the data, effectively reducing overfitting. An effective regularizer is one that makes a profitable trade, reducing variance significantly with minimal increase in the bias (Upadhyay, 2019). There are several techniques of regularization, including:

- L1: This technique is also called Lasso Regression. L1 shrinks the less important features' coefficient to zero; thus, removing some feature altogether. L1 works by adding a a penalty term to the cost function to adjust the networks weights and avoid overfitting (Peixeiro, 2019).
- L2: L2 is also named Ridge Regression. Ridge Regression forces the less important feature to decay towards zero, but never reach zero, so it is not excluded entirely. Similar to L1, L2

introduces a penalty term to the cost function to adjust the networks weights (Peixeiro, 2019).

- Early stopping: In early stopping, a part of the training set is taken as the validation set. When the performance on the validation set is getting worse, the network stops the training on the model.
- Dropout: Dropout provides an inexpensive but powerful method of regularization. At every iteration, dropout randomly selects some nodes and removes them from the rest of the connections. Each iteration involves a distinct set of nodes and results in a different set of outputs (Peixeiro, 2019).
- Data Augmentation: Data augmentation is a way to increase the diversity of data for training models without actually collecting new data. It is done by adding bias to the original data, then adding the data to the training set (Peixeiro, 2019).

2.5.5 Optimization

As mentioned in section 1, gradient descent is an algorithms to optimize neural networks. There are three ways to perform gradient descent:

- Batch Gradient Descent (BGD): BGD computes the gradient of the cost function for the entire training dataset (Walia, 2017). As we need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow. However, it makes smooth updates in the model parameters (Walia, 2017).
- Stochastic Gradient Descent (SGD): SGD performs a parameter update for each training example (Walia, 2017). While BGD updates the parameter after exploring the whole dataset, SGD performs one update after each training data. SGD is usually much faster than BGD, though its parameter update is so fluctuating or noisy (Walia, 2017).
- Minibatch Gradient Descent (MBGD): MBGD takes the best of both SGD and BGD. It performs an update for every mini-batch of n training examples (Walia, 2017). This way, it reduces the variance of the parameter updates, which made the update more stable than in SGD (Walia, 2017).

Optimizing the gradient descent

Optimizer is used to further improve gradient descent performance. An optimizer updates the weight parameters to minimize the loss function. There are a lot of different optimizer and the number keeps adding as research in neural networks grow. The author presents a few examples:

- Root Mean Square Propagation (RMSprop): RMSprop uses a moving average of the squared gradients to normalize the gradient (Stewart, 2019). In RMSProp learning rate gets adjusted automatically for each parameter (Stewart, 2019).

- Momentum: Momentum accelerates SGD by navigating along the relevant direction and softens the oscillations in irrelevant directions (Stewart, 2019). This is done by reducing unnecessary parameter updates, which leads to faster and stable convergence (Stewart, 2019).

- Adagrad: Adagrad adapts the learning rate to the parameters, performing smaller updates for parameters associated with frequently occurring features, and larger updates for parameters associated with infrequent features (Stewart, 2019). It is well-suited for dealing with sparse data.

- Adam: Adaptive Moment Estimation (Adam) computes adaptive learning rates for each parameter (Stewart, 2019). In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average similar to momentum (Stewart, 2019). Adam is a combination of RMSprop and momentum.

2.5.6 Activation Function

The activation function is used to add non-linearities to the system. It is used so that the network can learn complex functional mappings from data. The most common type of activation functions are:

- Sigmoid: Sigmoid unit saturates to 1 when the input is high and to zero when the input is low. This unit is more common in settings other than MLP networks, such as Recurrent Neural Network (RNN) (Goodfellow, Bengio, and Courville, 2016).

- Tanh: This function is similar to Sigmoid, only that it saturates to -1 when the input is low, which makes Tanh works well for a classifier (Goodfellow, Bengio, and Courville, 2016).

- Rectified Linear Unit (ReLU): ReLU is easy to optimize because it is similar to linear units. The only difference between ReLU and a linear unit is that it outputs zero for all input value lower than or equal to zero. ReLu is computationally cheaper than Tanh and Sigmoid (Goodfellow, Bengio, and Courville, 2016).

Before the introduction of ReLU, most neural networks used the Sigmoid or Tanh activation function (Goodfellow, Bengio, and Courville, 2016). Figure 2.6 shows Tanh, Sigmoid, and ReLU activation functions and formulas.

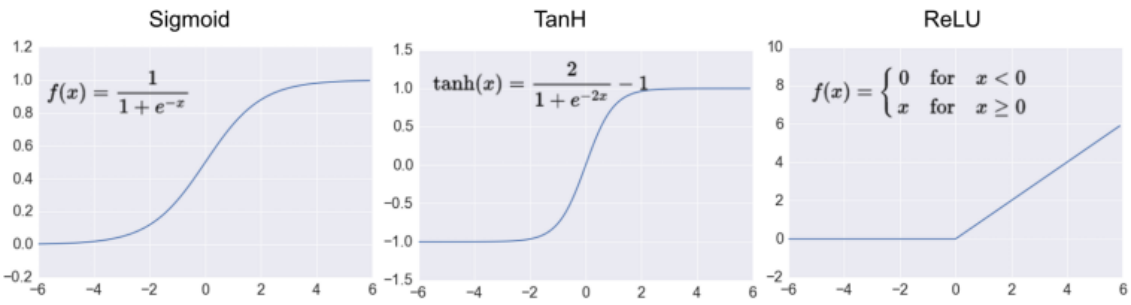

Figure 2.6: Three Activation Functions (Mayo, 2017)

2.5.7 Convolutional Neural Network

Convolutional Neural Networks are a kind of neural network for processing data that has a known grid-like topology (Goodfellow, Bengio, and Courville, 2016), such as time-series (1-D) and image (2-D).

CNN has two features that reduce the number of calculations: sparse interactions and parameter sharing (Goodfellow, Bengio, and Courville, 2016). Sparse interaction is the concept that a neuron's output unit may interact with only a subset of the neurons in the next layer (Goodfellow, Bengio, and Courville, 2016). It is different from the traditional networks where every neuron in a layer is connected to every other neuron in the next layer. (Goodfellow, Bengio, and Courville, 2016).

Parameter sharing is the idea of using the same parameter for more than one function in the model (Goodfellow, Bengio, and Courville, 2016). In a traditional neural net, each element of the weight matrix is used once when computing the output of a layer (Goodfellow, Bengio, and Courville, 2016). In CNN, one parameter is exposed to different kernels and filters.

Figure **??** displays an example of CNN. The essential components of CNN are convolutional layer and pooling layer. Convolutional layer is when the input is "scanned" by a set of filters to perform convolution operations. The pooling layer down sampled the convoluted inputs by summarizing the presence of features in patches of the feature map. The use of convolutional layers is computationally expensive, thus, it requires a longer time to train a CNN compare to other types of network.

CNN has achieved extraordinary performance in the visual domain (Dai Wei et al., 2016). Examples include image classification, face recognition, handwritten digit recognition, and recognizing traffic signs (Dai Wei et al., 2016).

CNN in SSL

Lately, CNNs have been applied to robot audition (Dai Wei et al., 2016), such as Hannun work on speech recognition and Dai. et al work for acoustic scene recognition (Dai Wei et al., 2016).

CNN has also been used for solving SSL. Unlike any other networks, CNN is commonly used to process 2D data. Weipeng He, Motlicek, and Odobez considered the audio input features as images

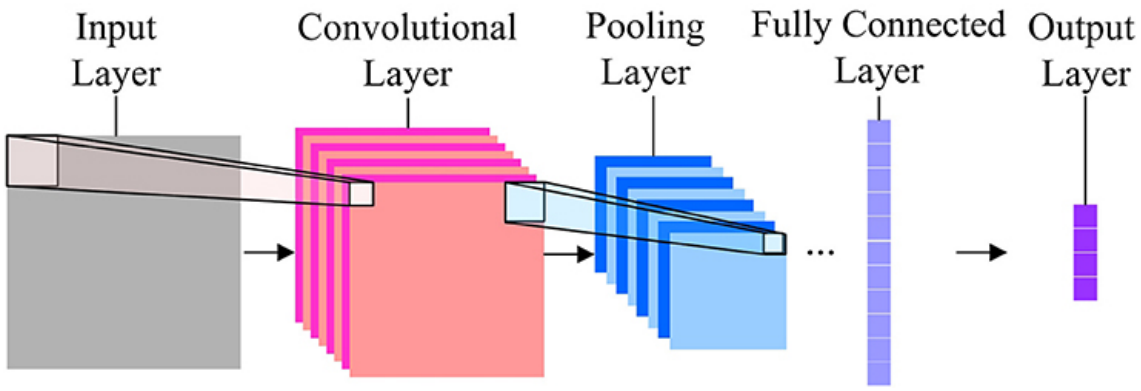

Figure 2.7: CNN Networks

(He, Motlicek, and Odobez, 2018). The recorded audio wave is transformed into a frequency-time graph which is known as a spectrogram. Filters are then applied to that image to estimate the direction of the sound source. In this research, a similar method may be applied.

Thuillier, Gamper, and Tashev developed a CNN-based localization to estimate the elevation angle of a sound source through classification; the system was trained with a sound source coming from one of a set number of angles, after that, it predicted from which angle the sound came from (Thuillier, Gamper, and Tashev, 2018). This research might adopt a similar approach to that of (Thuillier, Gamper, and Tashev, 2018). In addition to elevation, the azimuth angle is also classified.

2.5.8 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of networks that is suitable for processing sequential data, such as sound, speech, and text. Similar to CNN, there is parameter sharing in RNN networks. However, unlike CNN (parameter sharing across neurons), the parameter sharing in the RNN happens across different index of time or sequence.

There is a variant of RNN called Bidirectional RNN (BRNN). BRNN is a combination of an RNN that moves forward through time, with another that moves backward. Two most common effective models of RNN, as stated in (Goodfellow, Bengio, and Courville, 2016) are Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM). In terms of performance LSTM is more powerful, but GRU is simpler and lighter.

In addition to conventional backpropagation, there is backpropagation through time (BPPT) in an RNN network. The purpose of BPPT is to pass sequence or timesteps data across the network. However, BPTT can be computationally expensive as the number of timesteps increases.

RNN in SSL

Adavanne, Politis, Nikunen, and Virtanen developed a hybrid RNN model for localizing multiple sound events (Adavanne et al., 2018). Their system estimated both the azimuth and elevation angle by using

a sequence of spectrograms as the network inputs. A similar approach might be adapted to this study. Another option is to use input known Mel Frequency Cepstral Coefficient (MFCC) which is a type of sequence data commonly used for sound recognition (Ganchev, Fakotakis, and Kokkinakis, 2005).

