Research Article *Open Access*

Journal of Pioneering Artificial Intelligence Research

ISSN: 3069-0846

DOI: doi.org/10.63721/26JPAIR0143

A YOLO 26-Based Wildfire Flame Detection Software System: A Review, Technical Framework, and Application Prospect

Jinhao Liu* and Tongshu Wei

Independent Researcher, China

Citation: *Jinhao Liu, Tongshu Wei (2026) A YOLO26-Based Wildfire Flame Detection Software System: A Review, Technical Framework, and Application Prospect. J of Poin Artf Research 2(4), 1-10 WMJ-JPAIR-143*

Abstract

Early warning of forest, grassland, and open-area urban fires relies on detecting flames or smoke at the initial stage of a fire. Accurate and efficient identification of early-stage fires is crucial for reducing casualties, protecting property, and minimizing ecological damage. Today, computer vision and deep learning technologies are the preferred tools for achieving this function. Among many models, the YOLO family is particularly popular because it can perform image and video classification and analysis efficiently and in real time, and its relatively low deployment requirements allow for rapid promotion in engineering applications. As noted by Ultralytics, YOLO26 goes further by providing a truly end-to-end detection without the need for an external NMS step. It also abandons Distribution Focal Loss in favor of Progressive Loss, while introducing Small-Target-Aware Label Assignment and MuSGD, and offers better support for running on mainstream devices. These functions meet the monitoring requirements for various wildfire and open-area scenarios. This paper is a review and system framework study. It utilizes the official YOLO26 description, representative YOLO literature, and previous research on flame, smoke, and wildfire detection to propose a YOLO26-based wildfire flame detection software framework. The paper reviews related work, describes the characteristics of YOLO26, provides a qualitative comparison of early and mainstream versions including YOLOv5, YOLOv7, YOLOv8, YOLOv9, YOLOv10, and YOLO11, and outlines the system architecture, preliminary use scenarios, and validation steps.We still need to extensively use public datasets, forest monitoring videos, drone images, and experiments targeting specific scenarios for testing, ultimately providing a complete conclusion.

***Corresponding author**: Jinhao Liu, Independent Researcher, China.

Submitted: 25.06.2026 **Accepted**: 29.06.2026 **Published**: 08.07.2026

Keywords: YOLO26, Flame Detection, Wildfire Monitoring, Edge Computing, Intelligent Early Warning

Introduction

The scale of wildfires expands very rapidly, capable of developing to an uncontrollable magnitude in a short period of time, thereby increasing the difficulty of firefighting. Unlike indoor fires or industrial fires, wildfires involve a large amount of combustible material and cover a wide area. Therefore, the rapid identification of early flames and smoke is an important task and challenge in fire prevention and emergency management. This need is equally important in forests, scenic areas, and pastoral regions. Fire monitoring typically integrates multiple methods. Routine patrols, fire lookout towers, smoke/temperature sensors, thermal imaging technology, satellite remote sensing, and rule-based image analysis are widely employed. Human observers can quickly and efficiently assess situations, but continuous, large-scale patrolling is costly and challenging due to terrain, weather, and other adverse factors; fixed smoke and temperature sensors are difficult to deploy over large areas and require stable power and communication connections. Satellites offer broad coverage, but due to cloud interference, resolution limitations, and vertical spatial constraints, early fires can easily be missed. Rule-based visual methods are simple to implement, but shadows, glare, haze, clouds, red objects, vegetation, and changes in lighting often interfere with their operation, and strictly following rules written by humans often results in numerous false alarms [1-11].

Nowadays, deep learning technology has greatly transformed the technical foundation of visual fire detection. It no longer relies solely on manually set rules for color, motion, and image features, but instead learns these feature rules autonomously from a large number of images labeled with smoke and fire. Convolutional neural networks (CNNs) can represent visual features at multiple levels. Object detectors can also provide class labels and locations. The YOLO series has been widely used in the field of fire detection, as the image classification and real-time recognition capabilities of this series of models are highly compatible with fire detection tasks. In previous models, YOLOv1 treated detection as a regression problem, while YOLOv2/YOLO9000 and YOLOv3 expanded recognition capabilities and multi-scale prediction. YOLOv4 and YOLOv7 improved the balance between speed and accuracy in real-time applications. YOLOv9 introduced programmable gradient information, while YOLOv10 reduced reliance on non-maximum suppression [5, 7-9]. Ultralytics versions, such as YOLOv5 and YOLOv8, are also very important because they provide the exact workflows for training, inference, exporting, and deployment, which helps YOLO be widely used in the field of fire detection. YOLO11 and YOLO26 follow the same approach [10, 11]. This explains why YOLO detectors are widely used in research on flames, smoke and forest fires [12-15].

