Software Engineering Department

Braude College

Capstone Project Phase A-24-1-R-7

FireEye: AI-Powered Fire Detection System

Supervisor: Dr. Zeev Frankel

Majd Zbedat – Email: Zbedat.Majd2000@gmail.com

Rani Khoury – Email: ranikhoury20@gmail.com

Table of Contents

Abstract. Traditional fire detection methods often rely on human observation and sensor-detection devices in homes and public places. Though proficient, these methods are largely limited and often cause delayed responses. Leveraging the advancements in computer vision, our project implies integration of AI and machine-learning algorithms into video surveillance cameras and special infrared cameras to quickly discern rising flames from visual footage. The ability to analyze different data sources in real time and enhance the efficiency and accuracy of early fire detection will lead to rapid and precise identification of potential fire incidents.

Keywords: Fire detection, Computer vision, Dynamic prediction, Machine learning

1 Introduction

Throughout human history, fire has continued to plague humans, leaving devastating losses in human lives, properties, and revenues in its wake (Fig. 1). Over the years, many practical techniques have been implemented to combat fire incidents. Starting with the Corps of Vigiles in 6AD, an organization of men tasked with patrolling the streets with buckets filled with water as an emergency response, to the widespread adoption of electrical fire alarms and fire-detection sensors - humans have sought to protect themselves from fire.[8]

However, these techniques have serious limitations. Human monitoring of fire incidents has proven to be unreliable and unsatisfactory. Humans cannot see far, are forgetful, distracted, and may misjudge what they see. Electrical fire alarms and sensors offer better chances of fire detection but they are limited to location, effective mostly in indoor settings because they require close proximity to the fire. As a space gets larger, the performance of a sensor picking up flame signals drops.

Another limitation of electrical sensors is the delayed time for the sensors to pick up the fire flames or smoke [9]. This is called a transport delay. It takes a couple of minutes for carbon particles and smoke to reach the fire detector, depending on the intensity and size of the starting fire. This delay is dangerous and costs lives and properties. The sooner fire is detected, the better the chances of putting it out. Because of the limitations of the current detectors, it is crucial that fire detectors move to intelligent systems.[10]

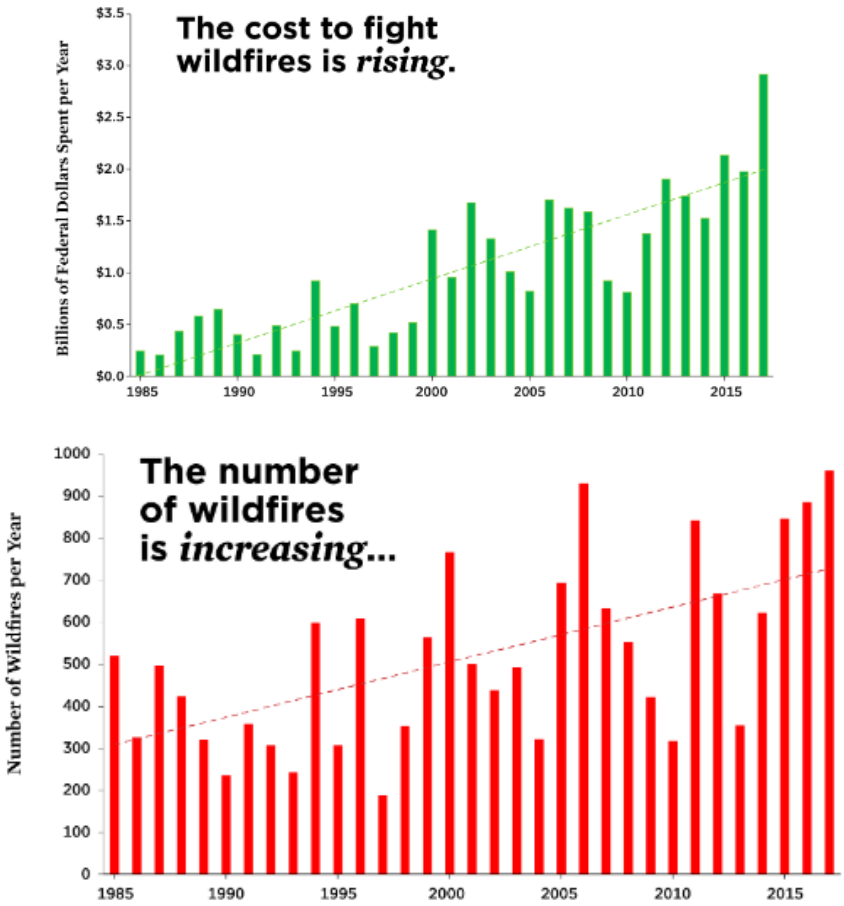


Figure 1: **Top:** Federal suppression costs from the National Interagency Fire Center[1] . **Bottom:** Data from the Monitoring Trends in Burn Severity program [1]. MTBS only includes large fires in the United States (>500 acres for the eastern US, >1000 acres for the west). Prescribed fires removed.

The motivation for this project is to build a more capable fire detection automation system that detects fire intelligently from video footage. Video surveillance cameras are everywhere, and can reach or see places where automated fire sensors cannot (Fig. 2). As the digital era unfolds, the integration of machine learning and AI into surveillance cameras offers a more promising solution to augment other methods of fire detection. The proposed solution, therefore,[11] is the development and evaluation of an AI-based system designed to recognize distinctive visual cues associated with rising flames from video footage. Machine learning algorithms, specifically convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants, will play a pivotal role in analyzing vast amounts of data, learning from patterns, and making predictions on the spreading of fire (Fig. 3).

To implement this solution, we will utilize popular deep learning frameworks such as TensorFlow and PyTorch. These frameworks provide efficient implementations of neural network architectures and offer a wide range of pre-trained models that can be fine-tuned for fire detection tasks. Additionally, we will leverage computer vision libraries like OpenCV for image and video processing tasks, enabling us to extract relevant features and preprocess data for training our models.[12]

Fig. 3: Camera (360-degree) views of the surrounding landscape uses to detect wildfire with AI

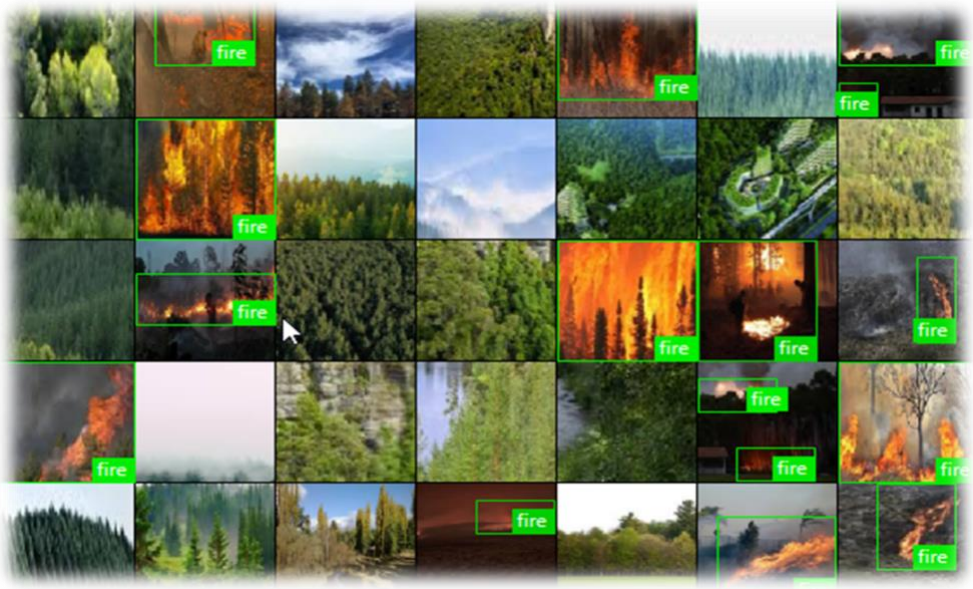


Figure 2. Examples of fire detection by computer vision

The integration of this technology into existing fire monitoring frameworks holds the potential to revolutionize the way we detect and respond to fires. Furthermore, our project contributes to the ongoing discourse on leveraging AI for public safety, emphasizing the significance of proactive fire detection strategies in an era characterized by increasing climate-related challenges.

