

Classifying Motorcycle Rider Helmet on a Low Light Video Scene using Deep Learning

John Paul Q. Tomas, Bonifacio T. Doma

School of Information Technology, Mapua University, Makati, Philippines

Abstract—For safety in transportation, it is important to always monitor the use of proper motorcycle helmet, especially at night. One way to enforce transportation rules and regulations in wearing proper motorcycle helmet is to use computer vision technology. This study focusses on classifying motorcycle rider helmet at low light video conditions, like at dusk and at night, using YOLOv5 and YOLOv7 with Deep SORT. In these deep learning methods, the study tunes and optimizes hyperparameters to attain high accuracy in classifying motorcycle rider helmet at this challenging environment. To accomplish this objective, a vast and diverse dataset was employed, containing classes such as riders, different types of helmets (valid and invalid), and instances of riders not wearing helmets at all in Metro Manila, Philippines. The results show that Hyperparameter 3 consistently outperformed other settings in terms of precision (95.6%), recall (91.2%), and mean average precision (mAP) scores across multiple scales and time frames with 95.1% on mAP@0.5 and 76.3% on mAP@0.95, owing to greater epochs, quicker learning rates, and lower batch sizes.

Keywords—Artificial intelligence; computer vision; computer vision problems; object detection; YOLOv5; YOLOv7; Deep SORT; deep learning

I. INTRODUCTION

Wearing motorcycle helmets is essential for the safety of riders especially in the Philippines, where over 1,000 motorcycle-related road crashes occurred in 2019 [1-2]. Research from the World Health Organization reveals helmets can reduce motorcycle crash death risk by up to 40% [3]. This emphasizes the necessity of helmets to prevent injury or death for riders [4-8]. Helmet use is mandated by many countries' laws. Yet, detecting helmet use in low-light conditions, at dusk or at night, remains a challenge. This study evaluates You Only Look Once (YOLO) version five models combined with Deep Simple Online Realtime Tracking (Deep SORT) for detecting helmet use in nighttime traffic situations in Metro Manila, Philippines. Meanwhile, Metro Manila's traffic persist due to car-centric policies and resistance to eco-friendly transportation [9]. Efforts like the Public Utility Vehicle Modernization Program face challenges due to historical norms and current crises. Addressing low light conditions in object detection is another hurdle, tackled by a study that integrates multi-scale detection, attention mechanisms, and Convolutional Block Attention Module (CBAM) for improved accuracy in identifying objects under limited light [10]. The YOLOv5 Small – Feature Combinatorial Grouping (FCG) model showcased up to 87.5% mean accuracy precision (mAP) in detecting objects like helmets in challenging low light, highlighting the potential of advanced techniques.

Combining YOLOv7 and Deep SORT offers cutting-edge object recognition and tracking, applied in real-time video analysis [11]. They contribute to safer roads by identifying helmetless riders. These open-source technologies continually evolve, serving the research community and the public. The YOLOv5 algorithm achieves high detection accuracy while maintaining real-time performance, making it suitable for a variety of computer vision applications. On the other hand, YOLOv7 is a one-stage object detection system that divides an image into grid cells and predicts bounding boxes and class labels for each. Both algorithms utilize a convolutional neural network (CNN), mainly employed for calculating bounding boxes, and the use of a SoftMax layer for class label prediction [12]. The architecture of the network consists of three main parts: the backbone network, the neck network, and the head network. The backbone network oversees extracting picture features. The neck network oversees fusing the backbone network's extracted features. The head network oversees predicting the bounding boxes and class labels for the objects in the dataset.

Moreover, the main contribution of this study is the tuning and optimization of hyperparameters within the framework of YOLOv5 and YOLOv7 with Deep SORT to classify accurately motorcycle helmets at low light video scenarios with respect to the authors' previously conducted study [13]. To the best of our knowledge, we believe that this is the first study done on this type of video scenes for this application. The enhancements made will definitely be beneficial to the implementation of transportation rules and regulation on wearing of proper motorcycle helmets at all times using computer vision technology.

II. RELATED WORKS

A. YOLOv5

YOLOv5 is an object detection technique that improves on the success of prior YOLO models by introducing a real-time object detection methodology. It performs object detection tasks by dividing an input image into grid cells and predicting bounding boxes and class probabilities for items within each grid cell using a deep CNN architecture [14]. Its methodology includes a novel approach known as a "cross-stage partial network" that improves the network's feature representation capabilities and enables more accurate object detection [12]. The algorithm achieves high detection accuracy while maintaining real-time performance, making it suitable for a variety of computer vision applications. In a study conducted by Jia et al. (2021) [15], they proposed an end-to-end motorcycle helmet detection using YOLOv5 wherein

motorcycle riders were detected including the riders wearing helmets. The model was able to achieve an exceptional accuracy having an mAP up to 97% for each classes. However, it is important to note that these results were only evaluated in a high-light condition since the dataset used only contains noon time footages of highways.

B. YOLOv7

YOLOv7 is an improvement of YOLOv5. It is also a one-stage object detection system that divides an image into grid cells and predicts bounding boxes and class labels for each. It also uses a CNN which primarily used to forecast the bounding boxes, and a SoftMax layer is used to predict the class labels [10]. Its network is divided into three sections: the backbone network, the neck network, and the head network. The backbone network oversees extracting picture features. The neck network oversees fusing the backbone network's extracted features. The head network oversees predicting the bounding boxes and class labels for the objects in the dataset. As for the backbone network in YOLOv7, it uses the CSPDarknet53 network [16–19]. CSPDarknet53 is a revision of the Darknet53 network that has been shown to improve performance and accuracy. In YOLOv7, the neck network is called the YOLOv7-Neck network, which is used in fusing the features extracted by the backbone network. The head network, on the other hand, is in charge of predicting the bounding boxes and class labels for the image's objects. It is trained using a dataset of tagged images with bounding boxes and class labels for the objects in the images. The training procedure is divided into two stages: pre-training and fine-tuning. Moreover, the network is trained on a huge dataset of photos tagged with bounding boxes and class labels for the objects in the images during the pre-training phase. The pre-training phase is used to understand the fundamental properties of objects.

According to Nandhakumar [19], YOLOv7 outperforms earlier object detection algorithms in terms of both speed and accuracy. It can reach real-time speeds of up to 160 frames per second while maintaining great precision. As a result, it is a helpful tool for a wide range of applications, including autonomous driving, video surveillance, and robots. Currently, there have been no studies yet that utilized YOLOv7 in detecting motorcycle helmet specifically in low-light conditions. It has been used before in detecting Camellia Oleifera Fruit in orchard scenes by Wu et al [20], as well as Chicory Plant by Gallo et al. [21]

C. Deep SORT

Deep SORT on the other hand is a method for tracking multiple objects that combines a deep learning-based object detector and the SORT (Simple Online and Realtime Tracking) algorithm. This method improves on traditional tracking methods by employing a deep neural network to generate high-quality embeddings that encode the appearance of seen objects [19-20]. These embeddings are then used in conjunction with a Kalman filter-based tracking framework to associate and track objects over successive frames. Deep SORT provides powerful and dependable tracking by combining appearance, motion dynamics, and temporal information. social distancing measures during the COVID 19 pandemic by Narinder Singh

Punn et al. (2020) [22], as well as Pear fruit detection by Addie Ira Borja Parico and Tofael Ahmed (2021) [23].

D. Scale

YOLO has multiple scales, from nano, small, medium, large, xlarge. The nano-scale model is the algorithm's smallest and fastest variant, but least in precision. It contains fewer layers and characteristics than the medium and large scales, making it better suited for deployment on resource-constrained devices. The YOLOv5 small-scale model performs effectively in object detection tasks despite its smaller size. It detects objects in an image using a single convolutional neural network (CNN) architecture, with a focus on small objects, and incorporates data from several sizes of the input image using a feature pyramid network (FPN) [11].

A single convolutional neural network (CNN) architecture is used in the medium-scale model to detect objects. It also anticipates object-bounding boxes through the use of anchor boxes [26]. It does, however, use more anchor boxes than the small-scale model, allowing it to distinguish objects with more precision. It can also detect objects with high precision while maintaining real-time performance as its major characteristic, making it suited for a wide range of real-world applications. Finally, the large-scale model operates similarly to the previous model scales, but because it employs larger layers and a greater number of anchor boxes than the small and medium-scale models, it allows for the identification of objects with even greater precision [14].

E. Inference

To use model testing and inference in YOLOv5, which is the process of using the trained model to make predictions based on new unseen data. The model would be prepared and configured the Python environment with the required dependencies. Then all the test images or videos should be collected to generate the YOLOv5 configuration file, this includes the model's weights and parameters. Then infer by running the test data through the model and generate bounding box predictions for object recognition. The predictions can still be refined by removing redundant detections with non-maximum suppression and scaling the coordinates to fit the original image size. Finally, plot the bounding boxes on the images or videos to visualize the results [1][16].

F. Hyperparameters

In YOLO, adjusting hyperparameters entails fine-tuning several settings to optimize the model's performance. Considering the speed and accuracy requirements, the architecture of the YOLOv5 model can also be adjusted by selecting different scales such as YOLOv5s, YOLOv5m, or YOLOv5l [27]. Experimentation with hyperparameters such as learning rate, weight decay, and batch size, can all have a major impact on training. To boost generalization, regularization techniques such as dropout and data augmentation can be used. It is critical to evaluate the model's performance using evaluation metrics such as mean average precision (mAP) and to iteratively alter the hyperparameters based on the results. By carefully tweaking these hyperparameters, the model's detection accuracy and overall performance can be enhanced [18]. Full model framework can

reference based on the image below Fig. 1, the application of Kalman filter in which Deep SORT is integrated for optimization [24] [28].

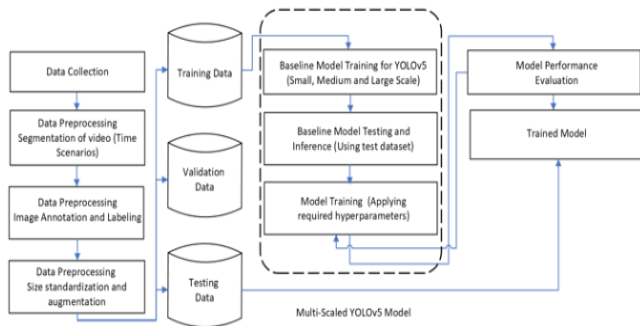


Fig. 1. YOLOv5 multi-scaled conceptual framework.

Deep SORT is a deep learning-based multi-object tracking algorithm that is capable of tracking objects over time even when they are occluded or partially visible. It uses a combination of appearance information and motion cues to track objects in real-time. By combining the YOLOv7 with Deep sort, the system works by first using YOLOv7 to detect objects in each frame of a video stream. The detected objects are then passed to the Deep Sort algorithm, which associates objects across frames and tracks them over time. By combining both object detectors, the system can track multiple objects in real time, even in complex and cluttered scenes.

It is critical to apply Kalman filter in improving Deep Sort algorithm's tracking performance as it adds temporal information and refine tracking predictions. The Kalman filter is initially initialized with the object's position and velocity represented by the state vector and covariance matrix, then forecasts the next state based on the object's motion model throughout each time step. When new detection or tracking data becomes available, the Kalman filter updates the measurement, including the measurements into the estimating process and changing the state estimate. This enables the Kalman filter to smooth out noisy detections, handle occlusions, and provide more accurate and consistent object tracking predictions in Deep SORT [24]. The full model framework can be referred to using the graphic below Fig. 2.

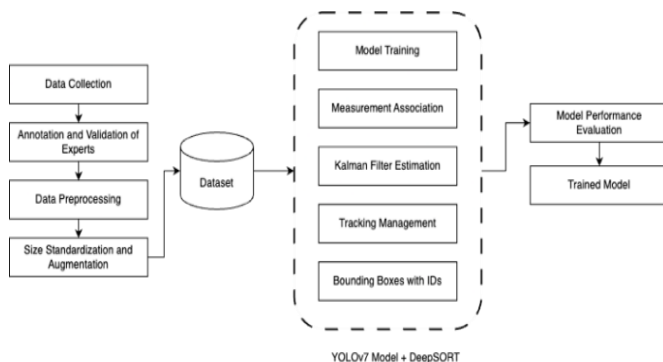


Fig. 2. YOLOv7 + Deep SORT conceptual framework.

G. Synthesis

Reviewed studies indicate that the performance of YOLOv5 is limited in low-light conditions while YOLOv7

remains unexplored along with the implementation of Deep SORT in terms of detecting motorcycle helmets in low-light scenarios. The researchers then aims to address this by exploring the performance of the two models with the help of hyperparameter tuning to achieve optimal results.

III. METHODOLOGY

The study compared two YOLO versions: YOLOv5 and YOLOv7. Deep sort was integrated as an optimization in YOLOv7 [24], [25].

A. Scaling

To ensure uniformity in image dimensions and maintain consistency during both the training and inference stages of the model training, each image were resized to a resolution of 640x640 pixels using Roboflow.

B. Inference

The dataset was divided into three parts: 70% for training set, 20% for validation set, and 10% for testing set with the use of Roboflow. After the model has been trained using the training set. The validation set was used to evaluate the performance of the model based on the initial training for the optimization of hyperparameters then the testing set was used to evaluate the unbiased performance of the model on new unseen data.

C. Tuning and Optimization of Hyperparameters

Three hyperparameter configurations were used: (1) Hyperparameter 1 with 0.01 LR, 64 Batch, and 50 Epochs, (2) Hyperparameter 2 with 0.02 LR, 32 Batch, and 75 Epochs, (3) Hyperparameter 3 with 0.03 LR, 16 Batch, and 100 epochs.

The study also included another version of the YOLO, which is version 7. Most of the initial processes of this model are the same as YOLOv5, but it differs in processes such as measurement association and Kalman Filter Estimation.

D. Data Gathering



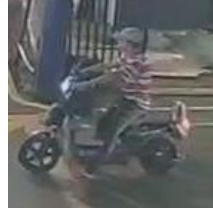
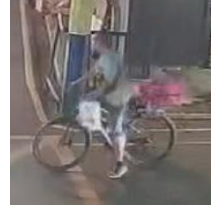
The dataset was gathered from a bustling street in Makati City, Philippines, renowned for its substantial motorcycle traffic. The data collection process was strategically conducted during three distinct timeframes to encompass diverse lighting conditions: 5-6 PM, 6-7 PM, and 7-8 PM.

E. Data Pre-Processing

In this step, Smart Video Player was used to monitor the video footage and to identify which timeframes would be relevant to be used in the model. This was followed by annotation. The collected footages were monitored to make sure that there were enough types of riders for each scenario: riders with helmets (half-faced or full faced), no helmets, and invalid situations.

The images were then annotated and separated using the training-validation-test splitting method, wherein 70% was used for training, 20% was used for validating, and 10% was used for testing. The images were annotated with four (4) label classes namely Motorcycle Rider, Helmet Full-Faced, Helmet Half-Faced, No Helmet, and Invalid Helmet as shown in Table I.

TABLE I. IMAGE LABEL CLASSES

Class Label	Image Sample
Rider Full Face	
Rider Half-Face	
Rider Helmet Invalid	
Rider No Helmet	

Three strategies were used for pre-processing, including auto-orientation, resizing, and class modification. Images were auto-orientated to align to a standard orientation before feeding them into the model. This technique ensures that the images are in the correct orientation for processing, which can increase the model's accuracy. Because the orientation of the motorcycle rider and helmet might change substantially in different photos, auto-orientation is especially important in the context of helmet recognition for motorcycle riders.

The images were then resized and expanded to the typical size of 640x640. This strategy ensures that all of the photos are of the same size, which is required for the model to efficiently assess the images. Resizing the photos can also help to minimize the model's computational complexity, making it faster and more efficient. In the area of motorcycle helmet recognition, scaling the photos to a consistent size can be very significant because the size of the motorcycle rider and helmet can vary greatly between images.

Finally, the images were class modified by mapping and dropping specific classes from the dataset. In this scenario, 16 classes were mapped while 0 were dropped. This strategy ensures that the model is trained on the relevant classes and can increase the model's accuracy. Dropping classes entails

deleting extraneous classes from the dataset whereas mapping classes requires merging comparable classes into a single class. In the domain of motorcycle helmet detection, mapping and removing classes can help to ensure that the model is trained on the necessary classes and can detect helmets effectively.

F. Data Augmentation

Various image enhancement techniques were employed to augment the dataset. These included flipping, rotation, cropping, grayscale conversion, and color distortion. Flipping involved horizontal or vertical image flipping, while rotation altered images in clockwise, counterclockwise, or upside-down orientations. Random cropping was applied to zoomed image sections. Grayscale conversion turned images to grayscale, and color distortion adjusted hue, saturation, brightness, and exposure [22]. Applying these enhancements resulted in three training examples per original image, expanding and diversifying the training dataset. These techniques increased the model's resilience to input photo variations.

Bounding box augmentation was also performed, applying the same techniques to helmet-bounding boxes for accurate post-transformation alignment [22]. Augmentation, in the context of motorcycle helmet identification, improved model resilience and accuracy. The augmented, pre-processed dataset was split for model training, validation, and testing. A YOLOv5 model was developed for different scales, with modifications based on three time frames.

The time frames (5-6 PM, 6-7 PM, 7-8 PM) accounted for varying luminance, affecting helmet visibility. Images were tagged into four classes as depicted in Table I, "Rider Full Face" depicted helmets covering the entire head. "Rider Half Face" showed helmets protecting the top and back, excluding the chin. "Other Helmets" encompassed non-motorcycle headgear. "Rider No Helmet" included images of unprotected riders. This approach facilitated effective helmet detection and classification.

G. Model Training

The dataset of photos or videos of motorcycles and motorcycle riders wearing helmets was prepared by ensuring the following: bounding boxes were drawn around the objects of interest (motorcycles and helmets), together with class labels indicating whether the object is a motorcycle or a helmet. It is critical to include a diverse range of motorcycle and helmet types in the dataset to guarantee that the model can detect these objects under a variety of situations.

After the dataset has been prepared, the YOLOv5 model architecture must be configured. This entails deciding on a model size (such as YOLOv5s, YOLOv5m, or YOLOv5l) and determining the number of classes to detect. In this particular scenario, the motorcycle and the rider's helmet.

After configuring the model architecture, the next step is to start the training process. During training, the model iteratively updates its weights based on the loss calculated between the predicted and ground truth bounding boxes and class probabilities. The objective is to minimize this loss function by adjusting selected hyperparameters until the model produces accurate and consistent predictions.

For each of the architecture, time frames were considered to segment changes in lighting conditions. Then, the three hyperparameters were used within the experiment. Aside from using the default hyperparameter setting, additional settings were included by changing the learning rate, batch size, and epoch during training. Learning rate is a hyperparameter that determines the magnitude of the update to the model's parameters during the training process. It controls how quickly the model converges to the optimal solution and can have a significant impact on the model's performance. Batch size refers to the number of training examples used in one iteration of the optimization algorithm during the training process. The batch size is a hyperparameter that can have a significant impact on the model's performance and training time. Lastly, in machine learning, an epoch refers to a complete iteration through the entire training dataset during the training process. The number of epochs is a hyperparameter that determines how many times the algorithm will cycle through the entire dataset. Each epoch is broken down into batches, and the model's parameters are updated based on the loss function calculated on each batch.

To further discuss the optimization process, different hyperparameter settings were applied in the different YOLOv5 architectures (YOLOv5s, YOLOv5m and YOLOv5l) and YOLOv7 as shown in Table II, specifically, the first hyperparameter setting has a configured learning rate of 0.01, batch size of 64, and 50 epochs. The second had a higher learning rate of 0.02, but with a smaller batch size of 32, and more epochs of 75. The third setting has the highest learning rate of 0.03, smaller batch size of 16, and more epochs of 100. Though these settings, and applied in the different time frames, optimized HP can be determined for each of the architectures.

TABLE II. TRAINING HYPERPARAMETERS

Hyperparameter Name	Learning Rate	Batch Size	# of Epochs
H1	0.01	64	50
H2	0.02	32	75
H3	0.03	16	100

Moreover, to improve the robustness of the model, data augmentation techniques such as random cropping, flipping, and resizing were used to create more variations in the training data. Once the training process was complete, the model was evaluated on a validation set to measure its performance. Metrics such as mAP and intersection over union (IoU) were used to evaluate the accuracy and robustness of the model.

H. Model Validation

To validate the model, the "detect.py" script, was used. This script effectively conducted bounding box predictions and class estimations on the designated test images. The core functionality of YOLOv5 was harnessed to accomplish this. Subsequent to the bounding box predictions and class estimations, an essential step known as inference computation was performed. This involved the integration of the predicted bounding box information with corresponding prediction rates.

For a comprehensive model validation process, the "val.py" script was applied. This script, crucially, made use of the

optimal model weights contained within the "best.pt" file, ensuring the utilization of the most refined model configuration. This choice of weights maximized model performance. Within the validation process, the script meticulously computed a suite of performance metrics. These metrics encompassed crucial elements such as classes identified, total images assessed, individual instances detected, as well as precision, recall, and mean average precision values. These calculated metrics collectively formed a robust quantitative representation of the model's efficacy and accuracy in object detection and classification [8].

IV. RESULTS

A. Effect of Timeframes

Three different timeframes were used. Each timeframe can be described as follows. During the 5-6 PM timeframe, as depicted in Fig. 3, the environment still retained traces of daylight, albeit with a diminishing intensity. The sun's descent casted elongated shadows, and the street came alive with a mixture of natural and emerging artificial light sources, such as streetlights and the headlights of vehicles.

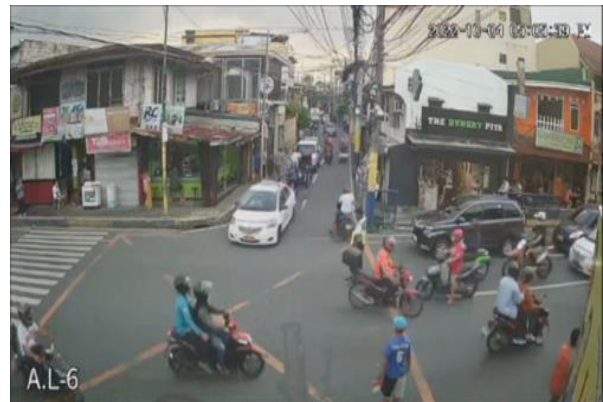


Fig. 3. 5-6 pm footage.

As the clock advanced to 6-7 PM, based on Fig. 4, the setting underwent a noticeable change. The sky took deeper hues of twilight, and the natural light waned further. Streetlights begin to dominate the scene, creating a stark interplay between light and shadow. Details became less discernible, and the environment embraced an ambiance of early evening.



Fig. 4. 6-7 pm footage.



Fig. 5. 7-8 pm footage.

By the time 7-8 PM arrived, as depicted in Fig. 5, darkness had firmly settled in. The streetlights and vehicle headlights became the primary sources of illumination, casting a subdued glow across the surroundings. The scene exuded an atmosphere of low light, with visibility significantly limited compared to earlier hours.

These deliberate timeframes were chosen to comprehensively capture the spectrum of lighting conditions that motorcycle rider's encounter. The resultant dataset's pre-processed videos effectively portrayed the evolving illumination scenarios, enriching the machine-learning model's ability to navigate and respond adeptly across varying levels of lighting intricacies.

B. YOLOv5

The model's performance was evaluated using different hyperparameters in various scenarios and summarized in Table III. In the YOLOv5 Small scale results from 5 to 6 PM, Hyperparameter 3 yielded the best performance, achieving an average precision of 94.4%, an average recall of 89.8%, an

average mAP@.5 of 94.3%, and an average mAP@.95 of 73.8%. Moving to the YOLOv5 results from 6 to 7 PM, Hyperparameter 3 also had the best performance, achieving an average precision of 75.6%, an average recall of 65.6%, an average mAP@.5 of 68.3%, and an average mAP@.95 of 40%, especially in the Rider class. In the YOLOv5 Small scale results from 7 to 8 PM, Hyperparameter 2 performed the most achieving an average precision of 62%, an average recall of 59.3%, an average mAP@.5 of 61.4%, and an average mAP@.95 of 49.2%.

Various hyperparameters yielded different results across three varying time frames. While different hyperparameters performed well on certain scenarios. The Hyperparameter 3 performed the most while striking a balance between precision and recall.

C. YOLOv7 with Deep SORT

The results presented in Table III reflects the performance of the YOLOv7 model under different hyperparameters and timeframes. Notably, precision, recall, and mAP (mean Average Precision) scores were evaluated to gauge the model's ability to accurately detect and classify objects within these specified contexts.

During the 5-6 PM timeframe, Hyperparameter 3 exhibited the highest precision (94.4%), recall (89.8%), mAP@.5 (94.3%), and mAP@.95 (73.8%). Shifting to the 6-7 PM timeframe, Hyperparameter 1 stood out with the highest precision (75.6%) and a relatively good recall (65%), leading to a commendable balance between the two metrics. Lastly, in the 7-8 PM timeframe, Hyperparameter 2 demonstrated the highest precision (72.3%), recall (68.2%), mAP (70.6%), and mAP@.95 (63%). These results show that the implementation of Deep SORT to the training of YOLOv7 models had outperformed the results of its predecessor YOLOv5 given the hyperparameter configurations mentioned in Table II

TABLE III. YOLOV5 AND YOLOV7 RESULTS

Model	Parameters			
	Precision (%)	Recall (%)	mAP@.5 (%)	mAP@.95 (%)
Timeframe 5-6 PM				
YOLOv5 (L) H3	94.4	89.8	94.3	73.8
YOLOv7 H3	95.6	91.2	95.1	76.3
Timeframe 6-7 PM				
YOLOv5 (L) H3	75.6	65.6	68.3	40
YOLOv7 H1	86	73	79.5	51
Timeframe 7-8 PM				
YOLOv5 (M) H2	62	59.3	61.4	49.2
YOLOv7 H3	72.3	68.2	70.6	63

V. DISCUSSION

Our study compared three hyperparameter settings in YOLOv5 and YOLOv7 with Deep SORT object identification models which demonstrated the importance of hyperparameter adjustments in achieving superior performance metrics [18]. Particularly for certain classes such as riders and invalid cases, H3 regularly displayed higher precision, recall, and mAP

scores. To address issues such as accurate helmet detection, however, fine-tuning and rigorous evaluation are still required. These findings contribute to the continuous developments in object detection models and provide useful insights for computer vision researchers and practitioners.

Based on the outcomes of the different models, it is recommended that researchers and practitioners working with

object identification models, particularly YOLOv5 and YOLOv7 with Deep SORT, take into account the ideal hyperparameter settings revealed in study [6] [10] [13] [12]. Specifically, Hyperparameter 3 regularly outperformed other scales and time frames, with higher precision, recall, and mean average precision (mAP) ratings.

Implementing Hyperparameter 3 (higher epochs, quicker learning rates, and smaller batch sizes) can result in enhanced object detection accuracy and better localization of certain classes such as riders and invalid cases. It is crucial to highlight, however, that further study and fine-tuning may be required to overcome difficulties with helmet identification, as this area demonstrated space for development.

Finally, this work emphasizes the need of doing comparative analyses and experimenting with different hyperparameter settings to maximize model performance. As the science of computer vision advances, it becomes increasingly important to adjust and refine hyperparameters to achieve the best results for individual applications.

Overall, this study gives useful insights on hyperparameter tuning in object detection models, as well as practical recommendations for scholars and practitioners in the field. Practitioners can improve the accuracy and performance of their object detection systems by evaluating the findings and using the recommended hyperparameter values, thereby contributing to breakthroughs in computer vision and its diverse applications.

Further research could explore innovative techniques and data augmentation methods tailored to low-light scenarios. Leveraging advancements in low-light image enhancement and night vision technologies could prove beneficial. Additionally, the incorporation of infrared or thermal imaging sensors in object detection models may enhance the accuracy of helmet detection under challenging lighting conditions. Moreover, collaborating with experts in the field of motorcycle safety to collect real-world data from diverse lighting environments and helmet types would be invaluable for training and evaluating detection models. Lastly, an emphasis on fine-tuning hyperparameters, such as those highlighted in this study, should continue to be a crucial aspect of future research efforts to achieve more robust and reliable helmet detection, ultimately contributing to increased safety for riders.

VI. CONCLUSION

In conclusion, our study comparing YOLOv5 and YOLOv7 with Deep SORT object identification models revealed that YOLOv7 with Hyperparameter 3 consistently outperformed YOLOv5 and other hyperparameter settings in terms of precision, recall, and mean average precision (mAP) scores. Specifically, Hyperparameter 3, characterized by higher epochs, quicker learning rates, and smaller batch sizes, proved to be the superior choice for achieving enhanced object detection accuracy, especially for certain classes like riders and invalid cases. These findings suggest that YOLOv7 with Hyperparameter 3 is the recommended configuration for practitioners and researchers working in the field of object detection, highlighting its potential to contribute significantly to advancements in computer vision applications.

REFERENCES

- [1] F. Zhou, H. Zhao, and Z. Nie, "Safety Helmet Detection Based on YOLOv5," in 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2020, pp. 1-5, doi: 10.1109/ICAICA51687.2020.9362711.
- [2] J. L. Lu, T. J. Herbosa, and S. F. Lu, "Analysis of Transport and Vehicular Crash Cases Using the Online National Electronic Injury Surveillance System (ONEISS) from 2010 to 2019," *Acta Medica Philippina*, vol. 56, 2022, doi: 10.47895/amp.v56i1.3874.
- [3] J. E. Espinosa, S. A. Velastin, and J. W. Branch, "Detection of Motorcycles in Urban Traffic Using Video Analysis: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 6115-6130, 2021, doi: 10.1109/TITS.2020.2997084.
- [4] J. Macalisang, D. P. Ordovez, M. K. C. Ledda, M. P. Melegrito, and A. M. C. Obon, "A Machine Vision-Based Deep Learning Inference Approach of Biker Safety Hat Detection System," in 2021 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2021, pp. 1-6, doi: 10.1109/HNICEM52687.2021.9484194.
- [5] M. Swapna, T. Wajeed, and S. Jabeen, "A Hybrid Approach for Helmet Detection for Riders Safety Using Image Processing, Machine Learning, Artificial Intelligence," *International Journal of Computer Applications*, vol. 182, pp. 50-55, 2019, doi: 10.5120/ijca2019918397.
- [6] M. Dasgupta, O. Bandyopadhyay, and S. Chatterji, "Automated Helmet Detection for Multiple Motorcycle Riders Using CNN," in 2019 IEEE Conference on Information and Communication Technology (CICT), 2019, pp. 1-6, doi: 10.1109/CICT48419.2019.9066191.
- [7] Y. Zhou, L. Jiang, Y. Liang, C. Ma, H. Sun, S. Nie, and Y. Zuo, "Helmet Detection Algorithm Based on Single Pixel Zoom," *Journal of Physics: Conference Series*, vol. 1682, 2020, doi: 10.1088/1742-6596/1682/1/012021.
- [8] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.
- [9] J. T. Sidel, "Averting 'Carmageddon' through Reform? An Eco-Systemic Analysis of Traffic Congestion and Transportation Policy Gridlock in Metro Manila," *Critical Asian Studies*, pp. 1-25, 2020, doi: 10.1080/14672715.2020.1793681.
- [10] P. Wang, H. Huang, M. Wang, and B. Li, "YOLOv5s-FCG : An Improved YOLOv5 Method for Inspecting Riders' Helmet Wearing," *Journal of Physics: Conference Series*, vol. 2024, 2021, doi: 10.1088/1742-6596/2024/1/012059.
- [11] Q. Zhou, F. Sun, and J. Zhang, "Research on Multi-Target Detection and Tracking Algorithm Based on Improved YOLOv5," *IOS Press eBooks*, 2022, doi: 10.3233/ATDE221115.
- [12] Y. Qian et al., "Real-Time Detection of *Eichhornia Crassipes* Based on Efficient YOLOV5," *Machines*, vol. 10, 2022, doi: 10.3390/machines10090754.
- [13] J. Paul and B. Doma, "Motorcycle Helmet Detection and Usage Classification in the Philippines Using YOLOv5 Algorithm," in *Proceedings of the 2022 International Conference on Computer Science and Artificial Intelligence*, 2022, pp. 1-5, doi: 10.1145/3581792.3581796.
- [14] Kisaiezehra et al., "Real-Time Safety Helmet Detection Using Yolov5 at Construction Sites," *Intelligent Automation & Soft Computing*, vol. 36, pp. 911-927, 2023, doi: 10.32604/iasc.2023.031359.
- [15] W. Jia et al., "Real-Time Automatic Helmet Detection of Motorcyclists in Urban Traffic Using Improved YOLOv5 Detector," *IET Image Processing*, 2021, doi: 10.1049/ipr.2.12295.
- [16] X. He et al., "Detection of the Floating Objects on the Water Surface Based on Improved YOLOv5," in 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), 2021, pp. 1-5, doi: 10.1109/ICIBA52610.2021.9688111.
- [17] T. Hong et al., "A Real-Time Tracking Algorithm for Multi-Target UAV Based on Deep Learning," *Remote Sensing*, vol. 15, pp. 2-2, 2022, doi: 10.3390/rs15010002.

- [18] 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" arXiv (Cornell University), 2022, doi: 10.48550/arxiv.2207.02696.
- [19] R.G. Nandhakumar, and S. Mohanapriya, "Smart Baby Monitoring System Using YOLOv7 Algorithm," 2022, doi:https://doi.org/10.1109/icitri56423.2022.9970217.
- [20] D. Wu et al., "Detection of Camellia Oleifera Fruit in Complex Scenes by Using YOLOv7 and Data Augmentation. Applied Sciences, 2022, doi:https://doi.org/10.3390/app122211318.
- [21] I. Gallo et al., "Deep Object Detection of Crop Weeds: Performance of YOLOv7 on a Real Case Dataset from UAV Images," Remote Sensing, vol. 15, p. 539, 2023, doi: 10.3390/rs15020539.
- [22] N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Monitoring COVID-19 Social Distancing with Person Detection and Tracking via Fine-Tuned YOLO v3 and Deepsort Techniques," arXiv:2005.01385 [cs], 2020, doi: 10.48550/arXiv.2005.01385.
- [23] A. I. B. Parico and T. Ahamed, "Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT," Sensors, vol. 21, p. 4803, 2021, doi: 10.3390/s21144803.
- [24] F. Yang, X. Zhang, and B. Liu, "Video Object Tracking Based on YOLOv7 and DeepSORT," arXiv:2207.12202 [cs], 2023, doi: 10.1109/icce-asia57006.2022.9954809.
- [25] D. N.-N. Tran, L. H. Pham, H.-H. Nguyen, and J. W. Jeon, "City-Scale Multi-Camera Vehicle Tracking of Vehicles Based on YOLOv7," in 2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 2022, pp. 1-5, doi: 10.1109/ICCE-Asia57006.2022.9954809.
- [26] F. H. Kamaru Zaman et al., "Visual-Based Motorcycle Detection Using You Only Look Once (YOLO) Deep Network," IOP Conference Series: Materials Science and Engineering, vol. 1051, 2021, doi: 10.1088/1757-899X/1051/1/012004.
- [27] J. Wu et al., "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," Journal of Electronic Science and Technology, vol. 17, pp. 26-40, 2019, doi: 10.11989/JEST.1674-862X.80904120.
- [28] S. E. Li et al., "Kalman Filter-Based Tracking of Moving Objects Using Linear Ultrasonic Sensor Array for Road Vehicles," Mechanical Systems and Signal Processing, vol. 98, pp. 173-189, 2018, doi: 10.1016/j.ymsp.2017.04.04.