EMERGING

METHODS FOR EARLY DETECTION

OF FOREST FIRES

**INTRODUCTION**

**PROJECT OVERVIEW**

Over the last few decades, forest fires are increased due to deforestation and global warming. Many trees and animals in the forest are affected by forest fires. Technology can be efficiently utilized to solve this problem. Forest fire detection is inevitable for forest fire management. The purpose of this work is to propose deep learning techniques to predict forest fires, which would be cost-effective. Unmanned aerial vehicles (UAVs) are promising options to patrol the forest by making them fly over the region. The proposed model deployed on an onboard UAV has achieved 1.24 seconds of classification time with an accuracy of 91% and an F1 score of 0.91. The onboard CPU is able to make a 3D model of the forest fire region and can transmit the data in real time to the ground station. The proposed model is trained on both dense and rainforests in detecting and predicting the chances of fire. The proposed model outperforms the traditional methods such as Bayesian classifiers, random forest, and support vector machines

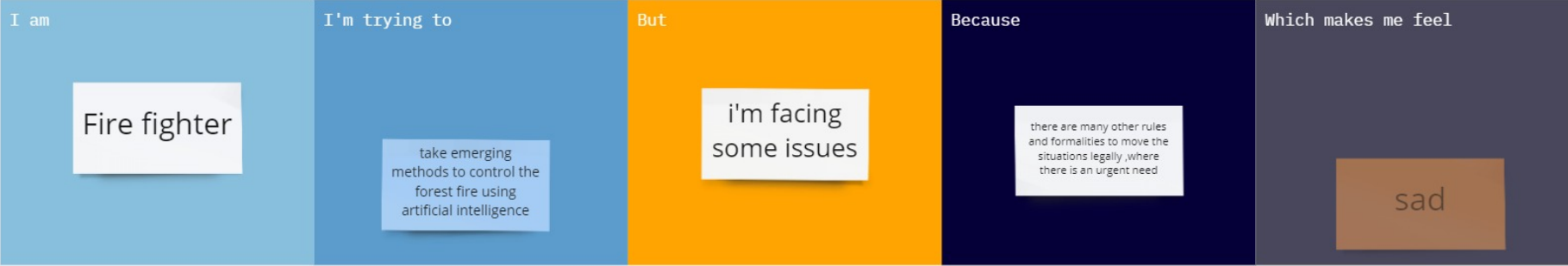
**PURPOSE**

Collected data in real time provide fast intervention of relevant services to extinguish the fire. Timely information about the appearance of fire reduce the number of areas affected by this fire and thereby minimizes the costs of fire extinguishing and the damage caused in the woods. The current way of detecting fire in an open area in Serbia is not  in real time, and due to this, it is necessary to implement modern technology of collecting data related to early detection of fires. This paper presents an integral project of forest-fire protection on the territory of Serbia in order to provide the reference for the application of terrestrial automated system for early detection and prediction of forest fires. An automated system could be comprised of infrared and high-resolution TV camera surveillance, covering a large part of the forest area and forest land.

**LITERATURE**

**SURVEY**

**EXISTING PROBLEM**

****

**REFERENCES**

1. Ollero A., rrue B. C., Martinez J. R. For reducing false alarms in forest-fires. *Computer Communications S0140366419308655–*2019 doi: 10.1016/j.comcom.2019.10.007.

2. Yuan C., Liu Z., Zhang Y. Fire Detection Using Infrared Images for UAV-Based Forest Fire Surveillance. Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS); June, 2017; Miami, FL, USA.

3. Wu X., Leung X., Leung H. An adaptive threshold deep learning method for fire and smoke detection. Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC); October, 2017; Banff, AB, Canada. pp. 1954–1959.

4. Yuan C., Liu Z., Zhang Y. Learning-based smoke detection for unmanned aerial vehicles applied to forest fire surveillance. *Journal of Intelligent & Robotic Systems .*2018;**93**(1-2):337–349. doi: 10.1007/s10846-018-0803-y.

5. Amer K., Samy M., ElHakim R., Shaker M., ElHelw M. Convolutional neural network-based deep urban signatures with application to drone localization. Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW); October, 2017; Venice, Italy. IEEE; p. p. 2138. –2145.

6. Girshick R. Fast r-cnn. Proceedings of the IEEE International Conference on Computer Vision; December, 2015; Santiago, Chile. pp. 1440–1448.

7. Ramakrishna R., Rajeevan M., Ramakrishna S. Prediction of severe thunderstorms over Sriharikota Island by using the WRF-ARW operational model; *SPIE Proceedings* ; 2016; 988214

8. HussinM R., Juhari R. Detection using image processing based techniques. *Indian Journal of Computer Science and Engineering .*2012;**41**

9. Guede-Fernández F., Martins L., de Almeida R. V., Gamboa H., Vieira P. A deep learning based object identification system for forest fire detection. *Fire .*2021;**4**(4):p. 75. doi: 10.3390/fire4040075

10. Mueller M., Kolesov P., Tannenbaum I., Tannenbaum A. Optical flow estimation for flame detection in videos. *IEEE Transactions on Image Processing .*2013;**22**(7):2786–2797. doi: 10.1109/tip.2013.2258353.

11. Chen P., Zhang T., Xin K., Yi N., Liu D., Liu H. A UAV-based forest fire detection algorithm using convolutional neural network. Proceedings of the 2018 37th Chinese Control Conference (CCC); June, 2018; Wuhan, China. pp. 10305–10310.

12. Jiao Z., Zhang Y., Xin X., et al. A deep learning based forest fire detection approach using UAV and YOLOv3. Proceedings of the 2019 1st International Conference on Industrial Artificial Intelligence (IAI); July, 2019; Shenyang, China. pp. 1–5.

13. Kinaneva D., Hristov G., Raychev J., Zahariev P. Early forest fire detection using drones and artificial intelligence. Proceedings of the 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO); May, 2019; Opatija, Croatia. pp. 1060–1065.

14. Wyder P. M., Chen Y. S., Lasrado A. J., et al. Autonomous drone hunter operating by deep learning and all-onboard computations in GPS-denied environments. *PLoS ONE .*2019;**14**(11) doi: 10.1371/journal.pone.0225092.e0225092

15. Cortez P., Morais A. A Data Mining Approach to Predict Forest Fires Using Meteorological Data. 2017.

16. Yadav G., Gupta V., Gaur V., Bhattacharya Mahua, OPTIMIZED FLAME. *Indian Journal of Computer Science and Engineering .*2012;**3**(2)

17. Lakshmanna K., Khare N., Khare N. Constraint-based measures for DNA sequence mining using group search optimization algorithm. *International Journal of Intelligent Engineering and Systems .*2016;**9**(3):91–100. doi: 10.22266/ijies2016.0930.09

18. Sudhakar S., Vijayakumar V., Kumar C. S., Priya V., Ravi L., Subramaniyaswamy V. Unmanned Aerial Vehicle (UAV) Based Forest Fire Detection and Monitoring. *Computer Communications .*2019;**149**

19. Poobalan K., Liew S. C. Fire Detection Algorithm Using Image Processing Techniques. Proceedings of the International Conference on Artificial Intelligence and Computer Science; October, 2017; Penang, Malaysia.

20. Haque M., Muhammad M., Swarnaker D., Arifuzzaman M. Autonomous quadcopter for product home delivery. Proceedings of the 2014 International Conference on Electrical Engineering and Information & Communication Technology; April, 2014; Dhaka, Bangladesh. pp. 1–5.

**PROBLEM STATEMENT DEFINITION**

A Large destructive fire that spread over a forest or area of woodland which leads to damage in Wildlife, humans, property and Environment. The major Causes Are Lightning. Sparks from Rock falls. Volcanic Eruption or any other manual Ignition from the Humans on purpose which leads to the following disadvantages: A forest fire sets up the potential for soil erosion to occur, Forest fires always bring death to life of humans and animals, Uncontrolled fires can cause localized air pollution, Homes can be destroyed without compensation

