|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  **SCHOOL OF ELECTRONICS AND TELECOMMUNICATIONS**  logo_128  **DESIGN PROJECT III REPORT**  ***Abnormal Sound Event Detection***  Instructor: Prof. Han Huy Dung   |  |  |  | | --- | --- | --- | | Student | ID | Class | | Nguyen Dinh Quoc | 20153060 | Electronics 06 – K60 | | Nguyen Minh Hieu | 20151336 | Electronics 03 – K60 | | Nguyen Duy Quang | 20152956 | Electronics 06 – K60 |   Hanoi, 1/2020 |

**PREFACE**

In audiobook related work, the elimination of abnormal sounds is extremely important. Strange sounds that are not reading sounds can be breath sounds, keypress sounds, cough sounds, book page turn sounds, etc.The task is to build an artificial intelligence system that has ability to detect these strange noises, thereby removing them to create better quality audiobooks. In the design project III course, we have built an abnormal sound detection system using different methods, from which to find the most optimal method. We would like to express our sincere thanks to Prof. Han Huy Dung for dedicated helping us in doing this project.

**TABLE OF CONTENTS**

[TABLE OF FIGURES ii](#_Toc30529911)

[TABLE OF TABLES iii](#_Toc30529912)

[ABTRACT iv](#_Toc30529913)

[CHAPTER 1. INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND ABNORMAL SOUND DETECTION 1](#_Toc30529914)

[1.1 Artificial intelligence, machine learning and deep learning 1](#_Toc30529915)

[1.1.1 Overview 1](#_Toc30529916)

[1.1.2 Artificial intelligence 2](#_Toc30529917)

[1.1.3 Machine learning 2](#_Toc30529918)

[1.1.4 Deep learning 4](#_Toc30529919)

[1.2 Abnormal sound detection problem 5](#_Toc30529920)

[CHAPTER 2. THE SYSTEM MODEL 6](#_Toc30529921)

[2.1 Overview 6](#_Toc30529922)

[2.2 Feature extraction 7](#_Toc30529923)

[2.2.1 Overview 7](#_Toc30529924)

[2.2.2 Short-time Fourier transform 7](#_Toc30529925)

[2.2.3 Mel filter bank 8](#_Toc30529926)

[2.3 Artificial neural network 11](#_Toc30529927)

[2.3.1 Overview 11](#_Toc30529928)

[2.3.2 Convolutional recurrent neural network 12](#_Toc30529929)

[2.3.3 Autoencoder 18](#_Toc30529930)

[2.3.4 Generative adversarial network 20](#_Toc30529931)

[CHAPTER 3. EXPERIMENTS AND RESULTS 23](#_Toc30529932)

[3.1 Datasets 23](#_Toc30529933)

[3.1.1 Basic-shaped signals 23](#_Toc30529934)

[3.1.2 Audiobook 25](#_Toc30529935)

[3.1.3 Audio recorded in SPARC laboratory 26](#_Toc30529936)

[3.2 Metric 27](#_Toc30529937)

[3.3 Experiments and results 29](#_Toc30529938)

[3.3.1 Experiment using basic-shaped signals dataset 29](#_Toc30529939)

[3.3.2 Experiment using audiobook dataset 35](#_Toc30529940)

[3.3.3 Experiment using audio recorded in SPARC laboratory dataset 38](#_Toc30529941)

[CONCLUSION 40](#_Toc30529942)

[REFERENCES 41](#_Toc30529943)

[APPENDIX 42](#_Toc30529944)

[Appendix 1. Short-time Fourier transform 42](#_Toc30529945)

[Appendix 2. Source code 45](#_Toc30529946)

# TABLE OF FIGURES

[Figure 1.1 The difference between AI, ML and DL 1](#_Toc30530234)

[Figure 1.2 Performance measure in ML 3](#_Toc30530235)

[Figure 1.3 An example of sound event detection system 5](#_Toc30530236)

[Figure 2.1 The system model 6](#_Toc30530237)

[Figure 2.2 Feature extraction block 7](#_Toc30530238)

[Figure 2.3 Mel filter banks with NFFT = 2048 and nframe = 10 9](#_Toc30530239)

[Figure 2.4 Mel filter banks is normalized 9](#_Toc30530240)

[Figure 2.5 Mel band energy process 10](#_Toc30530241)

[Figure 2.6 Mel band energy 10](#_Toc30530242)

[Figure 2.7 An artificial neuron 11](#_Toc30530243)

[Figure 2.8 General ANN architectures 12](#_Toc30530244)

[Figure 2.9 The FNN topology 12](#_Toc30530245)

[Figure 2.10 Typical CNN architecture 13](#_Toc30530246)

[Figure 2.11 Moving 2x2 filter 14](#_Toc30530247)

[Figure 2.12 Max pooling example 15](#_Toc30530248)

[Figure 2.13 RNN diagram with nodes shown 16](#_Toc30530249)

[Figure 2.14 Unrolled RNN 16](#_Toc30530250)

[Figure 2.15 Two major shortcomings of FNNs and solutions 17](#_Toc30530251)

[Figure 2.16 CRNN architecture 18](#_Toc30530252)

[Figure 2.17 A basic autoencoder 19](#_Toc30530253)

[Figure 2.18 An example of GAN 20](#_Toc30530254)

[Figure 2.19 The architecture of DCGAN generator 21](#_Toc30530255)

[Figure 3.1 Basic-shaped signals 23](#_Toc30530256)

[Figure 3.2 Labeling the signals 24](#_Toc30530257)

[Figure 3.3 Setup diagram for audio recoder in SPARC laboratory 26](#_Toc30530258)

[Figure 3.4 Actual images of hardware setup for the recording system 26](#_Toc30530259)

[Figure 3.5 Possible cases when predicting an event 27](#_Toc30530260)

[Figure 3.6 Spectrogram of basic-shaped signals 29](#_Toc30530261)

[Figure 3.7 Mel-spectrogram of basic-shaped signals 30](#_Toc30530262)

[Figure 3.8 the output of CRNN with the inputs are basic-shaped input signals. 32](#_Toc30530263)

[Figure 3.9 Model of the generator and the discriminator 33](#_Toc30530264)

[Figure 3.10 Discriminator and generator loss in training phase 33](#_Toc30530265)

[Figure 3.11 Details of CRNN model 36](#_Toc30530266)

[Figure 3.12 output value of CRNN model and the true label 37](#_Toc30530267)

[Figure 3.13 Details of autoencoder model 38](#_Toc30530268)

[Figure 3.14 Autoencoder implementation result 39](#_Toc30530269)

# TABLE OF TABLES

[Table 1.1 Some common problem in ML 3](#_Toc30530371)

[Table 3.1 Class of basic shape input 24](#_Toc30530372)

[Table 3.2 Files used in the audiobook dataset 25](#_Toc30530373)

[Table 3.3 Categories in the audiobook dataset 26](#_Toc30530374)

[Table 3.4 Output file format 31](#_Toc30530375)

[Table 3.5 Ouputs of the discriminator 34](#_Toc30530376)

[Table 3.6 Confusion matrix 37](#_Toc30530377)

[Table 3.7 Score with threshold MSE = 2e-2 39](#_Toc30530378)

# ABTRACT

This report presents the strange sound detection system, built by three different methods: Convolutional Recurrent Neural Network (CRNN), Autoencoder and Generative Adversarial Network (GAN). The dataset used in thiss project includes the sound recorded in SPARC laboratory and audiobooks. The implementation results of these three methods will be compared with each other, thereby identifying the advantages and disadvantages of each method.

# INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND ABNORMAL SOUND DETECTION

## Artificial intelligence, machine learning and deep learning

### Overview

Nowadays, in our modern life, we must have heard about artificial intelligence (AI). AI is already part of our everyday lives with so many applications in most fields, including face recognition, recommendation systems, computer games, disease diagnosis, etc. Let’s take a specific example, when Google DeepMind’s AlphaGo program defeated South Korean Master Lee Se-dol in the board game Go in 2016, the terms AI, machine learning (ML), and deep learning (DL) were used in the media to describe how DeepMind won. We have already known about AI, but what are ML and DL, and are they the same things? The answer is, AI is the idea that came first, then machine learning - which blossomed later, and finally deep learning - which is driving today’s AI explosion. Figure 1.1 shows the difference between AI, ML and DL [1].

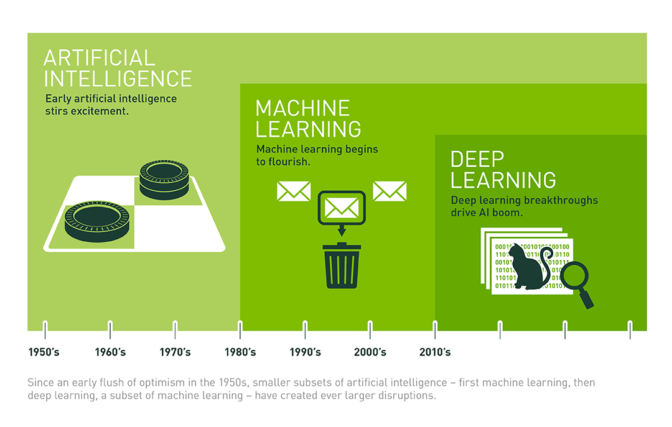


Figure 1.1 The difference between AI, ML and DL

From Figure 1.1, we can recognize that DL is a subset of ML and ML is a subset of AI. Now we will discuss each term in detail.

### Artificial intelligence

In summer 1956, at the Dartmouth Conferences held in Hanover, New Hampshire, USA, AI pioneers had a dream to construct complex machines that possessed the same characteristics of human intelligence. These machines have all our senses) and think just like we do. In the decades since, AI has alternately been heralded as the key to our civilization’s brightest future. AI has been growing over years, and especially since 2015, AI has actually exploded with the wide availability of GPUs that make parallel processing ever faster, cheaper, and more powerful.

AI technologies, such as image classification on Pinterest and face recognition on Facebook, exhibit some facets of human intelligence. But how does AI work, the next sectio, machine learning, will answer that.

### Machine learning

Machine learning (ML) is an approach to achieve AI. As defined by Arthur Lee Samuel in 1959, ML is the subfiled of computer science, that “gives computers the ability to learn without being explicitly programmed” [2]. We can simply understand that ML at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction, instead of hand-coding software routines with a specific set of instructions to accomplish a particular task.

There is one more concept we need to mention, that is computer program’s learning. According Tom M. Mitchell’s definition in 1997, a computer program is said to learn from experience E with respect to some tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [3]. Next we will explain each of these term.

* Task, T

ML tasks are often described as how a system handles a data point. Table 1.1 list some common problem in ML.

Table 1.1 Some common problem in ML

|  |  |  |
| --- | --- | --- |
| **Problem** | **Content** | **Example** |
| Classification | Indicate the label of a data point. Labels are divided into groups | Handwritten digit recognition |
| Regression | Problem where labels are real numbers instead | Gender and age prediction based on face (Microsoft app) |
| Machine translation | Translate a language into another language | Google Translate |
| Clustering | Divide data into groups based on each group’s data correlation | Devide customers into group based on buying behavior |
| Completion | Complete a data point with missing values | Recommendation system |
| Some other problems: ranking, information retrieval, denoising, etc | | |

* Performance measure, P

P is used to test the performance of a ML algorithm. Figure 1.2 describes how to evaluate a model performance

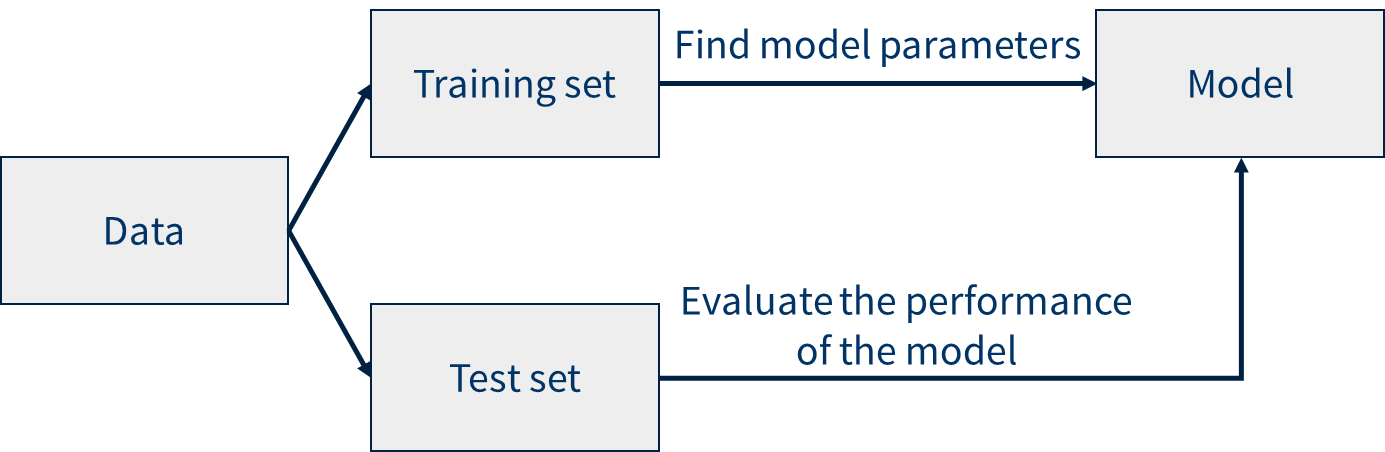


Figure 1.2 Performance measure in ML

The dataset is divided into two sets, namely training set and test set. The training set is used to find model parameters in order to build model, and the performance of the model will be evaluated by test set.

* Experience, E

Training ML models can be considered giving them experience on datasets (training sets). Based on properties of datasets, ML algorithms are divided into 2 groups: Supervised learning and Unsupervised learning

* Supervised learning: Predict the output of one or more new data points based on known pairs (input, output). Our work is to find a function mapping each element from the input set to a corresponding (approximate) element of the output set.
* Unsupervised learning: Only feature vectors of the input data are known, so the system has to rely on the data structure to perform a task. In this algorithm, we can not know the exact output of the input

### Deep learning

Deep learning (DL) is a technique for implementing ML. In the beginning, another algorithmic approach from the early ML crowd, artificial neural networks (ANN), came and mostly went over the decades. ANNs are inspired by our understanding of the biology of our brains – all those interconnections between the neurons.

In 2012, Andrew Ng, a computer scientist and statistician working at Google, had a breakthrough in AI technology. He took ANNs, and essentially make them huge, increase the layers and the neurons, and then run massive amounts of data through the system to train it. In his case it was images from 10 million YouTube videos. Then he put the “deep” in deep learning, which describes all the layers in these ANNs. From here the term “deep learning” was born. Back to the example at the beginning of section 1.1.1, Google’s AlphaGo learned the game, and trained for its Go match, it tuned its neural network by playing against itself over and over and over.

DL has enabled many practical applications of ML and by extension the overall field of AI. Driverless cars, better preventive healthcare, even better movie recommendations, are some significant application of deep learning in our life. AI is the present and the future. With Deep learning’s help, AI may even get to that science fiction state we’ve so long imagined.

## Abnormal sound detection problem

In our daily lives, we encounter a rich variety of sound events such as dog bark, footsteps, glass smash and thunder. Sound event detection (SED) is responsible for detecting the onset and offset times for each sound event in an audio recording [3]. With emergence of more advanced deep learning techniques, SED has attracted more interest in recent years with many applications in practice. For example, recognizing environmental sounds will give an idea about the local biodiversity (figure 1.3). Detecting sound events such as glass breaking and alarm detection can be used for surveillance. Moreover, SED can be applied in healthcare moitoring, urban sound analysis, multimedia event detection and bird call detection.

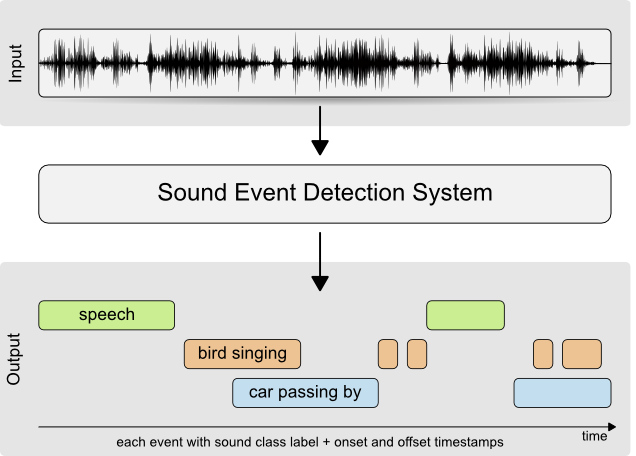


Figure 1.3 An example of sound event detection system

The abnormal sounds detection problem is a subset of SED. The term “abnormal sounds” refers to sounds that are not desired sounds. In audiobook, sounds that are not the reading voice will be considered as abnormal sounds. The purpose of the abnormal sounds detection system is to detect all the strange sounds happening at a time. Unlike real-life environment, where multiple sound events are very likely to overlap in time, abnormal sounds in audiobook rarely occur simultaneously at a time, this makes the sound events detection easier. In our project, we use two datasets in experiment, the first one is the sound recorded in our SPARC laboratory and the second one is audiobooks. We can see the difference between two datasets latter.

# THE SYSTEM MODEL

## Overview

Figure 2.1 presents an overview of our system model which will be used for acoustic sound analysis.

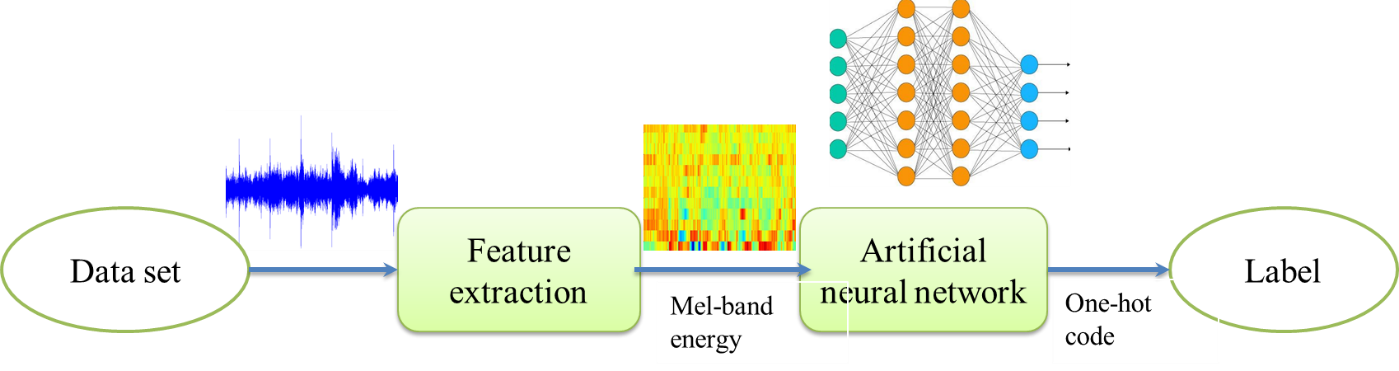


Figure 2.1 The system model

In this model, the dataset is audio recorded in our laboratory or audiobooks. The audio taken from the dataset is then put into the feature extraction block. The feature extraction block then extracts mel-band energy, as shown in the figure. After that, feature will be pushed into ANN to find out the function mapping Mel-band energy to its respective label. The label is one-hot code corresponding to label encoded from dataset.

## Feature extraction

### Overview

Figure 2.2 gives an overview of the feature extraction block.

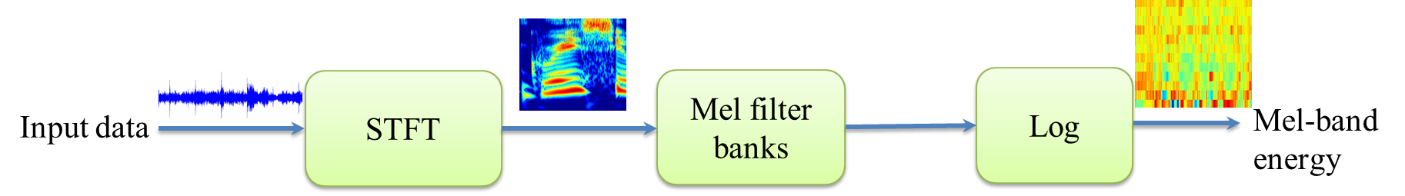


Figure 2.2 Feature extraction block

First, the input data is inserted into the STFT block to compute the short time Fourier transform (STFT). The STFT is used to extract the power spectrum of the input signal. Signals in time domain will give a sense of amplitude, but do not show frequency components. In constrast, signals in frequency domain gives frequency components but do not indicate order frequencies. STFT shows both frequency components and the order in which they occur. Next, the Mel filter banks block applies many mel-filters (a set of triangle filters used to mimic human ear) to spectrum of signal to obtain mel-frequency cepstrum (MFC). After that, the logarithm operation reduces the magnitude of the spectrum in MFC. Eventually, we have Mel-band energy.

### Short-time Fourier transform

This section only gives a rough overview of short-time Fourier transform (STFT), details of STFT are shown in Appendix. Below is the formula for discrite-time STFT.

where:

* x[n]: Signal is disjointed in time domain
* w: Type of window function
* m: Index of window function

### Mel filter bank

The content of this section is referenced from [6]. In many research articles on human sound reception, it was found that the human ear with each frequency of absorption is different. Therefore, to archive high effect in audio processing, we mimic human ear by filter band called Mel filter banks. In order to model Mel filter banks, we introduce a concept "Mel scale". Mel scale has effect of nonlinearity in the frequency domain of the signal, mimic the effect of human ear with each frequency:

where:

m : Mel scale

f : Frequency max divide by 2

In Mel scale domain (m), we linearize the signal, after that we use this result to compute frequency domain given by the following formula:



where:

m: Mel scale

f: Frequency max divide by 2

Mel filter banks includes 1 set of triangle filters, this can be expressed as:

where:

* : Function of triangle filter
* : Function converts from Mel scale domain to frequency domain

Figure 2.3 illustrates Mel filter banks with NFFT = 2048 and number of triangle filters is 10, each filter was presented by unique color. However, if we want to use these filters effectively, we need to normalize the triangle filters into the same area (energy), to avoid increasing value of noise in high-energy filter. To do this, we divide their height by the corresponding length (figure 2.4).

