Article | Received 11 October 2024; Accepted 31 December 2024; Published 20 January 2025
https://doi.org/10.55092/rl20250001

Quadrotor UAV-based smoke detector system using YOLOv8 towards wildfire prevention

Jesus Gonzalez-Alayon and Herman Castañeda*

Tecnologico de Monterrey, School of Sciences and Engineering, Monterrey, Mexico

* Correspondence author; E-mail: hermancc@tec.mx.

Highlights:

- An aerial smoke detector based on a quadrotor, ROS and YOLO V8 nano model.
- Towards wildfire prevention by fast smoke detection for places with hard access.
- Real time detection under defying conditions validated by laboratory and field tests.
- Optimized and customized smoke detector algorithm deployed in ROS.

Abstract: Wildfires represent an escalating global threat to both the environment and society due to their frequency, duration, and expansion, which is exacerbated by climate change. Furthermore, recent heat waves underscore the critical need for swift detection and response mechanisms to curb wildfires from escalating into fully developed and uncontrollable stages. In that sense, this manuscript addresses an aerial smoke detector, composed of an unmanned aerial vehicle equipped with a camera, where visual information is sent to a ground station based on robotic operating system, where the proposed smoke detection methodology is deployed in real time. Such approach is formulated on an optimized YOLOv8 nano model, which is specifically trained using a customized data base with smoke under defying conditions. This solution ensures peak performance even within limited computational resources. The experimentally tests conducted first using images and videos, then, taking the video from the drone under controlled laboratory conditions, and finally by unstructured field experiments, such scenarios determine its robustness under such challenged conditions, producing confidence over 70%, and reducing the bias by validation metrics such as 95% of precision, and 88.5% of recall, respectively.

Keywords: real-time smoke detection; UAVs; YOLOv8; wildfires monitoring

1. Introduction

In recent years, wildfires have caused a nearly two-fold increase in tree cover loss compared to two decades ago. A 2022 study shows an annual increase in fire-related tree cover loss at a rate of 110,000 hectares 3% over the past two decades with climate change emerging as the primary driver, creating conditions conducive to more substantial and frequent wildfires through intense heat waves [1]. However,

only 10% to 15% of wildfires occur naturally, such as those ignited by electrical lightning, whereas the remaining $85\% - 90\%$ result from human activities, many of which exhibit clear intent like uncontrolled burnings in agriculture [2, 3]. The environmental impact of these escalating wildfires is profound, resulting in pollution, endangerment of species like koalas [4], and disruption to surface and underground aquatic ecosystems [1].

On the other hand, a recent case that illustrates the consequences of delayed wildfire treatment for the society occurred on the Greek island of Rhodes [5]. In the midst of the most extensive heat wave ever recorded in the country, a massive wildfire erupted, causing the evacuation of 30,000 people who sought refuge in hotels, schools, sports centers and conference centers. This not only endangered residents and tourists but also strained firefighters who had been combating over 500 fires nationwide in the preceding 12 days.

In light of this context, it is recognized that it is necessary to combat wildfires in their initial treatable phases with an economical, consistent, and scalable method that avoids risking third parties. Thus, the use of unmanned aerial vehicles (UAVs) has emerged as a viable solution. The efficiency of UAVs lies, among other reasons, in their cost, maneuverability, speed, and wide vision scope [6]. When integrated with vision system techniques, it can achieve effective detection of elements associated with wildfires. In particular, the main advantage of the UAV-based smoke detector is its wide perspective, access to complex and dangerous environments such as mountains, deep forests, volcanoes, etc., compared to human resources, and less expensive than the use of full-size helicopters and airplanes. Thus, has reported a comprehensive survey assessing existing techniques for the detection of fires in outdoor environments using image and video analysis to compare the pros and cons of proposed solutions [7], which underscored the concentration of existing solutions in the detection of fires [6, 8, 9]. However, these solutions have a notable drawback of late detection, typically identifying the fire when it is clearly discernible and the wildfire has escalated to an advanced stage, thereby complicating extinction and control efforts.

Therefore, smoke detection helps to get information about a possible wildfire in its early-stage providing more time for taking action. Many of these proposals involve detection from embedded systems, which limits the range and poses challenges for large distances [8, 10–12]. Other proposals detect both smoke and fire, introducing issues such as class imbalance, increased inference time, model complexity, and prolonged training time [10]. The persistent issue across all wildfire detection solutions is a lack of robustness. Despite this potential, existing systems encounter challenges such as slow response times and lack of robustness, rendering them impractical for effective implementation. Additionally, the need for specialized computing resources to achieve optimal performance further compounds these shortcomings. Addressing these challenges is crucial for improving the effectiveness of wildfire response and monitoring strategies. Thus, despite the use of several detection methods, one of the promising applications is YOLO models for real-time and accurate inference. For instance, in [13, 14] and [15] the performance of YOLO architectures for this designated task have been evaluated, with the state-of-the-art YOLOv8 excelling in balanced performance. Some works explore the integration of UAV-based vision systems with YOLO models or other deep learning approaches such as those reported in [15–18]. However, a common limitation is the requirement of specialized computational resources [11]. Moreover, many solutions are solely validated through datasets, introducing biases in their real-life performance, as explained in [6, 19]. Finally, it is worth noting that there are papers that demonstrate the potential for improved object detection model performance through optimization via the hyper-parameter tuning [20].

Thus, motivated by providing a deployable solution for early-stage wildfires detection, which may start in difficult or dangerous places to access, an aerial smoke detector is proposed, it is composed of an unmanned quadrotor vehicle equipped with a standard camera, and an algorithm based on YOLOv8n, which has been optimized and customized for smoke detection in defying conditions. Furthermore, this work different from existed approaches, contributes in the following ways:

- Deployment of a UAV-based smoke detector system based on YOLOv8 nano model, using limited resources such as transmission of images at 30 FPS with standard quality, enabling a rapid detection and after location of early-stage wildfires.
- Data assembly and labeling for more than 2000 smoke images are presented to enhance model robustness under challenging conditions such as blurriness, brightness, luminosity, avoiding cloud and mist, interference, *etc*.
- Testing in challenged scenarios: First, verify the detection methodology on a laptop and then increase the difficulty such as image transmission, brightness, luminosity, and obstacles between camera and smoke images in the laboratory conditions. Finally, field unstructured experiments are subject to, in addition to the above, image quality due to the distance of the object, and detection under dynamic conditions influenced by wind, sunlight, and mist. These comprehensive tests not only confirm the robustness of the model, but also verify the effectiveness of the proposed smoke detector, mitigating potential biases that may arise from relying solely on validation metrics.
- Scalable method for object detection model optimization, involving hyper-parameter tuning and OpenVINO conversion format, ensuring swift and accurate inferences in resource-constrained environments.

