Pontificia Universidad JAVERIANA Cali

Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado
Ingeniería Electrónica

Fecha: 19 – 07 - 2023

Autores: Juan De Valdenebro y Michael Hernandez

Nombre del Proyecto de Grado: Performance evaluation of multi-label classification models for the automated classification of anuran calls in audio recordings

Director: Hernán Darío Benítez y Juan Sebastián Ulloa

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuaran, como consta en el Acta de Calificación correspondiente.

Hernan Dario Benitez Restrepo

Firma de Director(a) del Proyecto de Grado

Nota de Aceptación

Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero Electrónico.

Camilo Rocha

Dr. Hernán Camilo Rocha Niño
Decano de la Facultad de Ingeniería

Dr. LUIS EDUARDO TOBON LLANO
Director Carrera Ingeniería Electrónica.

Hernan Dario Benitez Restrepo

Dr. HERNAN DARIO BENITEZ
Director(a) Trabajo

Dr. LUIS EDUARDO TOBON LLANO
Jurado 1

Dr. JULIAN GIL GONZALEZ
Jurado 2

Pontificia Universidad
JAVERIANA
Cali

Facultad de Ingeniería
y Ciencias
Ingeniería Electrónica

Undergraduate Project

Performance evaluation of multi-label classification models for the automated classification of anuran calls in audio recordings

Michael Hernández Mera
Juan Sebastián De Valdenebro Herrera

Director
Dr. Hernán Darío Benítez

Codirector
Dr. Juan Sebastián Ulloa

July 22, 2023

Santiago de Cali, July 22, 2023

Señores
Pontificia Universidad Javeriana – Cali
Dr. Hernán Camilo Rocha Niño
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

Por medio de la presente nos permitimos presentarle el Trabajo de Grado titulado "Performance evaluation of multi-label classification models for the automated classification of anuran calls in audio recordings".

Esperamos que este trabajo reúna todos los requisitos académicos, cumpla el propósito para el cual fue creado y sirva de apoyo para futuros proyectos relacionados con la materia.

Atentamente,

Michael Hernandez

Michael Hernández Mera

Juan De Valdenebro

Juan Sebastián De Valdenebro Herrera

Santiago de Cali, July 22, 2023

Señores
Pontificia Universidad Javeriana – Cali
Dr. Hernán Camilo Rocha Niño
Decano
Facultad de Ingeniería y Ciencias
Ciudad

Cordial Saludo.

Certificamos que el presente Trabajo de Grado titulado "Performance evaluation of multi-label classification models for the automated classification of anuran calls in audio recordings", realizado por Michael Hernández Mera y Juan Sebastián De Valdenebro Herrera, estudiantes de Ingeniería Electrónica, se encuentra terminado y puede ser presentado para su sustentación.

Atentamente,

Hernan Dario Benitez Restrepo

Dr. Hernán Darío Benítez
Director Trabajo de Grado

Dr. Juan Sebastián Ulloa
Co-Director Trabajo de Grado

Acknowledgements

We, Juan Sebastian De Valdenebro Herrera and Michael Hernandez Mera, would like to express our sincere gratitude and appreciation for the exceptional support and guidance you provided throughout our project. Your invaluable contributions have played a pivotal role in its successful completion, and we are extremely grateful for your unwavering commitment and dedication.

First, we sincerely appreciate Hernan Dario Benitez and Juan Sebastian Ulloa, our esteemed project directors, for their outstanding mentorship and supervision. Your profound knowledge, insightful feedback, and continuous encouragement have been instrumental in shaping our project's trajectory and ensuring its fruition. We are truly fortunate to have had the privilege of working under your guidance.

We would also like to extend our gratitude to the Humbolt Institute for their support and resources. The Institute's commitment to fostering excellence in research and innovation has provided us with an enriching environment to develop our project.

Furthermore, we would like to express our appreciation to the Engineering Faculty at Javeriana Cali University for their guidance and academic support. The faculty's expertise and commitment to nurturing young minds have been essential in our project's development. We are grateful for the opportunity to learn from distinguished professors and researchers who have helped shape our academic journey.

Additionally, we would like to extend our deepest gratitude to our respective families for their unwavering love, support, and encouragement throughout this project. To Sebastian's family, including his father Manuel De Valdenebro, mother Luz Herrera, and sister Sofia De Valdenebro Herrera, we express our heartfelt appreciation for their constant belief in Sebastian's abilities and unwavering support.

To Michael's family, including his father Jose Arley Hernandez, mother Luz Dary Mera Astudillo, and brother Jhonier Hernandez Mera, we extend our sincere gratitude for their endless encouragement and understanding. Their unwavering support has been a constant source of strength, and we are truly fortunate to have such a loving and supportive family.

We would also like to express our gratitude to all the individuals who directly or indirectly contributed to the success of this project. Their assistance, advice, and feedback have been invaluable in shaping our research and development efforts.

Once again, we extend our heartfelt appreciation to each and every person who has supported us throughout this project. Your belief in our abilities and unwavering support have been the driving force behind our success. We are truly grateful for your contributions, and we look forward to the opportunities that lie ahead.

Thank you once again for your continued support.

Glossary

Acronyms:

- CNN: Convolutional neural network.
- F1-score: A performance metric that combines precision and recall to evaluate the accuracy of a machine learning model.
- Hz: Hertz.

Terms:

- Amphibian: A type of cold-blooded vertebrate that includes frogs, toads, and salamanders.
- Amplitude: The measure of the strength of a sound wave, which determines the loudness of the sound.
- Frequency: The number of cycles per second of a sound wave, measured in Hz.
- Mel spectrogram: A type of spectrogram that is commonly used in sound analysis and is based on the human auditory system's frequency response.
- Wavelength: The distance between two consecutive peaks or troughs of a sound wave, which determines the pitch of the sound.
- Data augmentation: The process of generating new data samples by applying transformations to existing data.
- Ecoacoustics: The ecological investigation and interpretation of environmental sound.
- Multi-label classification: A type of classification problem where each sample can belong to multiple classes.
- Neural network: A type of machine learning model that is inspired by the structure and function of the human brain.
- Preprocessing: The process of preparing data for machine learning by cleaning, transforming, and normalizing it.
- Spectrogram: A visual representation of the frequency content of a sound signal over time.
- Transfer learning: A technique where a pre-trained machine learning model is used as a starting point for a new model.

Resumen

Este proyecto de grado tiene como objetivo utilizar modelos de redes neuronales convolucionales (CNN) preentrenados para identificar tres especies diferentes de Anuros por sonido en una representación de tiempo-frecuencia. Las especies seleccionadas son: *Boana albopunctata, Physalaemus cuvieri, y Boana lundii.* Además, se analizó el rendimiento de diversos modelos y técnicas de aumentación de datos, para lograr una correcta clasificación multi-etiqueta, y las mejores prácticas para el procesamiento de audio, clasificación y automatización para las señales de sonido de los anfibios y también se discuten algunas referencias a las limitaciones asociadas con el monitoreo acústico de los anfibios y otras especies.

El diseño metodológico del proyecto se dividió en cuatro etapas: preprocesamiento, aumento de datos, entrenamiento del modelo y evaluación del rendimiento de los modelos entrenados. El núcleo del proyecto se desarrolló en Python, para la etapa de preprocesamiento de datos en este proyecto se diseñó un *"pipeline"* para procesar los datos crudos proporcionados por el Instituto Humboldt y consistió en recortar los archivos de audio, generar espectrogramas, y fusionarlos con las anotaciones para devolver un conjunto de datos bien estructurados para el entrenamiento, en la etapa de aumento de datos las técnicas utilizadas fueron técnicas de estiramiento de tiempo, enmascaramiento de tiempo y enmascaramiento de frecuencia, por último, la fase de evaluación del rendimiento se realizó extrayendo de los modelos entrenados (MobileNet, DenseNet121, InceptionV3 y Resnet50) la métrica de desempeño F1 utilizando un 30% del conjunto de datos no aumentado aislado del proceso de entrenamiento y comparando el rendimiento de cada modelo.

Se realizaron tres experimentos, variando los hiperparámetros y la arquitectura, y utilizando diferentes conjuntos de datos. Se seleccionaron los mejores modelos en función de su rendimiento. Los mejores modelos (MobileNet con 1 FC Layer, DenseNet con 1 FC Layer, InceptionV3 con 2 FC Layer) alcanzó un desempeño F1 medio del 81% para la clasificación multi-etiqueta de las tres diferentes especies mencionadas anteriormente.

Abstract

This undergraduate project aims to use pre-trained convolutional neural network (CNN) models to identify three different Anuran species by sound in a time-frequency representation. The species include *Boana albopunctata, Physalaemus cuvieri, and Boana lundii*, This work also analyzes the performance of the models, to achieve multi-label classification. The theory treated in this work includes sound theory, ecoacoustics, amphibian soundscape, and best practices for audio processing, classification, and automation for amphibian sound signals, and some references to limitations associated with acoustic monitoring of amphibians and other species are also discussed.

The methodology designs for the project were divided into four stages: pre-processing, data augmentation, model training, and performance evaluation. The core of the project was developed in Python, for the data pre-processing stage in this project was designed a pipeline for raw data provided by the Humboldt Institute and involved trimming audio files, generating spectrograms, and merging them with the annotation files, to return a well-structured dataset for training. In the data augmentation stage, the techniques used were time stretching, time masking, and frequency masking techniques, while model training involved tuning hyperparameters such as learning rate, batch size, and the number of epochs, as well as using regularization techniques and adding fully connected layers, finally the performance evaluation stage was performed extracting from the trained models (MobileNet, DenseNet121, Resnet50, InceptionV3), the F1-score metric using a 30 percent of the not augmented dataset isolated from the training process and comparing the model's performance.

Three experiments were conducted, varying hyperparameters and architecture, using different datasets. The best models were selected based on their performance, the best models (MobileNet with 1 FC Layer, DenseNet with 1 FC Layer, InceptionV3 with 2 FC Layer) achieved an average F1-Score of 81% for multi-label classification of the three different anuran calls specified earlier.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Preservation of biodiversity is a crucial issue in today's world, and acoustic monitoring has become an essential tool for studying and conserving species, which health is intrinsically connected with the ecosystem they are living in. Therefore, a technological approach has surged based on implementing AI for acoustic monitoring, improving and automating the classification of these species [1] [2], delivering an advancement in the data processing stage of the acoustic classification over specific species. Consequently, the use of deep learning for anuran call classification is a promising area of research[3], but there are still significant knowledge gaps and one of the main challenges is the limited amount of available data and its quality[3]. Collecting and labeling anuran call data with accuracy can be difficult and time-consuming, limiting the size and diversity of datasets that can be used to train deep learning models[4]. Additionally, environmental factors, such as background noise and temperature, can affect the quality of anuran call recordings, preventing deep learning models to accurately classify calls. Despite these challenges, these models have the potential to be a powerful tool for anuran call classification[5]. As the amount of available data increases and models become more sophisticated, deep learning is likely to become more accurate and reliable.

The purpose of this study is to demonstrate the applicability of AI models in the acoustic monitoring data classification process through the experimentation and analysis of pre-trained Convolutional Neural Networks (CNNs) in the context of transfer learning and multi-label classification to identify by their sound in a time-frequency representation (Spectrograms) of three specific anuran species, (*Boana albopunctata, Physalaemus cuvieri, and Boana lundii*), present on the site denoted as INCT41, located in the Bifurcação locality on the Cerrado biome in Brazil. In order to accomplish this, the following methodology was divided into these four stages: pre-processing, data augmentation, model training, and performance evaluation; Pre-processing techniques were used to prepare the raw audio data and annotations recorded by the Humboldt Institute on the INCT41 site. Subsequently, data augmentation techniques, Time Stretching, Time Masking, and Frequency Masking, were implemented to enhance the diversity of the training dataset.

The results obtained in this study will contribute to the understanding and effectiveness of using transfer learning and augmentation techniques in the context of acoustic species identification through multi-label classification. This research highlights the potential of deep learning techniques in acoustic monitoring for species classification.

This document presents the following structure: Chapter 1: Introduction – Provides an overview of the thesis and its objectives. Chapter 2: Problem description – Describes the problem that

the thesis aims to address, which is the need for accurate and efficient automated sound multi-label classification for acoustic monitoring of specific anuran species in Cerrado biome. Chapter 3: Justification – Explains the importance of the problem and why it is necessary to develop a solution. Chapter 4: Objectives – Outlines the specific objectives of the thesis, which include the usage of transfer learning and the evaluation of multi-label classification models for acoustic monitoring. Chapter 5: Theoretical framework – Describe a theoretical background for the thesis, including relevant subject areas and methods for sound classification. Chapter 6: Methodology and resources – Show the methodology used in the thesis, including the stages of the project and the resources and libraries used. Chapter 7: Results and discussion – Presents the results of the experiments and discusses their implications for acoustic monitoring. Chapter 8: Conclusions and recommendations – Summarizes the main findings of the thesis and provides recommendations for future research and development in the field of acoustic monitoring.

Chapter 2

Problem description

2.1 Problem statement

Climate change (CC) refers to the alterations in average weather conditions that can be identified over time periods of decades or longer. These changes may be the result of natural processes, like alterations in the Earth's orbit around the sun, or they may be induced by human activities, such as the burning of fossil fuels [6]. It is now generally accepted that CC is one of the most serious threats to biodiversity. The potential impacts of CC on biodiversity have been recognized for some time, and this has led to a concerted effort to monitor and predict the long-term biological responses of species in a changing climate [7]. Anuran population declines are far more severe than those of other vertebrate groups, with 30 percent of all species listed as threatened by the International Union for Conservation of Nature [8].

A multi-species classification system is proposed as part of a complete solution that is being developed by the Humboldt Institute of biological resources. The solution proposes the usage of a pre-processed dataset and the analysis of the performance of pre-trained convolutional neural network-based models to achieve an efficient multi-label classification. The use of pre-trained CNNs based models is known as transfer learning, this method (transfer learning) had shown its effectiveness in other studies when it was implemented in passive acoustic monitoring (PAM), obtaining results up to 82 percent on the F1 performance scale [9], which is defined as: "The combination between the precision and recall metrics into a single metric, resulting in the harmonic mean of precision and recall" [10], this research determined that the ResNet101V2 and ResNet152V2 models performed best on a binary classification problem on the data set.

To sum up, this undergraduate project will evaluate how to set transfer learning in multi-label classification problems using CNN based models and select which of those is better for the needed application, according to its performance. Then the research question would be: ¿How to evaluate the performance of pre-trained CNN models over a multi-label dataset using transfer learning to perform a proper classification of specific anuran species(*Boana albopunctata, Physalaemus cuvieri, and Boana lundii*) using the audios and annotations in a time-frequency representation (spectrograms) recorded by the Humboldt Institute on a specific site denoted INCT41 which is located in the Bifurcação locality on the Carrado biome, Brazil?

This answer to this question will provide relevant information to aid in the analysis of guidelines for the preservation of these species.

Chapter 3

Justification

The preservation of the different species existing on the planet is a serious and immense problem in the real world. This project seeks to contribute to the process of understanding how the effects of climate change on anurans have altered their behavior, thus generating adequate preservation policies. The immense biodiversity present in the Cerrado biome makes it relevant to carry out studies on the behavior of animals, more specifically on the sounds that some anurans reproduce, in order to identify patterns and traits that have been affected by climate change.

"In recent decades, advances in technologies that seek to automate the monitoring of wildlife using remote sensors and automated acoustic identification of species are transforming the way biologists study ecosystems. For this purpose, various pattern recognition methods have been suggested to investigate the sound production of birds, insects, and bats among others" [11]. However, a robust machine-learning technique to recognize frog and toad calls has still not been found. In anurans, the main goal of vocalization is an advertisement that presents unique acoustic properties per species. Therefore, their calls can be used as an efficient parameter for taxonomic classification and survey.

There is no doubt of the effort made in the automatic acoustic recognition field to enable a reliable classification of species. However, there are few studies in the literature that have been focused on anurans and previous work limits the study to a small number of species, so an improvement in the state of the art is necessary to identify successfully a larger dataset [12].

"Sound classification is the process of identifying and categorizing sounds based on certain characteristics. This process typically involves four steps: segmentation, feature extraction, feature selection, and classification. Researchers use machine learning techniques, such as deep learning, genetic algorithms, and fuzzy logic, along with methods like the Gaussian mixture model, Hidden Markov Model, and deep neural networks, to automatically classify sounds. These methods have been successful in identifying and classifying various species based on their sounds, particularly in the field of bioacoustics, which focuses on the sounds of insects, bats, birds, and frogs. However, challenges can arise when classifying species that produce similar sounds or when the number of species being classified increases, leading to decreases in classification accuracy. Researchers are therefore working on developing new methods to overcome these limitations". [13]

Consequently, this project seeks the implementation of a solution for specific species classification (*Boana albopunctata, Physalaemus cuvieri, and Boana lundii*) based on the analysis of audio signals generated by Anurans (raw data provided by the Humboldt research center on the INCT41 site located in the Bifurcação locality on the Cerrado biome, Brazil) through ML, specifically using pre-trained CNNs (Convolutional Neural Networks) models and the determination of which model performs best will help to analyze and synthesize really important information for the Humboldt

institute about the behavior of these species and which policies can be applied to preserve the fauna of the region.

Chapter 4

Objectives

4.1 General objective

To evaluate the performance of pre-trained CNN-based models in the detection and classification of calls from the anurans *Boana albopunctata, Physalaemus cuvieri and Boana lundii.*

4.2 Specific objectives

1. To pre-process the audio and annotations given by the Humboldt institute using pandas, NumPy, librosa and maad libraries, in order to obtain a dataset readable by the models to train.

2. To train the selected machine learning models (Densenet121, MobileNet, Resnet50 and InceptionV3) over the generated datasets, to classify the three species (*Boana albopunctata, Physalaemus cuvieri and Boana lundii*) of anuran calls spectrograms using multi-label classification and iterating over specific parameters like adding fully connected layers, Cross Validation, learning rate, L2 regularization, to achieve different performance metrics to evaluate each trained model.

3. To evaluate multi-label classification models over testing dataset, extracting the F1 score per label and comparing them in order to determine which model performs best.

Chapter 5

Theoretical framework

5.1 Subject areas

- Environmental science - Acoustics - Ecoacoustics
- Ecology - Ecoacustics - Amphibian vocalizations
- Acoustics - Signal theory
- Statistics - Artificial intelligence -Automated sound classification
- Ecoacoustics - Acoustic monitoring
- Mathematics - Physics - Electronic engineering - Signal processing
- Mathematics - Statistics - Statistical analysis
- Statistics - Data Science - Classification algorithms
- Statistics - Artificial intelligence - Deep learning
- Ecology - Conservation

5.2 Theoretical framework

5.2.1 Ecoacustic and sound theory

The sound theory is the study of the properties of sound waves and how they are detected, recorded, and analyzed. Sound waves are produced by a vibrating object, which causes pressure waves to propagate through a medium, such as air or water. The properties of sound waves include amplitude, wavelength, and frequency. Amplitude is the measure of the strength of the sound wave, and it determines the loudness of the sound. Wavelength is the distance between two consecutive peaks or troughs of the sound wave, and it determines the pitch of the sound. Frequency is the number of cycles per second, and it is measured in hertz (Hz).

Sound waves can be detected and recorded using microphones or hydrophones, which transduce the sound waves into electrical signals. These signals can then be digitized and stored as digital sound files. The frequency sensitivity of different transducers is also discussed, as well as the audible range of human hearing and the use of ultrasonic and infrasonic detectors. Finally, the sound

signal theory emphasizes the importance of signal processing and acoustic analysis in extracting ecological information from sound recordings. Automated methods are suggested as a 'first pass' to identify possible relevant sounds and reduce the size of the dataset that must subsequently be manually checked. Statistical analysis is also important in extracting ecological information from sound recordings.[14] Also an important concept related to this project is the eco acoustics as a new science, eco acoustics is the ecological investigation and interpretation of environmental sound. It is an emerging interdisciplinary science investigating natural and anthropogenic sounds and their relationships with the environment over multiple scales of time and space. Eco acoustics includes the realms of ecological investigation including populations, communities, ecosystems, landscapes, and biotic regions of the Earth system. Studies of Eco-acoustics in these realms can include terrestrial, freshwater, and marine systems. Eco acoustics thus extends the scope of acoustic investigations, including Bioacoustic and soundscape ecology. Eco acoustics studies involve the investigation of sound as a subject to understand the properties of sound, its evolution, and its function in the environment. Eco acoustics also considers sound as an ecological attribute that can be utilized to investigate a broad array of applications including the diversity, abundance, behavior, and dynamics of animals in the environment.[15]. Now is important to emphasize about the amphibians soundscape and how amphibians play a crucial role in various ecosystems, exhibiting diverse characteristics that set them apart from other animals. Among these unique traits are their distinct vocalizations, which they use for communication, territorial defense, and mating. These vocalizations vary considerably in complexity, duration, and frequency and are typically species-specific. The study of amphibian vocalizations is useful in examining population dynamics, species richness, and community structures in aquatic and terrestrial environments. However, amphibian populations are currently under threat globally from habitat destruction, climate change, and diseases, underscoring the need for more research and monitoring to aid conservation efforts. Acoustic monitoring is a valuable tool in the study of amphibian vocalizations, with advances in automated detection and classification methods enabling the efficient analysis of significant datasets.[14] Here are the best practices for audio processing, classification, and automation for amphibian sound signals:

- Develop a comprehensive validated call library of species of interest: This is important to train algorithms to detect and classify unknown sounds in new recordings. The call library should ideally have data recorded in a range of ambient sound situations.

- Use a combination of automated processing and manual validation: While automated classification methods are available, their accuracy is rarely high enough to enable fully-automated analysis. Therefore, a combination of automated processing and manual validation is recommended.

- Extract features from a sound describing its spectral and temporal characteristics: This includes features such as call duration, peak frequency, and frequency range. Feature extraction methods can be sensitive to factors such as recording quality and ambient noise levels.

- Use unsupervised feature extraction, dynamic time warping-based feature representations, and deep convolutional neural networks: These methods can learn discriminating representations

directly from spectrogram data, potentially improving their robustness for analysis of noisy, heterogeneous acoustic monitoring datasets.

- Standardize protocols in advance to minimize errors associated with skill level: If multiple analysts will be manually classifying data, it is important to standardize protocols in advance to minimize errors associated with skill level.

- Use high-quality reference materials for manual analysis: This includes books, published scientific literature, and published call libraries. Such data are deficient for many taxonomic groups, habitats, and regions.

- Cross-check automated results manually and/or with other software tools: All automated call ID tools are subject to error, which is often inadequately reported for proprietary software. Therefore, it is recommended to cross-check their outputs manually and/or with other software tools.

- Regularly cross-check and report error rates: Any robust analysis pipeline for processing acoustic data will involve regular cross-checking and reporting of error rates.

- Develop project-specific tools suited to your own data: User-friendly software that enables ecologists and conservation practitioners to develop project-specific tools suited to their own data would further improve the practicality of acoustic monitoring methods in ecology and conservation. Overall, these best practices emphasize the importance of using a combination of automated processing and manual validation, extracting relevant features from sound recordings, using high-quality reference materials, and regularly cross-checking and reporting error rates. Additionally, developing project-specific tools suited to your own data can improve the practicality of acoustic monitoring methods in ecology and conservation. [14]

Also exist some real limitations associated with acoustic monitoring of amphibians and other species, like:

- Lack of classifiers available for highly biodiverse tropical biomes: There is a lack of classifiers available for highly biodiverse tropical biomes, which makes it challenging to detect and classify unknown sounds in new recordings. This is coupled with significant analytical challenges associated with acoustic monitoring in very biodiverse areas, such as high degrees of interspecific call similarity.[16]

- Biases in the availability of species call libraries: There are biases in the availability of species call libraries with which to train classifiers. This can limit the accuracy of automated classification methods and make it difficult to detect and classify unknown sounds in new recordings.[17]

- Environmental factors: Environmental factors such as humidity and noise levels can affect the quality of recordings and make analysis more challenging. Humidity can severely damage microphones and recorders, while noise can mask target sounds and interfere with the statistical analysis of soundscapes. [18]

- Need for user-friendly software: Machine learning methods are often complex and require statistical training, which can limit their accessibility to non-statistically trained researchers. Therefore, there is a need for user-friendly software that enables ecologists and conservation practitioners to develop project-specific tools suited to their own data. [19]

- Lack of robust and transparent automated tools: There is a need for the development of robust and transparent automated tools with clearly reported methods and limitations for analysis. Currently, the accuracy and transparency of such tools are still rarely high enough to allow fully automated analysis. [20]

- Taxonomic and environmental biases: There are major taxonomic and environmental biases in the availability of automated sound identification systems. Several exist for bats and cetaceans, mainly covering temperate regions, while far fewer are available for other taxonomic groups such as invertebrates and fish.

- Environmental change: Many tropical ecosystems are experiencing high rates of environmental change, which can limit the effectiveness of acoustic monitoring programs.[18]

- Equipment damage: Wildlife damage to equipment can be a problem in highly biodiverse areas, such as in the tropics. In marine environments, biofouling or attachment of sessile organisms can also be a problem.[14]

  Overall, these limitations highlight the need for improvements in automated signal detection and classification, the development of user-friendly software, and the need for robust and transparent automated tools with clearly reported methods and limitations for analysis. Additionally, environmental factors, taxonomic and environmental biases, and equipment damage can all impact the feasibility and success of acoustic monitoring programs.[16]

5.2.2 Machine learning

5.2.2.1 Main concepts

- Neural Networks:

  The concept of neural networks has existed since the 1950s, but only in recent years with advances in computational power has their adoption become widespread. The most basic neural networks are feed-forward neural networks or multi-layer perceptron (MLP). MLPs are made up of many "neurons" or nodes that take an input multiplied by a weight, a bias, and sum these inputs. The result is then passed through an activation function to produce an output that is sent to the next layer of the network. The weights and biases are the parameters that are "learned" through the training process.[21]

- Convolutional Neural Networks:

  Convolutional neural networks (CNNs) are a type of deep learning algorithm that has been used in a variety of real-world applications. CNNs can be trained to classify images, detect ob-

jects in an image, and even predict the next word in a sentence with incredible accuracy. CNNs can also be applied to more complex tasks such as natural language processing (NLP).[22]

- Hyperparameters: Are the settings that control the learning process of a CNN. They are typically set before training begins, and their values cannot be learned from the data. Some examples of hyperparameters in CNNs include the number of layers, the number of filters per layer, the size of the kernels, and the learning rate.[23]

- Parameters: Are the values that are learned by a CNN during training. They represent the weights and biases of the network, and they are used to make predictions. Some examples of parameters in CNNs include the weights of the convolutional kernels, and the biases of the neurons in the fully connected layers.[24]

- Cross-Validation Training

  Cross-validation training is a technique for evaluating the performance of a CNN model on unseen data. The model is trained on a subset of the data, and then its performance is evaluated on a separate subset of the data. This process is repeated multiple times, creating different subsets named folds, and the average performance of the model is used as an estimate of its true performance.[25]

- Fold

  A fold is a subset of the dataset that is used for training or testing. The number of folds is typically a power of 2, such as 2, 4, 8, 16, etc. The folds are created by randomly partitioning the dataset. The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, with each fold used as the test set once. The results from all k iterations are averaged to get an estimate of the model's performance.[25]

- Transfer Learning:

  Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another.[26]

5.2.2.2 Hyperparameters

- Learning Rate:

  The learning rate is an important hyperparameter in deep learning that controls the adjustment of weights in the network with respect to the loss gradient descent. The Gradient Descent Algorithm updates the model weights in iterations by minimizing a cost function. To implement Gradient Descent, the learning rate must be set to an appropriate value. If the learning rate is too large, the algorithm might skip the optimal solution, and if it is too small, it will need many iterations to converge. Therefore, selecting a suitable learning rate is critical. In simpler terms, the learning rate determines how quickly the network adopts new concepts, leaving behind the previous ones it has learned.[27]

- Batch Size:

  One of the main hyperparameters that need to be tuned is the batch size, which is the number of images used in every epoch to train the network. Setting this hyperparameter too high can make the network take too long to achieve convergence (no more gain in accuracy); however, if it is too low, it will make the network bounce back and forth without achieving acceptable performance.[28]

- Epochs:

  An iteration over all the training examples is called an epoch. The number of epochs to consider is a parameter of the deep learning algorithms. The total number of iterations equals the number of epochs times the sample size n divided by m, the size of a batch.[29]

5.2.2.3 Parameters

- Fully Connected Layer:

  A fully connected layer with n input dimensions and m output dimensions is defined as follows. The layer output is determined by the following parameters: the weight matrix $W \in M_{m,n}(\mathbb{R})$ having m rows and n columns, and the bias vector $b \in \mathbb{R}^m$. Given an input vector $x \in \mathbb{R}^n$, the output of a fully-connected layer FC with activation function f is defined as

  $$FC(x) := f(Wx + b) \in \mathbb{R}^m$$

  In the formula above, Wx is the matrix product and the function f is applied componentwise.[30]

- L1 Regularization :

  L1 regularization is also known as regularization for sparsity, and is used to address the challenges posed by sparse vectors consisting mostly of zeros. Sparse vectors can lead to high-dimensional feature vector spaces, making the model difficult to handle. L1 regularization reduces the weights of uninformative features to zero by subtracting a small amount from the weight at each iteration, eventually making it zero. The regularization term in L1 regularization penalizes the absolute value of the weight.[31]

- L2 Regularization :

  Regularization for simplicity is another name for L2 regularization. When considering the complexity of a model in terms of its weights, a feature's complexity is proportional to its weight's absolute value. L2 regularization reduces the magnitude of weights, but does not force them to be exactly zero. Instead, it acts as a force that gradually reduces the size of the weights with each iteration. Therefore, the weights will never become exactly zero. The regularization term in L2 regularization penalizes the square of the weight.[31]

- Rectified Liner Unit (ReLU):

  The ReLU function is a piecewise linear function that is defined as follows:

$$f(x) = \max(0, x)$$

The ReLU function has several advantages over other activation functions, such as the sigmoid function and the tanh function. First, the ReLU function is non-linear, which allows neural networks to learn more complex relationships between the input and output data. Second, the ReLU function is computationally efficient, which makes it possible to train deep neural networks with large numbers of parameters. Third, the ReLU function is less prone to the vanishing gradient problem, which is a common problem that occurs when training deep neural networks.[32]

5.2.2.4 Metrics

The efficiency metrics more often used to evaluate Classification models are:

- Accuracy: Classification accuracy is perhaps the simplest metric to use and implement and is defined as the number of correct predictions divided by the total number of predictions, multiplied by 100.[33]

Defined by:

$$\text{Accuracy} = \frac{\text{TrueNegatives} + \text{TruePositive}}{\text{TruePositive} + \text{FalsePositive} + \text{TrueNegative} + \text{FalseNegative}}$$

- Precision: Precision is the ratio of true positives and total positives predicted. The precision metric focuses on Type-I errors(FP). A Type-I error occurs when we reject a true null Hypothesis(H). So, in this case, the Type-I error is incorrectly labeling cancer patients as non-cancerous.

  A precision score of 1 will signify that your model didn't miss any true positives, and is able to classify well between correct and incorrect labeling of cancer patients. What it cannot measure is the existence of Type-II error, which is false negatives – cases when a non-cancerous patient is identified as cancerous.

  A low precision score (<0.5) means your classifier has a high number of false positives, which can be an outcome of an imbalanced class or untuned model hyperparameters. In an imbalanced class problem, you have to prepare your data beforehand with over/under-sampling or focal loss in order to curb FP/FN.[33]

Defined by:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall/Sensitivity/Hit-Rate:

  A Recall is essentially the ratio of true positives to all the positives in the ground truth. The recall metric focuses on type-II errors(FN). A type-II error occurs when we accept a false null hypothesis(H). So, in this case, the type-II error is incorrectly labeling non-cancerous patients as cancerous.

  Recall towards 1 will signify that your model didn't miss any true positives, and is able to classify well between correctly and incorrectly labeling cancer patients.

  What it cannot measure is the existence of type-I error which is false positives, i.e. the cases when a cancerous patient is identified as non-cancerous.

  A low recall score (<0.5) means your classifier has a high number of false negatives, which can be an outcome of an imbalanced class or not tuned model hyperparameters. In an imbalanced class problem, you have to prepare your data beforehand with over/under-sampling or focal loss in order to curb FP/FN.[33]

Defined by:

$$\text{Recall } = \frac{TP}{TP + FN}$$

- F1-score:

  The F1-score metric uses a combination of precision and recall. In fact, the F1 score is the harmonic mean of the two. Now, a high F1 score symbolizes high precision as well as high recall. It presents a good balance between precision and recall and gives good results on imbalanced classification problems.

  A low F1 score tells you (almost) nothing, it only tells you about performance at a threshold. Low recall means we didn't try to do well on very much of the entire test set. Low precision means that, among the cases, we identified as positive cases, we didn't get many of them right.

  But low F1 doesn't say which cases. High F1 means we likely have high precision and recall on a large portion of the decision (which is informative). With low F1, it's unclear what the problem is (low precision or low recall?), and whether the model suffers from type-I or type-II error.[33].

