بسم الله الرحمن الرحيم

Sudan University of Science and Technology
College of Engineering
School of Electronics Engineering

Fire Detection based Image Recognition Using Machine Learning

كشف الحريق بواسطة الصور باستخدام التعلم الآلي

A Research Submitted in Partial fulfillment for the Requirements of the Degree of B.Sc. (Honors) in Electronics Engineering

Prepared By:

1. Alfatih Abdelbagi Adam Suliman
2. Eissa Khmeas Eissa Adam
3. Ismail Shaibo Abdalla Wadi
4. Mohamed Hamid Hary Hamid
5. Mohammedalgasem Eltalib Mohammed Adam

Supervised By:

Prof. Dr. Rashid A Saeed

October 2024

DEDICATION

We dedicate this report and great

achievement

to our parents, sisters and brothers🖤🖤.

ACKNOWLEDGMENT

We would like to express our gratitude and appreciation to all those who gave us the opportunity to complete this report. I extend my special thanks and appreciation to our supervisors and our professor **Prof. Dr. Rashid A SAEED** who stood as a solid dam in front of every obstacle we faced and kept encouraging us all the time and never got tired of devoting his time for us until the last letter in this report.

Our thanks extend to the College of Engineering in general and the School of Electronics Engineering in particular.

We would also like to deepen our thanks to those who were always and forever a link between us and the administration for their efforts day and night and for their efforts. Thank you, academic secretaries.

ABSTRACT

Fire detection is vital for safeguarding both the environment and human life, however, conventional fire detection methods, such as temperature and smoke sensors, often face inherent limitations. The machine learning models, specifically convolutional neural networks (CNNs), are commonly used for processing images to enhance the accuracy and efficiency of fire detection, enabling these systems to detect fire in real-time. We propose the YOLOv8 (You Only Look One Version8) model it's capable of analyzing high-resolution images of both fires and non-fire scenarios from diverse sources. The Fire dataset is typically split into 80% training, 15% validation, and 5% testing to ensure the model learns effectively. The accuracy of bounding box predictions during training the box loss starts at around 0.54 and progressively decreases to about 0.22, and the Recall starts at around 0.94 and steadily increases to about 0.99, the loss decreases from about 0.88 to 0.76, showing that the model is making fewer errors in bounding box regression on the validation data, a significant decrease in loss values, suggesting effective learning and optimization, while high and improving precision and recall demonstrate its strong detection performance. The consistently increasing mAP (mean Average Precision) values further indicate that the model is accurately predicting bounding boxes and classifying fire objects. Future work will focus on improving data, models, and integrating this approach with systems like firefighting control.

المستخلص

اكتشاف الحرائق أمر بالغ الأهمية لحماية البيئة وحياة الإنسان، ومع ذلك، فإن الطرق التقليدية لاكتشاف الحرائق، مثل مستشعرات الحرارة والدخان، غالبًا ما تواجه قيودًا جوهرية. تُستخدم نماذج التعلم الآلي، وخاصة الشبكات العصبية التلافيفية(CNNs) ، بشكل شائع لمعالجة الصور لتحسين دقة وكفاءة اكتشاف الحرائق، مما يمكّن هذه الأنظمة من اكتشاف الحرائق في الوقت الفعلي. نقترح استخدام نموذج YOLOv8 (You Only Look Once Version 8)، القادر على تحليل الصور عالية الدقة لكل من سيناريوهات الحرائق وغير الحرائق من مصادر متنوعة. باستخدام هذا الخوارزمية، يمكن للنموذج تقديم توقعات دقيقة للغاية تم تقسيم مجموعة بيانات الحرائق إلى 80٪ للتدريب، و15٪ للتحقق من الصحة، و5٪ للاختبار لضمان تعلم النموذج بفعالية. خلال عملية التدريب، تبدأ دقة تنبؤات الصناديق (bounding box predictions)مع خسارة الصندوق بحوالي 0.54 وتنخفض تدريجيًا إلى حوالي 0.22. تبدأ نسبة التذكير (Recall) بحوالي 0.94 وتزداد بثبات لتصل إلى حوالي 0.99، بينما تنخفض الخسارة الإجمالية من حوالي 0.88 إلى 0.76، مما يدل على أن النموذج يرتكب أخطاء أقل في تحديد الصناديق على بيانات التحقق. تشير الانخفاضات الكبيرة في قيم الخسارة إلى تعلم فعال وتحسين ملحوظ، بينما تُظهر الدقة العالية والتحسن المستمر في التذكير قوة أداء النموذج في الكشف. كما تشير القيم المتزايدة باستمرار لمتوسط الدقة (mAP – mean Average Precision)إلى أن النموذج يتنبأ بدقة بالمربعات ويصنف الأجسام المتعلقة بالحرائق، مما يجعله حلاً قويًا لمهام اكتشاف الحرائق. ستتركز الأعمال المستقبلية على تحسين البيانات والنماذج ودمج هذا النهج مع أنظمة مثل التحكم في مكافحة الحرائق.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
VR	Virtual Reality
RGB	Red Green Blue
R-FCN	Region-based Fully Convolution Networks
ML	Machine learning
CNNs	Convolutional Neural Networks
SVMs	Support Vector Machines
CVFD	Computer Vision-based fire detection
RPN	Region Proposal Network
R-FCN	Region-based Fully Convolutional Network
SSD	Single Shot MultiBox Detect
CCTV	Closed Circuit Television
YOLO v3	You Only Look Once Version-3
YOLO v8	You Only Look Once Version-8
ROI	Region of interest
OMT	Optical Mass Transport
IDE	Integrated Development Environment
NSD	Non-Smooth Data
IOT	Internet-of-Things
HSV	(Hue, Saturation, Value)
GPU	Graphics processing Unit

CHAPTER ONE

INTRODUCTIOIN

CHAPTER ONE

INTRODUCTIOIN

1.1 Overview

Fire detection is crucial for both environmental protection and human safety. Techniques based on image information tend to be more effective than those relying on multisensory wireless networks or sensor systems. These image-based methods enhance the accuracy, timeliness, and reliability of fire alarms [1]. Fire detection technologies can be categorized by various factors: (i) the type of detection, such as fire or smoke; (ii) the spectral range of the camera; and (iii) the system's operational range.

Traditional fire detectors, such as temperature sensors, smoke detectors, and optical detectors, often come with inherent limitations or restrictions due to their detection principles and system designs [2]. Since flames and smoke exhibit distinctive colors, textures, shapes, and other visual characteristics, there is growing interest in leveraging computer vision to enhance fire detection efficiency. For instance, video flame detection technology, which relies on image processing, is being explored. In most fire scenarios, both flames and smoke are present. This video-based fire detection method involves analyzing video footage to identify flames or smoke and perform real-time monitoring of fire scenes. The non-contact nature of this approach relies on computers to process

images and differentiate potential fire threats by analyzing and extracting the visual features of flames and smoke.

With the rapid advancement of computer technology and the widespread use of cameras, machine vision technology—driven by artificial intelligence (AI) and digital image processing—has found applications in various fields such as face detection, fire detection, object measurement, and surface defect detection. Machine Vision, an interdisciplinary field combining AI and digital image processing, is increasingly being used to replace manual operations and measurements with intelligent automation [3].

A typical machine vision system comprises a camera and an image processing unit. The camera captures images, which are then analyzed by visual recognition algorithms on a computer, and the image processing unit provides the recognition results through a terminal. Today, machine vision is a critical skill for professionals in image and video processing and a key subject in intelligent manufacturing, computer science, and related disciplines. The growing demand for AI expertise, particularly in natural language processing and digital image processing, reflects the expanding role of machine vision in modern technology.

Fire detection using image recognition with machine learning involves employing algorithms to analyze images or video frames and identify signs of fire or smoke. By training models on large datasets of fire and non-fire images, machine learning techniques can learn to distinguish between flames and other objects [4].

1.2 Problem Statement

As the frequency of wildfires and urban fires continues to rise, posing devastating impacts on the environment and human life, early detection of fires has become crucial for effective response and minimizing potential damage. Traditional detection methods often suffer from limitations in accuracy and efficiency, leading to delays in response and exacerbating the situation.

With the advent of machine learning and image recognition technologies, it has become possible to leverage these advancements to improve the efficiency and effectiveness of fire detection systems. However, challenges remain in developing a reliable model that is applicable across diverse real-world environments. These challenges include variations in generated images, increasing scene details, and fluctuating environmental factors that can impact detection processes.

In this study, the authors put forward a solution based on a CNN to detect forest fires from surveillance images for early detection. Using the fire and non-fire image datasets, the CNN is trained then provides a high level of fire detection in real-time monitoring systems [5].

This paper presents the proposal of a forest fire detection system using WSNs for early fire signs identification. It uses several sensors to observe elements of environment such as heat, moisture and concentrations of gases. The system analyzes this data to identify fire signs, and stimulates alert and early intervention procedures in case of potential fire occurrences [6].

Therefore, our study aims to explore and develop a machine learning-based model that focuses on utilizing image recognition techniques for accurate fire detection across various scenarios. This

endeavor seeks to provide in Echotive solutions that contribute to reducing detection time and alerting emergency response teams promptly.

1.3 The Aim and Objectives

The aim of research is to enhance the accuracy and efficiency of fire detection systems.

And the objectives of work summarized as following:

1. To quickly identify and alert authorities about the presence of fire or smoke, minimizing potential damage and ensuring timely response.

2. To reduce false positives and negatives compared to traditional fire detection methods, thereby improving reliability.

3. To automate the monitoring process, which can operate continuously and at scale, providing real-time analysis without human intervention.

4. To potentially lower costs associated with fire detection and monitoring through the use of machine learning technology rather than traditional methods.

5. To develop systems that can function in various environments and conditions by training on diverse image datasets, thus enhancing robustness.

1.4 Scope and Limitations

Prototype, limited to develop systems that can identify fire in images or video feeds in real time, enhancing early warning and prevention systems.

And typically involves computer vision techniques, machine learning algorithms and image processing methods. Using large datasets of fire and non-fire images to train and validate machine learning models.

1.5 Methodology

Fire detection methods using existing features use fire flame and smoke images of various sizes to train. Therefore, this also affects the detection rate since the result of feature extraction is not constant. Another approach to fire detection is to use machine learning algorithms.

Machine learning algorithms learn the shape or representation of data to detect fire and smoke in image.

The project starts by collection dataset of images containing fires and non-fires from various sources, Integrate the model and define the architecture of the machine learning model using python frameworks, split the dataset into training, validation, and test sets.

To approach this method there are several steps, from data collection and preprocessing to model training and evaluation. Python frameworks and libraries can greatly simplify this process. Here's a general approach using popular Python frameworks:

1. Data Collection: Gathering and labeling fire and non-fire images.

2. Data Preparation: Preprocessing and splitting the data.
3. Model Building: Creating a machine learning model using python frameworks.
4. Model Evaluation: Assessing the model's performance on the test data.
5. Prediction: Making predictions and visualizing the results.

To address these limitations, machine learning-based image analysis methods have been developed and widely adopted. These methods learn to identify the most relevant characteristics for effective analysis. However, creating training data for these models is costly and labor-intensive. Experts must manually mark regions of interest or make medical diagnoses, a process that also introduces subjective and qualitative factors due to human involvement.

1.6 Research Outline

This research consists of five chapters, chapter two represents a brief background, and the related works. Chapter three gives the research methodology; it’s explaining the model and architecture. chapter four explores the results and discussion. Chapter five concludes this research recommendations.

CHAPTER TWO

LITERATURE REVIEW

CHAPTER TWO

LITERATURE REVIEW

2.1 Fire Disasters

Accidental fire is a natural disaster that seriously threatens public safety. In recent years, accidental fire has frequently occurred in many places as shown in the following pictures Figure 2.1, including superstores, communities and forests, yielding huge losses to production and human life [6].

Concerned with the increasingly intense and catastrophic natural disasters, the scientific community leverages the power of technology to develop and perfect simulation tools to better study catastrophic scenarios.

Figure 2-1 : Fire in Real World

2.2 Fire Detection

Fire detection means the determination that a fire is present using various sensing technology and systems [7]. The main aim of fire detection is to identify a fire at the earliest possible stage that therefore allows for quick responses to combat it and prevent the fire from escalating, cause minimal damage and most importantly protect lives. Fire detection systems are normally based on detectors fixed to sample the environmental conditions for signs of fire such as smoke, heat, flame or gas. When these sensors find a condition that suggest fire, then the alarms are launched, which may launch other safety measures such as sprinklers, and notify the tenants or emergency services [8].

2.3 Traditional and Advancements Methods of Fire Detection

One should note the fact that it is necessary to focus on the primary, traditional approaches to fire detection, so that people could fully appreciate the possibilities, researchers and advancements in the field of machine learning. This paper provides a brief description of the primary types of traditional fire detection systems and their respective advantages and disadvantages.

2.3.1 Heat Detection

Heat detection is the oldest method of detection of fire, whereby temperature variations are used to detect a fire. Heat detectors are usually installed where smoke detectors may cause false alarms for example in the kitchen, the garage or in areas where dust or fumes are present in large

measures. Below is a further look at detection of heat [9]. They take longer to respond of other detection methods such as smoke detectors because the temperature high enough to set off the alarm from the normal level. This delay can lead to the growth of the fire and before it is detected. Even the best heat detectors used in various industries are not as predictive as the machine learning based systems. Al systems can study several sources of data, otherwise, it is capable of identifying risks of fire before they turn into real fires and also minimize false reports by establishing differentiations between poisonous and harmless causes of heat.

2.3.2 Current Advancements in Fire Detection

Fire detection also has seen some innovation in the recent past, due to the Enhanced fire detection speed and reliability some innovations includes the use of machine learning, which enhances the ability to examine not only the patterns and the data from all the numerous detectors.

Machine learning in Figure 2.2 is an enhanced capability for fire detection and presents prospect for upgrading accuracy, time, and efficiency of fire detection systems. It means that by uniting the basic pros of existing fire detection systems with potential and analytical abilities of

machine learning, these systems are more effective in preventing fire and minimizing the consequences [11].

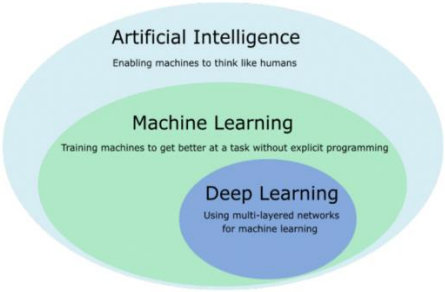

Figure 2-2: AI vs ML vs DL

2.4 Machine Learning in Image Analysis

Manifold application of ML has greatly revolutionized the area of image analysis, improving the effectiveness of the operation. Below you can find major course concepts and methods. Image analysis refers to the process, where, important information is mined from images this can involve identification, categorization and partitioning of objects in the image or video [7].

2.4.1 Types of Machine Learning

2.4.1.1 Supervised Learning

Models are trained on labeled datasets, learning to predict outcomes based on input features [12]. Common algorithms include:

- Convolutional Neural Networks (CNNs)

- Support Vector Machines (SVMs)

2.4.1.2 Unsupervised Learning

Models analyze unlabeled data to find patterns or groupings [12]. Examples include:

- Clustering algorithms (e.g., K-means)

- Autoencoders

2.4.1.3 Reinforcement Learning

Involves training models to make optimal decisions through trial and error [12].

2.4.2 Techniques in Image Analysis

Techniques to identify and quantify important features in images, such as edges and textures.

A subset of ML that uses neural networks with multiple layers to process images, particularly effective in:

- Image classification
- Object detection (e.g., YOLO, SSD)
- Image segmentation (e.g., U-Net, Mask R-CNN)

2.5 Applications of Machine Learning in Fire Detection

2.5.1 Computer Vision-based Fire Detection

See Figure 2.3 CVFD systems employ various digital image processing and machine learning algorithms to identify distinctive fire features such as color, motion, and dynamic patterns in a video sequence, to a varying level of sophistication [13]. Many variants of these systems have been introduced over the last two decades and the main techniques used in CVFD systems are:

2.5.1.1 Fire Color Segmentation

According to several studies in the RGB color space, a flame has highly distinct chromatic features in the range of the red- yellow band. The red, green, and blue components of fire are denoted by R. G and B,

respectively. As a result, many researchers in CVFD systems use color information to ascertain whether a pixel belongs to a candidate flame region in their algorithms' early stages [14].

2.5.1.2 RGB Color Models

Introduced three color space rules based on the RGB color components and the saturation values. According to the first rule, the red components of fire pixels are generally higher than the green ones, and the green ones are higher than the blue ones [15].

$$I(X)r>I(X)g>I(X)b \quad (2.1)$$

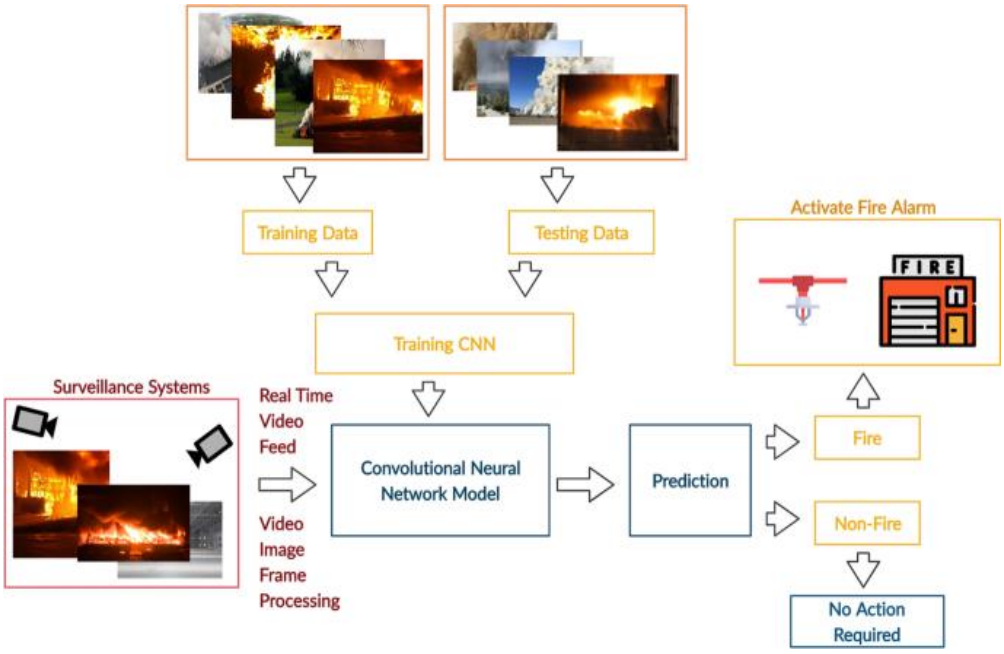

Figure 2-3 : Computer Vision-based Fire Detection

2.5.2 Fire Detection Algorithms Based on Convolutional Neural Networks

According to the principles of object detection algorithms, the flow of fire detection algorithms using convolutional neural networks (CNNs) is structured as follows: The detection CNN includes functions for region proposals, feature extraction, and classification.

First, the CNN processes an image as input, generating region proposals through techniques like convolution and pooling. Next, the region-based object detection CNN evaluates these proposed regions to determine the presence or absence of fire, utilizing convolutional layers, pooling layers, and fully connected layers [16].

See (Fig.2.4) The convolutional layer is the fundamental component of convolutional neural networks (CNNs). Unlike other neural networks that rely on connection weights and weighted sums, the convolutional layer employs image transformation filters known as convolution kernels to create feature maps from the original images. This layer consists of multiple convolution kernels, which slide over the images and compute new pixel values based on a weighted sum of the pixels they cover, resulting in a feature map. This feature map highlights specific characteristics of the original image. The calculation for the convolutional layer is represented by Equation:

$$Y = \sum_{j=0}^{j-1} \sum_{i=0}^{i-1} w_{ij}\, x_{m+1,n+j} + b,\ (0 \leq m \leq M, 0 \leq n \leq N) \qquad (2.2)$$

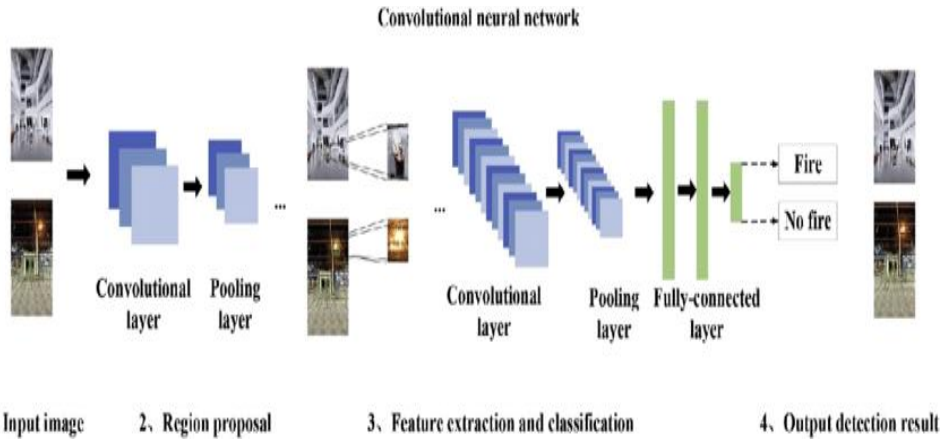

Figure 2-4: Convolutional Neural Networks Layer

Four highly effective image object detection networks—Faster R-CNN, R-FCN, SSD and YOLO v3—are chosen for developing fire detection algorithms due to their outstanding performance in both detection accuracy and speed [16].

2.5.2.1 Faster R-CNN

Faster Regions with CNN feature operates in two stages see Figure 2.4 (a). In the first stage, feature maps of the original image are created using feature extraction networks. A specific intermediate convolutional layer generates these feature maps, which the Region Proposal Network (RPN) uses to predict proposal regions along with their abjectness scores and locations [17, 18].

In the second stage, the proposed region locations are utilized to crop features from the same intermediate feature map through ROI pooling. The regional feature maps for each proposal are then fed into the remainder of the network to predict class-specific scores and refine bounding box locations. This approach allows for shared computations by cropping proposals from the feature map generated in the first stage, avoiding the need to input each proposal region into the front-end CNN for feature extraction. However, each proposal region must still be processed separately in the latter part of the network. Consequently, the detection speed is influenced by the number of proposal regions generated by the RPN [18].

2.5.2.2 Region-based Fully Convolutional Network R-FCN

Region-based Fully Convolutional Network has shown that CNNs need to be sufficiently deep to extract more complex features, which can enhance the networks' accuracy [19]. However, the feature maps

produced by deeper convolutional layers tend to be smaller and less sensitive to translation, resulting in less accurate localization for object detection tasks. To address this, Faster R-CNN incorporates a Region Proposal Network (RPN) in parallel with certain intermediate convolutional layers to retain crucial location information about objects Figure 2.5 (b) [20].

2.5.2.3 Single Shot MultiBox Detect SSD

Faster-RCNN and R–FCN, including a region proposal network, belong to the two-stage object detection networks. The detection speed of this kind of network is slower. Therefore, the one-stage object detection network, like SDD and YOLO v3, is proposed. It predicts the object class and location by a single forward CNN. SSD Figure 2.5 (c) [21].

2.5.2.4 You Only Look Once Algorithm

To maintain translation variance, SSD utilizes earlier layers to generate large-scale feature maps for detecting small objects. However, the features from these earlier layers are often too simplistic, resulting in poorer performance for smaller objects. To address these issues, YOLO v3 see Figure 2.5 (d) enhances detection accuracy by incorporating concepts from residual networks. Its one-stage approach excels in detection speed while improving object detection performance [16].

In [22] Corneli, proposed the custom YOLO model to retrieve real-time information from existing buildings. They trained neural networks and integrated into a system that utilizes Augmented Reality devices to capture images and display the results directly on-site.

In [23] The Smart Fire Detection System (SFDS), based on the YOLOv8 algorithm, offers an enhanced fire detection solution for smart cities. The system architecture includes four key players: Application,

Fog, Cloud, and IoT. By leveraging Fog and Cloud computing, alongside IoT, SFDS processes data in real time. It demonstrated state-of-the-art performance with a high precision rate of 97.1% across all detection categories.

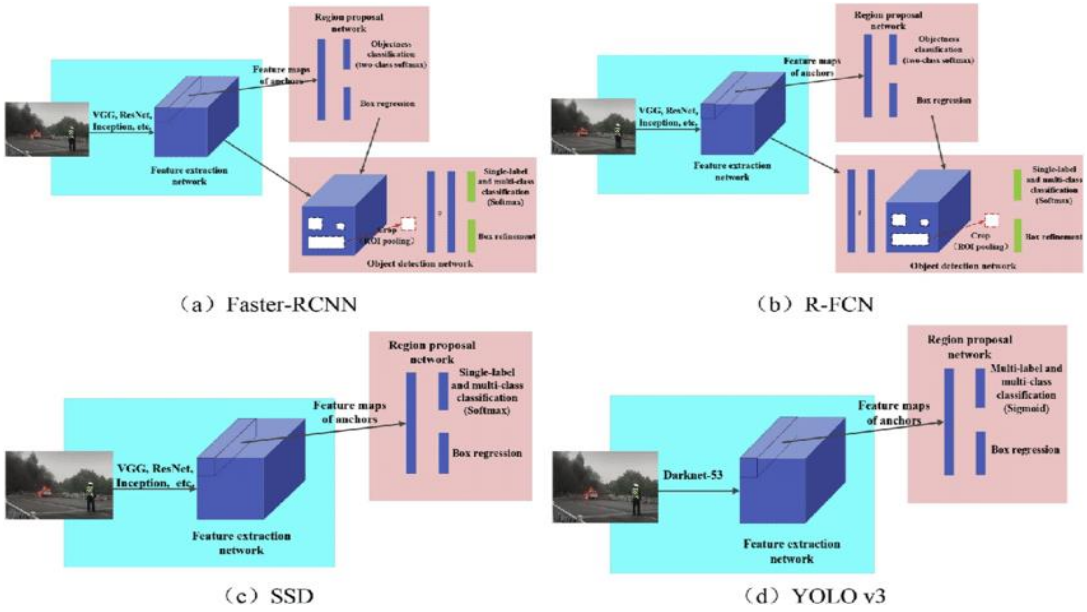

Figure 2-5 : Diagrams of fire detection algorithms based on the four CNNs

2.6 Related Works

2.6.1 Fire Detection Algorithm Using Image Processing Techniques

The research paper addressed the increasing prevalence of fire outbreaks in Malaysia and the significant damage these incidents caused to both nature and human interests. In response to this issue, the author proposed a fire detection algorithm grounded in image processing techniques that could be utilized with surveillance devices such as CCTV, wireless cameras, and UAVs(Unmanned areial vehicles) [24].

The algorithm employed the RGB color model to identify the fire's color, primarily focusing on the intensity of the red component. Fire growth was detected using Sobel edge detection, and a color-based segmentation technique was applied to isolate the region of interest (ROI) related to the fire by integrating the results from the previous two techniques.

After analyzing 50 different images of fire scenarios, the author reported that the algorithm achieved a final accuracy of 93.61% and an efficiency rate of 80.64% [24, 25].

2.6.2 Fire Detection Using Image Processing

The research paper discussed the challenges businesses faced regarding fire incidents, which caused significant property damage, injuries, and fatalities, despite having various preventive systems in place. It emphasized the critical importance of preparedness in dealing with fires, as they could spread uncontrollably and become difficult to contain. Early detection was identified as essential for effective fire containment.

The paper highlighted that traditional image fire detection relied heavily on algorithmic image analysis, which often suffered from lower accuracy and delayed detection. Common algorithms required substantial computations, including manual extraction of image features.

To address these Issues, the paper proposed a novel image detection method based on advanced object detection techniques, specifically the CNN model known as YOLO v3. The author found that this algorithm achieved an average precision of 81.76% and demonstrated strong robustness in detection performance, meeting the requirements for real-time fire detection [26].

2.6.3 Fire Detection Through Image Processing; A brief Overview

In the research paper, the author focused on enhancing fire detection systems to improve environmental protection and reduce hazards. The study highlighted the reliance on smoke detection technology and proposed advancements through the use of sensitive cameras capable of capturing spectral images. An algorithm for motion detection was deemed beneficial for early fire detection, thereby preventing potential damage.

To achieve this, the author utilized current algorithms with MATLAB's Image Processing Toolbox for extracting fire features and detecting motion. The proposed system emphasized video-based detection of fire and smoke, employing optical flow for vector extraction to train neural networks. This method allowed for the quantification of motion by analyzing changes between frames.

Additionally, a machine learning-based algorithm was implemented to identify the flame region. The author utilized optical flow estimators, specifically Optical Mass Transport (OMT) and Non-Smooth Data (NSD), to effectively detect fire and smoke. Overall, the research aimed at refining early detection strategies to mitigate fire-related risks [27].

2.6.4 Image Processing Based Forest Fire Detection Using Infrared Camera

In this research paper, the author addressed the problem of increasing wildfires due to technological advancements and the rise in natural and man-made disasters. They highlighted that wildfires pose a

significant danger, as they cause the burning of trees that provide oxygen, making it difficult to stop a forest fire If it is not detected early.

The aim of the project was to capture an infrared image to detect wildfires using an appropriate camera. The researcher utilized RGB and YCbCr color models to Isolate fire pixels from the background, separating luminance and chrominance from the original image. MATLAB Analyzer was used for image filtering and processing.

The method was tested on a selected image captured by the camera that contained a fire. Subsequently, a method was employed to compute and analyze the fire image, with the goal of differentiating between fire detection and false detection. The study successfully achieved effective detection of fire in the image [28].

2.6.5 Forest Fires Detection Using Machine Learning Techniques

In this research paper, the authors addressed the increasing problem of forest fires that cause significant damage worldwide. They presented machine learning regression techniques aimed at predicting forest fire-prone areas. The dataset used was sourced from the UCI machine learning repository and consisted of climate and physical factors from Montesinos Park in Portugal.

The researchers proposed three machine learning approaches: linear regression, ridge regression, and lasso regression, utilizing a dataset with 517 entries and 13 features per row. They implemented two versions of the analysis; the first included all features, while the second utilized only 70% of the features. The study employed a training set comprising 70% of the dataset, with the remaining 30% used as the test set The findings Indicated that the linear regression algorithm achieved greater

accuracy compared to the ridge regression and lasso regression algorithms [29].

2.6.6 Fire Detection and Recognition Optimization Based on Virtual Reality Video Image

In this research paper, the author addressed the significant challenge of fire detection due to Its potential to cause extensive damage to life and property rapidly. A recent solution involved the installation of IoT devices equipped with cameras for surveillance purposes. These devices could analyze captured video footage in real-time or transfer it to more powerful devices when the detection algorithm was resource-intensive. However, this cloud-based approach raised privacy concerns, as unauthorized access to the feed could compromise the footage.

The author proposed a fire detection system that achieved a high fire detection rate while also safeguarding the privacy of monitored areas. In this system, actual video streams were not used for fire detection; instead, the task was outsourced to cloud computing, which only relayed extracted features. The fire detection algorithm employed binary video descriptors and Convolutional Neural Networks (CNN) for classification. The method included both fire-related videos and videos wIth non-fire events to enhance reliability.

The results indicated that the new fire detection algorithm achieved a classification accuracy of 97.5%, slightly outperforming state-of-the-art algorithms that processed video feeds directly. Consequently, this system provided comparable reliability to modern solutions while offering improved protection against personal information leaks. Additionally, the author demonstrated that the proposed video descriptors could be implemented for real-time processing on an IoT device, specifically the

Raspberry 4 platform, with an average processing time of 100 milliseconds per frame, thus meeting practical requirements [6].

2.6.7 Fire Detection in the Buildings Using Image Processing

In efforts to reduce the loss of life and property from fire incidents, the author proposed an early warning fire detection system that utilized light detection and analysis methods. This system employed HSV (Hue, Saturation, Value) and YcbCr (Luminance and Chrominance) color models under specific conditions to effectively differentiate between orange, yellow, and high-intensity light against the background and ambient light [31].

Furthermore, the author analyzed and quantified fire growth through frame difference calculations. The experimental results indicated that the overall accuracy of the system surpassed 90%, showcasing its effectiveness in early fire detection [32].

2.6.8 Fire Detection Using Image Processing and Sensors

In today's world, It became mandatory to have a fire extinguisher In every building according to government regulations. However, during a fire outbreak, there often was chaos and confusion regarding whether to vacate the area or use the fire extinguisher. Additionally, proper training needed to be provided to employees or residents on how to operate the extinguisher.

To address these Issues, the authors developed a system that would automatically detect fire and activate the extinguisher without human intervention. They proposed placing cameras at various locations within the building or campus to detect fire using Image processing techniques. For enhanced accuracy, they also incorporated smoke and temperature sensors into the system.

The authors aimed to create an Intelligent fire detection system by first training it to Identify fire through advanced color recognition algorithms. Once the system learned to recognize fire, It could autonomously detect it and activate the extinguisher. Furthermore, the system would monitor the intensity of the fire and send alerts to the fire brigade, which previously had to be performed manually [33].

2.7 Chapter Summary

This chapter reviewed previous studies and theories related to Fire detection, focusing on key findings and methodologies from earlier works. It identified gaps or limitations in previous research on fire detection, highlighting areas that required further exploration. The chapter explained connections between earlier studies and the current project, establishing a foundation for the work. The review concluded by clarifying how this research addressed unanswered questions or built on prior knowledge in the field of fire detection.

CHAPTER THREE

METHODOLOGY

CHAPTER THREE

METHODOLOGY

3.1 Introduction

The machine learning model is capable of analyzing high-resolution images of both fires and non-fire scenarios from diverse sources. Using this algorithm, it can make highly accurate predictions. This automated approach holds significant potential for early fire detection and preemptive interventions, which may help extinguish fires in their early stages. The research gains importance as traditional fire detection methods, such as smoke detectors and heat sensors, often struggle with delayed response times and limited adaptability to different environments.

Deep learning models, particularly convolutional neural networks (CNNs), are commonly used for processing images and videos, enabling these systems to detect fire in real-time.

The remainder of this study delves into the specifics of putting CNN-based fire detection models into practice, covering topics including data collection, model architecture, training procedures, and performance assessments. The study aims to provide a comprehensive understanding of the methodologies involved and the potential impact.

Among these techniques, the You Only Look Once (YOLO) algorithm has become prominent due to its speed and accuracy in real-time object detection. This research introduces a novel application of the YOLO algorithm for detecting fire. By leveraging YOLO, we can efficiently process image data, allowing for rapid detection and response to potential fire incidents.

3.2 The Proposed Model and Requirements

This study is experimental in nature, aimed at designing and testing a deep learning-based fire detection model using the YOLO framework.

YOLO is a single stage detector, handling both the object identification and classification in a single pass of the network. YOLO is not the only single stage detection models (e.g. MobileNetSSDv2 is another popular single shot detector), but it is generally more performant in terms of speed and accuracy.

YOLO models are popular for a variety of reasons. First, YOLO models are fast. This enables you to use YOLO models to process video feeds at a high frames-per-second rate. Second, YOLO models continue to lead the way in terms of accuracy. Third, the YOLO family of models are open source, which has created a vast community of YOLO enthusiasts who are actively working with and discussing these models.

The YOLO algorithm is used for real-time object detection. Before YOLO, R-CNNs were among the most common method of detecting objects, but they were slow and not as useful for real-time applications. YOLO provides the speed necessary for use cases that require fast inference, such as detecting cars, identifying animals, and monitoring for safety violations [34].

In the following Figure 3.1 shows the framework of the research which with main processing units, that gives readers the main idea starting from dataset and model design, then going through how the model Evaluation. The flowchart represents the process of developing and evaluating a machine learning model YOLOv8.

1. **Data Collection**:

We Collect a dataset of images containing various scenarios with and without fire. This could include images from surveillance cameras, public datasets, or manually captured images. To ensure diversity and robustness in the dataset, gather images from different environments (e.g., indoor, outdoor, forests, urban areas) and under varying conditions (e.g., daytime, nighttime, fog, rain). Include examples of different fire types (e.g., wildfires, building fires, controlled fires) and non-fire scenarios that might visually resemble fire (e.g., sunlight reflections, red lights, or autumn leaves) to reduce false positives in the final model.

2. **Data Labeling**:

Assign labels to the images to create a supervised learning dataset. Images containing flames, smoke, or glowing embers are labeled as "fire", while images lacking these elements (e.g., natural landscapes, indoor scenes, or non-combustible objects) are labeled as "no fire". This process involves manual annotation using tool LabelStudio , with cross-validation by multiple annotators to ensure consistency. To address potential class imbalance, techniques such as oversampling underrepresented classes or augmenting fire images with synthetic smoke or varying brightness levels are applied. Metadata, such as image resolution, lighting conditions, or capture source, may also be recorded .to enhance model robustness against diverse scenarios

Feature Extraction:

Extract features from the images that help the model differentiate between "fire" and "no fire".

3. **Data Splitting**:

   Split the labeled dataset into three subsets:

   - **Training Data**: A majority of the images (80%) are used to train the model.
   - **Validation Data**: A smaller portion (15%) is used to tune the model's hyperparameters and assess its performance during training.
   - **Test Data**: The remaining images (5%) are reserved for evaluating the final model performance.

4. **Model Training**

The training data is utilized to train a deep learning-based fire detection model using the YOLOv8 (You Only Look Once version 8) architecture. YOLOv8, a state-of-the-art real-time object detection framework, is optimized for speed and accuracy. During training, the model learns to identify fire patterns by processing labeled images containing fire and non-fire instances. Key components include anchor box adjustments, bounding box regression, and classification loss optimization. Hyperparameters such as learning rate, batch size, and the number of epochs are fine-tuned to enhance performance. Data augmentation techniques (e.g., rotation, scaling, and flipping) are applied to improve generalization. The model iteratively updates its weights

through backpropagation, minimizing prediction errors using optimization.

5. **Trained Model**:

After training is complete, the model is saved as a trained fire detection model. It has learned to classify and detect fire in unseen images. . It achieves this by leveraging learned features such as flame color, texture, motion, and smoke patterns. Performance metrics like precision, recall, and mean Average Precision (mAP) are evaluated to ensure robustness.

6. **Evaluation**:

Test the trained model on the test data to measure its performance. Key evaluation metrics include Accuracy as how often the model correctly detects fire or no fire, Precision like how many of the fire detections are correct. Recall as how many of the actual fire images are detected. F1 Score A balance of precision and recall.

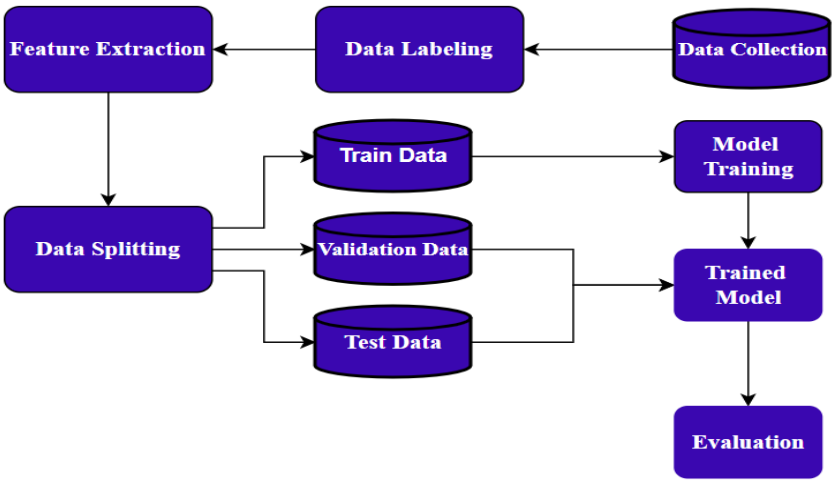

Figure 3-1: Model Prototype Architecture

3.2.1 Hardware Requirements

Graphics processing Unit (GPU): is crucial for faster training and inference, especially for larger models and datasets we are using NVIDIA GPU Tesla T4 [35].

RAM: 12-16 GB for free users; up to 32 GB for Pro users.

CPU: 2-4 cores for auxiliary tasks.

3.2.2 Software Requirements

Google Colab: Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free of charge access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning.

Google Drive: file sharing platform that provides a personal, secure cloud storage option to share content with other users.

PyCharm: is the most popular IDE used for Python scripting language, enables smoother code completion whether it is for built in or for an external package.

Roboflow: provides a comprehensive platform for data organization, labeling, and versioning which is essential for efficient machine learning workflows.

3.3 The System Developing and Implementation

The goal of the project is to develop a real-time fire detection system using the YOLOv8 object detection model, trained and prepared on Google Colab [36], with the dataset stored in Google Drive, and tested on local software like PyCharm. Below is the methodology broken down into distinct stages:

3.3.1 Data Collection and Preprocessing

The Fire dataset is a specialized collection of fire images curated for the development of object detection methods. Given the varied morphology of flames , the dataset from the Roboflow [37], It contains 4225 images, the dataset is typically split into **80% training**, **15% validation**, and **5% testing** to ensure the model learns effectively while preventing overfitting and allowing performance evaluation. The dataset contains a variety of images from different environments such as forests, buildings, urban areas, and industrial settings. **Fire Scenarios** Images with fire include various fire intensities (small to large flames), smoke, and different lighting conditions (day, night). And **Non-Fire Scenarios** the dataset also includes images without fire (to reduce false positives) and challenging situations, such as fog, light reflections, or bright lights, to improve robustness. Images may vary in resolution[38], but they are typically resized during preprocessing to fit YOLO's input size (640x640).

3.3.2 Model Training and Preparation on Google Colab

YOLOv8 was developed by Ultralytics [39], who also created the influential and industry-defining YOLOv5 model. YOLOv8 includes numerous architectural and developer experience changes and improvements over YOLOv5. YOLOv8 comes with a lot of developer-

convenience features, from an easy-to-use CLI to a well-structured Python package. YOLOv8 is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box.

3.3.3 YOLOv8 Architecture

The **YOLOv8 architecture** can be broadly divided into three main components as shown in Figure 3.2:

The backbone is responsible for feature extraction from the input image. Its role is to take the raw image and extract increasingly abstract features by passing the image through several layers of convolutional operations. The backbone block consists of three components, which aim to improve learning efficiency by splitting the feature map into two parts.

Cross Stage Partial Block (CSP-A): These blocks aim to improve learning efficiency by splitting the feature map into two parts, where one part goes through a series of convolutional operations and the other part bypasses them.

ConvModule: Standard convolution modules This helps the model capture spatial features from the input image. Spatial Pyramid Pooling—Fast (SPPF) this layer is often used at the end of the backbone to aggregate context information from different regions of the image.

The Neck component in the YOLOv8 architecture is responsible for further processing the feature maps obtained from the backbone [37]. Its primary goal is to refine these feature maps and combine information from different scales to help in detecting objects of various sizes more effectively. This block consists of three components:

The up sampling layers are used to increase the spatial dimensions of feature maps. Concatenation, This is a crucial operation in the neck where feature maps from different stages are concatenated.

Variant of Cross Stage Partial Block with Global Sub-sampling Convolutions (VoVGSCSP): This block is an enhanced version of the CSP block found in the Backbone. It applies sub-sampling and convolutions to further process and refine the feature maps.

Global Sub-sampling Convolution (GSConv): This layer is used to reduce the spatial dimensions of feature maps while retaining important contextual information.

The Head that takes the processed feature maps from the neck and produces the final predictions, the head is responsible for performing the actual object detection, and this block consists of three components.:

ConvModule: The Head uses several convolutional layers to further refine the feature maps received from the Neck. These layers extract the most relevant information needed for making precise predictions about objects in the image.

Conv2d [31]: The Conv2d layers are used to adjust the feature maps into a suitable format for detection. Detect Layer: This is the final output layer of the model, where object detection happens.

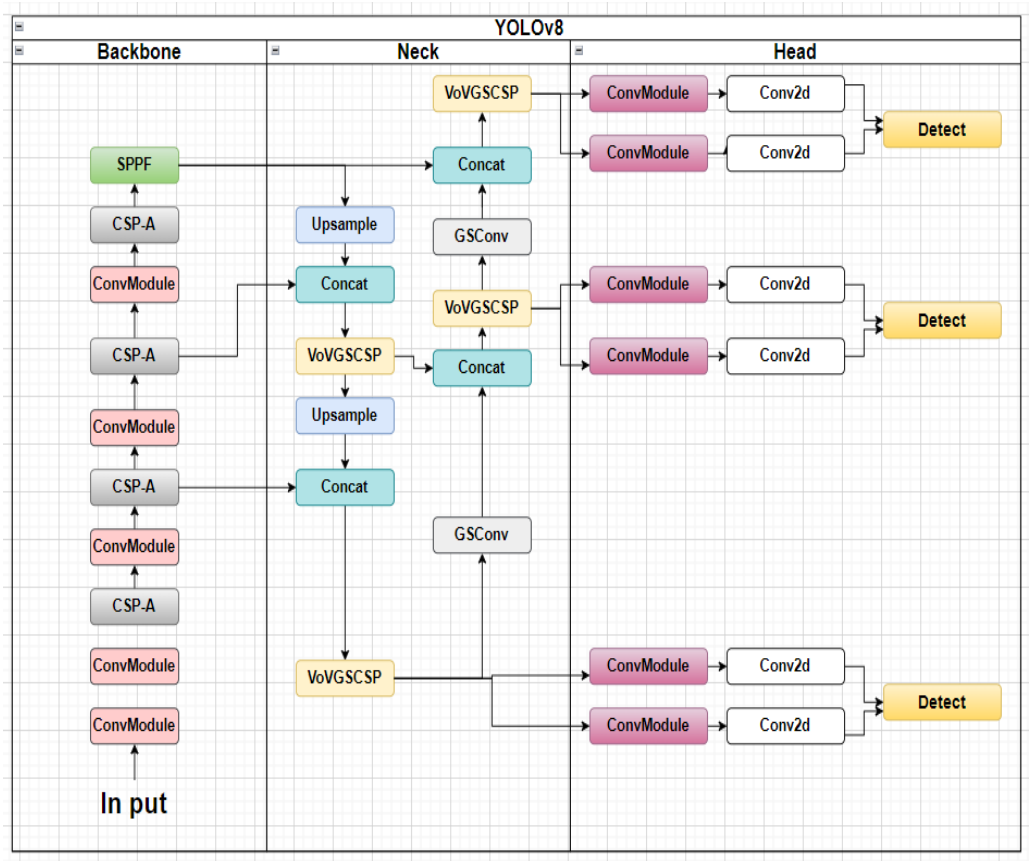

Figure 3-2 : the architecture of YOLOv8

3.3.3.1 Installing YOLOv8

YOLOv8 Installation: Install the YOLOv8 package in Colab using command:

(!pip install ultralytics)

3.3.3.2 Configuring the YOLOv8 Model

Pre-trained Weights: Start with a pre-trained YOLOv8 model (e.g., yolov8n.pt) for transfer learning, and Data Configuration Prepare a " data.yaml" file specifying the paths to the training, validation, and testing images, and the class names[41], after that Saving the Model After training, the final model weights and logs are saved in Google Drive for easy retrieval and further testing.

3.3.3.3 Validation During Training

During the training process, validation is performed after each epoch to evaluate model performance using metrics such as:

3.3.3.4 A Confusion Matrix Related to Fire Detection

Based on the image, the matrix compares predicted labels as the True Positive (TP) the top-left square (4104) represents the number of correctly predicted fire occurrences 97.2%, False Positive (FP) the top-right square (90) represents cases where the system predicted fire 2.1% of the overall.

False Negatives (FN): Incorrectly predicted negative cases is (31) it is 0.73% of the overall (bottom-left value in the matrix). True Negatives (TN): Correctly predicted negative cases is (0) in (bottom-right value in the matrix) [11].

Table 3-1: Confusion Matrix Related Predicted Labels

True Positive (TP)	4104 images	97.2%
False Positive (FP)	90 images	2.1%
False Negatives (FN)	31 images	0.73%
True Negatives (TN)	0 image	0 %

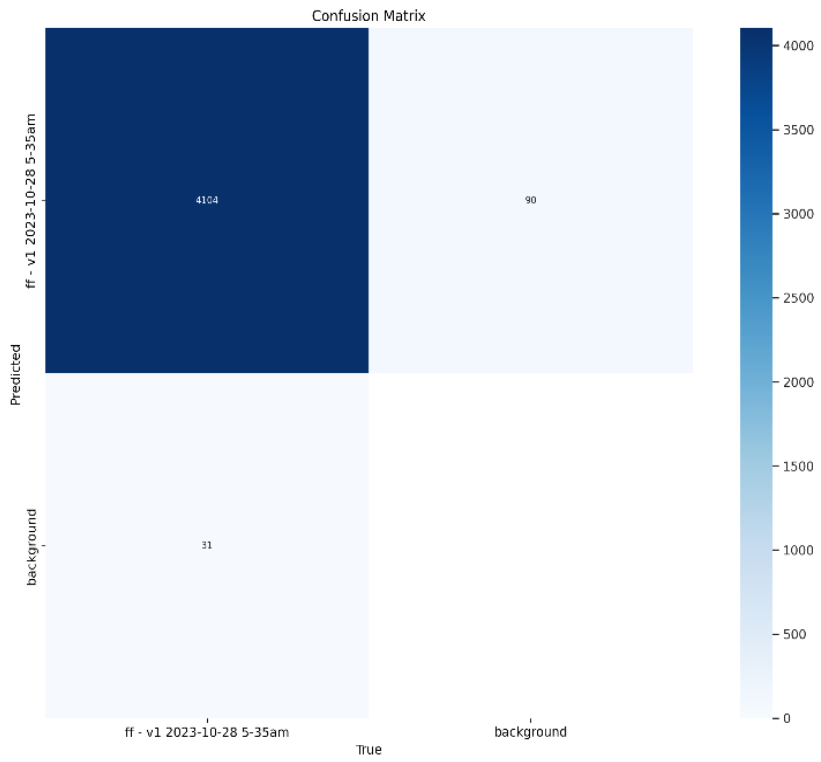

Figure 3-3 A confusion Matrix

3.3.3.5 Accuracy

Accuracy measures the proportion of correct predictions (both positive and negative) out of all predictions made. In this case, 97.14% of the model's predictions were correct overall, including both and "background" predictions [24].

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{3.1}$$

Accuracy represents the number of correctly classified data instances over the total number of data instances.

From the confusion matrix in Figure 3.3 , Accuracy = (4104 + 0)/(4104 + 90 + 0 + 31) = 0.99 and in percentage the accuracy is 99%.

3.3.3.6 Precision

In Figure 3.4 this tells us how many of the positive predictions made by the model were actually correct. A high precision means that when the model predicts a positive result. Precision focuses on the accuracy of the positive predictions.

Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal TP = TP +FP, this also means FP is zero. As FP increases the value of denominator becomes greater than the numerator and precision value decreases.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

So from the confusion matrix in Figure 3.3, precision = 4104/(4104+ 90) = 0.97

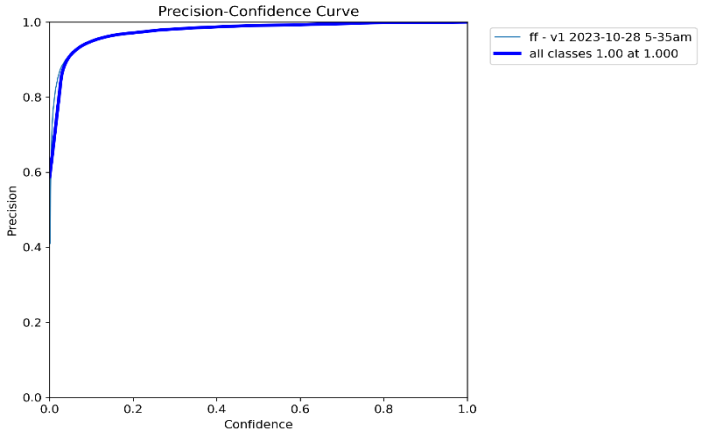

Figure 3-4 : Precision-Confidence Curve

3.3.3.7 Recall

This metric indicates how many of the actual positive cases were correctly identified by the model. A recall of 99.25% means that the model correctly identified almost all of the actual cases. Recall focuses on the model's ability to find all the relevant positive cases.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

Recall = 4104/(4104+ 31) = 0.9925

This graph illustrates the Recall-Confidence Curve, (y-axis) Represents how well the model identifies actual fires (true positives) across different confidence thresholds and (x-axis) Indicates the model's certainty in its predictions, ranging from 0 (no confidence) to 1 (high confidence) as shown in Figure 3.5.

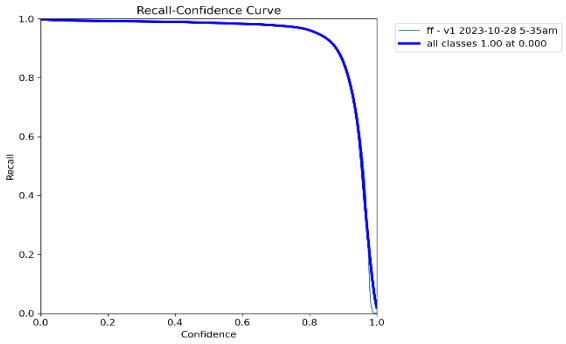

Figure 3-5 : Recall-Confidence Curve

3.3.3.8 F1 Score

In Figure 3.6 The F1 score is the harmonic mean of precision and recall, providing a balance between them. It is particularly useful when you want to consider both false positives and false negatives equally. A high F1 score indicates that the model performs well in both precision and recall [42].

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3.4)$$

F1 Score = 2* (0.97 * 0.99)/(0.97 + 0..99) = 0.98

The model performs very well, with strong accuracy, precision, recall, and F1 score. It is effective at both identifying positives and minimizing false predictions.

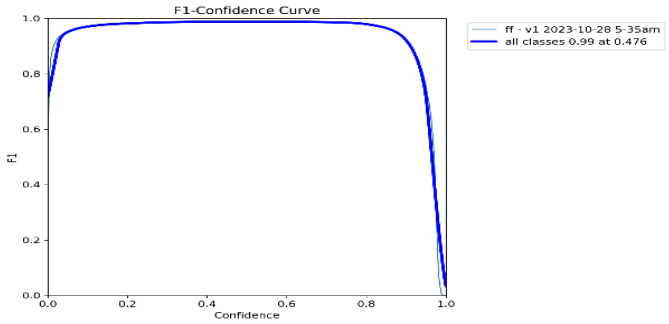

Figure 3-6 : F1-Confidence Curve

3.3.3.9 Precision-Recall

Represent How many predicted fires are actual fires. In this curve, precision remains high, which means that most of the predicted fires are indeed actual fires. With this curve reaching very high recall values, it suggests that the model is effectively identifying most actual fires.

The curve looks almost ideal, as the precision remains close to 1, even as recall increases, suggesting the model performs exceptionally well. The mean Average Precision for all classes is shown as 0.994, indicating strong overall performance in terms of both precision and recall as shown in Figure 3.7.