2.6 Reflection and critical appraisal of the literature review

In this literature review, the author browses through the history, technology, and theoretical aspects of SSL. The author now has to pick and choose the most exciting and relevant findings to be used for the methodology.

From the hardware perspective, a pan-tilt mechanism that enables the robot head to rotate in its vertical, as well as horizontal angle, is needed. An ear design that is simple and effective ear design that corresponds to high SSL accuracy is required. The author also has to choose the type of neural networks architectures to employ as well as their hyper arameter. Furthermore, there are types of audio representations, audio devices, and experiment setup that need to be determined.A further investigation of the methodology the author implement in the study is discussed in the next section.

3 Methodology

Methods make no sense, where there is no argument; and an argument makes no sense where no methods are used (Faludi, 2013). This chapter explains the methodologies implemented in this study and the arguments on why they are chosen.

The chapter is divided into six major sections: robot design, recording devices, audio processing, data set collection, neural networks, programming, and finally concluded in the summary of methods.

3.1 Robot Design

The author's robot consists of three parts: the pair of ears, head, and pan-tilt system.The ear design is heavily based on the works of (Hwang, Park, and Park, 2011). There are three reasons why their design is imitated. First, the ears performed well in localizing azimuth and elevation angles. They located a sound source position accurately for 80% of the time. Second, compared to the findings in the literature review, the design is relatively easy to replicate as they included the dimensions of the ear in the paper. Third, while there are artificial ears available in the market, it is hard to find the one that is compatible with our head and microphones.Therefore, it is better to design the ear rather than buying an already existing one.

Figure 3.1 shows the author's mechanical drawing of the ear. As mentioned in the literature review, the posterior wall of the concha is responsible for the spectral notches that serve as important front-back and up-down cues. Hwang's design utilizes the posterior wall to differentiate elevation angles. The ear starts to notice elevation differences when the sound source frequency is 6 kHz and above.

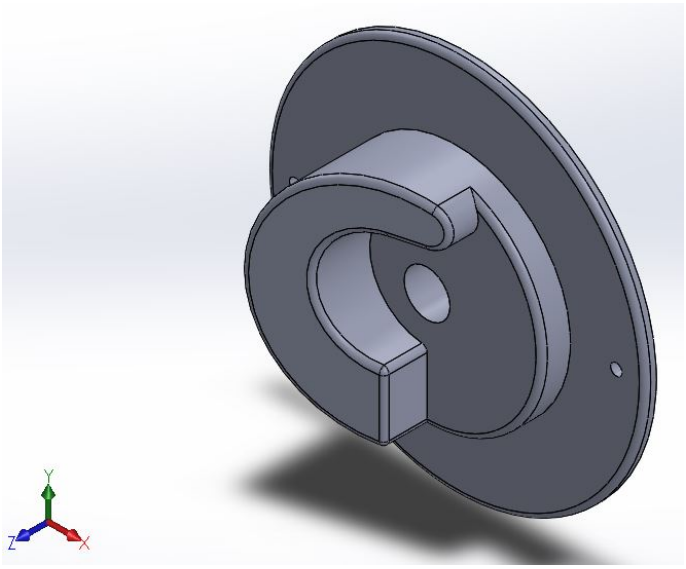

Figure 3.1: Robot Ear

A microphone is attached in each hole of the ears. The mic fits the hole perfectly, so the author doesn't need anything to keep it from moving. The ear is attached using nuts and bolts to a spherical head of 20 mm diameter, as shown in Figure 3.2. There are two sides of the head: left and right. When both sides are connected, it leaves a hollow part at the bottom, which is intended to give way for the threaded rod during rotation.

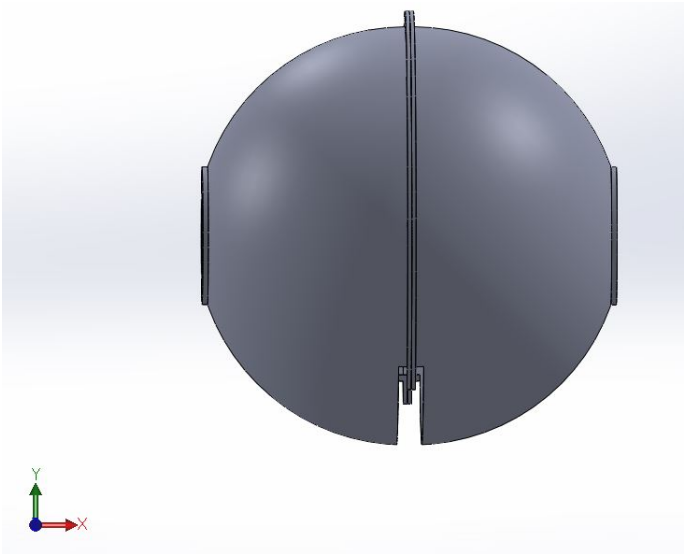

Figure 3.2: Robot Head

The robot head and ear are mounted on a pan-tilt system. The pan-tilt system (Figure 3.3) consists of several mechanical parts, including three 3D printed platforms, five threaded rods (in the figure is depicted to be unthreaded), a thrust ball bearing, and three couplings. The material for the 3D printed

components are from the family of polymers. The threaded rods are 8 mm in diameter with stainless steel material.

The platforms are used to hold the motors and the robot head-ear-motor-rod-coupling load. The author uses a coupling to take this load so that the bottom motor can maximize its power for object rotation. A bearing is put under the load to reduce rotational friction. The author utilizes threaded rods as a pillar for the platforms. Threaded rods are widely available and easily attached to the platform by using nuts. Furthermore, they make the height of the robot adjustable.

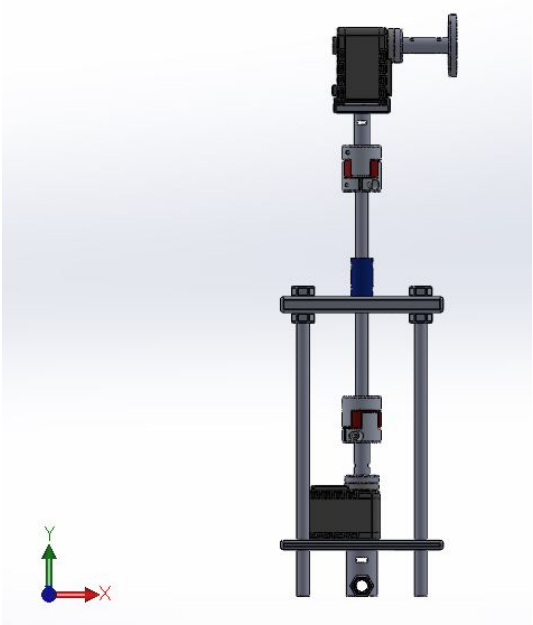

Figure 3.3: Pan-Tilt System

Trossen robotics PhantomX Pan Tilt inspired the author's pan-tilt system. Like that of the PhantomX, two Dynamixel servo motors are used to move the author's robot head. An interface controls the motor movement: Usb2dynamixel for one servo and U2D2 for another. The author's assign the robot motions through the Roboplus Dynamixel wizard software.

The top motor is connected to the right of the head, which is joined to the left side. It adjusts the head elevation. When the motor rotates, the microphones only vary in its orientation but not position as its axis inlines with the rotational axis.

A sound source is put on top of a platform to make it has the same height as the robot head. Four threaded rods support the platform. The author approximates that if the robot head and the speaker center points are connected, they form a horizontal line that is parallel to the ground.

The range for Azimuth movement is 0 to 360 degree, while for elevation are about -90 to 90 degrees. A 3D printer manufactured the two artificial ears, robot heads, and platforms. The link to the complete mechanical drawings of the robot are included in the Appendix.

3.2 Audio Devices

Each robot ear is equipped with a Boya omnidirectional microphone. This microphone can capture the sound uniformly in different orientations and has size and shape that fit the author's ear design.

A Behringer audio interface plugged with the two mics is connected to a laptop for recording purposes. The author uses Audacity software for the recordings and a JBL Bluetooth speaker as the sound source.

3.3 Dataset

The drawback of using the available dataset is that we may not be able to add more data, as there is a big chance that our environment is different from the environment in which the datasets are collected. In this project, we collect the dataset using our own robot. The reason behind this are: (1) The flexibility to add more data. (2) More opportunity to improve the work for future projects. (3) Contribute to the field of deep learning in supplying dataset. This section explains the dataset that the author collects.

3.3.1 Position Selection

The distance between the speaker and the head is kept fix by a 1 m threaded rod that spans from the speaker platform to the head platform. The author has a stationary sound source and a moving head. So, elevation and azimuth angles are formed by the head rotation. Figure 3.4 and **??** respectively show the robot and speaker configuration and the rotational convention.

Figure 3.4: Robot and Speaker

There are 15 data points for training and validation set, which is a combination between five azimuth angles (-60,-30,0,30,60 in degrees) and three elevation angles (-30,0,30 in degrees). The test set not only contains the 15 data points from the same position as the training and validation set but also 12 more data points to check the networks ability in generalization.

3.3.2 Recording Session

The recordings are done using a single sound source. The sound emitted from the source is a 6 kHz pure tone or sinusoidal waveform. The frequency selection is motivated by Hwang et al's paper (Hwang, Park, and Park, 2011), where the ear design also gained its inspiration. In the article, they explained that the system started to notice an elevation difference when the frequency was 6 kHz and above (Hwang, Park, and Park, 2011). The higher the audio frequency, the higher the sampling rate required to make the signal representable. So 6 kHz is still well represented by the recording device's sampling rate of 44.1 kHz and conform to Shannon's sampling theorem.

Claude Shannon formularized that the minimum rate at which a signal needs to be sampled is twice the signal's highest frequency. If not, the original signal would not be adequately recovered or reconstructed.

3.3.3 After recording summary

The dataset totals in about 1 hour of recording or 834 MB in Wav format. The sound was recorded with 16-bit audio, two channels, and a sampling rate of 44.1 kHz. The recording was performed in the author's flat during the daytime. The dataset contains a training set, a validation set, and a test set. The details about each set are:

- Training Set : The training set consists of recordings of 15 different positions. Each position is represented by an audio file of 162 seconds in duration, totals in 40 minutes and 30 seconds audio file.
- Development/Validation Set: Similar to the training set, the validation set contains 15 different position. Each position has a total audio files duration of 18 seconds, therefore the validation set consists of 4 minutes 30 seconds file.
- Test Set : The test set contains 27 position: 15 for accuracy test, and 12 for the generalization ability. Each position has a duration of 30 seconds, creating a total of 13 minutes 30 seconds audio file.

3.4 Audio Processing

Using signal processing techniques, we create two sets of features: mel frequency cepstral coeffient and log filter bank. MFCC is one of the most common representation of audio data used in deep learning based audition, examples including studies done in (He, Motlicek, and Odobez, 2018) and (Dai Wei et al., 2016). Another audio representation, log filterbank energies is selected to see whether the type of input representation affect the network performance. The author use Python Speech Feature library to transform the raw audio file into both MFCC and log filter bank forms.