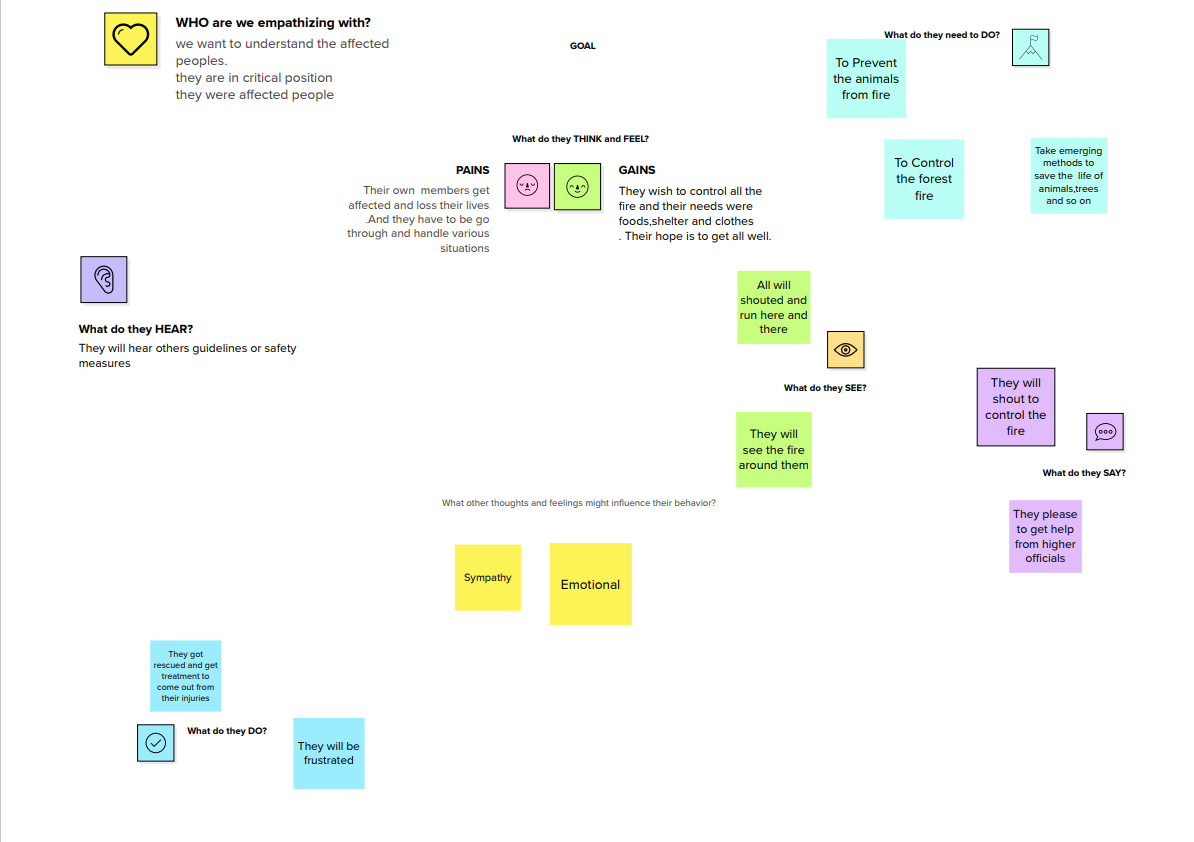


**IDEATION &**

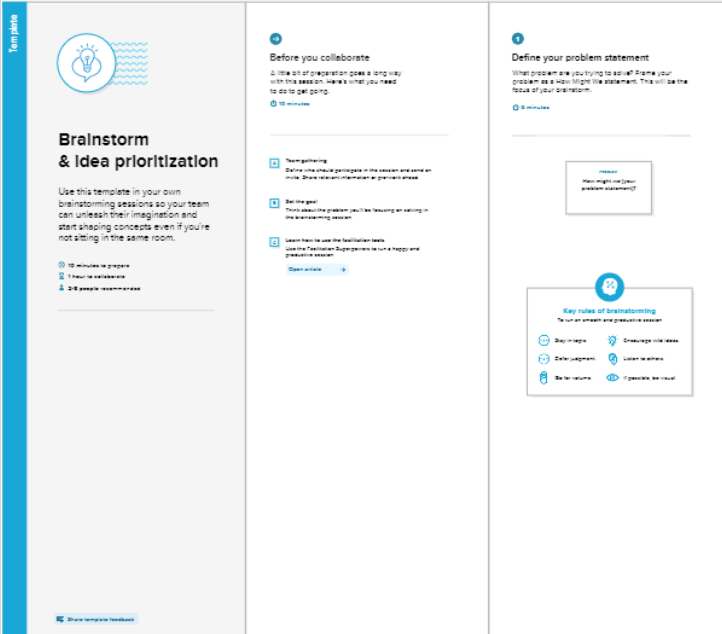
**PROPOSED**

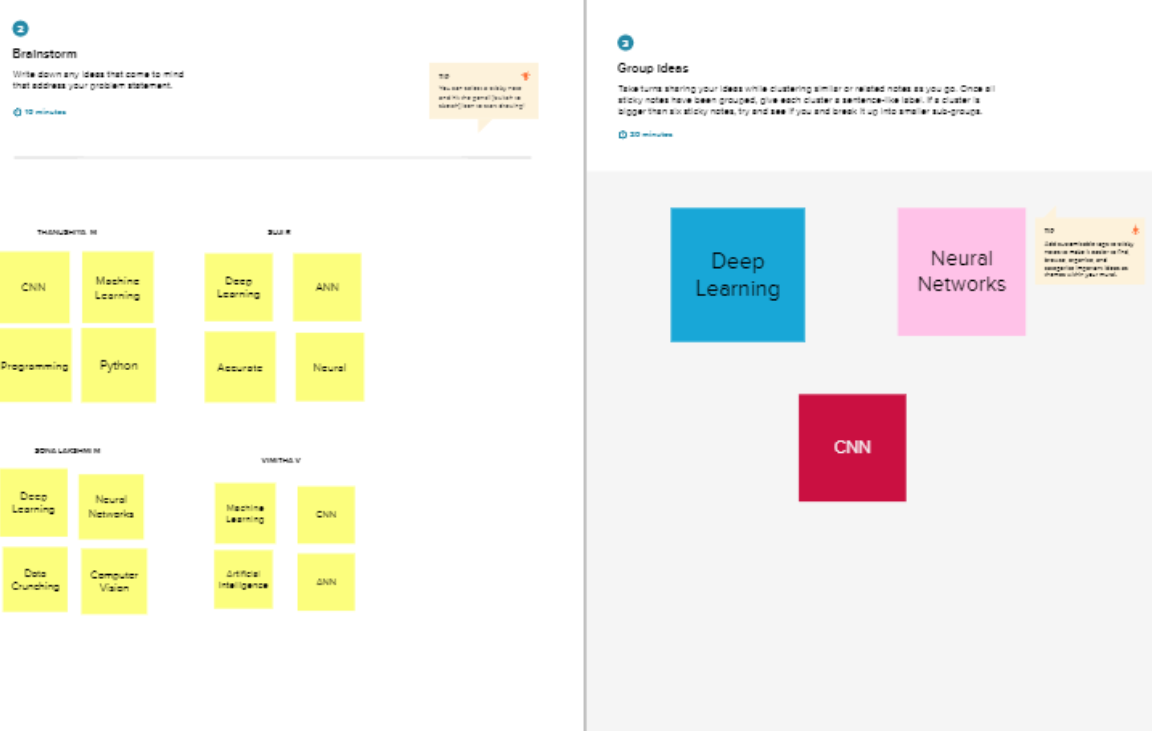
**SOLUTION**

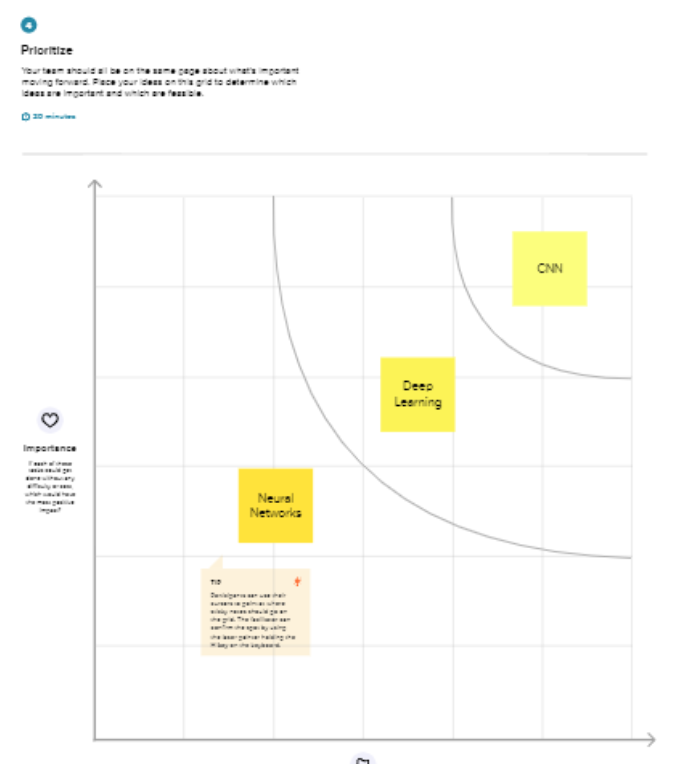
**EMPATHY MAP CANVAS**

****

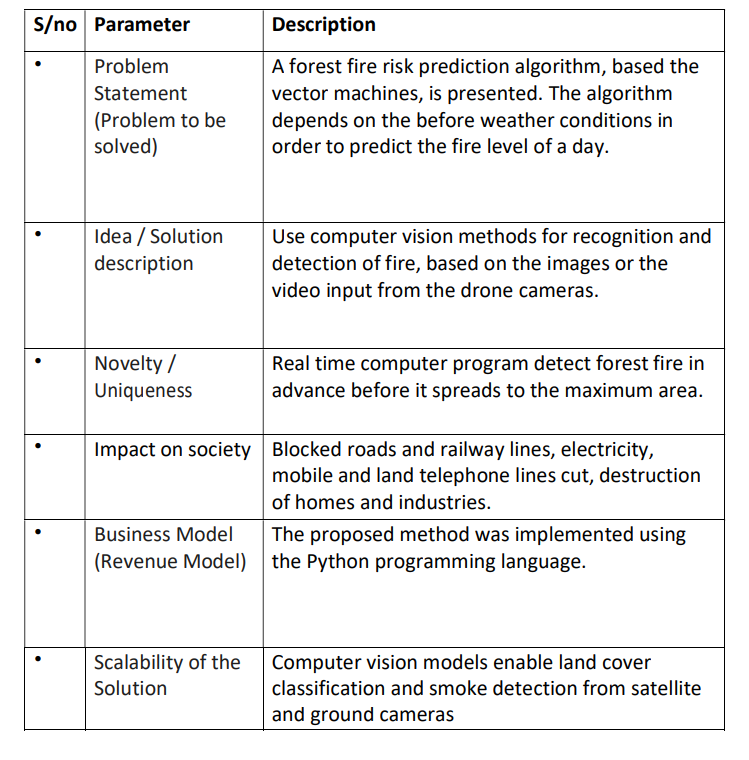
**IDEATION & BRAINSTORMING**



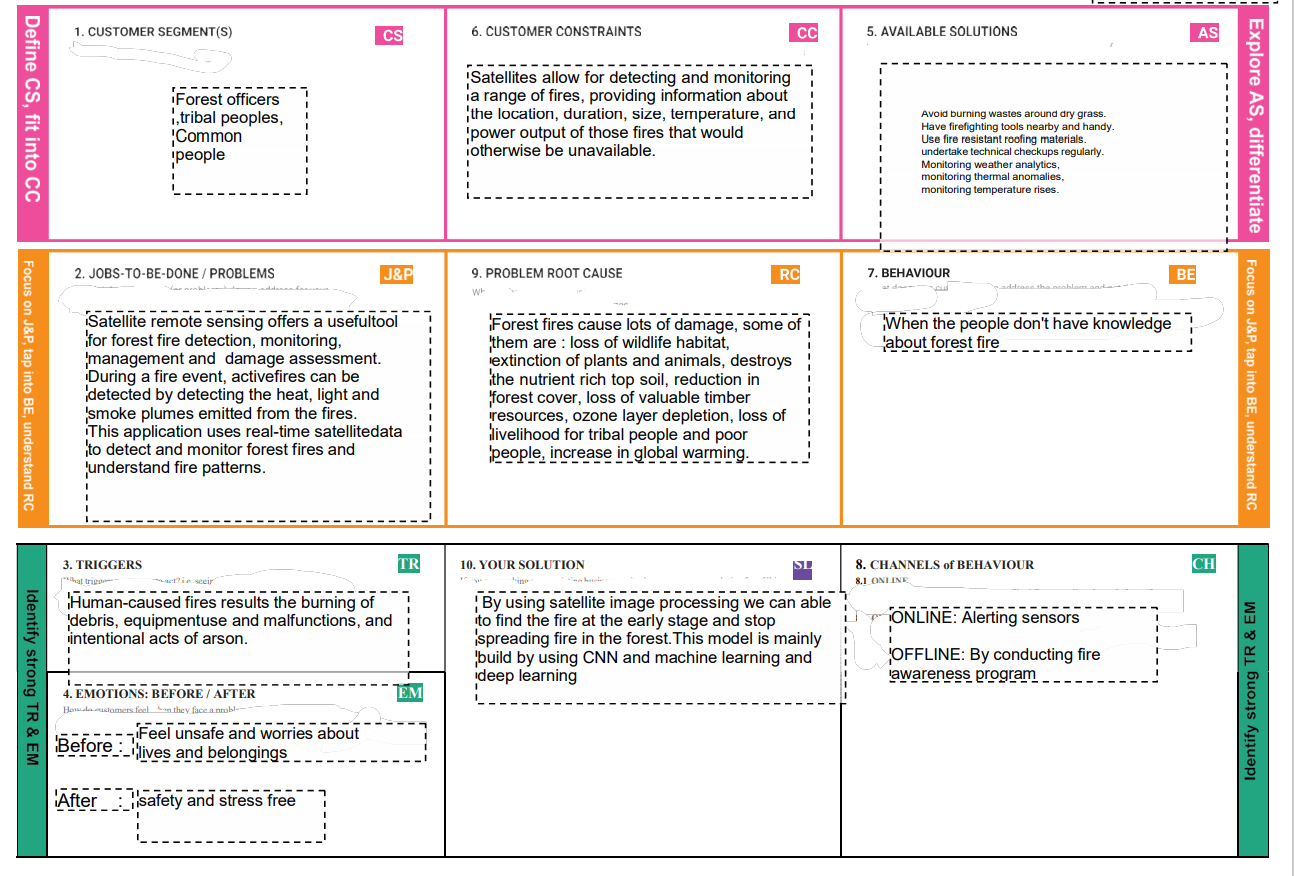




PROPOSED SOLUTION



PROBLEM SOLUTION FIT



**REQUIREMENT**

**ANALYSIS**

**FUNCTIONAL REQUIREMENT**

1. Introduction

1.1 Purpose of Document

This is a Requirements Specification document for Emerging methods for early detection of forest fire. This project is used to detect forest fire early so that it could reduce vulnerability of upcoming disaster. This document describes the scope, objectives and goal of the new system. In addition to describing non-functional requirements, this document models the functional requirements with use cases, interaction diagrams, and class models.

1.2 Project Summary Project Name :

Emerging methods for early detection of forest fire Project Leader : Vimitha V

Project Members :

Thanushiya M

Suji R

SonaLakshmi M

1.3 Background

Trees are valuable carbon repositories and play an important role for the climate. It takes decades to reforest areas ravaged by wildfires. Much of this ground lies fallow for a very long time, which takes a further toll on the climate forest fire, uncontrolled fire occurring in vegetation more than 1.8 meters (6 feet) in height. These fires often reach the proportions of a major conflagration and are sometimes begun by combustion and heat from surface and and ground fires. A big forest fire may crown—that is, spread rapidly through the topmost

Though forest fire is often seen as harmful, a number of forests are specifically fire-adapted: the species of plants and animals native to those ecosystems are enhanced by or dependent on the occurrence of fire to persist and reproduce. Lightning strikes in a dry forest occur naturally, and fire can improve ecosystem health by reducing competition, fertilizing the soil with ash, and decreasing diseases and pests. Some plant species even require fire for their seeds to germinate. In many regions that have historically experienced forest fires, such as forested areas of the western United States, years of fire exclusion and suppression in the 19th and 20th centuries