KLIK: Kajian Ilmiah Informatika dan Komputer
ISSN 2723-3898 (Media Online)
Vol 5, No 1, Agustus 2024, Hal 342-348
DOI 10.30865/klik.v5i1.1880
https://djournals.com/klik

Vehicle Counting Detection on Roadways using YOLO Algorithm with PyTorch Transfer Learning

Prihandoko[1,*], Ahsan Firdaus[2], Rini Sovia[2]

[1]Department of Informatics, Gunadarma University, Jakarta, Indonesia
[2]Department Information Technology, Universitas Putra Indonesia "YPTK" Padang, Padang, Indonesia
Email: [1,*]pri@staff.gunadarma.ac.id, [2]ahsan_firdaus@gmail.com, [3]rini_sovia@upuyptk.ac.id
Email Penulis Korespondensi: pri@staff.gunadarma.ac.id

Abstract−This research discusses the implementation of a vehicle counting detection system on roadways utilizing the YOLO (You Only Look Once) algorithm, integrated with PyTorch's transfer learning and fine-tuning techniques. The study is motivated by the rapid increase in private vehicle ownership in Indonesia, which has heightened concerns regarding traffic congestion and accident risks. The primary objective of this research is to develop an efficient vehicle counting system based on Convolutional Neural Networks (CNN), designed to process images and videos. The methodology encompasses a literature review, system analysis, design, implementation, and evaluation. The system is built using YOLOv8, tailored with transfer learning to enhance object detection, focusing on cars. To track and count vehicles, the Centroid Tracker algorithm is employed. A dataset of 1,667 images was used, partitioned into training (1,488 images), validation (118 images), and testing (61 images) sets. The model achieved a detection accuracy of 97.92%, though minor detection errors were observed. In video-based testing, the system effectively detected and tracked vehicles, assigning unique IDs to individual cars. In conclusion, the YOLOv8-based model, combined with the Centroid Tracker algorithm, demonstrates strong performance in detecting and counting vehicles, offering potential contributions to traffic monitoring systems and providing a foundation for more sophisticated applications in future research.

Keywords: Vehicle Counting Detection; PyTorch Transfer Learning; Centroid Tracker.

1. INTRODUCTION

The rapid development of technology has driven significant changes in transportation habits, especially with a rising preference for private vehicles. In Indonesia, studies have consistently shown a decline in public transport usage, correlating with increased ownership of personal vehicles. This shift is largely attributed to the poor quality of public transportation services [1]. As a result, Indonesia has seen a steady growth in vehicle sales, with data from Astra International indicating a surge in both car and motorcycle sales between 2020 and 2023 [2]. Badan Pusat Statistik (BPS) further reports a significant rise in vehicle ownership from 2021 to 2022, posing both opportunities for personal mobility and significant challenges in road safety and congestion [3].

Research shows a link between increasing vehicle ownership and higher traffic accident rates, with studies highlighting the characteristics of intersection accidents in dense areas, GIS-based identification of accident hotspots, and the socioeconomic and technological factors affecting accident rates [4-7], stressing the need for enhanced traffic management systems. Traffic congestion is another pressing issue linked to the increasing number of vehicles on the road. References [8], [9] have demonstrated that traffic density negatively impacts quality of life, productivity, and public health due to increased exposure to pollution and noise.

Given these challenges, effective traffic management solutions are essential. One promising solution is vehicle counting detection system, which provides valuable data for improving traffic flow, optimizing traffic light control, and informing advertising strategies on busy roads. These systems utilize crowd counting techniques in computer vision, allowing computers to replicate human visual perception and automatically estimate the number of vehicles in an image or video [10-12]. This technology enables real-time data collection for various applications, such as traffic tracking, adaptive traffic lights, and infrastructure planning [13], [14]. Therefore, integrating vehicle counting detection systems can significantly help in alleviating congestion and improving travel quality.

Artificial Intelligence (AI) has emerged as a transformative technology capable of simulating complex human skills, with applications spanning various domains and promising a new era of societal development [15]. Within the broader field of AI, Computer Vision stands out as a specialized discipline that enables machines to interpret and understand visual information from the real world, essentially replicating human visual functions through computational means [12]. One of the most powerful tools in Computer Vision is the Convolutional Neural Network (CNN), an architecture specifically designed to process two-dimensional data like images, excelling in tasks such as image classification, object detection, and image segmentation [16]. Building upon the strengths of CNNs, the YOLO (You Only Look Once) algorithm has revolutionized real-time object detection with its efficient and accurate approach, making it particularly well-suited for vehicle detection applications [17]. YOLO's ability to rapidly process images and accurately predict bounding boxes around vehicles, along with its adaptability to various environmental conditions, makes it an ideal solution for vehicle counting detection systems, offering robust performance in traffic monitoring systems.

The implementation of vehicle counting detection systems using the YOLO (You Only Look Once) algorithm has been explored in various studies. For instance, reference [18] discusses a CNN-based crowd counting method utilizing image pyramids to combine features, addressing multi-scale challenges while highlighting potential redundancy issues in information. Additionally, reference [19] illustrates YOLO's effectiveness for real-time object detection, enhancing traffic management systems and informing infrastructure planning and road safety measures. Moreover, reference [20]

underscores the significance of transfer learning in vehicle counting implementations, allowing models trained on one task to quickly adapt to another, thereby improving accuracy and efficiency in dynamic traffic environments.

Transfer learning is a machine learning technique where a model trained on one task is used as a starting point for a model on a second task. Transfer learning is particularly useful when the second task is similar to the first, or when limited data is available for the second task. It allows for leveraging knowledge gained from tasks with abundant training data to improve generalization in new tasks with limited data [21], [22]. This approach is especially beneficial given the limited dataset of approximately 500 annotated images in this study. Furthermore, transfer learning can enhance model robustness and stability, as pre-trained models have undergone rigorous validation and can recognize common objects [23]. This research focuses on implementing a vehicle counting detection system using YOLO, leveraging transfer learning techniques by utilizing pre-trained models and fine-tuning them on a new dataset. Specifically, this study aims to develop a program using transfer learning with the PyTorch framework to detect the number of four-wheeled vehicles (cars) on Jalan Raya Margonda in Depok, Indonesia. This proposed solution has the potential to contribute to traffic monitoring systems and serve as a foundation for more advanced applications in future research.

2. RESEARCH METHODOLOGY

The methodology for this research centers on implementing vehicle counting detection using the YOLO (You Only Look Once) algorithm, specifically YOLO v8, integrated with PyTorch transfer learning. YOLO v8, the latest iteration in the YOLO series, offers significant improvements in speed and accuracy compared to its predecessors, making it an ideal choice for real-time object detection tasks [24]. PyTorch, a flexible and efficient deep learning framework developed by Facebook AI Research lab, is utilized for its dynamic computational graph and GPU acceleration capabilities, which are crucial for handling large-scale data in deep learning applications [25]. The implementation of transfer learning techniques aims to leverage pre-trained models, reducing the dependency on extensive training data and computational resources [22]. For dataset management and annotation, Roboflow is employed, facilitating efficient image labeling and dataset organization [26]. Additionally, Kaggle's platform is used to access necessary computational resources for model training and evaluation [27]. The methodology is illustrated in a diagram that outlines the system's workflow, as shown in Figure 1. The process consists of several steps, beginning with data collection, followed by data preparation and preprocessing, transfer learning using the YOLO model, model evaluation, prediction using the model, results visualization, and concluding with data testing.

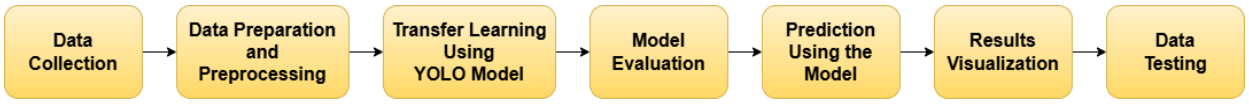

Figure 1. System Methodology

2.1 Data Collection

The data collection process involved gathering video footage from two specific locations along Jalan Raya Margonda in Depok: the pedestrian bridge at ITC Depok and the pedestrian bridge at Pondok Cina Station. Video recordings were captured using a Samsung Galaxy A73 5G smartphone, resulting in a total of three videos with a combined duration of approximately eight minutes. From these recordings, 675 images were extracted to form the initial dataset. The dataset focuses exclusively on four-wheeled vehicles, encompassing various types such as private cars and public transportation like angkot (public minivans). Following the data collection, manual labeling was performed on each image using bounding boxes, a crucial step in preparing the dataset for training the YOLO model, which relies on bounding box detection for object identification.

2.2 Data Preparation and Preprocessing

The data preparation and preprocessing step involves several key processes to ensure the dataset is properly organized, annotated, and optimized for training, as detailed in the following steps:

a) Project Creation: Set up the vehicle detection project on Roboflow's platform, utilizing its object detection features to establish a structured environment for data management and processing.
b) Data Upload: Transfer mp4 video recordings of road traffic to Roboflow, then extract image frames at predetermined intervals to create a comprehensive dataset representing various traffic conditions and vehicle positions.
c) Data Labeling: Employ Roboflow's annotation tools to manually label each extracted image, drawing precise bounding boxes around vehicles to provide accurate object location and classification data for model training.
d) Data Preprocessing: Apply a series of techniques including auto-orientation for correct image alignment, static cropping to focus on relevant areas, and resizing all images to 640x640 pixels to ensure dataset uniformity and compatibility with the YOLO algorithm.
e) Data Augmentation: Enhance dataset diversity by applying horizontal and vertical flipping, cropping with 0-20% zoom, and rotating images between -10° and +10°, simulating various viewing angles and conditions to improve model generalization.

f) Data Splitting: Partition the prepared dataset into training, validation, and testing sets, ensuring a robust evaluation framework for model performance and preventing overfitting.
g) Dataset Export: Generate the final dataset in a YOLO-compatible format, including properly normalized bounding box coordinates and class labels, ready for efficient model training and integration with PyTorch transfer learning techniques.

2.3 Transfer Learning using YOLO Model

This study utilizes the YOLOv8 model, optimized for real-time vehicle detection in road traffic environments. By applying transfer learning within the PyTorch framework, pre-trained weights are fine-tuned to enhance the model's performance in recognizing and localizing vehicles. This method reduces the need for large amounts of training data by building upon features learned from previous tasks, streamlining the adaptation process for the specific challenge of vehicle detection.

2.4 Model Evaluation

This study utilizes the YOLOv8 model, optimized for real-time vehicle detection in road traffic environments. By applying transfer learning within the PyTorch framework, pre-trained weights are fine-tuned to enhance the model's performance in recognizing and localizing vehicles. This method reduces the need for large amounts of training data by building upon features learned from previous tasks, streamlining the adaptation process for the specific challenge of vehicle detection.

2.5 Prediction Using the Model

Following training, the model is applied to the validation dataset to generate predictions. For each detected vehicle, the model outputs class labels and bounding box coordinates. These predictions are subsequently compared with ground truth annotations to measure the model's accuracy. This comparison provides a quantitative assessment of the model's performance in identifying and localizing vehicles across diverse traffic conditions, simulating real-world scenarios.

2.6 Results Visualization

To enable a more intuitive interpretation of the model's performance, the outputs are visually represented by overlaying annotations on the input images. Bounding boxes are drawn around each detected vehicle, along with confidence scores that indicate the model's certainty in its predictions. This visual representation offers a clear and immediate understanding of the model's ability to detect and localize vehicles across varying traffic conditions and environments, making the results more accessible for analysis.

2.7. Data Testing

The final evaluation involves testing the model on frame extracts from traffic CCTV footage, simulating real-world application scenarios. The model is applied to these test images to generate bounding boxes and confidence scores for each detected vehicle. This step is critical in assessing the model's ability to generalize to new, unseen data, providing valuable insights into its practical performance and reliability for real-world traffic monitoring applications.

3. RESULTS AND DISCUSSION

The data collection and preprocessing stages yielded a comprehensive dataset tailored for vehicle detection. From the initial video recordings, a total of 675 high-quality images were extracted, each featuring various four-wheeled vehicles under different traffic conditions. The preprocessing efforts resulted in a uniform dataset of 640x640 pixel images, optimized for YOLO algorithm compatibility. Data augmentation techniques significantly expanded the dataset's diversity, simulating a wide range of real-world scenarios including varied viewing angles and lighting conditions. The final dataset composition comprised 1,488 images (89%) for training, 118 images (7%) for validation, and 61 images (4%) for testing. This strategic distribution ensures a robust foundation for model training and evaluation. Notably, the manual labeling process using bounding boxes provided precise object location and classification data, critical for training the YOLO model. The resulting dataset, exported in YOLO-compatible format, includes normalized bounding box coordinates and class labels, forming a solid groundwork for the subsequent transfer learning phase with the YOLOv8 model.

3.1 Transfer Learning Implementation Using YOLO Model

The transfer learning implementation with YOLOv8n yielded promising results for vehicle detection. The model, comprising 168 layers and over 3 million parameters, was fine-tuned on the final dataset. Training was conducted for 20 epochs with an input image size of 640x640 pixels. During testing as shown in Figure 2, the model processed a 384x640 pixel image in 90.3ms for inference, with minimal additional time for pre-processing (2.8ms) and post-processing (1799.5ms). This rapid processing capability is crucial for real-time vehicle detection applications. In a sample test, the model successfully identified 7 cars in a single frame, indicating its ability to generalize to new, unseen data. The fine-

tuned model, saved as 'best.pt', struck a balance between computational efficiency and detection accuracy, making it suitable for deployment in various traffic monitoring scenarios.

```
# Run inference on an image with YOLOv8n
!yolo predict model='/kaggle/input/skripsi-dataset/best.pt' source='/kaggle/input/skripsi-dataset/car-in-highway-1.jpg'

Ultralytics YOLOv8.2.27 🚀 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3005843 parameters, 0 gradients, 8.1 GFLOPs

image 1/1 /kaggle/input/skripsi-dataset/car-in-highway-1.jpg: 384x640 7 cars, 90.3ms
Speed: 2.8ms preprocess, 90.3ms inference, 1799.5ms postprocess per image at shape (1, 3, 384, 640)
Results saved to runs/detect/predict
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

Figure 2. The Process of Vehicle Detection and Counting from a Single Image

For real-time vehicle tracking and counting, a CentroidTracker class was implemented to work alongside the YOLOv8n model. This tracker assigned unique IDs to detected vehicles and maintained their positions across video frames. The system was configured to consider a vehicle as "disappeared" if not detected for 50 consecutive frames, allowing for temporary occlusions or brief exits from the frame. When tested on a video input, the combined YOLOv8n model and CentroidTracker successfully detected and counted unique vehicles passing through the designated area. The system processed each frame, applying object detection, updating object positions, and maintaining a count of unique vehicles. This approach enabled accurate vehicle counting even in scenarios with multiple moving objects and varying traffic conditions, demonstrating the practical application of the trained model in a real-world traffic monitoring context.

3.2 Evaluation Results

The evaluation results of the vehicle detection model reveal strong performance metrics across various analyses as shown in Figure 3. The Precision-Recall Curve indicates a high precision of 0.986 for the 'car' class, demonstrating the model's effectiveness in accurately detecting vehicles while maintaining a low rate of false positives. This is further supported by the mean Average Precision (mAP) of 0.986 at an Intersection over Union (IoU) threshold of 0.5, which highlights the model's ability to detect a significant number of instances with minimal error. The Recall-Confidence Curve illustrates that at lower confidence levels, the model achieves nearly perfect recall, close to 1.0, thereby successfully identifying almost all vehicle instances. However, as confidence increases, recall gradually decreases, reflecting a conservative approach by the model to enhance precision, which may result in some undetected instances. The Precision-Confidence Curve corroborates these findings, showing that precision increases sharply with rising confidence, reaching nearly 1.0 at high confidence levels, with a notable precision of 1.00 at a confidence threshold of 0.863. The F1-Score Curve reveals a balanced relationship between precision and recall, peaking around a confidence of 0.1 with a high F1-score of 0.94 at a threshold of 0.689, indicating the model's capacity to effectively balance both metrics across varying confidence levels. Overall, these metrics suggest that the model exhibits excellent capabilities in detecting vehicles in videos, making it a reliable tool for applications demanding high accuracy and robust detection performance.

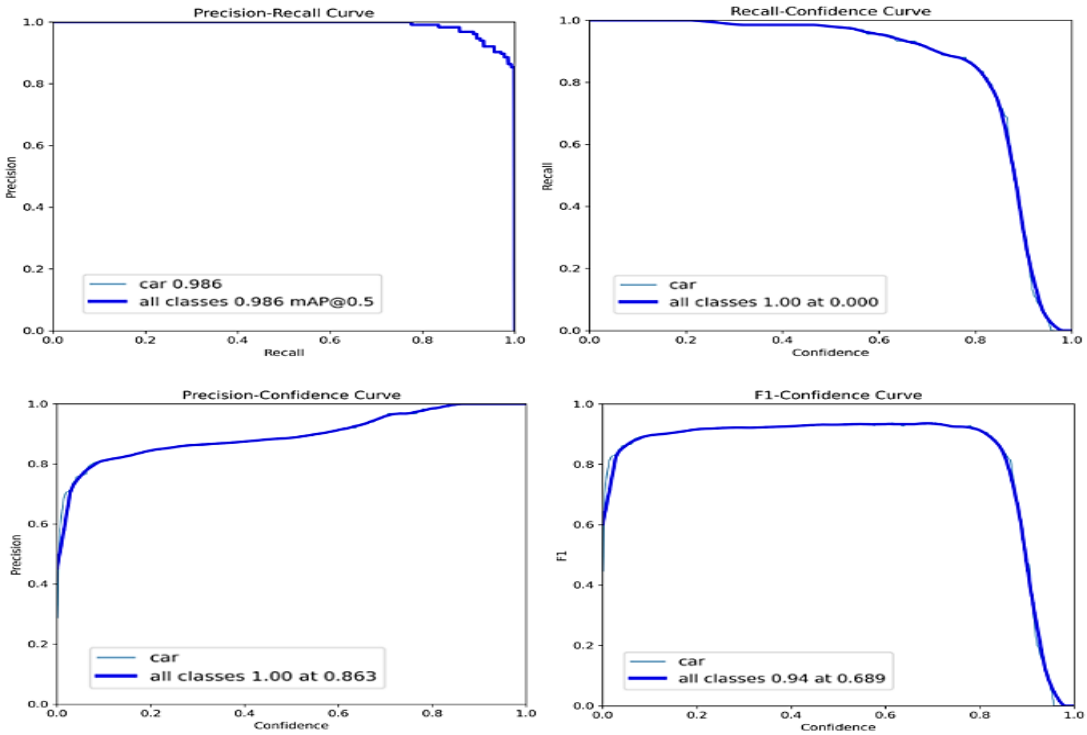

Figure 3. Evaluation Using Various Metrics

Figure 4. Testing Using Image Input from Internet

The second test involving a video input from a mobile device, the YOLOv8 model exhibited robust performance in detecting and counting vehicles on the road. As shown in Figure 5, two cars were accurately identified, each assigned a unique ID and a detection probability of around 83.5%. The model reported a total detection of eight cars, highlighting its effectiveness in tracking multiple vehicles across frames, despite potential discrepancies due to vehicles exiting the frame or ID reassignments.

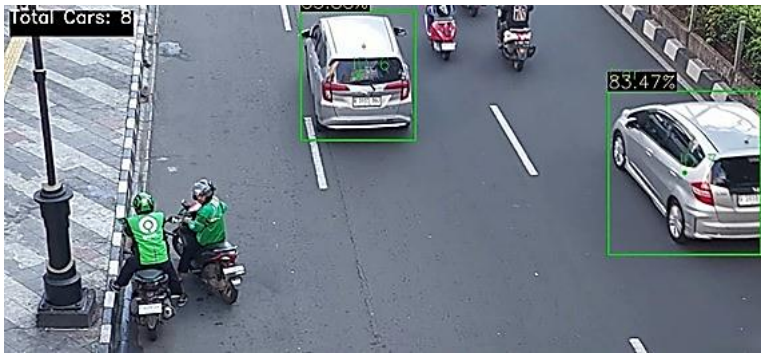

Figure 5. Testing Using Normal Video Input from Mobile Phone in Margonda Raya

The final test involved extracting frames from the previous video, resulting in a total of 24 images for analysis. The detection results for all frames are presented in Table 1, which compares the number of cars detected by the proposed system against manual counting. This comparison facilitates the calculation of error percentages and the overall accuracy of the system.

Table 1. Test Results Using Video Frame Input from Mobile Phone

Image	Model Detection	Manual Count	Error Percentage (%)
Image 00	0	0	0
Image 01	0	0	0
Image 02	2	2	0
Image 04	1	1	0
Image 05	2	2	0
Image 06	0	0	0
Image 07	0	0	0
Image 08	1	1	0
Image 09	0	0	0
Image 10	1	1	0
Image 11	0	0	0
Image 13	0	0	0
Image 14	1	1	0
Image 16	1	1	0
Image 18	2	2	0
Image 19	2	2	0
Image 20	2	2	0
Image 21	0	0	0
Image 23	1	1	0
Image 24	1	1	0

Image	Model Detection	Manual Count	Error Percentage (%)
Image 25	1	1	0
Image 26	1	2	50%
Image 28	2	2	0
Image 29	2	2	0
Average Error Percentage			2.08%
Average Accuracy Percentage			97.92%

The test results presented in Table 1 reveal both strengths and limitations of the vehicle detection model. While the model achieved a high overall accuracy rate of 97.92%, with an average error percentage of just 2.08%, one notable exception emerged. In a specific instance, the model failed to detect a particular type of vehicle known as an angkot, resulting in a significant 50% error rate for that case. This outlier highlights a potential area for improvement in the model's object recognition capabilities, particularly when confronted with less common or region-specific vehicle types. Despite this isolated shortcoming, the model demonstrated robust performance across various scenarios, effectively detecting and enumerating vehicles in diverse contexts. These findings suggest that the vehicle detection model holds considerable promise for practical applications such as traffic management. However, to enhance its reliability further, future research should focus on improving the model's ability to recognize a wider range of vehicle types and adapt to varying environmental conditions.

4. CONCLUSION

This study has successfully developed and evaluated a YOLOv8-based object detection model integrated with the CentroidTracker algorithm for detecting and counting vehicles on roads. The model demonstrated robust performance, achieving an impressive accuracy rate of 97.92% in detecting and enumerating vehicles across diverse scenarios. The development process encompassed data collection, preprocessing, labeling, augmentation, model training, and evaluation using a dataset of 1,667 images. Evaluation metrics, including confusion matrices, precision-recall curves, and F1-score curves, indicated optimal balance between precision and recall. The model's ability to assign unique IDs to detected vehicles and track them across video frames further showcased its potential for traffic monitoring applications.

Despite its high overall accuracy, the model exhibited some limitations, such as occasional misclassifications and failure to detect certain vehicle types like angkot. These findings suggest areas for future improvement, including the use of larger and more diverse datasets, implementation of more complex data augmentation techniques, and exploration of alternative tracking algorithms like SORT or DeepSORT. Additionally, integrating the model with other systems such as traffic prediction or license plate recognition could expand its practical applications. Further evaluation across various environmental conditions and traffic scenarios is recommended to ensure the model's robustness in real-world applications.

REFERENCES

[1] D. A. Kurniawan, "Mengapa Kendaraan Pribadi Terus Bertumbuh," *Pusat Studi Transportasi dan Logistik Universitas Gadjah Mada*: https://pustral. ugm. ac. id/2017/10/05/mengapakendaraan-pribadi-terus-bertumbuh, 2017.

[2] Astra, "Astra International Investor Relations." [Online]. Available: https://www.astra.co.id/investor-relations

[3] Badan Pusat Statistik, "Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis (Unit), 2021-2022." [Online]. Available: https://www.bps.go.id/id/statistics-table/2/NTcjMg==/number-of-motor-vehicle-by-type.html

[4] Z. Wang, L. Hu, F. Wang, M. Lin, and N. Wu, "Assessing the Impact of Different Population Density Scenarios on Two-Wheeler Accident Characteristics at Intersections," *Sustainability*, vol. 16, no. 5, p. 1737, 2024, doi: https://doi.org/10.3390/su16051737.

[5] Q. Ma, G. Huang, and X. Tang, "GIS-based analysis of spatial–temporal correlations of urban traffic accidents," *European transport research review*, vol. 13, pp. 1–11, 2021, doi: https://doi.org/10.1186/s12544-021-00509-y.

[6] N. Sohaee and S. Bohluli, "Nonlinear analysis of the effects of socioeconomic, demographic, and technological factors on the number of fatal traffic accidents," *Safety*, vol. 10, no. 1, p. 11, 2024, doi: https://doi.org/10.3390/safety10010011

[7] P. Trojanowski, A. Trusz, and B. Stupin, "Correlation between accidents on selected roads as fundamental for determining the safety level of road infrastructure," in *Design, simulation, manufacturing: the innovation exchange*, Springer, 2022, pp. 104–113. doi: https://doi.org/10.1007/978-3-031-06025-0_11.

[8] U. Jilani, M. Asif, M. Y. I. Zia, M. Rashid, S. Shams, and P. Otero, "A systematic review on urban road traffic congestion," *Wirel Pers Commun*, pp. 1–29, 2023, doi: https://doi.org/10.1007/s11277-023-10700-0.

[9] S. R. Samal, M. Mohanty, and S. M. Santhakumar, "Adverse effect of congestion on economy, health and environment under mixed traffic scenario," *Transportation in Developing Economies*, vol. 7, no. 2, p. 15, 2021, doi: https://doi.org/10.1007/s40890-021-00125-4.

[10] Y. Ma, V. Sanchez, and T. Guha, "CLIP-EBC: CLIP Can Count Accurately through Enhanced Blockwise Classification," *arXiv preprint arXiv:2403.09281*, 2024, doi: https://doi.org/10.48550/arXiv.2403.09281.

[11] L. Deng, Q. Zhou, S. Wang, J. M. Górriz, and Y. Zhang, "Deep learning in crowd counting: A survey," *CAAI Trans Intell Technol*, doi: https://doi.org/10.1049/cit2.12241.

[12] Y.-J. Zhang, "Computer Vision Overview," *in 3-D Computer Vision: Principles, Algorithms and Applications*, Springer, 2023, pp. 1–35. doi: https://doi.org/10.1007/978-981-19-7580-6_1.

[13] Kemenhub, "Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis." [Online]. Available: https://portaldata.kemenhub.go.id/content/dataset/10030

[14] P. A. Rosyady and M. R. Feter, "Prototype Lampu Lalu Lintas Adaptif Berdasarkan Panjang Antrian Kendaraan Berbasis Arduino Uno," *Circuit: Jurnal Ilmiah Pendidikan Teknik Elektro*, vol. 6, no. 2, pp. 173–186, 2022, doi: http://dx.doi.org/10.22373/crc.v6i2.13748.

[15] W. Hariri, "Unlocking the potential of ChatGPT: A comprehensive exploration of its applications, advantages, limitations, and future directions in natural language processing," *arXiv preprint arXiv:2304.02017*, 2023, doi: https://doi.org/10.48550/arXiv.2304.02017.

[16] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artif Intell Rev*, vol. 57, no. 4, p. 99, 2024, doi: https://doi.org/10.1007/s10462-024-10721-6.

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020, doi: https://doi.org/10.48550/arXiv.2004.10934.

[18] P. Li, M. Zhang, J. Wan, and M. Jiang, "Multi-Scale Guided Attention Network for Crowd Counting," *Sci Program*, vol. 2021, no. 1, p. 5596488, 2021, doi: https://doi.org/10.1155/2021/5596488.

[19] M. I. L. Tan, C. J. Calgo, S. E. P. Cabantac, J. L. E. Honrado, N. J. C. Libatique, and G. L. Tangonan, "Vehicle detection using YOLO and mobility tracking during COVID-19 pandemic lockdowns," in *2021 IEEE Global Humanitarian Technology Conference (GHTC)*, IEEE, 2021, pp. 1–7. doi: https://doi.org/10.1109/GHTC53159.2021.9612481.

[20] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies (Basel)*, vol. 11, no. 2, p. 40, 2023, doi: https://doi.org/10.3390/technologies11020040.

[21] M. S. Azari, F. Flammini, S. Santini, and M. Caporuscio, "A systematic literature review on transfer learning for predictive maintenance in industry 4.0," *IEEE access*, vol. 11, pp. 12887–12910, 2023, doi: https://doi.org/10.1109/ACCESS.2023.3239784.

[22] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020, doi: https://doi.org/10.1109/JPROC.2020.3004555.

[23] S. Chilamkurthy, "Transfer Learning for Computer Vision Tutorial." [Online]. Available: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

[24] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Mach Learn Knowl Extr*, vol. 5, no. 4, pp. 1680–1716, 2023, doi: https://doi.org/10.3390/make5040083.

[25] E. Smith, "Scientific Machine Learning with PyTorch," in *Introduction to the Tools of Scientific Computing*, Springer, 2022, pp. 359–410.

[26] N.-E.-A. Mimma, S. Ahmed, T. Rahman, and R. Khan, "Fruits classification and detection application using deep learning," *Sci Program*, vol. 2022, no. 1, p. 4194874, 2022, doi: https://doi.org/10.1155/2022/4194874.

[27] K. Banachewicz and L. Massaron, *The Kaggle Book: Data analysis and machine learning for competitive data science*. Packt Publishing Ltd, 2022.