Emerging Methods For Early Detection Of Forest Fires

Team ID: PNT2022TMID03442

Team Size: 4

Team Members

Mohammed Zaid	**212219220031**
Mohammed Suhaib	**212219220030**
Sureshram E	**212219220054**
Veeramuthuselvan T	**212219220058**

INDEX

Chapter-1

INTRODUCTION

1.1Project Overview

The ecological balance is maintained by the forests. It acts as an environment that enriches the diversity of various organisms. The motive of the project is to detect the forest fire as early as possible so that we can preserve the life of various species prevailing in it from the fire. Utilizing the currently available techniques of smoke sensors put in the buildings, fire detection can be incredibly challenging. Due to their outdated technology and design, they are costly and slow. The use of artificial intelligence for identification and issuing alerts with video from CCTV footage is critically examined in this study. For this project, a self-built dataset of videoframes with fire is used. The data is then preprocessed and a machine-learning model is built using CNN. The dataset's test set is used as input to verify the method, and experiments are recorded. The goal of the project is to create a machine that is both affordable and very precise and can be applied to practically any fire-detecting situation.

1.2 Purpose

One of the key elements in keeping the environment in balance is forests. When a fire breaks out in a forest, it can be very dangerous. However, a forest fire is typically discovered after it has spread across a significant area. It might not always be able to put out the fire. As a result, the environmental impact is worse than anticipated. The environment suffers because of the forest fire's large-scale carbon dioxide (CO_2) emissions. It would result in the global extinction of rare species. Additionally, it may have an effect on the weather, which may lead to serious problems like earthquakes, excessive rain, floods, and so forth. The forest is a big surface area covered with trees, tonnes of dried leaves, woodlands, and other things. When the fire first ignites, these substances help it grow. Fire might start from various causes, including smoking, fireworks-themed events, or high summer temperatures. Once a fire starts, it won't stop until it has entirely burned itself out. When the fire is noticed as early as feasible, the damage and the cost associated with identifying it due to a forest fire can be minimized. Therefore, in this case, fire detection is crucial. A good effect can be had by locating the fire's specific location and notifying the fire authorities as soon as the fire occurs. Thus it is crucial to implement a system to identify fires as soon as possible.

Chapter 2

LITERATURE SURVEY

2.1 Existing problem

Smoke alarms and heat alarms are being used to detect fires. One module is not enough to monitor all of the potential fire-prone areas, which is the fundamental drawback of smoke sensor alarms and heat sensor alarms. The only way to avoid a fire is to exercise caution at all times. Even if they are installed in every nook and cranny, it is not enough to constantly produce an efficient output. As the number of smoke sensors required rises, the price will rise by a factor of multiples. The suggested system can generate reliable and highly accurate alerts within seconds of an accident or a fire. One piece of software powers the entire surveillance network, which lowers costs. Data scientists and machine learning experts are actively conducting research in this area.

2.2 References

S. NO	TITLE	AUTHOR	YEAR
1.	Using Popular Object Detection Methods for Real Time Forest Fire Detection	Shixiao Wu, Libing Zhang	2013
2.	Forest Monitoring System for Early Fire Detection Based on Convolutional Neural Network and UAV imagery	Georgi Dimitrov Georgiev, Georgi Hristov	2020
3.	An energy efficient framework for detection and monitoring of forest fire using mobile agent in wireless sensor networks	Kartik Trivedi, Ashish kumar Srivastava	2015

2.3 Problem Statement Definition

Forest fires result in a wide range of negative effects, including the destruction of wildlife habitat, the extinction of plants and animals, the destruction of nutrient-rich top soil, the reduction of forest cover, the loss of valuable timber resources, the ozone layer being destroyed, the loss of livelihood for tribal and poor people, the acceleration of global warming, the increase in atmospheric carbon dioxide concentration, the degradation of catchment areas, the loss of biodiversity, the spread of disease, etc. Thus, Develop a system to detect forest fires at the earliest stage possible using the latest technologies.

Chapter 3

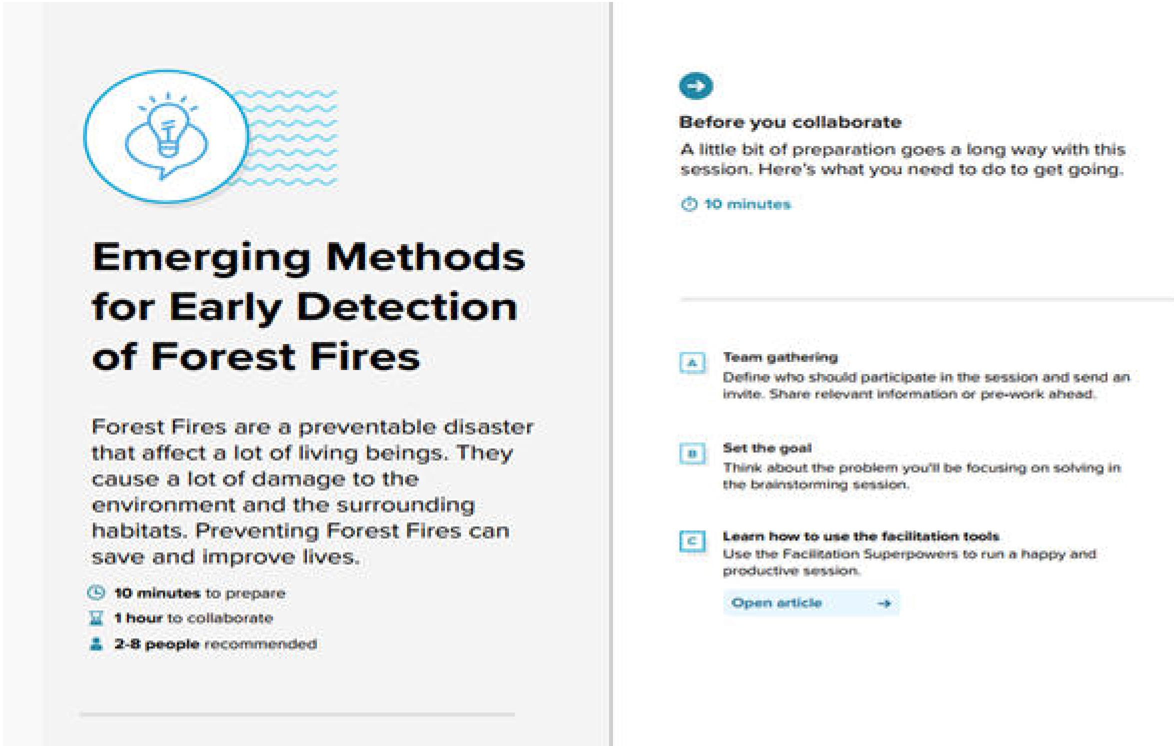
IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

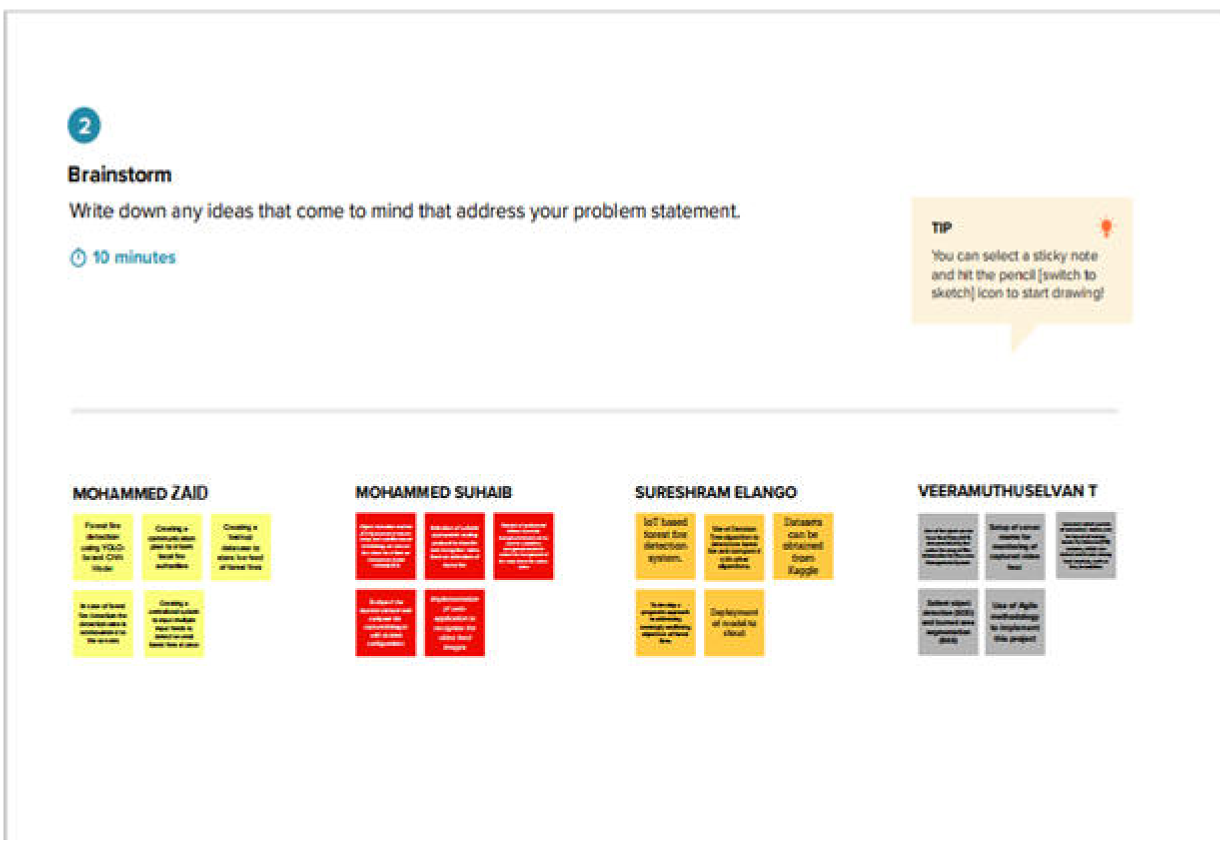
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

3.2 Ideation And Brainstorming

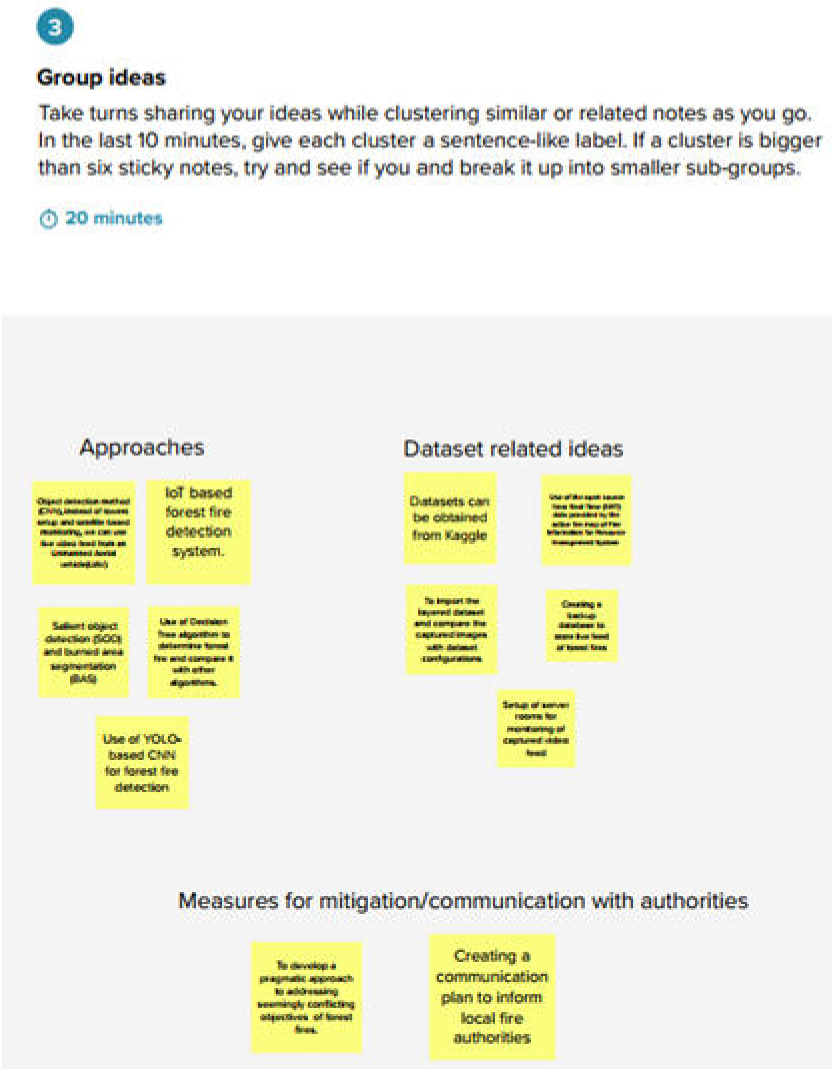
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem-solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate,helping each other develop a rich amount of creative solutions.