allowed fuels to accumulate, altering the vegetation communities present and leading to more extreme conflagrations when fires do occur. The use of prescribed fire, in which areas are burned intentionally and under controlled conditions, can restore those ecosystems and promote the conditions that were present historically prior to the removal of wildfire. There are a number of detection and monitoring systems used by authorities. These include observers in the form of patrols or monitoring towers, aerial and satellite monitoring and increasingly promoted detection and monitoring systems based on optical camera sensors, and different types of detection sensors or their combination. The following part presents a brief overview of automatic and semiautomatic detection and monitoring systems of fire protection in the world, experience with these systems in practical operation, and their evaluation in terms of efficiency, accuracy, versatility, and other key attributes. The most frequently used fire detection and suppression techniques employed by authorities can be summarized as follows: (i) controlled burning, (ii) fire weather forecasts and estimates of fuel and moisture, (iii) watch towers, (iv) lightning detectors which detect the coordinates of the strike, (v) infrared and spotter planes (vi) water tankers ,mobile /smart phone calls becoming increasingly common

**NON FUNCTIONAL REQUIREMENTS**

Reliability:

• The system shall be completely operational at least x% of the time.

• Down time after a failure shall not exceed x hours.

Usability:

• Customer should be able to use the system in his job for x days.

• A user who already knows what camera he is using should be able to connect and view that page in x seconds.

Performance:

• The system should be able to support x simultaneous users.

• The mean time to view a web page over a 56Kbps modem connection shall not exceed x seconds.

Security:

• The system shall provide password protected access to web pages that are to be viewed only by users.

Supportability:

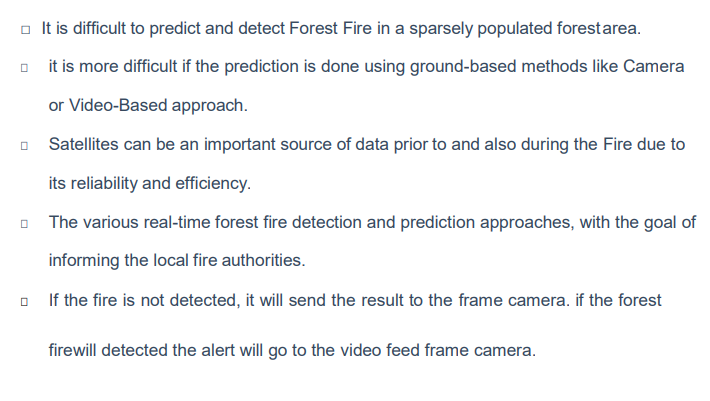
• The system should be able to accommodate many camera links. • The system web site shall be viewable from chrome or any browser.

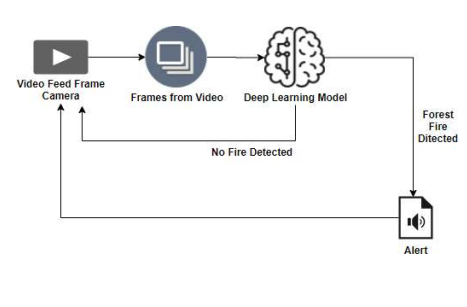
Interfaces: The system must interface with • The cloud and db for customer and customer log information

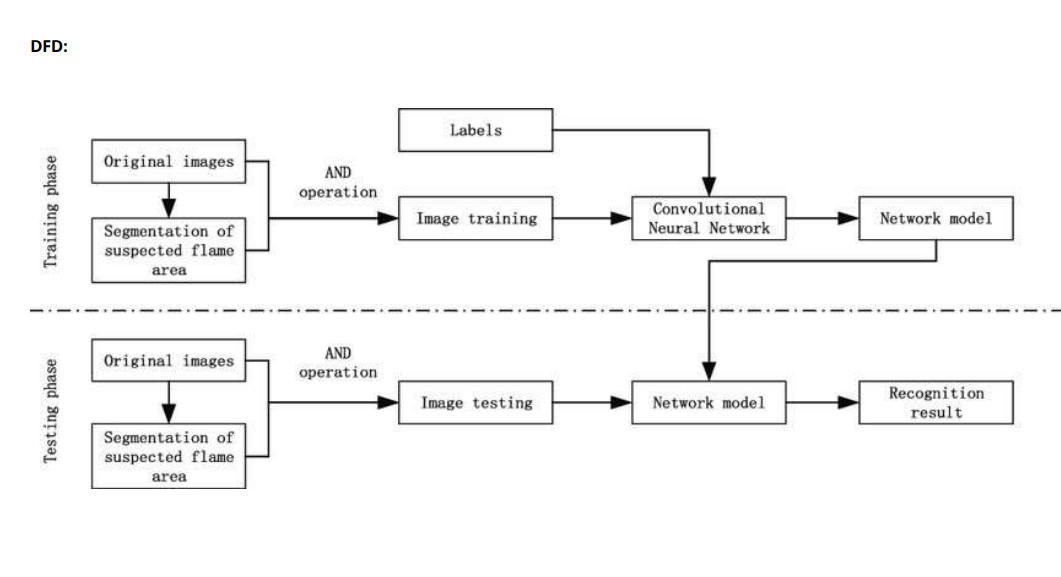
**PROJECT**

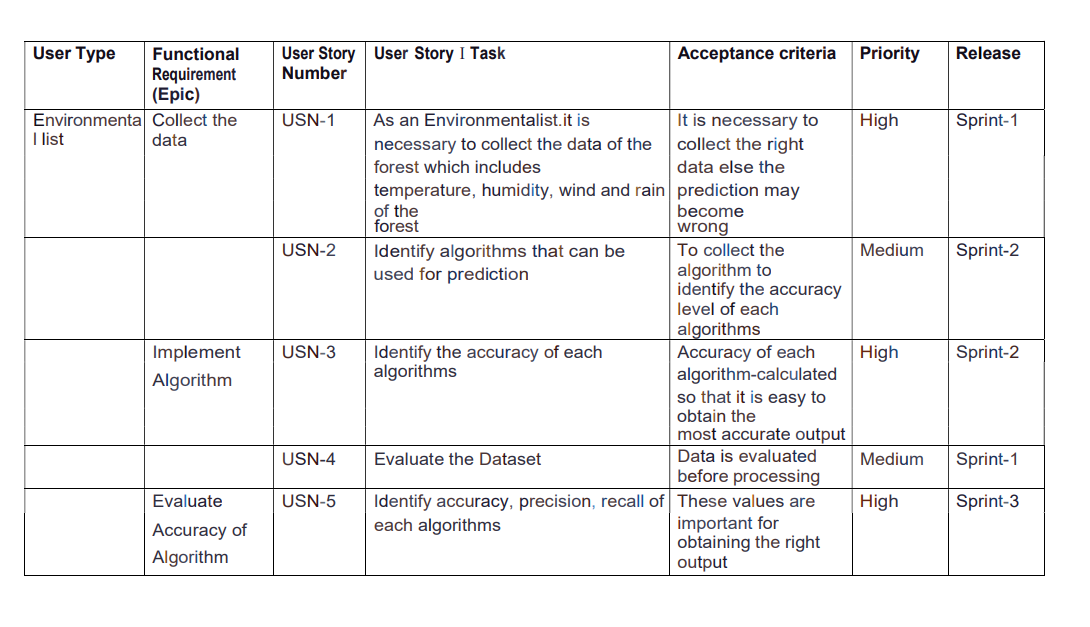
**DESIGN**

**DATA FLOW DIAGRAMS**

****

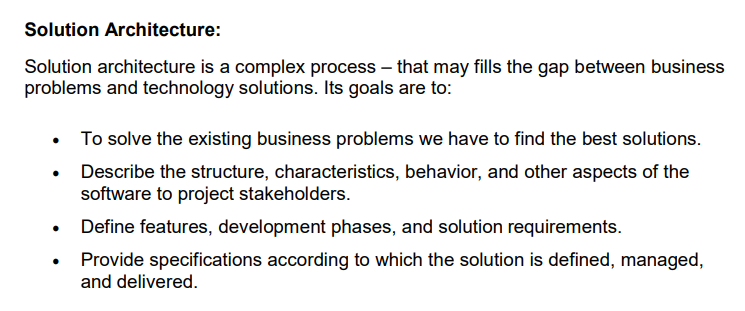


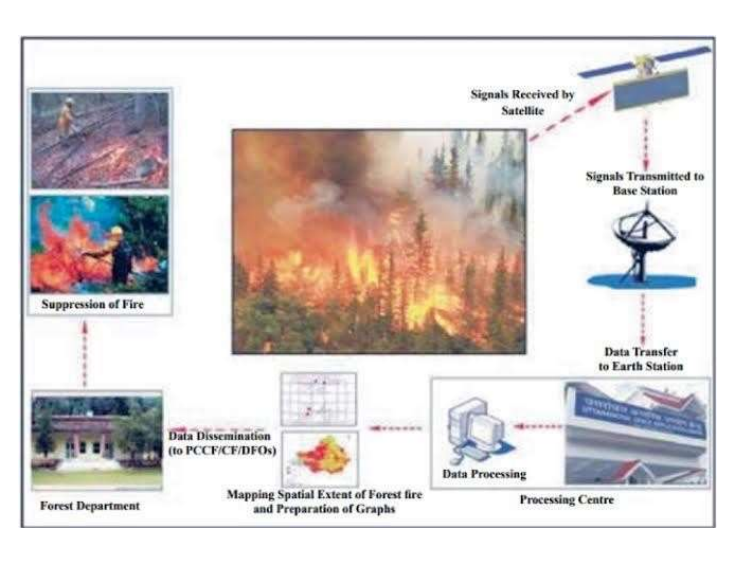




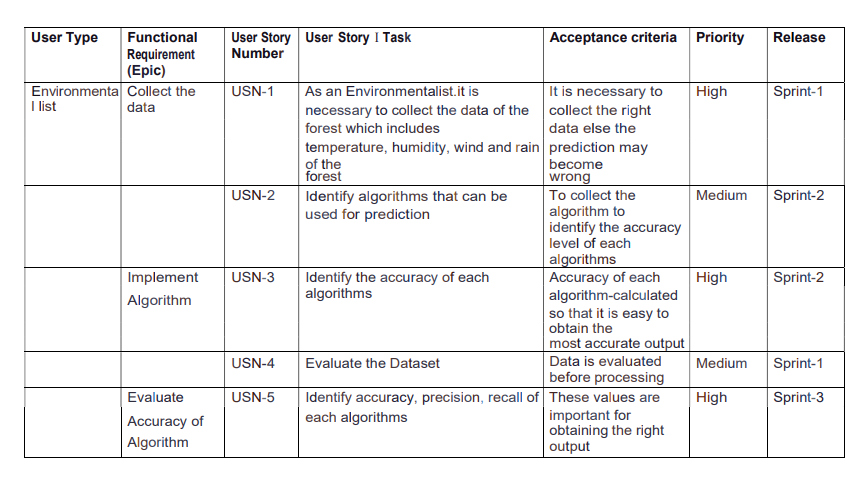
TECHNICAL & SOLUTION

ARCHITECTURE





**USER STORIES**

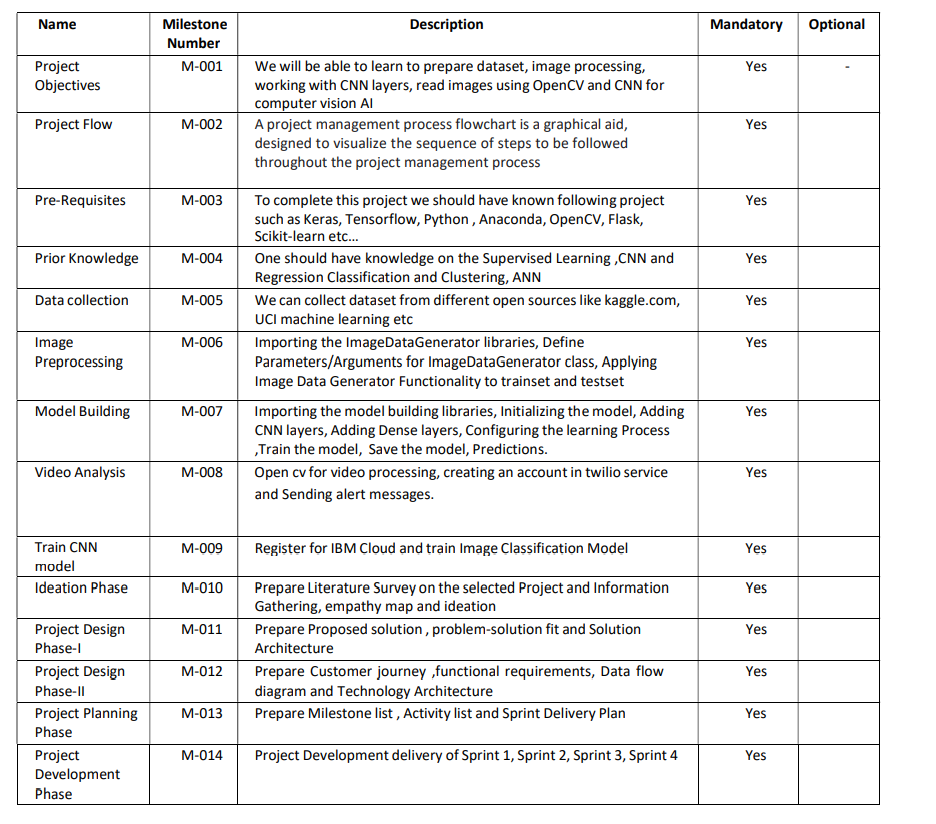
****

**PROJECT**

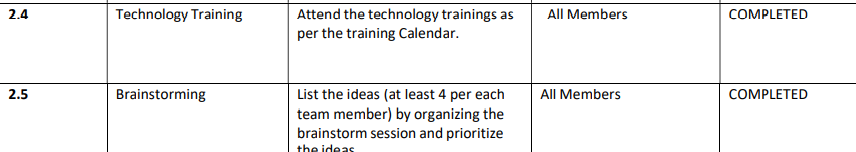
**PLANNING &**

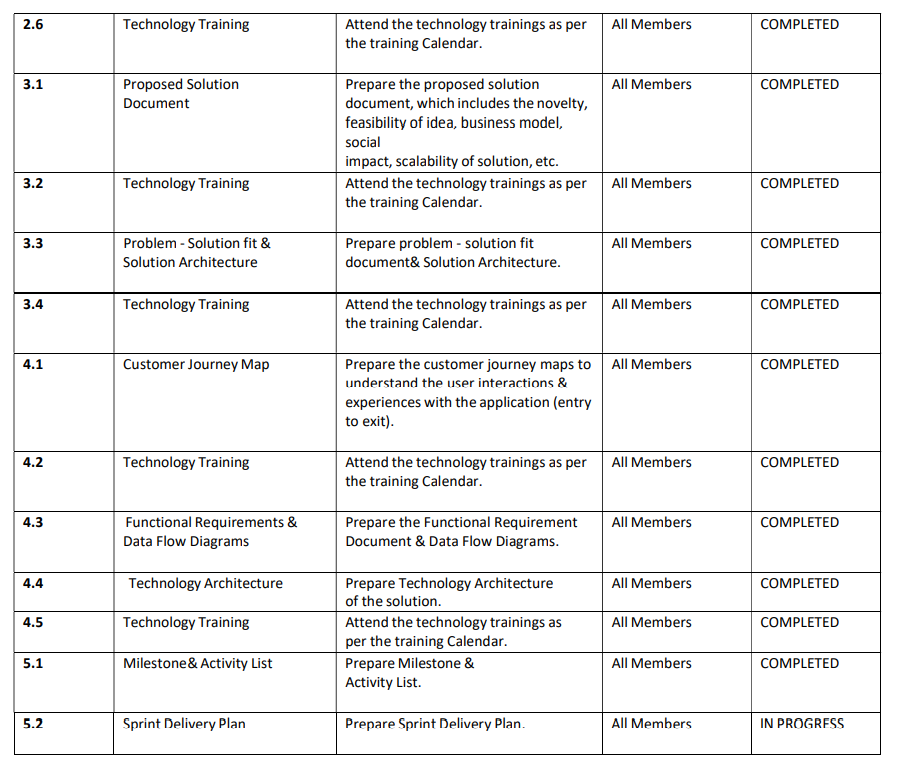
**DESIGNING**

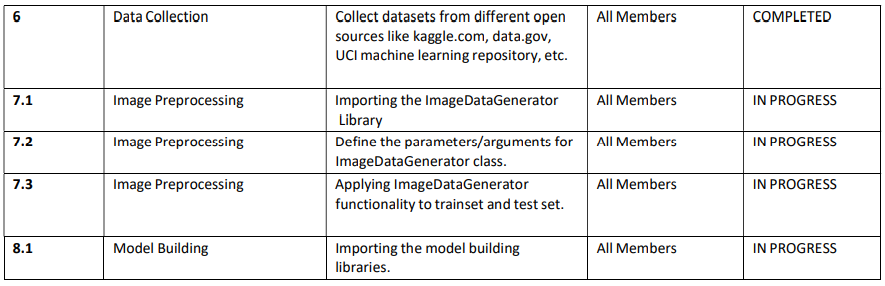
**SPRINT PLANNNG & ESTIMATION**

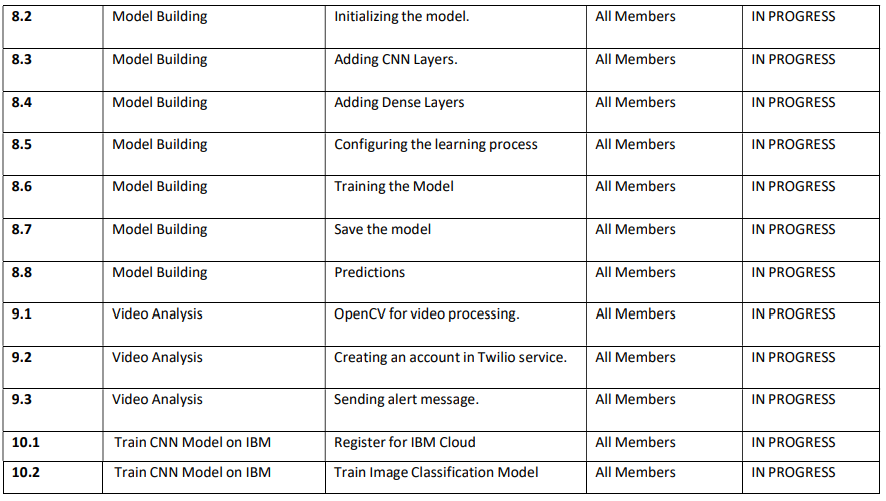
****

****

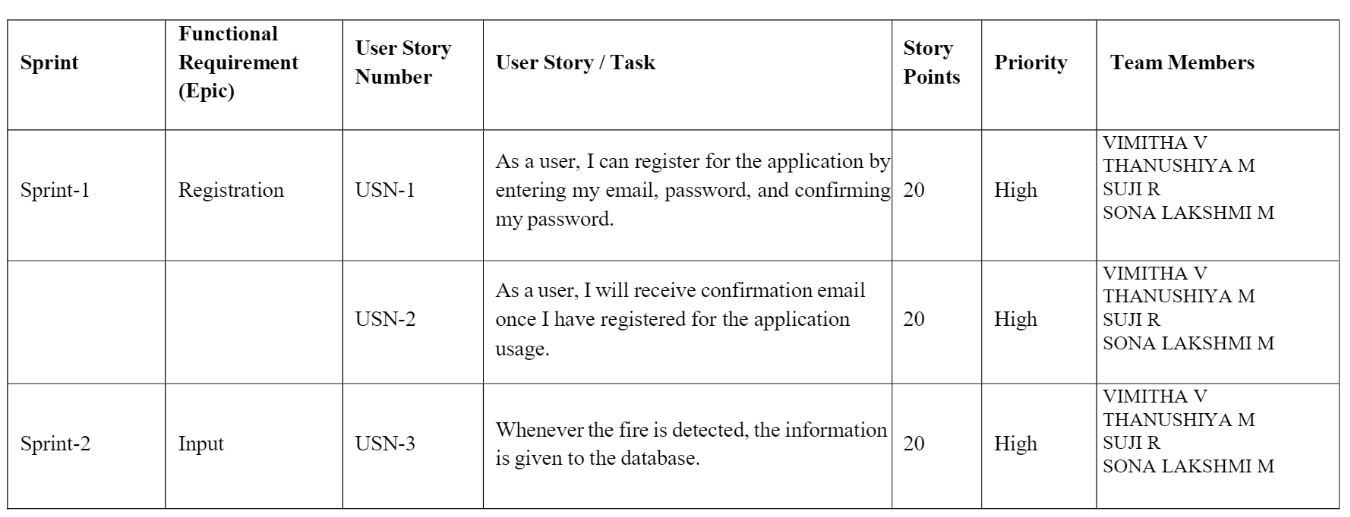
****

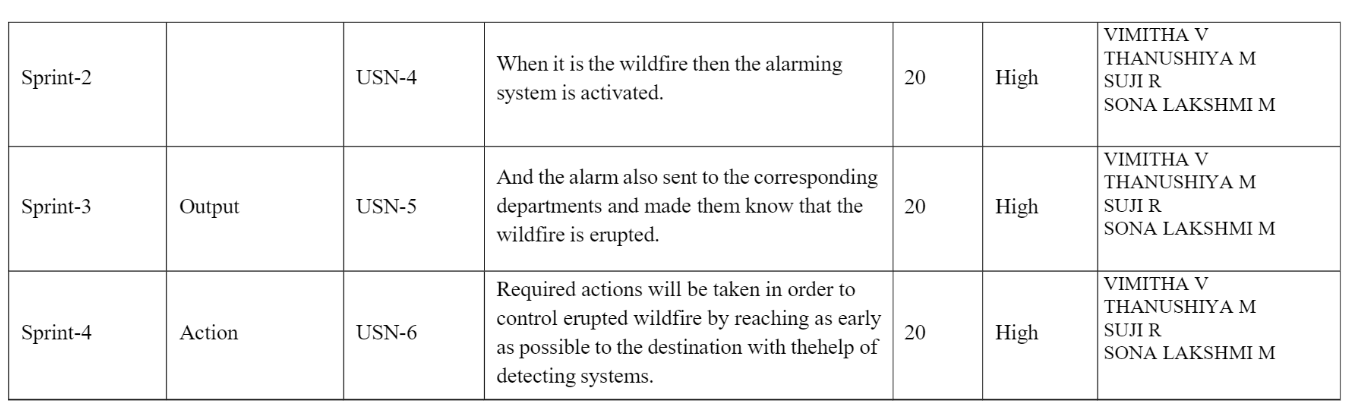
****

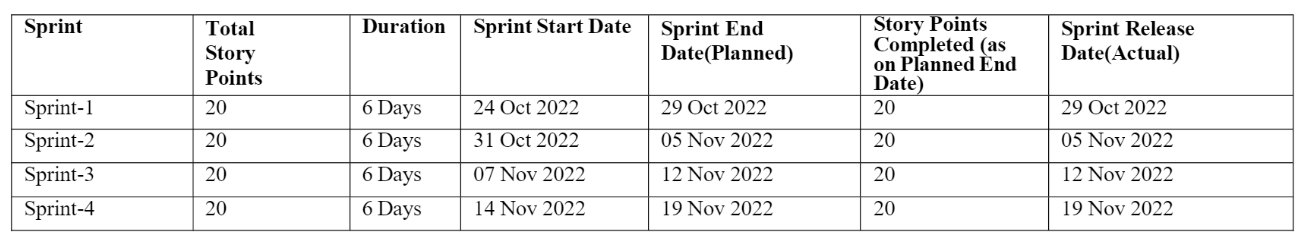
****

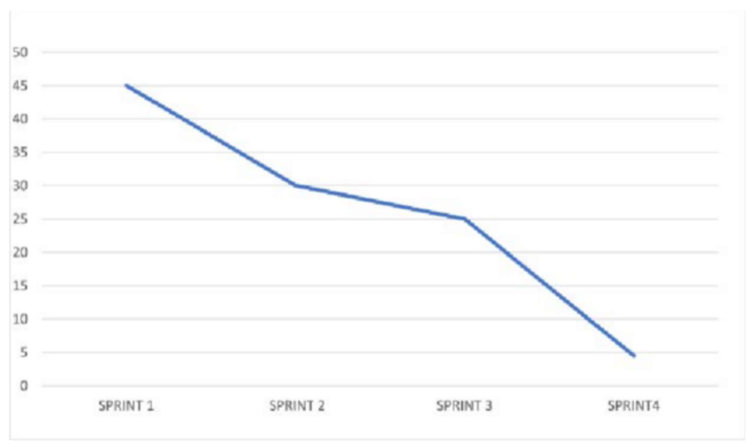
****

**SPRINT DELIVERY SCHEDULE**

****

****

****

****

**CODING &**

**SOLUTIONING**

**FEATURE 1**

Preprocessing includes steps that are required to shape the input image into a form

suitable for segmentation. Colour image is converted into grayscale image transform into binary image that means in the form of black in white image.

As each colour pixel is described by a triplet (R,G,B) of intensities for red, green and blue colour. we can map that to a single number giving a grayscale value. There are many approaches to convert colour image into grayscale. Here average method is used for colour to grayscale conversion.

