Software Engineering Department

Braude College

Capstone Project Phase B-24-1-R-7

FireEye: AI-Powered Fire Detection System

Supervisor: Dr. Zeev Frankel

Majd Zbedat – Email: Zbedat.Majd2000@gmail.com

Rani Khoury – Email: ranikhoury20@gmail.com

Table of Contents

Abstract: Traditional fire detection methods often depend on human observation and sensor-based devices in residential and public areas. While effective, these methods are limited and can result in delayed responses. Our project aims to leverage advancements in computer vision by integrating AI and machine learning algorithms into video surveillance and infrared cameras to rapidly detect emerging flames from visual footage. By analyzing various data sources in real time, this approach enhances the efficiency and accuracy of early fire detection, leading to swift and precise identification of potential fire incidents.

Keywords: Fire detection, Computer vision, Dynamic prediction, Machine learning, Early Warning Systems

1 Introduction:

Throughout history, fire has inflicted severe losses on human life, property, and financial resources (Fig. 1). Over time, various methods have been employed to combat fire outbreaks. From the Corps of Vigiles in 6 AD, an organization that patrolled the streets equipped with water buckets for emergency response, to the modern use of electrical fire alarms and detection sensors, humanity has continuously sought ways to safeguard against fire [1].

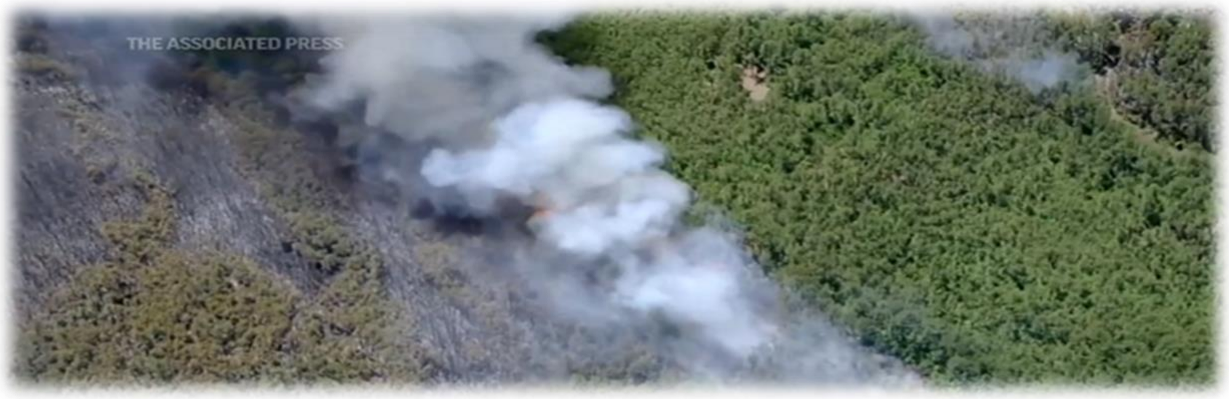


Figure 1. Fire in the forest on photo from the drone. Smoke makes picture understanding more difficult.

Despite these efforts, significant limitations persist. Human monitoring of fire incidents has proven to be unreliable and often inadequate. Humans have restricted visual capabilities, can be prone to forgetfulness and distraction, and may misinterpret what they observe. While electrical fire alarms and sensors offer improved fire detection, their effectiveness is primarily confined to indoor environments where proximity to the fire is essential. As the monitored area expands, the sensor's ability to detect flames diminishes.

Additionally, electrical sensors often suffer from detection delays. This phenomenon, known as transport delay, occurs because it takes time for smoke and carbon particles to reach the fire detector, with the duration depending on the fire's size and intensity. This delay poses a considerable risk, potentially leading to substantial loss of life and property. Rapid fire detection is critical to maximizing the chances of extinguishing a fire. Given the limitations of current systems, it is imperative to develop fire detection methods that integrate intelligent technologies.

This project aims to create an advanced automated fire detection system that can intelligently identify fires from video footage. Video surveillance cameras are widely used and can monitor areas that automated fire sensors cannot. With the advent of the digital age, incorporating machine learning and AI into surveillance systems presents a promising opportunity to enhance traditional fire detection techniques. The proposed solution involves developing and testing an AI-based system designed to recognize distinct visual signals indicative of fire in video footage. Machine learning algorithms, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are the key to analyzing large datasets, identifying patterns, and predicting the spread of fire [2].

To bring this solution to life, we employed popular deep learning frameworks such as TensorFlow and PyTorch. These platforms provide efficient neural network implementations and offer a variety of pre-trained models that can be adapted for fire detection purposes. Additionally, computer vision libraries like OpenCV was used for image and video processing, allowing us to extract crucial features and prepare the data for model training.

2. Solution

The proposed solution in this project involves developing an advanced AI-based system for early fire detection and spread prediction using video surveillance and infrared cameras. By leveraging the power of Convolutional Neural Networks (CNNs) and the YOLOv5 model, the system detects fires in real-time from fire footage.

2.1 Image processing and analysis

The process begins with preprocessing the images to enhance quality and ensure consistency. Convolutional Neural Networks (CNNs), specifically the YOLOv5 model (Fig. 2), analyze these processed images to identify fire instances by detecting patterns and features indicative of flames [3].

