**FLAME AND SMOKE DETECTION ALGORITHM BASED ON ODCONVBS-YOLOV5S**

**ABSTRACT**

**ABSTRACT**

Real-time and accurate detection of flame and smoke is a critical condition for decreasing fire loss. Traditional flame and smoke detection algorithms have several shortcomings such as poor accuracy, a high miss rate, low detection efficiency, and a low detection rate of small objects.We proposes the detection algorithm of flame and smoke based on ODConvBS in YOLOv5s. Firstly, the ordinary convolutional blocks in the backbone network of YOLOv5s are replaced with ODConvBS to achieve the extraction of attentional features from the convolutional kernel. Secondly, Gnconv is introduced into Neck to improve the high-order spatial information extraction ability of the model. Then, the Shuffle Attention module is added at the end of the Neck to facilitate the fusion of different groups of features. Finally, the prediction part uses a SIOU loss function that can take into account the angle of the prediction frame vectors to accelerate the model convergence. When utilizing the self-made dataset of flame and smoke, the upgraded YOLOv5s model mAP grew. At the same time, the accuracy rate, the recall rate, and the detection speed increased respectively.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study  2.1 Economic feasibility  2.2 Technical feasibility  2.3 Social feasibility |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

**INTRODUCTION**

**1. INTRODUCTION**

Fire will inflict significant damage to human life and property in daily activities, as well as do harm to the healthy development of society. However, in the early stages of a fire disaster, the flame is easily extinguished. As a result, by detecting flame and smoke accurately and quickly, the loss caused by the fire may be minimized to sustain normal production. Early flame detection frequently collects flame and smoke data using different temperature sensors, smoke sensors, and photosensitive sensors to assess whether a fire has occurred. However, the installation position and effective range of the sensor, as well as the external light and ambient humidity, will have a significant impact on the detection accuracy of the flame and smoke. The task of object detection is to find and classify all target objects in a picture, which is one of the fundamental tasks in the computer vision field. At the current stage, object detection algorithms are divided into two categories: Twostage and Onestage. The twostage architecture generates pre-selected boxes that may contain objects to be detected and extracted by features and then conducts classification and regression localization. The one stage architecture does not need to generate pre-selected boxes and can extract features directly in the network to predict the classification and location of an object.

* 1. **OBJECTIVE**

One essential requirement for reducing fire loss is the timely and precise detection of smoke and flames. Several drawbacks of traditional flame and smoke detection algorithms include low precision, a high miss rate, low detection efficiency, and a low rate of small object detection.

* 1. **PROBLEM STATEMENT**

Our suggestion is to use the ODConvBS-based flame and smoke detection algorithm in YOLOv5s. Initially, to extract attentional features from the convolutional kernel, ODConvBS replaces the regular convolutional blocks in the YOLOv5 backbone network. Second, Neck gains the ability to extract high-order spatial information by incorporating Gnconv. Then, at the end of the Neck, the Shuffle Attention module is added to help with the fusion of various feature groups. Ultimately, the prediction portion speeds up the model's convergence by using a SIOU loss function that can account for the angle of the prediction frame vectors.

**1.3 SOFTWARE REQUIREMENTS**

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**Platform –** In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer’s architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**APIs and drivers –** Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web browser –** Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

**1) Software : Anaconda**

**2) Primary Language : Python**

**3) Frontend Framework : Flask**

**4) Back-end Framework : Jupyter Notebook**

**5) Database : Sqlite3**

**6) Front-End Technologies : HTML, CSS, JavaScript and Bootstrap4**

**1.4 HARDWARE REQUIREMENTS**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1)Operating System : Windows Only**

**2)Processor : i5 and above**

**3)Ram : 8gb and above**

**4)Hard Disk : 25 GB in local drive**

**FEASIBILITY STUDY**

**2. FEASIBILITY STUDY**

**Feasibility Study**

A feasibility study evaluates a project's or system's practicality. As part of a feasibility study, the objective and rational analysis of a potential business or venture is conducted to determine its strengths and weaknesses, potential opportunities and threats, resources required to carry out, and ultimate success prospects. Two criteria should be considered when judging feasibility: the required cost and expected value.

**Types Of Feasibility Study**

A feasibility analysis evaluates the project’s potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

**1. Technical Feasibility**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn’t want to try to put Star Trek’s transporters in their building—currently, this project is not technically feasible.

**2. Economic Feasibility**

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent [project assessment](https://www.simplilearn.com/risk-assessment-project-management-article) and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

### **3. Legal Feasibility**

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws,[data protection](https://www.simplilearn.com/understanding-data-security-rar30-article) acts or social media laws. Let’s say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization’s ideal location isn’t zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

### **4. Operational Feasibility**

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization’s needs can be met by completing the project. Operational feasibility studies also examine how a [project plan](https://www.simplilearn.com/project-management-plans-in-project-environment-rar79-article) satisfies the requirements identified in the requirements analysis phase of system development.

### **5. Scheduling Feasibility**

This assessment is the most important for [project success](https://www.simplilearn.com/how-to-make-a-project-successful-article); after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

* Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
* Internal Corporate Constraints: Financial, Marketing, Export, etc.
* External Constraints: Logistics, Environment, Laws, and Regulations, etc.

**LITERATURE SURVEY**

**3. LITERATURE SURVEY**

**3.1 Object detection with deep learning: A review:**

<https://arxiv.org/abs/1807.05511>

**ABSTRACT:** Due to object detection's close relationship with video analysis and image understanding, it has attracted much research attention in recent years. Traditional object detection methods are built on handcrafted features and shallow trainable architectures. Their performance easily stagnates by constructing complex ensembles which combine multiple low-level image features with high-level context from object detectors and scene classifiers. With the rapid development in deep learning, more powerful tools, which are able to learn semantic, high-level, deeper features, are introduced to address the problems existing in traditional architectures. These models behave differently in network architecture, training strategy and optimization function, etc. In this paper, we provide a review on deep learning based object detection frameworks. Our review begins with a brief introduction on the history of deep learning and its representative tool, namely Convolutional Neural Network (CNN). Then we focus on typical generic object detection architectures along with some modifications and useful tricks to improve detection performance further. As distinct specific detection tasks exhibit different characteristics, we also briefly survey several specific tasks, including salient object detection, face detection and pedestrian detection. Experimental analyses are also provided to compare various methods and draw some meaningful conclusions. Finally, several promising directions and tasks are provided to serve as guidelines for future work in both object detection and relevant neural network based learning systems.

**3.2 Recent advances in deep learning for object detection:**

<https://arxiv.org/abs/1908.03673>

**ABSTRACT:** Object detection is a fundamental visual recognition problem in computer vision and has been widely studied in the past decades. Visual object detection aims to find objects of certain target classes with precise localization in a given image and assign each object instance a corresponding class label. Due to the tremendous successes of deep learning based image classification, object detection techniques using deep learning have been actively studied in recent years. In this paper, we give a comprehensive survey of recent advances in visual object detection with deep learning. By reviewing a large body of recent related work in literature, we systematically analyze the existing object detection frameworks and organize the survey into three major parts: (i) detection components, (ii) learning strategies, and (iii) applications & benchmarks. In the survey, we cover a variety of factors affecting the detection performance in detail, such as detector architectures, feature learning, proposal generation, sampling strategies, etc. Finally, we discuss several future directions to facilitate and spur future research for visual object detection with deep learning.

# 3.3 Multi-feature fusion based fast video flame detection:

<https://www.researchgate.net/publication/247161944_Multi-Feature_fusion_based_fast_video_flame_detection>

**ABSTRACT:** A video flame detection method based on the multi-feature fusion is presented in this paper. The temporal and spatial characteristics of flames, such as ordinary flame movement and color clues, a flame flickering detection algorithm is incorporated into the scheme to detect fires in color video sequences. An improved Gaussian mixture model method is firstly adopted to extract moving foreground objects from the still background of detection scenes; secondly, detected moving objects are then categorized into candidate and non-candidate flame regions by using a flame color filtering algorithm; finally, a flame flicker identification algorithm based on statistical frequency counting is used to distinguish true flames from fire-like objects in video images. Testing results show that the proposed algorithms are effective, robust and efficient. The processing rate of the flame detection method can achieve 24 fps with image size of 320 × 240 pixels on a PC with an AMD 2.04 GHz processor.

**3.4 A multi-disciplinary vision-based fire and smoke detection system:**

<https://ieeexplore.ieee.org/document/9297511>

**ABSTRACT:** Fire is the result of chemical reactions taking place between the oxygen and the rest of the atmosphere, which on large scales causes an ecological imbalance by giving out bi-products such as smoke. More than 90 Million cases of fire incidents have been reported since 1990 causing loss of life as well as endangered and vulnerable resources. The reason most of the time is unknown, but global warming is cited as one of the factors in the case of forest fires. Through this work, we develop a multi-disciplinary system that reports the traces of fire and smoke under any isolation. The system uses Computer Vision for analyzing and detecting fire or smoke in real-time through a Deep Learning algorithm and alerts back upon found.

**3.5 DeepSmoke: Deep learning model for smoke detection and segmentation in outdoor environments:**

<https://www.sciencedirect.com/science/article/abs/pii/S0957417421005662>

**ABSTRACT:** Fire disaster throughout the globe causes social, environmental, and economical damage, making its early detection and instant reporting essential for saving human lives and properties. Smoke detection plays a key role in early fire detection but majority of the existing methods are limited to either indoor or outdoor surveillance environments, with poor performance for hazy scenarios. In this paper, we present a Convolutional Neural Network (CNN)-based smoke detection and segmentation framework for both clear and hazy environments. Unlike existing methods, we employ an efficient CNN architecture, termed EfficientNet, for smoke detection with better accuracy. We also segment the smoke regions using DeepLabv3+, which is supported by effective encoders and decoders along with a pixel-wise classifier for optimum localization. Our smoke detection results evince a noticeable gain up to 3% in accuracy and a decrease of 0.46% in False Alarm Rate (FAR), while segmentation reports a significant increase of 2% and 1% in global accuracy and mean Intersection over Union (IoU) scores, respectively. This makes our method a best fit for smoke detection and segmentation in real-world surveillance settings.

**Table 3.1: Comparison Tabular Format for Literature Survey:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **TITLE & AUTHORS** | **METHODOLOGY** | **PROPOSED SYSTEM** | **CONS** | **CONCLUSION** |
| 1 | **TITLE:**  Fire Detection Method in Smart City Environments Using a Deep-Learning-Based Approach  **AUTHOR:**  Kuldoshbay Avazov ,Mukhriddin Mukhiddinov et.al.,  **LINK:** <https://www.mdpi.com/2079-9292/11/1/73>  **(2022)** | In this study, a fire detection system was developed using an enhanced YOLOv4-based convolutional neural network. The system utilized digital cameras and AI to detect small sparks and trigger an alarm within 8 seconds of a fire outbreak. The network was adapted for real-time monitoring on the Banana Pi M3 board, with an augmented training dataset and structural modifications to enhance accuracy in various weather conditions. | The proposed system employs an improved YOLOv4-based neural network to rapidly and accurately detect fires using digital cameras and AI. It can notify fire incidents within 8 seconds and operates effectively in different weather conditions. The system is designed to enhance fire safety in smart cities and urban areas. | 1. Limited information about the specific data augmentation and structural modifications.  2. No discussion on potential false alarms or challenges in a real-world environment. | The developed system, based on an improved YOLOv4 algorithm, demonstrates efficient fire detection in diverse weather conditions. While the study provides promising results, it should address scalability, potential false alarms, privacy, and cost implications for real-world implementation in smart cities. |
| 2 | **TITLE:**  Attention based CNN model for fire detection and localization in real-world images  **AUTHOR:**  Saima Majid , Fayadh Alenezi et.al.,  **LINK:** <https://www.sciencedirect.com/science/article/abs/pii/S0957417421014445>  **(2022)** | The study employs transfer learning with state-of-the-art convolutional neural networks (CNNs) using a real-world fire image dataset. The Grad-CAM method is utilized for fire visualization and localization, and an attention mechanism enhances network performance, particularly with the EfficientNetB0 model. The model's effectiveness is assessed using test accuracy and recall. | The system utilizes transfer learning with state-of-the-art CNNs and an attention mechanism for automated fire detection. It achieves a high test accuracy of 95.40% and a recall of 97.61, indicating strong reliability with minimal false negatives. | 1. Real-world challenges and environmental conditions for fire detection are not considered.  2. Privacy and ethical implications of using real-world fire images are not discussed. | The proposed system, based on transfer learning and attention mechanisms, demonstrates a high level of efficiency in fire detection with minimal false negatives. While promising, further consideration of dataset characteristics, computational requirements, real-world conditions, and ethical aspects would enhance the study's applicability and completeness. |
| 3 | **TITLE:**  A Small Target Forest Fire Detection Model Based on YOLOv5 Improvement  **AUTHOR:**  Zhenyang Xue ,Haifeng Lin et.al.,  **LINK:** <https://www.mdpi.com/1999-4907/13/8/1332>  **(2022)** | The study addresses small forest fire target detection using an enhanced YOLOv5 model. It incorporates modifications in the Backbone and Neck layers, utilizing Spatial Pyramid Pooling-Fast-Plus (SPPFP) and Convolutional Block Attention Module (CBAM). Transfer learning is applied due to a limited initial small-target forest fire dataset. | The proposed system enhances forest fire detection by adapting YOLOv5 with improved layers and attention mechanisms, targeting small fire areas in long-range images. It utilizes cameras as sensors and demonstrates a 10.1% improvement in mAP@0.5, indicating enhanced performance. | 1. Computational requirements and deployment feasibility are not addressed.  2. Ethical and privacy considerations related to forest fire image data usage are not explored. | The improved YOLOv5-based forest fire detection model shows promise with a significant mAP@0.5 improvement. Further research should consider dataset characteristics, real-world challenges, computational requirements, and ethical aspects for practical deployment in forest fire detection applications. |
| 4 | **TITLE:**  Fast forest fire smoke detection using MVMNet  **AUTHOR:**  Yaowen Hu  , Jialei Zhan et.al.,  **LINK:** <https://scholarsmine.mst.edu/cgi/viewcontent.cgi?article=3145&context=civarc_enveng_facwork>  **(2022)** | The proposed system, named MVMNet, introduces a multioriented detection approach for early forest fire smoke detection. It incorporates angle parameters, Softpool-spatial pyramid pooling (Soft-SPP), a value conversion-attention mechanism module (VAM), and hybrid non-maximum suppression methods (DIoU-NMS and Skew-NMS) to enhance smoke detection accuracy. | MVMNet introduces a novel multioriented detection method, utilizing angle parameters and Soft-SPP to handle varying image sizes. The VAM module extracts smoke color and texture features, while hybrid NMS methods improve detection accuracy. | 1. Fine-tuning and optimizing various parameters could be time-consuming.  2. The system's effectiveness in real-world, dynamic forest fire scenarios may require further validation. | MVMNet outperforms traditional methods with a high mAP of 78.92%, mAP50 of 88.05%, and a fast FPS of 122. This approach shows promise in early forest fire smoke detection, but real-world applicability and system complexity should be considered. |
| 5 | **TITLE:**  Adaptive linear feature-reuse network for rapid forest fire smoke detection model  **AUTHOR:**  Jiayong Li , Guoxiong Zhou et.al.,  **LINK:** <https://www.sciencedirect.com/science/article/abs/pii/S1574954122000334>  **(2022)** | The proposed ALFRNet aims to address issues in current smoke detection methods. It employs a Double Linear Feature-Reuse Module (DLFR Module) to reduce information loss, a Hybrid Attention-Guided Module (HAG Module) for enhanced feature emphasis, and an Adaptive Depthwise Convolution Module (ADC Module) for small smoke target recognition. Cluster NMS (CNMS) is used for more accurate bounding boxes. An IoT-based system is also developed. | ALFRNet integrates DLFR and HAG Modules to improve feature extraction, ADC Module for small smoke target recognition, and CNMS for precise bounding boxes. It outperforms mainstream methods with 87.26% mAP50 at 43 FPS on NVIDIA TITAN Xp, offering faster speed, higher accuracy, and improved detection accuracy. | 1. The system may demand significant computational resources.  2. Performance may be sensitive to the quality and quantity of training data. | ALFRNet excels in forest fire smoke detection, achieving 87.26% mAP50 at 43 FPS, surpassing mainstream methods in speed, accuracy, and precise bounding box localization. While promising, real-world deployment and computational requirements should be considered. |
| 6 | **TITLE:**  Light-FireNet: an efficient lightweight network for fire detection in diverse environments  **AUTHOR:**  Otabek Khudayberdiev, Jiashu Zhang et.al.,  **LINK:** <https://link.springer.com/article/10.1007/s11042-022-12552-5>  **(2022)** | Light-FireNet employs lightweight convolution mechanisms inspired by H-Swish and a novel architectural design. It utilizes deep learning for fire and smoke detection, focusing on optimizing model size and performance accuracy. | Light-FireNet is a cost-effective and lightweight fire and smoke detection system, with 32% fewer parameters than AlexNet and 3.03 MB lighter than MobileNetV2. It achieves a superior accuracy of 97.83%, outperforming most existing fire detection techniques. | 1. May require further fine-tuning for different real-world scenarios.  2. Potential sensitivity to variations in lighting and environmental conditions. | Light-FireNet offers an efficient solution for fire and smoke detection with reduced model size and improved accuracy. While it outperforms existing techniques, it may need adaptation for challenging real-world conditions and data availability constraints. |
| 7 | **TITLE:**  UFS-Net: A unified flame and smoke detection method for early detection of fire in video surveillance applications using CNNs  **AUTHOR:**  Ali Hosseini , Mahdi Hashemzadeh et.al.,  **LINK:** <https://www.sciencedirect.com/science/article/abs/pii/S1877750322000606>  **(2022)** | The research introduces a unified flame and smoke detection system called "UFS-Net" based on deep learning. It utilizes a tailored convolutional neural network architecture to classify video frames into eight classes, enhancing fire hazard identification. A decision module with a voting scheme is applied to improve reliability. Training and evaluation are conducted on the "UFS-Data" dataset and other benchmark datasets. | UFS-Net" is an efficient deep learning-based solution for fire and smoke detection in video frames. It classifies frames into eight categories, enhancing fire hazard recognition. The system employs a decision module for increased reliability and is trained on a rich dataset, "UFS-Data," comprising both real and artificial images. | 1. Deep learning-based systems like UFS-Net can demand significant computational resources.  2. The system's performance is heavily reliant on the quality and diversity of the training data. | The research presents UFS-Net as a robust solution for fire and smoke detection in video frames. Extensive experimentation and comparisons with state-of-the-art methods validate its high performance. However, challenges related to resource requirements, data quality, false positives, and practical deployment need to be addressed for wider adoption in surveillance systems. |
| 8 | **TITLE:**  Traffic Sign Detection via Improved Sparse R-CNN for Autonomous Vehicles  **AUTHOR:**  Tianjiao Liang  , Bao et.al.,  **LINK:** <https://www.hindawi.com/journals/jat/2022/3825532/>  **(2022)** | The study introduces an enhanced sparse R-CNN for traffic sign detection. It incorporates a coordinate attention block with ResNeSt, creates a feature pyramid, and employs tailored data augmentation techniques for complex traffic scenarios. A novel traffic sign dataset is introduced, and two modules, SAA and DTA, are designed to enhance algorithm robustness. | The proposed system is an improved sparse R-CNN for traffic sign detection. It integrates coordinate attention, ResNeSt, and a feature pyramid for accurate detection. Unique data augmentation methods are used, and two modules, SAA and DTA, enhance algorithm robustness, demonstrating effectiveness on traffic sign datasets and real on-road testing. | 1. Computational complexity may increase due to added attention blocks and feature pyramid.  2. Model training and dataset creation can be time-consuming and resource-intensive. | The study presents an enhanced sparse R-CNN for traffic sign detection, showcasing improved accuracy and robustness. While there are computational and resource challenges, the methodology and dataset offer promising results for autonomous vehicle applications in real traffic scenarios. |
| 9 | **TITLE:**  A novel lightweight real-time traffic sign detection  integration framework based on YOLOv4  **AUTHOR:**  Yang Gu , Bingfeng Si et.al.,  **LINK:** <https://pubmed.ncbi.nlm.nih.gov/35455150/>  **(2022)** | The study employs an enhanced sparse R-CNN by integrating coordinate attention with ResNeSt, constructing a feature pyramid, and employing specialized data augmentation. A unique traffic sign dataset is introduced, and self-adaption augmentation (SAA) and detection time augmentation (DTA) modules are designed to improve algorithm robustness. | The proposed system enhances traffic sign detection accuracy and robustness. It utilizes advanced techniques and a purpose-built dataset, along with SAA and DTA modules for algorithm resilience. | 1. Increased computational complexity.  2. Resource-intensive model training and dataset creation.  3. Real-world scene variability challenges. | The study's enhanced sparse R-CNN demonstrates improved accuracy and robustness for traffic sign detection. While computational and resource challenges exist, the approach and dataset hold promise for real-world autonomous vehicle applications. |
| 10 | **TITLE:**  Improved YOLOv5 network for real-time multi-scale traffic sign detection  **AUTHOR:**  Junfan Wang, Yi Chen et.al.,  **LINK:**  <https://link.springer.com/article/10.1007/s00521-022-08077-5>  **(2022)** | The proposed approach, AF-FPN, addresses the challenge of multi-scale traffic sign detection in unmanned driving systems. It incorporates an Adaptive Attention Module (AAM) and Feature Enhancement Module (FEM) to enhance feature consistency and representation in a feature pyramid network. AF-FPN replaces the YOLOv5 feature pyramid, improving multi-scale target detection while maintaining real-time performance. | AF-FPN is an enhanced feature pyramid model designed to improve multi-scale traffic sign detection in the YOLOv5 network. It uses AAM and FEM to reduce information loss and enhance feature representation, ensuring accurate detection. Additionally, an automatic data augmentation method enriches the dataset, enhancing model robustness for practical scenarios. | 1. Resource Requirements: Real-time implementation might necessitate powerful hardware.  2. The model's performance heavily relies on the quality and diversity of the dataset. | AF-FPN, with AAM and FEM, enhances multi-scale traffic sign detection within the YOLOv5 framework. It balances accuracy and real-time capabilities. Experimental results on the TT100K dataset demonstrate its universality and superiority compared to existing methods, promising advancements in unmanned driving systems. |

