

Design and Implementation Submarine Cable Object Detection YOLOv4 based with Graphical User Interface (GUI) for Remotely Operated Vehicle (ROV)

Fikri Arif Wicaksana, Eueung Mulyana, Syarif Hidayat, Rahadian Yusuf
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia

Abstract—The use of submarine cables as underwater transmission channels for distributing electrical energy in Indonesian waters is crucial. However, the detection and maintenance of submarine cables still heavily rely on human observation, leading to limitations in time and subjective interpretations. This research aims to design and implement an underwater object detection system based on YOLOv4 integrated with a Graphical User Interface (GUI) on a Remotely Operated Vehicle (ROV) for submarine cable detection. The YOLOv4 model was trained using a balanced dataset, achieving performance with precision of 0.89, recall of 0.85, and f1-score of 0.87. Detection of Good Condition (SC-Good-Condition) achieved an Average Precision (AP) of 97.62%, while Bad Condition detection (SC-Bad-Condition) had an AP of 87.54%, resulting in an overall mAP of 92.58%. The implemented GUI successfully detected submarine cables in two test videos with FPS rates of 0.178 and 0.083. The designed underwater object detection system using YOLOv4 and GUI on ROV demonstrated satisfactory performance in detecting submarine cables. However, further efforts are needed to improve the GUI's FPS to make it more responsive and efficient. This research contributes to the development of underwater detection technology that supports environmental observation and electrical energy distribution in Indonesian waters.

Keywords—Submarine cable; object detection; GUI; ROV; YOLOv4

I. INTRODUCTION

Electricity is a fundamental and crucial necessity for the livelihood of Indonesian society. As Indonesia's economic growth and population continue to expand, the demand for electricity increases. Furthermore, being an archipelagic nation with 17,504 islands, almost all activities in both rural and urban areas of Indonesia require electricity. Therefore, there is a need for a transmission medium that can distribute electricity from one island to another. Submarine cables are underwater transmission channels that can distribute electricity between Indonesian islands [1]. However, in practice, submarine cables require periodic maintenance to ensure their optimal condition. One stage in the process of maintaining submarine cables is underwater inspection to observe the surrounding environment of the cables.

A Remotely Operated Vehicle (ROV) is an underwater robot that can be controlled remotely [2]. Some ROVs are

equipped with computer vision technology-enabled cameras for underwater inspection purposes, including maintenance and observation in the vicinity of submarine cables [3]. Typically, the process of detecting submarine cables is visually performed by ROV operators who oversee the video feed from the camera mounted on the ROV. However, this approach has some limitations, such as dependency on human observation, which is susceptible to fatigue and time constraints, as well as subjective interpretation in identifying submarine cables [4].

To address these limitations, automated object detection techniques have been developed, including underwater object detection. One method that has shown good performance is YOLOv4 (You Only Look Once version 4), which enables fast and accurate object detection [5]-[7].

Additionally, implementing a Graphical User Interface (GUI) on the ROV can provide an intuitive and user-friendly interface for operators, facilitating cable observation and detection with higher efficiency [8]. However, despite several studies on underwater object detection using YOLOv4 and research on GUI implementation on ROVs, research specifically combining both aspects in the context of submarine cable detection is still limited.

Therefore, this research aims to fill this knowledge gap by designing and implementing an underwater object detection system based on YOLOv4, integrated with a GUI on the ROV, particularly in the context of submarine cable detection. This study is expected to enhance the effectiveness and efficiency of submarine cable detection more accurately and efficiently.

Several prior studies have explored submarine cable detection using various methods, including both Deep Learning and non-Deep Learning approaches, yielding reasonably accurate results.

One previous study employed CNN and YOLO model (YOLOv3) for submarine cable detection but lacked a desktop GUI application. The results were as follows: for the original image dataset, it achieved an Average Precision (AP) of 98.14%, an F1 Score of 95.79%, and an Average Time of 0.416 seconds. Meanwhile, after dataset enhancement, it achieved an AP of 98.95%, an F1 Score of 96.92%, and an Average Time of 0.452 seconds [9].

Another research used edge detection methods to detect submarine cables, utilizing the Hough Transform model. This study developed software for ROVs equipped with cameras for cable detection. It reported successful submarine cable detection using 100 scenes, resulting in 83 correct detections, 17 false detections, and a recognition rate of 83% in the first experiment. In the second experiment, out of 100 scenes, it achieved 98 correct detections, 2 false detections, and a recognition rate of 98% [10][11].

This research focuses on submarine cable detection using the more recent CNN model YOLOv4, enhanced by the inclusion of a GUI for computer vision on the ROV. This integration of YOLOv4 and GUI aims to improve the quality of ROV-acquired computer vision data compared to previous studies, providing a more advanced and comprehensive approach to submarine cable detection.

In line with the outlined objectives, this paper is organized as follows. Section I begins with an exploration of the background of the research, emphasizing the vital role of electricity in Indonesian nation, the challenges posed by its archipelagic nature, and the need for advanced techniques in submarine cable detection. It further elucidates the specific goals and contributions of this research, aiming to bridge the existing gap in the integration of YOLOv4-based object detection and GUI on ROVs for submarine cable detection. Moreover, this section provides a concise review of prior research endeavors related to underwater object detection, ROV applications, and GUI implementations in the marine domain.

Section II delves into the research design, detailing the methodologies, equipment, and procedures employed in developing the underwater object detection system using YOLOv4 and integrating it with the ROV's GUI. It sheds light on the technical aspects and considerations pivotal to the success of this innovative system.

Section III is dedicated to presenting the experimental findings and their subsequent analysis. The section elucidates the outcomes of deploying the YOLOv4-based system and GUI on the ROV during underwater cable inspections. It discusses performance metrics and GUI functionality test.

Section IV, we synthesize the findings into comprehensive conclusions and offer recommendations for future research in this field. We reflect on the implications of our work on the broader context of submarine cable maintenance and its potential contributions to the sustainability and efficiency of Indonesia's electricity distribution network. Furthermore, we suggest avenues for further research and improvements to enhance the capabilities of the integrated system, ensuring its continued effectiveness in the evolving landscape of underwater cable detection.

II. RESEARCH DESIGN

A. Research Stages

There are several steps carried out in this research, as shown in Figure 1.

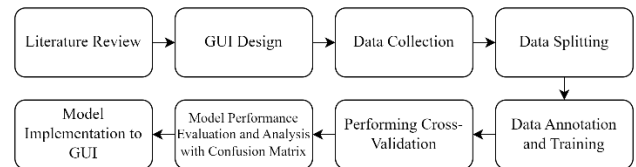


Fig. 1. Research stages.

- **Literature Review:** Conducting a comprehensive literature review on underwater object detection methods and techniques, the YOLOv4 approach, GUI utilization on ROVs, and related submarine cable research to understand the foundational theories and related studies.
- **GUI Design:** Designing and developing the Graphical User Interface (GUI) using PyQt (Python QML) library to enable interaction between the ROV operator and the YOLOv4-based underwater object detection system.
- **Data Collection:** Gathering the necessary image data for training and testing the submarine cable detection model. The image data used in this research was obtained from underwater inspection videos conducted by PT Syergie Indoprima in the year 2022.
- **Data Splitting:** Dividing the image data into two classes: SC-Good-Condition (good cable condition) and SC-Bad-Condition (bad cable condition). The entire image data will be divided into two subsets, with 80% of the data used for model training and 20% for model testing, both for imbalanced and balanced datasets.
- **Data Annotation and Training:** Annotating the image data with appropriate labels, i.e., SC-Good-Condition or SC-Bad-Condition, indicating the condition of the submarine cable in each image. Next, training the underwater object detection model using YOLOv4 with the annotated image data.
- **Performing Cross-Validation:** Conducting cross-validation on the trained model to measure its performance and accuracy in detecting submarine cables. The data is divided into 5 subsets (folds), and the model training and testing will be repeated on each fold to obtain more reliable results.
- **Model Performance Evaluation and Analysis with Confusion Matrix:** Using the cross-validation results, a confusion matrix is created to depict the overall model performance. The confusion matrix provides information on the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) in detecting submarine cables. Through the confusion matrix, the model's performance is analyzed and evaluated, including calculation of evaluation metrics such as precision, recall, and F1-score, to gain a deeper understanding of the model's ability to detect submarine cable conditions.

- **Model Implementation into GUI:** The trained and evaluated underwater object detection model is then implemented into the previously designed GUI application. This process involves a seamless integration between the model and GUI to enable real-time and non-real-time submarine cable detection through the ROV using a user-friendly GUI that provides clear and understandable results for the ROV operator.

B. GUI Design

Figure 2 is a mock-up of the GUI designed in this research. This GUI functions to display real-time and non-real-time computer vision camera views for detecting submarine cables.

