<u>EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES</u>

A PROJECT REPORT

Submitted by

Naveenrajan M (621519106050)

Vaitheswaran VC (621519106075)

Jeeva s (621519106030)

Dinesh M (621519106020)

BACHELOR OF ENGINEERING IN

ELECTRONICS AND COMMUNICATION

ENGINEERING

S.NO	TABLE OF CONTENT

1 INTRODUCTION

1.1. PROJECT OVERVIEW

Forests are the protectors of earth's ecological balance. Unfortunately, the forest fire is usually only observed when it has already spread over a large area, making its control and stoppage arduous and even impossible at times. The result is devastating loss and irreparable damage to the environment and atmosphere (30% of carbon dioxide (CO_2) in the atmosphere comes from forest fires), in addition to irreparable damage to the ecology (huge amounts of smoke and carbon dioxide (CO_2) in the atmosphere). Fast and effective detection is a key factor in forest fire fighting. To avoid uncontrollable wide spreading of forest fires it is necessary to detect fires in an early state and to prevent the propagation. Nowadays, image processing are critical components of the increasingly object detections . Such systems have a large applicability, and the environmental monitoring field can also benefit from their innovation.

1.2. PURPOSE:

. The purpose of the image processing concept is to capture the image from the real world and every day scenario appliances, etc., into intelligent interconnected virtual objects. By keeping the user informed on the state of things and giving the users control of things, a better global humans-devices-humans communication can be achieved.

2. LITERATURE SURVEY

2.1. EXISTING METHOD:

S.NO	AUTHOR	TITLE	YEAR
1.	G.V.Hristor Diyana kyuchukova Jordan Raychev	Emerging method for early detection of forest fires using unmanned Aerial vehicles and Lorawan sensor network	2018

S.NO	AUTHOR	TITLE	YEAR
2.	Panagiotis barmpoutis Konsmas dimitropoulas Nikos grammalidis.	A review on early forest fire detection systems using optical remote sensing	2020

S.NO	AUTHOR	TITLE	YEAR
3.	Hamdy soliman	S-mart forest fires early detection sensory system: Another approach of utilizing wireless sensor and neutral networks	2010

S.NO	AUTHOR	TITLE	YEAR
4.	Lilu cui Chaolong yao Zhengbo zou	The influence of climate change on forest fires in Yunnan province, Southwest china detected by GRACE satellites	2022

.

2.2. REFERENCES:

[1]Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[2]A. Grivei, A. Rdoi, C. Vduva and M. Datcu, "An Active-Learning approach to the query by example retrieval in remote sensing images," International Conference on Communications (COMM), pp. 377-380, 2016.

[3]G. Suciu, et al. “Remote Sensing for Forest Environment Preservation” WorldCIST, Recent Advances in Information Systems and Technologies, pp. 211-220, 2017.

[4]E. Olteanu, et al. "Forest Monitoring System Through Sound Recognition." In 2018 International Conference on Communications

(COMM), pp. 75-80. IEEE, 2018.

[4] Arasvathi, Nahalingham and Chelsea, Ferdyanti Kosasih “Study and Implementation of Internet of Things (IoT) Based Forest Fire Automation System to Detect and Prevent Wildfire”. INTI Journal, 1 (15) , pp. 1-5, 2018.

[5]J. Papán, M. Jurecka, J. Púchyová, “WSN for Forest Monitoring to Prevent Illegal Logging”, Proceedings of the Federated Cinference on Computer Science and Information Systems, pp. 809-812, 2012.

[6] Krivtsova et al. “Implementing a broadcast storm attack on a mission-critical wireless sensor network” In: International Conference on Wired/Wireless Internet Communication, 2016.

[7] Chen, Thou-Ho, Cheng-Liang Kao, and Sju-Mo Chang. "An intelligent real-time fire-detection method based on video processing." Security Technology, 2003. Proceedings. IEEE 37th Annual 2003 International Carnahan Conference on. IEEE, 2003.

[8] Chen, Thou-Ho, et al. "The smoke detection for early fire-alarming system base on video processing." Intelligent

Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06. International Conference on. IEEE, 2006

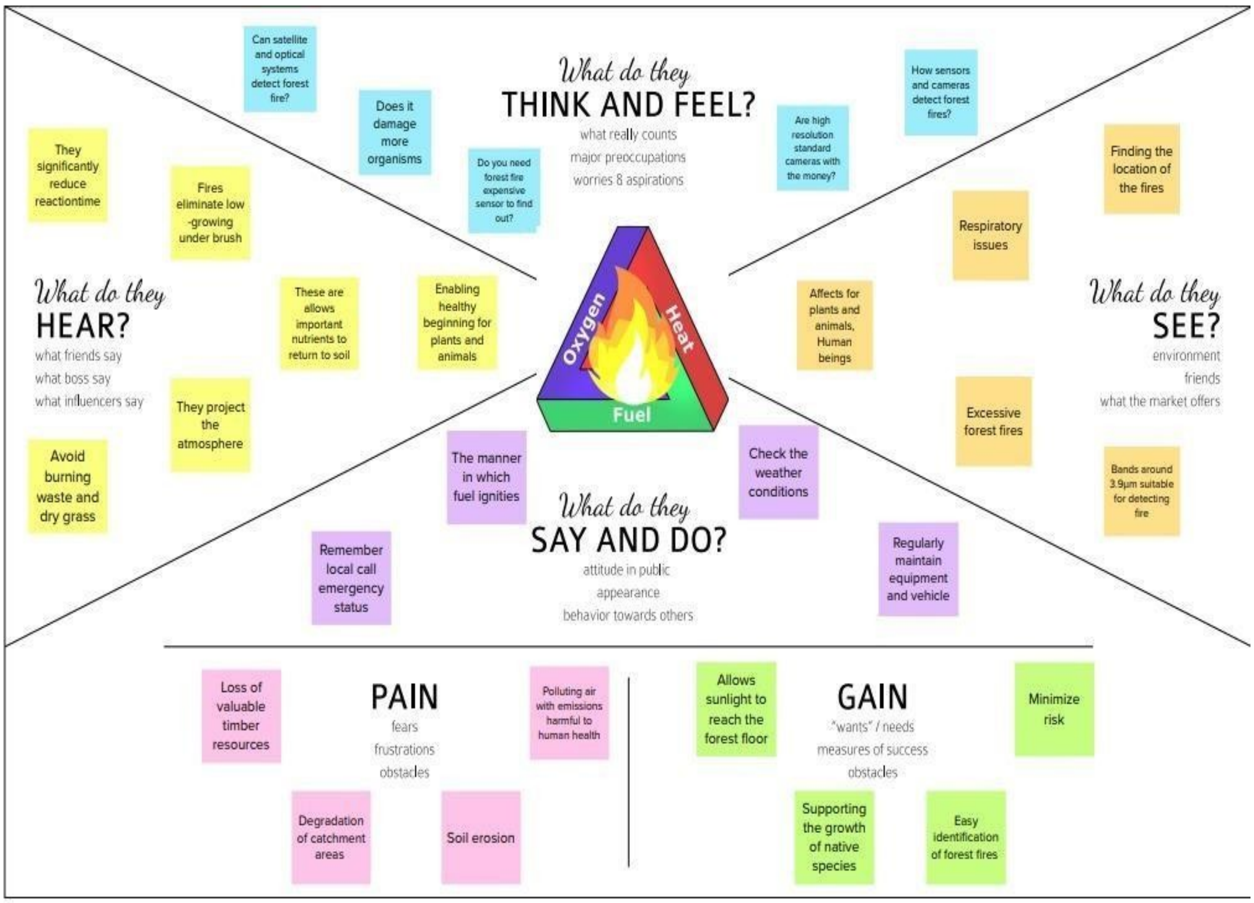
2.3. PROBLEM STATEMENT DEFINITION:

The most common hazard in forests is forests fire. They pose a treat not only to the forest wealth but also to the entire regime to fauna and flora seriously disturbing the bio – diversity and the ecology and environment of region.During summer , when there is no rain for months, the forests become littered with dry senescent leaves and twinges, which could burst into flames ignited by the slightest spark.Forest fire causes imbalances in nature and endangers biodiversity by reducing faunal and floral wealth.Traditional methods of fire prevention are not proving effective and it is now essential to raise public awareness on the matter , particularly among those people who live close to or in forested areas.

