A Texture Based Industrial Fault Diagnosis Model Using Gammatone Filter Bank and Transfer Learning

by

Atique Morshed
21141067
Farhan Md. Siraj
17101155
Abu Sadat MD. Munim
21341046
Tasnimul Karim Ayon
21141075
Saad Rafsan Goni
17301003

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

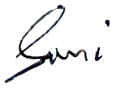
Department of Computer Science and Engineering
Brac University
September 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Atique Morshed
21141067

Farhan Md. Siraj
17101155

Abu Sadat MD. Munim
21341046

Tasnimul Karim Ayon
21141075

Saad Rafsan Goni
17301003

Approval

The thesis titled "A Texture based Industrial Fault Diagnosis using Gammatone and Transfer Learning" submitted by

1. Atique Morshed(21141067)
2. Farhan Md. Siraj(17101155)
3. Abu Sadat MD. Munim(21341046)
4. Tasnimul Karim Ayon(21141075)
5. Saad Rafsan Goni(17301003)

Of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on September 26, 2021.

Examining Committee:

Supervisor:
(Member)

Jia Uddin, PhD
Assistant Professor (Research Track)
AI and Big Data Department
Endicott College, Woosong University
Associate Professor (on leave)
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Faisal Bin Ashraf
Lecturer
School of Data and Sciences
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Rapid industrial growth has increased the vulnerability of systems to malfunction and permanent damage. Fault detection systems have been installed to prevent such occurrences. In order to eliminate potential life-threatening dangers or unforeseen obstacles that may jeopardize the manufacturing process, early fault detection has become an essential aspect of modern industry. Because artificial intelligence has become increasingly successful across numerous different domains, many researchers have employed deep learning models to classify faults and are always trying to find faster, more accurate ones. In this paper, we present a deep transfer learning architecture that consists of long short-term memory (LSTM) layers of Recurrent Neural Network to extract features enhanced by gammatone like spectrogram. For the dataset, we have used malfunctioning industrial machine investigation and inspection (MIMII) and ToyADMOS datasets. Our experimented results show that the proposed model detect the different faults with precision. Also, our modified gammatone fast fourier method outperforms traditional gammatone accurate method with consistent performance across all environments.

Keywords: Deep Learning; Transfer Learning; Fault Detection; RNN; LSTM; Gammatone Filter Bank

Dedication

This thesis is dedicated to all health-care personnel who are putting their lives on the line for the sake of society, helping COVID patients and striving to find COVID-19 cures.

Acknowledgement

All praise to the Almighty Allah for whom our thesis have been completed without any major interruption.
Second, Dr. Jia Uddin, our supervisor, for helping us through the entire procedure. This thesis would have been nearly impossible to complete without his guidance and inspiration.
Finally, we want to thank our parents for always believing in us and supporting us throughout our lives.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$ADMOS$ Anomaly Detection In Machine Operating Sounds

ADN Abnormal

DNN Deep Neural Network

ENV Environment

GA Gammatone Accurate

GF Gammatone Fast

$HO-scale$ half-O scale

$LSTM$ Long Term Short Memory

$MIMII$ Malfunctioning Industrial Machine Investigation and Inspection

ML Machine Learning

$N-scale$ Nine-scale

NOR Normal

RNN Recurrent Neural Network

TL Transfer Learning

Chapter 1

Introduction

1.1 Overview

Today's economy is heavily reliant on the industry thus on the machines. The growth of industries made machines complex, larger and prone to system vulnerability more often than before. As a result, a significant amount of time and effort has been invested in fault detection in order to prevent the aforementioned occurrences from happening. Most of these methods work better and are more appropriate in their specific environment than others [19]. But the necessity of faster and accurate fault detection methods remains constant. According to [4], A fault occurs when a system has at least one different feature from the list of accepted ones. Fault detection methods allow us to know the current situation of machines. If the methods are slower, preventative measures can not be implemented before life-threatening operator injury or massive production failure occur. On the other hand, inaccurate methods may result in unnecessary management and economic loss. Over the years, many methods for identifying and diagnosing faults have been invented. Some of these models were data-driven[1], some of those were based on models[3] while others were based on knowledge[7].

Conventionally, machine learning was used to classify faults. But these ML-based models had many lackings such as specific context, human interaction and so on. The rising of deep neural network and transfer learning models has been revolutionary to battle these shortcomings. By using the multi-layered architecture, DNN models can learn multiple representations of the input data [10]. This means that individual layers can achieve significantly better representation of data which can lead to the extraction of complex features [14]. Our choice of model is RNN. It can process the input of any length, model sizes have no effect on the input size, and it can remember previous memory. A typical RNN has a short-term memory. They can have a long-term memory when combined with an LSTM. We decided to use transfer learning because training data for new environments costs a lot of extra time which hinders our goal to find fast architecture. Thanks to the invention of TL, we are able to use previously trained model in new environment which drastically reduces time taken to train the new model. Also, selecting proper data and signal processing has been crucial to fasten the overall task. Various data preprocessing, data mining methods are used in accordance with fault detection architecture. Some of them

were demonstrated in [15]. Applications based on these methods are also analyzed in [12]. In the case of signal processing, we are applying gammatone filter banks to our categorized data. It is quite similar to the filtering process of mammals. The relation between the output of gammatone filter bank and the perception of ear was depicted in [5]. In the search for a more proficient method, we are using a faster technique of the aforementioned signal processing method and find out if the gammatone fast method is more accurate, faster than the gammatone accurate method. As the attention to industrial fault detection increased, many organizations are making different kinds of faulty datasets for research purposes. For this study, we have used malfunctioning industrial machine investigation and inspection (MIMII) and ToyADMOS datasets and made multiple environments out of those datasets. This stufy attempts to present a deep transfer learning based RNN-LSTM model while comparing the performance of gammatone fast method and the gammatone accurate method.

1.2 Aims and Objectives

In our study we had specific goals in mind. They are depicted below:

- Our objective is to successfully detect and classify normal and faulty data using RNN-LSTM deep transfer learning model.
- Ensuring proper utilization of transfer learning to reduce training time to obtain almost similar result.
- Also, we ran both the gammatone accurate and gammatone fast methods on different environments of MIMII and ToyADMOS datasets to determine their performance.
- Finally, our aim is to detect faults as early as possible accurately to increase their reliability, reduce energy, service, maintenance cost and prevention of long-lasting damage.
- The goal is to develop a more efficient system that will help and be applied to a broad spectrum of industrial processes.

1.3 Thesis Outline

Our goal is to provide an RNN-LSTM model based on deep transfer learning and compare the performance of the gammatone fast and accurate methods. The first chapter gives a gentle introduction to our study and depicts our research objectives. A detailed literature review is provided in the second chapter. All the work in this field is briefly discussed. Third chapter shows the idea to predict model using decision tree. Dataset preprocessing and normalizing were discussed in the fourth chapter. The detailed architecture of the proposed deep transfer learning model is presented in the fifth chapter. Chapter six depicts the experimental results. Finally, the paper is concluded in the chapter seven.

Chapter 2

Related Work

Deep learning algorithms and deep and complicated layers were used to create novel classification models for defect identification and diagnosis. The majority of shallow learning models extract only a few feature values from signals, reducing the original signals dimensional. Deep belief networks, limited Boltzmann machines, and auto encoders are some other deep neural network designs that have been effectively employed in this field of research. Because of their deep design, deep learning models are capable of learning more complicated structures from data sets than standard machine learning models. However, in order to attain better accuracy, they require bigger samples and more processing time.

In [17], To replicate genuine settings, researchers gathered 26,092 usual sound segments and 6,065 unusual sound segments and combined sound effects collected from several real factory's machineries. They also demonstrated an example of assessment for auto encoder-based unsupervised anomalous sound detection using the MIMII data set. The major difficulties for creating the unsupervised anomaly detector, they discovered, are non-stationary machine sound signals and noise. According to them, these findings can be used as a benchmark for improving the MIMII data set's anomaly detection accuracy.

Several efforts in the field of defect detection have been completed. Kang and Kim [9] suggested a model. They suggested a two-dimensional Shannon waveform for very accurate fault detection of a variety of induction motor problems. Furthermore, to generate more optimized and efficient inductive motor qualities, F. M. Khellah et al. used a well-known visual feature extraction approach. The model, however, had several faults. The medium is not considered in the transmission model, which is one of the Shannon-Weaver Model's faults. It's possible that the channel is loud, and the receiver is unable to decode it, producing communication problems.

Shen, Wang, Kong, and Tse [8] developed a defect diagnostic technique that can be used on both bearing and gear data sets. When compared to the defect diagnosis ratios, the method presented here provides a higher level of accuracy.

Janssens, Slavkovikj, Vervisch, Stockman, Loccufier, Verstockt, Walle, and Hoecke [11] reported in 2016 that a feature-learning system based on convolutional neural networks outperforms a conventional feature-engineering technique that employs

manually created features and a random forest classifier. The former has a precision of 93.61 percent, while the latter has a precision of 87.25 percent.

