Fire and Smoke Detection in Confined Space

Abstract

Object detection is a vital component of safety and monitoring systems, particularly in environments where fire and smoke pose significant risks. This study investigates the performance of advanced YOLO models—YOLOv8n, YOLOv9s, and YOLO11s—for detecting fire and smoke in confined settings such as warehouses and server rooms. A custom dataset, generated using advanced AI tools to simulate diverse fire and smoke scenarios, was utilized for training and evaluation. The models were assessed based on precision, recall, and mean average precision (mAP) to determine their suitability for real-world hazard detection. The results highlight the trade-offs between model accuracy and computational efficiency, offering insights into the most appropriate configurations for specific use cases. This research aims to advance fire and smoke detection systems by leveraging state-of-the-art deep learning architectures.

Introduction

Safety monitoring in confined environments such as warehouses and server rooms is essential to mitigate the risks posed by fire and smoke. Traditional sensor-based systems, while effective in many cases, can be limited by environmental factors such as airflow disruption or sensor malfunctions. Vision-based systems, powered by object detection models, provide a robust alternative by analyzing visual cues in real time.

This study focuses on evaluating three YOLO models—YOLOv8n, YOLOv9s, and YOLO11s—for their ability to detect fire and smoke in challenging environments. The dataset used in this research consists of synthetic images generated using advanced AI tools, capturing a variety of scenarios with different smoke densities, fire intensities, and environmental conditions. The inclusion of such diverse

scenarios ensures that the trained models can generalize effectively to real-world applications.

The primary objective is to assess and compare the performance of these models based on metrics such as precision, recall, and mean average precision (mAP). Each model offers unique advantages in terms of computational efficiency, architectural complexity, and detection accuracy. This research aims to provide a comprehensive understanding of the trade-offs between these models, guiding practitioners in selecting the most suitable option for implementing fire and smoke detection systems in confined environments.

Dataset Description

This dataset consists of a collection of images simulating smoke and fire scenarios in warehouse and server room environments. Generated using advanced generative AI tools, these images are designed to aid in the training of monitoring models, enhancing their ability to detect and respond to potential fire hazards in various settings.

Model Architecture

The object detection task was implemented using the several **YOLO** models. For a single model like YOLOv11, it builds upon its predecessors with enhanced capabilities for feature extraction and detection accuracy. Key components of the architecture include:

- **Backbone**: A deep CNN for feature extraction, optimized for hierarchical representation of input images.
- **Neck**: Incorporates feature pyramid networks (FPN) and path aggregation networks (PAN) to enhance multi-scale feature detection.
- **Head**: Outputs bounding boxes, objectness scores, and class probabilities in a single forward pass, enabling real-time performance.

Hyperparameter Choices

To optimize model performance for the flames and smoke detection task, the following hyperparameters were chosen for **Yolov11**:

1. **Learning Parameters**:
    - **Epochs**: The model was trained for 50 epochs to ensure convergence without overfitting.
    - **Batch Size**: Implicitly determined by the `yolo` command, tailored to available GPU memory.
    - **Image Size**: 640 × 640 pixels, balancing detection accuracy and computational efficiency.
2. **Optimization**:
    - **Optimizer**: YOLOv11's default optimizer, typically a variant of stochastic gradient descent (SGD) with momentum and weight decay.
    - **Learning Rate Scheduling**: Adaptive learning rate scheduling to improve convergence.
3. **Detection Thresholds**:
    - **Confidence Threshold (conf)**: Set to 0.25 during prediction to filter low-confidence detections.
    - **IoU Threshold**: Default settings to evaluate overlaps between predicted and ground-truth bounding boxes.
4. **Data Augmentation**:
    - Implemented default augmentations such as random scaling, flipping, and color jittering to enhance generalization.
5. **Evaluation**:
    - Performance metrics, including mAP (mean Average Precision) and confusion matrices, were used to assess model efficacy.

Results and Discussion

We have run three different yolo models (Yolov8n, Yolov9s and Yolov11s). Results of these three models have given below:

Model Complexity

YOLOv8n:

- Layers: 168
- Parameters: 3,005,843
- GFLOPs: 8.1 (lowest computational cost)

YOLOv9s:

- Layers: 486
- Parameters: 7,167,475
- GFLOPs: 26.7 (highest computational cost)

YOLO11s:

- Layers: 238
- Parameters: 9,413,187 (highest parameter count)
- GFLOPs: 21.3 (moderate computational cost)

Analysis:

- YOLOv8n is the most lightweight model in terms of parameters and computational cost, making it ideal for resource-constrained environments.
- YOLOv9s is the heaviest in terms of GFLOPs, indicating higher computational demand but with a balanced parameter count.
- YOLO11s has the highest parameter count, which might require more memory but less GFLOPs than YOLOv9s.

Performance Metrics

- **Precision (P)**: All three models achieve perfect precision (1.0), indicating no false positives.
- **Recall (R)**: All three models achieve perfect recall (1.0), indicating no false negatives.

- **mAP@50**:
    - All models: 0.995 (identical high accuracy at IoU threshold of 50%).
- **mAP@50-95** (generalized accuracy):
    - YOLOv8n: 0.851
    - YOLOv9s: **0.856 (highest)**
    - YOLO11s: 0.845

Analysis:

- YOLOv9s performs slightly better in generalized accuracy (mAP@50-95), but the difference is marginal.
- YOLOv8n and YOLO11s are nearly identical in performance, with YOLO11s being slightly lower.

Efficiency vs. Accuracy

- **YOLOv8n**: Strikes a good balance between computational cost and performance, making it suitable for real-time applications on less powerful hardware.
- **YOLOv9s**: Delivers the best accuracy (mAP@50-95) but at the cost of higher computational demand.
- **YOLO11s**: Does not outperform YOLOv9s in accuracy and requires more parameters than YOLOv8n, making it less efficient overall.

Challenges and Future Scope

Challenges

1. **Complexity of Scenarios**:
    - The dataset comprises simulated smoke and fire in warehouse and server room environments. While useful, these simulations may not

fully represent real-world complexities such as varying lighting conditions, occlusions, or irregular smoke patterns.

2. **Dataset Imbalance**:
    - Potential imbalance in the dataset, such as over-representation of specific fire or smoke types, may bias the model and reduce its ability to generalize across diverse scenarios.
3. **False Positives and Negatives**:
    - Due to the dynamic nature of smoke and fire, the model occasionally struggled with false positives (e.g., mistaking reflections or light artifacts for fire) and false negatives (e.g., missing faint smoke).
4. **Real-Time Constraints**:
    - Deploying the model in real-time monitoring systems requires optimization for latency and computational efficiency, particularly in resource-constrained environments.
5. **Generalization**:
    - While the YOLOv8n model performed well on the training dataset, its performance in real-world, unseen environments remains to be rigorously tested.

Future Scope

1. **Enhancing Dataset Diversity**:
    - Incorporating real-world data from diverse environments with varying smoke and fire conditions will improve the model's robustness and generalization.
2. **Improving Model Accuracy**:
    - Experimenting with advanced architectures or hybrid models combining YOLOv11 with transformers or attention mechanisms could further improve detection accuracy.

3. **Integration with Alarm Systems**:

    - Integrating the detection model with smart alarm systems for automated hazard responses could enhance safety and efficiency.
4. **Multi-Hazard Detection**:
    - Extending the model's capabilities to detect other hazards, such as water leakage or structural damages, would make it a comprehensive safety monitoring tool.
5. **Real-World Deployment and Testing**:
    - Collaborating with industries to deploy and validate the model in operational warehouses and server rooms will provide insights into its real-world performance.