- Log filterbank energies: For each audio file, the author uses Log Filter Bank windows of 0.1 second. The window is moved every 0.05 second, creating a test and validation set of 54000 clips and a test set of 16200 clips.
- Mel Frequency Cepstral Coefficient (MFCC): As for Log Filterbank, the authors takes MFCC over windows of 0.1 second. The window is also moved every 0.05 second, creating a test and validation set of 54000 clips and a test set of 16200 clips.

All features are normalized to have zero mean and unit standard deviation so that the gradient descent will converge faster. Each audio file in the dataset is split into a 1-second duration file. The likelihood of a class label for an audio file is the sum of the predicted class likelihood for each clip (Dai Wei et al., 2016). The class with the highest total probability becomes the predicted label for the file (Dai Wei et al., 2016).

3.5 Neural Networks

The author views the SSL problem as a single class classification problem, in which a data can only belongs to a class. There are fifteen possible classes for each data, which consists of combinations between three elevation angles and five azimuth angles.

3.5.1 Architectures Selection

The types of neural networks architectures that the author employs and compares in this study are Multi-Layered Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Bidirectional Neural Networks (BRNN).

The author selects MLP because it is the most natural form of neural network. It is straightforward to implement and usually faster to converge than other architectures. MLP has also shown a good performance in sound scene recognition task (Dai Wei et al., 2016).

RNN and BRNN are chosen because they are mainly used to deal with sequential data such as a sound clip. The type of RNN and BRNN the author select is the Long Short Term Memory (LSTM). CNN is selected because in the works done by (Dai Wei et al., 2016) it outperforms other architecture for audio scene recognition and classification task. Therefore, the author considers that a similar result might occur in performing SSL task.

Furthermore, these networks are among the most popular in the current deep learning research. The complexity of these networks is relatively lower than let's say of that Spiking Neural Network, that this study, with its limited timeframe, can covers.

When using CNN, the MFCC or log filter bank as seen by the network as an image, so the convolutions are performed by 2D filters. For RNN and BRNN, the input is recognized as sequential data, where order matters. An audio file is a sequence taken at successive equally spaced points in time. RNN and BRNN are build to process sequential data, because the networks pass the information of the data past (and next, for BRNN) states to its neuron. In MLP, the sequential data is flattened out before fed into a regular feedforward network.

3.5.2 Hyperparameter Tuning

The manner in which neural network hyperparemeters are tuned is almost empirical. Hyperparameter tuning is one of the most time-consuming aspects of this project. There are numerous tuning knobs: the number of layers, the number of neurons, activation functions, dropout units, regularization parameters, and optimization algorithms (Dai Wei et al., 2016).

The motivation of the hyperparameter tuning is usually to find the lowest generalization error subject to some runtime and memory budget (Dai Wei et al., 2016). If our model is too simple, it may have

high bias and low variance. Vice versa, if it is too complicated, it may have high variance and little bias. So, we need to find the right balance to avoid overfitting or underfitting the data.

Due to time constraints, only some hyperparameters are tuned in this project: number of neurons, number of layers, batch size, dropout rate and frequency, and number of epochs. The author chooses minibatch gradient descent with Adam for the optimizer and dropout for regularizer.

The rule of thumb in tuning the neural network is to change only one hyperparameter at a time before each training until it gets an optimum value and we move on to another hyperparameter. This rule avoids the confusion in deciding which of the hyperparameter corresponds to the performance.

The hyperparameters that are not tuned by the author and used uniformly across all networks are:

- Activation Functions: Each network use Rectified Linear Unit (ReLU) as its activation function. Before the discovery of ReLU, the popular activation functions are Sigmoid and Tanh. ReLu becomes more popular because it is computationally cheap and performs well on standard feed-forward network (Goodfellow, Bengio, and Courville, 2016).

- Gradient Descent: The author uses Minibatch Gradient Descent for network optimization. MBGD lies in the middle between Batch Gradient Descent (BGD) and Stochastic Gradient Descent (SGD). It splits the training dataset into small batches to calculate model error and update coefficients. Therefore it is faster than SGD and more accurate than BGD. The batch size is not uniform for each network and is tuned to get better performance.

- Optimizers: Adam is computationally efficient and straightforward to implement. Adam combines the virtue of momentum and RMSprop.

- Regularizer: Dropout is used for regularitation. It is simple and straightforward to implement. The dropout rate and frequency are tuned in each network.

- Validation Technique: The author use autovalidation in which a certain portion of the training set is used as validation set. Compare to k-fold cross-validation, auto validation requires less times and is reasier to implement

- Loss Function: Categorical cross-entropy acts as the loss function. The loss function judges the network's performance on classifying the audio data. Categorical cross-entropy applies for single class categorization or where an example can belong to one class only.

3.5.3 Model Selection

The model for each network architecture is selected by hyper parameter tuning and auto validation. The tuning are done manually by the author to find the best validation accuracy. The auto validation is

performed by the network by assigning a fractions (in our case is 10%) of the training set as validation set and optimizing the network based on validation accuracy metrics.

The author sums up each network hyper parameter tuning as:

- MLP: MLP required the most number of epochs to converge. It uses a small number of dropout rate (0.2 of the training data is dropped). It was better to put the dropout layer after each layer other than right before the output layer.
- CNN: The author uses four convolutional layers, two maxpooling layer, and a softmax layer for classification. CNN only needs less number of epochs to get appropriate accuracy. The author used 3x3 Keras conv2d filters with strides 1 and "same" padding.
- RNN: The author uses LSTM (Long Short Term Memory) instead of GRU (Gated Recurrent Unit). The author used 4 layers of LSTM layers. It is better too put the dropout rate at the end of each layer rather that before the output networks.
- BRNN: The bi-directional RNN has more temporal dynamics than RNN. There are 4 layers of LSTM layers and a softmax layer. BRNN and RNN required almost the same number of epoch to converge.

In the end, the author ensembles all the models mentioned above. There are four models use for the two input representations. Table 3.1 summarizes the structure of each network models.

Table 3.1: Architectures of each network models

MLP	CNN	RNN	BRNN
Dense 256	16x3x3	LSTM 256	Bi-LSTM 256
Dense 256	32x3x3	LSTM 128	Bi-LSTM 128
Dropout(0.2)	MaxPool(2,2)	Dropout(0.2)	Dropout (0.2)
Dense 128	Dropout(0.2)	LSTM 64	Bi-LSTM 64
Dense 128	64x3x3	LSTM 32	Bi-LSTM 32
Dropout(0.2)	128x3x3	Dropout(0.2)	Dropout (0.2)
Dense 64	Dropout(0.2)		
Dense 64			
Dense 32			
Dense 32			
Softmax 15			

The dropout rate in the table follows the Keras convention. This may be different from the definition of dropout rate from some sources, in which the rate refers to the probability of retaining an input. In

the author's case a dropout rate of 0.2 means that the network drops 0.2 (20%) fractions of it and retains the 0.8.

3.6 Programming Language and Computers

Python with Numpy, Tensorflow, and Keras frameworks is used to build the Neural Networks. These frameworks make building network easier without losing any significant flexibility to change the parameters. The author build the network models on an HP nvy Laptop with 6 GB Ram and Intel Core i7 node.

The project codes are available in the author's GitHub page. A link to the page is included in the Appendix.

3.7 Performance Metrics

- Accuracy and Generalization Accuracy: As mentioned earlier in section 3.3, during training each audio file (duration 1 second) is split into 0.1 second clip with a window step of 0.05 second. Means that every 1 second audio file makes 20 audio clips. The likelihood of a class label for an audio file is the sum of the predicted class likelihood for each clip. The class with the highest total probability becomes the predicted label for the file. To measure the accuracy of the neural networks, the number of accurate file predictions are divided by the total number of file.

- Time and Memory Cost: The time required to train the network and the trained model's file size are considered as the time and memory cost.

3.8 Summary of Methods

To conclude, there are three main components of the robots: a pan-tilt mechanism, audio devices, and software. The ear and head are based on literature review findings in (Hwang, Park, and Park, 2011), while pan-tilt system is inspired by Trossen PhantomX pan-tilt.

The author uses the robot to collect sound data in several robot orientations. The recording is done using two microphones, an audio interface, and a speaker with specification explained in Section 3.3. Using signal processing technique the author converts the audio file into two representations: MFCC (Mel Frequency Cepstral Coefficient) and log filterbank energies. These representations are provided as inputs to each networks.

The neural networks architectures that the author builds are MLP, CNN, RNN, and BRNN. Python, Tensorflow, and Keras are use to create the neural networks program. The neural networks types are

MLP, CNN, RNN, and BRNN. The link to the components mechanical drawing and the networks code are included in Appendix.

4 Results

Table 4.1 and 4.2 respectively show the training and test accuracy for the four neural network architectures over two features. The accuracy is split into its directional components, which are the elevation and azimuth angles.

Table 4.1: Train accuracy of each network

Network	Feature	Combined
MLP	MFCC	85.96%
	Log Filterbank	89.47%
CNN	MFCC	82.72%
	Log Filterbank	84.5%
RNN	MFCC	87.21%
	Log Filterbank	89.22%
BRNN	MFCC	86.72%
	Log Filterbank	88.98%

Table 4.2: Test accuracy of each network

Network	Feature	Elevation	Azimuth	Combined
MLP	MFCC	65.41%	98.53%	66.89%
	Log Filterbank	67.54%	99.12%	68.29%
CNN	MFCC	61.90%	98.72%	62.21%
	Log Filterbank	55.74%	98.73%	56.28%
RNN	MFCC	63.26%	99.2%	63.32%
	Log Filterbank	65.40%	98.46%	66.37%
BRNN	MFCC	64.48%	99.21%	65.93%
	Log Filterbank	64.36%	99.11	65.75%

The combined accuracy (last column) represents the performance where both direction – azimuth and elevation – are estimated. It is not the average between azimuth and elevation accuracy. In this project, the networks are trained to classify both azimuth and elevation angles at the same time, means that they regards the location classification as a single task (azimuth and elevation combined) instead of two different task.

The neural network's hyperparameter that the author tune are the learning rate, number of layer, number of neurons, dropout rate and number of training epoch. The author has summarized the selected hyperparameter at section 3.

The networks are trained for several number of epochs, depending how fast it converges. The selected models are those that have the highest validation accuracy. If, for example, a network achieves a validation accuracy of 80% at epoch number 5, and 75 % after epoch number 6, the system will keep using the one from number 5 an will not update the model until an improvement occurs in the next epochs.

MLP converges within 469 epochs for MFCC and 938 using log filterbank. An interesting phenomena happens during the network training as displayed by figure 4.1 and 4.2. The validation set loss went up drastically when the epoch is at 500 and after that the loss starts to get smaller even though never reach a better performance than when it reach epoch 469. A similar phenomenon is observe when training with log filter bank inputs.

