Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th June 2025. Vol. 4. No. 3; e-ISSN: 2808-4519

Utilization of YoloV8 Algorithm for in-Vehicle Video-Based Driver Monitoring System

M. Rajendra Aria Satya[1]*, Odi Nurdiawan[2], Fadhil M. Basysyar[3], Rahmat Hidayat[4]

[1]Program Study Teknik Informatics, STMIK IKMI Cirebon
[2]Informatics Management Study Program, STMIK IKMI Cirebon
[3]Information Systems Study Program, STMIK IKMI Cirebon
[4]Program Study Teknik Informatics, STMIK IKMI Cirebon
rajendraarya54@gmail.com[1*]

Abstract

Driving safety is a critical factor in reducing the risk of traffic accidents, which are often caused by unsafe driver behavior such as drowsiness, phone usage, or neglecting to wear seat belts. To address this, a real-time driver monitoring system is needed to detect and identify risky behaviors using the YOLOv8 algorithm. This study utilizes a secondary dataset titled "DMS Driver Monitoring System" from Kaggle, comprising 9,440 images of various driver behaviors. The dataset underwent preprocessing, including resizing 640x640 pixels and data augmentation to increase image diversity. The YOLOv8 model was trained for 100 epochs with a data split of 70% training, 20% validation, and 10% testing. Performance was evaluated using precision, recall, F1-score, and mean Average Precision (mAP). Results showed that the model achieved 89.6% precision, 87.2% recall, 88.0% F1-score, and 92.0% mAP50. The mAP50–95 score of 69.1% indicates room for improvement in more complex detection scenarios. Real-time video testing revealed the model could detect open eyes with 85% confidence and seat belt use with 35% confidence. The study concludes that YOLOv8 is effective for standard behavior detection but requires further optimization to handle varying lighting and camera angles for broader real-world deployment.

Keywords: Yolov8, Driver Monitoring, Behavior Detection, Real-Time, Driving Safety.

1. Introduction

Driving safety is one of the important things in the transportation sector, considering the high number of traffic accidents caused by driver negligence. Some of the main causative factors of accidents are fatigue, drowsiness, or lack of focus during driving. Weak public awareness of traffic regulations can lead to a low level of discipline in driving, thus causing a culture of indiscipline in society [1]. Artificial intelligence (AI) technology is rapidly developing to provide solutions to this problem, especially by utilizing video image processing in vehicles. The You Only Look Once (YOLO) algorithm as one of the deep learning approaches has shown excellence in real-time object detection with high precision.

Undetected drivers' behavior while driving is often the cause of fatal accidents. Manual monitoring systems that rely on human supervision have limitations in detecting changes in driver behavior in real-time. In addition, existing automated detection still faces challenges such as low accuracy, limited training data, and an inability to analyze complex behaviors. Some previous studies used object detection algorithms like the previous generation YOLO, but the results have not been optimal to detect factors such as head position, hand movements, or the driver's level of alertness.

The scope of previous research includes a variety of approaches to detecting drowsiness and risky driving behaviors. First, the research conducted by Christian Sanjaya used a combination of electrocardiography (ECG) instrumentation and visual Haar Classifier to detect the driver's drowsiness condition. This approach focuses on biomedical signal processing and visual imagery but requires additional devices for ECG signal capture that are less practical to be widely applied [2] Second, the research of Charlos Kurniawan Umbu Nggiku uses the facial landmark and eye aspect ratio (EAR) method to detect drowsiness through changes in facial position and eye condition. This method is simpler because it is based on image processing, but it has limitations in poor lighting conditions [3]. Third, Edmund Ucok Armin's research utilizes the YOLOv8 algorithm in detecting driver drowsiness. This approach demonstrates the advantages of YOLOv8 in detecting drowsiness conditions quickly and accurately, but its application is still limited to specific dataset tests and does not yet cover a wide range of other driver behaviors [4].

