electronics

MDPI

Article

Benchmarking Next-Generation YOLO Architectures for Multi-Platform Forest Fire Recognition

Iosif Polenakis *, Christos Sarantidis and Ioannis Karydis

Department of Informatics, Ionian University, 49100 Corfu, Greece; csarantidis@ionio.gr (C.S.); karydis@ionio.gr (I.K.)

* Correspondence: ipolenakis@ionio.gr

Abstract

Early and reliable detection of forest fires is essential for reducing environmental damage and ensuring public safety. Deep learning-based object detection enables automated fire monitoring across heterogeneous sensing platforms, including satellite, Unmanned Aerial Vehicle (UAV), and ground-based imaging systems. However, differences in spatial resolution, viewing geometry, and computational constraints present challenges for developing unified detection models. This study presents a comparative benchmarking analysis of the lightweight YOLOv26-nano model for forest fire detection using the FASDD dataset, comprising satellite, UAV, and ground-based imagery. A unified experimental protocol with five-fold cross-validation is adopted to ensure robustness and cross-platform generalization. Performance is enhanced through data augmentation, contrast-limited adaptive histogram equalization, and stochastic gradient descent optimization. Experimental results demonstrate that YOLOv26-nano achieves reliable detection accuracy and demonstrates promising computational characteristics under simulated resource-constrained edge-computing conditions. The proposed benchmarking framework provides a standardized reference for multi-platform fire detection and highlights the suitability of nano-scale object detection models for scalable wildfire monitoring and early-warning systems.

Keywords: YOLO; forest fire detection; UAV; drone; satellite imagery; edge computing; multi-platform; object detection; wildfire monitoring; deep learning

1. Introduction

Wildfires have become one of the most destructive and rapidly escalating natural hazards. Driven by climate change, extended drought periods, and shifting land-use practices, large-scale forest fires cause severe ecological damage, threaten human infrastructure, and release substantial quantities of greenhouse gases into the atmosphere. The speed at which wildfires propagate makes early detection critical: timely intervention not only reduces the extent of burned area but also saves lives and limits long-term environmental consequences.

Automated fire detection through computer vision and deep learning has emerged as a compelling complement to traditional monitoring approaches. Early approaches relied on handcrafted features such as colour histograms, motion analysis, and texture descriptors, which were fragile under variable illumination and complex backgrounds. The shift toward deep learning, and convolutional neural networks (CNNs) in particular, transformed detection capabilities by learning discriminative representations directly from data. Among CNN-based detectors, the YOLO (You Only Look Once) [1] family of single-stage detectors has achieved wide adoption owing to its single-pass architecture, competitive accuracy,

Academic Editor: Kefeng Ji

Received: 22 May 2026
Revised: 24 June 2026
Accepted: 25 June 2026
Published: 27 June 2026

and real-time inference speed [2], making it well-suited for deployment-oriented wildfire surveillance. Early visual indicators of fire, including smoke plumes, intense luminosity patterns, and characteristic orange-red coloration, can be reliably captured by trained neural networks even under challenging ambient conditions [3].

A key challenge in this domain is the heterogeneity of sensing platforms used for wildfire observation. Ground-based camera networks, deployed on hilltops, communication towers, and forest lookouts, provide persistent high-temporal-resolution monitoring near terrain level. Unmanned Aerial Vehicles (UAVs) offer flexible, on-demand deployment and high-resolution imagery at closer ranges, making them increasingly important in wildfire response. Satellite sensors provide unparalleled spatial coverage, enabling monitoring of large-scale fire fronts across vast geographic regions. However, each modality introduces distinct visual characteristics—differences in scale, perspective, contrast, and background complexity—that unified detection models must address. Satellite imagery typically features low spatial resolution, atmospheric distortion, and partial cloud occlusion; UAV imagery introduces motion blur, variable altitude, and changing illumination; ground-based cameras suffer from background clutter, vegetation obstruction, and perspective foreshortening of smoke plumes. Developing models that generalize across these conditions motivates the multi-platform benchmarking approach adopted in this work.

This paper presents a comprehensive benchmarking study of the YOLOv26-nano [4] architecture for forest fire detection using the FASDD dataset [5], comprising annotated fire imagery from satellite, UAV, and ground-based sources. Following the experimental methodology established in our prior work [6,7], we adopt a five-fold cross-validation protocol, investigate preprocessing strategies including Contrast Limited Adaptive Histogram Equalization (CLAHE) [8] and data augmentation with SGD optimization, and assess performance across standard detection metrics and simulated edge-computing constraints. It should be noted that this work does not introduce a new architecture; the novelty lies in the comprehensive cross-modality experimental evaluation and the unified benchmarking framework rather than architectural modifications to the detector itself.

1.1. Related Work

Next, we present the related approaches regarding the fire detection through the utilization of object recognition algorithms as well as the field of application regarding the platform they are deployed.

1.1.1. YOLO-Based Fire Detection

YOLO architectures have become the dominant paradigm in real-time wildfire detection due to their single-stage design and favorable accuracy-speed trade-off. Enhanced versions of early YOLO models demonstrated significant performance improvements over their predecessors in detection tasks [9]. More recent efforts have concentrated on lightweight YOLO variants optimized for edge devices. Alkhammash [10] conducted a comparative study of YOLOv9-tiny, YOLOv10-nano, and YOLOv11-nano for wildfire detection, establishing that newer nano variants deliver meaningful gains in mean Average Precision (mAP) and inference speed. Liu et al. [11] augmented YOLOv11 with a multi-scale convolutional attention (MSCA) module to improve smoke detection sensitivity, while Wang et al. [5] incorporated additional feature fusion modules into YOLOv8 to address recognition of thin and diffuse smoke plumes. EFA-YOLO [12] introduced attention-enhanced lightweight modifications targeting ultra-efficient edge deployment. Our prior work [6] systematically benchmarked YOLOv9-tiny through YOLOv13-nano on a ground-based smoke detection dataset, demonstrating that YOLOv13-nano with CLAHE

and detection-aware cropping achieves the most balanced trade-off between accuracy and computational efficiency.

1.1.2. Satellite-Based Fire Detection

Satellite remote sensing underpins the full wildfire management chain, from pre-fire risk characterization through near-real-time active fire detection to post-fire impact assessment. Comprehensive overviews of remote sensing contributions to wildland fire science emphasize the importance of multi-sensor integration and uncertainty characterization [13]. Deep learning approaches for satellite-based active fire detection have expanded rapidly; representative work includes semantic segmentation of multi-sensor imagery [14] and CNN-based detection on geostationary satellite data for near-real-time operational monitoring [15]. Domain-driven modifications to generic detectors, such as the EA-YOLO variant proposed for fire-specific detection [16], illustrate broader trends in adapting lightweight YOLO architectures to satellite-specific constraints such as small targets and high intra-class variability. Our prior benchmarking study on the FASDD satellite subset compared YOLOv12-nano and YOLOv13-nano, finding that YOLOv13-nano achieves superior peak detection performance while YOLOv12-nano offers more stable training dynamics and faster inference.

1.1.3. UAV and Drone-Based Fire Detection

The use of UAVs for wildfire surveillance has grown substantially in recent years, motivated by their flexibility, rapid deployment capability, and ability to obtain high-resolution imagery in terrain inaccessible to ground teams. Early UAV-based fire detection systems relied on thermal infrared cameras exploiting temperature differentials between fire-affected regions and the surrounding environment. Visual-spectrum UAV detection has since advanced through the application of deep learning, with YOLO-based detectors proving well-suited to the real-time constraints of drone-mounted computing platforms.

Detecting fire in UAV imagery introduces specific challenges that differ substantially from ground-level and satellite scenarios. Altitude variation during flight causes significant scale variation in fire and smoke signatures, demanding scale-invariant feature extraction. UAV motion, wind, and rapid frame-to-frame viewpoint changes produce motion blur and inconsistent framing. Proximity to the fire front can result in partial image saturation from intense heat and light, and UAV imagery tends to capture fire from oblique angles or directly above, producing perspective distortions not present in ground-level images.

Research has explored lightweight detector configurations for UAV-mounted platforms, leveraging knowledge distillation to meet the limited computational capabilities of onboard processors. The FASDD dataset [5] provides a standardized UAV subset (FASDD_UAV) that enables controlled evaluation of detection models under these conditions, supporting cross-platform comparisons with satellite and ground-based subsets.

1.1.4. SSD and Alternative Single-Stage Detectors

Beyond YOLO, several studies have examined alternative single-stage architectures. The Single Shot MultiBox Detector (SSD) [11] predicts bounding boxes and class probabilities across multi-scale feature maps and has been evaluated effectively on fire-related datasets. Lightweight backbone replacements for SSD, using architectures such as SqueezeNet and MobileNet, further reduce computational cost and enable real-time inference on edge hardware [17]. Ahmed et al. [18] compared multiple SSD-MobileNet configurations with a custom variant and found a favorable balance between accuracy and computational cost. Although SSD-based frameworks remain a viable choice for lightweight fire detection, YOLO-based models generally outperform them on complex satellite and drone imagery, particularly for small or partially occluded smoke regions.

1.1.5. Feature Pyramid Networks and Small-Object Detection

Detecting small-scale fire and smoke, a particularly acute problem in satellite and high-altitude UAV imagery, has motivated architectural adaptations centered on feature pyramid networks (FPNs). An et al. [17] proposed an improved FPN structure that preserves multi-scale information during down-sampling, demonstrating strong real-time performance on edge devices. Wang et al. [19] introduced FM-Net, a target detection framework based on an enhanced feature pyramid structure that mitigates information loss during the down-sampling process, achieving results indicating suitability for real-time fire detection.

1.1.6. Transformer-Based and Hybrid Approaches

Transformer-based architectures have gained attention for smoke detection because of their capacity to model long-range spatial dependencies. Wang et al. [20] proposed a cross-contrast patch embedding (CCPE) model built on a modified Swin Transformer backbone, demonstrating improved low-level feature representation for smoke identification in complex scenes. While promising, Transformer-based detectors typically incur substantially higher computational costs than YOLO variants, limiting their applicability to resource-constrained platforms and real-time deployments on edge hardware.

1.2. Motivation and Contributions

Existing YOLO-based wildfire detection studies have predominantly focused on individual sensing modalities and often employ heterogeneous datasets, training protocols, and evaluation procedures, making direct comparison across platforms difficult. Furthermore, the rapid evolution of lightweight object detection architectures necessitates continuous evaluation of their suitability for real-world wildfire monitoring applications. To address these limitations, this work presents a comprehensive multi-platform assessment of YOLOv26-nano using the FASDD dataset family. The main contributions of this study are summarized as follows:

- **A unified benchmarking framework for wildfire detection across heterogeneous sensing platforms.** We establish a consistent experimental protocol for evaluating fire detection performance in Ground-based, UAV, and Satellite imagery, enabling direct comparison of model behavior across different observation modalities.
- **The first comprehensive evaluation of YOLOv26-nano across all FASDD sensing modalities.** To the best of our knowledge, this is the first study that systematically investigates the performance of YOLOv26-nano on Ground, UAV, and Satellite wildfire imagery within a common evaluation framework, providing insights into its robustness and generalization capabilities.
- **A platform-specific analysis of image preprocessing strategies.** We investigate the impact of Contrast Limited Adaptive Histogram Equalization (CLAHE), data augmentation, and their combination on wildfire detection performance. The results reveal that preprocessing effectiveness is strongly dependent on the sensing modality, highlighting the importance of modality-aware optimization strategies.
- **An edge-deployment feasibility assessment for resource-constrained environments.** Beyond detection accuracy, we evaluate computational performance through inference latency, CPU utilization, and memory consumption under simulated edge-computing constraints designed to approximate Raspberry Pi and Jetson Nano hardware characteristics. These experiments provide an initial assessment of deployment feasibility rather than validation on physical devices.

2. Data Preprocessing Techniques

This section describes the preprocessing strategies evaluated in this study. A systematic comparison against the baseline (no preprocessing applied) is conducted to determine which combination offers the most consistent improvements in feature visibility and detection robustness across sensing modalities. In our approach, we investigate image preprocessing through Contrast Limited Adaptive Histogram Equalization (CLAHE) and data augmentation, as well as their combination, and assess the impact of SGD optimization on detection performance across sensing platforms. Figure 1 illustrates the pipeline through which the investigated preprocessing techniques are applied, note that, SGD refers to the optimization algorithm used during model training and is listed here as a training configuration variant, not an image preprocessing operation.

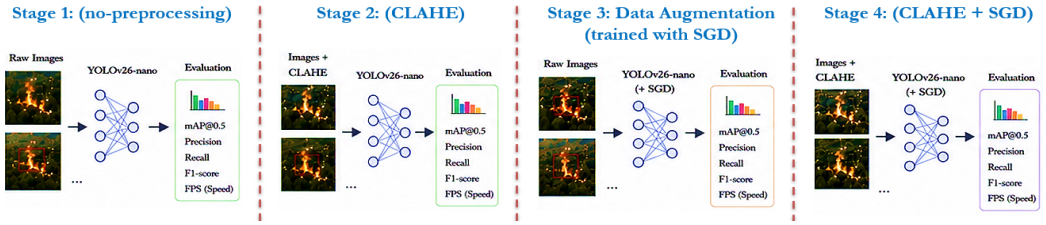


Figure 1. Deployment of image preprocessing techniques.

2.1. Contrast Limited Adaptive Histogram Equalization (CLAHE)

Contrast Limited Adaptive Histogram Equalization was adopted as a primary preprocessing technique owing to its effectiveness in enhancing global image contrast. CLAHE operates by redistributing pixel intensity values so that the resulting histogram is approximately uniform, thereby increasing separability between foreground structures and background regions. Let $r \in \{0, 1, \ldots, L-1\}$ denote the discrete gray-level intensity values of an image, where L is the total number of possible intensity levels. The probability distribution of intensity level r_k is defined as:

$$p(r_k) = \frac{n_k}{N}, \tag{1}$$

where n_k is the number of pixels with intensity r_k and N is the total pixel count. The equalization transformation is:

$$s_k = T(r_k) = (L-1) \sum_{j=0}^{k} p(r_j), \tag{2}$$

where s_k is the new intensity value after equalization. By stretching frequently occurring intensity bands and compressing less frequent ones, CLAHE amplifies subtle contrast variations that are otherwise difficult for convolutional filters to capture in raw images.

For imagery, CLAHE mitigates the effects of variable illumination as the drone changes altitude or orientation relative to the sun. However, CLAHE applies a global transformation and may introduce artifacts in regions of high texture complexity; it is therefore evaluated both in isolation and in combination with data augmentation.

2.2. Data Augmentation with SGD Optimization

Data augmentation and optimization strategy jointly determine the generalization behavior of object detection models. Standard augmentation techniques applied in this study include random scaling, horizontal flipping, color space transformations, mosaic augmentation, mixup, and random cropping. These transformations generate diverse input

distributions, compelling the model to learn invariant representations. The SGD parameter update rule is:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t), \tag{3}$$

where θ_t denotes the model parameters at iteration t, η is the learning rate, and $\nabla L(\theta_t)$ is the gradient of the loss function over an augmented mini-batch. SGD is enhanced with momentum and weight decay to stabilize training under high augmentation variance.

The interaction between augmentation and SGD is synergistic. Strong augmentation increases gradient stochasticity, which aligns with SGD's inherent randomness and produces several operational benefits:

- **Improved generalization**: Augmented samples expose the model to diverse feature distributions, while SGD prevents overfitting by avoiding sharp minima.
- **Regularization effect**: Augmentation acts as implicit regularization, complementing the momentum mechanism of SGD through noisy gradient updates.
- **Stable convergence**: Momentum-based SGD smooths gradient fluctuations caused by aggressive augmentations such as mosaic and mixup.
- **Scale-invariant learning**: Random scaling and cropping alter apparent object sizes, and SGD adapts weights incrementally to develop scale-invariant representations, which are critical for small-object detection in high-altitude UAV and satellite imagery.

2.3. Combined Effect of CLAHE and Data Augmentation

When CLAHE and data augmentation are applied jointly, their effects are complementary at different stages of the learning pipeline. CLAHE operates on the input image distribution prior to feature extraction, enhancing global contrast and improving the visibility of low-contrast structures such as faint smoke plumes. Data augmentation then introduces geometric and photometric variability, forcing the model to learn representations invariant to diverse visual conditions.

A key design consideration is whether the sequential order of applying CLAHE before versus after data augmentation affects model performance. CLAHE is a deterministic global transformation, while data augmentation consists of stochastic spatial and photometric operations. Empirically, both orderings yield similar distributions when averaged across training iterations, confirming that the combination of transformations matters more than their strict ordering. The stochastic exposure provided by SGD across mini-batches reduces sensitivity to transformation order and encourages the model to learn contrast- and geometry-invariant features simultaneously.

3. Challenges in Fire Recognition

Wildfire monitoring occupies a distinctive niche in the fire surveillance ecosystem. This section discusses the specific challenges that imagery presents for deep learning-based fire detection, describes the capabilities of drone platforms used in fire response, and outlines the detection methodology applied to the FASDD dataset.

3.1. Platforms for Wildfire Monitoring

Modern platforms used in wildfire monitoring span a wide range of capabilities. Fixed-wing drones offer extended endurance and large-area coverage at the cost of operational flexibility; multirotor platforms provide the hovering capability and rapid deployment needed for targeted observation of active fire fronts. Both types are routinely equipped with visual-spectrum RGB cameras, near-infrared sensors, or thermal imaging payloads. For the FASDD_UAV subset and the experiments described in this paper, visual-spectrum imagery is the primary modality, consistent with deployments where cost-effective commercial drone hardware is used.

Operational fire monitoring drones are increasingly equipped with onboard computing units, such as NVIDIA Jetson Nano or Raspberry Pi-class processors, that support real-time inference at the edge, eliminating the latency associated with transmitting raw video to a ground station. The nano-scale YOLO models evaluated in this study are specifically targeted at this computational profile.

3.2. Imagery Characteristics and Properties in Fire Detection

Detecting fire in UAV imagery involves a distinct set of challenges compared to satellite or ground-based scenarios:

- **Altitude and Scale Variation**: As a UAV ascends or descends during a mission, fire objects appear at very different apparent scales within the field of view. A smoke plume that fills a significant portion of the frame at low altitude may appear as a small, diffuse region when the drone climbs to a survey altitude. Detection models must therefore be robust to substantial intra-flight scale variation, favoring architectures with strong multi-scale feature extraction.
- **Motion Blur and Platform Instability**: UAV platforms are subject to wind-induced vibrations and rapid orientation changes that produce motion blur in captured frames. Unlike static ground cameras, the sensor's viewpoint can shift substantially between consecutive frames, disrupting temporal feature consistency. Preprocessing strategies that increase texture contrast, such as CLAHE, partially mitigate the impact of motion blur on feature detectability.
- **Perspective Distortion**: Fire observed from above (nadir view) exhibits a fundamentally different appearance than fire observed from a lateral ground perspective. Smoke plumes rising vertically appear as expanding circular or elliptic regions from nadir, while from an oblique perspective they appear as elongated, directional features. Detection models trained on multi-angle UAV data must generalize across both perspectives.
- **Dynamic Background Complexity**: At low altitude, UAV imagery captures fine-grained background texture including tree canopies, water surfaces, and terrain, which can visually resemble or partially occlude smoke. At high altitude, thin cloud layers and haze may be confused with distant smoke plumes. This dual-scale complexity demands robust feature discrimination.
- **Illumination and Glare**: Low sun angles at dawn and dusk, precisely the time when fires are most dangerous, introduce strong directional shadows, highlight glare, and rapid temporal illumination changes that degrade detection confidence. Data augmentation with color jitter and random brightness adjustments is particularly important for training models that must operate reliably under these conditions.

3.3. Detection Methodology

The detection pipeline for the FASDD subsets closely follows the five-fold cross-validation protocol applied uniformly to the dataset to ensure robustness of reported metrics across data splits. To this point, it should be noted that all deployment evaluations reported in this work correspond to simulated edge-device environments and not physical hardware implementations. In particular, the detection pipeline consists of the following stages:

- **Dataset preparation**: The FASDD_UAV and FASDD_CV subsets are prepared as a single-class fire detection dataset. Only fire annotations are retained, while smoke-only samples and smoke annotations are excluded, ensuring consistency with the FASDD_RS subset, which is annotated only for fire. Moreover, for CLAHE application

in FASDD_RS the images had to be converted from one channel to three channels, as images where initially in .tiff format.

- **Unified cross-validation protocol**: A five-fold cross-validation scheme is applied to the each one of the subsets. This reduces split-dependent bias and allows a fair comparison of detection performance across sensing platforms.
- **Preprocessing configurations**: YOLOv26-nano is evaluated under the four preprocessing settings used throughout the paper: baseline without preprocessing, CLAHE, data augmentation with SGD optimization, and the combined CLAHE+SGD configuration. This enables direct assessment of whether preprocessing benefits UAV imagery in the same way as ground-based or satellite imagery.
- **UAV-specific visual challenges**: UAV imagery introduces altitude-dependent scale variation, oblique and nadir viewpoints, motion blur, platform instability, and illumination changes. These factors make the UAV subset an important test case for evaluating whether YOLOv26-nano can maintain robust fire detection under operationally realistic aerial imaging conditions.
- **Detection evaluation**: Model accuracy is assessed using the same object detection metrics reported throughout the study, namely Precision, Recall, F1-score, mAP_50, and mAP_50–95. Since the task is formulated as single-class detection, all reported metrics refer exclusively to the fire class.
- **Deployment-oriented evaluation**: The trained models are also evaluated under simulated Raspberry Pi and Jetson Nano resource constraints. These experiments report latency, FPS, CPU utilization, and RAM utilization, allowing the UAV results to be interpreted not only in terms of detection accuracy but also in terms of edge-computing feasibility.

This methodology supports the central comparative goal of the paper: to determine whether YOLOv26-nano provides reliable fire detection across heterogeneous sensing platforms while remaining suitable for resource-constrained deployment. For early warning, this is particularly important because fire monitoring requires both high detection robustness and low-latency inference for onboard or near-real-time decision support.

It should be emphasized that the Raspberry Pi and Jetson Nano evaluations correspond to simulated edge-computing environments rather than physical hardware implementations. Therefore, the reported runtime, CPU, memory, and FPS values should be interpreted as relative indicators of computational feasibility and deployment efficiency. Final validation on physical UAV-mounted hardware remains necessary before drawing definitive conclusions about operational field performance.

4. Methodology

Next, we present the properties of the YOLOv26-nano investigated in this work and the evaluation metrics to measure the accuracy of the algorithm in real-time fire recognition over different platforms, while also discussing the training and inference configuration settings.

4.1. Investigated Model: YOLOv26-Nano

This study focuses on the YOLOv26-nano architecture, representing the latest generation in the YOLO family of lightweight single-stage object detectors. Building on the architectural lineage established through YOLOv12 and YOLOv13, which introduced hypergraph-enhanced feature representation, refined attention mechanisms, and improved gradient flow during training [21,22], YOLOv26-nano advances these principles within an ultra-compact parameter budget suitable for edge deployment. The nano variant is designed to operate on resource-constrained hardware including UAV onboard processors, Jetson Nano-class devices, and Raspberry Pi systems, while maintaining competitive de-

tection performance. Five-fold cross-validation experiments are conducted with identical hyperparameter configurations across all sensing modalities to ensure fair and reproducible comparisons. The three stages of the procedure workflow are illustrated in Figure 2.

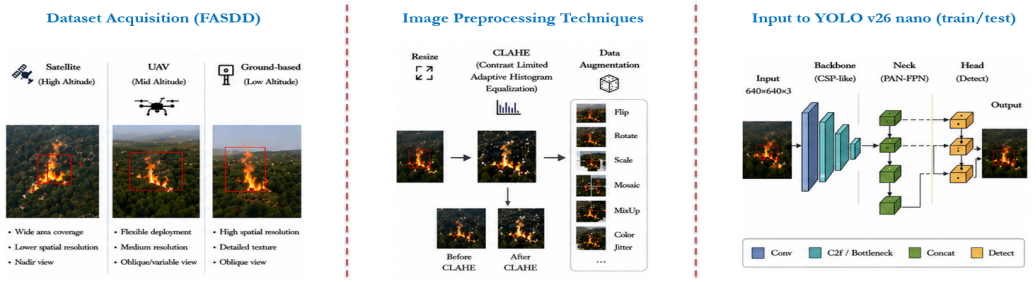


Figure 2. Methodology work-flow for forest fire recognition utilizing YOLO v26 nano in ground-based, UAV, and satellite imagery.

4.2. Effectiveness Metrics

Model performance is evaluated using standard object detection metrics, defined in terms of true positives (TP), false positives (FP), and false negatives (FN).
Precision quantifies the proportion of correctly identified objects among all predicted instances:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{4}$$

Recall quantifies the proportion of ground-truth fire instances successfully detected. Since the experimental protocol was formulated as a single-class fire detection task, all reported Precision, Recall, F1-score, AP, and mAP values correspond exclusively to the fire class:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{5}$$

F1-Score provides a balanced composite of precision and recall:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{6}$$

Mean Average Precision at IoU = 0.5 (mAP_{50}) summarizes precision-recall performance across all classes at an Intersection over Union (IoU) threshold of 0.50. For a single class, Average Precision (AP) is the area under the precision-recall curve:

$$\text{AP} = \int_0^1 \text{Precision}(\text{Recall})\, d\,\text{Recall}. \tag{7}$$

The mAP_{50} is computed as the mean AP across all C classes:

$$\text{mAP}_{50} = \frac{1}{C} \sum_{i=1}^{C} \text{AP}_i^{\text{IoU}=0.5}. \tag{8}$$

Training time and inference latency are also recorded to characterize computational efficiency and assess suitability for real-time deployment across sensing platforms.

4.3. Training and Inference Configuration

Regarding the training configuration, all experiments use a consistent hyperparameter configuration to ensure comparability across sensing modalities and preprocessing conditions. Table 1 summarizes the configuration and rationale for each hyperparameter choice. For the inference of the model, we used Windows Subsystem for Linux in order to

simulate the microcomputers' operating system, and Docker in order to limit the CPU and RAM usage.

Table 1. Training hyperparameter configuration and justification.

Hyperparameter	Justification
Epochs = 90, 140, 150	Sufficient for stable convergence on dataset subsets of this scale; sensitivity analysis demonstrated these are the appropriate epoch numbers.
Image size = 1000	Balances fine-detail preservation against computational cost.
Batch size = 39	Optimized for available GPU memory; maximizes throughput without overflow.
Optimizer = SGD	Strong generalization under heavy augmentation; well-characterized behavior with cosine decay scheduling.
Cross-validation	Five-fold scheme reduces evaluation variance and improves generalization estimates.
CLAHE Clip Limit = 2.0	This value provides balanced enhancement and better local contrast. It is highly recommended for YOLO preprocessing, wildfire datasets and UAV or Satellite imagery.
CLAHE Tile Grid Size = (8,8)	This Grid Size also achieves a high level of local enhancement and stronger effect.

The commands presented in Table 2 were used to simulate the constraints of each edge device investigated for the deployment of YOLO-nano v26.

Moreover, for the JETSON Simulation, we limited GPU clock speed to 921 MHz using the following command: `sudo nvidia-smi -lock-gpu-clocks = 921,921`. Additionally we locked the GPU power to 10 W, using the command: `sudo nvidia-smi -pl 10`. After the experiment, we restored the GPU to its initial settings, using the following commands: `sudo nvidia-smi -reset-gpu-clocks` and `sudo nvidia-smi -pl 200`.

The computer in which the experiments were conducted was equipped with an NVIDIA-GeForce RTX 4070 of 12 GB of Dedicated GPU memory and a 13th Gen Intel(R) Core(TM)i9-13900F CPU.

Table 2. Docker run commands used to achieve the simulated constraints for Raspberry Pi 5 and NVIDIA Jetson Nano.

Raspberry Pi 5	NVIDIA Jetson Nano

`docker run -it -rm`
`-cpus=4`
`-cpuset-cpus=0-3`
`-memory=4g`
`-memory-swap=4g`
`-shm-size=1g`
`-v ~/yolo-experiment:/workspace`
`-w /workspace`
`nvcr.io/nvidia/pytorch:23.10-py3`
`bash`
	`docker run -it -rm`
	`-gpus all`
	`-runtime nvidia`
	`-cpus=4`
	`-cpuset-cpus=0-3`
	`-memory=4g`
	`-memory-swap=4g`
	`-shm-size=2g`
	`-v ~/yolo-experiment:/workspace`
	`-w /workspace`
	`nvcr.io/nvidia/pytorch:23.10-py3`
	`bash`

It should be noted that the Raspberry Pi and Jetson Nano evaluations were performed in a simulated edge-computing environment rather than on physical hardware devices. Docker-based resource limitations and hardware configuration settings were employed to approximate the computational characteristics of these platforms. Therefore, the reported latency, throughput, CPU utilization, and memory-consumption measurements should be

interpreted as indicators of deployment feasibility and relative computational efficiency rather than definitive measurements of real-world edge-device performance.

It should be noted that the Raspberry Pi and Jetson Nano experiments were conducted in a deployment-oriented simulation environment rather than on physical edge devices. Hardware constraints representative of these platforms were emulated using Docker resource limitations and GPU configuration settings. Consequently, the reported computational performance metrics should be interpreted as estimates of deployment feasibility and relative computational efficiency rather than definitive measurements of real-world edge-device performance.

4.4. Workflow of the Proposed Scheme

Figure 3 illustrates the work-flow pipeline on the deployment of YOLO v26 nano in fire recognition. Initially, the FASDD dataset is partitioned into three subsets according to sensing modality: ground imagery (terrestrial captures), UAV imagery (drone-acquired), and satellite imagery (remote sensing). This separation allows the pipeline to evaluate how detector performance varies across fundamentally different viewpoints and image characteristics. All three modalities then feed into a shared image preprocessing stage, where each image is processed under four distinct conditions: a baseline (B) with no enhancement, CLAHE (C) for contrast-limited adaptive histogram equalization, data augmentation with SGD optimization (S), and a combined CLAHE + SGD (C/S). This produces multiple versions of the dataset, enabling a controlled comparison of how preprocessing choices affect downstream detection. The preprocessed images are passed to YOLOv26-nano, a lightweight object detection model optimized for efficiency, which is trained and run for inference across all modality–preprocessing combinations. The resulting detections are evaluated along two parallel tracks. The first is detection performance, measured using standard metrics: Precision, Recall, F1-score, mAP50, and mAP50-95,in order to assess how accurately the model identifies fire. The second is edge-oriented evaluation, which profiles the model under simulated hardware constraints representative of a Raspberry Pi and a Jetson Nano, capturing how the model behaves under the memory and computation limitations of real-world embedded deployment. Finally, results from both tracks are brought together in a comparative analysis that examines how performance varies across the three sensing modalities and across the different deployment configurations, surfacing the preprocessing and hardware trade-offs most relevant to practical fire detection systems.

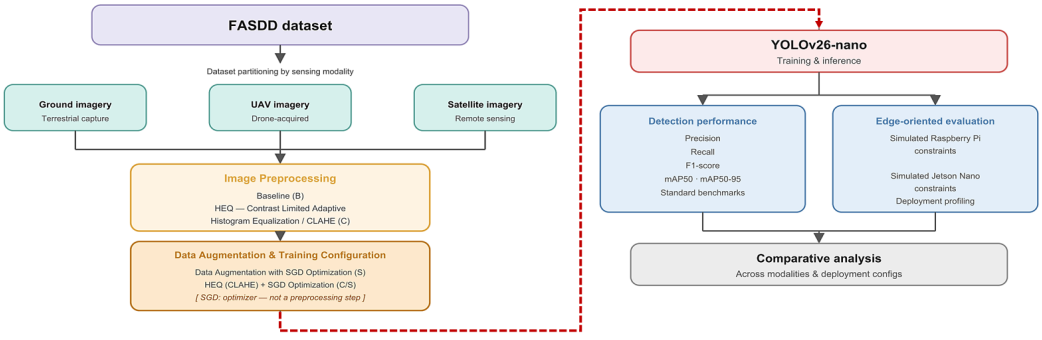


Figure 3. Work-flow pipeline on the deployment of YOLO v26 nano in fire recognition using the FASDD dataset.

5. Experimental Results

Next, we present the dataset used for the evaluation of the model in different platforms, and discuss its performance across different datasets (i.e., ground, uav, and satellite).

5.1. Dataset

The experiments in this work are conducted on the Flame and Smoke Detection Dataset (FASDD) [5], a large-scale and highly diverse benchmark specifically designed for object detection in complex fire-related scenarios. FASDD encompasses annotated imagery from three distinct sensing platforms, making it uniquely suited to evaluating models intended for deployment across heterogeneous real-world monitoring systems.

5.1.1. FASDD Overview

FASDD comprises fire, smoke, and confounding non-fire imagery acquired under a wide range of conditions, including varying observation distances, indoor and outdoor environments, diurnal illumination cycles, and multiple sensor types. All fire objects are annotated using the standard YOLO bounding box format, with class labels distinguishing fire from smoke instances (note that, in our case, we utilize only the fire labeled imagery). The dataset is organized into three primary subsets:

- **FASDD_CV**: General computer vision scenarios captured by ground-based surveillance cameras located on hills, mountains, and in forest environments. Images in this subset represent real-world monitoring conditions with complex vegetative backgrounds and variable atmospheric conditions.
- **FASDD_UAV**: Imagery captured by unmanned aerial vehicle platforms at variable altitudes and viewing angles. This subset includes fire observations from oblique and nadir perspectives, with scale variation introduced by altitude changes during flight.
- **FASDD_RS**: Satellite remote sensing imagery comprising 2223 images, at a resolution of 1000×1000 pixels. This subset captures large-scale fire events from orbital altitude that may span large geographic regions.

An additional subset, FASDD_RS_SWIR, provides pseudo-color short-wave infrared (SWIR) imagery to support flame detection under conditions where visual-spectrum detection is limited. For the present study, we focus on the three primary subsets to evaluate model generalization across visual-spectrum modalities.

5.1.2. Experimental Protocol

For controlled and reproducible evaluation, representative subsets were selected from each FASDD partition. From the FASDD_RS satellite subset, 2223 images were selected and randomly divided into 1333 training images, 445 validation images, and 445 test images. Five-fold cross-validation protocol was employed in order to ensure generalizability of the proposed work. Equivalent sampling procedures were applied to the FASDD_UAV and FASDD_CV subsets. It is important to note that we used only the fire class of each dataset. This choice was made because FASDD_RS was labeled only for the fire class and not for the smoke class. Hence, we experimented on the intersection of the labels for all three FASDD subsets. Additionally the subset contained 2223 images in total, and thus the same number of images was chosen from the other 2 subsets in order to have comparison over all three of them. Table 3 presents the Train, Validation, and Test images partitions as well as the number of fire instances in each FASDD subset.

Table 3. Dataset partition and instances count.

Dataset	Train	Validation	Test	Fire Instances
FASDD_RS	1333	445	445	3549
FASDD_UAV	1333	445	445	15,089
FASDD_CV	1333	445	445	4441

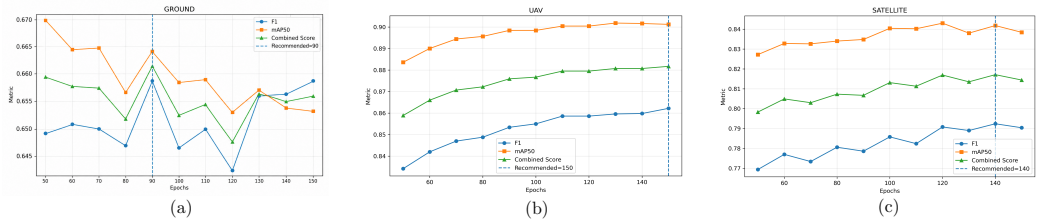
All images were annotated for fire objects under a consistent labeling protocol. Experiments were conducted in three stages: a baseline evaluation without preprocessing, followed by experiments incorporating preprocessing techniques as Contrast Limited Adaptive Histogram Equalization (CLAHE), Stochastic Gradient Descent (SGD) optimization, as well as their combination as a combined preprocessing technique. This progression enables isolation of the contribution of each preprocessing component to overall detection performance.

To this point, we should note that the original FASDD dataset provides annotations for both fire and smoke objects, and the present study focuses exclusively on fire detection. Accordingly, only images containing fire instances were considered during dataset preparation. Images labeled as fire-only and images containing both fire and smoke were retained, whereas smoke-only images were excluded. For images containing both fire and smoke, only the fire annotations were preserved and all smoke annotations were removed. Consequently, YOLOv26-nano was trained and evaluated as a single-class object detector, with fire being the only target category throughout all experiments.

5.2. Training over Different Epochs

The ground-based subset provides the highest spatial resolution and most direct visual observations of active fire fronts. Under these conditions, YOLOv26-nano achieves the highest absolute detection metrics among the three sensing modalities, reflecting the favorable match between training conditions for YOLO-based detectors and the ground-level observation geometry. Detection-aware cropping is particularly beneficial for the ground-based subset, as ground cameras often frame large scenes in which smoke plumes appear as small regions relative to the overall image extent. The combined preprocessing pipeline achieves the best overall F1-score on the ground-based subset among all preprocessing configurations.

Figure 4 illustrates the sensitivity analysis over different training epochs, where the dashed blue vertical line indicates the recommended point is at 90 epochs. To determine the optimal training duration for the ground-based fire detection model, a sensitivity analysis was conducted by evaluating three key metrics, F1-score, mAP_{50}, and a Combined Score, across a range of epochs from 50 to 150, in increments of 10. As illustrated in Figure 4a, all three metrics exhibit a peak at 90 epochs, with mAP_{50} reaching approximately 0.664, F1-score 0.659, and the Combined Score 0.661, establishing this point as the recommended training configuration. Prior to 90 epochs, the model demonstrates a gradual improvement trend, indicating that the network has not yet fully converged. Beyond 90 epochs, a consistent degradation is observed across all metrics, punctuated by a notable drop at 120 epochs where the F1-score falls to its lowest recorded value (0.642), suggesting the onset of overfitting as the model begins to memorize training data at the expense of generalization. Although a partial recovery is visible between 130 and 150 epochs, performance never recovers to the 90-epoch optimum. These findings confirm that 90 epochs represents the best trade-off between model convergence and generalization for ground-based fire detection, and this value was adopted for all subsequent ground-model experiments.



(a) (b) (c)

Figure 4. Sensitivity analysis on the performance of YOLO algorithm in ground-based imagery (**a**), UAV-based imagery (**b**), and satellite-based imagery (**c**) across training epochs.

In contrast to the ground-based model, the UAV-based fire detection model demonstrates a distinctly different convergence behavior, as illustrated in Figure 4b, where the dashed blue vertical line indicates the recommended point is at 150 epochs. All three metrics: F1-score, mAP_{50}, and Combined Score, exhibit a consistent and monotonically increasing trend throughout the entire epoch range from 50 to 150, without any sign of overfitting or performance degradation. mAP_{50} rises from 0.884 at 50 epochs to approximately 0.901 at 150 epochs, the F1-score improves from 0.837 to 0.863, and the Combined Score advances from 0.859 to 0.881, leading to the selection of 150 epochs as the recommended training configuration. The smooth, plateau-approaching curves suggest that the UAV dataset possesses sufficient diversity and volume to sustain continuous learning without memorization artifacts, and that the model had not yet reached full saturation even at the upper bound of the tested range. The considerably higher absolute metric values compared to the ground-based scenario, with mAP_{50} exceeding 0.90, reflect the more structured and consistent nature of UAV aerial imagery, where fire instances tend to appear with greater visual regularity and less background clutter. These findings indicate that UAV-based models may benefit from training beyond 150 epochs, and future work could explore extended training schedules to determine whether further gains are achievable.

The satellite-based fire detection model exhibits a convergence pattern that sits between the volatility of the ground-based model and the smooth monotonic progression of the UAV-based model, as illustrated in Figure 4c, where the dashed blue vertical line indicates the recommended point is at 140 epochs. All three metrics: F1-score, mAP_{50}, and Combined Score, display a generally ascending trend from 50 to 140 epochs, with minor oscillations at 70 and 110 epochs reflecting the greater complexity and heterogeneity inherent in satellite imagery. mAP_{50} rises from 0.827 at 50 epochs to a peak of approximately 0.843 at 140 epochs, while the F1-score improves from 0.770 to 0.792 and the Combined Score from 0.799 to 0.818, establishing 140 epochs as the recommended training configuration. Beyond this point, a mild but consistent decline is observed across all three metrics at 150 epochs, indicating the early onset of overfitting, which is less pronounced than in the ground-based case but sufficient to justify capping training at 140 epochs. The absolute metric values achieved by the satellite model are intermediate between the ground and UAV cases, reflecting the inherently more challenging nature of satellite fire detection, where lower spatial resolution, atmospheric interference, and spectral variability make fire pixel discrimination more difficult than in aerial imagery. These findings suggest that satellite-based models require a longer training schedule than ground-based models to extract sufficient discriminative features, yet remain susceptible to overfitting beyond a well-defined optimum, reinforcing the importance of careful epoch selection for this detection modality.

5.3. Performance on Ground-Based Imagery (FASDD_CV)

Figure 5 and Table 4 show that the unprocessed baseline provided the best overall ground-based detection performance. It achieved precision, recall, F1-score, mAP_50, and mAP_50–95 values of 0.687, 0.611, 0.647, 0.641, and 0.325, respectively. Augment (SGD) was effectively equivalent, with values of 0.686, 0.606, 0.644, 0.634, and 0.320. Thus, augmentation did not materially improve accuracy, although it preserved the baseline trade-off between sensitivity and false alarms.

CLAHE was clearly detrimental. Its precision, recall, F1-score, mAP_50, and mAP_50–95 fell to 0.573, 0.410, 0.477, 0.415, and 0.188. Adding SGD recovered precision to 0.608 and F1-score to 0.490, but recall remained 0.411 and localization remained far below baseline (mAP_50 = 0.416; mAP_50–95 = 0.192). The reviewer-relevant conclusion is therefore not merely that CLAHE failed to help, but that it changed the ground-image

distribution enough to impair both classification and localization; SGD only partially mitigated this loss. Because the same trained weights were evaluated on both devices, these accuracy metrics are platform-independent.

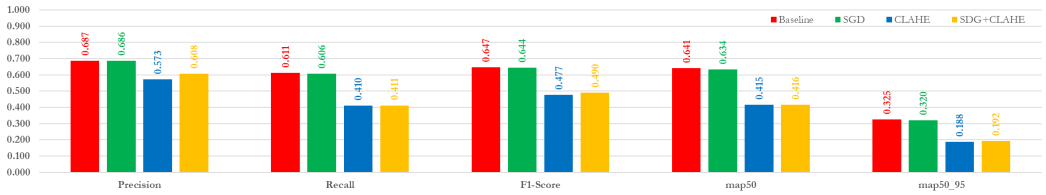


Figure 5. Averaged Precision, Recall, F1-Score, mAP_50 and mAP_50–95 metrics after the five-fold cross validation of YOLO v26 nano on the ground based imagery.

Table 4. Detection performance metrics for Ground imagery. Values are reported as mean ± standard deviation.

