



# UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

## **Bird Identification with Advanced Deep Learning Techniques for Biodiversity Conservation.**

Trabajo de integración curricular presentado como requisito para la  
obtención del título de ingeniero en tecnologías de la información

### **Autor:**

Farinango Romero Ricardo Joseph

### **Tutor:**

Astudillo León Juan Pablo, Ph.D.

Urcuquí, Mayo 2024

# Autoría

Yo, **Ricardo Joseph Farinango Romero**, con cédula de identidad 1727170035, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Mayo del 2024

---

Ricardo Joseph Farinango Romero

CI: 1727170035

# Autorización de publicación

Yo, **Ricardo Joseph Farinango Romero**, con cédula de identidad 1727170035, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Mayo del 2024

---

Ricardo Joseph Farinango Romero

CI: 1727170035

# Dedication

To my beloved parents, who have been my unwavering support and source of strength. I thank my mother, Ximena, for constantly encouraging me to keep moving forward and my father, José, for guiding me with wisdom. I express my heartfelt gratitude to my brothers, Gabriel and Pablo, for being fundamental pillars in my life. I dedicate this achievement to all those who believed in me and were essential to this journey.

Ricardo Joseph Farinango Romero



# Acknowledgment

I want to express my sincere gratitude to my advisor, Juan Pablo Astudillo León, Ph.D., for his constant help throughout the development process of this project and for allowing me to work with him on this thesis. I also thank Professor Maria Gabriela Cajamarca for guiding me in critical aspects that complemented the project's development. In addition, I would like to highlight the invaluable collaboration of Professor Gaby Arévalo, who helped me establish a solid foundation to move forward with the thesis. I am grateful to the Charles Darwin Foundation for providing helpful information. I would also like to thank my girlfriend Ivonne for giving me unconditional support throughout the trip. My gratitude is extended to each family member for their support and trust. I especially want to thank my friends, Galo, Micky, and Brandon, who always supported me in achieving my goals. Every outing with them was and will always encourage us to keep going and improve ourselves. I also thank all the people I met along the way, those with whom I shared good and bad moments and who were always willing to listen to me.

Ricardo Joseph Farinango Romero

# Resumen

El objetivo principal de esta tesis es contribuir a la conservación de las Islas Galápagos a través de la educación. Para lograr esto, se ha desarrollado una aplicación que puede ejecutarse tanto en dispositivos web como móviles, siendo este último el punto destacado del proyecto. La tesis presenta dos contribuciones fundamentales: En primer lugar, se realizó una exhaustiva búsqueda de aves de Galápagos utilizando tres fuentes distintas para generar un conjunto de datos de entrenamiento, ya que no existía para nuestro enfoque. Estos datos, obtenidos y etiquetados cuidadosamente, se basan en la guía de la página principal de la fundación de Charles Darwin. En segundo lugar, se aplicaron redes neuronales y transferencia de conocimiento para desarrollar un clasificador de aves, aportando así a la causa de la conservación. Se entrenaron las redes neuronales con 58 especies diferentes de aves, ya sean nativas, endémicas o visitantes.

Una vez alcanzado un rendimiento satisfactorio en el modelo de identificación de especies de aves, se procedió al desarrollo de una aplicación web/móvil. Se utilizó una solución que permite alojar la aplicación en un servidor web para que los usuarios puedan acceder de forma remota, facilitando así el acceso desde dispositivos móviles y cualquier ubicación. El propósito es exportar el modelo entrenado en el servidor para facilitar la identificación de especies de aves mediante la captura de imágenes.

La aplicación presenta una interfaz inicial que clasifica a los usuarios según su perfil, ya sea niños, turistas o biólogos, adaptando la información proporcionada de acuerdo con esta categorización. Los usuarios tienen la opción de informar sobre avistamientos mediante llamadas desde sus celulares, comunicándose directamente con las oficinas o interactuando con los guías durante la exploración de las islas. Estas acciones son fundamentales para la identificación de aves nativas, endémicas e invasoras, contribuyendo significativamente a la conservación al prevenir la introducción de especies no autóctonas que podrían poner en

peligro la biodiversidad de las Islas Galápagos, ya sea por actividades humanas o factores naturales.

**Palabras Clave:**

Biodiversidad, Redes Neuronales, Transferencia de Conocimiento, Conservación, Aplicación Web/Móvil, Identificación de Aves.

# Abstract

The main objective of this thesis is to contribute to the conservation of the Galapagos Islands through education. To achieve this, an application has been developed that can run on both web and mobile devices, with the latter being the project's strong point. The thesis presents two key contributions: Firstly, we conducted an exhaustive search for Galapagos birds using three different sources to generate a training dataset, as none existed for our approach. These carefully obtained and labeled data are based on the guidance from Charles Darwin's Foundation main page. Secondly, neural networks and knowledge transfer were employed to develop a bird classifier, contributing to its conservation. Neural networks were trained using 58 different species of birds, whether native, endemic, or visitors.

Once a satisfactory performance was achieved in the bird species identification model, a web/mobile application was developed. A solution allowing hosting the application on a web server was used, enabling remote access for users from mobile devices at any location. The purpose is to export the trained model to the server, facilitating bird species identification through image capture.

The application features an initial interface that categorizes users as children, tourists, or biologists, adapting the provided information accordingly. Users can report sightings through calls generated from their mobile phones, directly contacting offices, or interacting with guides during island exploration. These actions are crucial for identifying native, endemic, and invasive bird species, contributing significantly to its conservation by preventing the introduction of non-native species that could endanger the biodiversity of the Galapagos Islands, whether due to human activities or natural factors.

**Key Words:**

Biodiversity, Neural Networks, Knowledge Transfer, Conservation, Web/Mobile Applica-

tion, Bird Identification.

# Contents

<b>Dedication</b>	<b>iii</b>
<b>Acknowledgment</b>	<b>iv</b>
<b>Resumen</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>Contents</b>	<b>ix</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem statement . . . . .	2
1.3 Objectives . . . . .	3
1.3.1 General Objective . . . . .	3
1.3.2 Specific Objectives . . . . .	3
1.3.3 Contributions . . . . .	3
<b>2 Theoretical Framework</b>	<b>5</b>
2.1 Fundamentals of Artificial Intelligence . . . . .	5
2.1.1 What is Artificial Intelligence? . . . . .	5
2.1.2 Historical Evolution of Artificial Intelligence . . . . .	5
2.2 Neural Networks . . . . .	7
2.2.1 Neural network architecture . . . . .	7

2.2.2	Convolutional Neural Networks . . . . .	7
2.2.3	Image Classification . . . . .	8
2.2.4	Transfer Learning . . . . .	9
2.3	Pre-trained architectures . . . . .	10
2.3.1	DenseNet-201 . . . . .	10
2.3.2	EfficientNet-B0 . . . . .	12
2.3.3	InceptionV3 . . . . .	13
2.3.4	MobileNetV2 . . . . .	15
2.3.5	ResNet50 . . . . .	17
2.3.6	VGG16 . . . . .	18
2.3.7	VGG19 . . . . .	20
2.4	Web/Mobile Applications . . . . .	22
2.4.1	Implementation on Mobile Devices . . . . .	22
2.4.2	TensorFlow.js in web applications . . . . .	23
2.5	Deep Learning Considerations . . . . .	24
2.5.1	Dropout . . . . .	24
2.5.2	Early Stopping . . . . .	25
2.5.3	Dataset . . . . .	26
2.5.4	Activation functions . . . . .	27
2.5.5	Loss functions . . . . .	28
2.6	App validation . . . . .	29
2.6.1	System Usability Scale . . . . .	29
2.6.2	Validation questionnaires . . . . .	30
2.6.3	System scale . . . . .	31
2.6.4	System Score . . . . .	32
<b>3</b>	<b>State of the Art</b>	<b>33</b>
3.1	Classification of Galapagos Birds . . . . .	33
3.1.1	Methods Used in Previous Research . . . . .	34
<b>4</b>	<b>Methodology</b>	<b>38</b>
4.1	Data Collection . . . . .	39

4.1.1	Galapagos Bird Image Data Sources . . . . .	41
4.1.2	Galapagos bird dataset . . . . .	42
4.2	Model selection . . . . .	42
4.2.1	Sequential model . . . . .	43
4.2.2	Machine Learning Performance Metrics . . . . .	45
4.3	Model Training . . . . .	46
4.3.1	Experiments to be performed . . . . .	47
<b>5</b>	<b>Results and Discussion of the models evaluated</b>	<b>48</b>
5.1	Experiment 1 . . . . .	48
5.2	Experiment 2 . . . . .	50
5.3	Experiment 3 . . . . .	52
5.4	Experiment 4 . . . . .	54
<b>6</b>	<b>Implementation of the Web/Mobile Application</b>	<b>56</b>
6.1	Web/Mobile application . . . . .	56
6.2	Conversion of the Model . . . . .	57
6.3	Web Server . . . . .	58
6.4	User Interface . . . . .	59
6.4.1	Child User . . . . .	61
6.4.2	Tourist User . . . . .	62
6.4.3	Biologist User . . . . .	62
6.5	System Usability Scale Questionnaire . . . . .	65
<b>7</b>	<b>Conclusions</b>	<b>68</b>
7.1	Conclusions . . . . .	68
7.1.1	Achievements and Contributions . . . . .	68
7.1.2	Limitations and Challenges . . . . .	69
7.2	Future Work . . . . .	69
7.2.1	Model Performance Improvements . . . . .	69
7.2.2	Expansion of the Mobile Application . . . . .	69



## **Bibliography**

**70**

# List of Tables

2.1	Milestones in the History of Artificial Intelligence . . . . .	6
2.2	Description of layers and outputs in the DenseNet-201 model [1]. . . . .	11
2.3	Description of layers and outputs in the EfficientNet-B0 model [2]. . . . .	13
2.4	Description of layers and outputs in the InceptionV3 model [3]. . . . .	15
2.5	Description of layers and outputs in the MobileNetV2 model [4]. . . . .	16
2.6	Description of layers and outputs in the ResNet50 model [5]. . . . .	18
2.7	Description of layers and outputs in the VGG16 model [6]. . . . .	20
2.8	Description of layers and outputs in the VGG19 model [7]. . . . .	21
3.1	Previously applied methods . . . . .	34
4.1	Data Sources Information . . . . .	41
4.2	Metadata extract [8] . . . . .	42
4.3	Neural Network Configuration . . . . .	43
5.1	Experiment 1 . . . . .	49
5.2	Experiment 2 unfreezing 15 layers . . . . .	50
5.3	Experiment 2 unfreezing 4 layers . . . . .	51
5.4	Experiment 3 . . . . .	53
5.5	Experiment 4 . . . . .	54
5.6	General Experiments . . . . .	55
6.1	Responses to the questionnaire . . . . .	67

# List of Figures

2.1	Biological Neuron and Artificial Neuron Model [9],[10]. . . . .	7
2.2	Structure of a convolutional neural network for image classification [11]. . .	9
2.3	Transfer Learning. . . . .	9
2.4	DenseNet-201 [1]. . . . .	11
2.5	EfficientNet-B0 [2]. . . . .	12
2.6	InceptionV3 [12]. . . . .	14
2.7	MobileNetV2 [13]. . . . .	16
2.8	ResNet50 [14]. . . . .	17
2.9	VGG16 [6]. . . . .	19
2.10	VGG19 [7]. . . . .	21
2.11	Web server . . . . .	22
2.12	TensorFlow model [15]. . . . .	23
2.13	Dropout . . . . .	24
2.14	Early Stopping [16]. . . . .	26
2.15	Data split [16]. . . . .	27
2.16	Neural network activation functions [17]. . . . .	27
2.17	Comparison of Questionnaires [18]. . . . .	30
2.18	usability system scale [19]. . . . .	31
2.19	SUS Score [18]. . . . .	32
3.1	Galapagos birds [20]. . . . .	33
4.1	General Model . . . . .	38
4.2	Data flow diagram . . . . .	39

---

4.3	Charles Darwin Foundation[21]	40
4.4	Training Diagram	43
4.5	Data augmentation	44
5.1	EfficientNetB0 Loss vs Epoch	49
5.2	VGG19 (4 layers) Loss vs Epoch	51
5.3	VGG19 (4 layers) Loss vs Epoch	53
5.4	VGG19 Leaky ReLU Loss vs Epoch	55
6.1	Application Model	57
6.2	URL process	58
6.3	Ngrok	59
6.4	Access to the website	60
6.5	User selection	61
6.6	Bird Classification	63
6.7	Information	64
6.8	Charles Darwin Foundation	65
6.9	System Usability Scale	66

# Chapter 1

## Introduction

### 1.1 Background

Currently, image classification has experienced exponential growth driven by technological advances, especially in the field of Artificial Intelligence (AI)[22]. These advances significantly impact various areas, fundamental in their application in the conservation of fragile ecosystems, such as the Galapagos Islands, which require constant maintenance and control.

This concern becomes essential as it contributes to environmental conservation, where biodiversity loss results from natural factors, such as pests, and human actions, such as species introductions. Recent projects have emerged to address this challenge in the Galapagos Islands [23], [24].

This research focuses on developing a methodology to classify diverse bird species, including introduced, endemic, native, and migratory species. A vital element of this approach is the creation of a diversified dataset to serve as a training base. Web/mobile applications will enable identification through the bird classifier and encourage participation of the tourism community by reporting sightings to authorities. This process is beneficial when users identify birds not included in the app, suggesting the possibility of potentially invasive birds. This approach not only protects avian wildlife but also indirectly contributes to the conservation of the Galapagos ecosystem. The reports generated by application users will provide authorities with valuable data for monitoring and managing avian biodiversity.

In the course of this thesis, several issues will be addressed, such as the selection of models to evaluate their performance, the choice of the training dataset, and the definition of methods for the identification and classification of birds once the training process is completed, to determine the most efficient model.

## 1.2 Problem statement

The problem lies in conserving the Galapagos Islands, a unique and distinctive environment characterized by rich biodiversity [25]. Over time, this ecosystem faces natural and anthropogenic threats, jeopardizing its stability. Natural disturbances, such as invasions of avian fauna, can arise due to migrations from other regions or the presence of vagrant birds, generating territorial conflicts [26].

Simultaneously, by residents and visitors, human intervention contributes to introducing non-native species, triggering unwanted pests that affect food availability and the habitat of endemic species [27]. Human presence also causes displacement and stress on native species, either by the presence of tourists or the infiltration of new invasive species.

Differentiating between native and foreign species is essential in detecting and managing threats in the Galapagos Islands. Accurate identification of introduced birds during monitoring could be necessary for effective action. Therefore, exploring alternative approaches that can improve effectiveness in addressing these issues is crucial.

This proposal aims to develop an accurate and accessible method for bird identification designed for a diverse audience, including biologists, tourists, and children. With the slogan **“IF YOU KNOW, YOU CONSERVE”**, this diversified approach seeks to encourage user collaboration by informing authorized personnel of sightings, enabling a rapid response to potential threats. This initiative aims to promote active community participation in the conservation of biodiversity, fauna, and the ecological balance of the Galapagos Islands.

## 1.3 Objectives

### 1.3.1 General Objective

Contribute to preserving biodiversity in the Galapagos Islands, focusing primarily on protecting its avian fauna through a web/mobile application.

### 1.3.2 Specific Objectives

In pursuit of our general objective, we have established the following specific objectives:

- Collect images of birds in the Galapagos islands from different datasets, focusing on the information on the Charles Darwin Foundation website.
- Perform the data processing to organize and label a dataset with the collected images in a structured way.
- Experiment with different convolutions neural network architectures.
- Compare different architectures to select the best classifier using machine learning metrics.
- Implement a web/mobile application to classify birds in the Galapagos.

### 1.3.3 Contributions

This thesis makes valuable contributions to the field of bird identification. Firstly, it highlights the creation of a diverse dataset that includes 58 bird species, totaling 7194 images [8]. This resource establishes a solid foundation for future research, as this dataset will be published for researchers to access and continue to contribute to the field. Secondly, it addresses identifying and selecting an optimal classifier designed explicitly for the image recognition process. Finally, the development of a web/mobile application is presented, facilitating bird classification in an accessible manner. This application becomes a practical tool for children, tourists, and biologists, streamlining the identification of various species through its intuitive and user-friendly interface.

The main contributions of this thesis are summarized as follows:

- Creation of a dataset covering 58 bird species from the Galapagos Islands.
- Selection of an optimal classifier for the bird identification process.
- Development of a web/mobile application that allows bird classification in an easy and personalized manner for the user.



# Chapter 2

## Theoretical Framework

### 2.1 Fundamentals of Artificial Intelligence

#### 2.1.1 What is Artificial Intelligence?

Artificial intelligence, according to [28], denotes the ability of machines to use algorithms, learn from data and apply that learning in decision making. Its purpose is to enable machines to perform tasks autonomously, optimizing processes that would usually require human intervention. In turn, this computing discipline seeks to develop technologies that replicate human intelligence, making it easier for machines to carry out processes such as learning, reasoning and self-correction.

This interdisciplinary field combines computer science with robust datasets to solve problems. It also encompasses subfields, such as machine learning and deep learning, associated with artificial intelligence but with their own distinctive features. These subfields focus on creating expert systems capable of making predictions or classifications based on input data [29].

#### 2.1.2 Historical Evolution of Artificial Intelligence

The history of artificial intelligence is intertwined with advances in electronic computing, and over time, several milestones have marked its evolution. Table 2.1 provides a chronological overview of some of these turning points, highlighting significant events that have contributed to the progress of artificial intelligence.

Year	Event
1950	Alan Turing presents “Computing Machinery and Intelligence”, proposing the Turing Test to assess whether a computer can match human intelligence [30].
1955	John McCarthy introduces the term “artificial intelligence” at the first AI conference at Dartmouth College. The same year, Allen Newell, JC Shaw, and Herbert Simon develop Logic Theorist, the first functional AI program [31].
1958	Frank Rosenblatt creates the Mark 1 Perceptron, the first computer based on a neural network that “learns” through trial and error. Marvin Minsky and Seymour Papert publish “Perceptrons” that year, becoming a landmark work in neural networks [32].
1990	Neural networks using backpropagation algorithms for training gain popularity in AI applications [33].
1997	IBM’s Deep Blue supercomputer defeats world chess champion Garry Kasparov, marking a milestone in AI for strategic games [34].
2015	Baidu’s Minwa supercomputer uses convolutional neural networks to identify and categorize images more accurately than the average human [35].
2016	DeepMind’s AlphaGo program, backed by a deep neural network, defeats world Go champion Lee Sodol, highlighting the potential of deep neural networks in complex problem-solving [36].
2023	Tesla launches its fleet of autonomous taxis, utilizing a neural network called Dojo to process camera and sensor images, providing a safe and efficient transportation service [37].

Table 2.1: Milestones in the History of Artificial Intelligence

Table 2.1 highlights eight important milestones, organized chronologically by year and event. A significant milestone was in 1950, when Alan Turing presented “Computing Machinery and Intelligence”, proposing the Turing Test to assess the ability of computers to match human intelligence [30]. In 1958, Frank Rosenblatt created the Mark 1 Perceptron, the first neural network-based computer that learned by trial and error [32]. In 2016, DeepMind’s DeepMind AlphaGo program, backed by a deep neural network, beat world Go champion Lee Sodol, highlighting the potential of deep neural networks in solving complex problems [36].

## 2.2 Neural Networks

### 2.2.1 Neural network architecture

Artificial Neural Network (ANN) architecture is a branch of Artificial Intelligence that is inspired by the neural processes of the human brain. ANNs, composed of interconnected nodes, recognize complex patterns and perform cognitive tasks without human supervision. An example is the Multi-Layer Perceptron (MLP), common in deep learning, with hidden layers and nonlinear functions [38]. Its operation includes forward propagation and learning by backpropagation, adjusting weights to minimize a loss function. Figure 2.1 shows a representation of a biological neuron and an artificial neuron model. In the biological neuron, elements such as dendrites, cell body, axon, among others, play specific roles in signal transmission and synaptic connection. The artificial model replicates these functions, representing inputs as  $X_0$  and using mathematical functions for activation. Similarities include signal reception (dendrites/inputs), integration (cell body/weighted sum), transmission (axon/output), and activation function (synapse/weighted sum and activation). In summary, the artificial model seeks to emulate the complexity of a biological neuron to understand and replicate neural processes in computational environments.