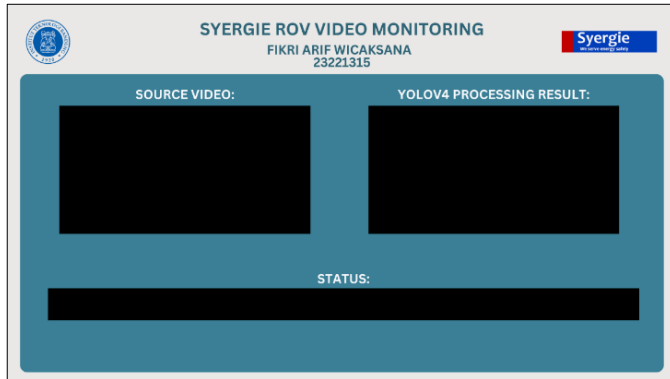


Fig. 2. Mock-up of ROV computer vision application GUI.

This GUI is equipped with 2 screen views, namely the "source video" screen to display real-time and non-real-time computer vision and the "YOLOv4 Processing Result" screen to show the submarine cable detection results from the trained YOLOv4 weights. Additionally, there is a status bar that displays the GUI duration, submarine cable detection class, AP (Average Precision) value, and bounding box dimensions.

C. Data Collection

The data used in this study was collected from underwater inspection video documentation by PT Syergie Indoprima. The documentation videos were converted into image frames using VLC Media Player. There are 4 underwater inspection videos used in this research, resulting in a total of 395 image frames. The data collection process is illustrated in Figure 3.

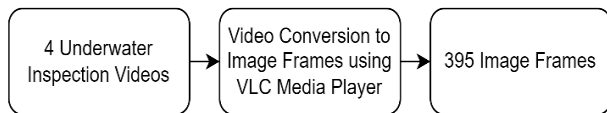


Fig. 3. Collecting data stages.

D. Data Splitting

After the data collection process, the data is divided into two classes: SC-Good-Condition and SC-Bad-Condition. This class division is based on the physical conditions observed in the underwater inspection videos that have been converted into 395 image frames. The following are some criteria for the submarine cable classes in this research, presented in Table I.

TABLE I. CRITERIA FOR SUBMARINE CABLE CLASSES

Criteria for SC-Good-Condition	Criteria for SC-Bad-Condition
The armor layer of the submarine cable is intact, with no peeling.	The armor layer of the submarine cable is peeling.
The submarine cable is not covered by underwater vegetation.	The submarine cable is covered by underwater vegetation [12]

Figure 4 is an example of SC-Good-Condition image, and Figure 5 is an example of SC-Bad-Condition image.



Fig. 4. Example of SC-good-condition image.

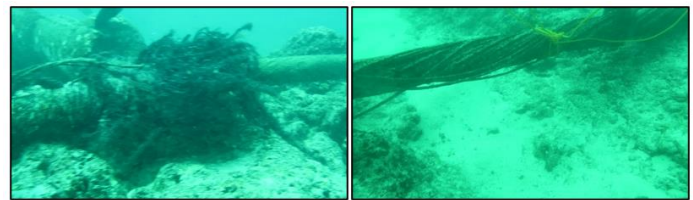


Fig. 5. Example of SC-Bad-condition image.

Out of the 395 submarine cable image frames obtained, the number of images for each class was imbalanced. Therefore, the image data was re-divided to create a balanced dataset for the research. The balanced dataset used is presented in Table II, and the data split between the training and testing sets is presented in Table III.

TABLE II. DATA IMAGE SPLIT INTO BALANCED DATASET

Class	Number of Images
SC-Good-Condition	100 Images
SC-Bad-Condition	100 Images

TABLE III. DATA IMAGE SPLIT BETWEEN TRAINING AND TESTING SETS

Number of Image Data	Training Data (80%)	Test Data (20%)
200 Images	160 Images	40 Images

Balanced and imbalanced datasets are important concepts in Deep Learning, including for object detection models like YOLOv4¹. When collecting image data, it is crucial to pay

¹Introduction to Balanced and Imbalanced Datasets in Machine Learning. (n.d.). Balanced and Imbalanced Datasets in Machine Learning [Full Introduction]. <https://encord.com/blog/an-introduction-to-balanced-and-imbalanced-datasets-in-machine-learning/>

attention to dataset balance, especially if there are differences in the number of examples between the SC-Good-Condition and SC-Bad-Condition classes. If the dataset is imbalanced, for instance, when there are fewer examples of bad submarine cables compared to good examples, the object detection model can become biased in learning relevant patterns and features of the bad submarine cables [13]. Therefore, it is necessary to perform proportional data splitting for training and testing the model effectively, enabling the model to accurately recognize both classes and avoid any undesired bias in submarine cable detection.

E. Data Annotation and Training

In this research, the tool LabelIm² was utilized for annotating the images used in submarine cable detection. This tool allows users to easily create annotations in the YOLOv4 format. It provides features to manually select and mark the positions and boundaries of submarine cables for both SC-Good-Condition and SC-Bad-Condition classes. The annotation process involves drawing bounding boxes around the cables in each image. These bounding boxes provide information about the coordinates (x, y) and dimensions (width and height) of the submarine cables.

For the training data process, custom configurations were used based on the number of classes being trained. A reference guide³ was employed for the configuration of submarine cable detection, as presented in Table IV for yolov4_train.cfg and Table V for yolov4_test.cfg.

TABLE IV. CONFIGURATION FOR YOLOV4_TRAIN.CFG

Reference	Used Configuration
$Filters = (number\ of\ class + 5) \times B$ B = number of bounding boxes	$Filters = (2 + 5) \times 3$ $Filters = 21$
$Max\ batches =$ $number\ of\ classes \times 2000$	$Max\ batches = 2 \times 2000$ $Max\ batches = 4000$
$Steps = 80\%, 90\% \text{ of } max\ batches$	$Steps = 80\%, 90\% \text{ of } 4000$ $Steps = 3200, 3600$
$Batch = 32$ $Subdivision = 16$	$Batch = 32$ $Subdivision = 16$

TABLE V. CONFIGURATION FOR YOLOV4_TEST.CFG

Reference [14]	Used Configuration
$Filters = (number\ of\ class + 5) \times B$ B = number of bounding boxes	$Filters = (2 + 5) \times 3$ $Filters = 21$
$Max\ batches =$ $number\ of\ classes \times 2000$	$Max\ batches = 2 \times 2000$ $Max\ batches = 4000$
$Steps = 80\%, 90\% \text{ of } max\ batches$	$Steps = 80\%, 90\% \text{ of } 4000$ $Steps = 3200, 3600$
$Batch = 1$ $Subdivision = 1$	$Batch = 1$ $Subdivision = 1$

²H. (2022, September 22). GitHub - heartexlabs/labelImg: LabelImg is now part of the Label Studio community. The popular image annotation tool created by Tzutalin is no longer actively being developed, but you can check out Label Studio, the open source data labeling tool for images, text, hypertext, audio, video and time-series data. GitHub. <https://github.com/heartexlabs/labelImg>

³A. (2023, June 20). GitHub - AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet). GitHub. <https://github.com/AlexeyAB/darknet>

- Filters: In the YOLOv4 configuration (cfg) file, the "filters" parameter refers to the number of filters needed in the last convolutional layer of the YOLOv4 network architecture. This number of filters is related to the number of object classes to be detected (plus 5), then multiplied by 3. $Filters = (Number\ of\ classes + 5) \times B$ indicates that for each object class to be detected, plus 5 (representing bounding box coordinates and confidence score), it will be multiplied by 3. The result is the total number of filters required in the last convolutional layer.
- Max batches: In the YOLOv4 configuration (cfg) file, the "max_batches" parameter determines the total number of iterations that will be used in the model training. Each iteration involves one batch of image data to train the model. The value of "max_batches" indicates the limit on the number of iterations to be performed during training. Therefore, from the formula $Max\ batches = Number\ of\ classes \times 2000$, it can be concluded that in this research, the maximum number of iterations for each training session is set to 4000 iterations.
- Steps: In the YOLOv4 configuration (cfg) file, the "steps" parameter is used to control when the learning rate will be adjusted during the training process. The "steps" value, expressed as a percentage of "max_batches," determines the points at which the learning rate will undergo changes. In this research, "steps" are set at 80% and 90% of "max_batches," which means there are two points where the learning rate will change during training: (1) At 80% of "max_batches": The learning rate will be adjusted when the training reaches 80% of the total scheduled iterations ("max_batches"). This adjustment usually involves decreasing the learning rate to help the model reach an optimal point during training. (2) At 90% of "max_batches": The learning rate will be adjusted when the training reaches 90% of the total scheduled iterations ("max_batches"). This adjustment typically involves further decreasing the learning rate to smooth the model's convergence process and improve the final results. The purpose of adjusting the learning rate is to optimize the training process and aid the model in achieving a good convergence, thereby enhancing object detection performance.
- Batch and subdivision: In the YOLOv4 configuration (cfg) file, the "batch" and "subdivisions" parameters are used to control how training or testing data is processed in each iteration. Here is an explanation for both values: (1) For the "yolov4_train.cfg": $Batch=32$: The "batch" value indicates the number of images processed in each training iteration. In this case, 32 images are processed simultaneously in one iteration. This means that 32 images are loaded into memory and used to update the model weights in one iteration. $Subdivisions=16$: The "subdivisions" value indicates how many weight updates will be performed before considering one iteration complete. In this case, every 16 weight updates will be performed before one

iteration is considered complete. This is useful to reduce memory load and speed up the training process. With this configuration, in each training iteration, 32 images are loaded and processed, and weight updates are performed every 16 times. This allows for YOLOv4 model training with efficiency and speeds up the training process. (2) For the "yolov4_test.cfg": Batch=1: The "batch" value is set to 1, which means that in each testing iteration, only 1 image will be processed. This is done because during the testing phase, we want to test each image individually to obtain accurate detection results and precise evaluation. Subdivisions=1: The "subdivisions" value is set to 1, which means that weight updates are not needed during the testing process. Each image is tested separately without any weight updates because no training is done at this stage. With this configuration, each image in the testing data will be tested separately, one by one, without any weight updates performed. This allows for accurate testing and precise evaluation of the previously trained model.

