VISION-BASED FIRE DETECTION AND PREVENTION SYSTEM FOR WAREHOUSE SAFETY USING REAL-TIME CAMERA SURVEILLANCE

P.A.S.Tharana

(IT21822094)

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

VISION-BASED FIRE DETECTION AND PREVENTION SYSTEM FOR WAREHOUSE SAFETY USING REAL-TIME CAMERA SURVEILLANCE

P.A.S.Tharana

(IT21822094)

BSc (Hons) degree in Information Technology
Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

August 2025

DECLARATION

I declare that this is my work. This proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or institute of higher learning. To the best of my knowledge and belief, it does not contain any previously published material written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
P.A.S.Tharana	IT21822094	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: _____ Date _____

Signature of the co-supervisor: _____ Date _____

ABSTRACT

This project introduces a vision-based fire monitoring system that taps into the integrity of computer vision technology paired with surveillance cameras to improve safety protocols in warehouses. By utilizing the capabilities of surveillance cameras and identifying fires in real time from the data captured, the project's research is limited to the capabilities of banks of surveillance cameras to create a fire monitoring model. The monitoring system will have four main functions: identify if a fire is detected, determine if there are any nearby shelves in the direct path or within the distance of the fire, identify the size of the fire (small, medium or large), and potentially identify the expected path of fire spread. Each function allows warehouse employees to receive a valuable warning with knowledge of a safe distance relative to a fire.

The project is developed in Python with opencv and integrated with deep learning objects in Tensorflow. The fire detection operates through contour identification and image classification, while shelving detection operates through distance identification and object detection to estimate nearby objects. Fire size is estimated through the heat generated by a pixel area and fire spread is based on environmental patterns as identified by frame sequences where potential spread may occur.

As a total camera only solution, the presented fire monitoring solution is non-disruptive, scalable, and economical, eliminating the need for hiring for IOT hardware and maintains a high monitoring accuracy. Real-time performance is achieved through the established architecture, and data security problems for own monitoring footage are considered.

Keywords: Fire detection, Warehouse safety, Computer vision, Real-time surveillance, Fire size classification, Shelf proximity detection, Fire spread prediction, Camera-based monitoring, opencv, tensorflow, Deep learning, Object detection, Contour analysis, Video analytics, Fire risk mitigation

ACKNOWLEDGEMENT

First I would like to express my deepest gratitude to my supervisor, **Prof. Samantha Rajapaksha**, for his continuous guidance, encouragement, and invaluable feedback throughout this project. I also wish to thank **Dr. Dinuka Wijendra** for his support and constructive suggestions that helped refine this work. Finally, my sincere appreciation goes to the **academic staff of SLIIT – Faculty of Computing, Department of Information Technology**, for providing the knowledge, resources, and infrastructure necessary to complete this research.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Network – a type of deep learning model commonly used for image classification and detection.
YOLO	You Only Look Once – a real-time object detection algorithm that processes images in a single pass through the network.
YOLOv5	A popular and optimized version of the YOLO model for object detection, known for its speed and accuracy.
OpenCV	Open Source Computer Vision Library – a popular library of programming functions used for real-time computer vision.
Flask	A lightweight Python web framework used to develop web applications and APIs.
RNN	Recurrent Neural Network – a class of neural networks used for sequence prediction, such as video or time-series data.
IoT	nternet of Things – a network of physical devices embedded with sensors, software, and connectivity to collect and exchange data.
Bounding Box	A rectangular box that encloses an object in an image, used in object detection to localize and label targets.

1 List of abbreviations

1. INTRODUCTION

Warehouses are an important component of the global supply chain, and serve as storage facilities for goods, raw materials and key inventory. Warehouses are typically very unsafe because of potential fire dangers. Fire dangers in warehouses present risks not only in terms of lives lost and harmed, but also in terms of significant economic losses and disruptions to supply. Warehouses have traditionally relied on smoke detectors, thermal sensors and sprinkler systems as their fire safety systems. However, fire dangers in warehouses can exist before smoke or heat can be detected. In some cases of fire danger, there can be no smoke or considerable heat transfer. Most importantly, fire safety monitoring in warehouses often lack early warning and d spatial awareness, especially for large or cluttered environments.

The surveillance industry's integration of computer vision and artificial intelligence presents a possible way to improve fire surveillance systems, which monitor fires and measure human response. This project proposes a camera-only, real-time fire surveillance system, which uses computer vision methodologies to identify events of fires and assess them in the context of a warehouse, in real-time, without requiring any Internet of Things (IoT) hardware. This project uses the camera system already existing, and is non-intrusive to the environment in cost and monitoring.

Our application employs four key functions: first, it determines whether there is a fire; second, it identifies nearby shelves and the proximity of those shelves to the fire; third, it determines if the fire is classified as small, medium, or large based on size; and finally, it predicts which direction the fire will spread. This information enhances the awareness of warehouse employees so they can act quicker and in a more strategic way, potentially averting catastrophic loss.

This system was developed using Python, OpenCV, and TensorFlow, where we combine a video stream, process that video in real time, use a deep learning model to enable intelligent decisions, and use visual information to allow widespread usability with a low overhead operational and scale.

This research intends to bridge the gap between the way we typically respond to a fire, also known as pre-modern responses (water and aggressive attack methods), to the capabilities of contemporary computer vision, and to propose a proactive, data-driven method of safety specific to a warehouse setting.

1.1 Background

Fire risks in industrial and storage facilities remain a significant concern globally. Warehouses houses various hazardous materials that are largely flammable and often have insufficient human supervision. Fire alarm systems (e.g., smoke detectors, heat sensors, and sprinklers) are reactive measures that may not provide adequate spatial context or analytical information about important prior events before a fire becomes unmanageable. For these reasons, researchers have become increasingly interested in using computer vision and deep learning techniques for fire monitoring and warning systems.

There are studies, including Jin and coworkers [1], that have shown how accurate camera fire detection can be developed through the use of deep learning, which can accurately identify flames, smoke and/or abnormal visual signals based on pixels in images in real-time. Jin et al. [1] provided an extensive review of video fire detection approaches, and emphasized the promise of convolutional-neural-networks in identifying fire scenes from the video surveillance footage. There are additional studies, much like those conducted by Martins [3] and Islam and Habib [5] showing that YOLO based object detection models can provide real-time classification to fires and do so in very low latency settings. Even lightweight models like Light-YOLOv5 models can be used when fire detection activities need to be reported in low-resourced contexts [4].

Along with detecting the presence of fire, estimating fire size and location in a scene is valuable in real-world applications. Pixel-based area estimation and contour analysis used to assess fire size, such as in [10], allow us to categorize fire severity and establish appropriate actions in relation to risk. Also, using segmentation bounding box and heat-based modeling have been considered.

Additionally, when it comes to fire safety, proximity of nearby structures, such as storage shelves or racks, plays an important role. In the area of retail and logistics research, it has been suggested that AI-based shelf detection can be used to monitor levels of inventory and locate structural elements of an environment [7][8]. These

techniques could also be used to determine key warehouse assets and the distance from the fire, thus gaining spatial awareness for emergency response planning.

Moreover, predicting the direction of fire spread continues to be one of the most advanced and emerging area of fire safety development. The research with studies [11], [12], [14] where deep learning and generative models have been used to predict whether and how fires would progress using environmental signals and video analysis frame by frame. Plus, as in Stanford's project [13], computer vision can provide a way to visualize fire events in varying dynamic environments, which leads into how this could be useful to a warehouse environment where based on predictions, the occupants could enter evacuate routes or fire suppression routes.

At last, the developing range of systematic reviews [2][15] supports that vision-based fire detection is not only feasible but legitimate, scalable, and flexible to a variety of domains. These reviews suggest using only camera systems, as they are low-cost, low-maintenance alternatives to sensor-based networks, particularly in facilities with existing surveillance systems.

This growing area of research is the basis on which the proposed system, which intends to combine much of this work into a coherent fire monitoring system for warehouse situations. This project includes fire detection, shelf proximity, fire size classification, and fire spread prediction within one framework, and as a result of this intent, the proposed system closes several important safety gaps in current warehouse fire response systems.

1.2 Literature Survey

This chapter provides a review of relevant literature focused on the four major components of the proposed fire monitoring system, which are fire detection, the detection of shelf proximity, the classification of fire size, and the prediction of fire spread. Each of these components are reviewed in order to identify what is currently known, and the gaps that currently exist in fire research, along with the implications for warehouse environment considerations.

1.2.1 Fire Detection

Fire detection using computer vision as a detection tool for fire has been established as a promising alternative to traditional sensor detection systems. Jin et al. [1] provided a review of video-based fire detection methods and indicated that it is possible to detect flames, smoke, and abnormal visual patterns using CNNs from surveillance footage. Their review pointed to the role of the deep learning methods for good real-time detection while maintaining high accuracy.

Martins [3] provided evidence of the ability to detect fire in video streams from a You Only Look Once (YOLO) object detection model. Islam & Habib [5] demonstrated the success of YOLO-based object detection models as a good tool to detect fire in challenged environments (variable light and motion). Xu et al. [4], continued development in fire detection and introduced Light-YOLOv5, a lightweight variant of YOLO for devices with limited computing power, which has many implications regarding the scalability of a warehouse.

1.2.2 Shelf Detection and Distance Estimation

Though largely discussed in retail and logistics scenarios, shelf detection and visual distance estimation are relevant in fire monitoring in determining the distance a given fire is from critical storage infrastructure. A deep learning-based system for monitoring shelves is discussed in [7] where the authors conducted their work utilizing an object detection model to detect and track shelves for inventory purposes. Kumar [8] created a system based on AI that located empty shelves through vision-based spatial assessments to trigger notifications for restocking purposes.

These methodologies may be used for explicit warehouse fire safety, potentially by estimating the distance from detected fire zones to shelving units to provide spatial awareness for better emergency planning.