I am	Humans are responsible for 75% of all forest fire. Naturally occurring forest fires can be caused by lightning, volcanic activity and coal seam fires , though these are relatively rare.
I'm trying to	Using the recent technologies to avoid forest fires in Deep learning based on pre-trained satellite image processing and forest officer can view the recommendable forest fires through Gmail sms so avoid overexposure.
But,	I don't know much about the recent technology that helps me predict forest fires, and I haven't found the right solutions for forest fires.
Becaus	I don't want to cause devasting damage to both nature and humans, air pollution,every fire huge amounts of gases released in the atmosphere .
Which makes me feel	I'm not capable of early detect the fires and maintaining the area clean of forest but I trying solution for this problem.

3 **IDEATION & PROPOSED SOLUTION**

3.1. EMPATHY MAP CANVAS



3.2. BRAINSTORMING :

PROBLEM :

Brainstorm:

Brainstorm

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

10 minutes

NAVEENRAJAN M

- IR sensors can be used to detect to the forest fire
- A UAV can detect forest fire due to high mobility in vehicles
- Forest officer can view the recommendable forest fire through SMS
- Deep learning based Mathematical for detecting forest fires

VAITHESWARAN VC

- Detecting fire using movement of Birds
- Collecting data using satellite image
- Detecting by the fire light and smoke plumes emitted from the fires
- Monitoring the forest using satellite

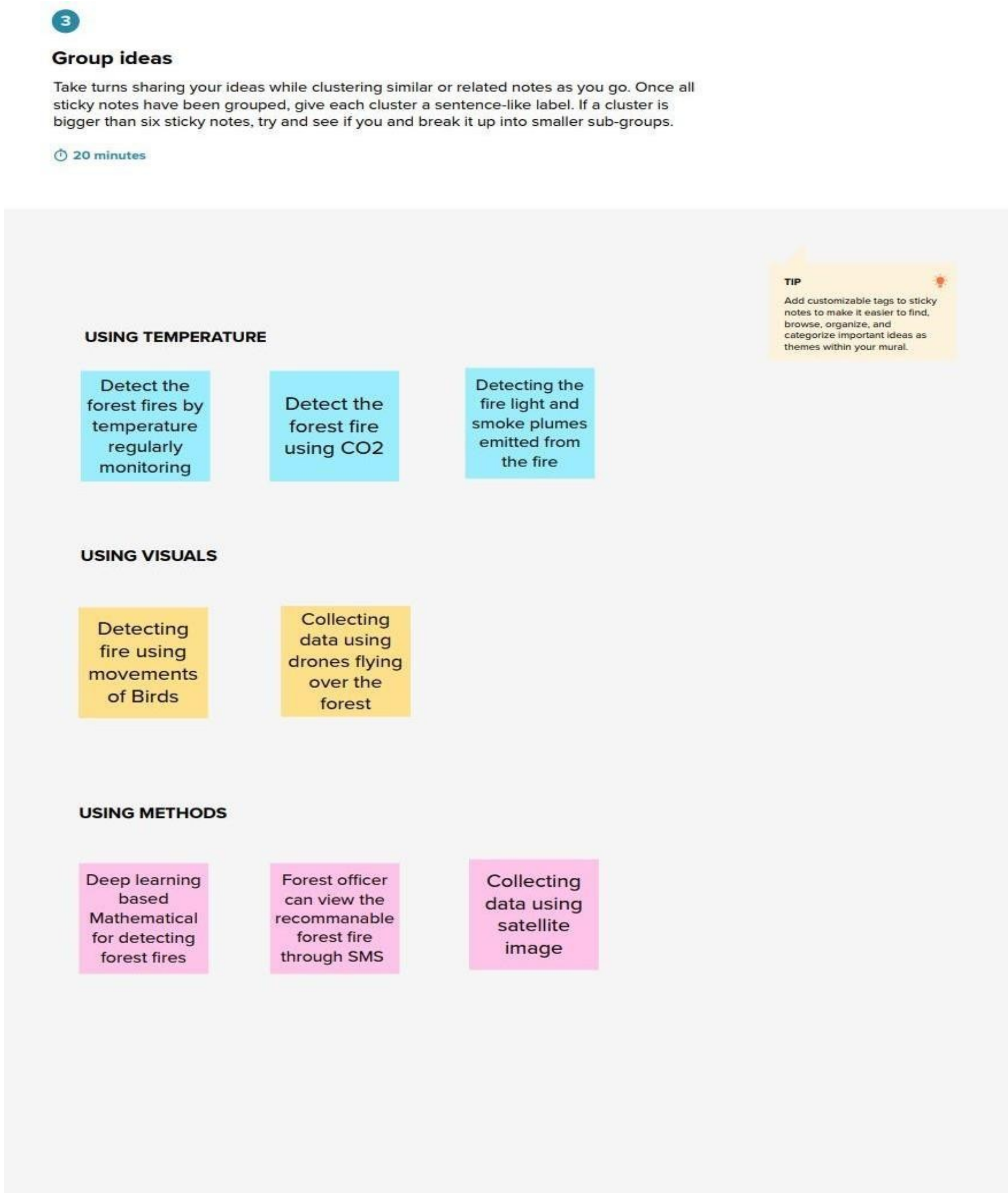
JEEVA S

- Optical sensor and Digital cameras can be used
- Fire detection using CNN modes
- Pre-trained model image processing
- Fire fighting robots can now use sensors such as flame sensor to detect fires

DINESH M

- Regularly remover dry leaves
- Detect the forest fire using CO_2
- Collecting data using drones flying over the forest
- Detect the forest fires by temperatures regularly monitoring

Group ideas:


3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
USING TEMPERATURE
Detect the forest fires by temperature regularly monitoring
Detect the forest fire using CO2
Detecting the fire light and smoke plumes emitted from the fire
USING VISUALS
Detecting fire using movements of Birds
Collecting data using drones flying over the forest
USING METHODS
Deep learning based Mathematical for detecting forest fires
Forest officer can view the recommanable forest fire through SMS
Collecting data using satellite image

Priortize:

