

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البلدية
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique



Final year project Dissertation

Master's degree in telecommunications

option: Telecommunication Systems

Presented by

KHENDOUKI Fedoua

Smart Fire Detection and Alert- System Drone: AI-IoT Integration

Supervised by: Dr. HEBIB Sami

Academic Year: 2023-2024

Dedications

The journey has been long and challenging, but by the grace of Allah, I have reached the end. Praise be to Allah, who eased the beginnings and guided me to this conclusion.

First and foremost, I dedicate this achievement to myself, to my determined and resilient self that withstood setbacks, persevered, and completed this journey despite the difficulties.

To my beloved mother, whose boundless love and sacrifices have been my guiding light.

To my father, whose wisdom and encouragement have continually shaped my path.

To my sisters, Sara, Rahma and my twin sister, Hadjer, whose unwavering support has been my strength, making every challenge easier to bear.

To all my family members, for their constant love, understanding, and support; and to my dear friends, whose presence and encouragement have made this journey truly memorable.

Acknowledgments

First and foremost, I express my deepest gratitude to Almighty God for granting me the health, determination, and patience to undertake and complete my journey and studies.

I extend my heartfelt appreciation to my supervisor, Mr. Sami HEBIB, for his invaluable guidance, unwavering support, and insightful feedback, which have been instrumental in shaping this work.

I would like to thank Mr. Yacine KABIR and Mrs. F. Zohra REGUIEG for their help and support, which have been crucial to the completion of this research. I would also like to thank Mr. Nanor Felix Samuel Tetteh for his help.

I am indebted to the esteemed members of the jury for their time, expertise, and constructive criticism, which have significantly contributed to the enhancement of this research.

I also extend my deepest thanks to my beloved parents, siblings, and friends for their unwavering encouragement, endless support, and motivation throughout this endeavor. Their belief in me has been my driving force.

I am grateful to all those who, directly or indirectly, have contributed to the completion of this thesis. Your support has been invaluable.

Lastly, I extend a sincere thank you to all the teachers of the Electronics Department at Saad Dahleb University in Blida for imparting their knowledge throughout my university studies.

ملخص: يمثل تزايد وتيرة وشدة حرائق الغابات تحديات كبيرة تتعلق بالبيئة والسلامة. يهدف هذا المشروع إلى تطوير نظام متكامل للكشف عن الحرائق والتنبيه بها يتم نشره على طائرات بدون طيار. باستخدام نموذج التعلم العميق YOLOv8 ومجموعات بيانات صور الحرائق، يكتشف النظام الحرائق في الوقت الفعلي ويصدر تنبيهات فورية. تثبتت مقاييس التقييم مثل الدقة والضبط والاستدعاء ودرجة F1 فعاليتها. تستكشف الدراسة تطبيقات التعلم العميق عبر مختلف المجالات وتناقش تفاصيل تنفيذ نموذج YOLOv8. تسلط نتائج الاختبار على أجهزة الكمبيوتر الشخصية و Raspberry Pi والطائرات بدون طيار الضوء على قدرات النظام والتحسينات المحتملة.

الكلمات المفتاحية: YOLO ، الذكاء الاصطناعي، إنترنت الأشياء، الكشف عن الحرائق، طائرة بدون طيار.

Résumé: La fréquence et la gravité croissantes des incendies de forêt présentent d'importants défis en matière d'environnement et de sécurité. Ce projet vise à développer un système de détection et d'alerte d'incendie intégré à l'IA et déployé sur des drones. À l'aide du modèle d'apprentissage profond YOLOv8 et des ensembles de données d'images d'incendie, le système détecte les incendies en temps réel et émet des alertes immédiates. Les mesures d'évaluation telles que l'exactitude, la précision, le rappel et le score F1 démontrent son efficacité. L'étude explore les applications d'apprentissage profond dans divers domaines et discute des détails de mise en œuvre du modèle YOLOv8. Les résultats des tests sur PC, Raspberry Pi et drones mettent en évidence les capacités du système et les améliorations potentielles.

Mots-clés: AI, IoT, détection d'incendie, drone, YOLO.

Abstract: The increasing frequency and severity of forest fire present significant environmental and safety challenges. This project aims to develop an AI-integrated fire detection and alert system deployed on drones. Using the YOLOv8 deep learning model and fire image datasets, the system detects fires in real-time and issues immediate alerts. Evaluation metrics like accuracy, precision, recall, and F1 score demonstrate its effectiveness. The study explores deep learning applications across various domains and discusses the implementation details of the YOLOv8 model. Results from testing on PCs, Raspberry Pi, and drones highlight the system's capabilities and potential improvements.

Keywords: AI, IoT, fire detection, drone, YOLO.

List of Acronyms and Abbreviations

| | |
|------|------------------------------------|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Networks |
| CCD | Charged Couple Device |
| CNN | Convolutional Neural Networks |
| CSP | Cross State Partial |
| CV | Computer Vision |
| DJI | Da Jiang Innovations |
| DL | Deep Learning |
| EOS | Earth Observing System |
| ESCs | Electronic Speed Controllers |
| EO | Earth Observation |
| FP | False Positive |
| FPV | First-Person View |
| GEO | Geostationary Earth Orbit |
| GPUs | Graphic Processing Units |
| GPS | Global Positioning System |
| IDE | Integrated Development Environment |
| IOU | Intersection Over Union |
| IoT | Internet of Things |
| IR | Infrared |

| | |
|----------|-----------------------------------------------|
| LEO | Low Earth Orbit |
| LED | Light Emitting Diode |
| MIMEText | Multipurpose Internet Mail Extensions Text |
| ML | Machine Learning |
| MODIS | Moderate Resolution Imaging Spectroradiometer |
| NASA | National Aeronautics and Space Administration |
| PANet | Path Aggregation Network |
| RF | Radio Frequency |
| RoR | Rate of Return |
| SMTP | Simple Mail Transfer Protocol |
| SSO | Sun-Synchronous Orbit |
| STARTTLS | Start Transport Layers Security |
| TP | True Positive |
| TPUs | Tensor Processing Units |
| TN | True Negative |
| UAVs | Unmanned Aerial Vehicle |
| UIDs | Unique Identifiers |
| VIIRS | Visible Infrared Imaging Radiometer Suite |
| VS Code | Visual Studio Code |
| YOLO | You Only Look Once |

Table of contents

| | |
|-------------------------------------------------------------------------|-------------------------------------|
| General Introduction | Error! Bookmark not defined. |
| Chapter 1: Introduction to Forest Fire Detection | |
| 1.1. Introduction..... | 2 |
| 1.2. Forest Fire in Algeria..... | 2 |
| 1.2.1. Weekly fire alerts in Algeria..... | 2 |
| 1.2.2. Tree cover loss due to fires in Algeria..... | 4 |
| 1.2.3. Comparative between wilayas..... | 5 |
| 1.2.4. Historical Fire Alerts in Algeria | 6 |
| 1.3. Forest Fire Detection Existing Methods..... | 7 |
| 1.3.1. Traditional sensor-based approaches..... | 7 |
| 1.3.2. Satellite and Aerial Surveillance | 11 |
| 1.3.3. Advanced AI and Machine Learning Techniques | 14 |
| 1.4. Forest Fire Detection Existing Systems | 14 |
| 1.4.1. Terrestrial Systems..... | 14 |
| 1.4.2. Unmanned Aerial Systems..... | 15 |
| 1.4.3. Satellite and Space-Based Observation Systems | 16 |
| 1.5. Conclusion | 17 |
| Chapter 2: General Overview of Deep Learning and the Internet of Things | |
| 2.1. Introduction..... | 19 |
| 2.2. Definition of Deep Learning | 19 |
| 2.3. Application domains of Deep Learning | 21 |
| 2.3.1. Autonomous vehicles | 21 |
| 2.3.2. Healthcare | 22 |
| 2.3.3. Robotics | 22 |
| 2.3.4. Agriculture | 22 |
| 2.4. Computer Vision..... | 23 |
| 2.4.1. Definition | 23 |
| 2.4.3. Computer Vision with Deep Learning Approach | 24 |
| 2.5. Internet of Things | 25 |
| 2.5.1. Definition..... | 25 |
| 2.5.2. Devices of Internet of Things..... | 25 |
| 2.5.3. Application domains of Internet of Things | 26 |
| 2.6. Conclusion | 27 |

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- 3.1. Introduction..... 28
- 3.2. Working Environment..... 28
 - 3.2.1. Hardware 28
 - 3.2.2. Software 30
 - 3.2.3. Libraries 31
- 3.3. Evaluation Metrics..... 32
 - 3.3.1. Confusion Matrix 32
 - 3.3.2. Accuracy 33
 - 3.3.3. Precision 33
 - 3.3.4. Specificity..... 33
 - 3.3.5. Recall 33
 - 3.3.6. F1-score 34
- 3.4. The Architecture of The Proposed System 34
 - 3.4.1. Downloading the data 34
 - 3.4.2. Fire detection model 35
 - 3.4.3. YOLOv8 network training process 37
 - 3.4.4. Email Alert System Using SMTP..... 39
 - 3.4.5. Implementation..... 40
- 3.5. Conclusion 45

Chapter 4: Results and Discussions

- 4.1. Introduction..... 46
- 4.2. Tests and Results 46
 - 4.2.1. Confusion Matrix 46
 - 4.2.2. Accuracy 47
 - 4.2.3. Precision 48
 - 4.2.4. Recall 48
 - 4.2.5. F1 Score 48
- 4.3. Discussions 48
 - 4.3.1. Results Interpretation..... 49
 - 4.3.2. Proposed Improvements 49
- 4.4. Testing The Model on the PC..... 50
 - 4.4.1. Tests on videos..... 50
 - 4.4.2. tests on the PC’s webcam..... 52
- 4.5. Testing The Model on Raspberry Pi 53
 - 4.5.1. Tests on video 53
 - 4.5.2. Tests on The Logitech External Camera 53
- 4.6. Testing The System on Drone Deployment..... 55

| | |
|-------------------------|-------------------------------------|
| 4.7. Conclusion | 57 |
| General Conclusion..... | Error! Bookmark not defined. |
| Bibliography..... | 58 |

List of figures

| | |
|---------------------------------------------------------------------------------------------|----|
| Figure 1.1. Fire alerts in Algeria from May 2023-May 2024..... | 3 |
| Figure 1.2. Fire alerts in Algeria week of May 2023 | 3 |
| Figure 1.3. Fire alerts in Algeria week of April 2024 | 4 |
| Figure 1.4. Tree cover loss in Algeria from 2001 to 2023..... | 4 |
| Figure 1.5. Tree cover loss in Algeria in 2017 | 4 |
| Figure 1.6. Map of cumulative burnt areas by Wilayas (2000-2022)..... | 5 |
| Figure 1.7. Map of the cumulative number of forest fire outbreaks by Wilayas | 6 |
| Figure 1.8. Historical Fire Alerts in Algeria 2012-2024..... | 7 |
| Figure 1.9. Generalized multispectral imaging systems for early fire detection | 17 |
| | |
| Figure 2.1. Relation between AI, ML and DL | 20 |
| Figure 2.2. Relationship between AI and CV | 23 |
| Figure 2.3. Structure of the Internet of Things..... | 26 |
| | |
| Figure 3.1. The architecture of the system..... | 34 |
| Figure 3.2. YOLO network architecture | 36 |
| Figure 3.3. Sample frames from the dataset..... | 37 |
| Figure 3.4. GPU information..... | 37 |
| Figure 3.5. Training progress information for the first 10 epochs..... | 38 |
| Figure 3.6. Training's progress information for the last 10 epochs..... | 39 |
| Figure 3.7. SMTP architecture | 39 |
| Figure 3.8. Installing updates on Raspberry Pi | 41 |
| Figure 3.9. Creating and activating a virtual environment on Raspberry Pi | 41 |
| Figure 3.10. Installing ultralytics on raspberry pi | 42 |
| Figure 3.11. Installing cvzone on Raspberry Pi | 43 |
| Figure 3.12. GPS and Raspberry Pi wiring diagram | 43 |

| | |
|------------------------------------------------------------------|----|
| Figure 3.13. Verifying GPS status..... | 44 |
| Figure 3.14. Activating the GPS application | 44 |
| Figure 4.1. Confusion matrix result | 46 |
| Figure 4.2. Test on video 65%..... | 50 |
| Figure 4.3. Test on video 61%..... | 50 |
| Figure 4.4. Email alert for the first video's tests..... | 51 |
| Figure 4.5. Test on video 87%..... | 51 |
| Figure 4.6. Test on video 90%..... | 51 |
| Figure 4.7. Email alert for the second video's tests..... | 51 |
| Figure 4.8. Test on webcam 56% | 52 |
| Figure 4.9. Test on webcam 74% | 52 |
| Figure 4.10. Email alert for webcam's tests..... | 52 |
| Figure 4.11. Test on video 87%..... | 52 |
| Figure 4.12. Email alert for video | 53 |
| Figure 4.13. Test on external webcam..... | 54 |
| Figure 4.14. Test on external webcam 87%..... | 54 |
| Figure 4.15. Email alert for the external webcam | 54 |
| Figure 4.16. System's deployment on the drone..... | 55 |
| Figure 4.17. testing the system on the drone | 56 |
| Figure 4.18. drone's test's result | 56 |
| Figure 4.19. Email alert for the drone's test..... | 57 |

List of tables

| | |
|---------------------------------------------------------------------------|----|
| Table 3.1. Characteristics of the ThinkPad Personal Computer | 28 |
| Table 3.2. The confusion matrix of a classification model | 32 |

In recent years, Algeria has faced significant challenges due to the increasing frequency and severity of wildfires in mountains and forests. Fire detection systems aim to promptly detect fires, triggering manual or automatic actions to prevent escalation and reduce risk to people and minimizing the likelihood of fire development and spread.

Fire detection systems have known a great evolution over the years. Some traditional fire detection systems include: smoke detectors, heat detectors, flame detectors, manual call points and sprinkler systems. With more technological advancements and AI, modern fire detection systems have been developed, such as: smart smoke detectors, video-based fire detection, IoT enabled fire alarms, wireless mesh networks, and predictive analytics.

Smart drones play a very important role in the fire detection field. Drones have undergone a significant evolution since their inception. They were initially developed in the military sector. Nowadays, they serve diverse purposes such as research, mapping, and observing natural phenomena. Over time, these drones have become more sophisticated with improvements in navigation systems, sensors, and data processing capabilities. This evolution has enabled them to be used in a wide range of industries, including agriculture, construction, infrastructure inspection, and environmental monitoring.

One notable advancement in recent years is the integration of Artificial Intelligence (AI) and Machine Learning algorithms into drone systems. This allows drones to analyse vast amounts of data in real-time, enabling them to make autonomous decisions and perform complex tasks with greater efficiency and accuracy. The development of smart fire detection system drones represents a significant milestone in this evolution. These drones are equipped with advanced sensors, such as infrared cameras and gas detectors, which allow them to detect fires quickly and accurately.

The objective of this project is to build a smart fire detection and alert system drone, integrating Artificial Intelligence and Internet of Things by using image-based fire detectors and IoT enabled fire alarms.

Chapter 1

Introduction to Forest Fire Detection

1.1. Introduction

Forest fires pose significant threats, leading to substantial economic and ecological damage while endangering human lives. The importance of early detection and continuous monitoring of forest fires has thus garnered global attention, aiming to preserve natural resources and ensure the safety of both people and property.

To mitigate the risks associated with forest fires, effective and rapid detection techniques are essential. Technological advancements, including drones, satellite imagery, and sophisticated machine learning algorithms, enhance the ability to promptly monitor and detect forest fires, facilitating quicker response and containment efforts.

This opening chapter aims to provide a comprehensive overview of forest fire detection, starting with a specific focus on the incidence of forest fires in Algeria. It then delves into the various existing methods for detecting forest fires. Following this, the chapter explores the different systems currently in use for forest fire detection.

1.2. Forest Fire in Algeria

1.2.1. Weekly fire alerts in Algeria

In Algeria, the peak fire season typically commences in mid-January and extends for 53 weeks. Between the 15th of May, 2023, and 13th of May, 2024, there were 94,005 VIIRS fire alerts reported. This figure is consistent with the data from previous years dating back to 2012 [1]. Figures 1.1, 1.2 and 1.3 show the range of fire alerts in Algeria from May 2023 to May 2024.