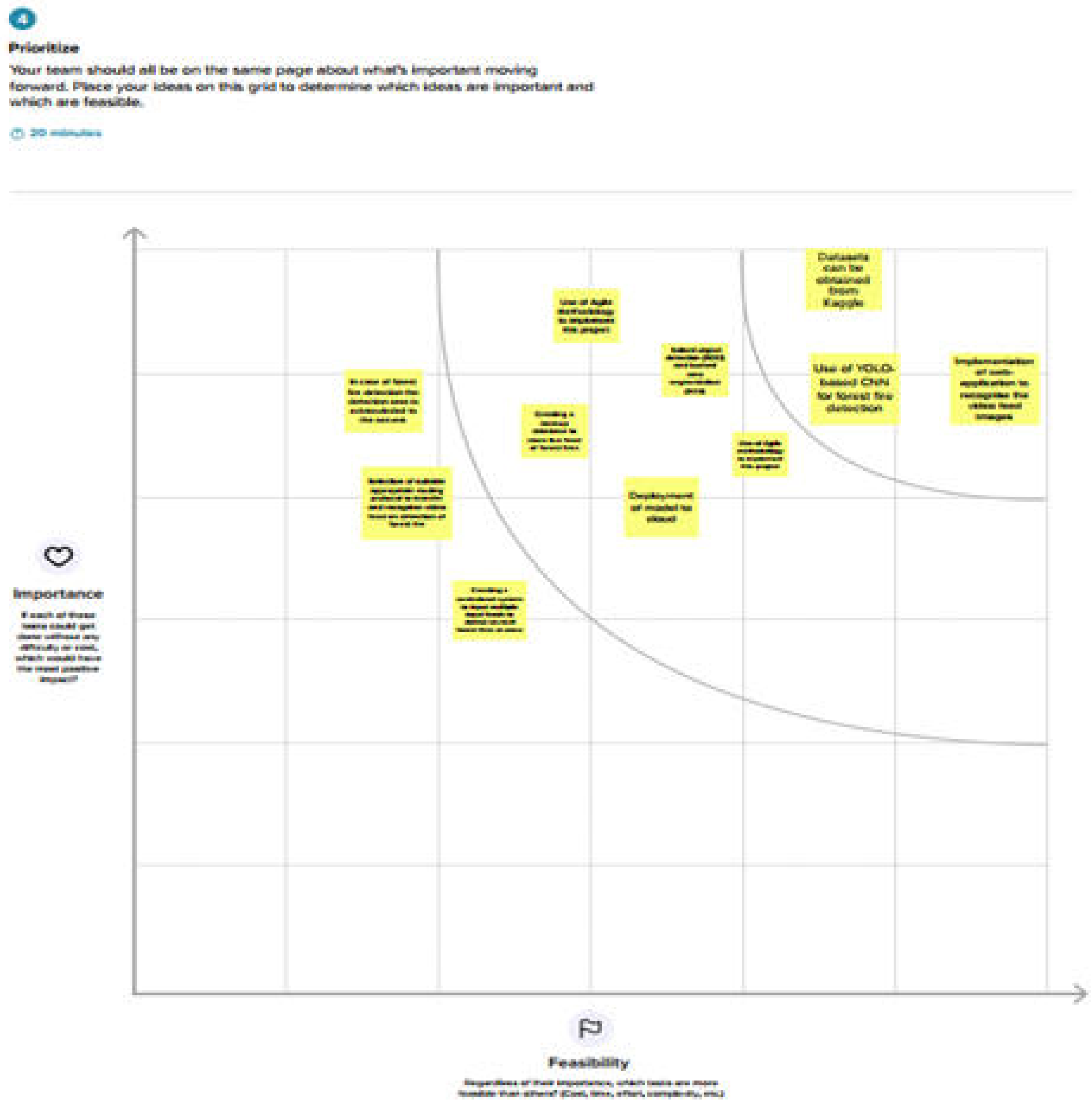


Brainstorm, Idea Listing and Grouping



Idea Prioritization





3.3 Proposed Solution

Proposed Solution:

S.No.	Parameter	Description

1.	Problem Statement (Problem to be solved)	Forest fires occur yearly with increasing intensity in the summer and autumn periods. Regardless of the reasons for the ignition of forest fires, they usually cause devastating damage to both nature and humans. Forest fires are also considered the main contributor to air pollution.
2.	Idea / Solution description	Our solution is to develop a model that uses deep learning algorithms such as CNN, trained to analyse and detect forest fires from image and video data along with computer vision in real-time. The model will predict the regions in which the fires could spread.
3.	Novelty / Uniqueness	The model is then used in unmanned aerial vehicles (UAVs) with specialized cameras to monitor vulnerable regions. A mobile application is developed as an alerting system to notify residents and forest departments once a forest fire is detected. WSNs can be used to monitor parameters that can cause forest fires.

4.	Social Impact / Customer Satisfaction	As the forests are prevented beforehand, huge catastrophes can be prevented such as ecological and economical losses. Habitats of flora and fauna can be conserved. Air pollution can be reduced. The livelihood of residents living in or nearby the forests can be sustained.
5.	Business Model (Revenue Model)	We believe that the mobile application would provide efficient service for the people, forest department, and as well as the government in the long term.
6.	Scalability of the Solution	Sparsely populated areas typically encounter complications during detection. However, the solution can monitor enormous forests and detect forest fires even in sparsely populated regions.

3.4 Problem Solution Fit

Problem-Solution fit canvas 2.0

Project Title : Emerging Methods for Early Detection of Forest Fires

Team ID: PNT2022TMID03442

Explore AS, differentiate

1. CUSTOMER SEGMENT(S) — CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

1.Federal agencies(forest fire management) such as National Disaster Management Authority (NDMA) USDA's Forest Service.

2.The Department of the Interior's Bureau of Indian Affairs, Bureau of Land Management, Fish and Wildlife Service, and National Park Service.

6. CUSTOMER CONSTRAINTS — CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

1.The triple constraint theory says that every project will include three constraints: budget/cost, time, and scope. And these constraints are tied to each other. Any change made to one of the triple constraints will have an effect on the other two.

2.With any project, there are limitations and risks that need to be addressed to ensure the project's ultimate success.

5. AVAILABLE SOLUTIONS — AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

From previous studies the available prototype model uses common sensors like Flame sensor ,temperature sensor, gas sensor for fire detection those sensors are attached to trees animals and birds in the forest to detect the forest fire.

Pros of existing solutions:
1.The forest fire area can be detected and can be located precisely,

Cons of existing solutions:
1.Complicated to manage.
2.Sensor attached to the animals and birds will affect their habitat and the comfortable way of migration

Focus on J&P, tap into

2. JOBS-TO-BE-DONE / PROBLEMS — J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

The process provides broad and detailed customer insights that are superior to typical market research methods and critical to developing better solutions for customers. It helped us understand a new space and identify the underserved needs so we could enter a new market in a differentiated manner

9. PROBLEM ROOT CAUSE — RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

1. The first step when performing root cause analysis is to analyze the existing situations. This is where the team identifies the factors that impact the problematic event. The outcome of this step is a statement that comprises the specific problem A small team is tasked with the definition of the problem. This could be research staff who assesses and analyzes the situation.

2. It describes the difference between the actual conditions and desired conditions.

7. BEHAVIOUR — BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Popular packages encompass processes involved in the maintenance of solar panels and solar power plants. This is critical: you must try to solve the right problem. Don't try to solve a problem the customer sees as low priority or unimportant. Identify the right problem by asking the right questions and observing. You cannot identify the customer's problems by presenting your products.

Focus on J&P, tap int

Identify strong TR & EM

3. TRIGGERS — TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Human-caused fires are the result of abandoned campfires unattended, burning debris, equipment use and malfunctions, discarded due to negligence cigarettes and arson

4. EMOTIONS: BEFORE / AFTER — EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

BEFORE: Encroachment through loss of diversity, reduced wildlife
AFTER :Forest surveillance systems can be used to monitor stress in the forest so we can prevent human and wildlife and economic damage

10. YOUR SOLUTION — SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

In case of forest fire detection the burning substances are primarily identified as sceptical flame regions using a division strategy to expel the non-fire structures and results are verified by a deep learning model. The technology used to locate a forest or a bush fire is based on the concept of deep learning and YOLO algorithm. This deep learning model is deployed on a UAV which helps in detection of fire, meanwhile it can be monitored by web application and the forest fire area can be located in order to prevent it in advance

8. CHANNELS of BEHAVIOUR — CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

Collect the date and form a dataset in order to compare the flames regions for forest fire detection

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

In case of forest fire detection the information is sent to forest authorities so that they will prevent it at ease.

Chapter 4
REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Video/Image surveillance	Capture surveillance through cameras.
FR-2	WSN	Continuous monitoring of forests through sensors.
FR-3	Detection of Fire	Fire is detected via a CNN model and Computer Vision.
FR-4	Cloud	Detected values are sent to the cloud.
FR-5	Alert	Alert the people through a fire alarm system.
FR-6	Mobile app	Users get a notification when the fire is detected.

4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	By detecting the Forest Fire earlier. Alerts according to the user location.
NFR-2	**Security**	This project doesn't contain any secured information so there is no role of security factors. There are no requirements for privacy.
NFR-3	**Reliability**	Since we are using a deep learning algorithm, the system is really good and has better accuracy.
NFR-4	**Performance**	The performance mostly depends on monitoring the forest by WSNs and giving alerts immediately without any delay.
NFR-5	**Availability**	The system shall take real input images of the surveillance camera and it should be helpful in a great way to suppress the fire without any great damage.
NFR-6	**Scalability**	The cost of establishing the cameras for the entire forest may be high. The system can be fitted anywhere in the forest.

Chapter 5

PROJECT DESIGN

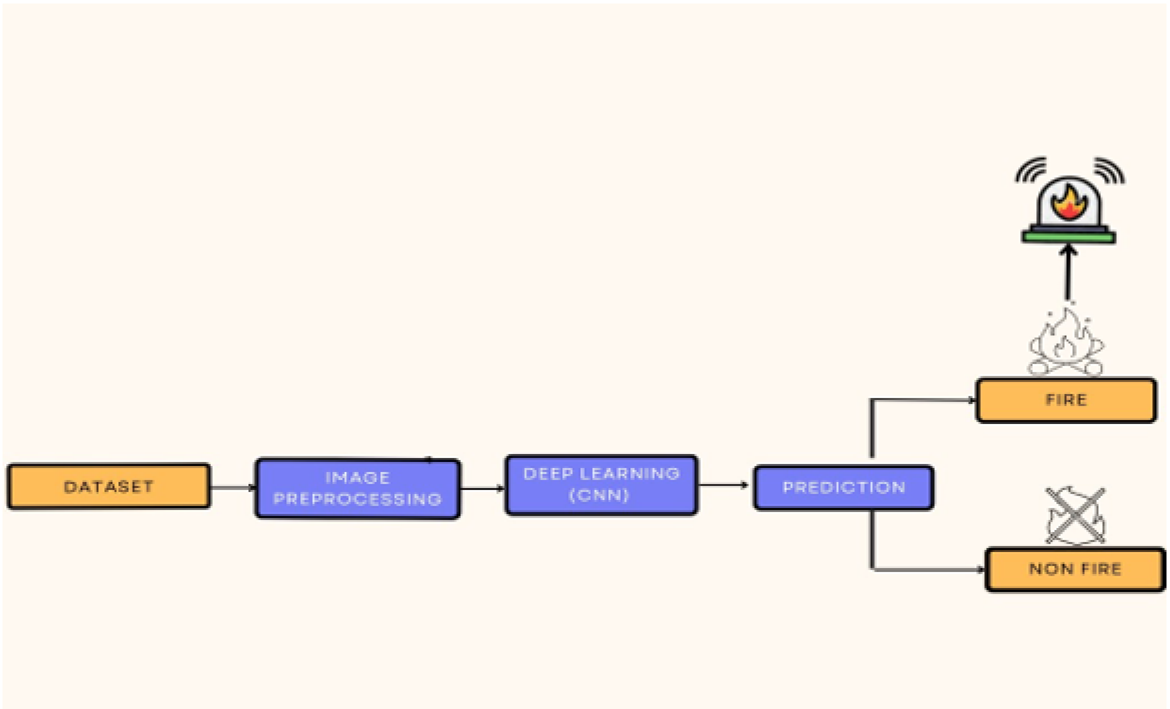
5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

FLOW:

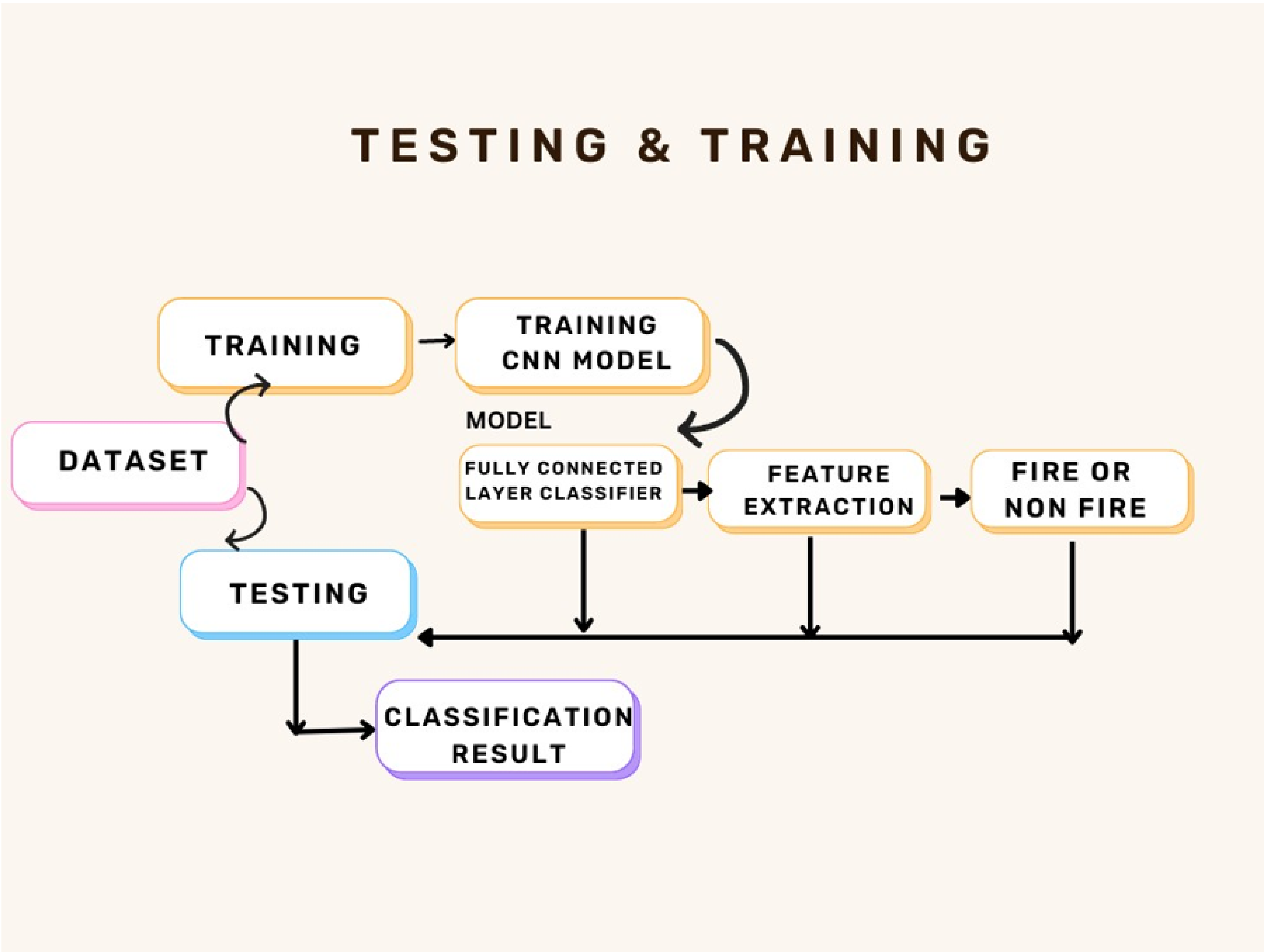
- Data is collected through surveillance video or image-based approaches. The image is preprocessed by using ImageDataGenerator.
- The various real-time forest fire detection and prediction approaches, with the goal of informing the local fire authorities.
- If the fire is not detected, it will send the result to the framing camera.
- If the forest fire is detected, the alert will send notification messages through a mobile app.
- The various real-time forest fire detection and prediction approaches, with the goal of informing the local fire authorities.

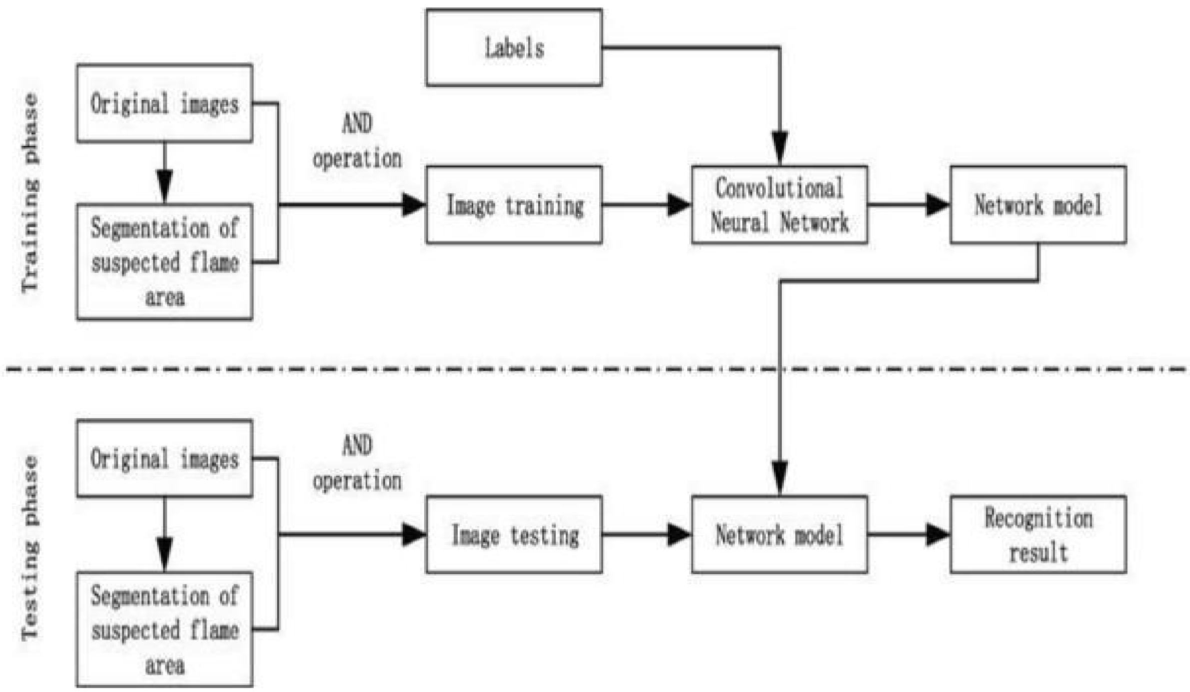
DIAGRAM



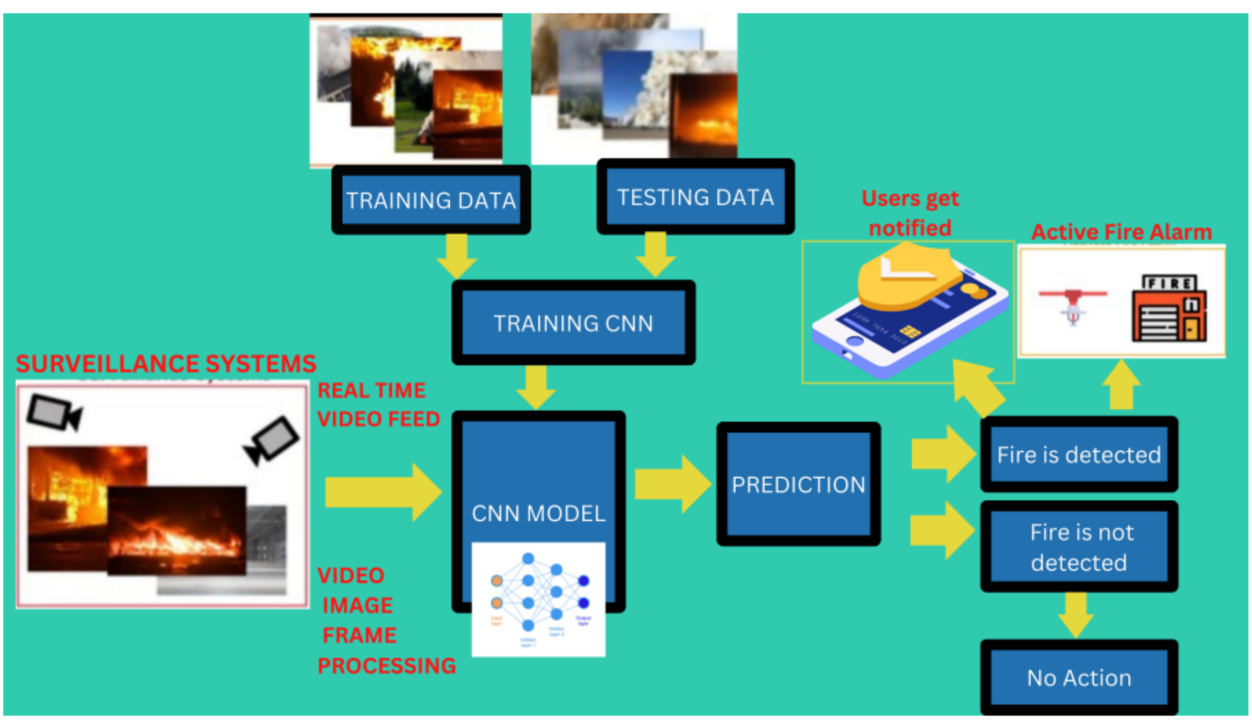
DATASET
IMAGE PREPROCESSING
DEEP LEARNING (CNN)
PREDICTION
FIRE
NON FIRE

5.2 Solution & Technical Architecture:





Solution Architecture:



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Environmental list	Collect the data	USN-1	As an Environmentalist.it is necessary to collect the data of the forest which includes data else the temperature,humidity,wind and rain prediction may of the forest	It is necessary to collect the right data else the prediction may of the forest become wrong	High	Sprint 1
	Preprocessing	USN-2	Dataset is further preprocessed by ImageDataGenerator.	The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features importan	High	Sprint 2

				t for further processi ng.		
	Splitting the dataset	USN-3	The collected dataset is split into train and test.	Separati ng data into training and testing sets is an importan t part of evaluatin g data mining models.	High	Sprin t 3

Chapter 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint -1	Collect the data	USN-1	As an Environmentalist.it is necessary to collect the data of the forest which includes data else the temperature, humidity, wind and rain prediction may of the forest	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -1	Splitting the dataset	USN-2	The collected dataset is split into train and test.	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -1	Image Pre-processing	USN-3	Dataset is further pre-processed by Image Data Generator.	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -2	Model Building	USN-4	Importing the model building libraries, Initializing, the model and adding the CNN and dense	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T

			layers. Configuring the learning process			
Sprint -2	Model Building	USN-5	Training and saving the model	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -2	Model Building	USN-6	Predictions	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -3	Video Analysis	USN-7	OpenCV for Video Processing	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -3	Video Analysis	USN-8	Creating an account in Twilio Service	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T