**FEATURE 2**

Once image pre-processing is done it is necessary to segment document into lines, lines into words and words into characters. When characters has been extracted from document we can extract features from it for recognition. Segmentation of image is performed to separate the characters from the image. Characters separation from the input image involves three steps as:

* Word Segmentation
* Line Segmentation
* Character Segmentation

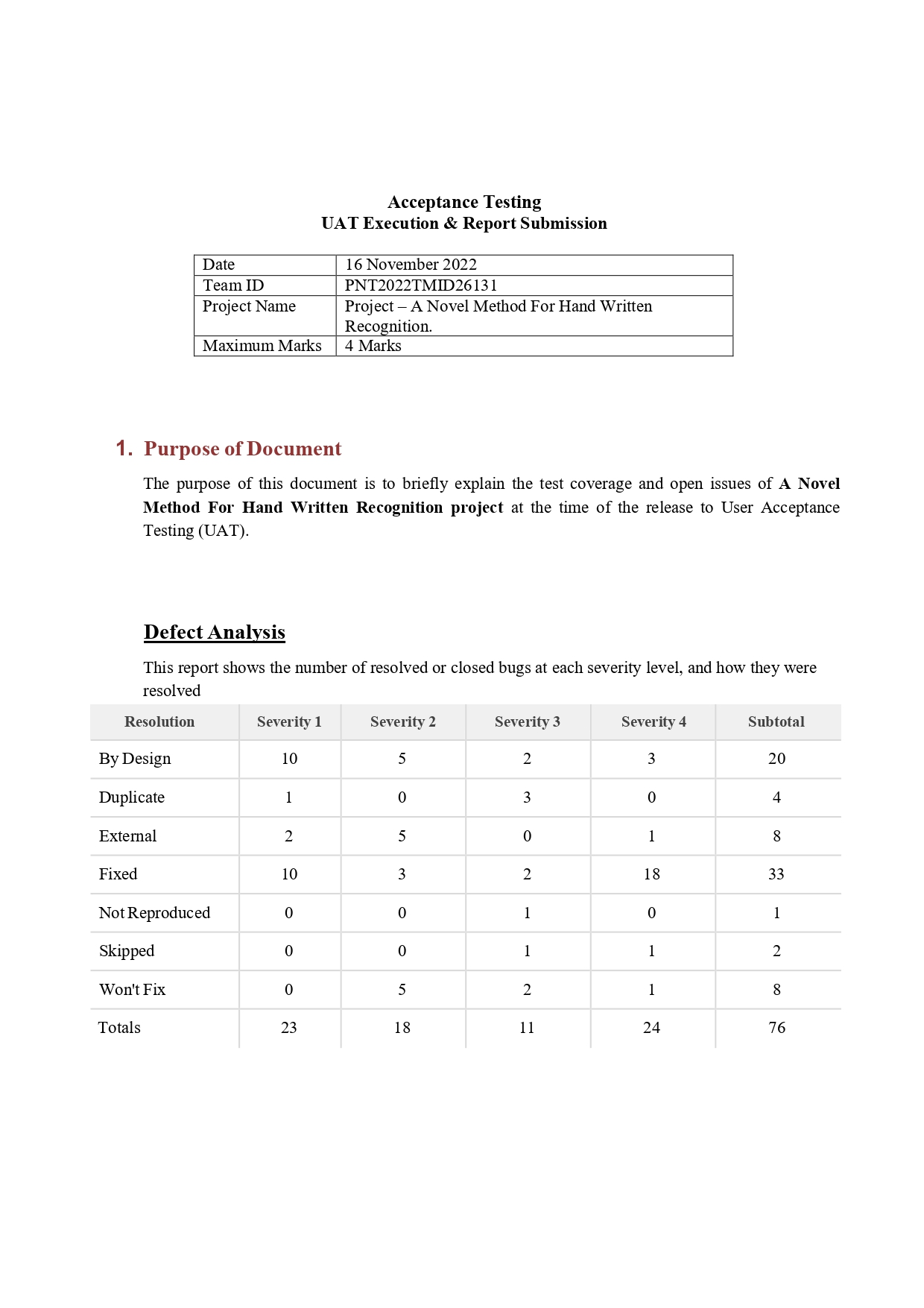
**TESTING**

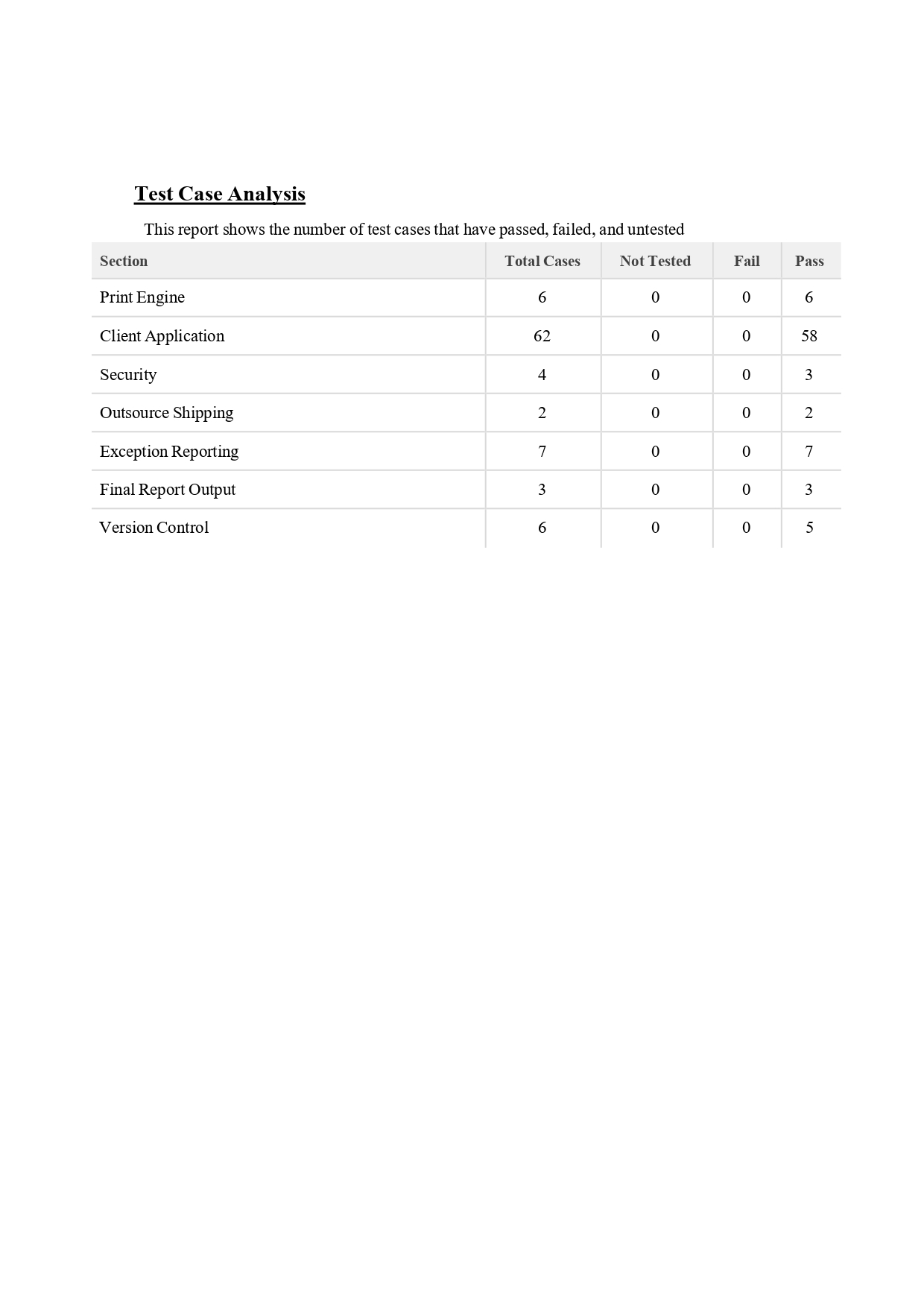
**TEST CASES**

* Check the target and the output predicted value every time while training the dataset and set the number of epochs corresponding to the error.
* Model check with checksum value removal and biased value removal after setting the weight value by the neural network on its set and set the dataset into the unbiased.
* Access the webcam and capture the image using python javascript and google colab display.
* Check the parameter of probability of the figure after normalisation and noise removal and scaling.
* Got highest predicted value? match the figure with the To make things shorter what has been done in this model- 1) Loaded self created data and MNIST dataset into the model
* Set the numpy array system to take input the kernel along with the data.
* Create the neural network model setting the input layer and the number of hidden layers and the output layers along with the activation functions used in different layers.
* Set the probabilistic statistical value into the biased dataset into unbiased.
* detected from the neural

**USER ACCEPTANCE TESTING**

This is a type of system or software testing where a system has been tested for availability. The purpose of this test is to check the business requirements and assess whether it will be accepted for delivery. In this part ADRIAN of pyrimagsearch has been referred to, who worked with the same platform and to check this project accepted by the delivery partner or not

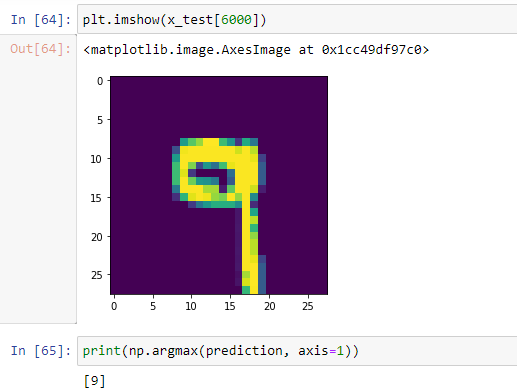




RESULTS

**PERFORMANCE METRICES**

There is always opportunity for improvement in your methods because machine learning is a topic that is continually developing; there will always be a fresh new idea that solves the same problem better. Three models were used to test the application: Multi-Layer Perceptron (MLP), Convolution Neural Network, and (CNN). We obtain a different classifier accuracy with each model, indicating which is superior.



## DATASET USED:

The MNIST collection of handwritten digits served as the dataset. It has a test set of 10. The digits have been centred in a fixed-size image and size-normalized. The photos are 28\*28 pixels in size. It is a useful database for those who want to practise new skills and pattern recognition algorithms on actual data with the least amount of pre-processing and formatting work.

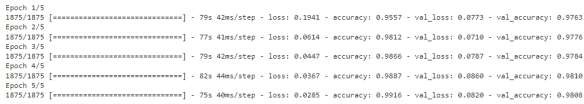


## ANALYSIS OF THE RESULTS:

System Analysis and Design is the term used in business to describe the process of analysing a business issue with the goal of improving it through improved practises and methodologies.

Designing organisations, enhancing performance, and attaining goals for profitability and expansion are all related to system analysis and design. The focus is on systems in action, the connections

between subsystems, and their role in achieving a common objective.