Chapter 1: Introduction to Forest Fire Detection

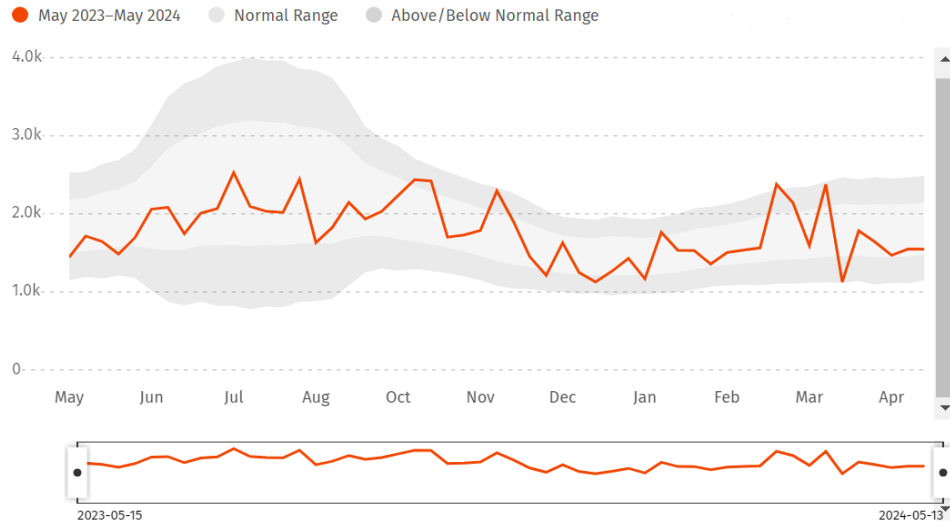


Figure 1.1. Fire alerts in Algeria from May 2023-May 2024 [1]

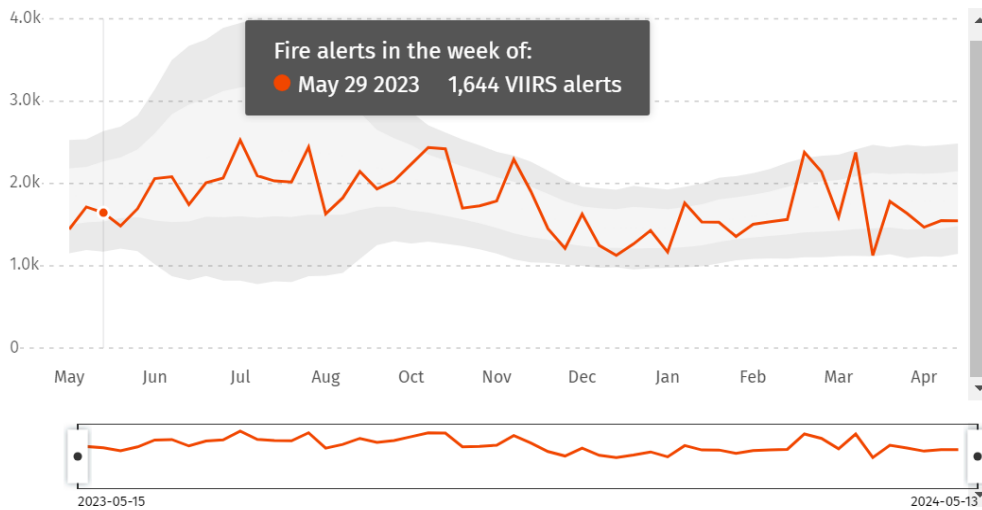


Figure 1.2. Fire alerts in Algeria week of May 2023 [1]

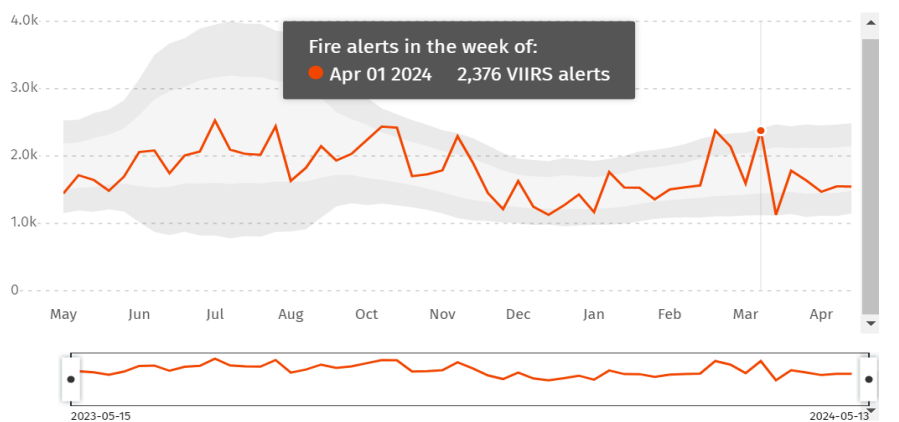


Figure 1.3. Fire alerts in Algeria week of April 2024 [1]

Chapter 1: Introduction to Forest Fire Detection

1.2.2. Tree cover loss due to fires in Algeria

From 2001 to 2023, Algeria experienced a loss of 168 thousand hectares of tree cover due to fires and an additional 59.0 thousand hectares from other causes. The year 2017 witnessed the highest tree cover loss attributable to fires during this period, with 41.7 thousand hectares burned, accounting for 91% of the total tree cover loss for that year. Figures 1.4 and 1.5 show the statistics of tree cover loss in Algeria from 2001 to 2023.

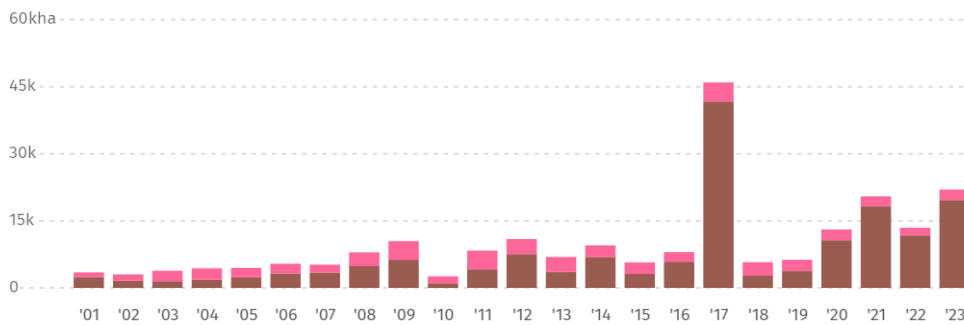


Figure 1.4. Tree cover loss in Algeria from 2001 to 2023 [1]

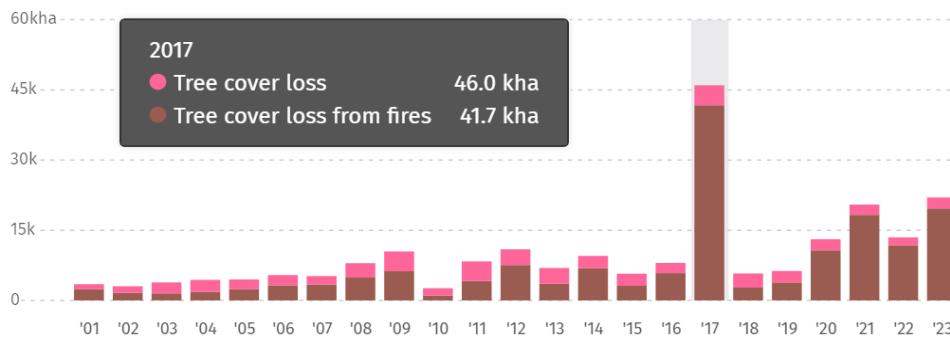


Figure 1.5. Tree cover loss in Algeria in 2017 [1]

- **Regions with the most tree cover loss due to fires in Algeria**

From 2001 to 2023, El Tarf experienced the highest rate of tree cover loss due to fires, averaging 1.71 kha per year. During this period, fires were responsible for 74% of the total tree cover loss in Algeria. The following is a breakdown of the average annual tree cover loss due to fires for the top ten regions:

1. El Tarf: 1.71 kha
2. Bejaia: 979 ha

Chapter 1: Introduction to Forest Fire Detection

3. Skikda: 843 ha
4. Guelma: 547 ha
5. Tizi Ouzou: 421 ha
6. Annaba: 374 ha
7. Jijel: 366 ha
8. Aïn Defla: 355 ha
9. Medea: 237 ha
10. Tipaza: 219 ha

1.2.3. Comparative between wilayas

- In terms of burned area

Figure 1.6 shows a map of cumulative burnt areas by Wilayas from 2000 to 2022 [2].

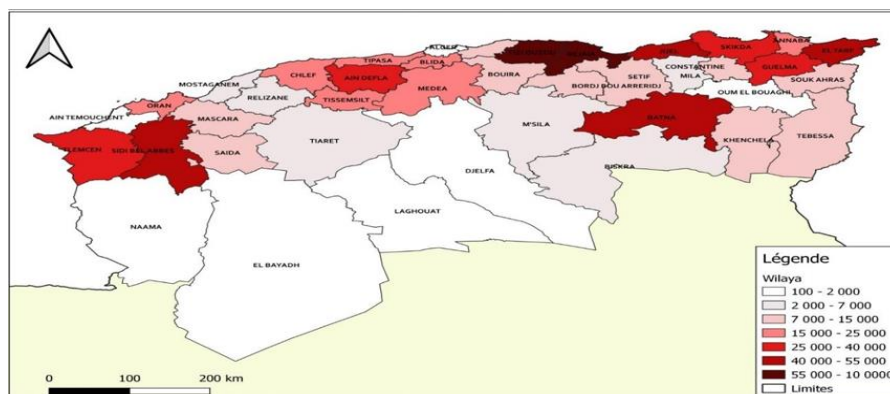


Figure 1.6. Map of cumulative burnt areas by Wilayas (2000-2022) [3]

This map indicates that Tizi Ouzou and Bejaia are the wilayas most affected by forest fires, with a cumulative area of 55,000 to 100,000 Ha burned over 22 years, followed by El Tarf, Batna, and Sidi Bel Abbès, each experiencing a burnt area of 40,000 to 55,000 Ha.

- In terms of the number of forest fire outbreaks

Figure 1.7 shows a map of cumulative number of forest fire outbreaks by wilayas from 2000 to 2022.

Chapter 1: Introduction to Forest Fire Detection

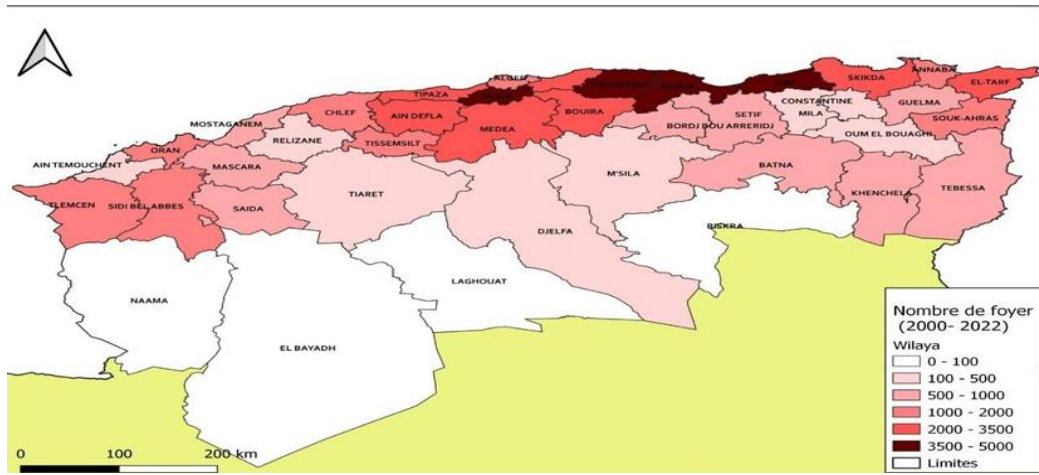


Figure 1.7. Map of the cumulative number of forest fire outbreaks by Wilayas [3]

This map indicates that Tizi Ouzou, Bejaia, Jijel, and Blida have experienced the highest number of forest fires, ranging from 3,500 to 5,000 incidents over 22 years. Followed by El Taref, Skikda, Boumerdes, Bouira, Medea, Ain Defla, and Tipaza, with 2,000 to 3,500 forest fires. Interestingly, although Batna and Sidi Bel Abbas do not have a high number of forest fire outbreaks, they are still among the most affected wilayas by burnt forest areas.

1.2.4. Historical Fire Alerts in Algeria

Between the 2nd of January 2012 and the 20th of May 2024, Algeria experienced a total of 428,655 VIIRS fire alerts. Figure 1.8 illustrates a histogram of the fire alerts in Algeria from 2012 to 2024 [1].

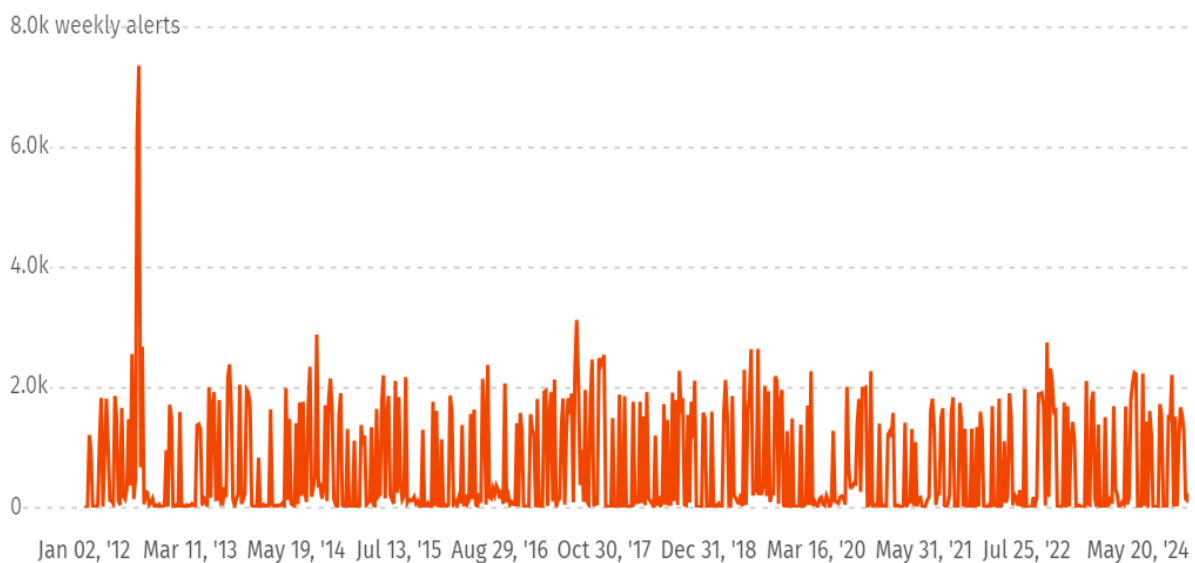


Figure 1.8. Historical Fire Alerts in Algeria 2012-2024 [1]

1.3. Forest Fire Detection Existing Methods

1.3.1. Traditional sensor-based approaches

Traditional sensor-based approaches for forest fire detection involve the use of various sensors to monitor environmental conditions and detect the presence of fire. These sensors include temperature sensors, smoke detectors, infrared (IR) sensors, and gas sensors, among others [4].

a) Thermal sensors

Fire detection systems based on heat sensing activate when ambient temperature or its rate of rise exceeds a predefined threshold. These sensors are unsuitable for early warning since they require proximity to the fire to activate. However, they are cost-effective, easy to install, and highly reliable in areas prone to false alarms from smoke detectors, such as kitchens, and in environments where smoke detectors are ineffective due to extreme temperatures.

Rate-of-rise (RoR) thermal sensing detectors respond to sudden changes in ambient temperature from a normal baseline. These detectors can trigger an alarm with any rapid temperature increase, allowing for a lower activation threshold compared to fixed set points.

An advanced thermal detector for fire detection is the distributed fiber optic temperature sensor. This sensor, utilizing an optical fiber cable, detects temperature fluctuations more quickly than traditional thermal detectors due to its low mass and is immune to various nuisance emissions. Introduced in the late 1980s, distributed fiber optic temperature sensors employing Rayleigh and Raman scattering have been used in challenging environments, including tunnels, underground railways, conveyor lines, steelworks, and the petroleum and chemical industries. Distributed optical fiber sensors using Rayleigh scattering measure temperature by detecting changes in reflected light when the fiber is micro-bent by heat. These sensors are primarily used in road tunnels and underground installations. Raman scattering-based fiber optic sensors detect temperature changes by measuring the ratio of Stokes to anti-Stokes backscattered intensity signals as a function of temperature. The Brillouin scattering-based fiber optic device is a potential successor to both Rayleigh and Raman scattering systems, offering superior spatial and thermal resolutions.

Chapter 1: Introduction to Forest Fire Detection

b) Smoke sensors

Smoke is generated significantly earlier than other fire signatures during the growth and development phases of a fire incident. Early detection of smoke at low levels by fire detection systems maximizes the chances of successful fire suppression, escape, and survivability. Historically, fire detection relied on visual inspection and confirmation by individuals. In the late 1930s, Swiss chemist Walter Jaeger accidentally discovered a sensor capable of detecting smoke while developing a device for toxic gases, leading to modern smoke sensing technologies. The first automatic smoke detector, developed in 1940, used americium-241 as a radioactive source to sense ionization current.

Smoke detection systems are relatively recent innovations, becoming common in residential and life safety applications in the 1970s and 1980s. There are two primary sensing approaches used in commercial fire detection devices: photoelectric detectors, which use light scattering, and ionization detectors. Smoke detectors, designed to mimic the human sense of smell, can identify fires in their smoldering or early flame phases. The main advantage of smoke detectors is their ability to detect fires at the incipient stage, making them the preferred method for life safety and high-value content applications as they provide critical time for emergency response before significant damage occurs. Properly selected and designed smoke detection systems offer high reliability with minimal false alarms, despite being more expensive to install.

Optical smoke sensing devices detect smoke by utilizing the light scattering properties of smoke particles. Ionization smoke sensors contain an ionization chamber with a radioisotope, typically americium. In the absence of smoke, ionized air molecules in the chamber allow a small electrical current to pass between charged electrodes. Smoke particles reduce this current, triggering a fire warning. Standard fire detection systems sense backscattered light from a Light Emitting Diode (LED) reflected by smoke particles. However, these devices often have slow response times, a higher incidence of false alarms, and are unable to detect non-smoke-producing fires.

Chapter 1: Introduction to Forest Fire Detection

c) Infrared IR sensors

Flame detectors mimic human sight and represent a major automated fire detection approach, utilizing line-of-sight systems that operate on infrared, ultraviolet, or combined principles. Radiant energy in the 4,000 to 7,700 angstrom range indicates a flaming condition. Flame detection devices identify this fire signature and transmit a signal to the fire notification module. Unlike thermal and smoke-based methods, flame detection requires a direct line of sight between the detector and the fire source. Despite this limitation, flame detectors are highly reliable in noisy environments and are often used in high-value energy and transportation applications where traditional sensors might fail due to spurious activities. Common applications include locomotive and aircraft maintenance facilities, refineries, fuel loading stations, and mines.

Infrared detectors recognize fire by detecting the characteristic flicker of flames, while ultraviolet detectors identify fires by detecting ultraviolet radiation emitted by flames. Flame detection devices can safeguard large areas and respond quickly as they do not rely on smoke or heat from combustion. However, false alarms can occur due to radiation from sources like welding, sunlight, and tungsten lamps. To mitigate these nuisance alarms, multi-wavelength radiation sensing and computational algorithms have been developed to determine object temperature, flame temperature, surface area, and flame presence.

Optical flame detectors offer higher reliability, greater long-term stability, and rapid response compared to smoke detection systems. They operate within specific spectral ranges to record electromagnetic energy at designated wavelengths, including infrared, visible, and ultraviolet.

Ultraviolet-only detectors operate at wavelengths shorter than 400 nm, detecting flames at speeds of 3-4 ms due to the high-energy radiation generated at ignition. Ultraviolet and infrared detectors compare threshold signals in two spectral ranges and their ratio to determine fire signal reliability and minimize false alarms. Commercial ultraviolet flame detectors, typically based on a Geiger-Muller counter—a quartz tube filled with inert gas that conducts electricity when a photon between 185 and 260 nm is detected—are large, expensive, high-voltage, short-lived, and can interfere with nearby electronics. Therefore, compact, lower-voltage semiconductor ultraviolet sensors are preferred.

D) Gas sensors

Volatile chemicals are typically released during fire combustions before smoke particles are generated. Therefore, fire detection devices based on chemical gas sensors may respond faster than current smoke detectors. The key hypothesis for developing gas sensor-based fire detection is that many fire types produce gases and volatiles before smoke, allowing for shorter response times.