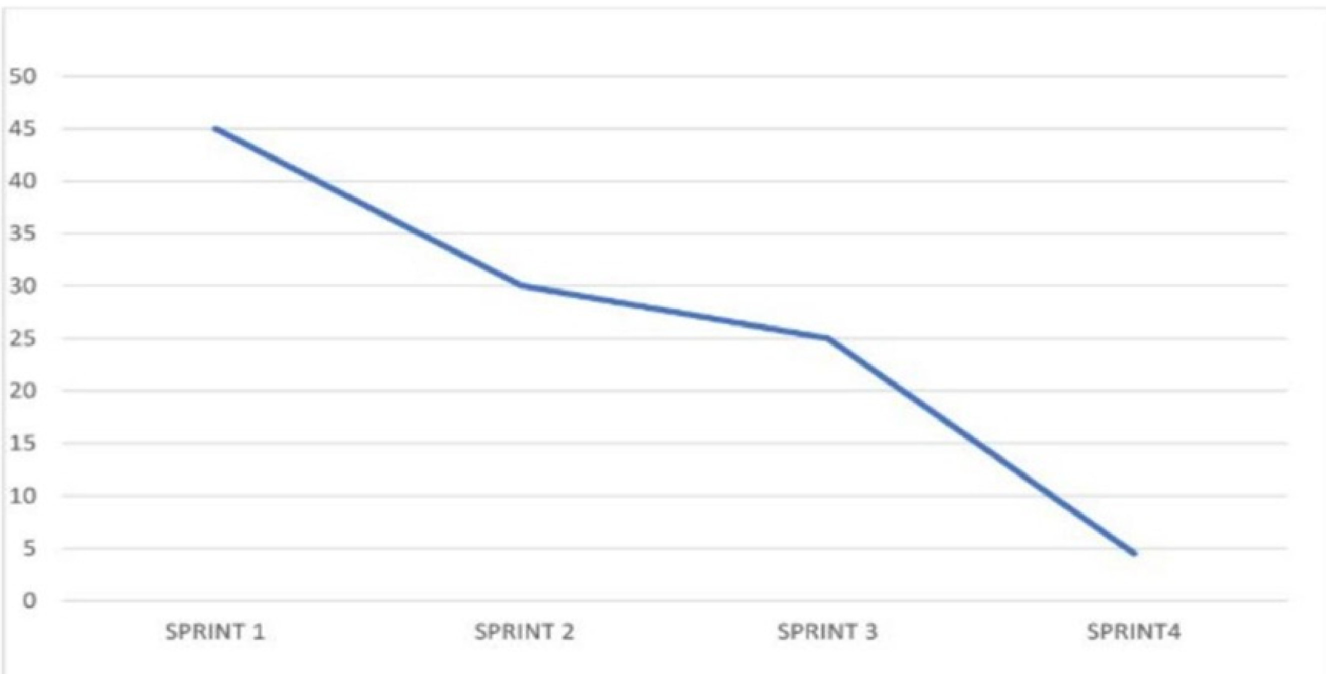
Sprint -3	Video Analysis	USN-9	Sending Alert Message	3	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -4	Training CNN Model on Cloud	USN-10	Registering on Cloud, Train Image Classification Miodel	5	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T
Sprint -4	Implementation	USN-11	Implementation of the model on real-time data	4	High	Mohammed Zaid Mohammed suhaib Sureshram E Veeramuthuselvan T

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	9	2 Days	11 Nov 2022	12 Nov 2022	9	12 Nov 2022
Sprint-2	9	2 Days	13 Nov 2022	14 Nov 2022	9	14 Nov 2022

Sprint-3	9	2 Days	15 Nov 2022	16 Nov 2022	9	16 Nov 2022
Sprint-4	9	2 Days	17 Nov 2022	18 Nov 2022	9	18 Nov 2022

6.3 Reports from JIRA



Chapter 7

CODING & SOLUTIONING

7.1 Feature 1

1. Preprocessing the dataset which consists of two classes of data(fire, no fire).

Image Preprocessing

#1. Importing the ImageDataGenerator Library

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

#2. Define parameters for ImageDataGenerator Class

```
train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)
#rescale => rescaling pixel value from 0 to 255 to 0 to 1
#shear_range=> counter clock wise rotation(anti clock)
test_datagen = ImageDataGenerator(rescale=1./255)
```

#3. Applying ImageDataGenerator Functionality to Trainset and Testset

```
x_train = train_datagen.flow_from_directory(r"D:\FFDDataset\train_set",
                                            target_size=(256,256),
                                            batch_size=32,
                                            class_mode="binary")
x_test = test_datagen.flow_from_directory(r"D:\FFDDataset\test_set",
                                          target_size=(256,256),
                                          batch_size=32,
                                          class_mode="binary")
```

Applying ImageDataGenerator functionality to train dataset

```
x_train = train_datagen.flow_from_directory(r"D:\FFDDataset\train_set",
                                            target_size=(256,256),
                                            batch_size=32,
                                            class_mode="binary")
```

```
Found 436 images belonging to 2 classes.
```

Applying ImageDataGenerator functionality to test dataset

```
x_test = test_datagen.flow_from_directory(r"D:\FFDDataset\test_set",
                                          target_size=(256,256),
                                          batch_size=32,
                                          class_mode="binary")
```

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

Building the Model

2.Building up a sequential model to train the dataset.

 # 1.Importing the Model Building Libraries

#Importing model libraries

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

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

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

from tensorflow.keras.optimizers **import** Adam , SGD, RMSprop

 # 2.Initializing the Model

model=Sequential()

 # 3.Adding CNN Layers

a. Adding Convolutional layer

model.add(Convolution2D(32,(3,3),input_shape=(256,256,3),activation="relu"))

b. Adding Pooling Layer

model.add(MaxPooling2D(pool_size=(2,2)))

c. Adding Flatten Layer

model.add(Flatten)

model.add(Flatten())

#Summary of model

model.summary()

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 flatten (Flatten)           (None, 516128)            0

 dense (Dense)               (None, 300)               154838700

 dense_1 (Dense)             (None, 200)               60200

 dense_2 (Dense)             (None, 1)                 201

=================================================================
Total params: 154,899,997
Trainable params: 154,899,997
Non-trainable params: 0
_________________________________________________________________
```

Prediction of data

from tensorflow.keras.models **import** load_model

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

model **=** load_model("fire.h5")

```
img = image.load_img(r"C:\Users\Isha\Pictures\Saved Pictures\egnofire.jpg",target_size=(256,256))
```

```
img
```

```
type(img)
```

```
PIL.Image.Image
```

```
x = image.img_to_array(img)
```

```
x
```

```
array([[[ 12.,  14.,   0.],
        [ 21.,  24.,   7.],
        [ 43.,  46.,  27.],
        ...,
```

```
 [ 21.,  19.,   7.],
 [ 13.,  15.,   2.],
 [ 52.,  60.,  11.]],

[[ 13.,  15.,   2.],
 [ 12.,  14.,   0.],
 [ 18.,  21.,   4.],
 ...,
 [ 17.,  15.,   2.],
 [ 10.,  11.,   3.],
 [ 58.,  65.,  23.]],

[[ 14.,  15.,   7.],
 [ 11.,  13.,   2.],
 [ 10.,  12.,   0.],
[ 19.,  18.,   0.],
 [ 17.,  18.,  13.],
 [ 62.,  66.,  39.]],

...,

[[ 14.,  15.,   7.],
 [ 51.,  56.,  26.],
 [ 48.,  57.,   2.],
 ...,
 [ 50.,  65.,  26.],
 [ 58.,  75.,  30.],
 [ 54.,  73.,  27.]],

[[ 17.,  19.,   8.],
 [ 49.,  54.,  24.],
 [103., 112.,  57.],
 ...,
 [ 65.,  80.,  41.],
 [ 61.,  78.,  33.],
 [ 64.,  83.,  37.]],
```

```
       [[ 18.,  18.,   8.],
        [ 36.,  39.,   8.],
        [ 77.,  85.,  28.],
        ...,
        [ 79.,  94.,  55.],
        [ 50.,  67.,  22.],
        [ 52.,  71.,  25.]]], dtype=float32)
```

```
 x.shape
```

```
(256, 256, 3)
```

```
import numpy as np


# convolution expects 4D
x = np.expand_dims(x,axis=0)


x.shape
```

```
(1, 256, 256, 3)
```

```
pred_prob = model.predict(x)
```

```
1/1 [==============================] - 0s 111ms/step
```

```
 pred_prob
```

```
array([[0.]], dtype=float32)
```

```
if(pred_prob==0):
   print("There is no fire")
else:
   print("There is a fire")
```

```
There is no fire
```

4. Accurancy:

Epoch 1/30

13/13 [==============================] - 36s 2s/step - loss: 2.2809 - accuracy: 0.5965 - val_loss: 0.6385 - val_accuracy: 0.5938

Epoch 2/30

13/13 [==============================] - 53s 4s/step - loss: 0.4557 - accuracy: 0.7822 - val_loss: 0.1618 - val_accuracy: 0.9062

Epoch 3/30

13/13 [==============================] - 44s 3s/step - loss: 0.2581 - accuracy: 0.8762 - val_loss: 0.0857 - val_accuracy: 0.9688

Epoch 4/30

13/13 [==============================] - 28s 2s/step - loss: 0.2146 - accuracy: 0.9059 - val_loss: 0.1209 - val_accuracy: 0.9688

Epoch 5/30

13/13 [==============================] - 31s 2s/step - loss: 0.1683 - accuracy: 0.9332 - val_loss: 0.0789 - val_accuracy: 0.9688

Epoch 6/30

13/13 [==============================] - 34s 3s/step - loss: 0.1468 - accuracy: 0.9381 - val_loss: 0.0531 - val_accuracy: 0.9896

Epoch 7/30

13/13 [==============================] - 35s 3s/step - loss: 0.1569 - accuracy: 0.9406 - val_loss: 0.1668 - val_accuracy: 0.9375

Epoch 8/30

13/13 [==============================] - 36s 3s/step - loss: 0.1830 - accuracy: 0.9158 - val_loss: 0.0514 - val_accuracy: 0.9896

Epoch 9/30

13/13 [==============================] - 32s 2s/step - loss: 0.1455 - accuracy: 0.9356 - val_loss: 0.0378 - val_accuracy: 0.9896

Epoch 10/30

13/13 [==============================] - 34s 3s/step - loss: 0.1761 - accuracy: 0.9307 - val_loss: 0.0352 - val_accuracy: 1.0000

Epoch 11/30

13/13 [==============================] - 35s 3s/step - loss: 0.1391 - accuracy: 0.9530 - val_loss: 0.0413 - val_accuracy: 0.9896

Epoch 12/30

13/13 [==============================] - 37s 3s/step - loss: 0.1264 - accuracy: 0.9505 - val_loss: 0.0580 - val_accuracy: 0.9792

Epoch 13/30

13/13 [==============================] - 34s 3s/step - loss: 0.1306 - accuracy: 0.9406 - val_loss: 0.0191 - val_accuracy: 1.0000

Epoch 14/30

13/13 [==============================] - 35s 3s/step - loss: 0.1083 - accuracy: 0.9554 - val_loss: 0.0361 - val_accuracy: 0.9792

Epoch 15/30

13/13 [==============================] - 35s 3s/step - loss: 0.0869 - accuracy: 0.9678 - val_loss: 0.0203 - val_accuracy: 0.9896

Epoch 16/30

13/13 [==============================] - 31s 2s/step - loss: 0.1200 - accuracy: 0.9579 - val_loss: 0.0275 - val_accuracy: 0.9896

Epoch 17/30

13/13 [==============================] - 31s 2s/step - loss: 0.1556 - accuracy: 0.9233 - val_loss: 0.0402 - val_accuracy: 0.9896

Epoch 18/30

13/13 [==============================] - 33s 3s/step - loss: 0.1405 - accuracy: 0.9406 - val_loss: 0.0595 - val_accuracy: 0.97

Epoch 19/30

13/13 [==============================] - 34s 3s/step - loss: 0.1334 - accuracy: 0.9356 - val_loss: 0.0559 - val_accuracy: 0.9896

Epoch 20/30

13/13 [==============================] - 33s 3s/step - loss: 0.1130 - accuracy: 0.9530 - val_loss: 0.0251 - val_accuracy: 0.9896

Epoch 21/30

13/13 [==============================] - 39s 3s/step - loss: 0.1073 - accuracy: 0.9406 - val_loss: 0.0313 - val_accuracy: 0.9896

Epoch 22/30

13/13 [==============================] - 29s 2s/step - loss: 0.1091 - accuracy: 0.9480 - val_loss: 0.0170 - val_accuracy: 1.0000

Epoch 23/30

13/13 [==============================] - 30s 2s/step - loss: 0.0939 - accuracy: 0.9567 - val_loss: 0.0128 - val_accuracy: 1.0000

Epoch 24/30

13/13 [==============================] - 29s 2s/step - loss: 0.0759 - accuracy: 0.9728 - val_loss: 0.0037 - val_accuracy: 1.0000

Epoch 25/30

13/13 [==============================] - 35s 3s/step - loss: 0.0758 - accuracy: 0.9777 - val_loss: 0.0118 - val_accuracy: 1.0000

Epoch 26/30

13/13 [==============================] - 34s 3s/step - loss: 0.0707 - accuracy: 0.9802 - val_loss: 0.0079 - val_accuracy: 1.0000

Epoch 27/30

13/13 [==============================] - 36s 3s/step - loss: 0.1081 - accuracy: 0.9480 - val_loss: 0.0235 - val_accuracy: 0.9896

Epoch 28/30

13/13 [==============================] - 35s 3s/step - loss: 0.0975 - accuracy: 0.9678 - val_loss: 0.0092 - val_accuracy: 1.0000

Epoch 29/30

13/13 [==============================] - 34s 3s/step - loss: 0.0746 - accuracy: 0.9752 - val_loss: 0.0072 - val_accuracy: 1.0000

Epoch 30/30

13/13 [==============================] - 35s 3s/step - loss: 0.0695 - accuracy: 0.9777 - val_loss: 0.0720 - val_accuracy: 0.9583

Training and Deploying the model in cloud

```
pwd

'/home/wsuser/work'

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='EHmhit2MD64AQnqYijN7mrXyaEYoh02jLsiuzU5mzGbt',