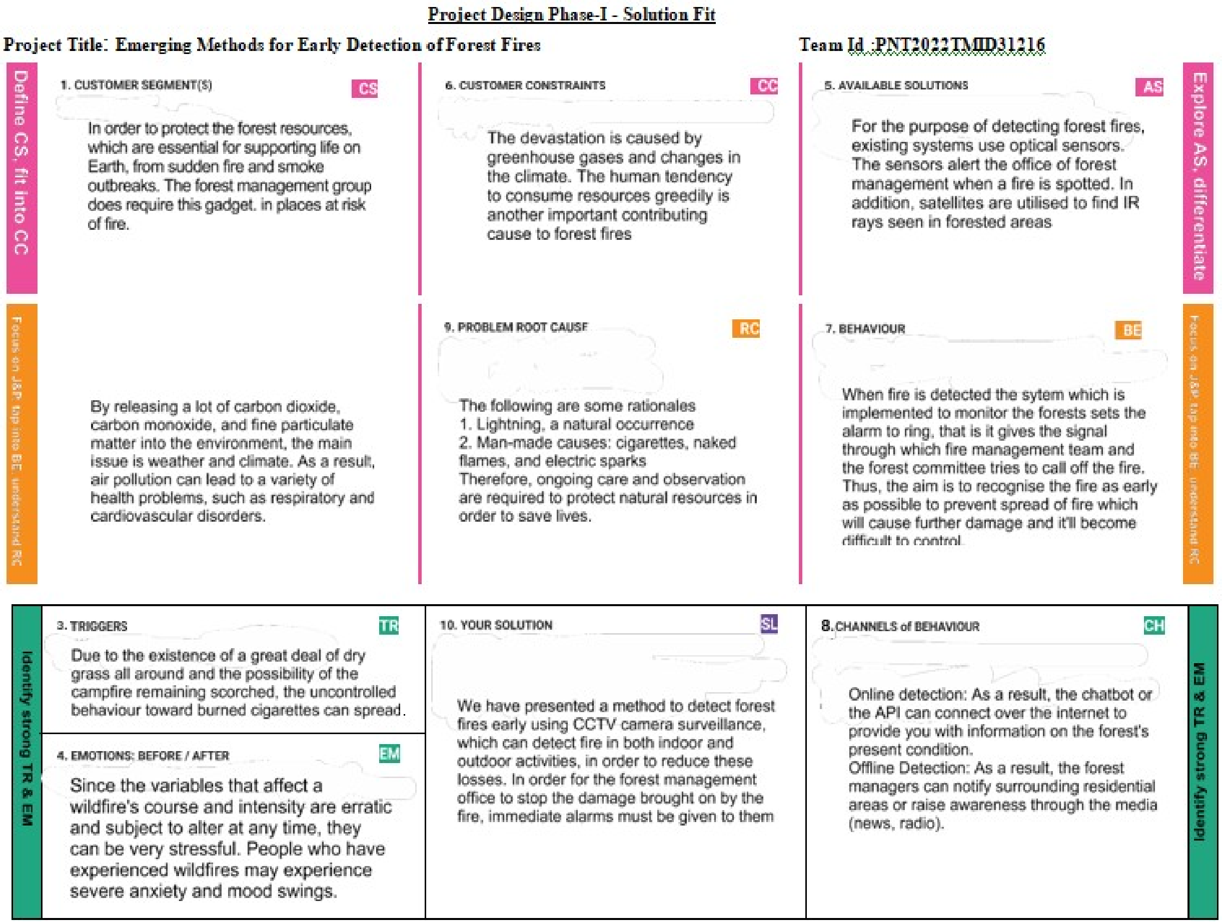

4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
Deep learning based Mathematical for detecting forest fire
Forest officer can view the recommanable forest fire through SMS
Collecting data using drones flying over the forest
Collecting data using satellite image
Detect the forest fires by temperature regularly monitoring
Detecting fire using Movement of birds
Detects the forest fires using CO2
Detecting the fire light and smoke plumes emitted from the fires
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
TIP
Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3. PROPOSED SOLUTION:

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Forests are one of the main factors in balancing the ecology.Forest fires are one of the most worrisome natural disasters,destroying thousands of acres of forests and nearby urban zones,affecting plant,animals and human life.So, the fire detection is important in this scenario. Finding of the exact location of the fire and sending notification to the fire authorities soon after the occurrence of fire can make a positive impact
2.	Idea / Solution description	Our solution aims at collecting the dataset to test and train the model . The damage and the cost for distinguish fire because of forest fire can be reduced when the fire detected early as possible. So, the fire detection is important in this scenario. Finding of the exact location of the fire and sending notification to the fire authorities soon after the occurrence of fire can make a positive impact. We have implemented a fire detection system to detect fire by capturing images. The system uses CNN(convolutional neural network), and image processing techniques.
3.	Novelty / Uniqueness	Real time computer program detect forest firein earliest before it spread to larger area. Our proposed system depends on using AI to make it cheaper and easier for the forest management. Accuracy and timely prediction using AI, CNN and API made it possible.
4.	Social Impact / Customer Satisfaction	The destroying homes, wildlife habitat and timber, and polluting the air with emissions harmful to human health. The proposed solution fulfills the satisfaction requirements of the customer as it provides instant alerts on fire detection which helps the forest officer to take action as soon as possible.

5.	Business Model (Revenue Model)	A working model in which mini cameras• continuously monitor the forest area and capture live images from satellites is a trained model that automatically detects fire or smoke. This proposed model can detect the exact• location of the fire and can be activated by SMS.The fire officer can implement quick responses and preventive measures.
6.	Scalability of the Solution	The device should be compatible with a minimum of 4GB RAM and WINDOWS 10 (x64 bit) and 100 GB ROM to support usage of various software like PYTHON 3.6.5. Testing and training undergo using latest technology like KERAS ,TENSORFLOW ,NUMPY and PILLOW

3.4. PROPOSED SOLUTION FIT:



Project Design Phase-I - Solution Fit

Project Title: Emerging Methods for Early Detection of Forest Fires **Team Id :PNT2022TMID31216**

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

In order to protect the forest resources, which are essential for supporting life on Earth, from sudden fire and smoke outbreaks. The forest management group does require this gadget. in places at risk of fire.

6. CUSTOMER CONSTRAINTS CC

The devastation is caused by greenhouse gases and changes in the climate. The human tendency to consume resources greedily is another important contributing cause to forest fires

5. AVAILABLE SOLUTIONS AS

For the purpose of detecting forest fires, existing systems use optical sensors. The sensors alert the office of forest management when a fire is spotted. In addition, satellites are utilised to find IR rays seen in forested areas

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

By releasing a lot of carbon dioxide, carbon monoxide, and fine particulate matter into the environment, the main issue is weather and climate. As a result, air pollution can lead to a variety of health problems, such as respiratory and cardiovascular disorders.

9. PROBLEM ROOT CAUSE RC

The following are some rationales
1. Lightning, a natural occurrence
2. Man-made causes: cigarettes, naked flames, and electric sparks
Therefore, ongoing care and observation are required to protect natural resources in order to save lives.

7. BEHAVIOUR BE

When fire is detected the sytem which is implemented to monitor the forests sets the alarm to ring, that is it gives the signal through which fire management team and the forest committee tries to call off the fire. Thus, the aim is to recognise the fire as early as possible to prevent spread of fire which will cause further damage and it'll become difficult to control.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

Due to the existence of a great deal of dry grass all around and the possibility of the campfire remaining scorched, the uncontrolled behaviour toward burned cigarettes can spread.

4. EMOTIONS: BEFORE / AFTER EM

Since the variables that affect a wildfire's course and intensity are erratic and subject to alter at any time, they can be very stressful. People who have experienced wildfires may experience severe anxiety and mood swings.

10. YOUR SOLUTION SL

We have presented a method to detect forest fires early using CCTV camera surveillance, which can detect fire in both indoor and outdoor activities, in order to reduce these losses. In order for the forest management office to stop the damage brought on by the fire, immediate alarms must be given to them

8. CHANNELS of BEHAVIOUR CH

Online detection: As a result, the chatbot or the API can connect over the internet to provide you with information on the forest's present condition.
Offline Detection: As a result, the forest managers can notify surrounding residential areas or raise awareness through the media (news, radio).

Identify strong TR & EM

4. *REQUIREMENT ANALYSIS*

4.1. FUNCTIONAL REQUIREMENTS:

FR No.	Functional Requirement(Epic)	Sub Requirement(Story/Sub-Task)
FR-1	Images surveillance start	Start surveillance from satellites is a trained model
FR-2	Image processing is being used to monitor the fire	Exact location monitoring through camera
FR-3	Detect the fire	Fire is detected through CNN model
FR-4	Aler	sending notification to the fire authorities

4.2. NON-FUNCTIONAL REQUIREMENTS

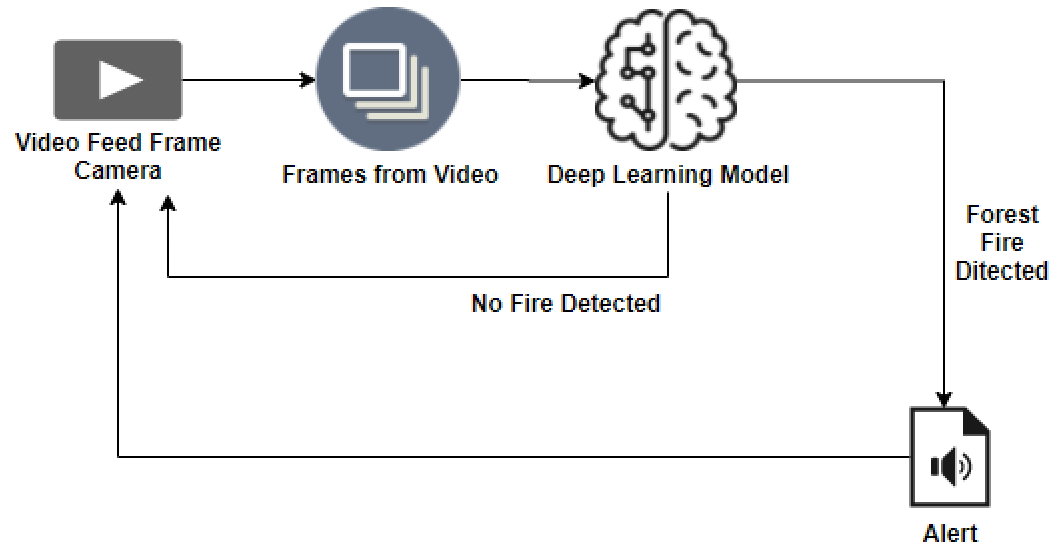
FR No.	Non-Functional Requirement	Description