The initial step in using chemical gas sensors for fire detection involves sensitivity analysis of various sensing technologies to combustion products, primarily carbon monoxide and carbon dioxide. However, the feasibility of using chemical sensors for fire detection is challenged by their high cross-sensitivity to water vapor and a range of volatiles produced during everyday activities, making them susceptible to false alarms. Thus, designing robust fire detection devices necessitates analyzing sensor sensitivity to combustion products and examining cross-sensitivity to interfering scenarios.

To address the drawbacks of cross-sensitivity and enhance robustness, sensor fusion algorithms have been explored. These algorithms aim to accurately detect fires while minimizing nuisance effects. Various computational methods, such as logic rules, neural networks, probabilistic neural networks, hierarchical linear discriminant analysis, and k-nearest neighbours, have been investigated, but further work is needed to improve detection times and reduce false alarms in chemical sensing-based fire detection.

A multi-sensor approach, combining multiple fire signatures, is motivated by the need to develop robust fire detection systems with fewer false alarms. This approach has driven recent research efforts towards developing advanced algorithms and image processing methods based on data from multiple sensors. Multi-sensor detectors, which combine different sensor types, effectively mitigate the shortcomings of single-sensor systems.

Chapter 1: Introduction to Forest Fire Detection

Fonollosa et al. (2016) presented a multi-sensor system based on chemical sensing for fire detection. This system, tested in a sensing chamber with various fire types and interferences, was able to distinguish fire from non-fire situations and activated alarms faster than conventional smoke detection systems for certain fire types. However, there is still a need to further reduce response time and enhance robustness in chemical sensing-based fire detection devices.

In an effort to increase sensitivity and reduce false alarms, Sowah et al. (2014) developed a multi-sensor fire detection and alarm system using fuzzy logic. A microcontroller processed data from smoke, temperature, and flame sensors to confirm fire status using a fuzzy logic algorithm.

Díaz-Ramírez et al. (2012) proposed and evaluated two algorithms for forest fire detection based on information fusion techniques. The first algorithm uses a threshold method with nodes attached to temperature, humidity, and light sensors. The second algorithm employs the Dempster-Shafer theory, assuming nodes use temperature and humidity sensors. While gas sensing-based fire detection systems are well-suited for rapid fire detection due to their high sensitivity to early volatiles release, they have low specificity due to responses to non-fire volatiles. Pattern recognition algorithms can enhance the effectiveness of gas sensor-based fire detection methods [3].

1.3.2. Satellite and Aerial Surveillance

a) Manned aircraft surveillance

Fixed-wing aircraft and helicopters equipped with advanced cameras, infrared sensors, and various other instruments are widely utilized for detecting smoke and other early indicators of forest fires. Pilots conduct aerial surveys over remote and inaccessible areas, using high-resolution cameras to meticulously scan for smoke plumes or fire hotspots [5].

These high-resolution cameras provide detailed images, enabling precise identification of potential fire outbreaks. Additionally, infrared sensors play a crucial role in this detection process, as they are capable of identifying heat signatures that emanate from wildfires, even through dense smoke or forest canopies.

Chapter 1: Introduction to Forest Fire Detection

The combination of these technologies allows for the effective monitoring of large forested areas, ensuring that any signs of fire are promptly detected and reported. This enables rapid response efforts to be mobilized, thereby mitigating the potential damage and spread of the fires, protecting both the environment and surrounding communities.

b) Drones or UAVs

Drones, or Unmanned Aerial Vehicles (UAVs), are effectively utilized for the detection and monitoring of forest fires. These drones can be equipped with cameras, thermal sensors, and various other instruments to detect smoke and heat signatures. The high-resolution cameras provide detailed images that help in identifying the precise location of fire outbreaks, while the thermal sensors detect heat signatures even through dense smoke or forest canopies. Additionally, drones are capable of mapping the location and extent of a fire, offering a comprehensive view of the affected area. They continuously monitor the fire's progression, providing real-time data that is crucial for coordinating firefighting efforts and strategizing response actions.

This continuous monitoring allows for the assessment of fire behaviour, aiding in predicting its spread and impact. Furthermore, the use of drones reduces the risk to human life by allowing remote surveillance of hazardous areas that are difficult or dangerous for manned aircraft to access. This technological integration enhances the efficiency and safety of forest fire management operations. Examples of aerial surveillance systems used for forest fire detection include the DJI Mavic 2 Enterprise Dual, the Lockheed Martin INDAGO, and the Insitu ScanEagle. The DJI Mavic 2 Enterprise Dual, equipped with visual and thermal cameras, quickly detects and monitors fires through smoke and in low-light conditions. The Lockheed Martin INDAGO, with its electro-optical and infrared camera, is rugged and weather-resistant, suitable for challenging environments and early fire warnings. The Insitu ScanEagle, a larger drone with a thermal camera, can stay aloft for up to 24 hours, covering large areas and transmitting real-time data to command centers.

These are just a few examples of aerial surveillance systems that incorporate drones for forest fire detection. Drones are useful for monitoring forested areas and can provide early warning of potential fires, allowing firefighters and other first responders to respond quickly and effectively.

c) Satellite Surveillance

Satellites equipped with sensors capable of detecting changes in temperature, reflectance, and other indicators are extensively used for forest fire detection. These advanced sensors can identify the heat emitted by active fires, enabling early detection and rapid response. Additionally, they can detect smoke plumes, providing visual confirmation of fire locations and helping to assess the spread and intensity of the fires. Furthermore, these sensors monitor changes in vegetation, such as scorching or burning, that occur as a result of a fire. By capturing comprehensive data on temperature fluctuations, smoke, and vegetation changes, satellite-based systems offer a powerful tool for continuous and large-scale monitoring of forested areas, enhancing the effectiveness of fire management and mitigation strategies. Examples of aerial surveillance systems for forest fire detection include MODIS, Landsat, and Sentinel-2 satellites. MODIS, on NASA's EOS satellites, detects global active fires and hotspots with sensors measuring surface heat, identifying fires as small as 10 meters with near-real-time data.

Landsat provides multispectral data to detect vegetation changes indicating fire risk, helping create fire risk maps. Sentinel-2, from the EU's observation mission, offers high-resolution imagery to monitor vegetation changes and potential fires, providing early warnings. Additionally, fire towers use optical instruments to spot smoke or fire and connect to central monitoring systems for responder communication.

Aerial surveillance provides an important complement to ground-based systems for forest fire detection. By using a combination of ground-based and aerial surveillance systems, forest managers can develop a comprehensive strategy for detecting and responding to wildfires. Aerial surveillance can also be used for forest fire prediction, in addition to detection. Here are some examples of how aerial surveillance is used for forest fire prediction.

1.3.3. Advanced AI and Machine Learning Techniques

The rapid development of digital camera technology and video processing has led to the replacement of conventional fire detection systems with computer vision-based systems. These systems operate through three stages: flame pixel classification, segmentation of moving objects, and analysis of candidate regions. The fire pixel classifier is crucial, as it identifies key areas for the system to analyse, requiring high accuracy and low false detection rates. A typical video flame detection system employs background subtraction and colour analysis to pinpoint potential flame regions in video frames, using attributes like colour probability to differentiate between fire and non-fire objects. Some methods also consider spatial variance, temporal variation, and contour variability of candidate regions, applicable to both greyscale and colour video sequences.

Deep learning has further enhanced fire detection by automating feature extraction and classification. Unlike traditional computer vision methods, where features are manually crafted by experts, deep learning algorithms automatically capture these features through extensive training on diverse datasets. This shifts the focus from feature engineering to designing effective neural network architectures and preparing comprehensive training data. In deep learning, the detector and classifier are trained simultaneously within the same neural network, making the design of an efficient network structure and training process critical for optimal performance [6].

1.4. Forest Fire Detection Existing Systems

Forest fire detection systems can be categorized into three major groups: terrestrial systems, UAV systems, and satellite systems [6].

1.4.1. Terrestrial Systems

Terrestrial-based early detection systems consist of either individual sensors, such as fixed, PTZ, or 360° cameras, or networks of ground sensors. To provide adequate visibility, these sensors must be strategically placed, typically in watchtowers situated at high vantage points to monitor high-risk areas. These watchtowers are used not only for fire detection but also for verification and localization.

Chapter 1: Introduction to Forest Fire Detection

There are two primary types of cameras used for early fire detection: optical cameras and infrared cameras, capable of gathering data with resolutions ranging from low to ultra-high for various detection scenarios. Recently, systems that combine both optical and infrared cameras have been introduced. Computer-based solutions can process large amounts of data while maintaining high accuracy and a low false alarm rate.

1.4.2. Unmanned Aerial Systems

Terrestrial imaging systems can detect both flame and smoke; however, it is often nearly impossible to view wildfire flames in a timely manner from ground-based or forest watchtower-mounted cameras. To address this limitation, autonomous unmanned aerial vehicles (UAVs) offer a broader and more precise view of fires from above, even in regions that are inaccessible or too hazardous for firefighting crews. Fixed or rotary-wing UAVs cover larger areas and offer greater flexibility in monitoring, though they are subject to weather conditions and have limited flight durations. UAVs dedicated to forest fire monitoring and detection are in high demand due to their rapid maneuverability and enhanced personnel safety.

A UAV-based forest fire surveillance system typically comprises a team of UAVs equipped with various on-board sensors and a central ground station. The system aims to use UAVs to detect and track flames, predict their spread, and provide real-time fire information to human firefighters, as well as to assist in fire suppression.

The system performs fire monitoring (searching for prospective fires), detection (identifying potential fires and alerting firefighting personnel), diagnosis (calculating fire position, extent, and evolution), and prognosis (predicting fire propagation).

A crucial component of the UAV-based forest fire monitoring system is the computer vision-based fire detection technique. This technique offers multiple advantages, including the ability to monitor a wide range of objects, provide intuitive and real-time images, and conveniently record information. Typically, charge-coupled device (CCD) cameras and infrared (IR) cameras are mounted on UAVs. Significant efforts have been dedicated to developing more effective image processing schemes for fire detection.

Chapter 1: Introduction to Forest Fire Detection

Colour and motion aspects in CCD camera visual images are typically used for fire detection. However, in some outdoor applications, CCD cameras are often considered insufficiently durable and reliable. Given the highly complex and unstructured forest environments, the possibility of smoke covering the fire, and the presence of fire analogues such as reddish leaves swaying in the wind and light reflections, the false fire alert rate is typically quite high. IR cameras, which can capture images in low or no light conditions and render smoke as transparent, are extensively used despite being more expensive than CCD cameras. They are capable of capturing monochromatic images both day and night. The use of IR cameras is anticipated to reduce the rate of false fire alarms and enhance the forest fire detection system's adaptability to various operating conditions.

1.4.3. Satellite and Space-Based Observation Systems

Satellite detection is a widely used technique for monitoring forest wildfires globally. Recently, the increase in satellite launches and the reduction in associated costs have spurred significant research efforts to detect forest fires from satellite images. Satellites designed for Earth observation (EO) are commonly used for environmental monitoring and meteorology. They are categorized based on their orbits, each offering distinct advantages and disadvantages.

The main categories include (a) geostationary orbit (GEO), which provides a constant view of the same surface area from an altitude of 35,786 kilometers; (b) low Earth orbit (LEO), which operates at altitudes of 2000 kilometers or less and offers high bandwidth and low communication latency; and (c) polar sun-synchronous orbit (SSO), which allows satellites to cross the equator at the same local time during each orbit, ensuring consistent observation conditions.

EO satellites often use low Earth polar SSO orbits to maintain consistent lighting and shadow conditions in their observations. Sun-synchronous satellites provide high spatial resolution data but have low temporal resolution, while geostationary satellites offer high temporal resolution but lower spatial resolution. For instance, satellites like Landsat and Sentinel have long revisit periods, making them unsuitable for real-time active forest fire detection but useful for less time-sensitive tasks such as estimating burnt areas. Figure 1.9 shows a generalized multispectral imaging system for early fire detection [3].

Chapter 1: Introduction to Forest Fire Detection

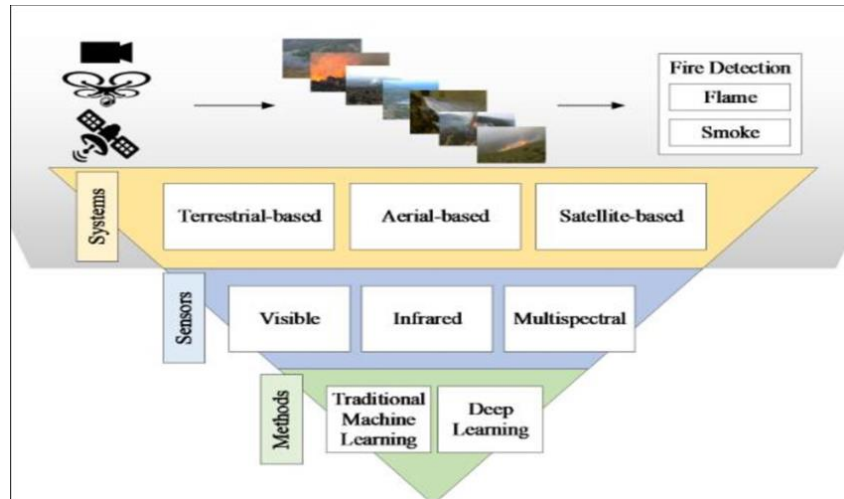


Figure 1.9. Generalized multispectral imaging systems for early fire detection [7]

1.5. Conclusion

This chapter highlights the essential need for effective forest fire detection to reduce the severe economic, ecological, and human toll of such events. Focusing on Algeria, the chapter outlines the extent and frequency of forest fires, underscoring the urgency for advanced detection methods.

Traditional sensor-based approaches, including thermal, smoke, infrared, and gas sensors, are foundational but have limitations like delayed responses and high false alarm rates. Innovations such as distributed fiber optic temperature sensors and multi-sensor systems improve detection speed and reliability.

Aerial surveillance, using manned aircraft and drones equipped with high-resolution and thermal cameras, provides detailed, real-time monitoring, especially valuable in remote or hazardous areas. Satellite surveillance enhances these efforts with continuous, large-scale monitoring using advanced sensors that detect temperature changes, smoke plumes, and vegetation alterations.

Advanced AI and machine learning techniques revolutionize fire detection by automating feature extraction, improving accuracy, and enabling better prediction and response.

Chapter 1: Introduction to Forest Fire Detection

By integrating terrestrial, UAV, and satellite-based systems, forest managers can develop a comprehensive strategy for early detection and effective response, safeguarding natural resources, property, and human lives from forest fires.

Chapter 2

General Overview of Deep Learning and Internet of Things

Chapter 2: General Overview of Deep Learning and Internet of Things

2.1. Introduction

Deep learning and the Internet of Things (IoT) are revolutionary technologies that drive innovation and efficiency across many industries. Deep learning, a branch of artificial intelligence, uses neural networks to analyze large datasets, enhancing tasks like image and speech recognition, predictive analytics, and autonomous control. Its applications span diverse fields, including autonomous vehicles, healthcare, robotics, and agriculture.

IoT connects physical devices to the digital world, facilitating data collection and exchange across networks. This connectivity supports applications in smart cities, industrial automation, and environmental monitoring, improving efficiency, safety, and sustainability.

This chapter provides an overview of deep learning and IoT, starting with their definitions and key applications. It explores deep learning's role in various domains, the principles of Convolutional Neural Networks (CNNs), and the evolution of computer vision. It also examines IoT's applications and development, emphasizing the synergy between these technologies. Understanding these foundations highlights their impact on modern technological advancements and solutions to contemporary challenges.

2.2. Definition of Deep Learning

Artificial Intelligence (AI) is a field within computer science dedicated to developing systems that can perform tasks typically requiring human intelligence. To accomplish this, AI involves programming machines with rules and algorithms that emulate human-like intelligence. AI is divided into two main subcategories: Machine Learning (ML) and Deep Learning (DL) [3] as shown in figure 2.1.

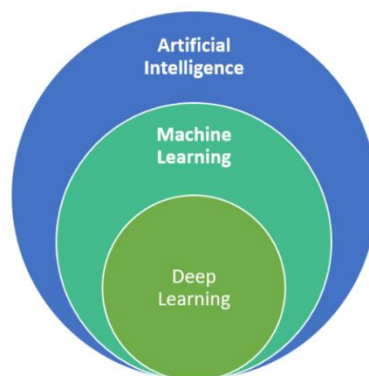


Figure 2.1. Relation between AI, ML and DL [8]

Chapter 2: General Overview of Deep Learning and Internet of Things

Machine Learning (ML) is a subset of AI that enables machines to learn from data without explicit programming. It involves processing data in a manner that allows machines to make decisions and adapt their algorithms based on new information. ML encompasses two main types: supervised learning, where the algorithm is trained on labelled data, and unsupervised learning, where the algorithm identifies patterns and structures in unlabelled data. The primary objective of ML algorithms is to minimize errors and enhance the accuracy of predictions.

Deep Learning (DL), is a subset of ML that uses computational models inspired by the architecture of biological neural networks in the human brain. Similar to the brain, DL processes new information by comparing it to known data, categorizing, and labelling it to derive meaning. This capability allows DL to extract relevant information and make precise predictions.

Artificial neural networks (ANNs) comprise interconnected nodes that perform mathematical operations on input data. These nodes are organized into three layers: input, hidden, and output. The input layer receives raw data, the hidden layers extract patterns and relationships, and the output layer generates the final result of data processing.

Deep Artificial Neural Networks (ANNs) are more complex than shallow ANNs due to their multiple hidden layers. This increased depth enables them to perform more intricate tasks by progressively recombining and refining simpler features into more complex ones as data passes through each layer.

Chapter 2: General Overview of Deep Learning and Internet of Things

In a traditional machine learning workflow, the initial step involves manually extracting relevant features from images to build a classification model. In contrast, deep learning automates this process by using neural network architectures to automatically extract pertinent features from images. Moreover, deep learning employs "end-to-end learning," where the network learns to complete a given task, such as classification, directly from raw data.

Deep Learning (DL) is particularly effective for unstructured data and generally provides greater accuracy than Machine Learning (ML). However, achieving these results with DL requires a substantial amount of training data and significant hardware and software resources.

2.3. Application domains of Deep Learning

The potential of deep learning is extensive and diverse, with applications spanning across multiple domains [3], including:

2.3.1. Autonomous vehicles

Deep learning is crucial in the development of self-driving cars, providing the technology necessary for these vehicles to identify objects, predict the behaviour of other vehicles and pedestrians, and determine the most efficient routes.

By leveraging advanced neural networks, self-driving cars can process vast amounts of sensor data in real-time, allowing them to make split-second decisions with a high degree of accuracy. This capability not only enhances the safety of autonomous vehicles by significantly reducing the likelihood of accidents but also improves overall traffic flow and efficiency. The integration of deep learning algorithms ensures that self-driving cars can adapt to complex and dynamic driving environments, continuously learning and improving their performance. As a result, the widespread adoption of self-driving cars powered by deep learning holds the potential to revolutionize transportation, leading to safer roads, reduced traffic congestion, and more efficient travel.

Chapter 2: General Overview of Deep Learning and Internet of Things

2.3.2. Healthcare

Deep learning is extensively utilized in medical imaging to enhance diagnostic accuracy, facilitate early disease detection, and advance drug discovery. By analysing complex medical images, deep learning algorithms can identify patterns and anomalies that may be difficult for human practitioners to detect, thereby improving the precision of diagnoses. Furthermore, deep learning techniques are instrumental in predicting patient outcomes and personalizing treatment plans. By processing vast amounts of patient data, these algorithms can identify the most effective treatments tailored to individual patients, optimizing therapeutic strategies. The application of deep learning in medical imaging not only accelerates the diagnostic process but also contributes to more targeted and effective healthcare, ultimately improving patient outcomes and advancing the field of medical science.

2.3.3. Robotics

Deep learning is integral to the fields of robot vision, motion control, and grasping, empowering robots to interact with their environment and execute complex tasks. By utilizing deep learning algorithms, robots can interpret visual data to recognize and analyse objects, navigate through diverse settings, and make precise movements. This capability allows robots to perform intricate tasks such as picking up and manipulating objects, assembling components, and even performing delicate operations. The integration of deep learning in robotics not only enhances the accuracy and efficiency of these tasks but also expands the potential applications of robots in various industries, from manufacturing and logistics to healthcare and service sectors.

Through continuous learning and adaptation, robots equipped with deep learning capabilities can improve their performance over time, leading to more intelligent and versatile robotic systems.

2.3.4. Agriculture

Deep learning is extensively utilized for crop and soil analysis, yield prediction, and disease detection, aiding farmers in making informed decisions to optimize their yields, thereby enhancing food security and sustainability.

Chapter 2: General Overview of Deep Learning and Internet of Things

In summary, deep learning is particularly suited for emerging application areas due to its reliance on large datasets, specialized hardware, and automatic feature engineering. The advent of high-performance hardware, such as GPUs, has significantly accelerated the training of deep learning models, enhancing their efficiency. Moreover, deep learning's capability to automatically extract high-level features from raw data renders it more effective than traditional machine learning approaches. Given these advantages, deep learning is poised to be applied in numerous new areas and is anticipated to provide innovative solutions to complex problems across various fields as the technology continues to advance and evolve.

2.4. Computer Vision

2.4.1. Definition

Computer vision (CV) is a multidisciplinary field that aims to replicate aspects of the human vision system, enabling computers to interpret and understand digital images. It is considered a subfield of artificial intelligence (AI) and machine learning (ML) that focuses on allowing computers to "see" and analyse visual data. The primary objective of computer vision is to extract meaningful information from images, such as identifying objects, generating text descriptions, or creating three-dimensional models. This often involves developing techniques that mimic the complexity of human vision. Figure 2.2 demonstrates the relationship between AI and CV [6].

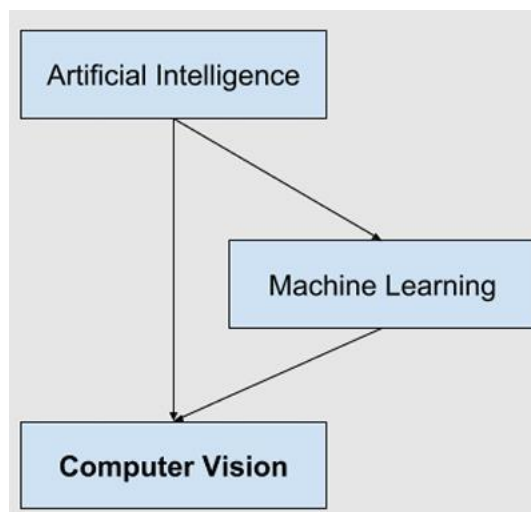


Figure 2.2. Relationship between AI and CV [6]

Chapter 2: General Overview of Deep Learning and Internet of Things

2.4.2. The Evolution of Computer Vision

Early computer vision research began in the 1950s, and by the 1970s, it was being applied commercially to distinguish between typed and handwritten text. Today, the applications of computer vision have advanced significantly.

Before the advent of deep learning, the capabilities of computer vision were quite limited and required extensive manual coding and effort from developers and human operators. For instance, running an object recognition model involves several steps:

- **Creating a database:** Individual images of all the subjects to be tracked had to be captured in a specific format.
- **Annotating images:** Numerous critical data points had to be manually entered for each individual image.
- **Capturing new images:** New images had to be captured, either from photographs or video content. Subsequently, the measurement process had to be repeated, noting key points in the new image. The angle from which the photo was taken also had to be considered. After all this meticulous labour, the application could compare the measurements in the new image to those in its database and determine if it matched any of the profiles being tracked. In reality, most of the work was done manually, with very little automation, and the margin of error remained substantial.

Recent advances in artificial intelligence, particularly in deep learning and neural networks, coupled with the vast amounts of data we generate today, have significantly enhanced computer vision. AI has now surpassed human capabilities in several tasks related to recognizing and labeling objects.

2.4.3. Computer Vision with Deep Learning Approach

Deep learning represents a fundamentally new approach to computer vision. It is based on neural networks, which are versatile functions capable of solving problems represented by instances. By providing a neural network with a large number of labeled data instances, it can identify common patterns and transform them into a mathematical equation to categorize future data. Deep learning is an extremely effective algorithm for computer vision tasks.

Chapter 2: General Overview of Deep Learning and Internet of Things

Developing a robust deep learning system typically involves accumulating a vast amount of labelled training data and fine-tuning parameters such as the type and number of layers.

Deep learning is easier and faster to design and deploy compared to previous methods. It is employed in most contemporary computer vision applications, including fire detection, self-driving cars, and facial recognition. Advances in hardware and cloud computing resources have enabled deep learning to evolve from a theoretical concept to a practical application.

2.5. Internet of Things

2.5.1. Definition

The Internet of Things (IoT) is a sophisticated network of interconnected computing devices, mechanical and digital machines, objects, and even living organisms, all equipped with unique identifiers (UIDs) and the ability to transfer and receive data over the Internet. This system operates seamlessly without necessitating human-to-human or human-to-computer interaction. IoT devices communicate autonomously within the network, enabling real-time data exchange and automation of various processes. This technology encompasses a wide range of applications, from smart home systems and industrial automation to healthcare monitoring and environmental management, thereby revolutionizing the way data is collected, analysed, and utilized across multiple domains.

2.5.2. Devices of Internet of Things

IoT hardware can be categorized into two categories: general devices and sensing devices.

- a) **General devices:** general devices serve as the core components of data hubs and information exchange systems, connected via either wired or wireless interfaces.
- b) **Sensing devices:** which include sensors and actuators, measure parameters such as temperature, humidity, light intensity, and other environmental factors.

A complete IoT system consists of four distinct components: sensors, connectivity, data processing, and a user interface. Sensors collect data from their environment and transmit it to the cloud. These sensors are connected to the cloud through various means, such as cellular networks, Wi-Fi, Bluetooth, or directly via Ethernet.

Chapter 2: General Overview of Deep Learning and Internet of Things

Once the data reaches the cloud, it is processed by software. The processed information is then made useful to the end-user, either through alerts (such as email, text, or notifications) or by enabling the user to perform actions that affect the data. Some actions may also be performed automatically. The figure below shows the structure of Internet of Things.

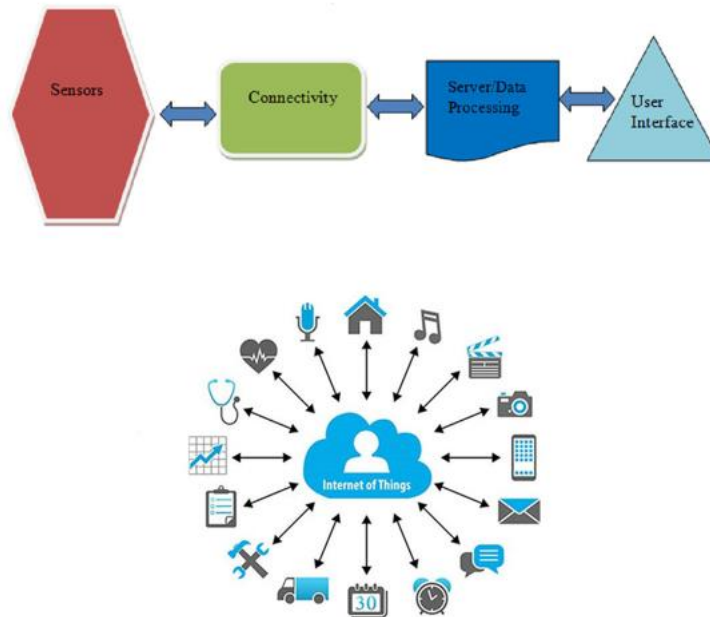


Figure 2.3. Structure of the Internet of Things [7]

2.5.3. Application domains of Internet of Things

The Internet of Things (IoT) has broad application domains, significantly impacting various sectors:

- **Industrial:** IoT enhances operational efficiency through predictive maintenance, real-time monitoring, and automation of manufacturing processes.
- **Medical:** IoT facilitates remote patient monitoring, improves diagnostic accuracy, and streamlines healthcare delivery via connected medical devices.
- **House Automation:** IoT enables smart homes where devices such as thermostats, lighting, and security systems are interconnected, providing convenience, energy efficiency, and enhanced security.

Chapter 2: General Overview of Deep Learning and Internet of Things

- **Business:** IoT optimizes supply chain management, improves customer experiences through personalized services, and increases operational efficiency with connected office equipment and infrastructure.

2.6. Conclusion

In conclusion, the exploration of deep learning and the Internet of Things (IoT) in this chapter sheds light on their transformative impact on various industries. Deep learning, as a subset of artificial intelligence, employs neural networks to analyse extensive datasets, enhancing tasks like image and speech recognition, predictive analytics, and autonomous control. Its applications span diverse fields, including autonomous vehicles, healthcare, robotics, and agriculture.

On the other hand, the Internet of Things connects physical devices to the digital world, facilitating data collection and exchange across networks. This connectivity supports applications in smart cities, industrial automation, and environmental monitoring, significantly improving efficiency, safety, and sustainability.

This chapter has provided a comprehensive overview of deep learning and IoT, beginning with their definitions and key applications. It has delved into deep learning's role in various domains, the principles of Convolutional Neural Networks (CNNs), and the evolution of computer vision. Additionally, it has examined IoT's applications and development, emphasizing the synergy between these technologies. Understanding these foundations highlights their impact on modern technological advancements and solutions to contemporary challenges.

Chapter 3

Design and Implementation of Forest Fire Detection Model Using YOLO

3.1. Introduction

Recent advancements in artificial intelligence (AI) and deep learning have significantly enhanced the efficacy of image-based modelling and analysis (e.g., classification, real-time prediction, and image segmentation) across various applications. Additionally, the advent of nanotechnology in semiconductors has enabled the development of a new generation of Tensor Processing Units (TPUs) and Graphics Processing Units (GPUs), which offer extraordinary computational capabilities for data-driven methods. Moreover, modern drones and unmanned aerial vehicles (UAVs) can now be equipped with compact edge TPU/GPU platforms, enabling real-time on-board processing to facilitate early fire detection and prevent catastrophic events.

To develop such a deep learning-based computer vision system for fire detection, two essential elements are required: the dataset and the trained model. In this chapter, we will discuss the techniques and approaches employed in designing and training the fire detection model.

3.2. Working Environment

3.2.1. Hardware

The hardware tools that have been used are:

- **A Thinkpad Personal Computer**

| | |
|-------------------------|------------------------------|
| Processor | AMD Ryzen 5 pro 4650U |
| RAM | 16Go |
| Hard disk | 512Go SSD |
| Operating system | Windows 11 pro x64 |
| Graphics card | AMD Radeon (TM) Graphics |

Table 3.1. Characteristics of the Thinkpad Personal Computer

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- **FPV drone (quadcopter):** First-person view (FPV) drones are high-performance unmanned aerial vehicles designed for speed, agility, and precise control. They are commonly used in drone racing and freestyle flying. FPV drones typically feature advanced components such as powerful brushless motors, efficient electronic speed controllers (ESCs), and sophisticated flight controllers, which enable responsive and stable flight. Their frames are often lightweight and durable, made from materials like carbon fiber, to withstand crashes and impacts. FPV drones are known for their high maneuverability and the ability to perform complex aerial maneuvers.
- **Raspberry pi 3 model B+:** Raspberry Pi 3 Model B+ is a compact single-board computer renowned for its versatility and affordability. It features a quad-core ARM Cortex-A53 processor running at 1.4GHz, coupled with 1GB of RAM, providing ample computing power for a wide range of applications. Equipped with built-in wireless networking (802.11ac Wi-Fi and Bluetooth 4.2), Gigabit Ethernet, HDMI output, and USB ports, the Raspberry Pi 3 Model B+ offers seamless connectivity and compatibility with various peripherals. Its compact form factor and low power consumption make it an ideal platform for educational projects, DIY electronics, prototyping, and embedded systems development.
- **Flysky remote:** Flysky remote is a handheld radio transmitter commonly used in remote control applications, particularly in the field of unmanned aerial vehicles (UAVs) such as FPV drones. It operates on radio frequency (RF) signals and provides precise and reliable control over the drone's flight parameters. With ergonomic design and intuitive controls, the Flysky remote offers ease of use and responsiveness, making it an essential tool for pilots in the operation and maneuvering of FPV drones. Featuring a reliable control range of up to 1 kilometer, the Flysky remote ensures stable communication with the drone over significant distances, enabling efficient and controlled flight operations.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- **BN220 GPS:** BN220 GPS module is a compact and highly accurate Global Positioning System (GPS) receiver designed for various navigation and positioning applications. It utilizes advanced satellite positioning technology to provide precise location data, including latitude, longitude, altitude, and time. With its small form factor and low power consumption, the BN220 GPS module is suitable for integration into a wide range of devices, including drones, robotics, and IoT (Internet of Things) devices, enabling accurate and reliable positioning capabilities for diverse applications.
- **USB Logitech webcam:** Logitech USB webcam is a compact imaging device designed for high-quality video capture and conferencing. It features a high-resolution camera sensor capable of capturing crisp and clear video at various resolutions, including HD (720p) or Full HD (1080p). With plug-and-play USB connectivity, a built-in microphone, and autofocus capabilities, the Logitech USB webcam provides an immersive and user-friendly video conferencing experience, making it ideal for remote communication, online meetings, and live streaming.

3.2.2. Software

For the software part, we have used:

- **Google Colab:** Google Colab, short for Google Colaboratory, is a cloud-based platform provided by Google that allows users to write, execute, and share Python code in a collaborative online environment. It provides free access to computing resources, including GPU and TPU accelerators, enabling users to run code without the need for expensive hardware. Google Colab is widely used for data analysis, machine learning, and scientific research, offering seamless integration with popular libraries such as TensorFlow, PyTorch, and OpenCV.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- **Google Drive:** Google Drive is a cloud-based storage service provided by Google that allows users to store, access, and share files securely from any device with an internet connection. It provides users with free storage space and offers seamless integration with other Google services such as Google Docs, Google Sheets, and Google Slides. With Google Drive, users can store various types of files, including documents, images, videos, and more, and easily access them from anywhere, at any time. Additionally, Google Drive allows for collaborative work, enabling multiple users to edit and comment on documents in real-time.
- **Visual Studio Code:** VS Code is a lightweight, free source code editor developed by Microsoft. It supports multiple programming languages and frameworks and provides a rich set of features for coding, debugging, and version control. With an intuitive user interface, built-in tools like IntelliSense and a terminal, and a vast library of extensions, VS Code offers a highly customizable and productive coding environment used by developers across various software development domains.
- **Python IDE:** Python Integrated Development Environment (IDE) is a software application designed for efficient Python programming. It offers features like syntax highlighting, code completion, debugging tools, and project management capabilities. Popular Python IDEs include PyCharm, Visual Studio Code, Spyder, and IDLE, enhancing developers' productivity and facilitating the creation of high-quality Python applications.

3.2.3. Libraries

The libraries that have been used in this project are:

- **Ultralytics:** Ultralytics is a software library focused on computer vision, offering state-of-the-art deep learning models and tools for various applications.
- **OpenCV:** (Open-Source Computer Vision Library) is a widely used open-source software library that provides a comprehensive set of tools and algorithms for real-time computer vision and image processing tasks.
- **Math:** the standard Python math library, providing mathematical functions.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- **Email.message:** Specifically, the `EmailMessage` class from the email.message module, used for constructing email messages.

- **Smtplib:** the standard Python library for sending emails using the Simple Mail Transfer Protocol (SMTP).

- **Datetime:** the standard Python library for manipulating dates and times.

- **Requests:** A popular library for making HTTP requests, often used for interacting with web services and APIs.

3.3. Evaluation Metrics

3.3.1. Confusion Matrix

The confusion matrix is a vital tool for summarizing the performance of classification models. To calculate various evaluation metrics, it is essential to first understand key components such as True Positive (TP), True Negative (TN), False Negative (FN), and False Positive (FP), all of which are derived from the confusion matrix [10] (table 3.1).

| | Actual value | Actual value |
|-----------------|--------------|--------------|
| Predicted value | TP | FN |
| Predicted value | FP | TN |

Table 3.2. The confusion matrix of a classification model

- **True Positive (TP):** Instances where the model accurately predicts the presence of fire in the input image or video.
- **True Negative (TN):** Instances where the model correctly identifies the absence of fire.
- **False Negative (FN):** Cases where the model incorrectly predicts the absence of fire.
- **False Positive (FP):** Cases where the model incorrectly identifies the presence of fire.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

3.3.2. Accuracy

The accuracy metric is the most commonly used and straightforward method for evaluating the performance of a trained deep learning model. It quantifies the proportion of correct predictions made by the model out of the total number of predictions [11] as defined in Equation 1:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.1)$$

3.3.3. Precision

Precision is a crucial evaluation metric that represents the proportion of true positives (correct predictions) to the total number of predicted positives. This metric indicates the accuracy of predicted bounding boxes in relation to the ground truth boxes [12]. The formula for calculating precision is as follows:

$$Precision = \frac{TP}{(TP+FP)} \quad (3.2)$$

3.3.4. Specificity

Specificity is another important evaluation metric used to assess model performance. It indicates the proportion of non-fire instances correctly identified by the model, thereby reflecting the model's ability to avoid false positives [13]. The specificity rate is calculated using the following equation:

$$S = \frac{TN}{TN+FP} \quad (3.3)$$

3.3.5. Recall

Recall, also known as sensitivity, is another essential evaluation metric used in object detection. It measures the proportion of true positives relative to the total number of actual positives in the ground truth [12]. It can be calculated using the following formula:

$$Recall = \frac{TP}{TP+FN} \quad (3.4)$$

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

3.3.6. F1-score

The F1-score is an evaluation metric that considers both Precision and Recall rates. It represents a weighted harmonic mean of Precision and Recall [14] and is calculated using the following equation:

$$F1\ Score = \frac{recall * precision}{recall + precision} \quad (3.5)$$

3.4. The Architecture of The Proposed System

Figure 3.1 shows the architecture of the system.

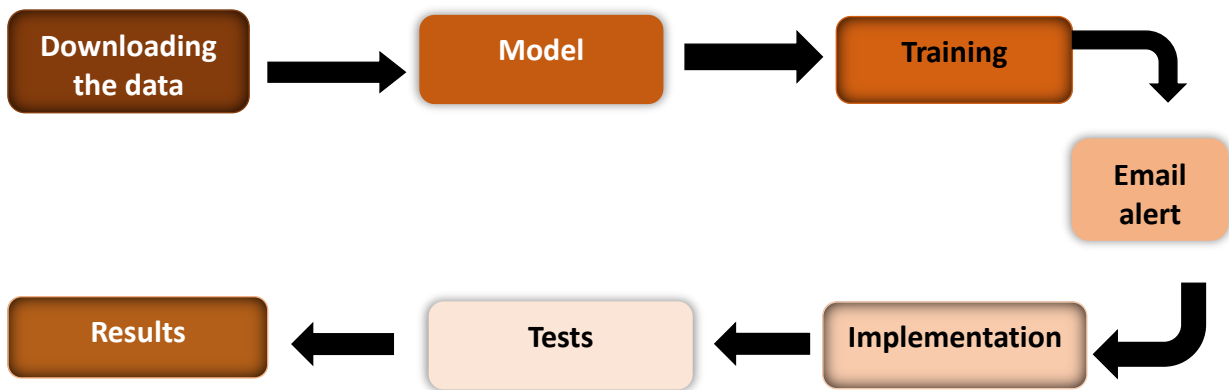


Figure 3.1. The architecture of the system

3.4.1. Downloading the data

The dataset used for training and evaluating the AI fire detection model was sourced from Kaggle and exported via Roboflow.ai on June 13, 2021, at 3:22 PM GMT. The dataset includes 270 images with fire annotated in YOLO v5 PyTorch format.

The following preprocessing was already applied to each image:

- Auto-orientation of pixel data (with EXIF-orientation stripping)
- Resize to 416x416 (Stretch)

Additionally, the following augmentations were applied to create three versions of each source image:

- 50% probability of horizontal flip

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- 50% probability of vertical flip
- Equal probability of one of the following 90-degree rotations: none, clockwise, counter-clockwise
- Randomly crop between 0 and 20 percent of the image
- Random shear of between -15° to $+15^\circ$ horizontally and -15° to $+15^\circ$ vertically
- Random exposure adjustment of between -25 and +25 percent
- Random Gaussian blur of between 0 and 10 pixels
- Salt and pepper noise applied to 5 percent of pixels

Despite extensive searching and inquiries, detailed information about the academic origin of the dataset was not provided. However, it is distributed under the GPL-2 license, which allows for use and distribution but prohibits modifications [9].

3.4.2. Fire detection model

After downloading the data that has already been pre-processed, we can proceed to undertake the development of a computer vision model. This process involves the construction, data input, training, testing, and deployment stages. As machine learning algorithms serve as the foundation upon which we enable deep learning and computer vision models, our initial step entails establishing a framework tailored to the computer vision model. To accomplish this, we will employ the YOLOv8 framework. YOLOv8, short for You Only Look Once version 8 is the latest iteration of the YOLO family of object detection models, known for their speed and accuracy.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

a) YOLOv8 architecture:

YOLOv8 architecture can be broadly divided into three main components [10]:

- **Backbone Network**

The backbone network is the foundation of YOLOv8, responsible for feature extraction from the input image. YOLOv8 employs CSPDarknet53, a variant of Darknet, as its backbone.

The CSPDarknet53 architecture introduces a novel Cross-Stage Partial (CSP) connection, enhancing the information flow between different stages of the network and improving gradient flow during training.

- **Neck and Head Structures**

YOLOv8 introduces a Path Aggregation Network (PANet) as the neck structure. PANet facilitates information flow across different spatial resolutions, enabling the model to capture multi-scale features effectively.

The head structure consists of multiple detection heads, each responsible for predicting bounding boxes, class probabilities, and objectness scores at different scales.

- **Detection Head**

The detection head of YOLOv8 is where the real innovation lies. It utilizes a modified version of the YOLO head, incorporating dynamic anchor assignment and a novel IoU (Intersection over Union) loss function. These improvements contribute to more accurate bounding box predictions and better handling of overlapping objects. Figure 3.2 illustrates the YOLO network architecture.

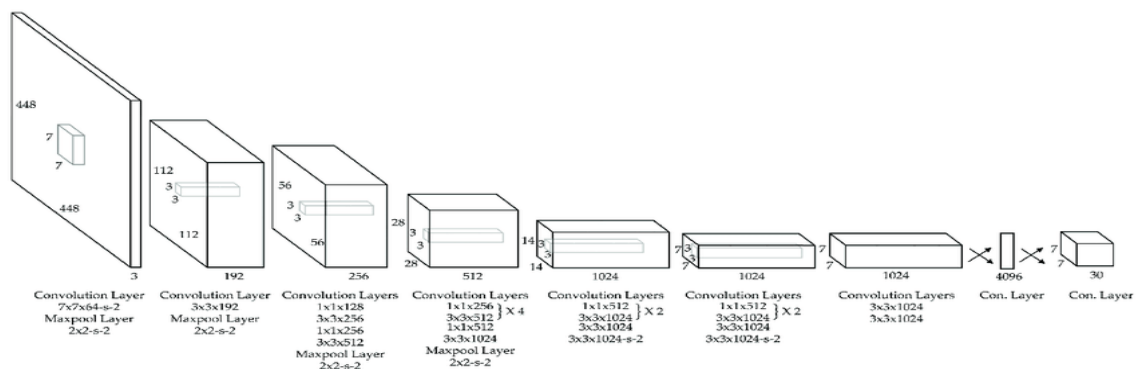


Figure 3.2. YOLO network architecture [10]

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

3.4.3. YOLOv8 network training process

To train our model, data was stored on Google Drive and accessed by mounting the drive to Google Colab. This facilitated easy handling and organization of datasets. Figure 3.3 shows sample frames from the dataset that has been used.

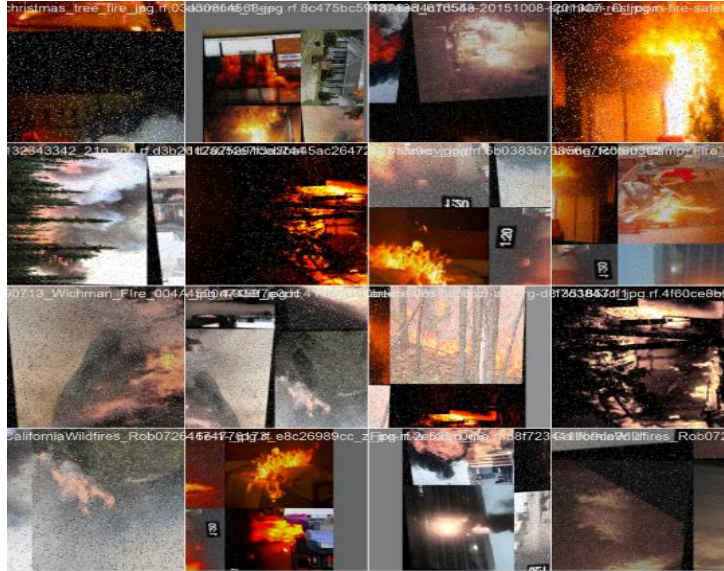


Figure 3.3. Sample frames from the dataset

First, we need to check the status of the NVIDIA GPU available for our session to make sure the GPU is correctly set up and available for use as shown in Figure 3.4.

```
NVIDIA-SMI 535.104.05           Driver Version: 535.104.05   CUDA Version: 12.2
-----
GPU  Name          Persistence-M | Bus-Id      Disp.A | Volatile Uncorr. ECC
Fan  Temp    Perf      Pwr:Usage/Cap |          Memory-Usage | GPU-Util  Compute M.
                                           |                      | GPU-MIG  Compute M.
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
  0  Tesla T4             Off | 00000000:00:04:0 Off |   0%   0
N/A  42C    P8             9W / 70W | 0MiB / 15360MiB |   0%      Default
                                           |                      | N/A

Processes:
GPU  GI  CI      PID  Type  Process name                      GPU Memory
ID  ID  ID                                     Usage
-----+-----+-----+-----+-----+-----+-----+
No running processes found
```

Figure 3.4. GPU information

For the model configuration and training, the YOLOv8 model (`yolov8n.pt`) was selected for its balance between accuracy and computational efficiency.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

The training process was initiated using the 'yolo' command with the following key parameters:

- 'task=detect': Specifies that the task is object detection.
- 'mode=train': Indicates the mode is training.
- 'model=yolov8n.pt': The pre-trained YOLOv8 model is used as the starting point.
- 'data=/content/drive/MyDrive/data2/data.yaml': The path to the dataset configuration file.
- 'epochs=1000': Number of training epochs.
- 'imgsz=640': The image size used for training.

Figures 3.5 and 3.6 illustrate the training process for the first and the last 10 epochs:

| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
|---------|---------|-----------|----------|----------|-----------|---------------------------------------------|
| 1/1000 | 2.38G | 0 | 115.7 | 0 | 0 | 640: 100% 9/9 [00:03<00:00, 2.38it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 1.88it/s] |
| all | 16 | 19 | 0.00313 | 0.789 | 0.0337 | 0.0159 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 2/1000 | 2.16G | 0 | 110.8 | 0 | 0 | 640: 100% 9/9 [00:01<00:00, 4.84it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 3.78it/s] |
| all | 16 | 19 | 0.00229 | 0.579 | 0.003 | 0.000599 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 3/1000 | 2.2G | 0 | 97.32 | 0 | 0 | 640: 100% 9/9 [00:02<00:00, 4.30it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 2.44it/s] |
| all | 16 | 19 | 0.00167 | 0.421 | 0.00485 | 0.00116 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 4/1000 | 2.19G | 0 | 94.51 | 0 | 0 | 640: 100% 9/9 [00:02<00:00, 3.83it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 2.23it/s] |
| all | 16 | 19 | 0.00104 | 0.263 | 0.00171 | 0.000331 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 5/1000 | 2.2G | 0 | 75.89 | 0 | 0 | 640: 100% 9/9 [00:02<00:00, 3.74it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 2.82it/s] |
| all | 16 | 19 | 0.00104 | 0.263 | 0.00106 | 0.00026 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 6/1000 | 2.2G | 0 | 70.65 | 0 | 0 | 640: 100% 9/9 [00:01<00:00, 6.02it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 3.53it/s] |
| all | 16 | 19 | 0.00125 | 0.316 | 0.00376 | 0.000601 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 7/1000 | 2.2G | 0 | 66.22 | 0 | 0 | 640: 100% 9/9 [00:01<00:00, 5.70it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 3.31it/s] |
| all | 16 | 19 | 0.00148 | 0.368 | 0.0014 | 0.000368 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 8/1000 | 2.2G | 0 | 62.38 | 0 | 0 | 640: 100% 9/9 [00:01<00:00, 6.09it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 3.83it/s] |
| all | 16 | 19 | 0.00219 | 0.526 | 0.00776 | 0.00138 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 9/1000 | 2.2G | 0 | 57.88 | 0 | 0 | 640: 100% 9/9 [00:01<00:00, 4.64it/s] |
| Class | Images | Instances | Box(P | R | MAP50 | MAP50-95): 100% 1/1 [00:00<00:00, 2.06it/s] |
| all | 16 | 19 | 0.00208 | 0.526 | 0.00446 | 0.00106 |
| Epoch | GPU_mem | box_loss | cls_loss | dfl_loss | Instances | Size |
| 10/1000 | 2.2G | 0 | 53.86 | 0 | 0 | 640: 100% 9/9 [00:02<00:00, 4.17it/s] |

Figure 3.5. Training progress information for the first 10 epochs

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

| Epoch | GPU_mem | box_loss | cls_loss | dfi_loss | Instances | Size |
|----------|---------|----------|-----------|----------|-----------|---------------------------------------------------|
| 92/1000 | 2.19G | 0 | 0.002811 | 0 | 0 | 640: 100% 9/9 [00:01:00:00, 5.71it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 3.31it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 93/1000 | 2.19G | 0 | 0.002785 | 0 | 0 | 640: 100% 9/9 [00:02:00:00, 4.21it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 3.46it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 94/1000 | 2.2G | 0 | 0.002775 | 0 | 0 | 640: 100% 9/9 [00:02:00:00, 3.72it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 4.60it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 95/1000 | 2.2G | 0 | 0.00272 | 0 | 0 | 640: 100% 9/9 [00:01:00:00, 6.47it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 5.32it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 96/1000 | 2.19G | 0 | 0.002662 | 0 | 0 | 640: 100% 9/9 [00:01:00:00, 5.68it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 4.52it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 97/1000 | 2.19G | 0 | 0.002532 | 0 | 0 | 640: 100% 9/9 [00:01:00:00, 5.80it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 4.44it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 98/1000 | 2.2G | 0 | 0.002484 | 0 | 0 | 640: 100% 9/9 [00:01:00:00, 5.21it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 4.40it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 99/1000 | 2.19G | 0 | 0.002275 | 0 | 0 | 640: 100% 9/9 [00:02:00:00, 3.76it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 3.16it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 100/1000 | 2.19G | 0 | 0.002002 | 0 | 0 | 640: 100% 9/9 [00:02:00:00, 3.52it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 3.77it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |
| 101/1000 | 2.19G | 0 | 0.001646 | 0 | 0 | 640: 100% 9/9 [00:01:00:00, 6.45it/s] |
| | Class | Images | Instances | Box(P | R | mAP50 mAP50-95): 100% 1/1 [00:00:00:00, 5.14it/s] |
| | all | 16 | 19 | 0.0333 | 0.0526 | 0.0432 0.0113 |

Figure 3.6. Training progress information for the last 10 epochs

3.4.4. Email Alert System Using SMTP

To implement the email alert system in this project, Python's `smtplib` library was utilized to send notifications via the Simple Mail Transfer Protocol (SMTP).

SMTP is a communication protocol used for sending emails across the Internet, designed to send emails from the sender's server to the recipient's server reliably and efficiently as explained in Figure 3.7 which demonstrates the SMTP architecture.

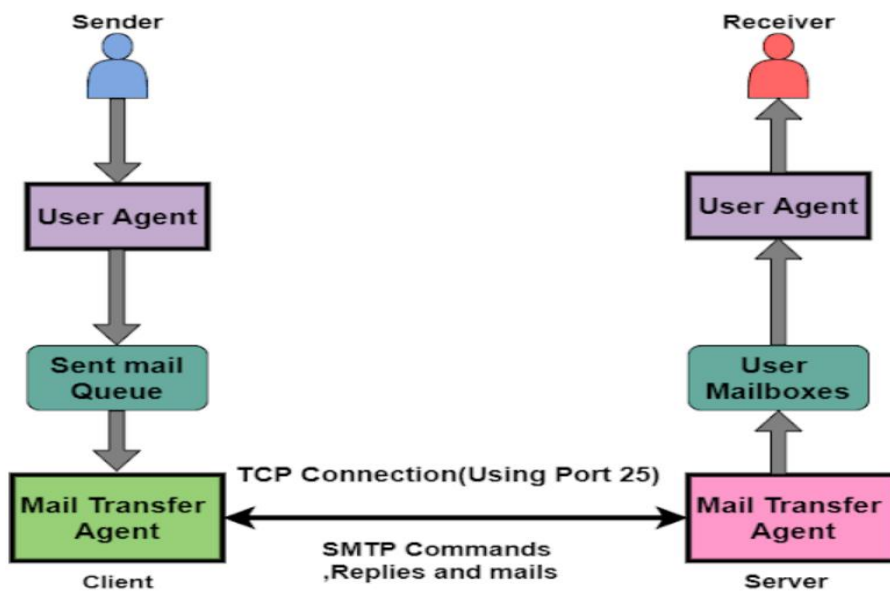


Figure 3.7. SMTP architecture [15]

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

The process involved the following steps:

- 1. SMTP Server Connection:** Connecting to the SMTP server using the email service provider's details.
- 2. Email Composition:** Creating the email content, including the subject, sender, and recipient details, using the `MIMEText` class.
- 3. Email Transmission:** Sending the email via the SMTP server, ensuring secure transmission with the STARTTLS command and authenticating using the sender's email credentials.

This approach ensured real-time and reliable email notifications, enhancing the responsiveness and reliability of the alert system within the project.

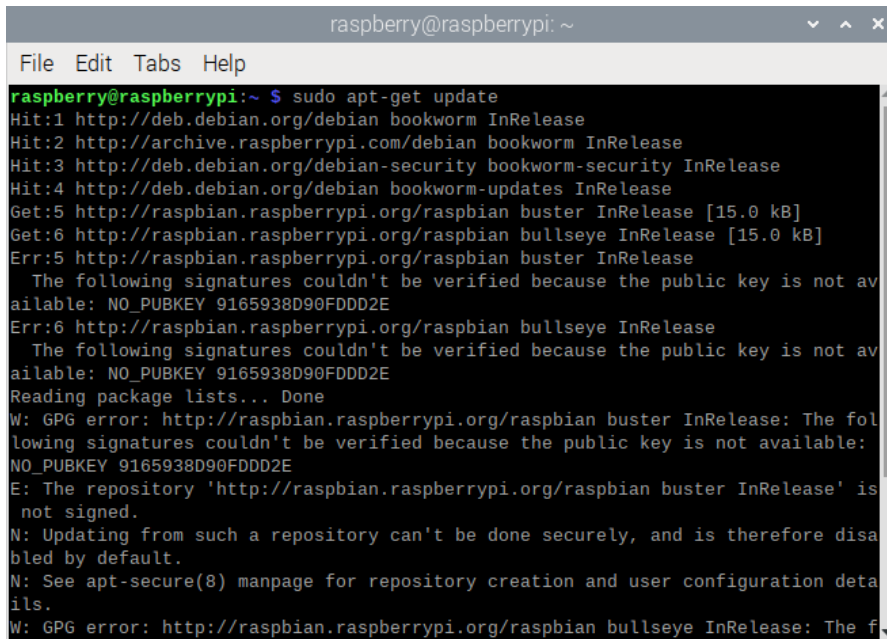
3.4.5. Implementation

After successfully testing the system on a PC, we implemented it on a Raspberry Pi board. This process involved configuring the Raspberry Pi and transferring the data and the trained model to the device.

The following steps were followed to implement the system on the Raspberry Pi:

- **Configuring the Raspberry Pi:** Initially, we installed the latest version of the Raspbian OS on the Raspberry Pi, we configured the Wi-Fi settings to ensure network connectivity, and we executed system updates to ensure all packages were current (Figure 3.8).

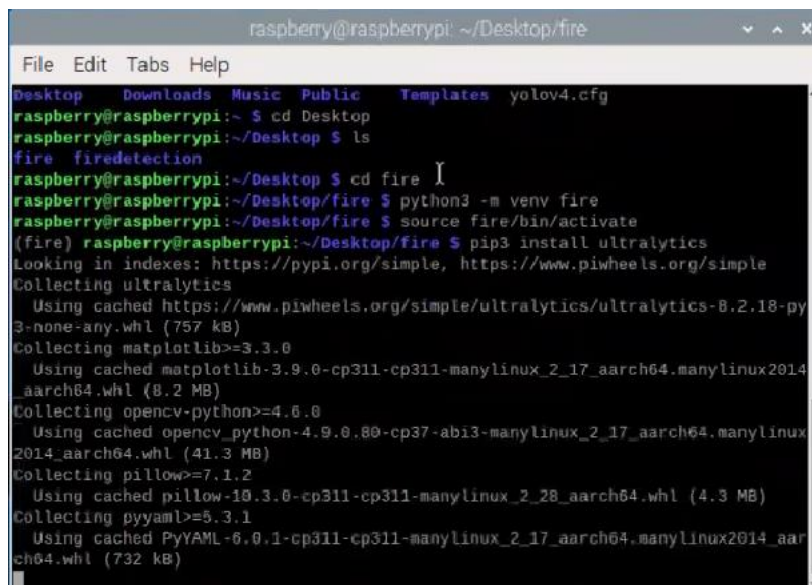
Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO



```
raspberrypi@raspberrypi: ~  
File Edit Tabs Help  
raspberrypi@raspberrypi:~ $ sudo apt-get update  
Hit:1 http://deb.debian.org/debian bookworm InRelease  
Hit:2 http://archive.raspberrypi.com/debian bookworm InRelease  
Hit:3 http://deb.debian.org/debian-security bookworm-security InRelease  
Hit:4 http://deb.debian.org/debian bookworm-updates InRelease  
Get:5 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]  
Get:6 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]  
Err:5 http://raspbian.raspberrypi.org/raspbian buster InRelease  
The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 9165938D90FDD2E  
Err:6 http://raspbian.raspberrypi.org/raspbian bullseye InRelease  
The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 9165938D90FDD2E  
Reading package lists... Done  
W: GPG error: http://raspbian.raspberrypi.org/raspbian buster InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 9165938D90FDD2E  
E: The repository 'http://raspbian.raspberrypi.org/raspbian buster InRelease' is not signed.  
N: Updating from such a repository can't be done securely, and is therefore disabled by default.  
N: See apt-secure(8) manpage for repository creation and user configuration details.  
W: GPG error: http://raspbian.raspberrypi.org/raspbian bullseye InRelease: The f
```

Figure 3.8. Installing updates on raspberry pi

- **Transferring data and the trained model:** subsequently, we have transferred the data and the trained model (the weights folder) using a USB key.
- **Creating a virtual environment:** for this step, we used `venv` on the Raspberry Pi command line, and conducted all subsequent work within this environment to ensure isolation and dependency management (figure 3.9).

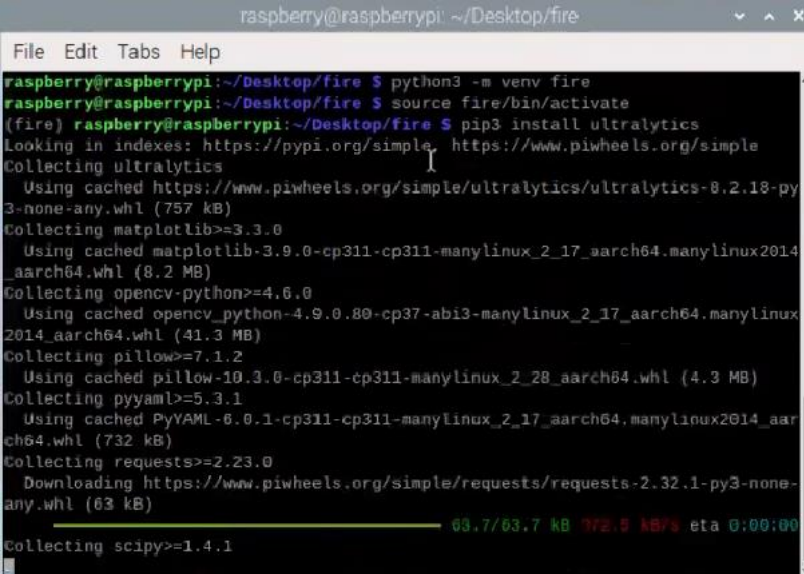


```
raspberrypi@raspberrypi: ~/Desktop/fire  
File Edit Tabs Help  
Desktop Downloads Music Public Templates yolov4.cfg  
raspberrypi@raspberrypi:~ $ cd Desktop  
raspberrypi@raspberrypi:~/Desktop $ ls  
fire fire-detection  
raspberrypi@raspberrypi:~/Desktop $ cd fire  
raspberrypi@raspberrypi:~/Desktop/fire $ python3 -m venv fire  
raspberrypi@raspberrypi:~/Desktop/fire $ source fire/bin/activate  
(fire) raspberrypi@raspberrypi:~/Desktop/fire $ pip3 install ultralytics  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting ultralytics  
Using cached https://www.piwheels.org/simple/ultralytics/ultralytics-8.2.18-py3-none-any.whl (757 kB)  
Collecting matplotlib<=3.3.0  
Using cached matplotlib-3.9.0-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (8.2 MB)  
Collecting opencv-python<=4.6.0  
Using cached opencv_python-4.9.0.80-cp37-abi3-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (41.3 MB)  
Collecting pillow<=7.1.2  
Using cached pillow-10.3.0-cp311-cp311-manylinux_2_28_aarch64.whl (4.3 MB)  
Collecting pyyaml<=5.3.1  
Using cached PYAML-6.0.1-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (732 kB)
```

Figure 3.9. Creating and activating a virtual environment on Raspberry Pi

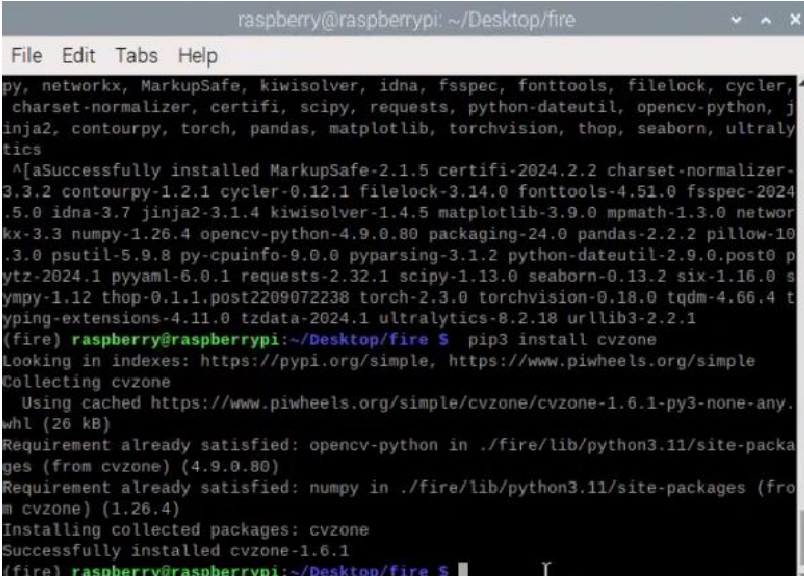
Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

- **Installing Required Libraries:** we installed the necessary Python libraries (e.g., OpenCV, ultralytics library) (figures 3.10 and 3.11).



```
raspberrypi@raspberrypi: ~/Desktop/fire
File Edit Tabs Help
raspberrypi@raspberrypi:~/Desktop/fire $ python3 -m venv fire
raspberrypi@raspberrypi:~/Desktop/fire $ source fire/bin/activate
(fire) raspberrypi@raspberrypi:~/Desktop/fire $ pip3 install ultralytics
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting ultralytics
  Using cached https://www.piwheels.org/simple/ultralytics/ultralytics-0.2.18-py3-none-any.whl (757 kB)
Collecting matplotlib>=3.3.0
  Using cached matplotlib-3.9.0-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (8.2 MB)
Collecting opencv-python>=4.6.0
  Using cached opencv-python-4.9.0.80-cp37-abi3-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (41.3 MB)
Collecting pillow>=7.1.2
  Using cached pillow-10.3.0-cp311-cp311-manylinux_2_28_aarch64.whl (4.3 MB)
Collecting pyyaml>=5.3.1
  Using cached PyYAML-6.0.1-cp311-cp311-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (732 kB)
Collecting requests>=2.23.0
  Downloading https://www.piwheels.org/simple/requests/requests-2.32.1-py3-none-any.whl (63 kB)
  03.7/03.7 kB 972.9 kB/s eta 0:00:00
Collecting scipy>=1.4.1
```

Figure 3.10. Installing ultralytics on Raspberry pi



```
py, networkx, MarkupSafe, kiwisolver, idna, fsspec, fonttools, filelock, cycler, charset-normalizer, certifi, scipy, requests, python-dateutil, opencv-python, jinja2, contourpy, torch, pandas, matplotlib, torchvision, thop, seaborn, ultralytics
^[[aSuccessfully installed MarkupSafe-2.1.5 certifi-2024.2.2 charset-normalizer-3.3.2 contourpy-1.2.1 cycler-0.12.1 filelock-3.14.0 fonttools-4.51.0 fsspec-2024.5.0 idna-3.7 jinja2-3.1.4 kiwisolver-1.4.5 matplotlib-3.9.0 mpmath-1.3.0 networkx-3.3 numpy-1.26.4 opencv-python-4.9.0.80 packaging-24.0 pandas-2.2.2 pillow-10.3.0 psutil-5.9.8 py-cpuinfo-9.0.0 pyparsing-3.1.2 python-dateutil-2.9.0.post0 pytz-2024.1 pyyaml-6.0.1 requests-2.32.1 scipy-1.13.0 seaborn-0.13.2 six-1.16.0 sympy-1.12 thop-0.1.1.post2209072238 torch-2.3.0 torchvision-0.18.0 tqdm-4.66.4 typing-extensions-4.11.0 tzdata-2024.1 ultralytics-0.2.18 urllib3-2.2.1
(fire) raspberrypi@raspberrypi:~/Desktop/fire $ pip3 install cvzone
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting cvzone
  Using cached https://www.piwheels.org/simple/cvzone/cvzone-1.6.1-py3-none-any.whl (26 kB)
Requirement already satisfied: opencv-python in ./fire/lib/python3.11/site-packages (from cvzone) (4.9.0.80)
Requirement already satisfied: numpy in ./fire/lib/python3.11/site-packages (from cvzone) (1.26.4)
Installing collected packages: cvzone
Successfully installed cvzone-1.6.1
(fire) raspberrypi@raspberrypi:~/Desktop/fire $
```

Figure 3.11. Installing cv zone on Raspberry Pi

- **Setting up the execution environment:** finally, we configured paths and dependencies and verified model compatibility and made adjustments to execute the python code on python 3 IDE.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

By following these steps, the fire detection system was successfully deployed on the Raspberry Pi, enabling real-time fire detection and alert notifications in a compact, portable setup.

After successfully implementing the system on the raspberry pi, we have added a GPS module to include the location of the fire in the email alert system, for this we have connected a Bn220 GPS module to the raspberry Pi as shows figure 6.12.

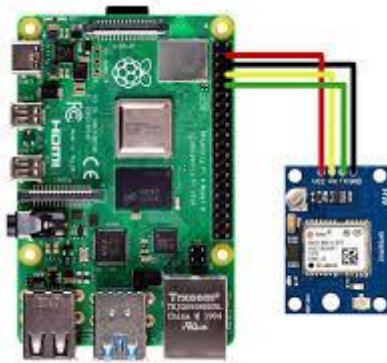


Figure 6.12. GPS and Raspberry Pi wiring diagram

After completing the wiring, we verified the GPS activation by executing “sudo service gpssd status.” Subsequently, we initiated the GPS by running the Python script located in the GPS libraries folder. We then read the data received from the GPS module on the Raspberry Pi, including altitude, longitude, and other relevant information, to utilize it later in the fire detection system's Python code. Figures 3.13, 3.14 and 3.15 illustrate the aforementioned steps performed with the GPS module.

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

```
raspberrypi@raspberrypi: ~  
File Edit Tabs Help  
● gpsd.service - GPS (Global Positioning System) Daemon  
  Loaded: loaded (/lib/systemd/system/gpsd.service; enabled; preset: enabled)  
  Active: active (running) since Fri 2024-06-21 00:21:50 BST; 4min 3s ago  
TriggeredBy: ● gpsd.socket  
  Process: 479 ExecStart=/usr/sbin/gpsd $GPSD_OPTIONS $OPTIONS $DEVICES (code>  
  Main PID: 507 (gpsd)  
    Tasks: 2 (limit: 755)  
     CPU: 597ms  
   CGroup: /system.slice/gpsd.service  
           └─507 /usr/sbin/gpsd -n /dev/ttyS0  
  
Jun 21 00:21:49 raspberrypi systemd[1]: Starting gpsd.service - GPS (Global Pos>  
Jun 21 00:21:50 raspberrypi systemd[1]: Started gpsd.service - GPS (Global Pos>  
  
lines 1-13/13 (END)
```

Figure 3.13. Verifying GPS status

```
raspberrypi@raspberrypi: ~  
File Edit Format Run Options Window Help  
#import the necessary libraries  
from gps import *  
import time  
from datetime import datetime  
import pytz, datetime.parser  
  
running = True  
  
def getPositionData(gps):  
    nx = gpsd.next()  
  
    if nx['class'] == 'TDV':  
        abstract gps info  
        latitude = getattr(nx, 'lat', 'Unknown')  
        longitude = getattr(nx, 'lon', 'Unknown')  
        speed = getattr(nx, 'speed', 'Unknown')  
  
    #change directory to wherever you want to put file to be locate  
    with open('/home/pi/Desktop/GPS/newoutput.txt', 'a') as file:  
        # Move read cursor to the start of file.  
        file.seek(0)  
        # If file is not empty then append '\n'  
        data = file.read(200)  
        if len(data) > 0:  
            file.write("\n")  
            #stores the current date/time  
            dateLine=datetime.now()  
            info="Time: " + str(dateLine) + " Position: lon = " + at  
            print (info)  
            #append gps info at the end of the file  
            file.write(info)  
  
    # Tell gpsd we are ready to receive messages  
    gpsd = gps(mode=WATCH_ENABLE|WATCH_NEWSTYLE)  
  
try:  
    print ("Application started")  
    while running:  
        #call function to extract and append GPS data  
        getPositionData(gps)  
        #delay running the program for 1 second  
        time.sleep(1)  
  
#If the user presses ctrl+c, the program will stop running  
except (KeyboardInterrupt):  
    running = False
```

```
Python 3.11.2 (main, Mar 13 2023, 12:18:28) [GCC 11.2.0] on Linux  
Type "help()", "copyright()", "credits()" or "license()" for more information.  
Application started  
===== RESTART: /tmp/xx-SKWP2/gps_info.py =====  
Application started
```

```
raspberrypi@raspberrypi: ~  
File Edit Tabs Help  
create a virtual environment using python3 -m venv path/to/venv.  
Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make  
sure you have python3-full installed.  
  
For more information visit http://rpit.liu/venv  
  
note: If you believe this is a mistake, please contact your Python installation  
OS distribution provider. You can override this, at the risk of breaking your P  
thon installation or OS, by passing --break-system-packages.  
note: See PEP 585 for the detailed specification.  
raspberrypi@raspberrypi: ~$ cd Desktop  
raspberrypi@raspberrypi: ~/Desktop$ cd fire  
raspberrypi@raspberrypi: ~/Desktop/fire$ source bin/fire/activate  
bash: bin/fire/activate: No such file or directory  
raspberrypi@raspberrypi: ~/Desktop/fire$ source fire/bin/activate  
(fire) raspberrypi@raspberrypi: ~/Desktop/fire$ pip install gps  
Looking in indexes: https://pypi.org/simple, https://ow.python.org/simple  
Requirement already satisfied: gps in /fire/lib/python3.11/site-packages (3.19)  
(fire) raspberrypi@raspberrypi: ~/Desktop/fire$ pip install pytz  
Looking in indexes: https://pypi.org/simple, https://ow.python.org/simple  
Requirement already satisfied: pytz in /fire/lib/python3.11/site-packages (2024  
(fire) raspberrypi@raspberrypi: ~/Desktop/fire$ scrot
```

Figure 3.14. Activating the GPS application

Chapter 3: Design and Implementation of Forest Fire Detection Model Using YOLO

```
Time:                2024-05-23T23:20:12.000Z (18)
Latitude:           36.52711710 N
Longitude:          2.98965530 E
Alt (HAE, MSL):    432.244, 288.737 ft
Speed:              1.15 mph
Track (true, var):  182.1, 1.2 deg
Climb:              84.06 ft/min
Status:             3D DGPS FIX (10 secs)
Long Err (XDOP, EPX): 1.11, +/- 13.6 ft
Lat Err (YDOP, EPY): 1.02, +/- 12.5 ft
Alt Err (VDOP, EPV): 13.04, +/- 63.0 ft
2D Err (HDOP, CEP): 7.10, +/- 94.9 ft
3D Err (PDOP, SEP): 14.85, +/- 47.6 ft
Time Err (TDOP):    7.20
Geo Err (GDOP):     16.50
Speed Err (EPS):    +/- 3.4 mph
Track Err (EPD):    +/- 191 deg
Time offset:        0.109871448 s
Grid Square:        JM167m86
```

Figure 3.15. Data received by the GPS module on Raspberry Pi

Once all preliminary work was completed, we connected a USB camera module to the Raspberry Pi. Initial fire detection tests were conducted on pre-downloaded videos. Following the connection of the external camera module, further tests were conducted using live footage.

3.5. Conclusion

This chapter detailed the design and implementation of a forest fire detection model using YOLO. We established the working environment, specifying the required hardware, software, and libraries. We then defined evaluation metrics to assess model performance and provided an overview of the system architecture, including data preparation, the fire detection model, the YOLOv8 training process, and the email alert system.

The implementation section covered practical steps such as verifying GPS functionality, connecting a USB camera module, and conducting fire detection tests on both pre-recorded videos and live footage.

Leveraging AI, deep learning, and edge computing, our model offers a robust solution for real-time forest fire detection, enhancing accuracy and speed. This scalable system can be integrated with UAV platforms for effective environmental monitoring and disaster prevention.

Chapter 4

Results and Discussions

4.1. Introduction

In this final chapter, we present the results and discussions of our trained forest fire detection model using the YOLO framework. This chapter aims to provide a comprehensive analysis of the model's performance, showcasing key metrics such as accuracy, loss, recall, and the confusion matrix. By examining these results, we aim to evaluate the effectiveness of the model in accurately detecting forest fires. Additionally, we will discuss the implications of these findings, potential limitations, and areas for future improvement. This analysis is crucial in understanding the model's strengths and weaknesses, thereby guiding further development and optimization efforts for enhanced real-time fire detection capabilities.

4.2. Tests and Results

4.2.1. Confusion Matrix

Figure 4.1 illustrates the confusion matrix graph of the trained model.

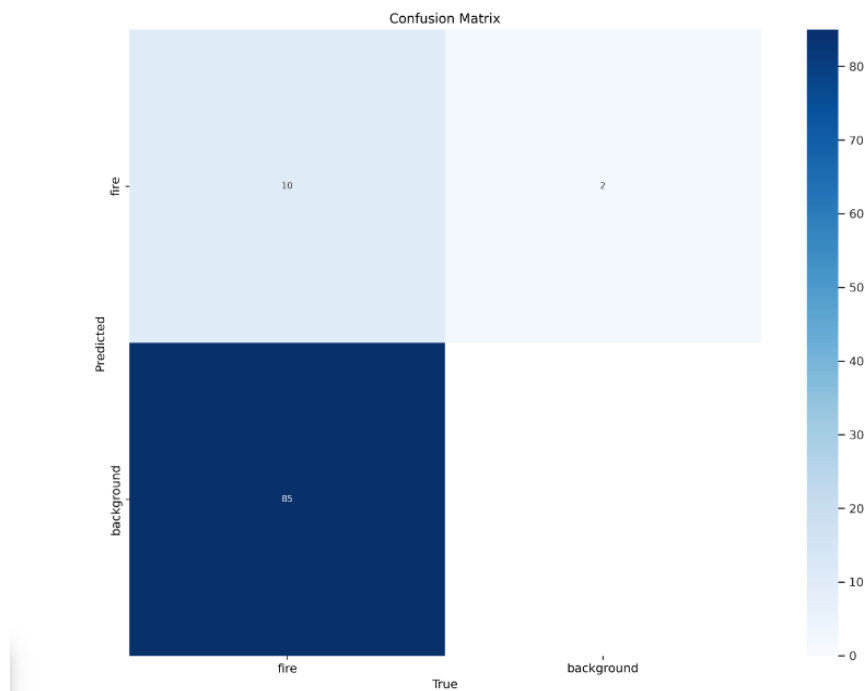


Figure 4.1. Confusion matrix result

The analysis of the confusion matrix reveals that the model correctly identified 10 instances of the positive class (TP = 10), but did not correctly classify any instances of the negative class (TN = 0). There were 2 instances where the model incorrectly classified the negative class as positive (FP = 2) and 85 instances where the model misclassified the positive class as negative (FN = 85).

These results indicate a notable bias towards predicting the positive class, potentially influenced by an imbalanced dataset or inadequate training data for the negative class. The high number of false negatives (85) and the absence of true negatives suggest that the model struggles significantly with identifying the negative class. This imbalance in the confusion matrix implies that the model is more inclined to predict the positive class, but with a considerable error rate in both positive and negative predictions.

From the values given by the confusion matrix, we can calculate the accuracy, recall, precision, and F1 score.

- True Positives (TP) = 10
- True Negatives (TN) = 0
- False Positives (FP) = 2
- False Negatives (FN) = 85

4.2.2. Accuracy

Accuracy is the ratio of correctly predicted observations to the total observations, we can calculate it using equation 3.1:

$$accuracy = \frac{10+0}{10+0+2+85} = 0.103$$

The accuracy is 0.103 (or 10.3%).

4.2.3. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives, we can calculate it using equation 3.2:

$$precision = \frac{10}{10+2} = 0.833$$

The precision is 0.833 or (83.3%).

4.2.4. Recall

Recall is the ratio of correctly predicted positive observations to all observations in the actual class, we can calculate it using equation 3.5:

$$recall = \frac{10}{10+85} = 0.105$$

The recall is 0.105 or (10.5%).

4.2.5. F1 Score

The F1 score is the harmonic mean of precision and recall, we can calculate it using equation 3.5:

$$F1\ score = 2 * \frac{0.833*0.105}{0.833+0.105} = 0.186$$

F1 score is 0.186 or (18.6%).

4.3. Discussions

The results of the classification model highlight several areas of success and opportunities for improvement. The model achieved a high precision of 83.3%, demonstrating its strength in accurately identifying positive instances and minimizing false positives.

However, the model's accuracy of 10.3% and recall of 10.5% indicate room for enhancement in correctly identifying all relevant instances. The F1 score of 18.7% underscores the potential to balance precision and recall further.

4.3.1. Results Interpretation

From the confusion matrix values, we derived:

- Accuracy: 10.3%
- Precision: 83.3%
- Recall: 10.5%
- F1 Score: 18.7%

The high precision indicates the model's proficiency in correctly identifying positive instances, effectively minimizing false positives. This strength is crucial for applications where the cost of false positives is high, showcasing the model's capability in making accurate positive predictions.

However, the accuracy of 10.3% and recall of 10.5% highlight critical areas for enhancement. The modest accuracy suggests that the model needs improvement in correctly classifying all instances. The recall rate points to the model's current challenge in identifying all positive instances, which is essential for a balanced and effective classification system.

The F1 score of 18.7% reflects the need to improve the balance between precision and recall. Enhancing this balance will lead to a more robust model capable of delivering consistent performance across different scenarios.

4.3.2. Proposed Improvements

Addressing these issues requires targeted improvements, primarily focusing on data augmentation and collection:

- **Enhanced Data Collection:** Future work should prioritize collecting more images of fire instances. This includes sourcing images from diverse environments and conditions to ensure the model can generalize well across different scenarios.
- **Data Augmentation:** Applying data augmentation techniques can artificially increase the diversity and volume of the dataset. Techniques such as rotation, flipping, scaling, and colour adjustment can help in creating a more robust training set.

- **Balanced Dataset:** Ensuring a balanced dataset with an adequate number of positive and negative instances will help the model learn more effectively and improve its ability to detect fire instances accurately.

Despite the current limitations, the model provides a strong foundation for developing an effective classification system. The results underscore the importance of a balanced and comprehensive dataset. The high precision achieved in identifying positive instances is promising, indicating the model's potential once provided with a more balanced training dataset. Future work will focus on augmenting the dataset, refining the model architecture, and exploring advanced training techniques to enhance performance.

4.4. Testing The Model on the PC

We conducted multiple tests on the model using a PC, beginning with evaluations on two MP4 videos and subsequently on the PC's webcam.

4.4.1. Tests on videos

a) tests on the first video

Figures 4.2 and 4.3 display the model's performance on the PC during the tests conducted using the first video.

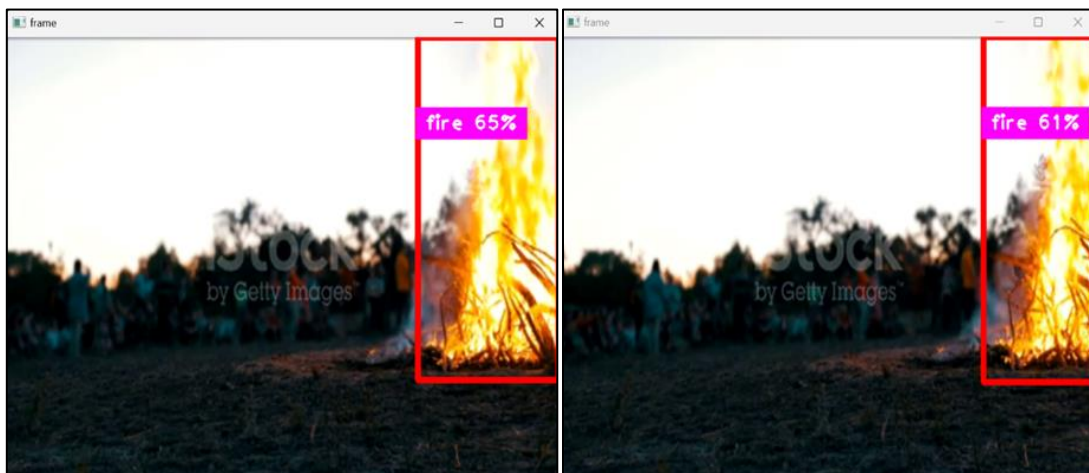


Figure 4.2. Test on video 65%

figure 4.3. Test on video 61%

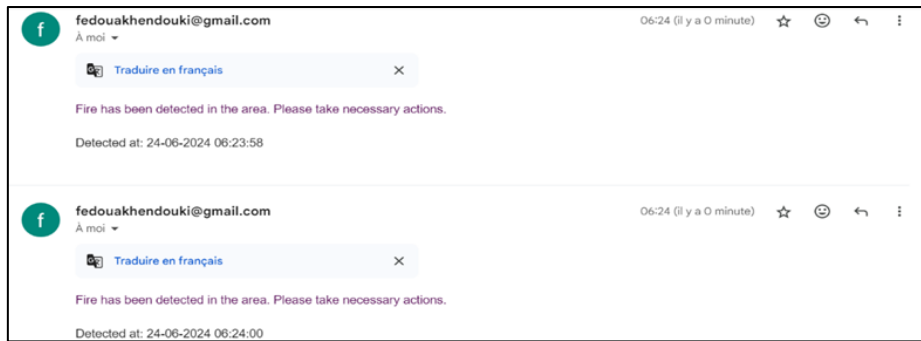


Figure 4.4. Email alert for the first video's tests

b) tests on the second video

Figures 4.5 and 4.6 display the model's performance on the PC during the tests conducted using the second video.

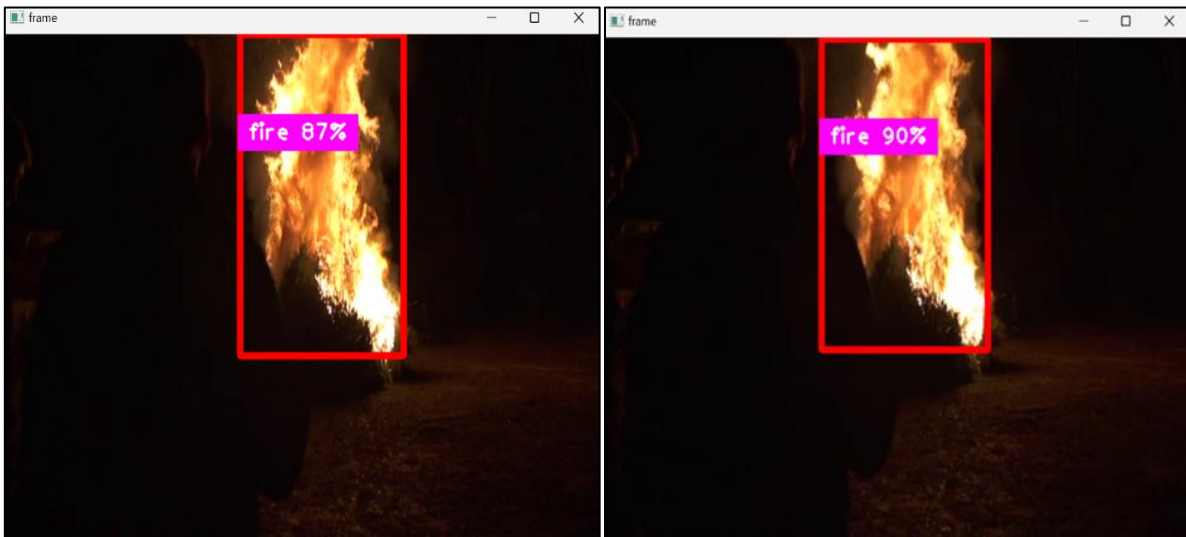


Figure 4.5. Test on video 87%

Figure 4.6. Test on video 90%

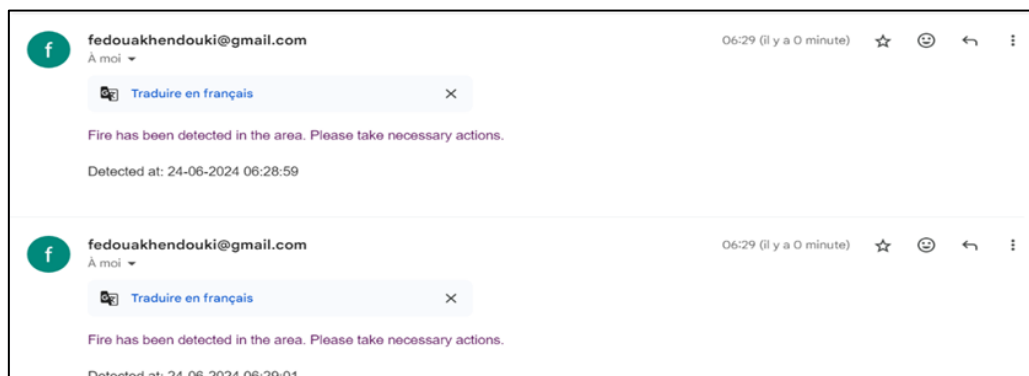


Figure 4.7. Email alert for the second video's tests

As shown, the model successfully detects fire in both videos and indicates the fire's intensity with each detection. Figures 4.4 and 4.7 illustrate the email alerts sent each time the model detects fire.

4.4.2. tests on the PC's webcam

Figures 4.8 and 4.9 display the model's performance on the PC's webcam.



Figure 4.8. Test on webcam 56%

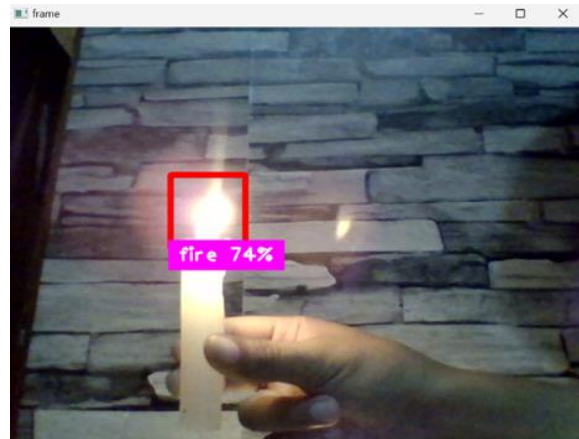


Figure 4.9. Test on webcam 74%

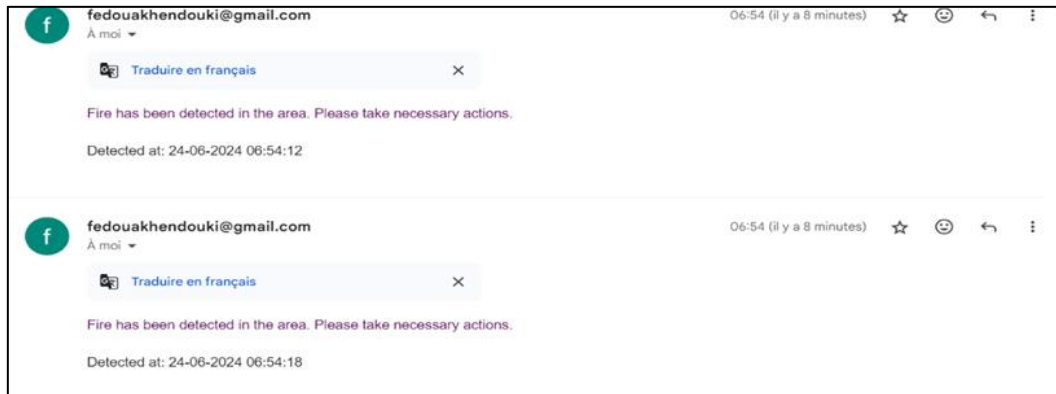


Figure 4.10. Email alert for webcam's tests

The model also detects fire from the candles placed next to the webcam and successfully sends email alerts for each detection, as illustrated in Figure 4.10.

4.5. Testing The Model on Raspberry Pi

4.5.1. Tests on video

Figures 4.11 and 4.12 display the test results on the video and the email alert sent upon fire detection.

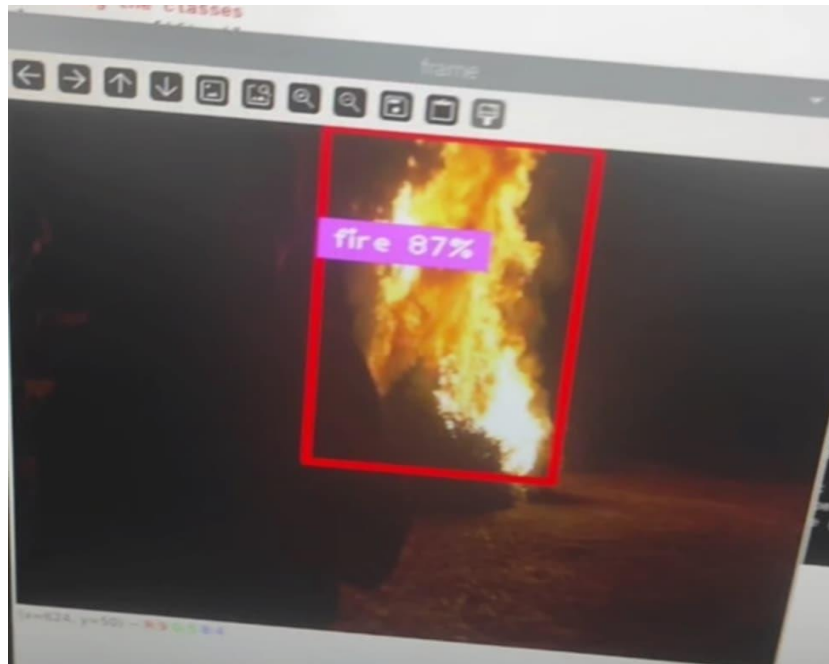


Figure 4.11. Test on video 87%

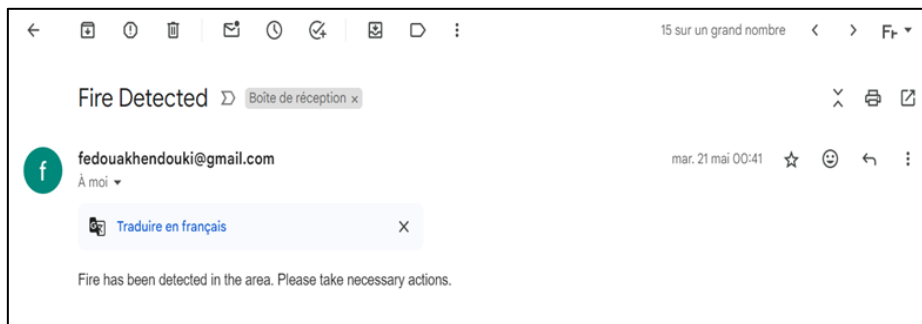


Figure 4.12. Email alert for video

4.5.2. Tests on The Logitech External Camera

In this step, we tested the system after integrating all components, including the external camera and GPS, with the Raspberry Pi. Figures 4.13 and 4.14 illustrate the tests conducted using the external camera connected to the Raspberry Pi.