    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 = 'ffdcnnmodelbook-donotdelete-pr-giva0vdmx0opfa'

object_key = 'forestfiredataset.zip'

streaming_body_3 = 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_3.read()),'r')

file_paths=unzip.namelist()

for path in file_paths:

    unzip.extract(path)

ls

Dataset/                    fire-classification-model.tgz  forest1.h5

fie-classification-model.tgz  fire.h5
```

Import the libraries

```
import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from matplotlib import pyplot as plt
```

Importing ImageDataGenerator from Keras

```
# image preprocessing (or) image augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator

#import the cnn layers
```

Defining the Parameters

train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)

#rescale => rescaling pixel value from 0 to 255 to 0 to 1

#shear_range=> counter clock wise rotation(anti clock)

test_datagen = ImageDataGenerator(rescale=1./255)

Applying ImageDataGenerator functionality to train dataset

```
x_train = train_datagen.flow_from_directory(r"/home/wsuser/work/Dataset/Dataset/train_set",
                                            target_size=(256,256),
                                            batch_size=32,
                                            class_mode="binary")
```

Found 436 images belonging to 2 classes.

Applying ImageDataGenerator functionality to test dataset

```
x_test = test_datagen.flow_from_directory(r"/home/wsuser/work/Dataset/Dataset/test_set",
                                          target_size=(256,256),
                                          batch_size=32,
                                          class_mode="binary")
```

Found 121 images belonging to 2 classes.

Importing Model Building Libraries

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.optimizers import Adam , SGD, RMSprop

x_train.class_indices

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

Intializing the model

model = Sequential()

Adding CNN layers

add convolution layer

model.add(Convolution2D(32,(3,3),input_shape=(256,256,3),activation="relu"))

32 indicates => no of feature detectors

#(3,3)=> kernel size (feature detector size)

#add max pooling layer

model.add(MaxPooling2D(pool_size=(2,2)))

#add flatten layer => input to your ANN

model.add(Flatten())

Add Dense layers

#hidden layer

model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))

model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))

#output layer

model.add(Dense(units=1,kernel_initializer="random_uniform",activation="sigmoid"))

Configuring the learning process

#compile the model

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

Summarize the model

model.summary()

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 flatten (Flatten)           (None, 516128)            0

 dense (Dense)               (None, 300)               154838700

 dense_1 (Dense)             (None, 200)               60200

 dense_2 (Dense)             (None, 1)                 201

=================================================================
Total params: 154,899,997
Trainable params: 154,899,997
Non-trainable params: 0
_________________________________________________________________
```

Training the model

model.fit(x_train,steps_per_epoch=13,epochs=30,validation_data=x_test,validation_steps=3)

#steps_per_epoch = no of training images/batch size

#validation_steps = no of testing images/batch size

```
Epoch 1/30
13/13 [==============================] - 49s 4s/step - loss: 1.8251 - accuracy: 0.6485 - val_loss: 0.2524 - val_accuracy: 0.8958
Epoch 2/30
13/13 [==============================] - 52s 4s/step - loss: 0.2757 - accuracy: 0.8726 - val_loss: 0.1387 - val_accuracy: 0.9479
Epoch 3/30
13/13 [==============================] - 52s 4s/step - loss: 0.3054 - accuracy: 0.8663 - val_loss: 0.0653 - val_accuracy: 0.9792
Epoch 4/30
13/13 [==============================] - 52s 4s/step - loss: 0.2152 - accuracy: 0.9084 - val_loss: 0.0805 - val_accuracy: 0.9896
Epoch 5/30
13/13 [==============================] - 47s 4s/step - loss: 0.1913 - accuracy: 0.9233 - val_loss: 0.1705 - val_accuracy: 0.9375
Epoch 6/30
13/13 [==============================] - 51s 4s/step - loss: 0.2007 - accuracy: 0.9158 - val_loss: 0.0850 - val_accuracy: 0.9688
Epoch 7/30
13/13 [==============================] - 51s 4s/step - loss: 0.1476 - accuracy: 0.9455 - val_loss: 0.0729 - val_accuracy: 0.9792
Epoch 8/30
13/13 [==============================] - 50s 4s/step - loss: 0.1483 - accuracy: 0.9356 - val_loss: 0.0579 - val_accuracy: 0.9792
Epoch 9/30
13/13 [==============================] - 50s 4s/step - loss: 0.1606 - accuracy: 0.9282 - val_loss: 0.1238 - val_accuracy: 0.9688
Epoch 10/30
13/13 [==============================] - 50s 4s/step - loss: 0.1764 - accuracy: 0.9158 - val_loss: 0.1050 - val_accuracy: 0.9688
Epoch 11/30
13/13 [==============================] - 52s 4s/step - loss: 0.1448 - accuracy: 0.9406 - val_loss: 0.0601 - val_accuracy: 0.9792
Epoch 12/30
13/13 [==============================] - 50s 4s/step - loss: 0.1229 - accuracy: 0.9554 - val_loss: 0.0309 - val_accuracy: 0.9896
Epoch 13/30
13/13 [==============================] - 53s 4s/step - loss: 0.1220 - accuracy: 0.9579 - val_loss: 0.0533 - val_accuracy: 0.9896
Epoch 14/30
13/13 [==============================] - 52s 4s/step - loss: 0.1291 - accuracy: 0.9455 - val_loss: 0.0525 - val_accuracy: 0.9792
Epoch 15/30
13/13 [==============================] - 50s 4s/step - loss: 0.1065 - accuracy: 0.9554 - val_loss: 0.0221 - val_accuracy: 0.9896
```

```
Epoch 16/30
13/13 [==============================] - 50s 4s/step - loss: 0.1161 - accuracy: 0.9554 - val_loss: 0.0206 - val_accuracy: 1.0000
Epoch 17/30
13/13 [==============================] - 52s 4s/step - loss: 0.1607 - accuracy: 0.9356 - val_loss: 0.0258 - val_accuracy: 0.9896
Epoch 18/30
13/13 [==============================] - 48s 4s/step - loss: 0.1090 - accuracy: 0.9629 - val_loss: 0.0293 - val_accuracy: 0.9896
Epoch 19/30
13/13 [==============================] - 51s 4s/step - loss: 0.1500 - accuracy: 0.9332 - val_loss: 0.0269 - val_accuracy: 0.9896
Epoch 20/30
13/13 [==============================] - 50s 4s/step - loss: 0.1445 - accuracy: 0.9431 - val_loss: 0.0187 - val_accuracy: 1.0000
Epoch 21/30
13/13 [==============================] - 51s 4s/step - loss: 0.1292 - accuracy: 0.9530 - val_loss: 0.0313 - val_accuracy: 0.9792
Epoch 22/30
13/13 [==============================] - 50s 4s/step - loss: 0.1079 - accuracy: 0.9554 - val_loss: 0.0496 - val_accuracy: 0.9792
Epoch 23/30
13/13 [==============================] - 51s 4s/step - loss: 0.1115 - accuracy: 0.9554 - val_loss: 0.0274 - val_accuracy: 0.9792
Epoch 24/30
13/13 [==============================] - 51s 4s/step - loss: 0.0999 - accuracy: 0.9579 - val_loss: 0.0221 - val_accuracy: 0.9896
Epoch 25/30
13/13 [==============================] - 52s 4s/step - loss: 0.0801 - accuracy: 0.9752 - val_loss: 0.0125 - val_accuracy: 0.9896
Epoch 26/30
13/13 [==============================] - 51s 4s/step - loss: 0.0761 - accuracy: 0.9736 - val_loss: 0.0234 - val_accuracy: 0.9792
Epoch 27/30
13/13 [==============================] - 51s 4s/step - loss: 0.0825 - accuracy: 0.9752 - val_loss: 0.0092 - val_accuracy: 1.0000
Epoch 28/30
13/13 [==============================] - 52s 4s/step - loss: 0.0738 - accuracy: 0.9703 - val_loss: 0.0167 - val_accuracy: 0.9896
Epoch 29/30
13/13 [==============================] - 51s 4s/step - loss: 0.0780 - accuracy: 0.9663 - val_loss: 0.0024 - val_accuracy: 1.0000
Epoch 30/30
13/13 [==============================] - 51s 4s/step - loss: 0.1051 - accuracy: 0.9455 - val_loss: 0.0151 - val_accuracy: 1.0000
```

Saving the model

model.save("fire.h5")

IBM Deployment

!pip install watson-machine-learning-client

```
Requirement already satisfied: watson-machine-learning-client in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.391)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.9.24)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-cl
ient) (0.10.0)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-
client) (0.5.0)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning
-client) (1.21.41)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->b
oto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>=
1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-lea
rning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machi
ne-learning-client) (2.11.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client)
(3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learn
ing-client) (2.0.4)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (20
21.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client)
(1.20.3)
```

```
from ibm_watson_machine_learning import APIClient

wml_credentials={

    "url":"https://us-south.ml.cloud.ibm.com",

    "apikey":"1AfypwQwqeHikzD7u4LIKT6DMnD-RPDTyYLRBofzNBPp"

}


client=APIClient(wml_credentials)

client

def guid_space_name(client,fire_deploy):

    space=client.spaces.get_details()

    return(next(item for item in space['resources'] if item['entity']['name']==fire_deploy)['metadata']['id'])

space_uid=guid_space_name(client,'cnn_fire')

print("Space UID "+space_uid)
```

Space UID def3a2d0-3dd4-4f16-9ba5-cb9feb7700a1

client.set.default_space(space_uid)

'SUCCESS'

client.software_specifications.list(200)

```
------------------------------  -----------------------------------  ----
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
runtime-22.2-py3.10-xc	5e8cddff-db4a-5a6a-b8aa-2d4af9864dab	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
spark-mllib_2.3-r_3.6	6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c	base

tensorflow_2.4-py3.7	65e171d7-72d1-55d9-8ebb-f813d620c9bb	base
spss-modeler_18.2	687eddc9-028a-4117-b9dd-e57b36f1efa5	base
pytorch-onnx_1.2-py3.6	692a6a4d-2c4d-45ff-a1ed-b167ee55469a	base
spark-mllib_2.3-scala_2.11	7963efe5-bbec-417e-92cf-0574e21b4e8d	base
spark-mllib_2.4-py37	7abc992b-b685-532b-a122-a396a3cdbaab	base
caffe_1.0-py3.6	7bb3dbe2-da6e-4145-918d-b6d84aa93b6b	base
pytorch-onnx_1.7-py3.7	812c6631-42b7-5613-982b-02098e6c909c	base
cuda-py3.6	82c79ece-4d12-40e6-8787-a7b9e0f62770	base
tensorflow_1.15-py3.6-horovod	8964680e-d5e4-5bb8-919b-8342c6c0dfd8	base
hybrid_0.1	8c1a58c6-62b5-4dc4-987a-df751c2756b6	base
pytorch-onnx_1.3-py3.7	8d5d8a87-a912-54cf-81ec-3914adaa988d	base
caffe-ibm_1.0-py3.6	8d863266-7927-4d1e-97d7-56a7f4c0a19b	base
runtime-22.2-py3.10-cuda	8ef391e4-ef58-5d46-b078-a82c211c1058	base
spss-modeler_17.1	902d0051-84bd-4af6-ab6b-8f6aa6fdeabb	base
do_12.10	9100fd72-8159-4eb9-8a0b-a87e12eefa36	base
do_py3.7	9447fa8b-2051-4d24-9eef-5acb0e3c59f8	base
spark-mllib_3.0-r_3.6	94bb6052-c837-589d-83f1-f4142f219e32	base
cuda-py3.7-opence	94e9652b-7f2d-59d5-ba5a-23a414ea488f	base
nlp-py3.8	96e60351-99d4-5a1c-9cc0-473ac1b5a864	base
cuda-py3.7	9a44990c-1aa1-4c7d-baf8-c4099011741c	base
hybrid_0.2	9b3f9040-9cee-4ead-8d7a-780600f542f7	base
spark-mllib_3.0-py38	9f7a8fc1-4d3c-5e65-ab90-41fa8de2d418	base
autoai-kb_3.3-py3.7	a545cca3-02df-5c61-9e88-998b09dc79af	base
spark-mllib_3.0-py39	a6082a27-5acc-5163-b02c-6b96916eb5e0	base
runtime-22.1-py3.9-do	a7e7dbf1-1d03-5544-994d-e5ec845ce99a	base
default_py3.8	ab9e1b80-f2ce-592c-a7d2-4f2344f77194	base
tensorflow_rt22.1-py3.9	acd9c798-6974-5d2f-a657-ce06e986df4d	base
kernel-spark3.2-py3.9	ad7033ee-794e-58cf-812e-a95f4b64b207	base
autoai-obm_2.0 with Spark 3.0	af10f35f-69fa-5d66-9bf5-acb58434263a	base

```
runtime-22.2-py3.10            b56101f1-309d-549b-a849-eaa63f77b2fb  base
default_py3.7_opence           c2057dd4-f42c-5f77-a02f-72bdbd3282c9  base
tensorflow_2.1-py3.7           c4032338-2a40-500a-beef-b01ab2667e27  base
do_py3.7_opence                cc8f8976-b74a-551a-bb66-6377f8d865b4  base
spark-mllib_3.3                d11f2434-4fc7-58b7-8a62-755da64fdaf8  base
autoai-kb_3.0-py3.6            d139f196-e04b-5d8b-9140-9a10ca1fa91a  base
spark-mllib_3.0-py36           d82546d5-dd78-5fbb-9131-2ec309bc56ed  base
autoai-kb_3.4-py3.8            da9b39c3-758c-5a4f-9cfd-457dd4d8c395  base
kernel-spark3.2-r3.6           db2fe4d6-d641-5d05-9972-73c654c60e0a  base
autoai-kb_rt22.1-py3.9         db6afe93-665f-5910-b117-d879897404d9  base
tensorflow_rt22.1-py3.9-horovod  dda170cc-ca67-5da7-9b7a-cf84c6987fae  base
autoai-ts_1.0-py3.7            deef04f0-0c42-5147-9711-89f9904299db  base
tensorflow_2.1-py3.7-horovod   e384fce5-fdd1-53f8-bc71-11326c9c635f  base
default_py3.7                  e4429883-c883-42b6-87a8-f419d64088cd  base
do_22.1                        e51999ba-6452-5f1f-8287-17228b88b652  base
autoai-obm_3.2                 eae86aab-da30-5229-a6a6-1d0d4e368983  base
runtime-22.2-r4.2              ec0a3d28-08f7-556c-9674-ca7c2dba30bd  base
tensorflow_rt22.2-py3.10       f65bd165-f057-55de-b5cb-f97cf2c0f393  base
do_20.1                        f686cdd9-7904-5f9d-a732-01b0d6b10dc5  base
pytorch-onnx_rt22.2-py3.10-edt  f8a05d07-e7cd-57bb-a10b-23f1d4b837ac  base
scikit-learn_0.19-py3.6        f963fa9d-4bb7-5652-9c5d-8d9289ef6ad9  base
tensorflow_2.4-py3.8           fe185c44-9a99-5425-986b-59bd1d2eda46  base
------------------------------  ------------------------------------  ----
```

```
software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
software_space_uid
```

'acd9c798-6974-5d2f-a657-ce06e986df4d'

```
ls
```

```
Dataset/                      fire-classification-model.tgz  forest1.h5
fie-classification-model.tgz  fire.h5
```

```
!tar -zcvf fire-classification-model.tgz fire.h5
fire.h5
model_details=client.repository.store_model(model='fire-classification-model.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"CNN Model Building",
    client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
})
model_id=client.repository.get_model_id(model_details)
model_id
'babd0250-5274-4923-850c-7fe9ce7e2409'
client.repository.download(model_id,'fire.tar.gb')
Successfully saved model content to file: 'fire.tar.gb'
'/home/wsuser/work/fire.tar.gb'
ls
Dataset/                    fire-classification-model.tgz  fire.tar.gb
fie-classification-model.tgz  fire.h5                    forest1.h5
```

7.2 Feature 2

1.Creation of twilio account

To send an outgoing SMS message from your Twilio account you'll need to make an HTTP POST to Twilio's Message resource.

Twilio's Python library helps you to create a new instance of the Message resource, specifying the To, From, and Body parameters of your message.

Replace the placeholder values for account_sid and auth_token with your unique values. You can find these in your Twilio console.

You’ll tell Twilio which phone number to use to send this message by replacing the from_ number with the Twilio phone number you purchased earlier.

Next, specify yourself as the message recipient by replacing the to number with your mobile phone number. Both of these parameters must use E.164 formatting (+ and a country code, e.g., +16175551212)

We also include the body parameter, which contains the content of the SMS we’re going to send.

Ahoy Mohammed, welcome to Twilio!

Learn to build your first SMS app by following these steps.

To send or receive an SMS with Twilio, you will need a virtual phone number from Twilio. A virtual phone number is a standard telephone number that is not locked down to a specific phone. It can route a voice call or text message to any phone or application workflow. In addition, you will need Twilio account SID and Auth token to connect Twilio with your application.

While your account is in trial, you can get one free USA or Canadian phone number. To get local phone numbers outside of the USA or Canada, you may need to upgrade your account and meet regulatory requirement

You’ve got a phone number!
View it in Account info below. You can also find your Twilio account SID and auth token in Account info.

Account Info

Account SID

AC74a73227a4fa4c514205086263a7dba7

Auth Token

••••••••••••••••••••••••••••••••

Show

Always store your token securely to protect your account. Learn more

My Twilio phone number

+19855455097

You are on a trial account. You can only send messages and make calls to verified phone numbers.
Learn more about your trial account

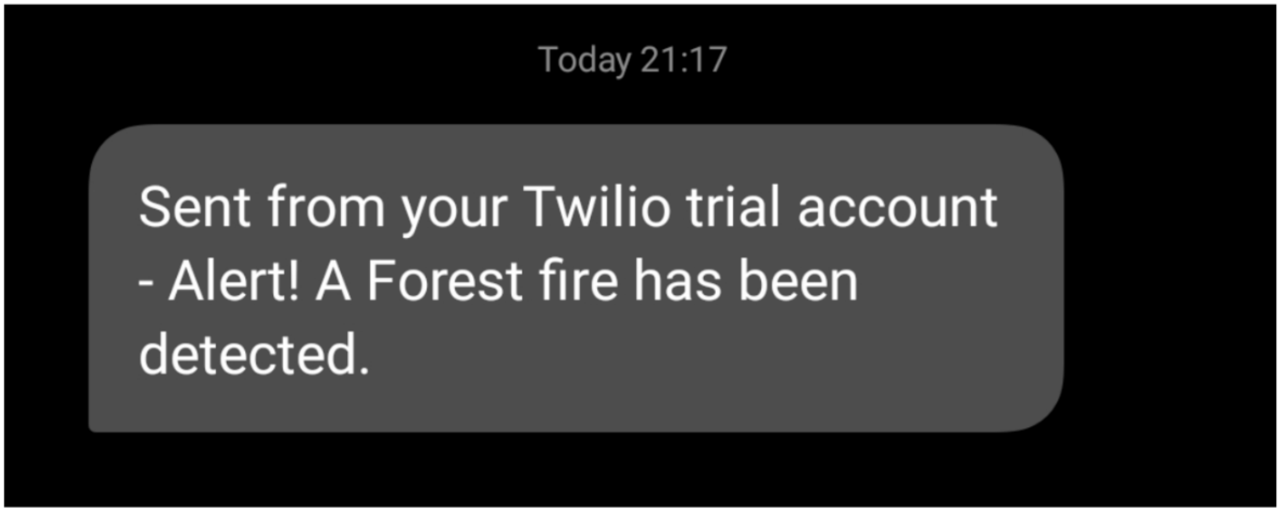
2. Sending SMS alert 👍

OpenCV for Video Processing

```
import cv2
import numpy as np
# importing image function from keras
from keras.preprocessing import image
# importing load_model from keras
from keras.models import load_model
#importing client from twilio API
from twilio.rest import Client
#importing playsound package from playsound
import playsound
```

```
#loading the saved model
model = load_model("fire.h5")
#define video
video = cv2.VideoCapture(0)
#defining the features
name = ['Forest','With fire']
```

Notification through SMS



Chapter 8

TESTING

8.1 Test Cases

Panel switches and keypads: TEST the operation of each control.

Visual indicators: TEST the operation of each visual indicator and alphanumeric display.

Battery: MEASURE system quiescent and maximum alarm currents in accordance with Appendix. Calculate the required battery capacity and CHECK the nominal capacity of the installed batteries is not less than the calculated capacity.

Verify that the measured currents are the same as recorded in the baseline data.

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	9	5	1	2	17
Duplicate	1	0	2	0	3
External	3	3	0	1	7
Fixed	10	2	3	20	35

Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	4	2	1	7
Totals	13	15	10	25	7 2

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	53	0	0	53
Security	2	0	0	2
Outsource Shipping	4	0	0	4
Exception Reporting	7	0	0	7
Final Report Output	3	0	0	3
Version Control	1	0	0	1

Chapter 9

RESULTS

9.1 Performance Metrics

1. Training the Model

Training the model

```
model.fit(x_train,steps_per_epoch=13,epochs=30,validation_data=x_test,validation_steps=3)
#steps_per_epoch = no of training images/batch size
#validation_steps = no of testing images/batch size
```

```
Epoch 1/30
13/13 [==============================] - 36s 2s/step - loss: 2.2809 - accuracy: 0.5965 - val_loss: 0.6385 - val_accuracy: 0.5938
Epoch 2/30
13/13 [==============================] - 53s 4s/step - loss: 0.4557 - accuracy: 0.7822 - val_loss: 0.1618 - val_accuracy: 0.9062
Epoch 3/30
13/13 [==============================] - 44s 3s/step - loss: 0.2581 - accuracy: 0.8762 - val_loss: 0.0857 - val_accuracy: 0.9688
Epoch 4/30
13/13 [==============================] - 28s 2s/step - loss: 0.2146 - accuracy: 0.9059 - val_loss: 0.1209 - val_accuracy: 0.9688
Epoch 5/30
13/13 [==============================] - 31s 2s/step - loss: 0.1683 - accuracy: 0.9332 - val_loss: 0.0789 - val_accuracy: 0.9688
Epoch 6/30
13/13 [==============================] - 34s 3s/step - loss: 0.1468 - accuracy: 0.9381 - val_loss: 0.0531 - val_accuracy: 0.9896
Epoch 7/30
13/13 [==============================] - 35s 3s/step - loss: 0.1569 - accuracy: 0.9406 - val_loss: 0.1668 - val_accuracy: 0.9375
Epoch 8/30
13/13 [==============================] - 36s 3s/step - loss: 0.1830 - accuracy: 0.9158 - val_loss: 0.0514 - val_accuracy: 0.9896
Epoch 9/30
13/13 [==============================] - 32s 2s/step - loss: 0.1455 - accuracy: 0.9356 - val_loss: 0.0378 - val_accuracy: 0.9896
Epoch 10/30
13/13 [==============================] - 34s 3s/step - loss: 0.1761 - accuracy: 0.9307 - val_loss: 0.0352 - val_accuracy: 1.0000
```

```
Epoch 11/30
13/13 [==============================] - 35s 3s/step - loss: 0.1391 - accuracy: 0.9530 - val_loss: 0.0413 - val_accuracy: 0.9896
Epoch 12/30
13/13 [==============================] - 37s 3s/step - loss: 0.1264 - accuracy: 0.9505 - val_loss: 0.0580 - val_accuracy: 0.9792
Epoch 13/30
13/13 [==============================] - 34s 3s/step - loss: 0.1306 - accuracy: 0.9406 - val_loss: 0.0191 - val_accuracy: 1.0000
Epoch 14/30
13/13 [==============================] - 35s 3s/step - loss: 0.1083 - accuracy: 0.9554 - val_loss: 0.0361 - val_accuracy: 0.9792
Epoch 15/30
13/13 [==============================] - 35s 3s/step - loss: 0.0869 - accuracy: 0.9678 - val_loss: 0.0203 - val_accuracy: 0.9896
Epoch 16/30
13/13 [==============================] - 31s 2s/step - loss: 0.1200 - accuracy: 0.9579 - val_loss: 0.0275 - val_accuracy: 0.9896
Epoch 17/30
13/13 [==============================] - 31s 2s/step - loss: 0.1556 - accuracy: 0.9233 - val_loss: 0.0402 - val_accuracy: 0.9896
Epoch 18/30
13/13 [==============================] - 33s 3s/step - loss: 0.1405 - accuracy: 0.9406 - val_loss: 0.0595 - val_accuracy: 0.9792
Epoch 19/30
13/13 [==============================] - 34s 3s/step - loss: 0.1334 - accuracy: 0.9356 - val_loss: 0.0559 - val_accuracy: 0.9896
Epoch 20/30
13/13 [==============================] - 33s 3s/step - loss: 0.1130 - accuracy: 0.9530 - val_loss: 0.0251 - val_accuracy: 0.9896
Epoch 21/30
13/13 [==============================] - 39s 3s/step - loss: 0.1073 - accuracy: 0.9406 - val_loss: 0.0313 - val_accuracy: 0.9896
Epoch 22/30
13/13 [==============================] - 29s 2s/step - loss: 0.1091 - accuracy: 0.9480 - val_loss: 0.0170 - val_accuracy: 1.0000
Epoch 23/30
13/13 [==============================] - 30s 2s/step - loss: 0.0939 - accuracy: 0.9567 - val_loss: 0.0128 - val_accuracy: 1.0000
Epoch 24/30
13/13 [==============================] - 29s 2s/step - loss: 0.0759 - accuracy: 0.9728 - val_loss: 0.0037 - val_accuracy: 1.0000
```

