DETECTION ON AUDIO ADVERSARIAL EXAMPLES USING DEEP LEARNING

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Jeet Samirkumar Shah

FALL
2020

DETECTION ON AUDIO ADVERSARIAL EXAMPLES USING DEEP LEARNING

A Project

by

Jeet Samirkumar Shah

Approved by:

_____, Committee Chair
Dr. Xuyu Wang

_____, Second reader
Dr. Pinar Muyan-Ozcelik

_____
Date

Student: Jeet Samirkumar Shah

I certify that this student has met the requirements for format contained in the University format manual, and this project is suitable for electronic submission to the library and credit is to be awarded for the project.

_____, Graduate Coordinator _____
Dr. Jinsong Ouyang Date

Department of Computer Science

Abstract

of

DETECTION ON AUDIO ADVERSARIAL EXAMPLES USING DEEP LEARNING

by

Jeet Samirkumar Shah

Nowadays, voice commands are used widely as input methods in various devices to accomplish tasks such as authentication, speech recognition, and pattern analysis. The improvements in Automatic Speech Recognition (ASR) system allows us to perform many tasks which can be done by human listening. These improvements are based on an ongoing development of Deep Neural Networks (DNNs) which has now become the main evaluation technique used in ASR. However, DNNs became more vulnerable to adversarial disturbance as shown by the results of recent research studies. Forcing DNNs to make the false transcription of the audio became an open invitation for the attackers. In recent years, adversarial attacks on audio constitute one of the most frequently happening and most challenging attacks.

Since adversarial attacks are major threat in the speech recognition field, prevention of attacks is an important concern in the fields of machine learning and deep learning. In the first part of this project, we created an adversarial attack system by creating adversarial examples of the original audio to fool the ASR system. We created adversarial examples which can attack a DNN of ASR models in the physical world. As a part of defense strategy, we implemented Generative Adversarial Network (GAN) and Convolutional Neural Network (CNN) an unsupervised learning and supervised learning method

respectively, which can detect adversarial audios using anomalies or perturbations added into the original audio. Implemented GANs achieved better performance results on audio dataset than other implemented GANs and One-Class SVM.

_____, Committee Chair
Dr. Xuyu Wang

_____
Date

DEDICATION

I dedicate my work to my lovely family, Mummy & Pappa, sister Zil who always had faith in me and gave me strength to keep moving forward. My friends who supported and motivated me to accomplish my goals. My teachers & professors, who always tried to help me become a better version of myself.

ACKNOWLEDGEMENTS

I would like to extend my heartfelt gratitude to Dr. Xuyu Wang and Dr. Pinar Muyan-Ozcelik for granting me the opportunity to work on this project. I would not have been able to complete this project without their invaluable guidance and timely feedback.

I would like to acknowledge Dr. Wang for advising and motivating me to overcome the challenges through the course of this project. I was able to improve my technical skills and blossom my abilities in the field of Computer Science while doing this research with his guidance. I would also like to thank him for grooming my presentation and writing skills.

I would like to acknowledge Dr. Muyan for helping me to grasp the foundation for the project. I would like to thank her for providing insights and precious suggestions while I worked on this project.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figures Page

CHAPTER 1

INTRODUCTION

DETECTION ON AUDIO ADVERSARIAL EXAMPLES USING DEEP LEARNING – is an implementation of defense against adversarial attacks. With the technology advances we had in recent years; security became one of the essential necessities. Adversarial attacks are one of the most challenging attacks to defend against. They target speech recognition tasks which involve voice commands used in systems such as Siri, Alexa, Cortana, and Google Voice Assistant. The voice commands relate to many tasks performed on mobile phones and thus, any type of attack on these voice commands can scrutinize our privacy. Hence, resolving these attacks is crucial. This project uses different deep learning techniques to improve the defense strategy against adversarial attacks instead of naïve approaches which are used in previous research.

Before presenting the details of this project, the speech recognition systems, attacks happening on these systems, and the contributions of this project are presented.

1.1 Speech Recognition System

Speech recognition is one of the most popular technologies used in recent years. The machine or device which will try to identify the words or characters uttered by any human is defined as a speech recognition system. The command facility it offers is one of the most notable benefits of speech recognition system. Various speech recognition systems are described in studies conducted by Rafal [1], Shen et al. [2], and Alzantot et al. [3]. With the help of speech recognition systems, we can conduct tasks without needing

any physical efforts. We can control devices such as TV, fan, tubelight, mobile etc. or we can write a book or create documents just by speaking. In fact, creating a document is easier with a speech recognition system since software behind this system can write the word as soon as the word is uttered (i.e., the system types much faster than any human can type).

Bennett et al. [4] developed a system which is used to create a full sentence from the query generated by words. Not just individuals but professionals, who work in the fields such as health, finance, and legal, can also benefit from this technology. For instance, Cohen [5] describes a method to monitor patients' condition using speech recognition. Many professionals, which provide customer care service via call, can use this technology to recognize sentences and the system can provide results accordingly. Also, it can save time for the customer as well as for the employee. The disadvantage of this system is that it cannot identify each word utterance correctly. This disadvantage can benefit the attackers by adding some perturbation or noise in the audio which will make help them successfully attack the system. Once hackers have gained unauthorized access within an application, then the user data would get compromised. Therefore, there is a dire need to detect such attacks which can help in protecting the security of ASR system.

1.2 Adversarial Attacks

Speech recognition systems accept frequency data as input vectors of an audio to interpret the words spoken in audio. Adversarial vectors are the reconstruction of an input vectors which are used to do adversarial attacks (i.e., to make speech recognition system interpret the words incorrectly). Adversarial vectors can be converted to adversarial audio

using Python libraries. These adversarial examples are used to fool the deep learning model by adding small perturbation into the original audio (or original image in systems that involve image recognition instead of speech recognition). They are created with noise which makes a deep learning model to misclassify them and make a false prediction.

For instance, suppose a self-driving car gets into an accident. Assume that the reason behind this accident is that it had ignored a stop sign before entering the highway. There was a stop sign, but someone placed an image over that stop sign, which looks like a parking prohibition sign for the self-driving agent while for humans it looks like a little dirt. Hence, when someone added a little dirt over the sign and its meaning for the software has totally changed.

Likewise, adding some noise model will change the prediction of the transcription of original audio from "ABCD" to "KDCD" as shown in Figure 1. The threat of adversarial attacks on speech recognition system is shown in studies conducted by Alzantot et al. [3], Carlini and Wagner [6], and Yakura and Sakuma [7] by creating an adversarial attack system. They created adversarial samples by adding noise or perturbation which can help attack speech recognition systems successfully. Since these attacks can be a major threat for speech recognition systems, the need to defense against these attacks is undeniable.

Many defense techniques have been proposed by the prior studies. For instance, Kwon et al. [8] proposed audio modification method to defend against adversarial attacks. Vlasveld [9] discussed the unsupervised One-Class SVM and Generative Adversarial Networks (GANs) are proposed in studies conducted by Rocca [10].

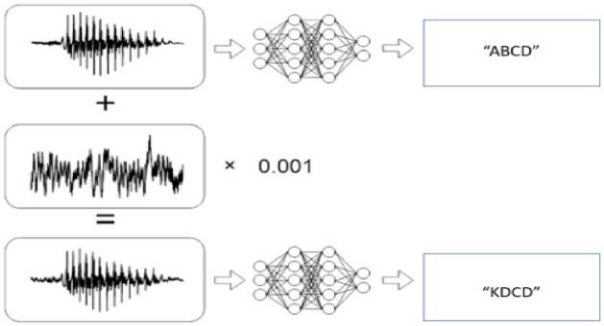

Figure 1: Audio adversarial example [6]

However, the results of GAN reported by Rocca [10] are not efficient for the audio dataset. Therefore, in this project, we proposed another GAN method with the cycle consistency for unsupervised learning and Convolutional Neural Network (CNN) for supervised learning.

1.3 Contributions

The main contributions of this work include:

- Building an attacking system which can fool the speech recognition system by creating adversarial examples from raw original examples in the LibriSpeech dataset by Povey [11].
- Introducing Band-Pass filter, impulse response, and White Gaussian noise methods to generate adversarial examples.
- Preprocessing the dataset of original and adversarial examples which can help us to implement defense methods more efficiently. Preprocessing includes Requantization and various feature extraction techniques such as MFCC, chroma, mel spectrogram, tonnetz etc.

- Introducing the supervised defense strategy that uses CNN which has been trained on adversarial examples. In addition, introducing unsupervised strategies which have never seen adversarial examples before like One-Class SVM, GAN developed by Zenati et al. [12] and our proposed GAN with the cycle consistency.
- Providing experimental results which show how efficient our adversarial attack is and how accurate our proposed GAN is in detecting the adversarial examples compared to the GAN proposed by Zenati et al. [12].

The rest of the report is structured as follows: In Chapter 2, we discuss related work. Chapter 3 presents the system architecture of this project which includes adversarial attacks, various defense strategies, GAN architecture, deep learning models such as CNNs, and One-Class SVM. Chapter 4 introduces the experimental setup and results for the implemented methods. The last few chapters contain conclusions and future work that can be done to improve speech recognition system.

CHAPTER 2

RELATED WORK

Qin et al. [13] introduced audio adversarial examples which helped in mitigating attacks on audio using automatic speech recognition. By adding minor distortion, they convert the audio waveform into any target transcription with 100% success. Carlini and Wagner [6] demonstrated that defensive distillation does not significantly increase the robustness of neural networks by introducing three new attack algorithms that were successful on both distilled and undistilled neural networks with 100% probability. Solanki et al. [14] demonstrated low-cost attacks that misuse speech recognition APIs for implementing fully automated attacks against popular captcha schemes. Overall, they argued that it is necessary to explore alternative captcha designs that fulfill the accessibility properties of audio captchas without undermining the security.

In another study, Alzantot et al. [15] used a black-box attack optimization algorithm to generate semantically and syntactically similar adversarial examples that can misclassify well-trained sentiment analysis and text generation models with a success rate of 97% and 70%, respectively. Moreover, they experimented on 20 human annotators who are used as a testing setup to classify the example. It resulted in 92.3% of successful sentiment analysis and the examples had 0.1% of change compared to each other.

One-Class SVM was implemented by Zenati et al. [12] to find the outliers present in the dataset. They developed it on the Mnist image dataset and KDD, the network intrusion dataset. Using this method, they achieved the result of an average 74% precision

count for both datasets. Efficient Generative Adversarial Networks (GANs) were introduced also in the study conducted by Zenati et al. [12]. They also developed the GAN model which can detect anomalies in the dataset of image and network intrusion. They developed it by using a discriminator and a generator. For this purpose, they created a model that optimized the loss value which can be calculated from the formula *Total loss = Discriminator loss + Generator loss*. Using this model and optimization loss function, they achieved 90% accuracy in detecting anomalies as adversarial data for the Mnist and KDD dataset.

CHAPTER 3

PROJECT ARCHITECTURE

3.1 System Architecture

This section provides the system design for this project which is outlined in Figure 2. The system consists of two major modules, namely, attack and defense. For the attack module, we generated the adversarial examples which will try to attack the ASR system using the adversarial algorithm as well as adversarial noise. The method to generate adversarial examples is discussed in Section 3.4.

In today's technology-driven world, the speech recognition system is vulnerable to attacks. Hence, this project implements two defense approaches which will try to detect such adversarial examples. The first approach is to train a model on adversarial examples along with the original examples, the implementation for which is done using Convolutional Neural Network (CNN). The second approach is for the case where the models have not been trained on adversarial examples. This strategy can be executed using algorithms like One-Class SVM and Generative Adversarial Network (GAN). Sections 3.4, 3.5, and 3.6 discuss data preprocessing, and the implementation of attack and defense modules, respectively.

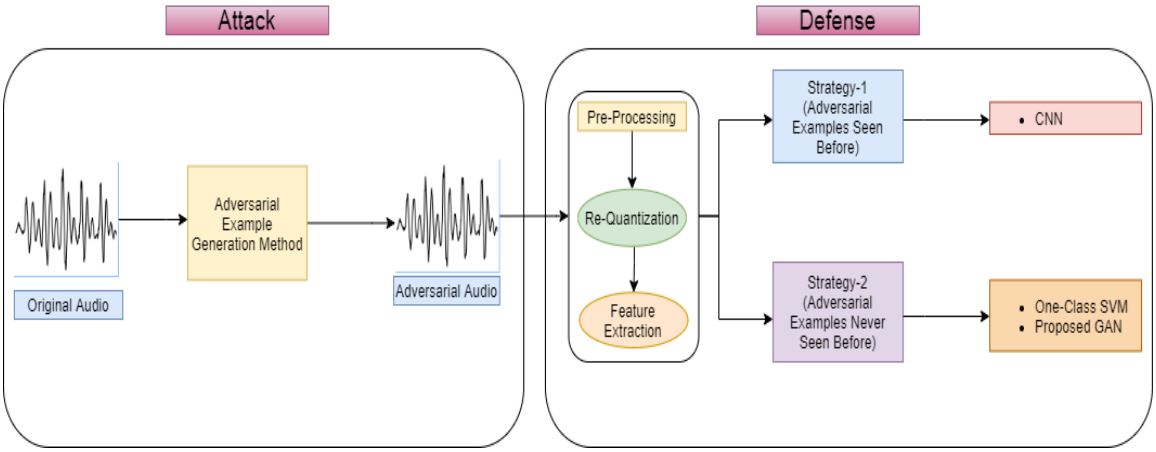

Figure 2: System architecture

3.2 Dataset

In this project, LibriSpeech [11] dataset is used. LibriSpeech dataset has an audio file with the *.flac* extension. So, the first step is to convert the *.flac* file to a *.wav* file. This can be done by using the Python's Pydub library. The dataset has a total of 905 original audio files which do not have purposefully added noise. Figure 3 shows a waveform of the sample original audio file from the dataset.

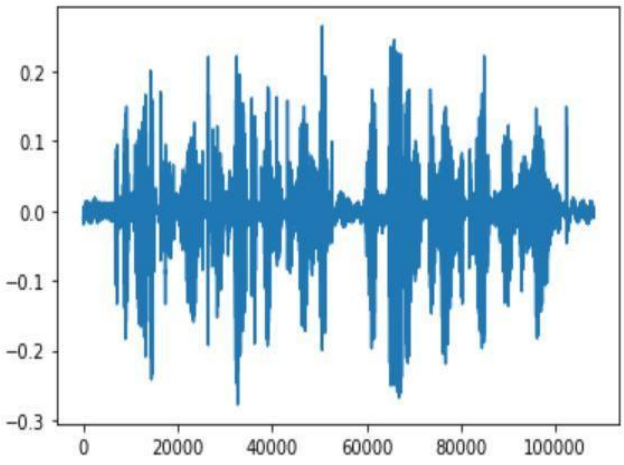

Figure 3: Sample original audio waveform from dataset

3.3 Mel-Frequency Cepstral Coefficients (MFCCs)

In sound processing, "the Mel-frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency" [16]. The coefficients that are relatively collected from MFC are called Mel-Frequency Cepstral Coefficients (MFCCs). Mel scale is the scale that tries to relate the frequency which is generated from the actual frequency signal. The rate change present in spectral bands is called Cepstral. Figure 4 shows how the MFCCs can be generated from the audio signal. As shown in Figure 4, Cepstral Coefficients are MFCCs. Figure 5 shows the MFCC feature for the sample audio where the x-axis shows the frequency-time in milliseconds and the y-axis shows the MFCC coefficients index.

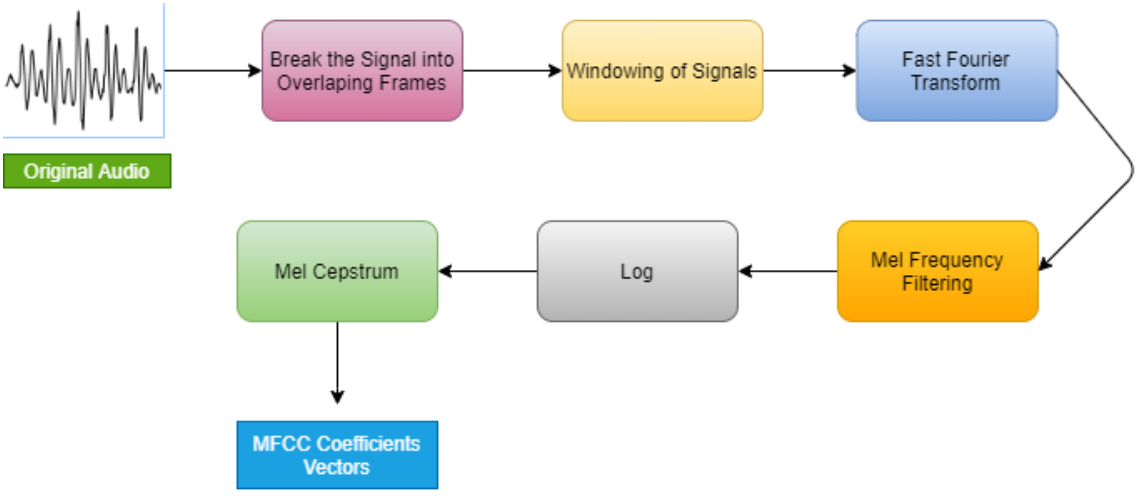

Figure 4: MFCCs generation

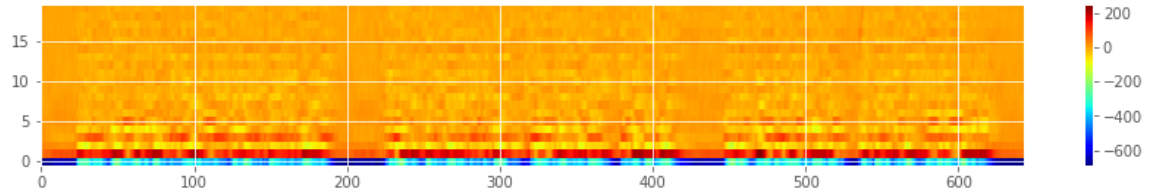

Figure 5: Sample original audio's MFCC feature

This project uses the Mel-frequency Cepstrum Coefficient (MFCC) to extract the features from audio waveforms to generate adversarial examples as explained in Section 3.4.

3.4 Generation of Adversarial Example

Attackers can attack any speech recognition system by adding some noise or perturbation into the original audio which leads the deep learning model to misclassify the true label of the audio. Such examples are generated using the Carlini and Wagner method [6]. Carlini and Wagner used DeepSpeech developed by Amodai [17] ASR and tried to deceive the system, while in this project we used Automatic Speech Recognition developed by Rafal [1].

ASR is a speech-to-text engine that takes audio waveform as an input and gives an output in the text form. The ASR used in this project was built using Google's TensorFlow framework which makes the implementation of it much easier. This ASR system does not require sensors or IoT devices to calculate the background noise or variation in the speech. The model directly learns a function by training with a Recurrent Neural Network (RNN) which is more robust in such cases. RNN is a well-optimized system since it uses multiple GPUs for both training and inference purposes. Moreover, it uses data synthesis techniques that allow it to collect data with a variation for training. RNN models take a generated

spectrogram image of the audio which is an array of pre-defined shapes. This data is processed by two convolutional layers. BatchNormalization and ReLU layer follow each convolutional layer. Then the output of the convolutional layer is reshaped so that it can be fed to the Bidirectional layer. There are five Bidirectional layers each followed by a dropout layer. Finally, there is a Dense layer which will produce the final output of the defined shape.

Generation of adversarial examples is based on a loss function which is defined to decrease the training loss using the process of gradient descent. This project uses the Mel-frequency Cepstrum Coefficient (MFCC) to extract the features from audio waveforms. MFCC optimizes the waveform of the entire audio using Adam optimizer. The loss function for the ASR system is calculated by Equation 1.

$$argmin_v Loss_f(MFCC(x+v), l) + \varepsilon||v|| \qquad (1)$$

In the above equation, x is input audio, v is perturbation and l is a target phrase. $MFCC(x+v)$ extracts the features and feeds it to the ASR system loss function along with the target phrase. Unfortunately, the MFCC method will fail to run successfully for an over-the-air system. However, robust adversarial examples can be generated for on an over-the-air system as explained in Section 3.4.1.

3.4.1 Robust Adversarial Example

Robust adversarial examples are generated with different techniques. These techniques will generate adversarial examples using optimized adversarial loss function. Before feeding this adversarial audio to the ASR system, three methods are applied to it to

make adversarial audio work on an over-the-air system. These methods are band-pass filter, impulse response, and White Gaussian noise.

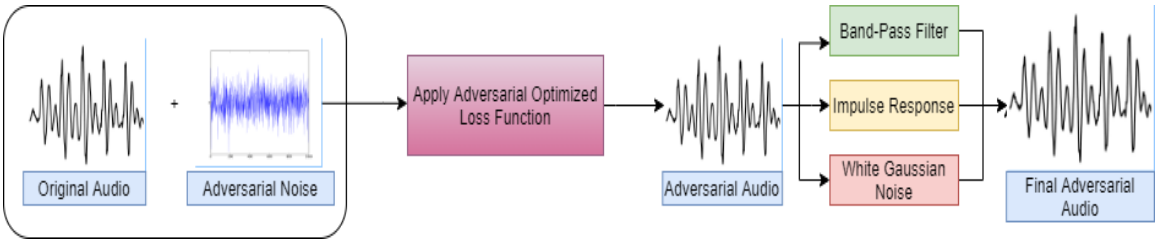

Figure 6: Adversarial example generation method

3.4.1.1 Band-Pass Filter

Nowadays, speakers and microphones are so efficient that the audio which has a frequency range of less than 20Hz and greater than 20000Hz will not be generated (the audio in this range is not audible to humans). And the microphone will eliminate the audios outside this range to reduce the perturbation or noise. Therefore, an adversarial example does not work in such a case. Band-Pass filters are helpful in such scenarios. "A band-pass filter (BPF) is a device that passes frequencies within a certain range and rejects frequencies outside that range" [18]. In this project, a band range from 1000Hz to 4000Hz is used which shows less perturbation.

$$argmin_v Loss_f(MFCC(x+v), l) + \varepsilon||v|| \qquad (2)$$

Using Equation 2 we can generate better robust adversarial examples. Equation 2 is similar to Equation 1. The only difference here is that input x is $x + BPF(v)$, where BPF ranges from 1000Hz to 4000Hz. This way, we can generate robust adversarial examples which will be better even though the bandwidth is outside the human audible range.

3.4.1.2 Impulse Response

"An impulse is a very loud and short sound event that is used for testing the response to sound in a room or to test the effectiveness of an acoustical system" [19]. Using this method, we can test impulses in different environments, and then we can feed it for the generation of adversarial examples to make them more robust.

Therefore, here the predicted values are taken over the impulses recorded in various environments. So, the loss function will change from what we have defined in the Band-Pass filter.

$$argmin_v E_{h\sim H}[Loss_f(MFCC(\bar{x}), l) + \varepsilon||v||] \qquad (3)$$

In Equation 3, $\bar{x} = Conv_h(x + BPF_{1000\sim4000Hz}(v))$, H is the impulse responses collected from the dataset and $Conv$ is the convolution of the impulse responses.

3.4.1.3 White Gaussian Noise

Many random processes occur in nature. So, to emulate the effect of those processes, White Gaussian Noise is used. It is also utilized to train the robust adversarial audio examples towards the background noise. Hence, the equation generated after impulse responses can be extended using Equation 4:

$$argmin_v E_{h\sim H, w\sim N(0,\sigma^2)}[Loss_f(MFCC(\bar{x}), l) + \varepsilon||v||] \qquad (4)$$

In Equation 4, $\bar{x} = Conv_h\big(x + BPF_{1000\sim4000Hz}(v)\big) + w$. This way, it can be made sure that our generated robust adversarial example shall not be affected by background noise generated by the microphone or room environment. Here, White Gaussian noise is

added before convolution, but we can add it after the convolution to make the process more optimized and efficient.

We generated such examples using 905 original audio files which can make ASR systems vulnerable. Now, we have a total of 1810 original and adversarial audio files which are used as a dataset for defense. Before feeding the dataset to defense methods, preprocessing is done on the audio files to improve the results as shown in Section 3.5.

3.5 Pre-Processing of Audio Files

Two techniques named Re-Quantization and Extracting features are used as a part of the pre-processing of the audio.

3.5.1 Re-Quantization

Re-Quantization is an important process in speech signal processing. It is a technique of converting large input values to a smaller set of values with the help of rounding and truncating the data values. The quantizer is an algorithm that is used to perform the quantization process. This process will help us in removing the background noise that humans may not be able to listen to.

In this project, we have all the audio files with a word length of 16 bits. With the help of the re-quantization process, they were converted to a word length of 8 bits. It is quantized with a uniform mid-tread quantizer. In a uniform mid-tread quantizer, quantization levels are uniformly spaced, and the quantization levels are in the odd numbers.

3.5.2 Feature Extraction

For the defense part, spectrogram images of the 1810 audio files were generated to feed it to the Convolutional Neural Network. The visual presentation of the spectrum of frequencies which shows the strength of the signal as it changes with time is called a spectrogram. The spectrogram can be used to determine how the energy of the signal changes over time which shows the duration when signal strength is loud (noisy), normal, or less noisy.

Figure 7 shows a sample spectrogram image that is used as input to the Convolutional Neural Network. In this figure, X-axis shows the time of a signal with a gap of 0.5 seconds and the Y-axis shows the frequency of the signal for that time.

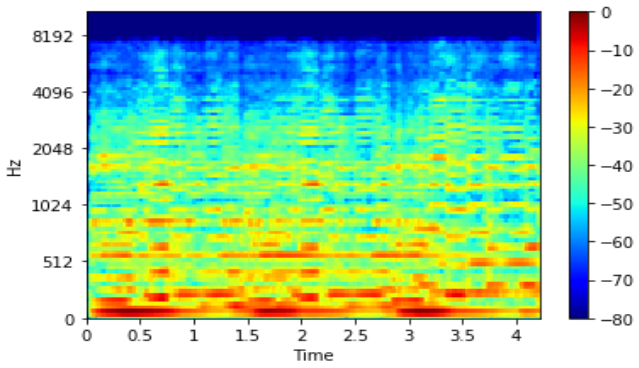

Figure 7: Sample input for CNN

Also, we created a separate dataset of these 1810 audio files' features like MFCCs, Chroma, Mel, Contrast, Tonnetz, etc. which can be given as an input to One-Class SVM and Generative Adversarial Network. This will reshape the dataset to (1810,193).

Chroma is an important feature in audio processing which shows the spectrum of frequencies distributed in 12 bins representing the 12 different chromas of the audio signal.

Capturing the harmonic and melodic frequencies of music is the main characteristic of the Chroma. Mel is a mathematical scale that measures the non-linear changes occurring in the frequencies over time. Humans can listen to a signal or sound when the frequency signals are at equal distance from each other. These signals construct the Mel scale of that signal. The contrast feature is used to analyze the root mean square error between the foreground and background music by partitioning the audio track into two parts. This feature would be useful to recognize the perturbations present in the audio signal. It is helpful to recognize the audios which have adversarial perturbations. The Tonnetz feature is also known as the tonal centroid feature. The representation of the features is dependent on the harmonic changes in the audio signal. It represents the 6-dimensional chroma features among two-dimensional vectors. Thus, these features are used as an input to train the GAN and One-Class SVM.

3.6 Defense Strategies Against Adversarial Examples

This section provides a brief description and implementation of the Convolutional Neural Network which are trained on adversarial examples. It also discusses the implementation of other models such One-Class SVM and GAN which have never seen adversarial examples before.

In the case of CNN, a supervised deep learning technique is used, while an unsupervised Deep learning technique is used for One-Class SVM and GAN. We proposed an efficient GAN model that outperformed both One-Class SVM and the GAN which was developed by Zenati et al. [12]. A classification method is used to analyze the results.

3.6.1 Convolutional Neural Network (CNN)

A CNN is one of the deep learning algorithms which are used to handle image datasets. CNN can take images as input and train the model based on the image features. It will assign importance to each feature object to differentiate that feature object from another feature object. Differentiating each feature by breaking the feature object into parts can be done by using a convolutional layer. This will help CNN to predict the class of the image if the model is a classification or mean score if the model is a regression since the model will learn accordingly. The CNN architecture is nearly related to neurons available in a human brain [20].

For the input, we created spectrograms of all 1810 audio files and created a Pandas dataset which consists of spectrogram images with respect to their original class. Since the input shape is 4-dimensional for CNN, we reshaped the dataset to 64*64 pixels with 3 channels. Then, the dataset was split into 70% training which has an input shape of (1267,64,64,3) and output shape of (1267,1), and 30% test data has the shape of (543,64,64,3) and (543,1) for input and output data respectively with the mixed partition of adversarial and original audios. The developed CNN model is shown in Figure 8.

The layers available in CNN architecture are as below:

1) Convolutional Layer – An image of shape 64*64 pixels is passed to the layer which will scan the pixels and fetch features from it and store them to map. It will help the model to predict the actual classification or class for the corresponding image. The convolutional layer’s output depends on the stride

defined in the model. The stride count has been predefined to 2. This means the layer will filter pixels one at a time which gives an output of shape 62*62 pixels.

2) Pooling Layer – The pooling layer will help the model to perform downsampling of the image features. This layer helps the model to eliminate outliers or non-critical information in the prediction of the class. This layer will help the model to maintain the most important features of the image. So, the output of shape 31*31 pixels only contains the features related to the prediction.
3) Fully Connected Input Layer – This layer is also known as a flatten layer which will take the output of the convolutional and pooling layers, then reshape them into a single-dimensional array which would be the input for the next layers.
4) First Fully Connected Layer – This layer takes the inputs from the feature map which have been created from the convolutional layers and then using pre-trained models or weights to predict the correct classification for the image.
5) Fully Connected Output Layer – This layer will give the final predictions for each class of shape (543,1).

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 30, 30, 64)        8256
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 13, 13, 128)       73856
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 4, 4, 256)         295168
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 2, 2, 256)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 500)               512500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 501
=================================================================
Total params: 891,177
Trainable params: 891,177
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: Proposed CNN model

Figure 8 shows that we used 4 convolutional layers which will help in extracting important features by reducing the image size for processing. This ensures that the data which is critical for prediction is not lost. The Pooling layer is responsible for reducing the spatial size of the Convolved feature. The Flatten layer will convert the 4-D shape into a 2-D shape which can be reduced to a size of one as we have one class to predict either original or adversarial using dense layers. A dropout layer is available in the model to avoid the overfitting of data. Moreover, we used Adam optimizer and Rectified Linear Unit (ReLU) activation layer to update the network weights based on training data. After defining the CNN model, we trained the CNN model on the training data and predicted class for each data on the test dataset. Binary-CrossEntropy was used to calculate the training loss.

3.6.2 Defense Strategy for Unseen Adversarial Examples

One-Class SVM and GAN models are unsupervised learning algorithms which take only normal data as an input and train model based on that data. Whenever the data is different from the trained data, it will classify that data as an outlier. In our case, adversarial examples are considered as outliers.

We generated a dataset that contains the features like MFCCs, Chroma, etc. of shape (193,) for each audio which we discussed earlier in Section 3.5.2. Therefore, the shape of the dataset is (1810,193) which contains normal as well as adversarial examples. Now, we divided the dataset into train and test data where train data only consists of original audio examples' features while test data contains both original and adversarial examples' features. After dividing, adversarial examples are removed from the train data

as our model can only be trained on original examples. This means that our models have never seen adversarial examples before. After pre-processing of feature generation and train-test split of data, the shape of train and test input-output data are (789,193), (789,), (509,193), and (509,) respectively. Next, let us introduce the One-Class SVM and GAN method.

3.6.2.1 One-Class SVM

There are many algorithms and methods which can solve or predict multi-class classification. Deep learning methods identify test data for the number of classes by using features or training data. But in the real world, many datasets have only one class for the training data available. We must distinguish the test data from those datasets to find out whether it is like training data or is it different than train data. One-Class Support Vector Machine predicts these types of datasets.

For example, suppose there is a lab having several computers under constant surveillance. The task of the surveillance system is to keep track of when something goes wrong; what is the cause of the problem. It is easy to gather the training data of the situations in the case where there is no problem at all. But when something is going wrong it is nearly impossible to gather data. There is no way we can train and predict the state of the computer since we do not have enough information for situations when something goes wrong.

To solve this kind of problem, the One-Class classification method is introduced which will train on one class dataset. At the time of prediction, if it will see different data than the trained one, it will predict it as out-of-class. This method creates a hyperplane F

which has all the points available in feature space and out-of-class labels will have maximum distance from the origin. The SVM function would return +1 for the data which is in a small region near to origin or else -1. Vlasveld [10] introduced Equation 5.

$$f(x) = sgn\left(\left(w \cdot \emptyset(x_i)\right) - \rho\right) = sgn(\sum_{i=1}^{n} \alpha_i K(x, x_i) - \rho) \qquad (5)$$

In equation 5, x is training data, $\emptyset$ stands for feature map $x \rightarrow f$, w is regularization factor, ρ shows the mean amplitude displacement, and K is the kernel type. Equation 5 creates a hyperplane characterized by w and ρ which has maximal distance from the origin in feature space F and separates all the data points from the origin [9].

In this One-Class SVM, we define a parameter for which we used the Radial Basis Function (RBF) as a kernel type. For the parameter 'nu' which is the upper bound on margin errors and lower bound on vectors related to the length of training data, we took the length is taken as test data divided by trained data. We set the Gamma parameter to 0.05 which is the coefficient for the kernel. After defining the parameter for the One-Class SVM, the model is trained and the class for each sample in the test dataset is predicted. The results are discussed in the Chapter 4.

3.6.2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks are models that can generate new data by itself similar to train data or features fed to the model. Before understanding GANs, let us understand how random variables are generated from distribution data.

The inverse transform method is an efficient method that will generate variables even though uniform variables are too simple. Uniform variables are variables that are

generated through an algorithm whose features and properties are nearly close to the random number sequences. A computer can generate a sequence of numbers by using a pseudorandom number generator that approximately follows a uniform random distribution between 0 and 1 which is described by Rocca [10]. To generate more complex variables, there are mathematical algorithms like the inverse transform method, rejection sampling, and Metropolis-Hasting algorithm.

In GANs, there are two major components: Discriminator and Generator. The discriminator is a component that will try to predict true labels or classification for the actual inputs while the Generator will always try to deceive the discriminator by generating complex random variables. Now, the question is how discriminator is deceived by generators. This leads to exact predictions as if we train data directly on generators. In this case, the discriminator knows exactly which labels are true and which are generated labels for any given point. However, if the distribution for any given point has a distance more than the threshold, then the discriminator will be able to classify labels correctly for that given point. If we minimize the distribution for any given point, it will make the generated distribution distance as small as the original one. If the two distribution distances are close to zero for all the given points, it would be nearly impossible for the discriminator to identify the class for the given point.

Let us understand the specifics of the generator and discriminator. The generator is a neural network that models a transform function [10]. It takes a simple random variable as input and once trained; it must return a random variable that follows the targeted distribution [10]. The discriminator is a separate neural network that models a

discriminative function. This function would take input from the original data at a certain point and return a predicted label for that point as a true label. Both neural networks can be trained on corresponding inputs but with opposite goals. The generator would try to deceive the discriminator. Thus, it will try to minimize the classification error because it is trained in that way. On the other hand, the discriminator is detecting the fake features or data which will try to maximize the classification error and it will update weights of train data at each iteration as well. Figure 9 shows the architecture of the GAN.

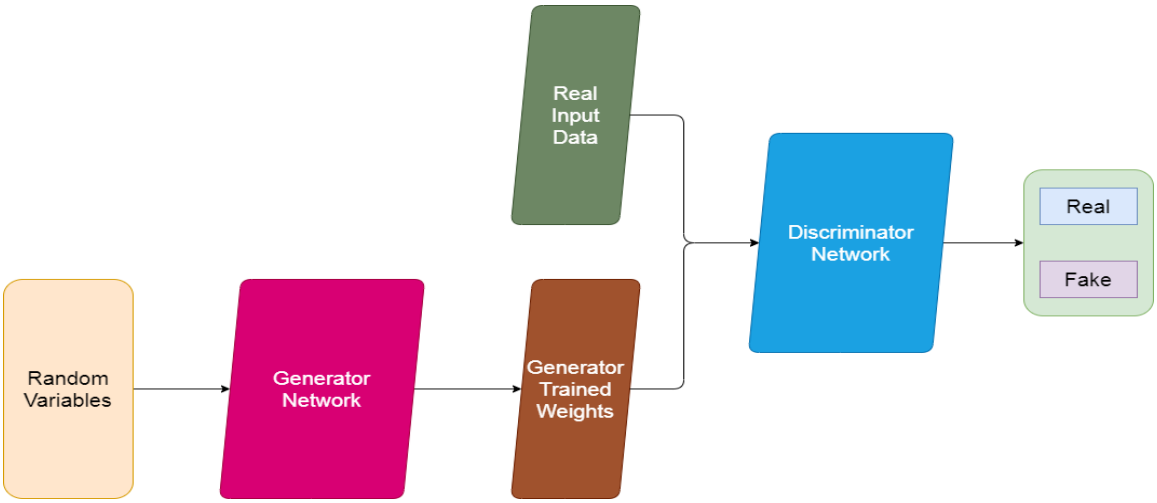

Figure 9: GAN architecture

Figure 9 shows that GAN is a generative model which are used for the estimation value via adversarial methods which include discriminator D and generator G models. The generators are trained on random data while real input data and trained generator data are used to train discriminators. The generator tries to capture all possible values and the probability of occurrence of that certain data. It tries to create a graph of how data is distributed. Discriminator evaluates the probability of data coming from a training dataset or a generator. The GAN developed by Zenati et al. [12] simply estimated the loss which

is generated from the discriminator and generator simultaneously. In general, their total loss would be the addition of discriminator loss and generator loss. The loss optimization function for the model used is:

$$V(D,E,G) = E_{x\sim px}\,[E_{z\sim pE\,(x)}\,[log\ log\ D(x,z)\,]] + E_{z\sim pz}\,[E_{x\sim pG\,(z)}\,[1 - log\ log\ D(x,z)\,]] \qquad (6)$$

In Equation 6, px is the distribution over the data, pz is the distribution over the latent representation, and $pE(x|z)$ and $pG(x|z)$ are the distributions induced by the encoder and generator respectively [12]. This loss value is not effective for the randomly distributed audio data features with added perturbation. For the audio dataset, this method [12] is getting 80% accuracy only. This means that it fails to classify or detect adversarial examples.

That is why we developed another GAN method using the cycle process which can solve the problem caused by GAN introduced by Zenati et al. [12]. The developed GAN will learn encoder E at the same time the discriminator and generator are being trained on the input data x. The encoder will try to map the input data x to a hidden representation of z which will decrease the computation time for testing. At the training time, an optimization function is required to optimize the data loss for the training data and random noise from the normal distribution of data z which has the sample size same as the training data. For that, the optimization function $MAX_D\ MIN_G\ V(D,E,G)$ is defined as below:

$$V(D,E,G) = E_{x\sim px}\,[E_{z\sim pE\,(x)}\,[log\ log\ D(x,z)\,]]$$
$$+ MAX\,(\left(E_{z\sim pz}\,\left[E_{x\sim pG\,(z)}\,[1 - log\ log\ D(x,z)\,]\right]\right),$$
$$(E_{x\sim px\,(x)}\,[\left|D\left(G(x)\right) - x\right|] + E_{z\sim pz\,(z)}[\left|G\left(D(z)\right) - z\right|])) \qquad (7)$$

In Equation 7, px is described for the distribution of the training data, pz for distribution of the random noise generated from the normal distribution which is also known as the hidden representation of data. The optimization function is an addition of the distribution of training data weights trained in discriminator and maximum optimized distribution of random noise data weights generated from generators and weights generated from the cycle process. When we try to optimize generator weights, it will not optimize weights of the data which has background noise in raw audio. So, actual data is passed into the generator as input, and the input of the discriminator would be the output of the generator. Similarly, random noise data is passed into the discriminator, and the output of the discriminator is passed to the generator. This process would train discriminator and generator for the inputs having background noise in raw audio like some of the original examples. By doing that, we can efficiently optimize generators. This is called a cycle process. The score function which will decide whether it is an adversarial example or not is discussed in Chapter 4.

CHAPTER 4

EXPERIMENTAL SETUP AND RESULTS

4.1 System Prerequisites

The Table 1 shows the list of technologies, packages, and libraries used along with the purpose of using them.

Table 1: Software technologies used

Technology Name	Usage	Version
Numpy	Data Preprocessing	1.19.2
Pandas	Data Preprocessing	0.24.2
TensorFlow	Machine Learning Framework for Implementation	1.15.0
ASR by Rafal [1]	Python Package for Speech Recognition System	1.0.4
Scipy	Machine Learning Library for Audio Processing	1.4.1
Keras	Neural Network API	0.3.3
Pycharm	Open-Source Web Application to Write Python Code	3.7
Jupyter Notebook	Open-Source Web Application to Write Python Code	3.7
Scikit-Learn	Machine Learning Library for Evaluations	0.21.2
Librosa	Machine Learning Library for Audio Processing	0.7.2
Python 3.7	Programming Language	3.7
Pydub	Machine Learning Library to Convert Audio Type	0.24.0
Matplotlib	Data Visualization	3.1.0

4.2 Result Analysis of Generated Adversarial Examples

To visualize the difference between original and adversarial examples, Figure 10 and 11 show a comparison of sample original and adversarial audio using Waveplot and MEL Spectrogram respectively.

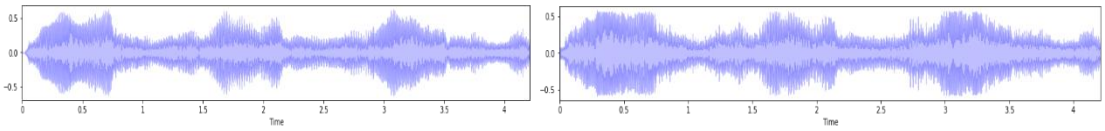

Figure 10: Waveplot comparison of original and adversarial audio

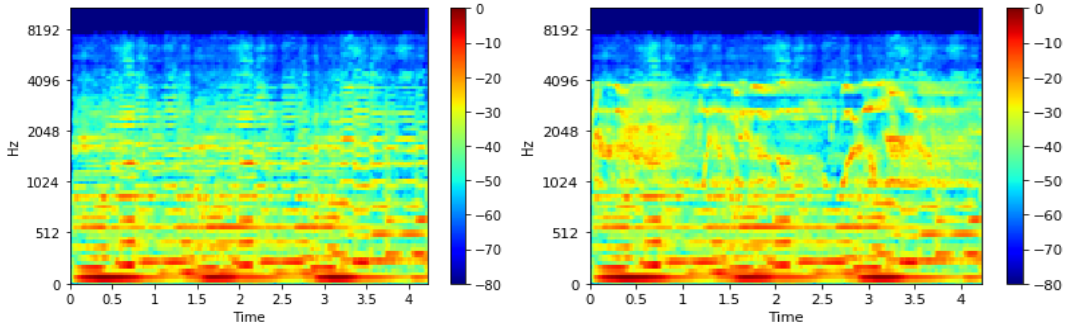

Figure 11: MEL spectrogram comparison of original and adversarial audio

Statistically, our generated examples are efficient enough to deceive the ASR system by using the evaluation method, Word Error Rate (WER) which is explained in Section 4.2.1.

4.2.1 Word Error Rate (WER)

To measure the performance of an ASR system that is used in voice-based applications, WER is an important method in testing. Also, it provides common metric-based results for the given audio. We can assume that the ASR system is powerful and efficient when WER is smaller, while it is less efficient with higher WER.

WER is evaluated using the method called “Levenshtein Distance”. Levenshtein distance can be calculated by comparing two strings. For example, the first string is “meet” and the second one is “neet”. The Levenshtein distance is 1 as there is a difference of only a single alphabet in these two strings. WER is the ratio of the total number of letters required to insert, delete, or replace in the predicted string to match with the original string with respect to the total number of letters in the original string. Suppose there are a total of 40 words in a string and the predicted string needs 9 additions, substitutions, and deletions to match with the original string. The WER would be 9 divides by 40 which is equal to 0.225. That is 22 percent of the WER. The Character Error Rate (CER) is the same as WER but only considers the number of differing characters. For the above example, the CER is 9 percent.

In this project, we calculated WER and CER for the original example and the adversarial example. So that we can measure how our adversarial examples are effective against the ASR system. Table 2 shows the mean value of WER and CER for the 905 original audio files calculated by the ASR system.

Table 2: Comparison table of WER for original and adversarial examples

	Word Error Rate (WER)	Character Error Rate (CER)
Original Examples	0.15	0.04
Adversarial Examples	0.92	0.74

Table 2 suggests that a WER rate of 0.15 means 15 percent occurred while transcribing original audios with the help of the ASR system and the WER rate of 0.92

means 92 percent error occurred for adversarial examples that we have generated. This shows that our generated adversarial audios are effective to deceive the ASR system.

4.3 Experimental Evaluation for CNN

Since CNN is a classification method, we can distinguish the equality between the original class and predicted class using the mean value of accuracy using the scikit-learn library of python. Figure 12 shows the result generated using this library. The report shows the accuracy, precision, recall, and f1-score count for each class. Here, class '0' is the original example while '1' suggests adversarial examples.

```
In [151]: print(metrics.classification_report(y_test_NN, pred))
```

	precision	recall	f1-score	support
0	0.95	0.99	0.97	273
1	0.98	0.95	0.97	270
accuracy			0.97	543
macro avg	0.97	0.97	0.97	543
weighted avg	0.97	0.97	0.97	543

Figure 12: Classification report for CNN

Figure 12 shows the average accuracy of 97% achieved by the CNN model which is the best because when the model has already been trained on adversarial examples, it will be easy for the model to derive a pattern of adversarial examples. Table 3 shows the different models we used along with the parameter tuning on their results.

Table 3: Different CNN models

CNN			
Optimization Method	**Neurons**	**Kernel Size**	**Accuracy**
Adam	32	(3,3)	0.97
	64	(3,3)	
	128	(3,3)	
	256	(3,3)	
Adam	32	(2,2)	0.95
	32	(2,2)	
	64	(2,2)	
	64	(2,2)	
Adagrad	32	(3,3)	0.94
	64	(3,3)	
	128	(3,3)	
	256	(3,3)	

Figure 13 is the graph that shows the Receiver Operating Characteristics (ROC) curve for the CNN models which are described in Table 4. ROC curve is a performance measurement for classification problems at various threshold settings [21]. This curve will show the probability for each classification where the x-axis and y-axis represent the False positive rate and True positive rate, respectively. If the area under the curve is higher, then the model is classifying the true class. This curve is plotted to differentiate between the true-positive rate and false-positive rate.

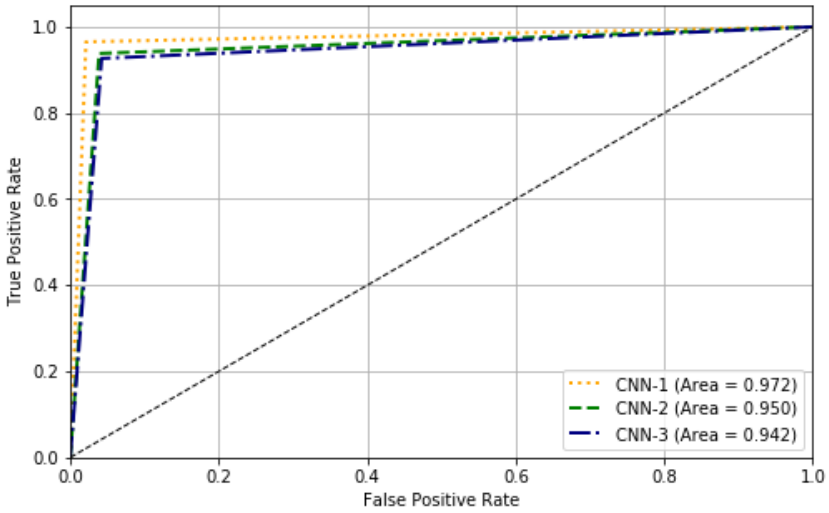

Figure 13: ROC curve for CNN

4.4 Results Achieved with One-Class SVM

To see the statistical result, a classification report is used which is shown in Figure 14. A classification report compares the true and predicted label for each sample and gives the mean value of different evaluations like Precision, Recall, F1-score, and Accuracy. 71% of accuracy was achieved on the test data. In the classification report, class -1 is an outlier or adversarial example. This means that they are different from the normal examples and class 1 is normal or original examples.

```
In [84]: print(metrics.classification_report(targs_1, preds_1))
```

	precision	recall	f1-score	support
-1	0.73	0.79	0.76	299
1	0.66	0.59	0.62	210
accuracy			0.71	509
macro avg	0.70	0.69	0.69	509
weighted avg	0.70	0.71	0.70	509

Figure 14: Classification report for One-Class SVM

To see the results visually, we plotted the graph using the matplotlib library of Python. Figure 15 gives the visualization of the comparison between predicted and actual data. In Figure 15, the first graph has a visual representation of the data which is predicted using the test data. The red dots are adversarial data and the blue dots are normal or original data while the second graph shows the actual test data. Comparing both the graphs, it can be seen that One-Class SVM is correctly identifying some of the adversarial examples, but it is misclassifying some original examples into adversarial examples and vice-versa. The reason behind this is that the model is not trained on examples that can have perturbation or noise which can differentiate them from the original examples. Moreover, this can happen because the model is not able to classify adversarial examples with the small perturbation and some of the original examples have more background noise in the raw audio which is misclassifying the actual classification. This problem or misclassification error is resolved by the GAN method explained in Section 3.6.2.2.

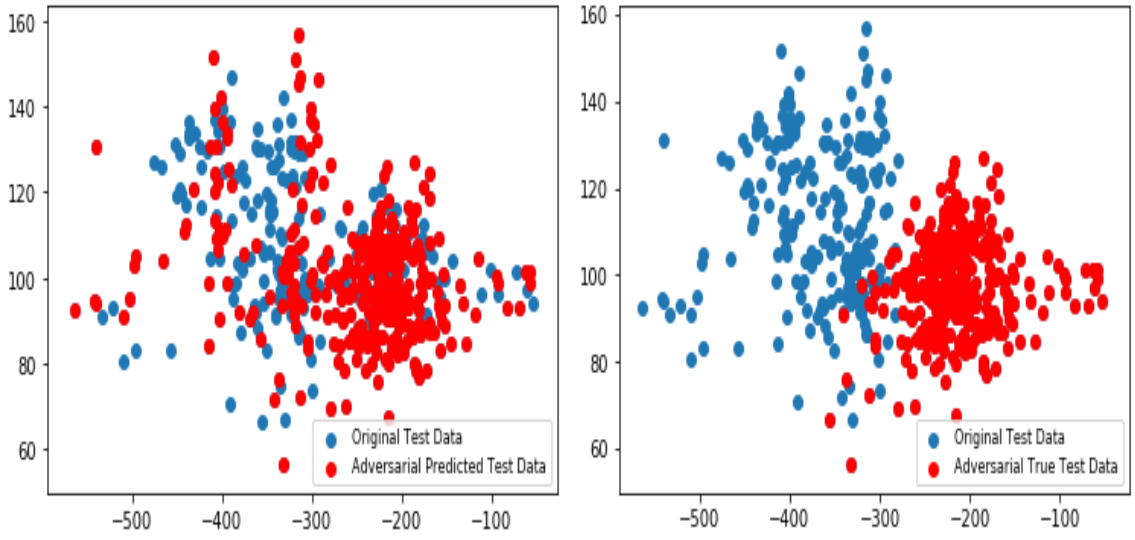

Figure 15: Comparison between actual and predicted data

4.5 GAN Evaluation

We used two GANs for evaluation purposes. Our results show that, the GAN introduced by Zenati et al. [12] achieves 80% accuracy in detecting adversarial samples while our proposed GAN achieves a 92% accuracy score for the test data. Table 4 and 5 shows the summary of the model for the discriminator and generator for the proposed GAN. Model parameters like neurons, different layers like dense, dropout, etc., activation functions, optimizers, and loss functions are shown in the tables.

Table 4: Generator model summary

GENERATOR					
Batch Size = 100, Latent Dimension $z = 32$, $\alpha = 10^{-6}$					
Optimizer	Number of Neurons	Layer	Activation Function	Loss Function	Dropout
Adam	16	Dense	ReLU	Sigmoid Cross-Entropy	0.5
	32	Dense	ReLU		
	64	Dense	ReLU		
	128	Dense	ReLU		
	193	Flatten Dense	Linear		

Table 5: Discriminator model summary

DISCRIMINATOR					
Batch Size = 100, Latent Dimension $z = 32$, $\alpha = 10^{-6}$					
Optimizer	Number of Neurons	Layer	Activation Function	Loss Function	Dropout
Adam	512	Dense	Leaky ReLU	Sigmoid Cross-Entropy	0.5
	256	Dense	Leaky ReLU		
	128	Dense	Leaky ReLU		
	64	Dense	Leaky ReLU		
	64	Dense	Leaky ReLU		
	32	Dense	Leaky ReLU		
	1	Flatten Dense	Sigmoid		

Since the model is trained on original data to optimize the generator, discriminator, and encoder loss, we defined a score function that will calculate the loss value for each sample. Then, we can decide whether that sample is normal or adversarial while testing. The score function is defined by equation 8:

$$Total\ Loss\ (x) = (1 - \alpha)\left(L_D(x)\right) + \alpha\ MAX(L_G(x), L_C(x)) \qquad (8)$$

In equation 8, $L_D(x)$ is discriminator loss, $L_G(x)$ is generator loss, $L_C(x)$ is cycle process loss and α is a learning parameter that updates with the iteration of each epoch according to the need of the model. The discriminator loss is derived in two ways, which are real discriminator loss and fake discriminator loss. Whenever a discriminator model is 100% sure about the data which can be fit in the real data distribution, the discriminator generates real discriminator loss. On the other hand, when the model ensures the data can be fit in randomly generated noise data distribution, it will generate fake discriminator loss. After calculating the testing score for each sample, if the score is greater than the threshold, then that sample is counted as an adversarial sample. After getting a score for each sample, we compared it with the true label for that sample which is showed using the confusion matrix. The confusion matrix is a representation of true-positive, true-negative, false-positive, and false-negative rates. The confusion matrix in Figure 16 interprets that the GAN model has successfully predicted the true-positive for the original samples and true-negative for the adversarial samples. However, the false-positive and false-negative rates are 8% on average for the whole test data.

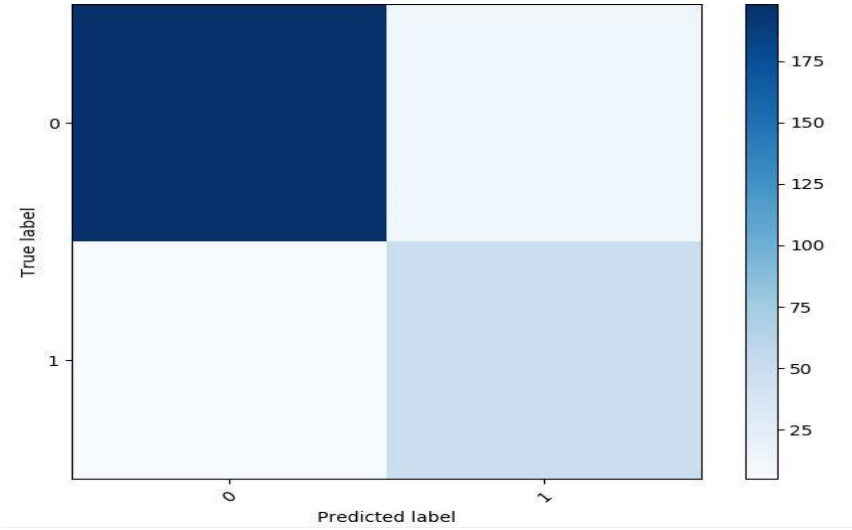

Figure 16: Confusion matrix for GAN

Figure 17 visualizes the change in loss values for the discriminator and generator while training. From this Figure, it can be seen that during each epoch in the training part, the generator will try to minimize the loss while the discriminator will try to maximize the loss value. Using the loss value, the score function will generate a score for each sample and that score will be used to approximate the threshold value to identify adversarial examples. Figure 17 shows the loss value for the 100 epochs.

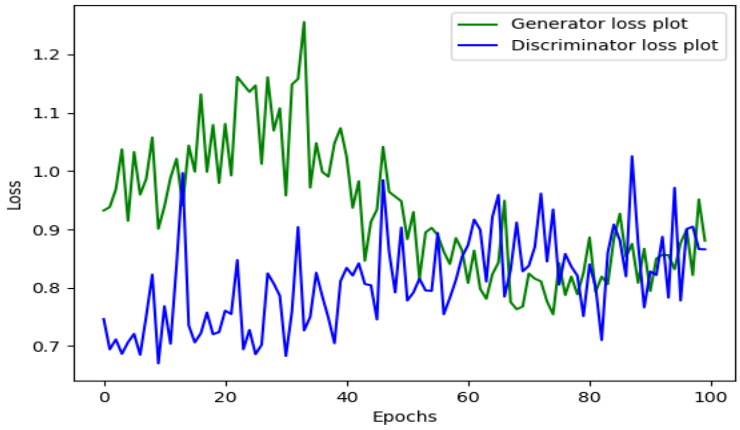

Figure 17: Generator and Discriminator loss

Defining a threshold value which will separate original and adversarial examples is an important factor in detecting adversarial examples. The generated score values from the loss are distributed among particular ranges for each sample which helps us to identify the difference between the original and the adversarial examples. If the score value for a particular sample is greater than the threshold value, then that sample will be identified as an adversarial sample otherwise it is an original sample. For this GAN model, the best threshold value is 80%. This means that among all distributed score values if a particular sample has more than 80% of the score value then that sample is identified as an adversarial sample. Figure 18 shows the accuracy value for different threshold values like 70%, 80%, and 90%.

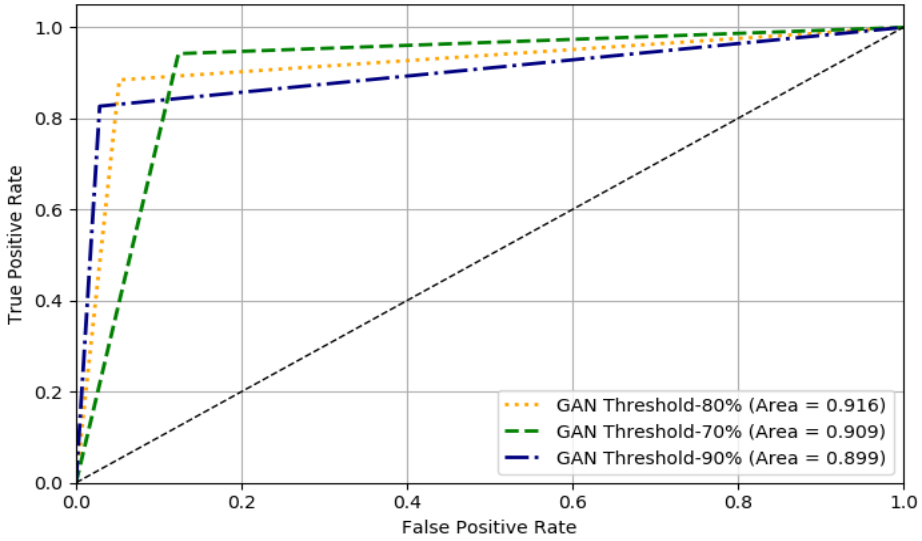

Figure 18: Different threshold value's ROC curve

The impact of epochs in the training and testing is another important factor. The number of epochs is related to the optimization process which happens at the time of training. If the epochs number is small, then that may lead the model to misclassify.

Increasing the number of epochs leads to error reduction when the model is being trained. At some point, increasing the number of epochs leads the model to over-fit the train data and the model will begin to lose performance to classify the data which are not trained (test data). To analyze a certain point where increasing the number of epochs leads models to misclassify, we generated graphs for the different epochs with regards to their accuracy and F1-score.

As shown in Figure 19, if we train data for just 10 epochs then the result was not good in terms of accuracy as well as F1-score. The model is performing best for 50 epochs in comparison to 100, 500, and 1000 epochs. If the model is trained for more than 100 epochs then the accuracy and F1-score is decreasing for the 500 epochs and there is no improvement in accuracy score for 1000 epochs which means the model is not learning anything from training data and the model starts to over-fit if we increase the number of epochs.

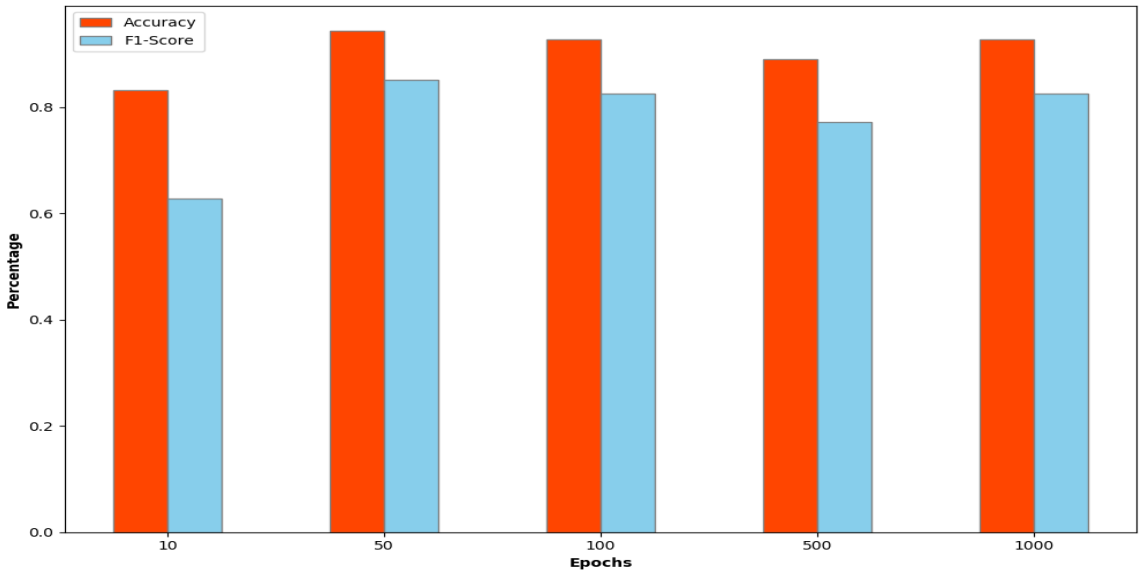

Figure 19: Impact of epochs in the optimization of model

This is the implementation that we did as a defense strategy. Results for each strategy (i.e., Convolutional Neural Network, One-Class SVM, and GAN) are shown in Table 6.

Table 6: Result of all implemented defense strategies

Defense Strategy	Accuracy
Convolutional Neural Network (CNN)	97%
One-Class SVM	71%
GAN introduced by Zenati et al. [12]	80%
Proposed GAN	92%

CHAPTER 5

PROJECT CONCLUSION AND FUTURE WORK

5.1 Conclusion

This project has described challenges associated with the adversarial attack in speech recognition systems. To understand the attack procedure, we generated adversarial examples that are robust to the room environment or to any noise which can target state-of-art Automatic Speech Recognition models. This can be used to attack the recognition model in the real world. To generate this type of example, we used techniques named band-pass filter, impulse responses, and White Gaussian noise. The results are evaluated using the Word Error Rate (WER) which is 15% for original examples and 92% for generated adversarial examples. Thus, it can be concluded that the generated example has a small amount of perturbation, which can attack in the real-world because it will not be able to deal with RNN models in the ASR system.

This project also describes defense strategies by detecting adversarial samples to deal with adversarial attacks. Two strategies are used to detect such examples. First is using Convolutional Neural Network (CNN) which has been trained on adversarial examples. The second is using models like One-Class SVM and Generative Adversarial Network (GAN) which have not been trained on adversarial examples previously. The achieved results using CNN is 97% and using GAN is 92% accuracy which out-performs the result of 80% achieved using the method introduced by Zenati et al. [12] on the LibriSpeech dataset.

5.2 Future Work

In the future, one can make extensive evaluation methods for the GAN. Moreover, one can introduce training strategies other than optimizing loss function and improve score function using that strategy. Also, the effects of encoder-decoder prediction accuracy can be analyzed on detecting adversarial samples.

Bibliography

1. R. Rafal. “Distill the Automatic Speech Recognition (TensorFlow),” [Online]. Available: http://github.com/rolczynski/Automatic-Speech-Recognition [Accessed: October 2020].
2. J. Shen, P. Nguyen, Y. Wu, Z. Chen, M. X. Chen, Y. Jia, ... and Y. He. “Lingvo: a modular and scalable framework for sequence-to-sequence modeling,” [Online]. Available: https://arxiv.org/pdf/1902.08295.pdf [Accessed: October 2020].
3. M. Alzantot, B. Balaji, and M. Srivastava. “Did you hear that? adversarial examples against automatic speech recognition,” [Online]. Available: https://arxiv.org/pdf/1801.00554.pdf [Accessed: October 2020].
4. I. M. Bennett, B. R. Babu, K. Morkhandikar, and P. Gururaj. “Distributed realtime speech recognition system,” U.S. Patent No. 6,633,846. Washington, DC: U.S. Patent and Trademark Office, October 2003.
5. K. H. Cohen. “Patient monitoring system including speech recognition capability,” U.S. Patent No. 6,014,626. Washington, DC: U.S. Patent and Trademark Office, January 2000.
6. N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” *in Proceedings of the IEEE Symposium on Security and Privacy*, pp. 39–57, May 2017.
7. H. Yakura and J. Sakuma, “Robust audio adversarial example for a physical attack,” *in Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 5334-5341, AAAI Press, August 2019.

8. H. Kwon, H. Yoon, and K. W. Park, "POSTER: Detecting Audio Adversarial Example through Audio Modification," *in Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pp. 2521-2523, November 2019.
9. R. Vlasveld. "Introduction to One-class Support Vector Machines," [Online]. Available: http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/ [Accessed: October 2020].
10. J. Rocca. "Understanding Generative Adversarial Networks (GANs)," [Online]. Available: https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29 [Accessed: October 2020].
11. D. Povey. "LibriSpeech ASR corpus," [Online]. Available: http://www.openslr.org/12/ [Accessed: October 2020].
12. H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar. "Efficient gan-based anomaly detection," [Online]. Available: https://arxiv.org/pdf/1802.06222.pdf [Accessed: October 2020].
13. Y. Qin, N. Carlini, G. Cottrell, I. Goodfellow, and C. Raffel, "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition," *in Proceedings of the International Conference on Machine Learning*, pp. 5231-5240, PMLR, May 2019.

14. S. Solanki, G. Krishnan, V. Sampath, and J. Polakis, “In (cyber) space bots can hear you speak: Breaking audio CAPTCHAs using OTS speech recognition,” *in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 69-80, November 2017.
15. M. Alzantot, Y. Sharma, A. Elgohary, B. J. Ho, M. Srivastava, and K. W. Chang, “Generating Natural Language Adversarial Examples,” *in Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 2890-2896, November 2018.
16. Wikipedia. “Mel-frequency cepstrum,” [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mel-frequency_cepstrum&oldid=987268448 [Accessed: November 2020].
17. D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, ... and J. Chen, “Deep speech 2: End-to-end speech recognition in english and mandarin,” *in Proceedings of the International conference on machine learning*, pp. 173-182, June 2016.
18. Wikipedia. “Band-pass filter,” [Online]. Available: https://en.wikipedia.org/w/index.php?title=Band-pass_filter&oldid=986133931 [Accessed: November 2020].
19. B. Phil. “What Is Impulse Response In Audio?,” [Online]. Available: https://www.soundassured.com/blogs/blog/what-is-impulse-response-in-audio [Accessed: October 2020].

20. S. Saha. "A Comprehensive Guide to Convolutional Neural Networks – The ELI5 way," [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [Accessed: October 2020].
21. S. Narkhede. "Understanding AUC - ROC Curve," [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 [Accessed: October 2020].
22. J. Donahue, P. Krähenbühl, and T. Darrell. "Adversarial feature learning," [Online]. Available: https://arxiv.org/pdf/1605.09782.pdf [Accessed: October 2020].
23. M. Patel, X. Wang, and S. Mao. "Data augmentation with Conditional GAN for automatic modulation classification," *ACM Workshop on Wireless Security and Machine Learning (WiseML 2020), in conjunction with the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (ACM WiSec 2020)*, Linz, Austria, July 2020.
24. X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *in Proceedings of the IEEE Transactions on Vehicular Technology*, vol.66, no.1, pp.763-776, January 2017.
25. X. Wang, X. Wang, and S. Mao, "Deep convolutional neural networks for indoor localization with CSI images," *in Proceedings of the IEEE Transactions on Network Science and Engineering, Special Issue on Network Science for Internet of Things (IoT)*, vol.7, no.1, pp.316-327, January 2020.

26. X. Wang, C. Yang, and S. Mao. “On CSI-based vital sign monitoring using commodity WiFi,” *ACM Transactions on Computing for Healthcare, vol.1, no.3*, pp.12:1-12:27, April 2020.

27. J. Purohit, X. Wang, S. Mao, X. Sun, and C. Yang, “Fingerprinting-based indoor and outdoor localization with LoRa and deep learning,” *in Proceedings of the IEEE GLOBECOM*, Taipei, Taiwan, December 2020.