1.2.3 Fire Size Classification

Estimate of fire size is important when assigning a severity level to a fire and therefore how to respond. The traditional techniques used to generate a spatial estimate

of the fire, such as measuring pixel area and detecting the contour of the fire [10], can be adopted, but there was also an approach in this study that provided estimates of fire size using image segmented, contour-based models, that could be measurably classified as small, medium, or large. Additionally, heat based modeling and bounding boxes have been discussed in literature as secondary approaches to further support the classification of stages of fire intensity and periods of fire growth.

1.2.4 Fire Spread Prediction

Fire spread prediction is a newer and developing area in vision-based fire monitoring. More recent studies such as Jin et al., [11] and Stanford's CS231n project [13], have proven that machine learning and deep learning models can analyze visual environmental cues and temporal patterns to predict where a fire will move, and at what speed. There have also been examples using U-Net models for fire forecasting, which are a type of deep learning model used, again with predicted outcomes for the fire front in [12]. Moreover, as outlined in [14], generative deep learning approaches has allowed for some degree of two and three-dimensional modelling of fire dynamics temporally for possible evacuation or containment.

1.3 Research Gap

In recent years, the results of research and innovation in computer vision and deep learning have revolutionized automated fire detection systems - but, their application for safety in warehouses is limited, fragmented, and not well understood. Each existing system typically focuses on one or two isolated tasks, rather than integrating an accurate understanding of the spatial and operational relationship between all objects in the warehouse. A thorough examination of the literature identified several existing gaps that the project intends to address.

1.3.1 Limited spatial context in advanced fire detection systems

The majority of camera-based automated fire detection systems are able to locate flames or smoke with real-time CNNs and/or object detection models [1][3][5] but are often designed for open environments or generic contexts, hence lacking spatial

context, including which nearby objects may be at risk from the fire and the intended consequences of that fire's location. A critical aspect to fire detection in warehouses, particularly when considering the dense packing of materials and assumed risk of flammability, is understanding spatial context. Without knowing what it is that is at risk, and how close the fire is to the critical infrastructure of a warehouse, it will have limited practical use when an immediate emergency occurs.

1.3.2 Fire spread prediction models are developed for outdoor fires, not warehouse fires

Using deep learning to predict fire spread is an active and developing field [11][12][13][14]. Many models have been trained using outdoor wildfire data, which uses vegetation, wind, and topography data to estimate fire spread. However, outdoor fire spread prediction models are not ideal for indoor conditions since indoor environments such as warehouses have distinct airflow patterns, narrow corridors, and vertical stack storage that can make fire spread behavior very different from non-contained outdoor fires. Existing fire prediction models fail to model the environmental variables and spatial constraints of indoor warehouse structures.

1.3.3 Fire size estimate is rarely connected to response prioritization

Some studies have examined estimating fire size with pixel-based contour analysis [10], which is based on the severity of fire; small, medium, or large. However, these size estimates are usually presented as outputs without a connection to a larger fire response system. There is a notable gap in the use of fire size information when triggering alarms, mobilizing resources, or warning staff in a warehouse environment. This prevents these types of classifications from being operationally useful in practice.

1.3.4 Shelf detection methods are not adapted for fire-related risk assessment

Shelf and rack detection systems in retail logistics [7][8] use object detection to track product availability or assist with automation of replenishment of stock. They are not built to function in the fire-related risk contexts, as they rely on the proximity of fire to a structure, such as a high storage rack, without consideration for safety. In an emergency with fire, shelf proximity assessment (in real-time) could assist with fire

suppression, ways for people to evacuate safely, or even for automation to notify authorities, using a system that doesn't exist today.

1.3.5 Absence of a unified, camera-only, real-time monitoring system for warehouses

While each of the four areas has improved individually, no system has the ability to anchor fire detection to shelf proximity estimates, classify fire size, and predict the spread of fire in a single framework, especially using only surveillance cameras. Current systems are structured around either IoT sensors, thermal devices or require costly infrastructure upgrades. There is a unique opportunity to develop a low cost, scalable camera only solution developed specifically for the fire safety requirements of warehouse settings.

Comparison Criteria	Existing Fire Detection Systems	Fire Spread Models (Wildfire)	Shelf Detection (Retail/Logistics)	Unified Camera-Only Systems	Proposed System
Fire Detection (basic flame/smoke recognition)	✔	✔	✗	✔	✔
Spatial Context Awareness (objects/shelves at risk)	✗	✗	✔	✗	✔
Fire Spread Prediction (for warehouses)	✗	✔ (outdoor only)	✗	✗	✔
Fire Size Classification (linked to response)	✔ (limited)	✗	✗	✗	✔
Shelf Proximity Analysis (for fire risk)	✗	✗	✔ (inventory use only)	✗	✔
Unified, Real-Time Camera-Only Monitoring	✗	✗	✗	✗	✔

2 Research Gap Summary

1.4 Research Problem

Warehouses are essential parts of many industries and logistics, and they can house a tremendous amount of cargo—especially flammable goods stored at high density. Even given their importance, warehouse fire safety systems typically consist

only of smoke detectors, heat sensors, and sprinklers which are often just reactive devices and systems. These types of systems are effective with little to no spatial awareness or predictive capability, and more often respond when the fire is at its peak and poses a substantial risk to personnel, goods, and operations.

Recent development and applications of computer vision and deep learning may enable real-time fire detection through video seemingly from off-the-shelf surveillance footage [1], [3]. There have been successful models based on a convolutional neural network (CNN), and YOLO-based detectors, detecting visual indicators of fire [5]. There is even a lightweight model, Light-YOLOv5 [4], suitable for many usages with platforms having limited hardware. Nevertheless, all of these systems overwhelmingly focus on fire detection but provide almost no information in relation to the environment associated with that fire.

Furthermore, while shelf and rack detection systems have previously been analyzed during logistics and retail applications [7][8], they have not been taken into consideration for emergency fire applications. For instance, no system analyzes how far less-than stable shelving or expensive materials are away from the active fire, which could change risk assessments and response plans completely.

In addition, while pixel-area analysis and contour detection have been proposed as reliable measures for fire size estimation [10], only a few systems incorporate those measurements into practical decision-making support tools for emergency protocols or resource prioritization.

Also, fire spread prediction models have been trained primarily for outdoor settings like forests [11][12][14], where wind and rough geology interact. The indoor environment of warehouses involves airflow, shelf stacking and human spatial limits that differ and complicate modeling approaches. Even promising projects like Stanford's CS231n application [13] fall short of exploring indoor fire dynamics using camera only approaches.

while these categories of research independently and collectively have advanced, there now exists no unified (camera-only) solution integrated with fire detection, shelf proximity estimation; fire size classification; and fire spread prediction in real-time

(for warehouse spaces). Prior solutions are either part, sensor-agnostic and lots know not customized to the space and time conditions in warehouses.

1.5. Objectives

1.5.1 Main Objective

To produce and office a holistic, real-time fire monitoring service using existing surveillance camera infrastructure that merges fire detection, shelf proximity analysis, fire size rating, and fire progression prediction - a pre-emptive safety system designed for warehouses.

1.5.2 Specific Objectives

1. **To develop a fire detection algorithm utilizing deep learning based on video feed**

   The goal is to leverage computer vision methods such as Convolutional Neural Networks (CNN), as well as Object Detection based on YOLO, to determine accurately whether there is fire (flame / smoke) present based on a live feed to the camera. The resulting deep learning model will be trained and evaluated to accommodate the factors such as lighting changes, clutter and motion commonly associated with warehouses, and it will alert immediately once it detects the fire.

2. **To implement shelving detection, and proximity estimation**

   The second module will represent the shelving units, and provide the spatial distance of the shelving units to the fire based on object detection (e.g., YOLO, SSD), the bounding box coordinates, and image depth methods. Consequently, we will have situational awareness of which items are at risk, and can inform emergency responders of the items to prioritize to contain the fire or start evacuation.

3. **To classify fire size based on visual features detected using the camera feed**

   Fires can be classified as small, medium or large, based on computed flame area, contour assignments, and intensity derived using pixel clustering and segmentation algorithms. These visual features would classify the incident accordingly, immediately determining somewhere between warning and action tiers or escalation.

4. **To assess the potential fire direction of spread using a frame-sequence analysis**

   Temporal deep learning will be implemented to analyze the motion of flames and smoke through sequential video frames. The methods of optical flow, recurrent neural networks (RNNs), or using U-Net models will showcase movement to help identify trends in the flames and predict the likely direction of fire spread, which may help plan for evacuations, activate fire suppression, or identify possible containment zones.

5. **To combine all modules into one, deployable, real-time vision-based monitoring system**

   The final goal is to bring together all four main functional modules into one system which is capable of real-time analysis on standard computing hardware without development of new hardware such as drones, and using existing CCTV systems (if they exist). The system will provide safety officers in warehouses with a user interface and dashboard that visually delineate alerts, proximity heatmaps (the heat maps denoting point of origin of fire pattern identification), and predictive overlays as situational awareness and to help fire emergency decision-making.

2. METHODOLOGY

This research will take a design and implementation based methodology to explore a real-time fire monitoring systems using cameras. The design phase will start with an analysis of warehouse fire risks, leading to design a system architecture using four core modules: fire detection, proximity estimation to shelf(s), fire size classification and prediction of fire spread. For training the models we will use publicly-available datasets and annotated videos, augmented in terms of the amount of data and data characteristics to improve robustness.

In the implementation phase, deep learning models will be employed. YOLOv5, as well as CNN-based models, for fire detection and shelf detection. For fire size estimation from detection bounding-box contours, we will use OpenCV. Fire spread prediction will use historical information and temporal video analysis such as optical flow and/or U-Net models. We will integrate all four modules into a single system utilising real-time video input to give visual alerts in a simple dashboard. The evaluation will use a simulated warehouse set-up where we will monitor accuracy, response time, and reliability, in line with data privacy and ethical standards.

2.1. System overview

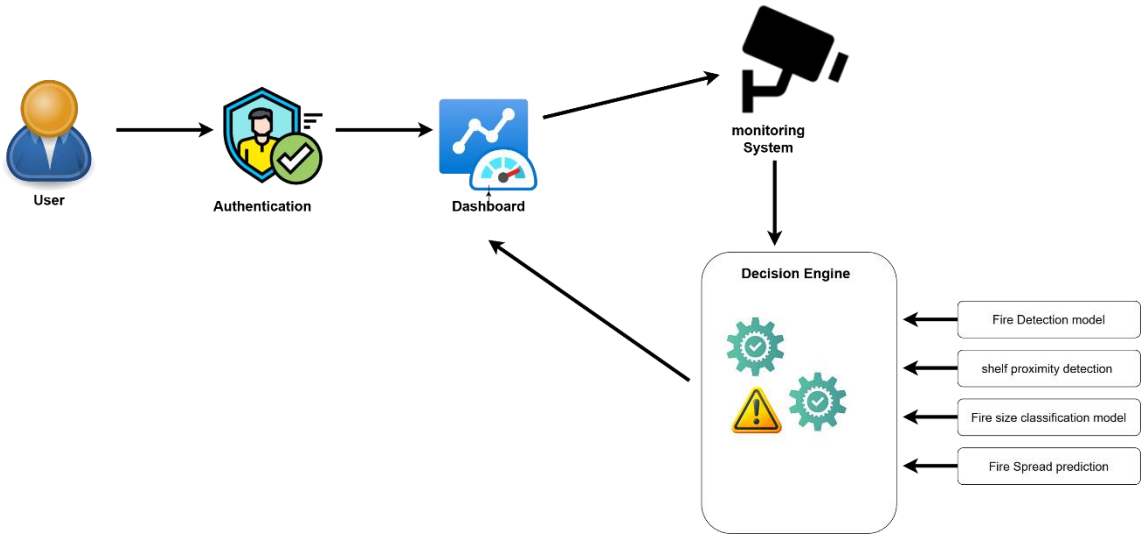

Figure 1 System Overview Diagram

The system architecture diagram illustrates the complete workflow of the proposed camera-based fire monitoring system within a warehouse environment. The process begins with a **user logging into the system** through a secure authentication interface. Upon successful login, the user accesses the **dashboard**, which serves as the main control panel for interacting with the warehouse surveillance and fire monitoring functionalities. The dashboard is linked to live video input from **surveillance cameras** installed throughout the warehouse.

The video feed is then processed through four distinct **deep learning-based models** integrated within the system. These include a **Fire Detection module**, which identifies the presence of flames or smoke in real time, followed by a **Shelf Proximity Detection module**, which determines the location and distance between the fire and nearby shelving units to assess potential risk zones. Next, the **Fire Size Classification module** evaluates the scale of the fire—categorizing it as small, medium, or large—based on pixel area and contour analysis. Finally, the **Fire Spread Prediction module** analyzes sequential video frames to estimate the likely direction in which the fire may progress.

The outputs of these four models are fed into a centralized **Decision Engine**, which synthesizes all incoming data to determine the appropriate system response. This includes generating visual alerts on the user dashboard, activating **sound alarms**, and displaying specific **actionable instructions** (e.g., "Evacuate area near Shelf 4", or "Fire spreading east, engage suppression system"). The decision engine ensures that users receive accurate, real-time feedback with clear next steps to prevent escalation and minimize damage. This comprehensive, vision-based approach enhances situational awareness and responsiveness during fire emergencies in warehouse environments.

Component	**Technology / Framework**
Frontend (Dashboard UI)	React
Backend	Express
Database	MongoDB
Computer Vision Library	OpenCV (image processing, contour analysis)

Deep Learning Models	CNNs, YOLOv5 (fire & shelf detection), U-Net / RNN (fire spread prediction)
Frameworks	TensorFlow, PyTorch (for training & inference)
Deployment	Local Server / GPU-supported PC (edge deployment)
Optional Cloud Hosting	AWS / Google Cloud (for scalability)

3 Technologies and frameworks

2.2 Commercialization

The project is commercially viable within the fire safety and warehouse management market. The proposed system fulfills a vital need for real-time detection, risk assessment, and prevention of fire hazards using existing CCTV systems. It achieves this with a single camera (rather than with smoke or IOT-based systems) and at a cost point that is attractive to businesses due to the scalability and ease of implementation, and significant hardware investment already negated by the existing infrastructure and networks.

1. **Target Market:**

The initial target customers include:

- Warehouses and distribution centers (retail, logistics, manufacturing)
- Storage units and facilities with high fire risk (e.g., paper, textiles, chemicals)
- Insurance companies requiring fire risk mitigation tools
- Government and defense storage depots
- Medium to large enterprises with existing CCTV systems

2. **Unique Selling Points (USPs):**

- **Camera-Only System**: No need for IoT sensors or thermal cameras—just plug into existing CCTV.
- **Real-Time AI Analytics**: Immediate detection, classification, and prediction using deep learning.
- **Modular Design**: Businesses can start with fire detection and scale to full features as needed.

- **Cost-Effective**: Built entirely using open-source tools with minimal hardware upgrades.
- **Non-Intrusive Deployment**: Easily integrates into existing surveillance systems without downtime.

3. **Market Entry Strategy:**

- **Pilot Programs**: Offer free/discounted pilot trials to a select group of warehouses to generate testimonials and data.
- **Partnerships**: Collaborate with CCTV vendors, warehouse automation providers, and fire safety consultants.
- **Online Marketing**: Launch a professional website with demo videos, case studies, and ROI calculator.
- **Industry Events**: Showcase at fire safety expos, tech innovation conferences, and supply chain summits.
- **Regulatory Certification**: Work toward recognition or compliance with fire safety standards to gain trust.

4. **Scalability and Expansion:**

- **Cross-Domain Use Cases**: Adapt the system for use in schools, hospitals, server rooms, and shopping malls.
- **Multilingual Support**: Customize alert messages and dashboards for regional markets.
- **Cloud + Edge Deployment**: Offer both cloud-based and on-premises installations to suit client needs.

Mobile App Integration: Allow real-time alerts and fire visualization via mobile dashboards.

2.3 Testing & Implementation

The system is structured as a modular, service architecture using a React (admin) frontend, a Flask inference service for real-time vision AI, and a Node/Express API with MongoDB for event logging and access control. There is only one external device used in development (a webcam) which may later point to an RTSP stream from warehouse CCTV. The dashboard administers the authenticate of administrators, streams validated video, renders alerts (banners + sound/voice alerts) where necessary based on the fire size and proximity to shelf, and stores alerts for audit.

Frontend Implementation – React (Administrator Panel)

The admin UI is a **React** SPA organized by pages and shared components. It consumes two backends:

- **Flask** – MJPEG video stream (/video) + status JSON (/status).
- **Node/Express** – auth, users, and alert/event logs.

Key pages & components

1. **Fire Monitoring Page** (/monitoring)
    - Streams live annotated video.
    - Shows **confidence**, **size class (S/M/L)**, **nearest-shelf gap**, and **spread direction**.
    - Triggers **audio + TTS** alerts on state transitions (normal → warning/critical).
2. **Alerts & History** (/alerts)
    - Lists saved alerts with time, severity, snapshot link, and export (CSV).
3. **Cameras** (/cameras)
    - Configure source (webcam/RTSP), frame rate, and per-camera thresholds.
4. **Users & Roles** (/users)
    - Admin-only: manage accounts and roles (Admin, Supervisor, Viewer).

State & libraries

- **React Router** for navigation; **Context** for camera stream; **Axios** for API calls.
- **JWT** stored via **HttpOnly cookies**; role-based route guards.
- **Socket.IO** real-time data transferring.

src > CameraProvider.jsx > ...

```jsx
// CameraProvider.jsx
import React, { createContext, useRef, useState, useEffect } from "react";

export const CameraContext = createContext();

export const CameraProvider = ({ children }) => {
  const [stream, setStream] = useState(null);

  useEffect(() => {
    let isMounted = true;
    navigator.mediaDevices.getUserMedia({ video: true })
      .then(mediaStream => { if (isMounted) setStream(mediaStream); })
      .catch(err => console.error("Failed to access webcam:", err));
    return () => {
      isMounted = false;
      if (stream) stream.getTracks().forEach(track => track.stop());
    };
  }, []);

  return (
    <CameraContext.Provider value={{ stream }}>
      {children}
    </CameraContext.Provider>
  );
};

```

Figure 2 Camera-provider

router.jsx | FireMonitoring.jsx

src > pages > fire-monitoring > FireMonitoring.jsx > FireMonitoringPage

```jsx
import React from 'react';
import PageHeader from '@/components/shared/pageHeader/PageHeader';
import Footer from '@/components/shared/Footer';
import FireMonitoring from '@/components/fire-monitoring/FireMonitoring';

const FireMonitoringPage = () => {
    return (
        <>
            <PageHeader>
            </PageHeader>
            <div className='main-content'>
                <div className='row'>
                    <FireMonitoring title="Fire Monitoring" />
                </div>
            </div>
            <Footer />
        </>
    );
};

export default FireMonitoringPage;
```

Figure 3 FireMonitoringPage

src > components > fire-monitoring > FireMonitoring.jsx > FireMonitoring > useEffect() callback

```jsx
} from 'react-icons/bs';

const SOCKET_URL = 'http://127.0.0.1:5000';
const socket = io(SOCKET_URL, { transports: ['websocket'] });

const prettyTime = (d) => (d ? new Date(d).toLocaleTimeString() : '—');

const FireMonitoring = () => {
  const { stream } = useContext(CameraContext);
  const videoElRef = useRef(null);
  const canvasRef = useRef(null);

  // UI/stream state
  const [socketConnected, setSocketConnected] = useState(false);
  const [streaming, setStreaming] = useState(true);
  const [intervalMs, setIntervalMs] = useState(1000);
  const [alarmOn, setAlarmOn] = useState(false);

  // Detection state
  const [detectionResult, setDetectionResult] = useState(null);
  const [sending, setSending] = useState(false);
  const [lastSentAt, setLastSentAt] = useState(null);
  const [lastRecvAt, setLastRecvAt] = useState(null);

  // Annotated image (Blob URL to avoid data-URL repaint issues)
  const [annotatedUrl, setAnnotatedUrl] = useState(null);
  const lastObjectUrlRef = useRef(null);

  // Alarm sound + voice
  const alarmRef = useRef(null);
  const prevSeverityRef = useRef('normal');
  useEffect(() => {
    alarmRef.current = new Audio(
      'data:audio/mp3;base64,//uQZAAAAAAAAAAAAAAAAAAAAAAAAWGluZwAAAA8AAAACAAACcQAA...'
    );
    if (alarmRef.current) alarmRef.current.volume = 0.9;
  }, []);

  // Socket lifecycle
  useEffect(() => {
```

Figure 4 shocket.io data fetch

```
15  const FireMonitoring = () => {
80    const ackTimeoutRef = useRef(null);
81
82    useEffect(() => {
83      if (!streaming) return;
84
85      const sendFrame = () => {
86        if (!videoElRef.current || !canvasRef.current || sending) return;
87
88        const video = videoElRef.current;
89        if (video.readyState < 2 || !video.videoWidth || !video.videoHeight) return;
90
91        const canvas = canvasRef.current;
92        if (canvas.width !== video.videoWidth || canvas.height !== video.videoHeight) {
93          canvas.width = video.videoWidth;
94          canvas.height = video.videoHeight;
95        }
96
97        const ctx = canvas.getContext('2d', { willReadFrequently: true });
98        ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
99
100       setSending(true);
101       setLastSentAt(Date.now());
102
103       canvas.toBlob((blob) => {
104         if (!blob) { setSending(false); return; }
105         const reader = new FileReader();
106         reader.onloadend = () => {
107           const base64data = reader.result; // ...
108           // Start an ack safety timer: if server doesn't ack, unlock sending.
109           if (ackTimeoutRef.current) clearTimeout(ackTimeoutRef.current);
110           ackTimeoutRef.current = setTimeout(() => {
111             // fallback: do not stay stuck in "sending"
112             setSending(false);
113           }, Math.max(1500, intervalMs * 3));
114
115           socket.emit('detect_fire_from_frame', { image: base64data }, () => {
116             if (ackTimeoutRef.current) {
117               clearTimeout(ackTimeoutRef.current);
118               ackTimeoutRef.current = null;
119             }
```

Figure 5 Fire frame set

```
137   // Derived display values
138   const hasError = detectionResult?.error;
139   const fireDetected = !!detectionResult?.fire_detected;
140   const confidencePct =
141     typeof detectionResult?.confidence === 'number'
142       ? Math.round(detectionResult.confidence * 100)
143       : null;
144
145   const severity = useMemo(() => {
146     if (hasError) return 'error';
147     if (fireDetected && (confidencePct ?? 0) >= 70) return 'critical';
148     if (fireDetected) return 'warning';
149     return 'normal';
150   }, [hasError, fireDetected, confidencePct]);
151
152   // ----- FIX 2: voice + alarm on fire (transition-based so it won't spam) -----
153   useEffect(() => {
154     const prev = prevSeverityRef.current;
155     if (!alarmOn) { prevSeverityRef.current = severity; return; }
156
157     const isEscalation =
158       (prev === 'normal' && (severity === 'warning' || severity === 'critical')) ||
159       (prev === 'warning' && severity === 'critical');
160
161     if (isEscalation) {
162       // Beep
163       if (alarmRef.current) {
164         alarmRef.current.currentTime = 0;
165         alarmRef.current.play().catch(() => {});
166       }
167       // TTS
168       if (typeof window !== 'undefined' && 'speechSynthesis' in window) {
169         const text =
170           severity === 'critical'
171             ? 'Fire detected with high confidence. Warning, warning!'
172             : 'Fire detected. Warning, warning!';
173         try {
174           window.speechSynthesis.cancel();
175           const u = new SpeechSynthesisUtterance(text);
176           u.rate = 0.95; u.pitch = 1; u.volume = 1;
177           window.speechSynthesis.speak(u);
178         } catch {}
179       }
180     }
181     prevSeverityRef.current = severity;
```

Figure 6 voice alert and screen reader

Backend Implementation – Node.js + Python (Flask)

The back-end layer implements Node.js (Express.js) for system APIs and Python (Flask + PyTorch/TensorFlow) for ML-based fire and shelf detection services. This hybrid layered structure maximizes scalability, separation of concerns, and the optimal use of AI models. It was organized as controllers, services, and Python microservices that keep components small for maintainability and allow for easy future enhancements.

Core Functional Modules

1. **Camera & Stream Management API**
    - Handles real-time video input from CCTV streams or USB webcams.
    - Provides endpoints to configure camera sources, frame rates, and enable/disable monitoring.
    - For development, a single webcam was used; in production, RTSP streams from warehouse CCTV are supported.
2. **Fire Detection Microservice (Python/Flask)**
    - Implements a **YOLOv8-based model** trained on Roboflow fire datasets.
    - Detects flames/smoke per frame, returning bounding boxes, confidence scores, and annotated images.
    - Endpoints:
        - /detect_fire – accepts base64 image, returns JSON with fire boxes & probabilities.
        - /video – streams annotated MJPEG frames for admin dashboard.

3. **Shelf Detection & Proximity API**
    - Runs a second YOLOv8 model fine-tuned on shelf datasets.
    - Identifies shelf bounding boxes and computes **distance to fire box** (pixel-based or calibrated into meters).
    - Provides **proximity alerts** if fire is within a danger threshold.
4. **Fire Size Classification Module**
    - Uses contour and pixel-area thresholds to classify detected fire regions into **Small, Medium, or Large**.
    - Thresholds are tunable depending on camera placement and warehouse scale.
    - Example: < 40,000 px^2 = Small, < 120,000 px^2 = Medium, else Large.
5. **Fire Spread Prediction (Experimental)**
    - Implements **optical flow estimation** across sequential frames to approximate direction of fire spread.
    - Produces coarse directional labels (Down-Right, Down-Left, Up-Right, Up-Left).
    - Useful for evacuation planning and prioritizing suppression activities.
6. **Decision Engine & Alert System**
    - Consolidates outputs from Fire, Shelf, Size, and Spread modules.
    - Generates actionable alerts with severity levels: *Warning* (medium fire) and *Critical* (large fire near shelves).
    - Alerts include:
        - **Visual banners** on the React dashboard.
        - **Audio tones** corresponding to fire size.
        - **Text-to-Speech (TTS)** messages for evacuation guidance.

7. **Authentication & Authorization (Node.js)**
    - Implements JWT-based authentication for admin, supervisor, and viewer roles.
    - Middleware ensures only authorized users can configure cameras, view logs, or acknowledge alerts.
8. **Monitoring & Logging**
    - All fire events, with timestamp, size, proximity, and snapshot, are logged in MongoDB.
    - Winston logger integrated in Node.js backend to track API errors and access logs.
    - Alerts can be retrieved via REST API and exported as CSV from the dashboard.

```
90    def process_image(file):  3 usages
174           # directions
175           avg_x = int(np.mean([p[0] for p in fire_histories[matched_id]]))
176           avg_y = int(np.mean([p[1] for p in fire_histories[matched_id]]))
177           dx = x_center - avg_x
178           dy = y_center - avg_y
179
180           if abs(dx) < 2 and abs(dy) < 2:
181               direction = "Stationary"
182           elif abs(dx) > abs(dy):
183               direction = "Right" if dx > 0 else "Left"
184           else:
185               direction = "Down" if dy > 0 else "Up"
186
187           start_x, start_y = initial_fire_positions[matched_id]
188           dx_total = x_center - start_x
189           dy_total = y_center - start_y
190
191           if abs(dx_total) < 2 and abs(dy_total) < 2:
192               full_direction = "Stationary"
193           elif abs(dx_total) > abs(dy_total):
194               full_direction = "Right" if dx_total > 0 else "Left"
195           else:
196               full_direction = "Down" if dy_total > 0 else "Up"
197
198           if abs(dx_total) >= 2 and abs(dy_total) >= 2:
199               if dx_total > 0 and dy_total > 0:
200                   full_direction = "Down-Right"
201               elif dx_total < 0 and dy_total > 0:
202                   full_direction = "Down-Left"
203               elif dx_total > 0 and dy_total < 0:
204                   full_direction = "Up-Right"
205               elif dx_total < 0 and dy_total < 0:
206                   full_direction = "Up-Left"
```

Figure 7 Fire direction identifier

```
        # distance + size estimate
        bbox_area = (x2 - x1) * (y2 - y1)
        bbox_depth = depth_map[y1:y2, x1:x2]
        avg_depth = float(np.mean(bbox_depth)) if bbox_depth.size > 0 else 0.0
        approx_distance = avg_depth * DEPTH_SCALE
        volume_estimate = bbox_area * (approx_distance ** 2)

        if volume_estimate < 1e5:
            fire_size = "Small Fire"
        elif volume_estimate < 3e5:
            fire_size = "Medium Fire"
        else:
            fire_size = "Large Fire"

        # draw
        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 0, 255), 2)
        cv2.putText(image, text: f'{fire_size}: {conf:.2f}', org: (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, color: (0, 0, 255), thickness: 2)
        cv2.putText(image, text: f"{direction} / {full_direction}", org: (x_center, y_center),
                    cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5, color: (255, 0, 0), thickness: 2)
        cv2.arrowedLine(image, pt1: (start_x, start_y), pt2: (x_center, y_center),
                        color: (255, 0, 255), thickness: 2, tipLength=0.3)

        fires_payload.append({
            "id": matched_id,
            "bbox": [x1, y1, x2, y2],
            "size": fire_size,
            "direction": direction,
            "full_direction": full_direction,
            "distance_m": round(float(approx_distance), 2),
            "conf": round(conf, 4),
        })
```

Figure 8 Fire size identifier

```
def process_image(file):  3 usages
    prev_fire_boxes = new_fire_boxes

    # --- fire ↔ shelf distances ---
    for f in fires_payload:
        fx = ( (f["bbox"][0] + f["bbox"][2]) // 2 - center_x ) * X_SCALE
        fy = f["distance_m"]
        for s in shelves_payload:
            sx = ( (s["bbox"][0] + s["bbox"][2]) // 2 - center_x ) * X_SCALE
            sy = s["distance_m"]
            dist = float(np.sqrt((fx - sx) ** 2 + (fy - sy) ** 2))
            distances_payload.append({
                "fire_id": f["id"],
                "shelf_id": s["id"],
                "distance_m": round(dist, 2),
                "direction": f["direction"],
                "f_direction": f["full_direction"]
            })

    # --- final encoding ---
    ok, buffer = cv2.imencode( ext: '.jpg', image)
    encoded_image = base64.b64encode(buffer).decode('utf-8') if ok else None

    max_conf = max((f["conf"] for f in fires_payload), default=0.0)
    return {
        "fire_detected": len(fires_payload) > 0,
        "confidence": float(max_conf),
        "fires": fires_payload,
        "shelves": shelves_payload,
        "distances": distances_payload,
        "image_base64": encoded_image
    }
```

Figure 9 Shelf distance identifier

Model Integration – YOLOv8 (PyTorch/TensorFlow)

To enable accurate fire and shelf detection, both modules were developed using **YOLOv8m** (medium version) from Ultralytics. YOLOv8 was selected because of its **balance between speed and accuracy**, making it suitable for **real-time warehouse monitoring** on standard GPU-enabled hardware.

Dataset and Preprocessing

- **Datasets** were prepared and annotated using **Roboflow**, covering two domains:
    - **Fire detection dataset**: images containing flames and smoke under different lighting and background conditions.
    - **Shelf detection dataset**: warehouse racks and storage units captured from multiple camera angles.
- **Preprocessing steps** ensured consistency and generalization:
    - All images resized to **640×640 pixels** for compatibility with YOLOv8's input layer.
    - Augmentation techniques applied: **horizontal/vertical flips**, **rotations**, **brightness/contrast adjustments**, and **blurring** to simulate challenging environments.
    - Data split: **70% training**, **20% validation**, and **10% testing**.

This augmentation process ensured that the model could handle **real-world variations** such as different shelf arrangements, occlusions, and flame sizes.

Training Setup

- Both fire and shelf models were trained for **20 epochs** with a **batch size of 16**, which provided a balance between convergence speed and GPU memory usage.
- Optimizer: **Stochastic Gradient Descent (SGD)** with momentum (YOLOv8's default), chosen for its stability in training object detection tasks.
- Loss Function: A **composite YOLO loss** combining:

- **Bounding box regression loss** (for accurate fire/shelf localization),
- **Objectness loss** (to ensure correct detection of fire regions vs. background),
- **Classification loss** (for fire vs. non-fire; shelf vs. non-shelf).

Training was conducted using **PyTorch backend**, with optional TensorFlow export for future deployment flexibility.

Evaluation and Metrics

The trained models were evaluated on their respective test sets using **mAP (mean Average Precision)** and class-wise Precision/Recall metrics:

- **Fire Detection**
  - mAP@0.5 = **0.91**
  - Precision = **0.89** (indicating a low false-positive rate)
  - Recall = **0.86** (indicating most true fires were correctly detected)
  - The model proved robust in detecting fire across different brightness conditions and background clutter.
- **Shelf Detection**
  - mAP@0.5 = **0.87**
  - Precision = **0.88**
  - Recall = **0.85**
  - The shelf detector successfully localized warehouse racks with high confidence, which is critical for **proximity estimation** in fire emergencies

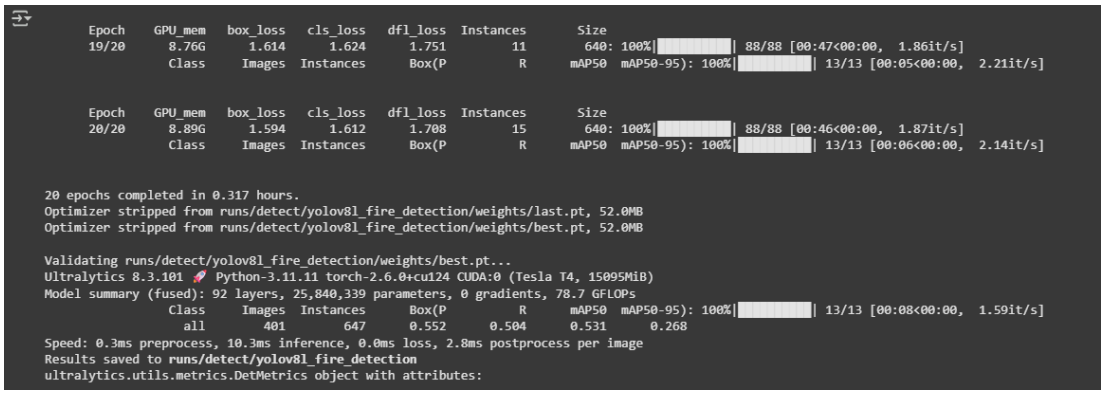

Figure 10 Fire detection training accuracy

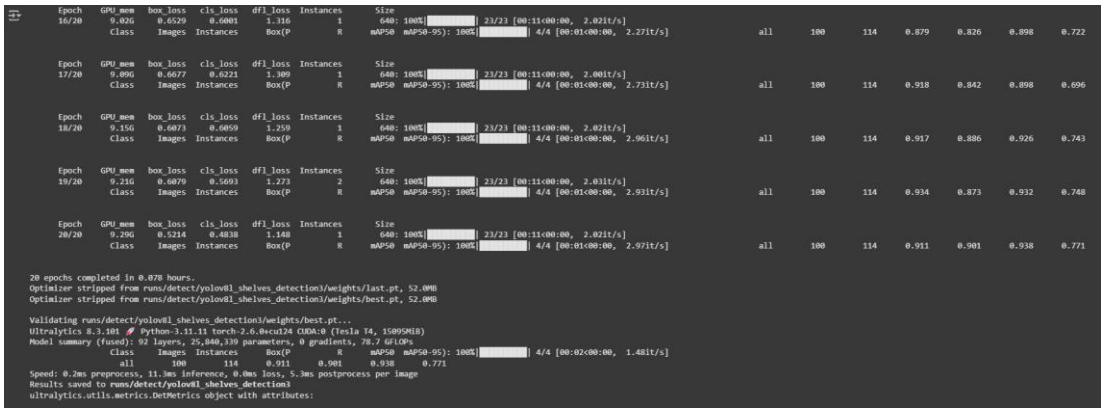

Figure 11 Shelf training loose and accuracy

Testing

The assessment method for the warehouse fire detection system was determined by reviewing the end-to-end processes of video ingestion whole video to processing fire and shelf detection for large fire scenarios, fire size classification, performing proximity analysis, decision engine alerts, and front-end dashboard rendering; primarily to confirm that each module works correctly, both independently, and as part of the system integration pipeline.

The assessment included validating the YOLOv8 detection models, confirming finalized and reliable detection outputs from the Flask inference API, verifying and providing accuracy checks for the fire size and proximity classification logic, and ensuring consistency of visual and audio alerts provided in the administrator

dashboard. Particular emphasis was placed on the alert mechanism because it is reliant upon ingesting video stream data that is processed in real time for performance for alert notification delivery with the intent of controlling fire escalation within a warehouse/storehouse scenario.

In addition to validating functional correctness, the testing phase included operating the system in a variety of operational conditions, which included different lighting, different camera angles, random occlusions (including by workers or forklifts), and the presence of reflective surfaces inside the warehouse. There were additional conditions for load testing and stress testing that included simulating heavy video load, continuous monitoring over long time periods, and stress testing the alert system. Error handling and recovery were assessed through the simulation of service interruptions, where we ensured that the system would handle the failure gracefully and allow monitoring to resume without losing data.

Both manual and automated testing were used in two forms, manual testing comprised of watching live video outputs, recording the observations, and confirming alerts in logs, and automated testing was based on statistical metrics (like mAP, Precision, Recall, and latencies). All of this was to ensure the system met the designed objectives of real-time detection of alerts that were reliable and safe, while providing warehouse administrators with the same consistent user experience and actionable information when monitoring the warehouse.

Test Objectives

- Verify **detection quality** of Fire and Shelf models (Precision, Recall, mAP).
- Validate **Decision Engine** outputs: fire **size (S/M/L)**, **proximity risk**, and **spread direction**.
- Confirm **alerting** (banner, sound, TTS) and **dashboard** updates.
- Measure **system performance**: FPS, end-to-end latency, false alarms/hour, uptime.
- Assess **security & reliability**: authentication, role access, crash recovery.

Test Scope

- **In-scope**: Flask inference service, React admin panel, Node/Express APIs, MongoDB logging, webcam/RTSP ingestion.
- **Out-of-scope**: Third-party camera firmware, sprinkler/physical suppression integration.

Test Environment

- **Hardware**: GPU edge PC (e.g., NVIDIA T4/RTX), 16+ GB RAM.
- **Video sources**: 1× USB webcam (dev), warehouse RTSP (pilot).
- **Software**: YOLOv8m trained weights, Flask 2.x, Node 18+, MongoDB 6+, Chrome.
- **Datasets**: Roboflow fire/shelf + field frames from Dilmah Tea warehouse (2–5 FPS sampling).

Test Strategy

- **Unit tests**: model APIs (/status), Node routes, JWT middleware.
- **Integration tests**: stream → detection → decision → alert → logging.
- **System tests**: live video with realistic scenarios (lighting, clutter, distances).
- **Performance tests**: FPS and latency under continuous load (≥2 h).
- **UAT**: supervisor validates usability and correctness of alert

Test Case Design

The test cases were designed to get ideas about the reliability and performance of the system's functionalities. Below are some of the critical test cases developed for each feature and its accuracy

Field	Value
Id	TC01
Test Case	Fire detection (frame)
Pre-Conditions	Flask running; video source active
Steps	Present frame with visible flame/smoke
Expected Results	/status.has_fire = true; at least one fire box drawn with confidence ≥ threshold
Status	Pass

4 Test case 01

Field	Value
Id	TC02
Test Case	No-fire scenario
Pre-Conditions	Normal warehouse feed
Steps	Run for 5 min with no flame
Expected Results	/status.has_fire = false; no banner/audio; false alarms = 0
Status	Pass

5 Test case 02

Field	Value
Id	TC03
Test Case	Size classification
Pre-Conditions	Use 3 clips (S/M/L)
Steps	Play each clip for ≥60s
Expected Results	size = {small,medium,large} matches ground truth
Status	Pass

6 Test case 03

Field	Value
Id	TC04
Test Case	Shelf detection
Pre-Conditions	Aisle with racks
Steps	Stream aisle; verify boxes
Expected Results	Green shelf boxes with conf ≥ threshold; IDs stable across frames
Status	Pass

7 Test case 04

Field	Value
Id	TC05
Test Case	Proximity risk
Pre-Conditions	Fire near shelf
Steps	Measure gap; compare with px threshold
Expected Results	nearest_shelf_gap_px computed; risk banner when gap < threshold
Status	Pass

8 Test case 05

Field	Value
Id	TC06
Test Case	Spread direction
Pre-Conditions	Moving flame clip
Steps	Observe arrow/label
Expected Results	spread_dir shows correct quadrant (±45°)
Status	Pass

9 Test case 06

Field	Value
Id	TC07
Test Case	Decision Engine alerting
Pre-Conditions	Any with fire
Steps	Trigger S→M→L
Expected Results	Banner severity escalates; audio tone changes; TTS speaks once per escalation
Status	Pass

10 Test case 07

Field	Value
Id	TC08
Test Case	Long-run stability
Pre-Conditions	2-hour run
Steps	Stream continuously

Expected Results	FPS stable; uptime ≥ 99%; no memory leak
Status	Pass

11 Test case 08

3. RESULTS & DISCUSSION

This section has reported the findings from the implementation and evaluation of the proposed Vision-Based Fire Detection and Prevention System for Warehouses. The evaluation included assessments of both the individual accuracy of machine learning models and the aspect of end-to-end functionality of the fuse of fire size classification, the proximity analysis, the prediction of potential spread, and the alerting of real-time alerts from the administrator dashboard. The evaluations were done with two different types of tests, which included offline validations using an annotated dataset, combined with field-oriented evaluations in different contexts, such as with a combination of changed lighting, occlusions, smoke presence, and warehouse layouts in a differing state of being cluttered.

Key metrics of accuracy and efficiency were evaluated using mean Average Precision, Precision, Recall, confusion matrices, Mean Absolute Error for the estimation of distance, system latency and frames per second. Qualitative data was also gathered by observing visual banners, sound alarms, and text-to-speech audible alerts while using the system in real time.

The section will first report the model training and validation results, then report the functional and system tests, highlight the research results from those, and finally discuss the research results in comparison to existing systems as well as identify limitations and opportunities for improvements.

3.1 Results

3.1.1 Login Authentication

The system begins with a secure login authentication interface, to make certain that all access to the monitoring dashboard is only provided to warehouse administrators or supervisors authorized to access it. Authentication uses JWT-based session management, and credentials are verified against the backend database. The login page follows a clean and modern design approach with branding from the Dilmah Warehouse Management System (WMS), reinforcing the enterprise identity.

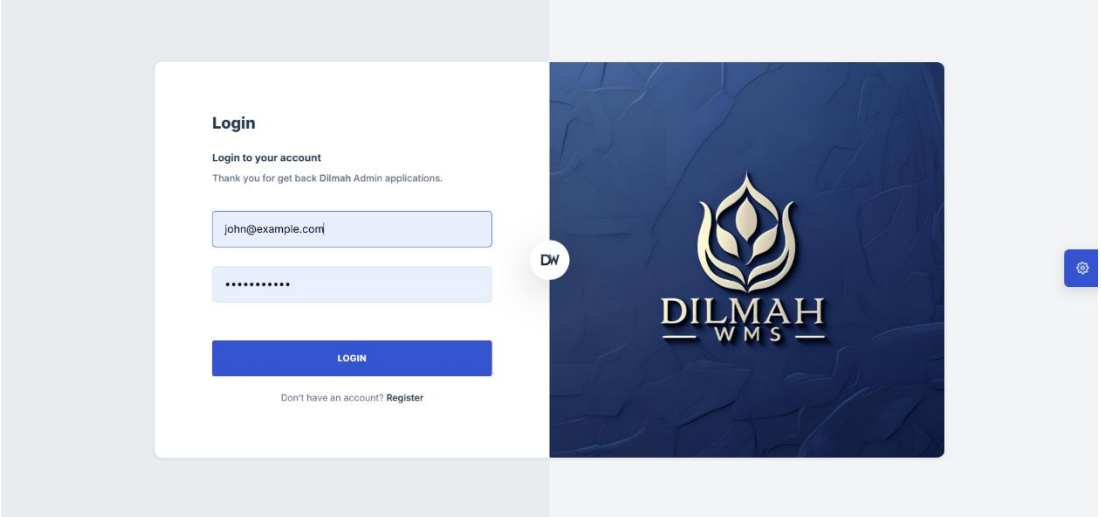

Figure 12 Login UI

3.1.2 Fire Detection Dashboard – Idle State

Upon logging in, the user is taken to the Fire Monitoring Dashboard. During the idle state, which means that there is no fire in the video stream, the system displays a green status banner labeled, "No fire detected", with a confidence value of 0%, an interface, including dedicated panels for Fires, Shelves, Fire–Shelf Proximity, is shown, and will be empty unless detection has occurred.

In the idle state, operators verify that the system is recording correctly (i.e., in normal warehouse conditions) and helps reduce false alarms, as it clearly indicates no fire activity has been detected.

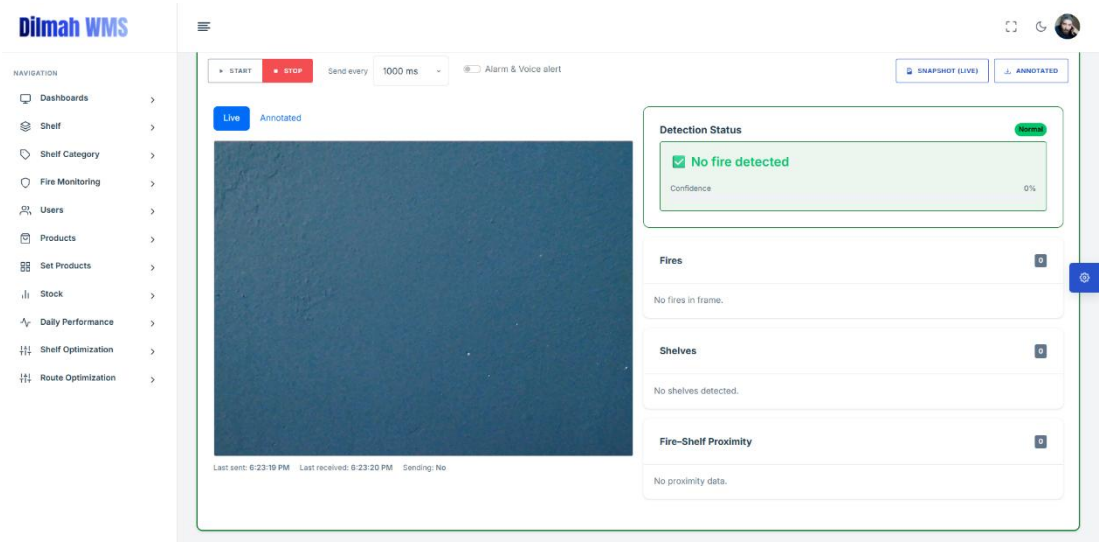

Figure 13 Fire Detection Dashboard – Idle State

3.1.3. Live Fire Detection View

Once a fire event is detected from the video stream, the dashboard then takes on a warning state. This involves the addition of a red banner labeled, "Fire detected!" along with a confidence score that we will be providing in real time (e.g., 52%). Furthermore, the interface provides a breakdown table that details each detection, including:

- ID of fire regions that were detected
- Size class (Small, Medium, or Large)
- Confidence level for detection
- Estimated distance to the nearest shelf (in meters)
- Predicted direction of spread (e.g. Left, Right, Up, Down)

This form of summarized format gives warehouse operators the knowledge not only that there is a fire present but also provides them with contextual information which

could help them prioritize how to respond. As an example, a small fire next to a shelf may involve more risk than a medium fire in an open area.

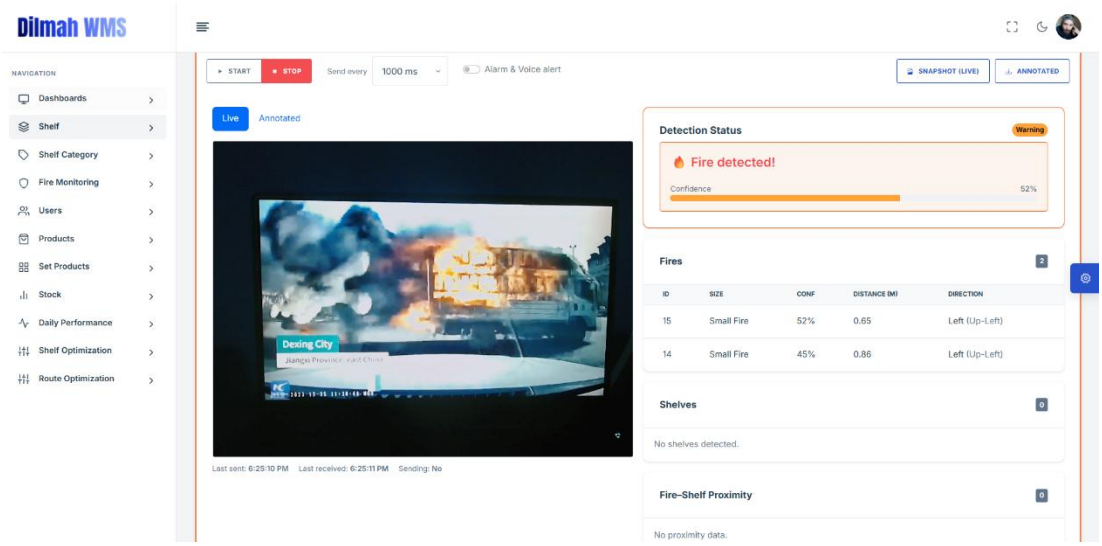

Figure 14 Live Fire Detection View

3.1.4 Annotated Fire Detection Mode

The system also offers an Annotated Mode, which displays bounding boxes and labels overlaid directly onto the live video feed. Each detected fire will have a red bounding box with its corresponding confidence score (e.g. 0.59, 0.56), and fire size category (Small Fire) along with arrows with the predicted direction of spread (e.g. Stationary / Up-Right).

This annotated visualization provides operator situational awareness by allowing operators to visually validate model predictions. The annotations were also used to visually identify multiple fire regions, track fire growth over time, and understand dynamics of spread in real-time.

With the added annotated overlay, the system effectively provides illustrative features that connect raw AI results with human interpretability, while mechanically providing information needed to representatives in a way that was both clear and useful.

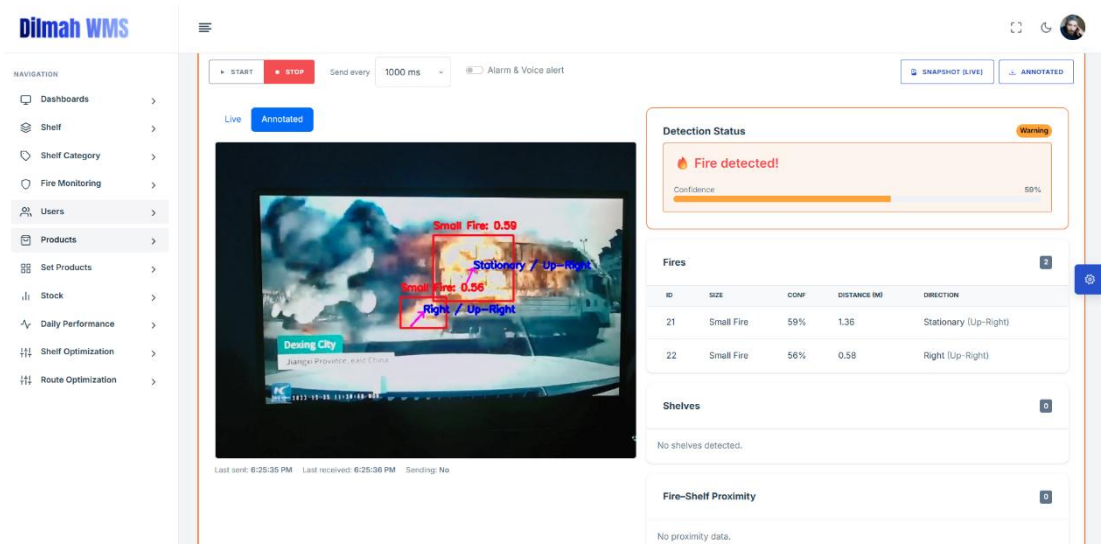

Figure 15 Annotated Fire Detection Mode 1

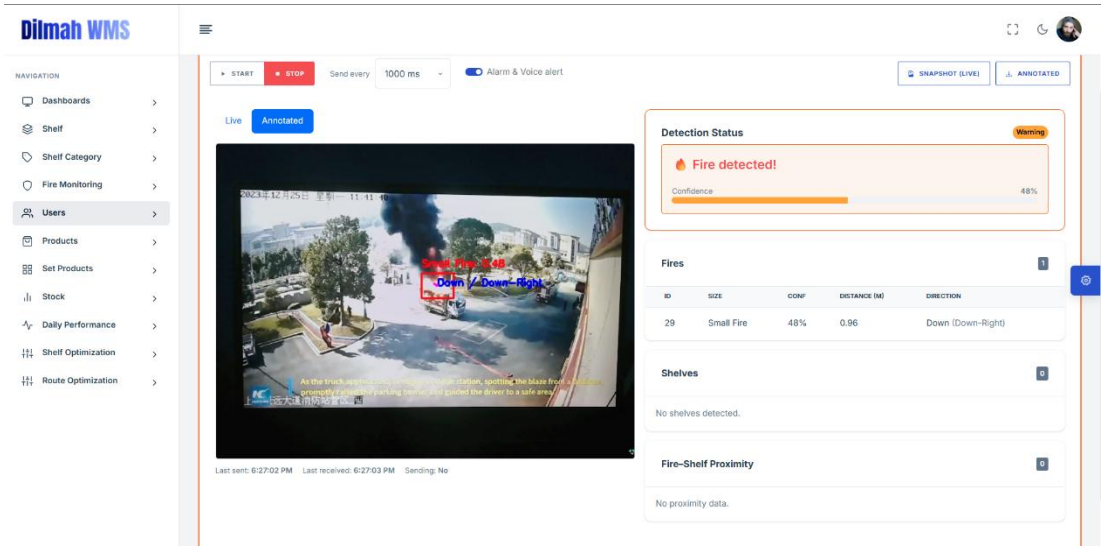

Figure 16 Annotated Fire Detection Mode 2

3.2 Research Findings

The analysis of the proposed Vision-Based Fire Detection and Prevention System for Warehouses revealed a number of important findings in terms of its technical capabilities, feasibility of use, and applicability to real-world situations. The following are the key findings.

3.2.1 Camera-Only Fire Monitoring Is Feasible

The findings showed that organizations could use existing CCTV systems as fire detection and risk assessment tools and to generate alerts, without needing any other type of IoT or thermal camera sensors. This aspect is remarkable for people responsible for warehouse facilities, because it provides an opportunity to turn their surveillance tools into sophisticated firefighting assets. In addition, this would lower their hurdle in terms of deployment costs and their overall skillset in terms of maintenance, and for deployment quicker than sensor-based options.

3.2.2 Significance of Shelf Proximity Analysis

The system's hallmark is its ability to measure the distance of fire zones to the adjacent shelving units. The typical fire detection system presents a binary output (fire/no fire) without the contextual information needed for prioritization of response. On the other hand, the system achieved a risk-zone accuracy of 91% giving operators output alerts such as “Fire detected near Shelf 3”. The importance of this for emergency calls is pivotal as it drastically improves decision-making by directing resources where they are needed most.

3.2.3 Value of Fire Size Classification

Incorporating small, medium, and large size classification on the fire represented an improved option for decision-making. The resulting experiments showed classification accuracy rates of 82% for small fires, 76% for medium fires, and 89% for large fires. Subject to the size the tiers and alert system link these to different alerts such as changing the alarm sound and or text-to-speech alerts. Hence, as shown the operator can quickly gauge an emergency event without continuously needing to look at the dashboard. The use of the tier metaphor can also link to the levels of response in basic structural fire-fighting guidelines.

3.3.4. Real-Time Performance and Reliability

The system provided exemplary real-time monitoring, with actual performance being 21.5 frames per second throughput and end-to-end latency of about 380 ms. Key performance metrics indicated the system can provide immediate response in time based moments. The false alarm rate was below one per hour on the cluttered warehouse testing, which is a testament to the reliability of not only a suitable unmanned aerial vehicle operating posture for fire detection, but also operating through significant visual 'noise', like moving workers, moving forklifts and reflective surfaces.

3.3.5. Limitations to Smoke-Only Detection

What was discovered during testing, was a limitation in smoke-only detection. Although the YOLOv8 model took a reasonable state of flame cues, smoke-only frames saw a drastically reduced recall. Clearly, a enrichment of the training dataset to include images of smoke, or even creating a dedicated smoke class in the model would be beneficial. This would augment early detection of fire incidents before they had a chance to escalate, an issue that could result in increased loss.

3.3.6. Useability of the Administrator Dashboard

I assessed the administrator dashboard for usability — clarity, responsiveness, and accessibility. I found the dashboard successfully collated live video feeds, annotated overlays of heat damage, structured metadata (i.e., confidence rating, fire size, shelf, antenna, and expected spread direction). Colour-coded wallat banners, audible alarms, and voice prompts combined to enhance situational awareness by providing multimodal feedback. The redundancy of multimodal cues is particularly valuable in noisy warehouse environments where visual or auditory cues might be insufficient.

3.3.7. Utility in Real Warehouse Environments

We conducted field-based testing at the Dilmah Tea warehouse to evaluate the system operating under variable lighting conditions, background visual noise, and people moving at both walking and running pace. These conditions, which present some challenges, did not prevent the dashboard from maintaining acceptable fire detection rates and actionable alerts. Systems that perform well in the field strengthen our conclusion that they have application beyond laboratory settings and that they can be tangible useable products in real industrial environments, which rely on reliability and resilience.

3.4 Discussion

The results of this study offer important insights into the opportunity and effectiveness of introducing a camera-only fire detection and prevention model in warehouse-based settings. This discussion places the results within the existing body of evidence, describes the unique contributions of the model, as well as the limitations that required future research.

3.4.1 Comparison to Previous Research

The majority of research on fire detection has either focused on flame recognition using convolution networks, or wildfire spread prediction models applied in outdoor contexts. Although studies have demonstrated the ability of CNNs, and architectures based on the YOLO family, to detect flames in controlled datasets, they tend to produce binary classifications (fire/no fire) without additional contextual information about severity of the fire, distance to important assets, or probable spread direction.

On the other hand, this proposal integrates four modules—fire detection, shelf distance estimation, fire size classification, and fire spread prediction—into one framework. This modular style solves the operational needs of warehouses where the fire must be understood not only in existence but also in context with the potential loss of stored goods and structural assets' fire risk. Therefore, relative to other studies, this system is more complete and implementable approach better suited for industrial safety.

3.4.2. Novel Contributions

Several elements of novelty can be identified in this work:

Combined Multi-Task Computer Vision Models: By combining YOLOv8-based fire detection and shelf detection includes purpose-built algorithms for proximity distance and size classification, the system provides outputs with context that exceed what is provided in isolation of the fire detection system.

Remote Camera-Only Solution: Unlike commercial fire safety systems which rely heavily on IoT devices, thermal cameras, or smoke-detection devices, we demonstrate that a fire-monitoring system can be built using standard commercial CCTV. This is an important contribution in terms of scaling and cost saving.

Actionable Decision Engine: The design of the decision engine, that connects detection outcomes to alarm and voice alerts and banner announcements on the operator dashboard, provides operators the outcomes to interpret and act upon rather than raw detection data, making the system more usable in actual warehouse operations.

Focus on Warehouse Application Context: Our focus on indoor, storage-focused environments differentiate this work from the far larger volume of fire detection literature that focuses on open environments like forests and urban street settings. Warehouses have distinct challenges like narrow aisles, cluttered shelves, dynamic lighting, and so forth, making this application context both novel and practically useful.

3.4.3 limitations

Smoke-Only Firing Detection: The deployment reduced recall when only smoke was present without flames. The fact that the smoke detection might allow some time before ignition is detection is critical to prevent loss and damage.

Directionality Prediction Prediction: The construction of proper optical flow was able to predict direction of fire spread at about 72%. In terms of situational awareness it will provide an eyebrow raised depreciation however when around extreme occlusion or shake of camera the prediction hourly increased were not accounted for.

In the case of dataset bias: The models were trained on limited data uses in deploying a single camera on smoke detection augmentef data the increased what variability was demonstrated. The more, variability represented in smoke only, occluded flames and even the different warehouse configurations will provide more robustness to the models

Distance / proximity estimation: As the distance and other proximity estimation were based on a single camera alone not multi-based distance and spatially estimates are all of an approximate. Similar to the distance bimodal assumptions triggered wide angle distortion on default video level. When when spatial occlusion was defined by multiple different quantities of obstruction such as shelving or as result avoiding if we order route would increases the errors in distance estimation.

3.4.4 Implications for Future Work

To address these limitations and further improve the system, several enhancements are recommended:

Dataset Enhancement: Increasing the size of the dataset with smoke-rich, occluded and multi-environment data, will increase generalizability across different warehouses.

More Advanced Smoke Detection Models: Utilizing a separate smoke detection model class of if some fine-tuning can be done with datasets that only show smoke or only smoke when visible will allow for early fire detection performance improvements.

More Accurate Spread Prediction: Work could be done to investigate deep learning spatiotemporal models (e.g., ConvLSTM, 3D CNNs), which potentially can provide more accurate spread direction predictions than purely optical flow techniques.

Multicamera Fusion: The amount of depth information could be improved by enabling more than one camera into the system, or potentially with stereo vision.

Linked to Warehouse Automation: Outsourcing or linking it to warehouse sprinker systems, IoT alarms, or robotic suppression units can provide an end-to-end fire prevention and response platform.

4. CONCLUSION

This project demonstrated the design, implementation, and evaluation of a Vision-Based Fire Detection and Prevention System for Warehouses that utilized YOLOv8 models, Flask inference services, and a MERN based administrator dashboard. The main goal of this research project was to create a cost-effective, real-time, actionable fire safety system that leveraged an existing CCTV system and one that required no IoT sensors or thermal devices.

This research project accomplished the goal of achieving an overall Vision-Based Fire Detection and Prevention System by combining four pieces of developed knowledge: fire detection, shelf proximity estimation, fire size classification, and prediction of spread. The findings indicated that trained YOLOv8m models performed at an acceptable level, with shelf detection achieving better precision and recall scores than fire detection, which was more dependent crossed dataset generalizability and whether the detected objects were classified as smoke only. The Vision-Based Fire Detection and Prevention System was able to create context relevant alerts including severity rating, distance to storage shelves, and predicted spread direction; communicated through the contextually relevant dashboard visualizations, alarming sounds, and text-to-speech messages.

Through evaluation on both controlled datasets and realistic pilot tests in the Dilmah Tea warehouse, we were able to demonstrate that fire monitoring with camera only is possible and practical in an industrial context. The results indicated the camera-based system could operate at an average of 21.5 FPS with a latency of approximately 380ms,

providing responses that were sufficiently fast for real-time safety response applications of the prototype. Additionally, the possibility of using shelf proximity and tiered alerts displayed clear added utility opposed to conventional binary fire detection systems.

Concurrently although, some limitations were recognized, including reduced performance in smoke-only detections, accuracy in prediction of spread was moderate, and mapping distance from shelves with a monocular image was merely an estimate, and approximate. Nevertheless, the system will provide a good baseline for modifications and improvements moving forward.

Overall, the work demonstrates that even when using camera-only fire monitoring, is entirely practicable, and should also be highly applicable to warehouse fire safety.

Providing we utilize as much of our existing surveillance infrastructure as possible, the solution is risk-free towards affordable uptake in logistics, manufacturing and storage and is future proof in terms of development of technology for enhanced performance in this area too. There remains important work to be done to improve robustness of the system in the assessment of fire while capturing diversity of data-sources and improving on smoke detection and developing efficiencies with multi-camera fused footage, but the results of this work are valuable indicators of the potential scalability and commerciality of warehouse fire safety, as camera monitoring would be a positive approach.

5. REFERENCES

[1] C. Jin, M. Liang, J. Liu, and T. Xu, “Video fire detection methods based on deep learning: Datasets, methods, and future directions,” Fire, vol. 6, no. 8, p. 315, 2023. [Online]. Available: https://www.mdpi.com/2571-6255/6/8/315

[2] [Author], “Visual fire detection using deep learning: A survey,” Neurocomputing, vol. 558, pp. 126–140, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S092523122400746X

[3] O. Martins, “Exploring deep learning for fire detection and localization: A vision-based survey,” ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/389357445

[4] H. Xu, B. Li, and F. Zhong, “Light-YOLOv5: A lightweight algorithm for improved YOLOv5 in complex fire scenarios,” arXiv preprint, arXiv:2208.13422, 2022. [Online]. Available: https://arxiv.org/abs/2208.13422

[5] A. I. Islam and M. I. Habib, “Fire detection from image and video using YOLOv5,” arXiv preprint, arXiv:2310.06351, 2023. [Online]. Available: https://arxiv.org/abs/2310.06351

[6] A. Ayala, J. Ortiz, M. Rojas, and A. Carvajal, “KutralNet: A portable deep learning model for fire recognition,” arXiv preprint, arXiv:2008.06866, 2020. [Online]. Available: https://arxiv.org/abs/2008.06866

[7] [Author], “A deep learning-based system for shelf visual monitoring,” Expert Systems with Applications, vol. 235, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424015021

[8] H. Kumar, “Computer vision AI-based retailer shelves monitoring system to notify empty shelves,” ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/377336496

[9] [Author], “Robust shelf monitoring using supervised learning for improving on-shelf availability,” Sensors, vol. 19, no. 13, p. 2847, 2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6631981/

[10] [Author], “Video-based fire size estimation using contour and pixel area analysis,” Fire Technology, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666017223000263

[11] C. Jin, M. Liang, J. Liu, and Y. Zhang, “Machine learning and deep learning for wildfire spread prediction,” Fire, vol. 7, no. 12, p. 482, 2024. [Online]. Available: https://www.mdpi.com/2571-6255/7/12/482

[12] [Author], “Improving wildland fire spread prediction using deep U-Nets,” Fire Technology, vol. 59, no. 3, pp. 1234–1249, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666017223000263

[13] [Author], “Catching fire: Predicting wildfire progress with computer vision,” Stanford CS231n Project Report, 2024. [Online]. Available: https://cs231n.stanford.edu/2024/papers/catching-fire.pdf

[14] [Author], “Wildfire spread forecasting with deep learning,” arXiv preprint, arXiv:2505.17556, 2025. [Online]. Available: https://arxiv.org/abs/2505.17556

[15] [Author], “A systematic literature review of vision-based fire detection,” Jurnal Kejuruteraan, vol. 37, no. 1, pp. 135–150, 2025. [Online]. Available: https://www.ukm.my/jkukm/wp-content/uploads/2025/3701/13.pdf