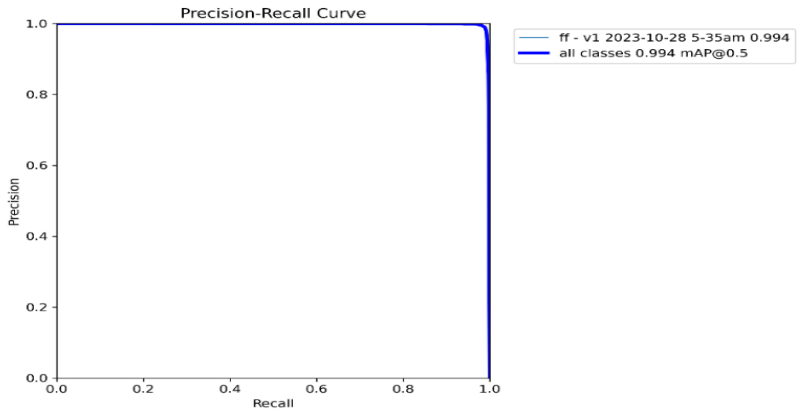

Figure 3-7: Precision-Recall Curve

3.3.4 Testing on Unseen Data

Once the model is trained, it's tested on a separate testing set (5% of the dataset) to evaluate how well it generalizes to unseen fire and non-fire images.

3.3.4.1 Testing and Inference Using PyCharm

3.3.4.1.1 Setting Up PyCharm

- **Install YOLOv8**: Set up PyCharm on your local machine and install YOLOv8 using pip: pip install ultralytics
- **Load Trained Model**: Load the trained YOLOv8 model from Google Drive or a local path: from ultralytics import YOLO.

model = YOLO('/path_to_trained_model/best.pt') Inference on Local Images or Video.

- **Running Inference**: Test the model on new images or videos to detect fire in real time using the flowing code :

results = model.predict(source='/path_to_image_or_video')

results.show() # Display predictions

The model will output bounding boxes and class labels for any fire detected in the images or video frames.

3.3.5 Visualization

Bounding Boxes: The results will include bounding boxes around fire regions, along with confidence scores, allowing real-time fire detection.

Figure 3-8 : Bounding Boxes

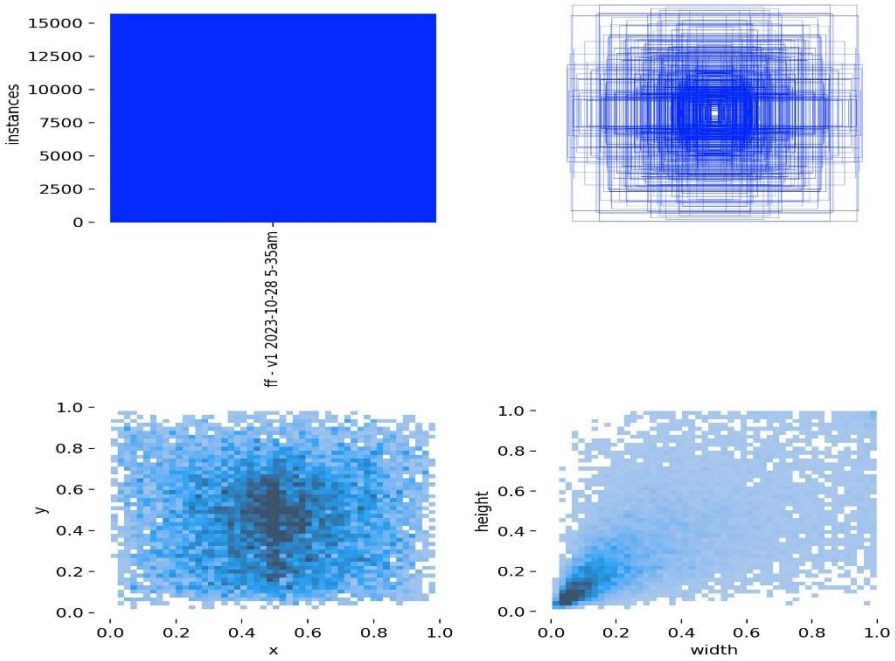

Figure 3-9 : Bounding Boxes and Confidence Scores

Blue boxes have been drawn around the regions where the model detected fire as shown in Figure 3.8. These boxes are labeled with the confidence score (ranging from 0.9 to 1.0), showing how sure the model is detected the fires.

Each detection is labeled (likely representing the fire detection model's version) the model's confidence scores are prominently displayed, indicating that it is highly confident (almost all detections have scores of 1.0).

The Figure 3.9 show four plots related to the analysis of fire detection instances, each representing different aspects of the detection process, particularly the distribution of bounding boxes and confidence scores.

Top-left (Instances bar):

This block shows a large solid blue square labeled, It seems to represent a uniform label for the detections, potentially indicating the number of

instances detected across all images. The solid color suggests a large number of detections, but without additional variation.

Top-right (Bounding Box Heatmap):

This grid appears to show an overlapping collection of bounding boxes from various detections. The boxes are blue and concentrated toward the center, implying that most detections occur around a central point of interest in the images. This could represent the locations where fire is most commonly detected within the frames.

Bottom-left (x vs y coordinates):

A heatmap showing the distribution of the center coordinates (x, y) of the bounding boxes. The density of points in the center indicates that most fires are detected near the middle of the images. The spread toward the edges implies that fires can appear anywhere in the frame, but tend to be more concentrated centrally.

Bottom-right (Width vs Height of bounding boxes):

Another heatmap showing the distribution of the width and height of the bounding boxes. It reveals that most bounding boxes are relatively small, as the concentration is toward the bottom left, where both width and height are smaller (close to 0). This suggests that the fire detection model identifies fire at various scales, but typically within smaller regions.

3.4 Chapter Summary

This methodology provides a structured approach to developing a YOLOv8-based fire detection system, from dataset preparation as Fire and non-fire images with annotations for bounding boxes, split into training, validation, and testing sets.

The YOLOv8 model train using Colab with GPU acceleration, leveraging Google Drive for dataset storage. Testing and evaluate the trained model locally using PyCharm, running inference on new images or video to detect fire in real time.

CHAPTER FUOR

RESULTS AND DISCUSSION

CHAPTER FUOR

RESULTS AND DISCUSSION

4.1 Introduction

This chapter represents the results and discussions of the Fire detection using machine learning Algorithm a YOLOv8. This analysis focuses on evaluating the performance of a YOLOv8 model trained for fire detection and its application to fire detection is particularly valuable for improving response times in safety-critical environments such as surveillance and emergency systems.

In the results, key metrics are analyzed over multiple training epochs to measure the model's ability to detect fire objects. Metrics like bounding box loss, classification loss, precision, recall, and mean Average Precision (mAP) provide insights into the model's performance and its generalization on unseen data. These metrics are derived from both the training and validation datasets, ensuring a balanced evaluation of the model's capabilities.

The model shows a significant decrease in loss values, suggesting effective learning and optimization, while high and improving precision and recall demonstrate its strong detection performance. The consistently increasing mAP values further indicate that the model is accurately predicting bounding boxes and classifying fire objects, making it a robust solution for fire detection tasks.

4.2 The YOLOv8 Training Results:

In Figure 4.1, shown the top row raining metrics, the loss is related to the accuracy of bounding box predictions during training. The box loss starts at around 0.54 and progressively decreases to about 0.22, indicating that the model improves at localizing the fire and smoke objects over time. And shows the classification loss during training, representing how well the model is classifying objects (fire or smoke) into their respective categories; the loss starts high (around 0.92) and decreases to approximately 0.19 by the end of training, indicating the model is learning to differentiate between fire, smoke, and other objects. And Distribution Focal Loss (DFL), related to the quality of bounding box regression predictions (how well the predicted boxes match the ground truth). The DFL loss decreases from around 0.95 to 0.81, showing a reduction in errors related to the regression task. Precision starts at around 0.93 and rises to approximately 0.99 by the 15th epoch, though there are fluctuations. And metrics recall measures how many of the actual positives (fire instances) there are fluctuations early on, but it stabilizes at a high value.

In Figure 4.1, Bottom Row (Validation Metrics) shows the bounding box loss on the validation set, which helps evaluate how well the model generalizes. The validation loss decreases from 0.44 to around 0.20, following a similar trend to the training box loss. But the classification loss on the validation set, similar to the training classification loss, drops from around 0.57 to about 0.16, which indicates the model is performing well on unseen fire images. And distribution focal loss (DFL) on the validation set, reflecting the quality of the bounding box regression. The loss decreases from about 0.88 to 0.76,

showing that the model is making fewer errors in bounding box regression on the validation data.

And the mean average precision (mAP) at an IoU threshold of 0.50. It evaluates the overlap between the predicted and ground truth bounding boxes. The mAP50 increases from around 0.97 to 0.99, showing that the model is performing very well in terms of detection accuracy and mAP at multiple IoU thresholds (from 0.50 to 0.95), giving a more comprehensive view of the model's performance across different IoU levels. The mAP50-95 metric starts at around 0.86 and increases to about 0.97.