The rest of the paper is as follows: The smoke detection method is presented in Section 2, while Section 3 shows the details of the dataset and the parameters optimization, the experiments are addressed in Section 4, finally, some conclusions are drawn.

2. Smoke detection

In this section, the basis of the smoke detector algorithm based on YOLOv8 is addressed. The YOLOv8 represents one of the latest advancements in object detection surpassing its predecessors in terms of accuracy and inference speed as can be seen in Figure 1. This enhanced performance is demonstrated through evaluations using the COCO dataset, which is [21]. Beyond its detection capabilities, YOLOv8 is positioned as a versatile framework designed to address various aspects of a machine learning model's life-cycle through the implementation of few lines of code, covering tasks from training and validation to deployment and real-world tracking. This adaptability extends to diverse hardware platforms, ranging from edge devices to cloud APIs, thanks to its streamlined design. Ultralytics also offers integration with leading AI platforms to improve an AI workflow simplify tasks such as dataset labeling, training, visualization, and model management.

The distinctive characteristic of YOLO models lies in their ability to balance speed and accuracy by treating object detection as a regression problem, predicting bounding box coordinates rather than classification. YOLOv8 continues this legacy by refining the model architecture and introducing significant changes, such as anchor-free detection and mosaic augmentation during training.

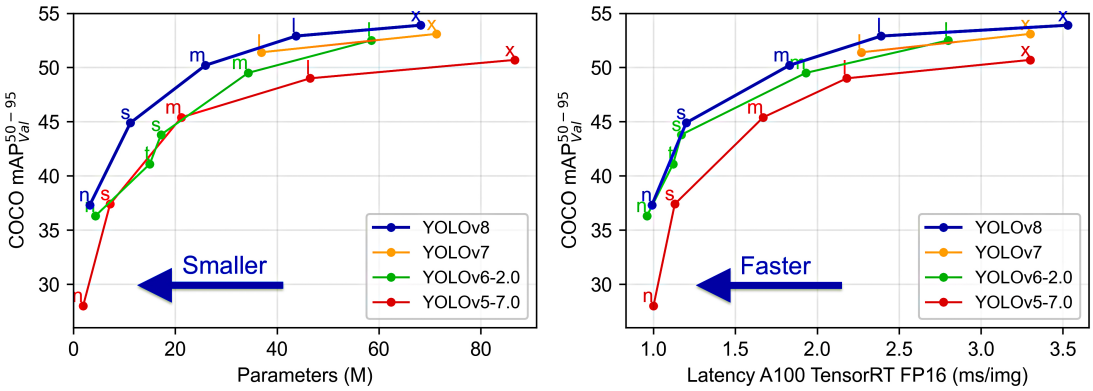


Figure 1. Benchmark of YOLO versions performance on COCO dataset [21]. Image Source: Ultralytics YOLO, by Jocher G, Qiu J, and Chaurasia A, licensed under AGPL-3.0.

Anchor-free detection signifies a shift where the model predicts the object's center directly, eliminating the need for offset predictions from known anchor boxes. This reduction in box predictions contributes to faster Non-Maximum Suppression (NMS), a post-processing step streamlining candidate detection after inference [22].

While an official paper is yet to be released, specific insights into the YOLOv8 model architecture, as analyzed by the GitHub user RangeKing, can be found.

Ultralytics, the developer of YOLOv8, also went a step further and has expanded its application beyond object detection to include segmentation and classification tasks. The YOLOv8 series offers models of varying sizes, labeled as yolov8-n-nano, s-small, m-medium, l-large, and x-extra-large. Notably, YOLOv8 Nano emerges as the smallest and fastest model, while YOLOv8 Extra Large (YOLOv8x) represents the most accurate albeit slower model exhibited in Figure 2. The model size demonstrates a linear relationship between the mean average precision (mAP) and an inverse relationship with the inference time. Considering the need for a lightweight real-time detection model, the YOLOv8 Nano model was chosen for deployment.

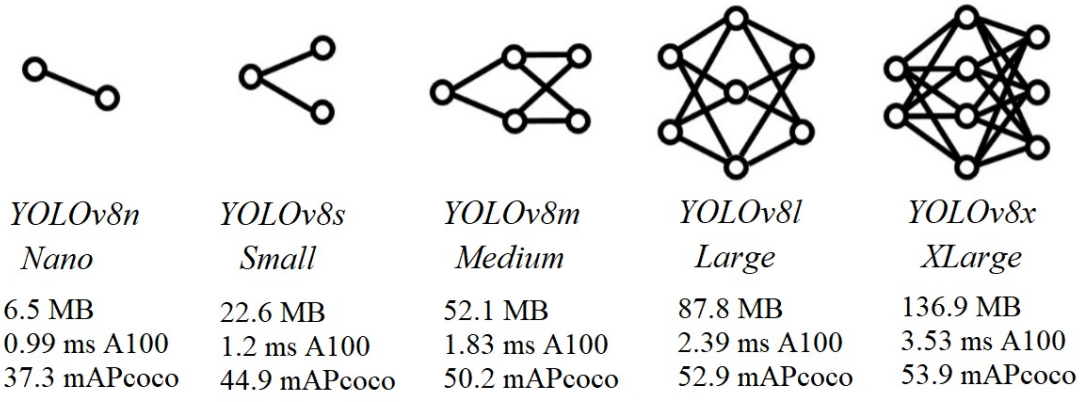


Figure 2. Benchmark of YOLOv8 model size performance on COCO dataset [21]. Image Source: Ultralytics YOLO, by Jocher G, Qiu J, and Chaurasia A, licensed under AGPL-3.0.