Method	Precision	Recall	F1-Score	mAP_50	mAP_50–95
Baseline	0.687 ± 0.028	0.611 ± 0.027	0.647 ± 0.021	0.641 ± 0.023	0.325 ± 0.012
Augment (SGD)	0.686 ± 0.028	0.606 ± 0.015	0.644 ± 0.018	0.634 ± 0.031	0.320 ± 0.014
HEQ (CLAHE)	0.573 ± 0.038	0.410 ± 0.033	0.477 ± 0.026	0.415 ± 0.026	0.188 ± 0.015
Augment + HEQ (SGD + CLAHE)	0.608 ± 0.020	0.411 ± 0.034	0.490 ± 0.021	0.416 ± 0.018	0.192 ± 0.014

The confusion matrices in Figure 6 support this conclusion. The baseline produced 589 correct fire predictions, 306 fire-to-background errors, and 300 background-to-fire errors. Augment (SGD) produced 576, 258, and 312, respectively. CLAHE reduced correct fire predictions to 394 while producing 295 fire-to-background and 495 background-to-fire errors; SGD + CLAHE yielded 397, 263, and 492. No configuration correctly classified a background sample. This persistent zero-background result is a central limitation and should be discussed as possible class imbalance, label construction, thresholding, or confusion-matrix interpretation rather than as a minor implementation detail. The fold-level dispersion reported in Table 4 indicates that the baseline and SGD results were also more consistent than the CLAHE result.

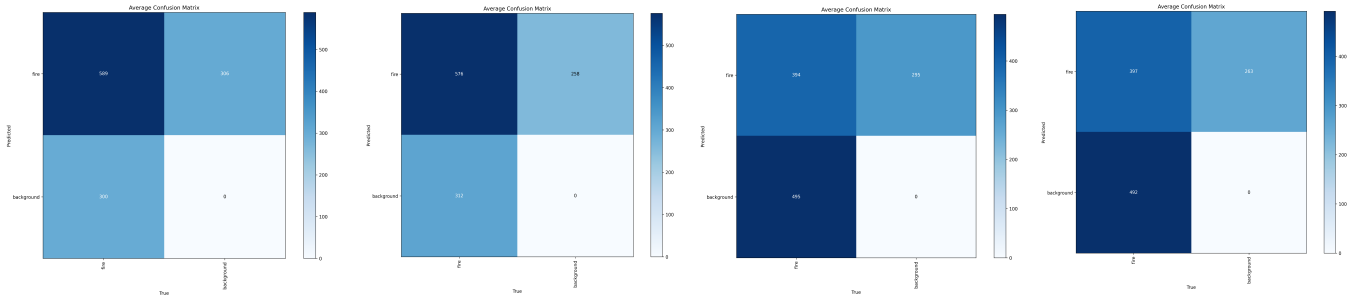


Figure 6. Confusion matrices for all the investigated cases of image preprocessing, no preprocessing (baseline) (**a**), Augment (SGD) (**b**), HEQ (CLAHE) (**c**), and deploying both augment and HEQ (SGD + CLAHE) (**d**), for the ground based imagery.

The threshold-dependent curves in Figure 7 provide the same ranking. Baseline and SGD sustain higher precision and broader F1 peaks, whereas both CLAHE configurations exhibit lower peaks and faster recall loss. Operationally, the baseline is the safest accuracy-oriented choice; SGD is a near-equivalent alternative, but CLAHE is not supported for ground-based deployment.

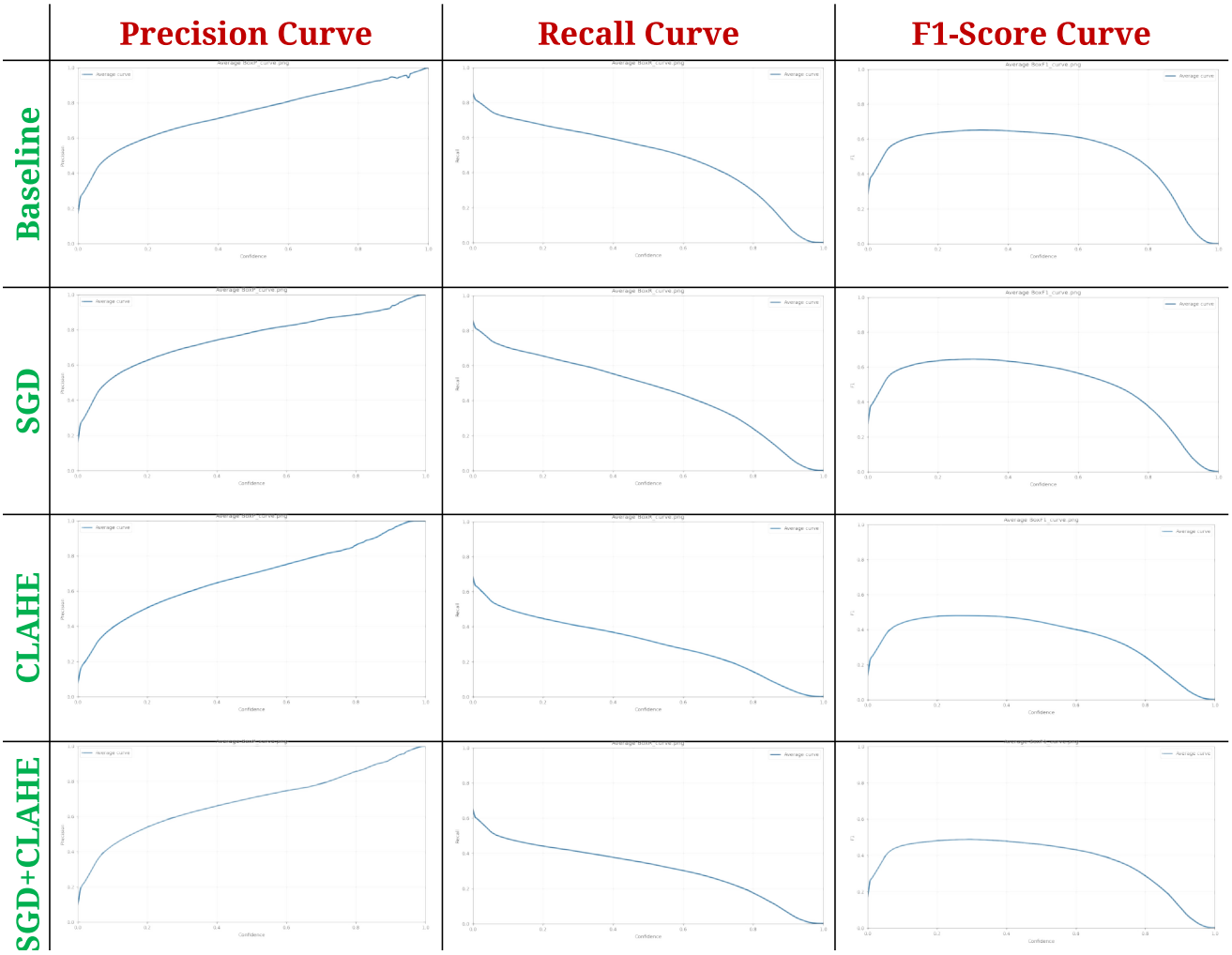


Figure 7. Precision, Recall, and F1-Score average curves for the Ground dataset.

5.3.1. Ground Fire Detection Under Simulated Raspberry Pi Constraints

Table 5 reports wall-clock performance, Table 6 reports CPU behavior, and Table 7 reports memory use on Raspberry Pi. The baseline processed the test set in 34.419 s at 77.346 ms/image and 12.932 FPS. Augment (SGD) changed throughput only slightly to 13.008 FPS. CLAHE reached 13.280 FPS, and SGD + CLAHE was fastest at 13.356 FPS and 74.874 ms/image, corresponding to approximately 3.3% higher throughput and 3.2% lower latency than baseline.

Table 5. Inference performance evaluation of YOLO v26 nano on Raspberry Pi for Ground imagery.

Method	Wall Time (s)	Wall Time (ms/Image)	FPS
Baseline	34.419 ± 0.507	77.346 ± 1.142	12.932 ± 0.193
Augment (SGD)	34.214 ± 0.556	76.886 ± 1.249	13.008 ± 0.211
HEQ (CLAHE)	33.521 ± 0.470	75.324 ± 1.057	13.280 ± 0.188
Augment + HEQ (SGD + CLAHE)	33.319 ± 0.480	74.874 ± 1.080	13.356 ± 0.191

Table 6. CPU performance metrics of YOLO v26 nano on Raspberry Pi for Ground imagery.

Method	Time (s)	ms/Image	Cycles (s)	Cycles/Image	Mean CPU (%)	Max CPU (%)
Baseline	94.765 ± 1.453	212.956 ± 3.266	94.760 ± 1.453	0.213 ± 0.003	8.590 ± 0.074	11.800 ± 2.112
Augment (SGD)	94.433 ± 1.603	212.210 ± 3.600	94.436 ± 1.604	0.212 ± 0.004	8.642 ± 0.145	11.860 ± 1.297
HEQ (CLAHE)	92.530 ± 1.262	207.934 ± 2.835	92.530 ± 1.260	0.208 ± 0.003	8.632 ± 0.084	10.900 ± 0.943
Augment + HEQ (SGD + CLAHE)	92.275 ± 1.145	207.358 ± 2.573	92.274 ± 1.148	0.207 ± 0.003	8.720 ± 0.065	11.920 ± 1.708

Table 7. Memory utilization of YOLO v26 nano on Raspberry Pi for Ground imagery.

Method	Mean RAM (%)	Max RAM (%)	Mean RAM (MB)	Max RAM (MB)
Baseline	10.736 ± 0.814	14.880 ± 1.130	2982.696 ± 260.030	4309.904 ± 358.729
Augment (SGD)	11.042 ± 0.421	15.220 ± 0.960	3080.352 ± 134.271	4416.250 ± 303.196
HEQ (CLAHE)	10.954 ± 0.406	15.220 ± 0.960	3053.130 ± 129.338	4413.798 ± 315.488
Augment + HEQ (SGD + CLAHE)	10.718 ± 0.749	14.900 ± 1.111	2976.320 ± 239.694	4313.142 ± 352.477

CPU demand remained similar across configurations: mean utilization ranged from 8.590% to 8.720%, while the combined configuration had the lowest CPU time (92.275 s; 207.358 ms/image). Memory differences were also small. Mean allocation ranged from 2976.320 MB for SGD + CLAHE to 3080.352 MB for SGD, with the combined configuration slightly below the 2982.696 MB baseline.

These measurements show a modest speed advantage for CLAHE-based pipelines without a resource penalty, but this gain is outweighed by their severe accuracy loss. Moreover, because the network architecture and input dimensions are unchanged, the small runtime differences should be interpreted as end-to-end pipeline and measurement effects, not as proof that contrast enhancement reduces neural computation. For ground imagery on Raspberry Pi, the baseline or SGD model is therefore preferable when detection quality is the primary requirement.

5.3.2. Ground Fire Detection Under Simulated Jetson Nano Constraints

Table 8 reports Jetson Nano wall-clock performance, while Tables 9 and 10 report CPU and memory behavior. Together, they show substantially higher throughput than the Raspberry Pi. The baseline achieved 76.514 FPS at 13.108 ms/image. SGD alone was marginally slower (75.784 FPS), CLAHE reached 78.202 FPS, and SGD + CLAHE reached 82.098 FPS at 12.254 ms/image. Relative to baseline, the combined configuration improved throughput by approximately 7.3% and reduced latency by 6.5%.

Table 8. Inference performance evaluation of YOLO v26 nano on Jetson Nano for Ground imagery.

Method	Wall Time (s)	Wall Time (ms/Image)	FPS
Baseline	5.834 ± 0.362	13.108 ± 0.813	76.514 ± 4.702
Augment (SGD)	5.902 ± 0.499	13.266 ± 1.121	75.784 ± 5.761
HEQ (CLAHE)	5.703 ± 0.312	12.814 ± 0.703	78.202 ± 4.060
Augment + HEQ (SGD + CLAHE)	5.453 ± 0.496	12.254 ± 1.117	82.098 ± 6.729

Table 9. CPU performance metrics of YOLO v26 nano on Jetson Nano for Ground imagery.

Method	Time (s)	ms/Image	Cycles (s)	Cycles/Image	Mean CPU (%)	Max CPU (%)
Baseline	9.319 ± 0.418	20.942 ± 0.940	9.320 ± 0.419	0.021 ± 0.001	5.538 ± 0.546	12.400 ± 2.300
Augment (SGD)	9.234 ± 0.471	20.748 ± 1.060	9.232 ± 0.469	0.021 ± 0.001	4.546 ± 0.165	8.580 ± 0.311
HEQ (CLAHE)	9.109 ± 0.352	20.470 ± 0.791	9.112 ± 0.353	0.020 ± 0.001	6.408 ± 2.466	23.580 ± 24.523
Augment + HEQ (SGD + CLAHE)	8.964 ± 0.569	20.142 ± 1.279	8.968 ± 0.563	0.020 ± 0.001	6.154 ± 0.135	16.020 ± 3.143

Table 10. Memory utilization of YOLO v26 nano on Jetson Nano for Ground imagery.

Method	Mean RAM (%)	Max RAM (%)	Mean RAM (MB)	Max RAM (MB)
Baseline	10.896 ± 0.562	11.500 ± 0.490	2952.454 ± 178.443	3144.826 ± 156.358
Augment (SGD)	10.930 ± 0.597	11.500 ± 0.663	2958.872 ± 189.775	3139.056 ± 211.261
HEQ (CLAHE)	10.806 ± 0.428	11.440 ± 0.336	2923.872 ± 140.423	3123.652 ± 106.179
Augment + HEQ (SGD + CLAHE)	10.846 ± 0.580	11.440 ± 0.467	2935.398 ± 186.615	3124.700 ± 159.449

The combined configuration also had the lowest CPU execution time (8.964 s; 20.142 ms/image), whereas SGD had the lowest mean CPU utilization (4.546%). CLAHE showed occasional CPU peaks, but mean memory remained tightly grouped between 2923.872 and 2958.872 MB. Thus, the speed gains did not require meaningful additional RAM.

For reviewers, the decisive trade-off is accuracy versus throughput: SGD + CLAHE is the fastest ground configuration, but its mAP_50–95 is only 0.192 compared with 0.325 for baseline. The Jetson platform supports real-time-class throughput for all configurations, so the baseline remains the defensible deployment choice unless latency is prioritized over detection quality. Physical-device validation and pipeline-level profiling are still required before attributing these timing differences to a specific preprocessing mechanism.

5.4. Performance on UAV Imagery (FASDD_UAV)

UAV imagery produced the strongest and most preprocessing-robust detection results (Figure 8; Table 11). The baseline led precision, F1-score, mAP_50, and mAP_50–95 with 0.880, 0.862, 0.896, and 0.544, while recall was 0.843. SGD, CLAHE, and SGD + CLAHE remained close, but none surpassed the baseline overall. The combined configuration attained the highest recall (0.845), yet its mAP_50 and mAP_50–95 decreased to 0.890 and 0.535. The practical implication is that preprocessing choice is less critical for UAV data than for ground or satellite data; the original image distribution already supports high classification and localization accuracy.

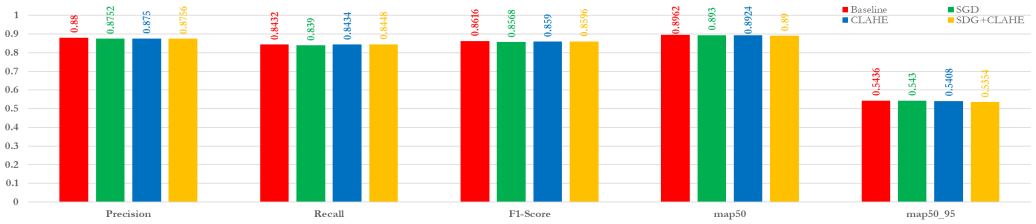


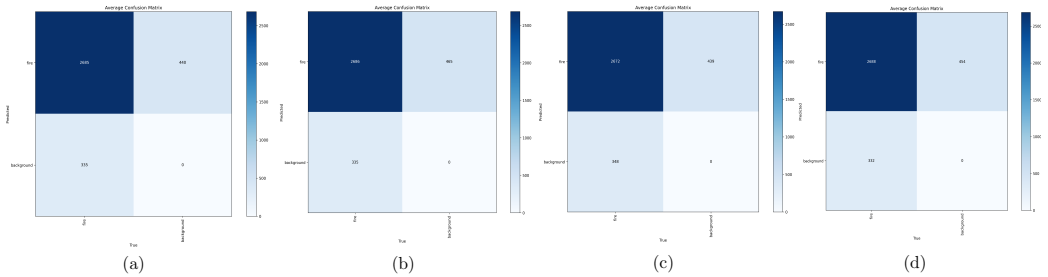
Figure 8. Mean precision, recall, F1-score, mAP_50, and mAP_50–95 after five-fold cross-validation of YOLOv26-nano on UAV imagery.

Table 11. Detection performance metrics for UAV imagery. Values are reported as mean ± standard deviation.

Method	Precision	Recall	F1-Score	mAP_50	mAP_50–95
Baseline	0.880 ± 0.007	0.843 ± 0.008	0.862 ± 0.004	0.896 ± 0.003	0.544 ± 0.006
Augment (SGD)	0.875 ± 0.008	0.839 ± 0.012	0.857 ± 0.007	0.893 ± 0.007	0.543 ± 0.008
HEQ (CLAHE)	0.875 ± 0.008	0.843 ± 0.008	0.859 ± 0.007	0.892 ± 0.007	0.541 ± 0.007
Augment + HEQ (SGD + CLAHE)	0.876 ± 0.011	0.845 ± 0.011	0.860 ± 0.004	0.890 ± 0.005	0.535 ± 0.007

Figure 9 shows similarly small differences. The baseline, SGD, CLAHE, and SGD + CLAHE models correctly identified 2685, 2686, 2672, and 2688 fire samples, respectively. Their fire-to-background errors were 440, 465, 439, and 454, while background-to-fire errors were 335, 335, 348, and 332. The combined method therefore produced the most correct fire predictions and fewest background-to-fire errors, but the improvement was marginal and did not translate into superior mAP. As in the ground dataset, all four matrices reported zero correctly classified background samples, which remains a dataset- or evaluation-level concern. Table 11 also shows low fold-to-fold dispersion across all methods.

The curves in Figure 10 are broad and well separated from the poorer ground profiles. Precision remains high over a wide threshold range, recall declines gradually, and all F1 curves have wide maxima. This threshold robustness strengthens the case for UAV imagery as the most reliable modality studied. The baseline is recommended for accuracy, while the combined method may be considered when slightly higher recall is operationally more important.



(a) (b) (c) (d)

Figure 9. Confusion matrices for UAV imagery: (**a**) Baseline, (**b**) Augment (SGD), (**c**) HEQ (CLAHE), and (**d**) Augment + HEQ (SGD + CLAHE).

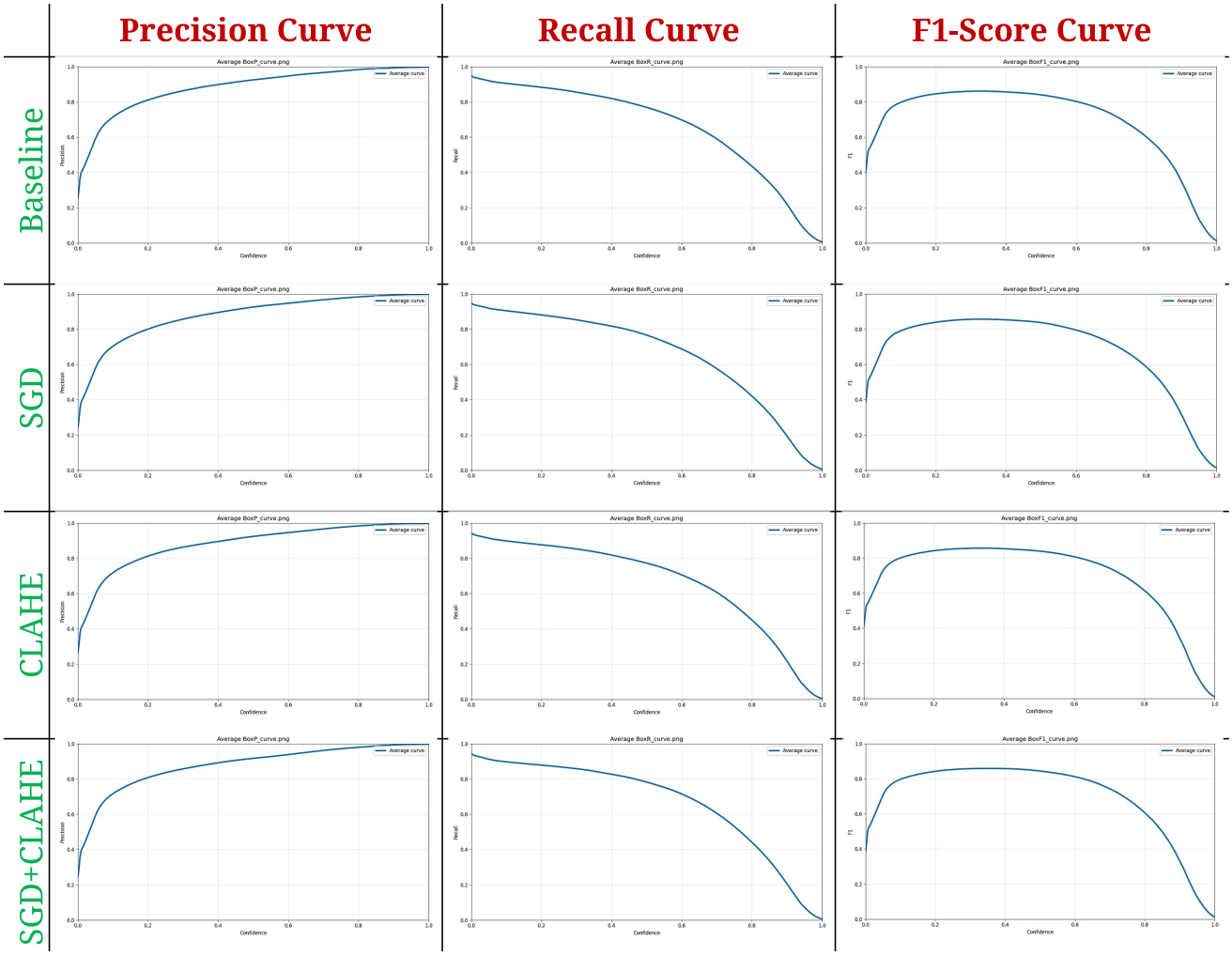


Figure 10. Precision, Recall, and F1-Score average curves for the UAV dataset.

5.4.1. UAV Fire Detection Under Simulated Raspberry Pi Constraints

For UAV inference on Raspberry Pi, Table 12 reports wall-clock performance, Table 13 reports CPU behavior, and Table 14 reports memory use. The baseline achieved 14.938 FPS at 66.976 ms/image. SGD increased throughput to 15.170 FPS, CLAHE to 15.526 FPS, and SGD + CLAHE to 15.572 FPS. The combined configuration therefore improved throughput by approximately 4.2% and reduced wall-clock latency by 4.0% relative to baseline.

Mean CPU utilization increased only from 8.172% for baseline to 8.388% for SGD + CLAHE. CLAHE had the lowest CPU time (75.290 s; 169.194 ms/image), whereas the combined method was slightly higher at 75.925 s. Mean RAM was lowest for baseline (2171.976 MB), highest for SGD (2281.432 MB), and near baseline for SGD + CLAHE (2185.750 MB).

Table 12. Inference performance evaluation of YOLO v26 nano on Raspberry Pi for UAV imagery.

Method	Wall Time (s)	Wall Time (ms/Image)	FPS
Baseline	29.805 ± 0.723	66.976 ± 1.625	14.938 ± 0.362
Augment (SGD)	29.336 ± 0.312	65.924 ± 0.701	15.170 ± 0.160
HEQ (CLAHE)	28.668 ± 0.478	64.422 ± 1.074	15.526 ± 0.260
Augment + HEQ (SGD + CLAHE)	28.599 ± 0.898	64.270 ± 2.017	15.572 ± 0.495

Table 13. CPU performance metrics of YOLO v26 nano on Raspberry Pi for UAV imagery.

Method	Time (s)	ms/Image	Cycles (s)	Cycles/Image	Mean CPU (%)	Max CPU (%)
Baseline	77.008 ± 1.879	173.052 ± 4.225	77.008 ± 1.881	0.173 ± 0.004	8.172 ± 0.086	12.160 ± 0.844
Augment (SGD)	76.787 ± 0.876	172.556 ± 1.969	76.786 ± 0.873	0.173 ± 0.002	8.246 ± 0.134	12.580 ± 0.986
HEQ (CLAHE)	75.290 ± 1.472	169.194 ± 3.308	75.290 ± 1.474	0.169 ± 0.003	8.296 ± 0.078	12.680 ± 2.235
Augment + HEQ (SGD + CLAHE)	75.925 ± 2.084	170.616 ± 4.682	75.924 ± 2.082	0.171 ± 0.005	8.388 ± 0.073	12.320 ± 1.472

Table 14. Memory utilization of YOLO v26 nano on Raspberry Pi for UAV imagery.

Method	Mean RAM (%)	Max RAM (%)	Mean RAM (MB)	Max RAM (MB)
Baseline	8.202 ± 0.327	9.440 ± 0.385	2171.976 ± 104.824	2566.602 ± 120.798
Augment (SGD)	8.542 ± 0.103	9.740 ± 0.182	2281.432 ± 33.695	2663.964 ± 66.313
HEQ (CLAHE)	8.448 ± 0.116	9.860 ± 0.351	2250.930 ± 38.359	2704.816 ± 112.799
Augment + HEQ (SGD + CLAHE)	8.248 ± 0.309	9.640 ± 0.305	2185.750 ± 97.717	2634.990 ± 90.426

Because detection accuracy was already similar across UAV configurations, these modest runtime differences can influence deployment choice more directly than in the ground case. SGD + CLAHE provides the highest wall-clock throughput with near-baseline memory, whereas CLAHE alone minimizes CPU time. Nevertheless, the baseline retains the best mAP values, so the final selection depends on whether localization accuracy or a roughly 4% throughput gain is more important.

5.4.2. UAV Fire Detection Under Simulated Jetson Nano Constraints

Table 15 reports Jetson Nano wall-clock performance, and Tables 16 and 17 report CPU and memory use. These measurements show the largest UAV speed gain from CLAHE. The baseline achieved 58.368 FPS, SGD 59.540 FPS, CLAHE 67.972 FPS, and SGD + CLAHE 66.984 FPS. CLAHE therefore increased throughput by approximately 16.5% and reduced latency from 17.182 to 14.732 ms/image; the combined method remained approximately 14.8% faster than baseline.

Table 15. Inference performance evaluation on Jetson Nano for UAV imagery.

Method	Wall Time (s)	Wall Time (ms/Image)	FPS
Baseline	7.646 ± 0.460	17.182 ± 1.033	58.368 ± 3.459
Augment (SGD)	7.491 ± 0.410	16.832 ± 0.921	59.540 ± 3.098
HEQ (CLAHE)	6.556 ± 0.278	14.732 ± 0.627	67.972 ± 2.845
Augment + HEQ (SGD + CLAHE)	6.656 ± 0.332	14.956 ± 0.742	66.984 ± 3.166

Table 16. CPU performance metrics during deployment on Jetson Nano for UAV imagery.

Method	Time (s)	ms/Image	Cycles (s)	Cycles/Image	Mean CPU (%)	Max CPU (%)
Baseline	10.942 ± 0.424	24.590 ± 0.954	10.942 ± 0.428	0.025 ± 0.001	5.372 ± 1.041	20.700 ± 16.704
Augment (SGD)	10.825 ± 0.447	24.324 ± 1.005	10.826 ± 0.447	0.024 ± 0.001	4.244 ± 0.071	8.200 ± 0.510
HEQ (CLAHE)	9.993 ± 0.298	22.458 ± 0.668	9.994 ± 0.293	0.022 ± 0.001	5.360 ± 0.536	13.960 ± 3.259
Augment + HEQ (SGD + CLAHE)	9.929 ± 0.380	22.314 ± 0.854	9.928 ± 0.381	0.022 ± 0.001	5.274 ± 0.572	13.440 ± 3.821

The combined configuration had the lowest CPU execution time (9.929 s; 22.314 ms/image), while SGD had the lowest mean CPU utilization (4.244%). Mean RAM ranged from 2782.308 MB for SGD to 2850.412 MB for CLAHE, so the speed improvements required less than a 3% increase over baseline memory.

Table 17. Memory utilization during deployment on Jetson Nano for UAV imagery.

Method	Mean RAM (%)	Max RAM (%)	Mean RAM (MB)	Max RAM (MB)
Baseline	10.396 ± 0.306	10.860 ± 0.358	2791.892 ± 94.397	2939.310 ± 113.065
Augment (SGD)	10.374 ± 0.347	10.860 ± 0.297	2782.308 ± 110.125	2936.212 ± 92.817
HEQ (CLAHE)	10.578 ± 0.265	10.880 ± 0.164	2850.412 ± 82.684	2945.172 ± 48.065
Augment + HEQ (SGD + CLAHE)	10.562 ± 0.341	10.980 ± 0.327	2842.186 ± 106.788	2974.814 ± 110.222

For UAV deployment, CLAHE-only offers the best wall-clock speed, while baseline offers the best detection and localization metrics. Since the accuracy differences are small, CLAHE is a credible throughput-oriented option on Jetson Nano. However, the unchanged network architecture again means that profiling should determine whether the gain arises from preprocessing, post-processing, detection counts, or system variability.

5.5. Performance on Satellite Imagery (FASDD_RS)

Satellite performance was intermediate between UAV and ground imagery (Figure 11; Table 18). SGD achieved the best classification balance, with precision 0.810, recall 0.779, and F1-score 0.793. The baseline, however, retained the best localization, with mAP_50 = 0.832 and mAP_50–95 = 0.515, compared with 0.828 and 0.507 for SGD. Thus, augmentation slightly improved classification but not bounding-box quality.

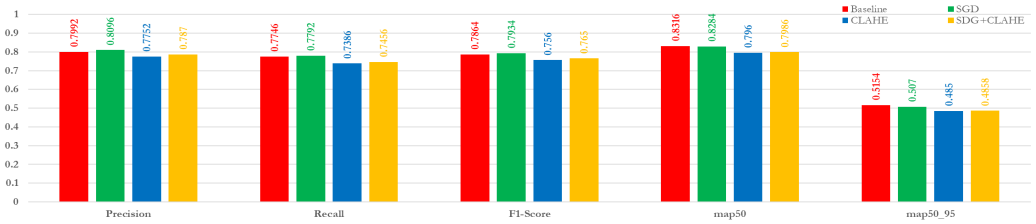


Figure 11. Mean precision, recall, F1-score, mAP_50, and mAP_50–95 after five-fold cross-validation of YOLOv26-nano on satellite imagery.

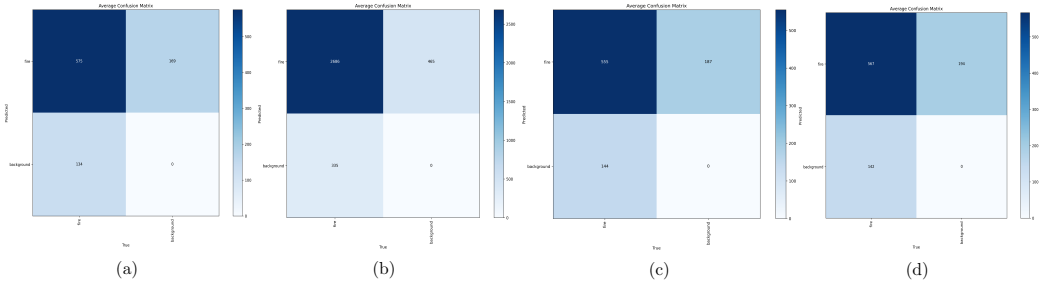
Table 18. Detection performance metrics for satellite imagery. Values are reported as mean ± standard deviation.

Method	Precision	Recall	F1-Score	mAP_50	mAP_50–95
Baseline	0.799 ± 0.039	0.775 ± 0.021	0.786 ± 0.025	0.832 ± 0.029	0.515 ± 0.026
Augment (SGD)	0.810 ± 0.032	0.779 ± 0.015	0.793 ± 0.016	0.828 ± 0.026	0.507 ± 0.023
HEQ (CLAHE)	0.775 ± 0.038	0.739 ± 0.020	0.756 ± 0.023	0.796 ± 0.020	0.485 ± 0.021
Augment + HEQ (SGD + CLAHE)	0.787 ± 0.040	0.746 ± 0.009	0.765 ± 0.020	0.799 ± 0.016	0.486 ± 0.019

CLAHE reduced all principal metrics: precision, recall, F1-score, mAP_50, and mAP_50–95 were 0.775, 0.739, 0.756, 0.796, and 0.485. SGD + CLAHE recovered precision to 0.787, recall to 0.746, and F1-score to 0.765, but localization remained below baseline (0.799 and 0.486). The result is a clear modality dependence: CLAHE was comparatively neutral for UAV accuracy, severely harmful for ground accuracy, and moderately harmful for satellite accuracy. A plausible interpretation is that local contrast enhancement distorts or amplifies low-resolution land-cover and atmospheric structures that overlap with fire cues; this mechanism should be presented as a hypothesis unless supported by feature-level analysis.

The confusion matrices in Figure 12 reinforce the advantage of SGD for classification. Baseline produced 575 correct fire predictions, 369 fire-to-background errors, and 134 background-to-fire errors. SGD improved correct fire predictions to 587 and reduced the corresponding errors to 190 and 123. CLAHE produced 555 correct fire predictions, 187 fire-to-background errors, and 144 background-to-fire errors; SGD + CLAHE produced 567, 194, and 142. All configurations again reported zero correctly classified background samples. Although SGD had the best confusion-matrix counts, the baseline remained

preferable under strict IoU criteria, illustrating why both classification and localization measures are required. The variability reported in Table 18 does not alter this ranking.



(a) (b) (c) (d)

Figure 12. Confusion matrices for satellite imagery: (**a**) Baseline, (**b**) Augment (SGD), (**c**) HEQ (CLAHE), and (**d**) Augment + HEQ (SGD + CLAHE).

Figure 13 shows that baseline and SGD maintain broader recall profiles and wider F1 maxima than the CLAHE variants. Satellite precision rises more gradually with confidence than in UAV imagery, indicating greater ambiguity at low and moderate thresholds. CLAHE and SGD + CLAHE are more threshold-sensitive and lose recall earlier. Consequently, the baseline is recommended when localization is central, while SGD is reasonable when classification balance is prioritized.

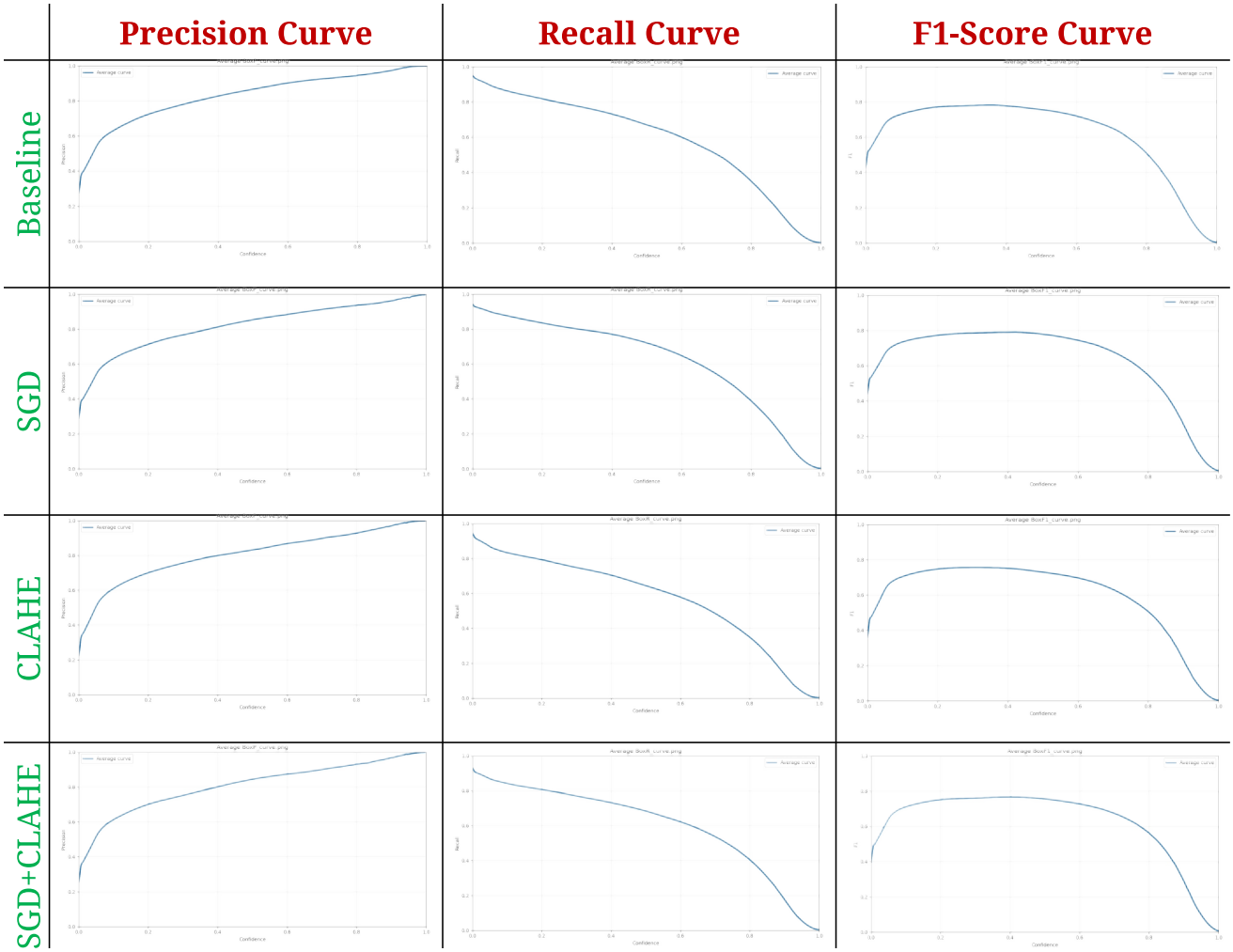


Figure 13. Precision, Recall, and F1-Score average curves for the satellite dataset.

5.5.1. Satellite Fire Detection Under Simulated Raspberry Pi Constraints

Satellite inference was the slowest Raspberry Pi case. Wall-clock, CPU, and memory results are reported in Table 19, Table 20, and Table 21, respectively. The baseline achieved 9.294 FPS at 107.602 ms/image, and SGD was nearly identical at 9.276 FPS. CLAHE reduced throughput to 8.700 FPS and increased latency to 114.960 ms/image; SGD + CLAHE was slowest at 8.652 FPS and 115.602 ms/image. Relative to baseline, CLAHE increased latency by approximately 6.8%, while the combined method increased it by 7.4%.

Table 19. Inference performance evaluation on Raspberry Pi for Satellite imagery.

Method	Wall Time (s)	Wall Time (ms/Image)	FPS
Baseline	47.884 ± 0.729	107.602 ± 1.635	9.294 ± 0.146
Augment (SGD)	47.985 ± 0.633	107.832 ± 1.423	9.276 ± 0.120
HEQ (CLAHE)	51.156 ± 0.419	114.960 ± 0.943	8.700 ± 0.070
Augment + HEQ (SGD + CLAHE)	51.443 ± 0.618	115.602 ± 1.389	8.652 ± 0.106

Table 20. CPU performance metrics during deployment on Raspberry Pi for Satellite imagery.

Method	Time (s)	ms/Image	Cycles (s)	Cycles/Image	Mean CPU (%)	Max CPU (%)
Baseline	134.752 ± 1.971	302.812 ± 4.429	134.750 ± 1.975	0.303 ± 0.004	8.864 ± 0.081	13.360 ± 1.698
Augment (SGD)	135.553 ± 1.806	304.612 ± 4.058	135.556 ± 1.808	0.305 ± 0.004	8.838 ± 0.050	11.740 ± 1.354
HEQ (CLAHE)	138.978 ± 1.423	312.310 ± 3.199	138.978 ± 1.425	0.312 ± 0.003	8.572 ± 0.070	13.300 ± 1.338
Augment + HEQ (SGD + CLAHE)	139.316 ± 1.562	313.070 ± 3.510	139.316 ± 1.560	0.313 ± 0.004	8.418 ± 0.038	10.300 ± 0.071

Table 21. Memory utilization during deployment on Raspberry Pi for Satellite imagery.

Method	Mean RAM (%)	Max RAM (%)	Mean RAM (MB)	Max RAM (MB)
Baseline	8.964 ± 0.146	11.100 ± 0.100	2415.394 ± 45.703	3098.774 ± 22.438
Augment (SGD)	9.116 ± 0.110	11.220 ± 0.148	2464.670 ± 35.066	3141.404 ± 44.026
HEQ (CLAHE)	8.866 ± 0.251	10.900 ± 0.394	2385.546 ± 79.799	3039.776 ± 127.082
Augment + HEQ (SGD + CLAHE)	8.826 ± 0.306	10.900 ± 0.453	2370.960 ± 98.179	3032.716 ± 140.293

CPU time increased from 134.752 s for baseline to 139.316 s for SGD + CLAHE, even though mean CPU utilization decreased from 8.864% to 8.418%. Memory showed the opposite pattern: SGD had the largest mean allocation (2464.670 MB), whereas SGD + CLAHE had the smallest (2370.960 MB). The lower CPU percentage and RAM footprint of the slower methods indicate that neither CPU saturation nor memory capacity explains the latency increase; preprocessing cost, post-processing, GPU/accelerator behavior, or other pipeline effects require direct profiling.

The baseline is therefore the strongest Raspberry Pi choice for satellite imagery because it combines the best runtime with the best localization accuracy. SGD is essentially equivalent in runtime and slightly better in classification, but CLAHE provides neither an accuracy nor a speed advantage.

5.5.2. Satellite Fire Detection Under Simulated Jetson Nano Constraints

Table 22, Table 23 and Table 24 report Jetson Nano wall-clock, CPU, and memory results, respectively. They show that baseline and SGD were again the only competitive configurations. Baseline achieved 65.532 FPS at 15.292 ms/image, while SGD was marginally faster at 65.948 FPS and 15.208 ms/image. CLAHE decreased throughput to 43.032 FPS and increased latency to 23.248 ms/image; SGD + CLAHE recovered slightly to 44.134 FPS and 22.694 ms/image. CLAHE therefore increased latency by approximately 52% and reduced throughput by approximately 34% relative to baseline.

Table 22. Inference performance evaluation on Jetson Nano for Satellite imagery.

Method	Wall Time (s)	Wall Time (ms/Image)	FPS
Baseline	6.805 ± 0.361	15.292 ± 0.811	65.532 ± 3.265
Augment (SGD)	6.767 ± 0.420	15.208 ± 0.942	65.948 ± 3.794
HEQ (CLAHE)	10.345 ± 0.217	23.248 ± 0.485	43.032 ± 0.884
Augment + HEQ (SGD + CLAHE)	10.098 ± 0.453	22.694 ± 1.017	44.134 ± 1.883

Table 23. CPU performance metrics during deployment on Jetson Nano for Satellite imagery.

Method	Time (s)	ms/Image	Cycles (s)	Cycles/Image	Mean CPU (%)	Max CPU (%)
Baseline	10.889 ± 0.389	24.470 ± 0.873	10.890 ± 0.386	0.024 ± 0.001	4.936 ± 0.555	9.500 ± 2.577
Augment (SGD)	10.696 ± 0.375	24.038 ± 0.844	10.694 ± 0.371	0.024 ± 0.001	4.886 ± 0.494	9.100 ± 3.085
HEQ (CLAHE)	14.434 ± 0.191	32.436 ± 0.427	14.436 ± 0.188	0.032 ± 0.000	4.208 ± 0.044	8.160 ± 0.270
Augment + HEQ (SGD + CLAHE)	14.225 ± 0.356	31.966 ± 0.803	14.224 ± 0.357	0.032 ± 0.001	5.532 ± 1.692	31.640 ± 32.006

Table 24. Memory utilization during deployment on Jetson Nano for Satellite imagery.

Method	Mean RAM (%)	Max RAM (%)	Mean RAM (MB)	Max RAM (MB)
Baseline	10.446 ± 0.393	11.220 ± 0.286	2813.832 ± 127.512	3065.374 ± 86.331
Augment (SGD)	10.420 ± 0.440	11.280 ± 0.497	2799.720 ± 139.964	3080.134 ± 161.431
HEQ (CLAHE)	10.214 ± 0.254	11.560 ± 0.261	2735.618 ± 81.121	3160.358 ± 82.578
Augment + HEQ (SGD + CLAHE)	10.292 ± 0.301	11.760 ± 0.182	2760.270 ± 93.728	3228.980 ± 53.356

The CLAHE model also required the longest CPU time (14.434 s) despite the lowest mean CPU utilization (4.208%). SGD had the lowest CPU time among the two accurate models (10.696 s), while SGD + CLAHE exhibited occasional CPU peaks. Mean RAM varied only from 2735.618 to 2813.832 MB; therefore, the large throughput loss cannot be explained by memory exhaustion.

For satellite imagery, SGD provides the best classification metrics and marginally highest Jetson throughput, whereas baseline provides the best localization metrics. Both are defensible; CLAHE-based configurations are not. More broadly, the cross-modality evidence shows that preprocessing must be validated per acquisition platform rather than assumed to generalize: CLAHE can improve measured throughput for UAV and ground pipelines, yet it degrades accuracy for ground and satellite data and sharply slows satellite inference. The deployment recommendation should therefore prioritize baseline/SGD models and treat CLAHE as a modality-specific option supported only for UAV throughput-oriented use.

6. Discussion

Next, we provide a cross-platform deployment summary and a discussion over the deployment on edge devices discussing its further limitations.

6.1. Cross-Platform Summary

Figure 14a–d compares the simulated deployment of YOLOv26-nano under Raspberry Pi and Jetson Nano resource constraints across all three sensing modalities, in terms of CPU utilization, RAM consumption, inference time, and FPS. CPU and RAM utilization remain largely stable across all preprocessing configurations. Under simulated Raspberry Pi constraints, mean CPU usage ranges from 8.17% to 8.86%, while the Jetson Nano stays below 5.54%, regardless of the enhancement strategy applied. Memory consumption follows a similar pattern: neither SGD nor CLAHE, individually or combined, meaningfully alters the runtime memory footprint, confirming that resource overhead is driven primarily by the hardware platform rather than the preprocessing choice.

More substantial differences emerge in inference time and throughput. Under simulated Raspberry Pi constraints, the baseline achieves 212.96 ms, 173.05 ms, and 302.81 ms per image for Ground, UAV, and Satellite imagery respectively, compared with 20.94 ms,

24.59 ms, and 24.47 ms under simulated Jetson Nano constraints. SGD optimization yields modest but consistent latency reductions across modalities. CLAHE produces more noticeable gains for Ground and UAV imagery, reducing Raspberry Pi latency to 207.93 ms and 169.19 ms respectively; however, it increases Satellite inference time, rising to 32.44 ms on the Jetson Nano, likely due to the additional texture complexity introduced by histogram equalization in large-scale scenes. The combined SGD + CLAHE configuration achieves the lowest latency for Ground imagery (207.36 ms/20.14 ms), but inherits the CLAHE-induced overhead for Satellite data. FPS results mirror these trends directly.

The exhibited results lead to the following observations. First, the Jetson Nano is the superior simulated edge platform, offering approximately one order of magnitude faster inference than the Raspberry Pi across all modalities. Second, SGD-based augmentation is the safest and most consistent preprocessing choice on both platforms, providing efficiency gains with negligible resource overhead. Third, preprocessing effectiveness is strongly modality-dependent: CLAHE benefits Ground and UAV pipelines but degrades Satellite inference speed and should not be applied uniformly. These findings confirm that edge hardware selection and preprocessing strategy must be treated as joint design decisions rather than independent choices.

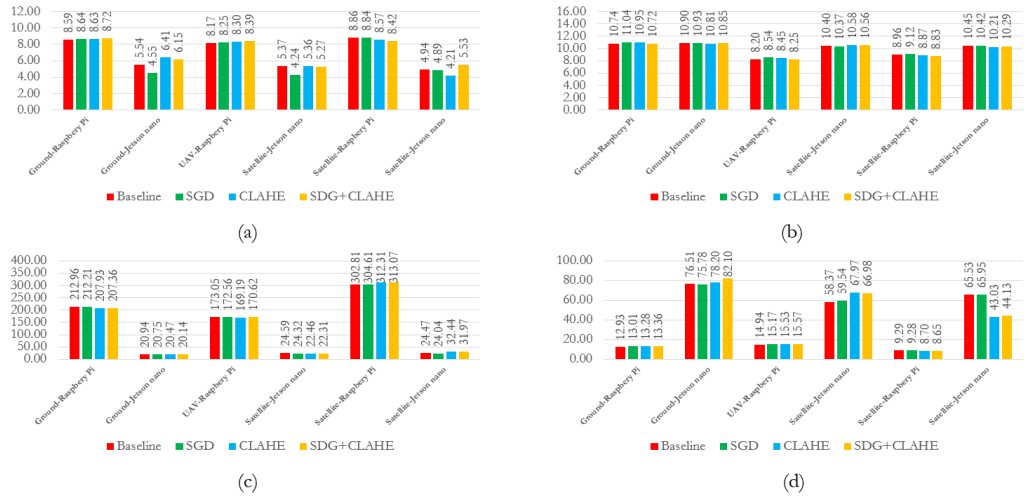


Figure 14. Cross-Platform and Cross-Dataset Deployment Comparison of YOLOv26-nano on Raspberry Pi and Jetson Nano, considering the mean CPU utilization (%) (**a**), the mean RAM utilization (%) (**b**), the exhibited inference time (ms per image) (**c**), and the achieved FPS (**d**).

6.2. Edge Device Deployment

The results of this study reinforce a key finding from our prior work: no single preprocessing strategy provides universally optimal performance across all sensing modalities. Data augmentation with SGD optimization is the most broadly beneficial strategy, improving detection across satellite, UAV, and ground-based imagery by increasing training set diversity and promoting robust feature learning. CLAHE offers additional gains specifically in low-contrast scenarios (satellite, high-altitude UAV), but introduces risks of feature distortion in high-contrast scenarios such as ground-based close-range observation.

The UAV modality occupies a middle ground between satellite and ground imaging, sharing characteristics of both: scale variation from altitude changes resembles the small-object challenge of satellite imagery, while the rich visual texture of low-altitude frames resembles the background complexity of ground-based scenes. This dual character makes the UAV subset the most demanding test of model generalization and the strongest argument for multi-platform benchmarking as a standard evaluation practice.

For operational deployment, the computational profile of YOLOv26-nano is well-matched to the resource constraints of drone-mounted and fixed-edge devices. Under simulated Jetson Nano-class constraints, inference latency below 10 ms per frame appears feasible, suggesting potential for high frame-rate detection (e.g., >100 FPS) pending physical hardware validation. A key operational advantage of the nano-scale YOLO architecture is its small memory footprint (typically under 10 MB of model storage), enabling deployment even on microcontroller-class devices with limited onboard storage and RAM. In drone applications, this supports fully autonomous onboard inference without dependency on ground-station connectivity, critical for operations in remote or mountainous terrain where wireless bandwidth may be unreliable.

6.3. Comparison with Related Approaches

The comprehensive benchmark presented in Table 25 provides a detailed comparison between classical object detection frameworks, transformer-based architectures, modern YOLO variants, and the proposed YOLOv26 nano models across ground, UAV, and satellite scenarios. The first column cites the literature where the results were reported; the second column depicts the type of the modality of the dataset; the third column presents the deployed algorithm; the next five columns show the performance metrics; the next column shows the inference time (per image); the tenth column shows the exhibited FPS; and the eleventh column depicts the required FLOPs. The last 12 rows of the table depict the results exhibited in this work, while the third column next to the algorithm denotes the corresponding preprocessing techniques, i.e., baseline (B), HEQ (CLAHE) (C), Augmentation with SGD optimization (S), and their combination (CLAHE+SGD) (C/S).

The results suggest that, under the simulated constraints, the proposed approach achieves a competitive balance between accuracy, robustness, and computational efficiency compared to existing state-of-the-art methods. It should be noted that the results summarized in Table 24 originate from different studies employing distinct dataset partitions, annotation protocols, training procedures, hyperparameter configurations, and evaluation settings. Consequently, the reported values should be interpreted as indicative comparisons rather than direct performance rankings. A fully controlled benchmark would require retraining all methods under identical experimental conditions.

As observed in Table 25, two-stage detectors such as Faster R-CNN and Dynamic R-CNN achieve competitive accuracy in controlled ground scenarios; however, they suffer from high computational cost and limited suitability for real-time applications. Transformer-based models such as Swin Transformer, RT-DETR, and DINO-4scale provide improved contextual reasoning capabilities, but this advantage comes at the expense of extremely high computational complexity and inconsistent precision-recall behavior across different datasets and sensing conditions. In particular, these models often exhibit either high recall with low precision or strong performance degradation when applied to satellite imagery.

In contrast, YOLO-based methods consistently demonstrate a better trade-off between accuracy and inference efficiency. Nevertheless, earlier YOLO versions (including YOLOv5 through YOLOv12 and YOLO11n) show clear limitations in maintaining stable performance across different sensing domains, particularly under satellite conditions where domain shift significantly affects detection quality. This highlights a persistent challenge in existing literature: achieving robustness across heterogeneous data sources while preserving real-time performance. Although Table 24 includes published YOLOv11 results for reference, direct performance comparisons with YOLOv26-nano should be interpreted cautiously because the models were not evaluated under identical experimental conditions. A controlled benchmark involving retraining and testing both architectures using the same

data partitions and training protocol would be required to determine whether the observed differences are statistically significant.

Table 25. Comprehensive benchmark comparison on FASDD dataset of various fire recognition approaches.

Ref.	FASDD	Algorithm	Prec.	Rec.	F1	mAP_50	mAP_50–95	Inf/img (s)	FPS	FLOPs
[23]	N/R	YOLO	–	–	–	–	–	0.11	–	–
[24]	CV	YOLOX	0.725	0.628	–	–	–	–	232.081	–
[25]	CV	YOLOv5x	–	–	–	0.841	–	–	–	–
[25]	RS	YOLOv5x	–	–	–	0.444	–	–	–	–
[25]	N/R	YOLOv5x	–	–	–	0.792	–	–	–	–
[26]	N/R	YOLOv5n	0.700	–	–	0.793	–	–	6.1	5.1 B
[27]	RS	YOLOv7	0.602	0.687	–	0.652	–	–	–	–
[26]	N/R	YOLOv8n	0.596	–	–	0.768	–	–	3.3	8.7 B
[27]	RS	YOLOv8	0.644	0.607	–	0.617	–	–	–	–
[27]	UAV	YOLOv8	0.894	0.874	–	0.923	–	–	–	–
[26]	N/R	YOLO11n	0.639	–	–	0.775	–	–	3.2	6.5 B
[28]	CV	YOLOv11 (baseline)	–	–	–	0.870	0.632	–	–	21.3 T
[28]	CV	YOLOv11 P5 + P6	–	–	–	0.906	0.655	–	–	22.5 T
[28]	CV	YOLOv11 + P5(C5Go)	–	–	–	0.890	0.655	–	–	21.8 T
[28]	CV	YOLOv11 + P6(C3K2)	–	–	–	0.902	0.653	–	–	22.3 T
[28]	CV	YOLOv11	–	–	–	0.314	0.141	–	–	22.5 T
[28]	RS	YOLOv11 (baseline)	–	–	–	0.291	0.127	–	–	21.3 T
[27]	RS	YOLOv12	0.712	0.639	–	0.656	–	–	–	–
[24]	CV	YOLO26 + SCHead + MCCM	0.914	0.837	–	–	–	–	–	–
[24]	CV	YOLO26 + SCHead	0.914	0.811	–	–	–	–	–	–
[24]	CV	YOLO26 + MCCM	0.900	0.798	–	–	–	–	–	–
[24]	CV	YOLO26	0.919	0.815	–	–	–	–	1195.56	–
[23]	N/R	YOLO + SSD	–	–	–	–	–	0.17	–	–
[27]	RS	Swin Transformer	0.294	0.875	–	0.699	–	–	–	–
[25]	CV	Swin Transformer	–	–	–	0.788	–	–	–	–
[25]	RS	Swin Transformer	–	–	–	0.533	–	–	–	–
[25]	N/R	Swin Transformer	–	–	–	0.732	–	–	–	–
[24]	CV	STCNet	0.582	0.654	–	–	–	–	132.568	–
[24]	CV	SSmokeDet	0.736	0.712	–	–	–	–	204.635	–
[24]	CV	SSD	0.650	0.697	–	–	–	–	38.375	–
[23]	N/R	SSD	–	–	–	–	–	0.30	–	–
[26]	N/R	RTM-DET Tiny	0.692	–	–	0.784	–	–	–	8.03 B
[28]	CV	RT-DETR	–	–	–	0.641	0.407	–	–	103.4 T
[28]	CV	RT-DETR (tuned)	–	–	–	0.895	0.648	–	–	103.4 T
[27]	RS	RT-DETR	0.683	0.602	–	0.603	–	–	–	–
[24]	CV	MCCS-Det	0.914	0.837	–	–	–	–	1143.52	–
[27]	RS	HybriDet (pruned)	0.721	0.594	–	0.681	–	–	–	–
[27]	UAV	HybriDet (pruned)	0.903	0.880	–	0.925	–	–	–	–
[27]	RS	HybriDet (original)	0.722	0.626	–	0.666	–	–	–	–
[27]	UAV	HybriDet (original)	0.898	0.879	–	0.923	–	–	–	–
[25]	CV	GFL	–	–	–	0.729	–	–	–	–
[25]	RS	GFL	–	–	–	0.406	–	–	–	–
[25]	N/R	GFL	–	–	–	0.682	–	–	–	–
[25]	CV	Faster R-CNN	–	–	–	0.613	–	–	–	–
[25]	RS	Faster R-CNN	–	–	–	0.312	–	–	–	–
[25]	N/R	Faster R-CNN	–	–	–	0.552	–	–	–	–
[24]	CV	Faster R-CNN	0.599	0.662	–	–	–	–	7.790	–
[23]	N/R	Faster R-CNN	–	–	–	–	–	0.52	–	–
[26]	N/R	Dynamic-RCNN	0.648	–	–	0.749	–	–	–	187 B
[26]	N/R	DINO-4scale	0.695	–	–	0.791	–	–	–	249 B
Ours	CV	YOLO v26 nano -B	0.687	0.611	0.647	0.641	0.325	0.077–0.012	12.93–76.51	5 T
	CV	YOLO v26 nano -C	0.686	0.606	0.644	0.634	0.320	0.077–0.013	13.01–75.78	5 T
	CV	YOLO v26 nano -S	0.573	0.410	0.477	0.415	0.188	0.075–0.013	13.28–78.20	5 T
	CV	YOLO v26 nano -C/S	0.608	0.411	0.490	0.416	0.192	0.075–0.012	13.36–82.10	5 T
Ours	UAV	YOLO v26 nano -B	0.880	0.843	0.862	0.896	0.544	0.067–0.017	14.94–58.37	5 T
	UAV	YOLO v26 nano -C	0.875	0.839	0.857	0.893	0.543	0.066–0.017	15.17–59.54	5 T
	UAV	YOLO v26 nano -S	0.875	0.843	0.859	0.892	0.541	0.064–0.015	15.53–67.97	5 T
	UAV	YOLO v26 nano -C/S	0.876	0.845	0.860	0.890	0.535	0.064–0.015	15.57–66.98	5 T
Ours	RS	YOLO v26 nano -B	0.799	0.775	0.786	0.832	0.515	0.116–0.015	9.29–65.53	5 T
	RS	YOLO v26 nano -C	0.810	0.779	0.793	0.828	0.507	0.108–0.015	9.28–65.95	5 T
	RS	YOLO v26 nano -S	0.775	0.739	0.756	0.796	0.485	0.115–0.023	8.70–43.03	5 T
	RS	YOLO v26 nano -C/S	0.787	0.746	0.765	0.799	0.486	0.116–0.023	8.65–44.13	5 T

The proposed YOLOv26 nano variants (B, C, S, and C/S configurations) directly address these limitations by providing a more balanced and domain-robust detection framework. As shown in Table 25, the proposed models consistently achieve strong performance across all sensing modalities. In UAV scenarios, the models reach their best performance, with F1-scores up to 0.862 and mAP50 values reaching 0.896, demonstrating strong adaptability to aerial viewpoints. Ground-based configurations also maintain stable and competitive results, while satellite-based configurations are favored with several baseline approaches in terms of robustness and consistency, despite the inherent difficulty of this domain, bearing in mind that these comparisons are indicative only, as methods were evaluated under different experimental conditions.

Another key contribution of this work is computational efficiency. Unlike transformer-based architectures and two-stage detectors, which require substantial computational resources, the proposed YOLOv26 nano models operate with extremely low computational cost (approximately 5 FLOPs) while maintaining competitive detection performance. This makes them particularly suitable for real-time deployment on edge devices, UAV platforms, and other resource-constrained environments where latency and efficiency are critical constraints.

The results reported in Table 25 demonstrate that the the proposed method demonstrates a competitive balance between accuracy, efficiency, and robustness. Most importantly, it significantly reduces the impact of domain shift across ground, UAV, and satellite data, which remains one of the most challenging problems in remote sensing object detection. These findings confirm that the proposed YOLOv26 nano framework is a strong and practical solution for real-world multi-domain detection tasks, and shows reduced sensitivity to domain shift compared to several methods in the literature, though controlled head-to-head evaluation would be required to confirm these findings.

6.4. Limitations

Several limitations of the present study should be acknowledged. With respect to dataset scale, the representative subsets extracted from FASDD may constrain the generalization capacity of the trained models to highly diverse geographic regions or unusual fire behavior patterns not well represented in the sampled data, and evaluation on broader, more geographically heterogeneous datasets would be necessary to confirm the robustness of the reported findings. Regarding single visual modality, this study focuses exclusively on RGB visual imagery, and the integration of thermal infrared or SWIR channels could substantially improve detection performance under conditions where visual-spectrum contrast is insufficient, such as nighttime fires or scenes obscured by dense smoke, representing a meaningful gap between the current system and a fully operational fire detection pipeline.

A limitation of the present study is that the edge-computing evaluation was performed under simulated Raspberry Pi and Jetson Nano resource constraints rather than on physical hardware platforms. Although the obtained results provide useful insights into the computational feasibility of deploying YOLOv26-nano in resource-constrained environments, actual field performance may be influenced by additional factors including device-specific optimizations, thermal throttling, power-management policies, operating-system overhead, and environmental conditions. Future work will therefore focus on validating the proposed framework on physical edge devices deployed in realistic wildfire-monitoring scenarios.

Concerning simulated edge deployment, the inference latency estimates reported here are based on resource-limited simulation and TensorRT benchmarks rather than physical hardware measurements, and validation on actual drone hardware under representative field conditions is required to confirm that real-world performance aligns with the benchmarked figures, particularly under variable thermal, vibration, and power supply conditions typical of UAV deployments. Finally, with regard to static preprocessing, the preprocessing strategies evaluated in this study are applied uniformly across all images without any adaptation to image-level characteristics; an adaptive preprocessing approach that dynamically adjusts CLAHE parameters based on estimated image contrast, atmospheric conditions, or acquisition altitude represents a promising direction for future work that could address the context-dependent performance degradation observed for CLAHE-based configurations in this study.

7. Conclusions

In this work we presented a multi-platform benchmarking study of YOLOv26-nano for forest fire detection using the FASDD dataset, encompassing satellite, UAV, and ground-based sensing modalities under a unified five-fold cross-validation protocol. The study evaluated the impact of CLAHE and SGD-coupled data augmentation on fire detection performance across all different platforms, with dedicated analysis of the specific challenges posed by each imaging modality (i.e., ground-based cameras, UAV cameras, and satellite sensors).

The principal findings are as follows. Data augmentation with SGD optimization consistently improves detection performance across all sensing modalities. CLAHE provides additional gains in low-contrast scenarios, notably satellite and high-altitude UAV imagery, but must be applied judiciously in high-contrast ground-level scenarios. YOLOv26-nano achieves real-time inference performance on edge-class hardware, supporting deployment on drone onboard processors and fixed-edge monitoring stations. UAV-based fire detection presents a unique combination of scale variation, motion blur, perspective diversity, and illumination instability; detection-aware cropping and altitude-stratified evaluation provide practical tools for characterizing and improving performance in this modality.

Future Directions

Several promising directions emerge from this study for future investigation. Regarding multimodal fusion, combining visual-spectrum detection with thermal infrared or SWIR data could substantially improve fire detection reliability under low-visibility conditions, including nighttime operations and dense smoke environments where RGB-only models are inherently limited. With respect to adaptive preprocessing, developing dynamic preprocessing pipelines that automatically select and parameterize CLAHE and augmentation strategies based on estimated input image statistics, such as contrast, noise level, and acquisition altitude, could improve cross-condition robustness and eliminate the need for manual, dataset-specific preprocessing tuning. In terms of temporal modeling, exploiting the temporal continuity available in video streams from both UAV and ground-based cameras through recurrent or transformer-based architectures to model smoke dynamics across consecutive frames could significantly reduce false alarms arising from transient confounders such as clouds, dust, or specular reflections. Concerning domain adaptation, developing unsupervised or semi-supervised techniques to transfer models trained on one sensing platform to another without full re-annotation represents a critical scalability challenge for operational multi-platform fire detection systems, where annotated data for every deployment modality may not always be available. Finally, regarding field validation, the deployment and real-world evaluation of the trained models on commercial drone hardware in controlled prescribed fire scenarios would provide ground-truth confirmation of the edge deployment performance estimates reported in this study, bridging the gap between laboratory benchmarking and operational readiness.

Author Contributions: Conceptualization, I.P., C.S. and I.K.; methodology, I.P., C.S. and I.K.; software, I.P. and C.S.; validation, I.P., C.S. and I.K.; formal analysis, I.P., C.S. and I.K.; investigation, I.P. and C.S.; resources, I.K.; data curation, I.P. and C.S.; writing—original draft preparation, I.P., C.S. and I.K.; writing—review and editing, I.P., C.S. and I.K.; supervision, I.P. and I.K.; project administration, I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The FASDD dataset is available in [5]. Additional code, trained weights, and experimental splits are available in GitHub [29].

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

References

1. 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]
2. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. Available online: https://arxiv.org/abs/2004.10934 (accessed on 20 May 2026).
3. 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]
4. Sapkota, R.; Cheppally, R.H.; Sharda, A.; Karkee, M. YOLO26: Key architectural enhancements and performance benchmarking for real-time object detection. *arXiv* **2025**, arXiv:2509.25164. Available online: https://arxiv.org/pdf/2509.25164 (accessed on 20 May 2026).
5. Wang, M.; Yue, P.; Jiang, L.; Yu, D.; Tuo, T.; Li, J. An open flame and smoke detection dataset for deep learning in remote sensing based fire detection. *Geo-Spat. Inf. Sci.* **2025**, *28*, 511–526. Available online: https://doi.org/10.1080/10095020.2024.2347922 (accessed on 24 June 2026).
6. Polenakis, I.; Sarantidis, C.; Karydis, I.; Avlonitis, M. Smoke Detection on the Edge: A Comparative Study of YOLO Variants. *Signals* **2025**, *6*, 60. [CrossRef]
7. Polenakis, I.; Sarantidis, C.; Karydis, I.; Avlonitis, M. Comparing the Effectiveness of YOLOv12 and YOLOv13 in Forest Fire Recognition based on Satellite Imagery. In *Artificial Intelligence and Mathematics*; Springer Nature: Cham, Switzerland, 2026; *in press*.
8. Reza, A.M. Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. *J. Vlsi Signal Process. Syst. Signal Image Video Technol.* **2004**, *38*, 35–44. [CrossRef]
9. Wang, Y.; Piao, Y.; Wang, H.; Zhang, H.; Li, B. An improved forest smoke detection model based on YOLOv8. *Forests* **2024**, *15*, 409. [CrossRef]
10. Alkhammash, E.H. A Comparative Analysis of YOLOv9, YOLOv10, YOLOv11 for Smoke and Fire Detection. *Fire* **2025**, *8*, 26. [CrossRef]
11. 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]
12. Pan, W.; Wang, X.; Huan, W. EFA-YOLO: An efficient feature attention model for fire and flame detection. *arXiv* **2024**, arXiv:2409.12635. Available online: https://arxiv.org/abs/2409.12635 (accessed on 20 May 2026).
13. Chuvieco, E.; Aguado, I.; Salas, J.; García, M.; Yebra, M.; Oliva, P. Satellite Remote Sensing Contributions to Wildland Fire Science and Management. *Curr. For. Rep.* **2020**, *6*, 81–96. Available online: https://link.springer.com/article/10.1007/s40725-020-00116-5 (accessed on 20 May 2026). [CrossRef]
14. Rashkovetsky, D.; Mauracher, F.; Langer, M.; Schmitt, M. Wildfire detection from multisensor satellite imagery using deep semantic segmentation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 7001–7016. Available online: https://ieeexplore.ieee.org/abstract/document/9468971 (accessed on 20 May 2026). [CrossRef]
15. Hong, Z.; Tang, Z.; Pan, H.; Zhang, Y.; Zheng, Z.; Zhou, R.; Ma, Z.; Zhang, Y.; Han, Y.; Wang, J.; et al. Active Fire Detection Using a Novel CNN Based on Himawari-8 Satellite Images. *Front. Environ. Sci.* **2022**, *10*, 794028. [CrossRef]
16. Wang, D.; Qian, Y.; Lu, J.; Wang, P.; Yang, D.; Yan, T. EA-YOLO: An efficient extraction and aggregation mechanism of YOLO for fire detection. *Multimed. Syst.* **2024**, *30*, 287. [CrossRef]
17. 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. Available online: https://link.springer.com/chapter/10.1007/978-981-19-5096-4_20 (accessed on 20 May 2026).
18. 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. Available online: https://ieeexplore.ieee.org/abstract/document/9544351/ (accessed on 20 May 2026).
19. Wang, J.; Yao, Y.; Huo, Y.; Guan, J. FM-Net: A New Method for Detecting Smoke and Flames. *Sensors* **2025**, *25*, 5597. [CrossRef] [PubMed]
20. Wang, C.; Xu, C.; Akram, A.; Wang, Z.; Shan, Z.; Zhang, Q. Wildfire smoke detection system: Model architecture, training mechanism, and dataset. *Int. J. Intell. Syst.* **2025**, *2025*, 1610145. [CrossRef]
21. Tian, Y. YOLOv12: Attention-Centric Real-Time Object Detectors. GitHub Repository. 2025. Available online: https://github.com/sunsmarterjie/yolov12 (accessed on 20 May 2026).
22. 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. Available online: https://arxiv.org/abs/2506.17733 (accessed on 20 May 2026).

23. Alpar, S.; Adilbayeva, M.; Karashbayeva, Z.; Alpar, S.; Adilbayeva, M.; Karashbayeva, Z. Comparative results of using deep learning models with ensemble methods for wildfire assessment. *Sci. J. Astana Univ.* **2025** , 23, 231–242. [CrossRef]
24. Xiong, Y.; Liu, X.; Ma, X.; Kuang, H. MCCS-Det: A Small-Scale Smoke Detection with YOLO26 Enhanced by Multi-Scale Context Compensation and a Self-Collaborative Head. In Proceedings of the 2026 9th International Conference on Advanced Algorithms and Control Engineering (ICAACE), Jinan, China, 20–22 March 2026 ; pp. 1233–1236. [CrossRef]
25. Wang, M.; Jiang, L.; Yue, P.; Yu, D.; Tuo, T. FASDD: An open-access 100,000-level flame and smoke detection dataset for deep learning in fire detection. *Earth Syst. Sci. Data Discuss.* **2023**, 2023, 1–26. [CrossRef]
26. Vazquez, G.; Zhai, S.; Yang, M. Edge-Friendly UAV Wildfire Smoke and Flame Detection Using Transfer Learning-Enhanced Lightweight Deep Learning Models. *Sensors* **2026**, *26*, 3197. [CrossRef] [PubMed]
27. Dong, F.; Wang, M. HybriDet: A Hybrid Neural Network Combining CNN and Transformer for Wildfire Detection in Remote Sensing Imagery. *Remote. Sens.* **2025**, *17*, 3497. [CrossRef]
28. Suh, S. Deep Learning-Based Fire Detector Robust to Smoke–Fog Ambiguity in Outdoor Scenes. *Appl. Sci.* **2026**, *16*, 1963. [CrossRef]
29. Fire Recognition YOLO Versions Comparison. Available online: https://github.com/ChristoSarantidis/Benchmarking_Next_Generation_YOLO_Architectures_for_Multi_Platform_Forest_Fire_Recognition (accessed on 20 May 2026).