Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Varun Srivastava

Classification of Fish Species Using Deep Learning Models

Master's thesis in Applied Computer Science
Supervisor: Siamak Khatami
Co-supervisor: Ahmad Hassanpour
May 2023

NTNU
Norwegian University of
Science and Technology

Varun Srivastava

Classification of Fish Species Using Deep Learning Models

Master's thesis in Applied Computer Science
Supervisor: Siamak Khatami
Co-supervisor: Ahmad Hassanpour
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Classification of Fish Species Using Deep Learning Models

Varun Srivastava

May 30, 2023

Abstract

This thesis presents a research project focused on the classification of specific river and freshwater species using computer vision and deep learning techniques. The project aims to contribute to the fields of aquaculture and fish farming by providing accurate species recognition, which plays a crucial role in helping the stakeholders make informed decisions and boost their business. The project investigates several research questions related to mitigating challenges posed by limited power and memory resources on edge computers, highly unbalanced data, and optimizing model performance. The performance of different deep learning architectures (ResNet-50, MobileNet V1, MobileNet V2, and MobileNet V3), when trained on balanced and unbalanced datasets under certain limitations imposed by the edge computer embedded with coral AI, is also discussed. Furthermore, the research examines the trade-offs between model accuracy and resource consumption for various deep learning models when applied to the classification of highly unbalanced classes on edge devices. A set of 38 experiments using an unbalanced and a balanced dataset with architecture setup 1 and architecture setup 2, indicate that MobileNet V1 performed better than the other models, namely ResNet-50, MobileNet V2 and MobileNet V3 on this specific classification task. The study considers scenarios where highly unbalanced data, limited power, and memory resources are common challenges, providing insights to guide the selection and optimization of models in similar contexts. Following that, the obtained results have been discussed regarding reusability and future works, and conclusions are drawn regarding their implications for real-world applications.

Sammendrag

Denne oppgaven presenterer et forskningsprosjekt fokusert på klassifisering av spesifikke elve- og ferskvannsarter ved bruk av datasyn og dyplæringsteknikker. Prosjektet har som mål å bidra til akvakultur og fiskeoppdrett ved å gi nøyaktig artsgjenkjenning, som spiller en avgjørende rolle i å hjelpe interessentene til å ta informerte beslutninger og øke virksomheten deres. Prosjektet undersøker flere forskningsspørsmål knyttet til å redusere utfordringer knyttet til begrensede kraft- og minneressurser på avanserte datamaskiner, svært ubalanserte data og optimalisering av modellytelse. Ytelsen til forskjellige dyplæringsarkitekturer (ResNet-50, MobileNet V1, MobileNet V2 og MobileNet V3), når de trenes på balanserte og ubalanserte datasett under visse begrensninger pålagt av kantdatamaskinen innebygd med coral AI, diskuteres også. Videre undersøker forskningen avveiningene mellom modellnøyaktighet og ressursforbruk for ulike dyplæringsmodeller når de brukes til klassifisering av svært ubalanserte klasser på edge-enheter. Et sett med 38 eksperimenter som bruker et ubalansert og et balansert datasett med arkitekturoppsett 1 og arkitekturoppsett 2, indikerer at MobileNet V1 presterte bedre enn de andre modellene, nemlig ResNet-50, MobileNet V2 og MobileNet V3 på denne spesifikke klassifiseringsoppgaven. Studien tar for seg scenarier der svært ubalanserte data, begrenset kraft og minneressurser er vanlige utfordringer, og gir innsikt for å veilede valg og optimalisering av modeller i lignende sammenhenger. Deretter har de oppnådde resultatene blitt diskutert angående gjenbrukbarhet og fremtidige arbeider, og konklusjoner trekkes angående deres implikasjoner for applikasjoner i den virkelige verden.

Acknowledgements

This thesis is submitted in partial fulfillment of the requirements for the degree of Master in Applied Computer Science at the Department of Computer Science, Norwegian University of Science and Technology (NTNU). The project explores various aspects of the identification of river and freshwater species using computer vision and deep learning techniques, aiming to contribute to the field of computer science.

I would like to express my sincere gratitude to my supervisor, Siamak Khatami, for his invaluable guidance, continuous support, and insightful feedback throughout the entire process. His expertise and dedication have been instrumental in shaping the direction of this thesis and enhancing its quality. Additionally, I extend my appreciation to my co-supervisor, Ahmad Hassanpour, for his valuable input, guidance, and constructive criticism. His expertise in the field has greatly enriched my research experience, and I am grateful for his support. I would also like to express my appreciation to Anuja Vats, for her valuable support and guidance. Furthermore, I would like to acknowledge the contribution of Magnus Rogne Myklebost and Saber Derouiche from our industry partner, Mohn Technology AS, for their collaboration and provision of resources. Their involvement has added a practical perspective to this research, making it more relevant to practical-world applications.

I am grateful to my study program coordinator Christopher Frantz and the faculty members of the Department of Computer Science at NTNU for their expertise and the knowledge they imparted throughout my academic journey. Their dedication to teaching and research has played a significant role in shaping my understanding of computer science principles and methodologies.

Finally, I would like to express my heartfelt appreciation to my family and friends for their unwavering support, encouragement, and understanding during the completion of this thesis. Their love and belief in me have been a constant source of motivation, and I am forever grateful for their presence in my life.

I sincerely hope that this thesis contributes to the existing body of knowledge in the field of computer science and serves as a foundation for future research endeavors.

Abbreviations

List of all abbreviations in alphabetical order:

- **AUC** Area Under the Curve
- **AI** Artificial Intelligence
- **CNN** Convolutional Neural Network
- **DBN** Deep Belief Network
- **DPM** Deformable Parts Model
- **IoT** Internet of Things
- **LSTM** Long Short-Term Memory
- **MVS** Machine Vision Systems
- **MHK** Marine and Hydrokinetic
- **mAP** mean Average Precision
- **NAS** Network Architecture Search
- **NMS** Non-Maximum Suppression
- **ROC** Receiver Operating Characteristic
- **RAS** Recirculating Aquaculture System
- **RNN** Recurrent Neural Network
- **SVM** Support Vector Machine
- **TF** TensorFlow

Declaration

The images of the fish used in this project were captured using a top-view perspective while the fish were being transferred through a channel from one location to another. It is important to emphasize that none of the fish were harmed or subjected to any form of harm or distress for the sole purpose of this experiment. The welfare of the fish was upheld throughout the data collection process, ensuring their well-being and minimizing any potential impact on their natural behavior or environment.

In accordance with the request from our industry partner, Mohn Technology AS, the data utilized for this project is subject to a non-disclosure agreement, which prohibits its public release for a duration of five years.

The GitHub repository with the code can be found in Appendix 6.1 and access can be granted upon request.

Contents

Figures

Tables

Chapter 1

Introduction

The fishing industry represents a significant sector in the global economy, generating substantial revenue and employment opportunities. According to the latest statistics from the World Bank, the annual global fish production reached an estimated 216 million metric tons in 2021 [1]. The estimated value of global production is valued at USD 406 billion where USD 265 billion is contributed by aquaculture. [2]. These figures underscore the economic importance of the fishing industry and highlight the need for innovative solutions to enhance its efficiency and sustainability.

Aquaculture is the practice of cultivating aquatic organisms, including fish, shellfish, and seaweed, in controlled environments such as ponds, tanks, or ocean enclosures. It involves a range of activities such as breeding, hatching, feeding, and harvesting aquatic organisms for commercial purposes. Fish farming is a type of aquaculture that specifically involves the breeding and raising of fish in tanks or enclosures for commercial purposes. Fish farming can be done in freshwater or saltwater environments and can involve a variety of species such as salmon, tilapia, catfish, and trout. The practice of aquaculture has become increasingly important in recent years due to the growing demand for seafood and the need to reduce pressure on wild fish populations [3].

Fish farming, or aquaculture, is significant for several reasons. Firstly, it provides a sustainable source of seafood that can help to reduce pressure on wild fish populations. Secondly, it can contribute to food security and rural livelihoods in many parts of the world. Finally, it has the potential to generate significant economic benefits through the production and sale of fish and other aquatic products. According to a report by the Food and Agriculture Organization (FAO), global aquaculture production was worth approximately USD 250 billion in 2018. Freshwater fish accounted for around 60% of the total, with the popular species being trout, carp, salmon, and a few others [4]. The value of freshwater fish production varies widely depending on the region and species involved. Trout is an important species in the aquaculture industry due to its popularity as a food fish and its high market value. Trout farming is particularly important in regions with cold water resources, such as North America, Europe, and parts of Asia. In addition to being

a valuable food source, trout farming can also provide economic benefits through the sale of live fish for stocking recreational fisheries [3].

Within the realm of trout, the rainbow trout (Oncorhynchus mykiss) stands out as one of the most favored species. Rainbow trout is the leading freshwater-farmed species in Europe, and almost all rainbow trout on the EU market come from aquaculture. In 2018, 156,000 tonnes of rainbow trout were produced in the EU, with more than two-thirds of the production grown in tanks and raceways [5]. Recirculating Aquaculture System (RAS) facilities have increased in recent years, mainly in Denmark. About 10% of rainbow trout is produced in RAS [5]. Rainbow trout is known for its vibrant colors and excellent taste, making it a preferred choice for both commercial and recreational fishing. However, in fish farming, it is essential to separate mature rainbow trout from the rest of the population. Mature rainbow trouts are typically larger and more valuable, commanding higher prices in the market. Efficiently identifying and separating mature rainbow trout from immature rainbow trout is crucial for optimizing fish farming operations and maximizing profitability.

The process of separating fish is typically done using nets or other equipment that allows farmers to capture and move fish from one location to another. In some cases, fish may be sorted by size or species using manual methods such as hand sorting or visual inspection. However, the specific methods used for separating fish can vary depending on the type of fish being farmed and the specific needs of the farmer [3]. Manual separation requires skilled labor, is time-consuming, and often leads to errors and inconsistencies. Moreover, the manual process becomes increasingly challenging as fish farms scale up their operations. To address these challenges, the integration of technology and automation has become a focal point in the fish farming industry. *Artificial intelligence* (AI) has emerged as a transformative tool for species identification and separation.

Underwater cameras have become an increasingly popular tool for monitoring fish abundance and conducting stock assessments. These cameras can capture high-resolution images and videos of fish in their natural habitats, providing valuable data for researchers and fisheries managers. Researchers have developed specialized camera systems that are optimized for underwater use. These systems often include features such as high-quality lenses, adjustable lighting, and advanced image stabilization technology to ensure that clear, high-resolution images can be captured even in challenging conditions. Some underwater camera systems are equipped with AI capabilities by working in conjunction with edge computing devices such as Coral by Google, which can further enhance their capabilities by enabling real-time analysis and decision-making at the edge of the network [6].

AI has been quite successful in several applications of object classification and detection such as automobile air-conditioning leakage detection, classify the type of vehicles, etc. [7]. Therefore AI can also help in identifying fish species by using deep learning algorithms to analyze visual and acoustic data from cameras and imaging sonar. The basic process involves training a deep neural network model on a large dataset of labeled fish images. The model learns to recognize patterns

in the data that correspond to different fish species, allowing it to classify new images or acoustic signals based on the patterns it has learned. Once the model has been trained, it can be used to automatically detect and classify fish species in new images or acoustic signals, providing valuable information for fisheries management and conservation efforts. It is important to note that this process requires a large amount of labeled training data and careful validation to ensure accurate results [8].

However, capturing underwater images to create a comprehensive dataset for training AI models presents its own unique set of obstacles. The aquatic environment, with its varying water conditions, lighting constraints, and unpredictable fish behavior, makes image capture challenging. Overcoming these challenges, a high-quality dataset to train AI models can be used to classify fish species, which is crucial for fish farming and aquaculture. This will help the associated stakeholders to make informed decisions and boost their business, as well as promote a sustainable fish ecosystem.

1.1 Project Description

The initiative of this project is centered around the classification of species in the context of freshwater and river ecosystems. Specifically, the focus is on the rainbow trout specie and the aim is to distinguish between mature and immature fish within this population. The primary objective is to provide valuable support to industrial applications such as aquaculture and fish farming. In these industries, precise species identification plays a crucial role in ensuring optimal management and production processes. Accurately identifying rainbow trout and differentiating between mature and immature specimens, enables better decision-making and more effective implementation of business strategies.

The business partner for this project, Mohn Technology AS, has placed camera systems in key areas throughout a variety of freshwater environments to advance this objective. These camera systems include coral AI edge computers, which let them run AI models right there on the hardware. With the aid of this cutting-edge equipment, pictures, and videos of freshwater and river species is recorded in their native habitats, which serves as the basis for our project dataset. The edge computer's currently deployed AI module distinguishes between fish and non-fish objects with an overall accuracy of over 90% (reported through internal tests by the partner company). By utilizing this feature, a labeled dataset was prepared with the assistance of subject-matter experts who helped in going over photos and videos and labeling the images.

A dataset made up of two distinct categories—immature rainbow trout and mature rainbow trout was assembled and processed. Given the striking similarities in appearance between these two categories of rainbow trout, this was a difficult task. Using the dataset, the aim is to research whether it is possible to classify immature and mature rainbow trout and then compare the performance of various deep learning models which are compatible with the edge computer and analyze

the results in the classification of mature and immature rainbow trout from a naturally unbalanced dataset.

In conclusion, this thesis project aims to harness the potential of computer vision and deep learning to accurately classify the mature and immature trout belonging to the rainbow trout specie, contributing to sustainable practices in industries such as aquaculture and fish farming. The following are the potential research questions that guide this investigation.

1.2 Research Questions

Continuous investigation and exploration of various aspects that contribute to improving model performance and understanding their behavior remain essential in the evolving field of deep learning. This section, which presents a set of research questions that specifically address critical elements related to the classification task within this project, aims to illuminate novel insights and explore uncharted territories in the context of model selection, data pre-processing, and training procedures.

The main objective of this research is to explore the feasibility of utilizing deep learning models to classify two distinct categories of rainbow trout: mature and immature trout. This task poses challenges due to their visually similar appearance. By delving into this research, additional research questions naturally emerge, further enriching our investigation; some of them are listed as follows:

1. Are there any specific data pre-processing techniques that can be employed to mitigate the challenges posed by limited power and memory resources and unbalanced data on an edge computer, and how do these techniques impact the performance of the models?
2. What is the impact of imbalanced class distribution in the dataset on the performance of models, and how does the limited availability of data affect their performance? Additionally, what potential solutions exist to address these challenges and enhance the model's output in such scenarios?
3. Can the performance of the models be further enhanced by employing specific data pre-processing techniques or model modifications that address the challenges associated with limited memory resources and naturally unbalanced data?
4. Which of the deep learning models compatible with the edge computer having limited resources, demonstrates the most effective performance in classifying unbalanced data?
5. What are the trade-offs between overall accuracy and resource consumption for the different deep learning models when applied to the classification of unbalanced classes?
6. What are the implications of the findings from explored models for real-world applications where highly unbalanced data, limited power, and memory

resources are common challenges, and how can these findings guide the selection and optimization of models for similar scenarios?

After clearly defining the objective and extent of the project, this thesis delves into a comprehensive exploration of the subject matter. The subsequent chapters encompass a thorough literature review, an in-depth examination of the methodology employed, an analysis of the results obtained from multiple experiments, thoughtful discussions that address the research questions, and ultimately, a conclusive summary.

Chapter 2

Literature Review

To determine the existence and abundance of different fish species, researchers have utilized a range of approaches for a long time. Some of the strategies used to study fish include traditional methods typically involving manual techniques such as direct observation, netting, or trapping. These methods required human involvement, and data collection was often limited to specific locations and time periods. They were also invasive and could potentially disrupt fish behavior and habitats [9]. In contrast, contemporary methods of fish data capturing have benefited from advancements in technology. These methods often utilize automated or remote sensing techniques, such as underwater cameras, acoustic telemetry, and hydroacoustic systems. These technologies allow for non-invasive, continuous, and widespread data collection. For example, underwater cameras can be deployed for extended periods to capture fish behavior and abundance, while acoustic telemetry can track fish movements using tags and receivers [9].

Optical videos offer a simple and easily understandable way to gather comprehensive information due to the natural human trait of visual perception. Thus, It has been successful to automate similar monitoring applications, such as video surveillance, utilizing computer vision and machine learning. However, there are problems remaining in the field. As an example, in underwater scenarios, there are sudden changes in light, uneven spectrum propagation, low contrast, floating plants as clutter, and changes in visibility brought on by turbidity. Studying coral reef fish species has received a lot of attention recently, especially with the help of open-access databases like *Fish4Knowledge* [10]. This research largely focuses on the identification of coral reef species using deep learning and machine learning to analyze underwater video. With the use of these cutting-edge technologies, researchers have been able to identify and categorize the various fish species that are found in coral reef ecosystems. These studies have improved the knowledge about complex marine habitats by using artificial intelligence to uncover new information about the behavior, distribution, and abundance of coral reef fish. [11].

The concept of deep learning utilizing neural networks has a long history, dating back to 1998 when researchers LeCun et al. [12] introduced this groundbreaking idea. A notable advancement in the field of fish detection and classification was

the introduction of LeNet5, a classifier composed of five layers, which was built upon the innovative concept of emphConvolutional Neural Networks (CNN) by researchers Cui et. al. [13]. In the paper, classifiers are employed to identify and distinguish various fish species based on visual characteristics extracted from images or video data. Deep learning networks, such as CNNs, are designed with multiple layers that progressively learn complex features from input data, enabling them to capture intricate patterns and relationships. This development in deep learning has been witnessed in recent years, mainly due to the significant advancements in computing power and the explosion of big data. The foundation of deep learning lies in the abundance of big data collected from various fields of study. The implementation of deep learning, as highlighted in the paper, specifically in the context of fish detection, showcases the effectiveness of CNNs in automatically learning discriminative features and improving the accuracy of fish species identification. The application of deep learning techniques, enabled by the availability of large datasets and enhanced processing power, has demonstrated substantial performance improvements in various sectors, including robotics systems [13].

To expedite response times and optimize bandwidth usage, edge computing utilizes a distributed computing paradigm that relocates computation and data storage in close proximity to the point of demand. This decentralized approach, as explored by Periola et. al. [14], enables real-time and low-latency data processing in underwater applications by performing data processing and storage at the network edge or within the devices themselves. In the underwater environment, edge computing finds particular significance due to its capability to efficiently transfer and analyze vast amounts of data with minimal latency. It can be applied to process data from underwater sensors and devices used for oceanographic research and marine life monitoring, facilitating tasks such as image recognition and classification. Moreover, edge computing enables real-time monitoring and control of underwater vehicles and equipment, leveraging data from underwater cameras to identify and categorize marine life based on their visual characteristics. This capability is invaluable for studying species distribution, abundance, and ecosystem changes over time, highlighting image classification as a key application of machine learning in underwater settings. [14].

Building upon the advancements in edge computing, researchers have successfully contributed to several important fields by applying deep learning, computer vision, and fish detection techniques. By harnessing the power of these cutting-edge technologies, new possibilities have emerged in underwater research and exploration. Notably, the integration of deep learning and computer vision has revolutionized the analysis of underwater data collected from sensors and devices, enabling a deeper understanding of oceanographic phenomena. The following is a list of applications that have benefited from this technology:

- **Live fish identification**: Fish identification is a crucial step in the development of intelligent breeding management tools or systems since precise and automated live fish identification can provide information for managing future output. Machine vision has the advantage of providing inexpensive,

long-term, nondestructive observation [15]. In order to detect live fish, deep learning is typically employed to determine whether a particular object is a fish. In a time when enormous amounts of visual data can be quickly obtained, deep learning has become a helpful machine vision solution [16]. Deep learning has a fundamental flaw in that it requires a large amount of annotated training data, and gathering and annotating a large number of images takes a lot of time and effort.

- **Species identification**: There are approximately 33000 different species of fish [17]. Species categorization in aquaculture is beneficial for yield forecast, production management, and ecosystem monitoring, according to Alcaraz et al [18]. Usually, various fish species may be distinguished from one another by their visual qualities, such as size, shape, and color. However, due to differences in light intensity, fish movement, and similarity in forms and patterns across several species, effective fish species categorization is challenging. Deep learning algorithms are able to recognize the specific visual characteristics of animals that are resilient to changes in their environment [19].
- **Analysis of behavior**: Since fish are sensitive to environmental changes, they respond to them by changing their behavior in a variety of ways [20]. Additionally, behavior serves as a helpful benchmark indicator for fish welfare and harvesting [21]. By monitoring pertinent behavior, especially unusual behaviors, in a non-destructive way, one can gain an early warning of fish status [22]. The fish activity must be continuously observed in order to understand their status and make decisions about when to catch and feed them [23].
- **Size estimation**: Fish physical characteristics, including length, breadth, weight, and area may be assessed more precisely when machine vision and deep learning are combined. The bulk of reported uses are either semi-supervised or supervised [24]. For example, the Mask R-CNN architecture was used to calculate the sizes of saithe, blue whiting, redfish, Atlantic mackerel, velvet belly lanternshark, Norway pout, Atlantic herring, and European hake. Another indirect method for determining fish size involves using a deep learning model to first determine the fish's head and tail, and then extrapolate the length of the fish from that data. While adding to the workload, this approach is suited for more intricate images [25].
- **Feeding decision-making**: The productivity and breeding expenses of intensive aquaculture are directly impacted by the quantity of fish supplied [26]. Fish growth will be affected by insufficient food, whereas excessive feeding will reduce productivity. Overfeeding also reduces feed conversion efficiency and contaminates the environment with leftover bait. Therefore, enhancing the feeding process can result in substantial economic gains [27].
- **Water quality prediction:** Dissolved oxygen and other indices of water quality are best predicted over time. With the right care, deep learning models like *Long Short-Term Memory* (LSTM), *Deep Belief Networks* (DBN), and

others may effectively mine time sequence data and deliver results that are satisfactory. How to use deep learning models to avoid or lessen the negative effects of uncertainty factors on prediction outcomes will therefore be a crucial area for improvement in jobs requiring the prediction of water quality [28].

In the field of marine research, the utilization of underwater camera systems equipped with edge computing capabilities allows researchers to capture real-time images of marine life and gather valuable data. This process entails employing underwater cameras or other sensors to record images of diverse marine species within their natural environments. Subsequently, the collected images are annotated with relevant species information. Once the dataset is prepared and labeled, researchers employ it to train a neural network model. This involves providing the model with a set of training images along with their corresponding labels and adjusting the model's parameters to facilitate the recognition of species-specific patterns and features. Finally, the trained neural network model can be used for on-the-fly image classification, enabling efficient and automated species identification [14].

Fish4Knowledge is a program that receives funding from the European Union Seventh Framework [29]. This initiative utilizes ten underwater cameras to capture live video feeds, serving as a testbed for exploring techniques applicable to multiple video stream acquisition, storage, analysis, and querying. As a result, the wealth of information gathered through this project is compiled into a comprehensive public database spanning two years. This database includes video summaries showcasing the various fish species observed, accompanied by relevant characteristics. To enhance accessibility and usability, the Fish4Knowledge project focuses on developing professional web-based interfaces. These interfaces provide marine researchers with unparalleled access to both current and archived films, as well as previously extracted data. By utilizing these interfaces, researchers can delve into the vast collection of videos and associated information to facilitate their studies and investigations [29]. Some of the projects which have used this dataset to develop their solutions are:

- In the conducted study by Choi [30], the main objective was to address the challenge of fish identification in underwater videos using deep CNNs. The unique complexity of underwater video images necessitated the inclusion of pre-processing and post-processing steps. To tackle these challenges, Choi combined a foreground detection approach with selective search techniques to detect potential fish regions within the videos accurately. By effectively identifying these candidate windows, subsequent classification and identification steps could be carried out more efficiently.
  During the training phase, a CNN model was employed to classify various fish species. However, due to the limited availability of labeled underwater video data specifically for fish classification, the CNN model was trained using data from other generic object classification tasks. This scarcity of

labeled data led to the adoption of transfer learning, which involves leveraging pre-trained models or learned representations from a source domain and adapting them to a target domain with limited labeled data.
The utilization of transfer learning facilitated efficient knowledge transfer, reducing the reliance on extensive labeled data in the target domain and expediting the learning process [31]. This approach allowed the network to leverage existing knowledge and generalize it to the task of fish species identification, enhancing the overall performance of the model.
The final identification results were obtained by combining the outputs from the CNN classification findings with further enhancements. This
post-processing step aimed to refine and improve the accuracy of the classification results, ensuring reliable and precise identification of fish species within the underwater videos. Overall, Choi's research highlighted the intricacies of fish identification in underwater video compared to standard picture classification tasks. The integration of foreground detection, selective search, and deep CNNs demonstrated the complexity and importance of addressing pre-processing and post-processing stages to achieve robust fish species identification in the challenging underwater environment.

- Li et al. [32] sought to leverage the exceptional detection accuracy of Fast R-CNN to develop an efficient and accurate fish detection and recognition system for underwater images. Fast R-CNN is a framework for object recognition tasks and the architecture enables accurate localization and classification of objects in images, making it suitable for tasks like fish detection and recognition in underwater images [33].
  The primary objective of research by Li et. al. was to contribute to the development of automated fish identification systems that could assist marine researchers in estimating fish numbers, and abundance, and gaining a deeper understanding of marine geography and biology. By utilizing the Fast R-CNN framework, Li et al. aimed to achieve both high detection accuracy and efficient processing speeds, addressing the need for real-time or near-real-time analysis of underwater images. The Fast R-CNN architecture combines a region proposal network with a CNN for object detection and recognition, enabling accurate localization and classification of fish species within the images.
  To evaluate the performance of their system, Li et al. conducted extensive experiments. The results demonstrated that their proposed detection system exhibited promising performance, with a higher *mean average precision* (mAP) of 81.4% as compared to other existing methods, such as Deformable Parts Models (DPM) with mAP of 70.2%. Additionally, the system showed the potential to achieve faster processing speeds than the conventional R-CNN approach. This research contributes to the ongoing efforts to develop advanced automated tools that aid in the monitoring and conservation of aquatic environments.
- Zhang et al. [34] addressed the pressing need for an automated segment-

ation technique to generate baseline annotations of fish areas, facilitating the estimation of fish abundance in underwater environments. To tackle this challenge, they proposed an unsupervised underwater fish detection methodology that leveraged the power of deep learning technology. A key focus of their work was to develop a method that eliminated the requirement for manual annotation and allowed for real-time online learning. To achieve these objectives, Zhang et al. first optimized the algorithm for speed. They then devised a multi-step approach that incorporated motion flow segmentation and selective search to create candidate regions for fish detection. By utilizing motion flow segmentation, they aimed to distinguish moving objects, such as fish, from the background, improving the accuracy of subsequent detection steps.

The selective search was employed to generate region proposals, which reduced the likelihood of incorrect labeling during the training phase. These region proposals were combined with the results of motion segmentation and fed into a CNN for detection. This fusion of region proposals and motion segmentation yielded superior performance compared to using the separate components independently. To refine the detection results, Zhang et al. implemented a modified *non-maximum suppression* (NMS) algorithm. This technique effectively reduced the number of overlapping windows, enhancing the localization precision of detected fish instances. The modified NMS algorithm optimized the suppression process to ensure that complete fish objects were accurately identified while minimizing the presence of redundant or overlapping windows. By developing this unsupervised underwater fish detection method, Zhang et al. offered a valuable solution for automatically segmenting fish areas without relying on manual annotation. Their approach, which incorporated motion flow segmentation, selective search, and CNN-based detection with modified NMS, demonstrated improved performance and efficiency. This research contributes to the development of advanced techniques for fish identification and abundance estimation in underwater environments.

- In their work, Rathi et al. [35] employed CNNs as a central component of their proposed technique, which aimed to enhance the efficiency and effectiveness of fish species classification, particularly when dealing with large datasets. By leveraging CNNs, their approach facilitated streamlined and robust operations for fish classification tasks. The process began with the pre-processing of underwater images, which involved several techniques such as Gaussian Blurring, Morphological Operations, Otsu's Thresholding, and Pyramid Mean Shifting. These pre-processing steps aimed to enhance the quality and clarity of the images, reducing noise and enhancing relevant features necessary for accurate classification.

  Following the pre-processing stage, the pre-processed images were fed into a CNN model for classification. The CNN architecture enabled the extraction of intricate features and patterns from the images, facilitating the identific-

ation and classification of different fish species. By leveraging the power of deep learning and CNNs, Rathi et al. achieved remarkable accuracy in their approach. According to their experimental findings, the suggested technique achieved an impressive accuracy rate of 96.29% in classifying fish species. This accuracy surpassed the performance of other existing methods commonly employed for this purpose. The superiority of their proposed approach highlighted the efficacy and potential of utilizing CNNs in underwater fish classification tasks.

By demonstrating the effectiveness and accuracy of their technique, Rathi et al. contributed to advancing the field of underwater fish species classification. Their approach, which incorporated CNNs and a series of preprocessing steps, showcased the capability of deep learning algorithms to handle large datasets and achieve superior classification results.

- In their study, Mandal et al. [36] highlighted the significant advantages of utilizing remote underwater video feeds for autonomous assessment of fish and species abundance. They emphasized that this approach offers substantial potential in terms of time and cost savings compared to traditional methods. To harness the full potential of this technology, the researchers employed an end-to-end deep learning technique to analyze the video streams and extract the necessary information for evaluation. Mandal et al. conducted a series of tests using various deep-learning models and performed a comprehensive analysis of their performance. The objective was to assess the effectiveness of these models in accurately identifying and quantifying a diverse range of marine species from the video feeds.

  The results were highly encouraging. The researchers achieved an impressive mAP of 82.4% across a substantial number of marine species. This mAP value indicated the overall accuracy and reliability of the deep learning models in successfully detecting and classifying different species present in the underwater videos. The findings of this study demonstrate the potential of employing end-to-end deep learning techniques for autonomous fish and species abundance assessment using remote underwater video feeds. By leveraging the capabilities of deep learning models, Mandal et al. showcased the feasibility of extracting valuable information from video streams and achieving high levels of accuracy in species identification and abundance estimation. This research contributes to advancing the field of underwater monitoring and conservation by providing an efficient and effective approach to assessing marine biodiversity.
- Marini et al. [24] introduced a novel and efficient video-automated approach to accurately track and estimate fluctuations in fish abundance under challenging real-world conditions, without the need to differentiate between various fish species. To achieve this, they developed a unique 'generic supervised machine learning' framework for fish recognition in images based on their content. This framework was extensively tested across a range of lighting scenarios, and variations in water turbidity, and even accounted

for biofouling-induced changes on the camera housing. To ensure the robustness and generalization performance of their image classifier, Marini et al. employed a K-fold Cross-Validation framework. This approach enabled them to select the most relevant image attributes for fish recognition and develop an automated image classifier that could effectively handle varying environmental conditions.

The proposed video-automated approach allowed for continuous monitoring of fish abundance, providing valuable insights into population dynamics and fluctuations. By utilizing the supervised machine learning framework, Marini et al. effectively addressed the challenges associated with fish recognition in diverse underwater environments. This approach not only eliminated the need for species differentiation but also demonstrated its effectiveness in accommodating variations in lighting, water turbidity, and biofouling. The research conducted by Marini et al. offered a robust and versatile solution for accurately tracking fish abundance in real-world conditions. The use of a generic supervised machine learning framework and the K-fold Cross-Validation technique ensured accurate and reliable fish recognition and counting, regardless of environmental variations. This study contributes to the development of automated approaches for underwater image analysis and provides a valuable tool for ecological research and fisheries management.

According to Xu and Matzner's study on underwater fish detection for water power applications [11], previous research in underwater fish optical analysis primarily focused on classifying coral fish in well-lit shallow seas or deep waters where fish populations were abundant and easily observable. However, there is a notable gap in the literature when it comes to detecting fish in more challenging underwater environments, particularly in the vicinity of *marine and hydrokinetic* (MHK) energy projects or river hydropower projects. These areas often present difficult conditions, such as high turbidity, fast water currents, and murky waters, where fish tend to have less distinct colors and are barely perceptible. Addressing this research gap is crucial due to the increasing interest in MHK and river hydropower projects, where assessing the potential impact on fish populations is vital. Xu and Matzner emphasize the need for accurate fish detection methods that can overcome the challenges of low visibility and dull-colored fish. By developing techniques based on deep learning algorithms, they aim to detect fish in underwater environments with improved accuracy and efficiency.

On the other hand, Nair et al. [37] shed light on the challenges associated with underwater fish species recognition. They highlight the difficulty of collecting representative samples for underwater photos, as it involves dealing with poor image quality and uncontrolled environmental factors. Furthermore, existing feature extraction methods often require continuous human supervision, which can be time-consuming and labor-intensive. Manual examination of fish images and videos by marine biologists is a common practice to extract essential information, further adding to the overall effort required. To address these challenges, Nair et

al. propose the development of automated methods for fish species recognition, reducing the reliance on manual intervention. Their research aims to explore innovative approaches and algorithms to streamline the process of fish analysis in underwater imagery.

The importance of underwater image datasets in the field of image processing is emphasized by Raveendran et al. [38]. The available datasets such as Fish-4Knowledge [10], TURBID [39] and the more recent ones MOUSS, AFSC, MBARI, NWFC, and RUIE [40], serve as invaluable resources for advancing image processing methods in underwater environments. They enable researchers to develop and evaluate novel algorithms, improving fish detection, species recognition, and other related tasks. With access to comprehensive underwater image datasets, researchers can devise more sophisticated techniques that minimize the need for human oversight, resulting in more efficient analysis of underwater imagery.

Another new technology known as *Machine Vision Systems* (MVS) has become an indispensable tool in aquaculture monitoring, offering numerous advantages such as non-intrusiveness, objectivity, and repeatability. Machine vision systems refer to computer-based system that utilizes image processing and analysis techniques to automatically interpret and understand visual information. Specifically, in the context of aquaculture and fish-related applications, a machine vision system is designed to analyze images or video footage captured in aquacultural settings, such as fish farms or underwater environments. However, traditional image processing techniques often struggle to yield satisfactory results in complex underwater environments. The intricate nature of underwater conditions, such as variations in lighting, water turbidity, and biofouling, presents challenges for accurate data processing using conventional methods.

To overcome these challenges, deep learning algorithms have gained significant attention in the field of underwater image processing. The remarkable feature extraction capabilities of deep learning algorithms make them highly suitable for handling the complexities of underwater imagery. By automatically learning and identifying relevant features from vast amounts of training data, deep learning algorithms can effectively analyze and interpret underwater images. The fusion of deep learning algorithms with machine vision techniques is crucial for advancing the automated monitoring of aquaculture systems. This integration enables the development of robust and efficient systems capable of performing tasks such as fish classification, detection, counting, behavior identification, and biomass estimation. The application of deep learning in these areas demonstrates the interdisciplinary nature of its impact on aquacultural machine vision systems [41]. In the context of aquaculture, the acquisition and processing of image data can be time-consuming. To address this, motion detection algorithms are utilized to extract relevant frames from video footage. However, manual processing is still required to analyze the extracted frames. To streamline this process and automate data extraction from images, machine vision techniques based on object recognition have proven instrumental. By accurately separating the subject of interest from the background, these techniques enable the application of computationally effi-

cient approaches such as optical flow analysis and segmentation based on pixel characteristics. This automated data extraction not only improves efficiency but also enhances the accuracy of aquacultural data analysis [42].

With the use of the *Internet of Things* (IoT), big data, cloud computing, artificial intelligence, and other cutting-edge information technologies, a new scientific field called *Smart Fish Farming* seeks to increase resource efficiency and promote the expansion of a sustainably managed aquaculture industry. Additionally, real-time data collection, quantitative decision-making, intelligent control, precise investment, and individualized service have all contributed to the formation of a new fisheries production mode. Data and information are the cornerstones of smart fish farming. The aggregation and sophisticated analyses of all or some of the data will lead to the ability to make decisions with a scientific basis. Smart fish farming generates a substantial volume of data, posing challenges due to its diverse sources, formats, and complexity. The data originates from various sources, including human inputs, environmental factors, tools, fish, and the breeding process. Furthermore, it exists in different formats, encompassing text, images, and audio. The complexity of the data further stems from variations in species, cultural eras, and styles. The aforementioned high-dimensional, nonlinear, and massive data present a very challenging problem [28].

CNNs and *Recurrent Neural Networks* (RNNs) are widely recognized as popular deep learning models that are extensively used in edge computing. Edge computing involves deploying a network of interconnected computing nodes in close proximity to end devices. This approach proves to be highly advantageous in fulfilling the demanding computational needs and low-latency requirements of deep learning tasks on edge devices. Moreover, edge computing offers numerous additional benefits, including enhanced privacy, improved bandwidth efficiency, and increased scalability. By leveraging CNNs and RNNs in edge computing, significant advancements have been made in processing data directly on edge devices. CNNs are particularly effective for image and video analysis tasks, as they excel in extracting meaningful features from visual data. RNNs, on the other hand, excel in sequential data processing, making them ideal for applications involving speech recognition, natural language processing, and time-series analysis.[43].

