TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS

A Major Project Mid-Term Report
On
संरक्षण: An Endangered Birds Recognition Portal

Submitted By:

Bhuwan Khatiwada	(THA075BEI009)
Bishwa Prakash Subedi	(THA075BEI011)
Niraj Duwal	(THA075BEI028)
Rewan Gautam	(THA075BEI032)

Submitted To:
Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

August, 2022

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS

A Major Project Mid-Term Report
On
संरक्षण: An Endangered Birds Recognition Portal

Submitted By:

Bhuwan Khatiwada	(THA075BEI009)
Bishwa Prakash Subedi	(THA075BEI011)
Niraj Duwal	(THA075BEI028)
Rewan Gautam	(THA075BEI032)

Submitted To:
Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics, Communication and Information Engineering

Under the Supervision of
Er. Kiran Chandra Dahal

August, 2022

ACKNOWLEDGEMENT

Firstly, we would like to dedicate our regards to the Institute of Engineering (IOE) for the inclusion of this Major Project on the syllabus for the course of Bachelors of Electronics, Communication and Information Engineering.

Also, we would like to thank our Department of Electronics and Computer Engineering, Thapathali Campus for the wonderful learning atmosphere throughout our time here at Thapathali Campus and for giving us this exciting opportunity to test our knowledge through this Major project.

The experience of doing this project will surely enrich our technical and teamwork skills to a great extent.

Bhuwan Khatiwada	(THA075BEI009)
Bishwa Prakash Subedi	(THA075BEI011)
Niraj Duwal	(THA075BEI028)
Rewan Gautam	(THA075BEI032).

ABSTRACT

Home to around 11,000 species of fauna, Nepal is a country rich in biodiversity. Among them about 900 species are birds. Due to various reasons like encroachment of their natural habitats, and rampage killing many of these species are facing the threat of extinction. Around 38 endangered birds in Nepal need conservation. The growing advancement in machine learning can support the preservation of those species and monitoring the status of birds in the ecosystem can assist researchers of Nepal's biodiversity in planning different strategies for their preservation. We propose an endangered birds classification system to identify bird calls from the audio data-set collected from Xeno-canto.org. This could be achieved by converting the time-domain audio data signal to time and frequency domain signal with wavelet transform, generating the scalograms and feeding it to the deep learning model architecture like efficientNet which is based on a convolutional neural network. A genetic algorithm can be used for parameter optimization.

Keywords: Audio, Birds, CNN, efficientNet, Genetic, ML

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
EfficientDet	Efficient Detection
EfficientNet	Efficient Network
GB	Giga Byte
GPU	Graphics Processing Unit
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IUCN	International Union For Conservation of Nature
ML	Machine Learning
MobileNet	Mobile Network
NumPy	Numerical Python
OS	Operating System
PC	Personal Computer
RAM	Random Access Memory
TPU	Tensor Processing Unit
UI	User Interface

1. INTRODUCTION

Artificial Intelligence (AI) is an attempt at mimicking human intelligence by machines. In this modern world, which is flowing towards the idea of automation in every field possible AI has become a big name. AI has found its presence in various places like search engines, voice assistants, different recognition software, automated vehicles, etc. Machine learning is a type of AI that learns by data through certain algorithms. It feeds on data to predict future values. A huge amount of data is available these days and proper utilization of it can empower the machine learning algorithm. So, by feeding the data corresponding to any real-life problem to a machine learning model better results can be expected. The applications of this are literally limitless and one of them can be in the field of endangered species conservation. Using audio recognition methods in natural habitats of various species, their presence can be detected and analyzed.

1.1. Background

Birds or Aves are one of the various classes of creatures that are found on this greenblue planet of ours. They have features like the presence of feathers, toothless beaked jaws, the laying of eggs, etc. There are about 11,162 species of birds that are found all over the world out of which most are generally found in the tropical region. Indonesia is home to the largest endemic bird species population in the world where about 500 species of birds are found. Australia is second in inhabiting a large number of bird species with a population of almost 360 bird species while the Philippines and Brazil inhabit almost 260 species. Endemic species of birds are not often found or are less found in higher latitude countries like Spain, Portugal, and other European countries.
According to reports, an estimated 159 bird species have gone extinct in the last 500 years. As per IUCN Red List in 2021, out of 11,162 species 1,445 species are classified as threatened or endangered which is about 13 % of the total species on earth i.e. around 1 in 7 species are endangered. Brazil is home to nearly 172 of these endangered species and Indonesia has about 106 endemic species that are threatened [1]
Nepal is home to 875 species of birds making it 27th on the list of having the most endemic bird population. Out of these about 38 species are globally threatened i.e. about 5 % of the total bird species. Among the 38 threatened species, 9 are critically 2 endangered, 8 are endangered and 21 species are vulnerable [2]. Some of the endangered birds of Nepal are Black-breasted Parrotbill, Black-necked Crane, Greysided Thrush, Rustic Bunting, Sarus Crane, Wood Snipe, Yellow-breasted Bunting, etc.

1.2. Motivation

All the plants and animals on this planet constitute an ecosystem that is very delicate. The loss of various species can cause harm to this delicate balance and degrade the biodiversity of the ecosystem. The rapid advancement of developmental activities which have been performed in an unplanned manner has caused the loss of habitat for different species. Other illegal activities like poaching and killing of animals for fur, tusks, and leather have caused the number of different wildlife to decrease to a very low number. Many birds and animals have become endangered because of these and many other reasons. The problem of overpopulation, pollution, and global warming is causing an adverse effect on not only humans but also wildlife. So, this project is aimed at a step toward the use of technology in the conservation of nature.

1.3. Problem statement

Birds are wildlife that can be found in the wild i.e. in dense and deep jungles and forests. Birds can vary in size, they can be big or small but still, they can be hard to see visually in their own natural habitat. Many birds have features that give them the ability to hide so visual recognition may not be the best approach for the classification of birds. For this, we can use the data set of different birds singing and use it to recognize birds using a machine learning algorithm that analyses the birds singing.

1.4. Objectives

The major objectives of this project is:

1. To recognize birds through their audio input data

1.5. Project Scope and Applications

This project can be of assistance to various conservationists and can be implemented in protected regions like conservational areas and national parks. It can help national parks recognize the migratory birds that arrive in their habitation just by analyzing their songs. It can also be used by adventurers, explorers, and tourists to learn about the birds they might encounter in their journey simply by recording the sound sung by the bird.

2. LITERATURE REVIEW

2.1. Deep Learning

Deep learning is a subset of machine learning capable of making machines mimic human intelligence to perform required tasks and get better themselves with experience or the data they collect. Deep neural networks are made up of several layers of interconnected nodes, each of which improves upon the prediction or categorization made by the one underneath it. Forward propagation refers to the movement of calculations through the network. A deep neural network's visible layers are its input and output layers. The deep learning model ingests the data for processing in the input layer, and the final prediction or classification is performed in the output layer. Backpropagation employs techniques like gradient descent to calculate prediction errors before changing the function's weights and biases by iteratively going back through the layers in an effort to train the model. A neural network can generate predictions and make necessary corrections for any faults with the forward propagation and backpropagation working together. The algorithm continuously improves in accuracy over time.

Computer vision (CV) is a field of study that helps computer "see" and differentiate the different objects or contents in an image or videos. With the advent of deep learning algorithms, the accuracy for object detection has increased drastically. Object detection and tracking are one of the most discussed applications in the field of deep learning. It has been a research area because of its tremendous application in various ways like medical diagnostics, automations, fraud detections, self-driving cars, and a lot more to discover. Complex models like Convolutional Neural Networks (CNN) are the key component of this kind of applications.

2.2. Convolutional Neural Networks

A convolutional neural network (CNN) is a type of artificial neural network comprised of neurons that self-optimize through learning. The neurons receive input and perform scalar product and non-linear function [3]. Convolutional neural networks outperform other neural networks when given input datasets are images, voice, or audio. They have layers like the Convolutional layer, Pooling layer, and Fully-connected layer.

A convolutional network's initial layer is the convolutional layer whereas the fully-connected layer is the last layer. Many convolutional layers or pooling layers can exist in between. The CNN becomes more complicated with each layer, detecting larger areas of the picture. Early layers emphasize basic elements like colors and borders.

2.3. Data Augmentation

Machine learning or deep learning model's accuracy and efficiency is highly dependent on the quality as well as the quantity of training data. Before training the model the available data-set is divided into training data set, validation data-set and test data-set. The model is trained only on the training data-set. Hence, a large amount of data is necessary for training a model. But, the necessary data-sets are not readily available and one of the biggest challenges during developing a deep learning model is data-scarcity.

The amount of data available can be increased by manual data collection but it is a very tedious, costly and time consuming method for increasing the quantity of data. For easy increment of the size of data we use data augmentation. Data augmentation is a technique of increasing the quantity and diversity of training data-set by applying random(but realistic) transformations [4]. It is a process of artificially increasing the amount of data by generating new data points from existing data. it increases the number of samples a machine learning model sees during training. The goal of data augmentation is to cover the problem space as much as possible by creating derived data points from original data points.

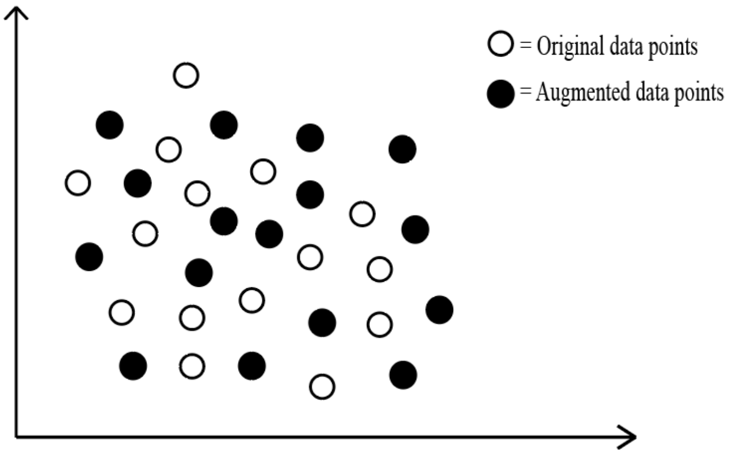


Figure 2.1: Original and augmented data points in problem space

While data augmentation is mostly performed in the field of image classification. It can be very useful in the field of audio recognition as well. For increasing the diversity of our data-sets, some of the transformations used for data augmentation are:

1. **Gaussian noise:** Gaussian noise is a noise or a signal which has its probability density function(pdf) equal to that of normal distribution(also known as Gaussian

distribution). Normal distribution is also known as bell-curve because of its bell shaped probability distribution. The Gaussian noise is a random noise, which makes it preferable in machine learning as most of the events in real world are random in nature. When we observe a random event that is the sum of many independent events, all random variables appear to be Gauss variables[5].

Gaussian noise is added to a data-set as follows:

(a) Random noise is calculated and assigned to the variable.

(b) The noise is added to the data-set($Data - set = Data - set + Noise$).

For Gaussian Distribution, probability density function is calculated using the formula:

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{2.1}$$

It helps in making the model more robust against noise.

2. **Time-shifting:** Time-shifting means shifting of an audio signal either to the left or right in the time axis by a certain value of time. If the signal is shifted to the left then it means the signal is advanced and if the signal is shifted to the right then it means that the signal is delayed. Say a signal is shifted left by 'x' seconds then there is silence for the last 'x' duration where as if the signal is shifted right by 'x' seconds then there is silence for first 'x' duration of the signal [6].

3. **Pitch-shifting:** Pitch-shifting is another transformation that can used for data augmentation. In this method, the pitch of the signal is either raised or lowered. In this method the pitch of the signal is changed without changing the speed of the audio signal.

2.4. Hyper-parameter Tuning

Parameters are the attributes that we use to execute specific model actions. Additionally, they explain how each member function works. Model parameters are the features that distinguish and characterize a model. Although the parameters are adjusted by the model itself, the values may be modified or altered to get the highest level of efficiency and accuracy. All of the parameters cannot be altered to meet the requirements.

Hyperparameters in machine learning are those parameters that are explicitly defined by the user to control the learning process. These hyperparameters are used to improve the learning of the model, and their values are set before starting the learning process of the model. Hyperparameters may be manually modified to get more accurate, efficient, and consistent outcomes. Their values can be adjusted to achieve the best solution.

Machine learning techniques require the hyper parameter in order to classify data. There are various hyper parameters for each machine learning approach. Making the appropriate parameter selection can have a big impact on the accuracy of the predictions. Because of this, it's crucial to adjust the hyperparameters rather than using the machine learning's default settings. By testing every feasible value, it is possible to manually determine these hyperparameters. However, because there are so many conceivable combinations, doing so takes a lot of time. For this reason, optimization algorithms like Genetic algorithm are frequently used to automatically determine the best hyper parameter. As a result, hyperparameter optimization may also be used to describe hyperparameter tuning.

The selection of the objective function and the performance matrix is the first step in the hyperparameter tuning procedure. The hyperparameters that need to be tuned are then recognized in their types, and the appropriate optimization strategy is identified. First, train the ML model with default or common values for the baseline model. Start the tuning process after selecting on a broad search space as the viable hyper parameter domain based on manual testing and knowledge domain. After that, you can reduce the search space based on the regions of currently used and effective hyper parameter values and, if necessary, look into alternative search spaces. The final response should be the hyper parameter configuration with the best performance.

2.5. Genetic Algorithm

Darwin's theory of evolution states that a population of people that differ from one another is maintained via evolution (variation). Better suited individuals have a higher likelihood of living, reproducing, and passing on their characteristics to the following generation (survival of the fittest). Inspired by this theory, genetic algorithms, a search heuristic, are able to overcome problems faced by traditional algorithms. Genetic algorithms can imitate the processes of natural selection, reproduction, and mutation and can generate optimized solutions to a variety of problems.

Initial population, Fitness function, Selection, Crossover, and Mutation are the phases of genetic algorithm. Genes are organized into chromosomes. A chromosome, for instance, may be represented as a binary string in which each bit corresponds to a gene. A population is a group of chromosomes, each of which serves as a representation of an individual. The individual is assessed based on their fitness scores, which are calculated by the fitness function, in each iteration. Superior fitness score recipients signify better solutions and are more likely to cross over and be passed on to the following generation. The accuracy of the model using those chosen features, for instance, would be the fitness function if it were a classification issue and genetic algorithms were employed for feature selection. Fitness score of the individual signifies the ability of an individual to compete.

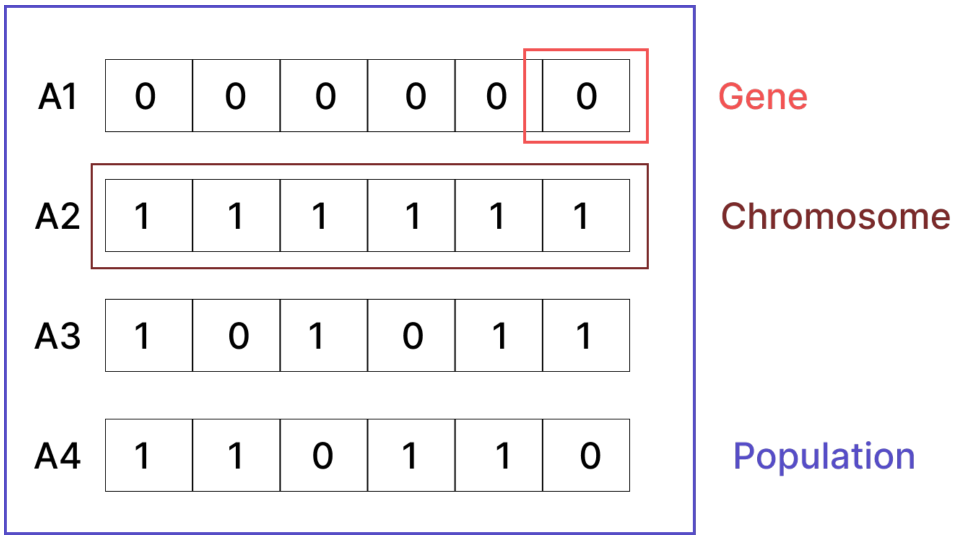


Figure 2.2: Gene, Chromosome and Population in Genetic Algorithm

The operators of genetic algorithm are discussed below:

1. **Selection Operator**

   The goal is to select individuals with highest fitness scores and let them pass on their genes to succeeding generations. So, fit offspring after born from fit parents.

2. **Crossover Operator**

   Crossover means mating between individuals. Two individuals of high fitness scores are selected using selection operator and crossover sites are chosen randomly. Then genes of the selected individuals are exchanged at these crossover sites thus creating a completely new individual (offspring).

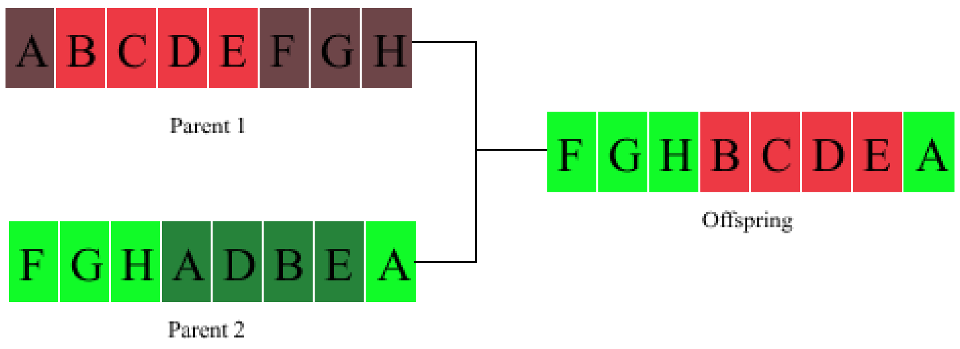


Figure 2.3: Crossover of genes producing new offspring.

3. **Mutation Operator** Mutation is then performed on the produced offspring so that the diversity is maintained in the population. This avoids premature convergence of the algorithm.

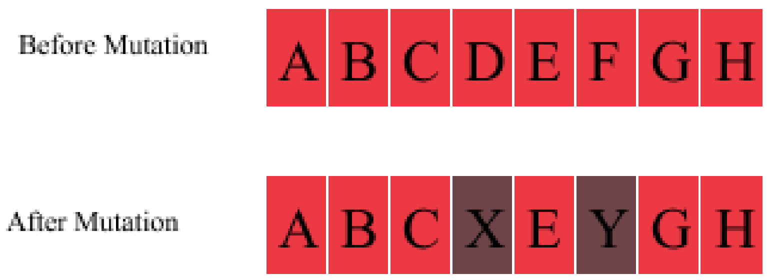


Figure 2.4: Mutation of new offspring.

In summary the genetic algorithm follows the following steps:

1. Randomly initialize populations ’p’
2. Determine fitness scores of population.
3. Until convergence repeat:

   a) Select parents from population

   b) Crossover and generate new population

   c) Perform mutation on new population

   d) Calculate fitness for new population

2.5.1. Hyper-parameter Tuning Using Genetic Algorithm

By reducing the number of function evaluations required during each optimization, it is possible to identify the best hyper-parameter for that model. Additionally, the definition of optimization is as follows: ”Given a function that accepts inputs and outputs a numerical value, how can it efficiently discover the inputs, or parameters, that maximize the function’s output?” As a result, when adjusting or optimizing the hyper-parameter, the user will use the input as a function of the hyper-parameter model and the output as a measurement of the model’s performance.

Using a genetic algorithm, hyper-parameter tuning process involves following steps:

1. Splitting the data-set into training and testing
2. Building the optimizer
3. Define the parameters and store them in dictionaries
4. Build a classifier that specifies the necessary number of generations, population size, offspring size, and the model’s hyper-parameters that must be optimized.

2.6. Wavelet Transform

We are clear about the problem at hand i.e we are classifying audio recordings of birds, meaning that this is an audio processing problem. Now to classify audio signals or any signals we have to analyse its features and for that we first need to be able to extract the features of the signal. For this, there are various methods like Fourier Transform, Short Time Fourier Transform(STFT), Wavelet Transform etc. Later on we will be discussing about the use of Wavelet Transform for feature extraction. So *What is Wavelet Transform?* and *Why Wavelet Transform?*. But first, prior to understanding the magic of Wavelet Transform, we need to understand about the Fourier Transform and its shortcomings.

Fourier Transform is a means by which we convert a signal given as a function of time or space into a function of its frequency. It is a mathematical function that decomposes a waveform, which is a function of time, into the frequencies that make it up. It explains a given function, f(t) as a combination of infinite number of sine/cosine waves. It is useful in analysing signals as it provides frequency information of signals.
Mathematically,

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt \tag{2.2}$$

Fourier Transform gives us the information about different frequency components of a signal but fails to tell us when in time do these frequency components exactly exist. Time information are not of much concern while studying stationary signals i.e the signals in which the frequency components remains constant throughout and doesn't change. Hence, Fourier Transform is ideal for studying stationary signals. But when it comes to non-stationary signals, its fails to tell the complete picture. This is the main drawback of Fourier transform, as the signals that we encounter in real life are generally non-stationary in nature. So for the better study of these signals we require frequency as well as time information. Therefore, we must move past Fourier Transform.

Since, the Fourier Transform lacked the ability fulfill our complete requirement we needed a new method for analyzing signals. A way or a method to analyse a signals both time and frequency information at the same time. This is where Wavelet Transform comes in handy. So, *What is Wavelet Transform?* and also *What is a Wavelet?*

A Wavelet is a waveform of a effectively limited duration that has an average value zero. Or simply said, a wavelet is a small waveform. Mathematically, it is defined as,

$$\psi_{\tau,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{2.3}$$

where,
τ= translation(position) parameter,
$s = \frac{1}{f}$= dilation(scale) parameter.

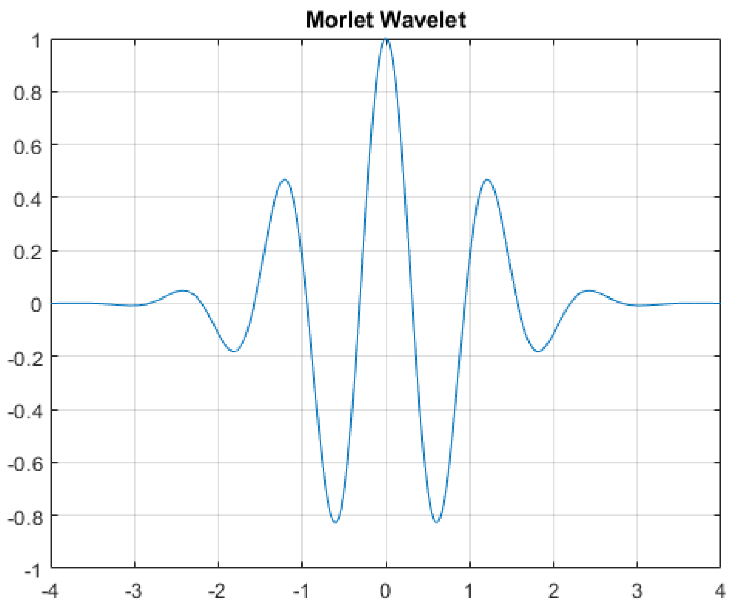


Figure 2.5: Morlet Wavelet

Some of the examples of wavelets are Morlet wavelet, Gaussian wavelets, Meyer wavelet, Shannon wavelet, Haar wavelet etc. Wavelets are defined by its two parameters: translation parameter and scale parameter. Translation(position) parameter dictates the position of the wavelet, it defines where in time the wavelet is positioned. Dilation (scale) parameter defines how stretched or compressed the wavelet is i.e it describes the width of the wavelet.

Wavelet Transform make use of wavelets as its basis function to extract the time and frequency information from a signal. It is a mathematical tool for analyzing data where features vary over different scales. For signals, features can be frequencies varying over time, transients, or slowly varying trends. While Fourier analysis consists of decomposing a signal into sine waves of specific frequencies, wavelet analysis is based on decomposing signals using wavelets of varying signals and locations.

Mathematically, Wavelet transform is defined as,

$$F(\tau, s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} f(t) \psi^{*} \left(\frac{t-\tau}{s} \right) \tag{2.4}$$

Here, $\psi^{*}\left(\frac{t-\tau}{s}\right)$= complex conjugate of scaled and translated $\psi(t)$ and,
$F(\tau, s)$= required wavelet coefficients

Approximate coefficients are those wavelet coefficients that represent low frequency of signal. Detailed coefficients are those wavelet coefficients that represent higher frequencies.

The way Wavelet Transform works is: we change the ’s’ value or the value of dilation parameter to change the width of the wavelet by either stretching or compressing it (known as scaling) and moving the scaled wavelet across the signal by changing the value of translation parameter. This is performed for different set of values of parameters until all the required information of the signal are extracted.

Wavelet transform results in analysing a signal into different frequencies at different resolutions, known as multi resolution analysis.

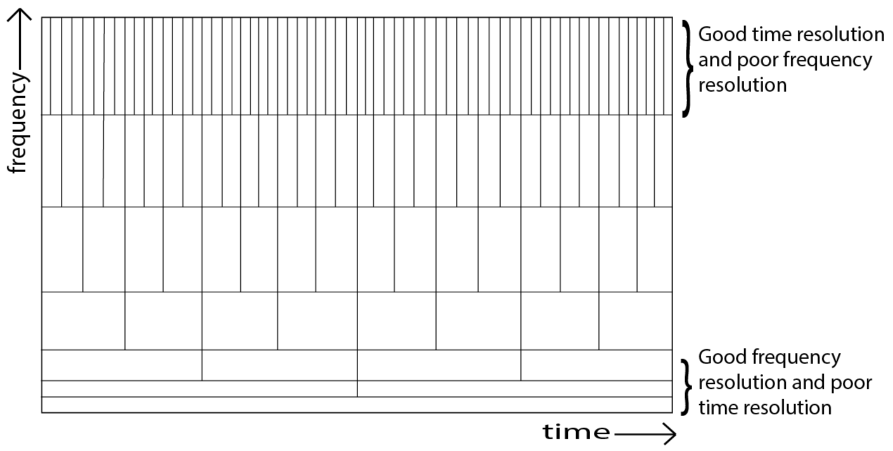

frequency
Good time resolution and poor frequency resolution
Good frequency resolution and poor time resolution
time

Figure 2.6: Tiling of time-frequency plane in multi-resolution analysis.

Expanded wavelets are better at resolving low frequency components of a signal with poor time resolution. Whereas, shrunken or compressed wavelets are better at resolving higher frequency components of signal with good time resolution.

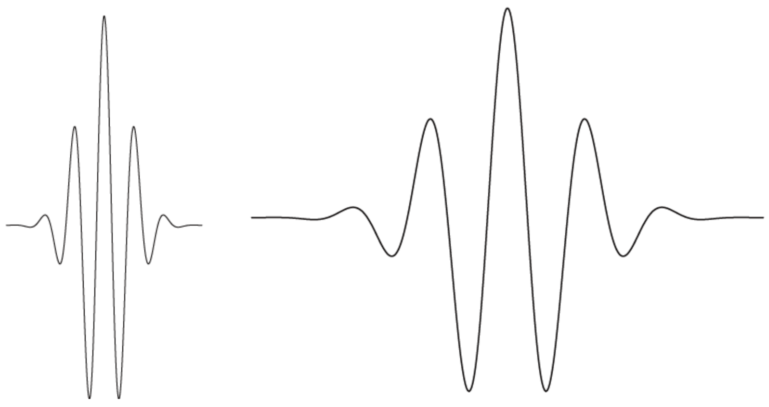


Figure 2.7: Compressed wavelet(on the left) and expanded wavelet(on the right).

In Continuous Wavelet Transform or simply Wavelet Transform, calculation of wavelet coefficients at every possible scale produces a large amount of data. If s and τ are chosen to be discrete, wavelet transform wont generate huge data. If s and τ are dyadic (i.e. based on power of 2) then analysis becomes much more efficient and accurate.

Discrete Wavelet Transform is defined as,

$$D[a,b] = \frac{1}{\sqrt{b}} \sum_{m=0}^{p-1} f[t_m] \psi \left(\frac{t_m - a}{b} \right) \tag{2.5}$$

where, $a = k \cdot 2^{-j}, b = 2^{-j}$

2.6.1. Scalogram

A spectrogram is a visual representation of the ”loudness” or signal strength over time at different frequencies contained in a specific waveform. One may observe the amount of energy at different frequencies, such as 2 Hz vs. 10 Hz, as well as how it changes with time. Frequencies of sound waves created by people, machines, animals, whales, jets, etc., as recorded by microphones, are frequently displayed using spectrograms in various scientific fields. Spectrograms are sometimes referred to as sonographs, voiceprints, or voicegrams when they are applied to an audio input.

A scalogram is a graph of the continuous wavelet transform coefficients of a signal that shows their absolute value as a function of frequency and time. It is used to evaluate whether a parameter creates a continuous Guttman scale, which spans from least to most favourable.

Scalograms are favoured over spectograms in wavelet transform because their resolution is finer at low frequencies and coarser at high frequencies. Scalograms are much more efficient for wavelet transform than spectograms because they can analyze any frequency variation at any time.

2.7. Previous Works

Some works have been done on bird classification using different approaches. Marini et al., 2013 classified bird species based on color features. Here, color segmentation was applied to remove the background elements and define candidate regions for the presence of the birds. The normalized color histograms were computed from there. The histogram bins were fed to the machine learning algorithm.

But it is true that it’s easier to hear birds than to see them. Extensive bird classification research can be observed with audio data. The annual BirdCLEF challenge [7] made this problem more popular. The 2021 edition of this challenge was to predict bird species among 397 birds in every 5 seconds of test data. The training dataset was provided in the challenge from all over the regions but test recording was provided only from four different places: New York, California, Costa Rica, and Colombia. The 10th place holder of the 2021 BirdCLEF challenge, Conde et al. [8] discussed a classification solution to this problem using custom augmentations and not limited to only the audio dataset. The additional features provided in the competition like the appearance of other birds in the audio, the rarity of the bird, and (latitude and longitude) were considered.

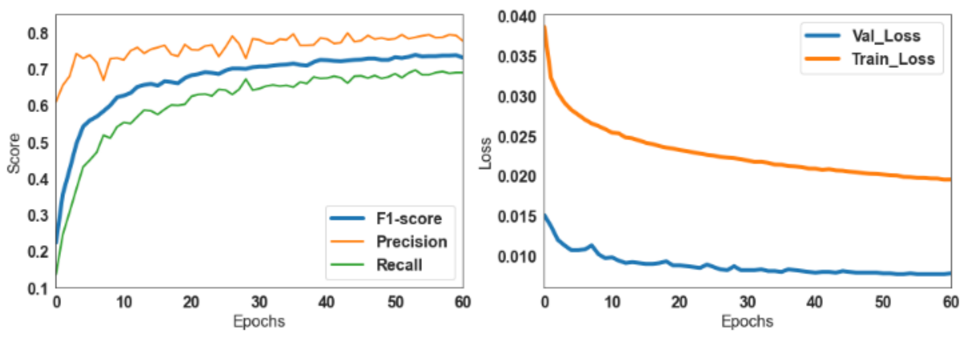


Figure 2.8: Evaluation of the ResNeSt-50 model

Since data augmentations were only applied during training, the training loss was comparatively lower than validation loss.

In 2019, Incze et al. used compact and performance-oriented MobileNet as their starting checkpoint. Spectrograms based on grayscale and jet color were used for the input to the neural network. In the grayscale, 0 represents white and 1 represents black while in jet color 0 represents blue, around 0.5 represents yellow and 1 represents red.

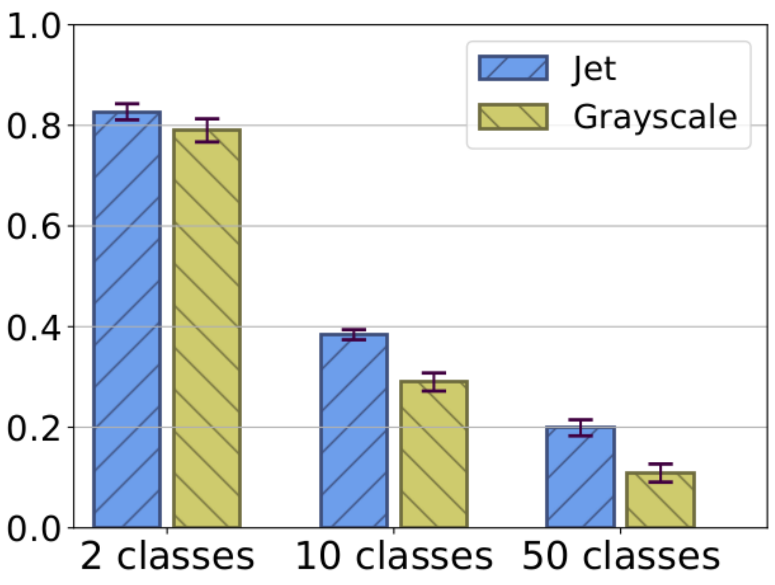


Figure 2.9: Accuracy score for Jet and Grayscale color map

The accuracy achieved with both the approaches were decreasing with the increase in the number of classes.

We propose our system to reduce the noise using wavelet transform and use a genetic algorithm for parameter optimization.

X. Xiao et al. [9] proposed Genetic Algorithm to optimize the hyper-parameters in CNNs. In their work, they did not restrain the depth of the model. Experimental results show that they can find satisfactory hyper- parameter combinations efficiently with accuracy about 88.92% and within 24.55 hours which is relatively better than random search algorithm.

Esteban Real et al. [10] came up with a mutation only evolutionary algorithm. The deep learning model grows gradually to find a satisfactory set of combinations. The evolutionary process is slow due to the mutation only nature.

X. Xiao et al. [11] proposed a Genetic Algorithm based method to select network on multi-node clusters. They tested the GA to optimize the hyper-parameter of a 3-layer CNN. The distributed GA can speed up the hyper-parameter searching process significantly.

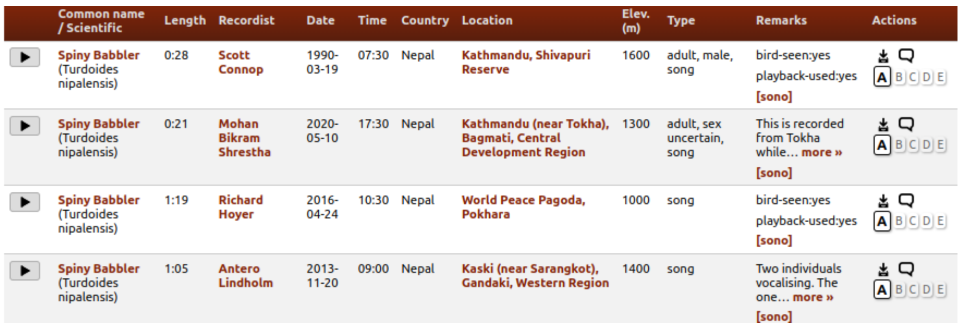
In 2020, J. Han, D. Choi, S. Park, S. Hong. [12] applied GA to solve the hyper-parameter optimization problem. In their work, the validation accuracy and the verification time are combined into the fitness function. The model is simplified to a single convolution layer and a single fully connected layer. They evaluated their method with two datasets, the MNIST dataset and the motor fault diagnosis dataset.