```
Epoch 24/30
13/13 [==============================] - 29s 2s/step - loss: 0.0759 - accuracy: 0.9728 - val_loss: 0.0037 - val_accuracy: 1.0000
Epoch 25/30
13/13 [==============================] - 35s 3s/step - loss: 0.0758 - accuracy: 0.9777 - val_loss: 0.0118 - val_accuracy: 1.0000
Epoch 26/30
13/13 [==============================] - 34s 3s/step - loss: 0.0707 - accuracy: 0.9802 - val_loss: 0.0079 - val_accuracy: 1.0000
Epoch 27/30
13/13 [==============================] - 36s 3s/step - loss: 0.1081 - accuracy: 0.9480 - val_loss: 0.0235 - val_accuracy: 0.9896
Epoch 28/30
13/13 [==============================] - 35s 3s/step - loss: 0.0975 - accuracy: 0.9678 - val_loss: 0.0092 - val_accuracy: 1.0000
Epoch 29/30
13/13 [==============================] - 34s 3s/step - loss: 0.0746 - accuracy: 0.9752 - val_loss: 0.0072 - val_accuracy: 1.0000
Epoch 30/30
13/13 [==============================] - 35s 3s/step - loss: 0.0695 - accuracy: 0.9777 - val_loss: 0.0720 - val_accuracy: 0.9583
```

Saving the model

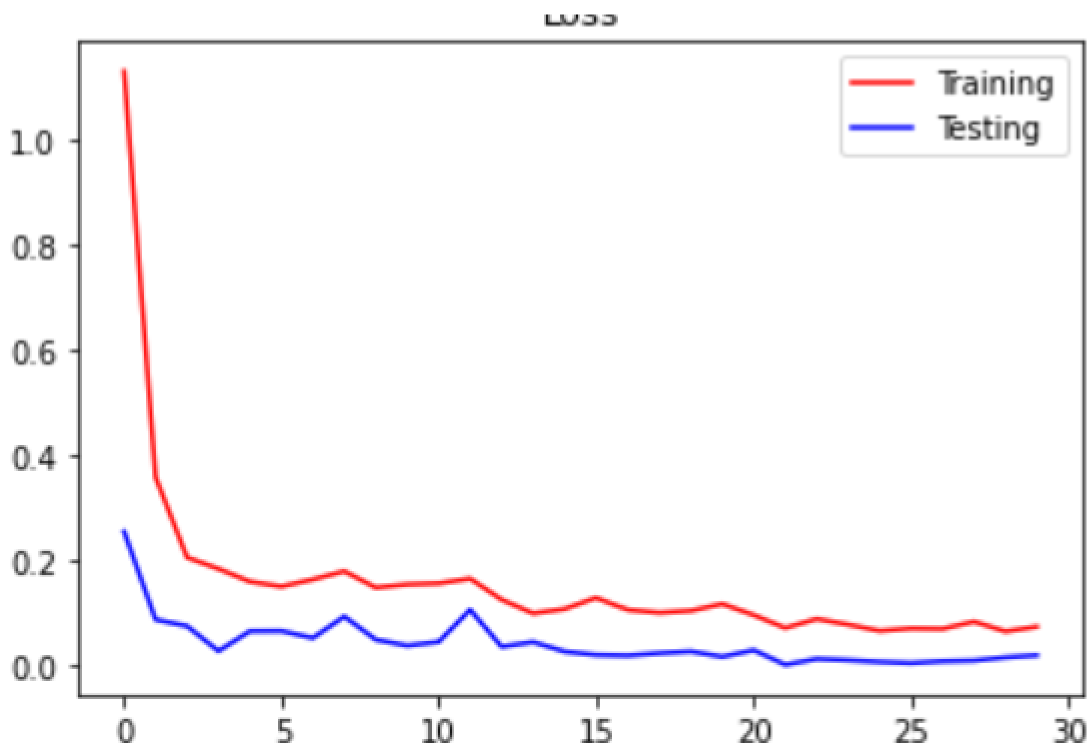
8]:

```
model.save("fire.h5")
```

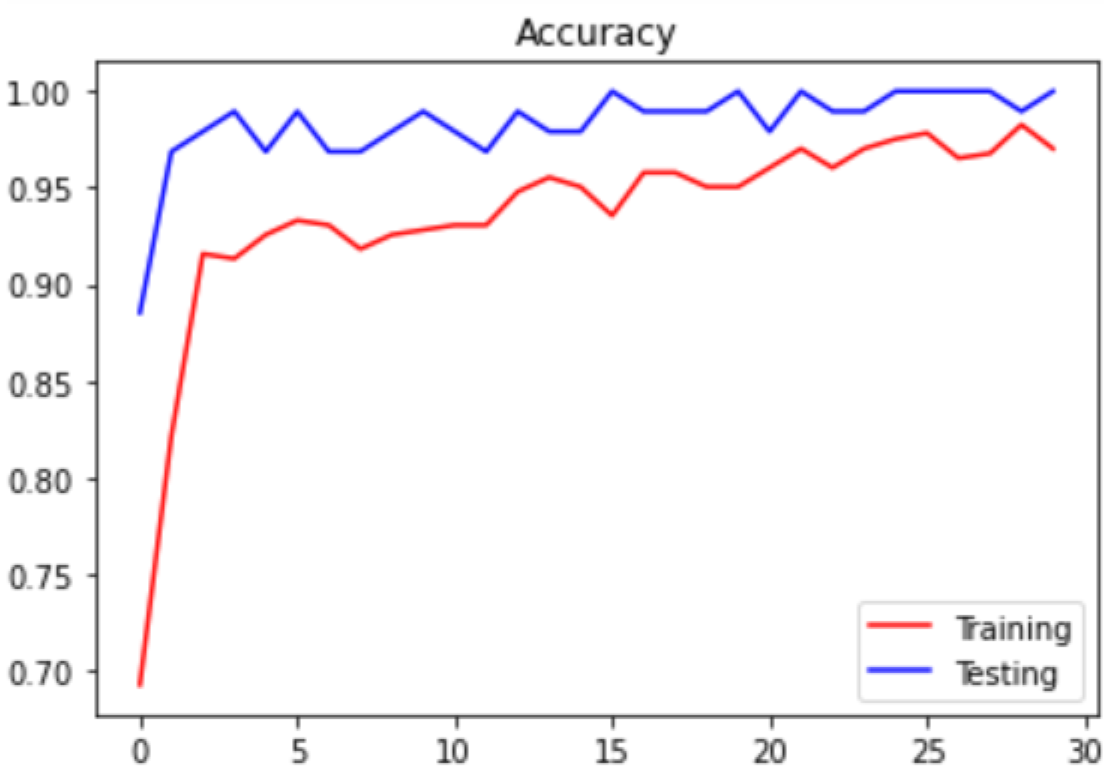
2. Loss or No loss

3. Accuracy Value

```
plt.figure(0)

plt.title("Loss")

plt.plot(hist.history['loss'], 'r', label='Training')

plt.plot(hist.history['val_loss'], 'b', label='Testing')

plt.legend()

plt.show()
```

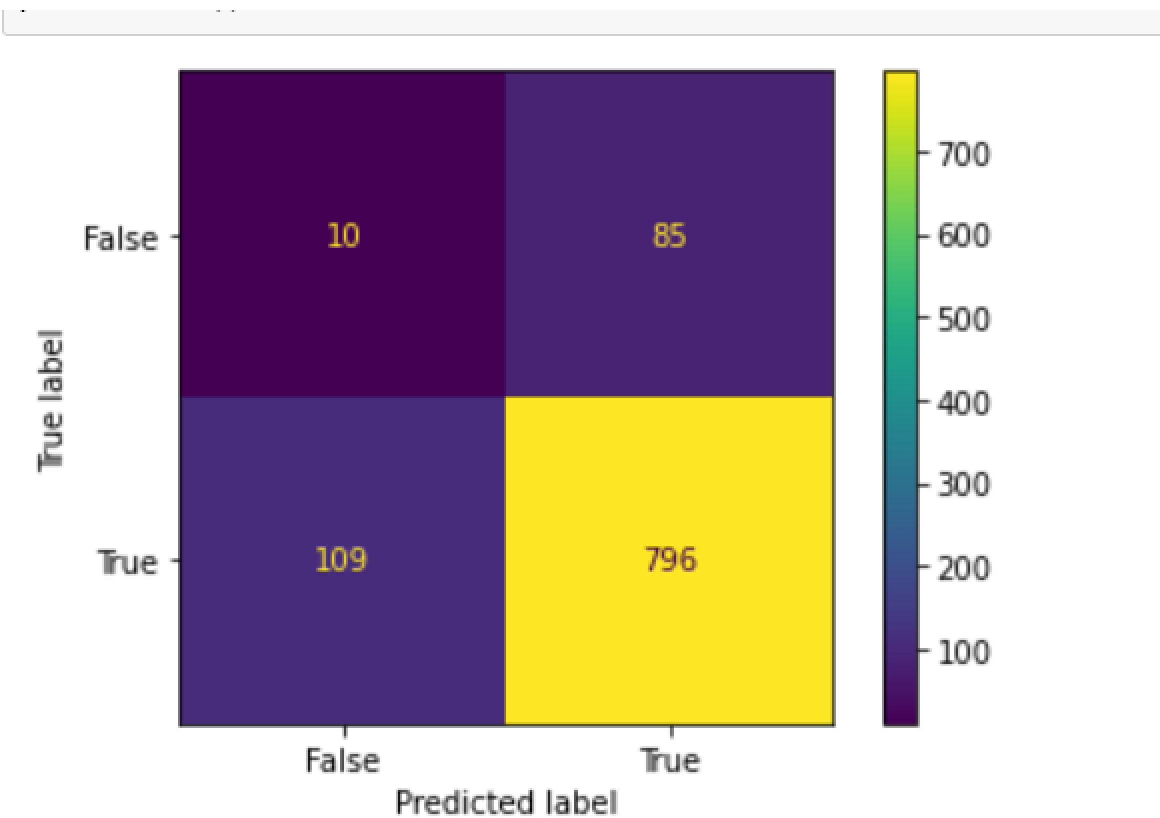


```
plt.figure(1)

plt.title("Accuracy")

plt.plot(hist.history['accuracy'], 'r', label='Training')

plt.plot(hist.history['val_accuracy'], 'b', label='Testing')

plt.legend()

plt.show()
```



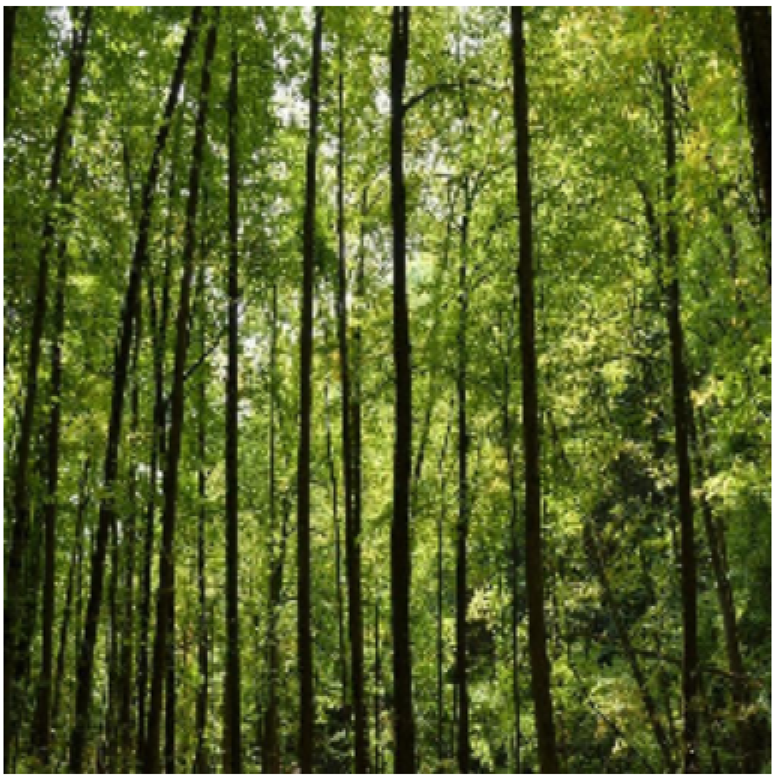
4. Confusion matrix

```
#confusion matrix
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics
actual = numpy.random.binomial(1,.9,size= 1000)
predicted = numpy.random.binomial(1,.9,size = 1000)
confusion_matrix = metrics.confusion_matrix(actual, predicted)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix,display_labels = [False,True])
cm_display.plot()
plt.show()
```



5. Predictions

```
from tensorflow.keras.models import load_model


from tensorflow.keras.preprocessing import image
model = load_model("fire.h5")

img = image.load_img(r"C:\Users\Isha\Pictures\Saved Pictures\egnofire.jpg",target_size=(256,256))


img
```



```
type(img)


PIL.Image.Image

x = image.img_to_array(img)
```

x

```
array([[[ 12.,  14.,   0.],
        [ 21.,  24.,   7.],
        [ 43.,  46.,  27.],
        ...,
        [ 21.,  19.,   7.],
        [ 13.,  15.,   2.],
        [ 52.,  60.,  11.]],

       [[ 13.,  15.,   2.],
        [ 12.,  14.,   0.],
        [ 18.,  21.,   4.],
        ...,
        [ 17.,  15.,   2.],
        [ 10.,  11.,   3.],
        [ 58.,  65.,  23.]],

       [[ 14.,  15.,   7.],
        [ 11.,  13.,   2.],
        [ 10.,  12.,   0.],
        ...,
        [ 19.,  18.,   0.],
        [ 17.,  18.,  13.],
       [ 62.,  66.,  39.]],
...,

       [[ 14.,  15.,   7.],
        [ 51.,  56.,  26.],
        [ 48.,  57.,   2.],
        ...,
        [ 50.,  65.,  26.],
        [ 58.,  75.,  30.],
        [ 54.,  73.,  27.]],
```

```
        [[ 17.,  19.,   8.],
         [ 49.,  54.,  24.],
         [103., 112.,  57.],
         ...,
         [ 65.,  80.,  41.],
         [ 61.,  78.,  33.],
         [ 64.,  83.,  37.]],

        [[ 18.,  18.,   8.],
         [ 36.,  39.,   8.],
         [ 77.,  85.,  28.],
         ...,
         [ 79.,  94.,  55.],
         [ 50.,  67.,  22.],
         [ 52.,  71.,  25.]]], dtype=float32)
```

```
x.shape
```

```
(256, 256, 3)
```

```
import numpy as np

# convolution expects 4D
x = np.expand_dims(x,axis=0)

x.shape
```

```
(1, 256, 256, 3)
```

```
pred_prob = model.predict(x)
```

```
1/1 [==============================] - 0s 111ms/step
```

```
pred_prob
```

array([[0.]], dtype=float32)

```
if(pred_prob==0):
    print("There is no fire")
else:
    print("There is a fire")
```

There is no fire

Chapter 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- The proposed system detects the forest fire at a faster rate compared to the existing system. It has an enhanced data collection feature.
- The major aspect is that it reduces false alarms and also has accuracy due to the various sensors present.
- It minimizes human effort as it works automatically.
- This is very affordable due to which it can be easily accessed.
- The main objective of our project is to receive an alert message through an app to the respective user.
- The arrangement is fire-proof and can withstand high temperatures, rugged, reliable, cost-effective, and easy to install.
- It is also easy to decode the data from satellites at the ground station and no experts are required to understand or decode the data from the satellite.
- All the components like the temperature sensor and the GPS are easy to interface.
- The approximate value of temperature and the GPS coordinates are obtained. Since we are using wireless sensing networks, the attenuation during the transmission of the signal or the data is minimised.
- It is More Reliable

DISADVANTAGES:

- The electrical interference diminishes the effectiveness of the radio receiver.
- The main drawback is that it has less coverage range areas.
- Even a small fault would cause the whole system to fail.

Chapter 11

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. Here, to obtain a more accurate outcome within the lowest latency, the analysis should take place continuously and camera monitoring should be effectively done. This system is well developed to fit any weather condition, climatic condition, or area. In the case of node deployment, cameras can be mounted at any place in the forest even with good connectivity and built-in network infrastructure. IR frame sensors are used to enhance the efficiency of the system. A unique feature that sends alert messages to the concerned authorities when the fire is detected is also added. Thus, By detecting forest fires we can reduce air pollution, landslides, and soil erosion by protecting strong-rooted trees, and the emission of CO2 into the air during fire causing no loss of life and resources.

Chapter 12

FUTURE SCOPE

- Right now we have designed the project for the control of two devices but it can be designed for more numbers of devices.
- It can be further expanded with a voice interactive system facility.
- A feedback system can also be included which provides the state of the device to the remote users.

Chapter 13

APPENDIX

Source Code

#Download the Dataset

pwd

#Load the Image Dataset from
google.colab import drive
drive.mount('/content/drive')

 # call load_data with allow_pickle implicitly set to true import
numpy as np

data **=** np**.**load('/content/drive/My Drive/Forest-Dataset/Dataset.zip', allow_pickle**=True**)
print('data loaded') cd **//**content**/**drive**/**MyDrive**/**Forest**-**Dataset

#Unzip the Dataset

!unzip Dataset**.**zip

#Image Preprocessing

#1.Importing the ImageDataGenerator Library **import** numpy **as** np **import** keras
from sklearn.model_selection **import** train_test_split **from** keras.models **import**
Sequential, load_model **from** keras.preprocessing.image **import**
ImageDataGenerator **from** keras.callbacks **import** ModelCheckpoint,
EarlyStopping, TensorBoard **from** keras.callbacks **import** ReduceLROnPlateau

from keras.layers **import** Conv2D, Dropout, Dense, Flatten, MaxPooling2D,
SeparableConv2D, Activation, BatchNormalization **import** matplotlib.pyplot **as**
plt **import** time **import** os

import tensorflow **as** tf

#2.Define parameters for ImageDataGenerator Class

```
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)

#3.Applying ImageDataGenerator Functionality to Trainset and Testset

#*a. For Dataset*

x_dataset =train_datagen.flow_from_directory(r"/content/drive/MyDrive/ForestDataset/forest_fire",target_size = (128,128), class_mode = "binary",batch_size=32)

#*b. For Trainset*

x_train =train_datagen.flow_from_directory(r"/content/drive/MyDrive/ForestDataset/forest_fire/Training and Validation",target_size = (128,128), class_mode = "binary",batch_size=32)

c. For Testset

x_test =test_datagen.flow_from_directory(r"/content/drive/MyDrive/ForestDataset/forest_fire/Testing",target_size = (128,128), class_mode = "binary", batch_size=32)

x_train.class_indices

 # Model Building

 # 1.Importing the Model Building Libraries

#Importing model libraries

from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense from tensorflow.keras.layers import Convolution2D from tensorflow.keras.layers import MaxPooling2D from tensorflow.keras.layers import Flatten

import warnings

warnings.filterwarnings('ignore')

##2.Initializing the Model

model=Sequential()

 # 3.Adding CNN Layers

#a. adding convolutional layer

model.add(Convolution2D(32,(3,3),input_shape=(256,256,3),activation="relu"))

#b. adding max pooling layer

model.add(MaxPooling2D(pool_size=(2,2)))

#c. adding flatten layer

model.add(Flatten())

#Model Summary

model.summary()

 # 4.Adding Dense Layers

#a. Adding Hidden layers

model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))

model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))

#b. Adding Output layer

model.add(Dense(units=1,kernel_initializer="random_uniform",activation="sigmoid"))

 # 5.Configuring the Learning Process

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

 # 6.Summarize the model

model.summary()

 # 7.Training the Model

#fit or train the model

```
r=model.fit_generator(x_train,steps_per_epoch=13,
           epochs=30,validation_data=x_test,
           validation_steps=3)
```

#plotting loss value import matplotlib.pyplot
as plt plt.plot(r.history['loss'],label='loss')
plt.plot(r.history['val_loss'],label='val_loss')
plt.legend()

#plotting accuracy value
plt.plot(r.history['accuracy'],label='acc')
plt.plot(r.history['val_accuracy'],label='val_acc')
plt.legend()

 # 8.Save the Model

model.save("fire.h5")

#Training and Deploying the model in cloud

pwd

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='EHmhit2MD64AQnqYijN7mrXyaEYoh02jLsiuzU5mzGbt',

    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 = 'ffdcnnmodelbook-donotdelete-pr-giva0vdmx0opfa'

object_key = 'forestfiredataset.zip'

streaming_body_3 = 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_3.read()),'r')

file_paths=unzip.namelist()

for path in file_paths:

    unzip.extract(path)

print(ls)
```

#Import the libraries

```
import keras
```

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from matplotlib import pyplot as plt

#Importing ImageDataGenerator from Keras

image preprocessing (or) image augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator

#import the cnn layers

#Defining the Parameters

train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)

#rescale => rescaling pixel value from 0 to 255 to 0 to 1

#shear_range=> counter clock wise rotation(anti clock)

test_datagen = ImageDataGenerator(rescale=1./255)

#Applying ImageDataGenerator functionality to train dataset

```
x_train = train_datagen.flow_from_directory(r"/home/wsuser/work/Dataset/Dataset/train_set",
                                            target_size=(256,256),
                                            batch_size=32,
                                            class_mode="binary")
```

#Applying ImageDataGenerator functionality to test dataset

```
x_test = test_datagen.flow_from_directory(r"/home/wsuser/work/Dataset/Dataset/test_set",
                                          target_size=(256,256),
                                          batch_size=32,
                                          class_mode="binary")
```

#Importing Model Building Libraries

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.optimizers import Adam , SGD, RMSprop

print(x_train.class_indices)

#Intializing the model

model = Sequential()

#Adding CNN layers

add convolution layer

model.add(Convolution2D(32,(3,3),input_shape=(256,256,3),activation="relu"))

32 indicates => no of feature detectors

#(3,3)=> kernel size (feature detector size)

#add max pooling layer

model.add(MaxPooling2D(pool_size=(2,2)))

#add flatten layer => input to your ANN

model.add(Flatten())

#Add Dense layers

#hidden layer

model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))

model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))

#output layer

model.add(Dense(units=1,kernel_initializer="random_uniform",activation="sigmoid"))

#Configuring the learning process

#compile the model

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

#Summarize the model

model.summary()

#Training the model

model.fit(x_train,steps_per_epoch=13,epochs=30,validation_data=x_test,validation_steps=3)

#steps_per_epoch = no of training images/batch size

#validation_steps = no of testing images/batch size

#Saving the model

model.save("fire.h5")

#IBM Deployment

```
!pip install watson-machine-learning-client

from ibm_watson_machine_learning import APIClient

wml_credentials={

    "url":"https://us-south.ml.cloud.ibm.com",

    "apikey":"1AfypwQwqeHikzD7u4LIKT6DMnD-RPDTyYLRBofzNBPp"

}

client=APIClient(wml_credentials)

print(client)

def guid_space_name(client,fire_deploy):

    space=client.spaces.get_details()

    return(next(item for item in space['resources'] if item['entity']['name']==fire_deploy)['metadata']['id'])

space_uid=guid_space_name(client,'cnn_fire')

print("Space UID "+space_uid)

client.set.default_space(space_uid)

client.software_specifications.list(200)

software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')

print(software_space_uid)

print(ls)

!tar -zcvf fire-classification-model.tgz fire.h5

model_details=client.repository.store_model(model='fire-classification-model.tgz',meta_props={

    client.repository.ModelMetaNames.NAME:"CNN Model Building",

    client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',

    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid

})

model_id=client.repository.get_model_id(model_details)

print(model_id)

'client.repository.download(model_id,'fire.tar.gb')

print(ls)
```

 # 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("fire.h5")
img=image.load_img(r"C:\Users\Isha\Pictures\Saved Pictures\egfire.jpg")
x=image.img_to_array(img)
res=cv2.resize(x,dsize=(256,256),interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)
pred=model.predict(x)
pred = int(pred[0][0])
print(pred)
if pred==1:
  print('Forest fire')
elif pred==0:
  print('No Fire')
```

#OpenCV for Video Processing

```
import cv2
import numpy as np
# importing image function from keras
from keras.preprocessing import image
# importing load_model from keras
from keras.models import load_model
#importing client from twilio API
from twilio.rest import Client
#importing playsound package from playsound
import playsound

model=load_model("fire.h5")
video = cv2.VideoCapture(0)
```

```
name = ['forest','with fire']
```

#Sending an Alert Message through Twilio

```
from twilio.rest import Client
from playsound import playsound
if pred==1:
  print('Forest fire')
  account_sid = 'AC74a73227a4fa4c514205086263a7dba7'
  auth_token = '4d8390c023f6fb46befde35c8e1c0a67'
  client = Client(account_sid, auth_token)
  message = client.messages \
 .create(body= 'Alert! A Forest fire has been detected.',from_='+18314804693',to='+919498400638')
  print(message.sid)
  print("Fire detected")
  print("SMS Sent!")
```

GitHub Link

GitHub Link

https://github.com/IBM-EPBL/IBM-Project-31438-1660200381