2 Background and Related Work

2.1 Background

The intersection of machine learning and fire detection has garnered significant attention in recent years, with researchers and practitioners exploring various approaches to harness the power of AI in safeguarding against fire incidents. Noteworthy studies have focused on utilizing computer vision techniques for analyzing images and videos to detect smoke and flames accurately. Image recognition algorithms, such as convolutional neural networks (CNNs), have demonstrated promising results in identifying fire-related patterns, contributing to the development of intelligent surveillance systems (Fig. 4).

Fig. 4: AI Wildfire Alert Network to Detect Destructive Blazes [2]

In addition to visual data analysis, researchers have delved into the integration of sensor data, such as temperature, humidity, and gas concentration, to improve the accuracy of fire detection algorithms. Machine learning models applied to sensor data interpretation have shown the potential to discern anomalies associated with fire, enabling proactive responses and reducing false alarms.[13]

While significant strides have been made, challenges remain, including the need for large and diverse datasets for effective training, ensuring the robustness of algorithms across various environmental conditions, and addressing ethical considerations in deploying AI-based fire detection systems in video surveillance.

2.2 Related Work

Uses aerial imagery and monitoring systems helping first responders to mitigate fire quicker. With accurate data and proper fire detection models, the fire's behavior can be predicted thereby helping first responders contain the spread.

2.2.1 EfficientNetB0

EfficientNetB0 (Fig. 5) is the smallest variant of the EfficientNet family of models. Introduced by Tan and Le in 2019, EfficientNetB0 is designed to achieve a balance between efficiency and accuracy in image classification tasks. It utilizes a compound scaling method to systematically scale up both the depth and width of the network, as well as the image resolution, resulting in improved performance across a variety of scales. Despite its compact size, EfficientNetB0 typically achieves competitive accuracy on benchmark datasets like ImageNet while requiring fewer computational resources compared to many other models.[7]

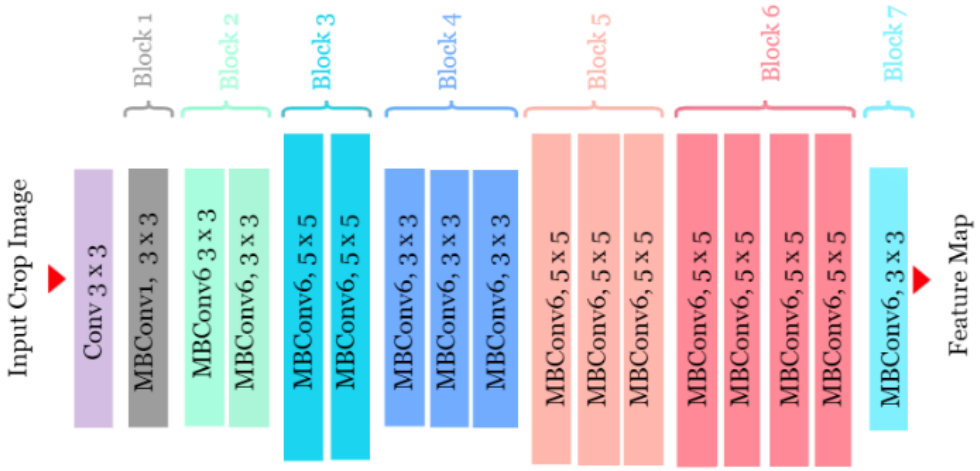


Fig. 5: EfficientnetB0 Model Architecture[3]

2.2.2 GoogLeNet

GoogLeNet, also known as Inception-v1, is a deep convolutional neural network (CNN) architecture developed by researchers at Google, notably Christian Szegedy et al., in 2014. It was one of the early CNN architectures that achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The defining feature of GoogLeNet is its inception module, which allows for parallel processing of different convolutional filter sizes within the same layer. This parallelization enables the network to capture features at multiple scales efficiently. GoogLeNet introduced the concept of network in network (NiN) modules, which use 1x1 convolutions to reduce the number of parameters and computational cost while increasing representational power (Fig. 6).

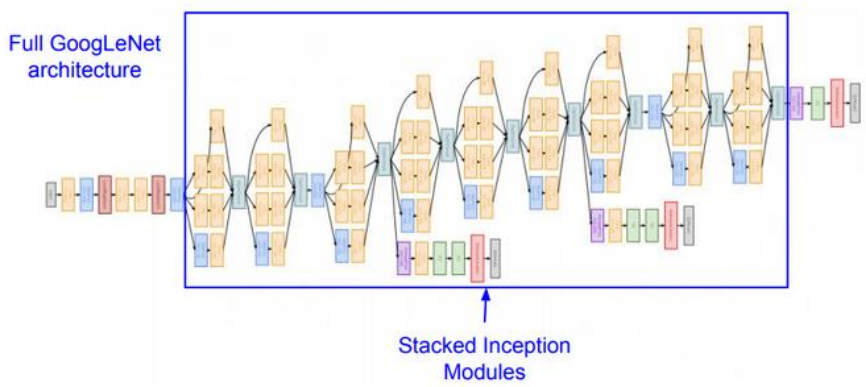


Fig. 6: GoogleNet - GM-RKB [4]

Overall, GoogleNet demonstrated significant improvements in both accuracy and computational efficiency compared to previous CNN architectures. It paved the way for subsequent versions such as Inception-v2, Inception-v3, and Inception-v4, each incorporating refinements and optimizations to further improve performance.

2.2.3 VGG16

VGG16 is a convolutional neural network (CNN) architecture introduced by the Visual Geometry Group (VGG) at the University of Oxford in 2014. It was developed by researchers Karen Simonyan and Andrew Zisserman. VGG16 is part of the VGG family of models and is named after its configuration with 16 weight layers, including 13 convolutional layers and 3 fully connected layers (Fig. 7).
The key characteristic of VGG16 is its simplicity and uniformity in architecture. It consists of a series of convolutional layers, each followed by a max-pooling layer, with fully connected layers at the end. VGG16 uses small 3x3 convolutional filters throughout the network, which allows it to capture detailed features at different scales.

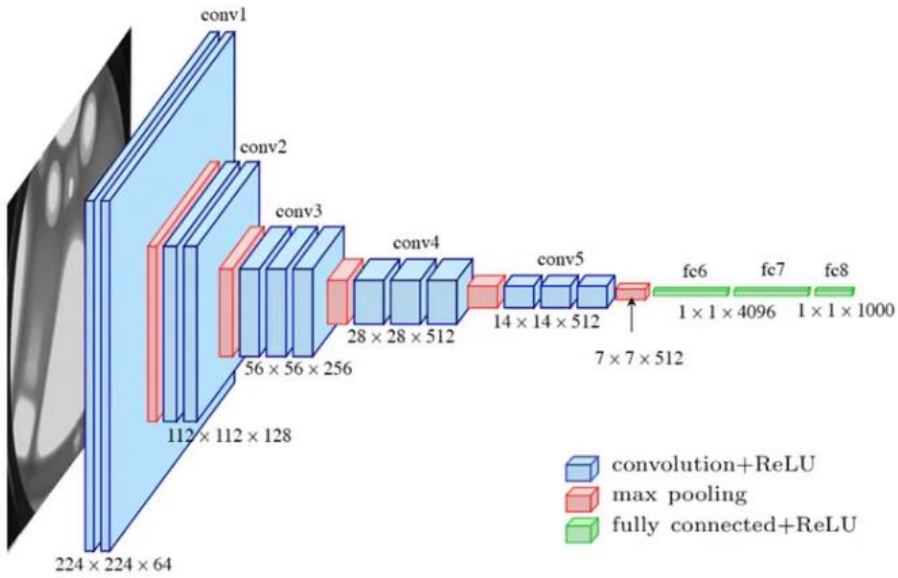


Fig. 7: **The architecture of VGG16**. [5] Source: Researchgate.net

While VGG16 has a straightforward architecture, it has demonstrated strong performance on various image classification tasks and served as a foundation for deeper and more complex CNN architectures. It was among the top-performing models in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. Additionally, VGG16 has been widely used as a feature extractor and pre-trained model in transfer learning applications.

2.2.4 ResNet50

ResNet50 is a convolutional neural network (CNN) architecture introduced by Microsoft Research in 2015 as part of the larger ResNet (Residual Network) family of models. It was developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. The "50" in ResNet50 refers to its depth, specifically consisting of 50 layers (Fig. 8). ResNet50 incorporates residual connections, which allow for the training of very deep neural networks by addressing the problem of vanishing gradients. These residual connections involve adding shortcut connections that skip one or more layers, allowing the network to learn residual mappings instead of directly fitting the desired underlying mapping. ResNet50 architecture comprises several residual blocks, each containing multiple convolutional layers. It also includes max-pooling layers and fully connected layers at the end for classification.

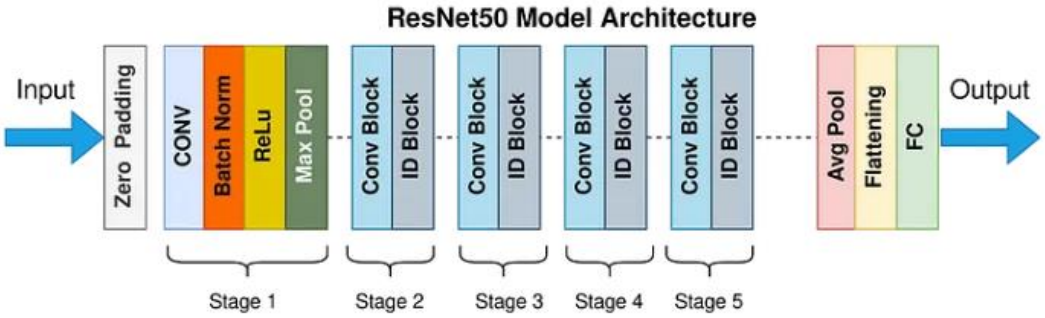


Fig. 8: Resnet-50 Model architecture [6]

ResNet50 has been widely adopted in various computer vision tasks, including image classification, object detection, and image segmentation. Its depth, along with the use of residual connections, enables it to achieve state-of-the-art performance on many benchmark datasets. Additionally, pre-trained versions of ResNet50 are commonly used in transfer learning scenarios, where the model's learned features are fine-tuned for specific tasks with limited labeled data.

2.3 Developing an Attention-Based CNN Model for Detecting and Locating Fires in Real-World Images

The authors propose an attention-based convolutional neural network, selecting the EfficientNetB0 model based on superior performance compared to other state-of-the-art models. The model's efficiency and lightweight nature make it an ideal choice. Utilizing transfer learning, slight architectural modifications including a dropout of 0.2 between dense layers were made to improve performance. Evaluation metrics including precision, recall, f-score, and accuracy were calculated, with the model achieving 95.4% accuracy, 91.77% precision, 97.61% recall, and 94.76% f-score over 20 epochs. While outperforming other models in accuracy and f-score, it ranked third in precision and second in recall. Despite this, its efficiency and reasonable performance led to its selection. Notably, the lack of publicly available datasets poses a challenge across various studies, impacting model performance as emphasized by Pereira et al [14] (Fig. 9). These statistical methods are calculated can be seen below.

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)} \quad (1)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (2)$$

$$Precision = \frac{TP}{(TP + FP)} \quad (3)$$

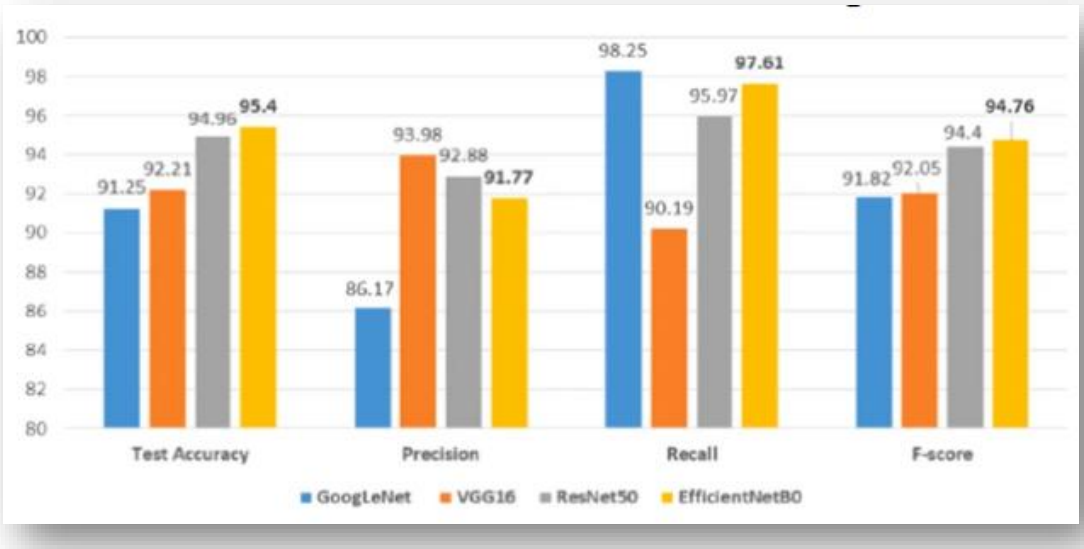


Fig. 9: Results from model comparison[7]

3 Algorithm

Pseudocode:

Fire detection algorithm:

First part: detecting fire

model = Load(pre-trained-model)

model = Add(custom layers for fire detection)

Compile(model)

dataset = Prepare(dataset (X_train, y_train))

dataset = Split(dataset) #into training and validation sets

Preprocess(images and labels)

Adjust(target_size) # based on the input size expected

trained_model = Train(model)

save(trained_model)

#Use the model for prediction

fire_frames = select(fire_video)

results = predict(trained_model, fire_frames)

Second part: fire spread and direction

Analyze(results):
Analyze_direction() # to predict fire spread direction
calculate_wind_speed() #calculate its speed
analyze_object_material() # to accurately calculate ignition

select_land_proprities() # to calculate fire navigation

4 Expected achievements and success criteria

Expected achievements

This project aims to achieve the following **outcomes**:

Early Detection and Rapid Response:

- The primary goal of a fire detection application is to provide early detection of fires based on captured videos. By identifying fires at an early stage, emergency services can be alerted promptly, leading to faster response times and potentially minimizing the extent of damage.

Improved Safety and Life Protection:

- The application aims to enhance safety. The quicker response enabled by the application can help protect lives and reduce the risk of injuries.

Reduced Economic Impact and Property Protection:

- Early detection and swift response can contribute to minimizing the economic impact of fires. By containing and extinguishing fires more quickly, the application may help reduce property damage, business interruption, and associated economic losses.

User-Friendly Interface:

- A well-designed user interface can make the application accessible to a broad audience. This includes both individuals seeking personal safety and organizations looking to enhance the protection of their assets.

Scalability and Adaptability:

- The application should be designed to scale and adapt to different environments and scenarios. This includes the ability to handle various types of data sources, integrate with different hardware setups, and accommodate updates and improvements over time.

By achieving these goals, a fire detection application using AI can make a significant positive impact on public safety, property protection, and emergency response efforts.

Success criteria

Defining **success criteria** for a fire detection application involves setting measurable and specific goals that align with the application's purpose and objectives. Here are some key success criteria for such an application:

1. **Detection Accuracy:**
   - Measure the accuracy of the application in correctly identifying fire incidents.
2. **Response Time:**
   - Evaluate the time it takes for the application to detect a fire and alert the relevant authorities or users.
3. **False Positive Rate:**
   - Assess the rate of false alarms generated by the application.
4. **Scalability:**
   - Ensure that the application can scale to handle different data sources, environments, and user loads.
5. **User Adoption and Satisfaction:**
   - Measure the adoption rate of the application among target users.
6. **Reliability and Availability:**
   - Assess the reliability and availability of the application.
7. **Feedback Mechanism and Continuous Improvement:**
   - Implement a feedback mechanism for users to report on the accuracy and performance of the application.

5 Research process

5.1 Motivation

The impetus for this investigation emanates from the imperative to augment methodologies associated with fire detection, a recognition underscored by the inherent deficiencies in extant systems. Fires persist as a ubiquitous threat, bearing substantial implications for both human safety and property integrity, thereby necessitating inventive resolutions. Traditional modalities, typified by human observation and conventional sensor deployment, manifest inadequacies typified by tardy responsiveness and restricted adaptability across diverse environmental contexts. The amalgamation of artificial intelligence (AI) and machine learning (ML) provides a prospect to overhaul fire detection paradigms through the assimilation of advancements in computer vision and sophisticated algorithms.
Furthermore, the genesis of my affinity for nature, cultivated through formative years immersed in natural environments, has instilled in me a fervor for the preservation of the natural world.

5.2 Research Process

We conducted comprehensive research to gain insights into existing methodologies and technologies related to fire detection using artificial intelligence.[7] Subsequently, we defined the specific objectives of our application, outlining key features such as real-time detection, scalability, and integration with emergency services. To build a robust dataset for training our AI model, we sourced diverse images and videos from publicly available datasets, simulated environments, and collaborated with local fire departments for access to controlled burn footage. Data preprocessing involved careful annotation and cleaning to ensure the quality and relevance of our training data. We selected a convolutional neural network (CNN) architecture, leveraging a pretrained model as a starting point for our fire detection model. The training process involved iterative optimization, with a focus on achieving high accuracy, low false positive rates, and minimal response time. In the future Throughout the developing phase, we will solicit feedback from potential users and iteratively refine the application based on their input. The final stage of our research will involve rigorous testing in simulated scenarios, to validate the application's performance according to predefined success criteria.[15]

5.3 Hyper Parameters

In deep learning training, hyper-parameter values profoundly influence model performance and require careful selection for optimization. This research will scrutinize and contrast various hyper-parameters, proposing an ideal set for segmentation tasks. Model accuracy will be evaluated through a comprehensive summary, alongside an examination of learning architectures to identify potential adjustments in learning rates or batch sizes for improved performance.

5.3.1 Learning rate

In segmentation tasks, the learning rate acts as a vital hyper-parameter, dictating the pace of model learning from training data. A high learning rate can expedite learning but risks overshooting the optimal solution, leading to subpar performance. Conversely, a low learning rate may slow learning but could yield a more optimal solution. This study will investigate the impact of varying initial learning rates, spanning from 0.000001 to 0.00001, employing Adam optimization method as recommended by our supervisor.

5.3.2 Epochs

In machine learning, an epoch signifies one traversal through the dataset during model training, with the number of epochs being a crucial hyper-parameter governing data exposure. While employing more epochs can enhance model performance, excessive epochs may result in overfitting, underscoring the importance of identifying the optimal number. This study will assess model performance using 50 and 100 epochs.

5.3.3 Loss function

In segmentation tasks, the selection of loss functions is pivotal for model performance. Among these, the Dice loss and Tversky loss are widely utilized and will be compared in this study.[14] The Dice loss quantifies the similarity between predicted and ground truth segmentation by computing the ratio of their intersection to their union, as expressed in Equation 1.

$$L_{dice} = 1 - \frac{2\sum_{i=1}^{N} p_i g_i + \epsilon}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} g_i + \epsilon} \quad (1)$$

Where, ϵ is a small number for avoiding dividing by zero. The Tversky loss is similar to the Dice loss and the Jaccard loss in that it measures the overlap between the predicted segmentation and the ground truth. However, the Tversky loss introduces two additional hyper-parameters, alpha and beta, which allow for more fine-grained control over the trade-off between precision and recall. The Tversky loss is defined as 2:

$$\mathrm{TL(p,t)} = 1 - \frac{p*t + (1-p)(1-t)}{p*t + \alpha*(1-p)*t + \beta*p*(1-t))} \quad (2)$$

5.3.4 Evaluation metric

We'll assess the model's accuracy using two evaluation metrics, with the first being the Jaccard Index, commonly used in segmentation tasks. It measures set similarity by evaluating intersection size. Comparing our predicted segmentation (p) with the ground truth (g), we will utilize the Jaccard Index to determine accuracy, defined by Equation 3.

$$\mathrm{JI} = \frac{\sum_{i=1}^{N} \mathrm{p_i g_i} + \epsilon}{\sum_{i=1}^{N} \mathrm{p_i} + \sum_{i=1}^{N} \mathrm{g_i} - \sum_{i=1}^{m} \mathrm{p_i g_i} + \epsilon} \quad (3)$$

ε represents a small number added to prevent division by zero. Another commonly employed evaluation metric for segmentation is the Dice coefficient, which we previously discussed in terms of the Dice coefficient loss. Utilizing the loss function outlined in Equation 1, the Dice coefficient (DSC) can be expressed as follows 4:

$$\mathrm{D_{sc}} = 1 - \mathrm{L_{dice}} \quad (4)$$

5.4 Expected Research Challenges

Dearth of a Diverse Database: The acquisition of an extensive and varied dataset for the training of a machine-learning model constitutes a significant challenge. The constrained availability of authentic data pertaining to real-world fire incidents necessitated exhaustive searches across diverse databases from various sources to ensure the model's reliability and the attainment of precise outcomes.

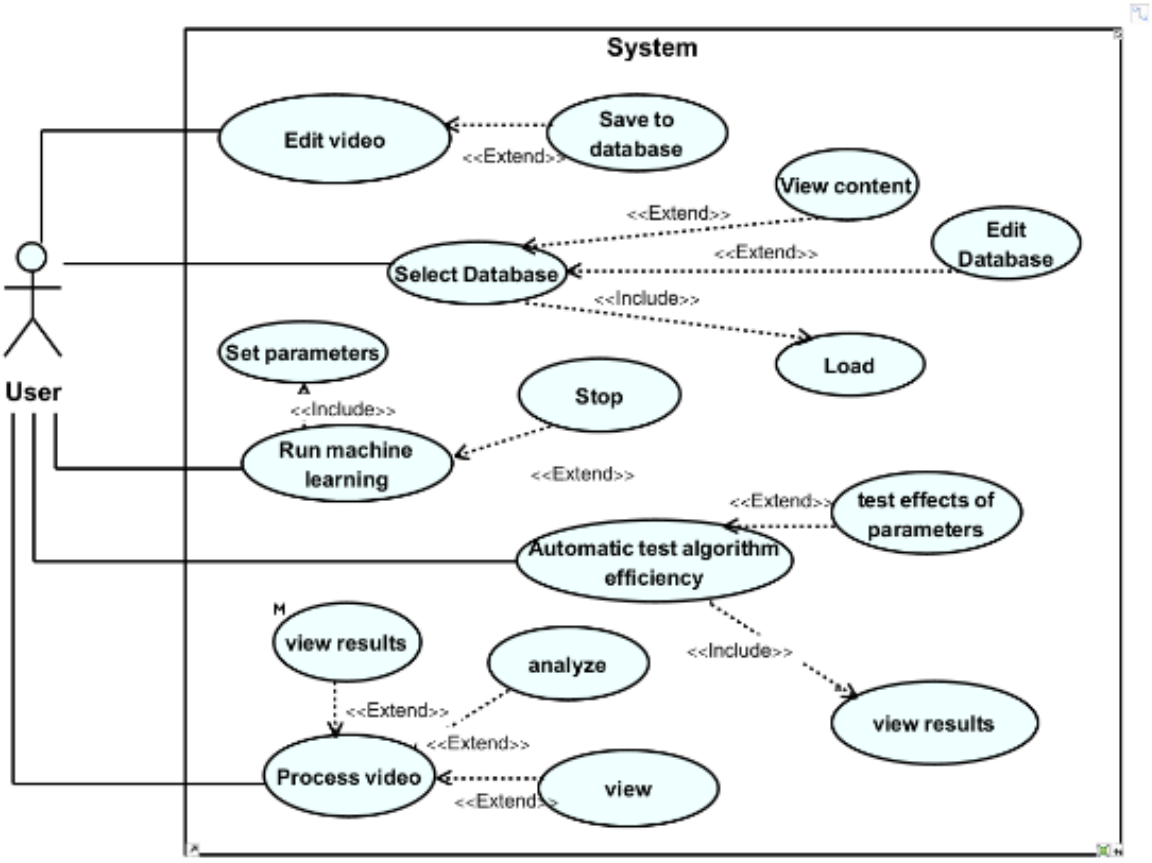
Algorithmic Selection Predicament: The selection of the most appropriate algorithm for fire detection introduced a complex dilemma. Striking a balance between the imperatives of accuracy and computational efficiency proved intricate, demanding a meticulous evaluation of diverse machine learning architectures and their performance characteristics within the specialized context of fire detection.

Validation Across Varied Environments: Ensuring the robustness and dependability of the proposed system across diverse environmental conditions emerged as a pivotal challenge. The adaptation of the algorithm to disparate settings, accommodation of varying lighting conditions, and consideration of potential variations in fire characteristics demanded extensive testing and validation efforts. These challenges elucidate the intricate nature of your research, underscoring the multifaceted complexities inherent in the development of an efficacious AI-based fire detection system.

6. Preliminary software engineering documentation

6.1 UseCase Diagram:

We will be using a use case diagram as a visual representation of the interactions between different elements in our system. This diagram will be similar to ones used in past projects and will help us understand and study how our system works. We have made changes to some of the existing algorithm properties and created a new optimized one which is more accurate, in order to improve the performance of our system.



In this use-case diagram, the user has the option to connect to a database and access a variety of databases, edit videos details, run machine learning, test algorithm efficiency and process video. Once the user has made his selections, he will need to set parameters for the algorithm for the system to run. Then, the system will analyze the results of the chosen algorithm and display them for the user to view. The user has the option to view the results and compare them to other existing algorithms.

6.2 Requirements:

Functional Requirements (FR):

The system shall detect fire incidents from video footage.

The system shall accurately localize the fire within the video frame.

The system shall enhance detection accuracy.

The system shall provide a user interface for viewing fire spread.

The system shall store video data.

Non-Functional Requirements (NFR):

The video footage will be classified to the appropriate database.

Localizing the fire will be within minimum possible time.

Enhanced detection accuracy is achieved by preprocessing video data.

The user interface should accessible to users with basic technical skills.

The video data and associated metadata will be stored in a database.

6.3 GUI:

Our GUI is built according to actions explained in the Use case diagram.

Main Page (Fig. 10): The main page of the GUI contains five main actions: Select database, Edit video, Process video, Run machine learning, Test algorithm efficiency.

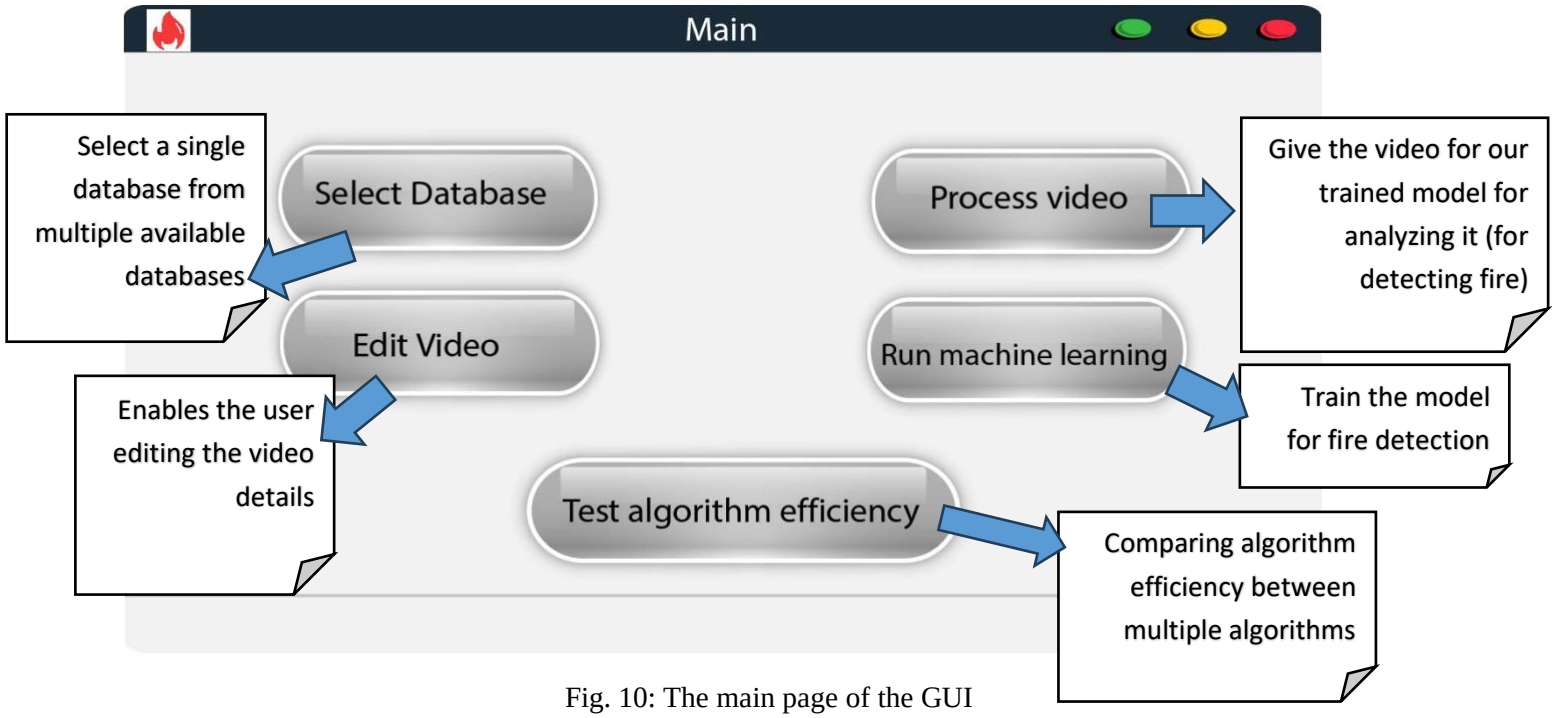


Fig. 10: The main page of the GUI

Edit video Page (Fig. 11): In this page, the user can edit the video's details and have an option to save it to the database.

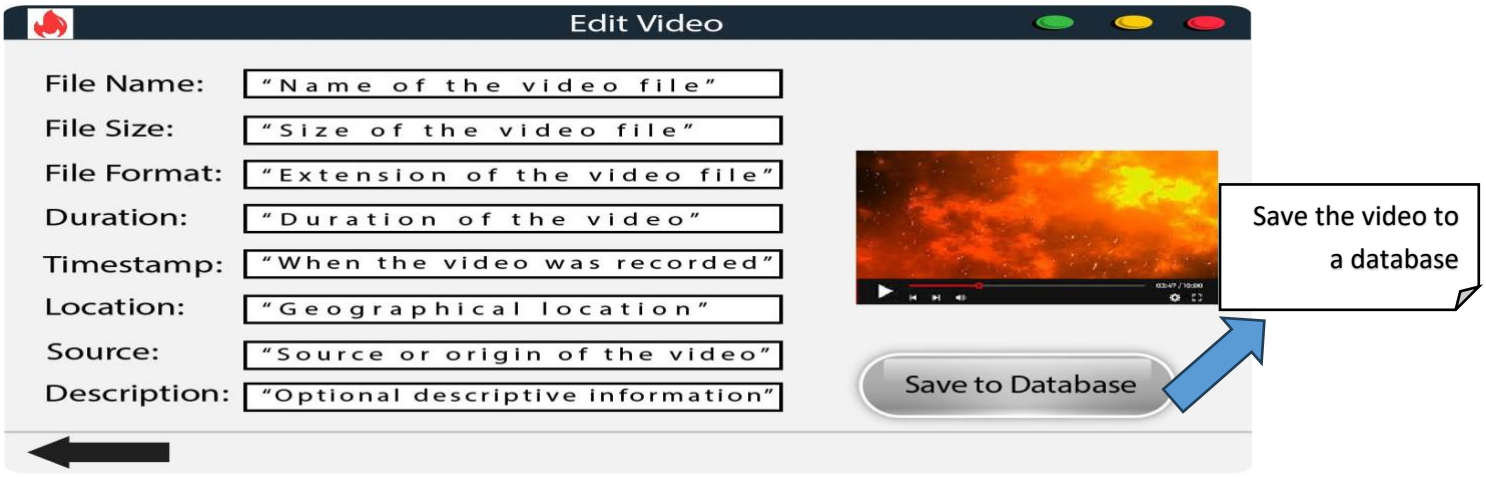


Fig. 11: Edit video Page.

Database Page (Fig. 12): The user can choose between multiple database options and then enter the password. The user can view the content or edit the selected database before loading it.

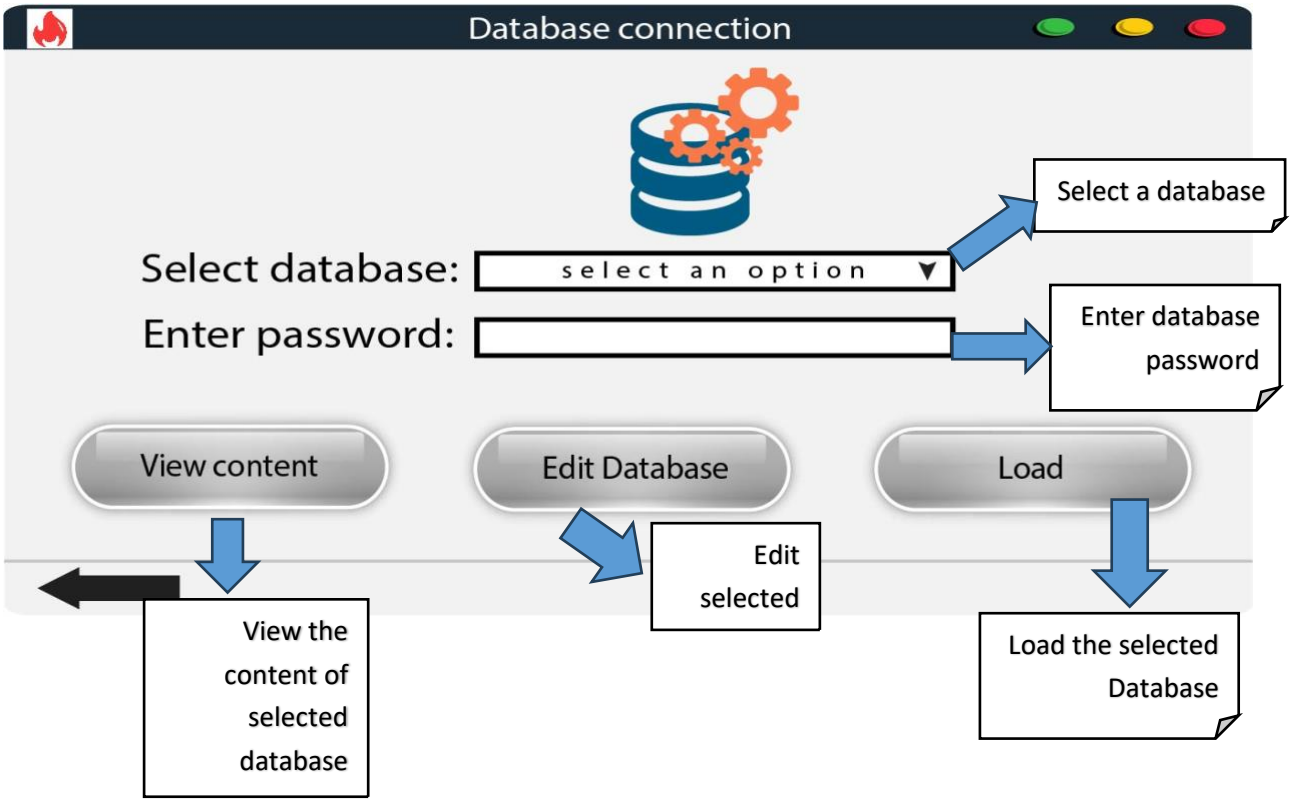


Fig. 12: Database connection Page.

Run machine learning Page (Fig. 13): In this page, the user can set different parameters which affects the training process of the algorithm. In addition we can stop the running.

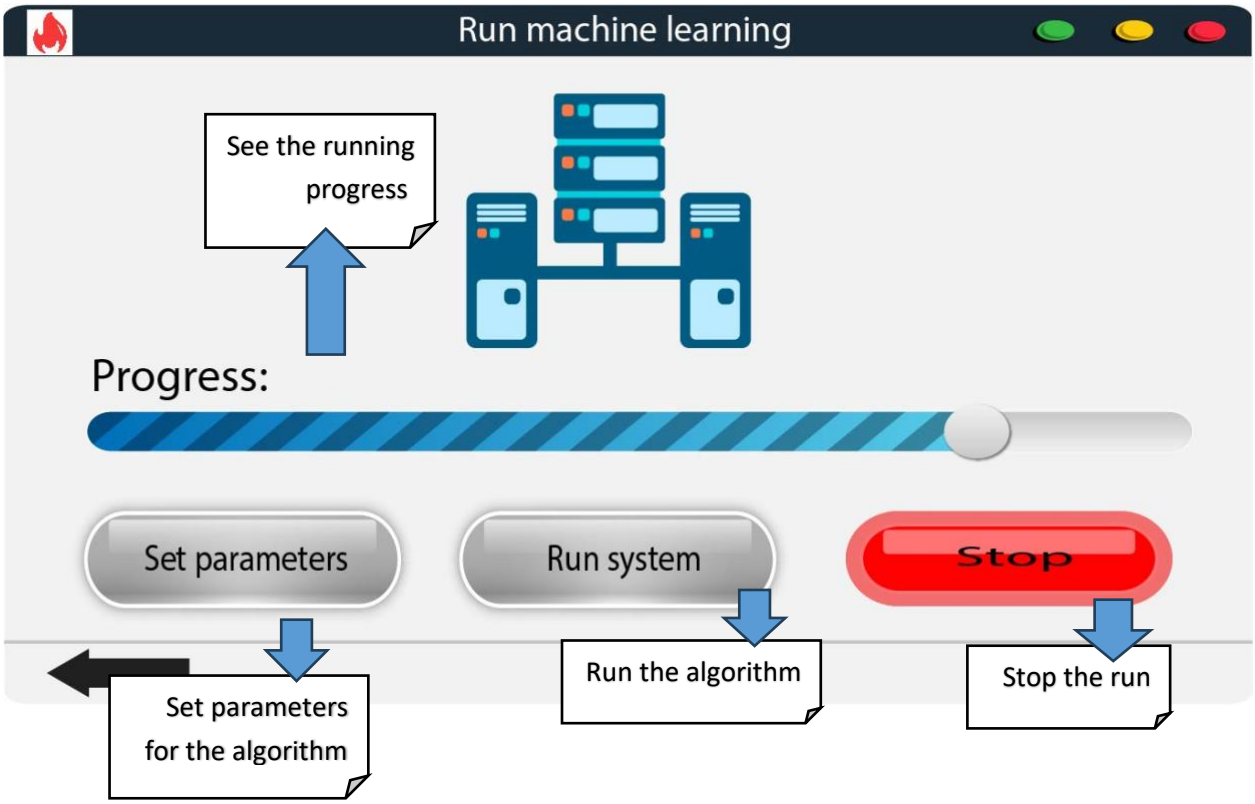


Fig. 13: Run machine learning Page.

Compute efficiency Page (Fig. 14): In this page, we can change parameters to watch how it will affect results and we can view the statistics of the algorithm running process.

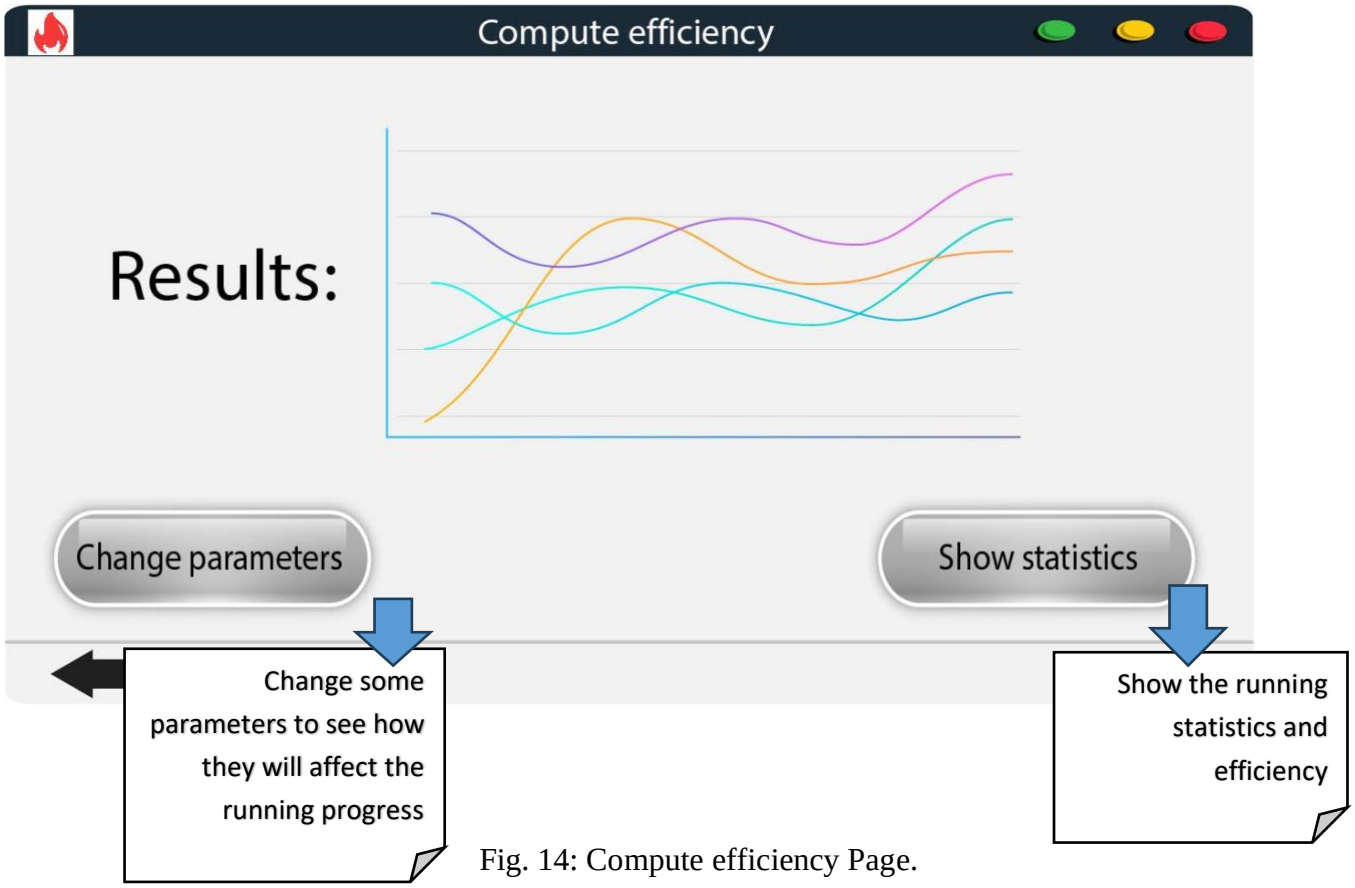


Fig. 14: Compute efficiency Page.

Process video Page (Fig. 15): In this page, we can view and select video stream that we want to run our algorithm on, after that we can analyze it and by the end viewing the analysis results.

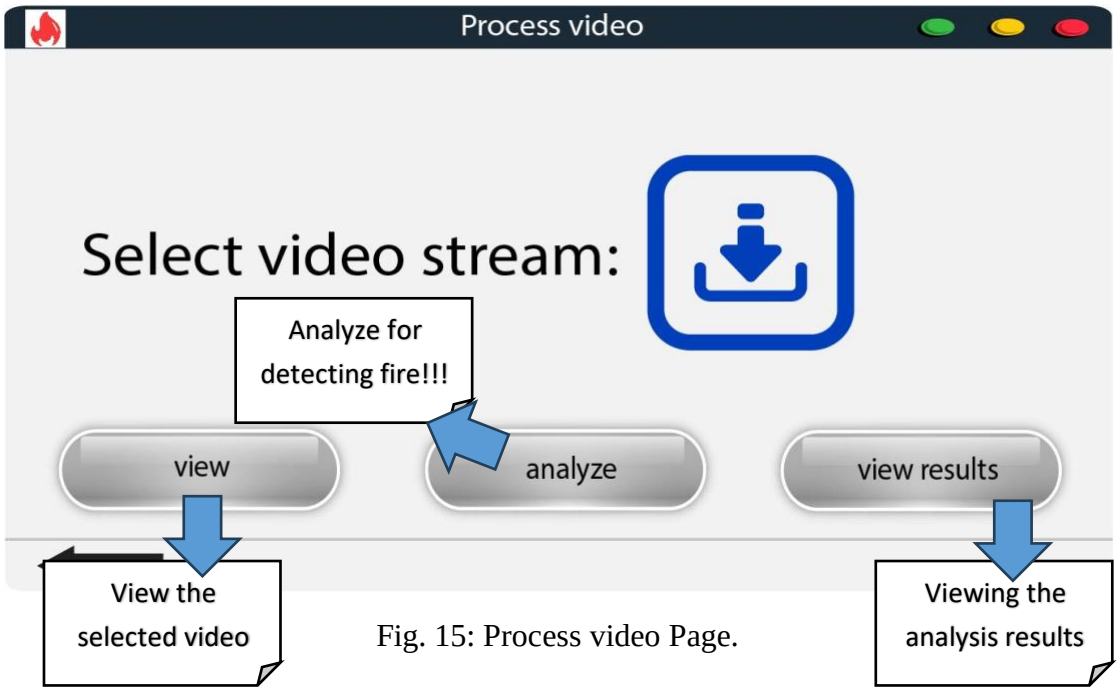


Fig. 15: Process video Page.

6.4 Verification plan:

Our testing approach will incorporate positive control tests, employing known inputs to validate the code's functionality and ensure accurate results are produced as expected.

6.4.1 GUI testing:

Testing GUIs created by Python typically involves a mix of automated and manual testing approaches. We will use PyAutoGUI to simulate user actions such as clicking, typing, and dragging to test the GUI.

Description	Steps	Expected results
Connect to database without filling at least one of the essential fields for establishing the connection	Clicking on “Connect” button without fill a required field	Alert:”Please fill all required details and then try again!!”
User try to process invalid video	Selecting corrupted data	Alert:”please choose valid video”
Run the system without selecting an algorithm	Click on “Run” without choosing an algorithm	Alert:”Please choose an algorithm”

6.4.2 Algorithm testing:

We have detailed some use cases in the following table and we intend to use the Pytest framework in order to test these use cases.

Description	Steps	Expected results
Database-Verify that the selected database successfully connected	Attempt to establish a connection to the selected database using the appropriate connection parameters such as host address, port number, username, and password.	Database connection is successfully established and server is running.
Input-Verify that the code can process the media which selected from database	Validate the selected media data to ensure it meets the expected format and specifications required by the code for processing.	The results meet our expected outcomes or predefined criteria to ensure correctness and accuracy.
Output- our program gives expected output or outcomes of the program based on its specifications, requirements, and objectives.	Prepare test data or input scenarios and check for errors, or deviations.	captured output is the same as the expected output.
Algorithm- Verify that the algorithm is able to identify the fire.	compare between expected outputs and actual output for each input data.	Returns True if there is fire in addition to its place and spread time. Else return False.
Algorithm- verify that the algorithm can handle different environments	using real-world data or data representative of the algorithm's intended usage environments.	The algorithm is scalable and adaptable to different environments.

7. References:

[1] Infographic: Wildfires and Climate Change: Visualizing the Connection in Five Sets of Photos and Charts. https://www.ucsusa.org/resources/infographic-wildfires-and-climate-change

[2] California's Sonoma County Adds AI to Wildfire Alert Network to Detect Destructive Blazes. https://www.enterpriseai.news/2021/03/19/california-adds-another-ai-tool-to-its-wildfire-alert-network/

[3] PhytoCare : A hybrid approach for identifying. https://www.researchgate.net/figure/EfficientnetB0-Model-Architecture_fig2_378395574

[4] GoogLeNet: Introduced by Szegedy et al. in Going Deeper with Convolution. https://www.gabormelli.com/RKB/GoogLeNet

[5] VGG16: https://neurohive.io/en/popular-networks/vgg16/

[6] ResNet50: Introduced by Microsoft Research in 2015, Residual Networks (ResNet in short) broke several records when it was first introduced in this paper by He. et. al.

[7] Using Neural Networks to Detect Fire from Overhead Images: https://link.springer.com/article/10.1007/s11277-023-10321-7#Fig1

[8] Khan, S. S., & Zaheer, A. (2018). A review on fire detection techniques. Fire Science Reviews, 7(1), 1-34.

[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[10] Varley, J. (2019). Artificial Intelligence in Fire Detection Systems. Fire Safety Science, 12, 247-258.

[11] Ma, J., & Perkins, S. (2018). Fire Detection in Video Scenes Based on Convolutional Neural Networks. IEEE Transactions on Circuits and Systems for Video Technology, 28(9), 2186-2200.

[12] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[13] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.

[14] Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295-2329.

[15] Jain, A., & Manda, K. (2018). A Comprehensive Survey of Fire Detection Methods using Image Processing. IETE Journal of Research, 64(2), 137-148.