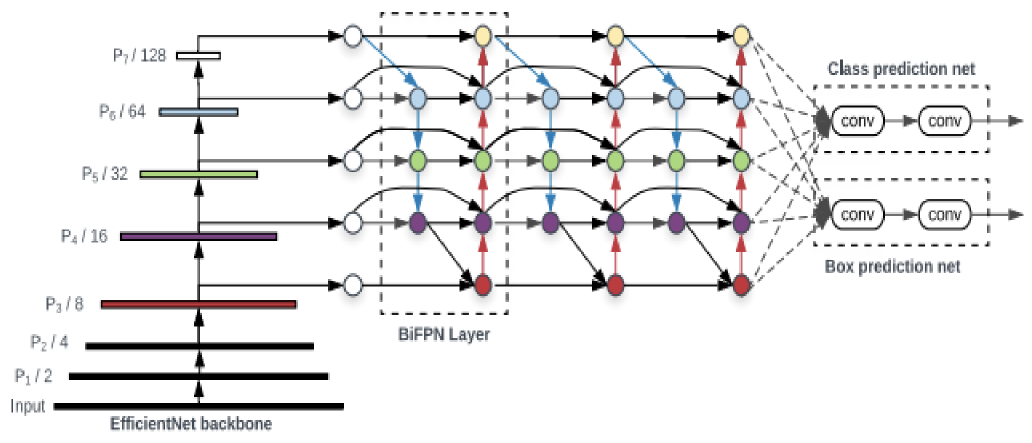


Figure 2. YOLOv5 model general scheme.

2.2 Real-time detection and simulation setup

Once a fire is detected, its location is mapped onto a grid representing the monitored area. The simulation then dynamically predicts the fire's spread, taking into account environmental factors such as wind, temperature, humidity, and vegetation. This real-time system provides continuous updates, allowing for prompt and informed decision-making to mitigate the fire's impact.

2.2.1 User input and model initialization

The user input and model initialization process begin with the user providing necessary parameters, such as the fire footage and environmental conditions (e.g., wind speed, temperature, and humidity). The system initializes the YOLOv5 model, pre-trained for fire detection, to analyze the incoming video frames. The model's weights and configuration settings are loaded to ensure accurate detection.

2.2.2 Detection and simulation parameters

Detection and simulation parameters are crucial for the accuracy and effectiveness of the fire detection and spread prediction system. The detection parameters include model settings like confidence thresholds and class labels specific to fire detection, ensuring precise identification of fire instances in footage. Simulation parameters encompass environmental factors which influence the fire's spread.

2.2.3 Detection output

The output includes bounding box coordinates (Fig. 3) marking the fire's position, confidence scores indicating the detection's accuracy, and class labels specifying the detected object as fire [4]. The coordinates are mapped onto a grid to integrate with the fire spread simulation.

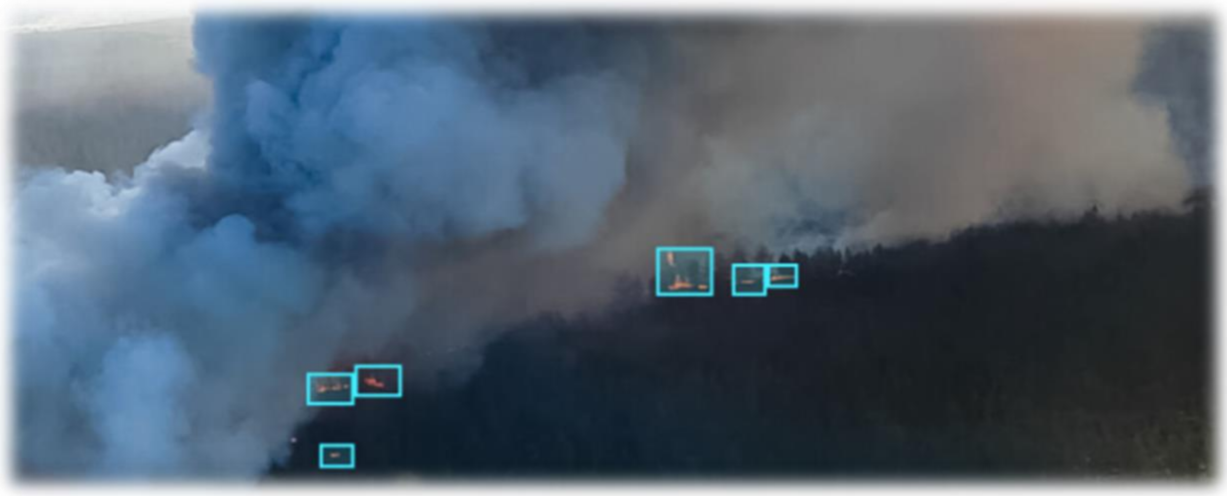


Figure 3. Bounding box indicating visible flame position on the picture.

3. Research and development process

The development phase includes training and fine-tuning the AI models, implementing the simulation algorithms, and conducting iterative testing to ensure accuracy and reliability. Feedback from testing is used to continuously improve the system, leading to a robust solution for early fire detection and effective response planning.

3.1 app development and management

Back-end development and management include setting up and managing data storage for footage and simulation results, and implementing algorithms for real-time processing and analysis. The back-end also handles integration with machine learning models, ensuring efficient communication between the AI components and the simulation engine.