System analysis includes examining a system to determine its effectiveness, the necessary adjustments, and the output's quality. Organizations are intricate systems made up of connected and interdependent subsystems

**ADVANTAGES & DISADVANTAGES**

* Fire may also play a role in recycling nutrients from the ground-layer vegetation and litter to the overstorey trees, thereby counteracting the infertile substrates and arrested decay
* Of course, these forests don't just remove trees; they kill and displace wildlife, alter water cycles and soil fertility, and endanger the lives and livelihoods of local communities. They also can rage out of control

**CONCLUSION**

The proposed system for forest fire detection using wireless sensor networks and machine learning was found to be an effective method for fire detection in forests that provides more accurate results

**FUTURE SCOPE**

This ﬁre alert system is power efﬁcient, low cost

and low maintenance, and the equipment is durable

and reliable. In future, we can install a wind sensor to

the system which helps to determine the direction of

the ﬁre and the rate at which it will spread. Along with

this we can implement an automatic ﬁre extinguisher

system. As soon as a sensor detects ﬁre, extinguisher

gets activated. GPS module can be added to the

nodes to get the exact location of ﬁre or smoke. By

adding IOT, the data can be sent to cloud databases

for storage and prediction purposes. In addition to

these, many other sensors can also be implemented

**import** keras

**import** os**,** types

**import** pandas **as** pd

**from** botocore.client **import** Config

**import** ibm\_boto3

**def** \_\_iter\_\_(self): **return** 0

*# @hidden\_cell*

*# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.*

*# You might want to remove those credentials before you share the notebook.*

cos\_client **=** ibm\_boto3**.**client(service\_name**=**'s3',

ibm\_api\_key\_id**=**'QVsKoX6yNRLn8Qv\_RWlO4N-PlH8ddMdZyPPjLiIW1oz6',

ibm\_auth\_endpoint**=**"https://iam.cloud.ibm.com/oidc/token",

config**=**Config(signature\_version**=**'oauth'),

endpoint\_url**=**'https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket **=** 'imageclassification-donotdelete-pr-ncc5wgfie3fmsd'

object\_key **=** 'Dataset.zip'

streaming\_body\_2 **=** cos\_client**.**get\_object(Bucket**=**bucket, Key**=**object\_key)['Body']

*# Your data file was loaded into a botocore.response.StreamingBody object.*

*# Please read the documentation of ibm\_boto3 and pandas to learn more about the possibilities to load the data.*

*# ibm\_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/*

*# pandas documentation: http://pandas.pydata.org/*

**from** io **import** BytesIO

**import** zipfile

unzip**=**zipfile**.**ZipFile(BytesIO(streaming\_body\_2**.**read()),'r')

file\_paths**=**unzip**.**namelist()

**for** path **in** file\_paths:

unzip**.**extract(path)

pwd

Out[91]:

'/home/wsuser/work'

In [92]:

**import** os

filenames**=**os**.**listdir('/home/wsuser/work/Dataset/train\_set')

In [93]:

**!**pip install libgl1-mesa-dev

**import** tensorflow **as** tf

**import** numpy **as** np

**from** tensorflow **import** keras

**import** os

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

**from** tensorflow.keras.preprocessing **import** image

ERROR: Could not find a version that satisfies the requirement libgl1-mesa-dev (from versions: none)

ERROR: No matching distribution found for libgl1-mesa-dev

**Define the parameters/arguments for ImageDataGenerator class**

In [94]:

train**=**ImageDataGenerator(rescale**=**1.**/**255,

shear\_range**=**0.2,

rotation\_range**=**180,

zoom\_range**=**0.2,

horizontal\_flip**=True**)

train **=** ImageDataGenerator(rescale**=**1**/**255)

test **=** ImageDataGenerator(rescale**=**1**/**255)

**Applying ImageDataGenerator functionality to trainset**

In [95]:

x\_train **=** train**.**flow\_from\_directory("/home/wsuser/work/Dataset/train\_set",

target\_size**=**(64,64),

batch\_size **=** 32,

class\_mode **=** 'binary' )

Found 436 images belonging to 2 classes.

**Applying ImageDataGenerator functionality to testset**

In [96]:

x\_test **=** test**.**flow\_from\_directory("/home/wsuser/work/Dataset/test\_set",

target\_size**=**(64,64),

batch\_size **=** 32,

class\_mode **=** 'binary' )

Found 121 images belonging to 2 classes.

In [97]:

x\_test**.**class\_indices

Out[97]:

{'forest': 0, 'with fire': 1}

**Import model building libraries**

In [98]:

*#to define linear initialisation import sequential*

**from** keras.models **import** Sequential

*#to add layer import Dense*

**from** keras.layers **import** Dense

*#to create convolution kernel import convolution2D*

**from** keras.layers **import** Convolution2D

*#import Maxpooling layer*

**from** keras.layers **import** MaxPooling2D

*#import flatten layer*

**from** keras.layers **import** Flatten

**import** warnings

warnings**.**filterwarnings('ignore')

**Initializing the model**

In [99]:

model **=**Sequential()

**Add CNN Layer**

In [100]:

model**.**add(Convolution2D(32,(3,3),input\_shape**=**(64,64,3),activation**=**'relu'))

*#add maxpooling layers*

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

*#add faltten layer*

model**.**add(Flatten())

**Add Hidden Layer**

In [101]:

*#add hidden layers*

model**.**add(Dense(150,activation**=**'relu'))

*#add output layer*

model**.**add(Dense(1,activation**=**'sigmoid'))

**Configure the learning process**

In [102]:

model**.**compile(loss **=** 'binary\_crossentropy',

optimizer **=** "adam",

metrics **=** ["accuracy"])

**Train the model**

In [103]:

model**.**fit\_generator(x\_train,steps\_per\_epoch**=**14,epochs**=**5,validation\_data**=**x\_test,validation\_steps**=**20)

Epoch 1/5

14/14 [==============================] - ETA: 0s - loss: 0.7689 - accuracy: 0.6628WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps\_per\_epoch \* epochs` batches (in this case, 20 batches). You may need to use the repeat() function when building your dataset.

14/14 [==============================] - 21s 1s/step - loss: 0.7689 - accuracy: 0.6628 - val\_loss: 0.3151 - val\_accuracy: 0.867

Epoch 2/5

14/14 [==============================] - 15s 1s/step - loss: 0.2759 - accuracy: 0.8853

Epoch 3/5

14/14 [==============================] - 14s 1s/step - loss: 0.1619 - accuracy: 0.9472

Epoch 4/5

14/14 [==============================] - 14s 1s/step - loss: 0.1192 - accuracy: 0.9541

Epoch 5/5

14/14 [==============================] - 14s 1s/step - loss: 0.1087 - accuracy: 0.9633

Out[103]:

**Save The Model**

In [104]:

model**.**save("/home/wsuser/work/archive(1)/forest1.h5")

**Predictions**

In [105]:

predictions **=** model**.**predict(x\_test)

predictions **=** np**.**round(predictions)

In [106]:

predictions

Out[106]:

array([[1.],

[1.],

[1.],

[0.],

[1.],

[1.],

[1.],

[0.],

[1.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[1.],

[1.],

[0.],

[1.],

[1.],

[1.],

[1.],

[1.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[1.]], dtype=float32)

In [107]:

print(len(predictions))

121

In [108]:

*#import load\_model from keras.model*

**from** keras.models **import** load\_model

*#import image class from keras*

**import** tensorflow **as** tf

**from** tensorflow.keras.preprocessing **import** image

*#import numpy*

**import** numpy **as** np

*#import cv2*

In [109]:

*#load the saved model*

model **=** load\_model("/home/wsuser/work/archive(1)/forest1.h5")

In [110]:

**def** predictImage(filename):

img1 **=** image**.**load\_img(filename,target\_size**=**(64,64))

Y **=** image**.**img\_to\_array(img1)

X **=** np**.**expand\_dims(Y,axis**=**0)

val **=** model**.**predict(X)

print(val)

**if** val **==** 1:

print(" fire")

**elif** val **==** 0:

print("no fire")

In [111]:

predictImage("/home/wsuser/work/Dataset/test\_set/with fire/19464620\_401.jpg")

WARNING:tensorflow:5 out of the last 12 calls to .predict\_function at 0x7f984fe019d0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental\_relax\_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling\_retracing and https://www.tensorflow.org/api\_docs/python/tf/function for more details.

[[1.]]

fire

**OpenCV For Video Processing**

In [75]:

pip install twilio

Collecting twilio

Downloading twilio-7.15.2-py2.py3-none-any.whl (1.4 MB)

|████████████████████████████████| 1.4 MB 16.7 MB/s eta 0:00:01

Requirement already satisfied: requests>=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from twilio) (2.26.0)

Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from twilio) (2.4.0)

Requirement already satisfied: pytz in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from twilio) (2021.3)

Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests>=2.0.0->twilio) (3.3)

Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests>=2.0.0->twilio) (2022.9.24)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests>=2.0.0->twilio) (1.26.7)

Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests>=2.0.0->twilio) (2.0.4)

Installing collected packages: twilio

Successfully installed twilio-7.15.2

Note: you may need to restart the kernel to use updated packages.

In [76]:

pip install playsound

Collecting playsound

Downloading playsound-1.3.0.tar.gz (7.7 kB)

Building wheels for collected packages: playsound

Building wheel for playsound (setup.py) ... done

Created wheel for playsound: filename=playsound-1.3.0-py3-none-any.whl size=7037 sha256=7a14e5d7212967bf1952d7558b36640342f7d1b687b5aa9dc5b0e950f2e73b31

Stored in directory: /tmp/wsuser/.cache/pip/wheels/ba/39/54/c8f7ff9a88a644d3c58b4dec802d90b79a2e0fb2a6b884bf82

Successfully built playsound

Installing collected packages: playsound

Successfully installed playsound-1.3.0

*#import opencv librariy*

*#import cv2*

*#import numpy*

**import** numpy **as** np

*#import image function from keras*

**from** keras.preprocessing **import** image

*#import load\_model from keras*

**from** keras.models **import** load\_model

*#import client from twilio API*

**from** twilio.rest **import** Client

*#imort playsound package*

**from** playsound **import** playsound

In [113]:

*#load the saved model*

model **=** load\_model(r'/home/wsuser/work/archive(1)/forest1.h5')

*#define the features*

name **=** ['forest','with forest']

**Creating An Account In Twilio Service**

In [114]:

account\_sid**=**'ACfb4e6d0e7b0d25def63044919f1b96e3'

auth\_token**=**'f9ae4fc4a617a527da8672e97eefb2d8'

client**=**Client(account\_sid,auth\_token)

message**=**client**.**messages \

**.**create(

body**=**'Forest Fire is detected, stay alert',

from\_**=**'+1 302 248 4366',

to**=**'+91 99400 12164'

)

print(message**.**sid)

SMfa58e2bf0ede24f765cd7bc345ca795d

**Sending Alert Message**

In [81]:

pip install pygobject

Collecting pygobject

Downloading PyGObject-3.42.2.tar.gz (719 kB)

|████████████████████████████████| 719 kB 9.3 MB/s eta 0:00:01

Installing build dependencies ... error

.