This study aims to utilize the YOLOv8 algorithm in detecting driver behavior in real-time through in-vehicle video, with a focus on improving the accuracy and speed of detection. The system is expected to identify risky behaviors such as cell phone use, drowsiness, or loss of focus. With the existence of a YOLOv8-based monitoring system, the potential for accidents due to driver negligence can be significantly minimized. This research also aims to contribute to the development of smart transportation technology that focuses on safety.

This study uses an experimental approach by implementing YOLOv8 to analyze video data in vehicles. The dataset used includes a variety of driver behaviors, such as holding a cell phone, looking sideways, or closing their eyes, collected from both open source and primary data. The model training process involves setting hyperparameters, augmenting data, and validating using the cross-validation method. Tests are carried out in a variety of environmental conditions, such as low lighting and unstable vehicle movement, to evaluate the reliability of the model.

The results of this study are expected to make a significant contribution to the development of transportation safety technology. The monitoring system developed can be used by vehicle manufacturers to improve safety features in new generation cars. In addition, this research can serve as a reference for regulators in designing policies that encourage the use of automated monitoring technology in the transportation sector. With wider implementation, the technology can be used on public vehicles such as buses or trucks to monitor driver compliance with safety standards. The research also opens opportunities for further integration with IoT technologies, such as early warning systems connected to vehicle control applications.

2. Literature Review

The literature review used in this study is by reviewing articles related to "Utilization of YOLOv8 Algorithm for In-Vehicle Video-Based Driver Monitoring System." The steps of literature review include 4 stages, namely (1) problem formulation, (2) literature search, (3) data evaluation, (4) analysis and interpretation [5].

Various previous studies have studied a lot of driver drowsiness detection systems with various approaches, ranging from biomedical signal analysis, digital image processing, to the use of deep learning algorithms. The image-based approach generally uses face and eye detection with methods such as Haar Cascade, Eye Aspect Ratio (EAR), as well as facial landmark detection [6][7]. Meanwhile, some studies have also developed physiological signal-based systems such as ECGs combined with classification algorithms, such as Random Forest, to detect changes in body conditions that indicate drowsiness [8]. On the other hand, CNN-based and YOLOv5/v8-based models are starting to be widely applied to detect facial expressions or behaviors related to drowsiness in real-time, showing high accuracy results [9][10].

However, most of the research still focuses on one aspect of detection, be it eye conditions, oral activity, or head posture, so it has not produced a holistic monitoring system. Some studies have even integrated other safety aspects such as seat belt wear detection but have not yet unearthed various parameters in one unified system. In addition, challenges such as reliance on lighting conditions, camera angles, and dataset limitations are major obstacles in the widespread application of the system in the real world. Therefore, the development of a system capable of detecting various aspects of driver safety in real-time with a lightweight and accurate model such as YOLOv8 is still indispensable in further research

3. Research Methods

This study uses an experimental method that aims to test the effectiveness of the YOLOv8 algorithm in detecting real-time and video-based driver behavior to be tested. The research method consists of several main stages, namely data collection, preprocessing, transformation, data mining, and evaluation. This stage is designed to ensure optimal data quality and produce accurate models that are in accordance with the following research objectives:

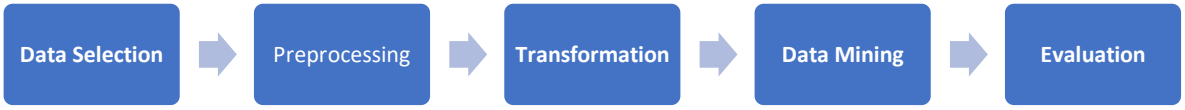

Fig. 1: Stages of Research Method

Based on Figure 1, this stage is designed to ensure that research runs systematically, from initial processing to evaluation of detection results. The following is a table of description of the activities of the research method based on the stages that have been described:

Table 1: Stages of Research Method

Stages	Activity	Activity Description
Data Selection	Dataset Selection	Select the "DMS Driver Monitoring System" dataset from Kaggle that is relevant to the research. The dataset consists of 9,440 images of driver behavior.
Preprocessing	Cropping and resizing	Change the image dimensions according to the needs of the YOLOv8 model input, which is 640x640 pixels.
	Data Augmentation	Perform data augmentation such as rotation, flipping, and lighting changes to increase the variety of training data.
Transformation	Labaeling (Anotation)	Annotates the image to identify the object.
	Dataset Sharing	Divide the dataset into three parts: train (70%), valid (20%), and test (10%).
Data Mining	Model Training (Training)	Training the YOLOv8 model uses a data train with 100 epochs to achieve optimal accuracy.
	Model Validation (Validation)	Validate the model using valid data to ensure good generalizations.
	Model Testing (Testing)	Test the model using test data to evaluate its performance in real conditions.
Evaluation	Model Performance Evaluation	Evaluate model performance using precision, recall, F1-score, and mAP metrics.

The stages in this research method are carried out through five main stages that are interconnected, namely data selection, preprocessing, transformation, data mining, and evaluation. This systematic approach aims to ensure optimal data quality and produce accurate and continuous models with the research objectives:

1. Data Selection

This dataset from Kaggle was chosen because it provides visual data relevant to the development of the Driver Monitoring System (DMS). Data selection is carried out to ensure that only images or images that support the development of the YOLOv8 model are taken, the image data obtained in the secondary dataset amounted to 9440 driver behavior images. This dataset includes various driver behavior conditions that have been uploaded on the roboflow.com website for preprocessing and labeling processes to obtain data transformation.

2. Preprocessing
   Given that this dataset covers a wide range of driver behavior conditions, this preprocessing stage includes resizing images with a size of 640x640 pixels. The size of 640x640 pixels was selected to match the YOLOv8 model input. Data augmentation processes such as rotation, flipping, or lighting changes to increase the variety of training data without adding raw data. Only using this augmentation process ensures that the model can handle the variation in the situation in the data.
3. Transformation
   This dataset is further processed through the roboflow.com website to carry out the transformation stage, which is a stage of labeling the image data or annotating the image data so that it can be read by the algorithm after being given a label or annotation. Then, the image data is divided into 3 parts, namely train, valid, and test. After this process has been carried out, the next step is the data training process on data mining.
4. Datamining
   At this stage, the YOLOv8 algorithm is applied to detect and classify driver behavior in the dataset. Data that has gone through the preprocessing and transformation stages is used as input for the model training, validation, and testing process. The model is trained to recognize objects or activities such as faces, eyes, cell phones, or seat belts with a high degree of accuracy. A good balance between accuracy and speed of training is therefore chosen 100 times epoch for data testing training.
5. Evaluation
   After the model is trained using this dataset, an evaluation stage is carried out to assess the model's performance. Metrics such as precision, recall, F1-score, and mean average precision (mAP) are used to evaluate the extent to which the model can detect risky driver behavior.

4. Results and Discussion

The results showed that the YOLOv8 model in the driver monitoring system had high performance. A precision value of 89.6% indicates accurate detection with minimal false positives, while a recall of 87.2% indicates a good ability to recognize driver behavior even if there are few false negatives. The mAP50 value of 92.0% reflects excellent detection accuracy, although the mAP50-95 drops to 69.1% on varied IoU. A fitness score of 71.4% indicates a balance between precision, recall, and mAP, indicating the model has good generalizations for a wide range of conditions.

Table 2: Metric Value

metrics/precision(B)	0.896
metrics/recall(B)	0.872
metrics/mAP50(B)	0.920
metrics/mAP50-95(B)	0.691
fitness	0.714

Based on the results of the evaluation of the value metrics in table 2, the YOLOv8 model used in this study showed quite optimal performance in detecting driver behavior. With an mAP50 value of 92% and an accuracy close to 90%, this model has a high level of accuracy in identifying driver's conditions. However, an mAP50-95 value of 69.1% indicates that the model can still be optimized, especially in more complex detections with higher IoU levels.

At this stage, the YOLOv8 model is trained using the dataset that has been labeled, and the dataset has been divided into 3 parts, namely 70% data training, 10% testing data and 20% validation data. The dataset includes images and labels of driver behavior that span five categories: waking eyes, sleepy eyes, cigarettes, phones, and seat belts. The following is the code used to conduct training and testing in Figure 4.6 for 100 epochs.

```
from datetime import datetime
start = datetime.now()
results = model.train(data='DMS.yaml', imgsz=640, epochs=100, batch=32, name="DMS_Output")
end = datetime.now()
```

Fig. 2: Code Runtime Data Mining

Based on the runtime code in Figure 2 it will train the model using the train function with the data configured in the DMS.yaml file, which defines the dataset and class labels. The model was trained using an image measuring 640x640 pixels (imgsz=640), for 100 epochs (epochs=100), and then determined the batch size, which is the number of samples processed at once in one iteration. Then, the model will process 32 images per batch with a batch size of 32 (batch=32). The training results, including the trained model and logs, will be stored in a directory named DMS_Output (name="DMS_Output"). The results of object detection with the following values:

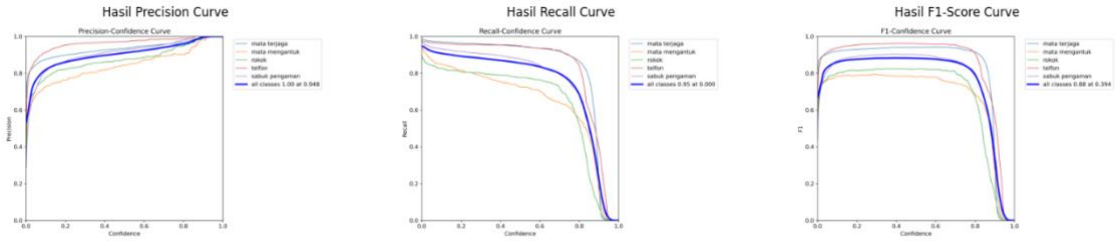

Fig. 3: Precision Value, Recall Value, And F1-Score Value

Based on figure 3 it is explained that the Precision value is 100.00%, Recall 95.00%, and F1-Score is 88.00% for all existing labels. The following is the code to display the results of the training and testing data as shown in figure 4 as follows:

```
[8]  # Create the full file path for 'results.csv' using the directory path and file name
     results_csv_path = os.path.join('/content/ultralytics/runs/detect/hasil/results.csv')

     # Load the CSV file from the constructed path into a pandas DataFrame
     df = pd.read_csv(results_csv_path)

     # Remove any leading whitespace from the column names
     df.columns = df.columns.str.strip()

     # Plot the learning curves for each loss
     plot_learning_curve(df, 'train/box_loss', 'val/box_loss', 'Box Loss Learning Curve')
     plot_learning_curve(df, 'train/cls_loss', 'val/cls_loss', 'Classification Loss Learning Curve')
     plot_learning_curve(df, 'train/dfl_loss', 'val/dfl_loss', 'Distribution Focal Loss Learning Curve')
```

Fig. 4: Code Runtime Evaluation Data Mining

After the code in Figure 4 is executed, the YOLOv8 model displays the results of the code. The following are the results of the training code described in figure 4 as follows:

Fig. 5: Box Loss Learning Curve

Based on figure 5, it can be observed that the value of train loss decreases consistently as the number of epochs increases, which indicates that the model is increasingly learning to reduce errors in object detection. Meanwhile, validation loss also decreased at the beginning of training, but after about the 40th epoch, the value of validation loss tended to be stagnant and did not experience a significant decrease, even appearing to fluctuate slightly. Then the Classification Loss Learning Curve is as follows:

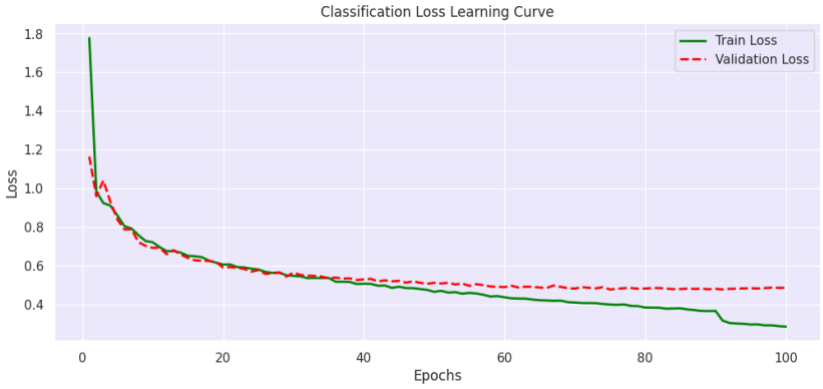

Fig. 6: Classification Loss Learning Curve

Based on Figure 6, at the beginning of the training, both train loss and validation loss were at a high value but experienced a significant decrease in the first few epochs. This suggests that the model quickly learns to recognize patterns in the data. After about the 20th epoch, both curves began to approach more stable values, with a small difference between train loss and validation loss. As for the Distribution Focal Loss Learning Curve as follows:

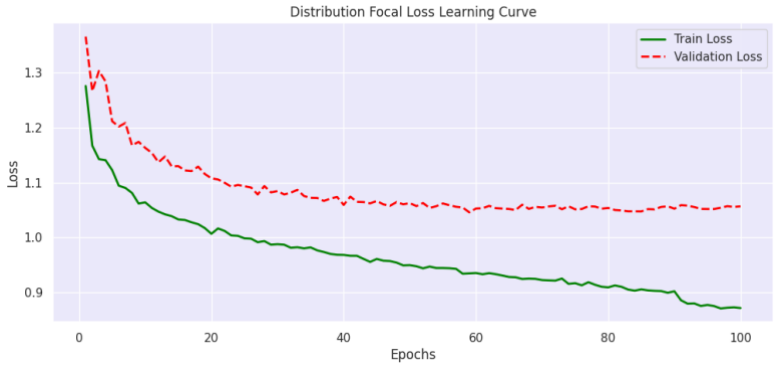

Fig. 7: Distribution Focal Loss Learning Curve

Based on Figure 7 at the beginning of the training, both train loss and validation loss have a high value, but it has decreased gradually as the number of epochs increases. The train loss curve shows a more significant downward trend compared to the validation loss, which indicates that the model continues to improve in its training process. However, after about the 40th epoch, validation losses slowed down and tended to stabilize, while train losses continued to decline. The stark differences between the two curves after the 50th epoch indicate the possibility that the model began to overmatch, where the learning model was too specific to the training data so that its performance on the validation data did not improve significantly.

After training, the model is applied to the video to test performance in real-world conditions. The video used comes from Boy William's YouTube channel, which includes various activities and driver behaviors, the first step for the youtube video to be processed on google colab is to enter the following script:

```
import os
from ultralytics import YOLO

# Load a pretrained model
model =
YOLO("/content/ultralytics/runs/detect/DMS_Output/weights/best.pt")

# YouTube video URL
youtube_url = "https://youtu.be/So2db5jdCQ0?si=H_gD48qWe5HSAafk"
video_path = "/content/ultralytics/video-sampel.mp4"
compressed_video_path = "/content/ultralytics/video-sampel.mp4"

# Download video  YouTube menggunakan yt-dlp
if not os.path.exists(video_path):  # Cek jika file video belum diunduh
    !yt-dlp -f 'mp4' -o "{video_path}" {youtube_url}  # Pastikan format
MP4
    print(f"Video berhasil diunduh ke: {video_path}")

# Compress video to 7 MB if necessary
if os.path.exists(video_path):
    original_size = os.path.getsize(video_path) / (1024 * 1024)  # Ukuran
dalam MB
    print(f"Ukuran asli video: {original_size:.2f} MB")
    if original_size > 7:
        print("Mengompresi video agar ukurannya tidak melebihi 7 MB...")
        !ffmpeg -i "{video_path}" -vf "scale=-2:720" -c:v libx264 -preset
slow -b:v 700k -pass 1 -an -y /dev/null && \
        ffmpeg -i "{video_path}" -vf "scale=-2:720" -c:v libx264 -preset
slow -b:v 700k -pass 2 -c:a aac "{compressed_video_path}"
        print(f"Video berhasil dikompresi ke: {compressed_video_path}")
        video_path = compressed_video_path  # Gunakan video yang sudah
dikompresi

# Define save directory
save_dir = "/content/ultralytics/runs/detect/predict"

# Run inference on the video
results = model.predict(video_path, save=True, save_dir=save_dir)  #
Jalankan prediksi pada video

# Optionally, process the results
for result in results:
    print(result)

print(f"Hasil disimpan di: {save_dir}")
```

Fig. 8: Codes to download youtube videos

Based on the code in Figure 8 it is a Python script to detect driver behavior in a video using a pre-trained YOLOv8 model. The script starts by loading the YOLO model from the directory that stores the best weights of the training results. Next, videos from YouTube are downloaded using yt-dlp if they don't already exist, making sure the format is MP4. If the video size exceeds 7 MB, the script will compress it using FFmpeg with parameters that keep the video quality at 720p resolution and 700 kbps bitrate. Once the video is ready, the script defines the storage directory for the detection results and then runs the prediction on the video using the model. predict function. The predicted results are stored in a predefined directory, and each detection result is displayed through iteration on the results object. Finally, the location where the detection results are stored is displayed as output. This code shows a comprehensive workflow from video downloading and compression to YOLOv8-based object detection. After the video has been successfully downloaded as described in figure 9, testing will be carried out on the video. The following is a video of the test data before detection as follows:

Fig. 9: Test Data on Youtube Videos Before Detection

In figure 9, this video test data shows the driver's condition without any marking or annotation. This is an original view of the driver's situation in the vehicle without any additional information about the objects in the video and has not been detected by the YOLOv8 model. While figure 10 is the result after the application of the YOLOv8 model as follows:

Fig. 10: Test Data on Youtube Videos After Detection

Based on figure 10, the test results show that the model can detect behaviors such as waking eye with a model confidence score of 0.85 or if it is adjusted to 85% and a seat belt with a visible model confidence score of 0.35 or if it is adjusted to 35%. Detections on video are also evaluated using an overlay bounding box to visualize the detection results. Qualitative analysis shows that the model has great potential to be applied in a direct driver monitoring system.

4.1. Effectiveness of the YOLOv8 Algorithm in Driver Behavior detection

Based on the results of the study, it can be concluded that the YOLOv8 algorithm shows high effectiveness in detecting driver behavior based on evaluation metrics such as precision, recall, F1-Score, and mean Average Precision (mAP). A precision value of 89.6% indicates that the model has good accuracy in detecting objects, with minimal false positives. Meanwhile, a recall value of 87.2% indicates that the model can detect most of the objects or behaviors that should have been detected, although there are still some cases of false negatives. The F1-Score value of 88.00% reflects a good balance between precision and recall, indicating that the model is not only accurate but also consistent in detecting driver behavior. In addition, an mAP50 value of 92.0% indicates excellent detection accuracy at the IoU threshold of 50%, although an mAP50-95 value of 69.1% indicates that the model can still be optimized to improve detection accuracy in more complex conditions. Thus, it can be concluded that the YOLOv8 algorithm has high effectiveness in detecting driver behavior based on the evaluation metrics used.

4.2. Performance of the YOLOv8 Algorithm in Driver Behavior detection

Based on the test results on the video, the YOLOv8 model shows good performance in real-time conditions. The model was tested on videos downloaded from YouTube, which included a wide range of driver activities and behaviors. The test results showed that the model was able to detect behaviors such as awakened eyes with a confidence score of 85% and seat belts with a confidence score of 35%. Despite variations in confidence scores, the model generally shows adequate ability to be applied under real-time conditions. Detections on video are also evaluated using an overlay bounding box to visualize the detection results. Qualitative analysis shows that the model has great potential to be applied in a direct driver monitoring system. However, there are some limitations, such as varying confidence scores, that suggest that the model can still be further optimized. Thus, it can be concluded that the YOLOv8 algorithm performs well in detecting driver behavior in real-time and video-based, although there is still room for further optimization.

5. Conclusion

Based on the results of the research and discussions that have been conducted, it can be concluded that the YOLOv8 algorithm has a good effectiveness in detecting driver behavior based on video-based and real-time evaluations. These results have answered the formulation of the problem and the purpose of the research quite well.

5.1. Value of the Effectiveness of the YOLOv8 Algorithm in Driver Behavior Detection

The results of the study showed the value of evaluation metrics such as precision (89.6%), recall (87.2%), F1-Score (88.00%), and mAP50 (92.0%). These values show that the YOLOv8 model has high effectiveness in detecting driver behavior. The analysis of the learning curve for box loss, classification loss, and distribution focal loss also provides a comprehensive picture of the model's performance during training. A significant decrease in loss values indicates that the model is increasingly able to learn patterns in the dataset effectively.

5.2. Measuring the Performance of the YOLOv8 Algorithm in Driver Behavior detection

Measuring the performance of the YOLOv8 algorithm under real-time and video-based conditions is tested through testing on videos downloaded from YouTube. The test results showed that the model was able to detect driver behavior such as awake eyes with a confidence score of 85% and seat belts with a confidence score of 35%. Despite variations in confidence scores, the model shows adequate ability to be applied to real-time conditions.

6. Suggestions

Based on the results of research on the implementation of the YOLOv8 algorithm in detecting video-based driver behavior, there are several suggestions that can be given for further research and development:

1. Improved Accuracy on Specific Behavior Categories
   Perform more varied data augmentation for low-accuracy categories, such as sleepy eyes and cigarettes, to improve the model's ability to detect behaviors with complex visual characteristics or small sizes. Add more and more varied training data, including low-light conditions, different camera shooting angles, and behavioral variations to enrich the representation of the dataset.
2. Optimasi Model YOLOv8
   Perform further hyperparameter tuning to get the best model configuration, such as adjusting the learning rate, batch size, or IoU threshold to optimize model performance. Implement techniques such as transfer learning with models that have been trained on similar datasets to speed up the training process and improve detection accuracy.
3. Testing in Real Conditions
   Test models in real-world environments with a variety of conditions, such as heavy traffic conditions, nighttime lighting, or extreme weather to measure the durability and accuracy of models outside of the laboratory environment. Use a high-quality camera or additional sensors to help detect behavior that is difficult to see, such as sleepy eyes or smoking.
4. Integration into Real-Time Monitoring Systems
   Optimize the speed of inference so that models can run more efficiently on low-spec hardware, such as IoT devices or car monitoring dashboards. Integrate YOLOv8 with an edge computing-based monitoring system to minimize latency in real-time driver behavior detection.
5. Development of Addition Behavior Categories
   Add other categories of driver behavior, such as using a cell phone while driving, looking back, or eating and drinking, to expand the scope of behavior detection and improve the system's application in driving safety.

References

[1] Titie Yustisia Lestari, Ridwan Tahir, and Irzha Friskanov. S, “Penyuluhan Hukum Tentang Karakter Siswa Yang Tertib Berlalu Lintas Di Madrasah Aliyah Ddi Lonja,” *J. Abdi Masy.*, vol. 7, no. 1, pp. 37–46, 2023, doi: 10.30737/jaim.v7i1.4951.

[2] C. Sanjaya, R. Setiawan, and N. F. Hikmah, “Sistem Pemantauan Kantuk Menggunakan Instrumentasi Elektrokardiografi dan Visual Haar Classifier pada Pengemudi,” *J. Tek. ITS*, vol. 9, no. 1, 2020, doi: 10.12962/j23373539.v9i1.45622.

[3] C. K. U Nggiku and A. Rabi, “Deteksi Kantuk Pada Pengemudi Mobil Menggunakan Eye Aspect Ratio Dengan Metode Facial Landmark,” *Semin. Nas. Fortei Reg.*, vol. 7, pp. 72–78, 2022, [Online]. Available: https://binaryupdates.com/

[4] E. U. Armin, A. Purnama Edra, F. I. Alifin, I. Sadidan, I. P. Sary, and U. Latifa, “Performa Model YOLOv8 untuk Deteksi Kondisi Mengantuk pada pengendara mobil,” *BRAHMANA J. Penerapan Kecerdasan Buatan*, vol. 5, no. 1, pp. 67–76, 2023, [Online]. Available: https://doi.org/10.30645/brahmana.v5i1.279

[5] T. Yuniati and M. F. Sidiq, “Literature Review : Legalisasi Dokumen Elektronik Menggunakan Tanda,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 6, pp. 1058–1069, 2020, [Online]. Available:

https://doaj.org/article/159b35f326d7453eb754b389eb4214c7%0Ahttps://repository.ittelkom-pwt.ac.id/5976/%0Ahttps://garuda.kemdikbud.go.id/documents/detail/1980051

[6] E. Lety Istikhomah Puspita Sari, I. Ketut Agung Enriko, J. Gegerkalong Hilir No, J. DI Panjaitan No, and J. Tengah, "Sistem Peringatan Tersemat untuk Pengemudi Mengantuk Embedded Alert System for Drowsy Drivers," *J. Ris. Rekayasa Elektro*, vol. 5, no. 1, 2023, [Online]. Available: http://jurnalnasional.ump.ac.id/index.php/JRRE

[7] C. Kurniawan Umbu Nggiku, A. Rabi, and S. Subairi, "Deteksi Kantuk Untuk Keamanan Berkendara Berbasis Pengolahan Citra," *J. JEETech*, vol. 4, no. 1, pp. 48–56, 2023, doi: 10.32492/jeetech.v4i1.4107.

[8] N. Nuryani, K. Nisak, and A. Dwijo Sutomo, "Pengembangan Sistem Deteksi Kantuk Menggunakan Pengklasifikasi Random Forest pada Sinyal Elektrokardiogram (Development of Drowsiness Detection System Using Random Forest Classifier on Electrocardiogram Signals)," *J. Nas. Tek. Elektro dan Teknol. Inf.* |, vol. 10, no. 3, pp. 265–271, 2021.

[9] K. Rohman and T. B. Sasongko, "FAST DETECTION OF SEATBELT DRIVER BASED ON IMAGE CAPTURING," *JURTEKSI (Jurnal Teknol. dan Sist. Informasi)*, vol. 9, no. 3, pp. 473–480, Jun. 2023, doi: 10.33330/jurteksi.v9i3.2276.

[10] S. N. Himawan, R. Sohiburroyan, N. B. Nugraha, J. Teknik Informatika, and P. Negeri Indramayu, "Deteksi Kantuk Pengemudi Menggunakan Deep Learning," *Semin. Nas. Ind. dan Teknol. (SNIT), Politek. Negeri Bengkalis*, no. November, pp. 1–8, 2022.