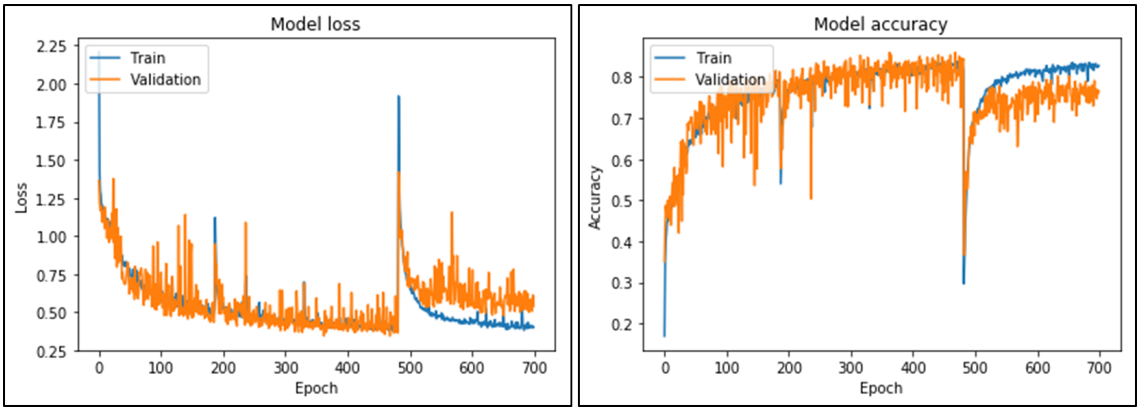

Figure 4.1: MLP Loss 1

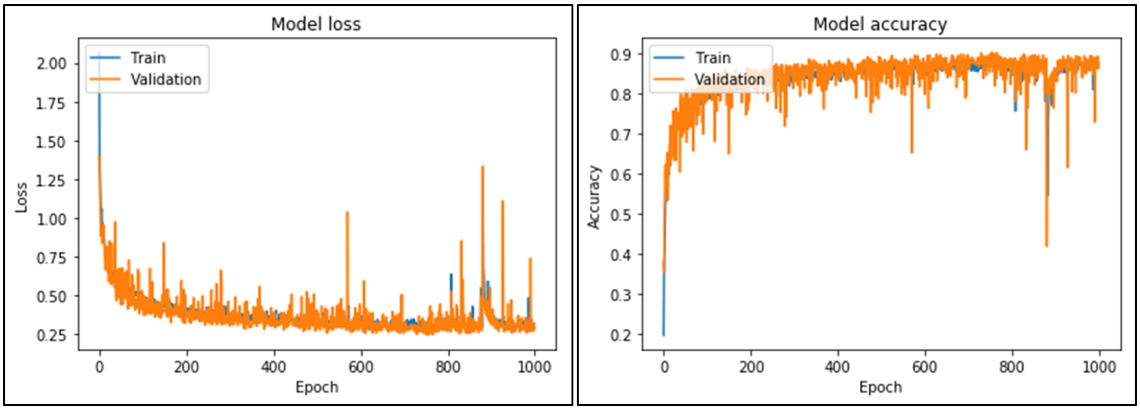

Figure 4.2: MLP Loss 2

CNN converges within a few numbers of epoch: 34 for MFCC and 38 for log filterbank energies.

RNN converges after 84 and 88 epoch for MFCC and log filterbank energies respectively. BRNN converges after 91 epoch for MFCC and 87 epoch when using log filterbank energies representation.

After selecting the models and employing it to the test set, the author looks for the class where most misclassification occurs. Figure 4.7 to 4.10 display the confusion matrix for each type of neural networks and input features combination. This visualization inform us the classification performance for each position and help us in figuring out the important properties of the model.

