

## **PROJECT REPORT**

Project Name: **EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES.**

Team id: **PNT2022TMID07050**

Team members: **DEVI SRAVANTI GUBBALA -TEAM LEAD.**

**AKSHARA YUVARAJ**

**DIVYASRI.P**

**ESTHER JANET.S**

### **1. INTRODUCTION**

#### **1.1 Project overview**

Wildfire, also called forest fire, bush or vegetation fire, can be described as any uncontrolled and non-prescribed combustion or burning of plants in a natural setting such as a forest, grassland, brush land or tundra, which consumes the natural fuels and spreads based on environmental conditions (e.g., wind, topography). Forest fires are a major environmental issue, creating economic and ecological damage while endangering human lives. There are typically about 100,000 wildfires in the United States every year. Over 9 million acres of land have been destroyed due to treacherous wildfires. It is difficult to predict and detect Forest Fire in a sparsely populated forest area and it is more difficult if the prediction is done using ground- based methods like Camera or Video-Based approach. Satellites can be an important source of data prior and also during the Fire due to its reliability and efficiency. The various real-time forest fire detection and prediction approaches, with the goal of informing the local fire authorities. This is a huge problem which needs to be tackled and thus through this project we provide a way to tackle the issue.

#### **1.2 Purpose**

The purpose of the project is to detect the forest fire earlier.

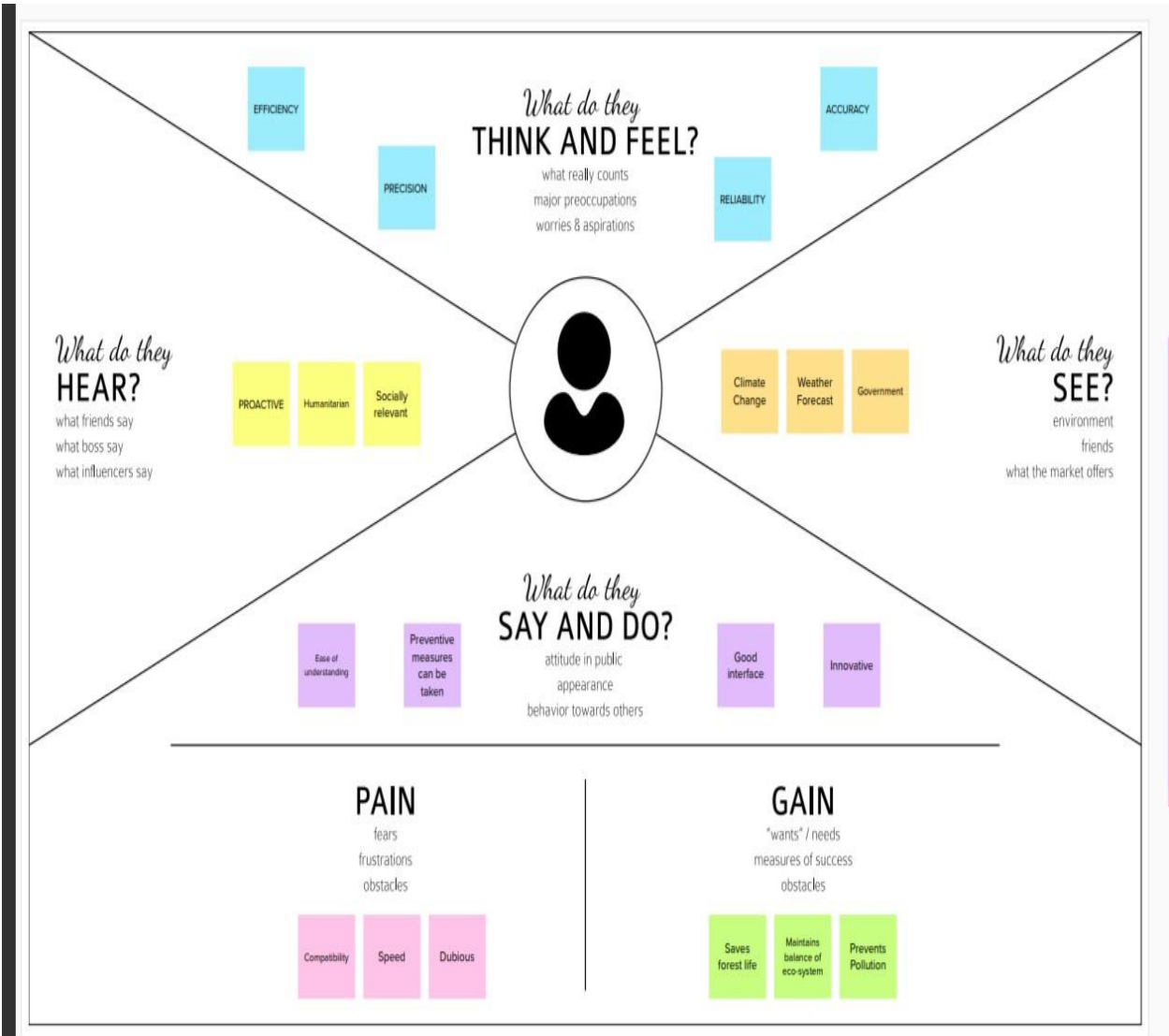
## 2. LITERATURE SURVEY

### 2.1 Reference


S. NO	TITL E	AUTHO R	YEAR
1.	Image Processing for Forest Fire Detection.	Priyadharshini	2016
2.	Forest fire prediction and detection system.	Faroudja Abid	2020
3.	systematic approaches in managing forest fires .	AdityaDhall	2020

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy map



## 3.2 Ideation & Brainstorming



### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

**Before you collaborate**  
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

- A Team gathering**  
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- B Set the goal**  
Think about the problem you'll be focusing on solving in the brainstorming session.
- C Learn how to use the facilitation tools**  
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →



3

### Group ideas

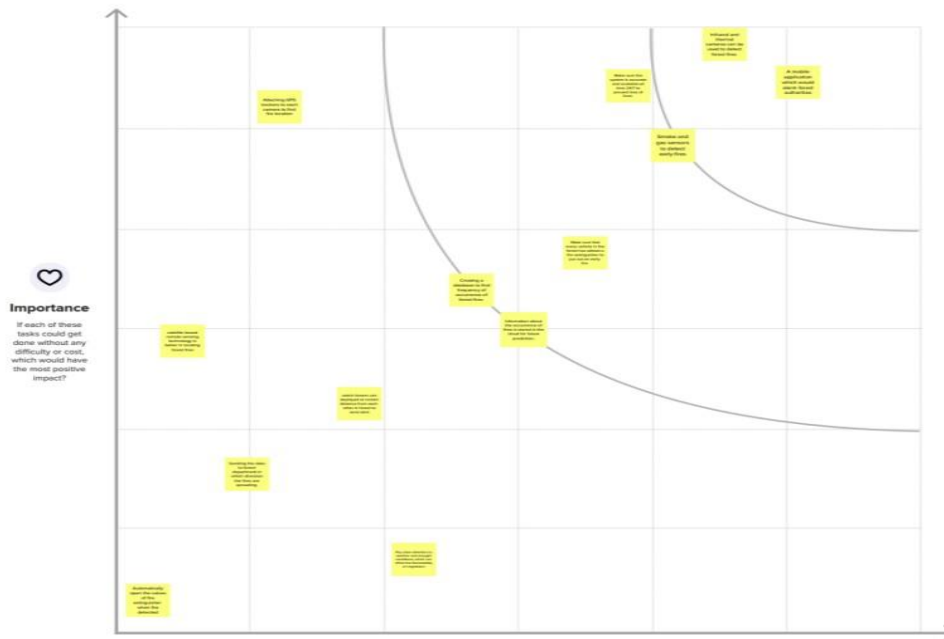
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.



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.



### 3.3 Proposed solution

S.NO:	PARAMETERS	REPRESENTATION
1.	<b>Problem Statement (Description of an issue to be addressed)</b>	<ul style="list-style-type: none"> <li>Fire was one of the first and greatest invention of man. But these days due to global warming and climate change, fires have become very violent and destructive.</li> <li>Forest fires are one such evil looming the Earth destroying all the flora and fauna with the devastating fumes and flares it carries with itself.</li> <li>Recent forest fires in California is an evident example of the intensity of the issue and the immediate action that needs to be taken.</li> </ul>
2.	<b>Plan of Design and Execution</b>	<ul style="list-style-type: none"> <li>The propose a platform that uses Unmanned Aerial Vehicles (UAVs), which constantly patrol over potentially threatened by fire areas.</li> <li>The UAVs also utilize the benefits from Artificial Intelligence(AI) and are equipped with on-board processing capabilities.</li> <li>This allows them to use computer vision methods for recognition and detection of smoke or fire, based on the still images or the video input from the drone cameras.</li> <li>The system is designed for monitor the causing factors of forest fires such as temperature, humidity , air pressure level,oxygen and Carbon dioxide on the surface of air.</li> <li>The user interacts with a web camera to read the video.</li> </ul>

		<ul style="list-style-type: none"> <li>Once the input image from the video frame is sent to the model, if the fire is detected, it is showcased on the console, and alerting sound will be generated and an alert message will be sent to the Authorities.</li> <li>We classify images using a Convolutional Neural Network and use other open CV tools.</li> </ul>
3.	Peculiarity/ Novelty	<ul style="list-style-type: none"> <li>Makes use of real time monitoring and allows pre-cursors to potential issues (such as corrosion) to be flagged up and immediately be addressed before major issues occur.</li> </ul>
4.	Social Outlook / Customer Friendly	<ul style="list-style-type: none"> <li>Will warn the customers before any fire outbreak.</li> <li>Prevents any potential devastation and issues precautions.</li> <li>Protects the flora and fauna from any unfortunate accidents.</li> <li>Saves forest and human life prevents desertification.</li> </ul>
5.	Business Model	<ul style="list-style-type: none"> <li>Focuses more on sensor probes, wireless sensor networks and machine learning which makes the deployment more easier.</li> </ul>
6.	Feasibility of Solution	<ul style="list-style-type: none"> <li>Cost effective</li> <li>More performance measure</li> <li>Economical</li> <li>Accurate</li> <li>Effective</li> <li>Reliable</li> <li>Socially intact</li> </ul>

### 3.4 Problem solution fit

Proposed solution fit.		
<b>1.Customer Segment</b>  -To adopt a new technology.  -For officers who works in forestry department.	<b>2.Problems/Pains</b>  -Deterioration of air quality,loss of property ,resources and animal.  -Sometimes devices may malfunction.	<b>3.Triggers and emotions</b>  -To get prior information of forest fire  -It would proceed the misinformation or late details about the forest fire.
<b>4.Customer Limitations</b>  -Should have knowledge about the devices.  -feature loaded device.	<b>5.Problem Root/Cause</b>  -The forest fire starts from natural cause such as lightning.  -Less humidity, high temperature may also cause forest fire	<b>6.Your Solutions</b>  -We train the model with required algorithm like CNN,images of smoke,fire  -Classifying the intensity of the flame using sensors.
<b>7.Available Solution</b>  -satellite based system give high resolution image but it provides image of entire earth for every two days,that is long time for fire scanning.	<b>8.Channels of Behavior</b>  -They should monitor and checj the device functionalitiy, to alert the smokejumpers.  -They should be present at the fire spot with extinguisher and with all saftey precautions.	<b>9.Behavior</b>  -It emits a large amount of CO2 which may lead to increase in global warming.  -It measures the intensity,light,colour and defines according to its behaviour.

## 4. REQUIREMENT ANALYSIS

### 4.1 Functional requirement

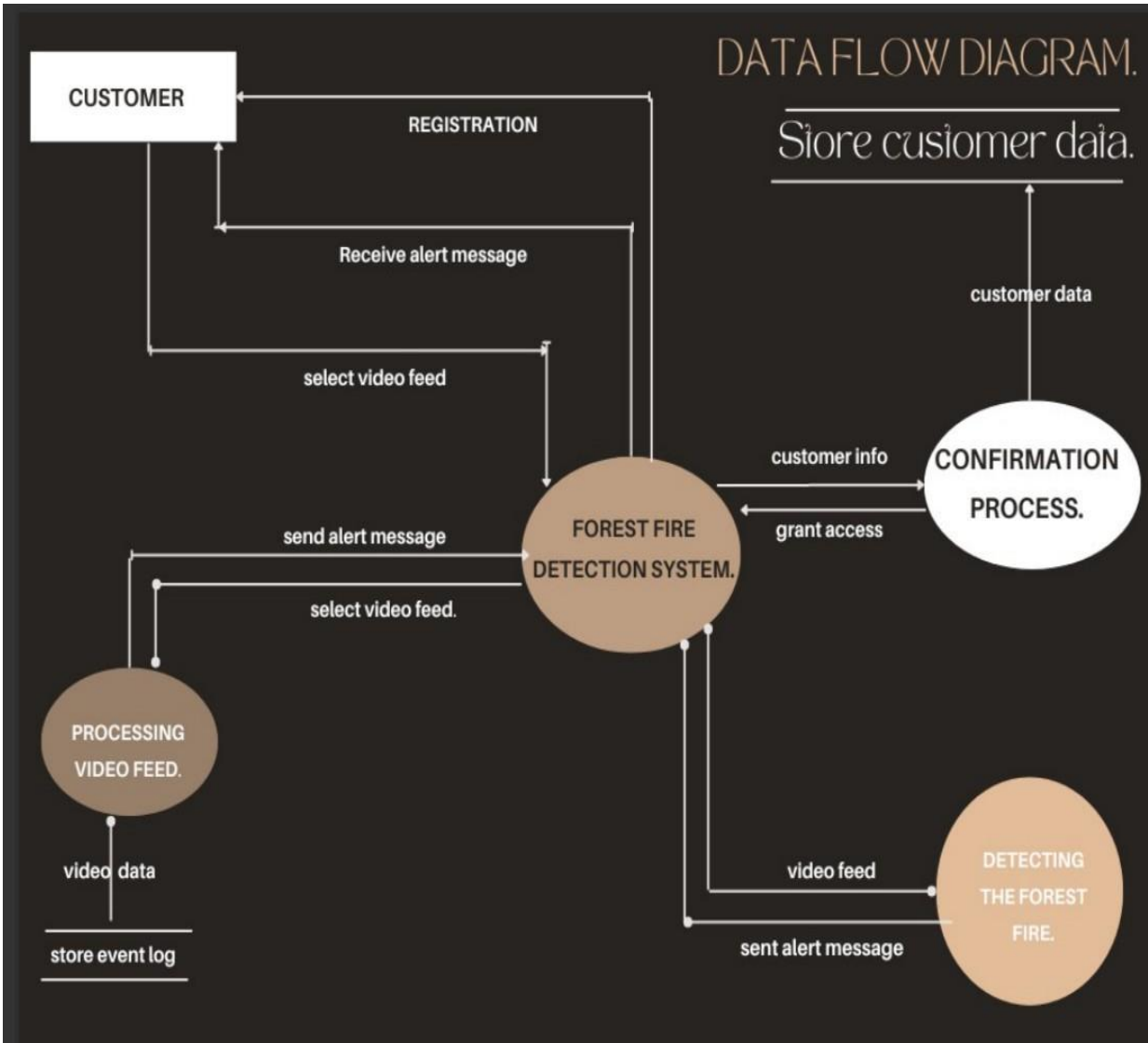
FR. NO.	Functional Requirement	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through wildfire portal.
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Data Prediction	Scientists create computer models to predict wildfire potential under a range of potential climate futures. Using different projections of temperature and downfall, scientists predict where and when wildfires are likely to occur

### 4.2 Non-Functional requirement

FR. NO.	Non-Functional Requirement	Description
NFR-1	Usability	Many methods have been proposed to detect forest fires, such as camera-based systems, WSN-based systems, and machine learning coating-based systems, with both positive and negative aspects and performance figures of detection.
NFR-2	Protection	We have designed this project to secure the forest from wild fires.
NFR-3	Performance	In the event of a fire, the primary objective of using drones is to gather situational consciousness, which can be used to direct the efforts of the firefighters in locating and controlling hot spots. Just like urban fires, forest fires to require monitoring so that firefighters know what they are dealing with.

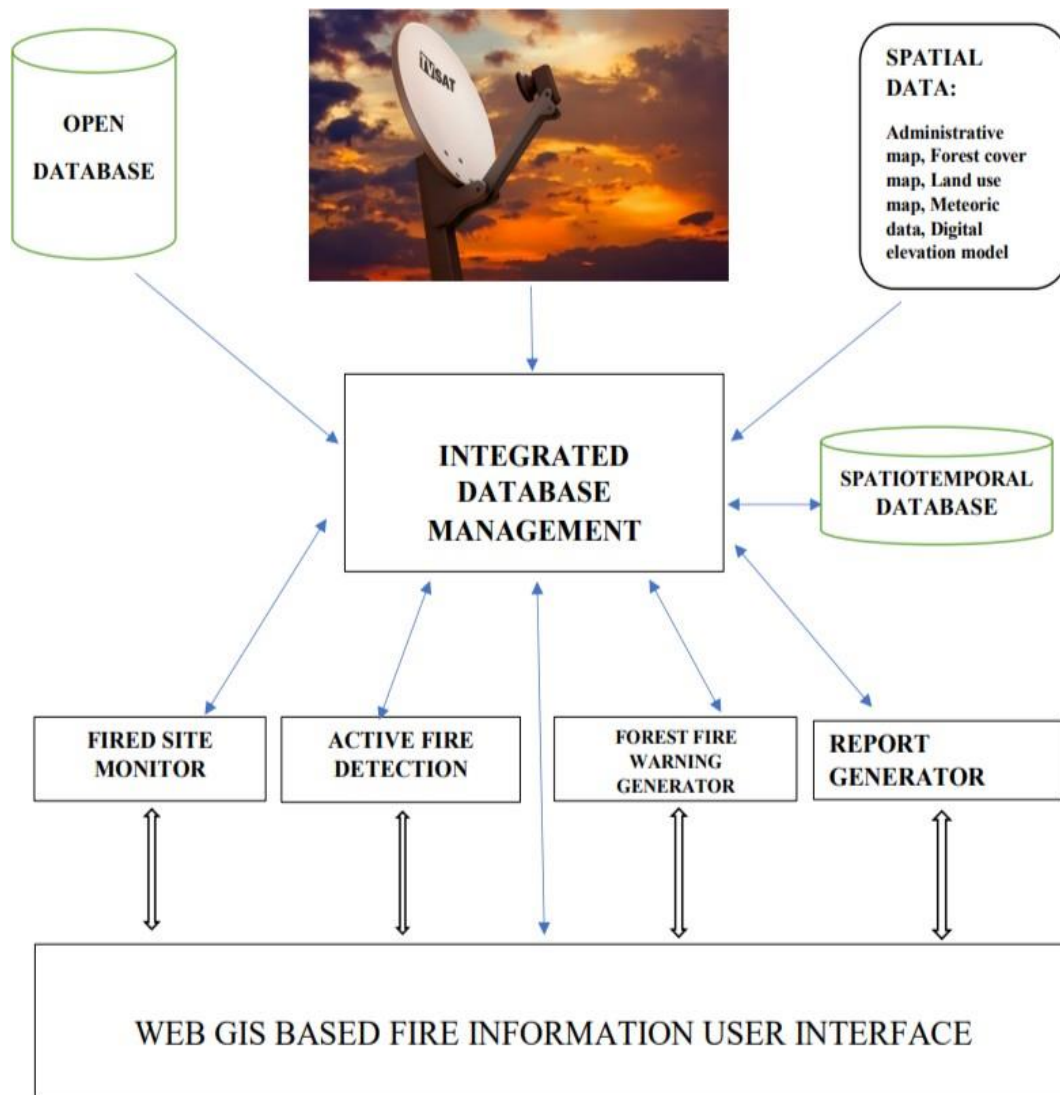
PROJECT DESIGN

4.3 Data Flow Diagrams





#### 4.4 Solution Architecture



## 5. PROJECT PLANNING & SCHEDULING

### 5.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Image Processing	USN-1	Processing the image to find the fire is detected or not.	1	Medium	1.Devi Sravanti 2.Esther 3.Divya Sri 4.Akshar a

Sprint-1		USN-2	The output would have to give high accuracy.	2	High	1.Devi Sravanti 2.Esther 3.Divya Sri 4.Akshar a
Sprint-2	Video Processing	USN-3	The drone videos will be split into frames to detect the fire.	3	High	1.Devi Sravanti 2.Esther 3.Divya Sri 4.Akshar a
Sprint-3	Alerting	USN-4	After the fire is detected the alert message has to be sent.	2	High	1.Devi Sravanti 2.Esther 3.Divya Sri 4.Akshar a

Sprint-4	Location tracking	USN-5	The exact location of the drone will be predicted and sent along with the alert message.	2	High	1.Devi Sravanti 2.Esther 3.Divya Sri 4.Akshar a
----------	-------------------	-------	--	---	------	--

### 5.2 Sprint delivery schedule

Project Tracker, Velocity & Burndown Chart:

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	25 Oct 2022	30 Oct 2022	30	30 Oct 2022
Sprint-2	20	6 Days	1 Nov 2022	06 Nov 2022	20	06 Nov 2022
Sprint-3	20	6 Days	08 Nov 2022	13 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	15 Nov 2022	20 Nov 2022	20	20 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's now calculate the team's average velocity (AV) per iteration unit (story points per day)

$$\begin{aligned} AV &= \text{Sprint duration} / \text{Velocity} \\ &= 20 / 6 = 3 \end{aligned}$$

### 5.3 SPRINT-1 (COLLECTION OF DATATSET)

```
In [1]: import tensorflow as tf
import numpy as np
from tensorflow import keras
import os
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
```

```
In [2]: train = ImageDataGenerator(rescale=1/255)
test = ImageDataGenerator(rescale=1/255)

train_dataset = train.flow_from_directory("/content/drive/MyDrive/train_set",
                                         target_size=(150,150),
                                         batch_size = 32,
                                         class_mode = 'binary')

test_dataset = test.flow_from_directory("/content/drive/MyDrive/test_set",
                                        target_size=(150,150),
                                        batch_size = 32,
                                        class_mode = 'binary')
```

```
Found 442 images belonging to 2 classes.
Found 121 images belonging to 2 classes.
```

```
In [3]: test_dataset.class_indices
```

```
Out[3]: {'forest': 0, 'with fire': 1}
```

## 5.4 SPRINT-2 (MODEL BUILDING AND CLASSIFICATION)

[illegible]

```
batch_size = 32,  
class_mode = 'binary')
```

```
test_dataset = test.flow_from_directory(r"/content/drive/MyDrive/test_set",  
                                         target_size=(150,150),  
                                         batch_size =32,  
                                         class_mode = 'binary')
```

Found 442 images belonging to 2 classes.

Found 121 images belonging to 2 classes.

```
test_dataset.class_indices
```

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

```
model = keras.Sequential()
```

```
model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
```

```
model.add(keras.layers.MaxPool2D(2,2))
```

```
model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))
```

```
model.add(keras.layers.MaxPool2D(2,2))
```

```
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
```

```
model.add(keras.layers.MaxPool2D(2,2))
```

```
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
```

```
model.add(keras.layers.MaxPool2D(2,2))
```

```
model.add(keras.layers.Flatten())
```

```
model.add(keras.layers.Dense(512,activation='relu'))
```

```
model.add(keras.layers.Dense(1,activation='sigmoid'))
```

```
model.summary()
```

Model: "sequential"

---

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

conv2d (Conv2D) (None, 148, 148, 32) 896

max\_pooling2d (MaxPooling2D (None, 74, 74, 32) 0  
)

conv2d\_1 (Conv2D) (None, 72, 72, 64) 18496

max\_pooling2d\_1 (MaxPooling (None, 36, 36, 64) 0  
2D)

conv2d\_2 (Conv2D) (None, 34, 34, 128) 73856

max\_pooling2d\_2 (MaxPooling (None, 17, 17, 128) 0  
2D)

conv2d\_3 (Conv2D) (None, 15, 15, 128) 147584

max\_pooling2d\_3 (MaxPooling (None, 7, 7, 128) 0  
2D)

flatten (Flatten) (None, 6272) 0

dense (Dense) (None, 512) 3211776

dense\_1 (Dense) (None, 1) 513

=====

Total params: 3,453,121

Trainable params: 3,453,121

Non-trainable params: 0

---

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
r = model.fit(train_dataset,  
              epochs = 10,  
              validation_data = test_dataset)
```

Epoch 1/10

14/14 [=====] - 156s 11s/step - loss: 0.5977 - accuracy: 0.7353 -  
val\_loss: 0.2603 - val\_accuracy: 0.9256

Epoch 2/10

14/14 [=====] - 31s 2s/step - loss: 0.3629 - accuracy: 0.8507 - val  
\_loss: 0.1304 - val\_accuracy: 0.9752

Epoch 3/10

14/14 [=====] - 29s 2s/step - loss: 0.2061 - accuracy: 0.9276 - val  
\_loss: 0.0353 - val\_accuracy: 0.9917

Epoch 4/10

14/14 [=====] - 30s 2s/step - loss: 0.1594 - accuracy: 0.9457 - val  
\_loss: 0.0253 - val\_accuracy: 1.0000

Epoch 5/10

14/14 [=====] - 29s 2s/step - loss: 0.1493 - accuracy: 0.9434 - val  
\_loss: 0.0274 - val\_accuracy: 1.0000

Epoch 6/10

14/14 [=====] - 30s 2s/step - loss: 0.1211 - accuracy: 0.9548 - val  
\_loss: 0.0222 - val\_accuracy: 1.0000

Epoch 7/10

14/14 [=====] - 30s 2s/step - loss: 0.1200 - accuracy: 0.9525 - val  
\_loss: 0.1301 - val\_accuracy: 0.9256

Epoch 8/10

14/14 [=====] - 31s 2s/step - loss: 0.1361 - accuracy: 0.9434 - val  
\_loss: 0.0206 - val\_accuracy: 0.9917

Epoch 9/10

14/14 [=====] - 32s 2s/step - loss: 0.1498 - accuracy: 0.9412 - val  
\_loss: 0.0352 - val\_accuracy: 1.0000

Epoch 10/10

14/14 [=====] - 32s 2s/step - loss: 0.0850 - accuracy: 0.9661 - val  
\_loss: 0.0065 - val\_accuracy: 1.0000

model.save("forest1.h5")

predictions = model.predict(test\_dataset)

predictions = np.round(predictions)

4/4 [=====] - 6s 1s/step

predictions

array([[1.,

[1.,

[0.,

[1.,

[0.,

[1.,

[0.,

[0.,

[1.,

[1.,

[0.,

[0.,

[0.,

[1.,

[0.,

[0.,

[0.,

[1.,

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[0.],

[1.],

[1.],

[0.],

[0.],

[1.],

[1.],

[0.],

[1.],

[1.],

[0.],



[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[1.],

[0.],

[0.],

[1.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[0.],

[0.],

[1.],

[1.],

[0.],

[1.],

[1.],

[0.],

[1.],

[0.],

[1.],

[1.],

[0.],

[0.],

[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

[1.],

[0.],

[1.],

[0.],

[1.],

[0.],

[1.],

[1.],

[1.],

[1.],

[0.],

[0.],

[0.],

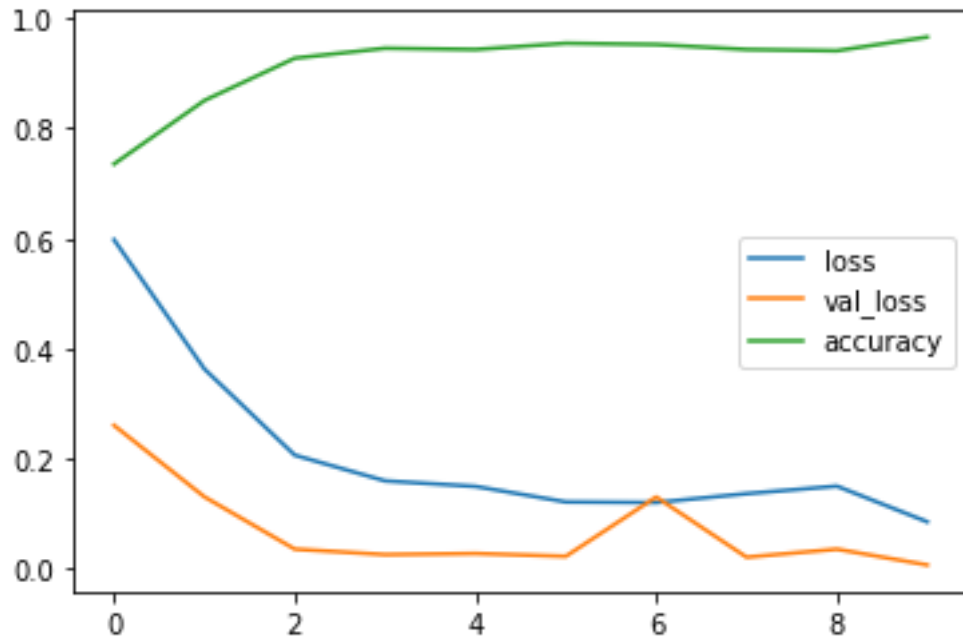
[0.],

[0.],

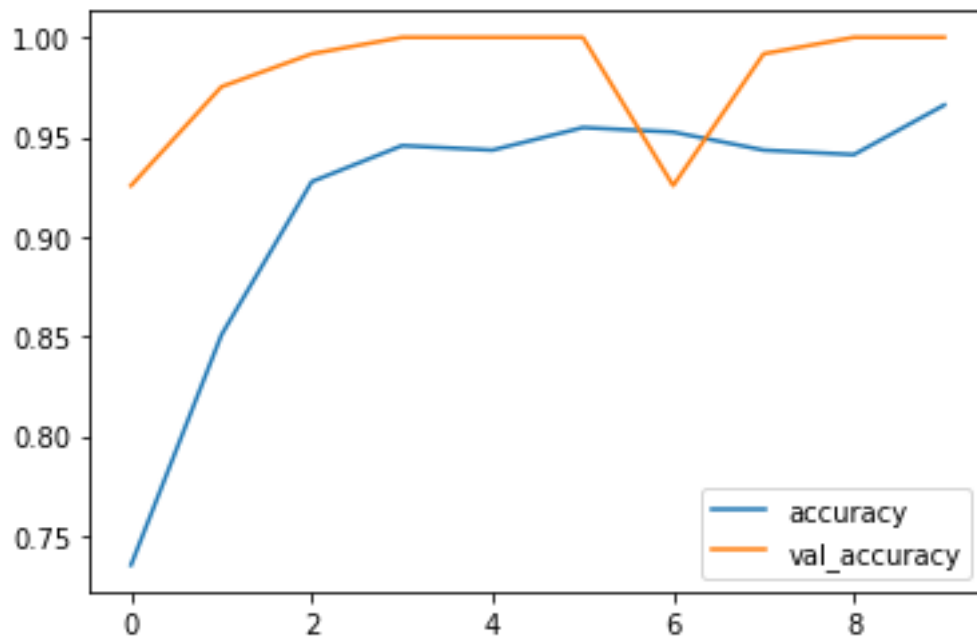
[1.],

```
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.], dtype=float32)
print(len(predictions))
121
import matplotlib.pyplot as plt
plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
plt.plot(r.history['accuracy'], label='accuracy')
plt.legend()
```

Out[11]:



```
plt.plot(r.history['accuracy'], label='accuracy')
plt.plot(r.history['val_accuracy'], label='val_accuracy')
plt.legend()
```



```
def predictImage(filename):
    img1 = image.load_img(filename,target_size=(150,150))
    plt.imshow(img1)
    Y = image.img_to_array(img1)
```

```

X = np.expand_dims(Y,axis=0)

val = model.predict(X)

print(val)

if val == 1:

    plt.xlabel("Fire")

elif val == 0:

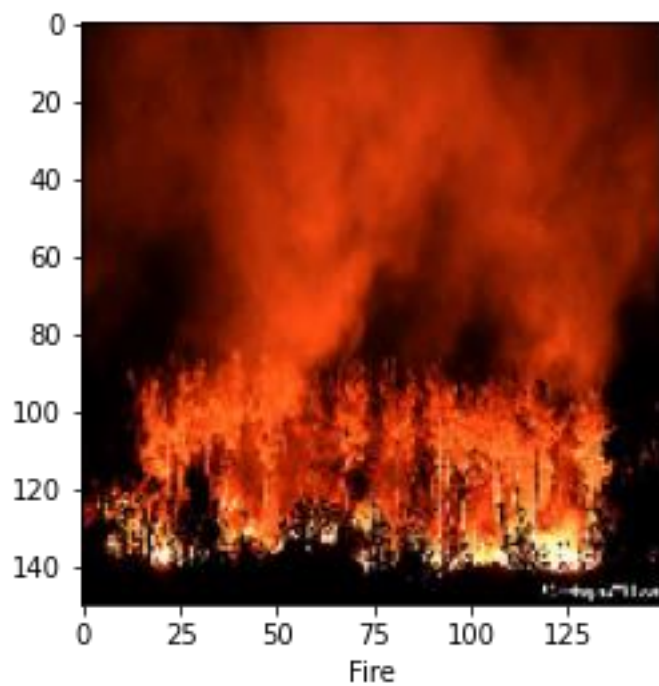
    plt.xlabel("No Fire")

predictImage(r"/content/drive/MyDrive/test_set/with fire/19464620_401.jpg")

1/1 [=====] - 0s 147ms/step

[[1.]]

```



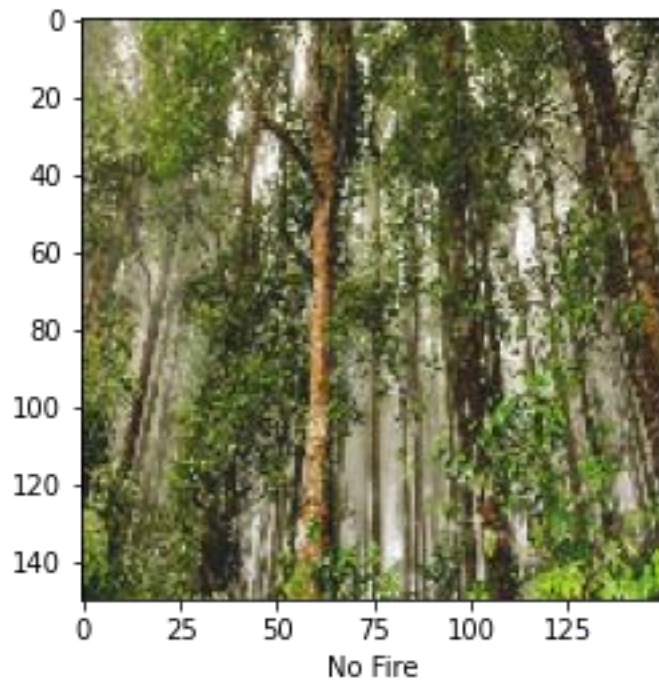
```

predictImage(r"/content/drive/MyDrive/test_set/forest/091318_LH_forest_loss_main_FREE.jpg")

1/1 [=====] - 0s 32ms/step

[[0.]]

```



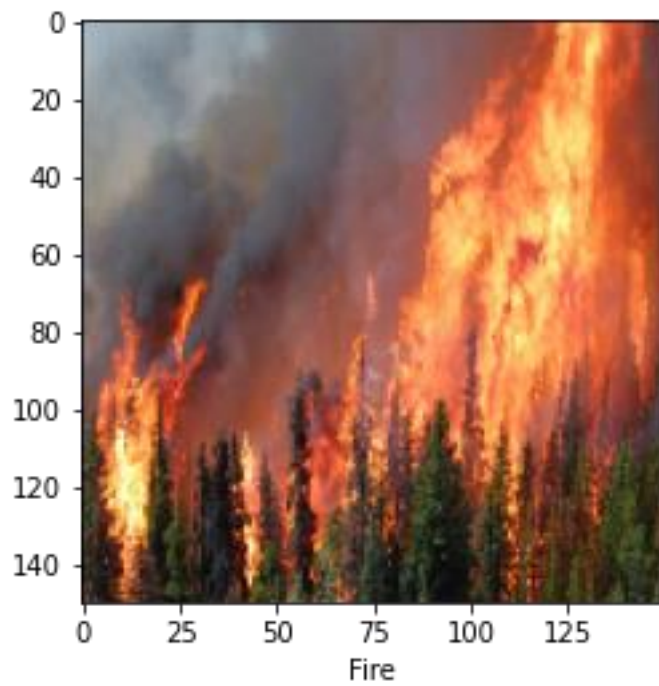
`predictImage(r"/content/drive/MyDrive/train_set/with fire/with fire (100).jpg")`

`[[1.]]`

`1/1 [=====] - 0s 31ms/step`

`[[1.]]`

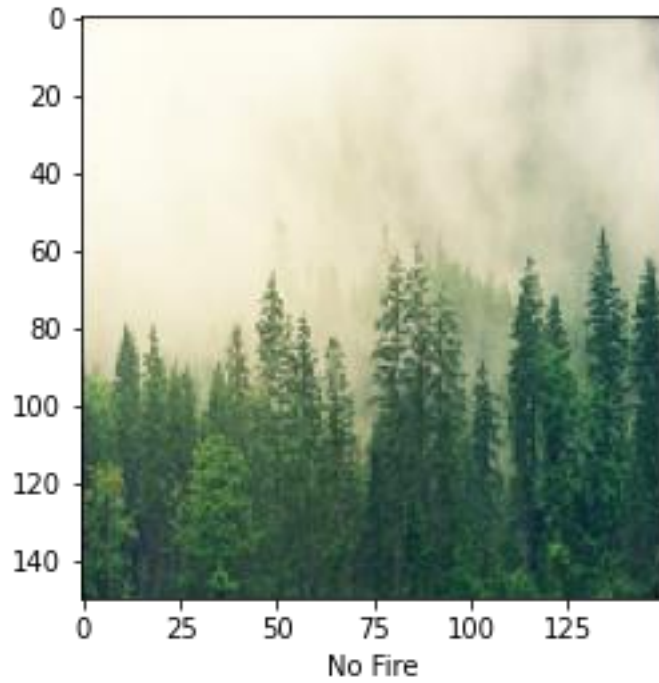
`[[1.0]]`



```
predictImage(r"/content/drive/MyDrive/test_set/forest/cold_daylight_environment_1423600_640x427.jpg")
```

```
1/1 [=====] - 0s 31ms/step
```

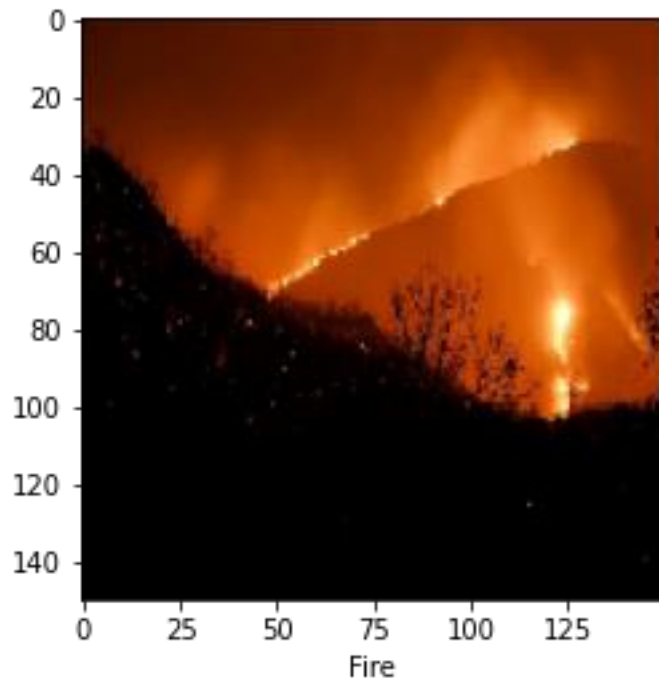
```
[[0.]]
```



```
predictImage(r"/content/drive/MyDrive/test_set/with fire/Fire_2_696x392.jpg")
```

```
1/1 [=====] - 0s 28ms/step
```

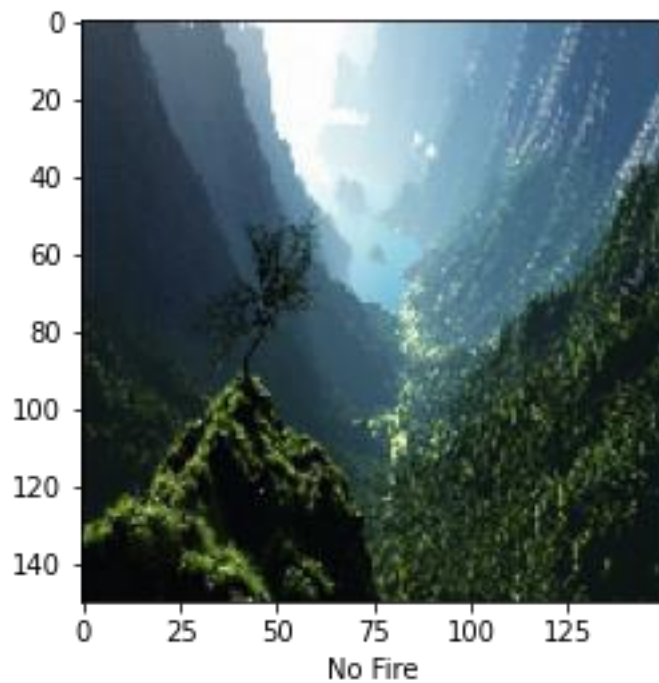
```
[[1.]]
```



`predictImage(r"/content/drive/MyDrive/train_set/forest/with_fire (104).jpg")`

1/1 [=====] - 0s 80ms/step

[[0.]]





## 5.5 SPRINT 3:

```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name":
      "python3",
      "display_name": "Python 3"
    },
    "language_info":
    { "name":
      "python"
    },
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "id": "cm0cXpbvyyBp",
        "outputId": "4bffc3ff-b763-4d6d-c12b-02b62f8c32fa"
      },
      "outputs": [
        {
```

```
"output_type": "execute_result",
"data": {
  "text/plain":
    [
      ""/content""
    ],
  "application/vnd.google.colaboratory.intrinsic+json": {
    "type": "string"
  }
},
"metadata": {},
"execution_count": 1
}
],
"source":
  ["pwd"
]
},
{
  "cell_type": "code",
  "source": [
    "!pip install keras\n",
    "!pip install tensorflow\n",
    "!pip install opencv-python"
  ],
  "metadata": {
    "colab": {

      "base_uri": "https:// localhost:8080/"
    },
    "id": "UnpPHFm0y4lm",
    "outputId": "543ceb28-d9f5-4c1c-9934-02075c827323"
```

```
},  
"execution_count": 2,  
"outputs": [  
  {  
    "output_type": "stream",  
    "name": "stdout",  
    "text": [  
      "Looking in indexes: https:// pypi.org/simple, https:// us-python.pkg.dev/colab-  
wheels/public/simple\n",  
      "Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages (2.9.0)\n",  
      "Looking in indexes: https:// pypi.org/simple, https:// us-python.pkg.dev/colab-  
wheels/public/simple\n",  
      "Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages  
(2.9.2)\n",  
      "Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages  
(from tensorflow) (1.15.0)\n",  
      "Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (2.9.0)\n",  
      "Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (14.0.6)\n",  
      "Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (0.4.0)\n",  
      "Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist- packages  
(from tensorflow) (3.19.6)\n",  
      "Requirement already satisfied: flatbuffers<2,>=1.12 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (1.12)\n",  
      "Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages  
(from tensorflow) (1.14.1)\n",  
      "Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from  
tensorflow) (21.3)\n",  
      "Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in  
/usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.0)\n",  
      "Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from
```

tensorflow) (3.1.0)\n",

"Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.7/dist-packages  
(from tensorflow) (1.3.0)\n",

"Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist

packages (from tensorflow) (1.6.3)\n",

"Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (4.1.1)\n",

"Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist- packages  
(from tensorflow) (2.1.0)\n",

"Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (1.50.0)\n",

"Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-  
packages (from tensorflow) (0.2.0)\n",

"Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist- packages  
(from tensorflow) (3.3.0)\n",

"Requirement already satisfied: keras-preprocessing>=1.1.1 in  
/usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)\n",

"Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dist-packages  
(from tensorflow) (1.21.6)\n",

"Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from  
tensorflow) (57.4.0)\n",

"Requirement already satisfied: tensorboard<2.10,>=2.9 in /usr/local/lib/python3.7/dist-packages  
(from tensorflow) (2.9.1)\n",

"Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in  
/usr/local/lib/python3.7/dist-packages (from tensorflow) (0.27.0)\n",

"Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.7/dist-  
packages (from astunparse>=1.6.0->tensorflow) (0.38.3)\n",

"Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist- packages  
(from h5py>=2.9.0->tensorflow) (1.5.2)\n",

"Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-  
packages (from tensorboard<2.10,>=2.9->tensorflow) (1.0.1)\n",

"Requirement already satisfied: markdown<3,>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (3.4.1)\n",

"Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.23.0)\n",

"Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.14.1)\n",

"Requirement already satisfied: tensorboard-plugin-wit<=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.8.1)\n",

"Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (0.4.6)\n",

"Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (0.6.1)\n",

"Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<2.10,>=2.9->tensorflow) (5.2.0)\n",

"Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages(from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (4.9)\n",

"Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist- packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (0.2.8)\n",

"Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist- packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow) (1.3.1)\n",

"Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist- packages (from markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (4.13.0)\n",

"Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (3.10.0)\n",

"Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist- packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9- >tensorflow) (0.4.8)\n",

"Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist- packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (3.0.4)\n",

"Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist- packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2022.9.24)\n",

"Requirement already satisfied: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9- >tensorflow) (1.24.3)\n",

"Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2.10)\n",

"Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages(from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9- >tensorflow) (3.2.2)\n",

"Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist- packages (from packaging->tensorflow) (3.0.9)\n",

"Looking in indexes: [https:// pypi.org/simple](https://pypi.org/simple), [https:// us-python.pkg.dev/colab- wheels/public/simple/](https://us-python.pkg.dev/colab-wheels/public/simple/)\n",

"Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (4.6.0.66)\n",

"Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages(from

```
opencv-python) (1.21.6)\n"
```

```
]
```

```
}
```

```
]
```

```
},
```

```
{
```

```
"cell_type": "code",
```

```
"source": [
```

```
"from keras.models import Sequential\n",
```

```
"from keras.layers import Dense\n",
```

```
"from keras.layers import Convolution2D\n",
```

```
"from keras.layers import MaxPooling2D\n",
```

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

```
],
```

```
"metadata": {
```

```
"id": "LXQp5JUy8by"
```

```
},
```

```
"execution_count": 3,
```

```
"outputs": []
```

```
},
```

```
{
```

```
"cell_type": "code",
```

```
"source": [
```

```
"from tensorflow.keras.preprocessing.image import ImageDataGenerator\n", "train
```

```
= ImageDataGenerator(rescale=1/255)\n",
```

```
"test = ImageDataGenerator(rescale=1/255)"
```

```
],
```

```
"metadata": {
```

```
"id": "-joU1JMNzBID"
```

```
},
```

```
"execution_count": 4,
```

```
"outputs": []
```

```
},
{
  "cell_type": "code",
  "source": [
    "pwd"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 35
    },
    "id": "3zbBmApYzzob",
    "outputId": "fa8fb36a-473c-4662-dbf7-67598141fa83"
  },
  "execution_count": 8,
  "outputs": [
    {

      "output_type": "execute_result",
      "data": {
        "text/plain":
          [
            ""/content""
          ],
        "application/vnd.google.colaboratory.intrinsic+json": {
          "type": "string"
        }
      },
      "metadata": {},
      "execution_count": 8
    }
  ]
}
```



```

},
{
  "cell_type": "code",
  "source": [
    "import os\n",
    "filenames = os.listdir('/content/drive/MyDrive/train_set')"
  ],
  "metadata": {
    "id": "hNu0gAxNz5wV"
  },
  "execution_count": 9,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "x_train = train_dataset = train.flow_from_directory('/content/drive/MyDrive/train_set',\n",
    "                                                    target_size= (64,64),\n",
    "                                                    batch_size = 32,\n",
    "                                                    class_mode = 'binary')\n",
    "x_test = test_dataset = test.flow_from_directory('/content/drive/MyDrive/test_set',\n",
    "                                                  target_size= (64,64),\n",
    "                                                  batch_size = 32,\n",
    "                                                  class_mode = 'binary')
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    }
  },
  "id": "Ewd7ALq80I2h",
  "outputId": "707c55af-f9b0-4164-e2cf-d642448ec7d1"
}

```

```

},
"execution_count": 10,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "Found 442 images belonging to 2 classes.\n",
      "Found 121 images belonging to 2 classes.\n"
    ]
  }
]
},
{
  "cell_type": "code",
  "source": [
    "x_test.class_indices"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "Ypg9hbSD0VMb",
    "outputId": "ddf6fee2-231b-4b2a-fc45-156d2c968517"
  },
  "execution_count": 11,
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "{ 'forest': 0, 'with fire': 1 }"
        ]
      }
    ]
  }
]

```

```

    ]
  },
  "metadata": {},
  "execution_count": 11
}
]

},
{
  "cell_type": "code",
  "source": [
    "model = Sequential()"
  ],
  "metadata": {
    "id": "qyQ20wPg0XZg"
  },
  "execution_count": 12,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))"
  ],
  "metadata": {
    "id": "3CCe1wJK0dq6"
  },
  "execution_count": 13,
  "outputs": []
},
{
  "cell_type": "code",

```

```
"source": [  
  "model.add(MaxPooling2D(2,2))"  
],  
"metadata": {  
  "id": "KiB6bTwt0gYI"  
},  
"execution_count": 14,  
"outputs": []  
},  
{  
  "cell_type": "code",  
  "source": [  
    "model.add(Flatten())"  
  ],  
  "metadata": {  
    "id": "gezPUvME0ixZ"  
  },  
  "execution_count": 15,  
  "outputs": []  
},  
{  
  "cell_type": "code",  
  "source": [  
    "model.add(Dense(512,activation='relu'))\n",  
    "model.add(Dense(1,activation='sigmoid'))"  
  ],  
  "metadata": {  
    "id": "jITuRAuQ0I7j"  
  },  
  "execution_count": 16,  
  "outputs": []  
},
```

```

{
  "cell_type": "code",
  "source": [
    "model.compile(optimizer=\"adam\",loss=\"binary_crossentropy\",metrics=[\"accuracy\"])"
  ],
  "metadata": {
    "id": "3VRs-oXD0quq"
  },
  "execution_count": 17,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "model.fit(x_train,steps_per_epoch=14
,epochs=10,validation_data=x_test,validation_steps=4)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "7Ek-Gm6P0vpW",
    "outputId": "f6a1472f-709a-4f36-9d4f-fa1946838e10"
  },
  "execution_count": 18,

  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [

```

```
"Epoch 1/10\n",
"14/14 [=====] - 157s 11s/step - loss: 3.7764 - accuracy:
0.5928 - val_loss: 0.3833 - val_accuracy: 0.8182\n",
"Epoch 2/10\n",
"14/14 [=====] - 22s 2s/step - loss: 1.0376 - accuracy:
0.6855 - val_loss: 0.1756 - val_accuracy: 0.9339\n",
"Epoch 3/10\n",
"14/14 [=====] - 21s 1s/step - loss: 0.2968 - accuracy:
0.8688 - val_loss: 0.1248 - val_accuracy: 0.9835\n",
"Epoch 4/10\n",
"14/14 [=====] - 21s 2s/step - loss: 0.2413 - accuracy:
0.9072 - val_loss: 0.1233 - val_accuracy: 0.9504\n",
"Epoch 5/10\n",
"14/14 [=====] - 21s 2s/step - loss: 0.1790 - accuracy:
0.9321 - val_loss: 0.0887 - val_accuracy: 0.9669\n",
"Epoch 6/10\n",
"14/14 [=====] - 21s 2s/step - loss: 0.1427 - accuracy:
0.9457 - val_loss: 0.0762 - val_accuracy: 0.9752\n",
"Epoch 7/10\n",
"14/14 [=====] - 21s 2s/step - loss: 0.1059 - accuracy:
0.9706 - val_loss: 0.0514 - val_accuracy: 0.9917\n",
"Epoch 8/10\n",
"14/14 [=====] - 21s 1s/step - loss: 0.0835 - accuracy:
0.9774 - val_loss: 0.0272 - val_accuracy: 1.0000\n",
"Epoch 9/10\n",
"14/14 [=====] - 21s 2s/step - loss: 0.0657 - accuracy:
0.9774 - val_loss: 0.0266 - val_accuracy: 0.9917\n",
"Epoch 10/10\n",
"14/14 [=====] - 23s 2s/step - loss: 0.0465 - accuracy:
0.9819 - val_loss: 0.0153 - val_accuracy: 1.0000\n"
]
```

```

{
  "output_type": "execute_result",
  "data": {
    "text/plain": [

      "<keras.callbacks.History at 0x7f2c0d8ec590>"

    ]
  },
  "metadata": {},
  "execution_count": 18
}
],
{
  "cell_type": "code",
  "source": [
    "model.save(\"forest1.h5\")"
  ],
  "metadata": {
    "id": "wqMJztaF00Qh"
  },
  "execution_count": 19,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "!tar -zcvf image-classification-model_new.tgz forest1.h5"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    }
  }
}

```


```
    },
    "id": "3jah6H9-2Znl",
    "outputId": "ed7217d8-6994-4f98-b136-98b1ff5f6b8b"
  },
  "execution_count": 20,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "forest1.h5\n"
      ]
    }
  ]
},
{
  "cell_type": "code",
  "source": [
    "ls -l"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "V9oOmNUW2cWk",
    "outputId": "c706f1e1-1c8d-4b26-d8dd-a4bcdfe60e5c"
  },
  "execution_count": 21,
  "outputs": [
    {
      "output_type": "stream",
```



```
"name": "stdout",
"text": [
  "\u001b[0m\u001b[01;34mdrive\u001b[0m\n",
  "forest1.h5\n",
  "image-classification-model_new.tgz\n",
  "\u001b[01;34msample_data\u001b[0m\n"
]
}
],
{
  "cell_type": "code",
  "source": [
    "!pip install watson-machine-learning-client --upgrade"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 1000
    },
    "id": "nQj_2bZ62ns3",
    "outputId": "103e599b-947e-46f0-eb02-0dd7142a2130"
  },
  "execution_count": 22,
```

```

"outputs": [
{
  "output_type":
    "stream",      "name":
      "stdout",
      "text": [

        "Looking in indexes: https:// pypi.org/simple, https:// us-python.pkg.dev/colab-
wheels/public/simple/\n",
        "Collecting watson-machine-learning-client\n",
        "  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)\n",
        "\u001b[K  538 kB 7.0 MB/s
\n", "\u001b[?25hRequirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from watson-machine-learning-client) (2.23.0)\n",

        "Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from
watson-machine-learning-client) (4.64.1)\n",

        "Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from
watson-machine-learning-client) (0.8.10)\n",


        "Collecting lomond\n",

        "  Downloading lomond-0.3.3-py2.py3-none-any.whl (35 kB)\n",


        "Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from
watson-machine-learning-client) (2022.9.24)\n",

        "Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from
watson-machine-learning-client) (1.24.3)\n",

        "Collecting boto3\n",

        "  Downloading boto3-1.26.11-py3-none-any.whl (132 kB)\n",
        "\u001b[K  132 kB 53.7
MB/s \n", "\u001b[?25hRequirement already satisfied: pandas in
/usr/local/lib/python3.7/dist-
packages (from watson-machine-learning-client) (1.3.5)\n",