Some challenges present in wildfire images may be less significant in general image detection techniques. In the early stages of a fire, visible flames may occupy only a few pixels in the image, smoke boundaries are blurred, and they change continuously between frames. Terrain or vegetation may obscure these small targets. In areas outside wildfire detection, sunlight, lighting, red signs, compression artifacts, lens smudges, and camera shake can further increase the uncertainty of image recognition. Additionally, certain hardware limitations, such as latency, camera pixel constraints, network delays, and other factors, may also impact detection efficiency [16, 17].

The YOLO26 wildfire detection system proposed in this context can serve as a general reference framework. It is not presented as a fully mature product, but rather provides a conceptual approach. This framework includes image detection, video detection, real-time camera input, result display, alert notifications, log storage, model weight settings, threshold settings, as well as subsequent integration with devices such as drones and forest cameras. Its aim is to offer a concept for further application development, rather than to create a highly mature system ready for large-scale deployment.

Related Work

Traditional Flame and Wildfire Detection Methods

Traditional wildfire detection can generally be divided into four main technical methods: sensor monitoring, remote sensing, manual patrol, and traditional image processing. Sensor systems monitor data such as smoke concentration (optical density), temperature, humidity, specific gas concentrations, or infrared radiation, making them suitable for indoor locations and fixed high-risk areas. However, deployment at the forest scale is very difficult, as installation, power supply, and periodic maintenance are hard to manage. Satellite and aerial remote sensing provide regional views, but

limitations in resolution, cloud interference, and the small visible size of early flames remain significant constraints [15,16].

Some detection methods relying on human observers are equally important. Observers can leverage their experience when judging uncertain visual signals, often achieving high efficiency and accuracy. However, this approach also has obvious limitations, including fatigue, weather, terrain, visibility, and labor costs. Traditional image processing relies on features such as flame color, brightness, texture, edges, motion, flicker, or smoke diffusion. It is lightweight and easy to interpret, but manually set parameters often fail when detecting rare situations.

Deep Learning-Based Flame and Smoke Detection

Deep learning uses annotated data to learn visual features rather than applying fixed rules. Convolutional Neural Network (CNN) classifiers can determine whether suspected flame or smoke targets appear in an image. Detectors such as Faster R-CNN, SSD, and YOLO can also locate candidate regions [18]. Two-stage detectors may offer strong localization capabilities in certain cases. One-stage detectors are generally more practical for real-time video due to their shorter inference paths, which is a natural advantage of YOLO [19].

Localization is useful beyond display. In a fire-monitoring system, the predicted box, class, and confidence can be shown to an operator, saved as evidence, or sent to an alarm module. The model alone does not decide system quality. Data diversity, annotation quality, negative samples, training settings, hardware, and post-detection rules all affect the final result.

YOLO Models in Flame, Smoke, and Wildfire Detection

YOLOv5 is widely used in current research on fires and smoke. It has a well-established foundation for engineering applications and a large number of datasets and workflows available for training and deployment. When researchers apply it to forest fire detection, they typically incorporate attention modules, lightweight structures, feature fusion variants, or small target designs. These studies provide important references for the improvement of YOLO. However, the data they report rely on their own specialized datasets and lack generalizability, requiring recalibration before being used in other systems [17].

In this article, the experience of using YOLOv7 is of significant importance because it focuses on efficient real-time detection. Current wildfire detection work has utilized YOLOv7's structural reparameterization, attention modules, and lightweight backbone network. YOLOv8 follows Ultralytics' workflow. It supports drone-based smoke detection as well as fine-grained fire detection achieved through integrated training, validation, prediction, and exporting. This workflow is convenient but has not demonstrated superior performance in the field of fire detection.[6, 20].

