

Improving Fire Detection Using YOLOv10: Techniques Used to Detect Flames and Smoke

K.YAMINIVARALAKSHMI¹, MRS N.SUREKHA²,

¹Final year MCA Student, Department of Computer Applications, GIET Engineering College, Andhra Pradesh, India

² Assistance Professor, GIET Engineering College, Andhra Pradesh, India

Abstract – This paper highlights the advanced algorithm for fire detection, which can be used for improving fire detection technology in order to provide better safety for people. To ensure more accurate detection of flames and smoke in different conditions, YOLOv10 algorithm was introduced by this research group. This project suggests several significant improvements of the YOLO framework in order to cope with such issues as the presence of cluttered environment, low visibility, various levels of intensity of fire as well as overlapping of other objects in the image. In order to solve these issues, the proposed system applies advanced feature extraction and attention techniques to detect small but important details related to fire and exclude all unnecessary background information from consideration. In addition, the proposed system implements the advanced bounding box regression technique to enable better localization of fire elements on images and in video sequences. Results of the experiments conducted with this model indicate superior performance compared to similar approaches like YOLOv5s in terms of mAP, precision and recall metrics. Besides, the processing speed is relatively fast, thus making the model applicable for industrial purposes.

1.INTRODUCTION

The increasing necessity for efficient fire detection systems has gained considerable attention because fires can be detrimental to human safety and property. Conventional fire detection systems like smoke detectors and heat sensors have certain drawbacks in terms of detecting flames and smoke in diverse and large-scale settings. This paper presents an innovative solution by designing a fire detection system using the You Only Look Once v10 algorithm, which makes use of contemporary deep learning technology to detect and locate fires and smoke. YOLOv10 has been selected as the backbone of this fire detection system because it can detect multiple objects at the same time with high accuracy and speed, thereby providing a perfect fit for

real-life applications. This is achieved by incorporating better feature extraction, use of attention model, and a superior bounding box regression scheme to overcome the problems related to the detection process caused by cluttered background, overlapping objects, and diverse fire conditions. The combination of these techniques ensures reliable detection of flames and smoke despite adverse conditions. Not only does this system deliver precise fire detection, but it also ensures real-time processing for immediate alerting in case of any fire occurrence. With a wide range of applications including industrial facilities, public places, residential sites, and even large scale forests, this system aims to deliver fire safety and prevention. In general, one of the key objectives of the current project is to create a powerful and efficient fire detection system capable of preventing fire accidents and helping to minimize damage to property and lives caused by fires. To achieve that, the proposed project uses some of the most advanced machine learning techniques, including transfer learning and hyperparameters tuning. Fine-tuning the pre-trained YOLOv10 algorithm allows to make this system much more efficient at detecting the objects of interest, i.e., flames and smoke clouds. Furthermore, using such advanced methods as data augmentation helps the system to deal effectively with any changes that might take place within the environment.

1.2 SCOPE OF THE PROJECT

This project will mainly deal with improving the accuracy, precision, and speed of detection of the flames and smoke using the YOLOv10 algorithm. The project will aim to detect the fire risks by analyzing the environment in real-time. The project will focus on detecting fire risks in different environments which include factories, residential buildings, public places, and even forests. The project takes care of challenges that may occur in detecting fire risk such as messy background, visibility, intensity of the fire risk, and overlapping

objects among others. The project will take advantage of the advanced features provided by the YOLOv10 algorithms in terms of object detection. The feature of the attention mechanism will allow the system to be focused in the right areas in order to avoid any unnecessary distractions. Another aspect is the improved algorithm in bounding box regression which helps in locating the flames and smoke precisely even in the complicated environments. Therefore, this project has the potential to become an excellent application for big projects such as cities and industries.

1.3 OBJECTIVE: The main objective of the project involves designing a highly efficient fire detection model that is based on the implementation of the YOLOv10 algorithm for fire flame and smoke localization. It is necessary to develop the proposed solution through using more sophisticated approaches than those which could be implemented within the initial version of YOLO (such as feature extraction and regression). In particular, special attention is paid to the design of attention-based methods for dealing with the problem of background clutter and improving performance in case of weak and strong fires. Another goal is associated with creating a model that is resistant to difficult working conditions related to poor visibility or presence of clutter around the camera (e.g., foliage, buildings, etc.). With the help of transfer learning and data augmentation, the developed solution is to become capable of operating in real-time mode.

1.4 EXISTING SYSTEM: The current system under investigation makes use of the YOLOv5s algorithm for the detection of flames and smoke in the input images/videos. The YOLOv5s algorithm is an object detection algorithm, popular among developers for its efficiency and speed. In relation to fire detection, this object detection algorithm is able to recognize and localize several objects in a scene, including flames and smoke, by using a bounding box. The current system relies on the capacity of YOLOv5s of being able to detect and classify objects in a single pass to make it possible for the application to operate in a real-time mode. To do so, YOLOv5s employs a standard feature extraction technique that makes use of pre-trained weights of the model to detect objects. This model proves to be efficient, yet its performance in complex environments could be affected by some limitations. For instance, YOLOv5s can experience difficulties detecting fires and smoke in cases of a cluttered environment or in conditions where the intensity of fires or visibility of the smoke is low. While still proving to be fairly effective and efficient, YOLOv5s

would find it hard detecting flames and smoke in such environments in identifying flames and smoke. The effectiveness of the YOLOv5s model can be described as relatively fast and efficient, but there could be certain challenges to its performance in identifying flames and smoke when there are overlapping objects or when the visibility of the flames and smoke is low. Nonetheless, this model serves as a good starting point to develop models that can detect fires effectively, and yet some improvements can still be made through the use of other advanced techniques to further enhance fire detection in real-world environments, thus leading to the development of YOLOv10.

1.4.1 EXISTING SYSTEM DISADVANTAGES:

→ Detection Accuracy: YOLOv5s might fail to capture small or partially hidden flames and smoke due to the possibility of false negative, while YOLOv10 provides high detection accuracy in cluttered scenarios.

→ Feature Extraction: The model has problems extracting features from flames and smoke, while YOLOv10 excels at this aspect.

→ Non-Priority Feature Extraction Capability: The model applies equal effort in processing all parts of the images, while YOLOv10 applies attention mechanisms that focus on fire related regions.

→ Regression Method: The regression methods used in YOLOv5s yield low accuracy while that of YOLOv10 yields high accuracy.

→ Performance in Dense Environments: YOLOv5s might not be able to detect fire and smoke in a dense scene, while YOLOv10 performs well under such conditions.

→ Limitations in Real-Time Object Detection: YOLOv5s might face issues in the detection of objects in high resolution at real-time speed, unlike YOLOv10.

1.5 LITERATURE SURVEY Title: Enhancing geometric factors in model learning and inference for object detection and instance segmentation Author: Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, Year: 2022. Description: The title of the paper under review is 'Enhancing geometric factors in model learning and inference for object detection and instance segmentation'. The paper was authored by Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo and published in the IEEE Transactions on Cybernetics, in August 2022. In this paper, the topic of incorporating

geometric factors in the process of learning and inference of object detection and instance segmentation models has been discussed. Traditional object detection models have successfully contributed to this field; however, they have failed to consider geometric aspects in the models, which would improve localization or segmentation performance significantly.

In this regard, the research proposes innovative techniques to integrate geometric factors into model learning and inference procedures. According to the author, through integration of the geometric cues like object geometry, orientation, and relative position in the scene, the detection and segmentation model will perform better since it will be able to accurately detect and segment even those objects that have very complicated or irregular shapes. The paper shows how geometric factors play an important role in improving the accuracy of feature extraction and bounding box estimation. With respect to instance segmentation where there is need to differentiate instances from one another and even where the instances are near or overlap one another, it is clear that the importance of the geometric features cannot be overlooked.

Title: CSPNet: A new backbone that can enhance learning capability of CNN

Author: C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh

Year: 2020.

Description: "CSPNet: A new backbone that can enhance the learning capability of CNN" is a paper by authors C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh that appeared in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), June 2020. In this paper, the authors propose a novel backbone architecture for Convolutional Neural Networks (CNN). The primary contribution of this paper is the introduction of CSPNet (Cross-Stage Partial Network), which is designed to increase the learning capability of CNN architectures. Conventionally, as CNN architectures become increasingly deep, they experience issues such as problems in gradient flow, difficulties in optimization, and inefficient learning processes. These drawbacks often result in slow convergence of the model during training as well as poor performance in other applications such as

image recognition, detection, and segmentation. Rather than applying feature extraction using only one phase, it segregates the feature maps into a number of different categories, where some feature maps go through without undergoing further processing, while the other undergo a partial processing phase before being merged together. This assists in focusing on the critical aspects of a network and avoids unnecessary computations that might lead to overfitting or inefficient operations. In addition, by using different phases for computations, CSPNet facilitates an efficient training process and ensures effective gradient propagation, enabling better learning capabilities for deep learning models. On the grounds of performance, the CSPNet model has been seen to surpass traditional CNN models due to its reduced computational requirements while maintaining its accuracy.

Title: Coordinate attention for efficient mobile network design

Author: Q. Hou, D. Zhou, and J. Feng

Year: 2021.

Description: The above-referenced article "Coordinate Attention for Efficient Mobile Network Design" authored by Q. Hou, D. Zhou, and J. Feng, delivered at the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) in June 2021, proposes a new type of attention architecture called the Coordinate Attention (CA). This method seeks to provide a way of achieving a compromise between performance and efficiency when designing attention mechanisms for mobile network applications. Typically, mobile networks are characterized by the limitation in terms of computational resources and capacity. For instance, computation capabilities, memory, and bandwidth in mobile networks may not be enough to support massive model deployment. This makes it imperative to develop light models that can perform efficiently on such devices.

In the proposed work, the Coordinate Attention aims at extracting dependencies along spatial and channel dimensions using the concept of coordinates of the feature map. Attention-based methods like CA and SA concentrate on the aspect of channel-wise or spatial attention separately but fail to consider dependency among both aspects. To address the issues, Coordinate Attention incorporates spatial and channel dependencies via the incorporation of the concept of coordinates of the

feature map in its attention process. This mechanism makes the algorithm computationally efficient since an appropriate approach is used to calculate the attention. In Coordinate Attention, the input feature maps are divided into two main parts, including the part responsible for processing the spatial dependencies of the maps and the other responsible for processing the channel-wise dependencies.

Title: Multi-temporal dependency handling in video smoke recognition: A holistic approach spanning spatial, short-term, and long-term perspectives **Authors:** F. Yang, Q. Xue, Y. Cao, X. Li, W. Zhang, and G. Li

Year: 2024

Description: The paper entitled "Multi-temporal Dependency Handling in Video Smoke Recognition: A Holistic Approach Spanning Spatial, Short-term, and Long-term Perspectives" by F. Yang, Q. Xue, Y. Cao, X. Li, W. Zhang, and G. Li, which appeared in the journal Expert Systems with Applications (July 2024), provides a novel technique for the recognition of smoke in video sequences. The authors address the problem of multi-temporal dependencies that arise in video sequences and present their proposed method for detecting smoke in the video. Traditional approaches for recognizing smoke in video have faced difficulty in dealing with multi-temporal dependencies. The reason is that the behavior of smoke in a video sequence may vary depending on different environmental conditions or other factors, which make the detection process difficult. In this regard, the authors present a method that includes the consideration of both spatial and temporal dependencies in the detection process. In essence, the method incorporates three key concepts: the inclusion of spatial information (that is, information on smoke appearing in certain locations in the image frame), as well as short-term (the immediate transition between the frames) and long-term (the general evolution of smoke during an extended period) temporal dependencies. As a result, the use of such a combination enables the model to take into account all three approaches to detect smoke in a much more reliable and accurate manner. In order to address these temporal dependencies, a new concept is introduced, which makes it possible to use both CNNs for spatial analysis and RNNs to model the temporal dependencies. While the former can be utilized to extract features within the spatial frame, RNNs can be used to learn the short-term temporal dependencies by analyzing adjacent video frames. Meanwhile, the long-term dependencies are accounted for through memory-based mechanisms, which allow

capturing the general trends in the evolution of smoke over extended periods.

Title: Improved fire detection by YOLOv8 and YOLOv5 to enhance fire safety

Author(s): M. N. Uddin, M. Sakibul Islam Sakib, S. Nawer, and R. T. Mohona

Year: 2023

Description: The above paper entitled "Improved Fire Detection by YOLOv8 and YOLOv5 to Enhance Fire Safety" written by M. N. Uddin, M. Sakibul Islam Sakib, S. Nawer, and R. T. Mohona was delivered in the 26th International Conference on Computer and Information Technology (ICCIT) held in December 2023. This research paper discusses improving fire detection systems based on advanced machine learning techniques for object detection, specifically YOLOv8 and YOLOv5. The objective of this research is to improve the fire detection systems that are critical to the application of fire safety measures, including those used in industries, commercial spaces, and residences. It is important to detect fire incidents in real-time so that prompt action can be taken to minimize any possible damages. The research compares the efficacy of two variants of the famous YOLO algorithm called YOLOv8 and YOLOv5 and their application in detecting fires. It should be mentioned that the problems of fire detection can be connected with the following issues: false alarms, inability to recognize fire in case of similar phenomena, different intensities of fires depending on environments. For the sake of solving these problems, YOLO models are implemented. The first model to be considered is YOLOv5 – a well-known object detection algorithm which serves as a strong basis for comparison with other models. The YOLOv8 is more recent than the YOLOv5 model. The latter algorithm contains enhanced feature extraction mechanisms, small objects recognition and objects recognition in various backgrounds.

1.5 PROPOSED SYSTEM

The proposed system uses YOLOv10 algorithm for efficient flame and smoke detection in pictures and video streams. The algorithm belongs to the YOLO family of algorithms and is designed to be superior to other algorithms such as YOLOv5s due to specific innovations. The system is capable of dealing with complex scenes including different background noises, fires with different

intensities and low visibility. It is important to highlight the fact that the proposed method uses a special algorithm for improved feature extraction which is supposed to improve accuracy of recognition.

Yet another innovation introduced by this work is the use of the advanced bounding box regression technique that helps detect flames and smoke in the most precise way possible. This new approach enables the model to effectively find fire-related objects on images, even when there are several overlapping objects on the image or when some parts of the object are occluded. Importantly, the new YOLOv10-based system is created specifically to be used for real-time fire detection. Thus, using this system, users will be able to receive timely notifications about the occurrence of fire-related objects in a variety of environments, whether it be an industrial plant, public space, or residential area. The proposed system can function effectively in different conditions and is adaptable. To sum up, the new fire detection system significantly outperforms the current YOLOv5s model concerning accuracy, recall, and mean average precision (mAP).

1.6.1 PROPOSED SYSTEM BENEFITS: • Increased Accuracy: YOLOv10 achieves higher accuracy when detecting flames and smoke under difficult circumstances, such as those with poor visibility or partial occlusions.

• Improved Feature Extraction: With YOLOv10, feature extraction is enhanced to detect flames and smoke in cluttered and dynamic environments with greater success.

2.1 PROJECT DESCRIPTION: This project is about the development of an advanced fire detection system based on the usage of the YOLOv10 (You Only Look Once) algorithm that would be capable of detecting flames and smoke in real time. The key idea behind the project development is overcoming several disadvantages of existing fire detection systems, which include poor performance when working in complicated environments and with different levels of fire intensity, limited capacity to function properly in low-visibility conditions, etc. The chosen algorithm can detect several objects at once in an image or a video stream. In particular, the system is aimed at detecting flames and smoke by means of implementing several innovative methods for improving the efficiency of the system's functioning. These methods are connected to the advanced implementation of feature extraction

algorithms, use of the attention mechanism for focusing on fire-related features, better implementation of the bounding boxes regression to provide higher accuracy of object detection. Transfer learning technology is used to benefit from pre-trained models, and data augmentation contributes to making the system more reliable under varied circumstances. • Advanced Attention Model: YOLOv10 pays attention to fire-related areas only, which results in faster and more accurate fire object detection due to filtering out non-fire data. • Advanced Bounding Box Regression Method: YOLOv10 provides better performance in locating fire-related objects thanks to its superior bounding box regression algorithm. • Excellent Fire Object Detection in Dense Environments: When compared to YOLOv5s, YOLOv10 can detect fire objects successfully in highly crowded scenes. • Optimized Fire Object Real-Time Detection: YOLOv10 processes HD images more efficiently, allowing fire object real-time detection at large scale.

The system's primary goal is to provide early, reliable detection of fires, reducing the risk of damage to property and loss of life. It is intended for use in diverse environments, including industrial plants, residential areas, public spaces, and even large-scale outdoor areas like forests. With real-time processing capabilities, the system ensures quick alerts, allowing for rapid response to fire incidents. The project aims to create a scalable, efficient, and adaptive fire detection solution that enhances fire safety and prevention measures in various contexts.

2.2 METHODOLOGIES

2.2.1 MODULES NAME :

- Objects Detection Engine
- Pre-Trained Network Model Adaptation
- Model Performance Tuning
- Data Augmentation and Enhancement
- Model Training and Validation
- Real Time Fire and Smoke Detection and Localization
- Module Performance Evaluation and Monitoring

2.2.2 MODULES DESCRIPTION

1) Objects Detection Engine: This is an important part of our project because the module is designed for objects identification and localization in order to identify any

objects present within an image or video frame. With the help of YOLOv10 framework, the object detection engine classifies the objects and identifies their location in the picture. This module will be used for real-time fire and smoke detection due to the speed at which it works.

2) Model Adaptation through Pre-Trained Networks:

The present module involves utilizing the pre-trained network for improving the performance of the model by utilizing the transfer learning mechanism. In the current system, we will start off with the pre-trained YOLOv10 model, which has already been trained in a large general dataset. The model is then further fine-tuned based on a smaller and domain-specific fire dataset. This helps the model to learn fire-related features more quickly and efficiently without requiring large amounts of labeled data.

3) Performance Tuning Module: The primary aim of the present module is to tune the model's parameters to ensure maximum performance in terms of accuracy and convergence. This will involve fine-tuning various parameters including the learning rate, batch size, and number of epochs to obtain accurate results.

4) Data Expansion and Augmentation: Data augmentation is performed via the data augmentation module. It artificially increases the number of samples in a training dataset through transformation techniques, such as image rotation, flipping, scale adjustment, changes in lighting or contrast. The purpose of using such a module is the creation of a robust solution which will be able to work well under various conditions.

5) Training and Validating a Model: This module is intended for the process of training a fire detection model on labeled data and validating its performance on other sets of labeled images and video frames. In other words, the model is being taught to distinguish various features characteristic of fire and after learning, its ability to perform classification tasks on previously unseen images or video frames is evaluated.

6) Real-time Detection of Flames and Smoke: After all training phases and validation are completed, the task is performed by the real-time fire and smoke detection and localization module. As soon as the system receives an image or video frame, this module recognizes fire-related features and indicates their presence in the scene as well as localizes the object or area where they appear.

7) Performance Evaluation and Monitoring:

Evaluation Module: This module helps in the assessment of the performance of the developed fire detection system by the help of various metrics including accuracy, precision, recall, F1 score etc. By means of this module, the performance of the system can be evaluated efficiently and effectively and thereby improvements can also be made where required. This module ensures the scalability of the system for large datasets as well.

2.3 TECHNIQUE USED OR ALGORITHM USED

2.3.1 EXISTING TECHNIQUE: - • YOLOV5S In the current work, the existing technique utilized is YOLOV5s i.e. You Only Look Once. This algorithm is basically used in order to detect objects in a real time environment. The main strength of this algorithm lies in its capability to conduct both classification and localization in a very effective manner. In this model, an image is divided into grids. After that, predictions for the bounding boxes containing the objects in the image as well as their respective classes are made by this model.

Still, despite its efficacy in performing object detection in general, there are some shortcomings that make this algorithm less useful in performing specific tasks, such as fire detection. For instance, the YOLOv5s algorithm may find it difficult to detect objects in cases where the environment is cluttered or in cases when there is poor visibility of the scene, such as in smoke detection cases. The problem here is that this algorithm may be incapable of detecting objects that are subtle in terms of their features, such as weak flames and light smoke. This is because the YOLOv5s algorithm is designed to prioritize speed over resolution and the ability to detect fine details. The YOLOv5s algorithm also lacks attention mechanisms, implying that it may not be able to highlight important areas in an image. This implies that it is likely to make mistakes when trying to detect fires and smoke, particularly when they are small, far away from the camera, and partially hidden in an image.

2.3.2 PROPOSED TECHNIQUE USED OR ALGORITHM USED:

• **YOLOv10:** The technique used for implementing the project in its advanced form is YOLOv10. YOLO stands for You Only Look Once, which is a type of technique that allows enhancing the efficiency of an object detection system. YOLOv10 represents the tenth version of such a detector, which means it outperforms all previous versions of this neural network in terms of performance,

especially when compared to the YOLOv5s model. YOLOv10 implements several new techniques, allowing better detection of flames and smoke in comparison to its predecessors. The first improvement implemented by YOLOv10 is the introduction of the enhanced feature extraction mechanism that allows better detection of subtle flames and smoke particles. The second important development of this model concerns its ability to apply selective attention to critical parts of an image. This process involves focusing on certain features and ignoring distracting details of a picture. Such approach makes it possible to achieve better results with the use of YOLOv10 in cases of flame and smoke detection in a noisy environment. Moreover, it is optimized for high resolution image and video processing, which enables it to be used for fire detection purposes in real-time environments such as industries, households, or public places. In addition to its improved capabilities, YOLOv10 has proved to be better than its predecessors when it comes to accuracy and adaptability due to varying environmental situations. Moreover, it is more accurate as compared to the previous versions such as YOLOv5s when it comes to precision, recall, and mAP.