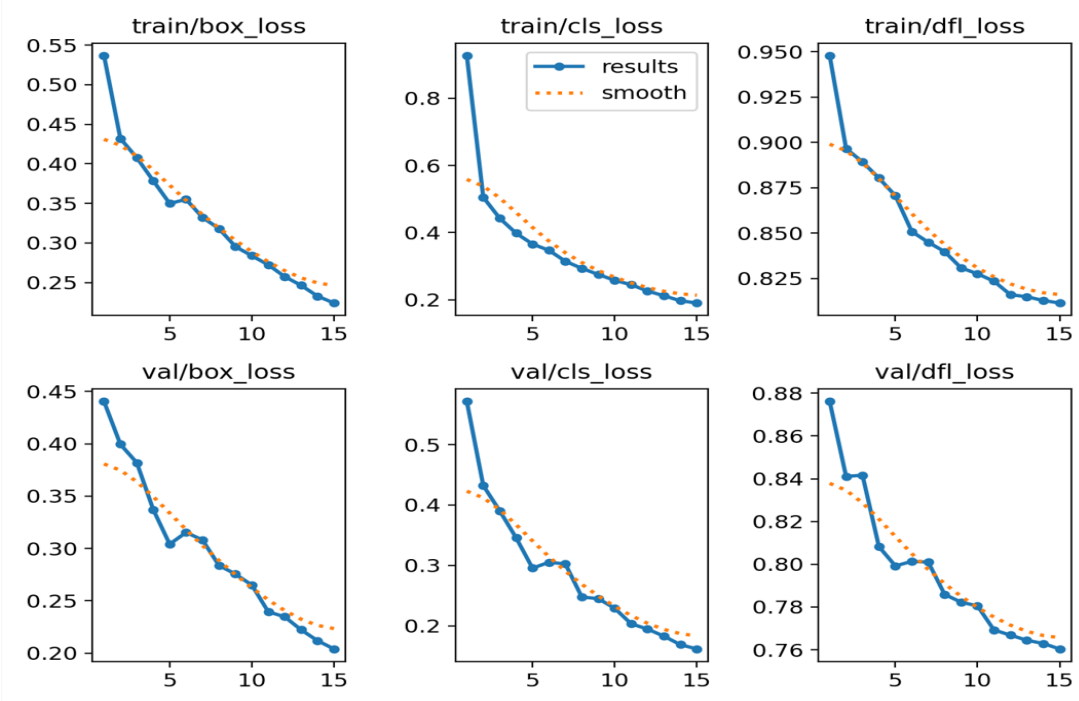

Figure 4-1 : YOLOv8 fire detection training results

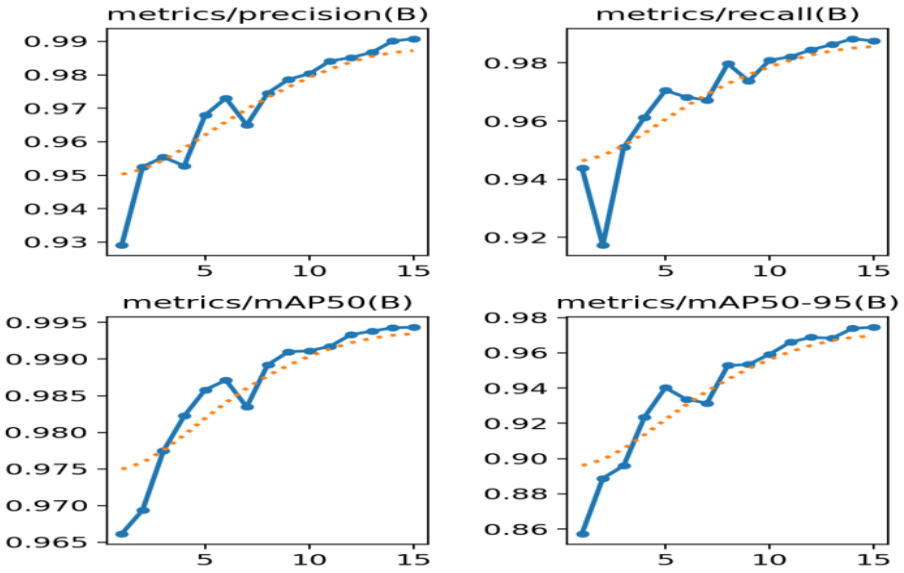

Figure 4-1: YOLOv8 fire detection training results

Table 4-1 Showing YOLOv8 Fire Detection Model Training Data:

Train Box loss	train/ cls loss	train/ DFL loss	metrics precisi	Metrics/ recall	Metrics mAP50	val/ box loss	val/ cls loss	val/ DFL loss
0.5369	0.9278	0.9478	0.9290	0.94389	0.96616	0.4405	0.5716	0.876
0.4319	0.5056	0.8965	0.9524	0.91729	0.96938	0.3994	0.4323	0.841
0.4076	0.4430	0.8891	0.9554	0.95091	0.97746	0.3818	0.3906	0.841
0.3785	0.3983	0.8802	0.9527	0.96127	0.98224	0.3370	0.3458	0.808
0.3498	0.3660	0.8705	0.9679	0.9705	0.9858	0.3040	0.2956	0.799
0.3550	0.3473	0.8508	0.9730	0.96825	0.98712	0.3152	0.3052	0.801
0.3321	0.3146	0.8449	0.9650	0.96723	0.98347	0.3079	0.3025	0.801
0.3181	0.2937	0.8397	0.9745	0.97973	0.98921	0.2837	0.2478	0.785
0.2953	0.2755	0.8308	0.9786	0.9738	0.99101	0.2754	0.2451	0.782
0.2837	0.2579	0.8275	0.9803	0.98089	0.99113	0.2649	0.2292	0.780
0.2723	0.2450	0.8234	0.9841	0.9821	0.99174	0.2394	0.2035	0.769
0.2574	0.2261	0.8160	0.9851	0.98452	0.99334	0.2346	0.1947	0.766
0.2463	0.212	0.8149	0.9866	0.98629	0.99381	0.2221	0.1829	0.764
0.2327	0.1971	0.8128	0.9900	0.9883	0.9942	0.2117	0.1689	0.762
0.2240	0.1904	0.8114	0.9907	0.9875	0.9943	0.2039	0.1618	0.760

4.3 Bounding Box Correlogram

In Figure 4.2 illustrates a correlogram for bounding box variables center coordinates (x and y), width, and height used in the YOLOv8 detection pipeline. It includes scatter plots for each pair of variables and histograms along the diagonal representing the distribution of each variable. Represents the normalized x-coordinate of the center of the fire's bounding box relative to the width of the input image. Values range from 0 (left edge of the image) to 1 (right edge of the image). And y (center y-coordinate) represents the normalized y-coordinate of the center of the fire's bounding box relative to the height of the input image. Values range from 0 (top edge of the image) to 1 (bottom edge of the image).

Width : The normalized width of the bounding box representing the detected fire, relative to the image's width. It shows how much of the horizontal span the fire covers in the image.

Height: The normalized height of the bounding box representing the detected fire, relative to the image's height. It indicates the vertical span of the fire.

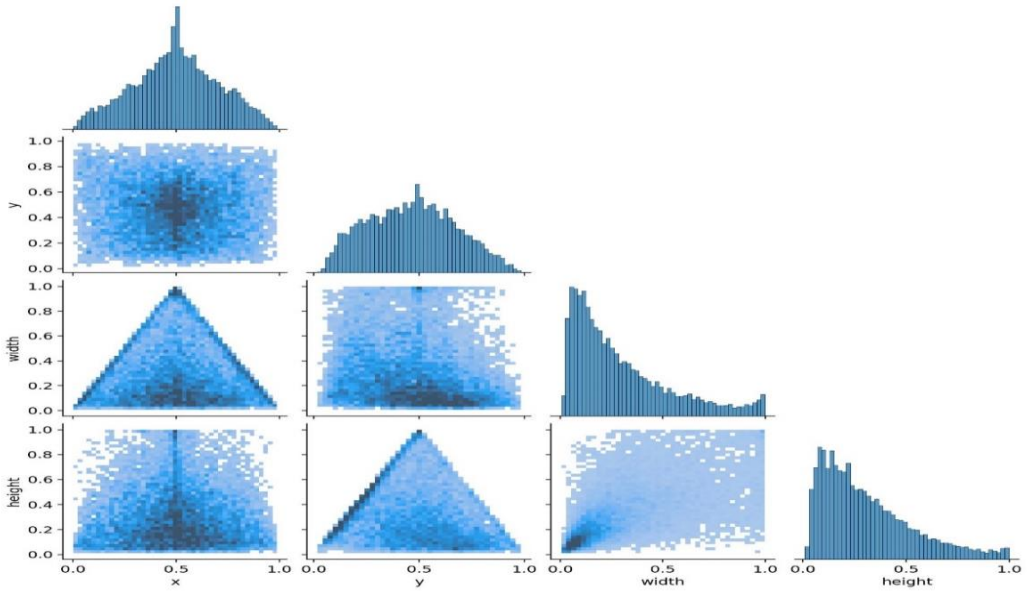

Figure 4-1 : labels correlogram

4.4 Chapter Summary

The model exhibits a notable reduction in loss values, indicating efficient learning and optimization. Additionally, the high and steadily improving precision and recall highlight its strong detection capabilities. The consistently rising mAP scores confirm the model's accuracy in predicting bounding boxes and classifying fire, positioning it as a reliable solution for fire detection tasks.

The decreasing loss values (box, classification, DFL) in both training and validation suggest that the model is learning effectively and generalizing well to unseen data.

The increasing precision, recall, mAP50, and mAP50-95 values indicate the model is becoming highly accurate at detecting fire and smoke, with minimal false positives and negatives.

The fluctuations in early training epochs are common in object detection models, but the trends stabilize over time, showing consistent improvement.

CHAPTER FIVE

CONCLUTIOIN AND RECOMMENDATION

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

The theme of this study explored the practical implementation of YOLOv8-based fire detection models, focusing on aspects such as data collection, model architecture, training processes, and performance evaluation.

The dataset, consisting of 4225 images, is typically divided into 80% for training, 15% for validation, and 5% for testing. This split ensures the model learns effectively, minimizes overfitting, and allows for a thorough evaluation of its performance.

Loaded the dataset and enhanced it using data augmentation techniques to improve the model's generalization capabilities. The YOLOv8 model trained using Colab with GPU acceleration, leveraging Google Drive for dataset storage.

Trained the YOLOv8 model on the chosen dataset while closely monitoring the loss curve and fine-tuning hyperparameters for optimal performance. Validated the model on a separate validation set to mitigate overfitting. Finally, assess the model's performance on a test set comprising unseen data.

The model demonstrates a significant decrease in loss values, signaling effective learning and optimization. The steadily increasing precision and recall further emphasize its strong detection capabilities. Consistently rising mAP scores validate the model's accuracy in predicting bounding boxes and classifying fire, establishing it as a reliable solution for fire detection.

The decline in box, classification, and DFL losses during both training and validation phases indicates that the model is learning efficiently and generalizing well to new data. Improvements in precision, recall, mAP50, and mAP50-95 values highlight the model's growing accuracy in detecting fire and smoke, with fewer false positives and negatives.

While early fluctuations in the training process are typical for object detection models, the overall trends stabilize, reflecting consistent improvement over time.

5.2 RECOMMEDATIONS

1. **Further Training with Diverse Data**: To improve the robustness of the model, additional training on more diverse datasets (such as different weather conditions, environments, and camera angles) is recommended. This would enhance the model's ability to generalize to a wider variety of real-world situations.
2. **Fine-tuning for Real-Time Performance**: For real-time fire detection systems, consider optimizing the model's inference speed without compromising its accuracy. This can be done through

hardware accelerators (e.g., GPUs or TPUs) or by reducing the model's complexity while maintaining sufficient performance.

3. **Deploy in Surveillance Systems**: Given the model's strong performance, it is well-suited for integration into fire detection systems for real-time surveillance and early warning applications. This can significantly enhance response times and improve safety measures in critical environments.
4. **Continuous Monitoring and Updates**: Regular retraining with new data and potential updates to the YOLOv8 architecture could help maintain the model’s accuracy and relevance as new fire detection challenges arise.

List of Publication

1. Mohamed Hary , Ismail Abdalla , Mohammedalgasem Mohammed, Alfatih Adam , Eissa Khmeas and Sallam O. F. Khairy " Fire Detection through Image Recognition Using YOLOv8 Machine Learning Model", "MENA Region Entrepreneurship Conference (MENAREC 2024) " .

References:

[1] S. B. Kukuk and Z. H. Kilimci, "Comprehensive analysis of forest fire detection using deep learning models and conventional machine learning algorithms," *International Journal of Computational and Experimental Science and Engineering,* vol. 7, no. 2, pp. 84-94, 2021.

[2] R. Aswathy *et al.*, "Optimized tuned deep learning model for chronic kidney disease classification," *Comput. Mater. Contin,* vol. 70, pp. 2097-2111, 2022.

[3] V. Wiley and T. Lucas, "Computer vision and image processing: a paper review," *International Journal of Artificial Intelligence Research,* vol. 2, no. 1, pp. 29-36, 2018.