YOLOv9 and YOLOv10 address different issues. YOLOv9 works on information preservation and gradient propagation through programmable gradient information and GELAN. YOLOv10 reduces dependence on non-maximum suppression in real-time end-to-end detection. For fire alarm software, the first issue relates to feature quality, while the second issue concerns the stability and length of the inference path. YOLO11 and YOLO26 are updated Ultralytics models, so their value in fire scenarios still needs to be verified through future system experiments [8, 9].

Multimodal Wildfire Detection Research

RGB cameras are common in fire monitoring due to their simple installation, high prevalence, and low cost. However, the reliability of RGB information decreases at night or under conditions of smoke, obstructions, glare, and poor visibility. Thermal infrared imaging captures heat-related information rather than reflected visible light. Therefore, when visible images are uncertain, RGB-T and other multimodal fusion methods are the most efficient. Existing UAV-based RGB-T datasets and adaptive modality learning methods have already supported this research direction [21, 22]. A practical software framework should not impede subsequent multimodal extensions. The first version can only handle images in the RGB color standard, while later versions should be able to add more functions, such as thermal imaging technology.

Limitations of Existing Studies

There are still some issues with fire and smoke detection. When small flames and distant smoke occupy very small areas of an image, they may blend into the background and become difficult to detect. Smoke, haze, fog, dust,

and clouds can cause false alarms. False alarms may also arise from flame-like objects, such as sunsets, lights, reflective objects, and red signs. Therefore, results from a single dataset alone do not indicate generalizability across different regions, seasons, devices, and perspectives [16, 20].

There is also a gap between benchmark accuracy and alarm software. A detector that performs well in benchmark tests cannot automatically be used in control rooms. During deployment, issues such as video input, disconnections, encoding, thresholds, visualization, evidence storage, false alarm review, hardware compatibility, edge or cloud operation, and links to emergency workflows must be addressed. Therefore, model research requires a system-level framework to serve as support.

Technical Characteristics of YOLO26

General Positioning of YOLO26

Ultralytics describes YOLO26 as a recent YOLO real-time detector for edge devices, low-power hardware, simplified inference, and low-latency applications [11, 13]. The model is also presented as part of the Ultralytics multi-task vision ecosystem. In this paper, YOLO26 is used only as a candidate detector for wildfire flame and smoke monitoring. The official description is not treated as direct evidence of fire-domain performance.

The possible relevance of YOLO26 comes from deployment constraints. Cameras may be placed in forests, scenic areas, warehouses, industrial zones, or UAV platforms. These settings often have limited GPU resources, bandwidth, power, and maintenance capacity. A detector that is easier to export and run on an edge device may reduce integration effort. This point remains a hypothesis until it is tested on fire-specific data.

DFL Removal

Distribution Focal Loss models bounding-box distributions and localization quality in dense detection. Ultralytics states that YOLO26 removes DFL to simplify box prediction, export, and hardware support. In software terms, this may make inference outputs easier to process and reduce reliance on runtimes that support specific operators []14]. DFL removal should be interpreted as a deployment change, not as proof of better flame localization. Early flame boxes can be small, and smoke boundaries may be unclear. The question remains whether this streamlined output can still maintain adequate bounding box precision for small-scale flame and smoke. Ablation experiments and scale-based localization analysis are needed.

End-to-End NMS-Free Inference

Most YOLO detectors first produce many candidate boxes. They then use Non-Maximum Suppression to remove overlapping results. NMS is effective, but it remains a separate post-processing step. Its behavior can vary across CPU, GPU, TensorRT, ONNX, OpenVINO, and other runtimes. YOLO26 documentation emphasizes end-to-end inference without an external NMS stage [9, 12].

In an alarm pipeline, removing NMS can shorten the route from frame input to candidate alarm. It may also simplify model export and reduce latency. This benefit is most relevant when the target is a small early flame rather than a large visible fire. The change alone does not show higher recall or fewer false alarms. Those outcomes need controlled video tests and real monitoring data.

ProgLoss and Small-Target-Aware Label Assignment

Ultralytics describes Progressive Loss and Small-Target-Aware Label Assignment as training changes for stability and small-object recognition. This connects to wildfire monitoring because the first visible flame or smoke plume may occupy a small region in a wide image. UAV altitude and field of view can make the fire point even smaller [11, 13]. A label-assignment method that emphasizes small targets may help early detection. The test should be direct: compare small-target recall, missed-detection rate, and false-alarm rate across object scales. Until such results are available, STAL should be treated as a possible benefit rather than a confirmed advantage.

MuSGD Optimizer

Ultralytics describes MuSGD as combining SGD with Muon-related ideas to improve training stability and efficiency. This may be relevant for fire datasets because they often include class imbalance, unclear boundaries, and hard negative samples that resemble fire. The optimizer still needs task-specific testing. A fair test should keep the dataset, augmentation policy, learning-rate schedule, and hardware fixed while comparing MuSGD with standard optimizers [13].

Edge Deployment and CPU Inference

Edge deployment depends on more than model architecture. Camera resolution, codec, operating system, runtime engine, temperature, power supply, memory, network quality, maintenance interval, and alarm integration may determine the user experience. The proposed system should therefore be tested as a complete application pipeline, not only as a neural network.

Table 1: YOLO26 Technical Features and Their Relevance to Flame Detection

YOLO26 feature	Description	Value for wildfire detection	Validation needed
DFL Removal	Simplifies box prediction and export according to Ultralytics.	May reduce deployment complexity on edge hardware.	Test small-flame localization.
End-to-End NMS-Free Inference	Removes external NMS post-processing.	May lower post-processing latency and simplify the alarm pipeline.	Test duplicate boxes, false alarms, and recall.
ProgLoss	Intended to improve training stability.	May support learning from difficult fire/smoke samples.	Compare with standard loss settings.
Small-Target-Aware Label Assignment	Designed to improve small-target recognition.	Relevant to distant flames, UAV imagery, and early fire points.	Run scale-based testing and ablation.
MuSGD	Optimizer for stable and efficient training.	May help with imbalanced and complex fire datasets.	Compare with other optimizers.
Edge-oriented design	Designed for CPU and low-power devices.	Relevant to forest cameras, UAVs, and edge boxes.	Measure latency, power, and stability.

Qualitative Comparison with Previous YOLO Models

The following comparison is qualitative. It uses official materials, major YOLO papers, and reported fire-detection applications as sources [1, 11]. It is not an experimental result produced in this study.

Table 2: Developmental Comparison of YOLO Models for Flame and Wildfire Detection [1, 11].

Model	Main characteristics	Use in fire/smoke detection	Potential use	Main limits	Relation to YOLO26
YOLOv5	Mature PyTorch tools; simple training and export; widely used baseline.	Used often for forest smoke, flame, and fire detection.	Easy reproduction; many examples; strong baseline value.	Often depends on NMS and task-specific tuning.	Baseline for future YOLO26 comparison.
YOLOv7	Real-time detector with trainable bag-of-freebies and efficient structure.	Used in fire and smoke detection with attention or re-parameterization modules.	Good real-time capability for video monitoring.	Engineering ecosystem is less unified than newer Ultralytics workflows.	Useful high-performance reference model.
YOLOv8	Ultralytics workflow with integrated training, validation, prediction, and export.	Used for UAV wildfire smoke detection and flame/smoke fine-tuning.	Practical deployment workflow; easy transfer to applied research.	Small fire points and false positives still need task-specific treatment.	Predecessor ecosystem for comparing YOLO26.
YOLOv9	Programmable Gradient Information and GELAN for information preservation.	Candidate for complex fire-detection tasks.	Strong feature-learning potential.	Fire-specific deployment evidence remains limited.	Modern baseline after YOLOv8.
YOLOv10	Real-time end-to-end detection with NMS-free direction.	Relevant to low-latency alarm systems.	End-to-end design may simplify deployment.	Fire-specific performance still needs validation.	Provides technical background for YOLO26 NMS-free design.
YOLO11	Recent Ultralytics model supporting multiple tasks and deployment workflows.	Potential recent baseline for flame/smoke detection.	Unified ecosystem and task support.	Fire-domain behavior depends on data and deployment.	Immediate predecessor for comparison.
YOLO26	Emphasizes DFL removal, NMS-free inference, ProgLoss, STAL, MuSGD, and edge deployment.	No author-conducted fire experiments are reported in this paper.	Theoretically suitable for small targets, low latency, and edge integration.	Actual fire-detection performance is still unverified.	Core model of the proposed framework.

Proposed YOLO26 Wildfire Flame Detection Software System

System Goal

This system is temporarily named the YOLO26 Wildfire Monitoring System. Its function is to serve as an auxiliary early warning tool and cannot replace the judgment of humans or fire departments. The software should support multiple practical functions, such as real-time video detection, image detection, model weight configuration, confidence level settings, flame/smoke target box display, alarm prompts, detection logs, evidence screenshots, and connection to the intelligent fire-fighting cloud platform. The design follows one rule: a detection result is only a candidate. It is not a confirmed fire. A single frame with a flame-like box should trigger further checking rather than direct action. Alarm decisions should combine confidence thresholds, multi-frame consistency, spatial filtering, historical context, and manual review.

Overall Workflow

The workflow has the following sequence:

In use, the pipeline begins with an image or video stream. Frames enter YOLO26 inference, and the predicted flame or smoke boxes are stored as candidates. The system then checks confidence, persistence across frames, and region-of-interest rules. After that, it displays results and records alarm prompts, logs, evidence files, manual review information, and emergency-linkage data.

This workflow keeps detection and alarm confirmation separate. The separation is needed because sunlight, vehicle lamps, reflections, red objects, smoke-like clouds, and industrial lights may all look like fire in a camera view.

System Modules

Software System Modules

Module	Function	Input	Output	Design considerations
Data input module	Accepts images, videos, RTSP streams, USB cameras, UAV videos, and forest monitoring streams.	Image files, video files, camera streams.	Standardized frames with timestamps.	Handle resolution, frame rate, disconnection, and codec differences.
Model inference module	Loads YOLO26 weights and detects fire, flame, smoke, and optional confusing classes.	Preprocessed frames.	Candidate boxes, classes, and confidence scores.	Support CPU, GPU, Jetson, edge boxes, and cloud runtime.
Visualization module	Draws boxes, labels, confidence values, timestamps, and alarm status.	Detection results.	Annotated images or videos.	Avoid hiding key scene information and support evidence saving.
Alarm decision module	Converts candidate detections into alarm levels using rules.	Detection sequence and configuration.	Candidate warning, suspected alarm, confirmed alarm.	Use thresholds, multi-frame logic, ROI filtering, and manual review.
Log and evidence module	Stores time, device ID, class, confidence, box coordinates, screenshots, clips, and review outcomes.	Detection and alarm records.	Logs, images, video clips, review files.	Support traceability and model improvement.
User interface module	Allows users to select model, input source, threshold, save path, and alarm method.	User settings.	Runtime configuration.	Design for forest staff, safety officers, and emergency operators.
Deployment module	Packages the system for Windows, Linux, CPU, GPU, Jetson, edge boxes, and cloud services.	Software and model files.	Operational system.	Consider dependency management, long-term maintenance, and security.

Data Input Module

The data input module accepts local images, local videos, USB cameras, network cameras, UAV video, and forest-monitoring video. RTSP streams and UAV links are important for field use. Before detection, the module should standardize frame size, color format, frame interval, and timestamp data. It also needs to handle abnormal frames, stream breaks, reconnection, compression artifacts, and unstable networks.

Model Inference Module

The model inference module uses YOLO26 as the detector. The initial labels may include fire, flame, and smoke. Later datasets should add confusing non-fire classes, such as non-fire light, sun reflection, vehicle light, and red objects.

The module also needs to support different model sizes and runtime backends. A server may run a larger model. A forest edge box, UAV computer, or CPU-only device may require a lighter model. This framework does not fix a final model size. The choice depends on hardware, video resolution, alarm tolerance, and validated performance.

Visualization Module

The visualization module should show boxes, class names, confidence values, timestamps, frame numbers, and alarm state. The display should remain readable for the operator. It should also save annotated screenshots and short video clips linked to alarms. In a multi-camera interface, each window should show the camera ID, location, connection state, and current alarm level.

Alarm Decision Module

The alarm decision module is responsible for converting the information output by the detector into decisions

on whether to trigger an alarm. High-risk alarms should not be based on a single frame alone. Stricter rules require that fire-related information be detected repeatedly across consecutive frames or within a short time window. The system should also undergo specialized training to handle roads, lighting, and highly reflective areas to reduce false alarm rates. The confidence threshold must be adjustable at any time or subject to automated control, as tolerance for false positives and false negatives varies with time and location.

A three-level alarm structure can show this uncertainty. Candidate warning means the model has found a possible flame or smoke region. Suspected alarm means the region persists across frames or meets stricter confidence rules. Confirmed alarm should rely on manual review, cross-sensor confirmation, or other operational evidence before emergency action. This design reduces the risk of treating a model score as fact.

Log and Evidence Storage Module

The log module should record detection time, camera ID, camera location, model version, input source, class name, confidence, box coordinates, alarm level, screenshot path, video-clip path, network status, and manual review result. These records support accountability, review, debugging, and future dataset construction. False alarms and missed detections reported by users should be added to the improvement loop.

User Interface Module

The user interface should serve operators as well as researchers. Basic controls include video-source selection, model-file loading, confidence-threshold adjustment, save-folder selection, alarm-sound settings, and review status. UAV use may require GPS coordinates, flight altitude, gimbal angle, and map location. Fixed forest cameras may require region name, device number, and installation site.

Contributions and Framework-Level Innovations

This paper limits its contribution to preprint-stage research ideas and system-framework design. The discussion should not be read as an experimentally verified performance claim. At the framework level, the paper applies YOLO26 to wildfire flame detection and connects the detector choice with software architecture, alarm logic, evidence storage, and deployment planning. It also discusses YOLO26 features, including NMS-free inference, DFL removal, ProgLoss, STAL, and MuSGD, in relation to fire detection. The focus is on small flames, low-latency alarms, and edge-device deployment [11, 13].

For later experiments, the comparison section provides a set of baseline candidates. It groups YOLOv5 to YOLO26 around issues that matter for flame and wildfire detection. The actual ranking should be decided by future tests. The architecture is not limited to model training. It defines data input, inference, visualization, alarm decision, logging, user interface, and deployment as connected parts of one software system. The framework covers images, videos, real-time cameras, UAVs, fixed forest cameras, and edge devices. It also includes multi-frame judgment, threshold control, region filtering, manual review, and evidence storage as operational layers. The paper also sets out a validation roadmap. It includes dataset construction, model comparison, ablation testing, multimodal fusion, and field-oriented deployment.

Practical Application Significance

If later experiments support the framework, its main practical value is earlier identification of suspicious flame or smoke in camera, video, and UAV streams. This may be most useful in monitoring centers with many video feeds. In such settings, operators cannot watch every frame continuously, and early cues may be missed. The application scenarios span multiple fields. In forest fire prevention, this system can be used for fire monitoring at fixed cameras, fire stations, and command centers. In scenic area operations, it can be applied to camping areas, forest edges, roads, and visitor activity areas. In warehouses and industrial sites, it can serve as an auxiliary tool for smoke alarms, thermal imaging detectors, and manual inspections. For drone surveillance, it can be used to continuously analyze long-duration aerial videos.

The framework may also support smart forestry, smart firefighting, emergency management, and ecological protection. Its direct output in these settings is visual evidence. Suspected events can be documented, reviewed, and linked with geographic information systems, weather data, sensor networks, and human inspection teams. The system should still be used as an auxiliary warning tool. Visual detectors can miss

fires and may also raise false alarms. Operational use needs manual review. When possible, it should also connect with sensor networks, thermal infrared detection, satellite monitoring, UAV inspection, and established emergency-response procedures.

Limitations

- This paper contains no author-generated experimental data, training results, ablation results, field-deployment results, or user-test results.
- YOLO26 has not yet been verified for flame, smoke, or wildfire detection. General-object detection descriptions from official sources cannot by themselves serve as fire-domain evidence.
- Results may change with dataset, region, season, fire type, camera angle, resolution, compression rate, and hardware platform.
- RGB images are unreliable in some conditions, including night scenes, heavy smoke, occlusion, rain, snow, fog, glare, and low visibility.
- Non-fire targets may be mistaken for flame. Examples include sunlight, lamps, vehicle lights, reflections, red objects, and industrial heat sources.
- Single-frame results may lead to false alarms. Multi-frame analysis, tracking, region rules, and manual review should be used.
- Field use also involves issues outside the model, such as network delay, power supply, dirty lenses, maintenance cost, data privacy, alarm responsibility, and links to emergency workflows.
- The effects of DFL removal, NMS-free inference, Prog Loss, STAL, and MuSGD must be checked through ablation experiments before being described as validated advantages.

Table 4: Limitations and Improvement Directions

Limitation	Potential impact	Improvement direction
No original experimental data yet.	The proposed framework cannot be treated as validated performance evidence.	Build datasets and run controlled experiments.
RGB images are sensitive to environment.	Night, glare, smoke, and occlusion may cause errors.	Add thermal infrared and multimodal fusion.
Small flames occupy few pixels.	Early fire points may be missed.	Use high-resolution inputs, small-object enhancement, and STAL evaluation.
Single-frame alarms are unstable.	False alarms may increase.	Use multi-frame consistency, tracking, ROI rules, and manual review.
Negative samples are visually complex.	Lights and reflections may be confused with flames.	Add confusing negative classes and hard-negative mining.
Edge hardware varies widely.	Latency, power, and compatibility are uncertain.	Test CPU, Jetson, edge boxes, and cloud deployment.

Future Work

Future work should begin by building a multi-scene dataset. It should include flame, smoke, wildfire scenes, non-fire lights, reflections, red objects, vehicle lights, night scenes, UAV views, fixed forest cameras, scenic areas, warehouses, and power transmission corridors. The label set may include fire, flame, smoke, non-fire light, sun reflection, vehicle light, and other hard negative categories.

YOLO26 should then be compared with YOLOv5, YOLOv7, YOLOv8, YOLOv9, YOLOv10, and YOLO11 under a shared training protocol. The evaluation should report Precision, Recall, F1-score, mAP@0.5, mAP@0.5:0.95, FPS, end-to-end latency, false-alarm rate, missed-detection rate, model size, memory use, and power consumption on different hardware.

Ablation studies are needed for DFL removal, NMS-free inference, ProgLoss, STAL, MuSGD, input resolution, negative-sample design, data augmentation, and multi-frame alarm logic. Small-target results should be reported separately because early flame detection is strongly affected by object scale.

Multimodal validation should compare RGB-only, thermal-only, and RGB-T models under daytime, nighttime, smoke, occlusion, and low-visibility conditions. Deployment tests should include Jetson devices, edge boxes,

CPU-only computers, GPU servers, UAV ground stations, and forest camera systems. Records should cover latency, stability, heat, power consumption, memory use, and maintainability [21, 22].

The final step is to connect operation and model improvement. Alarm logs, manual review results, false positives, and false negatives should be saved as evidence and used to improve later datasets and model versions.

Table 5: Future Experimental Validation Plan

Experiment direction	Content	Metrics	Purpose
Model comparison	Train and test YOLOv5 to YOLO26 under a unified protocol.	mAP, Recall, F1, FPS, latency.	Evaluate the relative suitability of YOLO26.
Small-target evaluation	Group objects by scale and test distant flames and UAV fire points.	Small-object AP, Recall, missed detection rate.	Verify STAL and small-object value.
Ablation study	Remove or replace key mechanisms and alarm strategies.	Metric changes under controlled settings.	Estimate the contribution of each component.
Multi-frame alarm	Compare single-frame and temporal alarm logic.	False alarm rate, response time, stability.	Improve alarm reliability.
Multimodal detection	Compare RGB, thermal, and RGB-T fusion.	Recall and false alarms under different environments.	Improve robustness in difficult scenes.
Edge deployment	Test CPU, Jetson, edge boxes, and cloud runtime.	Latency, power, memory, temperature.	Verify practical deployment feasibility.
Field-oriented testing	Use forest, scenic-area, warehouse, and UAV videos.	False positives, missed detections, review outcomes.	Assess real application value.

Conclusion

This paper proposes a YOLO26-based software framework for wildfire flame detection. The framework is developed from official YOLO26 descriptions, representative YOLO literature, and existing work on flame, smoke, and wildfire detection. End-to-end NMS-free inference, DFL removal, small-target-aware label assignment, Progressive Loss, MuSGD, and edge-oriented deployment may be relevant to early warning, especially when small objects, low latency, and software integration are central needs [11, 13].

The contribution is mainly organizational. The paper reviews related studies, explains YOLO26 features, compares YOLO models at a qualitative level, proposes a software architecture, discusses application scenarios, and gives a validation plan. The proposed system contains data input, YOLO26 inference, visualization, alarm decision, log and evidence storage, user interface, and deployment modules. These modules are prepared for later work in forest fire prevention, UAV inspection, scenic-area safety, warehouse fire monitoring, transmission-line inspection, smart forestry, and smart firefighting.

Because the paper reports no original data, its conclusions are limited to theoretical analysis and system design. It does not claim any YOLO26 result for fire-detection mAP, FPS, latency, false-alarm rate, or field performance. Future work should test the framework with public datasets, forest videos, UAV imagery, edge deployment, model comparison, ablation studies, and field evaluation.

References

1. J Redmon, S Divvala, R Girshick, A Farhadi (2016) You Only Look Once: Unified, Real-Time Object Detection. in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 779-788.
2. J Redmon, A Farhadi (2017) YOLO9000: Better, Faster, Stronger. in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 7263-7271.
3. J Redmon, A Farhadi (2018) YOLOv3: An Incremental Improvement. https://arxiv.org/abs/1804.02767.
4. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao (2020) YOLOv4: Optimal Speed and Accuracy of Object Detection https://arxiv.org/abs/2004.10934.
5. Ultralytics (2026) YOLOv5 repository and documentation. GitHub https://github.com/ultralytics/yolov5.
6. Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao (2023) YOLOv7: Trainable Bag-

of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) https://arxiv.org/abs/2207.02696.

7. Ultralytics (2026) Ultralytics YOLOv8," Ultralytics Documentation. https://docs.ultralytics.com/models/yolov8/
8. Chien-Yao Wang, I-Hau Yeh, Hong-Yuan Mark Liao (2024) YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. https://arxiv.org/abs/2402.13616.
9. Ao Wang, Hui Chen, Lihao Liu, Kai Chen and Zijia Lin, et al. (2024) YOLOv10: Real-Time End-to-End Object Detection https://arxiv.org/abs/2405.14458.
10. Ultralytics (2026) Ultralytics YOLO11. Ultralytics Documentation https://docs.ultralytics.com/models/yolo11/.
11. Ultralytics (2026) Ultralytics YOLO26. Ultralytics Documentation https://docs.ultralytics.com/models/yolo26/.
12. Ultralytics (2026) Understanding End-to-End Detection in Ultralytics YOLO26," Ultralytics Documentation https://docs.ultralytics.com/guides/end2end-detection/.
13. Vina (2026) Ultralytics YOLO26: The New Standard for Edge-First Vision AI," Ultralytics Blog https://www.ultralytics.com/blog/ultralytics-yolo26-the-new-standard-for-edge-first-vision-ai.
14. X Li, W Wang, L Wu, S Chen and X Hu (2020) Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection 14.
15. Alkhatib (2014) A Review on Forest Fire Detection Techniques. International Journal of Distributed Sensor Networks 10.
16. P Barmpoutis, PP Papaioannou, K Dimitropoulos, N Grammalidis (2020) A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing. Sensors 20.
17. J Yang, W Zhu, T Sun, X Ren and F Liu (2023) Lightweight Forest Smoke and Fire Detection Algorithm Based on Improved YOLOv5. PLOS ONE 18.
18. X Chen, Y Xue, Q Hou, Y Fu and Y Zhu (2023) RepVGG-YOLOv7: A Modified YOLOv7 for Fire Smoke Detection. Fire 6.
19. M Chetoui, MA Akhloufi (2024) Fire and Smoke Detection Using Fine-Tuned YOLOv8 and YOLOv7 Deep Models. Fire 7.
20. SN Saydirasulovich, Mukhriddin Mukhiddinov, Oybek Djuraev, Akmalbek Abdusalomov and Young-Im Cho (2023) An Improved Wildfire Smoke Detection Based on YOLOv8 and UAV Images. Sensors 23.
21. X Rui, Z Li, X Zhang, Z Li and W Song (2023) A RGB-Thermal Based Adaptive Modality Learning Network for Day-Night Wildfire Identification. International Journal of Applied Earth Observation and Geoinformation 125.
22. SDMW Kularatne, CA Casado, J Rajala, T Hanninen and MB Lopez, et al. (2024) FireMan-UAV-RGBT: A Novel UAV-Based RGB-Thermal Video Dataset for the Detection of Wildfires in the Finnish Forests. in Proc. 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA) 1-8.