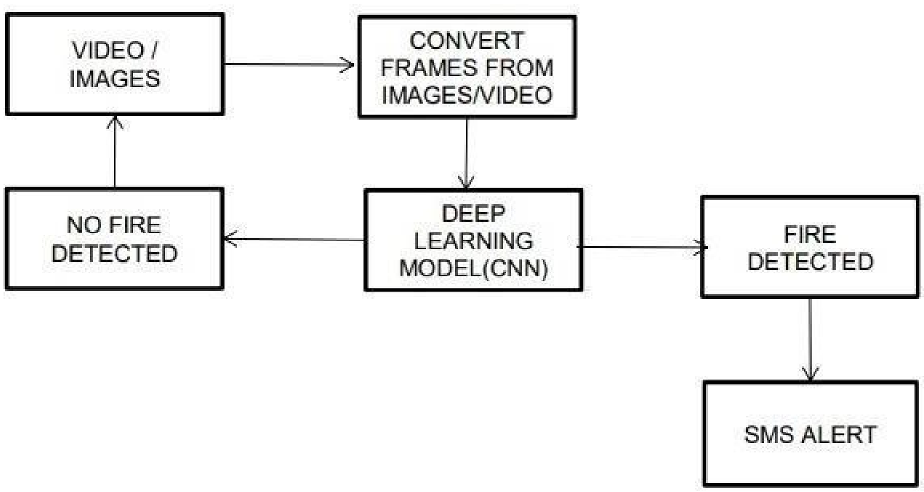
NFR-1	**Usability**	Usability is a unique and significant perspective to analyze user requirements, which can further improve the design quality, according to AI devices with machine learning.
NFR-2	**Security**	➢ HD and powerful CCTV cameras are used. ➢ The fire is found using image processing and 24-hour monitoring.
NFR-3	**Reliability**	A real-time and dependable fire detection method for an early warning system is required to ensure an effective response to an incident.
NFR-4	**Performance**	• The system is intended to monitor forest fires through image processing via a camera. • CCTV cameras are used to process images and detect forest fires. • The twilio module is used to send the forest officer an alert message
NFR-5	**Availability**	o By progressing to a more advanced system that uses real-time CCTV cameras to detect and alert on fires. o The convolutional neural network algorithm is extremely useful for detecting fire in captured images
NFR-6	**Scalability**	By detecting forest fires early, we can prevent loss of life as well as resource damage while decreasing air pollution, landslides, soil erosion, and Emission emissions into the environment

5. PROJECT DESIGN

5.1. DATA FLOW DIAGRAMS:

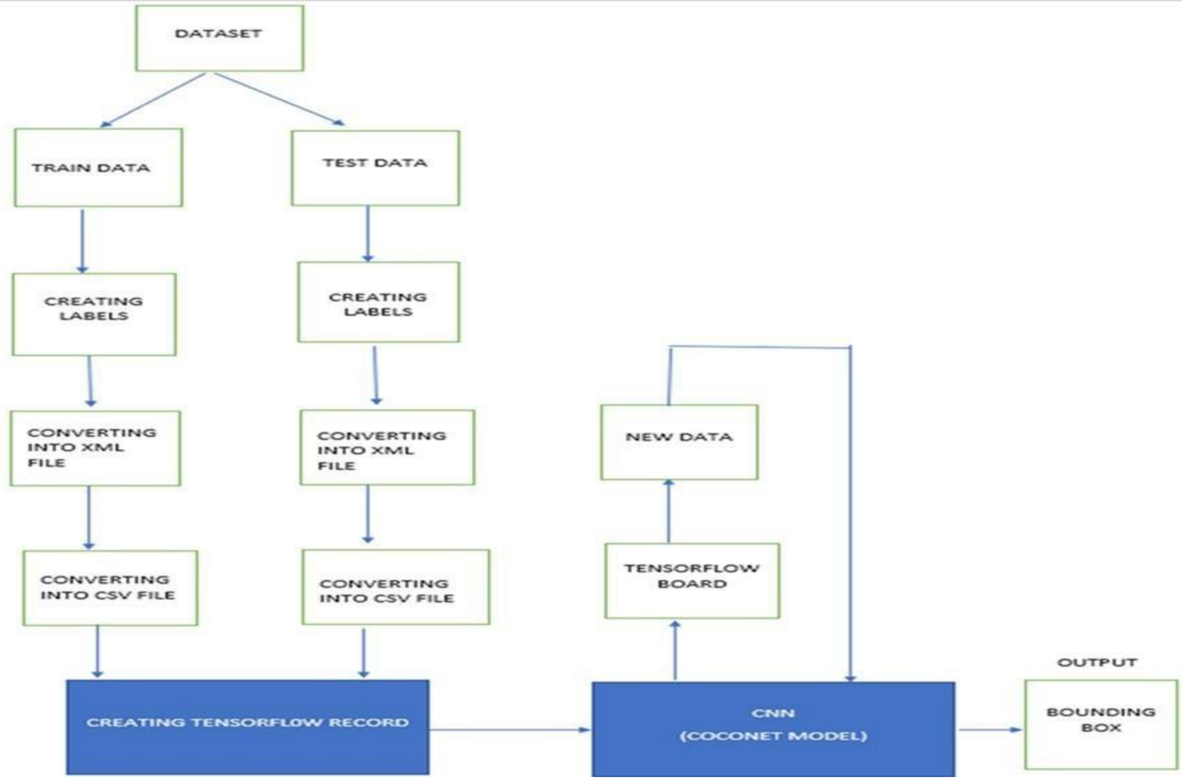
The traditional visual representation of how information moves through a system is a data flow diagram (DFD). A tidy and understandable DFD can graphically represent the appropriate amount of the system requirement. It demonstrates how information enters and exits the system, what modifies the data, and where information is kept.

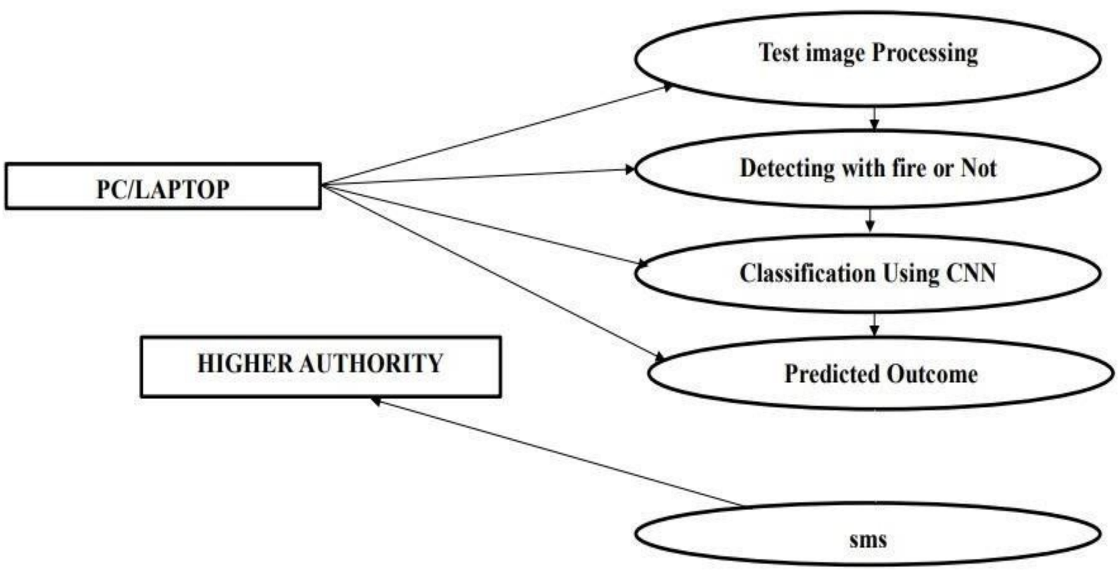




?5.2.SOLUTION AND TECHNICAL ARCHIETECTURE:

Solution Architecture:


DATASET
TRAIN DATA
TEST DATA
CREATING LABELS
CREATING LABELS
CONVERTING INTO XML FILE
CONVERTING INTO XML FILE
CONVERTING INTO CSV FILE
CONVERTING INTO CSV FILE
CREATING TENSORFLOW RECORD
NEW DATA
TENSORFLOW BOARD
CNN (COCONET MODEL)
OUTPUT
BOUNDING BOX


PC/LAPTOP
Test image Processing
Detecting with fire or Not
Classification Using CNN
Predicted Outcome
HIGHER AUTHORITY
sms

5.3.USER STORIES:

Use the below template to list all the user stories for the product.

User story	Functional Requirement(Epic)	User Story Number	User Story/Task	Acceptance criteria	Priority	Release
Environmentalist	Collect the data	USN-1	It is necessary for an animal rights activist to gather information about forest fires.	We must collect the correct data.because of prediction.	High	Sprint-1
		USN-2	Determine which algorithms can be used for prediction.	To gather the algorithms and determine each algorithm's accuracy .	Medium	Sprint-2
	Implement Algorithm	USN-3	Determine each algorithm's accuracy.	Accuracy of the algorithm is must to be calculated .	High	Sprint-2
		USN-4	assess the data set	Data is preprocessing before the training.	High	Sprint-1
	Evaluate Accuracy of Algorithm	USN-5	Decide the precision, accuracy, as well as recall of each algorithm.	Accuracy is important to detect the seviearity of fire	High	Sprint-3

6.PROJECT PLANNING & SCHEDULING:

6.1.SPRINT PLANNING & ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members

Sprint-1	Download data set	USN-1	The data is downloaded from the Kaggle website and then the data set is classified into training and testing images.	5	High	Naveenrajan Vaitheswaran Jeeva Dinesh
Sprint-2	Image preprocessing	USN-2	In Image processing technique the first step is usually importing the libraries that will be needed In the program. Import Keras library from that library and import the ImageDataGenerator Library to your Python script. The next step is defining the arguments for the ImageDataGenerator And next step is applying the ImageDataGenerator arguments to the train and test dataset.	10	Medium	Naveenrajan Vaitheswaran Jeeva Dinesh
Sprint-3	Training image	USN-3	In this training phase the ImageDataGenerator arguments is applied to the training images and the model is tested with several images and the model is saved.	5	High	Naveenrajan Vaitheswaran Jeeva Dinesh
Sprint-4	Testing Image, Evaluation metrics and accuracy	USN-4	In this testing phase the Image processing techniques is applied to the testing images and executed for prediction. In this phase the result, prediction, accuracy, and performance of the project are tested.	5	High	Naveenrajan Vaitheswaran Jeeva Dinesh

MILESTONE & ACTIVITY LIST:

ACTIVITY LIST

Activity Number	Activity	Sub Activity	Assigned To	Status
1.	PROJECT OBJECTIVES		All Members	Completed
2.	PROJECT FLOW		All Members	Completed
3.	PRE-REQUISITES		All Members	Completed
4.	DATA COLLECTION	4.1 Download the Dataset	All Members	Completed
5.	IMAGE PREPROCESSING	Import the ImageDataGenerator Library. Define the Parameters/Arguments for ImageDataGenerator class. Applying ImageDataGenerator Functionality to trainset and testset.	All Members	In Progress
6.	MODEL BUILDING	Importing the model building libraries. Initializing the model. Adding CNN layers. Adding dense layers. Configuring the	All Members	In Progress

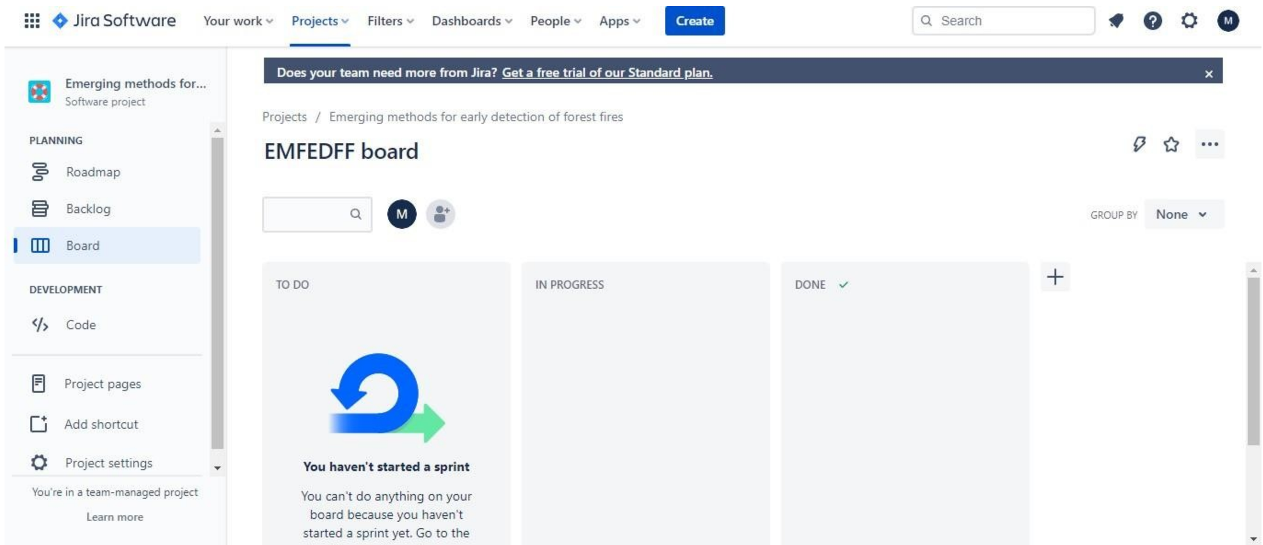
		learning process. Training the model. Saving the model. Predictions.		
7.	VIDEO ANALYSIS	OpenCV for video processing. Creating an account in Twilio service. Sending alert message.	All Members	In Progress
8.	TRAIN CNN MODEL ON IBM	Train image classification model. Register for IBM cloud.	All Members	In Progress
9.	IDEATION PHASE	Literature Review. Empathy map. Ideation.	All Members	Completed
10.	PROJECT DESIGN PHASE – I	Proposed Solution. Problem solution fit. Solution Architecture.	All Members	Completed
11.	PROJECT DESIGN PHASE -II	Customer journey. Functional requirement. Data flow Diagrams. Technology Architecture.	All Members	Completed
12.	PROJECT PLANNING PHASE	Prepare milestone and activity list. Sprint delivery plan.	All Members	Completed