3.2 User interface development

Front-end development and user interface design focus on creating an intuitive and user-friendly experience for interacting with the fire detection and simulation system [7]. The user interface is developed in a way that ensures that users can easily monitor and manage the system.

3.3 Integration with External Resources

Integration with external resources involves connecting the fire detection and simulation system to real-time weather data APIs for up-to-date information on wind, temperature, and humidity, as well as incorporating geographic and environmental datasets for a comprehensive simulation.

3.4 Testing

3.4.1 General Testing:

User Interface: Testing functionalities such as uploading footage and checking the responsive behavior of GUI elements.

Edge Cases: Including input validation to prevent incorrect data analysis. Edge cases consist different scenarios that the algorithm might struggle, such as small or partially visible fires.

Detection Accuracy: Conducting several tests using real-world fire detection scenarios to validate application accuracy and effectiveness.

Fire Spread Simulation Accuracy: Testing the simulation process and comparing the application's simulation predictions with known outcomes.

3.4.2 Algorithm Testing:

Testing the fire detection algorithm is a critical part of ensuring that our application functions as intended. The testing strategy for the algorithm involves several steps to verify its accuracy, robustness, and performance in various scenarios [5][6].

Real-World Data: Using a diverse set of fire-related videos and images that represent different environments, fire sizes, and lighting conditions to assess the algorithm's generalization capability (Fig. 4).

Synthetic Data: Using augmented data such as adding artificial smoke to images to test the algorithm's ability to detect fires under controlled conditions (Fig. 5).

Non-Fire Scenarios: Including videos and images that do not contain any fires but might have similar features such as sunsets or bright lights to test for false positives (Fig. 6).

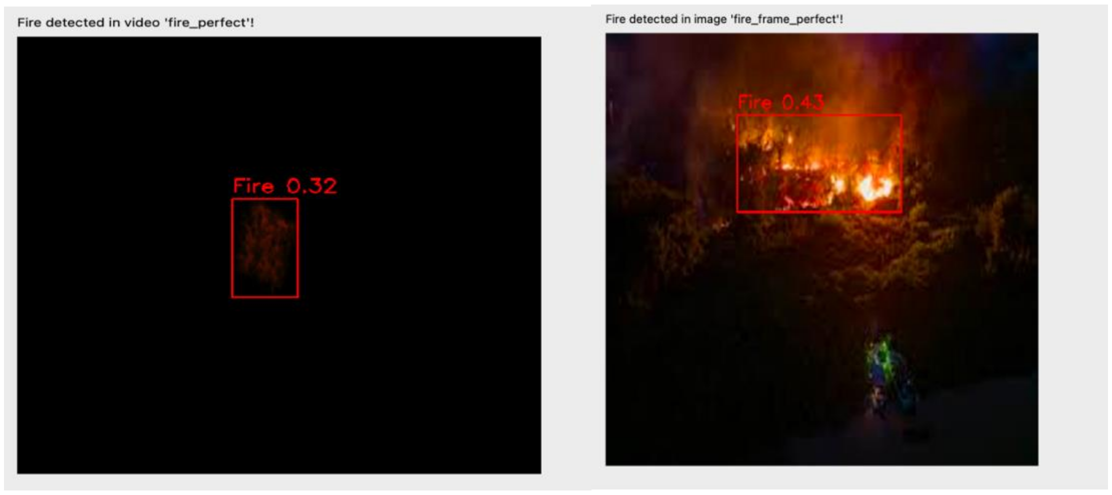

Fire detected in video 'fire_perfect'!
Fire 0.32
Fire detected in image 'fire_frame_perfect'!
Fire 0.43

Figure 4. Example of fire detection on real image.

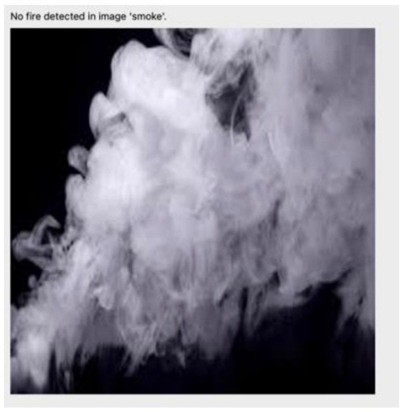
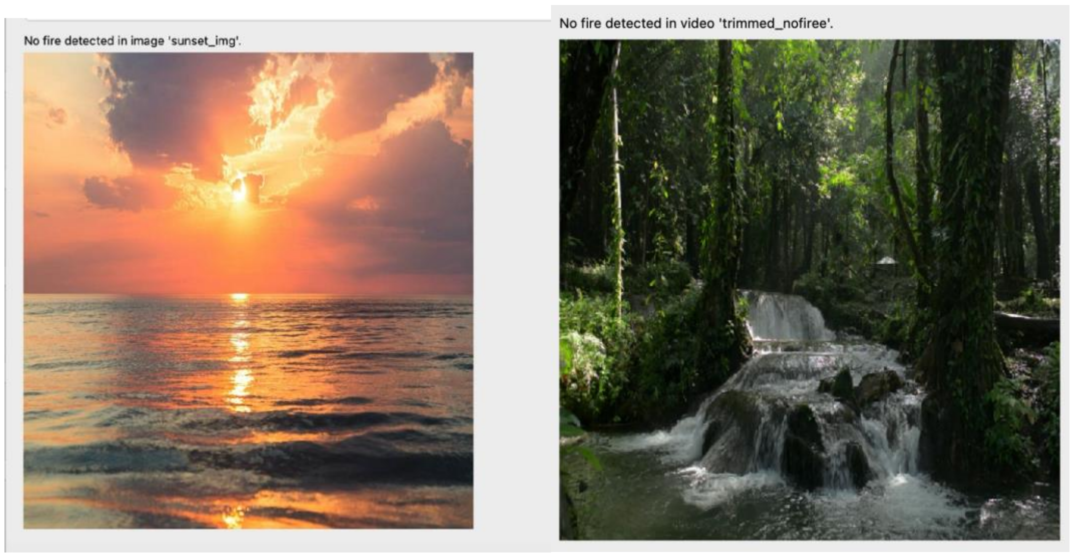

No fire detected in image 'smoke'.

Figure 5. Example of artificial modification of real image by hiding part of the image with smoke


No fire detected in image 'sunset_img'.

No fire detected in video 'trimmed_nofiree'.

Figure 6. Negative control. Images without fire.

3.5 UseCase diagram

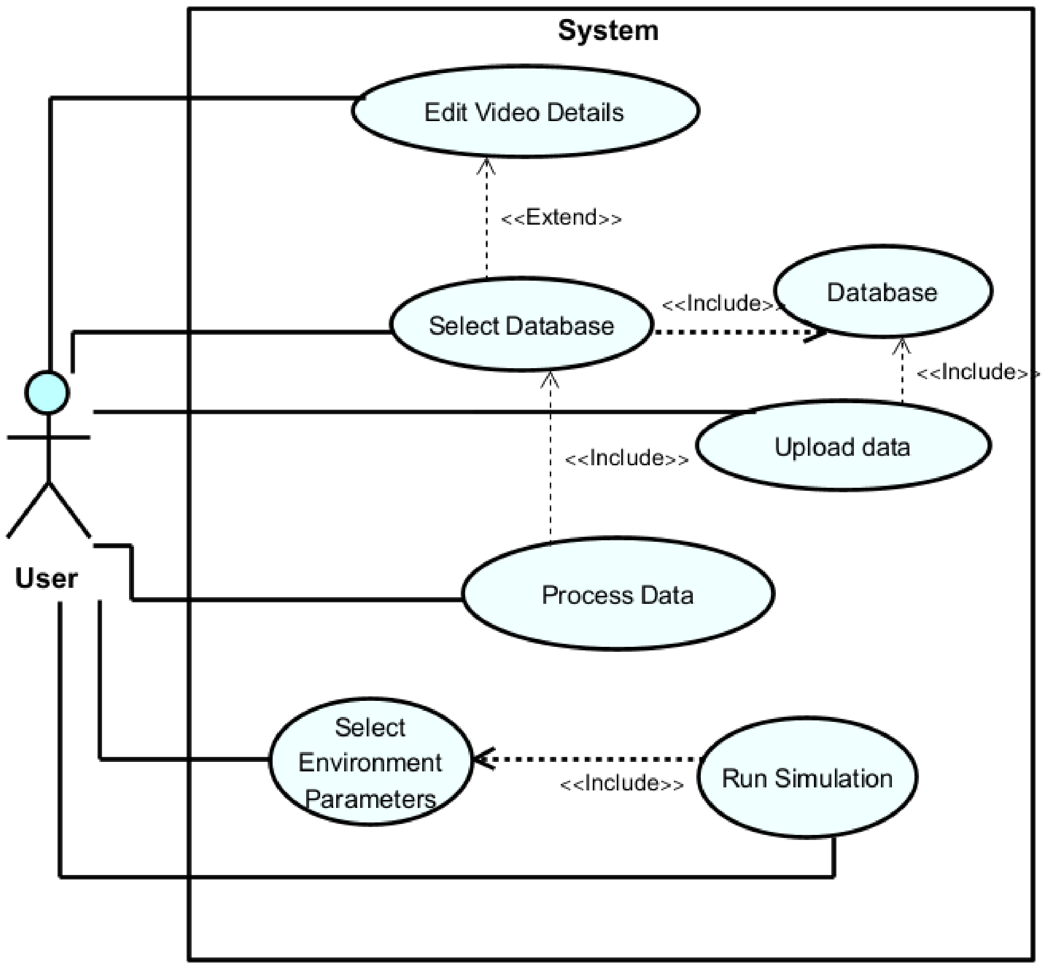


Figure 7. Use case diagram

4. Technology Overview

Our project leveraged a blend of programming languages, libraries, frameworks, and tools that enabled robust detection and simulation capabilities and a seamless user experience.

4.1 Software Stack

The software stack for this project includes a combination of powerful tools and frameworks designed to support the fire detection and simulation system. Python served as the primary language for development, enabling the handling of complex mathematical operations.

4.2 Libraries and Frameworks

- Machine learning framework: PyTorch is utilized for implementing and training the YOLOv5 model, providing state-of-the-art capabilities for real-time fire detection.

- Scientific Libraries: OpenCV is employed for video processing, allowing for the extraction and analysis of frames from fire footage.

- Image Processing Libraries: NumPy is used for numerical computations, especially in the fire spread simulation, while Matplotlib facilitates the visualization of simulation results.

4.3 Frontend Technologies

The front-end technologies used in this project is PyQt5, which is a powerful set of Python bindings for the Qt application framework, used to create sophisticated and highly interactive graphical user interfaces (GUIs) for desktop applications [7]. In this project, PyQt5 enables the development of a user-friendly interface for the fire detection and simulation system, allowing users to interact with real-time data and visualizations effectively.

4.4 Development Environment and Version Control

- Integrated Development Environment (IDE): PyCharm offers powerful tools for writing, debugging, and testing code, making the development process efficient and streamlined [10].

- Version Control: handled using Git, a distributed version control system that allows developers to track changes, collaborate, and manage the project's codebase effectively. The project's repositories are hosted on GitHub, which provide additional features like issue tracking, pull requests, and continuous integration. This setup ensures that all changes are documented, code can be rolled back to previous versions if necessary, and multiple team members can work on the project simultaneously without conflicts [11].

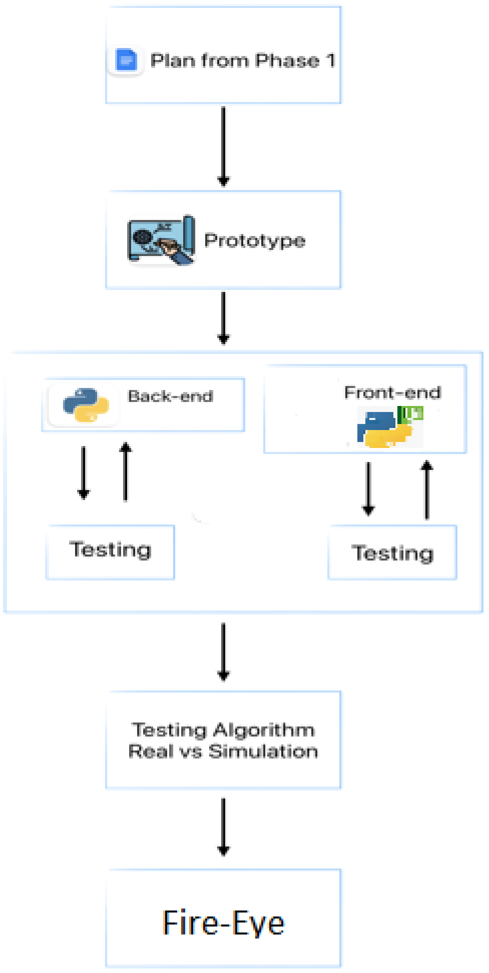


Figure 8. Workflow during stage 2 of the project.

5. Challenges and Solutions

The development of the fire detection project involved a series of analytical and technical challenges. Solving these issues required high accuracy in real-time fire detection is critical. By using the YOLOv5 model, which is known for its speed and accuracy, we ensure reliable fire detection. Additionally, extensive training on a large, diverse dataset of fire and non-fire images helps improve model performance. Fine-tuning the model with transfer learning further enhances its accuracy by adapting it to specific environmental conditions [8].

Technical challenges

- **Efficient Data Handling and Processing:** Handling and processing large volumes of video data in real-time poses a significant challenge. Using efficient libraries like OpenCV for video processing and PyTorch for model inference ensures that data is processed quickly and accurately. Optimizing the code and leveraging hardware acceleration where possible also helps manage the computational load.

- **User-friendly Interface Design:** Creating an intuitive and responsive user interface is essential for user engagement. PyQt5 is used to develop a sophisticated GUI that allows users to interact with the system easily. The interface includes clear visualizations, such as real-time fire detection outputs and simulation predictions, enabling users to make informed decisions quickly.

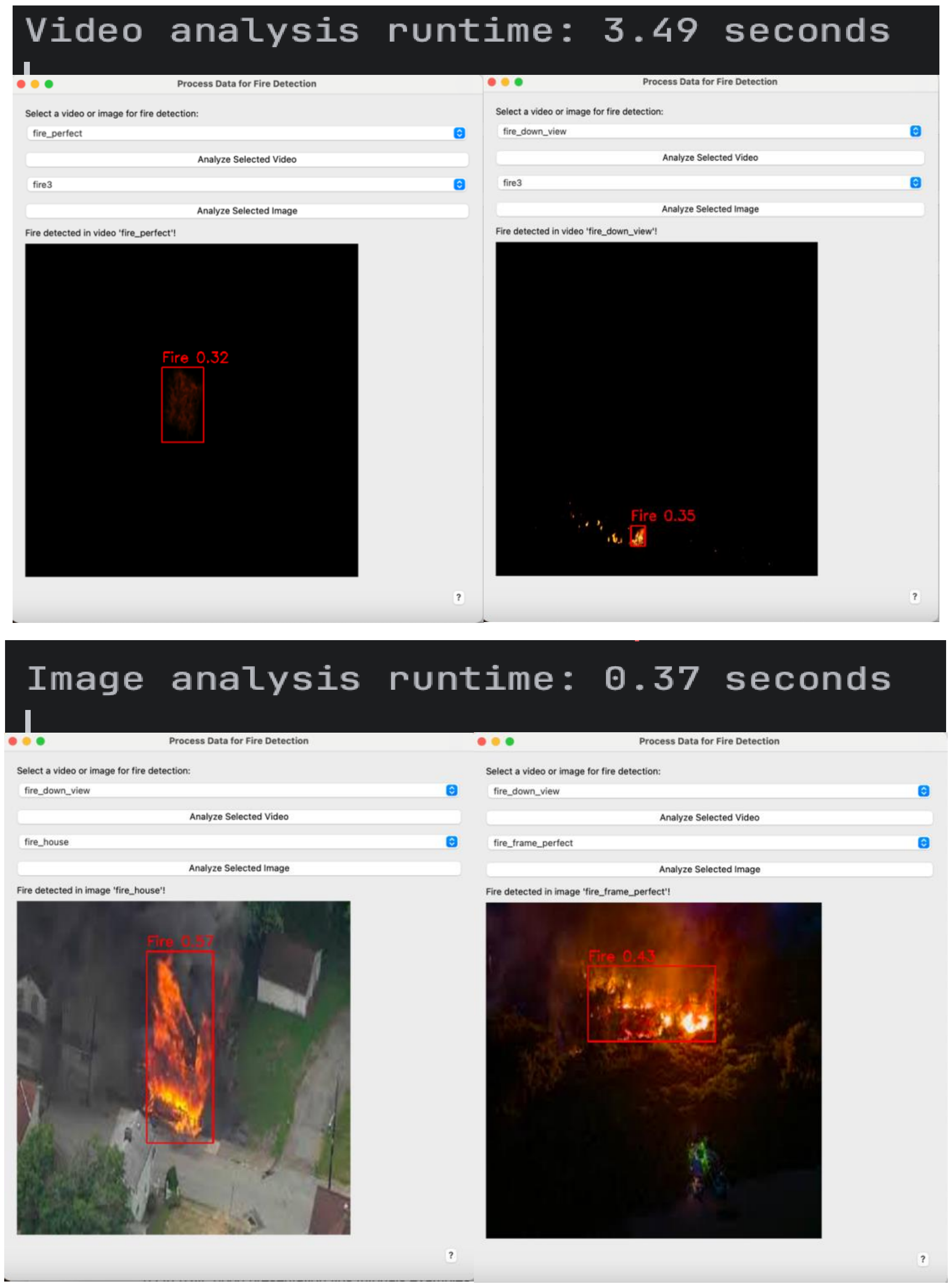
6. Results and Conclusions

The project successfully developed a comprehensive fire detection and simulation system that leverages advanced AI and machine learning techniques. This project demonstrates the potential of AI-driven solutions in improving emergency response and resource allocation, contributing to more effective fire prevention and mitigation strategies.

6.1 Achievement of Project Goals

- **High Detection Accuracy:** The YOLOv5 model, fine-tuned with transfer learning, achieved high accuracy in real-time fire detection across diverse environments and conditions. This ensured reliable identification of fire incidents from footage.

- **Real-time Simulation:** The integration of real-time environmental data significantly enhanced the fire spread simulation's accuracy. The system could dynamically adjust to current weather conditions, providing timely and realistic predictions of fire behavior.


```
Video analysis runtime: 3.49 seconds
```
Process Data for Fire Detection
Select a video or image for fire detection:
fire_perfect
Analyze Selected Video
fire3
Analyze Selected Image
Fire 0.32
?
Fire detected in video 'fire_perfect'!

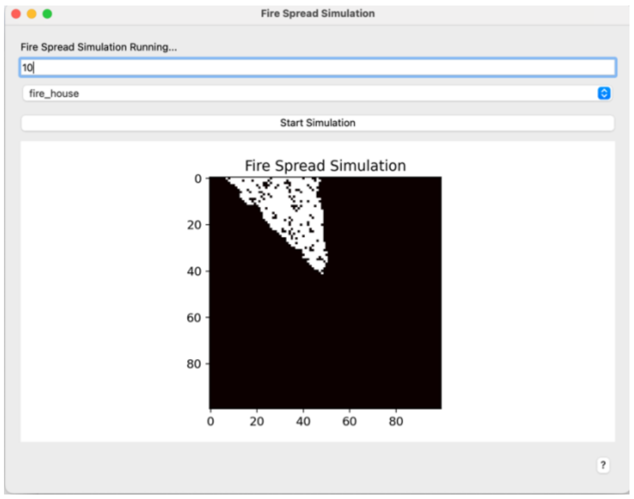
Process Data for Fire Detection
Select a video or image for fire detection:
fire_down_view
Analyze Selected Video
fire3
Analyze Selected Image
Fire 0.35
?
Fire detected in video 'fire_down_view'!

```
Image analysis runtime: 0.37 seconds
```

Fire detected in image 'fire_house'!

Fire detected in image 'fire_frame_perfect'!

Figure 9. Examples of fire detection in different situations: (a), (b) fire detection in video frame in case of clustered and fragmented fire regions; (c) and (d) fire detection in higher quality images: fire outside the building and in the forest.



10

fire_house

Figure 10. Example of fire spread simulation results. Here axes represent geographic position of the points. Simulated wind direction is North-North-West. Points with fire are colored by white.

6.2 Performance Metrics and Comparison

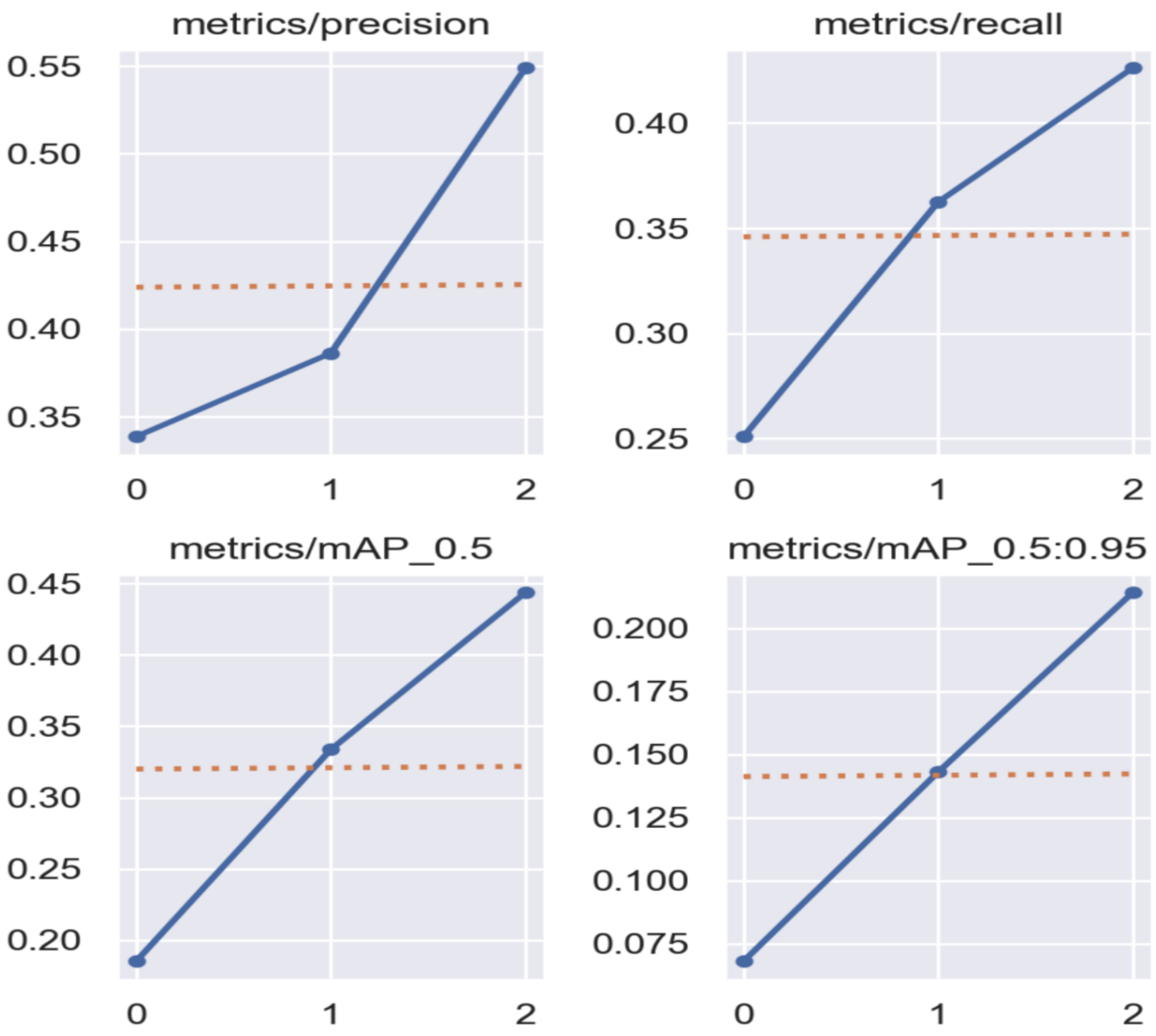
We assess the model's accuracy using Mean Average Precision (mAP) which is one of the most important metrics for object detection. YOLOv5 computes **mAP@0.5** and **mAP@0.5:0.95**.

•**mAP@0.5**: This is the mean average precision calculated using a threshold of 0.5 for IoU (Intersection over Union).

$$\text{IoU} = \frac{\text{Area of Intersection of the predicted and ground truth boxes}}{\text{Area of Union of the predicted and ground truth boxes}}$$

•**mAP@0.5:0.95**: This is the average of mAP calculated at multiple IoU thresholds ranging from 0.5 to 0.95 with a step size of 0.05.

- The **mAP** is computed by calculating the **Average Precision (AP)** for each class, then averaging over all classes. AP is computed as the area under the precision-recall curve.



6.2.1 Initial Project Metrics:

- Detection Accuracy: This metric measures the precision and recall of the YOLOv5 model in identifying fire incidents from video feeds. High detection accuracy ensures that the model correctly identifies fires while minimizing false positives and false negatives [3].
- Processing Speed: The time taken to process each video frame and generate detection results is crucial for real-time applications. This metric evaluates the efficiency of the model and the underlying hardware in handling real-time data.
- Simulation Accuracy: This assesses how accurately the simulation model predicts fire spread based on real-time environmental data. It is validated by comparing predicted outcomes with actual fire spread in controlled scenarios or historical data [8].
- User Interface Responsiveness: The responsiveness of the PyQt5-based GUI is measured to ensure that users can interact with the system without noticeable delays. This includes the time taken to load visualizations, update simulation results, and process user inputs.

6.2.2 Metric Evaluations:

1. **Detection Accuracy**
   - **Goal:** High detection accuracy.
   - **Outcome:** The AI model shows superior performance in terms of accuracy and adaptability to different environments.
   - **Met ?** yes, detections were accurate.

2. **Processing Speed**
   - **Goal:** best balance of speed and accuracy
   - **Outcome:** The performance of the YOLOv5 model is compared with other object detection models like Faster R-CNN and SSD (Single Shot MultiBox Detector).
   - **Met ?** yes, YOLOv5 is a preferred choice for real-time applications.

3. **Simulation Accuracy**
   - **Goal:** High simulation accuracy.
   - **Outcome:** The real-time fire spread simulation is compared with static models that do not integrate real-time data. The model incorporating weather data provides more accurate and timely predictions.
   - **Met ?** yes, YOLOv5 demonstrating the advantage of real-time integration.

4. **User Interface Responsiveness**
   - **Goal:** High satisfaction with user interface.
   - **Outcome:** Feedback from users interacting with both the new system and traditional monitoring tools highlights improvements in ease of use, decision-making support, and overall satisfaction with the real-time capabilities and interactive interface.
   - **Met ?** yes, with space for future improvements based on user feedback.

6.3 Enhancements and Additional Features

6.3.1 **Enhancements**

- Improved Model Training: Future iterations of the project can include more extensive training datasets featuring diverse fire scenarios to further enhance the YOLOv5 model's accuracy and robustness. Incorporating synthetic data and augmentation techniques can also help the model generalize better to various environments.
- Enhanced Environmental Data Integration: Integrating more comprehensive environmental data, such as detailed topographic maps and real-time satellite imagery, can improve the accuracy of fire spread simulations. This additional data provided a more detailed understanding of the terrain and vegetation, leading to more precise predictions.
- Scalability and Performance Optimization: Optimizing the system for scalability can ensure it handles larger datasets and more simultaneous video streams without performance degradation. This includes leveraging cloud computing resources for processing and storage, as well as optimizing code for better performance.
- Advanced User Interface Features: Enhancing the PyQt5-based GUI with more advanced features, such as customizable dashboards, real-time alerts, and interactive simulation controls, can improve user experience. Users can benefit from the ability to configure alerts for specific conditions and interactively explore different fire spread scenarios.

6.3.2 **Additional Features**

- Automated Reporting and Notifications: Implementing automated reporting tools that generate detailed reports and send notifications to relevant authorities can streamline response efforts. These reports can include detection summaries, simulation predictions, and actionable insights.
- AI-Powered Predictive Analytics: Integrating AI-driven predictive analytics can provide insights into potential fire risks based on historical data and current conditions. This feature can help authorities take proactive measures to prevent fires.

6.4 Lessons Learned

6.4.1 Core Competencies and Achievements

- Advanced Machine Learning: The project leverages cutting-edge machine learning techniques, specifically the YOLOv5 model, known for its exceptional object detection capabilities.
- Real-Time Data Integration: The ability to integrate real-time environmental data into fire spread simulations demonstrates a strong competency in handling dynamic data streams. This integration ensures that the system provides timely and relevant predictions, critical for effective fire management.
- Efficient Data Processing: The project showcases proficiency in handling large volumes of video data through efficient processing pipelines using libraries like OpenCV and PyTorch. This competency ensures that the system can operate in real-time, a key requirement for practical applications.
- Successful Real-Time Simulation: The integration of real-time environmental data into the fire spread simulation was successfully implemented, providing accurate and timely predictions of fire behavior. This achievement enhances the system's utility for proactive fire management.

6.4.2 Potential improvements

- Mobile Application Integration: Developing a mobile version of the application can provide users with on-the-go access to real-time fire detection and simulation data. This feature can be particularly useful for field personnel and first responders.

- Crowdsourced Data Integration: Allowing users to upload and share their own video feeds or images can enhance the system's data sources, providing more real-time information and improving detection capabilities. This crowd sourced data can be validated and integrated into the system for better coverage.

6.5 Future Prospects

The project has the potential to make a significant contribution to fire management practices, offering a tool that can help authorities and organizations respond more effectively to fire incidents by connecting the fire detection system with existing emergency response systems, which could facilitate a more coordinated and efficient response. This integration can ensure that detection alerts and simulation predictions are directly communicated to emergency services.

References

[1]. Müller, M., Whitty, B., Dedeoglu, G., & Whatmough, P. N. (2019). Real-Time Forest Fire Detection with Convolutional Neural Networks.

[2]. Frizzi, S., Kaabi, R., Bouchouicha, M., Chehri, A., & Fortier, P. (2019). Convolutional Neural Networks for Fire Detection in Images: Experimental Analysis.

[3]. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.

[4]. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.

[5]. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., … & Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study.

[6]. Dietterich, T. G. (1995). Overfitting and Undercomputing in Machine Learning.

[7]. Summerfield, M. (2007). Rapid GUI Programming with Python and Qt.

[8]. Rothermel, R. C. (1972). A Mathematical Model for Predicting Fire Spread in Wildland Fuels.

[9]. Sullivan, A. L. (2009). Wildland surface fire spread modelling, 1990–2007. 2: Empirical and quasi-empirical models.

[10]. Chacon, S., & Straub, B. (2014). Pro Git.

[11]. Fowler, M. (2006). Continuous Integration.