To achieve fast and cost-effective computing, we leveraged the Kaggle platform for the deployment of the code to train our model. Kaggle not only provides a convenient environment for uploading our datasets but also offers free limited access to GPUs to minimize training duration. Specifically, the NVIDIA TESLA

P100 was the preferred GPU for its proven effectiveness. For a detailed illustration of the Kaggle Kernel employed in the training process (you can access here: https://www.kaggle.com/jesusbgonzaleza/training-yolov8-smoke-detection-model-public).

It is noteworthy that, during the training process, several loggers integrated with YOLOv8 for experiment tracking, management, and visualization were employed. Concretely, CometML, ClearML and W&B. In the Kaggle Kernel shared above, a seamless connection to these loggers have established, allowing users to choose their preferred logger based on individual preferences.

Following a training process initially configured for 200 epochs and optimized as detailed in the upcoming sections, the model with the highest accuracy was retrieved through early stopping at epoch 103. This trained model to detect smoke was employed across a set of images not encountered during training, providing a comprehensive assessment of its performance. The resulting plots in Figures 3a–3d illustrate the behavior of the model.

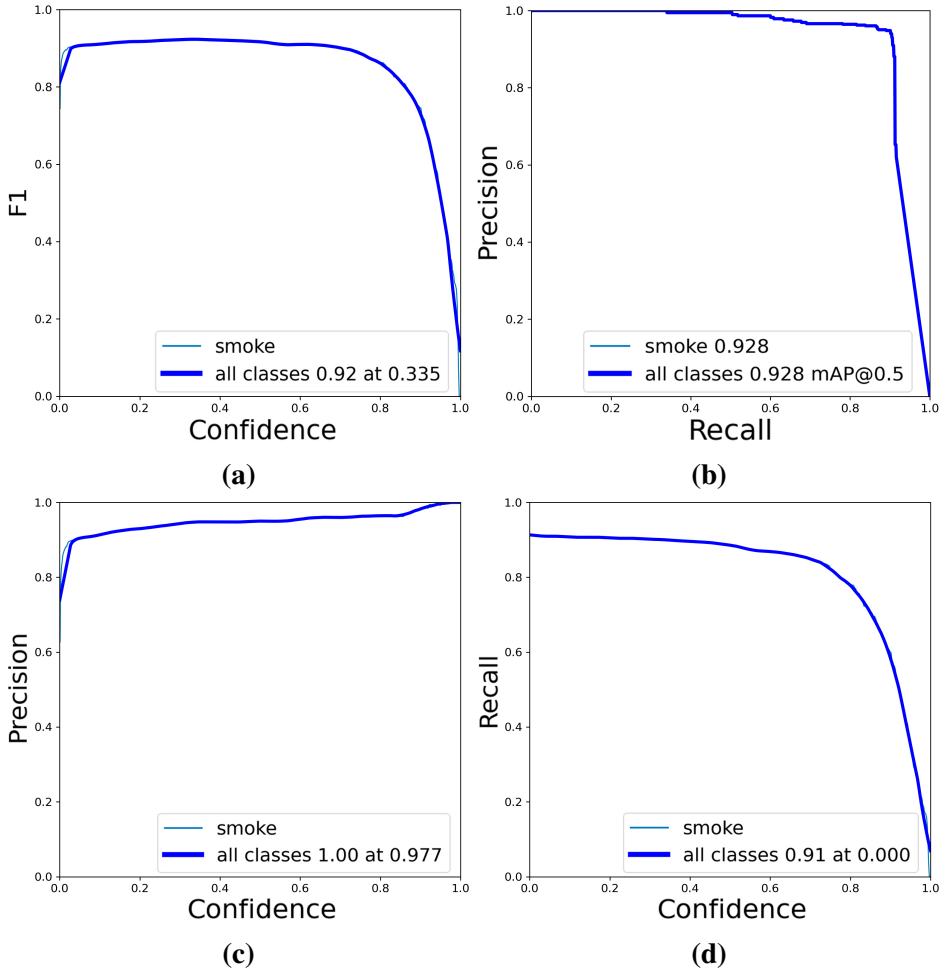


Figure 3. Confidence performance. **(a)** F1 Curve v. Confidence, **(b)** Precision v. Recall. **(c)** Precision v. Confidence, and **(d)** Model Recall v. Confidence.

As observed, the application of confidence and IoU (Intersection over Union) thresholds during inference plays a crucial role in fine-tuning the visualized predictions. Opting for a higher confidence threshold enhances the precision of the model at the cost of lower recall and vice versa. Analyzing, the plots reveals that a harmonious balance between precision and recall can be achieved with an approximated

confidence threshold of $0.4/0.5$. Nevertheless, to minimize the occurrence of false positives, a higher value of 0.7 has been selected. Additionally, an IoU threshold for inference of 0.1 has been chosen, considering smoke as a cohesive entity that should not exhibit self-overlapping.

For the deployment of the YOLOv8 model, we were constrained to the use of an Intel Core(TM) i5-7200U 2.50GHz CPU. This limitation affected the minimum inference times that could be reached and introduced variability in performance based on concurrent computer activities. To address this, the model underwent conversion to a format conducive to swift deployment on this hardware. OpenVINO optimization was chosen for this purpose, as it is a toolkit specifically designed to optimize and deploy deep learning models on Intel hardware, involving the exportation of the model to an intermediate representation. It is important to highlight that while this process significantly enhances inference speed, a marginal trade-off is encountered, resulting in a slight reduction in accuracy.

Following this optimization and the determination of threshold values, the model was deployed in Python, employing the same parameters and pre-processing techniques used during training. In this instance, image adjustments were made to a size of 640×640 pixels to ensure performance consistency with the training results. Further details of the implementation can be explored in the code available in the following GitHub repository: https://github.com/Jesus-Alayon/smoke_detection_yolov8_model/blob/94cd0abea637342e376d2afb6f119bc4e0541a06/Smoke_Detection_YOLOv8_script%20public.py.