Koizumi, Saito, Uematsu, Harada, and Imoto found out the deviation for ToyADMOS data set in their research [16]. Each sub data set was used to evaluate a basic unsupervised ADS task. To simplify the experiment, they merged the different findings of the sub-data set and used them as a one single sub data set.

Widman e. [21] took into account diagnostic limitations and presented a model that correctly reflects various temporal and spatial phenomena. It makes it easier for knowledge engineers to maintain consistency in the knowledge base. The rule-based technique, however, has a number of drawbacks, including a lack of generality and a poor handling of unforeseen circumstances.

Chapter 3

Recurrent Neural Network

Recurrent neural networks (RNNs), which are the state-of-the-art method for sequential data, are used in both Apple's Siri and Google's voice search [18]. Every one of the outputs and inputs in traditional neural networks are independent of each other; however, when anticipating the very next phrase in a phrase, the previous phrases are necessary, and therefore the prior phrases must be memorized. So as outcome, Recurrent neural networks was developed, and the developed model overcame this issue by using a Hidden Layer. The Hidden state, which remembers particular information about a sequence, is the most essential component of a Recurrent neural network [13]. In a conventional RNN, it consists of short-term memory. The model can have a long-term memory when combined with an LSTM. Long short-term memory networks (LSTMNs) are a form of recurrent neural network extension that extends memory. As an outcome, it's well-suited to study from significant events that happen over a long period of time. The layers of an RNN, sometimes referred to as an LSTM network, are built using the units of an LSTM. Thanks to LSTMs, RNNs can remember inputs for a long period. Because LSTMs are memory-based, they can store a lot of data comparable to a computer's memory. The LSTM can read, write, and erase information from its memory. There are three gates in an LSTM: input, forget, and output. These gates control whether to accept fresh data (input gate), discard information that is no longer relevant (forget gate), or allow it to impact the output at the current time step (output gate).

Imagine a network having three hidden levels, one output layer, and one input layer. Each hidden layer, like other neural networks, will have its own set of weights and biases, such as (w1, b1) for hidden layer 1, (w2, b2) for second hidden layer, and (w3, b3) for third hidden layer [y2]. This implies that each of these layers is self-contained, with no memory of prior outputs. Recurrent Neural Network is built on current state calculation, activation function, and output calculation formula. Current state computation:

$$h_t = f(h_{t-1}, X_t) \tag{3.1}$$

Here,
The present state is h_t, the prior state is h_{t-1}, and the input state is X_t.
The activation factor is determined using:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}X_t) \tag{3.2}$$

Here,
W_{hh} -> weight at recurrent neuron
W_{xh}-> weight at input neuron Lastly,
Computation function of the output:

$$Y_t = W_{hy}h_t \tag{3.3}$$

Here,
Y_t -> output
W_{hy} -> weight at output layer

A recurrent neural network functions in this manner.

Chapter 4

Data set Preparation

4.1 Dataset Description

To evaluate the proposed model, we have accumulated data from malfunctioning industrial machine investigation and inspection (MIMII) and ToyADMOS to assess the proposed model. The MIMII dataset contains recordings of four industrial machines (fan, pump, slider, and valve). The sounds were recorded in three different environments (-6 dB, 0 dB, and 6 dB) for all four machines. Each audio file has a sample rate of 16,000hz, which has 16 bits per sample. Each industrial machine contains both normal and abnormal sounds. The abnormal sound consists of different kinds of anomalies like rotating unbalance, leakage, rail damage, and contamination. An eight-channel microphone array was used to record the audio. Finally, the background noise of multiple factories was then added to the recorded audio. There are seven models, but the public version 1 release contains four models. The four public releases are tagged with ID 00, 02, 04, and 06 [17].

The ToyADMOS data contains normal and abnormal sounds of miniature machines where the machines are deliberately damaged to represent abnormal sounds. The normal sounds are of around 540 hours. There are a total of 12,000 samples of abnormal sounds. They were recorded at 48,000hz. For the recording, four different microphones were used. The datasets are divided into three distinct sections – ToyCar, ToyConveyor, ToyTrain. ToyCar dataset is used for production inspection tasks, whereas ToyConveyor and ToyTrain datasets are used for fault diagnosis tasks. Since we are working on a fault diagnosis system in this experiment, only ToyConveyor and ToyTrain datasets were used from the ToyADMOS dataset in our case. The ToyConveyor has three different environments. The difference between each of the environments is the size of the machine. The same manufacturing company produced them. To represent the abnormal condition, (1) extensive tension was applied over the tension pulley, and the tail pulley (2) many metric objects like a steel nut, steel washer, and screwdriver) were attached to the conveyor. (3) Over and under voltage was applied. The baseline voltage used was 3.0V, 2.5V was used to under volt the system, and 3.5V was used to overvolt the system. For the ToyTrain dataset, the sound was generated from four different environments. A Commuter and a bullet train were used. To create different environments, combinations of these two types of trains were used. So, there's a total of four environments.

For the railways, N-scale and HO-scale railways were used. The difference between them is the size. HO-scale is the larger railway, and N-scale is the smaller railway. To create abnormal conditions, (1) wheel axes were chipped using radio pliers. (2) the railway tracks were chipped using radio pliers and also obscured stones on the railway. (3) Tracks were disjointed. For each environment of ToyConveyor and ToyTrain, two different types of sound were recorded. The first type contains continuous sound. The sound was recorded continuously and was cut every 10 minutes. In this case, the abnormal sounds were not recorded. The second type contains individual sounds, which includes sounds of the entire operation, including starting sound and stopping sound, and each segment is 10s long. The individual variety of sounds contains normal as well as abnormal sounds [16].

4.2 Dataset Preprocessing

The MIMII dataset contains seven models as described above. To prepare the dataset, we have used the model "id00". Each model contains different numbers of sound files. The files were imported into MATLAB. Figure 4.1 depicts the signal of sample audio files for each machine of the MIMII dataset.

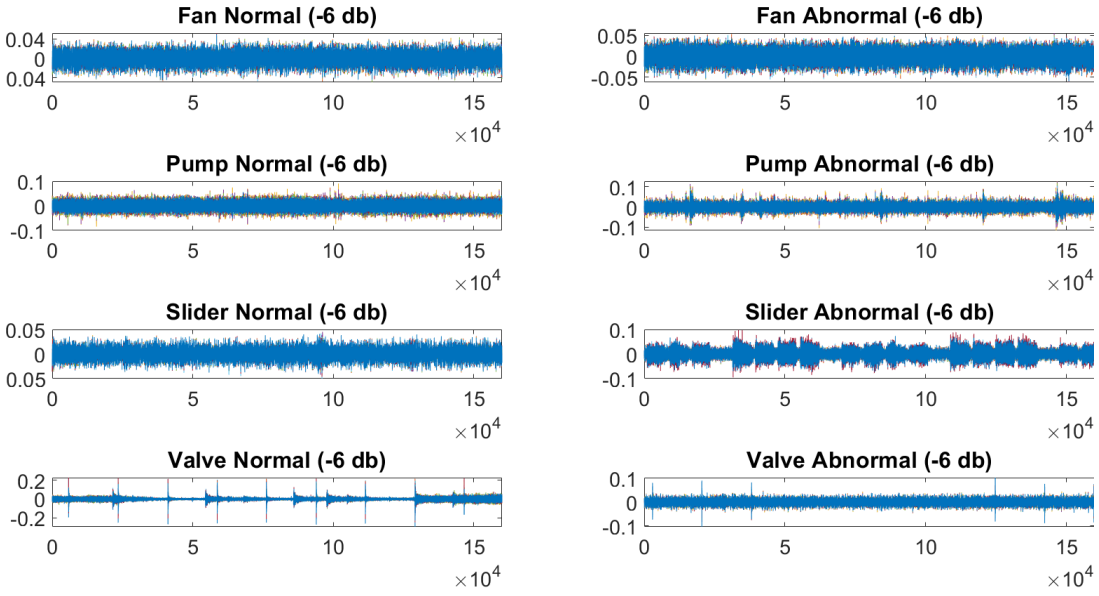


Figure 4.1: Sample normal and abnormal signals for all machines (fan, pump, slider, and valve) of the MIMII dataset.

As described before, an eight-channel microphone array was used to record the audio. These were placed evenly around the machine (360 degrees). Since there is a total of 8 microphones, they were 45 degrees apart from the machines. For our experiment, we took the audio channel closest to each of the machines. We took the 5th, 3rd, 7th and 1st channels for the fan, pump, slider, and valve, respectively. We also noticed that the number of normal and abnormal audio files for all the machines is different. So, we took a maximum of 400 audio files from each type for each machine. These include the normal audio for all four machines and also the abnormal audio of the

fan. But for other cases, the number of audio files was less than 400. So, we took all the available audio files for these cases. That means, for abnormal sounds of the pump, slider, and valve, we took 143, 356, and 119 files, respectively. Table 4.1 contains all the file selection information for the MIMII dataset.

