

EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

NALAIYA THIRAN PROJECT REPORT

IBM–PROJECT–12478-1659451972

TEAM ID: PNT2022TMID08418

A PROJECT REPORT

submitted by

MANIKANDAN.S (814319104026)

SANTHOSH.R (814319104045)

SARAVANAN.S (814319104048)

VINOTH.R (814319104060)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE

(AUTONOMOUS)

PERAMBALUR-621 212

- 1. INTRODUCTION**
 - 1.1 Project Overview
 - 1.2 Purpose
- 2. LITERATURE SURVEY**
 - 2.1 Existing problem
 - 2.2 References
 - 2.3 Problem Statement Definition
- 3. IDEATION & PROPOSED SOLUTION**
 - 3.1 Empathy Map Canvas
 - 3.2 Ideation & Brainstorming
 - 3.3 Proposed Solution
 - 3.4 Problem Solution fit
- 4. REQUIREMENT ANALYSIS**
 - 4.1 Functional requirement
 - 4.2 Non-Functional requirements
- 5. PROJECT DESIGN**
 - 5.1 Data Flow Diagrams
 - 5.2 Solution & Technical Architecture
 - 5.3 User Stories
- 6. PROJECT PLANNING & SCHEDULING**
 - 6.1 Sprint Planning & Estimation
 - 6.2 Sprint Delivery Schedule
 - 6.3 Reports from JIRA
- 7. CODING & SOLUTIONING (Explain the features added in the project along with code)**
 - 7.1 Feature 1
 - 7.2 Feature 2
 - 7.3 Database Schema (if Applicable)
- 8. TESTING**
 - 8.1 Test Cases
 - 8.2 User Acceptance Testing
- 9. RESULTS**
 - 9.1 Performance Metrics
- 10. ADVANTAGES & DISADVANTAGES**
- 11. CONCLUSION**
- 12. FUTURE SCOPE**
- 13. APPENDIX**
 - Source Code
 - GitHub & Project Demo Link

EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

ABSTRACT

Forest-fires are real threats to human lives, environmental systems and infrastructure. It is predicted that forest fires could destroy half of the world's forests by the year 2030. The only efficient way to minimize the forest fires damage is adopt early fire detection mechanisms. Thus, forest-fire detection systems are gaining a lot of attention on several research centers and universities around the world. Currently, there exists many commercial fire detection sensor systems, but all of them are difficult to apply in big open areas like forests, due to their delay in response, necessary maintenance, high cost and other problems. In this study, image processing based has been used due to several reasons such as quick development of digital cameras technology, the camera can cover large areas with excellent results, the response time of image processing methods is better than that of the existing sensor systems, and the overall cost of the image processing systems is lower than sensor systems. Accurate forest fires detection algorithms remain a challenging issue, because, some of the objects have the same features with fire, which may result in high false alarms rate. This project presents a new video-based, image processing forest fires detection method, which consists of four stages. First, a background-subtraction algorithm is applied to detect moving regions. Secondly, candidate fire regions are determined using RGB color space. Thirdly, features extraction is used to differentiate between actual fire and fire-like objects, because candidate regions may contain moving fire-like objects. Finally, convolutional neural network algorithm is used to classify the region of interest to either real fire or non-fire. The final experimental results verify that the proposed method effectively identifies the forest fires.

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Forest fires are natural phenomena that occur regularly on earth, a large quantity of forest area and wildlife are destroyed annually due to the forest fires. It causes drastic loss of lives and valuable natural resources and individual properties. The forest fire has significant effects on the global climate. The problem has become more severe than previous years. The human encroachment over forest areas is a major cause of forest fire. It is very essential to detect and avoid the fire at its initial state itself. Traditional fire protection methods use mechanical devices or humans to monitor the surroundings. The most frequently used fire smoke detection techniques are usually based on air transparency testing, particle sampling, and temperature sampling. An alarm is not raised unless the particles reach the sensors and activate them. The huge leap in image processing technology and the reduced cost of digital cameras make fire detection based on image processing more feasible than any other traditional methods like fire watch towers, sensors, satellites etc. In the case of fire watch towers, humans are made to observe the location throughout. The main limitation of this approach is the lack of accuracy due to operator fatigue, time of day, geographic location etc. The sensors are devices capable of sensing their environment and computing data.

1.2 PURPOSE

The sensors sense physical parameters like pressure, temperature and humidity, and chemical parameters such as carbon monoxide, carbon dioxide, and nitrogen dioxide. In a wireless sensor-based fire detection system, coverage of large areas in forest is impractical because of the requirement of regular distribution of sensors in close proximity and the battery charge is also a big challenge. Satellites based system can monitor a wide area, but the resolution of satellite images is low. A fire is detected when it has grown quite a lot, so real time detection cannot be provided. Moreover, these systems are very expensive. Weather condition (e.g. clouds) will seriously reduce the accuracy of satellite based forest fire detection.

2. LITERATURE REVIEW

2.1 EXISTING PROBLEM

2.2.1 Ahmed M. Elshewey et al., In the modern era, forest fires have emerged as one of the most significant issues that harm many regions worldwide. The study demonstrates machine learning regression techniques for identifying locations that are vulnerable to forest fires. The UCI machine learning repository contains the climate data set that was used in this paper and the natural elements of Portugal's Montesinos park. With a data set size of 517 entries and 13 features for each row, this study suggests three machine learning algorithms: linear regression, ridge regression, and lasso regression. This document employs two versions; the first version contains all features, while the second version contains just 70% of the features. This document employs two versions; the first version contains all features, while the second version contains just 70% of the features. The training set, which makes up 70% of the data set in the study, and the test set, which makes up 30% of the data set Ridge regression and lasso regression are less accurate than the linear regression technique in terms of accuracy. Physical models and mathematical models are just two of the methods available today for predicting the spread of fires . These models rely on modelling, lab testing, and data collection during forest fires. Efforts to define and forecast the spread of fire in various areas. In the recent past, simulation techniques have been employed to forecast forest fires, but these systems have encountered various issues, such as input data accuracy and execution time . A branch of artificial intelligence (AI) called machine learning is used to teach computers new things. Two categories of machine learning exist: supervised, unsupervised, and reinforcement learning. In supervised learning, a supervisor is there to inform the algorithm as to whether a choice or action is wise or not.

2.1.2 Binh Thai Pham et al., Forested areas are susceptible to the most severe natural disasters due to fires, which annually burn millions of hectares and cause losses in biodiversity, soil quality, and CO₂ capture. Communities in various land ecosystems across the world are deeply concerned about how easily fires can spread to nearby woods and human populations and infrastructure. Authorities and decision-makers are under pressure to define the forested areas in terms of their susceptibility to fires on a temporal and spatial scale as a result of increased changes in socioeconomic processes, the climate, and the natural environment. Designing fire control strategies and allocating fire fighting resources properly requires identifying regions with high/very high fire susceptibility . Despite the

extensive use of these techniques, the susceptibility of many parts of the world to fire has not yet been identified. Due to the variance in training data from different regions, no one model or method has yet been found to capture fire behaviour in all regions. Our study's goal was to create a set of predictive models based on the four machine learning techniques of Bayes Network, Naive Bayes, Decision Tree, and Multivariate Logistic Regression for the prediction of fire susceptibility in the Pu Mat National Park of Vietnam. This study filled a significant gap in fire prediction efforts.

2.1.3 Recognizing forest fires is crucial for safeguarding forest resources. Multiple monitors must be deployed from various angles in order to successfully monitor forest fires. However, the majority of conventional recognition models can only identify photos from a single source. A high proportion of false positives and negatives is caused by the neglect of multi-view images. This research suggests a graph neural network (GNN) model based on the feature similarity of multi-view photos to increase the accuracy of forest fire recognition. In particular, the input properties of graph nodes were converted into the correlation features between various photos by establishing correlations (nodes) between multi-view images and library images. Additionally, a fire area feature extraction approach based on picture segmentation was created with the goal of streamlining the difficult preprocessing of images and successfully extracting the important characteristics from them. The fire region was recovered from the photos by setting a threshold in the hue-saturation-value (HSV) colour space, and the dynamic features were extracted from the fire area's continuous frames. The results of the experiments demonstrate that our technique spotted forest fires 4% more accurately than the baselines. The ecosystem of woods is seriously threatened by forest fires. Early detection of the fire source before it develops into a catastrophe is crucial for preventing the spread and dangers of the fire. Forest fire monitoring based on computer vision has been a popular topic among researchers studying forest fire prevention due to the quick development of computer vision. One of the first methods for recognising fires is colour recognition. Based on the colour mode's movement, space, and time characteristics, this method may identify the flame. However, colour recognition can only quickly identify huge flames.

2.1.4 Mounir Grari et al..., Even after a wildfire has been put out, its effects can still undermine the public's health and prosperity. The world's biodiversity is under grave jeopardy as wildfires are becoming both more frequent and more intense. People, businesses, and governments are all suffering financially as a result of the fires. With the help of recent developments in computer vision, machine learning, and remote sensing technologies, researchers are creating novel methods for spotting and monitoring wildfires.

Active forest fire detection has become more effective thanks to IoT sensors. Natural defenders of the planet's ecological balance are forests. Unfortunately, forest fires are frequently only detected after they have grown widely, making control and extinguishment more challenging, and in some cases impossible. 30% of the carbon dioxide (CO₂) in the atmosphere is produced by forest fires, which causes catastrophic losses and irreparable harm to the environment. Unplanned, unwelcome, and uncontrolled wildfires. These fires begin in rural settings (like forests) with a small amount of flammable foliage and spread quickly due to winds and high temperatures. The majority of them are typically results of poor human behaviour, while other wildfires' causes are still unclear. Wildfires have a significant influence on a variety of industries. They can interfere with water supply, power and gas services, communications, and transportation. A significant portion of our planet's ecological balance is provided by forests. These essential natural barriers are, however, gravely in jeopardy. Large areas are frequently affected by wildfires, making control and extinction nearly impossible. These catastrophes happen suddenly, are unanticipated, and can be brought on by mankind, climate change, or even lightning. The risk of wildfires disrupting power, gas, water, communications, transportation, or other systems is very high. Air, crops, resources, animals, and people could all suffer consequences.

2.1.5 Disasters caused by forest fires are currently receiving a lot of attention due to global climate change. Global climate change is dramatically altering Earth's fire patterns. Accurate knowledge of the fire's origin, spread, and environmental effects is necessary for effective fire management. The authorities are guided by predictions of fire activity in the forest when making the best, most effective, and ethical fire management choices. This study intends to highlight current developments in the mapping of burned areas, detection, and propagation rates of forest fire events. Additionally, smoke emissions from fires increase the risk to human health and the natural system of the planet. Future policymaking can therefore be more precise in terms of saving billions of dollars and enhancing the ecological cycle and healthy environment for Earth's population. The use of various machine learning methods in managing forest fires or wildfires is thoroughly reviewed in this work. We have also highlighted some possible areas where new data and technology can aid in making better fire management decisions. Man is destroying forest land to make room for agriculture, industrial development, or the extraction of minerals and oil. In some areas, forest degradation is also done to make room for the construction of modern cities. Forest fires are one of the most frequent natural disasters in the world, and not all deforestation is intentional. Millions of hectares of forest are destroyed by fires every year. Fires are a natural occurrence that have significant negative effects on the economy, the environment, and social culture both globally and locally.

2.2 REFERENCES

- [1] Ahmed M. Elshewey, Forest Fires Detection Using Machine Learning Techniques,2020
- [2] Binh Thai Pham, Performance Evaluation of Machine Learning Methods for Forest Fire Modeling and Prediction,2022
- [3] Di Wu1, Forest Fire Recognition Based on Feature Extraction from Multi-View Images ,2021.
- [4] Mounir Grari, Early wildfire detection using machine learning model deployed in the fog/edge layers of IoT,2022.
- [5] Muhammad Arif, Role of Machine Learning Algorithms in Forest Fire Management: A Literature Review,2021

2.3 PROBLEM STATEMENT DEFINITION

There are currently methods for detecting fires and other hazards that use heat, temperature, smoke, or a combination of these sensors. These are mounted at heights, typically at floor level (or ceiling level). These contain separate sensors that are not aligned with one another, which causes alarm behaviour to be unpredictable and asynchronous. A smoke detector is a gadget that detects smoke, usually as a fire indicator. When smoke is detected, smoke alarms, also known as fire alarm systems, often sound or light a local alert. Smoke detectors are typically used in fire alarm systems since it is assumed that a fire will produce smoke. If smoke is found, a fire has been found. Even if there is a fire, the smoke may not be produced for some time after the surroundings have burned. Smoke may not always be produced by fires, or it may take a while for smoke detectors to pick it up.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

2

Brainstorm
Write down any ideas that come to mind that address your problem statement.

00 minutes

MANIKANDAN, S

CO2/Temperature sensors are deployed throughout the coverage area. CO2 levels can be monitored every 15 minutes (along with temperature, battery status etc).

Heat detectors are the most basic detection devices. They are available in several types. These types are spot and line.

The gateway provides sufficient coverage for up to 15KM or more outdoors, providing low power wireless access to the network of sensors.

The network can detect fires quickly while consuming energy efficiently.

SANTHOSH, R

In preprocessing unwanted distortions are removed and image is rectified and transformations of rectified image is performed.

High frequencies of an image are eliminated using SWT and the reconstruction of image are done by inverse SWT.

Uploading sensor data, the gateway transmits the data back via satellite in an optimized manner to a cloud platform dashboard.

This method involves three steps processing: SWT, histogram analysis.

SARAVANAN, S

In recent history and even the present day, several forest fire detection methods have been implemented such as watchtowers and video image processing methods are used.

Spot detectors are single units installed in single locations throughout the protected area by detecting the forest fires.

Forest fire detection uses the technique of optical sensors and digital camera based.

By using the video based cameras can be used to detect fire and with particular algorithms can detect hotspots with fire, smoke as well as flames for both detection and prediction of fire with less effort.

VINOTH, R

Sensor technology has been widely used in fire detection usually depending on sensing physical parameters such as changes in pressure, humidity and temperature, as well as chemical parameters such as carbon dioxide, carbon monoxide and nitrogen dioxide.

The first factor is the rapid development of digital camera technology and CO2 or CMOS digital camera which has resulted in a rapid increase in image quality and decreased cost of the cameras.

The second factor is that digital cameras can cover large areas with excellent results.

Third, the response time of image processing results is better than that of existing sensor models. Finally the overall cost of image processing systems is lower than existing systems.

3.3 PROPOSED SOLUTION

Fires are one of the biggest challenges in the world right now, due to the global warming that the planet is currently suffering from. We all know what fires are and what they are capable of causing great damage, whether to humans, animals, or other forms of life. Fire damages vary and these damages depend greatly on the cause of the fire, but no matter how many and different causes, the damage may be devastating on a large scale and difficult to predict, and in general, fire losses are divided into loss of lives and loss of Money. In this project, we proposed an algorithm which combines colour information of

the fire with the edge of the fire information. Then with the combined results from both this techniques, a parameter is created to segment out the necessary details from the images to detect and identify the fire. In this project propose a system which automatically detects the presence of fire based on the deep learning algorithm. Deep learning is a branch of machine learning that depends entirely on neural networks, as the neural network will simulate the human brain, so deep learning is also a kind of simulation of the human mind. In this proposed system we can implement preprocessing steps to eliminate the noises in images. And also implement features extraction to extract the color features and segment the fire regions. Finally classify the pixels using deep learning algorithm with efficient mobile alert system to corresponding authorities.

3.4 PROBLEM SOLUTION FIT

There are currently methods for detecting fires and other hazards that use heat, temperature, smoke, or a combination of these sensors. These are mounted at heights, typically at floor level (or ceiling level). These contain separate sensors that are not aligned with one another, which causes alarm behaviour to be unpredictable and asynchronous. We can use preprocessing techniques in this proposed system to get rid of image noise. Additionally, use feature extraction to separate the zones of fire and extract colour features. Finally, use a deep learning algorithm to categorise the pixels and alert the appropriate authorities via an effective mobile alert system.

Project Title: Emerging Methods For Early Detection Of Forest Fires		Project Design Phase-I - Solution Fit Template		Team ID: PNT2022TMID08418
Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Governments, Organizations	2. CUSTOMER ON IT RASTI Lack of resources Lack of data Rural area to monitor	3. AVAILABLE SOLUTIONS Aerospace Remote Sensing By detecting the emission of CO2	Explore AS, differentiate
	4. JOB-TO-BE-DONE / PROBLEM Forest fire is being detected and informed to the users of the system to prevent loss.	5. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the top? Carelessness of people who start the forest Lightening Volcanic activities Terrorist attacks Wildfire of the forest	6. BEHAVIOUR Detect the fire which is caused on the forest and the officers or the organizationally starting them.	
Focus on JAR, up to the	7. TRIGGERS Detect the forest fire and the system to prevent loss.	8. YOUR SOLUTION Applying the forest fire camera are fixed at the forest. OpenAI is used for processing the video, by detecting the image and applying image processing techniques and deep learning techniques the model is being trained. It alerts the user when the fire is being detected.	9. CHANNELS OF BEHAVIOUR 1. ONLINE Keep track of the forest through CCTV camera. 2. OFFLINE Broadcasting video.	Focus on JAR, up to the
	10. EMOTIONS: BEFORE / AFTER Before: No recognition After: Recognition Before: No action After: Action Before: No of loss After: Prevention of loss			

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

IMAGE SET UPLOAD

Early detection of wildfires is critical to the safety and security of environmental spaces and is one of the important and most large challenges in the government sector and forest fire managers. Forest fire is the important one is decreasing the space of the forest area. This fire detection technique also reduces human protocols and helps to monitor and protect the areas that are hard to protect. The new technique is used to facilitate the implementation of systems that allow monitoring is efficient in detailed areas, regardless of the state of the atmosphere or daytime. In this module, we can upload the image or videos which are captured from CCTV footages in forest. The satellite sensor is used to capture the forest fire image, but it has an increasing range of time resolution and space of the forest area. Satellite images also gave a fire-monitoring tool, management, and finding the damaged tool for compliance with burn areas to understand a favorable fire range. The principle of classifying this fire, such as materials from the original fire, is to check the color consistency. The proposed algorithm has rectified this problem and reduces the error. It not only detects fires but also distinguishes fires such as fire and materials. The parameters that were adopted in our proposed system operation to analyze the forest fire, threshold value, the detection of matrix value, and the differential matrix value of the system. If the input is video means, convert the videos into frames or if the input is images means, it can be any format or size

MEDIAN FILTERING

Pre-processing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightnesses). Image pre-processing methods are classified into four categories according to the size of the pixel neighborhood that is used for the calculation of new pixel brightness. The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, and translation) are classified among pre-processing methods here since similar techniques

are used. The user has to select the required lung frame image for further processing. Then each image is resized to 256*256. Then implement median filter to remove noises from lung images. The median filter is a nonlinear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise (but see discussion below), also having applications in signal processing. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. It is particularly effective at removing 'salt and pepper' type noise. The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels. The pattern of neighbors is called the "window", which slides, pixel by pixel over the entire image pixel, over the entire image. The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value. In this module, convert the RGB image into gray scale and also implement median filtering algorithm to remove the noises in images

COLOR FEATURES EXTRACTION

Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power or a classification algorithm which over fits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. Texture tactile or visual characteristic of a surface. Texture analysis aims in finding a unique way of representing the underlying characteristics of textures and represent them in some simpler but unique form, so that they can be used for robust, accurate classification and segmentation of objects. Though texture plays a significant role in image analysis and pattern recognition, only a few architectures implement onboard textural feature extraction. In this module implement color and texture features are implemented. HSV color features are extracted and Texture features include statistical features using Grab-cut method. In this

module, skin images are segmented using snake model. A snake is an energy minimizing, deformable spline influenced by constraint and image forces that pull it towards object contours and internal forces that resist deformation. Snakes may be understood as a special case of the general technique of matching a deformable model to an image by means of energy minimization. In this module, extract the features related to color or shape features. Based on these features, we can extract the fire regions from images.

FIRE RECOGNITION

The classification is the final step of the system. After analyzing the structure, each section individually evaluated for the probability of true positives. Brain diseases are classified using Convolutional neural network algorithm. CNNs represent feed-forward neural networks which encompass diverse combos of the convolutional layers, max pooling layers, and completely related layers and Take advantage of spatially neighborhood correlation by way of way of imposing a nearby connectivity pattern among neurons of adjacent layers. Convolutional layers alternate with max pooling layers mimicking the individual of complex and clean cells in mammalian seen cortex .A CNN includes one or extra pairs of convolution and max pooling layers and ultimately ends with completely related neural networks. The hierarchical structure of CNNs is steadily proved to be the most efficient and successful manner to analyze visible representations. We know that CNNs can accomplish competitive and even better performance than human being in some visual problems, and its capability inspires us to study the possibility of applying CNNs for classify the disease features. The CNN varies in how the convolutional and max pooling layers are realized and how the nets are trained. Finally classify the image regions using deep learning algorithm. And then improve the accuracy in classification

ALERT SYSTEM

The forests as a whole are heavily affected by human activity. The rapid growth of increasing the population and urbanization has led to the outbreak of forest regions. Forest fire is a natural hazard to the environment and the interference of the atmosphere system; the environment affects living organisms. Satellite imagery also provides a fire monitoring, management, and damage assessment tool for compliance with burn areas to understand a favorable fire range. Satellite image refers to the ability of images from dataset images taken in a remote area to receive specific information. In this module, send alert to the authority in terms of SMS at the time of fire detection. It can be useful to provide earlier detection.

4.2 NON FUNCTIONAL REQUIREMENTS

Usability

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

Scalability

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

Security

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.

Performance

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

Reliability

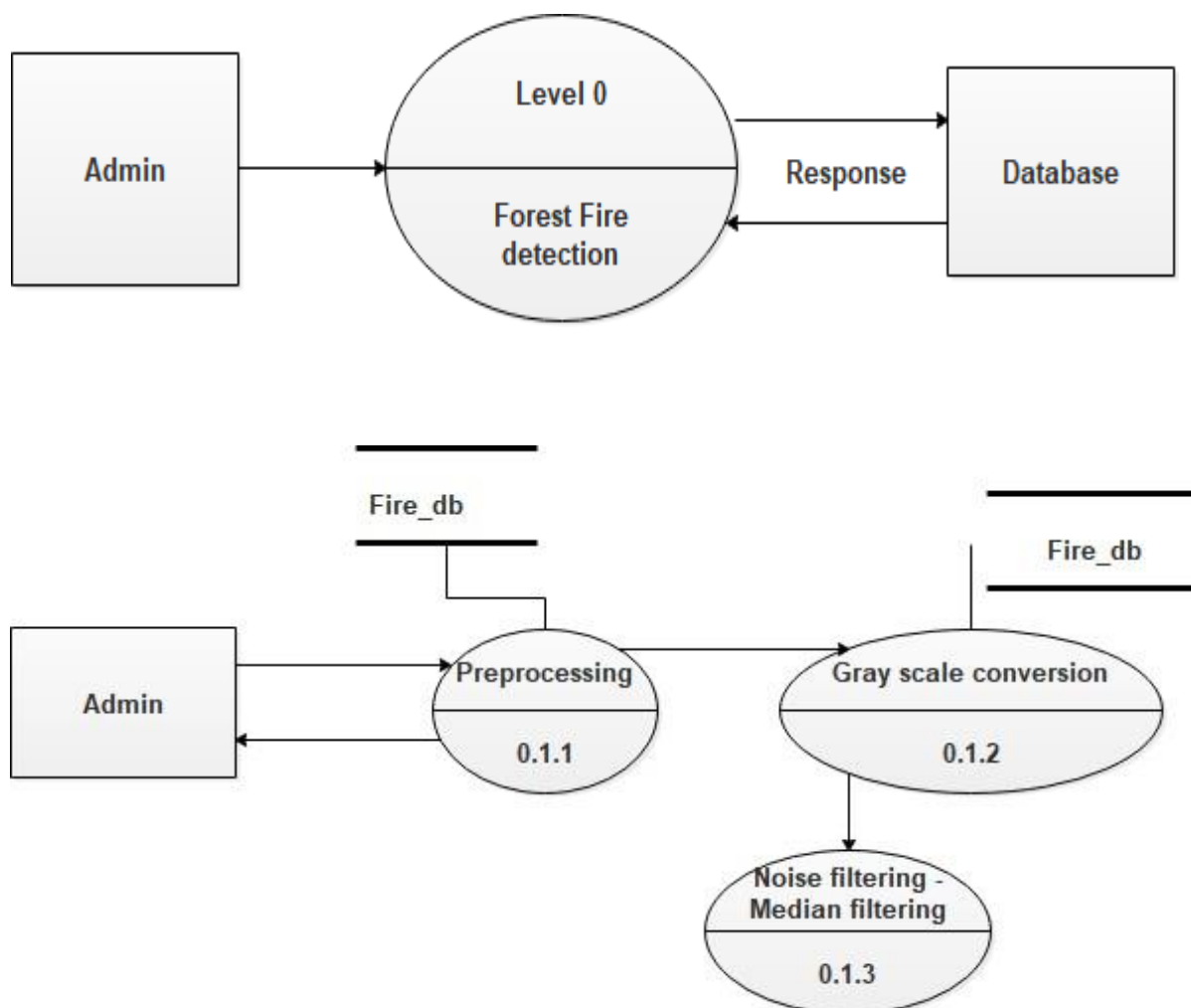
The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day

5. PROJECT DESIGN

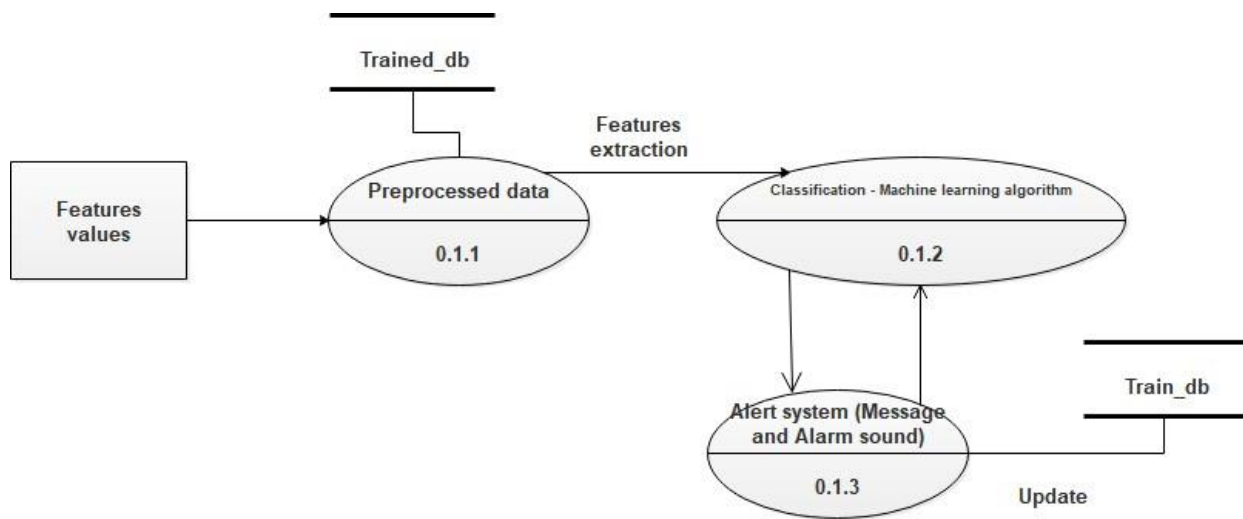
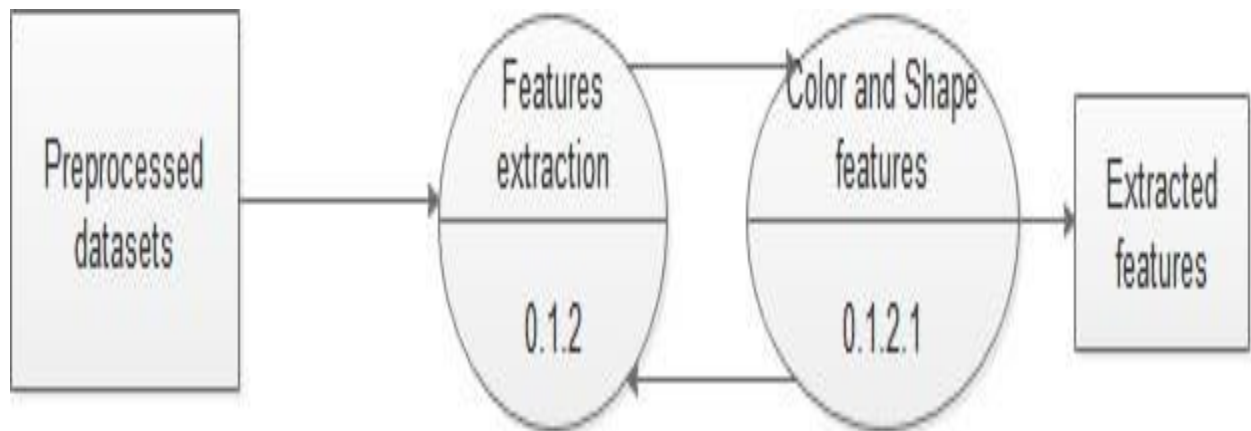
5.1 DATA FLOW DIAGRAMS

The classic visual representation of how information moves through a system is a data flow diagram (DFD). A tidy and understandable DFD can graphically represent the appropriate quantity of the system demand. It can be done manually, automatically, or both. It demonstrates how information enters and exits the system, what modifies the data, and where information is kept .A DFD's goal is to outline the boundaries and scope of a system as a whole. It can be utilised as a communication tool between a system analyst and any participant in the sequence that serves as the foundation for system redesign.