        "Collecting ibm-cos-sdk\n",

        "  Downloading ibm-cos-sdk-2.12.0.tar.gz (55 kB)\n",
        "\u001b[K  55 kB 3.9
MB/s \n", "\u001b[?25hCollecting jmespath<2.0.0,>=0.7.1\n",

```

[illegible]

"Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->watson-machine-learning-client) (2022.6)\n",

"Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages(from pandas->watson-machine-learning-client) (1.21.6)\n",

"Building wheels for collected packages: ibm-cos-sdk, ibm-cos-sdk-core, ibm-cos-sdk-s3transfer\n",

" Building wheel for ibm-cos-sdk (setup.py) ... \u001b[?25l\u001b[?25hdone\n",

" Created wheel for ibm-cos-sdk: filename=ibm\_cos\_sdk-2.12.0-py3-none-any.whl size=73931 sha256=841189e9104158317d85f159529014a3c34da1db4455cc140ecfd657ba3ed2ef\n",

" Stored in directory:

/root/.cache/pip/wheels/ec/94/29/2b57327cf00664b6614304f7958abd29d77ea0e5bbece2ea57\n",

" Building wheel for ibm-cos-sdk-core (setup.py) ... \u001b[?25l\u001b[?25hdone\n", "

Created wheel for ibm-cos-sdk-core: filename=ibm\_cos\_sdk\_core-2.12.0-py3-none-any.whl size=562962 sha256=6dd5fd11a6eb4cc566eefe7e82e573055238fbc5bdafc2604c164f8a6fa02255\n",

" Stored in directory:

/root/.cache/pip/wheels/64/56/fb/5cd6f4f40406c828a5289b95b2752a4d142a9afb359244ed8

d\n",

" Building wheel for ibm-cos-sdk-s3transfer (setup.py) ...

\u001b[?25l\u001b[?25hdone\n",

" Created wheel for ibm-cos-sdk-s3transfer: filename=ibm\_cos\_sdk\_s3transfer-2.12.0-py3-none-any.whl size=89778 sha256=3c9215c3ddaa7fc31a8c3783a78b5e3aa7a4cb9ea8d7dc1178e709c1ccb392a8\n",

" Stored in directory:

/root/.cache/pip/wheels/57/79/6a/ffe3370ed7ebc00604f9f76766e1e0348dcdcad2b2e32df9e1\n",

"Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-s3transfer\n", "Installing collected packages: urllib3, requests, jmespath, ibm-cos-sdk-core, botocore, s3transfer, ibm-cos-sdk-s3transfer, lomond, ibm-cos-sdk, boto3, watson-machine-learning-client\n",

" Attempting uninstall: urllib3\n",

" Found existing installation: urllib3 1.24.3\n",

```

    " Uninstalling urllib3-1.24.3:\n",
    " Successfully uninstalled urllib3-1.24.3\n",
    Attempting uninstall: requests\n",
    " Found existing installation: requests 2.23.0\n", "
      Uninstalling requests-2.23.0:\n",
    " Successfully uninstalled requests-2.23.0\n",

    "Successfully installed boto3-1.26.11 botocore-1.29.11 ibm-cos-sdk-2.12.0 ibm-cos-sdk-
core-2.12.0 ibm-cos-sdk-s3transfer-2.12.0 jmespath-0.10.0 lomond-0.3.3 requests-2.28.1 s3transfer-
0.6.0 urllib3-1.26.12 watson-machine-learning-client-1.0.391\n"
  ]
},
{
  "output_type": "display_data",
  "data": {
    "application/vnd.colab-display-data+json": {
      "pip_warning": {
        "packages": [
          "requests",
          "urllib3"
        ]
      }
    }
  },
  "metadata": {}
}
]

},
{
  "cell_type": "code",
  "source": [
    "def guid_from_space_name(client, space_name):\n",

```

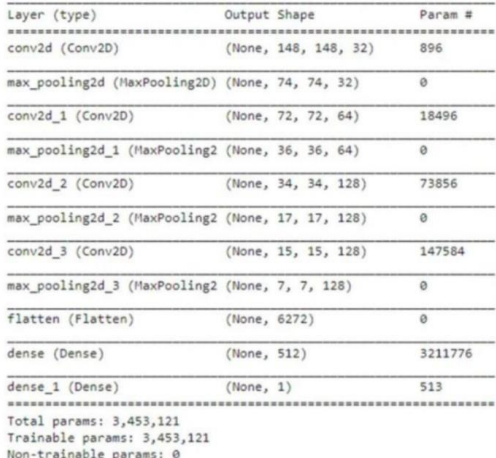
```

" space = client.spaces.get_details()\n",
" #print(space)\n",
" return(next(item for item in space['resources']if item['entity']['name'] ==
space_name)['metadata']['id'])"
],
"metadata": {
" id": "QSDKfvy_3H8Q"
},
"execution_count": 25,
"outputs": []
}
]
}

```

## 6 TESTING AND RESULTS

### 6.1 Performance Testing

S. No	Parameter	Values	Screenshot
1.	Model Summary	3,453,213	 <pre> Layer (type)                 Output Shape              Param # ----- conv2d (Conv2D)              (None, 148, 148, 32)     896 max_pooling2d (MaxPooling2D) (None, 74, 74, 32)       0 conv2d_1 (Conv2D)            (None, 72, 72, 64)       18496 max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)       0 conv2d_2 (Conv2D)            (None, 34, 34, 128)      73856 max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)      0 conv2d_3 (Conv2D)            (None, 15, 15, 128)      147584 max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)       0 flatten (Flatten)            (None, 6272)             0 dense (Dense)                (None, 512)              3211776 dense_1 (Dense)              (None, 1)                513 ----- Total params: 3,453,121 Trainable params: 3,453,121 Non-trainable params: 0 </pre>

2.	Accuracy	<p>Training Accuracy - 0.9663</p> <p>Validation Accuracy -0.9795</p>	<p>Epoch 1/10  14/14 [=====] - 96s 7s/step - loss: 0.5717 - accuracy: 0.6552 - val_loss: 0.2805 - val_accuracy: 0.8750  Epoch 2/10  14/14 [=====] - 84s 6s/step - loss: 0.3266 - accuracy: 0.8434 - val_loss: 0.1193 - val_accuracy: 0.9667  Epoch 3/10  14/14 [=====] - 74s 5s/step - loss: 0.2247 - accuracy: 0.9227 - val_loss: 0.1184 - val_accuracy: 0.9500  Epoch 4/10  14/14 [=====] - 75s 5s/step - loss: 0.1682 - accuracy: 0.9425 - val_loss: 0.0206 - val_accuracy: 1.0000  Epoch 5/10  14/14 [=====] - 82s 5s/step - loss: 0.1173 - accuracy: 0.9610 - val_loss: 0.0631 - val_accuracy: 0.9667  Epoch 6/10  14/14 [=====] - 76s 5s/step - loss: 0.0925 - accuracy: 0.9741 - val_loss: 0.0169 - val_accuracy: 1.0000  Epoch 7/10  14/14 [=====] - 80s 6s/step - loss: 0.0804 - accuracy: 0.9714 - val_loss: 0.0194 - val_accuracy: 1.0000  Epoch 8/10  14/14 [=====] - 72s 5s/step - loss: 0.0610 - accuracy: 0.9861 - val_loss: 0.0455 - val_accuracy: 0.9833  Epoch 9/10  14/14 [=====] - 70s 5s/step - loss: 0.1032 - accuracy: 0.9676 - val_loss: 0.0044 - val_accuracy: 1.0000  Epoch 10/10  14/14 [=====] - 92s 6s/step - loss: 0.1144 - accuracy: 0.9665 - val_loss: 0.0007 - val_accuracy: 0.9833</p>
----	----------	--	--

## 6.2 User acceptance testing

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	1	2	0	4
Duplicate	0	0	0	0	0
External	0	0	2	1	3
Fixed	4	2	4	1	11
Not Reproduced	0	0	0	0	0
Skipped	0	0	1	1	2
Won't Fix	0	0	0	1	1
Totals	5	3	9	4	21

**a. Test case**

				<b>Date</b>	<b>08-Nov-2022</b>			
				<b>Team ID</b>	<b>PNT2022TMID07050</b>			
				<b>Project Name</b>	<b>Emerging Methods for Early Detection of Forest fire</b>			
				<b>Maximum Marks</b>	<b>4 marks</b>			
<b>Test case ID</b>	<b>Feature Type</b>	<b>Component</b>	<b>Test Scenario</b>	<b>Expected Result</b>	<b>Actual Result</b>	<b>Status</b>	<b>BUGID</b>	<b>Executed By</b>
Home Page_TC_O01	UI	Home Page	Display the Emerging Methods of Forest Fires	Displaying the Home Page	Home Page displayed	Pass	-	Devi, Esther
Home Page_TC_O02	UI	Home Page	Displayed the Prediction Forest Fires	Displaying the content of home page	Content of Home page is displayed	Pass	-	Divya, Akshara
Home Page_TC_O03	Functional	Home page	Checks whether the Drop the Image Here! Button is visible	Displays the Button	Drop the Image Here! Button is pops up.	Pass	-	Devi, Esther
PredictedPageTC_OO4	Functional	Predicted page	Display the Prediction Page and Choose Image Button	Displays the Prediction Page and the Choose Image Button	Prediction page displayed. Choose image button was clicked.	Pass	-	Divya, Akshara, Devi, Esther.
PredictedPage_TC_OO4	Functional	Predicted page	Select the Image and Click the Predict Button	Display the selected Image	Displays the Selected Image	Pass	-	Devi, Esther



## b. Test case analysis

Section	Total Cases	Not Tested	Fails	Passes
Client Application	10	0	0	10
Security	2	0	0	2
Performance	2	0	0	2
Exception Reporting	2	0	0	2
Final Report Output	3	0	0	3

## 7. ENTIRE MODEL:

*#Importing Keras libraries*

**import** keras

*#Importing ImageDataGenerator from Keras*

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

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

*#Defining the Parameters*

train\_datagen=ImageDataGenerator(rescale=1./255, shear\_range=0.2, rotation\_range=180, zoom\_range=0.2, horizontal\_flip=**True**)

test\_datagen=ImageDataGenerator(rescale=1./255, shear\_range=0.2, rotation\_range=180, zoom\_range=0.2, horizontal\_flip=**True**)

*#Applying ImageDataGenerator functionality to train dataset*

x\_train=train\_datagen.flow\_from\_directory('/content/drive/MyDrive/train\_set', target\_size=(64, 64), batch\_size=32, class\_mode='binary')

Found 442 images belonging to 2 classes.

```
x_test=test_datagen.flow_from_directory('/content/drive/MyDrive/test_set',target_size=(64,64),  
batch_size=32,class_mode='binary')
```

Found 121 images belonging to 2 classes.

```
#to define the linear Initialisation import sequential
```

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

```
#to add layers import Dense
```

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

```
#to create Convolutional 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
```

```
model = Sequential()
```

```
#Adding CNN Layers
```

```
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 Dense layers
```

```
#add hidden layers
```

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

```
#add output layer
```

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

```
#configuring the learning process
```

```
model.compile(loss='binary_crossentropy',optimizer="adam",metrics=["accuracy"])
```

*#Training the model*

```
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_steps=4)
```

Epoch 1/10

14/14 [=====] - 152s 11s/step - loss: 1.2336 - accuracy: 0.6244 - val\_loss: 0.3944 - val\_accuracy: 0.8760

Epoch 2/10

14/14 [=====] - 23s 2s/step - loss: 0.3932 - accuracy: 0.8394 - val\_loss: 0.1940 - val\_accuracy: 0.9421

Epoch 3/10

14/14 [=====] - 22s 2s/step - loss: 0.2676 - accuracy: 0.8914 - val\_loss: 0.1266 - val\_accuracy: 0.9835

Epoch 4/10

14/14 [=====] - 23s 2s/step - loss: 0.2115 - accuracy: 0.9072 - val\_loss: 0.0966 - val\_accuracy: 0.9587

Epoch 5/10

14/14 [=====] - 22s 2s/step - loss: 0.1967 - accuracy: 0.9231 - val\_loss: 0.0950 - val\_accuracy: 0.9752

Epoch 6/10

14/14 [=====] - 24s 2s/step - loss: 0.1907 - accuracy: 0.9253 - val\_loss: 0.1514 - val\_accuracy: 0.9256

Epoch 7/10

14/14 [=====] - 23s 2s/step - loss: 0.2169 - accuracy: 0.9072 - val\_loss: 0.0874 - val\_accuracy: 0.9669

Epoch 8/10

14/14 [=====] - 26s 2s/step - loss: 0.1809 - accuracy: 0.9253 - val\_loss: 0.0743 - val\_accuracy: 0.9669

Epoch 9/10

14/14 [=====] - 23s 2s/step - loss: 0.1777 - accuracy: 0.9299 - val\_loss: 0.0670 - val\_accuracy: 0.9917

Epoch 10/10

14/14 [=====] - 26s 2s/step - loss: 0.2067 - accuracy: 0.9095 - val\_loss: 0.0617 - val\_accuracy: 0.9917

*#Save the model*

```

model.save("/content/drive/MyDrive/forest1.h5")
#Predictions
#import load model from keras.model
from keras.models import load_model
#import image from keras
from tensorflow.keras.preprocessing import image
import numpy as np
#import cv2
import cv2
#load the saved model
model=load_model("/content/drive/MyDrive/forest1.h5")
img=image.load_img('/content/drive/MyDrive/test_set/with
fire/Forest_fire_MNRF_esize_IMG_6743.jpg')
x=image.img_to_array(img)
res=cv2.resize(x,dsize=(64,64),interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)
pred=model.predict(x)
pred = int(pred[0][0])
pred
int(pred)
1/1 [=====] - 0s 139ms/step
1

```

pip install twilio

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting twilio

Downloading twilio-7.15.3-py2.py3-none-any.whl (1.4 MB)

 1.4 MB 6.5 MB/s

Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2022.6)

Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)

Collecting PyJWT<3.0.0,>=2.0.0

Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2.10)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2022.9.24)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)

Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (1.24.3)

Installing collected packages: PyJWT, twilio

Successfully installed PyJWT-2.6.0 twilio-7.15.3

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

**if** pred==0:

    print('Forest fire')

    account\_sid='AC4c9a105651d0150d1b85af1bd4cf090c'

    auth\_token='ee06c7d5053b02ef2ee7689157b255ee'

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

    message=client.messages \

    .create(

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

*#use twilio free number*

        from\_='+15134660214',

*#to number*

        to='+919361632961')

    print(message.sid)

    print("Fire detected")

    print("SMS Sent!")

**elif** pred==1:

    print('No Fire')

No Fire

*#Open cv for video processing*

pip install twilio

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Requirement already satisfied: twilio in /usr/local/lib/python3.7/dist-packages (7.15.3)  
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)  
Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.6.0)  
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2022.6)  
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)  
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (1.24.3)  
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2022.9.24)  
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2.10)

In [36]:

*#Creating An Account in Twilio Service*

*#Sending Alert Message*

**from** logging **import** WARNING

*#import opencv library*

**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

*#import playsound package*

**import** cv2

**import** numpy as np

**from** google.colab.patches **import** cv2\_imshow

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

```

import librosa
from tensorflow.keras.preprocessing import image
from keras.models import load_model
# Create a VideoCapture object and read from input file
# If the input is the camera, pass 0 instead of the video file name
cap = cv2.VideoCapture('/content/drive/MyDrive/FOREST FIRE (1).mp4')

# Check if camera opened successfully
if (cap.isOpened() == False):
    print("Error opening video stream or file")

# Read until video is completed
while(cap.isOpened()):
    # Capture frame-by-frame
    ret, frame = cap.read()
    if ret == True:

        cv2_imshow(frame)
        x=image.img_to_array(frame)
        res=cv2.resize(x,dsize=(64,64),interpolation=cv2.INTER_CUBIC)
        #expand the image shape
        x=np.expand_dims(res,axis=0)
        model=load_model("/content/drive/MyDrive/forest1.h5")
        pred=model.predict(x)
        pred = int(pred[0][0])
        pred
        int(pred)
        if pred==0:
            print('Forest fire')
            break
        else:
            print("no danger")
            break

```

```
# When everything done, release the video capture object  
cap.release()
```

```
# Closes all the frames  
cv2.destroyAllWindows()
```



```
1/1 [=====] - 0s 70ms/step
```

Forest fire

```
from twilio.rest import Client  
if pred==0:  
    print('Forest fire')  
    from twilio.rest import Client  
    account_sid='AC4c9a105651d0150d1b85af1bd4cf090c'  
    auth_token='ee06c7d5053b02ef2ee7689157b255ee'  
    client=Client(account_sid,auth_token)  
    message=client.messages \  
    .create(  
        body='forest fire is detected,stay alert',  
        #use twilio free number  
        from_='+15134660214',
```



```
#to number
to='+919361632961')
print(message.sid)
print("Fire detected")
print("SMS Sent!")
elif pred==1:
    print('No Fire')
Forest fire
SM6c3521055b9c8a7899bfb240b5ea1b51
Fire detected
SMS Sent!
```

## OUTPUT SCREENSHOT:



## 8. ADVANTAGES & DISADVANTAGES

### ADVANTAGES:

- The proposed system detects the forest fire at a faster rate compared to existing system. It has enhanced data collection feature.
- The major aspect is that it reduces false alarm and also has accuracy due to various sensors present.
- It minimize the human effort as it works automatically. This is meagre -cost due to which can be easily accessed.

- The main objective of our project is to receive an alert message through an app to the respective user.

#### **DISADVANTAGES:**

- The electrical interference diminishes the potency of radio receiver.
- The main drawback is that it has less coverage range areas

## **9. CONCLUSION**

This type of system is the first of its kind to ensure no further damage is then to forests when there is a fire breakout and instantly a message is sent to the user through the App. Immediate response or early warning to a fire breakout is mostly the only way to avoid losses and biology, cultural heritage damages to a great extent. Therefore the most important goals in fire surveillance are quick and authentic detection of fire. It is so much easier to suppress fire while it is in its early stages. info about the progress of fire is highly valuable for managing fire during all its stages. Based on this data the firefighting staff can be guided on target to block fire before it reaches cultural heritage sites and to suppress it quickly by utilizing required firefighting equipment and vehicles. With further research and invention, this project can be implemented in various forest areas so that we can save our forests and maintain great environs.

## **10. FUTURE SCOPE**

This project is far from complete and there is a lot of room for betterment. Some of the betterment that can be made to this project are as follows:

An Additional pump can be added so that it automatically sends water when there is a fire breakout. Also industrial sensors can be used for better ranging and accuracy.

→ This project has endless potential and can always be enhanced to become better. enforce this concept in the real world will benefit several industries and reduce the workload on many workers, enhancing overall work efficiency.

**GitHub :**

<https://github.com/IBM-EPBL/IBM-Project-19171-1659694051>