Downloading PyGObject-3.42.1.tar.gz (718 kB)

|████████████████████████████████| 718 kB 51.7 MB/s eta 0:00:01

Installing build dependencies ... error

ERROR: Command errored out with exit status 1:

command: /opt/conda/envs/Python-3.9/bin/python /tmp/wsuser/pip-standalone-pip-nxupggha/\_\_env\_pip\_\_.zip/pip install --ignore-installed --no-user --prefix /tmp/wsuser/pip-build-env-q\_zdbpt2/overlay --no-warn-script-location --no-binary :none: --only-binary :none: -i https://pypi.org/simple -- setuptools wheel pycairo

cwd: None

Complete output (38 lines):

Collecting setuptools

Using cached setuptools-65.5.1-py3-none-any.whl (1.2 MB)

Collecting wheel

Using cached wheel-0.38.4-py3-none-any.whl (36 kB)

Collecting pycairo

Using cached pycairo-1.21.0.tar.gz (340 kB)

Installing build dependencies: started

Installing build dependencies: finished with status 'done'

Getting requirements to build wheel: started

Getting requirements to build wheel: finished with status 'done'

Preparing wheel metadata: started

Preparing wheel metadata: finished with status 'done'

Building wheels for collected packages: pycairo

Building wheel for pycairo (PEP 517): started

Building wheel for pycairo (PEP 517): finished with status 'error'

**def** message(val):

**if** val**==**1:

**from** twilio.rest **import** Client

print('Forest fire')

account\_sid**=**'ACfb4e6d0e7b0d25def63044919f1b96e3'

auth\_token**=**'f9ae4fc4a617a527da8672e97eefb2d8'

client**=**Client(account\_sid,auth\_token)

message**=**client**.**messages \

**.**create(

body**=**'forest fire is detected, stay alert',

*#use twilio free number*

from\_**=**'+1 302 248 4366',

*#to number*

to**=**'+91 99400 12164')

print(message**.**sid)

print("Fire detected")

print("SMS Sent!")

**elif** val**==**0:

print('No Fire')

In [116]:

**from** matplotlib **import** pyplot **as** plt

*#import load model from keras.model*

**from** keras.models **import** load\_model

*#import image from keras*

**from** tensorflow.keras.preprocessing **import** image

img1 **=** image**.**load\_img('/home/wsuser/work/Dataset/test\_set/with fire/Wild\_fires.jpg',target\_size**=**(64,64))

Y **=** image**.**img\_to\_array(img1)

x **=** np**.**expand\_dims(Y,axis**=**0)

val **=** model**.**predict(x)

plt**.**imshow(img1)

plt**.**show()

WARNING:tensorflow:6 out of the last 13 calls to .predict\_function at 0x7f984fe36820> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental\_relax\_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling\_retracing and https://www.tensorflow.org/api\_docs/python/tf/function for more details.

img2 **=** image**.**load\_img('/home/wsuser/work/Dataset/test\_set/forest/1200px\_Mountainarea.jpg',target\_size**=**(64,64))

Y **=** image**.**img\_to\_array(img2)

x **=** np**.**expand\_dims(Y,axis**=**0)

val **=** model**.**predict(x)

plt**.**imshow(img2)

plt**.**show()

message(val)

No Fire

In [119]:

**from** ibm\_watson\_machine\_learning **import** APIClient

wml\_credentials**=**{"url":"https://us-south.ml.cloud.ibm.com","apikey":"TFXoHzN3M76f8UM68mdo\_MshGtF2Dk1H56fJ67oDagbV"}

client**=**APIClient(wml\_credentials)

In [120]:

**def** guid\_from\_space\_name(client,space\_name):

space**=**client**.**spaces**.**get\_details()

**return**(next(item **for** item **in** space['resources']**if** item['entity']["name"]**==**space\_name)['metadata']['id'])

space\_uid**=**guid\_from\_space\_name(client,'imageclassification')

print("Space UID= "**+**space\_uid)

Space UID= 1f4924ef-b4eb-471f-a528-7b437fc3efb9

In [122]:

client**.**set**.**default\_space(space\_uid)

Out[122]:

'SUCCESS'

In [123]:

client**.**software\_specifications**.**list()

----------------------------- ------------------------------------ ----

NAME ASSET\_ID TYPE

default\_py3.6 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base

kernel-spark3.2-scala2.12 020d69ce-7ac1-5e68-ac1a-31189867356a base

pytorch-onnx\_1.3-py3.7-edt 069ea134-3346-5748-b513-49120e15d288 base

scikit-learn\_0.20-py3.6 09c5a1d0-9c1e-4473-a344-eb7b665ff687 base

spark-mllib\_3.0-scala\_2.12 09f4cff0-90a7-5899-b9ed-1ef348aebdee base

pytorch-onnx\_rt22.1-py3.9 0b848dd4-e681-5599-be41-b5f6fccc6471 base

ai-function\_0.1-py3.6 0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda base

shiny-r3.6 0e6e79df-875e-4f24-8ae9-62dcc2148306 base

tensorflow\_2.4-py3.7-horovod 1092590a-307d-563d-9b62-4eb7d64b3f22 base

pytorch\_1.1-py3.6 10ac12d6-6b30-4ccd-8392-3e922c096a92 base

tensorflow\_1.15-py3.6-ddl 111e41b3-de2d-5422-a4d6-bf776828c4b7 base

autoai-kb\_rt22.2-py3.10 125b6d9a-5b1f-5e8d-972a-b251688ccf40 base

runtime-22.1-py3.9 12b83a17-24d8-5082-900f-0ab31fbfd3cb base

scikit-learn\_0.22-py3.6 154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base

default\_r3.6 1b70aec3-ab34-4b87-8aa0-a4a3c8296a36 base

pytorch-onnx\_1.3-py3.6 1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base

kernel-spark3.3-r3.6 1c9e5454-f216-59dd-a20e-474a5cdf5988 base

pytorch-onnx\_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f base

tensorflow\_2.1-py3.6 1eb25b84-d6ed-5dde-b6a5-3fbdf1665666 base

spark-mllib\_3.2 20047f72-0a98-58c7-9ff5-a77b012eb8f5 base

tensorflow\_2.4-py3.8-horovod 217c16f6-178f-56bf-824a-b19f20564c49 base

runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-a1b0-da66306ce658 base

do\_py3.8 295addb5-9ef9-547e-9bf4-92ae3563e720 base

autoai-ts\_3.8-py3.8 2aa0c932-798f-5ae9-abd6-15e0c2402fb5 base

tensorflow\_1.15-py3.6 2b73a275-7cbf-420b-a912-eae7f436e0bc base

kernel-spark3.3-py3.9 2b7961e2-e3b1-5a8c-a491-482c8368839a base

pytorch\_1.2-py3.6 2c8ef57d-2687-4b7d-acce-01f94976dac1 base

spark-mllib\_2.3 2e51f700-bca0-4b0d-88dc-5c6791338875 base

pytorch-onnx\_1.1-py3.6-edt 32983cea-3f32-4400-8965-dde874a8d67e base

spark-mllib\_3.0-py37 36507ebe-8770-55ba-ab2a-eafe787600e9 base

spark-mllib\_2.4 390d21f8-e58b-4fac-9c55-d7ceda621326 base

autoai-ts\_rt22.2-py3.10 396b2e83-0953-5b86-9a55-7ce1628a406f base

xgboost\_0.82-py3.6 39e31acd-5f30-41dc-ae44-60233c80306e base

pytorch-onnx\_1.2-py3.6-edt 40589d0e-7019-4e28-8daa-fb03b6f4fe12 base

pytorch-onnx\_rt22.2-py3.10 40e73f55-783a-5535-b3fa-0c8b94291431 base

default\_r36py38 41c247d3-45f8-5a71-b065-8580229facf0 base

autoai-ts\_rt22.1-py3.9 4269d26e-07ba-5d40-8f66-2d495b0c71f7 base

autoai-obm\_3.0 42b92e18-d9ab-567f-988a-4240ba1ed5f7 base

pmml-3.0\_4.3 493bcb95-16f1-5bc5-bee8-81b8af80e9c7 base

spark-mllib\_2.4-r\_3.6 49403dff-92e9-4c87-a3d7-a42d0021c095 base

xgboost\_0.90-py3.6 4ff8d6c2-1343-4c18-85e1-689c965304d3 base

pytorch-onnx\_1.1-py3.6 50f95b2a-bc16-43bb-bc94-b0bed208c60b base

autoai-ts\_3.9-py3.8 52c57136-80fa-572e-8728-a5e7cbb42cde base

spark-mllib\_2.4-scala\_2.11 55a70f99-7320-4be5-9fb9-9edb5a443af5 base

spark-mllib\_3.0 5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9 base

autoai-obm\_2.0 5c2e37fa-80b8-5e77-840f-d912469614ee base

spss-modeler\_18.1 5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b base

cuda-py3.8 5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e base

autoai-kb\_3.1-py3.7 632d4b22-10aa-5180-88f0-f52dfb6444d7 base

pytorch-onnx\_1.7-py3.8 634d3cdc-b562-5bf9-a2d4-ea90a478456b base

----------------------------- ------------------------------------ ----

Note: Only first 50 records were displayed. To display more use 'limit' parameter.

In [126]:

software\_spec\_uid**=**client**.**software\_specifications**.**get\_uid\_by\_name("tensorflow\_1.15-py3.6")

software\_spec\_uid

Out[126]:

'2b73a275-7cbf-420b-a912-eae7f436e0bc'

In [128]:

keras

Out[128]: