الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

Mémoire de MASTER

Domaine : Mathématiques et Informatique
Filière : Informatiques
Option : Réseaux, Systèmes et Applications Réparties

Par:
Hadj Mahmoud BEDERINA

THEME

Conception and Implementation of an AI-Powered Computer Vision Application Prototype For Detecting, Counting, And Classifying Tilapia Fish Eggs

---

Soutenu publiquement le 25-09-2022 devant le jury composé de:

Mr. Mohamed El Habib MAICHA	*M.C. (A)*	*Président*
Mr. Mohammed BELKHIRI	*Professeur*	*Examinateur*
Mr. Younes GUELLOUMA	*M.C.(A)*	*Examinateur*
Mr. Mohammed Redha BOUZIDI	*M.C.(A)*	*Encadreur*

Année Universitaire 2022/2023

"For a successful technology,
reality must take precedence over public relations,
for nature cannot be fooled."

Richard P. Feynman

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my family for their understanding and support during the immense effort I put into this research. Their encouragement was a constant source of strength and motivation.

Special thanks are extended to my brother, *Bilal*, who has been an irreplaceable pillar of support throughout this journey. His assistance has been invaluable in accomplishing everything that this work entails.

I am also indebted to my supervisor, *Dr. Mohammed Redha Bouzidi*, whose expertise and guidance have not only facilitated my research process but also opened up new opportunities for me. The insights and wisdom I have gained under their tutelage are immeasurable and will continue to influence my academic and professional future. To also extend my gratitude to the director of the Direction of Fisheries and Marine Resources in Laghouat, *Mr. Mourad Beida*, for his generous sharing of expertise in the field.

I would like to thank the members of the LTSS laboratory for their supportive environment and for providing me access to essential equipment that was crucial for this research. Their collegiality made a significant difference in my academic experience.

Lastly, I reflect on this academic endeavour with gratitude for the learning experiences it has provided me. I recognize the resilience and commitment required to complete this work, and it serves as a reminder of what can be achieved through focused and sustained effort.

المقدمة

يعتبر قطاع التربيات المائية عاملًا مهمًا في الأمن الغذائي العالمي وقد شهد نموًا متسارعًا، مساهمًا في أكثر من نصف إنتاج الأسماك العالمي. في الجزائر، ينمو هذا القطاع، مدعومًا بمبادرات حكومية تهدف إلى التنويع الاقتصادي والأمن الغذائي. تعتبر عملية العد والتصنيف اليدوي لبيض الأسماك، وتحديدًا سمك البلطي، عائقًا في كفاءة التربيات المائية، حيث تكون مكلفة من حيث اليد العاملة وعرضة لأخطاء العنصر البشري. تهدف هذه الدراسة إلى تطوير نظام نموذجي للتخفيف من هذه المشكلة، مع التركيز على تصنيف وعد بيض البلطي. يتألف النظام من تطبيق محمول مدمج مع نموذج Mask R-CNN لاكتشاف وتصنيف الأشياء.

نحن نستخدم تقنيات جديدة مثل تنقية البيانات لتصحيح مجموعة البيانات وأنبوب تعلم نقل متعدد المستويات لتدريب النموذج. وقد حقق نموذجنا قمة mAP تبلغ 7%.86، مع أداء قوي عبر مراحل التطوير المختلفة لبيض البلطي. تحمل النتائج وعدًا كبيرًا في تعزيز دورة الإنتاج في التربيات المائية من خلال أتمتة وتحسين عملية عد وتصنيف البيض. لهذا تأثيرات أوسع نطاقًا على صناعة الزراعة المائية، عالميًا وفي الجزائر، مقدمةً طريقًا لتحقيق الأهداف الإنتاجية.

Abstract

The aquaculture sector is instrumental in global food security and has seen exponential growth, contributing to over half of the global fish production. In Algeria, this sector is growing, backed by government initiatives aimed at economic diversification and food security. One bottleneck in aquaculture efficiency is the manual counting and classification of fish eggs, specifically tilapia, which is labour-intensive and prone to human error. This study aims to develop a prototype system to alleviate this issue, focusing on tilapia egg classification and counting. The system comprises a mobile application integrated with a Mask R-CNN model for object detection and classification.

We employ novel techniques such as dynamic dataset cleansing to rectify a skewed dataset and a multi-level transfer learning pipeline for model training. Our model achieved a peak mAP of 86.7%, with robust performance across different developmental stages of tilapia eggs. The results hold significant promise for enhancing the commercial production cycle in aquaculture by automating and optimizing egg counting and classification. This has broader implications for the aquaculture industry, both globally and in Algeria, offering a pathway to achieving the production goals set by the Ministry of Fisheries and Marine Resources.

Keywords: Aquaculture, Mask R-CNN, Transfer Learning, Automation, Dataset, Mobile Application

Résumé

Le secteur de l'aquaculture joue un rôle essentiel dans la sécurité alimentaire mondiale et a connu une croissance exponentielle, contribuant à plus de la moitié de la production mondiale de poisson. En Algérie, ce secteur est en pleine croissance, soutenu par des initiatives gouvernementales visant à la diversification économique et à la sécurité alimentaire. L'un des points critiques de l'efficacité de l'aquaculture est le comptage et la classification manuels des œufs de poisson, en particulier du tilapia, qui nécessitent beaucoup de main-d'œuvre et sont sujets à l'erreur humaine. Cette étude vise à développer un prototype de système pour pallier ce problème, en se concentrant sur la classification et le comptage des œufs de tilapia. Le système comprend une application mobile intégrée à un modèle Mask R-CNN pour la détection et la classification des objets.

Nous utilisons de nouvelles techniques telles que le nettoyage dynamique des ensembles de données pour rectifier un ensemble de données et un pipeline d'apprentissage par transfert à plusieurs niveaux pour l'entraînement du modèle. Notre modèle a atteint un pic de mAP de 86,7%, avec des performances robustes à différents stades de développement des œufs de tilapia. Les résultats sont très prometteurs pour l'amélioration du cycle de production commerciale en aquaculture grâce à l'automatisation et à l'optimisation du comptage et de la classification des œufs. Ceci a des implications plus larges pour l'industrie de l'aquaculture, à la fois au niveau mondial et en Algérie, offrant une voie pour atteindre les objectifs de production fixés par le Ministère de la Pêche et des Ressources Marines.

Mots-cls : Aquaculture, Mask R-CNN, Apprentissage par transfert, Automatisation, Ensemble de donnes, Application mobile

Contents

List of Figures

List of Tables

Introduction

Motivation

Aquaculture is a rapidly growing global industry that contributes significantly to the world's food security and livelihoods, particularly in developing countries[1]. The sector has experienced exponential growth over the past few decades, and today, it accounts for more than half of the total global fish production[2]. In Algeria, the aquaculture sector has seen significant development in recent years, with the government actively promoting its expansion to diversify the country's economy, create employment opportunities, and enhance food security[3], [4].

One of the most important aspects of aquaculture is the production of high-quality fish eggs, as it directly influences the success and sustainability of fish farming operations[5]. Tilapia (Oreochromis niloticus), a versatile and fast-growing fish species, has become increasingly popular in aquaculture worldwide, including in Algeria[2], [6]. Tilapia farming encompasses a range of practices, from breeding and hatchery operations to grow-out systems, such as ponds, tanks, floating cages, and recirculating aquaculture systems (RAS)[2]. A crucial part of the tilapia breeding process involves counting and classifying fish eggs, as it enables farmers to monitor reproductive success, optimize the stocking density of larval rearing systems, and manage broodstock effectively[5].

Accurate fish egg counting and classification are essential for improving productivity and sustainability in tilapia farming operations. However, several challenges exist in the current methods of counting and classifying tilapia eggs, including manual processes that are labour-intensive, time-consuming, and prone to human error[5], [7]. Additionally, there are challenges related to accurately counting fish eggs in Petri dishes due to issues such as specular reflections and low contrast between fertilized and unfertilized eggs[5].

Given these challenges, there has been a growing interest in developing automated systems and artificial intelligence (AI) applications to improve the efficiency and accuracy of fish egg counting and classification[5], [8]. Several commercial products, such as Syndel ProSorter and QuickSorter, have emerged to automate the process of sorting and counting fish eggs[9], [10].

Furthermore, research has been conducted to explore the application of deep learning techniques in automating fish egg counting, with promising results[8].

Problem and objectives

The manual counting of fish eggs is a labour-intensive, time-consuming, and error-prone process[5]. Specular reflections and low contrast in Petri dishes can make it challenging to accurately count fish eggs[5]. Inaccurate fish egg counting can lead to inconsistencies in aquaculture productivity and sustainability, impacting the overall efficiency of fish farming practices[11]. Ensuring accuracy in fish egg counting is crucial to the overall productivity and sustainability of aquaculture operations[5].

While many models prioritize real-time processing, the primary objective of this research is not necessarily real-time object detection but rather to focus on the accurate classification and counting of Tilapia eggs[12]. Given that the size of Tilapia eggs is generally small, the precision of detection becomes particularly critical. This underscores the need to opt for a more complex object detection and classification model, as the urgency for real-time processing becomes secondary to the model's accuracy[12].

Beyond the mere counting of fish eggs, the classification of these eggs into distinct developmental stages holds significant importance for the aquaculture industry. This classification serves a pivotal role in the commercial production cycle of sorting fish eggs. Eggs nearing hatching are prioritized for sale, those in the middle stages are prepared for hatching, and those in the early stages are moved to environments that accelerate and ensure their viability. This nuanced approach to egg classification and sorting has the potential to significantly optimize the production pipeline, thereby enhancing the overall efficiency and sustainability of aquaculture operations.

This innovative solution aims to revolutionize the current methods in practice by providing a more efficient, accurate, and cost-effective alternative to manual counting[8]. By automating the counting and classification process and prioritizing accuracy over speed, aquaculture operations can improve productivity and significantly reduce the margin of error[12].

In Algeria, aquaculture is an emerging industry with great potential for growth[6]. The government has initiated several aquaculture projects to boost the sector[3], [13], but there are still challenges to overcome, such as achieving the production goals set in 2005 by the Min-

istry of Fisheries and Marine Resources[14]. Implementing AI-based counting and classification methods for Tilapia eggs in Algeria could help address these challenges, driving the growth and sustainability of the aquaculture sector[8].

The development of a machine learning model prioritizing accuracy in object detection and classification of Tilapia eggs has the potential to revolutionize the current methods of fish egg counting. By providing a more accurate and cost-effective solution, this technology can significantly improve the productivity and sustainability of aquaculture on a global scale and in Algeria.

Scope and limitations

The **scope of the research** primarily centres around the application of Faster R-CNN and Mask R-CNN algorithm for fish egg counting and classification in Tilapia species [7][8][15][10]. It aims to delve into an analysis of the performance of these AI algorithms in automating the counting and classification process, with a particular focus on their accuracy and processing time [7][8][15][10]. Furthermore, the study seeks to explore the benefits of employing AI algorithms in aquaculture, specifically in the context of enhancing productivity and sustainability [11][5][6]. The research methodology will draw upon a multitude of sources, ranging from academic articles and aquaculture guides to insights provided by local experts.

There are, however, certain **limitations to the study**. First and foremost, the study's focus is limited to Tilapia species, suggesting that the findings may not be directly transferable to other fish species [11]. The research narrows its scope to Mask R-CNN and Faster R-CNN algorithms, thereby leaving out other AI algorithms and methods for fish egg counting and classification [7][8][15][10]. Technological limitations, such as the computational resources at our disposal and access to high-quality data, may impact the efficacy of the AI algorithms and consequently the study's outcomes [7][8][15][10]. Moreover, methodological constraints, such as the challenge of accurately counting fish eggs in Petri dishes or laboratory trays owing to issues like specular reflections and low contrast, may potentially influence the results [5].

The study also makes certain **assumptions**. It presupposes that the behaviour and characteristics of Tilapia fish eggs remain consistent across different environments and contexts [11]. The research hinges on the belief that the object detection algorithm Mask R-CNN is apt for the task of fish egg counting and classification [7][8][15][10]. Additionally, it is assumed that

the performance of these AI algorithms in this specific context will be representative of their performance in other aquaculture applications [7][8][15][10].

These **limitations and assumptions** have certain implications. The study's limitations may affect the extent to which the results can be generalized to other fish species and different AI algorithms for fish egg counting and classification. The assumptions made could impact the interpretation of the results, as they may not hold true in all contexts or for all fish species. Future research, therefore, should consider exploring the applicability of the findings to other fish species and testing the performance of different AI algorithms for fish egg counting and classification in aquaculture.

Structure of the thesis

This thesis is organized into multiple sections, each serving a specific purpose in guiding the research process.

Introduction

The introduction sets the stage for the research by providing the motivation behind the study, outlining the problem and objectives, and discussing the limitations of the study.

Chapter 1: Background and Literature Review

This chapter offers an exhaustive background study crucial for the understanding of the thesis. It covers:

- Tilapia Fish Farming and Reproduction Cycle
- Existing solutions and challenges in tilapia egg counting and classification
- Computer vision and AI applications in aquaculture
- Overview of object detection and object counting
- Metrics used to evaluate model performance
- Feature Pyramid Networks (FPNs)
- Residual Networks
- Faster R-CNN

- Mask R-CNN

Chapter 2: Methodology

This chapter details the methods and procedures employed in the research. It includes:

- Smartphone-based Data Acquisition System Architecture and Design Considerations
- Dataset Collection, Annotation, Augmentation, Analysis, and Dynamic Cleansing
- Model Development and Training, including hyperparameter tuning and a multi-level transfer learning pipeline
- Mobile Application Development, both server-side and client-side

Chapter 3: Results and Discussion

This chapter presents the findings of the research and discusses their implications. It focuses on:

- Dataset Variations and Their Impact
- Impact of Multi-level Transfer Learning
- Analysis of Metrics
- System Testing and Efficiency

Conclusions

The final section summarizes the research, discusses its contributions, and suggests directions for future work.

1. Chapter 1: Background and Literature Review

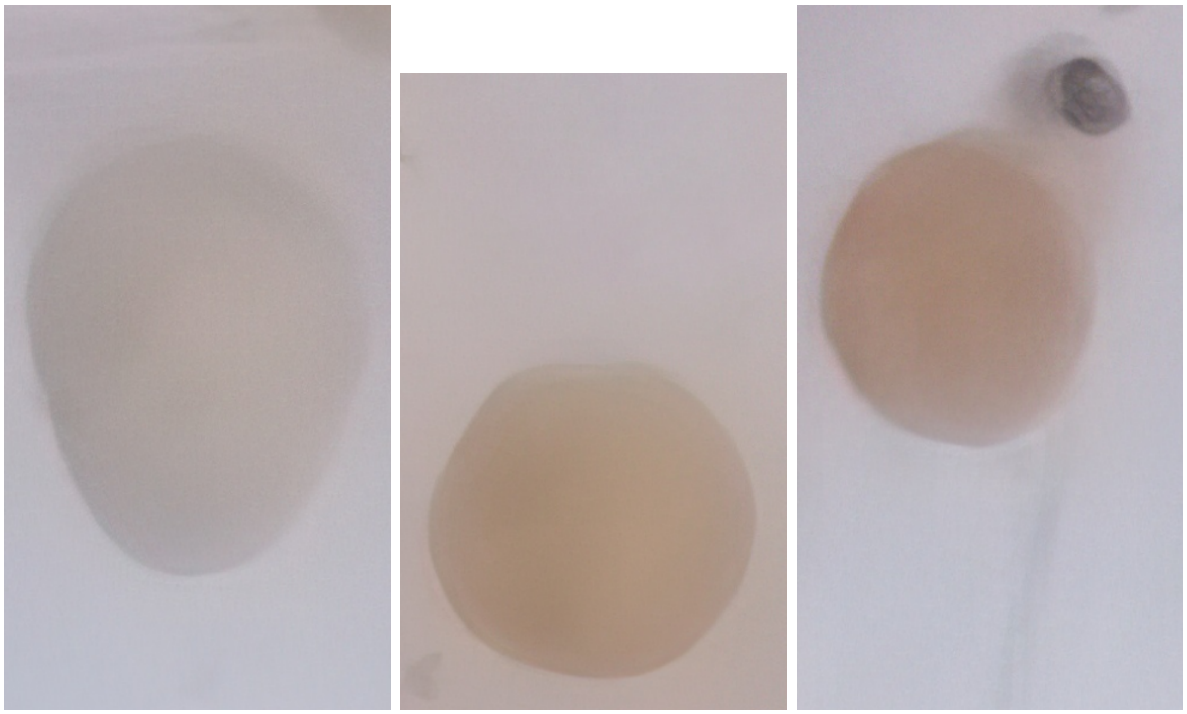
Introduction

The first chapter aims to establish the contextual and theoretical foundations upon which this research is built. It starts with an overview of Tilapia fish farming and its reproductive cycle, highlighting its significance in aquaculture. The chapter then moves on to discuss existing solutions and challenges specifically in the realm of tilapia egg counting and classification. This provides a segue into broader discussions on the applications of computer vision and AI in aquaculture. We further explain the underlying technologies and metrics that will be used, such as object detection and counting methods and performance evaluation metrics. Lastly, we delve into specific algorithms and architectures such as Feature Pyramid Networks, Residual Networks, Faster R-CNN, and Mask R-CNNthat.

1.1 Tilapia Fish Farming and Reproduction Cycle

The importance of tilapia fish farming has been well-established, especially for mono-sex red tilapia, which is a popular choice due to its high market demand, fast growth, and appealing colour[16], [17]. Tilapia farming has multiple benefits, such as being adaptable to various farming environments, efficient feed conversion, and resistance to diseases‘[18]. One major advantage of mono-sex red tilapia farming is the control of reproduction, as mono-sex populations (typically male) can significantly reduce uncontrolled breeding and enhance overall growth rates[19], [20].

The reproduction cycle of tilapia consists of several stages, including courtship, mating, fertilization, and incubation of eggs[16]. Some tilapia species, such as Nile tilapia, exhibit mouthbrooding behaviour, where the female incubates the eggs in her mouth until they hatch[21][22]. The development of tilapia eggs undergoes various stages, such as the cleavage stage (Figure 1(a)), blastula stage, gastrula stage (Figure 1(b))[22], and organogenesis stage (Figure 1(c)),



((a)) The cleavage stage

((b)) The blastula-gastrula stage

((c)) The organogenisis stage

Figure 1: The major developmental stages of Tilapia fish eggs.

with the eyed stage as an advanced stage of embryonic development[23]. Our research will not focus on these specific development stages, but rather a general perspective of fertilized and viable, in which the latter is the eyed stage. Table 1 shows the distinctive features of each developmental stage. The hatching process largely depends on water temperature, with warmer temperatures leading to faster hatching[23].

Table 1: The distinctive features between the major developmental stages of Tilapia fish eggs.

Stage	**Shape**	**Colour**
Cleavage	Round, oval, ovoid	White, pale-yellow
Between blastula and gastrula	Round, ovoid	Orange, Brown
Organogenesis	Round body, round black eyes, tail	Orange, Brown

1.2 Existing solutions and challenges in tilapia egg counting and classification

Current methods for counting and classifying tilapia eggs involve manual inspection, Automatic Visual Inspection (AVI) systems, and deep learning techniques. Manual inspection is labour-intensive, time-consuming, and prone to human error, which can lead to inaccurate results[24]. AVI systems have been employed for fish egg development analysis in commercial and experimental fish production systems and ecological monitoring[24]. These systems, however, may not be efficient or effective in all circumstances, and their accuracy may depend on the quality of the images used for analysis.

Deep learning techniques have shown promise in underwater fish detection and object localization, including fish eggs[25]. State-of-the-art deep learning models, such as YOLOv3, Mask-RCNN, and YOLOv4, have been applied for fish detection, classification, and tracking, combined with the Norfair tracking library[26]. These models have demonstrated success in various aquatic environments, but their performance may be impacted by factors such as water turbidity, lighting conditions, and the presence of other objects or organisms in the images.

One challenge in counting and classifying tilapia eggs is the accurate detection and calculation of group objects, such as fish eggs in a cluster[27]. Conventional methods often result in errors due to occlusions, overlaps, and varying sizes of the eggs. Researchers have been working on finding optimal detection and calculation solutions to address this issue[27].

Potential improvements to the current methods include refining deep learning models to better handle the specific challenges associated with fish egg detection and classification, such as occlusions and varying sizes. This could be achieved by using larger and more diverse datasets for training and by incorporating additional features, such as egg shape, texture, and colour, into the models.

Another avenue for improvement is the development of more advanced computer vision algorithms that can efficiently handle group object detection and counting. These algorithms should be capable of overcoming occlusion and overlap issues, as well as accounting for variations in egg size and shape.

1.3 Computer vision and AI applications in aquaculture

The applications of computer vision and AI in aquaculture have shown great potential in improving productivity, sustainability, and efficiency in the industry. These technologies have been employed in various aspects of aquaculture, including fish detection, counting, and monitoring, as well as addressing challenges related to water quality and fish susceptibility.

1. Fish detection and counting: Computer vision techniques have been developed for detecting and counting fish in underwater videos, which can help manage fish populations and optimize feeding strategies **AutomaticFishCounting**, [28], [29].
2. Biomass estimation: AI-based systems have been used to estimate fish biomass and size distributions, which can facilitate optimal feeding and improve overall farm management **DeepLearningFishWeight**, [30].
3. Water quality monitoring: AI-driven water quality monitoring and management systems have been employed in aquaculture to address the imbalance of the water environment and the decline in aquatic product quality [31]. This can help maintain optimal water conditions, prevent disease outbreaks, and improve fish health.
4. Feeding management: Machine learning algorithms have been used to predict feeding patterns and optimize feeding schedules in aquaculture, reducing waste and improving fish growth**FeedingPredictionModelCulturedFish**.
5. Fish health and behaviour analysis: Computer vision systems have been developed to analyze the swimming behaviour of farmed fish, which can help monitor fish health and detect early signs of stress or disease **SwimmingBehaviorFarmedFish**, [32].
6. Sustainability and environmental impact: AI applications in aquaculture have been explored to reduce the industry's environmental footprint and promote long-term success[29], [33].
7. Smart cage culture management: AI and IoT-based systems have been developed for managing cage culture in aquaculture, improving efficiency, and sustainability **AutomaticFishCounting** An example is the AIoT smart cage culture management system, which combines AI and IoT technologies to monitor and control various farm operations remotely[34].

These applications of AI and computer vision demonstrate the promising future of the aquaculture industry, offering potential benefits in terms of productivity, sustainability, and efficiency.

Further research and development in this area can lead to the more widespread adoption of these technologies and their continued evolution.

1.4 Overview of object detection and object counting

Object detection is a key task in computer vision that involves identifying and locating objects within images or video frames[35]. This process consists of two main steps: object localization, which determines the location of objects in the image, and object classification, which assigns a class label to each detected object[36].

Several object detection methods have been developed over the years, ranging from traditional computer vision techniques to modern deep learning-based approaches. Some popular object detection algorithms include:

1. R-CNN (Region-based Convolutional Neural Networks): This method uses selective search to generate region proposals and then utilizes a CNN to classify each region proposal. However, R-CNN tends to be slow due to the large number of region proposals and the independent processing of each proposal.
2. Fast R-CNN: An improvement over R-CNN, Fast R-CNN uses a single CNN to process the entire image and a region of interest (RoI) pooling layer to extract features from the proposed regions[37]. This method is faster than R-CNN but still relies on selective search for region proposals, which can be computationally expensive.
3. Faster R-CNN: This approach replaces the selective search step with a Region Proposal Network (RPN) to generate region proposals more efficiently[35]. Faster R-CNN achieves real-time object detection performance while maintaining high accuracy.
4. YOLO (You Only Look Once): YOLO divides an input image into a grid and predicts bounding boxes and class probabilities for each grid cell in a single forward pass of the network[35]. This method is known for its speed and real-time performance but may struggle with detecting small objects or objects in crowded scenes[38].
5. SSD (Single Shot MultiBox Detector): Similar to YOLO, SSD predicts multiple bounding boxes and class probabilities in a single forward pass of the network[37]. It uses a series of convolutional layers at different scales to improve the detection of objects of varying sizes.

6. Mask R-CNN: An extension of Faster R-CNN, Mask R-CNN adds a branch for predicting object masks, enabling instance segmentation[38]. This method is suitable for detecting objects and their precise boundaries in images.

In the context of aquaculture, object detection techniques can be employed for tasks such as fish counting, egg counting and classification, or identifying fish species. However, the specific challenges in aquaculture, such as varying lighting conditions, water turbidity, and fish or egg occlusion, may affect the performance of these algorithms. To address these challenges, custom training data and fine-tuning of the detection models may be required to achieve satisfactory performance[39][40][41].

In addition to the aforementioned object detection algorithms, there have been several advancements in the field that further improve the performance and applicability of these methods:

- YOLOv2, YOLOv3, and YOLOv4: These are successive improvements of the original YOLO algorithm, each iteration offering enhancements in terms of accuracy, speed, and the ability to detect objects of varying sizes[42]. These newer versions are particularly useful for real-time applications that demand high computational efficiency.
- Feature Pyramid Networks (FPNs): FPNs are used in combination with object detection methods like Faster R-CNN and SSD to improve the detection of objects at different scales[39]. By creating a multi-scale feature hierarchy, FPNs enable the detection of both small and large objects with high accuracy.
- Anchor-free object detection algorithms: Methods such as CenterNet and CornerNet do not rely on predefined anchor boxes for predicting object locations. Instead, they predict the object center or corners, respectively, which can lead to more accurate bounding boxes and potentially fewer false positives[41].
- EfficientDet: This is a family of efficient object detection models that scale across different levels of accuracy and computational requirements. EfficientDet models are designed to provide a balance between speed and accuracy, making them suitable for various applications, including those in aquaculture[43].

When applying object detection techniques in aquaculture, it is crucial to consider factors such as the quality and diversity of training data, as well as the need for domain-specific adaptations. For instance, incorporating data augmentation techniques like random rotation, flipping, and changes in brightness can help the model generalize better to varying conditions found in aquatic

environments[40].

Moreover, it is essential to evaluate the performance of the chosen object detection algorithm using relevant metrics, such as precision, recall, and F1-score. Depending on the specific task, it may also be necessary to account for real-time performance requirements or constraints on computational resources, which can influence the selection of an appropriate object detection method[39][36].

1.5 Metrics used to evaluate model performance

These metrics include accuracy, precision, recall, Intersection over Union (IoU), and mean Average Precision[44], [45].

- Precision: Precision is the proportion of true positive detections among all predicted positive detections. High precision indicates that the model has a low rate of false positives, which is important to avoid misidentifying objects or overestimating the number of fish eggs[46][47].
- Recall: Recall, also known as sensitivity or true positive rate, is the proportion of true positive detections among all actual positive instances. High recall means that the model can identify most of the actual objects of interest[46], [48].
- F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure between the two. An F1-score reaches its best value at 100% (perfect precision and recall) and worst at 0%[46], [48].
- Intersection over Union (IoU): IoU is a measure of the overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the area of the intersection divided by the area of the union of the two bounding boxes. IoU is commonly used as a threshold value to determine whether a predicted bounding box is considered a true positive or a false positive. IoU helps assess the accuracy of the model in localizing fish eggs within images[48], [49].
- mean Average Precision (mAP): mAP is a comprehensive metric that combines both precision and recall across all classes and is often used as the primary metric for evaluating object detection models. It calculates the average precision (AP) for each class and then computes the mean over all classes. The AP for each class is calculated by taking the area under the Precision-Recall curve. mAP effectively encapsulates the model's

ability to correctly identify and localize multiple object classes with varying degrees of overlap and scales. It is particularly useful in for evaluating how well the model can distinguish between different types of fish eggs, thereby providing a complete assessment of the model's performance[44]–[46].

These metrics are essential for evaluating the performance of object detection models, as they provide a comprehensive understanding of a model's ability to accurately identify and localize objects of interest. Our main focus is the mAP since it encapsulates all the other metrics.

1.6 Feature Pyramid Networks (FPNs) and their importance for the case study

Feature Pyramid Networks (FPNs) have emerged as an important architecture for object detection, particularly when it comes to detecting objects of varying sizes and scales[50]. For our specific implementation of tilapia egg detection and classification, the FPNs play a crucial role in improving the performance of object detection models by providing a richer representation of the hierarchical nature of features at different scales.

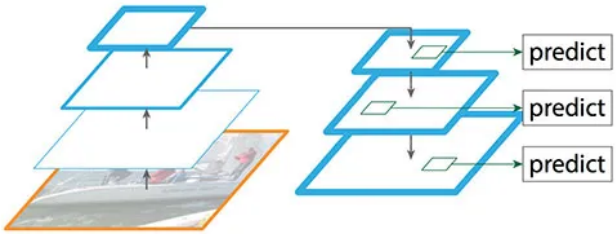


Figure 2: Data flow shaped as a pyramid of images in the feature pyramid network extractor. (Source [51]

FPNs are designed to address the limitations of previous object detection architectures by fusing low-resolution, semantically strong features with high-resolution, semantically weak features in a top-down pathway[50]. The FPN architecture consists of a bottom-up pathway that processes the input image through a series of convolutional layers, generating feature maps at multiple levels, and a top-down pathway that upsamples the coarser-resolution features and combines them with the corresponding bottom-up features using lateral connections (Figure 2)[50]. This combination of features enhances the ability of the model to detect objects at various scales, making FPNs particularly suitable for tasks like tilapia egg detection, where the eggs might be present in different sizes and orientations.

The advantages of FPNs over other object detection architectures are numerous. First, FPNs can effectively detect objects across a wide range of scales, unlike single-scale detectors, which often struggle to detect smaller objects[50]. Second, FPNs make use of a single, unified network for both region proposal and object detection tasks, which streamlines the detection pipeline and reduces the computational overhead. Lastly, FPNs can be easily integrated with existing object detection frameworks, such as Faster R-CNN[50], thereby improving their performance in handling multi-scale object detection tasks.

1.7 Residual Networks

Residual Networks, or ResNets, are a type of deep learning model that have significantly impacted the field of computer vision. They were introduced by Kaiming He et al. in their seminal paper "Deep Residual Learning for Image Recognition" in 2015[52][53].

Structure of ResNet

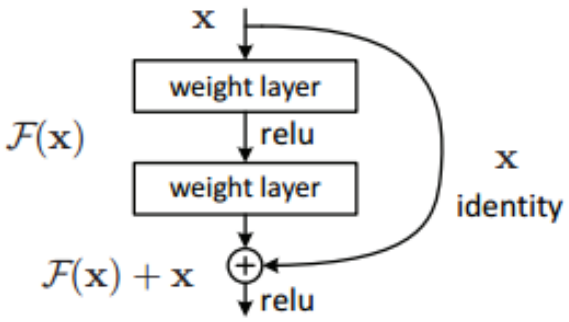


Figure 3: The figure shows the structure of a residual block in ResNet. The input x is added to the output of the stacked layers F(x) to form the final output y = F(x) + x.

The core idea behind ResNets is the introduction of "shortcut connections" or "skip connections", which allow the gradient to be directly backpropagated to earlier layers[52]. These connections essentially perform identity mapping, and their outputs are added to the outputs of the stacked layers[53]. This is illustrated in Figure 3.

In our implementation, these shortcut connections can be integrated into the model to improve its learning capability. This can potentially enhance the accuracy of fish egg classification and counting.

Advantages of ResNet

ResNets have several advantages. Firstly, they allow the construction of deeper networks without the problem of vanishing gradients. This is a significant advantage as deeper networks can represent more complex functions and are generally more powerful[52].

Secondly, ResNets are easier to optimize than traditional networks. This is because the identity mappings in their skip connections simplify the propagation of gradients during backpropagation[52].

These advantages can lead to a more accurate and efficient object detection model. This could result in more reliable information to fish breeders and hatcheries.

Variations of ResNet

There are several variations of ResNet, including ResNet50, ResNet101, and ResNet152. These numbers denote the depths of the networks, i.e., the number of layers in them. The deeper the network, the more complex functions it can represent, but it also becomes more computationally expensive. The specific differences between these variants lie in the number of layers in their architectures[52].

Application of ResNet in Computer Vision

ResNet has been widely used in various computer vision tasks. Due to its deep representations, it has achieved a 28% relative improvement on the COCO object detection dataset[52]. It has also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation[52]. These were important factors in choosing the ResNet model to implement in our application.

1.8 Faster R-CNN

Faster R-CNN with Feature Pyramid Networks (FPN) is a powerful architecture used for object detection tasks, as illustrated in Figure 5. It builds upon the Faster R-CNN framework by incorporating a top-down architecture with lateral connections for building high-level semantic feature maps at all scales[54].

Components of Faster R-CNN

Faster R-CNN is a state-of-the-art object detection network that introduces a Region Proposal Network (RPN) to hypothesize object locations**ren2015faster**. The RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position**ren2015faster**. It shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals**ren2015faster**.

The use of anchor boxes is another key component of Faster R-CNN, as depicted in Figure 4. Anchor boxes are pre-defined boxes of different scales and aspect ratios that are used as reference points for predicting object locations**ren2015faster**. They help the model to detect objects of various sizes and shapes.

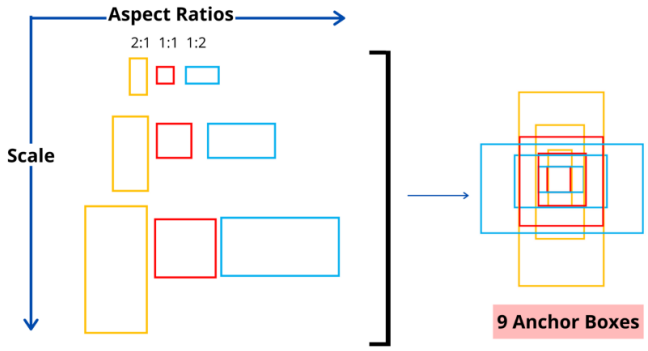


Figure 4: Anchor boxes of the RPN. (Source [55])

Role of FPN

Feature Pyramid Networks (FPN) improve the performance of Faster R-CNN by providing a top-down architecture with lateral connections for object detection[54]. The inherent multi-scale, pyramidal hierarchy of deep convolutional networks is exploited to construct feature pyramids with marginal extra cost[54]. This architecture shows significant improvement as a generic feature extractor in several applications[54].

ResNet as the Backbone

ResNet is often used as the backbone for Faster R-CNN with FPN due to its powerful feature extraction capabilities[52]. It allows the construction of deeper networks without the problem of vanishing gradients, making it a suitable choice for complex tasks like object detection[52].

The use of ResNet in Faster R-CNN with FPN is significant because it provides the model with deep representations that are of central importance for many visual recognition tasks[52]. It also makes the model easier to optimize and more powerful in representing complex functions[52].

Table 2: Comparison of object detection performance using Faster R-CNN with and without FPN on COCO dataset[47]

Model	Backbone	AP	AP50	AP75	APS	APM	APL
Faster R-CNN	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2

Considering all this, Faster R-CNN with FPN is a powerful and practical solution to multi-scale object detection. Its unique architecture and components, including the RPN, anchor boxes, FPN, and the use of ResNet as the backbone, contribute to its state-of-the-art performance. This is further evidenced by Table 2, which compares the object detection performance using Faster R-CNN with and without FPN on the COCO dataset[47].

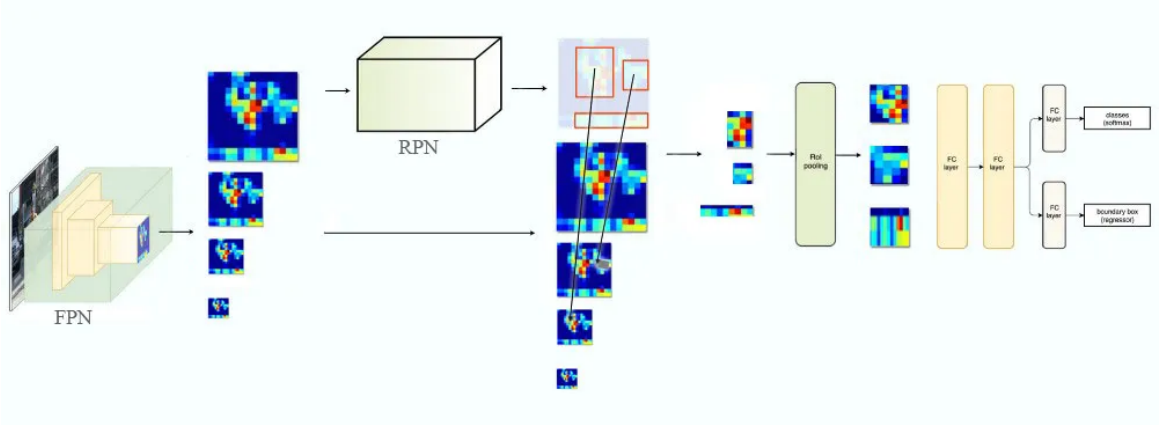


Figure 5: Faster R-CNN architecture with FPN and RPN. (Source [51])

1.9 Mask R-CNN

Mask R-CNN, an extension of Faster R-CNN, has gained attention for its capability to perform both object detection and instance segmentation with high accuracy[56]. In our casewhere precision is paramountMask R-CNN is a promising candidate[56].

Addressing Overlaps and Crowding

One of the limitations of Faster R-CNN is its propensity to generate overlapping bounding boxes[56]. Mask R-CNN addresses this by incorporating an additional mask prediction branch. This feature is particularly beneficial in the aquaculture domain generally, and in our case specifically, where images are often crowded, leading to the overlapping of tilapia eggs[57].

Segmentation

Mask R-CNN leverages the power of instance segmentation, not just bounding boxes, to isolate individual objects in an image. This is vital in scenarios where eggs are closely packed, nearly indistinguishable using traditional bounding boxes[57]. By achieving instance-level segmentation, Mask R-CNN provides a higher degree of localization, thereby increasing the accuracy of both detection and classification[56].

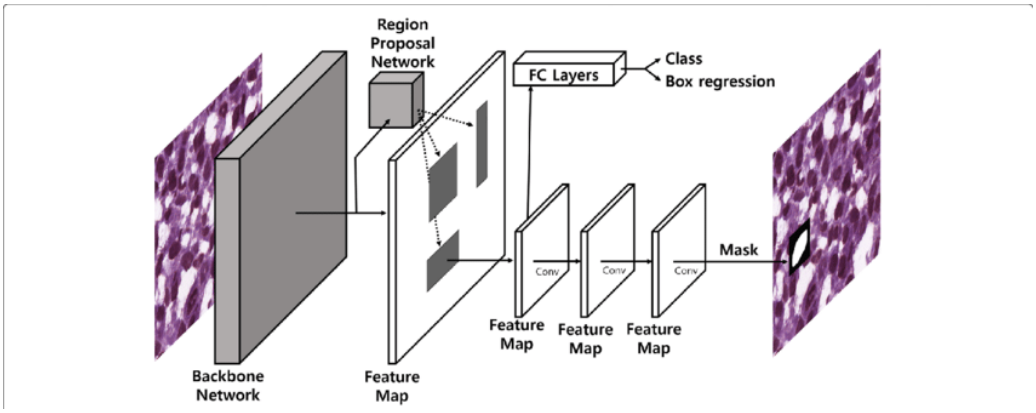


Figure 6: Mask R-CNN architecture. (Source [58])

Architectural Improvements

Mask R-CNN employs a Region of Interest Alignment (RoIAlign) layer, which replaces the RoIPool layer used in Faster R-CNN[59]. It also incorporates the Feature Pyramid Network (FPN) for better feature extraction across multiple scales, similar to Faster R-CNN[59]. The architecture uses ResNet as the backbone, making it compatible with the existing system and leveraging deep representations for enhanced performance[57].

The primary metric for evaluating Mask R-CNN in this context is the mean Average Precision (mAP), a reliable indicator of the model's ability to precisely detect and classify objects.

Choice Motivation

The potential applications of Mask R-CNN extend beyond aquaculture. For instance, its use in detecting features in satellite images demonstrates the algorithm's capability to handle small objects requiring high precision, akin to tilapia eggs[60].

Conclusion

Chapter has provided a comprehensive overview to ensure the research can be understood. We've explored the specific domain of Tilapia farming, identifying existing gaps and challenges that motivate this study. By surveying the current landscape of computer vision and AI applications in aquaculture, we've selected the technologies to be employed in this work. Furthermore, we've introduced and discussed the key algorithms, architectures, and metrics that form the underpinning of the methodologies and evaluations to follow.

This chapter serves as a primer, equipping readers with the necessary background to fully comprehend the complexities and nuances of the methodologies, results, and discussions that are to come in subsequent chapters.

2. Chapter 2: Methodology

Introduction

The methodology chapter is the blueprint of this research project, outlining the architectural design, data handling, model development, and application deployment strategies. The chapter is organized into distinct sections, each focusing on a specific aspect of the research methodology. It begins by discussing the architecture and design considerations for smartphone-based data acquisition, followed by an in-depth exploration of dataset collection, annotation, augmentation, analysis, and dynamic cleansing. Next, it delves into the model's developmental stages, covering everything from its basic configuration and hyperparameter tuning to the implementation of a multi-level transfer learning pipeline and how we solved the challenges posed by the various limitations. The chapter concludes with a detailed overview of mobile application development, examining both server-side and client-side considerations.

2.1 Smartphone-based Data Acquisition System Architecture and Design Considerations

The initial decision to utilize smartphones as the primary data acquisition device in this study was driven by their practicality, feasibility, and accessibility, as well as an inspiration from a related work named "Counting tilapia larvae using images captured by smartphones" by Celso Soares Costa and al.[61]. Smartphones are ubiquitous, cost-effective, and integrate essential components such as CPU, storage, RAM, camera, and screen, eliminating the need for additional modules[62]. This gives them a distinct advantage over alternatives like Jetson Nano or Raspberry Pis[62]. Furthermore, modern smartphones are equipped with high-quality cameras and possess decent processing capabilities, making them suitable for related tasks[63].

However, the limitations of smartphone use, particularly in the context of running deep learning models, necessitated a reconsideration of our system architecture. Despite the relative ease of application development on smartphones, the performance of deep learning models can be

constrained by the computational resources available on these devices[64]. To overcome this limitation, we initially considered optimizing the model for mobile deployment using techniques such as model pruning, quantization, and knowledge distillation. However, we also had to consider power consumption, as running deep learning models on smartphones can be energy-intensive, leading to rapid battery depletion[64].

Upon further reflection, we decided to expand the system architecture to handle the inference on a server. This decision was guided by the trade-off between local processing on the mobile application and processing on the server. Local processing would require the model to be included in the mobile application, significantly increasing the size of the app and potentially slowing down the inference time due to hardware limitations. On the other hand, server-side processing necessitates an internet connection, and the communication time and waiting for the response will depend on the user's upload and download speeds. Despite these considerations, the benefits of server processing, including reduced app size and potentially faster inference time, outweighed the drawbacks.

While smartphones remain our primary data acquisition device due to their wide availability, integrated functionalities, high-quality cameras, and ease of application development, we have shifted the computational demands of deep learning model inference to a server. This decision allows us to mitigate the limitations of smartphones, particularly in terms of computational resources and power consumption, while still leveraging their advantages.

2.2 Dataset

In this section we will delve into the most important part of the research which is the dataset. We are going to discuss the data collection and the challenges faced, the annotations process, the different techniques of augmentation, as well as a novel approach of cleansing the dataset through statistical analysis.

2.2.1 Dataset Collection

Collecting a comprehensive and diverse dataset was a multi-faceted endeavour fraught with challenges. Below are the various stages and details of the dataset collection:

Public Datasets

Despite several days of research, there was no public dataset about tilapia fish eggs. We had to resort to other alternatives.

Synthetic Dataset

Our first approach was to create a synthetic dataset using Blender (Figure 7), as well as painting small bearing balls to mimic Tilapia eggs (Figure 8). This proved unsuccessful due to the complexity of the task and the potential to skew the data [65]. Not to mention that there are specific features in real tilapia fish eggs that had to be replicated (Table 1).

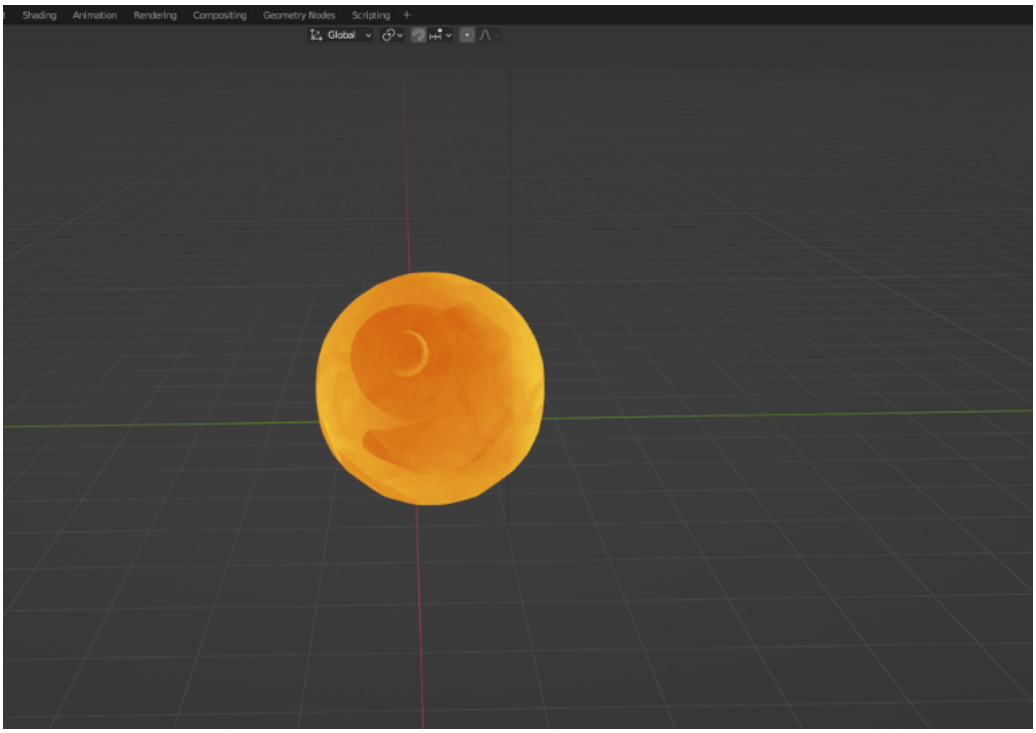


Figure 7: Our abandoned attempt at 3D modelling the fish eggs in Blender.

Dataset version: DS-April23

We then collaborated with Aqua Pro LLC, an Algerian fish breeding and hatchery company. This partnership yielded 17 high-quality images captured with an OPPO A54s smartphone with a camera resolution of 9.4 megapixels (Figure 9). Each image had a resolution of 4080 x 2296 pixels and collectively contained over 10,000 fish eggs. However, these images had limitations such as similar angles, zoom levels, lighting, and distributions, making them very crowded. We re-contacted them for better images with these specific instructions for improvement: "Take a

Figure 8: Our abandoned attempt after painting small bearing balls in white and gold.

video of the eggs inside the box. While shooting the video, every few seconds change the camera angle to up, down, left and right. Try to go slow and avoid fast movements. While shooting the video, use the camera flashlight (turn it on and change angles, then turn it off and change angles). Or turn the light of the room on and off and change the angle for each setting. While you are shooting the video, each time, shake the egg box or change their locations manually so that the eggs change places in the video. The duration of the video would preferably be between 2 and 4 minutes."[66].

Dataset version: DS-May23

After receiving no response for improved images from Aqua Pro LLC, we reached out to the local director of the Direction of Fisheries and Marine Resources in Laghouat. He provided samples of two stages, cleavage and blastula-gastrula. We took videos using a Samsung Galaxy M30s at Full-HD resolution and 30 fps, employing two different backgrounds (white and red) and varying lighting and zoom settings. Images were extracted at a 23 fps rate, resulting in 19,746 images. A Python script was developed to combine the videos, extract images, and remove duplicates and similar images using a simple ResNet152 feature extractor, as well as deleting blurry images which were identified using the variance of Laplacian focus measure[67]. After processing, we obtained 23 images containing over 3,000 tilapia fish eggs (Figure 10).

Figure 9: A sample of an image captured from the DS-April23 batch.

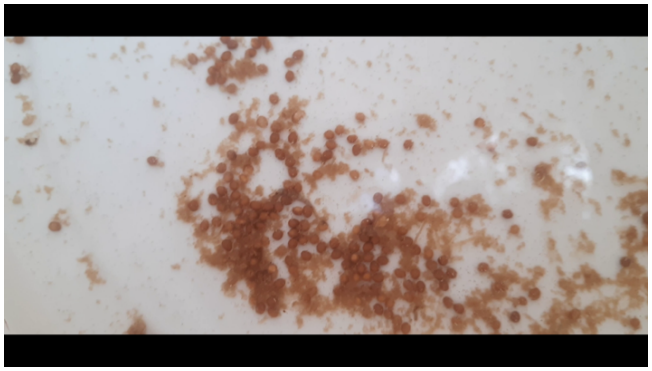


Figure 10: A sample of an image captured from the DS-May23 batch.

Dataset version: DS-August23

In pursuit of a more diverse dataset, we found an opportunity to collect fresh eggs directly from tilapia fish mouths. The majority were in the cleavage stage (~90%) and the rest in the blastula-gastrula stage (~10%). Multiple smartphone cameras were used, including Samsung Galaxy M30s, Samsung Galaxy J7 Pro, and Xiaomi Redmi 9 Pro. This time we took images instead of videos, varying in lighting, zoom, and angle settings. Various backgrounds were used: black and white ceramic plates (to introduce background reflection), black and white cloth fabrics (to mute reflections and introduce textured background noise), and a black plastic

tray. The eggs were sometimes in water to introduce water reflections. The image resolutions varied from 3000x4000 to 6000x8000 pixels, and some images were intentionally defocused. This effort resulted in 56 images containing 400 tilapia fish eggs (Figure 11).

Figure 11: A sample of an image captured from the DS-August23 batch.

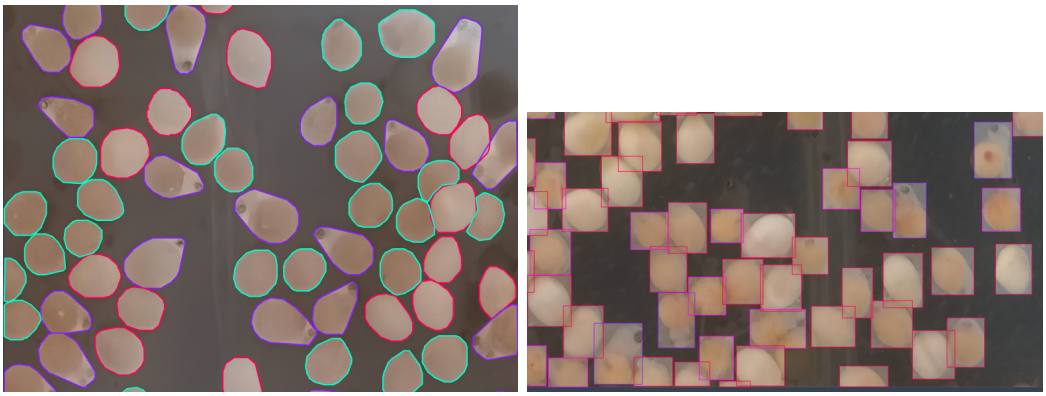
Challenges

Dataset collection was a difficult task due to the commercial focus and limited access to the sourcing locations. The tilapia fish reproductive cycle, ranging from 29 to 45 days, added time constraints[22]. Not to mention that extracting eggs from the tilapia fish mouths required both time and experience to ensure the eggs remained viable[22].

2.2.2 Dataset annotation

Following the data collection, we used Roboflow Annotate, a web-based AI-assisted annotation tool, to label the eggs[68]. Roboflow Annotate gives us the ability to annotate using both bounding boxes (Figure 12(b)) and polygons (Figure 12(a)), which we both experimented with.

As part of our study, the eggs were systematically sorted into three distinct groups according to their respective development phases (as seen in Table 1). The first group, labelled as 'cleavage,' corresponds to the initial day and is characterized by a light yellow hue (Figure 1(a)). The second grouping, spanning days two to four, embodies the intermediate transition from the blastula to the gastrula stages and exhibits a uniform shape with an orange-to-brown colouration (Figure 1(b)). Finally, the third category, indicative of the organogenesis stage on the fifth day, is easily discernable by the visibility of the eye and sometimes the appearance of a transparent tail (Figure 1(c))[22].



((a)) Polygonal annotations

((b)) Rectangular annotations (bounding boxes)

Figure 12: The polygonal and rectangular annotation types used.

2.2.3 Dataset Augmentation

Data augmentation serves as the third pillar in our dataset preparation strategy. The technique involves artificially expanding the dataset through various transformationsranging from simple geometric shifts to complex photometric adjustments[69].

We experimented with two tools for augmentations and used them separately, both with random values for each technique:

Roboflow Annotate Augmentations

We employed Roboflow Annotate for our initial set of augmentations. Table 3 summarizes the techniques used, along with their impact on our specific case.

Python Library: imgaug

In addition to Roboflow, we used the imgaug Python library for a more diverse set of augmentations[70]. The techniques and their specifics are listed in Table 4.

Table 3: Augmentation Techniques Using Roboflow Annotate

Technique	Parameter Values	Description and Importance
Flip	Horizontal, Vertical	Mirrors the image; adds variance
Rotation	90 CW, CCW, Upside Down	Rotational diversity; trains model for different orientations
	-15 to +15	Adds slight rotation; robustness against angled shots
Shear	15 Horizontal, 15 Vertical	Skews image; ensures model generalization
Color	Hue -5 to +5	Alters color; simulates different lighting conditions
	Saturation -25% to +25%	Alters color intensity; simulates different lighting conditions
	Brightness -21% to +21%	Adjusts brightness; robustness against varied lighting
	Exposure -7% to +7%	Alters exposure; trains model for different camera settings
Blur	Up to 1.5px	Adds blur; simulates focus variations
Mosaic	Applied	Combines multiple images; adds complexity
Bounding Box	Rotation -25 to +25	Rotates bounding box; robustness against object orientation
	Brightness -25% to +25%	Alters bounding box brightness; adds complexity
	Blur Up to 0.75px	Adds blur to bounding box; trains model for focus variations

Table 4: Augmentation Techniques Using imgaug

Category	Technique	Values	Description and Importance
Color	Add	-2, 2	Minor brightness adjustments; simulates varied lighting
	Multiply	0.9, 1.1	Alters brightness; simulates different lighting conditions
	Hue and Saturation	-3, 3	Modifies hue and saturation; adds color diversity

Continued on next page

Table 4 – continued from previous page

Category	Technique	Values	Description and Importance
	Gamma Contrast	0.9, 1.1	Alters contrast; robustness against varying contrast
Noise	JPEG Compression	70, 90	Introduces compression noise; robustness against quality variations
	Salt And Pepper	0.01, 0.1	Adds salt-and-pepper noise; robustness against sensor noise
	Coarse Dropout	0.01, size percent 0.01, 0.1	Adds dropout; simulates occlusions
	Blend Alpha-Simplex Noise	0.5	Adds noise; increases model robustness
Blur	Motion Blur	3, 7	Simulates motion; trains model for slight movements
	GaussianBlur	0.5, 1.0	Adds blur; simulates focus variations
Edge	Sharpen	0.5, 0.8	Enhances edges; aids in better feature extraction
	Emboss	0.1, 0.6	Adds embossing; aids in feature detection
	Edge Detect	0.1, 0.5	Highlights edges; assists in feature extraction
Flip	Fliplr, Flipud	0.6	Flips the image; adds variance
	Affine	rotate -90, 90, shear 10, 10	Rotates and shears; trains for different orientations

Continued on next page

Table 4 – continued from previous page

Category	Technique	Values	Description and Importance
Weather	Clouds	–	Adds clouds; simulates outdoor conditions
	Fog	–	Adds fog; simulates specular reflection

The techniques and values for both the Roboflow Annotate and imgaug augmenters were chosen carefully to prepare the model for a wide array of real-world scenarios and imaging conditions.

2.2.4 Dataset Analysis

Analyzing the dataset, specifically the annotations, is a pivotal step in any machine learning project.

Importance of Quality Annotations

Quality annotations are the bedrock of supervised learning models. Inaccurate or inconsistent annotations can introduce noise into the dataset, leading to poor model performance and misleading results. These annotations serve as the ground truth during model training and validation, making their quality crucial for effective learning and generalization[71].

Significance of Dataset Annotation Analysis

Beyond ensuring quality, it is essential to conduct a thorough analysis of the dataset's annotations. This analysis serves multiple functions: it can reveal patterns, imbalances, or outliers in the dataset that may necessitate corrective action. It also provides valuable insights for optimizing model parameters, making the analysis an indispensable step in the machine-learning pipeline[72].

Metrics and Statistics and Their Importance

- **Number of Annotations**: Indicates the volume of data available for each class.

- **Instances Per Class**: Helps in understanding class imbalance, which can be critical for model performance.
- **Images' Resolutions**: Affects the computational resources required for model training and inference.
- **Bounding Box Statistics**: Helps in understanding the sizes and aspect ratios of objects, which is crucial for anchor box selection in object detection models.
- **Polygons Statistics**: Gives information about the number of vertices for the polygons and segmentations.
- **Histogram Data**: Provides insights into the colour distribution within the dataset, which can be useful for data augmentation strategies.

2.2.5 Dynamic Dataset Cleansing

The analysis of the dataset gives us a variety of insights, in which we can interpret each one and perform an appropriate action, whether it is cleansing, optimization, or guidance.

Number of Annotations

The number of annotations is a direct reflection of the dataset's size. In machine learning, particularly deep learning, the size of the dataset often directly correlates with the model's ability to generalize to unseen data[73].

Interpretation: A low number of annotations (often less than a few hundred) can lead to overfitting, where the model memorizes the training data but performs poorly on new data. Therefore, a high number of annotations can be computationally expensive and time-consuming to process[74].

Remedies: The low and high numbers are determined as thresholds that are set manually.

- For low numbers, data augmentation can artificially increase the size of the dataset.
- For high numbers, random subsampling can be employed to reduce computational load without significantly impacting model performance.

Instances Per Class

This metric reveals any class imbalance in the dataset, which is crucial because a skewed distribution can cause the model to be biased towards the majority class[75].

Interpretation: A significant disparity in the number of instances per class should be addressed to prevent model bias.

Remedies: Implementing class weights during model training can also mitigate the effects of class imbalance. We didn't consider that because we took another approach in which we can gain artificially new data. We consider upsampling the minority class to balance the dataset. The standard mechanism is replicating the annotations would help balance the classes but can introduce the potential for overfitting since the instances would be increased in the same area pixel-wise. Instead, upon each instance upsampling of the same image we copy the image and rotate or flip it at random and copy only the minority class instances. This way we mitigate the class imbalance without the risk of overfitting.

Images' Resolutions

The resolution of images in the dataset impacts both computational cost and model performance.

Interpretation: High-resolution images can offer better feature extraction but are computationally expensive. Low-resolution images are faster to process but may lack essential details[76].

Remedies: We calculate a fixed image dimension in which all the images will be resized. The dimensions are determined to be as small as possible without losing details, so we use the bounding boxes statistics to check the instance with the minimum area and downscale its image just before it gets smaller than the area covered by 16x16 pixels (a predefined threshold, we found that to be the optimum value for our implementation). The new image dimensions after the downscaling will be applied to all the other images.

Bounding Box Statistics

These statistics provide insights into the geometric characteristics of the objects in the dataset.

Interpretation: The sizes and aspect ratios of bounding boxes can inform the selection of anchor boxes, which are essential for the RPN component of the model's architecture[77].

Remedies:

- Outliers can be removed or adjusted to fit within acceptable bounds. To determine these acceptable bounds we developed an algorithm that utilizes statistical methods to iteratively refine a dataset by identifying and removing outliers in bounding box sizes and aspect ratios. Specifically, it uses Z-score and Median Absolute Deviation (MAD) methods to calculate bounds for outlier detection[78]. The iterative process continues until the standard deviation of the aspect ratios stabilizes, ensuring a more robust dataset for model training.
- In the presence of incorrect annotation data, which is determined by the inconsistency across all the annotation instances, bounding boxes could be offset or misplaced and to fix those annotations we developed an algorithm to balance the distribution of bounding box sizes. It calculates a "compromise count" through a formula that uses the mean, standard deviation, and median of the counts of different bounding box sizes. The algorithm sequentially does two things: downsample sizes and upsample sizes. Downsampling sizes reduce the frequency of overly common bounding box sizes by adding noise to their dimensions (a few pixels at random). Upsampling sizes, on the other hand, increase the frequency of rare bounding box sizes by duplicating them, either in place or by flipping the image (introducing new artificial data). Both functions aim to bring the count of each bounding box size closer to the compromise count, thereby achieving a balanced dataset.
- Custom anchor boxes can be defined based on these statistics. We developed an algorithm that employs K-means clustering to determine optimal anchor sizes and aspect ratios. By analyzing the bounding boxes in the dataset, K-means clustering groups similar sizes and aspect ratios together. The centres of these clusters serve as the optimal values for anchor sizes and ratios, thereby enhancing the model's ability to accurately detect objects across varying dimensions. This data-driven approach ensures that the model's anchors are tailored to the specific characteristics of the dataset.

In any dataset, the presence of outliers can be attributed to different factors such as human error, and data augmentation techniques. Figure 13 shows what is considered as outliers as an example for aspect ratios of instances. The human factor may introduce inconsistencies due to subjective interpretation, fatigue, or lack of expertise, leading to anomalous bounding box sizes or aspect ratios. Data augmentation techniques, such as random cropping, rotation, or flipping, can inadvertently generate outliers by creating bounding boxes that are disproportionately large or small compared to the original distribution. These distortions can introduce outliers

that deviate significantly from the expected dimensions, thereby affecting the robustness and generalizability of models trained on such datasets. Consequently, the presence of outliers necessitates rigorous pre-processing steps to ensure data integrity and model performance.

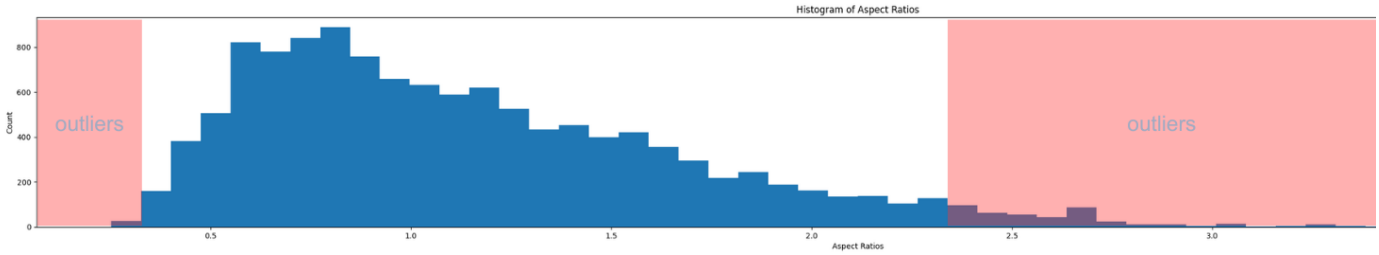


Figure 13: Plotting the histogram of aspect ratios in all existing instances in an annotations file and their count, highlighting the outliers.

Polygon Statistics

Understanding the complexity of annotated polygons can be essential for tasks like semantic segmentation.

Interpretation: A high number of vertices might indicate overly complex annotations, whereas too few could signify a loss of important details.

Remedies: We utilized the Ramer-Douglas-Peucker algorithm to reduce the number of vertices in complex polygons[79].

Histogram Data

This provides insights into the colour distribution of the dataset and is critical for tasks requiring colour-based features.

Interpretation: An uneven colour distribution might indicate a lack of diversity in the dataset, affecting the model's ability to generalize.

Remedies: Color augmentation techniques can enrich the dataset. We do that by scaling the saturation after we load the image in the HSV colour space. That may increase the colour variations, therefore, we use normalization strategies such as histogram equalization which brings the value channel from the HSV colour space to the average of all the other images[80].

2.3 Model development and training

The foundation of any machine learning application lies in the architecture and configuration of its model. In section 2.3.1, we delve into the intricacies of the model configuration that has been employed in our project. The choice of architecture, initialization methods, and configuration details form the highlights of this section. Understanding the model configuration is crucial as it sets the stage for subsequent processes such as hyperparameter tuning. We also introduce a novel approach based on transfer learning. This section also provides the groundwork for addressing the challenges related to environmental variables and hardware limitations.

2.3.1 Model Configuration

To develop a machine learning model with a high level of reliability and robustness for the task of identifying and classifying developmental stages of eggs in tilapia fish, we configured the model's components to be custom to our task. This approach spanned an array of model architectures and optimization techniques, each meticulously chosen to address the unique challenges inherent to our specific application domain. The foundational step in this journey was a rigorous evaluation of different backbone architectures for the model. We focused on three variants of the ResNet architecture: ResNet50, ResNet101, and ResNet152[81]. This choice was underpinned by the architectures' well-documented efficacy in a broad range of image classification tasks, from object detection to semantic segmentation.

To further bolster the feature extraction capabilities of these selected backbones, we integrated a Feature Pyramid Network (FPN) extractor[54]. This architectural enhancement was designed to improve the model's ability to recognize features across multiple scales. Specifically, we fine-tuned three layers from the final block of the FPN to establish top-down connections, thereby enabling a harmonious fusion of low-level and high-level features. This enriched feature representation proved to be more robust and versatile.

Recognizing the unique challenges posed by our dataset, which includes fish eggs at various developmental stages, we customized the Region Proposal Network (RPN). We introduced specific anchor sizes and aspect ratios that were empirically validated to improve the model's object detection capabilities. To expedite the training process, we initialized our model with pre-trained weights from the ImageNet dataset[82], thereby reducing the computational burden without sacrificing performance.

In our quest for the most suitable loss function, we opted for the Smooth L1 loss function. This choice was motivated by its dual advantages of providing a stable optimization landscape and facilitating faster convergence[83]. This was particularly crucial in our application, where the balance between precision and recall is of importance.

We initially employed Faster R-CNN for object detection but later transitioned to Mask R-CNN to accommodate the polygonal annotations in the later versions of our dataset. The shift was necessitated by Faster R-CNN's limitations in handling segmentation tasks, which Mask R-CNN could manage[56].

For performance evaluation, the dataset was partitioned into an 80%-20% configuration for training and validation sets, respectively. This allowed us to continuously monitor a suite of performance metrics, including training loss, validation loss, average precision, and average recall. These metrics were assessed based on varying Intersection over Union (IoU) thresholds, ranging from 0.50 to 1.00, across different object sizes and scales.

2.3.2 Hyperparamter Tuning

To achieve a balance between the complexity of the model and the efficiency of its training process, we integrated an automated method for hyperparameter tuning. We specifically employed Bayesian optimization, using the Optuna Python library[84]. The choice of the search algorithm was motivated by the need to conserve computational resources while still aiming to maximize the model's mean average precision (mAP). Bayesian optimization has the advantage of being more sample-efficient compared to other methods like grid search or genetic algorithms, thereby reducing the computational burden[85], [86]. To further accelerate the optimization process, we implemented parallel computing by launching multiple instances of the Bayesian optimization algorithm simultaneously.

The search algorithm starts by selecting values for the hyperparameters and starts training the model on a random subset of the dataset. We decided to make 100 iterations of the search with each one training the model for 11 epochs, a number selected to be as minimal as possible while giving enough iterations for the model to optimize itself.

The hyperparameters we focused on were carefully selected to include the most impactful ones: learning rate, which controls the step size in the optimization algorithm; momentum, which helps in overcoming local minima; weight decay for regularization; learning rate scheduler factor and patience, which adaptively adjust the learning rate during training; the number of epochs,

which dictates how many times the learning algorithm will work through the entire training dataset; and the type of optimizer, with options being ADAM and SGD (Stochastic Gradient Descent).

We consciously decided to limit the number of tunable hyperparameters due to time constraints. Adding more hyperparameters would exponentially increase the search space, thereby requiring more computational time and resources. Moreover, we chose not to alter the model's architecture, as the optimal values for its configuration were already statistically inferred as shown in previous sections of our study. This decision was also backed by the understanding that architectural changes could introduce more variables, making it difficult to attribute performance changes to hyperparameter tuning alone. Therefore, our approach was designed to efficiently identify a robust set of hyperparameters that would optimize the model's performance, without necessitating excessive computational power or time.

2.3.3 Multi-Level Transfer Learning Pipeline

In response to the challenges encountered during the initial training phase (both dataset quality and hardware limitations), we reformulated our approach and devised a complex, multi-level transfer learning pipeline. This was done in anticipation of enabling a more efficient and effective learning process. The pipeline uses different samples of the dataset at each level, providing a progressively more challenging learning environment for the model. This strategy includes three samples of the dataset as shown in Table 5:

1. The first sample is the smallest, used to expedite the initial training phase and facilitate faster convergence by using fewer images and having fewer annotations per image. 2. The second sample introduces a few augmentations. This is a slightly larger dataset that allows for more generalization in the model, avoiding overfitting. 3. The third sample of the dataset introduces multiple augmentations, offering the largest and most challenging dataset for the model.

Table 5: Datasets samples and their image count and annotations count per class.

Dataset sample	**Number of images**	**Total number of instances**
Sample 1	50	2295
Sample 2	589	29243
Sample 3	2756	198866

Each of these samples includes annotations in two forms: single class ("egg") and multi-class ("cleavage", "blastula_gastrula", and "organogenesis"). Training on a single class dataset provides a baseline and facilitates faster initial learning. Once the model has learned to identify general egg-like features, multi-class annotations help it distinguish between different developmental stages.

The training pipeline thus consists of four levels:

- Level 1: Training on a single class on dataset sample 1.
- Level 2: Training on a single class on dataset sample 2.
- Level 3: Training on multiple classes on dataset sample 2.
- Level 4: Training on multiple classes on dataset sample 3.

This sequence was carefully chosen to avoid redundancy and ensure efficient utilization of computational resources. It also guides the model's learning in a meaningful trajectory, starting from simple tasks and gradually moving to more complex ones (Figure 14).

The rationale behind this complex training pipeline is rooted in the principles of transfer learning and progressive learning[87]. By starting with a simpler task (single class detection on a smaller dataset), the model can learn basic features and patterns. As the complexity of the task increases (multi-class detection on larger datasets), the model can build upon the learned features, refining and expanding them to handle the increased complexity. This approach not only improves the efficiency of the training process but also enhances the model's ability to generalize and perform well on unseen data.

2.3.4 Solving Environmental Challenges and Hardware Limitations

In this section, we delve into the different challenges we encountered while attempting to train our model, along with the solutions we implemented to overcome each obstacle. Initially, the need for a robust computational environment led us to seek access to the university's high-performance computer (HPC) in the math and computer science lab. Unfortunately, due to various constraints, this was not feasible. However, we were fortunate to secure temporary access to an alternative HPC in the technology lab. This machine was equipped with an Intel Xeon E5-2690 processor featuring 10 cores, a substantial 100GB of RAM, and an Nvidia A5000 graphics card with 14GB of video memory.

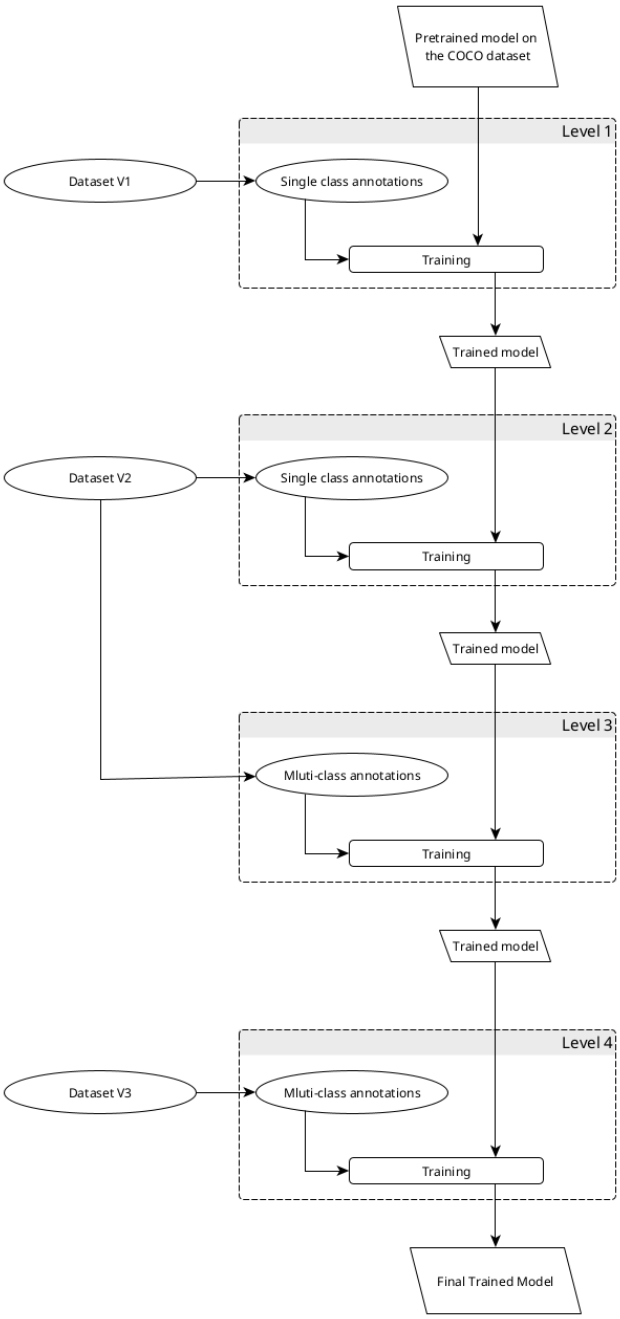


Figure 14: Diagram of the multi-level transfer learning and training pipeline

Power outages

We encountered several frequent power outages that disrupted the training process. To mitigate this, we modified the UEFI settings of the computer's motherboard to enable automatic reboot upon power restoration. We also developed a PowerShell script configured to initiate the training pipeline automatically. This script was set to run as a Windows service, allowing it to operate even when no user was logged into the system, thereby overcoming the login

requirement upon startup that we were instructed not to disable.

Checkpointing

To address the issue of resuming training from the point of interruption, we designed a custom logger. This logger recorded all the major actions in the training process, such as the best and last trained models, Optuna's search progress, current parameters and hyperparameters, as well as performance results. This ensured that the training could resume from the last saved state, eliminating the need to start from scratch.

Monitoring

Monitoring the training process presented another hurdle, given that the HPC was located in a remote lab requiring a long commute. To resolve this, we established a Virtual Private Network (VPN) using the ZeroTier service, enabling us to monitor the training remotely via the Remote Desktop Protocol (RDP)[88]. However, this solution was compromised by the university's unstable managed network, attributed to issues with the DHCP and DNS servers. Our initial attempt to stabilize the connection involved setting a static IP and utilizing a public DNS. That solution worked until the internet connection was not stable and we started getting a very low bandwidth which made connecting via RDP impossible; we installed a dedicated 4G router to ensure a stable and separate, network connection.

Memory constraints

Yet, even with a high-performance computer, hardware limitations persisted. The large dataset and complexity of the model frequently led to 'Out of Memory' errors. To counteract this, we implemented exception-handling mechanisms to catch these errors. Upon triggering, the system would automatically adjust specific parameters, such as reducing the batch size and modifying PyTorch's memory threshold, to enable the training to continue without manual intervention.

Summarizing, the implementation of a comprehensive set of strategies to ensure uninterrupted training ranging from hardware modifications and network adjustments to custom logging and adaptive error handling, all aimed at creating a resilient training pipeline capable of overcoming all faced challenges.

In conclusion, the model configuration serves as the foundational blueprint for the functioning of our machine-learning application. The choices made in this step have an effect on the effectiveness of hyperparameter tuning, the efficiency of the transfer learning pipeline, and the solution to environmental and hardware challenges. As such, meticulous attention to detail is imperative in the model configuration phase to ensure that the model is both robust and adaptable for future enhancements. The subsequent sections will build upon this foundation, exploring how the model can be fine-tuned, adapted, and scaled effectively.

2.4 Mobile application development

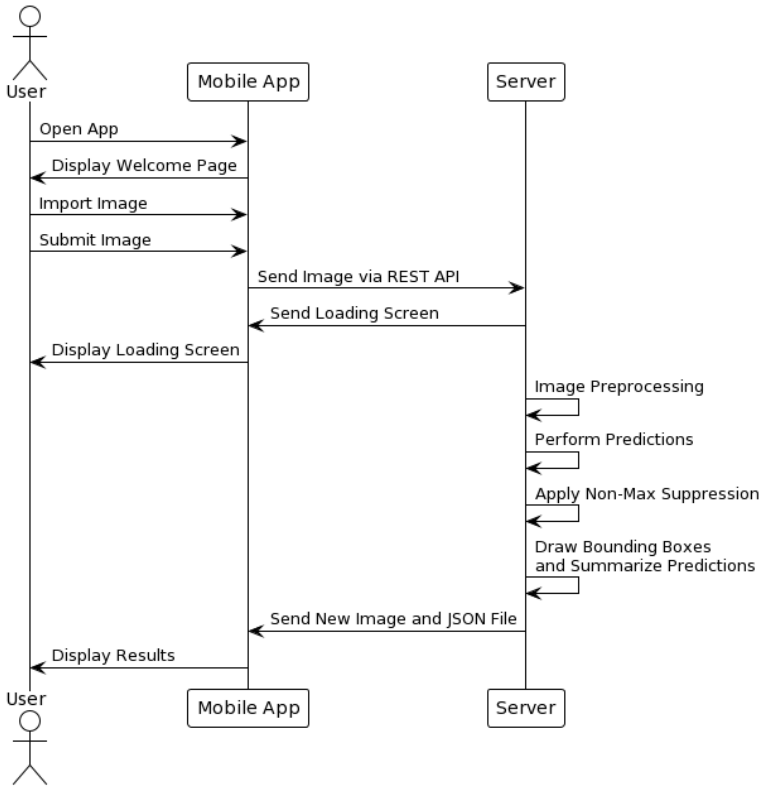


Figure 15: Sequence diagram of the communication between the user, client, and server

The integration of server functions and a mobile application forms the backbone of our system, facilitating seamless interaction between the user and the inference pipeline. This section provides a detailed examination of the server setup, the construction of the inference pipeline, and the interaction between the mobile application and the server. The sequence diagram in Figure 15 describes how the system works.

2.4.1 Server-side of the application

The server, constructed using Flask[89], a micro web framework written in Python, serves as the primary conduit for receiving the image from the mobile application, processing it through the inference pipeline, and returning the results to the client. We have repurposed an old computer to host the server. This computer runs on Ubuntu Server and has an i5-3500 CPU, 8GB of RAM, a 500GB hard drive, and an Nvidia GT 740 graphics card that can handle CUDA. To address the challenge of not having a static public IP address, DynuDNS is employed to implement Dynamic DNS (DDNS). This service regularly updates the IP address associated with the assigned domain name, ensuring uninterrupted accessibility.

To facilitate this, the Demilitarized Zone (DMZ) settings of the router are modified to forward all incoming internet traffic from a specific port directly to the server. Specifically, the local IP address of the server is assigned to the DMZ in the router's settings. This configuration allows external users to access the server while maintaining the security of the internal network. When external users access the domain name, the traffic is routed through the internet to the router's DMZ, which then forwards it to the server. DynuDNS ensures that even if the IP address changes, the DNS record is updated, thereby maintaining a consistent connection[90]. It is important to implement robust security measures, such as firewall configurations and regular updates, to safeguard the server, which we did.

The inference pipeline, a crucial component of the server functions, is structured as follows:

1. Preprocessing: The image undergoes an initial preprocessing phase, where auto contrast and brightness adjustments are applied, along with a sharpness kernel. This step ensures that the image quality is enhanced, providing the model with a clear and well-contrasted image for detection (Figure 16).
2. Prediction: The preprocessed image is then fed into the model for prediction. The model, which operates on the GPU to expedite the inference time, generates predictions for the presence of fish eggs in the image.
3. Non-Maximum Suppression: To eliminate overlapping bounding boxes based on their confidence scores, non-max suppression[91] is applied. This step ensures that the final prediction is as accurate as possible.
4. Drawing and Summarizing: The predictions are used to draw bounding boxes on the image, and the predictions are summarized and packaged into a JSON file. These steps

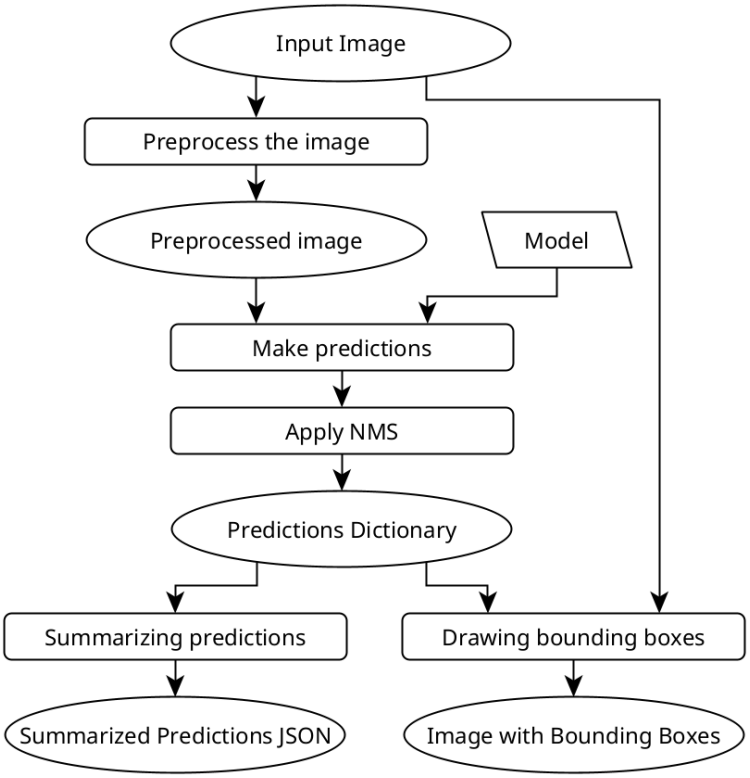


Figure 16: Diagram of the inference pipeline and the server processes

are performed concurrently to optimize the server response time.

The server then packages both the JSON file and the new image with the drawn bounding boxes and sends them back to the client through the API.

2.4.2 Client-side of the application

The mobile application, developed using Flutter[92], a cross-platform framework for building natively compiled applications, provides a user-friendly interface for the user to interact with the system. The user journey is as follows:

1. Welcome Page: The user is first presented with a welcome page that explains how the app works (Figure 23(a)).
2. Image Acquisition: The user can then choose to either import an image or use the camera to take a picture of the fish eggs (Figure 23(b)).
3. Image Submission: The user submits the image, which is sent to the server through a REST API[93].

4. Loading Screen: The user is then presented with a loading screen while the server processes the image (Figure 23(c)).
5. Results: Once the server has processed the image, the user is presented with the returned image and the statistics from the JSON file (Figure 23(d)). The content of the JSON file, including total egg count with an average confidence score, and for each of the three classes, the count and average confidence.
6. Restart: The user has the option to restart the process, which takes them back to the image acquisition page.

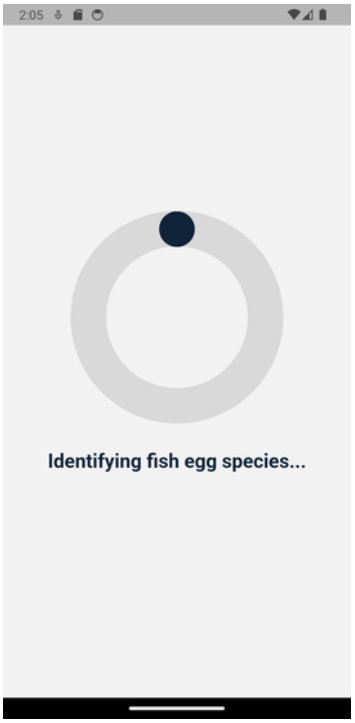
Conclusion

This chapter has outlined the methodologies that form this research. We have discussed the critical architectural choices made for smartphone-based data acquisition and elaborated on the various facets of dataset management. We have also revealed the intricacies of model development, including our innovative approach to multi-level transfer learning. Finally, we discussed the architecture and functionalities of the mobile application, both from the server and client perspectives.

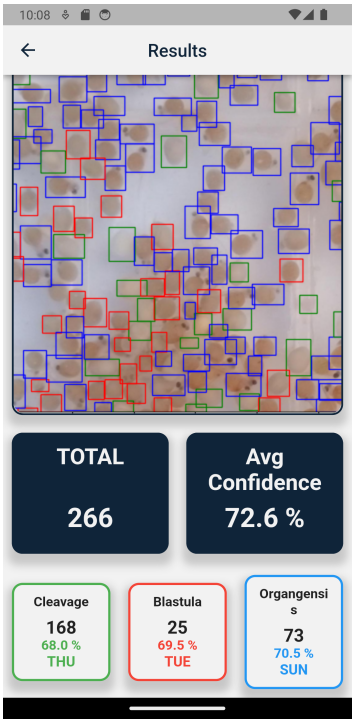
The methodologies provide a comprehensive overview of the multi-disciplinary strategies we've employed. They are designed to offer a balance between theoretical reasoning and practical applicability, setting the stage for the results and discussions that follow in Chapter 2.4.2.

((a)) Welcome page

((b)) Image acquisition and uploading page



((c)) Loading page



((d)) Results page

Figure 17: The mobile application's screen pages

3. Chapter 3: Results and Discussion

Introduction

This chapter aims to present and dissect the findings derived from various experimental setups and methodologies applied in this research. The focus is on evaluating the impact of multiple dataset variations, the efficacy of a novel multi-level transfer learning pipeline, an analysis of performance metrics, and an assessment of system testing and efficiency. We should note that the model's architecture was chosen initially and kept throughout the case study. All the results are the product of training a Mask R-CNN (Faster R-CNN for comparison) with an FPN and a ResNet50 backbone.

3.1 Dataset Variations and Their Impact

Comparing bounding boxes and polygons annotations

Initially, we exclusively used the DS-April23 dataset, which was marked rectangular bounding boxes (Figure 12(b)). This strategy presented several difficulties at the annotation process, particularly the difficulty in dealing with densely packed and overlapping fish eggs. The rectangular shape of these annotations led to considerable visual clutter due to unnecessary overlaps, making the annotation process both visually exhausting.

When it came to training the model using this first dataset, the outcomes were disappointing. Although the training loss was low, which usually indicates effective learning, the mean Average Precision (mAP) was not reasonable. This poor mAP score signified that the model was overfitting; performing well on the training data but poorly on new, unobserved data. Further inspection revealed a high value in both the average classifier loss and average objectness loss, suggesting that the model found it challenging to separate objects.

To address these issues, we shifted our focus to polygonal annotations, requiring us to reannotate the complete DS-April23 dataset (Figure 12(a)), and continued using polygons to annotate DS-May23 and DS-August23. This change also meant that we had to transition to using Mask

R-CNN, a model built to handle segmentations hence the polygonal annotations. Relying on Faster R-CNN would have been counterproductive since it cannot process polygon-based annotations, thereby defeating our initial reason for making the switch.

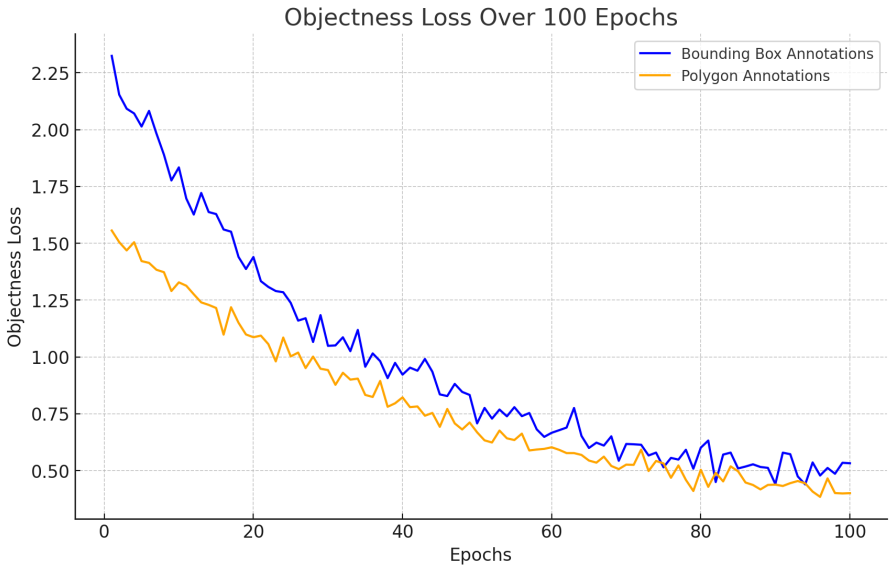


Figure 18: Objectness loss over the first 100 epochs for both instance annotation types.

As shown in Figure 18, the move to polygons led to a slight improvement in the model's learning capability. By cutting down on the excess background noise typically present in bounding box annotations, we observed a meaningful reduction in both objectness and classifier losses. These lower loss metrics indicate that the model is now more adept at distinguishing between object and background pixels, consequently improving its ability to correctly classify objects.

Combining the datasets into one

In an effort to increase the model's resilience to variations, we merged the three datasets together: DS-April23, DS-May23, and DS-August23. This unification served to enrich the training data, resulting in a noticeable improvement in mAP; rising from 74% to 81% after applying dataset cleansing techniques (Figure 19).

The impact of dynamic dataset cleansing

The cleansing mechanisms explained in the previous chapter had a transformative effect on the effectiveness of the model. Comparing the mAP of the combined dataset before and after applying these cleansing techniques in Figure 19. We can notice a clear improvement especially

since the mAP increased from 19% to 81%, highlighting the critical importance of data quality in determining a model's effectiveness.

We should mention that dataset cleansing is not a perfect solution, nor a guaranteed improvement. This was evident upon applying the techniques to the DS-April23 dataset when it was initially annotated by bounding boxes since we did not notice any improvement in mAP since it is the main metric to measure a model for both detection and classification (Figure 19).

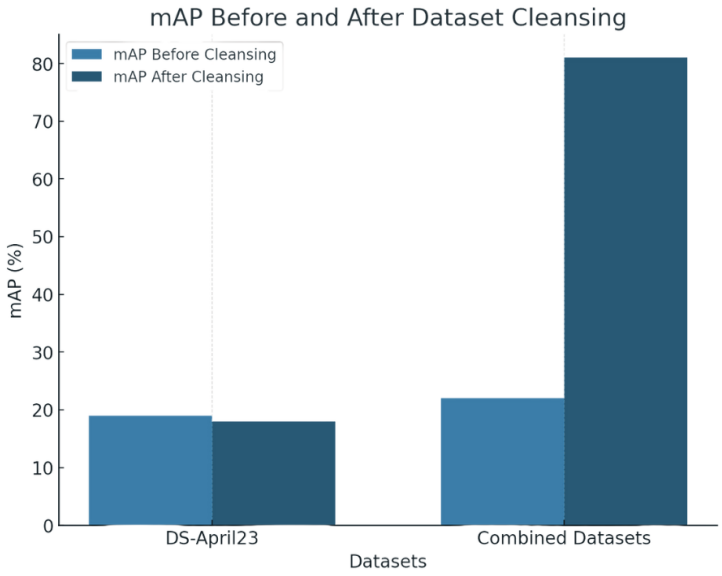


Figure 19: mAP before and after dynamic dataset cleansing.

3.2 Impact of Multi-level Transfer Learning

This section will compare the performance of this innovative approach with a standard training pipeline, each subjected to a total of 500 epochs. The epochs for the multi-level approach were distributed across four levels, with 50, 100, 150, and 200 epochs, respectively.

Efficiency in performance

As seen in Figure 20, the multi-level approach, the model's mean Average Precision (mAP) peaked at 86.7% stabilizing after completing 388 epochs, specifically at level 4. In comparison, the standard training pipeline's mAP stabilized after 374 epochs at a peak of 85.1%. This suggests that both approaches required a similar number of epochs to reach a stable performance level.

The quick rise of mAP in the multi-level approach is because the recall is unified since we have only one class, and the introduction of more data is observed to have a slight deceleration on the mAP improvement after epoch 50. We can also notice the sudden drop at epoch 150 due to switching to multi-class annotations, but the classifier catches up fairly quickly.

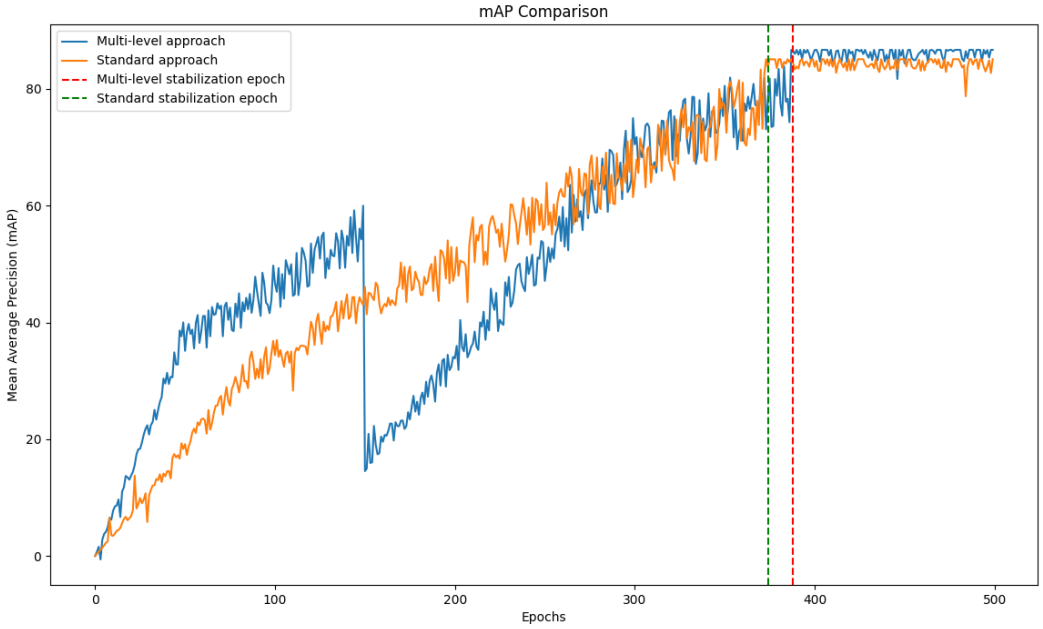


Figure 20: mAP trend comparison between multi-level transfer learning approach and the standard approach.

We can observe in Figure 20 a slow increase, spanning multiple epochs. That's because of the optimum learning rate found at the automated hyperparameter optimization phase, which is the best value found for the model not to overfit. The learning rate is low hence the slow rise. The steady increase observed in both approaches is because of the internal design of the core training pipeline, which has a patience threshold. The pipeline creates checkpoints whenever there's an improvement in both mAP and average loss. If the loss shows improvement but the mAP does not, or if the loss remains stagnant beyond the patience threshold of epochs, the training reverts to the last successful checkpoint. This mechanism is particularly observable in the final epochs, where the mAP stabilizes.

When it comes to mAP, the primary performance metric, no substantial differences were observed between the multi-level transfer learning pipeline and the standard training pipeline. Both methodologies yielded closely comparable mAP scores, implying that the novel, multi-level pipeline is as effective as the traditional method for this particular application.

Computational efficiency

An examination of the computational resources consumed during training revealed that both pipelines were similarly efficient. Neither approach showed a significant drain on computational resources, confirming that the added complexity of the multi-level pipeline did not result in any noticeable inefficiencies.

Further testing

The multi-level transfer learning pipeline was tested only once due to time constraints. For more robust validation, it would be advantageous to perform multiple iterations and average the results, especially given that the behaviour of the optimizer can vary between runs. This aspect remains unexplored due to time limitations.

3.3 Analysis of Metrics

Before diving into the analysis, it is important to note some crucial aspects of our model and dataset that set the context for the evaluation.

Firstly, the dataset comprises three classes of tilapia egg developmental stages: cleavage, blastula-gastrula, and organogenesis, initially containing 2,759, 2,065, and 813 instances, respectively. After undergoing data augmentation and cleansing, the dataset size was approximately quadrupled, and this augmented dataset was employed for testing and evaluation.

Secondly, while the peak performance of the model was reported at 86.7% mAP in the multi-level training pipeline (previous section), we were not able to accurately replicate it for analysis. The evaluation was conducted using the last iteration of the model trained using a standard pipeline. This yielded a slightly lower but still robust performance, with metrics shown in Table 6. The difference in performance between the model that peaked and this iteration could largely be attributed to the stochastic nature of the optimization algorithm (nondeterministic behaviour).

Confusion matrix analysis

The diagonal elements of the matrix represent the true positives for each class. The model has the highest accuracy in predicting cleavage with 1706 correct predictions, followed by

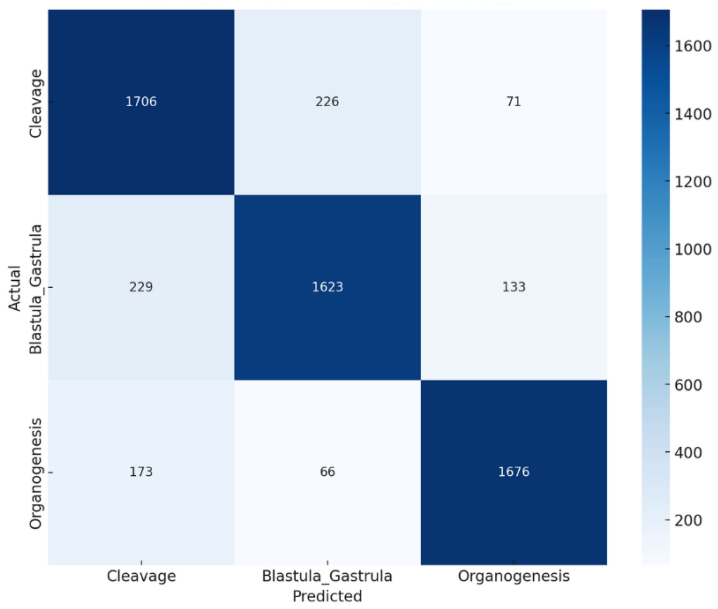


Figure 21: Confusion matrix of the evaluated model.

organogenesis with 1676, and blastula-gastrula with 1623.

The off-diagonal elements give us an insight into the model's mistakes. For instance, cleavage is often misclassified as blastula-gastrula 226 times compared to organogenesis 71 times. This might be attributed to their visual similarity, necessitating further feature engineering which was out of our timeline.

The confusion matrix reveals certain weak spots. Specifically, the model struggles the most with the blastula-gastrula stage, both in terms of false positives and false negatives. This indicates room for model improvement, particularly for this class. We should mention that without upsampling, the model had a bias towards the more represented classes, and left out the organogenesis class specifically. However, after the upsampling of the underrepresented classes, the model showed a more balanced performance across the classes, as reflected in the higher true positive counts for the organogenesis class.

Overall performance analysis

The final evaluation results, as summarized in Table 6, reveal a high degree of model effectiveness across multiple metrics. The mean Average Precision (mAP) stands at 85.4%, closely aligned with the Average Precision of 84.9%, indicating that the model not only identifies the objects accurately but also localizes them effectively within the images. The Average Recall

Table 6: Final evaluation results.

mAP	85.4%
Average Precision	84.9%
Average Recall	84.8%
F-1 Score	84.8%

of 84.8% suggests that the model is adept at capturing most of the relevant instances in the dataset, which is crucial for applications where missing an object could be costly. The F-1 Score, a harmonic mean of precision and recall, also stands at 84.8%, confirming that the model maintains a balanced trade-off between false positives and false negatives. These metrics collectively affirm the model's robustness and reliability, making it well-suited for practical applications in identifying tilapia egg developmental stages.

3.4 System Testing and Efficiency

Server-side performance

Our testing revealed that the server takes an average of 7 seconds to process a single request. This duration was measured by capturing the timestamps of the REST API request and response. Within this time frame, the inference pipeline is responsible for an average of 5 seconds. Notably, Non-Maximum Suppression (NMS) accounts for the majority of this time, consuming approximately 3 seconds on average. However, due to time limitations, we were unable to evaluate the server's capacity to handle multiple concurrent requests.

As for computational resources, the inference operation is relatively light, requiring a negligible amount of CPU and memory resources. It's worth noting that we did not conduct a formal security assessment of the server, choosing to rely solely on the initial security configurations, again primarily due to time constraints.

Client-side user experience

The mobile application was evaluated locally, with a focus on user interface (UI) and user experience (UX). The UI design is straightforward and uncluttered, while the UX is designed to be intuitive and user-friendly. One of the highlights is the loading screen, which takes up the majority of the user's time. To enhance user engagement, we incorporated visually appealing

animations for both the progress circle and the loading text.

Despite these positives, we encountered several issues that warrant attention. Specifically, the layouts appeared disorganized, and there were instances of pixel overflow across the borders when testing on different mobile phone resolutions. Due to time constraints, these issues remain unaddressed.

Limitations and future work

While our evaluation provides valuable insights into the system's performance and user experience, it also highlights several areas where additional testing could be beneficial. These include the server's ability to handle multiple requests, a detailed security assessment, and real-world user testing to gather more actionable feedback.

Conclusion

The chapter has provided a comprehensive analysis covering various facets of the study. We delved into the effects of using different dataset versions and discussed how our multi-level transfer learning pipeline compares with a standard training approach. Moreover, we analyzed critical performance metrics that define the success of our models. Finally, we evaluated the overall system performance and efficiency, both from the server-side and client-side perspectives.

Through this analysis, we have gained valuable insights into the capabilities and limitations of our methodologies. Although some areas require further investigation, the results indicate a promising trajectory for future work. The findings lay a robust foundation for subsequent research, opening avenues for optimizations and refinements that can further enhance the system's performance and utility.

Conclusions

Key accomplishments

This research successfully demonstrates that our prototype system is not only reliable but also performs admirably in the specific and complex task of detecting, counting, and classifying tilapia fish eggs across their various developmental stages. The model's performance has been evaluated using the appropriate evaluation metrics, and it has proved as highly adept.

Methodological advancements

One of the unique contributions of this work is the dynamic dataset cleansing approach. This innovative methodology was crucial in maintaining a high-quality dataset, which, in turn, was instrumental in training a well-performing model. The research methodology was developed incrementally; the failure or success of various techniques prompted us to adapt, modify, or even invent new approaches. Our exploration into multi-level transfer learning did not yield substantially improved results compared to traditional methods. However, the inconclusiveness of these findings suggests that the approach warrants further, more extensive testing.

Challenges and limitations

We navigated a plethora of challenges, including environmental variables, resource constraints, and limited time. Each limitation was either resolved directly or mitigated through alternative solutions. Our research is particularly ambitious in its aim to contribute to aquaculture technology. While filled with various complexities, it has led to the development of a workable prototype, especially significant for the future of commercial fish farming in Algeria.

Future work and long-term vision

Our immediate goal post finishing the thesis is to develop a fully automated, dedicated hardware system that builds upon the existing technologies but offers an improved user experience tailored to the specific environmental conditions where the device will operate. The research has unexplored dimensions that we plan to investigate further, including more exhaustive testing and a broader architectural comparison of deep learning models.

Final remarks

This work serves as an important stepping stone towards the advancement of aquaculture in Algeria. It aims to simplify and optimize the commercial production cycle of fish farming through an AI-assisted system for fish egg counting and classification. Despite the limitations and assumptions that constrained our study, this research lays the groundwork for future work that will aim to cover these gaps.

Appendix

A. Business Summary: Smart Aqua Solutions

Business name: Smart Aqua Solutions

Figure 22: Smart Aqua Solutions' Logo

Introduction

Project Idea

The project, named TilCount, originated from a need identified by AquaPro LLC, a fish hatchery and a representative of TilAqua International Company based in Holland. The primary focus is to address the inefficiencies in tilapia fish egg counting and classification. Designed as a standalone solution for immediate market entry, plans for future integrations with existing farm management systems are underway.

Value Proposition

TilCount aims to revolutionize tilapia fish farming by automating and optimizing egg counting and classification using AI. The core benefits include:

- Reduction in labor costs
- Minimization of human error
- Adaptability to complex environmental changes

While the immediate advantages are economic, the AI system is engineered to adapt to environmental changes, adding a layer of resilience to the operations.

Team

The core team currently comprises a single multi-talented individual who serves as the developer and project manager. Expansion plans are strategic and will be triggered by future needs. Mr. Bouzidi, an expert in the field, serves as the key advisor and mentor, providing valuable guidance and expertise.

Goals

Short-term Goals (6-12 months)

- Finalize the prototype within one month.
- Launch the solution in Algeria and target local fish hatcheries within four months.
- Offer custom solutions based on market feedback.

Long-term Goals (1-3 years)

- Stabilize the product and dominate the Algerian market within 1.5 years.
- Expand into neighboring countries and venture into the broader African market within 2-3 years.

Timeline

1. Finalize the prototype: 1 month
2. Launch in Algeria: 4 months
3. Update and stabilize in Algeria: 1.5 years
4. International Expansion: 2-3 years

Critical dependencies that could impact the timeline include technological development and sales performance.

Innovation

Technology

Machine Learning Algorithm

Utilizing Mask RCNN with a modified backbone aligns with the premium need for accuracy in egg classification and counting. This state-of-the-art algorithm ensures that the system performs at the highest standards.

Proprietary Technologies

While the system is built on open-license technologies, it still fulfills the specialized requirements set by the industry, avoiding the complications of proprietary limitations.

Scalability

While the technology has not yet been tested for scalability across different species, its design makes it highly plausible that it could adapt to such demands.

Hardware Requirements

The system operates primarily on server-side resources, necessitating a fast CPU and robust GPU for optimal performance.

Data Security

Data is treated with utmost confidentiality; it is encrypted during transmission and deleted immediately post-processing, ensuring stringent security protocols.

Future Integrations

IoT technologies are on the horizon for future iterations, providing avenues for further technological synergy and utility.

Use Case

User Interaction

The application is designed with a focus on user-friendliness and intuitive interaction. The user journey includes the following steps:

- Welcome Page
- Image Acquisition
- Image Submission via REST API
- Loading Screen
- Results Display
- Option to Restart
- Limitations and Constraints

According to real-world applications and the company's requirements, no notable limitations or constraints have been identified.

Environmental Adaptability

The technology is robust enough to handle complex environmental conditions in hatcheries, attributable to its powerful machine-learning model.

Training and Onboarding

The system is designed for ease of use, requiring no specialized training. The interface prioritizes functionality over design complexity.

User Feedback

A direct channel with the user's manager at the partnering company facilitates real-time feedback, allowing for continual system improvements.

Market Analysis

Market Segmentation and Targeting

Focus Hatcheries Our initial focus is Aqua Pro LLC, a key player in the Algerian aquaculture market. As the company scales, other opportunities within Algeria and beyond will be explored.

Market Needs

While there are no specific regulations demanding precise egg counting, the urgent need lies in hatcheries' desire for effective Business Intelligence. Accurate and timely data is crucial for scaling operations and financial forecasting.

Market Trends

Automated sorting devices present an opportunity to complement our solution. While AI startups in aquaculture command a $3.8B market share, growth rates are yet to be projected, indicating the need for further market testing.

Market Challenges

The primary barriers to entry include adoption inertia and regulatory hurdles. Cultural factors have also influenced the slow uptake of technology, although this may change with the introduction of effective solutions like TilCount.

Competitor Analysis

Current competitors lack specialization in tilapia egg counting. Our solution offers unique mechanisms for real-time results and scalability.

Pricing Strategy

While hatcheries are moderately sensitive to pricing, our product's uniqueness positions us as potential price setters in the market.

Distribution Channels

Collaboration with governmental bodies could significantly aid in distribution and market normalization.

Customer Relationships

Interactions with technology vendors in the industry lean more towards partnerships, aligning well with our approach.

Stakeholder and Regulatory Environment

An endorsement from TilAqua International could significantly boost market acceptance. No specific certifications or approvals have been identified as necessary for technology deployment in aquaculture.

Production

Production Infrastructure

Server Architecture

The core of the technology relies on a local server housed on-premises. The specifications are as follows:

- CPU: Intel i5
- GPU: NVIDIA GT740
- RAM: 8GB
- OS: Ubuntu
- Backend: Flask

Redundancy and Backup

As of now, there are no dedicated plans for redundancy and backup in case of system failure. Given the critical nature of the operations, future iterations will need to address this gap.

Software Updates

Updates to the software are executed internally and are made available to customers through in-app notifications for seamless upgrading.

Production Cycle & Workflow

Workflow

The system employs an end-to-end workflow that includes:

- Image Capture
- Pre-processing
- Model Inference
- Post-processing
- Insight Delivery

Latency

The system aims to deliver results within a maximum latency of 10 seconds from the time of capturing an image. This rapid turnaround is crucial for real-time decision-making.

Machine Learning Model Updates

Updates to the machine learning model occur as and when new data is collected. The absence of a formalized Quality Assurance process for validating model updates is a point to be addressed in future versions.

Scalability & Adaptability

Due to a focus on initial market entry, the system's current processing capacity in terms of concurrent images or farms has not been tested. Plans for geo-distributed servers to cater to international expansion are not in place yet but will be considered as the customer base grows.

Security & Compliance

Data Encryption

The system employs standard HTTPS encryption protocols for data in transit. Data at rest is stored in an encrypted format, ensuring a secure environment.

User Interaction & Feedback

The system currently lacks an automated mechanism for gathering user feedback. As it stands, direct communication with the company manager serves this role. There are no options for users to customize any part of the system, maintaining uniformity across different user experiences.

Financial Plan

Revenue Streams and Pricing

- **Pricing Model:** The TilCount product will offer a pricing scheme based on the number of images processed. For every 20 images, the cost will be 3,000.00DA.
- **Customer Lifetime Value (CLV):** The expected CLV is approximately 3,000,000.00DA.

Costs and Expenditures

- **Initial and Recurring R&D Costs:** Initial R&D expenditure is estimated at DZD 2,000,000, with an annual recurring cost of DZD 500,000 for improvements.
- **Hardware Costs:** A basic local server setup is anticipated to cost around DZD 1,000,000, with an annual maintenance fee of about DZD 200,000.
- **Marketing Budget:** Initial marketing expenditure is set at DZD 1,500,000, focusing mainly on digital campaigns and local expositions.

Financial Projections

- **First Year Revenue:** The target is to achieve a revenue of DZD 500,000 in the first year.

- **Scaling:** Anticipated revenue of DZD 6,000,000 in the third year and DZD 10,000,000 in the fifth year.
- **Profit Margins:** Gross profit margins are projected to be around 60%, with net profit margins in the range of 20-25%.

Key Performance Indicators (KPIs)

- **Customer Acquisition Cost (CAC):** Targeting a CAC of around DZD 50,000.
- **Average Revenue Per User (ARPU):** Aiming for an ARPU of approximately DZD 200,000 annually.

Break-even Point

Expected by the middle or end of the second year, assuming consistent customer acquisition and controlled overhead costs.

Risk Assessment and Mitigation

- **Financial Risks:** Currency fluctuations are currently a minor concern but will become significant in the case of international expansion.
- **Risk Mitigation:** Contracts that adjust for inflation and diversification into related aquaculture segments.

Funding and Investment

- **Seed Round:** Aim to raise DZD 3,000,000 at a pre-money valuation of DZD 15,000,000.
- **Series A:** A subsequent Series A round could aim for DZD 10,000,000 at a pre-money valuation of DZD 50,000,000.

Financial Governance

- **Financial Reporting:** Annual reporting will be standard, and quarterly if external investors are involved.
- **Financial Audits:** An annual financial audit.

Prototype

The prototype, named TilCount, is an AI-powered mobile application designed to automate and optimize tilapia fish egg counting and classification. Serving as a pilot project for AquaPro LLC, the app aims to significantly reduce labor costs and human error. It capitalizes on smartphones' ubiquity and computational power to offer a cost-effective and user-friendly solution.

Technology & Algorithms

TilCount employs Mask RCNN as the core machine learning algorithm. Although no alternative algorithms were tested, Mask RCNN has proven its efficacy in similar applications. The technology can process one image every seven seconds, offering reasonable speed for the task at hand.

System Requirements & Limitations

The system is hosted on a local server featuring an i5 CPU, a GT740 GPU, and 8GB RAM. It runs on an Ubuntu OS and employs Flask for the backend. Currently, there are no contingencies for system downtime or failures, and the server's scalability to handle a rapid increase in user numbers is yet to be considered.

User Interaction

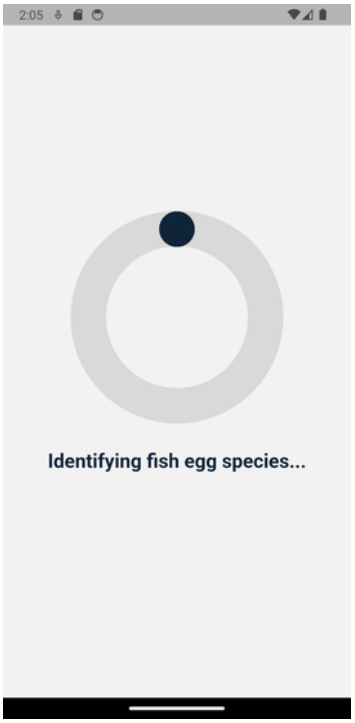
The app offers an intuitive user interface that starts with a welcome screen, guiding users through its functionalities. While the system is designed to be user-friendly, it currently lacks mechanisms to handle poor-quality images or those not meeting specific criteria.

Adaptability & Future Development

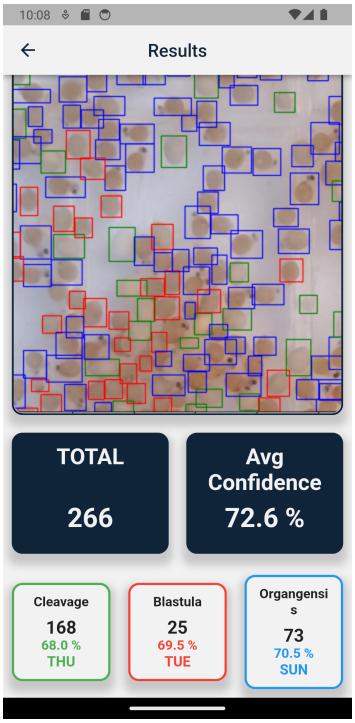
Plans are in place to introduce other machine learning models and algorithms. IoT integration is on the roadmap, aimed at enabling hardware specialization for each tray, thereby achieving parallelization and full automation. Additionally, the system can easily adapt to classify and count eggs of other aquatic species, provided new data is collected.

((a)) Welcome page

((b)) Image acquisition and uploading page



((c)) Loading page



((d)) Results page

Figure 23: The mobile application's screen pages

Security & Compliance

Standard encryption policies are in place to secure data, which is deleted immediately after processing. Although security audits and adherence to compliance standards are not currently relevant, a patent for the technology is pending.

Production Cycle & Workflow

The workflow starts with pre-processing steps like image enhancement and resizing before the image is fed into the machine-learning model. No post-processing steps are currently implemented to refine the results after model inference.

Scalability & Adaptability

No plans are currently in place for load balancing and data distribution, especially concerning potential international expansion. Handling a rapid increase in the number of concurrent users remains a subject for future consideration.

User Feedback & Customization

While user customization features are not planned, user feedback will be actively incorporated into future updates. Users can directly contact the development team, and necessary changes will be implemented in subsequent updates.

Stakeholder & Regulatory Environment

Governmental involvement could potentially aid distribution through interest-free public funds for startups, with a 49% equity stake held until the loan is repaid.

Future Considerations

Critical elements like system downtime contingencies, scalability to handle increased user numbers, and mechanisms to handle poor-quality images are subjects for future development.

B. Codebase

Link: https://github.com/elhadjmb/Xqua

Bibliography

[1] Food and A. O. of the United Nations, *The state of world fisheries and aquaculture 2020*, 2020. [Online]. Available: http://www.fao.org/3/ca9229en/online/ca9229en.html%5C#chapter-executive%5C%5Fsummary.

[2] "How the global fish market contributes to human nutrition", *Nature*, 2019. [Online]. Available: https://www.nature.com/articles/s41598-019-57280-3.

[3] *Algerian aquaculture: poised for growth*, The Fish Site, 2018. [Online]. Available: https://thefishsite.com/articles/algerian-aquaculture-poised-for-growth.

[4] *Aquaculture en algrie : leve d'obstacles sur 105 projets*, Algerie-Eco, 2022. [Online]. Available: https://www.algerie-eco.com/2022/03/16/aquaculture-levee-dobstacles-sur-105-projets/.

[5] "Automatic counting methods in aquaculture: a review", *ResearchGate*, 2020. [Online]. Available: https://www.researchgate.net/publication/345546225%5C%5FAutomatic%5C%5Fcounting%5C%5Fmethods%5C%5Fin%5C%5Faquaculture%5C%5FA%5C%5Freview.

[6] *Aquaculture in algeria overview*, Aquanet. [Online]. Available: https://www.aquanet.com/algeria%5C%5Fcountry.

[7] *Counting eggs article*, TroutLodge. [Online]. Available: https://www.troutlodge.com/en/articles/counting-eggs/.

[8] "Deep learning approach to automate fish egg counting: a systematic literature review", *ResearchGate*, 2022. [Online]. Available: https://www.researchgate.net/publication/361772622%5C%5FDeep%5C%5FLearning%5C%5FApproach%5C%5Fto%5C%5FAutomate%5C%5FFish%5C%5FEgg%5C%5FCounting%5C%5FA%5C%5FSystematic%5C%5FLiterature%5C%5FReview.

[9] *Syndel prosorter product page*, Syndel. [Online]. Available: https://syndel.com/product/prosorter/.

[10] *Syndel quicksorter product page*, Syndel. [Online]. Available: https://syndel.com/product/quicksorter/.

[11] *A quick guide to tilapia breeding and farming*. [Online]. Available: https://thefishsite.com/articles/a-quick-guide-to-tilapia-breeding-and-farming.

[12] D. Malowany, "The battle of speed vs. accuracy: single-shot vs. two-shot detection meta-architectures", *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/the-battle-of-speed-vs-23b61eb4225d.

[13] *Aquaculture: 20 fermes aquacoles entrent en production*, 2009. [Online]. Available: https://www.algerie360.com/aquaculture-20-fermes-aquacoles-entrent-en-production/.

[14] *La production aquacole en algerie*. [Online]. Available: https://agronomie.info/fr/la-production-aquacole-en-algerie/.

[15] *A guide to farming tilapia: on-growing techniques*, The Fish Site, 2023. [Online]. Available: https://thefishsite.com/articles/a-guide-to-farming-tilapia-ongrowing-techniques.

[16] Hussain and M. G., "Red tilapia: a new candidate for culture", *Aquaculture Asia*, vol. 5, no. 1, pp. 28–31, 2000.

[17] *Information on the advantages of culturing tilapia*. [Online]. Available: https://digitalarchive.worldfishcenter.org/bitstream/handle/20.500.12348/4334/41f2c2535d4223955538fc1848a1fadf.pdf?sequence=2.

[18] *Monosex tilapia*. [Online]. Available: https://www.aquaticcommunity.com/tilapia/monosex.php.

[19] Pandit, N. P., Nakamura, and M., "Effects of high temperature on the survival and hatching rate in the monosex nile tilapia and oreochromis niloticus l", *Aquaculture*, vol. 218, no. 1-4, pp. 471–478, 2003.

[20] *Monosex tilapia fish*. [Online]. Available: https://www.bestoffarming.com/monosex-tilapia-fish/.

[21] Baroiller, J. F., D'Cotta, and H., "Environment and sex determination in farmed fish", *Comparative Biochemistry and Physiology Part C: Toxicology and Pharmacology*, vol. 130, no. 4, pp. 399–409, 2001.

[22] M. M. Beida, *Director of the direction of fishery and marine resources in laghouat*, 2023.

[23] *Eyed stage of tilapia eggs*. [Online]. Available: http://fishconsult.org/?p=1012.

[24] C. Svellingen and A. Albani, "Deep learning for fish weight estimation in aquaculture using 3d optical stereo systems and computer vision techniques", *Journal of Imaging*, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0144860917301176.

[25] F. Lin and W. Zhang, "An intelligent aquaculture system based on deep learning and iot", *Computers and Electronics in Agriculture*, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954122002977.

[26] N. Steeves and D. Ames, "Automating the processing of shellfish aquaculture lease boundary data for the state of maine", *Frontiers in Marine Science*, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fmars.2021.823173/full.

[27] Y. Zhang and Z. Liu, "Automatic fish counting in noisy underwater videos based on fish shape and motion information", *Aquacultural Engineering*, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8976960/.

[28] Li, D., *et al.*, "Review of computer vision in fish and seafood industry", *Trends in Food Science and Technology*, vol. 93, pp. 1–12, 2019.

[29] Gephart and J. A. et al., "Environmental performance of blue foods", *Nature*, vol. 597, no. 7875, pp. 95–100, 2021.

[30] Christensen, A., Lautrup, and B., "A review of image analysis methods for estimating fish biomass and size distributions in fish farming", *Reviews in Aquaculture*, vol. 11, no. 4, pp. 882–897, 2019.

[31] Yang and Y. et al., "Ai-driven water quality monitoring and management in aquaculture", *Aquaculture International*, vol. 29, no. 3, pp. 1027–1040, 2021.

[32] Jothiswaran and J. et al., "Ai in aquaculture production", *Aquaculture Reports*, vol. 18, p. 100 505, 2020.

[33] Naylor, R. L., *et al.*, "Effect of aquaculture on world fish supplies", *Nature*, vol. 405, no. 6790, pp. 1017–1024, 2000.

[34] "Applying artificial intelligence (ai) techniques to implement a practical smart cage aquaculture management system", *Springer*, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s40846-021-00621-3.

[35] *Object detection*. [Online]. Available: http://en.wikipedia.org/wiki/Object%5C%5Fdetection.

[36] *Beginner's guide to object detection algorithms*. [Online]. Available: https://medium.com/analytics-vidhya/beginners-guide-to-object-detection-algorithms-6620fb31c375.

[37] *A step-by-step introduction to the basic object detection algorithms (part 1)*, 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/.

[38] *Object recognition with deep learning*. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[39] W. Wu and Z. Liu, "On the impact of object detection metrics for video analytics", *2021 IEEE 23rd International Conference on High Performance Computing and Communications; IEEE 19th International Conference on Smart City; IEEE 7th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/10099639.

[40] H. Liu, X. Ma, Y. Yu, L. Wang, and L. Hao, "Application of deep learning-based object detection techniques in fish aquaculture: a review", *Journal of Marine Science and Engineering*, vol. 11, no. 4, 2023, ISSN: 2077-1312. DOI: 10.3390/jmse11040867. [Online]. Available: https://www.mdpi.com/2077-1312/11/4/867.

[41] *Object detection using deep learning algorithms: a review.* [Online]. Available: https://arxiv.org/pdf/1807.05511.pdf.

[42] *Top 8 algorithms for object detection.* [Online]. Available: https://analyticsindiamag.com/top-8-algorithms-for-object-detection/.

[43] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: scalable and efficient object detection", *arXiv preprint arXiv:1911.09070*, 2019. [Online]. Available: https://arxiv.org/abs/1911.09070.

[44] Everingham, M., *et al.*, "The pascal visual object classes (voc) challenge", 2010.

[45] Rezatofighi, H., *et al.*, "Generalized intersection over union: a metric and a loss for bounding box regression", 2019.

[46] *On object detection metrics with worked example.* [Online]. Available: https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e.

[47] Lin, T. Y., *et al.*, "Focal loss for dense object detection", *arXiv preprint arXiv:1708.02002*, 2017. [Online]. Available: https://arxiv.org/abs/1708.02002v2.

[48] Unknown, "Introduction to object detection model evaluation", *Towards Data Science*, Unknown. [Online]. Available: https://towardsdatascience.com/introduction-to-object-detection-model-evaluation-3a789220a9bf.

[49] Girshick, R., *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation", 2014.

[50] L. T, D. P, G. R, and et al., "Feature pyramid networks for object detection", 2017. [Online]. Available: https://arxiv.org/abs/1612.03144.

[51] J. Hui, "Understanding feature pyramid networks for object detection (fpn)", *Medium*, 2018. [Online]. Available: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c.

[52] He, Kaiming, Zhang, *et al.*, "Deep residual learning for image recognition", *arXiv preprint arXiv:1512.03385*, 2015. [Online]. Available: https://arxiv.org/abs/1512.03385v1.

[53] Nepal and Prabin, *Deep residual learning for image recognition (resnet paper explained)*, 2020. [Online]. Available: https://medium.com/analytics-vidhya/deep-residual-learning-for-image-recognition-resnet-paper-explained-26b29e0fa73e.

[54] Lin, T., Dollár, P., Girshick, R., *et al.*, "Feature pyramid networks for object detection", in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 936–944.

[55] P. Raikote, "Object detection part 5: faster r-cnn", *Applied Singularity*, Jun. 2021. [Online]. Available: https://appliedsingularity.com/2021/06/08/object-detection-part-5-faster-r-cnn/.

[56] "Comparison: mask rcnn vs. faster r-cnn". Accessed: 2023-09-05. (2023), [Online]. Available: https://roboflow.com/compare/mask-rcnn-vs-faster-r-cnn.

[57] H. Tahir, M. Shahbaz Khan, and M. Owais Tariq, "Performance analysis and comparison of faster r-cnn, mask r-cnn and resnet50 for the detection and counting of vehicles", in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021, pp. 587–594. DOI: 10.1109/ICCCIS51004.2021.9397079.

[58] "The overall network architecture of mask r-cnn". (), [Online]. Available: https://www.researchgate.net/figure/The-overall-network-architecture-of-Mask-R-CNN%5C_fig1%5C_336615317.

[59] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN", *CoRR*, vol. abs/1703.06870, 2017. arXiv: 1703.06870. [Online]. Available: http://arxiv.org/abs/1703.06870.

[60] E. Odemakinde. "Everything about mask r-cnn: a beginner's guide". Accessed: 2023-09-05. (2023), [Online]. Available: https://viso.ai/deep-learning/mask-r-cnn/.

[61] Awalludin, E.A., Arsad, T.N., W. Yussof, and W.N., "Counting tilapia larvae using images captured by smartphones", *Aquaculture and Fisheries*, 2022. DOI: 10.1016/j.atech.2022.100160. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375522001241.

[62] *Raspberry pi 4 vs smartphone*. [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=260957.

[63] *Benchmark the ai performance of android devices*. [Online]. Available: https://benchmarks.ul.com/news/benchmark-the-ai-performance-of-android-devices.

[64] *Deep learning models on smartphones*. [Online]. Available: https://link.springer.com/article/10.1007/s11036-019-01445-x.

[65] Syntheticus, "The benefits and limitations of generating synthetic data", 2021. [Online]. Available: https://syntheticus.ai/blog/the-benefits-and-limitations-of-generating-synthetic-data.

[66] "Image data collection in 2023: what it is and best practices". (), [Online]. Available: https://research.aimultiple.com/image-data-collection/ (visited on 09/17/2023).

[67] M. Riaz, S. Park, M. Ahmad, W. Rasheed, and J.-A. Park, "Generalized laplacian as focus measure", vol. 5101, Jun. 2008, pp. 1013–1021, ISBN: 978-3-540-69383-3. DOI: 10.1007/978-3-540-69384-0_106.

[68] *Best image annotation tools for computer vision*. [Online]. Available: https://blog.roboflow.com/best-image-annotation-tools/.

[69] *A comprehensive guide to image augmentation using pytorch*. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-image-augmentation-using-pytorch-fb162f2444be.

[70] A. B. Jung, K. Wada, J. Crall, *et al.*, *imgaug*, https://github.com/aleju/imgaug, 2020.

[71] Ai and Ml, "Why data annotation is so significant for machine learning and ai", 2023. [Online]. Available: https://www.habiledata.com/blog/why-data-annotation-is-important-for-machine-learning-ai/.

[72] J.-C. Klie, R. E. de Castilho, and I. Gurevych, "Analyzing Dataset Annotation Quality Management in the Wild", *arXiv*, 2023. DOI: 10.48550/arXiv.2307.08153. eprint: 2307.08153.

[73] J. Brownlee, "Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates - MachineLearningMastery.com", *MachineLearningMastery*, Aug. 2020. [Online]. Available: https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates.

[74] M. Walia, "Overfitting in Machine Learning and Computer Vision", *Roboflow Blog*, Apr. 2023. [Online]. Available: https://blog.roboflow.com/overfitting-machine-learning-computer-vision.

[75] G. F. Volpi, "Class Imbalance: a classification headache - Towards Data Science", *Medium*, Dec. 2021. [Online]. Available: https://towardsdatascience.com/class-imbalance-a-classification-headache-1939297ff4a4.

[76] M. Fawzy, "Urban Feature Extraction From High Resolution Satellite Images", *ResearchGate*, Nov. 2020. DOI: 10.13140/RG.2.2.11702.93760/1.

[77] A. Christiansen, "Anchor Boxes—$$*Thekeytoqualityobjectdetection*$|$*TowardsDataScience*", *Medium*, Jun. 2022. [Online]. Available: https://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9.

[78] M. S. Mahmood, "Outlier Detection (Part 1) - Towards Data Science", *Medium*, Oct. 2022. [Online]. Available: https://towardsdatascience.com/outlier-detection-part1-821d714524c.

[79] S. Lee, "Simplify Polylines with the Douglas Peucker Algorithm", *Medium*, Jan. 2022. [Online]. Available: https://towardsdatascience.com/simplify-polylines-with-the-douglas-peucker-algorithm-ac8ed487a4a1.

[80] W. A. Mustafa and M. M. M. A. Kader, "A Review of Histogram Equalization Techniques in Image Enhancement Application", *J. Phys. Conf. Ser.*, vol. 1019, no. 1, p. 012 026, Jun. 2018, ISSN: 1742-6596. DOI: 10.1088/1742-6596/1019/1/012026.

[81] He, Kaiming, Zhang, *et al.*, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. DOI: 10.1109/ICCV.2015.123.

[82] *ImageNet Dataset*. [Online]. Available: https://image-net.org/index.php.

[83] *The Essential Guide to Pytorch Loss Functions*, [Online; accessed 17. Sep. 2023], Sep. 2023. [Online]. Available: https://www.v7labs.com/blog/pytorch-loss-functions.

[84] O. Team, *Optuna: an open source hyperparameter optimization framework in python*, GitHub repository, 2023. [Online]. Available: https://github.com/optuna/optuna.

[85] A. Nair, "Grid search vs random search vs bayesian optimization", *Towards Data Science*, 2022. [Online]. Available: https://towardsdatascience.com/grid-search-vs-random-search-vs-bayesian-optimization-2e68f57c3c46.

[86] R. Patel, "Hyperparameter search using bayesian optimization and an evolutionary algorithm", *Medium*, 2023. [Online]. Available: https://rohit10patel20.medium.com/hyperparameter-search-using-bayesian-optimization-and-an-evolutionary-algorithm-bdca6331de1c.

[87] Shin, Hoo-Chang, Roth, *et al.*, "Deep convolutional neural networks for computer-aided detection: cnn architectures and dataset characteristics and transfer learning", *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016. DOI: 10.1109/TMI.2016.2528162. [Online]. Available: https://ieeexplore.ieee.org/document/7404017.

[88] *ZeroTier | Global Area Networking*, [Online; accessed 17. Sep. 2023], Sep. 2023. [Online]. Available: https://www.zerotier.com.

[89] P. Projects, *Flask: a lightweight web application framework*, 2023. [Online]. Available: https://flask.palletsprojects.com/.

[90] *Dynu*, [Online; accessed 17. Sep. 2023], Sep. 2023. [Online]. Available: https://www.dynu.com/en-US.

[91] A. Rosebrock, *Non-maximum suppression for object detection in python*, 2014. [Online]. Available: https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/.

[92] Google, *Flutter: a ui toolkit for building natively compiled applications*, 2023. [Online]. Available: https://flutter.dev/.

[93] *Rest api: a set of rules for how a client should request data from a server and how the server should respond to those requests*, 2023. [Online]. Available: https://restfulapi.net/.