

STATE OIL COMPANY OF AZERBAIJAN REPUBLIC
BAKU HIGHER OIL SCHOOL

Faculty of Engineering

Process Automation Engineering Department

Speciality: 050628 – Process Automation Engineering

Group: PAM 2018

GRADUATION WORK

Title: Bird Classification with EfficientNet Architecture

Student: Bahruz Huseynov

Supervisor: Ramil Shukurov

Head of Department: Associate Professor Naila Allakhverdiyeva

Baku – 2023

Acknowledgement

I would like to express my deep gratitude to Ramil Shukurov for his valuable contributions throughout the completion of the diploma work. He provided me with a lot of information in order to successfully complete the task and always assisted me to search and gather necessary data.

Furthermore, I could easily share my opinions to cope with the problem and assume that it could be really challenging for me without his feedback and guidance as he made me deeply analyze the working principle of the models and architecture which played an essential role in the solution of the problem.

Finally, I appreciate the efforts and assistance of my supervisor as he frequently motivated me and gave exact directions during this academic journey.

Thank you,

Bahrüz Huseynov.

Abstract

Information about birds is essential in terms of their protection and advancement in the science. In some situations, differing a specific bird from other species can be challenging, however the problem can be solved by the computer vision. Thus, I applied two versions, both transfer-learning and architecture built from scratch, of EfficientNet-B0 and EfficientNet-B1 on the bird classification problem (multi-classification problem with 200 classes). EfficientNet is one of the effective models that became popular in recent years as its high accuracy on ImageNet. Generally, the superior purpose of the experiment was to analyze how EfficientNet can work on pure RGB pixels of the images with low quality (various shapes of images, blurry pictures, birds hidden behind objects, small birds in the large images) and even the fine-tuned model was used to reach the best accuracy on the dataset. The dataset used in the problem is another version of the famous dataset called “CUB-200-2011” and the highest accuracy on the original dataset at 78.48% is obtained by fine-tuning EfficientNet-B1, which was slightly more than the result reached by fine-tuned EfficientNet-B0 (76.96%) and followingly cropping the resolution of the bird and removing other parts increased the test accuracy to 83.8% and 84.64% for B0 and B1 models, respectively. To confirm that the mainly used dataset has decent quality, another dataset in which all images have the shape of 224x224 with 20 classes is used and even architecture constructed from scratch resulted in 77.31% accuracy. Moreover, the best performance of the pre-trained EfficientNet-B0 on a new dataset was 95.83%, which can originate from better quality, the standard shape of images and standard position of the birds in images and the number of classes. So, the experiment concludes that EfficientNet has an effective classification ability but it is dependent on the nature of the dataset, pre-processing, optimization algorithm and the number of labels.

Keywords: CNN, EfficientNet, Bird classification, Image processing, Transfer learning

Referat

Quşlar haqqında məlumat onların mühafizəsi və elmdə irəliləyişi baxımından vacibdir. Bəzən xüsusi quşu digər növlərdən fərqləndirmək çətin ola bilər, lakin problemi kompüter görməsi ilə həll etmək olar. Beləliklə, mən EfficientNet-B0 və EfficientNet-B1 modellərinin həm köçürmə-öyrənmə, həm də sıfırdan qurulmuş arxitektura variantını quşların təsnifatı probleminə (verilənlər bazası 200 sinifdən ibarətdir) tətbiq etdim. EfficientNet son illərdə ImageNet üzərindəki yüksək dəqiqliyi ilə məşhurlaşan effektiv modellərdən biridir. Eksperimentin əsas məqsədi EfficientNet arxitekturasının aşağı keyfiyyətli şəkillərin (müxtəlif formalı və bulanıq şəkillər, obyektlərin arxasında gizlənmiş quşlar, böyük ölçülü şəkillər) və RGB pikselləri üzərində necə işləyə biləcəyini təhlil etmək idi. Problemdə məşhur “CUB-200-2011” verilənlər toplusunun başqa versiyası istifadə olunur. Orijinal verilənlər bazasında ən yüksək dəqiqlik 78,48% EfficientNet-B1 modelinin dəqiq tənzimlənməsi ilə əldə edilir ki, bu da tənzimlənmiş EfficientNet-B0 (76,96%) ilə əldə edilən nəticədən bir qədər çox idi. Daha sonra şəkildə quşun pozisyasının kəsilib çıxarılması ilə B0 və B1 modelləri üçün test dəqiqliyini müvafiq olaraq 83,8% və 84,64%-ə qədər artdı. İstifadə olunan verilənlər toplusunun üstün keyfiyyətə malik olduğunu təsdiqləmək üçün bütün şəkillərin 20 sinifli 224x224 formasına malik olduğu başqa bir verilənlər bazasından istifadə olunur və hətta sıfırdan qurulmuş arxitektura 77,31% dəqiqliklə nəticələnib. EfficientNet-B0-nin yeni verilənlər bazasında ən yaxşı performans 95,83% idi ki, bu da daha keyfiyyətli şəkillərdən, təsvirlərin formasından, şəkillərdəki quşların mövqeyindən və siniflərin sayından irəli gəlir. Təcrübə EfficientNet arxitekturasının effektiv təsnifat qabiliyyətinə malik olduğu qənaətinə gəlir, lakin bu, verilənlər bazasının təbiətindən, əvvəlcədən emaldan, optimallaşdırma alqoritmindən və etiketlərin sayından asılıdır.

Açar sözlər: CNN, EfficientNet, Quşların təsnifatı, Şəkillərin işlənməsi, Transfer öyrənməsi

Contents

Acknowledgement	ii
Abstract.....	iii
Referat.....	iv
Figures.....	vi
Tables	vii
List of Equations	viii
List of Abbreviations.....	ix
Introduction	1
1. Literature Review.....	3
1.1. Rescaling model analysis and minor limitations	3
1.2. Related works	6
1.3. Motivation	8
2. Methodology	10
2.1. Data preprocessing.....	10
2.2. Hyperparameters	12
2.3. Deep Learning model.....	13
2.4. Padding, stride and pooling.....	15
2.5. Activation functions	17
2.5.1. Sigmoid.....	18
2.5.2. Rectified Linear Unit (ReLU).....	18
2.5.3. Sigmoid Linear Unit (SiLU).	19
2.6. Batch normalization.....	20
2.7. Dropout.	21
2.8. EfficientNet architecture.	22
2.9. Transfer learning.....	25
2.10. Loss function and optimizers.....	26
3. Discussion of the results	29
3.1. Training environment, parameters and framework	29
3.2. Evaluation metrics.....	29
3.3. Results and analysis of the experiment.....	31
Conclusion.....	42
References	43

Declaration.....	46
-------------------------	-----------

Figures

Figure 1.1 Model Scaling	4
Figure 1.2 ImageNet accuracy versus Deep Learning architecture	5
Figure 2.1 Stratified Splitting.....	12
Figure 2.2:a) Pooling types b) Convolution.....	17
Figure 2.3 Activaton functions.....	20
Figure 2.4 EfficientNet-B0.....	24
Figure 2.5 MBConv block.....	24
Figure 2.6 Squeeze-and-Excitation	24
Figure 2.7 Final layers.....	24
Figure 2.8 EfficientNet-B1 architecture.....	25
Figure 3.1 Confusion matrix for multi-classification.....	31
Figure 3.2 Sample images	33
Figure 3.3 EfficientNet-B0 (Fine-tuning) loss and accuracy graph.....	35
Figure 3.4 EfficientNet-B1 (Fine-tuning) loss and accuracy graph.....	35
Figure 3.5 How dataset 1.2 has been created	36
Figure 3.6 EfficientNet-B0 (Fine-tuning) loss and accuracy graph on Dataset 1.2	37
Figure 3.7 EfficientNet-B1 (Fine-tuning) loss and accuracy graph on Dataset 1.2	37
Figure 3.8 EfficientNet-B0 from scratch with SGD (Momentum).....	40
Figure 3.9 Pre-trained EfficientNet-B0 with Adam optimizer	40

Tables

Table 2.1 Datasets and their details.....	10
Table 2.2 Dropout rate and input picture resolution for the EfficientNet type.....	21
Table 3.1 Results of the models applied on the main dataset	32
Table 3.2 Results of the models applied on Dataset 2	32
Table 3.3 Some wrong predictions of EfficientNet-B1 (84.64%)	38

List of Equations

Equation 2.1	Output width after convolution.....	14
Equation 2.2	Output height after convolution.....	15
Equation 2.3	Sigmoid Function.....	18
Equation 2.4	ReLU Function.....	18
Equation 2.5	SiLU Function.....	19
Equation 2.6	Cross-Entropy Loss Function	26
Equation 2.7	Update Rule in SGD	27
Equation 2.8	Momentum parameter in SGD with Momentum.....	27
Equation 2.9	Update Rule in SGD with Momentum.....	27
Equation 2.10	Adam Update for First Moment Estimate.....	28
Equation 2.11	Adam Update for Second Moment Estimate.....	28
Equation 2.12	Bias-Corrected First Moment Estimate.....	28
Equation 2.13	Bias-Corrected Second Moment Estimate.....	28
Equation 2.14	Update Rule in Adam optimizer.....	28
Equation 3.1	Accuracy.....	31

List of Abbreviations

CNN	Convolutional Neural Network
SiLU	Sigmoid Linear Units
ReLU	Rectified Linear Units
PCB	Printer Circuit Board
SGD	Stochastic Gradient Descent

Introduction

In the current world, the classification of bird photos plays a vital role, especially in different fields including ecology, geography, and conservation. To better understand bird populations and habits, follow migratory patterns, and keep track of the health of bird populations, researchers can benefit greatly from images of birds. The bird classification is also extremely advantageous for researchers to acquire data over a large geographic region without personally visiting each area [1]. Additionally, it may be utilized as a teaching tool to help the general public and bird enthusiasts learn more about the local species and how they can support conservation efforts. Considering the mentioned nuances, the identification of birds based on Deep Learning or automated techniques is the perfect method in terms of protection of the endangered animals, and assessment of the quantity and diversity of the various species in several locations.

Sorting birds into categories is a crucial point in terms of ornithologists as they deal with various patterns of the bird [1]. Sometimes, a bird possesses so complicated characteristics that it becomes a headache for the scientists and researchers of the related field to differentiate one bird from another. By using computer vision, essential bird features like their colors, length of wings, type and look of the bark, color of the special parts like throat, and tail, can be extracted to understand the bird species. Briefly, the categorization of birds is a crucial tool for academics, scientists, and environmentalists. Today, it has been more straightforward to differentiate a special kind of bird from other species with the help of sophisticated Deep Learning techniques and by using them, researchers can easily analyze large datasets of bird images and identify special nuances.

Changing the focus point to city science, it is also the crucial field of science dealing with birds and their classification. People in the field are called citizen scientists and they are amateur researchers who have an interest in a particular field, such as birds, insects, plants and others. They can also help define and naturally expand the taxonomic structure

of a field to better reflect the interests of its actual users [2]. The categorization of birds soon leads to a reduction in the effort done by citizen scientists.

Deep Learning has evolved throughout time to produce more accurate findings and has employed a variety of techniques to address categorization issues. One of the contemporary models, EfficientNet, achieved state-of-the-art performance on a variety of picture classification tasks. VGGNet, ResNet, Inception, and MobileNet are a few further examples of these models, and they differ from EfficientNet in how they scale the model's depth, breadth, and resolution [3]. EfficientNet also employs a revolutionary method for scaling all three components together in a balanced way, allowing for more effective use of computational resources while maintaining high accuracy and short training times. This is the fundamental distinction between EfficientNet and other systems. Architecture is especially helpful for resource-constrained applications, such as those present in mobile devices and embedded systems. Moreover, this scaling approach has the ability to produce smaller, faster, and more accurate results than earlier models. So, it is one of the most ordered deep learning structures to be constructed recently, and it is becoming more well-known as a result of its outstanding outcomes. In the next part, previous works and methods related to the EfficientNet architecture will be discussed.

1. Literature Review

1.1. Rescaling model analysis and minor limitations

EfficientNet has emerged as a cutting-edge CNN architecture that has garnered significant interest within the computer vision community. The articles surrounding EfficientNet predominantly concentrate on its remarkable achievements in diverse image recognition tasks, all while upholding computational efficiency. Researchers have discovered that EfficientNet surpasses previous architectures in accuracy while simultaneously reducing the number of parameters and computational resources required, presenting an appealing option for real-world applications that prioritize efficiency. The architecture itself is founded on a compound scaling technique that uniformly adjusts the network's depth, width, and resolution, resulting in enhanced performance across a wide range of image scales [4]. The outstanding accomplishments of EfficientNet have firmly established it as a leading solution for image classification endeavors, thereby facilitating advancements in domains such as object detection, semantic segmentation, and visual search. Consequently, articles about architecture and its practical application appeared.

