Implementation of The YOLOv5 Model For Crowd Detection : Model Analysis

Erna Astriyani[1], Oleh Soleh[2], Adisa Lahmania Putri[*3], Fachry Riziq Huseini[4]
[1,2,3,4] Information Systems Faculty of Science and Technology, University of Raharja, Indonesia
E-mail: [1]erna.astriyani@raharja.info, [2]oleh.soleh@raharja.info, [*3]**adisa@raharja.info**, [4]fachry.huseini@raharja.info

Abstract

One of several deep learning object detection model ideas is YOLO. High-level characteristics obtained by Convolutional Neural Networks (CNN) can be used to overcome the challenges of prior assumptions. Because of the inherent complexity of the human crowd, prominent places in a high-density crowd may exhibit characteristics that differ from those in a normal density crowd. The most efficient deep learning model architecture for use in real-time video applications is provided by YOLO. The CrowdHuman dataset is annotated by researchers using the Roboflow platform. The outcomes are annotated files in the YOLOv5 format including around 4,500 pictures. Photographs of crowds at various elevations and distances are included in this image dataset. The Roboflow platform is used by researchers to convert this annotation and get the required results. This research is divided into four stages: the evolving process, the training process, the testing process, and the evaluation process. The training results are quite positive because they have a mAP value larger than 0.4. However, during the evaluation phase, the model continues to have issues detecting small and moving objects. The device used is a YOLOv5s, a smaller version of the YOLOv5.

Keywords *— YOLOv5 Model, CrowdHuman, CNN, Object Detection, Crowded*

1. INTRODUCTION

Handcrafted feature maps that use prior knowledge such as human face attributes as well as low-level features such as color, contrast, and intensity as assumptions for the majority of human crowd image saliency models [1]–[3]. Convolutional Neural Network (CNN) obtained high-level features can be utilized to overcome the difficulties of prior assumption and employing just low-level features [4]. When using CNN as a high-level feature extraction technique, the majority of existing saliency models predict saliency in a crowd of uniformly distributed people. Because of the intrinsic complexity of the human crowd [5], the salient areas in a high-density human crowd may have distinct traits from those in a standard density crowd. The outcome of the human visual system mechanism [6], [7] can be used to predict saliency in a human crowd image, allowing for the identification of attention-grabbing factors, which is where object detection comes in helpful. Object detection [8]–[10], a critical component of computer vision, involves locating examples of a certain class of visual objects within digital images. Image classification [11], human behavior analysis [12], [13], face recognition [14]–[16], autonomous driving [17], instance segmentation [18], [19], image captioning [20], [21], and object tracking [22]–[24] in digital images all rely on effective object identification, making it one of computer vision's key difficulties. YOLO [25]–[28] is one of many deep learning object detection

model designs. YOLO provides the most modern and efficient deep learning model architecture for use in real-time video applications. Using the R-CNN technique [29], [30], YOLO can handle real-time video quite successfully. Despite the fact that its development was discontinued, YOLO is still used in computer vision by other researchers. Moving to the Pytorch framework [31], [32] dramatically increases the performance of the YOLO design. NVidia graphics processing performance can be increased with the help of CUDA technology [33]–[36].

2. RESEARCH METHOD

In the section entitled Research Approaches (which can include analysis, architecture, problem-solving methods, and implementation), the author can outline how the research will be conducted.

2.1. Datasets

The CrowdHuman dataset [37] was used in this work, and the annotations were translated into the YOLOv5 format. Researchers use the Roboflow platform [38] to convert this annotation and obtain the desired results. Furthermore, the preprocessing and augmentation operations are performed using the Roboflow platform. The results are annotated datasets in the YOLOv5 format including around 4,500 photos.

Figure 1. Sample Dataset: (a) CrowdHuman Sample, (b) Annotated CrowdHuman sample, (c) Augmented CrowdHuman Sample 1, (d) Augmented CrowdHuman Sample 2

This image dataset includes photographs of crowds at various levels and distances. Furthermore, using photographs after augmentation is highly beneficial in altering the existing images so that the model can study the dataset further with a range of varying scenarios. Furthermore, to evaluate the performance capabilities of YOLOv5, real-time video data from locations around the researcher's environment is employed, as well as data from the open platform Youtube [39].

2.2. YOLO Architecture

R-CNN might be enhanced in a variety of ways, according to some. You only look once (YOLO) [40], which uses fixed-grid regression to detect objects, and fast R-CNN [41], which optimizes classification and bounding box regression at the same time, are two of these methods. Both of these algorithms are proposed enhancements to R-CNN. When compared to the baseline R-CNN, all of these versions significantly enhance detection performance, paving the way for immediate and exact object detection in real time [42], [43]. Joseph et al. were the first to coin the phrase "YOLO" in 2015. That detector was groundbreaking in terms of deep learning technology at the time. The name suggests that the authors will use the traditional detection paradigm of "proposal detection + verification," however they have rejected this technique totally [44]. The YOLO algorithm divides the data images to be investigated into SxS grids before assigning various detection tasks to each grid individually. Although the YOLO algorithm is effective at quickly locating things, it struggles when those targets are small in size. The rationale behind this is because if the grid is not divided into smaller portions, a huge number of targets might be located within a single grid. The YOLO-v5 algorithm makes use of the data loader to improve the quality of the training data as it is being transmitted. The You Only Look Once (YOLO)v5 model is a part of the YOLO family of computer vision models. Small (s), medium (m), large (l), and extra large are the four basic YOLOv5 versions (x) [45]. The accuracy rates rise from small to large to extra large, and each version has its own training time. Small (s), medium (m), large (l), and extra large are the sizes available (x) YOLOv5 is intended for use in applications that require features to be extracted from input photographs, such as object detection. These characteristics are the information that a prediction system takes into account before making predictions about the classes to which particular objects belong.

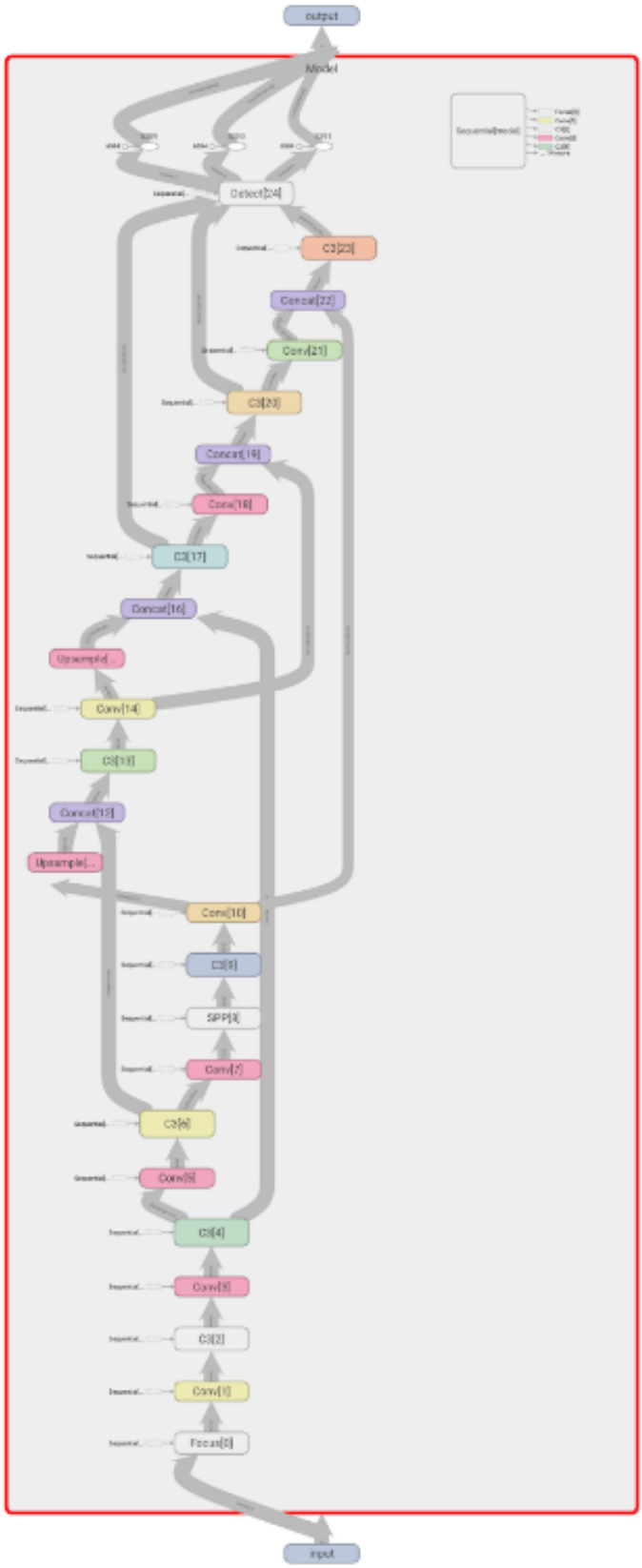

Figure 2. YOLOv5s Structure Architecture[46]

The three basic architectural building parts that comprise the YOLO family of models are the Backbone, the Neck, and the Head [46]. The YOLOv5 Backbone is a convolutional

neural network that accumulates and produces image features at various granularities. It accomplishes this by using CSPDarknet as the backbone for feature extraction from pictures constructed of cross-stage partial networks. YOLOv5 Neck is a series of layers that combine and integrate various picture data before sending it to the prediction step. It aggregates the features using PANet to build a feature pyramid network, and then delivers that information to Head for prediction. Consumes neck features and performs box and class prediction steps. Head YOLOv5. Layers that generate object detection predictions based on anchor boxes. Other than this one, the following alternatives are used for training in YOLOv5. The activation and optimization process: In YOLOv5, leaky ReLU and sigmoid activation are used, and the optimizer options SGD and ADAM are provided [45]. For its loss function, it employs binary cross-entropy in conjunction with logits loss.

2.3. Pytorch

PyTorch [47] is an efficient Deep Learning tensor library based on Python and Torch that is mostly used for GPU and CPU applications. PyTorch is preferred over other Deep Learning frameworks such as TensorFlow and Keras because it uses dynamic computation graphs and is entirely Pythonic [48], [49]. It enables scientists, developers, and neural network debuggers to execute and test code segments in real time. As a result, users do not have to wait for the complete code to be implemented before determining whether a portion of the code works or not. YOLOv5 by Ultralytics is the first large-scale implementation of YOLO in PyTorch, making it more accessible than ever before.

3. RESEARCH RESULTS AND DISCUSSION

Researchers attempted to apply the YOLOv5 model to a variety of congested settings in this investigation. The YOLOv5 architecture employed in this work comes in a tiny variant known as YOLOv5s. Researchers conducted their investigation using laptops equipped with AMD Ryzen 9 5900HS processor specifications, 16 GB RAM capacity, and a 4 GB NVidia RTX 3050 graphics processor. In general, its implementation is divided into three major processes: training, testing, and assessing.

3.1. Training Process

The training procedure is carried out by conducting training on the architecture employed as well as the available weights. The CrowdHuman dataset has been translated into YOLOv5 format for this training phase. To achieve the best results, the model is run with a maximum of 500 epochs. The model can operate successfully in the YOLOv5 model with a batch-size parameter configuration of 32 values, an image size of 256x256 pixels, and the Adam optimizer. However, before conducting the training process, the evolve process is done to fine-tune the YOLOv5s architecture's hyper-parameters. The evolve procedure is repeated for ten epochs in order to obtain a minor adjustment to the hyper-parameters that will be used in the training process.

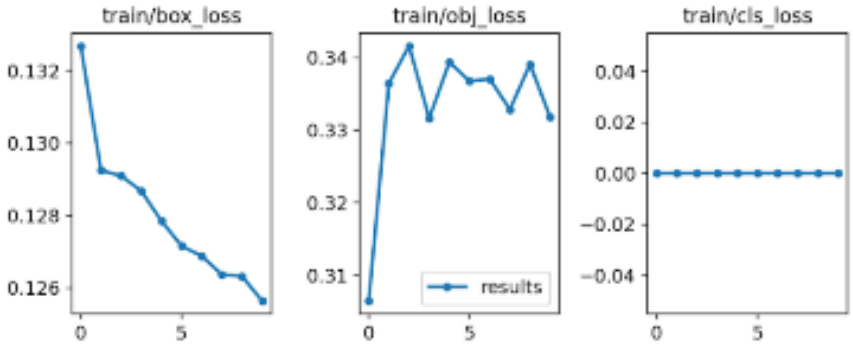

Figure 3. Evolve Process Result

When viewed through the graphic image of the evolve process outcomes above, the results of the adjustment of the 10 epochs are considered extremely satisfactory. The evolve process results in a minor bounding-box loss and an objectness loss of 0.33. The cls_loss (classification loss) findings are stated out to be 0 because there is only one item utilized for research in this study, so it cannot be seen how much loss is produced because it requires a comparison of other objects for its assessment. The training phase can be started after acquiring the hyper-parameter configuration from the evolve process. The training procedure is carried out using an image size of 256x256 pixels, a batch size of 32, and a maximum of 500 epochs. In addition, the researcher set the patience value to 50 epochs to avoid bottlenecks. The model successfully iterated for 50 epochs once the training procedure was completed, as shown in the graphic below.

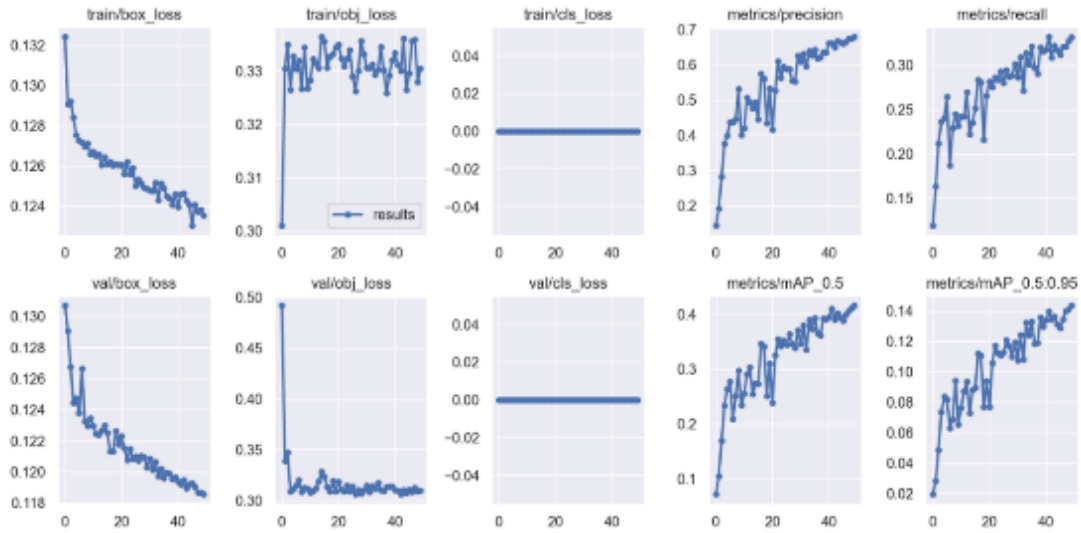

Figure 4. Training Process Result

The model succeeded in lowering the box loss value below 0.125, as seen in the graphic depiction of the training results above. With such a low box loss value, the model can accurately define the anticipated object. The accuracy graph, which continues to rise until it reaches a value of 0.7 with only 50 epochs, demonstrates the model's capacity to mark objects. Furthermore, the resulting objectness loss is quite low, with a range of values ranging from 0.32 to 0.4 and a recall value exceeding 0.3, indicating that the model is not overfitting because the value is not far from the precision value. The low recall value is the result of a

trade-off made during the training phase. This trade-off occurs because the model utilized is designed to properly anticipate the object under investigation, therefore it favors precision above recall. However, if the recall value is too low in comparison to the precision value, overfitting will occur and the model will not learn anything from the training phase. On the other hand, based on the mAP value achieved, the model utilized is fairly good. With only 50 epochs and a few changes, the model was able to achieve a mAP greater than 0.4. This result is sufficient to apply the model to the next process.

3.2. Testing Process

The testing method follows, and it is carried out on the testing dataset that was provided using the CrowdHuman dataset that was not used during the training phase. The findings of this test reveal various limits of the model utilized, including the algorithm's ability to detect humans only within specified ranges.

(a) (b)

Figure 5. Testing Result 1: (a) Image Parallel to Camera, (b) Image on eagle's point of view

According to the findings of the inference, the model detects people well when the image is parallel to the camera and does not surpass the eagle's point of view. The model also attempts to achieve good inference outcomes in various visual circumstances.

(a) (b)

(c)

Figure 6. Testing Result 2: (a) Extrem Dense Crowded Image, (b) Overlapping Inference Result, (c) Failing Inference Result

Even in extremely dense crowds, the model has difficulties differentiating people in the image. The model still recognizes persons in crowd image b, and there is significant overlapping that is difficult to define. When the model is asked to offer a confidence value in the same crowd situations, it can only detect a few persons, not all of them.

3.3. Evaluating Process

The assessing procedure is carried out by doing model inference on video data under various settings, particularly on real video data recorded directly at various locations under various conditions and video data from the open platform Youtube. There are three types of video data in the real video data used: remote video, close-up video, and dynamic video.

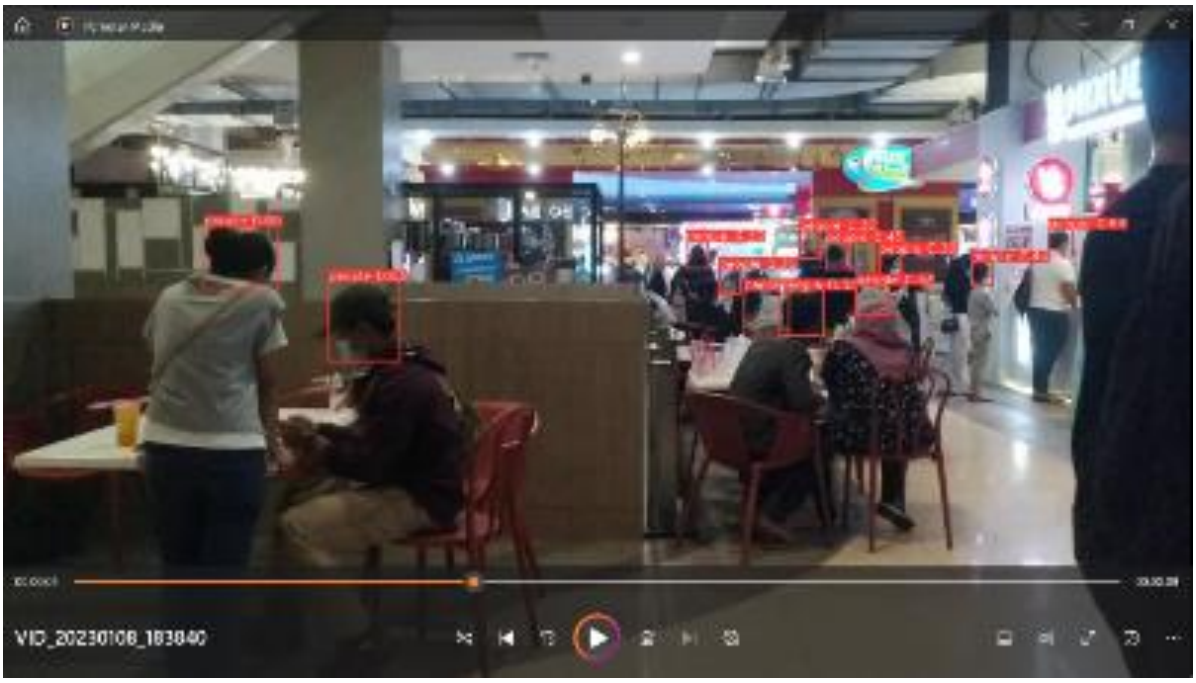

Figure 7. Evaluating Result 1: Indoor 1

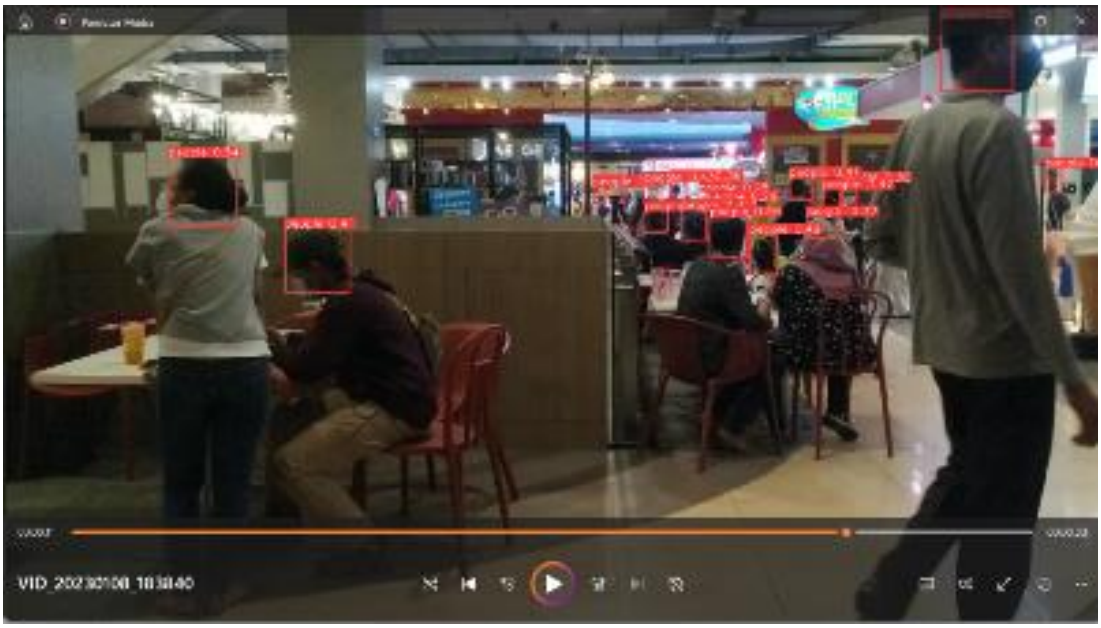

Figure 8. Evaluating Result 1: Indoor 2

In close-up footage, the model detects humans quite well, finding about 90% of the persons in the scene being forecasted. When the inference results are presented under brightness levels equal to room lighting, the model can detect humans in remarkable detail. According to Figure 7, the model detects people quite well but has difficulties detecting people who are very close to the camera, such as items on the right side of the image. Furthermore, the model has difficulties distinguishing objects that are far away with a faint color against the background, such as objects in black clothes and hijabs queuing on the image's back right side. However, as illustrated in figure 8, the model can detect people and objects in adequate detail even at a distance that is difficult for the camera to reach.

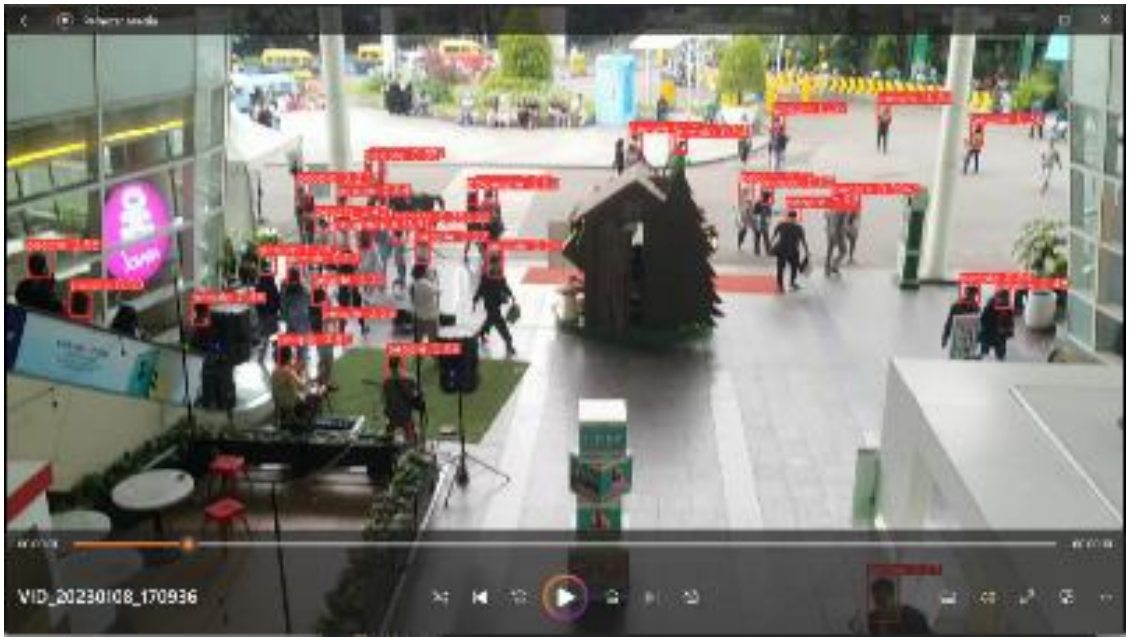

Figure 9. Evaluating Result 2: Outdoor 1

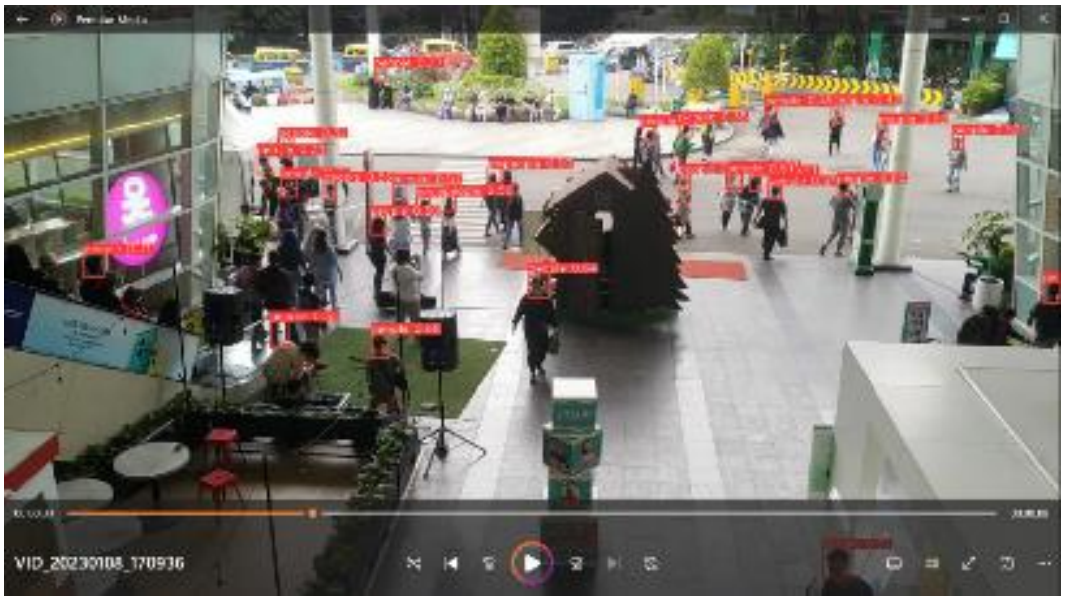

Figure 10. Evaluating Result 2: Outdoor 2

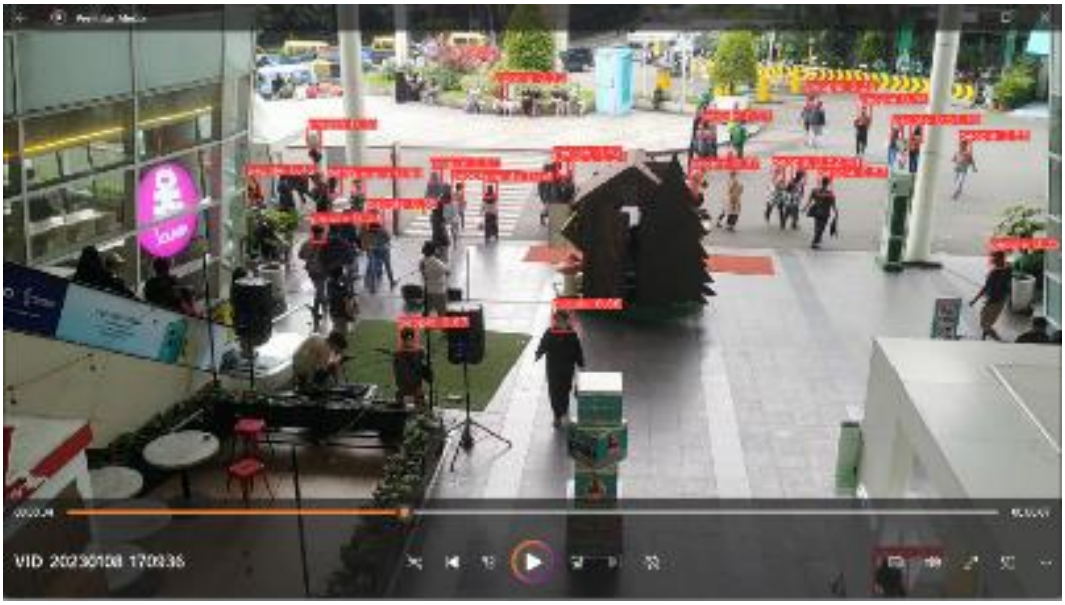

Figure 11. Evaluating Result 2: Outdoor 3

In addition, the model is utilized to infer remote video data in low-light settings. The model detects image objects in picture objects that have enough brightness and object distance from the camera. Furthermore, the background object influences the inference process. According to the results of distant inference figure 10, the model is still attempting to recognize people sitting in the garden with a plant background. Because the model does not investigate the condition of the object with a plant background, determining this condition is a little tricky. Furthermore, things with poor lighting conditions make it harder for the model to detect these objects. The remote inference results figure 11 show that the model is still unable to distinguish people on the escalator to the left of the image under low-light settings. The object's condition in this lighting causes it to merge with the background, making it harder for the model to detect. However, on this remote video data, the YOLOv5s model applied has very good inference performance.

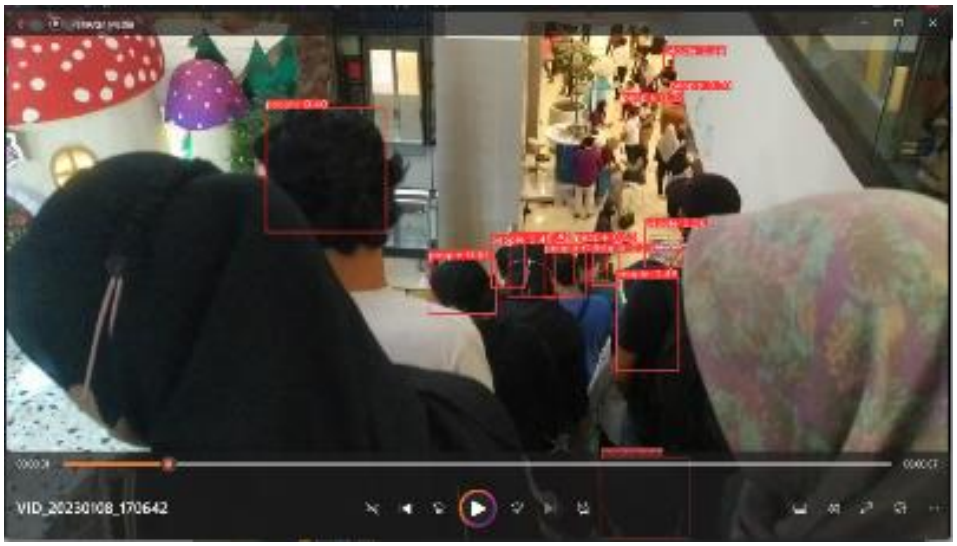

Figure 12. Evaluating Result 3: Dinamic 1

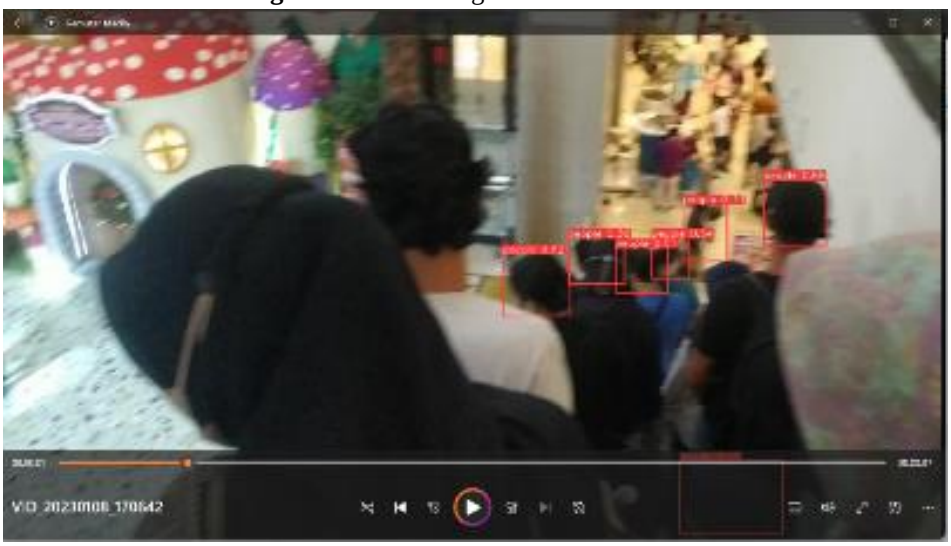

Figure 13. Evaluating Result 3: Dinamic 2

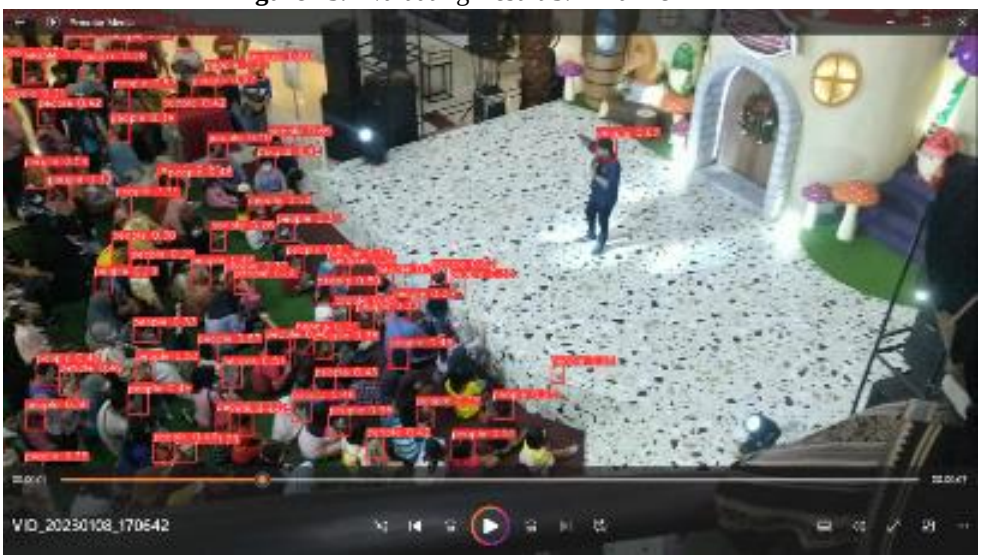

Figure 14. Evaluating Result 3: Dinamic 3

Figure 15. Evaluating Result 3: Dinamic 4

The model is then applied to dynamic video data where the camera captures in motion. According to the dynamic video inference results, the model can distinguish multiple human items in front of the camera, but it cannot detect objects that are too close. Furthermore, models that are too far away or have differing lighting conditions cannot be correctly identified. This condition may be seen in the dynamic inference findings figure 12 and 13. Furthermore, when the camera is focused at long-distance crowd situations, the model detects the crowd very effectively, with a detection performance of 90% of people objects correctly detected according to dynamic inference results 14 and 15.

Figure 16. Evaluating Result 4: Public Data 1

Figure 17. Evaluating Result 4: Public Data 2

Finally, the model is tested using crowd video data from YouTube (https://youtu.be/29uoGUIhFxU). The model's performance is very good, as demonstrated in the video data inference figure 16, since it can detect a huge number of objects. According to the inference results in figure 16, the model still has trouble detecting items that are very small in size or that are positioned far from the camera and are too crowded. Furthermore, the model used is still incapable of detecting quickly moving objects. The model fails to detect someone who travels swiftly, as evidenced by the inference findings figure 17 above. As a result, the model still requires additional in-depth training and evolution so that it can adapt to the state of the object under all scenarios.

4. CONCLUSION

The purpose of this research is to put the YOLOv5 model's architectural performance to the test under a variety of scenarios. The device utilized is the YOLOv5s, which is a smaller version of the YOLOv5. The CrowdHuman dataset, which contains 4500 photos of humans, was used for this training. The model goes through four stages: the evolving process, the training process, the testing process, and the evaluation process. The development process is repeated for ten epochs to achieve a hyper-parameter configuration that is appropriate for the problem under consideration. Furthermore, the training procedure was repeated 50 times. The training results are pretty encouraging because they have a fairly high mAP value that is greater than 0.4. Furthermore, the precision and recall levels are both greater than 0.7 and 0.3, respectively, indicating that they can correctly detect the object under study. Following that, the testing was done utilizing CrowdHuman data that had not been used during the training procedure. The testing results suggest that the model can recognize items in both indoor and outdoor environments. However, model testing results still show limitations in recognizing objects that are excessively crowded and tiny. Finally, the model goes through an evaluation process to see how well it performs on real video data. During the evaluation process, the model still has problems detecting small and moving objects. Furthermore, the model has trouble detecting items with varying lighting backgrounds, as well as backdrop images that are

overly complex, such as vegetation. However, the performance of the generated model with only 50 epochs is extremely good for application. Researchers will use the generated model for security demands and other purposes in the future.

5. REFERENCES

[1] A. M. Obeso, J. Benois-Pineau, M. S. García Vázquez, and A. Á. R. Acosta, "Visual vs internal attention mechanisms in deep neural networks for image classification and object detection," Pattern Recognit, vol. 123, p. 108411, Mar. 2022, doi: 10.1016/J.PATCOG.2021.108411.

[2] Y. Zhao, X. Yu, Y. Gao, and C. Shen, "Learning discriminative region representation for person retrieval," Pattern Recognit, vol. 121, p. 108229, Jan. 2022, doi: 10.1016/J.PATCOG.2021.108229.

[3] K. Boudjit and N. Ramzan, "Human detection based on deep learning YOLO-v2 for real-time UAV applications," https://doi.org/10.1080/0952813X.2021.1907793, vol. 34, no. 3, pp. 527–544, 2021, doi: 10.1080/0952813X.2021.1907793.

[4] Y.-H. ; Wang, W.-H. Su, Y.-H. Wang, and W.-H. Su, "Convolutional Neural Networks in Computer Vision for Grain Crop Phenotyping: A Review," Agronomy 2022, Vol. 12, Page 2659, vol. 12, no. 11, p. 2659, Oct. 2022, doi: 10.3390/AGRONOMY12112659.

[5] M. U. Farooq, M. N. M. Saad, and S. D. Khan, "Motion-shape-based deep learning approach for divergence behavior detection in high-density crowd," Visual Computer, vol. 38, no. 5, pp. 1553–1577, May 2022, doi: 10.1007/S00371-021-02088-4/METRICS.

[6] M. H. Guo et al., "Attention mechanisms in computer vision: A survey," Comput Vis Media (Beijing), vol. 8, no. 3, pp. 331–368, Sep. 2022, doi: 10.1007/S41095-022-0271-Y/METRICS.

[7] V. Saravanan, R. D. J. Samuel, S. Krishnamoorthy, and A. Manickam, "Deep learning assisted convolutional auto-encoders framework for glaucoma detection and anterior visual pathway recognition from retinal fundus images," J Ambient Intell Humaniz Comput, pp. 1–11, Jan. 2022, doi: 10.1007/S12652-021-02928-0/METRICS.

[8] W. Chen et al., "A Simple Single-Scale Vision Transformer for Object Detection and Instance Segmentation," pp. 711–727, 2022, doi: 10.1007/978-3-031-20080-9_41/COVER.

[9] T. Diwan, G. Anirudh, and J. v. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," Multimed Tools Appl, pp. 1–33, Aug. 2022, doi: 10.1007/S11042-022-13644-Y/TABLES/7.

[10] S. Antonelli et al., "Few-Shot Object Detection: A Survey," ACM Computing Surveys (CSUR), vol. 54, no. 11s, pp. 1–37, Sep. 2022, doi: 10.1145/3519022.

[11] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," Neurocomputing, vol. 469, pp. 28–51, Jan. 2022, doi: 10.1016/J.NEUCOM.2021.10.021.

[12] N. F. Greenwald et al., "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning," Nature Biotechnology 2021 40:4, vol. 40, no. 4, pp. 555–565, Nov. 2021, doi: 10.1038/s41587-021-01094-0.

[13] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," Future Generation Computer Systems, vol. 135, pp. 364–381, Oct. 2022, doi: 10.1016/J.FUTURE.2022.05.014.

[14] S. Laith, F. S. Tahir, and A. A. Abdulrahman, "Effectiveness of new algorithms for facial recognition based on deep neural networks," International Journal of Nonlinear Analysis and Applications, vol. 13, no. 1, pp. 2171–2178, Jan. 2022, doi: 10.22075/IJNAA.2022.5912.

[15] L. Nanni, A. Manfè, G. Maguolo, A. Lumini, and S. Brahnam, "High performing ensemble of convolutional neural networks for insect pest image detection," Ecol Inform, vol. 67, p. 101515, Mar. 2022, doi: 10.1016/J.ECOINF.2021.101515.

[16] H. N. Vu, M. H. Nguyen, and C. Pham, "Masked face recognition with convolutional neural networks and local binary patterns," Applied Intelligence, vol. 52, no. 5, pp. 5497–5512, Mar. 2022, doi: 10.1007/S10489-021-02728-1/FIGURES/8.

[17] V. M. Deshmukh, R. B, G. B. Krishna, and G. Rudrawar, "An Overview of Deep Learning Techniques for Autonomous Driving Vehicles," pp. 979–983, Feb. 2022, doi: 10.1109/ICSSIT53264.2022.9716433.

[18] W. Quantized Semantic, A. Ahamad, C.-C. Sun, and W.-K. Kuo, "Quantized Semantic Segmentation Deep Architecture for Deployment on an Edge Computing Device for Image Segmentation," Electronics 2022, Vol. 11, Page 3561, vol. 11, no. 21, p. 3561, Oct. 2022, doi: 10.3390/ELECTRONICS11213561.

[19] K. Gao et al., "A region-based deep learning approach to instance segmentation of aerial orthoimagery for building rooftop extraction," Geomatica, vol. 75, no. 1, pp. 148–164, 2022, doi: 10.1139/GEOMAT-2021-0009.

[20] Z. Zohourianshahzadi and J. K. Kalita, "Neural attention for image captioning: review of outstanding methods," Artif Intell Rev, vol. 55, no. 5, pp. 3833–3862, Jun. 2022, doi: 10.1007/S10462-021-10092-2/METRICS.

[21] P. J. Chun, T. Yamane, and Y. Maemura, "A deep learning-based image captioning method to automatically generate comprehensive explanations of bridge damage," Computer-Aided Civil and Infrastructure Engineering, vol. 37, no. 11, pp. 1387–1401, Sep. 2022, doi: 10.1111/MICE.12793.

[22] FanZhaoxin, ZhuYazhi, HeYulin, SunQi, LiuHongyan, and HeJun, "Deep Learning on Monocular Object Pose Detection and Tracking: A Comprehensive Overview," ACM Comput Surv, vol. 55, no. 4, Nov. 2022, doi: 10.1145/3524496.

[23] L. Vaquero, V. M. Brea, and M. Mucientes, "Tracking more than 100 arbitrary objects at 25 FPS through deep learning," Pattern Recognit, vol. 121, p. 108205, Jan. 2022, doi: 10.1016/J.PATCOG.2021.108205.

[24] X. Tang, Z. Zhang, and Y. Qin, "On-Road Object Detection and Tracking Based on Radar and Vision Fusion: A Review," IEEE Intelligent Transportation Systems Magazine, vol. 14, no. 5, pp. 103–128, 2022, doi: 10.1109/MITS.2021.3093379.

[25] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, [Online]. Available: http://arxiv.org/abs/1804.02767

[26] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[27] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[28] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022, doi: 10.48550/arxiv.2207.02696.

[29] S. Zhang, Z. Yu, L. Liu, X. Wang, A. Zhou, and K. Chen, "Group R-CNN for Weakly Semi-Supervised Object Detection With Points." pp. 9417–9426, 2022. Accessed: Jan. 15, 2023. [Online]. Available: https://github.com/jshilong/GroupRCNN.

[30] S. Rani, B. K. Singh, D. Koundal, and V. A. Athavale, "Localization of stroke lesion in MRI images using object detection techniques: A comprehensive review," Neuroscience Informatics, vol. 2, no. 3, p. 100070, Sep. 2022, doi: 10.1016/J.NEURI.2022.100070.

[31] S. Kim, H. Wimmer, and J. Kim, "Analysis of Deep Learning Libraries: Keras, PyTorch, and MXnet," 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications, SERA 2022, pp. 54–62, 2022, doi: 10.1109/SERA54885.2022.9806734.

[32] "Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and ... - Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili, Dmytro Dzhulgakov - Google Books." https://books.google.co.id/books?hl=en&lr=&id=SVxaEAAAQBAJ&oi=fnd&pg=PP1&dq=pytorch+review&ots=1cAKBPwDld&sig=jmIMZ1wLIX4RQ0B60UHtGs1anmA&redir_esc=y#v=onepage&q=pytorch%20review&f=false (accessed Jan. 15, 2023).

[33] Z. Chen, S. Yang, C. Liu, Y. Hu, K. Li, and K. Li, "EPMC: efficient parallel memory compression in deep neural network training," Neural Comput Appl, vol. 34, no. 1, pp. 757–769, Jan. 2022, doi: 10.1007/S00521-021-06433-5/METRICS.

[34] H. Ding, M. A. Latif, Z. Zia, M. A. Habib, M. A. Qayum, and Q. Jiang, "Facial Mask Detection Using Image Processing with Deep Learning," Math Probl Eng, vol. 2022, 2022, doi: 10.1155/2022/8220677.

[35] A. Jamwal, R. Agrawal, and M. Sharma, "Deep learning for manufacturing sustainability: Models, applications in Industry 4.0 and implications," International Journal of Information Management Data Insights, vol. 2, no. 2, p. 100107, Nov. 2022, doi: 10.1016/J.JJIMEI.2022.100107.

[36] H. Wei, Z. Zhao, and R. Luo, "Advancing MM/PBSA calculations with machine learning and cuda GPUs," Biophys J, vol. 121, no. 3, pp. 527a–528a, Feb. 2022, doi: 10.1016/j.bpj.2021.11.2779.

[37] S. Shao et al., "CrowdHuman: A Benchmark for Detecting Human in a Crowd," Apr. 2018, Accessed: Jan. 15, 2023. [Online]. Available: http://arxiv.org/abs/1805.00123

[38] "Roboflow: Give your software the power to see objects in images and video." https://roboflow.com/ (accessed Jan. 15, 2023).

[39] "YouTube." https://www.youtube.com/ (accessed Jan. 15, 2023).

[40] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo Algorithm Developments," Procedia Comput Sci, vol. 199, pp. 1066–1073, Jan. 2022, doi: 10.1016/J.PROCS.2022.01.135.

[41] G. Han, S. Huang, J. Ma, Y. He, and S.-F. Chang, "Meta Faster R-CNN: Towards Accurate Few-Shot Object Detection with Attentive Feature Alignment," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 1, pp. 780–789, Jun. 2022, doi: 10.1609/AAAI.V36I1.19959.

[42] A. Makalesi, A. Mawlood, A. Ghani Abdulghani, G. Gokce, and M. Dalveren, "Moving Object Detection in Video with Algorithms YOLO and Faster R-CNN in Different Conditions," European Journal of Science and Technology, no. 33, pp. 40–54, Jan. 2022, doi: 10.31590/EJOSAT.1013049.

[43] U. Mittal, P. Chawla, and R. Tiwari, "EnsembleNet: a hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models," Neural Comput Appl, pp. 1–20, Oct. 2022, doi: 10.1007/S00521-022-07940-9/METRICS.

[44] T. Diwan, G. Anirudh, and J. v. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," Multimed Tools Appl, pp. 1–33, Aug. 2022, doi: 10.1007/S11042-022-13644-Y/TABLES/7.

[45] "GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite." https://github.com/ultralytics/yolov5 (accessed Jan. 15, 2023).

[46] "Overview of model structure about YOLOv5 · Issue #280 · ultralytics/yolov5 · GitHub." https://github.com/ultralytics/yolov5/issues/280 (accessed Jan. 15, 2023).

[47] "PyTorch." https://pytorch.org/ (accessed Jan. 15, 2023).

[48] A. Paszke et al., "Automatic differentiation in PyTorch." Oct. 28, 2017.

[49] "Pytorch Background and History".