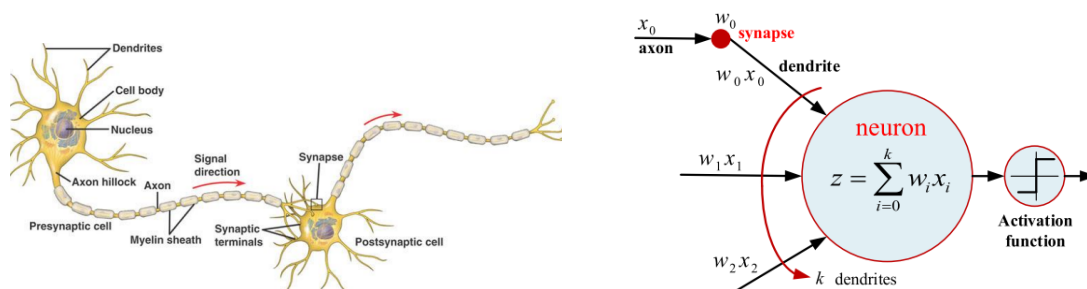


Figure 2.1: Biological Neuron and Artificial Neuron Model [9],[10].

### 2.2.2 Convolutional Neural Networks

Deep learning, a prominent branch of machine learning, relies on deep neural networks to process visual information and perform specific tasks, such as image classification. Facilitated by the availability of large datasets and software frameworks such as Tensorflow and PyTorch, deep learning has revolutionized areas such as Computer Vision, achieving

significant advances in object recognition and image segmentation [39].

Several techniques have been proposed within the field of image classification. Traditional techniques include Support Vector Machines (SVM), Decision Trees and Random Forests, Transfer Learning, Local Feature Descriptors such as SIFT (Scale-Invariant Feature Transform), and Ensemble Learning. Each technique presents specific approaches to address image classification, with particular applications and advantages [40, 41, 42, 43, 44].

### 2.2.3 Image Classification

In the field of image classification, Convolutional Neural Networks (CNNs) are fundamental in deep learning, standing out for their ability to identify visual patterns. The structure of a CNN, composed of interconnected layers, employs key operations such as convolution, activation functions, and pooling. Convolution involves applying filters to the input to highlight relevant patterns. Activation functions introduce nonlinearities into the model, allowing more complex representations to be learned. On the other hand, pooling reduces dimensions while preserving important features. This architecture specializes in classifying various types of data, including images, text, audios, generating accurate outputs for various classes [45]. Image classification, essential for assigning labels to images within predefined categories, plays a crucial role in the efficient analysis of digital images [46].

The CNN structure for image classification is visualized in Figure 2.2, presenting a schematic diagram highlighting the classification process. This includes key layers such as convolution, where filters are applied to the image to extract features, followed by pooling layers that reduce spatial dimensionality. The diagram illustrates additional convolution and pooling layers, followed by dense layers that connect all previous activations. The final output is represented by a dense layer reflecting the output classes, indicating in this case the possibility to classify into 1000 categories.

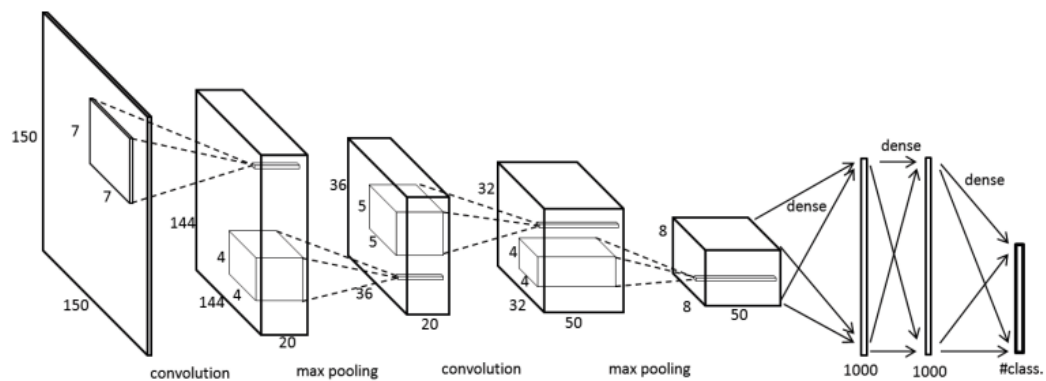


Figure 2.2: Structure of a convolutional neural network for image classification [11].

### 2.2.4 Transfer Learning

The Transfer Learning technique stands out as an efficient strategy in image classification. By utilizing pre-trained models on large datasets, this technique allows customizing models for specific tasks, significantly reducing training time and required resources [47]. The foundation of pre-trained models lies in the transfer of previously acquired knowledge, which can be applied to other related tasks. These techniques have proven effective in various applications, such as image classification [42], object recognition [48], and language translation [49], enhancing accuracy through adjustments in the weights of existing layers.

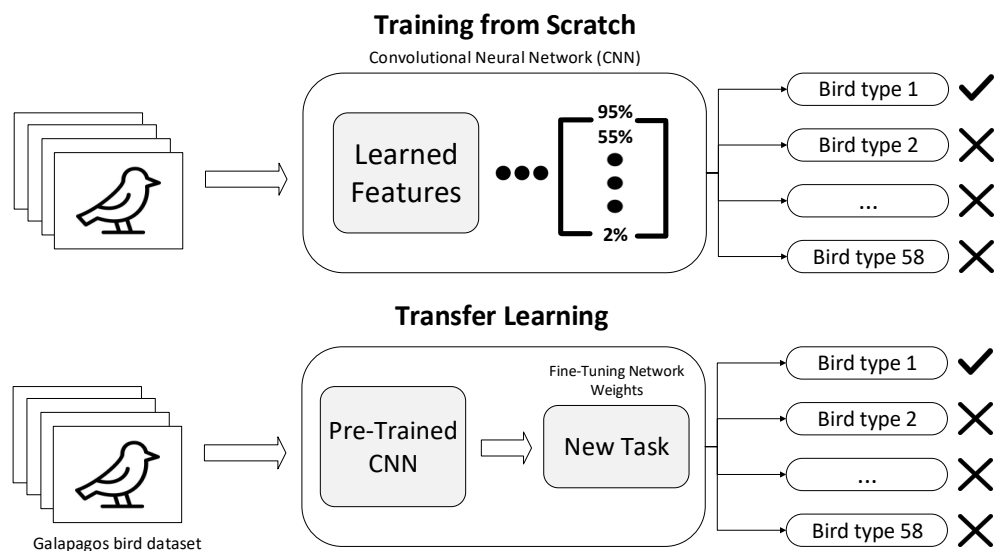


Figure 2.3: Transfer Learning.

Figure 2.3 illustrates two training approaches for machine learning models in the image classification task. The “Training from Zero” involves developing a Convolutional Neural Network (CNN) using a specific Galapagos bird dataset, learning unique features from scratch. On the other hand, “Transfer Learning” fits an already pre-trained CNN to the same dataset, leveraging prior knowledge for improved efficiency. The latter approach is generally preferred for speed and lower effort than full training from scratch.

## 2.3 Pre-trained architectures

The models chosen to perform the deep learning tests have been highlighted over time, as evidenced by works such as [50, 2, 51, 52, 14, 6, 53]. Some of their architectures will be examined in more detail below.

### 2.3.1 DenseNet-201

The use of the pre-trained DenseNet-201 model, recognized for its effectiveness in image classification, including the detection of objects [50]. This model excels on challenging datasets such as ImageNet, learning complex representations thanks to dense connections. The inclusion of transition layers between dense blocks improves efficiency and enables feature extraction at different scales, cementing DenseNet-201 as a versatile and powerful option in computer vision.

Figure 2.4 represents the architecture of the DenseNet-201 convolutional neural network used to classify images into various categories. The structure includes an input layer of dimensions  $224 \times 224 \times 3$  (“CP”) representing the image to be classified. Four dense layers (“D1”, “D2”, “D3”, “D4”) consist of dense blocks with interconnected convolutional layers, increasing the depth of the output. Three transformations (“T1”, “T2”, “T3”) between the dense layers reduce spatial and depth dimensions. The “GAP” layer implements “Global Average Pooling” to reduce the output to a one-dimensional vector, followed by an “FCL Softmax” layer that classifies the data into 1000 categories by Softmax activation.

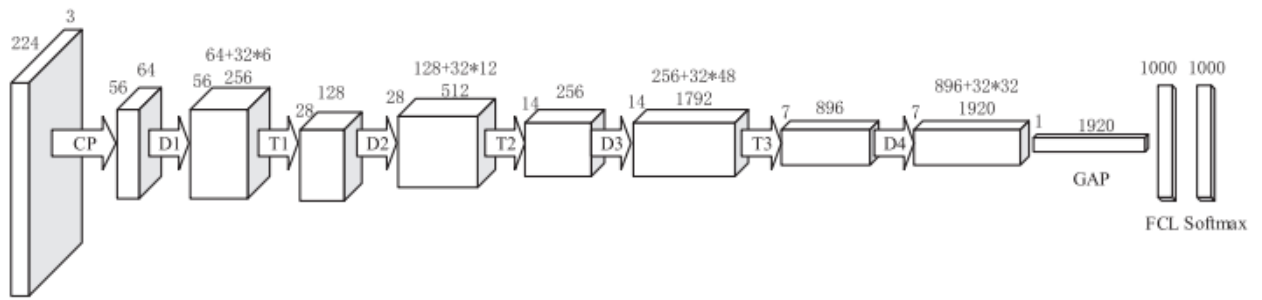


Figure 2.4: DenseNet-201 [1].

Table 2.2 describes in detail the architecture of the DenseNet-201 model, used for image classification. It starts with an input of dimensions  $224 \times 224 \times 3$  and goes through a  $7 \times 7$  convolutional layer with a stride of 2, reducing the dimensions to  $112 \times 112 \times 64$ . Then, a  $3 \times 3$  pooling with a stride of 2 is applied, resulting in an output of  $56 \times 56 \times 64$ . The architecture includes three dense blocks with different convolution and connection configurations. Between these blocks, Transition Layers are inserted to reduce the spatial and depth dimensions. After the last Dense Block, an average pooling of  $7 \times 7$  is applied, obtaining an output of  $1 \times 1 \times 1024$ . Finally, a fully connected layer (FCL) with softmax activation is found, classifying the data into 1000 categories. The overall structure of the model reflects a progression of dimensional reduction followed by dense blocks, culminating in an output layer that classifies the images into multiple categories.

Table 2.2: Description of layers and outputs in the DenseNet-201 model [1].

Layers	DenseNet-201	Output
Input	-	$224 \times 224 \times 3$
Conv	$7 \times 7 / 2$ conv,	$112 \times 112 \times 64$
Pooling	$3 \times 3 / 2$ pool	$56 \times 56 \times 64$
Dense Block 1	$[1 \times 1$ conv] X 6	$56 \times 56 \times 256$
Transition Layer 1	$1 \times 1$ conv, $2/x/2$ average pool	$28 \times 28 \times 128$
Dense Block 2	$[3 \times 3$ conv] X 12	$28 \times 28 \times 512$
Transition Layer 2	$1 \times 1$ conv, $2/x/2$ average pool	$14 \times 14 \times 256$
Dense Block 3	$[3 \times 3$ conv] X 48	$14 \times 14 \times 1024$
Transition Layer 3	$1 \times 1$ conv, $2/x/2$ average pool	$7 \times 7 \times 512$
Dense Block 4	$[3 \times 3$ conv] X 32	$7 \times 7 \times 1024$
Pooling	$7/7$ average pool	$1 \times 1 \times 1024$
FCL Softmax	1000-way fc, softmax	1000

### 2.3.2 EfficienNet-B0

The EfficientNet-B0 architecture has been used in the proposed dataset due to its recognized capability for outstanding feature extraction in images [2]. This architecture stands out for its efficiency, achieving an optimal balance between performance and efficient use of resources. Its uniform scaling approach improves performance with fewer parameters and floating-point operations, making it versatile for learning transfer and suitable for deployments on resource-constrained devices. In summary, EfficientNet-B0 is a solid choice for feature extraction tasks in images.

Figure 2.5 depicting the architecture of EfficientNet-B0, a convolutional neural network, shows a  $224 \times 224 \times 3$  RGB image input. It starts with a convolution layer of 32  $3 \times 3$  filters, followed by MBConv6 blocks that perform inverse transformations, each with expansion, deepening, and compression. The repetition of these blocks, some with average clustering layers, helps to extract features from different levels. Then, a convolution layer of 320  $1 \times 1$  filters with Swish activation function produces a  $7 \times 7 \times 1280$  output. A global average clustering operation reduces the output to  $1 \times 1 \times 1280$ , and finally, a fully connected layer with Softmax activation generates the probabilities of membership in each of the 1000 possible categories in the  $1 \times 1000$  output layer.

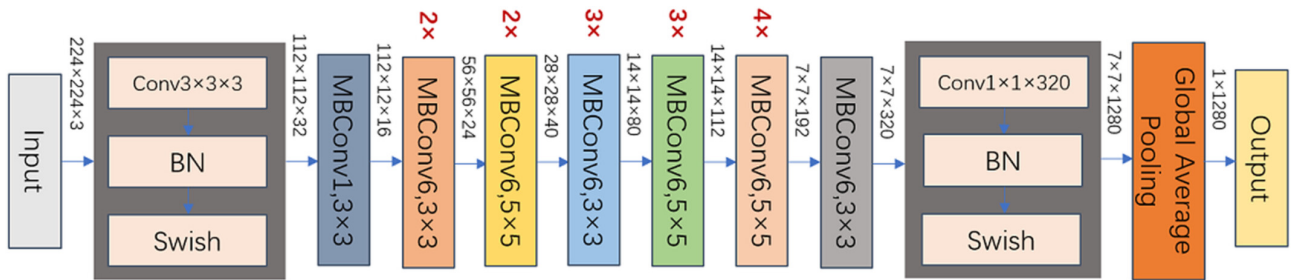


Figure 2.5: EfficientNet-B0 [2].

Table 2.3 EfficientNet-B0 shows the layers and outputs of the model, which starts with an input layer of a  $224 \times 224$  pixel image and three color channels. The second row uses a convolution layer with 32 filters of  $7 \times 7$  pixels. The third to tenth row consists of dense layers with MBConv blocks, where the MBConv1 block, for example, expands the input, deepens by a convolutional layer with Swish activation, and compresses the output. These MBConv blocks help improve model efficiency and performance by extracting features from different levels of the image. The eleventh row presents a global clustering layer that reduces



the spatial size of the output to 1 x 1 pixels, while the twelfth row is a fully connected layer with Softmax activation, producing the final image classification among the 1000 possible categories. MBCnv refers to MobileNetV2 Convolution, a special convolutional layer with expansion, deepening and compression, designed to improve model efficiency and performance.

Table 2.3: Description of layers and outputs in the EfficientNet-B0 model [2].

Layer	EfficientNet-B0	Output
Input	-	224x224x3
Convolution	7x7 /2 conv,	112x112x32
Dense 1	MBCnv1, k=3	112x112x16
Dense 2	MBCnv6, k=3	56x56x24
Dense 3	MBCnv6, k=5	28x28x40
Dense 4	MBCnv6, k=3	14x14x80
Dense 5	MBCnv6, k=5	14x14x112
Dense 6	MBCnv6, k=5	7x7x192
Dense 7	MBCnv6, k=3	7x7x320
Convolution	1x1 conv,	7x7x1280
Pooling	Global Average Pooling	1x1x1280
Fully Connected	1000-way fc, softmax	1000

### 2.3.3 InceptionV3

Similarly, we proceeded to use another widely used architecture, InceptionV3, which specializes in extracting high-resolution features in image detection and object classification [51]. InceptionV3 is known for its computational efficiency and its ability to capture patterns at different scales through the use of Inception modules. In addition, this architecture has proven to be effective in transfer learning tasks, allowing its reuse in contexts where data sets are smaller. Its application ranges from the recognition of objects in images to the classification of medical images, standing out for its good accuracy and versatility in various applications related to image processing.

Figure 2.6 shows a diagram of a CNN processing a 299x299x3 input image to generate a feature vector. The process flow is represented by connected circles, indicating various layers. Labels such as “Input: 299x299x3” and “Output: 8x8x2048” indicate the dimensions of the tensor at different points. The colors inside the circles represent operations such as Convolution (orange), AvgPool (blue), MaxPool (green) and Concat (red). At

the end, a block shows final operations such as Dropout (purple), Fully connected (pink) and Softmax (dark blue). A red arrow connects the tensor transformation from “Input:  $299 \times 299 \times 3$ ” to “Output:  $8 \times 8 \times 2048$ ”. In the upper right, “Final part:  $8 \times 8 \times 2048 - 1001$ ” is indicated, representing the transformation to the final vector.

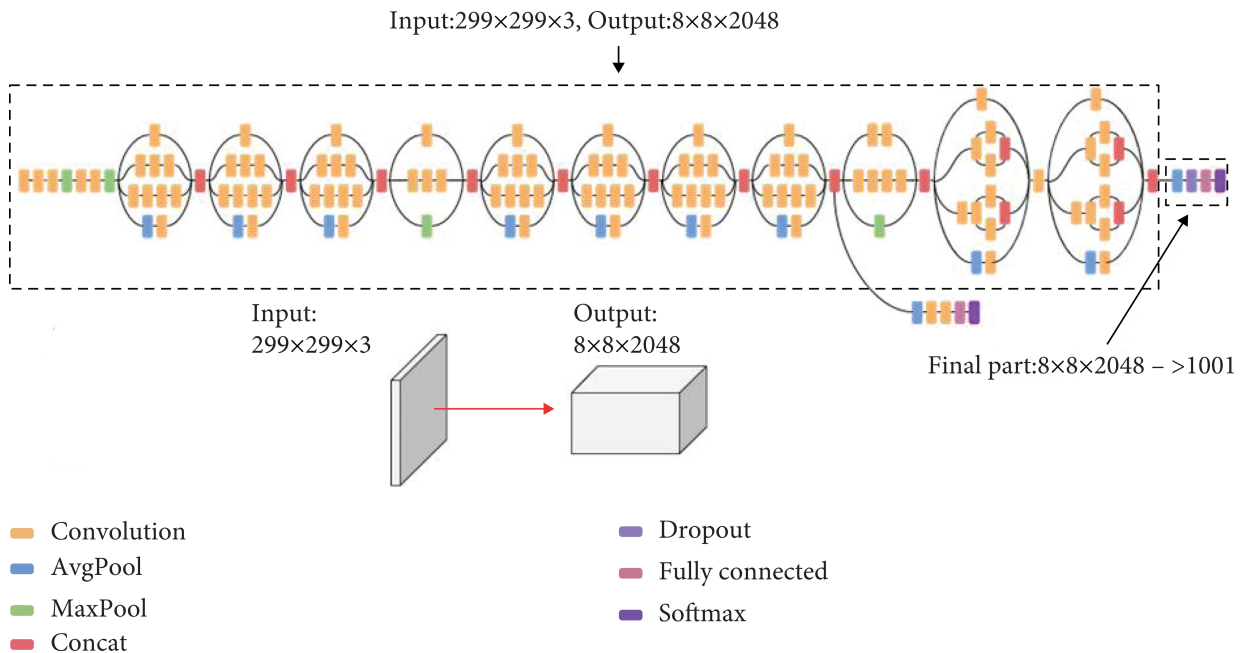


Figure 2.6: InceptionV3 [12].

Table 2.4 presents three columns detailing the layers of the InceptionV3 model and their respective process. The first row describes the input layer that receives a  $299 \times 299$  pixel image with three color channels. The next rows describe convolution layers that apply filters of different sizes and steps to extract low, medium and high level features from the image. Then, four Inception modules are presented that combine convolutional and clustering layers to increase the diversity and complexity of the extracted features. The eleventh row shows a global clustering layer that reduces the output to  $1 \times 1$  pixels. Finally, the twelfth row is a fully connected layer with Softmax activation that produces the classification of the image into one of the 1000 possible categories.

Table 2.4: Description of layers and outputs in the InceptionV3 model [3].

Layer	InceptionV3	Output
Input	-	299x299x3
Convolution	3x3 /2 conv,	149x149x32
Convolution	3x3 /1 conv,	147x147x32
Convolution	3x3 /1 conv,	147x147x64
Convolution	3x3 /2 conv,	73x73x80
Convolution	3x3 /1 conv,	71x71x192
Inception 1	Mixed 5b	35x35x288
Inception 2	Mixed 6a	17x17x768
Inception 3	Mixed 7a	8x8x1280
Inception 4	Mixed 8	8x8x2048
Pooling	8/8 average pool	1x1x2048
Fully Connected	1000-way fc, softmax	1000

### 2.3.4 MobileNetV2

In this context, the pre-trained MobileNetV2 model, recognized for its resource efficiency and learning transfer capability, was implemented. MobileNetV2 uses modified inverted residual blocks and dilated convolution to improve nonlinear representation without significantly compromising computational resources and memory. Its scalable architecture, coupled with depth- and width-separable convolutions, positions it as an optimal choice for computer vision tasks in hardware-constrained environments, facilitating deep training and enabling pre-trained knowledge transfer on large datasets [52].