The core article is “EfficientNet: Rethinking Model Scaling for CNNs” proposed by Mingxing Tan and Quoc V. Le and it aims to a novel approach for scaling convolutional neural networks (CNNs) to achieve better performance and efficiency by systematically scaling the model's depth, width, and resolution [5]. To get into details, compound scaling and formulas, model performance, its applications on ImageNet and other types of information are key points mentioned in the article. Thus, the authors of the paper argue that simply increasing depth, width, or resolution independently may not lead to optimal performance. Instead, a balanced scaling of all three dimensions is essential. Additionally, the core idea related to the EfficientNet architecture is also recorded in the paper that it consists of a baseline network, called EfficientNet-B0, which is then scaled

up to create larger models such as EfficientNet-B1, EfficientNet-B2, and so on [4, 3]. Figure 1.1 below visually displays how the scaling is done [5].

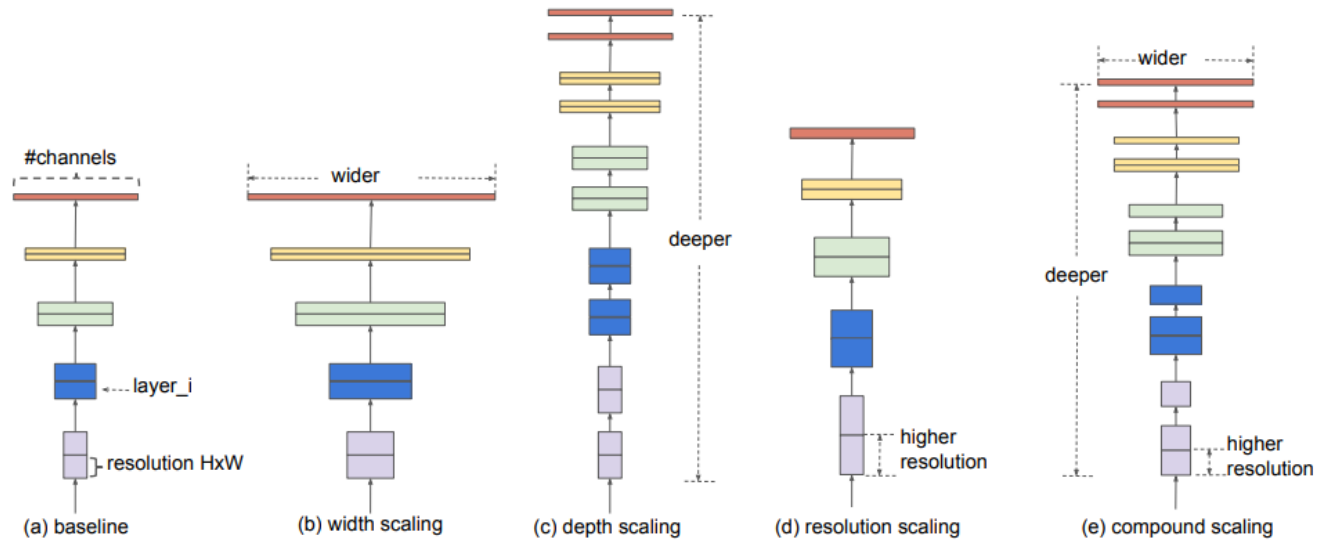


Figure 1.1 Model Scaling

Specifically, EfficientNet-B7 attains an unprecedented top-1 accuracy of 84.3%, surpassing all previous models [5]. Remarkably, it achieves this accuracy while being significantly smaller and faster than GPipe, with a reduction in size by a factor of 8.4 and an improvement in speed by a factor of 6.1. Similarly, EfficientNet-B1 exhibits impressive performance, being 7.6x smaller and 5.7x speedier than ResNet-152. Figure 1.2 shows the accuracy of various models on ImageNet [5]. ImageNet contains a lot of labeled pictures from a wide range of classes, including animals, objects, scenes and more. Moreover, it was created for the advancement of image recognition algorithms and you can see how EfficientNet results in high accuracy compared to other models. EfficientNet models bring certain limitations, including the training complexity, low test accuracy or high variance on smaller datasets and another type of problems. Firstly, analyzing the training complexity of EfficientNet is a challenging issue due to memory requirements and time manipulation. The model is effective for larger datasets as training on extensive datasets with high-resolution images demands substantial computational resources [5, 6].

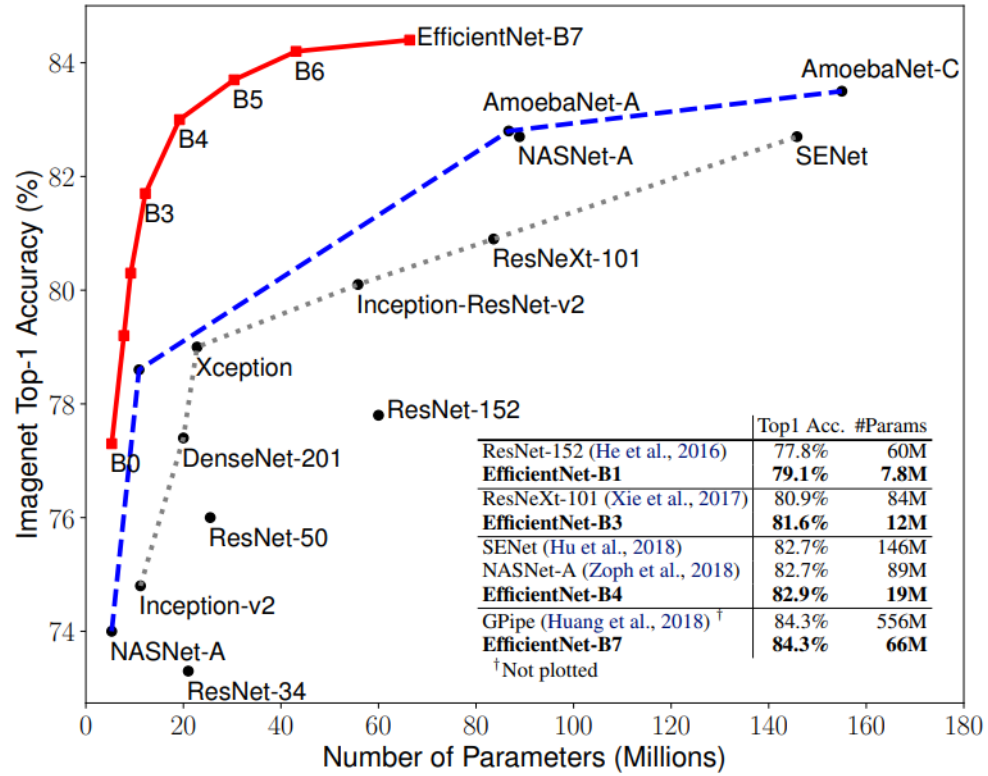


Figure 1.2 ImageNet accuracy versus Deep Learning architecture

Moreover, the memory requirements of architecture increase as their scale grows and this increase causes challenges in deployment procedures with limited memory, such as mobile or edge devices, as well. On the other side, fine-tuning EfficientNet on smaller datasets can achieve optimal performance both in the training procedure and testing. However, important techniques like hyperparameter tuning together with additional data augmentation should be taken into account for the generalization of the model to the dataset. The article above provides a limited explanation of the scaling formula of EfficientNet architecture which is used to determine the best values for scaling the model's depth, width, and resolution. A deeper understanding of the theory requires further research and analysis of the scaling formula.

Finally, EfficientNet's performance relies on several factors despite its impressive results in general image classification. Variability of the model across different task is one of the factors to achieve maximum performance in specific tasks. Depending on the case, fine-

tuning or customizing the architecture can be necessary. Lastly, EfficientNet's complex architecture poses challenges in terms of interpretability. Understanding the learned representations and decision-making processes within the network remains an ongoing challenge in the field of deep learning [3]. It is worth noting that subsequent studies may have addressed these restrictions as the research surrounding EfficientNet continues to evolve [7].

1.2. Related works

The purpose of this section is to analyze other articles attributed to EfficientNet architecture and similar solutions in classification problems.

To help the character recognition, students of Keimyung University worked on the classification of characters based on the images on the board with the help EfficientNet architecture. This research paper extensively acquired and analyzed a significant volume of character data about PCB components. Furthermore, data augmentation techniques were employed, taking into account the severely imbalanced nature of the databases. The most optimal model was chosen by evaluating EfficientNet models from B0 to B7, which were trained using a grid-sampled dataset consisting of 8000 images per class [8]. The authors also introduced a methodology for character recognition that involved constructing a robust dataset capable of accommodating diverse fonts and environmental variations, achieved through the utilization of a substantial amount of data. Additionally, they proposed two novel approaches: a corset for evaluating the effectiveness of deep learning models and an n-pick sampling-based base set that can adapt to continuously expanding datasets. The original data combined with the EfficientNet-B0 architecture achieved an accuracy of 97.74%. Additionally, the suggested technique significantly improved the accuracy to 98.274% when using an 8000-image-per-class coresets. Notably, the accuracy further increased to a little bit over 98.9% for the main dataset, which consisted of merely 1900 pictures per class. Considering the imbalanced dataset, the

accuracy is high, but the position of the character pixels in the image is various than that for birds. That is why getting higher efficiency for the model for birds can be challenging as the birds have been shot in several situations (on trees, flying, over the sea and so on).

Another work done by Francis Jesmar et al. is related to the improvement of disease detection which is a needy project for the world of medicine. The problem is a binary classification that is focused on an empirical evaluation of the recent B0 model based on its efficiency in detecting and diagnosing infections spread by malaria bacteria in the blood. Unlike other approaches, the work concentrated on assessing the EfficientNet-B0 primary model for extracting important patterns by minimizing image transformations in our sample preparation by transfer learning. Subsequently, the fine-tuned model also was used on the proposed layers and re-trained on the collected dataset and finally, the highest overall accuracy achieved was 94.70% after fifty epochs. The article concludes that when properly fine-tuned, the most recent B0 type of EfficientNet architecture may generate extremely accurate solutions in deep learning problems for recognizing whether there are malaria parasites in blood or not without the usage of extensive techniques such as pre-processing or data augmentation of images [3]. On the other hand, in our problem, fine-tuning also will be used to access better performance, however, it is a multi-classification problem rather than the work mentioned above and this factor together with the quality of a dataset can influence the accuracy in an unbelievable scale.

In 2022, students of Southwest University in China suggested a method for effectively identifying apple leaf diseases to promote smart agriculture, decrease pesticide usage, and improve the quality of apple fruit. To address this gap, a CNN called EfficientNet-MG is introduced and its main properties are its accuracy, lightweight nature, and robustness. It is the enhanced version of the EfficientNet network and experimental results demonstrate that EfficientNet-MG outperforms five classical CNN models by achieving a higher accuracy of 99.11% while utilizing fewer parameters [9]. These findings establish the

competitive advantage of EfficientNet-MG in apple leaf problem identification, affirming its efficacy for practical applications in the field.

Moreover, the crack detection problem in concrete samples is also one of the recent experiments attributed to EfficientNet-B0 architecture. The work has been done by Chao Su and Wenjun Wang and their research focuses on enhancing the EfficientNetB0 model that was used to determine whether there are cracks on the surface of concrete or not through the utilization of transfer learning. To evaluate the model's generalization capability, crack images from various locations are used in the testing phase. Comparative analysis is accomplished based on the comparison of the results or predictions of the model with several models, such as DenseNet201 and InceptionV3. So, the results demonstrate that the model significantly causes a decrease in the number of parameters. However, it reaches a high accuracy rate of 99% and additionally, it exhibits strong generalization capabilities. Therefore, it can be considered that the model is effective in crack detection, and it can be used for that goal in environments with low computational costs in future [10]. Although, the model mainly resulted in the high test accuracy, the bird classification is a little bit different than the problems above and in the following paragraph, the details will be mentioned.

1.3. Motivation

The problem in the focus is related to the supervised learning problem and multi-class image classification. It is a computer vision problem and the objective consists of categorization of the images into various pre-established classes. The aim is to develop a model capable of correctly assigning the appropriate label to an input image from a given set of classes.

In some articles, several approaches, changing based on the context and problem, have been used. These include cross-validation, transfer-learning, binary classification and

others which stimulated the model to have better accuracy in testing. In our problem, how transfer learning and architecture built from scratch works on the dataset with a low-quality level will be discussed and practically analyzed. Moreover, there will not be any data normalization, data augmentation or cross-validation techniques as the pure pixels of images are essential in the optimization of the model. Furthermore, as the number of classes increases, it gets more difficult for the model to predict the class as various label features can overlap, unlike a binary classification problem. Thus, some images may contain features that are shared among multiple classes, making it challenging for the model to make accurate distinctions. Briefly, transfer learning and fine-tuning, stratified data splitting and cropping the exact position of the images based on the segmented pixels will be the essential point in terms of completing the work.

2. Methodology

2.1. Data preprocessing

The dataset used in the classification problem is available on Kaggle and it is a various version of the CUB-200-2011 dataset. The dataset consists of a total of 11,788 photos of 200 distinct bird species and images have been collected from various sources and even there is some additional text on some images [2]. The dataset contains 200 folders representing the names of the birds correspondingly. The collection contains variously sized and high-quality photos of various bird positions, lighting setups, and backdrops. Moreover, images have different quality and shapes. For example, some pictures have been shot close to birds, but there are images taken of birds flying, standing on a tree or in their natural habitats. Briefly, this dataset is a valuable resource for researchers and practitioners working on bird classification and related tasks in computer vision. Additionally, another dataset having 20 classes and 3308 images is also used to check the performance of architecture constructed from scratch and for other purposes and the new dataset is used to make some clarifications related to the first dataset. Moreover, it was also accessible through the Kaggle platform.

Table 2.1 Datasets and their details

Dataset	Details
Dataset 1.1	Original images with 200 labels
Dataset 1.2	Cropped pixels of the birds based on their position in the images. Cropping has been done with the help of segmented pictures available in the original dataset
Dataset 2	Alternative to the main dataset due to low quality

As there are no preprocessing steps in the dataset as all pictures consist of true colors (RGB) and there will not be any data normalization in the dataset. In other words, each RGB image can be thought of as a matrix which represents RGB pixels and each pixel is characterized by three values between 0 and 255. It is understandable that the pixels reflect the intensity of the red, green, and blue color channels, correspondingly. Getting into details of RGB channels, every channel stands for a unique layer in the image and is equivalent to a grey-scale image. For instance, the red channel is responsible for the intensity of the red color in the image while brighter pixels show higher levels of red. Like the red ones, the green and blue channels indicate the intensity of green and blue light in the picture. In general, samples are kept as a tensor or an array containing pixel values between 0 and 255, which play the role of feature maps [1, 2].

The whole dataset is categorized into three additional datasets which are training, test and validation sets split in the percentages of 80, 10 and 10, respectively. The action of splitting the dataset is the primary reason that is used in the evaluation of the results (accuracy, generalization procedure and so on) of the model on the data that it has never seen before and to estimate how well it can generalize to unseen data. Thus, the model fits data in the training procedure whilst the test set allows for an unbiased estimate of the problem and is utilized to assess the model's ability on the predictions. Regarding the validation dataset, it is the key factor to check whether there is overfitting or not, it is needy to tune hyperparameters, like learning rate, batch size and number of epochs or not.

In the case of the splitting approach, two common methods are random splitting and stratified splitting. Random splitting involves dividing the dataset into subsets randomly, with each sample having an equal chance of being assigned to the corresponding set. This approach is commonly used when the distribution of the target variable is uniform across the entire dataset. On the other side, the dataset is partitioned in a manner that preserves the relative proportion of samples for each class in all sets. This

helps ensure that each class is fairly represented in both sets, thereby minimizing the risk of bias in evaluating the model's performance [11, 12]. Although the latter is employed when the target variable is imbalanced, this approach has been applied in order to overcome the problem. Figure 2.1 represents how the stratified splitting is done.

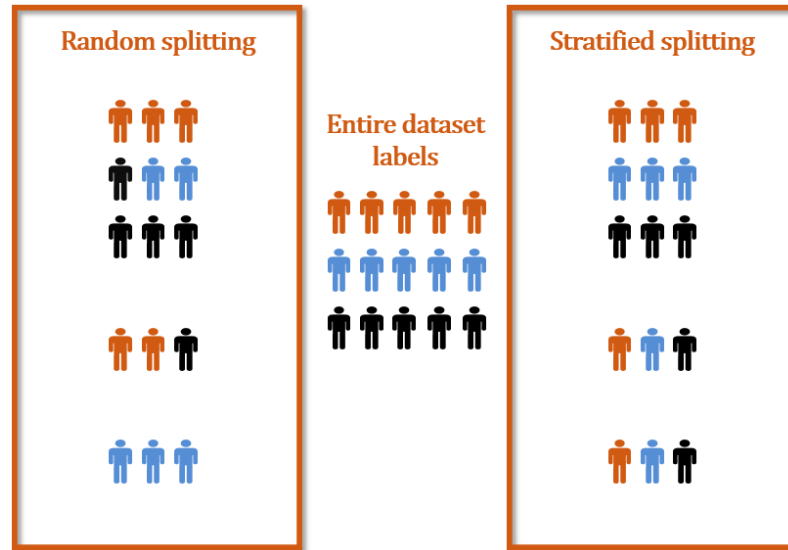


Figure 2.1 Stratified Splitting

2.2. Hyperparameters

There are essential parameters that should be tuned in the training procedure of the deep learning model. One of the hyperparameters is the learning rate which displays the step size in the optimization algorithms. The value of the learning rate should be balanced to find ideal weights. When it is high, it can result in overshoot and the weights can generate divergence, whereas a very low learning rate makes the model converge slowly. It can also cause the model to get stuck in a local minimum.

Another parameter is the number of epochs and in this context, an epoch means a single pass of all samples of the training dataset through a model [3]. Thus, the model's main goal is to train data and calculate the error. As a result, it can change the parameters to perform better on the task at hand. The number of epochs determines how many times the

network deals with training data. The number of epochs must be carefully chosen because using too few or too many epochs may result in underfitting or overfitting, respectively.

In deep learning, there is a term called batch size which is important for the optimization algorithm called mini-batch stochastic gradient descent. Thus, it can impact the model's training time, convergence speed, stability, and memory needs, picking the right batch size is essential. Smaller batch sizes can lead to faster convergence and more stable training, but slower training due to more frequent updates to the model's parameters. A larger batch size can reduce the training time, but may lead to less stable training and slower convergence, and more memory is required on the GPU.

2.3. Deep Learning model

To commence with, building Deep Learning models from scratch is a problematic task. It involves many steps including design procedure and implementation of neural network layers that can learn from large amounts of data. Mentioning the key factors and general description of the architecture, Google introduced EfficientNet in 2019, which possesses impressive feature extraction capabilities [10]. It fundamentally performs traditional neural network-related tasks by achieving higher accuracy while utilizing fewer parameters. The foundation of EfficientNet is constructed through a multi-classification neural architecture search process. Actually, it is a flexible architecture suitable for various computer vision tasks, including but not limited to animal detection. Although EfficientNet has been successfully utilized for animal detection, its applicability extends beyond this specific area and it can be effectively applied in numerous other computer vision applications [6, 13, 14].

They are connected networks representing the brain neurons and technologically, they are used to analyze images and videos. They are extensively employed in image detection, recognition and also classification tasks. It can also be understood as a

hierarchical model systematically extracting important features from images (RGB, grey-scaled). Generally, the design of CNNs is based on the organizational structure of the visual cortex of the human brain. It comprises interconnected nodes known as neurons arranged in multiple layers [14]. With each layer, it abstracts and captures feature information, ultimately enabling predictions to be made. It is well-known for its powerful performance in large-scale processing. CNN was the main central research focus in the last decades. In CNNs, there are several terms that are utilized to describe distinct aspects of network architecture and operations, and these include stride, padding, pooling, convolution and some other essential terms [10].

The pivotal element of CNN is the convolution layer, which conducts convolution operations on the input data. These operations entail moving a collection of filters across the input data and performing element-wise multiplication and summation to extract distinctive features [6]. Convolution is a crucial operation which involves applying convolution filters or kernels to the input data. These kernels, small matrices with optimized weights, enable CNNs to capture local patterns and spatial relationships in the data, such as edges, textures, and shapes in images [15]. By sliding the kernels over the input and performing element-wise multiplications, it generates feature maps that represent the filtered responses at each location. This hierarchical process is combined with other layers like pooling and fully connected one to allow architecture to learn changes [13]. So, abstract representations from the input picture can be really helpful in the generalization process of the model. Equations 2.1 and 2.2 shows how the width and height of the output of the convolution are found, respectively.

$$W_{\text{out}} = \frac{W_{\text{in}} - F + 2P}{S} - 1 \quad (2.1)$$

$$H_{\text{out}} = \frac{H_{\text{in}} - F + 2P}{S} - 1 \quad (2.2)$$

In the equations above, F is the filter size, P is the padding size and S is the value of stride. Figure 2.2-b on page 18 shows how the convolution process is accomplished.

2.4. Padding, stride and pooling

Padding is a valuable method used to retain spatial details and modify the dimensions of feature maps during the convolution process. It encompasses the addition of supplementary pixels or values surrounding the edges of input or intermediate feature maps prior to the convolution operation. Padding guarantees that the resultant feature map maintains either the same spatial dimensions as the input or a desired size. In the absence of padding, each subsequent convolution operation decreases the spatial dimensions of the feature map, gradually losing spatial information. Padding counteracts this reduction by introducing extra pixels around the borders, thereby upholding the spatial dimensions effectively [16]. Moreover, the edges of an input image or feature map carry important information, and convolution operations at the boundaries may not fully capture this information. Padding tackles this concern by introducing additional pixels around the edges, enabling the filters to comprehensively analyze the edge regions and capture pertinent features.

On other side, stride determines the shift or increment at which the filter moves both horizontally and vertically across the input data or feature map during convolution. It specifies the step size of the filter at each iteration, regulating the displacement applied to the filter. During convolution, the application of a filter, also referred to as a kernel, involves sliding it across the spatial dimensions of the input data. When the stride is set to 1, the filter moves over each pixel, but with a value of stride equal to 2, the filter moves two pixels for each action. The stride plays an important role to define the size of the output produced by a convolution. Moreover, when the stride is set to 1, the output keeps

the same spatial dimensions as the input and if the stride exceeds 1, it is understandable from the convolution operation that the output size is diminished. Using a greater value of stride leads to various consequences, such as a reduction in output size and information. Actually, the filter with an increased stride encompasses more extensive portions of the input data, resulting in the reduction of spatial samples, which can be advantageous in terms of reducing computational complexity and memory demands. In this instance, the filter, however, ignores a portion of the input data. This exclusion can result in the omission of intricate details and local patterns. Nonetheless, it can also facilitate the extraction of more generalized and advanced features.

Additionally, there is a term called pooling that is a crucial procedure trying to decrease the spatial dimensions, however, the action retains important information. Its purpose is to capture the fundamental characteristics of an image in a more compact form, thereby reducing computational complexity and assisting in achieving translation invariance, like stride. The size of the pooling is also known as the pool size or pool kernel. Its main goal is focused on balanced spatial dimensions. When the pooling operation is performed, the model achieves a balance between decreasing spatial dimensions for efficiency purposes [16]. Basically, it keeps enough information for precise classification. There are varied versions of pooling and they are explained below [13].

- Max Pooling. By using this type of pooling, the input feature map is divided into non-overlapping sections and then, it extracts the highest value within each section. Selecting the maximum value preserves the most prominent characteristic within that section, leading to a reduction in the spatial dimensions of the feature map.
- Avg Pooling. This has the same working or calculating mechanism as maximum pooling, but as implied by its name, average pooling calculates the mean value within each region of the feature map. This approach offers a smoothed representation of the input and assists in mitigating the influence of noisy or

insignificant features. However, it may not retain intricate details as effectively as max pooling.

- **Global Pooling.** This is alternatively referred to as global average pooling or global max pooling. It decreases computations on the complete feature map, amalgamating information from all spatial positions. By taking the average or maximum value, it condenses the feature map to a solitary value per feature channel. Global pooling proves beneficial in models where the input size can vary, as it guarantees a consistent output size regardless of the input dimensions.

Figure 2.2-a is a visual representation of the maximum and average (mean) pooling.

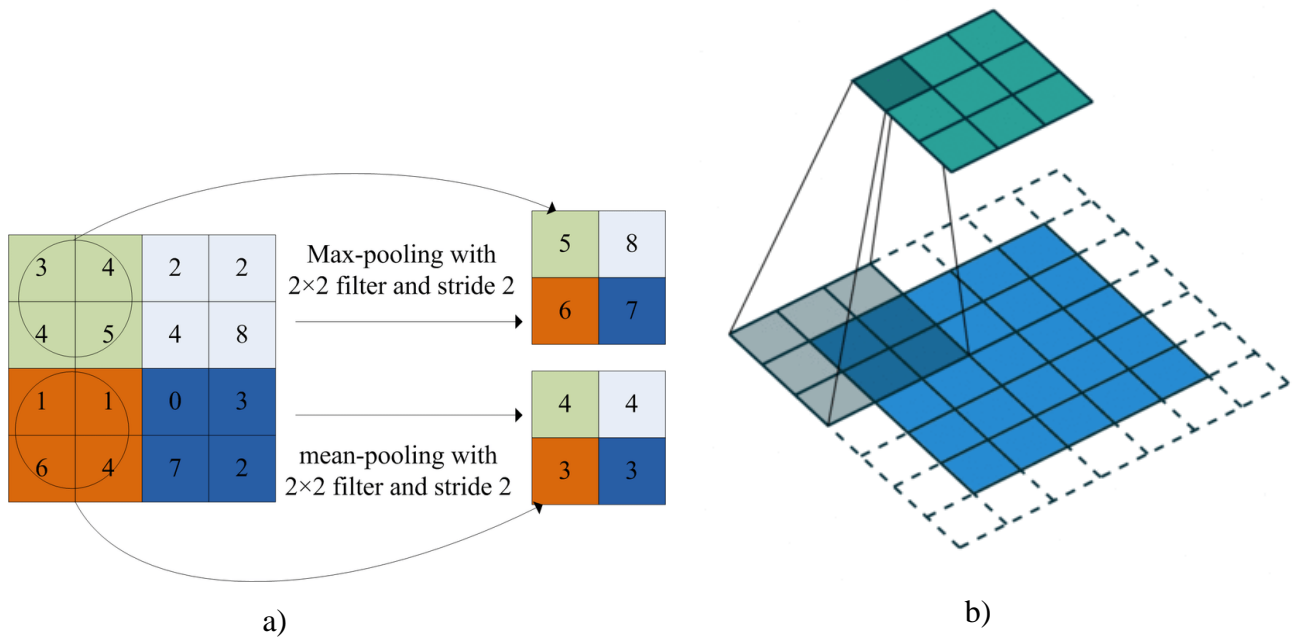


Figure 2.2:a) Pooling types b) Convolution

2.5. Activation functions

Activation functions have a crucial role in deep learning models as they introduce non-linear elements into the network. This inclusion enables the network to catch intricate connections between inputs and results, ultimately representing complex relationships effectively [10]. These functions are applied to the results of neurons or nodes within

artificial neural networks, determining the activation level of each neuron and facilitating the propagation of signals throughout the network.

Additional objective of activation functions is also related to their non-linearities as they are important in the decision-making process of the network. Without non-linear activation functions, deep neural networks would be limited to a series of linear transformations, impeding their capacity to learn essential patterns and generate meaningful predictions. By incorporating non-linear activation functions, neural networks can approximate any arbitrary function, thereby becoming universal functions [17].

2.5.1. Sigmoid.

The function is also called a logistic regression and is one of the mainly used mathematical functions both in Machine learning and Deep learning. The sigmoid function, which has also been used in the Squeeze Excitation block of EfficientNet architecture, accepts real-valued inputs and produces results ranging from 0 to 1. Equation 2.3 defines the formula of the sigmoid function [3, 17]:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

2.5.2. Rectified Linear Unit (ReLU).

This function gained popularity in Deep Learning as it is both straightforward and computationally efficient, serving to introduce non-linear behaviour within neural networks. It effectively transforms negative input values to zero, while maintaining positive values. Equation 2.4 is the formula of ReLU which can also be described as having piecewise equations [18, 17]. As it is mentioned, the output for the non-positive inputs will be 0.

$$f(x) = \max(0, x) \quad (2.4)$$

ReLU strictly generates non-linearities together with sparse activations as it has the ability to set negative values to zero. Potential sparse activations are beneficial as they contribute to the reduction of the computational complexity of the network. By emphasizing important features, they can also develop the generalization process of the model and lead the model to have a better performance in various tasks. ReLU should also be known for its certain drawbacks, including the "Dying ReLU" problem. In this problem, some neurons do not respond to the connections and produce a gradient equal to zero. This occurs when many neurons have negative weights and receive inputs resulting in negative activations. Approaches like leaky ReLU and parametric ReLU have been proposed to mitigate this issue. Another limitation is that ReLU is unbounded on the positive side, which can cause numerical instability in training, especially with large learning rates. Moreover, ReLU fails to preserve the magnitudes of positive values, resulting in an imbalanced output distribution where all positive values are mapped to the same value [18]. Commonly, it is one of the widely used activation functions dependent on the situation despite limitations.

2.5.3. Sigmoid Linear Unit (SiLU).

SiLU or swish activation function is a core part of the EfficientNet architecture that has been used many times in the blocks-generating layers of architecture [9]. It has even been utilized as a final layer activation function in which mainly Softmax function is used. SiLU is the combination of the sigmoid and identity functions and that is why it is a really suitable choice for neural networks for carrying out the double characteristics. It is a non-monotonic function having a shape looking like the ReLU, however, it does not directly make non-positive values zero. So it contributes to the performance of the model in many tasks related to picture manipulation and text processing [10, 19]. Here is the formula:

$$y(x) = x * \sigma(x) = \frac{x}{1 + e^{-x}} \quad (2.5)$$

One significant advantage of SiLU is its smoothness unlike ReLU and it brings an efficient gradient optimization during backward propagation and generates faster and more stable training. Moreover, SiLU's computational efficiency, involving simple element-wise operations, adds to its appeal. SiLU has been proposed as an alternative to popular activation functions like ReLU. SiLU is a mainly used activation function in EfficientNet architecture and it has demonstrated superior performance in certain scenarios.

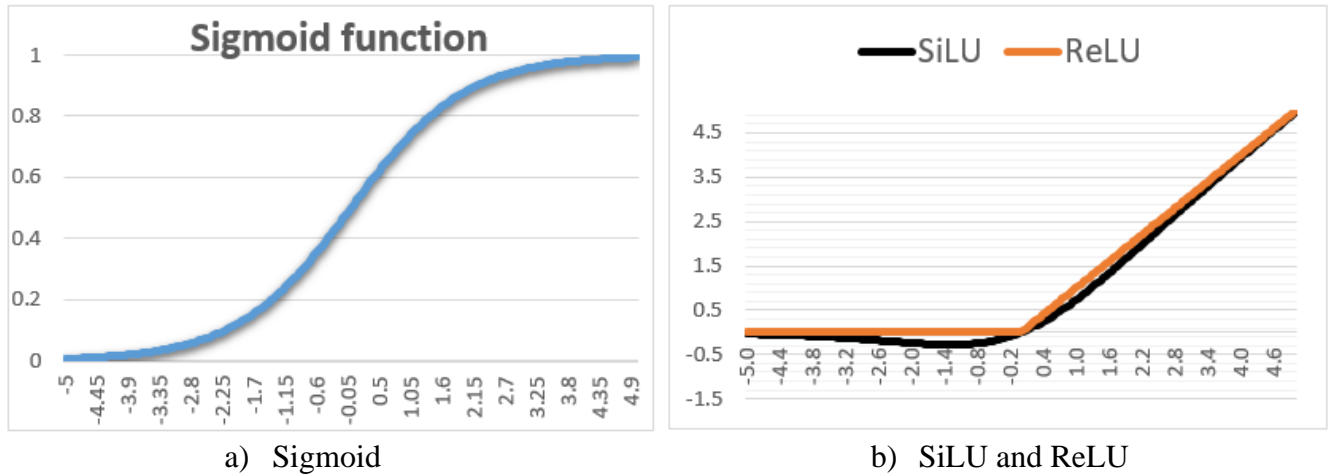


Figure 2.3 Activaton functions

2.6. Batch normalization.

Apart from the batch size, the EfficienNet architecture employs a concept known as Batch Normalization. The layer activations are normalized using this method. Each mini-batch is normalized by taking the mean away and dividing by the standard deviation. The normalized activations are further adjusted using trainable parameters, enabling the network to optimize the representation of each layer [20]. Batch normalization provides several advantages, including accelerated training speed by eliminating the need for careful initialization and facilitating higher learning rates. It acts as a regularization technique by introducing noise during training, which helps prevent overfitting and enhances generalization [21]. Additionally, batch normalization aids in addressing vanishing and exploding gradient problems. Generally, the method makes the internal

shift reduce in order to cope with the problem mentioned and consequently, stable gradients across the network is obtained. On other words, it is accomplished by reducing the internal shift and maintaining the gradients constant.

2.7. Dropout.

It is a regularization technique frequently used in Convolutional Neural Networks to cope with overfitting. During training, it randomly deactivates a portion of neurons within a layer. The primary objective of dropout is to foster the development of resilient and adaptable features by reducing the co-dependence among neurons. By selectively dropping out neurons, the network becomes less reliant on specific neuron weights and instead learns from various subsets of neurons. Consequently, it guards against excessive dependence on particular neurons and prevents the network from memorizing irrelevant details or noise in the training data.

Table 2.2 Dropout rate and input picture resolution for the EfficientNet type

EfficientNet architecture	Dropout rate	Input resolution
B0	0.2	224x224
B1	0.2	240x240
B2	0.3	260x260
B3	0.3	300x300
B4	0.4	380x380
B5	0.4	456x456
B6	0.5	528x528
B7	0.5	600x600

Thus, dropout proves effective in diminishing overfitting and enhancing generalization capabilities in CNNs [15]. It functions as a regularization mechanism by effectively creating an ensemble of multiple neural networks with shared weights. During testing, dropout is typically deactivated, and the network with all neurons kept active is utilized

for making predictions. The extent of neuron dropout is regulated by a parameter known as the dropout rate, which requires fine-tuning throughout the training process. For example, if the dropout rate is 0.2, it means that twenty percent of the neurons will stochastically become zero. So, the values of the rate depending on architecture are mentioned in Table 2.2 [22].

2.8. EfficientNet architecture.

To begin with, EfficientNetB0 applies a convolution with the kernel value equal to 3 (3x3) to the image. Subsequently, the subsequent 16 mobile inverted bottleneck convolution modules are used to extract additional image features [21]. Followingly, pointwise convolution, batch normalization, global average pooling, fully connected layers and activation function are applied to the output of the final mobile inverted bottleneck block [8, 10]. A fully connected neural network (FC) has a complete interconnection pattern because every neuron in the current layer is connected to every neuron in the following layer [16]. This interconnectedness enables information flow from all the neurons in the next layer to each neuron in the current layer. Each neuron in the fully connected layer receives inputs from the entire preceding layer, undergoes a linear transformation, and applies a non-linear activation function, facilitating complex feature extraction and representation learning in the network. This type of layer is commonly used in deep learning models to capture intricate relationships and patterns within the data. The linear transformation involves multiplying the inputs by learnable weights and adding a bias term. This step allows the network to learn the appropriate combination of features for the given task. Finally, By incorporating a non-linear activation function, the model gains the ability to capture intricate relationships and make non-linear predictions [20].

The model consists of two essential elements. Squeeze Excitation Blocks and inverted bottleneck convolution blocks (MBConv) are two examples. Blocks of inverted bottleneck convolution are crucial for feature extraction while balancing model

performance and efficiency [23, 10]. The block has an inverted bottleneck structure and is made up of a Squeeze-and-Excitation block, a pointwise (1x1) convolution layer, a depthwise convolution layer whose performance depends on the kernel size, and another pointwise convolution layer [9, 8].

Turning to Squeeze-and-Excitation (SE), it is an optional block that can enhance the representation power by making corrections in the channel-wise responses based on the task. The block consists of two main steps as it can be understood from the name. In the initial phase, termed the squeezing step, the objective is to gather global information from each channel [10]. This is accomplished by employing global average pooling, which involves reducing the spatial dimensions. Through this process, the channel's content is comprehensively captured by aggregating the spatial features using averaging. This pooling technique enables a holistic representation of the channel's information, facilitating a deeper understanding of the data. Then, the essential parameter of the block called the expansion factor is used. The parameter increases the number of channels before applying depth-wise convolution [24]. In the subsequent step, known as the exciting step, the learned squeezed information is passed through the activation function (SiLU), allowing for the exploration of channel-wise relationships [8]. By applying the activation function to the squeezed data, the model gains the capability to capture and understand the intricate connections and dependencies between different channels. This process enhances the ability of the model in the extraction of meaningful patterns which leads to improved performance. Taking every detail into account, the process allows the network to focus on informative features and discard less relevant ones. Furthermore, it is important to note that when the expansion factor in the block is equal to one, the initial point-wise convolution or three actions (Conv2d, Batch normalization and SiLU) of the block are skipped [8].

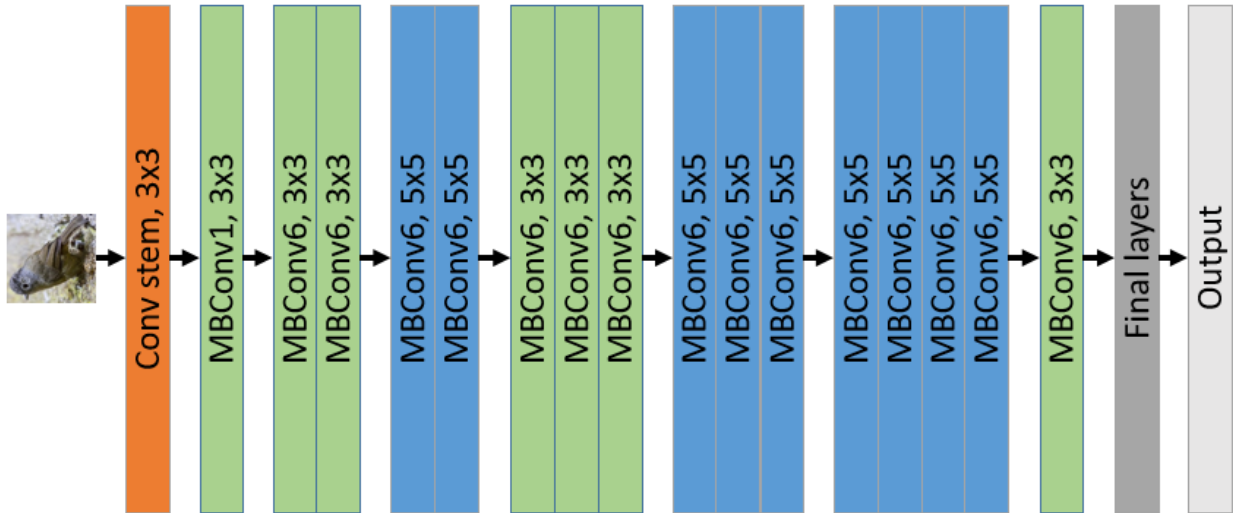


Figure 2.4 EfficientNet-B0

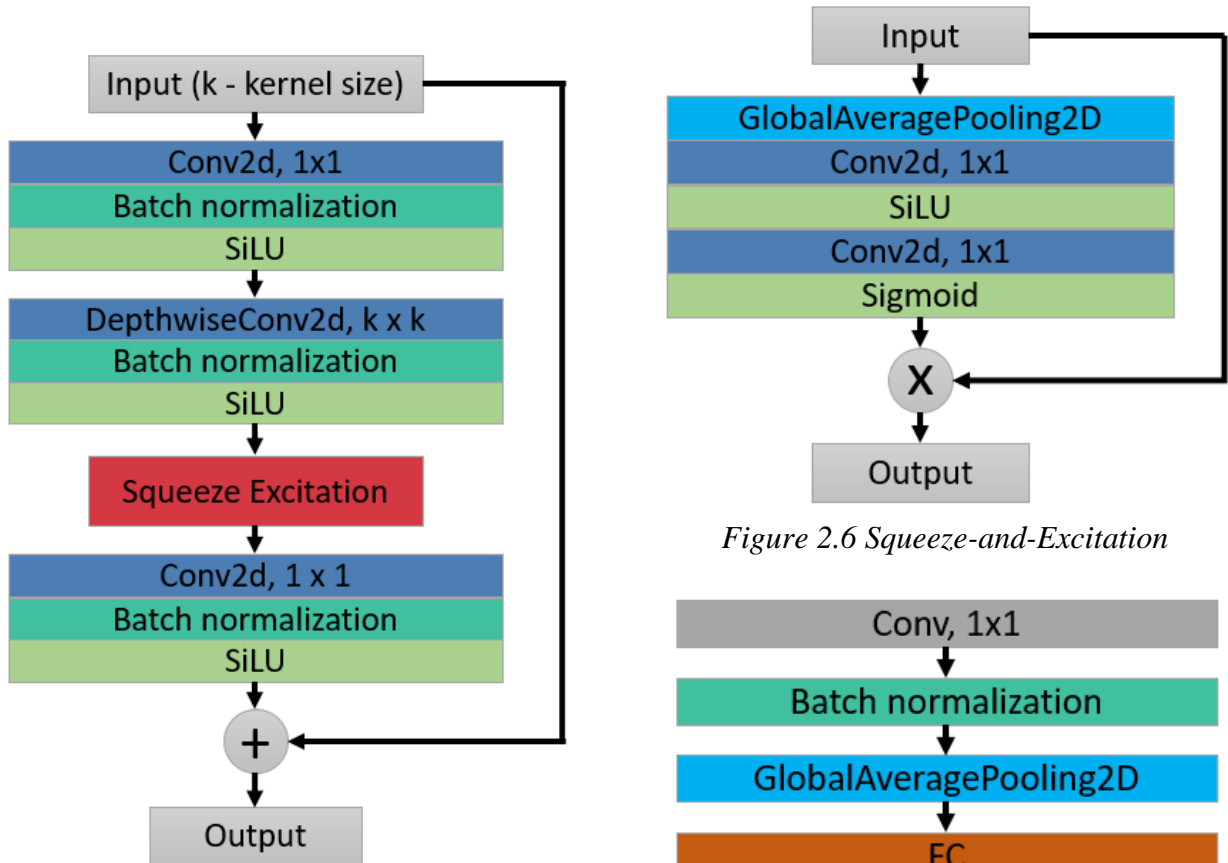


Figure 2.5 MBConv block

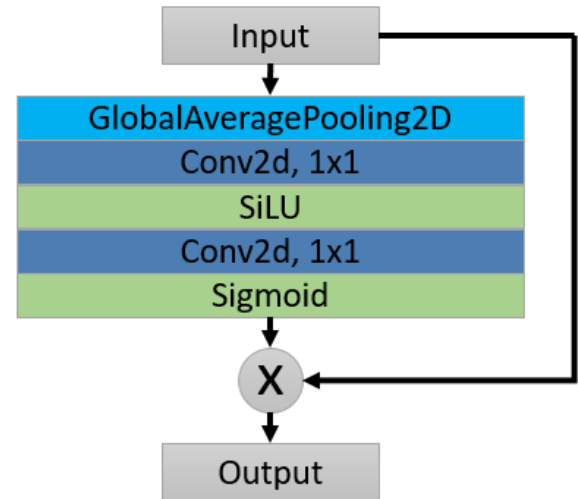


Figure 2.6 Squeeze-and-Excitation

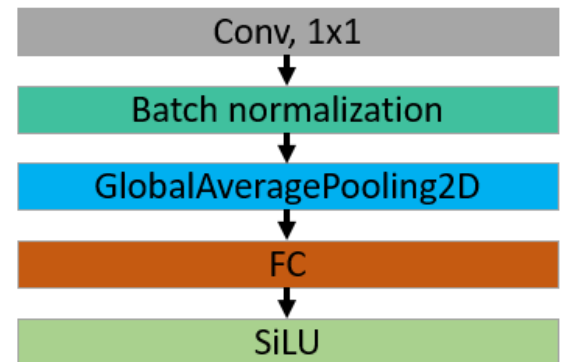


Figure 2.7 Final layers

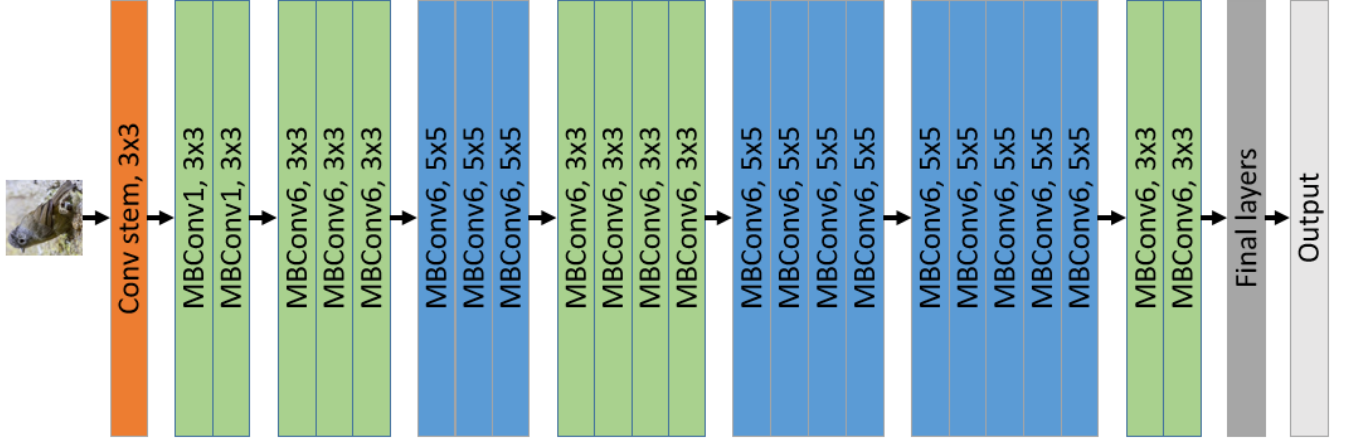


Figure 2.8 EfficientNet-B1 architecture

On the other hand, the EfficientNet-B1 (Figure 2.9) has a different number of MBConv blocks, but the rest is actually the same as the EfficientNet-B0 (Figure 2.5) [25]. Additionally, sub-blocks of the models have been represented below. So, Figure 2.6 is the MBConv block while Figure 2.7 shows the Squeeze-and-Excitation block. The final layers have been grouped in Figure 2.8.

2.9. Transfer learning.

A pre-trained model within the realm of deep learning covers a model that has undergone training on an extensive dataset and even it helps the model avoid overfitting [15]. These models undergo training using a copious amount of labeled data, enabling them to catch complex patterns and representations within the data. These models in the training process are beneficial in terms of significant computer resources. By leveraging the computational resources, the transfer learning method acquires a comprehensive understanding of the data and develops a remarkable capacity to generalize their knowledge as they cope with the problems with pre-trained models [22, 9, 3].

Although the EfficientNet-B0 architecture and EfficientNet-B1 architecture constructed from scratch have been used in the training process, a pre-trained model begins with tuned gradients and representations obtained from its previous training on a specific dataset. It

possesses learned features and patterns that are generally relevant to the problem domain. Conversely, a model built from scratch initiates with randomly set parameters, lacking any prior knowledge or understanding of the data or the problem it aims to solve.

Going through additional differences, utilizing pre-trained models is advantageous in order to save time and resources unlike training a deep learning model from scratch and it is blatantly clear that pre-trained models can result in better performance. The pre-trained models are not always appropriate for a problem. So, the ability to create a model from scratch enables developers to train the model exclusively on their own dataset [22]. Totally, it guarantees that it accurately captures the domain-specific features and patterns.

2.10. Loss function and optimizers.

In deep learning models, the loss function is a critical method because it determines the error between predicted and true values. For classification tasks in neural networks, the cross-entropy loss function is a popular cost function. Minimizing the value of this function means the weights approach their optimal values and it improves the ability of the model to predict the correct classes with higher probabilities. Generally, it enhances classification accuracy as the loss function plays a vital role in evaluating the model's performance. Throughout the training process, the loss function is continuously updated which directly reduces errors and achieves the best possible fit. Repeatedly, the cross-entropy loss function is a crucial technique used to define the deviation between the predicted and actual values [10, 3]. Its formula is represented below:

$$L = -\frac{1}{N} \sum_{i=1}^N \log \left(\frac{e^{a_k}}{\sum_j e^{a_j}} \right) \quad (2.6)$$

where N represents the number of neurons in the output layer and a_k is the input signal.

In Deep Learning, an optimizer is an algorithm utilized to adjust the parameters of a neural network model during the training process. The main objective of an optimizer is to minimize the loss function by updating the parameters iteratively. The selection of a suitable optimizer is crucial because it can significantly impact the training process. There are various optimizers having unique properties and choosing an appropriate optimizer is an essential step in developing a successful Deep Learning model [26]. In order to accomplish the bird classification problem, SGD with momentum and Adam optimization algorithms have been used in the programming part. Both of them are the advanced form of the SGD (Stochastic Gradient Descent) algorithm with some changes. Analysis of differences among these algorithms will be mentioned with all nuances [27].

SGD updates the parameters of a neural network with small steps proportional to the negative gradient of the loss function. In other words, it adjusts the parameters directly based on the current gradient. Here is the formula for SGD in which α is the learning rate, $\nabla_w L(W, X, y)$ is the gradient of the loss function L while W_{t-1} is the previous weights and W_t is the updated values:

$$W_t = W_{t-1} - \alpha \nabla_w L(W, X, y) \quad (2.7)$$

On the other hand, SGD with momentum introduces a concept of "momentum" to the update parameters. Momentum helps the model accelerate the learning process, especially when the loss function is noisy or has a lot of local optima. It achieves this by adding a fraction of the previous update vector to the current update. It is defined as the formula below:

$$V_t = \beta V_{t-1} + (1 - \beta) \nabla_w L(W_{t-1}, X, y) \quad (2.8)$$

$$W = W_{t-1} - \alpha V_t \quad (2.9)$$

In the equation above, V_t is the velocity at each iteration t , β is the momentum coefficient and W is the parameter that is updated based on the previous velocity. In the

update rule above, momentum controls the impact of the previous updates, and the learning rate determines the step size. The usual range for the momentum value in SGD with Momentum is between 0.9 and 0.99. A widely suggested default value is 0.9, which often yields good results across various scenarios. Nevertheless, the ideal value can differ based on the particular problem and dataset.

While SGD and SGD with momentum use a fixed learning rate, Adam dynamically adjusts the learning rate for each parameter. It computes adaptive learning rates based on estimates of the gradients' first and second moments [26, 27]. Unlike SGD with momentum, Adam incorporates momentum differently. Instead of accumulating previous update vectors, Adam maintains exponentially reducing averages of both the gradients and their squared values. These averages, known as the first moment (mean) and second moment (variance) of the gradients, respectively, are utilized to modify the learning rates for each parameter [10]. In order to analyze the optimization algorithm, the better option is to understand the formula of the optimization method in the equations below:

$$m = \beta_1 * m + (1 - \beta_1) * g_t \quad (2.10)$$

$$v = \beta_2 * v + (1 - \beta_2) * g_t^2 \quad (2.11)$$

$$m^{\wedge} = \frac{m}{1 - \beta_1^t} \quad (2.12)$$

$$v^{\wedge} = \frac{v}{1 - \beta_2^t} \quad (2.13)$$

To update the parameters above the following equation will be used:

$$\theta = \theta - \alpha * \frac{m^{\wedge}}{\sqrt{v^{\wedge}} + \varepsilon} \quad (2.14)$$

In this equation, α stands for the learning rate, beta values are the exponential decay rates, m and v are the mean and variance of the gradients.

3. Discussion of the results

3.1. Training environment, parameters and framework

The entire experiment and actions in this paper have been accomplished in the Google Colab environment with the help of PyTorch, a framework offering a dynamic computational graph and automatic differentiation and allowing an efficient training process of neural networks. Moreover, Google Colab provides an additional GPU (Graphics Processing Unit) with the type of T4 and it has several advantages compared to the CPU. Some of the plus features of GPU are mentioned below:

- Parallel processing because of a lot of cores allowing multiple computations simultaneously.
- Higher performance as popular deep learning frameworks have built-in support for GPU acceleration.
- Faster matrix operations, such as multiplications, convolutions, and cost efficiency.

Getting into training and model parameters, two models with various versions, including architecture built from scratch, a pre-trained model with the previously reached gradient values and fine-tuning in which the model starts to train the whole architecture by optimizing gradients, are the core point of the application on the datasets. Furthermore, the parameters like input resolution, batch size, number of epochs and so on are essential nuances to manage. The input resolution of the images is $3 \times 300 \times 300$ and $3 \times 224 \times 224$ for Dataset 1 and Dataset 2, respectively while the training process of models on the former is done with a batch size of 32, however, it is 16 for the latter. Commonly, the number of epochs ranges from 10 to 50 based on the application of the model and the dataset.

3.2. Evaluation metrics

In the context of Machine Learning and Deep Learning, TP, TN, FP and FN are essential terms to describe the results of the classification tasks. Fundamentally, they help the model's predictions compared to the ground truth labels of the dataset. The terms are explained below [25, 9, 10, 3, 24].

- True Positive (TP) is considered in the circumstances that the model accurately predicts that the target is positive.
- True Negative (TN) occurs when the model correctly classifies a negative example as negative.
- False Positive (FP or Type I error) happens after the model predicts the target as a negative instance, however, it is a positive one.
- False Negative (FN or Type II error) is encountered in the situation that the model predicts the sample as a positive instance but it is negative in actuality.

These concepts are frequently utilized to build a confusion matrix, which serves as a table representing the classification model's performance. The distribution of true positives, true negatives, false positives, and false negatives is represented visually in the confusion matrix in a structured manner. By using a confusion matrix, we can gain insights into how the model's predictions align with the actual ground truth labels and evaluate its overall performance in a more organized manner. Although these values are used to calculate several metrics, like precision, F1-score and others, merely accuracy has been used in the bird classification problem to estimate the model. Accuracy defines the ability of the model to correctly classify instances across all classes. It determines the overall performance of the model. The ratio of the total number of correct predictions to the total number of instances is the accuracy metric that provides every detail about how the model is effective at the prediction. The formula below shows the accuracy metric [4, 10, 25, 21]:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{True predicted}}{\text{Entire dataset}} \quad (3.1)$$

Additionally, there is a difference in building a confusion matrix between binary and multi-label classification problems. In contrast to binary classification, the absence of distinct positive or negative classes makes it initially challenging to identify the values. However, this task can be simplified by determining these metrics for each specific class separately. Therefore, the process involves finding TP, TN, FP, and FN for each individual class, allowing us to assess the model's performance in a more nuanced manner [24].

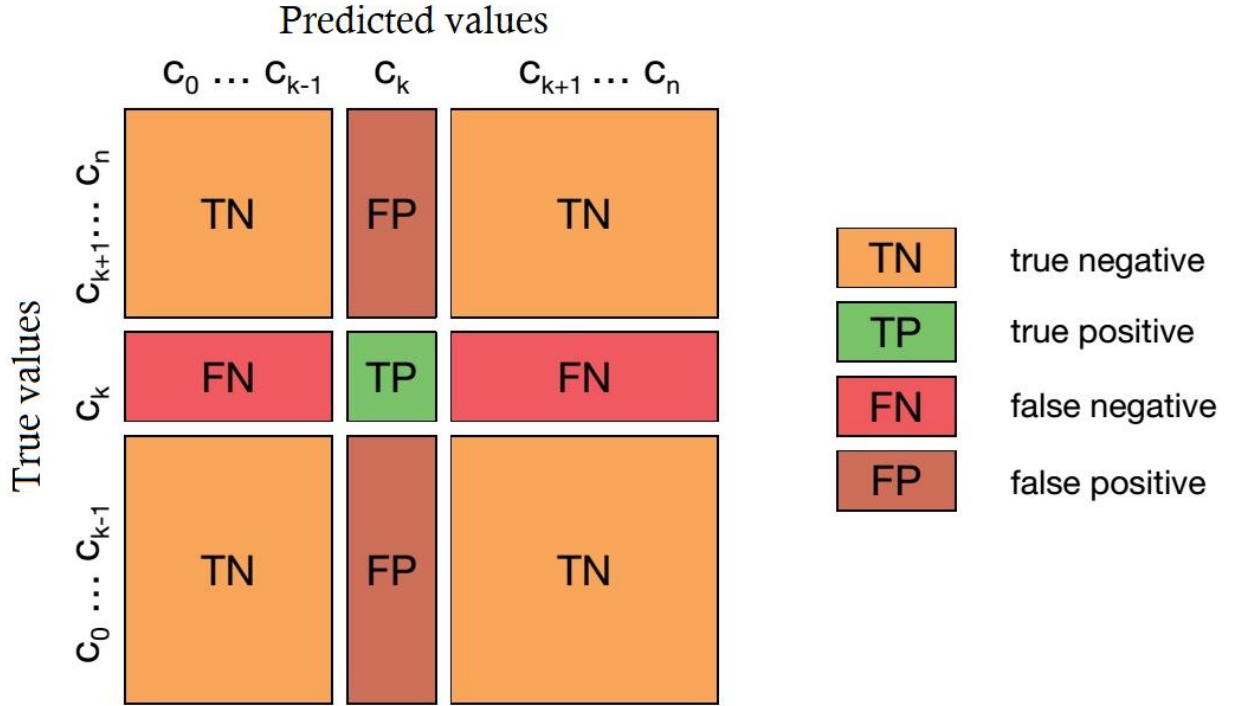


Figure 3.1 Confusion matrix for multi-classification

3.3. Results and analysis of the experiment

In the previous subchapters, how the models' efficiency will be calculated and how the architecture can affect the classification of the images with possible influences of the changing the values of the parameters were discussed, but currently, the main concentration point is just comparison of the models constructed with various features.

Table 3.1 represents the details of the observation accomplished on Dataset 1 (both on the original and cropped versions). As SGD with Momentum optimizer as a training parameter have been utilized in the experiment and batch size is equal to 32, there was no need to add ‘optimizer’ and ‘batch size’ columns to the table.

Table 3.1 Results of the models applied on the main dataset

EfficientNet Architecture	Dataset	Number of epochs	Learning rate	Test accuracy
B0 (scratch)	Dataset 1.1	50	0.1	35.36%
B0 (scratch)	Dataset 1.2	40	0.1	50.97%
B0 (Fine-tuning)	Dataset 1.1	20	0.1 (¹ Step scheduler)	76.96%
B0 (Fine-tuning)	Dataset 1.2	15	0.1 (Step scheduler)	83.8%
B1 (Fine-tuning)	Dataset 1.1	15	0.1 (Step scheduler)	78.48%
B1 (Fine-tuning)	Dataset 1.2	15	0.1 (Step scheduler)	84.64%

On the other side, Table 3.2 shows detailed information about the application of the model on the second dataset. Additionally, the batch size is equal to 16 for all training processes because of the lower samples in the dataset. Based on the results obtained from the practical application of two datasets, an extensive analysis of the work can be described.

Table 3.2 Results of the models applied on Dataset 2

EfficientNet Architecture	Number of epochs	Learning rate	Optimizer	Test accuracy
B0 (scratch)	30	0.005	SGD (Momentum)	77.31%
		0.001	Adam	74.33%
B1 (scratch)	30	0.005	SGD (Momentum)	76.49%
B0 (Pre-trained)	13	0.1 (Step scheduler)	SGD (Momentum)	95.24%
			Adam	95.83%
B1 (Pre-trained)	13	0.1 (Step scheduler)	SGD (Momentum)	93.45%

¹ Step scheduler is used to adjust the training process by reducing the learning rate after specific number of epochs

Initially, the main goal of the project was to understand the working principle of CNNs, possess knowledge about crucial terms like convolution, and backwards-propagation, and finally be able to type the architecture from scratch together with getting the best possible accuracy after the entire procedure. But an unexpected issue related to the dataset occurred and impacted the performance of the models in a tremendously negative manner. Shortly, the problem is the quality of the pictures in the dataset and in order to prove the issue a new dataset (Dataset 2) with better quality has been utilized.

At first glance, it is clearly visible from the statistics about test accuracies on two distinct datasets that the results accounting for Dataset 1 are less than that in Dataset 2. Thus, using only pre-trained EfficientNet-B0 on the second dataset led to more than 95% test accuracy, but even one of the effective techniques named Fine-tuning brought on accuracy a little bit less than 84%. Additionally, in the first case, original images of Dataset 2 were used and prompted the model to have amazing accuracy, in the second scenario some pre-processing steps or cropping operations have been fulfilled. In addition to the variation observed in pre-trained models, the accuracy of the model constructed without pre-training also exhibits a notable difference. Despite achieving a test accuracy of 50% on the initial dataset, which is considered the optimal outcome for the code developed from scratch, the same architecture demonstrates a higher performance of approximately 75%



a)



b)



c)

Figure 3.2 Sample images

on the second dataset based on the optimizer. In order to get a logical understanding of the problem, it would be a great step to analyze various bird images in the first dataset.

To begin with an analysis of the images due to their shapes and the position of the birds in the pictures, three images, Figure 3.2, are displayed above. In Figure 3.2-a (Species: Bobolink), the resolution of the image is 500x306, but the bird only captures the 86x93 part of the entire image. Sequentially, in Figure 3.2-b (Species: Prairie Warbler), the dimensions of the actual image are 500x346 whilst the bird is located in the part having a resolution of 438x301 and finally, in Figure 3.2-c (Species: Bronzed Cowbird), both the resolution of the image and the number of pixels generating the bird is really low at 208x137 and 91x106, accordingly. There are 2325 images in which the bird merely has width and height pixels less than 200 and images of these types are problematic for the generalization of the model.

These factors can also affect a deep learning model in terms of detection and feature representation. Detecting and precisely localizing a small bird within a large image can pose challenges for the model, potentially resulting in errors in identifying the bird's boundaries. Conversely, a large bird in a large image is generally easier to detect due to its prominent size and distinct features. Furthermore, when dealing with small images in the dataset, the model may face difficulties in object detection and recognition due to the limited information available.

Another nuance is the value of the learning rate for pre-trained models. Thus, the learning rate for all of them is 0.1 and if it is set to 0.1 and the deep learning model built for image classification is able to generalize well, commonly the model learned meaningful and crucial representations from the images. Furthermore, the models have efficiently captured the important patterns in the images.

Regarding the application of the models on the first dataset, it is visible that the best result on the original images (Dataset 1.1) occurs with fine-tuned EfficientNet-B1 model with a test accuracy of 78.48% while fine-tuned EfficientNet-B0 results in a test accuracy of 76.96%. In fact, it means that as the bird is deeply hidden in some images, the size and quality of the images are frustrating and that is why accuracy goes up as the structure of the model becomes more complex. Thus, the complex model can determine the main parts of the bird with more details. Even though B0 was trained additional 5 epochs with a learning rate of 0.0001, the generalization process happens almost in 8-11 epochs. Thus, the number of epochs for other pre-trained models applied on Dataset 1 is 15 epochs. In order to understand the generalization process, the loss graph and accuracy graph for each model can be analyzed. Figure 3.3 and Figure 3.4 shows the loss and accuracy graphic

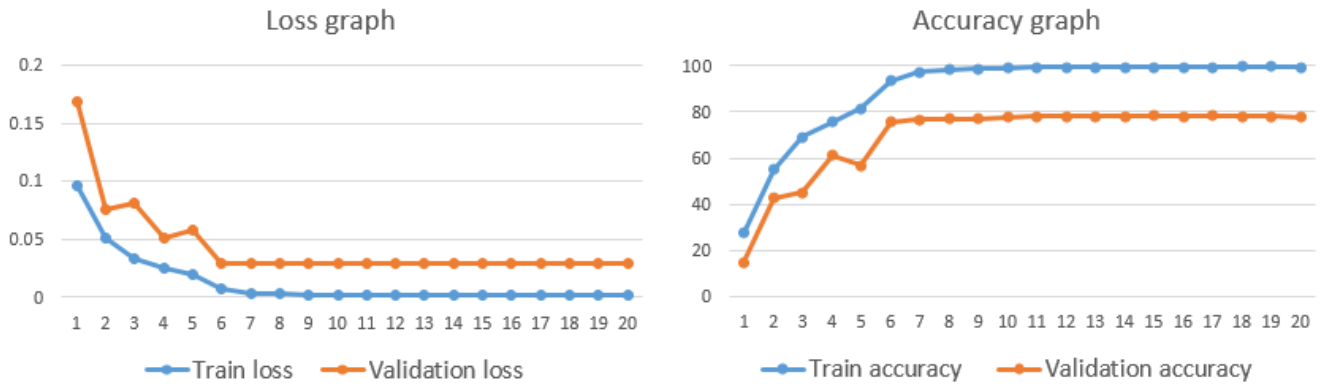


Figure 3.3 EfficientNet-B0 (Fine-tuning) loss and accuracy graph

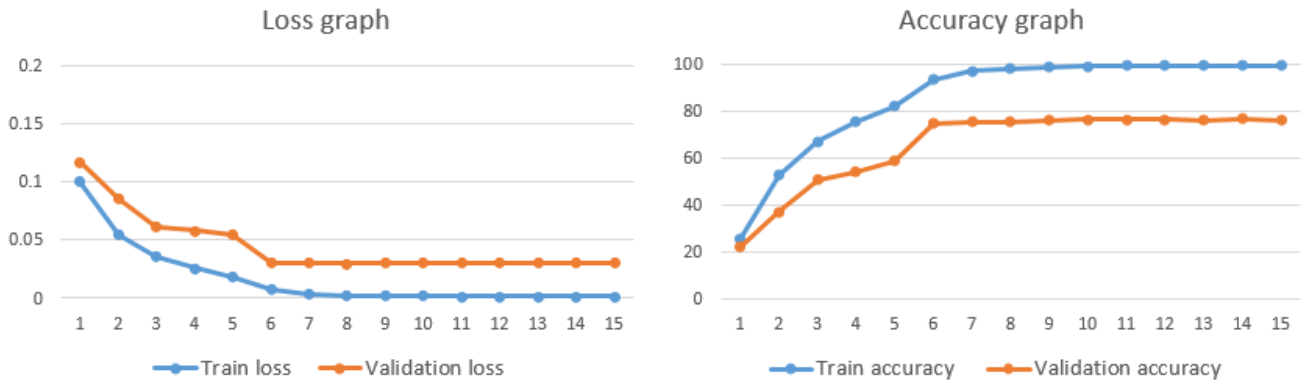


Figure 3.4 EfficientNet-B1 (Fine-tuning) loss and accuracy graph

representations for EfficientNet-B0 and EfficientNet-B1 on the first dataset, correspondingly.

It is also noticeable from the graphs that, after 7 or 8 epochs the training and validation lines get parallel to each other and the model has done its best in order to learn from the training procedure and reached the maximum ability to predict an unseen data. So, after the initial 5 epochs with a learning rate of 0.1, both training and validation loss critically decreases while the accuracy for both increases, however, then the accuracy values for validation get constant at around 78% while training accuracy obtains 100%.

Shifting the focus to Dataset 1.2, the results reveal that exactly cropping a bird image from the entire image can rise the accuracy. How the process is accomplished is explained in detail together with the actual representation in Figure 3.5.

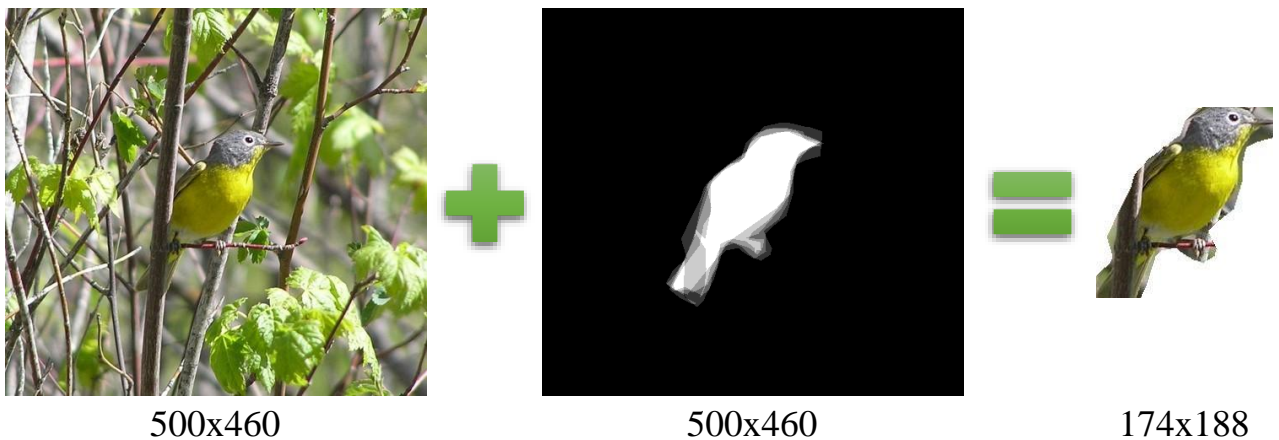


Figure 3.5 How dataset 1.2 has been created

Dataset 1.2 is just a version of Dataset 1.1 with a pre-processing step of cropping the exact bird from the image as it has been displayed above. The dataset in Kaggle also gives its users the segmented formats of each image in the whole dataset. By using these segmented pictures, the pixels forming the bird images can be clearly observed and found. Therefore, only by picking the white pixels or the maximum height and width referring to the segmented picture, the part of the shot containing the bird can be sliced and utilized in the training process. From the results shown in Table 3.1, how effective this action was is

understandable it totally increased the test accuracy by 6-7% for pre-trained models. On the other hand, while it gave 35% test accuracy for the model made from scratch on the original images, by using the method above its test accuracy climbed 15% and reached 50%. Unfortunately, the results for the model from scratch are not satisfactory and that is why the main concentration will be on the pre-trained models. Figure 3.6 and Figure 3.7 help us to delve into what is going on in the generalization processes on Dataset 1.2 for EfficientNet-B0 and EfficientNet-B1, respectively.

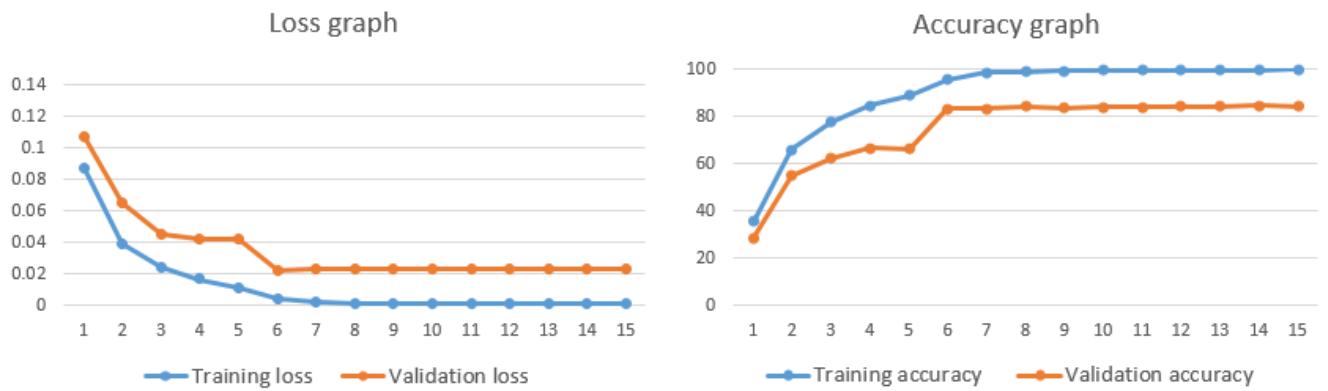


Figure 3.6 EfficientNet-B0 (Fine-tuning) loss and accuracy graph on Dataset 1.2

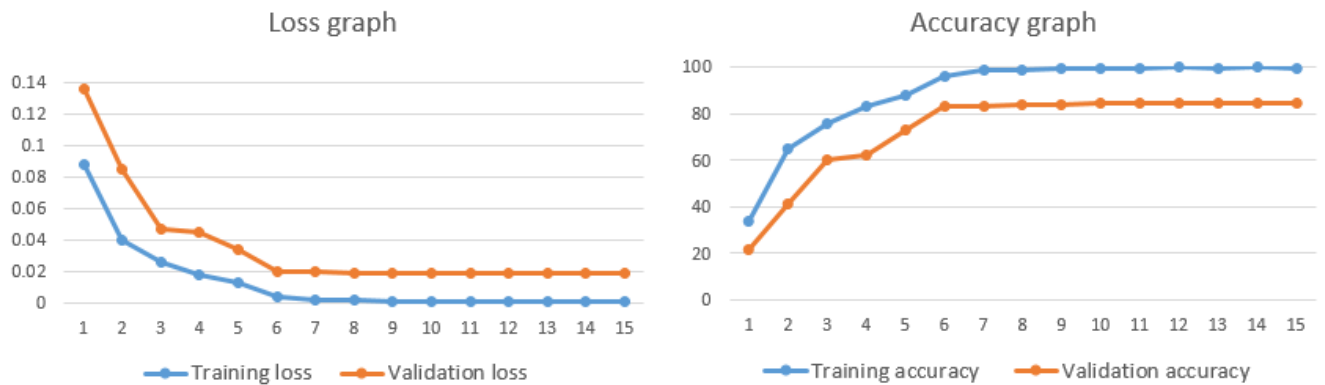

















Figure 3.7 EfficientNet-B1 (Fine-tuning) loss and accuracy graph on Dataset 1.2

In the current paragraph, the best test accuracy obtained with the help of fine-tuned EfficientNet-B1 on the first dataset will be discussed. Table 3.3 below represents some wrong predictions with additional information.

Table 3.3 Some wrong predictions of EfficientNet-B1 (84.64%)

Original image	Cropped image	Actual class (Label)	Predicted class (Label)	Predicted class sample image
		Acadian Flycatcher (37)	Least Flycatcher (39)	
		Caspian Tern (143)	Common Tern (144)	
		Yellow- bellied Flycatcher (43)	Yellow- breasted Chat (20)	
		Ring-billed Gull (64)	Ivory Gull (63)	
		Black- billed Cuckoo (31)	Yellow- billed Cuckoo (33)	

There were 1185 test samples and 1002 of them have been correctly guesstimated by the model which resulted in a test accuracy of 84.64%. At the final epoch, the training loss was 0.0008 and the validation loss was 0.0188, while the training accuracy was 99.69% and the validation accuracy was equal to 84.71%.

Graphs in B0 and B1 models started to stabilize after 6 epochs as it was mentioned earlier in other figures. In Table 3.3, there are explanotary data such as a picture of the predicted class, in order to understand the reason why the model made the erroneous determination in the bird classification task.

From the table above, it can be shown that the model misclassifies the samples with another type of the same species. For example, we can see that both actual and predicted species are flycatchers but one of them is Acadian and another one is Least in the first row of the table. This happens for Tern, Cuckoo and Gull, too. The difference between true and predicted ones can be visually understood for the Gull and Cuckoo. Ring-billed Gull has a black edge on its wing, however, Ivory Gull is fully white. This misclassification can rely on the many white pixels in the sample image of the Ring-billed Gull, but turning to the second row of the table, it is even challenging for a human to differentiate Caspian Tern from Common Tern as they totally look like each other. That is why it would not be a correct choice to blame the model for the misclassification of the Caspian Tern. For the third sample in the table, we can see that the model has difficulties analyzing the yellow colors as the species of the birds are exactly different but they are the same in the color. It can also be understood as both birds have almost the same body features, too.

To check how the model is effective, the model from scratch and the pre-trained model are applied to the new dataset. The EfficientNet-B0 architecture constructed from scratch resulted in 77.31% with SGD with momentum and 74.33% with Adam optimizer, but the learning rate for the former was 0.005 while that was 0.001 for the latter. It is clear that the model from scratch cannot reach a higher accuracy than the pre-trained one. From

a different perspective, the Adam optimizer brought 0.5% more accuracy by using the transfer learning technique with EfficientNet-B0 than the SGD optimizer. So, it reached the best test accuracy on the second dataset. Additionally, it can be seen that pre-trained EfficientNet-B1 caused less accuracy compared to the B0 model which can originate from the complexity of the model. Considering that, the number of classes in the new dataset is 10 times less than the original dataset and have a better quality, these perfect results obtained from the application of the models on the second dataset are obvious [2]. Figure 3.8 and Figure 3.9 show the loss and accuracy graphs in which the best test accuracy is reached with both model from the beginning and the pre-trained model, respectively.

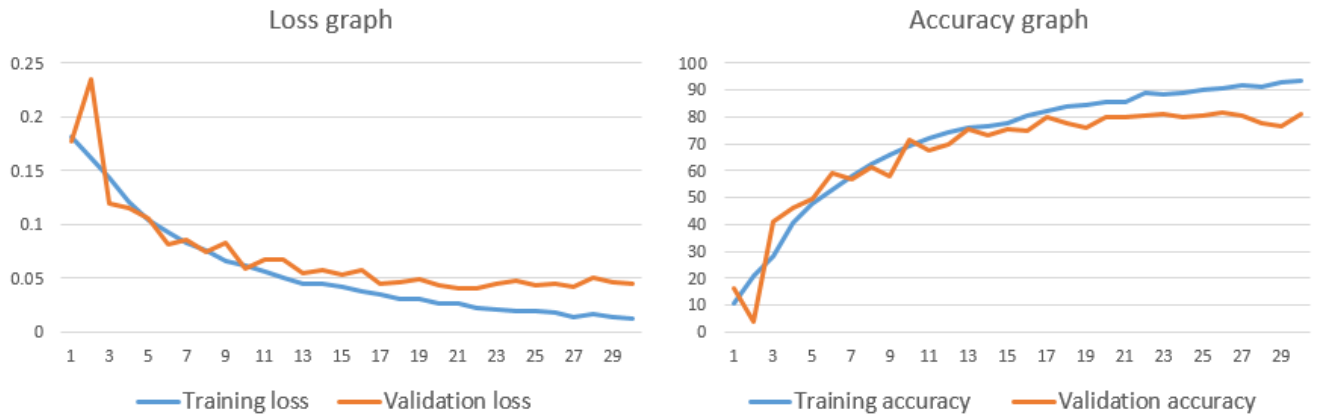


Figure 3.8 EfficientNet-B0 from scratch with SGD (Momentum)

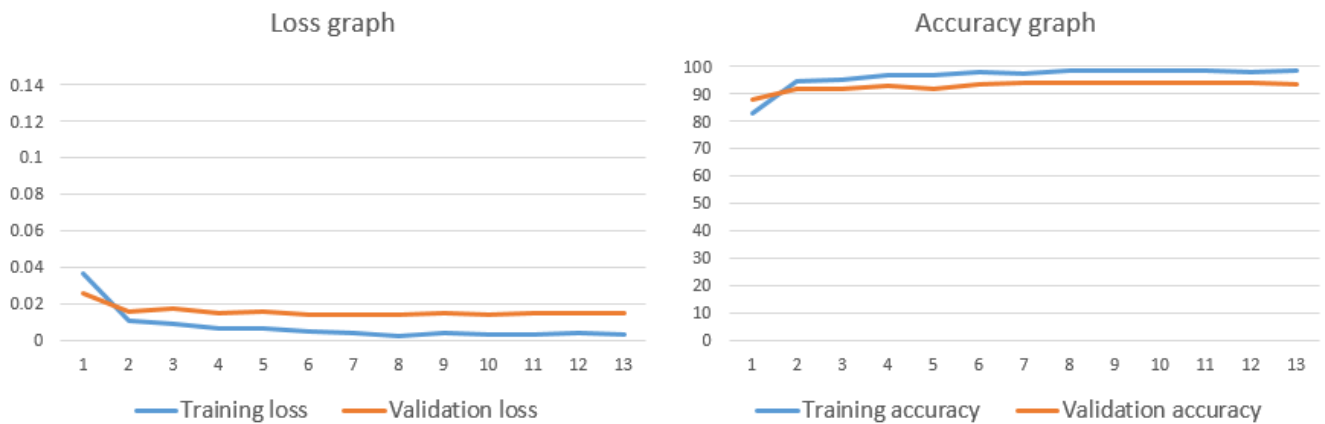


Figure 3.9 Pre-trained EfficientNet-B0 with Adam optimizer

It can be seen from Figure 3.8 that the validation accuracy and loss oscillate during the training procedure and even intersects the training values, however, after the 11th epoch a gap between the values of validation and training occurs and overall generalization happens. Generally, the result can be regarded as a perfect one compared to the initial dataset. Additionally, we can see from Figure 3.9 that the validation accuracy changes around 92% for the pre-trained EfficientNet-B0 from the initial epoch and this point represents how the pre-existed model is better than the model built from scratch.

Conclusion

In conclusion, I concentrated on classifying birds for my project using the EfficientNet architecture with both the pre-trained models of B0 and B1 types, and my own custom implementations of them. Using the B1 pre-trained model with fine-tuning, I was able to get a maximum test accuracy of about 85% using the low-quality initial dataset I worked with. To get the result a pre-processing technique that involved precisely cropping the bird from each image has been used, however, on this first dataset, the code typed from scratch did poorly. On the other side, I obtained a different, higher-quality dataset with fewer labels to address this problem. Thus, the best results were reached when using transfer learning with the EfficientNet-B0 model, with an accuracy of about 95%. On this new dataset, the model from the beginning produced an effective result of roughly 77%. My experiments showed that the initial dataset had significant limitations in terms of quality, affecting the performance of both the pre-trained models and the code developed from scratch. The utilization of transfer learning with the B0 model on a higher quality dataset resulted in superior classification accuracy, representing the importance of dataset quality for achieving better results in bird classification tasks.

References

- [1] Andreia Marini, Jacques Facon and Alessandro L. Koerich, "Bird Species Classification Based on Color Features," *IEEE*, 13-16 October 2013.
- [2] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, Serge Belongie, "Building a Bird Recognition App and Large Scale Dataset with Citizen Scientists: The Fine Print in Fine-Grained Dataset Collection.," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [3] Francis Jesmar Perez Montalbo and Alvin Sarraga Alon, "Empirical Analysis of a Fine-Tuned Deep Convolutional Model in Classifying and Detecting Malaria Parasites from Blood Smears," January 2021.
- [4] Tryan Aditya Putral, Syahidah Izza Rufaida, Jenq-Shiou Leu, "Enhanced skin condition prediction through Machine Learning using dynamic training and testing augmentation," *IEEE Access*, 24 February 2020.
- [5] Mingxing Tan and Quoc V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *36th International Conference on Machine Learning (ICML)*, 2019.
- [6] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Xinyu Xu, Tong Zhang, and Jianyuan Guo, "EfficientNet-eLite: Extremely Lightweight and Efficient CNN Models for Edge Devices by Network Candidate Search," *Signal Processing Systems*, 2021.
- [7] Hossam Mohamed Sarhan, Hesham Ali, Eman Ehab, Sahar Selim and Mustafa Elattar, "Efficient Pipeline for Rapid Detection of Catheters and Tubes in Chest Radiographs," 25 July 2022.
- [8] Sumyung Gang, Ndayishimiye Fabrice, Daewon Chung and Joonjae Lee, "Character recognition of components mounted on printed Circuit Board using Deep Learning," in *10th International Conference on Industrial Technology and Management (ICITM)*, Daegu, 2021.

- [9] Qing Yang, Shukai Duan and Lidan Wang, "Efficient Identification of Apple Leaf Diseases in the Wild," *Agronomy*, 2022.
- [10] Chao Su and Wenjun Wang, "Concrete Cracks Detection Using Convolutional Neural Network Based on Transfer Learning," *Advances in Civil Engineering*, 17 October 2020.
- [11] Yunming Ye, Qingyao Wu, Joshua Zhexue Huang, Michael K. Ng and Xutao Li, "Stratified sampling for feature subspace selection in random forests for high dimensional data," in *Pattern Recognition*, vol. 46, 2013, pp. 769-787.
- [12] Felipe Farias, Teresa Ludermir, Carmelo Bastos-Filho, "Similarity Based Stratified Splitting: an approach to train better classifiers," 13 October 2020.
- [13] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do and Kaori Togashi, "Convolutional neural networks: an overview and application in radiology," 22 June 2018.
- [14] Saad Albawi, Tareq Abed Mohammed and Saad Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, 2017.
- [15] X. Anitha Mary, A. Peniel Winifred Raj, C. Suganthi Evangeline, T. Mary Neebha, Vinoth Babu Kumaravelu, P. Manimegalai, "Multi-class Classification of Gastrointestinal Diseases using Deep Learning Techniques," 21 December 2022.
- [16] R. Shyam, "Convolutional Neural Network and its Architecture," *Computer Technology & Applications*, 21 October 2021.
- [17] Andrinandrasana David Rasamoelina, Fouzia Adjailia and Peter Sincak, "A Review of Activation Function for Artificial Neural Network," in *IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlany, Slovakia, 2020.
- [18] A. A. Fred, "Deep Learning using Rectified Linear Units (ReLU)," March 2018.
- [19] Prajit Ramachandran, Barret Zoph and Quoc V.Le, "Swish: A self-gated activation function," 27 October 2017.

- [20] Brandon Yang, Gabriel Bender, Quoc V. Le, Jiquan Ngiam, "CondConv: Conditionally Parameterized Convolutions for Efficient Inference," 10 April 2020.
- [21] William Mulim, Muhammad Farrel Revikasha, Rivandi, Novita Hanafiah, "Waste classification using EfficientNet-B0," in *IEEE*, Sakhir, Bahrain, 2022.
- [22] Anggi Zhaputri, Mardhiya Hayati and Arif Laksito, "Classification of Brain Tumour MRI images using Efficient Network," *IEEE*, 20 October 2021.
- [23] Sampurna Mandal, Swagatam Biswas, Valentina E.Balas and Rabindra Nath Shaw, "Motion Prediction for Autonomous Vehicles from Lyft Dataset using Deep Learning," in *IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India, 2020.
- [24] Marwa Ben Jabra, Anis Koubaa, Bilel Benjdira, Adel Ammar and Habib Hamam, "COVID-19 Diagnosis in Chest X-Rays Using Deep Learning," *Applied Sciences*, 23 March 2021.
- [25] M.Aqil, M.Azrai ,M.J.Mejaya ,N.A.Subekti, F.Tabri, N.N.Andayani, Rahma Wati, S.Panikkai, S.Suwardi, Z.Bunyamin, E.Roy, M.Muslimin, M.Yasin and E. Prakasa, "Rapid Detection of Hybrid Maize Parental Lines Using Stacking Ensemble Machine Learning," *Advances in Computational Intelligence and Soft Computing (ACISC)*, 26 April 2022.
- [26] "A Study of the Optimization Algorithms in Deep Learning," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, 2020.
- [27] Derya Soydaner, "A Comparison of Optimization Algorithms for Deep Learning," vol. 34, no. 13, February 2020.

Declaration

I hereby declare that this diploma work titled “Classification of Bird Species based on EfficientNet Architecture” is my original effort and experiment unless otherwise acknowledged. I confirm that the findings and ideas used to complete the project are exactly my personal capabilities and all sources in the references have been appropriately cited by myself.

Furthermore, I declare that this thesis has not been submitted to any other educational institution and it has not been plagiarized from any other sources as it accurately represents my own thoughts and conclusions.

The work was done under the supervision of Senior Lecturer Ramil Shukurov, at Baku Higher Oil School, Azerbaijan.

(Bahruz Huseynov)