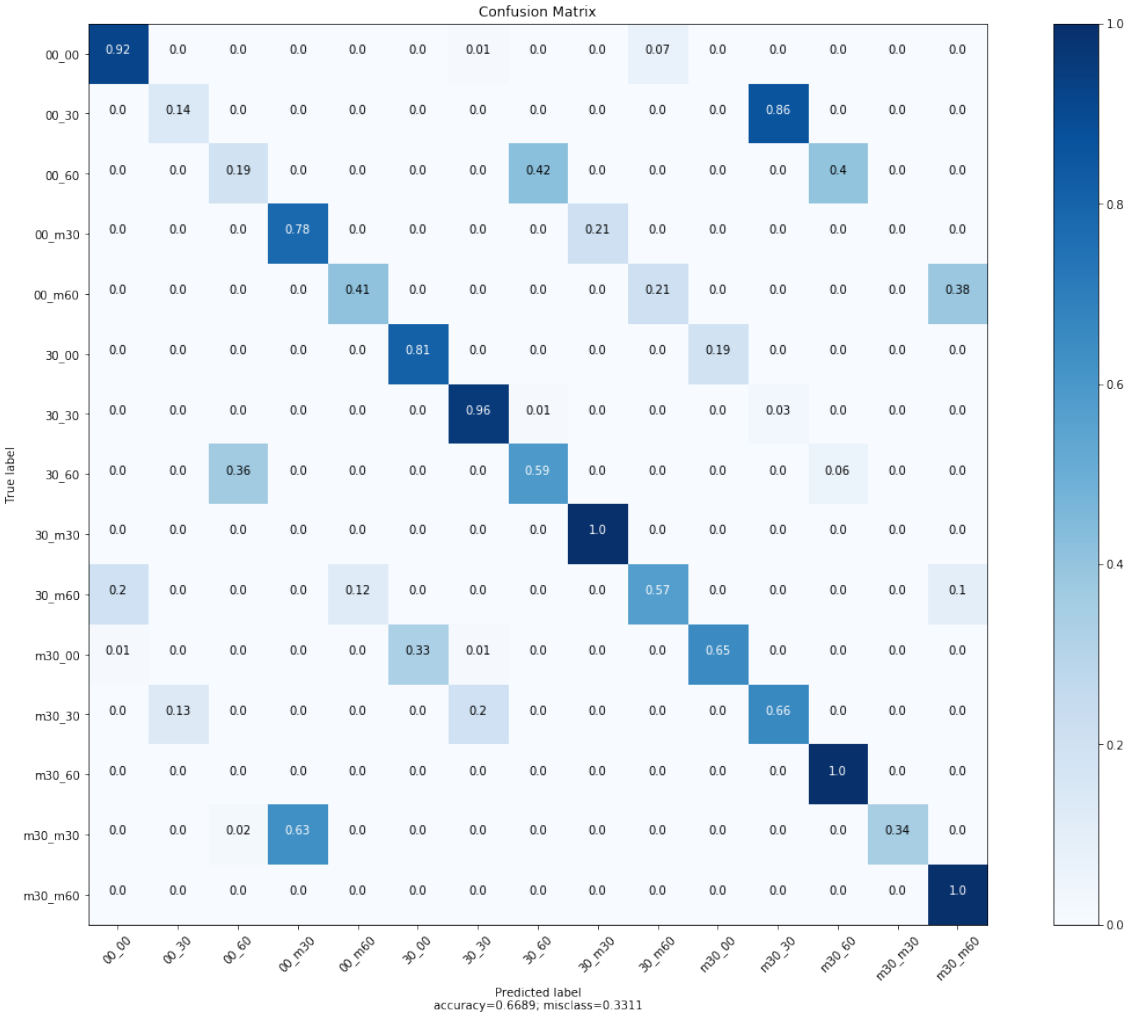

Figure 4.3: MLP with MFCC Confusion Matrix

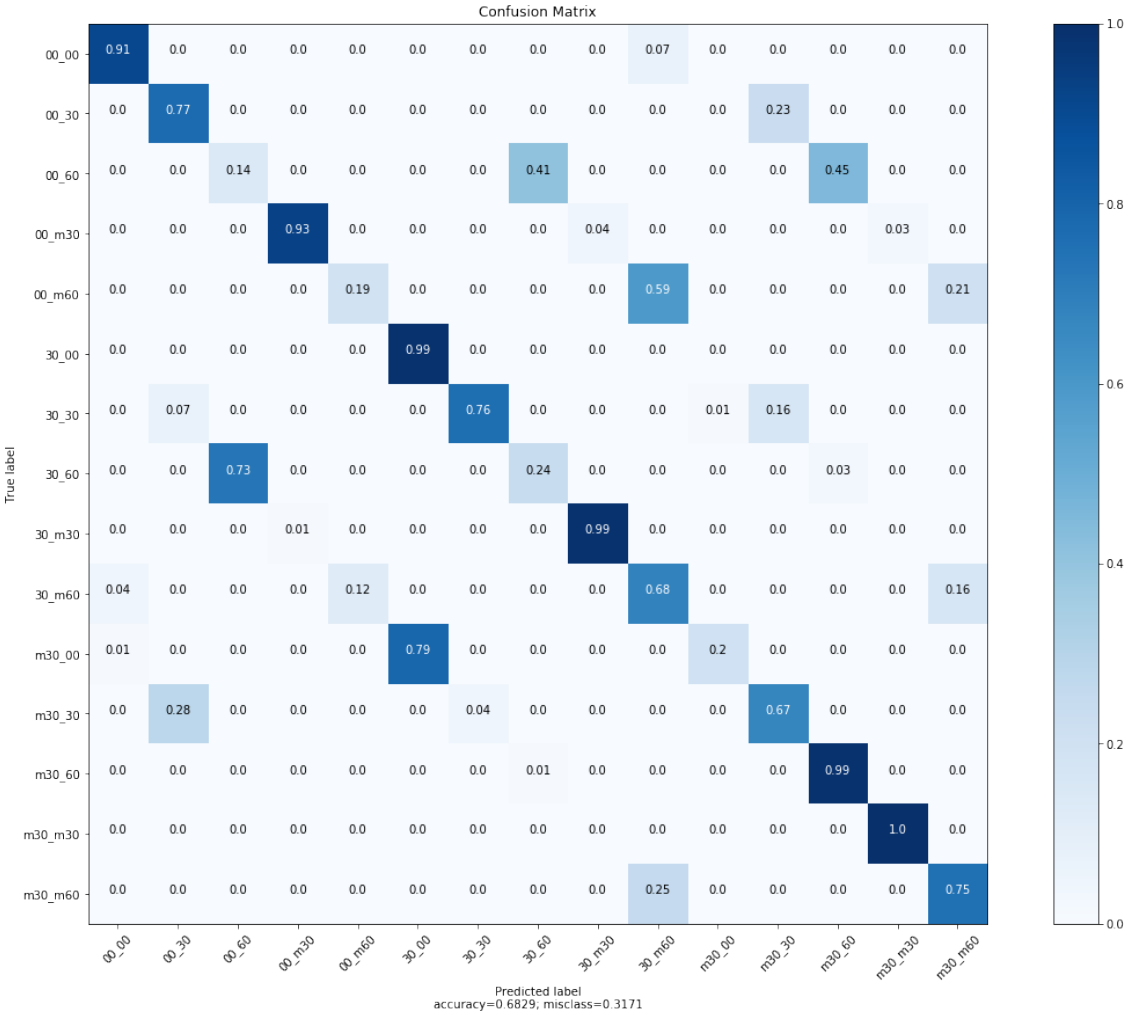

Figure 4.4: MLP with Log Filterbank Energies Confusion Matrix

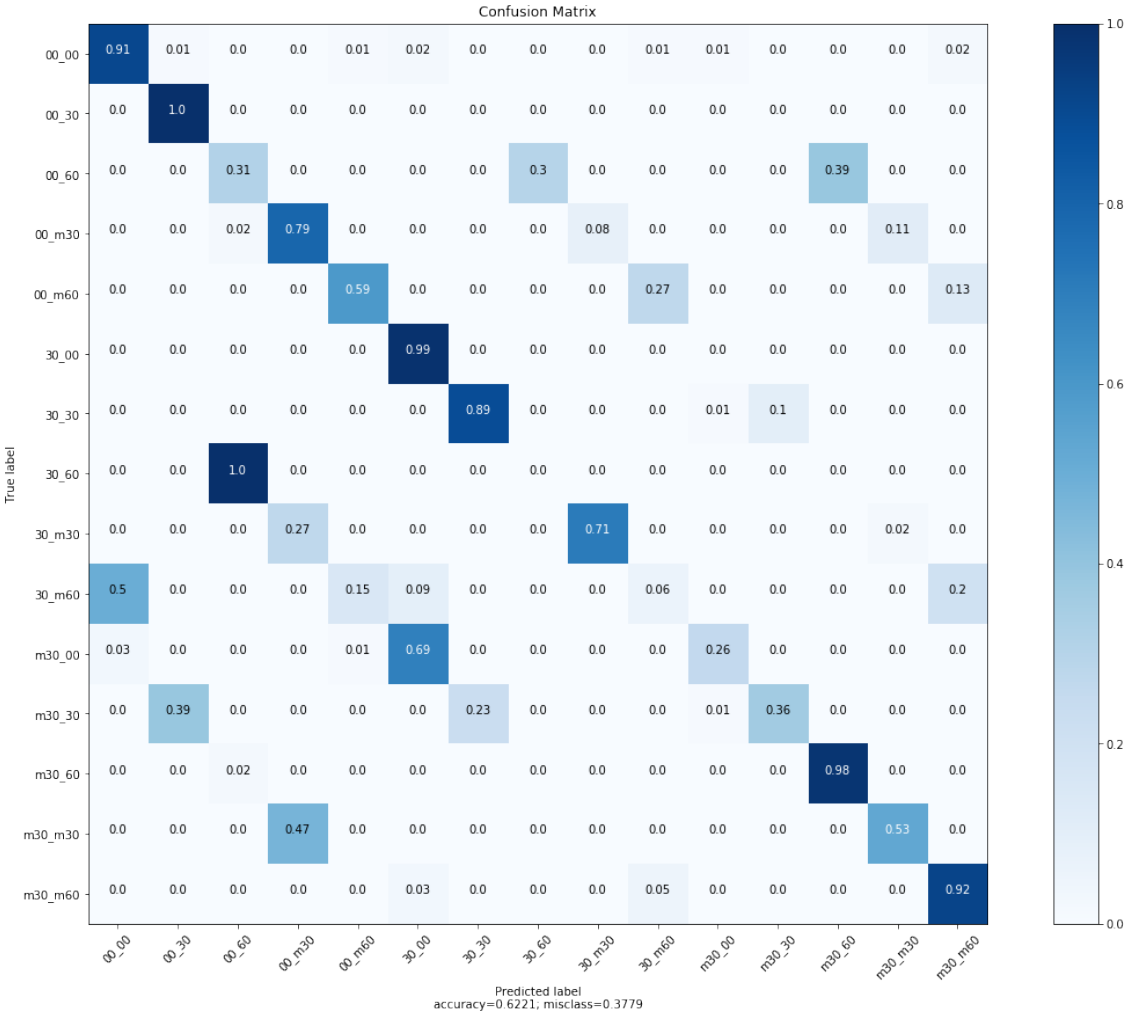

Figure 4.5: CNN with MFCC Confusion Matrix

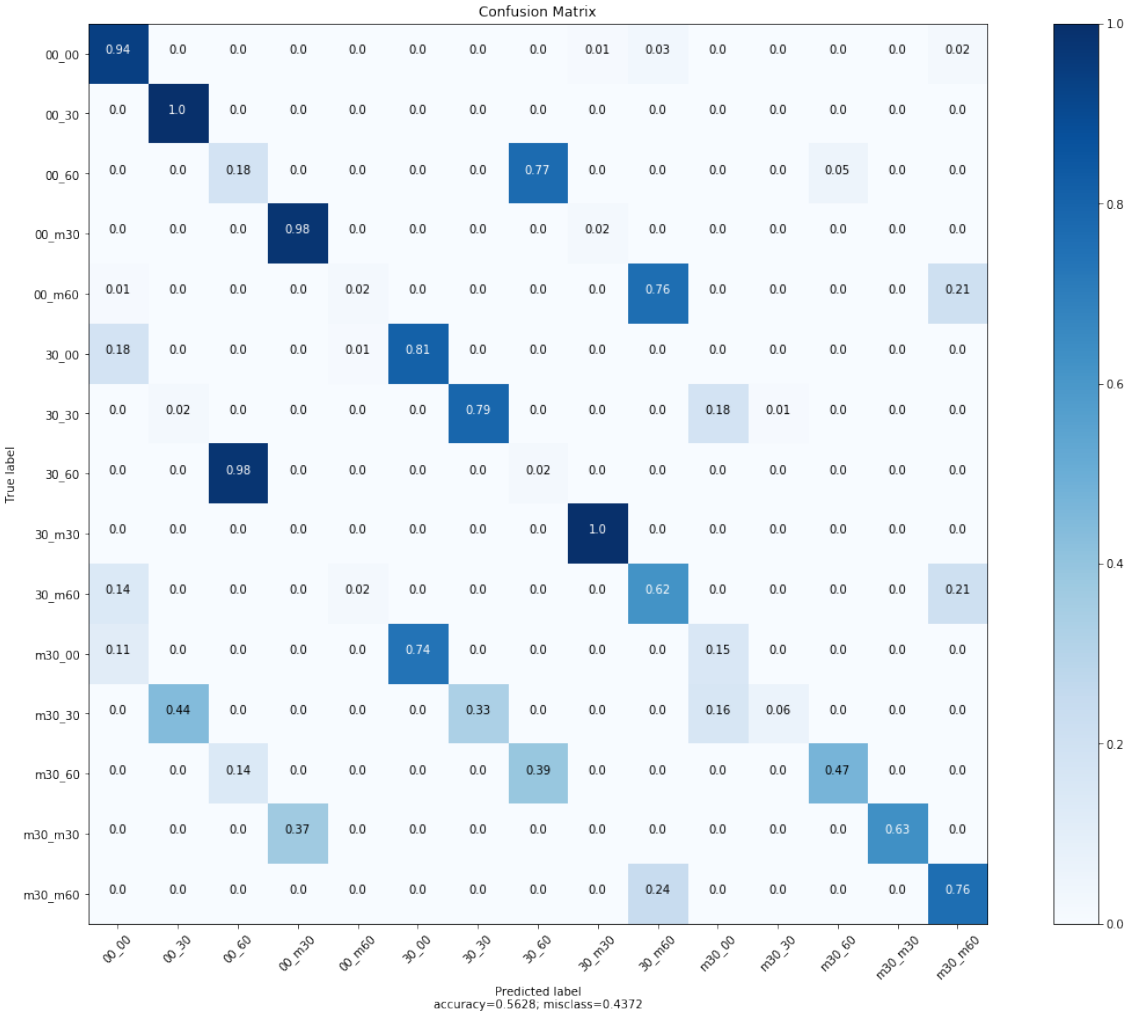

Figure 4.6: CNN with log filter bank energies Confusion Matrix

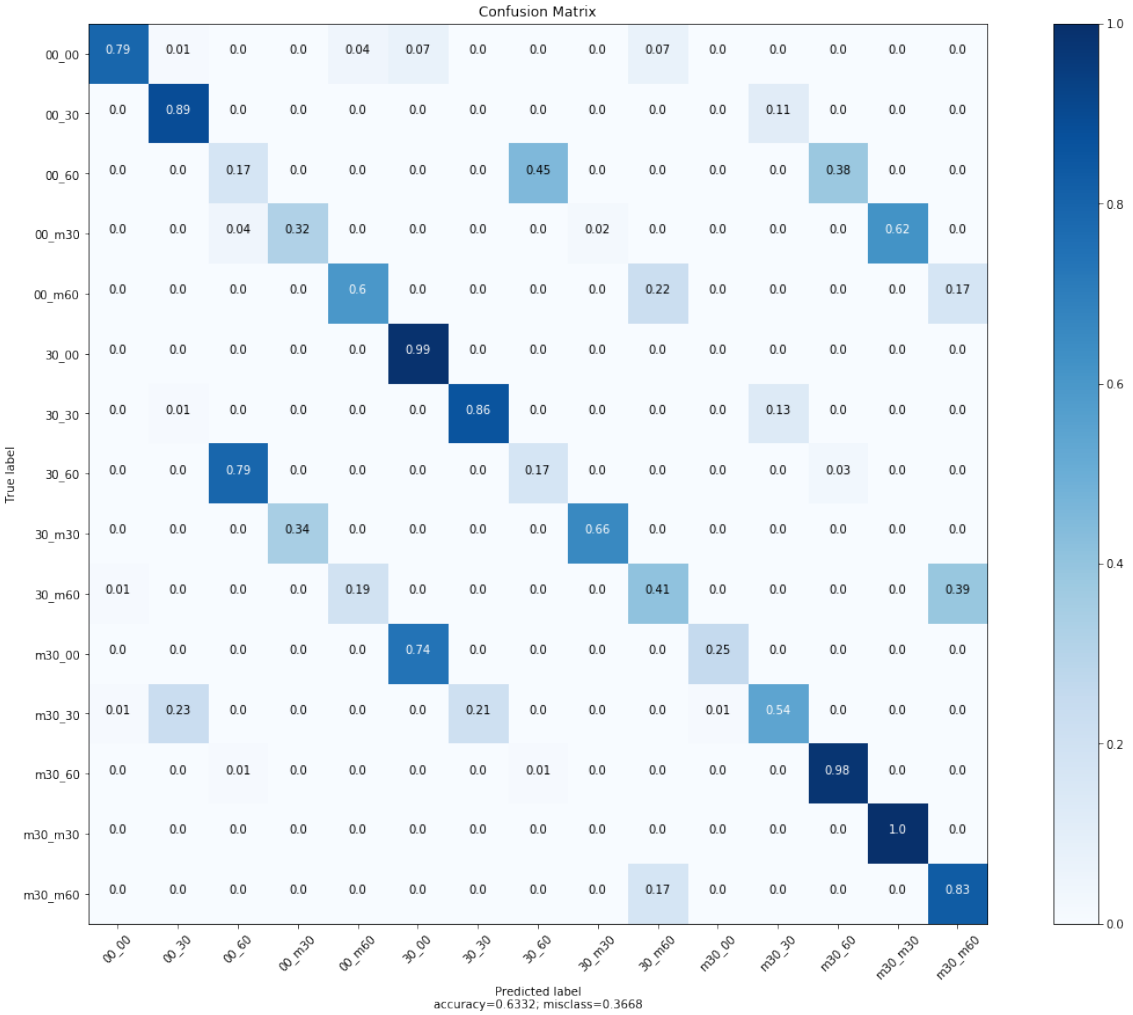

Figure 4.7: RNN with MFCC Confusion Matrix

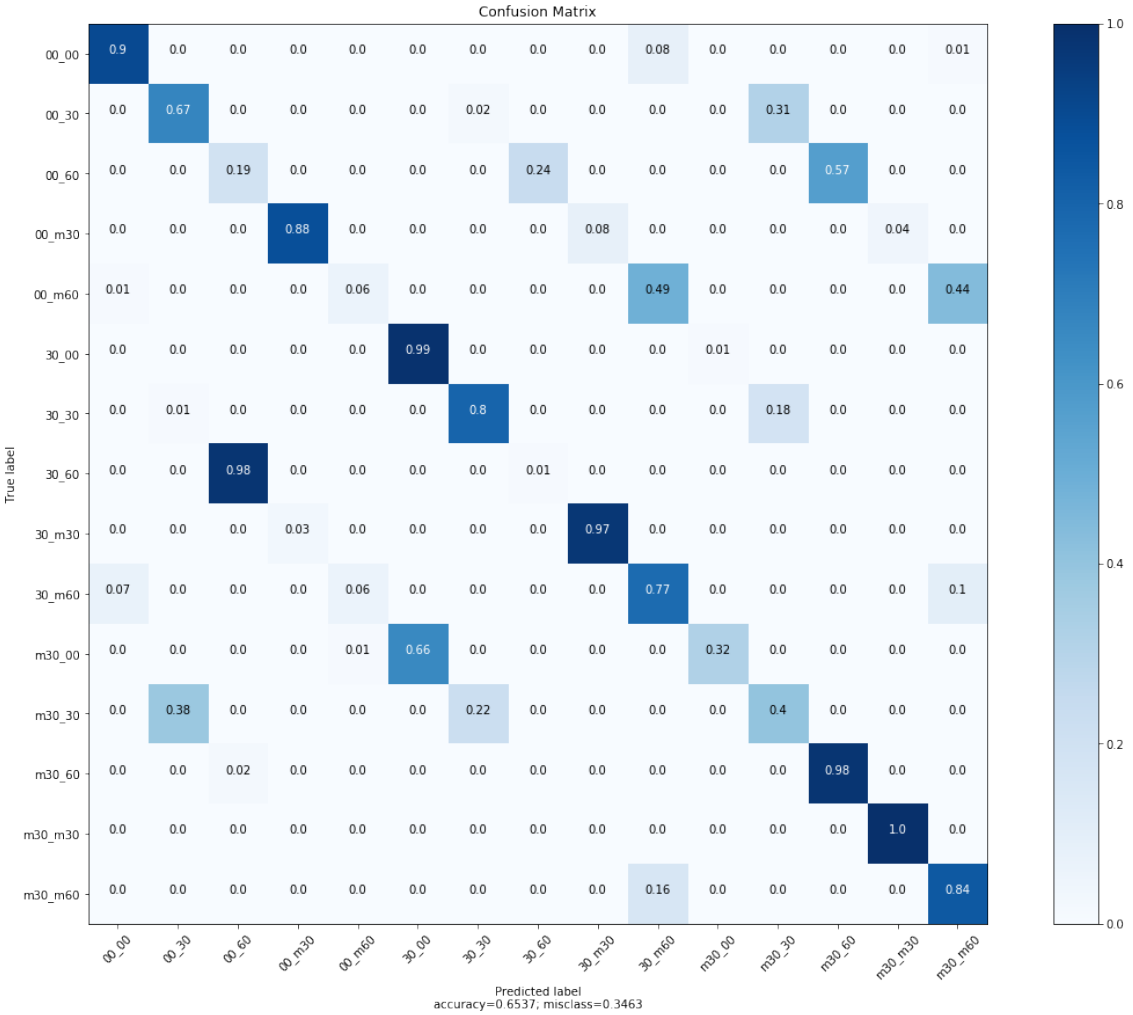

Figure 4.8: RNN with Log Filterbank Energies Confusion Matrix

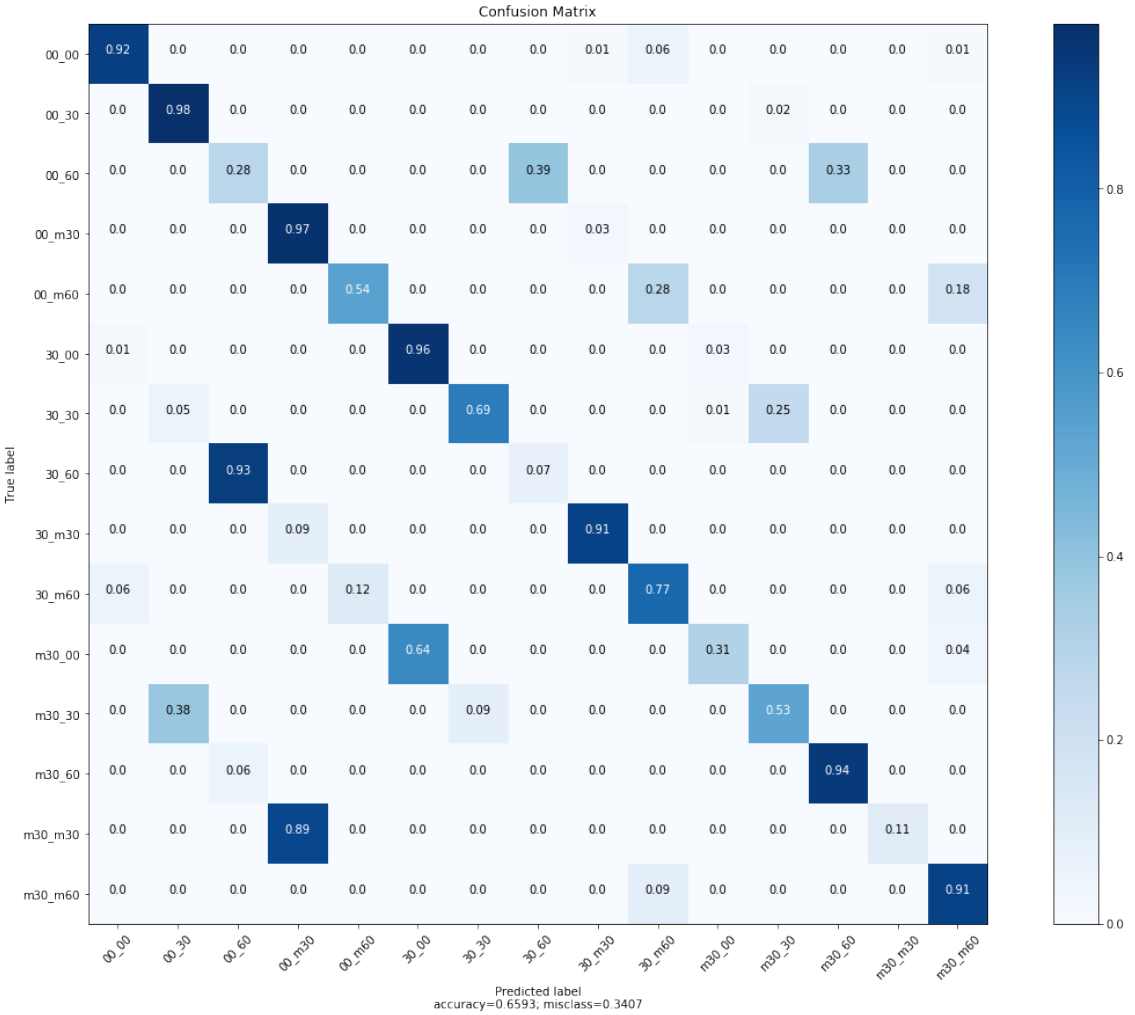

Figure 4.9: BRNN with MFCC Confusion Matrix

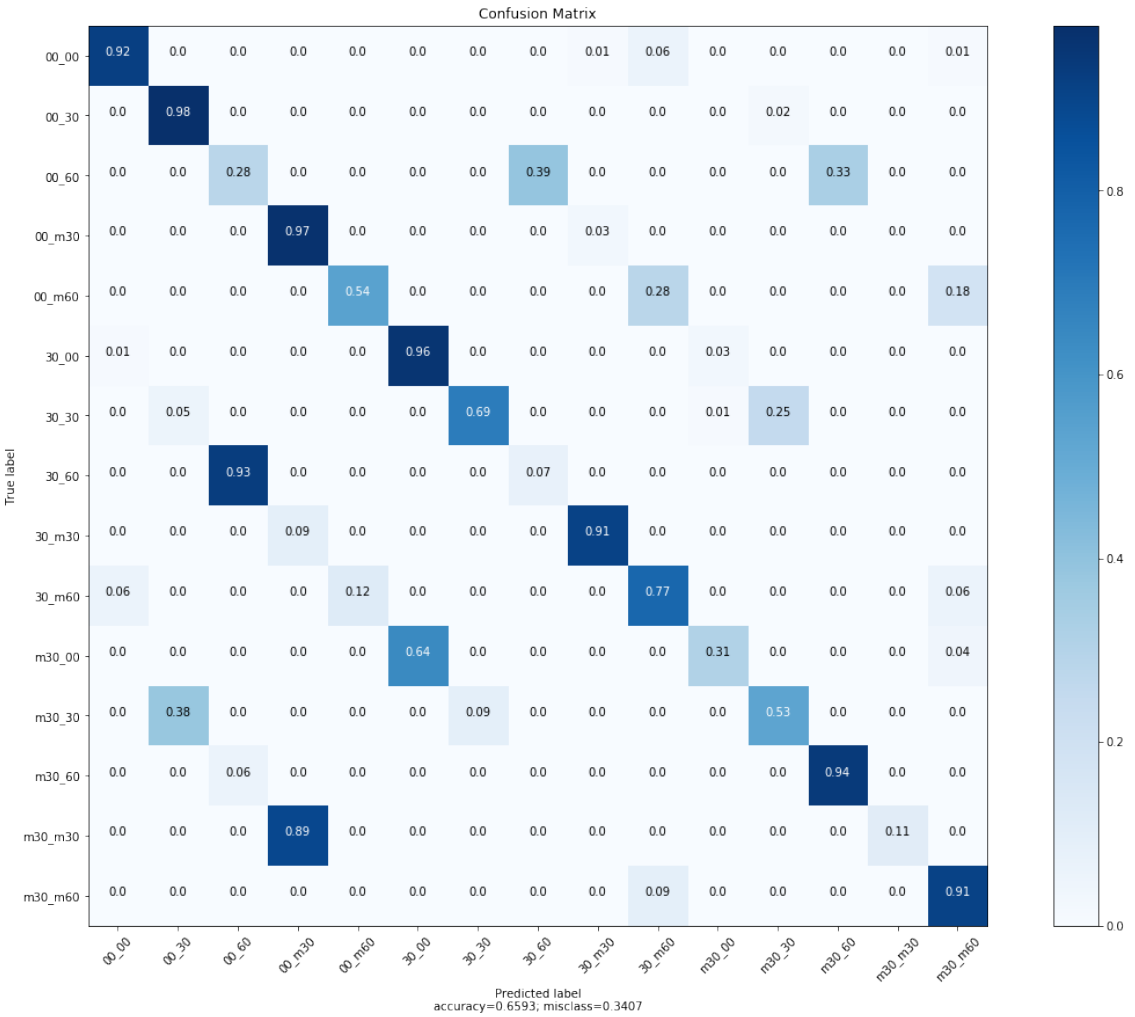

Figure 4.10: BRNN with log filter bank energies Confusion Matrix

Another performance metrics that the authors want to see is how well the network performs when exposed to sound data of an untrained position. As explained in the methodology, the author collect data in a set of positions that is located between the trained position. This dataset is used to see whether the network will classify the new position to its nearest trained position. For example, if we have trained the network in the positions of 0 degrees and 30 degrees, and then we feed the network with a test position of 25 degrees, a good network will classify it as 30 degrees. Table 4.3 displays the generalization accuracy for four neural network architectures over two features.

The generalization data is not visually fit the confusion matrix, as there are 12 untrained positions that generalizes to 6 trained position. Therefore, the author represent the error only by a table.

After training the models, the author obtains the time and memory consumption of each neural network models. Memory in neural networks is required to store input data, weight parameters, and activations as an input propagate through the network (Hanlon, 2019). In training, activations from a forward pass must be retained until they can be used to calculate the error gradients in the backward

Table 4.3: Generalization accuracy of each network

Network	Feature	Combined
MLP	MFCC	19.22%
	Log Filterbank	23.98%
CNN	MFCC	25.61%
	Log Filterbank	26.78 %
RNN	MFCC	22.40%
	Log Filterbank	23.39%
BRNN	MFCC	22.40%
	Log Filterbank	24.1%

pass.

Table 4.4 displays the time and memory consumption of each type of neural network architecture. The time cost describe how long it takes to train the neural network models while the memory indicates the file size of the built model. In real deep learning deployment, time and memory are usually used as constraints in selecting the neural network models.

Table 4.4: Time and Memory Consumption for each models

Network	Feature	Time	Memory
MLP	MFCC	1802 s	2.35 MB
	Log Filterbank	3552 s	3.09 MB
CNN	MFCC	2382 s	2.86 MB
	Log Filterbank	6736 s	3.32 MB
RNN	MFCC	3342 s	6.43 MB
	Log Filterbank	9616 s	6.58 MB
BRNN	MFCC	9562 s	4.42 MB
	Log Filterbank	12622 s	4.58 MB

5 Discussion and Conclusion

This chapter presents discussions on the results explained in the previous chapter and the conclusion of the project. The discussion part provides examinations of each performance metrics, followed by the author's ideas for future project. The conclusion part summarizes the project and gives a brief account on the future project ideas as well as the author's contribution.

5.1 Discussion

Three performance metrics, as indicated by each subsection titles, are compared and analyzed.

5.1.1 Accuracy

Based on the results from Table 4.1 and Table 4.2, MLP has the best test set accuracy. This architecture achieves an accuracy of 66.89% when exposed to MFCC, and 68.29% while using log filterbank energies. This accuracy comes with a comparatively lower memory and time cost, as explained in chapter 4. A similar result was obtained by (Dai Wei et al., 2016) though in an acoustic scene recognition. In their study, MLP outperforms CNN and RNN in inferring the place where the sound is recorded.

The author does not expect this to happen. In (Dai Wei et al., 2016) study, it is found that MLP has a limited temporal dynamics that might cause a problem in dealing with sequential data such as speech. (Dai Wei et al., 2016) explains in his study that unlike speech which has a long range dependency, his environmental sound data set lacks a coherent context. It might be that the pure tone is somehow similar to enviromental sound that makes MLP perform best in this and (Dai Wei et al., 2016) cases.

At first, the author expects BRNN and RNN to perform the most accurate prediction, as they are specialized in handling sequential data such as sound. BRNN achieves an accuracy of 65.93% with MFCC and 65.75% using log filter bank energies. RNN has a better accuracy tha BRNN when using log filterbank Energies, but not when using MFCC. CNN has the least accurate architectures in comparison.

Overall, the variation of input has not indicated a significant difference. In some cases MFCC corresponds to more accurate predictions, but in other cases log filter bank energies performs better. However, there are many audio file representation that the author has not used. The author believes that preprocession can hold a crucial role in optimizing neural network performances, especially when the network size is small (Dai Wei et al., 2016).

From the confusion matrix we can see that elevation misclassification happens in a different position in each network. For example, when using MLP with MFCC most of the classification at elevation 0

and azimuth -60 are wrong, but if we use log filter bank the same position is estimated a lot better. This proof that the networks model that the author build is not yet optimum and there are rooms for future improvement in choosing it.

It is clear that elevations are much harder to locate than azimuth. As stated by Rayleigh (Yost, 2017) and Wei (Dai Wei et al., 2016), sound localization depends mainly on the inter-aural level difference (ILD) and inter-aural phase difference (IPD) in estimating azimuth. Elevation estimation could not depend on these features, since the elevation rotation axis is in line with the microphone axis; thus, a rotation in elevation only change the mic orientation and not position, and without a change in position ILD and IPD are essentially the same for every position.

Since the main advantage of neural network is that it reduces the human need to interpret or model a function that produce a certain output, it is hard to go further onto why our system performs better in determining azimuth, other than cause by ILD and IPD.

The elevation needs to depend mainly on the spectral modification of the sound. The author design a pair of ear to do that. As (Hwang, Park, and Park, 2011) explains, their ear starts to notice elevation difference when the sound is 6 kHz and above. the higher the frequency, the more accurate the elevation estimation. So, the author's elevation estimation may be more accurate if the author uses a higher sound frequency.

In analyzing training and test data we usually discuss the term bias and variance. The ideal is to have both low bias and low variance. In our network, it is seen that there is a large gap between training and test accuracy. The author only use dropout for training. Another type of regularization technique might address this challenge.

To further clarify how well the network generalize, the test data should come from different distributions. In the next section, the author discuss the networks performance in classifying "new" position.

5.1.2 Generalization to Nearest Point

One of the significant advantages of neural networks is their ability to generalize new data. It means that a trained network could classify data from the same class as the learning data that it has never seen before. In this project, the authors prepares dataset in locations different from those during the training. The goal is to see whether the neural networks can locate this new data to their nearest position. Table X displays the generalization ability of each network.

CNN achieves the highest accuracy, with a value of 25.61% when exposed to MFCC, and 26.78% with log filterbank energies. MLP achieves the second-best generalization accuracy. BRNN outperforms RNN in terms of generalization ability, though it comes with higher memory and time cost.

Unlike in test accuracy, it can be seen that the type of input affects generalization ability. The networks using log filterbank performs better generalization accuracy than when dealing with MFCC.

There are several factors that improve generalization ability, including type of regularization, num-

ber of training set, and data augmentation. In this work, we choose dropout for each type of networks. What we tuned is the dropout rate.The higher and the more frequent we perform the dropout, the better the network generalization, but with less accuracy. It is therefore the tradeoff between generalization ability and accuracy that we tune.

5.1.3 Time and Memory Efficiency

Memory in neural networks is required to store input data, weight parameters, and activations as an input propagate through the network. In training, activations from a forward pass must be retained until they can be used to calculate the error gradients in the backward pass.

Table 4.3 displays the time and memory consumption of each type of neural network architecture. Overall, MFCC models are computationally cheaper than log filterbank energies. The author use the same window size and step for both feature representation, so the cost should relate solely to the features types.

From the networks perspective, RNN has the memory consumption. It is surprising that RNN takes more memory than its bidirectional counterparts. As the name suggests, in BRNN, an input feature has information about its past and next state thus has a more temporal dynamics.The type of BRNN and RNN type that the author uses are Long Short Term Meory (LSTM). Both BRNN and RNN memory cost are caused by the sequential information that is preserved in the recurrent network's hidden state. However, in the author case BRNN equires more time to training.

CNN has the third largest consumption. The memory in CNN is heavily affectd by the convolutional layer. In CNN, a single feature may be read by multiple filters at once, and the filter stride make it possible that the point that are already read in its previous position is read once again in the next. MLP has the least memory consumption among the models. It is interesting to see that MLP, though trained with most epoch still comes out to be the most accurate models.

5.2 Conclusion

The author presents a comparison of four neural networks architecture in their performances in localizing sound in both azimuth and elevation angle. The localization is performed by a pan-tilt robot head, equipped with a pair of artificial ears. Several works inspired the robot design, including (Hwang, Park, and Park, 2011) for the ear and (Robotics, 2019) for the pan-tilt system. The robot is used for dataset collection. The dataset contains 15 different positions for training and 27 positions for testing.

Overall, elevations are much harder to locate than azimuth. As stated by Rayleigh (Yost, 2017) and (Dai Wei et al., 2016), a network depends mainly on the inter-aural level difference (ILD) and inter-aural phase difference (IPD) in estimating azimuth. Elevation can not rely on these two features, since its rotation axis is in line with the microphone axis; thus, a rotation in elevation only change the mic

orientation and not position.

The performance is measured by three metrics: accuracy, generalization ability, and time and memory cost. Comparatively, the research finds that MLP achieves the highest accuracy while requires the lowest amount of memory, while CNN has the highest generalization capability.

