Drowsiness Detection in the Advanced Driver-Assistance System using YOLO V5 Detection Model

Muhammad Fauzan Ridho[1], Fransiskus Panca[2], Welly Yandi[3], Almeera Amsana Rachmani[4]

[1,3,4] Electrical Engineering Department, Faculty of Engineering Universitas Bangka Belitung;
email: muhammadfauzanridho@ubb.ac.id, wellyyandi.koto@gmail.com, almeera_ar@ubb.ac.id
[2] Information Technology Department, Faculty of Engineering Universitas Bangka Belitung; email: fransiskuspj@ubb.ac.id

[Received: 18 February 2024, Revised: 17 May 2024, Accepted: 28 May 2024]
Corresponding Author: Muhammad Fauzan Ridho

ABSTRACT — The development of Artificial Intelligence (AI) in the automobile industry, particularly in the Advanced Driver-Assistance System (ADAS), has been increasing rapidly in recent years. One of the essential features of ADAS is the drowsiness alert system, which monitors the state of the driver's eyes. This article presents a study on the development of ADAS that focuses on drowsiness detection using deep learning through a Convolutional Neural Network (CNN) approach. The study utilized the YOLO V5 model for object detection, which was trained using custom datasets containing images annotated with two labels: "Mengantuk" (Drowsy) and "Bangun" (Awake) The goal was to recognize whether the driver was drowsy or awake. The results of the study showed that the modified YOLO V5 CNN Model had high accuracy based on evaluation metrics in terms of accuracy, precision, and recall in detecting drowsiness around the area of the eyes, with a Mean Average Precision (mAP) of 99.5% and an F1-Score of 99.8%. For better understanding and visualization, the model was tested using real-time detection through a web camera, using Jetson Nano as the inference device. The model detected drowsiness in real time, with a confidence rate of 80% to 97%

KEYWORDS — Driver-Assistance System, Drowsiness Detection,Convolutional Neural Network, YOLO V5, F1-Score, Mean Average Precision.

I. INTRODUCTION

One of the leading causes of accidents on highways is drivers feeling drowsy and falling asleep at the wheel. This phenomenon of feeling sleepy for a few seconds while driving is known as microsleep. It is particularly common among car drivers and poses a serious risk to road safety [1]. Microsleep is a phenomenon where a person falls asleep for a very short period , usually ranging from 3 to 10 seconds [1], [2]. Drivers are sometimes unaware of this happening, which can result in a significant decrease in their response time to road conditions. This can potentially lead to collisions between vehicles.

Advanced Driver Assistance Systems (ADAS) installed in a car can help to lower the risk of accidents. ADAS is usually equipped with features that can monitor the driver's condition, including the level of drowsiness, and provide a warning to stop driving immediately if the driver is observed to be drowsy [3], [4].

There are various ways to monitor a driver's level of drowsiness, and one of them is by utilizing artificial intelligence. The data collection process can be done using sensors that detect brain and body wave activity such as electrocardiogram (ECG) [5], [6] or electroencephalography (EEG) [7]. The recorded data is processed, and relevant features are extracted and fed into machine learning-based classification algorithms such as Multilayer Perceptron (MLP), K-Nearest Neighbor (KNN), and Bayesian Network (BN) [6]. This data becomes the training data for the algorithm to classify and detect the level of sleepiness in the driver. However, this method has some drawbacks - one of which is that the driver must have electrodes attached to their body to measure body and brain wave activity. This can negatively impact the driving comfort.

Another effective way to detect driver drowsiness is by using cameras and artificial intelligence-based image processing techniques to monitor the condition of the driver's eyes. Deep learning can be applied to a sequence of images to determine if the driver is feeling sleepy. To pre-process the image sequence, we can use the Recurrent Convolutional Neural Network algorithm based on the EfficientNetB0 model [8]. Fuzzy logic can then be applied to the image processing results to determine the driver's level of sleepiness [8]. Alternatively, a Convolutional Neural Network (CNN) can be used with the InceptionV3, VGG16, ResNet50V2 model architectures to detect sleepiness in real-time via the camera. The performance of the three models can be compared to determine the most effective method [9]. The research proposes a method to detect drowsiness using a camera that captures the driver's eyes in real-time. This is achieved by employing a Convolutional Neural Network that identifies the state of driver’s eyes. The model architecture utilized in the study is You Only Look Once (YOLO), specifically the YOLO V5 model, which processes annotated images belonging to two classes, "Mengantuk" (Drowsy) and "Bangun" (Awake) for classification. Based on this classification, the system can determine whether the driver is alert or drowsy. The warning system is activated when the driver keeps their eyes closed for a predetermined length of time.

II. THEORY

A. YOLO V5

YOLO V5 is an object detection algorithm that belongs to a series of previous iterations of the YOLO (You Only Look Once) model. This model uses a Convolutional Neural Network (CNN) based algorithm. The architecture of the YOLO V5 model is an advancement over the previous version. However, unlike the previous version which used Darknet, YOLO V5 uses PyTorch [10].

Three main parts are included in all YOLO models for object detection: backbone, neck and classification head [11]. However, YOLO V5 differs from previous models in that it uses a cross section partial network (CSP), called CSPDarknet53, on the previous YOLO Darknet. This network is used as an image feature extractor in the YOLO V5 network [11]. The Neck section of YOLO V5 employs a Path Aggregation Network (PANet) model to connect the Backbone and Head, which enhances the flow of information. It gathers and refines low-level features from the image obtained by the Backbone, thereby strengthening the spatial and semantic information and enabling accurate object localization [11]–[13]. The Head section consists of three branches that generate object class probability maps, prediction bounding boxes, and confidence scores for the objects detected in the image [11]–[13]. To complete object detection, Non-Maximum Suppression (NMS) is used to filter overlapping bounding boxes with low confidence scores below the threshold [12], [14]. Architecture used for object detection model YOLO V5 is shown in Figure 1. Meanwhile, Figure 2 illustrates the object detection process in YOLO V5.

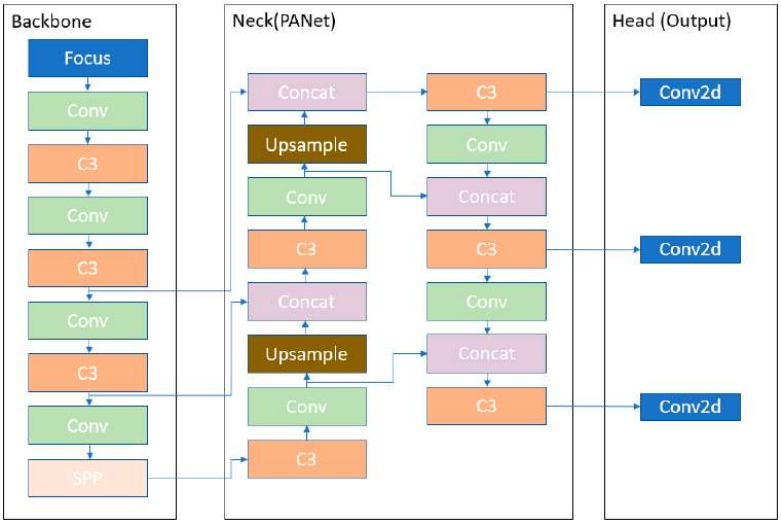

Figure 1. YOLO V5 network architecture [11].

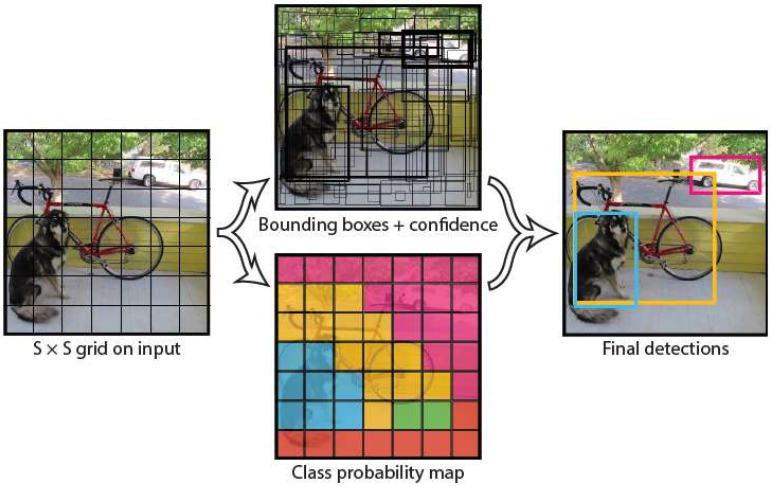

Figure 2. Object detection process in YOLO V5 [14].

III. METHODS

A. DATASET

Prior to implementing training procedures for YOLO V5 to enable object recognition based on customized data and labels, a preparatory process for the dataset is required. This research utilized a dataset comprising 200 images of the state of driver's eyes, each annotated with object class labels around the region of interest (ROI), specifically the eye area.

The object class label comprises two categories: "Bangun" and "Mengantuk". The image annotation process was automated using Roboflow. Subsequently, the 200 annotated images were segmented into training, validation, and test datasets. The division of the dataset employs a 70:20:10 ratio, with 70% of the data allocated to the training dataset, 20% to the validation dataset, and 10% to the test dataset. Figure 3 depicts the flow of the image dataset preparation process.

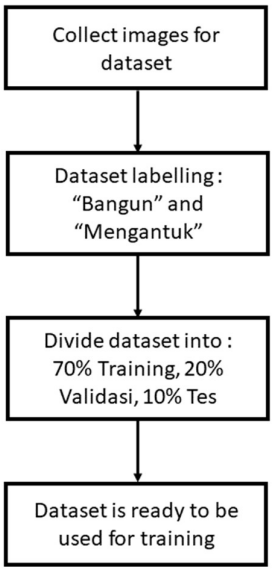

Figure 3. Image dataset preparation process

B. TRANSFER LEARNING

To detect the state of driver's eyes from a prepared dataset, the pre-trained YOLO V5 model used in this research requires re-training. This is done using the transfer learning method, where a new dataset replaces the classification head of the pre-trained model. The model is then fine-tuned based on the new target provided [15]. This process is faster and more efficient than training a new model from scratch and requires less data [16]. The YOLO V5 transfer learning process uses the Pytorch library for Python and is conducted through Google Collab. The process utilizes an Nvidia T4 GPU on the Google Collab cloud server, eliminating the need for a physical GPU to carry out the retraining process. To better understand the transfer learning process in YOLO V5, refer to Figure 4.

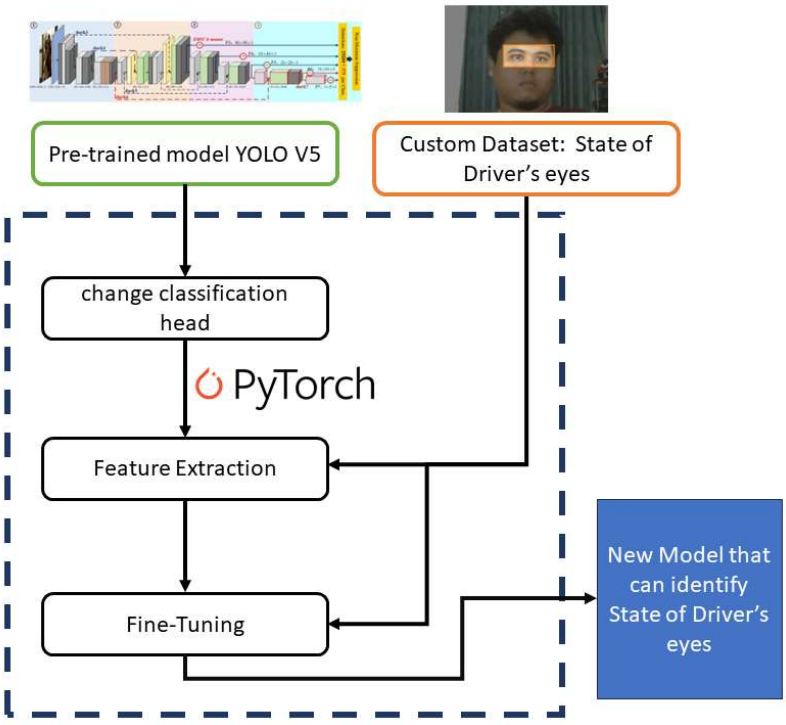

Figure 4. Transfer learning process for re-training YOLO V5 model

C. EVALUATION METRICS

Evaluation metrics, such as confusion matrix and mean average precision (mAP), are used to convey the performance of the trained model. A confusion matrix is a tool used in machine learning to evaluate the accuracy of a classification model. It provides a summary of the model's performance by comparing the predicted outcomes with the actual outcomes. The matrix displays the number of correct and incorrect predictions made by the model, broken down by class[17]. Each row of the matrix represents the

instances in an actual class, while each column represents the instances in a predicted class, as shown in Figure 5. This enables the model to be quickly identified as having a misclassification of classes and errors [17].

The confusion matrix consists of four classifications: True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP). These classifications are derived from the comparison of actual and predicted values. TP (True Positive) represents the number of correctly classified positive samples; TN (True Negative) represents the number of correctly classified negative samples; FP (False Positive) is the number of negative samples that are wrongly classified as positive; FN (False Negative) is the number of positive samples that are wrongly classified as negative [17], [18].

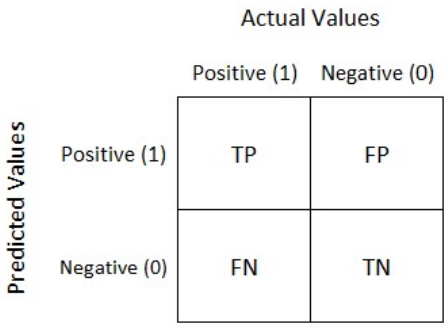

Figure 5. Confusion Matrix

The four parameters can be used to measure the model performance through Precision, Recall, and F1-Score values. Precision is a measure that indicates the proportion of true positives to the total number of projected positive data. It is an indicator of how accurately a classifier can identify positive instances [18], [19]. The precision formula is calculated by dividing the number of true positives (TP) by the total number of projected positive data. The variable FP plays a role in determining the precision value by acting as the divisor in the denominator [19], [20]. Equation (1) expressed mathematical formula of Precision [18]–[20].

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

On the other hand, Recall is a measure of how well a test or model identifies true positives. It is calculated by dividing the number of true positives by the total number of true positive and false negative cases [19], [20]. In other words, recall tells the proportion of actual positive cases that were correctly identified as positive. The mathematical formula of Recall expressed in (2) [18]–[20].

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

Recall and Precision have an inverse relationship. This means that when one goes up, the other goes down. For instance, if a model has a high Recall value, its Precision value will be lower and vice versa [19]. To find a balance between Recall and Precision, the F1-Score is used as metric that measures the harmonic mean of the two [18]–[20]. The F1-Score is commonly used to evaluate a classification model's overall performance. To calculate the F1-Score includes both Recall and Precision, which shown in (3) [18]–[20].

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision+} \quad (3)$$

Another way of evaluating the performance of a model is Mean Average Prediction (mAP). In object detection tasks, average precision (AP) is the most commonly used metric [21]. To calculate the AP, the area under the Precision-Recall curve (AUC) can be used as a measure of model performance [21], [22]. However, Precision and Recall curves can be challenging to visualize on a graph due to their zigzag-shaped nature and fluctuating values. For this reason, the AP number is calculated as the average precision of all Recall values from 0 to 1, using the 11 Point Interpolation method. This method involves interpolating the closest Precision value at 11 Recall points, including [0.0, 0.1, 0.2, 0.3, … , 1.0] as shown in (4) [22].

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} \rho_{interp}(r) \quad (4)$$

With:

$$\rho_{interp} = \max_{\tilde{r}:\, \tilde{r} \geq r} \rho(\tilde{r}) \quad (5)$$

If a model is designed to detect various classes of objects, the Average Precision (AP) for each class must be calculated. Subsequently, these AP values are averaged to produce a final performance metric known as the Mean Average Precision (mAP). To compute mAP at a certain IoU ($mAP@\alpha$), the formula presented in (6) may be employed [22]. Intersection over Union (IoU) quantifies the overlap between the predicted bounding box or segmented region and the ground truth bounding box or annotated region from a dataset.

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_i \;\; for\ n\ class \qquad (6)$$

D. MODEL INFERENCE AND TESTING

To test the YOLO V5 model's real-time detection capabilities, the model was deployed on computing hardware. In this stage, end users provide inputs to the computing system, which then processes the data, feeds it into the machine learning model, and returns prediction output to the users. This stage is called model inference. The hardware used for the inference system is Jetson Nano Single Board Computer as shown in Figure 6. The Single Board Computer was chosen to evaluate the model's performance when inferred on hardware with limited computing capabilities, so that it can be installed on a compact and practical device on the car. To capture videos and pictures, a Logitech C525 webcam camera was used. Additionally, a sound card and speaker were used to provide a warning sound to the driver. A 7-inch monitor was used to observe the detection results during the inference process.

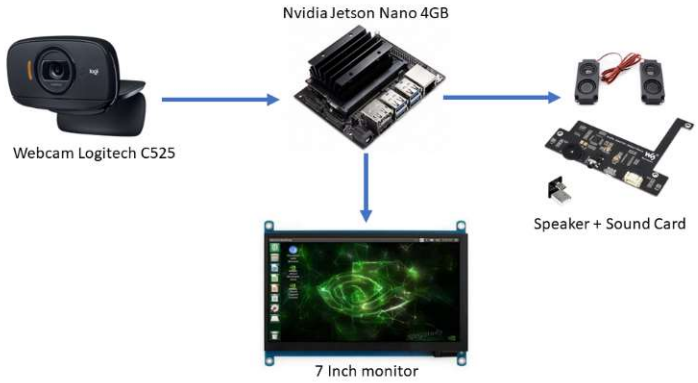

Figure 6. Hardware diagram for model inference testing

The inference process starts by capturing a sequence of the real-time image of the driver's eyes, and then the data is processed on the inference engine and fed to the model. The model then outputs the classification of new data, whether the driver detected drowsy or awake. The detection result is displayed on the screen along with the confidence score and bounding box. If the detection result is drowsy, a timer in the program is going to start counting until the counter count is more than 30 seconds to activate the warning system. Flowchart of state of driver's eye detection, model inference process, and warning system is shown in Figure 7.

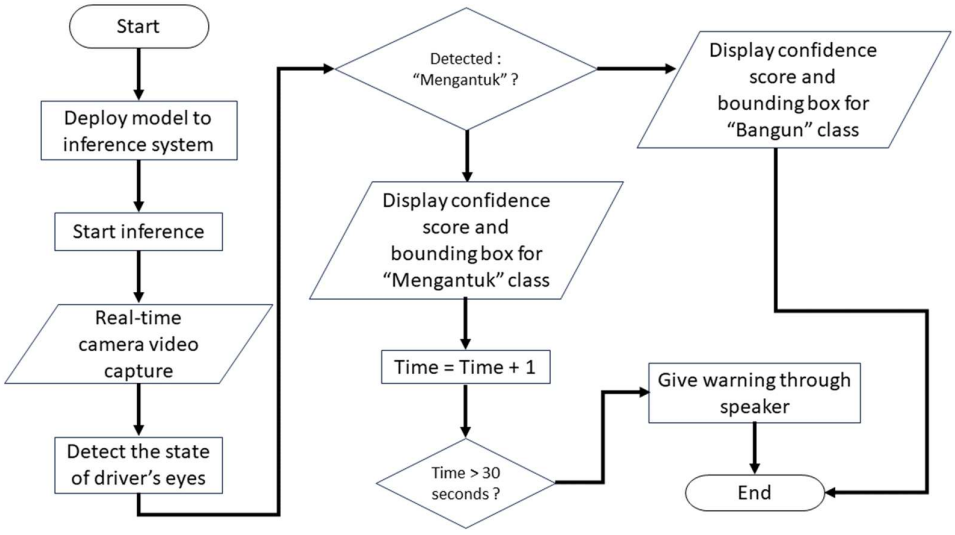

Figure 7. Model inference and drowsiness detection flowchart

IV. RESULTS AND DISCUSSION

The YOLO V5 model was trained with the MS COCO dataset, and then underwent a re-training process using a custom dataset. The custom dataset contained 139 training images, 40 validation images, and 20 test images. The model was trained over 100 epochs to classify "Bangun", "Mengantuk", and "Background" categories. The confusion matrix for the trained model is shown in Figure 8.