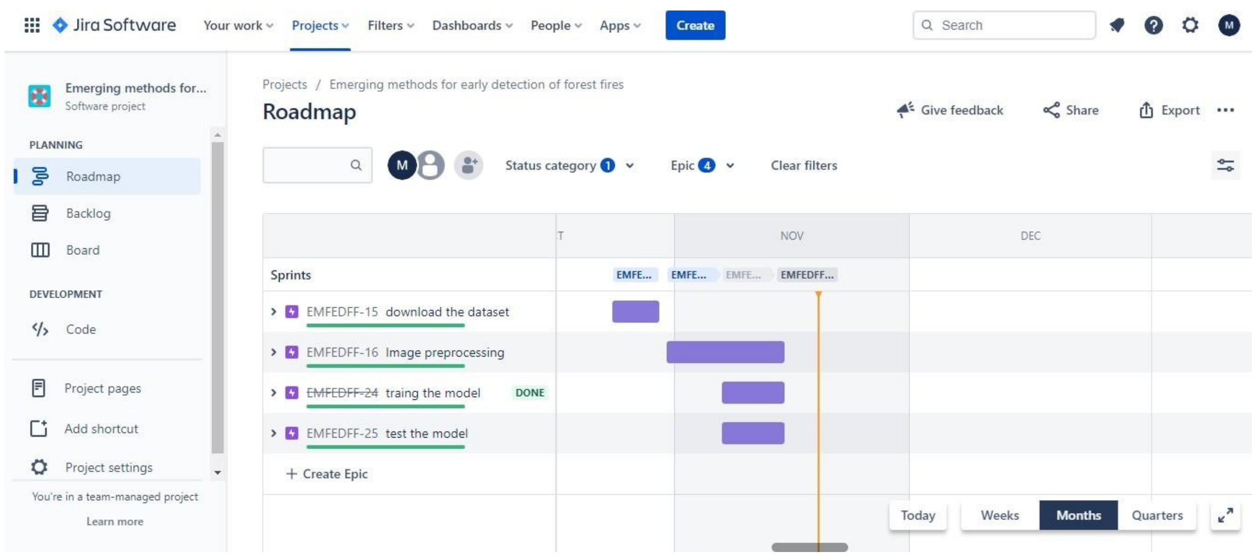
13.	PROJECT DEVELOPMENT PHASE	Project development-Delivery of Sprint-1. Project development-Delivery of Sprint-2. Project development-Delivery of Sprint-3. Project development-Delivery of Sprint-4.	All Members	In Progress

6.2. SPRINT DELIVERY SCHEDULE:

Sprint	Total Story Points	Duratio n	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	6 Days	24 Oct 2022	29 Oct 2022	5	29 Oct 2022
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	5	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	5	6 Days	14 Nov 2022	19 Nov 2022	5	19 Nov 2022

6.3. REPORTS FROM JIRA:





7.CODING & SOLUTIONING

7.1.FEATURE 1:

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.

- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

1. IMAGE DATA GENERATOR:

Keras ImageDataGenerator is used for getting the input of the original data and further, it makes the transformation of this data on a random basis and gives the output resultant containing only the data that is newly transformed. It does not add the data.

```
From keras.preprocessing.image import ImageDataGenerator
```

2. PARAMETRES

2.1. Rescale:

The ImageDataGenerator class can be used to rescale pixel values from the range of 0-255 to the range 0-1 preferred for neural network models. Scaling data to the range of 0-1 is traditionally referred to as normalization.

2.2. Shear Range:

Shear range means that the image will be distorted along an axis, mostly to create or rectify the perception angles. It's usually used to augment images so that computers can see how humans see things from different angles.

2.3. Rotation range:

ImageDataGenerator class allows you to randomly rotate images through any degree between 0 and 360 by providing an integer value in the rotation_range argument. When the image is rotated, some pixels will move outside the image and leave an empty area that needs to be filled in.

2.4. Zoom Range:

The zoom augmentation method is used to zooming the image. This method randomly zooms the image either by zooming in or it adds some pixels aroundthe image to enlarge the image. This method uses the zoom_range argument of the ImageDataGenerator class. We can specify the percentage value of the zooms either in a float, range in the form of an array.

2.5. Horizontal Flip:

Horizontal flip basically flips both rows and columns horizontally. So for this, we have to pass the horizontal_flip=True argument in the ImageDataGenerator constructor.

3. CONVOLUTION NEURAL NETWORK:

Convolutional neural network is one of the most popular ANN. It is widely used in the fields of image and video recognition. It is based on the concept of convolution, a mathematical concept. It is almost similar to multi-layer perceptron except it contains series of convolution layer and pooling layer before the fully connected hidden neuron layer.

3.1. Convolutional Layer:

Convolution layer: It is the primary building block and perform computational tasks based on convolution function.

3.2. Pooling Layer:

Pooling layer: It is arranged next to convolution layer and is used to reduce the size of inputs by removing unnecessary information so computation can be performed faster.

3.3. Fully Connected Layer:

Fully connected layer: It is arranged to next to series of convolution and pooling layer and classify input into various categories

7.2.FEATURE 2(CODE):

Image Pre-Processing:

```
from google.colab import drive
drive.mount('/content/drive')
```

Importing ImageDataGenerator from Keras

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

Defining the Parameters

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
```

```
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

val_datagen = ImageDataGenerator(rescale = 1./255)
```

Applying ImageDataGenerator functionality to train dataset

```
training_set = train_datagen.flow_from_directory('/content/drive
/MyDrive/Data set Collection',
                                        target_size = (512,512),
                                         batch_size = 8,
                                         class_mode = 'binary')
val_set = val_datagen.flow_from_directory('/content/drive/MyDriv
e/Data set Collection',
                                        target_size = (512,512),
                                        batch_size = 8,
                                        class_mode = 'binary')
x_train = train_datagen.flow_from_directory(r'/content/drive/MyD
rive/Data set Collection')
x_test = val_datagen.flow_from_directory('/content/drive/MyDrive
/Data set Collection')
```

Training code:

To train the model we will use the train.py file, which is located in the object_ detection/legacy folder. We will copy it into the object_detection folder and then we will open a command line and type default.

train.py - E:\foresttt\train.py (3.6.5)

File Edit Format Run Options Window Help

```python
import os
import tensorflow as tf

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
                     'Force clones to be deployed on CPU.  Note that even if '
                     'set to False (allowing ops to run on gpu), some ops may '
                     'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                     'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                     'Number of parameter server tasks. If None, does not use '
                     'a parameter server.')
flags.DEFINE_string('train_dir', '',
                    'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
                    'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
                    'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS
```

TESTING OBJECT DETECTOR

In order to test our newly created object detector, we can use the code which we already created.

final.py - E:\foresttt\final.py (3.6.5)

File Edit Format Run Options Window Help

```
sys.path.append('..')
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or la

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = 'training/labelmap.pbtxt'

detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABE
def email():
    with open('capture.jpg','rb')as f:
        file_data = f.read()
        file_type = imghdr.what(f.name)
        file_name = f.name
    msg.add_attachment(file_data, maintype = 'image', subtype = file_type, filen
    with smtplib.SMTP_SSL('smtp.gmail.com',465)as smtp:
        smtp.login(email_add,email_pass)
        smtp.send_message(msg)


def run_inference_for_single_image(image, graph):
    if 'detection_masks' in tensor_dict:
        # The following processing is only for single image
        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
        # Reframe is required to translate mask from box coordinates to image co
        real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
```

8.TESTING

8.1. Test Cases:

8.2. User Acceptance Testing:

1.Purpose of Document:

The purpose of this document is to briefly explain the test coverage and open issues of the Emerging Methods for Early Forest Fire Detection Project at the time of the release to User Acceptance Testing (UAT).

1. Defect Analysis:

This report shows the number of resolved or closed bugs at each severity level, and howthey were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subt otal
By Design	1	0	0	0	1
Duplicate	0	0	0	0	0
External	0	0	0	0	0
Fixed	0	0	0	0	0
Not Reproduced	0	2	0	0	2
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	1	2	0	0	3

2. Test Case Analysis:

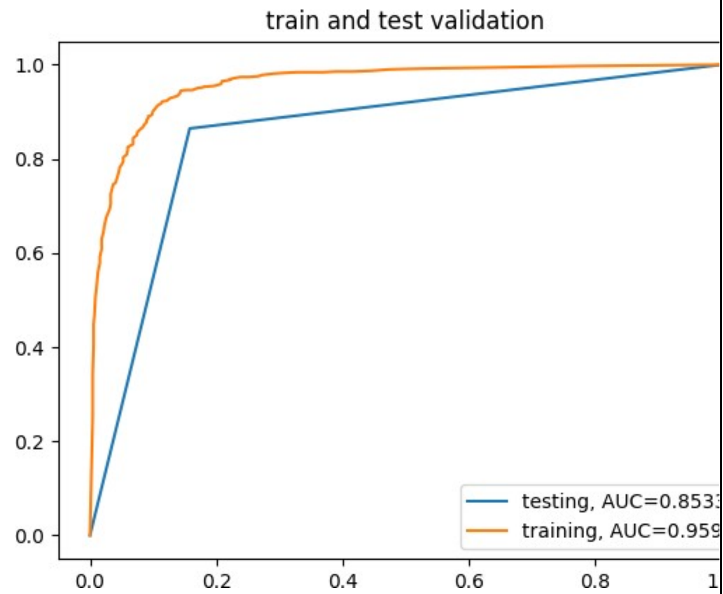
This report shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
Performance	5	0	0	5
UI	1	0	0	1
Security	3	0	0	3

9. RESULTS

9.1. PERFORMANCSE METRICS:

S.No.	Parameter	Values
1.	Model Summary	Model: "sequential_5" Layer (type) Output Shape Para conv2d_4 (Conv2D) (None, 126, 126, 32) 896 max_pooling2d_4 (MaxPooling 2D) (None, 63, 63, 32) 0 flatten_4 (Flatten) (None, 127008) 0 dense (Dense) (None, 256) 3251 dense_1 (Dense) (None, 1) 257 Total params: 32,515,457 Trainable params: 32,515,457 Non-trainable params: 0
2.	Accuracy	Training Accuracy - 94.50% Validation Accuracy - 98.35%

10. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

.

- ✓ CNN gives best accuracy when compared to machine learning techniques.
- ✓ Can be used for classification with above 96% accuracy.

DISADVANTAGES:

- The temperature sensor which has a particular range to detect fire in a forest.
- We can’t cover large area to detect fire in a whole forest.
- There is some specific range to sense the smoke using smoke sensor as same as temperature sensor which only covers limited area.

.

11. CONCLUSION

The recent improved processing capabilities of smart devices have shown promising results in surveillance systems for identification of different abnormal events i.e., fire, accidents, and other emergencies. Fire is one of the dangerous events which can result in great losses if it is not controlled on time. This necessitates the importance of developing early fire detection systems. Therefore, in this research article, we propose a cost-effective fire detection CNN architecture for forest architecture. Translations and content mining are permitted for academic research only. Although, this work improved the flame detection accuracy, yet the number of false alarms is still high and further research is required in this direction. In addition, the current flame detection frameworks can be intelligently tuned for detection of fire. This will enable the video surveillance systems on forest to handle more complex situations in real-world.

12. FUTURE SCOPE

- ❖ Supporting research to improve the understanding of forest fires and their ecology, ecological and social costs and benefits, causes and management options.
- ❖ Building awareness among st policy-makers, the public and the media of the underlying causes of catastrophic forest fires.
- ❖ Mandating and equipping managers to implement integrated fire management programs.

- ❖ Involving local communities and land managers in management planning and implementation, assisting them to participate effectively.
- ❖ Developing and enforcing compatible and mutually reinforcing land-use laws that provide a legal basis for the ecologically appropriate use of fire.
- ❖ Discouraging land management practices that predispose forests to harmful fires.
- ❖ Promoting management strategies to mimic natural fire regimes, including techniques such as prescribed burns and managed wildfires.
- ❖ Avoiding manipulating natural or well-established fire regimes.
- ❖ Establishing reliable fire monitoring systems that provide early warning of high fire risk and fire occurrence, and include evaluation of ecological and human impacts of fire.
- ❖ Preventing further forest loss and degradation from recurrent catastrophic fires, and reduce fire risk in forested landscapes, through ecologically appropriate restoration.

13. APPENDIX:

SOURCE CODE:

TRANING CODE:

```
import functools

import json
```

```
import os
import tensorflow as tf

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
                     'Force clones to be deployed on CPU. Note that even if '
                     'set to False (allowing ops to run on gpu), some ops may '
                     'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                     'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                     'Number of parameter server tasks. If None, does not use '
                     'a parameter server.')
flags.DEFINE_string('train_dir', '',
                    'Directory to save the checkpoints and training summaries.')
```

```
flags.DEFINE_string('pipeline_config_path', '',
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
                    'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
                    'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS


@tf.contrib.framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(_):
  assert FLAGS.train_dir, '`train_dir` is missing.'
  if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
  if FLAGS.pipeline_config_path:
    configs = config_util.get_configs_from_pipeline_file(
        FLAGS.pipeline_config_path)
    if FLAGS.task == 0:
      tf.gfile.Copy(FLAGS.pipeline_config_path,
                    os.path.join(FLAGS.train_dir, 'pipeline.config'),
                    overwrite=True)
  else:
    configs = config_util.get_configs_from_multiple_files(
```

```
      model_config_path=FLAGS.model_config_path,
      train_config_path=FLAGS.train_config_path,
      train_input_config_path=FLAGS.input_config_path)
  if FLAGS.task == 0:
    for name, config in [('model.config', FLAGS.model_config_path),
                         ('train.config', FLAGS.train_config_path),
                         ('input.config', FLAGS.input_config_path)]:
      tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                    overwrite=True)

model_config = configs['model']
train_config = configs['train_config']
input_config = configs['train_input_config']

model_fn = functools.partial(
    model_builder.build,
    model_config=model_config,
    is_training=True)

def get_next(config):
  return dataset_builder.make_initializable_iterator(
      dataset_builder.build(config)).get_next()

create_input_dict_fn = functools.partial(get_next, input_config)

env = json.loads(os.environ.get('TF_CONFIG', '{}'))
cluster_data = env.get('cluster', None)
```

```
cluster = tf.train.ClusterSpec(cluster_data) if cluster_data else None
task_data = env.get('task', None) or {'type': 'master', 'index': 0}
task_info = type('TaskSpec', (object,), task_data)

# Parameters for a single worker.
ps_tasks = 0
worker_replicas = 1
worker_job_name = 'lonely_worker'
task = 0
is_chief = True
master = ''

if cluster_data and 'worker' in cluster_data:
  # Number of total worker replicas include "worker"s and the "master".
  worker_replicas = len(cluster_data['worker']) + 1
if cluster_data and 'ps' in cluster_data:
  ps_tasks = len(cluster_data['ps'])

if worker_replicas > 1 and ps_tasks < 1:
  raise ValueError('At least 1 ps task is needed for distributed training.')

if worker_replicas >= 1 and ps_tasks > 0:
  # Set up distributed training.
  server = tf.train.Server(tf.train.ClusterSpec(cluster), protocol='grpc',
                           job_name=task_info.type,
                           task_index=task_info.index)
  if task_info.type == 'ps':
```

```
    server.join()
    return

  worker_job_name = '%s/task:%d' % (task_info.type, task_info.index)
  task = task_info.index
  is_chief = (task_info.type == 'master')
  master = server.target

graph_rewriter_fn = None
if 'graph_rewriter_config' in configs:
  graph_rewriter_fn = graph_rewriter_builder.build(
      configs['graph_rewriter_config'], is_training=True)

trainer.train(
    create_input_dict_fn,
    model_fn,
    train_config,
    master,
    task,
    FLAGS.num_clones,
    worker_replicas,
    FLAGS.clone_on_cpu,
    ps_tasks,
    worker_job_name,
    is_chief,
    FLAGS.train_dir,
    graph_hook_fn=graph_rewriter_fn)
```

```
if__name__== '_main__':
  tf.app.run()
```

TEST CODE:

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import sys
import tensorflow as tf
from distutils.version import StrictVersion
from collections import defaultdict
from object_detection.utils import ops as utils_ops
import os
from twilio.rest import Client



account_sid = 'AC17937b910b4774c95b3b07358838a842'
auth_token = '4b511b022adaac785f6b79b32cac4f28'
client1 = Client(account_sid, auth_token)
def sms():
    message = client1.messages \
            .create(
                body='fire detected successfully',
                from_='+13465214387',
                to='+9199434 63572'
                )
```

```
    print(message.sid)

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or later!')

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = 'training/labelmap.pbtxt'

detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
```

```
    od_graph_def.ParseFromString(serialized_graph)

    tf.import_graph_def(od_graph_def, name='')


category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)


def run_inference_for_single_image(image, graph):

    if 'detection_masks' in tensor_dict:

        # The following processing is only for single image

        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])

        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])

        # Reframe is required to translate mask from box coordinates to image coordinates and fit the image
size.

        real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)

        detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection, -1])

        detection_masks = tf.slice(detection_masks, [0, 0, 0], [real_num_detection, -1, -1])

        detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(

            detection_masks, detection_boxes, image.shape[0], image.shape[1])

        detection_masks_reframed = tf.cast(

            tf.greater(detection_masks_reframed, 0.5), tf.uint8)

        # Follow the convention by adding back the batch dimension

        tensor_dict['detection_masks'] = tf.expand_dims(

            detection_masks_reframed, 0)

    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')


    # Run inference

    output_dict = sess.run(tensor_dict,

                        feed_dict={image_tensor: np.expand_dims(image, 0)})
```

```
    # all outputs are float32 numpy arrays, so convert types as appropriate

    output_dict['num_detections'] = int(output_dict['num_detections'][0])

    output_dict['detection_classes'] = output_dict[

        'detection_classes'][0].astype(np.uint8)


    output_dict['detection_boxes'] = output_dict['detection_boxes'][0]

    output_dict['detection_scores'] = output_dict['detection_scores'][0]

    if 'detection_masks' in output_dict:

        output_dict['detection_masks'] = output_dict['detection_masks'][0]

    if output_dict['detection_classes'][0] == 1 and  output_dict['detection_scores'][0] > 0.70:

      print('FIRE')

      sms()


    if output_dict['detection_classes'][0] == 2 and  output_dict['detection_scores'][0] > 0.70:

      print('FIRE')

      sms()


    if ((output_dict['detection_classes'][0] == 1 or output_dict['detection_scores'][0] < 0.70) and
(output_dict['detection_classes'][0] == 2 or  output_dict['detection_scores'][0] < 0.70)):

      print('No Fire')


    return output_dict




import cv2

cap = cv2.VideoCapture(0)
```

```
try:
    with detection_graph.as_default():
        with tf.Session() as sess:
                # Get handles to input and output tensors
                ops = tf.get_default_graph().get_operations()
                all_tensor_names = {output.name for op in ops for output in op.outputs}
                tensor_dict = {}
                for key in [
                  'num_detections', 'detection_boxes', 'detection_scores',
                  'detection_classes', 'detection_masks'
                ]:
                    tensor_name = key + ':0'
                    if tensor_name in all_tensor_names:
                        tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
                      tensor_name)

                while True:
                    ret, image_np = cap.read()
                    # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
                    image_np_expanded = np.expand_dims(image_np, axis=0)
                    # Actual detection.
                    output_dict = run_inference_for_single_image(image_np, detection_graph)
                    # Visualization of the results of a detection.
                    vis_util.visualize_boxes_and_labels_on_image_array(
                        image_np,
                        output_dict['detection_boxes'],
```

```
          output_dict['detection_classes'],
          output_dict['detection_scores'],
          category_index,
          instance_masks=output_dict.get('detection_masks'),
          use_normalized_coordinates=True,
          line_thickness=8)
        cv2.imshow('Frame', cv2.resize(image_np,(800,600)))
        if cv2.waitKey(1) == ord('q'):
          cap.release()
          cv2.destroyAllWindows()
          break
except Exception as e:
  print(e)
  cap.release()
```

train.py - E:\foresttt\train.py (3.6.5)

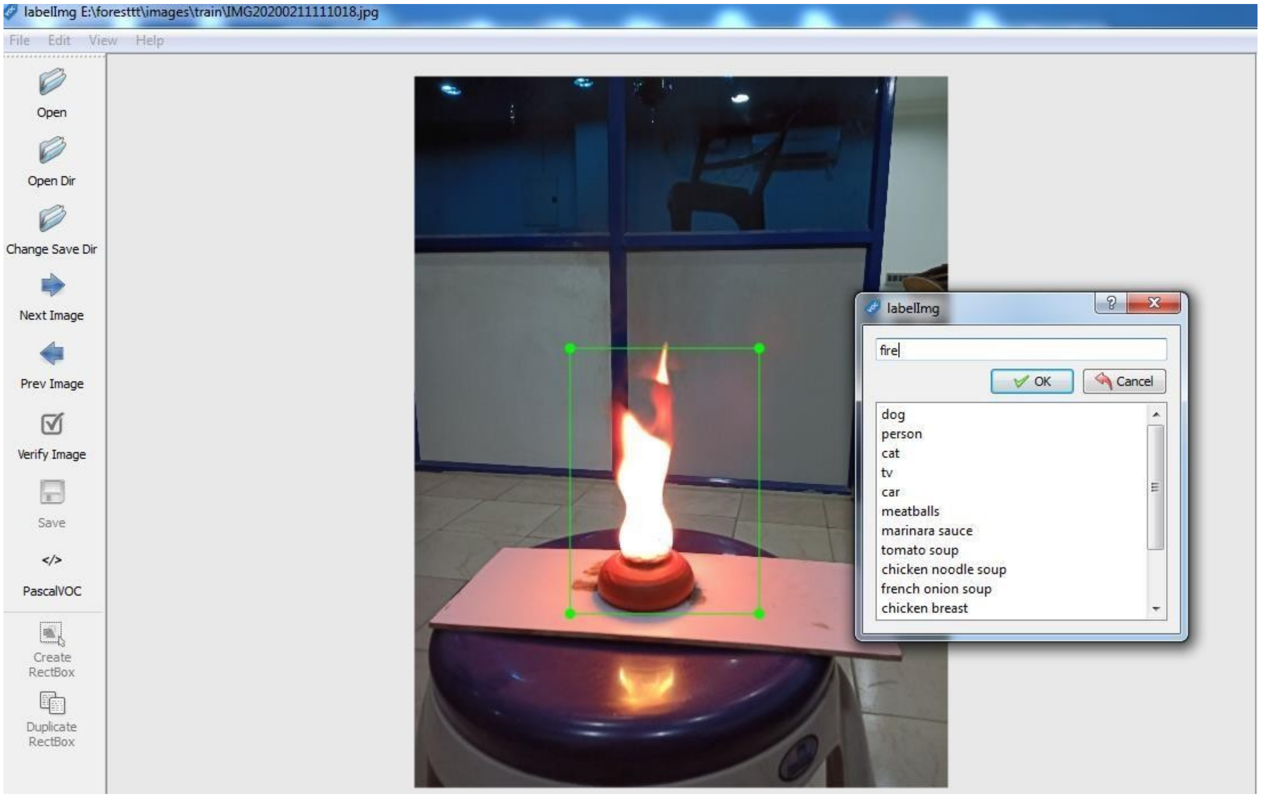
File Edit Format Run Options Window Help

```python
import os
import tensorflow as tf

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
                     'Force clones to be deployed on CPU.  Note that even if '
                     'set to False (allowing ops to run on gpu), some ops may '
                     'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                     'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                     'Number of parameter server tasks. If None, does not use '
                     'a parameter server.')
flags.DEFINE_string('train_dir', '',
                    'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
                    'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
                    'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS
```

final.py - E:\foresttt\final.py (3.6.5)

File Edit Format Run Options Window Help

```
sys.path.append('..')
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or la

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = 'training/labelmap.pbtxt'

detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABE
def email():
    with open('capture.jpg','rb')as f:
        file_data = f.read()
        file_type = imghdr.what(f.name)
        file_name = f.name
    msg.add_attachment(file_data, maintype = 'image', subtype = file_type, filen
    with smtplib.SMTP_SSL('smtp.gmail.com',465)as smtp:
        smtp.login(email_add,email_pass)
        smtp.send_message(msg)


def run_inference_for_single_image(image, graph):
    if 'detection_masks' in tensor_dict:
        # The following processing is only for single image
        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
        # Reframe is required to translate mask from box coordinates to image co
        real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
```



Our Github Link:https://github.com/IBM-EPBL/IBM-Project-4643-1658736950