Defined by:

$$F_1 - \text{ score } = 2 \times \frac{\text{Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

5.2.2.5 Models

- Resnet50:

  ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pre-trained version of the neural network trained on more than a million images from the ImageNet

database [34]. The pre-trained neural network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the neural network has learned rich feature representations for a wide range of images. The neural network has an image input size of 224-by-224[35].

- InceptionV3: Inception-v3 is a convolutional neural network that is 48 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database [34]. The pre-trained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images[36].

- DenseNet121: DenseNet is one of the new discoveries in neural networks for visual object recognition. DenseNet is quite similar to ResNet with some fundamental differences. ResNet uses an additive method (+) that merges the previous layer (identity) with the future layer, whereas DenseNet concatenates (.) the output of the previous layer with the future layer.[37]

- MobileNet: MobileNet is a streamlined architecture that uses depthwise separable convolutions to construct lightweight deep convolutional neural networks and provides an efficient model for mobile and embedded vision applications. The structure of MobileNet is based on depthwise separable filters[38]

5.3 Related work

In this section, papers and investigations are going to be reviewed in order to establish the actual importance and level of development of the main topics treated in this thesis.

1. "Passive acoustic monitoring of animal populations with transfer learning" : This paper exposes some relevant literature where several author's works for this project are mentioned, the relevant information extracted from the article is related to the wide use of pre-trained CNNs for audio classification such as the following models: ResNet50 [39]) [40] and ResNet18 [39]. This study aims to simplify the implementation of CNNs and the design decisions when creating them, and to remove time-consuming hyperparameter tuning phases. Twelve modern CNN architectures were implemented and compared across 4 passive acoustic datasets. The results revealed that transfer learning can result in up to 82 percent in F1 score while keeping CNN implementation details to a minimum, thus rendering this approach accessible, easier to design, and speeding up further vocalization annotations to create PAM robust models.[9]

   The following paper shows a different approach as a new theory for bioacoustic sound classification with a very good efficiency rate:

2. "An automated multispecies bioacoustic sound classification method based on a nonlinear pattern: Twine-pat" : This work presents a new multispecies bioacoustic sound dataset and novel machine learning model to classify bird and anuran species with sounds automatically. In this model, a new nonlinear textural feature generation function is presented by using the twine cipher substitution box (S-box), and this feature generation function is named twine-pat. By using twine-pat and tunable Q-factor wavelet transform, a multilevel feature generation network is presented. Iterative ReliefF (IRF) is employed to select the most effective/valuable features. Two shallow classifiers are used to calculate results. Our presented model reached 98.75 percent in accuracy by using the k-nearest neighbor (kNN) classifier. The results obviously demonstrated the success of the presented model. [13]

3. "Data-Efficient Classification of Bird call Through Convolutional Neural Networks Transfer Learning" : This research has developed a theory where it is evaluated the effectiveness of bird call classification using transfer learning from a larger base dataset to a smaller target dataset using the ResNet-50 CNN. Resulting in a performance of 79 percent average validation accuracy on the target dataset in 5-fold cross-validation. The methodology of transfer learning from an ImageNet-trained CNN to a project-specific and much smaller set of classes and images was extended to the domain of spectrogram images, where the base dataset effectively played the role of the ImageNet.[41]

4. "Beluga whale acoustic signal classification using deep learning neural network models" : In order to increase the accuracy of classification results and reduce the labeling time, the authors constructed an ensemble deep learning convolutional neural network model to classify beluga detections as true or false. Using a 0.5 threshold, the final model achieves 96.57 percent precision and 92.26 percent recall on the testing dataset. This methodology proves to be successful at classifying beluga signals, and the framework can be easily generalized to other acoustic classification problems.[42]

5. "CMKD: CNN/Transformer-Based Cross-Model Knowledge Distillation for Audio Classification" : In this article, the authors find an intriguing interaction between the two very different models - CNN and AST models are good teachers for each other. When is used either of them as the teacher and train the other model as the student via knowledge distillation (KD), the performance of the student model noticeably improves, and in many cases, is better than the teacher model. In the experiments with this CNN/Transformer Cross-Model Knowledge Distillation (CMKD) method, this work| achieve a new state-of-the-art performance on FSD50K, AudioSet, and ESC-50.[43]

6. "Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning" : The authors explored the influence of key retraining parameters, including the optimizer, the mini-batch size, the learning rate, and the number of epochs, on the classification accuracy and the

processing time needed in terms of sound preprocessing for the preparation of the scalograms and spectrograms as well as CNN training. The UrbanSound8K, ESC-10, and Air Compressor open sound datasets were employed. The Sound CNNs achieved better classification accuracy, reaching an average of 96.4 percent for UrbanSound8K, 91.25 percent for ESC-10, and 100 percent for the Air Compressor dataset.[44]

Chapter 6

Methodology and resources

6.1 Data collection

For this project, one main dataset was made with the data provided by the Humboldt Institute; the data came from a place named INCT41 which is located in the Cerrado biome in Brazil.

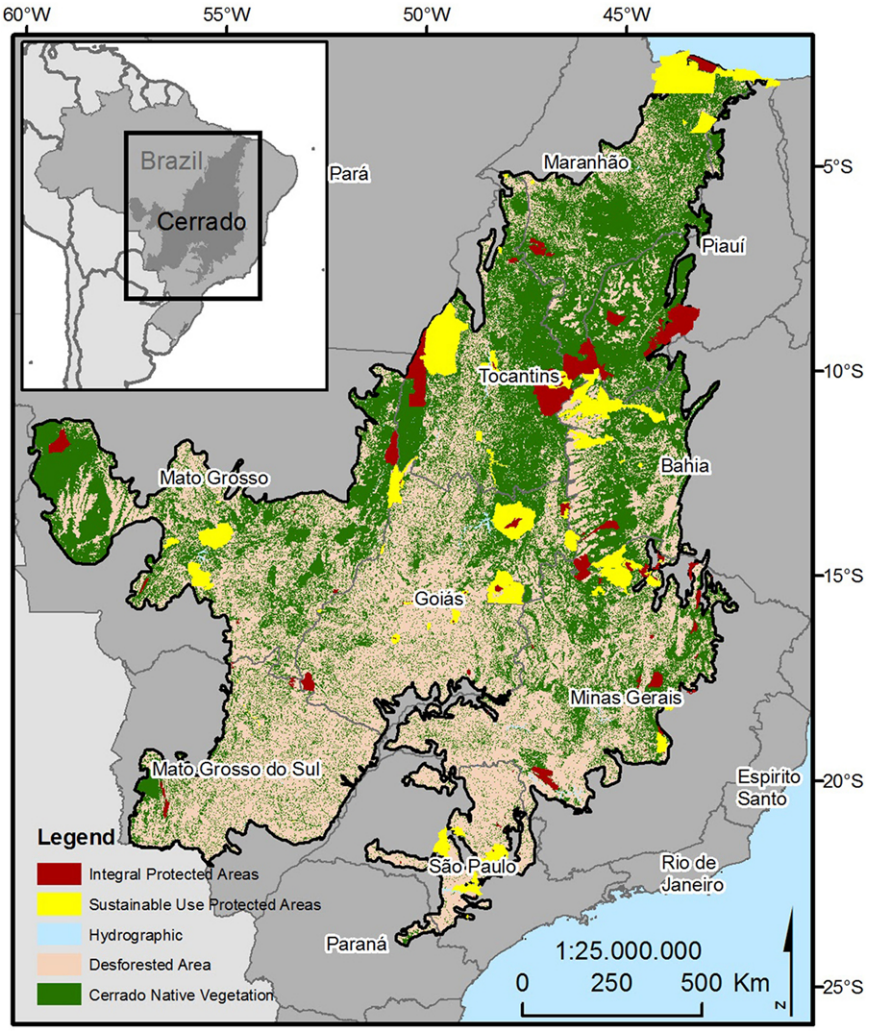

Figure 6.1: Map of Cerrado biome, located in the Bifurcação locality in Brazil.

In order to obtain this dataset, it was developed a script for trimming the audios, extracting each spectrogram, and combining the audios with their respective annotation; once this dataset was done, another 4 datasets were made from this original one, because of the implementation of 3 augmentation techniques (time stretching, time masking, and frequency masking, the 4th dataset was a union of the other augmented and not augmented datasets made, without forgetting to exclude the 30 percent of the main dataset to perform the testing of the trained models with all the different datasets created.

Each dataset was created with the same structure to avoid complications and minimize hard-coding in the training and pre-processing scripts. For clarification purposes, the names of each species were replaced by the following codes: *Boana albopunctata*: BOAALB, *Physalaemus cuvieri*: PHYCUV, and *Boana lundii*: BOALUN.

The datasets generated have the following structure:

NAME	Path	PHYCUV	BOAALB	BOALUN
INCT41_20201022_213000_3	../NEW_Orig	0	0	0
INCT41_20201211_231500_5	../NEW_Orig	0	0	0
INCT41_20201218_230000_2	../NEW_Orig	0	0	0
INCT41_20201113_010000_0	../NEW_Orig	0	0	0
INCT41_20210106_023000_5	../NEW_Orig	0	0	0
INCT41_20201031_180000_1	../NEW_Orig	0	0	0
INCT41_20201220_033000_8	../NEW_Orig	0	0	0

Figure 6.2: Dataset structure

6.2 Methodology

The methodology followed during the project was separated into 4 stages. First, the preprocessing stage over the INCT41 audio recordings and annotations database, second the different augmentations techniques implemented over the original dataset, third the training process of the models and lastly, the final stage, the performance evaluation for each model. The use of the silver standard [45] in this methodology is justified as it ensures that the work can be reproduced with greater ease and efficiency, the dependencies of the analysis can be easily downloaded and installed with a single command, and all key details for reproducing the work are documented using sphynx. Additionally, all random components in the analysis are set to be deterministic, which further ensures reproducibility. This level of standardization allows for greater transparency and reliability in the methodology, as it reduces the potential for errors or discrepancies in the analysis.[45]

As part of the deliverables, a **GitHub** repository was created, in this repository are available the created datasets used for all the experiments (Under the folder named "*Datasets*"), the coding

scripts used for all de stages of the projects (Under "*SCRIPTS*" folder), the documentation for the functions developed and implemented (under "*Documentation file*" folder), all the **models** generated in this project, and a notebook of **Google Colab** designed to train the models as easy as changing a path and set the cells of the script to run.

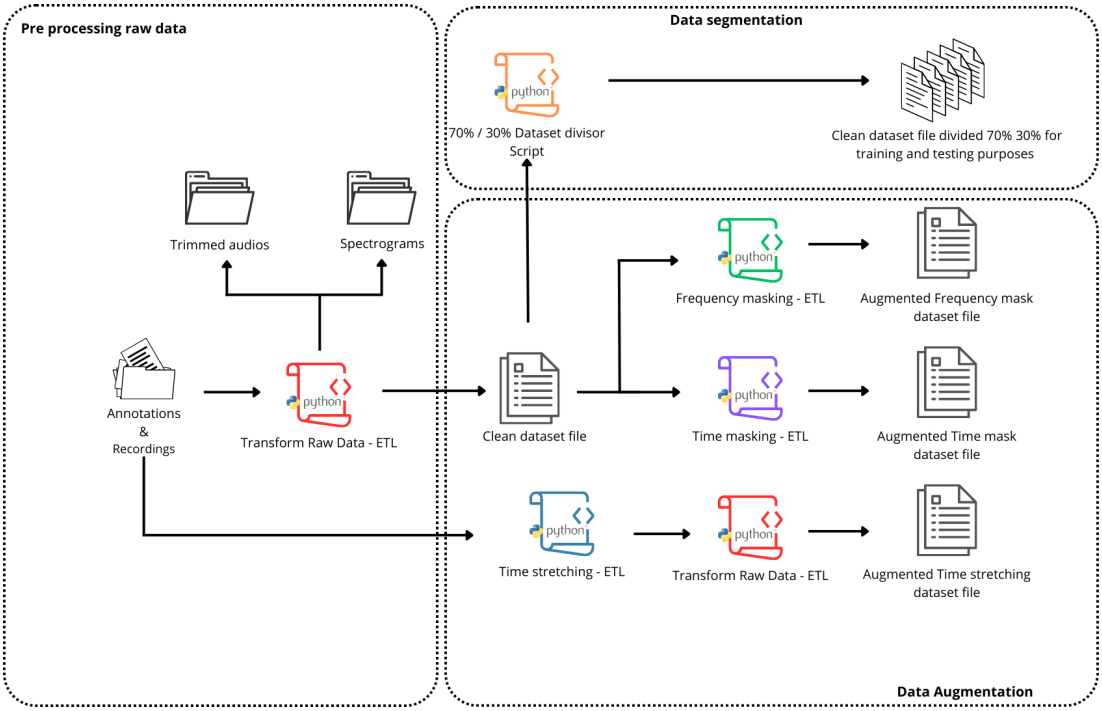

Figure 6.3: Pipeline of preprocessing, augmentation, and segmentation processes

6.3 Data Preprocessing

Since it was necessary to find a way to obtain a dataset from the raw audios and annotations that would contain enough samples for each label and that these samples would be spectrograms derived from the raw audio. A solution to this need emerged by developing this preprocessing stage, in which the action plan consisted of segmenting the raw audios that lasted 1 minute in segments of 5 seconds, in order to have a greater number of samples per species, obtain a spectrogram for each segment and correlate them with the raw annotations to identify whether or not if there is any presence of the species in that segment of audio converted into a spectrogram. In conclusion, this stage is the core foundation of the project's development.

These are the main steps followed in the script developed for preprocessing the raw audio and annotation files in order to obtain a clean dataset for training purposes:

- Set main variables: This section sets the main variables that will be used throughout the script, such as the target sampling rate for the audios (24000 Hz), the window length (5 secs) for formatted regions of interest (ROIs), window length, annotation and audio file paths, and the location to save new samples.

- Trimming audios: Loads all the annotations from the origin directory using the "load annotations function", which reads each annotation file and creates a pandas Data Frame with columns for the file name, the start time, and the duration of each labelled region of interest (ROI), as well as the label associated with each ROI. Then, it groups these annotations by audio file and cuts the audio into smaller chunks.

- Spectrogram files generation: Generation of spectrograms from the trimmed audio files, this involved a series of parameters (**Hop length** which is the number of samples between successive frames, and **NFFT** which stands for Non-equispaced Fast Fourier Transform.) variation for better spectrogram image quality, also a data frame was created as a guide for dataset creation. The spectrogram files were saved in a separate output folder. Next, a new data frame was made with columns containing the spectrogram file name and the location path. Then the spectrogram and label data frames are merged by the 'NAME' column to get a data frame that will become the clean dataset after some imputations.

- Spectrograms evaluation:

  For the generation of spectrograms, it was decided to test two approaches:

  - Use of SFFT and specshow library.
  - Use of Mel spectrogram function of librosa.

  To modify parameters and find the best way to display the calls in the spectrograms the following formula was used: FFT Length = (Sample Rate * Duration) / Hop Length

  - The Sample Rate of our audio is = 22050Hz.
  - Our audio duration is = 5sec.

  The Hop Lenght was modified based on powers of 2 decreasing from 512 and clearing the FFT Lenght value with each variation.

  Here a sample of the spectrogram images generated:

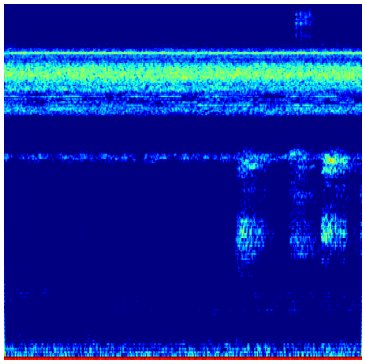

Figure 6.4: Spectrogram image sample

- Finally, some data from the merged dataframe goes through an imputation process, dropping unnecessary columns and merging some into new ones. Then the modified data frame is saved into a CSV file, the dataset.

6.4 Data augmentation

This stage was developed due to a huge imbalance(see image below) in the original dataset noticed during the first stage of the project, which led to the formulation of this hypothesis:

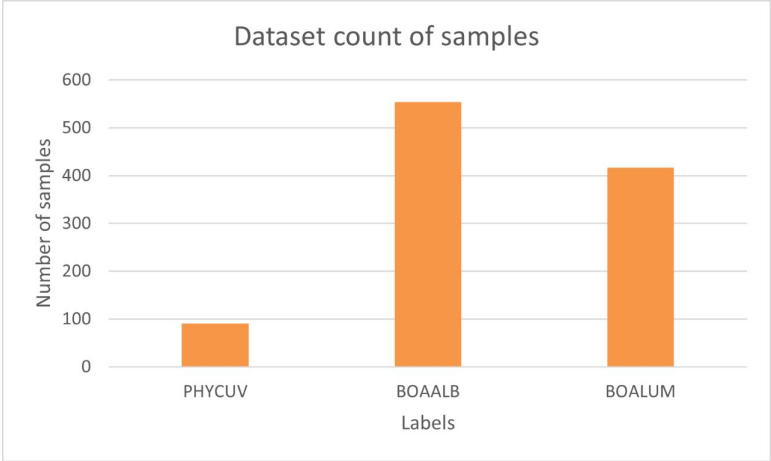

Figure 6.5: Barplot image of sample distribution

- The model robustness and prediction efficiency can be improved by using augmentation techniques combined with the original dataset in the training process of the models
- Certain augmentation techniques can be used to balance an unbalanced dataset and improve the performance of the trained models

The process followed to perform this task was:

- Definition of augmentation techniques to be implemented, based on articles and expert suggestions from the field.

  The selected augmentation techniques were:

  - Time Stretching: Time stretching is a data augmentation technique used to change an audio file's tempo without changing its pitch. This can be done by stretching or shrinking the file's duration. Time stretching can create various effects, such as making a song sound faster or slower, or creating a looping effect.[46][47]
  - Time Masking: Time masking is a data augmentation technique that is used to remove a portion of the time steps from an audio file. This can be done by either randomly masking time steps or by masking time steps related to a specific feature, such as pitch or loudness. Time masking can make audio files more challenging for machine-learning models to classify. [48]
  - Frequency Masking: A frequency channels $[f_0, f_0 + f)$ are masked. f is chosen from a uniform distribution from 0 to the frequency mask parameter F, and f_0 is chosen from $[0, \nu - f)$ where ν is the number of frequency channels.[49]

- ETL development for each augmentation process, for time and frequency masking. The dataset created by the initial preprocessing stage was used as the source for the data.

  For Time Stretching, a previous step even before the initial stage had to be made (developed in a script) to transform the raw audio files from 59 sec to 110 sec. A Dataset file was created to facilitate the model's training process for each augmentation process.

  Here is a sample of the spectrograms generated for each augmentation technique:

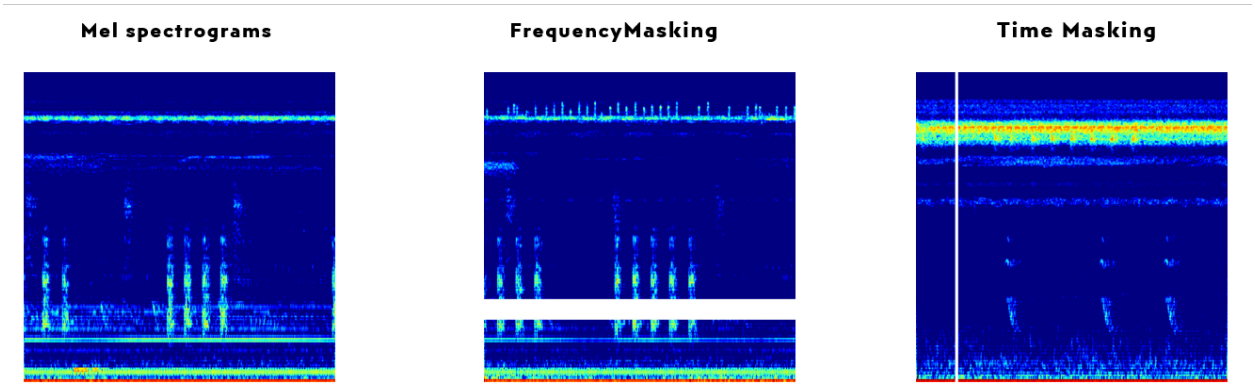

Figure 6.6: Augmented sample images.

6.5 Models training setup

Before going deeper into the Training and performance evaluation methodology used in this project, it is essential to understand the structure of the model setup for each training. The following sections outline the critical components defined in the code developed for this task.

- Base Model: The base model is created using the selected pre-trained model with its fully connected top layers removed. The include top parameter is set to False, to exclude the top layers and the input shape is set to (224, 224, 3) to specify the input size for the model. The weight parameter is set to 'ImageNet' to initialize the model with weights pre-trained on the ImageNet dataset.

- Freeze Layers: Then the trainable property of all the layers in the base model is set to False, to freeze them and prevent them from being retrained.

- Custom Output Layer: A custom output layer is added to the model to enable multi-label classification. The output layer includes a flattened layer to convert the output from the base model to a 1D vector. The final output layer consists of a Dense layer with three units (the number of classes) and a sigmoid activation function to predict the presence of each class.

- Model Compilation: The model is compiled using the Adam optimizer with a learning rate defined in each experiment, binary cross-entropy loss function, and F1 Score as evaluation Metric.

- Callbacks: Two callbacks are set up to monitor the training progress and save the best model weights. The Early Stopping callback stops training early if the validation loss does not improve for five epochs. The Model Checkpoint callback saves the best model weights based on the validation loss.

- Training: Finally, the models were trained for: 1000 epochs during the last two stages and 100 during the first stage maximum with a batch size of 32 using the fit() function, which takes the training data (X train and y train), validation data (X test and y test) previously divided in 70 percent 30 percent stratified and with a random state defined in 42, and callbacks as input. The training history is stored in the 'history' variable.

- F1 – score: To calculate the F1 score for a multi-label classification problem, it is necessary to compare the predicted labels (y pred) with the true labels (y test). The predicted labels are probabilities or scores indicating the likelihood of each label being present for a given instance. To convert these probabilities into binary predictions, a threshold of 0.5 is used in this case. Any predicted probability greater than 0.5 is considered a positive prediction for that label, while probabilities less than or equal to 0.5 are considered negative.

6.6 Model training and performance evaluation

In this stage, some high computational resources were needed, for this we used Google Colab PRO, which gave us an Nvidia TESLA P100 GPU with 40 GB, 40 GB of RAM, and 70 GB of Disk Usage. F1 score was the selected metric, it is used to measure the performance of the CNN pre-trained models in multi-label classification because it combines precision and recall into a single metric, offering a balanced assessment of model accuracy in handling multiple labels.[50]. A defined training plan for the selected models was also implemented:

Training and evaluation plan:

- Data preprocessing: First, it was necessary to preprocess the image data similarly for all the models. A script was developed to resize the images to a fixed size (224×224) and normalize the pixel values.
- Splitting the data: To split the dataset into training, and validation sets. We will use 70 percent of the data for training and 30 percent for validation.
- Model selection: The criteria for selecting the models to be trained were based on this chart:

Table 6.1: Models comparison

Model	Depth	Trainable Parameters	Size	#of Weights	#of Biases
VGG16	16	138,357,544	528MB	138,357,544	0
VGG19	19	143,667,240	548MB	143,667,240	0
ResNet50	50	25,636,712	98MB	25,583,592	53,120
Inception-V3	159	23,851,784	92MB	23,768,712	83,072
DenseNet-121	121	8,062,504	31MB	7,978,856	83,648
Xception	126	22,910,480	88MB	22,855,840	54,640
MobileNet	88	4,253,864	16MB	4,253,384	480

(6.1)

The model selection was then made based on the depth, therefore the selected models were: MobileNet, Inception-V3, DenseNet-121, and ResNet50.

All the models will be pre-trained on the ImageNet dataset and fine-tuned on the target dataset, following this process:

- It will be used for binary cross-entropy loss and sigmoid activation function for multi-label classification.
- Hyperparameter tuning: Hyperparameter tuning (or hyperparameter optimization) determines the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process.[51] Each trial is a complete execution of the training application with different values for the chosen hyperparameters(learning

rate, epochs, batch size), each of those hyperparameters has its own repercussion in the model training, starting with the learning rate, tuning this hyperparameter allows the model to learn faster however this could end in a suboptimal set of weights. Continuing with the epochs, this hyperparameter determines a balance between under training and overfitting the model. Finally, the batch size hyperparameter will determine the number of samples used in each epoch, resulting in variations in hardware usage and model convergence.
Needless to say, hyperparameter tuning is needed to find the best learning rate, batch size, number of epochs, number of fully connected layers, L2 Regularization, K – fold cross-validation application if needed.

- Process:
  - The process begins by varying the learning rate from 0.00001 to 0.01 in steps of 0.1. This range allows the exploration of different learning rates to find the one that achieves optimal convergence and avoids overshooting or slow convergence.
  - Next, the batch size is varied between 16, 32, and 64. The choice of batch size is crucial as it affects computational efficiency, memory requirements, and the accuracy of gradient estimation. In this case, a batch size of 32 is selected to balance these factors, considering memory limitations.
  - The number of epochs is then varied from 10 to 100 in steps of 10. Early stopping is implemented to prevent overfitting, and the epoch size is fixed at 100 to allow for thorough training and convergence exploration.
  - Different fully connected layer configurations are experimented with. One configuration includes adding a single fully connected layer with 256 units and the ReLU activation function. Another design includes adding two fully connected layers with 128 and 256 units, respectively, using the ReLU activation function. These configurations help explore different network depths and capacities.
  - L2 regularization is applied to each layer to mitigate overfitting and improve generalization. This regularization technique adds a penalty term to the loss function, encouraging the model to have smaller weights and reducing the impact of individual features on the predictions.
  - To assess the model's performance and avoid bias, K-fold cross-validation is applied. This technique helps evaluate the models on different subsets of the data, providing a more robust estimate of their performance.

- To avoid overfitting, techniques such as early stopping and regularization are defined. Early stopping allows the training to stop when the validation loss stops improving, preventing the model from memorizing the training data. The regularization techniques, such as L2 regularization, help control the model's complexity and reduce overfitting by discouraging large weight values.

- To overcome hardware limitations, a Google Colab Pro notebook is used to access better hardware resources for the training process. This ensures that the models can be trained effectively and efficiently.

- Each trained model is saved to facilitate later metrics extraction and evaluation. This step allows for a comprehensive analysis of the model's performance and comparison.

- Finally, the performance of each model is evaluated using the F1-score metric on a separate dataset. The F1-score is a suitable metric for assessing multi-label classification models, as it considers precision and recall. The models are compared based on their performance, and the ones with the best F1 scores are selected for further analysis and potential deployment.

6.7 Experimental design

This section presents the experimental approach taken to evaluate the performance of the models and gain insights into the effect of the different hyperparameters. These experiments aimed to determine the optimal settings of hyperparameters, provide valuable recommendations for future work in this area, and comprehensively evaluate the models.

- Learning Rate: Generally, a large learning rate allows the model to learn faster, at the cost of arriving at a suboptimal final set of weights. A lower learning rate may allow the model to learn a more optimal or even globally optimal set of weights, but may take significantly longer to train.[52] "The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyperparameter, tune the learning rate".[27]

- Batch Size: Batch Size is among the essential hyperparameters in Machine Learning. The hyperparameter defines the number of samples to work through before updating the internal model parameters. It can be a crucial step to ensure your models hit peak performance. It should not be surprising that there is a lot of research into how different Batch Sizes affect aspects of your ML pipelines.[53]

- Epochs: The number of epochs used in the training process is a vital hyperparameter that must be carefully selected, as too few epochs can result in an under-trained model, while too many generations can result in overfitting, where the model becomes too specialized to the training data and performs poorly on new, unseen data[54]

- Regularization: The main aim is to reduce the over-complexity of the machine learning models and help the model learn a simpler function to promote generalization.[55]

- Fully connected layer: The more layers in the network, the greater the complexity and (theoretically) the accuracy of the machine learning model. Each additional layer that processes the input data increases the model's ability to recognize objects and patterns in the data. The ReLU activation function is a commonly used non-linear activation function that returns the

input value if it is positive, and 0 if it is negative. ReLU helps to introduce non-linearity in the model and can improve the model's ability to learn complex patterns in the data.[56]

- K fold cross validation: Choosing the right k-value is essential as it can affect your level of accuracy, variance, and bias and cause you to misinterpret the overall performance of the model.[57]

6.7.1 Experiments design

The following experiments were conducted to assess our models and draw meaningful conclusions comprehensively.

1. **Models Training stage 1:**

   In the initial experiment, the models were trained using the original dataset to observe the effect of varying hyperparameters. Specifically, the learning rate, batch size, number of epochs, and regularization techniques (L1 and L2) varied.

   - **Models selected:**
     - DenseNet121.
     - ResNet50.
     - MobileNet.
     - Inception V3.
   - **The following hyperparameter values were modified:**
     - Learning rate: 0.0001 to 0.1 (in steps of 10)
     - Batch size: 16 to 128
     - Number of epochs: 100 to 1000

   The best-resulting model Hyperparameters were used for the next experiments

2. **Models training stage 2:**

   This experiment was performed with the same models as in Model Training 1 (Inception V3, DenseNet121, MobileNet, and ResNet50), but with the hyperparameter values that yielded the best performance in the previous experiment. Early stopping and model checkpoint techniques were used to prevent overfitting. It also included additional architecture and regularization techniques to the pre-trained models to determine their effect on performance.

   - **Datasets used:**
     - Original dataset.
     - Time stretching dataset.
     - Time masking dataset.

  - Frequency masking dataset.
- **Regularization techniques:**
  - L2 regularization.
  - No regularization.
- **Model architectures:**
  - One fully connected layer with 256 neurons and ReLU activation function.
  - Two fully connected layers with 128 and 256 neurons and ReLU activation function.

5-fold cross-validation and the usual .fit technique for the model's training were implemented.

The hyperparameters and architecture mentioned above varied for each model and dataset, resulting in several models trained and ready to be tested in a dataset isolated from the training data used. Later, the hyperparameters were used by the best models for the next experiment.

3. **Models Training stage 3:**

   This experiment was performed with the same models as in Model Training 1 and 2 (Inception V3, DenseNet121, MobileNet, and ResNet50), but with the hyperparameter values that yielded the best performance in the previous experiment.

   - **Datasets used:**
     - Original dataset + all augmented datasets altogether.
   - **Regularization techniques:**
     - L2 regularization.
   - **Model architectures:**
     - One fully connected layer with 256 neurons and ReLU activation function.
     - Two fully connected layers with 128 and 256 neurons and ReLU activation function.

2-fold cross-validation and the usual .fit technique were used to train the models.

We varied the hyperparameters and architecture mentioned above for each model and dataset, resulting in several models trained and ready to be tested in a dataset isolated from the training data used.

All information needed regarding the scripts and specific libraries used for developing this project can be found on the project's GitHub.

6.8 Resources

- Google Colab pro:
  - GPU: Nvidia TESLA P100 - 40 GB

  - RAM: 40 GB
  - Disk: 100 GB, 70 GB utilized

- Google Drive Storage (100 GB): Used for data storage and collaboration.
- GitHub Pro: Version control system for code management and collaboration.
- Code Spaces: Integrated development environment (IDE) for coding and project development.
- Visual Studio: IDE used for coding and project development.
- GIT: Distributed version control system used for code management.
- Microsoft Excel: Used for data manipulation and analysis.
- Python 3.11: Programming language used for implementing the project.
- Laptops: Used as local development machines for coding and testing.
- INCT41 audios and annotations shared by the Humboldt Institute: Dataset used for analysis and model training.
- The following are the main libraries used to develop this project:
  - NumPy: Utilized for efficient numerical computations and array manipulation in scientific computing.
  - Pandas: Employed for data analysis tasks, providing powerful data structures and tools.
  - Maad: Utilized for audio analysis, offering a comprehensive set of tools for working with audio data.
  - Glob: Used for filename globbing, providing a simple interface for matching filenames against patterns.
  - Sphinx: Employed for generating documentation in various formats from restructured text files.
  - Matplotlib: Utilized for creating high-quality plots and visualizations.
  - Seaborn: Employed for statistical data visualization, offering an attractive and informative interface.
  - Wave: Utilized for reading and writing WAV audio files.
  - TensorFlow: Used as a machine learning library, providing high-level and low-level APIs for building and training models.
  - Keras: Employed as a deep learning library, offering a high-level API for building and training deep learning models.
  - Scikit-Learn: Utilized as a machine learning library, providing a wide range of tools for data preprocessing, feature selection, model selection, and evaluation.

These resources were utilized in the project for various purposes. Google Colab Pro, with its powerful GPU, ample RAM, and sufficient disk space, facilitated the efficient execution of computationally intensive tasks. Google Drive Storage, GitHub Pro, and Code Spaces were used for data storage, version control, and collaborative development. Visual Studio and laptops served as the primary development environments. Microsoft Excel aided in data manipulation and analysis. Python 3.11 and the mentioned libraries formed the core programming framework for implementing the project. The INCT41 audio dataset, provided by the Humboldt Institute, was used for analysis and model training. The selected libraries played crucial roles in numerical computations, data analysis, audio analysis, file handling, documentation generation, plotting, visualization, audio file manipulation, and machine learning model development and evaluation.

Chapter 7

Results and discussion

7.1 Spectrograms generation

As mentioned previously, at the beginning of the project it was necessary to find the best combination of parameters to generate spectrograms of the audio clips that had been generated, therefore, after some research and consulting, it was determined that the parameters to be modified would be the hop length and the NFFT.
The functions developed to generate and store the spectrograms are defined below:

- Generate spectrogram function:

  This function generates a melted spectrogram from an audio file and saves it as an image. It takes several parameters, including the audio file path, output file path, sample rate (default is, 22050), FFT window length (default is 1024), hop length (default is 64), and the number of Mel bands to generate (default is 256). It uses the Librosa library to load the audio files and calculate the Mel spectrogram. The Mel spectrogram is then converted to a logarithmic scale using librosa.amplitude to db. The Matplotlib library was used to display and save images of the spectrograms.

- Generate spectrograms from folder function:

  This function generates spectrograms from WAV files in a specified input folder and saves them in an output folder. It accepts the path to the input folder and the path to the output folder as arguments. If the output folder does not exist, it is created with os.makedirs. This function recursively traverses the input folder using os.walk to find all the WAV files. For each WAV file, it generates a corresponding spectrogram by calling the generate spectrogram function. The path of the output file is constructed from the path of the input file and is saved with the PNG file extension.

- The get spectrogram paths function:

  This function returns a list of spectrogram file paths found in the specified directory. It takes the directory path as an argument. The function uses os.walk to traverse the directory, searching for files with .png extensions. For each matching file, the spectrum file path is constructed using os.path.join and added to the list of spectrum file paths. The list of spectrum paths is returned as output.
  These functions provide a convenient way to generate spectrograms from audio files, save

them as images, and retrieve the paths of the generated spectrograms for further processing or analysis.

Here are the image results of each variation performed:

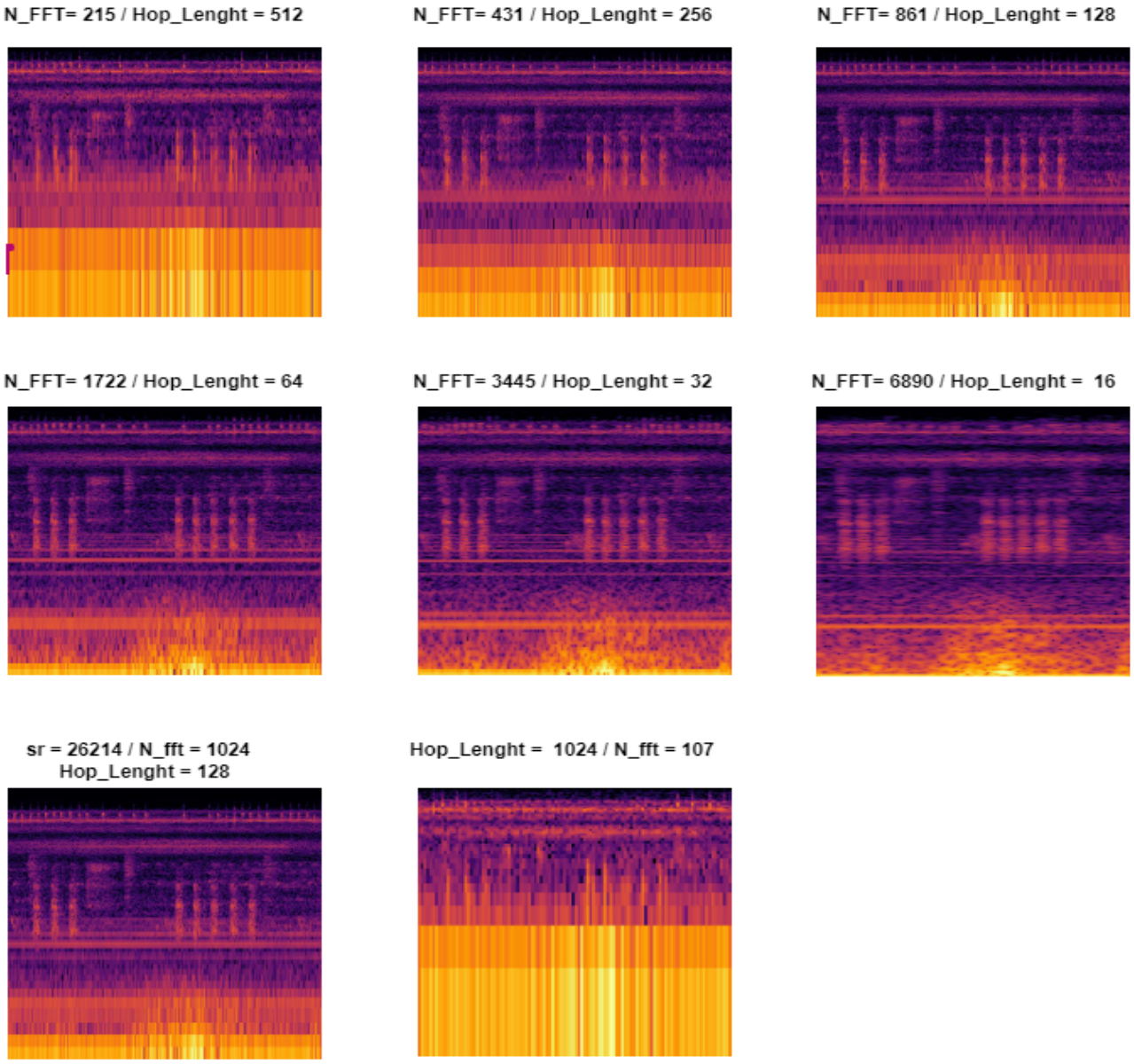

Figure 7.1: Spectrogram images results from changing parameters as Hop Lenght and N fft, graphic comparison of how the Anuran calls are being represented

From these tests it was concluded that this method for obtaining spectrograms is not appropriate for the application to be used, because the calls cannot be easily identified and in some cases, almost nothing can be distinguished, however, it was concluded that the combination of parameters HopLenght = 64, NFFT = 1722 produced the best image based on the visual analysis of the spectrograms that allowed to obtain better insights into the intricate details of the low-frequency components of each desired anuran call to classify, which were the most relevant for our research objectives. Furthermore, the performance of these spectrograms in our models provided further validation of their effectiveness in capturing and representing the critical low-frequency aspects of the data.

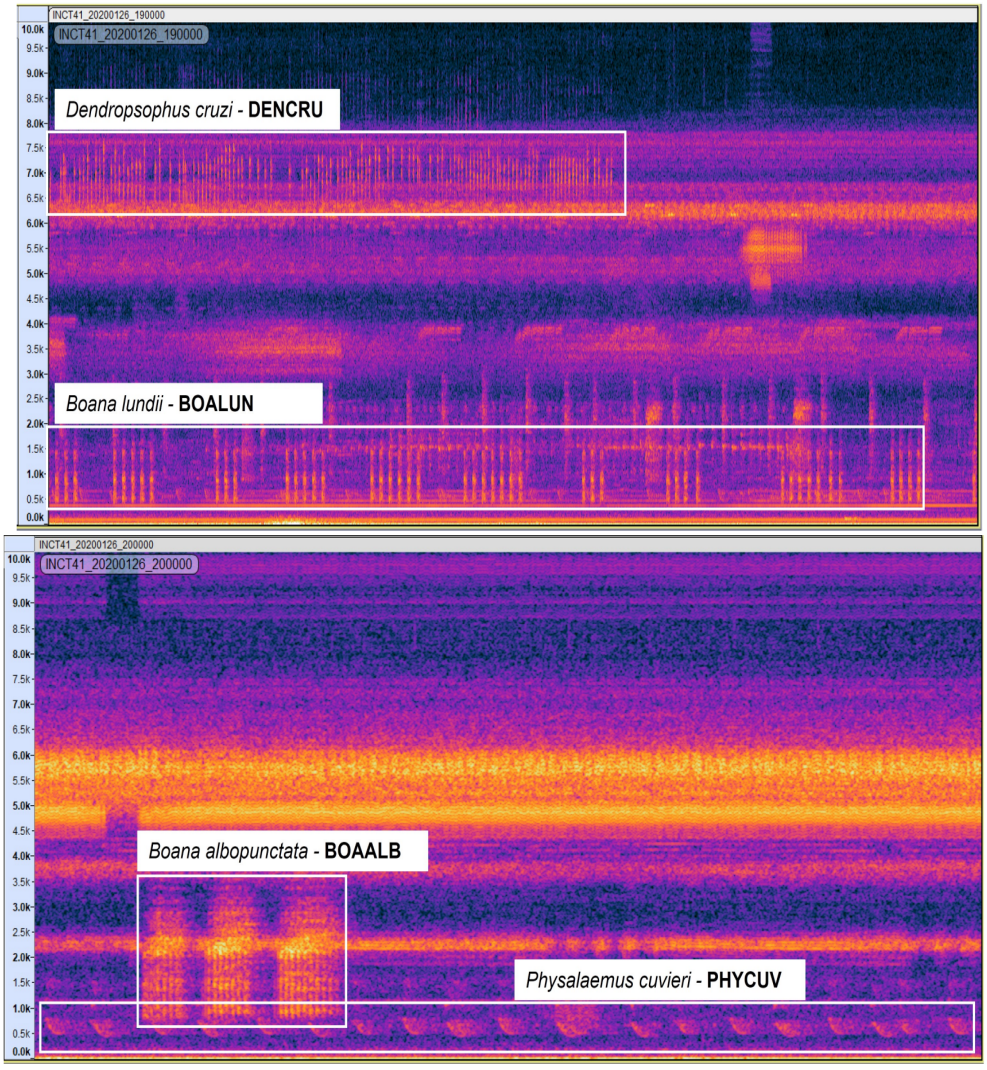

Figure 7.2: Acoustic signal of the target species at INCT41 site.

Therefore, these parameters were used in the following test with the Mel spectrograms genera-

tion, and these were the results:

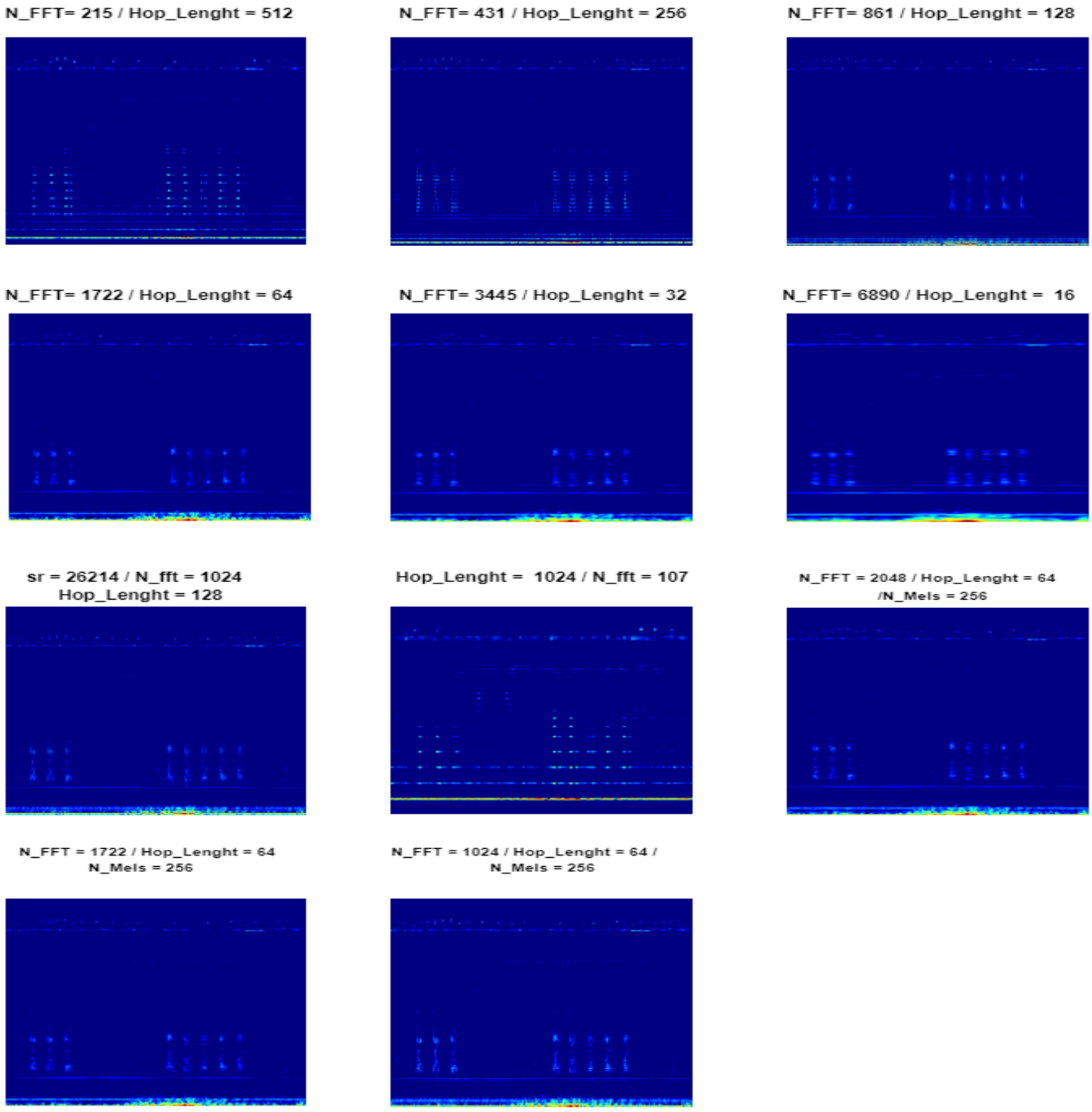

Figure 7.3: Mel spectrogram images results from changing parameters as Hop Lenght and N fft, graphic comparison of how the Anuran calls are being represented, based on previous experiment

From these tests, it was concluded that the set of parameters: NFFT = 1024 / HopLenght = 64 /NMels = 256 was the one that gave the best visual result based on the previously mentioned visual analysis, so it was decided to proceed with the training process of the models defined with the dataset composed of images obtained using these parameters.

7.2 Models training stage 1

This experiment aimed to explore the impact of varying hyperparameters on the performance of the convolutional neural network (CNN) models. Four CNN architectures were selected based on previously mentioned criteria, by modifying several key hyperparameters, including the learning rate, batch size, number of epochs, and regularization techniques (L1 and L2).
Selected Models: DenseNet121, ResNet50, MobileNet, and Inception V3.
To begin with, the models were trained using the original dataset and different combinations of hyperparameters to assess their effects on the performance of the selected models. The learning rate was varied from 0.0001 to 0.1, incrementing in steps of 10, this was repeated for every model adding in L2 regularization.
In total 22 models were trained variating the epochs, lr, L2, and L1 regularization techniques. Here is a table with the top results obtained from these 22 trained models

Table 7.1: Performance metrics table of first experiment evaluation, to determine the best hyperparameters.

Model Name	F1 Score
MobileNet_Reg_L2_lr_00001_Batch_32	0.680
Resnet50_Reg_L2__lr_0001_Batch_32	0.567
InceptionV3_Reg_L2_lr_00001_Batch_32	0.579
DenseNet_Reg_L2_lr_0001_Batch_32	0.598

After conducting the experiments, it was observed that the performance of the models varied significantly with different hyperparameter settings. Notably, the choice of learning rate had a substantial impact on the convergence and overall F1-Score of the models. Lower learning rates, such as 0.0001 and 0.001, resulted in slower convergence but often yielded a higher F1-Score. On the other hand, higher learning rates, such as 0.01 and 0.1, led to faster convergence but occasionally suffered from suboptimal F1-Score. Regarding the batch size, it was found that if a batch size larger than 32 is intended to be used, it is needed more than 16 GB of RAM memory due to the nature of the dataset composed of images that were transformed into Numpy arrays, therefore a batch size of 32 was standardized during the study.

Moreover, the number of epochs defined played a crucial role in the training process. While increasing this number to a maximum of 100 in this stage, potentially improved F1-Score, it wasn't

clear if an overfitting situation was happening.

Therefore, a function to avoid overfitting and under fitting must be used if the intention is to find models with real and functional metrics to solve this multi-labeling problem.

The effect of regularization techniques was investigated, and therefore implemented specifically L1 and L2 regularization techniques on the model training process, looking to improve the performance (F1-Score). The main reason to apply regularization was that it's a technique widely used to prevent overfitting and improve generalization. However, L2 Regularization proved to be more effective in this application.

It was found a huge imbalance in the number of samples per species in the dataset generated, therefore it was proposed to perform a balance of the original data using augmentation techniques known and used previously in other similar studies, intending to minimize the bias that could be caused by the imbalance in this original dataset. The total sample count encountered in the original dataset used, including the 30 percent isolated for testing, are:

Table 7.2: Labels description of the total samples count in original dataset

Species acronym	Count of samples
PHYCUV	91
BOAALB	554
BOALUN	417

- PHYCUV: Physalaemus cuvieri
- BOAALB: Boana albopunctata
- BOALUN: Boana lundii

7.3 Models training Stage 2

Based on the previous results, a series of hypotheses were made to orientate the strategies for this training stage:

Hypothesis:

- The addition of n fully connected layers to a pre-trained model can improve the model's ability to learn complex non-linear relationships in the data, resulting in improved performance for the task.

- The addition of n fully connected layers may result in overfitting the model to the training data, leading to poor generalization performance on unseen data.
- Models with a lower number of parameters can perform better for this multi-label classification problem.

The workflow for this stage was similar to the first one, but this time the architecture of the models is modified. On the output layer, this time one fully connected layer or two can be added depending on the experiment.

The specifications of these fully connected layers are the following:

- A Dense layer with 256 units, ReLU activation, and L2 regularization (with a coefficient of 0.01) follows the Flatten layer.
- A Dense layer with 128 units, ReLU activation, and L2 regularization (with a coefficient of 0.01) follows the Flatten layer.
- To mitigate overfitting, a Dropout layer with a dropout rate of 0.5 is included after each new layer is added.

Another important difference between this stage and the first one is that now each experiment that involves a hyperparameter change for a new model training process is going to be done over 3 different datasets: Time Masking Dataset, Frequency Masking Dataset, and Original Dataset.

Count of samples per species per augmented dataset combined with the 70 percent of the samples in the original dataset:

Table 7.3: Labels total samples count in frequency masking dataset, made to achieve a balanced distribution among labels and visualize it behavior on training process.

Species acronym	Samples in Original Dataset	Samples Augmented dataset	Total samples
PHYCUV	64	549	613
BOAALB	388	240	628
BOALUN	292	330	622

Table 7.4: Labels total samples count in time masking dataset made to achieve a balanced distribution among labels and visualize it behavior on training process.

Species acronym	Samples in Original Dataset	Samples Augmented dataset	Total samples
PHYCUV	64	260	324
BOAALB	388	61	449
BOALUN	292	72	364

It is important to clarify that the fact of completely balancing these datasets is a very difficult task due to the multi-label characteristics, meaning that if one image is augmented, probably it's going to be augmented more than 1 label at once. This is why the total sample count is not the same, but they are very close compared to the number of samples counted in the original dataset.

The models trained in this stage will have the following fixed parameters:

- Adam optimizer with learning rate 0.00001.
- Drop out (0.5).
- L2 Regularization.
- Top epochs 1000.
- Early stopping patience = 5.
- Loss: Binary Crossentropy.
- Top layer input shape (224,224,3).
- Weights: Imagenet.
- Output layer Activation: sigmoid.

And the following parameters per model will be variated:

1. Dataset: Augmented (time masking, frequency masking), Not augmented.
2. Number of extra fully-connected layers (1 or 2).
3. Cross-validation 5 folds.

As a summary of what was the training process per model (Resnet50, InceptionV3, Densenet121, MobileNet):

1. Dataset: Augmented – frequency masking
    - One fully connected Layer
        - CrossValidation-5folds
        - Normal Training

- Two fully connected Layer
  - CrossValidation-5folds
  - Normal Training

2. Dataset: Augmented – time masking

- One fully connected Layer
  - CrossValidation-5folds
  - Normal Training
- Two fully connected Layer
  - CrossValidation-5folds
  - Normal Training

3. Dataset: Not augmented – original

- One fully connected Layer
  - CrossValidation-5folds
  - Normal Training
- Two fully connected Layer
  - CrossValidation-5folds
  - Normal Training

Resulting in a total of 12 trained models per selected model (Resnet50, InceptionV3, Densenet121, MobileNet) that all together makes a total of **48** trained models.

Here are the best F1 score achieved in this stage by the trained models:

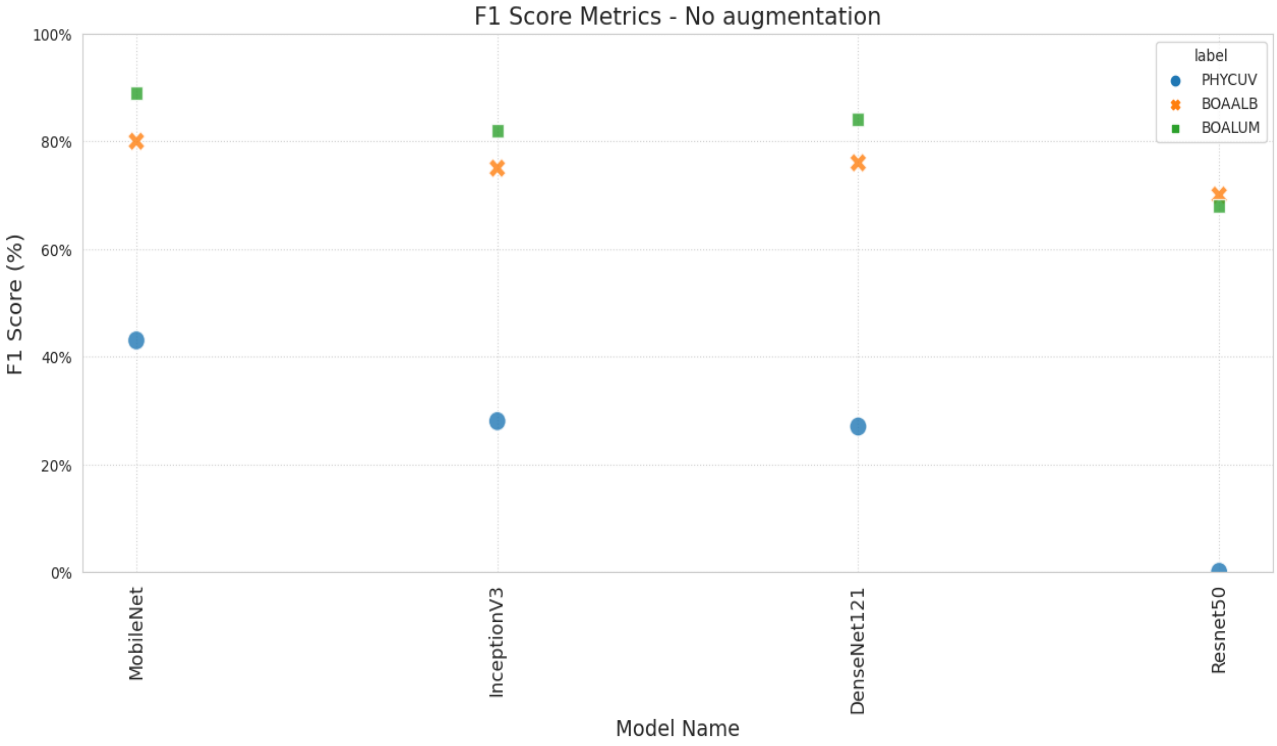

Figure 7.4: F1 – Score of the best models trained using the original dataset, analyzing this graph can be notice that the best performance was achieved by Mobilenet model, the worst label classified was PHYCUV, this could be due to the unbalance in the dataset.

In this result, particularly is evident the unbalanced characteristic of the dataset, as was shown previously, the PHYCUV samples were less in a matter of quantity, leaving the model with a very small group of images to learn from, while the BOAALB and BOALUM classification metrics were always really close between them and also achieving a greater F1 score. The reason can be several things, like how easy to distinguish are the calls, the sample quantity, and fewer mistakes made by the experts in annotations.
Here is the mean and standard deviation by each label from all the models of this stage and trained with the original dataset: All the models that returned the best metrics, were the ones trained with

Table 7.5: Samples count in augmented time-stretching and combined dataset

Species acronym	Time Stretching Dataset	Combined Dataset
PHYCUV	1290	2073
BOAALB	1871	2239
BOALUN	950	1372

the one fully connected layer architecture modification

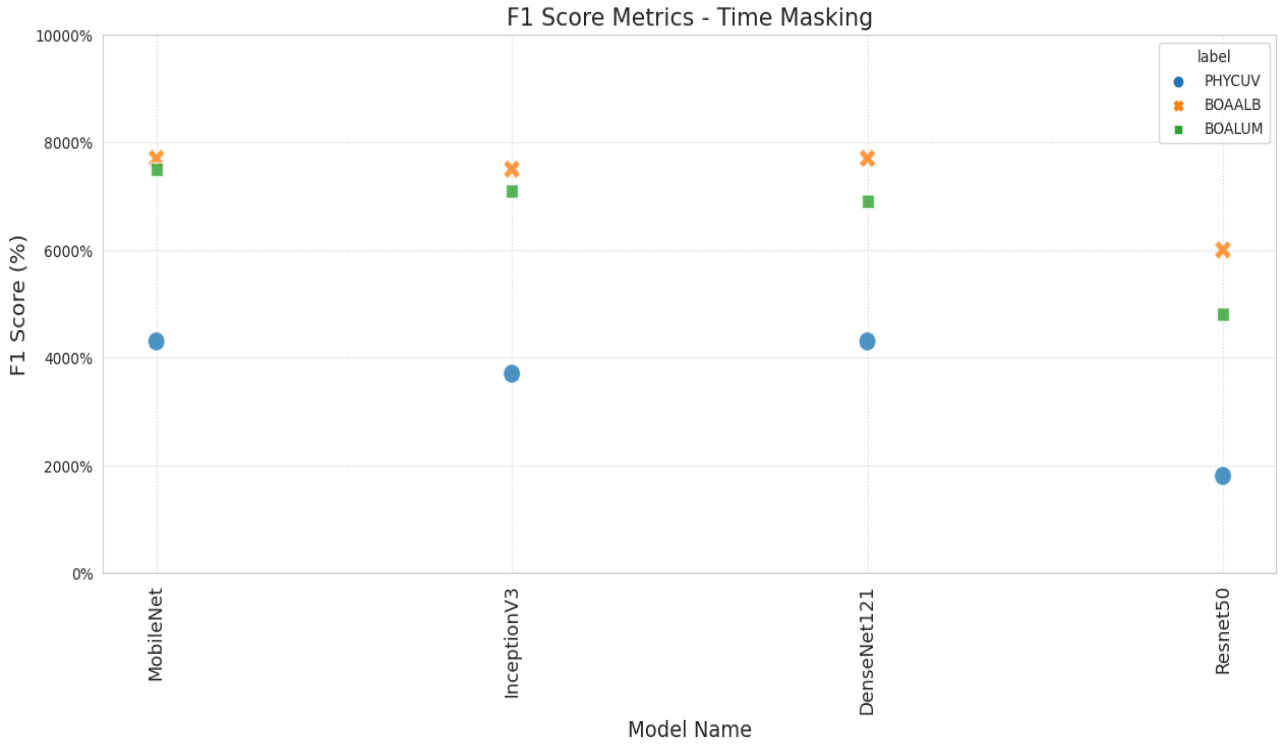

Figure 7.5: F1 – Score of the best models trained using the augmented time mask, this time Mobilenet model is still the best one over the rest and an improvement over the average classification effectiveness of the label PHYCUV can be noticed but a decrement in the other samples metrics is also noticed

The dataset used to train these models was the closest to a balanced one, despite this, the results were not as good as expected, this augmentation technique was the best of the two implemented, the best example is Resnet50, as you can visualize in the graph, the metrics score for label PHYCUV were increased, but the other labels were affected negatively. The best models were the ones that were trained using cross-validation, specifically the ones with K fold = 2

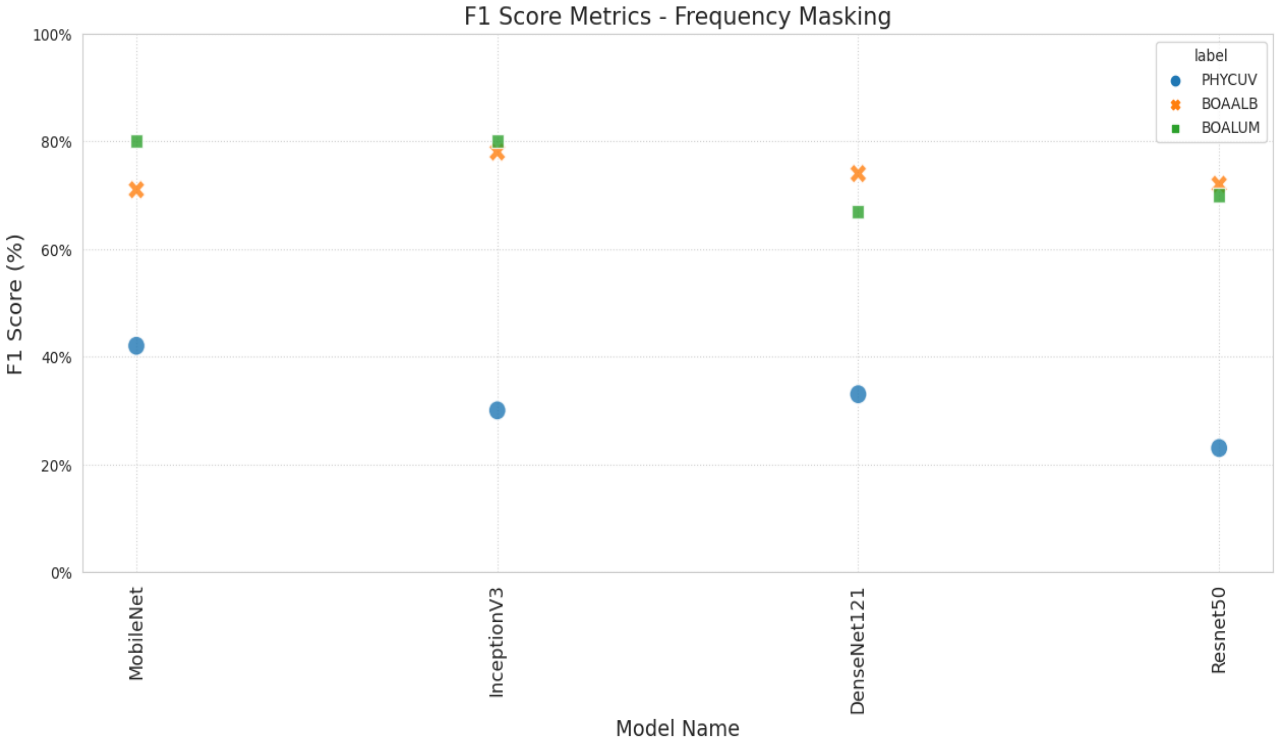

Figure 7.6: F1 – Score of the best models trained using the Augmented frequency mask, in this case the model InceptionV3 achieve the highest F1-Score for the BOAALB and BOALUM however for the PHYCUV label the MobileNet model achieved a better performance and a similar performance for BOALUM and BOAALB. In comparison with the time Masking augmentation, no significant improvements were made in the performance over the labels except from the ResNet50 model which had a notable increase for PHYCUV.

For this graphic analysis, is important to highlight that even if the dataset wasn't as balanced as the previous one, the results obtained are not totally bad, even the effect of PHYCUV was positive, the best model as it was in previous experiments have been the MobileNet. The best models in this experiment were mainly the ones produced by the k-fold cross-validation training.

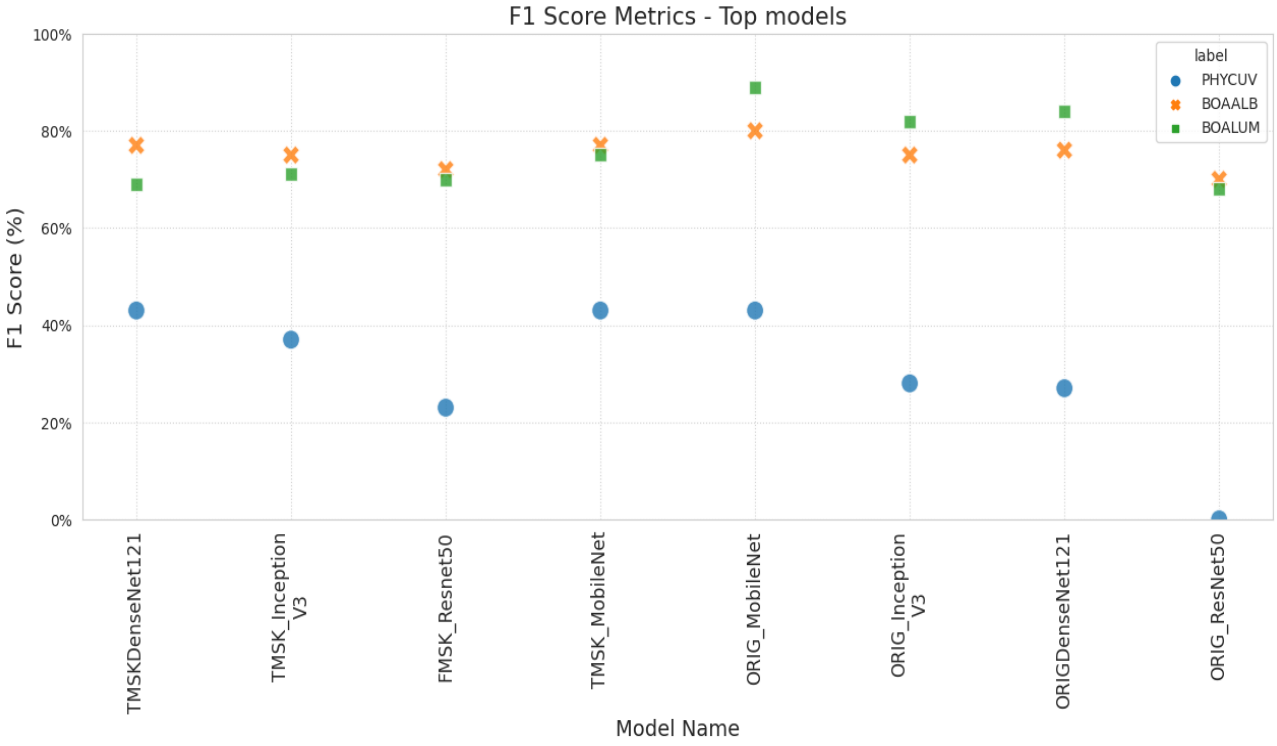

Figure 7.7: F1 – Score of best models of the stage, in this performance evaluations the models trained in the augmentation techniques known as time masking (TMSK), frequency masking(FMSK) and the ones trained with no augmentation techniques labeled as ORIG. There's a clear tendency over the BOAALB and BOALUM labels achieving metric values between 65 and 90 percent of effectiveness in the classification process.

Findings:
After reviewing these graphs and all the metrics from the trained models, some early insights came up:

- MobileNet trained models have the best performance overall trained models in stages 1 and 2.

- Adding 2 fully connected layers achieves a better performance in the models over other architectures while using a not-augmented dataset and time mask-augmented dataset.

- The combination of Adam optimizer, binary cross-entropy, and early-stopping has been demonstrated to be a useful combination to evaluate and improve the real performance of the trained models.

7.4 Models training stage 3

In this stage, the objective was to evaluate the performance of pre-trained models (Inception V3, DenseNet121, MobileNet, and ResNet50) on two new datasets following almost the same experimental structure made in Stage 2.
One dataset was built with the time-stretching augmentation technique, and the second dataset comprised the original data along with all the previous augmented datasets (time-mask, frequency-mask, and time-stretching techniques). Then determine if this new augmentation technique will improve the performance metrics and if the mix of all datasets in the training can also improve the metrics of the models.

Here are the samples per dataset obtained:

Table 7.6: Samples count in augmented time-stretching and combined dataset.

Species acronym	Time Stretching Dataset	Combined Dataset
PHYCUV	1290	2073
BOAALB	1871	2239
BOALUN	950	1372

Hypothesis:

- The increase in the number of samples in the dataset, thanks to the time-stretching augmentation technique, will result in an improvement in the performance of the models.
- The new dataset, composed of the original dataset combined with various augmented datasets, will result in more robust models, leading to an improvement in performance metrics compared to the previous experiments.

By training these models on the combined dataset, improvements in their performance metrics were expected, indicating their robustness and generalization capabilities had increased. The specifications for the architecture modifications, specifically the addition of fully connected layers, as in the previous experiment, can be found above :

- A Dense layer with 256 units, ReLU activation, and L2 regularization (with a coefficient of 0.01) follows the Flatten layer.
- A Dense layer with 128 units, ReLU activation, and L2 regularization (with a coefficient of 0.01) follows the Flatten layer.

- To mitigate overfitting, a Dropout layer with a dropout rate of 0.5 is included after each new layer is added.

The models trained in this stage will have the following fixed parameters as in the previous Stage:

- Adam optimizer with learning rate 0.00001.
- Drop out (0.5).
- L2 Regularization.
- Top epochs 1000.
- Early stopping patience = 5.
- Loss: Binary Crossentropy.
- Top layer input shape (224,224,3).
- Weights: Imagenet.
- Output layer Activation: sigmoid.

And the following parameters per model will be variated:

1. Dataset: Time-Streching augmented and combined (Time masking, Frequency masking, Time-Streching and not augmented).
2. Number of extra fully-connected layers (1 or 2).
3. Cross-validation 5 folds for Time-Streching Dataset.
4. Cross-validation 2 folds for Combined Dataset.

As a summary of what was the training process per model (Resnet50, InceptionV3, Densenet121, MobileNet):

1. Dataset: Time-Stretching augmented
    - One fully connected Layer
        - CrossValidation-5folds

  - Normal Training

- Two fully connected Layer

  - CrossValidation-5folds
  - Normal Training

2. Dataset: Combined (time masking, frequency masking, time-stretching and not augmented)

- One fully connected Layer

  - CrossValidation - 2 folds
  - Normal Training

- Two fully connected Layer

  - CrossValidation - 2 folds
  - Normal Training

Resulting in a total of 9 trained models per selected model (Resnet50, InceptionV3, Densenet121, MobileNet) while training with the time-streching dataset and 6 in the Combined dataset, altogether making a total of 60 trained models.

Here are the best F1-score performing models in this stage, trained on the Time-Stretching dataset :

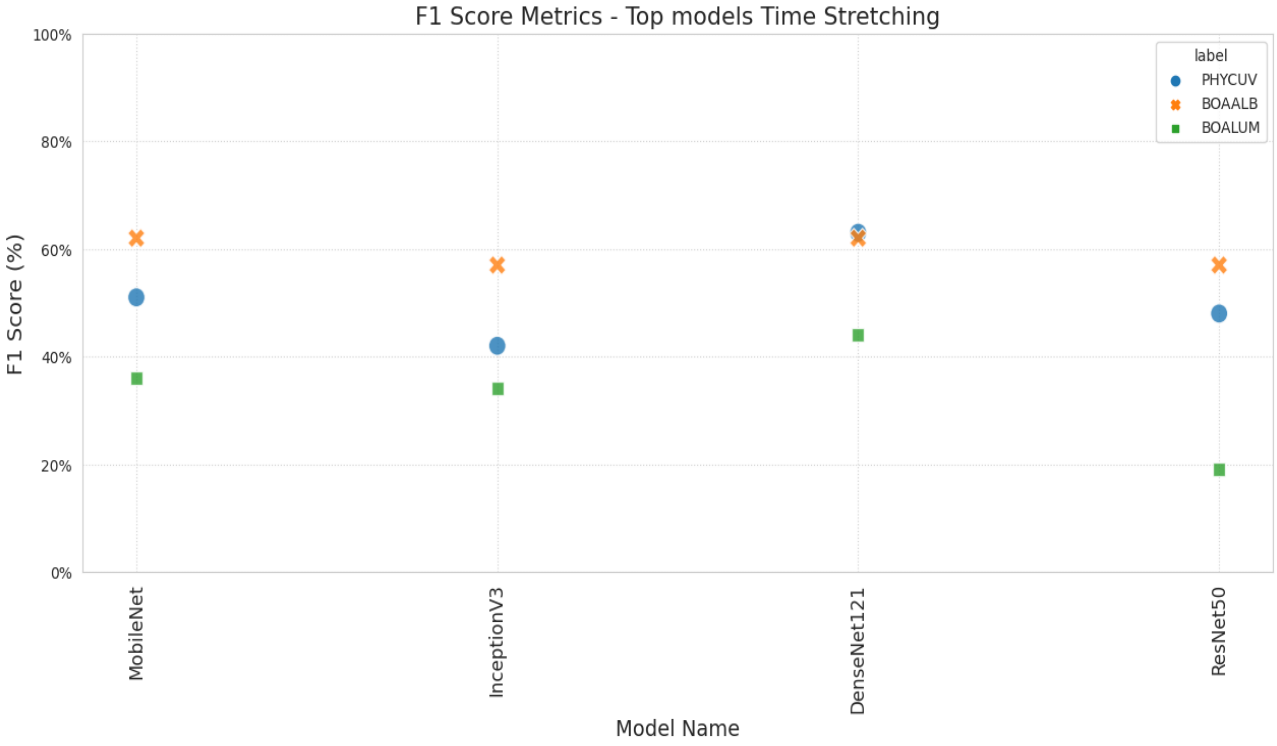

Figure 7.8: F1 – Score of the best models trained using the Augmented time Stretching dataset, this time a great improvement over the PHYCUV prediction is noticeable, however the other labels are being negatively affected.

From the given data, we can observe that for most models, there was a slight increase in the F1-Score for BOAALB and PHYCUV, while there was a significant improvement in the F1-Score for BOALUN. The model that showed the highest overall F1-Score for the three labels was DenseNet121_REG_L2_2LYR_Lr_00001_fold_0, with an F1-Score of 0.63 for PHYCUV, 0.62 for BOAALB, and 0.44 for BOALUN. This suggests that the time-stretching augmentation technique contributed positively to the model's performance in identifying the PHYCUV and BOAALB labels. However, it had a limited impact on the identification of the BOALUN label.

Overall, comparing the different data augmentation techniques, it is evident that each technique has different effects on the model's performance in identifying different labels. Time stretching showed promising results for PHYCUV and BOAALB, frequency masking enhanced the identification of BOALUN, and time masking demonstrated positive impacts on BOAALB and BOALUN recognition.

Therefore, to optimize model performance for all labels, a combination of these augmentation

techniques can be considered. Further experimentation and analysis are recommended to explore additional augmentation techniques or combinations that may yield even better results.

Here are the best F1-score performing models in this stage, trained on the Combined dataset :

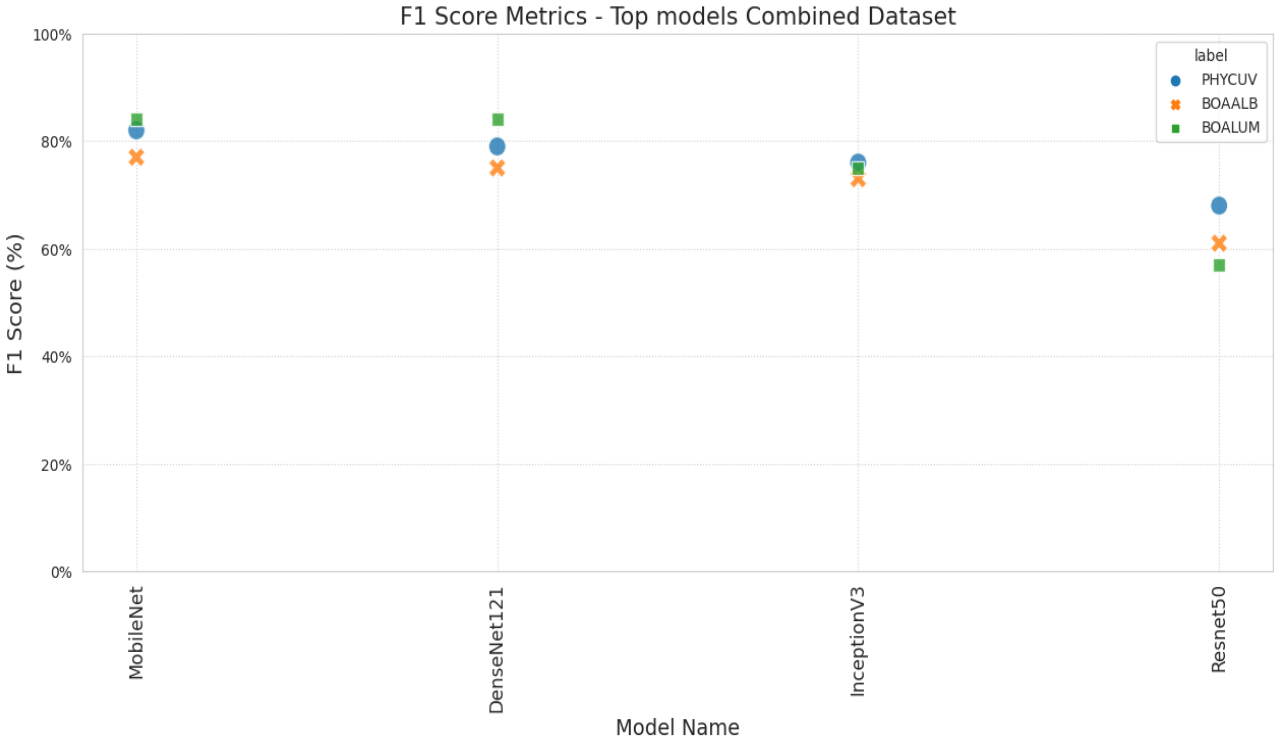

Figure 7.9: F1 – Score of the best models trained using the Combined dataset, here an overall improvement on all the labels to predict is shown, demonstrating that the combination of all the augmentation techniques represents an improvement for the trained models.

These results indicate that the MobileNet model with one fully connected layer exhibited the highest F1-Score values for the PHYCUV, BOAALB, and BOALUN species multi-label classification task, followed closely by DenseNet121 with one layer fold. Inception V3 with two layers and k fold = 0 and ResNet50 with one layer displayed relatively lower F1-Score values.

About overfitting during the training process, the fact of testing the trained models with unseen data and obtaining a good classification result, is the best proof to determine the absence of overfitting.

Here are two tables to compare the standard deviation and the mean of all metrics extracted

of the models trained with the combined dataset using cross-validation table 7.7 and using normal training and cross-validation table 7.8, the table highlights the significance of cross-validation as a valuable tool for assessing the stability and robustness of model predictions. By calculating the standard deviation of performance metrics across multiple folds, important insights can be obtained. Notably, a smaller variability indicates greater reliability and consistency in the predictions made by the models. In this particular cases, can be noticed that Resnet50 performs worst over all models and according to the standard deviation MobileNet and Densenet121 good candidate to be used for real applications.

Table 7.7: Standard deviation and median from all the models trained with the combined dataset using cross-validation.

Cross Validation		
Model	Median	Standard Deviation
MobileNet	66%	0.149756441
DenseNet 121	65%	0.147942536
Inception V3	67%	0.15302091
Resnet50	54%	0.219188461

Table 7.8: Standard deviation and median from all the models trained with the combined dataset using cross-validation and normal training.

Model	Median	Standard Deviation
MobileNet	77%	0.140417798
DenseNet 121	65%	0.146648622
Inception V3	66%	0.147278523
Resnet50	56%	0.2214677

Based on Figure 7.9, an analysis was performed according to the comparison between the model's weights and their average F1 score result, this analysis was made over this graph because these were the best results achieved.

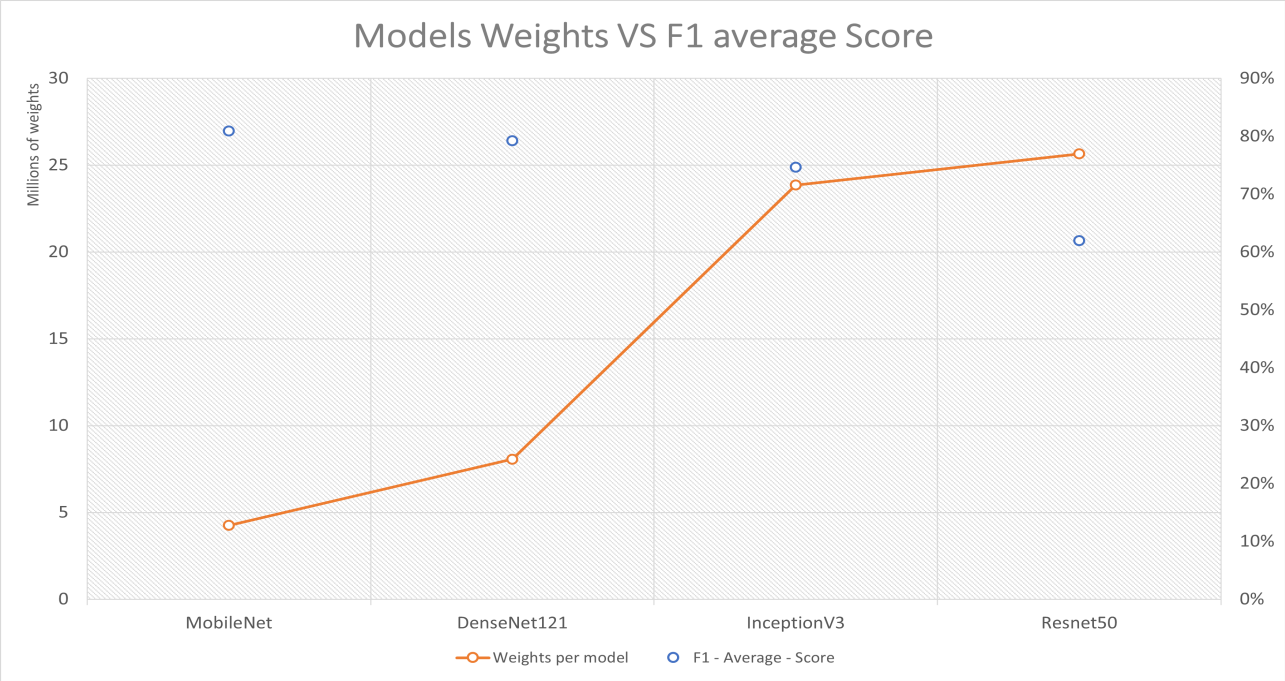

Figure 7.10: Model Weights VS F1 – Average Score – combined dataset, in this graph the relationship between each model weights and the model's trained with the combined dataset average performance, showing the best performance was achieved by the model with fewer weights.