In order to effectively identify and categorize various fish species, it is necessary to automatically extract characteristics from data. This is where machine learning and deep learning play a significant and important role in the categorization of fish. This is crucial in aquaculture, where the capacity to recognize fish species fast and precisely can boost productivity, cut costs, and guarantee the quality of the finished product. Other applications in aquaculture, including as behavior analysis, feeding choices, size or biomass estimation, and water quality prediction, can also be done using machine learning/deep learning in addition to fish classification [44]. Aquaculture professionals may improve results by using these tools to better understand their operations and make data-driven decisions [44]. Large datasets of fish photos can be used to train machine learning algorithms to automatically identify various species based on their distinctive characteristics,

such as color patterns, body shapes, and fin structures. Machine learning and deep learning can assist fish farms in optimizing their feeding tactics by estimating the ideal amount of feed needed for each fish depending on their size and weight, in addition to increasing the accuracy of fish classification. As a result, waste can be reduced and aquaculture operations can operate more effectively overall [45]. Researchers can create models that can precisely classify various species based on their physical traits by analyzing fish photos with machine learning algorithms. This can make it simpler for farmers to monitor the expansion and growth of their fish populations and to spot any potential health concerns or environmental issues that might be harming them [46]. Deep learning techniques such as CNNs have been used to classify different species of fish based on images captured by underwater cameras and the following are some work related to deep learning and smart fish farming that finds fish classification is one of the most popular fields of application [44]:

- Fish species recognition using deep learning techniques by Sarker et. al. [47] used a CNN to classify six different species of fish based on images captured by an underwater camera system. The study was conducted in Bangladesh and used a dataset of 1,200 images of six different fish species commonly found in the region. The authors trained a CNN using the dataset and achieved an accuracy rate of over 90% for all six species.
- Fish classification using a deep convolutional neural network with transfer learning by Kwon et. al. [48] is another study that used a combination of CNNs and *Support Vector Machines* (SVMs) to classify four different species of fish based on images captured by an underwater camera system. The study was conducted in South Korea and used a dataset of 1,000 images of four different fish species commonly found in the region. The authors trained a CNN using transfer learning and then used an SVM to classify the images based on the features extracted by the CNN. The combined approach achieved an accuracy rate of over 95%.
- Fish detection and species classification in underwater environments using deep learning with temporal information by Jalal et. al. [49] is a study where the authors used a dataset of underwater fish images captured by an underwater camera system. The authors used a deep learning model that combined a CNN with an LSTM to capture temporal information. The authors achieved an accuracy rate of 96.7% for fish detection and 92.3% for fish species classification using their proposed method.
- A novel deep learning approach for fish species recognition based on convolutional neural network and softmax regression by Sun et. al. [50] also used a dataset of fish images captured by an underwater camera system and the authors used a CNN model combined with softmax regression. The authors achieved an accuracy rate of 96% for fish species recognition using their proposed method and also noted the challenges in underwater fish recognition including variations in lighting conditions and water quality.
- Image-based monitoring of jellyfish using deep learning architecture by Kim

et. al. [51] used a dataset of jellyfish images captured by an underwater camera system and the authors used a CNN model combined with transfer learning. The authors achieved an accuracy rate of 95.5% for jellyfish detection using their proposed method and noted that challenges in jellyfish recognition include variations in jellyfish appearance due to changes in lighting conditions and water quality.

- Fish species recognition using deep learning techniques by Hossain et al. [52] used a dataset of 10,000 images of 10 different fish species, which were captured using underwater cameras in natural environments. They used a CNN model which consisted of several layers, including convolutional layers, pooling layers, and fully connected layers. The authors also used data augmentation techniques to increase the size of their dataset and improve the performance of their model and achieved an accuracy rate of 98% for fish species recognition. They also noted various challenges such as some fish species were more difficult to classify than others due to variations in their physical characteristics such as color and shape and limitations associated with using underwater cameras to capture images of fish, such as poor lighting conditions or low image quality in some cases.
- Fish biomass estimation using machine learning techniques by Singh et al. [53] collected a dataset of 1,200 images by capturing images of fish in aquaculture ponds using underwater cameras. They used a machine learning algorithm called Random Forest to estimate the biomass of fish in aquaculture ponds and achieved an accuracy rate of 95%. The authors noted that there were some challenges associated with this task, such as variations in lighting conditions and image quality, which can affect the accuracy of the biomass estimates.
- Fish recognition using convolutional neural networks by Al-Ali et al. [54] used a deep learning approach to recognize and classify fish species based on images captured using underwater cameras. The authors used a CNN model to classify the fish images and also used data augmentation techniques to increase the size of their dataset and improve the performance of their model. They achieved an accuracy rate of 92% by using a dataset of 1,000 images of 10 different fish species. Variations in lighting conditions and image quality were a challenge noted which can affect the accuracy of fish recognition models.

The studies mentioned above have focused on underwater images captured by camera systems and have achieved notable success in achieving high accuracy rates for species recognition. However, there are still significant problems and limitations that need to be addressed. One common challenge in fish species recognition is the variations in lighting conditions and water quality, which can affect the appearance of fish and hinder accurate identification. Additionally, the physical characteristics of different fish species, such as color and shape, pose challenges in classification, as some species may be more difficult to distinguish than others. Another important aspect is the amount of data collected and the

size of the labeled dataset. Having a large, labeled dataset is important to have a stable and reliable machine-learning model that will identify fish species accurately. However, acquiring and labeling a large dataset can be time-consuming and resource-intensive. Furthermore, the existing studies have mainly focused on specific regions or a limited number of fish species, which may not cover the diverse range of freshwater and river species found globally.

Rainbow trout holds significant importance in the field of aquaculture due to its economic value, high demand, and adaptability to various environmental conditions. As one of the most widely cultivated freshwater fish species, rainbow trout plays a crucial role in global aquaculture production. Rainbow trout is also the main freshwater species farmed at the European Union level, both in terms of volume and value [5]. Rainbow trout dominates global trout production, contributing to approximately 97%-98% of the total production each year between 2010 and 2019. In 2019 alone, the production of rainbow trout reached 916,365 tonnes. In comparison, other trout species such as sea trout accounted for only 0.3% of the production, equivalent to 3,252 tonnes, while brook trout constituted 0.2% with 1,702 tonnes. [55].

Given the prominence of rainbow trout in aquaculture, this project focuses on the specific challenge of classifying mature trout from immature trout using deep learning techniques. This distinction is crucial for the effective management and monitoring of fish stocks, as it enables farmers to determine the optimal harvesting time and ensures the sustainable growth of trout populations. However, accurately differentiating mature trout from immature trout based on visual observations alone can be challenging, particularly as they share many physical similarities.

By leveraging the power of deep learning algorithms, the aim is to investigate the performance of the deep learning models in classifying mature trout based on subtle differences in morphology, coloration, and other visual cues. This approach will involve training the model using a comprehensive dataset comprising images of both mature and immature trout, annotated with appropriate labels - **mature** for mature trout and **trout** for immature trout. Through this research, the aim is to contribute to the advancement of aquaculture practices by providing fish farmers and industry stakeholders with an automated and reliable tool for trout classification. By harnessing the potential of deep learning and computer vision, the project also seeks to streamline the process of identifying mature trout, facilitating more efficient management strategies, optimized harvesting operations, and sustainable growth in rainbow trout aquaculture.

Chapter 3

Method

In this chapter, a detailed account has been presented of the research design, data collection, analysis, and the steps taken to ensure the validity and reliability of the study. The aim of this chapter is to provide the readers with a clear understanding of the research methodology and the rationale behind the chosen approach with respect to the deep learning models chosen and related work. The following sections will detail the process of classifying fish species using deep learning, including the data collection and pre-processing, model architecture, training, and evaluation, and the steps taken to ensure the accuracy and robustness of the results.

3.1 Data Collection

As discussed earlier, the thesis aims at the classification of fish species which are found in freshwater like rivers and lakes. Collecting data for species that are in a river is generally a very difficult task considering the water flow, depth of the stream, and other natural conditions such as the growth of algae, weeds, lighting conditions, etc. The industrial partner developed an autonomous camera system that can be positioned underwater in Norway's rivers and fjords specifically for this use. In order to reduce the stress on the users, the system combines a combination of effective and powerful machine vision algorithms. The hardware is able to withstand adverse situations and lengthy service intervals because it is durable, light, and compact. The camera system has an embedded local computer with an AI-specific accelerator which has the capacity to process over 1 million images per day. The AI accelerator by Coral features on-device capabilities that enable the development of productive, discreet, quick, and offline products. Thus, the fish detection system that is installed locally on the system allows the accelerator to record videos when there is a fish detection by the algorithm. These videos are further processed into frames which results in images of the fish. No specific species is targeted during the detection of the fish by the algorithm. Therefore a large number of fish images is collected at the central repository.

These images are further identified and approved by the specialists in this field and labeled correctly as fish or no-fish by the business partner. Since there is a large data available for fish, it is possible to further label the fish according to the specie they belong to using labelbox. Labelbox (`https://www.labelbox.com/`) is a platform where teams can collaboratively label data for machine learning applications for collaborative data labeling. It offers a web-based interface for labeling text, polygonal segmentation masks, object bounding boxes, pictures, videos, and other sorts of data. Once the data is labeled, it can be used as a dataset for various machine-learning applications such as object detection, object classification, etc. For the purpose of this project, the species that was identified was rainbow trout. Within the rainbow trout specie, there are two categories of interest for fish farmers and aquaculture which are immature trout and mature trout. Mature trout are simply adult trout that have reached sexual maturity and are capable of reproducing. The table 3.1 shows the distribution of data that was provided for the project.

Trout Category	**Count**	**Distribution**
immature trout	25270	≈84%
mature trout	4696	≈15%
damaged	58	≈<1%

Table 3.1: Data distribution

(a) Dataset image 1 **(b)** Dataset image 2

Figure 3.1: Sample images from dataset

Since the aim of this project was to assess the feasibility of classifying mature and immature trout, considering the number of images in the 'damaged' category very low, only the immature trout and mature trout categories were chosen to proceed with. The total labeled images that were available for the project were 29966 images. The data is not balanced, which has to be considered when working with any deep-learning model for classification. It is important to note that

all the images are taken in the same environment with respect to light and background conditions. These images can contain multiple or single fish belonging to the mature or immature trout category. A couple of sample images that are from the labeled dataset are presented in Fig. 3.1.

The sections further will explain in detail how this data has been processed and prepared to be used in training models for the classification of immature trout and mature trout.

3.2 Data Pre-processing

The data that was available for two labeled categories **immature trout** (labeled as **trout**) and **mature trout** (which is labeled as **mature**) are images with single or multiple fish in the frame. Fig. 3.2 presents a sample of what the fish images with detection frames look like. [1]

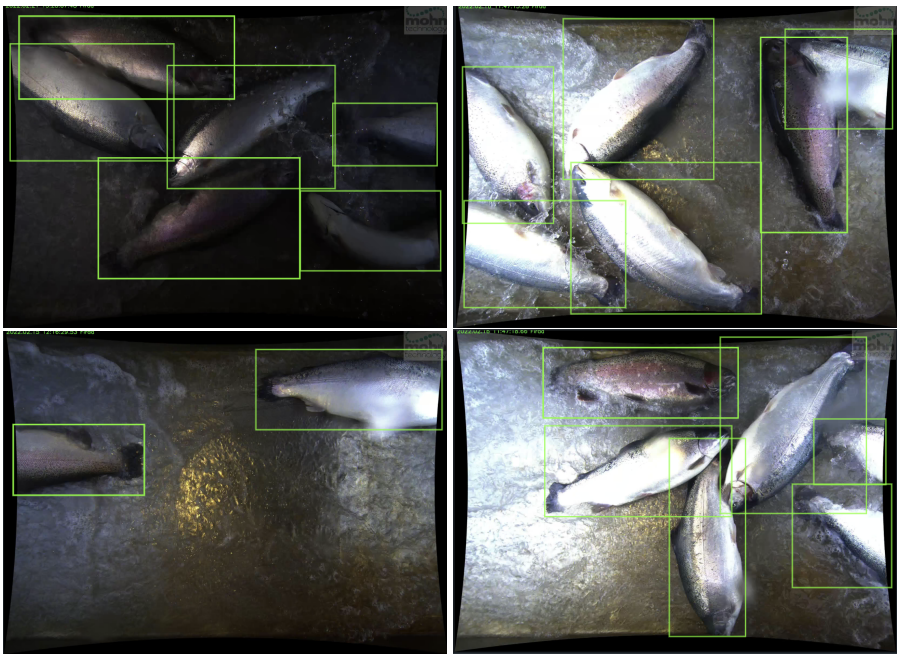


Figure 3.2: Sample images with detection boxes

The data which was available at labelbox, was downloaded using API support. Using the project id, which is related to the dataset, all the images that are labeled were downloaded. Each image also has the associated annotation files in Pascal VOC format. These files contain some relevant information about the related images such as bounding box coordinates for the detection and the associated class label. In this case, the labels in the discussion are **trout** and **mature**. Since the

[1] Throughout this thesis, the terms **trout** and **mature** are also used as a shorthand for **immature trout** and **mature trout** respectively for brevity and ease of reference.

aim is to look forward to a classification algorithm, it is important to have the data sorted based on the class so that it can be prepared for training, validation, and test sets. To prepare the dataset, a Python script was written to loop through the Pascal VOC file which has the respective filename for the image, loop through the labels in each image and create a cutout of each detection using the bounding box values. Each detection is then saved to the respective directories which are labeled trout and mature. In the end, a dataset including 25270 images of trout and 4696 images of the mature category has been achieved. With a digital cutout of the target class from the detection images, a more precise image of the class that is required to train the algorithm was obtained. In this way, the extra 'noise' in the original images such as the background and the empty spots can be ignored, and a more concrete dataset that has only the images, relevant to the class, is acquired.

A sample of images from the trout class is presented in Fig. 3.3.

Figure 3.3: Sample images of immature trout

A sample of images from the mature class is presented in Fig. 3.4.

Figure 3.4: Sample images of mature trout

The two categories of fish have quite a lot of similarities in their size and dimensions, as can also be seen in the images. The mature category has a slight red tinge on the body. This may or may not be a very significant differentiating factor, and the lighting condition in the images can also affect how easy or difficult it is manually to classify an image as mature or not.

A few samples of images from the mature category, which has quite a lot of variation in the color shade as compared to the trout category are presented in figure 3.5

(a) Lighter shade **(b)** Darker shade

Figure 3.5: Color variations in mature trout

Since the two categories of fish in the discussion are not very different from each other, some types of image pre-processing techniques such as gray scaling of images (which will convert the images to grayscale), Gaussian blurring (which is used to blur the image and reduce noise) and histogram equalization are not

applied to the dataset because these techniques can alter the color balance of the image [56].

TensorFlow (TF) is a powerful open-source machine learning framework that provides a wide range of libraries and tools. It enables efficient development and deployment of deep learning models, offering extensive support for tasks like data manipulation, neural network design, and model optimization [57]. The project utilizes TensorFlow 2, which is one of the latest versions of the framework known for its simplicity, flexibility, and improved performance.

When the data is loaded as a dataset in the code, it also undergoes several pre-processing. The `tf.keras.preprocessing.image_dataset_from_directory` function creates a `tf.data.Dataset` object from image files stored in a directory. This function performs several pre-processing steps on the images, including:

- Resizing the images to the specified `image_size` which is (224,224) and is the required input image size for the compatible models which is available at the website (`https://coral.ai/models/image-classification/`) [58].
- Batching the images into batches of the specified `batch_size`
- Shuffling the batches of images randomly
- The `seed` parameter in the function sets the random seed for shuffling the batches of images.

Once the data is loaded into the memory, it is also scaled from the range of [0, 255] to [0, 1]. This is done so that the input data is normalized to a small range of values, typically [0, 1] [59].

These pre-processing steps are performed to prepare the data for training a model. In this light, resizing the images ensures that they all have the same dimensions, which is necessary for feeding them into a neural network. Besides, converting the images to floating-point tensors and scaling their pixel values to the range [0, 1] is also highly recommended for compatibility with neural network models [59]. In addition, batching the images into smaller groups allows the model to process them more efficiently during training [60]. The last but not the least, shuffling the batches randomly helps to ensure that the model sees a diverse set of images during each epoch of training and that all weight updating is done sporadically [61]. In the end, applying a lambda function using the `map` method, the resulting `tf.data.Dataset` object contains the scaled images and their corresponding integer labels (0 for mature and 1 for trout), which can be used for training the model.

The next step in the pre-processing is preparing the dataset for the training of the model. For the classification task, a train, validation and test set is required. The training dataset will be used to train the model while the validation dataset is used to assess the performance and generalization of the model and make relevant changes for improvement if required. The test dataset will be used to evaluate the final performance and generalization of the model. All three datasets are independent of each other to avoid any bias. The splitting of the data into training, validation, and test sets is done using the `tf.data.Dataset` API in TensorFlow. The

length of the entire dataset data is used to compute the number of samples to allocate to each set. Given that, the dataset is split into 70%, 20%, and 10% of the samples as the training set, validation set, and test set, respectively. The `take` method is used to extract the first `train_size` number of samples from the dataset data, which becomes the training set. Then, the `skip` method is used to skip over the training samples, and then the `take` method is used again to extract the next `val_size` number of samples from the remaining data, which becomes the validation set. Finally, the `skip` and `take` methods are used again to extract the remaining `test_size` number of samples, which becomes the test set.

3.3 Data Augmentation

As discussed in the section above, there is a high imbalance in the number of images for trout and mature labels. The images for the trout category are almost 6 times more than the images for the mature category. There should be a balance in the dataset so that the model is not biased toward the majority class and poorly toward the minority class. The performance of the model will be affected and which is why it is important to have data augmentation because this has been one of the proven ways to improve the accuracy of models dealing with classification tasks [62].

For the project, two methods were used to balance the data set. One of the ways was to balance the majority class with the minority class where the trout class, which has 25570 images, was selected, and using a Python script, randomly 5000 files were selected and moved to a new directory. The results from this distribution of data will be discussed in the further sections. The other method was to balance the minority class with the majority class by applying data augmentation on the 4696 images that are available in the mature class, increasing the data for this class to 6 times and coming closer to the count for the trout class. To artificially increase the amount of training data available by generating modified versions of the original data, the Keras `ImageDataGenerator` class was used. The different parameters defined to generate the images are [63]:

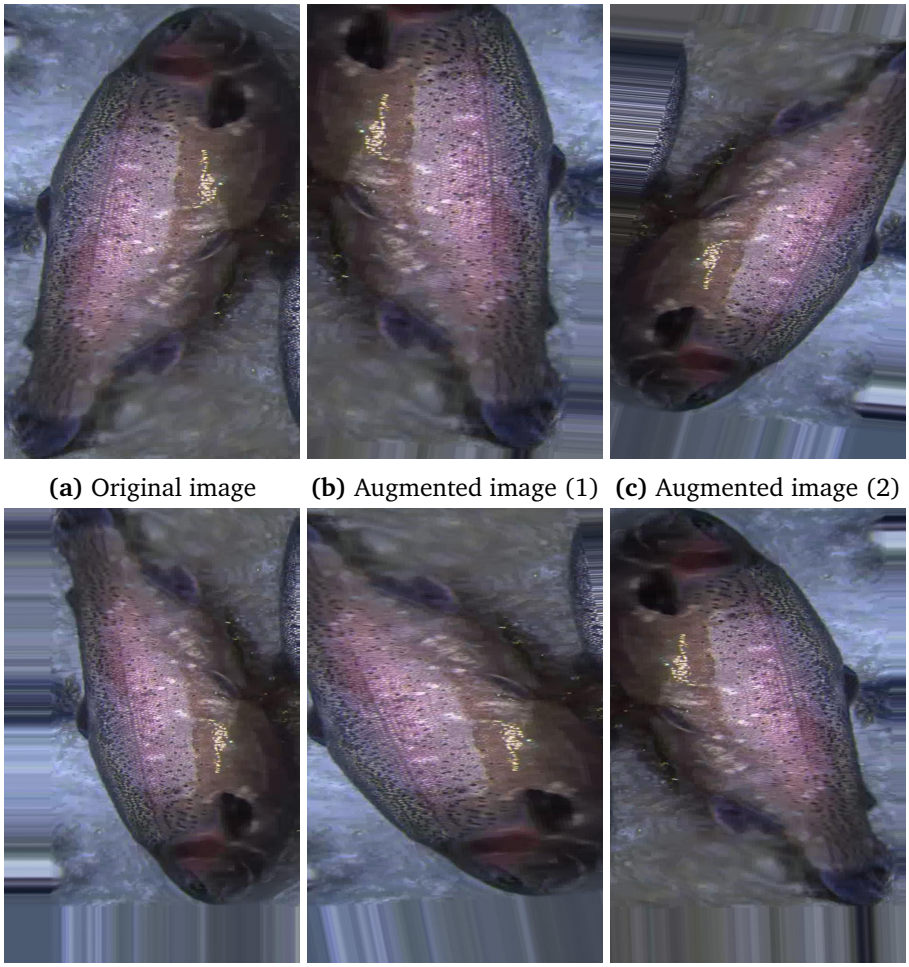
- `rotation_range`: Specifies the range (in degrees) for random rotations of the image. For example, if rotation_range=10, the image can be rotated randomly between -10 and 10 degrees.
- `width_shift_range`: Specifies the range (as a fraction of the total width) for horizontal shifts of the image.
- `height_shift_range`: Specifies the range (as a fraction of the total height) for vertical shifts of the image.
- `shear_range`: Specifies the range (in degrees) for random shearing transformations of the image.
- `zoom_range`: Specifies the range for random zooms of the image.
- `horizontal_flip`: Specifies whether to randomly flip images horizontally.
- `vertical_flip`: Specifies whether to randomly flip images vertically.

- `fill_mode`: Specifies how to fill in any new pixels created during the transformation. The 'nearest' mode is used, which means that the nearest pixel will be used to fill in the new pixel values.

The `ImageDataGenerator` object is created with these parameters, and the code uses the `flow` method [63] of the `ImageDataGenerator` object to generate batches of augmented data. The different parameters used here are:

- `img`: This is the input image that will be augmented. It is a 3-dimensional NumPy array representing the image. NumPy is a fundamental library in Python that enables efficient numerical computations and offers extensive support for working with multi-dimensional arrays, such as a 3-dimensional NumPy array commonly used to represent images [64].
- `batch_size`: This parameter specifies the number of images to be generated in each batch.
- `save_prefix`: This parameter specifies a prefix to use when saving the generated images. The actual file names will be the prefix followed by an index number and the specified file format.
- `save_format`: This parameter specifies the file format to use when saving the generated images, which in this case is 'jpg'.
- `save_to_dir`: This parameter specifies the directory in which to save the generated images. If None (the default), the images will not be saved to disk and will only be generated in memory, but for the project, the path has been specified to a directory so that the images can be saved on disk.

Using a Python script, each image was passed through the `flow` method to generate five augmented images from one single image. This way, the data was increased artificially for the mature category. Fig. 3.6 presents a sample group of images where additional five pictures are generated by the augmentation method by using one image. The images belong to the mature class (which is the minority class) as per the data distribution.



(a) Original image **(b)** Augmented image (1) **(c)** Augmented image (2)

(d) Augmented image (3) **(e)** Augmented image (4) **(f)** Augmented image (5)

Figure 3.6: Sample images from mature class generated using augmentation

The distribution of data between the two classes before and after augmentation is presented in Table 3.2.

Label	**Label count before**	**Label count after**	**Distribution before**	**Distribution after**
trout	25270	25270	≈84%	≈50%
mature	4696	25612	≈16%	≈50%
total	29966	50882	100%	100%

Table 3.2: Data distribution before and after augmentation

3.4 Pre-trained Model Architectures

The industry partner which has collaborated with this project, expressed a preference to work with TensorFlow 2.0 [65], therefore the chosen pre-trained models were the ones that are compatible with TensorFlow 2.0. A list of the models that are compatible with Coral Edge TPU at the time of writing this project is presented in Table 3.3.

Model Name	Dataset	Input Size	Top-1 Accuracy
MobileNet V1	ImageNet	224x224x3	69.5%
MobileNet V2	ImageNet	224x224x3	73.2%
MobileNet V3	ImageNet	224x224x3	77.5%
ResNet-50	ImageNet	224x224x3	73.6%

Table 3.3: Image classification models compatible with TF v2.0

All the models mentioned in Table 3.3 are trained on the *ImageNet* dataset. ImageNet is a large-scale visual database that contains millions of labeled images representing a wide variety of object categories. It is widely used in computer vision research to develop and evaluate models for tasks such as image classification, object detection, and image segmentation [66]. The Top-1 accuracy represents the percentage of correctly predicted labels for the most probable class. For the ImageNet dataset, Top-1 accuracy is highest for MobileNet V3 (77.5%) and least for MobileNet V1 (69.5%).

The dataset that is being used for the project is new and has never been tested before for any image classification tasks. Therefore, there is no existing baseline model that can be used for comparison with the results obtained from this project. ResNet was the winning model of the ImageNet (ILSVRC) 2015 competition and is a popular model for image classification [67]. the architecture of each model that has been used for the experiments will be discussed in detail. ResNet-50 is a CNN architecture, while MobileNet v1, MobileNet v2, and MobileNet v3 are variants of MobileNets, which are also CNN architectures but specifically designed for efficient mobile and embedded applications.

CNN is a type of deep learning algorithm that is widely used in image and video recognition tasks. CNNs are designed to recognize patterns in images by using a series of convolutional layers that extract features from the input image. These features are then passed through fully connected layers to classify the image into different categories. CNN architecture is useful for feature recognition because it can automatically learn and extract relevant features from images without the need for manual feature engineering. This makes CNNs highly effective in recognizing complex patterns and structures in images, such as edges, corners, shapes, and textures [68]. An example is the ResNet model, which was developed by re-

searchers at Microsoft Research Asia. The ResNet model uses residual connections to allow for deeper network architectures while avoiding the vanishing gradient problem that can occur with deep neural networks [68].

3.4.1 Pre-trained Model Descriptions

To understand each of the pre-trained models better that will be used for the classification task, the following sections present a short introduction to the architecture of the pre-trained ResNet-50, MobileNet V1, MobileNet V2 and MobileNet V3.

ResNet-50

ResNet is a deep residual learning framework for image recognition. It is a neural network architecture that allows for the creation of much deeper networks than previously possible, while still maintaining high accuracy and ease of optimization. The ResNet framework reformulates layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. This approach makes it easier to train very deep neural networks and has been shown to be effective in various image recognition tasks as mentioned in the paper by He et al. [69]. The ResNet architecture has achieved state-of-the-art results on several benchmark datasets, including ImageNet, CIFAR-10, and COCO [69]. A 50-layer net with a 3-layer bottleneck block, resulting in a 50-layer ResNet, was introduced in the previous research [69]. They show that the 50-layer ResNet is more accurate than the 34-layer one by a considerable margin and does not suffer from the degradation problem where the accuracy of the network starts to degrade as the depth of the network increases. The basic architecture diagram of ResNet-50 is presented in Fig. 3.7.

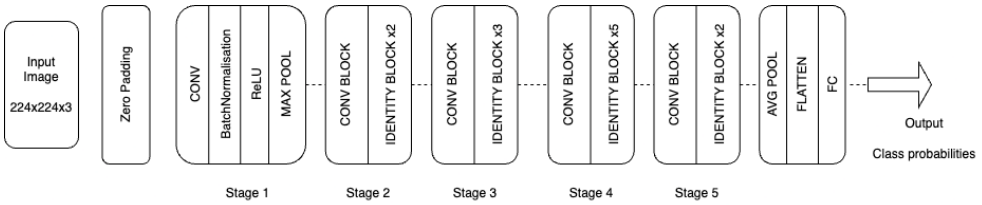


Figure 3.7: ResNet-50 architecture

MobileNet V1

MobileNet v1 is a class of efficient models for mobile and embedded vision applications. It is based on a streamlined architecture that uses depth-wise separable convolutions to build lightweight deep neural networks [70]. MobileNet v1 introduces two simple global hyper-parameters that efficiently trade off between latency and accuracy, allowing the model builder to choose the right-sized model

for the application based on the constraints of the problem. Extensive experiments have shown strong performance compared to other popular models on ImageNet classification and a variety of different applications and use cases. MobileNet v1 is much more lightweight than ResNet (which is discussed previously) and also a couple of other deep learning models like VGG16, and GoogleNet while still achieving comparable accuracy on ImageNet classification [70]. The basic architecture diagram of MobileNet V1 is presented in Fig. 3.8.

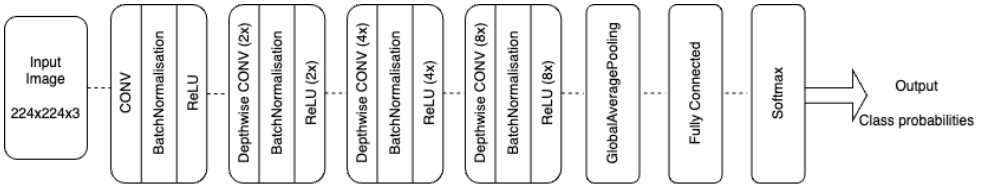


Figure 3.8: MobileNet V1 architecture

MobileNet V2

MobileNetV2 is a mobile architecture that was introduced in a paper by Google researchers in 2018 [71]. One of the key features of MobileNetV2 is its use of inverted residuals and linear bottlenecks. These techniques allow for more efficient use of computation and memory resources while maintaining high accuracy. Inverted residuals involve using a bottleneck layer with fewer filters followed by a layer with more filters, which reduces computation while maintaining accuracy. Linear bottlenecks involve using linear activations instead of non-linear activations in the bottleneck layers, which reduces memory usage. Another advantage of MobileNetV2 is its efficiency in terms of model size and computation. The authors compare MobileNetV2 to other popular mobile architectures, such as ResNet and MobileNetV1, and show that it outperforms them on most benchmarks while using fewer parameters and less computation. This makes it an attractive option for applications where computational resources are limited, such as mobile devices or embedded systems [71]. Overall, the results suggest that MobileNetV2 is a strong performer in comparison to other architectures for image classification tasks due to its use of inverted residuals and linear bottlenecks, as well as its efficiency in terms of model size and computation. The basic architecture diagram of MobileNet V2 is presented in Fig. 3.9.

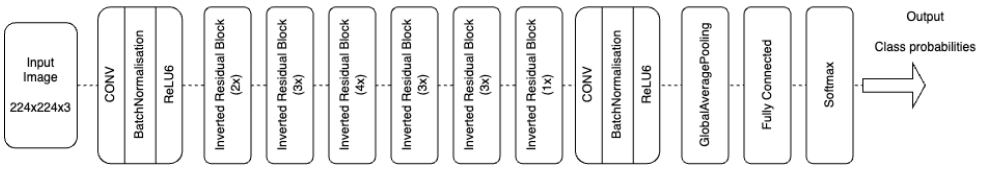


Figure 3.9: MobileNet V2 architecture

MobileNet V3

MobileNetV3 is the next generation of MobileNets, which are designed to work efficiently on mobile phone CPUs. It is a neural network model that is specifically optimized for image classification tasks. Compared to previous versions, such as MobileNetV1 and V2, MobileNetV3 offers significant improvements in accuracy and efficiency [72]. One of the key features of MobileNetV3 is its combination of hardware-aware NAS and the NetAdapt algorithm. This allows the model to be optimized for mobile phone CPUs, resulting in faster inference times and lower power consumption. In terms of accuracy, MobileNetV3 outperforms both ResNet and previous versions of MobileNets. This is due to its novel architecture design, which includes a combination of squeeze-and-excitation blocks, hard-swish activation functions, and other improvements. Overall, the advantages of MobileNetV3 include its high accuracy, efficiency on mobile phone CPUs, and ability to perform well on a variety of image classification tasks. Its novel architecture design sets it apart from other models and makes it a promising option for on-device computer vision applications [72]. The basic architecture diagram of MobileNet V3 is presented in Fig. 3.10.

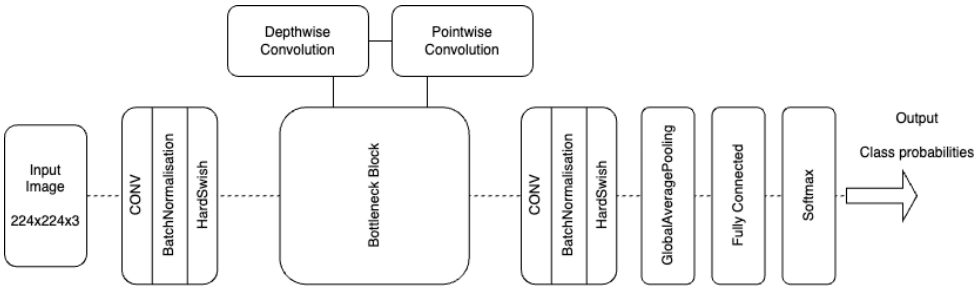


Figure 3.10: MobileNet V3 architecture

In conclusion, ResNet-50, MobileNet V1, MobileNet V2, and MobileNet V3 are pre-trained models that have significantly contributed to image recognition and classification. ResNet-50 stands out for its ability to train deep networks with high accuracy, addressing the degradation problem. MobileNet V1 offers a streamlined architecture for mobile and embedded applications, achieving a good trade-off between latency and accuracy. MobileNet V2 improves computational and memory efficiency through inverted residuals and linear bottlenecks, outperforming other mobile architectures. MobileNet V3, optimized for mobile CPUs, combines hardware-aware network architecture search and novel features for improved accuracy and efficiency. Overall, these models provide a range of strengths, from deep network training to lightweight and efficient architectures, advancing on-device computer vision applications.

3.5 Model initialization

For the classification task on the dataset, the **ResNet50** [73], **MobileNet** [74], **MobileNetV2** [75], **MobileNetV3Large** [76] objects were used from the `tf.keras.applications` module. There are various arguments that can be used with the object of the models initialized. The following have been used when loading the ResNet-50, MobileNet, MobileNetV2 and MobileNetV3 objects:

- `weights='imagenet'`: This specifies the weight initialization to be used for the model. In this case, the pre-trained weights from the ImageNet dataset are used, which is a large database of labeled images that are commonly used to pre-train deep learning models.
- `include_top=False`: This specifies whether or not to include the fully connected layer at the top of the network. By setting this to `False`, the final layer of the pre-trained model is removed, which is typically used for image classification, and allows for customization of the output layer.
- `input_shape=(224, 224, 3)`: This specifies the expected shape of the input data to the model. In this case, it is set to `(224, 224, 3)`, which is a common input size for image classification tasks. The first two values represent the height and width of the input image, respectively, and the third value represents the number of color channels (3 for RGB images). The sample representation of images in 224x224 dimensions is presented in Appendix 6.1.

The base_model variable then contains the object of the model initialized, which is then to be used for further customization and training. Afterwards, by iterating over all the layers in the base_model object, the `trainable` attribute is set to `False` for each layer. The purpose of setting the trainable attribute to False is to freeze the weights of the pre-trained model during the training process. By freezing the weights, the layers are prevented from being updated during the training process and allows to use the pre-trained features extracted by the model as a fixed feature extractor for this project's classification task. The base of the model is used to extract features from input images, and then new layers are added on top of the base to classify the images into one of two classes, trout or mature.

3.6 Architecture setup

There can be several combinations of layers that can be added at the end of each pre-trained model to experiment with the performance of the model. For this project, two different architecture setups were explored with each of the models discussed in the previous section. [2]

[2]Throughout this thesis, the terms **setup 1** and **setup 2** are used as a shorthand for **Architecture setup 1** and **Architecture Setup 2** respectively for brevity and ease of reference.

3.6.1 Architecture Setup 1

In this setup, the following new layers were added to the pre-trained model object to create a new model that is specifically tailored to this image classification task:

- **Pre-trained model output**: The output of the pre-trained model is taken as the input for the new layers. Specifically, `x` is set to the output of the last layer of the model.
- **GlobalAveragePooling2D()**: A global average pooling layer is added to the model. Global average pooling is a technique commonly used in computer vision tasks that involves computing the average value for each feature map in the output of the model [77]. The result is a single value for each feature map, which can then be fed into the next layer.
- **Dense**: A dense layer with 512 units and a *ReLU* activation function is added to the model. The ReLU activation function is a non-linear function that outputs the input value if it is positive, or zero otherwise [78]. The dense layer is a fully connected layer where each unit in the layer is connected to every unit in the previous layer. The ReLU activation function helps introduce non-linearity to the model. ReLU is a preferred choice for many neural network applications due to its ability to promote sparsity and its faster training speed. [79].
- **Predictions**: A final dense layer is added to the model with two units (corresponding to the two classes in our classification task) and a *sigmoid* activation function. The sigmoid activation function outputs a probability value between 0 and 1 for each class, representing the likelihood that the input image belongs to that class and it is commonly used in the output layer of binary classification problems [80].

An overview of this setup when used with pre-trained models is presented in Fig. 3.11.

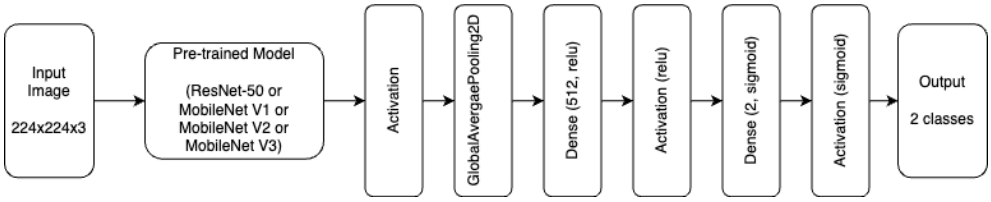


Figure 3.11: Architecture setup 1 overview

3.6.2 Architecture Setup 2

In addition to the layers described in setup 1 (3.6.1), some new layers were added to the classification model. The idea was to combine various techniques to achieve good performance and prevent overfitting. In addition, these new layers were expected to strike a balance between model complexity and regularization and make it suitable for the classification task.

- **Pre-trained model output**: The output of the pre-trained model is added as the input for the new layers. Specifically, `x` is set to the output of the last layer of the model.
- **GlobalAveragePooling2D**: A global average pooling layer is added to the model. Global average pooling is a technique commonly used in computer vision tasks that involves computing the average value for each feature map in the output of the model [77]. The result is a single value for each feature map, which can then be fed into the next layer.
- **Dense with L2 Regularization**: A Dense layer with 512 units and ReLU activation is added next. This layer introduces non-linearity to the model and enables it to learn complex patterns in the data. The inclusion of regularization using *L2 regularization* (with a regularization strength of 0.01) is used to prevent overfitting by encouraging the model to generalize well to unseen data. L2 regularization adds a penalty term to the loss function that encourages smaller weights in the model by adding the sum of squared weights multiplied by a regularization parameter [81].
- **BatchNormalization**: Batch Normalization is applied to the output of the previous layer. It is used to normalize the activations by standardizing them and stabilize the learning process by reducing the internal covariate shift. Batch Normalization can speed up training and improve the model's ability to generalize [82].
- **Dropout**: Dropout is used as a regularization technique to prevent overfitting. It randomly sets a fraction of input units to 0 during training, which helps to reduce the reliance of the model on specific features. A dropout rate of 0.5 means that 50% of the input units will be randomly dropped during training.
- **Predictions**: A final dense layer is added to the model with two units (corresponding to the two classes in the classification task) and a sigmoid activation function. The sigmoid activation function outputs a probability value between 0 and 1 for each class, representing the likelihood that the input image belongs to that class and it is commonly used in the output layer of binary classification problems [80].

An overview of this setup when used with the pre-trained models is presented in Fig. 3.12.

Figure 3.12: Architecture setup 2 overview

Finally, at the end of each setup, a `model` object was created by defining a new

`tf.keras.Model` object, which takes `base_model.input` (the input tensor to the base of the initialized model) as its input, and `predictions` (the output tensor from the classification layers you added on top of the base) as its output. This is essentially combining the pre-trained initialized model with the new classification layers that were added, resulting in a new model that takes an input image, processes it through the pre-trained model layers to extract features, and then uses the new classification layers to predict as to which class the image belongs to. With this, an end-to-end model was created that can be used for training on the new dataset. The weights of the pre-trained initialized model layers are frozen, so only the weights of the new classification layers will be updated during training. This allows to leverage the pre-trained model as a powerful feature extractor while fine-tuning the classification layers to work well on our specific dataset.

3.7 Model Compilation

Once all the layers are added and the model is prepared, the next step is to compile the model. This step is necessary before training the model. The `model.compile()` [83] configures the model for training. Specifically, it sets the optimization algorithm, the loss function, and the evaluation metric(s) to be used during training. The following arguments have been used for the model:

- **Adam**: The optimization algorithm that will be used to update the weights of the model during training is added here. In this case, the 'Adam' optimizer is used, which is a recommended optimization algorithm by the literature for classification purposes in deep learning [84]. The suggested default value of `learning_rate` is 0.001 [85], and additionally, a learning_rate of 0.00001 and 0.000001 was also experimented with.
- **Sparse Categorical Crossentropy**: Next, the loss parameter which specifies the loss function that will be used to calculate the difference between the predicted output and the true output is added. In this case, `SparseCategoricalCrossentropy()` is used as the loss function.
- **metrics=["accuracy"]**: The metrics parameter specifies the metric(s) that will be used to evaluate the performance of the model during training. In this case, 'accuracy' is used as the evaluation metric. In addition to the accuracy, there is a set of other metrics such as precision, recall, F1 score, and confusion matrix to evaluate the model after training which is discussed in Section 3.9.

Once the model is compiled, the training can be started on the dataset using `model.fit()` [86]. During training, the model will use the optimizer, loss function, and evaluation metric(s) that were specified in the model.compile() step to update the weights and improve its performance.

3.8 Training Process

After several pre-processing steps, defining custom classification layers and compilation, the next step is where the model is trained with the data defined. *TensorBoard* [87], which is a tool to visualize and understand the behavior of machine learning models, was used. Several TensorBoard callbacks were used [88] to monitor the training progress, log the details and visualize the model performance, which is listed as follows:

- `TensorBoard`: This callback is used to visualize training and validation metrics during training. It is initialized with a `logdir` parameter which specifies the directory where TensorBoard logs will be written. The `histogram_freq` parameter, which is set to 1, specifies how often to compute histograms of layer activations and gradients, and the `write_graph` parameter, which is set to True, determines whether to write the graph definition to the logs.
- `ModelCheckpoint`: This callback is used to save the model weights at certain intervals during training. It is initialized with a `filepath` parameter, which specifies where to save the checkpoint files, and a `save_weights_only` parameter, set to True, specifies whether to save only the model weights or the entire model. In this case, only the weights are saved. The `save_freq` parameter, set to epoch, specifies how often to save the weights, and the `period` parameter set as 1, specifies the number of epochs between saves.
- `TimeHistory`: This is a custom-defined callback that is used to record the training time for each epoch. This is specifically helpful in calculating the time taken for the entire model to train.

Once the callbacks are defined, the next step is where the model actually starts to train. The pre-existing layers which were frozen in the beginning, will not be trained and only the new layers that have been added either using setup 1 (3.6.1) or setup 2 (3.6.2). Following is a detailed summary of the training setup of the model: The `tensorflow.keras.model.fit` [89] method is called to train the deep learning model. In most cases of this project, the number of training epochs, `num_epochs` is set to 150 which is the number of times the model will iterate over the entire training dataset during training. A variable `history` of type **History** is used as an object that is returned by the model.fit() function after the training process is completed. The history object contains information about the training history, such as the loss and metrics values recorded during each epoch. The different parameters passed to the `model.fit()` function are as follows:

- `train_data`: This is the training dataset, which contains input images and corresponding class values, which in this case is class label 0 for the mature category and class label 1 for the trout category.
- `validation_data`: This is the validation dataset, which is used to evaluate the model's performance during training. It contains input images and corresponding class values that are not used for training but helps to monitor the model's progress and detect overfitting.

- `epochs`: This parameter specifies the number of epochs for which the model should be trained. In most training cases for the project, it is set to 150, as discussed earlier.
- `callbacks`: This parameter specifies which callbacks should be included in the training for visualization and monitoring. The three TensorBoard callbacks defined above are passed as a list to this parameter.
- `shuffle`: This parameter which is set to True, determines whether the training data should be shuffled at the beginning of each epoch. It is used to prevent the model from learning the order or patterns within the dataset and improve the generalization.

3.9 Evaluation Metrics

After the models have completed their training, evaluation plays a crucial role in assessing the performance and effectiveness of the trained model. Evaluation generally involves testing the model on a separate dataset that was not used during training to provide an unbiased measure of its capabilities. For this purpose, there was a `test_data` separated at the beginning, which is 10% of the original dataset available. There are several evaluation criteria that are commonly used to analyze the model's performance, such as accuracy, precision, etc. For this project, the following evaluation metrics have been calculated:

- **Accuracy**: Accuracy is the most basic evaluation metric, representing the percentage of correctly classified images out of the total number of images. While it provides a general measure of overall performance, it can be misleading if the dataset is imbalanced [90]. Thus, a combination of other metrics which can support accuracy shortcomings is necessary. The `tf.keras.metrics.Accuracy` function [91] has been used to evaluate this.
- **Precision and Recall**: Precision and recall are commonly used for binary classification tasks. Precision measures the proportion of correctly classified positive instances out of all instances predicted as positive. Recall, also known as sensitivity or true positive rate, measures the proportion of correctly classified positive instances out of all actual positive instances [90]. The `tf.keras.metrics.Precision` function [92] and `tf.keras.metrics.Recall` function [93] has been used to evaluate this.
- **F1 Score**: The F1 score which is also a metric used to evaluate the performance of a classification model, combines precision and recall into a single value, providing a balanced measure of a model's overall accuracy. The F1 score is the harmonic mean of precision and recall, calculated as $F1\ score = 2 * (precision * recall)/(precision + recall)$. The overall F1 score represents the average performance of the model across all classes. It provides a general measure of the model's effectiveness in terms of precision and recall, considering all classes collectively.
- **Confusion Matrix**: A confusion matrix provides a detailed breakdown of

the model's predictions for each class. It shows the true positive, true negative, false positive, and false negative counts, enabling a deeper analysis of the model's performance [94]. The `sklearn.metrics.confusion_matrix` function [95] has been used for the evaluation.

- **Receiver Operating Characteristic (ROC) Curve**: The ROC curve is a graphical representation of the true positive rate (sensitivity) against the false positive rate (1 - specificity) at various classification thresholds. The area under the ROC curve (AUC score) is often used as a summary metric, with higher values indicating better performance [96]. The `sklearn.metrics.roc_curve` and `sklearn.metrics.auc` have been used for this evaluation [97] [98].

3.10 System Setup

The system setup involves a GPU-enabled system provided by NTNU Gjøvik, which consists of two GPUs. The first GPU, **GPU 0**, is an NVIDIA GeForce with 8192 MiB of memory, and the second GPU, **GPU 1**, also has an NVIDIA GeForce with the same memory capacity.

To facilitate the project, a conda environment was created with specific software versions. The environment includes multiple packages among which the most prominent ones are the following:

- Python v3.8.16
- Pip v23.0.1
- Cuda v11.6
- TensorFlow-GPU v2.5.0
- Pandas v2.0
- NumPy v1.20.3
- Keras v2.12.0
- Matplotlib v3.7.1

The setup enabled efficient utilization of the GPUs for accelerated computations. It was expected that the GPUs' computational power will significantly enhance the speed and performance of the related tasks. Additionally, the included software versions of various libraries such as TensorFlow, pandas, numpy, Keras, and matplotlib serve as necessary tools to perform the desired tasks effectively.

Chapter 4

Results

4.1 Results with Unbalanced Dataset

In this section, the results are published for the experiments with the deep learning models using all the data available was used. These are all original images without any data augmentation. A total of **29966** images belonging to both classes were used in this setup. The data distribution is presented in Table 4.1.

Label	**Class**	**Images**	**Distribution**
Mature	0	4696	≈16%
Trout	1	25270	≈84%

Table 4.1: Data distribution of unbalanced dataset

Tables 4.2 and 4.3 provide detailed results for the two different model architecture setup 1 (3.6.1) and setup 2 (3.6.2), each associated with a specific variation. These results offer comprehensive insights into the performance and characteristics of each model within the two different model architecture setups. Analyzing these tables will help in understanding the impact of different architectural choices and parameter configurations on the models' performance and training duration.

Model Architecture Setup 1 (3.6.1)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Batch Size	32	64	64	64
Total Data	937	468	468	468
Train Size	655 (70%)	328 (70%)	328 (70%)	328 (70%)
Validation Size	187 (20%)	93 (20%)	93 (20%)	93 (20%)
Test Size	95 (10%)	47 (10%)	47 (10%)	47 (10%)
Total Params	24,637,826	3,754,690	2,914,882	4,883,330
Trainable Params	1,050,114	525,826	656,898	656,898
Learning Rate	0.001	0.001	0.001	0.001
Epochs	300	150	150	150
Training Time (mins)	301	86	87	88
Model Architecture Setup 2 (3.6.2)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Batch Size	64	64	64	64
Total Data	468	468	468	468
Train Size	328 (70%)	328 (70%)	328 (70%)	328 (70%)
Validation Size	93 (20%)	93 (20%)	93 (20%)	93 (20%)
Test Size	47 (10%)	47 (10%)	47 (10%)	47 (10%)
Total Params	24,639,874	3,756,738	2,916,930	4,885,378
Trainable Params	1,051,138	526,850	657,922	657,922
Learning Rate	0.00001	0.00001	0.00001	0.00001
Epochs	150	150	150	150
Training Time (mins)	133	87	91	90

Table 4.2: Model configurations for unbalanced dataset

In setup 1, the choice of batch sizes varied between the models. ResNet-50 was experimented to use a smaller batch size of 32, while MobileNet V1, V2, and V3 used a batch size of 64. In setup 2, all models utilized a batch size of 64. Therefore dataset sizes differed in setup 1, where ResNet-50 had a larger training dataset of 655 batches, while MobileNet V1, V2, and V3 used a training dataset of 328 batches. In setup 2, the dataset sizes were the same for all models, with 328 batches in the training dataset and 93 batches in the validation and test datasets. It is observed that the ratio of the split between the dataset remained constant throughout both the setups and followed a 70%-20%-10% split for train, validation and test datasets respectively.

Regarding the total and trainable parameters, ResNet-50 had the highest values in both setups 1 and 2. It had a total of 24,637,826 parameters and 1,050,114 trainable parameters in Type 1, and 24,639,874 total parameters and 1,051,138 trainable parameters in Type 2. MobileNet V3, MobileNet V1, and MobileNet V2 followed ResNet-50 in terms of parameter counts. The higher number of total and trainable parameters in ResNet-50 compared to the other models in both setups suggests a more complex and potentially deeper architecture in ResNet-50.

The learning rate and epochs settings differed between setups 1 and 2. In setup 1, all models were trained with a learning rate of 0.001 for 150 epochs, except for ResNet-50, which was trained for 300 epochs for experimental purposes. On the other hand, in setup 2, all models were trained with a learning rate of 0.00001 for 150 epochs. The variation in learning rate and epochs settings between setup 1 and setup 2 implies that different training strategies were employed to optimize the models' performance in each setup.

Training times varied depending on the model and the setup. In setup 1, ResNet-50 had the longest training time of 301 minutes, while MobileNet V1, V2, and V3 took 86-88 minutes most likely due to the larger number of batches to process and that it was trained longer than other models. In setup 2, the training times ranged from 87 to 133 minutes, with MobileNet V1 having the shortest training time. The longer training time required for ResNet-50 in setup 1 and setup 2 may suggest that it is a more computationally intensive model compared to the MobileNet variants.

Having discussed the setup of each model for both setups, it is essential to delve into the performance and results attained by the various models. This analysis will shed light on their overall performance and by examining key evaluation metrics, valuable insights can be gained into how well these models performed in the given context. Table 4.3 presents the evaluation metrics for each model.

Evaluation Metrics				
Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Model Architecture Setup 1 (3.6.1)				
Accuracy	**86.6%**	83.8%	85.0%	84.9%
Precision	**91.8%**	90.1%	91.1%	85.5%
Recall	92.4%	90.2%	91.8%	**99.1%**
F1 Score	**0.91**	0.90	**0.91**	**0.91**
Model Architecture Setup 2 (3.6.2)				
Accuracy	**87.0%**	84.9%	84.9%	84.8%
Precision	**92.4%**	90.9%	90.8%	84.8%
Recall	92.3%	91.3%	91.6%	**100%**
F1 Score	**0.92**	0.90	0.90	0.91

Table 4.3: Evaluation metrics for unbalanced dataset

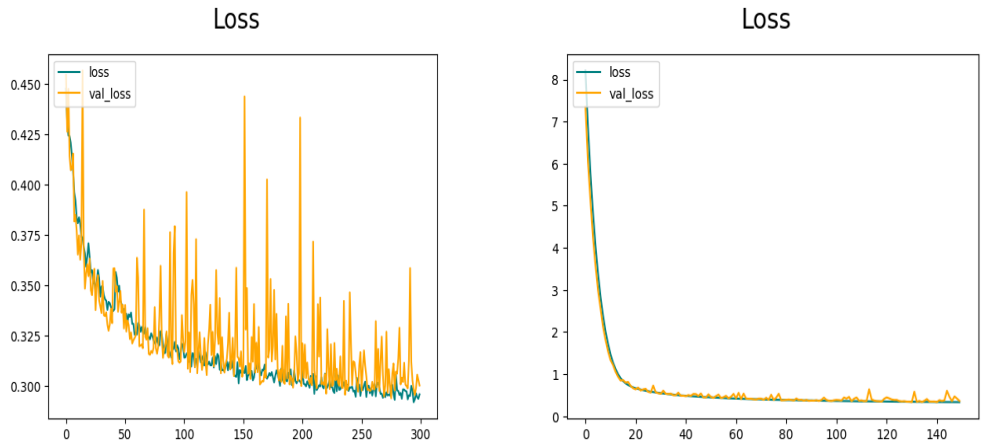
In setup 1, the results indicate strong overall performance across the models, as evidenced by the high accuracy scores ranging from 83.8% to 86.6%. The consistently high precision scores suggest that the models exhibit a good ability to correctly identify positive instances. The variation in recall scores suggests that MobileNet V3 shows particularly high sensitivity in capturing positive instances. The consistently high F1 scores reflect a balance between precision and recall, indicating the models' effectiveness in achieving accurate and comprehensive predictions.

Moving to setup 2, the competitive performance across the models is evident from the accuracy scores ranging from 84.8% to 87.0%. The precision scores, although varying, generally indicate a decent ability to correctly identify positive instances. Notably, the recall scores show that MobileNet V3 has a higher capability in capturing positive instances compared to MobileNet V2. The F1 scores, ranging from 0.90 to 0.92, further emphasize the models' ability to achieve a favorable balance between precision and recall.

These evaluation results suggest that both setups exhibit strong performance overall, with some variations in specific metrics among the models. Understanding these nuances helps assess the models' suitability for different requirements and aids in making informed decisions for future applications. In order to understand the performance of each model better, the results for ResNet-50, MobileNet V1, MobileNet V2 and MobileNet V3 are presented.

4.1.1 ResNet-50

The results for ResNet-50 on the unbalanced dataset with setup 1 and setup 2 are presented in this section. The loss and accuracy curves are presented to have an overview of the model while it was training followed by a confusion matrix and AUC score for the ROC curve by evaluation of test data.

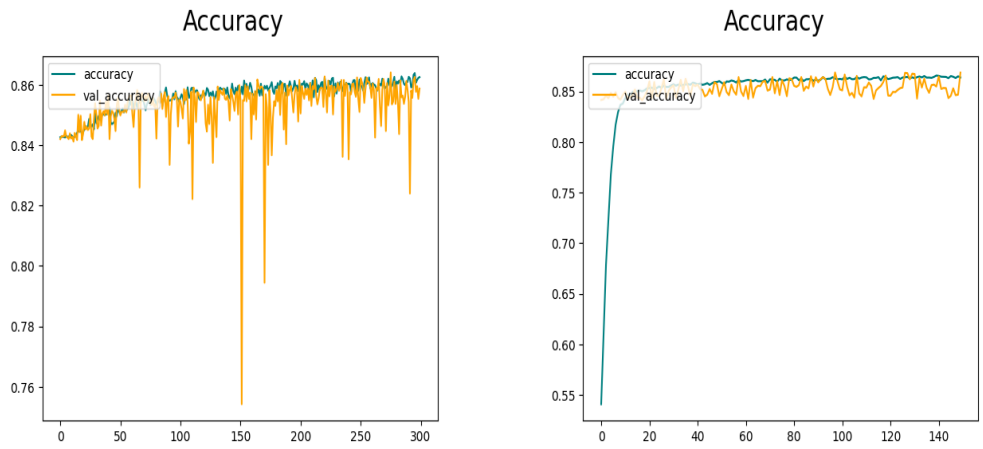


(a) Loss curve setup 1

(b) Loss curve setup 2

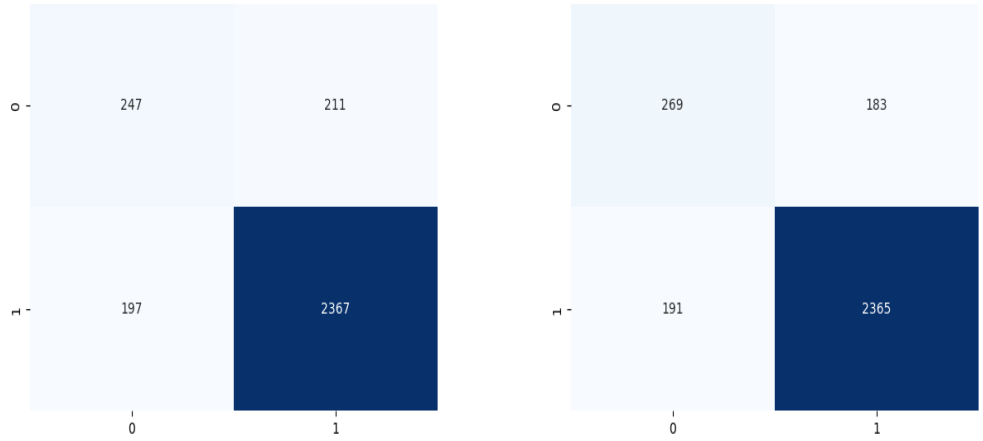
Figure 4.1: ResNet-50 loss curves for unbalanced data

The loss curves for ResNet-50 are presented in Fig. 4.1. The spikes in the validation loss in setup 1 indicate that the model's performance on unseen data (validation set) may not be improving consistently, and this can also mean that the model may be overfitting to the training data by not capturing the underlying patterns in the data effectively. It might be overly sensitive to variations or noise in the validation set, leading to inconsistent performance. As for setup 2, it can be observed that the loss starts at a relatively high value of 8 and decreases to less than 1, indicating that the model quickly learns and adjusts its predictions during the initial training epochs, and the decreasing trend of the loss curve suggests that the model continues to improve its performance over time. The overall smoothness of the loss curve implies that the model's performance is relatively stable during training.



(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.2: ResNet-50 accuracy curves for unbalanced data



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

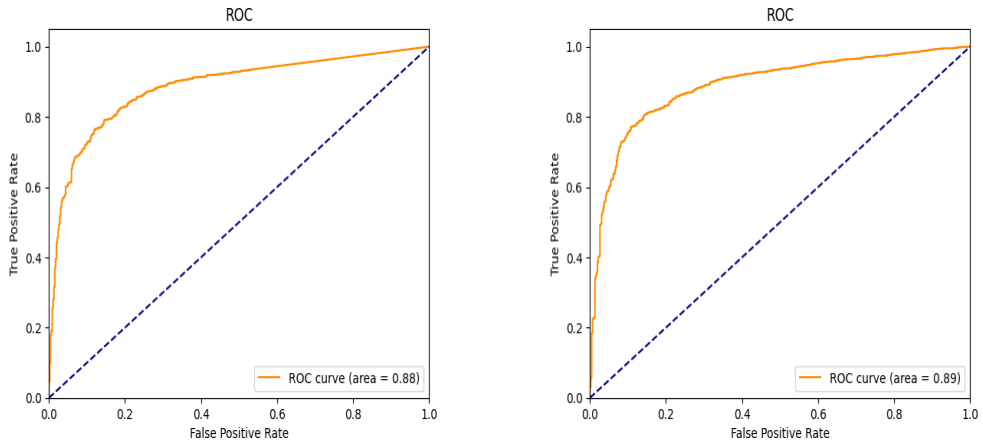
Figure 4.3: ResNet-50 confusion matrix for unbalanced data

The accuracy curves for ResNet-50 are presented in Fig. 4.2. The overall trend of the accuracy curve shows a gradual increase from around 0.84 to 0.86 over the course of 300 epochs. This suggests that the model is learning and making more accurate predictions over time. The presence of spikes in the accuracy curve indicates that there are instances where the model's performance experiences sudden drops or increases. The spikes and fluctuations in the accuracy curve may indicate that the model is encountering difficulties in converging to a stable and optimal solution. The significant drop at the 150th epoch suggests a potential setback or instability in the learning process. In the case of setup 2, the accuracy graph demonstrates notable progress, starting from a lower scale of 0.55 and reaching a range above 0.85. This suggests that the model is learning and making substantial

improvements in its predictions as the training progresses. Despite the spikes in the validation curve, the fact that they remain within the range of 0.80 to 0.85 suggests a relatively consistent level of accuracy during validation. This indicates that the model is able to maintain a reasonable level of performance, although with some fluctuations.

The confusion matrix for ResNet-50 is depicted in Fig. 4.3. By examining the provided confusion matrices for setup 1 and setup 2, the percentages of correct and incorrect classifications for each class can be assessed.
In setup 1: **Class 0 (mature)**: Correct classification is 53.95% and the incorrect classification is 46.05% **Class 1 (trout)**: Correct classification is 92.29% and the incorrect classification is 7.71%
In setup 2: **Class 0 (mature)**: Correct classification is 59.55% and the incorrect classification is 40.45% **Class 1 (trout)**: Correct classification is 92.53% and the incorrect classification is 7.47%

These percentages provide insights into the accuracy of classification for each class within the respective setups.



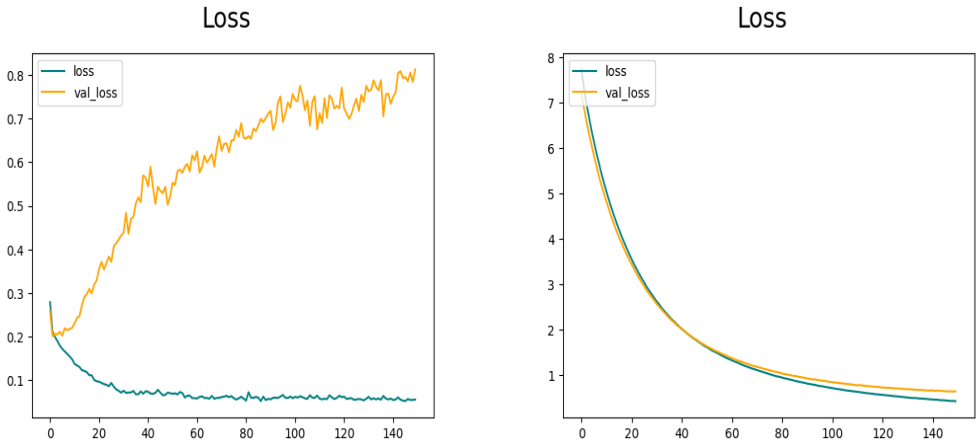
(a) Setup 1 **(b)** Setup 2

Figure 4.4: ResNet-50 AUC score for unbalanced data

The ROC curve's AUC score (Area Under the Curve) is displayed in Fig. 4.4. The AUC score, ranging from 0 to 1, serves as a measure of the classifier's performance. A score of 0.5 suggests a random classifier, while a score of 1 indicates a flawless classifier. Examining the given scores, for setup 1, the AUC score is 0.88 and for setup 2, the AUC score is 0.89. These AUC scores highlight the models' performance in distinguishing between classes. The higher the AUC score, the better the classifier's ability to correctly rank and differentiate the classes.

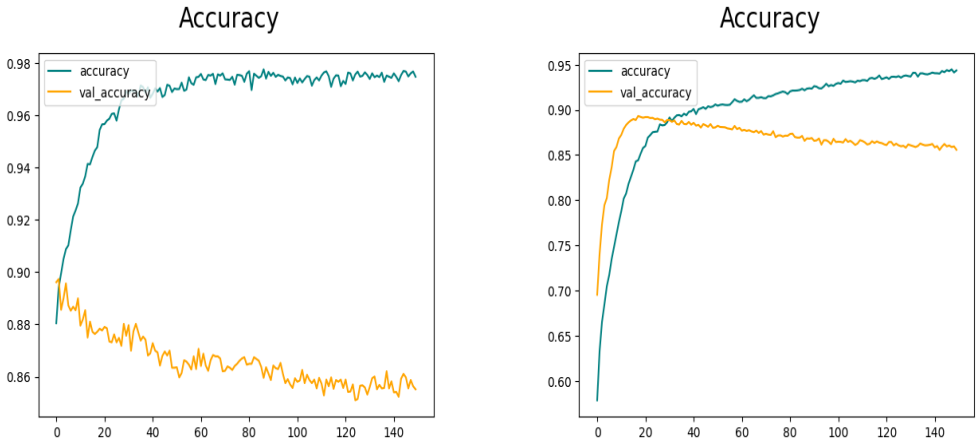
4.1.2 MobileNet V1

In this part, the results of MobileNet V1 are presented with setup 1 and setup 2 for unbalanced dataset. The loss and accuracy curves present an overview of the model performance while training. The confusion matrix and AUC score present an overview of the effectiveness of the classifier on test data.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.5: MobileNet V1 loss curves for unbalanced data



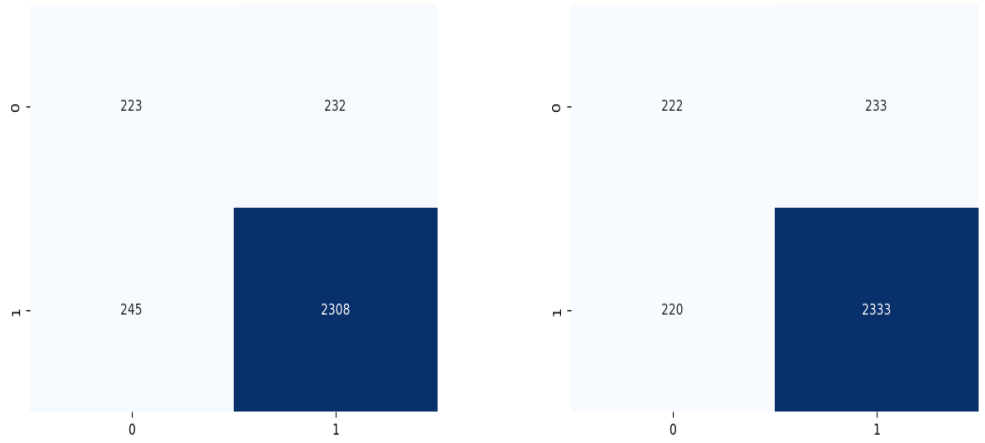
(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.6: MobileNet V1 accuracy curves for unbalanced data

The loss curves for MobileNet V1 are presented in Fig. 4.5. In setup 1, MobileNet V1 shows promising results in terms of train loss reduction, indicating successful learning from the training data. However, the increasing validation loss suggests a lack of generalization and potential overfitting. The validation loss curve

starts at 0.3, but instead of decreasing or stabilizing, it goes up to over 0.8. This suggests that the model's performance on the validation data is not as good as on the training data.
In the case of setup 2, both the train and validation loss curves exhibit a smooth descent without significant spikes or fluctuations. The validation loss curve starts at around 7 and follows a similar pattern as the train loss curve. It gradually decreases and converges around 1. This smoothness suggests that the model is consistently learning and generalizing well without encountering major obstacles or fluctuations in performance.

The accuracy curves for MobileNet V1 are presented in Fig. 4.6. For setup 1, The validation accuracy starts just below 0.9 and experiences some fluctuations with spikes during the training. It eventually settles around 0.86. While the accuracy is relatively high, the presence of spikes suggests that the model's performance on the validation data is less stable compared to the training accuracy. It indicates that the model might be overfitting to some extent and not generalizing optimally to unseen data. It demonstrates a higher train accuracy and some fluctuations in the validation accuracy, indicating potential overfitting.
In the case of setup 2, the validation accuracy starts around 0.7 and gradually increases to a range of 0.85-0.90. It remains relatively stable within this range, indicating that the model's performance on the validation data is consistent. The consistent improvement and relatively high accuracy suggest that the model is generalizing well to unseen data. It shows a consistently improving train accuracy and a stable validation accuracy within a relatively high range.

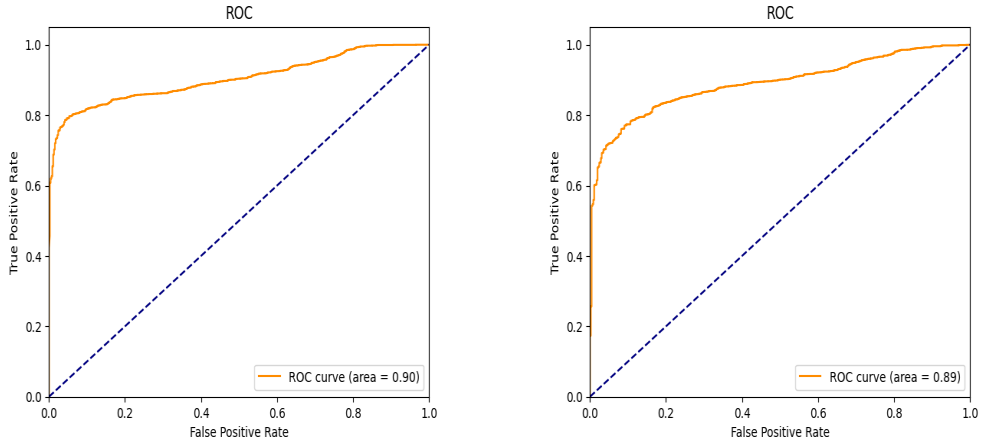


(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.7: MobileNet V1 confusion matrix for unbalanced data

The confusion matrix for MobileNet V1 is presented in Fig. 4.7. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 49.0% and the incorrect

classification is 51.0% **Class 1 (trout)**: The correct classification is 90.4% and the incorrect classification is 9.6%
For setup 2: **Class 0 (mature)**: The correct classification is 48.8% and the incorrect classification is 51.2% **Class 1 (trout)**: Correct classification is 91.4% and the incorrect classification is 8.6%



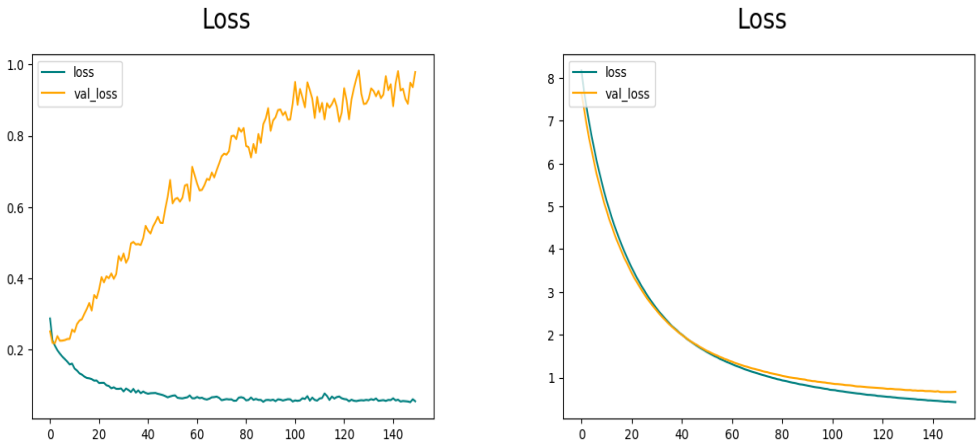
(a) Setup 1 **(b)** Setup 2

Figure 4.8: MobileNet V1 AUC score for unbalanced data

The AUC score for the ROC curve for MobileNet V1 is presented in Fig. 4.6. The AUC score for setup 1 is 0.90 and for setup 2 is 0.89. Based on the provided scores, MobileNet V1 exhibits a strong discriminatory power in distinguishing between different classes, particularly in setup 1 where it achieves a higher AUC score of 0.90.

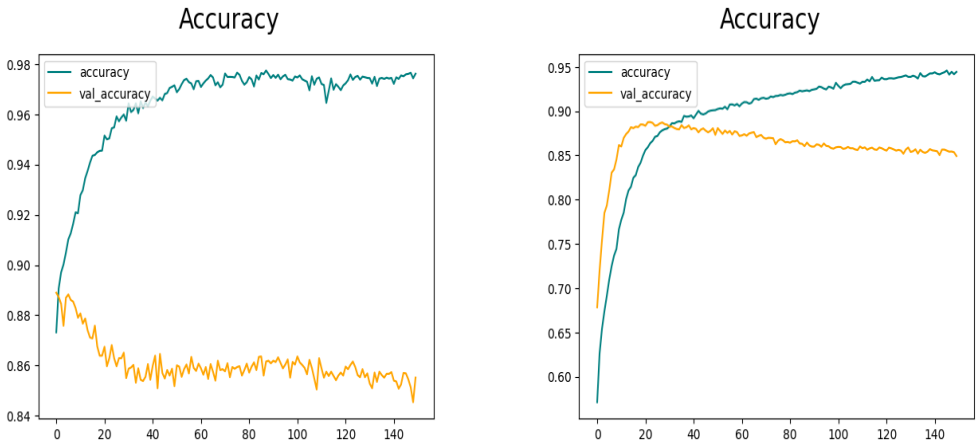
4.1.3 MobileNet V2

The results for MobileNet V2 for setup 1 and setup 2 using unbalanced dataset are presented here. This includes model performance details such as loss and accuracy curves along with performance evaluation using a confusion matrix and AUC score.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.9: MobileNet V2 loss curves for unbalanced data



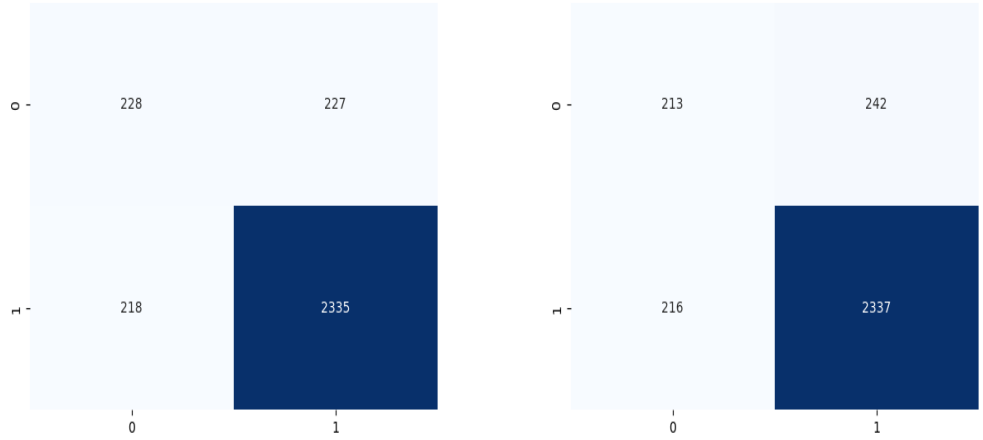
(a) Accuracy curves setup 1 **(b)** Accuracy curves setup 2

Figure 4.10: MobileNet V2 accuracy curves for unbalanced data

Figure 4.9 illustrates the loss curves for MobileNet V2. Analyzing these curves provides valuable insights into the model's learning and generalization abilities. For MobileNet V2, the train loss curve showcases a notable reduction, indicating that the model successfully learns from the training data. However, the increasing validation loss curve suggests a lack of generalization and potential overfitting. Despite starting just over 0.2, the validation loss does not decrease or stabilize, reaching a value of approximately 1.0. This indicates that the model's performance on the validation data is not as strong as on the training data.
In the case of setup 2, both the train and validation loss curves demonstrate a smooth descent without significant spikes or fluctuations. The validation loss curve initiates at around 8 and follows a similar pattern to the train loss curve.

Gradually, it decreases and converges around 1. This smoothness suggests that the model consistently learns and generalizes well without encountering major obstacles or fluctuations in performance. The smoothness of the loss curves in setup 2 implies that the model maintains stable learning and effective generalization, indicating its potential for reliable predictions.

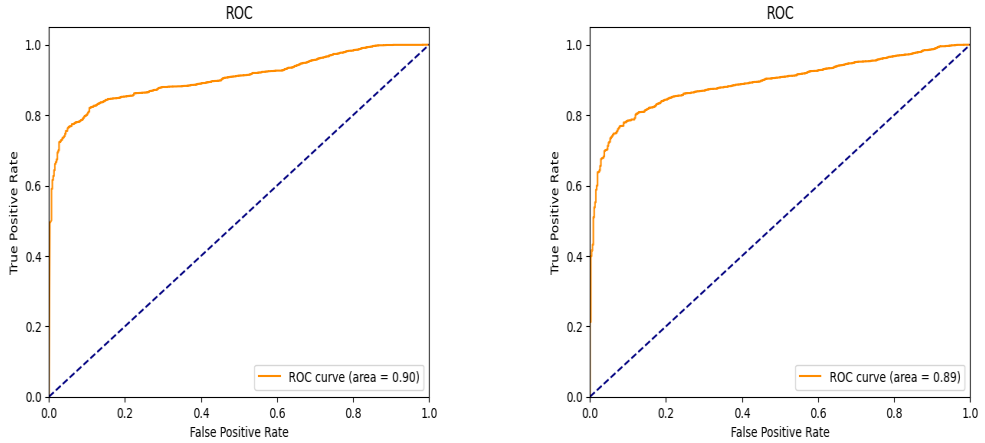
The accuracy curves for MobileNet V2 are presented in Fig. 4.10. For setup 1, The validation accuracy starts around 0.88 and experiences some spikes during the training. It eventually settles between the ranges 0.84 and 0.86. The presence of spikes suggests that the model's performance on the validation data is less stable compared to the training accuracy. It indicates that the model might be overfitting to an extent and is not able to generalize new data.
For setup 2, the validation accuracy starts below 0.7 and gradually increases to a range of 0.85-0.90. It remains relatively stable within this range, indicating that the model's performance on the validation data is consistent. The consistent improvement and relatively high accuracy suggest that the model is generalizing well to unseen data. It shows a consistently improving train accuracy and a stable validation accuracy within a relatively high range.



(a) Setup 1 **(b)** Setup 2

Figure 4.11: MobileNet V2 confusion matrix for unbalanced data

The confusion matrix for MobileNet V2 is presented in Fig. 4.11. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 50.11% and the incorrect classification is 49.89% **Class 1 (trout)**: The correct classification is 91.34% and the incorrect classification is 8.66%
For setup 2: **Class 0 (mature)**:The correct classification is 46.81% and the incorrect classification is 53.19% **Class 1 (trout)**: The correct classification is 91.47% and the incorrect classification is 8.53%



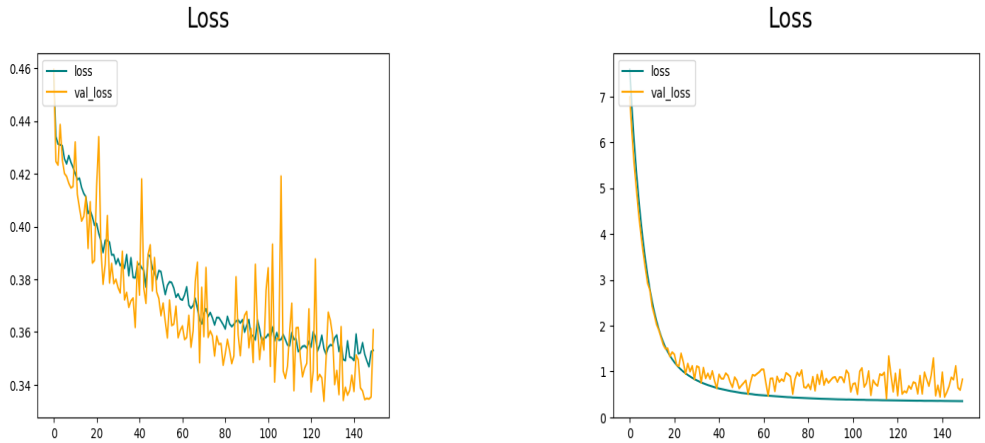
(a) Setup 1 **(b)** Setup 2

Figure 4.12: MobileNet V2 AUC score for unbalanced data

The AUC score represents the overall performance of a classification model, considering both sensitivity and specificity. Based on the provided scores, the AUC score for setup 1 is 0.90 and for setup 2 is 0.89 which states that the model has a high chance of classifying correctly.

4.1.4 MobileNet V3

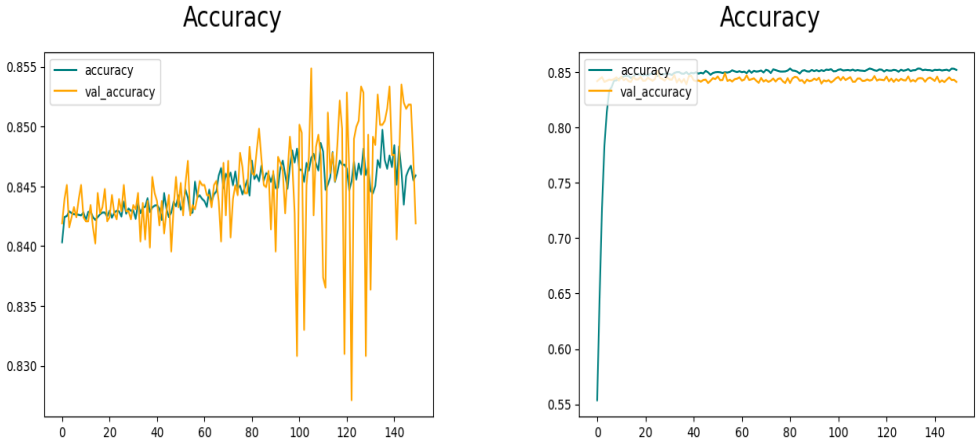
This section presents the results for MobileNet V3 for setup 1 and setup 2 when using the unbalanced dataset. Loss and Accuracy curves for both setups are presented which helps to understand the model performance while training and then for the evaluation, the confusion matrix, and AUC score are presented.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.13: MobileNet V3 loss curves for unbalanced data

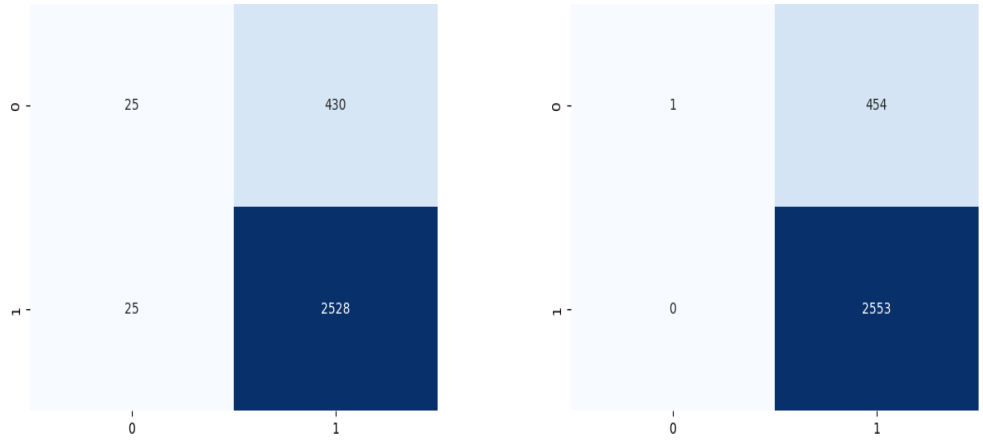
The loss curves for MobileNet V3 are presented in Fig.4.13. MobileNet V3 shows quite a lot of fluctuation and spikes as it starts from around 0.46 and settles in the range of 0.38 and 0.34. This also suggests a lack of generalization to unseen data and potential overfitting. This suggests that the model's performance on the validation data is not as good as on the training data. The training loss also has fluctuations and short spikes while it reduces gradually which can suggest that the learning rate is quite high and the model is not learning efficiently. It can also suggest that the chosen model architecture may not be suitable for the given dataset.
In the case of setup 2, both the train and validation loss curves exhibit a smoother descent as compared to setup 1, although the validation loss still has significant spikes and fluctuations. The training loss has a smooth descent down from a little over 7 units to a range between 0 and 1. The validation loss curve starts at above 7 and follows a similar pattern as the train loss curve but has some significant spikes. It does not converge in the run of 150 epochs but the loss curve suggests that the model is consistently learning and can generalize better than setup 1.



(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.14: MobileNet V3 accuracy curves for unbalanced data

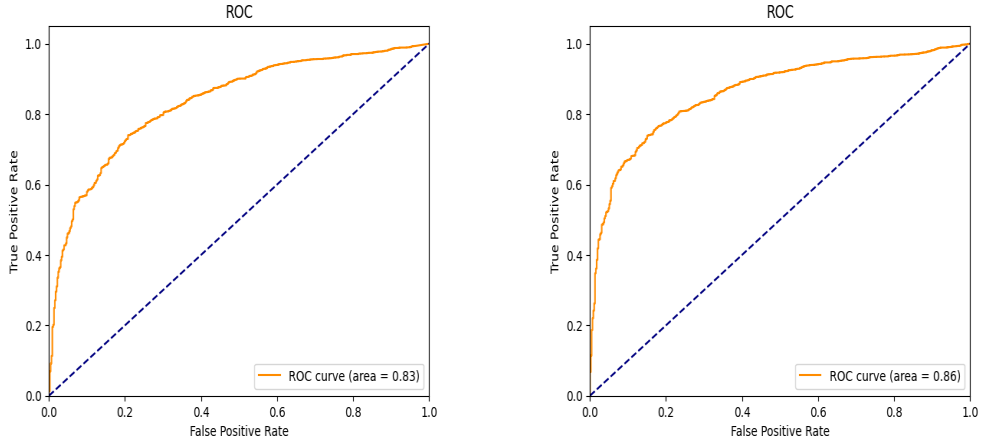
The accuracy curves for MobileNet V3 are presented in Fig. 4.14. For setup 1, The validation accuracy starts around 0.84 and experiences heavy fluctuations and spikes during the training. It is highly inconsistent and suggests that the model is not able to generalize the new data. The train loss is also not a smooth curve and has quite a lot of fluctuations and spikes. This also suggests that the model is not able to learn efficiently and probably has a high learning rate.
For setup 2, the validation accuracy and the training accuracy remains relatively stable within the range of 0.80 and 0.85, much closer to 0.85. This indicates that the model's performance on the validation data is consistent. The consistent improvement and stable curve suggest that the model is generalizing well to unseen data.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.15: MobileNet V3 confusion matrix for unbalanced data

The confusion matrix for MobileNet V3 is presented in Fig. 4.15. Based on the provided confusion matrices for setup 1 and setup 2, the correct and wrong classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 5.49% and the incorrect classification is 94.51% **Class 1 (trout)**:The correct classification is 99.02% and the incorrect classification is 0.98%
For setup 2: **Class 0 (mature)**: The correct classification is 0.22% and the incorrect classification is 99.78% **Class 1 (trout)**: The correct classification is 100% and the incorrect classification is 0%.



(a) Setup 1 **(b)** Setup 2

Figure 4.16: MobileNet V3 AUC score for unbalanced

The AUC score for MobileNet V3 is presented in Fig. 4.16. Based on the provided scores, the AUC score for setup 1 is 0.83 and for setup 2 is 0.86. This states that

the models possess a relatively high classification performance of a classification model, considering both sensitivity and specificity.

4.2 Results with Balanced Data

In this section, the results are presented when each of the deep learning models is trained using an almost balanced dataset. This has been achieved in two cases: one where the amount of data from the majority class (trout) has been cut down to almost match with the data with the minority class (mature) and the other case where the data in the minority class (mature) has been augmented or artificially increased to almost match the amount of data for the majority class (trout). After having two datasets, one without any augmentation and the other with augmentation, the results of the experiments are presented.

4.2.1 Without Data Augmentation

The first attempt to almost balance the data was without applying any augmentation and reducing the amount of data for the majority class (trout). Using a Python script, a random generator has been used to select 5000 images from the trout class, and along with the original 4696 images from the mature class, a dataset of total **9696** images was created. The data distribution is presented in Table 4.4.

Label	**Class**	**Images**	**Distribution**
Mature	0	4696	≈49%
Trout	1	5000	≈51%

Table 4.4: Balanced dataset without augmentation

Tables 4.5 and 4.6 provide detailed results for the two different setups (3.6.1 and 3.6.2). These results offer valuable insights to the performance and characteristics of each model. With this data, we will be able to analyze the impact of different hyperparameters on the models' performance and training, as well as understand if there is any impact by having a balanced dataset.

Model Architecture Setup 1 (3.6.1)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3 Large**
Batch Size	64	64	64	64
Total Data	152	152	152	152
Train Size	106 (70%)	106 (70%)	106 (70%)	106 (70%)
Validation Size	30 (20%)	30 (20%)	30 (20%)	30 (20%)
Test Size	16 (10%)	16 (10%)	16 (10%)	16 (10%)
Total Params	24,637,826	3,754,690	2,914,882	4,883,330
Trainable Params	1,050,114	525,826	656,898	656,898
Learning Rate	0.001	0.001	0.001	0.001
Epochs	150	150	150	150
Training Time (mins)	48	28	32	32

Model Architecture Setup 2 (3.6.2)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Batch Size	64	64	64	64
Total Data	152	152	152	152
Train Size	106 (70%)	106 (70%)	106 (70%)	106 (70%)
Validation Size	30 (20%)	30 (20%)	30 (20%)	30 (20%)
Test Size	16 (10%)	16 (10%)	16 (10%)	16 (10%)
Total Params	24,639,874	3,756,738	2,916,930	4,885,378
Trainable Params	1,051,138	526,850	657,922	657,922
Learning Rate	0.00001	0.00001	0.00001	0.00001
Epochs	150	150	150	150
Training Time (mins)	50	28	32	32

Table 4.5: Model configuration for balanced dataset without augmentation

In both setup 1 and setup 2, several factors were examined and compared across the models. Regarding the batch size, all models employed a consistent value of 64, ensuring uniformity in the training process for both setups. For dataset sizes, the training, validation, and test sizes remained constant across all models in both setup 1 and setup 2. The dataset consisted of 106 batches for training, 30 batches for validation, and 16 batches for testing. This standardized dataset allocation facilitates fair comparisons between the models.

Analyzing the total and trainable parameters, a notable disparity was observed across the models. In setup 1, ResNet-50 exhibited the highest number of parameters, with a total of 24,637,826 and 1,050,114 trainable parameters. On the other hand, MobileNet V2 had the lowest number of parameters, with 2,914,882 total parameters and 525,826 trainable parameters. Similarly, in setup 2, ResNet-50 had the highest number of parameters (24,639,874), while MobileNet V2 had the lowest (2,916,930). These variations in parameter count suggest differences in model complexity and capacity to capture intricate patterns and features.

In terms of learning rate and epochs, setup 1 employed a fixed learning rate of 0.001 throughout all models for 150 epochs. Conversely, in setup 2, all models were trained using a learning rate of 0.00001 for the same duration of 150 epochs. This learning rate and epoch configuration allow for consistent training conditions within each setup.

Examining the training time, setup 1 demonstrated that ResNet-50 had the longest training duration, taking 48 minutes. In contrast, MobileNet V1, V2, and V3 had shorter training times ranging from 28 to 32 minutes. In setup 2, the training times ranged from 28 to 50 minutes, with MobileNet V1 exhibiting the shortest training time. These differences in training time can be attributed to variations in model architecture and complexity.

After comprehensively exploring the configurations of each model in both setup 1 and setup 2, it becomes crucial to delve into their performance and the results they achieved. By closely examining key evaluation metrics, insights into the overall effectiveness of these models can be better understood in the given context. Table 4.6 provides a comprehensive overview of the evaluation metrics for each model.

Evaluation Metrics				
Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Model Architecture Setup 1 (3.6.1)				
Accuracy	76.2%	82.5%	**82.6%**	71.5%
Precision	72.8%	85.7%	**86.5%**	68.8%
Recall	**87.8%**	80.3%	79.6%	84.5%
F1 Score	0.79	**0.82**	**0.82**	0.75
Model Architecture Setup 2 (3.6.2)				
Accuracy	72.9%	**85.9%**	83.7%	53.8%
Precision	69.1%	**87.4%**	85.4%	53.4%
Recall	88.9%	85.9%	83.3%	**99.4%**
F1 Score	0.77	**0.86**	0.84	0.69

Table 4.6: Evaluation metrics for balanced dataset without augmentation

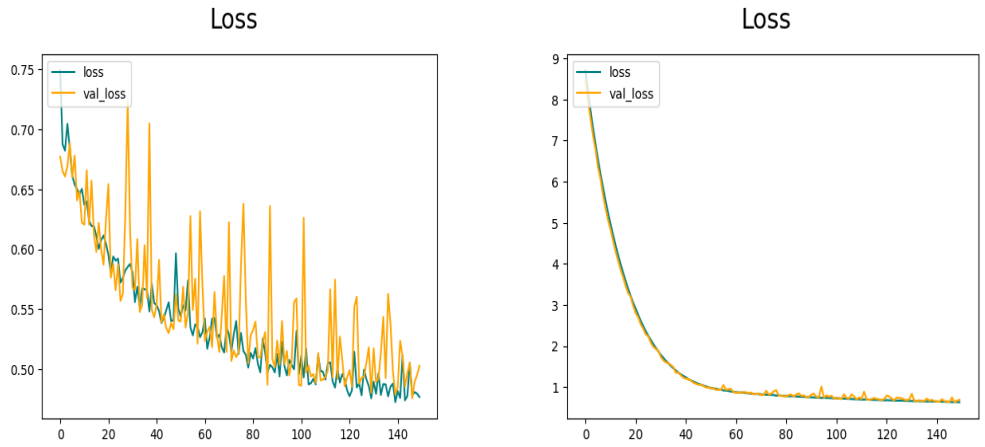
In model setup 1, the evaluation metrics reveal that the models exhibit a decent overall performance. The accuracy scores range from 71.5% for MobileNet V3 to 82.6% for MobileNet V2, indicating that the models are able to make correct predictions with a moderate level of accuracy. However, the precision scores range from 68.8% for MobileNet V3 to 86.5% for MobileNet V2, suggesting that there is room for improvement in correctly identifying positive instances. On the other hand, the recall scores range from 79.6% for MobileNet V2 to 87.8% for ResNet-50, indicating that the models have a good ability to capture positive instances. The F1 scores, which reflect the balance between precision and recall, are generally high, ranging from 0.75 to 0.82, indicating a reasonable trade-off between these two metrics.

The models in setup 2 exhibit a wider variation in performance. The accuracy scores range from 53.8% for MobileNet V3 to 85.9% for MobileNet V1, indicating a significant difference in their ability to make correct predictions. The precision scores also vary, ranging from 53.4% for MobileNet V3 to 87.4% for MobileNet V1, suggesting that the models have different levels of accuracy in identifying positive instances. The recall scores, on the other hand, range from 83.3% for MobileNet V2 to 99.4% for MobileNet V3, indicating that the models are generally effective in capturing positive instances. The F1 scores in setup 2 range from 0.69 to 0.86, suggesting that the models exhibit varying levels of balance between precision and recall. Overall, the evaluation metrics highlight the variation in performance across the models in setup 2, emphasizing the need to carefully consider their strengths and weaknesses in differentiating positive instances.

To further understand the performance of each model under the specific conditions, the results for ResNet-50, MobileNet V1, MobileNet V2 and MobileNet V3 are presented.

ResNet-50

This section presents the results for ResNet-50 using both setup 1 and setup 2 for an almost balanced dataset without augmentation. The loss and accuracy curves highlight the training process of the model while the confusion matrix and the AUC score will shed light on the evaluation of the model when exposed to the test dataset.

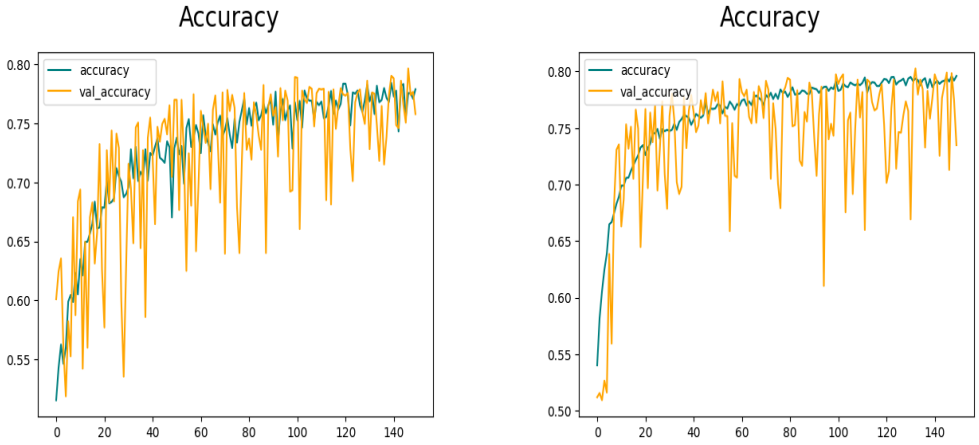


(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.17: ResNet-50 loss curves for balanced data without augmentation

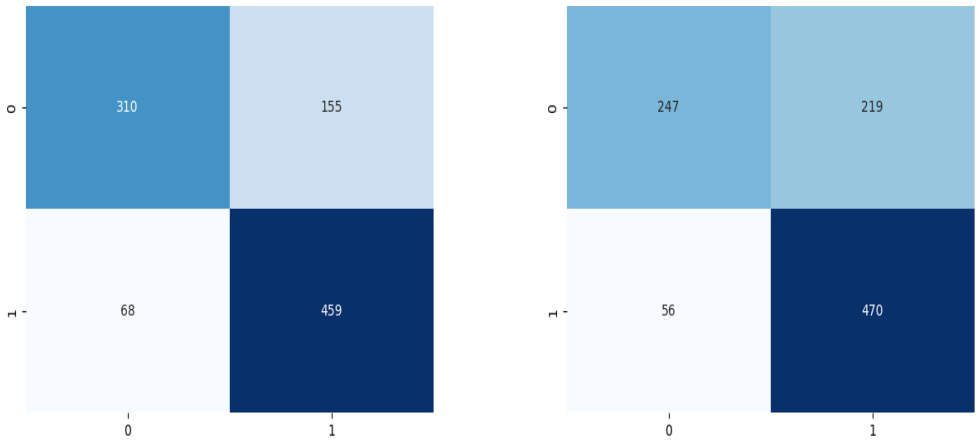
The loss curves for ResNet-50, as shown in Fig. 4.17, provide valuable insights into the model's performance. In setup 1, the spikes observed in the validation loss curve indicate that the model's ability to generalize to unseen data, represented by the validation set, may not be consistent. This suggests that the model may be overfitting to the training data, failing to effectively capture the underlying patterns present in the data. The validation loss curve starts at approximately 0.7 and fluctuates irregularly, descending slightly below 0.75. These fluctuations could be attributed to the model's sensitivity to variations or noise within the validation set, resulting in inconsistent performance. In contrast, for setup 2, the loss curve begins at a higher value of around 9 and consistently decreases to less than 1. This indicates that the model quickly learns and adjusts its predictions during the initial training epochs. The decreasing trend of the loss curve suggests that the model continues to improve its performance over time. Although minor spikes can be observed in the validation curve, the overall smoothness of the loss curve implies that the model's performance remains relatively stable during training. This indicates that the model is able to generalize well to unseen data and make

consistent predictions.



(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.18: ResNet-50 accuracy curves for balanced data without augmentation



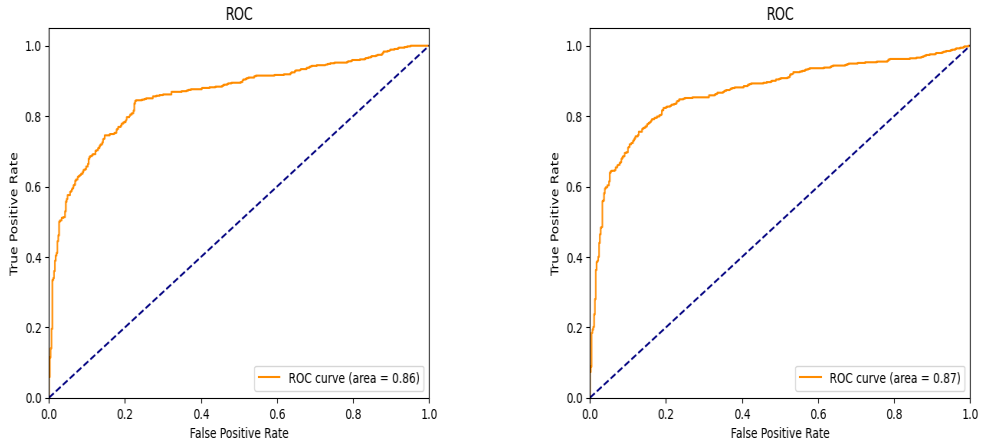
(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.19: ResNet-50 confusion matrix for balanced data without augmentation

The accuracy curves for ResNet-50, as depicted in Fig. 4.18, provide insights into the model's performance across different setups. In setup 1, the accuracy curve exhibits a gradual increase from approximately 0.55 to just below 0.80 over the 150 epochs. However, the curve shows numerous irregularities, characterized by sharp spikes in the validation curve. Similarly, the training curve displays fluctuations, indicating that the model struggles to effectively learn new features. The model's performance experiences sudden drops and increases, suggesting difficulties in converging to a stable and optimal solution. In setup 2, where a

reduced learning rate was employed, the training curve exhibits less fluctuation and appears smoother. It steadily increases from around 0.55 to approximately 0.80, indicating a positive learning trend. However, the validation curve remains irregular, indicating that the model struggles to generalize to unseen data. The validation curve starts around 0.50 and ascends irregularly to approximately 0.70 after the 150 epochs. This indicates that the model fails to maintain a satisfactory level of performance on the validation data. These observations emphasize the need for further analysis and potential adjustments to enhance the model's performance and generalization capabilities.

The confusion matrix for ResNet-50 is presented in Fig. 4.19. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 66.67% and the incorrect classification is 33.33% **Class 1 (trout)**: The correct classification is 87.13% and the incorrect classification is 12.87%
For setup 2: **Class 0 (mature)**: The correct classification is 52.99% and the incorrect classification is 47.01% **Class 1 (trout)**: The correct classification is 89.39% and the incorrect classification is 10.61%



(a) Setup 1 **(b)** Setup 2

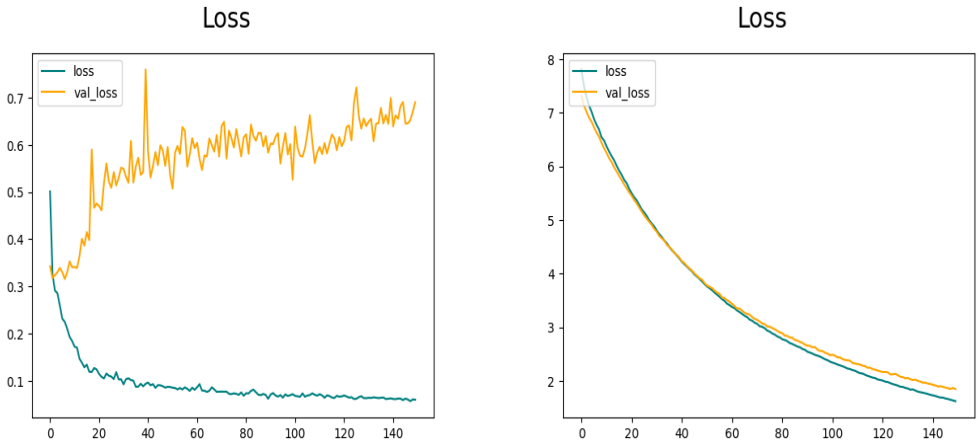
Figure 4.20: ResNet-50 AUC score for balanced data without augmentation

The AUC score for the ROC curve is presented in Fig. 4.20. Based on the provided scores, the AUC score for setup 1 is 0.86 and for setup 2 is 0.87. This suggests that the model has relatively high effectiveness and will have a high chance to correctly classify the images.

MobileNet V1

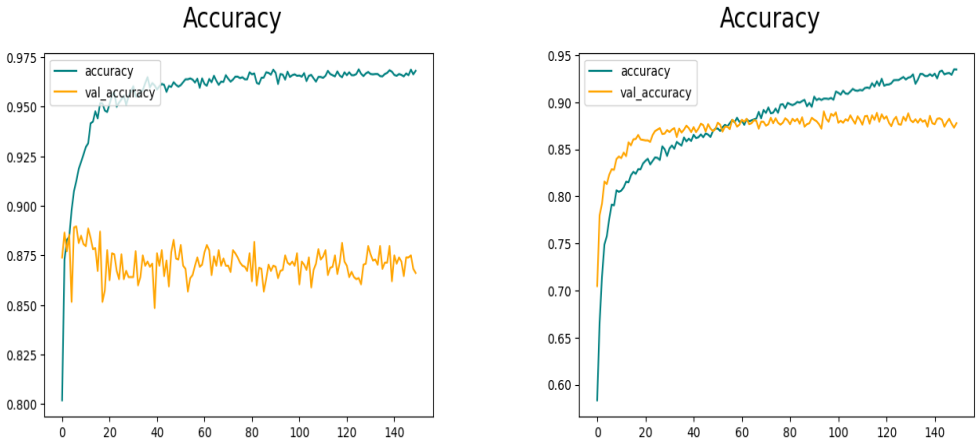
This section presents the results when MobileNet V1 was used for the classification task using setups 1 and 2 for an almost balanced dataset without augmentation.

The model's performance while training is presented through the loss and accuracy curves and the evaluation of the model is presented using the confusion matrix and AUC score.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.21: MobileNet V1 loss curves for balanced data without augmentation

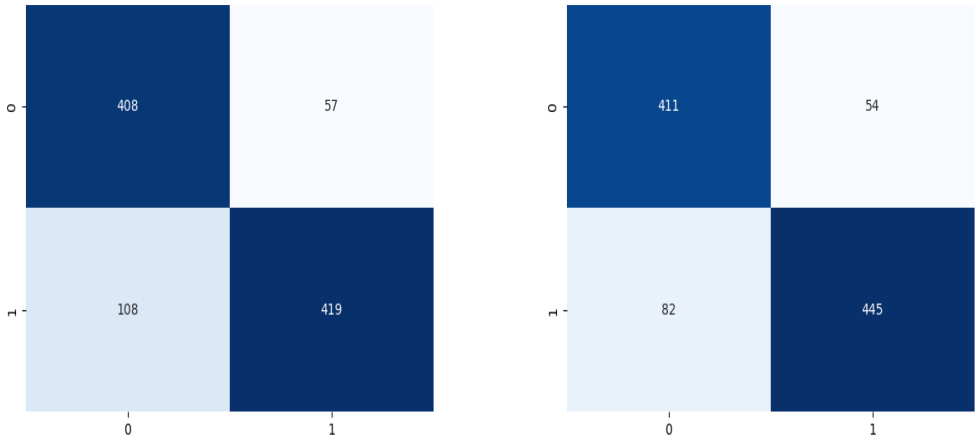


(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.22: MobileNet V1 accuracy curves for balanced data without augmentation

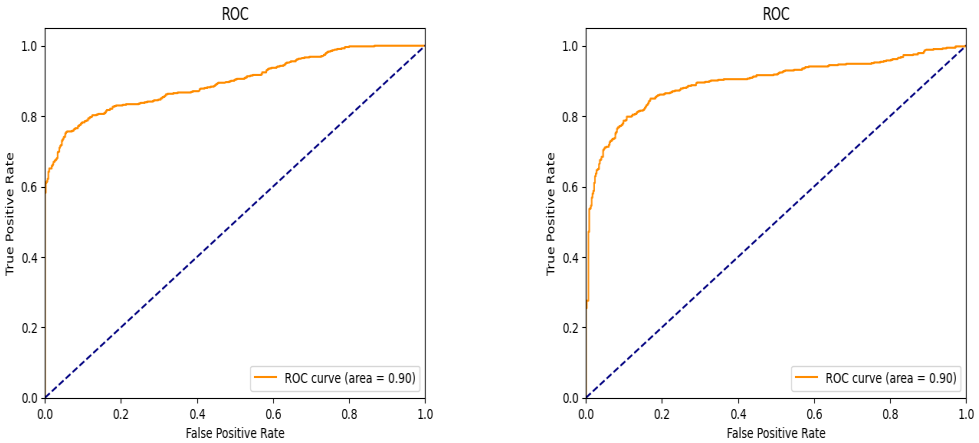
The loss curves for MobileNet V1, as shown in Fig. 4.21, provide valuable insights into the model's performance in different setups. In setup 1, the training curve exhibits a gradual decrease with some minor spikes, indicating successful learning from the training data. However, the validation curve starts at a higher value and shows frequent spikes, failing to converge. This suggests that the model may be overfitting to the training data and lacking appropriate regularization tech-

niques. The spikes in the validation curve further emphasize the need to address overfitting issues and improve the model's generalization capabilities. In contrast, setup 2 demonstrates a different behavior. The loss curve begins at a relatively high value, typically ranging between 8 and 9, and consistently decreases over time. This indicates that the model quickly adapts and adjusts its predictions during the initial training epochs, resulting in improved performance. The decreasing trend of the loss curve signifies ongoing improvement, suggesting that the model continues to learn and generalize well. This observation indicates that the model performs better in terms of generalization to unseen data and exhibits greater stability throughout the training process.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.23: MobileNet V1 confusion matrix for balanced data without augmentation



(a) Setup 1 **(b)** Setup 2

Figure 4.24: MobileNet V1 AUC score

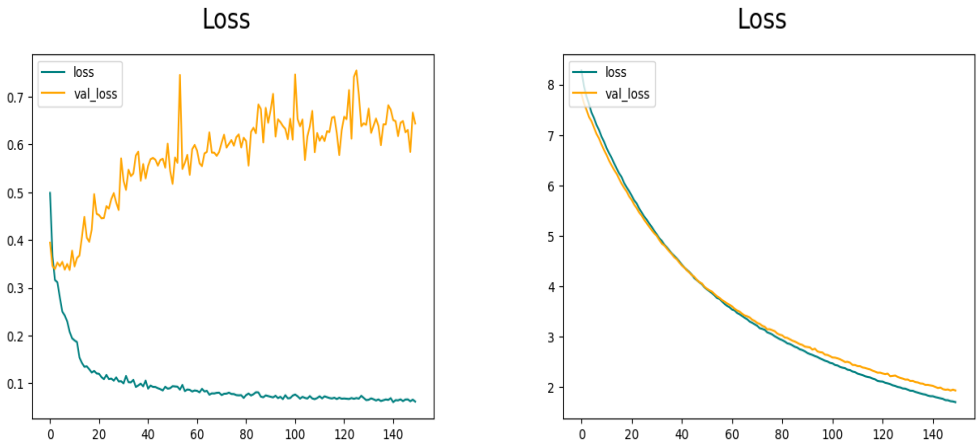
The accuracy curves for MobileNet V1, as depicted in Fig. 4.22, offer valuable insights into the model's performance across different setups. In setup 1, the train accuracy curve demonstrates a consistent upward trend, gradually increasing from approximately 0.8 to a range between 0.95 and 0.975 over the course of 150 epochs. However, the validation accuracy curve exhibits irregularities with spikes observed throughout. These spikes indicate that the model is not effectively learning new features, leading to inconsistent performance. The sudden drops and increases in the validation accuracy curve further suggest that the model faces challenges in converging toward a stable and optimal solution. This inconsistency highlights the need for further improvements in the model's ability to generalize and learn new patterns effectively in setup 1. Conversely, in setup 2, characterized by a reduced learning rate, both the training and validation curves display reduced fluctuations and smoother progress. The training curve steadily increases from around 0.60 to a range between 0.85 and 0.95, indicating progressive learning and improved performance over time. The validation curve suggests that the model's performance remains stable and consistent when evaluated on unseen data. This observation signifies the model's ability to generalize well and maintain a reasonable level of accuracy in setup 2.