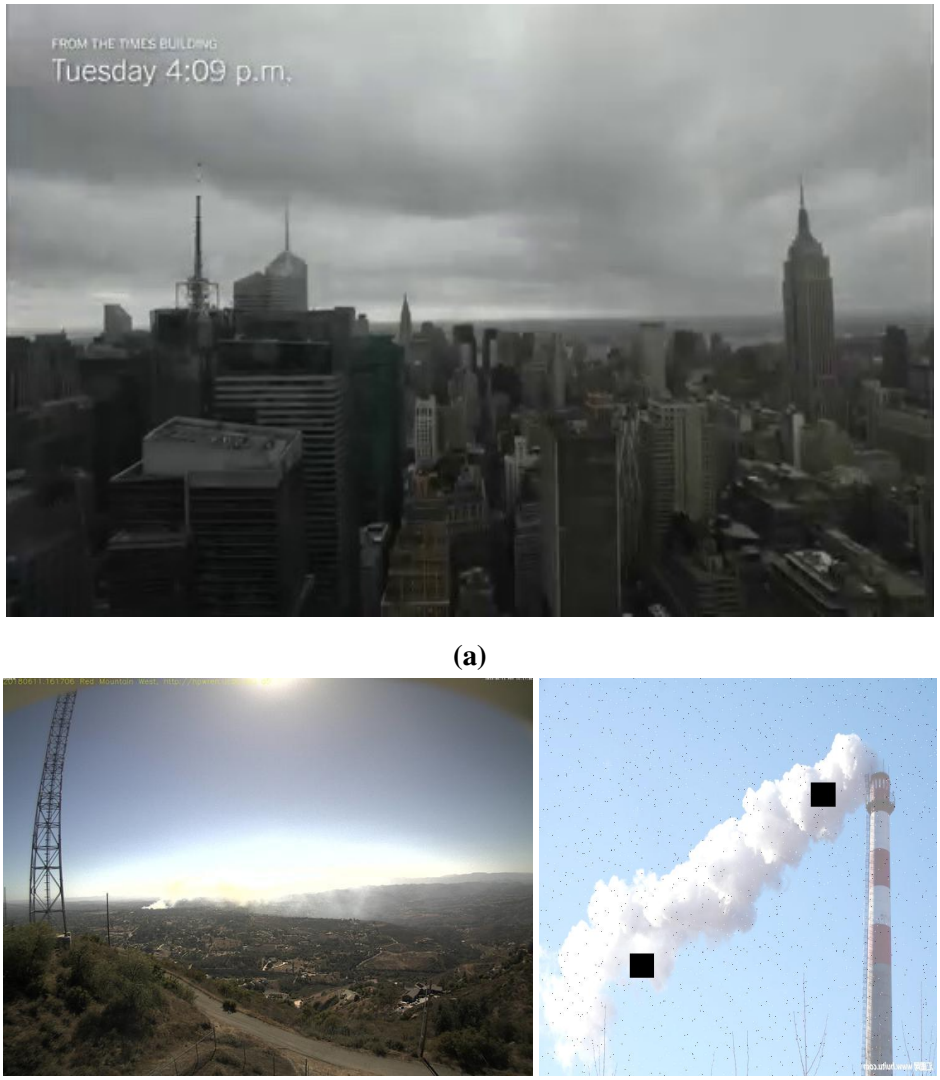
3. Dataset

The initial dataset comprised $14,808$ meticulously selected images sourced from various datasets, including those reported in [23–25]. This diverse dataset encompasses images with varying imperfection levels, scenarios, points of view, resolutions, smoke density types, and features. Additionally, negative samples are incorporated to restrain the model from false detection of elements resembling smoke features (see Figure 4a). This dataset furnished ample information to effectively train a model to navigate and address real-world challenges in detection. During the image labeling process, the dataset underwent size reduction due to challenges in establishing accurate ground truth bounding boxes for certain samples. Such reduction was prompted by difficulties arising from the inherent characteristics of smoke, image resolution, artifacts and contrast with the background, rendering it imperceptible for the human eye to confidently delineate boundaries in fuzzy smoke images as Figure 4b.

The precision of indicating the object during training is paramount. Failure to accurately represent the object, such as omitting pixels due to incomplete object coverage, can impede the learning process of the model by introducing inconsistent information. This underscores the importance of meticulous labeling to ensure optimal model training. In that sense, we consider that labeling conflicted images is crucial to achieve model robustness. Thus, building upon the information provided earlier, 2001 images were labeled using the Roboflow platform as annotation tool due to its seamless compatibility with YOLOv8, its valuable assets like the magic wand for automated labeling, its user-friendly interface, free limited access and inclusion of features such as dataset management, batch annotation workflows and multiple exportation and conversion formats.

Furthermore, after completing the labeling process, Roboflow facilitated the organization and distribution of images into sets such as train, validation, and test, each serving a distinct purpose. This platform not only streamlines dataset management but also offers additional benefits such as final pre-processing options for the images and the application of random augmentations to enhance the

training set. The dataset was exported by resizing the images to 640×640, applying auto-orientation, and filtering the images to retain only 5% as negative samples. Additionally, several augmentations have introduced such as Figure 4c to the training set, including horizontal flipping, brightness modification within the range of -25% to 25%, exposure modification within the range of -25% to 25%, the introduction of 1% pixel noise, and the inclusion of 2 cutouts, each covering 6% of the image. The dataset is exported in the YOLOv5 PyTorch TXT annotation format, a modified version of the Darknet annotation format. The final dataset exported with a total of 3107 images is available in Kaggle with the following link: https://www.kaggle.com/datasets/jesusbgonzaleza/smoke-representative-dataset-v3.



(a)

(b) **(c)**

Figure 4. **(a)** Negative sample in dataset, **(b)** fuzzy smoke image discarded, and **(c)** augmented image in training set. Images Source: authors from [23–25], under Creative Commons Attribution 1.0, and 4.0 International License, respectively.

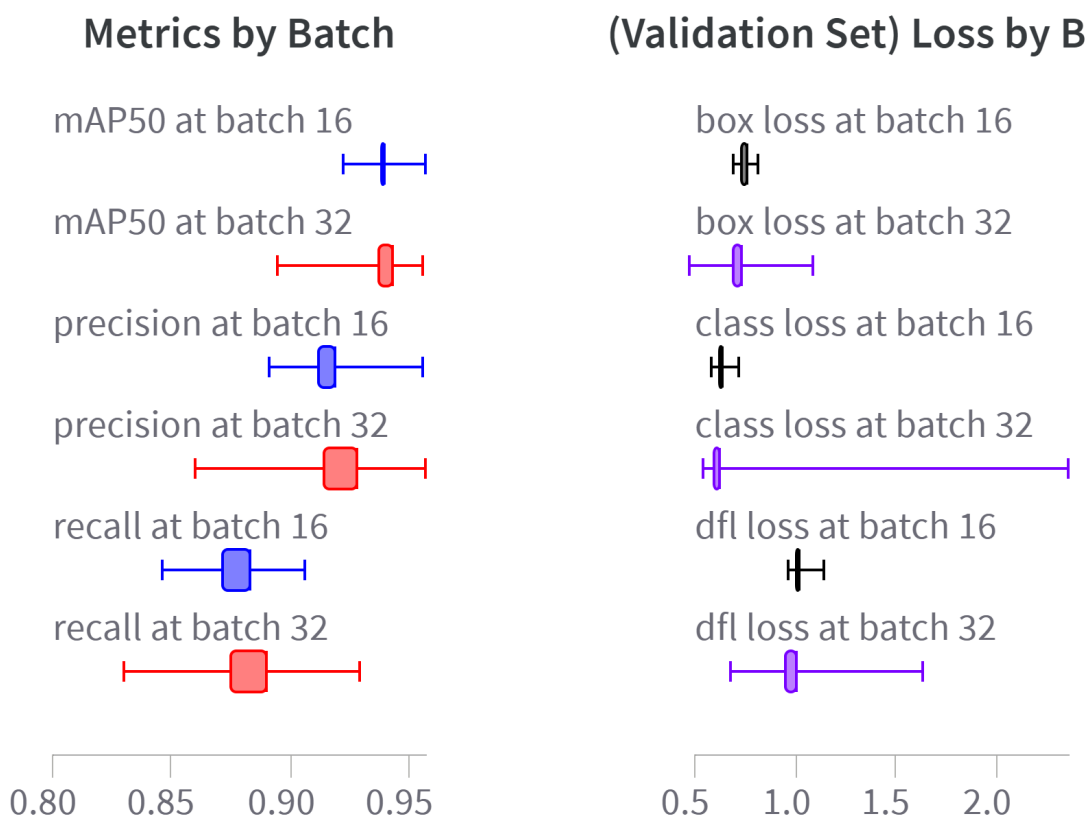
3.1. Hyper-parameter optimization

It is well-established that, beyond the dataset itself, the hyper-parameters chosen for the model training process play a crucial role in determining its performance as explained in [18]. Relying solely on default

settings may introduce biases during model benchmarking and prevent from fully realizing the potential of the model. As a result, the hyperparameters used in the training of a YOLOv8 nano model for smoke detection were systematically optimized.

Given the constraints of limited computational resources and the time-intensive nature of the task, Bayesian optimization was primarily utilized. This method focuses on the most promising values to optimize a predefined metric. To streamline the tuning process, the number of runs, epochs per run, and defined tryout values for each hyper-parameter were simplified due to these resource considerations.

For fast and low-cost computing, the platform Kaggle as mentioned above was used to deploy the process. On rare moments when the quota limit per week of GPU was exceeded, a handful of runs on Google Colab which offered a similar service, but slower and unreliable was made. On the other hand, the platform Weights & Biases has been considered for the automation, tracking and analysis of the hyper-parameter search. For a more in-depth understanding of the implementation details, it can delve into the Kaggle Kernel available at the following link: https://www.kaggle.com/jesusbgonzaleza/w-b-hyperparameter-optimization-yolov8-public.



(a) **(b)**

Figure 5. Criteria for hyper-parameter optimization.

Finally, the optimization of hyper-parameters was restricted to those deemed to have a more significant impact on training an object detection model. After conducting multiple sets of runs with various modifications, the most promising values for each of the mentioned hyper-parameters were identified, and

stated in the following Table 1, and Figures 5a-5b, respectively.

Table 1. Hyper-parameter setup.

Hyper-parameter	Value	Hyper-parameter	Value
batch size	32	Optimizer	SGD
Cosine learning rate scheduler	True	AMP training	False
initial learning rate	0.0036	final learning rate	0.0001
momentum	0.88	weight decay	0.00035
box loss gain	5	class loss gain	0.5
DFL gain	1.5	normal batch size	64
epochs	Defined by early stopping	patience	50

Notice that this represents the most optimal configuration found within the constraints of the available resources, time per run, number of attempts, and influence of non-deterministic behaviors during the runs. Consequently, further analyses can be conducted, there exists the potential to identify an even more optimal configuration. Considering the infinite number of possible combinations, the decision to continue with the analyses or conclude is at the discretion of the users. For a more in-depth understanding of the hyper-parameter tuning process, please refer to the following document.

3.2. Aerial smoke detector deployment

For the implementation of the smoke detector within a UAV-based vision system, the Robomaster Tello Talent drone have been leveraged, which its ease of programming in Python through the Robomaster SDK and the presence of an onboard camera made it a suitable choice [26].

Figure 6. Quadrotor Tello Talent.

The TT drone (see Figure 6) can deliver a consistent video stream, configurable through the SDK [27]. The following configuration has been chosen: a resolution of $720 \times 1020p$, a bitrate of 5Mbps, and a frame rate of 30 fps. Subsequently, robotic operating system ROS was selected to facilitate node threading. Specifically, one node for continuous information exchange with the drone, a second node to issue flight instructions and a third node to execute YOLOv8 model inference on the received images. Additionally, the second node facilitated manual control over the motion of the drone or the initiation of a predefined routine. A pseudo code in Algorithm 1 and flow chart in Figure 7 are depicted the proposed methodology, respectively. Further insights into the implementation details can be explored by reviewing the GitHub repository provided in the following link: https://github.com/Jesus-Alayon/smoke_detection_yolov8_model.git.

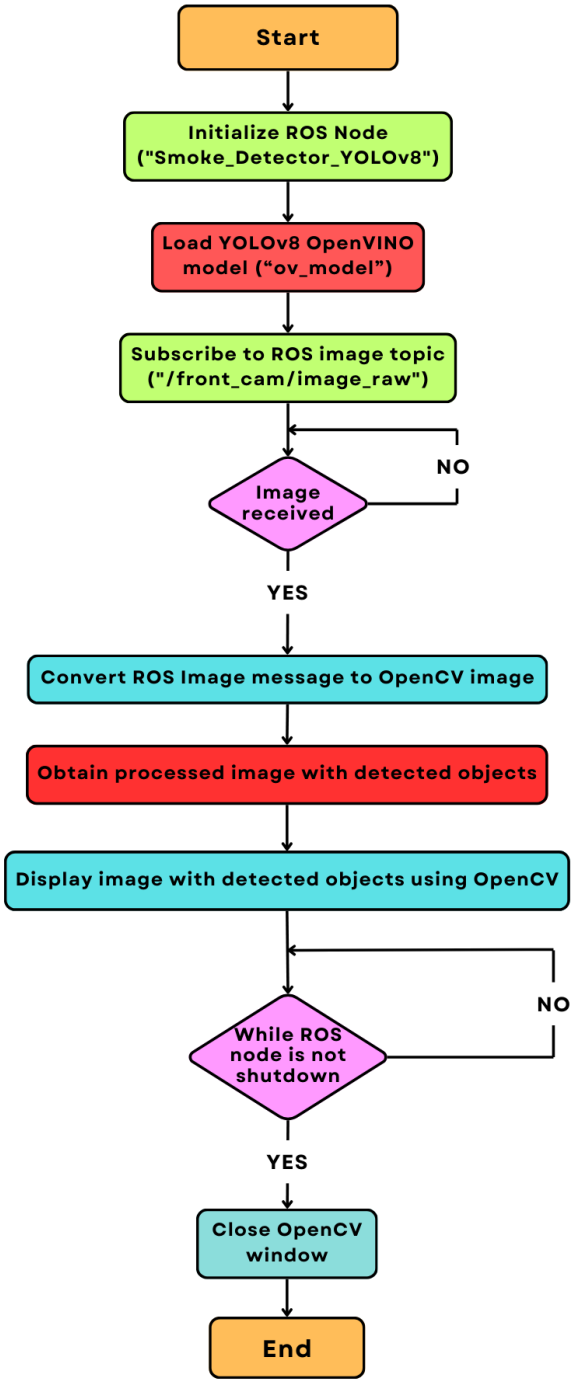


Figure 7. Drone smoke detection algorithm.

Algorithm 1 ROS Node for smoke detection using YOLOv8

Require: ROS, OpenCV, ultralytics.YOLO, CvBridge

function CALLBACK(msg)
    $image \leftarrow$ bridge.imgmsg_to_cv2(msg) ▷ Convert ROS image message to OpenCV image
    $results \leftarrow$ ov_model(cv2.resize($image, (640, 640)$), $iou = 0.1, conf = 0.7$) ▷ Detect objects
    cv2.imshow($'YOLOv8Interface'$, results[0].$plot()$) ▷ Display detected objects
    cv2.waitKey(1) ▷ Refresh OpenCV window
end function

function MAIN
    rospy.init_node($'Smoke\_Detector\_YOLOv8'$) ▷ Initialize ROS node
    ov_model $\leftarrow$ YOLO('path_to_model/', task = 'detect') ▷ Load YOLOv8 model
    bridge $\leftarrow$ CvBridge() ▷ Initialize CvBridge
    sub $\leftarrow$ rospy.Subscriber($'/front\_cam/image\_raw'$, Image, callback, queue_size = 1) ▷ Subscribe to image topic
    print("$Detectingnow...$")
    while ¬rospy.is_shutdown() **do** ▷ Keep the node running
        rospy.spin()
    end while
    cv2.destroyAllWindows() ▷ Close OpenCV window
end function

4. Experimental results

In this section, experimental results are addressed in order to evaluate the proposed aerial smoke detector performance. In that sense, three different tests under increased challenging conditions were conducted:

4.1. In the laptop

The initial experiment involved detection using a variety of videos sourced from our collected datasets. The algorithm and the source images are running on a laptop equipped with an Intel Core(TM) i5-7200U 2.50GHz CPU, avoiding communication and frequency issues related to image transmission. The purpose is to verify the trained methodology. Some results are showed in Figure 8, where the smoke detection of several situations, smoke color and density can be observed, and each detection is illustrated in a bounded box with its corresponding percentage of accuracy.

4.2. Laboratory conditions

The second test consisted of guiding the TT drone along a designated flight path that passed in front of multiple monitors displaying smoke and non-smoke images. The closed loop of the drone-camera, the detection algorithm, and the motion commands to the drone are illustrated in Figure 9.

Figure 8. Results of the detector running on the laptop. Images Source: Images Source: authors from [23–25], under Creative Commons Attribution 1.0, and 4.0 International License, respectively.

The drone moves continuously to provide visual feedback at 30FPS to the fixed station (laptop) by wi-fi, where light, distance, and blurriness affect the performance of the proposed smoke detector. Three monitors displayed a loop of three photos each: two with distinct smoke presentations and a third one resembling smoke features. The laboratory contains ropes and nets around the test area, a dark environment from the lab floor (shiny), reflecting luminosity. In addition, the monitors are located outside the net. Then, the aforementioned interferes with the image from the drone and the images displayed on the monitors, representing non-ideal operating conditions, verifying robustness of the model under such conditions.

Each test session lasted approximately a minute. The routine was orchestrated through ROS. Conversely, the inferences of the model were made through a 0.7 confidence threshold and displayed via a pop-up window.

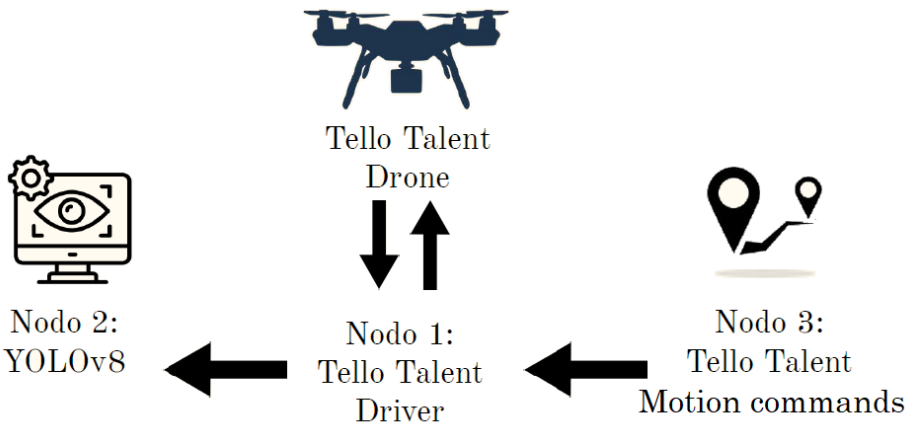


Figure 9. Flow diagram of the smoke detector.

4.3. Field experiments

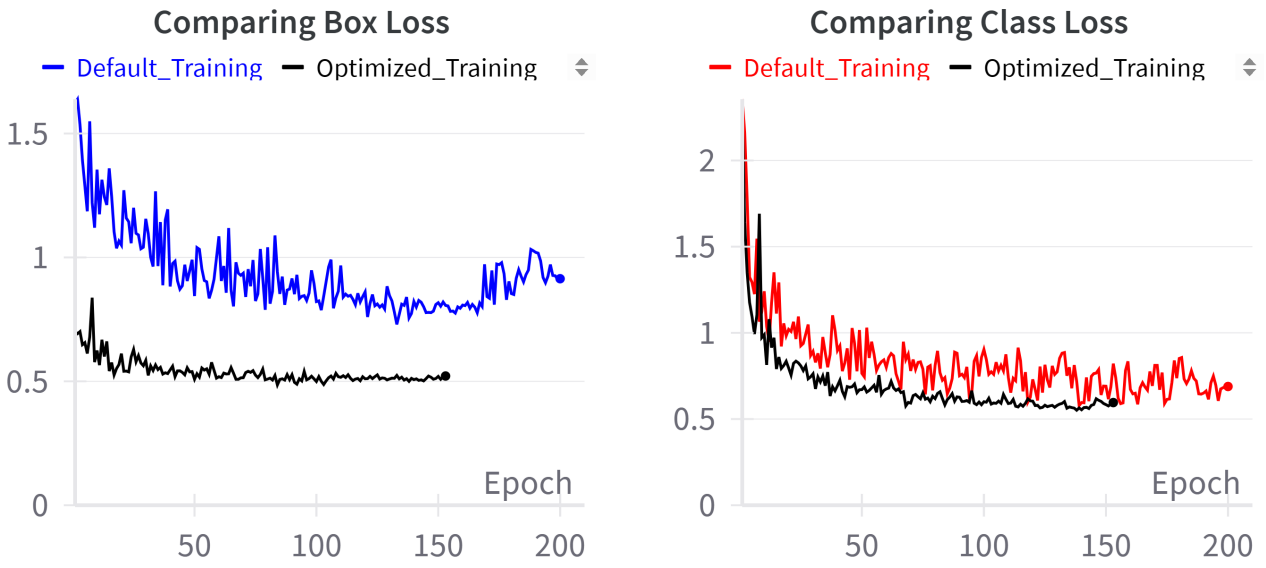
The test entailed guiding the drone along a different predefined path, circling a building and covering greater distances, following the same closed loop configuration displayed in Figure 9. However, in this scenario, the communication between the drone and computer could present delays, image quality issues stemming from the distance of the object, and the difficulty of detecting smoke in dynamic conditions influenced by wind, sunlight, and mist, represents a more challenged test than the previously addressed. The experiment also was employed a real smoke using smoke flares. It is remarkable that the same confidence threshold 0.7 as laboratory conditions was maintained. Additionally, A picture of the laboratory set as well as field experiments is given in Figure 10.

Figure 10. Quadrotor executing the smoke detection. Top Laboratory, and bottom Field experiments.

For a visual demonstration of the experiments, refer to the video available in https://youtu.be/DtLGGAuJNEU. On the other hand, the final model performance on the validation set yielded the following metric values: 95% precision, 88.5% recall, 91.8% mAP50, and 76% mAP50-95, extracted from Table 2. This robust performance on unseen data is corroborated by its deployment in real-life scenarios, as evidenced the experiment results. The model demonstrated minimal false positive predictions, effectively detecting smoke images even when they were presented as burned images due to the high brightness of the monitors.

However, it is notable that the model encounters challenges in detection when faced with numerous obstacles and increased distances. Despite this, the model maintains high recall and minimal false positives, initially observed in obscure scenarios. Therefore, a comparison is provided between the models

with and without hyperparameter tuning and exportation in OpenVINO. This highlights the enhancements in performance and inference speed. The plots in Figures 11a and 11b distinctly illustrate the overfitting of the model trained with the default configuration, as evidenced by significant and unstable loss values during inferences on the validation set.



(a) **(b)**

Figure 11. Overfitting evidenced with loss during training.

Table 2. Model benchmarking.

Metrics	Hyperparameter and Optimization	Hyperparameter and OpenVINO Optimization
mAP50	92.8%	91.8%
mAP50-95	78.0%	76.0%
Precision	94.8%	95.0%
Recall	90.0%	88.5%
Inference Time	3.5 fps	33.3 fps

5. Conclusions and further work

A real-time implementation of a UAV-based smoke detector system using YOLOv8 was developed. In this regard, smoke images were gathered and labeled to enhance the robustness of the proposed aerial detector. The method was parameterized in such a way that confident smoke detection was achieved. Furthermore, three experiments were conducted to evaluate performance and effectiveness under diverse and challenging conditions. Firstly, through offline detection, followed by deployment on a UAV under laboratory conditions and ultimately, in field experiments which posed greater complexity compared to the preceding tests, confirming the advantages. Based on the aforementioned, this proposal can facilitate early wildfire localization in their initial stages, even when deployed on devices possessing limited computational power. Finally, this work enhanced the performance of object

detection models, regardless of the context or resources in which they are deployed. The described workflow can be adapted to optimize models in other scenarios, especially when using images from the specific context. For future work, this work can be employed in collaboration with rapid response strategies, such as the deployment of designated drone fleets for fire-extinction, the potential for a tangible mitigation in the frequency and impact of wildfires arises. This translates into less pollution, increased safety for both society and the environment, and a reduction in annual costs and resources allocated to wildland firefighting.

Acknowledgments

The authors thank to the Multi-robot system laboratory of the Tecnologico de Monterrey their support with work space and facilities to carried out the experiments.

Conflicts of Interests

Authors declare that they do not have any conflict(s) of interest.

Author's contribution

Conceptualization, methodology, validation, writing—review and editing, Herman Castañeda; methodology, software, writing—original draft preparation, Jesus Gonzalez-Alayon. All authors have read and agreed to the published version of the manuscript.

References

[1] MacCarthy J, Tyukavina S, Weisse M, Harris N. The Latest Data Confirms: Forest Fires Are Getting Worse. World Resources Institute. 2023, Available: https://www.wri.org/insights/nuevos-datos-los-incendios-forestales (accessed on 20 July 2024).

[2] Morales F. [In Spanish] Quemas agrícolas ponen en riesgo vidas humanas. CIMMYT. 2023, Available: https://www.cimmyt.org/es/noticias/quemas-agricolas-ponen-en-riesgo-vidas-humanas/#:~:text=Las%20quemas%20agr%C3%ADcolas%20que%20se,fomentadas%20para%20reducir%20estos%20riesgos (accessed on 25 July 2024).

[3] Wolters C. Wildfires can move shockingly fast. Here's how they start—and how to stop them. National Geographic. 2023, Available: https://www.nationalgeographic.es/medio-ambiente (accessed on 29 July 2024).

[4] Daly N. No, koalas aren't 'functionally extinct'—yet. National Geographic. 2019, Available: https://www.nationalgeographic.com/animals/article/koalas-near-extinction-myth-australia-fires (accessed on 30 June 2024).

[5] Mendonca D, Liakos C, Labropoulou E. [In Spanish] Turistas huyen del incendio de Rodas en la mayor evacuación de la historia de Grecia. CNN. 2023, Available: https://cnnespanol.cnn.com/2023/07/23/turistas-incendio-rodas-evacuacion-grecia-trax/ (accessed on 25 July 2024).

[6] Chen X, Hopkins B, Wang H, O'Neill L, Afghah F, *et al*. Wildland Fire Detection and Monitoring Using a Drone-Collected RGB/IR Image Dataset. *IEEE Access* 2022, 10:121301–121317.

[7] Chaturvedi S, Khanna P, Ojha A. A survey on vision-based outdoor smoke detection techniques for environmental safety. *ISPRS J. Photogramm. Remote Sens.* 2022, 185:158–187.

[8] Hashemzadeh M, Zademehdi A. Fire detection for video surveillance applications using ICA K-medoids-based color model and efficient spatio-temporal visual features. *Expert Syst. Appl.* 2019, 130:60–78.

[9] Sadeghi S. Mixed reality and remote sensing application of unmanned aerial vehicle in fire and smoke detection. *J. Ind. Inf. Integr.* 2023, 15:42–49.

[10] Saponara S, Elhanashi A, Gagliardi A. Real-time video fire/smoke detection based on CNN in antifire surveillance systems. *J. Real-Time Image Process.* 2020, 18:889–900.

[11] Sun B, Wang Y, Wu S. An efficient lightweight CNN model for real-time fire smoke detection. *J. Real-Time Image Process.* 2023, 20(74):1–12.

[12] Zhou Z, Shi Y, Gao Z, Li S. Wildfire smoke detection based on local extremal region segmentation and surveillance. *Fire Saf. J.* 2016, 85:50–58.

[13] Casas E, Ramos L, Bendek E, Rivas-Echeverría F. Assessing the Effectiveness of YOLO Architectures for Smoke and Wildfire Detection. *IEEE Access* 2023, 11:96554–96583.

[14] Jia X, Wang Y, Chen T. Forest Fire Detection and Recognition Using YOLOv8 Algorithms from UAVs Images. In *5th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, Shenyang, China, July 14–16, 2023, pp. 646–651.

[15] Johnston J, Zeng K, Wu N. An Evaluation and Embedded Hardware Implementation of YOLO for Real-Time Wildfire Detection. In *2022 IEEE World AI IoT Congress (AIIoT)*, Seattle, USA, June 6–9, 2022, pp. 138–144.

[16] Dilli B, Suguna M. Early Thermal Forest Fire Detection using UAV and Saliency Map. In *5th International Conference on Contemporary Computing and Informatics (IC3I)*, Uttar Pradesh, India, December 14–16, 2022, pp. 1523–1528.

[17] Jiao Z, Zhang Y, Mu L, Xin J, Jiao S, *et al*. A YOLOv3-based Learning Strategy for Real-time UAV-based Forest Fire Detection. In *Chinese Control And Decision Conference (CCDC)*, Hefei, China, August 22–24, 2020, pp. 4963–4967.

[18] Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 2020, 42(2):318–327.

[19] Liu H, Lei F, Tong C, Cui C, Wu L. Visual Smoke Detection Based on Ensemble Deep CNNs. *Displays*, 2021, 69:1–10.

[20] Aditya N, Maliah S, Abisado M, and Avelino G. Toward Accurate Fused Deposition Modeling 3D Printer Fault Detection Using Improved YOLOv8 With Hyperparameter Optimization. *IEEE Access* 2023, 11:74251–74262.

[21] Jocher G, Qiu J, Chaurasia A. Ultralytics YOLO. 2023, Available: https://github.com/ultralytics/ultralytics/tree/main (accessed on 17 July 2024).

[22] Solawetz J, Francesco. What is YOLOv8? The Ultimate Guide. Roboflow. 2023, Available: https://blog.roboflow.com/whats-new-in-yolov8/#the-yolov8-annotation-format (accessed on 27 July 2024).

[23] Hopkins B, O'Neill L, Afghah F, Razi A, Rowell E, *et al*. FLAME 2: Fire detection and modeLing: Aerial Multi-spectral imagE dataset. *IEEE Dataport* 2022.

[24] Grammalidis N, Dimitropoulos K, Cetin E. FIRESENSE database of videos for flame and smoke detection. Geneva: Zenodo, 2017. Available: https://doi.org/10.5281/zenodo.836749 (accessed on 27 July 2024).

[25] Steffensbola, furg-fire-dataset. 2019, Available: https://github.com/steffensbola/furg-fire-dataset (accessed on 27 July 2024).

[26] Robomaster TT Tello Talent User Manual. 2021, Available: https://dl.djicdn.com/downloads/RoboMaster+TT/RoboMaster_TT_Tello_Talent_User_Manual_en.pdf (accessed on 10 July 2024).

[27] Robomaster TT SDK 3.0 User Guide. 2021, Available: https://dl.djicdn.com/downloads/RoboMaster+TT/Tello_SDK_3.0_User_Guide_en.pdf (accessed on 10 July 2024).