As can be noticed, while the model weights increase, the average performance starts to decrease. An explanation is that the model with fewer weights is less likely to overfit the data, this can happen when a model has too many parameters, allowing it to learn the noise in the data as well as the signal intended to classify, so a model with fewer weights is less likely to overfit because it has fewer parameters to learn. Therefore, the model with a higher number of parameters requires more data samples to decrease overfitting.

Best model architecture detail: The best model architecture is based on the MobileNet architecture, which has been modified by adding custom layers on top. The model consists of several convolutional layers followed by batch normalization and ReLU activation. These layers are responsible for learning hierarchical features from the input images. The model also includes depthwise separable convolutions, reducing computational complexity while maintaining the representative capacity. The architecture further incorporates global average pooling to reduce spatial dimensions and a fully connected layer with 256 units and ReLU activation. Dropout regularization is applied to prevent overfitting. Finally, the model has an output layer with sigmoid activation to perform multi-label classification.

The key aspect of this model's success lies in the combination of augmented datasets, specifically

employing time stretching, time masking, and frequency masking techniques. By augmenting the training data, the model is exposed to a wider variety of patterns and variations, enabling it to generalize better to unseen examples. The model's hyperparameters have been tuned to optimize its performance on the multi-label classification task. The model has been trained using the Adam optimizer with a learning rate of 0.00001 and evaluated using F1 score metric. The achieved metrics on the test set demonstrate its effectiveness, with PHYCUV achieving 82% F1 score metric, BOAALB achieving 77% F1 score metric, and BOALUM achieving 84% F1 score metric. These results highlight the model's ability to accurately classify the target species based on their mel-spectrograms, demonstrating its potential to be used in acoustic monitoring and species classification applications.
Here is a detail of the model architecture:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_2 (InputLayer)         [(None, 224, 224, 3)]     0

 conv1 (Conv2D)               (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizatio  (None, 112, 112, 32)      128
 n)

 conv1_relu (ReLU)            (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288

 conv_dw_1_bn (BatchNormaliz  (None, 112, 112, 32)      128
 ation)

                              .
                              .
                              .
                              .
                              .

 conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0

 flatten_1 (Flatten)          (None, 50176)             0

 dense_2 (Dense)              (None, 256)               12845312

 dropout_1 (Dropout)          (None, 256)               0

 dense_3 (Dense)              (None, 3)                 771

=================================================================
Total params: 16,074,947
Trainable params: 12,846,083
Non-trainable params: 3,228,864
_________________________________________________________________
```

Figure 7.11: Best model top and out layers brief summary.

Chapter 8

Conclusions and recommendations

The study's purpose was to demonstrate the applicability of AI models in the acoustic monitoring data classification process through the experimentation and analysis of pre-trained Convolutional Neural Networks (CNNs) in the context of transfer learning and multi-label classification to identify by their sound in a time-frequency representation (Spectrograms) of three specific anuran species (Boana albopunctata, Physalaemus cuvieri, and Boana lundii), present on the site denoted as INCT41, located in the Bifurcação locality on the Cerrado biome in Brazil, the obtained results show that is possible and appropriate to implement pre-trained CNN models to classify anuran calls into the acoustic monitoring field, in this case the best performance achieved had an average F1-Score of 81% with a maximum standard deviation of 0.039, indicating that a proper architecture and hyperparameters were selected to achieve this task, there was identified a relation between the models weights and its performance. Therefore, it is important to conclude that due to the imbalance on the amount of samples provided and the size of the dataset generated, augmentation were used and the combination of these techniques to build a dataset result in more robust models and better performance over all.

Throughout this project, several valuable insights were gained, offering guidance for future works within the same thesis theme. Firstly, among the four pre-trained Convolutional Neural Networks (CNNs) employed, MobileNet and DenseNet121 consistently exhibited superior performance. Notably, these CNNs also possessed fewer weight parameters, suggesting their efficiency. Additionally, it was observed that combining multiple augmentation techniques into one dataset can significantly enhance model performance and robustness. Moreover, to further refine these models, it is recommended to generate new augmented datasets utilizing different augmentation techniques and replicate the experiments, keeping in mind that to mitigate overfitting, it is advised to employ early stopping, model checkpoint, regularization, and dropout techniques from the onset of the experimental training process. Lastly, exploring alternative pre-trained CNNs with fewer weight parameters and comparing the obtained F1 scores with those achieved in this project could provide valuable insights for future researchers. Finally, this work provides a walkthrough for audio processing, multi-label classification and model training for three specific species identification(Boana albopunctata, Physalaemus cuvieri, and Boana lundii). In addition, the paper highlights the limitations associated with the implementation of deep learning models for acoustic monitoring and suggests solutions to overcome them. Overall, this paper provides useful guidance for researchers and practitioners working on acoustic monitoring of species and can help improve the effectiveness and efficiency of acoustic monitoring programs in conservation and ecology.

Chapter 9

Appendices

Attachment 1 – Links

GitHub Repository

https://github.com/SDV244/TESIS-VScode

Models Drive

https://drive.google.com/drive/folders/1-FrYv5gDVSJ2NitxHwGnQIMweEXfiilL?usp=share_link

Colab notebook used for training the models:

https://colab.research.google.com/drive/10VlyLztoJith66moBENbkqZUE1N8t-II?usp=sharing

Bibliography

[1] J. Strout, B. Rogan, S. M. Seyednezhad, K. Smart, M. Bush, and E. Ribeiro, "Anuran call classification with deep learning," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2662–2665.

[2] M. Zhong, J. LeBien, M. Campos-Cerqueira, R. Dodhia, J. L. Ferres, J. P. Velev, and T. M. Aide, "Multispecies bioacoustic classification using transfer learning of deep convolutional neural networks with pseudo-labeling," *Applied Acoustics*, vol. 166, p. 107375, 2020.

[3] D. Dechmann and M. Vences, "Anuran bioacoustics: A call for more data," *Frontiers in Ecology and Evolution*, vol. 5, p. 156, 2017.

[4] F. Glaw, J. Köhler, and M. Vences, "Anuran conservation: A global overview," *Biological Conservation*, vol. 176, pp. 191–204, 2014.

[5] Y. Chen, Q. Li, and L. Zhang, "A deep learning approach for anuran call classification," *Ecological Informatics*, vol. 47, p. 100910, 2019.

[6] J. H. Kotir, "Climate change and variability in sub-saharan africa: a review of current and future trends and impacts on agriculture and food security," *Environment, Development and Sustainability*, vol. 13, no. 3, pp. 587–605, 2011.

[7] M. C. Urban, G. Bocedi, A. P. Hendry, J.-B. Mihoub, G. Pe'er, A. Singer, J. Bridle, L. Crozier, L. De Meester, W. Godsoe *et al.*, "Improving the forecast for biodiversity under climate change," *Science*, vol. 353, no. 6304, p. aad8466, 2016.

[8] D. B. Wake and V. T. Vredenburg, "Are we in the midst of the sixth mass extinction? a view from the world of amphibians," *Proceedings of the National Academy of Sciences*, vol. 105, no. supplement_1, pp. 11 466–11 473, 2008.

[9] E. Dufourq, C. Batist, R. Foquet, and I. Durbach, "Passive acoustic monitoring of animal populations with transfer learning," *Ecological Informatics*, vol. 70, p. 101688, 2022.

[10] J. Korstanje, "The f1 score," Aug 2021. [Online]. Available: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6

[11] A. Henríquez, J. B. Alonso, C. M. Travieso, B. Rodríguez-Herrera, F. Bolaños, P. Alpízar, K. López-de Ipina, and P. Henríquez, "An automatic acoustic bat identification system based on the audible spectrum," *Expert Systems with Applications*, vol. 41, no. 11, pp. 5451–5465, 2014.

[12] J. J. Noda, C. M. Travieso, and D. Sanchez-Rodriguez, "Methodology for automatic bioacoustic classification of anurans based on feature fusion," *Expert Systems with Applications*, vol. 50, pp. 100–106, 2016.

[13] E. Akbal, S. Dogan, and T. Tuncer, "An automated multispecies bioacoustics sound classification method based on a nonlinear pattern: Twine-pat," *Ecological Informatics*, vol. 68, p. 101529, 2022.

[14] E. Browning, R. Gibb, P. Glover-Kapfer, and K. Jones, *Passive acoustic monitoring in ecology and conservation*, 10 2017.

[15] A. Farina and S. H. Gage, *Ecoacoustics: The ecological role of sounds.* John Wiley & Sons, 2017.

[16] T. M. Aide, C. Corrada-Bravo, M. Campos-Cerqueira, C. Milan, G. Vega, and R. Alvarez, "Real-time bioacoustics monitoring and automated species identification," *PeerJ*, vol. 1, p. e103, 2013.

[17] D. T. Blumstein, D. J. Mennill, P. Clemins, L. Girod, K. Yao, G. Patricelli, J. L. Deppe, A. H. Krakauer, C. Clark, K. A. Cortopassi, S. F. Hanser, B. McCowan, I. Ali, and A. N. G. Kirschel, "Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus," *Journal of Applied Ecology*, vol. 48, no. 3, pp. 758–767, 2011.

[18] J. Sueur and A. Farina, "Ecoacoustics: The ecological investigation and interpretation of environmental sound," *BioScience*, vol. 68, no. 6, pp. 456–468, 2018.

[19] D. Stowell, M. D. Wood, H. Pamuła, Y. Stylianou, and H. Glotin, "Automatic acoustic detection of birds through deep learning: The first bird audio detection challenge," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 368–380, 2019.

[20] N. D. Merchant, K. M. Fristrup, M. P. Johnson, P. L. Tyack, M. J. Witt, P. Blondel, and S. E. Parks, "Measuring acoustic habitats," *Methods in Ecology and Evolution*, vol. 6, no. 3, pp. 257–265, 2015.

[21] A. Pfau, "Multi-label classification of underwater soundscapes using deep convolutional neural networks," Dec 2020. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA364024.pdf

[22] F. meAjitesh KumarI have been recently working in the area of Data analytics including Data Science and K. A. Machine Learning / Deep Learning. I am also passionate about different technologies including programming languages such as Java/JEE, "Real-world applications of convolutional neural networks," Nov 2021. [Online]. Available: https://vitalflux.com/real-world-applications-of-convolutional-neural-networks/

[23] W.-C. Yeh, Y.-P. Lin, Y.-C. Liang, and C.-M. Lai, "Convolution neural network hyperparameter optimization using simplified swarm optimization," *arXiv preprint arXiv:2103.03995*, 2021.

[24] "Difference between Model Parameter and Hyperparameter - Javatpoint — javatpoint.com," https://www.javatpoint.com/model-parameter-vs-hyperparameter, [Accessed 18-Jul-2023].

[25] X. Zhou, Y. Wu, and J. Luo, "A survey of cross-validation methods for convolutional neural networks," *arXiv preprint arXiv:2201.07669*, 2022.

[26] J. Blanch, "What is transfer learning? exploring the popular deep learning approach." Mar 2020. [Online]. Available: https://builtin.com/data-science/transfer-learning

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[28] A. links open overlay panelIbrahem Kandel and A. hyperparameters have to be tuned to have a robust convolutional neural network that will be able to accurately classify images. One of the most important hyperparameters is the batch size, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959519303455

[29] "deep learning." [Online]. Available: https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-hdstat-rnn-deep-learning.pdf

[30] J. Teuwen and N. Moriakov, "Convolutional neural networks," in *Handbook of medical image computing and computer assisted intervention*. Elsevier, 2020, pp. 481–501.

[31] S. Yann, "L1 and l2 regularization explained," May 2020. [Online]. Available: https://towardsdatascience.com/l1-and-l2-regularization-explained-874c3b03f668

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Rectified linear units improve neural network acoustic models," *arXiv preprint arXiv:1505.00853*, 2015.

[33] A. Bajaj, "Performance metrics in machine learning [complete guide]," Jul 2022. [Online]. Available: https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide

[34] "Imagenet," http://www.image-net.org, [Accessed: May 9, 2023].

[35] Matlab, "Resnet-50." [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/resnet50.html#mw_7e914435-71af-41f7-9ec9-8f2092892982_sep_mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe

[36] MathWorks, *Inception-v3 Convolutional Neural Networks*, MathWorks, 2023, [Accessed: May 9, 2023]. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/inceptionv3.html

[37] Pluralsight, "Introduction to densenet with tensorflow," https://www.pluralsight.com/guides/introduction-to-densenet-with-tensorflow, 2023, [Accessed: May 9, 2023].

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[40] M. Sankupellay and D. Konovalov, "Bird call recognition using deep convolutional neural network, resnet-50," in *Proceedings of ACOUSTICS*, vol. 7, no. 9, 2018, pp. 1–8.

[41] D. B. Efremova, M. Sankupellay, and D. A. Konovalov, "Data-efficient classification of birdcall through convolutional neural networks transfer learning," in *2019 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2019, pp. 1–8.

[42] M. Zhong, M. Castellote, R. Dodhia, J. Lavista Ferres, M. Keogh, and A. Brewer, "Beluga whale acoustic signal classification using deep learning neural network models," *The Journal of the Acoustical Society of America*, vol. 147, no. 3, pp. 1834–1841, 2020.

[43] Y. Gong, S. Khurana, A. Rouditchenko, and J. Glass, "Cmkd: Cnn/transformer-based cross-model knowledge distillation for audio classification," *arXiv preprint arXiv:2203.06760*, 2022.

[44] E. Tsalera, A. Papadakis, and M. Samarakou, "Comparison of pre-trained cnns for audio classification using transfer learning," *Journal of Sensor and Actuator Networks*, vol. 10, no. 4, p. 72, 2021.

[45] M. Heil, G. Tsanaktsidis, and M. Weller, "Towards a unified framework for natural language processing with graph neural networks," *arXiv preprint arXiv:2102.00797*, 2021.

[46] V.-V. Eklund, "Data augmentation techniques for robust audio analysis," *Trepo*, 2020. [Online]. Available: https://trepo.tuni.fi/bitstream/handle/10024/117251/EklundVille-Veikko.pdf?sequence=2

[47] Steinberg. (2023) Time stretching in wavelab elements. [Online]. Available: https://steinberg.help/wavelab_elements/v11/en/wavelab/topics/offline_processing/time_stretching_c.html

[48] J. Park, T. Chan, S. Lee, D. Kim, J. Kim, and S. Heo, "Specaugment: A simple data augmentation technique for automatic speech recognition," *arXiv preprint arXiv:2004.08914*, 2020.

[49] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "scikit-learn: Machine learning in Python," https://scikit-learn.org, 2011, accessed: May, 2023.

[51] N. AI, "Hyperparameter tuning in python: a complete guide," *neptune.ai*, April 2023. [Online]. Available: https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide

[52] J. Brownlee, "How to configure the learning rate when training deep learning neural networks," Aug 2019. [Online]. Available: https://machinelearningmastery.com/.

[53] D.-M. L. M. Simple, "How does batch size impact your model learning," May 2023. [Online]. Available: https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa

[54] N. N/A, "What is epoch in machine learning?" Apr 2023. [Online]. Available: https://intellipaat.com/blog/epoch-in-machine-learning/?US#:~:text=https://intellipaat.com/blog/epoch-in-machine-learning

[55] U. Tewari, "Regularization understanding l1 and l2 regularization for deep learning," Nov 2021. [Online]. Available: https://medium.com/analytics-vidhya/regularization-understanding-l1-and-l2-regularization-for-deep-learning

[56] S. Lookwood, "Convolutional neural network: Benefits, types, and applications," Apr 2023. [Online]. Available: https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/

[57] A. Nisha, "Why use k-fold cross validation?" Jul 2022. [Online]. Available: https://www.kdnuggets.com/2022/07/kfold-cross-validation.html#:~:text=Choosin.