Figure 2.7 illustrates the architecture of a convolutional neural network, specifically MobileNetV2, detailing the process of transforming an input image through various blocks and layers to obtain a feature map. Beginning with a 224x224x3-pixel input image of stacked apples, the “Conv” convolutional layer reduces dimensions to 112x112x32 through a convolution operation and ReLU6 activation. The “BnR Block 1”, a residual block, further decreases dimensions to 112x112x16 while maintaining spatial size. Subsequent “BnR Block 2” to “BnR Block 17” residual blocks follow a similar pattern, gradually reducing spatial dimensions and increasing depth. The final “Conv” layer processes the output of the last residual block to 7x7x1280. A colorful abstract map of 7x7x1280 represents the model’s final features, enabling classification into categories such as fruits, animals, or objects.

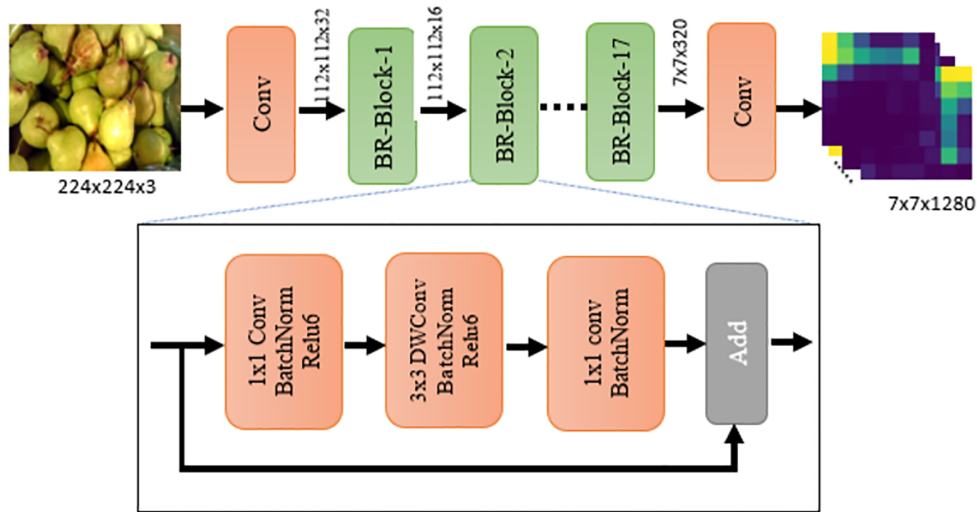


Figure 2.7: MobileNetV2 [13].

Table 2.5 provides a detailed description of the layers and components of the MobileNetV2 model. It starts with the input layer, which receives 224 x 224 pixel images. Key layers are highlighted, such as the first convolution layer that applies 32 filters, and the inverted residual blocks that improve the efficiency of the model. These blocks include expansions, deepening and compressions to extract features at different levels. Throughout the layers, operations such as average clustering and convolution are performed to reduce the spatial size and extract high level features. The model culminates in a fully connected layer with Softmax activation to classify the image into one of 1000 possible categories.

Table 2.5: Description of layers and outputs in the MobileNetV2 model [4].

Layer	MobileNetV2	Output
Input	-	224x224x3
Convolution	3x3 /2 conv,	112x112x32
Residual Block 1	MBCConv1, k=3	112x112x16
Residual Block 2	MBCConv6, k=3	56x56x24
Residual Block 3	MBCConv6, k=5	28x28x32
Residual Block 4	MBCConv6, k=3	14x14x64
Residual Block 5	MBCConv6, k=5	14x14x96
Residual Block 6	MBCConv6, k=5	7x7x160
Residual Block 7	MBCConv6, k=3	7x7x320
Convolution	1x1 conv,	7x7x1280
Pooling	Global Average Pooling	1x1x1280
Fully Connected	1000-way fc, softmax	1000

### 2.3.5 ResNet50

In this case, it is proposed to use ResNet50, a widely recognized pre-trained model, which stands out for its 50-layer deep architecture and the use of residual connections. This neural network has proven to be highly effective in image classification tasks, thanks to its ability to handle deep networks and its learning efficiency. Its use is based on transfer learning, where it is pre-trained on large datasets, such as ImageNet, and then tuned for specific tasks with smaller datasets [14].

Figure 2.8 depicts the architecture of a convolutional neural network divided into two sections. The left part showcases a pre-trained model named ResNet-50, which has been previously trained on diverse images. This section consists of layers with specific functions such as convolutions and residual blocks, each identified by its name and color associated with the output size. The right part, labeled "Trained on new dataset," indicates the extension of the pre-trained model to address a new image classification problem. This section includes an additional residual block, a fully connected layer, and a Softmax function. The output layer has neurons corresponding to the classification categories, and the model's prediction is based on the category with the highest probability calculated by the Softmax function.

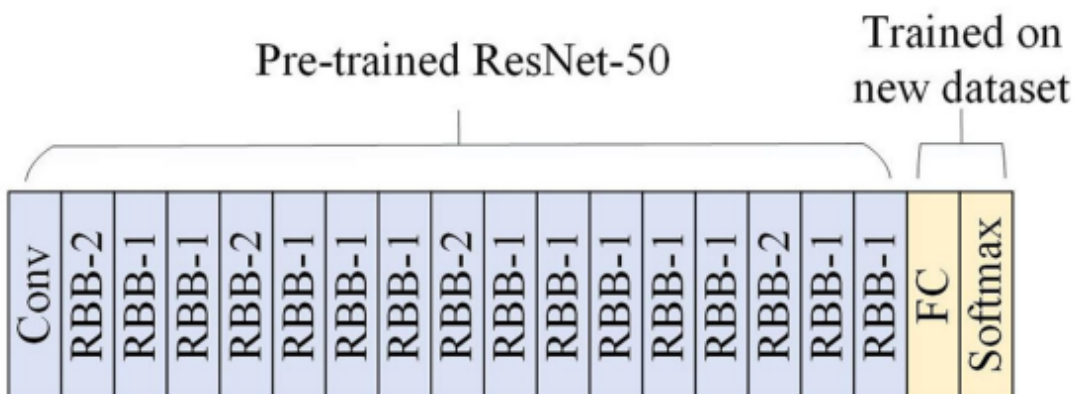


Figure 2.8: ResNet50 [14].

Table 2.6 provides a detailed view of the ResNet50 architecture, organized in three columns: Layer, ResNet50 and Output. Each row represents a layer or component of the model. Starting with the input layer for 224x224 pixel RGB images, the architecture includes layers for convolution, maximum clustering, and residual blocks. These blocks, grouped into sets of different sizes, apply convolutional layers, ReLU activation, and batch normalization, preserving information through residual connections. Residual blocks help improve model performance and efficiency without changing the spatial size of the input. The architecture culminates in a convolution layer, followed by a global clustering layer and a fully connected layer with Softmax activation for classification into 1000 possible categories. The detailed structure provides a complete understanding of how the network processes and extracts features throughout its layers.

Table 2.6: Description of layers and outputs in the ResNet50 model [5].

Layer	ResNet50	Output
Input	-	224x224x3
Convolution	7x7 /2 conv,	112x112x64
Pooling	3x3 /2 max pool	56x56x64
Residual Block 1	3x (1x1, 3x3, 1x1) conv	56x56x256
Residual Block 2	4x (1x1, 3x3, 1x1) conv	28x28x512
Residual Block 3	6x (1x1, 3x3, 1x1) conv	14x14x1024
Residual Block 4	3x (1x1, 3x3, 1x1) conv	7x7x2048
Pooling	Global Average Pooling	1x1x2048
Fully Connected	1000-way fc, softmax	1000

### 2.3.6 VGG16

This comparison has included the VGG16 network, known for its efficiency in using blocks composed of a progressive number of convolutional layers with constant size 3x3 filters. This architecture, proposed by the Visual Geometry Group (VGG) in 2014, stands out for its simplicity and deep learning capability, having demonstrated good results on a variety of tasks, as documented in previous studies [6].

The architecture of the VGG16 convolutional neural network is presented in Figure 2.9. The structure follows a flow from data input to output, passing through several convolutional (Conv), pooling (Pool) and fully connected (FC) layers. Data is fed into the convolutional layers where it is processed and then pooled to reduce dimensions. This process is repeated several times before reaching the fully connected layers and finally the output which determines the final classification.

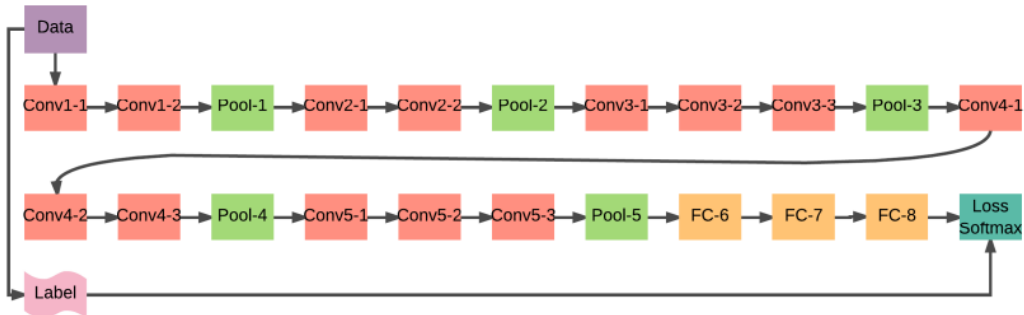


Figure 2.9: VGG16 [6].

Table 2.7 comprehensively details the architecture of the VGG16 convolutional neural network, providing information about each layer or component and the associated output. The sequence starts with the input layer, intended to receive 224x224 pixel RGB images, which are the object of classification among the 1000 possible categories. The second row presents a convolution layer that applies 64 filters of 3x3 pixels with a step of 1 pixel, generating an output of 224x224 pixels. These filters, with ReLU activation, focus on extracting low-level features such as edges and colors. The third row includes a maximum binning layer that reduces the output to 112x112 pixels, contributing to parameter reduction and prevention of overfitting. This pattern of convolution layers followed by grouping is repeated in subsequent rows, gradually increasing the complexity and level of abstraction of the extracted features. The structure culminates in fully connected layers, where the resulting vectors are transformed and Softmax activation is applied to the final layer, generating a probabilistic classification of the image into the 1000 categories previously mentioned. Each component conforms to a coherent design of 3x3 filters and ReLU activations, and maximum clustering layers are incorporated to regulate model complexity and prevent overfitting. This modular and progressive design of VGG16 has proven effective in image classification tasks across a variety of applications.

Table 2.7: Description of layers and outputs in the VGG16 model [6].

Layer	VGG16	Output
Input	-	224x224x3
Convolution	2x (3x3 /1 conv, ReLU)	224x224x64
Pooling	2x2 /2 max pool	112x112x64
Convolution	2x (3x3 /1 conv, ReLU)	112x112x128
Pooling	2x2 /2 max pool	56x56x128
Convolution	3x (3x3 /1 conv, ReLU)	56x56x256
Pooling	2x2 /2 max pool	28x28x256
Convolution	3x (3x3 /1 conv, ReLU)	28x28x512
Pooling	2x2 /2 max pool	14x14x512
Convolution	3x (3x3 /1 conv, ReLU)	14x14x512
Pooling	2x2 /2 max pool	7x7x512
Fully Connected	4096-way fc, ReLU	4096
Fully Connected	4096-way fc, ReLU	4096
Fully Connected	1000-way fc, softmax	1000

### 2.3.7 VGG19

The VGG19 architecture stands out as a CNN model of remarkable quality due to its simplicity and depth. Developed by the Visual Geometry Group at the University of Oxford, its structure is distinguished by its uniform design, composed of blocks of convolutional and pooling layers, followed by fully connected layers. With 19 layers, VGG19 uses small filters (3x3) and maximum pooling, making it easy to implement and understand. Its generalization and transfer learning capability in image classification tasks is outstanding, although it has disadvantages such as high computational cost due to the number of parameters. Although more modern models have emerged, the simplicity and effectiveness of VGG19 keep it as a significant benchmark in the field of computer vision [53].

Figure 2.10 shows the architecture of a VGG19 model, which is a convolutional neural network used for image classification. On the left, there is an image of a cougar that is fed into the model. The model consists of several layers, including convolutional (CONV) and max-pooling (Max-pooling) layers, to extract features from the image. These layers are represented by blue and orange rectangles. The blue section represents feature extraction and the green section, on the right, represents classification where the most likely label or category for that image is identified.



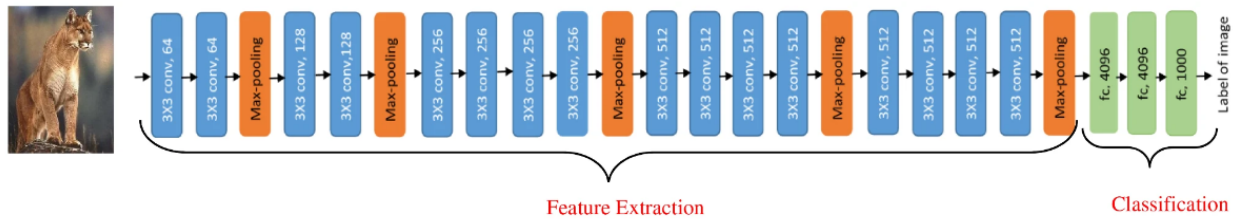


Figure 2.10: VGG19 [7].

Table 2.8 represents a VGG19 convolutional neural network model consisting of 19 layers, including convolution layers, ReLU activation, and fully connected layers. The network takes a 224x224 pixel RGB image as input and goes through several convolution and maximum clustering layers to extract low-, medium-, and high-level features from the image. Finally, the network produces a classification of the image into one of 1000 possible categories through a fully connected layer with Softmax activation.

In summary, VGG19 is a deep model that uses convolution and max-clustering layers to extract features from an image and produce an accurate classification into one of 1000 possible categories.

Table 2.8: Description of layers and outputs in the VGG19 model [7].

Layer	VGG19	Output
Input	-	224x224x3
Convolution	2x (3x3 /1 conv, ReLU)	224x224x64
Pooling	2x2 /2 max pool	112x112x64
Convolution	2x (3x3 /1 conv, ReLU)	112x112x128
Pooling	2x2 /2 max pool	56x56x128
Convolution	4x (3x3 /1 conv, ReLU)	56x56x256
Pooling	2x2 /2 max pool	28x28x256
Convolution	4x (3x3 /1 conv, ReLU)	28x28x512
Pooling	2x2 /2 max pool	14x14x512
Convolution	4x (3x3 /1 conv, ReLU)	14x14x512
Pooling	2x2 /2 max pool	7x7x512
Fully Connected	4096-way fc, ReLU	4096
Fully Connected	4096-way fc, ReLU	4096
Fully Connected	1000-way fc, softmax	1000

## 2.4 Web/Mobile Applications

### 2.4.1 Implementation on Mobile Devices

The implementation of image classification on mobile devices has undergone significant advances thanks to technological evolution. This progress is materialized through the integration of mobile applications that incorporate artificial intelligence models specifically tuned for the classification task. These models are transferred to mobile devices, enabling the execution of classifications in real time [54]. This capability capitalizes on the computational power of mobile devices, leveraging it to efficiently perform the tasks for which they were trained.

In this context of mobile technology implementation, Figure 2.11 illustrates the interaction of a user with a web application through a browser. In this process, the user sends a request to the web server, which uses JavaScript and PHP to process it and generate a response. The resulting response contains elements such as HTML, CSS, images and JavaScript, which are displayed in the user's browser. The flowchart uses symbols to represent actions, decisions, connection points and information flow in this specific context.

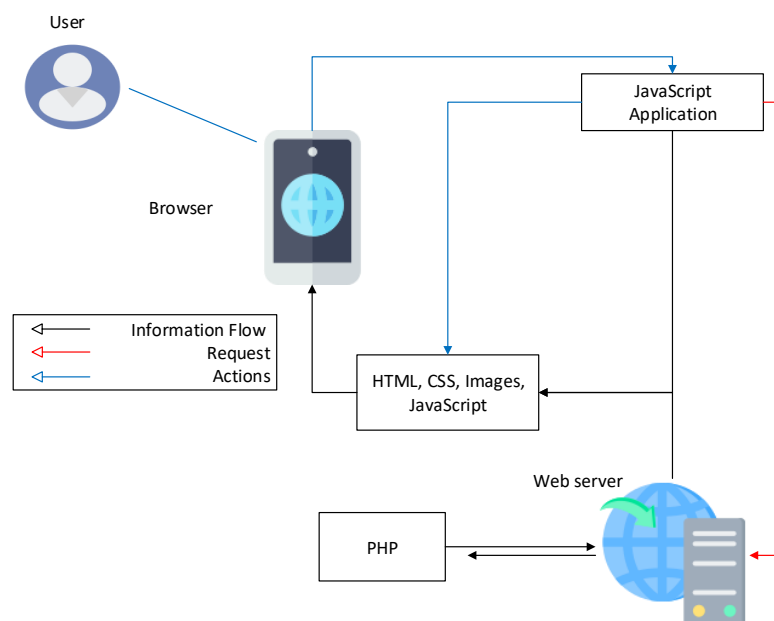


Figure 2.11: Web server

## 2.4.2 TensorFlow.js in web applications

TensorFlow.js, an open-source library developed by Google, plays a crucial role in enabling machine learning tasks directly within web browsers and JavaScript-based environments [55]. It facilitates the construction, training, and efficient execution of models on the client side, providing developers with the capability to seamlessly integrate machine learning functionalities into web applications without relying on external services. Its flexible API and interactive demonstrations make model creation intuitive, requiring no specific prerequisites other than JavaScript in the development environment. In summary, TensorFlow.js is a powerful tool for incorporating machine learning into web applications through JavaScript.

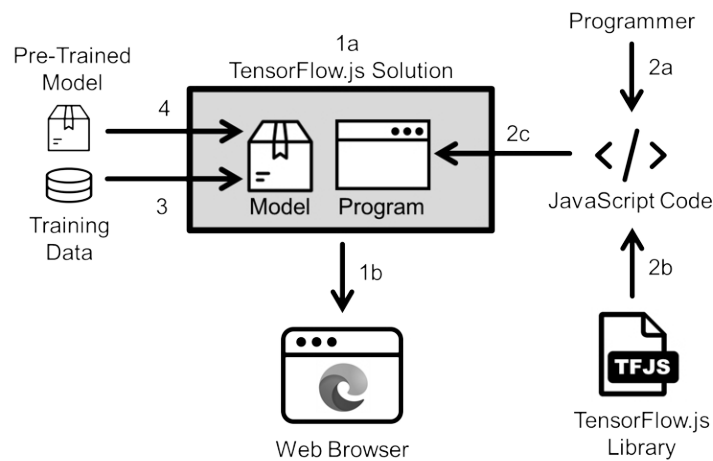


Figure 2.12: TensorFlow model [15].

Figure 2.12 describes how a pre-trained machine learning model is combined with specific training data to implement a solution in TensorFlow.js, which is then run in a web browser. A programmer writes JavaScript code that interacts with the TensorFlow.js library, allowing the solution, which includes the model and the program, to run within the browser. This facilitates the creation of interactive web applications that harness the power of machine learning directly from the user's browser.

## 2.5 Deep Learning Considerations

This section will discuss some important methods to perform a correct training of a neural network model.

### 2.5.1 Dropout

Dropout consists of randomly turning off some nodes in the hidden layer during each iteration of training, which makes the network less dependent on certain connections and more adaptive to the data. This helps prevent overfitting, which is when the network fits the training data too closely and loses generalization capability [56].

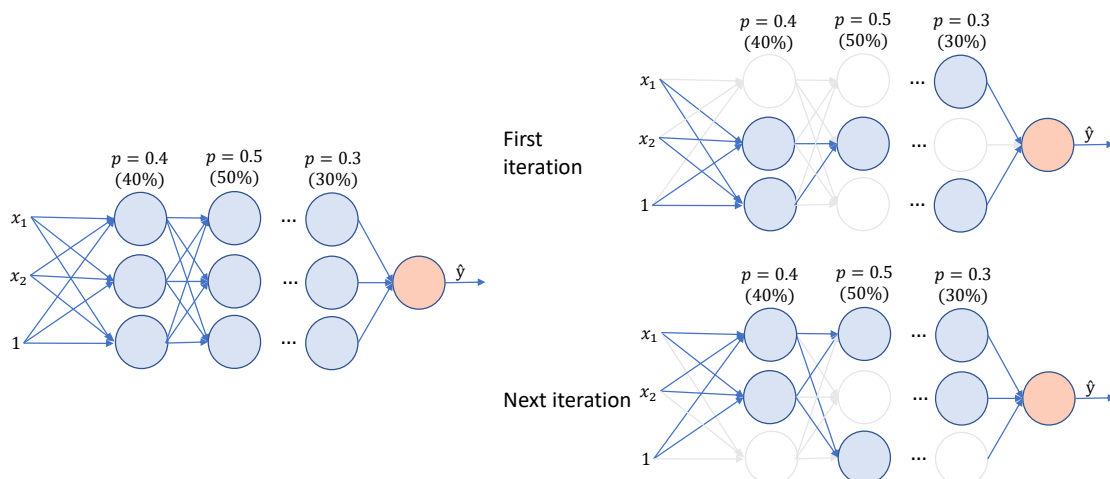


Figure 2.13: Dropout

Figure 2.13 illustrates an example of how dropout works in a neural network with three layers: an input layer, a hidden layer, and an output layer. The variables being used are:

- $x_1, x_2$ : These are inputs to the nodes in the hidden layer. They can be data features such as color, shape, size, etc.
- $p = 0.4, p = 0.5, p = 0.3$ : These are the retention probabilities for nodes in the hidden layer during dropout. They indicate the likelihood of each node remaining active or

being "turned off" during a specific training iteration. For instance,  $p = 0.4$  means there is a 40% chance for the node to stay active and a 60% chance to be turned off.

- $y$ : This is the output of the network, which can be a prediction, classification, regression, etc.

Figure 2.13, two different iterations with applied dropout can be observed. In the first iteration, some nodes in the hidden layer are shaded, indicating they have been turned off and do not contribute to the output calculation. In the second iteration, different nodes are turned off, creating a different network. This way, dropout generates several distinct networks that combine to form a more robust and diverse network.

## 2.5.2 Early Stopping

Early stopping is a regularization technique to avoid overfitting when training a machine learning model. Overfitting occurs when the model fits too closely to the training data and loses the ability to generalize to new data. Early stopping consists of stopping the training before the model over-fits the data, using a validation set to evaluate the model's performance [57].

Figure 2.14 illustrates the evolution of the loss function over epochs. Here, the training loss (depicted by the blue line) and the validation loss (depicted by the orange line) are observed for each training cycle (epoch). During training, the loss value may decrease as the number of epochs increases, as seen in the blue line. However, the actual performance of the model on unseen data, indicated by the orange line, may not show a significant change. As depicted in the graph, the validation loss converges after 200 epochs, suggesting that the model no longer improves its predictive ability beyond that point. Therefore, overfitting is occurring for epochs greater than 200.

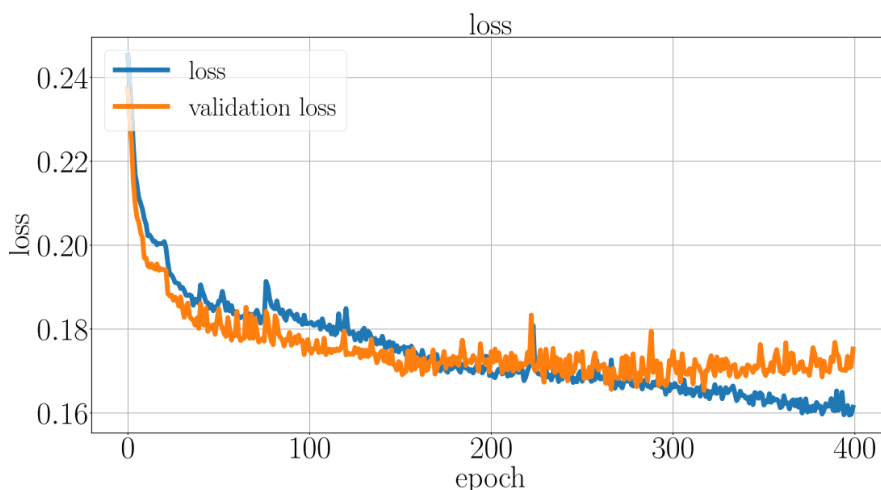


Figure 2.14: Early Stopping [16].

### 2.5.3 Dataset

Data splitting is the process of dividing a data set into smaller subsets for different purposes in machine learning. The proportion of splitting depends largely on the nature and quality of the available data. For example, the data set can be split into training, validation and test sets, as shown in Figure 2.15.

- Training is the phase where the model learns from the data and adjusts its parameters. The largest subset of data is used, around 60% of the total.
- Validation is the phase where the model's performance is evaluated, and the best among various candidates is selected. A smaller subset of data is used, typically around 20% of the total.
- Testing is the phase where the final performance of the model on new and unseen data is measured. The smallest subset of data is used, typically around 20% of the total. It is not used to adjust or select the model but to estimate its generalization capacity.

The figure visually represents the typical proportions in which datasets are divided: light green for “Training”, yellow for “Validation”, and red for “Testing”. These proportions may vary based on the dataset size, complexity, and the model used.

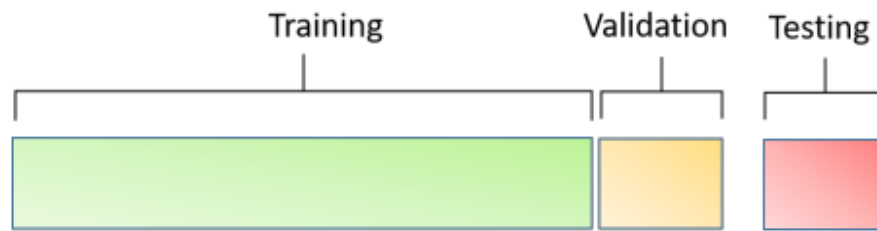


Figure 2.15: Data split [16].

## 2.5.4 Activation functions

At the output of a neuron, a filter, limiter function, or threshold can be applied to adjust the resulting value or set a limit that must be exceeded to pass to another neuron. This component is called an activation function and acts as a means of transmitting information generated by the linear combination of weights and inputs. In other words, activation functions are crucial for routing information through the output connections of the neural network [58].

Activation functions can allow the direct transmission of information without modification, known as the identity function, or restrict the transmission of information. Since the main goal is to train the neural network to address increasingly complex problems, the use of nonlinear activation functions is suggested. Figure 2.16, some activation functions can be observed.

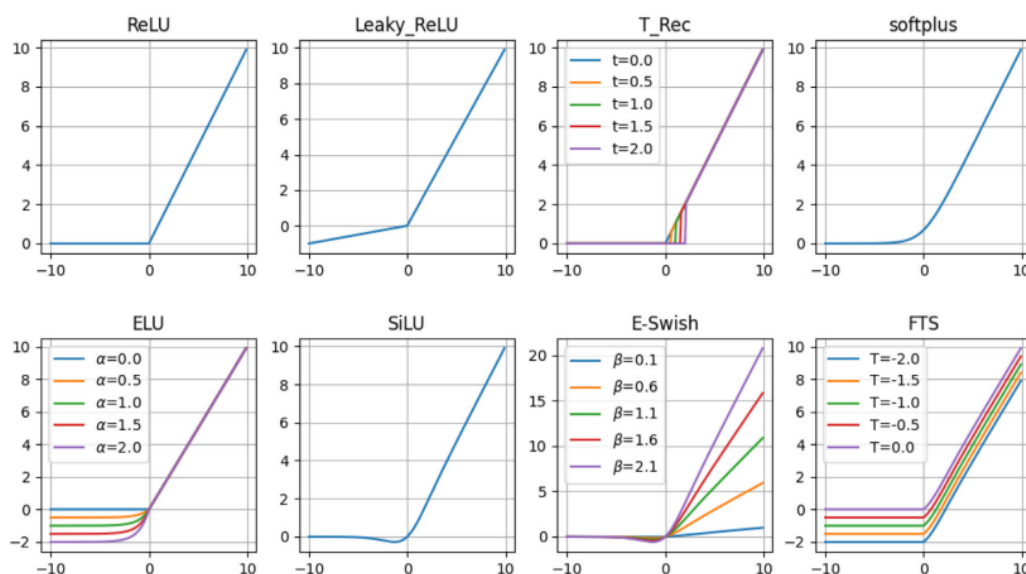


Figure 2.16: Neural network activation functions [17].

Two commonly used activation functions, ReLU and Leaky ReLU, play a crucial role in neural networks. ReLU transmits positive inputs directly and outputs zero for negative inputs, potentially causing the problem known as “dead neurons”, when all inputs are negative, leading to stagnant learning. Leaky ReLU addresses this issue by allowing a small slope for negative inputs. This flexibility makes Leaky ReLU valuable for maintaining information flow, especially in scenarios where negative inputs might be prevalent, enhancing its effectiveness in building and training neural networks [59].

### 2.5.5 Loss functions

In machine learning, the loss function, also called the cost or objective function, plays a crucial role in evaluating the effectiveness of the model and guiding its optimization.

The loss function measures the discrepancy between model predictions and actual values, providing an essential metric for adjusting model parameters. During neural network training, this function drives optimization by adjusting weights and biases to minimize divergence and improve accuracy [60].

To address the challenges of multiclass classification, categorical cross-entropy loss is used. Next, the weighted cross-entropy function (Equation 2.1), along with its components, is described.

$$J_{wce} = -\frac{1}{M} \sum_{k=1}^M \sum_{m=1}^K w_k \cdot y_m^k \cdot \log(h_{\theta}(x_m, k)) \quad (2.1)$$

Where:

- $J_{wce}$  is the weighted categorical cross-entropy loss.
- $M$  is the total number of samples.
- $K$  is the number of classes.
- $w_k$  is the weight associated with class  $k$ .
- $h_{\theta}(x_m, k)$  is the predicted probability that sample  $x_m$  belongs to class  $k$ , according to the model with parameters  $\theta$ .



therefore the loss function, when used in classification problems, strengthens the model fitting process by effectively quantifying the differences between the model predictions and the true labels, thus facilitating the optimization of system performance.

## 2.6 App validation

### 2.6.1 System Usability Scale

Originally devised by John Brooke at Digital Equipment Corporation in the United Kingdom in 1986. It stands as a crucial tool in the evaluation of systems and technologies. This questionnaire, designed to measure usability, emphasizes three essential aspects. Firstly, it assesses effectiveness by measuring users' ability to achieve their objectives. The second aspect focuses on efficiency, quantifying the effort and resources users expend to reach their goals. Lastly, the third aspect highlights user satisfaction, providing valuable insights into the overall quality of the user experience [19].

The SUS comprises 10 questions presented to the user after they have already tested the system, generating feedback on the system used.

- I think I would use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think I would need the help of a person with technical knowledge to use this system.
- I found the system functions well integrated.
- I think there was too much inconsistency in this system.
- I imagine most people would learn to use this system very quickly.
- I found the system cumbersome to use.
- I felt confident using the system.
- I needed to learn many things before I could start using the system.

## 2.6.2 Validation questionnaires

Figure 2.17 shows the percentage of “correct” conclusions as a function of sample size for different methods, including SUS, QUS, CSUQ and Words. As the sample size increases, the percentage of “correct” conclusions also increases for all methods.

The system usability scale (SUS) is a 10-item scale that measures user satisfaction with a system, while the questionnaire for user interaction satisfaction (QUS) is a 50-item scale that measures the quality of a user’s interaction with a system. CSUQ, for computer system usability questionnaire, is a 19-item scale that measures the overall usability of a system. Words stand for Words to Rate Software, a 12-item scale that measures user attitude toward software.

The graph in Figure 2.17 suggests that SUS is the most accurate and consistent method for measuring the usability of a system, as it has the highest percentage of “correct” conclusions with the smallest sample size. QUS and CSUQ are less accurate and consistent than SUS but more so than Words, which has words as the least accurate and consistent method, as it has the lowest percentage of “correct” conclusions with the largest sample size.

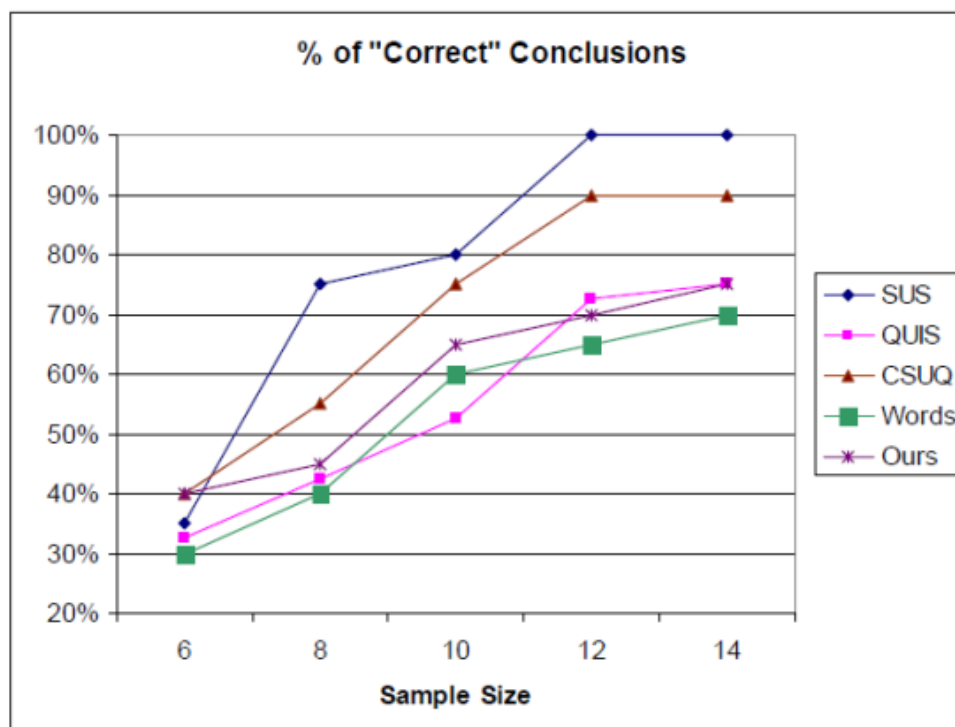


Figure 2.17: Comparison of Questionnaires [18].

### 2.6.3 System scale

Each participant is tasked with rating each statement using a 5-point Likert scale, ranging from “Strongly Disagree” (1) to “Strongly Agree” (5). On this scale, assigning a value of 1 indicates that the participant strongly disagrees with the statement, 2 indicates disagree, 3 indicates neutral, 4 indicates agree, and 5 indicates strong agreement. Intermediate values represent graded degrees of agreement or disagreement. In the context of the SUS scale, higher scores suggest a positive perception of system usability, while lower scores indicate the opposite. This detailed assessment is explained in the Figure 2.18.

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure 2.18: usability system scale [19].

## 2.6.4 System Score

Subsequently, a calculation is performed to obtain an overall usability score for the system. Figure 2.19 addresses the SUS Score, which is a measure of the usability of a system or product. The SUS Score ranges from 0 to 100 and is divided into acceptability ranges, alphabetical grades, and descriptive adjectives. For instance, a SUS Score of 68 is considered marginal, earns a grade of C, and is described as acceptable. A SUS Score of 85 is deemed acceptable, receives an A grade, and is described as excellent.

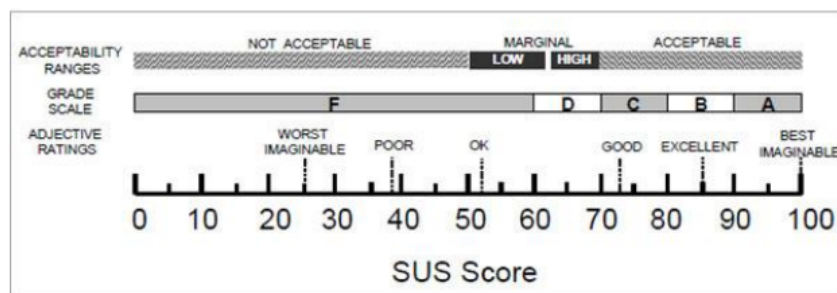


Figure 2.19: SUS Score [18].

# Chapter 3

## State of the Art

### 3.1 Classification of Galapagos Birds

Exploring avian diversity in the Galapagos Islands highlights the importance of understanding the variety of endemic species that have endured over time in this unique environment. Among the species present on these islands, such as Darwin’s finches, mockingbirds, flycatchers and canaries as illustrated in Figure 3.1. The critical need to conserve and preserve this exceptional ecosystem is evident as each of these species has distinctive characteristics, underlining the importance of adopting specific measures to ensure their survival in this fragile environment [61].

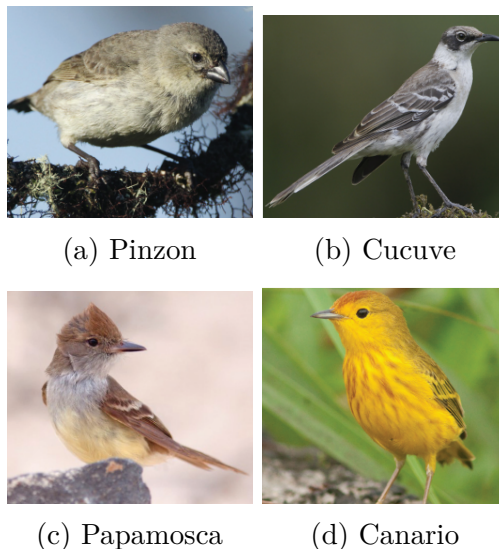


Figure 3.1: Galapagos birds [20].

### 3.1.1 Methods Used in Previous Research

Species identification and preservation are aided by the classification of birds in environments such as the Galapagos Islands, providing information. This process provides clues about changes in the environment, facilitating the monitoring and preservation of biodiversity. Over time, various methods have been employed in these investigations. Table 3.1 presents some prominent studies in bird classification, detailing the methods used, the metrics obtained, and the datasets employed. Each of these approaches will be explored below.

Table 3.1: Previously applied methods

Area	Method	Accuracy	Dataset	Cite
Bird Image Classification	CNN	75%	CUB-200-2011	[62]
Bird Image Classification	Approach based on color characteristics	90%	CUB-200-2011	[63]
Bird Image Classification	Transfer Learning using the Mask-RCNN pre-training model	55.67%	Indian Bird Dataset	[64]
Bird Image Classification	CNN and radar and image data fusion	94.63%	9312 Original Images	[65]
Bird Sound Analysis	RNN/CNN	67%	(CBC) 2020	[66]
Fruit Image Classification	Transfer Learning with GANs	88.75%	Lemon Dataset	[67]
Bird Sound Classification	SVM, ANN, DT	97.5%	400 bird recordings	[68]
Bird Image Classification	Transfer Learning using the pre-trained model ResNet-50	98.8%	Kaggle-180-birds	[69]
Bird Sound Analysis	ResNet50, DenseNet201, InceptionV3, Xception y Efficient Net	97.43%	BirdCLEF2020	[70]
Bird Image Classification	Transfer Learning using the pre-trained ResNXt model	84.43%	CUB-200-2011	[71]
Bird Image Classification	DenseNet201, InceptionResNetV2, MobileNetV2, ResNet50, ResNet152V2 y Xception	96.71%	5000 Bangladesh images	[72]

The study mentioned in [62] employs a CNN with four convolutional layers, two clustering layers and one fully connected layer on the CUB-200-2011 dataset, an accuracy of 75% is achieved in the classification of 200 bird categories. This work highlights the relevance of using advanced methods for the identification and protection of endangered species.

The article [63] presents a method focusing on color features extracted from images. Using color descriptors together with an SVM classifier, the study achieves an outstanding accuracy of 93.5% using the CUB-200-2011 database. Compared to shape- and texture-based approaches, the superiority of color in bird classification is highlighted. Results on the dataset reveal a 90.5% correct segmentation rate, underlining the effectiveness of color feature-based approaches for avian classification and their contribution to machine vision in biology.

The paper [64] introduces a transfer learning-based approach with multistage training for bird species classification. Developed during the Third Workshop on Computer Vision Applications (WCVA 2018), the study uses the pre-trained Mask-RCNN model along with a joint model with Inception Nets to achieve accurate localization and identification of bird species in images. Although it achieves an F1 score of 55.67% on the Indian Bird Dataset dataset, this approach does not represent a promising solution to the bird species classification challenge, given the variability among species and the need for fine-grained feature learning and accurate localization.

J. Niemi and J. T. Tantt [65] present a system for bird identification using technologies such as radar, a reflex camera, a pre-trained CNN, and a data augmentation method. The combination of radar data and CNN predictions achieves a high sensitivity of 94.63% on an original data set of 9312 images. It highlights the effectiveness of data augmentation and its positive impact on classifier performance, with outstanding accuracy for critical species such as white-tailed eagles and lesser black-backed gulls, and its applications in offshore wind farm environmental monitoring.

In [66] highlights the application of deep learning techniques for large-scale acoustic analysis of birds. Using spectrograms from the Cornell Bird Challenge Dataset (CBC) 2020, the authors propose a hybrid model that combines a CNN for the spatial representation and a Recurrent Neural Network (RNN) for the temporal component. The approach achieves an average accuracy of 67% in the classification of 100 bird species, with an outstanding 90%

accuracy for the red crossbill. In addition, visual analysis reveals intuitive groupings among related species, providing a unique empirical interpretation of the learned representations.

An Interesting idea presented in [67] is a strategy to improve the classification of fruit quality and defect images using data augmentation generated by a Conditional Generative Adversarial Network (GAN). The study highlights an approach that combines fine-tuning, transfer learning, and generative model-based data augmentation, achieving 88.75% accuracy in the classification of lemon images. The authors suggest that conditional GANs effectively address data limitations by generating additional data. This method shows promising potential in image classification of fruit quality and defects, applicable in image processing and classification of natural and organic elements. A lemon dataset was used in the training process.

The article [68] highlights the relevance of bird classification based on sound patterns, contributing significantly to ornithology. By applying various machine learning algorithms, such as support vector machines (SVM), ANN and decision trees (DT), this study provides tools to understand species and behaviors through the analysis of vocalizations. By achieving 97.5% accuracy in classifying birds in a dataset comprising 400 recordings, this approach marks a milestone in the technical analysis of bird sounds, being essential for conservation and environmental monitoring.

The study mentioned in [69] addresses bird species classification using a pre-trained CNN ResNet-50 in ImageNet, adapted to the Kaggle-180-birds dataset with 180 species. The method employs fine-tuning and data augmentation strategies, achieving an impressive 98.8% accuracy in bird species classification, outperforming other transfer learning approaches. The study also explores the challenges and applications of bird species classification, highlighting its relevance to avian research and conservation.

Kumar et al. [70] highlights using deep learning and transfer learning models in bird sound classification. Using an extensive BirdCLEF2020 dataset, various feature extraction techniques are applied. Models such as ResNet50, DenseNet201, InceptionV3, Xception, and Efficient Net show remarkable effectiveness, with DenseNet201 and ResNet50 standing out with an accuracy of 97.43% on the validation set. This approach demonstrates the effectiveness of knowledge transfer for predicting bird species from sound recordings.

The paper [71] highlights the use of deep learning and transfer learning techniques in



bird species classification, focusing on the ResNeXt model as a reference. The research optimizes performance by tweaking the loss function, learning rate, and implementing techniques such as data augmentation and a full connection layer. The model achieves a remarkable recognition rate of 84.43% on the CUB-200-2011 dataset. These findings underscore the effectiveness of transfer learning in accurate avian species identification, offering valuable insights for avian recognition systems.

In [72] addresses local bird recognition in Bangladesh using various CNN architectures and transfer learning. Six pre-trained models are evaluated, including DenseNet201, InceptionResNetV2, MobileNetV2, ResNet50, ResNet152V2 and Xception. These models are fitted to a limited dataset consisting of 5000 bird images from Bangladesh by applying data augmentation techniques. Among the evaluated models, MobileNetV2 stands out with a performance of 96.71%.

# Chapter 4

## Methodology

This chapter will discuss the methodology used in this thesis, divided into several stages represented in Figure 4.1.

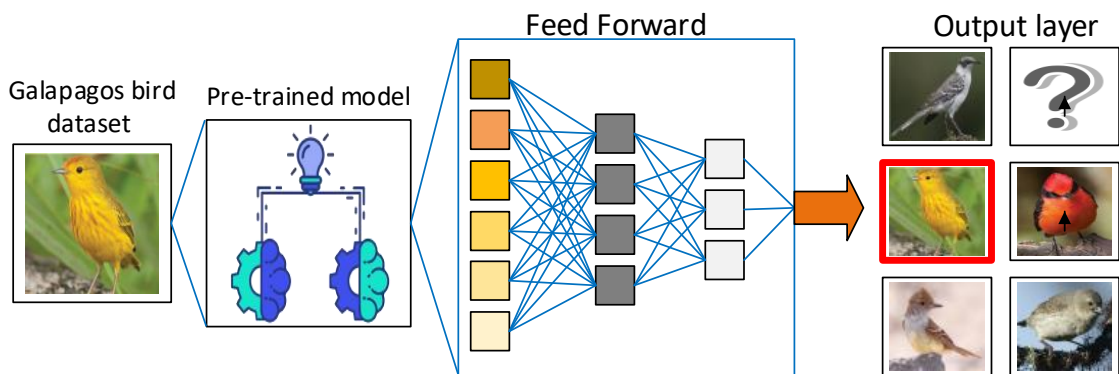


Figure 4.1: General Model

As can be seen, the first stage involves the creation of the data, generating information from scratch that will serve as input for our pre-trained model. Among the pre-trained models that we will employ are DenseNet\_201, EfficientNetB0, InceptionV3, MobileNetV2, ResNet50, VGG16 and VGG19. In our models we will apply fine tuning with different models and at the end a feed forward architecture. Depending on the results obtained, we will select the most effective model for the bird classification task.

## 4.1 Data Collection

This section will address the data collection process, which is a fundamental part of this work. Data collection was carried out based on the project's central purpose, which focuses on assisting in identifying animals in the Galapagos Islands, specifically with regard to bird species. This process is based on searching and obtaining data on various open-access platforms like Kaggle <sup>1</sup>.

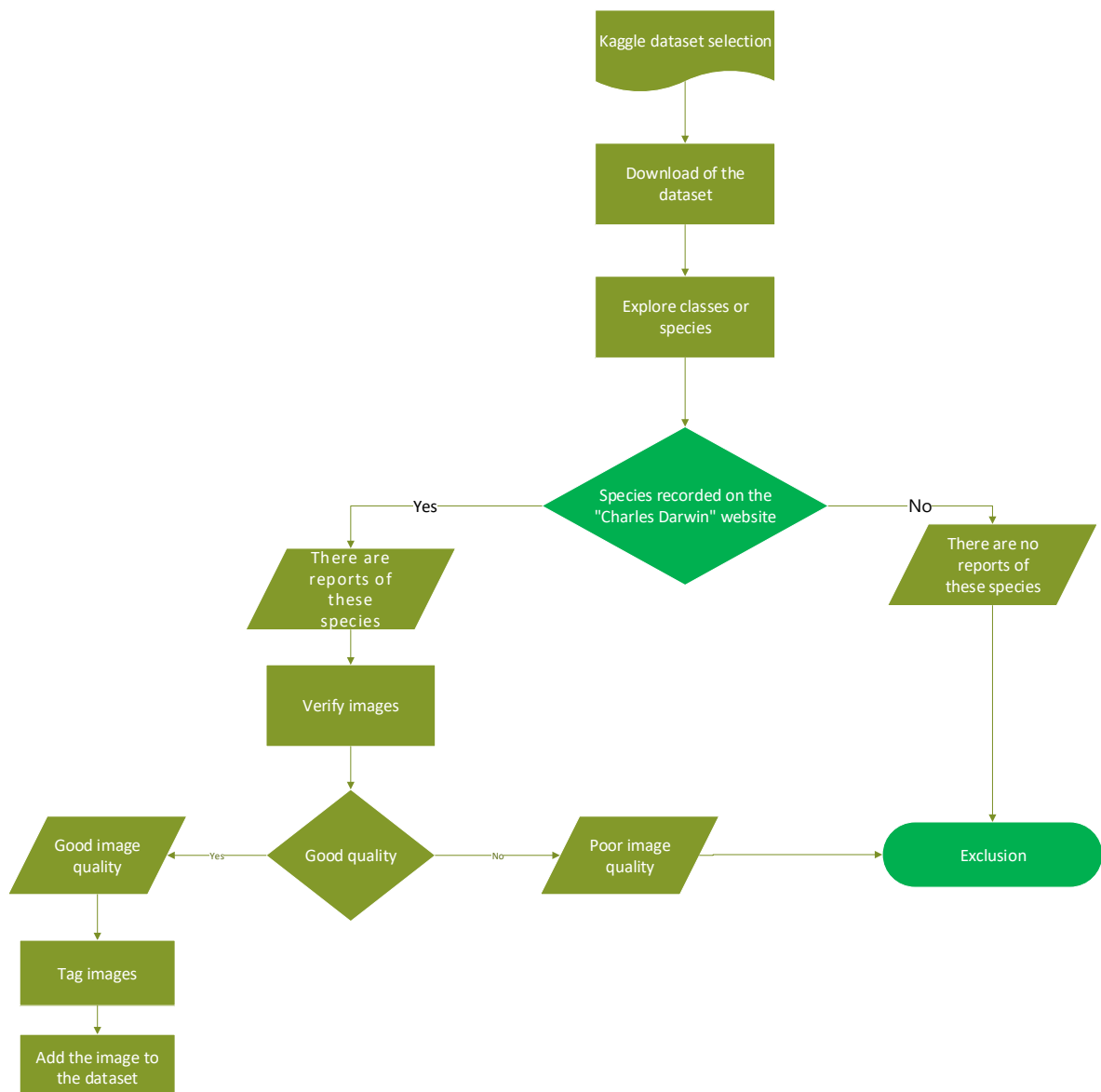


Figure 4.2: Data flow diagram

<sup>1</sup>Kaggle datasets, available at: <https://www.kaggle.com/>

Figure 4.2 illustrates the process carried out for the creation of the dataset that has been segmented in several stages:

- **Building dataset for Galapagos:** Initially, we downloaded the data from the Kaggle platform, as it is essential to review and verify all the information contained in the base datasets that we will use to create a training dataset.
- **Folder Classification:** In the data collection phase, a manual classification of folders was carried out to ensure correspondence with the official “Charles Darwin Foundation” information, as shown in Figure 4.3. Each class or species in the Kaggle datasets was verified using information provided on the foundation’s website. This meticulous process resulted in the identification of 58 classes or species, encompassing those present in the Galapagos Islands according to official information from the foundation.
- **Building Galapagos bird dataset:** With the 58 classes or species identified, we created a new dataset that will be used to train pre-trained models. This new dataset contains exclusively bird species from the Galapagos Islands.

HOME ABOUT US RESEARCH DATAZONE BLOG VACANCIES GIVE MENU ☰

Search by Species Name or Family

Filter by Species Checklist

Filter by IUCN Category

Filter by Origin

48 results found in species checklist 'Aves' with taxon origin 'Endemic':

- *Anas bahamensis galapagensis* (Ridgway, 1890) — Ánade Cariblanco de Galápagos, Patillo de Galápagos, White-cheeked Pintail
- *Anous stolidus* (Linnaeus, 1758) — Gaviotín de cabeza blanca, Brown Noddy, common noddy
- *Ardea herodias cognata* Bangs, 1903 — Garzón Azul de Galápagos, Garza morena, Great Blue Heron
- *Ardea herodias* Linnaeus, 1758 — Garza morena, Great Blue Heron
- *Asio flammeus* (Pontoppidan, 1763) — Lechuza de campo, Short eared Owl
- *Asio flammeus galapagoensis* (Gould, 1837) — Lechuza de campo, Short eared Owl
- *Buteo galapagoensis* (Gould, 1837) — Gavián de Galápagos, Galapagos Hawk
- *Butorides striata sundevalli* (Linnaeus, 1758) — Garcilla de Lava, Garza estriada, Striated Heron
- *Camarhynchus heliobates* (Snodgras & Heller, 1901) — Pinzón Manglero, Pinzón de mangle, Mangrove Finch
- *Camarhynchus pallidus* (P.L.Sclater & Salvin, 1870) — Pinzón carpintero, artesano, Woodpecker Finch
- *Camarhynchus parvulus* (Gould, 1837) — Pinzón de árbol pequeño, Pinzón arboreo pequeño, Small Tree Finch
- *Camarhynchus pauper* Ridgway, 1890 — Pinzón de Árbol Mediano, Pinzón arboreo mediano, Medium Tree Finch
- *Camarhynchus psittacula* Gould, 1837 — Pinzón de Árbol Grande, Pinzón arboreo grande, Large Tree Finch
- *Certhidea fusca* Sclater & Salvin, 1870 — Pinzón Reinita Gris, Pinzón cantor gris, Gray Warbler Finch
- *Certhidea olivacea* Gould, 1837 — Pinzón Reinita Verde, Pinzón cantor verde, Green Warbler Finch

Figure 4.3: Charles Darwin Foundation[21]

The dataset process will be explained in more detail in the following subsection.

### 4.1.1 Galapagos Bird Image Data Sources

In the data collection process, three specific sources of datasets available on Kaggle were leveraged, as detailed in Table 4.1. These sources proved crucial in providing images of Galapagos birds and other species. The exhaustive search involved reviewing each folder to identify the species, utilizing information from the Charles Darwin Foundation page<sup>2</sup>, which classifies birds based on their status on the island (endemic, native, vagrant, introduced, visiting, etc.). This process allowed for the creating of a dataset from scratch, enriched with relevant information to train the models effectively.

Data Source	Relevant Information	Cite
<b>25 Indian Bird Species</b>	<ul style="list-style-type: none"> <li>• Total Images: 22,600</li> <li>• Creator: Arjun Basandrai</li> <li>• 25 classes</li> </ul>	[73]
<b>Birds 525 Species-Image Classification</b>	<ul style="list-style-type: none"> <li>• Total Images: 84,635</li> <li>• Creator: Gerald Piosenka</li> <li>• 525 classes</li> </ul>	[74]
<b>200 Bird Species</b>	<ul style="list-style-type: none"> <li>• Total Images: 11,788</li> <li>• Creator: Data Scientist at NA Bengaluru</li> <li>• 200 classes</li> </ul>	[75]

Table 4.1: Data Sources Information

<sup>2</sup><https://www.darwinfoundation.org/en/datazone/checklist>

### 4.1.2 Galapagos bird dataset

A metadata has been created, as depicted in Table 4.2, showcasing crucial information about the birds. This metadata is segmented into categories: Scientific Name, Common Name in English, Geographic Distribution, Habitat, and Conservation Status. A comprehensive metadata has been generated for the 58 species comprising the dataset, offering a detailed and organized insight into essential information about each bird.

Scientific name	Common English name	Geographical distribution	Habitat	State of conservation
Diomedea exulans	Wandering albatross	Oceans of the southern hemisphere, especially the Atlantic and Indian oceans	Pelagic zones, far from the coast	Vulnerable according to IUCN
Fulica americana	American coot	North, Central and South America, from Canada to Tierra del Fuego	Lakes, lagoons, marshes and rivers with aquatic vegetation	Least concern according to IUCN
...	...	...	...	...
Numenius phaeopus	Whimbrel	Worldwide, except Antarctica and some oceanic islands	Coasts, estuaries, saltwater lagoons and beaches	Least concern according to IUCN

Table 4.2: Metadata extract [8]

## 4.2 Model selection

For model selection, the pre-trained models mentioned in Chapter 2 will be used, which are DenseNet\_201, EfficientNetB0, InceptionV3, MobileNetV2, ResNet50, VGG16 and VGG19. Transfer of learning will be performed during training of a sequential model. The diagram in Figure 4.4 depicts this process, starting with the dataset as input, followed by the pre-trained model. Subsequently, a feedforward architecture representing the sequential model to be trained is incorporated. This approach capitalizes on the prior knowledge of the pre-trained models to improve the effectiveness of the sequential model in the classification task. In the following subsections, we will explain the process.

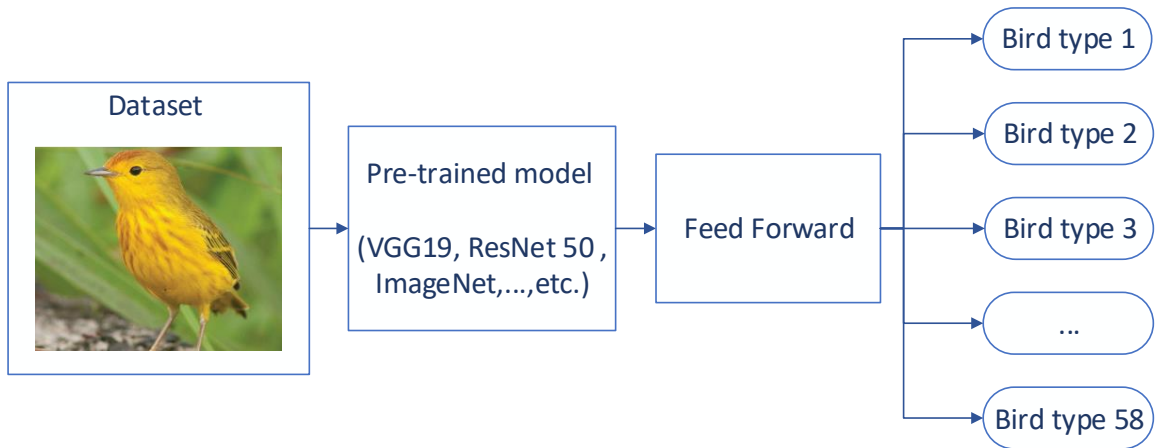


Figure 4.4: Training Diagram

### 4.2.1 Sequential model

We have developed a sequential model to apply transfer learning in this case. A visual representation of how this sequential model is configured and the various hyperparameters we employ with over-pre-trained models is presented in Table 4.3.

	<b>Parameter</b>	<b>Value</b>
<b>Architecture</b>	Neural network type	Feedforward
	Layer type	Dense
	Intermediate layer type	Dropout (0.3)
	Input activation function	ReLU, Leaky ReLU
	Intermediate activation function	ReLU, Leaky ReLU
	Output activation function	softmax
<b>Learning process</b>	Loss function	categorical cross-entropy
	Optimizer	adam
	Metrics	accuracy, F1-Score
<b>Network training</b>	Validation dataset	0.1
	Testing dataset	0.1
	Number of epochs	30
	Validation-based	Early stopping

Table 4.3: Neural Network Configuration

About the hyperparameters outlined in Table 4.3, two activation functions, namely ReLU and Leaky ReLU, were tested, with the latter chosen to address the problem of dead neurons [76]. The softmax activation function was applied for the output layer, suitable for multiclass classification involving a 58 classes. Categorical Cross Entropy served as the loss

function, designed to evaluate the model's performance in multiclass classification by measuring the disparity between predicted and actual probability distributions of the classes. The adam optimizer, a stochastic optimization algorithm, was selected to adjust the neural network weights. The Dropout regularization technique was incorporated to prevent overfitting and enhance model generalization by randomly deactivating some neurons during training. Early stopping technique was employed. This method establishes a threshold to halt training if the model shows no improvement, aiming to enhance generalization and prevent overfitting.

The dataset splitting was divided into 80% for training, 10% for validation, and an additional 10% for testing. To augment the dataset, an augmentation technique was employed, involving the creation of additional training examples through random transformations of the existing images, as illustrated in the Figure 4.5, the. This helps enhance the model's ability to generalize to unseen data and improves overall performance. A more complete description of the above-mentioned concepts is given in chapter 2.

The data enhancements that were made are as follows:



Figure 4.5: Data augmentation



- **Rescale:** Normalize pixel values (1./255) for deep learning model training.
- **Rotation\_range:** Randomly rotate within a 20-degree range for increased data variability.
- **Width\_shift\_range and Height\_shift\_range:** Randomly shift horizontally or vertically (20%) for varied camera positions and increased model robustness.
- **Shear\_range:** Apply shear transformation (up to 0.2 radians) for simulated perspectives and distortions.
- **Zoom\_range:** Zoom in/out (up to 20%) for varied camera zooming and model scale invariance.
- **Horizontal\_flip:** Horizontally flip for varied orientations and increased data symmetry.

#### 4.2.2 Machine Learning Performance Metrics

In image classification as in other areas, the key metrics for evaluating model performance are Accuracy, F1-Score, Precision, and Recall [77]. These metrics play a crucial role in providing a comprehensive measure of the quality of a model's predictions.

These metrics are defined as follows:

- **Accuracy:** Accuracy is a measure of the overall correctness of a prediction, including both positive and negative outcomes. It is determined by dividing the sum of true positives (TP) and true negatives (TN) by the sum of all categories (including true positives, true negatives, false positives, and false negatives).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

- **Precision:** Precision measures the proportion of correct positive predictions out of all positive predictions. It is calculated as the number of true positives (TP) divided by the sum of true positives and false positives (FP).

$$\mathbf{Precision} = \frac{TP}{TP + FP} \quad (4.2)$$

- **Recall:** Commonly known as sensitivity or true positive rate, recall gauges the percentage of real positive instances accurately recognized by a model. It is computed by dividing the count of true positives by the sum of true positives and false negatives (FN).

$$\mathbf{Recall} = \frac{TP}{TP + FN} \quad (4.3)$$

- **F1-Score:** The F1-Score is a metric that combines precision and recall into one value. It aims to balance these two metrics and is particularly useful when there is a class imbalance. Use the following formula to calculate:

$$\mathbf{F1-Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (4.4)$$

It is important to note that, in this analysis, we will focus on using Accuracy and F1-Score. The choice of F1-Score inherently includes Precision and Recall metrics, making it a comprehensive metric for evaluating the balance between precision and recall of the model.

## 4.3 Model Training

In this thesis, we have employed several platforms and tools to perform our work efficiently. We have used cloud computing specifically Google Colab played a crucial role in allowing us to write and run training codes with various configurations. We took advantage of the "T4" type GPU available on this platform.

Among the most prominent tools were the TensorFlow libraries [78], which were fundamental for feature extraction and the use of pre-trained models. These architectures played a crucial role in training and performance evaluation, allowing us to adjust their configurations according to our specific needs.

### 4.3.1 Experiments to be performed

Comparison between various models will allow us to identify the specific strengths and weaknesses of each, providing us with the ability to make informed decisions to improve our solutions. In addition, it will provide a clear understanding of the effectiveness of the selected model for the bird species classification task compared to other alternatives.

- **Experiment 1:** Training will be conducted both with and without data augmentation to assess the influence of this technique on performance.
- **Experiment 2:** In this experiment, the training will include fine tuning by unfreezing a given number of layers, with 15 and 4 layers being selected respectively. The evaluation will be performed both with and without the use of data augmentation.
- **Experiment 3:** A comparison will be made between the previously trained models and those not trained.
- **Experiment 4:** A comparison of training with different activation functions, ReLU and Leaky ReLU, will be conducted to determine which one yields better performance.

In the following chapter, we will elucidate the results obtained from the experiments.

# Chapter 5

## Results and Discussion of the models evaluated

We will now analyze the results obtained in each of the experiments to gain a detailed understanding of the efficacy and performance of the models evaluated so that we can present the evolution of the training and validation for the best architecture.

### 5.1 Experiment 1

In this analysis, a comparative table of untrained models is presented, each using their architectures without recourse to pre-trained weights. Table 5.1 details the performance of various neural network model architectures on a specific data set, evaluating their accuracy and F1-Score both with and without data augmentation. Among the models considered, EfficientNetB0 emerges as the one that yielded the most favorable outcomes, demonstrating an improvement with data augmentation, achieving an accuracy of 0.66% and an F1-Score of 0.55%. The application of data augmentation extends the total time to 9211.77 seconds. On the other hand, DenseNet\_201 stands out as the best model in this experiment, demonstrating improvement without data augmentation, achieving an accuracy of 0.63% and an F1-Score of 0.51% in a shorter time of 7737.49 seconds. It is possible that the overall inferior performance observed in models without pre-trained weights is due to a lack of adaptation to specific patterns in the data set, given that they lack prior knowledge. In summary, EfficientNetB0 stands out as the strongest option, and it is suggested to explore the use of pre-trained models to boost the overall performance of these untrained models.

Architecture	Method	Accuracy	F1 Score	Time (s)
DenseNet_201	Without data augmentation	0.63	0.51	7737.49
	With data augmentation	0.10	0.00	3236.83
<b>EfficientNetB0</b>	Without data augmentation	0.44	0.27	2999.91
	With data augmentation	0.66	0.55	9211.77
InceptionV3	Without data augmentation	0.12	0.00	2039.03
	With data augmentation	0.33	0.14	4619.67
MobileNetV2	Without data augmentation	0.14	0.00	4615.17
	With data augmentation	0.13	0.00	5665.00
ResNet50	Without data augmentation	0.32	0.20	5970.94
	With data augmentation	0.31	0.13	5202.54
VGG16	Without data augmentation	0.13	0.00	2095.27
	With data augmentation	0.10	0.00	3772.54
VGG19	Without data augmentation	0.46	0.35	8071.08
	With data augmentation	0.13	0.00	3881.19

Table 5.1: Experiment 1

In the Figure 5.1, the evolution of the highest performing architecture and how the loss varied over the epochs in the EfficientNetB0 architecture is presented.

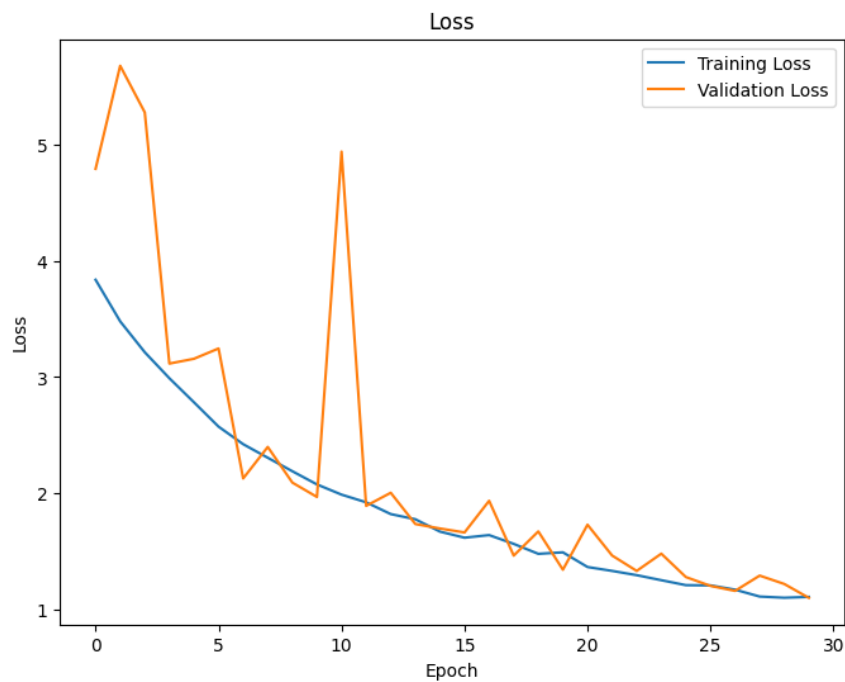


Figure 5.1: EfficientNetB0 Loss vs Epoch

## 5.2 Experiment 2

In these evaluations, we focused on the fine-tuning process by unfreezing 15 and 4 layers of each model, aiming to adapt them to the prior knowledge acquired during the initial training. Table 5.2 highlights that the most outstanding model was DenseNet\_201. Without data augmentation, it achieved an accuracy of 0.41% and an F1-Score of 0.23% in 2500.69 seconds. Conversely, the model that showed the best performance with data augmentation was VGG19, with an accuracy of 0.27% and an F1-Score of 0.15%, although with a longer training time of 4,342.29 seconds.

In the Table 5.3, the VGG16 model stood out as the most notable model. Without data augmentation, it achieved an accuracy of 0.70 and an F1-Score of 0.58% in 2930.18 seconds. On the other hand, the model that achieved the best performance with data augmentation was VGG19, with an accuracy of 0.74% and an F1-Score of 0.64%, albeit with a longer training time of 4876.27 seconds.

Architecture	Method	Accuracy	F1 Score	Time (s)
DenseNet_201	Without data augmentation	0.41	0.23	2500.69
	With data augmentation	0.19	0.05	6377.05
EfficientNetB0	Without data augmentation	0.13	0.00	3699.43
	With data augmentation	0.12	0.00	6783.19
InceptionV3	Without data augmentation	0.36	0.22	2509.27
	With data augmentation	0.25	0.08	3761.70
MobileNetV2	Without data augmentation	0.14	0.00	2019.81
	With data augmentation	0.04	0.00	3859.17
ResNet50	Without data augmentation	0.19	0.07	3091.98
	With data augmentation	0.12	0.02	3160.04
VGG16	Without data augmentation	0.09	0.00	5022.81
	With data augmentation	0.19	0.06	5975.80
VGG19	Without data augmentation	0.11	0.00	3761.46
	With data augmentation	0.27	0.15	4342.29

Table 5.2: Experiment 2 unfreezing 15 layers

Architecture	Method	Accuracy	F1 Score	Time (s)
DenseNet_201	Without data augmentation	0.38	0.21	4276.64
	With data augmentation	0.22	0.07	7382.31
EfficientNetB0	Without data augmentation	0.12	0.00	5400.99
	With data augmentation	0.09	0.00	3804.72
InceptionV3	Without data augmentation	0.36	0.22	1835.82
	With data augmentation	0.23	0.10	5322.55
MobileNetV2	Without data augmentation	0.12	0.00	1739.17
	With data augmentation	0.11	0.00	3461.60
ResNet50	Without data augmentation	0.25	0.09	2241.37
	With data augmentation	0.14	0.02	6858.71
VGG16	Without data augmentation	0.70	0.58	2930.18
	With data augmentation	0.71	0.58	3349.66
<b>VGG19</b>	Without data augmentation	0.69	0.57	2558.24
	With data augmentation	0.74	0.64	4876.27

Table 5.3: Experiment 2 unfreezing 4 layers

As shown in Figure 5.2, better performance is observed when unfreezing four layers, with the VGG19 model offering the best performance in both training and validation losses.

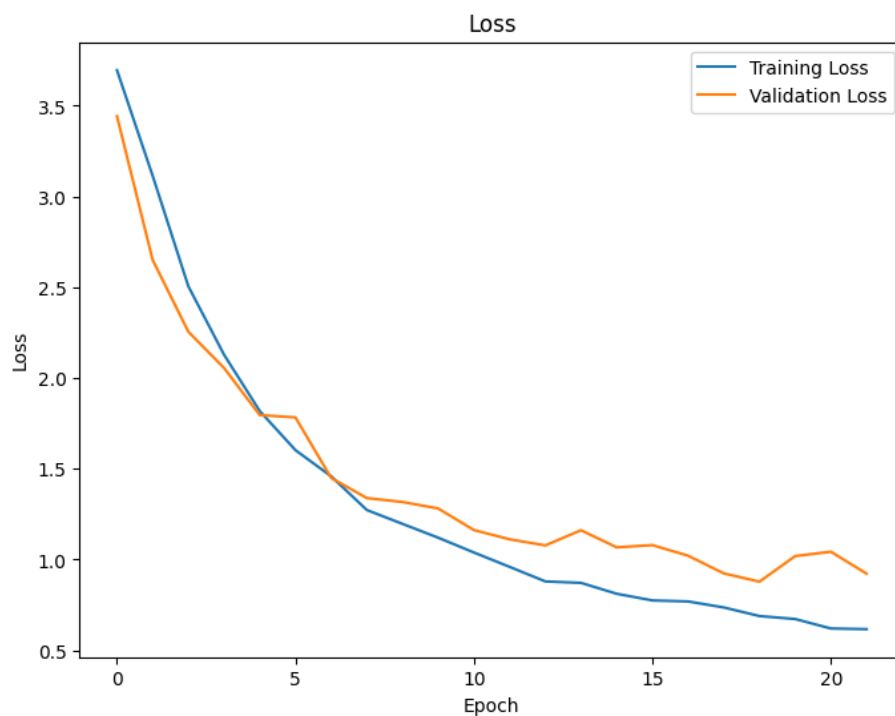


Figure 5.2: VGG19 (4 layers) Loss vs Epoch

The choice of unfreezing 15 and 4 layers is based on experimental considerations, as prior experiments were conducted with different numbers of unfrozen layers. We opted for the configuration that exhibited better performance compared to other options. It is essential to note that the model's adaptation to the unfreezing process can vary, as evidenced in previous works [79, 80]. Where different numbers of unfrozen layers were used, and a more effective adaptation was observed. In this context, there is no specific number, and the choice is based on the model's adaptability to the unfreezing experimentation.

### 5.3 Experiment 3

In this context, a comparative table has been prepared for trained and untrained models. It is important to note that, in the case of untrained models, we utilized the same architecture as pretrained models but without activating the previously learned weights. This implies that these models share the same structure but do not leverage prior knowledge. Furthermore, pretrained models were selected for their outstanding performance in previous tests with various configurations, choosing the best results from Table 5.2 and Table 5.3.

Upon reviewing Table 5.4, it stands out that the model with the best results was DenseNet\_201 without data augmentation, achieving an accuracy of 0.63%, an F1-Score of 0.51%, and a time of 7737.49 seconds. Concerning data augmentation but with untrained models, the top-performing model was EfficientNetB0, with an accuracy of 0.66%, an F1-Score of 0.55%, and a time of 9211.77 seconds. Meanwhile, selected from Table 5.3, which includes pretrained models with data augmentation, VGG19 achieved an accuracy of 0.74%, an F1-Score of 0.64%, and a time of 4876.27 seconds; and without data augmentation, VGG16 obtained an accuracy of 0.70%, an F1-Score of 0.58%, in a time of 2930.18 seconds. Thus, we can appreciate that using pretrained models can yield better results.



Architecture	Method	Accuracy	F1 Score	Time (s)
DenseNet_201	without prior information	0.63	0.51	7737.49
	with prior information	0.41	0.23	2500.69
EfficientNetB0	without prior information	0.66	0.55	9211.77
	with prior information	0.13	0.00	3699.43
InceptionV3	without prior information	0.33	0.14	4619.67
	with prior information	0.36	0.22	1835.82
MobileNetV2	without prior information	0.14	0.00	4615.17
	with prior information	0.14	0.00	2019.81
ResNet50	without prior information	0.32	0.20	5970.94
	with prior information	0.25	0.09	2241.37
VGG16	without prior information	0.13	0.00	2095.27
	with prior information	0.71	0.58	3349.66
<b>VGG19</b>	without prior information	0.46	0.35	8071.08
	with prior information	0.74	0.64	4876.27

Table 5.4: Experiment 3

In this comparison, the progression of the highlighted models without prior knowledge and with prior knowledge is examined. In the Figure 5.3, it is evident that the model as VGG19 unfreezing 4 layers incorporating prior information obtained superior results.

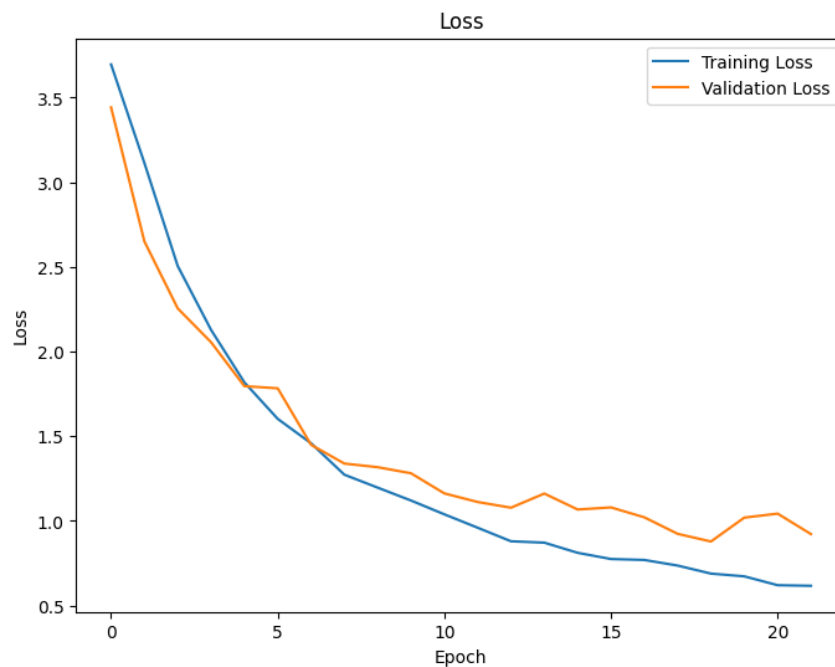


Figure 5.3: VGG19 (4 layers) Loss vs Epoch

## 5.4 Experiment 4

In this comparison, we have employed two different activation functions, ReLU and LeakyReLU, to accurately evaluate the performance of each. Table 5.5 presents the results of training with the unfreezing of 4 layers and ReLU or Leaky ReLU activation functions.

Upon analyzing the table, VGG19 model with ReLU activation achieved an accuracy of 0.74%, an F1-score of 0.64%, and a training time of 4876.27 seconds. On the other hand, with Leaky ReLU activation, superior performance was attained, with an accuracy of 0.76%, an F1-score of 0.69%, and a slightly more efficient training time of 4015.15 seconds. This improvement is due to the implementation of the Leaky ReLU activation function, which solves problems linked to inactive neurons, leading to better results.

Architecture	Method	Accuracy	F1 Score	Time (s)
DenseNet_201	ReLU	0.41	0.23	2500.69
	LeakyReLU	0.37	0.24	5064.55
EfficientNetB0	ReLU	0.13	0.00	3699.43
	LeakyReLU	0.12	0.00	3814.81
InceptionV3	ReLU	0.36	0.22	1835.82
	LeakyReLU	0.39	0.25	5579.54
MobileNetV2	ReLU	0.14	0.00	2019.81
	LeakyReLU	0.14	0.02	2815.63
ResNet50	ReLU	0.25	0.09	2241.37
	LeakyReLU	0.16	0.09	2168.70
VGG16	ReLU	0.71	0.58	3349.66
	LeakyReLU	0.75	0.62	7182.29
<b>VGG19</b>	ReLU	0.74	0.64	4876.27
	LeakyReLU	0.76	0.69	4015.15

Table 5.5: Experiment 4

For this experiment, we present the Figure 5.4 that represent the evolution of the loss in both training and validation. It can be observed that in terms of good evolution, the Leaky ReLU function performs slightly better.

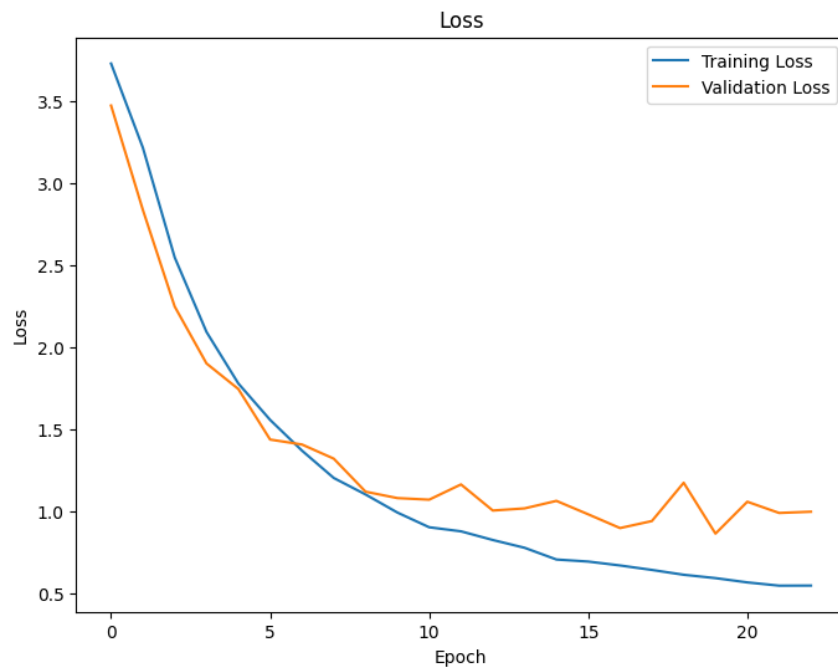


Figure 5.4: VGG19 Leaky ReLU Loss vs Epoch

Table 5.6 displays top results from diverse experiments, considering data augmentation and prior information presence. The standout model was VGG19 with Leaky ReLU activation, employing fine-tuning and unfreezing four layers. It achieved an accuracy of 0.76%, an F1-Score of 0.69%, and a training time of 4015.15 seconds. These results are crucial for selecting the model, which will be exported for web/mobile application implementation.

Experiments	Method	Architecture	Accuracy	F1 Score	Time (s)
1	Without data augmentation	DenseNet_201	0.63	0.51	7737.49
	With data augmentation	EfficientNetB0	0.66	0.55	9211.77
2	Without data augmentation	VGG16	0.70	0.58	2930.18
	With data augmentation	VGG19	0.74	0.64	4876.27
3	without prior learning	EfficientNetB0	0.66	0.55	9211.77
	with prior learning	VG19	0.74	0.64	4876.27
4	ReLU	VGG19	0.74	0.64	4876.27
	<b>LeakyReLU</b>	<b>VGG19</b>	<b>0.76</b>	<b>0.69</b>	<b>4015.15</b>

Table 5.6: General Experiments

# Chapter 6

## Implementation of the Web/Mobile Application

### 6.1 Web/Mobile application

The web application utilizes key TensorFlow components [55] and integrates into a mobile application through servers like ngrok tunnels [81]. The deployment process, illustrated in Figure 6.1, outlines how to implement a TensorFlow model for bird image classification in a mobile application. Access is expedited through a web/mobile application managed by a server, requiring only a link without the need for app installation.

1. **Conversion of the TensorFlow Model to TensorFlow.js:** The model is converted from TensorFlow/Keras model to TensorFlow.js file using the TensorFlow.js library. This file contains the information needed to run the model in a web browser.
2. **Local Hosting of the Model and Application:** The TensorFlow.js file and the bird image classification application are stored locally on a computer and remotely accessible via IP or URL.
3. **Web/Mobile Application:** It utilizes the TensorFlow.js file and the classification application to identify birds in images. Communication with local hosting is done through an ngrok tunnel, exposing the local web server to the internet. The mobile application uses JavaScript, HTML, an interpreter, and cores to process the image and display the results.

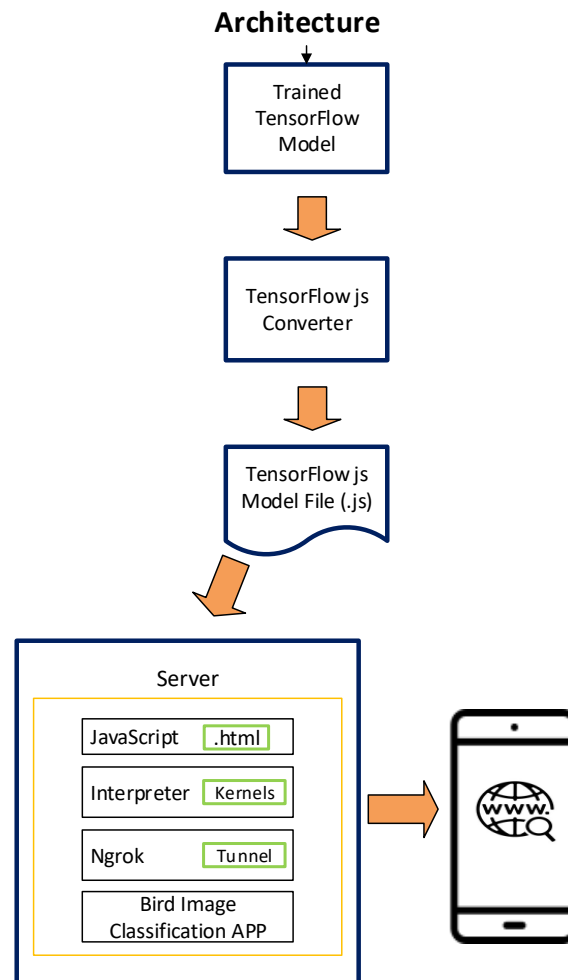


Figure 6.1: Application Model

## 6.2 Conversion of the Model

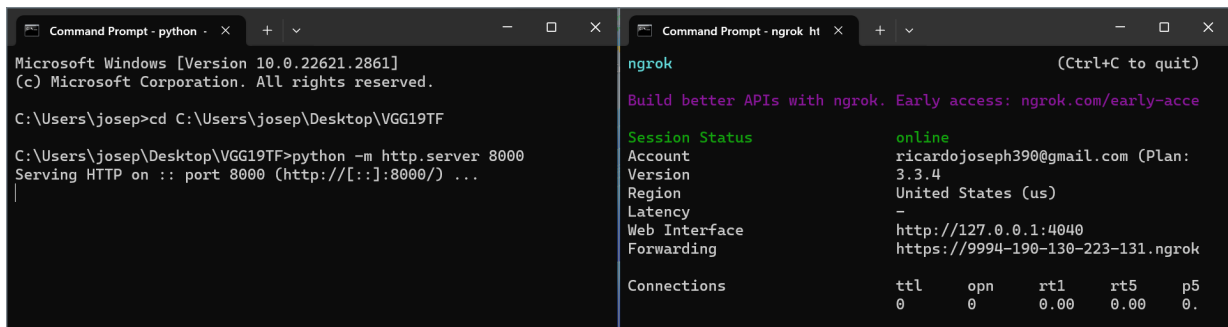
In the model conversion process, after identifying the optimal model from Chapter 5, we save it as an HDF5-formatted file using `#model.save('MYVGG19.h5')`. The HDF5 format is commonly used for storing large datasets, including deep learning models.

Subsequently, the `#!pip` command installs the “tensorflowjs” library, facilitating the conversion of TensorFlow models to formats compatible with JavaScript. A new directory, `#output_folder`, is then created to store the converted model files. The command `#!tensorflowjs_converter --input_format keras MYVGG19.h5 output_folder` transforms the Keras-format model into a TensorFlow.js-compatible format, saving the result in `#output_folder`.

## 6.3 Web Server

In the context of servers, it is essential to understand their primary function. A server, commonly referred to as a computer system, provides services or resources to other computers, called clients, within a network [82]. These services range from the delivery of applications and websites to the management of various user services.

The Figure 6.2 describes the process by which a URL provided by Ngrok is obtained. First using the command prompt, a port is opened in the folder where the mobile application resides and then Ngrok is run on that port. In this way, the application can be executed through a link provided by Ngrok, allowing its use on cell phones and various devices.



```

Command Prompt - python - x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\josep>cd C:\Users\josep\Desktop\VGG19TF
C:\Users\josep\Desktop\VGG19TF>python -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...

Command Prompt - ngrok ht x + v
ngrok (Ctrl+C to quit)
Build better APIs with ngrok. Early access: ngrok.com/early-acce

Session Status      online
Account             ricardojoseph390@gmail.com (Plan:
Version             3.3.4
Region              United States (us)
Latency              -
Web Interface        http://127.0.0.1:4040
Forwarding           https://9994-190-130-223-131.ngrok

Connections          ttl    opn    rt1    rt5    p5
                    0      0      0.00  0.00  0.

```

Figure 6.2: URL process

Ngrok simplifies the creation of secure tunnels between local and remote servers, enabling global access over the Internet without complex configurations. This tool is valuable for the development and testing of web applications across various devices, as well as for sharing local servers without the need for additional software [81]. Figure 6.3, visualizes how clients, identified as “Client 1”, “Client 2” and “Client 3”, connect over the Internet and pass through a firewall without requiring configuration. They use Ngrok as a means to access various servers. The firewall, symbolized by a burning wall icon, facilitates communication between the clients and the Ngrok server. The latter, acting as an intermediary, enables secure connections to three servers labeled “Server 1 Port 8080”, “Server 2 Port 8000” and “Server 3 Port 7000”. In short, ngrok facilitates the secure connection between clients and servers, allowing data to flow efficiently and securely.

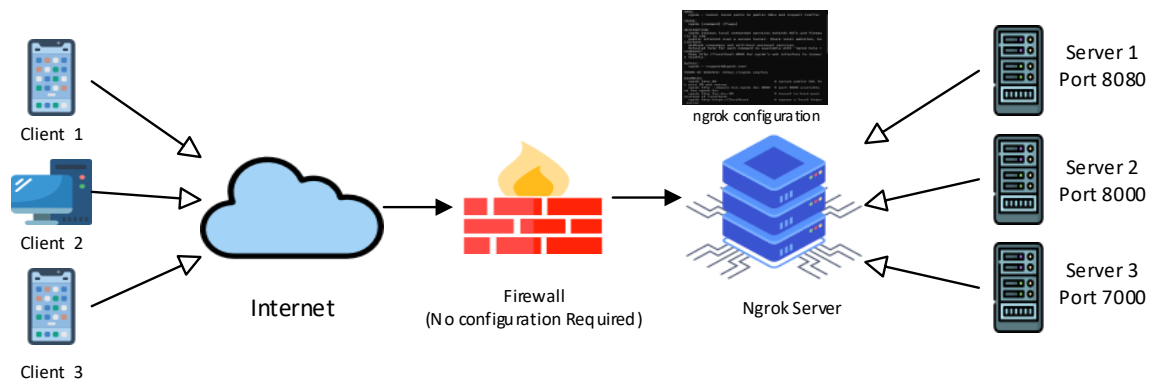


Figure 6.3: Ngrok

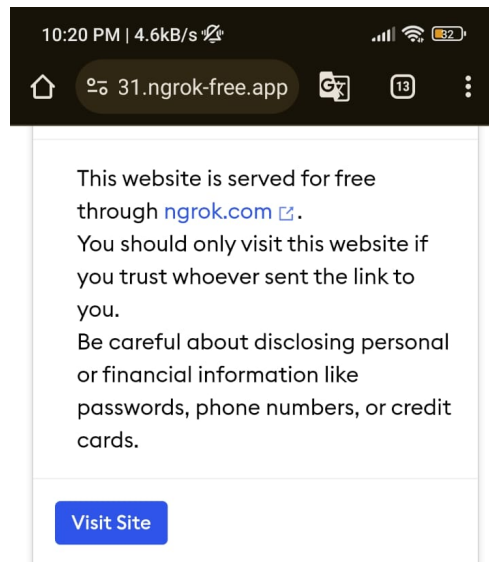
To implement the mobile application, we start by saving the top performance model in “.bin” and “.json” files. These files are obtained by converting a keras model that is stored in a .h5 file to a format that TensorFlow can understand and use. Then, we create an HTML file with JavaScript code to visualize the application as a web platform, allowing the camera to be viewed through a local server on HTTP port 8000.

The use of Ngrok is critical as the use of simplifies the secure exposure of the web application by allowing easy and secure remote access, ensuring a seamless and protected mobile experience. This application allows us to create a secure tunnel between a public network and our local network. By assigning Ngrok to the port, such as 8000 in this case, we obtain a link that can be used on mobile devices. This link allows the identification of birds through the device, effectively transforming it into a mobile application. This approach ensures the security and accessibility of the system on different devices.

## 6.4 User Interface

The application interface provides a straightforward user experience, focused on bird detection through the device’s camera. Upon accessing the link provided by Ngrok, Figure 6.4 shows that Ngrok warns about accessing a site created using secure tunnels, requesting information about whether you are the developer of the page. A ”visit” button redirects you to the mobile app. Ngrok performs this redirect to raise awareness of potential security

and privacy issues before displaying the actual website content. If you trust the site, you can click "Visit Site". If you are the developer, Ngrok provides instructions to remove this warning via a specific header or a paid account.



#### Are you the developer?

We display this page to prevent abuse.  
Visitors to your site will only see it once.

#### To remove this page:

Set and send an `ngrok-skip-browser-warning` request header with any value.  
Or, set and send a custom/non-standard browser `User-Agent` request header.  
Or, please [upgrade](#) to any paid ngrok account.

---

**ngrok** [Learn how ngrok fights abuse](#)

Figure 6.4: Access to the website

After clicking the "Visit Site" link, we are redirected to an interface where the user can choose their type, as depicted in Figure 6.5. This idea of dividing the interface into three categories - children, tourists, and biologists - was suggested by a biology professor. This distinction results in the customization of the information provided to each user in the subsequent interface.





Figure 6.5: User selection

### 6.4.1 Child User

When the user selects the child, they are redirected to the same bird identification page. However, by clicking on “more information”, they are directed to a section specially designed for children. In this section, information about the selected bird is presented in a more understandable manner, tailored to their cognitive level. Aspects such as the bird’s scientific name, diet, and other relevant characteristics are explained in a way that ensures an engaging and accessible informational experience for children. This didactic and mobile-focused approach aims to encourage learning in children as they explore different bird species using their phones, providing an interactive educational experience.

The significance of children learning through this project lies in incorporating crucial information in the section designed for them, facilitating a more enjoyable and playful educational process. By employing a didactic approach tailored to their needs and preferences, it ensures that children can absorb knowledge more effectively, turning the experience of learning about birds into something fun and memorable.

### 6.4.2 Tourist User

Similarly, when dealing with a tourist user, the procedure remains identical, but different information is presented, geared towards an adult audience capable of grasping basic bird details. The data presentation is less childlike and more serious, covering aspects such as scientific name, conservation status, geographical location, among others. This is significant as tourist users can acquire knowledge and learn more about the avian fauna of the Galapagos. This, in turn, encourages tourists to develop greater awareness regarding the necessary conservation efforts, thereby contributing to the preservation of the ecosystem.

By providing more detailed and adult-oriented information, the aim is not only to enhance the tourist experience but also to promote a more conscious and responsible attitude towards the natural environment of the Galapagos.

### 6.4.3 Biologist User

When the user is a biologist, the interface maintains similarities with that of other users, but the distinction lies in the depth of the provided information. In this context, the platform goes beyond offering basic data and encompasses crucial details relevant to substantial biological research. Aspects such as weight, frequented habitats, and taxonomic classification are presented in a detailed manner, providing biologists with a rich source of essential information for specific investigations. This approach is fundamental as it enables biologists to quickly access crucial data on different birds, not only expediting their research but also contributing to monitoring and understanding changes in bird behavior, allowing for better adaptation to changing environmental factors.

The significance of this adaptation extends beyond efficiency in biological research, as it also has a positive impact on biodiversity. By furnishing biologists with detailed information, the platform enables them to swiftly identify deviations in bird behavior, signaling potential challenges or changes in the environment. This responsiveness aids biologists in understanding and proactively addressing threats that could impact birds, thereby promoting the conservation and preservation of biological diversity.

After selecting a user, the application redirects to an interface that allows switching between the front and rear cameras, as shown in Figure 6.6. The rear camera is chosen to focus on an albatross, and the model loads to classify the bird, correctly identifying it as an albatross. At the bottom of the interface, specific information is presented according to the user type selected, and a link called “more information” redirects to additional details about the bird.

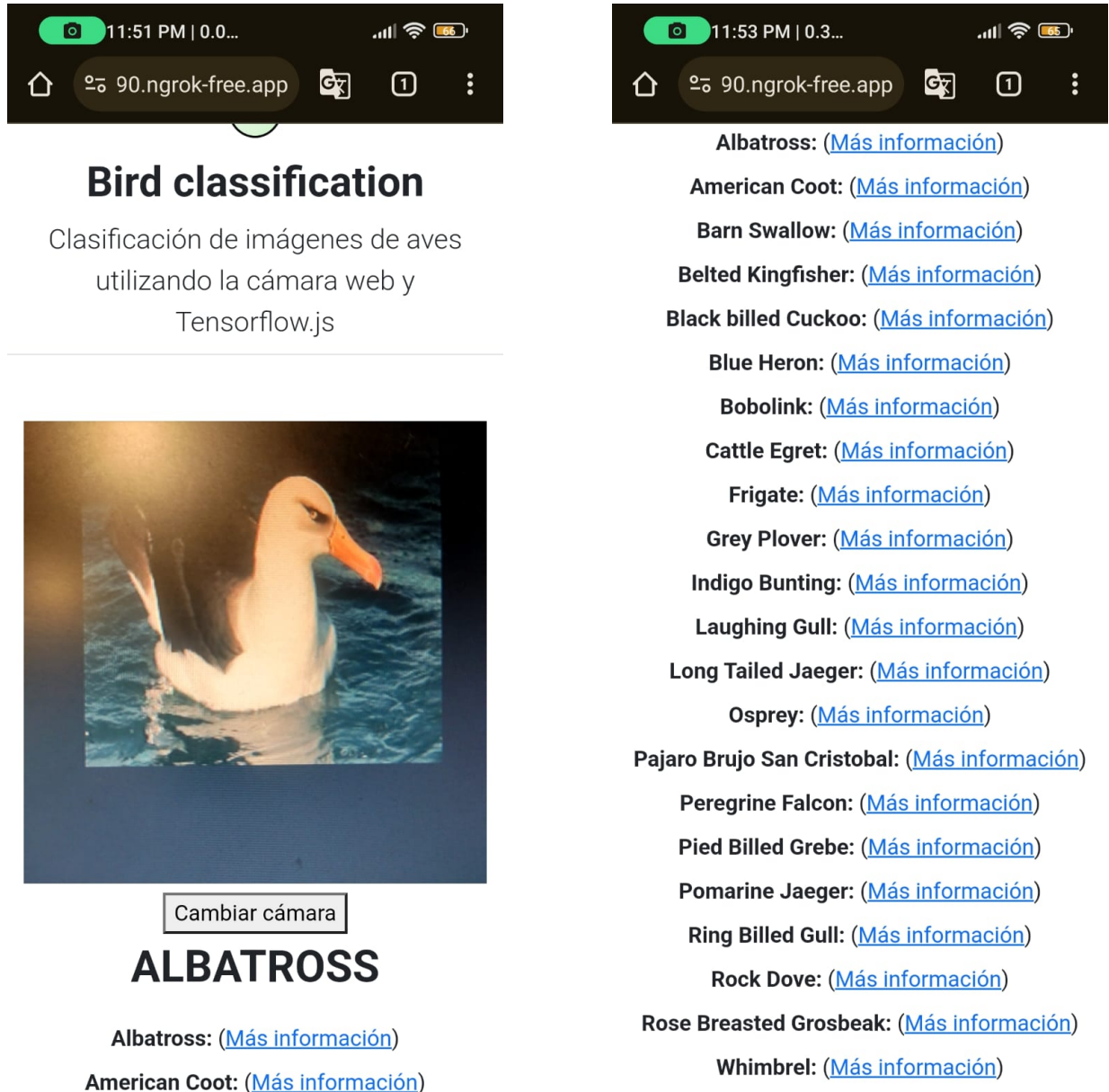


Figure 6.6: Bird Classification

Clicking on the “more information” link opens a window that presents specific details about the bird, tailored to the type of user, as shown in Figure 6.7. If the user is a child, the information will be simple and easy to understand. For a tourist, the information will be a bit more serious but equally accessible. In the case of a biologist, the information will be more detailed and serious, providing essential data such as weight, scientific name, among others, that may be relevant to the biologist’s specific needs regarding the bird.



Figure 6.7: Information

Finally, we go to the main part of the application, which is where the camera is used to classify or identify the type of bird. We go to the bottom of the mobile application, as you can see in Figure 6.8. we can find important data such as the link from which the information was obtained or used to create the dataset, and also so you can visit a page that is a foundation for the conservation of the Galapagos Islands. You can find many ways to help preserve these unique islands.

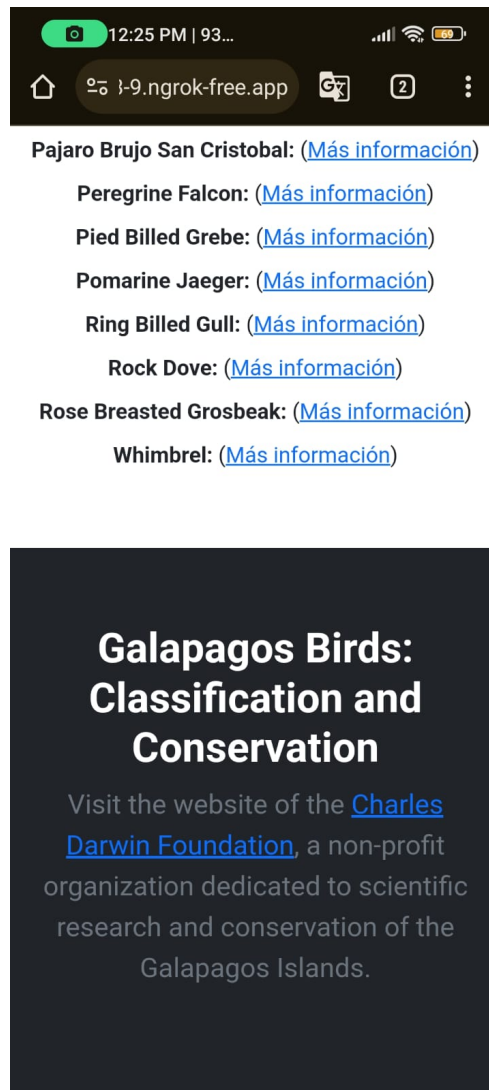


Figure 6.8: Charles Darwin Foundation

## 6.5 System Usability Scale Questionnaire

To validate the application, we relied on the System Usability Scale (SUS), as shown in Figure 6.9. The implementation of these questions is available in a Google Forms survey. A minimum of 12 responses is recommended for a comprehensive evaluation [83]. This assessment was conducted by applying the 10 questions based on the SUS questionnaire, assigning a weighting from 1 to 5 on the Likert scale. Each user first used the application and then proceeded to complete the survey, ensuring that the feedback was more subjective.

## Usabilidad del sistema Galapagos Bird

La base de este cuestionario radica en evaluar la usabilidad del sistema Galapagos bir. Los participantes califican cada afirmación en una escala Likert de 5 puntos, que abarca desde "Totalmente en desacuerdo" hasta "Totalmente de acuerdo". Posteriormente, se lleva a cabo un cálculo para obtener una puntuación general de usabilidad del sistema.

ricardo.farinango@yachaytech.edu.ec [Cambiar de cuenta](#)

No compartido

---

1. Creo que usaría este sistema con frecuencia

1    2    3    4    5

Totalmente en desacuerdo                    Totalmente de acuerdo

---

2. Encontré que el sistema era innecesariamente complejo

1    2    3    4    5

Totalmente en desacuerdo                    Totalmente de acuerdo

---

3. Pensé que el sistema era fácil de usar

Figure 6.9: System Usability Scale

When conducting the survey with randomly selected individuals, specific calculations are performed to assess usability.

The evaluation is done individually for each person using the following formula:

$$x = \sum_{i=1,3,5,7,9} \text{Answer}_i - 5 \quad (6.1)$$

$$y = 25 - \sum_{i=2,4,6,8,10} \text{Answer}_i \quad (6.2)$$

$$\text{Result} = (x + y) \times 2.5 \quad (6.3)$$

This process would be repeated for each participant, allowing for obtaining a measure of perceived usability for each individual based on their responses to the questionnaire.

In the table Table 6.1, the participant responses were recorded. The table consists of 12 rows corresponding to the individuals who participated and 10 columns representing the 10 relevant questionnaire questions. Additionally, there is an extra column indicating the usability percentage generated by applying the formula that calculates usability for each user.

Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	SUS
1	4	1	5	1	3	2	5	2	4	2	82.5
2	5	2	3	2	5	2	5	2	2	2	75
3	2	2	5	2	4	3	3	2	4	2	67.5
4	5	4	5	1	4	1	5	1	5	1	90
5	5	1	5	3	5	1	5	1	5	2	92.5
6	4	4	5	3	5	3	5	1	5	2	77.5
7	3	1	4	2	4	1	5	1	5	1	87.5
8	5	2	5	2	5	2	5	2	4	2	85
9	4	2	5	1	4	2	5	1	5	1	90
10	5	1	4	2	4	1	4	2	5	1	87.5
11	3	2	5	1	5	2	4	2	4	2	80
12	3	3	4	2	4	3	1	5	1	1	47.5

Table 6.1: Responses to the questionnaire

After collecting the results from each participant, we will proceed to sum all the responses and divide them by the total number of participants, which in this case is 12 [83]. This process will allow us to obtain the usability index, which is evaluated on a scale from 0 to 100.

$$\frac{82.5 + 75 + 67.5 + 90 + 92.5 + 77.5 + 87.5 + 85 + 90 + 87.5 + 80 + 47.5}{12} = 80.20$$

By calculating the average of all the responses provided by the users, we observe that the application has reached an "Good" level, as indicated by the score in Figure 2.19.

# Chapter 7

## Conclusions

### 7.1 Conclusions

#### 7.1.1 Achievements and Contributions

Upon concluding this project, the successful development of a mobile application dedicated to the identification of various bird species in the Galapagos Islands has been achieved. This initiative aligns with the general objective of contributing to biodiversity preservation, with a specific focus on protecting avian fauna.

From collecting images of birds in Galapagos to organizing and structurally labeling a diverse dataset, the specific objectives have been fulfilled. Experiments with hyperparameters and training different neural network architectures allowed for the selection of the best classifier for bird classification in this unique environment.

The successful implementation of the mobile application not only signifies the achievement of a specific objective but also provides a practical tool for raising awareness about avian diversity. The application aims not only to facilitate bird identification but also to promote understanding of interactions between different species and raise awareness about the potential negative impacts of introducing non-native species.

Ultimately, this mobile application stands as a valuable contribution to the management and conservation of avian biodiversity in the Galapagos Islands, effectively integrating the diverse elements addressed in the specific objectives of the project.



## 7.1.2 Limitations and Challenges

Throughout the project's development, we encountered several limitations that significantly impacted its progress. Notably, the absence of a pre-existing dataset was a considerable hurdle, necessitating the creation of a dataset from scratch. While essential, this crucial task consumed a substantial amount of time, affecting the project's timeline.

Another critical challenge we faced was the acquisition of resources for model training. The reliance on the free resources the Google Colab platform provided proved inadequate. This limitation led to significant delays during model training, hampering the project's efficiency.

These limitations are a stark reminder of the importance of having sufficient resources and datasets available for projects of this nature. Furthermore, they highlight the necessity of optimizing model training performance to ensure more efficient and timely results.

## 7.2 Future Work

### 7.2.1 Model Performance Improvements

The purpose is to promote the utilization of the dataset. Hence, this information has been shared on the GitHub platform [8] and will be shared on more platforms. This measure will enable other users to leverage the dataset for their projects. The goal is to disseminate knowledge about the Galapagos avifauna and contribute to future research advancement. Additionally, a plan is to update the dataset by adding new images over time. This strategy aims to enhance the quality of the dataset and, consequently, the outcomes of future projects.

### 7.2.2 Expansion of the Mobile Application

The final version of the mobile application developed in this thesis does not mark the end of possible improvements. The application is intended to continue evolving, both in terms of content and improving the user interface. In addition, it plans to expand the application to support different operating systems. This approach reflects a continued commitment to improve and adapt as new opportunities and technologies arise.

# Bibliography

- [1] S.-H. Wang and Y.-D. Zhang, “Densenet-201-based deep neural network with composite learning factor and precomputation for multiple sclerosis classification,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2s, pp. 1–19, 2020.
- [2] X. Chen, X. Pu, Z. Chen, L. Li, K.-N. Zhao, H. Liu, and H. Zhu, “Application of efficientnet-b0 and gru-based deep learning on classifying the colposcopy diagnosis of precancerous cervical lesions,” *Cancer Medicine*, vol. 12, no. 7, pp. 8690–8699, 2023.
- [3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [5] I. Z. Mukti and D. Biswas, “Transfer learning based plant diseases detection using resnet50,” in *2019 4th International conference on electrical information and communication technology (EICT)*. IEEE, 2019, pp. 1–6.
- [6] H. Qassim, A. Verma, and D. Feinzimer, “Compressed residual-vgg16 cnn model for big data places image recognition,” in *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*. IEEE, 2018, pp. 169–175.
- [7] M. Bansal, M. Kumar, M. Sachdeva, and A. Mittal, “Transfer learning for image classification using vgg19: Caltech-101 image data set,” *Journal of ambient intelligence and humanized computing*, pp. 1–12, 2021.

- [8] R. Joseph, “Galapagosbird: A repository for avian enthusiasts,” <https://github.com/RicardoJoseph1307/GalapagosBird.git>, 2024.
- [9] J.-W. Lin, “Artificial neural network related to biological neuron network: a review,” *Advanced Studies in Medical Sciences*, vol. 5, no. 1, pp. 55–62, 2017.
- [10] J. P. A. Leon, F. J. Rico-Novella, and J. Luis, “Predictive traffic control and differentiation on smart grid neighborhood area networks,” *IEEE access*, vol. 8, pp. 216 805–216 821, 2020.
- [11] L. Kang, J. Kumar, P. Ye, Y. Li, and D. Doermann, “Convolutional neural networks for document image classification,” in *2014 22nd international conference on pattern recognition*. IEEE, 2014, pp. 3168–3172.
- [12] S. Ramaneswaran, K. Srinivasan, P. D. R. Vincent, and C.-Y. Chang, “Hybrid inception v3 xgboost model for acute lymphoblastic leukemia classification,” *Computational and Mathematical Methods in Medicine*, vol. 2021, pp. 1–10, 2021.
- [13] T. B. Shahi, C. Sitaula, A. Neupane, and W. Guo, “Fruit classification using attention-based mobilenetv2 for industrial applications,” *Plos one*, vol. 17, no. 2, p. e0264586, 2022.
- [14] L. Wen, X. Li, and L. Gao, “A transfer convolutional neural network for fault diagnosis based on resnet-50,” *Neural Computing and Applications*, vol. 32, pp. 6111–6124, 2020.
- [15] A. Hashmi, *Machine Learning with TensorFlow.js*. GitBook, 2023. [Online]. Available: <https://adnanhashmi.gitbook.io/machine-learning-with-tensorflow-js/>
- [16] L. Lemus Cárdenas, “Enhancement of vehicular ad hoc networks using machine learning-based prediction methods,” 2020.
- [17] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, “A survey on modern trainable activation functions,” *Neural Networks*, vol. 138, pp. 14–32, 2021.
- [18] J. Brooke, “Sus: a retrospective,” *Journal of usability studies*, vol. 8, no. 2, pp. 29–40, 2013.

- [19] W.-H. Cheah, N. Mat Jusoh, M. M. T. Aung, A. Ab Ghani, and H. Mohd Amin Re-buan, “Mobile technology in medicine: Development and validation of an adapted system usability scale (sus) questionnaire and modified technology acceptance model (tam) to evaluate user experience and acceptability of a mobile application in mri safety screening,” *Indian Journal of Radiology and Imaging*, vol. 33, no. 01, pp. 036–045, 2022.
- [20] S. Kleindorfer, B. Fessler, K. Peters, and D. Anchundia, “Guía de campo. aves terrestres residentes de galápagos,” 2019.
- [21] “Galapagos species checklist,” <https://www.darwinfoundation.org/en/datazone/checklist/>, Fundación Charles Darwin, Año actual, recuperado de la base de datos de la Fundación Charles Darwin.
- [22] Y. Lu, “Artificial intelligence: a survey on evolution, models, applications and future trends,” *Journal of Management Analytics*, vol. 6, no. 1, pp. 1–29, 2019.
- [23] B. A. Carrera Castro, “Diagnóstico de la presencia de la especie invasiva wasmannia auropunctata hormiga de fuego, a partir de información bibliográfica 2014–2018, en las islas galápagos.” B.S. thesis, La Libertad: Universidad Estatal Península de Santa Elena, 2021., 2021.
- [24] K. V. Ascencio Lárraga *et al.*, “Interacciones bióticas entre especies invasoras en las islas galápagos,” B.S. thesis, Quito, 2018.
- [25] A. Mathis and J. Rose, “Balancing tourism, conservation, and development: a political ecology of ecotourism on the galapagos islands,” *Journal of Ecotourism*, vol. 15, no. 1, pp. 64–77, 2016.
- [26] L. Brewington, “The politics of invasion: defining and defending the natural, native and legal in the galapagos islands of ecuador,” Ph.D. dissertation, The University of North Carolina at Chapel Hill, 2011.
- [27] B. Fessler, D. Anchundia, J. Carrión, A. Cimadom, J. Cotin, F. Cunninghame, M. Dvorak, D. Mosquera, E. Nemeth, C. Sevilla *et al.*, “Aves terrestres de galápagos (paser-

- iformes, cuclillos y palomas): estado, amenazas y brechas de conocimiento,” *Informe Galápagos 2015–2016*, pp. 151–162, 2017.
- [28] L. Rouhiainen, “Inteligencia artificial,” *Madrid: Alienta Editorial*, pp. 20–21, 2018.
- [29] R. C. Schank, “Where’s the ai?” *AI magazine*, vol. 12, no. 4, pp. 38–38, 1991.
- [30] A. M. Turing, *Computing machinery and intelligence*. Springer, 2009.
- [31] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, “A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955,” *AI magazine*, vol. 27, no. 4, pp. 12–12, 2006.
- [32] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [33] S. E. Dreyfus, “Artificial neural networks, back propagation, and the kelley-bryson gradient procedure,” *Journal of guidance, control, and dynamics*, vol. 13, no. 5, pp. 926–928, 1990.
- [34] F.-H. Hsu, *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2002.
- [35] A. B. Lowndes, “Deep learning with gpu technology for image & feature recognition,” Ph.D. dissertation, Tesis de Grado]. University of Leeds, 2015.
- [36] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, “Google deepmind’s alphago: operations research’s unheralded role in the path-breaking achievement.” *Or/MS Today*, vol. 43, no. 5, pp. 24–30, 2016.
- [37] F. Chen and D. Zhao, “Computing technology in autonomous vehicle,” in *Advanced Driver Assistance Systems and Autonomous Vehicles: From Fundamentals to Applications*. Springer, 2022, pp. 49–114.
- [38] I. N. Da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, S. F. dos Reis Alves, I. N. da Silva, D. Hernane Spatti, R. Andrade Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, *Artificial neural network architectures and training processes*. Springer, 2017.

- [39] P. Wang, E. Fan, and P. Wang, “Comparative analysis of image classification algorithms based on traditional machine learning and deep learning,” *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.
- [40] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, “Medical image classification with convolutional neural network,” in *2014 13th international conference on control automation robotics & vision (ICARCV)*. IEEE, 2014, pp. 844–848.
- [41] M. A. Chandra and S. Bedi, “Survey on svm and their application in image classification,” *International Journal of Information Technology*, vol. 13, pp. 1–11, 2021.
- [42] M. Shaha and M. Pawar, “Transfer learning for image classification,” in *2018 second international conference on electronics, communication and aerospace technology (ICECA)*. IEEE, 2018, pp. 656–660.
- [43] Q. Li and X. Wang, “Image classification based on sift and svm,” in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*. IEEE, 2018, pp. 762–765.
- [44] D. Xue, X. Zhou, C. Li, Y. Yao, M. M. Rahaman, J. Zhang, H. Chen, J. Zhang, S. Qi, and H. Sun, “An application of transfer learning and ensemble learning techniques for cervical histopathology image classification,” *IEEE Access*, vol. 8, pp. 104 603–104 618, 2020.
- [45] I. T. Young, J. J. Gerbrands, and L. J. Van Vliet, *Fundamentals of image processing*. Delft University of Technology Delft, 1998, vol. 841.
- [46] M. M. Petrou and C. Petrou, *Image processing: the fundamentals*. John Wiley & Sons, 2010.
- [47] P. Marcelino, “Transfer learning from pre-trained models,” *Towards data science*, vol. 10, p. 23, 2018.
- [48] V. Viitaniemi and J. Laaksonen, “Techniques for image classification, object detection and object segmentation,” in *Visual Information Systems. Web-Based Visual Informa-*

- tion Search and Management: 10th International Conference, VISUAL 2008, Salerno, Italy, September 11-12, 2008. Proceedings 10.* Springer, 2008, pp. 231–234.
- [49] W. Dai, Y. Chen, G.-R. Xue, Q. Yang, and Y. Yu, “Translated learning: Transfer learning across different feature spaces,” *Advances in neural information processing systems*, vol. 21, 2008.
- [50] T. Lu, B. Han, L. Chen, F. Yu, and C. Xue, “A generic intelligent tomato classification system for practical applications using densenet-201 with transfer learning,” *Scientific Reports*, vol. 11, no. 1, p. 15824, 2021.
- [51] A. E. Tio, “Face shape classification using inception v3,” *arXiv preprint arXiv:1911.07916*, 2019.
- [52] Y. He, X. Zhang, Z. Zhang, and H. Fang, “Automated detection of boundary line in paddy field using mobilev2-unet and ransac,” *Computers and Electronics in Agriculture*, vol. 194, p. 106697, 2022.
- [53] J. Jaworek-Korjakowska, P. Kleczek, and M. Gorgon, “Melanoma thickness prediction based on convolutional neural network with vgg-19 model transfer learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.
- [54] T. Maekawa, T. Hara, and S. Nishio, “Image classification for mobile web browsing,” in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 43–52.
- [55] C. Gerard and C. Gerard, “Tensorflow. js,” *Practical Machine Learning in JavaScript: TensorFlow. js for Web Developers*, pp. 25–43, 2021.
- [56] P. Baldi and P. J. Sadowski, “Understanding dropout,” *Advances in neural information processing systems*, vol. 26, 2013.
- [57] L. Prechelt, “Early stopping-but when?” in *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69.

- [58] B. Ding, H. Qian, and J. Zhou, “Activation functions and their characteristics in deep neural networks,” in *2018 Chinese control and decision conference (CCDC)*. IEEE, 2018, pp. 1836–1841.
- [59] O. Sharma, “Deep challenges associated with deep learning,” in *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 2019, pp. 72–75.
- [60] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for neural networks for image processing,” *arXiv preprint arXiv:1511.08861*, 2015.
- [61] D. A. Wiedenfeld and G. Jiménez-Uzcátegui, “Critical problems for bird conservation in the galápagos islands,” *Cotinga*, vol. 29, pp. 22–27, 2008.
- [62] J. Martinsson, “Bird species identification using convolutional neural networks,” 2017.
- [63] A. Marini, J. Facon, and A. L. Koerich, “Bird species classification based on color features,” in *2013 IEEE international conference on systems, man, and cybernetics*. IEEE, 2013, pp. 4336–4341.
- [64] A. Kumar and S. D. Das, “Bird species classification using transfer learning with multistage training,” in *Computer Vision Applications: Third Workshop, WCVA 2018, Held in Conjunction with ICVGIP 2018, Hyderabad, India, December 18, 2018, Revised Selected Papers 3*. Springer, 2019, pp. 28–38.
- [65] J. Niemi and J. T. Tanttú, “Deep learning case study for automatic bird identification,” *Applied sciences*, vol. 8, no. 11, p. 2089, 2018.
- [66] G. Gupta, M. Kshirsagar, M. Zhong, S. Gholami, and J. L. Ferres, “Comparing recurrent convolutional neural networks for large scale bird species classification,” *Scientific reports*, vol. 11, no. 1, p. 17085, 2021.
- [67] J. J. Bird, C. M. Barnes, L. J. Manso, A. Ekárt, and D. R. Faria, “Fruit quality and defect image classification with conditional gan data augmentation,” *Scientia Horticulturae*, vol. 293, p. 110684, 2022.



- [68] M. Raghuram, N. R. Chavan, R. Belur, and S. G. Koolagudi, “Bird classification based on their sound patterns,” *International journal of speech technology*, vol. 19, pp. 791–804, 2016.
- [69] M. Alswaitti, L. Zihao, W. Alomoush, A. Alrosan, and K. Alissa, “Effective classification of birds’ species based on transfer learning,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 4, pp. 4172–4184, 2022.
- [70] Y. Kumar, S. Gupta, and W. Singh, “A novel deep transfer learning models for recognition of birds sounds in different environment,” *Soft Computing*, pp. 1–21, 2022.
- [71] P. Wu, G. Wu, X. Wu, X. Yi, and N. Xiong, “Birds classification based on deep transfer learning,” in *Smart Computing and Communication: 5th International Conference, SmartCom 2020, Paris, France, December 29–31, 2020, Proceedings 5*. Springer, 2021, pp. 173–183.
- [72] A. A. Biswas, M. M. Rahman, A. Rajbongshi, and A. Majumder, “Recognition of local birds using different cnn architectures with transfer learning,” in *2021 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE, 2021, pp. 1–6.
- [73] Kaggle, “25 indian bird species with 22.6k images — kaggle,” 2021. [Online]. Available: <https://www.kaggle.com/datasets/arjunbasandrai/25-indian-bird-species-with-226k-images>
- [74] —, “100 bird species — kaggle,” 2021. [Online]. Available: <https://www.kaggle.com/datasets/gpiosenska/100-bird-species>
- [75] —, “200 bird species with 11,788 images — kaggle,” 2021. [Online]. Available: <https://www.kaggle.com/discussions/general/126169>
- [76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

- [77] S. Orozco-Arias, J. S. Piña, R. Tabares-Soto, L. F. Castillo-Ossa, R. Guyot, and G. Isaza, “Measuring performance metrics of machine learning algorithms for detecting and classifying transposable elements,” *Processes*, vol. 8, no. 6, p. 638, 2020.
- [78] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, “Tensorflow distributions,” *arXiv preprint arXiv:1711.10604*, 2017.
- [79] M. S. Tanveer, M. U. K. Khan, and C.-M. Kyung, “Fine-tuning darts for image classification,” in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 4789–4796.
- [80] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, “An ensemble of fine-tuned convolutional neural networks for medical image classification,” *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 31–40, 2016.
- [81] K. Praghash, V. S. Eswar, J. Y. Roy, A. Alagarsamy, and S. Arunmetha, “Tunnel based intra network controller using ngrok framework for smart cities,” in *2021 5th international conference on electronics, communication and aerospace technology (ICECA)*. IEEE, 2021, pp. 39–43.
- [82] P. Barford and M. Crovella, “Generating representative web workloads for network and server performance evaluation,” in *Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, 1998, pp. 151–160.
- [83] S. McLellan, A. Muddimer, and S. C. Peres, “The effect of experience on system usability scale ratings,” *Journal of usability studies*, vol. 7, no. 2, pp. 56–67, 2012.