Two different audio representations – the Mel Frequency Cepstral Coefficient and Log Filter Bank – are provided for the neural network. The results indicate that in average networks with log filter bank as its input feature performs better in generalizing input than the ones that use MFCC. This means that preprocess can hold a crucial role in optimizing neural network performances, especially when the network size is small.

MLP, which is the most basic form of neural networks, achieves the highest accuracy. It is somewhat surprising because, in a study by (Dai Wei et al., 2016), it is stated that MLP has a limited temporal dynamics that might cause a problem in dealing with sequential data such as sound. However, as our sound is pure tone, temporal dynamics seems to matter less. Furthermore, our MLP models require less time and memory cost than the other models.

CNN reaches an accuracy of 62.21% when exposed to MFCC, and 56.28% with Log Filter bank. This accuracy comes with a larger memory and time consumption. Unlike RNN and MLP, CNN uses multiple filters when analyzing a single feature. This is known as parameter sharing, in which a single parameter in CNN is used many times. CNN also achieves the highest generalization accuracy among the models.

BRNN outperforms RNN in terms of generalization ability. Both models has a very similar test accuracy, but RNN comes with higher memory cost while BRNN requires a lot more time to train.

It is hard to figure out how the ear design affects the localization accuracy because the author does not compare it with other ear design. As mentioned in the methodology, elevations estimation depends on the spectral modification caused by the ear because there is no positional difference of the microphone with varying elevation angles.

Elevation estimation needs to rely mainly on the spectral modification of the sound. This estimation may be more accurate if the author uses a higher sound frequency. As (Hwang, Park, and Park, 2011) explains, the ear starts to notice elevation difference when the sound is 6 kHz and above. However, for future project, there are two reasons why the author thinks higher frequency might not be appropriate: First, the high-frequency sound is hard to sample. Each device has a limited sampling rate that might not be enough to represent the sound wave. Second, humans mainly communicate in a relatively low frequency. If the motivation for applying SSL is for human-robot interaction, it is not suitable to expose the robot with a high frequency.

5.2.1 Future project ideas based on current limitations

All human knowledge is uncertain, and none of it can escape criticism. The author own critics on current project limitations is the basis of the future project ideas. The author suggests seven key ideas:

The first is to use and compare other neural network architecture for SSL purposes. There are a lot of different deep learning methods like Spiking Neural Network, I-Vector, and Gaussian Mixture Model that might work. A study by (Dai Wei et al., 2016) suggests that I-Vector has an adequate capability in scene recognition (Dai Wei et al., 2016). I-Vector might also perform well in sound localization, as SSL also involves dealing with audio data.

The second is to add more data by collection or augmentation. The author's data set does not cover a wide range of combination of source positions. Since these results could be affected by the limited amount of training data, data augmentation might be more practical. The system also hasn't dealt with the different sound tones and types, multiple simultaneously active sound sources, and different environments.

The third is to combine other perceptual capabilities such as vision, sound extraction, and separation with SSL. Human capabilities are not limited to sound localization. We can also extract, within a group of speakers talking simultaneously, the utterance emitted by the person we wish to focus on.

Another possibility is to combine visual with the auditory feature. Sound can also complement an image or video for better interaction. (He, Motlicek, and Odobez, 2018) compare different neural networks model for sound localization combine with visual information from Pepper robot camera. Antoin Deleforge performs recordings of a single static sound source emitting white noise or speech from different positions. The sound source is a loud-speaker equipped with a visual target manually placed at different positions around the system. This recording is available to the public and might be a valuable dataset SSL robotics.

Furthermore, Rascon said that learning-based approaches could be used for distance estimation. However, to make this improvement on the device must be added, such as adding mobility to the robot or use a multi-microphone array.

The fourth involves modifying the ear design. (Hwang, Park, and Park, 2011) design obtains to increase localization accuracy by introducing asymmetry into their ear design. Asymmetry happens in nature, such as an owl. Another approach is to use an anthropomorphic design such implemented by (Murray and Erwin, 2011) might be another option.

The fifth involves using different input representations. It is rarely the case that the raw audio file is used for neural networks sound processing. Preprocessing helps the network converge faster and more accurate. As indicated by our works, different input representations form corresponds to different performances in generalization ability.

The sixth is to perform sound localization in real-time where a robot could orientate the head directly after a sound source emit a sound. There should be a program that interfaces the neural networks

and the movement of the motor so that real-time movement can be achieved.

The seventh is to use different hyperparameter especially those that the author have not used. The most time-consuming part of this project is tuning the neural network. The neural network has many tuning knobs; therefore, it is not possible to explore them all. There might be things that the author overlook that affects the performance of the network significantly.

Based on the author's experienced there are two hyperparameter that the author thinks will be critical: validation techniques and input representations. K-fold cross-validation could replace the auto validation that the author used. In cross-validation, no data is wasted during training. It does this by splitting the training dataset into k subsets and takes turns training models on all subsets except one which is held out. The process is repeated until all subsets are given an opportunity to be the held-out as the validation set.

Input representations affects the localization performance. In the author case, log filterbank energies defeat MFCC in terms of generalization. Most studies in a neural network for SSL use MFCC for the input representation, such as done by (He, Motlicek, and Odobez, 2018) and (Murray and Erwin, 2011). There are a lot of representation that the author finds has never been used for localization purposes. Examples includes spectral subband centroids. To convert audio feature into other representation is relatively easy, as there are a lot of libraries available for this purpose, such as Python Speech Feature and Smile6k.

5.2.2 Contributions

The author contributions are three-fold:

1. A comparison of neural network performances in terms of its accuracy, generalization ability, and time and memory cost
2. A dataset for future SSL projects
3. Design of a pan-tilt system, robot head, artificial ear, as well as deep learning code for SSL purposes which is made to be open source.

References

Adavanne, S., Politis, A., Nikunen, J., and Virtanen, T. (2018). Sound Event Localization and Detection of Overlapping Sources Using Convolutional Recurrent Neural Networks. *IEEE Journal of Selected Topics in Signal Processing*.

Argentieri, S., Danes, P., and Souères, P. (2015). A survey on sound source localization in robotics: From binaural to array processing methods. *Computer Speech & Language*. 34.1, pp. 87–112.

Dai Wei, J. L., Pham, P., Das, S., and Qu, S. (2016). Acoustic scene recognition with deep neural networks (DCASE challenge 2016). *Robert Bosch Research and Technology Center*. 3.

Faludi, A. (2013). *A decision-centred view of environmental planning*. Vol. 38. Elsevier.

Flynn, A. M., Brooks, R. A., Wells III, W. M., and Barrett, D. S. (1989). *Squirt: The prototypical mobile robot for autonomous graduate students*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB.

Ganchev, T., Fakotakis, N., and Kokkinakis, G. (2005). "Comparative evaluation of various MFCC implementations on the speaker verification task". In: *Proceedings of the SPECOM*. Vol. 1. 2005, pp. 191–194.

Gardner, B., Martin, K., et al. (1994). *Hrft measurements of a kemar dummy-head microphone*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Guanga, A. (2011). *Machine Learning: Bias VS. Variance*. Available from: `https://becominghuman.ai/machine-learning-bias-vs-variance-641f924e6c57`. (accessed: 17.09.2019).

Hanlon, J. (2019). *Why is So Much Memory Needed for Deep Neural Networks*. Available from: `https://www.graphcore.ai/posts/why-is-so-much-memory-needed-for-deep-neural-networks`. (accessed: 15.09.2019).

He, W., Motlicek, P., and Odobez, J.-M. (2018). "Deep neural networks for multiple speaker detection and localization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 74–79.

Huang, J., Ohnishi, N., and Sugie, N. (1997). Building ears for robots: sound localization and separation. *Artificial Life and Robotics*. 1.4, pp. 157–163.

Hwang, S., Park, Y., and Park, Y.-s. (2011). Sound direction estimation using an artificial ear for robots. *Robotics and Autonomous Systems*. 59.3-4, pp. 208–217.

Lax, P. D. and Phillips, R. S. (1990). *Scattering theory*. Vol. 26. Academic press.

Mayo, M. (2017). *Neural Network Foundations, Explained: Activation Function*. Available from: `https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html`. (accessed: 15.09.2019).

Murray, J. C. and Erwin, H. R. (2011). "A neural network classifier for notch filter classification of sound-source elevation in a mobile robot". In: *The 2011 International Joint Conference on Neural Networks*. IEEE, pp. 763–769.

Nakadai, K., Matsuura, D., Okuno, H. G., and Kitano, H. (2003). "Applying scattering theory to robot audition system: Robust sound source localization and extraction". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. Vol. 2. IEEE, pp. 1147–1152.

Nakamura, S., Hiyane, K., Asano, F., Nishiura, T., and Yamada, T. (2000). "Acoustical Sound Database in Real Environments for Sound Scene Understanding and Hands-Free Speech Recognition." In: *LREC*.

Peixeiro, M. (2019). *How to Improve a Neural Network With Regularization*. Available from: `https://towardsdatascience.com/how-to-improve-a-neural-network-with-regularization-8a18ecda9fe3`. (accessed: 17.09.2019).

Potisk, T. (2015). "Head-related transfer function". In:

Rascon, C. and Meza, I. (2017). Localization of sound sources in robotics: A review. *Robotics and Autonomous Systems*. 96, pp. 184–210.

Robotics, T. (2019). *PhantomX Pan Tilt*. Available from: `https://www.trossenrobotics.com/phantomx-pan-tilt`. (accessed: 15.09.2019).

Rodemann, T., Ince, G., Joublin, F., and Goerick, C. (2008). "Using binaural and spectral cues for azimuth and elevation localization". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2185–2190.

Stewart, M. (2019). *Neural Network Optimization*. Available from: `https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0`. (accessed: 15.09.2019).

Thuillier, E., Gamper, H., and Tashev, I. J. (2018). "Spatial audio feature discovery with convolutional neural networks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6797–6801.

Upadhyay, Y. (2019). *Regularization techniques for Neural Networks*. Available from: `https://towardsdatascience.com/regularization-techniques-for-neural-networks-e55f295f2866`. (accessed: 15.09.2019).

Vera-Diaz, J., Pizarro, D., and Macias-Guarasa, J. (2018). Towards end-to-end acoustic localization using deep learning: From audio signals to source position coordinates. *Sensors*. 18.10, p. 3418.

Walia, A. (2017). *Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent*. Available from: `https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f`. (accessed: 17.09.2019).

Wall, J. A., McDaid, L. J., Maguire, L. P., and McGinnity, T. M. (2012). Spiking neural network model of sound localization using the interaural intensity difference. *IEEE transactions on neural networks and learning systems*. 23.4, pp. 574–586.

Yost, W. A. (2017). “History of sound source localization: 1850-1950”. In: *Proceedings of Meetings on Acoustics 173EAA*. Vol. 30. 1. ASA, p. 050002.

6 Appendix

Link for the dataset, robot design, and code: https://github.com/arief25ramadhan/Sound-Source-Localization