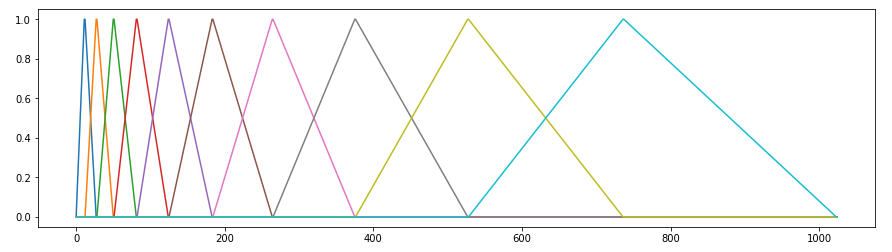


Figure 2.3 Mel filter banks with NFFT = 2048 and nframe = 10

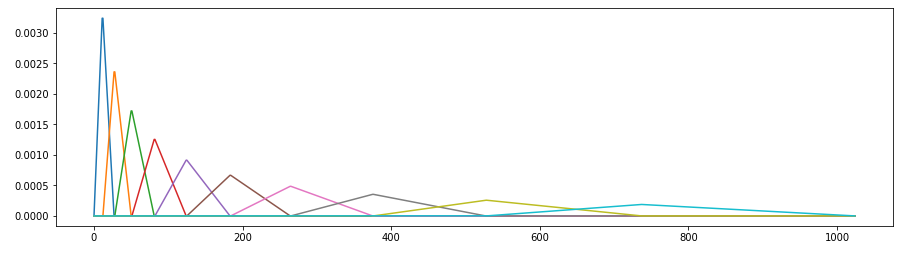


Figure 2.4 Mel filter banks is normalized

Aims to bring the sound obtained through the microphone to the visual sound of the human ear, we put the power spectrum signal through the mel filter banks. After that, we take logarithm base 10 to reduce magnitude of result. The result of this process is Mel band energy. Figure 2.5 illustrates method to achieve Mel band energy

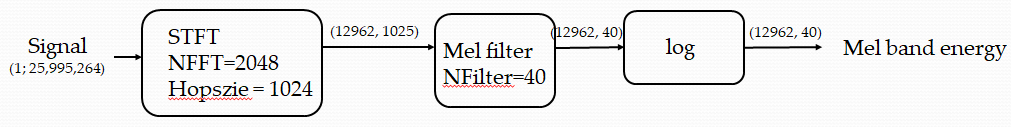


Figure 2.5 Mel band energy process

Figure 2.6 above illustrates Mel band energy 3.5s speech with NFFT = 512 and fmax = 8kHz.

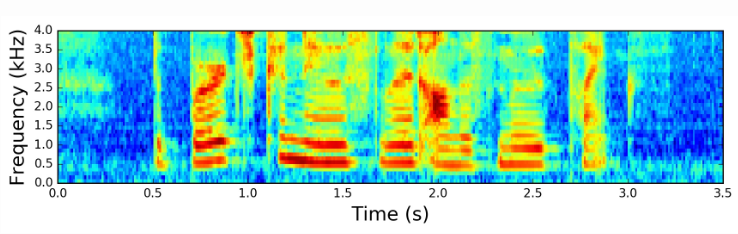


Figure 2.6 Mel band energy

Finally, the logarithm operation will be applied to mel-frequency cepstrum (MFC) to reduce the magnitude of the spectrum, and Mel-band energy is created.

## Artificial neural network

### Overview

The term “neural” is derived from the human (animal) nervous system’s basic functional unit “neuron” or nerve cells which are present in the brain and other parts of the human (animal) body. Biological neural network, in general, is a highly interconnected network of billions of neuron with trillion of interconnections between them. And Artificial neural networks (ANN) are the biologically inspired simulations performed on the computer to perform certain specific tasks like clustering, classification, pattern recognition, etc.

Neural networks resemble the human brain in two ways:

* A neural network acquires knowledge through learning
* A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights

ANN can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges with weights are connections between neuron outputs and neuron inputs (figure 2.7).

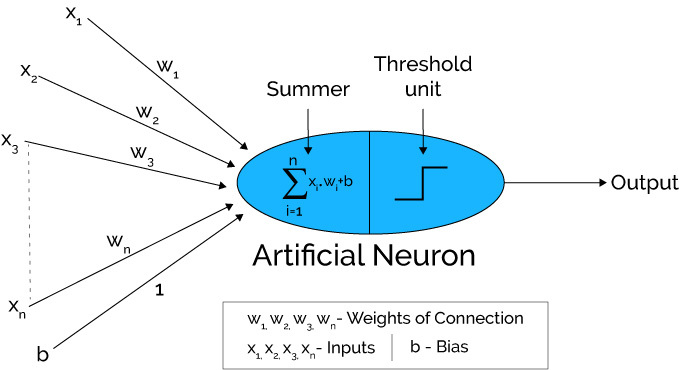


Figure 2.7 An artificial neuron

There are a number of types of artificial neural network. Figure 2.8 shows some general ANN architecures. Within the scope of this project, we only introduce three types: Convolutional recurrent neural network (CRNN), autoencoder and Generative adversarial network (GAN).

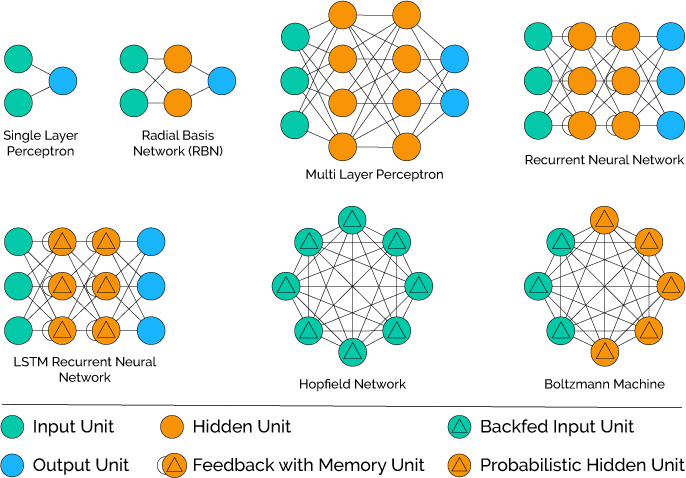


Figure 2.8 General ANN architectures

### Convolutional recurrent neural network

Before presenting about Convolutional recurrent neural network (CRNN), we will discuss about Feedforward neural network (FNN), Convolutional neural network (CNN) and Recurrent neural network (RNN), that leads to the appearance of CRNN.

#### Feedforward neural network

A feedforward neural network (FNN) is an artificial neural network wherein connections between the nodes do not form a cycle. The FNN was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes, as shown in figure 2.9.

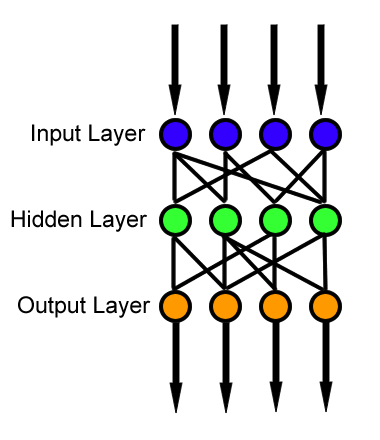


Figure 2.9 The FNN topology

In terms of network topology, FNN has two kinds: single-layer perceptron and multi-layer perceptron. The single-layer perceptron network is the simplest kind, consisting of a single layer of output nodes, the inputs are fed directly to the outputs via a series of weights. Meanwhile, the multi-layer perceptron network consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer [7]. Both two architecture kinds of FNN have been shown in Figure 2.8.

#### Convolutional neural network

The content of this section is referenced from [8]. Convolutional neural network (CNN) is feedforward network, where there are relations between weights. CNN composes of one or more convolutional layers with fully connected layers on top. It uses tied weights and pooling layers. In particular, max-pooling. Convolution is a specialized kind of linear operation. CNNs are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Figure 2.10 shows a typical CNN architecture.



Figure 2.10 Typical CNN architecture

The CNN architecture consists of multiple convolutional filters including rectified linear unit (ReLu) activations, followed by pooling / sub-sampling. Then we have another layer of convolution and pooling. The number of channels (the stacked blue squares) and the reduction in the x, y sizes of each channel as the sub-sampling / down-sampling occurs in the pooling layers. Finally, we reach a fully connected layer before the output. Now we are going to discuss about convolutional layer, ReLu function, pooling layer and fully connected layer.

* Convolutional layer

Moving filter is a concept most commonly associated with CNN. It passes through the image, applies to a certain neighbourhood of nodes, as shown in figure 2.11, where the filter applied is 0.5 x the node value.

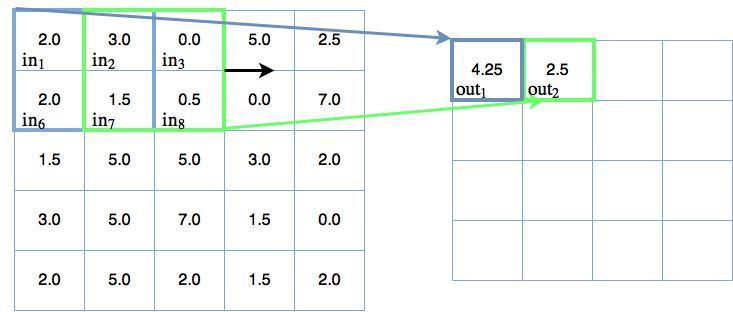


Figure 2.11 Moving 2x2 filter

The output of the convolutional mapping is then passed through some form of non-linear activation function, often the rectified linear unit (ReLu) activation function.

* Rectifier linear unit function

The rectifier is an activation function defined as:

f(x)=max(0,x)

where x is the input to a neuron.

A unit employing the rectifier is called a rectified linear unit (ReLU). Hence, this function is also called ReLu function

* Pooling layer

The idea of pooling in CNNs is to do two things:

* Reduce the number of parameters in your network (so that pooling is also called “down-sampling”)
* To make feature detection more robust by making it more impervious to scale and orientation changes

Pooling is a sliding window like in convolutional layer, but instead of applying weights, the pooling applies some sort of statistical function over the values within the window. Max() is the most common function used, so max pooling take the maximum value within the window. Figure 2.12 shows an example of max pooling with padding nodes. The filter is a 2x2 window and we need these nodes, so max pooling filter can make 3 steps in vertical or horizontal directions with a stride of 2, despite there being only 5 nodes to traverse in either the vertical or horizontal directions.

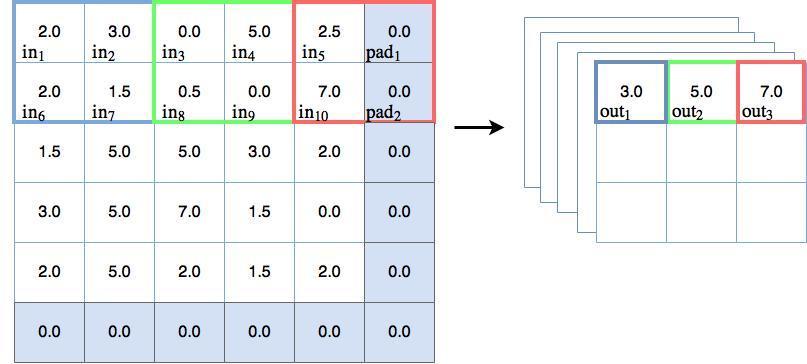


Figure 2.12 Max pooling example

* Fully connected layer

The purpose of fully connected layer is to make classifications regarding these objects. As shown in figure 2.6, the output of the final pooling layer is many channels of x x y matrices. To connect the output of the pooling layer to the fully connected layer, we need to flatten this output into a single (N x 1) tensor.

#### Recurrent neural network

The content of this section is referenced from [9]. A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. The key difference to FNNs is the introduction of time, in particular, the output of the hidden layer in a recurrent neural network is fed back into itself. Figure 2.13 illustrate a RNN diagram.

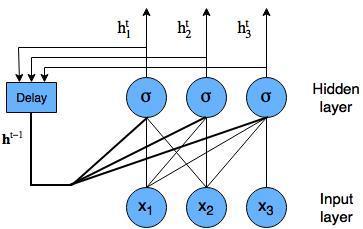


Figure 2.13 RNN diagram with nodes shown

We have a simple RNN with three input nodes. These nodes are fed into a hidden layer, with sigmoid activations, as per any normal densely connected neural network. The output of the hidden layer is passed through a conceptual delay block to allow the input of into the hidden layer, then fed back into the same hidden layer.

The type of flow of information through time (or sequence) in a recurrent neural network is shown in figure 2.14, which unrolls the sequence.

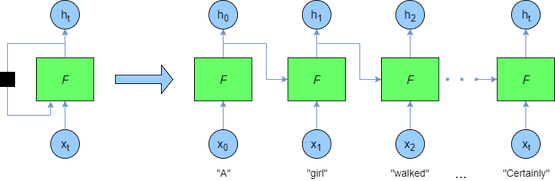


Figure 2.14 Unrolled RNN

On the left-hand side of figure 2.10, we have basically the same diagram as figure 2.9. We in fact only ever supply finite length sequences to the network, therefore we can unroll the network as shown on the right-hand side of the diagram above. This unrolled network shows how we can supply a stream of data to the recurrent neural network. First we supply the vector to the network F, the output of the nodes in F are fed into the next network and also act as a stand-alone output . The next network F at time t=1 takes the next vector and the previous output into its hidden nodes, producing the next output and so on.

#### The advent of CRNN

In sound events related work, FNNs, CNNs and RNNs are commonly used methods. FNNs have been used in monophonic and polyphonic SED in real-life environments by processing concatenated input frames from a small time window of the spectrogram. There are two major shortcomings presented by the FNN architecture, which are shown in Figure 2.15.

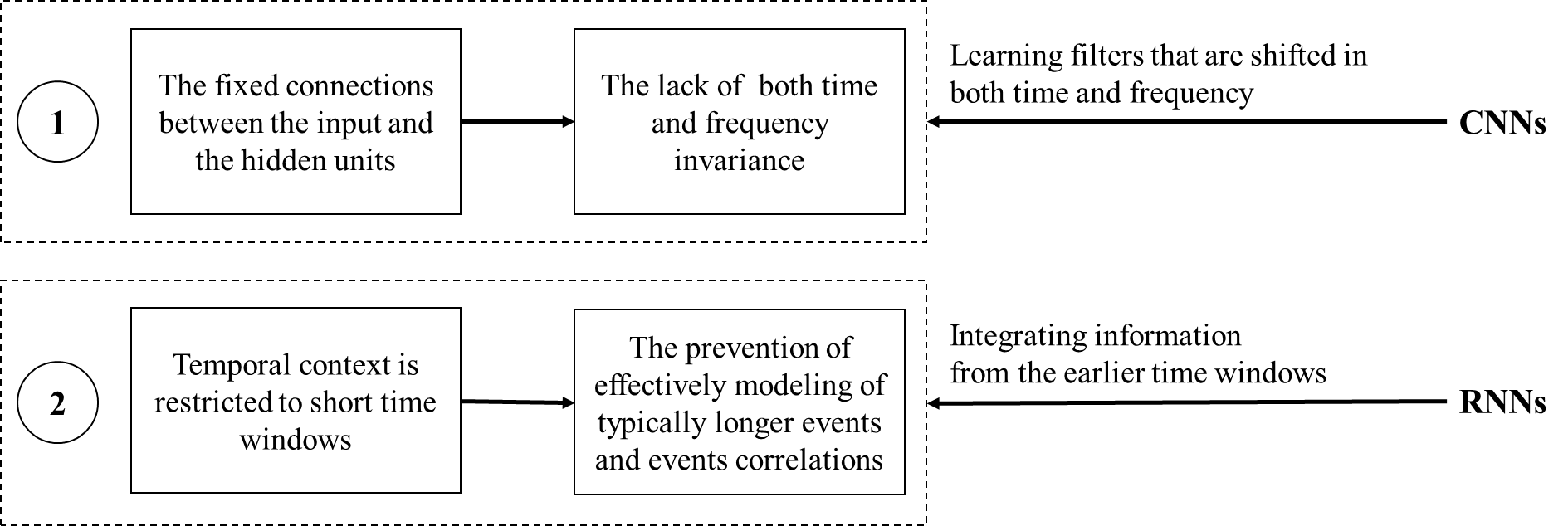


Figure 2.15 Two major shortcomings of FNNs and solutions

In the first drawback, FNNs lack both of time and frequency invariance due to the fixed conntection between the input and the hidden units – which allows to model small variations in the events. CNNs can solve this limitation by learning filters shifted in both time and frequency, lacking however longer temporal context information.

The remaining shortcoming of FNNs is, temporal context is restricted to short time windows, preventing effective modeling of typically longer events (e.g. rain) and events correlations. RNNs can address this restriction by integrating information from the earlier time windows, presenting a theoretically unlimited context information.

So neither CNNs and RNNs can completely address the problem. CNNs lack longer temporal context information, while RNNs do not easily capture the invariance in the frequency domain, rendering a high-level modeling of the data more difficult. CRNN was created to benefit from both approaches, in which the two architectures are combined into a single network with convolutional layers followed by recurrent layers [5].

#### CRNN architecture

As described in the previous section, CRNN is the combination of two of the most prominent neural networks, where CNN followed by RNN. CRNNs are usually used in the context of audio signal processing. The first part of CRNN architecture is CNN, which is used to extract information about amplitude in some range of frequency from Mel-band energy. This information is called “feature map” and represents the identified small time. In order to link this information between consecutive time periods, we need to use RNN following CNN. RNN is essentially a neural network with feedback stage, where the information from one time depends on the information from previous times. Figure 2.16 illustrates the architecture of CRNN.

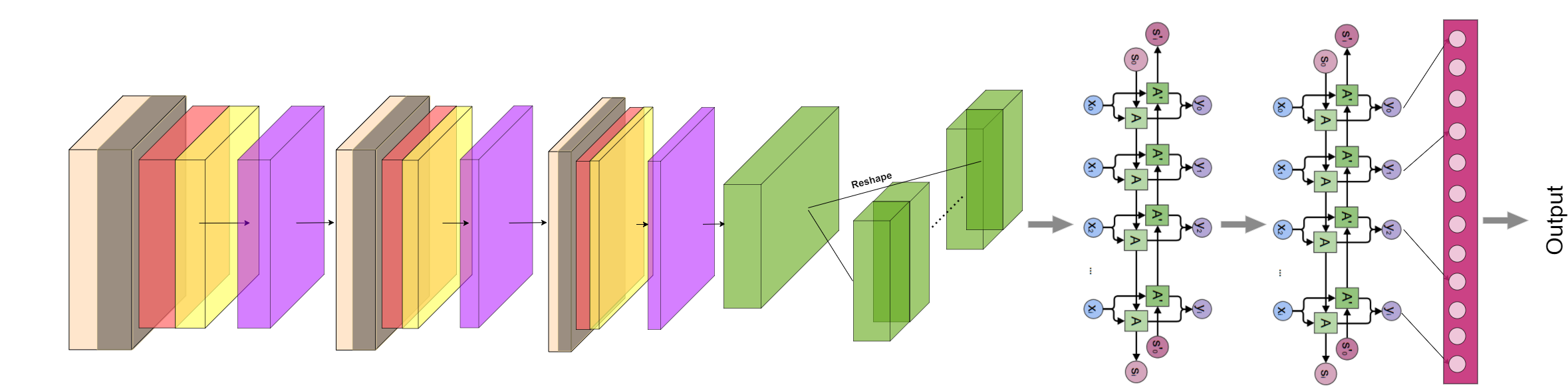


Figure 2.16 CRNN architecture

The network starts with the traditional 2D convolutional neural network followed by batch normalization, max-pooling and dropout with a dropout rate of 50%. Three such convolution layers are placed in a sequential manner with their corresponding activations. The permute layers change the direction of the axes of the feature vectors, which is followed by the reshape layers, which convert the feature vector to a 2-dimensional feature vector. Finally, the output of the bidirectional layers is fed to the time distributed dense layers followed by the fully connected layer [10].

### Autoencoder

The content of this section is referenced from [11]. An autoencoder is a type of artificial neural network that learn efficient data codings in an unsupervised manner. An autoencoder learns a representation for a set of data, typically for dimensionality reduction, by training the network to ignore signal noise. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input.

Figure 2.17 shows the simplest form of an autoencoder.

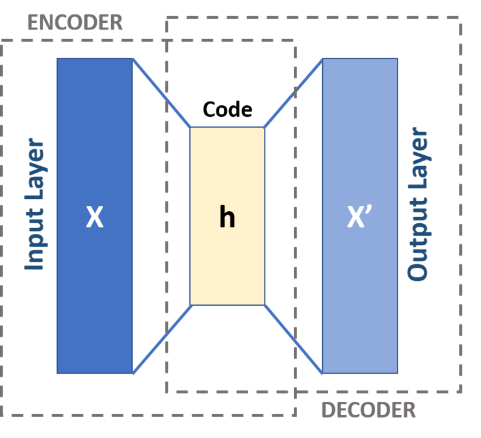


Figure 2.17 A basic autoencoder

This is a feedforward, non-recurrent neural network similar to single layer perceptrons that participate in multilayer perceptrons (MLP). It has an input layer, an output layer and one or more hidden layers. The purpose of an autoencoder is to reconstruct the inputs (minimizing the difference between the input and the output) instead of predicting the target value Y given inputs X.

An autoencoder consists of two parts, the encoder and the decoder. This can be mathematically written as:

where:

* X: Vector space of the input layer
* F: Vector space of the code
* Y: Vector space of the output layer
* : Mapping of the encoder
* : Mapping of the decoder

The h image is referred as code, latent variables or latent representaion. It is mapped from the input x by the encoder

H=σ(Wx+b)

where:

* w: Weight matrix
* b: Bias vector

The decoder maps h to the reconstruction x’:

x’= σ’(W’h+b’)

Autoencoders need to be trained to minimize reconstruction errors:

### Generative adversarial network

A generative adversarial network (GAN) is a class of ML systems invented by Ian Goodfellow and his colleagues in 2014. Two neural networks contest with each other in a game (in the sense of game theory, often but not always in the form of a zero-sum game). Given a training set, this technique learns to generate new data with the same statistics as the training set. An example of GAN is shown in figure 2.18.

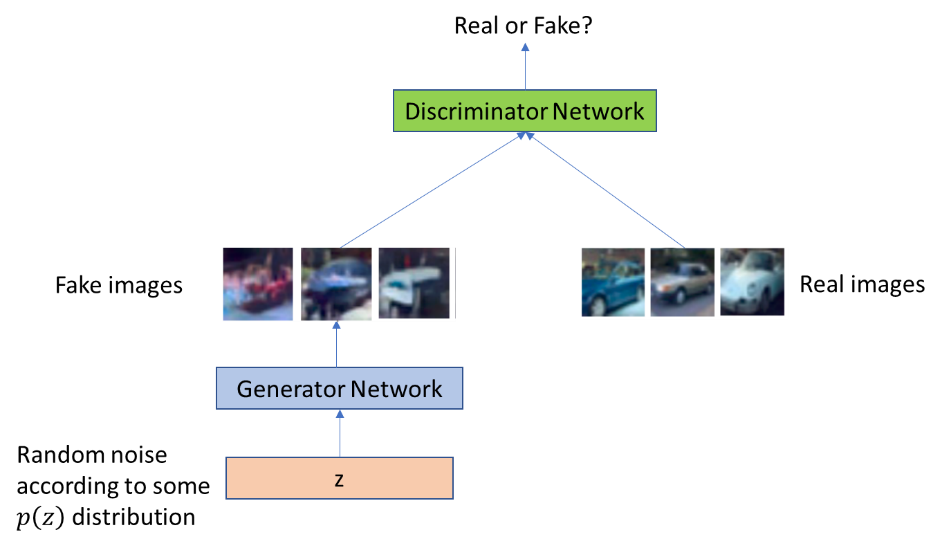


Figure 2.18 An example of GAN

In this example, the generator network is a GAN, which creates fake photographs that look like real images. In other words, the generator tries to fool the discriminator by generating more realistic images. Meanwhile, the discriminator tries to distinguish fake images from real ones. Our task is to train the generator and the discriminator, so that the generator can create pictures looking as similar as possible the real ones. The generative training objective is to increase the error rate of the discriminator.

The generator is typically a deconvolutional neural network, meanwhile the discriminator is a convolutional neural network. Both the generator and the discriminator have fully connected layers. Figure 2.19 illustrates the architecture of Deep Convolutional GAN (DCGAN), which is a class of CNN that have certain architectural constraints, anddemonstrate that they are a strong candidate for unsupervised learning.

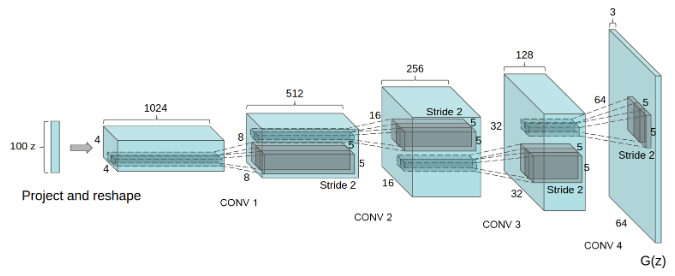


Figure 2.19 The architecture of DCGAN generator

In this topology of DCGAN generator, a 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions then convert this high level representation into a 64×64 pixel image. This architecture has no fully connected or pooling layers used [12].

Two models Generative model G and Discriminative model D participate in a mimax-two player game, defined by the following equation:

where:

* D: A multilayer perceptron is also defined that outputs a single scalar. represents the probability that x came from the data rather than the generator’s distribution
* G: To learn over data x, a prior is defined on input noise variables p(z), then represent a mapping to data space as , where G is a differentiable function represented by a multilayer perceptron with parameters .

The discriminator wants to find such that is close to 1 for real images and is close to zero for fake ones. Then, Generator wants to find such that is close to one [13].

# EXPERIMENTS AND RESULTS

## Datasets

### Basic-shaped signals

This dataset is used to test the abnormal sound detection system with basic-shaped signal − sinusoidal signal, square wave signal and triangle wave signal. They are all periodic waveforms and are widely used in electronics. To create the dataset, we use MATLAB, a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. The dataset consists of 6 wav files generated by MATLAB with sampling frequency is 44100 Hz. The length of each signal is 20 seconds, and the total length of six signals is 120 seconds, which is classified to 6 classes.

Figure 3.1 illustrates the shape of these signal.

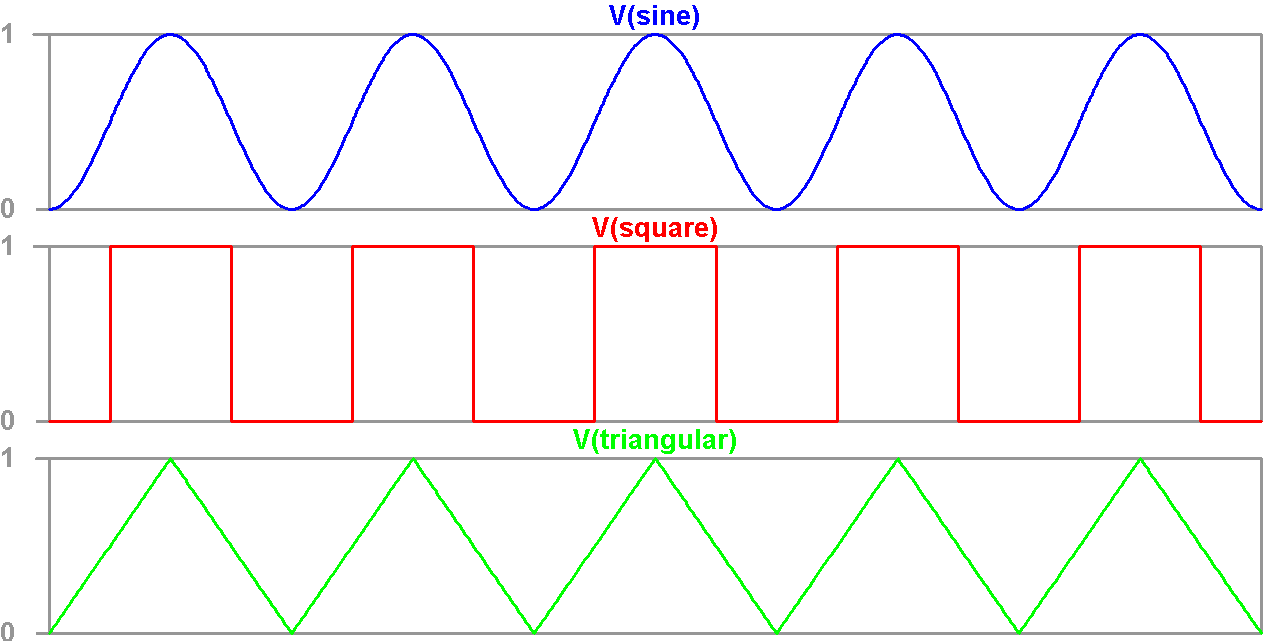


Figure 3.1 Basic-shaped signals

To facilitate the deployment, we encoded the classes as one-hot vector as shown in table 3.1.

Table 3.1 Class of basic shape input

|  |  |  |
| --- | --- | --- |
| **Class name** | **Encoding label** | **Label** |
| Sin (100Hz) | [1 0 0 0 0 0] | 2 |
| Sin (198Hz) | [0 1 0 0 0 0] | 3 |
| Square (25Hz) | [0 0 1 0 0 0] | 4 |
| Square (99Hz) | [0 0 0 1 0 0] | 5 |
| Triangle (50Hz) | [0 0 0 0 1 0] | 6 |
| Triangle (198Hz) | [0 0 0 0 0 1] | 7 |

Encoding label is represented by one hot vector: [y1 y2 y3 y4 y5 y6], where yi =1 mean this segment include ith label (i=1,2, ...6). Figure 3.2 describes how signals are labeled in the dataset.

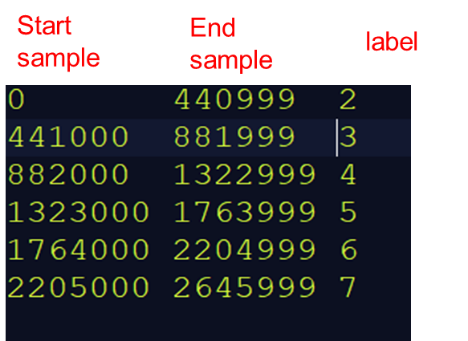


Figure 3.2 Labeling the signals

### Audiobook

In this section, we introduce an Audiobook dataset made by ourselves. This dataset contains reading speeches (in Vietnamese), performed by 2 middle-aged Vietnamese women (Northern accent). They read each chapter of different books and record them with 2 smartphones. Each chapter is saved as a wav file (stereos). All six wav files used have different lengths (from 1200 to 2220 seconds) and each file has only the voice of a single reader. Table 3.2 lists files used in this dataset.

Table 3.2 Files used in the audiobook dataset

|  |  |  |
| --- | --- | --- |
| **Name** | **Length (second)** | **Use** |
| p1 | 990 | test |
| p2 | 1353 | train |
| p3 | 1593 | train |
| p4 | 1244 | train |
| p5 | 1678 | train |
| p6 | 2272 | train |

The sounds that appear in this dataset are classified into six main categories, as described in table 3.3, with label 1 means normal sounds and label 0 implies abnormal sounds). If the segment have both speech and other class, it had labeled as another class (abnormal).

Table 3.3 Categories in the audiobook dataset

|  |  |
| --- | --- |
| **Class name** | **Label (1 mean normal; 0 mean abnormal)** |
| Speech | 1 |
| Flip page | 0 |
| Nasal congestion | 0 |
| Sound by bamboo chair | 0 |
| Impact with micro | 0 |

### Audio recorded in SPARC laboratory

We set up a audio recorder in Signal Processing and Radio Communications (SPARC) laboratory at School of Electronics And Telecommunications of Hanoi University Of Science And Technology.. The system is illustrated in figure 3.3.

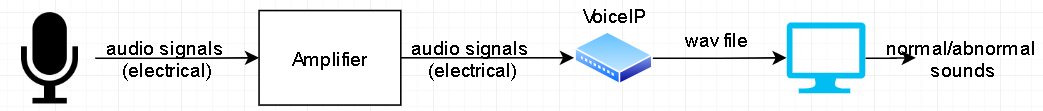


Figure 3.3 Setup diagram for audio recoder in SPARC laboratory

First, a microphone records sounds in the room. Next, electrical audio signals are inserted into an amplifier, and then a VoiceIP device generate wav file from the recorded signals. These files are used for training and testing the abnormal sound detection system. Figure 3.4 is actual images of our recording system.



Figure 3.4 Actual images of hardware setup for the recording system

## Metric

To evaluate the performance of the system, we use F1 metricwhich is usually used in binary classification problem.

F1 is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

When we predict a event (have two case: positive – or happen and negative – or not happen), so we have four cases as in figure 3.5

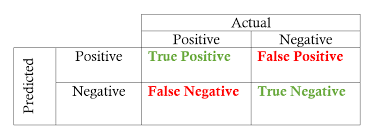


Figure 3.5 Possible cases when predicting an event

Figure 3.5 illustrates possible cases when predicting an event. When predicting an event, we have TP=1 if we predicted event happens (or positive, …), else, TP=0, so on and so ford

The F1 statistic was used as the accuracy measure. In the context of binary classification, F1 is defined as:

where (in each segment I of K segments):

* True positive (TP(i)): total number of events active in both reference and system output segment.
* False positive (FP(i)): total number of events active in system output segment but not in reference
* False negative (FN(i)): total number of events active in reference segment but not in system output

From the equation above can see that FN and FP play a symmetric role in penalizing the accuracy measure F1. The F1 measure is a harmonic mean and suited for situations when the average of rates is desired.

## Experiments and results

### Experiment using basic-shaped signals dataset

#### Experimental setup

From six files in the first dataset, we extract five files (total length is 60 seconds, each class is 10 seconds long) as training data, and one file with a length of 60 seconds (each class is 10 seconds long) as test data.

According to the model in figure 2.1, the data is first put into the feature extraction block. In this block, the STFT block transforms the data to spectrogram, and then the mel filter banks block transforms it to Mel-spectrogram. After going through the Log block, the data at this point is Mel-band energy, and ít is ready to be push into the next block – ANN.

Figure 3.6 shows the spectrogram of basic-shaped signals, which is the output of STFT block.

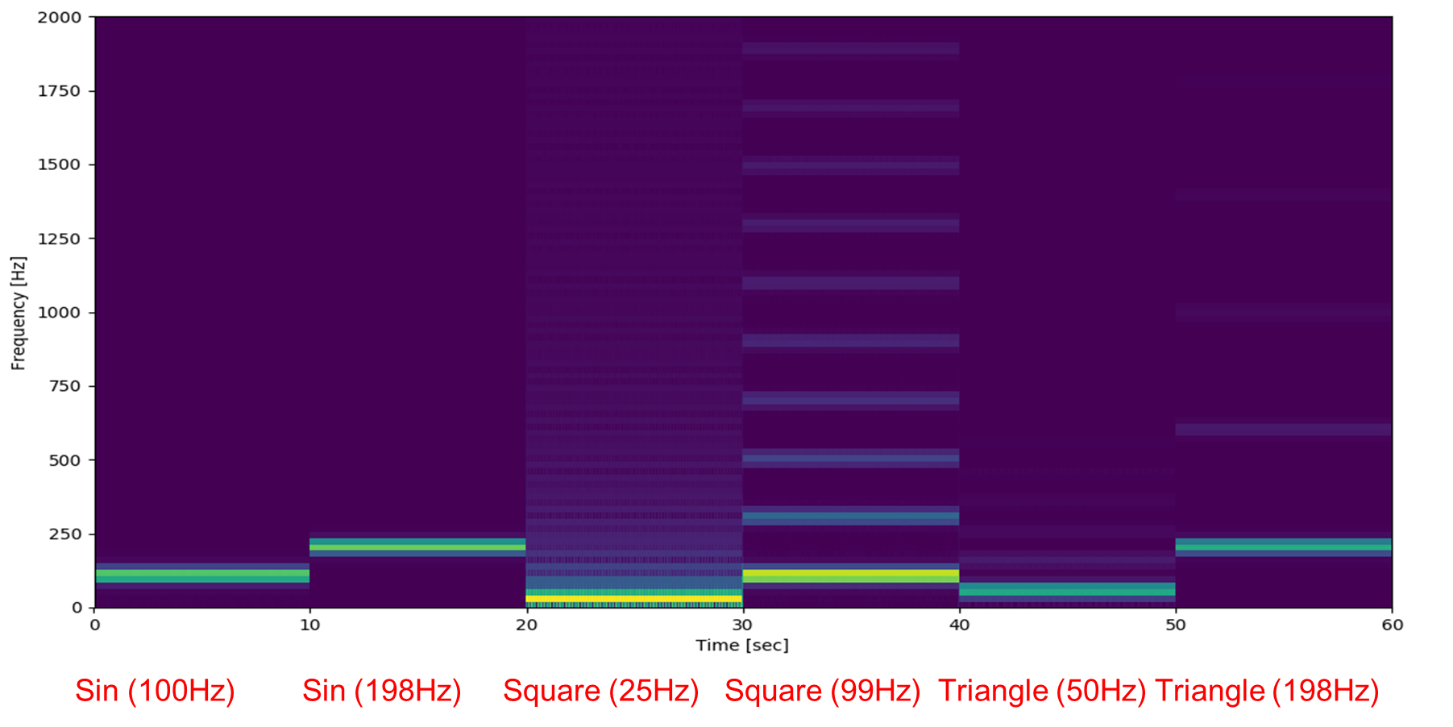


Figure 3.6 Spectrogram of basic-shaped signals

Figure 3.7 shows the Mel spectrogram of basic-shaped signals, which is the ouput of Mel filter banks.

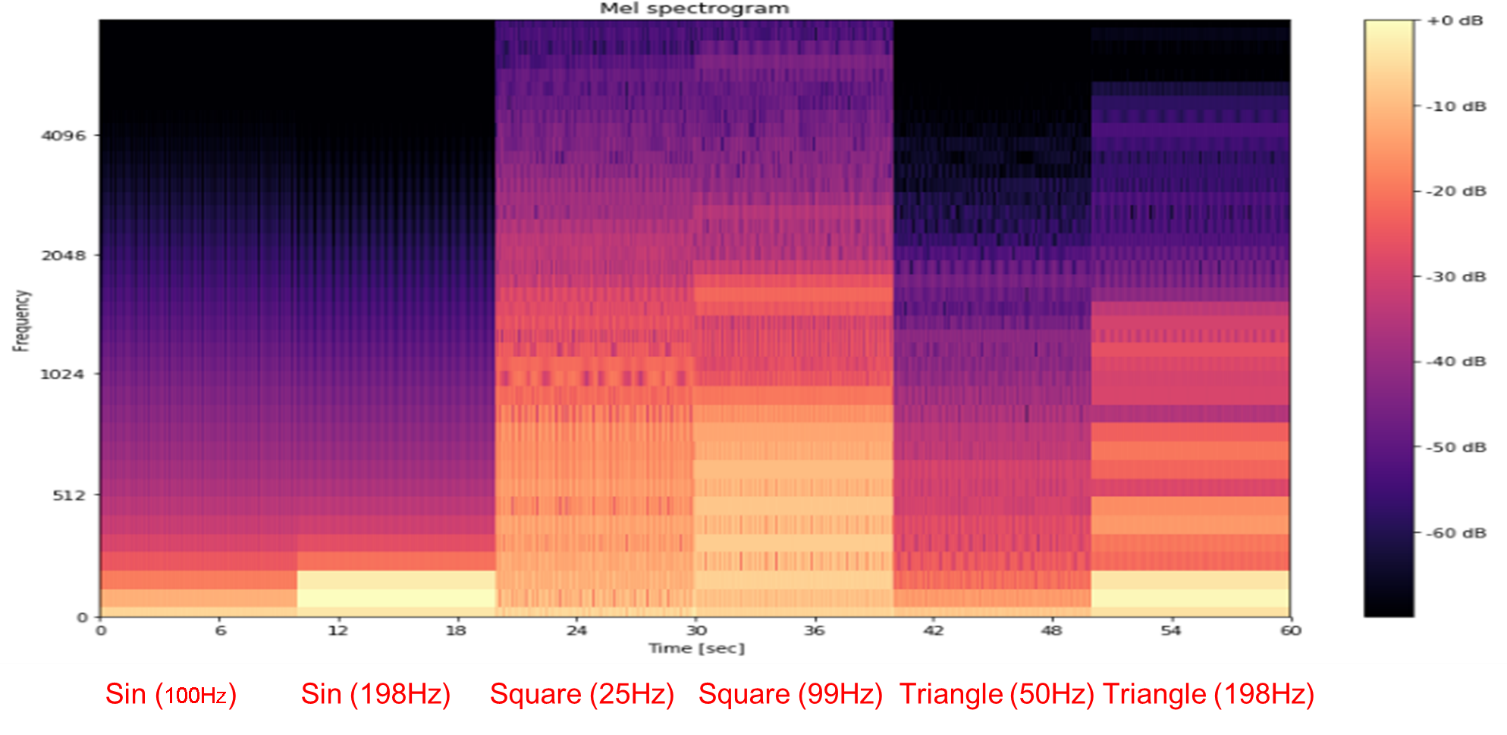


Figure 3.7 Mel-spectrogram of basic-shaped signals

The Mel-spectrogram is then fed into the ANN block. The neural networks that we implement to test are CRNN and GAN.

#### Experiment with CRNN

The output of CRNN is a Nx1 vector (where N is number of class)

[y1 y2 y3 y4 … yN]

whrere yi is the probability presence of the ith sound (i=1,2,..N)

If yi > threshold (ex: 0.5), we determine that ith sound is active in this frame time

After the calculation is complete, we need to save the results to a file so that the user can understand visually, easily access, view, etc. Therefore, we decided to save the file as csv format, because it is easy to open at any operating system, no need special software to open, save storage, and it could bring out intuitively understand when open by Microsoft Excel or a other suitable software.

The content of output file includes 3 field: timestamp, output model value, and output prediction that are presented in table 3.4. The output presents the probability the sound in their timestamp is abnormal. The output prediction has two value 0 and 1 corresponding to normal or abnormal sound detected)

Table 3.4 Output file format

|  |  |  |
| --- | --- | --- |
| **Timestamp (second)** | **Output** | **Output prediction** |
| 14.1234 | 0.123 | 0 |
| 14.1236 | 0.987 | 1 |

In table 3.4, the first example implies that: the output at 14.1234 seconds is 0.123, and it is diagnosed as normal (label 0). The resolution of timestamp depends on a parameter at STFT block. The example 2 is similar.

Figure 3.8 represents the output of CRNN with the inputs are basic-shaped input signals.

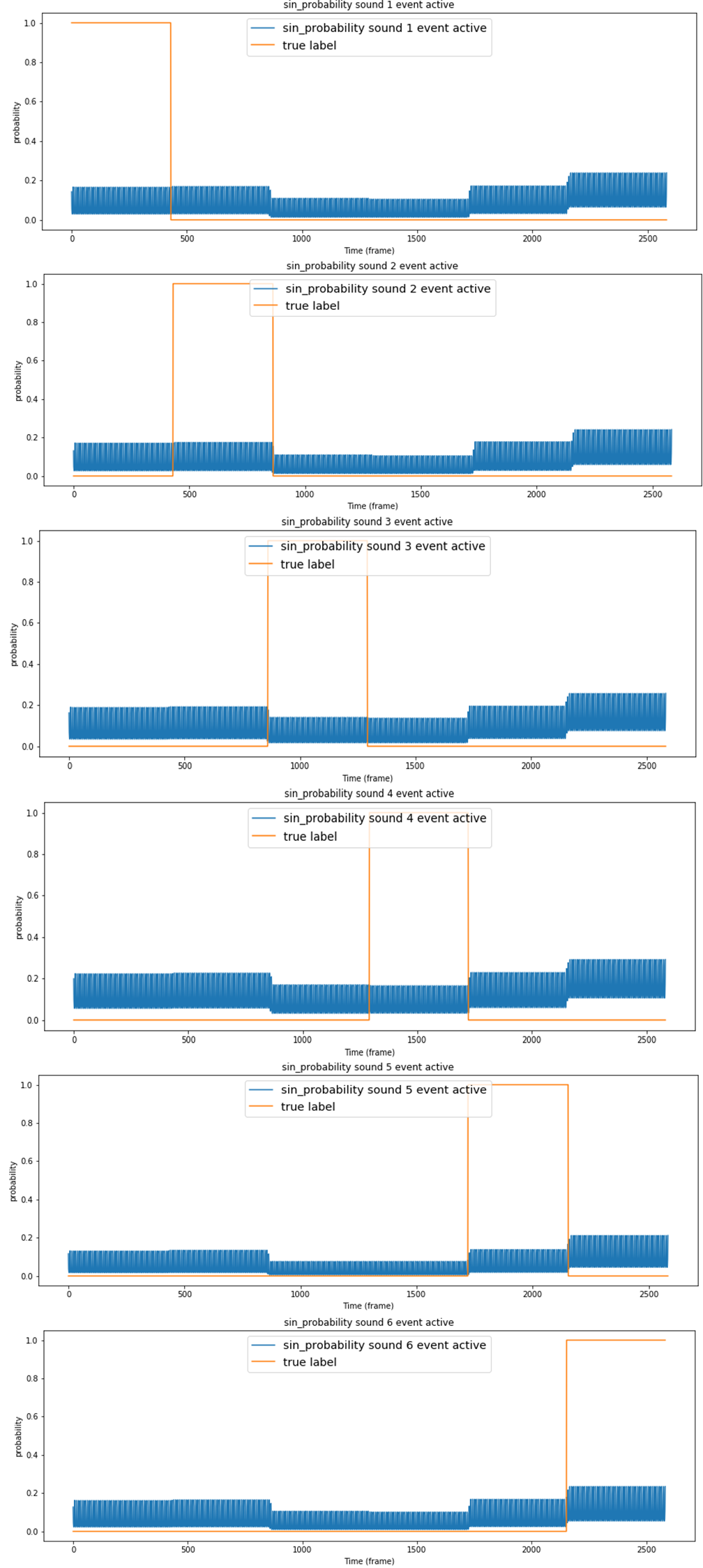


Figure 3.8 the output of CRNN with the inputs are basic-shaped input signals.

#### Experiment with GAN

When feeding Mel-spectrogram into GAN, we separate each signal type from mel-spectrogram to 6 image (64x64 RGB). We use one image to train GAN and another ones to test. We consider different signal type is abnormal case, the output of the discriminator presents probability the input image x is abnormal. We applies a small discriminator model (figure 3.9) so that it can converges. BN is a batch normalization layer following by an activation layer Ac (Leaky Relu).

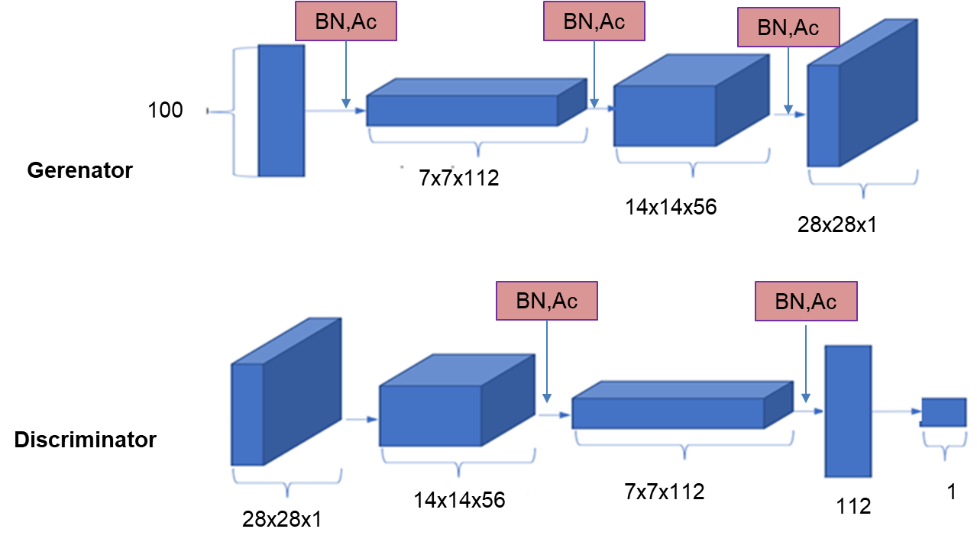


Figure 3.9 Model of the generator and the discriminator

Figure 3.10 shows the discriminator and the generator loss in the training phase.

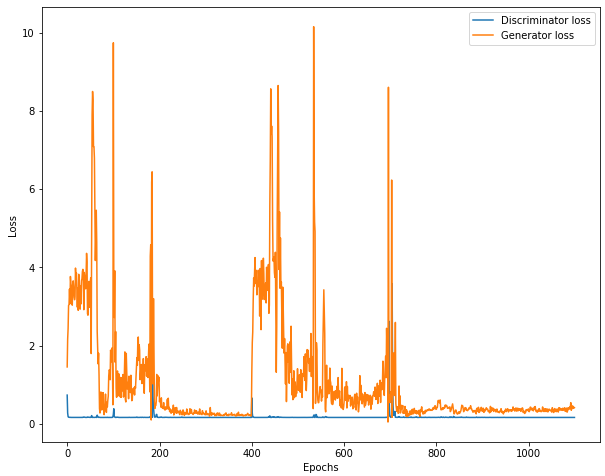
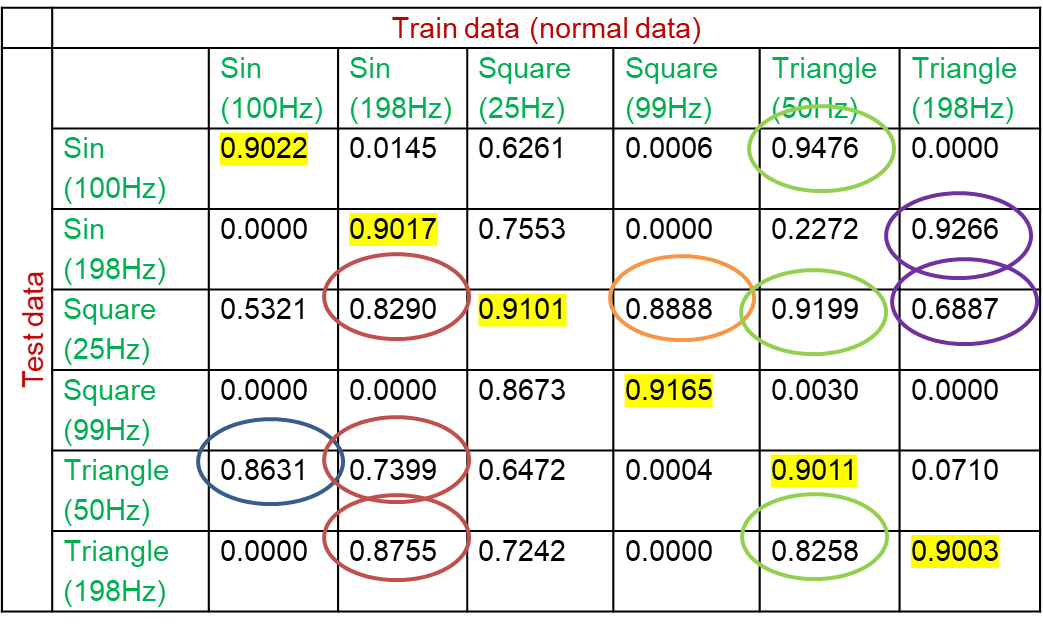


Figure 3.10 Discriminator and generator loss in training phase

As we can observe, the discriminator loss is small, so that the discriminator would have good predictions. Meanwhile, the generator loss at the end of the training phase is also small, proving that G(z) has become more and more like the training data during the training process. Table 3.5 indicates the outputs of the discriminator.

Table 3.5 Ouputs of the discriminator



Let’s take an value at row Sin(100Hz) and column Sin(198Hz) in table 3.5. This value is the discriminator output when using mel-frequency cepstrum (MFC) image of Sin(100Hz) to train and MFC image of Sin(198Hz) to test (close to 1 i.e. the input is more like the train data normal case). However, the discriminator confuses at some images like the others. For example, the blue circle indicates that MFC of sin (100Hz) is like MFC of Triangle (50Hz).

### Experiment using audiobook dataset

#### Experimental setup

In the training phase, we chose the optimizing method is adam [1] . At STFT block, window length for DTFT is 2048 samples (1 time slot is 0.0464s and overlap 50% of window length. In training phase, number of epochs is 111. To evaluate the performance of the system, we use F1 Metric, which usually used at binary classification problem. The Binary-cross entropy is used as loss function:

where:

* : Loss function Binary-crossentropy
* , : True value, predicted value
* : Total of predicted value

#### Experiment with CRNN

Details of the model are illustrated in figure 3.11.



Figure 3.11 Details of CRNN model

Table 3.6 shows the result when using trained model to test file p1.wav.

Table 3.6 Confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Predict** | |
| **Normal** | **Abnormal** |
| **True label** | **Normal** | 41869 | 268 |
| **Abnormal** | 224 | 135 |

The accuracy is 98.84%, but at F1 score, we have Recall = and Precision = . It is because we just have 359 abnormal segments in total 42137 normal segments. In 359 abnormal segments, we just predicted 135 segments is true. Figure 3.12 illustrates the output value of CRNN model and the true label.

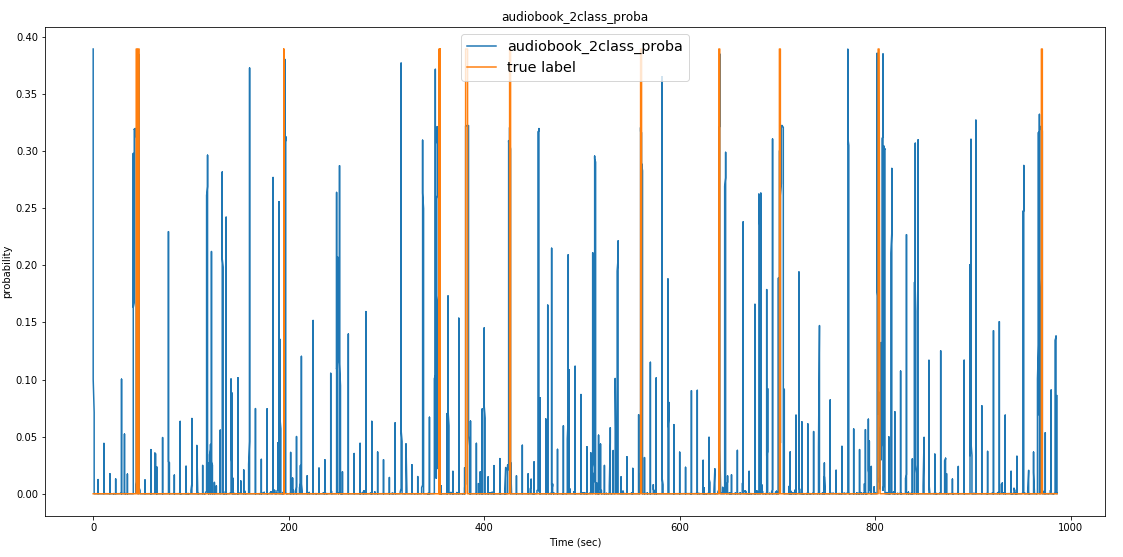


Figure 3.12 output value of CRNN model and the true label

There are a lot of normal segments having high output (i.e we predicted it more like abnormal segments). A reason for low performance is that some segments contain both speech and one abnormal segment sounds also labeled is abnormal segments. Another reason is that in many normal segments there are abnormal sounds, but they are too small for humans to hear. Therefore they are labeled as normal.

### Experiment using audio recorded in SPARC laboratory dataset

Because of litmited working time, we only implement the autoencoder for this dataset. The sounds recorded in our lab have a length of 30 minutes and a ample rate of 8000 Hz. Normal sounds are hum of micro, human speech, and other sounds are anomalous like ‘knock knock’, ‘cuc cuc’, etc.

We use a traditional autoencoder model with the following architecture:

* Input: 200 dimensional vector
* Latent: 512\*3 – 40 – 512\*3
* Output: 200 dimensional vector

To archive the input vector the window has frame\_len is 512 and hop\_len is512/2=206. The Mel-filter band has the number of filters is 40. We calculate Mel band energy (MBE) and combine 5 MBE to archive input vector(40\*5). Figure 3.13 describes details of the autoencoder model

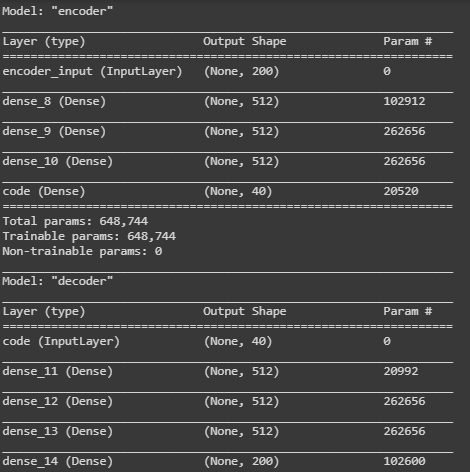


Figure 3.13 Details of autoencoder model

Figure 3.14 is a drawing of the result with label (blue line) and mean squared error (orange line).

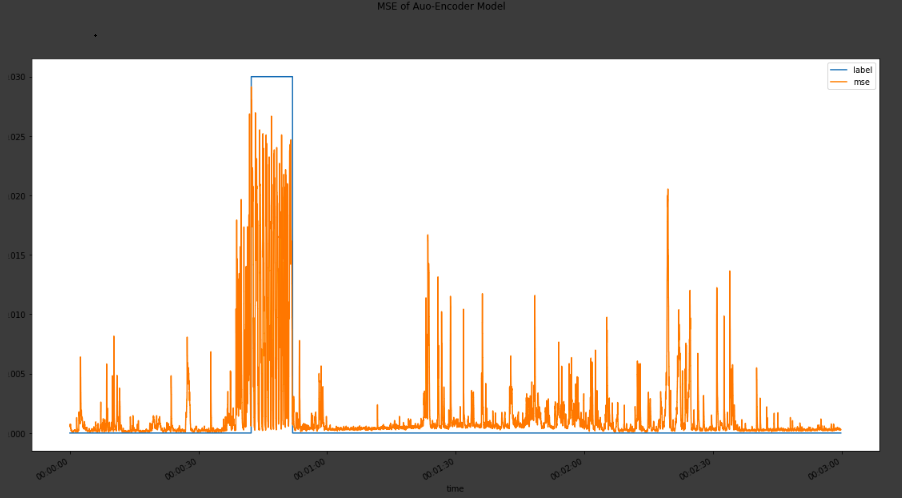


Figure 3.14 Autoencoder implementation result

Table 3.7 shows the evaluation with threshold MSE=2e-2

Table 3.7 Score with threshold MSE = 2e-2

|  |  |  |
| --- | --- | --- |
| **Score** | **MSE = 2e-2** | **MSE = 1.5e-2** |
| **Miss-Detection** | 0% | 0% |
| **False-Detection** | 0 time | 3 times |
| **Acuracy** | 100% | Frame by frame: 74,2%  Window: 100% |

# CONCLUSION

Finally, we have completed the installation of our running system. The environment sound is collected by the microphone and processed transmitted to the computer via Voice-IP terminal. Audio data from ethernet is collected by VoipIP.exe program and is stored as wav file. Sound Anomaly Detection (SAD) software load the wav files and predicts whether the files contain anomaly sound. The analysis results help the administrators further analyses the events to recognize the causes at early stage.

In addition, by comparing the different methods, we found that the CRNN model did not work well for the sound anomaly detection problem with F1 = 34.9%. Moreover, we found that the autoencoder works quite well with this problem, giving F1 = 100% if the threshold is selected correctly. With Voice-IP Japanese dataset, AE give F1=92.68%. However, we also notice the problem with the autoencoder network is that it will give mean square error quite big for some normal segments. So, in the future, we need to keep improving the algorithm!

# REFERENCES

[1] <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>, last accessed on January 18th, 2020

[2] Samuel Arthur L, "Some Studies in Machine Learning Using the Game of Checkers“, IBM Journal of Research and Development, 1959

[3] Tom M Mitchell et al, *Machine learning*, McGraw-Hill Science/Engineering/Math, 1997

[4] Sharath Adavanne, Pasi Pertila, Tuomas Virtanen, “Sound Event Detection Using Spatial Features And Convolutional Recurrent Neural Network”, Tampere University of Technology, 2017

[5] Emre C¸ akır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen, “Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection”, Tampere University of Technology, 2017

[6] <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>, last accessed on January 18th, 2020

[7] <https://en.wikipedia.org/wiki/Feedforward_neural_network>, last accessed on January 18th, 2020

[8] <http://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-tensorflow/>, last accessed on January 18th, 2020

[9] <https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/>, last accessed on January 18th, 2020

[10] https://towardsdatascience.com/an-approach-towards-convolutional-recurrent-neural-networks-a2e6ce722b19, last accessed on January 18th, 2020

[11] <https://en.wikipedia.org/wiki/Autoencoder>, last accessed on January 18th, 2020

[12] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah and Mohammed Bennamoun, *A Guide to Convolutional Neural Networks for Computer Vision,* Morgan & Claypool Publishers, 2018

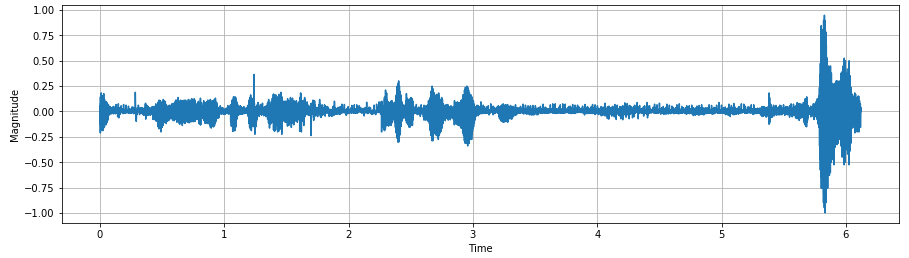
[13] <https://medium.com/@sh.tsang/review-gan-generative-adversarial-nets-gan-e12793e1fb75>, last accessed on January 18th, 2020

# APPENDIX

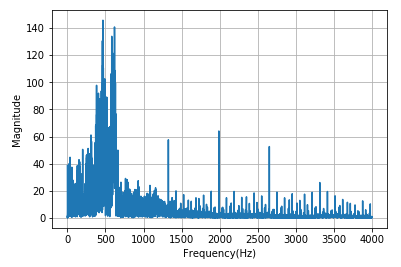
## Appendix 1. Short-time Fourier transform

* Idea of STFT

Signal in time domain will give a sense of amplitude, but do not know the frequency components in it.



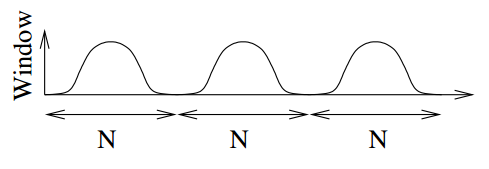
Signals in frequency domain will give clarity on frequency components, but do not show the order in which these frequencies appear in the time domain.



Thus, we need a transformation that satisfies both frequency clarity and the order in which these frequencies occur, which is the origin of STFT.

* STFT
* Windowing

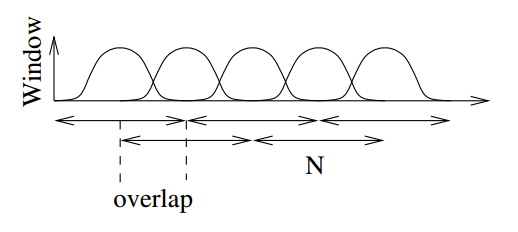
To split signal to frames, we use windows likes Hanning, Hamming, … instead of using rectangular window, because of using rectangular window causes signal to be wrong in frequency domain.



Look at the figure above, we see that magnitude of signal decreases to zero at both ends of each window. It means that the signal is attenuated in amplitude in time domain. So we need a solution to solve this problem, “Overlap” is one of solutions.

* Overlap

Overlap is a technique of stacking windows on top of each other with a certain percentage (20%, 30%, 50%...) to avoid attenuation at both ends of each window. The figure below illustrates windowing with 50% overlap.



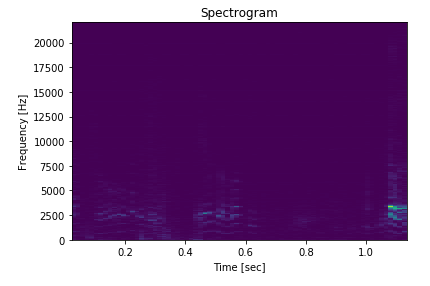
* STFT

After windowing, the original signal is disjointed into frames. To achieve STFT, we perform DFT (Discrete Fourier transform) in each frame:

where:

* x[n]: Signal is disjointed in time domain
* w: Type of window function
* m: Index of window function

To visualize STFT, we use a chart called spectrogram, this is power spectrum of disjointed signal. The figure below is spectrogram of disjointed signals.



We introduce another concept "Power spectrum" given by:



where:

* P: Power spectrum
* x: disjointed signal
* NFFT: Length of DFT window

## Appendix 2. Source code

Our source code is available at:

<https://drive.google.com/open?id=1INh5o1JDYaRVtKCEXTyt50upX1t2EgWY>