The confusion matrix for MobileNet V1 is presented in Fig. 4.23. Based on the provided confusion matrices for setup 1 and setup 2, the correct and wrong classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 87.74% and the incorrect classification is 12.26% **Class 1 (trout)**: The correct classification is 79.5% and the incorrect classification is 20.5%
For setup 2: **Class 0 (mature)**: The correct classification is 88.38% and the incorrect classification is 11.62% **Class 1 (trout)**: The correct classification is 84.4% and the incorrect classification is 15.6%

The AUC score for the ROC curve is presented in Fig. 4.24. Based on the provided scores, the AUC score for setup 1 is 0.90 and for setup 2 is 0.90. This indicates a reasonably high value for the model in classifying the two different classes.

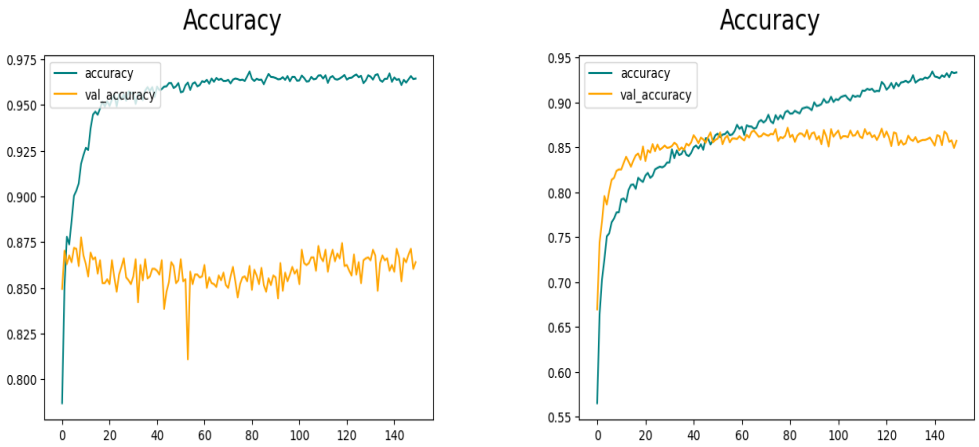
MobileNet V2

This section presents the results of MobileNet V2 with setup 1 and setup 2 when using the almost balanced dataset without augmentation. To understand the model's training, the loss and accuracy curves are presented and to evaluate the performance of the model, the confusion matrix and AUC score is presented.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.25: MobileNet V2 loss curves for balanced data without augmentation



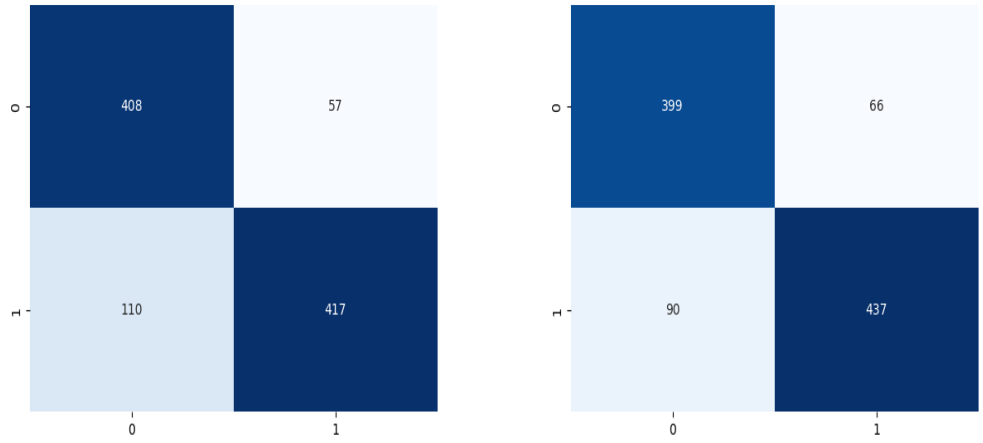
(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.26: MobileNet V2 accuracy curves for balanced data without augmentation

The loss curves for MobileNet V2 are presented in Fig. 4.25. For setup 1, the train loss curve exhibits a gradual decrease from approximately 0.5 to below 0.1, with minor spikes along the way. However, the validation loss curve starts at a higher value, shows frequent spikes, and fails to converge. These observations suggest that the model may be overfitting, as it struggles to generalize to unseen data. The spikes in the validation loss curve further indicate a lack of appropriate regularization techniques, leading to inconsistent performance and potentially limited generalization capabilities. As for setup 2, the loss curve begins at a relatively high value between 8 and 9 but consistently decreases over time. This downward trend signifies that the model rapidly learns and adjusts its predic-

tions during the initial training epochs. The continued decrease in the loss curve indicates ongoing improvement in the model's performance. These observations suggest that the model exhibits better generalization abilities to unseen data and demonstrates greater stability in setup 2.

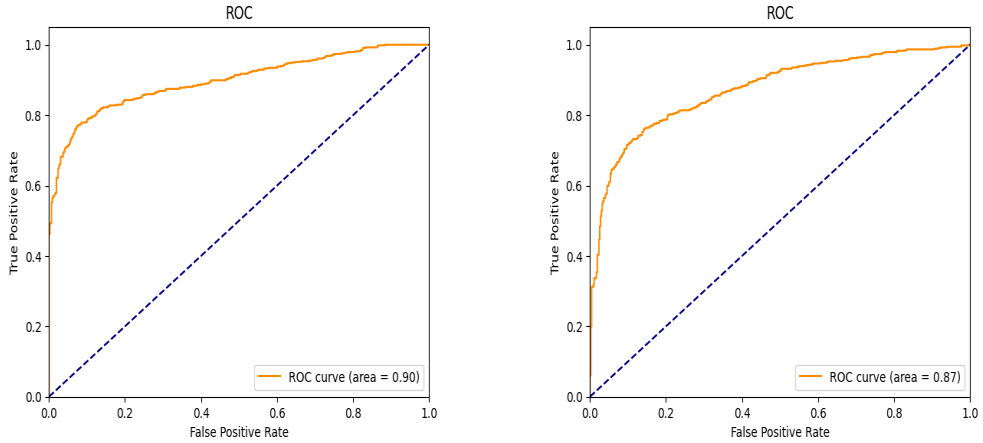
The accuracy curves for MobileNet V2 are presented in Fig. 4.26. In setup 1, the train accuracy curve shows a steady and consistent upward trend. Starting at approximately 0.8, it progressively climbs to a range of 0.95 to 0.975 over the course of 150 epochs. However, the validation accuracy curve exhibits irregular spikes throughout its progression. These spikes indicate that the model struggles to effectively learn new features, leading to inconsistent performance. On the other hand, in setup 2, a reduced learning rate promotes smoother and more consistent progress in both the training and validation curves. The training curve demonstrates a gradual increase, starting from around 0.55 and steadily reaching a range between 0.85 and 0.95. Notably, the validation curve suggests that the model achieves stability and consistently performs well on unseen data. This implies that the model in setup 2 exhibits improved generalization capabilities and maintains a higher level of performance.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.27: MobileNet V2 confusion matrix for balanced data without augmentation

The confusion matrix for MobileNet V2 is presented in Fig. 4.27. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 87.74% and the incorrect classification is 12.26% **Class 1 (trout)**: The correct classification is 79.1% and the incorrect classification is 20.9%
For setup 2: **Class 0 (mature)**: The correct classification is 85.8% and the incorrect classification is 14.2% **Class 1 (trout)**: The correct classification is 82.9% and the incorrect classification is 17.1%



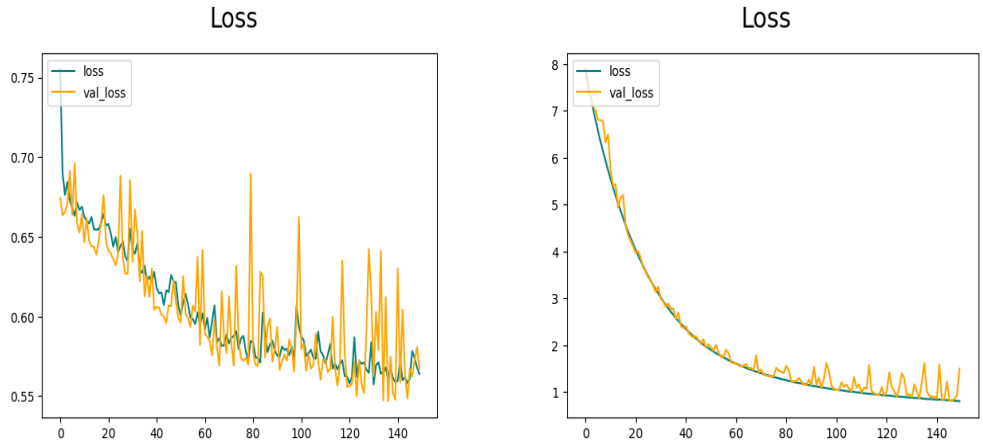
(a) AUC score setup 1 **(b)** AUC score setup 2

Figure 4.28: MobileNet V2 AUC score for balanced data without augmentation

The AUC score for MobileNet V2 is presented in the Fig. 4.28. Assessing the provided scores, we observe that the AUC score for setup 1 is 0.90 and for setup 2 is 0.87 which states that in both setups, the model has high effectiveness and precision in classifying the data.

MobileNet V3

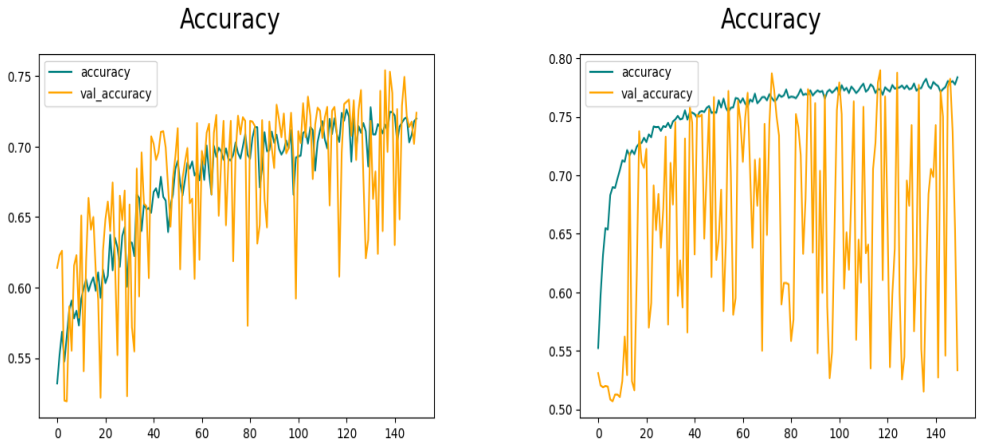
In this section, the results of MobileNet V3 with setup 1 and setup 2 are presented when using an almost balanced dataset without any kind of data augmentation. In order to assess the performance while training, the loss and accuracy curves are presented and the confusion matrix and AUC score are presented as the evaluation metrics.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.29: MobileNet V3 loss curves for balanced data without augmentation

The loss curves for MobileNet V3 are presented in Fig. 4.29. For setup 1, the train loss curve exhibits a gradual reduction, starting from approximately 0.75 and converging to a range between 0.55 and 0.6. However, the validation loss curve starts around 0.65 and displays numerous fluctuating spikes throughout its progression. The lack of convergence and generalization in the validation curve indicates that the model struggles to fit well into unseen data. These spikes further suggest that the model may not have sufficient regularization techniques in place to ensure consistent performance. In this setup, the model's performance is suboptimal, as it fails to effectively capture the underlying patterns in the validation data. As for setup 2, the loss curve starts at a relatively higher value, ranging between 7 and 8, but consistently decreases over the training epochs. This decreasing trend indicates that the model rapidly learns and adjusts its predictions during the initial stages of training. The model continues to improve its performance over time, as evidenced by the decreasing loss values. However, it is worth noting that the validation loss curve still exhibits fluctuations towards the end of the training process, suggesting that the model struggles to fit well to unseen data in this setup as well.

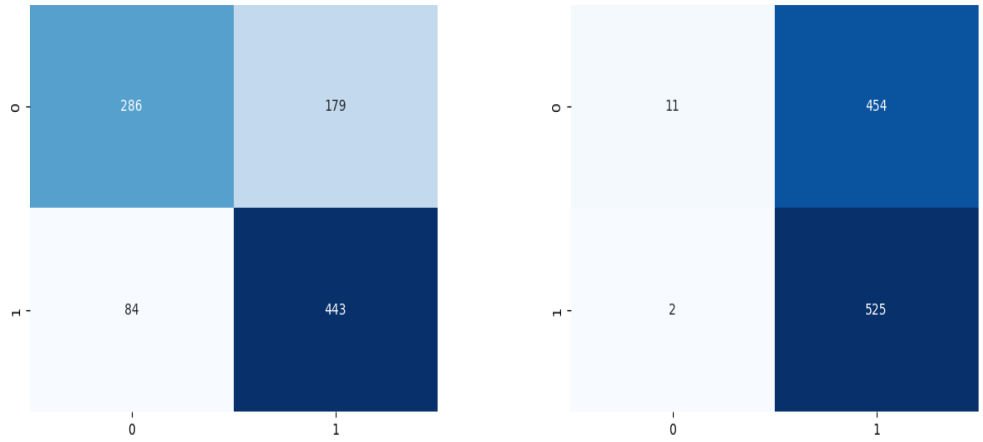


(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.30: MobileNet V3 accuracy curves for balanced data without augmentation

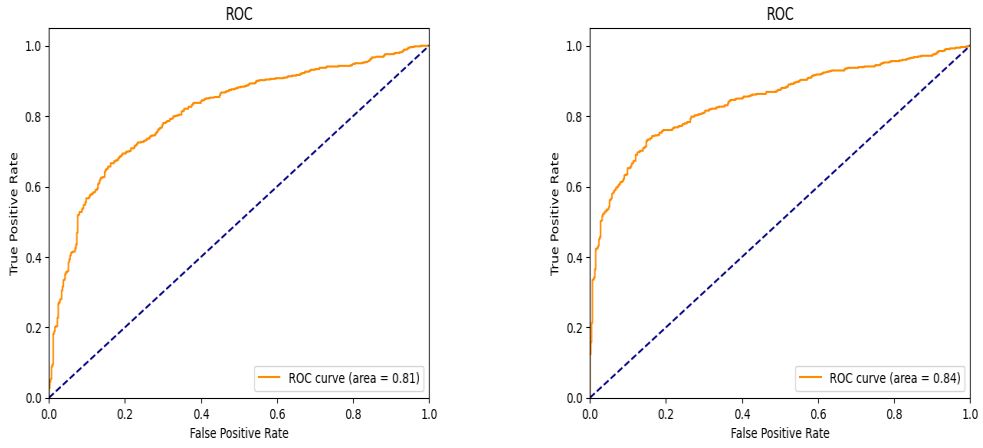
The accuracy curves for MobileNet V3 are presented in Fig. 4.30. In setup 1, the train accuracy curve displays fluctuations, starting from approximately 0.55 and gradually reaching a range between 0.70 and 0.75 over 150 epochs. However, the validation accuracy curve exhibits highly irregular spikes throughout its progression. These irregularities indicate that the model struggles to effectively learn new features, resulting in sudden drops and increases in performance. This suggests difficulties in converging to a stable and optimal solution. The inconsistent behavior of the validation accuracy curve further emphasizes the model's challenge in achieving consistent performance. On the other hand, in setup 2,

despite the reduced learning rate, the validation accuracy curve begins at around 0.5 and faces significant fluctuations ranging between 0.5 and 0.8. On the other hand, the training curve shows a gradual increase, starting from approximately 0.55 and reaching a range between 0.75 and 0.8. However, the validation curve's instability suggests that the model's performance lacks consistency and struggles to maintain a reasonable level of accuracy across unseen data.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.31: MobileNet V3 confusion matrix for balanced data without augmentation



(a) AUC score setup 1 **(b)** AUC score setup 2

Figure 4.32: MobileNet V3 AUC score for balanced data without augmentation

The confusion matrix for MobileNet V3 is presented in Fig. 4.31. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.

For setup 1: **Class 0 (mature)**: The correct classification is 61.5% and the incorrect classification is 38.5% **Class 1 (trout)**: The correct classification is 84.1% and the incorrect classification is 15.9%
For setup 2: **Class 0 (mature)**: The correct classification is 2.36% and the incorrect classification is 97.64% **Class 1 (trout)**: The correct classification is 99.6% and the incorrect classification is 0.4%

The AUC score for MobileNet V3 is presented in Fig. 4.32. Assessing the provided scores, it is observed that the AUC score for setup 1 is 0.81 and for setup 2 is 0.84. This indicates a decent effectiveness and precision of the model in the classification of the data.

4.2.2 With Augmentation

After having stated results for each model trained on a balanced dataset without data augmentation, the next setup is where the dataset is almost balanced across the two classes by increasing the images on the mature class (which is the minority class) by using data augmentation 3.3. Over 20,000 images were generated using augmentation and including the original images, a dataset was prepared. A total of **50882** images were used in this setup. The data distribution is presented in Table 4.7.

Label	**Class**	**Images**	**Distribution**
Mature	0	25612	≈50%
Trout	1	25270	≈50%

Table 4.7: Balanced dataset with data augmentation

Tables 4.8 and 4.9 provides detailed results for the two different model setups, setup 1 (3.6.1) and setup 2 (3.6.2). These results offer valuable insights into the performance and characteristics of each model. The data helps to analyze the impact of different hyperparameters on the models' performance and training as well as understand if there is any impact on the overall accuracy by having an increased dataset.

Model Architecture Setup 1 (3.6.1)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Batch Size	64	64	64	64
Total Data	796	796	796	796
Train Size	557 (70%)	557 (70%)	557 (70%)	557 (70%)
Validation Size	159 (20%)	159 (20%)	159 (20%)	159 (20%)
Test Size	80 (10%)	80 (10%)	80 (10%)	80 (10%)
Total Params	24,637,826	3,754,690	2,914,882	4,883,330
Trainable Params	1,050,114	525,826	656,898	656,898
Learning Rate	0.001	0.001	0.001	0.001
Epochs	150	150	150	500
Training Time (mins)	247	158	145	584

Model Architecture Setup 2 (3.6.2)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Batch Size	64	64	64	64
Total Data	796	796	796	796
Train Size	557 (70%)	557 (70%)	557 (70%)	557 (70%)
Validation Size	159 (20%)	159 (20%)	159 (20%)	159 (20%)
Test Size	80 (10%)	80 (10%)	80 (10%)	80 (10%)
Total Params	24,639,874	3,756,738	2,916,930	4,883,330
Trainable Params	1,051,138	526,850	657,922	656,898
Learning Rate	0.00001	0.00001	0.00001	0.0001
Epochs	150	150	150	1000
Training Time (mins)	240	162	144	870

Table 4.8: Model configuration for the balanced dataset with augmentation

The experimental setup for the models in both setup 1 and setup 2 involved several key factors. Firstly, all models were trained using a batch size of 64, ensuring consistency in the training process across the setups. Regarding dataset sizes, the train, validation, and test sizes remained constant for all models in both setups. Specifically, the train set consisted of 557 batches, the validation set had 159 batches and the test set contained 80 batches. This ensured a standard distribution of data for evaluation and testing purposes.

In terms of the model parameters, there were variations between the different models. For setup 1, the total number of parameters ranged from 2,914,882 for MobileNet V2 to 24,637,826 for ResNet-50. Similarly, the number of trainable parameters varied, with MobileNet V1 having the lowest at 525,826 and ResNet-50 having the highest at 1,050,114. In setup 2, the total parameters followed a similar pattern, with MobileNet V2 having the lowest and ResNet-50 having the highest while in the case of trainable parameters, MobileNet V1 had the lowest and ResNet-50 having the highest.

The learning rate and number of epochs also differed for each model. In setup 1, the learning rate was set at 0.001 for 150 epochs, except for MobileNet V3, which was trained for 500 epochs as an experimental variation. In setup 2, all models except MobileNet V3 were trained with a learning rate of 0.00001 for 150 epochs, while MobileNet V3 was trained for 1000 epochs with a learning rate of 0.0001 as a part of experimental variation.

The training times varied across the models and setups. In setup 1, MobileNet V3 had the longest training time at 584 minutes due to the extended number of epochs, while ResNet-50 took longer to train compared to MobileNet V1 and V2. In setup 2, the training time for MobileNet V3 was 870 minutes due to the longer duration of training. ResNet-50, MobileNet V1, and MobileNet V2 had training times ranging from 144 to 240 minutes, with MobileNet V2 having the shortest training time.

These factors such as batch size, dataset sizes, model parameters, learning rate, number of epochs and training time were considered and adjusted accordingly to ensure consistency in the experimental setup and gain valuable insights from the models' performance during evaluation.

Evaluation Metrics				
Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Model Architecture Setup 1 (3.6.1)				
Accuracy	84.5%	**90.5%**	89.4%	75.5%
Precision	85.1%	**91.7%**	90.1%	68.2%
Recall	82.9%	88.6%	88.1%	**93.6%**
F1 Score	0.83	**0.89**	**0.89**	0.79
Model Architecture Setup 2 (3.6.2)				
Accuracy	80.5%	**90.9%**	90.6%	79.6%
Precision	73.3%	**90.5%**	89.9%	79.3%
Recall	**94.3%**	91.0%	90.9%	78.7%
F1 Score	0.82	**0.90**	0.89	0.78

Table 4.9: Evaluation metrics for balanced dataset with augmentation

In setup 1, the performance of the models can be evaluated based on different metrics. The accuracy ranges from 75.5% for MobileNet V3 to 90.5% for MobileNet V1, indicating a decent level of performance across the models. This suggests that the models are able to make correct predictions to a reasonable extent. Precision scores, which measure the ability to correctly identify positive instances, range from 68.2% for MobileNet V3 to 91.7% for MobileNet V1, showing promising results in terms of correctly identifying positive instances. The recall scores, which indicate the model's ability to capture positive instances effectively, range from 82.9% for ResNet-50 to 93.6% for MobileNet V3, implying good sensitivity in detecting positive instances. The F1 scores, which provide a balance between precision and recall, range from 0.79 to 0.89, suggesting a reasonable trade-off between these two metrics.

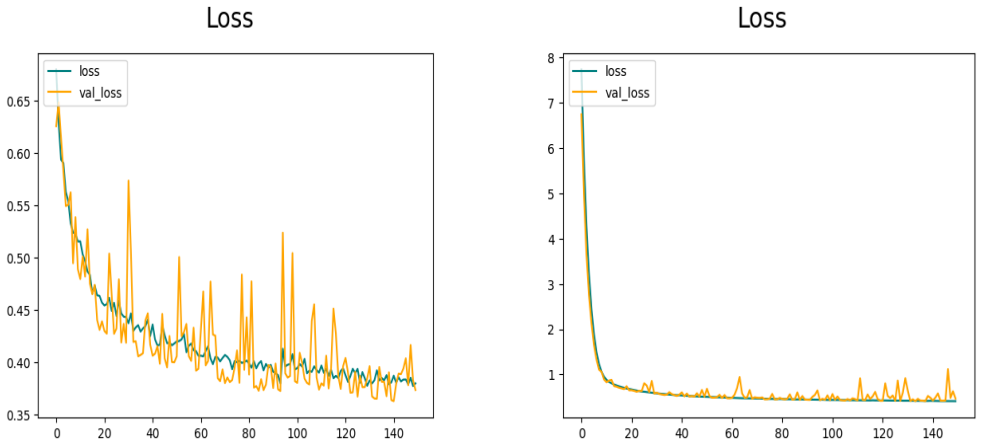
In the case of setup 2, the performance across the models exhibits variations. The accuracy ranges from 79.6% for MobileNet V3 to 90.9% for MobileNet V1, indicating a significant difference in performance among the models. This suggests that some models are more accurate in their predictions than others. Precision scores range from 79.3% for MobileNet V3 to 90.5% for MobileNet V1, indicating varying degrees of correct positive identification. This implies that some models have a higher precision in correctly identifying positive instances than others. Recall scores range from 78.7% for MobileNet V3 to 94.3% for ResNet-50, indicating different abilities to capture positive instances effectively. This suggests that certain models perform better in correctly capturing positive instances. The F1 scores, ranging from 0.78 to 0.90, reflect a varying balance between precision and recall,

indicating that different models achieve different trade-offs between these two metrics.

To understand the performance of each model under the specified setup, the results of ResNet-50, MobileNet V1, MobileNet V2 and MobileNet V3 are presented in the following sections.

ResNet-50

This section presents the results of ResNet-50 using setup 1 and setup 2 for an almost balanced dataset using data augmentation. The loss and accuracy curves provide an overview during the training process while the confusion matrix and AUC score serve as an insight into the model's evaluation performance.



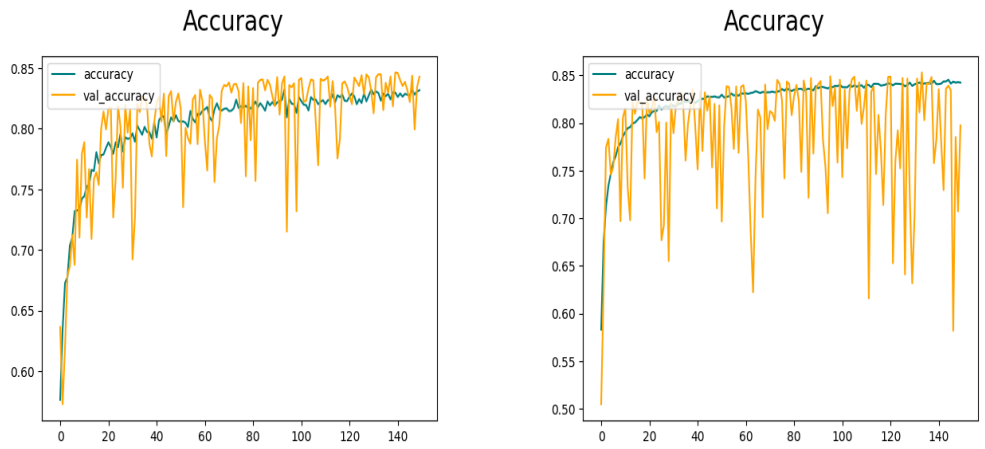
(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.33: ResNet-50 loss curves for balanced data with augmentation

The loss curves for ResNet-50 are presented in Fig. 4.33. In setup 1, the presence of spikes in the validation loss curve indicates that the model's performance on unseen data, specifically the validation set, is not consistently improving. This suggests a potential issue of overfitting, where the model struggles to generalize well and becomes overly sensitive to variations or noise in the validation set. The validation loss curve exhibits an initial value of approximately 0.65 and shows a descending trend with irregular spikes, ultimately reaching just below 0.4. As for setup 2, the loss curve begins at a relatively high value of around 8 but steadily decreases over time, dropping below 1. This pattern suggests that the model quickly learns and adjusts its predictions during the initial training epochs. The decreasing trend of the loss curve indicates that the model continues to improve its performance as training progresses. Although small spikes can still be observed on the validation curve, the overall smoothness of the loss curve implies that the model's performance remains relatively stable throughout the training process.

It can be inferred that in setup 1, the model may be overfitting, leading to

sub-optimal generalization and inconsistent performance on unseen data. Conversely, in setup 2, the model demonstrates improved learning capabilities, with a more stable and steadily decreasing loss curve, indicating better performance and adaptability. These findings highlight the importance of monitoring loss curves to assess the model's ability to generalize and optimize its predictions effectively.

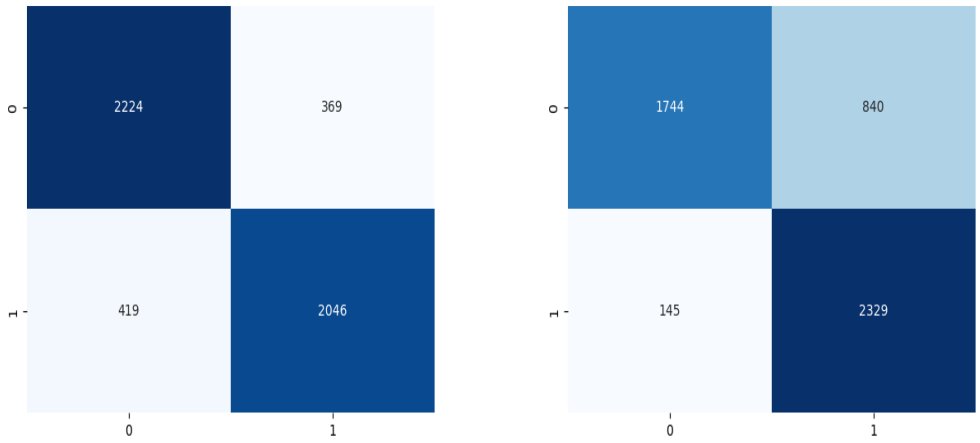


(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.34: ResNet-50 accuracy curves for balanced data with augmentation

The accuracy curves for ResNet-50 are presented in Fig. 4.34. In setup 1, the train accuracy curve shows a fluctuating upward trend, starting around 0.55 and gradually reaching a range of 0.80 to 0.85 over the course of 150 epochs. However, the validation accuracy curve displays a highly irregular pattern throughout, indicating that the model faces challenges in effectively learning new features. As a result, the model's performance exhibits sudden drops and increases, indicating difficulty in converging to a stable and optimal solution. On the other hand, in setup 2, even with a reduced learning rate, the validation curve begins at approximately 0.5 and experiences heavy fluctuations, spanning a wide range between 0.5 and 0.85. The training curve, on the other hand, exhibits a gradual increase from around 0.55 to a range between 0.8 and 0.85. While the training curve suggests improvements in performance, the instability observed in the validation curve implies that the model's accuracy varies significantly and fails to maintain consistency.

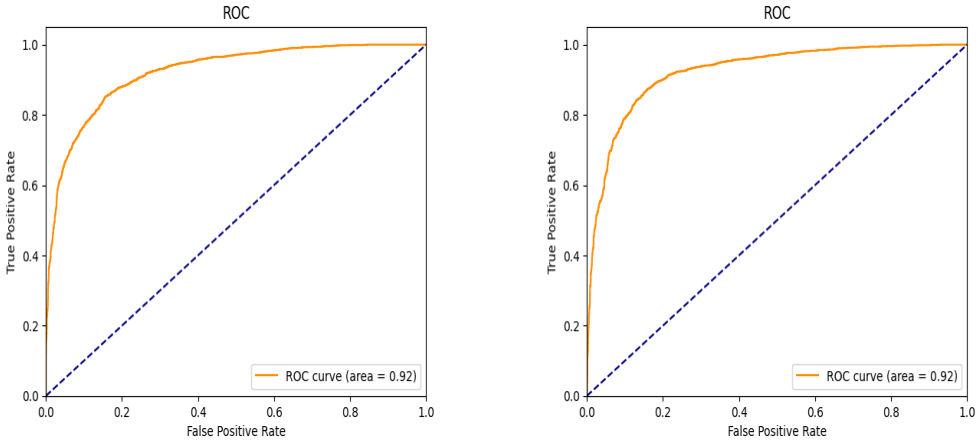
It can be inferred that in setup 1, the model encounters difficulties in effectively learning new features, as indicated by the irregular validation accuracy curve. This inconsistency in performance hinders the model from converging to a stable and optimal solution. In setup 2, despite the reduced learning rate, the model's performance remains unstable, as evidenced by the fluctuating validation curve. This lack of consistency suggests that the model struggles to maintain a reliable level of accuracy.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.35: ResNet-50 confusion matrix for balanced data with augmentation

The confusion matrix for ResNet-50 is presented in Fig. 4.35. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 85.7% and the incorrect classification is 14.3%. **Class 1 (trout)**: The correct classification is 83% and the incorrect classification is 17%.
For setup 2: **Class 0 (mature)**: The correct classification is 67.5% and the incorrect classification is 32.5%. **Class 1 (trout)**: The correct classification is 94.1% and the incorrect classification is 5.9%.



(a) Setup 1 **(b)** Setup 2

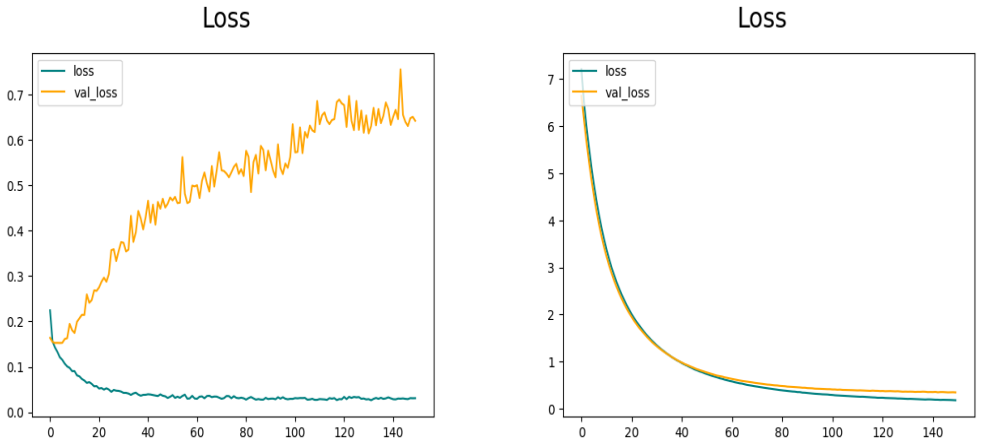
Figure 4.36: ResNet-50 Area under the Curve

The AUC score for ResNet-50 is presented in Fig. 4.36 Assessing the provided scores, it is observed that for setup 1 the AUC score is 0.92 and for setup 2, the

AUC score is 0.92. This indicates a high value of effective classification of the data.

MobileNet V1

In this section, the results of MobileNet V1 are presented using a balanced dataset with augmentation and with setup 1 and setup 2. The loss and accuracy curves provide information on the model performance while training and the confusion matrix and AUC score provide information on the evaluation of the model.



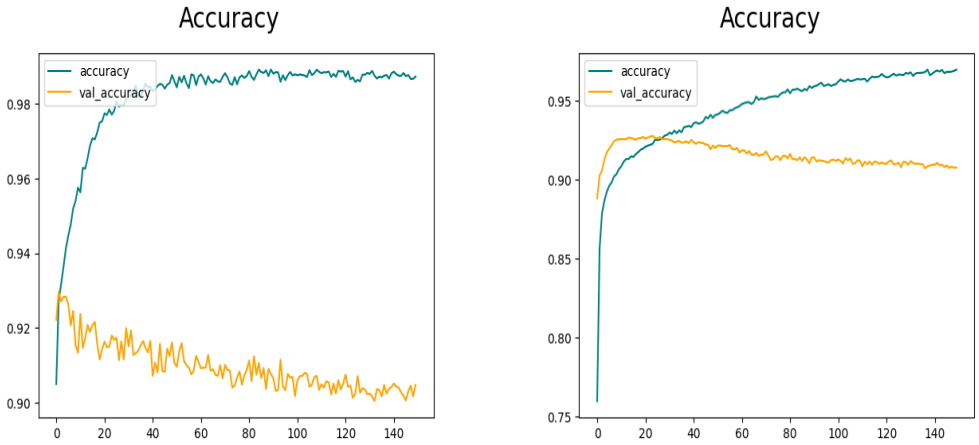
(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.37: MobileNet V1 loss curves for balanced data with augmentation

The loss curves for MobileNet V1 are presented in Fig. 4.37. In setup 1, The validation loss curve exhibits irregular fluctuations, starting in the range between 0.1 and 0.2, but constantly increasing and reaching up to 0.7. These spikes indicate that the model's performance on unseen data is not consistently improving, which suggests a potential issue of overfitting. Overfitting occurs when the model becomes overly sensitive to variations or noise in the validation set, failing to effectively capture the underlying patterns in the data. As for setup 2, the loss curve initially starts at a relatively high value of around 7. However, it consistently decreases over time and eventually stabilizes below 1. This decreasing trend signifies that the model quickly learns and adjusts its predictions during the initial training epochs. The overall smoothness of the loss curve indicates a relatively stable performance during training, suggesting that the model's performance continues to improve.

It can be inferred that in setup 1, the model faces challenges in generalizing to unseen data, as indicated by the increasing and fluctuating validation loss curve. This behavior suggests that the model may be overfitting, emphasizing the need for appropriate regularization techniques to improve its performance. In contrast, in setup 2, the model demonstrates a more favorable trend, with consistent improvement and stability in its loss curve, indicating better generalization and

overall performance.

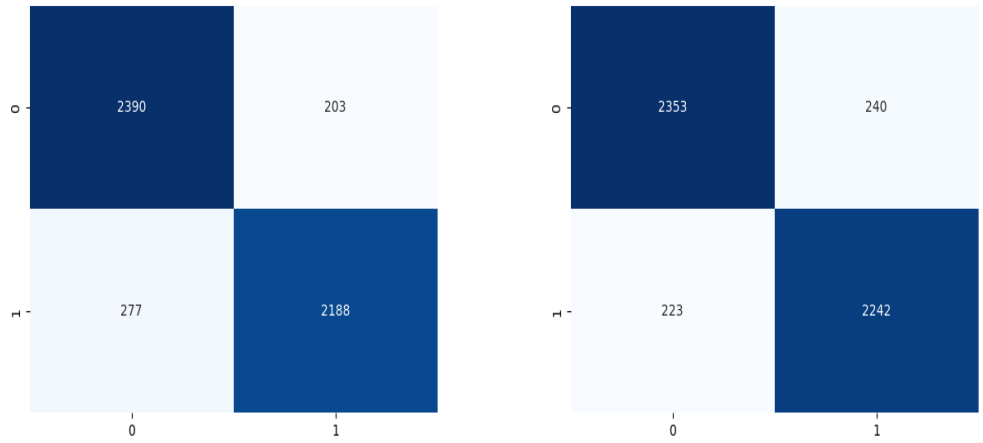


(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.38: MobileNet V1 accuracy curves for balanced data with augmentation

The accuracy curves for MobileNet V1 are presented in Fig. 4.38. In setup 1, The train accuracy curve demonstrates a relatively steady upward trend, starting at around 0.90 and reaching a range of approximately 0.98 over the course of 150 epochs. However, the validation accuracy curve exhibits irregular patterns throughout, indicating that the model struggles to efficiently learn new features. This lack of consistency in the validation curve results in sudden drops and increases in the model's performance, suggesting challenges in converging to a stable and optimal solution. The case of setup 2, which incorporates a reduced learning rate, demonstrates a more favorable performance. The validation curve remains relatively stable, fluctuating within the range of 0.90 to 0.95. This indicates that the model is effectively fitting to the unseen data and has the potential to provide more reliable results.

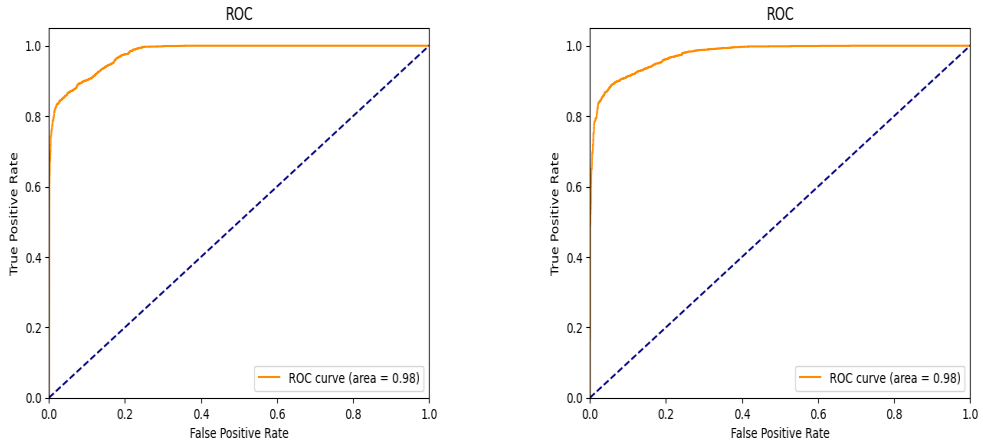
The presence of sudden drops and increases in setup 1 suggests that the model struggles to maintain a consistent and optimal performance. This highlights the need for further improvements, such as implementing regularization techniques or adjusting the model architecture, to enhance its capacity to learn new features effectively. On the other hand, in setup 2, the reduced learning rate contributes to a more stable validation curve, indicating improved generalization capabilities. This suggests that the model can better adapt to unseen data, potentially leading to more reliable and consistent predictions.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.39: MobileNet V1 confusion matrix for balanced data with augmentation

The confusion matrix for MobileNet V1 is presented in Fig. 4.39. Based on the provided confusion matrices for setup 1 and setup 2, the correct and incorrect classification percentages for each class can be analyzed.
For setup 1: **Class 0 (mature)**: The correct classification is 92.2% and the incorrect classification is 7.8% **Class 1 (trout)**: The correct classification is 88.8% and the incorrect classification is 11.2%
For setup 2: **Class 0 (mature)**: The correct classification is 95.5% and the incorrect classification is 4.5% **Class 1 (trout)**: The correct classification is 90.9% and the incorrect classification is 9.1%



(a) Setup 1 **(b)** Setup 2

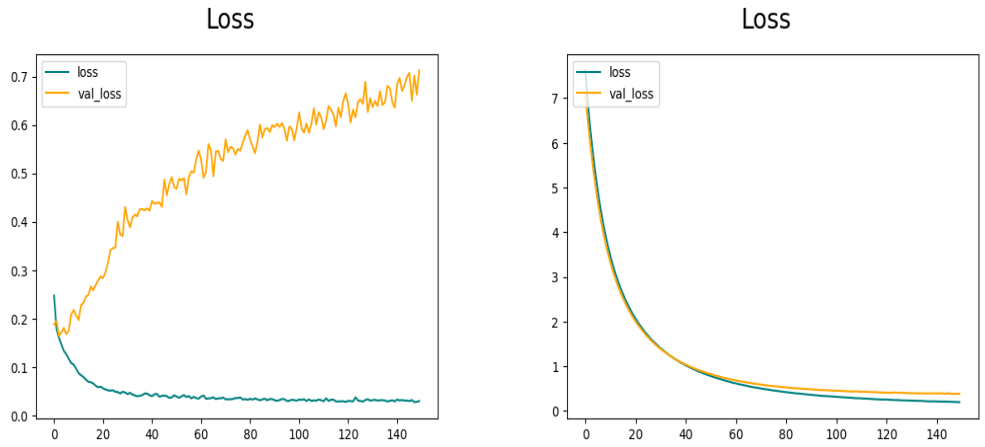
Figure 4.40: MobileNet V1 AUC score

The AUC score for MobileNet V1 is presented in Fig. 4.40. Assessing the provided

scores, it is observed that the AUC score for setup 1 is 0.98 and for setup 2, the AUC score is 0.98. It can be inferred that the models have demonstrated strong performance in terms of their ability to correctly classify instances and assign higher probabilities to positive instances compared to negative ones. This suggests that the models have learned meaningful patterns and features from the data, leading to accurate predictions.

MobileNet V2

In this section, the results for MobileNet V2 are presented with setup 1 and setup 2 when using the augmented balanced dataset. The loss and accuracy curves will be helpful to analyze the performance of the model while training while for evaluation purposes, the confusion matrix and AUC score will be helpful for understanding.



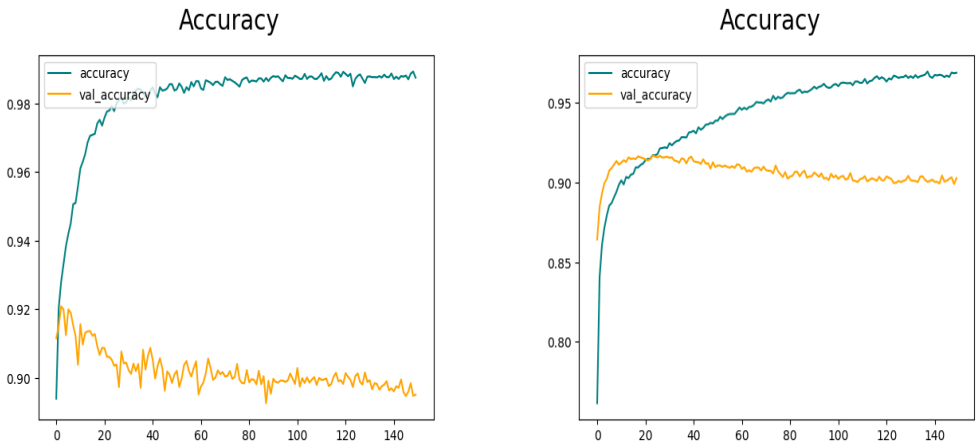
(a) Loss curve setup 1 **(b)** Loss curve setup 2

Figure 4.41: MobileNet V2 loss curves for balanced data with augmentation

The loss curves for MobileNet V2 are presented in Fig. 4.41. In setup 1, the validation loss curve exhibits irregular spikes, indicating that the model's performance on unseen data is not improving consistently. The initial value of the validation loss is in the range of 0.1 to 0.2, but it progressively increases and reaches up to 0.7 during the training process, accompanied by fluctuations. This behavior suggests a potential issue of overfitting, where the model struggles to capture the underlying patterns in the data and becomes overly sensitive to noise or variations present in the validation set. As for setup 2, the loss curve for MobileNet V2 starts with a relatively high value of approximately 7, but it consistently decreases over time and eventually stabilizes at a value below 1. This decreasing trend indicates that the model quickly learns and adjusts its predictions during the initial training epochs. The continued improvement of the loss curve suggests that the model's performance is progressively getting better. Moreover, the overall smoothness of

the loss curve indicates that the model's performance remains relatively stable throughout the training process.

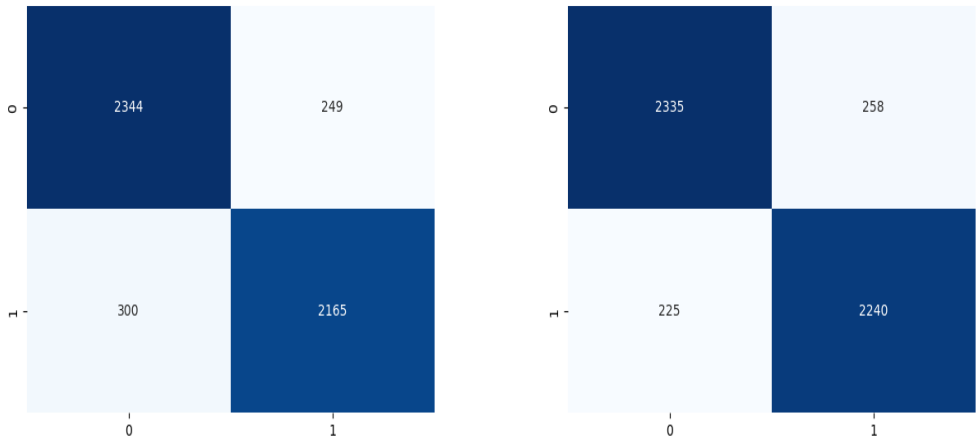
It can be inferred that in setup 1, the MobileNet V2 model may be struggling to generalize well to unseen data, as indicated by the increasing and fluctuating validation loss. In setup 2, the model demonstrates better performance with consistent improvement and stability, suggesting its ability to learn effectively and generalize well to new data.



(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.42: MobileNet V2 accuracy curves for balanced data with augmentation

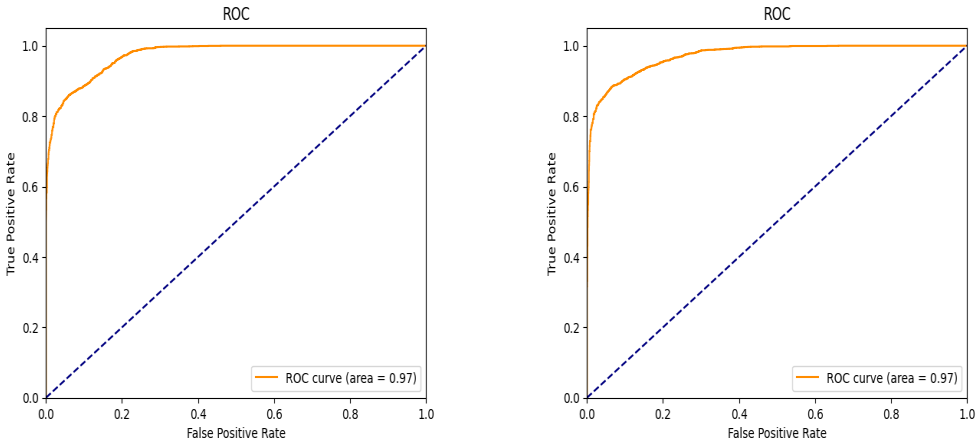
The accuracy curves for MobileNet V2 are presented in Fig. 4.42. In setup 1, the train accuracy curve exhibits a relatively smooth upward trend, starting at around 0.90 and gradually increasing to a range of approximately 0.98 over the course of 150 epochs. However, the validation accuracy curve displays some irregular patterns, indicating that the model encounters challenges in effectively learning new features. As a result, the model's performance shows sudden fluctuations, indicating difficulties in converging to a stable and optimal solution. The validation accuracy remains within the range of 0.92 to 0.90, suggesting a relatively consistent but slightly lower performance compared to the training accuracy. In the case of setup 2, with a reduced learning rate, the validation accuracy curve consistently stays above the range of 0.90 and demonstrates greater stability with fewer fluctuations. Additionally, the gap between the train and validation accuracy curves appears to be smaller and more consistent. This indicates that the model is able to effectively generalize to unseen data and is likely to provide more reliable results. These findings highlight the importance of fine-tuning the learning rate to enhance the model's performance and its ability to generalize effectively.



(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.43: MobileNet V2 confusion matrix for balanced data with augmentation

The confusion matrix for MobileNet V2 is presented in Fig. 4.43. Based on the provided confusion matrices for setup 1 and setup 2, the accuracy for each class can be analyzed. For setup 1: **Class 0 (mature)**: The correct classification is 90.4% and the incorrect classification is 9.6% **Class 1 (trout)**: The correct classification is 87.8% and the incorrect classification is 12.2%
For setup 2: **Class 0 (mature)**: The correct classification is 90.0% and the incorrect classification is 10% **Class 1 (trout)**: The correct classification is 90.9% and the incorrect classification is 9.1%



(a) AUC score setup 1 **(b)** AUC score setup 2

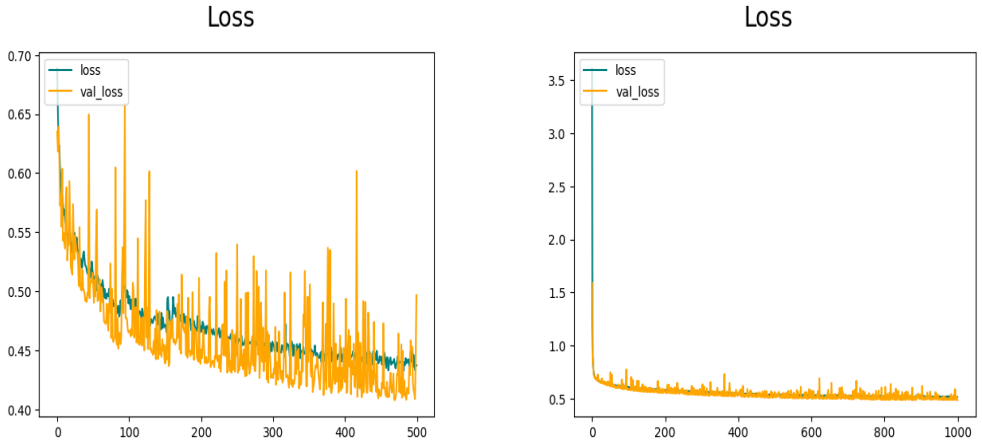
Figure 4.44: MobileNet V2 AUC score for balanced data with augmentation

The AUC score for MobileNet V2 is presented in Fig. 4.44. Assessing the provided scores, it can be observed that the AUC score for setup 1 is 0.97 and for setup 2,

the AUC score is 0.97. The similarity in AUC scores suggests that the models in both setups were effective in distinguishing between the positive and negative instances in the dataset, providing reliable predictions.

MobileNet V3

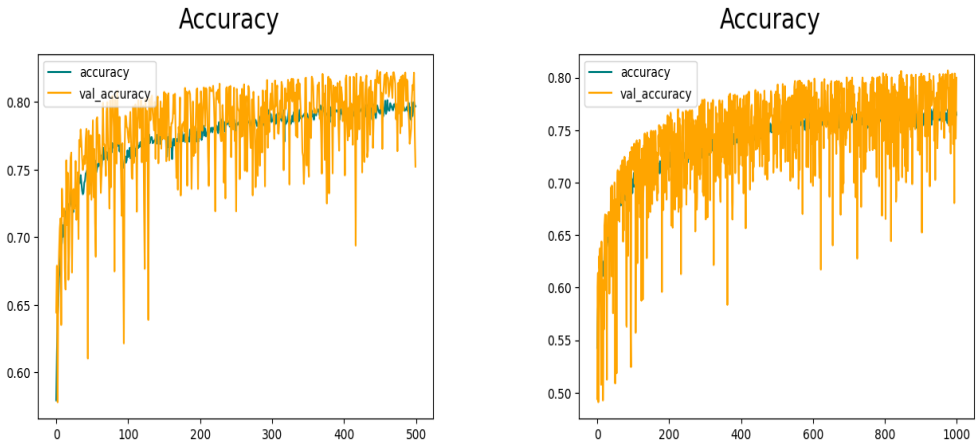
In this section, the results of MobileNet V3 using a balanced dataset with augmentation and setup 1 and setup 2 are presented. The loss and accuracy curves indicate the performance of the model while training. The confusion matrix and the AUC score are used to understand the evaluation of the model.



(a) Loss curve setup 1 **(b)** Loss curve setup 2

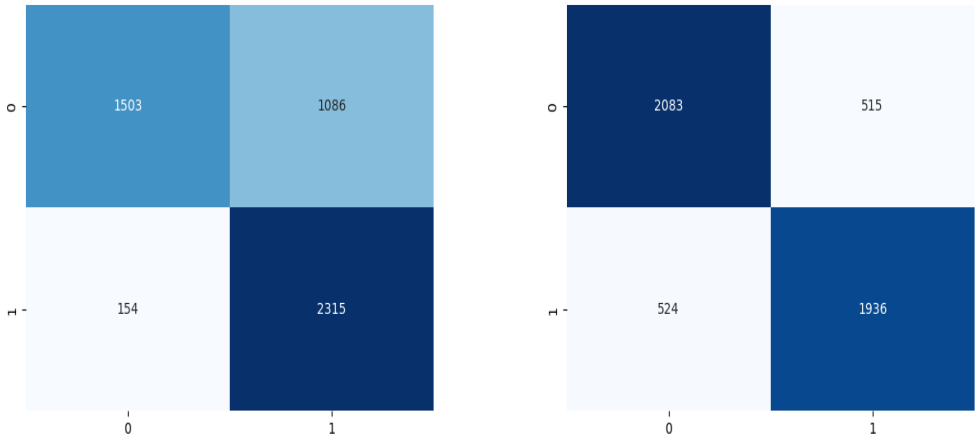
Figure 4.45: MobileNet V3 loss curves for balanced data with augmentation

The loss curves for MobileNet V3 are presented in Fig. 4.45. In setup 1, the presence of spikes in the validation loss curve indicates that the model struggles to improve its performance on unseen data. From the start, the validation loss hovers between 0.65 and 0.7, with fluctuating spikes throughout the 500-epoch training period. These observations suggest that the model may have encountered difficulties in effectively capturing the underlying patterns in the data, potentially due to a high learning rate. Consequently, the model becomes overly sensitive to variations and noise present in the validation set. As for setup 2, the validation loss curve begins at a higher value of approximately 1.5 and steadily decreases to less than 1, reaching around 0.5 during the 1000-epoch training process. Despite the presence of some spikes, the overall decreasing and relatively stable trend of the loss curve indicates that the model is learning to adjust its predictions during the initial training epochs. This suggests that the model is capable of gradually improving its performance over time. In summary, the loss curves for MobileNet V3 highlight the differences in performance between setup 1 and setup 2.



(a) Accuracy curve setup 1 **(b)** Accuracy curve setup 2

Figure 4.46: MobileNet V3 accuracy curves for balanced data with augmentation



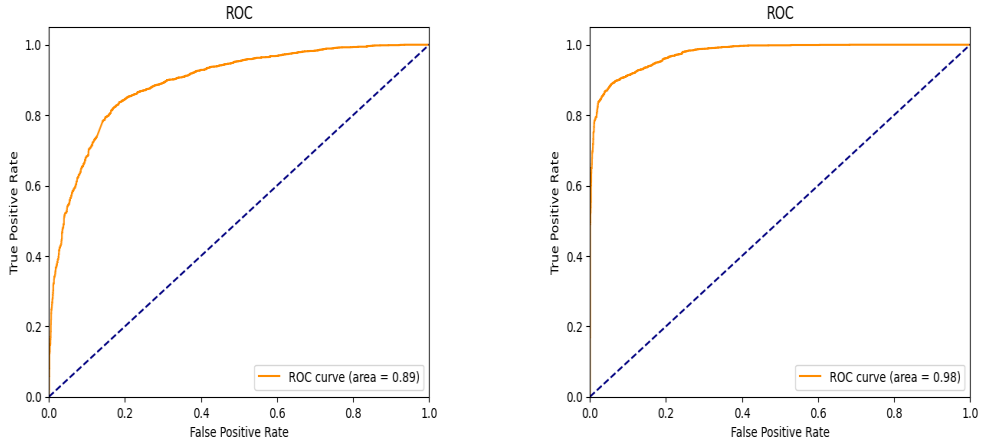
(a) Confusion matrix setup 1 **(b)** Confusion matrix setup 2

Figure 4.47: MobileNet V3 confusion matrix for balanced data with augmentation

The accuracy curves for MobileNet V3 are presented in Fig. 4.46. In setup 1, the train accuracy curve shows a relatively stable upward trend, starting around 0.90 and reaching a range of approximately 0.75 to 0.8 over 500 epochs. However, the validation accuracy curve exhibits a highly irregular pattern with numerous low and high spikes. This erratic behavior indicates that the model struggles to effectively learn new features, resulting in inconsistent performance. The sudden drops and increases in the validation accuracy further suggest that the model faces challenges in converging to a stable and optimal solution. In the case of setup 2, even with a reduced learning rate, the validation accuracy curve continues to exhibit instability and fails to generalize well to the data. It fluctuates between the

range of 0.5 and 0.75, indicating that the model's performance is not satisfactory for the given classification task with the current set of hyperparameters. Therefore, it can be inferred that the accuracy curves for MobileNet V3 highlight the limitations of the model in both setup 1 and setup 2.

The confusion matrix for MobileNet V3 is presented in Fig. 4.47. Based on the provided confusion matrices for setup 1 and setup 2, the accuracy for each class can be analyzed. For setup 1: **Class 0 (mature)**: The correct classification is 58.0% and the incorrect classification is 42.0% **Class 1 (trout)**: The correct classification is 93.7% and the incorrect classification is 6.3%
For setup 2: **Class 0 (mature)**: The correct classification is 80.1% and the incorrect classification is 19.9% **Class 1 (trout)**: The correct classification is 78.7% and the incorrect classification is 21.3%



(a) Setup 1 **(b)** Setup 2

Figure 4.48: MobileNet V3 AUC score

The AUC score for MobileNet V3 is presented in Fig. 4.48. Assessing the provided scores, it is observed that for setup 1, the AUC score is 0.89 and for setup 2, the AUC score is 0.98. This difference in AUC scores suggests that setup 2 may have a more effective model configuration or hyperparameter settings, resulting in improved performance compared to setup 1. The higher AUC score in setup 2 implies a better ability of the model to correctly classify instances, potentially leading to more reliable and accurate predictions.

4.3 Additional Experiments

In this section, an analysis of some additional results obtained through a specific model fine-tuning approach is presented. Specifically, the effects of unfreezing a subset of layers, adding additional layers, and compiling the resulting model was explored. By incorporating this technique, the aim was to investigate the impact

of further fine-tuning on the performance and capabilities of the models. The approach focuses on unfreezing a subset of **10 layers** within the pre-trained models and adding the additional setup of layers as discussed in section 3.6.2. By unfreezing these layers, the weights of these layers are also included in the training process. It was expected that this approach of fine-tuning will allow the model to learn more task-specific features and adapt to the intricacies of the target domain. It is worth noting that the choice of 10 layers as the number to unfreeze can be further refined and optimized based on empirical analysis and experimentation. Different tasks, datasets, and model architectures may require varying numbers of layers to be unfrozen for optimal performance. Therefore, the choice of 10 layers serves as a starting point, and it can be adjusted and fine-tuned based on the specific requirements and characteristics of each individual scenario. [1]

The data that has been used for this setup is balanced with augmentation on the minority class (mature class). A total of **50882** images were involved in this setup. The distribution is presented in Table 4.10.

Label	**Class**	**Images**	**Distribution**
Mature	0	25612	≈50%
Trout	1	25270	≈50%

Table 4.10: Dataset for additional experiments

The following tables 4.11 and 4.12 provide detailed results for the setup. These results offer valuable insights into the performance and characteristics of each model. This data is useful to analyze the impact of different hyper-parameters on the models' performance and training as well as understand if there is any impact by having a balanced dataset.

[1]The results from unfreezing 10 layers using the models with setup 1 (3.6.1) is available in Appendix 6.1. Also, the results from unfreezing 20 layers and using the models with setup 1 (3.6.1) is available in Appendix 6.1. This data can be used to further analyze the performance based on this approach of fine-tuning the models for future works.

Model Architecture Setup 2 (3.6.2)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Batch Size	64	64	64	64
Total Data	796	796	796	796
Train Size	557 (70%)	557 (70%)	557 (70%)	557 (70%)
Validation Size	159 (20%)	159 (20%)	159 (20%)	159 (20%)
Test Size	80 (10%)	80 (10%)	80 (10%)	80 (10%)
Total Params	24,639,874	3,756,738	2,916,930	4,885,922
Trainable Params	5,516,802	2,115,074	1,390,402	1,889,922
Learning Rate	0.000001	0.00001	0.00001	0.00001
Epochs	150	150	150	150
Training Time (mins)	274	138	167	133

Table 4.11: Model configuration for additional experiments

In the conducted experiment, several factors were considered for the models. Firstly, all models were trained with a batch size of 64. The dataset sizes were consistent across the models, with 557 batches in the training set, 159 batches in the validation set, and 80 batches in the test set.

The total number of parameters differed among the models, with ResNet-50 having the highest number of parameters at 24,639,874, while MobileNet V2 had the lowest with 2,916,930 parameters. Similarly, the number of trainable parameters varied, with ResNet-50 having the highest at 5,516,802 and MobileNet V2 having the lowest at 1,390,402.

Regarding the learning rate and epochs, all models except ResNet-50 were trained using a learning rate of 0.00001. However, ResNet-50 was trained with a lower learning rate of 0.000001 and for 150 epochs, which was an experimental decision.

Finally, the training time for the models ranged from 133 minutes to 274 minutes. ResNet-50 required the longest training time, while MobileNet V3 had the shortest. These considerations are essential in understanding the experimental setup and interpreting the subsequent results.

Evaluation Metrics				
Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Accuracy	87.5%	**90.6%**	90.2%	84.5%
Precision	84.9%	**89.2%**	88.5%	79.5%
Recall	90.5%	91.9%	91.7%	**92.0%**
F1 Score	0.87	**0.90**	0.89	0.85

Table 4.12: Evaluation metrics for additional experiments

The evaluation of the models' performance yielded the following results. The accuracy of the models ranged from 84.5% for MobileNet V3 to 90.6% for MobileNet V1, indicating a strong overall performance across the models. These accuracy scores signify the proportion of correctly classified instances in relation to the total number of instances.

Precision scores, which measure the accuracy of positive predictions, varied from 79.5% for MobileNet V3 to 89.2% for MobileNet V1. These scores indicate the models' ability to correctly identify positive instances while minimizing false positives.

The recall scores, which measure the ability to capture positive instances, ranged from 90.5% for ResNet-50 to 90% for MobileNet V3. Higher recall scores indicate a higher sensitivity in identifying positive instances and a lower rate of false negatives.

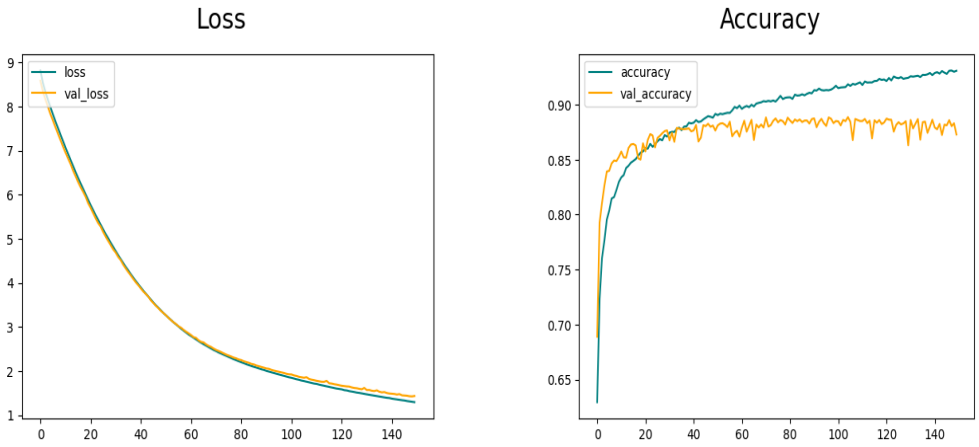
The F1 scores, which represent the harmonic mean of precision and recall, ranged from 0.85 for MobileNet V3 to 0.90 for MobileNet V1. These scores provide a measure of the balance between precision and recall, with higher F1 scores indicating a better balance between these two metrics.

Overall, the evaluation metrics suggest that the models achieved good performance in terms of accuracy, precision, recall, and F1 score. MobileNet V1 appears to have consistently demonstrated strong performance across all metrics.

In order to understand the performance of each model better, the following sections present the results from ResNet-50, MobileNet V1, MobileNet V2, and MobileNet V3.

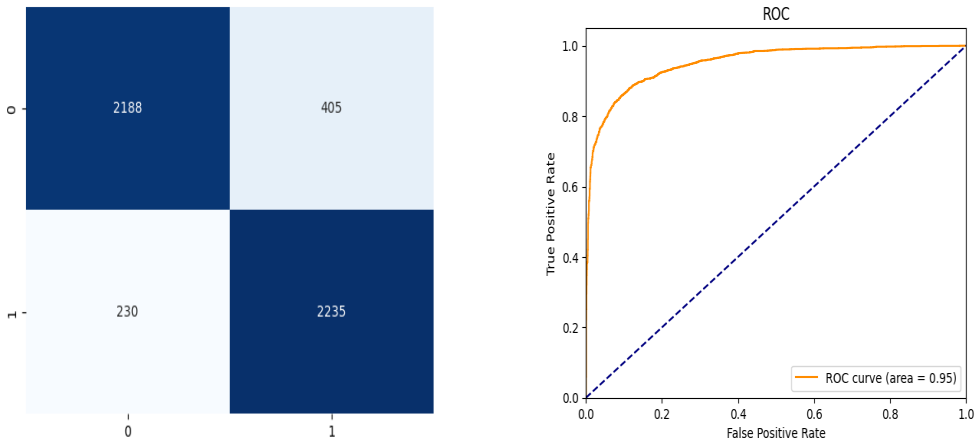
4.3.1 ResNet-50

In this section, the results from ResNet-50 are presented when additional 10 layers of the model are added to the training using a balanced dataset with setup 2. The loss and accuracy curve along with the confusion matrix and AUC score are presented.



(a) Loss curve **(b)** Accuracy curve

Figure 4.49: ResNet-50 loss and accuracy curves for additional experiments



(a) Confusion matrix **(b)** AUC score

Figure 4.50: ResNet-50 Confusion Matrix and AUC score for additional experiments

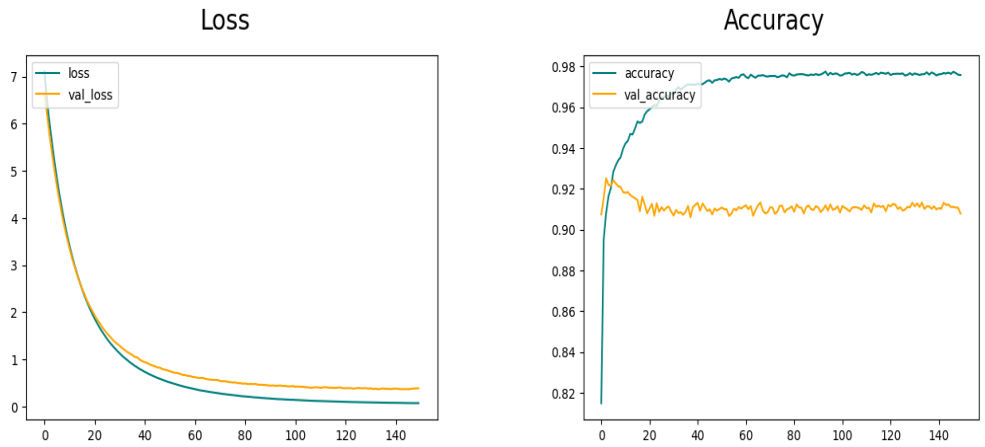
The loss and accuracy curves for ResNet-50 after further fine-tuning are presented in Fig. 4.49. The loss curve exhibits an initial high value of approximately 8, but gradually decreases over time, reaching a range between 1 and 2. This consistent descent indicates that the model is effectively adapting to new features without overfitting to unseen data. The smoothness of the loss curve suggests that the model's performance remains stable throughout the training process. The validation accuracy curve demonstrates an upward trend, starting from around 0.7 and steadily progressing to a range between 0.85 and 0.90. This indicates that the model is successfully learning and generalizing to unseen data by capturing new features. Despite the presence of minor spikes, the accuracy range remains relat-

ively consistent during the 150 epochs. These findings indicate that the fine-tuned ResNet-50 model is capable of capturing and leveraging new features, leading to enhanced performance in classification tasks.

The confusion matrix for ResNet-50 after further fine-tuning is presented in Fig. 4.50. Based on the provided confusion matrices, the correct and incorrect classification percentages for each class can be analyzed.
Class 0 (mature): The correct classification is 84.4% and the incorrect classification is 15.6% **Class 1 (trout)**: The correct classification is 90.6% and the incorrect classification is 9.4%
The AUC score, which measures the performance of a classifier, and the AUC score for ResNet-50 after further fine-tuning is 0.95 which is a high value for the curve suggesting that the model is highly effective in separating the two classes.

4.3.2 MobileNet V1

In this section, results for MobileNet V1 are presented when additional 10 layers are used in the training process while using setup 2 and a balanced dataset with augmentation. To understand the training process, the loss and accuracy curves are presented. To understand the evaluation, the confusion matrix, and AUC score are presented.

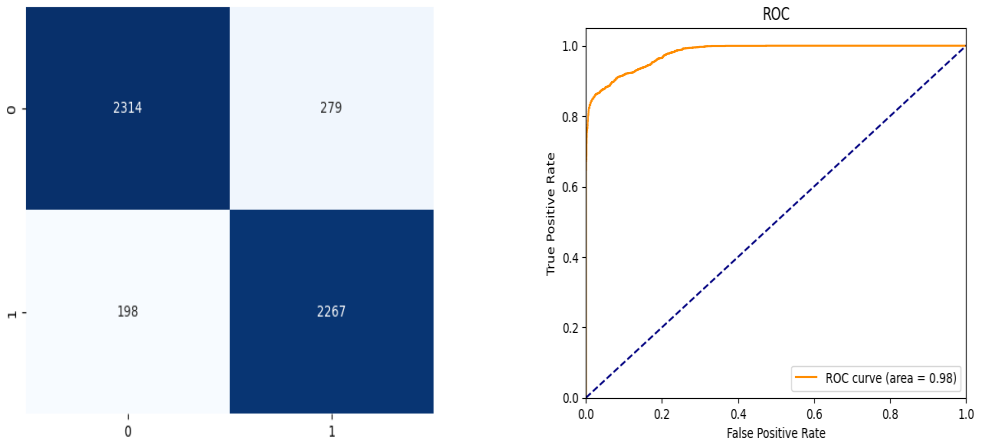


(a) Loss curve **(b)** Accuracy curve

Figure 4.51: MobileNet V1 loss and accuracy curves for additional experiments

The loss and accuracy curves for MobileNet V1 after further fine-tuning are presented in the Fig. 4.51. The loss curve initially exhibits a relatively high value of around 7 but gradually decreases to a range between 0 and 1. This consistent descent indicates that the model is effectively adapting to new features without overfitting to the unseen data. The overall smoothness of the loss curve suggests that the model's performance remains relatively stable during the training process. Regarding the validation accuracy curve, it can be observed that it starts at

a value around 0.9 and steadily increases to a range between 0.9 and 0.92. This indicates that the model is successfully learning new features and can generalize well to unseen data. Despite the presence of minor spikes, the accuracy range remains relatively consistent throughout the 150 epochs. These findings imply that the fine-tuned MobileNet V1 model can successfully leverage new information, leading to enhanced performance in classification tasks.



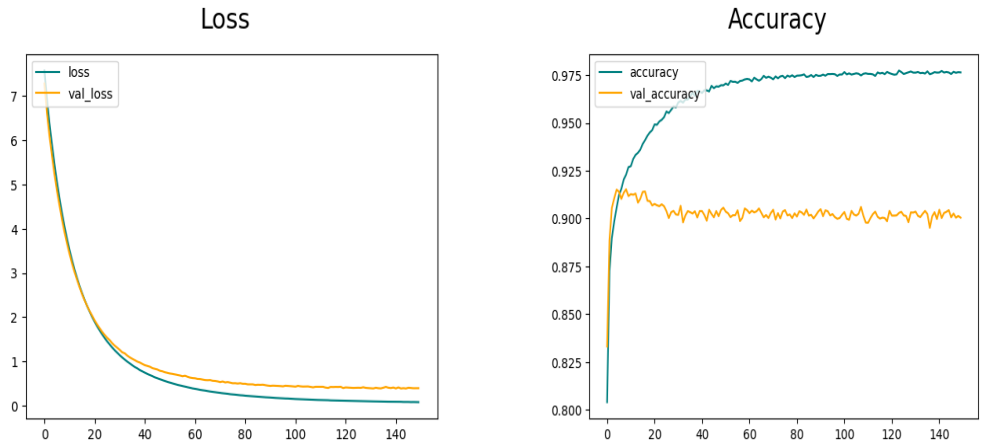
(a) Confusion matrix **(b)** AUC score

Figure 4.52: MobileNet V1 Confusion Matrix and AUC score for additional experiments

The confusion matrix and AUC score for MobileNet V1 after further fine-tuning are presented in Fig. 4.50. Based on the provided confusion matrices, the accuracy for each class can be analyzed. **Class 0 (mature)**: The correct classification is 89.2% and the incorrect classification is 10.8% **Class 1 (trout)**: The correct classification is 91.9% and the incorrect classification is 8.1%
The AUC score MobileNet V1 after further fine-tuning is 0.98 which is a high value for the curve suggesting that the model has been able to classify the data with high efficiency and precision.

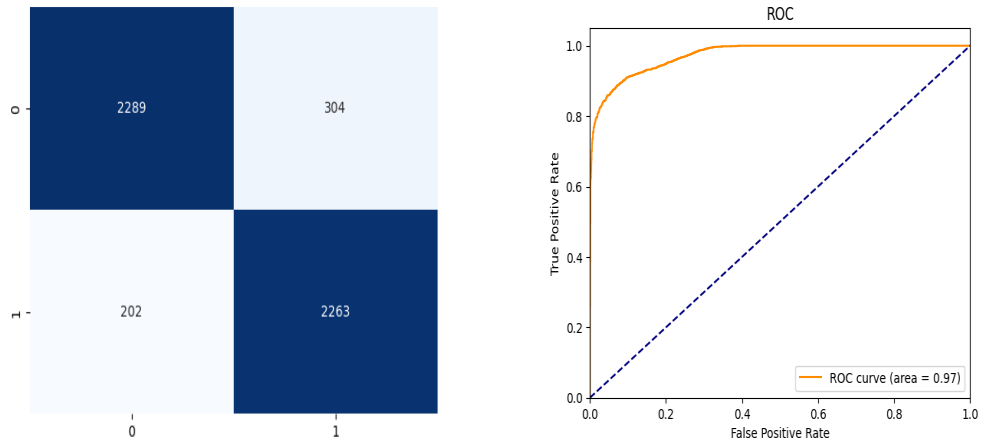
4.3.3 MobileNet V2

In this section, the results for MobileNet V2 are presented which will be useful to analyze the performance of the model better. The loss and accuracy curves share information about the training process and the evaluation metrics such as the confusion matrix and AUC score share information on the overall accuracy of the model.



(a) Loss curve **(b)** Accuracy curve

Figure 4.53: MobileNet V2 loss and accuracy curves for additional experiments



(a) Confusion matrix **(b)** AUC score

Figure 4.54: MobileNet V2 confusion matrix and AUC score for additional experiments

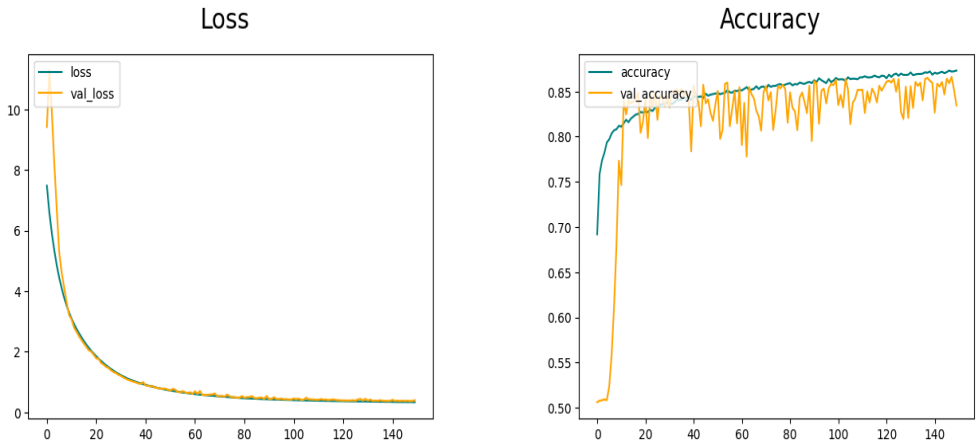
The loss and accuracy curves for MobileNet V2 after further fine-tuning are presented in Fig. 4.53. The initial loss value of the curve is relatively high, around 7, but gradually decreases within the range of 0 to 1. This consistent descent indicates that the model is effectively adapting to new features while avoiding overfitting the unseen data. The overall smoothness of the loss curve suggests that the model's performance remains relatively stable during the training process. Regarding the validation accuracy curve, it can be observed that it starts at approximately 0.82 and steadily increases to a range between 0.90 and 0.92. This suggests that the model is successfully learning new features and demonstrates the ability to generalize well to unseen data. Although minor spikes are present,

the accuracy range remains consistent over the course of 150 epochs. The overall stability of the loss and accuracy curves throughout the training process further supports the model's robustness. These findings suggest that MobileNet V2, after fine-tuning, can be a valuable choice for tasks requiring feature adaptation and generalization in classification scenarios.

The confusion matrix for MobileNet V2 after further fine-tuning is presented in Fig. 4.54. Based on the provided confusion matrix, the accuracy of the model for each class can be analyzed. **Class 0 (mature)**: The correct classification is 88.2% and the incorrect classification is 11.8% **Class 1 (trout)**: The correct classification is 91.8% and the incorrect classification is 8.2%
The AUC score for MobileNet V2 after further fine-tuning is 0.97 which suggests that the model has high efficiency in the classification task.

4.3.4 MobileNet V3

In this section, the results for MobileNet V3 are presented when additional 10 layers are included in the training on the balanced dataset with setup 2. The loss and accuracy curves are useful in understanding the training process. The confusion matrix and the AUC score are useful in understanding the performance of the model.

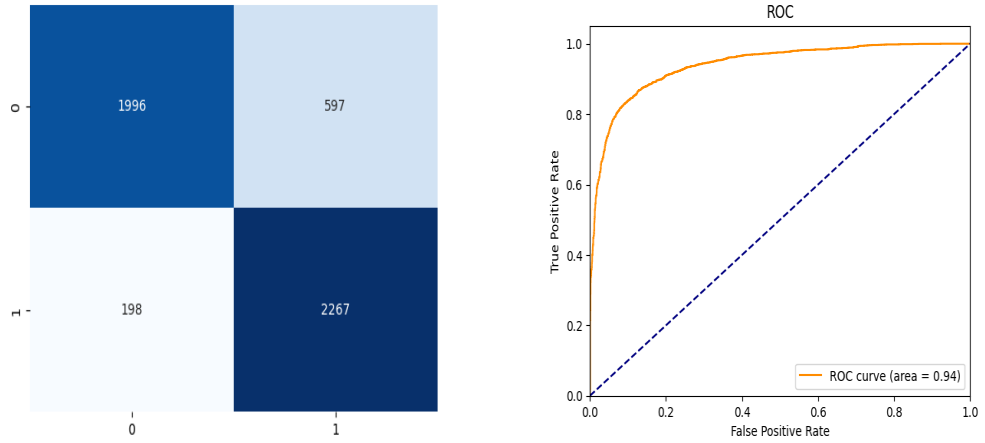


(a) Loss curve **(b)** Accuracy curve

Figure 4.55: MobileNet V3 loss and accuracy for additional experiments

The loss and accuracy curves for MobileNet V3 after further fine-tuning are presented in the Fig. 4.55. Initially, the loss curve starts at a relatively high value of around 10 and gradually descends within the range of 0 to 2. This consistent descent indicates that the model is effectively adapting to new features while avoiding overfitting the unseen data. The overall smoothness of the loss curve suggests that the model's performance remains relatively stable during the training process. Examining the validation accuracy curve, it can be observed that it

starts at approximately 0.50 and gradually increases to a range between 0.75 and 0.85. However, fluctuations can be observed as the training progresses, indicating sudden losses and gains in the model's efficiency. This suggests that the model is encountering challenges in generalizing to unseen data, which may be attributed to the usage of a higher learning rate during training. Adjusting the learning rate or employing other optimization techniques could potentially enhance the model's generalization performance. Overall, further investigation and fine-tuning are recommended to improve the model's ability to generalize unseen data effectively.



(a) Confusion matrix **(b)** AUC score

Figure 4.56: MobileNet V3 confusion matrix and AUC score for additional experiments

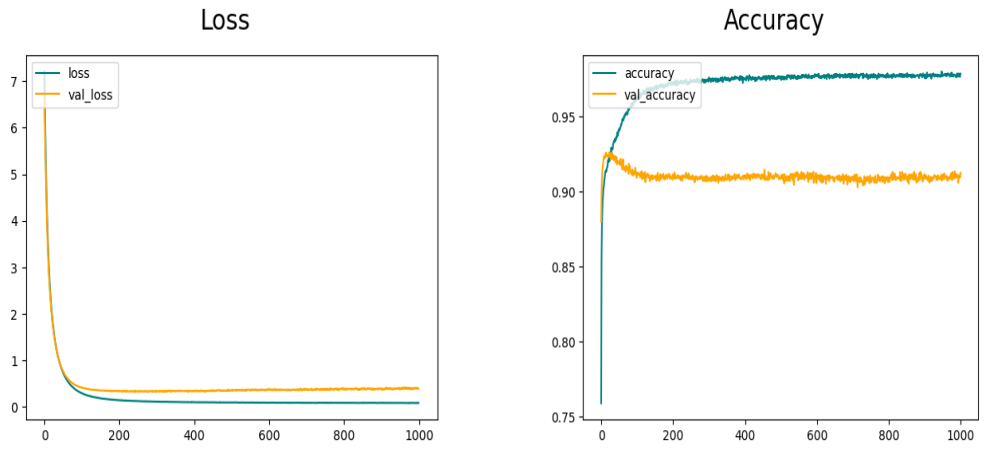
The confusion matrix and AUC score for MobileNet V3 after further fine-tuning are presented in Fig. 4.56. Based on the provided confusion matrices, the accuracy of the model for each class can be analyzed. **Class 0 (mature)**: The correct classification is 76.9% and the incorrect classification is 23.1% **Class 1 (trout)**: The correct classification is 91.9% and the incorrect classification is 8.1%
The AUC score for MobileNet V2 after further fine-tuning is 0.94 suggesting that the model has a relatively high efficiency and precision in the classification of the two classes.

4.4 Further Evaluation of MobileNet V1

As per results in table 4.9, it is observed that MobileNet V1 has one of the best metrics in comparison to other models when it is combined with setup 2 (3.6.2) and using the balanced dataset with augmentation described in table 4.7. To evaluate the stability of the model further, the model was run with the same configuration as mentioned in table 4.8 except for the number of epochs. For this setup, `epochs = 1000` was used which resulted in a training time of 799 minutes. The results are presented in Table 4.13.

MobileNet V1 Evaluation Metrics for 1000 epochs			
Accuracy	**Precision**	**Recall**	**F1 Score**
90.67%	90.4%	90.48%	0.90

Table 4.13: MobileNet V1 further evaluation

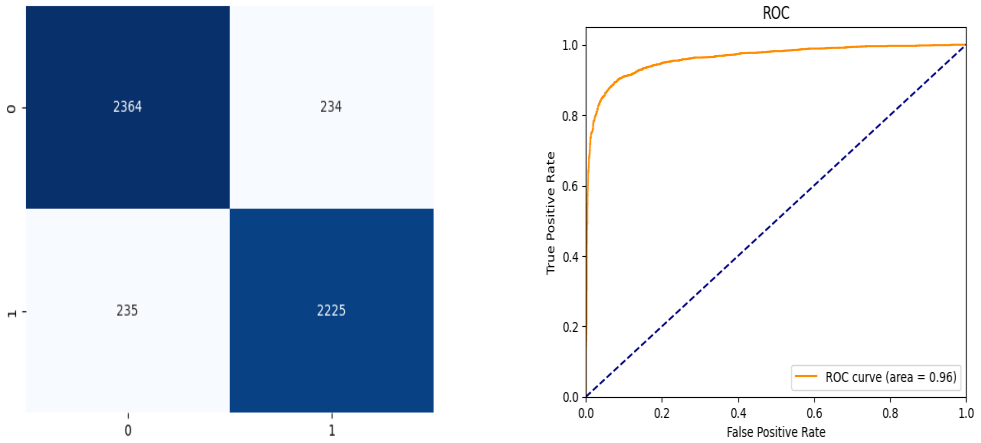


(a) Loss curve **(b)** Accuracy curve

Figure 4.57: MobileNet V1 loss and accuracy curves for further evaluation

The loss and accuracy curves for MobileNet V1 after further training it to 1000 epochs are presented in Fig. 4.57. Initially, the loss curve starts at a relatively high value of around 7 and gradually descends within the range of 0 to 1. This consistent descent indicates that the model is effectively learning and adapting to new features without overfitting the unseen data. The overall smoothness of the loss curve suggests that the model is performance remains relatively stable throughout the training process. Examining the validation accuracy curve, it can be observed that it starts at approximately 0.90 and gradually increases to a range between 0.90 and 0.95. While some minor fluctuations are observed as the training progresses, the overall stability of the curve indicates that the model is able to generalize well to unseen data. This suggests that the model is consistently performing at a high level and can be relied upon to provide reliable and consistent results.

These findings indicate that the model's performance can be further improved with extended training. However, it is important to monitor for any signs of overfitting, as the validation accuracy curve may plateau or decline after a certain point. Overall, the results obtained from the extended training of MobileNet V1 are promising and encourage further exploration and optimization.



(a) Confusion matrix **(b)** AUC score

Figure 4.58: MobileNet V1 Confusion Matrix and AUC score for further evaluation

The confusion matrix and AUC score for MobileNet V1 after further training it to 1000 epochs are presented in Fig. 4.58. Based on the provided confusion matrices, the accuracy for each class can be evaluated. **Class 0 (mature)**: The correct classification is 90.9% and the incorrect classification is 9.1% **Class 1 (trout)**: The correct classification is 90.4% and the incorrect classification is 9.6%
The AUC score for MobileNet V2 after further fine-tuning is 0.96 which is quite a high value that suggests that the model is highly precise and efficient in the given classification task.

Another evaluation was performed on MobileNet V1 where a different kind of setup was used. In this setup, only the train and validation images were processed through the augmentation method to balance the minority class (mature) with the majority class (trout). The test dataset consisted of only original images which did not go through augmentation. The purpose of this setup was to simulate a real-world scenario where the data is naturally unbalanced. The original dataset which was available and is presented in Table 4.1, had ≈16% of mature class images. To replicate this scenario, a test dataset was prepared which had ≈15% of the mature class images. Table 4.14 presents the data distribution for this setup.

	Train Dataset	**Validation Dataset**	**Test Dataset**
Mature (Class 0)	17695	5632	460
Trout (Class 1)	17695	5013	2548
Total Images	35390	10645	3008

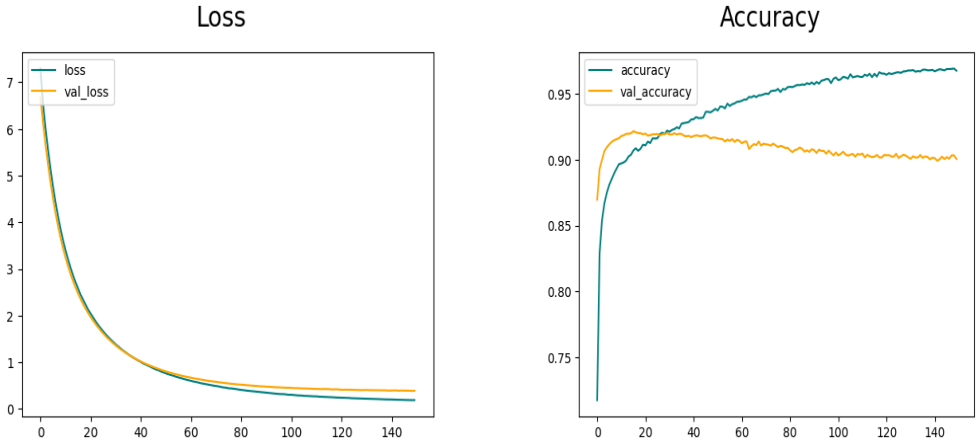
Table 4.14: Dataset for MobileNet V1 additional experiments

This model was compiled with setup 2 (3.6.2) with a learning rate of 0.00001. It took around 76 minutes for the training to complete and the results are presented in Table 4.15.

Results			
Accuracy	**Precision**	**Recall**	**F1 Score**
84.5%	91.5%	89.9%	0.90

Table 4.15: MobileNet V1 further evaluation using new dataset

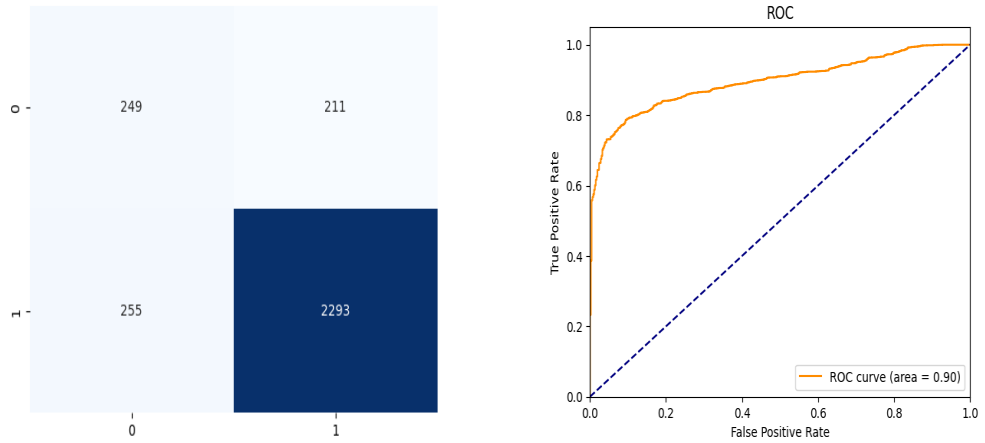
To facilitate understanding the training process of the model in this setup, the loss is accuracy curves are presented in Fig. 4.59. To analyze the classification accuracy of the model overall and for each class, the confusion matrix and the AUC score are presented in Fig. 4.60.



(a) Loss curve **(b)** Accuracy curve

Figure 4.59: MobileNet V1 loss and accuracy curve for the additional experiment using new dataset

The loss curve starts from a value of around 7 points and descends smoothly to a range between 0 and 1. The stability and smoothness of the curve suggest that the model was able to quickly adapt to new data and generalize. This also suggests that the model is not struggling while training on the dataset. The validation accuracy curve which starts between the range 0.85 and 0.90, suggests that the model is able to learn from the features of the data and is not struggling to generalize as well. The validation accuracy can be seen stabilizing without a lot of fluctuations suggesting that the model is able to hold stability and can provide reliable results.



(a) Confusion matrix **(b)** AUC score

Figure 4.60: MobileNet V1 confusion matrix and AUC score for additional experiment using new dataset

As for the confusion matrix, it is observed that the correct classification for **Class 0 (mature)** is **54.1%** and the correct classification for **Class 1 (trout)** is **89.9%**. This suggests that the model has better accuracy in classifying the trout class and further adjustments can be performed on the model to improve the accuracy of the mature class. The high value of the AUC score which is 0.90 suggests that the model has high overall precision and effectiveness in classifying the given dataset under the given configurations setup.

In this chapter, the results from many different experiment setups for different models have been presented and analyzed. In the next chapter, a deeper analysis will be conducted to understand the potential reasons behind the observed variations in performance and explore potential strategies for improving the models. The impact of factors such as hyperparameter selection, model architecture, and training techniques will be thoroughly examined to gain insights into their influence on the results. Furthermore, the implications of these findings in real-world applications will be discussed, and suggestions for future research will be put forward with the aim of enhancing the performance and efficiency of deep learning models.

Chapter 5

Discussion

This chapter provides an in-depth analysis of the experimental findings and their implications in evaluating various models and setups which was presented in the previous chapter. The performance of the models will be examined using different evaluation metrics, including accuracy, precision, recall, F1 score, confusion matrix, and AUC score. The relationships between these metrics will be explored to identify any alignment or divergence. Possible reasons for the observed variations in performance will be discussed, along with an investigation of the factors that may have influenced the models' predictive capabilities. Strategies for further improvement will be explored, and the implications of the findings in real-world applications will be examined. The insights derived from this discussion will serve as valuable guidance for future research, aimed at enhancing the performance and efficiency of the models considered in this study.

5.1 Comprehensive Analysis

Unbalanced Dataset

In general, the accuracy values align with the performance observed in the confusion matrix. For instance, MobileNet V3 with setup 2 achieves the highest accuracy among the models, and the confusion matrix reveals that it has a perfect correct percentage (100%) for Class 1 (trout). This indicates a strong ability to correctly classify trout instances. However, the confusion matrix also reveals a very low correct percentage (0.22%) for Class 0 (mature), suggesting a poor performance in identifying mature instances.

Comparing setup 1 and setup 2, it can be observed that setup 2 consistently outperforms setup 1 in terms of correct percentages for both classes across all models. This aligns with the higher accuracy values seen in setup 2.

However, it's worth noting that the confusion matrix provides more detailed insights into the performance of each class. For example, although setup 2 generally performs better overall, MobileNet V2 with setup 1 achieves a higher correct

percentage (approximately 59.55%) for Class 0 (mature) compared to all other models.

The Precision-Recall analysis and F1 scores provide complementary information about the models' performance in terms of precision, recall, and their balance. ResNet-50 consistently shows higher precision values, while MobileNet V3 excels in recall and achieves high F1 scores. The choice between setup 1 and setup 2 depends on the specific metric of interest and the desired trade-off between precision and recall.

Balanced Dataset Without Augmentation

MobileNet V1 achieves the highest overall accuracy in setup 2 (85.9%) and the second-best accuracy in setup 1 (82.5%), with very close performance to the best performer, MobileNet V2 in setup 1. This suggests that MobileNet V1 has better overall performance in correctly predicting the class labels. This aligns with the observations from the confusion matrix, where MobileNet V1 consistently demonstrates higher correct percentages for both classes, particularly in setup 2. Therefore, the accuracy data coincides with the trends observed in the confusion matrix, reinforcing the reliability of the accuracy metric as an indicator of the models' performance.

Furthermore, when comparing precision, recall, and F1 scores, it is observed that different models excel in different metrics and setups. MobileNet V2 achieves the highest precision score in setup 1, while MobileNet V1 achieves the highest precision score in setup 2. This suggests that both MobileNet V1 and V2 exhibit good precision in their respective setups. In terms of recall, ResNet-50 performs best in setup 1, whereas MobileNet V3 excels in setup 2, achieving a recall score of 99.4%. This indicates that ResNet-50 is more effective in avoiding false positives in setup 1, while MobileNet V3 demonstrates superior performance in setup 2. MobileNet V1 consistently achieves the highest F1 scores in both setups, indicating a good balance between precision and recall and strong overall performance.

Considering the AUC scores, MobileNet V1 consistently outperforms the other models, demonstrating higher AUC scores of 0.90 in both setups. This indicates that MobileNet V1 has a better ability to discriminate between the positive and negative classes compared to the other models. ResNet-50 also performs well but has slightly lower AUC scores, while MobileNet V3 lags behind, suggesting weaker discrimination power.

In summary, the analysis of precision, recall, F1 scores, and the AUC score provides additional insights into the models' performance. MobileNet V1 consistently stands out with its strong overall performance, exhibiting high accuracy, precision, F1 scores, and AUC scores. The observations from the confusion matrix further support the performance trends observed in the evaluation metrics. The choice of the best model and setup depends on the specific metric of interest and the trade-off between precision and recall.

Balanced Dataset With Augmentation

Regarding accuracy, MobileNet V1 consistently outperforms the other models in both setup 1 and setup 2, achieving accuracy scores of 90.5% and 90.9%, respectively. This indicates that MobileNet V1 has a higher overall correct classification rate compared to the other models. In line with the performance observed in the accuracy metric, the analysis of the confusion matrix also reveals that MobileNet V1 consistently outperforms the other models in terms of correct classification rates and relatively low wrong classification rates. This further reinforces MobileNet V1's strong ability to accurately classify the samples.

Examining precision, MobileNet V1 maintains its superiority, consistently achieving the highest precision scores among the models. With precision scores of 91.7% in setup 1 and 90.5% in setup 2, MobileNet V1 demonstrates a low rate of false positives, making it reliable in predicting positive samples accurately.When it comes to recall, MobileNet V3 stands out in setup 1, achieving a recall score of 93.6%. This suggests that MobileNet V3 has a higher ability to correctly identify positive samples compared to the other models in this specific setup. Analyzing the F1 scores, MobileNet V1 once again proves its strength, consistently achieving the highest F1 scores among the models. With F1 scores of 0.89 in both setup 1 and setup 2, MobileNet V1 demonstrates a good balance between precision and recall, indicating its robust performance in classification tasks.

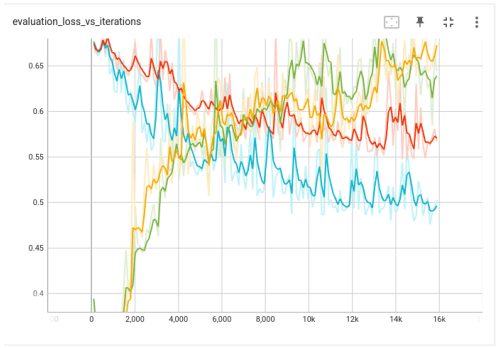
Considering the AUC scores, both MobileNet V1 and MobileNet V2 perform well, achieving scores of 0.98 and 0.97, respectively. These high scores indicate their strong performance in distinguishing between positive and negative samples. The choice between the two models may depend on additional factors such as computational efficiency or specific requirements of the application.

Overall, the evaluation of accuracy, precision, recall, F1 scores, confusion matrix, and AUC scores consistently highlights MobileNet V1 as the top-performing model. It demonstrates superior accuracy, precision, and F1 scores, while also achieving high correct classification rates and low wrong classification rates in the confusion matrix analysis. These findings indicate that MobileNet V1 is a reliable and effective model for the given classification task.

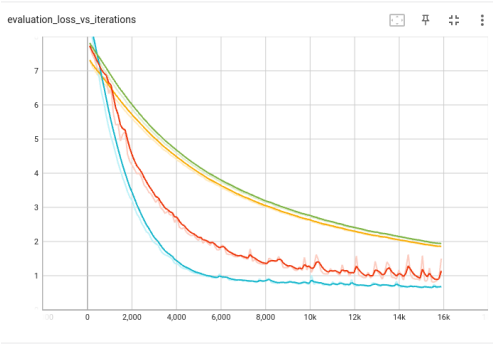
Model Architecture Setup 1 (3.6.1) vs Model Architecture Setup 2 (3.6.2)

Based on the provided information and results, it is evident that setup 2, which includes additional components such as L2 regularization, Batch Normalization, and Dropout, performs better compared to setup 1. The loss curves associated with setup 2 demonstrate greater stability and smoothness throughout the training process, which indicates improved convergence and generalization capabilities. In contrast, the loss curves of setup 1 appear more irregular and oscillating, suggesting potential challenges in convergence and a higher risk of overfitting. The inclusion of L2 regularization in setup 2 helps prevent overfitting by adding a penalty term to the loss function. This regularization technique promotes smaller

weights, thereby reducing the model's reliance on specific features and enhancing its ability to generalize to unseen data. Additionally, the presence of BatchNormalization in setup 2 contributes to a more stable learning process by normalizing activations between layers. Furthermore, the incorporation of a Dropout layer in setup 2 plays a role in enhancing the model's performance. By randomly deactivating a fraction of neurons during training, Dropout encourages the model to learn robust and independent representations. The smoother and more stable loss curves observed in setup 2 can be attributed to the combined effects of L2 regularization, BatchNormalization, and Dropout. These components work in synergy to regularize the model, stabilize the learning process, and improve generalization. As a result, setup 2, with its additional components, demonstrates better performance compared to setup 1, highlighting the significance of incorporating these techniques to enhance model convergence and overall effectiveness. The Fig. 5.1 presents the difference between loss curves for all four models when trained on the balanced dataset without augmentation and it is observed that the problem of overfitting is reduced and the loss curves seem to converge as the training progresses.



(a) Setup 1



(b) Setup 2

Figure 5.1: Loss comparison of setup 1 and setup 2 on the unbalanced dataset without augmentation

Unbalanced vs Balanced Data and Effect of Data Augmentation

Model	Dataset	Class 0 (mature)	Class 1 (trout)	Overall Accuracy	F1 Score
ResNet-50	Unbalanced	59.55%	92.53%	87%	0.92
	Balanced (no augmentation)	52.99%	89.39%	72.9%	0.77
	Balanced (augmentation)	**67.5%**	**94.1%**	**80.5%**	**0.82**
MobileNet V1	Unbalanced	48.8%	91.4%	84.9%	0.90
	Balanced (no augmentation)	88.38%	84.4%	85.9%	0.86
	Balanced (augmentation)	**95.5%**	**90.9%**	**90.9%**	**0.90**
MobileNet V2	Unbalanced	46.81%	91.47%	84.9%	0.90
	Balanced (no augmentation)	85.8%	82.9%	83.7%	0.84
	Balanced (augmentation)	**90.0%**	**90.9%**	**90.6%**	**0.89**
MobileNet V3	Unbalanced	0.22%	100%	84.8%	0.91
	Balanced (no augmentation)	2.36%	99.6%	53.8%	0.69
	Balanced (augmentation)	**80.1%**	**78.7%**	**79.6%**	**0.78**

Table 5.1: Performance of models on unbalanced and balanced datasets

As the discussion continues further with results for setup 2, Table 5.1 provides the important metrics that will provide the basis for discussion between having an unbalanced dataset or a balanced dataset. The results clearly demonstrate the importance of a balanced dataset for achieving better model performance. In all models evaluated (ResNet-50, MobileNet V1, MobileNet V2, and MobileNet V3), the performance on the balanced datasets, both with and without augmentation, outperformed the results obtained from the unbalanced data. This observation is evident in the higher accuracy percentages and F1 scores for the balanced datasets.

Furthermore, when comparing the balanced datasets, the results show that using data augmentation techniques to increase the data size further enhances

model performance. The balanced datasets with augmented data consistently yielded higher accuracy percentages and F1 scores compared to the balanced datasets without augmentation. This suggests that increasing the dataset size through augmentation allows the models to learn more robust features and generalize better, leading to improved performance in both classes.

These findings suggest that a balanced dataset is crucial for better model performance, and augmenting the data can further enhance the model's ability to accurately classify both classes. Therefore, it is recommended to balance the dataset and leverage data augmentation techniques to achieve optimal results in similar classification tasks.

MobileNet V1 as the potential best model for this classification task

Based on the overall comparison of the models and the additional metrics provided, it can be inferred that MobileNet V1 emerges as the best model for this particular classification task. Among the evaluated models, MobileNet V1 demonstrates superior performance across various aspects. Firstly, in terms of accuracy, MobileNet V1 consistently achieves high correct percentages for both classes, outperforming ResNet-50 and MobileNet V2 in the balanced datasets with data augmentation. Its overall accuracy of 90.9% is the highest among the models, indicating its ability to effectively classify the target classes.

Furthermore, MobileNet V1 exhibits a favorable F1 score of 0.90, indicating a good balance between precision and recall. This reflects its capability to accurately capture both classes in the classification task. The F1 score of MobileNet V1 is higher compared to the other models, including ResNet-50 and MobileNet V2.

Additionally, MobileNet V1 proves to be a lightweight model, requiring significantly fewer trainable parameters (526,850) compared to ResNet-50 (1,051,138), MobileNet V2 (657,922), and MobileNet V3 (656,898). This reduced number of parameters indicates that MobileNet V1 is more computationally efficient and consumes less memory, allowing for faster inference times.

Moreover, considering the training time for 150 epochs on the balanced dataset with augmentation, MobileNet V1 stands out as an efficient model, taking only 162 minutes. Although MobileNet V2 requires a slightly shorter training time of 144 minutes, MobileNet V1's training time is still comparable and relatively close. In comparison, ResNet-50 requires a longer training time of 240 minutes. This demonstrates that MobileNet V1 achieves high accuracy within a reasonable training time, making it an effective choice for this classification task.

Therefore, based on the available data and metrics, MobileNet V1 emerges as the preferred model for this classification task. Its lightweight architecture, faster training time, higher accuracy (90.9%), and favorable F1 score make it a strong choice. Utilizing MobileNet V1 can provide efficient and accurate classification results for similar tasks while optimizing computational resources.

Further evaluation of MobileNet V1

MobileNet V1 was further trained for 1000 epochs in order to thoroughly evaluate its stability and potential for improvement over an extended training period, the results presented in the section 4.4. Firstly, the overall accuracy of 90.67% indicates that the model consistently performs well in correctly classifying both classes. This accuracy demonstrates the model's ability to generalize to unseen data, suggesting that it has learned robust features and is capable of making consistent predictions.

Precision, recall, and F1 score, all hovering around 90%, further reinforce the model's effectiveness in achieving a balance between correctly identifying positive samples and avoiding false positives. Analyzing the individual class correctness, we find that both Class 0 and Class 1 have high correct percentages of 90.9% and 90.4%, respectively. This balanced performance on both classes indicates that the model can accurately classify mature and trout instances without favoring one class over the other.

The stability in the loss curve and accuracy curve suggests that the model has the ability to generalize to unseen data and consistently make accurate predictions while being consistent in its performance during training. Moreover, considering the case where only the training and validation data were augmented and the test data was kept unbalanced to simulate a practical implementation, MobileNet V1 achieved an overall accuracy of 84.5% along with a stable and descending loss curve.

Based on these results, it can be observed that the MobileNet V1 model, when trained for longer epochs, demonstrates stability in its performance and exhibits the potential for further investigation and improvement. Further research and experimentation can be conducted to explore additional optimization techniques and fine-tuning strategies to enhance the model's performance even further.

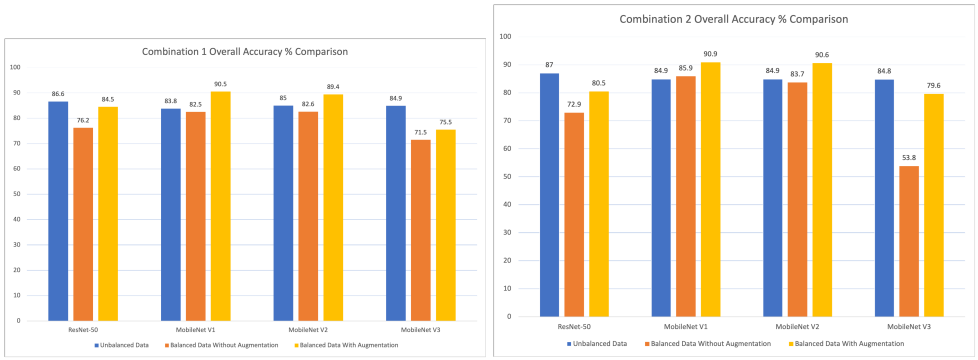
Additional Experiments by further fine-tuning of models

The additional experiments which involved fine-tuning the models ResNet-50, MobileNet V1, MobileNet V2 and MobileNet V3 by unfreezing 10 layers and training them, have been presented in section 4.3. Based on the evaluation metrics in table 4.12, it is evident that MobileNet V1 consistently outperforms the other models. It achieved the highest accuracy, precision, and F1 score among all models, indicating its superior performance in correctly classifying samples and achieving a balance between precision and recall. MobileNet V1 demonstrated an overall accuracy of 90.6%, precision of 89.2%, recall of 91.9%, and an F1 score of 0.90. These results highlight the effectiveness of MobileNet V1 in accurately classifying the given dataset.

Additionally, when examining the loss curves for all models, it can be observed that they exhibit stability and smoothness throughout the training process. The loss curve starts at a relatively high value between 8-10 and gradually descends,

reaching a range between 0 and 2. The consistent descent of the curve indicates that the models are effectively learning to adapt to new features and are not overfitting to the unseen data. The overall smoothness of the loss curve implies that the models' performance remains relatively stable during training, further supporting their ability to generalize well to unseen data.

These results suggest that fine-tuning the models by unfreezing a portion of the layers can lead to improved performance. The combination of a balanced dataset with augmentation, coupled with fine-tuning, has proven to be a successful strategy for achieving higher performance. MobileNet V1, with its lightweight architecture, exhibited superior performance in terms of accuracy, precision, recall, and F1 score. Further exploration and refinement of the fine-tuning process may lead to even better results, reinforcing the potential of MobileNet V1 for this classification task.



(a) setup 1 **(b)** setup 2

Figure 5.2: Comparison of overall accuracy %

5.2 Addressing the Research Questions

Are there any specific data pre-processing techniques that can be employed to mitigate the challenges posed by limited power and memory resources and unbalanced data on an edge computer, and how do these techniques impact the performance of the models ?

The use of data augmentation techniques, such as image rotation, flipping, and scaling, can be employed to increase the diversity and quantity of the available data for training. By artificially expanding the dataset, these techniques can help mitigate the challenges posed by limited data in the context of highly unbalanced classes. Furthermore, the adoption of lightweight models, such as MobileNet V1, can address the constraints of limited power and memory resources on an edge computer. MobileNet V1 achieved high accuracy and competitive performance while consuming fewer computational resources compared to ResNet-50. This suggests that model selection plays a crucial role in mitigating resource

constraints.

Therefore employing data augmentation techniques, utilizing lightweight models, and exploring fine-tuning approaches are effective strategies to mitigate the challenges posed by limited power and memory resources and highly unbalanced data on an edge computer. These techniques can positively impact the performance of the models by improving accuracy and reducing computational requirements, enabling more efficient and reliable classification in resource-constrained settings.

What is the impact of imbalanced class distribution in the dataset on the performance of models, and how does the limited availability of data affect their performance? Additionally, what potential solutions exist to address these challenges and enhance the model's output in such scenarios?

The performance of the models significantly differs when trained on a dataset with highly unbalanced classes. The results indicate that the models exhibit varying degrees of accuracy, precision, recall, and F1 score across the different imbalance scenarios. When trained on the unbalanced dataset, the models generally struggled to accurately classify the minority class. This can be attributed to the limited amount of available data for the minority class, leading to a lack of exposure and learning opportunities for the models. Consequently, the models tended to favor the majority class, resulting in lower performance metrics for the minority class. However, when the dataset was balanced, either without augmentation or by augmenting the available data, the models showed improved performance. The impact of the limited amount of available data was evident in the performance of the models. With a scarcity of data, the models faced challenges in accurately learning the underlying patterns and characteristics of the classes. Therefore, addressing the issue of limited data availability is crucial for improving the models' performance, as a larger and more diverse dataset can provide richer learning experiences and yield more accurate and robust classification results.

Can the performance of the models be further enhanced by employing specific data pre-processing techniques or model modifications that address the challenges associated with limited memory resources and naturally unbalanced data?

Applying data augmentation techniques can help generate synthetic samples and expand the dataset, potentially improving the model's ability to generalize and perform better on unseen data. Leveraging pre-trained models and fine-tuning them on the limited data available can help accelerate the training process and potentially improve the performance of the models. Additional modifications such as L2 regularization in the dense layer, a batch normalization layer for improved training stability, and a dropout layer for regularization, aim to address overfitting, improve model generalization, and potentially enhance performance. By employ-

ing these specific data preprocessing techniques and model modifications, there is a potential to further enhance the performance of the models in the given context. However, it is important to carefully evaluate and validate these techniques to ensure they are compatible with the limited power and memory resources of the edge computer.

Which of the deep learning models compatible with the edge computer having limited resources, demonstrates the most effective performance in classifying unbalanced data?

MobileNet V1 achieves the highest accuracy among the models, followed closely by MobileNet V2. These models exhibit better capability in accurately classifying highly unbalanced data, indicating their effectiveness in this classification task. The limited power and memory resources on an edge computer make MobileNet V1 and MobileNet V2 favorable choices as they strike a balance between accuracy and resource consumption. Therefore, based on the performance analysis and considering the limited power and memory resources on an edge computer, MobileNet V1 and MobileNet V2 emerge as the most effective models in classifying highly unbalanced data. These models demonstrate competitive accuracy while being efficient in terms of resource utilization, making them suitable candidates for classification tasks under the constraints of the task.

What are the trade-offs between overall accuracy and resource consumption for the different deep learning models when applied to the classification of unbalanced classes?

Models trained on balanced datasets or with data augmentation techniques tend to have improved accuracy as they have better exposure to both classes and can learn their distinguishing features more effectively. In the context of edge computing, where resources are limited, models with lower resource requirements can be advantageous. Lightweight models, such as MobileNet V1 and MobileNet V2, tend to consume less power and memory compared to heavier models like ResNet-50. Therefore, the trade-offs between model accuracy and resource consumption on an edge computer when dealing with highly unbalanced classes are evident. Models that achieve high accuracy may require more computational resources, potentially straining the limited power and memory available on edge devices. On the other hand, models with lower resource requirements may sacrifice some accuracy. Finding the right balance between accuracy and resource consumption is crucial to ensure efficient and effective classification on edge devices with limited resources.

What are the implications of the findings from these models for real-world applications where highly unbalanced data, limited power, and memory resources are common challenges, and how can these findings guide the selec-

tion and optimization of models for similar scenarios?

The performance comparison between ResNet-50, MobileNet V1, V2, and V3 provides insights into the models' effectiveness in handling highly unbalanced data on resource-constrained edge devices. Based on the results, MobileNet V1 consistently demonstrated competitive performance, making it a favorable choice for similar scenarios. The evaluation of training time and the number of trainable parameters offers insights into the computational requirements of the models. MobileNet V1, with its relatively shorter training time and lower number of parameters compared to ResNet-50, MobileNet V2, and MobileNet V3 showcases its suitability for limited power and memory resources. The experiments with balanced datasets and data augmentation highlight the benefits of addressing the issue of class imbalance. The models trained on balanced datasets generally showed improved performance, emphasizing the importance of data-balancing techniques in enhancing classification results. Additionally, model modifications like compression and transfer learning can be explored to optimize models for resource-constrained edge devices. By leveraging the insights gained from these experiments, practitioners can make informed decisions to address the challenges and improve the effectiveness and efficiency of their models in similar scenarios. Further research and experimentation can build upon these findings to explore additional techniques and optimizations tailored to specific application requirements

Chapter 6

Conclusion

This project aimed to classify the immature and mature trout while addressing the challenges of classifying highly unbalanced data with limited power and memory resources on an edge computer, specifically focusing on fish species classification. The research explored various aspects, including data preprocessing techniques, data augmentation, and the evaluation of different deep learning models. The initial problem of working with highly unbalanced data was addressed by carefully selecting and implementing a combination of data preprocessing techniques. These techniques helped to balance the dataset and mitigate the impact of class imbalance on model performance. Additionally, data augmentation was employed to increase the size and diversity of the dataset, enhancing the generalization capabilities of the models.

Multiple experiments were conducted using different deep learning models, including ResNet-50, MobileNet V1, V2, and V3, on a) an unbalanced dataset, b) a balanced dataset without augmentation, and c) a balanced dataset with augmentation. Evaluation metrics such as accuracy, precision, recall, and F1 score were utilized to assess the performance of the models. The results demonstrated that MobileNet V1 consistently outperformed the other models across various metrics, exhibiting higher accuracy, precision, and F1 score. Further analysis revealed the trade-offs between model accuracy and resource consumption, indicating that MobileNet V1 achieved superior performance in most cases while consuming fewer resources (time and memory) compared to other models. This finding highlights the expediency and robustness of MobileNet V1 for edge devices with limited computational capabilities.

The project also investigated the stability of MobileNet V1 by training it for a longer duration of 1000 epochs. The results showed that the accuracy of the models remained consistently high throughout training, demonstrating their ability to generalize well to unseen data and maintain stable performance. Furthermore, the implications of the findings from this project extend to real-world applications where highly unbalanced data, limited power, and memory resources are common challenges. The research highlights the importance of appropriate data preprocessing techniques, such as balancing the dataset and augmenting the data,

to improve model performance. The selection of an efficient deep learning model like MobileNet V1 can significantly enhance accuracy while minimizing resource consumption.

In conclusion, this project provides insights into the classification of highly unbalanced data for edge devices with limited resources. It demonstrates the effectiveness of data preprocessing techniques, the benefits of data augmentation, and the superior performance of MobileNet V1 in fish species classification tasks. The findings offer valuable and reproducible guidance for similar scenarios and contribute to the development of practical solutions for real-time monitoring and analysis of aquatic ecosystems, environmental conservation, and decision-making processes.

6.1 Future work

In light of the findings and insights gained from the present study, several potential avenues for future research and improvement emerge. These opportunities aim to further enhance the performance and applicability of the models in the context of the given classification task. Future work can focus on various aspects, including data augmentation techniques, model optimization strategies, and the exploration of alternative deep learning architectures. By delving into these future directions, researchers can further advance the performance, robustness, and interpretability of the models for similar classification tasks, opening doors to a wide range of practical applications.

- **Model Optimization**: While the models evaluated in this study have shown promising results, further optimization efforts can be undertaken to improve their performance. This can involve exploring different network architectures, fine-tuning hyperparameters, and implementing model compression techniques specifically tailored for edge devices by taking the limited computational powers into account.
- **Hybrid Approaches**: Investigating the effectiveness of hybrid approaches that combine multiple models or ensemble techniques can be a worthwhile avenue for future research. By leveraging the strengths of different models, it may be possible to achieve even higher accuracy and robustness in classifying highly unbalanced data on edge devices.
- **Real-time Species Classification**: Compiling the trained model for edge device to extend the capabilities of the deployed edge device to not only detect the presence of fish but also accurately classify them into different species. This can provide valuable insights into the practical implications and challenges associated with deploying deep learning models for classifying naturally unbalanced data in different conditions.
- **Further enhancements**: Evaluating the robustness and generalization of the deployed model in real-world scenarios with varying lighting conditions, water conditions, and fish appearances. This includes testing the perform-

ance of the models across different locations, seasons, and fish species to ensure their reliability and applicability in diverse environments.

By exploring these aspects related to real-time testing and real-time fish detection and species classification on edge devices, researchers can contribute to the development of effective and practical solutions for monitoring and understanding aquatic ecosystems, supporting environmental conservation efforts, and enabling real-time decision-making in various applications such as fisheries management or aquatic research.

Bibliography

[1] W. Bank. 'Oceans, fisheries and coastal economies.' (2021), [Online]. Available: https://data.worldbank.org/indicator.

[2] *Global fisheries and aquaculture at a glance — fao.org*, https://www.fao.org/3/cc0461en/online/sofia/2022/world-fisheries-aquaculture.html, [Accessed 21-May-2023].

[3] R. L. Naylor, R. W. Hardy, A. H. Buschmann, S. R. Bush, L. Cao, D. H. Klinger, D. C. Little, J. Lubchenco, S. E. Shumway and M. Troell, 'A 20-year retrospective review of global aquaculture,' *Nature*, vol. 591, no. 7851, pp. 551–563, Mar. 2021. DOI: 10.1038/s41586-021-03308-6. [Online]. Available: https://doi.org/10.1038/s41586-021-03308-6.

[4] Food and Agriculture Organization, *The state of world fisheries and aquaculture 2018 (SOFIA)*. Rome, Italy: Food & Agriculture Organization of the United Nations (FAO), Aug. 2018.

[5] E. C. D. G. for Maritime Affairs, Fisheries. and EUMOFA., *Freshwater aquaculture in the EU*. Publications Office, 2021. DOI: 10.2771/594002. [Online]. Available: https://data.europa.eu/doi/10.2771/594002.

[6] R. M. Connolly, D. V. Fairclough, E. L. Jinks, E. M. Ditria, G. Jackson, S. Lopez-Marcano, A. D. Olds and K. I. Jinks, 'Improved accuracy for automated counting of a fish in baited underwater videos for stock assessment,' *Frontiers in Marine Science*, vol. 8, p. 658 135, 2021.

[7] S. Shakya *et al.*, 'Analysis of artificial intelligence based image classification techniques,' *Journal of Innovative Image Processing (JIIP)*, vol. 2, no. 01, pp. 44–54, 2020.

[8] V. Kandimalla, M. Richard, F. Smith, J. Quirion, L. Torgo and C. Whidden, 'Automated detection, classification and counting of fish in fish passages with deep learning,' *Frontiers in Marine Science*, p. 2049, 2022.

[9] C. Spampinato, Y.-H. Chen-Burger, G. Nadarajan and R. B. Fisher, 'Detecting, tracking and counting fish in low quality unconstrained underwater videos.,' *VISAPP (2)*, vol. 2008, no. 514-519, p. 1, 2008.

[10] B. J. Boom, M. Pollefeys, L. Van Gool, P. L. Goethals, H. De Meyer, S. Verstockt and R. E. T. Vanstreels, 'Fish4knowledge: Collecting and analyzing massive coral reef fish video data using object recognition techniques,' in *Proceedings of the ACM international conference on Multimedia*, 2012, pp. 1409–1412.

[11] W. Xu and S. Matzner, 'Underwater fish detection using deep learning for water power applications,' in *2018 International conference on computational science and computational intelligence (CSCI)*, IEEE, 2018, pp. 313–318.

[12] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition,' *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] S. Cui, Y. Zhou, Y. Wang and L. Zhai, 'Fish detection using deep learning,' *Applied Computational Intelligence and Soft Computing*, vol. 2020, 2020.

[14] A. Periola, A. Alonge and K. Ogudo, 'Edge computing for big data processing in underwater applications,' *Wireless Networks*, vol. 28, no. 5, pp. 2255–2271, 2022.

[15] C. Zhou, C. Sun, K. Lin, D. Xu, Q. Guo, L. Chen and X. Yang, 'Handling water reflections for computer vision in aquaculture,' *Transactions of the ASABE*, vol. 61, no. 2, pp. 469–479, 2018.

[16] A. Salman, A. Jalal, F. Shafait, A. Mian, M. Shortis, J. Seager and E. Harvey, 'Fish species classification in unconstrained underwater environments based on deep learning,' *Limnology and Oceanography: Methods*, vol. 14, no. 9, pp. 570–585, 2016.

[17] T. Oosting, B. Star, J. H. Barrett, M. Wellenreuther, P. A. Ritchie and N. J. Rawlence, 'Unlocking the potential of ancient fish dna in the genomic era,' *Evolutionary applications*, vol. 12, no. 8, pp. 1513–1522, 2019.

[18] C. Alcaraz, Z. Gholami, H. R. Esmaeili and E. Garcıa-Berthou, 'Herbivory and seasonal changes in diet of a highly endemic cyprinodontid fish (aphanius farsicus),' *Environmental Biology of Fishes*, vol. 98, no. 6, pp. 1541–1554, 2015.

[19] A. A. dos Santos and W. N. Gonçalves, 'Improving pantanal fish species recognition through taxonomic ranks in convolutional neural networks,' *Ecological Informatics*, vol. 53, p. 100 977, 2019.

[20] S. Mahesh, A. Manickavasagan, D. Jayas, J. Paliwal and N. White, 'Feasibility of near-infrared hyperspectral imaging to differentiate canadian wheat classes,' *Biosystems Engineering*, vol. 101, no. 1, pp. 50–57, 2008.

[21] B. Zion, 'The use of computer vision technologies in aquaculture–a review,' *Computers and electronics in agriculture*, vol. 88, pp. 125–132, 2012.

[22] C. Rillahan, M. D. Chambers, W. H. Howell and W. H. Watson III, 'The behavior of cod (gadus morhua) in an offshore aquaculture net pen,' *Aquaculture*, vol. 310, no. 3-4, pp. 361–368, 2011.

[23] V. M. Papadakis, I. E. Papadakis, F. Lamprianidou, A. Glaropoulos and M. Kentouri, 'A computer-vision system and methodology for the analysis of fish behavior,' *Aquacultural engineering*, vol. 46, pp. 53–59, 2012.

[24] S. Marini, E. Fanelli, V. Sbragaglia, E. Azzurro, J. Del Rio Fernandez and J. Aguzzi, 'Tracking fish abundance by underwater image recognition,' *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[25] C.-H. Tseng, C.-L. Hsieh and Y.-F. Kuo, 'Automatic measurement of the body length of harvested fish using convolutional neural networks,' *Biosystems Engineering*, vol. 189, pp. 36–47, 2020.

[26] L. Chen, X. Yang, C. Sun, Y. Wang, D. Xu and C. Zhou, 'Feed intake prediction model for group fish using the mea-bp neural network in intensive aquaculture,' *Information Processing in Agriculture*, vol. 7, no. 2, pp. 261–271, 2020.

[27] C. Zhou, D. Xu, K. Lin, C. Sun and X. Yang, 'Intelligent feeding control methods in aquaculture with an emphasis on fish: A review,' *Reviews in Aquaculture*, vol. 10, no. 4, pp. 975–993, 2018.

[28] X. Yang, S. Zhang, J. Liu, Q. Gao, S. Dong and C. Zhou, 'Deep learning for smart fish farming: Applications, opportunities and challenges,' *Reviews in Aquaculture*, vol. 13, no. 1, pp. 66–90, 2021.

[29] R. B. Fisher, Y.-H. Chen-Burger, D. Giordano, L. Hardman, F.-P. Lin *et al.*, *Fish4Knowledge: collecting and analyzing massive coral reef fish video data*. Springer, 2016, vol. 104.

[30] S. Choi, 'Fish identification in underwater video with deep convolutional neural network: Snumedinfo at lifeclef fish task 2015.,' in *CLEF (Working Notes)*, 2015.

[31] S. Jain, H. Salman, A. Khaddaj, E. Wong, S. M. Park and A. Madry, 'A data-based perspective on transfer learning,' *Journal Name*, vol. X, no. X, pp. X–X, 2023.

[32] X. Li, M. Shang, H. Qin and L. Chen, 'Fast accurate fish detection and recognition of underwater images with fast r-cnn,' in *OCEANS 2015-MTS/IEEE Washington*, IEEE, 2015, pp. 1–5.

[33] R. Girshick, 'Fast r-cnn,' in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[34] D. Zhang, G. Kopanas, C. Desai, S. Chai and M. Piacentino, 'Unsupervised underwater fish detection fusing flow and objectiveness,' in *2016 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, IEEE, 2016, pp. 1–7.

[35] D. Rathi, S. Jain and S. Indu, 'Underwater fish species classification using convolutional neural network and deep learning,' in *2017 Ninth international conference on advances in pattern recognition (ICAPR)*, Ieee, 2017, pp. 1–6.

[36] R. Mandal, R. M. Connolly, T. A. Schlacher and B. Stantic, 'Assessing fish abundance from underwater video using deep neural networks,' in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, pp. 1–6.

[37] S. S. Nair, S. NM, F. A. Mon and K. Suthendran, 'Under water fish species recognition,' *International Journal of Pure and Applied Mathematics*, vol. 118, no. 7, pp. 357–361, 2018.

[38] S. Raveendran, M. D. Patil and G. K. Birajdar, 'Underwater image enhancement: A comprehensive review, recent trends, challenges and applications,' *Artificial Intelligence Review*, vol. 54, no. 7, pp. 5413–5467, 2021.

[39] M. F. Duarte, T. Cunha, P. Deusdado, J. M. Almeida and J. P. Barroso, 'Turbid: Toward ubiquitous underwater robotic vision through image dataset,' *Journal of Field Robotics*, vol. 33, no. 3, pp. 397–417, 2016.

[40] J. Yang, X. Zhang, J. Liang, Y. Wang and X. Liu, 'Underwater image enhancement using an adaptive multi-scale decomposition method based on retinex theory for improving object detection performance in turbid water environments,' *Sensors*, vol. 19, no. 22, p. 4915,

[41] D. Li and L. Du, 'Recent advances of deep learning algorithms for aquacultural machine vision systems with emphasis on fish,' *Artificial Intelligence Review*, vol. 55, no. 5, pp. 4077–4116, 2022.

[42] G. G. Monkman, K. Hyder, M. J. Kaiser and F. P. Vidal, 'Using machine vision to estimate fish length from images using regional convolutional neural networks,' *Methods in Ecology and Evolution*, vol. 10, no. 12, pp. 2045–2056, 2019.

[43] J. Chen and X. Ran, 'Deep learning with edge computing: A review,' *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019. DOI: 10.1109/JPROC.2019.2921977.

[44] X. Yang, Z. Song, Y. Liu and C. Zhou, 'Deep learning for smart fish farming: Applications, opportunities and challenges,' *Preprint*, 2020.

[45] K. M. Knausgård, A. Wiklund, T. K. Sørdalen, K. T. Halvorsen, A. R. Kleiven, L. Jiao and M. Goodwin, 'Temperate fish detection and classification: A deep learning based approach,' *Applied Intelligence*, vol. 51, no. 4, pp. 1685–1697, 2021.

[46] S. Zhao, S. Zhang, J. Liu, H. Wang, J. Zhu, D. Li and R. Zhao, 'Application of machine learning in intelligent fish aquaculture: A review,' *Aquaculture*, vol. 531, p. 735 963, 2021.

[47] M. A. R. Sarker, M. A. Hossain and M. A. R. Sarkar, 'Fish species recognition using deep learning techniques,' in *Proceedings of the International Conference on Computer Science and Engineering*, 2018, pp. 1–6.

[48] S. Kwon, J.-H. Kim and J.-H. Lee, 'Fish classification using deep convolutional neural network with transfer learning,' *Journal of Marine Science and Engineering*, vol. 7, no. 6, p. 186, 2019.

[49] A. Jalal, A. Salman, A. Mian, M. Shortis and F. Shafait, 'Fish detection and species classification in underwater environments using deep learning with temporal information,' *Ecological Informatics*, vol. 57, p. 101 088, 2020.

[50] Y. Sun, X. Wang and Y. Tang, 'A novel deep learning approach for fish species recognition based on convolutional neural network and softmax regression,' *Journal of Ocean University of China*, vol. 17, no. 5, pp. 1079–1084, 2018.

[51] H. Kim, J. Koo, D. Kim, S. Jung, J.-U. Shin, S. Lee and H. Myung, 'Image-based monitoring of jellyfish using deep learning architecture,' *IEEE sensors journal*, vol. 16, no. 8, pp. 2215–2216, 2016.

[52] M. S. Hossain, M. A. Rahman, M. A. Islam and M. A. Hossain, 'Underwater fish species recognition using deep learning techniques,' in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 2019, pp. 1–6. DOI: 10.1109/ICREST.2019.8711657.

[53] S. Singh, R. Singh, A. Singh and V. Singh, 'Fish biomass estimation using machine learning techniques,' *Reviews in Aquaculture*, vol. 11, no. 3, pp. 725–739, 2019.

[54] A.-R. Al-Ali, R. Al-Sayyed and S. Al-Maadeed, 'Fish recognition using convolutional neural network,' in *2018 International Conference on Computer and Applications (ICCA)*, IEEE, 2018, pp. 1–5.

[55] E. C. D. G. for Maritime Affairs, Fisheries. and EUMOFA., *Portion trout in the EU: price structure in the supply chain focus on Germany, Italy and Poland.* Publications Office, 2021. DOI: 10.2771/98441. [Online]. Available: https://data.europa.eu/doi/10.2771/98441.

[56] M. Sonka, V. Hlavac and R. Boyle, 'Image pre-processing,' in *Image Processing, Analysis and Machine Vision*, Springer US, 1993, pp. 56–111. DOI: 10.1007/978-1-4899-3216-7_4. [Online]. Available: https://doi.org/10.1007/978-1-4899-3216-7_4.

[57] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal

Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[58] *Models - Image classification | Coral — coral.ai*, https://coral.ai/models/image-classification/, [Accessed 02-May-2023].

[59] S. Madeleine, *Normalization, zero centering and standardization of ct images — imaios.com*, https://www.imaios.com/en/resources/blog/ct-images-normalization-zero-centering-and-standardization, [Accessed 24-May-2023].

[60] B. K. P, *WHEN and WHY are batches used in machine learning ? — medium.com*, https://medium.com/analytics-vidhya/when-and-why-are-batches-used-in-machine-learning-acda4eb00763, [Accessed 24-May-2023].

[61] *Tf.keras.utils.image_dataset_from_directory | TensorFlow v2.12.0*, https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory, [Accessed 24-May-2023].

[62] L. Perez and J. Wang, 'The effectiveness of data augmentation in image classification using deep learning,' *arXiv preprint arXiv:1712.04621*, 2017.

[63] *Tf.keras.preprocessing.image.ImageDataGenerator*, https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator, [Accessed 24-May-2023].

[64] *NumPy — numpy.org*, https://numpy.org/, [Accessed 24-May-2023].

[65] *Effective Tensorflow 2 | TensorFlow Core*, https://www.tensorflow.org/guide/effective_tf2, [Accessed 04-May-2023].

[66] *ImageNet — image-net.org*, https://www.image-net.org/, [Accessed 26-May-2023].

[67] M. Coşkun, Ö. YILDIRIM, U. Ayşegül and Y. Demir, 'An overview of popular deep learning methods,' *European Journal of Technique (EJT)*, vol. 7, no. 2, pp. 165–176, 2017.

[68] J. C. Ovalle, C. Vilas and L. T. Antelo, 'On the use of deep learning for fish species recognition and quantification on board fishing vessels,' *Marine Policy*, vol. 139, p. 105 015, 2022.

[69] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[70] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, 'Mobilenets: Efficient convolutional neural networks for mobile vision applications,' *arXiv preprint arXiv:1704.04861*, 2017.

[71] 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.

[72] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le and H. Adam, 'Searching for mobilenetv3,' *arXiv preprint arXiv:1905.02244*, 2019.

[73] *Tf.keras.applications.resnet50.ResNet50 | TensorFlow v2.12.0*, `https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/ResNet50`, [Accessed 04-May-2023].

[74] *Module: Tf.keras.applications.mobilenet | TensorFlow v2.12.0*, `https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet`, [Accessed 06-May-2023].

[75] *Module: Tf.keras.applications.mobilenet_v2 | TensorFlow v2.12.0*, `https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2`, [Accessed 06-May-2023].

[76] *Tf.keras.applications.MobileNetV3Large | TensorFlow v2.12.0*, `https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV3Large`, [Accessed 06-May-2023].

[77] M. Lin, Q. Chen and S. Yan, 'Network in network,' *arXiv preprint arXiv:1312.4400*, 2013.

[78] A. F. Agarap, 'Deep learning using rectified linear units (relu),' *arXiv preprint arXiv:1803.08375*, 2018.

[79] R. Parhi and R. D. Nowak, 'The role of neural network activation functions,' *IEEE Signal Processing Letters*, vol. 27, pp. 1779–1783, 2020.

[80] S. Sharma, S. Sharma and A. Athaiya, 'Activation functions in neural networks,' *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[81] C. Cortes, M. Mohri and A. Rostamizadeh, 'L2 regularization for learning kernels,' *arXiv preprint arXiv:1205.2653*, 2012.

[82] S. Ioffe and C. Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift,' in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[83] *tf.keras.Model | TensorFlow v2.12.0*, `https://www.tensorflow.org/api_docs/python/tf/keras/Model#compile`, [Accessed 04-May-2023], 2023.

[84] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization,' *arXiv preprint arXiv:1412.6980*, 2014.

[85] *Tf.keras.optimizers.Adam | TensorFlow v2.12.0*, `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam`, [Accessed 06-May-2023].

[86] *tf.keras.Model*, https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit, [Accessed 04-May-2023], 2023.

[87] *TensorBoard | TensorFlow*, https://www.tensorflow.org/tensorboard, [Accessed 06-May-2023].

[88] *Tf.keras.callbacks.TensorBoard*, https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/TensorBoard, [Accessed 06-May-2023].

[89] *tf.keras.Model | TensorFlow v2.12.0*, https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit, [Accessed 06-May-2023], 2023.

[90] J. Ni, J. Li and J. McAuley, 'Justifying recommendations using distantly-labeled reviews and fine-grained aspects,' pp. 188–197, 2019.

[91] *Tf.keras.metrics.Accuracy | TensorFlow v2.12.0*, https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Accuracy, [Accessed 07-May-2023].

[92] *Tf.keras.metrics.Precision | TensorFlow v2.12.0*, https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Precision, [Accessed 07-May-2023].

[93] *Tf.keras.metrics.Recall | TensorFlow v2.12.0*, https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Recall, [Accessed 07-May-2023].

[94] Y. Liu, Z. Wang and J. Li, 'Comparing deep learning models with relative confusion matrix,' *Proc. ACM Mach. Learn. Res.*, vol. 8, no. 1, pp. 1–29, 2021, ISSN: 2376-3639. DOI: 10.1145/3411764.3445673. [Online]. Available: https://doi.org/10.1145/3411764.3445673.

[95] *Sklearn.metrics.confusion_matrix*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html, [Accessed 07-May-2023].

[96] A. P. Bradley, 'The use of the area under the roc curve in the evaluation of machine learning algorithms,' *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[97] *Sklearn.metrics.auc*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html, [Accessed 11-May-2023].

[98] *Sklearn.metrics.roc_curve*, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html, [Accessed 11-May-2023].

A - Github repository

All code used in this document is included in the Github repository linked below. Access can be granted upon request.

Github repository link

- `https://github.com/1904varun/master_thesis`

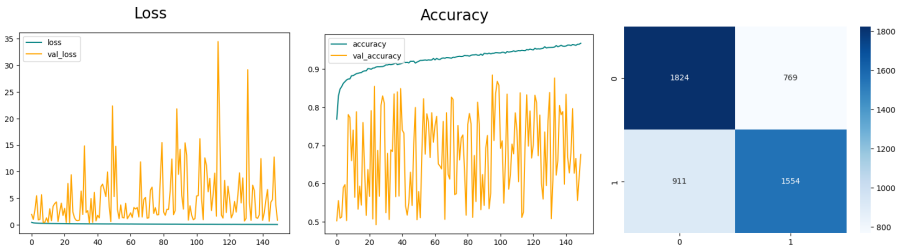
B - Additional Results

B1 - Unfreezing 10 layers

Model Combination Type 1 (3.6.1)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Total Images	50882	50882	50882	50882
Trout Images	25270	25270	25270	25270
Mature Images	25612	25612	25612	25612
Batch Size	64	64	64	64
Train Size	557	557	557	557
Validation Size	159	159	159	159
Test Size	80	80	80	80
Total Params	24,637,826	3,754,690	2,914,882	4,883,330
Trainable Params	5,515,778	2,114,050	1,389,378	1,888,898
Learning Rate	0.001	0.001	0.001	0.001
Epochs	150	150	150	150
Training Time (mins)	263	155	146	126
Accuracy	66.8%	91.6%	90.6%	84.8%
Precision	67.1%	91.0%	90.8%	85.5%
Recall	62.7%	91.8%	89.9%	83.1%
F1 Score	0.64	0.91	0.89	0.84

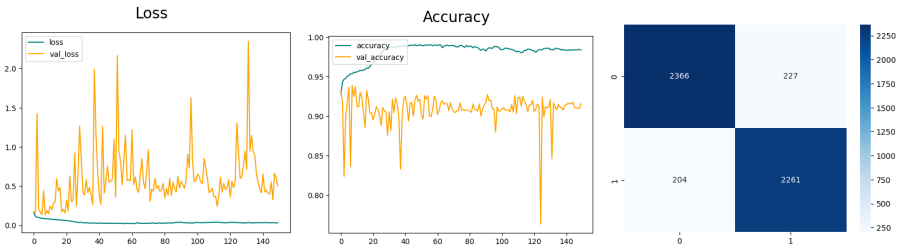
Table B.1: Results for models with unfreezing 10 layers

The loss curves, accuracy curves and confusion matrix for each of the model when fine-tuned with unfreezing 10 layers are presented in the Fig. B.1, Fig. B.2, Fig B.3.



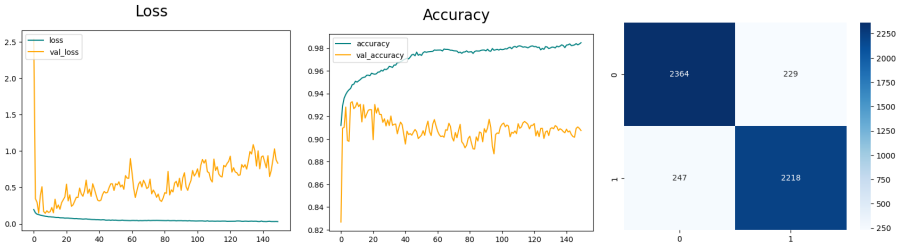
(a) Loss curve **(b)** Accuracy curve **(c)** Confusion matrix

Figure B.1: ResNet-50 performance with further fine-tuning



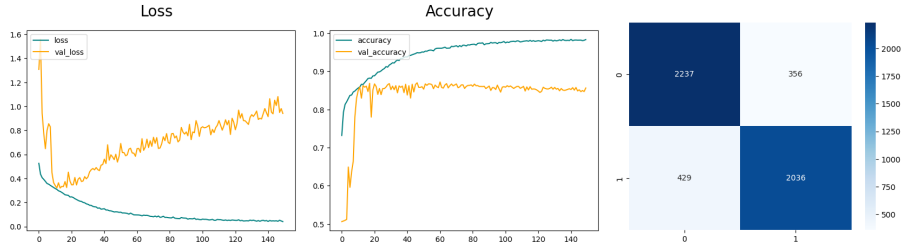
(a) Loss curve **(b)** Accuracy curve **(c)** Confusion matrix

Figure B.2: MobileNet V1 performance with further fine-tuning



(a) Loss curve **(b)** Accuracy curve **(c)** Confusion matrix

Figure B.3: MobileNet V2 performance with further fine-tuning



(a) Loss curve **(b)** Accuracy curve **(c)** Confusion matrix

Figure B.4: MobileNet V3 performance with further fine-tuning

B2 - Unfreezing 20 layers

Model Combination Type 1 (3.6.1)				
Model Metrics	**ResNet-50**	**MobileNet V1**	**MobileNet V2**	**MobileNet V3**
Total Images	50882	50882	50882	50882
Trout Images	25270	25270	25270	25270
Mature Images	25612	25612	25612	25612
Batch Size	64	64	64	64
Train Size	557	557	557	557
Validation Size	159	159	159	159
Test Size	80	80	80	80
Total Params	24,637,826	3,754,690	2,914,882	4,883,330
Trainable Params	9,981,442	2,388,482	1,862,978	2,427,778
Learning Rate	0.001	0.001	0.001	0.001
Epochs	150	150	150	150
Training Time (mins)	288	156	168	172
Accuracy	63.9%	91.0%	91.1%	85.7%
Precision	93.9%	88.9%	90.0%	85.2%
Recall	27.9%	93.2%	92.0%	85.9%
F1 Score	0.42	0.90	0.90	0.85

Table B.2: Results for models with unfreezing 20 layers

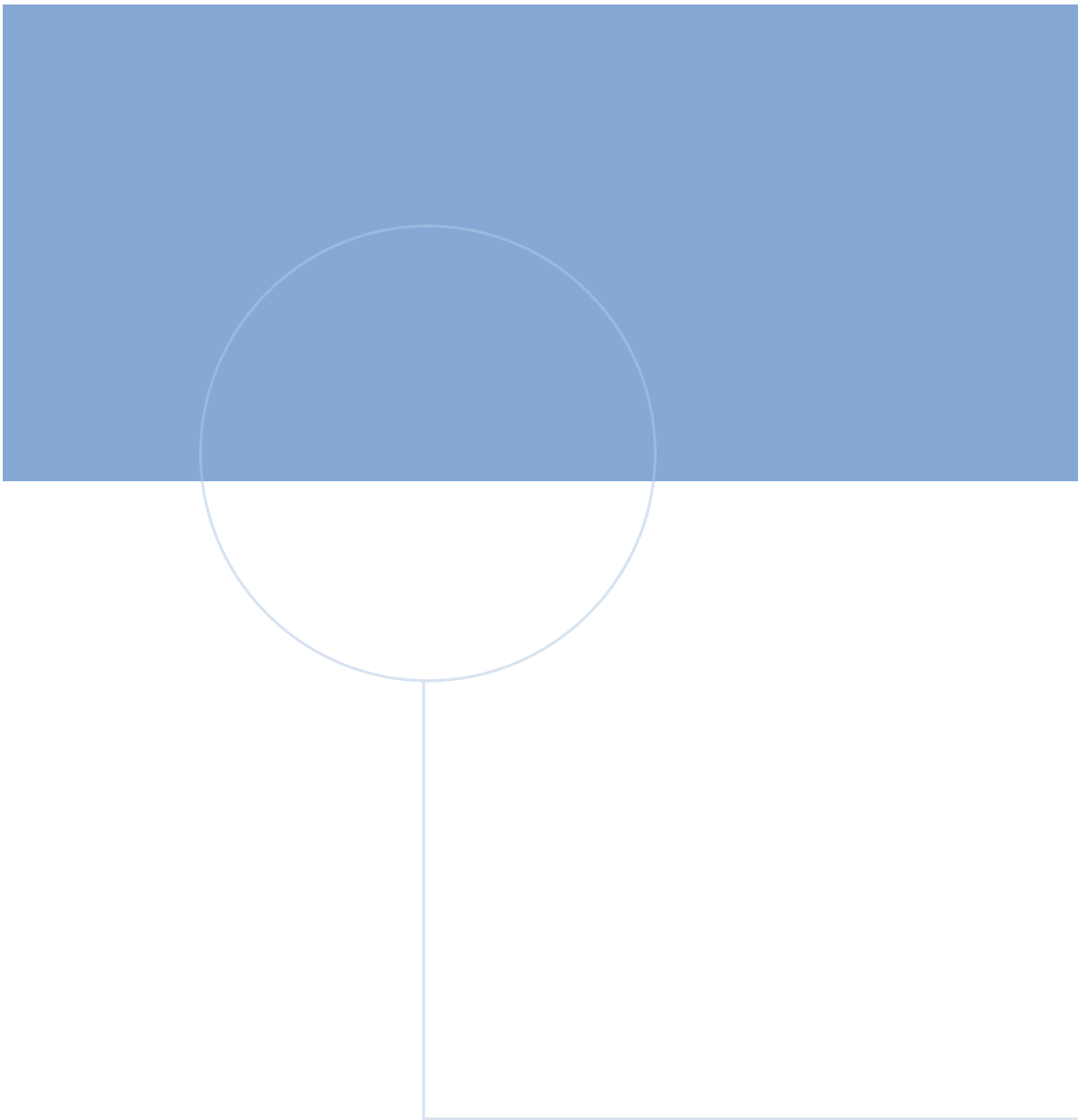
C - Sample Images

C1 - Images with dimensions 224x224 pixels which is input size for pre-trained models

(a) Sample from trout class

(b) Sample from mature class

Figure C.1: Sample images with size 224x224 pixels



NTNU

Norwegian University of
Science and Technology