Deep Learning and Domain Adaptation for Wildfire Detection: Design Indicators for Scalable Camera-Monitoring Networks

Lucas da Silveira Nascimento
Computer Engineering Undergraduate
CEFET-MG
Belo Horizonte, Minas Gerais
silveira0182@gmail.com

Ana Paula Batista
Electrical Engineering Department
CEFET-MG
Belo Horizonte, Minas Gerais
ana@cefetmg.br

Everthon de Souza Oliveira
Electrical Engineering Department
CEFET-MG
Belo Horizonte, Minas Gerais
oliveira@cefetmg.br

Abstract*—Wildfires increasingly threaten ecosystems and public safety. This work presents a comprehensive wildfire-detection pipeline that combines a YOLOv11 backbone with Unsupervised Domain Adaptation (UDA) to deliver robust, real-time performance across diverse visual domains: we first curate and augment two public wildfire datasets (D-Fire and WildFire-Smoke-Dataset-YOLO) with mosaic, MixUp, geometric, photometric, and scaling transforms to simulate challenging conditions; then integrate adversarial feature alignment and pseudo-label self-training directly into YOLOv11 to align labeled source and unlabeled target feature distributions and mitigate performance drops under new lighting, vegetation, and camera settings. A systematic evaluation across five input resolutions (64–1024 px) shows that UDA raises a 640×640 model to 96.37% precision and a 1024×1024 model to 91.69% mAP@0.50:0.95—narrowing the accuracy gap with high-resolution baselines by over 40%. Finally, we derive a deployment-aware coverage model that links resolution, object size, and field-of-view to estimate required camera counts for any area, demonstrating that increasing resolution from 640 to 1024 px in a 10 ha scenario cuts hardware needs from 503 to 196 cameras (over 60% savings). Together, these contributions—domain-adapted detection, realistic augmentation, multi-resolution benchmarks, and quantitative deployment guidelines—offer a scalable, cost-effective framework for edge-capable wildfire monitoring.

***Index Terms*—Wildfire detection, YOLOv11, Domain adaptation, Real-time object detection, Camera deployment, Edge AI.**

I. Introduction

Wildfires have become a growing concern due to their severe impacts on the environment, economy, and human populations. Their increasing frequency is largely driven by human-induced climate change, with rising global temperatures, prolonged dry seasons, and more frequent heatwaves creating ideal conditions for fires to ignite and spread [1]. In addition to causing widespread ecological destruction and threatening wildlife, wildfires generate significant pollution, worsening air quality and contributing to climate change [2], [3]. Regions such as North America, Australia, and Southern Europe have experienced more frequent, intense, and longer-lasting wildfires, highlighting the urgent need for advanced prevention and response systems [3], [4].

Fire detection operations traditionally use space-based imagery combined with human monitors at lookout towers and observational platforms. These detection methods face various shortcomings because they provide delayed alerts while offering restricted spatial and temporal detail and functioning best in conditions of good weather and visibility [5]. The process of manual observation becomes impractical for extensive monitoring across distant regions because it requires excessive human work [6].

The growing limitations of traditional wildfire detection methods—such as reliance on human observation, satellite latency, and sensitivity to weather conditions—have accelerated the adoption of artificial intelligence (AI) technologies in wildfire monitoring. AI-based systems offer significant advantages by enabling continuous, automated, and real-time analysis of visual and environmental data streams. Machine learning algorithms, particularly deep learning models, can be trained to recognize subtle indicators of early-stage fires, such as faint smoke patterns or thermal anomalies, with greater speed and consistency than manual observation. Furthermore, AI enhances the integration of heterogeneous data sources—including satellite imagery, drone footage, weather forecasts, and ground sensor inputs—allowing for more comprehensive and context-aware situational awareness. These capabilities not only improve early warning systems but also support predictive analytics for fire spread modeling, risk assessment, and resource allocation. As climate change continues to increase the frequency and intensity of wildfires, AI-driven solutions are becoming essential tools for building more resilient environmental monitoring infrastructures and enabling proactive wildfire management strategies.

Despite the success of one-stage detectors like YOLO in general object detection tasks—where they balance speed and accuracy for real-time applications—there remains a critical challenge when deploying these models for wildfire detection in diverse environments. Models trained on one region or sensor configuration often suffer severe performance degradation under different lighting, vegetation types, and camera settings. While Unsupervised Domain Adaptation (UDA)

techniques have proven effective in reducing domain shift for classification and segmentation tasks, their integration with YOLO-based detectors in the context of wildfire monitoring is scarce. Moreover, existing studies have not systematically evaluated how UDA can enable lower-resolution, cost-effective camera networks to match the performance of high-resolution setups. This gap limits the practical deployment of real-time, edge-capable wildfire detection systems across heterogeneous landscapes.

Unlike previous works that focused solely on detection performance, this research proposes a triple contribution: (i) a domain-adapted deep learning model using UDA for robust wildfire detection, (ii) a practical camera deployment model based on resolution, object size, and coverage equations, allowing cost-effective real-world system planning, and (iii) a comparative evaluation across multiple resolutions (64–1024 px), showing that UDA enables lower-resolution models to reach comparable performance, facilitating edge deployment.

II. YOLO and UDA Overview

Computer vision and deep learning have emerged as powerful tools for early fire detection. These AI-driven systems turn raw visual data—whether from surveillance cameras, drones, or ground sensors—into actionable insights by teaching neural networks to spot the tell-tale signs of smoke and flame in real time [7]. At the heart of these systems are deep convolutional neural networks (CNNs), which learn in stages: early layers pick up basic features like edges and textures, while deeper layers assemble those features into more complex patterns—say, the wispy gradients of a smoke plume or the flicker of a small flame—allowing the model to generalize across different lighting, weather, and background conditions [8].

Among modern detectors, the YOLO family ("You Only Look Once") achieves its remarkable efficiency and accuracy through three key design principles. First, it employs single-pass detection, meaning that, rather than generating and classifying thousands of candidate regions one by one (as two-stage detectors like R-CNN do), it divides the input image into a fixed grid and, in a single forward pass, predicts both bounding boxes and class probabilities for every cell—drastically reducing inference overhead. Second, its unified pipeline merges region proposal, feature extraction, and classification into one end-to-end network, eliminating the need for a separate proposal stage to tune or debug and thus simplifying training and deployment. Finally, YOLO strikes a balanced speed–accuracy trade-off by continuously refining its backbone architectures (from v4 through v8), incorporating attention modules that focus on salient regions (for example, wispy smoke trails), and applying advanced data-augmentation strategies during training. These combined innovations deliver state-of-the-art detection performance at the high frame rates essential for real-time wildfire monitoring.

These design choices make YOLO especially appealing for wildfire scenarios, where you often run on lightweight, edge-level hardware in remote forests. A model that processes 30+ frames per second on a small GPU or even an embedded board can detect fires sooner and trigger alerts before manual watchers—or slower systems—ever notice.

Yet even a fast, well-tuned detector can stumble in new environments. A YOLO model trained on clear-day drone footage may misfire at dusk, in fog, or against dense foliage. That's where Unsupervised Domain Adaptation (UDA) comes in: it teaches the network to "see" the new domain without labeling a single extra image. Methods like adversarial alignment (making source and target features indistinguishable), discrepancy minimization (reducing statistical gaps), and pseudo-label self-training let the same YOLO model adapt from sunny testbeds to nighttime camera feeds—cutting annotation costs and boosting real-world robustness.

Recent advances in wildfire detection have been propelled by the release of comprehensive, labeled datasets—such as D-Fire [9] and WildFire-Smoke-Dataset-Yolo [10]—which span diverse lighting conditions and viewpoints. These benchmarks have enabled researchers to train, compare, and refine CNN-based models under consistent evaluation protocols, fostering collaboration across academia and industry.

Early vision-based approaches, like Toreyin et al.'s HMM-driven system, leveraged color and temporal features to detect indoor flames with a 91.3% success rate but faltered under complex outdoor lighting and required manual feature engineering [11]. The advent of deep learning shifted focus toward real-time edge deployment: Muhammad et al. designed a lightweight CNN optimized for Raspberry Pi and Jetson TX1 devices, achieving 94.7% accuracy and sub-0.5s inference per frame across indoor and forest scenes [12]. Following this trend, Li et al. fine-tuned YOLOv3 on over 5,000 smoke images—applying rotation, brightness adjustment, and synthetic overlays—to reach an mAP of 88.4% on early-stage smoke detection, outperforming thermal-based alternatives [13].

More recent work has begun to tackle domain shift directly: Xu et al. integrated adversarial UDA into a Faster R-CNN framework, aligning features between synthetic fire images and unlabeled real-world data to boost cross-region mAP from 67.8% to 81.3% [14]. These studies trace a clear progression—from handcrafted, indoor-focused detectors to lightweight, CNN-driven edge solutions and, finally, to cross-domain learning techniques that improve robustness against environmental variability.

Building on this trajectory, our research embeds UDA within a YOLOv11 architecture and conducts a systematic evaluation across multiple datasets and input resolutions, demonstrating that domain-adapted, lower-resolution models can achieve performance on par with high-resolution baselines—thereby enabling cost-effective, real-time wildfire detection on resource-constrained edge platforms.

III. Augmentation Techniques

To ensure that the YOLOv11 detector, combined with Unsupervised Domain Adaptation, is exposed to scenarios with variations in lighting, perspective, and occlusions, we apply a suite of augmentation techniques. In this section,

we describe the mosaic and MixUp transforms, geometric variations, photometric adjustments, and scaling operations that comprise our synthetic data pipeline.

Some augmentation methods are:

- Mosaic Augmentation: The Mosaic Augmentation technique combines four images into one during training to enhance detection of small and partially occluded objects through simultaneous learning of multiple contexts.
- MixUp: MixUp generates ambiguous object boundaries by combining training images and their labels which leads to better generalization performance in ambiguous situations.
- Geometric Transformations: The implementation of horizontal and vertical flips served to generate different camera positions and drone flight paths.
- Photometric Adjustments: This approach applied changes to hue, saturation, brightness, and contrast levels to replicate scenarios with daylight fluctuations and smoke-related blockages and poor visibility conditions.
- Scaling and Random Cropping: The transformations included scaling along with random cropping operations to enable the model detect fire and smoke at various distances and spatial scales.

These augmentation techniques are fundamental for generating a diverse training dataset because public datasets contain limited fire and smoke instances. All of the presented techniques were applied in the dataset, balancing and creating a more compatible dataset for the problem.

IV. Resolution and Distance Relationship

Understanding how resolution relates to distance is crucial when designing any vision system—be it for drones, surveillance cameras, or autonomous vehicles. In simple terms, the farther an object is from a camera, the fewer pixels it occupies in the image. That makes it harder to detect, recognize, or classify the object accurately.

This relationship is commonly quantified using a metric called Ground Sample Distance (GSD), which defines the real-world area represented by each pixel in an image. As the distance increases, so does the GSD, meaning each pixel "sees" a larger portion of the world and captures less detail about smaller objects [15].

The GSD can be calculated based on the camera's sensor size, lens focal length, and the altitude or distance to the target. The Field of View (FOV) of the lens also plays a central role—wide-angle lenses can cover more area, but at the cost of lower detail per pixel. In contrast, telephoto lenses offer more detail but less coverage [16].

When designing a vision system for monitoring, agriculture, surveillance, or autonomous robotics, one of the key design challenges is estimating how many cameras are required to effectively cover a given area. This estimation depends not only on the total area but also on the resolution of each camera, the size of the objects of interest, and how many pixels those objects must occupy in the captured image to be reliably detected or recognized.

To address this, we can use the following equation:

$$P = \frac{W_{\text{object}} \cdot R}{2D \cdot tan(\frac{\theta}{2})} \tag{1}$$

where D is the maximum distance between the camera and the object, W_{object} is the real width of the object to be detected, R is the horizontal resolution of the camera (in pixels), P is the number of pixels for reliable detection or classification, and θ is the Field of View

This formula is derived from the principle of GSD and assumes a square, uniformly distributed field of view. It essentially calculates the largest possible scene size where an object of a given size still appears with enough pixels in the image to be usable [15].

In our camera-based monitoring system, the imaging plane is always a rectangular grid of R pixels in width and R pixels in height. Although the lens's field of view defines a three-dimensional conical (or pyramidal) volume, that volume is projected onto this two-dimensional sensor.

To determine the maximum area that a single camera can cover, the following equation is used. It ensures that an object will not appear farther than the camera's capability to capture it at a resolution of R, while maintaining an object size of P pixels in the image.

$$\frac{a \cdot \sqrt{2}}{2} \leq D \tag{2}$$

where a is the side length of the square area covered by the camera.

Knowing the side length, the maximum area A_{max} that a camera can cover can be calculated by:

$$A_{\text{max}} = a^2 \tag{3}$$

Once the maximum coverage area per camera is known, the number of cameras required to cover the full area A_{total} can be estimated as:

$$N_{\text{cameras}} = \frac{A_{\text{total}}}{A_{\text{max}}} \tag{4}$$

This estimation is particularly useful in applications where precise spatial resolution is critical — such as in urban surveillance, precision agriculture, and infrastructure monitoring. It allows for a quantitative understanding of how sensor resolution, object size, and detection requirements impact the scalability and cost of a camera-based system [17].

In practical deployments, other factors like overlap between fields of view, occlusions, camera angle, and terrain elevation must also be considered. However, this model provides a valuable starting point for system-level planning.

V. Methodology

The methodology applied to this study revolves around the life cycle of a machine learning project. The stages were as follows: (i) Dataset Definition (ii) Data preprocessing, (iii) Implementation, (iv) Evaluation.

1) Dataset Definition Our study utilized two public wildfire datasets: D-Fire and WildFire-Smoke-Dataset-YOLO. The D-Fire dataset comprises 21.527 images with 1.164 annotations for fire, 5.867 for smoke and 4.658 for both. The WildFire-Smoke-Dataset-YOLO contains 737 images and annotations. About the train-test split, were used 70%/20%/10% random split to ensure representative distributions across all subsets.
2) Data preprocessing
   The first step of preprocessing involved complete evaluation of the two datasets D-Fire and WildFire-Smoke-Dataset-YOLO. Both datasets contain JPEG (JPG) images that support standard image processing libraries while allowing effective feature extraction during training.
   The second step was applying the data augmentation techniques mentioned in section IV. The parameters are shown in Table I.

Table I: Data augmentation hyperparameters. Parameters 'fliplr' and 'flipud' denote the probabilities of applying horizontal and vertical flips, respectively, while 'scale' defines the range for random scaling transformations. Further details on these augmentation techniques are provided in Section III.

Parameter	Value
lr0	0.0005
lrf	0.2
momentum	0.937
weight_decay	0.0005
warmup_epochs	3.0
warmup_momentum	0.8
warmup_bias_lr	0.1
mosaic	1.0
mixup	0.2
fliplr	0.5
flipud	0.5
scale	0.5
hsv_h	0.015
hsv_s	0.7
hsv_v	0.4

   During preprocessing, all images are resized and padded to the specified resolution, and the network adapts its convolutional layers and detection heads accordingly. Higher resolutions typically improve detection accuracy, especially for small objects, while lower resolutions reduce computational cost and inference time.

3) Implementation
   The implementation was the stage in which the model was trained and the results obtained. The training process was carried out on Google Colab, utilizing an NVIDIA T4 GPU to accelerate computations and reduce training time. This setup enabled efficient model convergence while maintaining accessibility through a cloud-based environment.
   For comparison purposes, the same model was trained with multiple types of image resolution. All models were trained and evaluated in the same way, taking into account the same dataset and the number of 50 epochs for training.
   Unsupervised Domain Adaptation (UDA) functions as a machine learning approach which addresses situations where training data from the source domain differs from application data in the target domain. The main objective of UDA involves modifying a model that received source domain labels to achieve effective performance in the unlabelled target domain.
4) Evaluation
   As the task is object detection, the models were evaluated using standard performance metrics including precision, recall, mean Average Precision (mAP), and a fitness score. Precision measures the proportion of true positive (TP) detections among all predicted positives, indicating the model's accuracy in identifying only relevant instances. Recall measures the proportion of true positives detected out of all actual positive instances, reflecting the model's ability to identify all relevant objects. The mean Average Precision (mAP) is a comprehensive metric that calculates the average of precision across recall levels and is typically evaluated at different Intersection over Union (IoU) thresholds, such as mAP@0.5 or the averaged mAP@[0.5:0.95], providing insight into the model's overall detection quality. Additionally, a fitness score was used to combine these metrics into a single value for model selection and comparison, with higher weights typically assigned to mAP due to its importance in real-world performance.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$\text{AP}_{\text{IoU}=t} = \int_0^1 p(r)\, dr \tag{7}$$

$$\text{mAP}_{[0.5:0.95]} = \frac{1}{10} \sum_{t=0.5}^{0.95} \text{AP}_{\text{IoU}=t} \tag{8}$$

$$\text{Fitness} = 0.1 \cdot \text{Precision} + 0.1 \cdot \text{Recall} + 0.8 \cdot \text{mAP}_{@0.5} \tag{9}$$

Considering:

- FP as False Positive
- FN as False Negative
- TP as True Positive

A thorough evaluation of the system performance was conducted using a combination of precision, recall, and mean Average Precision (mAP) at various Intersection over Union (IoU) thresholds. This chosen metrics are commonly used in object detection benchmarks and are effective for measuring both detection accuracy and localization quality.
The experiments were conducted at multiple different input resolutions, 64x64, 128x128, 256x256, 640×640 1024×1024 pixels, to evaluate the trade-off between model accuracy and computational efficiency.

Furthermore, altering only the imgsz parameter in YOLO training effectively changes the input resolution of the model without requiring modifications to the network architecture. This is possible because YOLO is designed to handle variable input sizes through dynamic resizing and automatic adjustment of internal feature maps, anchors, and target boxes.

VI. Results

Based on the methodology described above, the following section presents the results obtained from training YOLOv11-based detection models on the D-Fire dataset.

A summary of the performance metrics for both model configurations is provided in Table II, which includes key comparisons and performance trends.

Table II: Main training metrics for the YOLO model.

Metric	Model 64	Model 128	Model 256	Model 640	Model 1024
Fitness	0.1827	0.2890	0.4064	0.4484	0.4959
Precision	0.5267	0.6558	0.7172	0.7272	0.7881
Recall	0.3246	0.4566	0.6037	0.6698	0.7089
mAP@0.50	0.3256	0.4954	0.6612	0.7304	0.7923
mAP@0.50:0.95	0.1668	0.2661	0.3781	0.4170	0.4630

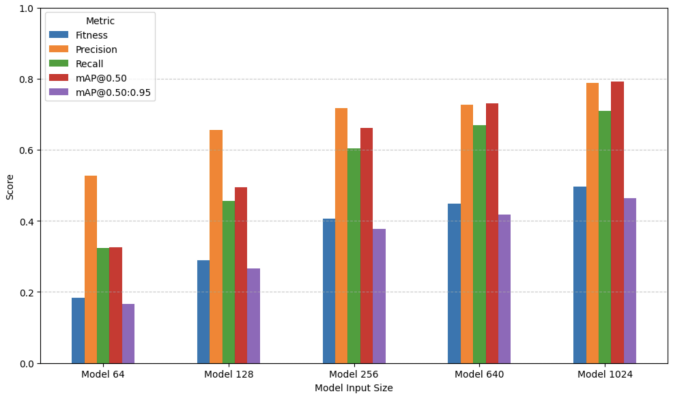

Figure 1: Performance metrics of YOLOv11 models at various input resolutions, based on results from Table 1.

The model's detection abilities and behavioral patterns became clearer after we calculated the average precision (AP) for each class at an IoU threshold superior than 0.50. This metric measures the overlap between the predicted box and the actual box. With threshold greater than 0.5, it means that the predicted box covers at least half of the actual box. The detection performance evaluation for Fire and Smoke target classes becomes possible through this metric which allows us to assess the model's sensitivity and reliability for different visual features. The results are summarized in Table III.

Table III: The table shows AP at IoU 0.50 for Fire and Smoke classes using YOLO models

Class	Model 128	Model 256	Model 640	Model 1024
Fire	0.36816	0.45924	0.47468	0.52233
Smoke	0.16405	0.29714	0.35936	0.40370

The computational efficiency of detection models needs evaluation through accuracy-related metrics because real-time applications in edge computing require limited processing resources. To evaluate real-time performance, we measured average execution times (in milliseconds) for preprocessing, model inference, loss calculation, and postprocessing stages in both model configurations. The results are presented in Table IV.

Table IV: The table shows the average processing time for each inference stage in YOLO models.

Metric	Model 64	Model 128	Model 256	Model 640	Model 1024
Preprocess	0.0060	0.0125	0.0291	0.15	0.34
Inference	0.1815	0.31	0.5930	2.99	7.81
Loss calc.	0.0000726	0.000157	0.000131	0.0004	0.0036
Postprocess	2.0569	1.92	2.0157	2.32	2.24

Figure 2: Detection results using YOLO models with input sizes of 640 (top) and 1024 (bottom), showing predictions for fire and smoke.

After completing the initial training phase we added Unsupervised Domain Adaptation (UDA) techniques to the training pipeline to enhance the two best model's robustness when deployed in environments different from the labeled training domain. UDA aims to counter the performance drop that occurs from domain shift by aligning feature distributions between source and target domains without needing target domain labels. The results following the implementation of UDA are shown in Table V.

The evaluation of Unsupervised Domain Adaptation (UDA) on the model's capability to differentiate between wildfire-related phenomena of different types was performed by computing the average precision (AP) at an IoU threshold of 0.50 for the Fire and Smoke classes separately. The results, presented in Table VI, demonstrate the model's performance

Table V: The table summarizes the performance of YOLO models at 640×640 and 1024×1024 resolutions after UDA, including fitness score, precision, recall, and mAP at IoU 0.50 and 0.50:0.95.

Metric	Model 640	Model 1024
Fitness	0.8867	0.9229
Precision	0.9637	0.9285
Recall	0.9550	0.9498
mAP@0.50	0.9812	0.9778
mAP@0.50:0.95	0.8762	0.9169

in detecting each target class after adaptation to a new domain.

Table VI: The table shows the AP at IoU 0.50 for Fire and Smoke classes using YOLO models with 640×640 and 1024×1024 resolutions after applying UDA.

Class	Model 640	Model 1024
Fire	0.87069	0.87815
Smoke	0.88162	0.95555

The operational feasibility of real-time deployment were evaluated for the adapted models by measuring the average time spent (in milliseconds) at each stage of the inference pipeline: preprocessing, inference, loss calculation, and postprocessing. The metrics in Table VII show detailed computational trade-offs for different input resolutions after Unsupervised Domain Adaptation (UDA) implementation.

Table VII: The table shows the average inference time for each processing stage—preprocessing, inference, loss calculation, and postprocessing—for YOLO models trained at 640×640 and 1024×1024 resolutions after applying Unsupervised Domain Adaptation (UDA).

Metric	Model 640	Model 1024
Preprocess	1.19	0.57
Inference	7.38	10.08
Loss calc.	0.00375	0.00053
Postprocess	2.31	6.07

In addition to precision and speed, practical deployment of wildfire monitoring systems must consider how many cameras are needed to cover large areas. For example, the number of cameras for 10 hectares was calculated based on the formula presented in Section IV:

Where:

- W_{object}: 2 meters (e.g., minimum detectable fire or smoke region [18]);
- R: 1024 pixels (camera resolution);
- P: 64 pixels (number of pixels required for reliable detection [19]);
- A_{total}: 100.000 (in m²);

To estimate the number of cameras required for complete coverage, the proposed model incorporates parameters such as object width, camera resolution, and minimum detectable pixel size. By substituting the initial values into the Equation 1, using a resolution of 1024 pixels, the resulting detection distance was calculated as 16 meters. Using the Equation 2, was calculated that the side of square that a camera can cover is 22.6 m. This led to a maximum coverage area per camera of approximately 510.76 m² by Equation 3, resulting in a

Figure 3: Detection results after applying UDA, using YOLO models with input sizes of 640 (top) and 1024 (bottom).

total of 196 cameras, by Equation 4, needed to monitor an area of 100,000 m². When the resolution was reduced to 640 pixels, the same process yielded a detection distance of 10 meters, decreasing the maximum area per camera to 198.81 m². Consequently, the required number of cameras increased significantly to 503. These results highlight the direct impact of input resolution on the spatial coverage and infrastructure requirements for fire and smoke detection systems.

VII. Discussion

Building on the results presented in the previous section, the experimental findings indicate that increasing the input resolution of YOLOv11 models and applying Unsupervised Domain Adaptation (UDA) techniques significantly enhances the detection performance of wildfire-related events, particularly under cross-domain conditions.

A comparison of models trained at varying input resolutions demonstrates that the 1024×1024 model consistently outperforms its lower-resolution counterparts in both class-wise average precision (AP) and overall mAP. Prior to domain adaptation, the 1024×1024 model achieved an AP of 0.52233 for the Fire class and 0.40370 for the Smoke class, outperforming the 640×640 model, which attained 0.47468 and 0.35936, respectively. These results underscore the importance of high-resolution inputs in enhancing the model's ability to capture fine-grained spatial features, such as flame contours and diffuse smoke patterns.

Despite the superior accuracy of high-resolution models, lower-resolution variants like 128×128 and 256×256 offer

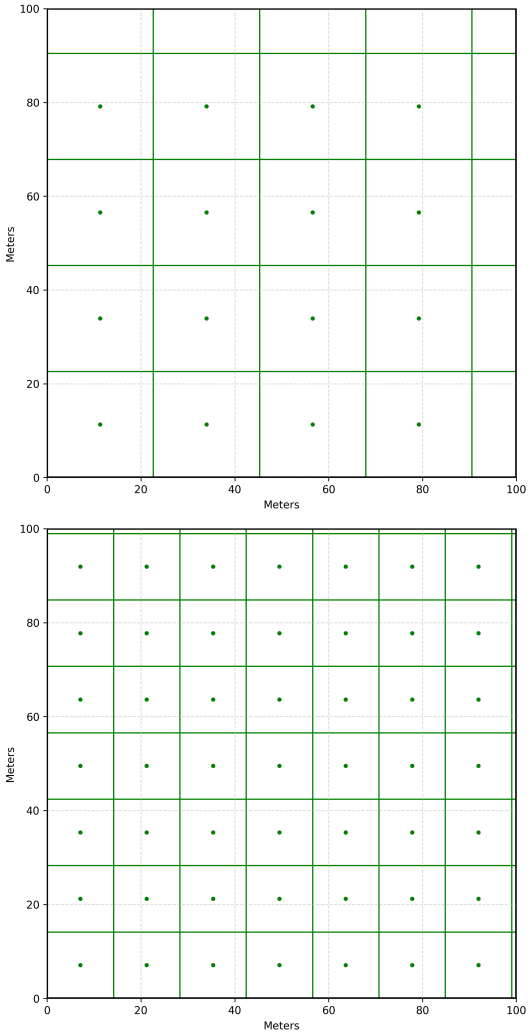

Figure 4: Camera coverage for one hectare with resolution of 1024 and 640

meaningful benefits in resource-constrained environments, such as edge devices or sensor networks with limited computational power. These models are particularly useful for real-time applications where speed and energy efficiency are critical.

For example, the 128×128 model completes a full inference cycle in under 3.5 ms on average, making it ideal for lightweight alert systems. However, this gain in efficiency comes at the cost of reduced accuracy, especially under challenging conditions like variable lighting or dense smoke. While the 128 and 256 models are better suited for preliminary or backup detection, the 640×640 configuration strikes a strong balance between performance and deployability across a range of scenarios.

The application of UDA techniques resulted in substantial performance improvements for both models. The 1024×1024 model achieved a precision of 0.9285, recall of 0.9498, and mAP@0.50:0.95 of 0.9169. The 640×640 model showed slightly higher precision (0.9637), but slightly lower recall (0.9550) and mAP@0.50:0.95 (0.8762). These results highlight UDA's effectiveness in enhancing generalization and reducing the performance gap across resolutions, particularly in unseen domains without labeled data.

From a class-wise perspective, domain adaptation was especially beneficial for smoke detection. The 1024×1024 model achieved an AP of 0.95555 for the Smoke class, surpassing the 640×640 model's 0.88162, indicating that high-resolution inputs combined with UDA are better suited to identifying subtle smoke features. Fire detection also improved for both models, with comparable AP values (0.87815 for 1024×1024 and 0.87069 for 640×640), reflecting consistent performance across classes.

Figures 2 and 3 provide a qualitative comparison of model outputs at different stages of the training pipeline. In Figure 2, prior to applying UDA, the 1024×1024 model (bottom) detects fire regions with broader and more confident bounding boxes than the 640×640 model (top). Increasing the input resolution (from 640 to 1024) results in better definition and confidence in the detections, with notable improvements in box positioning accuracy and a reduction in unnecessary overlaps.

Figure 3 shows the detection results after UDA application. Both models display notable improvements in detection confidence, bounding box consistency, and coverage. Notably, the 640×640 model (top) achieves better performance than the 1024×1024 model (bottom), suggesting that UDA can effectively compensate for lower input resolution under certain conditions. This is particularly important for real-time systems where computational resources are limited, as it enables the deployment of smaller models without drastically compromising detection quality.

Despite the accuracy benefits, increasing resolution also increased computational demand. The inference time on Table VII for the 1024×1024 model was 10.08 ms compared to 7.38 ms for the 640×640 model post-UDA. Likewise, postprocessing was significantly longer for the higher-resolution model (6.07 ms vs. 2.31 ms), reflecting the additional complexity in bounding box refinement and non-maximum suppression.

Interestingly, preprocessing time was lower for the 1024×1024 model (0.57 ms vs. 1.19 ms), likely due to pipeline optimization in loading and resizing operations. Although inference cost grows with input size, these results suggest that trade-offs can be mitigated through efficient implementation techniques.

As detailed in Section VI and illustrated in Figure 4, the calculations demonstrate that increasing camera resolution significantly reduces the number of devices required to monitor large forest areas. These findings underscore the importance of resolution not only for detection performance, but also for optimizing infrastructure and deployment costs.

In practical terms, although high-resolution models demand higher computational resources, they offer greater spatial efficiency and potentially lower total deployment costs, especially when factoring in installation, connectivity, and edge processing capacity.

Therefore, selecting the optimal resolution is not merely a trade-off between speed and accuracy—but a strategic de-

cision that affects the entire monitoring system's scalability and sustainability. The proposed analytical framework allows decision-makers to design AI-driven fire detection systems based on quantitative coverage models, bridging the gap between computer vision performance and operational deployment planning.

VIII. Conclusion

This study investigated the enhancement of wildfire detection systems through the integration of Unsupervised Domain Adaptation (UDA) techniques into deep learning models based on YOLOv11. Addressing a critical limitation in many existing detection systems—their inability to generalize well to new, unseen environments—this work proposed a comprehensive approach that combines architectural robustness, domain adaptation, and practical deployment planning.

Experimental results confirmed that UDA significantly improves detection performance across visual domains, especially for the Smoke class. Notably, after applying UDA, the 1024×1024 model achieved high accuracy scores, including a precision of 92.85%, recall of 94.98%, and mAP@0.50:0.95 of 91.69%. However, a key finding of this study is that the 640×640 model, although using lower resolution, achieved a higher precision of 96.37% and a comparable recall (95.50%), with only a moderate drop in mAP (87.62%). This shows that UDA can effectively compensate for lower input resolution, allowing smaller, faster models to remain highly effective under resource constraints.

In addition to detection performance, this work introduced a practical framework for estimating the number of cameras needed to monitor a given area based on resolution, object size, and field of view. The results show that increasing camera resolution from 640 to 1024 pixels reduces the number of cameras needed to monitor 10 hectares from 503 to 196—representing over 60% savings in hardware. This analytical model bridges the gap between AI model design and real-world system planning, supporting more cost-effective, scalable deployment.

This work presents a robust wildfire detection pipeline leveraging YOLOv11 and Unsupervised Domain Adaptation (UDA), demonstrating effectiveness even in unseen domains. Additionally, a deployment-oriented framework is proposed to quantify the relationship between input resolution, coverage area, and overall system cost. Finally, a comparative analysis is conducted to evaluate the trade-offs between model resolution and detection precision.

This combination supports the development of intelligent wildfire monitoring systems that are not only accurate and generalizable, but also practical to deploy in large-scale, resource-constrained environments.

For future work, we recommend exploring temporal modeling using video data, integrating multi-modal sources such as thermal and meteorological sensors, and optimizing lightweight models for edge computing, and conducting a cost analysis of embedded solutions to assess feasibility and scalability in real-world deployments. These directions could further enhance detection reliability and reduce false alarms, supporting faster and more effective wildfire response.

References

[1] K. C. Short, "Spatial and temporal patterns of wildfires in the united states, 1992–2012," *International Journal of Wildland Fire*, vol. 24, no. 8, pp. 900–914, 2015.

[2] D. M. Bowman, J. K. Balch, P. Artaxo, W. J. Bond, J. M. Carlson, M. A. Cochrane, C. M. D'Antonio, R. S. DeFries, T. W. Doyle, S. P. Harrison *et al.*, "Fire in the earth system," *Science*, vol. 324, no. 5926, pp. 481–484, 2009.

[3] M. A. Moritz, E. Batllori, R. A. Bradstock, A. M. Gill, J. Handmer, P. F. Hessburg, J. Leonard, S. McCaffrey, D. C. Odion, T. Schoennagel *et al.*, "Learning to coexist with wildfire," *Nature*, vol. 515, no. 7525, pp. 58–66, 2014.

[4] A. L. Westerling, "Increasing western us forest wildfire activity: sensitivity to changes in the timing of spring," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 371, no. 1696, p. 20150178, 2016.

[5] L. Giglio, J. Descloitres, C. O. Justice, and Y. J. Kaufman, "An enhanced contextual fire detection algorithm for modis," *Remote sensing of environment*, vol. 87, no. 2-3, pp. 273–282, 2003.

[6] X. Li, Y. Wang, Z. Liu, and S. Liu, "Automatic wildfire detection based on surveillance cameras: A review," *Fire Safety Journal*, vol. 105, pp. 183–200, 2019.

[7] B. C. Ko, H.-B. Cheong, and J.-Y. Nam, "Fire detection based on vision sensor and image processing," *Fire Safety Journal*, vol. 44, no. 3, pp. 322–329, 2009.

[8] K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, and S. W. Baik, "Efficient deep cnn-based fire detection and localization in video surveillance applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1419–1434, 2019.

[9] P. V. A. B. de Venâncio, A. C. Lisboa, and A. V. Barbosa, "An automatic fire detection system based on deep convolutional neural networks for low-power, resource-constrained devices," *Neural Computing and Applications*, 2022, available online.

[10] Ahemateja19BEC1025, "Wildfire-smoke-dataset-yolo," 2023, accessed on April 23, 2025. [Online]. Available: https://www.kaggle.com/datasets/ahemateja19bec1025/wildfiresmokedatasetyolo

[11] B. U. Toreyin, Y. Dedeoglu, and A. E. Cetin, "Flame detection in video using hidden markov models," in *IEEE International Conference on Image Processing, 2005.*, vol. 2. IEEE, 2005, pp. 1230–1233.

[12] K. Muhammad, J. Ahmad, Z. Lv, and S. W. Baik, "Efficient deep learning model for wildfire detection in forest monitoring systems," *Computers, Materials & Continua*, vol. 55, no. 3, pp. 429–442, 2018.

[13] X. Li, D. Zeng, T. Liu, and Y. Li, "Wildfire smoke detection based on yolov3 using transfer learning," *Remote Sensing*, vol. 12, no. 20, p. 3343, 2020.

[14] H. Xu, W. Zhang, L. Chen, and T. Han, "Wildfire smoke detection using unsupervised domain adaptation and adversarial learning," *IEEE Transactions on Image Processing*, vol. 31, pp. 1523–1534, 2022.

[15] R. A. Schowengerdt, *Remote sensing: models and methods for image processing*. Academic Press, 2006.

[16] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Pearson, 2018.

[17] M. S. Nixon and A. S. Aguado, *Feature extraction and image processing for computer vision*. Academic Press, 2019.

[18] T. Celik, H. Demirel, H. Ozkaramanli, and M. Uyguroğlu, "Fire and smoke detection without sensors: wavelet analysis-based algorithm for vision surveillance," *Vision, Image and Signal Processing, IET*, vol. 152, no. 2, pp. 235–243, 2007.

[19] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.