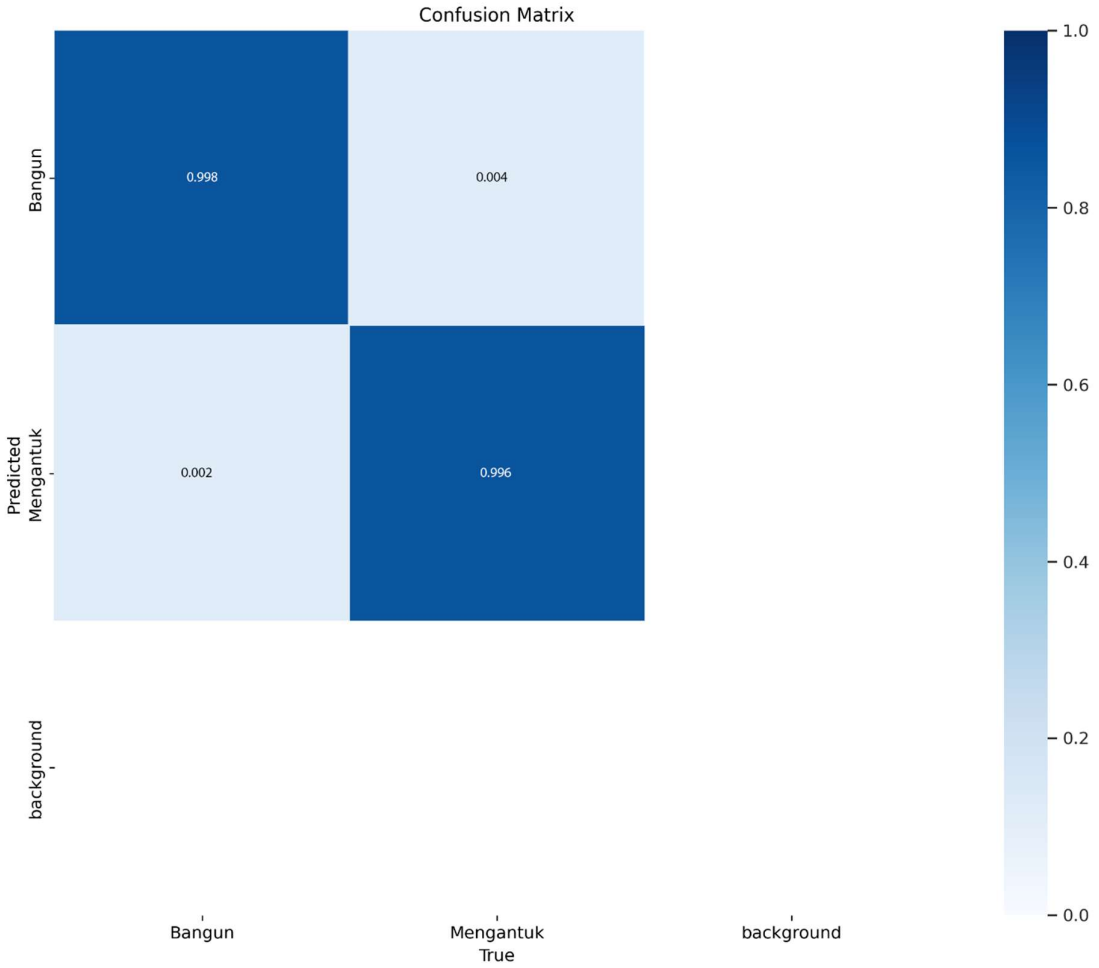

Figure 8. Confusion matrix of trained YOLO V5

The model validation results show a TP (true positive) value of 0.998 for the "Bangun" class and a TP value of 0.996 for the "Mengantuk" class. The FN (false negative) value for both classes is 0. On the other hand, the FP (false positive) value is 0.002 for the “Bangun" class, which was predicted as "Mengantuk", and 0.004 for the "Mengantuk" class, which was predicted as "Awake". Based on these results, we can calculate the Precision model value using (1) and get 0.998 or 99.8% for the "Awake" class and 0.996 or 99.6% for the "Drowsy" class. The overall model Precision value, considering both classes, is 0.997 or 99.7%. Regarding the Recall model results, based on (2), we obtained a value of 1 or 100% for both classes and overall. You can see the overall model evaluation metric results in Figure 9.

Class	Images	Instances	P	R
all	40	40	0.997	1
Bangun	40	26	0.998	1
Mengantuk	40	14	0.996	1

Figure 9. The overall value of the model evaluation metrics

The F1-Score value of the model is derived from the Precision and Recall values using (3). The calculations show that the F1-Score value is 0.998 or 99.8%, indicating that the model has a high accuracy level. The graph in Figure 10 displays the model's performance, which is consistent with the high F1-Score value. Meanwhile, the results of mAP 0.5, shown in Figure 11, also demonstrate the model's high accuracy, with a value of 0.995 or 99.5%. The characteristics of the model performance, as depicted in the graph in Figure 10, suggest that the model is neither overfitting nor underfitting. Overall, the model appears to be performing well and delivering accurate results.

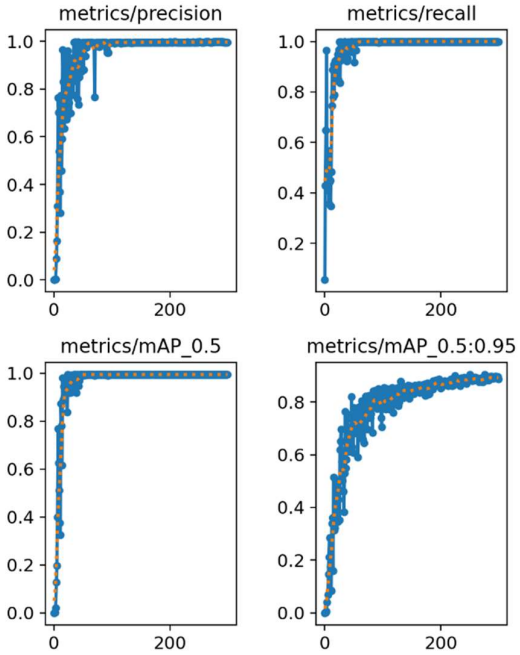

Figure 10. Model performance graph

mAP50	mAP50-95:
0.995	0.888
0.995	0.884
0.995	0.892

Figure 11. Overall mAP value of trained model

Inference testing demonstrates that the model can distinguish between "Mengantuk" and "Bangun" states. Figure 12 shows the detection results from the trained model's inference procedure.

Figure 12. Detection of the state of driver's eyes using the custom YOLO V5 model

V. CONCLUSION

The YOLO V5 model can accurately determine the state of the driver's eyes with a Precision value of 99.7%, Recall 100%, F1-Score 99.8%, and mAP 0.5 99.5%, according to the research's findings and discussion. Performance graphs and inference testing demonstrate that the model is capable of inference on devices with limited computational power and does not exhibit overfitting or underfitting. The model performance value is influenced by variations in the dataset and the quantity of the dataset. The author suggests that in order to get a model with good performance and prevent overfitting and underfitting, the dataset should have variations in angles, brightness levels, and photo backgrounds. It should also increase the training dataset and distinguish it from the validation data.

CONFLICT OF INTEREST

All authors declare that they have no conflicts of interest

REFERENCES

[1] S. Jay and R. Anis, "Microsleep: What Is It, What Causes It, and Is It Safe?," 2023. https://www.sleepfoundation.org/how-sleep-works/microsleep (accessed Oct. 18, 2023).
[2] S. Felson, "What to Know About Microsleep," WebMD, 2023. https://www.webmd.com/sleep-disorders/what-to-know-microsleep (accessed Oct. 18, 2023).
[3] M. Q. Khan and S. Lee, "A comprehensive survey of driving monitoring and assistance systems," Sensors (Switzerland), vol. 19, no. 11, 2019, doi: 10.3390/s19112574.
[4] A. Ziebinski, R. Cupek, D. Grzechca, and L. Chruszczyk, "Review of advanced driver assistance systems (ADAS)," AIP Conf. Proc., vol. 1906, 2017, doi: 10.1063/1.5012394.
[5] M. Gromer, D. Salb, T. Walzer, N. M. Madrid, and R. Seepold, "ECG sensor for detection of driver's drowsiness," Procedia Comput. Sci., vol. 159, pp. 1938–1946, 2019, doi: 10.1016/j.procs.2019.09.366.
[6] N. S. Nor Shahrudin and K. A. Sidek, "Driver drowsiness detection using different classification algorithms," J. Phys. Conf. Ser., vol. 1502, no. 1, 2020, doi: 10.1088/1742-6596/1502/1/012037.
[7] Z. Mardi, S. Naghmeh, M. Ashtiani, and M. Mikaili, "EEG-Based Drowsiness Detection for Safe Driving Using Chaotic Features and Statistical Tests," vol. 1, no. 2, pp. 130–137, 2011
[8] E. Magán, M. P. Sesmero, J. M. Alonso-Weber, and A. Sanchis, "Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images," Appl. Sci., vol. 12, no. 3, 2022, doi: 10.3390/app12031145.
[9] R. Florez, F. Palomino-Quispe, R. J. Coaquira-Castillo, J. C. Herrera-Levano, T. Paixão, and A. B. Alvarez, "A CNN-Based Approach for Driver Drowsiness Detection by Real-Time Eye State Identification," Appl. Sci., vol. 13, no. 13, 2023, doi: 10.3390/app13137849.
[10] J. Solawetz, "What is YOLOv5? A Guide for Beginners.," 2020. https://blog.roboflow.com/yolov5-improvements-and-evaluation/ (accessed Jan. 03, 2024).
[11] U. Nepal and H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs," Sensors, vol. 22, no. 2, 2022, doi: 10.3390/s22020464.
[12] H. Liang, J. Chen, W. Xie, X. Yu, and W. Wu, "Defect detection of injection-molded parts based on improved-YOLOv5," J. Phys. Conf. Ser., vol. 2390, no. 1, 2022, doi: 10.1088/1742-6596/2390/1/012049.
[13] B. Aydin and S. Singha, "Drone Detection Using YOLOv5," Eng, vol. 4, no. 1, pp. 416–433, 2023, doi: 10.3390/eng4010025.
[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.
[15] Dhruv Matani, "A Practical Guide to Transfer Learning using PyTorch," KDnuggets, 2023. https://www.kdnuggets.com/2023/06/practical-guide-transfer-learning-pytorch.html (accessed Jan. 03, 2024).
[16] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," J. Big Data, vol. 9, no. 1, 2022, doi: 10.1186/s40537-022-00652-w.
[17] G. Zeng, "On the confusion matrix in credit scoring and its analytical properties," Commun. Stat. Methods, vol. 0, no. 0, pp. 1–14, 2019, doi: 10.1080/03610926.2019.1568485.
[18] Y. Zhang, T. Zuo, L. Fang, J. Li, and Z. Xing, "An Improved MAHAKIL Oversampling Method for Imbalanced Dataset Classification," *IEEE Access*, vol. 9, pp. 16030–16040, 2021, doi: 10.1109/ACCESS.2020.3047741.
[19] Z. Ning, X. Wu, J. Yang, and Y. Yang, "MT-YOLOv5: Mobile terminal table detection model based on YOLOv5," J. Phys. Conf. Ser., vol. 1978, no. 1, 2021, doi: 10.1088/1742-6596/1978/1/012010.
[20] Teemu Kanstrén, "A Look at Precision, Recall, and F1-Score," Towards Data Science, 2020. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec (accessed Jan. 03, 2024).
[21] M. Otani, R. Togashi, Y. Nakashima, E. Rahtu, J. Heikkila, and S. Satoh, "Optimal Correction Cost for Object Detection Evaluation," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2022-June, pp. 21075–21083, 2022, doi: 10.1109/CVPR52688.2022.02043.
[22] Kristian Wilianto, "Evaluation Metrics pada Computer Vision dari Klasifikasi hingga Deteksi Objek," Data Folks Indonesia, 2021. https://medium.com/data-folks-indonesia/evaluation-metrics-pada-computer-vision-dari-klasifikasi-hingga-deteksi-objek-5049d3fd90d2 (accessed Jan. 03, 2024).