[4] A. Gaur, A. Singh, A. Kumar, A. Kumar, and K. Kapoor, "Video flame and smoke based fire detection algorithms: A literature review," *Fire technology,* vol. 56, pp. 1943-1980, 2020.

[5] A. Arul, R. H. Prakaash, R. G. Raja, V. Nandhalal, and N. S. Kumar, "Fire detection system using machine learning," in *Journal of Physics: Conference Series*, 2021, vol. 1916, no. 1: IOP Publishing, p. 012209.

[6] X. Huang and L. Du, "Fire detection and recognition optimization based on virtual reality video image," *IEEE Access,* vol. 8, pp. 77951-77961, 2020.

[7] E. S. A. Ahmed, Z. T. Mohammed, M. B. Hassan, and R. A. Saeed, "Algorithms optimization for intelligent IoV applications," in *Handbook of research on innovations and applications of AI, IoT, and cognitive technologies*: IGI global, 2021, pp. 1-25.

[8] R. Bogue, "Sensors for fire detection," *Sensor Review,* vol. 33, no. 2, pp. 99-103, 2013.

[9] H. A. Zahl and M. J. Golay, "Pneumatic heat detector," *Review of Scientific Instruments,* vol. 17, no. 11, pp. 511-515, 1946.

[10] A. H. Altowaijri, M. S. Alfaifi, T. A. Alshawi, A. B. Ibrahim, and S. A. Alshebeili, "A privacy-preserving iot-based fire detector," *IEEE Access,* vol. 9, pp. 51393-51402, 2021.

[11] R. F. Mansour, N. M. Alfar, S. Abdel-Khalek, M. Abdelhaq, R. A. Saeed, and R. Alsaqour, "Optimal deep learning based fusion model for biomedical image classification," *Expert Systems,* vol. 39, no. 3, p. e12764, 2022.

[12] E. F. Morales and H. J. Escalante, "A brief introduction to supervised, unsupervised, and reinforcement learning," in *Biosignal processing and classification using computational learning and intelligence*: Elsevier, 2022, pp. 111-129.

[13] S. Geetha, C. Abhishek, and C. Akshayanat, "Machine vision based fire detection techniques: A survey," *Fire technology,* vol. 57, no. 2, pp. 591-623, 2021.

[14] T. Celik and K.-K. Ma, "Computer vision based fire detection in color images," in *2008 IEEE Conference on Soft Computing in Industrial Applications*, 2008: IEEE, pp. 258-263.

[15] C. E. Premal and S. Vinsley, "Image processing based forest fire detection using YCbCr colour model," in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, 2014: IEEE, pp. 1229-1237.
[16] P. Li and W. Zhao, "Image fire detection algorithms based on convolutional neural networks," *Case Studies in Thermal Engineering,* vol. 19, p. 100625, 2020.
[17] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Light-head r-cnn: In defense of two-stage object detector," *arXiv preprint arXiv:1711.07264,* 2017.
[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, no. 6, pp. 1137-1149, 2016.
[19] Q. Zhang, J. Xu, L. Xu, and H. Guo, "Deep convolutional neural networks for forest fire detection," in *2016 International forum on management, Education and information technology application*, 2016: Atlantis Press, pp. 568-575.
[20] Y. Wang, X. Ji, Z. Zhou, H. Wang, and Z. Li, "Detecting faces using region-based fully convolutional networks," *arXiv preprint arXiv:1709.05256,* 2017.
[21] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 2016: Springer, pp. 21-37.
[22] A. Corneli, B. Naticchia, M. Vaccarini, F. Bosché, and A. Carbonari, "Training of YOLO neural network for the detection of fire emergency assets," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, 2020, vol. 37: IAARC Publications, pp. 836-843.
[23] O. O. Khalifa, M. H. Wajdi, R. A. Saeed, A. H. Hashim, M. Z. Ahmed, and E. S. Ali, "[Retracted] Vehicle Detection for Vision-Based Intelligent Transportation Systems Using Convolutional Neural Network Algorithm," *Journal of Advanced Transportation,* vol. 2022, no. 1, p. 9189600, 2022.
[24] O. Khalifa *et al.*, "An IoT-Platform-Based Deep Learning System for Human Behavior Recognition in Smart City Monitoring Using the Berkeley MHAD Datasets. Systems 2022, 10, 177," ed, 2022.
[25] K. Poobalan and S.-C. Liew, "Fire detection algorithm using image processing techniques," in *Proceedings of the 3rd international conference on artificial intelligence and computer science (AICS2015)*, 2015, pp. 160-168.
[26] B. S. Lakshmi, "Fire detection using Image processing," *Asian Journal of Computer Science and Technology,* vol. 10, no. 2, pp. 14-19, 2021.
[27] H. S. Munawar, U. Khalid, and A. Maqsood, "Fire detection through Image Processing; A brief overview," in *Int. Conf. Cult. Technol*, 2017, pp. 203-208.
[28] N. Ya'acob, M. S. M. Najib, N. Tajudin, A. L. Yusof, and M. Kassim, "Image processing based forest fire detection using infrared camera," in *Journal of Physics: Conference Series*, 2021, vol. 1768, no. 1: IOP Publishing, p. 012014.
[29] A. M. Elshewey and A. A. Elsonbaty, "Forest fires detection using machine learning techniques," *Journal of Xi'an University of Architecture & Technology,* vol. 12, no. 9, 2020.
[30] A. Venugopal and R. Paul, "Machine Learning Based Early Fire Detection System," in *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, 2022: IEEE, pp. 1-6.
[31] N. F. M. Osman, A. A. A. Elamin, E. S. A. Ahmed, and R. A. Saeed, "Cyber-physical system for smart grid," in *Artificial intelligence paradigms for smart cyber-physical systems*: IGI global, 2021, pp. 301-323.
[32] J. Seebamrungsat, S. Praising, and P. Riyamongkol, "Fire detection in the buildings using image processing," in *2014 Third ICT International Student Project Conference (ICT-ISPC)*, 2014: IEEE, pp. 95-98.

[33] S. C. Pol, A. H. Wagh, P. T. Ramole, and S. H. Sharma, "Fire detection using image processing and sensors," *International Journal of Engineering Trends and Applications,* vol. 3, no. 2, pp. 85-87, 2016.
[34] A. Khan *et al.*, "PackerRobo: Model-based robot vision self supervised learning in CART," *Alexandria Engineering Journal,* vol. 61, no. 12, pp. 12549-12566, 2022.
[35] L. Elmoiz Alatabani, E. Sayed Ali, R. A. Mokhtar, R. A. Saeed, H. Alhumyani, and M. Kamrul Hasan, "[Retracted] Deep and Reinforcement Learning Technologies on Internet of Vehicle (IoV) Applications: Current Issues and Future Trends," *Journal of Advanced Transportation,* vol. 2022, no. 1, p. 1947886, 2022.
[36] M. Luppichini, M. Bini, and R. Giannecchini, "CleverRiver: an open source and free Google Colab toolkit for deep-learning river-flow models," *Earth Science Informatics,* vol. 16, no. 1, pp. 1119-1130, 2023.
[37] S. K. Shandilya, A. Srivastav, K. Yemets, A. Datta, and A. K. Nagar, "YOLO-based segmented dataset for drone vs. bird detection for deep and machine learning algorithms," *Data in Brief,* vol. 50, p. 109355, 2023.
[38] R. P. Nieuwenhuizen *et al.*, "Measuring image resolution in optical nanoscopy," *Nature methods,* vol. 10, no. 6, pp. 557-562, 2013.
[39] V. Nagpal and M. Devare, "Computer Vision in the Sky: Ultralytics YOLOv8 and Deep SORT Synergy for Accurate Vehicle Speed Monitoring in Drone Video," *Journal of Electrical Systems,* vol. 20, no. 10s, pp. 116-122, 2024.
[40] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction,* vol. 5, no. 4, pp. 1680-1716, 2023.
[41] S. Krum *et al.*, "Hiera: Separating Data from Code," *Pro Puppet: The Definitive Guide to Selling Abroad Profitably,* pp. 263-294, 2013.
[42] O. O. Khalifa, A. A. Omar, M. Z. Ahmed, R. A. Saeed, A. H. A. Hashim, and A. N. Esgiar, "An automatic facial age proression estimation system," in *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, 2021: IEEE, pp. 1-6.

Appendix-1

Configuring the YOLOv8 Model

```
!nvidia-smi

!pip install ultralytics

from ultralytics import YOLO

!yolo task = detect model = train model = yolov8n.pt data = /content/drive/MyDrive/data/data.yaml epochs = 15 imgsz =640
```

https://colab.research.google.com/drive/1lEr0IAir2xq18JgsTS-8i_s0KZ3mVVZd#scrollTo=p3yE7A_i0Zs4

Appendix-2

Test code

```
import cv2

from torch.fx.experimental.unification.multipledispatch.dispatcher import source

from ultralytics import YOLO

import cvzone

import pandas as pd

import time

# Load model and class names

model = YOLO("best.pt")  # Adjust path if needed

classNames = ['fire','smoke']

# Open video source (adjust path or use webcam as needed)

cap = cv2.VideoCapture(0)
```

```
# Create DataFrame for storing detections
df = pd.DataFrame(columns=['Class', 'Confidence', 'X1', 'Y1', 'X2', 'Y2'])
# Initialize variables for FPS calculation
prev_frame_time = 0
new_frame_time = 0
while True:
    new_frame_time = time.time()
    # Capture frame
    success, img = cap.read()
    if not success:
        print("Error reading frame from video stream.")
        break
    # Perform fire detection
    results = model(img, stream=True)
    for r in results:
        for box in r.boxes:
            # Extract bounding box coordinates, confidence, and class
            x1, y1, x2, y2 = int(box.xyxy[0][0]), int(box.xyxy[0][1]), int(box.xyxy[0][2]), int(box.xyxy[0][3])
            conf = float(box.conf[0])
            cls = int(box.cls[0])
            try:
                # Handle potential class index out of range
                if 0 <= cls < len(classNames):
                    # Draw bounding box and label using cvzone
                    cvzone.cornerRect(img, (x1, y1, x2 - x1, y2 - y1))
                    cvzone.putTextRect(img, f'{classNames[cls]} {conf:.2f}', (max(0, x1), max(35, y1)), scale=1, thickness=1)
                else:
                    print(f"Warning: Class index {cls} out of range. Using default label.")
```

```
                cvzone.putTextRect(img, "Unknown Class", (max(0, x1), max(35, y1)), scale=1, thickness=1)
                # Add detection to DataFrame
                df = pd.concat([df, pd.DataFrame({'Class': classNames[cls], 'Confidence': conf, 'X1': x1, 'Y1': y1, 'X2': x2, 'Y2': y2}, index=[0])], ignore_index=True)
            except IndexError:
                print("Error: Index out of range. Skipping detection.")
    # Calculate and display FPS
    fps = 1 / (new_frame_time - prev_frame_time)
    prev_frame_time = new_frame_time
    cv2.putText(img, f"FPS: {int(fps)}", (40, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    # Show the frame
    cv2.imshow("Image", img)
    # Exit on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release resources
cap.release()
cv2.destroyAllWindows()
```