**SYSTEM ANALYSIS**

**4. SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

In literature they introduced the development of a multi-disciplinary vision-based fire and smoke detection system using computer vision and deep learning algorithms. The system aims to analyze and detect fire or smoke in real-time by analyzing the environment through cameras and applying a deep learning algorithm. The system uses a Convolutional Neural Network (CNN) called FireDetectionNet to extract features from the real-time recordings and accurately detect the required features for fire and smoke detection. They collected fire and smoke contexted images from Google Images and created a dataset by combining them with a dataset containing generic images. They trained the CNN model with this dataset and achieved an reliable accuracy in detecting fire and smoke. Their system can detect the presence of fire but not smoke during the night.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

* The existing work's system is limited in its ability to detect smoke during the night. This is a significant drawback as fire incidents can occur at any time, and the inability to detect smoke at night could lead to delayed responses and potential dangers.
* The existing work employs a Convolutional Neural Network (CNN) called FireDetectionNet for feature extraction. While it achieved reliable accuracy for fire detection, the absence of smoke detection capabilities highlights a potential limitation in accurately identifying both fire and smoke instances in real-time.
* The existing work's dataset is created by combining fire and smoke contexted images from Google Images with a dataset containing generic images. This approach might introduce issues of bias and could limit the model's ability to generalize to various environments and scenarios, potentially leading to false positives or negatives.

# 4.2 Proposed System:

We proposes the detection algorithm of flame and smoke based on ODConvBS in YOLOv5s. Firstly, the ordinary convolutional blocks in the backbone network of YOLOv5s are replaced with ODConvBS to achieve the extraction of attentional features from the convolutional kernel. Secondly, Gnconv is introduced into Neck to improve the high-order spatial information extraction ability of the model. Then, the Shuffle Attention module is added at the end of the Neck to facilitate the fusion of different groups of features. Finally, the prediction part uses a SIOU loss function that can take into account the angle of the prediction frame vectors to accelerate the model convergence.

# 4.2.1 Advantages of proposed system:

1. Our work is capable of detecting both fire and smoke during nighttime scenarios, enhancing its practical utility and responsiveness.
2. Our work introduces the ODConvBS (Attentional Ordinary Differential Convolutional Block), which indicates a more advanced approach to feature extraction compared to the existing work's. This advanced feature extraction technique could lead to better discrimination between fire, smoke, and other objects in the scene.
3. The integration of Gnconv (Group Normalization Convolution) into the Neck of our work's model implies a higher-order spatial information extraction capability. This enhancement could result in more accurate feature representation and better discrimination of fire and smoke instances, especially in complex environments.
4. The addition of the Shuffle Attention module at the end of the Neck in our work suggests a more efficient and effective way of fusing different groups of features. This feature fusion improvement could contribute to better overall performance in distinguishing between fire, smoke, and other objects.
5. The SIOU loss function introduced in the prediction part of our work takes into account the angle of prediction frame vectors. This innovation indicates a higher level of precision and convergence in the model's training process, potentially leading to more accurate fire and smoke detection results

### **4.3 FUNCTIONAL REQUIREMENTS**

1. Data Collection

2. Image processing

3. Data augmentation

4. Training model

5. Final outcome

### **4.4 NON FUNCTIONAL REQUIREMENTS**

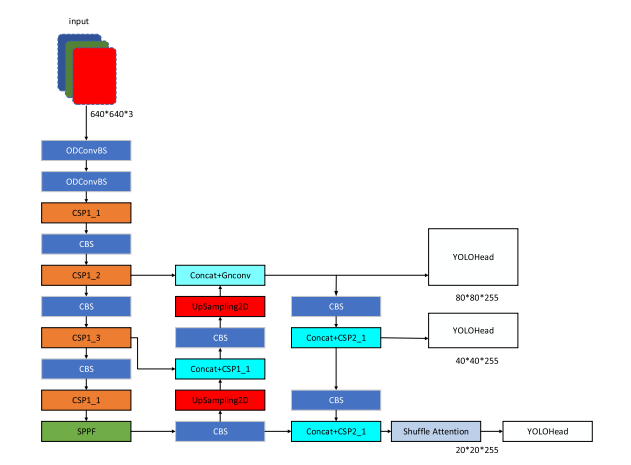
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**SYSTEM DESIGN**

**5. SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE:**

****

**Fig.5.1.1 System architecture**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

**Import libraries**

**VERIFY**

**NO PROCESS**

**Yes NO**

**Importing the dataset**

**Image processing**

**Loading the pretrained model**

**Data augmentation**

**Building the model in colab -** **SSD - FasterRCNN - Yolo V3 - Yolo V4 - Yolo V5s - Yolo V5x6 - TTA - Yolo V5 - ConBS - Yolo V5 - GhostNet - Yolo V8**

**Training the model**

**Signup & sign in**

**User input**

**End process**

**Final outcome**

**5.2 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**Use case diagram:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



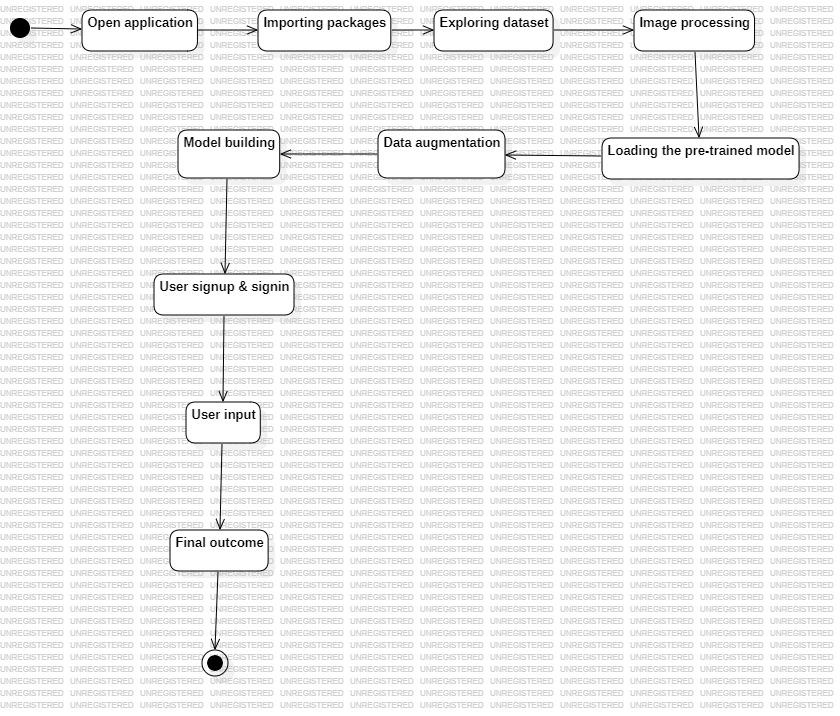
**Class diagram:**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



**Activity diagram:**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



**Sequence diagram:**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



**Collaboration diagram:**

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

****

**Component diagram:**

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



**Deployment diagram:**

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

****

**IMPLEMENTATION**

1. **IMPLEMENTATION**

MODULES:

* + Data exploration: using this module we will load data into system
  + Image processing: Using the module we will process of transforming an image into a digital form and performing certain operations to get some useful information from it.
  + Data augmentation: using this module used to address both the requirements, the diversity of the training data, and the amount of data
  + Model generation: Building the model in colab - - SSD - FasterRCNN - Yolo V3 - Yolo V4 - Yolo V5s - Yolo V5x6 - TTA - Yolo V5 - ConBS - Yolo V5 - GhostNet - Yolo V8.
  + User signup & login: Using this module will get registration and login
  + User input: Using this module will give input for prediction
  + Prediction: final predicted displayed

**Dataset Used :**

======

<https://roboflow.com/convert/labelbox-json-to-yolov5-pytorch-txt>

Extension

----------

In the base paper, the author mentioned to use different models for detecting the remote aerial satellite image using super yolo model and compare with other models, from which yolo-ConBS got 87% mAP,

However, we can further enhance the performance by exploring other techniques such as Yolo V5 - Ghost and YoloV8 for Detection got 0.90mPA,

With the above As an extension we can build the front end using flask framework for user testing with authentication.

**Algorithms:**

SSD - Single Shot MultiBox Detector. SSD is a single-stage object detection method that discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location.

FasterRCNN - Faster R-CNN is an object detection model that improves on Fast R-CNN by utilising a region proposal network (RPN) with the CNN model. The RPN shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals.

Yolo V8 - YOLOv8 is a new state-of-the-art computer vision model built by Ultralytics, the creators of YOLOv5. The YOLOv8 model contains out-of-the-box support for object detection, classification, and segmentation tasks, accessible through a Python package as well as a command line interface.

Yolo V5 - YOLOv5 uses a Convolutional Neural Network (CNN) backbone to form image features. These features are combined in the model neck and sent to the head. The model head then interprets the combined features to predict the class of an image.

YOLOV4 – YOLOv4 is a powerful and efficient object detection model that strikes a balance between speed and accuracy. Its use of unique features and bag of freebies techniques during training allows it to perform excellently in real-time object detection tasks.

YOLOV3 – YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. The YOLO machine learning algorithm uses features learned by a deep convolutional neural network to detect an object.

GhostNet - A GhostNet is a type of convolutional neural network that is built using Ghost modules, which aim to generate more features by using fewer parameters (allowing for greater efficiency). GhostNet mainly consists of a stack of Ghost bottlenecks with the Ghost modules as the building block.

**6.2 SAMPLE CODE:**

**# YOLOv5 🚀 by Ultralytics, GPL-3.0 license**

**"""**

**Run inference on images, videos, directories, streams, etc.**

**Usage:**

**$ python path/to/detect.py --source path/to/img.jpg --weights yolov5s.pt --img 640**

**"""**

**import argparse**

**import sys**

**import time**

**from pathlib import Path**

**import cv2**

**import numpy as np**

**import torch**

**import torch.backends.cudnn as cudnn**

**FILE = Path(\_\_file\_\_).absolute()**

**sys.path.append(FILE.parents[0].as\_posix()) # add yolov5/ to path**

**from models.experimental import attempt\_load**

**from utils.datasets import LoadStreams, LoadImages**

**from utils.general import check\_img\_size, check\_requirements, check\_imshow, colorstr, is\_ascii, non\_max\_suppression, \**

**apply\_classifier, scale\_coords, xyxy2xywh, strip\_optimizer, set\_logging, increment\_path, save\_one\_box**

**from utils.plots import Annotator, colors**

**from utils.torch\_utils import select\_device, load\_classifier, time\_sync**

**@torch.no\_grad()**

**def run(weights='yolov5s.pt', # model.pt path(s)**

**source='data/images', # file/dir/URL/glob, 0 for webcam**

**imgsz=640, # inference size (pixels)**

**conf\_thres=0.25, # confidence threshold**

**iou\_thres=0.45, # NMS IOU threshold**

**max\_det=1000, # maximum detections per image**

**device='', # cuda device, i.e. 0 or 0,1,2,3 or cpu**

**view\_img=False, # show results**

**save\_txt=False, # save results to \*.txt**

**save\_conf=False, # save confidences in --save-txt labels**

**save\_crop=False, # save cropped prediction boxes**

**nosave=False, # do not save images/videos**

**classes=None, # filter by class: --class 0, or --class 0 2 3**

**agnostic\_nms=False, # class-agnostic NMS**

**augment=False, # augmented inference**

**visualize=False, # visualize features**

**update=False, # update all models**

**project='runs/detect', # save results to project/name**

**name='exp', # save results to project/name**

**exist\_ok=False, # existing project/name ok, do not increment**

**line\_thickness=3, # bounding box thickness (pixels)**

**hide\_labels=False, # hide labels**

**hide\_conf=False, # hide confidences**

**half=False, # use FP16 half-precision inference**

**):**

**save\_img = not nosave and not source.endswith('.txt') # save inference images**

**webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(**

**('rtsp://', 'rtmp://', 'http://', 'https://'))**

**# Directories**

**save\_dir = increment\_path(Path(project) / name, exist\_ok=exist\_ok) # increment run**

**(save\_dir / 'labels' if save\_txt else save\_dir).mkdir(parents=True, exist\_ok=True) # make dir**

**# Initialize**

**set\_logging()**

**device = select\_device(device)**

**half &= device.type != 'cpu' # half precision only supported on CUDA**

**# Load model**

**w = weights[0] if isinstance(weights, list) else weights**

**classify, suffix = False, Path(w).suffix.lower()**

**pt, onnx, tflite, pb, saved\_model = (suffix == x for x in ['.pt', '.onnx', '.tflite', '.pb', '']) # backend**

**stride, names = 64, [f'class{i}' for i in range(1000)] # assign defaults**

**if pt:**

**model = attempt\_load(weights, map\_location=device) # load FP32 model**

**stride = int(model.stride.max()) # model stride**

**names = model.module.names if hasattr(model, 'module') else model.names # get class names**

**if half:**

**model.half() # to FP16**

**if classify: # second-stage classifier**

**modelc = load\_classifier(name='resnet50', n=2) # initialize**

**modelc.load\_state\_dict(torch.load('resnet50.pt', map\_location=device)['model']).to(device).eval()**

**elif onnx:**

**check\_requirements(('onnx', 'onnxruntime'))**

**import onnxruntime**

**session = onnxruntime.InferenceSession(w, None)**

**else: # TensorFlow models**

**check\_requirements(('tensorflow>=2.4.1',))**

**import tensorflow as tf**

**if pb: # https://www.tensorflow.org/guide/migrate#a\_graphpb\_or\_graphpbtxt**

**def wrap\_frozen\_graph(gd, inputs, outputs):**

**x = tf.compat.v1.wrap\_function(lambda: tf.compat.v1.import\_graph\_def(gd, name=""), []) # wrapped import**

**return x.prune(tf.nest.map\_structure(x.graph.as\_graph\_element, inputs),**

**tf.nest.map\_structure(x.graph.as\_graph\_element, outputs))**

**graph\_def = tf.Graph().as\_graph\_def()**

**graph\_def.ParseFromString(open(w, 'rb').read())**

**frozen\_func = wrap\_frozen\_graph(gd=graph\_def, inputs="x:0", outputs="Identity:0")**

**elif saved\_model:**

**model = tf.keras.models.load\_model(w)**

**elif tflite:**

**interpreter = tf.lite.Interpreter(model\_path=w) # load TFLite model**

**interpreter.allocate\_tensors() # allocate**

**input\_details = interpreter.get\_input\_details() # inputs**

**output\_details = interpreter.get\_output\_details() # outputs**

**int8 = input\_details[0]['dtype'] == np.uint8 # is TFLite quantized uint8 model**

**imgsz = check\_img\_size(imgsz, s=stride) # check image size**

**ascii = is\_ascii(names) # names are ascii (use PIL for UTF-8)**

**# Dataloader**

**if webcam:**

**view\_img = check\_imshow()**

**cudnn.benchmark = True # set True to speed up constant image size inference**

**dataset = LoadStreams(source, img\_size=imgsz, stride=stride, auto=pt)**

**bs = len(dataset) # batch\_size**

**else:**

**dataset = LoadImages(source, img\_size=imgsz, stride=stride, auto=pt)**

**bs = 1 # batch\_size**

**vid\_path, vid\_writer = [None] \* bs, [None] \* bs**

**# Run inference**

**if pt and device.type != 'cpu':**

**model(torch.zeros(1, 3, \*imgsz).to(device).type\_as(next(model.parameters()))) # run once**

**t0 = time.time()**

**for path, img, im0s, vid\_cap in dataset:**

**if onnx:**

**img = img.astype('float32')**

**else:**

**img = torch.from\_numpy(img).to(device)**

**img = img.half() if half else img.float() # uint8 to fp16/32**

**img = img / 255.0 # 0 - 255 to 0.0 - 1.0**

**if len(img.shape) == 3:**

**img = img[None] # expand for batch dim**

**# Inference**

**t1 = time\_sync()**

**if pt:**

**visualize = increment\_path(save\_dir / Path(path).stem, mkdir=True) if visualize else False**

**pred = model(img, augment=augment, visualize=visualize)[0]**

**elif onnx:**

**pred = torch.tensor(session.run([session.get\_outputs()[0].name], {session.get\_inputs()[0].name: img}))**

**else: # tensorflow model (tflite, pb, saved\_model)**

**imn = img.permute(0, 2, 3, 1).cpu().numpy() # image in numpy**

**if pb:**

**pred = frozen\_func(x=tf.constant(imn)).numpy()**

**elif saved\_model:**

**pred = model(imn, training=False).numpy()**

**elif tflite:**

**if int8:**

**scale, zero\_point = input\_details[0]['quantization']**

**imn = (imn / scale + zero\_point).astype(np.uint8) # de-scale**

**interpreter.set\_tensor(input\_details[0]['index'], imn)**

**interpreter.invoke()**

**pred = interpreter.get\_tensor(output\_details[0]['index'])**

**if int8:**

**scale, zero\_point = output\_details[0]['quantization']**

**pred = (pred.astype(np.float32) - zero\_point) \* scale # re-scale**

**pred[..., 0] \*= imgsz[1] # x**

**pred[..., 1] \*= imgsz[0] # y**

**pred[..., 2] \*= imgsz[1] # w**

**pred[..., 3] \*= imgsz[0] # h**

**pred = torch.tensor(pred)**

**# NMS**

**pred = non\_max\_suppression(pred, conf\_thres, iou\_thres, classes, agnostic\_nms, max\_det=max\_det)**

**t2 = time\_sync()**

**# Second-stage classifier (optional)**

**if classify:**

**pred = apply\_classifier(pred, modelc, img, im0s)**

**# Process predictions**

**for i, det in enumerate(pred): # detections per image**

**if webcam: # batch\_size >= 1**

**p, s, im0, frame = path[i], f'{i}: ', im0s[i].copy(), dataset.count**

**else:**

**p, s, im0, frame = path, '', im0s.copy(), getattr(dataset, 'frame', 0)**

**p = Path(p) # to Path**

**save\_path = str(save\_dir / p.name) # img.jpg**

**txt\_path = str(save\_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'\_{frame}') # img.txt**

**s += '%gx%g ' % img.shape[2:] # print string**

**gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # normalization gain whwh**

**imc = im0.copy() if save\_crop else im0 # for save\_crop**

**annotator = Annotator(im0, line\_width=line\_thickness, pil=not ascii)**

**if len(det):**

**# Rescale boxes from img\_size to im0 size**

**det[:, :4] = scale\_coords(img.shape[2:], det[:, :4], im0.shape).round()**

**# Print results**

**for c in det[:, -1].unique():**

**n = (det[:, -1] == c).sum() # detections per class**

**s += f"{n} {names[int(c)]}{'s' \* (n > 1)}, " # add to string**

**# Write results**

**for \*xyxy, conf, cls in reversed(det):**

**if save\_txt: # Write to file**

**xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() # normalized xywh**

**line = (cls, \*xywh, conf) if save\_conf else (cls, \*xywh) # label format**

**with open(txt\_path + '.txt', 'a') as f:**

**f.write(('%g ' \* len(line)).rstrip() % line + '\n')**

**if save\_img or save\_crop or view\_img: # Add bbox to image**

**c = int(cls) # integer class**

**label = None if hide\_labels else (names[c] if hide\_conf else f'{names[c]} {conf:.2f}')**

**annotator.box\_label(xyxy, label, color=colors(c, True))**

**if save\_crop:**

**save\_one\_box(xyxy, imc, file=save\_dir / 'crops' / names[c] / f'{p.stem}.jpg', BGR=True)**

**# Print time (inference + NMS)**

**print(f'{s}Done. ({t2 - t1:.3f}s)')**

**# Stream results**

**im0 = annotator.result()**

**if view\_img:**

**cv2.imshow(str(p), im0)**

**cv2.waitKey(1) # 1 millisecond**

**# Save results (image with detections)**

**if save\_img:**

**if dataset.mode == 'image':**

**cv2.imwrite(save\_path, im0)**

**else: # 'video' or 'stream'**

**if vid\_path[i] != save\_path: # new video**

**vid\_path[i] = save\_path**

**if isinstance(vid\_writer[i], cv2.VideoWriter):**

**vid\_writer[i].release() # release previous video writer**

**if vid\_cap: # video**

**fps = vid\_cap.get(cv2.CAP\_PROP\_FPS)**

**w = int(vid\_cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))**

**h = int(vid\_cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))**

**else: # stream**

**fps, w, h = 30, im0.shape[1], im0.shape[0]**

**save\_path += '.mp4'**

**vid\_writer[i] = cv2.VideoWriter(save\_path, cv2.VideoWriter\_fourcc(\*'mp4v'), fps, (w, h))**

**vid\_writer[i].write(im0)**

**if save\_txt or save\_img:**

**s = f"\n{len(list(save\_dir.glob('labels/\*.txt')))} labels saved to {save\_dir / 'labels'}" if save\_txt else ''**

**print(f"Results saved to {colorstr('bold', save\_dir)}{s}")**

**if update:**

**strip\_optimizer(weights) # update model (to fix SourceChangeWarning)**

**print(f'Done. ({time.time() - t0:.3f}s)')**

**def parse\_opt():**

**parser = argparse.ArgumentParser()**

**parser.add\_argument('--weights', nargs='+', type=str, default='./runs/train/exp4/weights/best.pt', help='model.pt path(s)')**

**parser.add\_argument('--source', type=str, default='data/images', help='file/dir/URL/glob, 0 for webcam')**

**parser.add\_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], help='inference size h,w')**

**parser.add\_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')**

**parser.add\_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')**

**parser.add\_argument('--max-det', type=int, default=1000, help='maximum detections per image')**

**parser.add\_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')**

**parser.add\_argument('--view-img', action='store\_true', help='show results')**

**parser.add\_argument('--save-txt', action='store\_true', help='save results to \*.txt')**

**parser.add\_argument('--save-conf', action='store\_true', help='save confidences in --save-txt labels')**

**parser.add\_argument('--save-crop', action='store\_true', help='save cropped prediction boxes')**

**parser.add\_argument('--nosave', action='store\_true', help='do not save images/videos')**

**parser.add\_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')**

**parser.add\_argument('--agnostic-nms', action='store\_true', help='class-agnostic NMS')**

**parser.add\_argument('--augment', action='store\_true', help='augmented inference')**

**parser.add\_argument('--visualize', action='store\_true', help='visualize features')**

**parser.add\_argument('--update', action='store\_true', help='update all models')**

**parser.add\_argument('--project', default='runs/detect', help='save results to project/name')**

**parser.add\_argument('--name', default='exp', help='save results to project/name')**

**parser.add\_argument('--exist-ok', action='store\_true', help='existing project/name ok, do not increment')**

**parser.add\_argument('--line-thickness', default=3, type=int, help='bounding box thickness (pixels)')**

**parser.add\_argument('--hide-labels', default=False, action='store\_true', help='hide labels')**

**parser.add\_argument('--hide-conf', default=False, action='store\_true', help='hide confidences')**

**parser.add\_argument('--half', action='store\_true', help='use FP16 half-precision inference')**

**opt = parser.parse\_args()**

**opt.imgsz \*= 2 if len(opt.imgsz) == 1 else 1 # expand**

**return opt**

**def main(opt):**

**print(colorstr('detect: ') + ', '.join(f'{k}={v}' for k, v in vars(opt).items()))**

**check\_requirements(exclude=('tensorboard', 'thop'))**

**run(\*\*vars(opt))**

**if \_\_name\_\_ == "\_\_main\_\_":**

**opt = parse\_opt()**

**main(opt)**

**SOFTWARE ENVIRONMENT**

**7. SOFTWARE ENVIRONMENT**

**YOLO:**

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

This article introduces readers to the YOLO algorithm for object detection and explains how it works. It also highlights some of its real-life applications.

### **Introduction to object detection**

Object detection is a phenomenon in [computer vision](https://www.section.io/engineering-education/computer-vision-straight-lines/) that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals.

This phenomenon seeks to answer two basic questions:

1. What is the object? This question seeks to identify the object in a specific image.
2. Where is it? This question seeks to establish the exact location of the object within the image.

Object detection consists of various approaches such as [fast R-CNN](https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97?gi=fea1a85170b6), [Retina-Net](https://developers.arcgis.com/python/guide/how-retinanet-works/), and [Single-Shot MultiBox Detector (SSD)](https://iq.opengenus.org/single-shot-detection-ssd-algorithm/). Although these approaches have solved the challenges of data limitation and modeling in object detection, they are not able to detect objects in a single algorithm run. **YOLO algorithm** has gained popularity because of its superior performance over the aforementioned object detection techniques.

### **What is YOLO?**

YOLO is an abbreviation for the term ‘You Only Look Once’. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

### **Why the YOLO algorithm is important**

YOLO algorithm is important because of the following reasons:

* **Speed:** This algorithm improves the speed of detection because it can predict objects in real-time.
* **High accuracy:** YOLO is a predictive technique that provides accurate results with minimal background errors.
* **Learning capabilities:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

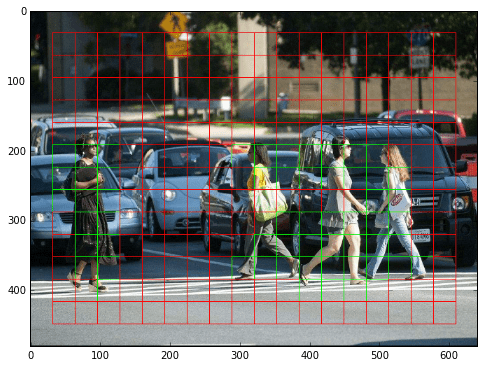
### **How the YOLO algorithm works**

YOLO algorithm works using the following three techniques:

* Residual blocks
* Bounding box regression
* Intersection Over Union (IOU)

#### Residual blocks

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.



[Image Source](https://www.guidetomlandai.com/assets/img/computer_vision/grid.png)

In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

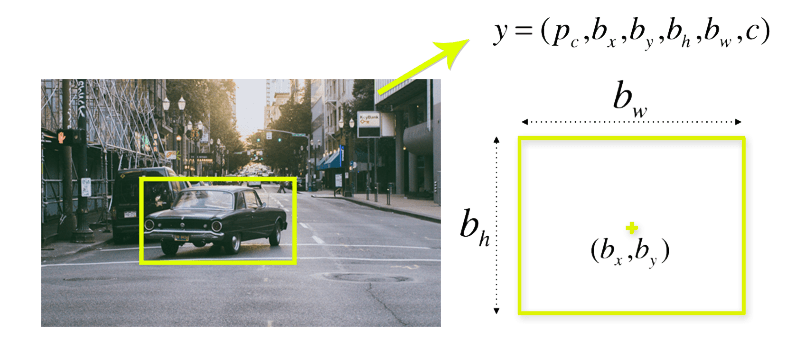
#### Bounding box regression

A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

* Width (bw)
* Height (bh)
* Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
* Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.



[Image Source](https://appsilondatascience.com/assets/uploads/2018/08/bbox-1.png)

YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

#### Intersection over union (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

### **Applications of YOLO**

YOLO algorithm can be applied in the following fields:

* **Autonomous driving:** YOLO algorithm can be used in autonomous cars to detect objects around cars such as vehicles, people, and parking signals. Object detection in autonomous cars is done to avoid collision since no human driver is controlling the car.
* **Wildlife:** This algorithm is used to detect various types of animals in forests. This type of detection is used by wildlife rangers and journalists to identify animals in videos (both recorded and real-time) and images. Some of the animals that can be detected include giraffes, elephants, and bears.
* **Security:** YOLO can also be used in security systems to enforce security in an area. Let’s assume that people have been restricted from passing through a certain area for security reasons. If someone passes through the restricted area, the YOLO algorithm will detect him/her, which will require the security personnel to take further action.

**PYTHON LANGUAGE:**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

## Features in Python:

There are many features in Python, some of which are discussed below as follows:

### **1. Free and Open Source**

[Python](https://www.geeksforgeeks.org/python-programming-language/)language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. [Download Python](https://www.python.org/downloads/) Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### **2. Easy to code**

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of [Python is Object-Oriented programming](https://www.geeksforgeeks.org/python-oops-concepts/). Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Extensible feature**

Python is an **Extensible** language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an integrated language**

Python is also an integrated language because we can easily integrate Python with other languages like C, [C++](http://www.geeksforgeeks.org/c-plus-plus/), etc.

### **11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, [Java](https://www.geeksforgeeks.org/java/), etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **byte code**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as [regular expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-testing/), web browsers, etc.

### **13. Dynamically Typed Language**

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

### **14. Frontend and backend development**

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### 15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

**LIBRARIES/PACKGES:-**

**Tensor flow**

Tensor Flow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provide a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**SYSTEM TESTING**

**8. SYSTEM TESTING**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

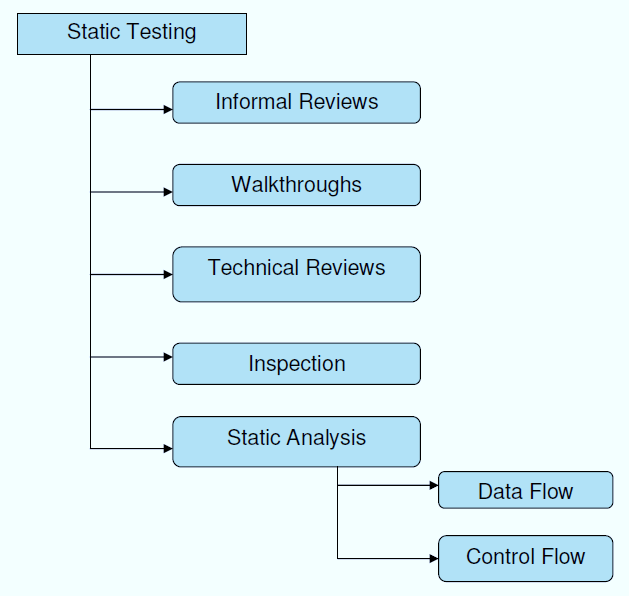
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**8.1Software Testing Strategies:**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

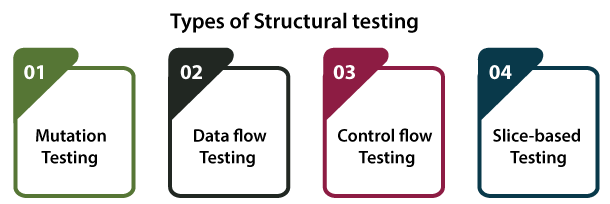
Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



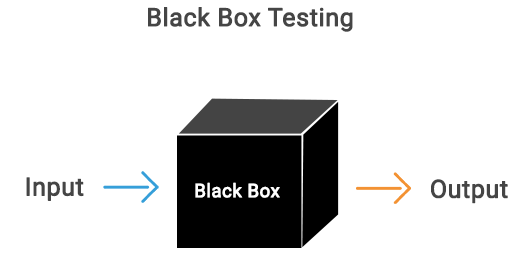
**Structural Testing:**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software’s structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system’s behavior by comparing the tests’ outcomes with the results of previous iterations (control flow testing).



**Behavioral Testing:**

The final stage of testing focuses on the software’s reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user’s point of view. QA engineers usually have some specific information about a business or other purposes of the software (‘the black box’) to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.



**8.2 TEST CASES:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User sign in | User get login into the application | There is no process |
| 3 | Enter input for prediction | Prediction result displayed | There is no process |

**SCREENS**

1. **SCREENSHOTS**

SCREENS:

**CONCLUSION**

**10. CONCLUSION**

We proposes the use of YOLOv5s based on the ODConvBS model to recognize flames and smoke, further improving detection efficiency and accuracy. Therefore, the work suggests using an ODConvBS-based YOLOv5s method to recognize flame and smoke to increase detection efficiency and accuracy. The following are the main contributions of our work such as Propose the ODConvBS module based on Omni-dimensional dynamic convolution, Use Gnconv-FPN to improve the feature pyramid structure of the original network, Add the SA attention mechanism to improve the model’s ability to extract image information, Improve the prototype loss calculation by using the SIOU loss function. The testing findings reveal that the In the case of the best settings, the deep learning algorithm of upgraded YOLOv5s has a map of 87.6%, which is much better than the original YOLOv5s, SSD, Faster R-CNN, and other algorithms. When compared to the original model, the updated model may consider detection efficiency while enhancing accuracy. In the future, we will explore lightweight backbone networks and new attention mechanisms to simplify the network, further improve detection efficiency and accuracy, and achieve a flame and smoke detection algorithm that balances both high precision and speed, thereby enabling real-time application in industrial scenarios.

**BIBILOGRAPHY**

**11. REFERENCES**

[1] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, ‘‘Object detection with deep learning: A review,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[2] X. Wu, D. Sahoo, and S. C. Hoi, ‘‘Recent advances in deep learning for object detection,’’ Neurocomputing, vol. 396, pp. 39–64, Jul. 2020.

[3] J. Chen, Y. He, and J. Wang, ‘‘Multi-feature fusion based fast video flame detection,’’ Building Environ., vol. 45, no. 5, pp. 1113–1122, May 2010.

[4] A. Pawar, ‘‘A multi-disciplinary vision-based fire and smoke detection system,’’ in Proc. 4th Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA), Nov. 2020, pp. 900–904.

[5] S. Khan, K. Muhammad, and T. Hussain, ‘‘Deepsmoke: Deep learning model for smoke detection and segmentation in outdoor environments,’’ Expert Syst. Appl., vol. 182, Nov. 2021, Art. no. 115125.

[6] G. Lin, Y. Zhang, G. Xu, and Q. Zhang, ‘‘Smoke detection on video sequences using 3D convolutional neural networks,’’ Fire Technol., vol. 55, no. 5, pp. 1827–1847, Sep. 2019.

[7] L. Wang, W. Li, and W. Li, ‘‘Appearance-and-relation networks for video classification,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 1430–1439.

[8] P. Li and W. Zhao, ‘‘Image fire detection algorithms based on convolutional neural networks,’’ Case Stud. Thermal Eng., vol. 19, Jun. 2020, Art. no. 100625.

[9] S. Ren, K. He, and R. Girshick, ‘‘Towards real-time object detection with region proposal networks,’’ 2019, arXiv:1506.01497.

[10] J. Dai, Y. Li, and K. He, ‘‘R-FCN: Object detection via region-based fully convolutional networks,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 29, 2016, pp. 1–11.

[11] W. Liu, D. Anguelov, and D. Erhan, ‘‘SSD: Single shot multibox detector,’’ in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 21–37.

[12] J. Redmon and A. Farhadi, ‘‘YOLOv3: An incremental improvement,’’ 2018, arXiv:1804.02767.

[13] S. Saponara, A. Elhanashi, and A. Gagliardi, ‘‘Real-time video fire/smoke detection based on CNN in antifire surveillance systems,’’ J. Real-Time Image Process., vol. 18, no. 3, pp. 889–900, 2021.

[14] M. Torabian, H. Pourghassem, and H. Mahdavi-Nasab, ‘‘Fire detection based on fractal analysis and spatio-temporal features,’’ Fire Technol., vol. 57, pp. 2583–2614, May 2021.

[15] K. Avazov, M. Mukhiddinov, F. Makhmudov, and Y. I. Cho, ‘‘Fire detection method in smart city environments using a deep-learning-based approach,’’ Electronics, vol. 11, p. 73, Dec. 2022