Pontificia Universidad
JAVERIANA
Cali

Facultad de Ingeniería
y Ciencias
Ingeniería Electrónica

Monografía de Trabajo de Grado

APPLICATION OF DATA AUGMENTATION METHODS IN TRANSFER LEARNING ALGORITHMS TO IDENTIFY AMPHIBIAN SPECIES IN BIOACOUSTIC SIGNALS

Adriana Lucía Melo Ordóñez

Director
Dr. Luis Eduardo Tobón Llano

September 16, 2024

Acknowledgements

I want to express my sincere gratitude to my advisor, *Luis Eduardo Tobon*. His guidance aided me in constructing the experiments, describe them and to better understand the algorithms presented here. His assistance played a crucial role in the successful completion of this investigation. I am particularly grateful to him for his unwavering belief in and unconditional support of my project. Additionally, I would like to express my gratitude to the Engineering Faculty at *Javeriana University* for providing me with the necessary knowledge and preparing me to undertake my undergraduate project.

Additionally, I would like to express my heartfelt gratitude to my family for their unwavering support throughout this entire process. I am forever grateful to my mother, *Maria*, who has consistently been my inspiration, encouraging me to always move forward, give my utmost, and never surrender. My aunts, *Adriana* and *Carmen Elisa*, have also played a significant role in my journey, constantly showing their support and concern for my wellbeing. I am immensely thankful to my grandparents, *Eduardo* and *Elisa*, for their invaluable wisdom and the joyous moments we've shared. Last but not least, I must acknowledge my beloved cat, *Frida*, who has been a source of solace during moments of stress.

Finally, I would like to express my heartfelt appreciation to my dear friends who stood by my side and supported me throughout this challenging journey. I am especially grateful to one of my best friends, *Tomás*, who has been the biggest source of support. Thank you for always believing in me, being my confidant, and sharing countless laughs and valuable advice. I also want to extend my gratitude to *Angelita*, *Gabriel*, and *Carlos*. Your encouragement and efforts to help me relax and have fun have meant a lot to me. And, in general, to all my friends who have always been there for me, believing in my abilities, I am truly grateful.

Glossary

Símbolos

α	denotes a parameter or factor, which may or may not be random, that multiplies a function.
β	employed to represent a random factor that multiplied a function.
Φ	symbolizes an equalizing function from EMDA.
θ	represents the parameter vector of a model.
ψ	used to present a set of parameters from EMDA.

Acrónimos y Abreviaturas

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
dB	Decibels
MFCCs	Mel-Frequency Cepstral Coefficients
PAM	Passive Acoustic Monitoring
ReLU	Rectified Linear Units
RNN	Recurrent Neural Network
TB	Terabyte
TCN	Temporal Convolutional Network
VGG	Visual Geometry Group

Términos

Anurans	amphibians classified under the anuran order, indicating the absence of a tail in their adult phase.
Bioacoustics	a discipline that studies the production, transmission, and reception of acoustic sounds produced by animals.
Convolutional Neural Network	a type of deep learning neural network architecture based on convolutional layers and filter optimization to learn features.
Data augmentation	a technique that applies certain distortions to different types of signals (such as images, audio) with the intention to generate a new artificial signal.
Dynamic range	the ratio between the largest and the smallest value present in an audio signal, typically expressed in decibels.
Flops	an acronym for floating-point operations per second, serves as a benchmark for measuring the computational performance of a processor based on the volume of floating-point arithmetic calculations it can perform within a second.

Gradient	a vector that consists of partial derivatives that quantify the variation in all weights in relation to the change in error.
Gradient Descent	an optimization algorithm used in machine learning which uses a cost function to determine a local minimum of the cost function finding its optimal parameters.
Mel-spectrogram	a spectrogram that represents frequencies in the mel-scale, which is designed to approximate how humans perceive pitch.
MFCCs	coefficients used to represent the spectral characteristics of a sound, based on how humans perceive auditory signals. These coefficients are generated by dividing the frequency bands into sub-bands using the mel scale and then extracting the Cepstral Coefficients through the Discrete Cosine transform.
Multi-label classification	considered a generalization of multiclass classification, where multiple labels are assigned to a single sample, indicating that it can belong to more than one class simultaneously.
Passive Acoustic Monitoring	an observational tool that involves monitoring and characterizing wildlife by employing non-invasive sensors.
Torches	an essential component of the Torch library, encompassing n-dimensional arrays or tensors that support a series of operations such as indexing, slicing, resizing, cloning, and more.
Transfer learning	a machine learning technique that comprises using a pre-trained model or pre-acquired knowledge from a related task to enhance performance.
Vanishing gradient problem	an issue encountered during the training of neural networks where the gradients that update the network diminish to an extremely low value or vanish, preventing the network from learning.

Resumen

El calentamiento global y sus efectos se han establecido como asuntos importantes en la actualidad. Las consecuencias y evidencias del cambio climático deberían representar la urgencia de medidas más estrictas para prevenir secuelas irreversibles. De esta manera, es crucial reunir evidencia que corrobore el grado de efecto del calentamiento global, y el Monitoreo Acústico Pasivo, PAM en inglés, es un método para cumplir este objetivo. PAM puede supervisar especies que se encuentran en riesgo de extinción y que también son especialmente sensibles a los cambios de temperatura como es el caso de los anuros. Consecuentemente, estas especies son fundamentales en determinar el impacto del calentamiento global y la escala de urgencia para abordarlo.

El estudio y supervisión de señales, reunidos de la aplicación de PAM puede implicar un desafío debido a la extensa cantidad de horas de datos que se necesitan analizar, lo que puede ser una tarea demandante y que consume mucho tiempo. Entonces, el uso de Machine Learning aparece como una herramienta efectiva para automatizar la identificación de señales bioacústicas y facilitar su estudio. Sin embargo, con el fin de alcanzar resultados excepcionales con algoritmos de Machine Learning se requieren una cantidad de datos considerables, la cual no siempre puede estar disponible. Con el objeto de afrontar la falta de datos y mejorar el desempeño de los algoritmos, técnicas como la aumentación de datos y el aprendizaje por transferencia han sido desarrolladas.

Este trabajo de grado pretende probar la eficacia de estas dos técnicas para clasificar espectrogramas multi-etiqueta generados de llamados de especies de anuros. Los experimentos involucraron comparar el desempeño de tres arquitecturas de redes neuronales convolucionales (ResNet, VGG y EfficientNet) en dos bases de datos. Los experimentos concluyeron que EfficentNet obtuvo los resultados más significativos, consiguiendo en promedio un F1-score de 0.83 cuando se uso junto con la aumentación de datos y el aprendizaje por transferencia.

Palabras Clave: aprendizaje automático, aprendizaje por transferencia, aumentación de datos, clasisficación multi-etiqueta, aprendizaje profundo, bioacústica, clasificación de anuros.

Abstract

Global warming and its impacts has been firmly established as an important current topic. The consequences and evidence of climate change should represent the urgence of stringent measures to avoid more irreversible sequels. Consequently, it is crucial to gather evidence that substantiates the extent of global warming, and employing Passive Acoustic Monitoring (PAM) is one method to accomplish this objective. PAM can follow species that are at a high risk of extinction and that also are especially sensitive to changes in temperature, such as the case of anurans. Consequently, this species can be instrumental in assessing the impact of global warming and determining the level of urgency to address it.

The study and surveillance of the signals gathered from PAM can signify a challenge due to the extensive quantity of hours of data, which can be a time-consuming and demanding task. Then, the use of machine learning emerges as an effective tool for automating the identification of acoustic signals and facilitating its study. However, in order to achieve exceptional results with machine learning algorithms, a substantial amount of data containing significant information is typically required, which may not always be available. In order to address the insufficient data and enhance the performance of machine learning algorithms, techniques like data augmentation and transfer learning have been developed.

This undergraduate project aims to test the efficacy of these two techniques to classify multi-label spectrogram calls from anuran species. The experimentation involved comparing the performance of three CNN architectures (ResNet, VGG, and EfficientNet) on two datasets, to which different audio and spectrogram augmentation techniques were applied to these datasets. The experiments concluded that EfficientNet yielded the most significant results, attaining an averaged F1-score of 0.83 when coupled with transfer learning and augmentations.

Keywords: machine learning, transfer learning, data augmentation, multi-label classification, deep learning, bioacustics, anuran classification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

According to the Red List of Threatened Species [1], amphibians have a high percentage of species that are threatened by extinction. Anurans have a high sensitivity to changes in temperature in their environment. It is worth noting that temperature plays a crucial role in the lives of anurans. It not only affects their body temperature [2], but also determines the environmental conditions during the breeding season[3]. Additionally, temperature can impact their sexual differentiation [4].Therefore, any fluctuation in temperature could potentially impact the behavior and survival of species Furthermore, the absence of certain species can have a profound effect on the ecosystem, compromising diversity and potentially endangering other species. This underscores the crucial importance of preserving every single species. Given the large number of species currently at risk and their vulnerability to habitat changes, urgent action is imperative to safeguard their future, which would imply to gather data that serves as evidence to check the urgence in certain habitat to apply more stringent measures.

Passive Acoustic Monitoring (PAM) is a widely used technique for monitoring species populations and dynamics [5]. It involves the deployment of non-invasive sensors in inaccessible areas where human presence can complicate data collection. By utilizing this method, it becomes possible to study species that are difficult to observe directly. The process entails the use of sensors and devices with storage capabilities to record environmental data within a pre-defined time frame. The recorded data is then subjected to pre-processing to enhance signal clarity and extract useful features. This is a key focus within the field of bioacoustics. The objective of this approach is to analyze and extract specific features from signals, such as spectrograms, which can help calculate species density and identify patterns indicative of breeding behavior.

There are numerous challenges inside bioacoustics that require attention and can be utile to develop future work [5]. One of them is related to the extensive quantity of data that needs to be recorded and analyzed. As was suggested before monitoring and studying certain species can comprehend a considerable quantity of hours of recording, which Machine Learning, specially Convolutional Neural Network, appears as a suitable candidate to help in the preservation of anuran species as highlighted by Stowell [6], who also mentioned that there are still further work in development that could prove machine learning as a reliable tool for the classification and identification of bioacoustics signals.

One important consideration when using CNNs is that they typically require a large amount of data to achieve remarkable results. However, there are situations where this abundance of data is not available. In such cases, alternative techniques can be employed to achieve good performance even with a limited number of samples. These techniques, which have been used in this investigation and in other studies by Nanni et al. [7] and Sun et al. [8], include data augmentation and transfer

learning. Data augmentation involves applying slight distortions to the signals to generate new samples and provide additional context for the model to learn from and improve its capacity. In the case of spectrogram augmentation, were found that the augmentations could be applied to the spectrogram signal as well as to the audio signal [7][9].Meanwhile, transfer learning involves utilizing weights from a previously trained model to enhance the performance of the model during training it for a different task.

Another challenge observed in using CNNs and bioacoustics is that most of the works have focused on multi-class classification rather than multi-label classification. This means that a sample could belong to different classes simultaneously, which is a more realistic context considering that multiple species can coexist in a real habitat. Additionally, the works that have attempted multi-label classification, such as those by Cañas et al. [10], A and Rajan [11], Xie et al. [12], and Moummad et al. [13], did not primarily focus on the use of transfer learning and different types of data augmentation. However, they did provide a test to evaluate the potential of these techniques for multi-label spectrogram classification.

Therefore, the current investigation is focused on examining the efficacy of data augmentation and transfer learning in classifying multi-label spectrograms containing various calls from neotropical anuran species. The primary objective is to validate the capacity of machine learning in species identification and improving passive acoustic monitoring to preserve these species.

To showcase the process and outcomes of the previously described objective, the document is organized into several chapters. Chapter 2 provides an in-depth analysis of the problem statement, presenting the details of the issue at hand. Chapter 3 delves into the justification for this study, offering further insights into the project's impact and significance. Moving on to Chapter 4, the general and specific objectives of the investigation are defined, which will be pursued through the utilization of transfer learning and data augmentation techniques. The subsequent chapter, Chapter 5, introduces the theoretical framework, elucidating the critical concepts and theoretical background underlying the methods and components applied throughout the project. Chapter 6 then delves into the methods and resources employed, outlining the various stages involved in the implementation process, including model selection and the application of transfer learning. In Chapter 7, the obtained results from experimentation are examined and their usefulness in the classification of multi-label bioacoustic signals is discussed. Lastly, Chapter 8 and Chapter 9 respectively offer a comprehensive discussion on the conclusions and recommendations derived from this investigation.

Chapter 2

Problem Statement

In recent decades, the influence of human activities and industrialization produced notable effects in nature. Global warming has emerged as the predominant consequence of utmost concern among various experts and organizations. Its repercussions extend beyond detrimental effects on human health, as it also entails pernicious consequences for the environment. The fluctuations in habitat temperature resulting from global warming induce alterations in species' behavior, notably affecting breeding interactions. As a result, the decrease in population can lead to the extinction of the species. An investigation published by the National Academy of Sciences of the United States of America reported that, from 538 species studied, 57-70% may face extinction as a response to climate change [14]. Also, an article made by the Center of Biological Diversity proposes the alarming projection that over one third of Earth's animals and plant species will be extinguished by 2050 [15]. Amphibians are particularly vulnerable to climate conditions, making them at high risk of extinction across species. According to the Red List of Threatened Species, 41% of the species at risk of extinction are amphibians [1]. These alarming data brings the necessity of find tools to bring more evidence of how species activities have changed so it will be possible to generate more alerts and demand more severe politics to avoid more tragic consequences in world's diversity.

In this manner, Machine Learning has the potential to be utilized as a valuable tool in determining various aspects of species such as density, activity, population, etc. More importantly, its algorithms are efficient and precise, so analyzing the data typically acquired during monitoring will be time-consuming and labor-intensive for a person. As a result, by employing a powerful processor and a functional algorithm, it will be possible to achieve favorable results within a reasonable time frame. Although the process of designing, training, and testing the programs requires experts' knowledge. This was proved by Kahl [16] in his work implementing deep artificial neural networks for recognition of birds sounds. Furthermore, in a related study conducted by Acevedo et al. [17], the performance of various Machine Learning algorithms was evaluated to classify bird and amphibian species, resulting in precision percentages ranging from 71.45% to 94.95%. From these works, one can infer that Machine Learning has the potential to bring new and important possibilities to the bioacoustics field. However, it is important to contemplate that machine learning algorithms need a substantial amount of data to generate precise predictions. Then, data augmentation techniques can help to increase the quantity of data to improve the machine learning algorithm performance, as shown in the investigation of Nanni et al. [7]. The supplied data enables the algorithm to identify numerous cases accurately, resulting in a reliable prediction tool that saves time and resources by avoiding the need for additional data collection.

Therefore, this project will be a study of how different data augmentation methods will influence the performance of Transfer Learning algorithms to identify amphibian species in bio-acoustic signals

in order to evaluate its effectiveness in training an algorithm that has a high precision with a limited amount of data; in order to help the readers to understand the utility of data augmentation techniques and machine learning algorithms in the bioacoustics field, and also how it can help experts to determine species activities, which can be valuable data for the development to further works and discover significant information about actual questions like the effects of global warming in species interaction.

Chapter 3

Justification

The phenomenon of global warming and its correlation with human activities has resulted in significant alterations to various species on a global scale. These changes have directly impacted crucial factors, such as temperature, humidity, and other environmental variables, which are indispensable for the survival and reproduction of these species. Given that any change in their environment has the potential to affect their behaviors and restrict the reproductive atmosphere, resulting in a decrease in the number of species. Amphibians are susceptible to climate change. As ectotherms live in a tropical region, they are more exposed to higher limits and at more risk of changing their behavior [18]. As stated by the Red List of Threatened Species by the International Union of Conservation of Nature (IUCN), over 37,400 species are at risk of extinction, and 41% are amphibians [1]. Therefore, it has become a substantial issue for experts and environmental organizations to monitor different species' activity to ensure their reproduction and avoid extinction.

A variety of techniques have been implemented. Among the various techniques, passive acoustic monitoring has acquired significant popularity and is proven effective. Using non-invasive sensors, this approach gathers data without disturbing wildlife. As a result, experts can detect and analyze species behavior. Then, scientists can measure species occupancy, abundance, population, density, and composition. Although it is a powerful tool to access wildlife without intruding on species' habitats, several challenges exist with data analysis caused by the quantity and complexity of the information received. In some applications, like the one suggested in the project of Llusia et al.[19], the data recorded can reach over 6 TB of audio files. Examining the data individually would be an inefficient and impractical process. As a result, establishing a competent system becomes imperative for achieving continuous and precise monitoring. Different fields have implemented learning algorithms to make data analysis affordable, autonomous, and reliable. Studies implementing these algorithms achieve an 86% precision compared to a 68% reached by human analysis [20]. In another case, a machine learning algorithm was used to recognize frog chorusing, identifying four acoustic patterns of two competitive frog species with a precision of 76.7% [21].

The information gathered from algorithms could be a handful to experts, helping to answer ecological questions like determining the tendencies in the species' behavior and population. Then, it would be possible to make a quick alert of the critical reduction in their breed and to take more severe actions or propose more decisive politics to stop their extinction. Also, the models and algorithms designed can help other data scientists and engineers boost even more tools that help the data analysis in the bioacoustics field.

Chapter 4

Objectives

4.1 General Objective

Recognize anuran species from the neotropics in bioacoustic signals through data augmentation and multilabel classification.

4.2 Specific Objectives

- Construct the training, validation, and test sets for the learning models based in transfer learning.
- Test different techniques of data augmentation in transfer learning algorithms to detect and recognize anuran in bioacoustic signals.
- Evaluate the learning algorithms in the database Anuraset.

Chapter 5

Theoretical framework

5.1 Thematic Areas

- Computational and artificial intelligence - Artificial intelligence - Machine learning - Transfer learning
- Computational and artificial intelligence - Neural networks - Artificial neural networks - Convolutional neural networks
- Computers and information processing - Data systems-Data handling-Data augmentation
- Computers and information processing - Pattern recognition - Sound recognition
- Signal processing - Acoustic signal processing
- Geoscience and remote sensing - Environmental monitoring
- Engineering in medicine and biology - Biodiversity
- Signal processing - Spectrogram

5.2 Theoretical framework

5.2.1 Bioacoustics

During the middle of the 20th century, the examination of animal sounds commenced by devising instruments like the spectrograph, which were utilized to monitor aquatic species [22]. During these examinations, it was established that species calls differ from one another, and that these sounds can alter according to the environmental and behavioral conditions [23]. As a result, it piqued interest for some scientists and since then, bioacoustics have studied the process of sound production and its reception. The study of bioacoustics also centers on the exploration of how communication is influenced by the surrounding environment across different species.

As explained by Erbe and Dent [24], its primary motivations revolve around conservation, technological advancement, and acquiring a deeper understanding of human anatomy. The monitoring of the coexistence environment plays a critical role in evaluating its health and its correlation with changes in species behavior, as well as metrics like abundance and distribution. Moreover, the intricate properties of biological acoustic systems have the potential to catalyze the development of

biomimetic technology. The elucidation of the hearing mechanisms in select animal species has the potential to enhance our understanding of human auditory anatomy and potentially facilitate the development of therapeutic interventions for specific hearing impairments.

With the development of new technologies, it has been possible to generate specialized devices for all kinds of habitats, such as hydrophones for marine species and ultrasonic detectors for bat calls. These devices offer the possibility of storing a considerable quantity of audio material while programmed to record during certain hours of the day. By employing this method, it becomes feasible to observe species that are visually challenging to study, with the sole requirement being that they emit a distinct sound that can be detected. One of the most studied species is birds due to its characteristics. The best way to approach them is with acoustic equipment. In addition, birds are susceptible to climate change, so studying their behavior can give a better understanding of the magnitude of global warming in the ecosystems.

Given the information at hand, it becomes evident that bioacoustics encompasses three distinct approaches, which integrate diverse investigations and devices from disciplines such as biology, mechanics, engineering, and others. The scopes can be grouped into various categories, which will be explained in the following sections, based on the information presented by Erbe and Dent [24] and Penar et al. [22].

Bioacoustics research can be approached from a biological or anatomical perspective. This includes studying the anatomy of organs involved in producing sounds or signals. Similarly, the process of generating these signals can be examined. Additionally, the mechanism for receiving signals, along with the neurophysiological process and the organs related to it, can also be studied. The possibility of incorporating other disciplines arises from the suggestion of developing devices based on animal organ functionality. The next approach has a greater potential to encompass various disciplines. It comprises signal analysis. Within a signal is possible to obtain information about the species (richness, density, occurrences, etc.), behavior, the relation with other species or calculate indices that lets explore more complex attributes like the entropy in the environment, and its effects in animal behavior. The last focus has recently been an additional concern in the field, thus changes in acoustic dynamics can be a good indicator of climate change; a clear example of this is the increasing ocean acidification, which has caused changes in acoustic absorption and, as a result, inciting potential behavior changes in marine species i.e. the case of whales that can increase the duration of their sounds, their intensity and modulate their frequency [25] [26]. These signals have multiple uses. They can help understand extinct species' biological activity, recreate species sounds based on their morphology and behavior, and detect ecosystem changes. This knowledge improves our understanding of climate change's impact on species and enhances endangered species protection.

In the next section will be explored one of the most common approaches used in bioacoustics: passive acoustic monitoring. The technique has broadly been used by experts in this field, thus it offers the benefit of constantly monitoring an environment without significantly affecting it or the dynamics inside it. This discipline is intimately related to the study of animal signals and how information can be extracted from them. All of this information will be detailed in further detail below.

5.2.1.1 Passive Acoustic Monitoring

In bioacoustics, passive acoustic monitoring is highly utilized and effective. It provides a non-invasive and affordable method to gather data from wildlife. This has provided valuable insights to numerous scientists and ecologists regarding species' behavior, as well as the impact of global warming and human influence on nature. This approach will enable the assessment of the severity of the present circumstances, providing substantial evidence for environmental institutions to advocate for more stringent ecological measures. Browning et al. [5] demonstrated that the utilization of various sensor types enables the acquisition of information from extremely hostile habitats. These data are represented as waveforms. Fourier transform enables the generation of spectrograms from recorded audio, simplifying specific analysis procedures such as studying animal call patterns and their relationship with their behavior. To conclude, by consolidating gathered data and employing supplementary tools, one can obtain information pertaining to species occupancy, population, density, abundance, and, importantly, the inter-species interactions within a designated territory.

5.2.1.1.1 Phases involved in Passive Acoustic Monitoring

The complexity of bioacoustics computational analysis necessitates a series of steps, each of which will be detailed below, including the associated challenges. This discussion will be based on the researches of K. V. S. N. et al. [23] and Browning et al. [5].

5.2.1.1.1.1 Data Collection

As its name suggests, it consists in the process of recording the sound of the species of interest and storing it in a digital format. The recording device is the principal responsible for this procedure, thus its transducer translates the vibrations produced by a sound wave in the microphone into an electric signal. The type of transducer depends on the application. For example, if the sound of interest corresponds to a frequency below the audible range, then it would be needed a specialized infrasonic detector. The quality of the documented material is influenced by sample rate and bit-depth. These factors define the frequency and amplitude resolution of the recorded signal. Finally, some other characteristics that are important to consider in these devices are how sufficient data they can store and how much time it can record. Fortunately, actual digital recorders have several other benefits, including longer recording times because they use SD cards, which can have a considerable storage capacity. They also offer the possibility of directly downloading the recorded files to a computer.

The challenges of this process are related principally to the equipment needed to record. The high expense associated with state-of-the-art devices and sensor networks can restrict the scope of certain investigations, despite the growing trend towards developing affordable sensors. In addition, maintenance of the equipment could be difficult because of environments that are difficult to access. Proper maintenance of the devices is of utmost importance, particularly when operating in environments such as tropical forests. The climatic conditions in these areas pose a risk to the sensors and recorders, as the humidity might impact the functionality of the microphones. Additionally, in

areas with high biodiversity, certain species may be a threat to the equipment, underscoring the need for vigilance in upkeep. Equipment can still be stolen in crowded environments. Therefore, exists some extra consideration to protect the sensor from the conditions of the environment, animals within the area or even human influence. Finally, another difficulty found in this process is the lack of standardization of protocols for data collection, which is crucial to add comparability between survey programs.

5.2.1.1.1.2 Pre-processing

During this step, the data collected could go through certain signal processing techniques to get it ready for data analysis. To initiate this process, it is essential to transform the data into a specific data type that facilitates the visualization of key characteristics within the recorded audios. Raw audio data alone does not offer adequate information for computational analysis or the retrieval of data, such as species identification or environmental analysis. That is why a representation in the frequency or time domain is utilized.

The choice of audio features to use depends on the application. For example, some features used to highlight spectral are temporal features, such as spectrograms or MFCCs. Another data visualization is based on entropy to measure the diversity or complexity of noise in a signal. Other approaches are settled by the fragmentation of the signal, such as using syllables, phrases, or calls. Moreover, as outlined by K. V. S. N. et al. [23], alternative investigations have integrated Linear Predictive Coding, a codebook containing frame level features or the Marsyas framework. As was mentioned, these can subtract crucial information about the environment and the species living in it.

One of the signal processing techniques that can be employed is the utilization of an oscillogram to monitor the temporal fluctuations in audio. Nevertheless, the prevalent technique revolves around the utilization of a Fourier transform on the signal, enabling the generation of a spectrogram. This visualization displays the data in relation to frequency and time, with the vertical and horizontal axes denoting these dimensions, respectively. As it is depicted in Figure 5.1, there are several signals that can be detected by its amplitude (energy) shown as color density that goes from blue to yellow. As a result, spectrograms appear as an advantageous application that enables the visual representation of three signal properties: time, frequency, and amplitude. One obstacle faced in this data visualization is the trade-off between time and frequency resolution. When performing Fast Fourier Transform, larger windows length will produce better frequency resolution but reduced time resolution, and vice versa. Another viable option is to use an alternative signal processing method, such as time domain wave-form analysis or cestrum-based feature extraction, in order to extract all relevant information. The choice should be guided by the benefits offered by the method and the specific signal processing requirements.

After acquiring the data visualization, there are some additional steps that can be taken. One option is to use techniques like wiener filtering or Minimum Mean Square Error (MMSE) to reduce audio noise and ensure a clear signal. Another possibility is to apply Syllable Extraction and Feature Extraction to the recording. Syllable Extraction helps gather the most important segments of the

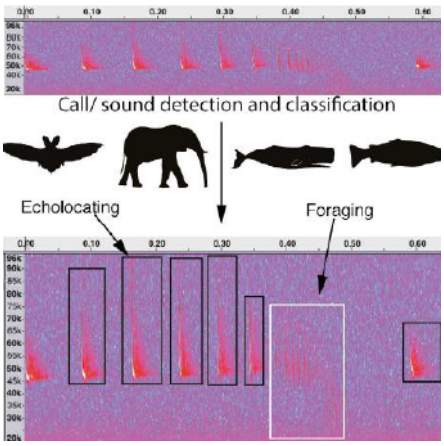

Figure 5.1: An example of a spectrogram that displays different frequencies of various species.Source:[5]

signal, while Feature Extraction captures the most significant information from these segments[23].

5.2.1.1.1.3 Data Analysis

This stage depends completely on the application. According to the explanation provided, bioacoustics analysis encompasses specific trends such as population estimation, species identification, studying animal behavior, and assessing biodiversity in the environment. In general, the operations of these procedures can be categorized into two primary groups: detection, which involves the identification of a particular signal within an audio, and classification, which entails assigning the signals to specific classes. This phase can be hand designed or learned from a data structure with the help of deep CNN (Convolutional Neural Networks). These algorithms and methods are used to classify and label the information gathered from the sensors [5]. Data analysis emerges as one of the primary obstacles in bioacoustics, as indicated by Browning et al. [5] and the forthcoming project led by Llusia et al. [19].

These operations can be made manual, semi-automated or automated. Given the quantity and complexity of the data, conducting a manual review would be both inefficient and infeasible, while also demanding a substantial amount of time. Therefore, it is imperative to establish efficient procedures for information processing, enabling the derivation of precise and crucial conclusions from environmental signals.

With automated and semi-automated tools, Machine Learning appears as an appealing implementation to treat the huge amount of data, offering automated predictions with reliable precision. The tool's adaptability across various fields makes it a versatile solution for addressing multiple problems and examining the resulting outcomes. As discussed by Tavasoli [27], Machine Learning has shown great potential in delivering precise predictions that can facilitate informed decision-making and enable timely, error-free actions. Several algorithms have been evaluated to prove its

performance. In bioacoustics, the commonly utilized architectures include artificial neural networks (ANN), Convolutional Neural Networks (CNN), Hidden Markov Models, and Support Vector Machines. An example of application of these algorithms in bioacoustics is iBatsID a platform created to monitor bat populations and which its learning model is based in Artificial Neural Networks.

As it will be explained in Section 5.2.3, the learning process of a model is divided into two stages: feature extraction and classification. Fully automated use of machine learning models is still in the process of investigation due to the highly rare precision rate, especially in multi-species classification, thus is typically assumed that one species is present in a sample. This assumption cannot be presumed in a recording containing multiple species, as their frequency bands may overlap and potentially interfere with each other. To achieve exceptional performance, a large training dataset is typically required. This dataset should include thousands of samples, including those with ambient noise and without the presence of the species of interest, to enhance the model's robustness. In the field, acquiring these can be a challenging task due to specific conditions, including environmental circumstances and acoustic surroundings. Moreover, variations in species population and detectability can further complicate the process. This relates to a major challenge in this field.

Another challenge related to the use of machine learning models is the complexity of using and training this algorithm. Usually, it requires an expert or a certain level of knowledge, but this is not possible in every case. As a result, it is required to develop more user-friendly platforms that can be utilized by disciplines other than computational sciences, hence boosting its utility for acoustic monitoring.

After training the model or performing manual identification, the method to assess certain characteristics of interest must be applied. In this step, the methods can encompass acoustic indices, making plots or using statistics metrics like Shanon Wiener. These methods can give a better insight into information like population density, the richness in the habitat, species behavior, etc. Finally, with these metrics in hand, it is possible to draw conclusions and even to take certain actions according to them. The use of acoustic indices has become a trend in analyzing the acoustic energy in recordings and determining the diversity of vocalizing animals. While the relationship is not yet fully established, studies are currently underway to determine the effectiveness of acoustic indices in measuring ecological characteristics. Additionally, researchers are investigating the potential use of these indices as features for machine learning models. Consequently, this presents yet another challenge within this discipline.

5.2.1.1.2 Task related to Passive Acoustic Monitoring

Following the explanation of the steps involved in passive acoustic monitoring, it is possible to delineate the typical applications of this field, which include species and population studies, animal behavior analysis, and assessment of acoustic communities and biodiversity. This section will outline the aforementioned categories using the information provided by Browning et al. [5] as the primary reference.

5.2.1.1.2.1 Studies of species and population

In the first category, as it suggests, sensors are used to monitor species and populations. These studies can compress monitoring endangered species or much complex species to detect like bats or insects. Even within this group, there have been studies conducted to monitor illegal activity through the identification of auditory cues, such as gunshots. This information serves multiple purposes, including highlighting the success of conservation measures, checking the impact of climate change on species, monitoring commercial species like fish, and expanding knowledge about specific species and their behavior.

Various techniques can be employed to determine the population density or any other demographic measure. Some of the most common methods are estimating population in small areas inside the habitat to calculate the approximate total of individuals from the species. An alternative method involves the determination of specimen identification probability through the consideration of the distance between the animal and the detector as a fundamental parameter. Another approach for counting individuals is to estimate their numbers via spatial replication. The most recent technique employed is variance estimation, which encompasses two approaches: analytic variance estimation and bootstrap variance estimation. The first estimation uses random and constant components, while the second estimation utilizes resampling of various devices to create a bootstrap dataset for calculating density. Statistical methods are primarily the main challenges faced by this group; thus, it must consider environmental conditions that can complicate the detection or that a call can belong to the same specimen.

An example of an investigation within this group is the monitoring of the endangered species Vaquita Marina conducted by the government of Mexico. Here, 44 Cetacean PODs were displayed to control the population of these cetaceans from 2011 to 2015. This was made to keep control of the noticeably decreasing population of this species. The final analysis revealed a mean annual decline of -34% Vaquita population, provoking that the Mexican Government declared a 2-year ban for gill-net activities

5.2.1.1.2.2 Animal Behavior

The second category studies animal spatiotemporal patterns to understand wildlife behavior and their relationship with the environment. This research can reveal details regarding the effects of human activity on animal conduct, especially in zones where industry or human habitation has participated in the habitat. It also investigates behavioral responses to other species inhabiting its zone or environmental noise. While acquiring and identifying patterns can be challenging due to interference from other species' calls or background noise, it is possible only if the individual has a unique acoustic signature. Examples of applications within this field of study include Munger et al. [28] use of hydrophones to monitor North Pacific right whale migration patterns, and Gil et al. [29] investigation on changes in bird behavior near airports.

5.2.1.1.2.3 Acoustic communities and biodiversity

The last group of studies is focused more on the richness of species and diversity in certain areas. This subfield has taken the name eco-acoustics or soundscape ecology. This approach allows for the measurement of environmental soundscape dynamics within an ecological acoustic community.
There are statistics that have been typically used to explore biodiversity in a zone such as the Shanon Wiener statistic. Another popular method is employing acoustic indices or checking spectrograms to quantify the biotic level present in a recording. However, as previously said, several types of acoustic indices are still being studied to discover if they may quantify specific ecological traits. An alternative strategy involves the utilization of soundscape, wherein microphones are positioned in a manner that they face each other, facilitating the acquisition of recordings of higher quality. In this approach, the signal is downsampled, noise is filtered out, and correlation is used to decrease false positives, ultimately achieving a clearer signal.
Investigations in this line of work included a study by Sankupellay et al. [30], who examined the use of spectral indices to create false-color spectrogram images in order to summarize 24-hour recordings. Another study by Depraetere et al. [31] utilized the Acoustic Richness index (AR) and the former dissimilarity index (D) to explore diversity in various forest habitats.

According to Penar et al. [22], amphibians are the most threatened vertebrate species presently. Data information and methods for monitoring this species are typically scarce. Furthermore, certain species are difficult to identify visually, therefore their detection is call-dependent. That is the reason passive acoustic monitoring appears as an appropriate approach to obtain information from amphibians. As mentioned earlier, the quantity of data obtained from a monitoring exercise can be substantial, encompassing thousands of samples, imposing a laborious and time-consuming burden on human annotation. This way automatic tools like machine learning have been a considerable alternative to identify species, and the information related to them. However, its application's effectiveness is still being investigated, as it has not yet attained the required level of precision and lacks the necessary data for achieving superior results. Further, as mentioned by K. V. S. N. et al. [23], multi-species classification has not been further explored. This project evaluates the efficacy of three machine learning models by employing transfer learning and data augmentation techniques. These methods aim to address the limited sample size and determine their effectiveness in classifying amphibian species in recordings. Successful results might pave the way for their application as automated tools for bioacoustics and passive acoustic monitoring.

5.2.2 Data Augmentation

This technique primarily stems from the need for Machine Learning algorithms of large amounts of data to produce accurate results, exhibiting effective learning from the data provided. This technique is widely used in image and text recognition but is also being applied in audio recognition. Data augmentation is a procedure that involves the application of techniques to existing signals, where certain characteristics are modified in subtle ways without altering the semantic composition. This results in creating synthetic data to be fed into the algorithm. As stated in Pluščec and

Šnajder [32] and Feng et al. [33], data augmentation can improve a model in a variety of ways. The principal benefit of data augmentation is the overall enhancement of model performance. It is commonly employed when data is scarce, since it can help to increase the number of samples while also introducing data diversity to improve robustness and producing tough data situations that can boost the model's precision to new unknown data. Depending on the dataset, it can help to lessen imbalance by including samples from the class with the fewest samples. Finally, it can be an effective regularization strategy, preventing overfitting in a model without changing its architecture. Data augmentation can serve as an indication of the model's ability to mitigate bias and, in certain scenarios, generate synthetic data for safeguarding individuals' privacy, particularly when dealing with sensitive information like biomedical records. The strategies employed vary depending on the type of data being researched. Various techniques for data augmentation have been devised for applications involving text or images. Image augmentation can involve the use of transformations like color, geometric, or others. As for text augmentation, methods such as paraphrasing and swapping can be employed. The techniques must be based on the problem too, thus adding techniques without considering the context of the application could harm the training process. Data augmentation should rarely modify the semantic content of the data and should instead simulate possible data values. Though it depends on the model employed, certain models can benefit from data augmentation to improve their feature extraction capacity. The present study focuses on several audio augmentation approaches that, according to Doshi [34] and Lucas Ferreira-Paiva et al. [9], can be classified based on whether the technique applies to the raw audio or the generated spectrogram. In the subsections that proceed, the strategies employed in both categories for this project will be discussed.

5.2.2.1 Audio signal techniques

The techniques that will be discussed in this section are directly applied to the audio signal.

5.2.2.1.1 Echo effect:

This approach is based on the article from Lehman [35], the audio signal is played back after a delay time, which can comprise from milliseconds to a few seconds. Finally, this operation is represented by the equation $y(n) = x(n) + x(n - T)$, where signal $y(n)$ acquires and echo effect.

5.2.2.1.2 DRC:

Dynamic Range Compression (DRC) is designed to compress the dynamic range of the signal, as indicated by its name. Its functionality is based on a compressor and expander, proposed by Zolzer [36]. This method employs an RMS level detector to monitor the signal level and select between the compressor and the expander. The compressor reduces the dynamic range in the signal meanwhile the expander increases the dynamics by scaling the small changes on the input signal to higher ones in the output. Equation 5.1 represents the operation of this function.

$$G = \begin{cases} CS * (CT - X) & X > CT \\ 0dB & ET \leq X \leq CT \\ ES * (ET - X) & X > ET \end{cases} \\ = min(0, CS * (CT - X), ES * (ET - X)) \tag{5.1}$$

5.2.2.1.3 Time Stretch:

Zolzer [36] defines time scaling as the manipulation of an audio signal's duration, wherein the signal can be expanded or contracted while preserving its pitch. There are multiple approaches to implementing this technique, and the one employed in Scaper, the library utilized in this study, is referred to as WSOLA. Described in Verhelst and Roelands [37] and Driedger and Müller [38], in this algorithm, the principal purpose is to produce a waveform that maintains the maximum local similarity. To achieve this objective, given a frame position in the output signal, the algorithm needs to find a shift within a range that works as a tolerance region, which will assure a natural continuation to next frame according to the waveform in the original signal. This process will be repeated till completing the desired length.

5.2.2.1.4 Signal Shuffling:

This technique, described in the investigation by Inoue et al. [39], is based on the assumption that is possible to create new samples by shuffling the order of a sound segment. Therefore, in this implementation, the audio input is divided into a random number of sections, in which order will be mixed, as in can be seen in Figure 5.2.

Figure 5.2: shuffling operation

5.2.2.1.4.1 Shuffling and Mixing:

This technique is similar to the one described above, except it requires two audios as inputs. The procedure involves the creation of a new audio composition through the fusion and rearrangement of segments from both recordings. This method makes the assumption that when two audios from the same class are mixed, the resulting audio will have the same label. The category remains the same, even with shuffled order. The intended result will be achieved by randomly dividing both input audios into multiple segments. Additionally, a corresponding array will identify the audio source ($audio0$ or $audio1$) for each segment in the output. This operation is shown in Figure 5.3, with both arrays: order and audio origin.

Figure 5.3: shuffling and mixing operation

5.2.2.1.5 Clipping:

Clipping is the phenomenon that occurs when the amplitude of a signal surpasses a specific threshold, resulting in the signal being constrained and flattened. In this work, a Hard Clipping method was used to completely restrict and flatten the signal. The process is represented by a nonlinear clipping function in Equation 5.2, where U_{Limit} and L_{Limit} define the upper and lower limits of the function. The audio sample is denoted as x, and $f(x)$ represents the sample after the clipping operation [40].

$$f(x) = \begin{cases} U_{Limit} & x > U_{Limit} \\ x & L_{Limit} < x < U_{Limit} \\ L_{Limit} & x < L_{Limit} \end{cases} \tag{5.2}$$

The implemented function in this work is built upon the clipping function from the Audiomentations library by Jordan [41]. The function employs a random uniform distribution to select a specific percentage of the signal and subsequently determines the upper and lower limits using Numpy's percentile function.

5.2.2.1.6 Wow resampling:

Short-term variations in speed lead to a degradation known as the Wow effect. The procedure bears resemblance to pitch shift augmentation, with the distinction being the varying intensity over time. Therefore, is perceived as a pitch fluctuation in the audio. Equation 5.3, proposed by Nanni et al. [7], represents the operation which is repeated several times and is applied to the signal where the wow effect is represented by a fixed sine wave function.

$$F(x) = x + a_m \frac{sin(2\pi f_m x)}{2\pi f_m} \tag{5.3}$$

In Equation 5.3, x represents the position of the elements in the original audio array, thus this operation will define the mapping from the old audio to generated one. In addition, a_m and f_m are factors given as inputs and represent the intensity of change and the frequency of change in the operation. After calculating the signal $F(x)$, the original signal is up sampled and then sampled using the nearest neighbor algorithm.

5.2.2.1.7 Rolling:

By employing a random uniform distribution, this technique determines a percentage that is subsequently multiplied by the size of the signal elements. This calculation yields the number of indexes,

dictating the number of spaces by which the audio signal will be rolled. The order in which the signal will be rolled, either upward or downward, is determined randomly. The principal difference with a delay augmentation is that the last element of the signal becomes the first pushing the rest of elements.

5.2.2.1.8 VTLP:

The investigation conducted by Sarkar and Tan [42] introduced VTLP, which stands for Vocal Tract Length Perturbation. This concept is derived from Vocal Tract Length Normalization, a technique used in speech recognition to reduce the effect of speaker variability generated by the difference of Vocal Tract Length between speakers. Here, frequency warping is used to normalize the effect of spectral variations. However, in this case and as its name suggests, frequency warping is used to add perturbations in the spectra by scaling the frequency axis of the signal. The warped frequency of the audio is expressed as a piecewise linear warping, represented by Equation 5.4. The VTLP factor α is decided by a random normal distribution within a range from 0.9 to 1.1.

$$f' = \begin{cases} f * \alpha & f \leq f_0 * \frac{min(\alpha,1)}{\alpha} \\ \frac{sr}{2} - \frac{\frac{sr}{2} - f_0 * min(\alpha,1)}{\frac{sr}{2} - f_0 * \frac{min(\alpha,1)}{\alpha}} * (\frac{sr}{2} - f) \end{cases} \tag{5.4}$$

5.2.2.1.9 Noise Injection:

According to the findings from Nanni et al. [7] and Sun et al. [8], the type of noise that can be added to a sample can be divided in two categories: Additive White Gaussian noise and the mixture of two audios, both will serve as background noise. This operation is represented in Equation 5.5, where $x(t)$ represents the audio sample and $z(t)$ represents one of the two types of noise.

$$y(t) = x(t) + z(t) \tag{5.5}$$

5.2.2.1.9.1 Additive White Gaussian Noise:

As defined by Chaudhari [43], this noise has three characteristics present in its name. Its additive characteristic is represented in the Equation 5.5, thus the output signal is equal to the original signal plus the noise. The white characteristic refers to the concept that it has uniform power in all frequencies. The name Gaussian originates from the probabilistic distribution that characterizes the signal, indicating that the distribution's mean is zero. Positive and negative values can both be possessed by the signal, with values closer to zero being more probable. Therefore, in this function, a noise signal is created with a random normal distribution with zero mean and then is added up to the input audio signal.

5.2.2.1.9.2 Audio Mix:

As it is expressed in Equation 5.5, in this strategy, the noise signal $z(t)$ is added up to the audio signal $x(t)$. In this context, the signal $z(t)$ is composed of a signal derived from a distinct form of sound,

such as the sound of rain or wind, which can be classified as ambient noise. The Scaper library was used for this purpose [44]. This library is composed of more complex and complete transformations; thus, several augmentations can be applied to the audio through the same command, such as Pitch Shift and Time Stretch, explained in previous sections. With this toolkit, it is possible to choose several audios from different folders assigned by the user. In addition, it is possible to choose the start time of the background sound in the source file and one of its fundamental characteristics is that the library can automatically accommodate the timing of the audios even in both have different duration and adapt the source time to the duration that was given as input to the function.

5.2.2.1.10 Delay:

This approach is a time shifting method that relies on the operations outlined by Chaparro [45], wherein the signal $x(n)$ is either delayed by T seconds to yield the signal $x(n-T)$, or advanced by this factor to obtain the signal $x(n+T)$. Achieved by padding the input array and randomly shifting the audio to left or right. In addition, the padding-displaced zone is filled with white noise to avoid similitudes with masking augmentations.

5.2.2.1.11 Pitch Shift:

The concept of pitch shift, as described by Zolzer [36], involves transposing the shift of the original signal to either decrease or raise it. Time stretching and resampling are necessary to achieve this result. This way, it is assumed that the spectrum is compressed or expanded over the frequency axis by the resampling operation but preserving its harmonic relation. Thus, it is not altered but scaled. The amplitude of the harmonics is then kept constant, and the time-stretching procedure rescales the pitch-shifted signal to its original length. The time stretching function is generated by the WSOLA algorithm, which is detailed in Section 5.2.2.1.3.

5.2.2.1.12 Harmonic Distortion:

a quadratic distortion is applied, which signifies that the function contains a quadratic component that contributes to the output's nonlinearity [46]. As indicated by Equation 5.6, the method involves the consecutive application of a sine function, with x denoting the number of iterations to generate an audio saturation effect.

$$S_{aug} = sin^{x}(2\pi S_{audio}) \tag{5.6}$$

It is possible to include an optional argument in this function, which determines the number of times the Harmonic Distortion should be applied, the level of distortion increases proportionally with the number of repetitions.

5.2.2.1.13 Flip:

This method implements a function that reverses the order of the elements in the audio array. For this project, the time axis is flipped with audio signals.

5.2.2.2 Spectrogram augmentation techniques:

The techniques outlined in this section are directly implemented on the spectrogram.

5.2.2.2.1 Time Masking:

This method was tested by Sun et al. [8], Nanni et al. [7] and Park et al. [47]. Its procedure consists in mask a range of frequency bands $[t, t_0+t)$ by setting the range of values in this interval to cero, t_0 represents the initial time index value, where the masking will start and t define the percentage of elements in the array that will be masked. Both values are chosen by a random uniform distribution. In this investigation, the function randomly decides the percentage of elements in the spectrogram and the number of parts in which that percentage will be divided. Therefore, several frequency bands of the spectrogram are masked with diverse sizes.

5.2.2.2.2 Time Swapping:

This technique consists in select two random ranges from the spectrogram array to alternate its positions Song et al. [48]. The function selects a percentage from a random uniform distribution. This value is used to obtain the number of elements that will constitute the range represented by the variable t. Additionally, there are two different variables defined by a random distribution: t_0 and t_1, these variables are the initial indexes for the two ranges that will be exchanged. Subsequently, the function generates two distinct ranges: $[t_0, t_0 + t)$ and $[t_1, t_1 + t)$.

5.2.2.2.3 EMDA:

Equalized Mixture Data Augmentation [49] benefits from the property of mixing two separate audios from the same class, producing an audio that corresponds to the same label. This method involves the fusion of two distinct audio tracks from the same label, employing randomly selected timings. Furthermore, a different type of perturbation occurs when both audios are altered using an equalizing function that includes a peak equalizing filter with five parameters: the center frequency f_0, the gain g, the quality factor Q, and the sampling frequency. Except for the sampling frequency, each of these factors is determined at random by a uniform distribution. Pre-defined ranges determine the value of each parameter after following that. The center frequency is determined by the signal and the frequencies that comprise it. The range is defined by the minimum and maximum frequencies gained from the array. Meanwhile, the gain and the Q factor diverge from $[-8, 8]$ and $[1, 9]$, respectively. Then, Equation 5.7 defines the process to obtain the augmented audio.

$$S_{aug} = \alpha\Phi(s_1(t), \psi_1) + (1 - \alpha)\Phi(s_2(t - \beta T), \psi_2) \tag{5.7}$$

The values of α and β are chosen at random from a uniform distribution, and Φ denotes the parametric equalizing filter, whose parameters are specified by ψ. A random delay T is also applied to the signal s_2.

5.2.2.2.4 Techniques founded on image augmentation

Spectrogram augmentation strategies related to color and other image transformation were reported in Zhou et al. [50] and Lucas Ferreira-Paiva et al. [9] survey of methodologies. Some of these strategies will be discussed in these sections. The RGB, Negative-Positive Inversion, and color reduction functions were developed by [51].

5.2.2.2.4.1 Brightness and saturation:

The operation in this approach is defined by Equation 5.8, and I_{blend} represents a pure black image or a gray scaling image depending on whether the brightness or saturation is to be changed. Both images are multiplied by an α factor drawn from a random uniform distribution spanning a range of 0.1 to 0.9 for brightness and a random choice function covering the same range for saturation.

$$I_{aug} = (1 - \alpha)I_{blend} + \alpha I_{original} \tag{5.8}$$

5.2.2.2.4.2 Negative-Positive Filter:

In this transformation, the darker the sections, the brighter they appear, while the brightest area appears as the darkest. This was accomplished with Numpy by subtracting the pixel value from the maximum value (255).

5.2.2.2.4.3 Color Reduction:

The number of colors that appear in the image is minimized with this technique. This effect is achieved by discretizing the pixel values, applying a floor division to the array, then multiplying the result by the number of colors in the palette.

5.2.2.2.4.4 Color Filtering:

By setting the other channels in the array to zero, this color transformation produces single-color images. As a result, the function generates three images: a red image, a green image, and a blue image. As was observed during the experiments, the blue channel image is overly dark. Subsequently, the qualities of the spectrogram are undetectable. Therefore, it is not recommended to use it for augmentation since it might disorient the CNN.

5.2.2.3 Online vs. offline augmentation

Augmentation can also be classified depending on the moment where applications techniques are applied. The differences between these categories will be explained in the next sections.

5.2.2.3.1 Online Augmentation:

Some sources have defined online augmentation with its benefits TAO [52] and test its performance with image datasets such as RIWA or Cityscapes [53]. Augmentation is considered online when the transformations are applied while training the model with the data. This form of augmentation offers the advantage of not consuming computer storage, preventing the need to store augmented samples in memory. This also means that the model can be continuously trained in new images, because each technique produces a new image. Although this procedure will require more processing power and more epochs to converge, this approach can iterate faster than using offline augmentation. There are multiple ways to tackle the first issue, one of which involves utilizing threads for data loading. How the techniques are applied depends on the investigation and how it is proposed. With Wagner et al. [53], the augmentation of each sample is determined by a probability set by the operator. The user can specify the desired number of techniques, and each technique has its own probability of being applied. Finally, the resulting image is ready for training. As shown by Wagner et al. [53], this approach is recommended for already extensive datasets, so it will not take much storage and it will have a faster training time than offline augmentation.

5.2.2.3.2 Offline augmentation:

TAO [52] also gave a definition of offline augmentation, while Wagner et al. [53] made a comparison of its performance with online augmentation and made recommendations of which approach to use depending on the conditions. In contrast to online augmentation, the augmentation techniques are implemented prior to model training in this instance. The data generated by augmentation is stored in memory. This denotes the final quantity of samples before training the model and usually the model converges in fewer epochs than with online augmentation. It offers the benefit of controlling the quality of the augmentation in the dataset and check if the augmentations are appropriately implemented. Though, depending on the quantity of samples, this approach can require considerable storage space. Thus, multiplying each sample by a certain quantity of augmentations can cause a large and heavy dataset, and this cannot be handled by every memory. Ultimately, performing augmentation in advance provides a more accurate method for mitigating imbalance compared to online augmentation. This is accomplished by calculating the necessary number of augmentations for a class with fewer occurrences in order to correspond closely with the class containing the highest number of samples. Similarly to online augmentation, the process and the techniques are applied depending on the purpose of the investigation. Wagner et al. [53] suggests that this approach is suitable for datasets with limited samples, as it effectively increases both the quantity and diversity of samples while minimizing storage requirements.

5.2.3 Machine Learning

This concept has become popular in recent years. Currently, know about it and see a large quantity of projects and publications related to the topic it is common. Also, people interact with its algorithms in their daily life from spam identification to image recognition. Machine learning is

present in several aspects of technology. Then, this concept has different formal definitions but principally it can be said that is the field of study that programs computers to give them the ability to learn from data with no need of a specific program to make a task as exposed by Géron [54]. This implies that the computer will acquire the ability to perform the necessary task based on the given information, rather than developing a dedicated program for that particular task. For example, with the labels and data provided to an algorithm, it will be possible for it to learn to classify spam emails, without existing the necessity of create a long and heavy program that makes the same task and that possibly needs constant human supervision.

First, there are some concepts that are important to understand how Machine Learning works extracted from Howard and Gugger [55]:

- **Weights** are the variables that are going to be implemented. And a weight assignment is a particular choice of values for those variables. These values will determine how the algorithm will operate. Also, its use will be clarified in the next section, where Convolutional Neural Networks are explained.
- The **Model** is an especial type of program. It will be the entity designed to acquire knowledge and perform various tasks for which it was intended. Its performance will be defined by the weights.
- Actual **Performance** refers to how well the algorithm can perform.
- The mechanism of the algorithm will improve the model performance by changing the weights.

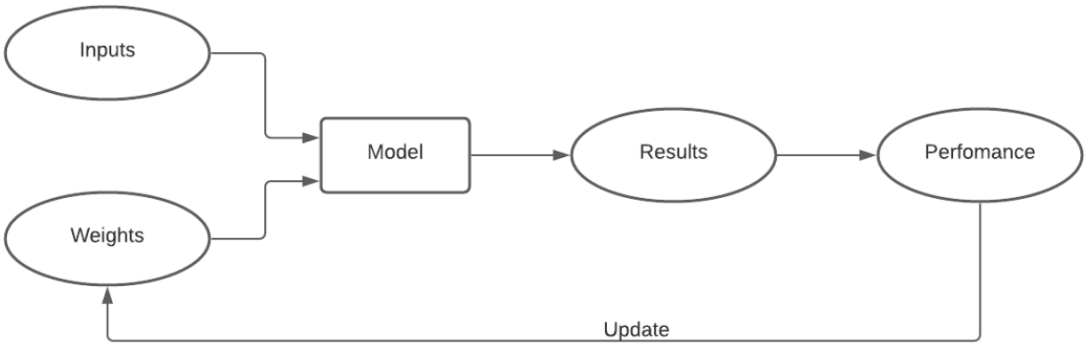

Figure 5.4: training a machine learning mode. Source: [55]

In Figure 5.4 is illustrated the process and elements that compose the training of a Machine Learning Model. First, the model receives some inputs and weights. Depending on these, it will make decisions and produce results. These results show the actual performance of the algorithm, which will help it learn thanks to the automatic mechanism that adjusts parameters based on the evaluated performance.

As stated in Goodfellow lan [56], Aggarwal [57], and Géron [54], after training a model, two major phenomena that could be observed are underfitting and overfitting. Both faults are related to the algorithm's ability to generalize, which represents how it performs with previously encountered data. Overfitting happens when a model learns subtle patterns in data, including noise patterns, which provokes that the algorithm has an excellent performance in training data but will perform poorly in unseen data. This mistake manifests as a significant discrepancy between the training and test errors, and its impact can be mitigated by employing regularization approaches such as those described in Section 5.2.3.1.1.4. Conversely, there is an additional issue referred to as underfitting, which arises when the model demonstrates inadequate performance on both the training and test datasets, showing that the model is too simplistic for learning structured data. This inaccuracy can be remedied by using a more complex model or boosting the model's parameters relative to other measurements.

For this works objectives and from the information that has been exposed and what was learned from Mac [58], for machine learning to work, it is necessary to provide data as an input and certain label to train a model, and then receive predictions from it. For biological signals, the Essential Biodiversity Variables can be an important tool for the training of the algorithm, because they can contain important parameters as the phenology, morphology, reproduction, or movement of certain species, giving crucial information that the machine learning algorithm can learn as shown in the research made by Kissling et al. [59].

As evidenced by Goodfellow lan [56], machine learning may be used in several tasks ranging from regression, in which the model should return a numerical input as a prediction based on the input values provided. It can do language-related tasks, such as transcription, machine translation, and the recognition of structured outputs. Additional specific tasks for machine learning can include anomaly detection and denoising, which essentially means extracting a clean signal from a corrupted sample. In this study, the primary emphasis is placed on the utilization of machine learning for classification, which is widely acknowledged as one of the most frequent applications of this field. Classification can be further categorized into four distinct groups [54].

- **Binary Classification:** in binary classification task, the data is categorized into one of two classes [57].
- **Multiclass Classification:** the purpose of multiclass classifiers is to differentiate between multiple classes.
- **Multi-label classification:** in this category, data is not assigned solely to one exclusive class. Some tasks require the model to return different classes for each input.
- **Multi-output Classification:**It is also known as multi-output multiclass classification. It can be thought of as a generalization of multi-label classification in which each label has various attributes, which it considered as it could be multiclass.

This research project will be a multi-label classification task; thus, each audio spectrum may contain multiple species. The model and metrics will then be built based on the type of categorization challenge, which will be detailed further. In the following sub-sections will be explained

some important concepts and their relevant theory for Machine Learning and the development of this project.

5.2.3.1 Convolutional Neural Network

This is one of the key concepts of machine learning, because it gives the possibility of solving any kind of problem by just varying its weights, as was explained by Howard and Gugger [55]. Just as its name implies, it is made up of networks composed of neurons. The design of this system was inspired by biological neurons. It has the structure to learn and make decisions. In Machine Learning, neurons carry the computational operation. In regular cases, the neurons form many weighted inputs, then a nonlinear transfer function does the sum and transmits the resulting output activation to the next level that is probably composed of a larger number of neurons. Its functionality wants to simulate the same of the biological neuron and the nonlinearity is implemented to limit the outputs to a certain range. This can be achieved with the use of different methods like the hyperbolic tangent, the exponential or sigmoid function, according to the theory planted by Jukes [60]. Further information about the components of a convolutional neural network can be found in Section 5.2.3.1.1.

From the information provided by Kahl [16], it is important to understand that convolutional networks are connected to a certain number of preceding neurons to configure the output instead of being connected to all units in the following layer. These types of networks learn global patterns while convolutional layers acquire local patterns, which can be recognized in any part of the input data and supply significantly improved generalization capabilities. Then, a convolutional neural network can use its structural composition and simplify tasks when the learned patterns dispense better representations by taking advantage of local value correlation.

The convolutional neural network architecture can be divided into two stages according to its function: feature extraction and classification. In the first stage, and, as its name suggests, it is the process of extract certain crucial characteristics from the data given to the model; in the second stage, the model compares the data with the closest match from a library previously learned and the gives the percentage of which class is most probable to belong.

The components of a convolutional neural network, including inputs, outputs, layers, and an example of its structure, are illustrated in Figure 5.5. The relationship between these layers is established through weights, which dictate the activation of the succeeding layer. Ultimately, this leads to generating a value that the final function translates into a prediction, presented to the user. Due to its design, this system offers flexibility in addressing a wide range of problems by adjusting the appropriate weights. And, as it was explained, convolutional networks are limited to a certain number of neurons in the next layer.

5.2.3.1.1 Architecture

Convolutional Neural Networks are primarily made up of a stack of convolutional layers, followed by a ReLU layer, and then a pooling layer, which can be followed by more convolutional layers with additional polling levels and ReLU layers. The last set of layers is linked to a feed-forward network,

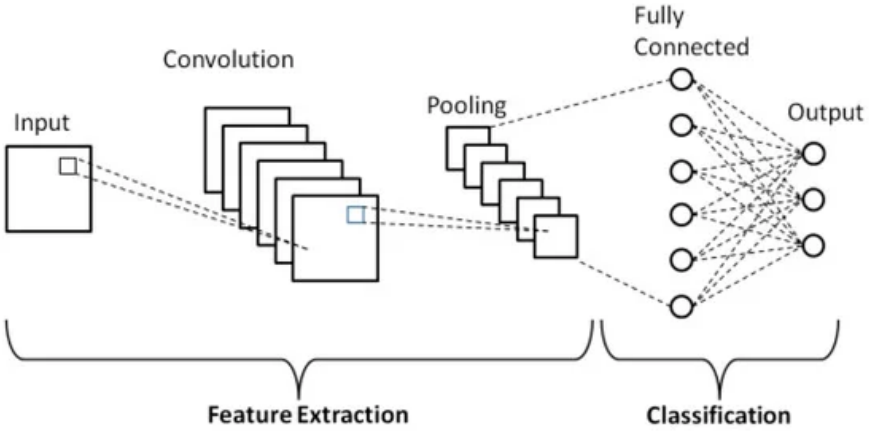

Figure 5.5: Convolutional Neural Network Architecture. Source: [61]

comprising several fully connected layers connected to a Rectified Linear Unit and a final layer that generates the prediction. Typically, this prediction is an output unit, such as a Softmax layer. Although a CNN typically comprises various layers beyond these, one can employ supplementary layers or leverage techniques like regularization and batch normalization to address overfitting and enhance model performance. Some of these concepts will be explained next.

5.2.3.1.1.1 Convolutional layers:

As defined by Aggarwal [57] and Géron [54], it is the most important block in a CNN, thus from this operation comes the convolutional term in its name. Each convolutional layer has a grid structure of 3 dimensions: height, width, and depth. The depth in the layers refers to the number of channels in the layer, different to the depth in the model, which alludes to the number of layers in a neural network. The importance of this dimension in neural networks is attributed to its purpose of capturing the independent properties along the channels of the data. These properties correspond to the different attributes extracted from local regions in the image.

Except for the first layer, the input of the actual layer is only connected to neurons in the previous layer through a grid. The convolution operation involves positioning the grid in all possible positions within the images in order to achieve complete overlap. It then performs a dot product between the parameters in the filter and the matching grid in the input, which is repeated for all valid positions in the inputs and filters. Although many neural network libraries feature cross-correlation, which is comparable to convolution, but it does not flip the kernel.

The form of layers can differ, making stride and padding crucial concepts to consider. In the case of a smaller input, padding is employed by surrounding it with zeros in order to achieve alignment with the layer. Although it is possible to connect larger inputs to smaller layers by spreading out the receptive fields. The transition from filters, commonly known as kernels, are essential elements within convolutional layers. It comprises three-dimensional units where the parameters are organized and are usually square-shaped. The number of filters has a direct effect on the model's capacity. Its

functionality employs weights to disregard everything in the receptive field except where the filters are aligned, resulting in any other value being multiplied by zero except where the filter is active. Then, a layer of neurons that employs the same filter generates a feature map that emphasizes the portions of the input image that trigger the filter the most. The main objective of the model is to acquire the necessary filter values in their appropriate locations for the designated task [56], while the layers will effectively learn to combine these filters into more complex patterns.

