JIT Journal of Innovation and Technology
e-ISSN: 2721-8562; p-ISSN: 2721-8570
Vol. 5, No. 1, April 2024. Page: 5 – 10

Intelligent Eyes on the Battlefield: Developing an AI-Vision Based Military Vehicle and Infantry Detection System

Pasha R A Wibowo[1], Khairul Ummah[1,*], Ony Arifianto[1], Djarot Widagdo[1], Akhmad Riszal[2], Yanuar Zulardiansyah Arif[3], Mahardi Sadono[1]

[1]*Faculty of Mechanical and Aerospace Engineering, Institut Teknologi Bandung, Jalan Ganesha 10, Bandung 40123, Indonesia*
[1]*Department of Mechanical Engineering, Faculty of Engineering, Universitas Lampung, Jl. Prof. Soemantri Brojonegoro 1, Bandar Lampung, 35143, Indonesia*
[1]*Department of Electrical and Electronic, Faculty of Engineering, Universiti Malaysia Sarawak (UNIMAS), 94300 Kota Samarahan, Sarawak, Malaysia.*

* Corresponding email: khairulu@gmail.com

Article history
Received: 29.12.2023
Revised: 21.02.2024
Accepted: 29.02.2024

DOI:10.31629/jit.v5i1.7381

Abstract
The importance of accurate, real-time intelligence in modern warfare is crucial, especially in reconnaissance and surveillance operations. Currently, drones are widely used for reconnaissance, but generally rely only on the operator's ability to monitor operation targets. This research is aimed at developing an AI vision assistance system to enhance the ability to detect military vehicles and infantry. The method used is computer vision trained to recognize and differentiate several military objects. The YOLO model is used to detect and distinguish objects. To improve detection capabilities, the YOLO v8 model was retrained with an additional dataset sourced from battle recordings on the battlefield. The results show a detection accuracy rate of 95% in detecting vehicles and infantry under normal visual conditions. The model from this research can be used to enhance the capabilities of reconnaissance drones and the effectiveness of monitoring operations.

Keywords: *YOLO, Image Recognition, Military, Reconnaissance*

1. Introduction

The utilization of unmanned aerial vehicles (UAVs) namely commercial off the shelf style quadcopters in military operations has become increasingly prevalent due to their ability to provide real-time intelligence in combat situations. Traditionally, the effectiveness of these reconnaissance efforts has relied heavily on manual interpretation of aerial imagery. However, the integration of artificial intelligence (AI), particularly advanced object detection systems, has the potential to revolutionize this aspect of modern warfare. AI-driven systems can automate and enhance the detection and monitoring of military assets, providing crucial data more swiftly and accurately.

Despite the advances in UAV technology and AI, there remains a significant challenge in automating the detection process under diverse operational conditions. Many existing AI models struggle with issues such as occlusion, variable lighting, and rapid scene changes, which are common in military environments.

This study aims to address these challenges by developing and evaluating an AI-assisted vision system, employing the latest iteration of the You Only Look Once (YOLO) framework, YOLOv8. This model is designed to improve the accuracy and speed of detecting military vehicles and infantry in drone-captured images, enhancing the operational efficiency of military surveillance.

The research focuses on the application of the YOLOv8 model to drone footage obtained from typical reconnaissance missions, with the goal of detecting various military assets under different visual conditions. The study tests the model's effectiveness in accurately identifying targets in scenarios that simulate real-world operational environments.

1.1 Deep Learning For Object Detection: YOLOV8 Model

A. Overview of Deep Learning in Computer Vision

Deep learning tackles complex mappings by breaking them down into a series of more straightforward mappings, each represented by a different layer within the model. From the layers, a series of hidden information can then be extracted. A pre-trained model is required to determine which hidden information helps describe the relationship of the data [1]. The role of what each layer does is stored in the “weights”, which are just a set of numerical values that the algorithm uses to accurately map the input data to their corresponding targets [2]

B. Evolution of the YOLO

You Only Look Once (YOLO) is one of the many approaches available for object detection in computer vision. This section provides a comprehensive overview of YOLO, from its historical development, underlying mechanism, selection and rationale, and comparative performance overview throughout its versions.

The YOLO (You Only Look Once) series has significantly evolved since its inception in 2015, introducing pivotal advancements in real-time object detection technology. YOLOv1 laid the foundation with its integrated approach to bounding box and class prediction, although it was limited by low recall and higher localization errors. Subsequent versions improved upon this, with YOLOv2 introducing batch normalization and anchor boxes, and YOLOv3 enhancing detection across multiple scales with the Darknet-53 architecture. A major shift occurred with YOLOv4 and beyond, incorporating advanced techniques like CSPDarknet-53 backbone, Mish activation, and the innovative "bag of freebies" optimization strategy. The latest iterations, YOLOv5 through YOLOv7, transitioned to the PyTorch framework and introduced features such as scalable model sizes, anchor-free detection, and enhanced training efficiency aimed particularly at industrial applications, showcasing significant gains in accuracy and processing speed. [3]

Ultralytics YOLOv8, the latest iteration in the YOLO series, represents a substantial evolution in object detection technologies developed by Ultralytics. Building on the foundations set by YOLOv5, YOLOv8 enhances performance and flexibility for a wide range of computer vision tasks including object detection, tracking, segmentation, and image classification. Key integrations with AI platforms like Roboflow enhance its capabilities in dataset training, labeling, visualization, and management, facilitating a more efficient workflow in various applications [4].

C. You Only Look Once (YOLO) version 8

YOLOv8 introduces significant architectural enhancements, particularly in its backbone and network modules. The model utilizes the C2f module (cross-stage partial bottleneck with two convolutions) to more effectively integrate and process high-level features, enhancing accuracy by leveraging contextual information from the background. Additionally, the Spatial Pyramid Pooling Fast (SPPF) feature in the 'Neck' section allows YOLOv8 to handle inputs of varying sizes by pooling features into a fixed-size output, thus accommodating diverse input dimensions without loss in performance. [4]

The structure of YOLOv8 is further optimized by decoupling its 'head' into three primary tasks: objectness determination, classification, and precise bounding box regression. This separation allows each component to specialize, enhancing the overall accuracy and efficiency of the model. The transition from anchor-based to anchor-free detection marks a significant shift, offering improvements in speed and accuracy. Advanced loss functions like CloU improve the alignment of predicted boxes with actual ground truth, solidifying YOLOv8’s utility in real-time applications and setting new benchmarks for performance in the YOLO series. [4]

D. Label What You See Technique

Label what you see technique is a method where a person manually labels visible objects. In this paper, the researchers label visible tomatoes, both ripe and unripe using the graphical image annotation tool “labelImg”. Emphasizing only the visible parts of the tomatoes especially those that are heavily obscured, bounding boxes are drawn based on their visible shape and estimated size. This approach is vital for training models to recognize partially visible tomatoes, a common occurrence in agricultural settings. To enhance accuracy and reliability by ensuring there are no missed annotations.

2. Implementation Procedure

A. Data Gathering

The initial stage involves collecting relevant video footage that serves as the raw data for the project. This footage is sourced from publicly accessible datasets and online platforms such as Telegram channels, Reddit subreddits like r/CombatFootageUkraine and r/UkraineWarVideoReport, and various news outlets. Criteria for video selection include having an isometric or top-down perspective and originating from contemporary conflict zones like the Russian invasion of Ukraine. Videos are assumed to be filmed using commercially available drones. These videos are then meticulously screened to ensure a diverse representation of military vehicles and infantry across different environmental settings. Each selected video is downloaded in .MP4 format, ensuring a broad spectrum of conditions such as variations in illumination, occlusions, and overlaps of targets are included. A total of 18 videos are gathered, with three reserved specifically for detailed analysis and the remaining 15 earmarked for splicing and use in model training.

B. Dataset Construction

In the second phase, the collected videos are processed to construct a robust dataset. This involves splicing the videos into individual frames, which are then annotated and augmented to prepare them for the training phase. The annotation process is facilitated by Roboflow, a platform designed to streamline the management and annotation of datasets for computer vision applications. Each image frame is carefully annotated to identify and label all visible military assets, employing techniques such as 'Label What You See' (LWYS) to ensure accuracy. The annotations include precise bounding box coordinates for each object of interest, reflecting the meticulous detail required for effective model training. Following annotation, the images undergo augmentation processes to enhance the dataset's robustness, including adjustments to resolution, grayscale modifications, contrast auto-adjustment, and randomized hue shifts.

C. Model Training

With a prepared dataset, the next stage involves training the YOLOv8 model using the Ultralytics framework within the Google Colab environment. This stage is critical as it involves setting up the computational environment, importing necessary libraries like numpy for numerical operations and cv2 for image processing, and configuring the training parameters detailed in the dataset configuration files. Training the model involves loading the pre-annotated and augmented images and employing advanced machine learning techniques to teach the model how to accurately detect and classify various objects within the frames. The model is trained over several iterations, adjusting parameters to optimize accuracy, reduce false positives, and increase the reliability of the system under various operational conditions.

D. Target Detection

The final stage tests the trained model's effectiveness by applying it to new, unseen video inputs. This phase is crucial for evaluating the model's practical application in real-world scenarios. The model's performance is assessed based on its ability to accurately identify and classify military vehicles and infantry in video frames. This involves processing the video through the model, which detects targets and classifies them according to the learned characteristics. The results are then analyzed to determine the model's accuracy, speed, and reliability in detecting military assets, providing valuable insights into its potential deployment in military reconnaissance operations.

3. Result and Discussion

A. Visualization of output

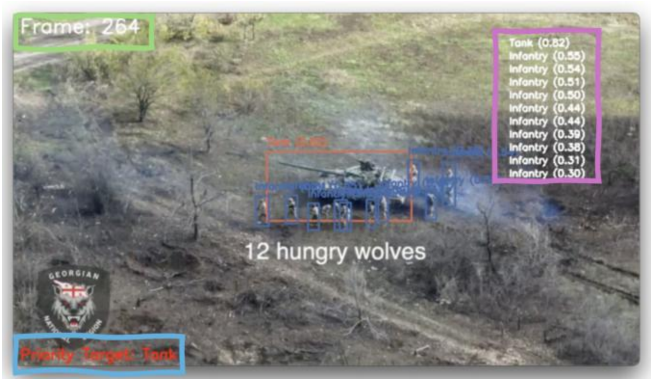

Fig. 1 Visualization of Output

Above is the output visualization of the bounding boxes. Moving from the top left corner of the screen, highlighted using the green box, is the frame index of the current image being shown in the video. The frame index indicates the specific sequence within the video sequence; in this case, the number '264' means that the image being shown is the 264th frame in the video. Moving to the bottom left of the display, highlighted by the blue box, is the priority target currently visible, which in this case is labeled as 'Priority Target: Tank.' On the right side of the screen, highlighted by the purple box, is a list showing all the objects currently being detected within the frame. The text displayed includes labels such as 'Tank (0.82)' and various instances of 'Infantry' with their respective confidence scores.

B. Analysis and Interpretation of Sample Video 1

In, sample video 1, it shows twelve (12) infantry and one (1) tank, throughout the approximately 13 seconds of the video, as the video has a frame rate of 13 seconds the expected occurrence to have 304 instances of "Tank" and 3,648 instances of "Infantry".

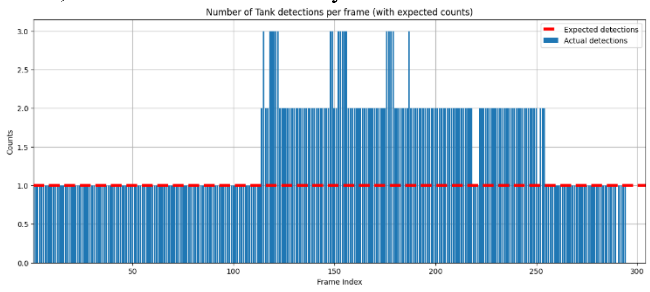

Fig. 2. Number of tank detections per frame (with expected count) for video sample 1

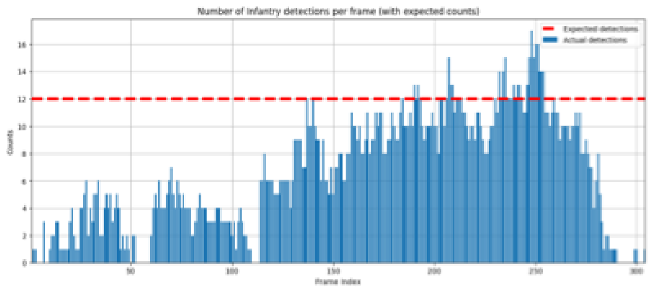

Fig. 3. Number of infantry detections per frame (with expected count) for video sample 1

Table I. NON-NMS RESULT FOR VIDEO SAMPLE 1

True Positive	2214	Precision	0.91791045
False Positive	198	Recall	0.57342657
False Negative	1647	F1	0.70588235

Table II. NMS RESULT FOR VIDEO SAMPLE 1

True Positive	1906	Precision	0.97096281
False Positive	57	Recall	0.49365449
False Negative	1955	F1	0.65453297

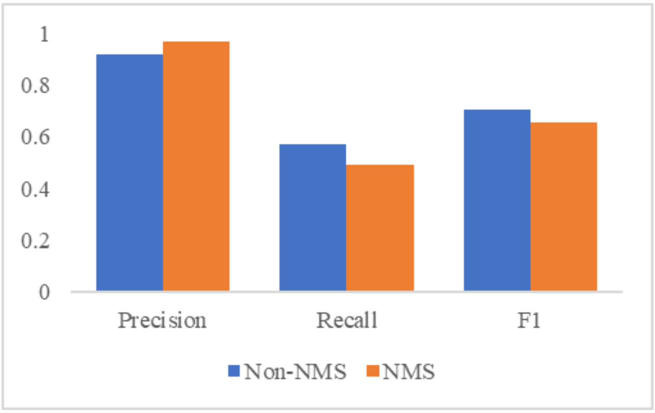

Fig. 4. Video sample 1 Non-NMS vs NMS Model Performance

It can be seen that overall the model is very robust with accuracy and precision above 90% on both instances. some things to note are that firstly, precision increases after NMS implying that the model is becoming more precise and conservative meaning the model is making less false positive detections. However, the recall value drops after NMS which means the model is missing more positive instances. Some possible explanation for this is that the increase in precision after NMS post-processing suggests that the model is successful in removing overlapping or redundant detections leading in a reduction of false positives, however the decrease in recall means that in the process of removing redundant bounding boxes, a lot of true positives are coincidentally removed. In the case of video sample 1 this is possibly due to the occlusion of objects with the same "case" as it shows 12 infantries crowded on a tank. Overall it can be seen that NMS post processing causes the model to be more conservative.

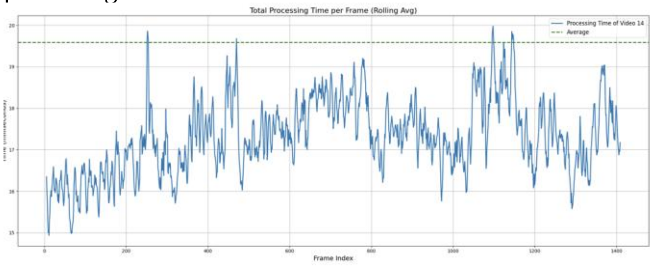

Fig. 5. Rolling average processing time of 5 frames for video sample 1

Fig. 5 shows the processing time taken for each frame in video 14, as the video is 24 frames per second which means that 42 milliseconds and it can be seen that it on average takes 19.5 milliseconds which is less than the 42 milliseconds threshold and way below the 93 milliseconds for real time detection. A snippet of video 14 can be seen in figure 4.7.

C. Analysis and Interpretation of video sample 2

Video sample 2 shows a variety of conditions which can be segmented and summarized as shown in table 4.5

Table III. Detection summary of video sample 2

Frame index		Detection
Start	End	
1	80	2 "Tank" 1 "IFV"
81	212	1 "Tank"
213	301	No detection
302	353	1 "Tank" 1 "IFV"

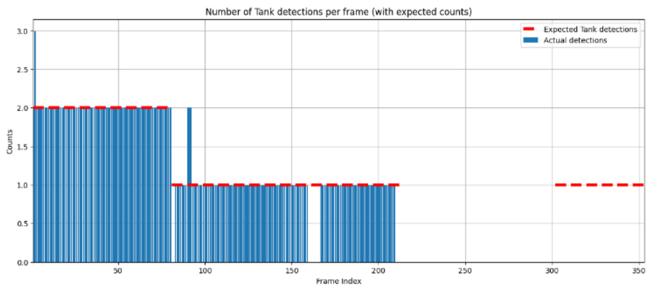

Fig. 6. Number of tank detections per frame (with expected count) for video sample 2

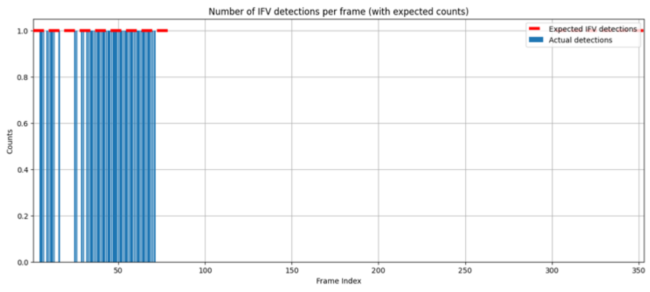

Fig. 7. Number of IVF detections per frame (with expected count) for video sample 2

Table IV. NON-NMS RESULTS FOR VIDEO SAMPLE 2

true positive	333	Precision	0.98813056
false positive	4	Recall	0.91483516
false negative	31	F1	0.95007133

Table V. NMS RESULTS FOR VIDEO SAMPLE 2

true positive	333	Precision	0.99107143
false positive	3	Recall	0.91483516
false negative	31	F1	0.95142857

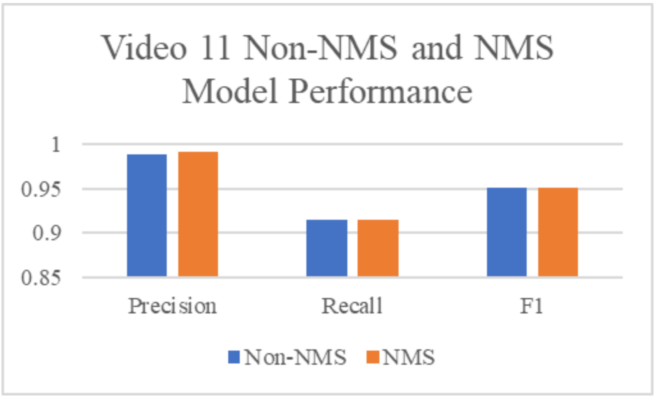

Fig. 8. Video sample 2 Non-NMS and NMS model Performance

In the case of video 11, where it shows 3 vehicles with no obstruction or occlusion, the precision is high at 98.81% before NMS and 99.1% after NMS indicating that the model's prediction is very accurate even without NMS. The recall values for both instances remain unchanged at 91.48% suggesting that the model is consistently identifying high proportion of true positives throughout the entire video. The same goes with the F1 score, showing no significant changes before and after NMS indicating that the model is providing accurate and reliable detections, and that NMS does not significantly change the performance of the model

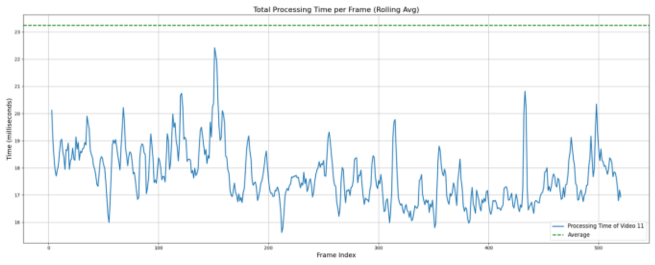

Fig. 9. Rolling average processing time of 5 frames for video sample 2

Same as before, the model average processing time does not exceed 23 milliseconds and with a video with a frame rate of 24 frames per second, the threshold in that it can be done real time is 42 milliseconds a video, still much higher than the actual detection rate.

4. Conclution

This research has demonstrated the capability of a machine vision system employing the YOLOv8 algorithm, augmented with custom weights, to enhance the detection of vehicles and infantry from footage captured by commercial-grade drones. The findings articulated in the previous section of this paper confirm that the precision of the developed model surpasses the baseline requirement of 78.21%, achieving a precision rate of 90% in all evaluated scenarios. For optimal detection, each object must maintain a minimum size of 83x83 pixels. Moreover, the system's processing speed is sufficiently rapid for real-time applications, as evidenced by its ability to process images in under 23 milliseconds—quicker than the duration a single frame is displayed in a standard video at 24 frames per second. This efficiency supports the potential utility of the system in operational environments where timely data processing is critical.

Acknowledgment

This research owes its completion to the unwavering support and encouragement of several key figures in my life. I am profoundly grateful to my parents, affectionately referred to as Mum and Pops, whose steadfast belief in my abilities has been a pillar of strength and motivation throughout this endeavor. Additionally, I extend my gratitude to my siblings for their patience, understanding, and the moments of laughter and respite they provided during challenging times. I would also like to express my deep appreciation to my advisors, Mr. Khairul Ummah and Mr. Ony Arifianto, for their invaluable guidance and insights throughout the research process. Their expertise and mentorship were crucial in shaping both the direction and success of this work. Their collective support was instrumental in navigating the journey of this research.

References

[1] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, The MIT Press, 2016.

[2] C. François, Deep Learning with Python, Manning, 2017.

[3] H. Muhammad, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," *Machines,* pp. 667-693, 2023.

[4] J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO," *Machine Learning and Knowledge Extraction,* 2023.

[5] A. Z. M. J. Mariya Yao, Applied Artificial Intelligence: A Handbook For Business Leaders, TOPBOTS Incorporated, 2018.

[6] Y. B. a. A. C. Ian Goodfellow, Deep Learning, Cambridge: The MIT Press, 2016.

[7] r. Chollet, Deep Learning with Python, Manning Publications, 2017.

[8] R. Szeliski, Computer Vision: Algorithms and Applications (Texts in Computer Science) 2011th Edition, Springer, 2010.

[9] S. J. D. Prince, Computer Vision: Models, Learning, and Inference, Cambridge University Press, 2012.

[10] V. H. ,. R. B. Milan Sonka, Image Processing, Analysis and Machine Vision, CL Engineering, 2007.

[11] A. B. a. C.-Y. W. a. H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020.

[12] S. D. R. G. A. F. Joseph Redmon, "You Only Look Once: Unified, Real-Time Object Detection," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2015.

[13] R. E. Dupré, "Guide to Imagery Intelligence," *The Intelligencer Journal of U.S. Intelligence Studies,* vol. 18, pp. 61-64, 2011.

[14] C. Carlos and C. Carolina, "A Method to Detect Victims in Search and Rescue Operation Using Template Matching," *IEEE International Safety, Security, and Rescue Robotics, Workshop,* 2005.

[15] O. L. Lawal, "Tomato Detection Based On Modified YOLOv3 framework," *Scientific Reports,* 2021.

[16] S. Abhishek, D. Shubhra and B. Anupama, "Object Detection for Autonomous Driving using YOLO algorithm," *2nd International Conference on Intelligent Engineering and Management,* pp. 447-451, 2021.

[17] K. Vijay and D. Bala, Data Science: Concepts and Practice 2nd Edition, Morgan Kaufmann, 2018.

[18] C. François, Deep Learning with Python, Manning Publications, 2017.

[19] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, 2011.

[20] M. Sonka, V. Hlavac and R. Boyle, mage Processing, Analysis, and Machine Vision 3rd Edition, CL Engineering, 2007.

[21] S. J. D. Prince, Computer Vision 1st Edition, Cambridge University Press, 2012.

[22] N. Ketkar, Introduction to PyTorch. In: Deep Learning with Python, Bangalore: Apress, Berkeley, CA, 2017.