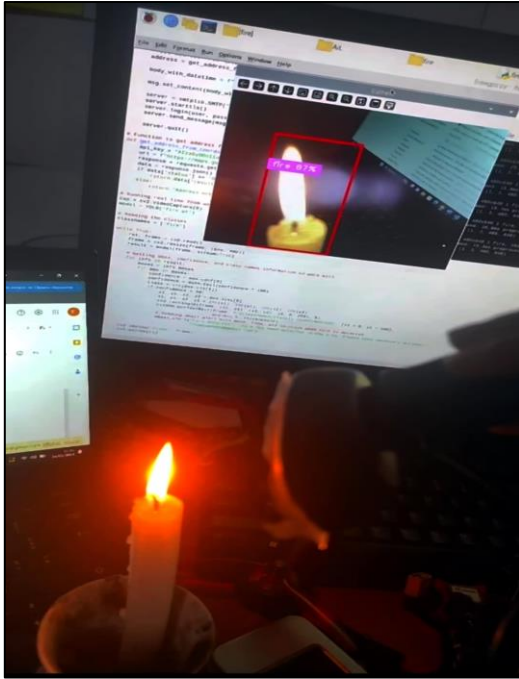


Figure 4.13. Test on external webcam

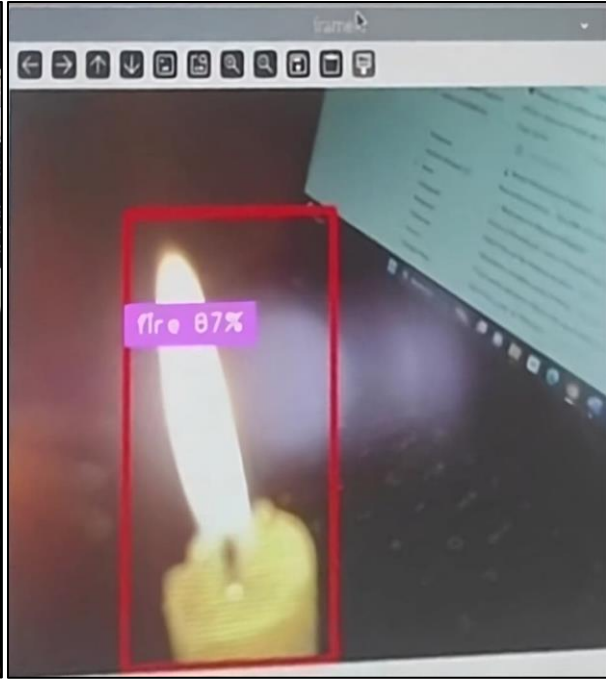


Figure 4.14. Test on external webcam 87%

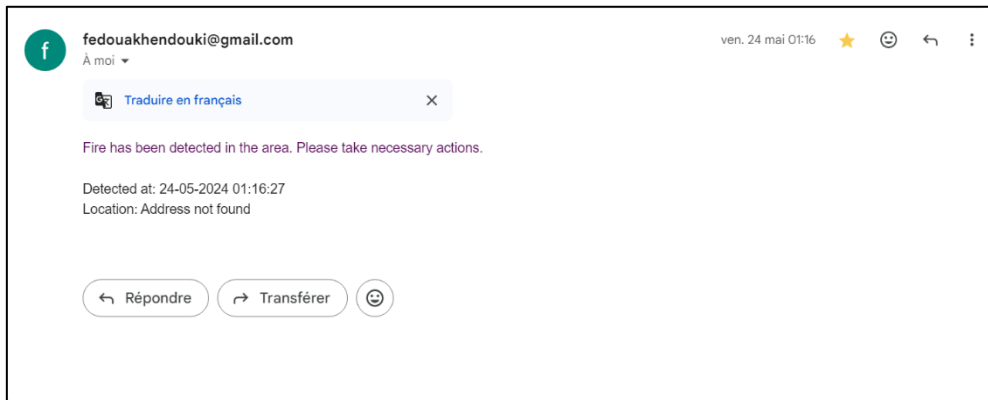


Figure 4.15. Email alert for the external webcam

The system effectively detects fire and sends email alerts upon detection. Figure 4.15 demonstrates an email alert generated when the system identified fire using the Logitech webcam. The alert includes the date and time of the fire detection. Although the alert is designed to report the fire's location, it states "Location: Address not found" in this instance. This indicates that the GPS could not acquire sufficient data to determine the exact location of the fire. This limitation arose because the tests were conducted indoors, where GPS signal reception is inadequate. Future tests will be conducted in various outdoor environments to improve location accuracy and achieve more reliable results.

4.6. Testing The System on Drone Deployment

After conducting all necessary tests to ensure the system detects fire successfully on both PC and Raspberry Pi using videos and webcams, we deployed the system on the drone, as shown in figure 4.16:



Figure 4.16. System's deployment on the drone

We connected the 12V LiPo battery (the drone's power source) to the Raspberry Pi. Specifically, we used a 5V amplifier to connect two cells (4.12V) from the battery to the Raspberry Pi, ensuring protection for the Raspberry Pi.

After deploying the system on the drone, we conducted tests at a height of 1.65 meters. Figures 4.17, 4.18, and 4.19 show the test results of the system on the drone.

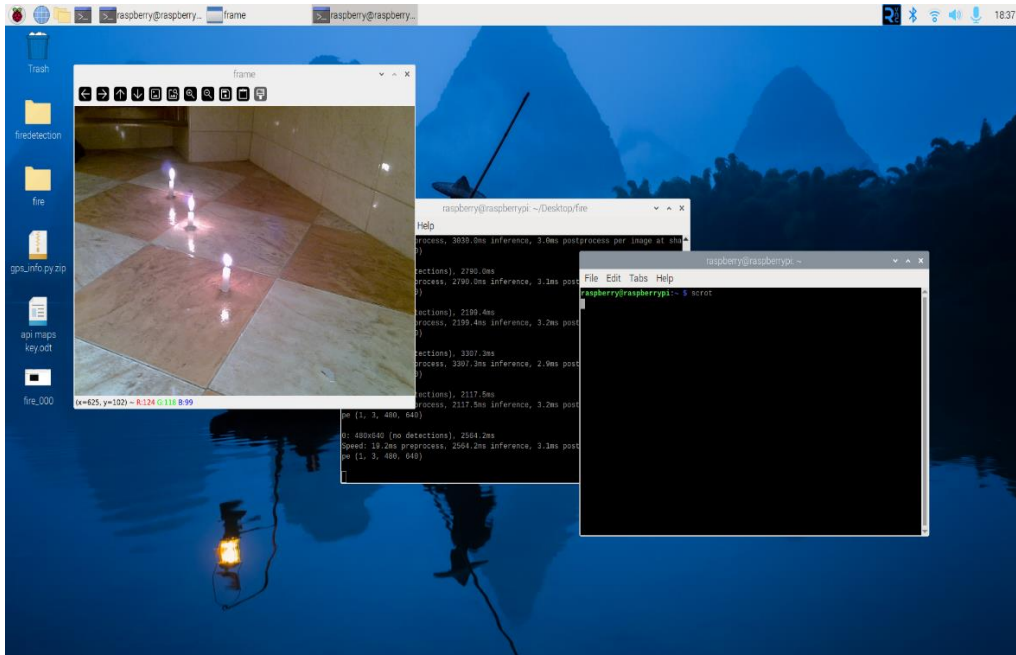


Figure 4.17. testing the system on the drone

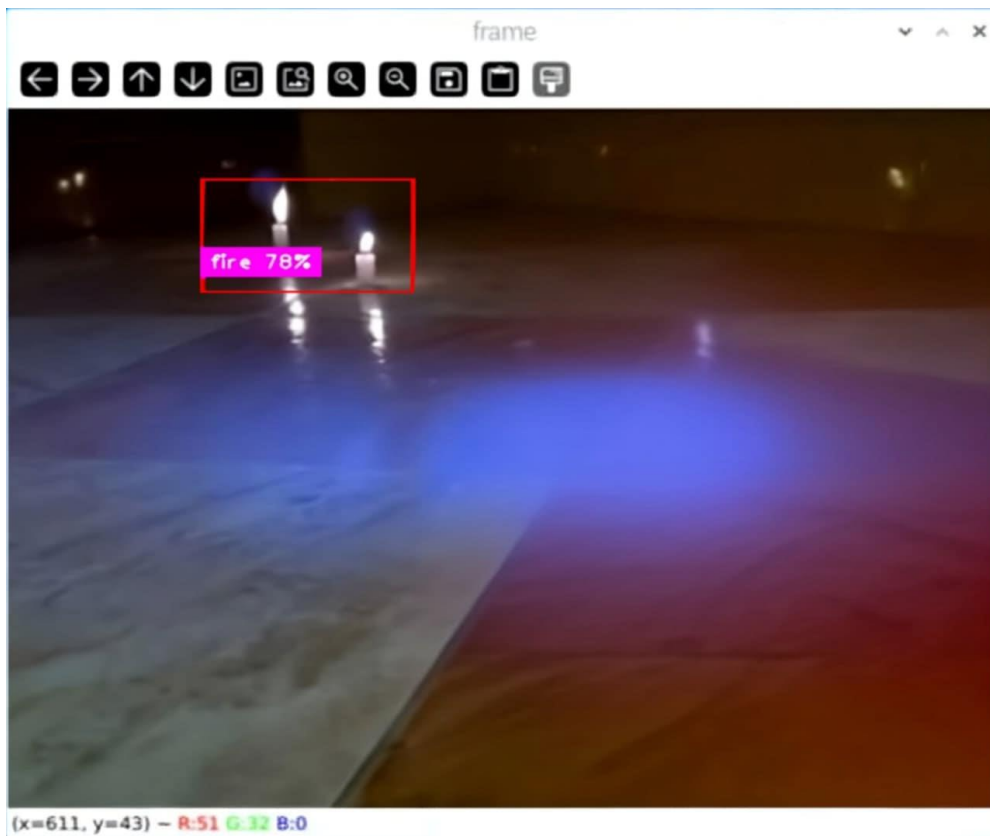


Figure 4.18. drone's test's result

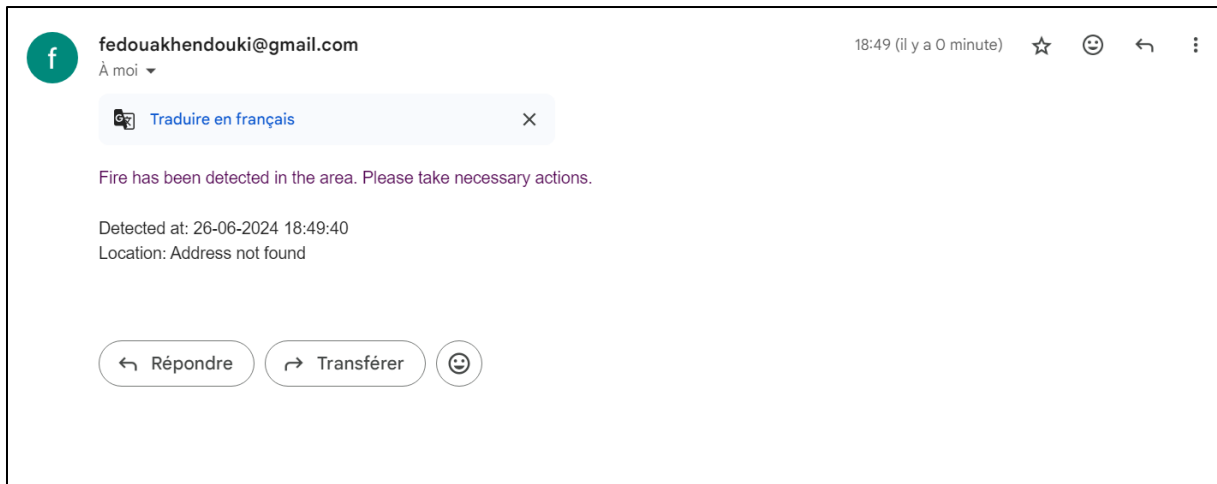


Figure 4.19. Email alert for the drone's test

The system successfully detected fire and sent the email alert; however, due to the model's modest accuracy and recall, fire detection from long distances was challenging while the drone was in motion. To address this, we stabilized the drone and reduced the distance slightly, ensuring the model could still detect fire, which it did successfully. Due to limited funds and time, we could not perform outdoor tests with the drone at heights exceeding 1.70 meters. Future tests will be conducted outdoors, with the model improved as proposed in section 4.3.2.

4.7. Conclusion

This chapter presented the results and discussions of our trained classification model. Despite achieving high precision in identifying positive instances, the model demonstrated limitations in overall accuracy and recall. The analysis of the confusion matrix revealed a bias towards predicting the positive class, indicating challenges primarily stemming from dataset imbalance. To address these limitations, future work should prioritize dataset augmentation, algorithmic refinement, and extensive testing to enhance the model's effectiveness in real-world classification scenarios.

In response to the escalating wildfire threat in Algeria's mountainous and forested regions, this work developed an advanced smart fire detection system integrating AI, ML, and IoT technologies. This system aims to enhance timely fire detection and mitigation, minimizing the impact on natural habitats, wildlife, and human communities. The shift from traditional fire detection methods to AI-driven approaches marks a critical advancement in wildfire management, leveraging real-time data processing and autonomous decision-making for swift and accurate fire detection.

Key findings reveal high accuracy in negative instance detection with the YOLO-based fire detection model, but limitations in identifying positive instances due to data scarcity and class imbalance. This highlights the need for comprehensive datasets encompassing diverse fire scenarios for improved model robustness.

Future advancements in AI algorithms, sensor technologies, and IoT integration are crucial for enhancing the scalability and adaptability of fire detection systems. Expanding dataset repositories, employing rigorous data augmentation techniques, balancing datasets, enhancing data collection, and using better camera modules will further bolster model generalization capabilities.

In conclusion, this project has laid the groundwork for an effective smart fire detection system. Addressing the identified challenges and leveraging emerging technologies will pave the way for more resilient wildfire management strategies, protecting ecosystems and communities from wildfires in Algeria and beyond.

- [1] Global Forest Watch, consulted: 10/05/2024,
<https://www.globalforestwatch.org/dashboards/country/DZA/?location=WyJjb3VudHJ5liwiRFpBll0%3D>
- [2] Direction Générale des Forêts, consulted: 25/05/2024, <http://dgf.org.dz/fr>.
- [3] Nesrine Touahria, Romaisa Bouhamam , UAV Aerial Image-Based Forest Fire Detection Using Artificial Intelligence, Univeristy Saad Dahlab Blida, 2022/2023.
- [4] A. Hassan and A. I. Audu*, TRADITIONAL SENSOR-BASED AND COMPUTER VISION-BASED FIRE DETECTION SYSTEMS: A REVIEW, Department of Computer Engineering, University of Maiduguri, Maiduguri, Nigeria, 10/04/2022.
- [5] Ahmad Alkhatib and Khalid Jaber, Advances in Forest Fire Detection, Prediction and Behavior: A Comprehensive Survey, 1Department of Cyber Security, Al-Zaytoonah University of Jordan, Airport St, Amman, Jordan 2Department of Computer Science, Al-Zaytoonah University of Jordan, Airport St, Amman, Jordan, 04/12/2023.
- [6] Akila Keddous, UAV Aerial Image-Based Forest Fire Detection Using Deep Learning, Blida 01 University 2020/2021.
- [7] Dye-sensitized solar cells (DSSCs) as a potential photovoltaic technology for the self-powered internet of things (IoT) applications, Research gate, consulted: 22/06/2024,
https://www.researchgate.net/figure/a-Structure-of-Internet-of-Things-and-b-different-types-of-sensors-94_fig4_343049219.
- [8] Artificial Intelligence, Machine Learning, and Deep Learning: Same context, Different concepts, consulted: 20/06/2024,
<https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/>.
- [9] fire detection dataset in YOLO format, consulted: 25/04/2024,
<https://www.kaggle.com/datasets/ankan1998/fire-detection-in-yolo-format>

Bibliography

- [10] YOLOv8 Architecture: A Deep Dive into its Architecture, YOLO.org, consulted 20/05/2024, <https://yolov8.org/yolov8-architecture/>.
- [11] FATOUMATA Y & AMOR A, Machine learning pour la maintenance prédictive, Mémoire de fin d'études, Université Larbi Ben Mhidi d'Oum El-Bouaghi, juillet, 2021.
- [12] MIFDAL R, Application des techniques d'apprentissage automatique pour la prédiction de la tendance des titres financiers, L'obtention De La Maitrise, Sous la direction de M. Edmond Miresco, École De Technologie Supérieure Université Du Québec, 2019,. p 176.
- [13] Kaleem Nawaz Khan, Deep learning based classification of unsegmented phonocardiogram spectrograms leveraging transfer Learning, 27 September 2021.
- [14] S. Deepak, P.M. Ameer, Classification des tumeurs cérébrales à l'aide de fonctionnalités CNN approfondies via l'apprentissage par transfert. Informatics in Biology and Medicine, 2019.
- [15] what is SMTP?, consulted: 22/06/2024 <https://www.studytonight.com/computer-networks/smtp-protocol>.