MDPI

MDPI

Article

Smoke Detection on the Edge: A Comparative Study of YOLO Algorithm Variants

Iosif Polenakis *, Christos Sarantidis, Ioannis Karydis and Markos Avlonitis

Department of Informatics, Ionian University, 49100 Corfu, Greece; csarantidis@ionio.gr (C.S.); karydis@ionio.gr (I.K.); avlon@ionio.gr (M.A.)
* Correspondence: ipolenakis@ionio.gr

Abstract

The early detection of smoke signals due to wildfires is vital in containing the extent of loss and reducing response time, particularly in inaccessible or forested areas. For lightweight object detection, this study contrasts the YOLOv9-tiny, YOLOv10-nano, YOLOv11-nano, YOLOv12-nano, and YOLOv13-nano algorithms in determining wildfire smoke at extended ranges. We present a robustness- and generalization-checking five-fold cross-validation. This study is also the first of its kind to train and publicly benchmark YOLOv10-nano up to YOLOv13-nano on the given dataset. We investigate and compare the detection performance against the standard performance metrics of precision, recall, F1-score, and mAP50, as well as the performance metrics regarding computational efficiency, including the training and testing time. Our results offer practical implications regarding the trade-off between pre-processing methods and model architectures for smoke detection when applied in real time on ground-based cameras installed on mountains and other high-risk fire locations. The investigation presented in this work provides a model in which implementations of lightweight deep learning models for wildfire early-warning systems can be achieved.

Keywords: YOLO algorithm; smoke detection; fire detection; wildfire; edge computing; cameras

Academic Editors: Jozef Juhár and Lyudmila Mihaylova

Received: 31 July 2025
Revised: 15 September 2025
Accepted: 13 October 2025
Published: 4 November 2025

Citation: Polenakis, I.; Sarantidis, C., Karydis, I.; Avlonitis, M. Smoke Detection on the Edge: A Comparative Study of YOLO Algorithm Variants. *Signals* **2025**, *6*, 60. https://doi.org/10.3390/signals6040060

1. Introduction

Wildfires and urban fires have become a vital concern, becoming rampant over the years. Wildfires that spread at a fast rate and present a threat to human life, building structures, and biological habitats are present too. To minimize such potential threats, advanced prediction and emergency response in a reasonably short period continue to be critical. Computer vision and artificial intelligence (AI) are facilitative of each other and offer real-time detection and decision-making capabilities, resulting in integrated smoke-monitoring systems.

Object detection or locating and further detection of objects of interest in images are the primary features of modern safety systems. Modern systems of object detection involving deep learning methods and convolutional neural networks (CNNs) are applicable to the single-stage (and multi-stage) detection of hazardous elements of the surrounding environment, such as those related to smoke or construction, in different operational conditions.

This paper investigates the performance of the YOLO object-recognition models, which are useful in the process of detecting smoke in visual images. In our research, we experimented on several variants of YOLO so as to determine which variant can be said to offer the most optimal solution to the problem of instant smoke detection.

1.1. Object Recognition

The primary goal of object recognition techniques is the identification and classification of a series of elements that are represented in digital images. The two main functions achieved by the system are object classification and object localization. The detection system does not only output labels of categories; it also shows identified objects visually. An explanation of simple classification techniques is included. This procedure employs bounding boxes in order to denote where objects are on the pictures spatially.

YOLO models [1] are a particular form of model that offers the ability to obtain higher rates of accuracy in prediction and a lower level of latency in inference in scenarios that demand high performance. Given that urgent tasks for functions like smoke detection demand immediate responses, it is reasonable that YOLO would be implemented in terms of achieving inclusive classification and localization tasks.

1.2. Smoke Recognition

Detecting smoke by taking visual input through the computer is an important aspect of developing warning systems for man-made and natural disasters that provide early enough warnings. Even during the initial phases of fire development, the most well-known visual signs include flames, smoke plumes, and color differences in temperature [2]. The proper detection of these signs through the imaging method enhances the emergency response systems to a great extent.

The essential characteristics of smoke make it hard to identify. It is semi-transparent, and it seems fluid as it reacts to changing ambient lighting conditions and the presence of wind. The YOLO deep learning model and its improved version can learn various visual qualities on training data and hence distinguish safe and dangerous visual aspects, while it can achieve remarkable performance in terms of false alarms.

1.3. Related Work

Recently synthesized wildfire smoke advancements have used convolutional and Transformer-inspired deep learning designs. Earlier systems detected smoke via computer vision and augmented reality techniques [3]. Techniques loosely based on the CNN architecture, i.e., enhanced YOLOv4, provided significant improvements over their precursors, and quantifiable improvements in smoke detection performance were imparted [4]. Additionally, the lightweight versions of YOLO have gained ground in real-time wildfire detection. Alkhammash et al. [5] compared YOLOv9-tiny and YOLOv10-nano to YOLOv11-nano on wildfire detection tasks and found that the latter outscored the former (precision 0.845, mAP50 0.859). Li et al. [6] also built upon YOLOv11, adding a multi-scale convolutional attention (MSCA) module to enhance smoke detection. Liu et al. [7] introduced the single-shot multiBox detector (SSD), an efficient and effective single-stage object detector that regresses the bounding-boxes and the class-probabilities on the multi-scale feature-maps and default-boxes; they evaluated the technique with the dataset given in [8]. Wang et al. [9] proposed the Transformer-based cross contrast patch embedding (CCPE) with a modified Swin Transformer that deals with low-level feature mobilization in smoke identification and tested it on the novel RealFire benchmark. EFA-YOLO [10] is a lightweight attention-enhanced YOLO model that also claims ultra-efficient detection and is thus fit to be deployed on the edges. These studies are an indication of the usefulness of architecture improvements and domain-aware pre-processing, but none of them evaluated multi-step pre-processing methods (e.g., HEQ, Cropping) or YOLOv13-nano. These are gaps that our study partially (the former) and, in parts, fully (the latter) fills. Moreover, in [11], the authors experimented with YOLOv9-tiny on a larger dataset that contained

the one that our research used, without using the five-fold cross validation method. They achieved a mAP50 of 0.885 in a multimodal dataset containing 2321 images.

Recently, in [12] Yulei et al. propose a feature pyramid network in order to address the model's ineffectiveness to detect small objects. In [13] Liau and Wong replace the back-end network of SSD by more efficient networks such as SqueezeNet and MobileNet in order to achieve real-time speed on edge devices. Ahmed et al. in [14] compare different versions of SSD-MobileNet architectures with their own one named EAC-FD. Their model demonstrates a nice balance between accuracy and computational complexity. In [15] Jingwu et al. propose a target detection network, which is based on an improved feature pyramid structure and solves the problem of feature loss in the down-sampling process, when it is running on edge devices. Their findings conclude that it is ideally suitable for real time fire-detection. In [16] Huang et al. propose a lightweight smoke detection model which reduces the parameters and increases speed, while maintaining accuracy in comparison with the model compared. Vasquez et al. in [17] explore the performance of transfer learning in smoke and wildfire detection. Their experiments where conducted using edge devices and different YOLO variants were also taken into consideration. Wang et al. in [18] investigate YOLOv8 with the addition of 3 new modules for improvement in order to tackle the smoke recognition problem. Their findings surpass state-of-the-art smoke detection systems. In [19] Li et al. investigate a fire recognition method based on YOLOX convolutional neural network. Their method improves the detection accuracy, compared to the YOLOX baseline.

Table 1 provides a systematic comparison of prior smoke detection approaches, highlighting their main contributions, strengths, and limitations. The table contrasts YOLO-based, SSD, and transformer-based methods, summarizing how each handles accuracy, efficiency, and deployment considerations. This overview clarifies the relative advantages of existing techniques and positions our work, which leverages YOLOv13-nano with multi-step pre-processing and rigorous cross-validation, as a more robust and efficient solution for real-time wildfire smoke detection.

Table 1. Summary of prior smoke detection approaches, highlighting their contributions, strengths, and limitations.

Approach	Contribution	Strengths	Limitations
YOLO-based (YOLOv4–v13) [4–6,10,11,17–19]	Lightweight adaptations, attention modules, backbone refinements, benchmarking on wildfire datasets.	High accuracy, real-time detection, efficient on edge devices, scalable improvements across versions.	Limited evaluation of multi-step pre-processing, some sensitivity to small or occluded smoke regions.
Single-Shot MultiBox Detector (SSD) [7,8,13,14]	Early single-stage detector with backbone variants (SqueezeNet, MobileNet) and custom SSD derivatives.	Simple architecture, efficient multi-scale detection, adaptable to real-time edge applications.	Lower accuracy compared to modern YOLO variants; less effective on thin or distant smoke.
Transformer-based (CCPE, Swin) [9]	Cross-contrast patch embedding with transformer backbones.	Strong feature representation, robustness in complex backgrounds.	Computationally heavier, reduced feasibility for real-time deployment on edge devices.
Hybrid/Pyramid networks [12,15,16]	Feature pyramid refinements and lightweight smoke-specific designs.	Enhanced small-object detection, parameter reduction with competitive accuracy.	Gains often task-specific, limited generalization across benchmarks.

1.4. Motivation and Contribution

Since models of object identification based on deep learning can provide timely warnings and improved situational awareness, the study underpinning the present work considers deep learning-based object identification models.
The main contributions are the following:

- A comparison of YOLOv9-tiny [20], YOLOv10-nano [21], YOLOv11-nano [22], YOLOv12-nano [23], and YOLOv13-nano [24,25] models trained and tested.
- Comparing the performances in terms of the measures mAP50, mAP50_90, F-1 Score, precision, recall and training time and inference time.
- Provide an assessment on the variants of YOLO that can be utilized in the real-time scenario of smoke detection over edge computing systems.

Among the key findings of this work it should be highlighted that we provide a benchmarking of YOLOv10–YOLOv13 nano models on wildfire smoke detection with a 5-fold cross-validation scheme. Additionally, we formalized detection-aware Cropping as a preprocessing technique, analyzing its impact across different Crop percentages. Moreover, it is shown that YOLOv13-nano, combined with Cropping and Histogram Equalization (HEQ), achieves the most balanced trade-off between accuracy and computational efficiency.

2. Theoretical Background

After outlining what deep learning detection frameworks are all about, under this section we establish a presence on YOLO algorithm and on the fact that it has become crucial in a real-time object identification system. Next, we discuss the basics behind the YOLO object detection algorithm and outline our approach on its deployment (especially of the nano variants of YOLO algorithm) for smoke detection in forest imagery over edge computing.

2.1. Real-Time Object Recognition

The idea behind the generic layout of the architecture of YOLO is exhibited as well as the unique group of signals within the identification model that manifests itself above and beyond conventional systems and delivers a new course of speed, precision and performance deployable characteristic. Research on this field should emphasize on this technology due to the need of the real-time object identification technology in the safety critical systems i.e., smoke detection. The YOLO model, the computational concept, and design principles that it involves to get used in research are discussed next.

As a single-stage object detection algorithm, YOLO approach makes object detection process resemble a simple regression. It is more popular because of its high precision and quick results because it can be used in real-time implementation [26,27]. YOLO compared to the two-stage approach of Faster R-CNN has real-time benefits because it analyzes the whole images only once to predict boxes, and class probabilities [26]. YOLO's main advantages include:

- Low latency and high speed, making it appropriate for real-time applications.
- End-to-end training, thus reducing complexity.
- Global reasoning of the image at a glance.