The structure of these layers and the architecture of the network adds a tendency where, in the earlier layers, the model focus on small and more primitive shapes or features whereas, in deeper layers, recognizes higher-lever features and more complex compositions in these shapes.

5.2.3.1.1.2 Pooling layers:

According to the books Géron [54], Goodfellow lan [56], and Aggarwal [57], there are two ways to depict a Convolutional Network: as a network composed of three distinct layers or as a complete layer consisting of three separate stages. The initial phase involves implementing the convolution operation as outlined in Section 5.2.3.1.1.1 and connecting it to a nonlinear activation function (ReLU layer) detailed in Section 5.2.3.3.6.6. The last and third stage involves the utilization of the pooling function, which will introduce further modifications to the outputs. Small grid patches linked to a finite number of neurons in the preceding layers are operated on by pooling layers. It is necessary to define the side, stride, and padding type. Although the depth is the same as in previous layers, the pooling also generates a feature map. Its common goal is to sub-sample and lower the size of the layer's input, resulting in a reduction in computational burden, memory requirements, and parameter number. However, its most significant contribution is the incorporation of translation invariance into the model, guaranteeing constant output values from the output layer despite any translation of the input. This allows similar images to be classified independently of the relative placements of their different forms, so the layer focuses on the presence of a feature rather than its location. Additionally, the integration of pooling, particularly max pooling, may contribute to achieving rotational and slight scale invariance. When employed in separately parametrised convolution, the model may learn which transformations to become invariant to.

Certain drawbacks are possessed by pooling operations, including the reduction in input size, leading to the loss of values, which can negatively impact model performance. Furthermore, invariance may not be desirable in other applications, such as semantic segmentation, where changes in the input should cause equivalent changes in the output [54].

Pooling functions include the average rectangular neighborhood, L^2 norm, and minimal value [56]. Although the most commonly used function is max pooling, which sends just the maximum input values from each grid zone to the next layer while dropping the rest.

5.2.3.1.1.3 Fully connected layer:

Also known as dense layers, its name derives from the fact that all of its neurons are connected to every neuron in the preceding layer [54]. This connection implies numerous parameters that must be considered when designing a neural network if resource constraints exist. In fully connected

layers, the input vector x is multiplied by a weight matrix w and a bias vector b before being sent to a nonlinear function f to produce the output vector y [62], as represented by Equation 5.9.

$$y = f(w * x + b) \tag{5.9}$$

The inclusion of fully connected layers depends on the application. However, using more than one in a neural network can boost processing capability. Despite this, it is typical to employ this layer as the output layer in most convolutional neural networks. In that context, it is connected to each network in the penultimate layer and assigned a weight. An activation function like Softmax or sigmoid may be employed, depending on the task [57], as outlined in Section 5.2.3.3.6.1. The main purpose of the final layer function is to yield the class scores, implying that it handles the final categorization.

5.2.3.1.1.4 Regularization techniques:

The optimization of an algorithm is sometimes not achievable solely through adjustments to the majority of hyperparameters. Additionally, modifying certain functions or layers can further compromise the model's performance. In addition, neural networks can be constituted by tens of thousands, even millions, of parameters; this constitution gives the model the capacity to predict model data but also could increment its tendency to overfitting [54]. That is why regularization is necessary.

Regularization can be defined as the techniques and modifications applied to a model intending to reduce the complexity of it and its generalization error but without significantly changing its training performance [56]. In most cases, regularization can be represented as it is depicted in Equation 5.10 and Equation 5.11, where the regularization factor is an extra factor added to the loss function, which also has a value α that controls the strength of the regularization. With this function implemented, it is possible to limit the value of the weights in the model and reduce the signals of overfitting while increasing the model capacity of generalization.

5.2.3.1.1.5 Dropout:

It is a popular regularization technique for deep neural networks. As stated in Géron [54], it can help increase the performance and accuracy of a neural network. The dropout strategy assigns a probability p to deactivate each neuron in the network temporarily at every training step during implementation. Stated differently, this neuron will be excluded during one training phase but might become active in the next. After training, neurons are not dropped again. The hyperparameter, the probability, p is known as dropout rate and is commonly set at 50%. Neurons, when exposed to dropout during training, lose the ability to co-adapt with nearby neurons, requiring them to be maximally useful independently. This forces them to focus on each of their input neurons and reduce their sensitivity to insignificant input variations. It is as if each step generates a new neural network, and the final neural network after training is an average of the prior ones. This strengthens the neural network and improves its ability to generalize.

5.2.3.1.1.6 L1 and L2 regularization:

Based on the information provided by Goodfellow lan [56] and Géron [54]. These strategies are used to minimize and mitigate the impacts of overfitting in models. As a result, they improve the model's capacity for generalization by restricting the neural network's connection weights while leaving biases unregularized; thus, biases require fewer data than weights to fit accurately, and if they are strongly regularized, underfitting occurs. Using these strategies reduces model complexity by causing various weights to resemble zero. In these two regularization techniques, the regularization parameter (α) determines the strength of regularization. If α is really large, all weights will be near to zero and the graph will have a flat line shape. The regularization term is added to the loss function $L(w)$, which is often the MSE (Mean Squared Error), resulting in a new regularized cost function $\tilde{L}(w)$.

Equation 5.10 represents the loss function while using ℓ_1 regularization, which is commonly known as or closely related to Lasso regularization. In this situation, the regularization term is the weight vector absolute values (ℓ_1 norm). This approach tends to eliminate the least significant features completely by assigning them a value of zero, introducing sparsity into the model and making it valuable for feature selection in certain models.

$$\tilde{L}(w) = L(w) + \alpha \sum_{i=1}^{n} |\theta_i| \tag{5.10}$$

Some literature refers to ℓ_2 regularization as ridge regression or Tikhonov regularization. Equation 5.11 represents the operation, which adds a regularization term equal to half the square of the weight vector ℓ_1 norm. The ℓ_2 norm is defined as the root of the weight vector's sum of squares. The weights in ℓ_2 regularization are reduced to a level near to zero, preventing the sparsity present in ℓ_1 regularization.

$$\tilde{L}(w) = L(w) + \frac{\alpha}{2} \sum_{i=1}^{n} \theta_i^2 \tag{5.11}$$

5.2.3.1.1.7 Batch normalization:

Based on what Géron [54] explains, this technique was created for handling the vanishing gradient problem. Empirical studies have confirmed that it decreases the model's sensitivity to weight initialization, accelerates the learning process even with high learning rates, and can function as a regularizer, reducing the reliance on other regularization methods.

Batch normalization entails adding an operation before or after the activation function of a hidden layer to zero-center and normalize the input. The output is then scaled and shifted using two parameter vectors per layer: one for scaling and one for shifting. Subsequently, the model acquires an understanding of the optimal scale and the mean for the input of every layer. Using this technique in the first layer eliminates the need to standardize the training set. To achieve normalization, the method examines and calculates the mean and standard deviation of each input over the mini batch used in the current step, which is accomplished by utilizing a moving average of

the input layer mean and standard deviation during training. In order to achieve precise calculation of these two measures, four parameter vectors are acquired in each layer when employing this approach: the output scale vector, the output offset vector, the final input mean vector, and the final standard deviation vector. The mean and deviation are thus estimated during training but only used afterwards.

This strategy has some negative aspects, such as heightened model complexity and diminished prediction speed resulting from the increased computational load of batch normalization.

5.2.3.1.2 Convolutional Neural networks architectures employed in this investigations

The following section will provide a description of the convolutional neural network architectures employed in this study.

5.2.3.1.2.1 Resnet:

proposed by He et al. [63], this model is built based on the concept of Residual Learning, where the principal purpose is to obtain the residual function instead of making the model fit a desired underlying mapping. This operation was thought to be more efficient and straightforward than optimizing the original mapping. Furthermore, identity mapping made it easier to push the residual to zero than fitting an identity mapping using a stack of nonlinear layers. This means that operations performed with the ReLU function will not cause the input image to lose any information. Resnet architecture was created to address the diminishing accuracy of deeper network models. As the network converges, the accuracy saturates and subsequently deteriorates. The architecture of Resnet networks is based principally in a plain network where some shortcut connections, that perform the identity mapping, are inserted after a certain number of layers and their outputs are added to the outputs of the stacked layers without adding extra parameters or computational complexity. This operation turns the network into a residual version that is shown in Figure 5.6. The composition of the architecture is similar to VGG nets with mostly convolutional layers that have $3x3$ filters and depending on the size of the output is decided the number of filters, and with the network ending in a global average pooling layer and a 1000-way fully connected layer with Softmax. The number after Resnet represents the number of layers that compose the model and usually are composed by 3-layer blocks with the shortcuts described before.

5.2.3.1.2.2 VGG:

This model, created by Simonyan and Zisserman [65], emerged as an experiment of increasing the depth of networks while achieving high results. One of its principal characteristics is the use of $3x3$ kernel-sized filters one after another instead of large kernel sized- filters such as $7x7$, reducing the number of parameters of the network and making the decision function more discriminative. The pre-defined input comprised an image of size $224x224$. The image is then processed through a stack of convolutional layers, each with a small $3x3$ filter that can capture up/down, center, and

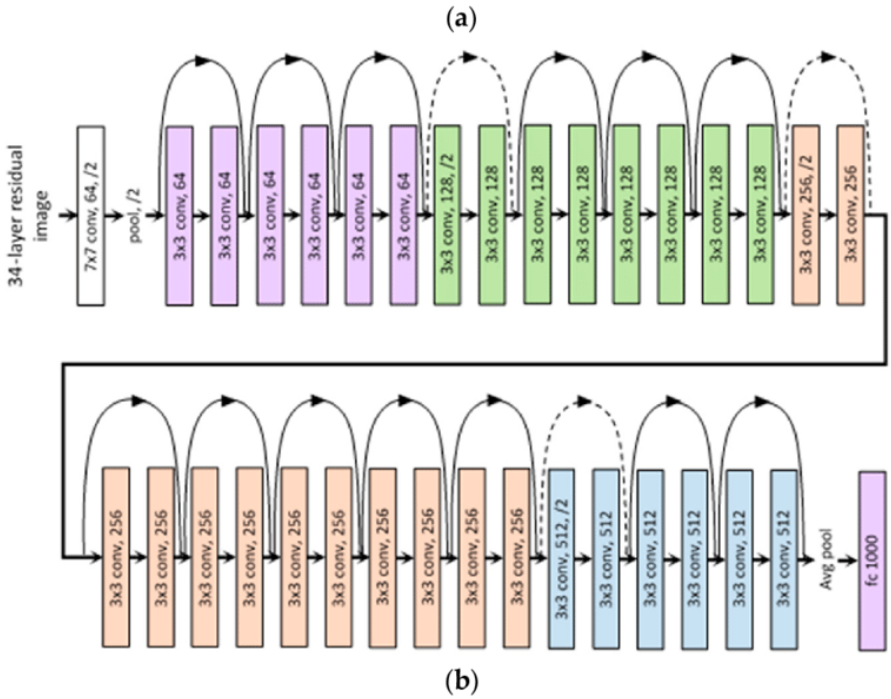

Figure 5.6: Architecture of ResNet-34. Source: [64]

left/right notions. Linear transformations are executed on the input channel by the network's $1x1$ convolution filters, and spatial pooling is performed after some of the convolution layers by five max-pooling layers.

Throughout the network, several Rectified Linear Units (ReLU) are added to introduce nonlinearity and aid in the modeling of complex relationships within the data. At the end of the network, after the stack of convolutional layers, three Fully Connected (FC) layers are attached: the first two have 4096 channels each, and the third performs classification with 1000 channels, which corresponds to the number of classes that needed to be classified. The complete architecture is illustrated in Figure 5.7. Its most renown configurations are the networks with 16 (VGG16) and 19 (VGG19) layers, thus have showed high accuracy with datasets like ImageNet.

5.2.3.1.2.3 EfficientNet:

is built upon research conducted by Tan and Le [67], which highlights the ineffectiveness of solely scaling one factor to enhance accuracy in a model. Instead, this approach introduces unnecessary complexity and strains computer resources. It surges from the intuition that "if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained pattern on the bigger image" [67]. The experimentation involved an examination of the impact on accuracy when increasing a model's depth, width, and resolution. It also revealed that different scaling dimensions were interdependent. Scaling up any dimension of

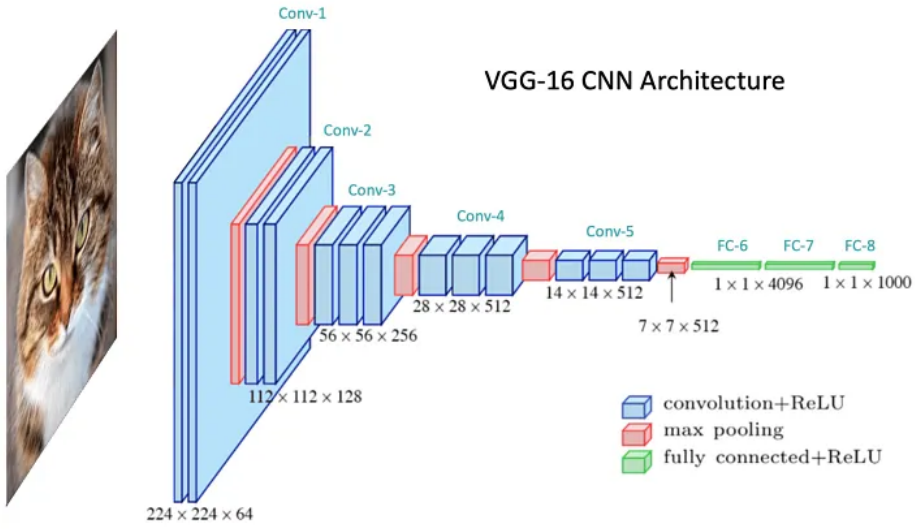

Figure 5.7: Architecture of VGG-16. Source: [66]

width, depth, or resolution enhances accuracy, but the accuracy gain reduces at a certain point; therefore, to attain improved accuracy and efficiency, all the dimensions of the network must be balanced. Consequently, Tan and Le [67] has put forward a compound scaling technique that uniformly adjusts the width, depth, and resolution of a network by employing a predetermined set of scaling coefficients. The comparison of these coefficients with other scales is illustrated in Figure 5.8. The baseline network demonstrated the effectiveness of the scaling method, but it was also a mobile-size baseline known as EfficientNet, which was created using multi-objective neural architecture search to optimize accuracy and FLOPs. Figure 5.9 depicts the architecture of EfficientNet-B0, with the fundamental building piece being MobileNet's mobile inverted bottleneck MBConv, to which squeeze-and-excitation optimization has been added. By beginning with this baseline, the compound scales up method was applied to build the subsequent variations of Efficient Net. The approach consists of two phases that use a fixed compound coefficient ϕ to find the optimal depth α, width β, and resolution γ values to scale up the baseline network.

5.2.3.2 Transfer Learning

Various approaches to Machine Learning exist, and for this project, Transfer Learning has been selected. Transfer Learning involves repurposing a pre-existing model for a specific task and training it to address a new, yet related, problem. As stated in Jukes [60], this methodology is specifically developed to enhance the learning of a target task by facilitating faster learning and enhancing performance. As explained in Pan and Yang [68], there are three main issues to define in Transfer Learning:

- What to transfer: it is important to decide which part of the knowledge can be transferred

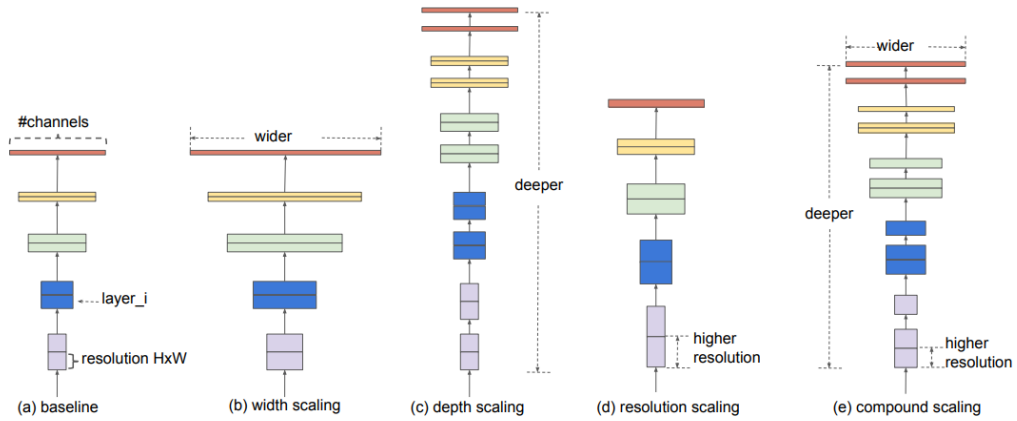

Figure 5.8: Exemplification of model scaling implemented by EfficientNet. Source: [67]

across tasks. Some can only apply to an individual task, while others may be common between the two fields and even can improve the performance of the target task.

- When to transfer: it must be decided in which situations the transfer of knowledge should be done and in which not. If the source domain and target domain are not related, force the transfer would be ineffective and it can even negatively affect the performance of learning in the target domain.

- How to transfer: assuming that the target and source domain are related, another important issue it is how to avoid negative transfer.

There are different categories in which Transfer Learning can divide depending on the situation between source and target domain:

- Inductive transfer learning: when the target task differs from the source task. It can be classified into two distinct groups based on the availability of labeled data in the source domain: one where there is a large amount of labeled data, and the other where there is no labeled data.

- Transductive Transfer Learning: the source and target tasks are the same, while the source and target domains are different. Also, it can be categorized in two cases when the feature spaces between the source and target space are different and when the feature spaces between domains are the same.

- Unsupervised Transfer Learning: like Inductive Transfer Learning, the target task is different but related to the source task. This category is especially focused on solving Unsupervised Learning Tasks in the target domain, defining clustering, dimensionality reduction, and density estimation.

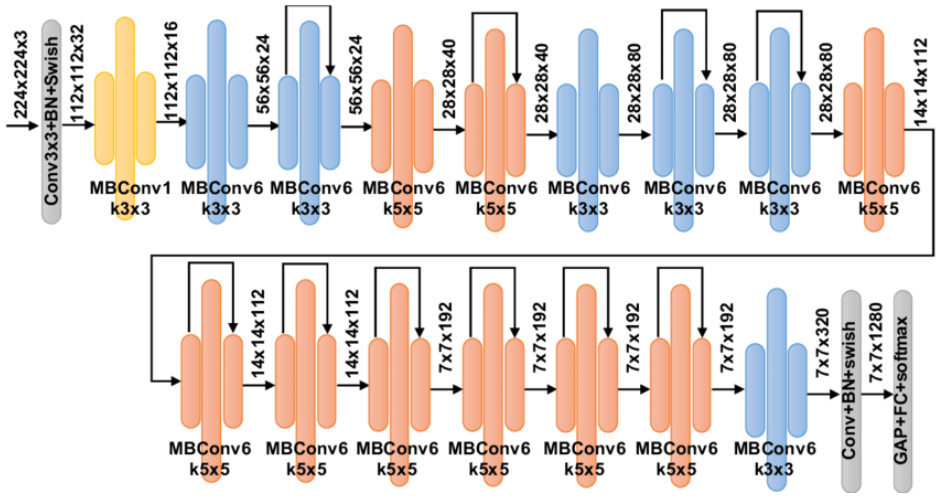

Figure 5.9: Architecture of EfficientNet-B0. Source: [67]

5.2.3.3 Hyperparameters

Machine Learning involves a specific parameter that directly influences the learning process. This is one of the most important distinctions between model parameters, thus hyperparameters are set before training and remain constant throughout training [54]. In contrast, the initialization and modification of model parameters occur throughout the learning process [69]. Tuning hyperparameters is an important procedure for model building since it specifies the model architecture and has a direct impact on model performance. The goal is to develop an algorithm that generates optimal metrics while minimizing loss and exhibiting no indications of overfitting or underfitting. The following section will describe some hyperparameters that were utilized during this investigation.

5.2.3.3.1 Epoch:

refers to one loop of the entire training set [70]. At the end of each epoch, measures like accuracy and loss are calculated. The number of iterations across the training set is determined by the total sample count and batch size [58]. Weights are initially set and then changed until the following epoch [71]. The goal is to have an enough number of epochs to avoid underfitting while also avoiding overfitting. To put it another way, a satisfactory number is needed to obtain enough data from the training set without causing the model to over-train.

5.2.3.3.2 Number layers:

As the name implies, it refers to the number of hidden layers in a model. The intermediate layers, between the input and output layers, execute complex non-linear operations on the presented data. Due to the absence of expected output information during training, these layers are referred to as hidden. It is one of the main components to learn complex patterns and accomplish optimal

performance [72]. Actual models can contain multiple hidden extra layers to learn the most complex task, but at a high computational cost. However, it is unnecessary to build numerous hidden layers. Plenty of problems can be solved with one or two layers; however, if the performance does not match the task or the problem is too complicated, the best solution is to increase the number of hidden layers gradually until the model fits the training set [54].

5.2.3.3.3 Number of neurons per layer:

As previously stated, layers are composed of single neurons that mimic the behavior of biological neurons. Based on the information presented in Géron [54], the number of units in each layer is determined by the difficulty of the problem and the characteristics of the data, considering factors like the size of the image or file. Although it is common practice to construct a pyramid based on neuron size, reducing the number of neurons at each layer; for example, if one layer contains 256 neurons, the subsequent layer may only have 128. However, this does not always ensure the best model performance; in most cases, having the same number of units in all hidden layers can perform just as well, if not better. Nevertheless, under certain conditions, it could be advantageous to have a larger initial hidden layer compared to the others. Finding the best number of layers and neurons is still a complex task that typically necessitates a certain amount of testing, but a simpler approach could involve selecting a model with more layers and neurons than required, or incrementally increasing the number of neurons in each layer while preventing overfitting through early stopping.

5.2.3.3.4 Learning rate:

It is a key factor when designing a model. In each iteration, weight and bias adjustments made by gradient descent are determined by its step size [58]. Equation 5.12 represents this operation, where n_k is the learning rate [73]. Defining a good learning rate is critical for model performance; hence, when implementing a constant rate, if it is too low, the model will take too much time to converge, and if it is too high, the model would most likely be unable to converge.

$$\theta_{k+1} = \theta_k - n_k g_k \tag{5.12}$$

5.2.3.3.5 Batch size:

Batch specifies the optimization procedure that employs the whole training set. This algorithm processes all the training samples immediately in a huge batch. It basically means processing all the training data in a single iteration. However, as shown by Goodfellow lan [56], this word is frequently used to designate mini-batches, which are groups of samples from the training set. As a result, the term batch size is commonly used to describe the number of examples that contain a mini batch and are used in a single iteration of the training process. The size of this hyperparameter has a significant impact on model training time and performance. The batch size should not be too small; otherwise, even if the training time is fast, the algorithm will not receive enough input, resulting

in low precision. On the contrary, using a large batch size will significantly increase training time while providing a more precise estimate of the gradients. However, the goal is to find the best optimization of training time and precision. In most circumstances, a batch size of 10 to 32 suffices to achieve good performance.

5.2.3.3.6 Activation function:

This hyperparameter is specific to feed-forward neural networks. It is a critical function to choose when implementing a hidden layer, as it computes the hidden layer's output value [56]. The majority of the hidden units are described by Equation 5.13, where x represents the vector of inputs, W are the weights, and b denotes bias. Following this process, the element-wise nonlinear function $g(z)$ is applied; this function distinguishes most hidden units from one another.

$$z = W^T x + b \tag{5.13}$$

There are various types of activation functions, and it is difficult to determine which one to employ in each circumstance, as it also relies on whether the layer is hidden or output. Nevertheless, a few of the frequently employed methods will be briefly outlined below, as described in Géron [54]and Goodfellow lan [56].

5.2.3.3.6.1 Output Units:

5.2.3.3.6.2 Sigmoid Units:

Sigmoid units, primarily used in logistic regression, estimate the probability of an instance belonging to a class. Therefore, it has been mostly used for binary classification and for multi-label classification where the label is one hot encoded, as in of this investigation. In multi-label classification, sigmoid activation independently predicts the probability of each class, allowing for instances to belong to multiple classes simultaneously. This function is based on a sigmoid output along with maximum likelihood, that ensures a strong gradient when the model has a wrong answer. The operation in this activation function is defined by Equation 5.14, where the model performs a weighted sum of the input features along with a bias term, resulting in a logistic output. This result is a number that lies between the interval of 0 and 1 defined by Equation 5.15. While sigmoid activation ensures strong gradients for incorrect predictions, it can suffer from saturation issues, leading to vanishing gradients in deep networks. To address this, alternative activation functions like ReLU are often employed.

$$\hat{p} = h_\theta(x) = \sigma(x^T \theta + b) \tag{5.14}$$

$$\sigma(t) = \frac{1}{1 + exp(-t)} \tag{5.15}$$

The sigmoid function outputs the probability that an x instance belongs to a certain class. A threshold of 0.5 is commonly used to determine the predicted class, although this can be adjusted

for imbalanced datasets. If the probability is more or equal to 0.5 then the corresponding class is 1. On the contrary, if the probability is less than 0.5 then the class of the sample is 0. And, if considering Equation 5.14, the sigmoid activation saturates to 0 if $x^T\theta$ has a very negative value, or 1 if $x^T\theta$ is a very negative.

5.2.3.3.6.3 Softmax Units:

considered a generalization of the sigmoid function, are commonly used in multiclass classification tasks because of their ability to produce probability distributions over discrete variables with n possible results.

When an instance x is obtained, the model computes a score $s_k(x)$ for each class, providing the unnormalized log probability. This computation, represented by Equation 5.16, calculates the logit of log-odds, akin to Linear Regression prediction. Each class has its own specialized parameter vector $\theta^{(k)}$, typically organized as rows in a parameter matrix Θ.

$$s_k(x) = x^T\theta^{(k)} \tag{5.16}$$

After obtaining the scores, the likelihood of each class is calculated using the Softmax function, also known as the normalized exponential. The vector $\hat{P}_k$ is computed by exponentiating each score and normalizing it by the sum of all exponentials, ensuring a legitimate probability distribution. Equation 5.17 represents this operation, where K is the number of classes, $s(x)$ is a vector containing the scores of each class for the sample x, and $\sigma(s(x))_k$ is the probability that x belongs to class k.

$$\hat{P}_k = \sigma(s(x))_k = \frac{exp(s_k(x))}{\sum_{k=1}^{K} s_j(x)} \tag{5.17}$$

Finally, the argmax operator then yields the value of k that maximizes the calculated probability, as shown in Equation 5.18. This function provides the "max" in Softmax. Meanwhile, the term "soft" refers to the fact that this activation function is both continuous and differentiable.

$$\hat{P}_k = argmax_k\sigma(s(x))_k \tag{5.18}$$

It's important to note that Softmax units predict only one class at a time and are not suitable for multi-label tasks where classes are not mutually exclusive. For such tasks, alternative approaches or activation functions are recommended.

5.2.3.3.6.4 Other output units:

The most frequent activation functions include linear, sigmoid, and Softmax output units. Other types of output layers can also be developed using maximum likelihood as a guideline. The goal of these designs is to create a function that can specify the predictions, namely the variance for the probability distribution and the cost function for these predictions. This design is determined by the task's details and other aspects, such as computational cost. For example, the heteroscedastic model includes variance as one of the prediction model's output values. Additionally, Gaussian

mixtures are employed in generative models of speech and physical object movements. This model uses multimodal regression to predict values from a conditional distribution that may have various peaks in y space for the same value of x. Ultimately, the goal is to model larger vectors of y with more variables and apply more complicated structures to these output variables. Although this complexity may become too great, it would be preferable to employ other ways to obtain more precise predictions.

5.2.3.3.6.5 Hidden Units:

5.2.3.3.6.6 Rectified Linear Units (ReLU):

The operation in this function is defined by $g(z) = \max(0, z)$. While continuous, ReLU is not differentiable at $z = 0$, leading to abrupt changes in slope that can cause Gradient Descent to fluctuate. Its derivative 0 for negative values and z for all other values. Despite this non-differentiability, ReLU is favored in neural networks for its computational efficiency and lack of maximum output value, simplifying optimization. It is mostly based on the concept that models with more linear behavior are easier to optimize. One of the main limitations of Rectified Linear Units (ReLU) is its inability to learn from gradient-based methods when the activation is zero. There are multiple generalizations of this activation function, most of which have the potential to outperform or match Rectified Linear Units. They are primarily designed to address some weaknesses of rectified linear units or to exploit special circumstances in object image identification to recognize specific image properties.

5.2.3.3.6.7 Logistic Sigmoid and Hyperbolic Tangent (tanh):

Sigmoid functions are more commonly used as output units. Nevertheless, it can also function as a hidden unit. It saturates to a high value when z is very positive and to a low value when z is negative; when z is 0, the function becomes extremely sensitive. This saturation can make gradient-based learning extremely difficult, hence its use as a hidden unit is discouraged. As a result, when a function of this type is suitable for use, the hyperbolic tangent activation function outperforms the logistic sigmoid.

The Hyperbolic Tangent Function is comparable to the logistic function because it is S-shaped, continuous, and differentiable. Its value might range between -1 and 1. This allows the output to be more or less centered on zero at the start of the training, accelerating convergence.

Because these two functions are closely connected, the hyperbolic tangent (Equation 5.19) can alternatively be represented as Equation 5.20.

$$g(z) = tanh(z) \tag{5.19}$$

$$g(z) = 2\sigma(2z) - 1 \tag{5.20}$$

Sigmoid and hyperbolic tangent activations are more common in other types of networks or models than in feed-forward networks, such as recurrent networks, where using linear activation functions degrades model performance

5.2.3.3.6.8 Other hidden units:

Other types of hidden units exist but are rarely used. Furthermore, the design of concealed units remains a study field, as are many activation functions. Their design can begin by incorporating characteristics from other activation functions and applying them to other activities or limiting their functionality, such as hard tanh. Some of them can even perform similarly to the more popular functions outlined above. Still, they must be subjected to a series of tests to demonstrate their utility and superiority over alternative activation functions available. However, the majority of activation functions cannot match the reliability of the most often used ones.

5.2.3.3.7 Optimizer:

it has been noted by Ketkar [74] that a key difficulty in model training lies in determining the parameters that minimize the loss function, which measures the model's performance and complexity. The minimizing of the loss function becomes more complex due to the possibility of having multiple parameters that can be represented as a scalar, vector, or matrix. The Steepest Descent is commonly used for this purpose, changing the parameters of x to minimize the loss $L(x)$. This is achieved by calculating the gradient $\nabla_x L(x)$, indicating the direction where the loss function increases the most. The model parameters are updated in the opposite direction of the gradient, with a step size α determining the step size of the update. This process is represented in Equation 5.21. The ideal is to avoid a high value of α; thus, it leads to a larger step towards u and overshooting an optimal solution. This iterative process, also known as gradient descent, is the basis for optimization algorithms, such as stochastic gradient descent.

$$x = x - \alpha \nabla_x L(x) \tag{5.21}$$

Next, will be briefly explained some of the most common optimization algorithms based on the descriptions from Ketkar [74]. As noted earlier, certain techniques mentioned here are adaptations of the Stochastic Gradient Descent algorithm.

5.2.3.3.7.1 Batch Descent:

The entire dataset is used in this approach, thus for the update is considered the gradient of the loss functions evaluated over the entire dataset. The advantage of this technique is its accuracy, thus it computes everything with the entire dataset, although it is considerably computationally expensive.

5.2.3.3.7.2 Stochastic:

There are two types of cases, Stochastic single and stochastic mini batch. In the first type, the algorithm chooses a random single sample to calculate the update. In the second type, a limited subset is chosen from the database in each iteration, with each iteration being assigned a distinct, randomly selected set of samples. The algorithm is widely employed due to its cost-effective computational nature and generally superior accuracy.

5.2.3.3.7.3 Momentum:

The equation for momentum is similar to Equation 5.21 but takes into account the underlying intuition behind this approach during the update process. The idea is to use a fraction of the previous update as a parameter for the current update. Equation 5.22 is an enhanced version of Equation 5.21 with momentum, where u_{s-1}denotes the update in the previous step. The momentum term causes the new step direction to be biased by the previous step direction, which can reduce oscillation and facilitate convergence.

$$x = x - (\gamma u_{s-1} + \alpha \nabla_x L(x)) \tag{5.22}$$

5.2.3.3.7.4 Nesterov Accelerated Gradient (NAS):

Equation 5.22 exhibits similarity to Equation 5.23, with the distinction being the incorporation of a "correction factor" in the latter. The objective of this algorithm is to look one step ahead and approximate the behavior of the next gradients to improve results.

$$x = x - (\gamma u_{s-1} + \alpha \nabla_x L(x - \gamma u_{s-1})) \tag{5.23}$$

5.2.3.3.7.5 Adagrad:

In previous methods, the learning rate remained constant while the parameters were updated. Adagrad adapts the learning rate for each parameter in the loss function based on prior gradients. Equation 5.25 represents the operation where: x_i represents each of the parameters of the loss function and g_i^S denotes the gradient for x_i at step s. G, calculated in Equation 5.24, is the sum of the squares of the gradients for each actual step to the previous step, given for steps $0, 1, \ldots, S-1$, which have the corresponding gradients $g_i^0, g_i^1, \ldots, g_i^S$. One of this algorithm's deficiencies is that when G grows, the learning rate in each parameter falls, and progress stops.

$$G = (g_i^0)^2 + (g_i^1)^2 + ... + (g_i^{S-1})^2 \tag{5.24}$$

$$x_i = x_i - \frac{\alpha}{G^{\frac{1}{2}}} g_i^S \tag{5.25}$$

5.2.3.3.7.6 RMSProp:

Rather than using the entire collection of gradients, this approach computes G over the previous W steps. It is more computationally efficient than methods such as Adagrad, thus it calculates the sum of the exponentially declining average square of gradients rather than storing all gradient values and then computing G. Equation 5.26 calculates the cumulative squared gradients, with ρ being the decay.

$$E[(g_i)^2]^S = \rho E[(g_i)^2]^{S-1} + (1-\rho)(g_i^S)^2 \tag{5.26}$$

Equation 5.28, similar to the Adagrad update, represents the computation of the update, where $G^{\frac{1}{2}}$ is the root-mean-square of g_i (Equation 5.27).

$$RMS[g_i] = G^{\frac{1}{2}} = \sqrt{E[(g_i)^2]^S} \tag{5.27}$$

$$x_i = x_i - \frac{\alpha}{RMS[g_i]} g_i^S \tag{5.28}$$

5.2.3.3.7.7 Adadelta

It is considered a variant of AdaGrad. This algorithm is constructed based on whether the unit of the parameter and the update of the parameter are the same. The update rule of this algorithm is depicted in Equation 5.29, where the factor α disappears. So, this algorithm does not use the learning rate as a parameter. As it can be seen, $RMS[\Delta x_i]$ should represent the root mean squared error for the parameter updates. But since this value is unknown, it is used the approximation calculated with the parameter's updates until the previous step $RMS[\Delta x_i]^{S-1}$).

$$x_i = x_i - \frac{RMS[\Delta x_i]^{S-1}}{RMS[g_i]} g_i^S \tag{5.29}$$

5.2.3.3.7.8 Adam:

Its term is derived from adaptive moment estimation. It is a combination of momentum and RMSProp, as it uses an exponentially decaying average of past gradients as momentum optimization and, like RMSProp, one of its parameters is the exponentially decaying average of past squared gradients. This operation is represented in Equation 5.30.

$$x_i = x_i - \frac{\alpha}{E[(g_i^{S-1})^2]} E[g_i^{S-1}] \tag{5.30}$$

At the start of training, both $E[g_i^{S-1}]$ and $E[(g_i^{S-1})]^2$ are biased towards zero. Therefore, there are two decay rates, ρ_1 and ρ_2, which are employed for $E[g_i^{S-1}]$ and $E[(g_i^{S-1})^2]$ respectively. This helps to correct the bias by computing the operations shown in Equation 5.31 and Equation 5.32.

$$E[g_i^{S-1}] = \frac{E[g_i^{S-1}]}{1-\rho_1} \tag{5.31}$$

$$E[(g_i^{S-1})^2] = \frac{E[(g_i^{S-1})^2]}{1-\rho_2} \tag{5.32}$$

5.2.3.4 Evaluation Metrics and validation techniques.

Evaluation metrics come to hand to measure the performance of the model, which helps to make conclusions about how the model is doing, make improvements to the model or make comparisons between models' performance. Moreover, validation techniques play a crucial role in guaranteeing the reliability of the model's predictions. The following section will conceptually explain the metrics and validation techniques used in this investigation.

5.2.3.4.1 Cross-validation:

A popular method for evaluating model performance is to divide the dataset into training and validation sets, which can also be partitioned into a separate test set. Nevertheless, performing a solitary test and evaluation may lead to statistical uncertainty and does not ensure the absence of overfitting. Cross validation was designed to prove the model's robustness. Its objective is to repeat the training and testing with subsets of the original dataset [56]. Cross-validation divides the data into k equal parts. One of the k partitions is employed as a test set, while the remaining segments serve as a training set. The process is then repeated k times, with each iteration evaluating the training process in a separate k segment [57]. In addition, some evaluation metrics can be applied in each iteration. One of the most popular is to examine and estimate the mean accuracy of the complete cross-validation procedure. Other evaluation measures include the F1-score, the confusion matrix, and even graphs of validation and training accuracy and loss. The objective is to observe the performance of these measures and assess the average performance of the model. Another goal is to ensure its reliability by avoiding signals of overfitting in all iterations. The selection of the number of iterations should be based on tests and specific applications. Typically, the most common values for k are 10 or 5. However, in certain cases, they may be equal to the number of classes present in the dataset.

The Keras library offers support for a type of cross-validation known as StratifiedKFold Cross Validation [75]. This approach uses stratified sampling to generate folds with a representative ratio for each class. The previously outlined technique is used in every iteration to produce model performance measurements. Unfortunately, this approach does not support multi-label datasets. For this experiment, a variant of the Keras stratified cross validation function was used. This variant was generated by Bradberry [76] using the algorithm described in Sechidis et al. [77], which separates the training into subgroups while retaining the percentage of samples for each label. First, the algorithm determines the number of samples that should be included in each subset, as well as the percentage of samples from each label. Then it looks at one label in each iteration, the one with the fewest remaining samples and the one with the most desired examples. Finally, the samples are

added to the current subset and then removed from the dataset. This step is performed until the dataset is empty. The algorithm's goal is to identify a suitable subset for distribution for each label.

5.2.3.4.2 Accuracy:

Determines how frequently predictions match the labels of the training sets. The method used to calculate accuracy varies according to the type of classification task. As an illustration, in multiclass classification projects, the metric typically used is Categorical Accuracy, which is employed to compare one-hot encoded labels. The primary metric used in the analysis of this investigation is Binary Accuracy. Binary accuracy generates two local variables: "total" and "count", which are used to calculate the frequency with which predictions match actual labels. The result is defined as binary accuracy and is calculated by dividing the "total" by the "count", reflecting the ratio of correctly classified samples to the total number of samples [78].

Accuracy can also be calculated using elements from the confusion matrix. It can be expressed as the sum of the True Positives and True Negatives, divided by the total number of samples, which is the sum of all elements in the confusion matrix, as shown in Equation 5.33.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.33}$$

5.2.3.4.3 Loss:

Loss functions, alternatively referred to as error or cost functions, quantify the disparity between the expected output of a model and the true labels derived from a training set, as explained by Wu et al. [79]. The loss function provides a score based on this difference, which evaluates the model's rules as it learns from the dataset. It also defines an error minimization target in terms of the data collected that is expected to be seen during the training epochs. This score may also assess the model's performance. This allows us to determine the model's behavior in terms of certain aspects, such as model optimization, how it responds to imbalanced data, and the use of complex metrics. With respect to this analysis, consider making some model adjustments, such as hyperparameter tuning. Additionally, it can be used to compare the performance of several models.

Keras allows you to implement loss functions as standalone functions or as parameters to modules like "fit" or "compile". Just like accuracy, there are multiple approaches to compute loss, which differ based on the classification or regression problem (binary, multiclass, multilabel, etc.), the data type, and the research or training objective. The model needs to update the weights properly, lower the loss for future epochs, and avoid problems such as data bias, data imbalance, incorrect labeling, and so on. Using the improper loss function causes these issues. As an example, if the goal is to prepare the model for difficult samples in a binary classification job, Binary Focal Cross Entropy Loss [80] may be a viable alternative, since it uses a focal factor to reduce the weight of easy occurrences and focus on the most difficult ones. The loss function employed in this project is Binary Cross Entropy.

Binary Cross Entropy is mostly applied to binary classification problems and some multi-label applications. Cross-Entropy calculates a score by measuring the average disparity between the

predicted and actual probability distributions. This score is logarithmic, meaning it becomes significantly small for minute differences and extremely larger for substantial disparities [Brownlee]. [81]. The score is then minimized, and the closer the score to zero, the better the model's performance.

5.2.3.4.4 F1 score:

It represents the harmonic mean of precision and recall [78]. This metric's values vary between 0 and 1, with the closest value to 1 reflecting the model's best performance. Thus, when precision and recall are similar, this metric approaches the mean. However, if precision and recall are considerably different, $f_1 - score$ will be close to zero. As a result, it is a reliable indicator for handling imbalanced data or assessing overall model performance. Then it will be easy to determine which variables require attention and which improvements to apply to the model. The metric's value can be derived using the True Positives, False Positives, and False Negatives (Equation 5.34), or by utilizing the Recall and Precision values (Equation 5.35).

$$f_1 - score = \frac{2 * TP}{2 * TP + FP + FN} \tag{5.34}$$

$$f_1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.35}$$

There are various variations of this statistic that consider the weights of each class or respond to changes in class distribution. These versions can be employed in certain cases of data imbalance to obtain a more precise determination of the weights for each class.

5.2.3.4.5 Recall:

The percentage of positive class samples that were successfully predicted is calculated as the ratio of True Positives to the actual number of positive samples in the sample set [78], as represented by Equation 5.36.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{5.36}$$

5.2.3.4.6 Precision:

This metric calculates the ratio of True Positives to the total number of positive samples predicted by a model, showing the portion of samples that were accurately predicted as Positives within the positive predictions [82] [58]. This operation is shown in Equation 5.37:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{5.37}$$

5.2.3.4.7 Confusion Matrix:

It summarizes a model's correct and erroneous predictions based on some test data [58]. The structure comprises a two-dimensional square matrix, where one axis signifies the true category of the samples, and the other axis represents the predicted label by the model [82]. The number of elements in each axe is determined by the classes in the collection. This metric shows the number of false positives, true positives, false negatives, and true negatives predicted by the classifier. Figure 5.10 shows an example of this measure.

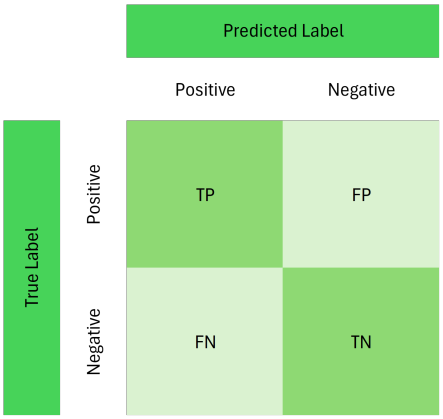

Figure 5.10: Binary Confusion Matrix

In comparison, Figure 5.11 depicts a confusion matrix that arises from a multiclass classification task encompassing numerous classes. The matrix in this scenario showcases the confusion between every two classes. In sklearn [83], there is also a confusion matrix intended for multi-label classification problems, where class-wise confusion matrices are computed that resemble the confusion matrix in Figure 5.10, representing the count of positive and negative predicted samples for each class that were correct or incorrect.

5.3 Related Work

This undergraduate focus on implementing several of the augmentations techniques from the investigations that will be mentioned in this section. In addition, it will test three of the most used and powerful networks to check the utility of transfer learning in the bioacoustics field. While previous studies have explored data augmentation and transfer learning for bioacoustics tasks, few have investigated their combined effect on multi-label classification of anuran species. This research investigates the potential of combining data augmentation techniques from various bioacoustics studies with transfer learning in CNN architectures to improve the classification accuracy for multi-label anuran calls. The methods and tools implemented in this project are based on the information obtained from the studies presented in the next subsections.

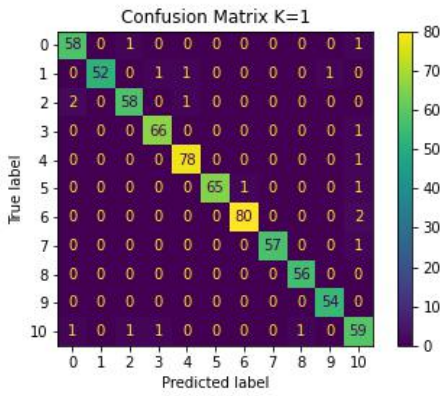

Figure 5.11: Multiclass Confusion Matrix

5.3.1 Bioacoustics

Browning et al. [5] conducted a comprehensive study on the implementation of Passive Acoustic Monitoring, addressing various aspects and challenges associated with it. Their research highlighted that analyzing the vast amount of data obtained from recording periods posed a significant challenge. They also suggested that machine learning could be a valuable tool for identifying and studying Passive Acoustic Monitoring and bioacoustics signals in general.

Then, observing what was proposed by Browning et al. [5], machine learning has proved to be an effective tool for processing bioacoustics data; in this project, it will be especially focused on animal classification, primarily anuran classification. One of the completest works about this task is the one from Stowell [6] who provided a complete roadmap to display the state-of-the art and developments in bioacoustics using machine learning. Also, it gave some guidelines to follow considering how most investigations in bioacoustics using machine learning are developed. These recommendations involve the utilization of well-known CNN architectures, implementation of data augmentation techniques, selection of appropriate data types for model input (spectrogram, MFCCs), and adherence to effective training practices such as regularization methods and hyperparameter tuning. This guide provides references for various taxons such as birds, cetaceans, bats. It also discusses the acoustic features used, including spectrograms, waveforms, and MFCCs. Additionally, it explores the different tasks in which machine learning is applied, such as classification, clustering, and denoising. Finally, it highlights potential topics and gaps in the field that can be explored for further contributions to bioacoustics with machine learning. It was a good reference for complementary investigations to this work and also to check information about good practices in bioacoustics analysis with machine learning.

In the papers reviewed, some recommended by Stowell [6], mostly the investigators focused on one of the two principal focus points of the current investigation: data augmentation or machine learning. In the next following subsections, some works related to these topics will be described. Firstly, the investigations that have primarily contributed to implementing data augmentation techniques in

this project will be described. And, afterwards, will be presented some works reviewed for CNNs architectures with transfer learning applied. Nevertheless, it is worth noting that certain works may be mentioned in both segments.

5.3.2 Data Augmentation

Data augmentation has been used for multi-label anuran classification, such is the case of the projects from Moummad et al. [13] and Cañas et al. [10] who used SpecAug, an augmentation library for spectrogram signals. This library was developed by Park et al. [47], who elaborated three techniques: Time Warping, Frequency Masking and Time Masking. Moreover, the influence of this library extends to the Time Masking technique discussed in Section 5.2.2.2.1, and its application has been observed in diverse projects involving animal sound classification. An illustrative case is A and Rajan [11], where these modifications were incorporated for multi-label bird classification. Other works have also implemented several augmentations techniques and were directed to animal classification, such is the case of Sun et al. [8], who investigated several augmentations techniques to classify groups of species in a rainforest. The utilization of augmentation techniques in their work encompassed foreground sound noise, cropping axis from the spectrogram, and rolling the signal. Notably, the present project also considered the inclusion of foreground noise and rolling.

It is also necessary to mention two more papers that are directly connected to spectrogram augmentation but not to bioacoustics analysis, which were also examined for this project. These projects are from Song et al. [48] and Inoue et al. [39]. Song et al. [48] introduced SpecSwap (Section 5.2.2.2.2), a technique that enables the swapping of sections within spectrogram signals for speech recognition. In a similar vein, Inoue et al. [39] proposed "Shuffling and Mixing" (Section 5.2.2.1.4.1), a technique that involves shuffling and mixing segments of spectrograms, resulting in enhanced performance of CNNs in the classification of environmental sounds.

It is important to note that all previous investigations only considered augmentation for spectrogram signals, not other types of signals, as will be the case with Nanni et al. [7]. Nanni et al. [7] suggested using a variety of signal processing techniques to augment two different datasets: bird songs (Xeno-canto) and cat sounds. The augmentations were divided into four categories: Standard Image Augmentation, Standard Signal Augmentation, Spectrogram Augmentation and Signal Augmentation. The distribution of these categories was determined based on the type of signal used for augmentation (waveform or spectrogram) and whether it relied on established computer science augmentation techniques. In total were tested 33 augmentation techniques in this investigation. Though it is not directly related to multi-label anuran classification, the work from Nanni et al. [7] inspired the categorization implemented in the present undergraduate project, thus it introduced both audio and spectrogram augmentation.

It was necessary to obtain more information to implement audio augmentations given the ones provided by Nanni et al. [7], that is the reason the investigation from [41], Salamon et al. [44] and Lucas Ferreira-Paiva et al. [9] were explored. Jordan [41] and Salamon et al. [44]developed two python libraries that contain augmentations functions, specifically created for audio signals. Some audio augmentation functions implemented in this work were based on the functionality of

Jordan's Audiomentations [41] library. Meanwhile, Scaper Library [44] was directly implemented in this project, it consists in a pretty complete library for audio augmentation. It has optimized and complex functions such as time stretching, pitch shifting and mixing the signal with a background noise. Additionally, Scaper provides the capability to blend these three effects and apply them to the signal. It also allows users to specify other parameters, such as the duration of the effects, the point in the signal where the augmentation should begin, and the signal-to-noise ratio.

Spectrogram augmentations were also investigated according to the methods exposed by Nanni et al. [7], were inquired Takahashi et al. [49] who proposed Equalized Mixture Data augmentation (paragraph 5.2.2.2.3), and Kim et al. [84] and Sarkar and Tan [42] who developed Vocal Tract Length Perturbation (paragraph 5.2.2.1.8). These augmentations techniques were created for speech recognition but could also be used for animal classification, thus they perform complex operations to warp the spectrogram signal.

Finally, similarly to Nanni et al. [7], Lucas Ferreira-Paiva et al. [9] presented a comprehensive survey of data augmentation techniques. These techniques encompass audio, spectrogram, and even image-oriented augmentation methods, some of which served as inspiration for the ones employed in this project. When considering the last type of augmentation mentioned, the study conducted by Zhou et al. [50], which focused on implementing image augmentations for cardiac spectrogram analysis, was taken into consideration.

Overall, the cited works showcased the ability of data augmentation to enhance the performance of certain models, resulting in improvements ranging from a marginal increase of 0.5% to a significant accuracy gain of 50%. However, Nanni et al. [7] and Lucas Ferreira-Paiva et al. [9] also emphasized that augmentation methodologies should be carefully selected based on the application, as some image or audio augmentation techniques may be harmful to the model because they significantly change the semantic content of the signal. Regarding this work, it is crucial to mention that it has not been found a work that implements several augmentation techniques from audio to image to multi-label anuran signals.

5.3.3 Transfer Learning

In this subsection will be provided several investigations that were mostly focused on evaluating CNN architecture with transfer learning. While some researchers, namely Dufourq et al. [85] and Palanisamy et al. [86], utilized well-known augmentation techniques, such as time shifting, time stretching, and pitch shifting. The main focus was to evaluate the effectiveness of various CNN architectures using transfer learning.

Then the CNN architectures implemented in this project were selected based on similar studies that compared different neural network architecture with transfer learning in bioacoustics classification tasks or for image or audio classification. Following the analysis, two approaches were identified: those that compared multiple CNN designs [7] [85] [87] [88] [86] [89] [12], and those that assess the performance of a single CNN with diverse sets of experiments [10] [8] [90]. The analysis revealed that VGG, Resnet, DenseNet, Inception, and MobileNet were among the most frequently used architectures. Notably, DenseNet, VGG, and Resnet consistently yielded the most impressive

outcomes. Finally, the various metrics used to measure the models' performances were noted; the most popular ones were the confusion matrix, accuracy, and F1-score, which were also employed in the current experiments.

In summary, the aforementioned studies highlighted the capacity of transfer learning to identify audios or images, especially in bioacoustics. Also, it was also possible to determine which function fulfilled each layer for transfer learning in the process of feature extraction and fitting a new task as mentioned by Palanisamy et al. [86]. In all these investigations, it is particularly underlined the capacity of these models to achieve high metrics while having faster convergence and shorter training time, thus it is recommended for projects with limited resources. However, some of them also mentioned the limits of these algorithms. Dufourq et al. [85] recommends conducting more experiments on small datasets to evaluate their performance in such limited contexts. Nevertheless, it is worth noting that data augmentation techniques can significantly enhance overall performance. Khalighifar et al. [90] notice how the accuracy of a model can decrease if the number of classes is too big and if the calls between the species have similar attributes. The significance of data quality and the selection of suitable augmentation techniques were emphasized by Nanni et al. [7], Zhou et al. [50], and Lucas Ferreira-Paiva et al. [9] as means to enhance the model's performance. Furthermore, additional conclusions regarding the impact of network architecture were drawn. For instance, Yang et al. [88] concluded that the number of layers in a neural network does not necessarily correlate with improved performance. This was evident in the comparison between Resnet and EfficientNet, where the metrics of the latter slightly outperformed the former.

Except for A and Rajan [11], Moummad et al. [13], and Cañas et al. [10], most of the investigations mentioned above were multiclass classification tasks; thus, multi-label classification applied to bioacoustics signals is still a recent and emerging field, and these investigations are typically focused on bird species, as with A and Rajan [11], who implemented a multi-label bird classification model with spectrogram augmentation techniques but using a BiGRU and a Swin-transformer model. Anurans have been the subject of previous investigations conducted by Xie et al. [12], Moummad et al. [13], and Cañas et al. [10]. The CNNs AlexNet, CaffeNet, and VGG16 were tested in Xie et al. [12], resulting in an average accuracy of 0.71. The author also recommends exploring the potential of data augmentation for similar experiments. Cañas et al. [10]generated an extensive dataset with 42 different species from Brazilian biomes and it were tested in different Resnet architectures, the best macro F1-score obtained was 37.8 from ResNet152 leaving it as a new challenge for further exploring in different experiments such as the one in this investigation. Moummad et al. [13] approached this challenge by testing a framework that leverages mixing regularization methods and testing in MobileNet.

As it was described, only two works were focused on testing data augmentation and transfer learning, which were the ones from Nanni et al. [7] and Sun et al. [8]. However, it did not give particular attention to either multi-label classification or anuran species. The next works found in were mainly centered on transfer learning. Investigations in the realm of anuran multi-label classification primarily focused on a single type of CNN (Resnet or MobileNet), neglecting the exploration of diverse augmentation techniques. However, Xie et al. [12] departed from this approach by examining different neural network architectures and emphasizing the adoption of augmentation

techniques to enhance model performance.

Chapter 6

Materials and Methods

6.1 Process Design.

Figure 6.1: Phases of the project

The aim of this undergraduate project is to assess the efficiency of combining data augmentation with transfer learning for classifying multi-label samples of anuran sounds. The project follows several steps, as shown in Figure 6.1. These steps will be explained in the subsequent sections of this chapter.

The first stage is Data Collection (Section 6.3), where we describe the method used to generate or obtain the samples of anuran chants. The next phase, described in Section 6.4, involves preprocessing the data before augmentation and providing it to the model.

Explained in Section 6.5, the data augmentation algorithm used in the third phase includes the selection and application of augmentations. Section 6.6 discusses the selection of the models mentioned in Section 5.2.3.1.2.

Later in the document, the training process and experiment design are discussed in Section 6.7, focusing on the selected models. The process for selecting evaluation metrics and their utilization is described in Section 6.8.

Additionally, Section 6.2 provides important details about the materials used in this project, including information about the hardware, libraries, and other relevant aspects.

6.2 Resources.

The experiments were performed using a desktop PC with the following specifications:

- Nvidia RTX 3060 Ti.
- 32 GB RAM.
- AMD Ryzen 7 5700G, 3.80GHz.

Furthermore, it was used Google Drive Storage of 100 GB to use it as a backup storage for project files.

The experimental setup was implemented in the ecosystem anaconda with Python 3.8.7. Machine learning models were implemented using Keras 2.4.3 and TensorFlow 2.3.0. Additional libraries were included to apply augmentation techniques and process audio signal. The principal libraries used in the current investigation are:

- **Pandas(v1.5.3):** It comprises an essential library to manipulate DataFrames. The usual representation for this data is a two-dimensional array type, composed of columns and rows. The main application of this is to create or represent databases and manipulate them. This library offers various functions. Some of these functions include creating a DataFrame, filtering it by values in its columns or rows, adding a column or a row to an existing DataFrame, and storing it in different formats such as csv or pickle. The library can also assist in reading files like csv from Excel.

- **Torch(v2.0.1):** This library was specifically designed to support tensor operations on GPUs and facilitate construing deep neural networks. Its functionality includes creating layers and activation functions to construct blocks for these models. Only tensor operations were used in this project to implement augmentation functions and apply them. Specifically, torches were converted to Numpy arrays using some functions, and other operations like the short-time Fourier Transform and the inverse short-time Fourier Transform were also performed.

- **Torchaudio(v2.0.2):** Usually included in Pytorch, it comprises a library that provides several audio and signal processing functions using torches. For this investigation, it was mainly used for audio pre-processing: for reading audios, resizing them, resampling them, standardize it in one channel and to generate spectrograms.

- **Scikit-learn(v1.2.0):** is a widely used module for machine learning and data mining, offering support for classification, clustering, and regression algorithms. It also provides tools for feature selection, data pre-processing, and model performance evaluation. In the current investigation, Scikit-learn was utilized specifically for evaluating the performance of our model. The Classification Report and confusion matrix, showcased in the Metrics section, were implemented using this library. Additionally, the Cross Validation algorithm employed in this project is based on the Stratified Cross Validation technique from Scikit-learn.

- **Scaper(v1.6.5):** This library was principally used to implement audio augmentations. The audio augmentations included were Time Stretching (Section TimeStrecth), Pitch Shifting (Section PitchShifting) and Audio Mixing (Section NoiseInjection). This library offers some benefits, such as creating several new soundscapes from the audios in the signaled folders. Additionally, it is possible to integrate these augmentations, manage the timing of their implementation within the original audio, determine the duration of the new soundscape, and adjust the signal-to-noise ratio within it.

- **Numpy(v1.23.5):** A renown library in python has several operations and is broadly used for operations with arrays. In relation to the present investigation, specific augmentations, such as clipping, were used. However, its primary usage entailed generating random numbers from a normal distribution to obtain some parameters for certain augmentations functions.
- **Matplotlib(v3.7.1):** is primarily used for creating statistical, animated, or interactive visualizations in Python. Therefore, in this project, Matplotlib was used to generate graphs for various metrics, including confusion matrices, accuracy, and loss curves.

6.3 Data Collection.

Acquiring multi-label datasets for anurans can be quite challenging, thus there is limited availability of public data. The number of samples for each subspecies is insufficient, and even those available are typically not multi-label. However, luckily for this investigation, two approaches were implemented to tackle this issue. The first was generated using samples from databases like Toledo [91] and Lis [92], with more details described in Section 6.3.1. The second datasets, explained in Section 6.3.2, is Anuraset reunited by Cañas et al. [10], it is composed of several audios reunites from different locations in Brazil.

6.3.1 Created dataset

This approach comprised creating "synthetic samples" by randomly combining multiclass samples from Toledo [91] and Lis [92]. From these sources, it was reunited sounds from 11 different species. The files had varying lengths, ranging from a few seconds (20, 30, etc.) to several minutes (1 to 20). In order to generate more samples, it was developed an algorithm to divide the original audio files into smaller segments of 5 seconds. By doing this, it was possible to integrate them with other audio files from the dataset, consequently generating a larger repertoire of samples.

The pseudocode in Algorithm 1 displays the integration algorithm. It takes in the actual DataFrame, the classes, and the directory to save the new samples as input. The algorithm determines the number of times the sample should be mixed, either 3 or 4, which represents the iterations in a for loop. Every iteration determines the classes to mix for that sample. The number can vary from 1 to 5, producing a class vector and a mixed index vector corresponding to that sample. In this operation, there is a fixed probability where smaller numbers are more likely to appear than larger numbers. Following that, the function conducts a comparison between the mixed indexes and other mixed indexes to prevent the merging of duplicate samples. The mixing process is performed using a for loop, with the maximum number of iterations defined by the number of elements in the index array. In each iteration, the algorithm selects an index from the vector, loads it, standardizes it, and adds it to a variable that accumulates the audios. Finally, the algorithm generates the file name with the mixed indexes, creates the file path, and stores it using Torchaudio. Ultimately, the algorithm includes this information in the dataset with the other samples.

With 7044 samples, the resulting dataset showcases the distribution of samples per class after the mixing, as shown in Figure 6.2. Evidently, there is an imbalance among the classes which will

Algorithm 1 Algorithm to divide the audio in segments and combine them

Require: Original dataframe, classes, folder to save new samples
Ensure: New metadata file with the divided and combined samples

```
final_dataframe= copy of the original dataframe;
number_items= length(Original_Dataframe);
read file name;
create a new column in dataframe copy to describe the index mixed;
default_folder= establish the default folder where the samples will be stored;
for i in number_items do
    temp_sample= copy of Original_dataframe[i];
    obtain the label from the sample;
    number_times=randomly decide the number of times to mix the sample (3 or 4);
    for j in number_times do
        number_classes= randomly choose the number of classes to mix (from 1 to 5);
        vector_classes= randomly decides which classes will be mixed according to number_classes;
        vector_idx= randomly chooses the indexes to be mixed and compared them to avoid repeated
        samples;
        mixed_audio=0;
        final_name= string();
        for m in vector_idx do
            Audio= read audio;
            Channel conversion of Audio to 1;
            Resample Audio to 44100;
            Resize Audio to 5s;
            mixed_audio= mixed_audio + Audio;
            update label values in temp_sample;
            if m=0 then
                final_name= string(vector_idx(m))
            else
                final_name= final_name + "_" +string(vector_idx(m))
        path_to_audio= default_folder + final_name + ".wav"
        store new audio in drive;
        temp_sample[File Path]=path_to_audio;
        temp_sample[Index Mixed]=vector_idx;
        concatenate new sample with other samples in final_dataframe;
return final_dataframe;
```

be addressed through data augmentation in Section 6.5. The disparity is evident in the contrasting numbers of examples between the class "Leptodactylus Labyrinthicus" and the class "Ameerega Picta", with the former nearly doubling the latter.

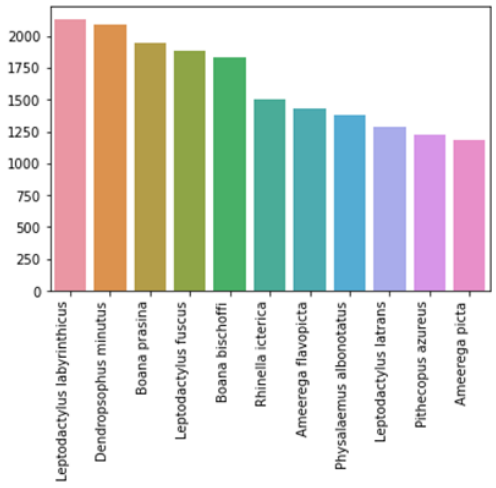

Figure 6.2: Distribution of the mixed samples dataset

Table 6.1 shows how the dataset is organized. There are 13 main columns. The "File Path" column shows the location of the audio file. Different species are represented by the following columns, indicating their presence (1) or absence (0). The species included in these columns are Leptodactylus Labyrinthicus, Dendropsophus Minutus, Boana Prasine, Leptodactylus Fuscus, Boana Bischoffi, Rhinella Icterica, Physalaemus Albonotatus, Ameerega Flavopicta, Leptodactylus Latrans, Pithecopus Azureus and Ameerega Picta. The last column contains the indexes of the original dataset that were mixed to create the multi-label dataset.

File_Path	Lept...icus	Den...nutus	Boa...sina	Lept...scus	Boa...offi	Rhin...rica	Phys...tus	Ameer...icta	Lepto...rans	Pithe...reus	Amee... picta	Index Mixed
\Audios\...\FNJV_0001409...part 0.wav	0	1	0	0	0	0	0	0	0	0	0	
\Audios\...\0_1541.wav	0	1	0	0	1	0	0	0	0	0	0	[0, 1541]
\Audios\...\0_420_925_638.wav	0	1	1	1	0	1	0	0	0	0	0	[0, 420, 925, 638]
\Audios\...\0_1549_492_1064.wav	1	1	0	1	0	0	0	0	0	0	1	[0, 1549, 492, 1064]
\Audios\...\FNJV_0001409...part 1.wav	0	1	0	0	0	0	0	0	0	0	0	
\Audios\...\1_712.wav	0	1	0	0	0	1	0	0	0	0	0	[1, 712]
\Audios\...\1_1556_1081_1276.wav	1	1	0	0	0	0	1	0	0	0	1	[1, 1556, 1081, 1276]
\Audios\...\1_306_492_1546_1287.wav	0	1	0	1	1	0	1	1	0	0	0	[1, 306, 492, 1546, 1287]
\Audios\...\1_1459.wav	0	1	0	0	1	0	0	0	0	0	0	[1, 1459]

Table 6.1: Information structure in the mixed samples database

6.3.2 Anuraset.

This is a project presented by Cañas et al. [10]. It aimed to tackle the scarcity of accessible data for anurans, which are among the most endangered vertebrate species, while also advocating for machine learning initiatives to enhance the effectiveness of biodiversity conservation programs. The dataset is derived from audio recordings collected as part of a collaborative PAM program in Brazil. These recordings, amounting to approximately 27 hours, were captured across four different biomes of Brazil, as depicted in Figure 6.3. After acquiring the recording, the data underwent annotation and pre-processing procedures. The annotation process comprised three steps.

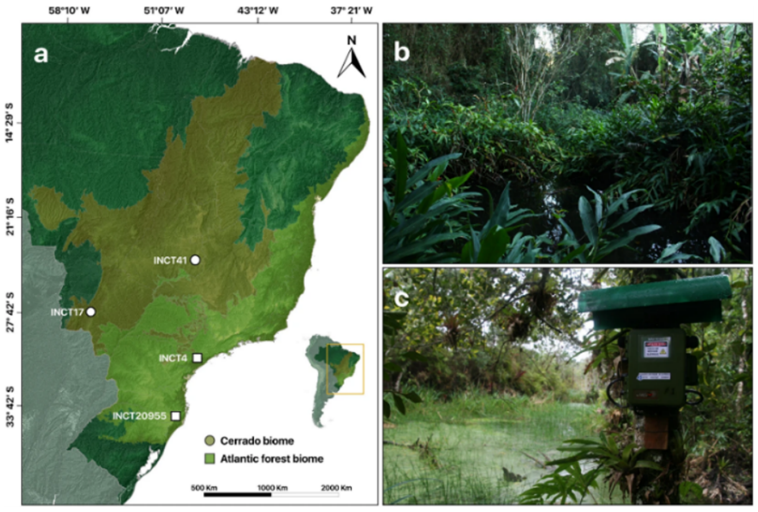

Figure 6.3: Geographical location where the data was collected to create the Anuraset. Source: [10]

First, audio sampling was conducted, wherein 1612 1-minute audios were chosen from four distinct sites shown in Figure 6.3. These audios were selected in quantities ranging from 50 to 100 per site, corresponding to three different seasonal periods (early, middle, and late) and three different daily periods (early, middle, and late night).

The second step involved weak labeling, where the labels were assigned based on the level of calling activity, ranging from absence to low, moderate, and high labels. This classification depended on factors such as signal-to-noise ratio, overlapping calls, and visualizability in the spectrogram.

Finally, the third step entailed strong labeling, which involved identifying the temporal boundaries of various anuran species that might be present in each sample. Both weak and strong labels were annotated by local herpetologists and experts in bioacoustics.

In the pre-processing step, the samples were divided into 3-second segments using a fixed length window. Each segment was labeled with multiple labels, indicating the presence of a species. This labeling process was done using the scikit-maad python package.

Next, the frequencies were limited to a range between 1Hz and 10000Hz. Then, a band-pass filter and Butterworth filter were applied to the audio signal. Finally, the signal was normalized to a maximum amplitude of 0.7 decibel full-scale. The processed audio was saved in an uncompressed WAV format.

The datasets are divided into four sets: INCT17, INCT20955, INCT41, and INCT4. These sets correspond to the sites shown in Figure 3 and contain 42.5%, 33%, 13.5%, and 11% of the annotated data, respectively. In total, information was collected from 42 anuran species, 12 genera, and 5 families, totaling 31 hours of audio duration. The chosen dataset is "INCT41", which comprises 7032 samples from 5 different species. The decision to use this dataset is based on its smaller number of classes compared to the dataset employed in Section 6.3.1. This enables the evaluation of how varying class numbers may impact performance and metrics. The distribution of the samples among the species is shown in Figure 6.4. This dataset has a more pronounced class imbalance compared

to the "created dataset" in Section 6.3.1. Specifically, the "PITAZU" class has significantly fewer occurrences compared to the "BOAALB" class. The impact of this imbalance will be observed in the results after training, and we will also explore whether data augmentation can help mitigate these cases.

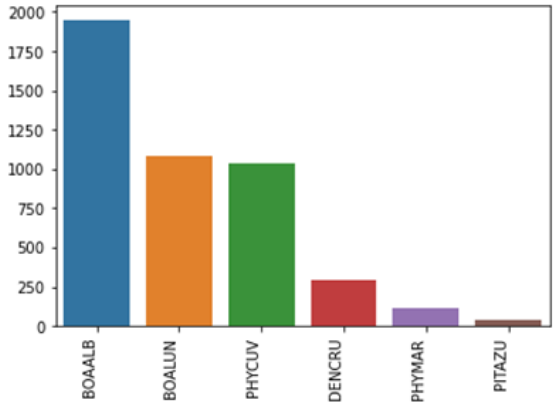

Figure 6.4: Distribution of INCT41 dataset

Similar to Section 6.3.1, Table 6.2 summarizes the dataset's organization. It consists of 14 columns, each with its own significance, which will be explained below.

- **sample_name:** This column corresponds to the name of the wav file of the sample.
- **fname:** It represents the name of the original 1-min audio from which the sample was extracted.
- **min_t** and **max_t:**denote the initial and final seconds of the sample in the original audio.
- **site:** This column contains the name of the site where the audios were extracted.
- **date:** It refers to the date when the sample was recorded.
- **species_number:** provides information about the number of species observed in the sample.
- **File_Path:** It provides the path to access the sample.

The columns that follow the "File_Path" header represent the occurrence of species in their respective column names. In this dataset, there are 5 species that may be present. These species are Physalaemus cuvieri (PHYCUV), Dendropsophus cruzi (DENCRU), Boana lundii (BOALUN), Boana albopunctata (BOAALB), Physalaemus marmoratus (PHYMAR), and Pithecopus azureus (PITAZU). In the dataset containing records from different sites, the last columns can extend up to 42, representing the diverse range of species present in the entire dataset. The data was filtered from the Anuraset archive, which included samples from four different sites.

Specifically, the information corresponding to the "INCT41" from the cite column was filtered. Additionally, some samples were filtered using the "min_t" and "max_t" columns, ensuring a difference of 3 second between them. This means that the next sample started at the second 1, rather than the 3rd second of the original audio. Filtering samples was decided to prevent overfitting and avoid using practically identical samples, using the "max_t" of previous samples as the "min_t" for the next sample. This helped introduce more variability and prevent a repetition of the same seconds.

sample_name	fname	min_t	max_t	site	date	species_number	File_Path	PHYCUV	DENCRU	BOALUN	BOAALB	PHYMAR	PITAZU
SAMPLE_27258.wav	INCT41_...	0	3	INCT41	2020-...	0	\Audios\...0_0_3.wav	0	0	0	0	0	0
SAMPLE_27261.wav	INCT41_...	3	6	INCT41	2020-...	0	\Audios\...0_3_6.wav	0	0	0	0	0	0
SAMPLE_27264.wav	INCT41_...	6	9	INCT41	2020-...	0	\Audios\...0_6_9.wav	0	0	0	0	0	0
SAMPLE_27267.wav	INCT41_...	9	12	INCT41	2020-...	0	\Audios\...0_9_12.wav	0	0	0	0	0	0
SAMPLE_27270.wav	INCT41_...	12	15	INCT41	2020-...	0	\Audios\...0_12_15.wav	0	0	0	0	0	0
SAMPLE_27273.wav	INCT41_...	15	18	INCT41	2020-...	0	\Audios\...0_15_18.wav	0	0	0	0	0	0
SAMPLE_27276.wav	INCT41_...	18	21	INCT41	2020-...	0	\Audios\...0_18_21.wav	0	0	0	0	0	0
SAMPLE_27279.wav	INCT41_...	21	24	INCT41	2020-...	0	\Audios\...0_21_24.wav	0	0	0	0	0	0
SAMPLE_27282.wav	INCT41_...	24	27	INCT41	2020-...	0	\Audios\...0_24_27.wav	0	0	0	0	0	0

Table 6.2: Information structure in the anuraset database

6.4 Pre-Processing.

Before applying the augmentations and giving it to the model, the samples should be subjected to some pre-processing to assure that all samples have certain characteristics to avoid error or certain difference that can damage model training. The preprocessing performed here is based on the preprocessing from Doshi [93]. This process is composed of three steps: channel conversion, resampling, and resize. These processes will be detailed in the following sections. Additionally, a description will be given on how the spectrograms were created.

6.4.1 Channel Conversion.

In order to avoid complications arising from differences in mono and stereo signals, all signals were normalized to a single type. This was done to simplify the generation of spectrograms, as it is easier to transform a mono signal and single channel spectrograms are more commonly used. Although this project includes the option to normalize to both 1 or a 2-channel signal, the implementation focused on the conversion methods proposed by Doshi [93] for the sake of preprocessing simplicity. To convert a mono signal to stereo, the signal was duplicated and concatenated, resulting in a 2D array (as depicted in Figure 6.5). Conversely, to convert a stereo signal to mono, only the first channel of the signal was selected (as illustrated in Figure 6.6).

6.4.2 Resampling.

If the audios were not pre-processed and were not recorded using the same device, it is possible that the sampling rates may differ. This could cause differences in the dimensions of the audio signals. Therefore, it is crucial to normalize all the data at the same sample rate. In this project, it was

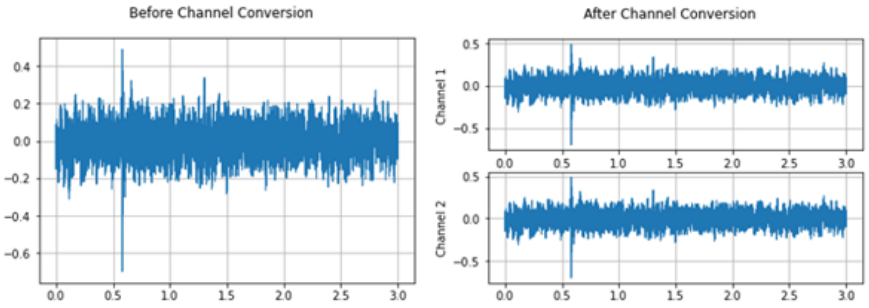

Figure 6.5: Mono to Stereo Conversion

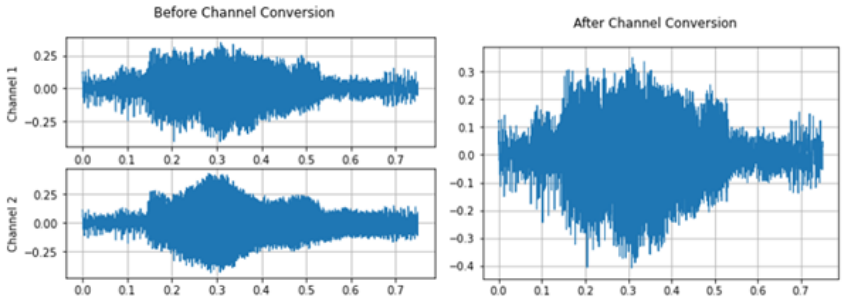

Figure 6.6: Stereo to Mono Conversion

achieved by utilizing the resampling function from Torchaudio. This function allows us to specify the resampling method to be used. Specifically, there are two methods available: the Hann window filter and the Kaiser window. By default, the Hann window filter is applied during the resampling process.

As explained by Chen and Hira [94], the resampling process involves creating a low-pass filter by computing the ideal filter coefficients based on the original and new sampling rates. The Hann window is then applied to shape the filter coefficients. Subsequently, the windowed filter is used in a polyphase filtering technique, which performs the interpolation (up-sampling) and decimation (down-sampling) to re-sample the audio signal.

For the Anuraset data, a sampling rate of 22050 was chosen. For the dataset created from mixed data, a sampling rate of 44100 was established.

6.4.3 Resizing.

To ensure standardized sample dimensions, the audio durations need to be equalized. This involves calculating the maximum length of the array. In order to do this, the first step is to multiply the audio sampling rate by the desired maximum duration in milliseconds. The algorithm then compares the length of the selected signal to the predefined maximum. If the signal is longer, it is

truncated to the desired length. Conversely, if the signal is shorter, the algorithm adds silence at the beginning and end of the signal (Figure 6.7). This is achieved by padding the signal with an array of zeros. For the multi-label dataset, a duration of 5 seconds has been established. As for the Anuraset samples, a maximum duration of 3 seconds has been set to match the existing dataset's duration.

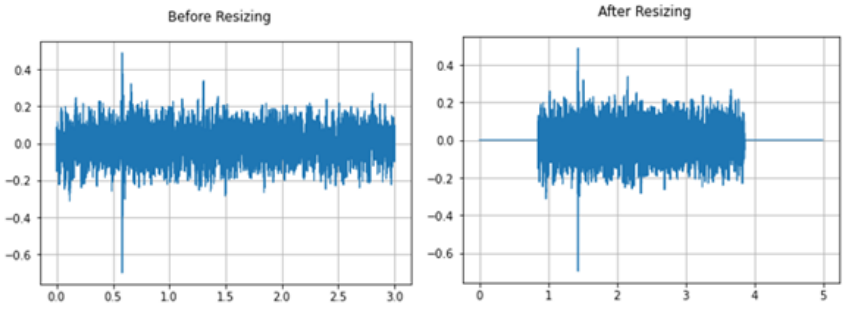

Figure 6.7: Example of resize illustration

6.4.4 Spectrogram generation.

Spectrograms play a crucial role in audio classification for machine learning. They provide more detailed information about the audio data, such as frequency, time, and amplitude, compared to using a waveform. This makes it easier for neural networks to identify specific characteristics. As a result, spectrograms are widely used in audio and deep learning research. They are an important part of the pre-processing stage, and their parameters will be explained in the following paragraph. Both the mixed multiclass data and Anuraset datasets used the mel-spectrogram function from Torchaudio. Amplitude visibility in the spectrogram's colors was enhanced using the decibel scale. This function performs a Fast Fourier Transform to generate the spectrogram and also applies mel-scale conversion, allowing for a closer examination of specific frequency ranges. In both datasets, a window size of 1024, 128 Mel filterbanks, and a range of 80 dB were used. This configuration results in the spectrogram that can be observed in Figure 6.8.

6.5 Data Augmentation Application.

One of the fundamental objectives of this undergraduate project is to test the effect of data augmentation on multi-label spectrogram classification. As previously stated in Section 5.2.2, this technique provides various benefits, including diversifying the data, aiding in the mitigation of imbalance, and increasing the sample size. Moreover, it has been observed in the data distribution of both the created dataset and Anuraset that there is a noticeable imbalance among certain classes. Therefore, the augmentation process was divided into two phases. In Section 6.5.1, an algorithm will be presented, which was implemented to address data imbalance. Additionally, Section 6.5.2

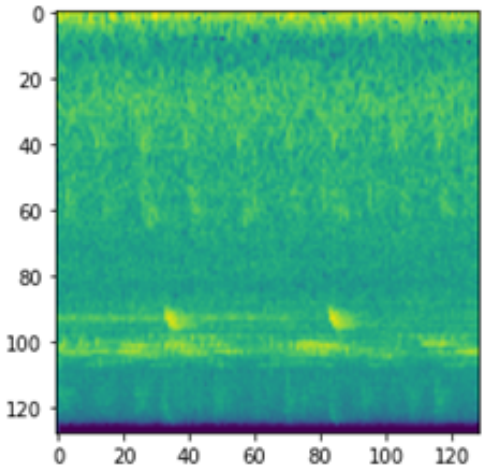

Figure 6.8: Example of the spectrograms generated

will delve into the process of selecting and applying data augmentation techniques.

6.5.1 Data Imbalance Mitigation.

As mentioned, one aim is to address data imbalance with data augmentation. This will be done by increasing the number of samples with fewer occurrences to increase their detectability by the model. The increment will be done by considering the level of imbalance. The computation of this level involved the development of the algorithm illustrated in Figure 6.9, which relied on the data provided in Imb [95]. First, it was calculated the proportion of the minority class regarding the number of samples. With this percentage, it was defined 4 ranges that define the type of imbalance:

- Over 50% (0): No imbalance.
- From 40% to 49% (1): Mildly imbalanced.
- From 20% to 40% (2): Highly imbalanced.
- Less than 20% (3): Extremely Imbalanced.

The imbalance factor per class was then calculated by dividing the total number of samples by the number of samples in each class. Additionally, for multi-label datasets, an extra factor is considered. It's crucial to acknowledge that dealing with imbalances in multi-label datasets is distinctively different from addressing it in multiclass data. It's important to keep in mind that there may be instances belonging to the least frequent category that are also found within the category with higher instances. Hence, it is important to exercise caution while increasing samples for the low occurrence category to avoid any impact on the high occurrence category. The level of difficulty depends on the extent to which samples are shared between the classes. Consequently, the

algorithm takes into consideration the classes represented in the sample and determines the number of augmentations to apply accordingly.

Before finding this number, two vectors are generated: augmentation factor and extra augmentation factors. The augmentation factor is calculated by dividing the total number of samples by the number of samples per class. The extra augmentation factor is generated by creating a vector that contains a sequence from 0 to N-1, where N represents the number of classes. This vector is then multiplied by the type of imbalance (0, 1, 2, or 3). Each array element is assigned to a class based on sample quantity, with high values for classes with few samples and low values for the ones with more samples, in order.

Upon obtaining the vectors of augmentation factors and the additional augmentation factor, the calculation to determine the number of augmentations to apply is conducted. This number is determined by considering the classes present in the sample. The augmentation factors and extra factors are selected from a dictionary based on these classes. If all the classes in the sample are below the mean of the dataset, it is used the sum of the maximum values between the vectors of augmentation factors and extra augmentation factors.

Conversely, if a label in the sample exceeds the dataset mean, the minimum value from the augmentation factors vector is added to the division between the maximum factor of the extra augmentation factors and the number of elements inside the extra augmentation factors vector. Additionally, it is important to mention that the algorithm has an input that determines whether the dataset is multi-label or multiclass. Based on this information, an extra augmentation factor is used for multi-label datasets. However, for multiclass datasets, only the augmentation factor is needed to balance the dataset.

There is an additional condition for the class: it cannot exceed 100 augmentations. As explained in Section 6.5.2, certain augmentations can be used multiple times due to a random factor involved. By utilizing these augmentations, it is possible to generate samples up to 100 with a low chance of repetition. However, exceeding this limit could increase the likelihood of repeated samples and consequently, the risk of overfitting the model. The outcomes of implementing this algorithm and the challenges it poses will be observed and discussed in the Results.

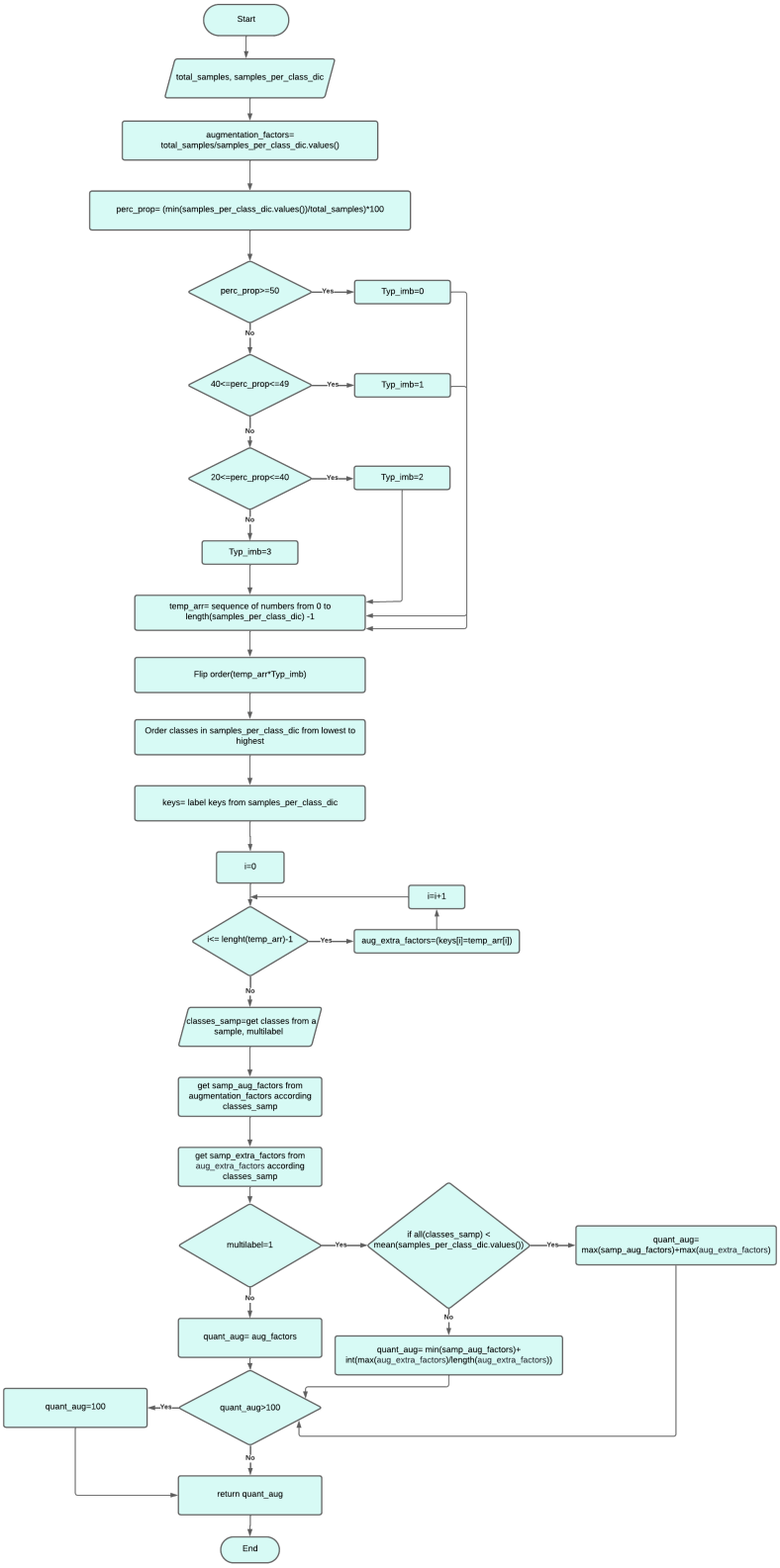

Figure 6.9: Data Imbalance Mitigation Algorithm

6.5.2 Augmentation Application.

The augmentation techniques were selected according to several of the sources consulted: Nanni et al. [7], Lucas Ferreira-Paiva et al. [9], Cañas et al. [10], Sun et al. [8] and other sources exposed in Section 5.3. In some of them, it was mostly used just one type of augmentation, such as the Time Shifting augmentation, applied by Dufourq et al. [85]. Others works, such as the case of Nanni et al. [7] and Lucas Ferreira-Paiva et al. [9], exposed several augmentation techniques that are focused not only on spectrogram augmentation like the ones presented in Park et al. [47]. These augmentations included several audio techniques and even techniques that are based on image augmentations. They were implemented and further explained in Section 5.2.2.

There are 24 augmentation techniques chosen from consulted sources.Park et al. [47], Wen et al. [96], and Mumuni and Mumuni [97] have shown that data augmentation has advantages, such as improving feature extraction in the model. Applying different transformations introduces diversity into the data, helping the model learn more varied and representative features in the spectrogram. Techniques like Time Stretching, Noise Injection, and Dynamic Range Compression were selected to improve the task. On the other hand, techniques like Sound Mix, Equalized Mixture Data augmentation, and Rolling can enhance classification in the model. These techniques generate data that reflects real-world scenarios, helping the model accurately identify certain classes despite variations and distortions in the data. It's important to note, as advised by Nanni et al. [7] and Lucas Ferreira-Paiva et al. [9], that the augmentation should not significantly change the semantic data of the spectrogram, as it could harm the model's performance. Techniques like flipping the spectrogram upwards, which alter the frequency information and confuse the neural network, were discarded. The objective of augmentations is to improve the model's ability to handle data variations, resulting in better generalization with unseen data and ultimately enhancing model performance and increasing model metrics.

Besides the aforementioned augmentations, mixed augmentations are also put forth as a proposal. These augmentations were implemented based on the works by Kumar et al. [98] and Salamon et al. [44], which mixed some augmentations techniques to increase the quantity of synthetic data. This work implemented a similar approach to increase augmentations and diversify data, while avoiding repetition of techniques. It is crucial to acknowledge that the potential combinations of augmentations are countless, and certain combinations may not enhance the augmentation's quality or the model's performance.

Some combinations serve as an experimental setup to assess their impact on the neural network. Therefore, the number of combinations was limited to 5. The augmentations were selected according to the type of variation added to the data and its utility for increasing model performance. As a result, it was opted for techniques centered on spectrograms and techniques that leverage audio variations. The assessment included the quality of the variations and the synergy of the techniques. Moreover, the assessment also factored in its capacity to emulate effectively a plausible situation within a real-world environment. Based on these considerations, the following combinations were determined:

- Echo and Sound Mix: adds an extra noise such as rind or wind and echoes as it would be in

an echoed habitat.

- Rolling and Dynamic Range Compression: In an attempt to replicate another sample, time axis is moved, and the volume of the highest sounds is diminished, while amplifying the volume of the lowest sounds.
- Shuffling and Mixing and Noise Injection: By blending two samples from the same class, partitioning the resulting sample into various segments, and carefully rearranging their sequence, a new sample is generated. Furthermore, the incorporates white noise to generate a distortion, as in a recording.
- Time Masking and VTLP: The inclusion of two augmentations in the spectrogram serves the purpose of introducing diverse features and imposing a challenging scenario for the model, intending to boost its performance.
- Time Swapping, rolling and "wow" resampling: Adds a spectrogram augmentation with the moving the signal in time axis and adding a distortion in the audio's speed.

Finally, with the mixed augmentations, there are 29 techniques to apply. Based on the information in Table 6.3 and Table 6.4, the techniques were ordered, with certain criteria established to determine the most useful augmentation techniques. The selected criteria focus on the technique's capacity to emulate a plausible context for data in a real environment, its extent of frequency axis modification, its effectiveness in improving data visualization, and its impact on classification performance. To measure the contribution, it was conducted some multiclass classification and evaluated the impact of augmentations on model performance. Experiment results are documented in Chapter 10, which also presents the groups where augmentations were tested. To accelerate the testing process, the augmentations were organized into 8 groups of three, with the objective of pairing at least one spectrogram augmentation with audio augmentations. The grouping was determined by the perceived benefits for the model. Finally, it must be clarified that some weights were assigned to each criterion based on their significance and impact on enhancing the model's performance.

	Weights	AWGN	Clipping	Harmonic distortion	Sound shuffle	DRC	Wow resampling	Delay	Echo	Flip	Rolling	VTLP	Sound Mix	Time stretch	Pitch Shift	Shuffled mixed
Emulates the a possible case related to the context of the classification	4	4	3	2	3	4	3	3	4	2	4	3	5	3	3	3
Modifies the frequency axis	3	4	3	2	4	4	3	5	4	3	5	3	2	4	3	4
Improves data visualization	3	3	3	4	3	4	3	3	3	3	3	4	2	3	3	3
Contribution to classification (after results)	4	2	3	1	2	3	3	2	3	1	2	2	2	3	2	3
Total		45	42	30	41	52	42	44	49	30	48	41	40	45	38	45

Table 6.3: Criteria and rating for audio augmentations.

Given this information, and also giving more priority to audio augmentations, the order established is: Echo Effect, Dynamic Range Compression, Time Stretch, Shuffling and Mixing, Clipping, "Wow" resampling, Rolling, Shuffling, VTLP, Noise Injection, Delay, Pitch Shifting, Sound Mix, Harmonic Distortion, Flip, Brightness and Saturation, Color Reduction 32, Green Filter, Time Masking, Time Swapping, EMDA, Negative Positive, Color Reduction 128, Red Filter. After the

	Weights	Time Masking	Time swapping	EMDA	Bright and Saturation	Negative Positive	Color Reduction 32	Color Reduction 128	Red filter	Green filter
Emulates the a possible case related to the context of the classification	4	3	4	3	3	2	2	2	2	2
Modifies the frequency axis	3	4	4	3	4	4	4	4	3	3
Improves data visualization	3	3	3	3	3	4	4	2	2	3
Contribution to classification (after results)	4	2	2	2	3	2	3	2	1	3
Total		41	45	38	45	40	44	34	27	38

Table 6.4: Criteria and rating for spectrogram augmentations.

single augmentation, which mean the class of the label needs more than 24 augmentations, are implemented in the next order: Sound Mix and Echo Effect, Rolling and Dynamic Range Compression, Shuffling and Mixing and Noise, Time Masking and VTLP, and Rolling with "Wow" Resampling and Time Swapping.

In case augmentations exceed 29, another measure was implemented to balance the dataset. This measure takes advantage of a "random factor" that has some techniques. The random factor refers to certain parameters that are randomly adjusted in these techniques, resulting in a unique sample every time the augmentation function is called. For example, using the rolling technique, it is possible to create another sample by moving the spectrogram by a different factor. Therefore, the next augmentations are used to be repeated and produce more samples giving better changes to balance the dataset: Echo Effect, Dynamic Range Compression, Time Stretch, Shuffling and Mixing, Clipping, "Wow" resampling, Rolling, Shuffling, VTLP, Noise Injection, Delay, Pitch Shifting, Sound Mix, Time Masking, Time Swapping, EMDA, Negative Positive, Color Reduction 128, Red Filter, Sound Mix and Echo Effect, Rolling and Dynamic Range Compression, Shuffling and Mixing and Noise, Time Masking and VTLP, and Rolling with "Wow" Resampling and Time Swapping.

With all the functionalities explained, the function assigns the augmentations according to the number returned from the function explained in Section 6.5.1. The function returns an array of elements from a class. This class contains the information of the signal augmented, the technique of augmentation applied, the type of signal returned (spectrogram or audio) and the number of repeated augmentations. This number is used if over 29 augmentations are required. It aids in naming the files for the additional augmentations and prevents file overwriting.

With this information, augmented samples are added to the data frame, including a new column describing the applied augmentations. Meanwhile, the signal type plays a role in selecting the signals that should be converted into spectrograms prior to their storage as images. The files are labeled by incorporating the name of the original file, the type of augmentations, and the repeated count of augmentations. Ultimately, the function stores all the augmented samples in a pre-defined folder and returns the resulting dataframe with the augmented samples' information, which will be used for model training.

6.6 Model selection and configuration.

To test the effectiveness of transfer learning, three different neural network architectures were selected. These architectures were selected from a wide range of available models based on their fre-

quency of use in the investigated sources, particularly those focused on animal audio classification, and their achieved results in these investigations. The objective is to examine the performance of the selected CNNs and how effectively it will classify multi-label spectrograms. During the training, it was important to configure appropriately each model to obtain the best possible results. Therefore, the next considerations were taken for the models to obtain the final model's configuration and start the training.

The model's underlying framework involved a basic model architecture with frozen layers, enabling the utilization of pre-trained weights for spectrogram classification. These weights were acquired from the model's pre-training on the ImageNet dataset. Trainable parameter was set to false in all models used for the experiment. It is also important to clarify that in all models, the top layer is not included, instead it implemented a custom layer where its number of units is equal to the number of classes to classify. Furthermore, the final layer of each model incorporates the Sigmoid activation function (Section 5.2.3.3.6.2), which proves valuable for one hot encoded labels, such as the labels employed in this investigation. This allows for the calculation of independent probabilities for each label within the vector.

After implementing the final classification layer, the compile function from Keras must be used to configure the model for training. Certain hyperparameters must also be configured for this function. The parameters described next are equal for all the models. To begin the implementation, the Adam optimizer (Section 5.2.3.3.7.8) was introduced, following the example of models used in comparable tasks, such as the one proposed by Sun et al. [8]. The decision to adopt this optimizer was driven by its rapid convergence and dependable training process, which are advantageous when dealing with imbalanced datasets. Within this parameter, the learning rate must also be established, which will be adjusted by observing the behavior of the model during training. Subsequently, the metrics and loss function were configured. The metrics chosen are "binary cross entropy" and "binary accuracy". These metrics, like the sigmoid activation, determine their metric value by comparing the probabilities of each value in the label vector.

Additional configurations are performed after test trainings, depending on the model's behavior. Thus, the basic model with the output layer may not always exhibit optimal performance. Therefore, some additional layers or techniques that must be implemented to increment the performance of the model, which implies incrementing the metrics or the fitting between the validation and training curves. Specifically, fully connected layers, regularizations, batch normalization, and pooling methods were incorporated. The underlying theory of these methods was explained in Section 5.2.3.1.1.3, Section 5.2.3.1.1.4, Section 5.2.3.1.1.7, and Section 5.2.3.1.1.2, respectively. Subsequent subsections will detail how the selected models employ these techniques.

Following a discussion on the overall considerations applied to each model, the subsequent section will summarize the configuration for each model. This will include a concise explanation of its architecture and the reason behind its selection. For a more comprehensive understanding of the model architecture, additional details can be found in Section 5.2.3.1.2.1, Section 5.2.3.1.2.2, and Section 5.2.3.1.2.3, where the architectures are extensively elaborated.

6.6.1 ResNet 50.

As explained in Section 5.2.3.1.2.1, ResNet is characterized by a neural network structure that is based on the principle of residual learning. This technique effectively mitigates vanishing gradients, resulting in faster training and a reduction in training error. Therefore, is one of the most powerful and effective models for image classification, as it is vastly used in several classification problems with outstanding results. This observation is also supported by the sources consulted [85][88][86][89][10]. That is the underlying justification for its inclusion in the current investigation. Although this model has some limits, which do not make it suitable or the best in performance in certain datasets. These limitations stem from the model's high complexity compared to other models, making it susceptible to overfitting. This complexity necessitates higher memory and computational requirements.

In Table 6.5, it is possible to observe the configuration used for this investigation with at the end showing the total of parameters also describing the non-trainable and trainable parameters for the model. Alongside the base model, a global average pooling operation was introduced to minimize the model's parameter count and mitigate overfitting. Then a batch normalization layer was included to reduce the internal covariate shift and accelerate training by also adding some regularization to the model. Finally, a dense layer was added, which was followed by a ReLU activation function and an L2 regularizer was applied. The purpose of the dense layer is to process and enhance the features that were extracted, while the regularizer helps prevent overfitting. Among the three, this model possesses the highest number of parameters.

Layer	Ouput Shape	Number Parameters	Additional information
Input layer	224x224x4	0	
Base model: ResNet50	7x7x2048	23587712	
Global Average Pooling	2048	0	
Batch Normalization	2048	8192	
Dense Layer	64	131136	ReLU was used as activation function. L2 Regularization applied with value 0.001
Output Layer	11	715	
Total Parameters:		23,727,755	
Trainable Parameters:		135,947	
Non-trainable Parameters:		23,591,808	

Table 6.5: Additional architecture for Resnet model.

6.6.2 VGG 19.

As described in Section 5.2.3.1.2.2, it was created for image classification tasks with deep neural networks. The project aimed to incorporate deeper neural networks while ensuring optimal performance, as reflected in its architecture predominantly consisting of convolutional, pooling, and fully connected layers, and the utilization of small kernels (3×3). Notable features of this model include high accuracy metrics, strong generalization, detailed feature extraction, and relatively simpler architectures than those found in other models. Therefore, it is also an architecture that has been

used numerous times in several image classification investigations and specially used in animal sound classification with augmentations as it was used in the works from Sun et al. [8] and Nanni et al. [7]. Consequently, this architecture has found significant application in many image classification investigations, particularly in the realm of animal sound classification, incorporating augmentations akin to those employed in the studies conducted by Sun et al. [8] and Nanni et al. [7]. Nevertheless, like ResNet, the model's complexity can make it susceptible to overfitting, demanding extensive memory and computational resources. Moreover, it may prove less efficient in specific scenarios compared to contemporary architectures like ResNet.

Such as the case of Resnet, Table 6.6 shows the configuration implemented for this model. For the model it was implemented a global average polling layer to reduce the dimensionality and the number of parameters in the model. After this layer, there are two sets of dense layers followed by batch normalization functions. These layers enhance the model's complexity and ability to learn. Batch normalization ensures that the training process remains efficient and stable. Lastly, a flatten function is applied to prepare the feature map for the output classification layer. In terms of parameters, this model is positioned in the middle, with a number that is near, but not greater than, the number from ResNet.

Layer	Ouput Shape	Number Parameters	Additional information
Input layer	224x224x4	0	
Base model: VGG19	7x7x512	20024384	
Global Average Pooling	512	0	
Dense Layer	128	65664	ReLU was used as activation function
Batch Normalization	128	512	
Dense Layer	256	33024	ReLU was used as activation function
Batch Normalization	256	1024	
Flatten	256	0	
Output Layer	11	2827	
Total Parameters:		20,127,435	
Trainable Parameters:		102,283	
Non-trainable Parameters:		20,025,152	

Table 6.6: Additional architecture for VGG model.

6.6.3 EfficientNetB0.

Explained further in Section 5.2.3.1.2.3, the objective of developing this model is to enhance performance by employing a technique that uniformly scales three dimensions (width, depth, and resolution) through the computation of a predetermined set of coefficients. This network architecture has been proof of achieving high accuracy and metrics performance while being effective and fast to train. Its adaptive design enables it to be used in a significant variety of applications. According to several consulted sources [89] [88], EfficienNet has showed great utility in spectrogram classification, surpassing other models like ResNet in terms of accuracy. It is worth noting that

this model differs from the others because it includes a rescale layer, rendering a separate rescaling operation unnecessary. Thus, the reason for selecting it for testing in this investigation is that it has fewer parameters compared to the models presented in this study, yet shows promising performance. While in certain scenarios, particularly when dealing with extensive and intricate datasets, there might be some limitations in its ability to capture complex patterns and it may not be as resilient to noisy data. Even though it is relatively simple, it remains vulnerable to overfitting.

Similarly to the previous models, in Table 6.7 are presented the configuration used in for the datasets used in this investigation and that got the best results. The configurations of this model slightly differ from the tested datasets, unlike the other models. The intention was to achieve optimal performance in each dataset, by observing the training and validation curves after training. With the Anuraset dataset, a fully connected layer with followed by flatten layers was integrated to amplify the model's complexity and learning capability, while ensuring efficient training, preventing overfitting and preparing the data for the output layer. In the case of the dataset that was generated, a l2 regularization was applied to control signals of overfitting. Among the available models, this one stands out for having the fewest parameters, significantly fewer than ResNet and VGG.

Layer	Ouput Shape	Number Parameters	Additional information
Input layer	224x224x4	0	
Base model: EfficientNetB0	7x7x1280	4049571	
Dense Layer	7x7x128	163968	ReLU was used as activation function. When trained with the mixed samples dataset, L2 Regularization applied with value 0.001
Flatten	6272	0	
Output Layer	11	69003	
Total Parameters:		4,282,542	
Trainable Parameters:		232,971	
Non-trainable Parameters:		4,049,571	

Table 6.7: Additional architecture for EfficientNetB0 model (Anuraset).

6.7 Training process and experimental design.

The pipeline of the experiments is further detailed in Figure 6.10. The initial step was data collection, outlined in Section 6.3. This involved two datasets: Anuraset and a dataset created from mixed samples. The latter dataset required an algorithm to produce multi-label samples. Then, the samples are submitted to pre-processing (Section 6.4) where the samples are normalized to avoid incompatibilities between the model, the algorithms, and the data. During this phase, the spectrograms are also generated.

The subsequent phase involves data augmentation, which was discussed in Section 6.5. With offline augmentation (Section 5.2.2.3.2), the augmentations are applied prior to using the data for

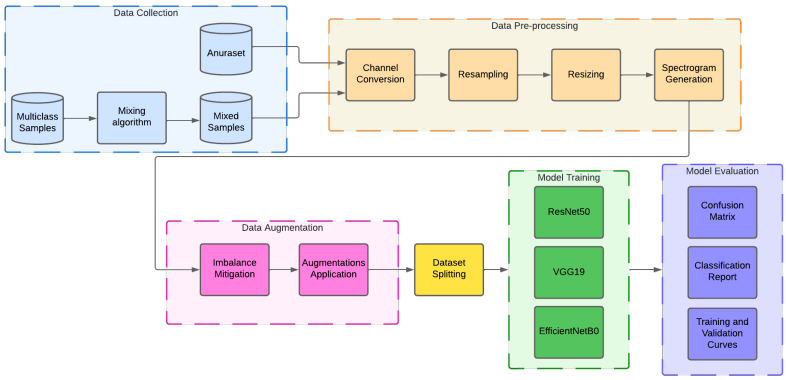

Figure 6.10: Block diagram of the system

training in the model. The samples were stored in a drive and later merged with the sample information file to form the final database, which was then used to train the model. This technique allowed for assessing the quality of the augmentations and using an algorithm to address the imbalance or enhance the samples from the class with fewer occurrences. The dataset was prepared to begin the training and validation process following this procedure. The set of images was created using the "ImageDataGenerator" function provided by Keras. This function uses an efficient algorithm to generate batches of images, using file paths to access the samples and labels required for training. Additionally, it necessitates specifying the batch size and target size of the images, which in this case were set to 224×224 pixels.

Initially, the dataset is divided into training and validation sets, and then provided to the model. This division is done using the Stratified KFold Cross-Validation Algorithm, which was explained in Section 5.2.3.4.1. This algorithm splits the dataset into two partitions, with varying partitions determined by user-specified iterations. The goal of the stratification cross-validation algorithm is to assign label combinations in each fold based on the number of folds specified and the available data for each label. In order to evaluate effectively the training and validation sets, 5 folds have been chosen for this investigation. This decision takes into consideration the potential training time required and ensures that the training will be assessed across a wide range of training and validation sets. The cross-validation algorithm is used to assess the model's performance across different folds and ensure its ability to generalize effectively. This entails analyzing the accuracy and loss curves in each fold for indications of appropriate learning. If any of the folds exhibit noisy behavior, overfitting, or underfitting signals, it is necessary to adjust certain hyperparameters and the model's architecture in order to achieve optimal performance.

The model configuration was configured with 50 epochs, which is considered a good initial hyperparameter setting, as explained by Goodfellow lan [56], and also has been used in several investigations. Once the training and validation curves are obtained, it becomes possible to determine whether it is necessary to increase the number of epochs for the model. This can be determined by observing if the curve cannot stabilize at a specific value after the training and instead continue

to exhibit an ascending behavior. Another important hyperparameter for training performance is the batch size, which has been previously established. As recommended by Géron [54], a batch size of 32 was chosen as it provides sufficient information for the model to train quickly and achieve convergence. However, if the model shows signs of overfitting, low accuracy, or slow convergence, the batch size can be adjusted to mitigate these effects.

After observing the behavior of the training and validation curves, it is possible to perform additional hyperparameter tuning. This includes adjusting the learning rate, incorporating regularization techniques, or adding extra layers, such as fully connected layers, pooling layers, or batch normalization layers. In this case, a learning rate of 0.00001 was set for all models. This value yielded loss curves that did not exhibit linear or increasing behavior, suggesting overfitting in the model. Regarding the model configurations, Section 6.6.1, Section 6.6.2, and Section 6.6.3 present the final setups for ResNet50, VGG19, and EfficientNetB0, respectively.

Prior to the training, a few call-backs were introduced. Two of them were custom call-backs, one for tracking epoch training time and another for monitoring GPU memory usage. These call-backs enabled the calculation of total training time and average memory usage. Additionally, a callback obtained from Keras called "Model Checkpoint" was used to save the weights based on specific performance metrics, in this case, the validation loss.

There are four sets of experiments proposed to check the effectiveness of data augmentation and transfer learning. The first consists in test the model with both data augmentation and transfer learning applied. The next two proposed experiments were to use one of these techniques for the model: transfer learning or data augmentation. Finally, the model was tested without transfer learning and data augmentation. The intention is to show the individual effects or data augmentation and transfer learning. The performance will be assessed using the evaluation metrics detailed in Section 6.8. This will facilitate comparison of results across experiments and model performances.

6.8 Model evaluation.

As suggested in Section 6.7, there are certain characteristics that the curve must have to be considered the learning curve as an appropriate learning. As exposed by CS2 [99], the accuracy curve must increase its value during training and find certain stability, meanwhile the loss curve must decrease and also flatten at a certain point. The gap between the training and validation curves should be minimal. A gap between the curves could suggest an overfitting presence in the model. While not guaranteeing perfect learning, these curves could be an indicator overfitting or underfitting in the model. Therefore, there are other metrics that help to measure and understand the behavior model more deeply.

To evaluate the model's performance and specifically measure its performance for each class, it was used the classification report and confusion matrices. The classification matrix (Section 5.2.3.4.7) provides insights into the number of samples correctly and incorrectly classified, represented by False Positives, False Negatives, True Positives, and True Negatives. These values in the confusion matrix are used to calculate the metrics presented in the classification report, which includes the next range of metrics:

- Accuracy: As detailed in Section 7.8, accuracy is defined as the proportion of correct predictions relative to the total number of evaluated samples. During the experiments, accuracy was computed for each label, after which the mean was calculated to derive the average accuracy per dataset. Subsequently, these values were averaged to determine the overall accuracy for each model.
- Precision: Expanded in Section 5.2.3.4.6, it helps to measure how many of the predicted positive predictions were correct, measuring how exact is the model to detect positive samples for each class.
- Recall: This metric, mentioned in Section 5.2.3.4.5, helps evaluate the model's capacity to detect positive samples accurately for each class.
- F1-score: It is considered the harmonic mean between precision and recall (Section 5.2.3.4.4) and helps to measure the proportion between precision and recall, providing a balance evaluation of the model's performance.
- Support: shows how many samples are in each class.
- Micro Average: This metric is calculated by considering all the True Negatives, True Positives, False Positives and False Negatives for each class. These values are summed to calculate the micro average for each metric.
- Macro Average: It is the average of each metric (Precision, Recall and F1-score) over all classes.
- Weighted Average: Consists in the average of each metric weighted by the support of each class.
- Samples Average: Calculates the three metrics (Precision, Recall and F1-score) for each instance and the averages them.

This metric enables the possibility to assess the performance of each class, providing a deeper understanding of how the model is performing for each individual class. It allows us to focus our attention on classes with low metrics, investigating the underlying reasons for these values. Alternatively, it could confirm that the model can exhibit exceptional metrics regardless of the class.

In addition to classification metrics, other metrics were implemented to check different characteristics of the model. Callbacks were used to monitor training time, GPU memory usage, and model convergence. The results of loss and accuracy curves were stored and converted to a logarithmic scale to compare the speed of convergence between models. This allowed for a comparison of classification metrics, training time, memory usage, and convergence information, which could be useful for applications with limited computational requirements.

Also, it is important to note the cross validation was another technique used to evaluate the performance of the model. With cross-validation, it was possible to monitor the behavior of the

model during the 5 folds of training. Thus, it is possible to confirm that there are no indications of over-fitting in the training folds, and the training results were not coincidental but rather independent of the data used to train and validate the model. In summary, this method shows the robustness and generalization of the model across different data samples provided by the stratified cross-validation algorithm for multi-label datasets.

Chapter 7

Results and Discussion

In the previous chapter were described the theatrical framework and the methods used in this investigation. The results from applying these concepts, materials, and methods are presented in this chapter. The aim was to test the effects of data augmentation and transfer learning in spectrogram classification, specifically in anuran species classification. These results are divided into two sections.

The results of the augmentations discussed in Section 5.2.2 will be presented in Section 7.1, demonstrating how each variation affects the spectrogram signal. Additionally, it will depict the impact of the imbalance mitigation algorithm proposed in Section 6.5.1 and discuss specific cases where the algorithm performs better.

After, in Section 7.2, will be presented the results of the experiments described in Section 6.7. Each subsection corresponds to a specific case, evaluating the application of data augmentation and transfer learning, implementing either technique, and the model's response in the absence of both. These were tested in both the mixed samples dataset and for Anuraset. The objective was to assess technique effectiveness in a dataset with various classes, compared to a dataset with fewer classes.

7.1 Data Augmentation.

This section will present the results got after applying augmentations to the presented datasets. It will discuss the effects of implementing audio and spectrogram augmentations on a sample and provide an illustration of these effects. Subsequently, it will present a comprehensive analysis of the results achieved through the utilization of the algorithm specifically developed to tackle the imbalance in multi-label datasets.

7.1.1 Application of Augmentations.

As it was previously stated in Section 5.2.2 and Section 6.5.2, 27 different augmentations applied to signal previous the training. These augmentations are based on spectrogram and audio augmentations consulted in similar investigations to diversify the quantity of variations and increase the number of samples. Hence, by diversifying variations and increasing the number of samples, the aim is to enhance the model's ability to generalize by exposing it to challenging scenarios and improving its capability to identify rare samples. Consequently, the model will effectively classify noisy data and be better equipped to categorize unseen data. However, when applying augmentations to the signal, it is important to set a limit on the extent to which they distort the signal. It is crucial to consider that the augmentation should not alter the frequency context of the sample significantly.

Therefore, it is necessary to strike a balance in the signal variations. These distortions should not be too subtle, as they would essentially produce the same sample, but also not too drastic, as they would completely change the semantic content of the sample and confuse the mode.

In Figure 7.1 is illustrated the effect of different audio augmentations on a signal. These audio augmentations were reunited considering that the functions need the audio signal as input. The objective also was to observe how these distortions affect the spectrogram, though applied to the audio signal. As it is observed in Figure 1, some of them comprise adding distortions to the signal. The type of distortion depends on the augmentation.

Distortions were applied to the pitch and time in the audio, similar to the ones found in the Scaper library. These distortions affect the time axis by either expanding or compressing its components. Meanwhile, pitch shifting slightly moves the components along the frequency axis.

Other types of distortions involve dividing the audio into sections and rearranging them, altering the appearance of the spectrogram on the time axis. Dynamic range compression was implemented, directly affecting the signal's amplitude by adjusting the volume of the sounds in the sample. The application of these augmentations results in changes to the intensity of specific components in the spectrogram, and even affects the overall energy. Another set of distortions that can affect the energy in the spectrogram includes noise injection, sound mixing, harmonic distortion, and clipping. These manipulate the components within the spectrogram by adding, limiting, or complicating its structure.

Lastly, there are augmentations that change the signal without affecting its frequency components. These include techniques like rolling, delay, and flipping, which primarily manipulate the time axis by displacing or inverting its components.

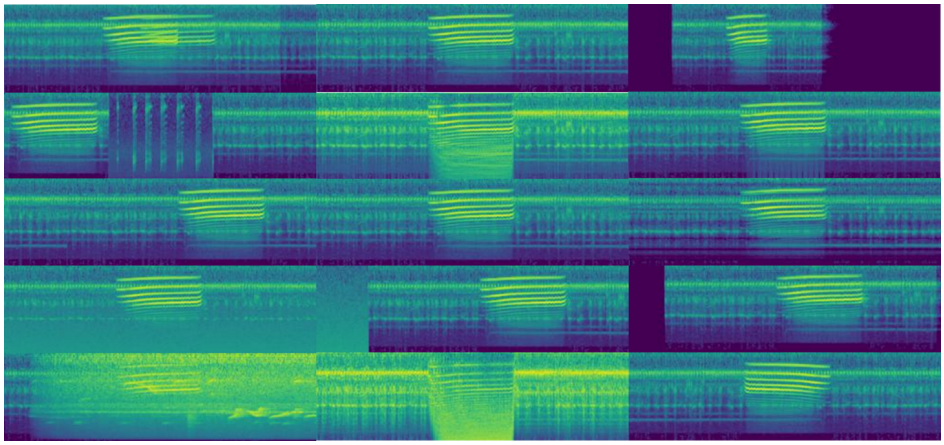

Figure 7.1: Audio Augmentation techniques. Augmentation order(from left to right): Echo Effect, Dynamic Range Compression, Time Stretch, Shuffling and Mixing, Clipping, Wow Resampling, Rolling, Shuffling, VTLP, White Noise Injection, Delay, Pitch Shift, Sound Mix, Harmonic Distortion, and Flip.

Figure 7.2 depicts the impact of spectrogram augmentation on a sample. The main aim of

spectrogram augmentation is to change the visual attributes of the spectrogram. For instance, some augmentations focus on obscuring a portion of the spectrogram's visual field, like time masking. Other augmentations involve applying specific filters to the spectrogram image, thus emphasizing particular visual features. EMDA stands out in this category of spectrograms, as it involves mixing two signals and applying an equalizing function. This process introduces an additional frequency component by combining another sample from the same class. As a result, the resulting spectrogram exhibits increased blurriness and greater variability in its characteristics.

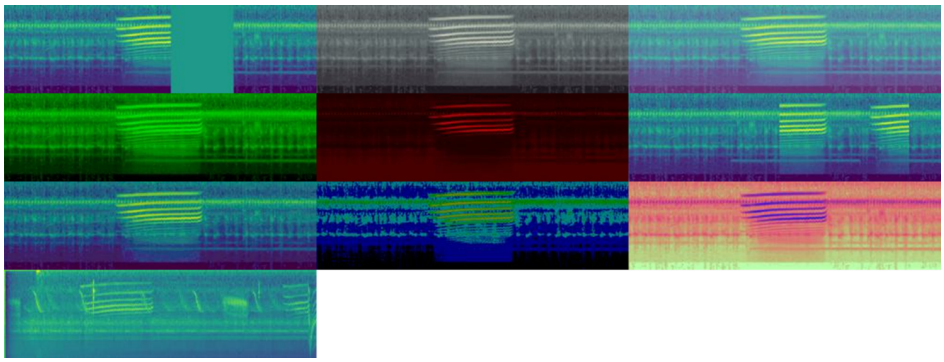

Figure 7.2: Spectrogram augmentation techniques. Augmentation order(from left to right): Time Masking, Saturation, Brightness, Green Filter, Red Filter, Time Swapping, Color Reduction 32, Color Reduction 128, Positive Negative and EMDA.

7.1.2 Case mixed samples.

In Figure 7.3 is represented the results of the application of augmentations to the dataset with mixed samples. Compared to Figure 6.2, the level imbalance in this dataset has decreased. This is noticeable in the "Ameerega Picta" class, where the difference between this class and the one with the most samples is not as significant as seen in Figure 6.2. Table 7.1 provides a more precise presentation of the number of samples and how the sample count increased after augmentation. As Figure 7.3 suggested, the sample count has significantly increased, and the disparity between them has slightly leveled out. Therefore, the algorithm that mitigates imbalance has fulfilled its aim for this dataset. This could be advantageous for the model, allowing it to generalize well and accurately identify any of the classes within the dataset. With 99761 samples, the final dataset showcases a notable expansion compared to the initial 7044.

7.1.3 Case Anuraset.

Using the algorithm that mitigates imbalance gives a different result for Anuraset. As it is possible to see in Figure 7.4, for this dataset, the imbalance was not completely mitigated or considerable improved. The difference between the class with the fewest samples and the class with most samples is still very large. Although compared to the dataset shown in Figure 6.4 there is an increment in the samples from the rarest classes. Moreover, this increment is verified in Table 7.2, illustrating

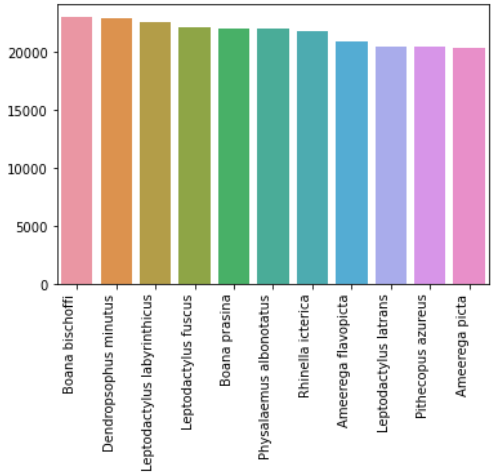

Figure 7.3: Distribution of the mixed samples dataset after augmentations.

	With No Augmentations	Augmented
Boana bischoffi	1836	23052
Dendropsophus minutus	2086	22973
Leptodactylus labyrinthicus	2130	22590
Leptodactylus fuscus	1885	22155
Boana prasina	1947	22098
Physalaemus albonotatus	1383	22007
Rhinella icterica	1505	21832
Ameerega flavopicta	1428	20963
Leptodactylus latrans	1282	20518
Pithecopus azureus	1222	20439
Ameerega picta	1182	20386

Table 7.1: Number of samples per class after augmentation.

the transformation of sample quantities from classes such as "PITAZU" and "PHYMAR" from 40 and 114 samples to 2544 and 6061 samples, respectively. This increment could benefit the model so it augments the possibility that the model could recognize these classes, incrementing its capacity to generalize with unseen data.

Certain conditions must be met in order to mitigate effectively the imbalance when using data augmentation to address imbalance in multi-label datasets. This has been proven with the dataset in question. It is important that the class with the fewest samples does not have an extremely low number of samples. In this dataset, for example, the "PITAZU" label only has 40 samples compared to the other classes that have over a hundred or even a thousand samples. This means that in order to balance the dataset, more than a hundred augmentations would be needed for these types of cases. However, it is not guaranteed that having such a large number of augmentations will not cause overfitting.

Furthermore, with multi-label datasets, the class with the fewest samples should not share a significant number of its samples with one class that has the most samples. This limitation affects the number of augmentations that can be applied to the samples without increasing the number of samples for the classes that are already well-represented, thus maintaining the same level of imbal-

ance. However, increasing the number of samples for these "rare classes", as evident in Figure 7.4 and Table 7.2, can be beneficial as it allows the model to better identify them by having more material to learn the features specific to these classes. At the end, the dataset passed from 7320 samples to 38193 samples.

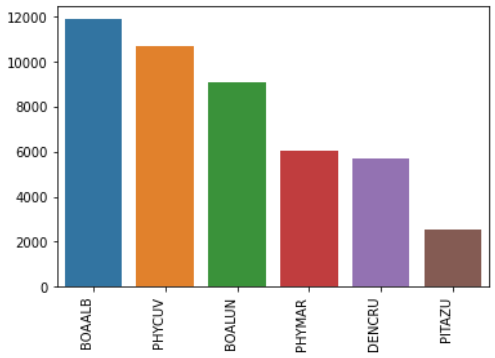

Figure 7.4: Distribution of the Anuraset dataset after augmentations.

	With No Augmentations	Augmented
BOAALB	1947	11896
BOALUN	1082	9086
PHYCUV	1033	10694
DENCRU	296	5676
PHYMAR	114	6061
PITAZU	40	2544

Table 7.2: Number of samples per class after augmentation.

7.2 Model Evaluation and training.

In this section will be presented the results from training the model configuration presented in Section 6.6. These model configurations are tested in four different experiments. It was tested by applying augmentations with transfer learning, using either of these techniques and using neither of them. In these experiments, the results will be depicted considering the classification metrics, such as the F1-score and other metrics that are more related to performance of the model in question of training time and memory occupation. The main objectives comprise observing the effects of data augmentation and transfer learning on the given metrics, while also evaluating the performance of three different CNN architectures using these metrics. Section 7.2.1 will describe the results when augmentations and transfer learning were applied. Meanwhile, Section 7.2.2, Section 7.2.3 and Section 7.2.4 will show how the model reacts when one or both techniques are absent.

One important clarification is that, in the subsections presented next, the learning curves and confusion matrices from just one fold will be displayed. The remaining confusion matrices and learn-

ing curves can be found in Chapter 10, Section 10.0.2, Section 10.0.4 and Section 10.0.6 representing each subsection, respectively.

7.2.1 Augmentations and transfer learning.

The first metric to compare will be the accuracy and loss curves. These curves show if the model appropriately learned from the data provided. The appropriate learning is represented by a learning curve that increases its accuracy during training and that flatten in a certain point in the epochs. If the curve has a noisy behavior or does not stabilize, then it must be done some hyperparameter tunning or it is a signal that the model cannot learn from the data. This behavior is observed in both the validation and training curves; thus, the values of the validation set should exhibit similar patterns to those in the training set. Furthermore, it is essential that the validation and training values exhibit a minimal disparity at the end of the epochs. The presence of a gap between these values shows the occurrence of overfitting in the model. The same concepts apply to the loss curves, but instead of an increasing curve, the curve must decrease and stabilize at a certain value.

The behavior previously described is visible in the curves from the three models and using both Anuraset and the one created from mixed samples. This behavior is achieved by using the configuration described in Section 6.6.1, Section 6.6.2 and Section 6.6.3 with the application of Transfer Learning and by including the augmented samples.

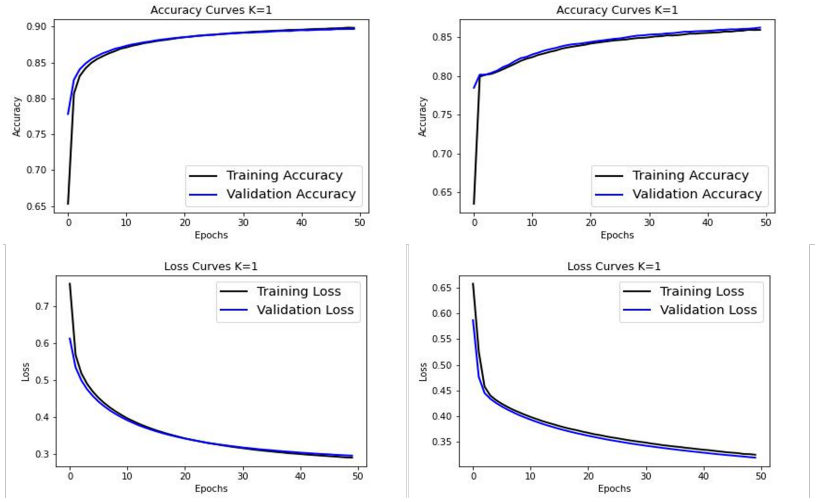

Figure 7.5: Resnet learning curves with transfer learning and augmentation. Left: Mixed samples dataset. Right: Anuraset.

Figure 7.5 illustrates the performance of the Resnet model configuration when applying the aforementioned techniques. In both datasets, the learning curve shows a positive trend, showing effective learning. The accuracy steadily increases and reaches a stable point. Notably, there is a slight dip in the initial epochs, which can be attributed to the model familiarizing itself with the data features or to the initial instability of batch normalization, which estimates mean and

variance. The loss curves exhibit a similar pattern to the accuracy curves, gradually decreasing and stabilizing. Importantly, the values of the training and validation sets are closely aligned, suggesting no overfitting.

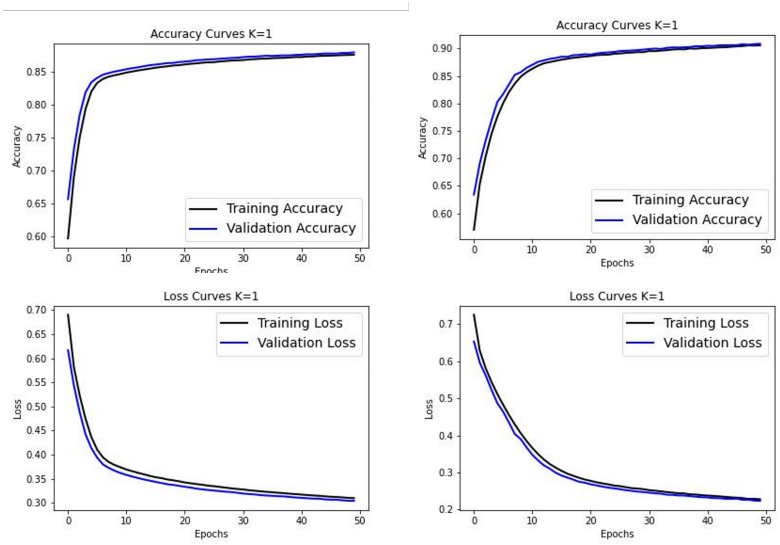

Figure 7.6: VGG learning curves with transfer learning and augmentation. Left: Mixed samples dataset. Right: Anuraset.

In Figure 7.6, a comparison is shown between the accuracy and loss curves of Anuraset and the mixed samples dataset when tested using the VGG19 architecture. Like the results got with Resnet, the configuration achieved a learning curve that exhibits a "good learning" behavior. This is showed by the increasing accuracy values as the epochs progress. The same trend can be observed in the loss curves, indicating that the model requires less aggressive correction and, therefore, the loss values decrease.

Finally, the outcomes of training EfficientNet with both datasets are depicted in Figure 7.7. Similar to the results discussed in the previous paragraph, the training curves of EfficientNet demonstrate the model has successfully learned the data features. By the end of the epochs, the model has achieved a stable and relatively high accuracy value, along with a low loss value. Moreover, this is further supported by the behavior and values of the validation set, which closely align with those of the training set.

While the previously presented graphs show a relatively high accuracy value and can be used to detect overfitting, relying solely on this metric may not provide a completely reliable assessment of performance. It is crucial to utilize other metrics to validate the findings from the accuracy loss curve and acquire a deeper understanding of how the model performs for each label. Therefore, additional metrics, such as the confusion matrix and classification report, will be introduced to offer a more comprehensive evaluation of the model's performance.

From Figure 7.8 till Figure 7.13, the resulting confusion matrices from the training are presented. After analyzing these matrices, it becomes evident that the models have varying levels of success

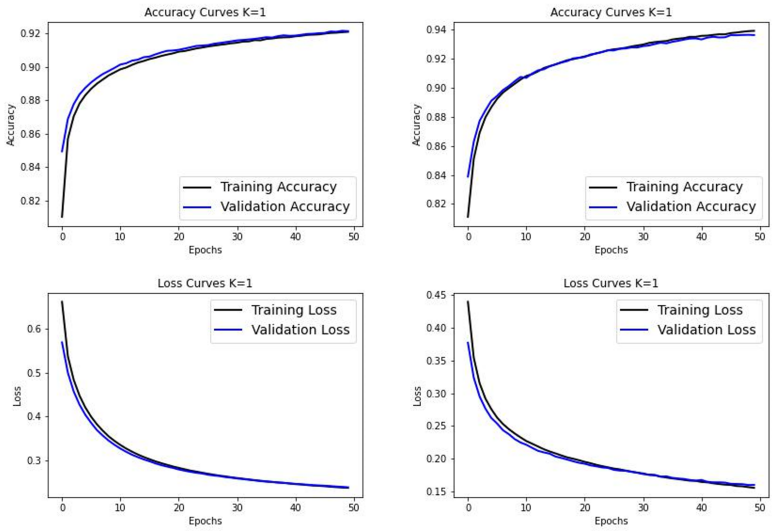

Figure 7.7: EfficientNet learning curves with transfer learning and augmentation. Left: Mixed samples dataset. Right: Anuraset.

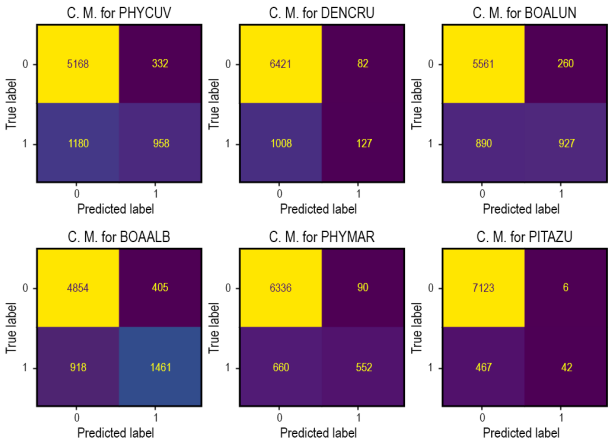

Figure 7.8: Confusion Matrix Anuraset(ResNet)

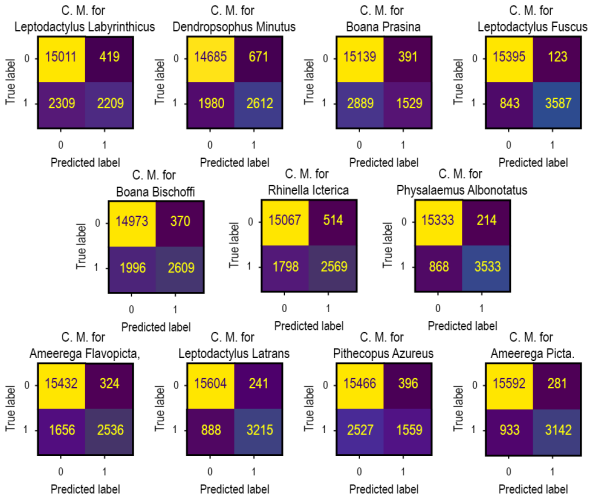

Figure 7.9: Confusion Matrix Mixed samples(ResNet)

in detecting different classes. This discrepancy is particularly noticeable in the "BOAALB" and "PHYCUV" classes from Anuraset, where there are significantly fewer false negatives and false positives compared to the other classes. These results can be attributed to the substantial difference in sample quantities between classes, with some classes having a higher occurrence rate even after applying augmentations, as shown in Figure 7.4.

However, it is important to note that not all models struggled with classifying the "rarest"

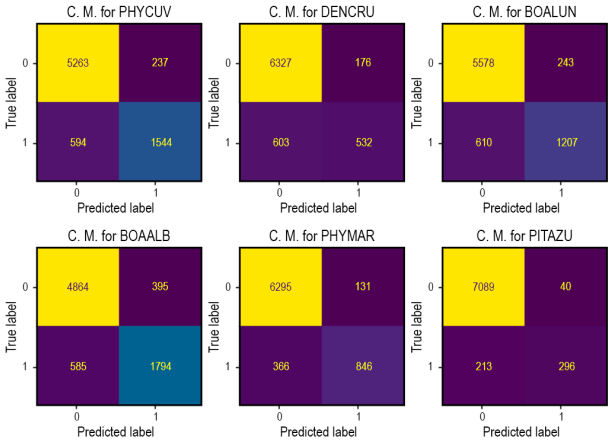

Figure 7.10: Confusion Matrix Anuraset(VGG)

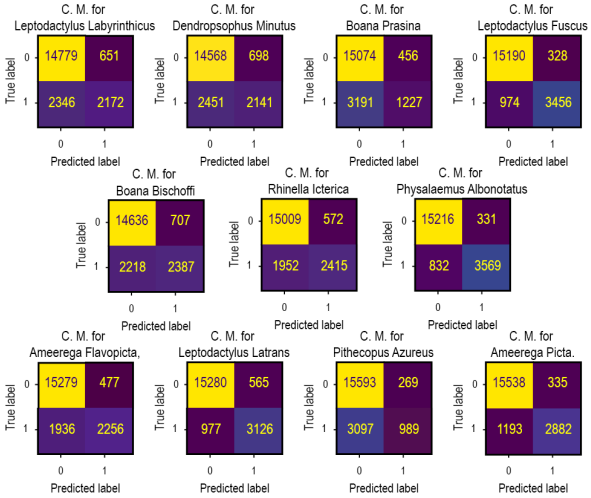

Figure 7.11: Confusion Matrix Mixed samples(VGG)

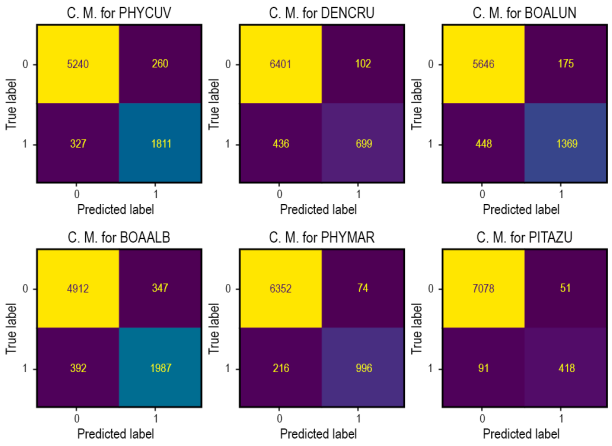

Figure 7.12: Confusion Matrix Anuraset(EfficientNet)

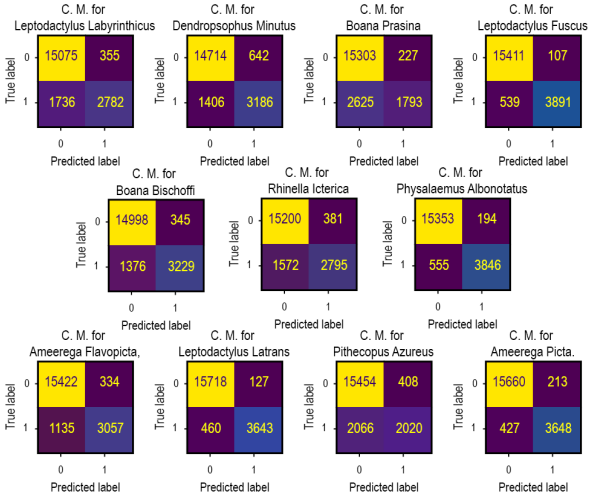

Figure 7.13: Confusion Matrix Mixed samples (EfficientNet)

classes. EfficientNet, for example, had fewer confusions in detecting classes like "PITAZU" and could even detect more samples from the more common classes compared to ResNet. Alternatively, the VGG architecture was able to classify classes with fewer samples but showed more confusion and had less capacity than the EfficientNet. Lastly, as suggested, ResNet exhibited the poorest results, with a considerable number of false negatives and false positives compared to true positives. The only class it could effectively classify was the "BOOALB" class, which had the most samples within Anuraset.

When analyzing the dataset of mixed samples, it was observed a difference compared to the Anuraset dataset. While they follow the same pattern, EfficientNet outperforms the three CNNs, followed by VGG, and finally, the ResNet architecture. Furthermore, it is evident that CNNs have varying capabilities in detecting different classes. Among all the classes in this dataset, "Boana Prasina" and "Phithecopus azureus" prove to be the most challenging to classify. Interestingly, this low performance cannot be solely attributed to an imbalance, as seen in Figure 7.3. Unlike Anuraset, the dataset of mixed samples does not exhibit a significant imbalance, and "Boana Prasina" does not belong to the classes with the fewest instances. Hence, there are additional factors that contribute to the inferior performance of these classes.

The metrics observed and compared in Table 7.3 provide further evidence supporting the earlier discussion on the confusion matrices. This table was derived from the average of the classification reports gathered throughout the iterations, with the standard deviation also included, and it is composed of different metrics such as recall, precision and f1-score with also computing the averaging of these metrics.

Based on the previously mentioned metrics, it can be concluded that EfficientNet performs the best among the models. In both Anuraset and mixed samples, most metrics exceed 0.7, making it a reliable model for multi-label classification. Depending on the dataset, VGG or ResNet could be the models that follow EfficientNet in terms of classification metrics.

When using Anuraset, VGG19 outperforms ResNet, with slightly higher precision and a larger difference in recall values. Notably, ResNet struggles with recall, particularly in class 5 ("PITAZU") where it fails to achieve even 0.1, while VGG achieves approximately 0.56. These results indicate ResNet is sensitive to class imbalance and struggles to identify labels with few occurrences, even with data augmentation. However, in datasets without insignificant balance issues or with more complex data, ResNet may outperform VGG. Such as with the mixed samples dataset, which comprises 11 different labels, ResNet achieves higher metrics than VGG.

To conclude, EfficientNet surpasses other models in terms of metrics and remains resilient to class imbalance and data complexity. This can be attributed to the integration of a scale-up algorithm that adapts the model's complexity to the data, enabling it to effectively identify and differentiate essential features [88][89]. Conversely, ResNet's performance can be affected by class imbalance, and VGG is not well-suited for effectively handling complex data.

Upon closer examination of the values in Table 7.3, the precision of the EfficientNet architecture is exceptionally high. This indicates that the model can accurately predict the correct class when it detects a sample containing a potential class. However, the recall metric does not achieve similar high values. This observation holds true for the other models as well. It suggests that the models face some difficulty in distinguishing relevant samples belonging to a specific class. Notably, among the two datasets, there are three classes that have lower values compared to the rest of the classes. These classes include "DENCRU" from Anuraset, as well as "Boana Prasina" and "Phithecopus azureus". This finding is further supported by examining the Confusion matrices depicted from Figure 7.8 till Figure 7.13 . However, upon analyzing the samples belonging to these classes, it was discovered that the low recall values were not directly associated with the model's ability. Instead, certain circumstances hindered the frequency patterns of these classes. For instance, sometimes, the

background noise had higher intensity than the call itself, resulting in the call being masked. There were occasions when a single sample was associated with multiple classes, resulting in the overlap of one class with another and hindering its identification. An illustration of such cases can be seen in Figure 7.14, where a sample got from the Anuraset dataset corresponds to the "DENCRU" class. However, the background noise obscures the clear identification of the spectral features of the class.

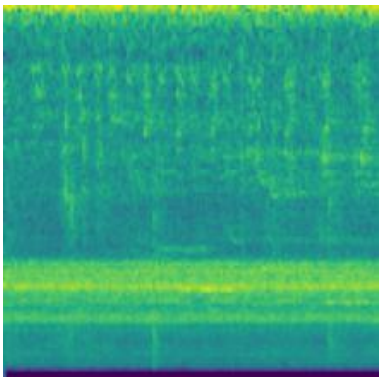

Figure 7.14: Sample with high background nosie from "DENCRU" class.

It is important to note that the metrics presented in Table 7.3 do not fully align with the training and validation curves shown in Figure 7.5, Figure 7.6 and Figure 7.7, nor do they match the averaged accuracy values close to 0.9 observed across all experiments, including those in Section 7.2.2 to Section 7.2.4. This discrepancy arises because binary accuracy is calculated by averaging element-wise correct predictions. Therefore, even if the model makes correct predictions for the majority of samples, the overall accuracy might still appear high. Meanwhile, when calculating average accuracy using the confusion matrix, the accuracy value can be disproportionately high if the number of True Negatives is large, even if the model fails to correctly predict many False Positives. This is particularly problematic for imbalanced datasets, as it does not adequately reflect the model's ability to detect all classes. To gain a more comprehensive understanding of the model's performance, it is crucial to consider the additional metrics provided in Table 7.3 . These metrics offer insight into the model's effectiveness across different classes, providing a more accurate evaluation of its performance.

Using Augmentations and Transfer Learning																				
Model	ResNet50						VGG19						EfficienNetB0							
Dataset	Anuraset			Mixed samples			Anuraset			Mixed samples			Anuraset			Mixed samples				
Metric	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score		
0	0,763 ± 0,014	0,43 ± 0,016	0,55 ± 0,011	0,817 ± 0,016	0,498 ± 0,01	0,619 ± 0,005	0,855 ± 0,023	0,72 ± 0,025	0,781 ± 0,009	0,798 ± 0,027	0,456 ± 0,019	0,58 ± 0,011	0,909 ± 0,026	0,81 ± 0,033	0,856 ± 0,008	0,89 ± 0,016	0,605 ± 0,022	0,72 ± 0,012		
1	0,657 ± 0,063	0,122 ± 0,012	0,205 ± 0,018	0,805 ± 0,01	0,572 ± 0,011	0,669 ± 0,008	0,738 ± 0,025	0,473 ± 0,04	0,575 ± 0,024	0,754 ± 0,011	0,466 ± 0,022	0,575 ± 0,013	0,855 ± 0,018	0,648 ± 0,033	0,736 ± 0,016	0,849 ± 0,013	0,687 ± 0,014	0,759 ± 0,006		
2	0,779 ± 0,006	0,512 ± 0,004	0,618 ± 0,004	0,809 ± 0,008	0,342 ± 0,009	0,481 ± 0,008	0,83 ± 0,026	0,679 ± 0,04	0,746 ± 0,016	0,737 ± 0,007	0,271 ± 0,009	0,396 ± 0,009	0,859 ± 0,021	0,794 ± 0,031	0,824 ± 0,007	0,853 ± 0,026	0,434 ± 0,018	0,575 ± 0,011		
3	0,772 ± 0,008	0,61 ± 0,007	0,682 ± 0,005	0,967 ± 0,003	0,814 ± 0,005	0,884 ± 0,003	0,834 ± 0,027	0,742 ± 0,033	0,784 ± 0,008	0,919 ± 0,005	0,775 ± 0,004	0,841 ± 0,002	0,866 ± 0,019	0,832 ± 0,016	0,848 ± 0,004	0,977 ± 0,003	0,876 ± 0,006	0,923 ± 0,003		
4	0,846 ± 0,012	0,437 ± 0,021	0,576 ± 0,021	0,863 ± 0,01	0,583 ± 0,013	0,696 ± 0,009	0,865 ± 0,011	0,696 ± 0,009	0,772 ± 0,007	0,778 ± 0,018	0,518 ± 0,037	0,621 ± 0,022	0,922 ± 0,027	0,834 ± 0,021	0,875 ± 0,005	0,898 ± 0,01	0,704 ± 0,015	0,789 ± 0,007		
5	0,77 ± 0,082	0,073 ± 0,008	0,133 ± 0,015	0,827 ± 0,014	0,58 ± 0,009	0,682 ± 0,007	0,828 ± 0,039	0,557 ± 0,023	0,666 ± 0,022	0,801 ± 0,013	0,543 ± 0,021	0,647 ± 0,011	0,909 ± 0,015	0,749 ± 0,044	0,821 ± 0,023	0,879 ± 0,015	0,649 ± 0,022	0,747 ± 0,01		
6				0,948 ± 0,004	0,8 ± 0,004	0,868 ± 0,002				0,934 ± 0,015	0,789 ± 0,022	0,855 ± 0,007				0,961 ± 0,007	0,863 ± 0,009	0,909 ± 0,005		
7				0,879 ± 0,009	0,606 ± 0,008	0,717 ± 0,004				0,824 ± 0,011	0,534 ± 0,013	0,648 ± 0,007				0,899 ± 0,025	0,733 ± 0,033	0,807 ± 0,01		
8				0,931 ± 0,004	0,78 ± 0,005	0,849 ± 0,002				0,874 ± 0,02	0,739 ± 0,025	0,8 ± 0,008				0,964 ± 0,006	0,887 ± 0,007	0,924 ± 0,002		
9				0,805 ± 0,014	0,367 ± 0,01	0,504 ± 0,008				0,716 ± 0,055	0,271 ± 0,031	0,391 ± 0,024				0,819 ± 0,031	0,49 ± 0,03	0,612 ± 0,016		
10				0,92 ± 0,004	0,77 ± 0,003	0,838 ± 0,003				0,897 ± 0,02	0,705 ± 0,02	0,789 ± 0,006				0,955 ± 0,007	0,888 ± 0,004	0,92 ± 0,002		
Micro Avg	0,776 ± 0,005	0,436 ± 0,005	0,558 ± 0,005	0,879 ± 0,003	0,609 ± 0,002	0,72 ± 0,001	0,831 ± 0,011	0,675 ± 0,016	0,745 ± 0,006	0,835 ± 0,003	0,551 ± 0,004	0,664 ± 0,002	0,882 ± 0,005	0,792 ± 0,007	0,835 ± 0,002	0,91 ± 0,004	0,709 ± 0,005	0,797 ± 0,002		
Macro Avg	0,765 ± 0,021	0,364 ± 0,005	0,46 ± 0,006	0,87 ± 0,003	0,61 ± 0,002	0,71 ± 0,001	0,825 ± 0,009	0,645 ± 0,014	0,721 ± 0,008	0,821 ± 0,002	0,552 ± 0,004	0,649 ± 0,002	0,887 ± 0,002	0,778 ± 0,007	0,827 ± 0,003	0,904 ± 0,005	0,71 ± 0,005	0,79 ± 0,002		
Weighted Avg	0,767 ± 0,012	0,436 ± 0,005	0,535 ± 0,004	0,869 ± 0,003	0,609 ± 0,002	0,709 ± 0,001	0,83 ± 0,01	0,675 ± 0,016	0,742 ± 0,006	0,82 ± 0,003	0,551 ± 0,004	0,649 ± 0,002	0,883 ± 0,004	0,792 ± 0,007	0,833 ± 0,003	0,904 ± 0,005	0,709 ± 0,005	0,789 ± 0,002		
Samples Avg	0,39 ± 0,006	0,361 ± 0,007	0,363 ± 0,007	0,874 ± 0,002	0,667 ± 0,002	0,727 ± 0,001	0,602 ± 0,01	0,59 ± 0,014	0,584 ± 0,012	0,818 ± 0,002	0,608 ± 0,003	0,665 ± 0,002	0,707 ± 0,005	0,702 ± 0,006	0,694 ± 0,005	0,922 ± 0,003	0,761 ± 0,004	0,808 ± 0,002		
	Average Accuracy		0,862	Average Accuracy		0,897	Average Accuracy		0,907	Average Accuracy		0,878	Average Accuracy		0,937	Average Accuracy		0,921		
	Total Average Accuracy			0,879			Total Average Accuracy			0,893			Total Average Accuracy			0,929				

Table 7.3: Evaluation metrics comparison.

Now it will be analyzed other parameters not directly related to the capacity of the model to classify the classes in the dataset. These parameters are related to how much it costs to train these architectures in terms of training time and memory utilization. The purpose of gathering these characteristics was to examine the distinct advantages of each model compared to others, beyond their classification performance. These metrics are presented in terms of the average quantity of minutes that took each training and the average percentage of memory utilization in each fold.

As shown in Table 7.4, there is not a significant difference between the models in terms of these metrics. VGG required less training time than EfficientNet and ResNet, but this was only observed for the mixed samples datasets. When it comes to Anuraset, EfficientNet was the model that took the least amount of time, but it still required 3 hours of training. In terms of memory usage, the difference between the models is also not substantial. On average, VGG used less GPU memory when trained on mixed samples, while ResNet used less memory when used with Anuraset. This may be surprising considering that EfficientNetB0 has fewer parameters, suggesting that it should require less time and memory compared to the other models. However, these results can be explained by the number of layers and the architecture of these models

EfficientNet has a smaller number of parameters but has 241 layers. On the other hand, ResNet and VGG have 177 and 26 layers, respectively. Additionally, EfficientNet uses depthwise convolution, a form of grouped convolution, which can cause slower and more resource-intensive performance, particularly in the initial layers. This explains why it may take longer and require more resources than expected [100][101]. The scaling up method used by EfficientNet is not always computationally efficient, although it can yield good accuracy results, as shown by the classification metrics mentioned earlier. With VGG and ResNet, VGG has a simpler and more uniform architecture, enabling it to operate faster in certain cases where ResNet needs more time to learn specific patterns. In case of needing a more computationally efficient model, there are version of EfficientNet such as EfficientNetV2 or to check the support for low-level libraries such as cuDNN and OneDNN which helps to make efficient the use of hardware.

	ResNet50		VGG19		EfficientNetB0	
	Anuraset	Mixed Samples	Anuraset	Mixed Samples	Anuraset	Mixed Samples
Avg. Training Time	227,4728	616,1866478	226,1307	584,8391693	221,9355	603,9588
Avg. Memory utilization	0,855439	0,898107422	0,882914	0,862926758	0,858161	0,8730

Table 7.4: Training time and memory utilization of each model.

The last metric analyzed was the convergence speed between models. It compared the convergence of three different models using both datasets. Figure 7.15 shows the convergence when the models were trained with the mixed samples dataset, while Figure 7.16 presents the results after the model was trained using Anuraset as the dataset.

From these figures, it is clear that EfficientNet starts with higher accuracy than the other models and maintains a stable increase until the end of the epochs. ResNet shows a similar behavior to EfficientNet, but with lower accuracy values. VGG starts with the lowest accuracy value among the three models, but quickly learns the features and reaches its maximum value, maintaining stability thereafter.

It is also worth noting that, as suggested by Table 7.3, the accuracy values of VGG surpass

those of ResNet in Anuraset. This suggests that VGG may have better performance than ResNet in datasets that are not complex.

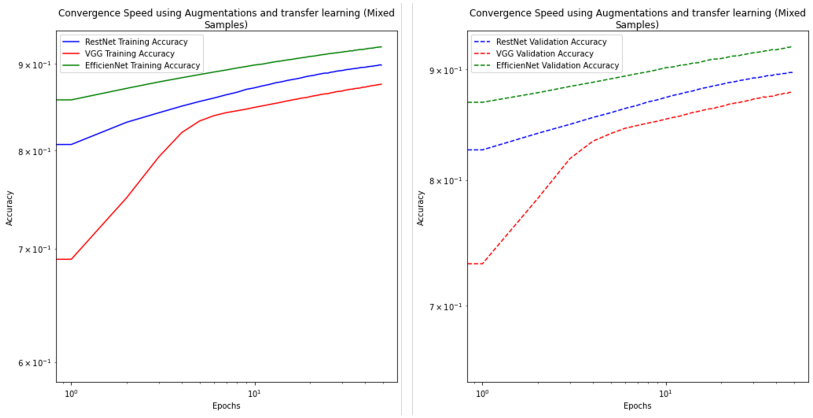

Figure 7.15: Convergence speed of models trained with mixed samples dataset.

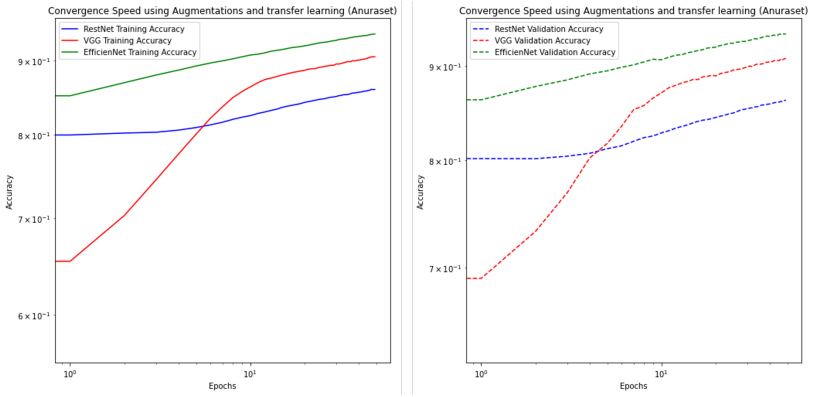

Figure 7.16: Convergence speed of models trained with Anuraset.

7.2.2 No augmentations and transfer learning.

Similar to Section 7.2.1, Figure 7.17, Figure 7.18, and Figure 7.19 illustrate the validation and loss curves obtained from training the CNNs. These models were trained using the dataset without augmentations but with transfer learning applied. The same configurations as presented in Section 6.6.1, Section 6.6.2, and Section 6.6.3 were used.

The curves indicate that the models tend to overfit when data augmentation is not used. This can be observed when the accuracy in the training curve exceeds the accuracy in the validation curve. The loss curve also exhibits similar characteristics, where a lower loss value in the training curve compared to the validation curve implying overfitting.

These patterns are clear in the curves of both datasets during the training of EfficientNet and in the training of the mixed samples dataset in ResNet.

With VGG, these signals previously described are not visible. But still the signals do not show also a behavior than can be considered appropriate learning, thus the signals do not show signals to converge at a certain point neither for the accuracy curve nor for the loss curve, it would appear the values in the accuracy curve will be still increasing meanwhile the loss appears as it would be keep decreasing. This could suggest that the model needs more epochs to converge, or the data is insufficient for the model to learn.

There is a unique scenario observed in ResNet trained with Anuraset, where the data seems to converge perfectly. However, there is no discernible gap between the accuracy and validation curves. It is important to verify this by examining the classification metrics to determine if the model effectively learned the dataset or if this convergence is a result of averaging in binary accuracy.

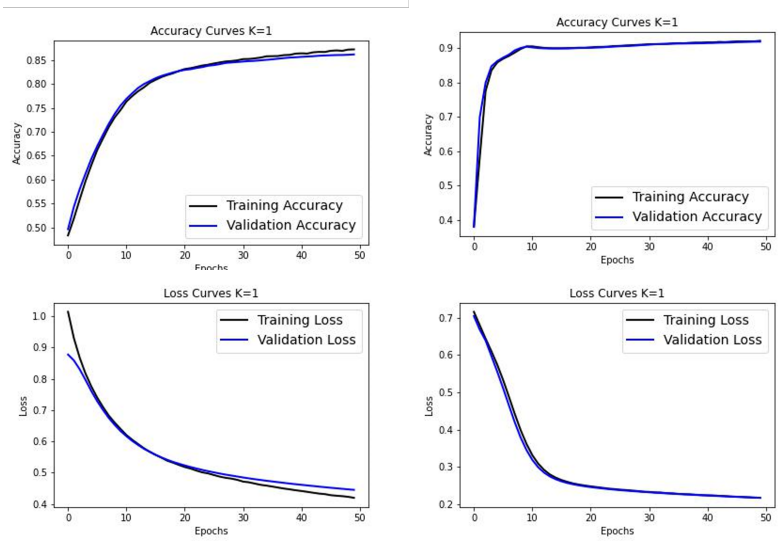

Figure 7.17: Resnet learning curves with transfer learning but without augmentations. Left: Mixed samples dataset. Right: Anuraset.

From Figure 7.20 till Figure 7.25, the confusion matrices from the training of each model are depicted. The performance of the models has decreased significantly. This is noticeable when examining the matrices of the Anuraset dataset, where classes with a small number of samples like "DENCRU," "PHYMAR," and "PITAZU" have mostly zero True Positives across all models, except for VGG and EfficientNet. In the Mixed samples dataset, the difference is not as clear when considering the use of augmentations, although it may appear that the performance has slightly decreased.

Table 7.5 can confirm more precisely what was suggested by the confusion matrices. As it was proposed with the confusion matrices, the difference in performance is more notable in Anuraset than in the dataset of mixed samples. The clearest evidence is the class “PITAZU”. When using augmentations, this class could achieve a f1-score of 0.82 and 0.67 with EfficientNet and VGG,

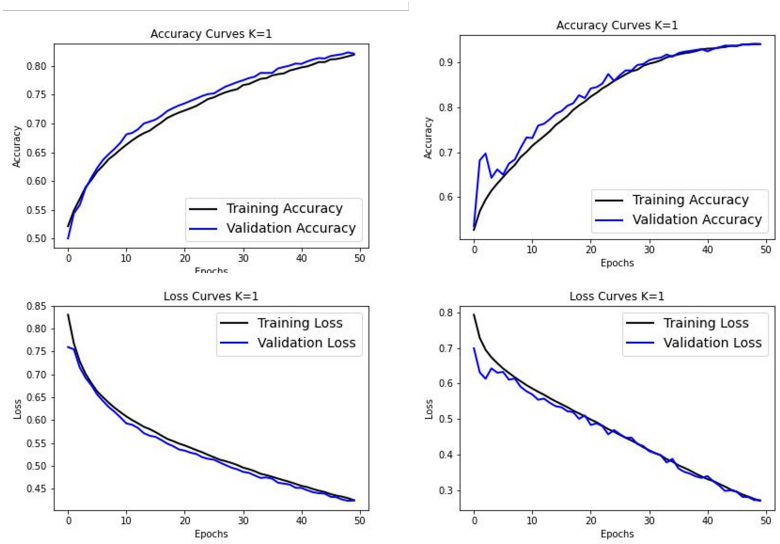

Figure 7.18: VGG learning curves with transfer learning but without augmentations. Left: Mixed samples dataset. Right: Anuraset.

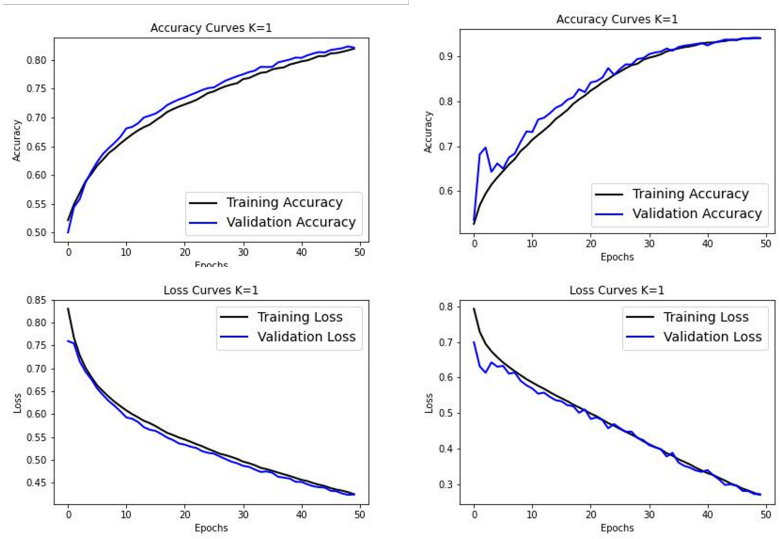

Figure 7.19: EfficientNet learning curves with transfer learning but without augmentations. Left: Mixed samples dataset. Right: Anuraset.

respectively. Meanwhile, when using no augmentations, the metrics are practically 0 except for VGG19, which, on average, was the only model that could slightly detect samples from this label. The order in the model's performance does not change radically in this set. Looking at the global performance of the model, which implies looking at the averaged metrics of the classification report (micro, macro, weighted and samples), EfficientNet is the one that has the best performance in

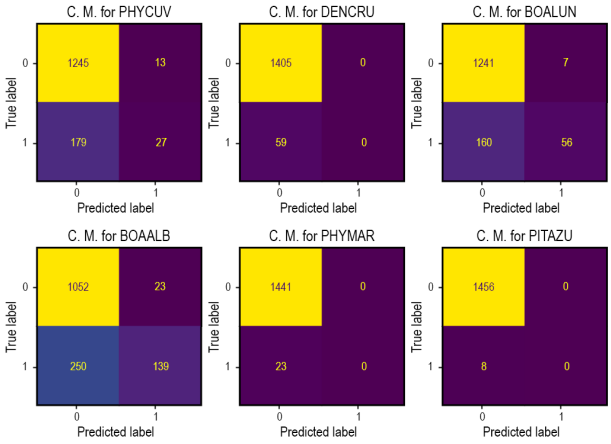

Figure 7.20: Confusion Matrix Anuraset(ResNet)

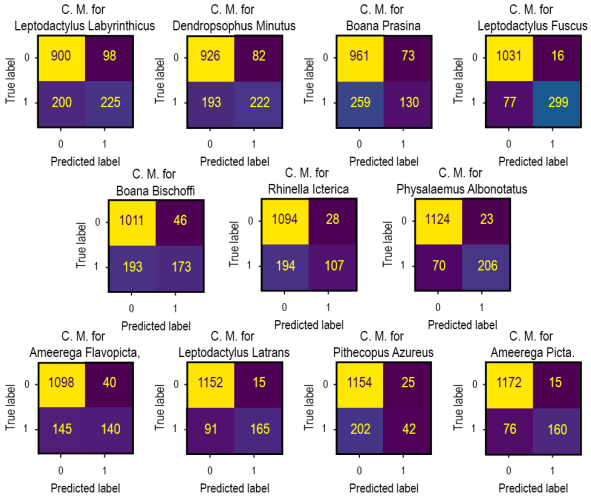

Figure 7.21: Confusion Matrix Mixed samples(ResNet)

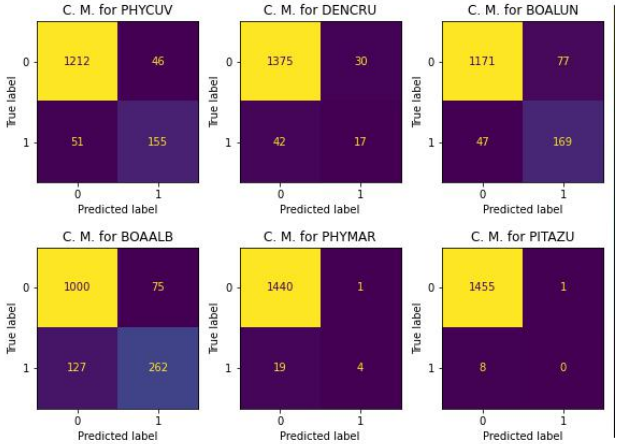

Figure 7.22: Confusion Matrix Anuraset(VGG)

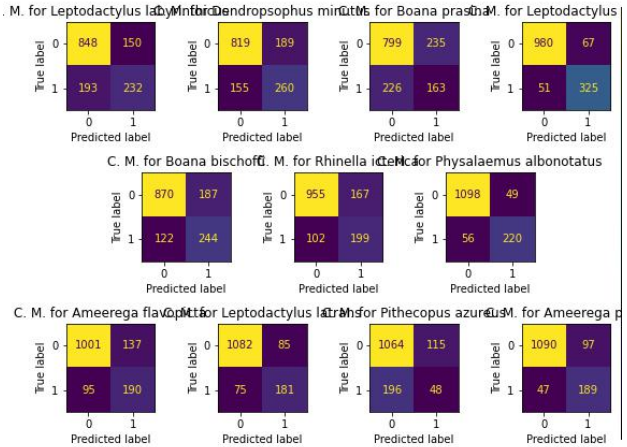

Figure 7.23: Confusion Matrix Mixed samples(VGG)

general. It can be followed by ResNet for the Mixed samples dataset and VGG for Anuraset. Although it is important to note that the values the recall of VGG were better than the rest for the models, especially in Anuraset. By evaluating the Samples Average of Anuraset, it can be concluded that VGG performed marginally better than EfficientNet in terms of metrics. This suggests that VGG can serve as a proficient model without the need for transfer learning.

Moreover, it is crucial to highlight that the recall metric, particularly in Anuraset, is greatly affected by the absence of augmentation. If only precision is considered, the difference does not change significantly. Even in classes with a large number of samples, the precision is slightly reduced. However, upon examining the recall values in each model, it becomes apparent that certain values

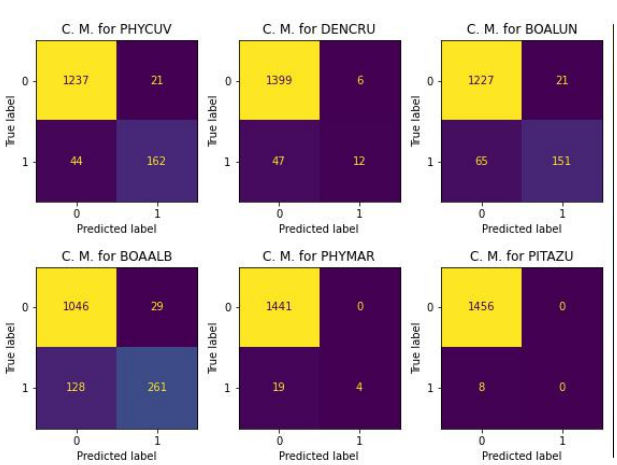

Figure 7.24: Confusion Matrix Anuraset(EfficientNet)

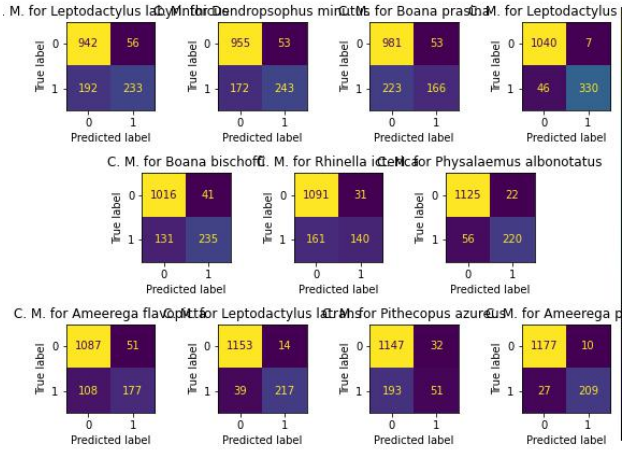

Figure 7.25: Confusion Matrix Mixed samples (EfficientNet)

are considerably diminished, especially in classes that are considered rare, those with only a few samples. Interestingly, the classes that exhibited the lowest values with augmentations are the same classes in this set of experiments that showed the lowest recall values ("DENCRU," "Boana Prasina," and "Phithecopus azureus"), along with additional classes like "PITAZU" and "PHYMAR." The underlying reasons for these low values can be attributed to the dataset imbalance, and the factors discussed in Section Section 7.2.1. Although data augmentation can be advantageous in such cases, improving the model's ability to identify potential samples and refine its pattern recognition.

No Augmentations and application Transfer Learning																				
Model	ResNet50						VGG19						EfficienNetB0							
Dataset	Anuraset			Mixed samples			Anuraset			Mixed samples			Anuraset			Mixed samples				
Metric	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score		
0	0,756 ± 0,073	0,143 ± 0,016	0,241 ± 0,026	0,696 ± 0,004	0,497 ± 0,036	0,579 ± 0,025	0,778 ± 0,015	0,734 ± 0,05	0,754 ± 0,026	0,606 ± 0,034	0,545 ± 0,018	0,574 ± 0,021	0,918 ± 0,032	0,75 ± 0,038	0,825 ± 0,013	0,837 ± 0,03	0,576 ± 0,045	0,682 ± 0,038		
1	0 ± 0	0 ± 0	0 ± 0	0,738 ± 0,017	0,542 ± 0,044	0,624 ± 0,032	0,435 ± 0,056	0,324 ± 0,058	0,37 ± 0,052	0,569 ± 0,014	0,612 ± 0,019	0,589 ± 0,014	0,77 ± 0,123	0,192 ± 0,043	0,306 ± 0,059	0,799 ± 0,03	0,63 ± 0,028	0,704 ± 0,021		
2	0,856 ± 0,043	0,245 ± 0,023	0,38 ± 0,025	0,642 ± 0,01	0,308 ± 0,022	0,416 ± 0,019	0,67 ± 0,031	0,78 ± 0,035	0,72 ± 0,01	0,438 ± 0,033	0,398 ± 0,044	0,416 ± 0,033	0,873 ± 0,004	0,698 ± 0,048	0,775 ± 0,03	0,78 ± 0,024	0,393 ± 0,032	0,522 ± 0,03		
3	0,866 ± 0,016	0,403 ± 0,031	0,55 ± 0,029	0,94 ± 0,007	0,789 ± 0,008	0,858 ± 0,006	0,788 ± 0,012	0,693 ± 0,025	0,738 ± 0,014	0,792 ± 0,028	0,826 ± 0,023	0,809 ± 0,022	0,888 ± 0,012	0,723 ± 0,041	0,796 ± 0,025	0,976 ± 0,007	0,842 ± 0,022	0,904 ± 0,015		
4	0 ± 0	0 ± 0	0 ± 0	0,782 ± 0,015	0,478 ± 0,024	0,593 ± 0,018	0,402 ± 0,245	0,141 ± 0,058	0,202 ± 0,082	0,558 ± 0,016	0,631 ± 0,021	0,592 ± 0,013	0,927 ± 0,101	0,211 ± 0,037	0,342 ± 0,052	0,85 ± 0,024	0,616 ± 0,039	0,713 ± 0,02		
5	0 ± 0	0 ± 0	0 ± 0	0,726 ± 0,049	0,4 ± 0,03	0,514 ± 0,024	0,1 ± 0,224	0,025 ± 0,056	0,04 ± 0,089	0,548 ± 0,028	0,674 ± 0,03	0,605 ± 0,029	0 ± 0	0 ± 0	0 ± 0	0,811 ± 0,043	0,506 ± 0,039	0,622 ± 0,034		
6				0,917 ± 0,016	0,764 ± 0,03	0,833 ± 0,019				0,854 ± 0,021	0,822 ± 0,026	0,837 ± 0,021				0,923 ± 0,016	0,823 ± 0,034	0,87 ± 0,026		
7				0,798 ± 0,022	0,467 ± 0,049	0,588 ± 0,041				0,624 ± 0,035	0,62 ± 0,058	0,62 ± 0,033				0,822 ± 0,041	0,598 ± 0,03	0,692 ± 0,028		
8				0,91 ± 0,027	0,623 ± 0,022	0,739 ± 0,016				0,657 ± 0,03	0,723 ± 0,033	0,688 ± 0,012				0,941 ± 0,014	0,833 ± 0,034	0,883 ± 0,016		
9				0,568 ± 0,056	0,165 ± 0,016	0,255 ± 0,02				0,322 ± 0,029	0,213 ± 0,019	0,256 ± 0,015				0,626 ± 0,039	0,251 ± 0,037	0,357 ± 0,04		
10				0,874 ± 0,025	0,623 ± 0,051	0,727 ± 0,041				0,64 ± 0,026	0,748 ± 0,033	0,689 ± 0,023				0,94 ± 0,014	0,844 ± 0,049	0,888 ± 0,026		
Micro Avg	0,848 ± 0,023	0,266 ± 0,017	0,404 ± 0,02	0,795 ± 0,006	0,515 ± 0,008	0,625 ± 0,007	0,727 ± 0,012	0,679 ± 0,028	0,702 ± 0,012	0,606 ± 0,004	0,617 ± 0,014	0,612 ± 0,008	0,888 ± 0,003	0,669 ± 0,031	0,763 ± 0,021	0,861 ± 0,005	0,623 ± 0,014	0,723 ± 0,01		
Macro Avg	0,413 ± 0,017	0,132 ± 0,009	0,195 ± 0,01	0,781 ± 0,008	0,514 ± 0,009	0,612 ± 0,007	0,529 ± 0,029	0,45 ± 0,024	0,471 ± 0,01	0,601 ± 0,005	0,619 ± 0,015	0,607 ± 0,009	0,729 ± 0,025	0,429 ± 0,021	0,567 ± 0,018	0,846 ± 0,006	0,628 ± 0,016	0,712 ± 0,012		
Weighted Avg	0,752 ± 0,026	0,266 ± 0,017	0,383 ± 0,018	0,776 ± 0,007	0,515 ± 0,008	0,611 ± 0,006	0,719 ± 0,009	0,679 ± 0,028	0,693 ± 0,013	0,6 ± 0,005	0,617 ± 0,014	0,606 ± 0,009	0,877 ± 0,007	0,669 ± 0,031	0,747 ± 0,019	0,845 ± 0,007	0,623 ± 0,014	0,71 ± 0,011		
Samples Avg	0,123 ± 0,009	0,092 ± 0,007	0,1 ± 0,008	0,8 ± 0,006	0,577 ± 0,008	0,637 ± 0,008	0,241 ± 0,009	0,254 ± 0,01	0,24 ± 0,008	0,651 ± 0,009	0,662 ± 0,015	0,618 ± 0,01	0,263 ± 0,009	0,252 ± 0,01	0,252 ± 0,01	0,875 ± 0,008	0,68 ± 0,014	0,735 ± 0,012		
	Average Accuracy		0,919	**Average Accuracy**		0,857	**Average Accuracy**		0,941	**Average Accuracy**		0,819	**Average Accuracy**		0,957	**Average Accuracy**		0,890		
	Total Average Accuracy			0,888			**Total Average Accuracy**			0,880			**Total Average Accuracy**			0,924				

Table 7.5: Evaluation metrics comparison.

Regarding the metrics of training time and memory utilization in Table 7.6, the most notorious difference can be observed in the training time, which is considerably reduced if compared with the training time in Section 7.2.1. It is essential to note that the number of samples has been substantially reduced, resulting in datasets comprising approximately over 7000 samples with no augmentations. Then the models can faster learn the actual samples with the actual batch configuration, leading to a training that takes approximately 40 minutes for each model. The order is not the same order that was observed in Section 7.2.1. Among the three models tested on Anuraset,

VGG required the least amount of training on average. However, when using mixed samples, EfficientNet had the shortest training time compared to the other two models. Therefore, there is not a similar a pattern to follow for training time. Although still ResNet is the model that needs a more amount of time to be trained.

In terms of memory utilization, there is no consistent pattern that repeats. However, across both datasets, ResNet consistently occupied less memory compared to other models. Additionally, in Section 7.2.1, it was found that, on average, VGG also used less memory for the mixed samples dataset. Surprisingly, the percentage of memory used did not decrease with the use of fewer samples, showing that the models always use the maximum available memory space.

	ResNet50		VGG19		EfficientNetB0	
	Anuraset	**Mixed Samples**	**Anuraset**	**Mixed Samples**	**Anuraset**	**Mixed Samples**
Avg. Training Time	46,8896	44,09062121	41,55936	43,61567439	44,8245	43,1138
Avg. Memory utilization	0,867244	0,850817871	0,889703	0,88885498	0,908022	0,8605

Table 7.6: Training time and memory utilization of each model.

Finally, when analyzing the convergence speed of the model in this set of experiments, it is evident from Figure 7.26 and Figure 7.27 that the absence of augmentation impacted the model's convergence. EfficientNet exhibited a similar behavior, starting from higher values compared to other models and continuing to increase. However, the rate of increase in Anuraset was less pronounced than in Section 7.2.1. In all models, the growth was less stable than in Section 7.2.1. VGG and ResNet both started from low values and gradually increased, with VGG showing no point of stability and continuing to increase with each epoch. On the other hand, ResNet reached a stable point that persisted until the end of the epochs. Furthermore, the curves in the graph exhibited some noisy behavior, particularly in VGG. These observations suggest that augmentation not only helps the model achieve more stable training but also facilitates faster convergence.

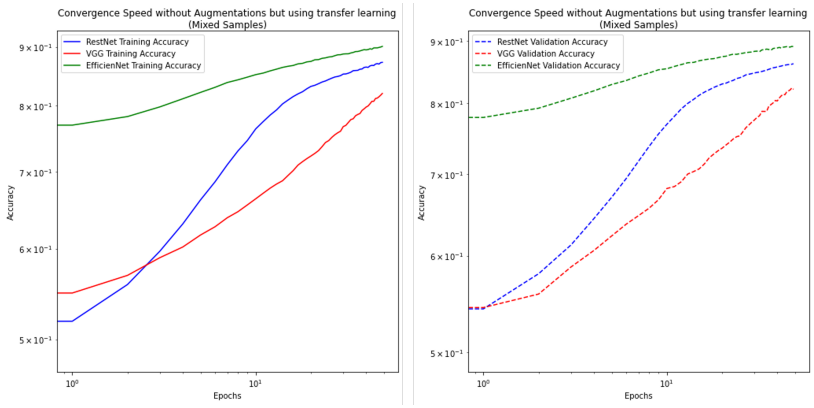

Figure 7.26: Convergence speed of models trained with mixed samples dataset.

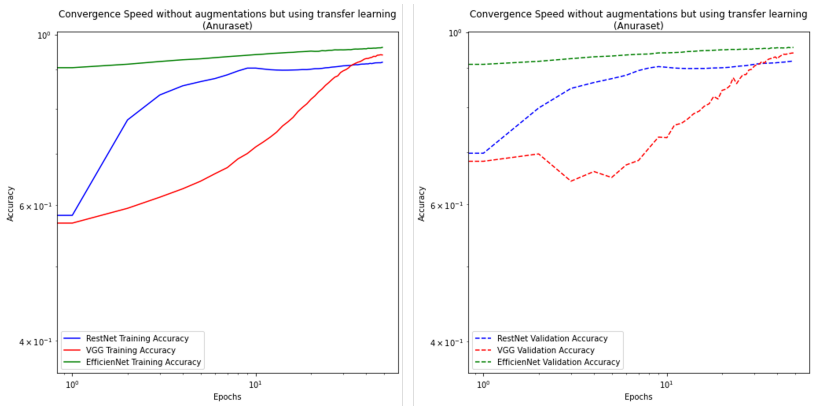

Figure 7.27: Convergence speed of models trained with Anuraset.

7.2.3 Augmentations and no transfer learning.

The next set of experiments was done by using the dataset with augmentations but without transfer learning applied in the model, therefore the model does not start with the weights learned with "ImageNet" neither the layers of the model were frozen. The models will learn from scratch the features and patterns corresponding to each class in the dataset. Learning curves from training ResNet, VGG, and EfficientNet are shown in Figure 7.28, Figure 7.29, and Figure 7.30, respectively, with the previous considerations applied. When analyzing these figures, it is possible to see that there are signs of overfitting in all the models. As explained in Section 7.2.2, if the accuracy of the validation set is lower and the loss is higher compared to the training set, it shows that the model is overfitting. This means that the model performs worse on the validation set because it has not properly learned the underlying characteristics of the data, but has only memorized the patterns from the training set. It is worth noting that binary accuracy considers the results from all classes, so even correctly predicting the classes with more samples should achieve high accuracy. Another common characteristic observed in most of the graphs, except for EfficientNet, is the presence of considerable noise in the validation set. The curve shows multiple peaks, some more pronounced than others, as observed when comparing the validation curves of ResNet and VGG.

Contrary to the learning curves that were previously presented, the confusion matrices (Figures 7.31 to 7.36) do not show erratic values. In fact, the number of True Positives is greater than those achieved with augmentations and transfer learning. ResNet could accurately identify classes like "DENCRU" and "PITAZU," which were challenging for this model when using transfer learning. Additionally, when training the model from scratch, they could identify more samples of classes that were previously difficult, such as "Boana Prasina" and "Phithecopus azureus." These metrics suggest that the models are even more capable of identifying all the classes in the dataset, although it may appear to be overfitting.

The confusion matrices indicate that the metrics have improved compared to those presented in Section 7.2.1 as shown in Table 7.7. The extent of improvement varies depending on the model and the specific metric being observed. For example, when comparing EfficientNet with its values

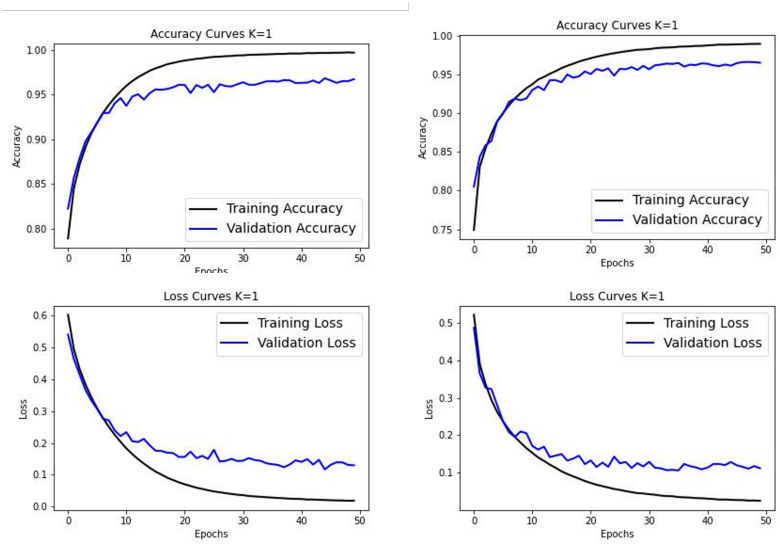

Figure 7.28: Resnet learning curves with augmentations but without transfer learning. Left: Mixed samples dataset. Right: Anuraset.

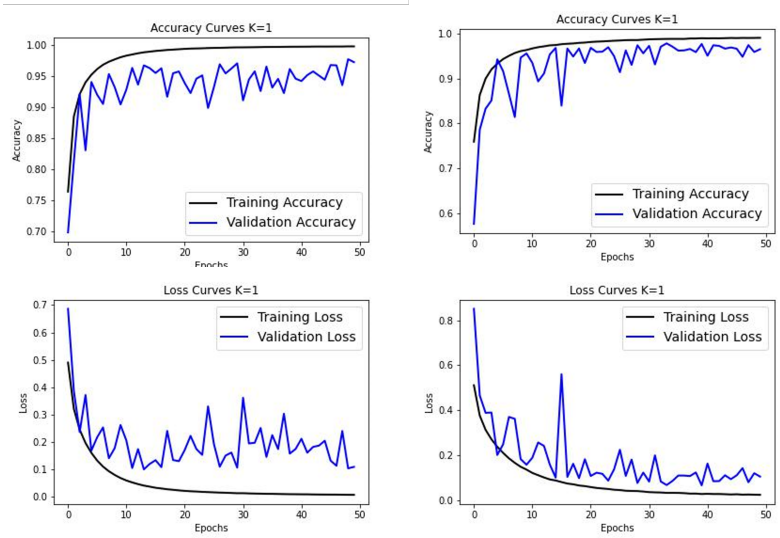

Figure 7.29: VGG learning curves with augmentations but without transfer learning. Left: Mixed samples dataset. Right: Anuraset.

in Section 7.2.1, the averaged metrics show little change, remaining around 0.8. However, the most noticeable improvement is seen in recall, with most values surpassing 0.8, except for the samples average, which also shows a slight increase of 0.05.

Similar patterns are observed with the mixed samples datasets, where precision does not change drastically, but there is a larger increase in recall and consequently in the F1-score. Here, the

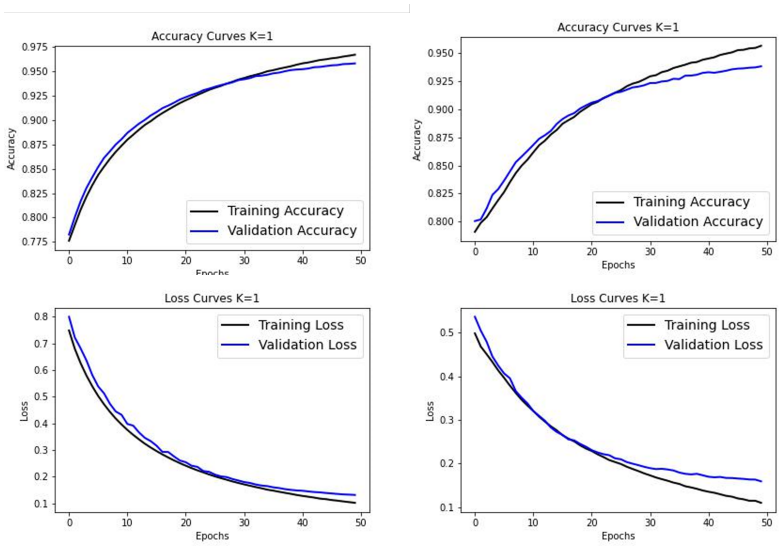

Figure 7.30: EfficientNet learning curves with augmentations but without transfer learning. Left: Mixed samples dataset. Right: Anuraset.

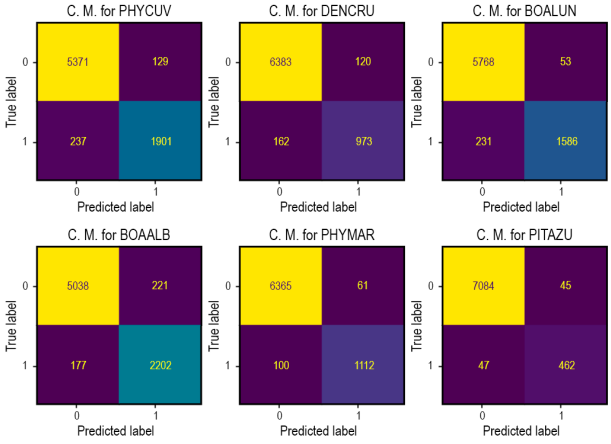

Figure 7.31: Confusion Matrix Anuraset(ResNet)

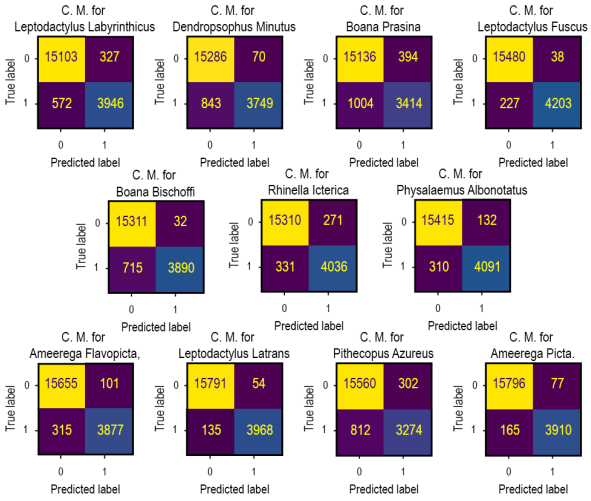

Figure 7.32: Confusion Matrix Mixed samples(ResNet)

increase in recall is slightly greater than with Anuraset.

In VGG and ResNet, the change is more significant than with EfficientNet. The averaged weights now mostly exceed 0.9. Both metrics are affected by the use of augmentations during model training from scratch. Recall shows considerable growth, especially for ResNet, indicating that the models have improved in their ability to identify and classify samples correctly. This improvement could be attributed to the observed overfitting in the learning curves, which is more likely to occur in models

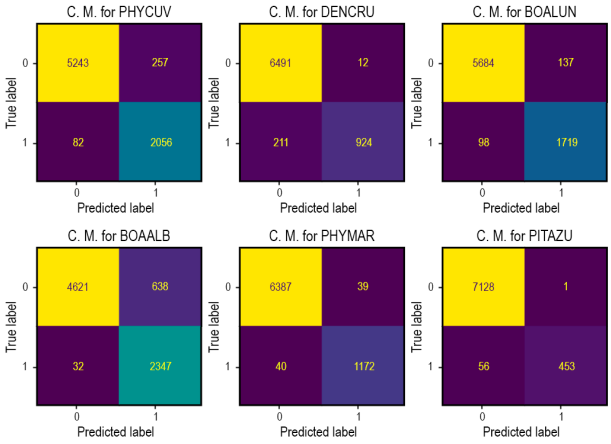

Figure 7.33: Confusion Matrix Anuraset(VGG)

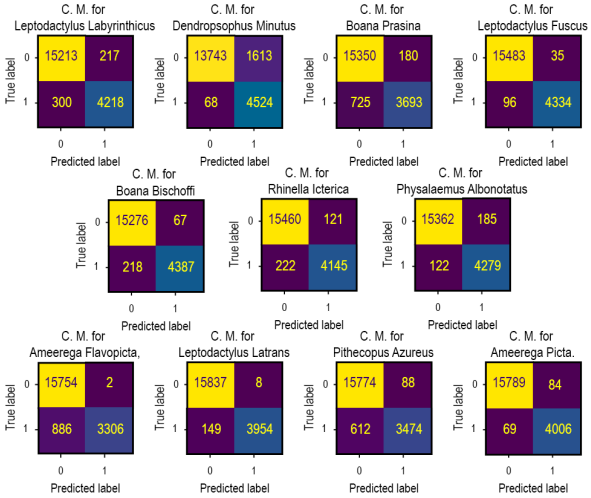

Figure 7.34: Confusion Matrix Mixed samples(VGG)

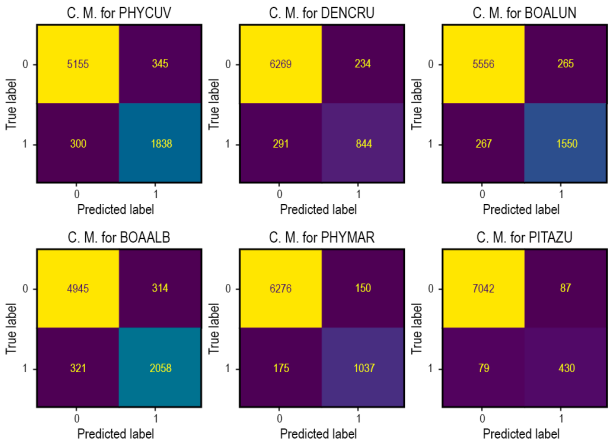

Figure 7.35: Confusion Matrix Anuraset(EfficientNet)

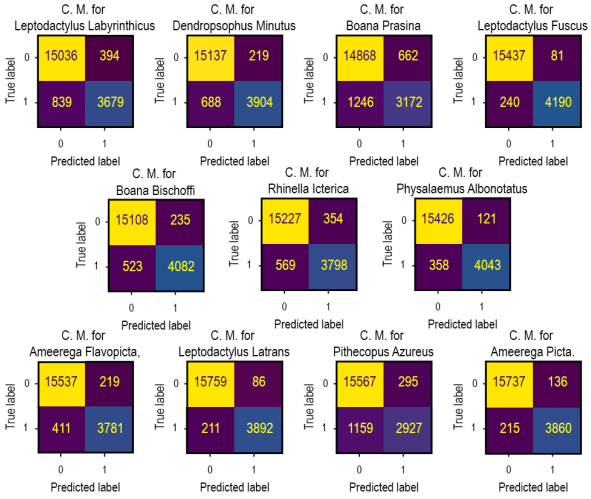

Figure 7.36: Confusion Matrix Mixed samples (EfficientNet)

like VGG and ResNet because of their large number of parameters and complex architecture.

Another important observation is that the recall has increased for classes that previously caused more confusion in the model, such as "DENCRU," "Boana Prasina," and "Phithecopus azureus". This suggests that the model has learned to identify the specific patterns of these classes, even when they are obscured by other factors.

In summary, it is possible that when the model learns from scratch using the spectrogram, it becomes accustomed to the different frequency patterns and becomes better at differentiating them

compared to transfer learning, which uses weights from "ImageNet" trained on different images rather than spectrograms. However, it is important to consider that these good metrics may also result from overfitting, which does not guarantee similar high values with other test sets.

Augmentations and No Transfer Learning																		
Model	ResNet50						VGG19						EfficienNetB0					
Dataset	Anuraset			Mixed samples			Anuraset			Mixed samples			Anuraset			Mixed samples		
Metric	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
0	0,94 ± 0,012	0,897 ± 0,023	0,918 ± 0,008	0,938 ± 0,03	0,831 ± 0,051	0,88 ± 0,019	0,92 ± 0,069	0,929 ± 0,042	0,922 ± 0,022	0,96 ± 0,024	0,913 ± 0,05	0,935 ± 0,019	0,848 ± 0,011	0,856 ± 0,007	0,852 ± 0,005	0,905 ± 0,013	0,806 ± 0,014	0,852 ± 0,003
1	0,887 ± 0,03	0,889 ± 0,02	0,888 ± 0,016	0,944 ± 0,03	0,898 ± 0,05	0,919 ± 0,015	0,987 ± 0,011	0,756 ± 0,123	0,851 ± 0,077	0,863 ± 0,123	0,935 ± 0,074	0,889 ± 0,038	0,782 ± 0,013	0,761 ± 0,02	0,771 ± 0,006	0,933 ± 0,013	0,872 ± 0,017	0,902 ± 0,004
2	0,941 ± 0,044	0,882 ± 0,034	0,909 ± 0,009	0,809 ± 0,08	0,844 ± 0,05	0,822 ± 0,021	0,886 ± 0,087	0,946 ± 0,041	0,911 ± 0,034	0,86 ± 0,145	0,882 ± 0,051	0,861 ± 0,065	0,862 ± 0,009	0,856 ± 0,016	0,859 ± 0,007	0,841 ± 0,023	0,71 ± 0,021	0,77 ± 0,006
3	0,934 ± 0,025	0,914 ± 0,018	0,923 ± 0,004	0,965 ± 0,026	0,965 ± 0,012	0,965 ± 0,008	0,897 ± 0,064	0,961 ± 0,02	0,927 ± 0,029	0,985 ± 0,017	0,977 ± 0,01	0,981 ± 0,005	0,873 ± 0,008	0,865 ± 0,004	0,869 ± 0,004	0,984 ± 0,002	0,945 ± 0,004	0,964 ± 0,001
4	0,956 ± 0,013	0,915 ± 0,012	0,935 ± 0,005	0,959 ± 0,048	0,907 ± 0,045	0,93 ± 0,013	0,96 ± 0,024	0,901 ± 0,13	0,924 ± 0,069	0,988 ± 0,01	0,932 ± 0,039	0,958 ± 0,017	0,881 ± 0,009	0,847 ± 0,017	0,863 ± 0,006	0,944 ± 0,009	0,895 ± 0,006	0,919 ± 0,004
5	0,938 ± 0,04	0,895 ± 0,023	0,915 ± 0,012	0,95 ± 0,02	0,909 ± 0,024	0,929 ± 0,005	0,986 ± 0,01	0,893 ± 0,053	0,936 ± 0,027	0,907 ± 0,146	0,944 ± 0,039	0,918 ± 0,077	0,823 ± 0,035	0,83 ± 0,029	0,826 ± 0,03	0,923 ± 0,011	0,858 ± 0,018	0,889 ± 0,006
6				0,973 ± 0,003	0,932 ± 0,004	0,952 ± 0,003				0,986 ± 0,017	0,899 ± 0,103	0,937 ± 0,056				0,976 ± 0,003	0,913 ± 0,005	0,943 ± 0,002
7				0,98 ± 0,004	0,91 ± 0,015	0,944 ± 0,006				0,986 ± 0,018	0,912 ± 0,072	0,946 ± 0,036				0,943 ± 0,005	0,9 ± 0,011	0,921 ± 0,004
8				0,988 ± 0,004	0,958 ± 0,007	0,973 ± 0,003				0,919 ± 0,153	0,978 ± 0,012	0,941 ± 0,088				0,979 ± 0,003	0,942 ± 0,004	0,96 ± 0,003
9				0,907 ± 0,03	0,815 ± 0,03	0,858 ± 0,008				0,983 ± 0,007	0,802 ± 0,044	0,883 ± 0,024				0,891 ± 0,016	0,735 ± 0,018	0,805 ± 0,006
10				0,985 ± 0,006	0,955 ± 0,01	0,97 ± 0,003				0,873 ± 0,159	0,952 ± 0,083	0,9 ± 0,085				0,969 ± 0,003	0,944 ± 0,004	0,956 ± 0,001
Micro Avg	0,933 ± 0,016	0,9 ± 0,012	0,916 ± 0,004	0,942 ± 0,013	0,902 ± 0,012	0,921 ± 0,004	0,917 ± 0,04	0,914 ± 0,022	0,915 ± 0,021	0,92 ± 0,035	0,921 ± 0,019	0,92 ± 0,015	0,852 ± 0,003	0,844 ± 0,006	0,848 ± 0,004	0,936 ± 0,002	0,865 ± 0,001	0,899 ± 0,001
Macro Avg	0,933 ± 0,014	0,899 ± 0,01	0,915 ± 0,005	0,945 ± 0,01	0,902 ± 0,011	0,922 ± 0,004	0,939 ± 0,024	0,898 ± 0,031	0,912 ± 0,024	0,937 ± 0,027	0,921 ± 0,019	0,923 ± 0,013	0,845 ± 0,005	0,836 ± 0,007	0,84 ± 0,005	0,935 ± 0,002	0,865 ± 0,001	0,898 ± 0,001
Weighted Avg	0,934 ± 0,016	0,9 ± 0,012	0,916 ± 0,004	0,945 ± 0,011	0,902 ± 0,012	0,921 ± 0,004	0,925 ± 0,034	0,914 ± 0,022	0,913 ± 0,022	0,937 ± 0,026	0,921 ± 0,019	0,923 ± 0,013	0,852 ± 0,003	0,844 ± 0,006	0,848 ± 0,004	0,935 ± 0,002	0,865 ± 0,001	0,898 ± 0,001
Samples Avg	0,808 ± 0,006	0,804 ± 0,01	0,8 ± 0,006	0,951 ± 0,009	0,919 ± 0,01	0,922 ± 0,004	0,799 ± 0,024	0,81 ± 0,021	0,798 ± 0,022	0,932 ± 0,029	0,934 ± 0,017	0,919 ± 0,016	0,736 ± 0,004	0,751 ± 0,005	0,732 ± 0,005	0,947 ± 0,002	0,888 ± 0,001	0,901 ± 0,001
	Average Accuracy		0,967	Average Accuracy		0,966	Average Accuracy		0,966	Average Accuracy		0,965	Average Accuracy		0,939	Average Accuracy		0,958
	Total Average Accuracy			0,967			Total Average Accuracy			0,965			Total Average Accuracy			0,949		

Table 7.7: Evaluation metrics comparison.

After examining the metrics related to training time and memory utilization (Table 7.8), both have increased compared to the corresponding metrics in Section 7.2.1. However, there is an exception for ResNet when trained with Anuraset. Here, training from scratch requires more time and memory for the model compared to when transfer learning is applied. Unfortunately, there is no discernible pattern that would allow us to identify a model that consistently uses less memory or takes more training time. Furthermore, when comparing the parameters to those in Section 7.2.2, it is logical to see an increase in training time because of the larger number of samples. However, the memory utilization does not consistently increase or decrease.

	ResNet50		VGG19		EfficientNetB0	
	Anuraset	Mixed Samples	Anuraset	Mixed Samples	Anuraset	Mixed Samples
Avg. Training Time	226,8844	687,0430358	234,1254	675,7468086	229,1372	617,7635
Avg. Memory utilization	0,839943	0,918595215	0,883479	0,882944336	0,864054	0,8918

Table 7.8: Training time and memory utilization of each model.

The final analysis of these experiments, as well as others, focuses on the convergence of the model represented in Figure 7.37 and Figure 7.38. When training the model from scratch, EfficientNet had the lowest starting value, while VGG had the highest starting value among the three when using the mixed samples dataset. ResNet had the highest starting point when using Anuraset. In the training set, the model's growth is stable, although slightly more pronounced in certain models, such as EfficientNet and ResNet with the mixed samples dataset, and VGG with Anuraset. The behavior of the validation set is like what was observed in Section 7.2.2, where the curve can exhibit some noise, as seen in VGG. It is worth noting that EfficientNet consistently displays stable behavior regardless of the set of experiments.

7.2.4 No augmentations and no transfer learning.

The final set of experiments used the original datasets with no augmentations or transfer learning. The learning curves obtained after training are depicted in Figure 7.39, Figure 7.40, and Figure 7.41. These curves exhibit a similar but more pronounced erratic behavior compared to the learning

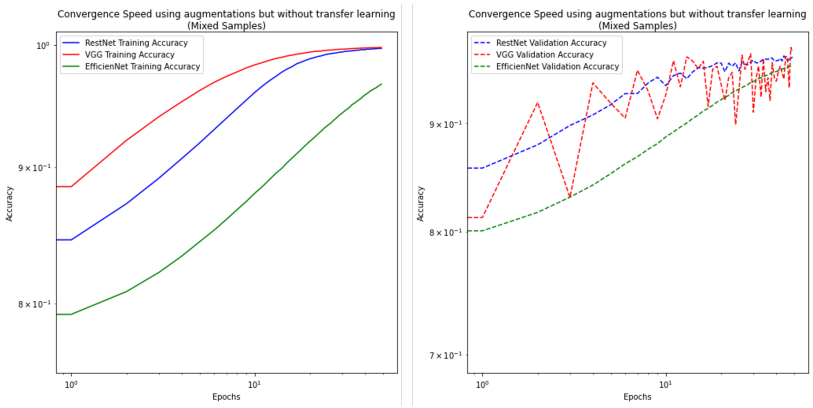

Figure 7.37: Convergence speed of models trained with mixed samples dataset.

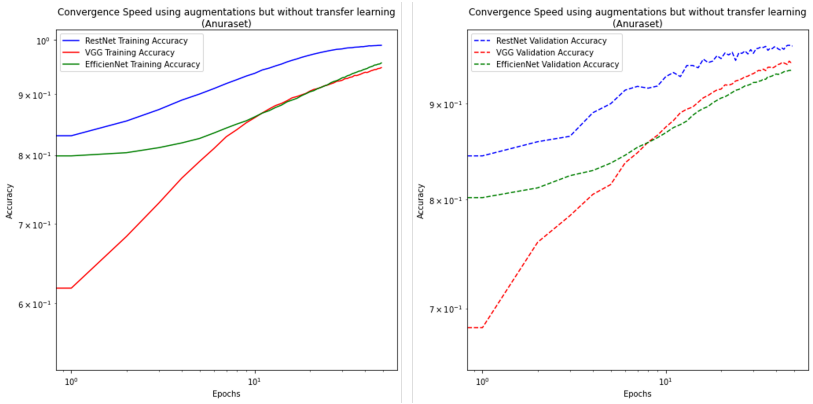

Figure 7.38: Convergence speed of models trained with Anuraset.

curves discussed in Section 7.2.3. Notably, the validation set exhibits significantly lower accuracy and higher loss values than the training set, indicating a potential issue of overfitting. Additionally, with VGG, the behavior of the validation set is highly fluctuating, with multiple peaks observed throughout the curve. This suggests that the accuracy of the validation set fluctuates during epochs and cannot stabilize even at the end. For EfficientNet, the curve does not show convergence at any specific point but rather displays a consistent increase in accuracy and a continuous decrease in loss.

To conduct a thorough analysis of the model's performance, it was examined the confusion matrices obtained from evaluating the model. The information presented in Figures 7.42 to 7.47 is similar to what was discussed in Section 7.2.2 regarding transfer learning without augmentations. In most cases, classes like "PITAZU" from Anuraset had zero True Positives, showing difficulty in detection. Additionally, classes such as "Boana Prasina" and "Pithecopus Azureus" posed challenges for the models, resulting in more False Positives or False Negatives compared to True Positives. Interestingly, the VGG model performed better at detecting samples from these challenging classes compared to the other models.

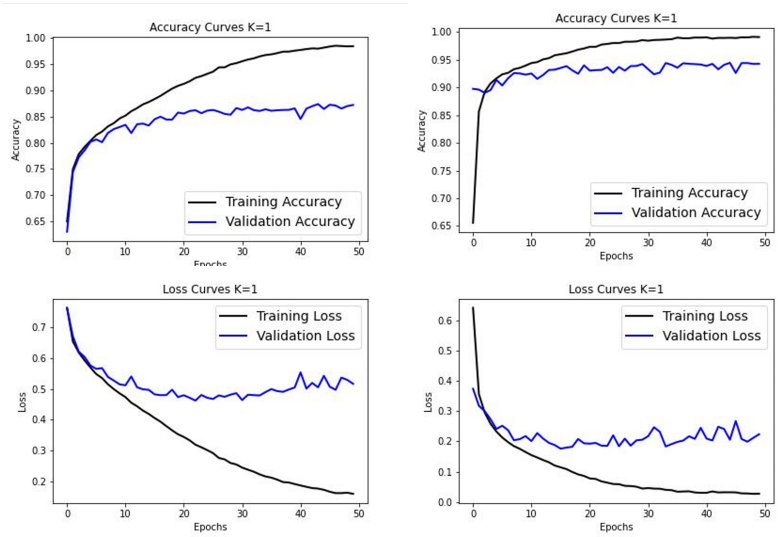

Figure 7.39: Resnet learning curves with no transfer learning and without augmentations. Left: Mixed samples dataset. Right: Anuraset.

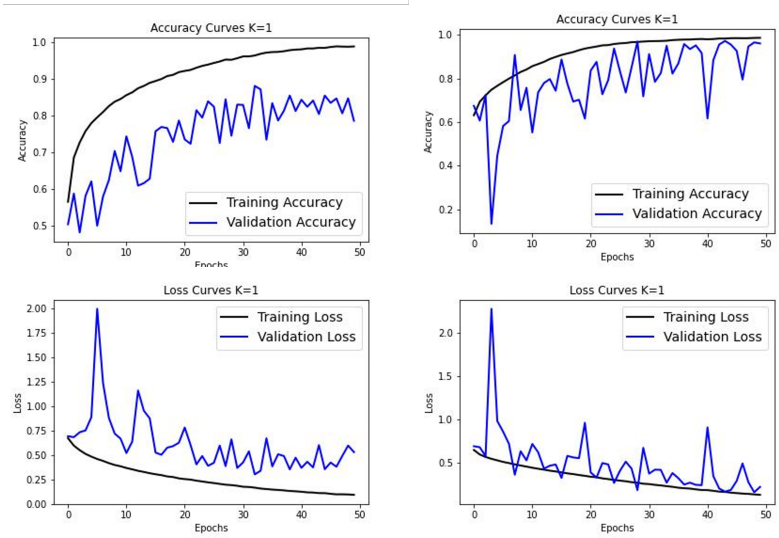

Figure 7.40: VGG learning curves with no transfer learning and without augmentations. Left: Mixed samples dataset. Right: Anuraset.

To validate the findings presented in the confusion matrices, Table 7.9 showcases the evaluation metrics obtained by averaging the results from the training folds. These values can be considered the least favorable, depending on the model and metric being analyzed. Specifically, the performance of EfficientNet was significantly hindered, particularly with Anuraset, where the classes "PHYMAR," "DENCRU," and "PITAZU" were almost unidentifiable by this architecture. This is in contrast to

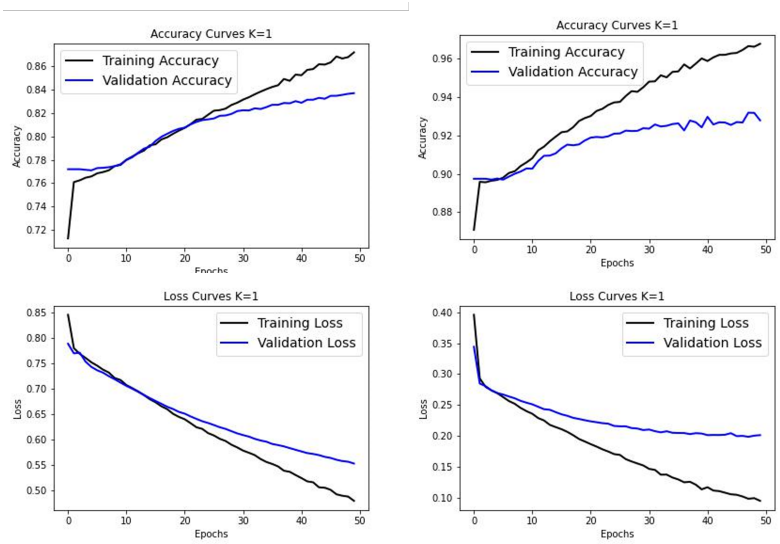

Figure 7.41: EfficientNet learning curves with no transfer learning and without augmentations. Left: Mixed samples dataset. Right: Anuraset.

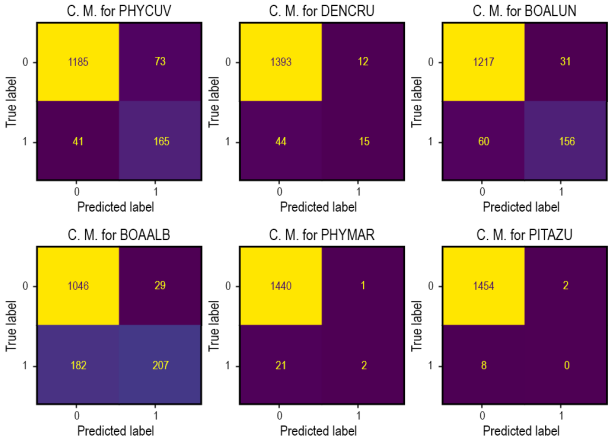

Figure 7.42: Confusion Matrix Anuraset(ResNet)

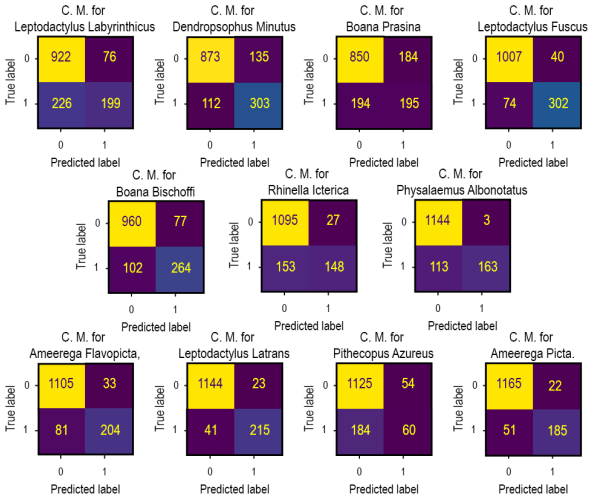

Figure 7.43: Confusion Matrix Mixed samples(ResNet)

the results got when using augmentations and transfer learning. Furthermore, when mixed samples were introduced, the performance of EfficientNet was even worse compared to the other experiments. When compared to the other two models, EfficientNet's performance without augmentations and transfer learning remained the lowest. Although the Precision of EfficientNet was better than the one from VGG, its recall was significantly lower in comparison.

Both VGG and ResNet showed superior performance when trained without augmentations and

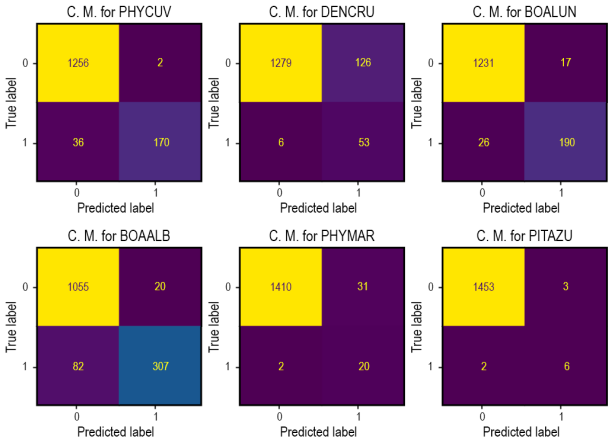

Figure 7.44: Confusion Matrix Anuraset(VGG)

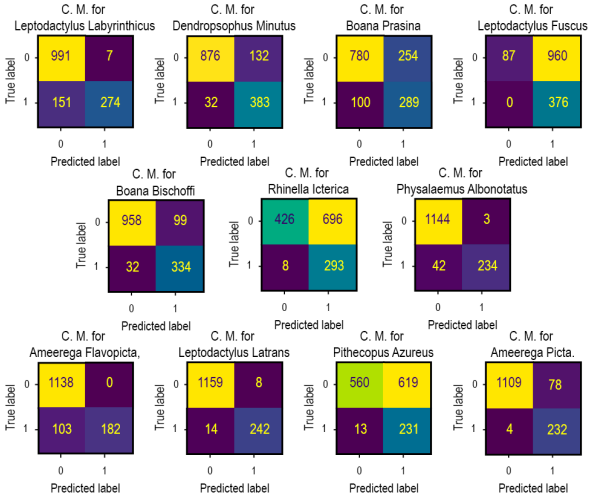

Figure 7.45: Confusion Matrix Mixed samples(VGG)

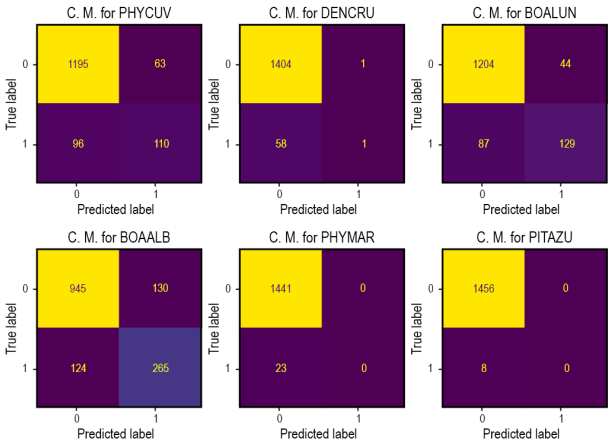

Figure 7.46: Confusion Matrix Anuraset(EfficientNet)

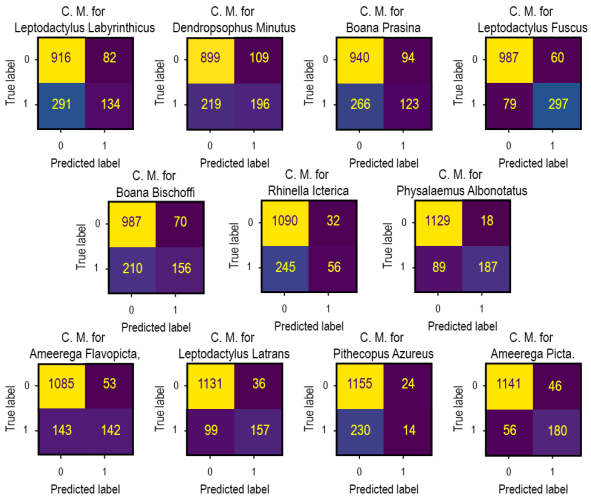

Figure 7.47: Confusion Matrix Mixed samples (EfficientNet)

transfer learning, compared to when both techniques were employed, particularly when using the Anuraset dataset. In challenging classes such as "DENCRU" and "PHYMAR," these models could identify them to some extent, with slightly higher values than 0. Among the two models, VGG improved results in Anuraset, particularly in terms of recall, which also improved when trained with the mixed samples dataset. With the mixed samples dataset, both models performed similarly, with ResNet having better precision and VGG having better recall. Therefore, VGG can be considered the superior model without augmentations and transfer learning.

It is important to note that the better results obtained from VGG and ResNet compared to EfficientNet could be due to overfitting, suggesting that these CNNs may not perform well when tested with other datasets. Additionally, it is worth mentioning that the results obtained in this set of experiments were not as impressive as those got in Section 7.2.3, where augmentations were applied and all models achieved remarkable evaluation metrics.

In conclusion, upon close examination of the metrics for each class, it was observed that the classes previously mentioned ("DENCRU", "PITAZU", "PHYMAR", "Boana Prasina", and "Pithecopus Azureus") consistently exhibited smaller metrics compared to the other classes. This discrepancy can be attributed to the dataset's imbalance or external factors, such as distinct frequency patterns that are challenging to identify without specific conditions. However, even with these challenges, VGG outperformed ResNet and EfficientNet in identifying and classifying these classes, as demonstrated in the experiments conducted with the specified conditions.

No Augmentations and No Transfer Learning																				
Model	ResNet50						VGG19						EfficienNetB0							
Dataset	Anuraset			Mixed samples			Anuraset			Mixed samples			Anuraset			Mixed samples				
Metric	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score		
0	0,793 ± 0,092	0,653 ± 0,113	0,705 ± 0,036	0,61 ± 0,086	0,583 ± 0,098	0,585 ± 0,017	0,806 ± 0,369	0,876 ± 0,087	0,773 ± 0,29	0,68 ± 0,26	0,824 ± 0,137	0,701 ± 0,114	0,675 ± 0,028	0,53 ± 0,017	0,594 ± 0,014	0,609 ± 0,033	0,327 ± 0,032	0,425 ± 0,031		
1	0,496 ± 0,201	0,334 ± 0,216	0,32 ± 0,092	0,698 ± 0,08	0,687 ± 0,088	0,685 ± 0,021	0,518 ± 0,361	0,866 ± 0,145	0,557 ± 0,28	0,812 ± 0,263	0,737 ± 0,216	0,714 ± 0,12	0,1 ± 0,224	0,003 ± 0,008	0,007 ± 0,015	0,637 ± 0,013	0,485 ± 0,048	0,55 ± 0,035		
2	0,75 ± 0,131	0,766 ± 0,077	0,747 ± 0,054	0,58 ± 0,103	0,446 ± 0,061	0,495 ± 0,013	0,915 ± 0,034	0,882 ± 0,04	0,897 ± 0,007	0,626 ± 0,227	0,611 ± 0,2	0,561 ± 0,078	0,695 ± 0,039	0,598 ± 0,037	0,642 ± 0,03	0,615 ± 0,075	0,304 ± 0,038	0,406 ± 0,047		
3	0,813 ± 0,051	0,683 ± 0,101	0,736 ± 0,046	0,875 ± 0,049	0,813 ± 0,035	0,842 ± 0,008	0,872 ± 0,117	0,852 ± 0,081	0,852 ± 0,034	0,703 ± 0,3	0,934 ± 0,085	0,759 ± 0,198	0,731 ± 0,05	0,65 ± 0,022	0,687 ± 0,013	0,827 ± 0,026	0,744 ± 0,032	0,783 ± 0,025		
4	0,383 ± 0,286	0,061 ± 0,039	0,103 ± 0,065	0,778 ± 0,109	0,688 ± 0,107	0,718 ± 0,028	0,396 ± 0,237	0,691 ± 0,173	0,445 ± 0,159	0,728 ± 0,275	0,831 ± 0,17	0,724 ± 0,115	0 ± 0	0 ± 0	0 ± 0	0,685 ± 0,032	0,454 ± 0,053	0,544 ± 0,038		
5	0,2 ± 0,447	0,025 ± 0,056	0,044 ± 0,099	0,793 ± 0,056	0,596 ± 0,078	0,675 ± 0,036	0,676 ± 0,205	0,5 ± 0,25	0,516 ± 0,196	0,693 ± 0,257	0,855 ± 0,103	0,727 ± 0,158	0 ± 0	0 ± 0	0 ± 0	0,636 ± 0,019	0,213 ± 0,02	0,319 ± 0,024		
6				0,904 ± 0,091	0,686 ± 0,112	0,77 ± 0,059				0,821 ± 0,24	0,872 ± 0,073	0,822 ± 0,115				0,909 ± 0,017	0,649 ± 0,026	0,757 ± 0,022		
7				0,851 ± 0,061	0,68 ± 0,079	0,751 ± 0,037				0,964 ± 0,049	0,744 ± 0,118	0,833 ± 0,059				0,72 ± 0,018	0,496 ± 0,033	0,587 ± 0,024		
8				0,947 ± 0,028	0,781 ± 0,056	0,855 ± 0,026				0,865 ± 0,177	0,956 ± 0,025	0,898 ± 0,107				0,827 ± 0,032	0,644 ± 0,038	0,723 ± 0,02		
9				0,507 ± 0,081	0,286 ± 0,064	0,356 ± 0,025				0,272 ± 0,146	0,908 ± 0,161	0,387 ± 0,112				0,426 ± 0,056	0,051 ± 0,012	0,091 ± 0,021		
10				0,906 ± 0,045	0,745 ± 0,077	0,814 ± 0,037				0,74 ± 0,318	0,969 ± 0,026	0,795 ± 0,271				0,787 ± 0,035	0,704 ± 0,038	0,743 ± 0,029		
Micro Avg	0,743 ± 0,054	0,652 ± 0,063	0,691 ± 0,02	0,744 ± 0,027	0,637 ± 0,02	0,686 ± 0,005	0,707 ± 0,255	0,858 ± 0,044	0,749 ± 0,195	0,551 ± 0,068	0,828 ± 0,027	0,659 ± 0,046	0,706 ± 0,027	0,546 ± 0,015	0,615 ± 0,006	0,725 ± 0,009	0,457 ± 0,005	0,561 ± 0,005		
Macro Avg	0,572 ± 0,128	0,42 ± 0,036	0,442 ± 0,015	0,768 ± 0,013	0,636 ± 0,018	0,686 ± 0,008	0,697 ± 0,153	0,778 ± 0,054	0,673 ± 0,115	0,719 ± 0,066	0,84 ± 0,028	0,72 ± 0,04	0,367 ± 0,034	0,297 ± 0,009	0,322 ± 0,005	0,698 ± 0,007	0,461 ± 0,005	0,539 ± 0,004		
Weighted Avg	0,756 ± 0,052	0,652 ± 0,063	0,682 ± 0,022	0,756 ± 0,016	0,637 ± 0,02	0,682 ± 0,007	0,83 ± 0,1	0,858 ± 0,044	0,812 ± 0,088	0,721 ± 0,062	0,828 ± 0,027	0,717 ± 0,035	0,643 ± 0,02	0,546 ± 0,015	0,587 ± 0,006	0,693 ± 0,007	0,457 ± 0,005	0,537 ± 0,004		
Samples Avg	0,251 ± 0,013	0,247 ± 0,022	0,242 ± 0,017	0,787 ± 0,022	0,698 ± 0,018	0,706 ± 0,005	0,307 ± 0,054	0,341 ± 0,019	0,315 ± 0,038	0,561 ± 0,08	0,861 ± 0,025	0,654 ± 0,055	0,194 ± 0,007	0,197 ± 0,007	0,189 ± 0,006	0,727 ± 0,007	0,519 ± 0,009	0,568 ± 0,009		
	Average Accuracy		0,940	Average Accuracy		0,836	Average Accuracy		0,920	Average Accuracy		0,800	Average Accuracy		0,930	Average Accuracy		0,834		
	Total Average Accuracy			0,888			Total Average Accuracy			0,860			Total Average Accuracy			0,882				

Table 7.9: Evaluation metrics comparison.

The following paragraph will analyze the performance of the models without augmentations and transfer learning in terms of training time and memory utilization (Table 7.10). As expected, the training times are lower compared to the ones observed when augmentations were applied. However, when compared to the training time in Section 7.2.2, there seems to be a slight increase, except for ResNet when trained with Anuraset. This discrepancy can be attributed to the fact that the model is learning from scratch instead of utilizing transfer learning, as suggested in Section 7.2.3. Moreover, out of the three models, ResNet exhibited the shortest training time in this set of experiments.

When it comes to memory utilization, the models do not exhibit a consistent pattern of significant decrease or increase compared to the other experiments. There is also no discernible trend indicating that one model consistently requires more memory than the others. For instance, when no transfer learning or augmentations are employed, ResNet stands out as the model with the lowest memory usage among the three.

	ResNet50		**VGG19**		**EfficientNetB0**	
	Anuraset	**Mixed Samples**	**Anuraset**	**Mixed Samples**	**Anuraset**	**Mixed Samples**
Avg. Training Time	45,28803	45,7379268	47,01797	46,46665876	46,5652	46,2382
Avg. Memory utilization	0,855293	0,849600098	0,868402	0,893605957	0,865574	0,8718

Table 7.10: Training time and memory utilization of each model.

The final evaluation of this model revolves around the observation of the convergence of each model, without the utilization of data augmentation and transfer learning. These observations are

depicted in Figure 7.48 and Figure 7.49. EfficientNet consistently performs the best among the models, starting from a high point and maintaining stability throughout the epochs. Although the accuracy growth is slow initially, it becomes more pronounced towards the end of the epochs.

VGG starts with a lower accuracy value compared to the other models, but quickly increases its value. However, similar to the graphs shown in Section 7.2.3, its behavior with the validation set can be quite noisy, making it difficult to observe clear convergence.

ResNet also exhibits a pronounced growth similar to VGG, but its curve is more stable in the validation set compared to this model. Interestingly, both ResNet and EfficientNet show a more significant growth with the Mixed Samples compared to Anuraset. This could be because of the complexity of the dataset affecting their performance.

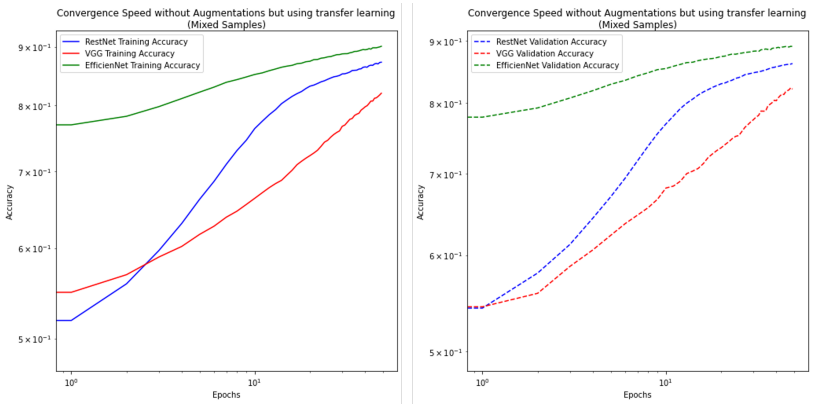

Figure 7.48: Convergence speed of models trained with mixed samples dataset.

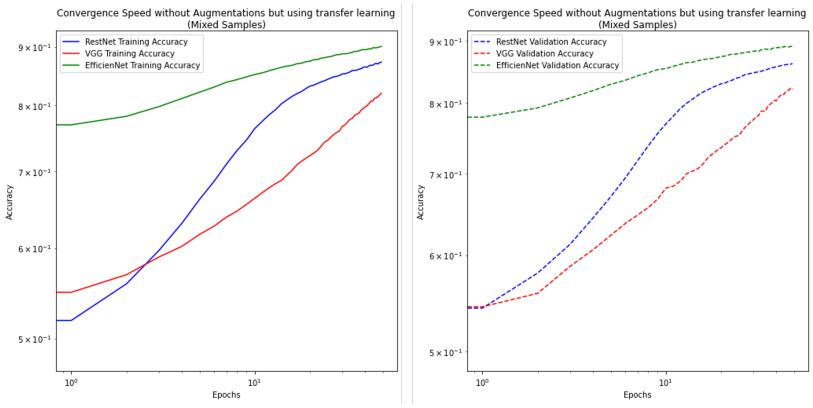

Figure 7.49: Convergence speed of models trained with Anuraset.

Chapter 8

Conclusions

The present investigation aimed to test the effectiveness of data augmentation and transfer learning in classifying multi-label spectrograms corresponding to calls from different anuran species. To achieve this objective, the investigation process was divided into several stages, as outlined in Chapter 6: data collection, data augmentation model training, and final model evaluation.

During the data collection stage, two datasets were selected and processed: INCT41 from Anuraset and a dataset of mixed samples. These datasets had different characteristics; one was more imbalanced than the other, and one had more classes. This variation was intended to illustrate the effectiveness of data augmentation and transfer learning in different scenarios.

The next stage involved executing data augmentation. The algorithms described in Section 6.5 were applied to alleviate imbalances in certain datasets and introduce diverse distortions. This was done to enhance the variability of the samples and improve the models' classification capabilities.

After applying data augmentation, three models were selected for testing their capacity to classify the multi-label spectrogram images. The chosen models were ResNet50, VGG19, and EfficientNetB0. Various experiments were conducted using these models, involving transfer learning, data augmentation, a combination of both techniques, or none. The performance of these models was measured using the techniques described in Section 5.2.3.4.

After obtaining the results from Chapter 7, it is possible to conclude that data augmentation can help to add different distortions to generate new samples and also assess to mitigate imbalance in certain cases depending on the level of imbalance and the number of samples shared between classes. Furthermore, while data augmentation cannot guarantee complete reduction of the imbalance, it can aid in increasing the sample count for classes with limited samples. This enhances the likelihood of models learning and accurately identifying their patterns.

After the application of data augmentation and training the models, the results suggested EfficientNet was the model that benefitted the most from the use of data augmentation and transfer learning, thus presented the highest metrics from the three models even with challenging classes such as the "PITAZU" class from Anuraset that achieved approximately an F1-score of 0.821 while also achieving high precision and recall between the three models. With the mixed samples dataset, EfficientNet also had a slightly better metrics than the rest when using data augmentation and transfer learning. The subsequent model that typically succeeded EfficientNet in terms of performance varied based on the dataset. VGG exhibited superior metrics compared to ResNet when considering Anuraset, suggesting it is less sensible to imbalances than ResNet. In contrast, ResNet exhibited superior performance in the Mixed samples dataset compared to VGG. This observation implies that ResNet may be a more resilient model in scenarios where the imbalance is not significant and the dataset is more complex.

Through the experiments, Anuraset was the dataset that also benefitted the most from using augmentations. The changes between experiments were considerable when applied augmentations than where augmentations were not used, especially in classes such as “PHYMAR” and “PITAZU” which without augmentations are practicable unrecognizable by the models but with augmentations could achieve higher metrics. In cases similar to the one observed in the mixed samples dataset, where the imbalance is not appreciably marked, the influence of data augmentation and transfer learning is more modest. In general, the effect of data augmentation was more visible in recall metric than in precision, thus it was the metric that increased when data augmentation was available. This implies that data augmentation improves the capacity to detect the presence of significant samples for each class. Additionally, the application of data augmentation aids in the identification of difficult classes that pose a challenge not solely due to class imbalance but also because of external factors, such as background noise. This phenomenon was observed in instances involving "Boana Prasina" and "Phithecopus azureus" from the mixed samples, where a marginal improvement in recall and even precision was observed in comparison to the absence of augmentations. Then it confirms the benefits of data augmentations, even when used in multi-label datasets.

The influence of transfer learning is more visible in the learning curves by adding more stability during training and also controlling possible signals of overfitting, which can be even improved by the use of data augmentation. As suggested in the results of Section 7.2, EfficientNet is the architecture which its metrics improved with transfer learning that without it. With VGG and ResNet the results were better when training the model from scratch than when applying transfer learning. The results from these two models could be further explored to check that were not caused by overfitting, thus as suggested by the learning curves, the number of parameters and the architecture of these two models they are more prone than EfficientNet to suffer from overfitting.

Additionally, it is crucial to consider other factors when using a model, including the fact that the number of parameters has no impact on the computational weight or training time of the model. While there exist alternative options that can be evaluated for their potential to attain comparable metrics, such as variants of EfficientNet like EfficientNetV2, which could demonstrate superior metrics even with demanding datasets while requiring minimal computational resources. Through the extra performance metrics, it was also possible to observe that augmentations and transfer learning helped the model to have a more stable training and also to converge faster than without augmentations, this was especially visible in some training and validation sets where the curves were noisy or had late signals of convergence.

In summary, EfficientNet shows great promise as a model for multi-label classification, especially when combined with transfer learning and data augmentation techniques. Data augmentation can be particularly effective in datasets with classes that have a limited number of samples, as it improves the model’s ability to identify these classes. However, when the class imbalance is not significant, the improvement in classification metrics is not substantial. Furthermore, the use of transfer learning and data augmentation can also help reduce overfitting and enhance model convergence and stability.

However, during the investigation, an experiment suggested that training the model from scratch could be beneficial, although further research is needed to verify this and determine if it is not due

to overfitting. Also for future work, it would be worthwhile to consider different models depending on the computational limitations of the context. For example, EfficientNetV2 could be a suitable alternative in situations where computational resources are limited.

Chapter 9

Recommendations

The results presented in Chapter 7 exhibit promising result for data augmentation associated with transfer learning. Nevertheless, it is possible to offer suggestions and propose additional experiments or modifications that can further validate the impact of augmentation with transfer learning or the utilization of these techniques. Alternatively, corrections can be made to enhance precision or acknowledge the specific circumstances in which these techniques may be applicable. The suggestions, observations and possible further work will be described in this section. The initial focus will be on the recommendations pertaining to data augmentation, followed by the discussion of recommendations concerning the employed models and transfer learning.

Regarding Data Augmentation, and form what was observed in Section 7.1, Data Augmentation can be effective in diversifying the data in a dataset although it must be considered certain conditions. As observed in Section 7.1.3, careful consideration must be given to the dataset's imbalance, and thorough revisions should be made when conducting experiments on a multi-label dataset. Hence, the use of data augmentation to mitigate imbalance in these types of datasets can be challenging. As it was observed in Anuraset (Section 7.1.3), the imbalance in the dataset was considerably marked and some classes with fewest samples shared samples with most occurrences, which limited the capacity of data augmentation to improve the differences between classes. Hence, it becomes crucial to consider the disparities in classes and the amount of shared samples among them, particularly when comparing the most frequent classes with the rarest ones. By doing so, it is possible to assess the potential impact of data augmentation on the dataset. Despite the presence of significant class imbalance, augmenting the samples with few occurrences can enhance the model's ability to recognize labels as the "PITAZU" class in Anuraset mentioned in Section Section 7.2.1.

Other aspect that is important to inspect is the condition of the data, if the samples even after data-preprocessing are difficult to identify, could also complicate the process of classification and data augmentation could not contribute significantly to improve the identification of the labels in some samples. This was a case observed with classes such as the "DENCRU" from Anuraset, and "Boana Prasina" and "Phithecopus azureus" from the data set of mixed samples, where the frequency information was obstructed from background noise or other species' calls. While it is indeed crucial to generate challenging scenarios to enhance the model's ability to identify data even in difficult conditions and prepare it for unforeseen data, it is important to acknowledge that if the frequency patterns in the data are practically unrecognizable, it might harm the model rather than benefit it.

Another way to challenge the models and assess the effectiveness of data augmentation and transfer learning is to utilize datasets that exhibit more imbalance and complexity in terms of the number of classes. For instance, Anuraset offers additional sets with varied recording locations,

each containing different species, as shown in Figure 9.1. The distribution of data in the "INCT17" subset from Anuraset demonstrates an increase in the number of classes and a noticeable imbalance. By applying data augmentation and transfer learning techniques, it is possible to determine if it is feasible to achieve favorable evaluation metrics, even for classes with limited occurrences.

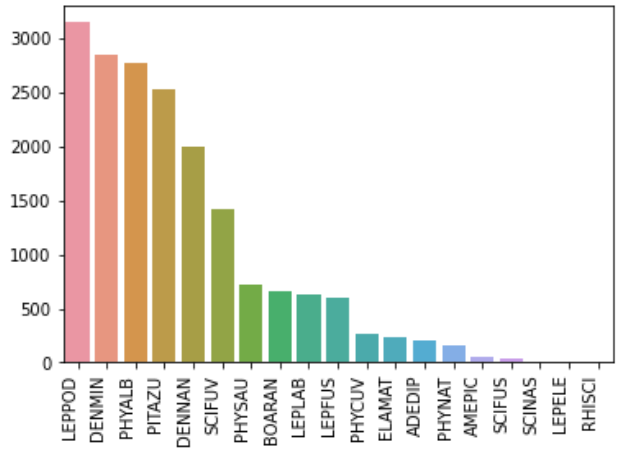

Figure 9.1: Data distribution INCT17 dataset.

With respect to model selection and transfer learning, there are other suggestions that can be made for future work. An advised approach is to explore alternative models and model configurations in order to assess the potential increase in certain metrics or identify more suitable models for multi-label classification. Then the changes in configuration can comprehend minor changes, such as changing the number of additional layers, activation functions optimizer. For example, it will be possible to test Stochastic Gradient Descent or variations of it to check its utility in multi-label problems.

Another potential change could be to explore alternative CNN architectures or different versions of existing architectures, incorporating transfer learning to assess their performance compared to EfficientNet. It would also be interesting to investigate these architectures with varying focuses or characteristics. For instance, as mentioned in Section 7.2.1, testing different variants of EfficientNet, like EfficientNetV2, would allow us to evaluate its computational efficiency while determining its effectiveness with multi-label datasets. The objective would be to analyze different attributes of the models under certain considerations and recommend their usage in specific scenarios, such as when computational resources are limited.

A final and more radical approach will be to use other types of machine learning architectures and apply transfer learning if possible. As suggested by Stowell [6], it may be feasible to employ various architectures, such as RNNs or TCNs, in conjunction with components like Attention Networks, and even incorporate transfer learning. By testing these combinations, it can be determined whether they can yield significantly improved performance and establish themselves as more suitable architectures for spectrogram multi-label classification.

Another approach would be to circumvent the use of transfer learning and instead fully fine-tune the model, as suggested in Section 7.2.3. This approach has shown remarkable metrics, even for the most challenging classes. To further evaluate the effectiveness of frozen layers and transfer learning compared to fine-tuning, it would be beneficial to employ other complex and complementary metrics. The evaluation of metrics in this study should extend beyond learning curves and confusion matrices to accurately identify the occurrence of overfitting. Moreover, training a model from scratch might yield better results in the classification of multi-label spectrograms. Additional recommended metrics that can be explored include the Brier Score and the Matthews correlation coefficient (MCC) [102] or variations of the F1-score such as the Fbeta-score.

The final recommendation concerning the utilization of metrics will rely on the observations made in Section Section 7.2.1 and subsequent experiments, which consistently demonstrated higher accuracy values in the learning curves compared to the evaluation metrics. This discrepancy was attributed to the averaging method employed in "binary accuracy", as mentioned in the aforementioned section. An alternative option, considering the model's imbalance, would be to employ "weighted binary accuracy" and "weighted binary cross entropy" for more accurate values. This would necessitate the calculation of weights for each class and the development of a custom function to refine the model's precision with respect to each class.

Another potential change that is unrelated to metrics, data augmentation, and transfer learning involves altering the representation of acoustic features. Instead of relying solely on spectrograms, alternative representations such as MFCCs or even waveforms can be explored. The aim is to determine whether utilizing these different representations can lead to improved performance. It may even be beneficial to combine various types of acoustic data representations with different model architectures.

Chapter 10

Appendix

Appendix 1 - Groups of augmentations

1st Group	2nd Group	3rd Group	4th Group	5th Group	6th Group	7th Group	8th Group
Harmonic Distortion	Clipping	Pitch shift	Sound shuffle	Time Stretch	AWGN	Echo	Rolling
Red Filter	Wow resampling	Sound Mix	VTLP	Shuffled Mixed	Delay	DCR	Time Masking
Flip	green filter	Color Reduction 128	EMDA	Color Reduction 32	Negative Positive	Bright and saturation	Time Swapping

Table 10.1: Groups of augmentations.

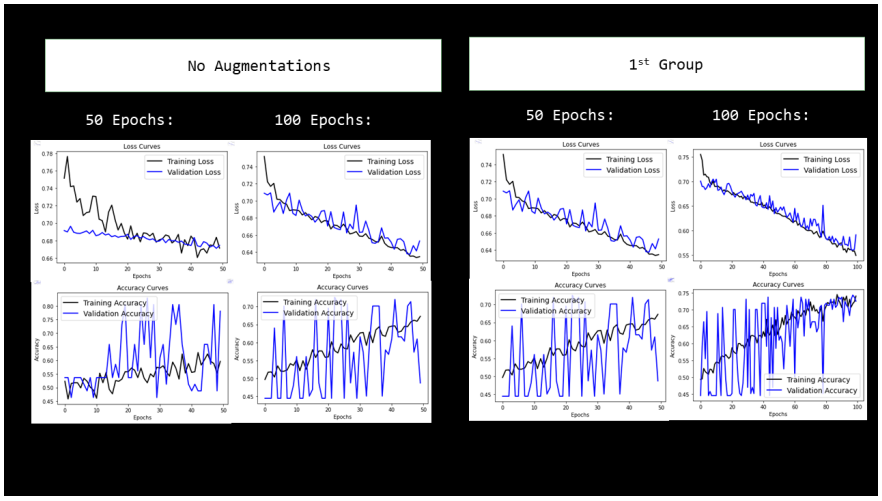

Figure 10.1: Classification results from No augmentations and 1st Group

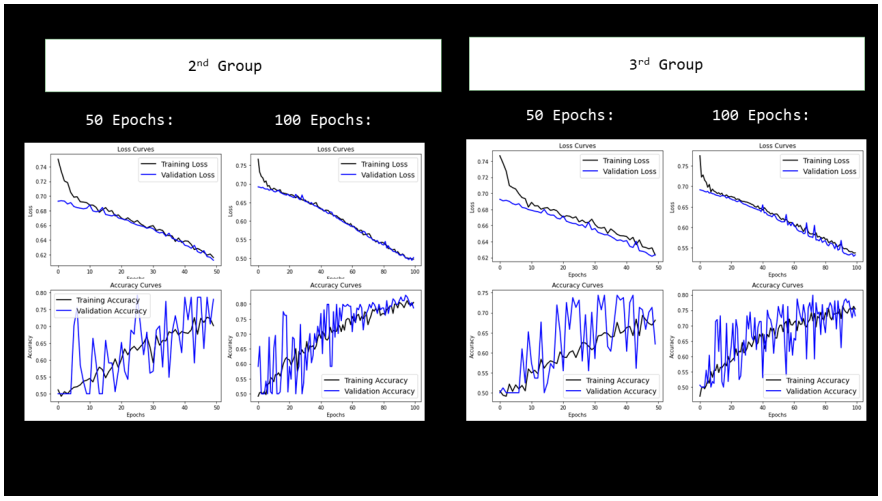

Figure 10.2: Classification results from 2nd Group and 3rd Group

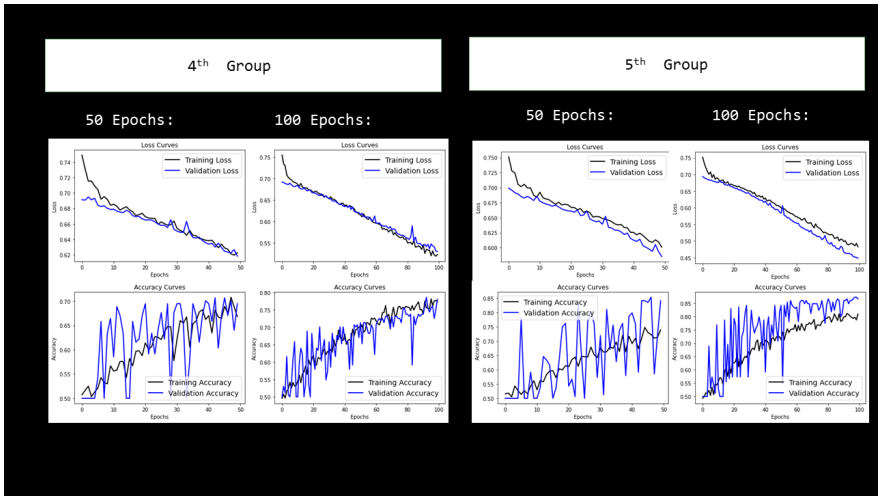

Figure 10.3: Classification results from 4th Group and 5th Group

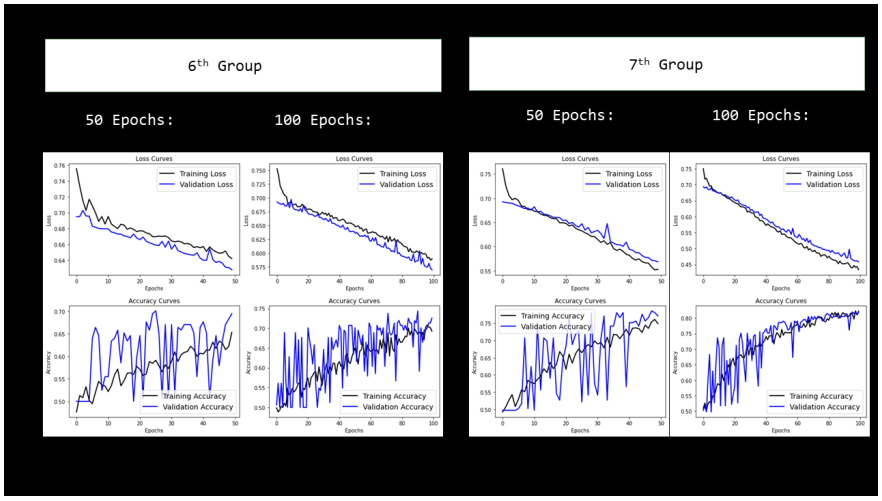

Figure 10.4: Classification results from 6th Group and 7th Group

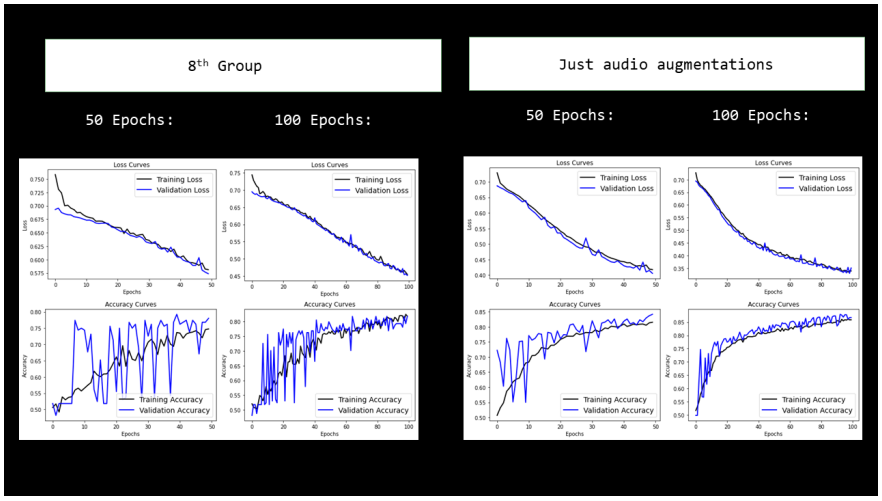

Figure 10.5: Classification results from 8th Group and just audio augmentations

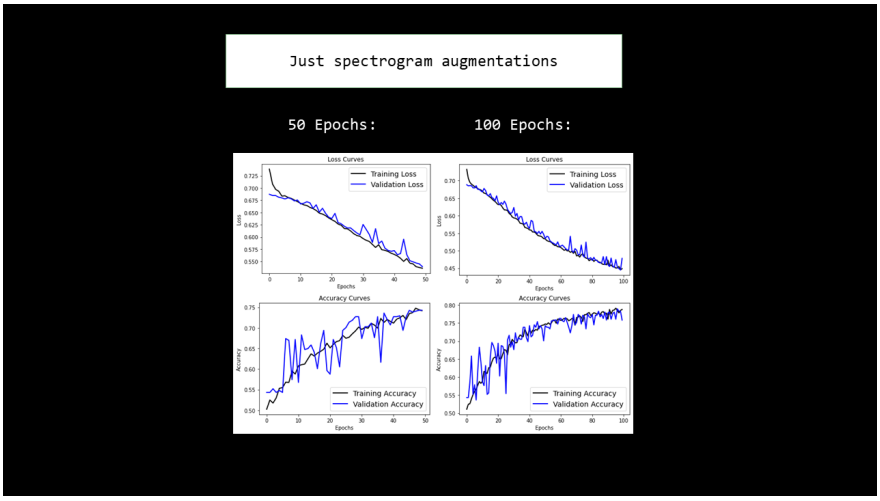

Figure 10.6: Classification results just spectrogram augmentations

Experiments	Num. Epochs	Classes	Precision	Recall	F1 score	Accuracy
No Augmentations	50 Epochs	0	1	0,56	0,71	0,77
		1	0,67	1	0,8	
	100 Epochs	0	0,85	0,61	0,71	0,74
		1	0,67	0,88	0,76	
1st Group	50 Epochs	0	0,51	0,97	0,67	0,53
		1	0,83	0,12	0,21	
	100 Epochs	0	0,67	0,86	0,76	0,72
		1	0,82	0,59	0,69	
2nd Group	50 Epochs	0	0,9	0,74	0,81	0,83
		1	0,78	0,92	0,84	
	100 Epochs	0	0,78	0,9	0,83	0,82
		1	0,88	0,74	0,8	
3rd Group	50 Epochs	0	0,57	0,91	0,7	0,61
		1	0,78	0,31	0,45	
	100 Epochs	0	0,72	0,88	0,79	0,76
		1	0,84	0,65	0,73	
4th Group	50 Epochs	0	0,72	0,74	0,73	0,72
		1	0,73	0,71	0,72	
	100 Epochs	0	0,81	0,77	0,79	0,79
		1	0,78	0,81	0,8	
5th Group	50 Epochs	0	0,88	0,76	0,81	0,82
		1	0,78	0,89	0,83	
	100 Epochs	0	0,9	0,79	0,84	0,85
		1	0,81	0,91	0,85	
6th Group	50 Epochs	0	0,88	0,54	0,67	0,73
		1	0,67	0,92	0,77	
	100 Epochs	0	0,87	0,57	0,69	0,74
		1	0,68	0,92	0,78	
7th Group	50 Epochs	0	0,76	0,79	0,78	0,78
		1	0,79	0,77	0,78	
	100 Epochs	0	0,91	0,77	0,83	0,85
		1	0,81	0,93	0,86	
8th Group	50 Epochs	0	0,78	0,72	0,75	0,76
		1	0,73	0,79	0,76	
	100 Epochs	0	0,81	0,74	0,78	0,78
		1	0,76	0,83	0,79	
9th experiment	50 Epochs	0	0,87	0,79	0,83	0,83
		1	0,8	0,88	0,84	
	100 Epochs	0	0,86	0,91	0,89	0,88
		1	0,9	0,85	0,88	
10 experiment	50 Epochs	0	0,79	0,75	0,77	0,78
		1	0,77	0,8	0,78	
	100 Epochs	0	0,93	0,62	0,74	0,79
		1	0,72	0,96	0,82	

Table 10.2: Metrics results from each group of augmentations.

Appendix 2 – Learning Curves and Confusion Matrices from experiments with augmentations and transfer learning

10.0.1 Learning curves.

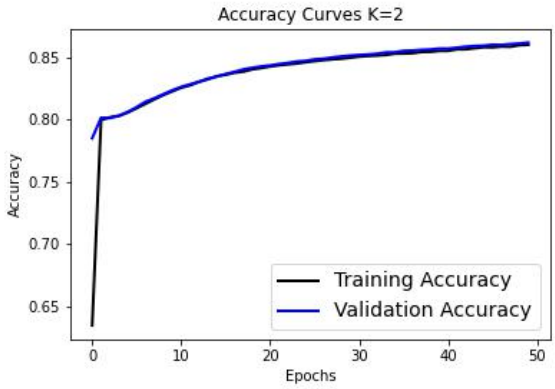

Figure 10.7: Accuracy Curve Fold 2 (ResNet)

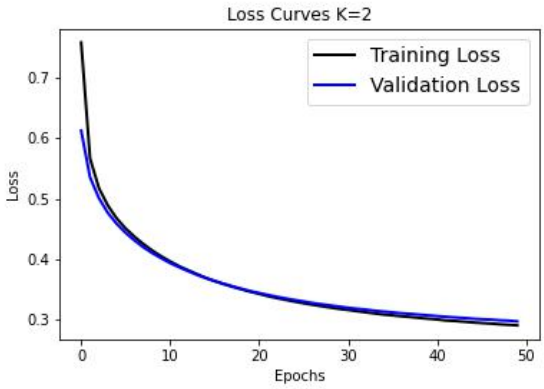

Figure 10.8: Loss Curve Fold 2 (ResNet)

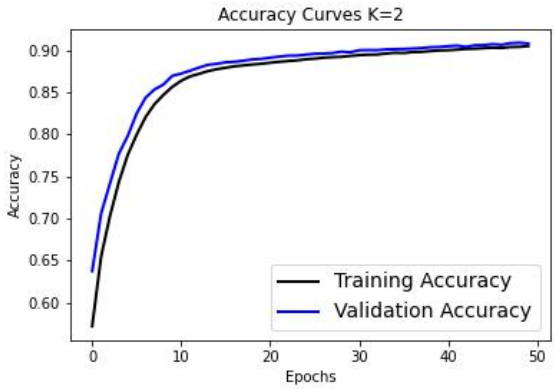

Figure 10.9: Accuracy Curve Fold 2 (VGG)

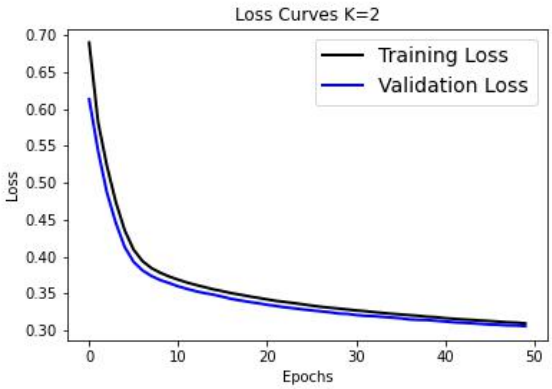

Figure 10.10: Loss Curve Fold 2 (VGG)

10.0.2 Confusion Matrices

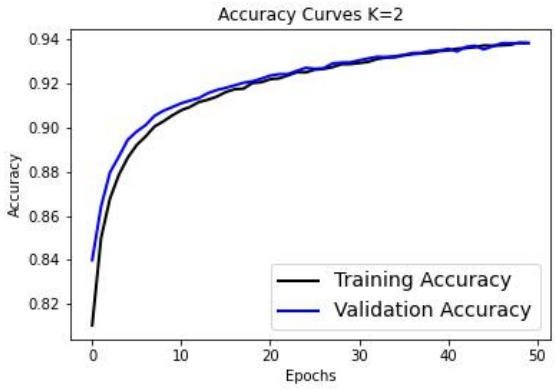

Figure 10.11: Accuracy Curve Fold 2 (EfficientNet)

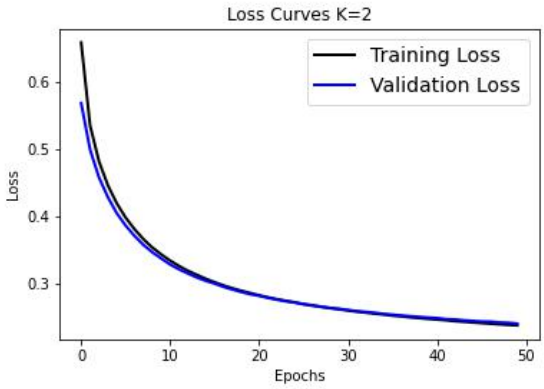

Figure 10.12: Loss Curve Fold 2 (EfficientNet)

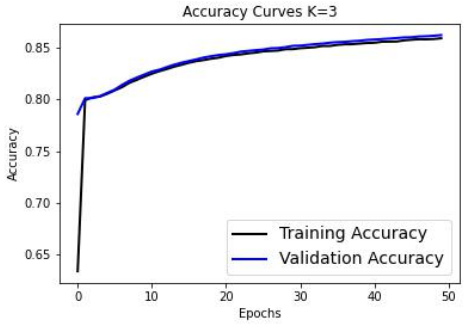

Figure 10.13: Accuracy Curve Fold 3 (ResNet)

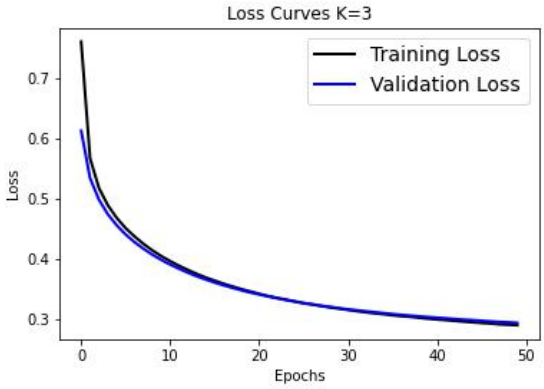

Figure 10.14: Loss Curve Fold 3 (ResNet)

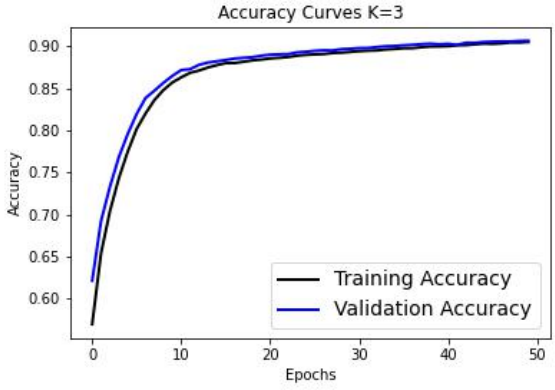

Figure 10.15: Accuracy Curve Fold 3 (VGG)

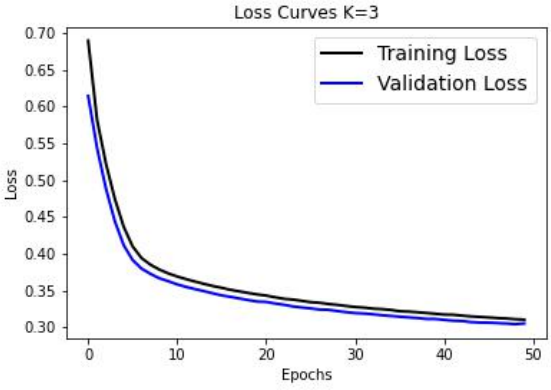

Figure 10.16: Loss Curve Fold 3 (VGG)

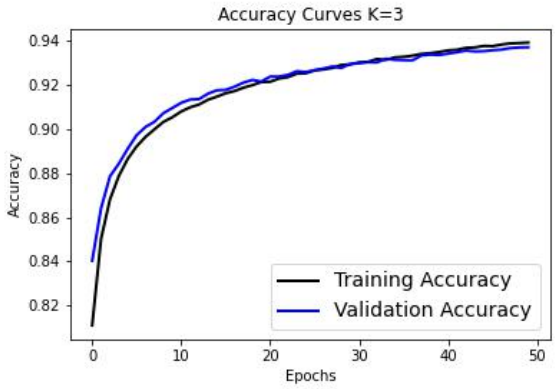

Figure 10.17: Accuracy Curve Fold 3 (EfficientNet)

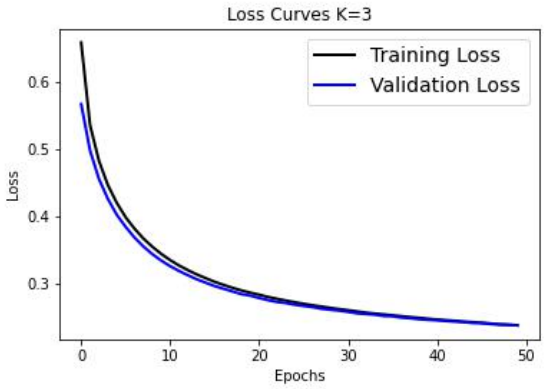

Figure 10.18: Loss Curve Fold 3 (EfficientNet)

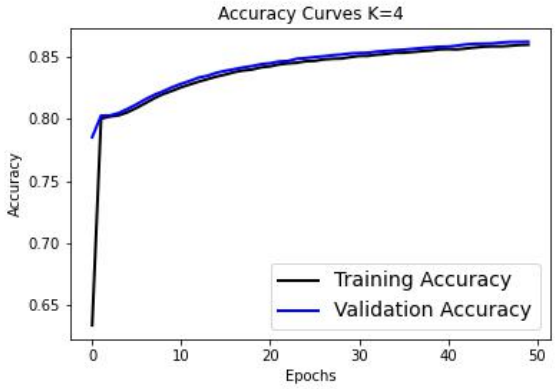

Figure 10.19: Accuracy Curve Fold 4 (ResNet)

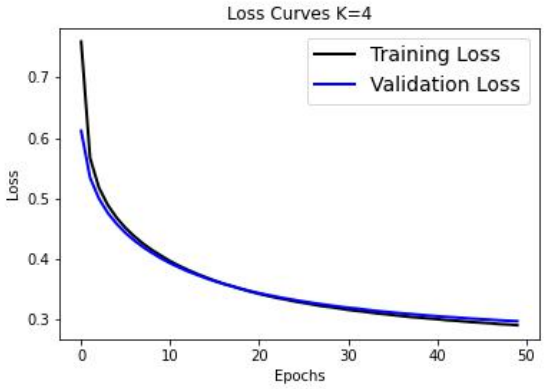

Figure 10.20: Loss Curve Fold 4 (ResNet)

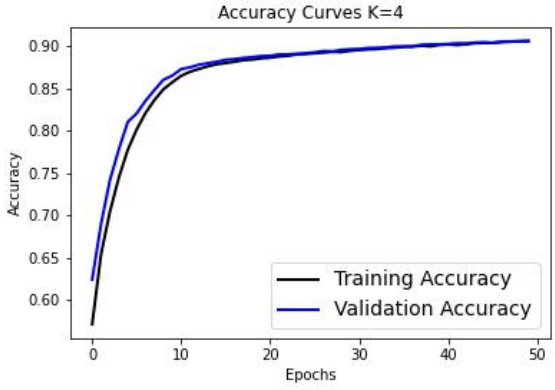

Figure 10.21: Accuracy Curve Fold 4 (VGG)

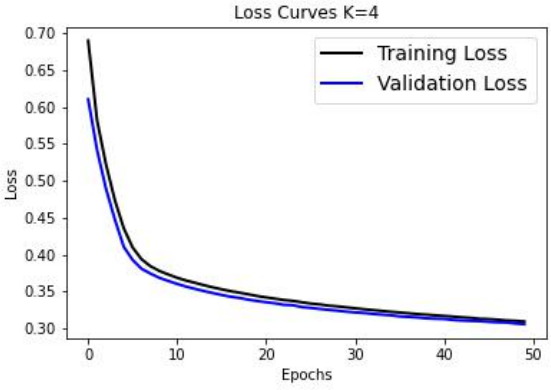

Figure 10.22: Loss Curve Fold 4 (VGG)

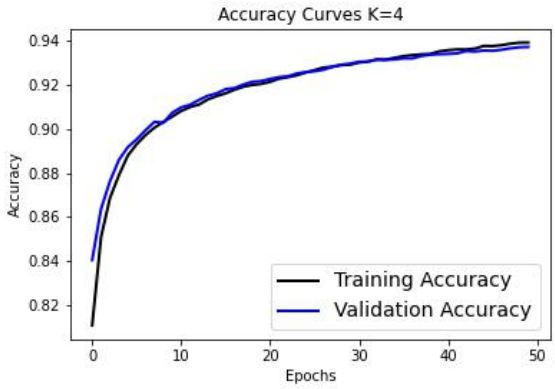

Figure 10.23: Accuracy Curve Fold 4 (EfficientNet)

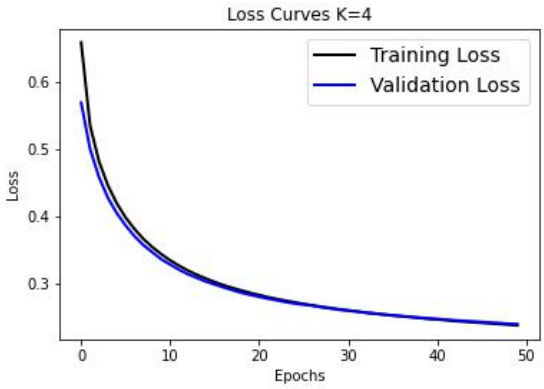

Figure 10.24: Loss Curve Fold 4 (EfficientNet)

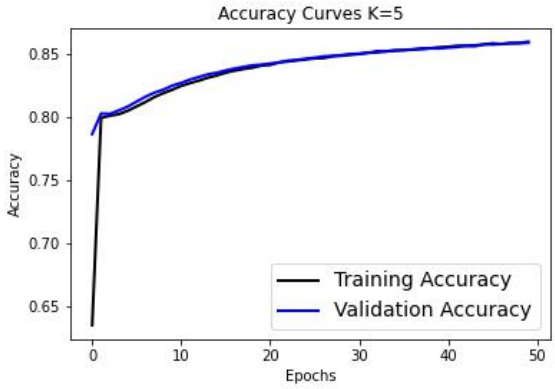

Figure 10.25: Accuracy Curve Fold 5 (ResNet)

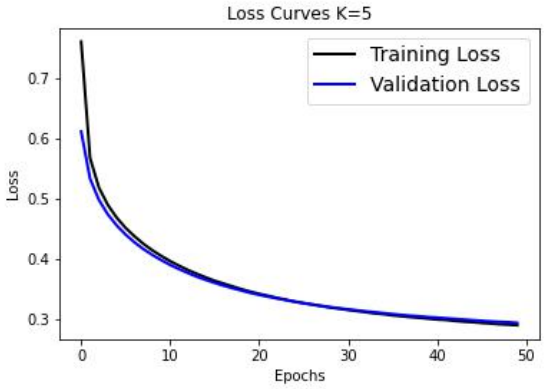

Figure 10.26: Loss Curve Fold 5 (ResNet)

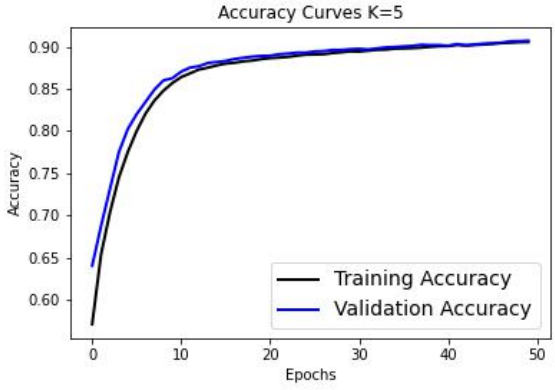

Figure 10.27: Accuracy Curve Fold 5 (VGG)

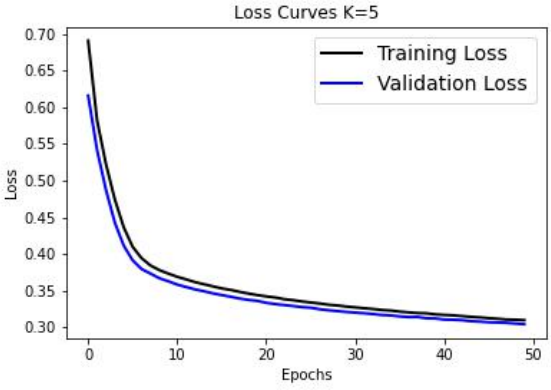

Figure 10.28: Loss Curve Fold 5 (VGG)

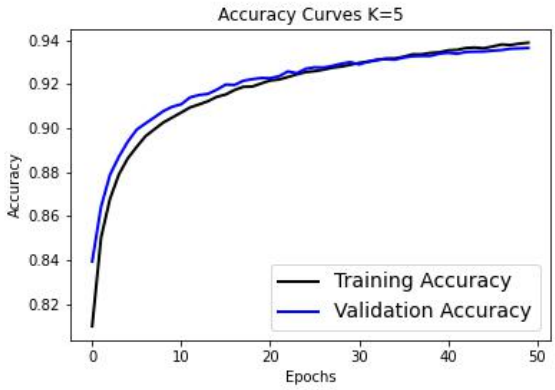

Figure 10.29: Accuracy Curve Fold 5 (EfficientNet)

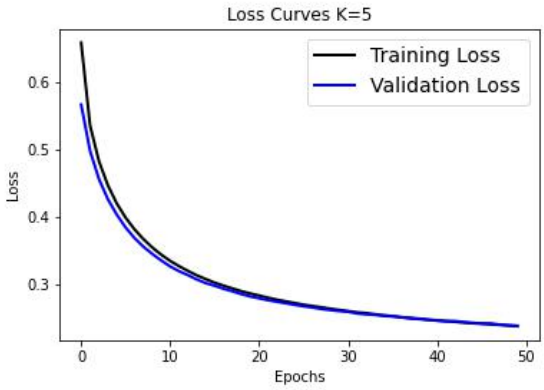

Figure 10.30: Loss Curve Fold 5 (EfficientNet)

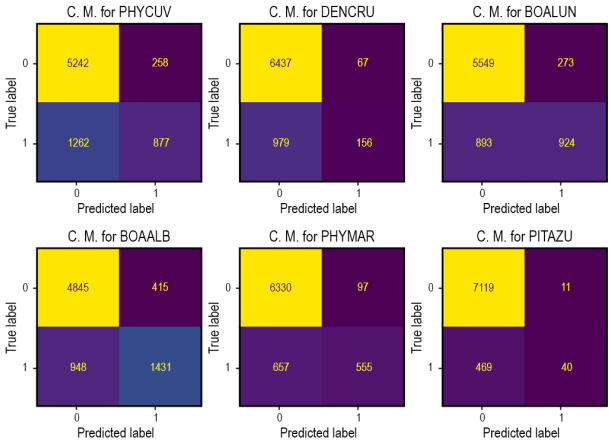

Figure 10.31: Confusion Matrix Anuraset Fold 2 (ResNet)

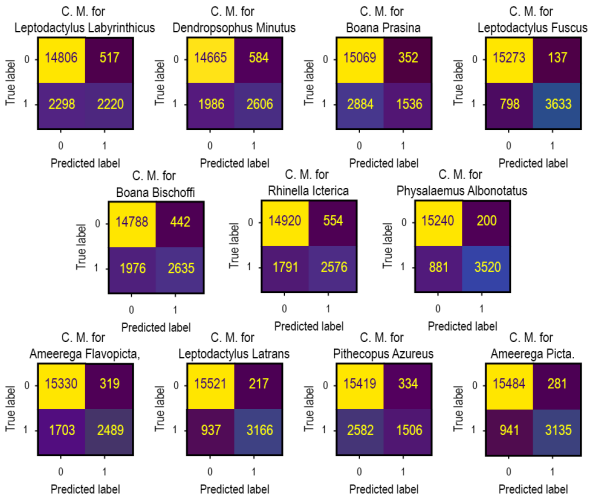

Figure 10.32: Confusion Matrix Mixed samples Fold 2 (ResNet)

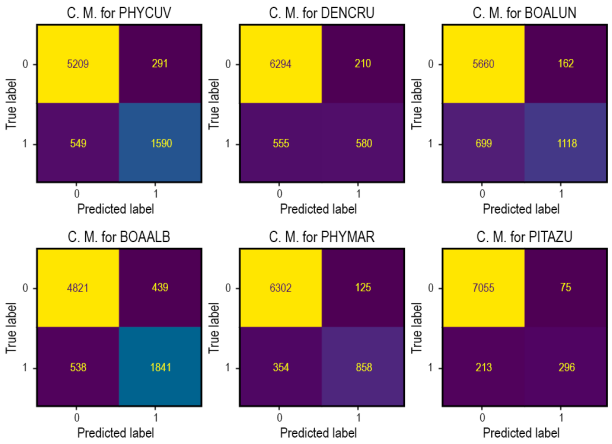

Figure 10.33: Confusion Matrix Anuraset Fold 2 (VGG)

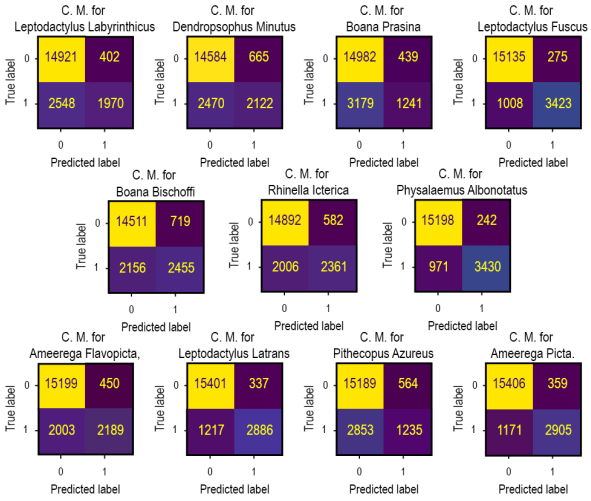

Figure 10.34: Confusion Matrix Mixed samples Fold 2 (VGG)

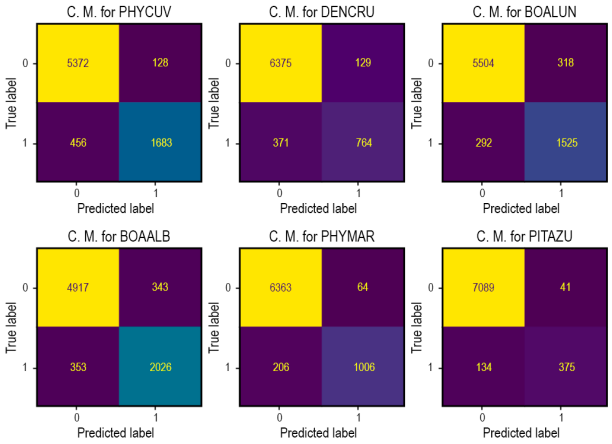

Figure 10.35: Confusion Matrix Anuraset Fold 2 (EfficientNet)

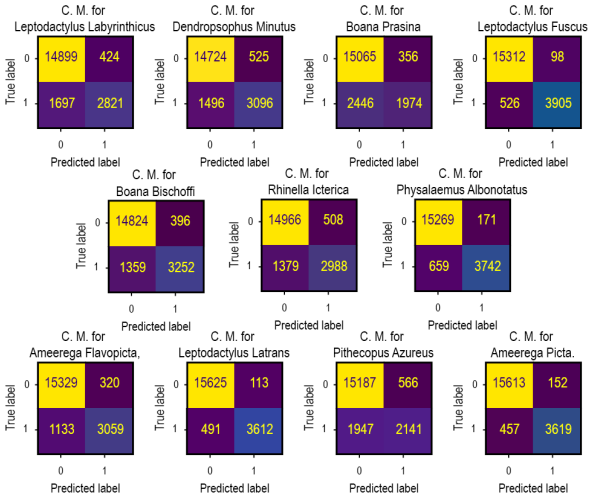

Figure 10.36: Confusion Matrix Mixed samples Fold 2 (EfficientNet)

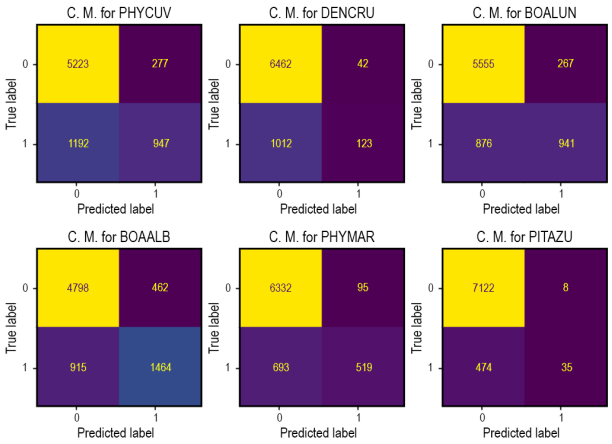

Figure 10.37: Confusion Matrix Anuraset Fold 3 (ResNet)

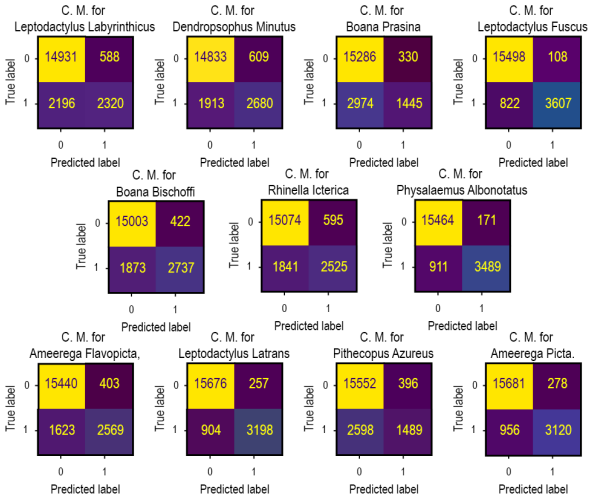

Figure 10.38: Confusion Matrix Mixed samples Fold 3 (ResNet)

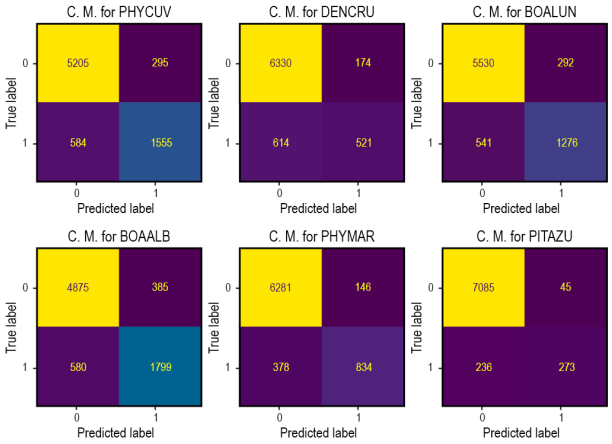

Figure 10.39: Confusion Matrix Anuraset Fold 3 (VGG)

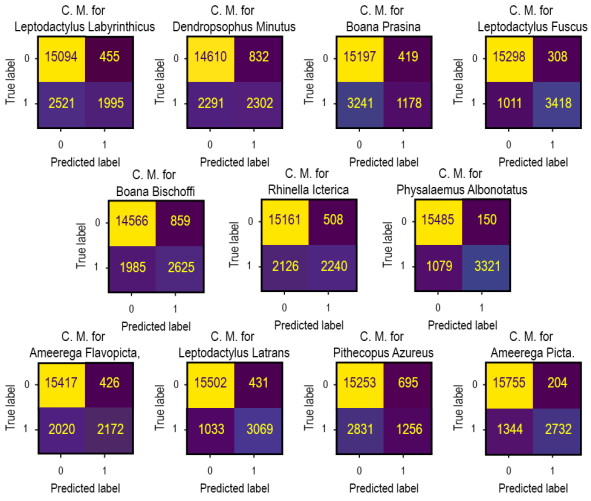

Figure 10.40: Confusion Matrix Mixed samples Fold 3 (VGG)

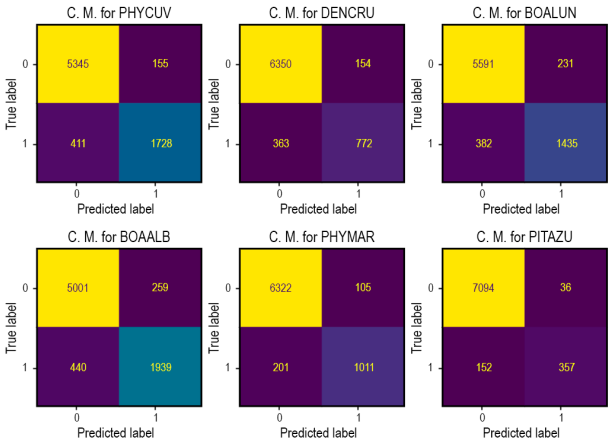

Figure 10.41: Confusion Matrix Anuraset Fold 3 (EfficientNet)

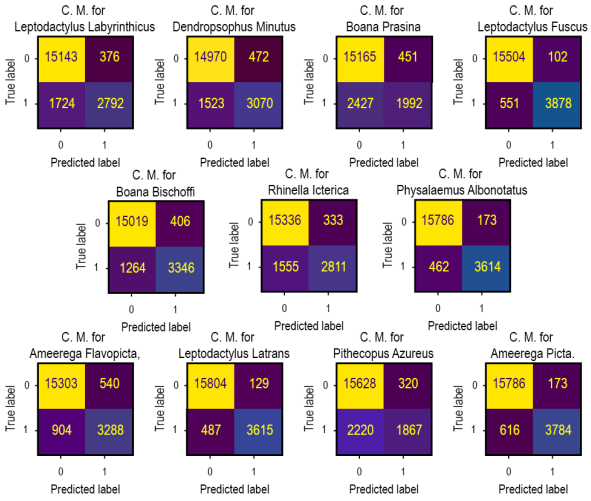

Figure 10.42: Confusion Matrix Mixed samples Fold 3 (EfficientNet)

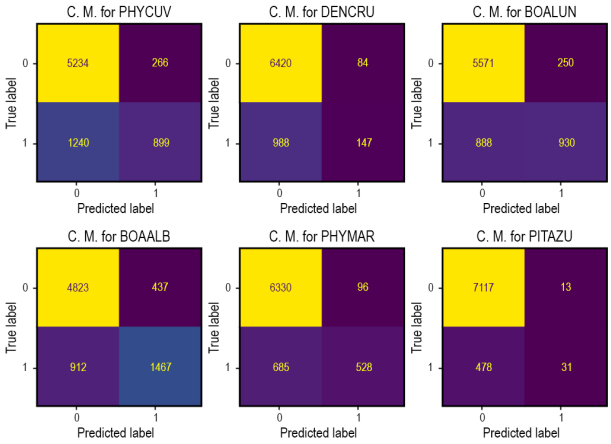

Figure 10.43: Confusion Matrix Anuraset Fold 4 (ResNet)

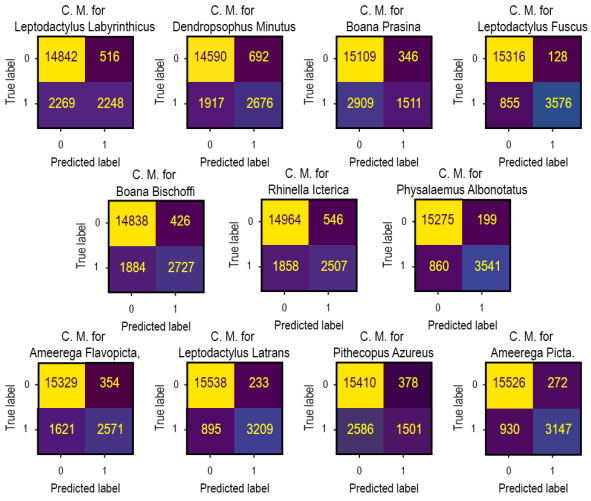

Figure 10.44: Confusion Matrix Mixed samples Fold 4 (ResNet)

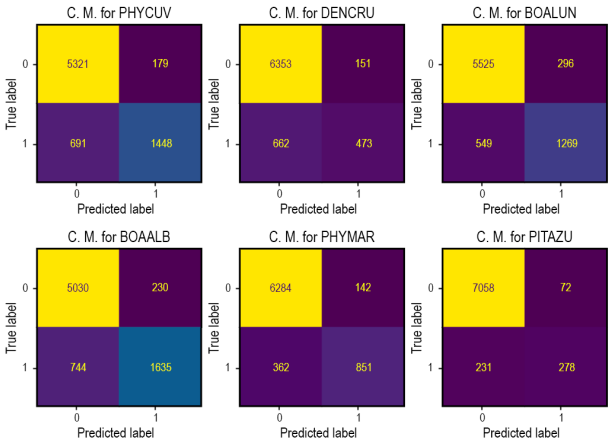

Figure 10.45: Confusion Matrix Anuraset Fold 4 (VGG)

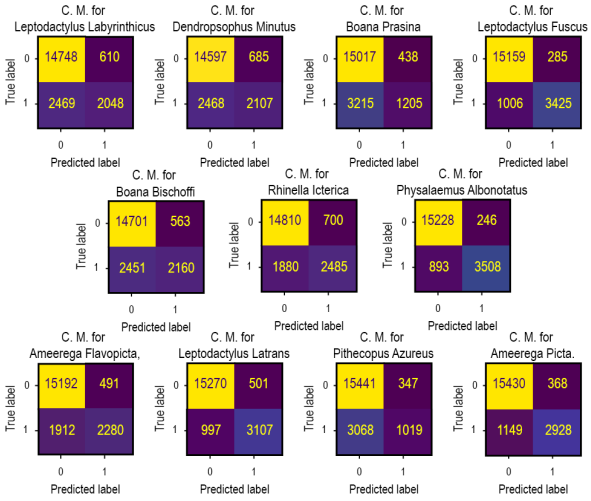

Figure 10.46: Confusion Matrix Mixed samples Fold 4 (VGG)

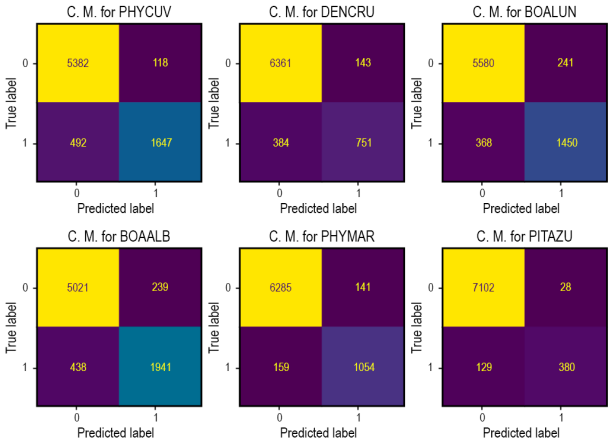

Figure 10.47: Confusion Matrix Anuraset Fold 4 (EfficientNet)

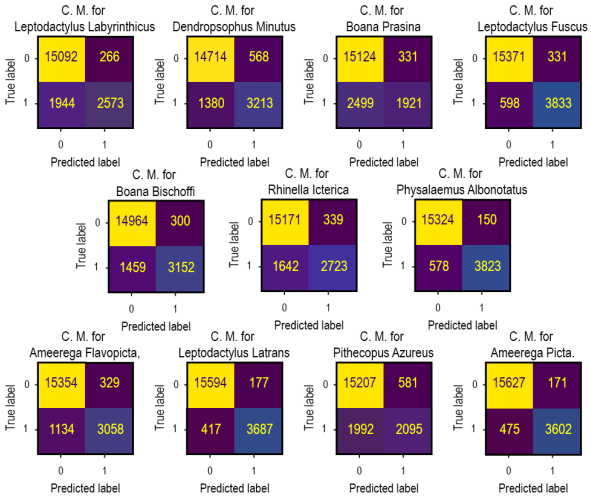

Figure 10.48: Confusion Matrix Mixed samples Fold 4 (EfficientNet)

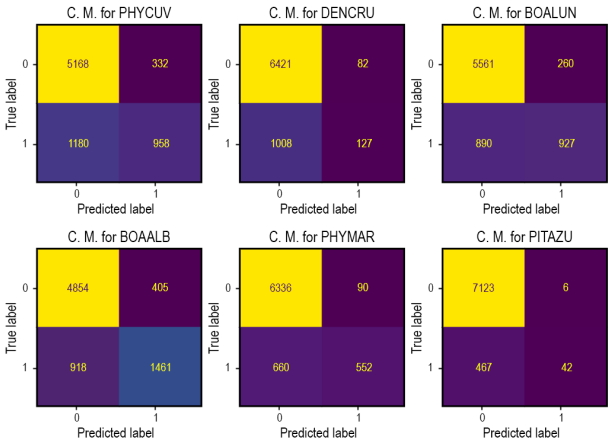

Figure 10.49: Confusion Matrix Anuraset Fold 5 (ResNet)

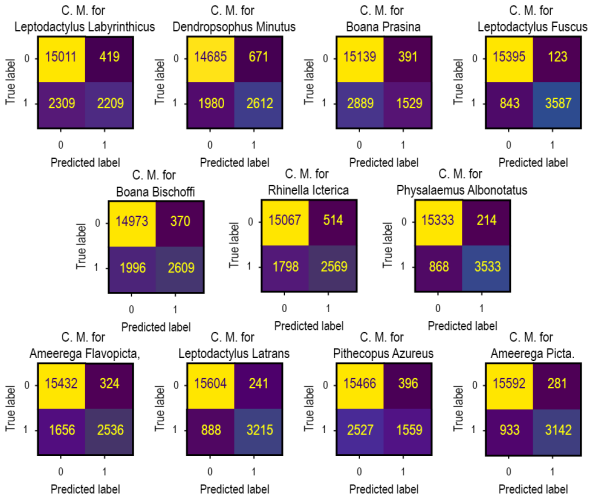

Figure 10.50: Confusion Matrix Mixed samples Fold 5 (ResNet)

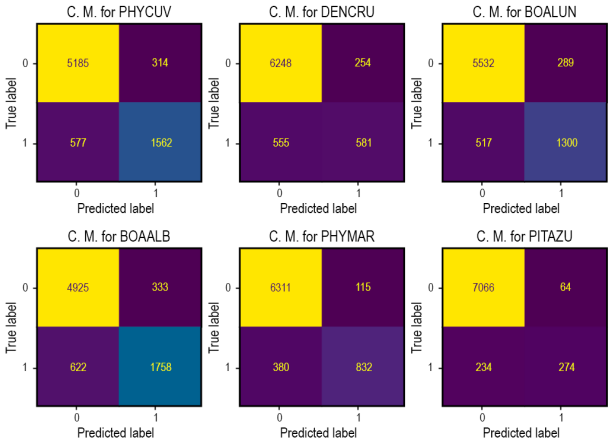

Figure 10.51: Confusion Matrix Anuraset Fold 5 (VGG)

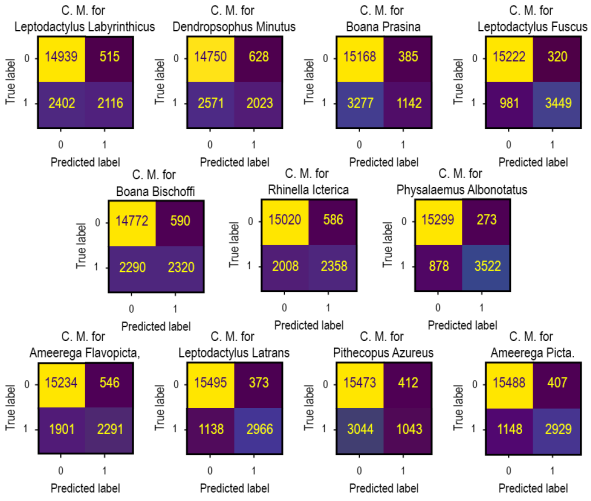

Figure 10.52: Confusion Matrix Mixed samples Fold 5 (VGG)

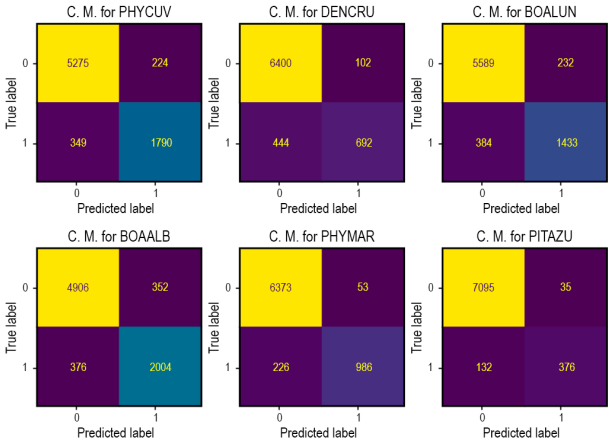

Figure 10.53: Confusion Matrix Anuraset Fold 5 (EfficientNet)

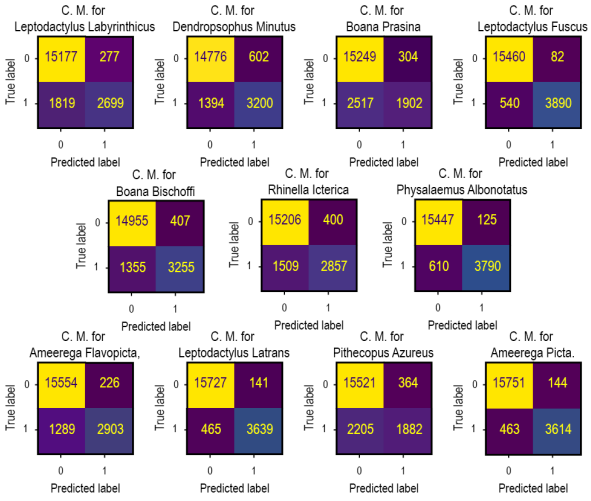

Figure 10.54: Confusion Matrix Mixed samples Fold 5 (EfficientNet)

Appendix 3 – Learning Curves and Confusion Matrices from experiments with no augmentations but using transfer learning

10.0.3 Learning curves.

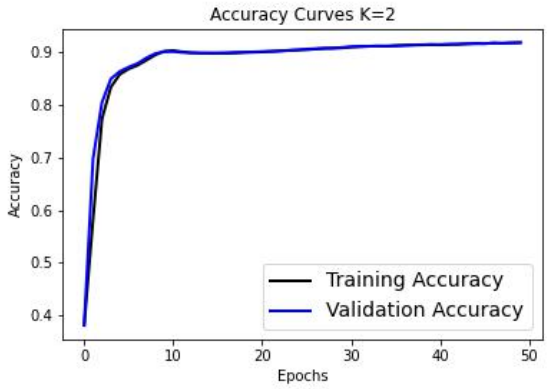

Figure 10.55: Accuracy Curve Fold 2 (ResNet)

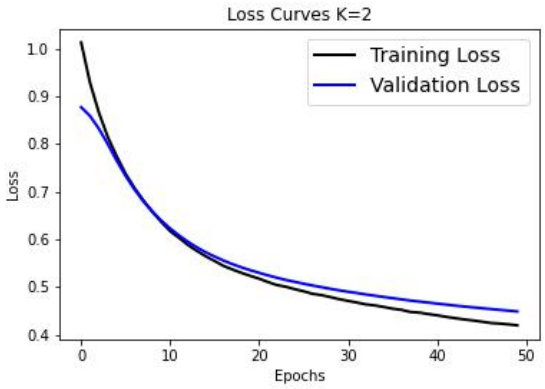

Figure 10.56: Loss Curve Fold 2 (ResNet)

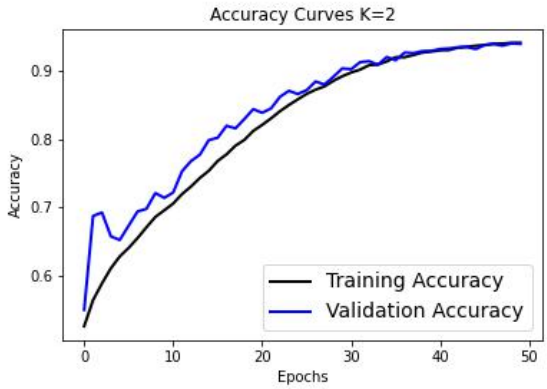

Figure 10.57: Accuracy Curve Fold 2 (VGG)

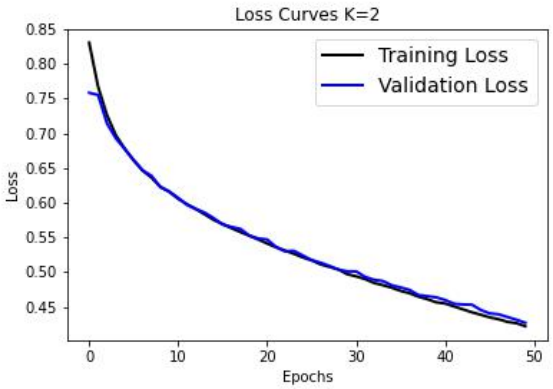

Figure 10.58: Loss Curve Fold 2 (VGG)

10.0.4 Confusion Matrices

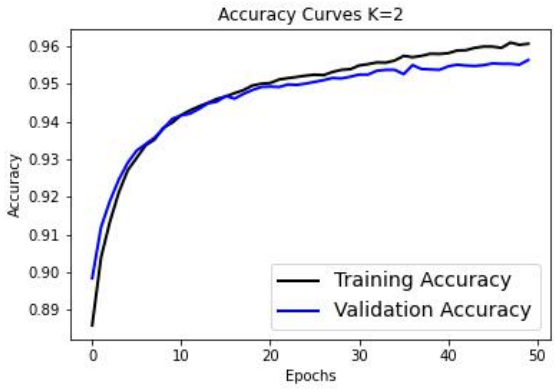

Figure 10.59: Accuracy Curve Fold 2 (EfficientNet)

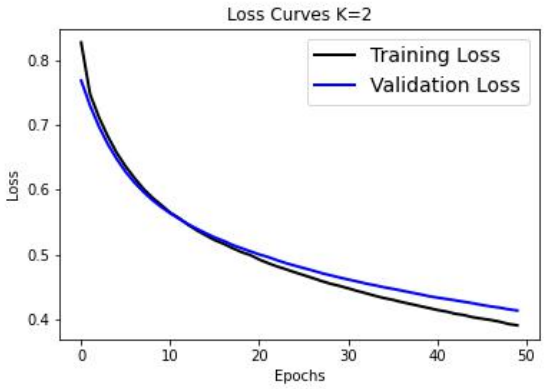

Figure 10.60: Loss Curve Fold 2 (EfficientNet)

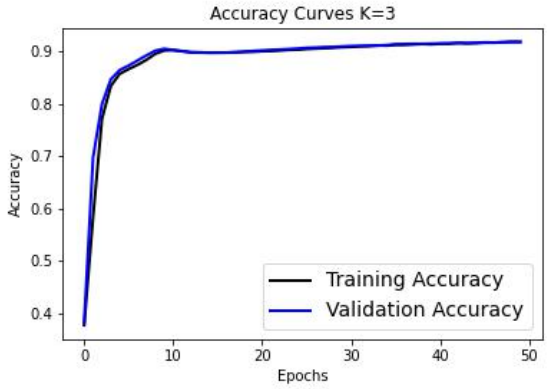

Figure 10.61: Accuracy Curve Fold 3 (ResNet)

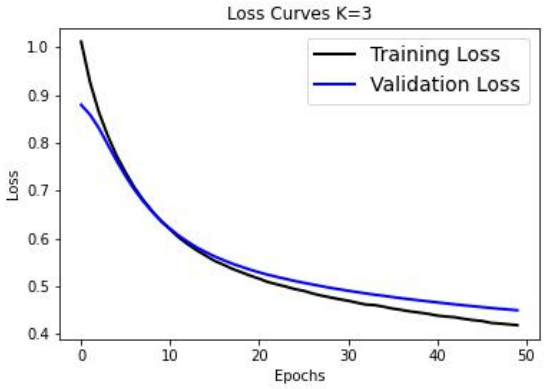

Figure 10.62: Loss Curve Fold 3 (ResNet)

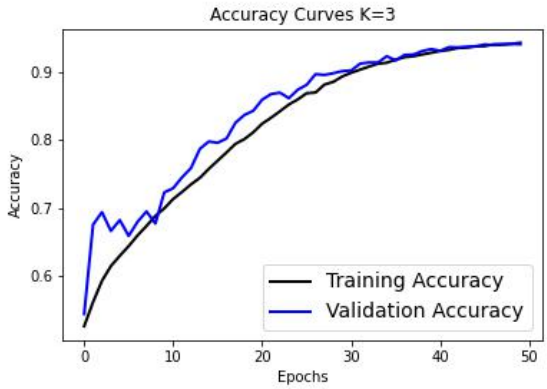

Figure 10.63: Accuracy Curve Fold 3 (VGG)

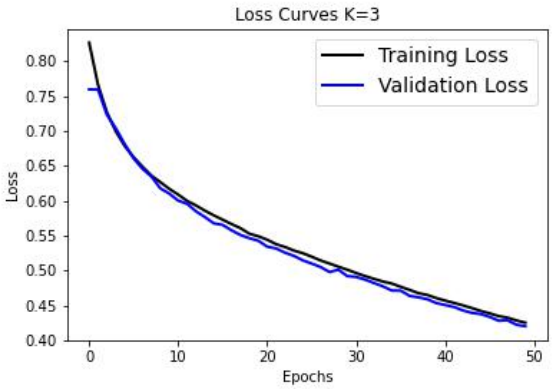

Figure 10.64: Loss Curve Fold 3 (VGG)

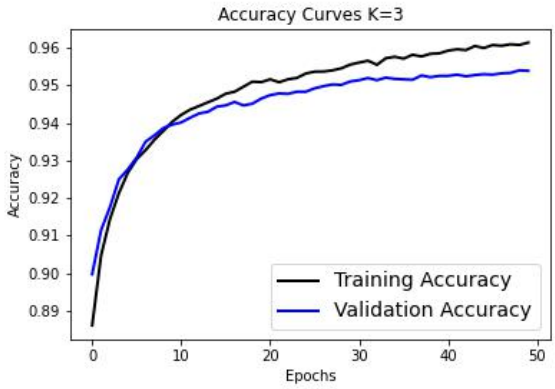

Figure 10.65: Accuracy Curve Fold 3 (EfficientNet)

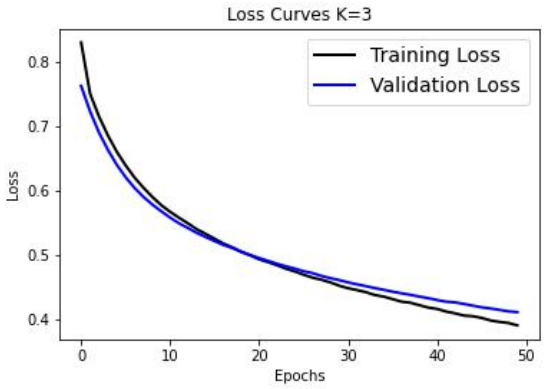

Figure 10.66: Loss Curve Fold 3 (EfficientNet)

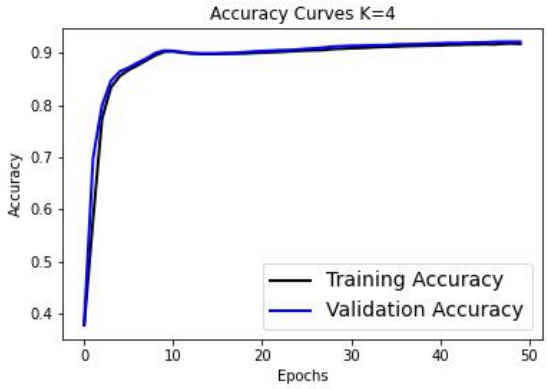

Figure 10.67: Accuracy Curve Fold 4 (ResNet)

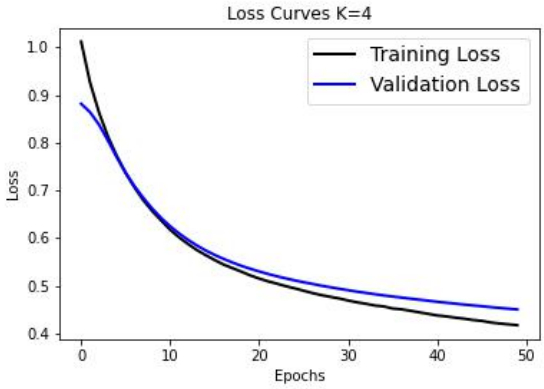

Figure 10.68: Loss Curve Fold 4 (ResNet)

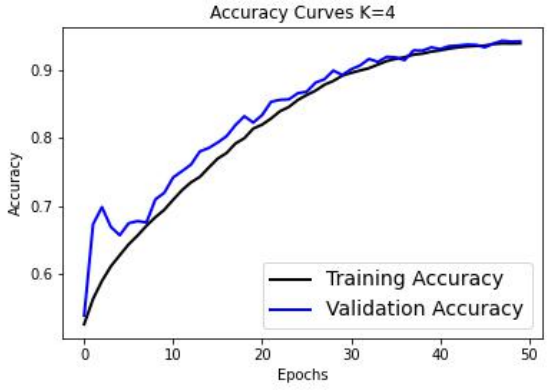

Figure 10.69: Accuracy Curve Fold 4 (VGG)

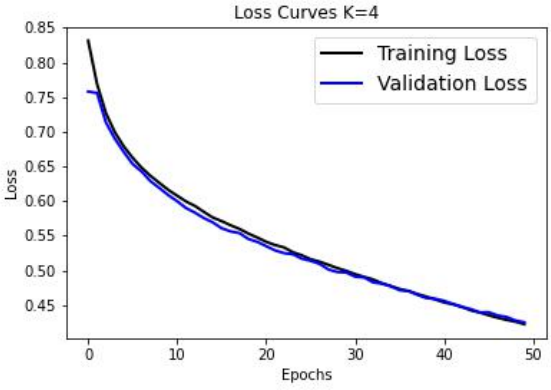

Figure 10.70: Loss Curve Fold 4 (VGG)

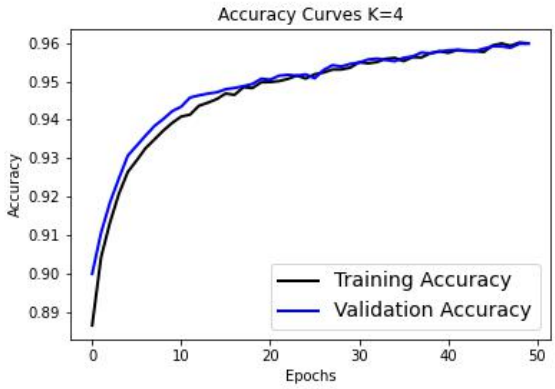

Figure 10.71: Accuracy Curve Fold 4 (EfficientNet)

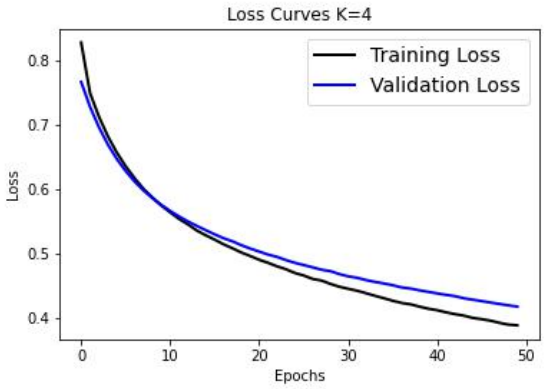

Figure 10.72: Loss Curve Fold 4 (EfficientNet)

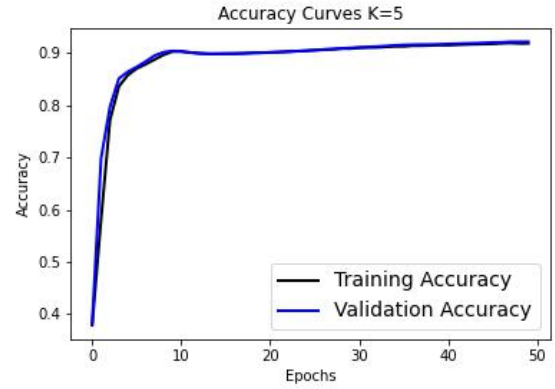

Figure 10.73: Accuracy Curve Fold 5 (ResNet)

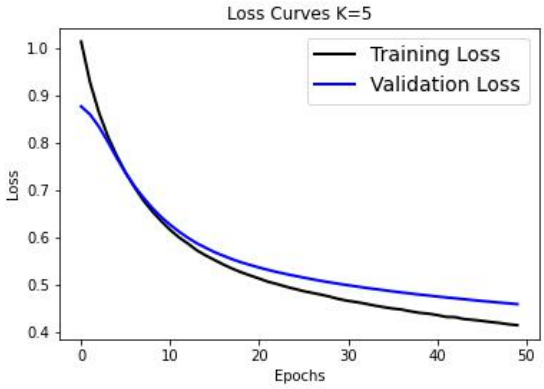

Figure 10.74: Loss Curve Fold 5 (ResNet)

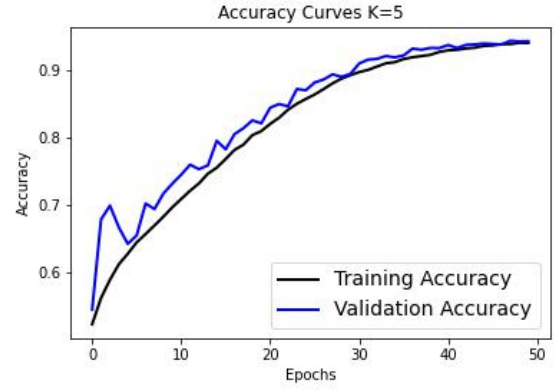

Figure 10.75: Accuracy Curve Fold 5 (VGG)

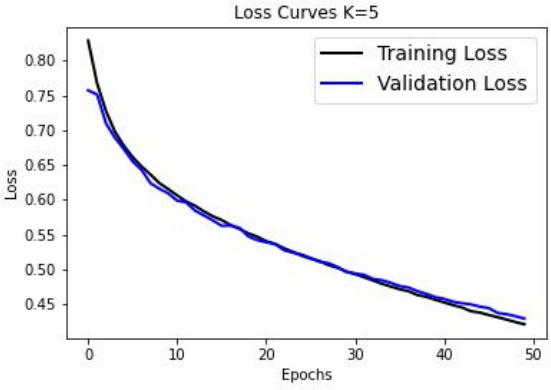

Figure 10.76: Loss Curve Fold 5 (VGG)

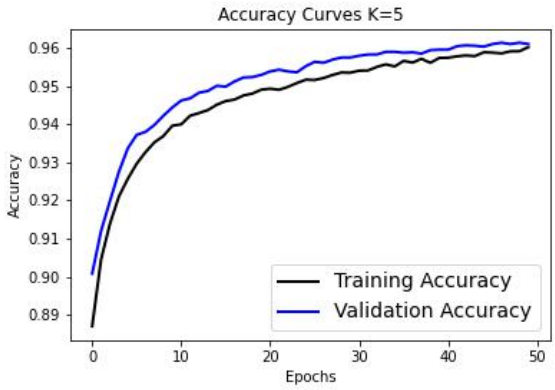

Figure 10.77: Accuracy Curve Fold 5 (EfficientNet)

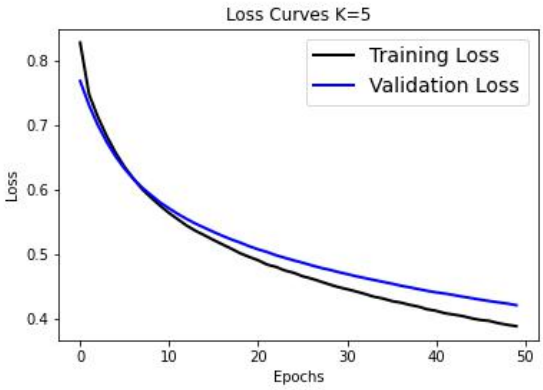

Figure 10.78: Loss Curve Fold 5 (EfficientNet)

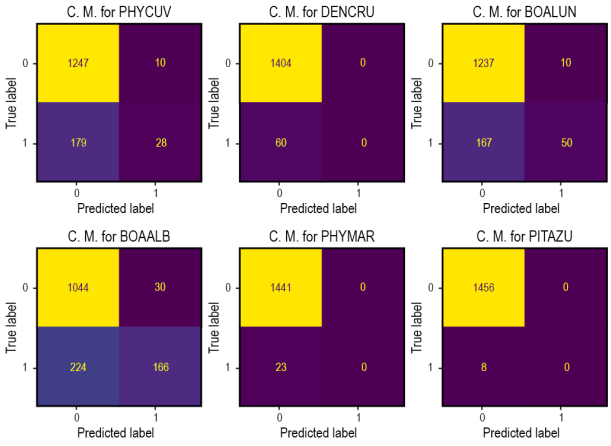

Figure 10.79: Confusion Matrix Anuraset Fold 2 (ResNet)

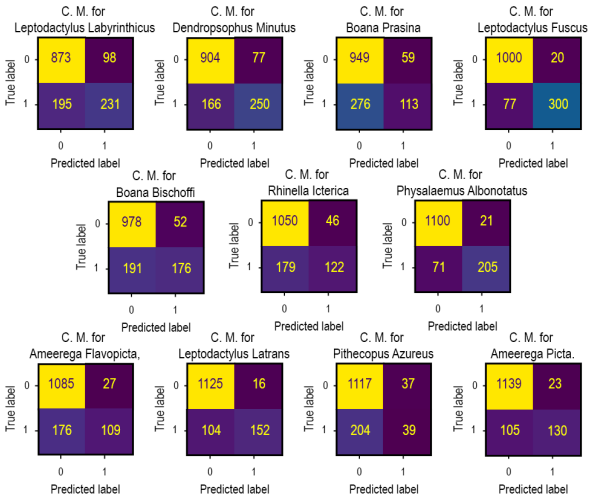

Figure 10.80: Confusion Matrix Mixed samples Fold 2 (ResNet)

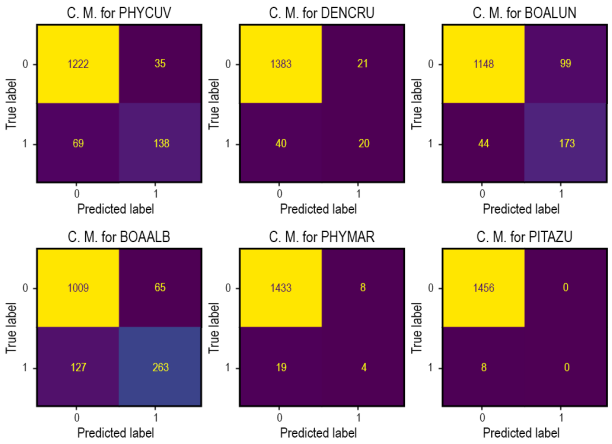

Figure 10.81: Confusion Matrix Anuraset Fold 2 (VGG)

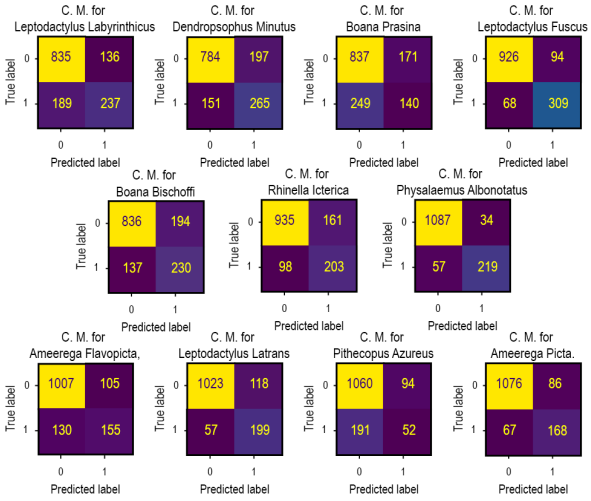

Figure 10.82: Confusion Matrix Mixed samples Fold 2 (VGG)

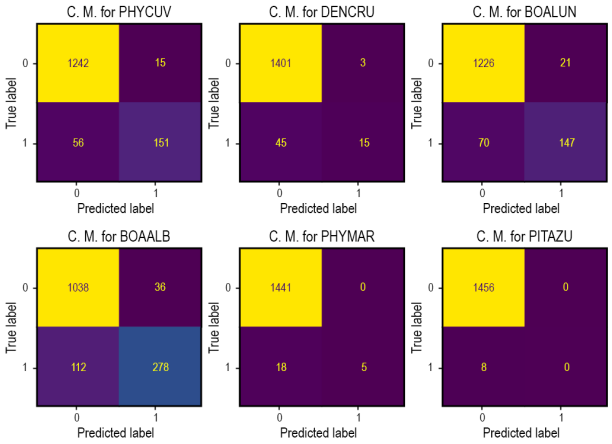

Figure 10.83: Confusion Matrix Anuraset Fold 2 (EfficientNet)

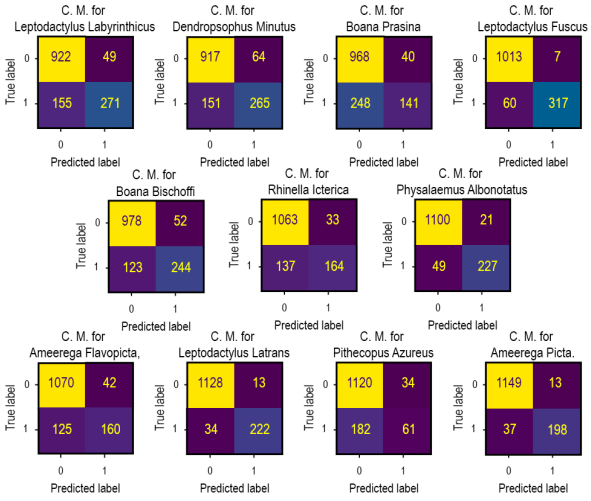

Figure 10.84: Confusion Matrix Mixed samples Fold 2 (EfficientNet)

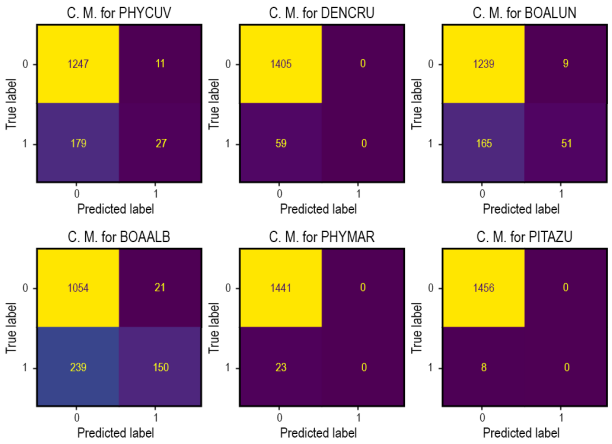

Figure 10.85: Confusion Matrix Anuraset Fold 3 (ResNet)

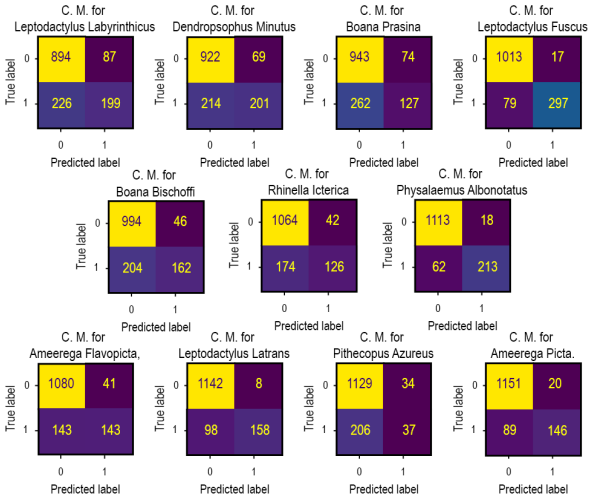

Figure 10.86: Confusion Matrix Mixed samples Fold 3 (ResNet)

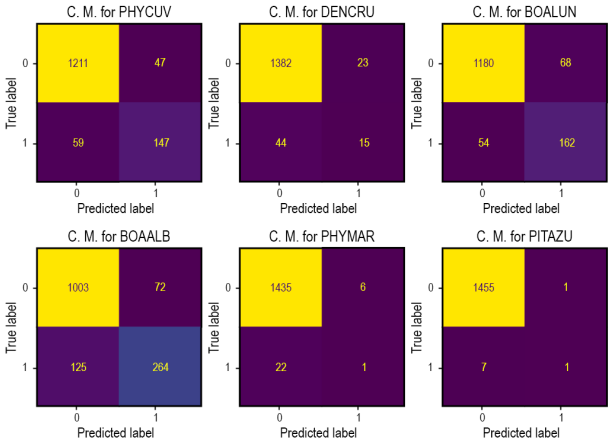

Figure 10.87: Confusion Matrix Anuraset Fold 3 (VGG)

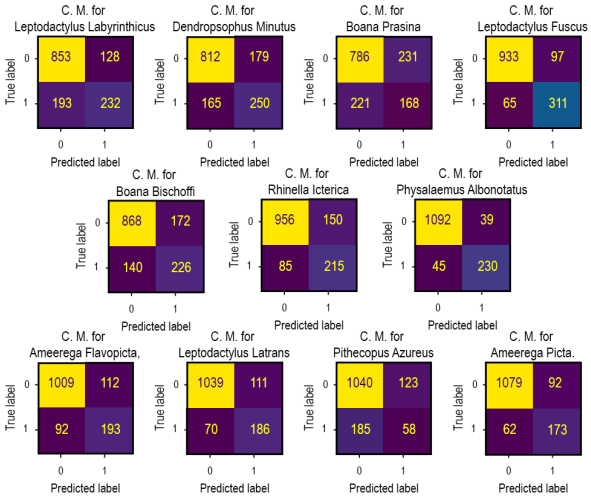

Figure 10.88: Confusion Matrix Mixed samples Fold 3 (VGG)

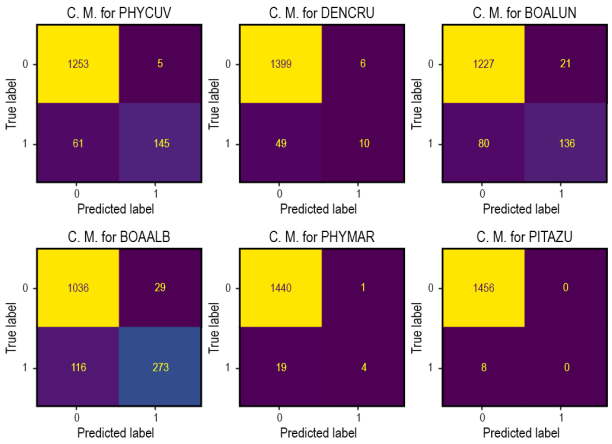

Figure 10.89: Confusion Matrix Anuraset Fold 3 (EfficientNet)

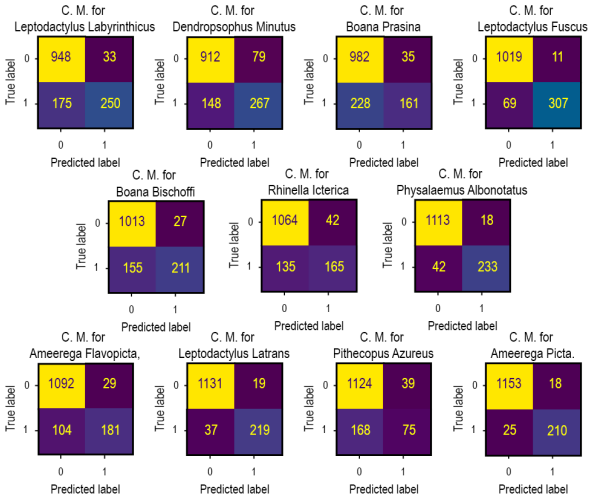

Figure 10.90: Confusion Matrix Mixed samples Fold 3 (EfficientNet)

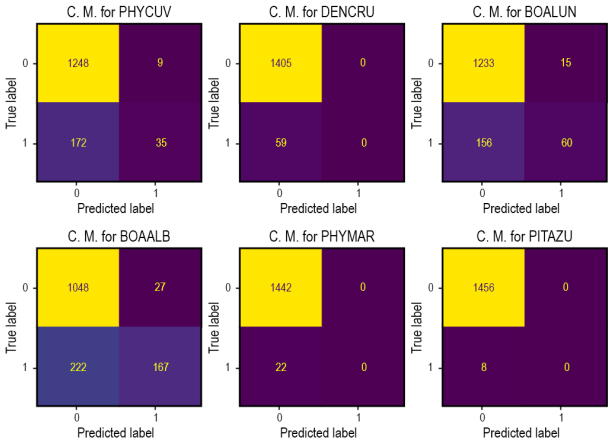

Figure 10.91: Confusion Matrix Anuraset Fold 4 (ResNet)

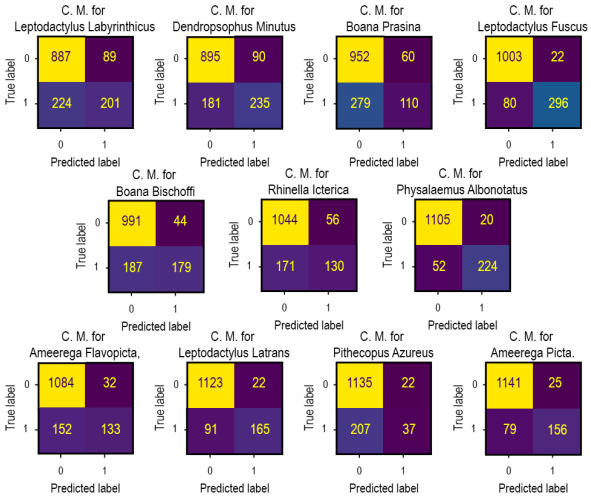

Figure 10.92: Confusion Matrix Mixed samples Fold 4 (ResNet)

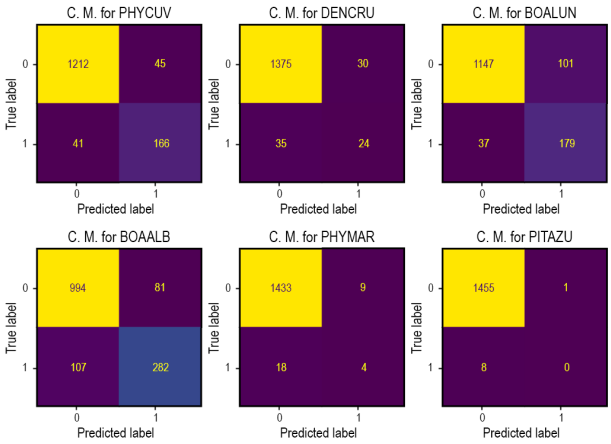

Figure 10.93: Confusion Matrix Anuraset Fold 4 (VGG)

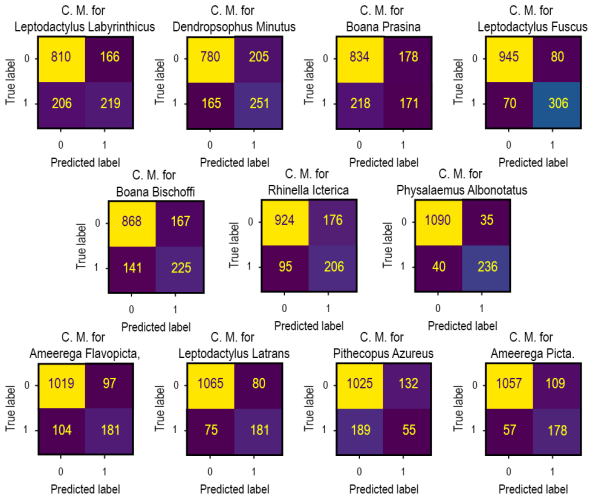

Figure 10.94: Confusion Matrix Mixed samples Fold 4 (VGG)

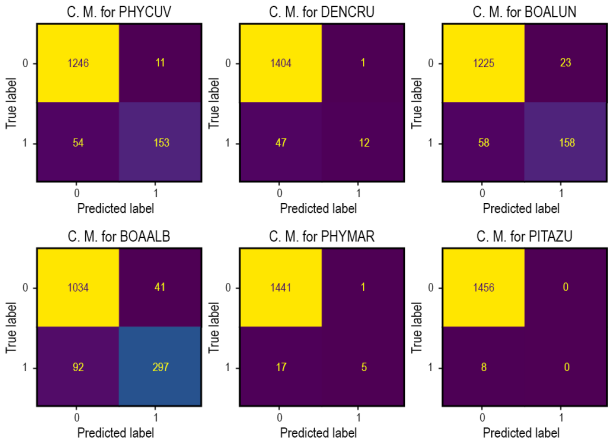

Figure 10.95: Confusion Matrix Anuraset Fold 4 (EfficientNet)

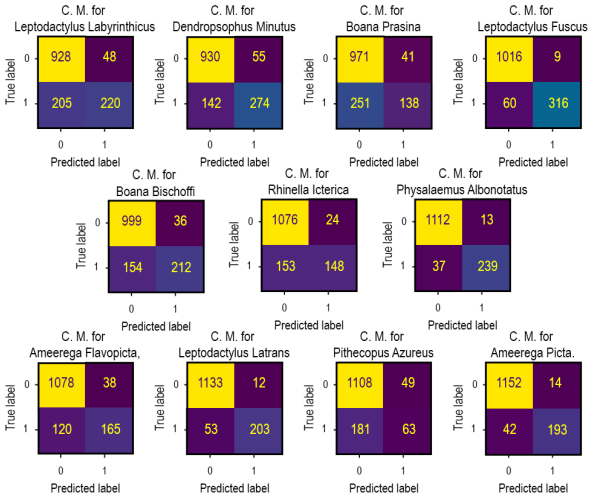

Figure 10.96: Confusion Matrix Mixed samples Fold 4 (EfficientNet)

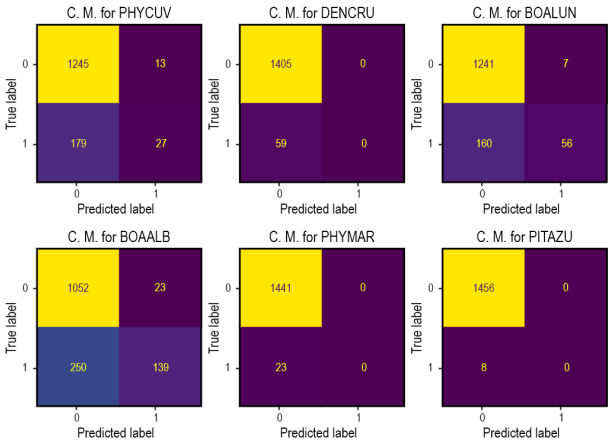

Figure 10.97: Confusion Matrix Anuraset Fold 5 (ResNet)

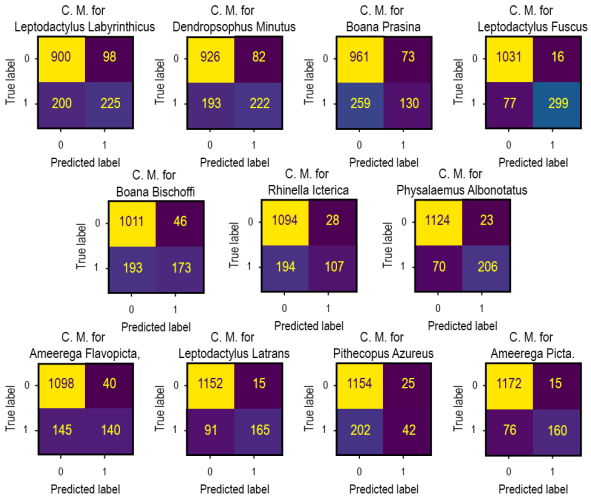

Figure 10.98: Confusion Matrix Mixed samples Fold 5 (ResNet)

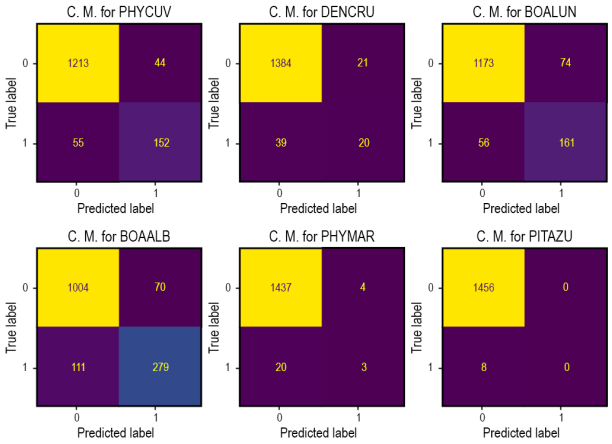

Figure 10.99: Confusion Matrix Anuraset Fold 5 (VGG)

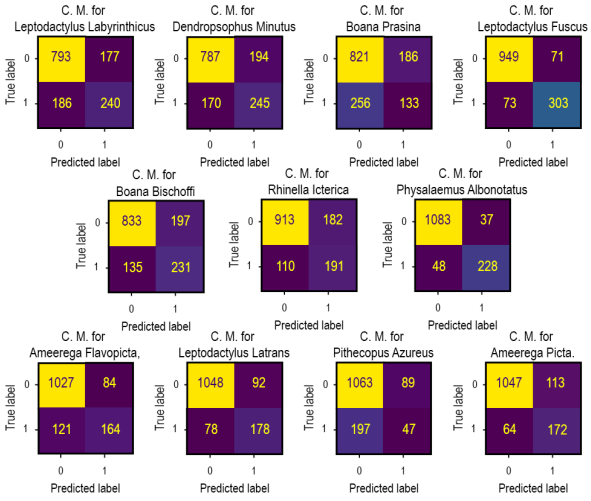

Figure 10.100: Confusion Matrix Mixed samples Fold 5 (VGG)

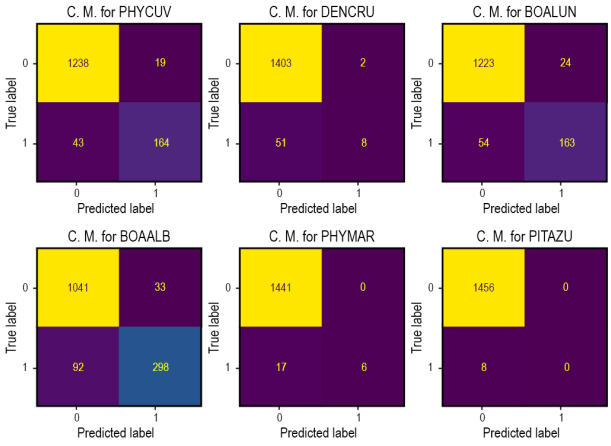

Figure 10.101: Confusion Matrix Anuraset Fold 5 (EfficientNet)

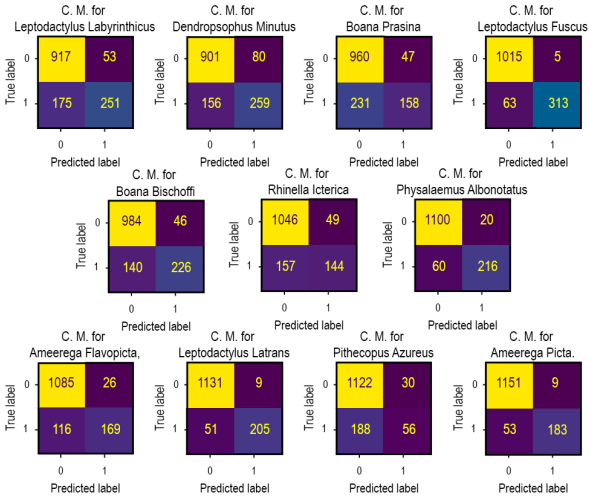

Figure 10.102: Confusion Matrix Mixed samples Fold 5 (EfficientNet)

Appendix 4 – Learning Curves and Confusion Matrices from experiments with augmentations but without transfer learning

10.0.5 Learning curves.

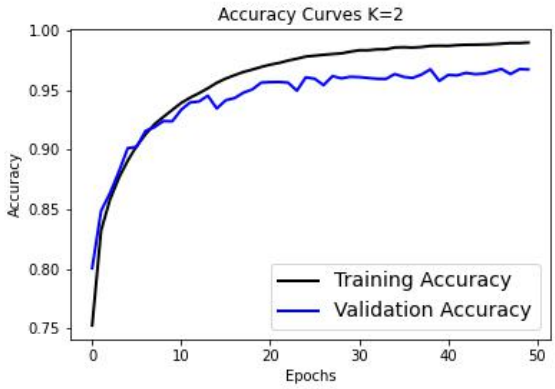

Figure 10.103: Accuracy Curve Fold 2 (ResNet)

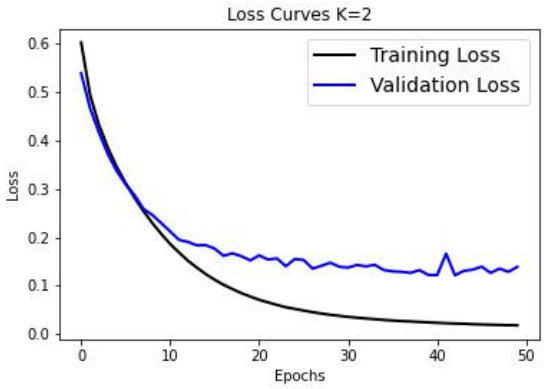

Figure 10.104: Loss Curve Fold 2 (ResNet)

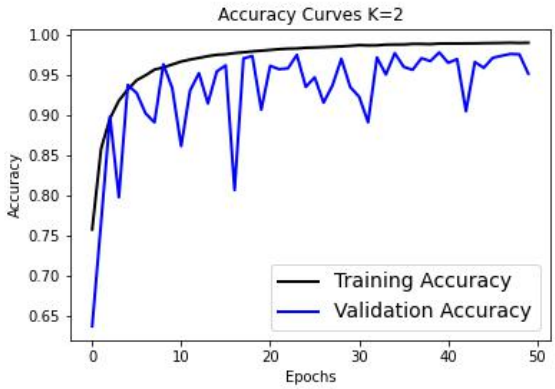

Figure 10.105: Accuracy Curve Fold 2 (VGG)

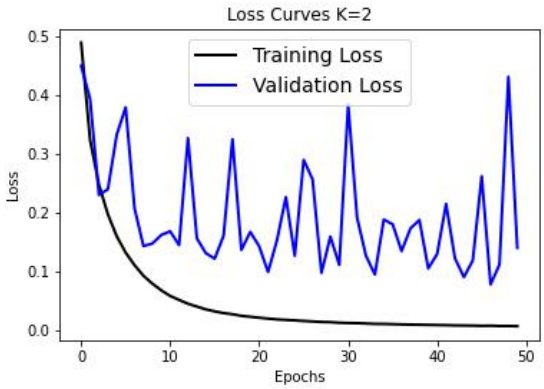

Figure 10.106: Loss Curve Fold 2 (VGG)

10.0.6 Confusion Matrices

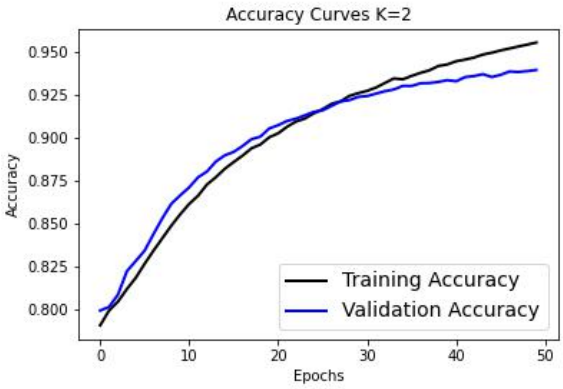

Figure 10.107: Accuracy Curve Fold 2 (EfficientNet)

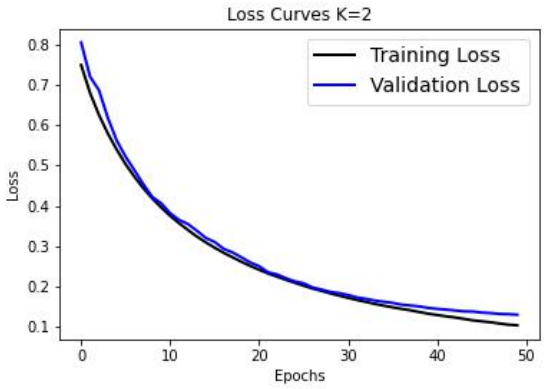

Figure 10.108: Loss Curve Fold 2 (EfficientNet)

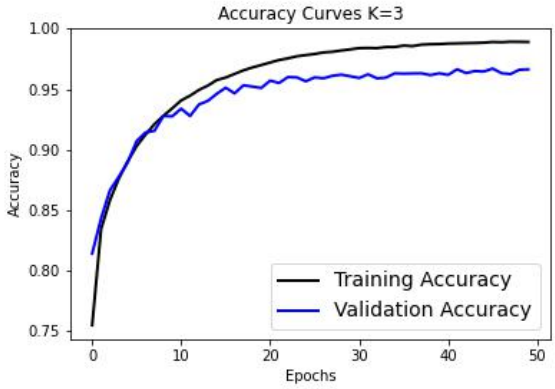

Figure 10.109: Accuracy Curve Fold 3 (ResNet)

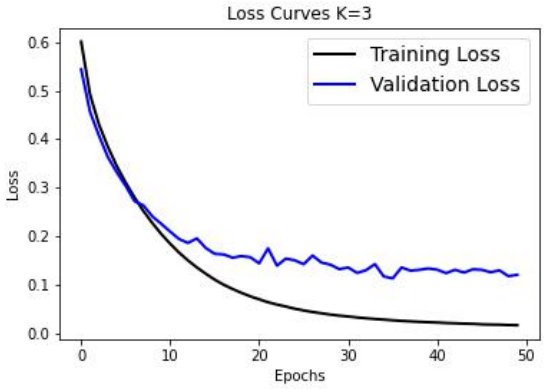

Figure 10.110: Loss Curve Fold 3 (ResNet)

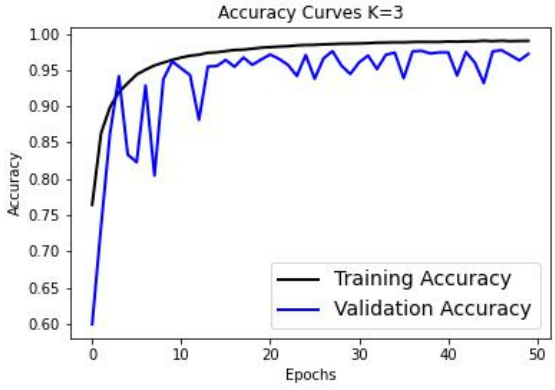

Figure 10.111: Accuracy Curve Fold 3 (VGG)

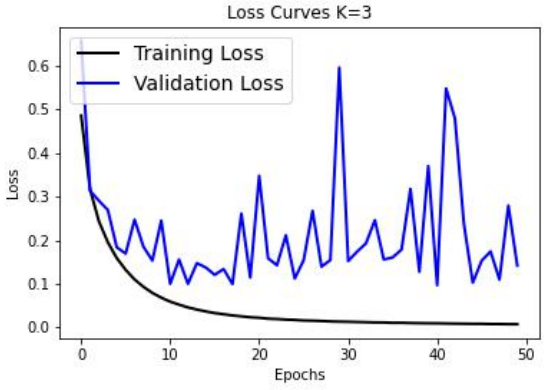

Figure 10.112: Loss Curve Fold 3 (VGG)

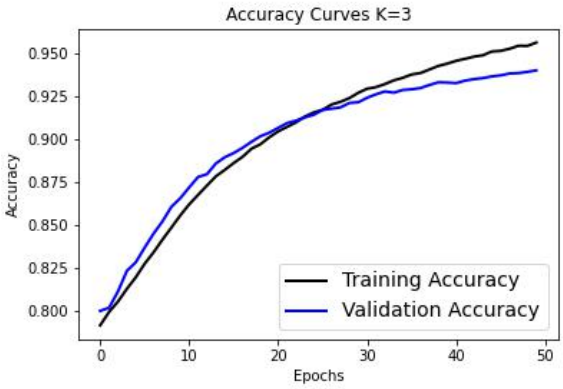

Figure 10.113: Accuracy Curve Fold 3 (EfficientNet)

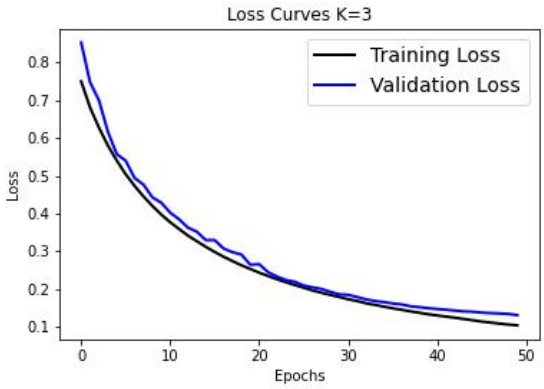

Figure 10.114: Loss Curve Fold 3 (EfficientNet)

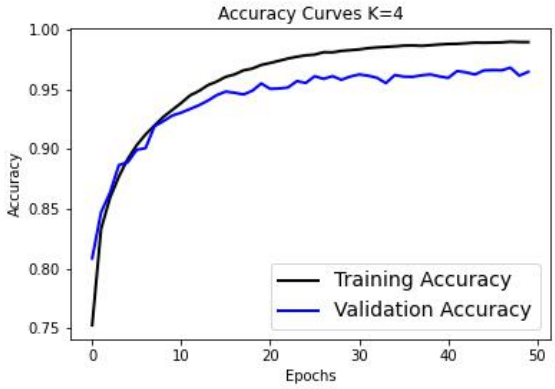

Figure 10.115: Accuracy Curve Fold 4 (ResNet)

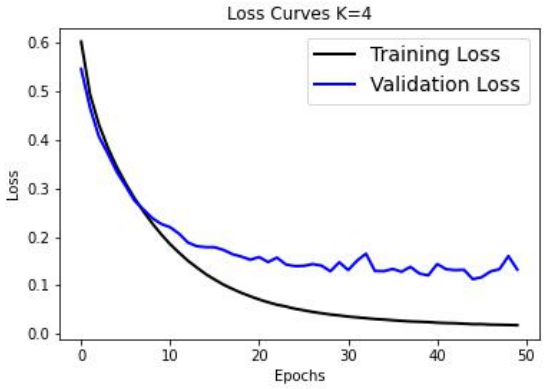

Figure 10.116: Loss Curve Fold 4 (ResNet)

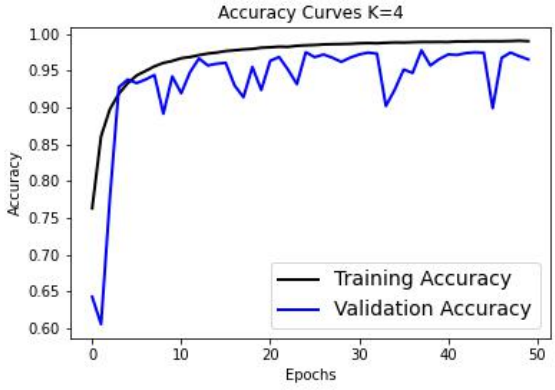

Figure 10.117: Accuracy Curve Fold 4 (VGG)

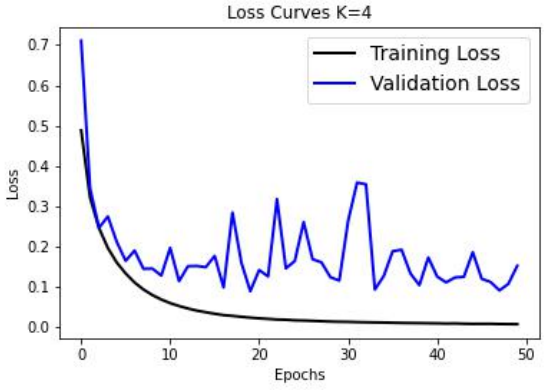

Figure 10.118: Loss Curve Fold 4 (VGG)

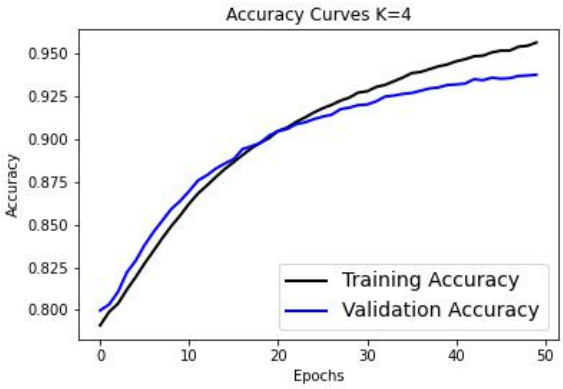

Figure 10.119: Accuracy Curve Fold 4 (EfficientNet)

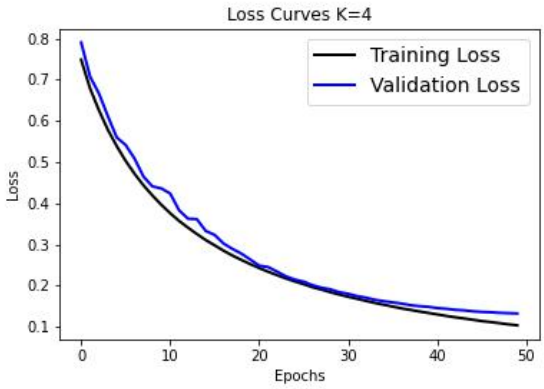

Figure 10.120: Loss Curve Fold 4 (EfficientNet)

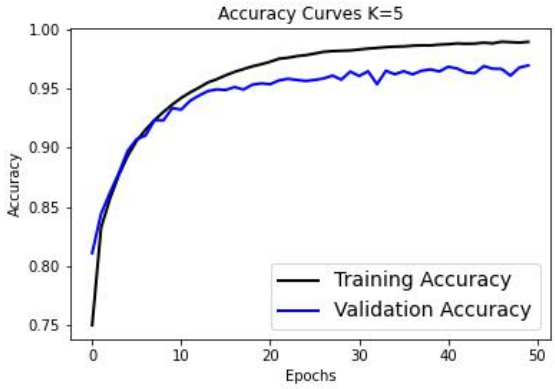

Figure 10.121: Accuracy Curve Fold 5 (ResNet)

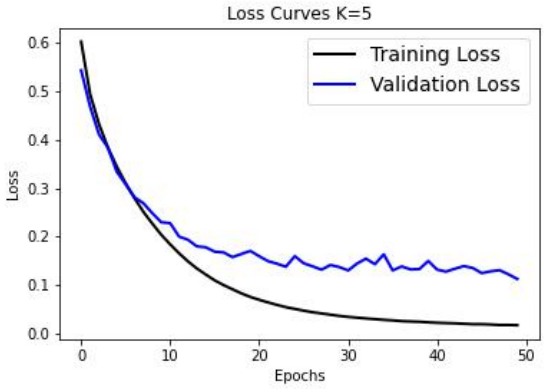

Figure 10.122: Loss Curve Fold 5 (ResNet)

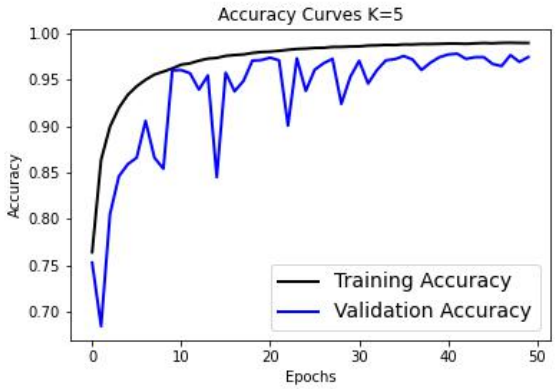

Figure 10.123: Accuracy Curve Fold 5 (VGG)

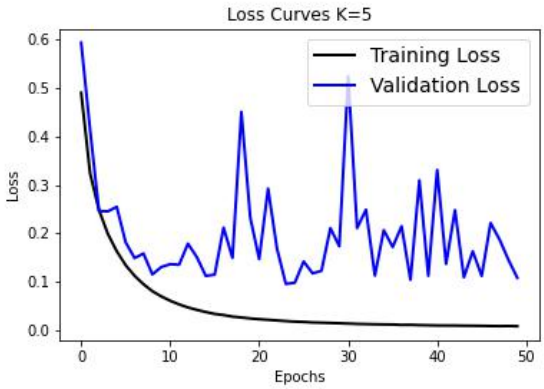

Figure 10.124: Loss Curve Fold 5 (VGG)

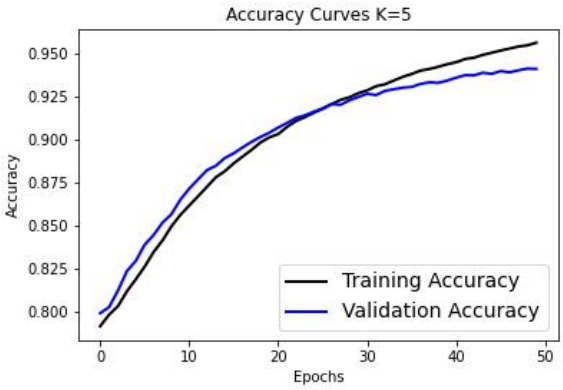

Figure 10.125: Accuracy Curve Fold 5 (EfficientNet)

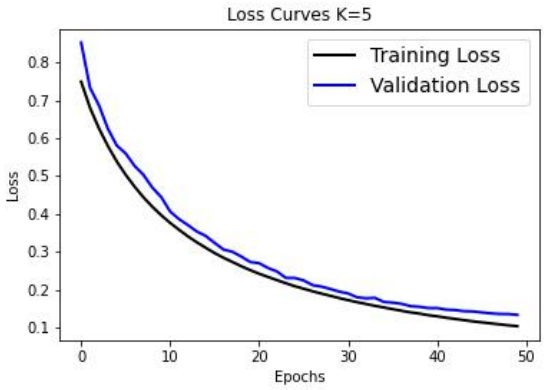

Figure 10.126: Loss Curve Fold 5 (EfficientNet)

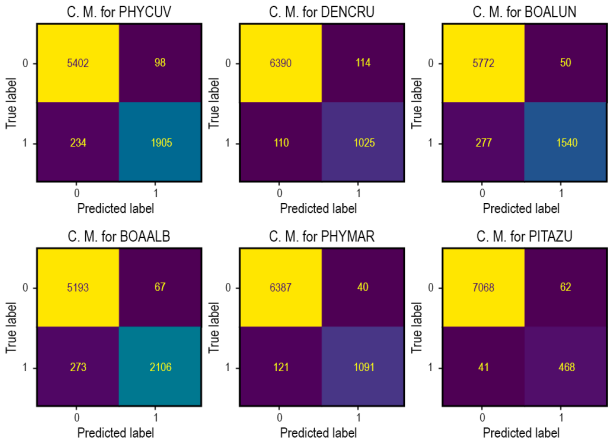

Figure 10.127: Confusion Matrix Anuraset Fold 2 (ResNet)

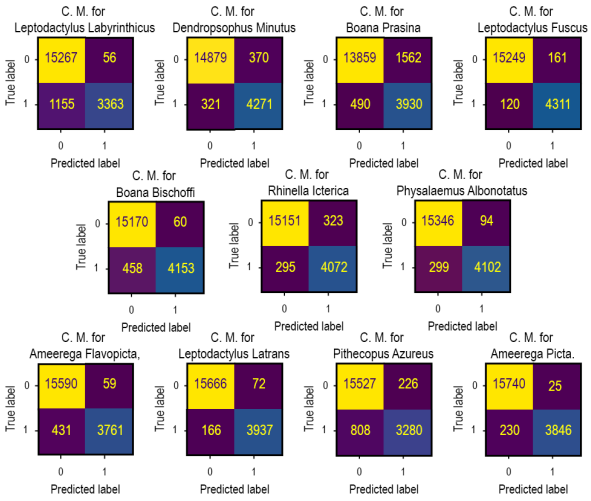

Figure 10.128: Confusion Matrix Mixed samples Fold 2 (ResNet)

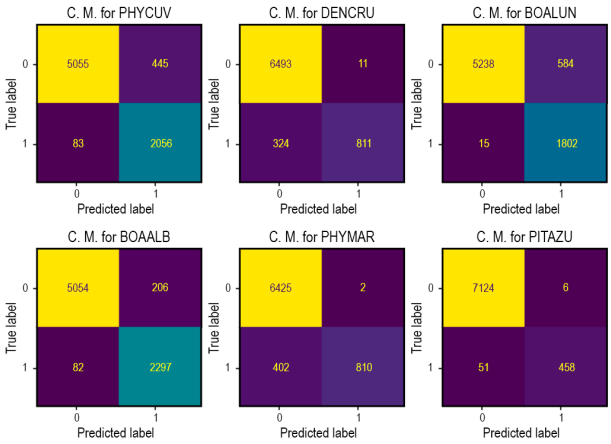

Figure 10.129: Confusion Matrix Anuraset Fold 2 (VGG)

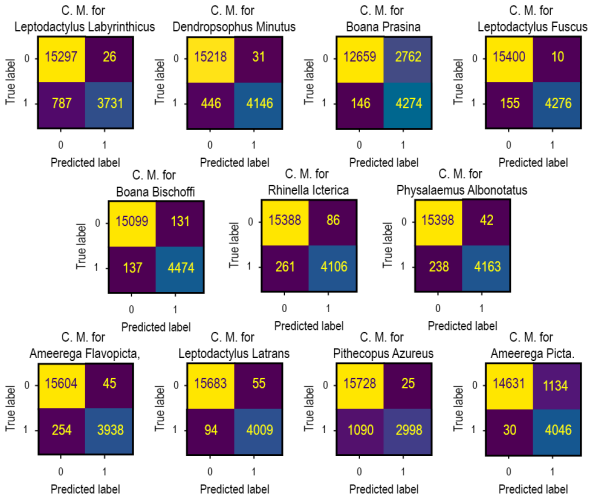

Figure 10.130: Confusion Matrix Mixed samples Fold 2 (VGG)

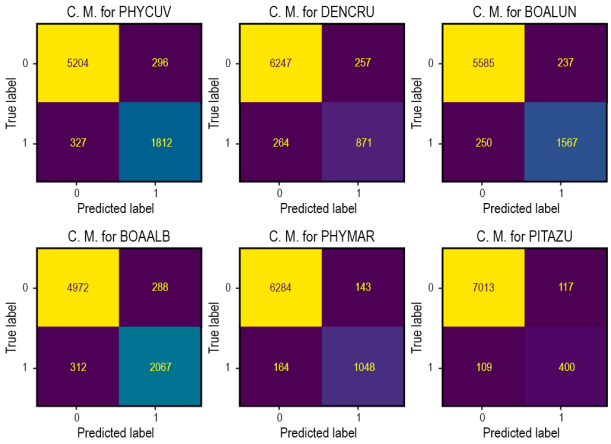

Figure 10.131: Confusion Matrix Anuraset Fold 2 (EfficientNet)

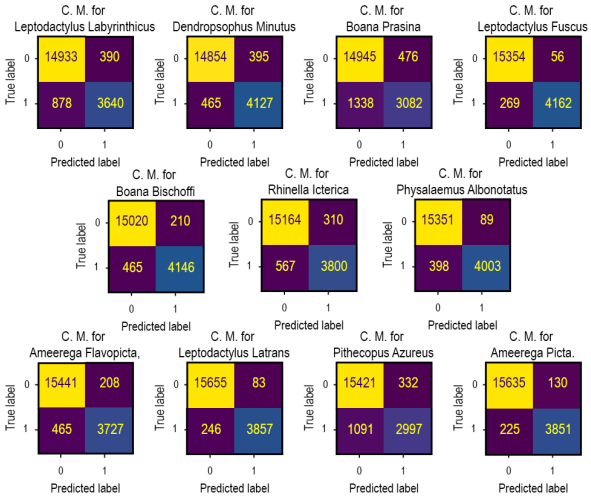

Figure 10.132: Confusion Matrix Mixed samples Fold 2 (EfficientNet)

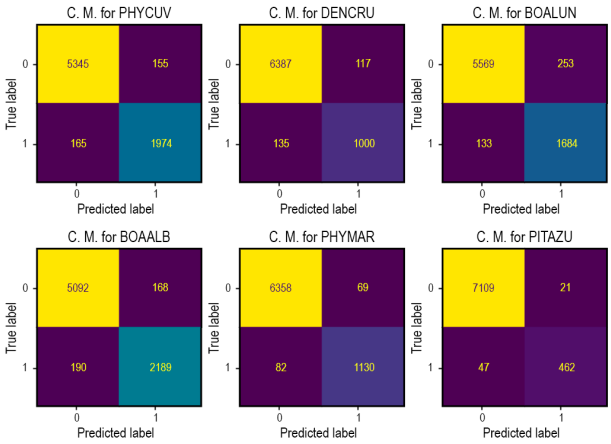

Figure 10.133: Confusion Matrix Anuraset Fold 3 (ResNet)

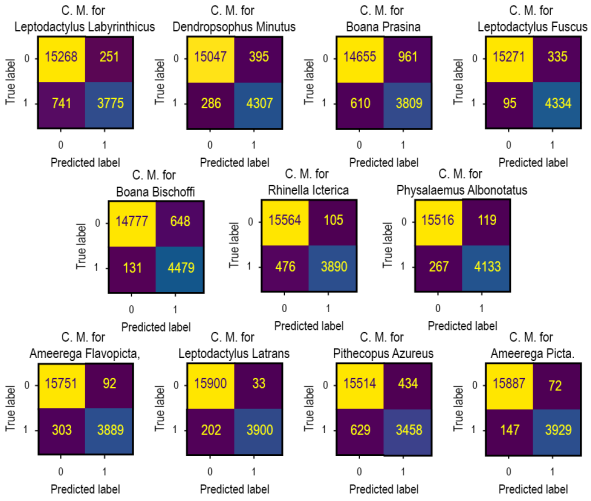

Figure 10.134: Confusion Matrix Mixed samples Fold 3 (ResNet)

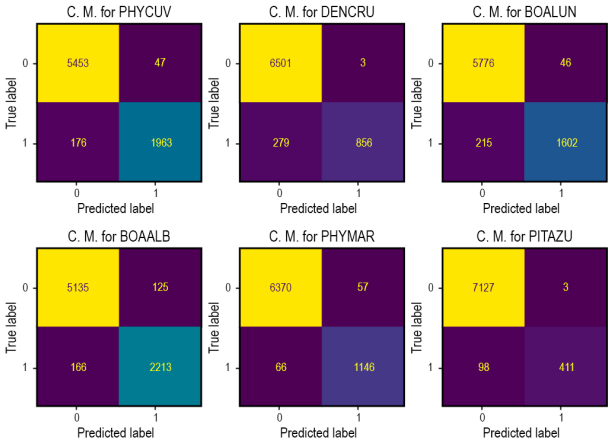

Figure 10.135: Confusion Matrix Anuraset Fold 3 (VGG)

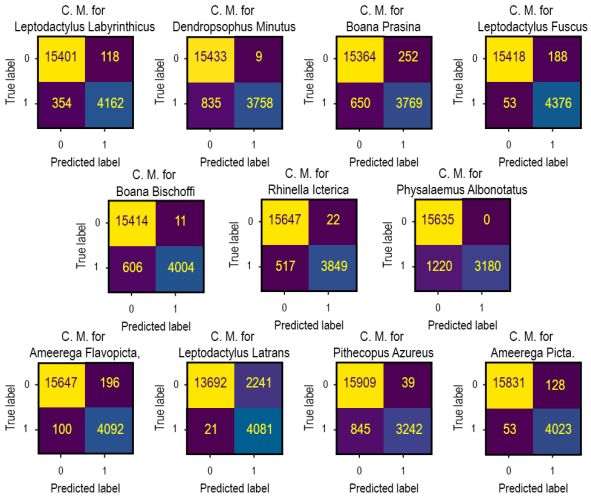

Figure 10.136: Confusion Matrix Mixed samples Fold 3 (VGG)

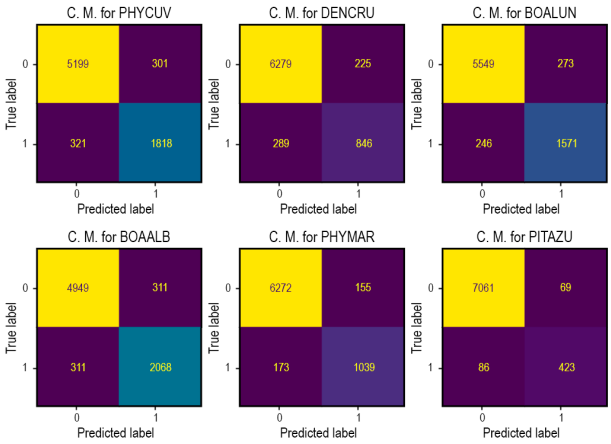

Figure 10.137: Confusion Matrix Anuraset Fold 3 (EfficientNet)

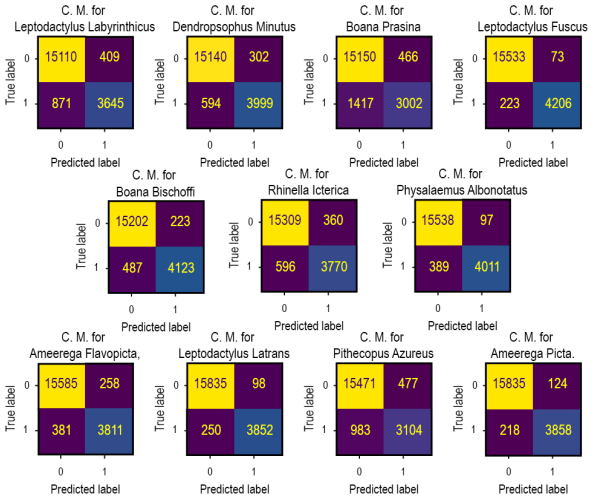

Figure 10.138: Confusion Matrix Mixed samples Fold 3 (EfficientNet)

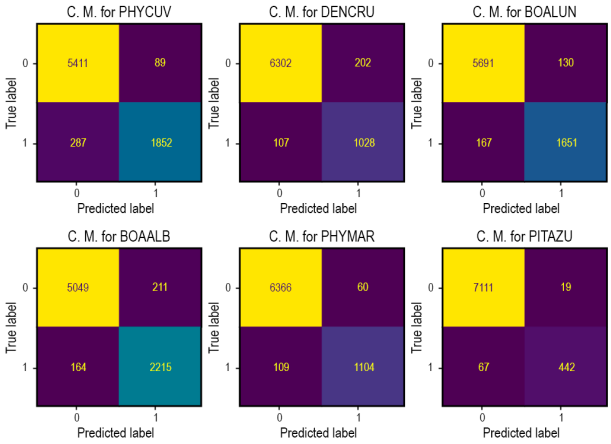

Figure 10.139: Confusion Matrix Anuraset Fold 4 (ResNet)

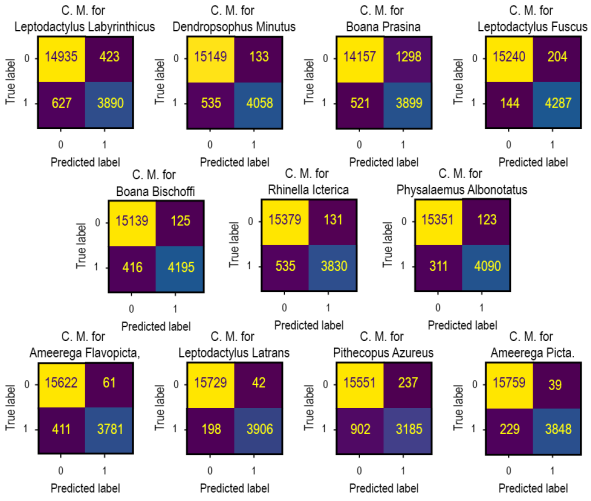

Figure 10.140: Confusion Matrix Mixed samples Fold 4 (ResNet)

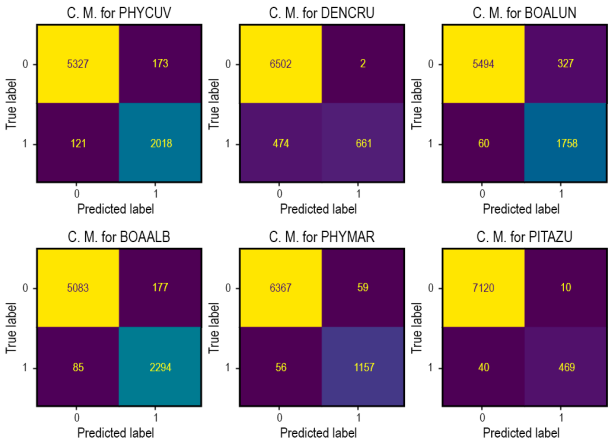

Figure 10.141: Confusion Matrix Anuraset Fold 4 (VGG)

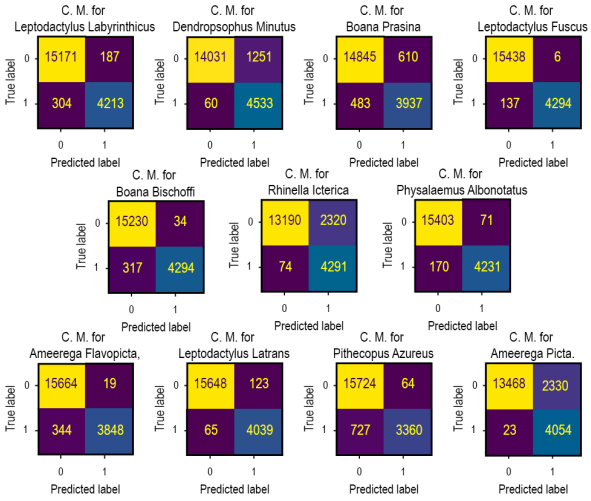

Figure 10.142: Confusion Matrix Mixed samples Fold 4 (VGG)

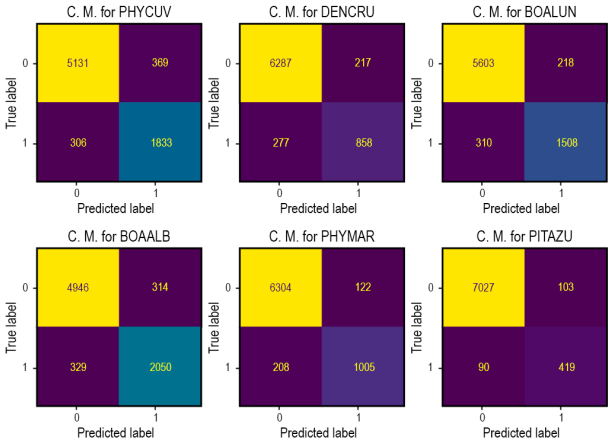

Figure 10.143: Confusion Matrix Anuraset Fold 4 (EfficientNet)

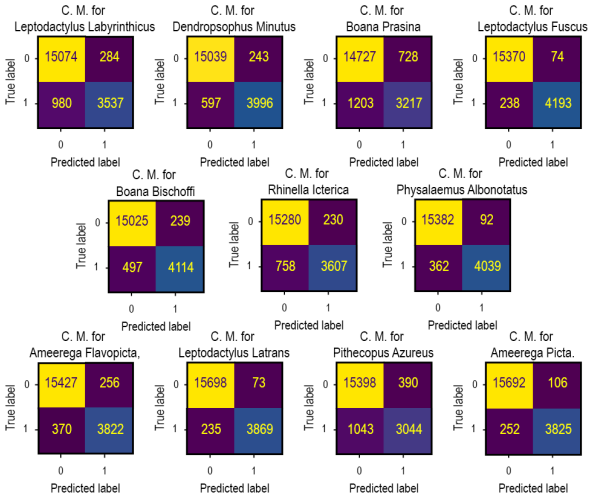

Figure 10.144: Confusion Matrix Mixed samples Fold 4 (EfficientNet)

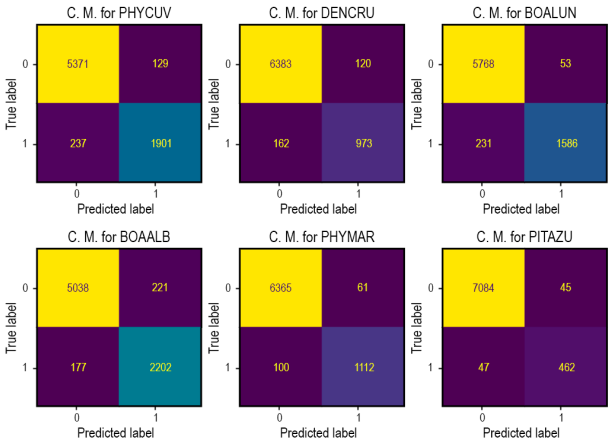

Figure 10.145: Confusion Matrix Anuraset Fold 5 (ResNet)

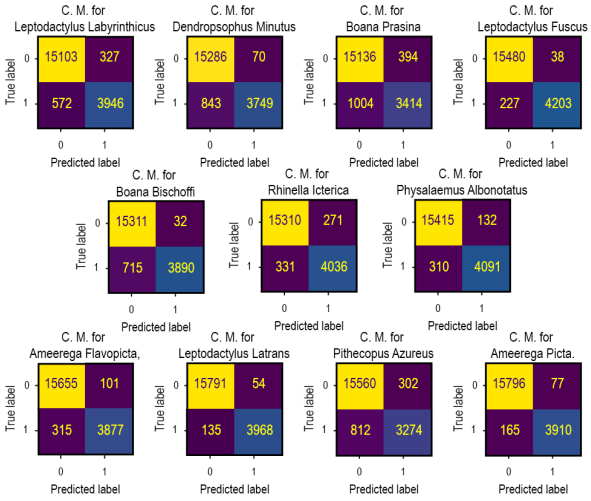

Figure 10.146: Confusion Matrix Mixed samples Fold 5 (ResNet)

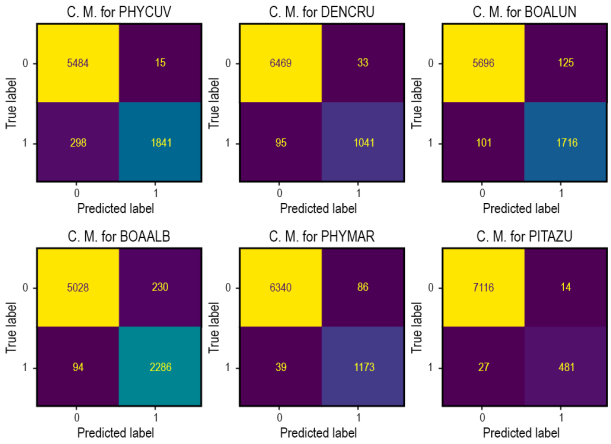

Figure 10.147: Confusion Matrix Anuraset Fold 5 (VGG)

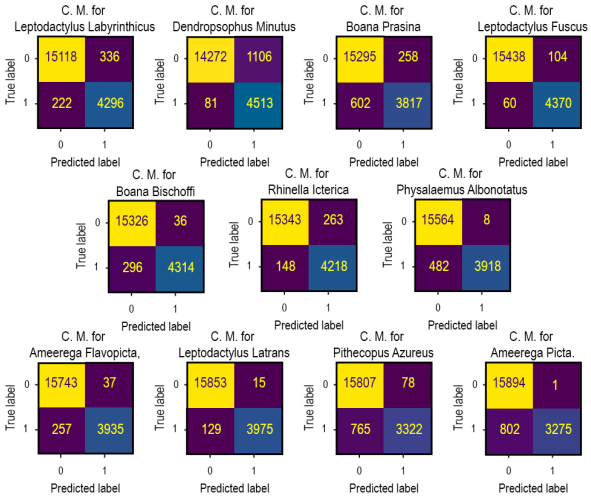

Figure 10.148: Confusion Matrix Mixed samples Fold 5 (VGG)

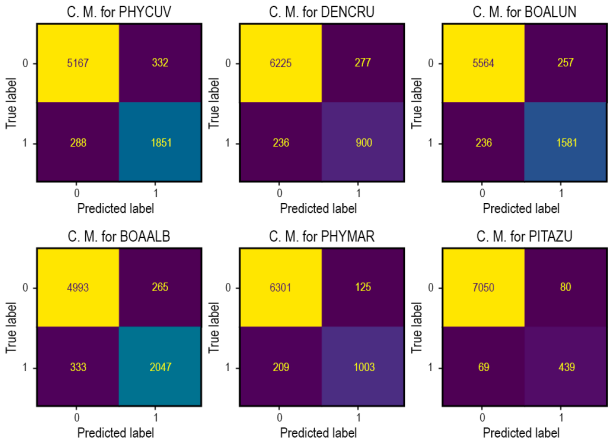

Figure 10.149: Confusion Matrix Anuraset Fold 5 (EfficientNet)

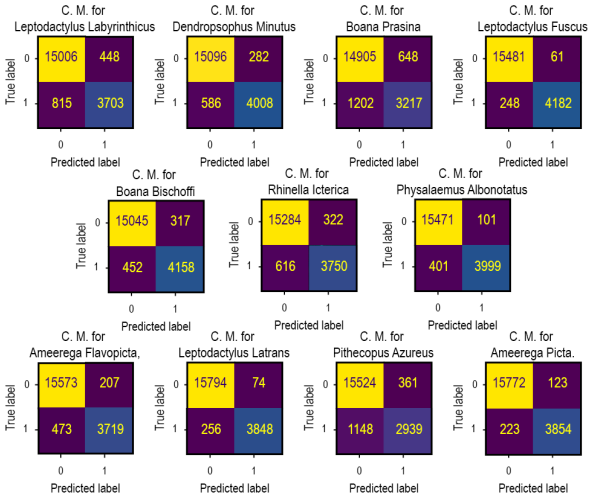

Figure 10.150: Confusion Matrix Mixed samples Fold 5 (EfficientNet)

Appendix 5 – Learning Curves and Confusion Matrices from experiments with no augmentations and no transfer learning

10.0.7 Learning curves.

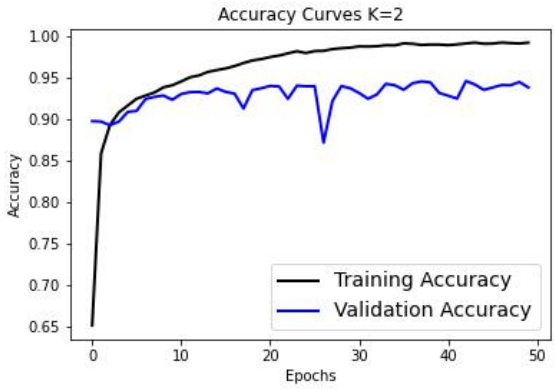

Figure 10.151: Accuracy Curve Fold 2 (ResNet)

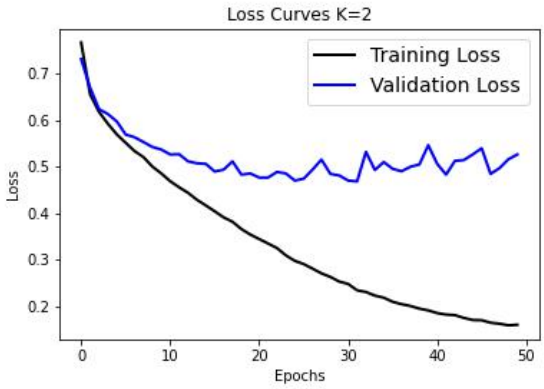

Figure 10.152: Loss Curve Fold 2 (ResNet)

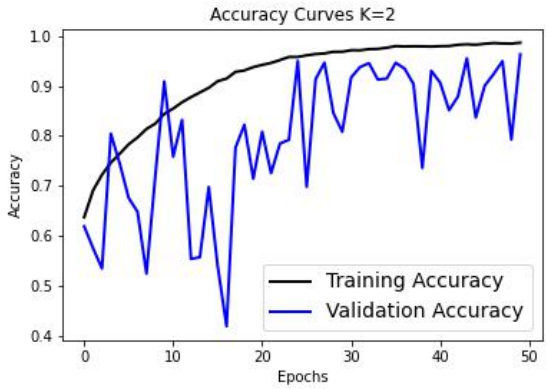

Figure 10.153: Accuracy Curve Fold 2 (VGG)

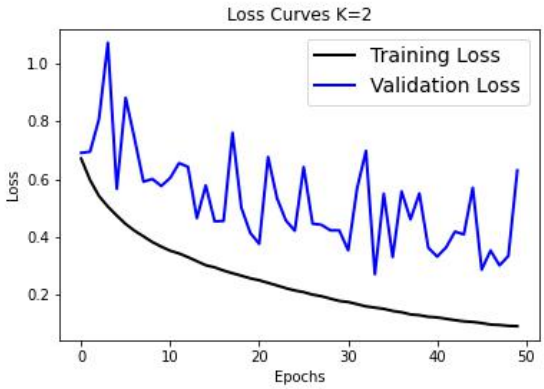

Figure 10.154: Loss Curve Fold 2 (VGG)

10.0.8 Confusion Matrices

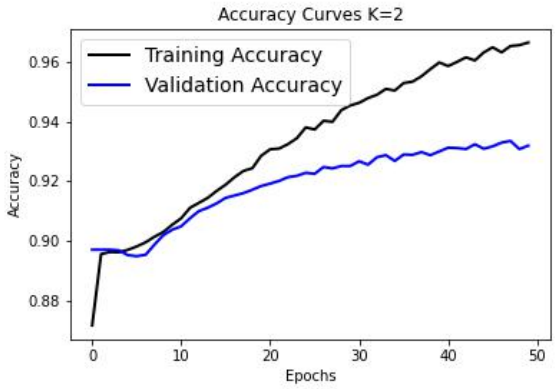

Figure 10.155: Accuracy Curve Fold 2 (EfficientNet)

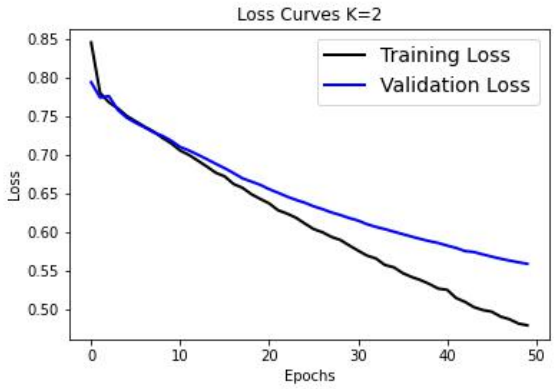

Figure 10.156: Loss Curve Fold 2 (EfficientNet)

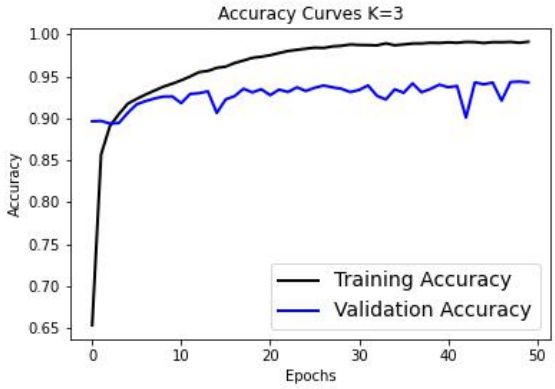

Figure 10.157: Accuracy Curve Fold 3 (ResNet)

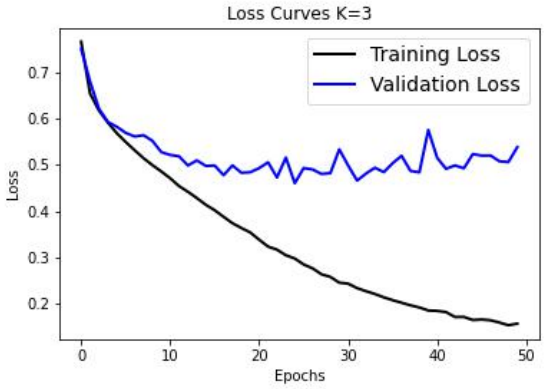

Figure 10.158: Loss Curve Fold 3 (ResNet)

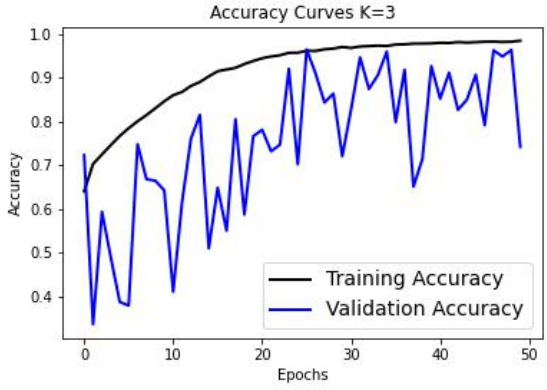

Figure 10.159: Accuracy Curve Fold 3 (VGG)

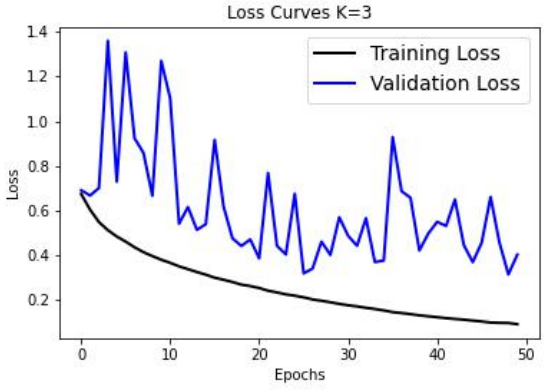

Figure 10.160: Loss Curve Fold 3 (VGG)

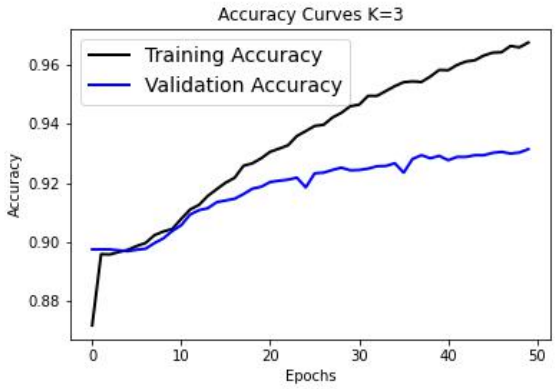

Figure 10.161: Accuracy Curve Fold 3 (EfficientNet)

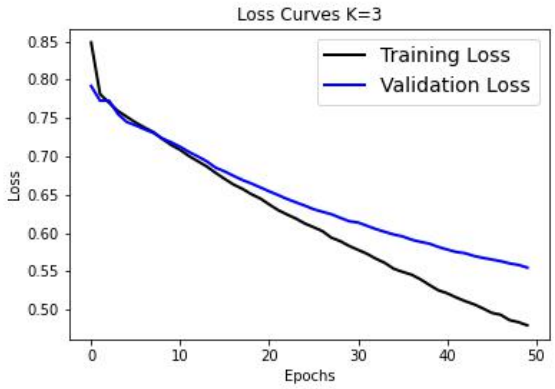

Figure 10.162: Loss Curve Fold 3 (EfficientNet)

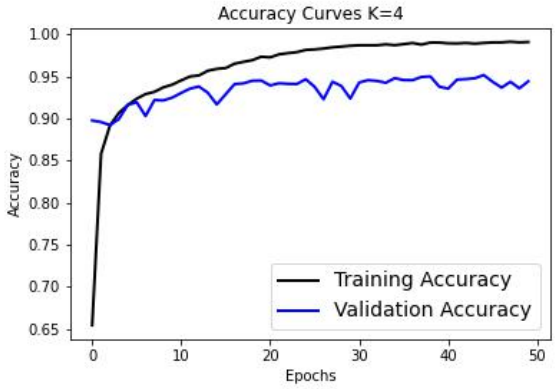

Figure 10.163: Accuracy Curve Fold 4 (ResNet)

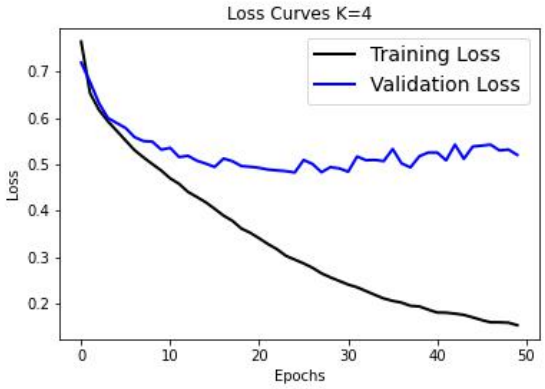

Figure 10.164: Loss Curve Fold 4 (ResNet)

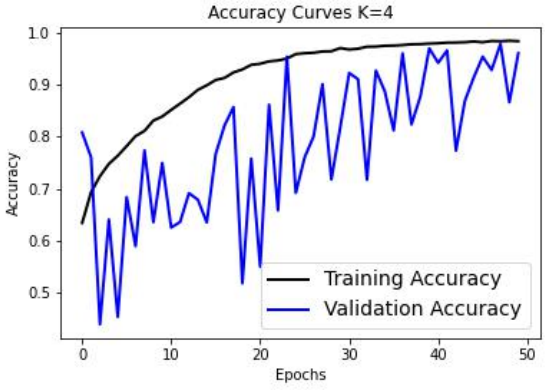

Figure 10.165: Accuracy Curve Fold 4 (VGG)

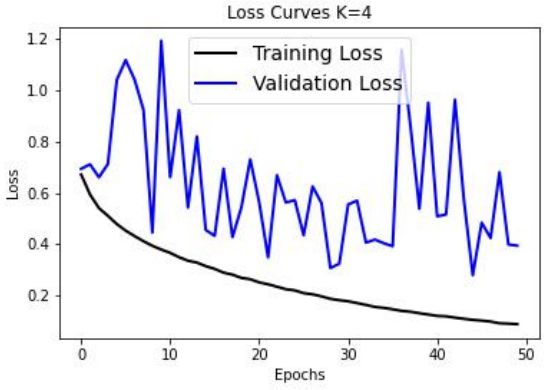

Figure 10.166: Loss Curve Fold 4 (VGG)

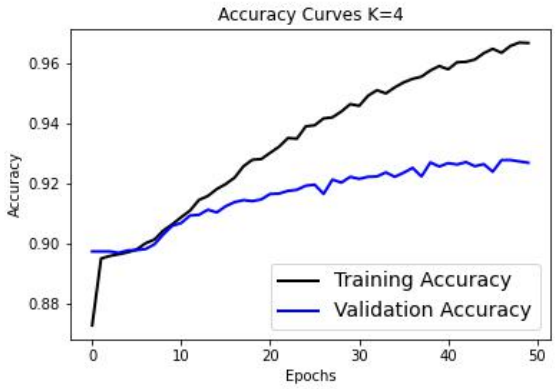

Figure 10.167: Accuracy Curve Fold 4 (EfficientNet)

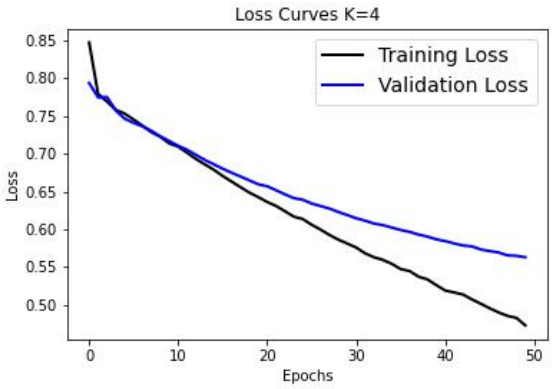

Figure 10.168: Loss Curve Fold 4 (EfficientNet)

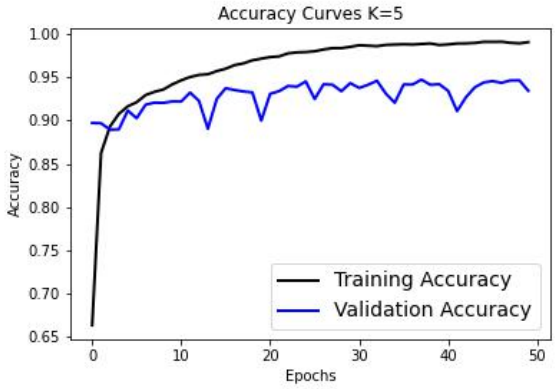

Figure 10.169: Accuracy Curve Fold 5 (ResNet)

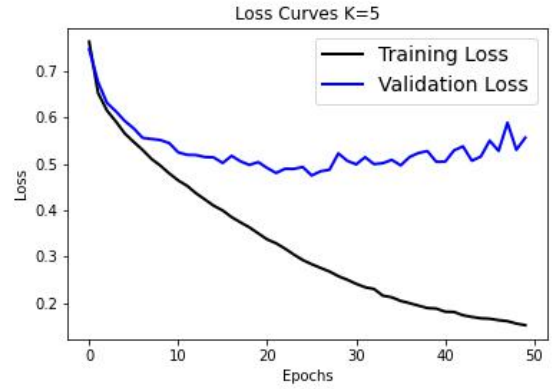

Figure 10.170: Loss Curve Fold 5 (ResNet)

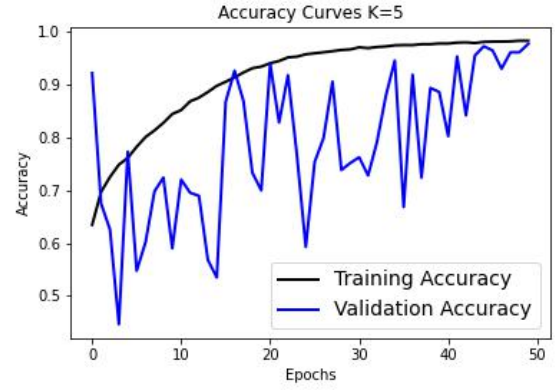

Figure 10.171: Accuracy Curve Fold 5 (VGG)

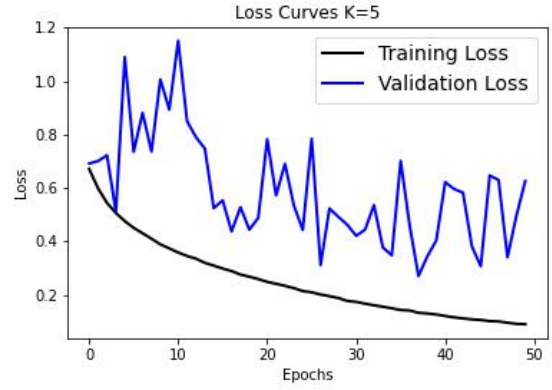

Figure 10.172: Loss Curve Fold 5 (VGG)

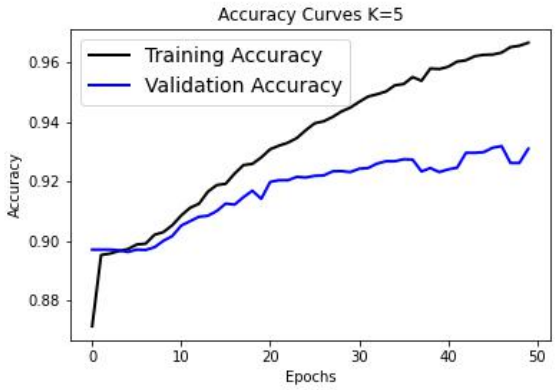

Figure 10.173: Accuracy Curve Fold 5 (EfficientNet)

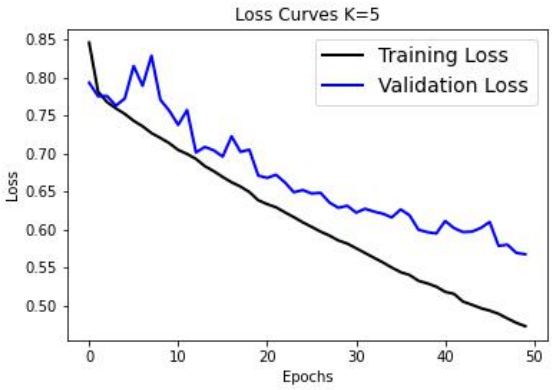

Figure 10.174: Loss Curve Fold 5 (EfficientNet)

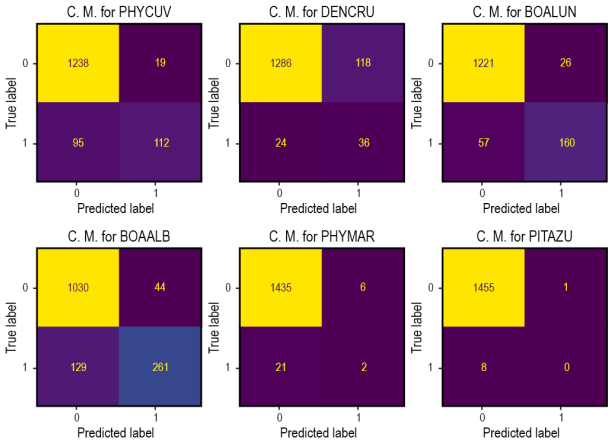

Figure 10.175: Confusion Matrix Anuraset Fold 2 (ResNet)

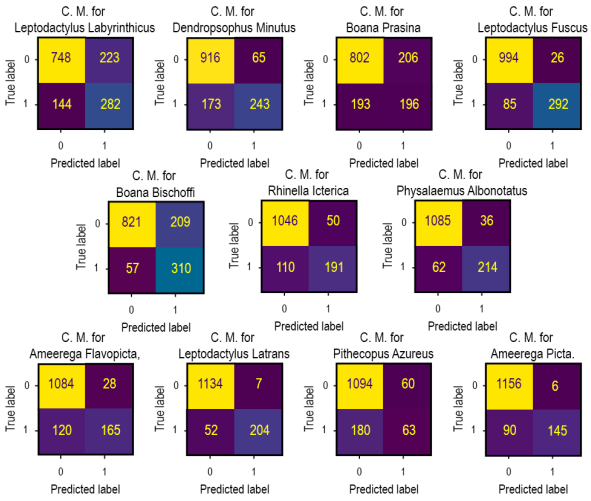

Figure 10.176: Confusion Matrix Mixed samples Fold 2 (ResNet)

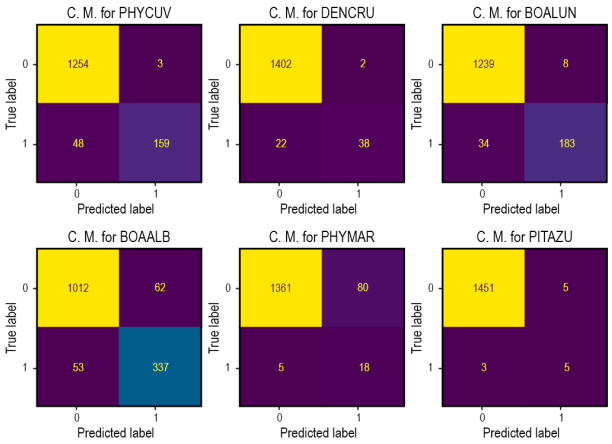

Figure 10.177: Confusion Matrix Anuraset Fold 2 (VGG)

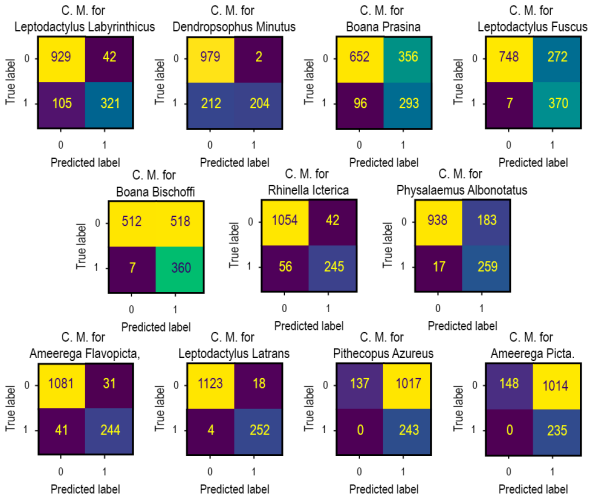

Figure 10.178: Confusion Matrix Mixed samples Fold 2 (VGG)

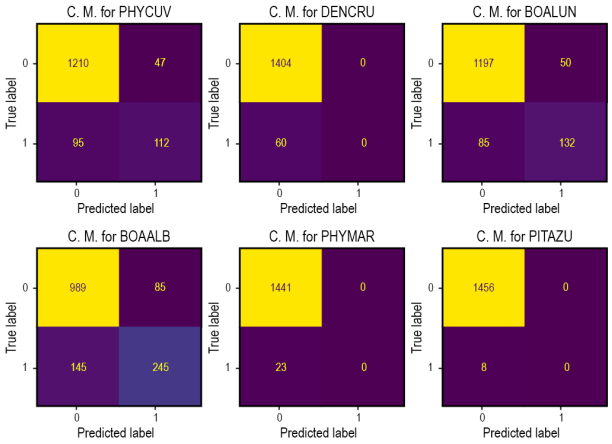

Figure 10.179: Confusion Matrix Anuraset Fold 2 (EfficientNet)

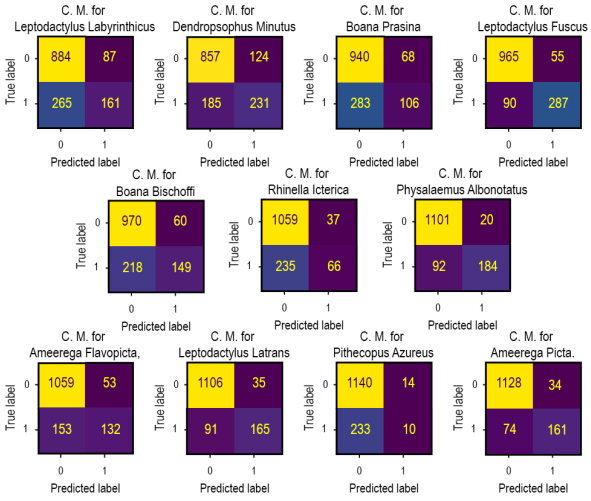

Figure 10.180: Confusion Matrix Mixed samples Fold 2 (EfficientNet)

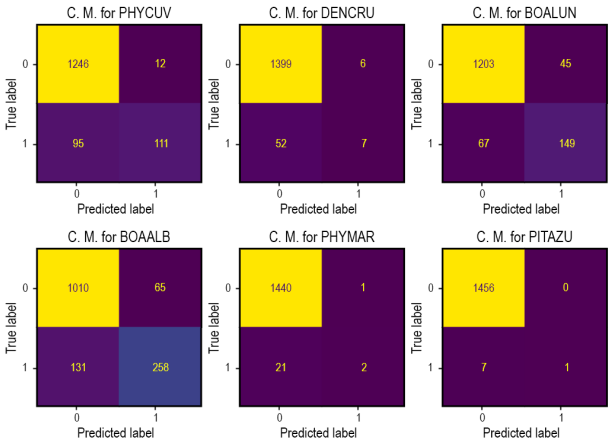

Figure 10.181: Confusion Matrix Anuraset Fold 3 (ResNet)

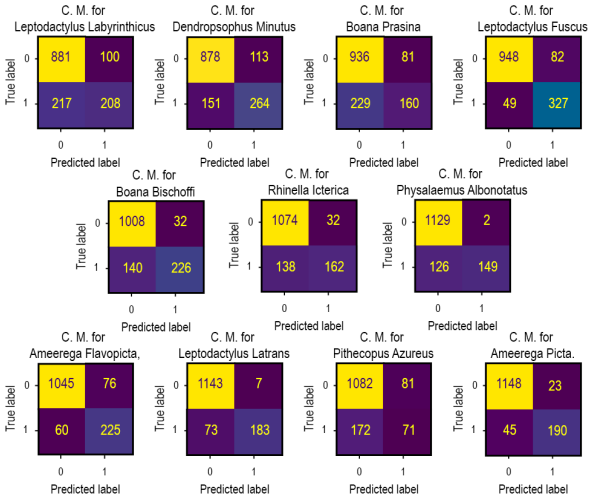

Figure 10.182: Confusion Matrix Mixed samples Fold 3 (ResNet)

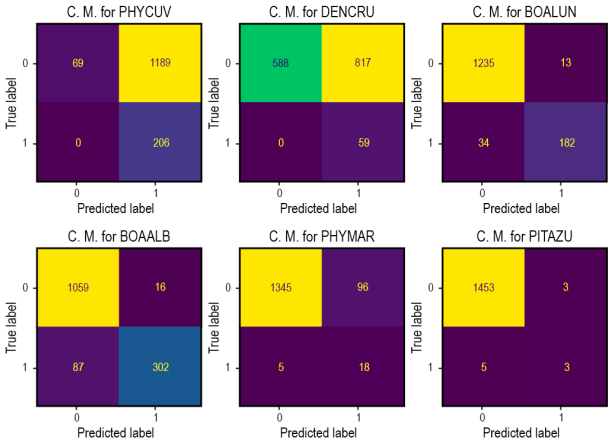

Figure 10.183: Confusion Matrix Anuraset Fold 3 (VGG)

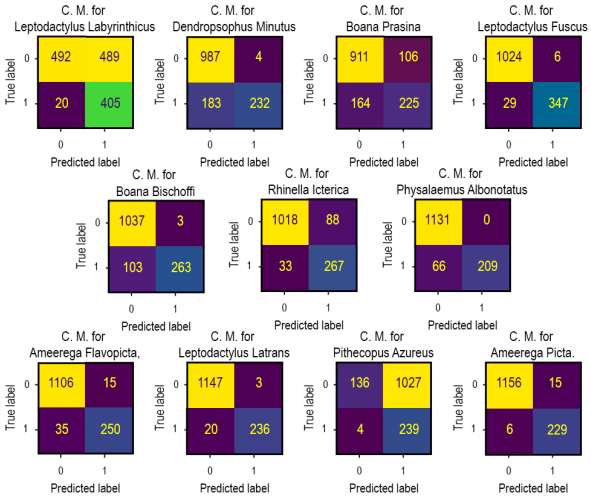

Figure 10.184: Confusion Matrix Mixed samples Fold 3 (VGG)

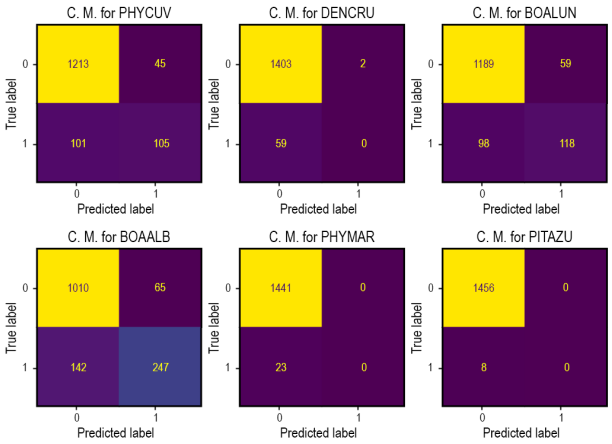

Figure 10.185: Confusion Matrix Anuraset Fold 3 (EfficientNet)

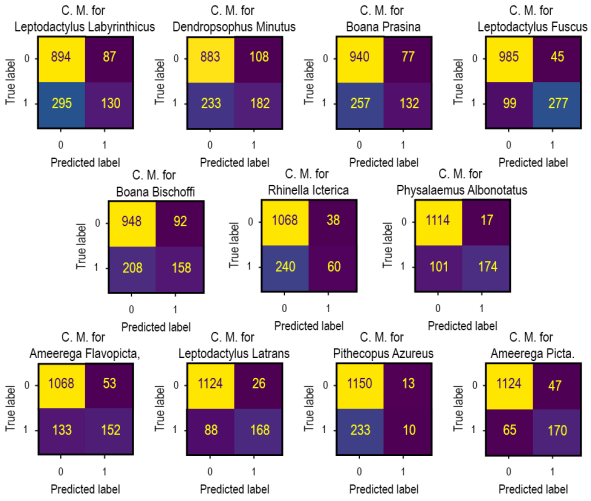

Figure 10.186: Confusion Matrix Mixed samples Fold 3 (EfficientNet)

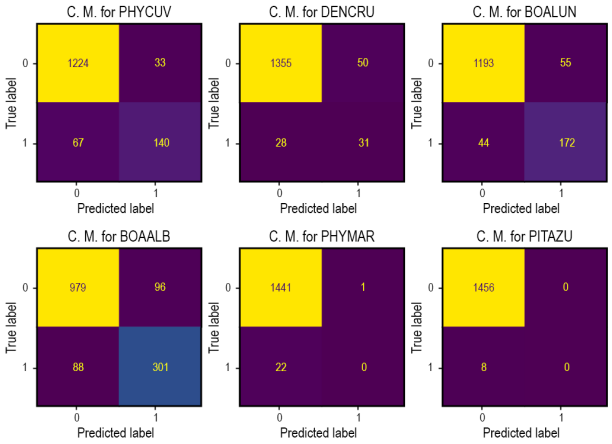

Figure 10.187: Confusion Matrix Anuraset Fold 4 (ResNet)

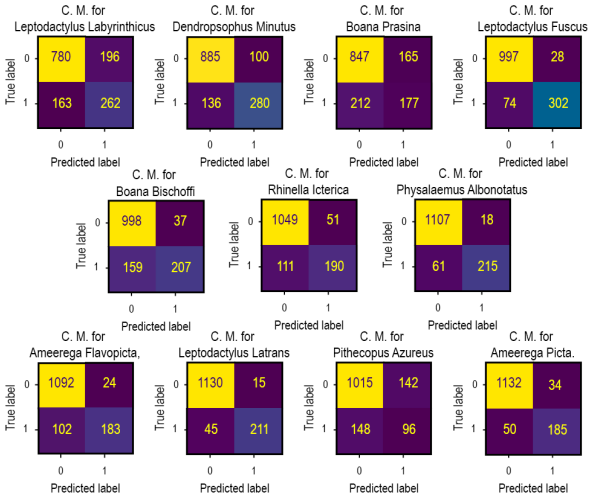

Figure 10.188: Confusion Matrix Mixed samples Fold 4 (ResNet)

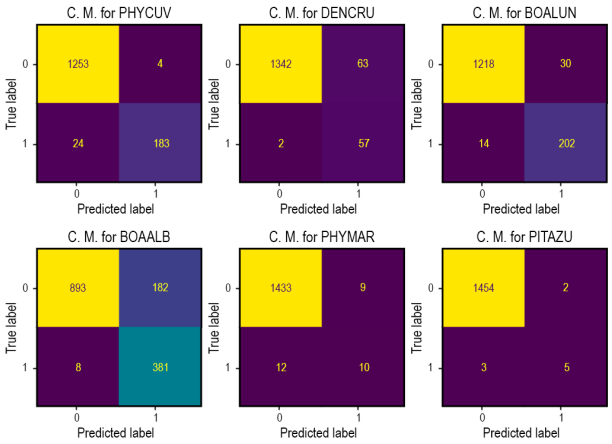

Figure 10.189: Confusion Matrix Anuraset Fold 4 (VGG)

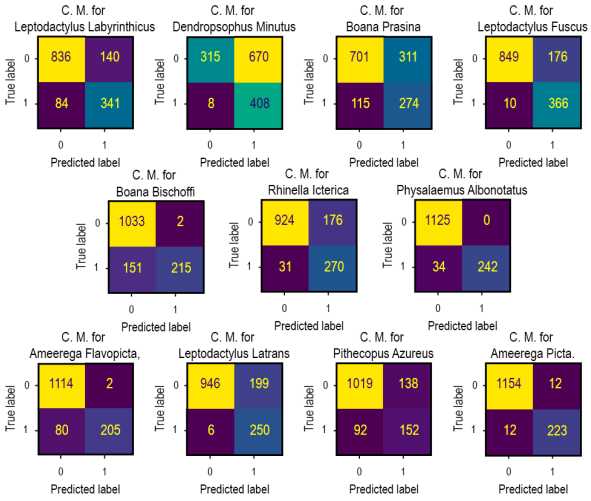

Figure 10.190: Confusion Matrix Mixed samples Fold 4 (VGG)

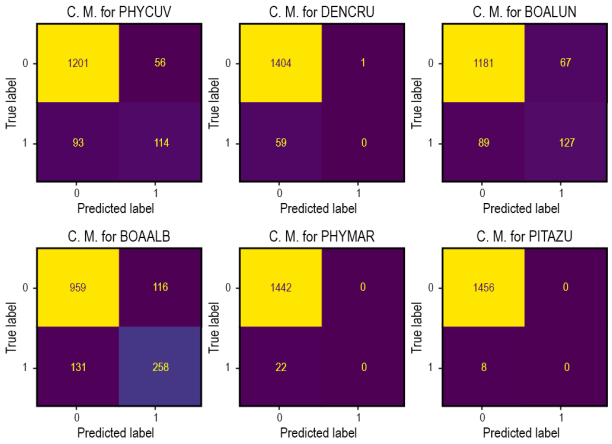

Figure 10.191: Confusion Matrix Anuraset Fold 4 (EfficientNet)

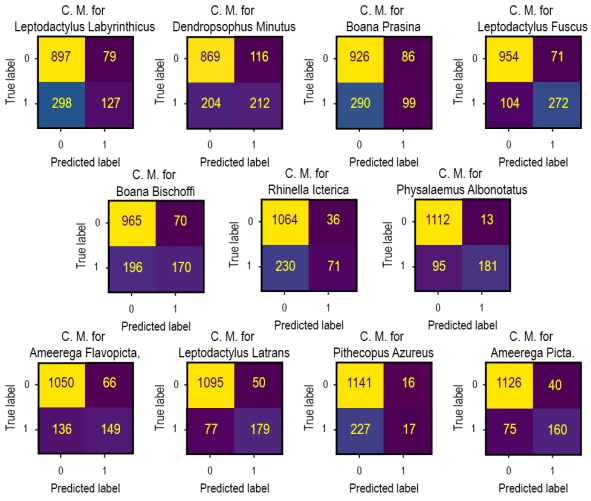

Figure 10.192: Confusion Matrix Mixed samples Fold 4 (EfficientNet)

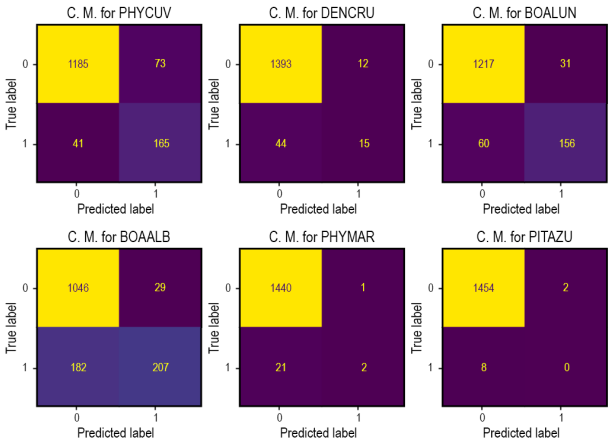

Figure 10.193: Confusion Matrix Anuraset Fold 5 (ResNet)

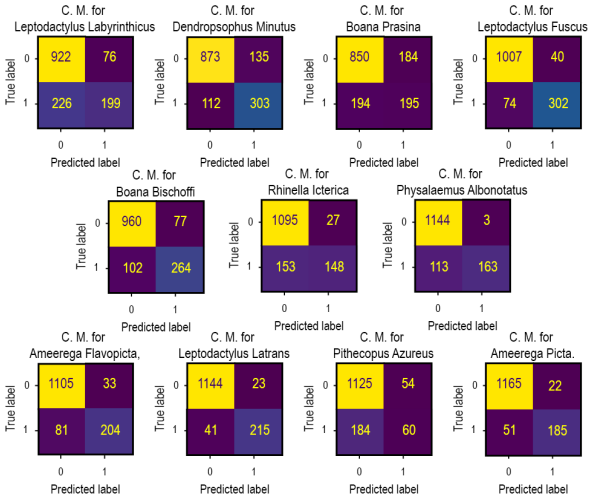

Figure 10.194: Confusion Matrix Mixed samples Fold 5 (ResNet)

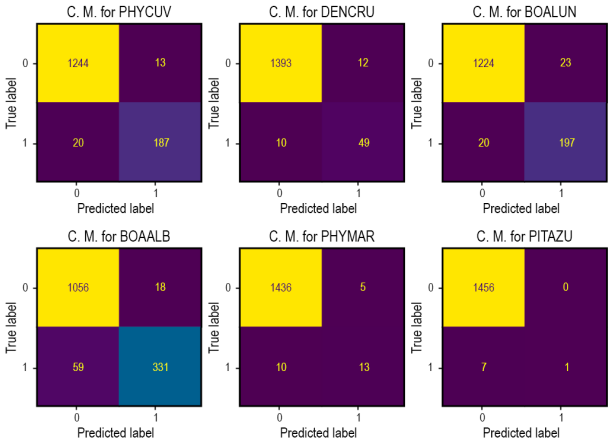

Figure 10.195: Confusion Matrix Anuraset Fold 5 (VGG)

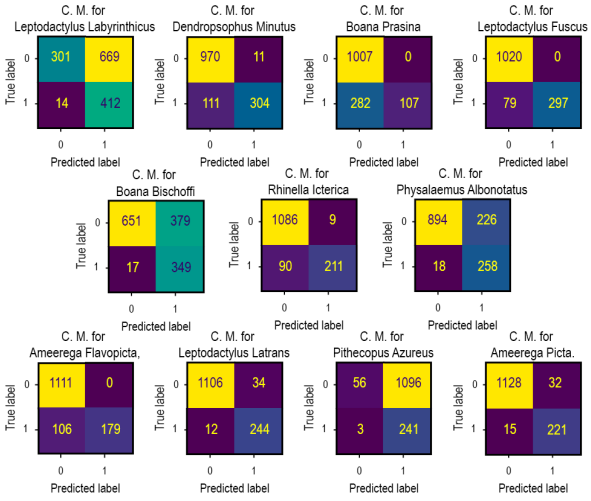

Figure 10.196: Confusion Matrix Mixed samples Fold 5 (VGG)

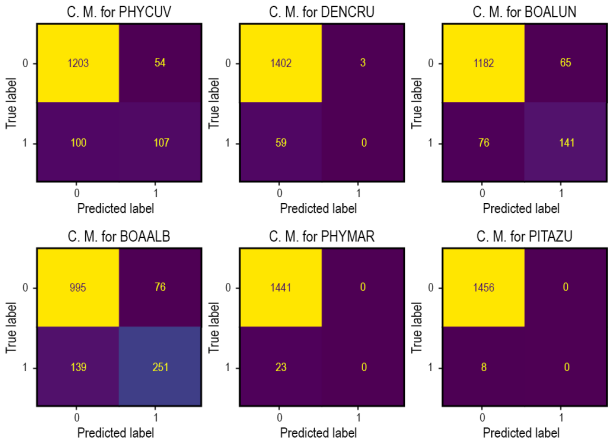

Figure 10.197: Confusion Matrix Anuraset Fold 5 (EfficientNet)

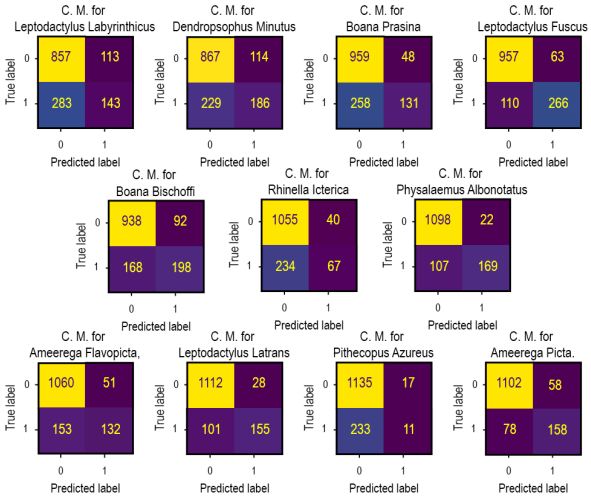

Figure 10.198: Confusion Matrix Mixed samples Fold 5 (EfficientNet)

Bibliography

[1] "The IUCN Red List of Threatened Species," 2023. [Online]. Available: https://www.iucnredlist.org

[2] E. Sanabria and L. Quiroga, "The body temperature of active desert anurans from hyper-arid environment of South America: The reliability of WorldClim for predicted body temperatures in anurans," *Journal of Thermal Biology*, vol. 85, p. 102398, 10 2019.

[3] K. Ceron, D. J. Santana, E. M. Lucas, J. J. Zocche, and D. B. Provete, "Climatic variables influence the temporal dynamics of an anuran metacommunity in a nonstationary way," *Ecology and Evolution*, vol. 10, no. 11, pp. 4630–4639, 6 2020.

[4] A. Ruiz-García, S. Roco, and M. Bullejos, "Sex Differentiation in Amphibians: Effect of Temperature and Its Influence on Sex Reversal," *Sexual Development*, vol. 15, no. 1-3, pp. 157–167, 2021.

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

[6] D. Stowell, "Computational bioacoustics with deep learning: a review and roadmap," *PeerJ*, vol. 10, p. e13152, 3 2022.

[7] L. Nanni, G. Maguolo, and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics*, vol. 57, 2020.

[8] Y. Sun, T. M. Maeda, C. Solis-Lemus, D. Pimentel-Alarcon, and Z. Burivalova, "Classification of animal sounds in a hyperdiverse rainforest using Convolutional Neural Networks," 11 2021.

[9] Lucas Ferreira-Paiva, Elizabeth Alfaro-Espinoza, Vinicius M. Almeida, Leonardo B. Felix, and Rodolpho V. A. Neves, "A Survey of Data Augmentation for Audio Classification," 10 2022.

[10] J. S. Cañas, M. P. Toro-Gómez, L. S. M. Sugai, H. D. Benítez Restrepo, J. Rudas, B. Posso Bautista, L. F. Toledo, S. Dena, A. H. R. Domingos, F. L. de Souza, S. Neckel-Oliveira, A. da Rosa, V. Carvalho-Rocha, J. V. Bernardy, J. L. M. M. Sugai, C. E. dos Santos, R. P. Bastos, D. Llusia, and J. S. Ulloa, "A dataset for benchmarking Neotropical anuran calls identification in passive acoustic monitoring," *Scientific Data*, vol. 10, no. 1, p. 771, 11 2023.

[11] N. A and R. Rajan, "Multi-label Bird Species Classification Using Hierarchical Attention Framework," in *2022 IEEE 19th India Council International Conference (INDICON)*. IEEE, 11 2022, pp. 1–6.

[12] J. Xie, R. Zeng, C. Xu, J. Zhang, and P. Roe, "Multi-Label Classification of Frog Species via Deep Learning," in *2017 IEEE 13th International Conference on e-Science (e-Science)*. IEEE, 10 2017, pp. 187–193.

[13] I. Moummad, N. Farrugia, R. Serizel, J. Froidevaux, and V. Lostanlen, "Mixture of Mixups for Multi-label Classification of Rare Anuran Sounds," *arXiv preprint arXiv:2403.09598*, 2024.

[14] C. Román-Palacios and J. J. Wiens, "Recent responses to climate change reveal the drivers of species extinction and survival," *Proceedings of the National Academy of Sciences*, vol. 117, no. 8, pp. 4211–4217, 2 2020.

[15] "Global Warming and Endangered Species Initiative." [Online]. Available: https://www.biologicaldiversity.org/campaigns/global_warming_and_endangered_species/index.html

[16] S. Kahl, "Identifying Birds by Sound: Large-scale Acoustic Event Recognition for Avian Activity Monitoring," *Scientific series of media computer science dissertations*, 12 2019.

[17] M. A. Acevedo, C. J. Corrada-Bravo, H. Corrada-Bravo, L. J. Villanueva-Rivera, and T. M. Aide, "Automated classification of bird and amphibian calls using machine learning: A comparison of methods," *Ecological Informatics*, vol. 4, no. 4, 2009.

[18] C. Hof, M. B. Araújo, W. Jetz, and C. Rahbek, "Additive threats from pathogens, climate and land-use change for global amphibian diversity," *Nature*, vol. 480, no. 7378, pp. 516–519, 12 2011.

[19] D. Llusia, J. Sebastián Ulloa, L. Sayuri, M. Sugai, H. Benítez, R. Pereira Bastos, and D. Fernando López, "Machine listening to monitor climate change impacts on neotropical amphibians," 2021.

[20] T. Dema, L. Zhang, M. Towsey, A. Truskinger, S. Sherub, Kinley, J. Zhang, M. Brereton, and P. Roe, "An Investigation into Acoustic Analysis Methods for Endangered Species Monitoring: A Case of Monitoring the Critically Endangered White-Bellied Heron in Bhutan," in *2017 IEEE 13th International Conference on e-Science (e-Science)*. IEEE, 10 2017, pp. 177–186.

[21] H. Gan, J. Zhang, M. Towsey, A. Truskinger, D. Stark, B. van Rensburg, Y. Li, and P. Roe, "Recognition of Frog Chorusing with Acoustic Indices and Machine Learning," in *2019 15th International Conference on eScience (eScience)*. IEEE, 9 2019, pp. 106–115.

[22] W. Penar, A. Magiera, and C. Klocek, "Applications of bioacoustics in animal ecology," *Ecological Complexity*, vol. 43, p. 100847, 8 2020.

[23] R. R. K. V. S. N., J. Montgomery, S. Garg, and M. Charleston, "Bioacoustics Data Analysis – A Taxonomy, Survey and Open Challenges," *IEEE Access*, vol. 8, pp. 57 684–57 708, 2020.

[24] C. Erbe and M. L. Dent, "What is animal bioacoustics," *J. Acoust. Soc. Am*, vol. 139, no. 4, p. 2004, 2016.

[25] J. Miller, L. Kloepper, G. Potty, A. Spivack, S. D'Hondt, C. Turner, and A. Simmons, "The effects of pH on acoustic transmission loss in an estuary," 2015, p. 005001.

[26] L. Kloepper and A. Simmons, "Bioacoustic monitoring contributes to an understanding of climate change," *Acoustics Today*, vol. 10, pp. 8–15, 5 2014.

[27] S. Tavasoli, "The Importance of Machine Learning for Data Scientists," https://www.simplilearn.com/importance-of-machine-learning-for-data-scientists-article, 7 2023.

[28] L. M. Munger, S. M. Wiggins, S. E. Moore, and J. A. Hildebrand, "North Pacific right whale (Eubalaena japonica) seasonal and diel calling patterns from long-term acoustic recordings in the southeastern Bering Sea, 2000–2006," *Marine Mammal Science*, vol. 24, no. 4, pp. 795–814, 10 2008.

[29] D. Gil, M. Honarmand, J. Pascual, E. Pérez-Mena, and C. Macías Garcia, "Birds living near airports advance their dawn chorus and reduce overlap with aircraft noise," *Behavioral Ecology*, vol. 26, no. 2, pp. 435–443, 2015.

[30] M. Sankupellay, M. Towsey, A. Truskinger, and P. Roe, "Visual Fingerprints of the Acoustic Environment: The Use of Acoustic Indices to Characterise Natural Habitats," in *2015 Big Data Visual Analytics (BDVA)*. IEEE, 9 2015, pp. 1–8.

[31] M. Depraetere, S. Pavoine, F. Jiguet, A. Gasc, S. Duvail, and J. Sueur, "Monitoring animal diversity using acoustic indices: Implementation in a temperate woodland," *Ecological Indicators*, vol. 13, no. 1, pp. 46–54, 2 2012.

[32] D. Pluščec and J. Šnajder, "Data Augmentation for Neural NLP," 5 2023.

[33] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A survey of data augmentation approaches for NLP," *arXiv preprint arXiv:2105.03075*, 2021.

[34] K. Doshi, "Audio Deep Learning Made Simple (Part 3): Data Preparation and Augmentation," https://towardsdatascience.com/audio-deep-learning-made-simple-part-3-data-preparation-and-augmentation-24c6e1f6b52, 12 2021.

[35] S. Lehman, "Delay," 1996.

[36] U. Zolzer, *DAFX: Digital Audio Effects*, 2nd ed. Wiley Publishing, 2011.

[37] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," in *IEEE International Conference on Acoustics Speech and Signal Processing*. IEEE, 1993, pp. 554–557.

[38] J. Driedger and M. Müller, "A Review of Time-Scale Modification of Music Signals," *Applied Sciences*, vol. 6, no. 2, p. 57, 2 2016.

[39] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, A. Munawar, B. J. Ko, N. Greco, and R. Tachibana, "Shuffling and mixing data augmentation for environmental sound classification," 2019.

[40] J. O. Smith, *Physical Audio Signal Processing*, 2010th ed. http://ccrma.stanford.edu/~jos/pasp/, 2010.

[41] I. Jordan, "Audiomentations," 3 2023.

[42] A. k. Sarkar and Z.-H. Tan, "Vocal Tract Length Perturbation for Text-Dependent Speaker Verification With Autoregressive Prediction Coding," *IEEE Signal Processing Letters*, vol. 28, pp. 364–368, 2021.

[43] Q. Chaudhari, "Additive White Gaussian Noise (AWGN)." [Online]. Available: https://wirelesspi.com/additive-white-gaussian-noise-awgn/

[44] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello, "Scaper: A library for soundscape synthesis and augmentation," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 10 2017, pp. 344–348.

[45] L. Chaparro, *Signals and Systems Using MATLAB*. Elsevier, 2015.

[46] G. S. Kendall, C. Haworth, and R. F. Cádiz, "Sound Synthesis with Auditory Distortion Products," *Computer Music Journal*, vol. 38, no. 4, pp. 5–23, 12 2014.

[47] 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," 4 2019.

[48] X. Song, Z. Wu, Y. Huang, D. Su, and H. Meng, "SpecSwap: A Simple Data Augmentation Method for End-to-End Speech Recognition," in *Interspeech 2020*. ISCA: ISCA, 10 2020, pp. 581–585.

[49] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Detection," 4 2016.

[50] G. Zhou, Y. Chen, and C. Chien, "On the analysis of data augmentation methods for spectral imaged based heart sound classification using convolutional neural networks," *BMC Medical Informatics and Decision Making*, vol. 22, no. 1, p. 226, 8 2022.

[51] "Image processing with Python, NumPy," https://note.nkmk.me/en/python-numpy-image-processing/#::text=By%20reading%20the%20image%20as,images%2C%20concatenate%20images%2C%20et 10 2020.

[52] "TAO Toolkit Use Cases - 3. Quick prototyping with a small dataset," https://developer.nvidia.cn/tao-toolkit-usecases-whitepaper/3-quick-prototyping-small-dataset.

[53] F. Wagner, A. Eltner, and H.-G. Maas, "River water segmentation in surveillance camera images: A comparative study of offline and online augmentation using 32 CNNs," *International Journal of Applied Earth Observation and Geoinformation*, vol. 119, p. 103305, 5 2023.

[54] A. Géron, *Hands-on Machine Learning*, 2017, vol. 53, no. 9.

[55] J. Howard and S. Gugger, *Deep Learning for Coders with fastai & PyTorch (BOOK)*, 2020, vol. 66, no. 3.

[56] C. A. Goodfellow lan, Bengio Yoshua, "Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books," 2016.

[57] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*, 2023.

[58] "Machine Learning Google for Developers," https://developers.google.com/machine-learning/crash-course?hl=es-419.

[59] W. D. Kissling, R. Walls, A. Bowser, M. O. Jones, J. Kattge, D. Agosti, J. Amengual, A. Basset, P. M. van Bodegom, J. H. C. Cornelissen, E. G. Denny, S. Deudero, W. Egloff, S. C. Elmendorf, E. Alonso García, K. D. Jones, O. R. Jones, S. Lavorel, D. Lear, L. M. Navarro, S. Pawar, R. Pirzl, N. Rüger, S. Sal, R. Salguero-Gómez, D. Schigel, K.-S. Schulz, A. Skidmore, and R. P. Guralnick, "Towards global data products of Essential Biodiversity Variables on species traits," *Nature Ecology & Evolution*, vol. 2, no. 10, pp. 1531–1540, 9 2018.

[60] E. Jukes, "Encyclopedia of Machine Learning and Data Mining (2nd edition)," *Reference Reviews*, vol. 32, no. 7/8, pp. 3–4, 9 2018.

[61] V. H. Phung and E. J. Rhee, "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets," *Applied Sciences*, vol. 9, no. 21, p. 4500, 10 2019.

[62] "Fully connected layer - MATLAB," https://www.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fully

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2016, pp. 770–778.

[64] S. Bangar, "Resnet Architecture Explained - Siddhesh Bangar - Medium," https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d, 7 2022.

[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations (ICLR 2015)*. Computational and Biological Learning Society, 2015, pp. 1–14.

[66] K. Abeywardana, "Very Deep Convolution Networks For Large-Scale Image Recognition: A Brief Summary," https://medium.com/@kdwa2404/very-deep-convolution-networks-for-large-scale-image-recognition-a-brief-summary-9ff28a54652d, 1 2024.

[67] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, 2019.

[68] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 10 2010.

[69] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220311693

[70] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY: Springer New York, 2009.

[71] S. Afaq and S. Rao, "Significance Of Epochs On Training A Neural Network," *International Journal of Scientific & Technology Research*, vol. 9, no. 06, 2020.

[72] P. Antoniadis, "Hidden Layers in a Neural Network Baeldung on Computer Science," https://www.baeldung.com/cs/hidden-layers-neural-network, 3 2024.

[73] K. P. Murphy, *Machine learning: a probabilistic perspective (adaptive computation and machine learning series)*, 2012, vol. 621485037.

[74] N. Ketkar, *Deep Learning with Python - A Hands-on Introduction*, 2017.

[75] "sklearn.modelselection.StratifiedKFold," https://scikit-learn.org/stable/modules/generated/sklearn.modelse

[76] T. Bradberry, "iterative-stratification," https://github.com/trent-b/iterative-stratification, 2018.

[77] K. Sechidis, G. Tsoumakas, and I. Vlahavas, "On the Stratification of Multi-label Data," 2011, pp. 145–158.

[78] A. Tharwat, "Classification assessment methods," *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, 1 2021.

[79] C. Wu, Y.-F. Li, J. Li, J. Xu, and P. Bouvry, "Trustworthy AI: Deciding What to Decide," *ArXiv*, vol. abs/2311.12604, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265309238

[80] K. Team, "Keras documentation: Probabilistic losses," https://keras.io/api/losses/probabilisticlosses/.

[81] J. Brownlee, "Loss and Loss Functions for Training Deep Learning Neural Networks," https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/#comments, 10 2019.

[82] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010.

[83] "sklearn.metrics.multilabelconfusionmatrix," https://scikit-learn.org/stable/modules/generated/sklearn.metr

[84] C. Kim, M. Shin, A. Garg, and D. Gowda, "Improved Vocal Tract Length Perturbation for a State-of-the-Art End-to-End Speech Recognition System," in *Interspeech 2019*. ISCA: ISCA, 9 2019, pp. 739–743.

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

[86] K. Palanisamy, D. Singhania, and A. Yao, "Rethinking CNN Models for Audio Classification," 7 2020.

[87] 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, 12 2021.

[88] Y. Yang, L. Zhang, M. Du, J. Bo, H. Liu, L. Ren, X. Li, and M. J. Deen, "A comparative analysis of eleven neural networks architectures for small datasets of lung images of COVID-19 patients toward improved clinical decisions," *Computers in Biology and Medicine*, vol. 139, p. 104887, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482521006818

[89] K. W. Gunawan, A. A. Hidayat, T. W. Cenggoro, and B. Pardamean, "A Transfer Learning Strategy for Owl Sound Classification by Using Image Classification Model with Audio Spectrogram," *International Journal on Electrical Engineering and Informatics*, vol. 13, no. 3, pp. 546–553, 9 2021.

[90] A. Khalighifar, R. M. Brown, J. Goyes Vallejos, and A. T. Peterson, "Deep learning improves acoustic biodiversity monitoring and new candidate forest frog species identification (genus Platymantis) in the Philippines," *Biodiversity and Conservation*, vol. 30, no. 3, pp. 643–657, 3 2021.

[91] L. F. Toledo, "Fonoteca Neotropical Jacques Vielliard (FNJV)," 8 2016. [Online]. Available: www.ib.unicamp.br/fnjv/

[92] "List of Call and Video Files on AmphibiaWeb." [Online]. Available: https://amphibiaweb.org/lists/sound.shtml

[93] K. Doshi, "Audio deep learning made simple: Sound classification, step-by-step," 5 2021. [Online]. Available: https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5

[94] C. Chen and M. Hira, "Audio Resampling." [Online]. Available: https://pytorch.org/audio/main/tutorials/audio_resampling_tutorial.html#sphx-glr-tutorials-audio-resampling-tutorial-py

[95] "Imbalanced data | machine learning | google for developers." [Online]. Available: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

[96] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time Series Data Augmentation for Deep Learning: A Survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. California: International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 4653–4660.

[97] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 12 2022.

[98] T. Kumar, M. Turab, A. Mileo, M. Bendechache, and T. Saber, "AudRandAug: Random Image Augmentations for Audio Classification," 6 2023.

[99] "CS231N: Deep Learning for Computer Vision." [Online]. Available: http://vision.stanford.edu/teaching/cs231n/

[100] C. van Leeuwen, "Why EfficientNet is not very computationally efficient," 7 2021. [Online]. Available: https://medium.com/@casparvanleeuwen/why-efficientnet-is-not-very-computationally-efficient-505fc08087b9

[101] E. Randellini, "Image classification: ResNet vs EfficientNet vs EfficientNet_v2 vs Compact Convolutional Transformers," 1 2023. [Online]. Available: https://medium.com/@enrico.randellini/image-classification-resnet-vs-efficientnet-vs-efficientnet-v2-vs-compact-convolutional-c205838bbf49

[102] O. A. Montesinos López, A. Montesinos López, and J. Crossa, "Overfitting, Model Tuning, and Evaluation of Prediction Performance," in *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Cham: Springer International Publishing, 2022, pp. 109–139.