F. Performing Cross-Validation

In this research, cross-validation method is used to evaluate and validate the performance of the underwater object detection model. Cross-validation is a statistical method employed to assess and validate a model on a limited dataset⁴. The following are the steps of cross-validation conducted in this study:

- **Data Splitting:** The dataset used is divided into several subsets called "folds". In this research, the dataset is divided into 5 folds, as shown in the data split in Table VI.
- **Cross-Validation Iterations:** The cross-validation is performed 5 times, where each iteration uses one fold as the testing data, and the remaining folds are used as the training data. For example, in the first iteration, fold 5 is used as the testing data, while folds 1 to 4 are used as the training data. In the second iteration, fold 4 is used as the testing data, and folds 5 and 1 to 3 are used as the training data, and so on.
- **Model Training:** In each iteration, the submarine cable detection model is trained using the designated training data. The training process is conducted using the predetermined techniques and parameters.
- **Model Testing:** After training the model in each iteration, the model is evaluated using separate testing data. The model's performance is measured using relevant evaluation metrics such as precision, recall, and F1-score.
- **Selecting the Best Iteration Result:** After completing the cross-validation iterations, the performance evaluation results of the model in each iteration are recorded. Then, the best iteration result is chosen based on evaluation metrics like precision, recall, and F1-

score. The best iteration result is selected as the final outcome representing the model's best performance.

TABLE VI. DATASET FOLD SPLITTING

Iteration	Fold 1 (20%)	Fold 2 (20%)	Fold 3 (20%)	Fold 4 (20%)	Fold 5 (20%)
Iteration 1	Train				Test
Iteration 2	Train			Test	Train
Iteration 3	Train		Test	Train	
Iteration 4	Train	Test	Train		
Iteration 5	Test	Train			

Through the cross-validation method, this research can obtain more stable and reliable estimations of the performance of the submarine cable detection model. By dividing the dataset into different subsets for training and testing, this study can objectively test the model on various data and identify its strengths and weaknesses.

G. Model Performance Evaluation and Analysis with Confusion Matrix

This stage involves using the confusion matrix to evaluate and analyze the performance of the submarine cable detection model. The following are the steps involved:

- **Building the Confusion Matrix:** Using the testing data, the submarine cable detection model will make predictions for the SC-Good-Condition and SC-Bad-Condition object classes. From the prediction results and the ground truth labels, the confusion matrix will be constructed. The confusion matrix is a table with four cells representing the number of correct and incorrect predictions for each object class. The cells in the confusion matrix include True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [14]. The visualization of the confusion matrix is shown in Figure 6.
- **Based on the visualization of the confusion matrix above, in the context of submarine cable detection with classes SC-Good-Condition and SC-Bad-Condition, the definitions of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) can be stated as follows:**
 - **True Positive (TP):** TP occurs when the model correctly detects a submarine cable in good condition (SC-Good-Condition). This means that the model predicts correctly that the example belongs to the SC-Good-Condition class and indeed represents a submarine cable in good condition.
 - **True Negative (TN):** TN occurs when the model correctly detects a submarine cable in bad condition (SC-Bad-Condition). This means that the model predicts correctly that the example belongs to the SC-Bad-Condition class and indeed represents a submarine cable in bad condition.

⁴Cross Validation: Teknik Evaluasi Machine Learning, 6 Metode. (2022, August 14). Digital Polar. <https://digitalpolar.com/cross-validation/>

- False Positive (FP): FP occurs when the model incorrectly detects a submarine cable in bad condition (SC-Bad-Condition) as a submarine cable in good condition (SC-Good-Condition). This means that the model mistakenly predicts that the example belongs to the SC-Good-Condition class, whereas it actually belongs to the SC-Bad-Condition class.
- False Negative (FN): FN occurs when the model incorrectly detects a submarine cable in good condition (SC-Good-Condition) as a submarine cable in bad condition (SC-Bad-Condition). This means that the model mistakenly predicts that the example belongs to the SC-Bad-Condition class, whereas it actually belongs to the SC-Good-Condition class.

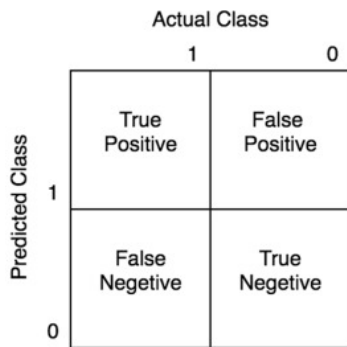


Fig. 6. Confusion matrix visualization [15].

- Calculating Evaluation Metrics: Based on the confusion matrix, various performance evaluation metrics for the model can be calculated, such as precision, recall, and F1-score. Precision measures the extent to which the model's positive predictions are correct, while recall measures the extent to which the model can correctly identify positive objects. F1-score is a combined measure that takes into account both precision and recall to provide a balanced performance overview⁵. The calculation algorithms for each metric are presented in Table VII.

TABLE VII. ALGORITHM FOR CALCULATING EVALUATION METRICS

Metric	Calculation Algorithm
Precision	$\frac{TP}{(TP + FP)}$ (1)
Recall	$\frac{TP}{TP + FN}$ (2)
F1-score	$\frac{2 \times (Precision \times Recall)}{(Precision + Recall)}$ (3)
mAP	$\frac{1}{N} \sum_{i=1}^N AP_i$ (4) Note: N = Number of AP data AP = Average Precision

⁵Kumar, A. (2023, March 17). Accuracy, Precision, Recall & F1-score - Python Examples - Data Analytics. Data Analytics. <https://vitalflux.com/accuracy-precision-recall-f1-score-python-example>

- Interpretation of Results: Through the confusion matrix and calculated evaluation metrics, the performance results of the model can be interpreted. Analyzing the number of TP, TN, FP, and FN for each object class provides insights into the model's ability to detect submarine cables in both SC-Good-Condition and SC-Bad-Condition classes. Observing the values of precision, recall, and F1-score for each object class helps in understanding the strengths and weaknesses of the model in detecting submarine cables.
- Visualization of Confusion Matrix: To facilitate understanding, the confusion matrix can be visualized using graphs or heat maps. This helps to clearly visualize the distribution of prediction results and errors that occur across all object classes.
- Testing Submarine Cable Detection on 20 Image Samples: To ensure that the model can accurately detect submarine cable conditions, a test for submarine cable detection is performed on 20 images listed in Table VIII, which are then analyzed for their results.

TABLE VIII. DETAILS OF THE SUBMARINE CABLE DETECTION TEST IMAGES

File Name	Class
Sample01.png – Sample10.png	SC-Good-Condition
Sample11.png – Sample20.png	SC-Bad-Condition

Through the evaluation and performance analysis of the model using the confusion matrix, this research provides detailed insights into how the submarine cable detection model operates, the extent of errors that occur, and the model's performance across all object classes. This aids in understanding and reporting the model's performance more accurately and informatively.

H. Model Implementation to GUI

This stage involves integrating the object detection model for underwater objects into the previously designed GUI. The following are the steps involved in this implementation:

- Model Preparation: The trained and tested submarine cable detection model (weights) will be prepared for integration into the GUI.
- Integration with GUI Library and Framework: The model will be integrated with the GUI library and framework used in this research, which is PyQt (Python QML).
- Functional Testing: After the integration is complete, functional testing of the GUI will be conducted to ensure that the submarine cable detection model operates smoothly within the GUI. At this stage, non-real-time submarine cable detection will be tested on several videos, and the details of these test videos are presented in Table IX.
- Evaluation and Refinement: After testing the functionality, the performance of the GUI and the submarine cable detection model within the GUI

environment is evaluated. Several aspects that can be evaluated include the mAP of the detection results according to equation (4) and also the calculation of the GUI's Frames per Second (FPS) performance, which can be calculated using the following equation:

$$FPS = \frac{\text{Number of Detected SC Frames}}{\text{GUI Processing Time}} \quad (5)$$

TABLE IX. DETAIL OF SUBMARINE CABLE DETECTION TEST VIDEOS ON GUI.

File Name	Video Duration
SampelVideo1.mov	54 seconds
SampleVideo2.mov	60 seconds

Subsequently, areas that need improvement or enhancement can be identified, and refinements can be made as necessary.

By integrating the submarine cable detection model into the GUI, this research provides an intuitive and interactive interface for users to perform submarine cable detection practically and efficiently. The GUI facilitates both real-time and non-real-time usage of the model and streamlines the interpretation of detection results within a more structured environment.

III. RESULT AND ANALYSIS

In this chapter, the results of the experiments, evaluation, and performance analysis of the model, as well as the functionality test of the GUI, are explained in accordance with the previously described design.

A. Experiment using Imbalanced Dataset

A total of 16 training experiments were conducted using the imbalanced dataset, wherein four datasets with varying numbers of images were used. The number of images in the SC-Good-Condition class was smaller compared to the number of images in the SC-Bad-Condition class, with a ratio of 40:60. The details of the image distribution for the imbalanced dataset are presented in Table X and Table XI.

From the division of the imbalanced dataset, training was conducted with varying numbers of iterations. Each dataset was trained using 1000, 2000, 3000, and 4000 iterations. The training results produced 16 metric outcomes, presented in Table XII and visualized in the graph in Figure 7.

TABLE X. THE NUMBER OF IMAGES IN THE SC-GOOD-CONDITION CLASS AND THE SC-BAD-CONDITION CLASS USED IN THE IMBALANCED DATASET

Dataset Name	Number of Images Data	SC-Good-Condition Class (40%)	SC-Bad-Condition Class (60%)
Dataset 1	395 Images	158 Images	237 Images
Dataset 2	790 Images	316 Images	474 Images
Dataset 3	1185 Images	474 Images	711 Images
Dataset 4	1580 Images	632 Images	948 Images

TABLE XI. COMPOSITION OF IMBALANCED DATASET

Dataset Name	Original Images	Rotated Images 90°	Rotated Images 180°	Rotated Images 270°	Total Number of Image Data
Dataset 1	395 Images	0 Images	0 Images	0 Images	395 Images
Dataset 2	395 Images	395 Images	0 Images	0 Images	790 Images
Dataset 3	395 Images	395 Images	395 Images	0 Images	1185 Images
Dataset 4	395 Images	395 Images	395 Images	395 Images	1580 Images

TABLE XII. TRAINING RESULTS USING THE IMBALANCED DATASET

Name of Weight	Precision	Recall	F1-score	mAP @0.5
yolov4_imb_395_1000	0.69	0.58	0.63	64.62%
yolov4_imb_395_2000	0.87	0.82	0.85	87.67%
yolov4_imb_395_3000	0.87	0.83	0.85	87.37%
yolov4_imb_395_4000	0.87	0.83	0.85	87.67%
yolov4_imb_790_1000	0.71	0.14	0.24	45.04%
yolov4_imb_790_2000	0.72	0.59	0.64	49.95%
yolov4_imb_790_3000	0.76	0.65	0.70	68.58%
yolov4_imb_790_4000	0.82	0.64	0.72	70.98%
yolov4_imb_1185_1000	0.62	0.06	0.11	18.86%
yolov4_imb_1185_2000	0.70	0.26	0.38	33.15%
yolov4_imb_1185_3000	0.70	0.24	0.36	31.84%
yolov4_imb_1185_4000	0.69	0.23	0.35	30.97%
yolov4_imb_1580_1000	0.54	0.05	0.09	17.18%
yolov4_imb_1580_2000	0.65	0.29	0.40	37.52%
yolov4_imb_1580_3000	0.62	0.29	0.40	38.12%
yolov4_imb_1580_4000	0.69	0.27	0.38	38.63%

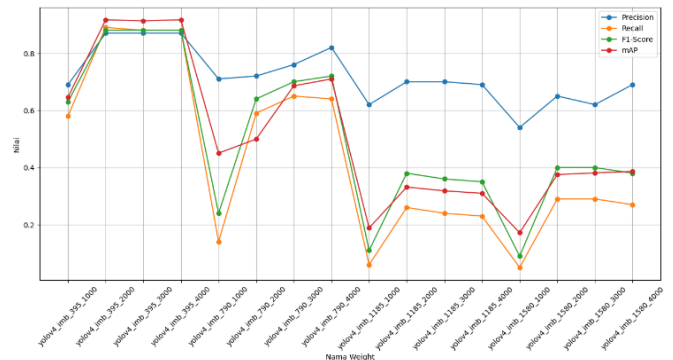


Fig. 7. Graph of evaluation metrics from training using the imbalanced dataset.

From Table XII and Figure 7, it can be observed that there is variation in the performance metrics across different datasets and iterations. It is evident that increasing the number of images in the dataset and the number of iterations does not always result in consistent improvement in the measured metrics.

For example, in the mAP (mean Average Precision) column, it can be seen that some combinations have lower

values compared to others, even when using a higher number of images and iterations. For instance, in the "yolov4_imb_395" dataset, the mAP for 4000 iterations (87.67%) does not show a significant improvement compared to 2000 iterations (87.67%). This indicates that after reaching a certain threshold of images and iterations, further increases do not provide significant benefits in the measured metrics.

This highlights that performance improvement is not solely dependent on increasing the number of data and iterations. There are other factors that need to be considered, such as data quality, balance of data between classes, and algorithmic factors used in model training. In this experiment with the imbalanced dataset, the best metric results were achieved by the "yolov4_imb_395_4000" weight.

B. Experiment using Balanced Dataset

A total of 20 training experiments were conducted with the balanced dataset using the cross-validation method. Among these experiments, the training process involved 5 iterations of cross-validation, with 5 fold data images for both training and testing data, as indicated in Table VI. Each cross-validation iteration utilized a different number of training iterations, similar to the training of the imbalanced dataset, which included 1000, 2000, 3000, and 4000 training iterations.

The number of images in the SC-Good-Condition class was balanced with the number of images in the SC-Bad-Condition class, with a 1:1 ratio. The details of image distribution for the balanced dataset are provided in Table II and Table III, as discussed previously. The training results from the balanced dataset yielded 20 sets of performance metrics, presented in Table XIII, and visualized in Figure 8.

TABLE XIII. TRAINING RESULTS USING THE BALANCED DATASET

Name of Weight	Precision	Recall	F1-score	mAP @0.5
yolov4_blc_itr1_1000	0.62	0.32	0.42	51.98%
yolov4_blc_itr1_2000	0.70	0.68	0.69	64.54%
yolov4_blc_itr1_3000	0.80	0.78	0.79	75.65%
yolov4_blc_itr1_4000	0.79	0.76	0.77	72.86%
yolov4_blc_itr2_1000	0.78	0.62	0.69	73.68%
yolov4_blc_itr2_2000	0.86	0.80	0.83	86.06%
yolov4_blc_itr2_3000	0.81	0.73	0.76	78.44%
yolov4_blc_itr2_4000	0.89	0.85	0.87	92.58%
yolov4_blc_itr3_1000	0.55	0.45	0.49	51.28%
yolov4_blc_itr3_2000	0.91	0.77	0.84	77.01%
yolov4_blc_itr3_3000	0.84	0.77	0.81	77.08%
yolov4_blc_itr3_4000	0.89	0.77	0.83	77.53%
yolov4_blc_itr4_1000	0.55	0.38	0.45	36.53%
yolov4_blc_itr4_2000	0.70	0.55	0.61	49.87%
yolov4_blc_itr4_3000	0.53	0.38	0.44	34.20%
yolov4_blc_itr4_4000	0.75	0.57	0.65	61.09%
yolov4_blc_itr5_1000	0.47	0.35	0.40	40.90%
yolov4_blc_itr5_2000	0.59	0.47	0.53	47.13%
yolov4_blc_itr5_3000	0.72	0.57	0.64	59.87%
yolov4_blc_itr5_4000	0.82	0.68	0.74	72.25%

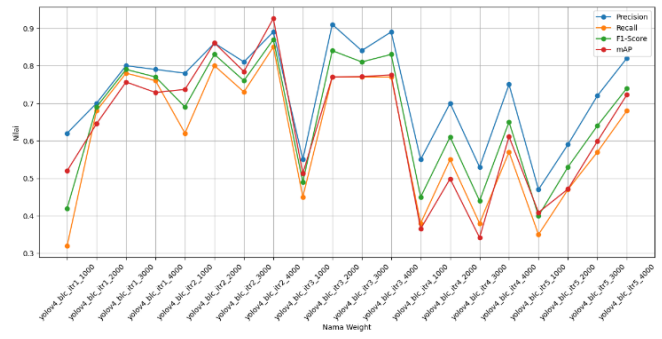


Fig. 8. Graph of evaluation metrics from training using the balanced dataset.

From the table and graph above, there is data on the performance metrics from various dataset combinations with different iterations during the model training. Here are some analyses based on the available data:

- **Differences in Dataset Combinations:** It is evident that each dataset combination exhibits different performances in the measured metrics. For example, if we observe the Precision column, some dataset combinations like "yolov4_blc_itr2_2000" (0.86) and "yolov4_blc_itr2_4000" (0.89) show higher precision values compared to other combinations.
- **Influence of Iterations:** In some cases, increasing the number of iterations consistently improves the measured metrics. For instance, if we consider the dataset combinations "yolov4_blc_itr1" and "yolov4_blc_itr2," it can be seen that higher numbers of iterations result in better performance in metrics like precision, recall, and F1-score.
- **Interrelation of Metrics:** In certain cases, there is a connection observed between the measured metrics. For example, the dataset combination "yolov4_blc_itr2_4000" exhibits higher values of precision (0.89), recall (0.85), and F1-score (0.87) compared to the dataset combination "yolov4_blc_itr2_1000" (precision: 0.78, recall: 0.62, F1-score: 0.69). This indicates that improvements in precision and recall also contribute to an increase in the F1-score.
- **Dependency on Dataset and Iterations:** The analysis reveals that performance improvement is not solely dependent on the number of iterations but also relies on the dataset combination used. For instance, in the dataset combination "yolov4_blc_itr3," increasing the number of iterations does not lead to significant improvements in the measured metrics, particularly in precision and recall.

This analysis shows that performance enhancement cannot be guaranteed solely through increasing the number of iterations. Additionally, other factors such as dataset quality, data variation, and parameter tuning should be considered to improve the overall model performance.

Subsequently, from each cross-validation iteration, the best-performing weight was selected, and the results were

summarized in the cross-validation results table shown in Table XIV and visualized in Figure 9.

TABLE XIV. CROSS-VALIDATION RESULT

Iteration	Precision	Recall	F1-score	mAP @0.5
Iteration 1	0.80	0.78	0.79	75.65%
Iteration 2	0.89	0.85	0.87	92.58%
Iteration 3	0.89	0.77	0.83	77.53%
Iteration 4	0.75	0.57	0.65	61.09%
Iteration 5	0.82	0.68	0.74	72.25%

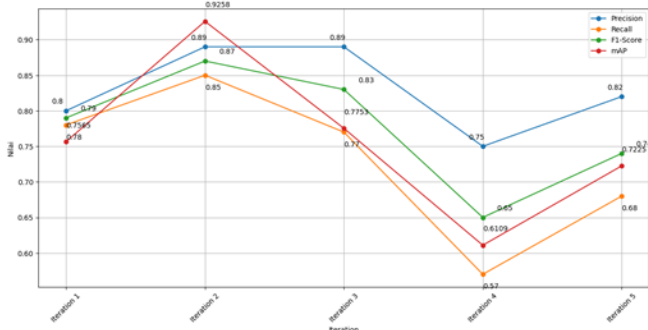


Fig. 9. Graph of evaluation matrix from cross-validation result.

From the cross-validation results above, several analysis can be drawn as follows:

- Precision: Precision measures how accurately the model classifies the positive class. The average precision from five iterations is 0.83, with the highest value reaching 0.89 in the second iteration. This indicates that the model tends to perform well in identifying the positive class.
- Recall: Recall measures how well the model recognizes instances of the positive class. The average recall from five iterations is 0.73, with the highest value reaching 0.85 in the second iteration. Although the average recall is relatively high, it should be noted that there is variation in recall values between different iterations.
- F1-score: The F1-score is a combined measure of precision and recall. The average F1-score from five iterations is 0.76, indicating a balanced performance between precision and recall in classifying the positive class.
- mAP @0.5: The mAP (mean Average Precision) at threshold 0.5 is a commonly used evaluation metric in object detection tasks. The average mAP from five iterations is 76.62%, with the highest value reaching 92.58% in the second iteration. This shows that the model has a good ability to detect objects with confidence levels that meet this threshold.

Thus, based on the cross-validation results, the model demonstrates good performance in classifying the positive class with a relatively high level of accuracy. Consequently, we select the weight associated with the best metric, which is the

metric from iteration 2: "yolov4_blc_itr2_4000," for further analysis and comparison in our research.

C. Comparison of Imbalanced Dataset and Balanced Dataset Experiment

After conducting training experiments on both imbalanced and balanced datasets, the best-performing weights from each dataset were selected for comparison. The comparison of these weights is presented in Table XV and the graph in Figure 10.

TABLE XV. COMPARISON OF EVALUATION METRICS FOR TRAINING ON IMBALANCED AND BALANCED DATASETS

Name of Weight	Dataset Type	Precision	Recall	F1-score	mAP @0.5
yolov4_imb_395_4000	Imbalanced	0.87	0.83	0.85	0.8767
yolov4_blc_itr2_4000	Balanced	0.89	0.85	0.87	0.9258

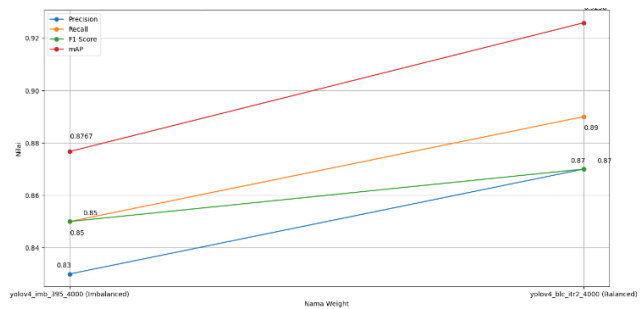


Fig. 10. Comparison graph of evaluation metrics for training on imbalanced and balanced datasets.

- Precision: The model with the balanced dataset has slightly higher precision (0.89) compared to the model with the imbalanced dataset (0.87). Precision measures how accurately the model classifies the positive class, and higher results indicate that the model with the balanced dataset is better at accurately recognizing the positive class.
- Recall: Recall in the model with the balanced dataset (0.85) is slightly lower than the model with the imbalanced dataset (0.83). Recall measures how well the model recognizes instances of the positive class. Although recall in the model with the balanced dataset is higher, the difference is not significant.
- F1-score: The model with the balanced dataset (0.87) has a higher F1-score compared to the model with the imbalanced dataset (0.85). F1-score is a measure of the harmonic mean between precision and recall, and the higher result in the model with the balanced dataset indicates better overall performance in classifying the positive class.
- mAP @0.5: The model with the balanced dataset has a higher mAP @0.5 (0.9258) compared to the model with the imbalanced dataset (0.8767). mAP @0.5 is a commonly used evaluation metric in object detection tasks, and the higher result in the model with the balanced dataset indicates better ability to detect objects with confidence levels that meet the threshold.

Based on the performance evaluation results in the table and graph above, the weight "yolov4_blc_itr2_4000" using the balanced dataset outperforms the weight "yolov4_imb_395_4000" using the imbalanced dataset in terms of precision, recall, F1-score, and mAP @0.5. The model with the balanced dataset has better ability to accurately classify the positive class, with higher F1-score and mAP @0.5.

In addition to better metric results, using the balanced dataset also provides several other advantages. By using a balanced dataset, we can avoid bias in the model as the dataset reflects an even distribution of data. Additionally, a balanced dataset can provide more balanced classification results and better control over prediction errors.

Therefore, based on the comparison results and the advantages obtained, the model with the balanced dataset ("yolov4_blc_itr2_4000") can be chosen as the best in classifying SC-Good-Condition and SC-Bad-Condition.

D. Model Performance Evaluation and Analysis

After selecting the "yolov4_blc_itr2_4000" weight using the balanced dataset as the best training result based on the metrics, the performance of the model will be further evaluated and analyzed using the confusion matrix.

From testing 40 validation images on the balanced dataset, a confusion matrix is constructed as a tool to analyze the trained classification model. Figure 11 displays the confusion matrix of the selected weight.

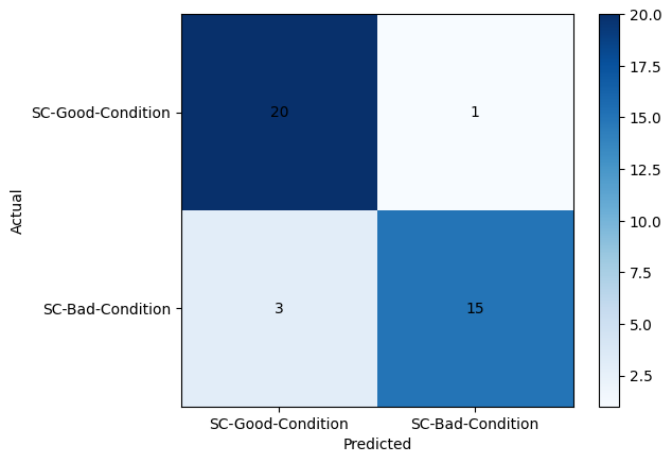


Fig. 11. Confusion matrix of the selected weight.

Based on the confusion matrix above, the model weight can be evaluated as follows:

- TP (True Positives): There are 20 images correctly detected as SC-Good Condition.
- TN (True Negatives): There are 15 images correctly detected as SC-Bad Condition.
- FP (False Positives): There is 1 image wrongly detected as SC-Bad Condition, whereas it should be SC-Good Condition.

- FN (False Negatives): There are 3 images misclassified as SC-Good Condition, whereas they should be SC-Bad Condition.

To improve the model's performance in handling false positives (FP) and false negatives (FN) cases, several additional improvements can be implemented:















- Increase the Amount of Data: Collecting more submarine cable images with a wider variation. Having a larger and more representative dataset allows the model to learn more complex patterns and enhance its detection capabilities for cases of false positives and false negatives.
- Further Data Augmentation: Perform data augmentation on submarine cable images with even broader variations, such as rotation, translation, zooming, cropping, and other distortions. This will help the model recognize various possible conditions in submarine cables and improve its adaptability.
- Hyperparameter Configuration Tuning: Perform hyperparameter tuning on the YOLOv4 model. Some hyperparameters that can be examined include learning rate, max_batches, batch size, subdivision size, input image size, and other parameters related to the YOLOv4 architecture.



E. Detection Test on Submarine Cable Image Samples

In this stage, a detection test is conducted on 20 sample submarine cable images, comprising 10 images with good condition and 10 images with bad condition. These images are outside of the balanced dataset used for training. The results of the detection test on the sample images are shown in Table XVI. Additionally, the Average Precision (AP) graphs for the SC-Good-Condition and SC-Bad-Condition classes are presented in Figure 12 and Figure 13, respectively.

TABLE XVI. RESULTS OF THE DETECTION TEST ON SUBMARINE CABLE IMAGE SAMPLES

File Name	Detection Image Result	Class	AP @0.5
Sample01.png		SC-Good-Condition	0.87
Sample02.png		SC-Good-Condition	0.94
Sample03.png		SC-Good-Condition	0.87
Sample04.png		SC-Good-Condition	0.82

Sample05.png		SC-Good-Condition	0.94
Sample06.png		SC-Good-Condition	0.91
Sample07.png		SC-Good-Condition	0.95
Sample08.png		SC-Good-Condition	0.82
Sample09.png		SC-Good-Condition	0.96
Sample10.png		SC-Good-Condition	0.97
Sample11.png		SC-Bad-Condition	0.82
Sample12.png		SC-Bad-Condition	0.90
Sample13.png		SC-Bad-Condition	0.94
Sample14.png		SC-Bad-Condition	0.83
Sample15.png		SC-Bad-Condition	0.87
Sample16.png		SC-Bad-Condition	0.88
Sample17.png		SC-Bad-Condition	0.66
Sample18.png		SC-Bad-Condition	0.98

Sample19.png		SC-Bad-Condition	0.96
Sample20.png		SC-Bad-Condition	0.91

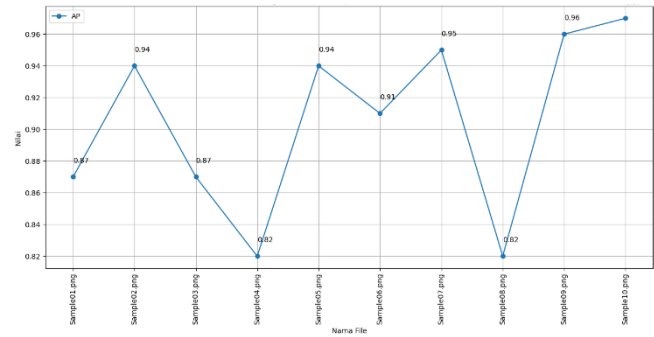


Fig. 12. Graph of average precision results for the detection test on SC-Good-Condition class image samples.

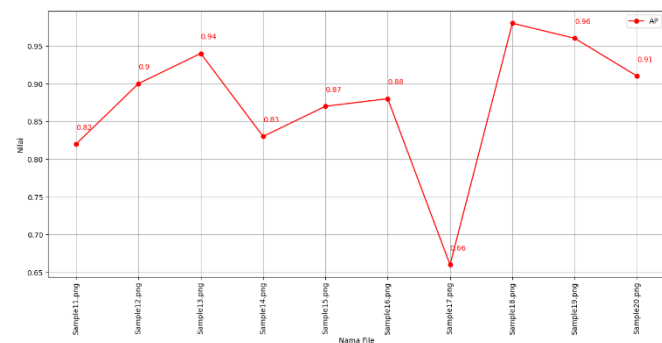


Fig. 13. Graph of Average Precision results for the detection test on SC-Bad-Condition class image samples.

Based on the table and graphs above, the AP (Average Precision) values indicate how well the model can detect and classify objects with high accuracy. The higher the AP value, the better the detection performance on those images. It can be observed that some SC-Good-Condition and SC-Bad-Condition images have high AP values, such as Sample10.png (SC-Good-Condition) with AP 0.97 and Sample18.png (SC-Bad-Condition) with AP 0.98. This indicates that the model has a good ability to detect and classify both conditions of submarine cables. There is variation in the detection performance among the submarine cable images. Some images achieve high AP values, showing accurate detection, while others obtain lower AP values, suggesting possible difficulties in detection for those images. Therefore, it is necessary to evaluate the causes of the low AP values and identify factors that may influence the detection results in those images. This may involve visual analysis and further investigation of the images to discover patterns or difficulties that the model may encounter in accurately classifying submarine cable images.

Here are some assumptions about the factors that may influence the detection results with low AP values on the submarine cable images:

- **Blurry or Unclear Image Quality:** Images with blurry or unclear quality can cause difficulties in detecting submarine cables. The lines or contours of the submarine cables may not be clearly visible, making it challenging for the model to classify accurately.
- **Inappropriate Image Scale or Size:** Image size that is either too small or too large can affect the model's ability to detect submarine cables accurately. Images that are too small may lead to the submarine cable object being too tiny to detect, while images that are too large may cause the details of the submarine cable object to be lost or distorted.
- **Variations in Lighting or Image Contrast:** Differences in lighting or contrast in submarine cable images can affect the model's ability to recognize and classify the submarine cables accurately. Low lighting or low contrast can make it difficult to see the details of the submarine cable object, leading to detection challenges.
- **Presence of Confusing Objects:** The presence of other objects that have visual similarities with submarine cables, such as other cables or structural elements, or the presence of other objects obstructing the submarine cable, such as fish, rocks, etc., can confuse the model and disrupt the process of accurate detection and classification.
- **Variations in Submarine Cable Shapes or Types:** Images in the dataset may have variations in the shape or type of submarine cables, which can pose challenges for the model in recognizing these variations. The model may struggle to understand the variations in texture, shape, or size of different types of submarine cables.

F. GUI Functionality Test

In this stage, the functionality of the GUI is tested by conducting non-real-time submarine cable detection on several videos presented in Table IX. The results of the GUI functionality test are presented in Table XVII and graph images in Figure 14 for the testing on the SampleVideo1.mov file.

TABLE XVII. RESULTS OF GUI FUNCTIONALITY TEST ON SAMPLEVIDEO1.MOV

Video Duration	GUI Processing Duration	Detected Class	AP Value @0.5
00:01	00:06	SC-Bad-Condition	0.75
00:01	00:11	SC-Bad-Condition	0.85
00:02	00:16	SC-Bad-Condition	0.89
00:03	00:20	SC-Bad-Condition	0.91
00:04	00:25	SC-Bad-Condition	0.97
00:05	00:29	SC-Bad-Condition	0.97

00:06	00:33	SC-Bad-Condition	0.98
00:06	00:38	SC-Bad-Condition	0.93
00:07	00:42	SC-Bad-Condition	0.95
00:08	00:47	SC-Bad-Condition	0.98
00:09	00:51	SC-Bad-Condition	0.98
00:09	00:56	SC-Bad-Condition	0.98
00:10	01:00	SC-Bad-Condition	0.99
00:11	01:04	SC-Bad-Condition	0.99
00:12	01:09	SC-Bad-Condition	0.97
00:12	01:14	SC-Bad-Condition	0.96
00:14	01:18	SC-Bad-Condition	0.81
00:14	01:22	SC-Bad-Condition	0.73
00:15	01:26	SC-Bad-Condition	0.95
00:16	01:31	SC-Bad-Condition	0.98
00:17	01:35	SC-Bad-Condition	0.91
00:18	01:39	SC-Bad-Condition	0.82
00:18	01:44	SC-Good-Condition	0.93
00:19	01:48	SC-Good-Condition	0.94
00:20	01:52	SC-Good-Condition	0.91
00:21	01:57	SC-Good-Condition	0.94
00:25	02:16	SC-Bad-Condition	0.89
00:26	02:20	SC-Bad-Condition	0.90
00:27	02:25	SC-Bad-Condition	0.50
00:28	02:29	SC-Bad-Condition	0.85
00:30	02:38	SC-Good-Condition	0.75
00:30	02:42	SC-Good-Condition	0.78
00:31	02:47	SC-Good-Condition	0.79
00:32	02:56	SC-Good-Condition	0.66
00:34	03:00	SC-Bad-Condition	0.73
00:34	03:05	SC-Good-Condition	0.54
00:39	03:24	SC-Bad-Condition	0.82
00:39	03:28	SC-Bad-Condition	0.85
00:41	03:33	SC-Bad-Condition	0.32
00:42	03:42	SC-Bad-Condition	0.76
00:44	03:51	SC-Bad-Condition	0.89
00:44	03:55	SC-Bad-Condition	0.85
00:45	04:00	SC-Bad-Condition	0.90
00:47	04:09	SC-Bad-Condition	0.75
00:47	04:13	SC-Bad-Condition	0.71
00:48	04:17	SC-Bad-Condition	0.79
00:48	04:22	SC-Bad-Condition	0.79
00:50	04:26	SC-Good-Condition	0.41
00:51	04:35	SC-Good-Condition	0.78
00:52	04:40	SC-Good-Condition	0.54

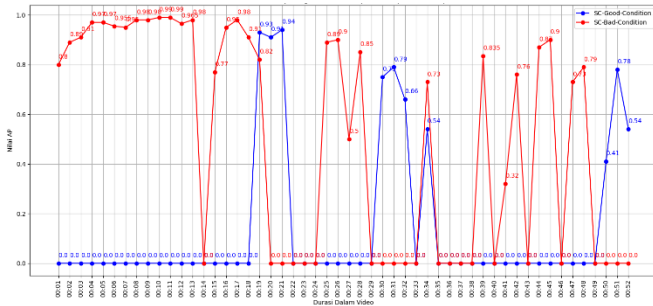


Fig. 14. Graph of GUI functionality test results on SampleVideo1.mov.

Based on the table and graph above, the analysis is as follows:

- The GUI successfully detected SC-Good-Condition at the video timestamps: 00:18, 00:19, 00:20, 00:21, 00:30, 00:31, 00:32, 00:34, 00:50, 00:51, and 00:52. The GUI successfully detected SC-Good-Condition 11 times out of a total of 50 detection frames.
- The GUI successfully detected SC-Bad-Condition at the video timestamps: 00:01, 00:02, 00:03, 00:04, 00:05, 00:06, 00:07, 00:08, 00:09, 00:10, 00:11, 00:12, 00:13, 00:15, 00:16, 00:17, 00:18, 00:19, 00:25, 00:26, 00:27, 00:28, 00:34, 00:39, 00:41, 00:42, 00:44, 00:45, 00:47, and 00:48. The GUI successfully detected SC-Bad-Condition 30 times out of a total of 50 detection frames.
- Several frames are at the same second, such as frames of SC-Bad-Condition at video timestamps: 00:01, 00:06, 00:09, 00:12, 00:14, 00:18, 00:34, 00:39, 00:44, 00:47, and 00:48, which were detected twice at each second. As for SC-Good-Condition frames, there is only one video timestamp (00:30) with two frames detected in that second.
- Several frames are at the same second but have different detection classes. At video timestamps: 00:18 and 00:34, each has 2 frames with different detection classes, one frame of SC-Good-Condition, and one frame of SC-Bad-Condition.
- To calculate the GUI's FPS (Frames per Second) performance for SampleVideo.mov, the formula in equation (V) is used:

$$FPS = \frac{50 \text{ frame}}{280 \text{ detik}} = 0.178 \text{ fps}$$

From the calculations above, the GUI performance for detecting SampleVideo1.mov still has a low FPS, taking a total time of 4 minutes and 40 seconds (280 seconds).

- To calculate the mAP (mean Average Precision) performance, the formula in equation (IV) is used. The results of mAP calculations for each class in the GUI functionality test on VideoSample01.mov are shown in Table XVIII.

TABLE XVIII. RESULTS OF MAP CALCULATION FOR GUI FUNCTIONALITY TEST ON VIDEOSAMPLE01.MOV

Class Name	mAP @0.5
SC-Good-Condition	0.725
SC-Bad-Condition	0.861

Based on the table above, the detection and classification results of submarine cables using the GUI have achieved a good mAP value.

Next, the functionality test continues for the video SampleVideo2.mov. The results of the GUI functionality test for SampleVideo2.mov are presented in Table XIX and graph images in Figure 15.

TABLE XIX. RESULTS OF GUI FUNCTIONALITY TEST ON SAMPLEVIDEO2.MOV

Video Duration	GUI Processing Duration	Detected Class	AP Value @0.5
00:02	00:07	SC-Good-Condition	0.48
00:02	00:12	SC-Good-Condition	0.95
00:02	00:16	SC-Good-Condition	0.56
00:03	00:20	SC-Good-Condition	0.79
00:04	00:24	SC-Good-Condition	0.35
00:11	00:53	SC-Good-Condition	0.45
00:11	00:58	SC-Good-Condition	0.43
00:11	01:02	SC-Good-Condition	0.92
00:12	01:06	SC-Good-Condition	0.86
00:13	01:11	SC-Good-Condition	0.51
00:14	01:15	SC-Good-Condition	0.49
00:14	01:19	SC-Good-Condition	0.29
00:20	01:35	SC-Good-Condition	0.27
00:30	02:31	SC-Good-Condition	0.56
00:38	03:08	SC-Bad-Condition	0.39
00:44	03:43	SC-Bad-Condition	0.58
00:50	04:09	SC-Bad-Condition	0.41
00:51	04:14	SC-Bad-Condition	0.72
00:51	04:18	SC-Bad-Condition	0.94
00:52	04:22	SC-Bad-Condition	0.98
00:53	04:27	SC-Bad-Condition	0.90
00:54	04:31	SC-Bad-Condition	0.94
00:54	04:36	SC-Bad-Condition	0.74

Based on the table and graph above, the analysis is as follows:

- The GUI successfully detected SC-Good-Condition at the video timestamps: 00:02, 00:03, 00:04, 00:11, 00:12, 00:13, 00:14, 00:20, and 00:30. The GUI successfully detected SC-Good-Condition 9 times out of a total of 23 detection frames.

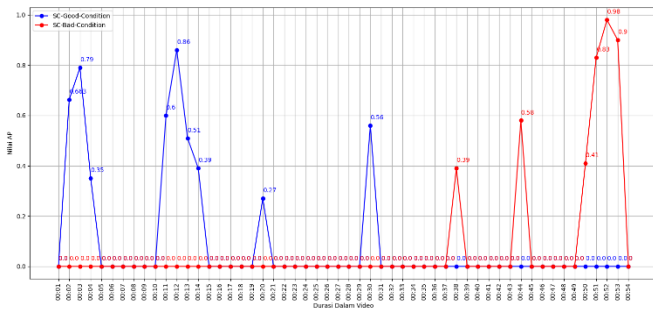


Fig. 15. Graph of GUI functionality test results on SampleVideo2.mov.

- The GUI successfully detected SC-Bad-Condition at the video timestamps: 00:38, 00:44, 00:50, 00:51, 00:52, and 00:54. The GUI successfully detected SC-Bad-Condition 6 times out of a total of 23 detection frames.
- Several frames are at the same second, such as SC-Bad-Condition frames at video timestamps: 00:51 and 00:54, which were detected twice at each second. SC-Good-Condition frames at video timestamps: 00:13 and 00:14 were detected three times in each second.
- To calculate the GUI's FPS (Frames per Second) performance for SampleVideo.mov, the formula in equation (V) is used:

$$FPS = \frac{23 \text{ frame}}{276 \text{ detik}} = 0.083 \text{ fps}$$

Based on the calculations above, the GUI performance for detecting SampleVideo2.mov still has a low FPS, taking a total time of 4 minutes and 36 seconds (276 seconds).

- From the testing of SampleVideo1.mov and SampleVideo2.mov, both GUI FPS performances are relatively low. The following are some assumptions that may cause the low FPS performance:
 - Model Complexity: The low FPS may be caused by the complexity of the YOLOv4 detection model used. Models with many layers and parameters may require longer processing time.
 - Computational Load: Object detection using the YOLOv4 model requires intensive processing and consumes a lot of computational power. This can reduce processing speed and result in low FPS.
 - Hardware Limitations: The use of less powerful hardware, such as a CPU with low computational power, can affect the GUI's performance and cause low FPS.
- To improve FPS and enhance the GUI's responsiveness in submarine cable detection, several efforts can be made:
 - Model Optimization: Optimize the YOLOv4 detection model by reducing the number of unnecessary layers or parameters and using a lighter model.

- Use More Powerful Hardware: Use GPUs with parallel processing capabilities to improve processing speed and FPS.
- Resolution Reduction: Reduce the video resolution or convert the video format to a lighter format to improve FPS.
- Data Streaming: Use data streaming techniques to process videos in real-time and enhance GUI responsiveness.

By implementing the above efforts, it is expected that FPS on the GUI can be increased, making the application more responsive and providing users with a better experience in submarine cable detection. Continuous and iterative evaluation is necessary to ensure optimal performance improvements.

- To calculate the mAP (mean Average Precision) performance, the formula in equation (IV) is used. The results of mAP calculations for each class in the GUI functionality test on VideoSample02.mov are shown in Table XX.

TABLE XX. RESULTS OF MAP CALCULATION FOR GUI FUNCTIONALITY TEST ON VIDEOSAMPLE02.MOV

Class Name	mAP @0.5
SC-Good-Condition	0.554
SC-Bad-Condition	0.681

G. Comparison of Functionality Test Performance

From the two functionality tests with two different videos, VideoSample01.mov and VideoSample02.mov, two performances were obtained for comparison. The comparison of functionality test performances is presented in Table XXI.

TABLE XXI. COMPARISON OF GUI FUNCTIONALITY TEST RESULTS FOR VIDEOSAMPLE01.MOV AND VIDEOSAMPLE02.MOV

File Name	Video Duration	GUI Processing Time	Number of SC Frame Detected	FPS	mAP @0.5
VideoSample01.mov	00:54	04:40	50 frames	0.178	0.829
VideoSample02.mov	01:00	04:36	23 frames	0.083	0.605

Overall, the GUI has performed well in detecting submarine cables in both videos. This can be seen from the relatively high mean Average Precision (mAP) @0.5 values, which are 0.829 for VideoSample01.mov and 0.605 for VideoSample02.mov. mAP @0.5 measures the object detection accuracy at an Intersection over Union (IoU) threshold of 0.5, and the higher the mAP value, the more accurate the detection results.

However, there are differences in the number of detected submarine cable frames between the two videos. In VideoSample01.mov, the GUI successfully detected 50 SC frames, while in VideoSample02.mov, the number of detected submarine cable frames was only 23. This difference is due to the varying video conditions between the two samples.

VideoSample02.mov has a more blurry, shaky, and often unfocused video quality, making submarine cable detection more challenging. This affects the GUI's performance in detecting SC in that video. Another factor that can affect the number of detected submarine cable frames is the complexity of the background and the presence of other objects that are similar to the submarine cable, causing some submarine cable frames to go undetected.

Regarding FPS, both VideoSample01.mov and VideoSample02.mov have low FPS values. This indicates that the GUI's processing is still relatively slow, resulting in slow detection. Increasing the FPS is one of the efforts that need to be made to improve the quality and efficiency of the GUI in detecting submarine cables.

IV. CONCLUSION

In this research, a YOLOv4-based underwater detection system integrated with a Graphical User Interface (GUI) for Remotely Operated Vehicle (ROV) in submarine cable detection has been successfully designed and implemented. Based on the experimental results and analysis, the following are the conclusions drawn from this research:

1) *Development of Submarine Cable Detection Model:* The performance evaluation of the model on the balanced dataset weight showed satisfactory results with precision of 0.89, recall of 0.85, F1-score of 0.87, and mAP of 92.58%. This indicates the model's ability to recognize both classes effectively.

2) *Implementation of Graphical User Interface (GUI):* The performance evaluation of the designed GUI showed promising results. In VideoSample01.mov, the GUI successfully detected 50 frames of submarine cable images with an mAP of 0.829 and GUI FPS of 0.0178. In VideoSample02.mov, the GUI detected 23 frames of submarine cable images with an mAP of 0.605 and GUI FPS of 0.083. The implementation of the GUI with the submarine cable detection model on the ROV successfully reduces dependency on human observation. With this automated system, issues of fatigue and subjective interpretation in identifying submarine cable conditions can be addressed. This provides the benefit of facilitating the submarine cable maintenance process.

In light of the successful development and implementation of the YOLOv4-based underwater detection system integrated with a GUI for ROV in submarine cable detection, there are several key areas for future research and improvements:

1) *Future work* should aim to enhance the scalability and adaptability of the system. This could involve expanding the dataset to encompass a broader range of underwater environments and conditions. Additionally, exploring the integration of machine learning techniques for automatic parameter tuning, especially in varying lighting and water clarity conditions, would further bolster the system's performance and reliability. Furthermore, the system could benefit from the incorporation of real-time anomaly detection

algorithms to promptly identify potential cable issues and facilitate proactive maintenance.

2) *There* is potential to extend the application of this technology to broader marine infrastructure management. Researchers can explore its utility in tasks beyond submarine cable detection, such as pipeline monitoring, marine biodiversity assessment, and archaeological exploration. Adapting the system for these diverse applications could significantly contribute to the advancement of marine sciences and industries. Additionally, research efforts should be directed toward refining the user interface and operator interaction aspects of the GUI to ensure user-friendliness and efficiency. This includes incorporating features that enable operators to annotate and validate detected cable segments, fostering human-machine collaboration for more accurate results. By addressing these areas in future research endeavors, we can further enhance the capabilities and impact of this innovative underwater detection system.

REFERENCES

- [1] M. Jamin and A. Sugiyono, "Pengembangan Kelistrikan Nasional."
- [2] Susianti, E., Syahputra, N. A., Wibowo, A. U., & Maria, P. S. (2021). Rancang Bangun Robot Observasi Bawah Air-ROV (Remotely Operated Vehicle) Menggunakan Arduino UNO. Jurnal Elektro dan Mesin Terapan, 7(2), 136-146.
- [3] Saputro, B. S., Djunarsjah, E., Setiyadi, J., & Negara, A. K. (2015). Pengoperasian Remotely Operated Vehicle (ROV) Mendukung Pekerjaan Bawah Air (Studi Kasus Pendeteksian Kabel Bawah Laut Menggunakan ROV H800 Di Perairan Selat Bangka Belitung): Remotely Operated Vehicle (ROV) Operation Supports Underwater Work (Case Study of Detecting Submarine cables Using ROV H800 in the Waters of the Bangka Belitung Strait). Jurnal Hidropilar, 1(2), 95-111.
- [4] Noyes, R. Y. (1994). Inspection methods for underwater cables (Doctoral dissertation, National Technical Information Service).
- [5] Zhang, M., Xu, S., Song, W., He, Q., & Wei, Q. (2021). Lightweight underwater object detection based on yolo v4 and multi-scale attentional feature fusion. Remote Sensing, 13(22), 4706.
- [6] Rosli, M. S. A. B., Isa, I. S., Maruzuki, M. I. F., Sulaiman, S. N., & Ahmad, I. (2021, August). Underwater animal detection using YOLOV4. In 2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE) (pp. 158-163). IEEE.
- [7] Zhang, C., Zhang, G., Li, H., Liu, H., Tan, J., & Xue, X. (2023). Underwater target detection algorithm based on improved YOLOv4 with SemiDSCov and FIoU loss function. Frontiers in Marine Science, 10, 1153416.
- [8] García-Valdovinos, L. G., Salgado-Jiménez, T., Bandala-Sánchez, M., Nava-Balanzar, L., Hernández-Alvarado, R., & Cruz-Ledesma, J. A. (2014). Modelling, design and robust control of a remotely operated underwater vehicle. International Journal of Advanced Robotic Systems, 11(1), 1.
- [9] Li, Y., Zhang, X., & Shen, Z. (2022). YOLO-Submarine cable: An Improved YOLO-V3 Network for Object Detection on Submarine cable Images. Journal of Marine Science and Engineering, 10(8), 1143.
- [10] Matsumoto, S., & Ito, Y. (1995, October). Real-time vision-based tracking of submarine-cables for AUV/ROV. In 'Challenges of Our Changing Global Environment'. Conference Proceedings. OCEANS'95 MTS/IEEE (Vol. 3, pp. 1997-2002). IEEE.
- [11] Fatan, M., Daliri, M. R., & Shahri, A. M. (2016). Underwater cable detection in the images using edge classification based on texture information. Measurement, 91, 309-317.
- [12] Burnett, D. R., Beckman, R., & Davenport, T. M. (Eds.). (2013). Submarine cables: the handbook of Law and Policy. Martinus Nijhoff Publishers.

- [13] Adesokan, A. A. (2021). Covid-19 Control: Face Mask Detection Using Deep Learning for Balanced and Unbalanced Dataset. Available at SSRN 4181373.
- [14] Confusion Matrix - an overview | ScienceDirect Topics. (n.d.). Confusion Matrix - an Overview | ScienceDirect Topics. <https://doi.org/10.1016/B978-0-12-818366-3.00005-8>
- [15] Sharma, D. K., Chatterjee, M., Kaur, G., & Vavilala, S. (2022). Deep Learning applications for disease diagnosis. In Deep Learning for medical applications with unique data (pp. 31-51). Academic Press.