CHAPTER 3 REQUIREMENTS ENGINEERING

3.1 GENERAL

From the results we can observe that the error rates are very minimal owing to the discriminative ability of features as well as the regression abilities of classifiers. In terms of highest accuracy (which corresponds to the lowest error rate), our results are very promising when compared to previous works.

3.2 HARDWARE REQUIREMENTS

The hardware requirements will form the basis of a contract in the implementation process of the system and hence need to be comprehensive in describing the system. The hardware requirements will be used by software engineers as a starting point in designing the system. The hardware requirements should indicate what the system does and not how it should do things.

- PROCESSOR :DUAL CORE 2 DUOS
- RAM :4GB DD RAM
- HARD DISK :250 GB

3.3 SOFTWARE REQUIREMENTS

The software requirements will provide the specifications of the system. The software requirements will consist of the requirement definitions and specifications. The software requirements document is a list of what the system is expected to do but not how it should achieve this. The software requirements will form a basis of the software requirements specifications.

- Operating System : Windows 7/8/10
- Platform : Anaconda
- Programming Language : Python • Front End : VS code

3.4 FUNCTIONAL REQUIREMENTS

Functional requirements state functionality provided by the software-system or its component. Functionality is stated as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

3.5 NON-FUNCTIONAL REQUIREMENTS

Following are the main non-functional requirements of the system Usability The system consists of completely automated process hence there is no or little user intervention. Reliability The system is quite reliable due to the attributes, which are inherited by the chosen platform Python. The code generated by Python is reliable. Performance This system is developed using high level languages and advanced back-end technology; therefore, it gives faster response time to the end user on client machine. Supportability This system is designed to be cross platform supportable. This system is supported on a wide variety of hardware and any software platform, which is embedded in this system. Implementation This system is developed using Jupyter Notebook web platform.

CHAPTER 4

DESIGN ENGINEERING

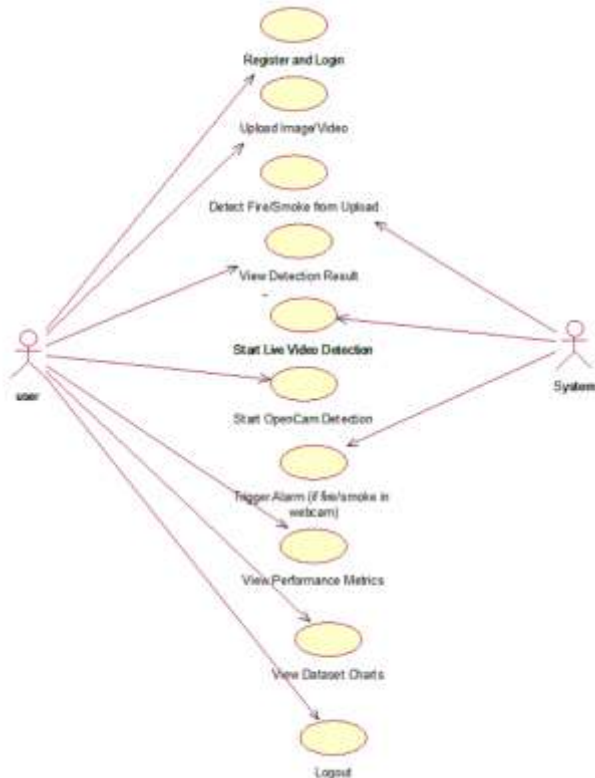
4.1 INTRODUCTION

Design Engineering focuses on the use of different UML diagrams for the development of project. A design is a sensible engineering depiction of something that is going to be made. Software design is a process of converting the

needs into a form of software depiction. Design is where the quality gets depicted within software engineering.

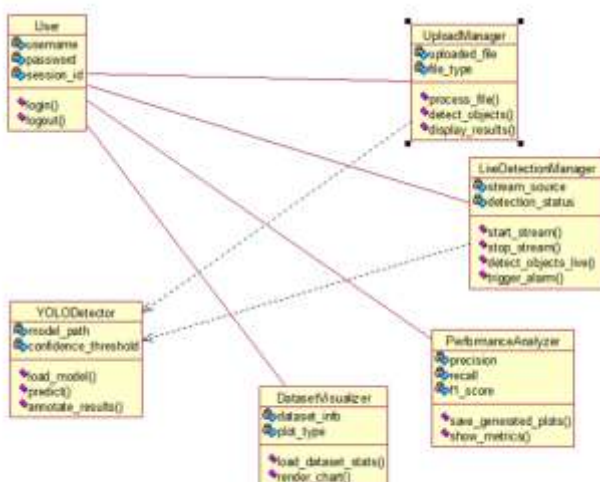
4.2 UML DIAGRAMS

4.2.1 USE CASE DIAGRAM



EXPLANATION: The primary objective of a use case diagram is to illustrate the functions that are performed for which actor within the system. The roles of the actors within the system can be explained. The above use case diagram comprises user as an actor. Every one of them plays a specific role to achieve the concept.

4.2.2 CLASS DIAGRAM



EXPLANATION

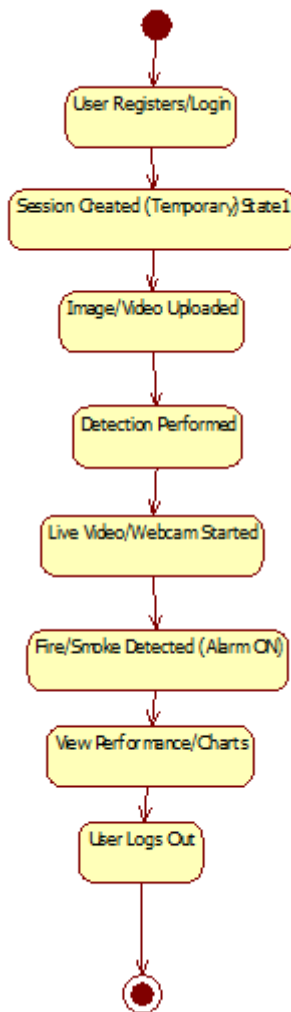
In the above class diagram explains about the relation between the classes with attributes and methods that performs verification using security. The classes involved in this project are displayed in the above diagram.

4.2.3 OBJECT DIAGRAM



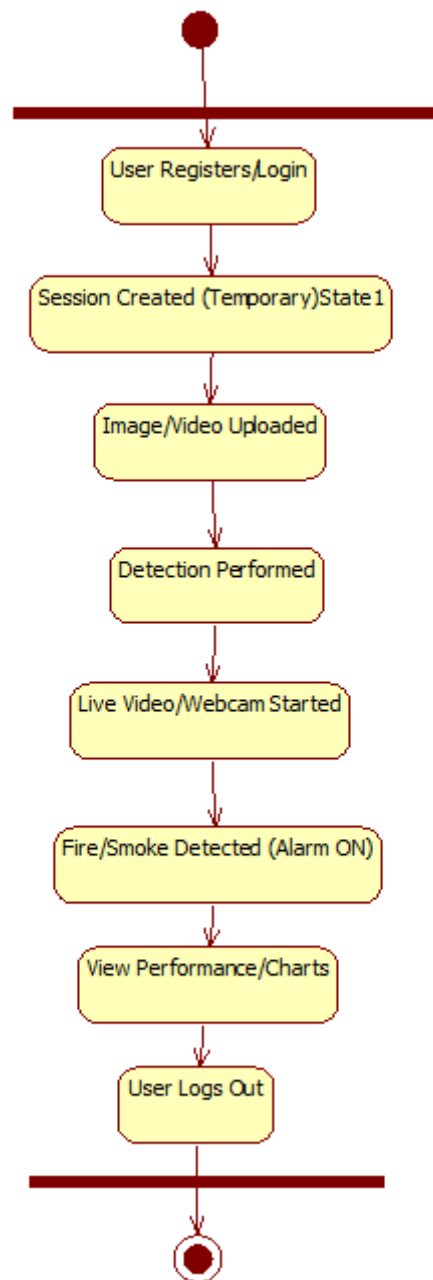
EXPLANATION: In the above diagram, the flow of objects between classes is explained. Object diagrams illustrate a snapshot of the structure of a system at a particular point in time. In this, the object diagram explains about how the classes with attributes and methods are linked together to perform the verification with security.

4.2.4 STATE DIAGRAM



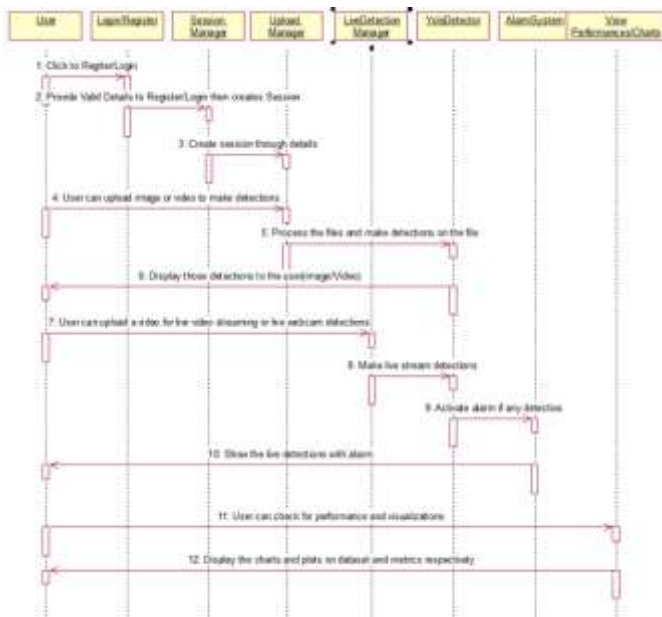
EXPLANATION: A state diagram provides a general-purpose mechanism for representing workflows of stepwise activities within a system. Many forms of state diagrams exist, which are slightly different and have different semantics. Generally, state diagrams require that the system under consideration is made up of a finite number of states; this is often a reasonable abstraction, although occasionally this needs to be relaxed.

4.2.5 ACTIVITY DIAGRAM



EXPLANATION: An activity diagram is a type of diagram that illustrates the workflows of stepwise activities and actions. It depicts the overall flow of control. Activity diagrams can be used to describe business and operational workflows within a system.

4.2.6 SEQUENCE DIAGRAM



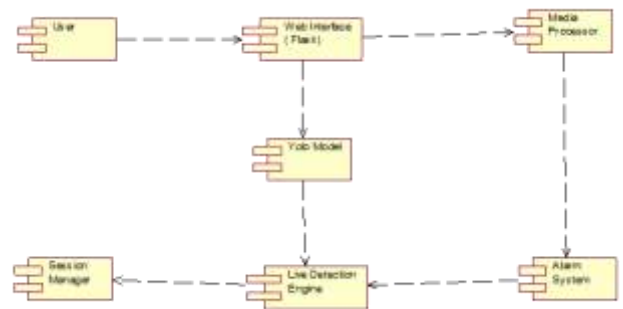
DESCRIPTION: Sequence Diagram is an interaction diagram in Unified Modeling Language (UML) that captures the way processes interact with each other and in which order. A sequence diagram is a component of a Message Sequence Chart. In a sequence diagram, interactions between different objects happen in time sequence manner. The interactions of objects involved in a particular process are depicted by means of a sequence diagram.

4.2.7 COLLABORATION DIAGRAM



DESCRIPTION: Collaboration diagram (also referred to as a communication diagram or interaction diagram) is a diagram representation of the relationships between different software objects in Unified Modeling Language (UML). The idea is not new and goes back over a decade though it has improved with time as modeling has changed.

4.2.8 COMPONENT DIAGRAM



EXPLANATION: Component diagrams represent how components are interconnected to form larger components or software systems in the Unified Modeling Language. These are often used to demonstrate the architecture of arbitrarily complicated systems. User provides the primary question; it is then broken down into sub-queries and delivered to the data aggregators via data dissemination. The answers should be displayed to the user by data aggregators. All boxes are components, while arrows symbolize dependencies.

4.2.9 DATA FLOW DIAGRAM

Level 0



Level 1

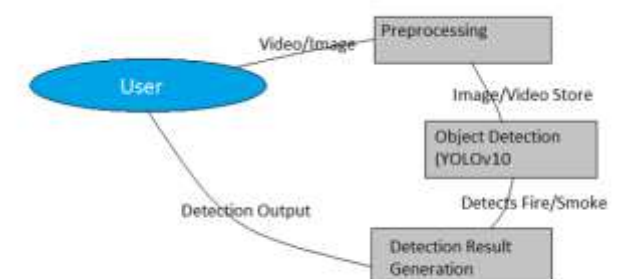
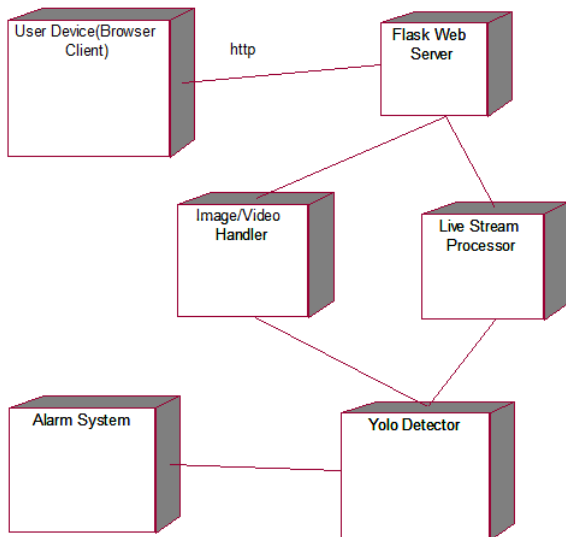


Fig 4.9: Data Flow Diagrams

EXPLANATION: A data flow diagram (DFD) is a chart that represents the "flow" of data through an information system by modeling its process aspects. DFDs are usually used as a first step in order to get an overview of the system and then elaborate on the subject. DFDs may also be employed in visualizing data processing. A DFD describes what data is input to and output from the

system, the sources and destinations of such data, and storage areas. A DFD does not include information regarding the scheduling of tasks within the process or whether processes run concurrently or sequentially.

4.2.10 DEPLOYMENT DIAGRAM



EXPLANATION: A Deployment Diagram is a kind of diagram that explains the physical hardware on which the software system will be running. This also defines how the software will be deployed in the underlying hardware. It is mapping the software components of a system into the devices where the software will run.

SYSTEM ARCHITECTURE:

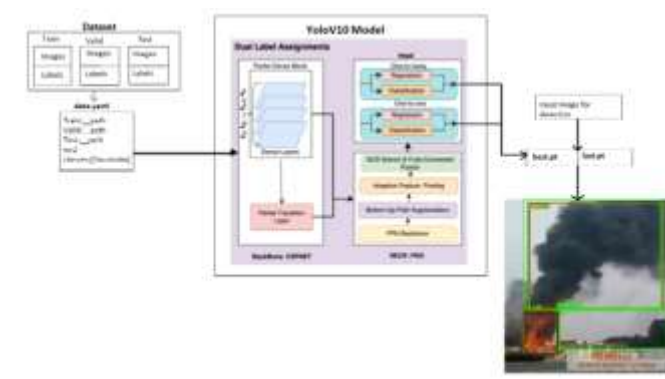


Fig 4.11: System Architecture.

Chapter 5 Development Tools

5.1 Python

Python is a high-level, interpretive, interactive and object-oriented programming language. Python was

created to be highly portable. Its syntax uses English keywords commonly where as other languages use punctuation, and it has less syntactical structure than other languages.

5.2 History of Python

Python was originally created by Guido Van Rossum during the late eighties and early nineties at Centrum Wiskunde & Informatica (CWI) in the Netherlands. Python is based on various other languages, which includes ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other languages. Python is copyrighted. Similar to Perl, Python source code is presently available under the GNU General Public License (GPL). Nowadays, Python is maintained by a group of community developers, but Guido Van Rossum still holds an important position in its development path.

5.3 Importance of Python

- Python is Interpreted – The Python interpreter executes the programs. It requires no compilation step before execution and is executed the same way script languages are executed. This is one of those aspects that Python gets from PERL and PHP.
- Python is Interactive – It allows the programmer to use the interpreter interactively.
- Python is Object-Oriented – Python facilitates the programmer to develop using the Object-Oriented style or technique.
- Python is a Beginner's Language – Python is highly readable and is recommended as a first language for beginners.

5.4 Features of Python

- Easy-to-learn – Python has fewer keywords and straightforward structure thus it allows students to pick-up the language rapidly.
- Easy-to-read – Python code is easily readable and can be written with indentations.
- Easy-to-maintain – Code maintainability with other languages is quite good.
- Broad Standard Library – Python has a rich library which covers various areas like internet protocols, string operations, operating system interfaces etc.

- Interactive Mode – Python provides support for an interactive mode which allows interactive testing and debugging of snippets of code.

- Portable - Python works with multiple hardware platforms and has the same interface on all hardware platforms.

- Extensible - Low-level modules can be added to the Python interpreter. Such functionality allows programmers to extend or modify their tool in an effective way.

- Databases - Interfaces are provided by Python to access most commercial databases.

- Graphical user interface programming - It is possible to write GUI-based programs in Python that can run on any system calls, libraries and windowing systems, such as Windows MFC, Macintosh and the X Window System of UNIX.

- Scalability - Python has a better support for large programs than shell scripting. Apart from the above-listed features, here are some more advantages of using Python –

- Python supports both procedural and object-oriented programming approaches.

- The language can be used for scripting as well as compiled into byte-code for developing large applications.

- Python offers highly dynamic data types and dynamic type checking.

- Automatic garbage collection is supported by the language.

- Integration with other languages is also easy, i.e. Python can be easily interfaced with C, C++, COM, ActiveX, CORBA and Java.

5.5 Python libraries: • numpy – useful primarily for its N-dimensional array objects.

- pandas – data analysis library in Python that includes data frames as structures.

- matplotlib – 2-D graphics package that creates publication quality plots.

- scikit-learn – machine learning algorithms used in data analysis and mining.



Figure: NumPy, Pandas, Matplotlib, Scikit-Learn

CHAPTER 6

IMPLEMENTATION

6.1 GENERAL Code:

```
from flask import Flask, render_template, request,
redirect, url_for, send_from_directory, flash, Response,
session
```

```
from ultralytics import YOLO import os,
subprocess, cv2, sqlite3, time, threading, pytsx3
```

```
from werkzeug.utils import secure_filename
```

```
app = Flask(__name__)
```

```
app.secret_key = 'supersecretkey'
```

```
UPLOAD_FOLDER = 'uploads'
```

```
RESULTS_FOLDER = 'results'
```

```
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
```

```
os.makedirs(RESULTS_FOLDER, exist_ok=True)
```

```
app.config['UPLOAD_FOLDER'] =
UPLOAD_FOLDER
```

```
app.config['RESULTS_FOLDER'] =
RESULTS_FOLDER
```

```
model = YOLO('best.pt')
```

```
# ----- DATABASE -----
```

```
def init_db():
```

```
conn = sqlite3.connect('app.db')
```

```
c = conn.cursor()
```

```
c.execute("CREATE TABLE IF NOT EXISTS users ( id
INTEGER PRIMARY KEY AUTOINCREMENT,
username TEXT UNIQUE, password TEXT )")
```

```
c.execute("CREATE TABLE IF NOT EXISTS
detections ( id INTEGER PRIMARY KEY
AUTOINCREMENT, class_name TEXT, detected_at
DATETIME DEFAULT CURRENT_TIMESTAMP )")
```

```
conn.commit()
```

```
conn.close()
```

```
init_db()
```

```
def register_user(username, password):
```

```
conn = sqlite3.connect('app.db')
```

```
c = conn
```

```
conn.close()
```

```
def validate_user(username, password):
```

```
conn = sqlite3.connect('app.db')
```

```
c = conn.cursor()
```

```
c.execute("SELECT * FROM users WHERE
username=? AND password=?", (username, password))
```

```
user = c.fetchone()
```

```
conn.close()
```

```
return user
```

```
def save_detection(cls):
```

```
conn = sqlite3.connect('app.db')
```

```
c = conn.cursor()
```

```
c.execute("INSERT INTO detections (class_name)
VALUES (?)", (cls,))
```

```
conn.commit()
```

```
conn.close()
```

```
# ----- VOICE -----
```

```
engine = pyttsx3.init()
```

```
engine.setProperty('rate', 150)
```

```
voice_lock = threading.Lock()
```

```
def speak_alert(class_name):
```

```
def run():
```

```
with voice_lock:
```

```
try:
```

```
if class_name.lower() == "fire":
```

```
text = "Fire is detected" elif "smoke" in
class_name.lower():
```

```
text = "Smoke is detected"
```

```
else:
```

```
text = f'{class_name} detected'
```

```
engine.say(text)
```

```
engine.runAndWait()
```

```
except Exception as e:
```

```
print("Voice error:", e)
```

```
threading.Thread(target=run, daemon=True).start()
```

```
# ----- SETTINGS -----
```

```
CONF_THRESHOLD = 0.7
```

```
camera_running = False
```

```
recording = False
```

```
video_writer = None
```

```
# ----- ROUTES -----
```

```
@app.route('/')
```

```
def home():
```

```
return render_template('home.html')
```

```
@app.route('/index')
```

```
def index():
```

```
if 'user' not in session:
```

```
return redirect('/login')
```

```
return render_template('index.html')
```

```
@app.route('/live')
```

```
def live():
```

```
if 'user' not in session:
```

```
return redirect('/ @login')

else:

flash('Username exists', 'danger')

return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

u = request.form['username']

p = request.form['password']

if validate_user(u, p):

session['user'] = u

return redirect('/index')

else:

flash('Invalid credentials', 'danger')

return render_template('login.html')

@app.route('/logout')

def logout():

session.pop('user', None)

return redirect('/login')

# ----- UPLOAD -----

@app.route('/upload', methods=['POST'])

def upload_file():

file = request.files['file']

if not file:

return redirect('/')

filename = secure_filename(file.filename)

path = os.path.join(UPLOAD_FOLDER, filename)

file.save(path)

# check extension

ext = os.path.splitext(filename)[1].lower()
```

```
img_exts = ('.jpg', '.jpeg', '.png', '.bmp', '.webp')

is_video = ext not in img_exts

result_filename = 'result_' + filename

# --- IMAGE PROCESSING ---

if not is_video:

# (rest of image processing stays same)

results = model(path)

annotated_frame = results[0].plot()

# (rest of saving result to disk etc...)

return render_template('result.html',

original_file=filename,

result_file=result_filename, is_video=False)

# --- VIDEO PROCESSING ---

else:

cap = cv2.VideoCapture(path)

# ORIGINAL PROPERTY RECOVERY original_width

= int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))

original_height

=

int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

fps = cap.get(cv2.CAP_PROP_FPS)

# --- RESIZE LOGIC FOR

out = cv2.VideoWriter(temp_raw_path, fourcc, fps,

(new_width, new_height)) spoken_classes = set()

print(f"Processing video: {filename} at

{new_width}x{new_height}")

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

# Step 1. Resizing first for faster processing

if scale != 1.0:

frame = cv2.resize(frame, (new_width, new_height))
```

```
# Step 2. Detect objects with YOLO

results = model(frame, verbose=False)

annotated_frame = results[0].plot()

# Step 3. Alerts

for box in results[0].boxes:

if float(box.conf[0]) > CONF_THRESHOLD:

cls = model.names[int(box.cls[0])]

save_detection(cls)

if cls not in spoken_classes:

speak_alert(cls)

spoken_classes.add(cls)

out.write(annotated_frame)

cap.release()

out.release()

# --- CONVERT TO BROWSER FRIENDLY MP4 ---

try:

# Making sure that pix_fmt is set to yuv420p which
ensures maximum browser compatibility

cmd = [

'ffmpeg', '-y', '-i', temp_raw_path,

'-vcodec', 'libx264', '-pix_fmt', 'yuv420p',

'-crf', '23', # Quality setting

final_path

]

# Conversion process starts

process = subprocess.run(cmd, capture_output=True,
text=True)

if process.returncode != 0:

print("FFmpeg Error:", process.stderr)

# Failure in converting will cause video not to be
displayed in browser

except FileNotFoundError:

print("ERROR

# ----- LIVE -----

def generate_frames():

global camera_running, recording, video_writer

cap = cv2.VideoCapture(0)

cap.set(cv2.CAP_PROP_FPS, 20)

camera_running = True

while camera_running:

ret, frame = cap.read()

if not ret:

break

results = model(frame)

frame_out = results[0].plot()

# ----- RECORD -----

@app.route('/start_recording')

def start_recording():

global recording, video_writer

recording = True

path = os.path.join(RESULTS_FOLDER,
f"live_{int(time.time())}.mp4")

video_writer = cv2.VideoWriter(path,

cv2.VideoWriter_fourcc(*'mp4v'), 20, (640,480))

return 'Recording Started'

@app.route('/stop_recording')

def stop_recording

# recording optimization

if recording and video_writer:

if int(time.time() * 10) % 2 == 0:

video_writer.write(cv2.resize(frame_out, (640,480)))
```

```

for box in results[0].boxes:
if float(box.conf[0]) > CONF_THRESHOLD:
cls = model.names[int(box.cls[0])]
speak_alert(cls)
_, buffer = cv2.imencode('.jpg', frame_out)
yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' +
buffer.tobytes() + b'\r\n') cap.release()

@app.route('/video_feed')

def video_feed():
return Response(generate_frames(),

mimetype='multipart/x-mixed-replace;
boundary=frame')

@app.route('/stop_camera')

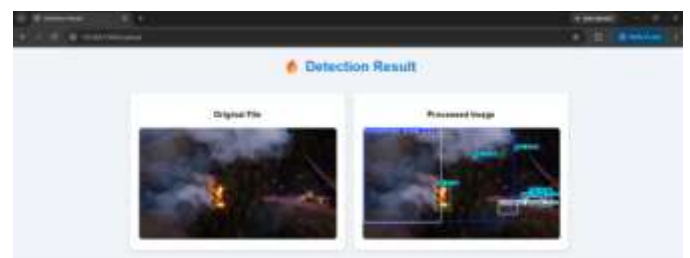
def stop_camera():
global camera_running
camera_running = False
return "Stopped"
# ----- RECORD -----

```

@app.route

CHAPTER 7

SNAPSHOTS CHAPTER



8 SOFTWARE TESTING

8.1 GENERAL

Testing is carried out to identify errors. Testing involves making attempts to discover all possible defects or weaknesses in the work products. It is used to determine the working ability of components, sub assemblies, assemblies, and / or a completed product. It is the process by which software is tested in order to ensure that the system under test conforms to the requirements and does

not exhibit any faults. There are many kinds of tests. These tests are aimed at addressing particular test requirements.

8.2 DEVELOPMENT OF METHODOLOGIES

The process of testing begins with the development of a comprehensive plan to test both general features and special characteristics of the product using a number of platforms. Quality control processes are rigorously observed throughout the process. This process ensures that the application being tested fulfills all requirements stated in the system requirements document and is free of defects. These are some considerations that have been taken into account while developing the framework for the test methodology.

8.3 Types of Tests

8.3.1 Unit Testing

Unit testing includes designing tests for validating whether the internal logic in the program is working correctly and inputs yield valid outputs. All decision branches and code flows should be tested. Unit testing is the testing of individual software units within the application, after completing an individual unit before integration. It is structural testing which depends on the structure of the application, is invasive in nature and tests a specific business process, application, and/or system configuration. It ensures that all unique paths of a business process perform accurately to the documented specification with clearly defined inputs and output.

8.3.2 Functional Test

Functional testing provides a systematic validation that the functions have been tested based on requirements stated in business and technical requirements documents, system documentation, and user manuals.

The key areas covered by functional tests are:

Valid Input : Valid inputs must be successfully tested and accepted.

Invalid Input : Invalid inputs must be rejected.

Function : Functions should be invoked.

Output : Outputs should be validated.

Systems/Procedures: The systems/procedures interface should be invoked.

8.3.3 System Test

System testing makes sure that the complete integrated software system satisfies requirements. This type of test verifies a configuration for ensuring predicted and known results. The configuration oriented system integration test is an example of system testing. This is driven by process descriptions and flow, focusing on pre-process links.

8.3.4 Performance Test

Performance testing makes sure that the output generated is done according to time restrictions and the amount of time needed by the system for processing and compilation, responding to the user's request.

8.3.5 Integration Testing

Integration testing of software is the incremental testing of the two or more integrated software components in order to generate failures resulting from interface problems. The main task of integration testing is to verify if the components and software applications, i.e. component interactions in software systems and application interactions in companies, occur without any problem.

8.3.6 User acceptance testing

Performing user acceptance testing is an important step of any development process, which involves the participation of the end user. User acceptance testing also makes sure that the system is able to fulfill the functional requirement.

Acceptance testing for data synchronization: → Acknowledgements will be sent by sender node after packets are received by destination node .

→ Route add operation should be done only when there is route request to be performed .

→ Status of nodes information will be done automatically during cache update process

8.3.7 Develop the test

plan Project can always be broken down into units, which can be further worked on in detail for their processing. The testing strategies can then be planned for all these units individually. Unit testing allows you to find the defects in the component; therefore, the defects can be fixed.

Chapter 9

Future Enhancements

9.1 Future enhancements: Some of the possible future enhancements of the fire detection system include improvement of its accuracy, scalability, and adaptability. First, the system could be enhanced by developing machine learning algorithms capable of detecting other hazardous events like gas leaks and chemical spills. This would ensure that the system would be able to serve as an all-round hazard detection system, protecting different areas such as industrial sites and cities from various types of emergencies. In addition, the implementation of edge computing can ensure that the processing time will be greatly reduced, thus enhancing the speed of the process. With the help of edge computing, processing will take place directly on the edge device, making it possible to deliver real-time alerts without necessarily using cloud servers. The next possible enhancement relates to the use of drones or UAVs to detect fires across large and hard to reach regions like forests or industrial areas. Finally, the fire detection could incorporate a multi-modal sensor network with the inclusion of temperature sensors, gas sensors, and smoke detectors among others. Lastly, the algorithm can be tailored to function under various weather conditions by integrating the algorithms based on adaptive methods taking into consideration different environmental aspects including but not limited to fog, heavy rainfall or variations in lighting. This measure will make the system functional at all times, making it a reliable method of increasing safety measures and prevention.

CHAPTER 10 CONCLUSION AND REFERENCES

10.1 CONCLUSION

By making improvements like enhanced feature extraction, attention mechanisms, and better bounding box regression, the system becomes capable of accurately detecting flames and smoke. Generally, this project plays a key role in the creation of advanced fire detection systems that incorporate other features such as multi-hazard detection, edge computing and IoT connectivity. through the integration of features such as enhanced feature extraction, attention mechanisms, and better bounding box regression, the system becomes capable of accurately locating the position of flames and smoke in difficult circumstances. The ability of the system to

operate in real time and adapt to changing environments makes it useful in enhancing fire detection safety and preventive measures. The technology will prove useful in any environment where the fire hazard is present. Summary As can be seen from this project, it is evident that the YOLOv10 algorithm has the capability of improving fire detection systems in a way that is both effective and precise through its ability to effectively address issues such as clutters, different intensities of fires, and poor visibility among others.

10.2 REFERENCES

- [1] A. K. Jaiswal, P. Tiwari, S. Kumar, D. Gupta, A. Khanna, and J. J. P. C. Rodrigues, "Identifying pneumonia in chest X-rays: A deep learning approach," *Measurement*, vol. 145, pp. 511–518, Oct. 2019.
- [2] D. M. Gavrilu and V. Philomin, "Real-time object detection for 'smart' vehicles," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, Sep. 1999, pp. 87–93.
- [3] L. Wei, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision— ECCV*. Cham, Switzerland: Springer, 2016.
- [4] S. Chen, J. Hong, T. Zhang, J. Li, and Y. Guan, "Object detection using deep learning: Single shot detector with a refined feature-fusion structure," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, Aug. 2019, pp. 219–224.
- [5] L. Cui, R. Ma, P. Lv, X. Jiang, Z. Gao, B. Zhou, and M. Xu, "MDSSD: Multi-scale deconvolutional single shot detector for small objects," *Sci. China Inf. Sci.*, vol. 63, no. 2
- [6] P. Shamsolmoali, M. Zareapoor, E. Granger, J. Chanussot, and J. Yang, "Enhanced single-shot detector for small object detection in remote sensing images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2022, pp. 1716–1719.
- [7] D. Jiang, B. Sun, S. Su, Z. Zuo, P. Wu, and X. Tan, "FASDD: A feature fusion and spatial attention-based single shot detector for small object detection," *Electronics*, vol. 9, no. 9, p. 1536, Sep. 2020.

- [8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, arXiv:2004.10934.
- [9] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proc. IEEE Conf. Comput. Vis. Pattern Recogn. Yu, Y., Ma, Y., & Yoshie, O., "PP-YOLOv2: A Practical Object Detector," 2021. arXiv preprint arXiv:2104.10419.
- [10] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A., "You only look once: Unified, real-time object detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 779-788.
- [11] Redmon, J. & Farhadi, A., "YOLOv3: An incremental improvement," 2018. arXiv preprint arXiv:1804.0276
- [12] C. Chaoxia, W. Shang, and F. Zhang, "Information-guided flame detection based on faster R-CNN," IEEE Access, vol. 8, pp. 58923–58932, 2020.
- [13] H. Yar, Z. A. Khan, F. U. M. Ullah, W. Ullah, and S. W. Baik, "A modified YOLOv5 architecture for efficient fire detection in smart cities," Exp. Syst. Appl., vol. 231, Nov. 2023, Art. no. 120465.
- [14] M. N. Uddin, M. Sakibul Islam Sakib, S. Nawer, and R. T. Mohona, "Improved fire detection by YOLOv8 and YOLOv5 to enhance fire safety," in Proc. 26th Int. Conf. Comput. Inf. Technol. (ICIT), Dec. 2023, pp. 1–6.
- [15] Z. Dai, "Image flame detection method based on improved YOLOv3," IOP Conf. Ser., Earth Environ. Sci., vol. 693, Mar. 2021, Art. no. p. 012012.
- [16] Y.-Y. Qin, J.-T. Cao, and X.-F. Ji, "Fire detection method based on depthwise separable convolution and YOLOv3," Int. J. Autom. Comput., vol. 18, no. 2, pp. 300–310, Apr. 2021.
- [17] X. Wu, Y. Cao, X. Lu, and H. Leung, "Patchwise dictionary learning for video forest fire smoke detection in wavelet domain," Neural Comput. Appl., vol. 33, no. 13
- [18] F. Yang, Q. Xue, Y. Cao, X. Li, W. Zhang, and G. Li, 'Multi-temporal dependency handling in video smoke recognition: A holistic approach spanning spatial, short-term, and long-term perspectives', Exp. Syst. Appl., vol. 245, Jul. 2024, Art. no. 123081.
- [19] Y. Liu, Z. Shao, and N. Hoffmann, 'Global attention mechanism: Retain information to enhance channel-spatial interactions', 2021, arXiv:2112.05561.
- [20] Q. Hou, D. Zhou, and J. Feng, 'Coordinate attention for efficient mobile network design', in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2021, pp. 13713-13722.
- [21] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, 'ECA-net: Efficient channel attention for deep convolutional neural networks', in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 11531-11539.
- [22] S. Liu, D. Huang, and Y. Wang, 'Learning spatial fusion for single-shot object detection', 2019, arXiv:1911.09516.
- [23] X. Wang, T. Xiao, Y. Jiang, S. Shao, J. Sun, and C. Shen, 'Repulsion loss: Detecting pedestrians in a crowd', in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 7774-7783.
- [24] T. Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Proc. IEEE Conf. Comput. Vis. Pattern Recognit., July 2017, pp. 2117–2125.
- [25] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," Proc. IEEE Conf. Comput. Vis. Pattern Recognit., June 2018, pp. 8759–8768.
- [26] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," Proc. Eur. Conf. Comput. Vis. (ECCV), Cham, Switzerland: Springer, 2018, pp. 3–19.
- [27] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolution.