https://doi.org/10.32815/jitika.v19i2.1184

Volume 19, Issue 2, 2025

Jurnal Ilmiah Teknologi Informasi Asia

Journal Homepage: https://jurnal.asia.ac.id/index.php/jitika

Article

Implementation of the YOLOv5 Algorithm on an Android Platform for counting catfish fry (*clarias sp.*)

Mohammad Zainuddin *, Muhammad Saifuddin Zuhri

Teknik Informatika, Institut Teknologi Dan Bisnis Asia Malang, Malang City, 65113, Indonesia

***Abstract*—Catfish aquaculture in Indonesia faces an efficiency challenge in its fry counting process, which still relies on manual methods. This research aims to develop and evaluate a mobile application based on the Android operating system that implements the YOLOv5 algorithm for the automated, real-time detection and counting of catfish fry. The model was trained using an image dataset from Roboflow and integrated into an application developed with the Flutter framework. The model's performance was quantitatively assessed using Precision, Recall, and F1-Score metrics across three scenarios: normal, clustered (occlusion), and shadowed conditions. The test results show that the best performance was achieved under normal conditions, with an F1-Score of 0.949. Performance decreased when the fry was clustered (F1-Score of 0.874) due to object occlusion, and also under shadowed conditions (F1-Score of 0.786) because of false positive detections. These findings confirm the suitability of YOLOv5 for fry counting applications on mobile devices while also highlighting critical areas for improvement, particularly in handling lighting variations and overlapping objects.**

***Keywords*—android; catfish fry; image processing; object detection; yolov5.**

1. Introduction

The catfish (*Clarias sp.*) is a freshwater fish species with high economic value and market demand in Indonesia (Yumna et al., 2019). Moreover, catfish possess several advantages, such as rapid growth, adaptability to poor environmental conditions, and high nutritional content (Ciptawati et al., 2021). The demand for catfish continues to increase in correspondence with population growth and changing consumption patterns, which has subsequently affected the growth of catfish production (Sitio et al., 2017). Furthermore, the annual growth of catfish production averaged 11.77% between 2013 and 2018, indicating its significant role in the aquaculture sector (Fauziyah et al., 2019). One of the most critical phases in the cultivation cycle is fry management, where the counting process is a fundamental activity for stock estimation, feed planning, and harvest prediction.

Currently, the method for counting fry among farmers is still performed manually. This process is inefficient, labor-intensive, and requires a substantial amount of time—it can take 1.5 to 2 hours to count a single bag containing approximately 2,000 fry. Additionally, the accuracy of manual counting is often compromised by factors such as the very small size of the fry and their constant, agile movements, which increases the potential for human error (Dendi & Sunardi, 2021).

* Corresponding author.
E-mail Address: mzein@asia.ac.id (M. Zainuddin)

Author E-mail(s): MZ (mzein@asia.ac.id), MSZ (sauzum73@gmail.com)
Digital Object Identifier 10.32815/jitika.v19i2.1184
Manuscript submitted 19 July 2025; revised 25 September 2025; accepted 26 September 2025.
ISSN: 2580-8397(O), 0852-730X(P).

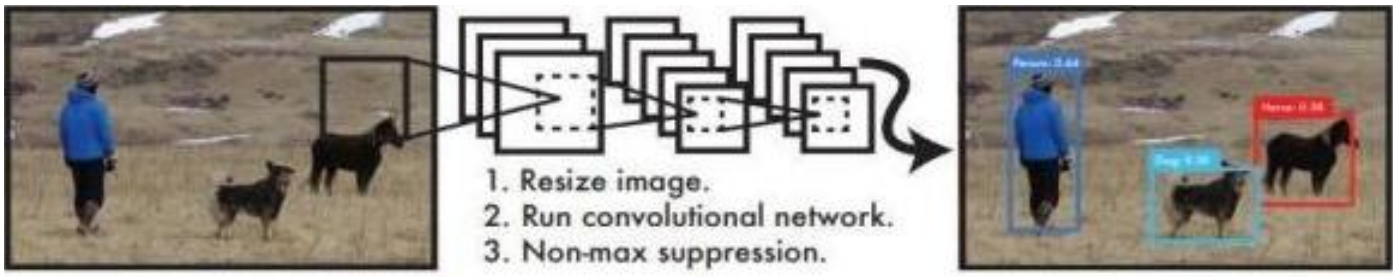

Fig. 1. Example of bounding box

The rapid advancement of deep learning technology, particularly in the field of computer vision, offers a potential solution to automate and improve the accuracy of this process. Image processing is necessary to obtain the desired information from an image, which is a form of information that can be analyzed and used by humans (Sulistiyanti et al., 2016). The You Only Look Once (YOLO) algorithm is an advanced object detection model that has revolutionized real-time detection by processing an entire image in a single pass to simultaneously generate bounding box coordinates and class probabilities, as illustrated in Fig. 1. Several previous studies have demonstrated the successful implementation of YOLO for agricultural object detection (Arganata et al., 2020; Badgujar et al., 2024), including the counting of gourami fry using a webcam. The developed system was able to detect and count the fish fry with a detection accuracy of 82–85% and a counting accuracy of up to 100% when using a mode-based approach. However, a major limitation of that research was its reliance on a static platform (a computer), which reduces its practicality and mobility for direct use in fish farm environments.

To address this limitation, the present study focuses on the development and implementation of a YOLOv5 model integrated into a mobile application for the Android operating system. The objective is to provide a portable, fast, and accurate tool for fish farmers. The main contribution of this research is the evaluation of the model's performance under various realistic environmental conditions (normal, clustered, and shadowed) to measure the system's robustness and identify its limitations in practical field application scenarios.

The results of this study indicate that the developed application can achieve excellent detection performance under ideal conditions (F1-Score of 0.949), but experiences a significant performance decline in more complex situations. Specifically, performance decreases when the fry was clustered (F1-Score of 0.874) due to failures in detecting overlapping objects, and it declines further under shadowed conditions (F1-Score of 0.786) due to object misidentification. These findings provide quantitative evidence of the system's capabilities and limitations.

The remainder of this paper is organized as follows: Section 2 describes the research methodology, including theoretical foundations, model training process, application implementation, and testing scenarios. Section 3 presents and discusses the detailed test results. Finally, Section 4 summarizes the conclusions of this study and provides recommendations for future development.

2. Method

2.1. Theoretical foundation

2.1.1. You Only Look Once (YOLO)

YOLO is a family of single-stage object detection algorithms designed for speed and efficiency. In the YOLO framework, object detection components are placed within a single neural network, allowing for end-to-end training while maintaining a good level of precision.

YOLO divides the input image into an S×S grid, where S is 7, and the input image size is 448 x 448 pixels, as shown in Fig. 2. The process of forming a bounding box is accomplished through convolution on the input image. This results in a bounding box size of S×S×(B×5 + C), where B is the number of bounding boxes (typically 2) in a single grid cell, and C is the number of classifiable classes. The value B is multiplied by 5 because each bounding box requires storage of five values: the x-coordinate, y-coordinate, width, height, and a confidence score (Armalivia et al., 2021). Next, the system runs a single convolutional network on the image. Finally, the system provides object detections based on the confidence level according to the trained model.

2.1.2. Convolutional neural network (CNN)

A CNN is an advancement of the Multilayer Perceptron (MLP) specifically designed to process two-dimensional data. CNNs are classified as Deep Neural Networks because they have a very deep network structure and are frequently applied to image data. For image classification tasks, MLPs are less effective because they do not preserve the spatial information of the image data and treat each pixel as an independent feature, which leads to unsatisfactory results (Putra, 2016).