	Fan		Pump		Slider		Valve	
dB	NOR	ADN	NOR	ADN	NOR	ADN	NOR	ADN
-6	400	400	400	143	400	356	400	119
0	400	400	400	143	400	356	400	119
6	400	400	400	143	400	356	400	119

Table 4.1: Detailed information about the selected audio files of MIMII dataset

The sound files were then imported into MATLAB. Figure 2 depicts the sample normal and abnormal signals for all machines. The audio files have a sample rate of 16000hz and each audio file has a length of 10 seconds. So, there are (16000*10) = 1,60,000 points. For our case, we extracted 40,000 points from each audio file. That gives us 2.5 seconds of audio from each file.

For the ToyADMOS dataset, all three environments were used from the ToyConveyor dataset, and three out of four environments were used in the case of the ToyTrain dataset. The individual type of sounds was used for our experiment as it contains both normal and abnormal sounds [16]. To keep the signal amount similar to the MIMII dataset, we used 100 sound files from each environment and for both normal and abnormal cases. They were sampled at 48,000hz, and each file contains 10 seconds of audio. Since each file contains the starting and ending sound of the machine, we had to take the full 10 seconds of audio from each file. Figure 4.2 depicts the sample signal of the sound files for all the machines of the ToyADMOS dataset.

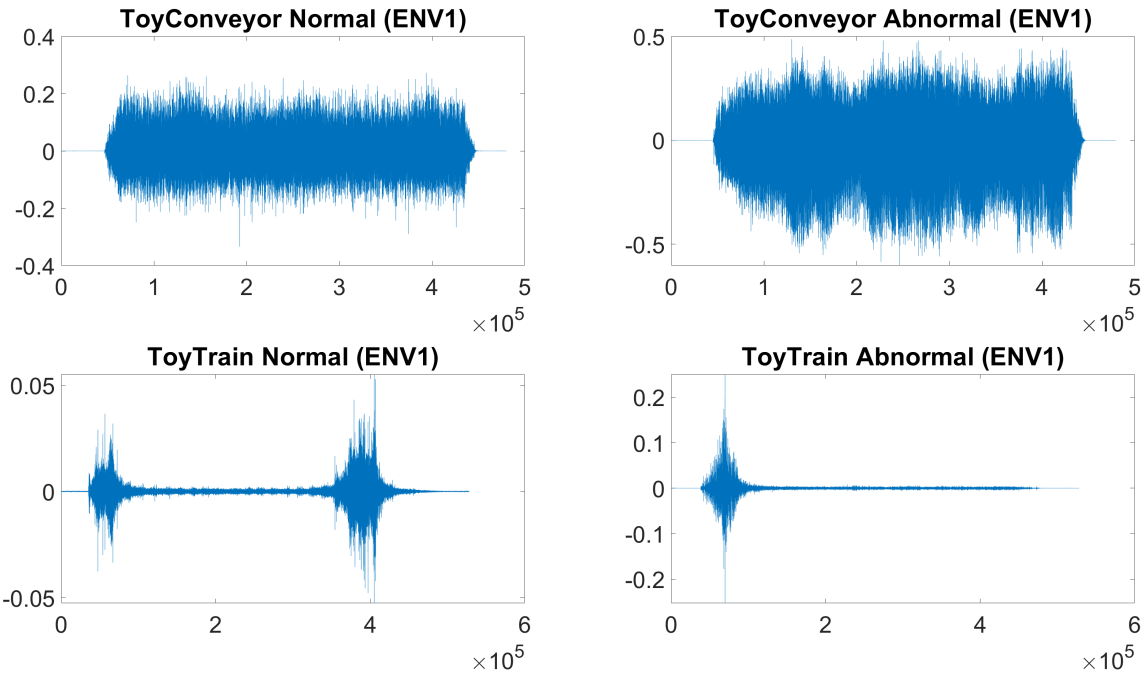


Figure 4.2: Sample normal and abnormal signals of ToyConveyor and ToyTrain dataset.

After properly categorizing our dataset, we applied Gammatone filter bank. Gammatone filter banks are one of the most popular linear approximation techniques. It is widely used in models of auditory systems. It is similar to the filtering that is performed by our ears. But there is some major difference between these techniques. As the frequency increases, our ear's frequency sub-bands get wider. This is not the case for the Gammatone-like spectrograms, where the bandwidth is constant [6]. Equation 1 shows the impulse function's Gammatone function in the time domain, which produces a gamma distribution and sinusoidal tone [20].

$$g(t) = at^{n-1} \cdot \cos(2\pi ft + \phi) \exp(-2\pi bt) \tag{4.1}$$

A popular implementation of the Gammatone filter bank is the Auditory Toolbox Version 2 of Malcolm Slaney [2]. In this implementation, a signal is processed by every filter bank. To Visualize the signals in the time-frequency domain, it is essential to add all the energy within regular time bins. While this may be the most accurate option, this is also computationally expensive. There is also a faster approach to this which gives a very similar result to the accurate method, but is about thirty to forty times faster than the accurate method. This is done by calculating the fixed-bandwidth spectrogram, and then combining the FFT-based spectra into the Gammatone response [6]. Both accurate and fast methods were applied in our dataset. Figure 4.3 depicts the difference between the Gammatone accurate method and the Gammatone fast method.

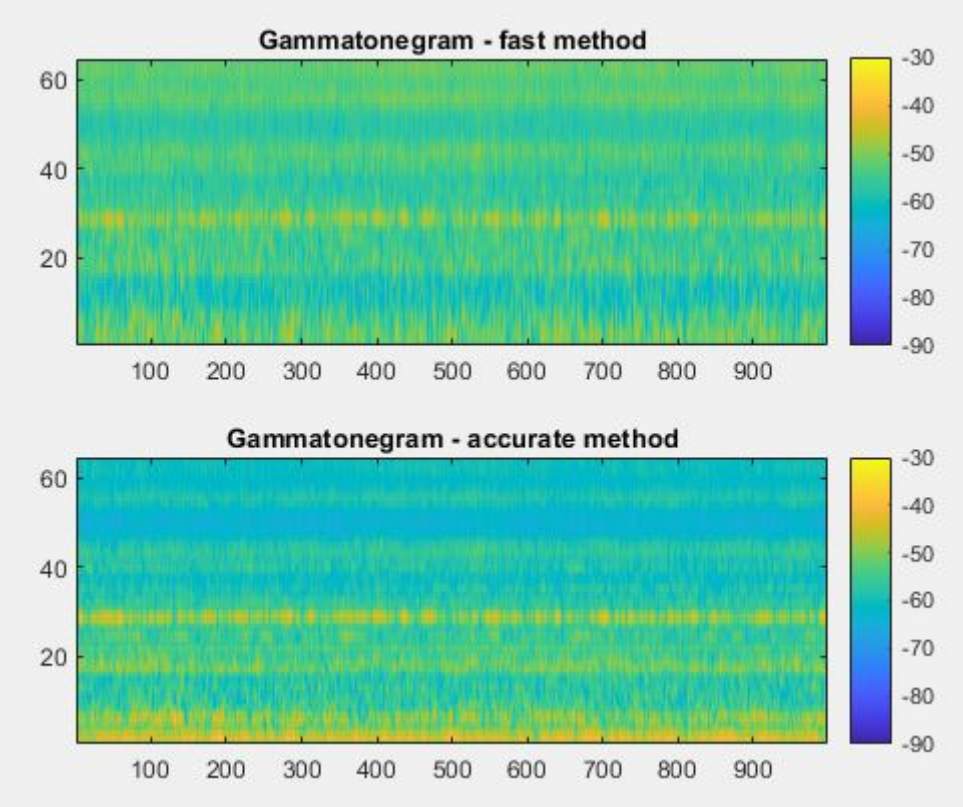


Figure 4.3: A sample signal of the Gammatone fast and Gammatone accurate method.

We applied the fast and accurate method on all environments of the MIMII and ToyADMOS dataset to compare the results. So, each environment has a Gammatone accurate method variant and a Gammatone fast method variant. Table 4.2 contains the required time for applying the Gammatone fast and accurate method on all environments of the combined MIMII and ToyADMOS dataset.

Name	Dimension	Total Files	Time(sec) [GF]	Timer(sec) [GA]
ENV1	65378x32x32	3018	36.0108	625.736
ENV2	65378x32x32	3018	33.5237	655.8263
ENV3	65378x32x32	3018	36.1919	659.2903

Table 4.2: Required time for applying Gammatone fast (GF) and Gammatone accurate (GA) on all environments of the combined MIMII and ToyADMOS dataset

It is clear that the Gammatone accurate method is computationally expensive. The final result showed that the Gammatone fast method performs better than the Gammatone accurate method, while applying the Gammatone filter bank takes less time.