LEVEL 0



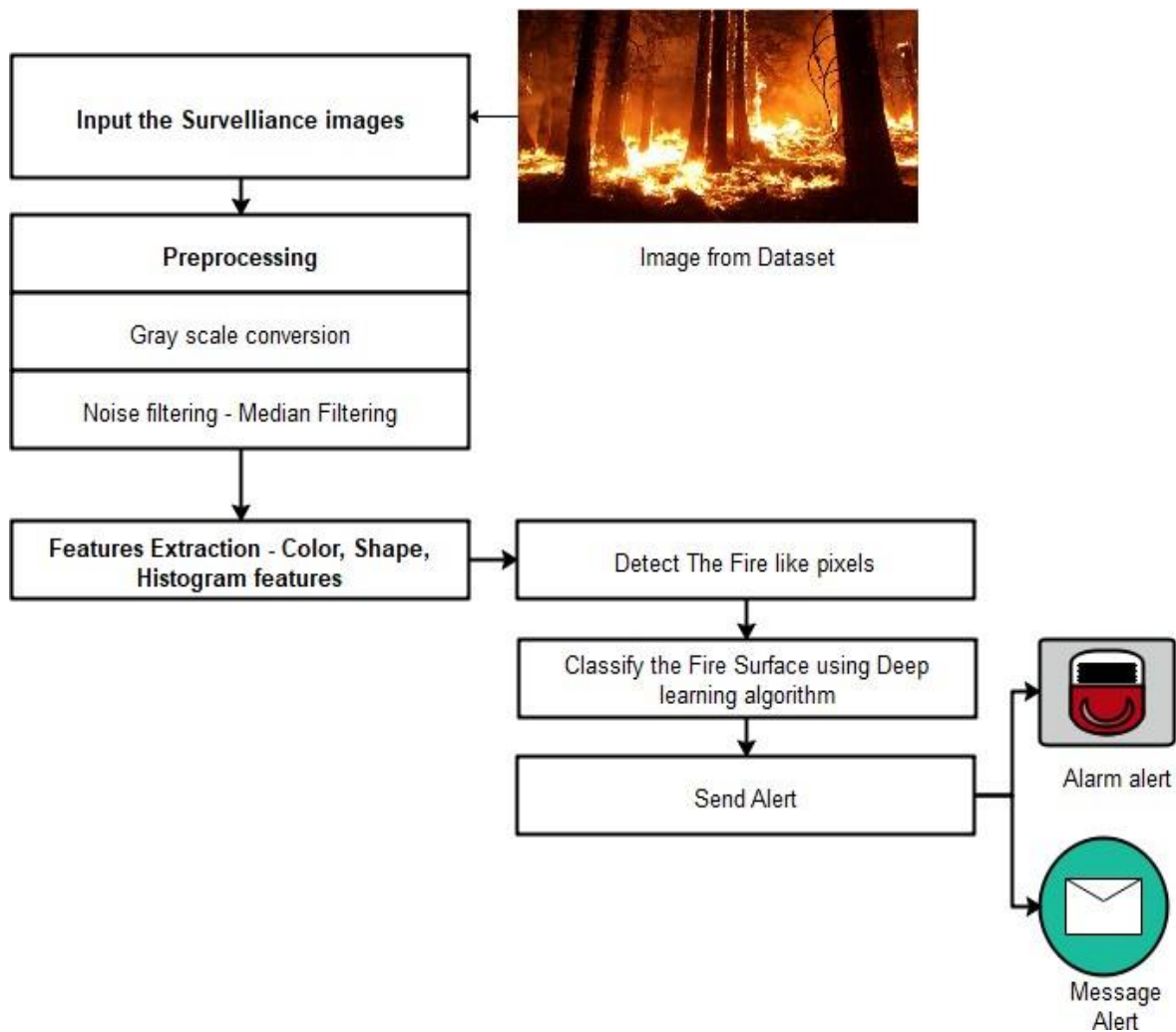
LEVEL 2



5.2 SOLUTION & TECHNICAL ARCHITECTURE

SOLUTION ARCHITECTURE

In this architecture, image can be uploaded from datasets. Then perform preprocessing steps to eliminate the noises in image and perform median filtering algorithm to remove the noises in image. Then extract the features from images. Finally classify the pixels whether it is fire or not. Based on features, send alert about fire.

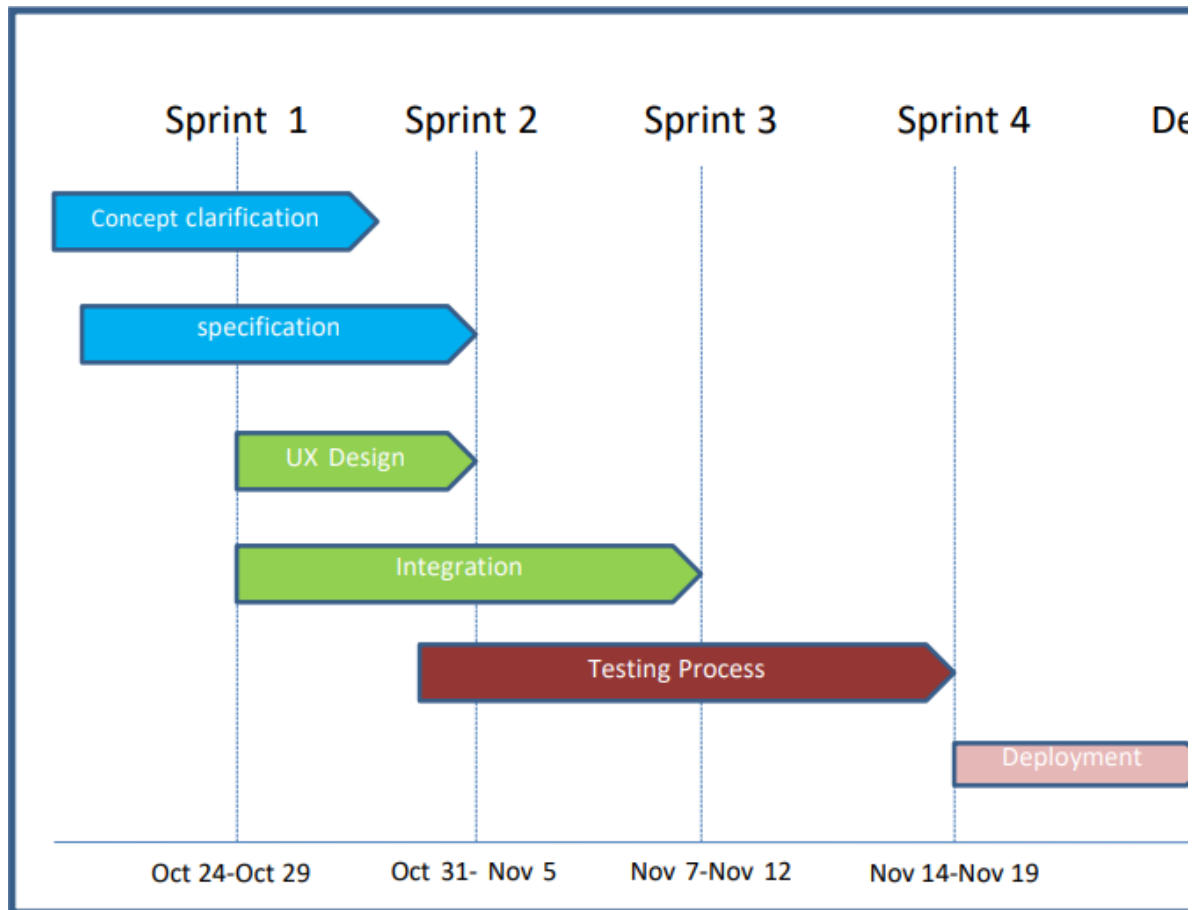


5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story Task	Acceptance criteria	Priority	Release
Environmental list	Collect the data	USN-1	As an Environmentalist, it is necessary to collect the data of the forest which includes temperature, humidity, wind and rain of the forest	It is necessary to collect the right data else the prediction may become wrong	High	Sprint-1
		USN-2	Identify algorithms that can be used for prediction	To collect the algorithm to identify the accuracy level of each algorithm	Medium	Sprint-2
	Implement Algorithm	USN-3	Identify the accuracy of each algorithm	Accuracy of each algorithm-calculated so that it is easy to obtain the most accurate output	High	Sprint-2
		USN-4	Evaluate the Dataset	Data is evaluated before processing	Medium	Sprint-1
	Evaluate Accuracy of Algorithm	USN-5	Identify accuracy, precision, recall of each algorithm	These values are important for obtaining the right output	High	Sprint-3

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION



6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

JIRA has categorized reports in four levels, which are –

- Agile
- Issue Analysis
- Forecast & Management
- Others

VELOCITY: SPRINT - 1

Sprint duration = 5 days

Velocity of team = 20 points

$$\text{Average Velocity (AV)} = \frac{\text{Velocity}}{\text{Sprint duration}}$$

$$AV = 20/5 = 4$$

Average Velocity = 4

VELOCITY: Sprint 1 - 4

Sprint duration = 20 days

Velocity of team = 80 points

$$\text{Average Velocity (AV)} = \frac{\text{Velocity}}{\text{Sprint duration}}$$

$$AV = 80/20 = 4$$

Total Average Velocity = 4

7. CODING & SOLUTION

7.1 Feature

```
from flask import render_template, redirect, url_for, request

import warnings

import datetime

import cv2


app = Flask(__name__)

app.config['DEBUG']

app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'


@app.route("/")

def homepage():


    return render_template('index.html')


@app.route("/Training")

def Training():

    return render_template('Tranning.html')


@app.route("/Test")

def Test():

    return render_template('Test.html')


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

def train():
```

```

if request.method == 'POST':

    import model as model

    return render_template('Tranning.html')

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

def testimage():

    if request.method == 'POST':

        file = request.files['fileupload']

        file.save('static/Out/Test.jpg')

        img = cv2.imread('static/Out/Test.jpg')

        if img is None:

            print('no data')

        img1 = cv2.imread('static/Out/Test.jpg')

        print(img.shape)

        img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))

        original = img.copy()

        neworiginal = img.copy()

        cv2.imshow('original', img1)

        gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

        img1S = cv2.resize(img1, (960, 540))

        cv2.imshow('Original image', img1S)

        grayS = cv2.resize(gray, (960, 540))

        cv2.imshow('Gray image', grayS)

```

```
gry = 'static/Out/gry.jpg'
```

```
cv2.imwrite(gry, grayS)
```

```
from PIL import ImageOps, Image
```

```
im = Image.open(file)
```

```
im_invert = ImageOps.invert(im)
```

```
inv = 'static/Out/inv.jpg'
```

```
im_invert.save(inv, quality=95)
```

```
dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)
```

```
cv2.imshow("Nose Removal", dst)
```

```
noi = 'static/Out/noi.jpg'
```

```
cv2.imwrite(noi, dst)
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
import tensorflow as tf
```

```
classifierLoad = tf.keras.models.load_model('firemodel.h5')
```

```
import numpy as np
```

```
from keras.preprocessing import image
```

```
test_image = image.load_img('static/Out/Test.jpg', target_size=(200, 200))
```



```
img1 = cv2.imread('static/Out/Test.jpg')

# test_image = image.img_to_array(test_image)

test_image = np.expand_dims(test_image, axis=0)

result = classifierLoad.predict(test_image)


out = ""

pre = ""

if result[0][0] == 1:


    out = "Fire"


elif result[0][1] == 1:


    out = "Nofire"


sendmsg("8838719456","Alert"+out)

org = 'static/Out/Test.jpg'

gry = 'static/Out/gry.jpg'

inv = 'static/Out/inv.jpg'

noi = 'static/Out/noi.jpg'


return render_template('Test.html',result=out,org=org,gry=gry,inv=inv,noi=noi)
```

7.2 Features

Part 1 - Building the CNN

Importing the Keras libraries and packages

```
from keras.models import Sequential
```

```
from keras.layers import Convolution2D
```

```
from keras.layers import MaxPooling2D
```

```
from keras.layers import Flatten
```

```
from keras.layers import Dense
```

```
from keras.models import model_from_json
```

```
import matplotlib.pyplot as plt
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

```
batch_size = 32
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

All images will be rescaled by 1./255

```
train_datagen = ImageDataGenerator(rescale=1/255)
```

Flow training images in batches of 128 using train_datagen generator

```
train_generator = train_datagen.flow_from_directory(
```

```
    'Data', # This is the source directory for training images
```

```
    target_size=(200, 200), # All images will be resized to 200 x 200
```

```
    batch_size=batch_size,
```

```
    # Specify the classes explicitly
```

```
    classes = ['fire','nofire'],
```

```

# Since we use categorical_crossentropy loss, we need categorical labels
class_mode='categorical')

import tensorflow as tf

model = tf.keras.models.Sequential([

    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color

    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),

    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(128, activation='relu'),

    # 5 output neurons for 5 classes with the softmax activation

```

```
tf.keras.layers.Dense(2, activation='softmax')
])

model.summary()

from tensorflow.keras.optimizers import RMSprop
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])

total_sample=train_generator.n

n_epochs = 10

history = model.fit_generator(
    train_generator,
    steps_per_epoch=int(total_sample/batch_size),
    epochs=n_epochs,
    verbose=1)

model.save('firemodel.h5')
```

```
acc = history.history['accuracy']
```

```
loss = history.history['loss']
```

```
epochs = range(1, len(acc) + 1)
```

```
# Train and validation accuracy
```

```
plt.plot(epochs, acc, 'b', label=' accuracy')
```

```
plt.title(' accuracy')
```

```
plt.legend()
```

```
plt.figure()
```

```
# Train and validation loss
```

```
plt.plot(epochs, loss, 'b', label=' loss')
```

```
plt.title(' loss')
```

```
plt.legend()
```

```
plt.show()
```

```
J:\PyNew\ForestFire\venv\Scripts\python.exe J:/PyNew/ForestFire/model.py
```

```
2022-11-13 10:48:48.262324: W tensorflow/stream_executor/platform/default/dso_loader.cc:64]
Could not load dynamic library 'cudart64_110.dll'; dLError: cudart64_110.dll not found
```

```
2022-11-13 10:48:48.263525: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dLError if you do not have a GPU set up on your machine.
```

```
Found 1520 images belonging to 2 classes.
```

```
2022-11-13 10:49:08.227643: W tensorflow/stream_executor/platform/default/dso_loader.cc:64]
Could not load dynamic library 'nvcuda.dll'; dLError: nvcuda.dll not found
```

2022-11-13 10:49:08.228335: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2022-11-13 10:49:08.248149: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN

2022-11-13 10:49:08.249072: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-9BF8NUN

2022-11-13 10:49:08.280592: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 198, 198, 16)	448
max_pooling2d (MaxPooling2D)	(None, 99, 99, 16)	0
)	
conv2d_1 (Conv2D)	(None, 97, 97, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 32)	0
	2D)	
conv2d_2 (Conv2D)	(None, 46, 46, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 64)	0
	2D)	

conv2d_3 (Conv2D) (None, 21, 21, 64) 36928

max_pooling2d_3 (MaxPooling (None, 10, 10, 64) 0
2D)

conv2d_4 (Conv2D) (None, 8, 8, 64) 36928

max_pooling2d_4 (MaxPooling (None, 4, 4, 64) 0
2D)

flatten (Flatten) (None, 1024) 0

dense (Dense) (None, 128) 131200

dense_1 (Dense) (None, 2) 258

=====

Total params: 228,898

Trainable params: 228,898

Non-trainable params: 0

Epoch 1/10

WARNING:tensorflow:AutoGraph could not transform <function
Model.make_train_function.<locals>.train_function at 0x00000233F7AE3F78> and will run it as-is.

Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output.

Cause: 'arguments' object has no attribute 'posonlyargs'

To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert

2022-11-13 10:49:12.287279: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2022-11-13 10:49:12.890860: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 38539264 exceeds 10% of free system memory.

2022-11-13 10:49:14.309721: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 38539264 exceeds 10% of free system memory.

2022-11-13 10:49:14.501970: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 40144896 exceeds 10% of free system memory.

2022-11-13 10:49:14.502170: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

47/47 [=====] - 37s 698ms/step - loss: 0.4586 - accuracy: 0.8226

Epoch 2/10

47/47 [=====] - 27s 563ms/step - loss: 0.2094 - accuracy: 0.9328

Epoch 3/10

47/47 [=====] - 27s 581ms/step - loss: 0.1390 - accuracy: 0.9503

Epoch 4/10

47/47 [=====] - 36s 776ms/step - loss: 0.1481 - accuracy: 0.9590

8.TESTING

8.1 TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

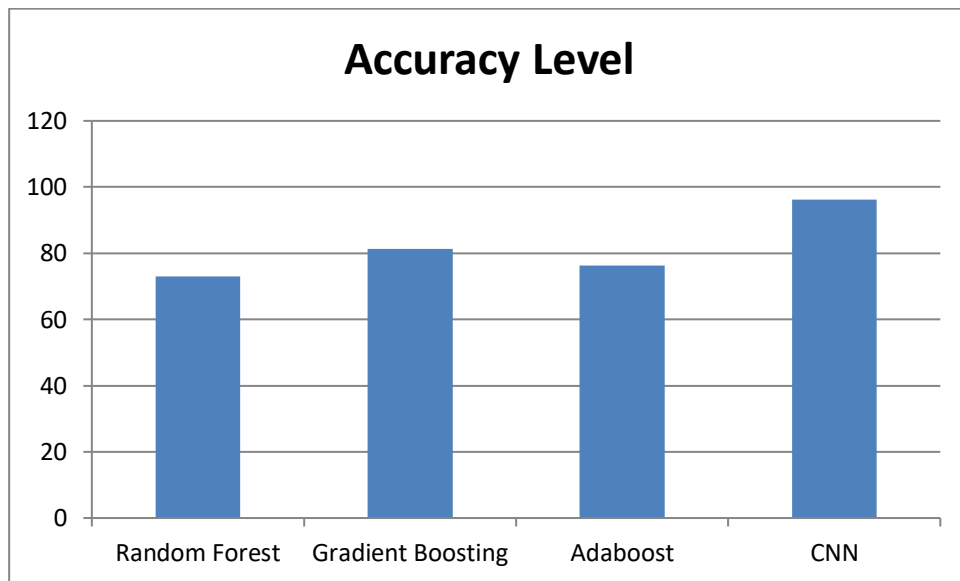
S. NO	Scenario	Input	Excepted output	Actual output
1	Admin Login Form	User name and password	Login	Login success.
2	Upload Image dataset	Forest fire Image	Upload successfully	Image dataset stored in database.
3	Preprocessing	Filtering the noise in uploaded image	Noise reduction	Filtering the Uploaded Image
4	Feature Extraction	Extracting color, shape and histogram features	Detect the fire	Extracting the dataset
5	Fire Recognition	Using deep learning algorithm	Classify the fire surface	Detecting the fire region in the uploaded dataset
6	Sending Alert	Using SMTP and ASAP Protocol	SMS and Alarm alert	Identify the fire by using the alarm

8.2 USER ACCEPTANCE TESTING

Acceptance testing can be defined in many ways, but a simple definition is the succeeds when the software functions in a manner that can be reasonable expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to fine whether the inputs are accepted by the database or other validations. For example accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

9 RESULTS

9.1 PERFORMANCE METRICS



10 ADVANTAGES & DISADVANTAGES

DISADVANTAGES

- Manual analysis of fire detection
- Need additional sensors to detect fire
- Time complexity is high
- Accuracy is less
- Irrelevant features are extracted

ADVANTAGES

- Automated analysis of fire detection
- Reduce the time and computational complexity
- Relevant features are extraction
- Improved accuracy rate

11 CONCLUSION

In this project, the process of the forest fire image recognition algorithm based on CNN is presented. Its main feature is that the flame image is employed for training and testing. Then, CNN model is introduced, and an adaptive pooling method combined with color features is proposed for the problem that the traditional pooling method in Back propagation neural network (BPNN) may weaken the image features in some cases. The effects of learning rate, batch size, and other parameters on the performance of CNN are analyzed based on experiments, and the optimal parameters are determined. Candidate flame area is extracted based on color feature; thus, the image feature of non-flame area in the hidden layer is reduced, and the feature, such as shape and texture, is enhanced. The information loss of image are avoided as adaptive pooling is adopted, and the rate of flame recognition in which fire area is segmentation than that of original image is adopted without segmentation. It is shown that the proposed algorithm has high recognition rate and isfeasible.

12 FUTURE SCOPE

In future, we can extend the framework to implement various deep learning algorithms to predict fires in multiple images with improved accuracy rate. This project can be implemented in various forest areas with additional research and innovation, helping to preserve the environment and save our forests. If temperature and humidity sensors were added to the apparatus, it would be possible to detect fires in forests without wasting valuable trees. In addition to serving as a haven for a wide variety of plants and animals, forests are an important source of oxygen for the ecosystem. This project can be expanded to find an effective method of determining the fire's location, gravity, and direction, spread, burned area, and numerous others.

13 APPENDIX

13.1 SOURCE CODE

```
from flask import render_template, redirect, url_for, request

import warnings

import datetime

import cv2


app = Flask(__name__)

app.config['DEBUG']

app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'


@app.route("/")

def homepage():


    return render_template('index.html')


@app.route("/Training")

def Training():

    return render_template('Tranning.html')


@app.route("/Test")

def Test():

    return render_template('Test.html')


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

def train():

    if request.method == 'POST':

        import model as model
```

```

        return render_template('Tranning.html')

@app.route("/testimage", methods=['GET', 'POST'])
def testimage():
    if request.method == 'POST':

        file = request.files['fileupload']

        file.save('static/Out/Test.jpg')

        img = cv2.imread('static/Out/Test.jpg')

        if img is None:

            print('no data')

        img1 = cv2.imread('static/Out/Test.jpg')

        print(img.shape)

        img = cv2.resize(img, ((int)(img.shape[1] / 5), (int)(img.shape[0] / 5)))

        original = img.copy()

        neworiginal = img.copy()

        cv2.imshow('original', img1)

        gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

        img1S = cv2.resize(img1, (960, 540))

        cv2.imshow('Original image', img1S)

        grayS = cv2.resize(gray, (960, 540))

        cv2.imshow('Gray image', grayS)

        gry = 'static/Out/gry.jpg'

```



```
cv2.imwrite(gry, grayS)

from PIL import ImageOps, Image

im = Image.open(file)

im_invert = ImageOps.invert(im)
inv = 'static/Out/inv.jpg'
im_invert.save(inv, quality=95)


dst = cv2.fastNlMeansDenoisingColored(img1, None, 10, 10, 7, 21)
cv2.imshow("Nosie Removal", dst)
noi = 'static/Out/noi.jpg'

cv2.imwrite(noi, dst)


import warnings

warnings.filterwarnings('ignore')


import tensorflow as tf

classifierLoad = tf.keras.models.load_model('firemodel.h5')


import numpy as np

from keras.preprocessing import image

test_image = image.load_img('static/Out/Test.jpg', target_size=(200, 200))

img1 = cv2.imread('static/Out/Test.jpg')

# test_image = image.img_to_array(test_image)
```

```
test_image = np.expand_dims(test_image, axis=0)
```

```
result = classifierLoad.predict(test_image)
```

```
out = "
```

```
pre = "
```

```
if result[0][0] == 1:
```

```
    out = "Fire"
```

```
elif result[0][1] == 1:
```

```
    out = "Nofire"
```

```
sendmsg("8838719456","Alert"+out)
```

```
org = 'static/Out/Test.jpg'
```

```
gry = 'static/Out/gry.jpg'
```

```
inv = 'static/Out/inv.jpg'
```

```
noi = 'static/Out/noi.jpg'
```

```
return render_template('Test.html',result=out,org=org,gry=gry,inv=inv,noi=noi)
```

```
def sendmsg(targetno,message):
```

```
import requests
```

```
requests.post("http://smsserver9.creativepoint.in/api.php?username=fantasy&password=596692&to=" + targetno + "&from=FSSMSS&message=Dear user your msg is " + message + " Sent By FMSG FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True, use_reloader=True)
```

```
# Part 1 - Building the CNN
```

```
# Importing the Keras libraries and packages
```

```
from keras.models import Sequential
```

```
from keras.layers import Convolution2D
```

```
from keras.layers import MaxPooling2D
```

```
from keras.layers import Flatten
```

```
from keras.layers import Dense
```

```
from keras.models import model_from_json
```

```
import matplotlib.pyplot as plt
```

```

import warnings

warnings.filterwarnings('ignore')

batch_size = 32


from tensorflow.keras.preprocessing.image import ImageDataGenerator


# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1/255)


# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
    'Data', # This is the source directory for training images
    target_size=(200, 200), # All images will be resized to 200 x 200
    batch_size=batch_size,
    # Specify the classes explicitly
    classes = ['fire', 'nofire'],
    # Since we use categorical_crossentropy loss, we need categorical labels
    class_mode='categorical')


import tensorflow as tf


model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution

```

```

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# The third convolution
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# The fourth convolution
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# The fifth convolution
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
tf.keras.layers.MaxPooling2D(2,2),
# Flatten the results to feed into a dense layer
tf.keras.layers.Flatten(),
# 128 neuron in the fully-connected layer
tf.keras.layers.Dense(128, activation='relu'),
# 5 output neurons for 5 classes with the softmax activation
tf.keras.layers.Dense(2, activation='softmax')
])

model.summary()

from tensorflow.keras.optimizers import RMSprop
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['accuracy'])

```

```
total_sample=train_generator.n
```

```
n_epochs = 10
```

```
history = model.fit_generator(  
    train_generator,  
    steps_per_epoch=int(total_sample/batch_size),  
    epochs=n_epochs,  
    verbose=1)
```

```
model.save('firemodel.h5')
```

```
acc = history.history['accuracy']
```

```
loss = history.history['loss']
```

```
epochs = range(1, len(acc) + 1)
```

```
# Train and validation accuracy
```

```
plt.plot(epochs, acc, 'b', label=' accuracy')
```

```
plt.title(' accuracy')
```

```
plt.legend()
```

```
plt.figure()
```

```
# Train and validation loss
```

```
plt.plot(epochs, loss, 'b', label=' loss')
```

```
plt.title(' loss')
```

```
plt.legend()
```

```
plt.show()
```

```
J:\PyNew\ForestFire\venv\Scripts\python.exe J:/PyNew/ForestFire/model.py
```

```
2022-11-13 10:48:48.262324: W tensorflow/stream_executor/platform/default/dso_loader.cc:64]
Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
```

```
2022-11-13 10:48:48.263525: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above
cudart dlerror if you do not have a GPU set up on your machine.
```

```
Found 1520 images belonging to 2 classes.
```

```
2022-11-13 10:49:08.227643: W tensorflow/stream_executor/platform/default/dso_loader.cc:64]
Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
```

```
2022-11-13 10:49:08.228335: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call
to cuInit: UNKNOWN ERROR (303)
```

```
2022-11-13 10:49:08.248149: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169]
retrieving CUDA diagnostic information for host: DESKTOP-9BF8NUN
```

```
2022-11-13 10:49:08.249072: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176]
hostname: DESKTOP-9BF8NUN
```

```
2022-11-13 10:49:08.280592: I tensorflow/core/platform/cpu_feature_guard.cc:151] This
TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the
following CPU instructions in performance-critical operations: AVX AVX2
```

```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
=====		

conv2d (Conv2D) (None, 198, 198, 16) 448

max_pooling2d (MaxPooling2D (None, 99, 99, 16) 0
)

conv2d_1 (Conv2D) (None, 97, 97, 32) 4640

max_pooling2d_1 (MaxPooling (None, 48, 48, 32) 0
2D)

conv2d_2 (Conv2D) (None, 46, 46, 64) 18496

max_pooling2d_2 (MaxPooling (None, 23, 23, 64) 0
2D)

conv2d_3 (Conv2D) (None, 21, 21, 64) 36928

max_pooling2d_3 (MaxPooling (None, 10, 10, 64) 0
2D)

conv2d_4 (Conv2D) (None, 8, 8, 64) 36928

max_pooling2d_4 (MaxPooling (None, 4, 4, 64) 0
2D)

flatten (Flatten) (None, 1024) 0

dense (Dense) (None, 128) 131200

dense_1 (Dense) (None, 2) 258

=====

Total params: 228,898

Trainable params: 228,898

Non-trainable params: 0

Epoch 1/10

WARNING:tensorflow:AutoGraph could not transform <function Model.make_train_function.<locals>.train_function at 0x00000233F7AE3F78> and will run it as-is.

Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output.

Cause: 'arguments' object has no attribute 'posonlyargs'

To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert

2022-11-13 10:49:12.287279: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

2022-11-13 10:49:12.890860: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 38539264 exceeds 10% of free system memory.

2022-11-13 10:49:14.309721: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 38539264 exceeds 10% of free system memory.

2022-11-13 10:49:14.501970: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 40144896 exceeds 10% of free system memory.

2022-11-13 10:49:14.502170: W tensorflow/core/framework/cpu_allocator_impl.cc:82] Allocation of 80289792 exceeds 10% of free system memory.

47/47 [=====] - 37s 698ms/step - loss: 0.4586 - accuracy: 0.8226

Epoch 2/10

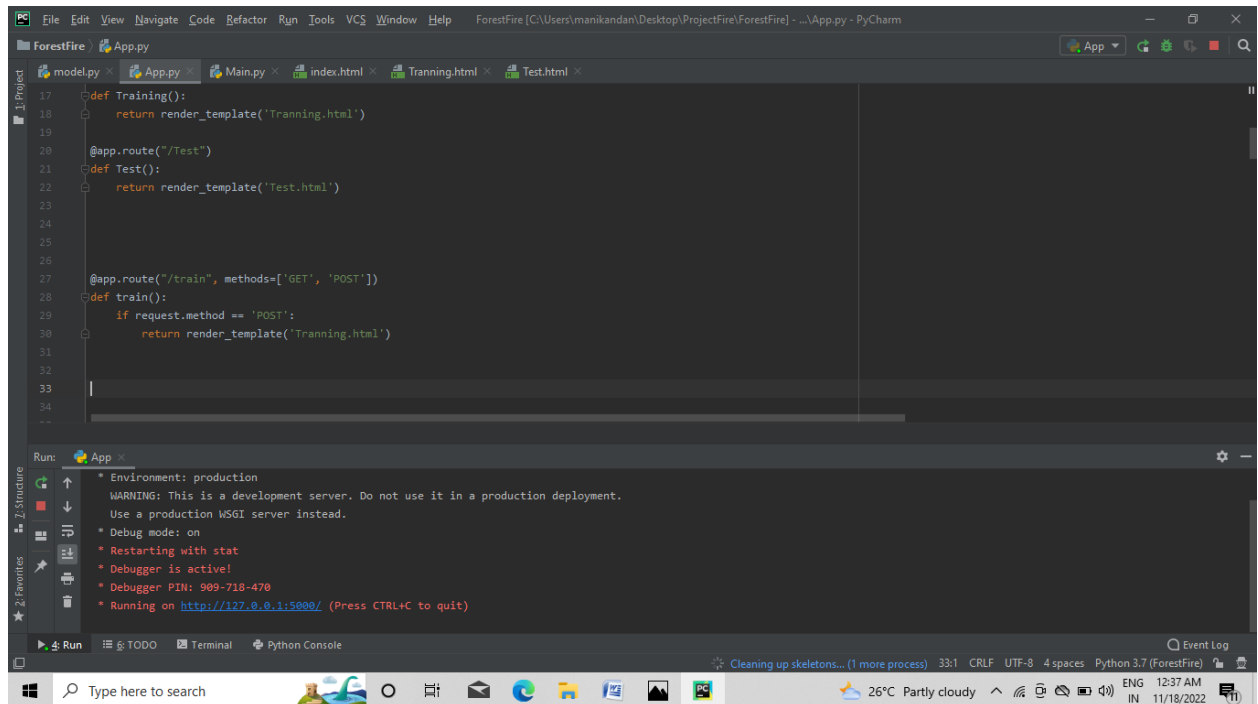
47/47 [=====] - 27s 563ms/step - loss: 0.2094 - accuracy: 0.9328

Epoch 3/10

47/47 [=====] - 27s 581ms/step - loss: 0.1390 - accuracy: 0.9503

Epoch 4/10

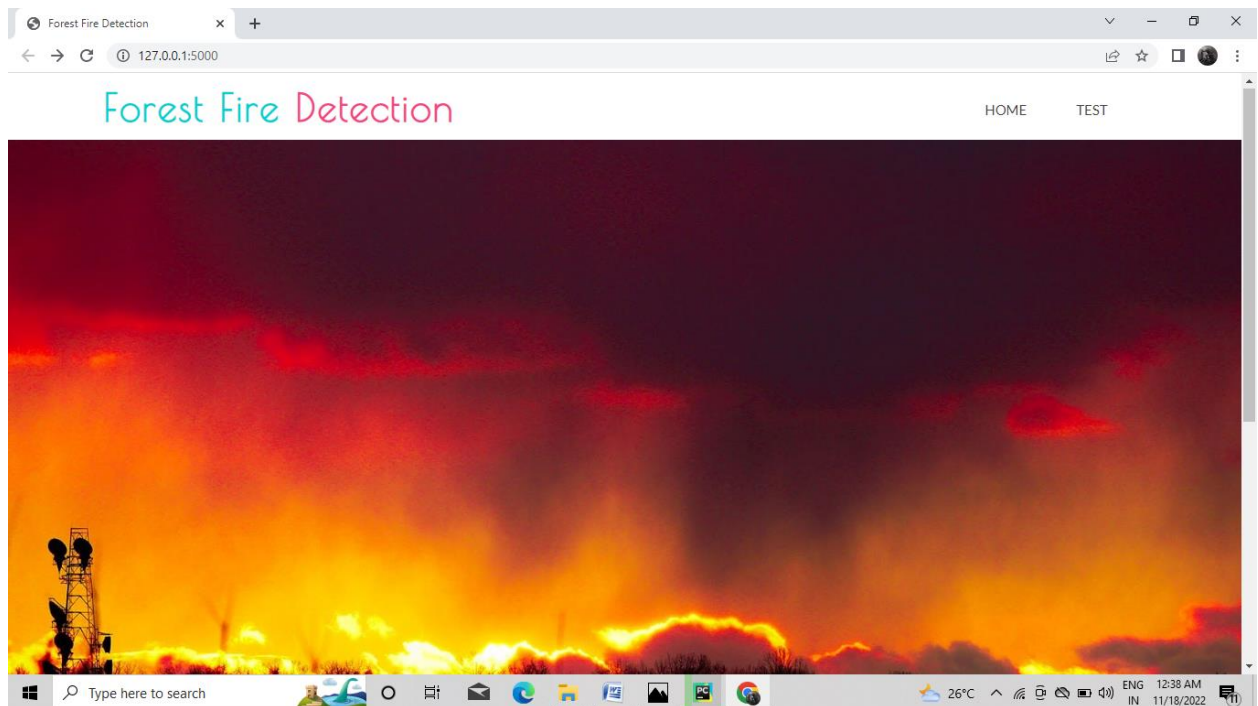
47/47 [=====] - 36s 776ms/step - loss: 0.1481 - accuracy: 0.9590

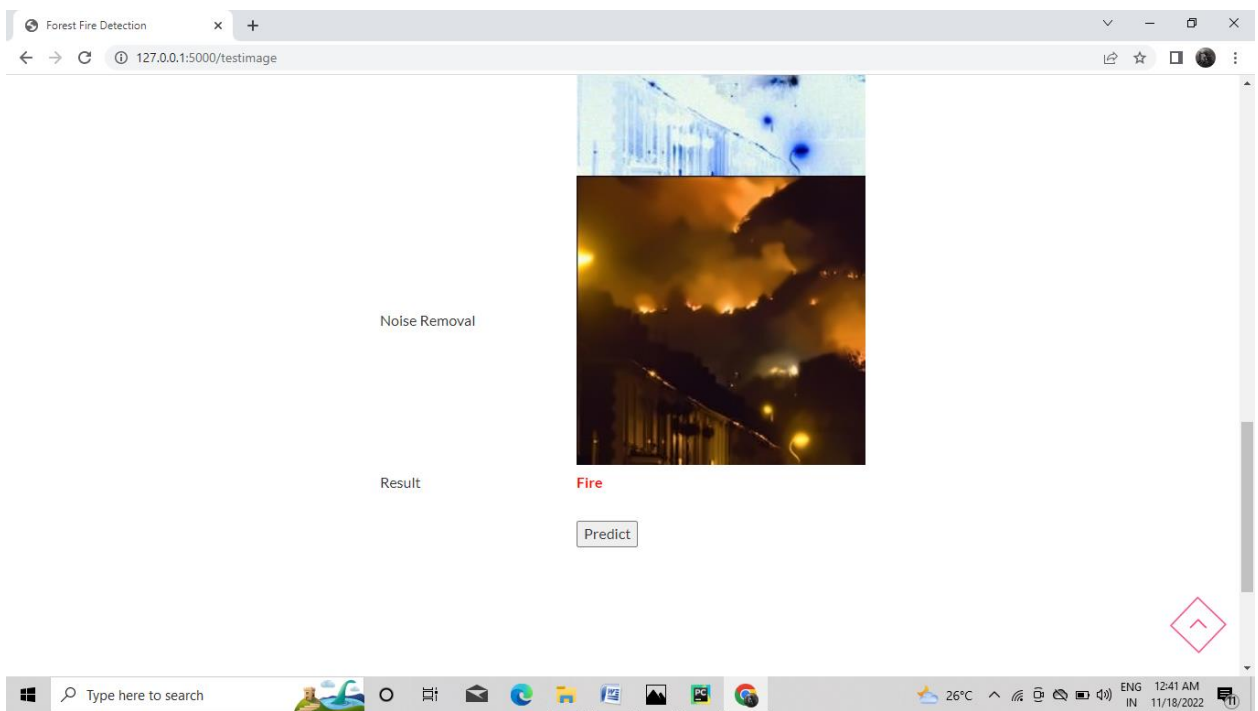
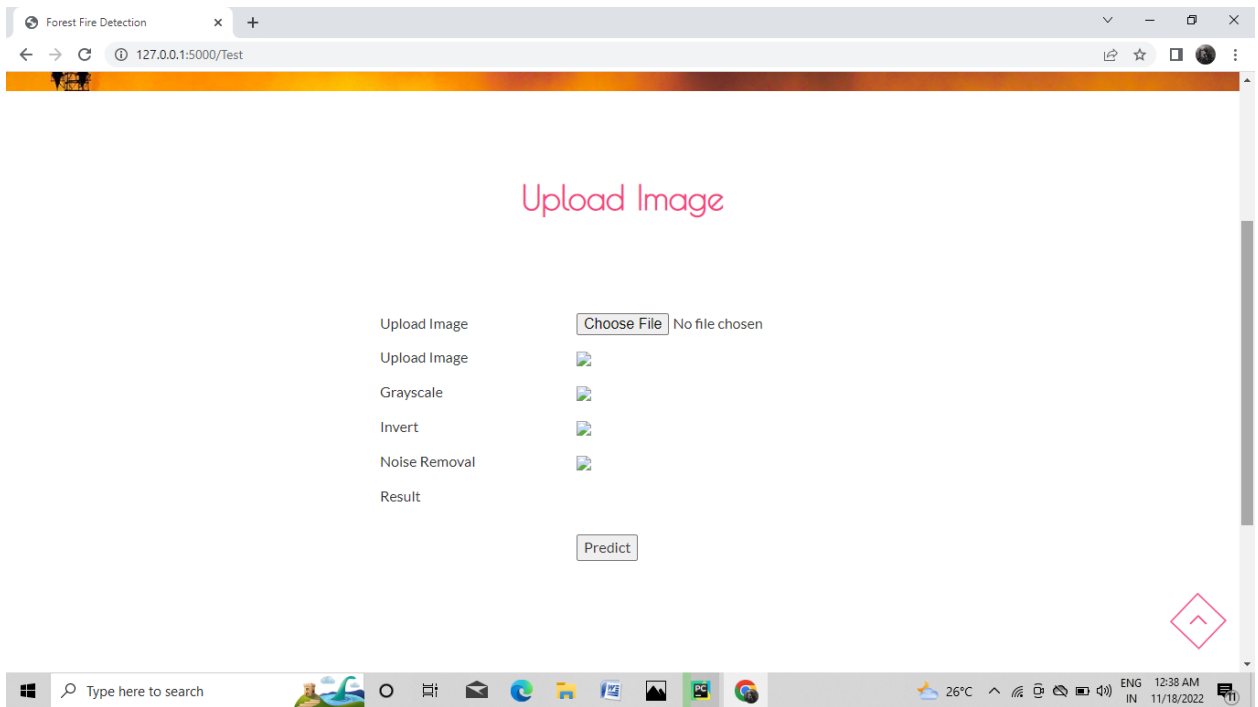


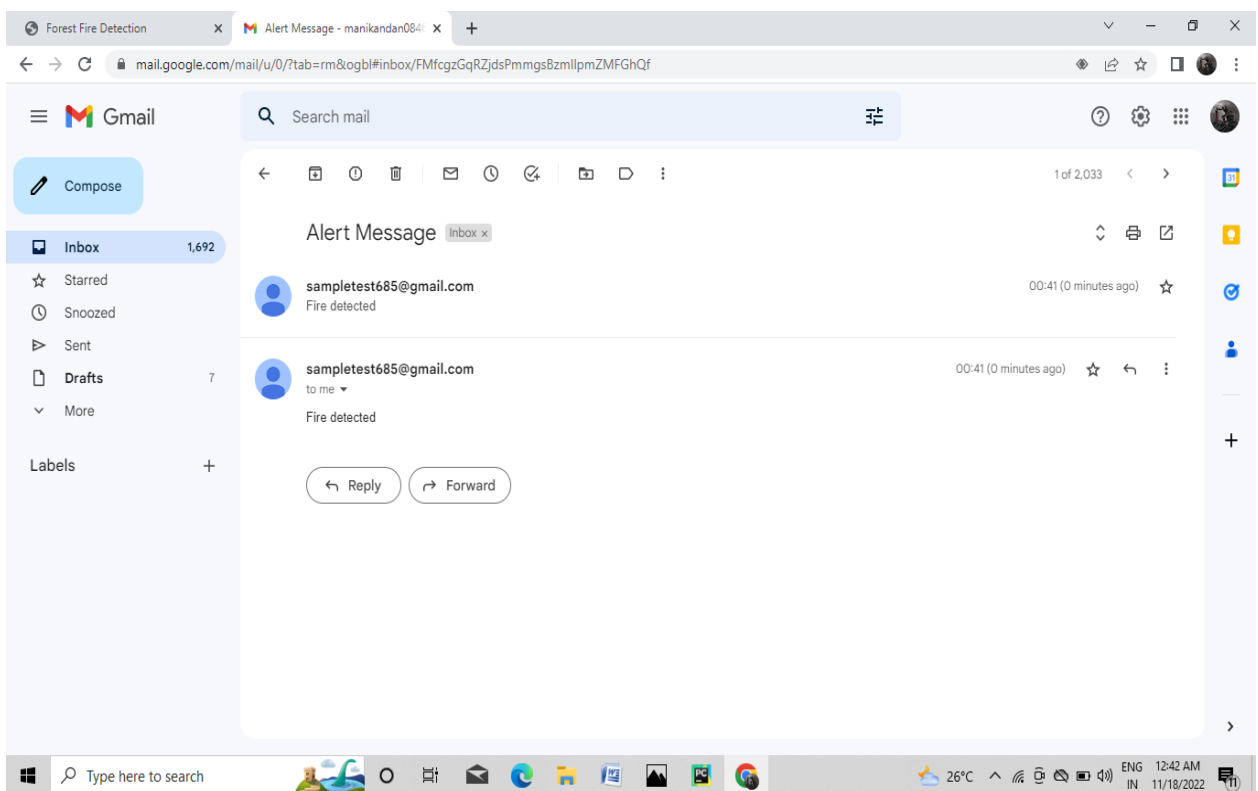
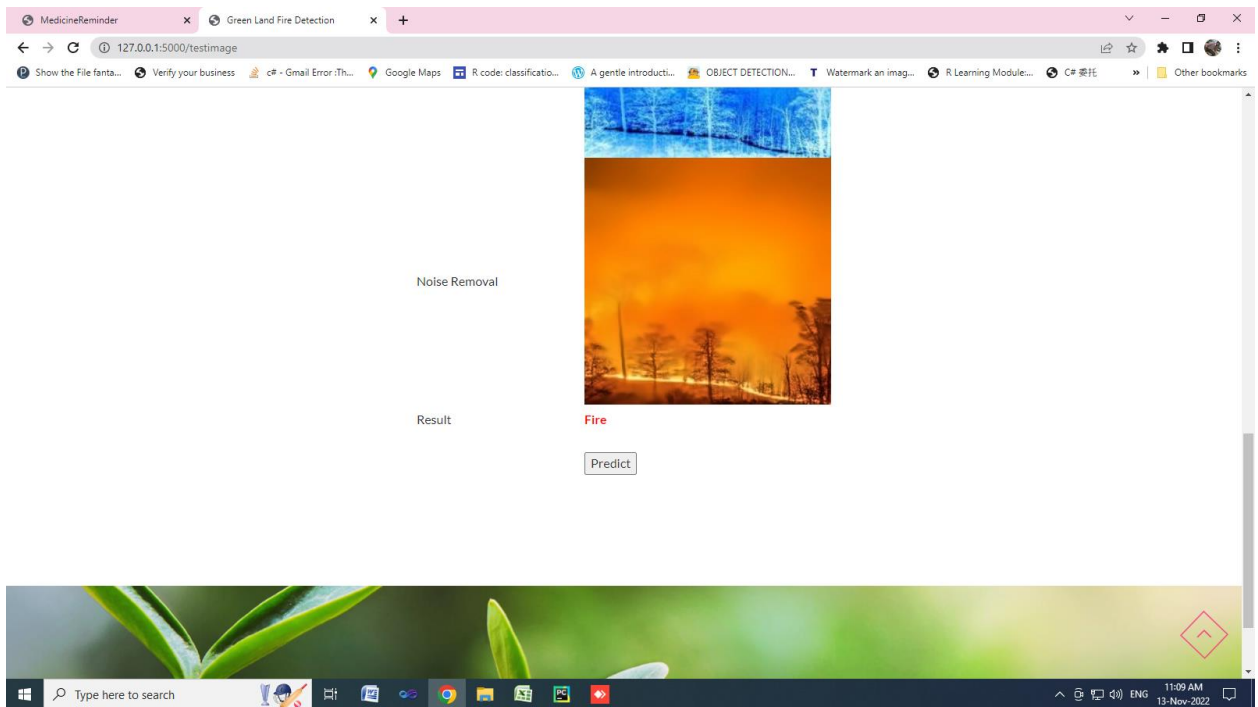
The image shows the PyCharm IDE interface. The main editor displays the code for `ForestFire.py`. The code includes a `Training()` function that returns `render_template('Tranning.html')`, a `Test()` function that returns `render_template('Test.html')`, and a `train()` function that checks the request method and returns `render_template('Tranning.html')`. The Run console at the bottom shows the output of the application, indicating it is running on `http://127.0.0.1:5000/` and providing a warning about the development server.

```
def Training():  
    return render_template('Tranning.html')  
  
@app.route("/Test")  
def Test():  
    return render_template('Test.html')  
  
@app.route("/train", methods=['GET', 'POST'])  
def train():  
    if request.method == 'POST':  
        return render_template('Tranning.html')
```

Run: App
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 909-718-470
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)







GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-12478-1659451972>

& PROJECT DEMO LINK :

<https://drive.google.com/file/d/15wtXHMLouH22A8K-6fNcD0r1pRkhoHGm/view?usp=drivesdk>