In 2021, Sanghyeop Lee et. al [13] performed Alzheimer's disease classification using a genetic algorithm. The model's parameter that minimizes the loss function was found using an optimization technique. Network configuration and hyperparameters were both chosen as search spaces in order to identify the best network architecture for the input, and the evolutionary algorithm was used to determine the best structure. For their algorithm implementation, binary representation was used to encode connections between convolutional layers. Also, the activation functions and optimization algorithm were encoded into bits and separated using bars. The possibility of optimizing the network architecture using GA, where its search space includes both network structure configuration and hyperparameters is illustrated. To verify the performance of their algorithm, they used an amyloid brain image dataset that is used for Alzheimer's disease diagnosis. As a result, this algorithm outperformed Genetic CNN by 11.73% on a given classification task

3. REQUIREMENT ANALYSIS

3.1. Dataset Analysis

On the website xeno-canto.org, bird sounds from throughout the globe are shared. The recordings are uploaded on the website by the contributors who travel around.



Common name / Scientific	Length	Recordist	Date	Time	Country	Location	Elev. (m)	Type	Remarks	Actions
Spiny Babbler (Turdoides nipalensis)	0:28	Scott Connop	1990-03-19	07:30	Nepal	Kathmandu, Shivapuri Reserve	1600	adult, male, song	bird-seen:yes playback-used:yes [sono]	A B C D E
Spiny Babbler (Turdoides nipalensis)	0:21	Mohan Bikram Shrestha	2020-05-10	17:30	Nepal	Kathmandu (near Tokha), Bagmati, Central Development Region	1300	adult, sex uncertain, song	This is recorded from Tokha while... more » [sono]	A B C D E
Spiny Babbler (Turdoides nipalensis)	1:19	Richard Hoyer	2016-04-24	10:30	Nepal	World Peace Pagoda, Pokhara	1000	song	bird-seen:yes playback-used:yes [sono]	A B C D E
Spiny Babbler (Turdoides nipalensis)	1:05	Antero Lindholm	2013-11-20	09:00	Nepal	Kaski (near Sarangkot), Gandaki, Western Region	1400	song	Two individuals vocalising. The one... more » [sono]	A B C D E

Figure 3.1: Xeno-canto.org User Interface

We collected 2215 audio recordings from this website of 41 birds of which 38 of them are endangered species. The dataset was increased to 6733 audio recordings after the 10-second splitting and data augmentation through gaussian noise addition for birds having less than 30 files. A total of 5407 audio recordings were used for training, 639 for validation, and 687 for testing purposes.

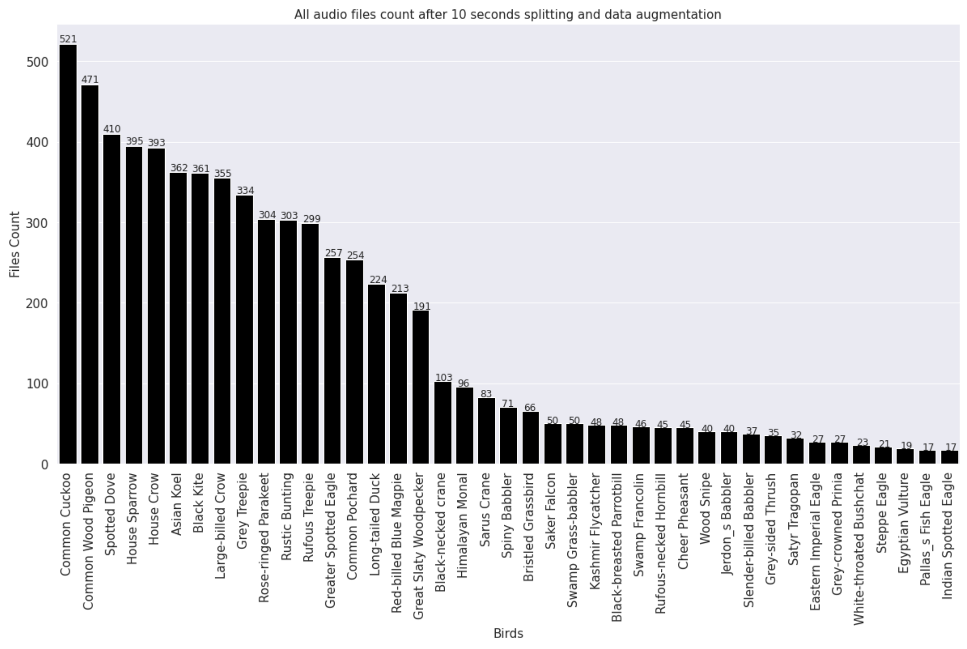


Figure 3.2: Audio Files Count for each Bird

3.2. Hardware Requirement

Any normal PC is preferable for training the model using Google Colab but if we want to train it on our own system, high CPU processors, high GPU or TPU and RAM are necessary. This is because deep learning must deal with lots of data and would cost us time. For the website, any system that supports a browser and the internet is suitable.

3.3. Software Requirement

The system requires coding which is written in python for backend purpose. Under the hood, frameworks like TensorFlow contribute to deep learning. Some utility libraries like Librosa, PyDub, Audiomentations are used for working on audio dataset. For training of model, we used Google Colab which is free cloud service that provides free GPU. The testing of the model can be done in VSCode locally. For the deployment part, we will be using FastAPI and AWS cloud service.

3.4. Feasibility Study

3.4.1. Economic Feasibility

The system is more software than hardware. So, it will be economically feasible. But to make the model or to train it, we need a normal PC which is present everywhere nowadays. Anyone who can buy a system that has a browser and internet facility can use it.

3.4.2. Technical Feasibility

The system will be trained with lots of data using deep learning. Deep learning algorithms have the greatest accuracy compared to any other approach. The aim of the project will be to achieve as higher accuracy as possible. This will certainly make the project technically feasible.

3.4.3. Operational Feasibility

The website will be created using HTML, CSS, and JavaScript. These tools are one of the most used tools for web development. So, the system will be user-friendly and anyone with knowledge of computers and websites can easily operate it.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1. System Architecture

4.1.1. Data Collection and Feature Extraction

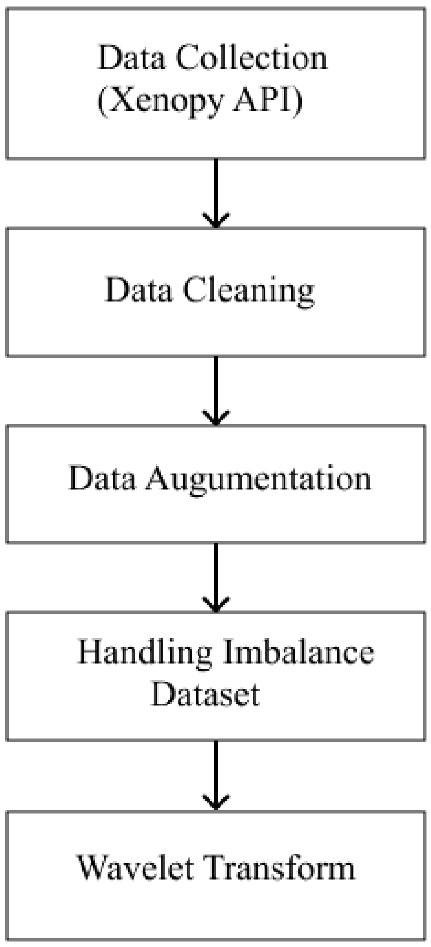
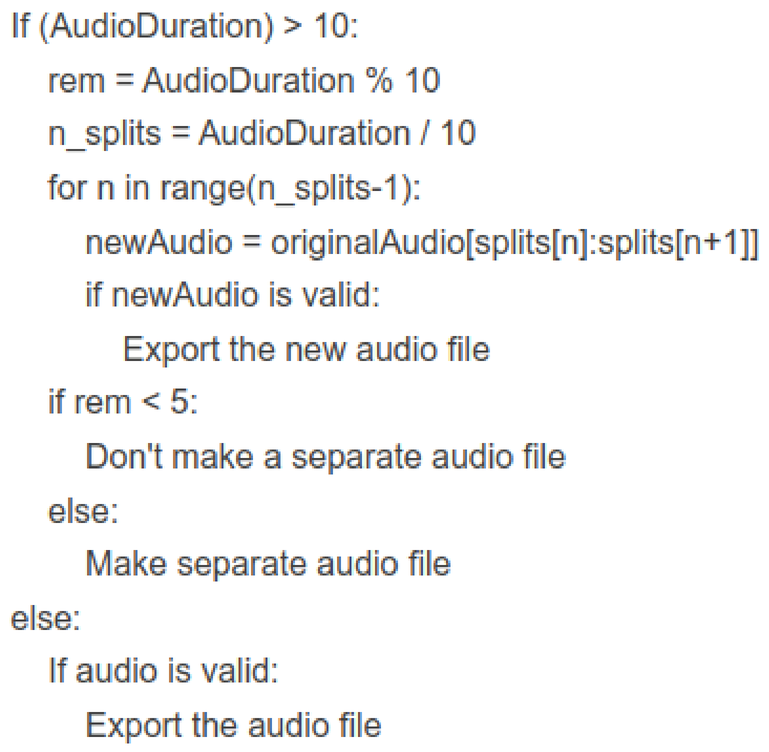


Figure 4.1: Data Collection and Feature Extraction Block Diagram

The methodology for data collection and feature extraction is shown above. The audio dataset we require for our task is collected using Xenopy API [14] which can download the dataset available at https://xeno-canto.org/. A lot of data might be misleading and needs to be cleaned. Since the dataset might not be enough, we perform data augmentation by inserting noise to increase the dataset. The dataset might contain an uneven distribution of samples for each class of birds. This leads to the bias of the machine learning model toward the majority class ignoring the minority class. The imbalanced dataset is handled using sampling techniques. Wavelet transform is used for the removal of noise and feature extraction since it gives us the information of both the time and frequency domains.

4.1.2. Audio Splitting Algorithm



```
If (AudioDuration) > 10:
    rem = AudioDuration % 10
    n_splits = AudioDuration / 10
    for n in range(n_splits-1):
        newAudio = originalAudio[splits[n]:splits[n+1]]
        if newAudio is valid:
            Export the new audio file
    if rem < 5:
        Don't make a separate audio file
    else:
        Make separate audio file
else:
    If audio is valid:
        Export the audio file
```

Figure 4.2: Audio File Splitting Algorithm

The dataset contains the varying duration of audio files. For maintaining uniformity and also to increase the dataset, we split the dataset into a 10-second duration file. It might be the case that the audio file split might contain only silence or some random environmental noise. To cope with this problem, we validated if the new audio file is valid audio or not by simply checking if it contains at least half of the maximum amplitude the original file contains. Also, if the audio file is small enough, instead of creating a separate audio file, we appended it to the last file that was created on the split.

4.1.3. Training

The training process after feature extraction is shown in block diagram below. After feature extraction, the data needs to be split into training and testing because the model should not be tested on the same data it was trained on. This is an important step to accurately evaluate our model to see if it is generalized for the new dataset. Then, we define the deep convolutional neural network model, learning rate to determine how fast or slow to move for finding the minimum loss, batch size to train in batches, and epochs to determine how many times we want to train the whole data. The parameters we choose might not be the best ones and should be tuned. A genetic algorithm helps to execute several solutions iteratively till a satisfactory solution is found.

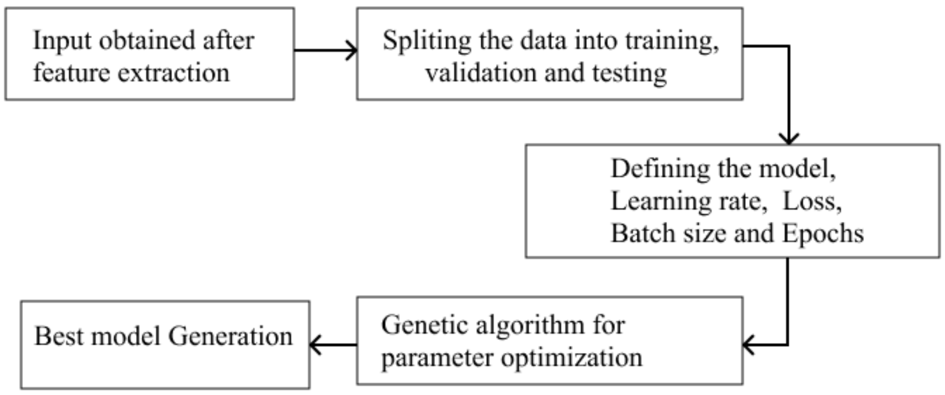

Input obtained after feature extraction
Spliting the data into training, validation and testing
Defining the model, Learning rate, Loss, Batch size and Epochs
Genetic algorithm for parameter optimization
Best model Generation

Figure 4.3: Training Block Diagram

4.1.4. Parameter Optimization

For the parameter optimization, first, we need to initialize any random values for it. Then, the initial random parameters population of chromosomes are generated. The model is trained with the initial parameters using the dataset which is preprocessed accordingly. The model is a convolutional neural network architecture. Different evaluation metrics can be implemented to see the performance of the model with the given parameters. If the stopping criteria are reached like the number of generations, evaluation threshold, etc, the obtained optimized parameters are saved and fed to be used in the best model. If it's not the case, the selection can be done by the fittest individuals, and their genes are interchanged to perform crossover. Using mutation, random change of genes in a new offspring can be done to maintain diversity.

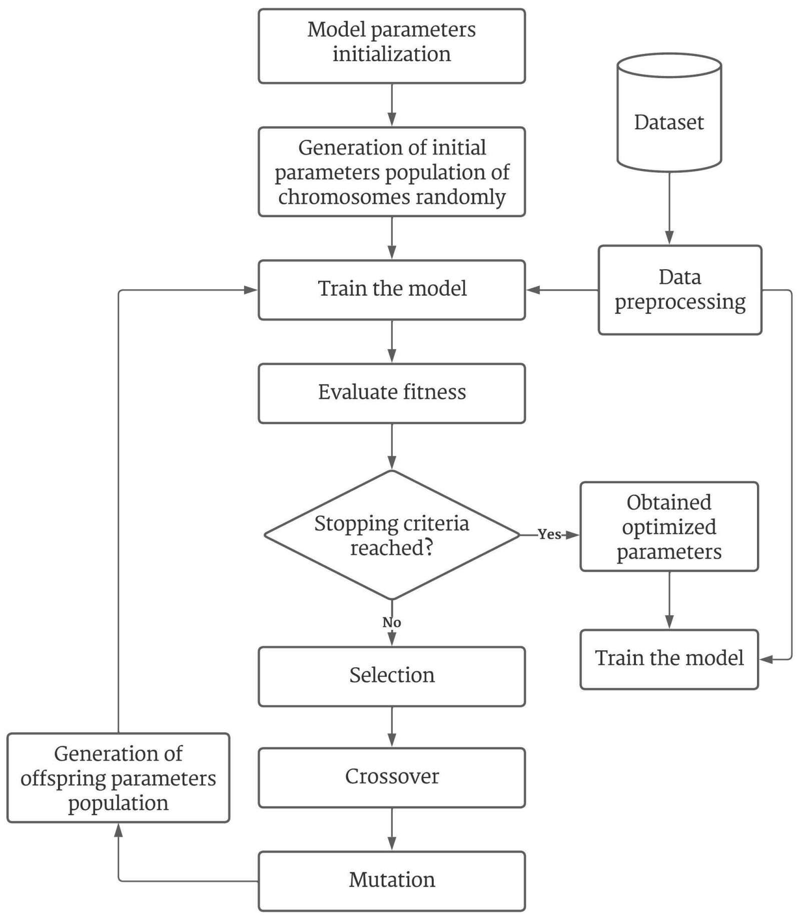

Model parameters initialization
Dataset
Generation of initial parameters population of chromosomes randomly
Train the model
Data preprocessing
Evaluate fitness
Stopping criteria reached?
Yes
Obtained optimized parameters
No
Selection
Train the model
Generation of offspring parameters population
Crossover
Mutation

Figure 4.4: Parameter Optimization with Genetic Algorithm

4.2. System Methodology

4.2.1. EfficientNet Algorithm

EfficientNet is a light-weight convolutional neural network architecture used for image classification. While other CNN architectures use only depth scaling, efficientNet uses a technique called compound scaling for the development of efficient image classification model.

As mentioned by Mingxing Tan and Quoc V. Le in their research ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”[15], performing scaling in any of the dimension of network i.e. depth, width and resolution increases the accuracy of the model but as the model gets bigger the accuracy decreases and the model faces the problem of vanishing gradient. They also concluded that for better accuracy and efficiency of the model, a balanced scaling of all the dimensions of network depth, width and resolution is required. An image of higher resolution represents a larger amount of data. As the amount of data increases the number of neural layers required for training a model also increases.For this scaling is performed, either depth scaling which increases the depth of the layers and width scaling which increases the number of channels or the number of feature maps.

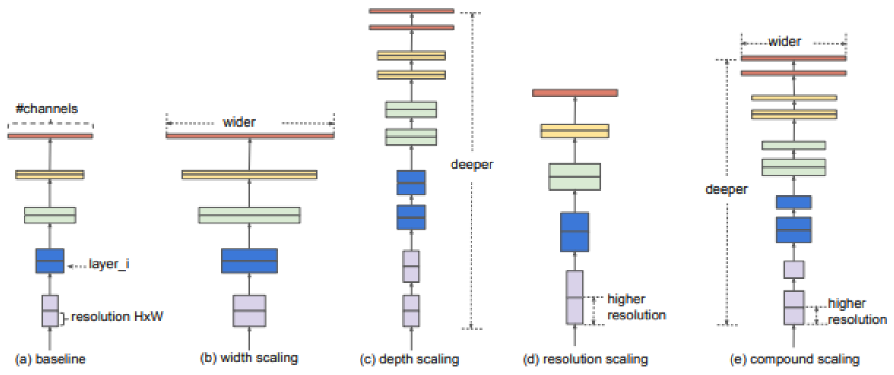
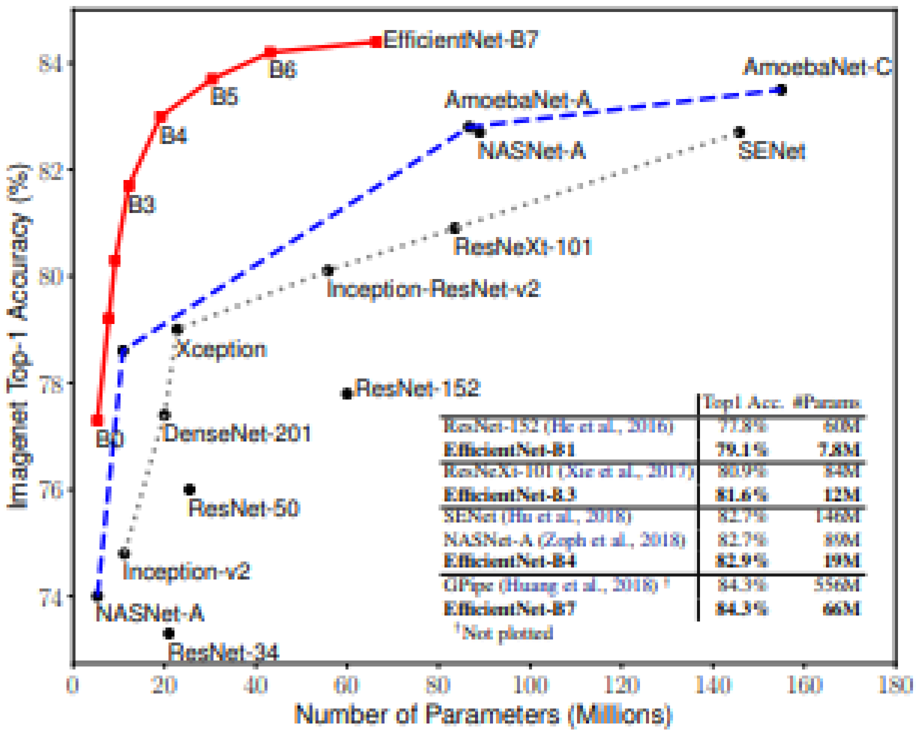


Figure 4.5: Different methods of model scaling.

The following graph shows the comparison of various neural networks based on their accuracy for a given number of parameters.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Figure 4.6: Accuracy of different neural networks for a given number of parameters.

From the figure, it is clear that efficientNet provides greater accuracy with far less number of parameters as compared to other networks.

We discussed that efficientNet uses a technique called compound scaling. Compound scaling is the balanced scaling of all three dimension depth, width and resolution in a constant ratio. The ratio for different scaling is given by the formula:

$$f = \alpha \cdot \beta^{\phi} \cdot \gamma^{\phi} \tag{4.1}$$

where, α is depth scaling factor, β is width scaling factor, γ is resolution scaling factor and $f$is network scaling factor. Here, α, β, γ are constant values and ϕ is a variable.

For scaling the depth, width and resolution we require a baseline model. EfficientNet B0 is the baseline model and all other models efficientNet B1, efficientNet B2, efficientNet B3 and so on are scaled versions of the baseline model. The baseline model is a fixed network architecture that is developed using neural architecure search (NAS) i.e it is made up of other neural networks. For efficientNet B0 or our baseline model, the value for α is 1.2, β is 1.1, γ is 1.1 and ϕ is 1. These values are determined by performing grid search.

5. IMPLEMENTATION DETAILS

5.1. Google Colab

We have used Google Colab to write and execute our code in python. Since, it is easier, faster and provides us with free cloud RAM and GPU, it can help us to train our model quickly.

5.2. Librosa

Librosa is a python package that analyzes music and audio. It contains the components required to construct audio information retrieval systems [16]. Through librosa, we can analyze the audio data of the birds in our dataset, and is helpful to extract features for our machine learning models. We used librosa in our project to load the audio file and perform operations on it.

5.3. NumPy

NumPy helps us in dealing with the numerical calculations we need while performing different operations. There is a privilege of parallel computing with NumPy so it makes the vector operations faster.

5.4. Scikit-learn

Scikit-learn commonly known as sklearn is the python library which is mostly used for machine learning. It is open source and free package. It provides efficient tools for machine learning and statistically modeling that includes classification, regression, clustering etc. through a consistent interface in python. In our project, scikit-learn can be useful for any utility function for obtaining the confusion matrix, classification report, ROC and AUC scores and also for creating some baseline model.

5.5. Matplotlib

Data visualization and analysis is performed using Matplotlib. The accuracy, loss, classes scores heatmaps, confusion matrix generated using scikit-learn is plotted using Matplotlib.

5.6. TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning developed by the Google Brain team for internal Google use [17]. Built-in 2015, it is the most widely used machine learning and deep learning package. Tensors are the multidimensional arrays that are commonly operated in neural networks and hence the name 'TensorFlow'. It helps us to create multi-layers neural networks to implement classification, regression, discovery prediction, and creation. TensorFlow in our project implements the deep learning models used in our project for our classification task. TensorFlow contains the pre-trained efficientNet model and we can use the weights values. The model can be adjusted to our use case by removing the last layer of the model and adding some layers with output layer size equal to the number of classes i.e 41 in our case. Also, TensorFlow helps create the CNN architecture as it makes it easier to specify the layer, the required number of units in the layers, kernel size, stride, padding size, loss function, etc.

5.7. PyWavelets

Audio signal processing involves feature extraction and selection of the signal. And for that purpose we must understand the behaviour of the signal in frequency as well as in time domain. Thus, in order to study both of these characterstics simultaneously we perform wavelets transform to the signal.

PyWavelets, is an open-source python library for wavelet transform. It combines a simple high level interference with low level C and Cython performance. It is quite simple to use and get started by simply importing the library. Few of the main features of PyWavelets are:

1. 1D, 2D and nD Forward and Inverse Discrete wavelet transform (DWT and IDWT)
2. 1D, 2D and nD Multilevel DWT and IDWT

3. 1D, 2D and nD Stationary Wavelet Transform (Undecimated Wavelet Transform)
4. 1D and 2D Wavelet Packet decomposition and reconstruction
5. 1D Continuous Wavelet Transform
6. Computing Approximations of wavelet and scaling functions

In addition to these capabilities, Pywavelets additionally supports over 100 built-in wavelet filters, custom wavelets, single and double precision computations, real and complex calculations, and outputs that are compatible with Matlab Wavelet Toolbox.

The wavelet we choosed for our task is 'Morlet' Wavelet. The scale size is choosen as 100. The sampling rate of the audio signal is decreased to 500 to make the wavelet coefficent range smaller for computational reasons.

5.8. Pydub

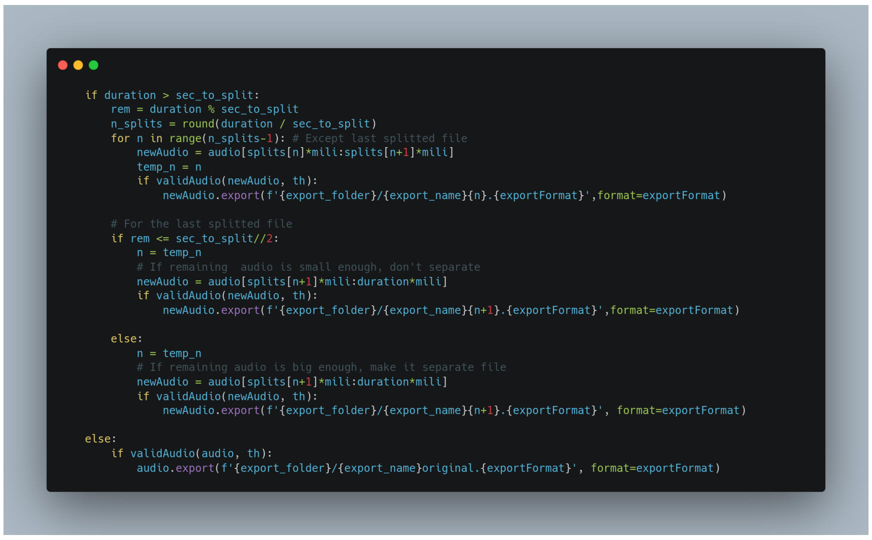
The available datasets were not error-free and had irregular durations. Therefore, we must divide the available datasets of larger duration in order to acquire the datasets with a specified time (10 seconds in our case). By splitting and segmenting the audio datasets, the incorrect noises that don't meet our requirement or the portions that donot contribute to our datasets were also removed.

Manually carrying out these tasks is tedious and inconvenient. An open source Python package called Pydub comes to the rescue here. We can only work with .wav (audio) files when using Pydub to work with audio files. It is not built in python library so first we have to install and can start working by just importing the library.

Following are some functionalities that can be performed by pydub:

1. Playing audio file.
2. Extracting information of file like length channels
3. Increase/Decrease volume of given .wav file.
4. Merging two or more audio files.
5. Exporting an audio file.
6. Splitting an audio file

We splitted the audio file into 10 second of each. The algorithm was shown in the methodology part. The python implementation snippet of it is shown below:



```python
if duration > sec_to_split:
    rem = duration % sec_to_split
    n_splits = round(duration / sec_to_split)
    for n in range(n_splits-1): # Except last splitted file
        newAudio = audio[splits[n]*mili:splits[n+1]*mili]
        temp_n = n
        if validAudio(newAudio, th):
            newAudio.export(f'{export_folder}/{export_name}{n}.{exportFormat}',format=exportFormat)

    # For the last splitted file
    if rem <= sec_to_split//2:
        n = temp_n
        # If remaining  audio is small enough, don't separate
        newAudio = audio[splits[n+1]*mili:duration*mili]
        if validAudio(newAudio, th):
            newAudio.export(f'{export_folder}/{export_name}{n+1}.{exportFormat}',format=exportFormat)

    else:
        n = temp_n
        # If remaining audio is big enough, make it separate file
        newAudio = audio[splits[n+1]*mili:duration*mili]
        if validAudio(newAudio, th):
            newAudio.export(f'{export_folder}/{export_name}{n+1}.{exportFormat}', format=exportFormat)

else:
    if validAudio(audio, th):
        audio.export(f'{export_folder}/{export_name}original.{exportFormat}', format=exportFormat)
```

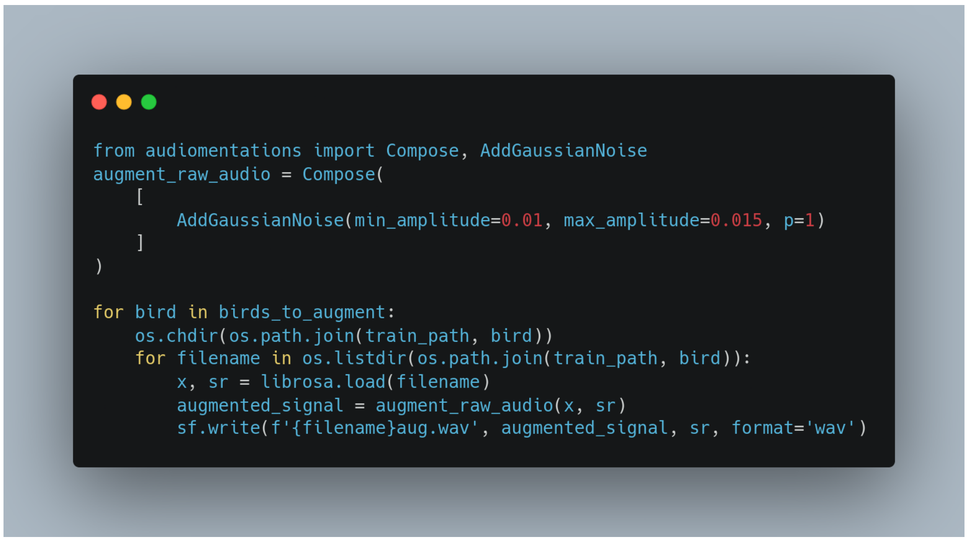
Figure 5.1: Audio File Splitting Python Implementation

5.9. Audiomentation

The available dataset might not be enough because machine learning requires a large number of datasets for model training, validation, and testing. So, we increase the number of dataset by agumenting the available dataset. Generating new files with similar properties is the called augmentation. The audio data agumentations involves addition of gaussian noise, time shifting, Pitch shifting, impulse response addition,etc.

We need an open source python package, called audiomentation that can perform audio data augumentation. Audiomentation is capable of performing these augumentations techniques. Apart from these we can add background noise, apply band pass filter, band stop filter, clipping distortion, FrequencyMask and many more.

As of now, we have implemented Gaussian Noise to the birds that have counts of less than 30.



```
from audiomentations import Compose, AddGaussianNoise
augment_raw_audio = Compose(
    [
        AddGaussianNoise(min_amplitude=0.01, max_amplitude=0.015, p=1)
    ]
)

for bird in birds_to_augment:
    os.chdir(os.path.join(train_path, bird))
    for filename in os.listdir(os.path.join(train_path, bird)):
        x, sr = librosa.load(filename)
        augmented_signal = augment_raw_audio(x, sr)
        sf.write(f'{filename}aug.wav', augmented_signal, sr, format='wav')
```

Figure 5.2: Data Augmentation using Gaussian Noise Snippet

5.10. Amazon Web Services(AWS)

Amazon web service is a service of Amazon that provides on-demand cloud computing platforms and APIs to individuals and companies on a metered pay-as-you-go basis. Rather than building an actual physical server, Amazon markets AWS to subscribers as a way of obtaining large-scale computing capacity more quickly and cheaply through cloud computing [18]. Through the AWS EC2 instance, we will get the virtual servers for deploying our model.

6. RESULTS AND ANALYSIS

6.1. Data Augmentation

The original audio waveform of the Cheer Pheasant bird is shown in the figure below:

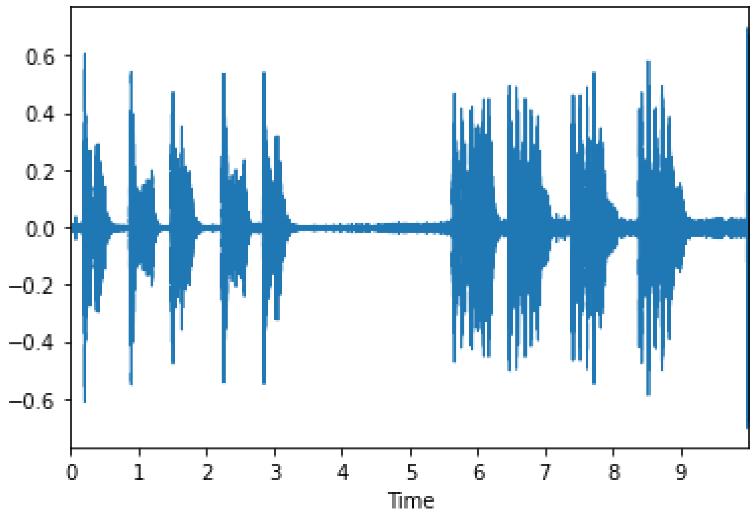


Figure 6.1: Cheer Pheasant Bird Audio Waveform

Using different data augmentation techniques, the following results show the change in the waveform.

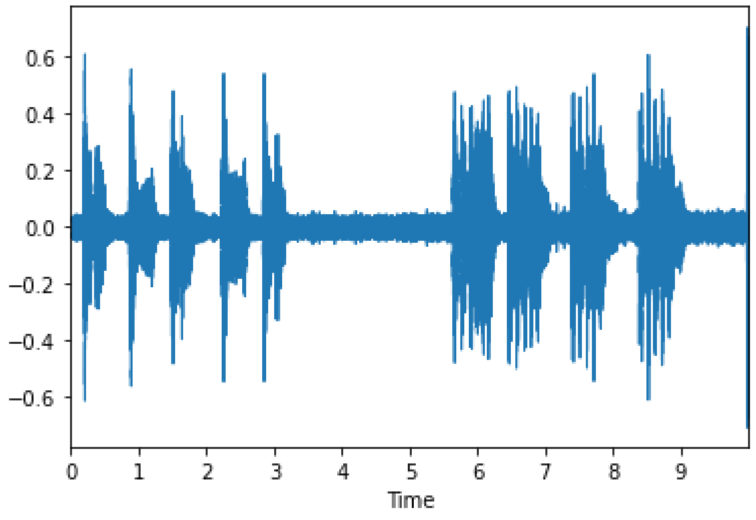


Figure 6.2: Audio Waveform with Gaussian Noise

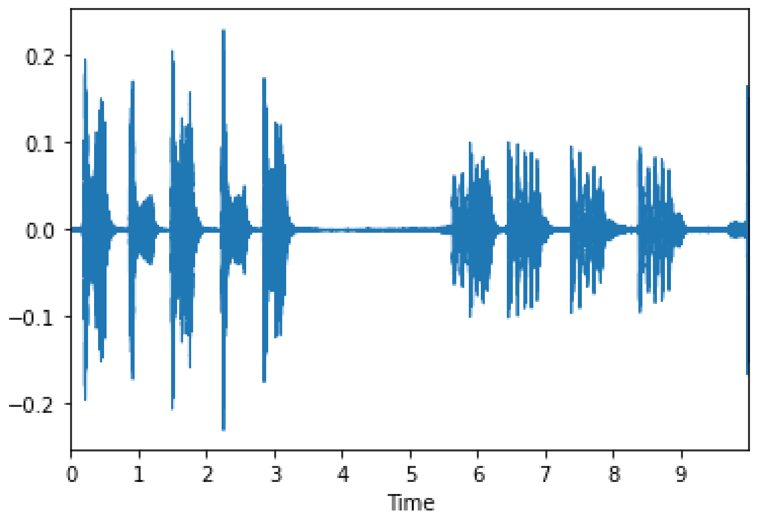


Figure 6.3: Audio Waveform with High Pass Filter

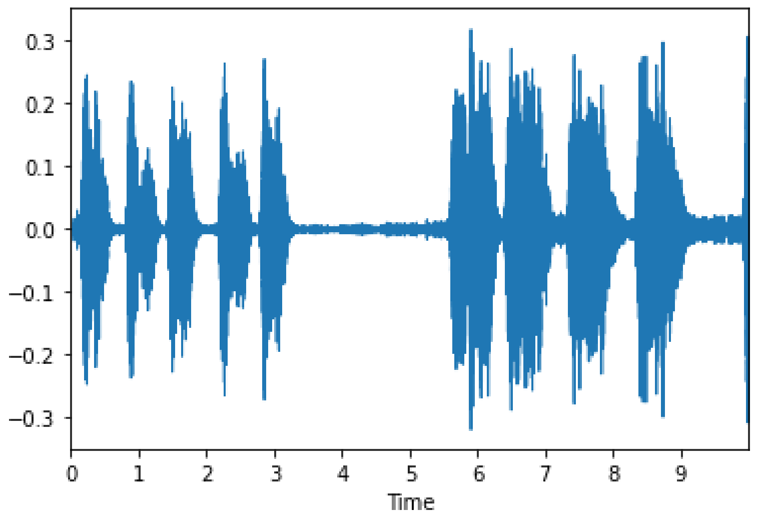


Figure 6.4: Audio Waveform with Pitch Shift

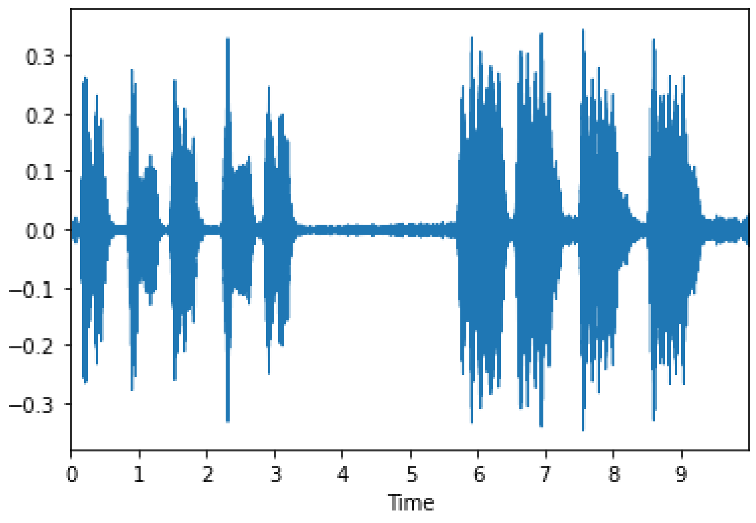


Figure 6.5: Audio Waveform with Time Stretch

6.2. Wavelet Transform

The scalogram obtained after wavelet transform on the Spiny Babbler audio data file is shown below. The x-axis represents the time and the y-axis represents the scales.

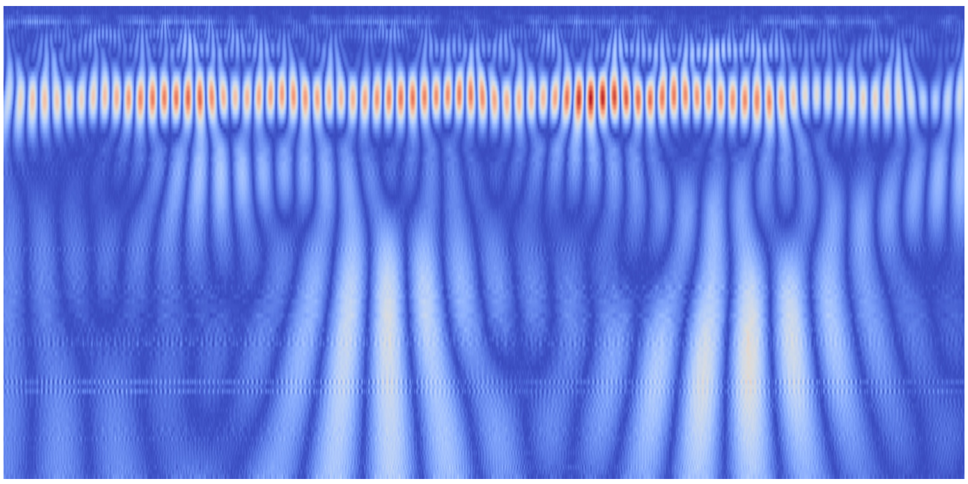
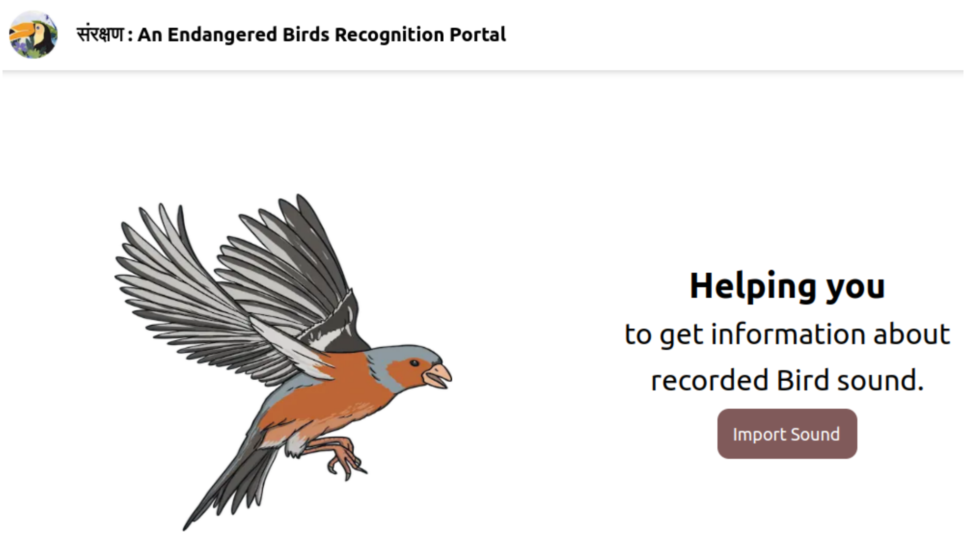


Figure 6.6: Scalogram of Spiny Babbler Audio File

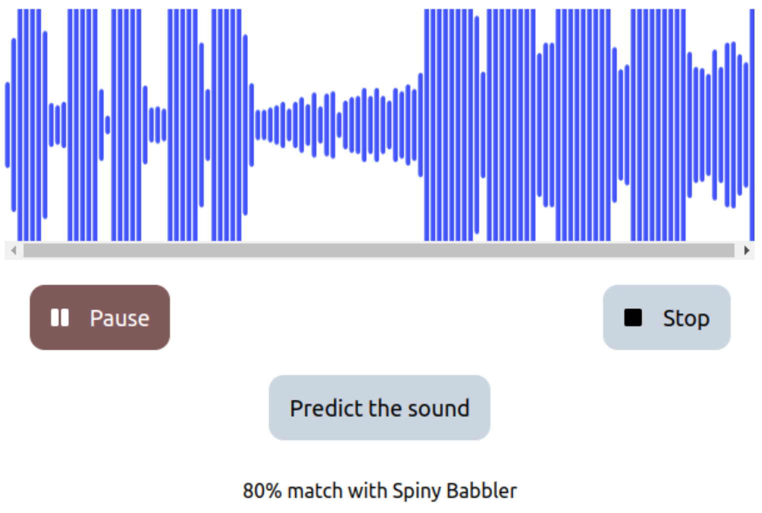
6.3. Website Interface

The website user interface is shown below where we can upload the audio file for prediction and the result is displayed with the accuracy percentage. The result is a dummy result as of now because we have not worked on the prediction algorithm yet.


संरक्षण : An Endangered Birds Recognition Portal

Helping you to get information about recorded Bird sound.

Figure 6.7: Website User Interface Home Page



80% match with Spiny Babbler

Figure 6.8: Website User Interface Prediction Page

7. REMAINING TASKS

The work on collecting the dataset is done but if we are able to find some source for new datasets, we will consider it as our remaining task of data collection.

Some major tasks that are remaining are:

(i) EfficientNet architecture implementation.

(ii) Parameter optimization using genetic algorithm.

(iii) Website development and deployment.

8. APPENDICES

APPENDIX A : Project Budget

ITEMS	EXPECTED COST(In Rs.)
AWS Computing Resources	3,000.00
Miscellaneous	1,000.00
Total	4,000.00

Table 8.1: Expected Budget

We are expecting the project to complete within a budget of Rs. 4,000

APPENDIX B : Project Schedule

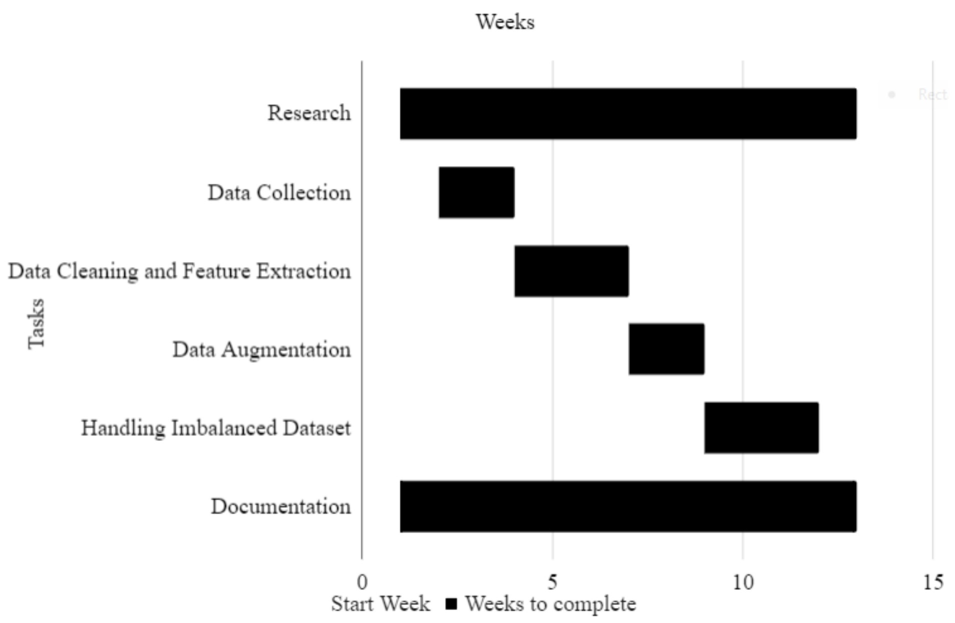
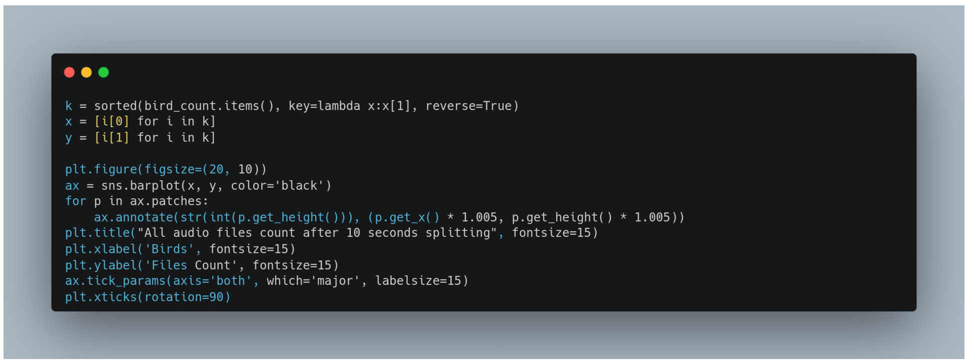


Table 8.2: Gantt Chart

APPENDIX C : Used Commands

• %%capture: Cell magic to suppress unwanted output
• cd data: Changes the current directory to the data directory
• mkdir fold: Creates a folder named fold
• ls: List files in current directory
• pip install package_name: To install python package package_name
• pip freeze –local > requirements.txt: To create requirements.txt file which consist of packages installed within the environment with it's version
• virtualenv env: Creating a new virtual environment env in the current directory

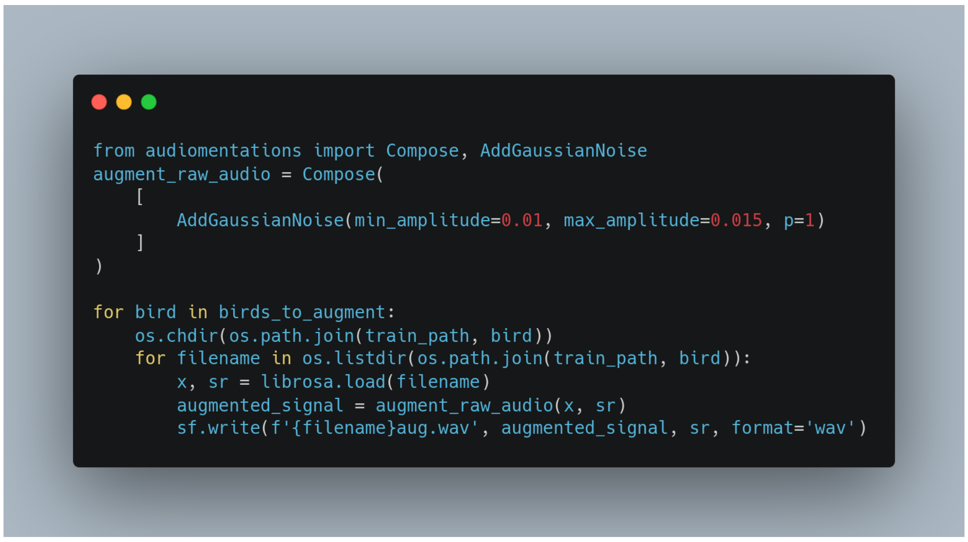
APPENDIX D : Code Snippets



```python
k = sorted(bird_count.items(), key=lambda x:x[1], reverse=True)
x = [i[0] for i in k]
y = [i[1] for i in k]

plt.figure(figsize=(20, 10))
ax = sns.barplot(x, y, color='black')
for p in ax.patches:
    ax.annotate(str(int(p.get_height())), (p.get_x() * 1.005, p.get_height() * 1.005))
plt.title("All audio files count after 10 seconds splitting", fontsize=15)
plt.xlabel('Birds', fontsize=15)
plt.ylabel('Files Count', fontsize=15)
ax.tick_params(axis='both', which='major', labelsize=15)
plt.xticks(rotation=90)
```

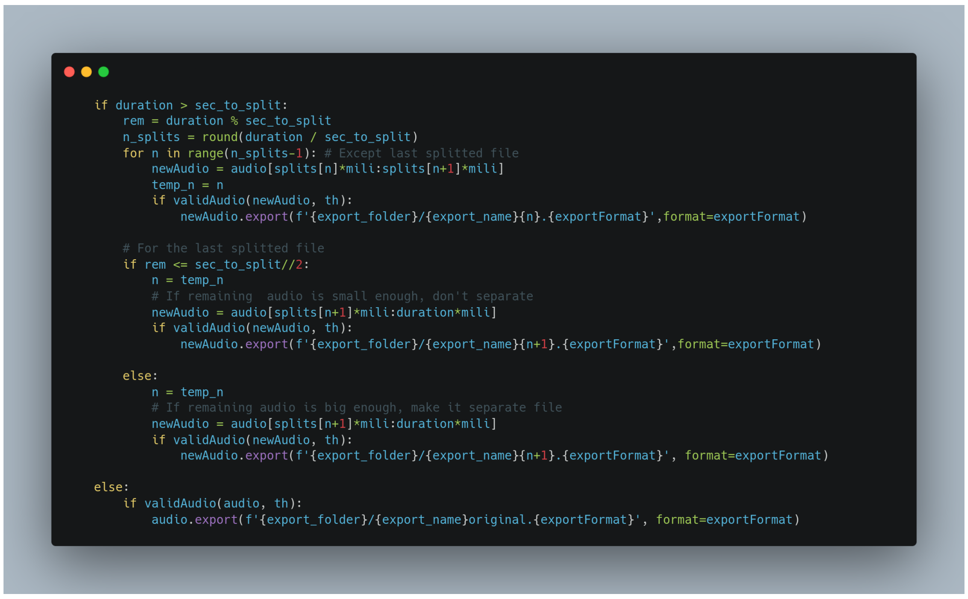
Figure 8.1: Matplotlib Plotting Code Snippet



```python
from audiomentations import Compose, AddGaussianNoise
augment_raw_audio = Compose(
    [
        AddGaussianNoise(min_amplitude=0.01, max_amplitude=0.015, p=1)
    ]
)

for bird in birds_to_augment:
    os.chdir(os.path.join(train_path, bird))
    for filename in os.listdir(os.path.join(train_path, bird)):
        x, sr = librosa.load(filename)
        augmented_signal = augment_raw_audio(x, sr)
        sf.write(f'{filename}aug.wav', augmented_signal, sr, format='wav')
```

Figure 8.2: Data Augmentation Code Snippet



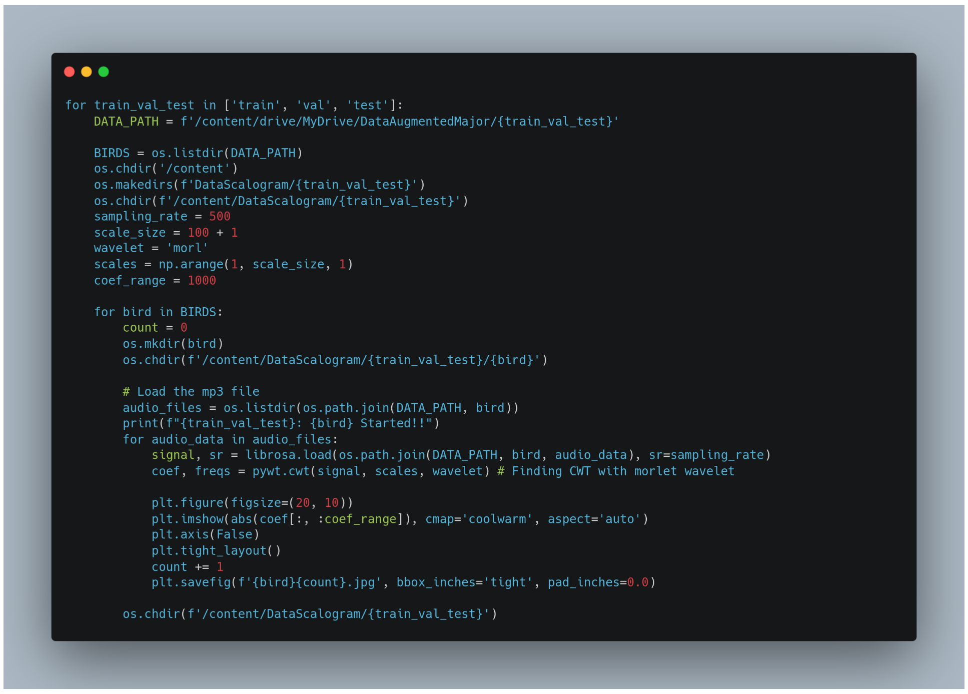
```python
if duration > sec_to_split:
    rem = duration % sec_to_split
    n_splits = round(duration / sec_to_split)
    for n in range(n_splits-1): # Except last splitted file
        newAudio = audio[splits[n]*mili:splits[n+1]*mili]
        temp_n = n
        if validAudio(newAudio, th):
            newAudio.export(f'{export_folder}/{export_name}{n}.{exportFormat}',format=exportFormat)

    # For the last splitted file
    if rem <= sec_to_split//2:
        n = temp_n
        # If remaining  audio is small enough, don't separate
        newAudio = audio[splits[n+1]*mili:duration*mili]
        if validAudio(newAudio, th):
            newAudio.export(f'{export_folder}/{export_name}{n+1}.{exportFormat}',format=exportFormat)

    else:
        n = temp_n
        # If remaining audio is big enough, make it separate file
        newAudio = audio[splits[n+1]*mili:duration*mili]
        if validAudio(newAudio, th):
            newAudio.export(f'{export_folder}/{export_name}{n+1}.{exportFormat}', format=exportFormat)

else:
    if validAudio(audio, th):
        audio.export(f'{export_folder}/{export_name}original.{exportFormat}', format=exportFormat)
```

Figure 8.3: Data Splitting Code Snippet



```python
for train_val_test in ['train', 'val', 'test']:
    DATA_PATH = f'/content/drive/MyDrive/DataAugmentedMajor/{train_val_test}'

    BIRDS = os.listdir(DATA_PATH)
    os.chdir('/content')
    os.makedirs(f'DataScalogram/{train_val_test}')
    os.chdir(f'/content/DataScalogram/{train_val_test}')
    sampling_rate = 500
    scale_size = 100 + 1
    wavelet = 'morl'
    scales = np.arange(1, scale_size, 1)
    coef_range = 1000

    for bird in BIRDS:
        count = 0
        os.mkdir(bird)
        os.chdir(f'/content/DataScalogram/{train_val_test}/{bird}')

        # Load the mp3 file
        audio_files = os.listdir(os.path.join(DATA_PATH, bird))
        print(f"{train_val_test}: {bird} Started!!")
        for audio_data in audio_files:
            signal, sr = librosa.load(os.path.join(DATA_PATH, bird, audio_data), sr=sampling_rate)
            coef, freqs = pywt.cwt(signal, scales, wavelet) # Finding CWT with morlet wavelet

            plt.figure(figsize=(20, 10))
            plt.imshow(abs(coef[:, :coef_range]), cmap='coolwarm', aspect='auto')
            plt.axis(False)
            plt.tight_layout()
            count += 1
            plt.savefig(f'{bird}{count}.jpg', bbox_inches='tight', pad_inches=0.0)

        os.chdir(f'/content/DataScalogram/{train_val_test}')
```

Figure 8.4: Scalogram Generation Code Snippet

References

[1] H. Ritchie and M. Roser, “Biodiversity,” *Our World in Data*, 2021, https://ourworldindata.org/biodiversity.

[2] “Birdlife data zone,” *Datazone.birdlife.org*, 2022, http://datazone.birdlife.org/country/nepal.

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*. IEEE, Aug. 2017.

[4] “Data augmentation,” https://www.tensorflow.org/tutorials/images/data_augmentation, accessed 2022.

[5] M. Arslan, M. Guzel, M. Demirci, and S. Ozdemir, “Smote and gaussian noise based sensor data augmentation,” in *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2019, pp. 1–5.

[6] “Data augmentation for audio,” https://medium.com/@makcedward/data-augmentation-for-audio-76912b01fdf6, accessed Jun 1, 2019.

[7] A. Marini, J. Facon, and A. L. Koerich, “Bird species classification based on color features,” in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 4336–4341.

[8] M. V. Conde, K. Shubham, P. Agnihotri, N. D. Movva, and S. Bessenyei, “Weakly-supervised classification and detection of bird sounds in the wild. a birdclef 2021 solution,” *arXiv preprint arXiv:2107.04878*, 2021.

[9] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, “Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm,” *arXiv preprint arXiv:2006.12703*, 2020.

[10] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 2902–2911.

[11] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, “Optimizing deep learning hyper-parameters through an evolutionary algorithm,” in *Proceedings of the workshop on machine learning in high-performance computing environments*, 2015, pp. 1–5.

[12] J.-H. Han, D.-J. Choi, S.-U. Park, and S.-K. Hong, "Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network," *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 721–726, 2020.

[13] S. Lee, J. Kim, H. Kang, D.-Y. Kang, and J. Park, "Genetic algorithm based deep learning neural network structure and hyperparameter optimization," *Applied Sciences*, vol. 11, no. 2, p. 744, 2021.

[14] "Github - realzza/xenopy: Xenopy: Python wrapper for xeno-canto api 2.0. supports multiprocessing." https://github.com/realzza/xenopy, accessed: 09- Jun-2022.

[15] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[16] "Librosa.org, 2022," https://librosa.org/doc/latest/index.html, accessed: 09- Jun-2022.

[17] "TensorFlow. 2021," https://www.tensorflow.org/, accessed 14 December 2021.

[18] "Amazon Web Services," https://aws.amazon.com/, accessed: 17- Feb- 2022.