Our final dataset is divided into three environments. The first environment consists of the processed signal of -6db MIMII dataset and environment 1 of ToyConveyor and ToyTrain dataset. We've combined the 0db MIMII dataset with the 2nd environment of the ToyConveyor and ToyTrain dataset for the second environment. Finally, the third environment consists of the 6db MIMII dataset and environment 3 of the ToyConveyor and ToyTrain dataset. We then used the first environment to evaluate our proposed model. The weights and architecture of the above result were saved to apply transfer learning on the second and third environments.

Chapter 5

Model Implementation

5.1 An Overview of the Workflow

The initial objective of this study was to prepare the model in such a way that ensures proper utilization of transfer learning which would eventually lead to a decent fault diagnosis system. In order to do so, a qualitative approach had been adopted. The datasets used were audio files that had been preprocessed and normalized to be fed into the model. The primary concern was not only accuracy but also loss values and how many epochs it took to reach an acceptable threshold. The details of the workflow are given below:

- Acquiring the datasets. Both MIMII and ToyADMOS.
- Preprocess the datasets for optimal model efficacy. The datasets were preprocessed with gammatone filters. Both Gammatone Accurate and Gammatone Fast Fourier were applied and both filter's datasets ahd been used in further study.
- In order for the model to weigh the features efficiently, the datasets were normalized so that the values would be in the range of 0 to 1.
- Three environments were created with datasets that would be used to implement transfer learning. The first one had been chosen for further tasks.
- In each of the environments, the dataset had been split into an 80:20 ratio of train set and test set respectively.
- Initial training and testing had been done in the first environment. After the results came out the environment had been frozen and the attributes were carried over to the other environments for further study. This concludes the contribution of the first environment
- The model had been trained again in the second and the third environments and tested thereafter.
- Finally, results from all of the environments had been analyzed and compared. Therefore, a conclusion had been drawn.

An illustration of the workflow can be found in Figure 5.1

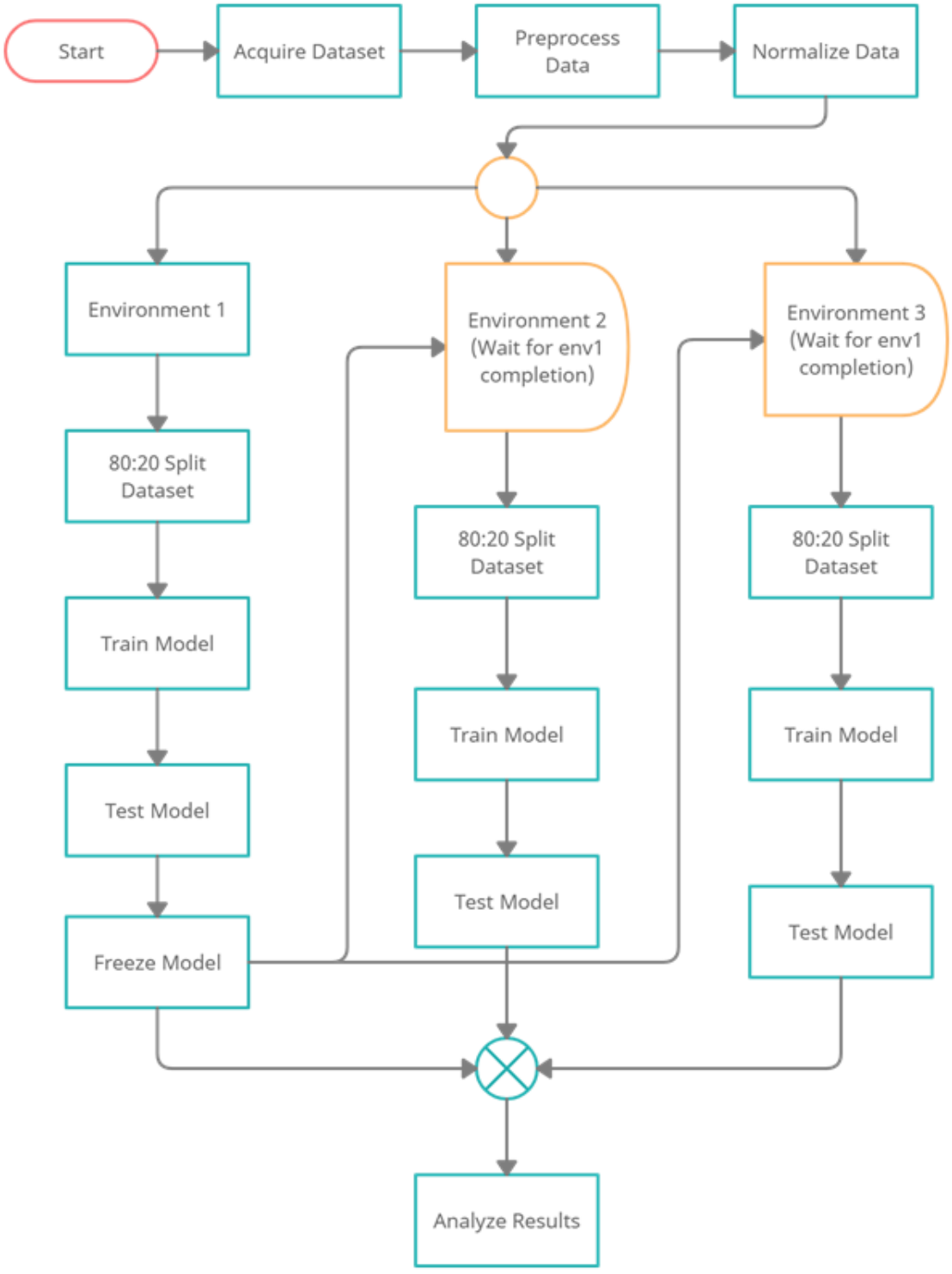


Figure 5.1: A detailed step by step illustration of the workflow

5.2 Implementation of the Model

After preparing the dataset it was time for implementation. Since the study focused on transfer learning it involved the usage of multiple environments. The model of choice was RNN with some added layers and in multiple environments. The implementation process is described below:

Transfer Learning: Opposed to conventional deep learning, models that had been trained earlier are used as a starting point in transfer learning. Pre-trained models carry over what they have previously learned and add new findings to their previously earned knowledge base. This is aimed to drastically increase efficacy along with reduced training time to obtain similar or better results. Since this study utilized transfer learning, three environments were created with the datasets to train and test the model.

- **First Environment:** Initially the dataset had been fed to the first environment and the other environments had been stalled. 80 percent of data chosen on a random basis had been used to train the model. On completion of training, the rest 20 percent of data had been used to test the model's performance. In order to obtain a borderline satisfactory result, the model had to go through 100 epochs. Thereafter the properties of the model had been frozen and carried over to the second and the third environments.

- **Second Environment:** The pre-trained model from the first environment had been used here. Again, the dataset had been split randomly into an 80:20 ratio to training and validation respectively. The starting point of this environment had been the already trained model. This environment had been aimed towards obtaining a better result than the first environment with the least number of epochs. The model required only 30 epochs to surpass the first model and obtained a very satisfactory result.

- **Third Environment:** The goal of this environment was to compare with the second environment and contribute to forming a conclusion. In correspondence to the previous environment, only 30 epochs had been required to surpass the first environment and obtain a result comparable to the second environment.

5.3 Layers of the Model

In order to optimize the model and also to prevent overfitting of the data, certain layers were added to the model. In each of the environments, the layers were present. Those layers had been implemented only after the dataset had been split on a random basis.

The first imposed layer was a LSTM layer which had been placed immediately after the input layer. The LSTM layer had consisted of 64 units leading to another 64 units of the second LSTM layer. A dense layer had been added after the second LSTM layer which also had 64 units. To avoid data overfitting, here a dropout layer

had been added. This dropout layer had also consisted of 64 units. Finally, another dense layer of 64 units is followed by the output layer. In Figure 5.2 a high-level representation of the layers is illustrated.

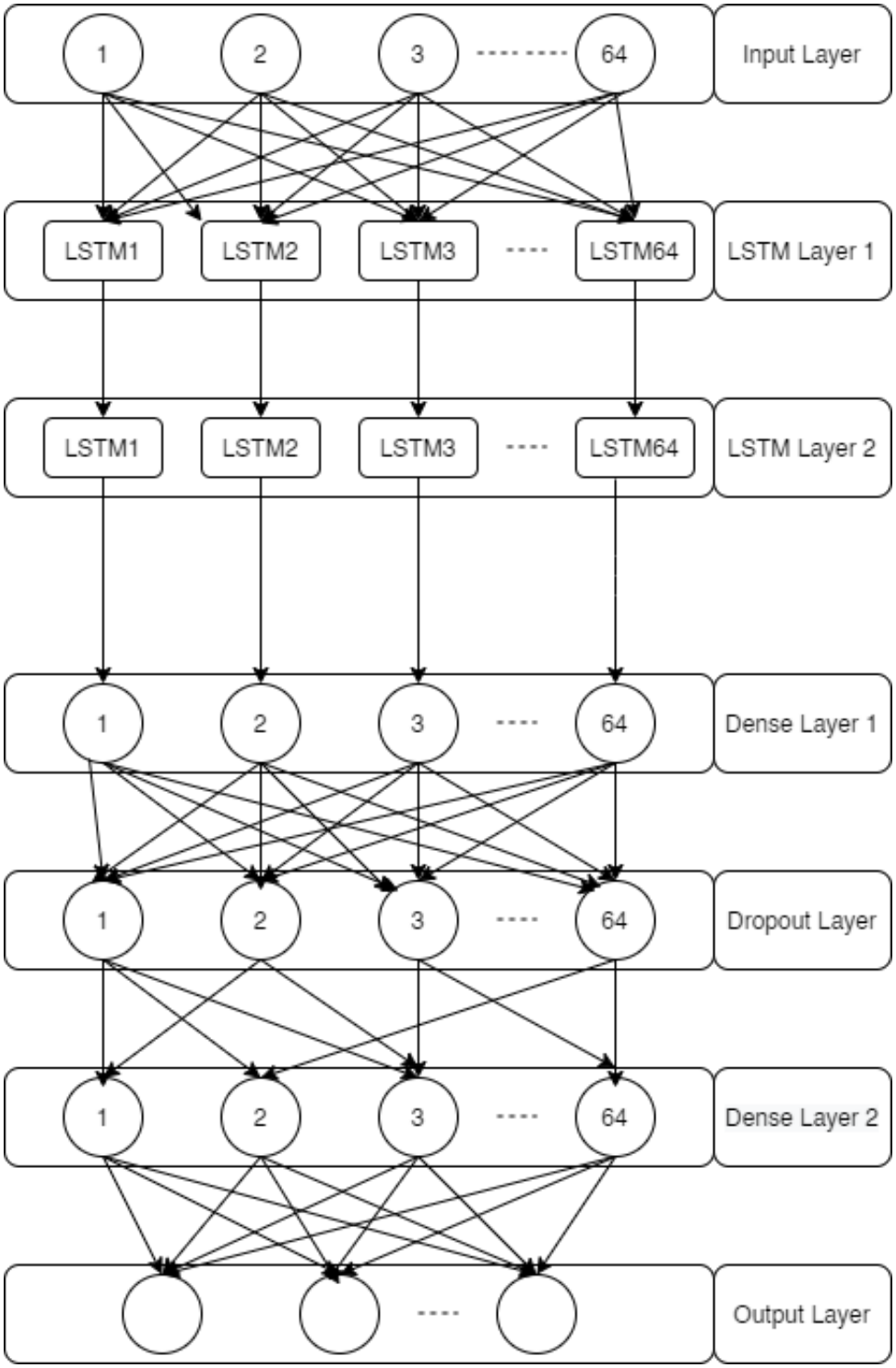


Figure 5.2: An overview of the layers

Chapter 6

Experimental Results and Analysis

6.1 Analysis of Gammatone Accurate Environments

First Environment: The first environment utilizes the non-pre-trained model with a Gammatone Accurate filter and completed 100 epochs. The filter had been applied on both MIMII and ToyADMOS datasets. During validation, the lowest precision and f1-score this environment had achieved was 0.92 and 0.96 respectively, by VALVE_N while VALVE_A had scored the lowest in recall with a 0.94. However, the highest score across all evaluating criteria was 1.00 and it had been observed multiple times.

Components	Precision	Recall	F1-score
FAN_N	0.99	0.96	0.98
FAN_A	0.98	1.00	0.99
PUMP_N	0.99	1.00	0.99
PUMP_A	1.00	1.00	1.00
SLIDER_N	1.00	0.96	0.98
SLIDER_A	1.00	1.00	1.00
VALVE_N	0.92	0.99	0.96
VALVE_A	0.99	0.94	0.97
ToyConv_N	1.00	1.00	1.00
ToyConv_A	1.00	1.00	1.00
ToyTrain_N	1.00	1.00	1.00
ToyTrain_A	1.00	1.00	1.00

Table 6.1: Detailed result of the GA Environment 1

Second Environment: This environment follows the transfer learning paradigm using the pre-trained model of the first environment for further operations and 30 epochs. In this environment, The precision score of VALVE_N had increased to 0.98 implying an improvement of 0.06, the f1 score of VALVE_N had also seen an improvement of 0.01. Furthermore, VALVE_A also had an improvement in terms of recall value by 0.03 reaching 0.97. However, the lowest precision value observed was 0.94 scored by SLIDER_N, the lowest recall was 0.95 scored by VALVE_N,

and the lowest f1-score was 0.95, again scored by SLIDER_N. The highest value observed was 1.00 in various segments.

Components	Precision	Recall	F1-score
FAN_N	0.99	0.96	0.98
FAN_A	0.95	1.00	0.98
PUMP_N	1.00	1.00	1.00
PUMP_A	1.00	1.00	1.00
SLIDER_N	0.94	0.96	0.95
SLIDER_A	1.00	1.00	1.00
VALVE_N	0.98	0.95	0.97
VALVE_A	1.00	0.97	0.98
ToyConv_N	1.00	1.00	1.00
ToyConv_A	1.00	1.00	1.00
ToyTrain_N	1.00	0.99	1.00
ToyTrain_A	0.99	1.00	1.00

Table 6.2: Detailed result of the GA Environment 2

Third Environment: After 30 epochs some drastic changes were observed in this environment. While most of the segments had a perfect score of 1.00, in others, steep declines in scores had been found. In this environment, the lowest observed precision score was 0.81, the lowest recall value was 0.59, and the lowest f1-score was 0.74 scored by SLIDER_N, VALVE_A, and VALVE_A respectively.

Components	Precision	Recall	F1-score
FAN_N	0.85	1.00	0.92
FAN_A	0.99	0.99	0.99
PUMP_N	1.00	1.00	1.00
PUMP_A	1.00	1.00	1.00
SLIDER_N	0.81	0.82	0.81
SLIDER_A	1.00	1.00	1.00
VALVE_N	0.94	0.99	0.97
VALVE_A	1.00	0.59	0.74
ToyConv_N	1.00	1.00	1.00
ToyConv_A	1.00	1.00	1.00
ToyTrain_N	1.00	1.00	1.00
ToyTrain_A	1.00	1.00	1.00

Table 6.3: Detailed result of the GA Environment 3

6.2 Analysis of Gammatone Fast Environments

First Environment: Although this environment used a non-pre-trained model preprocessed with the gammatone fast method, the scores were perfect. Each of the segments had scored 1.00. The differences with the previous first environment are the methods used in data preprocessing. 100 epochs were completed in this environment.

Components	Precision	Recall	F1-score
FAN_N	1.00	1.00	1.00
FAN_A	1.00	1.00	1.00
PUMP_N	1.00	1.00	1.00
PUMP_A	1.00	1.00	1.00
SLIDER_N	1.00	1.00	1.00
SLIDER_A	1.00	1.00	1.00
VALVE_N	1.00	1.00	1.00
VALVE_A	1.00	1.00	1.00
ToyConv_N	1.00	1.00	1.00
ToyConv_A	1.00	1.00	1.00
ToyTrain_N	1.00	1.00	1.00
ToyTrain_A	1.00	1.00	1.00

Table 6.4: Detailed result of the GF Environment 1

Second Environment: A model pre-trained with Gammatone Fast method had been used in this environment and had completed 30 epochs. Although this was an implementation of transfer learning, the scores did not surpass the first environment. The observed lowest precision was 0.98 scored by VALVE_N, the lowest recall was 0.98 scored by SLIDER_N, and the lowest f1-score was 0.99 scored by both SLIDER_N and VALVE_N.

Components	Precision	Recall	F1-score
FAN_N	1.00	1.00	1.00
FAN_A	1.00	1.00	1.00
PUMP_N	1.00	1.00	1.00
PUMP_A	1.00	1.00	1.00
SLIDER_N	1.00	0.98	0.99
SLIDER_A	1.00	1.00	1.00
VALVE_N	0.98	1.00	0.99
VALVE_A	0.99	1.00	1.00
ToyConv_N	1.00	1.00	1.00
ToyConv_A	1.00	1.00	1.00
ToyTrain_N	1.00	1.00	1.00
ToyTrain_A	1.00	1.00	1.00

Table 6.5: Detailed result of the GF Environment 2

Third Environment: This environment also utilized the pre-trained model of environment 1. However, in contrast to the first environment, not all of the segments achieved a perfect score of 1.00 due to VALVE_N scoring 0.99 indicating an improvement of 0.01 over the second environment. This environment had also completed 30 epochs before comparison.

