Application of Machine Learning and Deep Learning in Bioacoustics

Kripa Nath

A dissertation submitted for the partial fulfilment of BS-MS dual degree in Science

Indian Institute of Science Education and Research, Mohali

April, 2022

Dedicated

To my younger self
who always dreamed of doing something
in the field of
mathematics and programming.

Certificate of Examination

This is to certify that the dissertation titled **Application of Machine Learning and Deep Learning in Bioacoustics** submitted by **Kripa Nath** (Reg. No. MS17008) for the partial fulfillment of BS- MS Dual Degree programme of the institute, has been examined by the thesis committee duly appointed by the institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Neeraja Sahasrabudhe
(Committee member)

Dr. Shane D'Mello
(Committee member)

Dr. Manjari Jain
(Co-Supervisor)

Dr. Vaibhav Vaish
(Supervisor)

Dated: April 28, 2022

Dated: April 28, 2022

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Vaibhav Vaish and Dr. Manjari Jain at the Indian Institute of Science Education and Research, Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Kripa Nath
(Candidate)
Dated: April 28, 2022

In my capacity as the supervisor/co-supervisor of the candidates project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Dr. Manjari Jain
(Co-Supervisor)

Dated: April 28, 2022

Dr. Vaibhav Vaish
(Supervisor)

Dated: April 28, 2022

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Dr. Vaibhav Vaish. Without his help and supervision, this thesis would have never been possible. I would also like to thank him for giving me space and freedom to explore the interdisciplinary field. His constant guidance and support have proved to be a massive help during the course of this thesis work.

I would also like to thank Dr. Manjari Jain, my co-supervisor and our collaborator in this thesis project, for trusting me with the problem from her field and for her guidance throughout the project.

I would also like to thank my thesis committee members, Dr. Neeraja Sahasrabudhe and Dr. Shane D'Mello, for their inputs and support during this thesis work.

I am thankful to my family, who have been there all these years, strongly supporting and guiding me in life. My sister Shweta and my brother Kumar Gaurav have been the reason I would always keep going, for I know their aspirations when they look up to me.

I would also like to express my deepest gratitude to my best friend, Prarthna Saraswat, for her advice, her patience, her faith in me, and her constant love and support. I am also grateful to my brothers from another mother, Abhishek Roy, and Parbhat Kumar, for their constant love and guidance in academics and beyond.

I would also like to take this opportunity to express my heartfelt thanks to my senior, Vishvadeep Vashisht and Nishant Rajora, who has provided me with enormous help and guidance throughout. I am grateful to Vivek Chauhan and Manisha Rohilla for making me believe in a better tomorrow when I was at my lowest. I would also like to thank Khusbhoo Yadav, Nitika Goyal, Anjali Gupta, Utkarsh Diwakar, Shivangini Jaryal, Kaustuv Ghosh, Gaurav Singh, C Afzal, Sneha Mehra, Vineet Gaurav, Agantuk Saha, and Pallav Baraya for their constant love and support.

I wholeheartedly thank IISER Mohali for providing me with the workspace and a sound environment to pursue my research.

To conclude, above all, it was almighty God and his blessings which has made this endeavor a successful venture.

Abstract

Today, the field of artificial intelligence is making major progress in almost all its standard sub-areas, and its breakthrough advancement has excellent application in a variety of domains. It has applications in biomedical imaging, drug discovery, surveillance and monitoring, speech recognition, autonomous vehicles, etc. This has been possible because of the accumulation of volumes of data which are the building blocks of these advancements.

The environment around us is rich in acoustic information. One such piece of information is based on the vocalizations of animals, also known as bioacoustics. This includes not only the vocalizations of animals such as birds and mammals but also the sounds that insects can produce. Depending upon the research purposes, bioacoustic data can be generated by deploying a recorder in the fields, but these field recordings may or may not contain the specific target, and hence, it requires a large number of bioacoustic recordings for proper analysis. However, it will still require much human labor to extract the specific target out of this massive data if done manually. Even after retrieving the target, we still rely on the manual procedure to obtain onsets/offsets, features of the acoustic signals, or any other relevant applications.

Through this thesis research, we have first developed an understanding of machine learning and deep learning. We have explored its application to reduce this labor and vigorous process by building a model that can automatically detect the region of interest from the field recordings and can extract them for further analysis. We also explore an automated method where the extracted target recordings can be automatically analyzed to extract various features for research purposes.

List of Figures

Contents

Chapter 1

Introduction

1.1 What is Data?

In today's world, data is everywhere. With the arrival of digital technology, a vast amount of data is generated every second. It is collected, analyzed, shared, hacked, bought, and sold. Data are collected from observations or measurements represented as text numbers or multimedia. We may immediately think of numbers when we think of data, but data can also be field notes, videos, audio recordings, photographs, and documents. Data can be broadly categorized as qualitative or quantitative. **Quantitative data** can be expressed as a number and counted or compared on a numerical scale. At the same time, **qualitative data** is observation or description and cannot be expressed as a number.

In this emerging world of technology, "data" plays a significant role, especially in the field of artificial intelligence. Data can be in any form, including numbers, images, videos, recordings, etc. When a collected set of relevant data are processed, one can find many hidden trends or significant relationship among those sets of data, and by working on those details, one can draw a meaningful result. This process is called **data mining**. Data mining is analyzing data, pre-processing data to transform it into an appropriate format, and discovering the insights or trends hiding behind the data. These insights and patterns are mined and extracted using various tools and techniques ranging from data visualization, statistical modeling, machine learning or deep learning, etc.

The environment around us is rich in acoustic information. Acoustics is the science of sound. The sound moves through a medium by alternately contracting and expanding parts of the medium it is traveling through. This compression and expansion create a pressure variation relative to the medium pressure that propagates in the form of waves to which the animal ear is sensitive. When particles of the medium come closer, it is called compression, whereas when the particles go farther than their normal position, it is called rarefaction, and these regions are alternately the region of high pressure and low pressure. These changes

in air pressure create a wave. Therefore, a sound wave can be represented as pressure variation in the medium through which it propagates. These sounds can be recorded using a microphone, and the digital form of sound is referred to as audio. Microphones convert these sounds into a digital format so that a computer can store them. It is a kind of analog-to-digital converter that converts pressure variation into some numerical parameters that define the properties of a sound. These parameters are the amplitude of a sound wave representing its volume and the frequency representing its pitch. Unlike tabular data, audio data does not follow a very clear and organized structure, and therefore, it is hard to work with it.

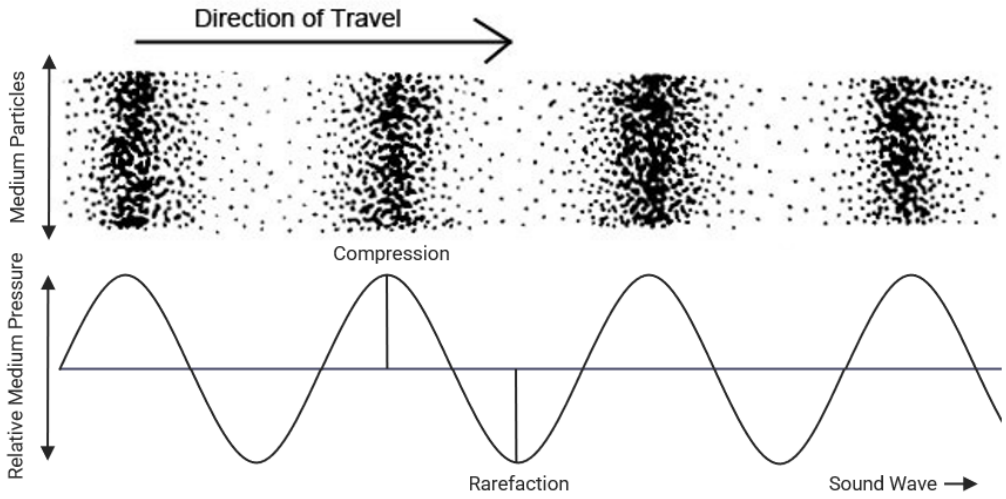

Figure 1.1: Representation of Sound Waves

1.2 Problem Statement

This thesis research focuses on machine learning and deep learning, aiming to model the problems based on specific acoustic signals belonging to the animal kingdom. Bioacoustics is a branch of science concerned with sound production and its effects on living organisms. It has a range of novel application areas, such as acoustic surveillance, essential for automated wildlife monitoring and overall ecosystem health. It also contributes to research on climate change [Penar 20].

Bioacoustic in conservation has a lot of potential uses. Because we can not only hear the animals themselves that we are trying to study, like the birds, the insects, the frogs, even some mammals, but we can also hear what is happening to the habitat. The other

advantage of bioacoustics is that it is easy not to be disturbing the biodiversity we are studying. We can deploy our recorders in the field, leave them for the appropriate period, and it can record volumes of data that can be used for research purposes. Depending upon the research purposes, these field recordings may or may not contain the specific target. To deal with this, the recordings are done for a long time, resulting in a massive volume of data. However, it still requires much human labor to extract the specific target out of this massive data if done manually. Through this project, we are trying to build a model that automatically detects the region of interest from the field recordings and can extract them for further analysis. We also explore an automated method where the extracted target recordings can be analyzed further automatically for extracting various features for research purposes.

1.3 Understanding Audio data

Like any other signal, anything that passes information from one source to another, sound signals also carry information through the vibration. Two main features of any sound signal are amplitude and frequency.

1.3.1 Time Domain Representation of Sound Wave

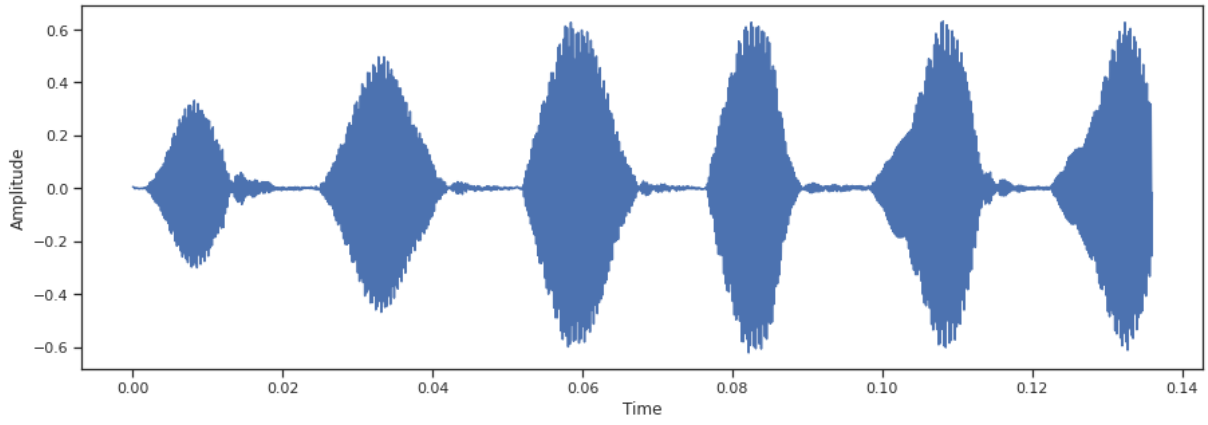

Figure 1.2: Time Domain Representation of Sound Wave

The above visualization of sound is called the time-domain representation of a sound wave. It represents how a sound wave's amplitude(or loudness) changes with time. The region with amplitude zero represents silence.

1.3.2 Frequency Domain Representation of Sound Wave

The amplitudes in the time domain representation talk about the loudness of sound, and therefore they are not very informative. The other way to understand these signals is the frequency domain representation, which tells us how much of the signal lies within each given frequency band over a range of frequencies. This time-domain to frequency-domain conversion is done via Fourier Transform.

Audio files with just one frequency at the same time are very rare. In general, any sound signal is a superposition of many sound signals, and when sound is recorded, we only capture the resultant amplitudes of those multiple waves. But we can decompose a signal into its constituent frequencies by using Fourier Transform.

Mathematically, for any continuous signal, g(t), its Fourier transform is given by

$$\hat{g}(f) = \int g(t).e^{-2\pi ft}dt \tag{1.1}$$

Similarly, for discrete signal, x(t), we have,

$$\hat{x}(f) = \sum x(n).e^{-2\pi fn} \tag{1.2}$$

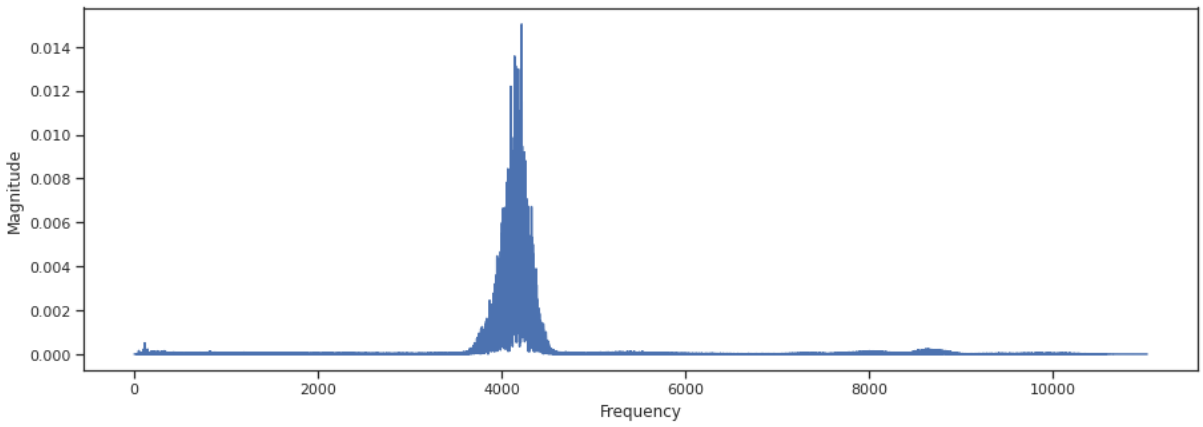

Figure 1.3: Frequency Domain Representation of Sound

Fig 1.3 represents the frequency domain representation of a sound which is obtained by computing the Fourier transform of the sound wave shown in Fig 1.2. The magnitudes for frequencies are higher only for the frequency value ranging between 3500 Hz to 4500 Hz and have a very small magnitude in other regions as most of these frequencies are probably due to the noise.

1.3.3 Time-Frequency Representation - Spectrograms of Sound Wave

However, when we apply Fourier transform to our signal, it only gives us the frequency values, and all the information based on time is lost. For example, a sound recording of a human speech will make no sense after applying Fourier transformation as it will only give frequency values. To preserve both time and frequency information, we use the visual representation of the spectrum of frequency of a signal as it varies with time which is known as **spectrograms**. A spectrogram is a frequency-time domain representation of a sound where it chops up the time duration of the signal into a small time window(generally in milliseconds) and then computes the frequencies contained in each window by applying the Fourier transform for each smaller window. Also, the consecutive windows overlap with each other so that we don't lose any of the frequencies. It then combines the Fourier transforms for all those segments into a single plot.

A human ear can easily differentiate between pairs of lower frequencies than the pairs of higher frequencies, even if the difference between those pairs is the same in both cases. This implies that even though the frequency difference between the two sets of sounds is the same, our perception of those differences is not. It has been found that humans perceive sound frequencies logarithmically and not linearly, and that log scale is referred to as the mel scale.

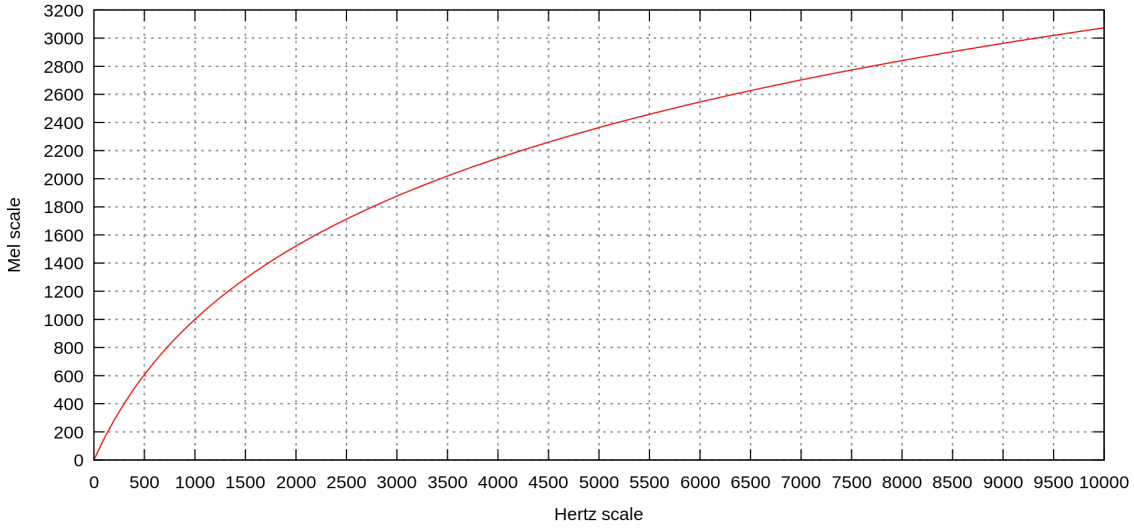

Figure 1.4: Mel Scale vs Hertz Scale

The transformation of frequency from Hertz scale to mel scale is given by:

$$mel = 1127.\ln(1 + \frac{f}{700}) \tag{1.3}$$

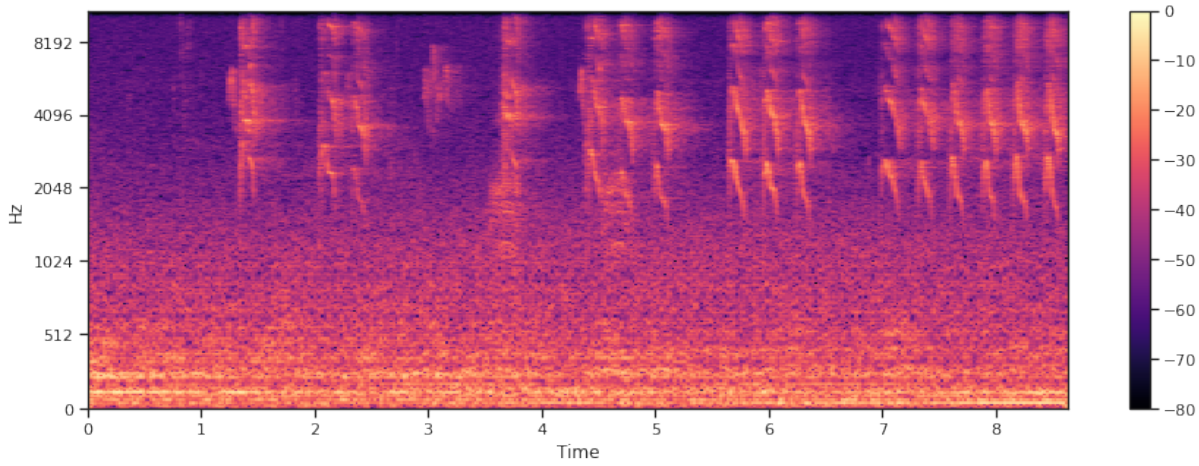

Figure 1.5: Mel-Spectrogram of a Sound Wave

This mel spectrogram now is a proper representation of sound waves without losing time or frequency information.

The recorded audio can be analyzed based on different parameters extracted from time-domain representation, frequency domain representation, and frequency-time domain representation. These features are the basis for sound event detection and segmentation of audio into voiced and unvoiced regions. Also, these feature helps in creating a feature set for the targeted regions of recording, which can be used for further analysis and to create models.

1.4 Audio Features

A sound signal can have a voiced region or unvoiced region (either silence or noise or both). It may also have many voiced regions with different levels of amplitudes and frequencies. Based on the different representations of a sound wave, one can extract useful features from these signals and these different regions. When the features are calculated for the whole signal, it will give small or little information about the signal because the amplitude and frequency vary with time for a sound signal. Therefore, observing these features locally for small-time windows that carry more information is important. Some of these features are discussed below:

1.4.1 Zero Crossing Rate

The zero-crossing Rate(or ZCR) of a discrete-time signal is defined as the number of times the zero axes are crossed per frame. When observed locally, ZCR correlates with the frequency content of the signal, and hence it helps in identifying the voiced and unvoiced

region of the signal as ZCR for the voiced region is usually low when compared to the unvoiced region, which has a higher ZCR value.

$$ZCR = \frac{1}{n}.\sum_{1}^{n} |sgn(x(n+1)) - sgn(x(n))| \tag{1.4}$$

where, x(n) is the signal and

$$sgn(k) = \begin{cases} 1, & \text{if k} > 0 \\ 0, & \text{if k} = 0 \\ -1. & \text{if k} < 0 \end{cases} \tag{1.5}$$

1.4.2 Energy

The energy of a sound wave is directly proportional to the square of the wave's amplitude. The average energy of a discrete-time signal x(n) is given by,

$$E(x(n)) = \frac{\sum_{1}^{n}(x(n))^2}{n} \tag{1.6}$$

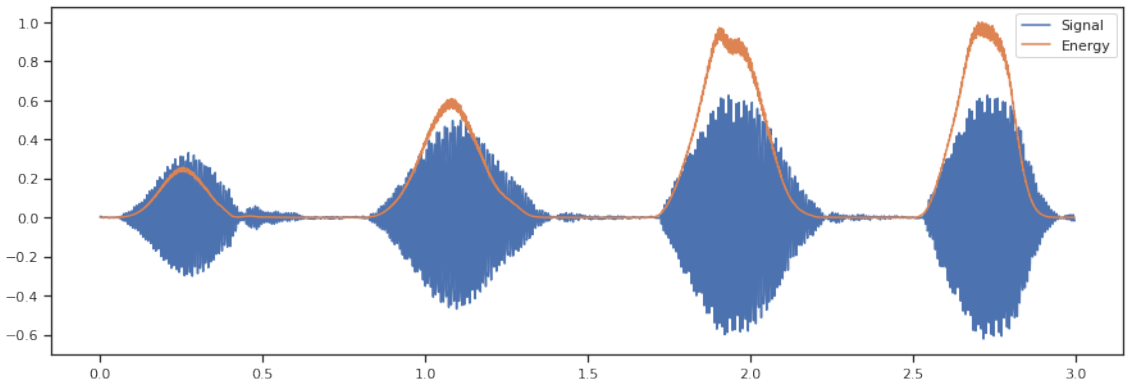

Figure 1.6: Short Term Energy

When energy is computed for the whole signal, it gives little information about time-dependent features for many audio signals. Since amplitude varies with time for a sound wave, computing this energy for small-time windows will give more information. This energy will have a higher value for the voiced region than the unvoiced region.

1.4.3 Amplitude Envelope

The amplitude envelope of an audio signal is a curve that shows the boundary within which the signal is contained. This can be obtained by computing the analytical signal for the given audio signal. An analytic signal is a complex-valued function that has no negative

frequency components. The real and imaginary parts of an analytic signal are real-valued functions related to each other by the Hilbert transform. The Hilbert transform of a signal x(t) is the convolution of $x(t)$ with $1/\pi t$. Mathematically, Hilbert transform is defined as:

$$x_H(t) = x(t) * \frac{1}{\pi t} = \int_{-\infty}^{\infty} \frac{x(t)}{\pi(t-\tau)} dt \tag{1.7}$$

Fourier transform of convolution of two functions is the product of their Fourier Transform. Therefore, Fourier transform of Hilbert transform of signal x(t) will be given by,

$$F(x_H(t)) = F(x(t)).F(\frac{1}{\pi t}) \tag{1.8}$$

We also know that,

$$sgn(t) \xleftrightarrow{\text{F.T}} \frac{2}{i\omega} \tag{1.9}$$

$$\implies \frac{2}{it} \xleftrightarrow{\text{F.T}} 2\pi.sgn(-\omega) \tag{1.10}$$

$$\implies \frac{1}{\pi t} \xleftrightarrow{\text{F.T}} -i.sgn(\omega) \tag{1.11}$$

$$\implies x_H(\omega) = -i.sgn(\omega).x(\omega) \tag{1.12}$$

From expression, 1.11, we can observe that the Hilbert transform shifts he phase of original signal by $\frac{\pi}{2}$. Using this, we can represent our sound signal in form of analytic signal.

$$\implies x_{analytic}(t) = x(t) + ix_H(t) \tag{1.13}$$

And $|x_{analytic}(t)|$ represent the amplitude envelope. These envelopes in themselves are very useful. These envelopes give a rough idea of loudness, and they can also be used for the onset detection of the voiced region in an audio signal and segmentation of the audio signal into voiced and unvoiced regions.

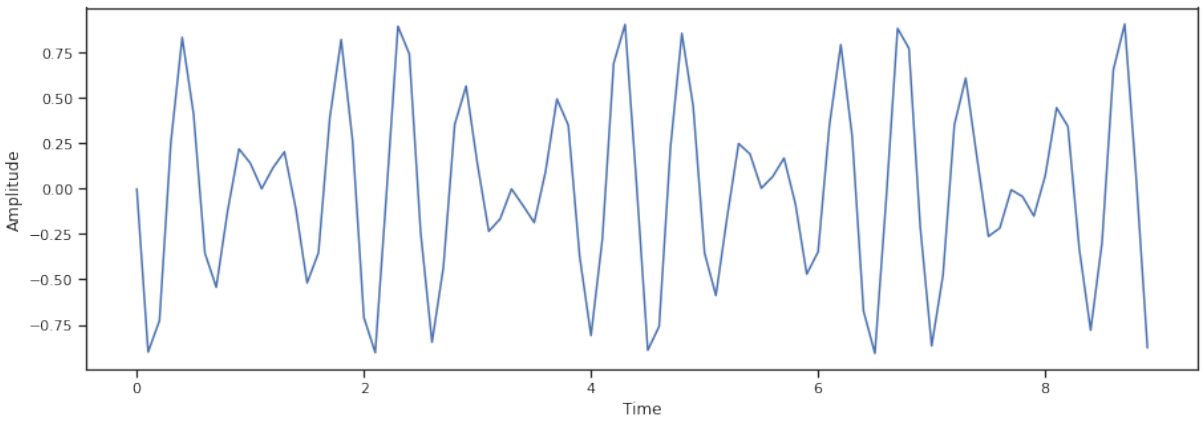

Figure 1.7: Original Audio Signal

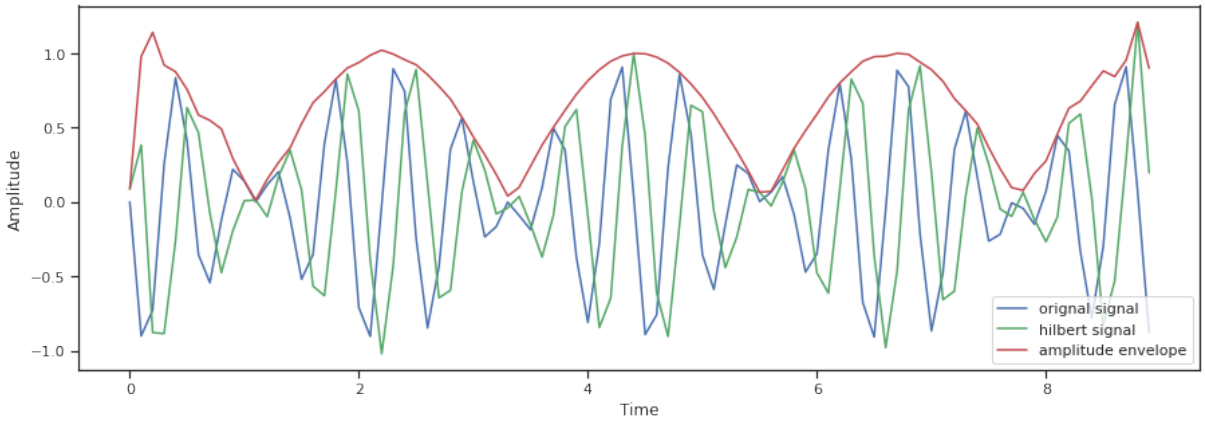

Figure 1.8: Signal with Hilbert Transform and Amplitude Envelope

1.4.4 Signal Variance

The variance of the signal is defined as the variance of the amplitudes. It is defined as,

$$Var(x(n)) = \frac{1}{n-1}\sum_{1}^{n}(x(n) - x(\bar{n}))^2 \tag{1.14}$$

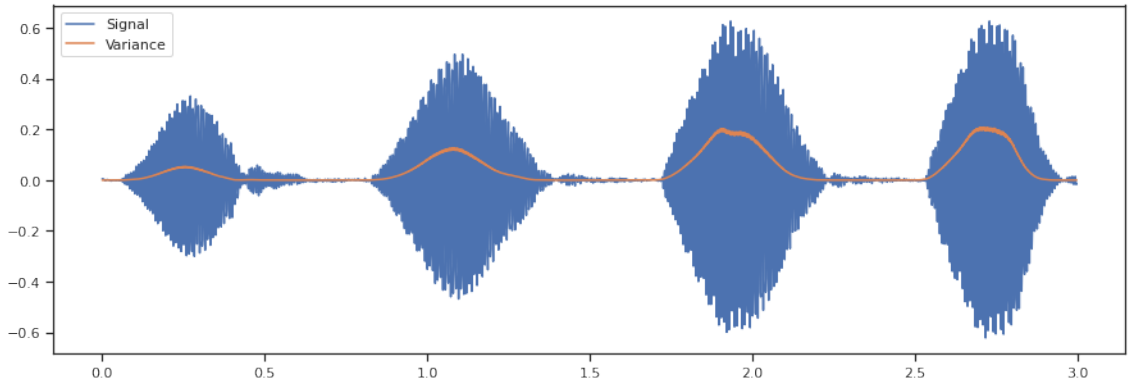

Figure 1.9: Short Term Amplitude Variance

Computing the variance for the whole signal will give a single numerical value, while computing it for small-time windows shows the variation of signal amplitude locally. The unvoiced region usually has a smaller amplitude variance, while the voiced region has a higher amplitude variance.

1.4.5 Spectral Centroid

The spectral centroid is the center of gravity of the magnitude of the frequency representation of the signal. It is defined as the weighted mean of the frequencies present in the

signal, determined using a Fourier transform, with their magnitudes as the weights.

$$SC = \frac{\sum_1^n A(k).k}{\sum_1^n A(k)} \tag{1.15}$$

where k is the frequency and A(k) is the corresponding magnitude of frequency.

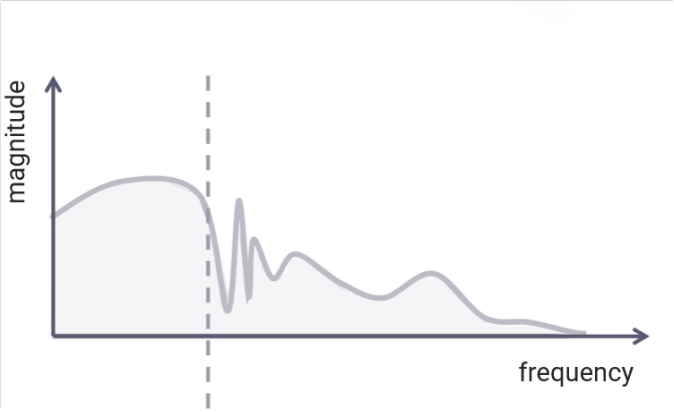

Figure 1.10: Spectral Centroid

1.4.6 Spectral Bandwidth

The spectral bandwidth is measured as the deviation of the frequency of a signal from the spectral centroid.

$$SB(k) = \sqrt{\frac{\sum_1^n A(k).(k-SC)}{\sum_1^n A(k)}} \tag{1.16}$$

where k is the frequency and A(k) is the corresponding magnitude of frequency and SC is the spectral centroid of the signal.

In one of our thesis experiment (Chap 4, Part A), we have implemented the approach mentioned above in python to automate the procedure of segmentation of voiced and unvoiced regions and feature extraction.

Chapter 2

Theoretical Background

2.1 The Traditional Programming Paradigm

Traditional computer programming has been around for more than a century. It includes the computer program, which is created manually after figuring out the specific rule of thumbs between inputs and outputs. Then, putting back those rules and inputs in the machine will give us the desired output. There is a growing variety and volumes of data available in the current data-driven world. Sometimes the data we collect can be complex and challenging to utilize unless organized and appropriately categorized. Hence finding the rule of thumb for our traditional programming setup becomes more arduous, and we may not be able to find the exact algorithm every time for every given data. Moreover, that is precisely where artificial intelligence and machine learning come into play. The main characteristic feature of machine learning is finding rules using existing examples.

2.2 Artificial Intelligence and Machine Learning

In 1956, Dartmouth Professor John McCarthy was the first to coin the term ”artificial intelligence.” He was the first computer scientist and one of the founders of the discipline of artificial intelligence. AI is the branch of computer science that deals with developing an intelligent machine capable of performing any task with human-like intelligence. AI enables machines to mimic, develop and demonstrate human cognition or behavior. The main motive of AI is to make these machines less artificial and more intelligent. Later in 1959, Arthur Lee Samuel popularized the term “machine learning.” According to him, machine learning(or ML) is the field of study that gives computers the ability to make intelligent decisions based on their learning without being explicitly programmed.[Samuel 59] It involves training the machine by making a model using different algorithms with a mathematical foundation and testing the level of training of the model using a test set.

Figure 2.1: Traditional Programming Paradigm

According to Tom M.Mitchell, the concept of learning in ML is that it learns from experience E w.r.t to some task T and some performance measure P if its performance on T as measured by P improves with experience E.[Mitchell 83] Unlike traditional programming, ML is the concept where we feed the input and output to the machine, which tries to learn the relationship between them after a proper analysis. ML is built upon a statistical framework. It is evident because ML involves data, and data has to be described using a statistical framework. It is an interdisciplinary field that uses linear algebra, probability & statistics, and computer science to learn from data and provide insights that can be used to make predictions.

2.2.1 Types of Machine Learning:

ML comprises several types of learning based on the problem, and three major recognized categories are supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning

In supervised learning, an ML model is trained using labeled data where the model needs to find the mapping function to map the independent variable with the dependent variable. It is a type of predictive learning algorithm in which data comes with specific labels where the labels are the target we are interested in predicting. Consider an image classification problem where we have images of dogs and cats, and each of them is labeled as 'dog' and 'cat', respectively. Now the ML model would use previous data to predict the label of new data points.

Unsupervised Learning

Unsupervised learning differs from supervised learning because this ML model is trained using unlabeled data. However, since unlabeled data only consist of features, the goal here is to find the underlying structure of the dataset, group that data based on similarities, and represent the data set in the compressed structure. Consider the same example of image classification mentioned in supervised learning but this time with no labels. We can still differentiate both the animals based on similarities of their features using the method of

unsupervised machine learning, which will group them in some way by similarity, even without knowing what each group represents.

2.3 Model, Objective Function, and Parameter Learning - Core of Machine Learning

The primary objective of any ML model is to make an accurate prediction after learning from the data. It is an interdisciplinary field that uses probability and statistics, and therefore to understand how an ML algorithm learns from data and improves its learning to predict an outcome with high accuracy, we need to understand the underlying concepts involved in training the algorithm.

2.3.1 Dependent and Independent Variable:

Independent variables are the inputs of a specific process, and the outputs of the process are the dependent variables. Independent variables are also referred to as features.

2.3.2 Population and Sample

The entire set of all the elements associated with a problem statement is population. Since we can never measure the entire population, we always consider a subset of the population to draw inferences about the whole population. This subset considered for studies is called the sample. Any value that describes the character of an entire population is a parameter of population, and any statistics of the sample is the measure that describes the sample.

2.3.3 Probability and Statistics

Probability tells about the likelihood of future events, whereas statistics deals with analyzing the observations of past events. The probability occurrence of an event is defined as the number of times the event occurred (or the number of ways the particular event can occur) divided by the total number of incidents observed (or the total number of possible outcomes). Statistics is any measurable function of the data, for example, mean, median, etc. Statistics term is used both for the function and for the value of the function on a given sample.

2.3.4 Estimators, Likelihood Function and Maximum Likelihood Estimator

Given a data set, estimation in the statistical approach to find an estimate of a population parameter, and when a sample statistics are used for estimating a population parameter, it is called an estimator.

For example, the sample mean is an estimate of the population mean. Sample variance is an estimate of the population variance. Sample standard deviation is an estimate of the population standard deviation.

The difference between the expected value of an estimator of a population parameter and the respective true parameter value gives the **bias** of an estimator. When this difference is zero for an estimator, we call it an **unbiased** estimator of the population parameter.

For a statistical experiment resulting in n independent and identically distributed random variables X_i where i = 1,2,...n, with a probability distribution P_θ parametrized by some parameter θ, the **likelihood function**(L) is defined as a probability when the true value of the parameter is θ.

$$L(\theta|X_1,X_2,...X_n) = \prod P_\theta(X_i = x_i) \tag{2.1}$$

The maximum likelihood principal is a method of obtaining the optimum values of the parameters that define a model. Moreover, while doing so, we increase the likelihood of our model reaching the "true" model. For the above model with a given likelihood function, the maximum likelihood estimator of θ is defined as:

$$\hat{\theta}_n^{MLE} = argmax_\theta L(\theta|X_1,X_2,...X_n) \tag{2.2}$$

2.3.5 Loss Function

The loss function is a technique to evaluate the performance of an algorithm or model. It is a function that measures how far an estimated value is from its true value. Tuning the model parameter based on its loss functions helps us to improve the model accuracy. The objective of any model or algorithm is to minimize the loss function and tune the model parameter using some optimization techniques.

$$J(\theta) = \frac{\sum f(predicted,target)}{n} \tag{2.3}$$

where *f(predicted,target)* is a function of model parameter θ

Types of Loss Function:

Mean Absolute Error: For any dataset and model associated with it, MAE is measured as the average of the absolute difference between the estimated value(predicted output) and its true value(actual output).

$$Loss = \frac{\sum_1^n ||\hat{y}^i - y^i||}{n} \tag{2.4}$$

where $\hat{y}^i$ is predicted output and y^i is actual output.

Mean Squared Error: For any dataset and model associated with it, MSE is measured as the average of the square of the difference between the estimated value(predicted output) and its true value(actual output).

$$Loss = \frac{\sum_1^n ||\hat{y}^i - y^i||^2}{n} \tag{2.5}$$

where $\hat{y}^i$ is predicted output and y^i is actual output.

Cross-Entropy Loss: For any dataset and model associated with it, cross-entropy loss or log loss is a measure of divergence between the probability distributions.

$$Loss = \sum_1^n -y^i log(\hat{y}^i) \tag{2.6}$$

where $\hat{y}^i$ is predicted output and y^i is actual output.

2.3.6 Optimizers in Machine Learning: Newton Raphson and Gradient Descent Method

The model's performance is evaluated by the loss function, and this loss can be reduced if model parameters can learn from the loss associated with them and then by updating the model parameters accordingly to reduce the loss. This process is repeated till the convergence of the loss function, and the process is known as optimization. There are several optimization techniques used in machine learning.

For any function $f(x)$, continuous and differentiable, then for any h in neighbourhood of x, using Taylor series for first order approximation, we can write:

$$f(x+h) = f(x) + hf^{'}(x) + O(h^2) \tag{2.7}$$

Therefore, in first order approximation, $f(x+h)$ is given by $f(x)$ and $f^{'}(x)$ at x. Now it's clear that in order to get maximum leverage, for small h, moving in the direction of

negative gradient will decrease f. Therefore, $h = -\alpha f^{'}(x)$ where $\alpha > 0$. Then,

$$f(x - \alpha f^{'}(x)) = f(x) - \alpha f^{'}(x) f^{'}(x) \tag{2.8}$$

Also for small value of α,

$$f(x - \alpha f^{'}(x)) \leq f(x) \tag{2.9}$$

This means if use $x - \alpha f^{'}(x)$ instead of x, $f(x)$ will decline. This method, when repeated continuously, under some constraints, $f(x)$ will converge to a minimum value. This optimization method is known as **gradient descent**. In machine learning, it is used to minimize the cost function parameterized by the model's parameter by updating the parameters in the opposite direction of the gradient of the cost function with respect to the model parameters.

$$\theta_j : \theta_j - \alpha * \frac{dJ}{d\theta_j} \tag{2.10}$$

where J is the loss function and dependent on parameter θ_j and α is the learning rate that scales the magnitude of parameter updates.

- Since the first-order approximation is good only in the small neighbourhood of x, a smaller value of α is used.
- If α is very small, the algorithm takes very long to converge to the minimum and consumes more computation power.
- If α is very high, then there is a high chance of overshooting away from the minimum as our first order approximation will lose its virtue and there will be the contribution from higher order terms.

Another more accurate approach is the second-order approximation compared to the first-order approximation. For a small value of h, by using the Taylor series, a second-order approximation can be written as:

$$f(x+h) = f(x) + hf^{'}(x) + \frac{1}{2}h^2 f^{''}(x) + O(h^3) \tag{2.11}$$

On differentiating w.r.t. h,

$$f^{'}(x+h) = f^{'}(x) + hf^{''}(x) \tag{2.12}$$

Thus, the first order condition for the value of h, that minimizes $f^{'}(x+h)$ is

$$0 = f^{'}(x) + hf^{''}(x) \tag{2.13}$$

$$\implies h = -\frac{f^{'}(x)}{f^{''}(x)} \tag{2.14}$$

Considering the condition that h lies in the neighbourhood of x, we can re-write h as $h = -\alpha \frac{f'(x)}{f''(x)}$ where $\alpha > 0$.

Therefore if use $x - \alpha \frac{f'(x)}{f''(x)}$ instead of x, $f(x)$ will decline and under valid constraints, it converges to the local minima if applied iteratively. This optimization technique via second-order approximations is known as **Newton-Raphson method**. To minimize the cost function of a machine learning model, Newton's method is more efficient in such cases as they use the curvature to find the direction of minima in the curve. To minimize the cost function, J, parameterized by model parameters θ, the parameters will be updated by the following rule:

$$\theta_j : \theta_j - \alpha * \frac{\nabla J(\theta_j)}{\nabla^2 J(\theta_j)} \tag{2.15}$$

where J is the loss function and dependent on parameter θ_j and J has to be twice differentiable in order to calculate $\nabla^2 J(\theta_j)$ and α is the learning rate that scales the magnitude of parameter updates.

- If α is very small, the algorithm takes very long to converge to the minimum and consumes more computation power.

- If α is very high, then there is a high chance of overshooting away from the minimum.

Wherever applicable, Newton's method converges much faster towards a local maximum or minimum than gradient descent. But for a model with a large number of model parameters, the computation of the Hessian matrix(second order derivative) and finding the inverse of the Hessian in high dimensions can be an expensive operation. It also fails whenever a stationary point is encountered. Therefore, in general, gradient descent is used for optimizing the cost functions of an ML model. However, there are several ways to boost this algorithm. Generally, a model uses the whole training set to update the model parameter in each iteration. For a large volume of data, it is very heavy on computation. To fasten this process, instead of using the whole training set at once for each iteration, we can randomly pick one data point from the whole training set at each iteration to reduce the computations enormously. This method is called **stochastic gradient descent(or SGD)**. We can also sample a small number of data points from the training set instead of using just one at each iteration and it is called **mini-batch gradient descent(or MBGD)**.

Sometimes the cost function for a model can have local minima, and if it is a bit shallow, the normal gradient descent algorithm finds it hard to escape, and sometimes it becomes a vigorous process, but SGD and MBGD can escape it easily.

2.4 Fundamental Machine Learning Model

To establish notation for this section, we will use n to denote the number of total features, m to denote total number of data points in a dataset and x_j^i represents the j^{th} feature's value in the i^{th} training example where i=1,2,...,m and j=1,2,...,n. Let x^i denote the i^{th} input features of the training example, and y^i is the corresponding output. The pair (x^i,y^i) is called a training example, and the whole data set for i = 1,2,...,m is called the training set. Considering a training example,linear combination of features is processed in the hypothesis function depending upon the model and this happens for all

$$\theta_0^i + x_1^i\theta_1^i + ... + x_n^i\theta_n^i \tag{2.16}$$

where the term θ_0^i is called bias.

This can be further generalized in matrix form using all the features and including the constant variable 1 in the feature of each training example. This matrix is known as the feature matrix, the matrix of corresponding parameters is known as the parameter matrix, and it is given by,

$$X = \begin{bmatrix} 1 & x_1^1 & x_2^1 & ... & x_n^1 \\ 1 & x_1^2 & x_2^2 & ... & x_n^2 \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ 1 & x_1^m & x_2^m & ... & x_n^m \end{bmatrix} \quad \Theta = \begin{bmatrix} \theta_0^1 & \theta_1^1 & \theta_2^1 & ... & \theta_n^1 \\ \theta_0^2 & \theta_1^2 & \theta_2^2 & ... & \theta_n^2 \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ \theta_0^m & \theta_1^m & \theta_2^m & ... & \theta_n^m \end{bmatrix} \tag{2.17}$$

2.4.1 Linear Regression:

It is one of the most popular supervised machine learning algorithms. It is a statistical method used to predict output based on given features. The output is the linear combination of one or more independent variables. The hypothesis function used here is given by

$$h(\theta) = \Theta X^T \tag{2.18}$$

$$y^i = h^i(\theta) = \theta_0^i + x_1^i\theta_1^i + ... + x_n^i\theta_n^i \tag{2.19}$$

Cost Function: In this model, we usually use the mean squared error to calculate the cost and update the model parameters. Squared error functions are differentiable almost everywhere and give non-negative errors. Also, being a quadratic function, it has only one minimum, and gradient descent performs very well in this case. Absolute error functions or higher-order functions are avoidable because of non-differentiable behavior or because they may have more than one local minima, which does not give an optimal solution for our model parameters.

2.4.2 Logistic Regression:

It is also a popular supervised machine learning algorithm. The important thing to note about logistic regression is that it is not a regression but a classification algorithm. Using this model, we try to find out a decision boundary that can divide the dataset into different classes. It is used to predict the categorical dependent variable, and therefore, the outcome must be a discrete value(or categorical). Because of this, the hypothesis function is a sigmoid function that gives probabilistic values between 0 to 1, and with some threshold parameter, the values above/below the threshold value are classified as 1/0 (or True/False), respectively.

$$h(\theta) = \frac{1}{1+e^{(-\Theta X^T)}} \tag{2.20}$$

$$y^i = h^i(\theta) = \frac{1}{1+e^{(-\theta_0^i - x_1^i\theta_1^i - \ldots - x_n^i\theta_n^i)}} \tag{2.21}$$

Cost Function: Consider a case of dog and cat image classification problem. An ideal model will predict,

$$P_{\theta^i}(y^i|x^i) = \begin{cases} 1, & \text{if } y^i = 1(\text{dog}) \\ 0, & \text{if } y^i = 0(\text{cat}) \end{cases} \tag{2.22}$$

We can also write this as,

$$P_{\theta^i}(y^i|x^i) = P_{\theta^i}(y^i = 1|x^i)^y x P_{\theta^i}(y^i = 0|x^i)^{1-y} \tag{2.23}$$

By using logistic regression, this unknown probability function is modeled as

$$P_{\theta^i}(y^i|x^i) = \frac{1}{1+e^{(-\theta^i x^i)}} \tag{2.24}$$

We can write the likelihood function as,

$$L(\theta^i|y,x) = \prod_{i=0}^{i=m} P_{\theta^i}(y^i|x^i) \tag{2.25}$$

Using expression 2.23, we can re-write likelihood function as:

$$L(\theta^i|y,x) = \prod_{i=0}^{i=m} P_{\theta^i}(y^i = 1|x^i)^y x P_{\theta^i}(y^i = 0|x^i)^{1-y} \tag{2.26}$$

Instead of analyzing likelihood function, we can also analyze log-likelihood.

$$\log L(\theta^i|y,x) = \sum_{i=0}^{i=m} [y^i . \log P_{\theta^i}(y^i = 1|x^i) + (1-y^i) . \log P_{\theta^i}(y^i = 0|x^i)] \tag{2.27}$$

We need to maximize this log-likelihood, or we can minimize the negative of this log-likelihood, which can be done via gradient descent to estimate the model parameters. Hence the cost function (also known as cross entropy loss or log loss) in this case is given by,

$$J(\theta) = -\sum_{i=0}^{i=m} [y^i . \log P_{\theta^i}(y^i = 1|x^i) + (1-y^i) . \log P_{\theta^i}(y^i = 0|x^i)] \tag{2.28}$$

2.5 The Problem of Under-fitting, Over-fitting and Regularization Techniques

While training any model on the given data set, the aim is to learn from the data and create a generalized set of rules that can perform well on the new data set. Sometimes a model fails to learn from the data if it does not get trained until the convergence of the cost function, too many input features where some of them are irrelevant or missing. This case is referred to as the **under-fitting**. In the case of under-fitting, a model has low efficacy both on training and test data.

Over-fitting is a case in which a model fits the training data too well but performs very poorly on test data. Instead of learning from the hidden trends, the model also starts to learn from noise, which negatively impacts the model's ability to generalize.

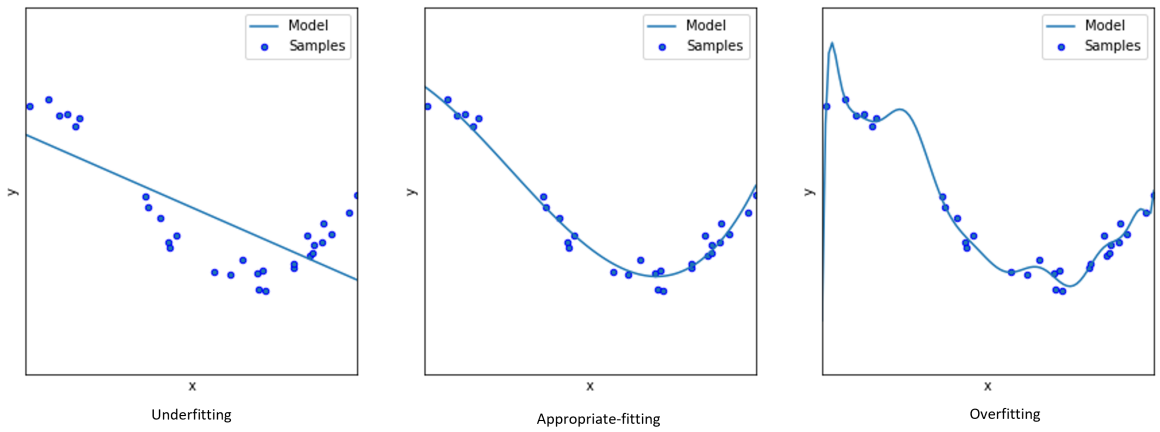

Figure 2.2: Under-fitting vs Appropriate-fitting vs Over-fitting

Regularization is a technique that is used to avoid the problem of overfitting for any model. In practice, it penalizes the model parameters so that the model does not learn the more complex features to generalize the model. Depending upon the different learning models, there are different regularization techniques. These regularization techniques can be applied to the cost function, which optimizing gives a better model without affecting the training accuracy and also increases the overall accuracy of the model.

$$Los(New) = Loss + \lambda * \sum g(||model parameters||) \tag{2.29}$$

where λ is the regularization parameter and g is some function of model parameters.

Chapter 3

Artificial Neural Network and Convolutional Neural Networks

3.1 Deep Learning

Machine learning is just a subset of AI that focuses on teaching an algorithm to learn from experiences without being explicitly programmed. Deep learning takes this idea even further, and it is a subset of machine learning that focuses on using neural networks to automatically extract useful patterns in raw data and then using these patterns or features to learn to perform that task.

3.2 Artificial Neural Networks

”Artificial neural network(or ANN)” refers to a biologically inspired sub-field of artificial intelligence modeled after the human brain. An artificial neural network consists of a pool of computing units called perceptron(also known as an artificial neuron) which communicate by sending signals to each other over a large number of weighted connections.

3.2.1 Basic Components of ANN:

Perceptron:

Perceptrons were first developed by Frank Rosenblatt, inspired by the work of Warren McCulloch and Walter Pitts. Perceptrons are the structural building block of artificial networks and deep learning. It takes one or more input variables and produces an output variable and considering the impact factor of different input variables.

Weight and Bias:

Weights are associated with each input variable depending on its influence on the model, which decides the impact factor of an input variable. The more the weight of input, the more it will influence the network. A perceptron receives input as a linear combination of input variables and weights and a bias term. Bias is like the intercept in a linear equation. It is used to adjust the output along with the weighted sum of input variables that help to shift the activation function.

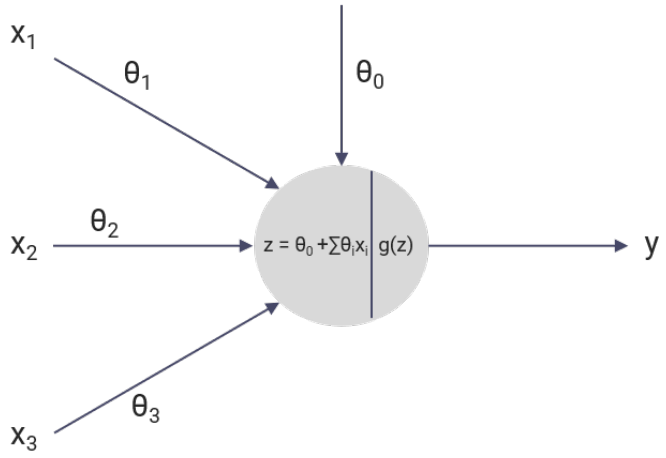

Figure 3.1: Perceptron with input variables x_i, weights θ_i, bias θ_0 and output y

Activation Function:

When the weighted sum of input variables along with the bias term is passed through a perceptron, it takes that sum and passes it through an activation function, g , to produce the final output.

$$y = g(z) \quad ,where \quad z = \theta_0 + \sum_1^3 \theta_i x_i \tag{3.1}$$

The main purpose of the activation function is to introduce the non-linearity in the network, which allows us to actually deal with non-linear data, and this is extremely important in real life, especially because in the real world, data is almost always non-linear. Some of the commonly used activation functions are **sigmoid**, **softmax**, **hyperbolic tangent**, **rectified linear unit**, etc.

3.2.2 Hidden Layers and Deep ANN:

Depending upon the volume of data, the ANN can have a more complex architecture. These complex architectures have several layers, and each layer has several perceptrons. In its simplest form, ANN has only three layers: an input layer, an output layer, and multiple

hidden layers. Instead of just taking the input features and passing them through a perceptron to produce output, the inputs from the first layer, i.e., the input layer, are passed to the next layer, which is the hidden layer, and output from each perceptron of this layers is then passed on to the next hidden layer and so on, and in the final layer, the output of the previous layers are used to predict the output. The learning of ANNs can be improved in a similar way to humans by filtering information through multiple hidden layers. The deeper the network, the higher the efficacy of the model, which is referred to as deep learning.

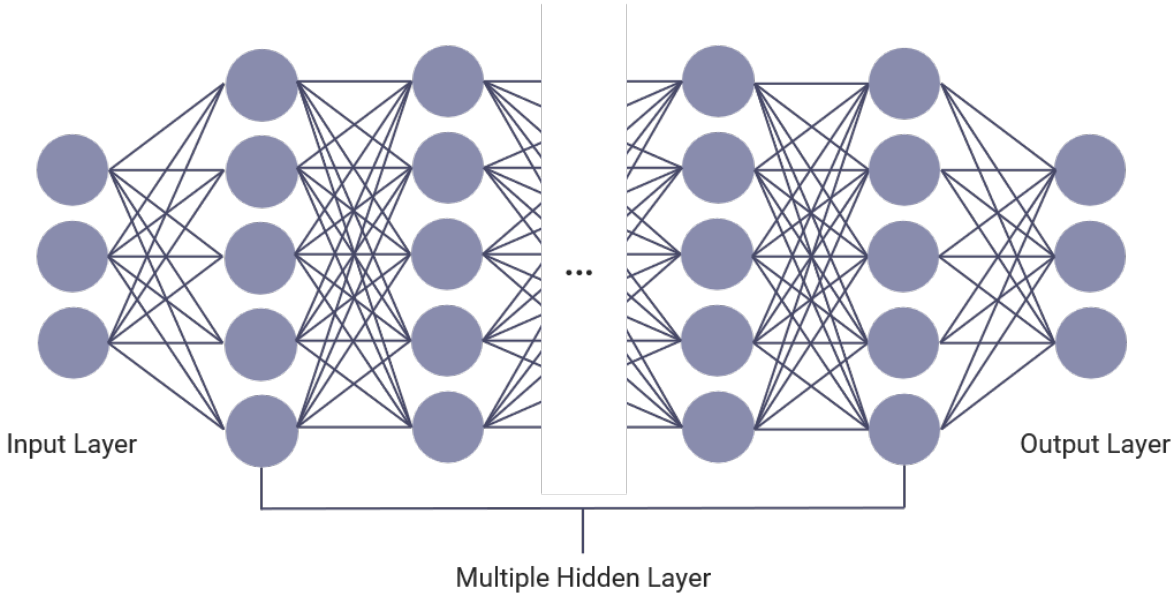

Figure 3.2: Multilayered ANN

3.2.3 Computing Gradients: Backpropagation

The cost function for a neural network usually becomes quite complicated as there are many hidden layers between the input and the output layers. Gradient descent involves the computation of a lot of partial derivatives, and these errors propagate through the network from the output layer to the input layer, which is called backward propagation or simply backpropagation. Backpropagation is an efficient algorithm for computing gradients on neural networks using the chain rule. In this, instead of calculating the gradient each time for each weight and bias, the gradients obtained in each layer are stored and used further in previous layers. The method of backpropagation helps to boost the computation of gradient descent for the multilayered ANNs.

Consider the ANN network given below(Fig 3.3) with cost function $J(\theta)$. It shows the last three layers of a simple 'L' layered ANN architecture, each having one node. θ_0^k are bias term and θ_1^k are the weights. Using the gradient descent on the output layer, i.e., L^{th} layer,

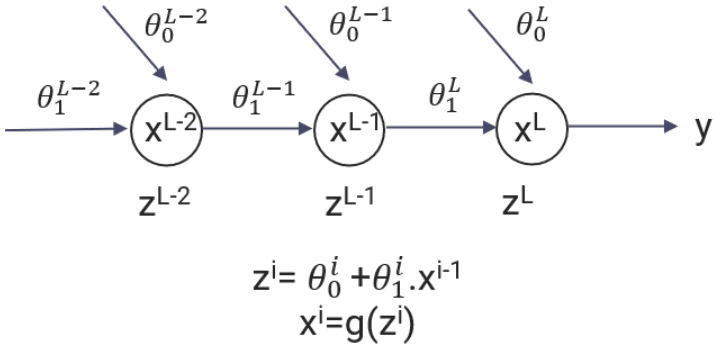

Figure 3.3: An example of ANN for computing gradients via backpropagation

the error in the weight θ_1^L is given by,

$$\frac{dJ}{d\theta_1^L} = \frac{dJ}{dx^L} * \frac{dx^L}{dz^L} * \frac{dz^L}{d\theta^L} \tag{3.2}$$

$$\implies \frac{dJ}{d\theta_1^L} = \frac{dJ}{dx^L} * g^{'}(z^L) * x^{L-1} \tag{3.3}$$

Here, $\frac{dJ}{dx^L}$ can be calculated using the cost function of the model, $g^{'}(z^L)$ is the derivative of the activation function g at point z^L. Let,

$$\varepsilon = \frac{dJ}{dx^L} * g^{'}(z^L) \tag{3.4}$$

On applying gradient descent on the second last layer, i.e., $(L-1)^{th}$, the error in the weight θ_1^{L-1} is given by,

$$\frac{dJ}{d\theta_1^{L-1}} = \frac{dJ}{dx^L} * \frac{dx^L}{dz^L} * \frac{dz^L}{dx^{L-1}} * \frac{dx^{L-1}}{dz^{L-1}} * \frac{dz^{L-1}}{d\theta_1^{L-1}} \tag{3.5}$$

$$\implies \frac{dJ}{d\theta_1^{L-1}} = \frac{dJ}{dx^L} * g^{'}(z^L) * \theta_1^L * g^{'}(z^{L-})) * x^{L-2} \tag{3.6}$$

$$\implies \frac{dJ}{d\theta_1^{L-1}} = \varepsilon * \theta_1^L * g^{'}(z^{L-1}) * x^{L-2} \tag{3.7}$$

Here we can notice that the gradient obtained for the weight in the second last layer (equations 3.4 and 3.5) appears in the second last layer's gradient update formula (equations 3.8 and 3.9). This represents the propagation of error in the backward direction, i.e., from one layer to its previous layer.

3.2.4 Regularization in ANN

In deep learning, two most important regularization techniques are **dropout and early stopping**. While training the model using ANN and using the dropout technique, some

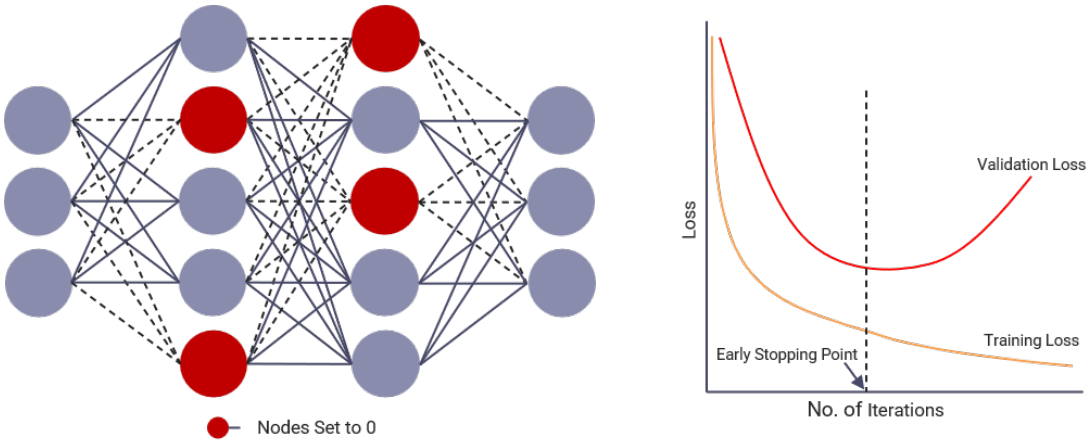

Figure 3.4: Dropout(left) and Early Stopping(right) Regularization

randomly selected activation nodes are set to zero. Generally, 50% of nodes from each layer is set to zero in each iteration during the training process. It lowers the capacity of our neural network so that they have to learn to perform better on test sets because sometimes, on training sets, it just simply cannot rely on some of those parameters, so it has to be able to be resilient to that kind of dropout. It also means that they're easier to train because, at least on every forward passive iteration, we're training only 50 of the weights and only 50 of the gradients, so that also cuts our gradient computation time down in by a factor. So because now we only have to compute half the number of neuron gradients. It basically forces the network to learn how to take different pathways to get to its answer, and it can't rely on any of the one pathways too strongly and over-fit that pathway.

The other way regularization method in ANN is by training the model for a particular number of iterations. In general, while training the model with the given training set and validation set, the loss function keeps decreasing for the training set. However, for the validation set, it decreases and then increases when the model starts over-fitting the data. Now training the model till this point where the loss for validation set starts increasing gives us a better generalization for unseen data.

3.3 Convolutional Neural Networks

As mentioned earlier, today, data are stored in numeric form as well as non-numeric forms such as images, audio, video, etc. Visual data like images and videos can be analyzed by computers, and the concerned field is known as Computer Vision. Computers recognize a digital image as a matrix with each of its entries ranging from 0 to 255, and each entry is known as pixels representing the intensity value. A convolution neural network is a subclass of neural networks that are modeled to imitate human vision. It has very great applications

in the field of Computer Vision. Examples: Object detection, Image Classification, etc. Every image data has a certain set of features that are used to differentiate them from other images, and that's exactly what the CNN model tries to learn from the image data. It uses the concept of convolution and tries to extract the local features of an image for a model as well as reduce the image data without losing any important features.

3.3.1 Basic Components of CNN:

Any CNN architecture usually consists of a combination of convolution layers and pooling layers at the beginning and one or more fully connected layers at the end, followed by a softmax classifier to classify the input into various categories.

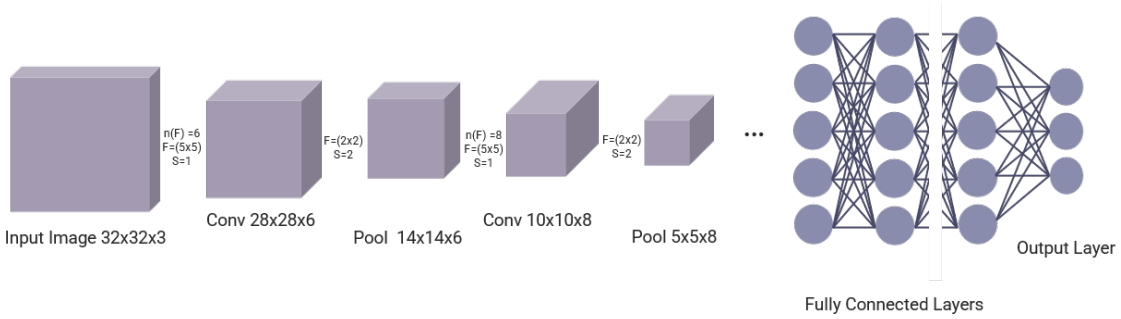

Figure 3.5: Basic CNN Architecture

Kernels and Convolution:

Kernels or filters are the way to extract features from the images. It is a matrix that is convolved with a sub-region of an input image to create a feature set of that image(Fig 3.6).

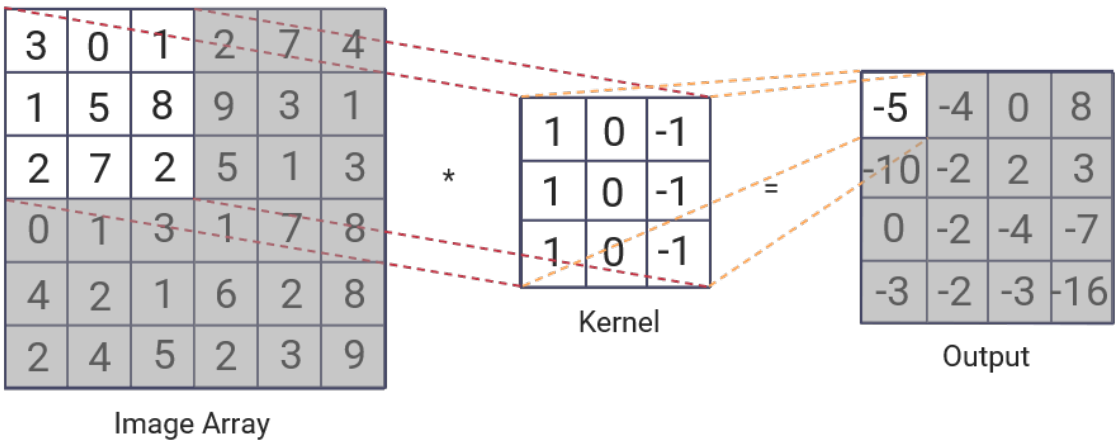

Figure 3.6: Convolution in CNN(Stride=1, Filter Size = 3x3)

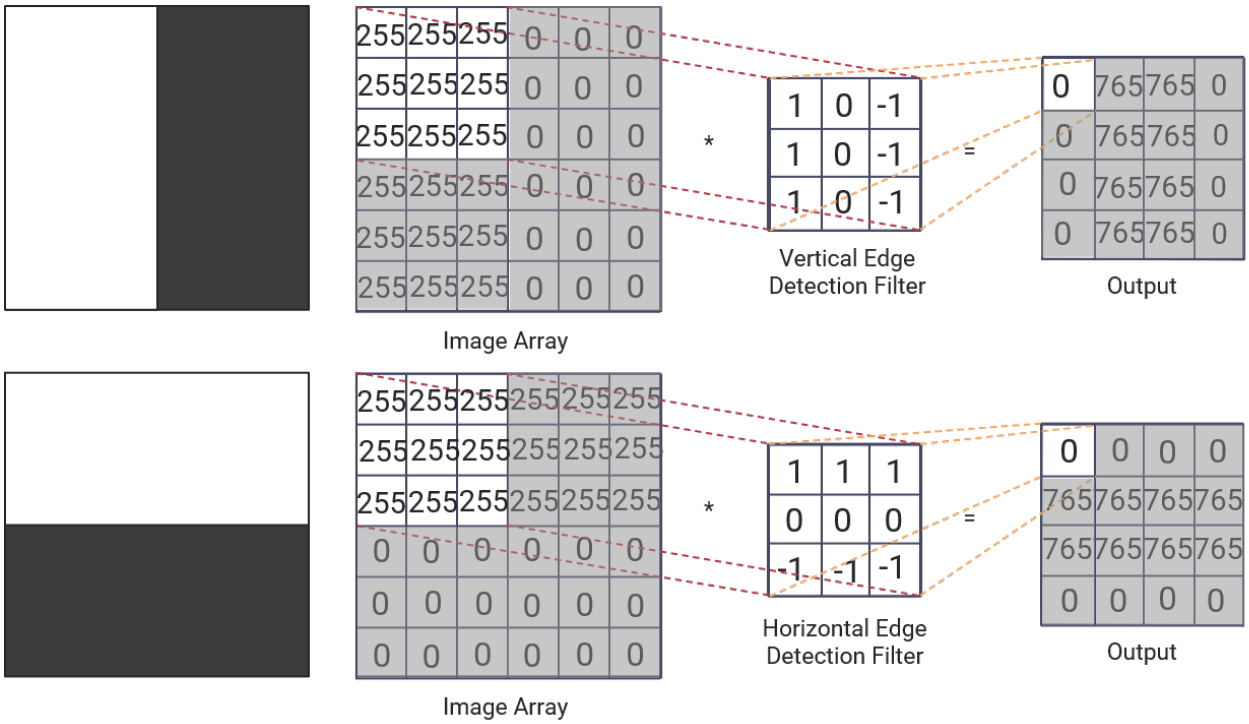

Figure 3.7: Example of Filters - Vertical and Horizontal Edge Detection Filter

Pooling:

Pooling is an operation that is used to reduce the dimensionality of our inputs and our feature maps while still preserving spatial invariants. The size of the pooling operation is always smaller than the size of the feature map. More specifically, it is almost always 2x2 pixels applied with a stride of 2 pixels. Max-pooling and Average-pooling are two types of the pooling operation. Max-pooling filter simply selects the maximum pixel value within the given filter size.

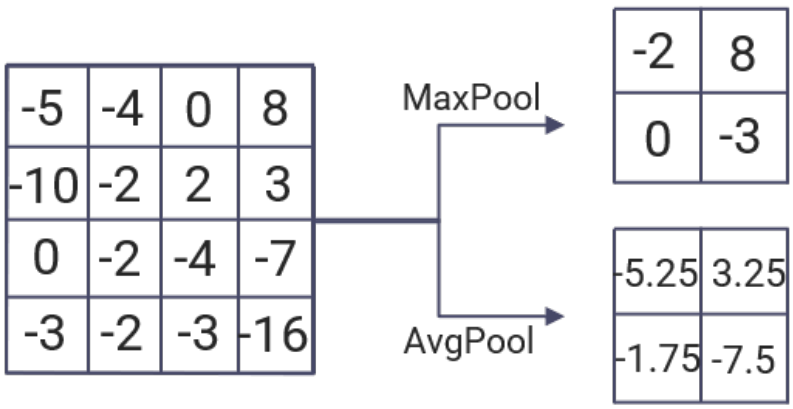

Figure 3.8: Pooling in CNN (Stride = 2, Filter Size = 2x2)

Introducing Non-Linearity:

Feature maps produced after each convolution is basically a matrix and each element of this matrix is the weighted sum obtained from previous layer. To introduce non-linearity here, ReLU(Rectified Linear Unit) is used as a activation function that replaces all the negative pixels value by zero.

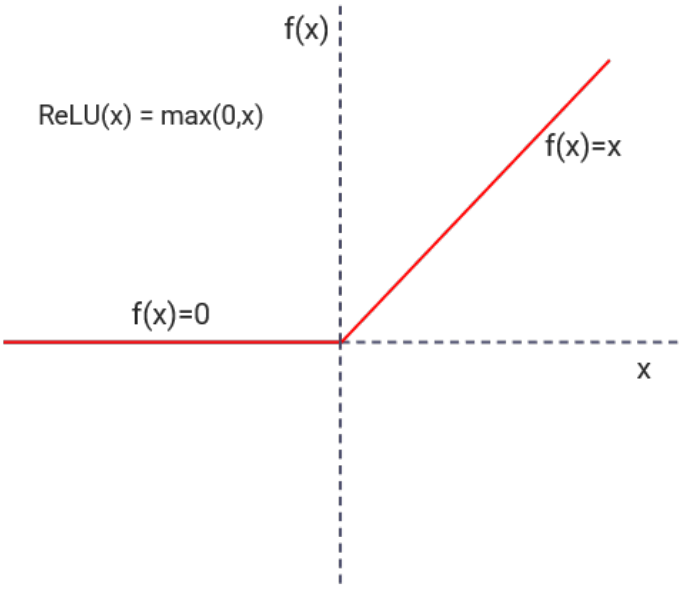

Figure 3.9: ReLU Activation Function

Padding:

While performing convolution operation, we lose information that is there on the edge of the image, and to avoid this problem, we use padding, which basically extends the area of an image on which convolution operations are performed.

0	0	0	0	0	0
0	-5	-4	0	8	0
0	-10	-2	2	3	0
0	0	-2	-4	-7	0
0	-3	-2	-3	-16	0
0	0	0	0	0	0

-5	-5	-4	0	8	8
-5	-5	-4	0	8	8
-10	-10	-2	2	3	3
0	0	-2	-4	-7	-7
-3	-3	-2	-3	-16	-16
-3	-3	-2	-3	-16	-16

Figure 3.10: Different types of Padding Operation

Output size before pooling operation:

$$OutputSize = \frac{InputSize - KernalSize}{Stride} + 1 \tag{3.8}$$

Output size after pooling operation:

$$OutputSize = \frac{InputSize - KernalSize + 2 * Padding}{Stride} + 1 \tag{3.9}$$

3.3.2 Backpropagation in CNN:

Just like ANN, CNN also uses a backpropagation algorithm to compute the gradient and tune the parameters. In ANN, we used it to tune the weight and biases, while in CNN, we use it to update weights and bias in the fully connected layer and also to tune the kernels or filters(each element of the filters are tuned) so that the model can choose appropriate filters for the given task.

In general, CNN is difficult to train on CPUs because it is a deep network that involves huge amounts of matrix multiplications and other operations, and CPUs don't have much memory bandwidth. For faster computation, one can use GPU instead of CPU for faster training and deployment. GPUs have more cores than CPUs that can be used to perform the same operation on multiple data items in parallel. Hence, a GPU can process the vast volume of data, speeding up the required tasks beyond what a CPU can handle.[Madiajagan 19] For example: In this project a CNN model was trained (discussed in Chapter 5, Part B) on ACER INSPIRE 7 - 8 GB RAM - CPU: Intel Core i5 - 9^{th} Gen - GPU: NVIDIA GEFORCE GTX 1650 -4 GB. On CPU, it took around 5 min per iteration to train 5000 - 224x224 RGB images, whereas on GPU, it only took 3 min per iteration to train 10,000 - 224 x 224 images over the same CNN model.

Machine Learning and Deep Learning have very great applications in several fields. We have used these concepts to create a model in one of our experiments(Chapter 4, Part B).

Chapter 4

Experiments, Results and Discussion

4.1 Part A: Analysis of Cricket's Acoustics

Crickets are nocturnal ectotherms that communicate using sound. Their calling activity can be limited by several environmental, physiological, and ecological factors. Acoustic communication in crickets is to attract mates, promote copulation, and aggressive interactions with rivals. The male cricket insects make sounds by stridulatory organs located on the left forewing, which has lots of teeth or comb-like structure, and when the left forewing is rubbed against the right forewing, they produce chirps.

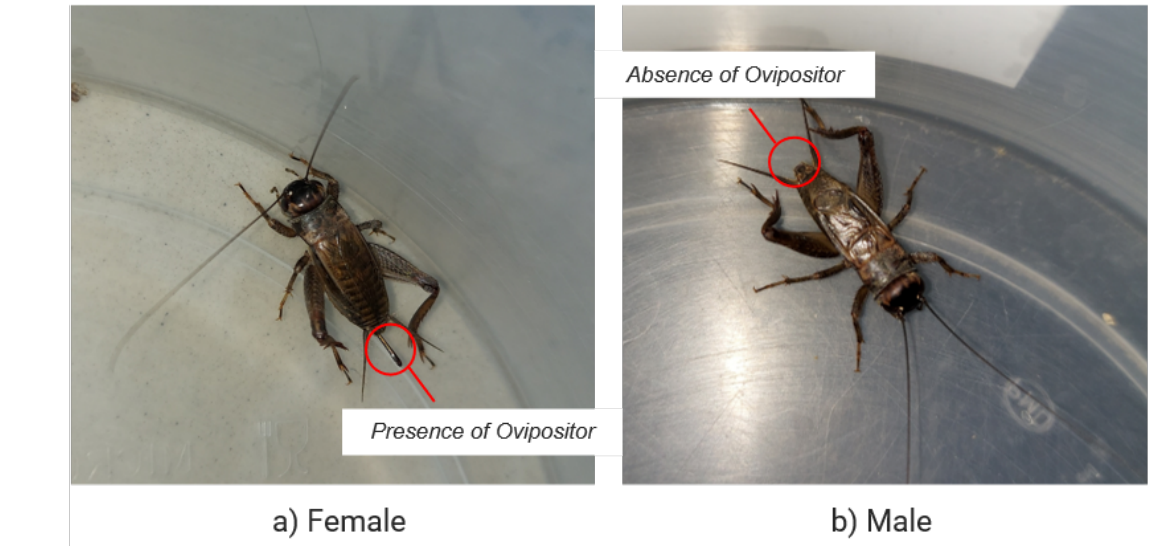

Figure 4.1: Male and Female of Acanthogryllus Asiaticus collected from IISERM

In crickets, only males sing, and a sexually receptive female, upon hearing the "calling song" of a conspecific male, moves towards the sound source. Once the male and female come into physical contact, the male switches to a distinct signal, "courtship song", which is part of a multimodal display that entices the female to mate. Males also sing "rivalry songs"

during and, more typically, after winning, aggressive encounters with rivals.[Singh 21]

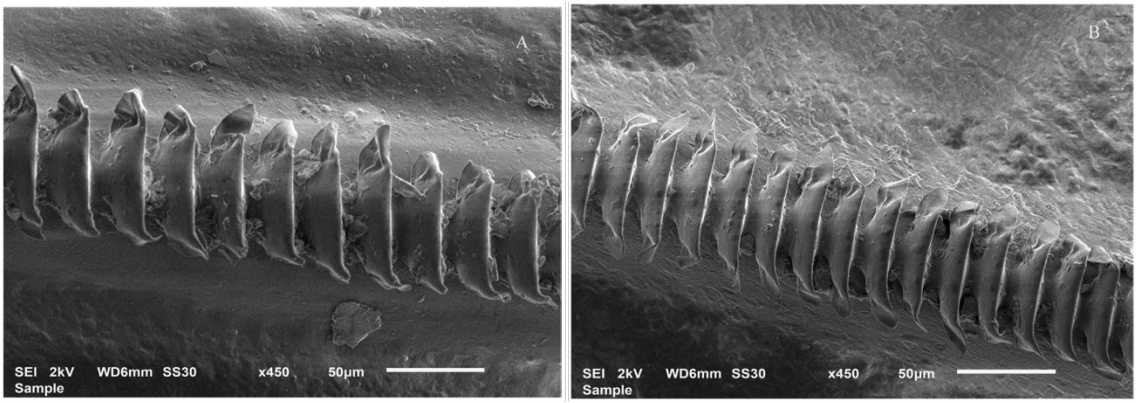

Figure 4.2: Scanning Electron Microscope images of Stridulatory Teeth present in (A) Teleogryllus mitratus, (B) Teleogryllus occipitalis [JAISWARA 21]

4.1.1 Problem Statement:

The stridulatory teeth of the crickets are almost equidistant due to which the acoustic signals produced by them on rubbing occur with a certain time difference. The sound produced due

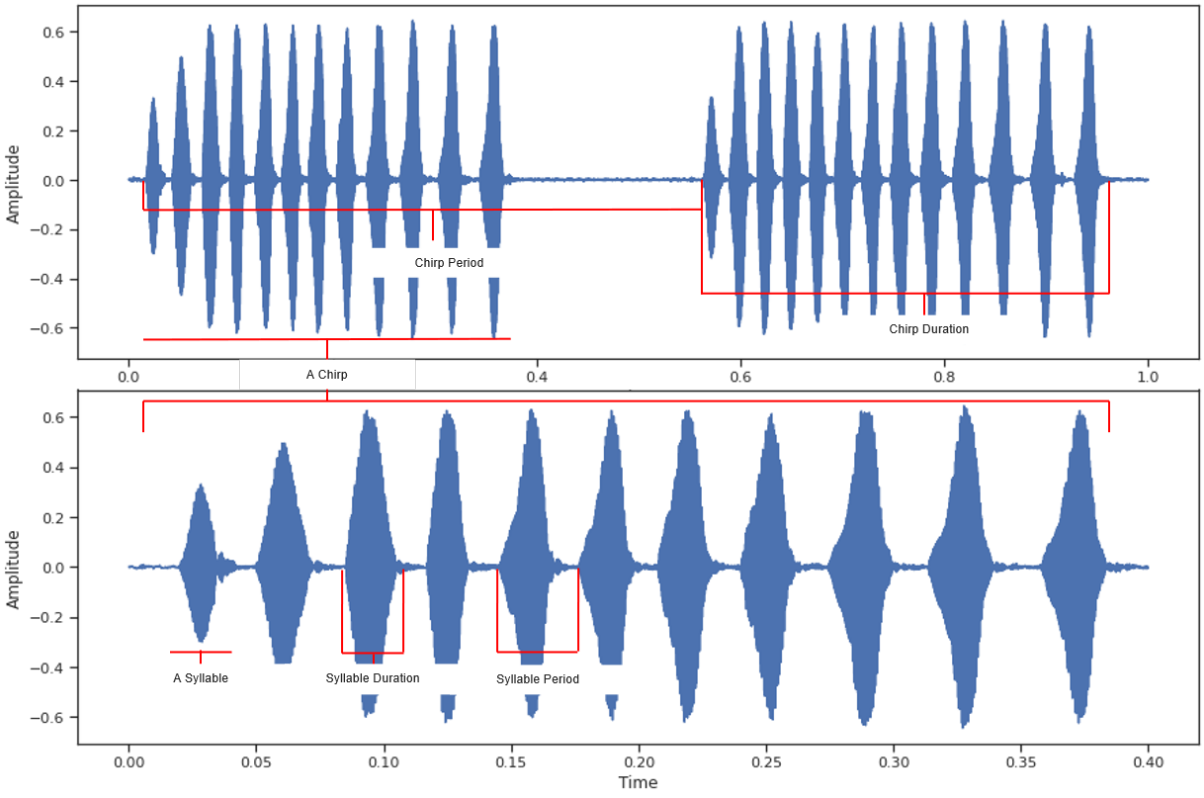

Figure 4.3: Temporal Patterns in Cricket's Call

to each tooth is referred to as a syllable, and in one go, several syllables are produced, which

are referred to as a chirp. This syllable has certain onset/offsets and the same for chirps, and once determined, these onsets/offsets can be used to determine important features for each syllable and chirps, which can be used further for research purposes. Also, some of the key features can also be used to create species classification models based on machine learning and deep learning techniques. But determining the onset/offset of each syllable and features based on them for all acoustic recordings can be a vigorous task when done manually. We can automate this method of onset/offset detection and feature extraction by using our proposed method.

4.1.2 Data Collection:

The data provided to us were the acoustic recordings of male crickets which were taken by the members of Evolutionary Biology Lab, IISER Mohali. Those recordings were taken in a sound proof setup by using Sony ICD-UX533F voice recorder with 44.1 kHz sampling rate in a 16 bit wav file at room temperature and humidity being in the range of 40-70%. The recordings were made in a box covered with anechoic foam, so that there would be no echoes that might distort the recordings, with recorder attached at the cap of it.

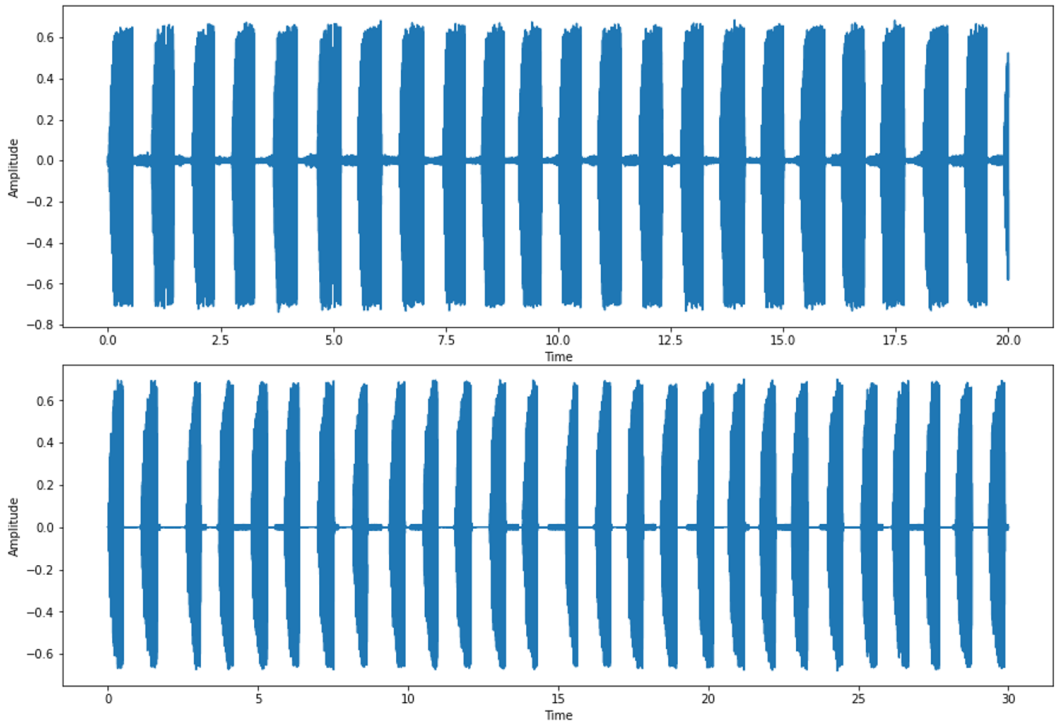

Figure 4.4: Time-Domain Representation of Cricket's Calls

4.1.3 Proposed Model and Results:

The onset and offset of a square wave-like signal can be easily calculated by several methods. One of them is by computing the point where the slope is changing its value from zero to a positive value and from a negative value to zero. It is a kind of extension to a mathematical concept known as a turning point. Mathematically, a turning point is a point at which the slope of a function changes its sign. Hence the product of the slope at a point just before and after the turning point is a negative value. In our case, we can extend this condition of a turning point, where the slope at a point just before and after the turning point is a non-positive value, given that both of them can't be zero simultaneously.

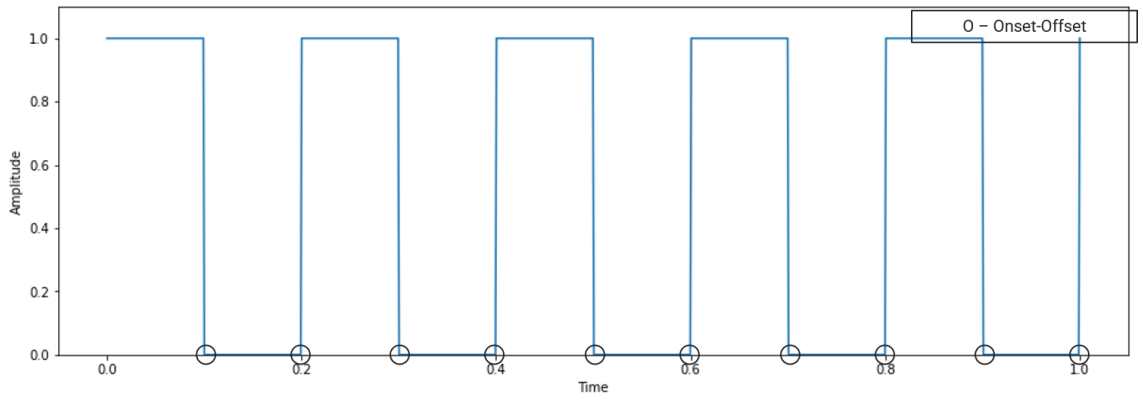

Figure 4.5: Enveloping the Signal (Syllable View)

In our proposed method, we have approximated our sound signal with a square wave like signal in such a way that the onset/offset of the square wave lies just inside the syllables and in the small neighbourhood of onset/offset of the syllables. This onset/offset of the square wave like signal helps in the determination of the actual onset/offset of the syllables, which are best approximated by the turning points in the cricket signals, which are just before/after the onset/offset of the estimated square wave like signals. Instead of using the original cricket calls, a better approach is to use the amplitude envelope of the syllables and turning points of the amplitude envelope, as it will have the same onsets/offsets for syllables, and it also ignores the turning points which have negative amplitude. This amplitude envelope for the audio signal can be obtained by using the Hilbert transform(Fig 4.6,4.7).

The required square wave-like signal can be obtained by using a suitable threshold value, below which the value of the amplitude will be set to zero, and in other cases, the value of the amplitude will be set to the threshold value itself. In the case of the higher threshold value, the generated square wave will have non-negligible differences in terms of onset/offset w.r.t. our original onset/offset. The optimal threshold value for better approximation can be calculated in terms of the mean amplitude of the envelope. On calculating the average amplitude of a cricket call, it is found to be in the order of 10^{-6} or 10^{-7} seconds

since it rises and falls symmetrically above and below the zero references. Therefore instead of using this value, the mean amplitude of the envelope signal is used, which has some significant value. Even in the case of envelope signal, since the time period for each syllable is usually in the order of 10^{-1} or 10^{-2} seconds, the region of non zero amplitude is very small, and therefore, because of this and silence region between two consecutive chirps, most of the amplitude values lie near the x-axis only in the time domain representation and the mean amplitude in this case also is as compared to the maximum amplitude.

On calculating, we observed that in most the case, this threshold could be set to the mean amplitude because of the fact that for any syllable, ideally, all possible turning points are the point of maximum amplitude and one before and after before the point of maximum amplitude which is the actual onset/offset. And since the value of this mean amplitude is very small, it can also avoid the presence of any other turning points between the point of maximum amplitude and onset/offset of the syllable. In the model, we have allowed the user to put any linear transformation of the mean amplitude of the envelope signal to tune the results further.

Once the threshold is decided, it can be used to approximate the syllables with a square wave like a signal. Further, we extracted all the stationary points of the envelope signal whose amplitude is less than our threshold value. Out of all the extracted stationary points, the one that was just before the onset of the square wave and the one just after the offset of the square wave was approximated as onset and offset for syllables of our original signal. Fig 4.7 to Fig 4.20 shows the result of each step followed and the results for the cricket's signal given in Fig 4.6. Data frames for both syllables and chirps with some of their features which are obtained on the basis of detected onset/offset, are also shown in Fig 4.21 to Fig 4.24.

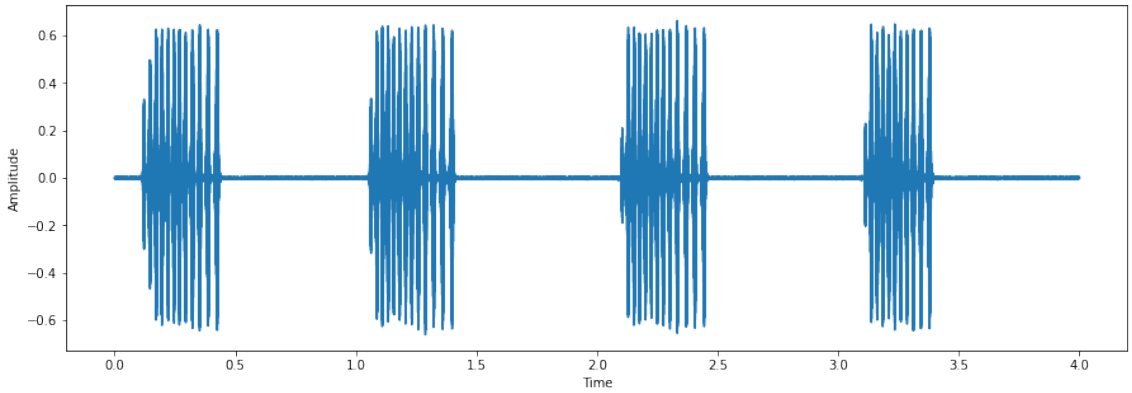

Figure 4.6: Cricket Signal

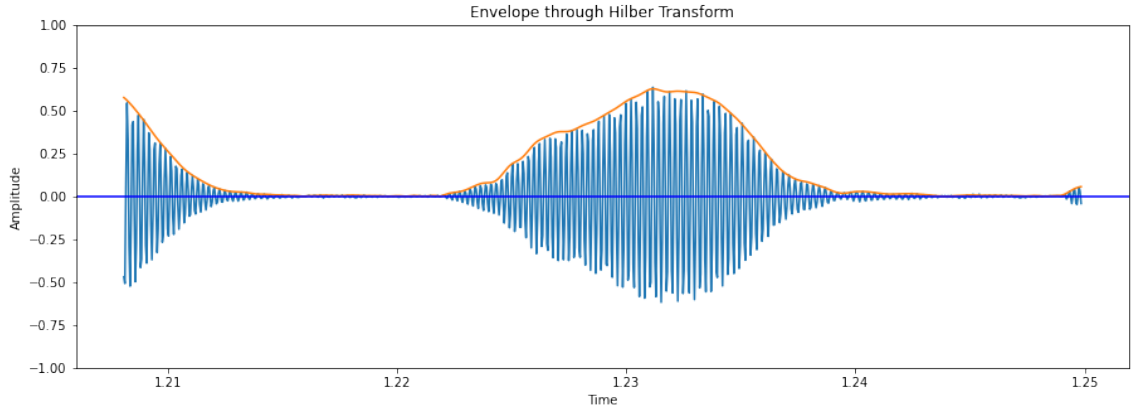

Figure 4.7: Amplitude Envelope (Syllable View)

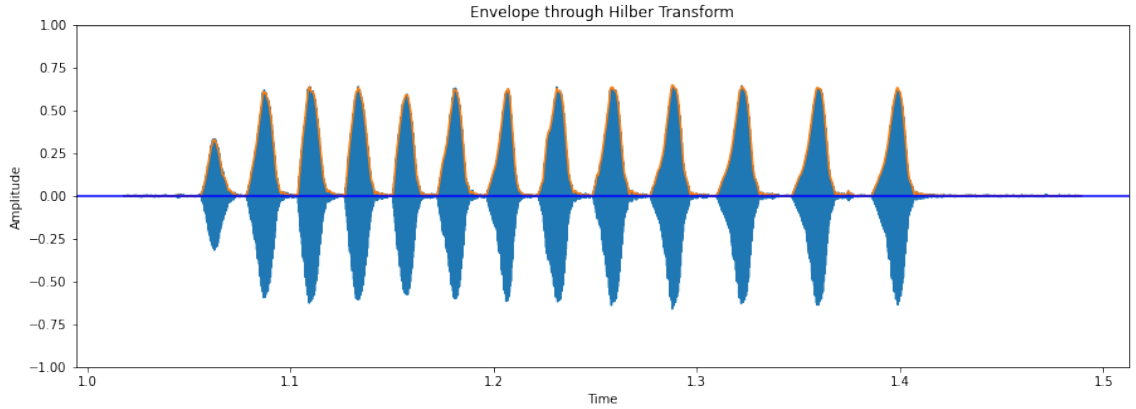

Figure 4.8: Amplitude Envelope (Chirp View)

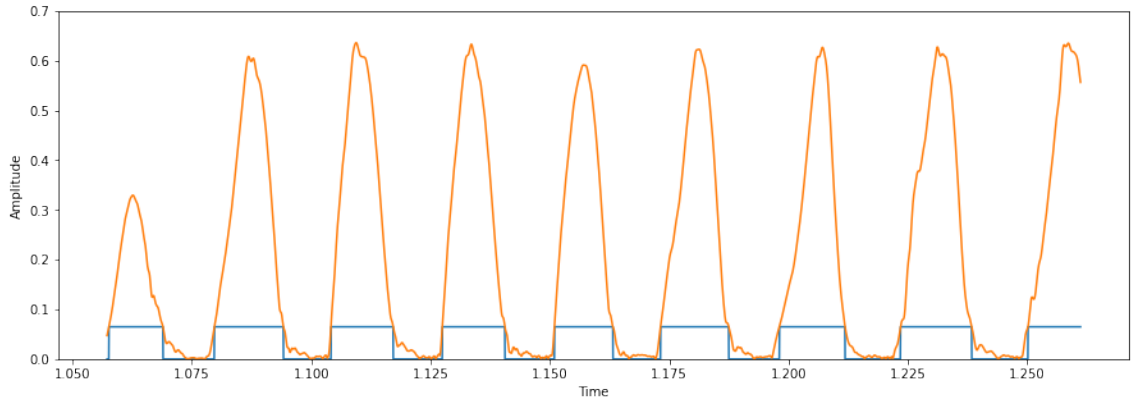

Figure 4.9: Square Wave Signal via Thresholding

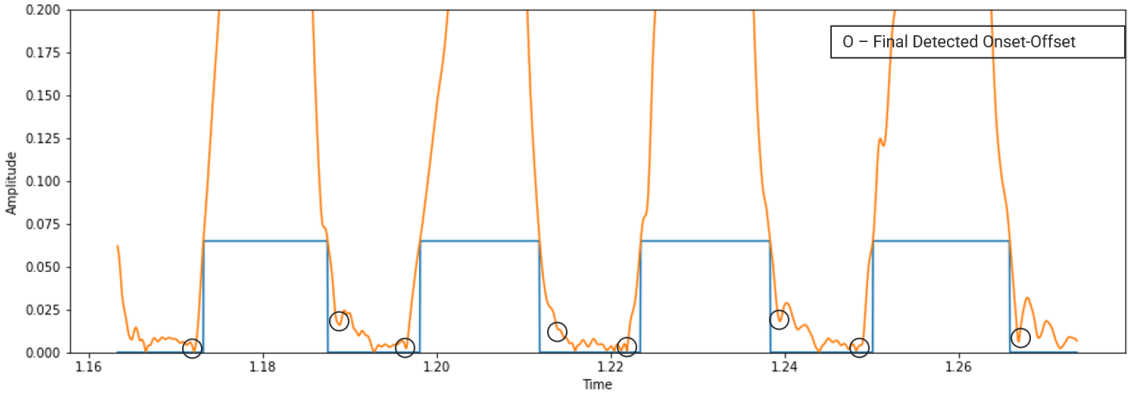

Figure 4.10: Detected Onset-Offset

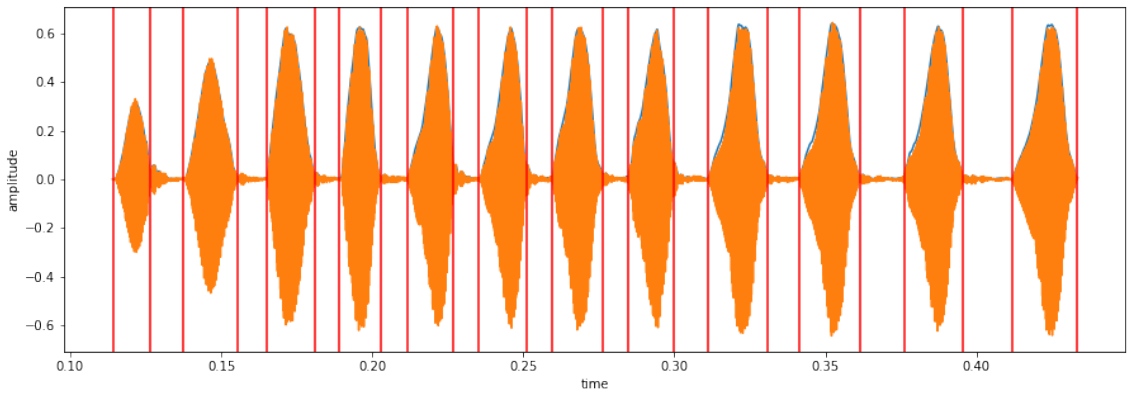

Figure 4.11: Chirp with detected onset-offset(vertical red lines)

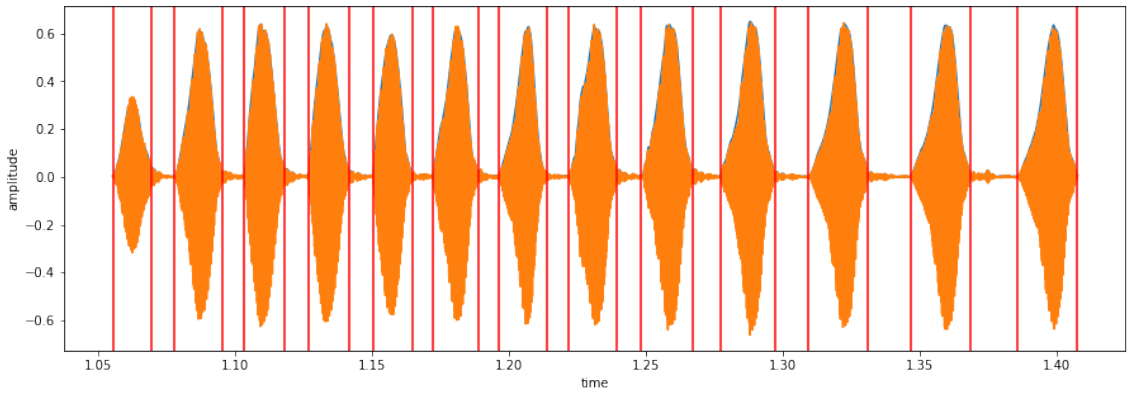

Figure 4.12: Chirp with detected onset-offset(vertical red lines)

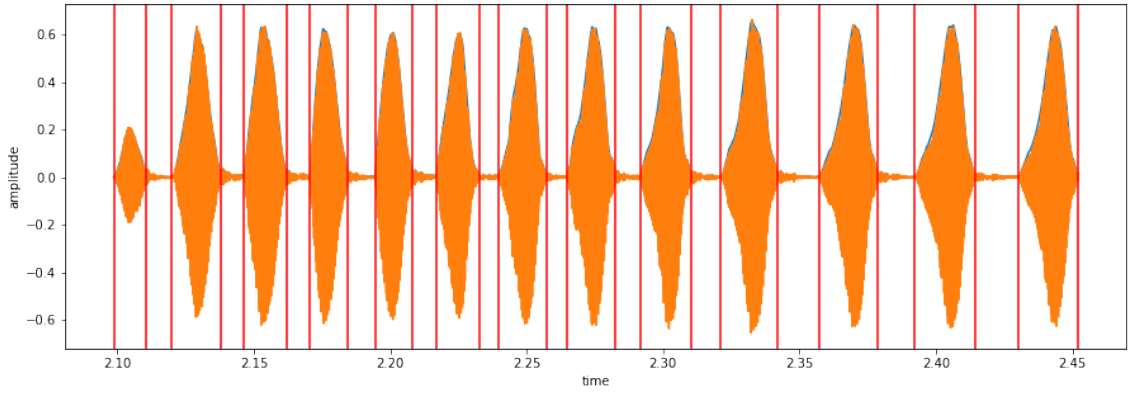

Figure 4.13: Chirp with detected onset-offset(vertical red lines)

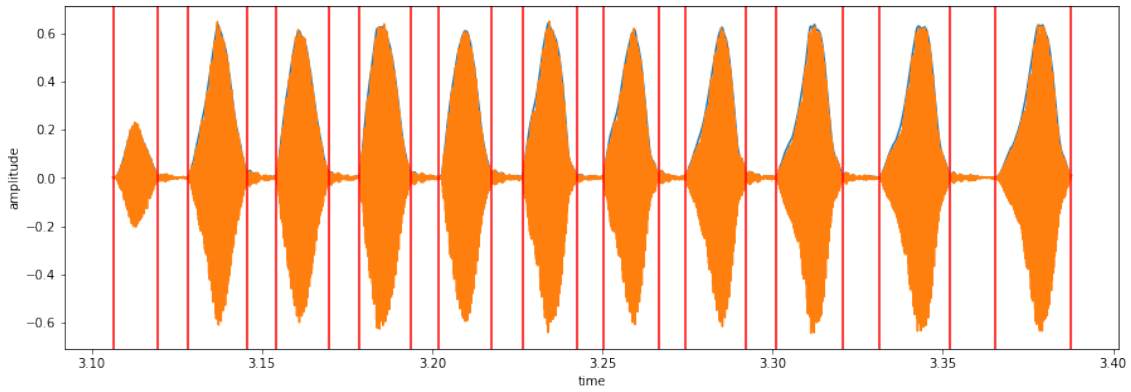

Figure 4.14: Chirp with detected onset-offset(vertical red lines)

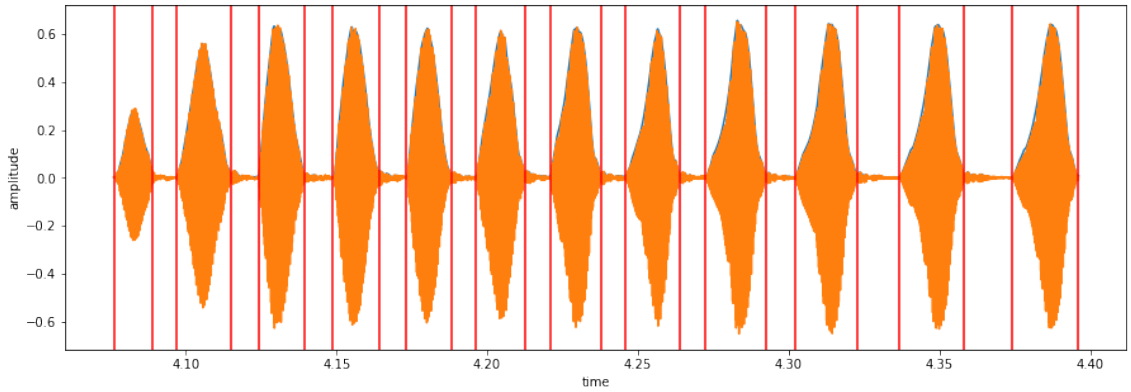

Figure 4.15: Chirp with detected onset-offset(vertical red lines)

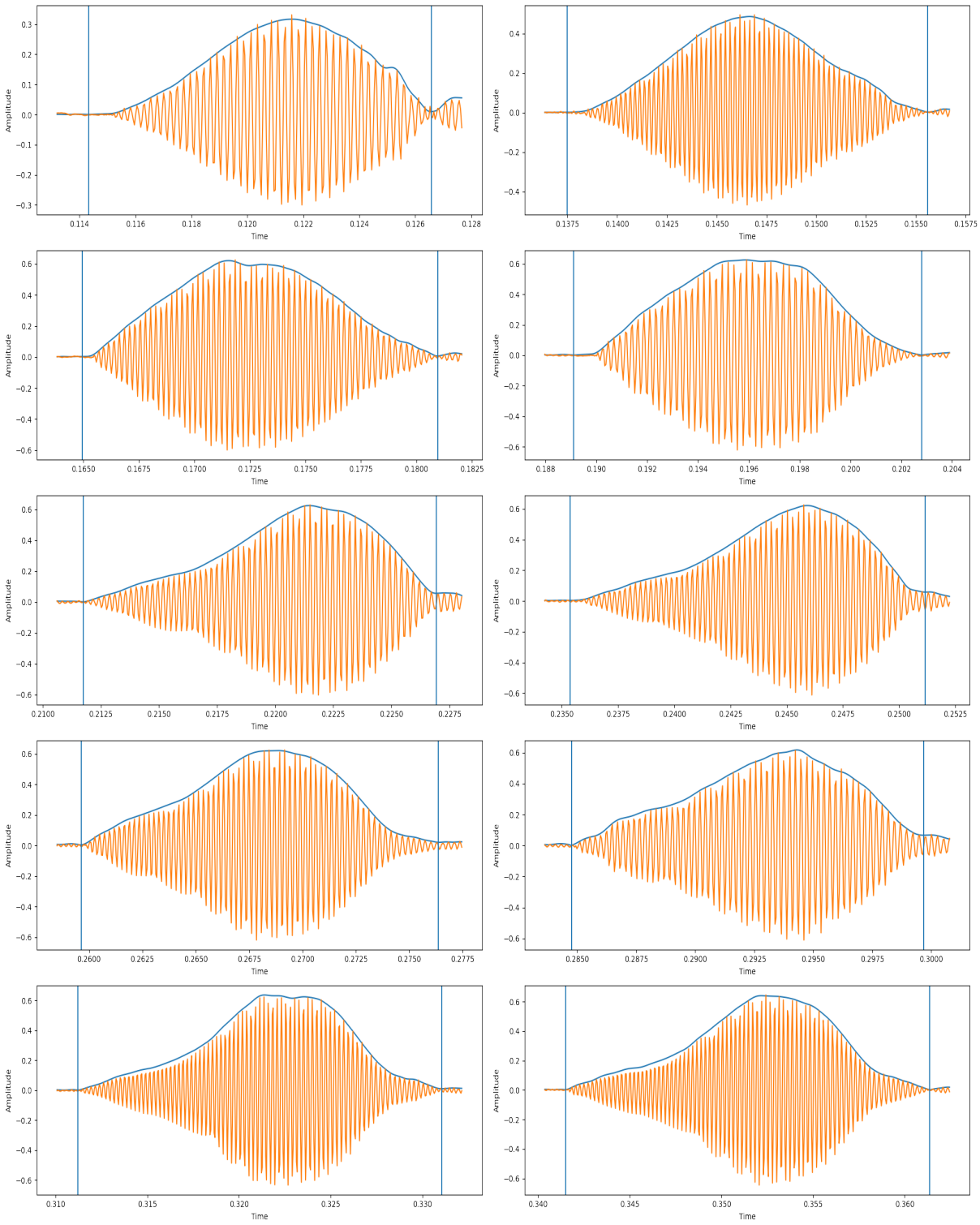

Figure 4.16: Syllable with detected onset-offset(vertical blue lines)

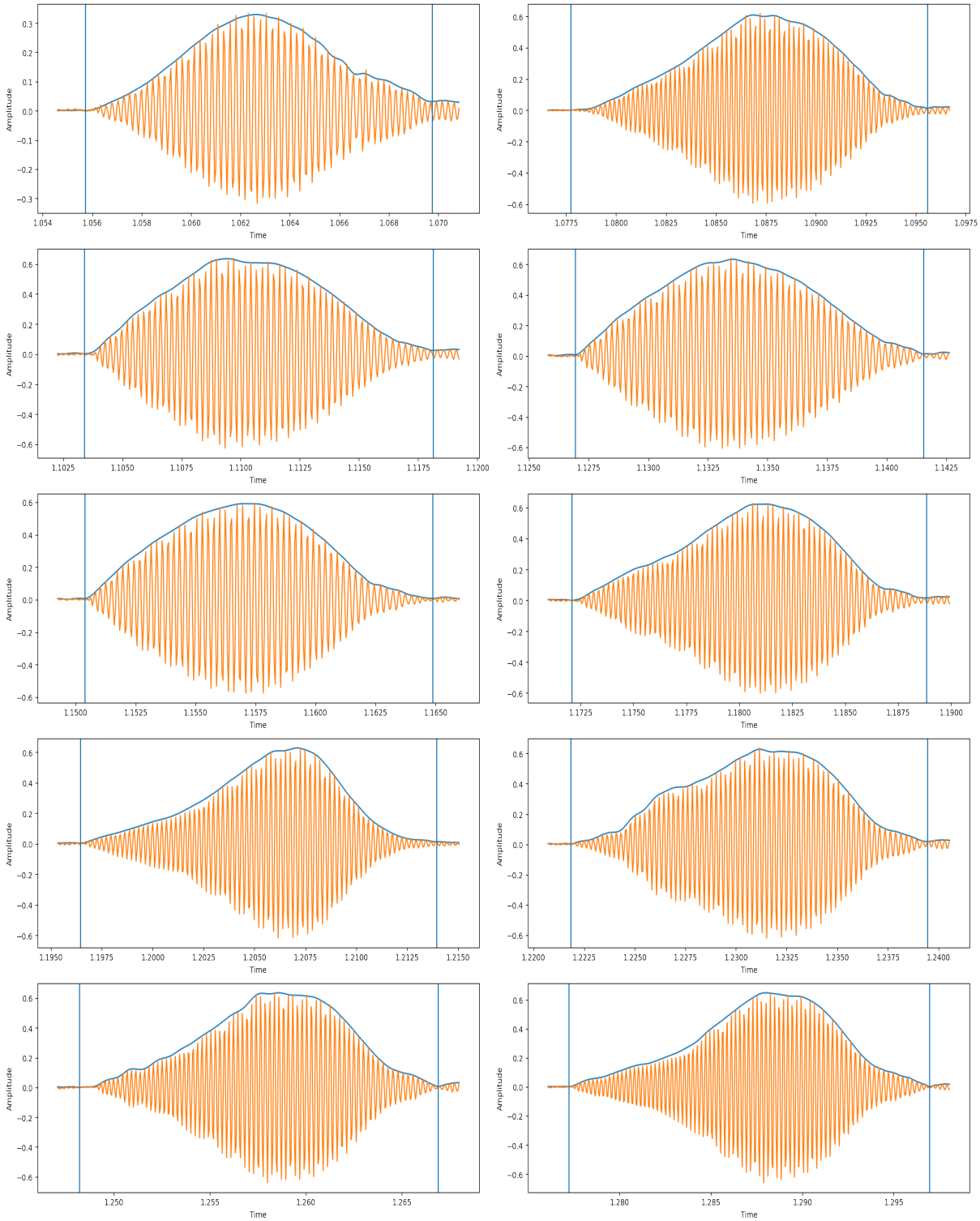

Figure 4.17: Syllable with detected onset-offset(vertical blue lines)

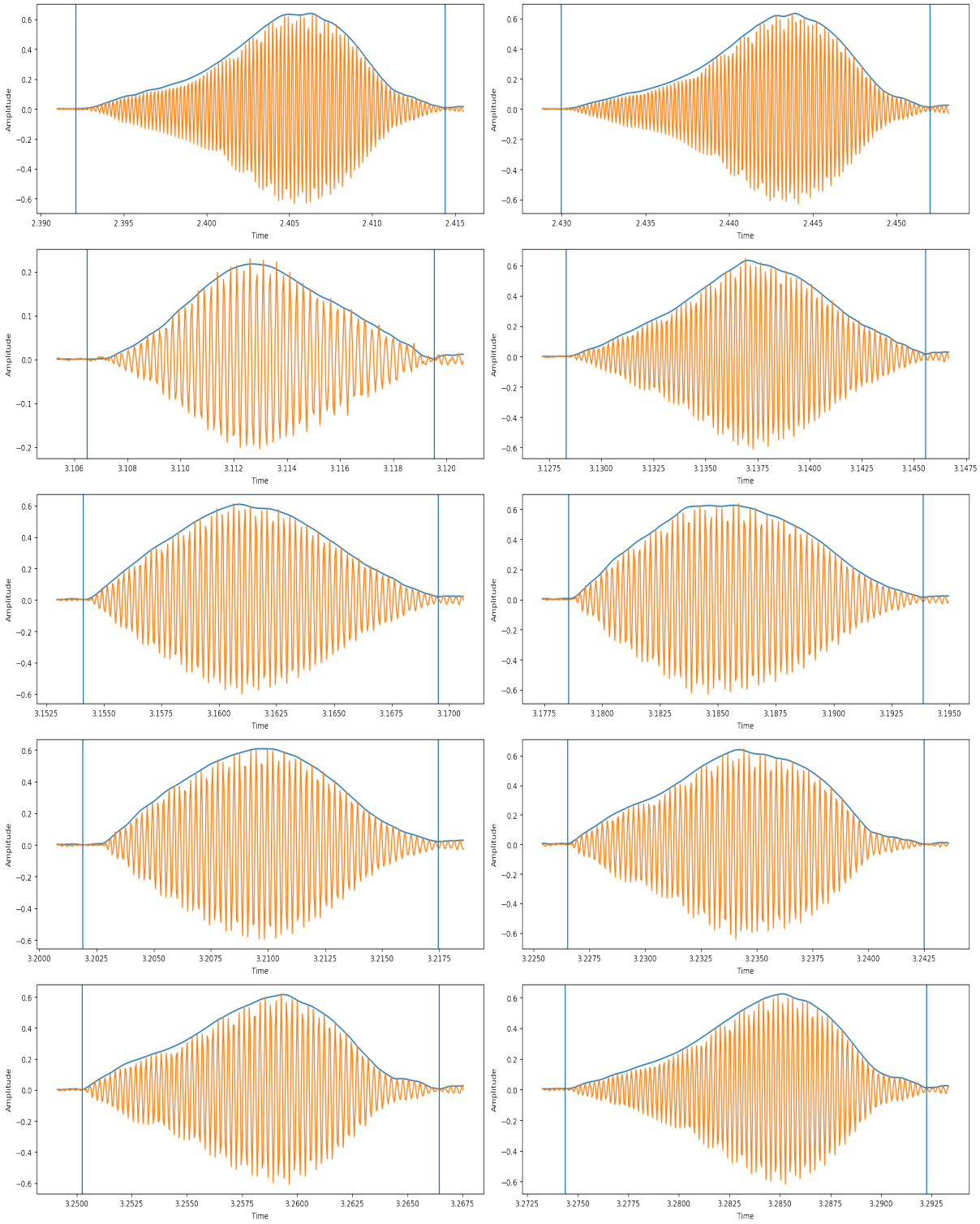

Figure 4.18: Syllable with detected onset-offset(vertical blue lines)

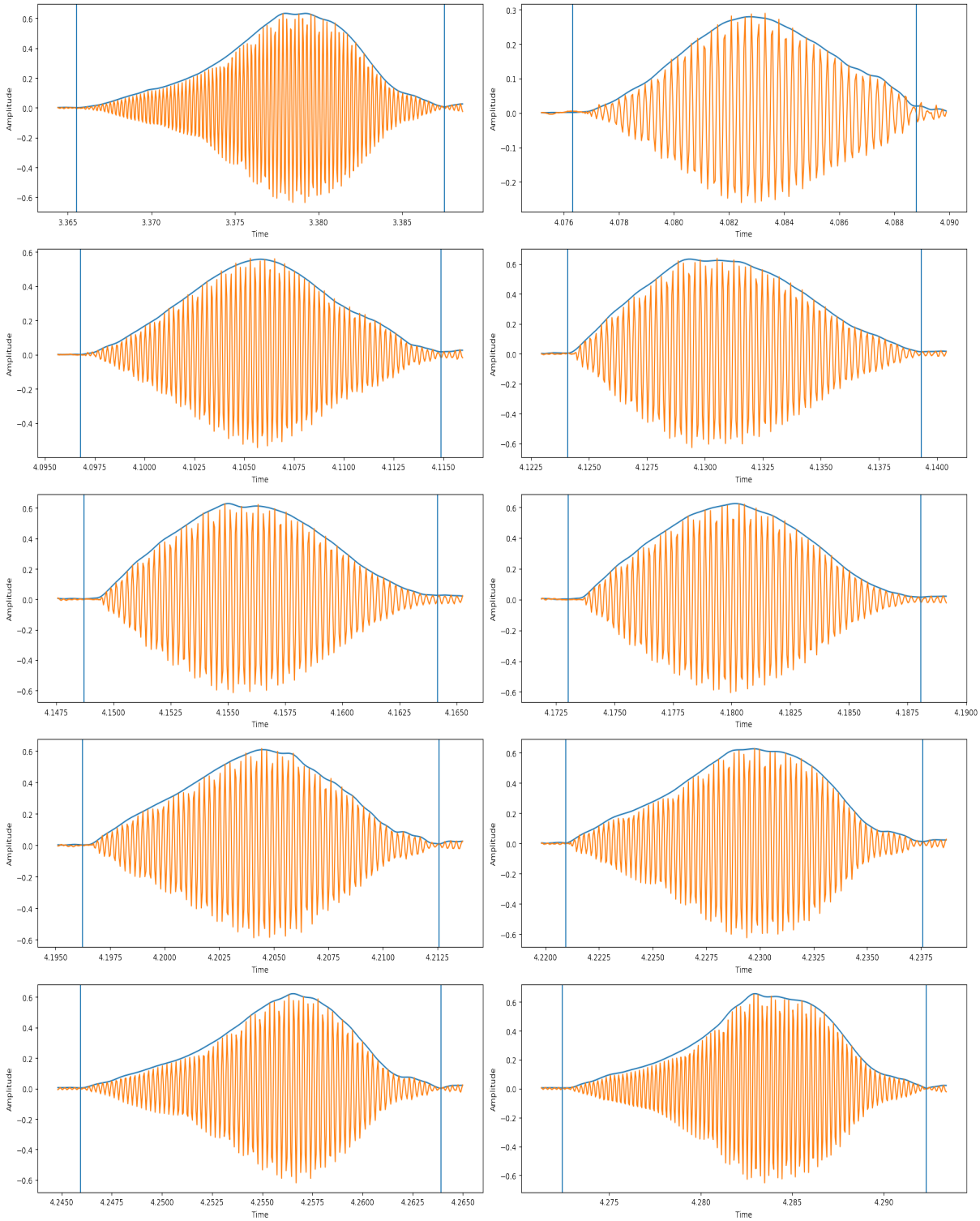

Figure 4.19: Syllable with detected onset-offset(vertical blue lines)

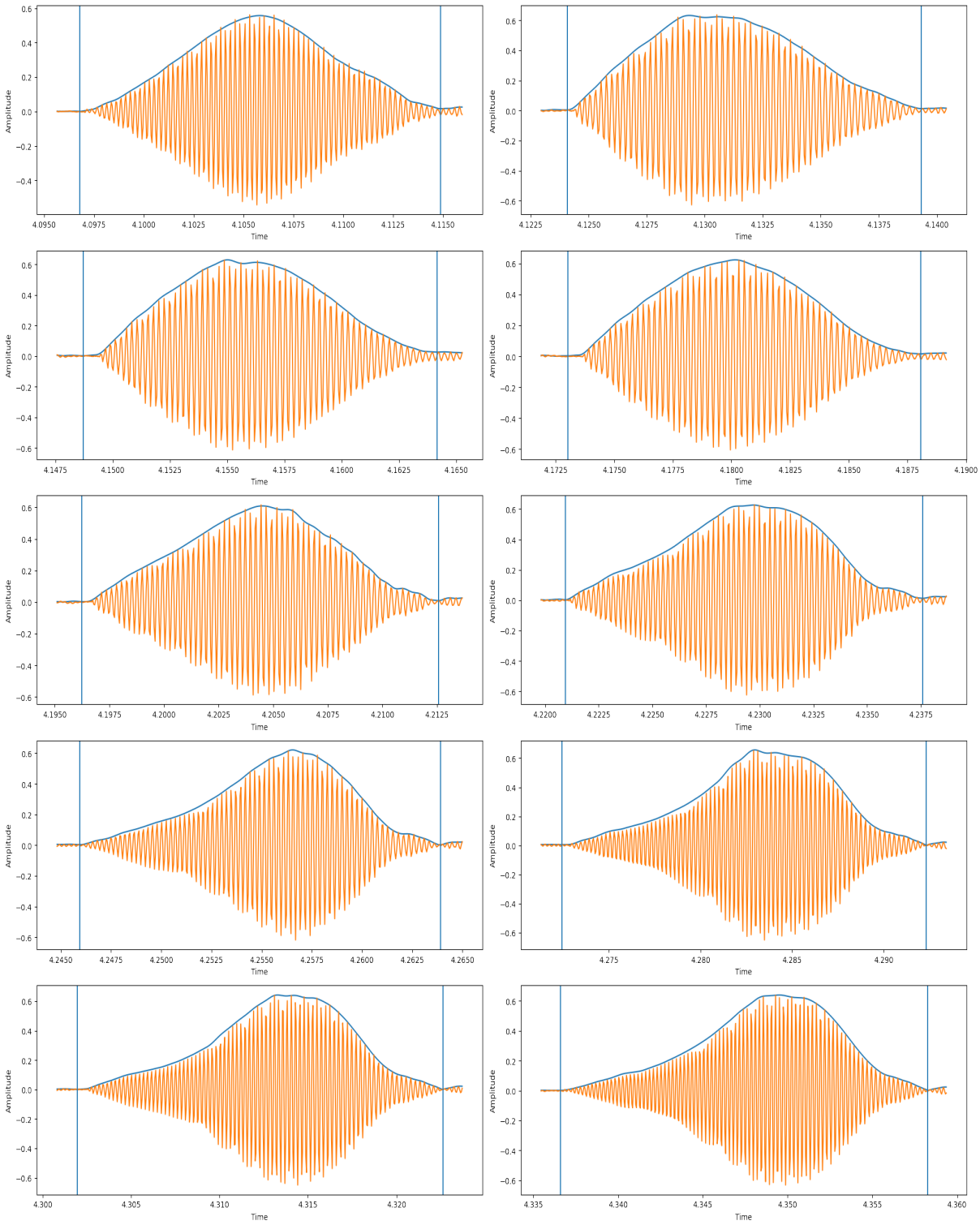

Figure 4.20: Syllable with detected onset-offset(vertical blue lines)

	Syllable Starting Point	Syllable End Point	Duration	Syllable Period (Acc. to CSP)	Syllable Period (Acc. to CEP)	Energy	Avg Amp	Max Amp	ZCR	Var	Avg. Spectral Centroid	Avg. Spectral Bandwidth
0	0.114331	0.126576	0.012245	0.023175	0.029025	0.020293	-2.180084e-04	0.499923	0.381481	0.020369	4453.487669	1447.552961
1	0.137506	0.155601	0.018095	0.027483	0.025351	0.042289	-1.382481e-04	0.748227	0.368421	0.042395	4260.790851	1102.741616
2	0.164989	0.180952	0.015964	0.024127	0.021859	0.081714	-2.161102e-04	0.944801	0.372159	0.081947	4271.137776	1017.164817
3	0.189116	0.202812	0.013696	0.022630	0.024127	0.084750	-1.234900e-04	0.944901	0.370861	0.085031	4329.069331	1023.004743
4	0.211746	0.226939	0.015193	0.023628	0.024218	0.072949	-2.275971e-04	0.951264	0.379104	0.073167	4291.283448	1003.461469
5	0.235374	0.251156	0.015782	0.024263	0.025215	0.066958	1.704273e-04	0.944568	0.379310	0.067151	4274.413663	1049.476833
6	0.259637	0.276372	0.016735	0.025125	0.025170	0.073606	-1.120349e-05	0.943483	0.371274	0.073806	4294.163099	1016.050393

Figure 4.21: Extracted Features for each Syllable

	Syllable Starting Point	Syllable End Point	Duration	Syllable Period (Acc. to CSP)	Syllable Period (Acc. to CEP)	Energy	Avg Amp	Max Amp	ZCR	Var	Avg. Spectral Centroid	Avg. Spectral Bandwidth
count	49.000000	49.000000	49.000000	45.000000	45.000000	49.000000	49.000000	49.000000	49.000000	49.000000	49.000000	49.000000
mean	1.703479	1.720959	0.017480	0.027055	0.027861	0.065624	-0.000003	0.899684	0.374764	0.065797	4306.484408	1029.424545
std	1.102745	1.102984	0.002900	0.004969	0.004934	0.017641	0.000102	0.151538	0.006664	0.017693	44.664340	152.458336
min	0.114331	0.126576	0.011610	0.020862	0.021859	0.008524	-0.000228	0.315809	0.352941	0.008555	4245.565775	880.911400
25%	1.055737	1.069705	0.015465	0.023628	0.024127	0.063692	-0.000077	0.933712	0.371134	0.063822	4282.830908	961.283608
50%	1.385624	1.407211	0.017551	0.025261	0.025442	0.067203	0.000010	0.944254	0.375676	0.067392	4294.322319	999.208532
75%	2.392109	2.414376	0.019728	0.029569	0.030295	0.073606	0.000058	0.957066	0.379310	0.073806	4311.670251	1040.590383
max	3.365533	3.387528	0.022449	0.039683	0.038821	0.088877	0.000189	1.000000	0.384454	0.089140	4453.487669	1574.197559

Figure 4.22: Some Statistics on Extracted Features of Syllables

	Chirp Starting Point	Chirp Ending Point	Duration	Chirp Period (Acc. to CSP)	Chirp Period (Acc. to CEP)	Energy	Avg Amp	Max Amp	ZCR	Var	Avg. Spectral Centroid	Avg. Spectral Bandwidth
0	0.114331	0.433333	0.319002	0.941406	0.973878	0.042311	-0.000004	0.973009	0.377595	0.042317	4321.982403	1081.234957
1	1.055737	1.407211	0.351474	1.043175	1.044807	0.044139	0.000003	0.972296	0.371613	0.044144	4309.021485	1031.350238
2	2.098912	2.452018	0.353107	1.007528	0.935510	0.042823	0.000002	1.000000	0.372977	0.042828	4305.442393	1033.048888
3	3.106440	3.387528	0.281088	NaN	NaN	0.044638	-0.000004	0.978072	0.371249	0.044645	4302.354343	1051.754323

Figure 4.23: Extracted Features for each Chirp

	Chirp Starting Point	Chirp Ending Point	Duration	Chirp Period (Acc. to CSP)	Chirp Period (Acc. to CEP)	Energy	Avg Amp	Max Amp	ZCR	Var	Avg. Spectral Centroid	Avg. Spectral Bandwidth
count	4.000000	4.000000	4.000000	3.000000	3.000000	4.000000	4.000000e+00	4.000000	4.000000	4.000000	4.000000	4.000000
mean	1.593855	1.920023	0.326168	0.997370	0.984732	0.043478	-8.287670e-07	0.980844	0.373358	0.043484	4309.700156	1049.347102
std	1.293774	1.279298	0.033910	0.051639	0.055451	0.001091	3.745686e-06	0.013027	0.002920	0.001092	8.629479	23.181522
min	0.114331	0.433333	0.281088	0.941406	0.935510	0.042311	-4.156674e-06	0.972296	0.371249	0.042317	4302.354343	1031.350238
25%	0.820385	1.163741	0.309524	0.974467	0.954694	0.042695	-4.025517e-06	0.972831	0.371522	0.042701	4304.670380	1032.624226
50%	1.577324	1.929615	0.335238	1.007528	0.973878	0.043481	-8.803379e-07	0.975540	0.372295	0.043486	4307.231939	1042.401606
75%	2.350794	2.685896	0.351882	1.025351	1.009342	0.044264	2.316412e-06	0.983554	0.374131	0.044270	4312.261715	1059.124482
max	3.106440	3.387528	0.353107	1.043175	1.044807	0.044638	2.602281e-06	1.000000	0.377595	0.044645	4321.982403	1081.234957

Figure 4.24: Some Statistics on Extracted Features of Chirps

4.2 Part B: Bird Sound Detection

4.2.1 Problem Statement:

Assessing the presence and abundance of birds is important for monitoring specific species as well as overall ecosystem health. Many birds are most readily detected by their sounds, and thus, passive acoustic monitoring is highly appropriate.[Stowell 19]

Bio-acoustic data of birds are usually recorded in fields by placing a recorder for a long period of time. Researchers record these acoustics over a period of months for analysis, and these are huge volumes of data. But sometimes, it may happen that recordings may or may not contain the bird songs at all therefore analyzing every recording and extracting the region with bird calls is a vigorous task if performed manually. We can automate this method of bird sound detection by using our proposed model based on deep learning.

4.2.2 Data Collection:

The data provided to us were the acoustic recordings from fields considering the presence of birds in the area. The recordings were taken by the members of Evolutionary Biology Lab, IISER Mohali. The recordings were done in various locations in IISER Mohali by using AudioMoth v1.2.0 and were used with Audiomoth IPx7 case for protection during data collection. Recordings were taken by deploying the audiomoth on trees(Height - 5m approx). The configurations were done using Audiomoth Configuration App (Sample rate- 48kHz, Gain- Medium, Recording duration- 55s, Sleep duration- 5s) No filtering of frequency or amplitude threshold was enabled. So frequencies between 0-24kHz were recorded and were automatically converted to WAV file format.

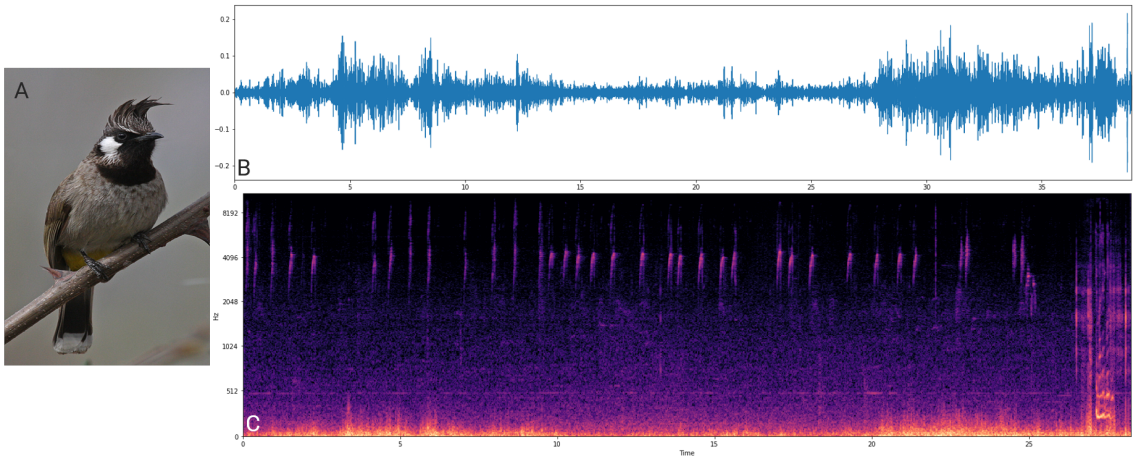

Figure 4.25: A) Himalayan Bulbul B) Time Domain representation of Field Recording of Bulbul C) Mel-Spectrogram of Field Recording of Bulbul

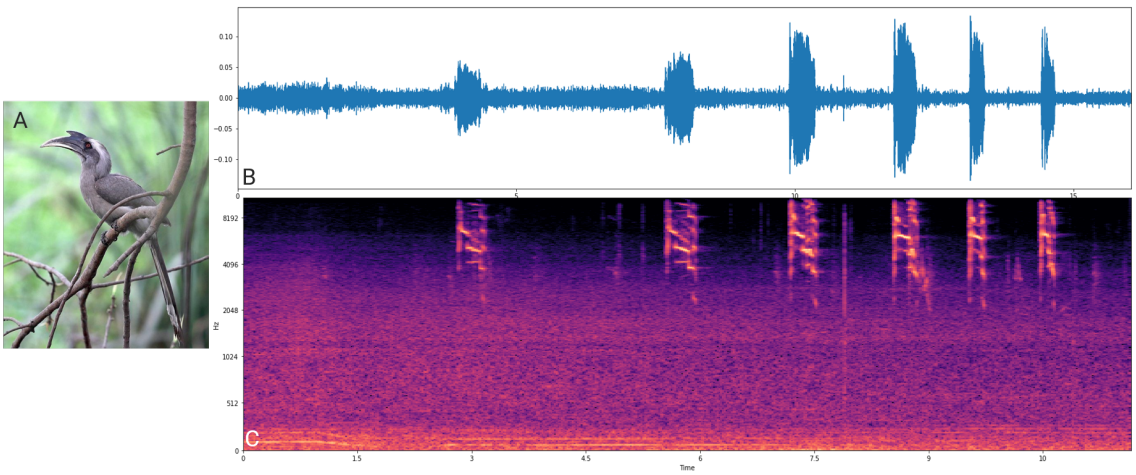

Figure 4.26: A) Indian Gray Hornbill B) Time Domain representation of Field Recording of Hornbill C) Mel-Spectrogram of Field Recording of Hornbilll

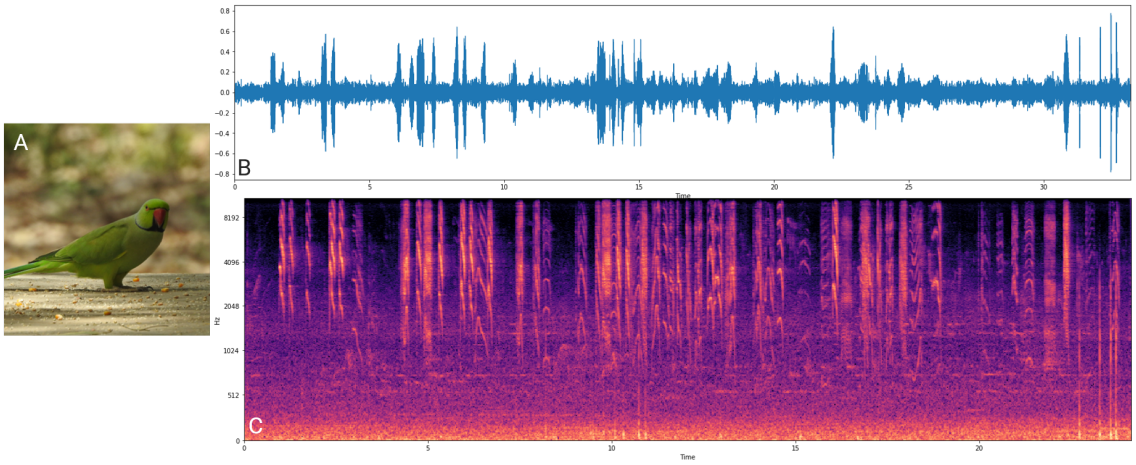

Figure 4.27: A) Rose Ringed Parakeet B) Time Domain representation of Field Recording of Parakeet C) Mel-Spectrogram of Field Recording of Parakeet

4.2.3 Proposed Model

Mel-spectrogram represents the frequency of a signal as it varies with time. These spectrograms are a unique representation of most of the sound signals from our surroundings. Human speech, dog barks, bird calls, train whistle, crickets chirps, etc. All these sounds lie in different frequency regions and follow a unique frequency-time representation. We can extract these mel-spectrograms for a small window of a field recording, and each of that mel-spectrogram can be mapped to a different class to which they belong. For example, a mel-spectrogram of a region that is similar to the mel-spectrogram of birds will be classified as a bird call. Using the mel-spectrogram, we can classify the different classes of sounds. We can use the CNN-based model to classify these spectrograms. This CNN model makes a prediction on each small time window of a field recording by obtaining the spectrograms

of all those windows. Hence we have reduced our problem of audio classification to image classification.

Data Preprocessing and Feature Extraction:

From each 55s of field recordings, 2-sec audio was trimmed out manually using Audacity on Windows, and each was labeled with 1 if there is a bird call present and 0 if there are no bird calls or calls other than of birds. For each labeled recording, mel-spectrogram was generated using Python and stored with previous respective labels. A total of 14,000 mel-spectrogram were generated, 7000 for bird calls and 7000 for non-bird, which is around 7.7 hours of recording. Out of 14,000, 10,000(5.5 hours) mel-spectrograms were used for training, 2000(1.1 hours) for validation, and 2000(1.1 hours) for testing.

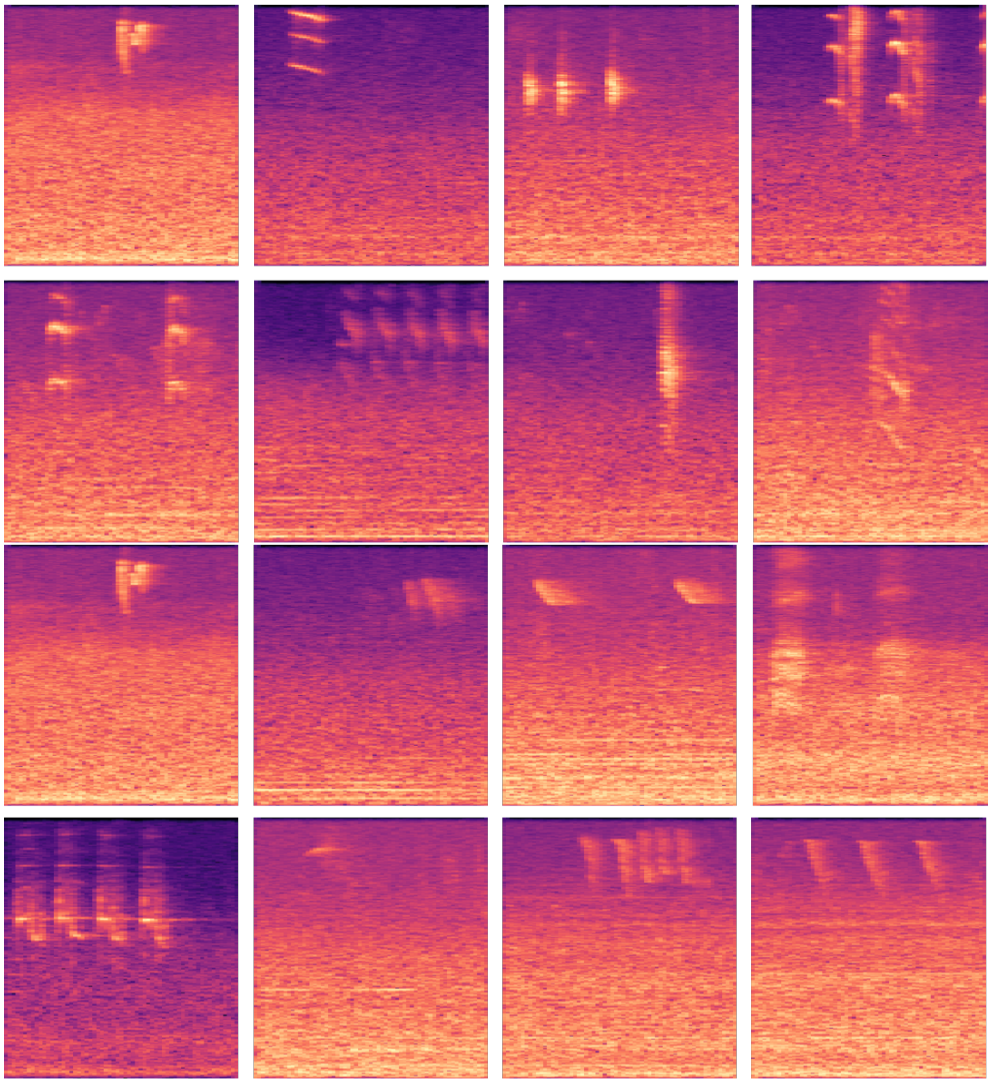

Figure 4.28: Training Data: Spectrogram of 2 sec audio clip of Bird Calls.

Based on the extracted feature, a CNN model was created, which has a total of 27,823,938 trainable parameters according to the architecture shown in Fig 4.29:

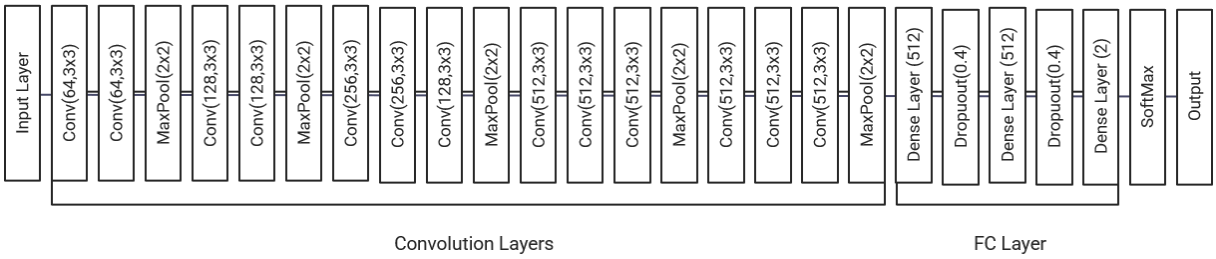

Figure 4.29: CNN Architecture

4.2.4 Results:

The model was trained on 10,000 labeled images along with a validation set of 2000 images for 70 epochs and was tested on 2000 images (1000 for each category). Since it is an classification model, cross entropy loss function was used for training which was optimized through stochastic gradient descent on the batch size of 32 training data points. It took around 4 hours on GPU to complete the training process. The overall **training accuracy** was **93.7%** and **validation accuracy** was **94.6%**.

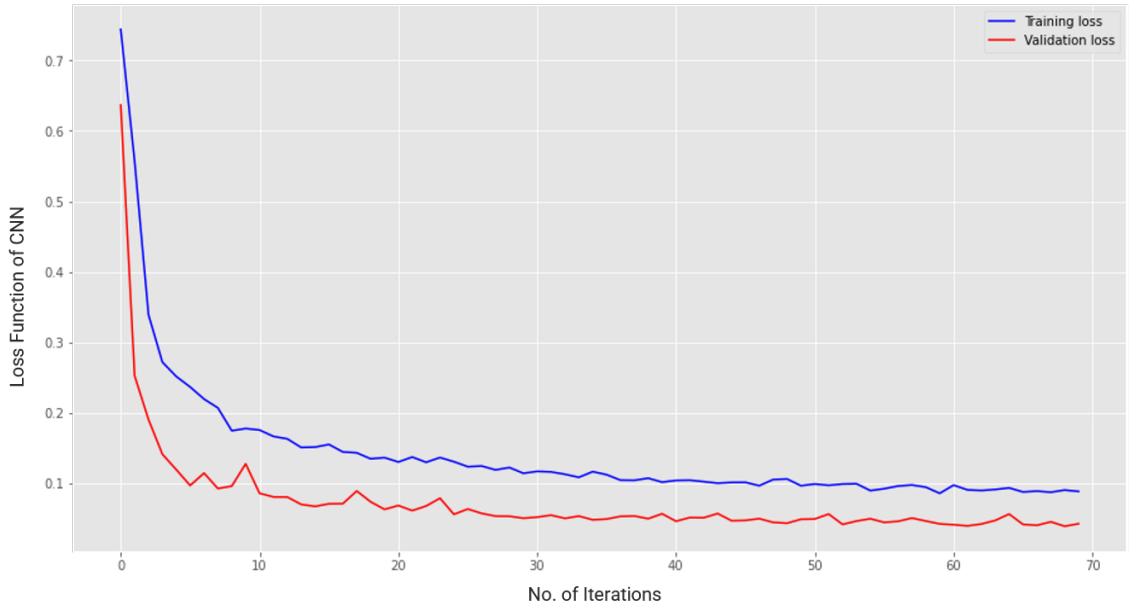

Figure 4.30: Value of Loss function at each iteration

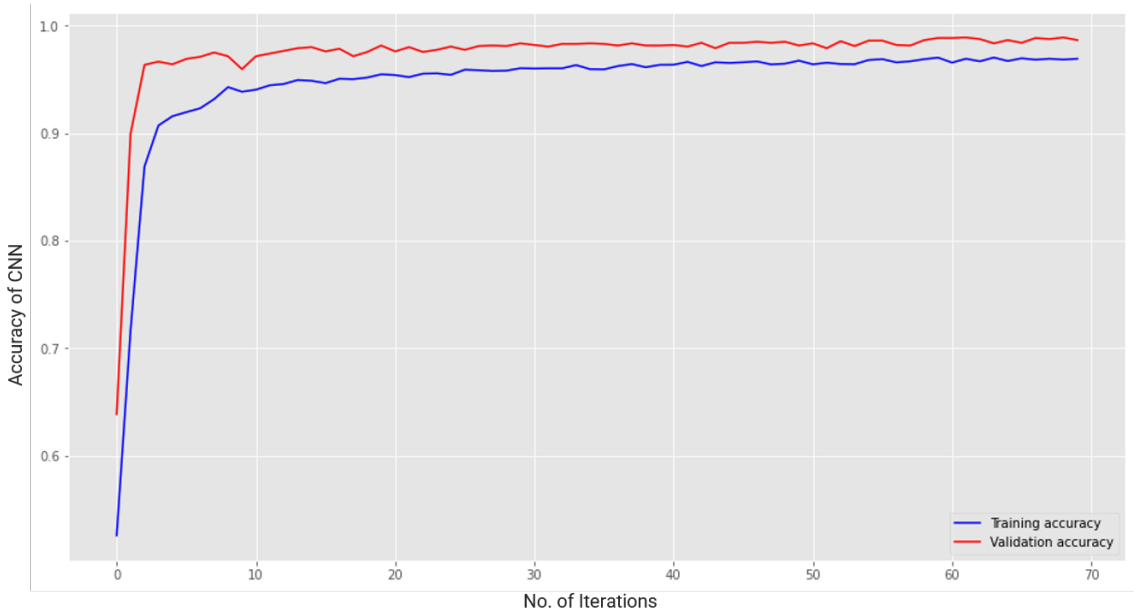

Figure 4.31: Accuracy of proposed model at each iteration

After training, the model was tested on 2000 labeled mel-spectrogram images, and the confusion matrix was calculated, which is used for evaluating the performance of a classification model. The confusion matrix compares the actual target values with those predicted by the model. From the confusion matrix, we can also calculate the model's accuracy, precision, false-positive rate, etc.

Predicted Label	Actual Label	
	Bird	Other
Bird	980 TP	11 FP
Other	20 FN	989 TN

Figure 4.32: Confusion Matrix

- A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class. In our case value of true positives and true negatives are 980 and 989 respectively.

- A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class. In our case value of false positives and false negatives are 11 and 20 respectively.

- Accuracy is the most intuitive performance measure that the model correctly classified over the total number of records.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = 0.9845 \tag{4.1}$$

- Precision talks about, out of those predicted positive, how many of them are actual positive. It measures the model's accuracy in classifying a sample as true positive.

$$Precision = \frac{TP}{TP + FP} = 0.9889 \tag{4.2}$$

- False Positive Rate: It's the measure that a positive result will be given when the true value is negative.

$$FPR = \frac{FP}{TN + FP} = 0.0110 \tag{4.3}$$

For false negatives, the most common observation was that some data points contained very faint bird sound, often needing multiple listens to be sure it was present. Also, as we were trying to classify between bird sound and non-bird sound, the training data for non-bird sound includes silence, train whistle, car/bike horn, etc. But out of all, data for silence region was in the majority then all the other non-bird songs hence training the model is slightly compromised for those which are in minor, and that might be the most probable reason of misclassification and hence false positives, were observed.

Appendix A

List of Python Libraries and Functions

A.1 Libraries

```
import cv2
import math
import numpy
import scipy
import pandas
import librosa
import tensorflow
import matplotlib.pyplot
from statistics import mode
from IPython.display import Audio
from scipy.fftpack import fftfreq
from scipy.signal import hilbert
from scipy.fft import fft, fftfreq, ifft
from scipy.signal import medfilt
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.models import Model
from keras.layers import Flatten
from keras.layers import Dropout
from tensorflow.keras.optimizers import SGD, Adam, RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from tensorflow.keras.models import load_model
from sklearn.model_selection import train_test_split
from PIL import Image, ImageOps
```

Figure A.1: Libraries used in our Model

A.2 Functions

Here, **list** represents, list of extracted syllable or chirps.

```
def energy(list):
  l=list
  en=[]
  for i in range(len(l)):
    s=np.sum([x*x for x in l[i]])
    en.append(s/len(l[i]))
  return en
#avgamplitude
def avamp(list):
  l=list
  en=[]
  for i in range(len(l)):
    s=np.mean(l[i])
    en.append(s)
  return en
#variance
def var(list):
  l=list
  en=[]
  for i in range(len(l)):
    mean=np.mean(l[i])
    s=np.sum([((x - mean)**2/(len(l[i])-1)) for x in l[i]])
    en.append(s)
  return en
```

Figure A.2: Function for computing Average Energy, Average Amplitude and Signal Variance(top to bottom)

```
#zcr
def zcr(list):
  l=list
  en=[]
  for i in range(len(l)):
      s=0
      for j in range(len(l[i])-1):
        if l[i][j]*l[i][j+1]<0:
          s=s+1
      en.append(s/len(l[i]))
  return en

#spectralcentroid
def spcen(list):
  l=list
  en=[]
  for i in range(len(l)):
    spectral_centroids = li.feature.spectral_centroid(np.array(list[i]), sr=sr)[0]
    en.append(np.mean(spectral_centroids))
  return en

#spectralbandwidth
def spbd(list):
  l=list
  en=[]

  for i in range(len(l)):
    spectral_bd = li.feature.spectral_bandwidth(np.array(list[i]))[0]
    en.append(np.mean(spectral_bd))
  return en
```

Figure A.3: Function for computing ZCR, Spectral Centroid and Spectral Bandwidth(top to bottom)

```python
def cnnmodel():
    model = Sequential()
    model.add(Conv2D(input_shape=(),filters=,kernel_size=(),padding="same", activation="relu"))
    model.add(Conv2D(filters=,kernel_size=(),padding="same", activation="relu"))
    model.add(MaxPooling2D(pool_size=(,),strides=(,)))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(MaxPooling2D(pool_size=(,),strides=(,)))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(MaxPooling2D(pool_size=(,),strides=(,)))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(MaxPooling2D(pool_size=(,),strides=(,)))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(Conv2D(filters=, kernel_size=(,), padding="same", activation="relu"))
    model.add(MaxPooling2D(pool_size=(,),strides=(,)))
    model.add(Flatten())
    model.add(Dense(, activation='relu'))
    model.add(Dropout())
    model.add(Dense(, activation='relu'))
    model.add(Dropout())
    model.add(Dense(, activation='softmax'))
    optimzer =
    model.compile(optimizer=optimzer, loss='loss function', metrics=['accuracy'])
    return model
```

Figure A.4: Function for CNN Model

Bibliography

[JAISWARA 21] RANJANA JAISWARA, LAURE DESUTTER-GRANDCOLAS & MANJARI JAIN. *Taxonomic revision of Teleogryllus Mitratus (Burmeister, 1838) and T. occipitalis (Serville, 1838) in India, with the description of teleogryllus rohinae jaiswara amp; Jain sp. nov. and a key for teleogryllus species from India (orthoptera: Gryllidae)*. Zootaxa, vol. 5016, no. 1, page 81–106, 2021.

[Madiajagan 19] M. Madiajagan & S. Sridhar Raj. *Parallel Computing, Graphics Processing Unit (GPU) and New Hardware for Deep Learning in Computational Intelligence Research*. pages 1–15, 2019.

[Mitchell 83] Tom M. Mitchell, Jaime G. Carbonell & Ryszard S. Michalski. *AN OVERVIEW OF MACHINE LEARNING*. pages 3–23, 1983.

[Penar 20] Weronika Penar, Angelika Magiera & Czesław Klocek. *Applications of bioacoustics in animal ecology*. Ecological Complexity, vol. 43, page 100847, 2020.

[Samuel 59] A. L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers*. IBM Journal of Research and Development, vol. 3, no. 3, pages 210–229, 1959.

[Singh 21] Richa Singh & Manjari Jain. *Variation in call types, calling activity patterns and relationship between call frequency and body size in a field cricket, Acanthogryllus asiaticus*. Bioacoustics, vol. 30, no. 3, pages 284–302, 2021.

[Stowell 19] Dan Stowell, Michael D. Wood, Hanna Pamuła, Yannis Stylianou & Hervé Glotin. *Automatic acoustic detection of birds through deep learning: The first Bird Audio Detection challenge*. Methods in Ecology and Evolution, vol. 10, no. 3, pages 368–380, 2019.