A CNN is composed of a 3-dimensional arrangement of neurons: width, height, and depth. The width and height represent the size of the layer, while the depth indicates the number of layers in the network. A CNN can have tens to hundreds of layers, where each layer is trained to recognize specific features in an image. The image processing is applied to each training image at different resolutions. The output from processing each image is then combined and becomes the input

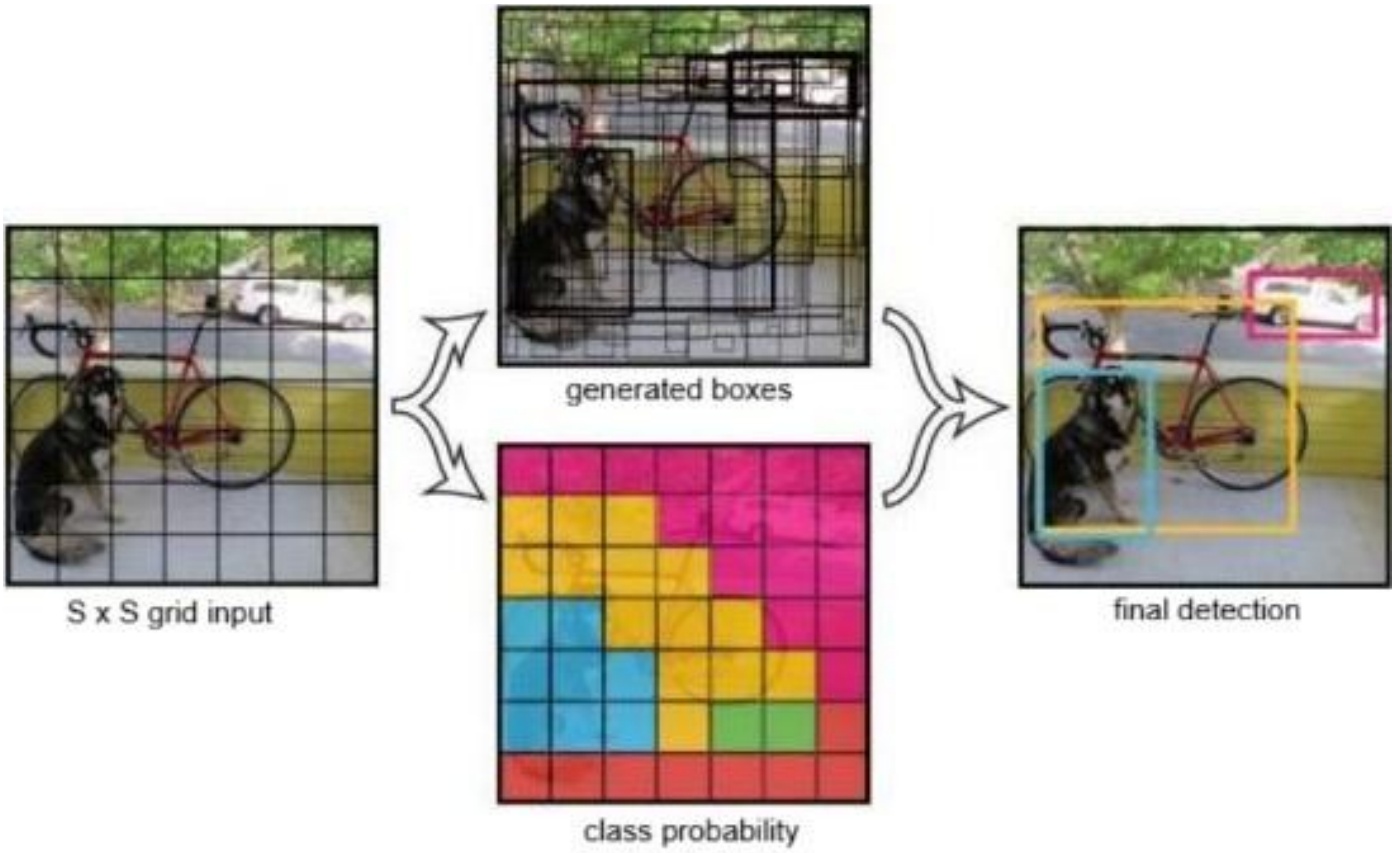

Fig. 2. YOLO workflow for object detection

Table 1. Google Colab Pro+ Specifications

No.	Component	Specification
1	RAM	334.6 GB
2	Runtime	TPU V2
3	Operating System	Linux
4	Disk Space	225.3 GB
5	Programming Language	Python 3.10.12
6	Library	PyTorch, OpenCV

for the next layer. This image processing can start with very simple features, such as edge detection, and progress to more complex features, such as recognizing faces or specific objects (Ilahiyah & Nilogiri, 2018).

2.2. Model training and validation

The system implementation is a critical phase in project development that ensures all designed components can function correctly as a whole. An optimal development environment setup is the first step toward successful system implementation. For this research, Google Colab Pro+ was used as the primary development environment; its specifications are detailed in Table 1.

The model training process is an essential stage in the development of a machine learning-based system. The use of appropriate evaluation metrics is very important for model validation. Metrics such as mean Average Precision (mAP), precision, and recall were used to measure the performance of the YOLOv5 object detection model in this study. The mAP metric measures the average precision across various levels of recall, while precision and recall measure how well the model detects correct objects without producing too many errors.

In this study, the YOLOv5 model was trained using an image dataset of catfish fry obtained from the public repository Roboflow. The training process was executed in the Google Colab Pro+ cloud computing environment for 50 epochs using the PyTorch library. During training, the model's performance on the validation dataset was monitored regularly. Fig. 3 shows the performance metrics of the best model achieved, with a precision value of 0.917, a recall of 0.96, and an mAP@0.5 of 0.966, indicating the model's excellent generalization capability on previously unseen data.

2.3. Implementation on the Android platform

After training was completed, the best model was exported to the TorchScript (.pt) format, which is optimized for inference on various platforms, including mobile devices. The Android application was then developed using the Flutter framework

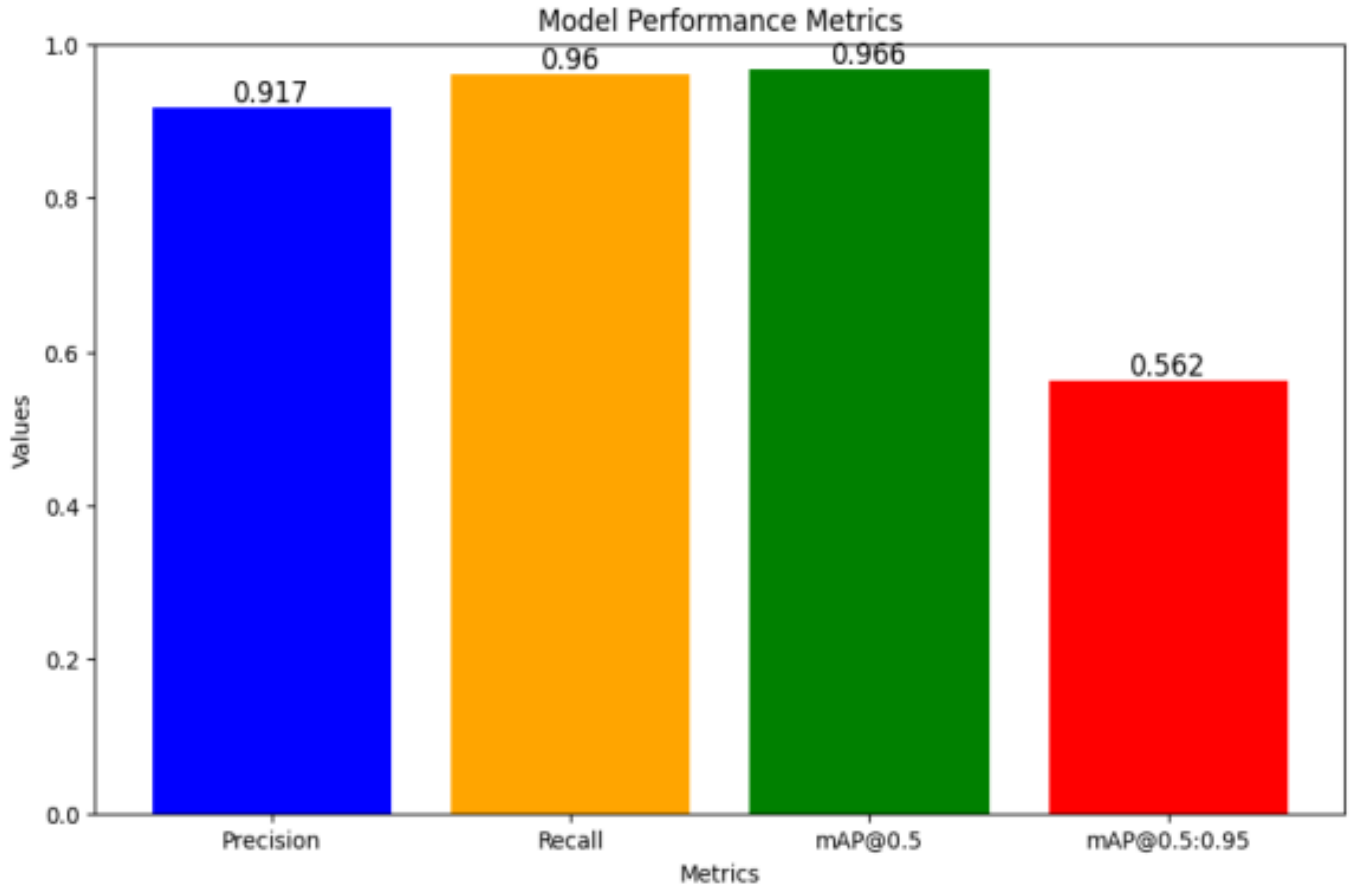

Fig. 3. Performance metrics of the best model after 50 epochs

because of its ability to build a responsive user interface from a single codebase. The model integration into the application was facilitated by the flutter_pytorch plugin, which provides a bridge to load the model and run inference directly on the device's hardware. This architecture allows the entire workflow—from image capture (via the camera or gallery), the inference process by the model, to the visualization of detection results—to run completely locally (on-device). As a result, the application can function without an internet connection and preserves user data privacy. Fig. 4 displays the application's user interface, from the main screen to the detection results screen.

The model file was placed in the assets/models/ directory of the Flutter project, along with the label file in assets/labels/. During initialization, the model is loaded using the FlutterPytorch.loadObjectDetectionModel() function with parameters for the model path, number of classes, input size (640×640), and the label file. This process is performed once when the application is initiated (main.dart). Subsequently, the application offers two data source options: the camera (camera_screen.dart) and the gallery (gallery_screen.dart). If the user selects the camera, each frame is captured using the camera plugin. If the gallery is chosen, an image is selected via the image_picker plugin.

The image obtained from the camera or gallery is sent to the model through the getImagePrediction() function. The output is a list of detected objects, containing bounding box coordinates and class labels. The detection results are visualized on the application interface by drawing bounding boxes on the image and displaying the number of identified fish fry.

Table 2. Detailed Test Results of Catfish Fry Detection

No.	Fry Count	Condition	Response (ms)	TP	FN	FP
1	1	normal	2450	1	0	0
2	2	normal	2545	2	0	0
3	3	normal	2235	3	0	0
4	4	normal	1649	4	0	0
5	4	clustered	1692	3	1	0
6	5	clustered	1592	5	0	0
7	5	normal	1657	5	0	0
8	6	normal	2291	3	3	0
9	6	normal	1647	5	1	1
10	6	normal	1675	6	0	0
11	7	normal	1823	7	0	0
12	7	Shadow	1965	4	3	3
13	7	Shadow & Clustered	2124	7	0	0
14	10	clustered	1990	7	3	0
15	10	normal	1858	10	0	0
16	10	clustered	2405	8	2	0
17	10	clustered	2184	8	1	1

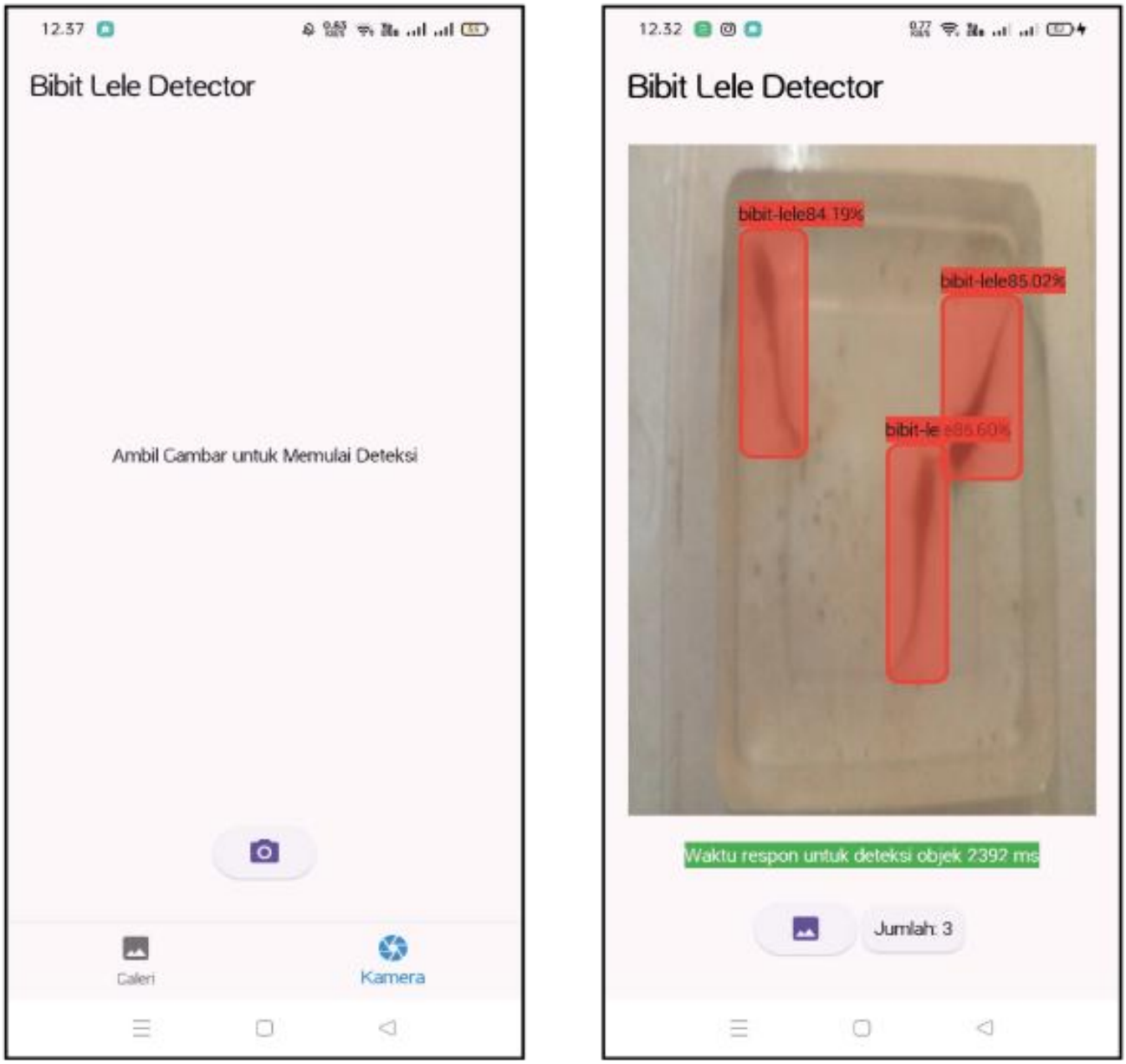

Fig. 4. User interface (UI) of the catfish fry detection application

2.4. *Testing Scenarios and Evaluation Metrics*

To evaluate the application's performance in an environment that approximates real-world conditions, testing was conducted on 17 images that had not been used in either training or validation. These images were selected to represent three primary scenarios frequently encountered in the field:

a. Normal: Good and even lighting, with fry well-distributed and no overlapping.
b. Clustered: Several fry are close together and overlapping, causing partial occlusion.
c. Shadow: Significant shadows are cast over the fry or the surrounding area.

Quantitative performance was measured using standard metrics in object detection evaluation: Precision, Recall, and F1-Score.

a. Precision = TP / (TP + FP), which measures the accuracy of the detections made.
b. Recall = TP / (TP + FN), which measures the completeness of the detections.
c. F1-Score = 2 * (Precision * Recall) / (Precision + Recall), which is the harmonic mean of Precision and Recall.

Where TP (True Positive) is the number of fry correctly detected, FP (False Positive) is an erroneous detection, and FN (False Negative) is the number of existing fry that were missed

Fig. 5. Input image with 10 catfish fry

Fig. 6. Detection result on the application (Count: 10)

Table 3. Summary of Model Performance per Condition

Condition	*Precision*	*Recall*	F1-Score	Brief Qualitative Note
Normal	0.979	0.920	0.949	Very good performance; accurate detection on clear images.
Clustered	0.969	0.795	0.874	Recall decreases (missed detections) due to occlusion (high FN).
Shadow	0.786	0.786	0.786	Precision decreases (high false positives).

by the detector.

3. Results

This section presents the findings from the application testing. A visual example of the detection process is shown in Fig. 5 and Fig. 6. In this case, Fig. 5 is the original input image containing 10 catfish fry. Fig. 6 shows the application's output, where the model successfully detected all 10 fry.

The comprehensive test results are detailed in Table 2, which includes the actual number of fry, the environmental condition, and the TP, FN, and FP values for each test image. The data from Table 2 were then aggregated by condition to calculate the overall performance metrics. Table 3 summarizes these calculations.

4. Discussion

The test results clearly indicate that the model's performance is highly dependent on the visual conditions of the input image.

Under normal conditions, the model demonstrated very satisfactory performance, achieving an F1-Score of 0.949. As seen in Table 3, the high Precision value (0.979) signifies that nearly all objects detected were indeed catfish fry. Meanwhile,

the high Recall value (0.920) indicates that the model successfully found most of the fry present. This confirms that under ideal conditions, the model can function as a reliable counting tool.

A significant challenge arose under clustered conditions, where the F1-Score dropped to 0.874. This decline was specifically caused by the drop in the Recall value to 0.795. This means the model failed to locate a number of fry that were present (an increase in False Negatives). This phenomenon occurs because several fry overlap one another, causing the visual features of each individual to become ambiguous.

The most challenging condition was the shadow condition, where the F1-Score dropped sharply to 0.786. This decline was primarily driven by the low Precision value of 0.786, indicating that the model generated many incorrect detections (an increase in False Positives). It is probable that dark areas or high-contrast regions created by shadows were mistakenly interpreted by the model as features similar to those of catfish fry.

It should be noted that there was an anomalous result in the test with the combined "shadow and clustered" condition (Table 2, row 13), where the model achieved perfect detection. This result contrasts with the model's lower performance under simpler, individual conditions. It is suspected that the specific image used for this test had unique characteristics that did not represent the true challenge of the combined conditions. Therefore, this result is considered an outlier.

5. Conclusion

This research successfully designed and evaluated an Android application based on YOLOv5 for the automated counting of catfish fry. The application was proven to have excellent performance under ideal conditions (F1-Score of 0.949). However, the study also identified two major weaknesses. First, its performance degraded under clustered conditions due to occlusion, which caused a decrease in Recall. Second, its performance dropped sharply under shadowed conditions, which caused a decrease in Precision due to false positive detections.

Based on these findings, future development is recommended to focus on several strategic areas. First, enriching the training dataset is a priority. This can be done by collecting more image data that represent difficult conditions and using data augmentation techniques. Second, further research could explore image pre-processing methods, such as lighting normalization or contrast enhancement. Finally, testing on a larger and more standardized dataset is necessary to validate the model's generalization and reliability before wider implementation.

Data Availability

All data generated or analyzed during the study are available within this article.

Declaration of Conflict of Interest

The authors declare that they have no known conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

Author Contributions

All authors designed the article, contributed to content writing, and revised the manuscript. MZ was the lead author of the first draft of the manuscript, and MSZ revised the manuscript. All authors read and approved the final version of the manuscript.

References

Arganata, A. R., Rasmana, S. T., & Kusumawati, W. I. (2020). *Analisis Perhitungan Bibit Ikan Gurame Menggunakan Webcam dengan Metode Yolo (You Only Look Once)*. https://repository.dinamika.ac.id/id/eprint/5195/

Armalivia, S., Zainuddin, M. S., & MT., P. D. I. A. (2021). *Penghitungan Otomatis Larva Udang Menggunakan Metode Yolo*. https://repository.unhas.ac.id/id/eprint/12479/

Badgujar, C. M., Poulose, A., & Gan, H. (2024). Agricultural Object Detection with You Look Only Once (YOLO) Algorithm: A Bibliometric and Systematic Literature Review. *ArXiv*. https://doi.org/10.48550/arXiv.2401.10379

Dendi, A. S., & Sunardi, S. (2021). *Alat Penghitung Benih Ikan Lele Menggunakan Pengolahan Citra*. http://repository.polman-babel.ac.id/id/eprint/350/

Putra, W. S. E. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, *5*(1). https://doi.org/10.12962/j23373539.v5i1.15696

Fauziyah, N., Nirmala, K., Supriyono, E., & Hadiroseyani, Y. (2019). Evaluasi Sistem Budidaya Lele: Aspek Produksi dan Strategi Pengembangannya (Studi Kasus: Pembudidaya Lele Kabupaten Tangerang). *Jurnal Kebijakan Sosial Ekonomi Kelautan Dan Perikanan*, *9*(2), 129. https://doi.org/10.15578/jksekp.v9i2.7764

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem dan Teknologi Informasi Indonesia)*, *3*(2), 49–56. https://doi.org/10.32528/justindo.v3i2.2254

Ciptawati, E., Budi Rachman, I., Oktiyani Rusdi, H., & Alvionita, M. (2021). Analisis Perbandingan Proses Pengolahan Ikan Lele terhadap Kadar Nutrisinya. *IJCA (Indonesian Journal of Chemical Analysis)*, *4*(1), 40–46. https://doi.org/10.20885/ijca.vol4.iss1.art5

Sitio, M. H. F., Jubaedah, D., & Syaifudin, M. (2017). Kelangsungan Hidup dan Pertumbuhan Benih Ikan Lele (Clarias sp.) Pada Salinitas Media yang Berbeda. *Jurnal Akuakultur Rawa Indonesia*, *5*(1), 83–96. https://doi.org/10.36706/jari.v5i1.5810

Sulistiyanti, S., Setyawan, F. A., & Komarudin, M. (2016). *Pengolahan Citra Dasar dan Contoh Penerapannya*. Teknosain. https://www.researchgate.net/profile/Muhamad-Komarudin/publication/337928062_PENGOLAHAN_CITRA_DASAR_DAN_CONTOH_PENERAPANNYA/links/67f71aa495231d5ba5bd8e5f/PENGOLAHAN-CITRA-DASAR-DAN-CONTOH-PENERAPANNYA.pdf

Yumna, A. S., Rukmono, D., Panjaitan, A. S., & Mulyono, M. (2019). Penigkatan Produktifitas Ikan Lele (Clarias sp.) Sistem Bioflok Di Pesantren Modern Darul Ma'arif Legok, Indramayu. *Jurnal Kelautan Dan Perikanan Terapan (JKPT)*, *2*(2), 113. https://doi.org/10.15578/jkpt.v2i2.8080

Mohammad Zainuddin received a Master's degree in Informatics Engineering from Dian Nuswantoro University, Semarang, in 2017. He currently serves as a Lecturer in the Informatics Engineering Study Program at the ASIA Institute of Technology and Business Malang.

Muhammad Saifuddin Zuhri. earned a Bachelor of Computer Science degree from the ASIA Institute of Technology and Business, Malang, in 2024. He is currently working as a Full-stack Developer at PT. Mulia Bagus Digital. His research interests include computer vision and mobile application development.