Components	Precision	Recall	F1-score
FAN_N	1.00	1.00	1.00
FAN_A	1.00	1.00	1.00
PUMP_N	1.00	1.00	1.00
PUMP_A	1.00	1.00	1.00
SLIDER_N	1.00	1.00	1.00
SLIDER_A	1.00	1.00	1.00
VALVE_N	0.99	1.00	1.00
VALVE_A	1.00	1.00	1.00
ToyConv_N	1.00	1.00	1.00
ToyConv_A	1.00	1.00	1.00
ToyTrain_N	1.00	1.00	1.00
ToyTrain_A	1.00	1.00	1.00

Table 6.6: Detailed result of the GF Environment 3

6.3 Analysis of Loss in Gammatone Accurate Filter

As the datasets used in this study were secondary, it was expected to have a minimal loss throughout the iterations. However, the drastic changes in the loss value were surprising indeed. In the dataset preprocessed with the Gammatone accurate method, a loss value of 1.6 had been observed for the train set. However, after a few epochs, a sudden drop had been observed lowering the value to about 0.9 followed by a somewhat stable decline for the next 20 or so epochs leading to an incredibly stable descend till 100 epochs. The minimum loss value obtained after 100 epochs is close to zero.

On the other hand, for the validation set, environment 1 of the Gammatone accurate method's loss value had not been stable. Fluctuations had been observed throughout the span of 100 epochs. The initial loss was about 1.25 which finally became almost zero. However, the lowest value had not been observed in the last epoch but within the range of 92-95 epochs. An illustration of the phenomena can be found in Figure 6.1.

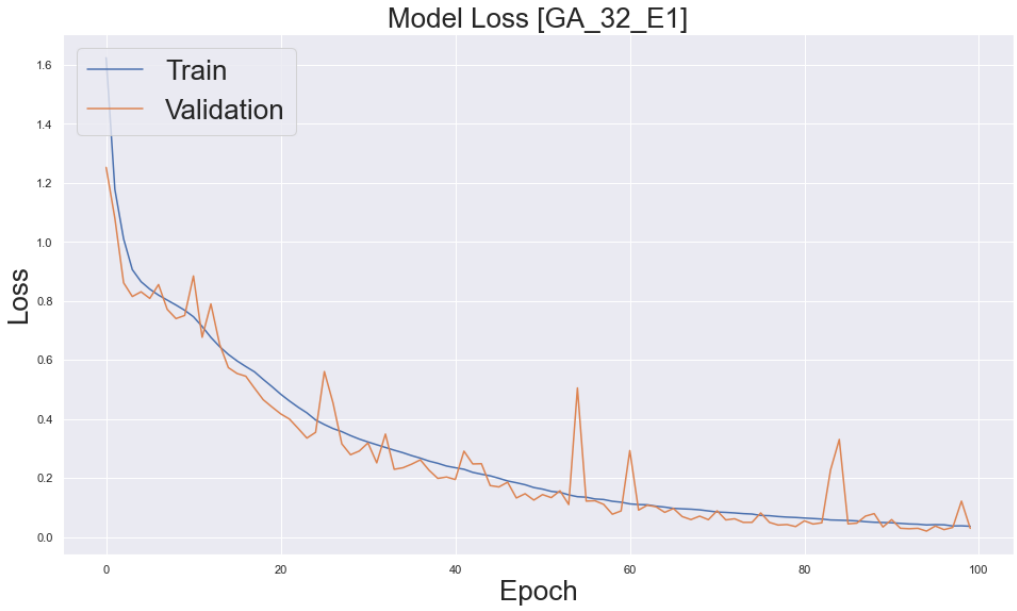


Figure 6.1: GA Environment 1 loss over iterations

After implementing transfer learning our loss curve did change quite significantly. The most noteworthy factor would be the declining stability found in the train set of environments two and three. In both cases, the initial value was about 0.6 and the final value was near zero. However, the third environment had seen a more steep decline in the first few epochs. During validation, both of these environments had confronted sudden spikes in loss with environment two having more frequently. In this environment, loss started at 0.4 with a stable decline to about 0.15 till 5 epochs following by a small spike to about 0.18. At epoch no. 10, a tremendous spike in loss had been observed that surpassed the initial value by exceeding 0.5. However,

it had been short-lived and after a few epochs, the loss dropped down to near 0.05. The environment ran 30 epochs and the lowest loss was found in near epoch 25. The findings are portrayed in Figure 6.2.

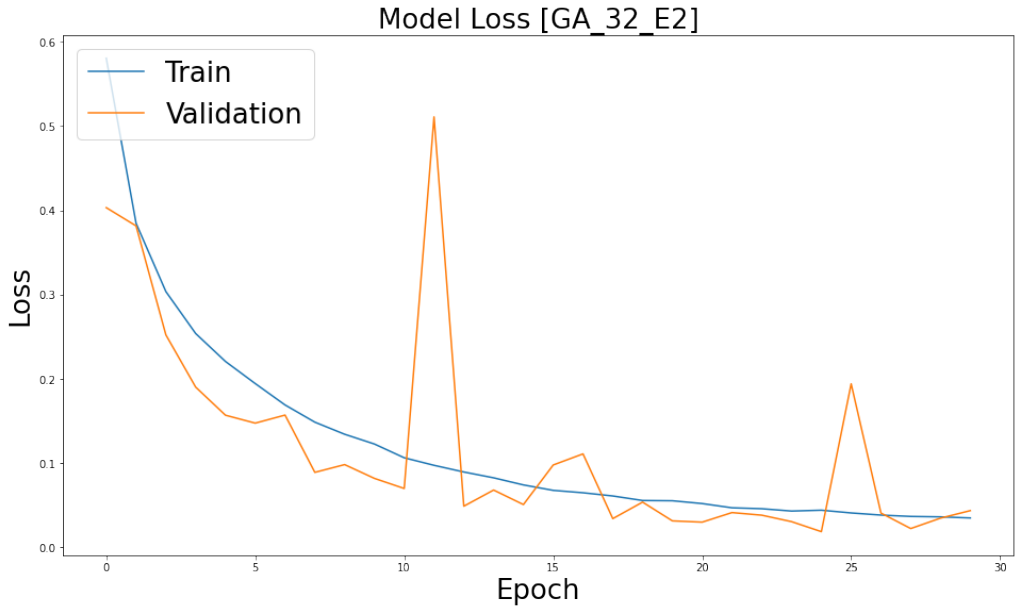


Figure 6.2: GA Environment 2 loss over iterations

Interestingly, in the third environment, the initial loss was even lower starting near 0.3 which peaked at about 0.4 after a few epochs. Moreover, Frequent spikes had been observed and the most prominent one being at around epoch no. 17. Finally, at 30 epoch, there had been another unexpected spike in the loss that exceeded 0.1. Figure 6.3

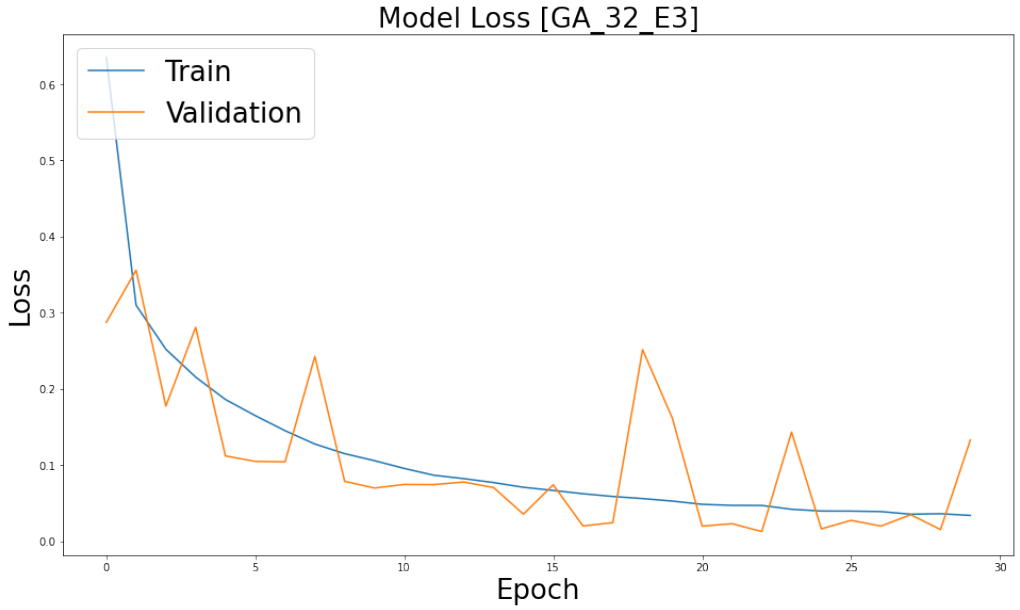


Figure 6.3: GA Environment 3 loss over iterations

6.4 Analysis of Loss in Gammatone Fast Filter

Following the previously observed trend, the environments processed with the Gammatone Fast method triumphed over the other filter by demonstrating a more acceptable loss over the iterations.

From Figure 6.4, in the first environment of Gammatone Fast Bank, the starting loss was at about 1.4 which is 0.2 lower than the initial loss of the Gammatone Accurate's first environment. Within a few epochs, there had been a decline in loss similar to the previous first environment. In the range of 100 epochs that had been run on the test set, the tendency of the loss function had been incredibly stable. However, in the case of the validation set, the same cannot be stated. Frequent spikes had been observed that were mostly present within the first 40 epochs with the biggest one being near 80 epoch that had obtained a value of about 1.1 which had been the highest we had observed for the validation set. After 100 epochs, the loss was near zero for both of the sets.

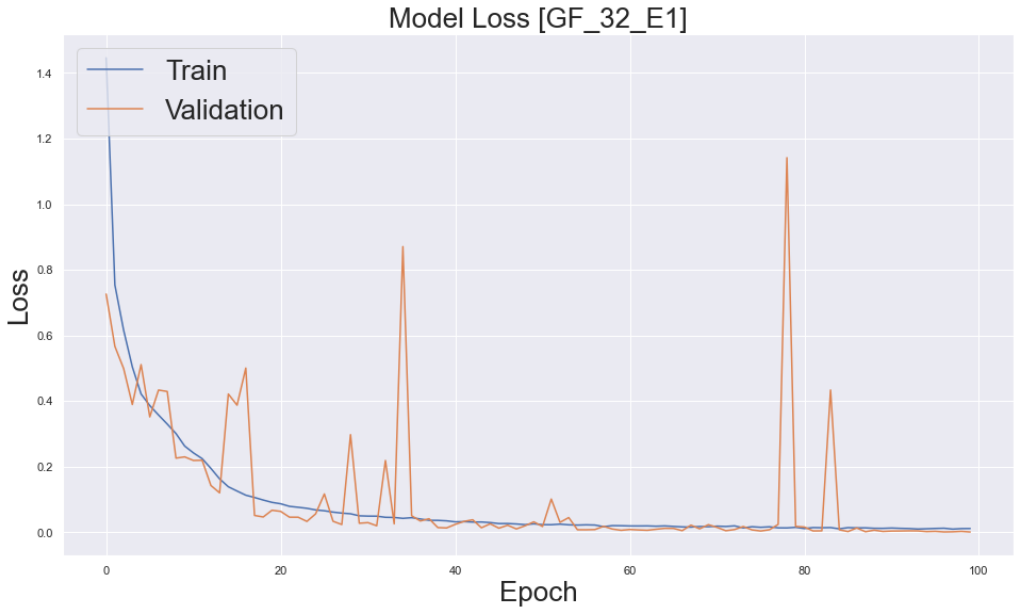


Figure 6.4: GF Environment 1 loss over iterations

The second environment had 30 epochs and no anomaly had been observed in the train set for the time being. The lowest loss was about zero with the highest being at about 0.9. In the validation set, the initial loss was 0.4. During the 30 epochs, there were some fluctuations in the loss value. Of all the fluctuations, the major ones had been observed within the first 18 or so epochs that were followed by a somewhat stable decline leading to the lowest loss of about zero. Figure 6.5 illustrates the situation.

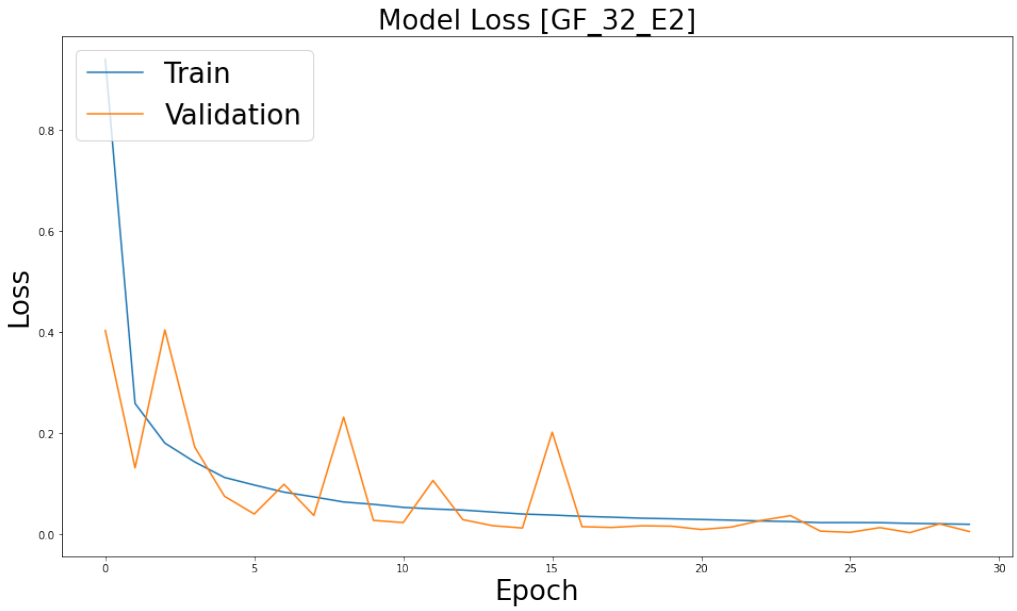


Figure 6.5: GF Environment 2 loss over iterations

Similarly enough, the third environment also displayed an indistinguishable tendency in loss values in the case of the train set. However, the initial value this time was about 0.5, 0.4 lower than the previous environment. When the validation set was run, an astonishing curve was found that had the least amount of fluctuations and was the most stable loss curve among all the loss curves in this study. The initial loss value was 0.3 which was similar to the Gammatone accurate's third environment. Only two major spikes had been observed in this case but none of them exceeded one-third of the initial value. Other than these, no anomalies were observed in loss values. The details can be found in Figure 6.6

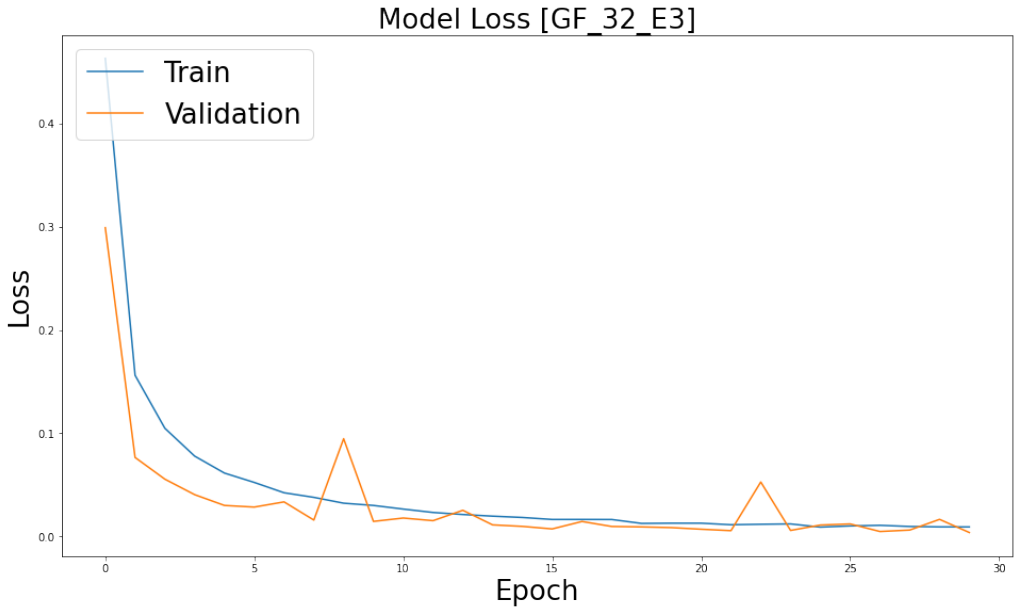


Figure 6.6: GF Environment 3 loss over iterations

6.5 Comprehensive Comparison of Accuracy

Environments where the Gammatone Accurate method was implemented initially had a higher accuracy. However, after implementing transfer learning the accuracy of one of the environments had declined.

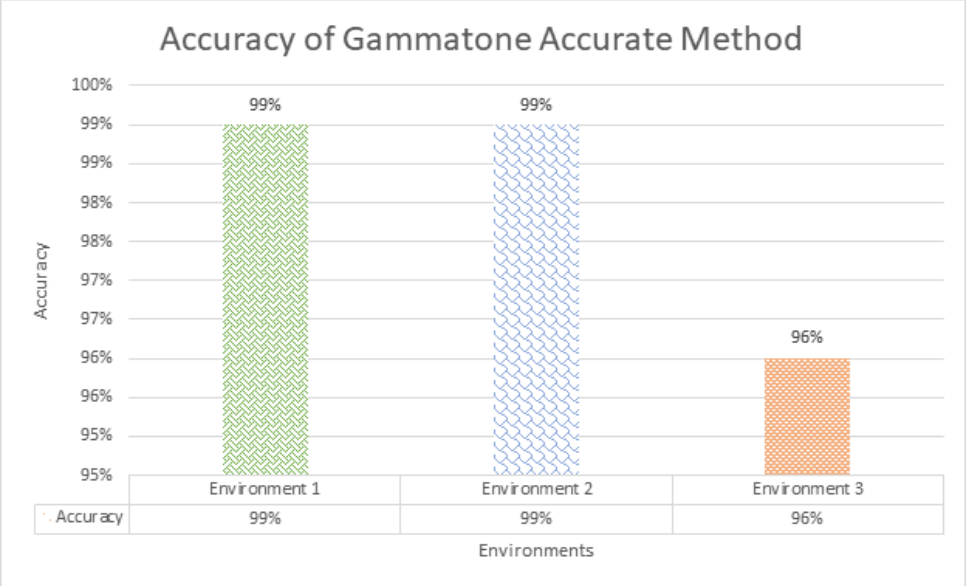


Figure 6.7: Accuracy comparison of environments with Gammatone Accurate filter

On the other hand, Gammatone Fast's environments had a constant accuracy regardless of the implementation of transfer learning.

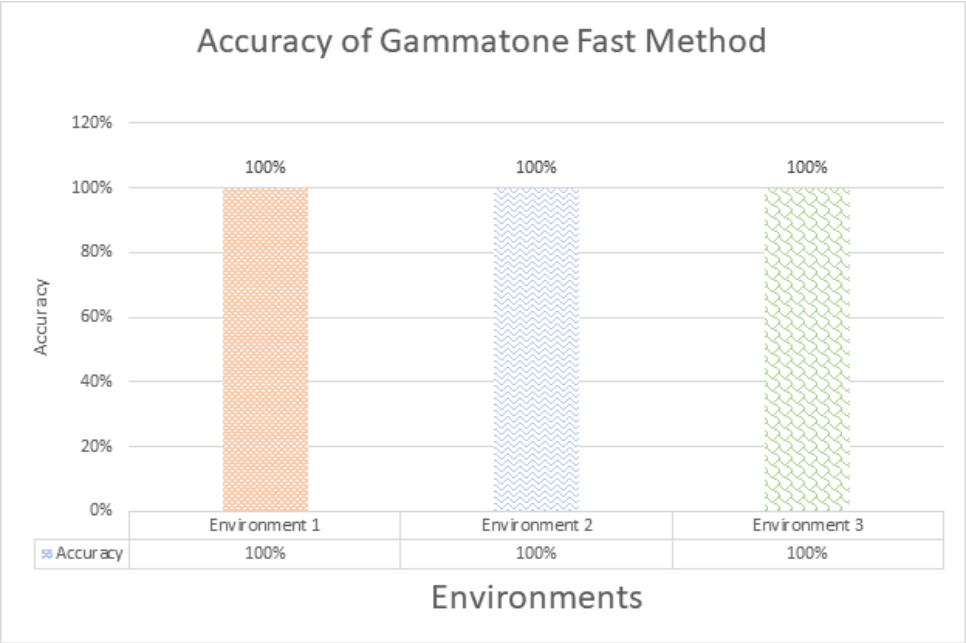


Figure 6.8: Accuracy comparison of environments with Gammatone Fast filter

6.6 Evaluation

The preprocessing method used had a significant impact on the overall performance and acceptability of the environments. In general, the Gammatone Fast's environments were better performant than their Gammatone Accurate counterparts. All Gammatone Fast's environments had a stable accuracy whereas the accuracy of Gammatone Accurate's environments faltered. Furthermore, no fluctuation or sudden demise had been observed in other parameters such as precision, recall, and f1-score for the Gammatone Fast method. When taking a look at the loss values over the iterations, it can be found that the Gammatone Fast method's environments had a more stable loss function than the Gammatone Accurate's environments. Therefore, a conclusion can be drawn stating that in the case of transfer learning, the Gammatone Fast Method worked better in this study.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Industries today demand a system that can not only avoid but also forecast failures. As deep transfer learning networks become more prominent, they are being included in more and more sectors, to improve the efficiency of the specific task and use transfer learning to reduce the time required for the next one. As such it is becoming increasingly vital that the electrical equipment and processes require autonomous monitoring and fault detection techniques. This paper portrayed a deep transfer learning based RNN-LSTM model to extract features enhanced by gammatone spectogram. Also, it was found that while working with gammatone spectogram, modified gammatone fast fourier method was faster and more accurate than the gammatone accurate method. Gammatone fast method provided a performance increase on the suggested model. The proposed model showed consistent performance in all environments. All in all, we can conclude that our RNN-LSTM model enhanced by gammatone fast method can be used as an effective mean of autonomous fault detection in today's industries.

7.2 Future Work

There are a lot of places left behind for improvement in this field. In the future we want to work with more advanced complex models. Recently, hybrid DL models have seen an upsurge. We want to work with these models and see if we can prouduce a more efficient architecture. Also, we wish to see how other signal processing method fare with our proposed model. We have only worked with fault detection methods in this paper. We would like to explore fault diagnosis methods and prevent lives and massive economic loss before they could even take place.

Bibliography

[1] M. Basseville, "On-board component fault detection and isolation using the statistical local approach," *Automatica*, vol. 34, no. 11, pp. 1391–1415, 1998.

[2] S. Malcolm, "Auditory toolbox version 2," *https://engineering. purdue. edu/~ malcolm/interval/1998-010/*, 1998.

[3] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part iii: Process history based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[4] R. Isermann, "Model-based fault-detection and diagnosis–status and applications," *Annual Reviews in control*, vol. 29, no. 1, pp. 71–85, 2005.

[5] A. Rabaoui, M. Davy, S. Rossignol, and N. Ellouze, "Using one-class svms and wavelets for audio surveillance," *IEEE Transactions on information forensics and security*, vol. 3, no. 4, pp. 763–775, 2008.

[6] D. P. Ellis, "Gammatone-like spectrograms," *web resource: http://www. ee. columbia. edu/dpwe/resources/matlab/gammatonegram*, 2009.

[7] N. Laouti, N. Sheibat-Othman, and S. Othman, "Support vector machines for fault detection in wind turbines," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 7067–7072, 2011.

[8] C. Shen, D. Wang, F. Kong, and W. T. Peter, "Fault diagnosis of rotating machinery based on the statistical parameters of wavelet packet paving and a generic support vector regressive classifier," *Measurement*, vol. 46, no. 4, pp. 1551–1564, 2013.

[9] M. Kang and J. Kim, "Reliable fault diagnosis of multiple induction motor defects using a 2-d representation of shannon wavelets.," pp. 1–13, 2014.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. Van de Walle, and S. Van Hoecke, "Convolutional neural network based fault detection for rotating machinery," *Journal of Sound and Vibration*, vol. 377, pp. 331–345, 2016.

[12] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data mining and analytics in the process industry: The role of machine learning," *Ieee Access*, vol. 5, pp. 20 590–20 616, 2017.

[13] GeeksforGeeks, "Introduction to recurrent neural network," *web resource: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/*, 2018.

[14] H. Qiao, T. Wang, P. Wang, S. Qiao, and L. Zhang, "A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series," *Sensors*, vol. 18, no. 9, p. 2932, 2018.

[15] J. Zhu, Z. Ge, Z. Song, and F. Gao, "Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data," *Annual Reviews in Control*, vol. 46, pp. 107–133, 2018.

[16] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "Toyadmos: A dataset of miniature-machine operating sounds for anomalous sound detection," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE, 2019, pp. 313–317.

[17] H. Purohit, R. Tanabe, K. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "Mimii dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," *arXiv preprint arXiv:1909.09347*, 2019.

[18] N. Donges, "A guide to rnn: Understanding recurrent neural networks and lstm networks," *web resource: https://builtin.com/data-science/recurrent-neural-networks-and-lstm*, 2021.

[19] N. P. Kumar, A. S. Kumar, *et al.*, "Fault detection and isolation in industrial processes based on rnn: Deep learning approach," *International Journal of Grid and Distributed Computing*, vol. 14, no. 1, pp. 631–638, 2021.

[20] N. M. M. Ning, "An efficient implementation of gammatone filters,"

[21] L. E. Widman and K. A. Loparo, "Artificial intelligence, simulation, and modeling.," *web resource: https://pubsonline.informs.org/doi/abs/10.1287/inte.20.2.48*, Retrieved : 1990.