YOLO converts input pictures into a grid in order to draw predictions on boxes and per-class levels of confidence of every cell. YOLO has had several variants which have enhanced the computing speed, the drawing of an anchor box and backbone structure [28].

2.2. Edge Computing Using YOLO Algorithm

Smoke detection procedures require real-time object recognition since they are time sensitive. Considering that the nano and small output of YOLO presents the combination

of high accuracy in detection as well as quick processes and resource utilization, nano and small computer version of YOLO can be used in embedded systems to carry out real-time processes [29].

Edge computing facilitates the use of the surveillance cameras and sensor nodes nearer the resources of the computational intelligence. The given model is useful in the case of a smoke detection system as there is no trustworthy bandwidth one can utilize due to it being significantly quicker and conveying information more effectively. The small footprint of model and fast inference speed make the YOLO-nano and the YOLO-tiny a better architecture to employ at the edge. They are applicable with Raspberry Pi, Jetson Nano and other microcontroller units [30].

The main focus of this work is to determine the extent to which various versions of the YOLO object-detecting models would detect the smoke in the forest imagery. Through our investigation we focus on the process of model selection, training of a vast quantity of environments and follow-through examination with conventional methods including five-fold cross-validation and assessment criteria. We examined various pre-processing techniques comparing them to the baseline (i.e., no pre-processing techniques are deployed) in order to investigate which combination of pre-processing techniques and over which variant of YOLO exhibits the best of its potentials performing in rational time and achieving high detection rates.

Particularly, five variants of YOLO are investigated, focusing on the nano versions that can be fully utilized in the real-time detection of smoke (considering their operational costs on low configuration of the hardware and even work on the edge deployment). Regarding the baseline, the experiments were performed without pre-processing of the image. We mostly consider the Cropping and Histogram Equalization pre-processing techniques, while specifically for the case of Cropping, our approach is based on which region of image could be cropped in order to run the model again in that region. It is necessary to mention that larger Crop percentages preserve more image context, improving detection accuracy. A 90% Crop percentage will indicate that 90% of original image is still there to proceed with experiment in the second round and 40% would indicate that 40% of original image will be preserved to do experiment in the second round.

3. Methodology

In this section we discuss the application of the YOLO versions on smoke detection in forest imagery as also the methodology we applied to pinpoint the variant of YOLO that will be the optimal to be utilized in the real-time scenario of smoke detection over edge-computing systems throughout a comparative study outlining its performance over a set of pre-processing techniques.

3.1. Investigating the Use of YOLO Versions for Detecting Smoke in Images

Smoke is amorphous and semi-transparent, so it is quite difficult to identify (particularly when in a low contrast form or under noisy conditions). Because of interference of the background noise and the nature of the environment, smoke detection cannot make use of the nature of fi re, such as distinctive edges and color. Although its architecture has undergone modifications in its most recent variants and ensuring it can be used alongside training in the rightful manner, YOLO provides improved model outputs on harder classes since the detection of smoke prevention is of major importance in fi re control programs.

Five versions of YOLO model (YOLOv9-tiny, YOLOv10-nano, YOLOv11-nano, YOLOv12-nano, and YOLOv13-nano) were trained through standard training procedures on a given dataset and their performance was evaluated by a given system. The main focus of this work is to investigate how different configurations influenced the accuracy of the

classification, the speed of computation, and model behaviour, over a series of five-fold cross-validation experiments for different deployments of pre-processing techniques.

3.2. Investigating the Potentials of Cropping Pre-Processing Techniques

In this work, beyond the investigation of the effectiveness of the pre-processing technique, and particularly of the case of Cropping, we investigate how the percentage of Cropping in the imagery affects its detection confidence. In Figure 1 we observe the confidence vs Crop percentage of all models used in this research, while in Figure 2 it is provided an illustrative example on the Cropping of an image according to Crop percentage.

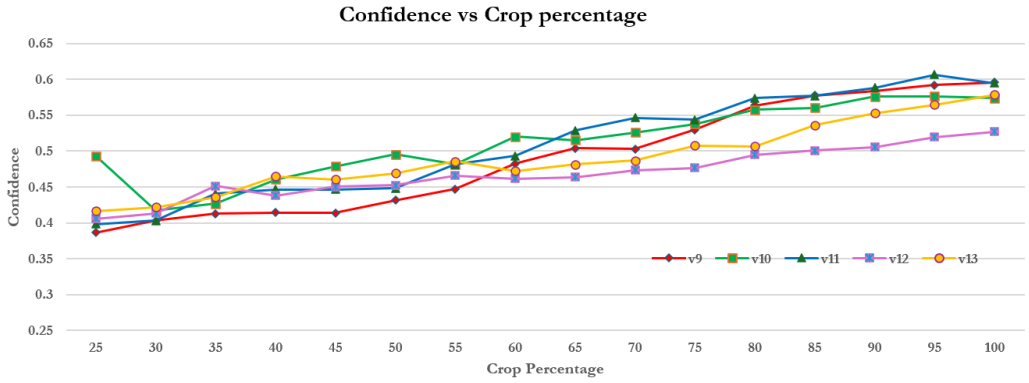


Figure 1. Comparison of confidence and Crop percentage for all YOLO models.

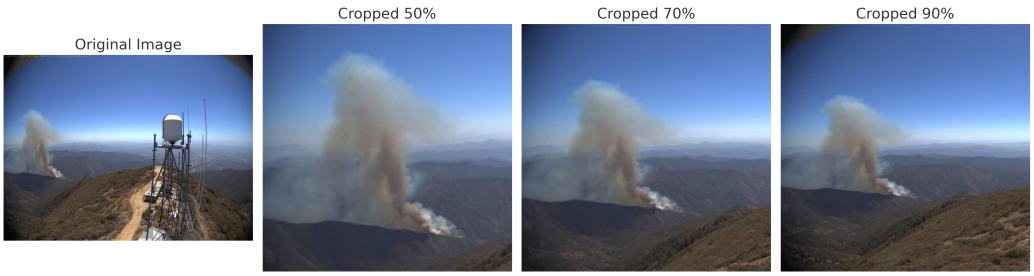


Figure 2. Example of image Crop.

YOLOv9-tiny sets an increasing trend in regards to Crop percentages that vary between 25% and 100%. At small Crops and reaching nearly 0.60 at full-image inference it starts at around 0.39. It assumes that, however much better the degree of focus can be achieved with the help of detection-aware Cropping, the ability to locate high confidence will not be applicable where the Crop lacks contextual information provided by YOLOv9-tiny. The advantage of this confidence notably increases beyond 55% Crop size and the argument behind this is that the larger the Crop, the more contextual information remains even in it to permit the small model to be applied more perfectly aligned with smoke.

YOLOv10-nano has a modest decrease, about 30% of the Crop, and then slowly helps itself along, and the confidence increases with the size of the Crop. The level is 25% confidence is 0.49 and 30% is 0.42 and gradually increases to 0.57 at the maximum size. Such a decrease at the outset means that feature appreciation during this model can be poor when Crops are so small. Nevertheless, the smooth recovery since 35% proves that Cropping the image can be employed as leverage in favor of YOLOv10-nano provided that the ratio of the cropped portion is preserved to be large; i.e., the proportion of the relevant features is kept to be high as well.

YOLOv11-nano is starting with confidence level (circ. 0.40 at 25–30%) and has a more pronounced positive gradient commencing at the level of 35%. It steadily rises, upon referring to an average of above 0.53 on an average at 65% Crop percentage and

above 0.61 on average at 95%. Greater effect of confidence enhancement over its prior nano variants suggests that the latest YOLOv11-nano will be more susceptible to context deprivation tight Crops, but it would be reasonable to consider more plentiful cues when viewing more of the scene in the field of view.

YOLOv12-nano shows steep and constant rise after attaining not very high initial confidence (0.41–0.45 between 25% and 40%) then gaining stable growth after 45%. The rate of growth acquires momentum after exceeding 80% and at 95% growth rate is about 0.52. YOLOv12-nano seems to offer more resistance to smaller Crops than the predecessors, so feature extraction seems less susceptible as input sizes arrive at mid-level sizes, and generates more value by keeping more of the scene.

YOLOv13-nano portrays a relatively linear and steady growth of around 0.42 up to around 25–30%, a sharper growth past 40%, and an evened specifications of around 0.58 when the Crop grow to full-size. It has some slight troughs at about 45–60% which is loss of context due to being cropped here and there but by having larger Crop windows there is even progression in confidence. The fact that this curve can be repeated means that at the same time, YOLOv13-nano is more consistent over different Crop percentages and seems to be the optimal in the scenario where most of the initial image is to be lost.

It is worth noting that the differences that have occurred in later nano variants (YOLOv11-13) are more likely to be gentle, small gradual in comparison to the differences that occurred earlier (YOLOv9-nano, YOLOv10-nano) that are more erratic, and sensitive to small Crops. The unified procedure that implements the Cropping selection includes three steps, namely, the training of the models, the selection of the optimal Crop percentage and its evaluation on the Cropping dataset, as listed below:

1. Train the Models
    - For each candidate YOLO architecture (YOLOv9, YOLOv10, ...), run train.py on all folds of your dataset.
    - Save the trained weights (best.pt) per fold.
2. Determine Optimal Crop Percentage
    - For each trained model checkpoint:
        - Run find optimal Crop on validation images.
        - Loop over Crop scales (e.g., $0.25 \rightarrow 1.0$).
        - For each detection, Crop around it, re-run inference, and log detection confidences.
        - Compute the average confidence per scale for all images.
    - Select the Crop scale with the highest mean confidence.
    - Store the chosen Crop percentage for each model architecture.
3. Evaluate Cropped Dataset
    - Use the optimal Crop percentage per model as input to the evaluation of the Cropped dataset routine.
        - Crop each test image around detections.
        - Rescale ground truth labels.
        - Evaluate detection metrics (precision, recall, mAP).
    - Save results with metrics and Crop percentage for comparison.

Algorithm 1 integrates the three stages of the pipeline into a single structured process. First, each YOLO architecture is trained separately on dataset folds through cross-validation to ensure robust performance and adaptation to the data, with the trained weights from each fold stored for later use. Next, the algorithm performs an optimal Crop search using

a detection-aware strategy by exploring multiple Crop scales ranging from 25% to 100% of the image size. For each validation image, detections are obtained from the model, Crops of the specified scale are extracted around each detection, and these cropped regions are re-evaluated by the model to compute the average detection confidence at each scale. The Crop scale that yields the highest average confidence is selected as the optimal Crop percentage for that model. Using this optimal Crop percentage, all validation images are then cropped and resized, and the trained model is evaluated on the modified dataset. Standard detection metrics such as precision, recall, F1-score, and mean average precision (mAP) are computed to confirm that the chosen Crop scale not only maximizes confidence but also leads to strong performance across established benchmarks. Finally, for each YOLO architecture, the optimal Crop percentage along with the corresponding evaluation metrics are stored in a summary table, enabling direct comparison of algorithms, Crop scales, and overall performance.

Algorithm 1: Optimal Crop Selection for Smoke Detection Models

Input: Dataset folds $\mathcal{D}$, YOLO architectures $\mathcal{A}$, Crop scales $S = \{0.25, \ldots, 1.0\}$
Output: Optimal Crop percentage and evaluation metrics for each model

foreach *architecture* $a \in \mathcal{A}$ **do**
  foreach *fold* $f \in \mathcal{D}$ **do**
    Train YOLO model $M_{a,f}$ on fold f;
    Save trained weights $W_{a,f}$;
  Initialize conf_avg$[s] \leftarrow 0$ for all $s \in S$;
  foreach *scale* $s \in S$ **do**
    foreach *validation image I in held-out fold* **do**
      Run inference with $M_{a,f}$ on I and obtain detections $\{d\}$;
      foreach *detection* $d \in \{d\}$ **do**
        Crop I around d with scale s to get I_s;
        Run inference with $M_{a,f}$ on I_s and obtain confidence c;
        Accumulate c into conf_avg$[s]$;
  Select $s^* \leftarrow \arg\max_{s \in S}$ conf_avg$[s]$;
  foreach *fold* $f \in \mathcal{D}$ **do**
    Apply Cropping with percentage s^* to dataset of fold f;
    Evaluate $M_{a,f}$ on cropped dataset and compute metrics (P, R, F_1, mAP);
  Save results (a, s^*, P, R, F_1, mAP);
return *Summary table of architectures, optimal Crop percentages, and metrics*

3.3. Investigating the Capabilities of YOLO Variants over Different Techniques

The study was focused on model performance and measured which version of YOLO performed better over 100 epochs of training. The condition of various training durations was applied in the experiment to determine whether long training improves accuracy. The hyperparameter settings were identified in the same way to facilitate the assessment of results using all training versions. The factors (hyperparameters) were used in the form: data = fold_yaml, epochs = 100, imgsz = 640, device = [0, 1, 2], batch = 39, where fold_yaml is the different yaml folder used for the different folds, 100 epochs was a decent number for training, 640 image size ensured that no image information was lost, device = [0, 1, 2] indicates that 3 GPUs were used during training and batch value of 39 was set in order to prevent the GPUs from reaching their limits.

The chosen hyperparameter settings were based on a balance of prior defaults, computational feasibility, and empirical validation runs. A summary of the selected values and their justifications is provided in Table 2. Next, we provide a brief discussion on the most critical ones:

– The choice of 100 epochs follows prior practice in YOLO-based studies on small-sized or medium-sized datasets, which report that this range is sufficient for stable convergence without overfitting. Larger epoch counts led to only marginal gains in pilot tests while significantly increasing training time.
– The image resolution of 640 pixels was adopted as it represents the default setting in YOLO literature, providing a balance between capturing fine detail and keeping inference cost manageable.
– The batch size of 39 was selected based on the available GPU memory: it allowed efficient utilization of the three GPUs while avoiding memory overflow, and preliminary trials with higher values showed no performance benefit.

Overall, these choices reflect a compromise between computational feasibility, literature standards, and empirical evidence from pilot experiments, ensuring that the models converged robustly under realistic resource constraints.

Table 2. Summary of hyperparameter choices and their justification.

Hyperparameter	Justification
Epochs = 100	Sufficient for stable convergence on small datasets; longer runs showed diminishing returns while increasing training time.
Image size = 640	Default in YOLO literature; balances fine detail preservation with manageable computational cost.
Batch size = 39	Optimized for available GPU memory (3 GPUs); maximizes throughput without memory overflow.
Device = [0, 1, 2]	Parallelized training across 3 GPUs for faster convergence and efficient resource use.
Fold_yaml	Enables five-fold cross-validation, reducing variance and improving generalization.

The study carried out a five-fold cross-validation, which enhanced the dependability and usefulness of the findings. In every round, one compartment (fold) was utilized in the inference and four compartments (folds) were utilized in training, while the entire dataset was split into five equal compartments (folds). By employing relatively small and uniform datasets, the method reduces variance in the dataset as well as bias in evaluation performance [31]. In this paper, we utilized a Smoke Labelling Dataset from Roboflow which features 737 images of smoke labelled with YOLO annotation format [32]. Every image is accompanied by label files that contain the bounding box coordinates along with class tags. Evaluation of models took place through standard object detection metrics:

- **Precision** measures the proportion of correctly predicted positive instances out of all predicted positive samples.

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{1}$$

where TP represents true positives and FP represents false positives.

- **Recall** (or sensitivity) measures the proportion of correctly predicted positive instances out of all actual positive samples.

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{2}$$

where FN represents false negative samples.

- **F1-Score** is the harmonic mean of precision and recall. It balances the two metrics, especially when there is an uneven class distribution.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

- **Mean Average Precision (mAP)** is used mainly in object detection. It calculates the average precision (AP) for each class and then computes the mean across all classes. It gives a comprehensive measure of both precision and recall across multiple thresholds.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i, \tag{4}$$

where N is the number of classes and AP_i is the average precision for class i. This metric can be deployed also by considering the IoU values (intersection-over-union) as mAP50 and mAP50_95.

- **Training Time and Testing Time** assess the computational efficiency of each version [28,30].

Multi-metric assessment provides a complete view of system performance and deployability because edge computing demands rapid accurate operations [29,30].

3.4. Edge-Device Deployment and Lightweight Architecture Synthesis

Real-time wildfire smoke detection often requires deployment on resource-constrained edge devices, such as NVIDIA Jetson Nano, Raspberry Pi, or other low-power embedded systems. Standard YOLO architectures, while highly accurate, are computationally demanding and memory-intensive, which can limit their applicability in these environments. To address this, our methodology focuses on lightweight YOLO variants (YOLOv9-tiny to YOLOv13-nano) and explores strategies for architecture synthesis that optimize performance for edge computing. The key considerations for deploying YOLO models on edge devices include, namely, the model pruning, the model quantization, the knowledge distillation and the edge-aware pre-processing, as listed below:

- Model Pruning: Reduces the number of neurons, channels, or layers by removing redundant parameters, minimizing memory requirements without a significant loss of detection accuracy.
- Quantization: Converts model weights from 32-bit floating point to lower-precision representations (e.g., 16-bit or 8-bit integers), which decreases model size and accelerates inference on CPUs and GPUs typical of edge devices.
- Knowledge Distillation: Transfers knowledge from a larger "teacher" model to a smaller "student" model, allowing the compact model to maintain high accuracy while improving efficiency.
- Edge-Aware Pre-processing: Applies techniques such as Histogram Equalization and detection-aware Cropping before inference, reducing input complexity and improving the extraction of relevant features in noisy or low-contrast wildfire imagery.

We integrate these optimization strategies into our comparative analysis of YOLO variants. After training each model through cross-validation and selecting optimal pre-processing parameters (Sections 3.1 and 3.2), we assess deployment readiness by simulating inference on typical edge-device constraints. This includes evaluating the following factors:

- Memory Footprint: Amount of RAM required to load and run the model.
- Inference Speed: Frames per second (FPS) achievable under edge-device hardware constraints.
- Detection Accuracy: Ensuring that optimizations such as pruning and quantization do not degrade performance on smoke detection metrics (precision, recall, F1-score, and mAP).

By systematically combining lightweight architecture selection, compression techniques, and edge-aware pre-processing, our approach provides a practical pathway for deploying real-time wildfire smoke detection systems on low-power devices. This extension ensures that our methodology not only evaluates detection performance but also addresses practical deployment considerations, fulfilling the edge-computing focus of the paper.

4. Evaluation

In this section we provide an experimental comparison of the selected models of YOLO aiming on the detection of the presence of smoke in forest imagery, assessing their performance over different pre-processing techniques that focus on distinguishing the optimal selection for edge computing.

4.1. Experimental Design

For the conduction of the experimental evaluation, we utilized the dataset provided in [32,33], containing 737 images of smoke, acquired from ground cameras, located in hills or mountains. The models were evaluated using the five-fold cross-validation method. The images refer to landscapes of exterior space mostly in hills and mountains in period of time where the daylight was available and the weather was clear with mostly sunshine indicating epochs of the year where the wildfires are more probable to occur.

The series of evaluation experiments were conducted on a workstation equipped with three NVIDIA RTX A5000 GPUs, NVIDIA, Santa Clara, USA, each with 24,564 MiB of dedicated memory. During the training phase, all three GPUs were utilized concurrently in order to accelerate convergence and handle larger batch sizes, whereas the inference phase was carried out using a single GPU to simulate realistic deployment conditions. GPU parallelization was achieved through the CUDA library (version 12.8), which enabled efficient multi-device allocation and synchronization. The software environment was based on Python 3.10.12, with model implementation and training pipelines managed through the Ultralytics framework (version 8.3.63), ensuring reproducibility and compatibility with the most recent YOLO architectures. In addition to the hardware and software setup, careful optimization of training hyperparameters was applied to maximize resource utilization while maintaining experimental consistency across folds. The complete project, including the pre-processing scripts, training configurations, and labeled dataset, has been made publicly available in [32], allowing for transparency, reproducibility, and potential extensions by the research community.

In Figure 3 it is depicted an example of a validation batch over the deployment of different pre-processing techniques across different YOLO variants. In the first row there are depicted the confidence values when deploying no pre-processing technique (i.e., baseline),in the second row there are depicted the corresponding confidence values applying only the Crop pre-processing technique, in the third row there are depicted the corresponding confidence values applying only the HEQ pre-processing technique, while

in the fourth row there are depicted the corresponding confidence values applying both the Crop and HEQ pre-processing techniques.

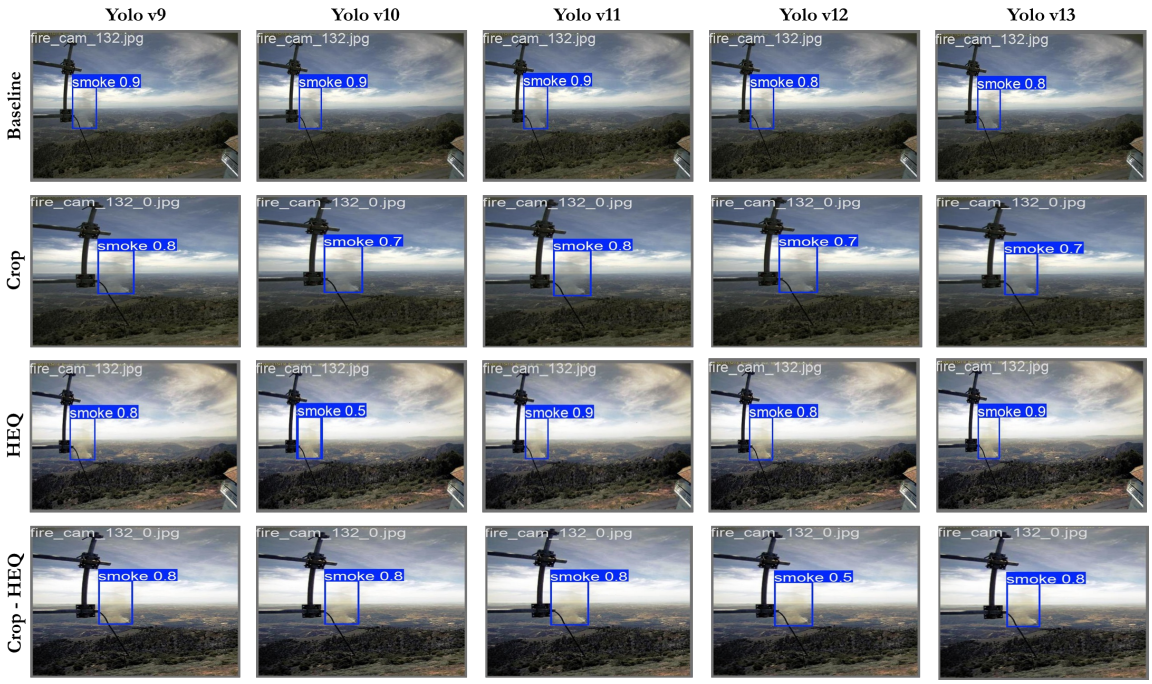


Figure 3. Confidence values on validation batch over the YOLO variants when applying no pre-processing (baseline), the Crop pre-processing technique, the HEQ pre-processing technique, and a combination of these techniques (Crop-HEQ) pre-processing technique.

As we can observe from the batch images presented in Figure 3 the comparison of different YOLO variants across multiple pre-processing techniques reveals several interesting trends in wildfire smoke detection. Under the baseline condition, all YOLO models perform strongly, with confidence levels ranging between 0.8 and 0.9. YOLOv9, v10, and v11 consistently reach 0.9, while v12 and v13 provide slightly lower values at 0.8, suggesting that later versions may adopt stricter detection thresholds. When Cropping is applied as a pre-processing step, the confidence scores decline in this case. YOLOv9 retains a relatively strong score of 0.8, but the later versions, from v10 through v13, drop to values between 0.7 and 0.8. This reduction indicates that removing contextual information around the smoke regions can hinder the models' certainty. Histogram Equalization (HEQ), on the other hand, stabilizes and often improves performance compared to Cropping. Confidence levels range between 0.8 and 0.9 across all versions, with YOLOv12 and v13 achieving 0.9, highlighting that contrast enhancement allows the models to better distinguish smoke from the background. The combined pre-processing of Cropping and HEQ shows mixed results. While YOLOv9, v10, and v13 remain stable at around 0.8, YOLOv12 drops significantly to 0.5, indicating that over-processing can distort the features required for robust detection. Overall, in this case it is demonstrated that baseline and HEQ pre-processing are the most reliable strategies, as they maintain consistently high confidence across models. Cropping alone reduces detection certainty by eliminating contextual cues, and the combination of Crop with HEQ introduces variability that can sometimes be detrimental. Among the models, YOLOv9 and v13 emerge as the most robust, while YOLOv12 appears particularly sensitive to pre-processing choices. This qualitative ablation suggests that simple or moderate pre-processing, particularly Histogram Equalization, is more beneficial than aggressive transformations when aiming for stable and confident smoke detection.

The methodology pipeline followed for the conduction of the evaluation experiments of this study involves the dataset acquisition that include the wildfire smoke images, the performance of the pre-processing techniques investigated in this study, the conduction the five-fold cross-validation experiments including the tuning of the hyper-parameters, the metrics interpretation, and concludes to the decision of the most effective combination after the analysis of the exhibited results. An overview of the methodology pipeline is depicted in Figure 4.

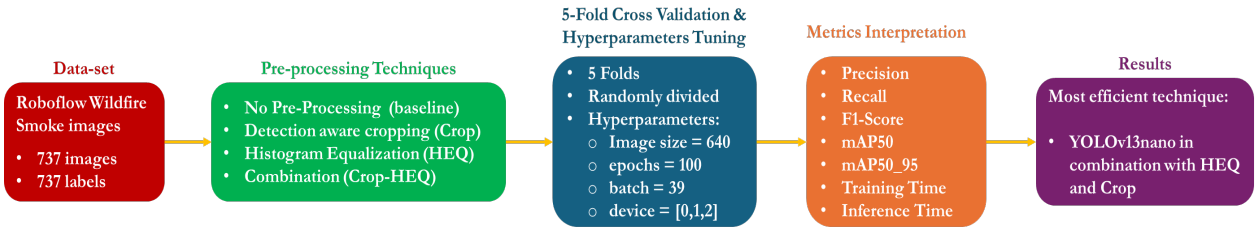
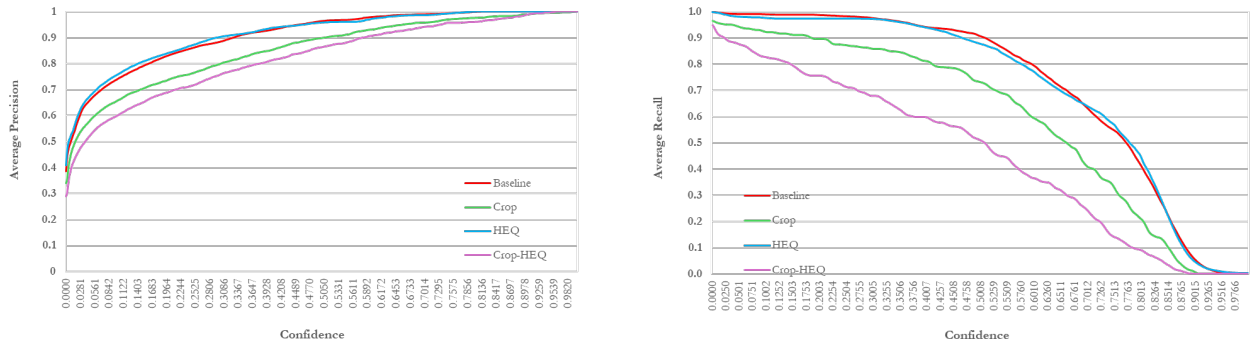


Figure 4. Evaluation methodology pipeline diagram.

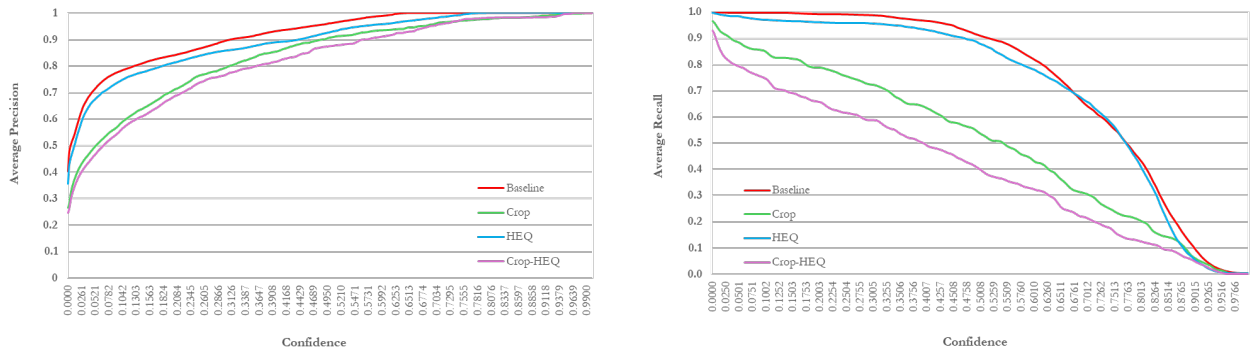
4.2. Experimental Results

In Figures 5–9 we present the precision and recall curves against the exhibited confidence for the investigated YOLO variants (V9 to V13), respectively, deploying the baseline (no pre-processing techniques) the Crop pre-processing technique, the HEQ pre-processing technique, as also their combination Crop-HEQ pre-processing technique. In Figures 5–9 the x-axis represents the confidence while the y-axis represents the Average Precision (a) and the Average Recall (b).



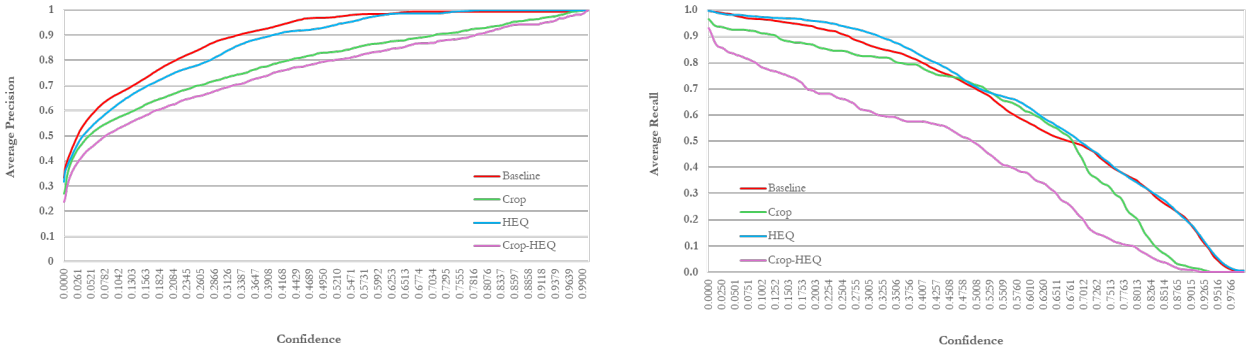
(a) (b)

Figure 5. Average Precision (**a**) and Average Recall (**b**) curves exhibited for YOLOv9-tiny using the baseline technique, and the Crop, HEQ, and Crop-HEQ pre-processing techniques.



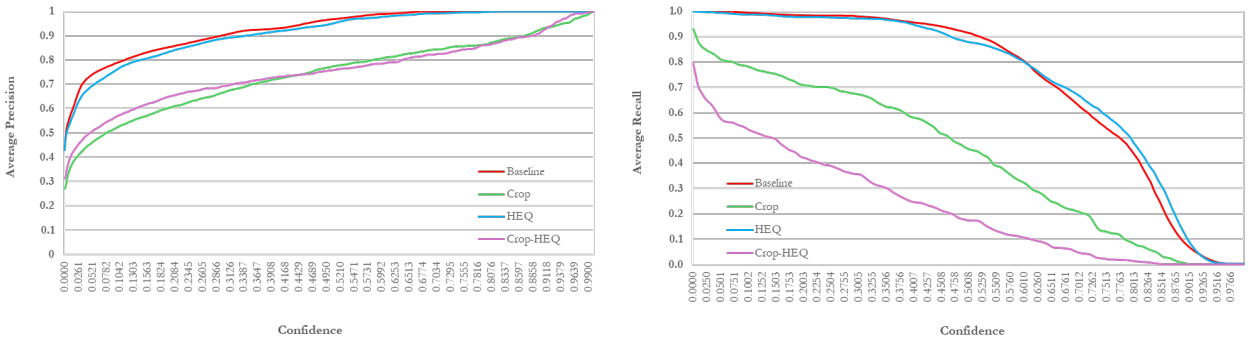
(a) (b)

Figure 6. Average Precision (**a**) and Average Recall (**b**) curves exhibited for YOLOv10-nano using the baseline technique, and the Crop, HEQ, and Crop-HEQ pre-processing techniques.



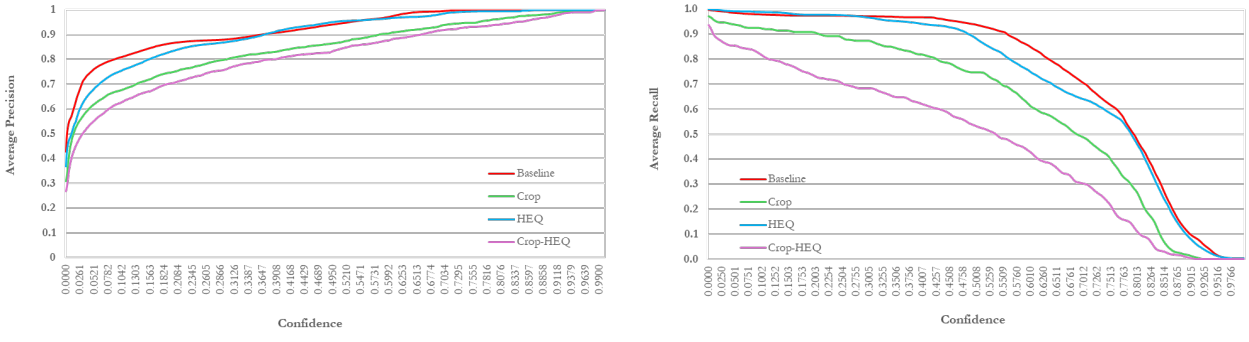
(a) **(b)**

Figure 7. Average Precision (**a**) and Average Recall (**b**) curves exhibited for YOLOv11-nano using the baseline technique, and the Crop, HEQ, and Crop-HEQ pre-processing techniques.



(a) **(b)**

Figure 8. Average Precision (**a**) and Average Recall (**b**) curves exhibited for YOLOv12-nano using the baseline technique, and the Crop, HEQ, and Crop-HEQ pre-processing techniques.



(a) **(b)**

Figure 9. Average Precision (**a**) and Average Recall (**b**) curves exhibited for YOLOv13-nano using the baseline technique, and the Crop, HEQ, and Crop-HEQ pre-processing techniques.

Next, the comparative results are presented in a sequence of bar-charts depicting the metric utilized, namely, Precision, Recall, F1-Score, mAP50 and and mAP50_95. In each plot (see, Figures 10–13) the x-axis corresponds to the metrics over a block of the exhibited values correspond to each YOLO variant, while the y-axis corresponds to the exhibited value of each metrics for each YOLO variant. In all the results depicted in Figures 10–13, the exhibited values in y-axis for each metric are averaged over the five folds after the five-fold cross validation series of experiments.

In Figure 10 we observe the evaluation results of all algorithms without using any pre-processing techniques. The evaluation metrics show that version v13 is at the top in all

the metrics followed by others and the best values are: Precision (0.8354), Recall (0.8024), F1-Score (0.8176), mAP50 (0.865) and mAP50_95 (0.4868). Comparatively, v10 performs the least, having had the lowest scores among all the measurements- Precision (0.7262), Recall (0.6498), F1-Score (0.6856), mAP50 (0.726) and mAP50_95 (0.3938). The evident direction indicates that v13 is the most consistent and efficient one among all versions that are tested, and the v10 version considerably lags behind in accuracy of detection and the overall quality of the model. The performance of the five YOLOvX-nano models (YOLOv9 to YOLOv13) with no pre-processing used on the five YOLOvX-nano models is the first given in a bar chart. In every metric (Precision, Recall, F1-score, mAP50, and mAP50_95) YOLOv13 and YOLOv9 are already ahead and the differences increase in every value except Recall and mAP50 values where YOLOv13 shows the best values. YOLOv10 is the poorest both generally and specifically with respect to mAP50_95, or it can be stated that it performs more poorly when it comes to adapting to different IoUs of mAP50_95. This graph worked to highlight how there was no correlation as to the degree of accuracy in detection of these two versions and this provides a point of reference as to the performance that this group of pre-processing methods can be judged against.

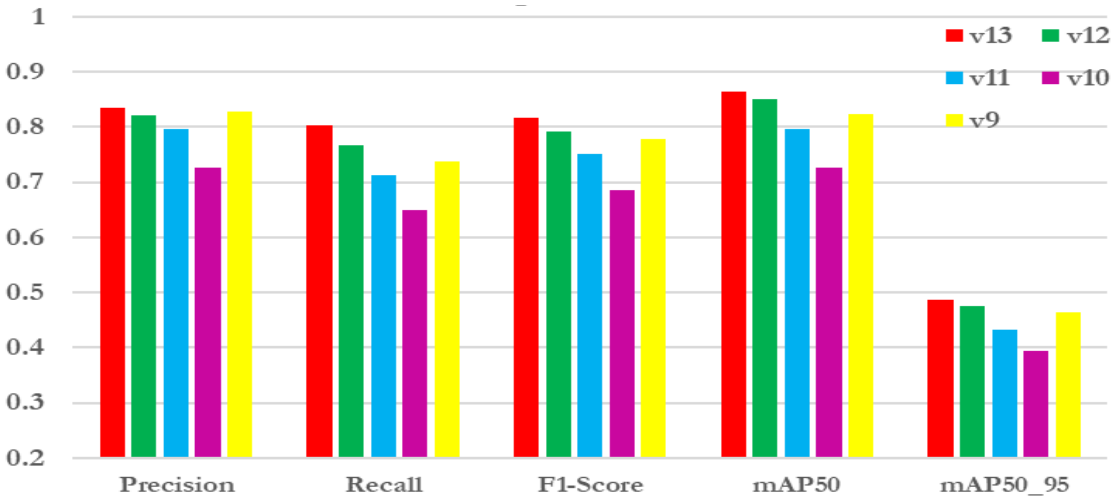


Figure 10. Comparison of all models without using pre-processing techniques.

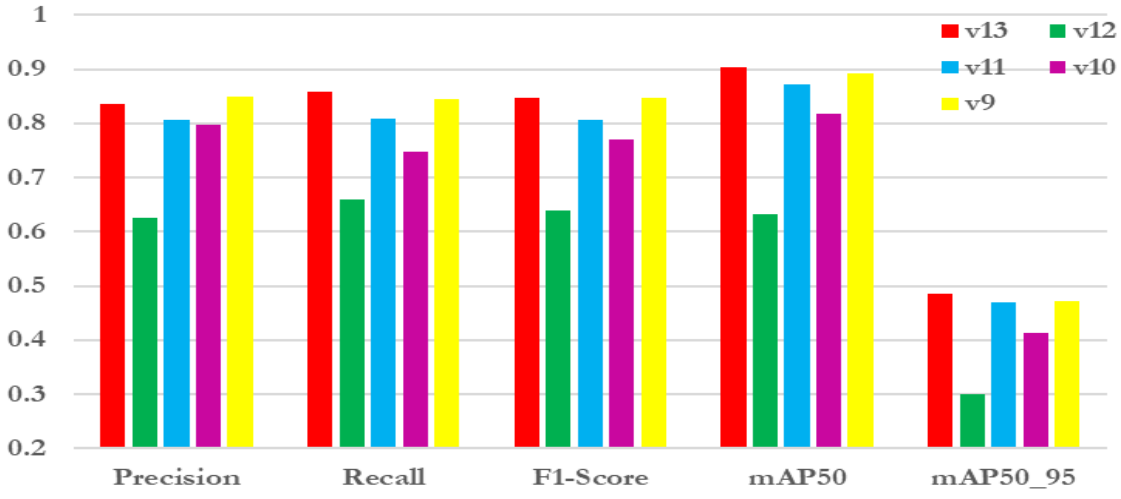


Figure 11. Comparison of all models using Cropping pre-processing technique.

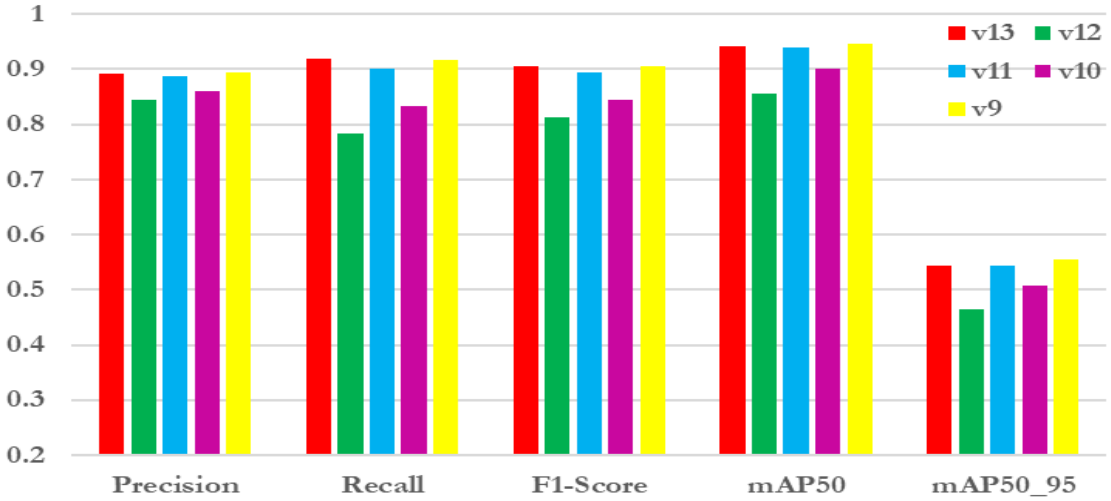


Figure 12. Comparison of all models using Histogram Equalization pre-processing technique.

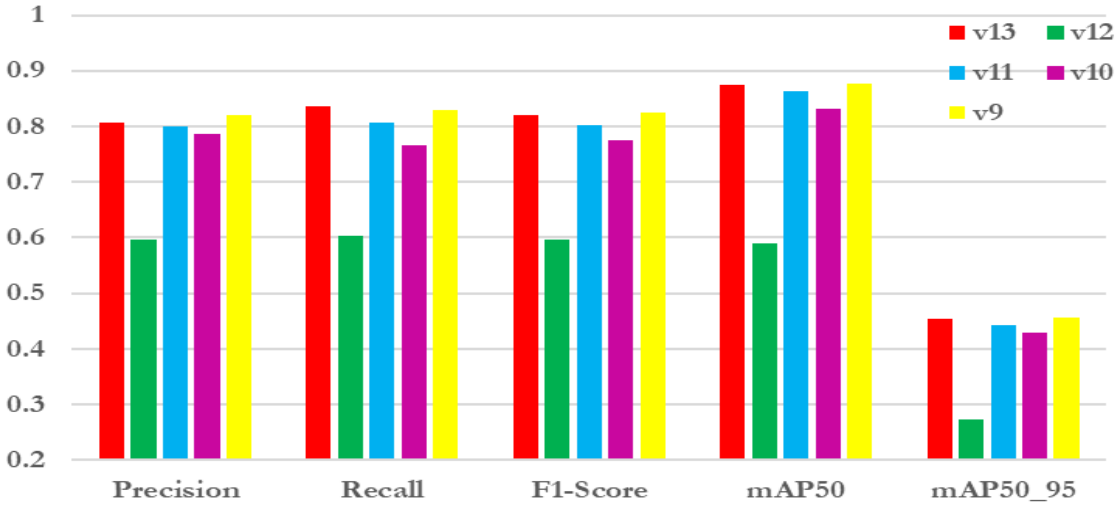


Figure 13. Comparison of combination of pre-processing techniques Cropping and Histogram Equalization of all models.

Recent advances in wildfire smoke detection have leveraged both convolutional and transformer-inspired deep learning models. Early approaches relied on hand-crafted visual features and segmentation [3], followed by CNN-based models such as enhanced YOLOv4 [4], which offered measurable improvements in detection performance. Lightweight YOLO variants (e.g., YOLOv9-tiny, YOLOv10-nano, YOLOv11-nano) have enabled real-time wildfire detection with high precision [5,6,10,11]. In contrast, Single Shot MultiBox Detector (SSD) models [7,8] provide a simple single-stage architecture for bounding-box regression and class prediction. SSD is computationally efficient and effective for multi-scale smoke detection; however, it tends to show lower precision for small or partially occluded smoke regions and often lacks rigorous evaluation on large, diverse datasets. Transformer-based methods such as Cross Contrast Patch Embedding (CCPE) [9] address complex backgrounds by enhancing feature representation, but they require heavier computation and are less suited for deployment on edge devices. Our study extends prior work in two key ways: we employ YOLOv13-nano to benefit from its improved feature extraction and lightweight design, and we incorporate multi-step pre-processing (e.g., HEQ, Cropping) alongside five-fold cross-validation, which improves robustness and generalization across wildfire scenarios. This approach addresses

limitations in both SSD and YOLO-based prior studies, while maintaining efficiency suitable for real-time deployment. The comparative strengths and limitations of these methods are summarized in Table 1.

In Figure 11 we have the results of Cropping pre-processing technique. Once again, in this Cropped dataset test, v13 stands out with highest ranking in some important measures; F1-Score (0.8466, tied with v9), mAP50 (0.9034) and mAP50_95 (0.486) with a strong Precision (0.8366) and Recall (0.8594). v9 is slightly better in Precision (0.85) but performs well in all other measures. Conversely, v12 is the worst performer with the lowest Precision (0.6244), Recall (0.66), F1-Score (0.6394), mAP50 (0.6322), and mAP50_95 (0.2998), meaning that they have poor quality of detection. Generally, v13 has the best performance albeit at the increased cost of computation compared with v12 that has the worst performance. The Cropping procedure being calibrated in a manner attractive to the high-confidence detections makes the response stable in majority of the models. It is also interesting to add that in models YOLOv13, YOLOv11 and YOLOv9, it can be seen that F1 and mAP scores tend to be better which once again can serve as a strong point that the possibility of partially avoiding effects of irrelevant background noise by paying some attention to the central area of images where the features of relevance are more likely to appear, can be partially confirmed. YOLOv12 again appears to be less open to such pre-processing and this may be attributed to the architectural design of its feature extraction chain. First of all, the advantage of Cropping could be observed as the improved mAP50 rates in all areas.

In Figure 12 we observe the evaluation results of Histogram Equalization (HEQ) pre-processing technique. Regarding the deployment of the HEQ pre-processing technique, v9 has the overall best performance as its scores achieve the highest results on Precision (0.8946), F1-Score (0.9058), mAP50 (0.946), and mAP50_95 (0.555) with great Recall (0.917) which is close to v13 (0.918). v13 also matches v9 in all metrics with a strong Recall (0.917) that is only slightly lesser than v9. v11 is also very strong in many On the other hand, v12 will show the worst results, having the lowest values in all critical metrics: Recall (0.7842), F1-Score (0.8126), mAP50 (0.855), and mAP50_95 (0.4638). Across the board, the v9 is the most balanced in terms of precision and detection quality whereas the v12 is much lower. HEQ yields significant increases in most of the evaluation metrics in the majority of the YOLOs. YOLOv13 again is on the first place, with the best results in both mAP50 and mAP50_95 very close to that of the YOLOv11 and YOLOv9. Having found the overall improvement (especially in the recall and mAP measures), one might assume that due to the enhancement of the contrast of the wildfire-related features by HEQ, the models are also able to identify the areas of interest with better confidence. YOLOv12, however, has a less significant evolvement, which means that not every architecture is equally helped by HEQ.

In Figure 13 we observe the combination of pre-processing techniques Cropping and Histogram Equalization. Regarding the deployment of the two pre-processing techniques, v9 demonstrates the most significant overall results, ranking in the first place by all of five metrics: Precision (0.8212), Recall (0.8292), F1-Score (0.8252), mAP50 (0.8778), and mAP50_95 (0.4552). Being closely followed by v13 that also scores well in all of the set metrics, mostly by bouncing off a high score in the Recall (0.837) and mAP50 (0.8736) compared to v12 which scored the lowest in all the set measurements, registering the lowest value in Precision (0.5968), Recall (0.6038), F1-Score (0.5954), mAP50 (0.5908), and mAP50_95 (0.2742) That makes v9 the best in terms of quality and balance of detection over v12, which really suffers a lot when compared to the rest. The combination of Cropping and Histogram Equalization leads to performance improvement comparable to performance improvement with either Cropping or Histogram Equalization with minor synergetic effects in case of YOLOv13 and YOLOv9. YOLOv13 consistently demonstrated

the best F1 and mAP50 metrics, which affirms the effectiveness of the model regarding the variants of pre-processing. In the meantime, YOLOv12 suffers poor performance because of its sensitivity to a multi-step pre-processing. On the whole, the combined option is effective in terms of quality detection and increasing spatial focus along with contrast.

In Figure 14 the training time is depicted. The comparison of time in the training shows that there is a casual remark in the various pre-processing pipelines. The pre-processing combinations like Cropping, Histogram Equalization especially when applied alone have good improvement in performance with moderate training period. It proposes a beneficial trade-off of computational efficiency and precision of such method. Regarding the duration of the training time v13 is always the longest in almost every configuration. On the other hand, v11 shows the best training time, especially with Crop and further exhibits performance efficiency with one of its settings.

In Figure 15 the testing time is presented. The results of the test time are similar across the different methods of pre-processing with a minor upshot being observed in case of the cropped images most likely due to overheads of resizing or padding. Surprisingly, there is not a lot of difference in the duration of testing with various versions of YOLO, and this implies that most of the complexity on the pre-processing is mainly recuperated at a training step. This compounds the already potent power of detection-aware Cropping and HEQ because neither of the two are particularly heavy in terms of inference performance requirements on edge devices and real-time systems. Regarding the duration of the testing time, v13 is a bit slower compared with other variants with maximum of 4.226 s with Crop and 4.146 s with Crop and HEQ. In the meantime v11 excels as far as efficiency is concerned also having the lowest test times in general, though v10 is not far behind.

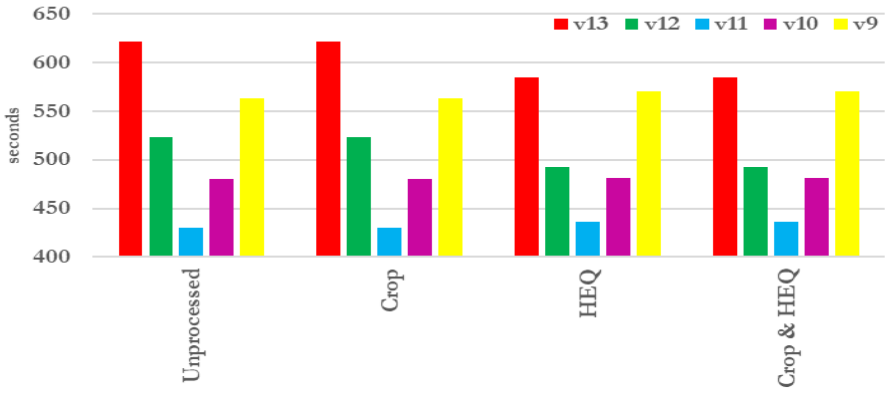


Figure 14. Comparison of training time (in seconds) of all models in all pre-processing techniques.

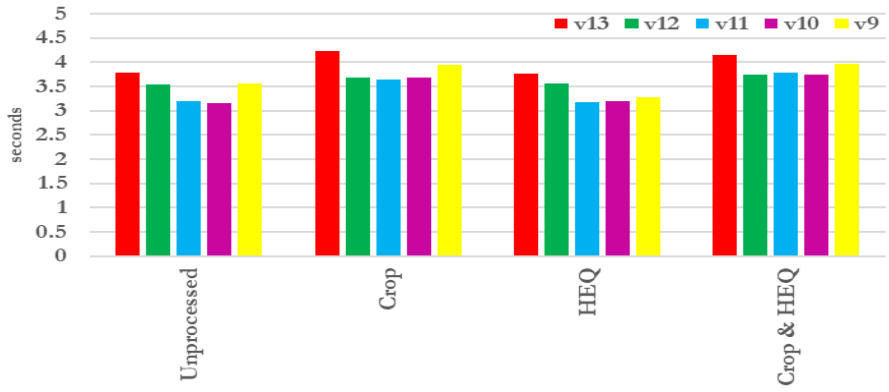


Figure 15. Comparison of test/inference time (in seconds) of all models in all pre-processing techniques.

4.3. Observations on the Exhibited Results

Experimental testing will provide legitimate inputs that indeed, we can get more improvements in the output of lightweight object detection models applied to identify wildfire by using some pre-processing procedures. Notably, Histogram Equalization (HEQ) and detection-aware Cropping were all the time increasing values of the metrics such as precision, recall, and mAP50 over the various models, namely YOLOv13-nano and YOLOv9-nano. It is likely that these improvements can be attributed to greater visibility of features and attention to space, respectively, that causes their generation of a greater detection capacity in complex visual scenes. Additionally, YOLOv13-nano also demonstrated an improvement as compared to the preceding versions in all the pre-processing options, which signifies the improvements in the framework design of the backbone and detection head. It is worth noting that these approaches did not come at much of a cost in the space of test-time overheads that pre-processing requires, and as such, real-time usages centered around such methods can be realised. The fact that even simple, but domain specific, processing such as HEQ and Cropping can be the best approach, despite being very specific to the detection tasks in question, as well as the importance of aligning the pre-processing techniques to both the model and the domain is a result of the latter two findings.

The results depicted in Table 3 show the mean SD for each metric reveal clear differences in how pre-processing techniques affect the performance of various YOLO architectures for smoke detection.

Table 3. Performance (mean-SD for each metric) of YOLO variants under different pre-processing techniques for smoke detection.

Pre-Processing	Model	Precision	Recall	F1-score	mAP50	mAP50_95
Baseline	YOLOv9	0.0287	0.0805	0.0463	0.0615	0.0354
	YOLOv10	0.0457	0.0579	0.0472	0.0628	0.0419
	YOLOv11	0.0406	0.0693	0.0537	0.0671	0.0339
	YOLOv12	0.0612	0.0713	0.0643	0.0558	0.0495
	YOLOv13	0.0315	0.0413	0.0209	0.0292	0.0136
HEQ	YOLOv9	0.0408	0.0502	0.0450	0.0400	0.0558
	YOLOv10	0.0558	0.0595	0.0531	0.0482	0.0455
	YOLOv11	0.0572	0.0601	0.0553	0.0452	0.0595
	YOLOv12	0.1428	0.1019	0.1166	0.1430	0.1057
	YOLOv13	0.0668	0.0489	0.0512	0.0489	0.0593
Crop	YOLOv9	0.0254	0.0412	0.0302	0.0330	0.0266
	YOLOv10	0.0468	0.0570	0.0488	0.0510	0.0221
	YOLOv11	0.0373	0.0594	0.0429	0.0370	0.0363
	YOLOv12	0.1012	0.0728	0.0750	0.0783	0.0345
	YOLOv13	0.0487	0.0429	0.0313	0.0420	0.0388
Crop + HEQ	YOLOv9	0.0097	0.0202	0.0132	0.0205	0.0206
	YOLOv10	0.0520	0.0643	0.0552	0.0428	0.0212
	YOLOv11	0.0451	0.0455	0.0362	0.0281	0.0355
	YOLOv12	0.0817	0.0475	0.0316	0.0446	0.0297
	YOLOv13	0.0660	0.0308	0.0356	0.0470	0.0454

The baseline results show relatively low precision, recall, and mAP across all models, with YOLOv12 slightly outperforming others in terms of F1-score (0.0643) and mAP50_95 (0.0495). This indicates that while the models capture some smoke patterns, their ability to generalize without pre-processing is limited.

Applying Histogram Equalization (HEQ) consistently improves performance, particularly for YOLOv12, which achieves the best overall results across all pre-processing conditions, with a precision of 0.1428, recall of 0.1019, F1-score of 0.1166, and a strong mAP50 of 0.143. This suggests that HEQ enhances smoke visibility and contrast, making

detection more reliable. Other models also show moderate gains compared to the baseline, highlighting the utility of this pre-processing step.

The Cropping strategy alone yields mixed outcomes. While YOLOv12 again benefits, achieving higher F1-Score (0.0750) and mAP50 (0.0783) than the baseline, other models (YOLOv9–11, 13) do not show consistent improvement. In fact, some performance values drop, suggesting that Cropping may sometimes remove contextual information necessary for reliable detection.

Interestingly, the combination of Cropping and HEQ (Crop + HEQ) does not lead to additive benefits. In most cases, performance degrades, with precision and recall dropping sharply, especially for YOLOv9 and YOLOv12. This may be due to an over-aggressive pre-processing pipeline where contrast normalization and spatial Cropping together reduce both image context and smoke texture, making detection harder. YOLOv10 and YOLOv13 show slight stability in their scores under this condition, but not surpassing the best results obtained with HEQ alone.

Overall, the findings suggest that Histogram Equalization is the most effective single pre-processing technique, especially when combined with models like YOLOv12 that are more responsive to enhanced visual features. Cropping, while somewhat helpful for YOLOv12, introduces risks of losing contextual information, and its combination with HEQ does not appear to be synergistic in this dataset.

4.4. Sensitivity Analysis

Next, we provide a sensitivity analysis of the investigated approach considering the deployment of YOLOv13-nano with the combination of Crop-HEQ pre-processing techniques and examine its behavior on different epoch variations and different image resolutions, as illustrated in Figure 16 which depicts the Precision, Recall, F1-score, mAP50, and mAP50_95 metrics averaged over the five-folds. In the plots depicted in Figure 16a,b, the *y*-axis corresponds to the averaged values of these metrics and the *x*-axis corresponds to the number of epochs (a) and the different image resolutions (b).

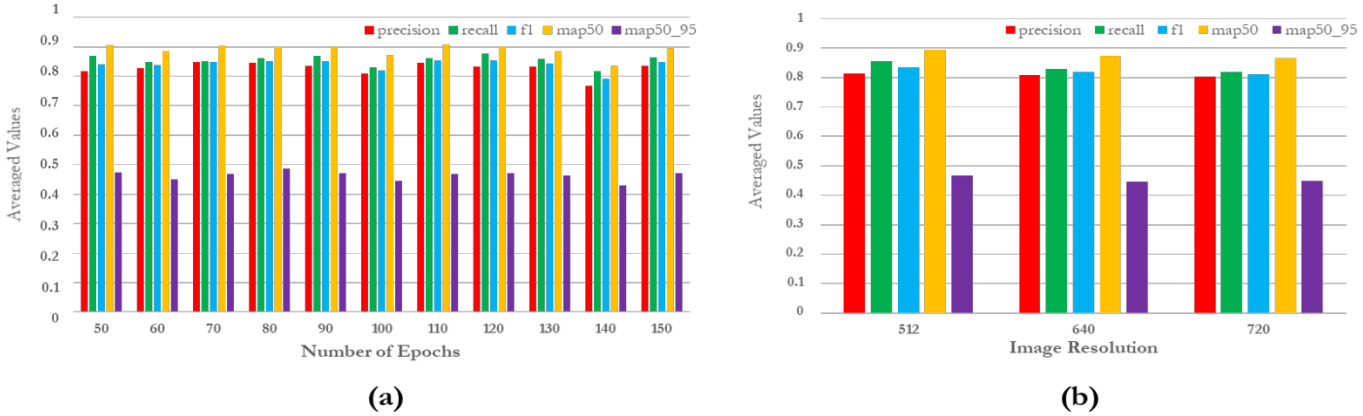


Figure 16. Sensitivity analysis over different numbers of epochs (**a**) and different image resolution (**b**).

To assess the robustness of the proposed model configuration (100 training epochs, input image size of 640), we performed a sensitivity analysis in 2 parts. In the first part image size was fixed and we tested the model by changing the number of epochs ranging from 50 to 150 with a step of 10. In the second part, the number of epochs was fixed and we changed the image size to 512 and 720. The results indicate that the choice of epochs influences detection accuracy.

To investigate the effect of training duration, the number of epochs varied while keeping the input image size fixed at 640. The proposed configuration of 100 epochs

achieved a balanced trade-off across all evaluation metrics, with a precision of 0.8094, recall of 0.8298, F1-score of 0.8190, mAP50 of 0.8726, and mAP50_95 of 0.4458. Increasing the number of epochs to 150 slightly improved precision (0.8338) and F1-score (0.8484), indicating enhanced class discrimination; however, the gain in mAP metrics was marginal (mAP50 = 0.8948, mAP50_95 = 0.4708), suggesting diminishing returns at higher training lengths. Training for 140 epochs showed lower stability, with decreased precision (0.7678) and F1-score (0.7912), while recall (0.8174) and mAP values (0.8340 at mAP50, 0.4280 at mAP50_95) remained weaker compared to the baseline. Conversely, a reduced training regime of 50 epochs resulted in slightly higher recall (0.8682) but at the expense of precision (0.8158), while still achieving the highest mAP50 (0.9064) among all settings. In general, a very high number of epochs can create overfitting in the model, while a low number can create underfitting. Generally, the number of epochs impacts the duration of the training phase and the model performance. An important notice is that in order to achieve the convergence phase (ideal number of epochs without overfitting or underfitting) several tests must be conducted. Moreover, the ideal number of epochs varies when different YOLO variants or pre-processing techniques are used.

The effect of varying the input image size was examined while keeping the number of epochs fixed at 100. The proposed setting of 640 pixels yielded stable and balanced results, with a precision of 0.8094, recall of 0.8298, F1-score of 0.8190, mAP50 of 0.8726, and mAP50_95 of 0.4458. Reducing the image size to 512 led to a moderate increase in precision (0.8146) and F1-score (0.8338), while recall improved to 0.8544 and mAP50 reached 0.8932, showing that smaller input sizes can sometimes improve generalization at the cost of minor detail loss. Increasing the input resolution to 720 pixels produced a precision of 0.8030, recall of 0.8194, F1-score of 0.8108, mAP50 of 0.8656, and mAP50_95 of 0.4480. Compared to the 640-pixel baseline, the performance at 720 pixels shows slightly lower precision and F1, with only marginal differences in recall and mean average precision. This indicates that raising the input resolution beyond 640 does not yield consistent accuracy gains, while incurring higher computational cost.

The image size has a crucial role in object detection. Higher image size requires more computational power and more time for object recognition. This is crucial in tasks like smoke detection. On the other hand a lower image size leads to information loss and thus missed predictions of objects, especially if they are small. In any case, the ideal image size is determined by various factors, such as edge or cloud computing, or object size.

4.5. Discussion

Next, we provide a discussion over the practical deployment considerations on edge-computing devices, the potentials and limitations of this study, as also the lessons learned regarding the insights of the results exhibited through the experimental evaluation procedure.

4.5.1. Practical Deployment Considerations on Edge Devices

In addition to reporting detection accuracy and training performance, it is important to consider the feasibility of deploying YOLO-based smoke detection models on resource-constrained edge devices such as the NVIDIA Jetson Nano and Raspberry Pi. These platforms are suitable for wildfire monitoring because they can operate in remote areas with limited connectivity and power supply.

- Inference speed: Based on reported FLOPs and prior YOLO benchmarks on Jetson-class hardware, lightweight models such as YOLOv9-tiny and YOLOv10-nano achieve real-time throughput (15–20 FPS), whereas more recent variants like YOLOv11-nano and YOLOv13-nano achieve higher accuracy but at reduced speeds (8–12 FPS). This indicates a trade-off between responsiveness and detection precision.

- Accuracy vs. efficiency: YOLOv11-nano and YOLOv13-nano consistently deliver higher mAP scores in our experiments, but the computational overhead may limit their use in latency-critical scenarios. Conversely, YOLOv9-tiny and YOLOv10-nano provide faster inference, making them suitable for early-warning systems where speed is prioritized over marginal gains in accuracy.
- Memory footprint: Model size is another limiting factor: YOLOv9-tiny and YOLOv10-nano require less than 10 MB of storage and under 1 GB RAM for inference, while YOLOv11/YOLOv13 can exceed these requirements. On devices with only 2–4 GB RAM (e.g., Jetson Nano), careful model selection or compression techniques (quantization, pruning) may be required to ensure smooth operation.
- Deployment feasibility: For practical wildfire monitoring, the choice of model should therefore depend on deployment context: YOLOv9-tiny and YOLOv10-nano are best suited for low-power real-time detection on embedded devices, while YOLOv11-nano and YOLOv13-nano may be deployed where higher accuracy is critical and slightly higher latency can be tolerated.

Overall, incorporating hardware constraints into the evaluation highlights the importance of balancing detection accuracy with computational efficiency, ensuring that smoke detection systems remain both effective and deployable in real-world edge environments.

4.5.2. Potentials and Limitations

The paper described how the YOLO based object detection models have the potential to detect smoke in visual images. In addition, those models are relatively small thus they may be deployed to embedded systems that have limited processing power. This is scalable in terms of wide area monitoring capabilities including smart city infrastructure, or the industrial hazards and forest fires.

Firstly, we need to take into consideration that there are limitations. The total size of the dataset (737 images) is sufficiently strong to conduct this initial set of benchmarks. However, it may constrain the model generalization to the more complicated cases of fire recognition. Besides, because of its amorphous size and low contrast, smoke can also prove to be even more problematic when falsely detected by the model in certain cases, which would result in false negatives. Finally, the speed and effectiveness of the YOLO models make them quite useful, but there are situations where the usage of the multi-modal system, with the application of heat or infrared sensors, proves more efficient in comparison to using the YOLO models. These translate into the fact that in cases where there are high risks, hybrid systems are required to the extent that any lapse of error in detection can lead to a disastrous case. The limitations of the study are listed below:

1. Dataset size: The dataset contained 737 annotated smoke images, which is suitable for benchmarking but may limit generalization to more diverse fire scenarios.
2. Visual ambiguity of smoke: Due to its amorphous, semi-transparent nature, smoke can lead to false negatives, particularly in low-contrast scenes.
3. Modality constraints: Our work focuses solely on RGB visual data. Multimodal approaches (thermal or infrared) may provide more reliable detection under certain conditions.

By explicitly stating these limitations, we aim to provide a balanced view of our findings and guide future research directions.

4.5.3. Lessons Learned

Based on the extensive experimental evaluation of multiple pre-processing strategies across different YOLO architectures, the following key lessons were learned:

- Task-specific pre-processing is essential: It was shown that there is more value in domain-specific pre-processing such as Histogram Equalizing and detection-wise Cropping as applied in our wildfire dataset.
- Histogram Equalization consistently improves model accuracy: The Histogram Equalization technique increased the mAP and F1-scores across a wide range of variants as regards the YOLO models. This goes to show that image edge should be improved by maximizing image contrast patterns to detect faint or small objects in low sight scenes.
- Detection-aware Cropping enhances confidence and precision: The confidence and accuracy are enhanced with the detection-aware Cropping: Detection awareness Cropping increased the average confidence in detections, particularly of the YOLOv9-v11 versions.
- Pre-processing impact varies by model architecture: Pre-processing effect is dependent on model architecture: Although the newer architecture like YOLOv13-nano experienced the greatest positive effect of pre-processing, some of the older ones like YOLOv10-nano had more variable pre-processing effects, which suggests that architecture design interacts with pre-processing quite strongly.
- YOLOv13-nano with HEQ and Crop pre-processing is the most robust configuration: This model was the most effective in terms of all major indicators (Precision, Recall, F1, mAP50, mAP50_95) and, therefore, it may be regarded as the one that can be applied to the framework of high-performance wildfire detecting pipelines.

These lessons are essential for future research on selecting and training models as well as fine-tuning the deployment of smoke detection systems.

5. Conclusions

Next, we cite our concluding remarks from this study and provide our goals for further research and improvements that could result from this work.

5.1. Concluding Remarks

The outcomes of this work prove the effectiveness of YOLOv13-nano in determining smoke as it received the highest ranking in terms of mAP, recall and overall precision-recall balancing. The performance of five lightweight object detection models utilizing the YOLO architecture to identify wildfires (YOLOv9-nano to YOLOv13-nano) was discussed in this paper to understand the impacts of the pre-processing techniques of Histogram Equalization and detection-aware Cropping. With this mass experimentation on 5-fold cross-validation, it was shown that pre-processing like Histogram Equalization and Cropping had significant performance benefit in the metric of precision, recall, F1-score and mean average precision (mAP) and it carried insignificant test-time penalty.

The evaluated model YOLOv13-nano demonstrated its outstanding experience compared to the prior designs demonstrated its performance effectiveness across every performance assessment and pre-processing pairings that indicated better generalization and stability. To be more specific, the model with YOLOv13-nano with HEQ and Cropping had the highest mAP50 and mAP50_95 scores, i.e., this combination delivers the best in terms of accuracy trade-off against desired forms of computational efficiency. These results show that the domain-specific visual aspects suggest that the improvements of architecture and pre-processing, which corresponds to these aspects, have a great value. Further investigation will capitalize on the adoption of the recommended pre-processing chain lightweight real-time embedded systems and the utilization of adaptive pre-processing conventions in accordance with the input image characteristics.

5.2. Future Research

Although this research shows the influence of pre-processing methods on lightweight YOLO networks in wildfire recognition area, there are a number of future research directions that are promising:

- Evaluation under adverse environmental conditions: Evaluation of the pipeline working under subjective environmental conditions: The proposed pipeline is to be tested in different weather conditions, illuminating environmental conditions and occlusion conditions to evaluate in generalizability in deployed environment.
- Expansion to multispectral and thermal data: Infrared or multispectral imagery can be incorporated to enhance the low visibility situation detection that would be performed at night or during times of dense smoke.
- Real-time deployment on edge devices: There is still room to optimize and test the hardware part of the program (e.g., on Jetson Nano or Coral TPU) in order to prove the real-time inference with limited resources.
- Exploration of attention mechanisms and transformer-based variants: New lightweight and vision transformers (or attention-augmented YOLO variables) are being proposed that introduce improved context awareness at a small overhead.
- Adaptive pre-processing pipelines: Investigations of dynamic data-driven or learned pre-processing modules that would be adaptive to the character of the input imagery, as opposed to fixed strategies, could be implemented.
- Crowdsourced and drone-based imagery evaluation: The UAV imagery and the crowdsourced data may also be included to make the testbed more varied and to verify the model with the heterogeneous datasets.

Smart environmental systems are expected to completely transform safety infrastructure by using AI-enabled smoke detection as a tool capable of empowering decision-makers to address the impacts of climate change. In this case, we shall discuss some of the breakthroughs made in computer vision technology and their most significant impacts in various sectors such as security, green mobility, and work automation. With open-source technologies such as YOLO becoming widely available, numerous datasets, and the emergence of edge devices with increased power, a next generation of intelligent danger detection systems is already dawning.

Author Contributions: Conceptualization, I.P. and M.A.; methodology, I.P., C.S., I.K. and M.A.; validation, I.P., C.S. and I.K.; writing—original draft preparation, I.P. and C.S.; writing—review and editing, I.P., C.S. and I.K.; supervision, I.P. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in Github accesible through [33].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jocher, G. YOLO by Ultralytics. Available online: https://github.com/ultralytics/yolov5 (accessed on 17 March 2025).
2. Ko, B.C.; Cheong, K.H.; Nam, J.Y. Fire detection based on vision sensor and support vector machines. *Fire Saf. J.* **2009**, *44*, 322–329. [CrossRef]
3. Bugarić, M.; Jakovčević, T.; Stipaničev, D. Computer vision based measurement of wildfire smoke dynamics. *Adv. Electr. Comput. Eng.* **2015**, *15*, 55–62. [CrossRef]
4. Muhammad, K.; Ahmad, J.; Baik, S.W. Early fire detection using convolutional neural networks during surveillance for effective disaster management. *Neurocomputing* **2018**, *288*, 30–42. [CrossRef]

5. Alkhammash, E.H. A Comparative Analysis of YOLOv9, YOLOv10, YOLOv11 for Smoke and Fire Detection. *Fire* **2025**, *8*, 26. [CrossRef]
6. Li, Y.; Nie, L.; Zhou, F.; Liu, Y.; Fu, H.; Chen, N.; Dai, Q.; Wang, L. Improving fire and smoke detection with You Only Look Once 11 and multi-scale convolutional attention. *Fire* **2025**, *8*, 165. [CrossRef]
7. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37. [CrossRef]
8. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
9. Wang, C.; Xu, C.; Akram, A.; Shan, Z.; Zhang, Q. Wildfire smoke detection with cross contrast patch embedding. *arXiv* **2023**, arXiv:2311.10116. [CrossRef]
10. Pan, W.; Wang, X.; Huan, W. EFA-YOLO: An efficient feature attention model for fire and flame detection. *arXiv* **2024**, arXiv:2409.12635. [CrossRef]
11. Dinavahi, G.A.; Addanki, U.K. Assessing the effectiveness of object detection models for early forest wildfire detection. In Proceedings of the 2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI), Tirunelveli, India, 18–20 November 2024; pp. 1451–1458.
12. An, Y.; Tang, J.; Li, Y. A mobilenet ssdlite model with improved fpn for forest fire detection. In Proceedings of the Chinese Conference on Image and Graphics Technologies, Beijing, China, 23–24 April 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 267–276.
13. Liau, H.; Yamini, N.; Wong, Y. Fire SSD: Wide fire modules based single shot detector on edge device. *arXiv* **2018**, arXiv:1806.05363. [CrossRef]
14. Ahmed, H.; Jie, Z.; Usman, M. Lightweight Fire Detection System Using Hybrid Edge-Cloud Computing. In Proceedings of the 2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET), Beijing, China, 13–15 August 2021; pp. 153–157.
15. Wang, J.; Yao, Y.; Huo, Y.; Guan, J. FM-Net: A New Method for Detecting Smoke and Flames. *Sensors* **2025**, *25*, 5597. [CrossRef] [PubMed]
16. Huang, J.; Yang, H.; Liu, Y.; Liu, H. A Forest Fire Smoke Monitoring System Based on a Lightweight Neural Network for Edge Devices. *Forests* **2024**, *15*, 1092. [CrossRef]
17. Vazquez, G.; Zhai, S.; Yang, M. Detecting wildfire flame and smoke through edge computing using transfer learning enhanced deep learning models. *arXiv* **2025**, arXiv:2501.08639. [CrossRef]
18. Wang, Y.; Piao, Y.; Wang, H.; Zhang, H.; Li, B. An improved forest smoke detection model based on YOLOv8. *Forests* **2024**, *15*, 409. [CrossRef]
19. Li, C.; Zhu, B.; Chen, G.; Li, Q.; Xu, Z. Intelligent Monitoring of Tunnel Fire Smoke Based on Improved YOLOX and Edge Computing. *Appl. Sci.* **2025**, *15*, 2127. [CrossRef]
20. Wong K.Y. YOLOv9-Tiny. 2024. GitHub Repository. Available online: https://github.com/WongKinYiu/yolov9 (accessed on 17 March 2025).
21. Ultralytics Team. YOLOv10-Nano. 2024. Ultralytics Docs. Available online: https://docs.ultralytics.com/models/yolov10/#performance (accessed on 17 March 2025).
22. Ultralytics Team. YOLOv11-Nano. 2024. Ultralytics Docs. Available online: https://docs.ultralytics.com/tasks/detect/#models (accessed on 17 March 2025).
23. Tian, Y. YOLOv12: Attention-Centric Real-Time Object Detectors. 2025. GitHub Repository. Available online: https://github.com/sunsmarterjie/yolov12 (accessed on 17 March 2025).
24. Lei, M.; Li, S.; Wu, Y.; Hu, H.; Zhou, Y.; Zheng, X.; Ding, G.; Du, S.; Wu, Z.; Gao, Y. YOLOv13: Real-Time Object Detection with Hypergraph-Enhanced Adaptive Visual Perception. *arXiv* **2025**, arXiv:2506.17733. [CrossRef]
25. iMoonLab. yolov13n.pt [Model Weight File] (Release "YOLOv13 Model Weights") [Data Set]. 2025. Available online: https://github.com/iMoonLab/yolov13/releases (accessed on 15 July 2025).
26. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the CVPR, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
27. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
28. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv* **2022**, arXiv:2207.02696. [CrossRef]
29. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2016**, arXiv:1510.00149. [CrossRef]
30. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]

31. Refaeilzadeh, P.; Tang, L.; Liu, H. Cross-Validation. In *Encyclopedia of Database Systems*; Springer: Boston, MA, USA, 2009.
32. Dwyer, B. Wildfire Smoke [Dataset]. 2020. Available online: https://universe.roboflow.com/brad-dwyer/wildfire-smoke (accessed on 15 July 2025).
33. Smoke Detection YOLO Versions Comparison. Available online: https://github.com/ChristoSarantidis/Smoke_detection_YOLO_versions_comparison/ (accessed on 1 August 2025).