DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
GOVERNMENT COLLEGE OF TECHNOLOGY, COIMBATORE-13.

EMERGING METHODS FOR THE EARLY DETECTION OF FORESTS FIRES

DOMAIN – ARTIFICIAL INTELLIGENCE

IBM – NALAIYA THIRAN

PROJECT REPORT

Team Members:

1. Sivanesh M
2. Siva Sankar R
3. Sethuraman P
4. Sathish R

Team ID: PNT2022TMID06939

Batch ID: B8-2A4E

NOVEMBER 19, 2022

GOVERNMENT COLLEGE OF TECHNOLOGY, COIMBATORE-13.

INDEX

1. INTRODUCTION

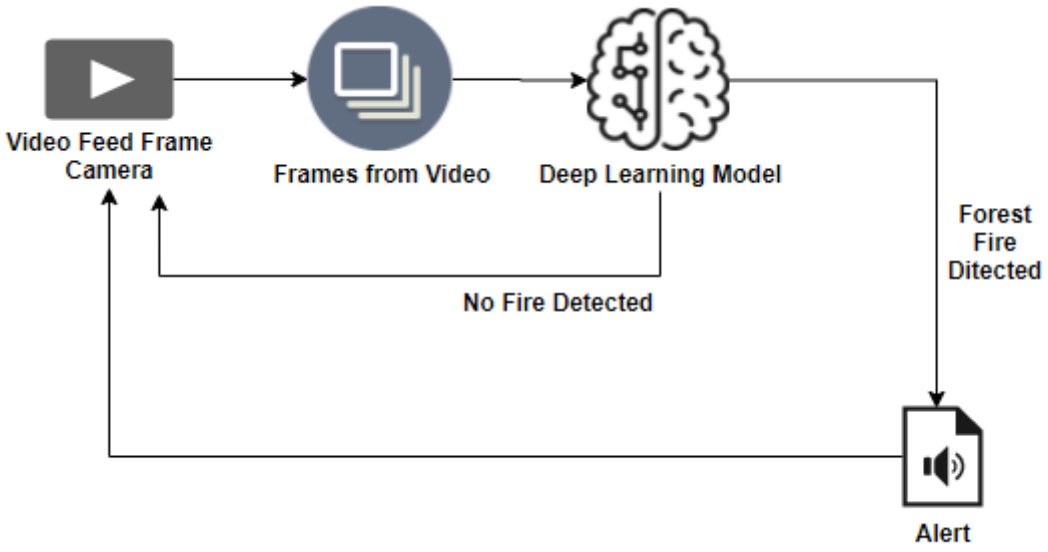
Nowadays, fire incidents have become a critical issue, which must be dealt with on time without any unnecessary delay to avoid the loss in lives and belongings. According to the National Fire Protection Association (NFPA), two- third of U.S. household fires occur in premises with no working smoke alarms, alarms with no proper maintenance, or misplaced alarms. The appropriate allocation of fire alarms with a proactive warning could save lives and reduce property losses. The concept of Artificial Intelligence (AI) nowadays is applied in many applications ranging from the smart industry, smart agriculture to smart healthcare and smart home application .Nowadays, fires can get out of control because people intend to save money rather than installing proper fire alarm systems. Some problems are still on, such as affordability, effectiveness and responsiveness. Many studies have been conducted to address these issues like; however, fire detection issues are not addressed properly since these systems rely on machine vision, where the algorithms need more images to train, and the detection rate is not satisfactory.

Thus, this paper aims to minimize false alarms, provide faster response, and a new AI approach. The contribution is as follows:

1. To determine which combinations and algorithms of cameras can accurately and quickly detect fires,
2. We have designed and then developed a system that detects fire and send messages to the people.
3. The proposed system evaluates the situation and initiates a message where the message system was designed separately, and
4. The system analyses the collected data using IBM Watson platform which results in a faster response. Thus, the highlighted four points make the proposed system superior in terms of affordability, effectiveness, and responsiveness.

a) Project Overview and Purpose

Emerging methods for the early detection of forest fire has been developed to detect the forest fire by means of established camera. The cameras captures every scenes lively and the videos are collected as a dataset and the dataset is analyzed by performing AI algorithms. The videos are preprocessed, analyzed and alerting message system is created. If the cameras detects the fire, as fast as the SMS Notification send to the users as established. Thus, this project find useful for preventing huge loss of lives due to rapid forests fires.



2. LITERATURE SURVEY

In this paper, the proposed methodology consists of different stages. The stages include

A. Acquisition of Dataset,
B. Data Preprocessing,
C. Feature Extraction,
D. Building model,
E. Validation and testing.

A. Acquisition of Dataset Data is in form of video frames which are obtained from CCTV footages, but for the ease custom made videos are to be used to perform training and test. the collection of such videos with fire is a tedious task. The frames with fire and without fire are then stored as respectively. Then we divide the dataset as training set and test set. This is to be done with great care because if the data fed to the neural network is faulty, the results will be corrupted and fail to produce an accurate system.

B. Data preprocessing is the next stage of building a quality machine learning model. Here, the data gets cleaned and processed or simply make the data fit for use. Data preprocessing consist of removing noises and other unwanted objects from the frame. The algorithm must require relevant data otherwise it may produce undesired results.

C. Feature Extraction For the neural network to accurately detect fire, it needs to know the features of fire, how it looks like in computer's vision. The feature of fire is easily identifiable by human eye. Fire emits reddish color; it has a shape under different circumstances and motion depending on the fuel it uses to burn. In this paper, the shape, color and motion of fire and smoke is used for the detection. We extract the features from different frames in the training set. The neural network extracts these features using feature extraction network in the CNN which is powered by a custom algorithm. After extracting the features these video frames are classified into fire and non-fire scenarios. The features are extracted using bounding boxes using image descriptors.

D. Building the model The extracted features are then passed to the network to build a model. This model is a set of thresholds to help the network to accurately detect fire. The model learns from the features extracted and set a standard for analyzing new input data.

E. Validation and testing Validation of the machine learning model is essential because it is clearly important to get the accuracy and see if the system is working. The validation process is executed using another set of video frame which is completely unique from the dataset provided to build the model. According, the test results the system achieved about 93 % accuracy with the validation set.

a) Existing problem

Fire disasters are man-made disasters, which cause ecological, social, and economic damage. To minimize these losses, early detection of fire and an autonomous response are important and helpful to disaster management systems. Therefore, in this article, we propose an early fire detection framework using fine-tuned convolutional neural networks for CCTV surveillance cameras, which can detect fire in varying indoor and outdoor environments. To ensure the autonomous response, we propose an adaptive prioritization mechanism for cameras in the surveillance system. Finally, we propose a dynamic channel selection algorithm for cameras based

on cognitive radio networks, ensuring reliable data dissemination. Experimental results verify the higher accuracy of our fire detection scheme compared to state-of-the-art methods and validate the applicability of our framework for effective fire disaster management.

b) References

[1] Saponara, S., Elhanashi, A. & Gagliardi, A. Realtime video fire/smoke detection based on CNN in antifire surveillance systems. J Real-Time Image Proc 18, 889–900 (2021).

[2] Qingjie Zhang, Jiaolong Xu, Liang Xu and Haifeng Guo, Deep Convolutional Neural Networks for Forest Fire Detection. IFMEITA 2016.

[3] Pragati, S. S. (2019-2020). International Journal Of Advance Scientific Research. Forest Fire Detection Using Machine Learning, 1,2.

[4] A Arul, R. S. (2021, May). Fire Detection System Using Machine Learning. Retrieved from ResearchGate: https://www.researchgate.net/publication/351926970_ Fire_Detection_System_Using_Machine_Learning.

[5] Yuanbin Wang, L. D. (2019). Journal of algorithms and Computational technology. Forest fire image recognition based on convolutional neural network, 1.

[6] Kim, Byoungjun, and Joonwhoan Lee. 2019. "A Video-Based Fire Detection Using Deep Learning Models" Applied Sciences 9, no. 14: 2862

[7] Xu, R.; Lin, H.; Lu, K.; Cao, L.; Liu, Y. A Forest Fire Detection System Based on Ensemble Learning. Forests 2021, 12, 217. https://doi.org/10.3390/f12020217

[8] BoWFire Dataset. Available online: https://bitbucket.org/gbdi/bowfire-dataset/downloads/ (accessed on 1 January 2021). 29.

[9] FD-Dataset. Available online: http://www.nnmtl.cn/EFDNet/ (accessed on 1 January 2021).

[10] ForestryImages. Available online: https://www.forestryimages.org/browse/subthumb.cf m?sub=740 (accessed on 1 January 2021). 31. [10].

[11] VisiFire. Available online: http://signal.ee.bilkent.edu.tr/VisiFire/ (accessed on 1 January 2021).

[12] Alkhatib, A. A. (2014). International Journal of distributed sensor networks. A Review on Forest Fire Detection Techniques.

[13] Begoña C. Arrue, A. O. (2000). An Intelligent System for FalseAlarm Reduction in Infrared ForestFire Detection. Retrieved from IEEE: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnu mber=846287

[14] Eric den breejen, M. B. (1998, November). FOREST FIRE DETECTION USING MACHINE LEARNING. Retrieved from Researchgate: https://www.researchgate.net/profile/KlamerSchutte/publication/2478027_Autonomous_Forest_ Fi re_Detection/links/0912f514831a07aee6000000/Auto nomous-Forest-Fire-Detection.pdf
Problem Statement Definition

c) Problem Statement Definition

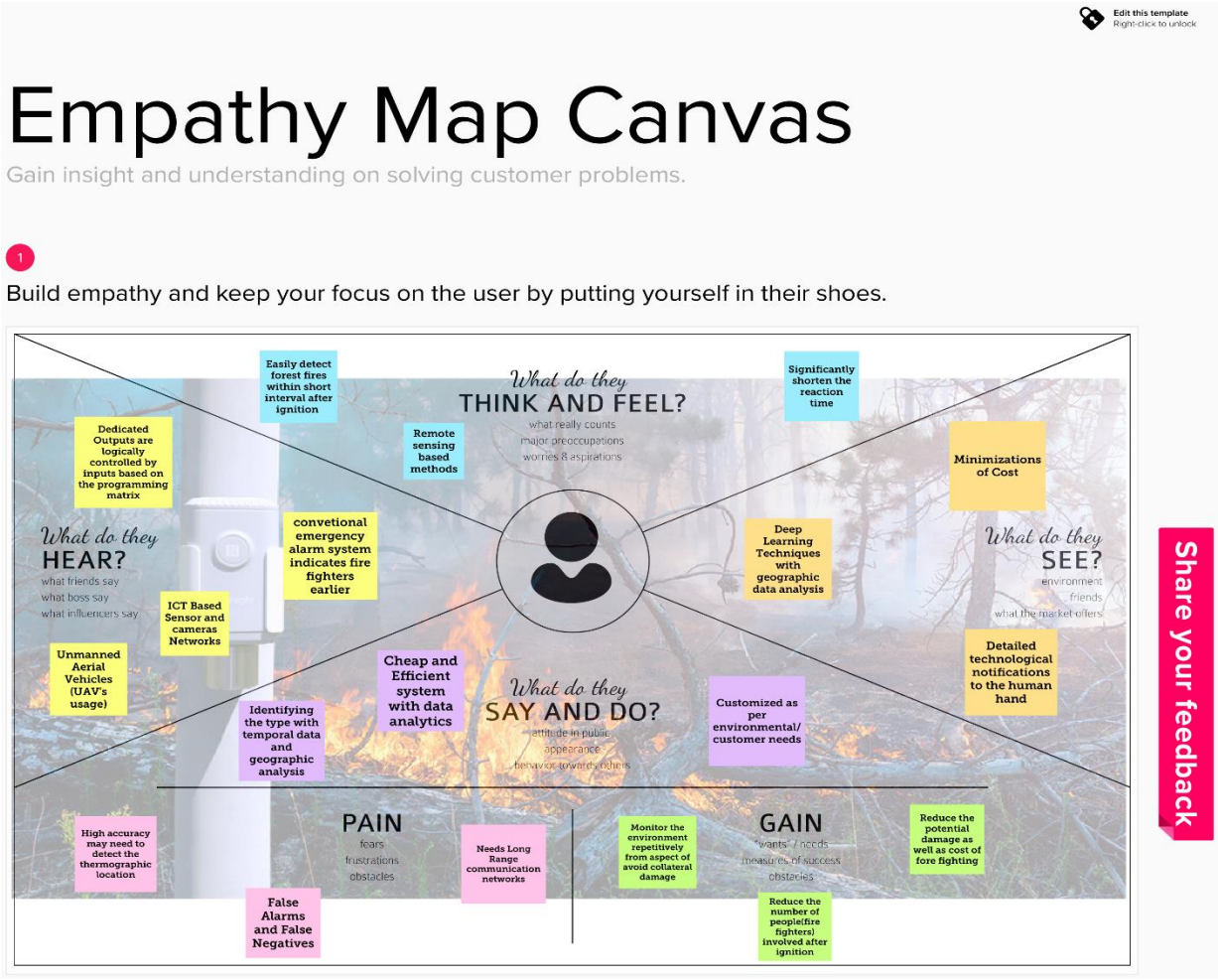
Emerging Methods for Early Detection Of Forest Fires: Forest fires are a major environmental issue, creating economic and ecological damage while endangering human lives. There are typically about 100,000 wildfires in the United States every year. Over 9 million acres of land have been destroyed due 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 to 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.

3. IDEATION & PROPOSED SOLUTION

Ideation is expressed via graphical, written, or verbal methods, and arises from the past or present knowledge, influences, opinions, experiences and personal convictions.

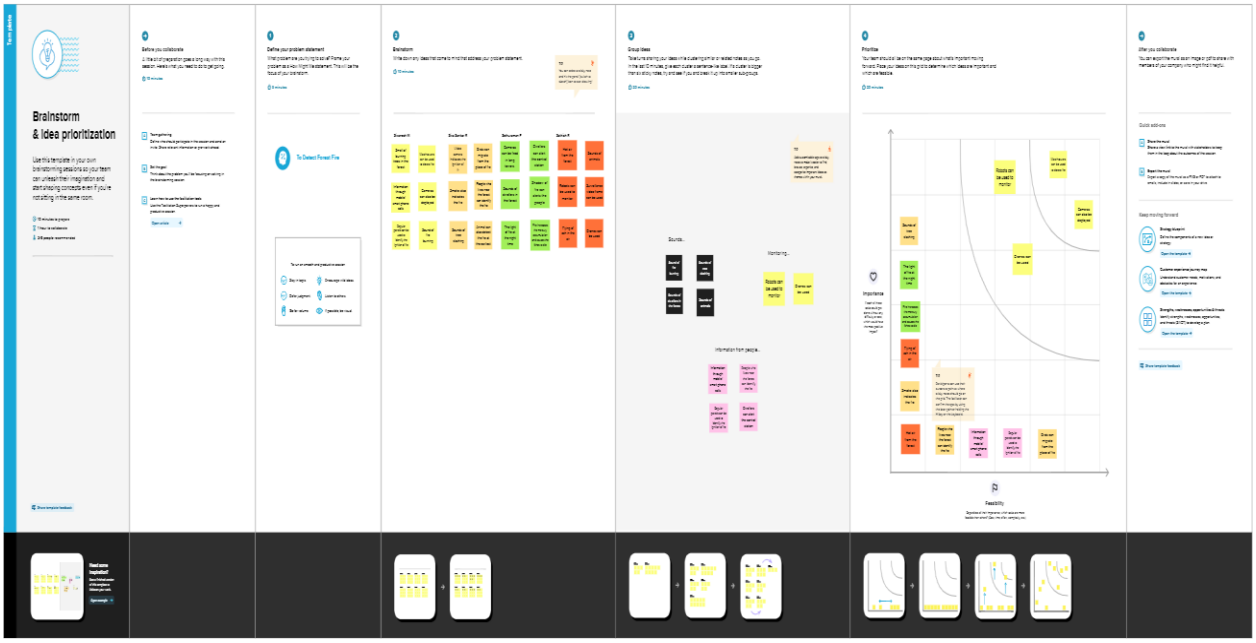
a) Empathy Map Canvas

It serves as a foundation for understanding user experiences, which focuses on providing the experience customer want rather than forcing design teams to relay on guess work.



b) Ideation and Brainstorming

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas.



c) Proposed Solution

It should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved.

Project Design Phase-I
Proposed Solution Template

Date	24 September 2022
Team ID	PNT2022TMID06939
Project Name	Project-Early detection of forest fire using deep learning

Proposed Solution:

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	1.Forest fires are a major environmental issue, creating economic and ecological damage while endangering human lives. 2.It is difficult to predict and detect Forest Fire in a sparsely populated forest area. 3. So, it is necessary to detect the fire in an early stage to control it.

2.	Idea / Solution description	1.The model will detect forest fires automatically with the help of image processing using deep learning and with the use of satellite image data to observe, detect and report fire events.
3.	Novelty / Uniqueness	When the fire is detected, the station will get a notification via message and an alarm system will be activated automatically to alert the user.
4.	Social Impact / Customer Satisfaction	1.This can reduce the forest fire in the beginning stage, by alerting users. 2.The user can also use this as a surveillance 3.Camera to monitor the forest. Saving the most essential Forest cover.
5.	Business Model (Revenue Model)	1.This application will be available in a subscription-based model. 2.Supply chain, power & supply, Fire stations, and government by providing services.
6.	Scalability of the Solution	1.This application can monitor different places simultaneously and can detect fire accurately 2.This application can handle a large number of users and data simultaneously.

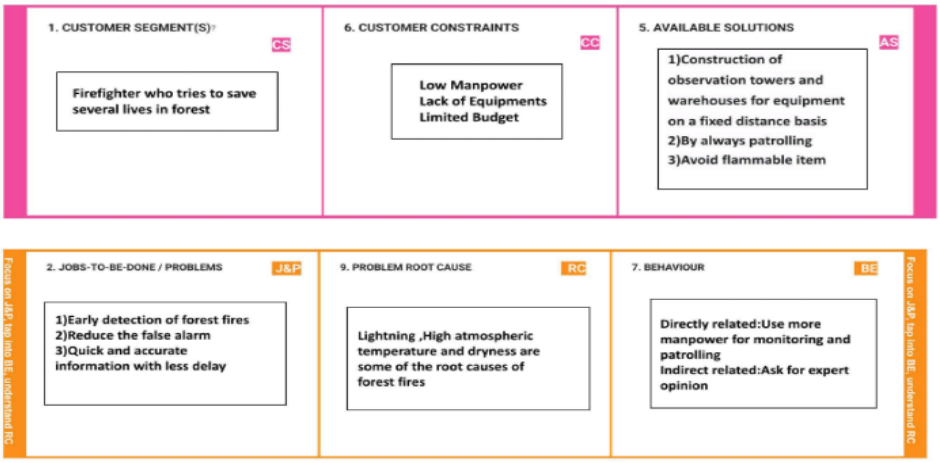
d) Problem Solution Fit

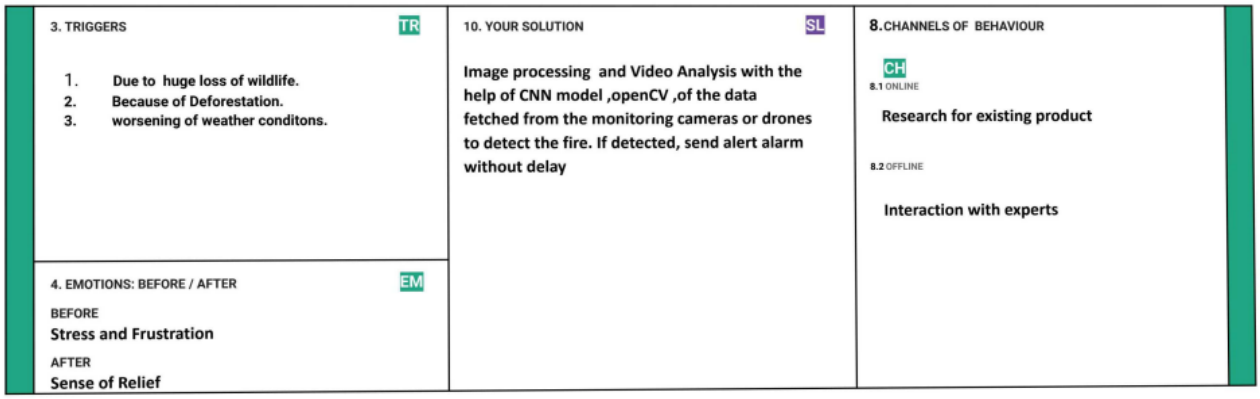
It means we have to found a problem with our customer and that the solution we have realized for it actually solves our customers problem.

Project Design Phase-I - Solution Fit Template

Project Title: Emerging Method for Early Detection of Forest Fires

TEAM ID:PNT2022TMID06939





By
Sivanesh
Sathish
Sethuraman
Siva Sankar

4. REQUIREMENT ANALYSIS

Solution requirements describe specific characteristics that a product must have to meet the needs of the stakeholders and the business itself. They fall into two large groups.

• Functional requirements define what a product must do, what its features and functions are.

• Non-functional requirements describe the general properties of a system. They are also known as quality attributes.

a) Functional Requirements

Following are the functional requirements of the proposed solution.

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	15 October 2022
Team ID	PNT2022TMID06939
Project Name	Emerging methods for early detection of forest fires
Maximum Marks	4 Marks

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	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 cíeate computeí models to píedict wildfiíe potential undeí a íange of potential climate futuíes. Using diffeíent píojections of tempeíatuíe and píecipitation, scientists píedict wheíe and when wildfiíes aíe most likely to occuí.
FR-4	**Using Sensors**	This Bosch environment sensors installed in the forest fire detection system using artificial intelligence deployed as early wildfire warming tool.

b) Non-Functional Requirements

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

Non-functional Requirements:

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

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 application-based systems, with both positive and negative aspects and performance figures of detection.
NFR-2	**Security**	We have designed this project to secure the forest from wild fires.
NFR-3	**Reliability**	It has achieved 1.24 seconds of classification time with an accuracy of 91% and F1 score of 0.91.
NFR-4	**Performance**	In the event of a fiie, the piimaíy objective of using díones is to gatheí situational awaíeness, which can be used to diíect the effoíts of the fiíefighteís in locating and contíolling hot spots. Just like uíban fiíes, foíest fiíes to íequiíe monitoíing so that fiíefighteís know what they aíe dealing with.
NFR-5	**Availability**	Forest fires (wildfires) are common hazards in forests, particularly in remote or unmanaged areas. It is possible to detect forest fires, elevated CO2, and temperature levels using AI
NFR-6	**Scalability**	A widely used measure of fire intensity is fireline intensity, which is the rate of heat transfer per unit

5. PROJECT DESIGN

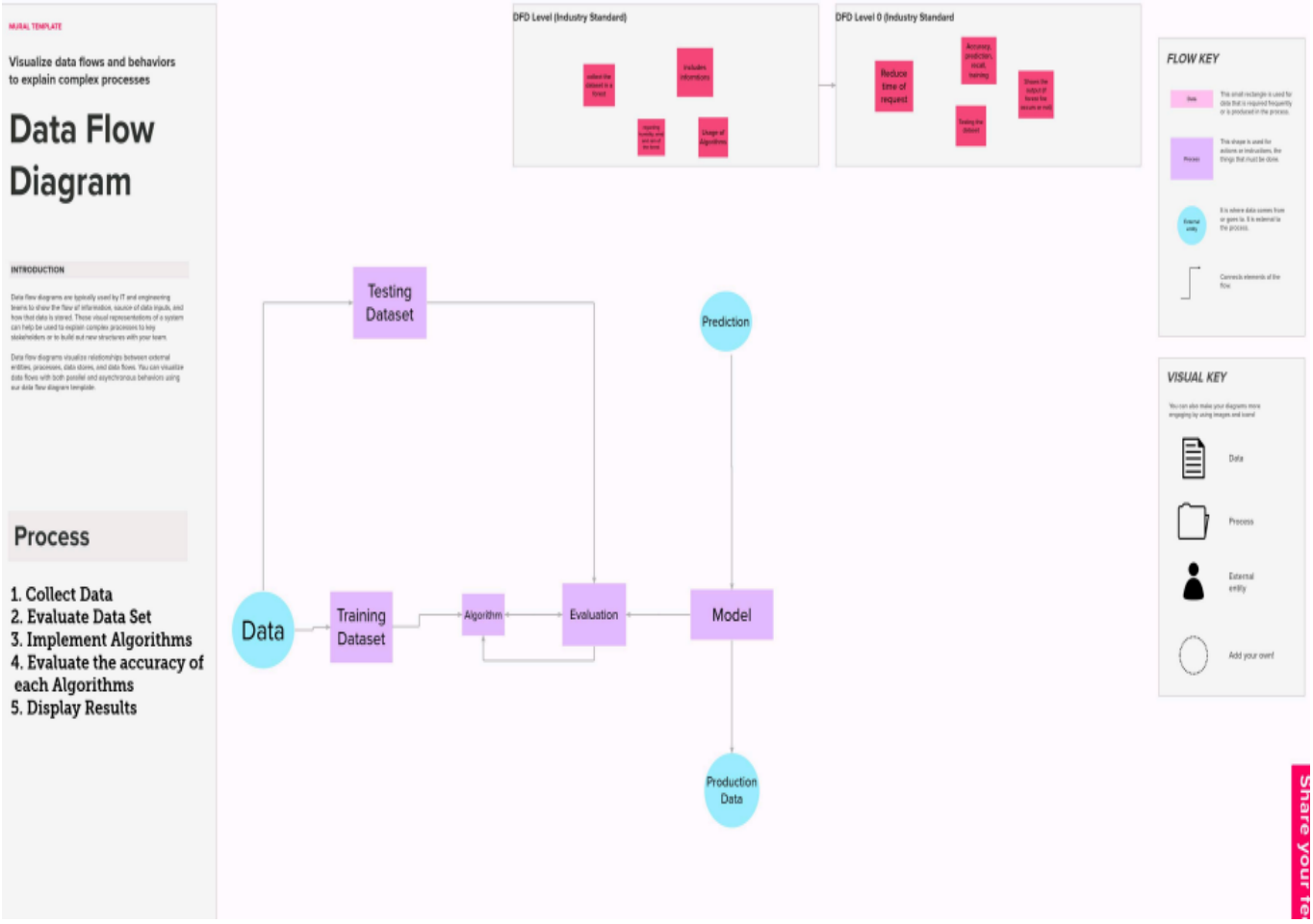
Project design is an early phase of the project life cycle where ideas, processes, resources and deliverables are planned out.

a) Data Flow Diagrams

It is a graphical or visual representation using a standardized set of symbols and notations to describe business's operation through data movement.

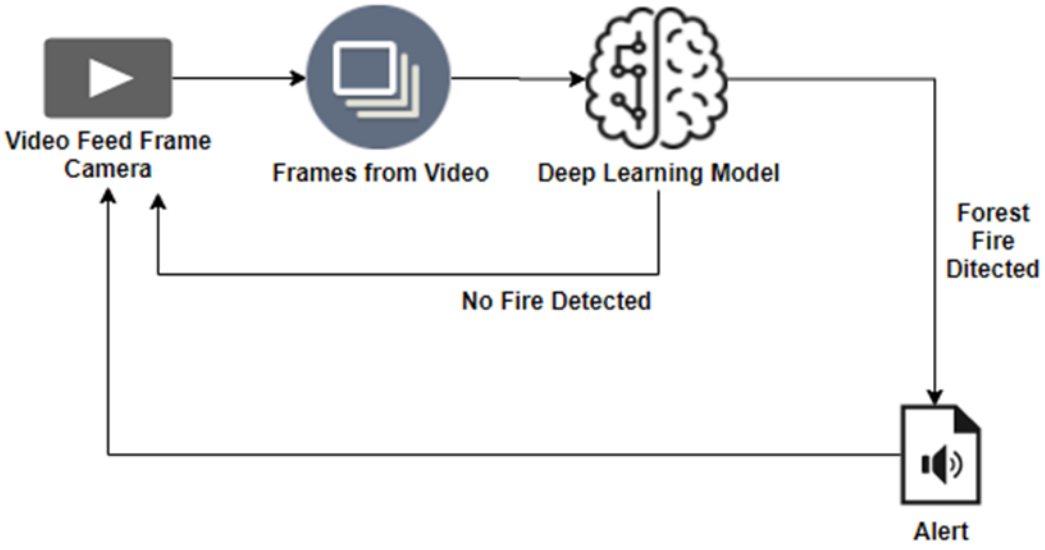
PROJECT DESIGN PHASE - II
DATA FLOW DIAGRAM & USER STORIES

DATE	15 October 2022
TEAM ID	PNT2022TMID06939
PROJECT NAME	Emerging Methods for earlier forest fire detection
MAXIMUM MARKS	4 Marks



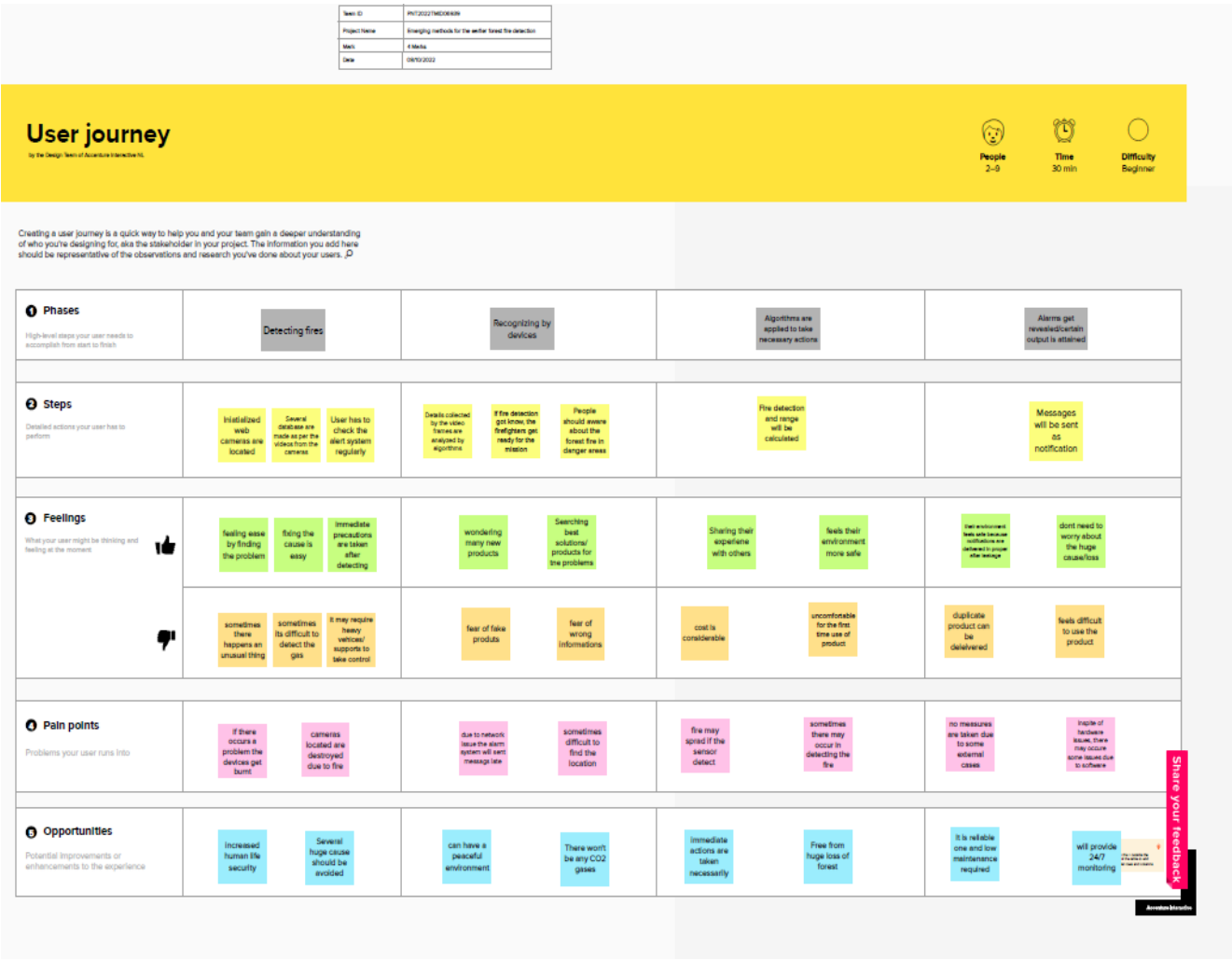
b) Solution and Technical Architecture

Technical architecture is a form of IT architecture that is used to design computer systems.



c) User Stories

User story is an informal, general explanation of a sftware feature return from the perspective of the end user/customer.



6. PROJECT PLANNING & SCHEDULING

a) Sprint Planning and Estimation

Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	22 October 2022
Team ID	PNT2022TMID06939
Project Name	Emerging Methods for Early Detection of Forest Fires
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	20	High	SATHISH R SIVANESH M SIVA SANKAR R SETHURAMAN P
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application usage.	20	High	SATHISH R SIVANESH M SIVA SANKAR R SETHURAMAN P
Sprint-2	Input	USN-3	Whenever the fire is detected, the information is given to the database.	20	High	SATHISH R SIVANESH M SIVA SANKAR R SETHURAMAN P
Sprint-2		USN-4	When it is the wildfire then the alarming system is activated.	20	High	SATHISH R SIVANESH M SIVA SANKAR R SETHURAMAN P

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Output	USN-5	And the alarm also sent to the corresponding departments and made them know that the wildfire is erupted.	20	High	SATHISH R SIVANESH M SIVA SANKAR R SETHURAMAN P
Sprint-4	Action	USN-6	Required actions will be taken in order to controlled erupted wildfire by reaching as early as possible to the destination with the help of detecting systems.	20	High	SATHISH R SIVANESH M SIVA SANKAR R SETHURAMAN P

b) Sprint Delivering Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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

Team ID	**PNT2022TMID06939**
Project Name	**Emerging Methods for Early Detection of Forest Fires**
Team members	SIVANESH M SIVA SANKAR R SATHISH R SETHURAMAN P

MILESTONE LIST

Milestone Name	Milestone Number	Description	Mandatory	
Project Objectives	M-01	We will be able to learn to prepare dataset, image processing, working with CNN layers, read images using OpenCV and CNN for computervision AI	Yes	-
Project Flow	M-02	A project management process flowchart is a graphical aid, designed to visualize the sequence of steps to be followed throughout the project management process	Yes	
Pre-Requisites	M-03	To complete this project, we should have known following project such as Keras, TensorFlow, Python ,Anaconda, OpenCV, Flask, Scikit-learn etc.…	Yes	

Prior Knowledge	M-04	One should have knowledge on the Supervised Learning ,CNN and Regression Classification and Clustering, ANN	Yes	
Data collection	M-05	We can collect dataset from different open sources like kaggle.com, UCI machine learning etc.	Yes	
Image Preprocessing	M-06	Importing the ImageDataGenerator libraries, Define Parameters/Arguments for ImageDataGenerator class, Applying Image Data Generator Functionality to trainset and test set	Yes	
Model Building	M-07	Importing the model building libraries, Initializing the model, Adding CNN layers, Adding Dense layers, Configuring the learning Process, Train the model, Save the model, Predictions.	Yes	
Video Analysis	M-08	Opencv for video processing, creating an account in twilio service and sending alert message	Yes	
Train CNN model	M-09	Register for IBM Cloud and train Image Classification Model	Yes	
Ideation Phase	M-10	Prepare Literature Survey on the selected Project and Information Gathering, empathy map and ideation	Yes	
Project Design Phase-I	M-11	Prepare Proposed solution , problem-solution fit and Solution Architecture	Yes	
Project Design Phase-II	M-12	Prepare Customer journey ,functional requirements, Dataflow diagram and Technology Architecture	Yes	
Project Planning Phase	M-13	Prepare Milestone list , Activity list and Sprint Delivery Plan	Yes	
Project Development Phase	M-14	Project Development delivery of Sprint 1, Sprint 2, Sprint 3, Sprint 4	Yes	

ACTIVITY LIST

Activity Number	Activity	Sub Activity	Assigned To	Status
1.	PROJECT OBJECTIVES		All Members	Completed
2.	PROJECT FLOW		All Members	Completed
3.	PRE-REQUISITES		All Members	Completed
4.	DATA COLLECTION	4.1 Download the Dataset	SIVANESH M	Completed
5.	IMAGE PREPROCESSING	5.1 Import the ImageDataGenerator Library. 5.2 Define the Parameters/Arguments for ImageDataGenerator class. 5.3 Applying ImageDataGenerator Functionality to trainset and testset.	All Members	In Progress
6.	MODEL BUILDING	6.1 Importing the model building libraries. 6.2 Initializing the model. 6.3 Adding CNN layers. 6.4 Adding dense layers. 6.5 Configuring the	All Members	In Progress

		learning process. 6.6 Training the model. 6.7 Saving the model. 6.8 Predictions.		
7.	VIDEO ANALYSIS	7.1 Open CV for videoprocessing. 7.2 Creating an account in Twilio service. 7.3 Sending alert message.	SIVANESH M	In Progress
8.	TRAIN CNN MODEL ON IBM	8.1 Train image classification model. 8.2 Register for IBM cloud.	SIVANESH M	In Progress
9.	IDEATION PHASE	9.1 Literature Review. 9.2 Empathy map. 9.3 Ideation.	All Members	Completed
10.	PROJECT DESIGN PHASE – I	10.1 Proposed Solution. 10.2 Problem solution fit. 10.3 Solution Architecture.	All Members	Completed
11.	PROJECT DESIGN PHASE -II	11.1 Customer journey. 11.2 Functional requirement. 11.3 Data flow Diagrams. 11.4 Technology Architecture.	All Members	Completed
12.	PROJECT PLANNING PHASE	12.1 Prepare milestone and activity list. 12.2 Sprint delivery plan.	All Members	Completed
13.	PROJECT DEVELOPMENT PHASE	13.1 Project development-Delivery of Sprint-1. 13.2 Project development-Delivery of Sprint-2. 13.3 Project development-Delivery of Sprint-3. 13.4 Project development-Delivery of Sprint-4.	All Members	In Progress

7. PROJECT FLOW

- ✓ The user interacts with a web camera to read the video.
- ✓ 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.

a) Activities and Tasks

To accomplish this, we have completed all the activities and tasks listed below

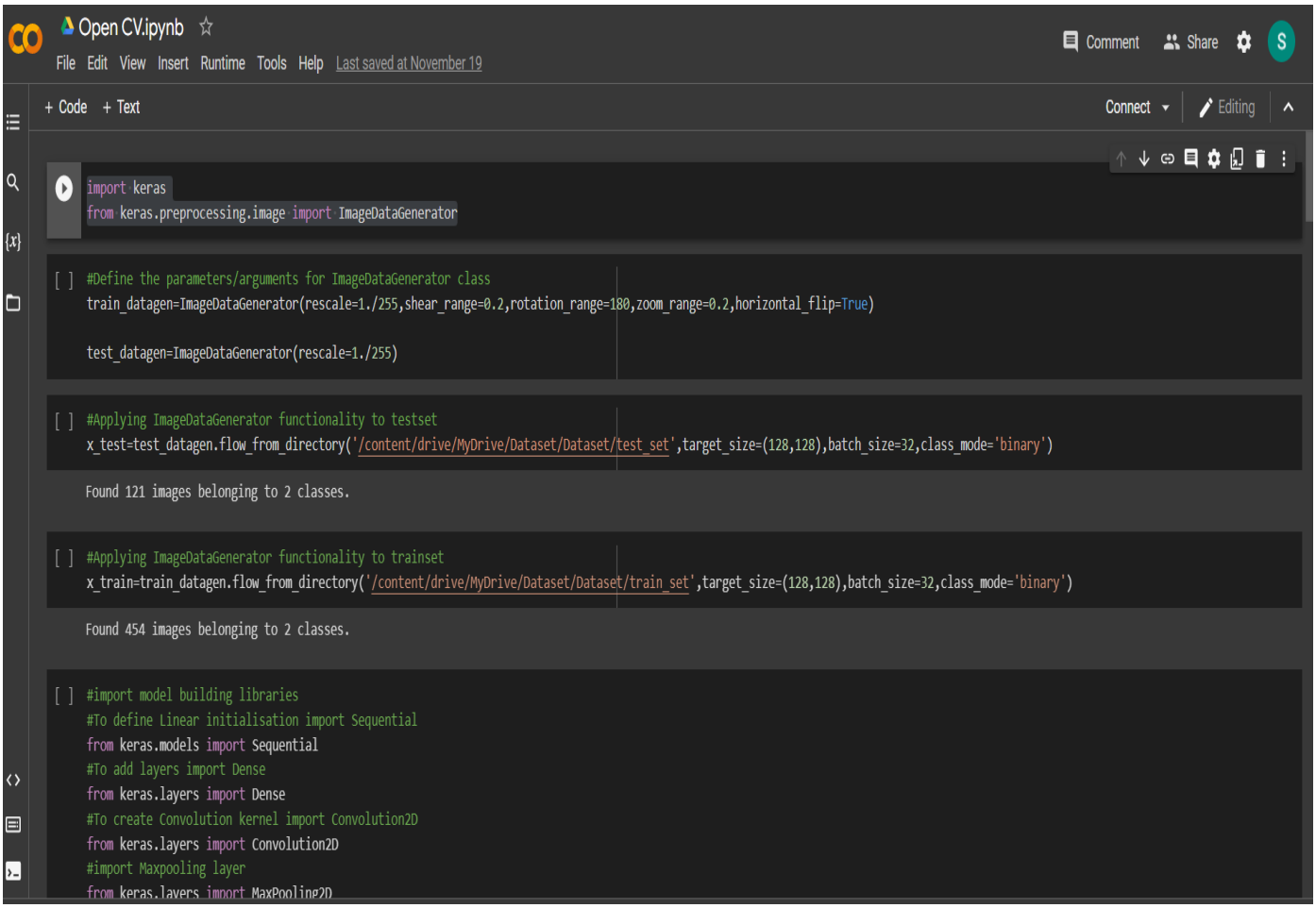
- Data Collection
  - ✓ Collect the dataset or create the dataset.
- Image Pre-processing
  - ✓ Import Image Data Generator Library.
  - ✓ Define the parameters/arguments for Image Data Generator class

- ✓ Applying Image Data Generator on trainset and test set.

- Model Building
  - ✓ Import the model building Libraries
  - ✓ Initializing the model
  - ✓ Adding CNN Layers
  - ✓ Adding Hidden Layer
  - ✓ Adding Output Layer
  - ✓ Configure the Learning Process
  - ✓ Training and testing the model
  - ✓ Optimize the Model
  - ✓ Save the Model
- Video Streaming and alerting
  - ✓ Open CV for video processing
  - ✓ Creating an account in Twilio service
  - ✓ Use Twilio API to send messages.

8. CODING & SOLUTIONING

a) Image Pre-processing



b) Model Building

```
[ ] #import model building libraries
    #To define Linear initialisation import Sequential
    from keras.models import Sequential
    #To add layers import Dense
    from keras.layers import Dense
    #To create Convolution kernel import Convolution2D
    from keras.layers import Convolution2D
    #import Maxpooling layer
    from keras.layers import MaxPooling2D
    #import flatten layer
    from keras.layers import Flatten
    import warnings
    warnings.filterwarnings('ignore')
```

```
[ ] #initializing the model
    model=Sequential()
```

```
[ ] #add convolutional layer
    model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
    #add maxpooling layer
    model.add(MaxPooling2D(pool_size=(2,2)))
    #add flatten layer
    model.add(Flatten())
```

```
[ ] #add hidden layer
    model.add(Dense(150,activation='relu'))
    #add output layer
    model.add(Dense(1,activation='sigmoid'))
```

```
[ ] #configure 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 [==============================] - 118s 8s/step - loss: 2.1155 - accuracy: 0.7180 - val_loss: 0.1380 - val_accuracy: 0.9669
    Epoch 2/10
    14/14 [==============================] - 27s 2s/step - loss: 0.3859 - accuracy: 0.8910 - val_loss: 0.1455 - val_accuracy: 0.9504
    Epoch 3/10
    14/14 [==============================] - 27s 2s/step - loss: 0.3946 - accuracy: 0.8768 - val_loss: 0.2072 - val_accuracy: 0.9504
    Epoch 4/10
    14/14 [==============================] - 25s 2s/step - loss: 0.3493 - accuracy: 0.8957 - val_loss: 0.1262 - val_accuracy: 0.9504
    Epoch 5/10
    14/14 [==============================] - 25s 2s/step - loss: 0.3119 - accuracy: 0.8626 - val_loss: 0.1760 - val_accuracy: 0.9174
    Epoch 6/10
    14/14 [==============================] - 27s 2s/step - loss: 0.2713 - accuracy: 0.8839 - val_loss: 0.1608 - val_accuracy: 0.9174
    Epoch 7/10
    14/14 [==============================] - 26s 2s/step - loss: 0.1896 - accuracy: 0.9336 - val_loss: 0.0827 - val_accuracy: 0.9669
    Epoch 8/10
    14/14 [==============================] - 25s 2s/step - loss: 0.1512 - accuracy: 0.9408 - val_loss: 0.0829 - val_accuracy: 0.9835
    Epoch 9/10
    14/14 [==============================] - 26s 2s/step - loss: 0.1398 - accuracy: 0.9408 - val_loss: 0.1616 - val_accuracy: 0.9504
    Epoch 10/10
    14/14 [==============================] - 27s 2s/step - loss: 0.1734 - accuracy: 0.9289 - val_loss: 0.1682 - val_accuracy: 0.9504
    <keras.callbacks.History at 0x7ff091afb5d0>
```

```
[ ] !pip install tensorflow
    !pip install opencv-python
    !pip install opencv-contrib-python
```

```
predictions = model.predict(x_test)
predictions = np.round(predictions)
predictions
```

```
4/4 [==============================] - 5s 1s/step
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
```

```
#import load_model from keras.model
from keras.models import load_model
#import image class from keras
import tensorflow as tf
from tensorflow.keras.preprocessing import image
#import numpy
import numpy as np
#import cv2
import cv2
```

```
model = load_model("/content/drive/MyDrive/forest1.h5")
```

```
def predictImage(filename):
  img1 = image.load_img(filename,target_size=(128,128))
  Y = image.img_to_array(img1)
  X = np.expand_dims(Y,axis=0)
  val = model.predict(X)
  print(val)
  if val == 1:
    print(" fire")
  elif val == 0:
      print("no fire")
```

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

```
1/1 [==============================] - 0s 82ms/step
[[1.]]
 fire
```

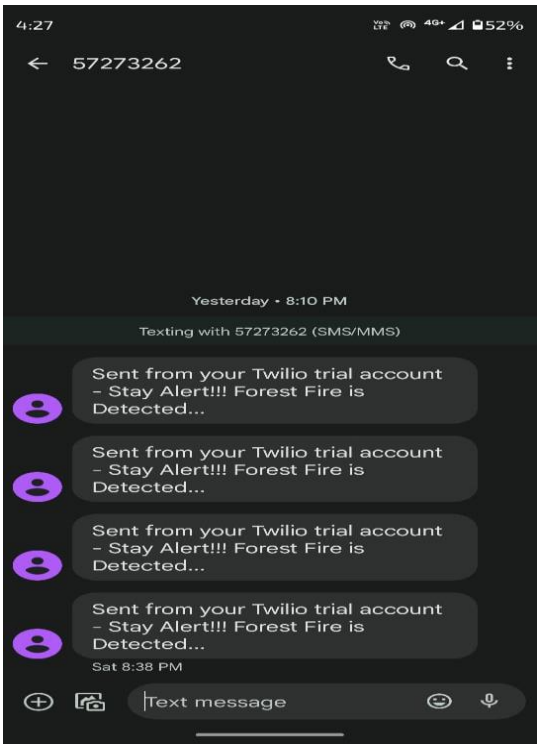
c) Video Streaming and Alerting

```
#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
#imort playsound package
from playsound import playsound

#load the saved model
model = load_model(r'forest1.h5')
#define video
video = cv2.VideoCapture(0)
#define the features
name = ['forest','with forest']

account_sid = 'AC6981743c2fda3548359b3cba6459b343'
auth_token = '07335faf2a5a6aa032080639bd3f4190'
client = Client(account_sid, auth_token)

message = client.messages \
    .create(
         body='Stay Alert!!! Forest Fire is Detected...',
         from_='+18312734662',
         to='+917373083830'
     )

print(message.sid)

#import opencv library
import cv2
#import numpy
import numpy as np
#import images and load_model function from keras
from keras_preprocessing import image
from keras.models import load_model
#import client from twilio API
from  twilio.rest import Client
#import playsound package
from playsound import playsound

#load the saved model
model = load_model(r'forest1.h5')
video = cv2.VideoCapture(0)
name = ['forest','with fire']

while(1):


    success, frame=video.read()
    cv2.imwrite("image.jpg",frame)
    img=image.load_img("image.jpg",target_size=(128,128,3))
    x=image.img_to_array(img)
    x=np.expand_dims(x,axis=0)
    pred=model.predict(x)
    p=pred[0]
    print(pred)
    ##cv2.putText(frame,"predicted class= "+str(name[p]), (100,100),
      ##          cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)
    pred=model.predict(x)
```

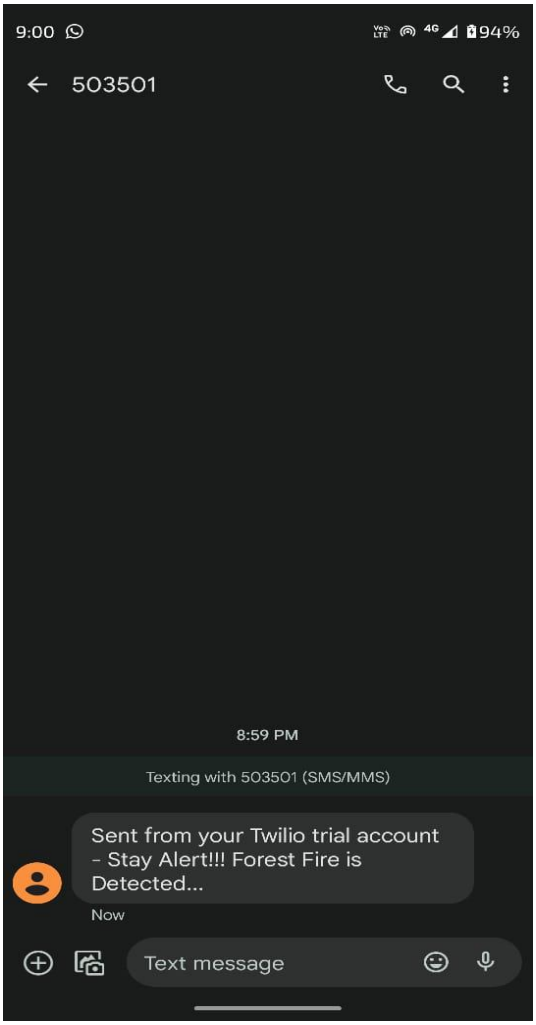
```
    if pred[0]==1:


        account_sid = 'AC6981743c2fda3548359b3cba6459b343'
        auth_token = '07335faf2a5a6aa032080639bd3f4190'
        client = Client(account_sid, auth_token)
        message=client.messages\
        .create(
        body='Stay Alert!!! Forest Fire is Detected...',
        from_='+18312734662',to='+917373083830')

        print(message.sid)
        print('Fire Detected')
        print('SMS sent')
        playsound(r'"/content/drive/MyDrive/Dataset/alarm_buzzer.mp3"')

    else:
        print("No Danger")

        cv2.imshow("image",frame)

    if cv2.waitKey(1) & 0xFF ==ord('a'):
        break
video.release()
cv2.destroyAllWindows()
```

9. TESTING

```
        ##cv2.putText(frame,"predicted class= "+str(name[p]), (100,100),
          ##           cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)
        pred=model.predict(x)


        if pred[0]==1:


            account_sid = 'AC6981743c2fda3548359b3cba6459b343'
            auth_token = '07335faf2a5a6aa032080639bd3f4190'
            client = Client(account_sid, auth_token)
            message=client.messages\
            .create(
            body='Stay Alert!!! Forest Fire is Detected...',
            from_='+18312734662',to='+917373083830')

            print(message.sid)
            print('Fire Detected')
            print('SMS sent')
            playsound(r'"/content/drive/MyDrive/Dataset/alarm_buzzer.mp3"')

        else:
            print("No Danger")

            cv2.imshow("image",frame)

        if cv2.waitKey(1) & 0xFF ==ord('a'):
            break
    video.release()
    cv2.destroyAllWindows()
```


4:27
52%
57273262
Yesterday • 8:10 PM
Texting with 57273262 (SMS/MMS)
Sent from your Twilio trial account - Stay Alert!!! Forest Fire is Detected...
Sent from your Twilio trial account - Stay Alert!!! Forest Fire is Detected...
Sent from your Twilio trial account - Stay Alert!!! Forest Fire is Detected...
Sent from your Twilio trial account - Stay Alert!!! Forest Fire is Detected...
Sat 8:38 PM
Text message

10. RESULTS


9:00
94%
503501
8:59 PM
Texting with 503501 (SMS/MMS)
Sent from your Twilio trial account - Stay Alert!!! Forest Fire is Detected...
Now
Text message

11. ADVANTAGES & DISADVANTAGES

a) Advantages

- Monitoring of the potential risk areas and an early detection of fire can significantly shorten the reaction time and also reduce the potential damage as well as the cost of fire fighting.
- Automatic alerting of admin as well as fire authorities using SMS.
- There is no loss of property or lives and wide cover range of forests will be saved.

b) Disadvantages

a) If the physical device is damaged the entire operation is collapsed.
b) Need large database since many data is stored in cloud database every second.

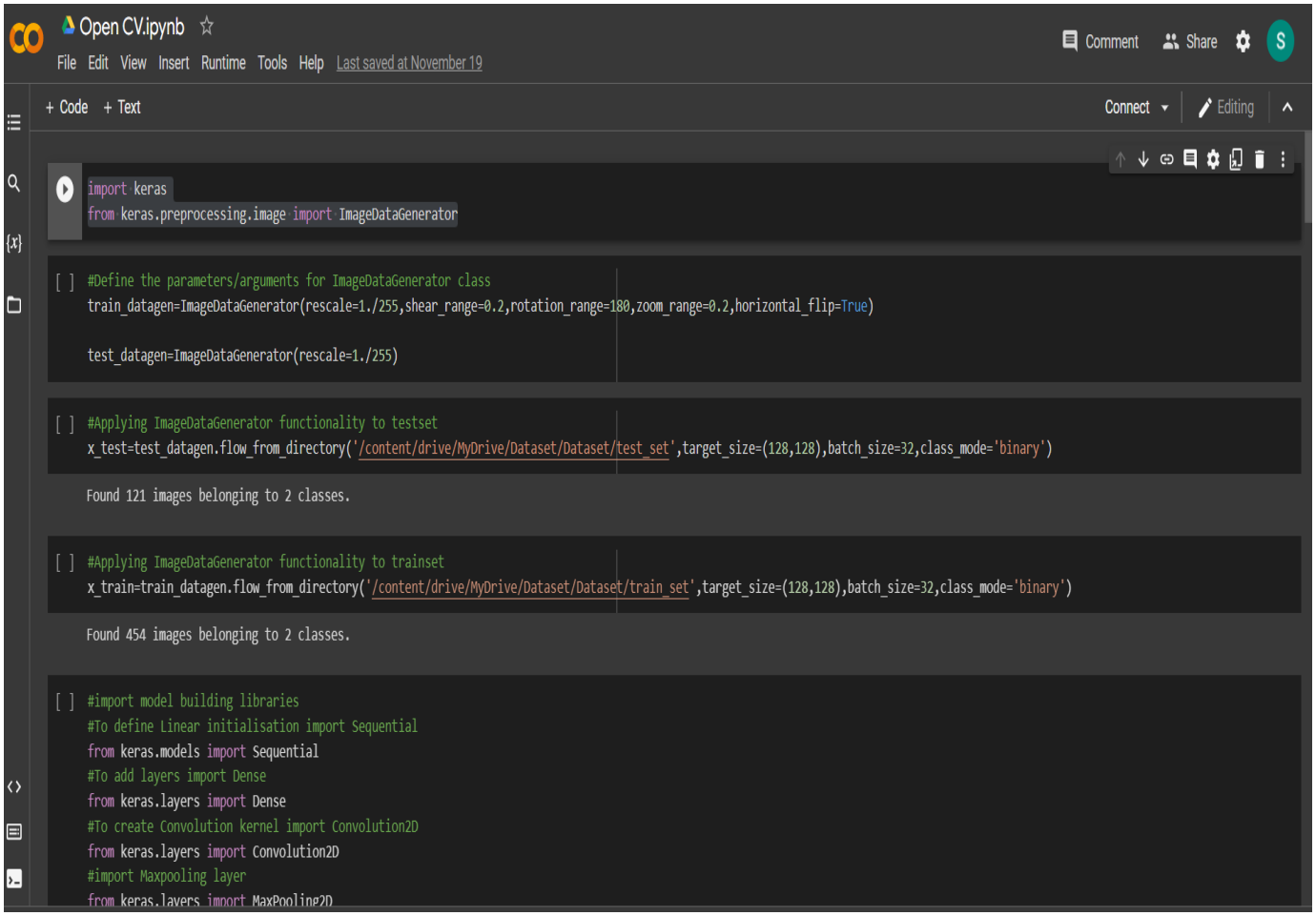
12. CONCLUSION

Finally, concluding that our problem premise is solved using Artificial Intelligence methods(AI) by creating a smart management system that solves many inherent problems in the traditional forests fire management system like actively monitoring for fire breakouts and sending SMS alerts to the admin as well as to the fire authorities .

13. FUTURE SCOPE

The established methods/model can be modified to work in different specialized environment as well as scale to house use to big labs[Since fire accidents can cause major loss in human lives in homes to big industries] as well as it can be used in public places , vehicles.

SOURCE CODE




```
[ ]  #import model building libraries
     #To define Linear initialisation import Sequential
     from keras.models import Sequential
     #To add layers import Dense
     from keras.layers import Dense
     #To create Convolution kernel import Convolution2D
     from keras.layers import Convolution2D
     #import Maxpooling layer
     from keras.layers import MaxPooling2D
     #import flatten layer
     from keras.layers import Flatten
     import warnings
     warnings.filterwarnings('ignore')
```

```
[ ]  #initializing the model
     model=Sequential()
```

```
[ ]  #add convolutional layer
     model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
     #add maxpooling layer
     model.add(MaxPooling2D(pool_size=(2,2)))
     #add flatten layer
     model.add(Flatten())
```

```
[ ]  #add hidden layer
     model.add(Dense(150,activation='relu'))
     #add output layer
     model.add(Dense(1,activation='sigmoid'))
```

```
[ ] #configure 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 [==============================] - 118s 8s/step - loss: 2.1155 - accuracy: 0.7180 - val_loss: 0.1380 - val_accuracy: 0.9669
Epoch 2/10
14/14 [==============================] - 27s 2s/step - loss: 0.3859 - accuracy: 0.8910 - val_loss: 0.1455 - val_accuracy: 0.9504
Epoch 3/10
14/14 [==============================] - 27s 2s/step - loss: 0.3946 - accuracy: 0.8768 - val_loss: 0.2072 - val_accuracy: 0.9504
Epoch 4/10
14/14 [==============================] - 25s 2s/step - loss: 0.3493 - accuracy: 0.8957 - val_loss: 0.1262 - val_accuracy: 0.9504
Epoch 5/10
14/14 [==============================] - 25s 2s/step - loss: 0.3119 - accuracy: 0.8626 - val_loss: 0.1760 - val_accuracy: 0.9174
Epoch 6/10
14/14 [==============================] - 27s 2s/step - loss: 0.2713 - accuracy: 0.8839 - val_loss: 0.1608 - val_accuracy: 0.9174
Epoch 7/10
14/14 [==============================] - 26s 2s/step - loss: 0.1896 - accuracy: 0.9336 - val_loss: 0.0827 - val_accuracy: 0.9669
Epoch 8/10
14/14 [==============================] - 25s 2s/step - loss: 0.1512 - accuracy: 0.9408 - val_loss: 0.0829 - val_accuracy: 0.9835
Epoch 9/10
14/14 [==============================] - 26s 2s/step - loss: 0.1398 - accuracy: 0.9408 - val_loss: 0.1616 - val_accuracy: 0.9504
Epoch 10/10
14/14 [==============================] - 27s 2s/step - loss: 0.1734 - accuracy: 0.9289 - val_loss: 0.1682 - val_accuracy: 0.9504
<keras.callbacks.History at 0x7ff091afb5d0>
```

```
[ ] !pip install tensorflow
    !pip install opencv-python
    !pip install opencv-contrib-python
```

```
predictions = model.predict(x_test)
predictions = np.round(predictions)
predictions
```

```
4/4 [==============================] - 5s 1s/step
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
```

+ Code + Text

```
Epoch 8/10
14/14 [==============================] - 25s 2s/step - loss: 0.1512 - accuracy: 0.9408 - val_loss: 0.0829 - val_accuracy: 0.9835
Epoch 9/10
14/14 [==============================] - 26s 2s/step - loss: 0.1398 - accuracy: 0.9408 - val_loss: 0.1616 - val_accuracy: 0.9504
Epoch 10/10
14/14 [==============================] - 27s 2s/step - loss: 0.1734 - accuracy: 0.9289 - val_loss: 0.1682 - val_accuracy: 0.9504
<keras.callbacks.History at 0x7ff091afb5d0>
```

```
!pip install tensorflow
!pip install opencv-python
!pip install opencv-contrib-python
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
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.9.2)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.19.6)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (4.1.1)
Requirement already satisfied: flatbuffers<2,>=1.12 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.12)
Requirement already satisfied: tensorboard<2.10,>=2.9 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.1)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.1.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.27.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.14.1)
```

+ Code + Text

```
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
```

```
print(len(predictions))
```

```
121
```

```
model.save("/content/drive/MyDrive/forest1.h5")
```

```
#import load_model from keras.model
```

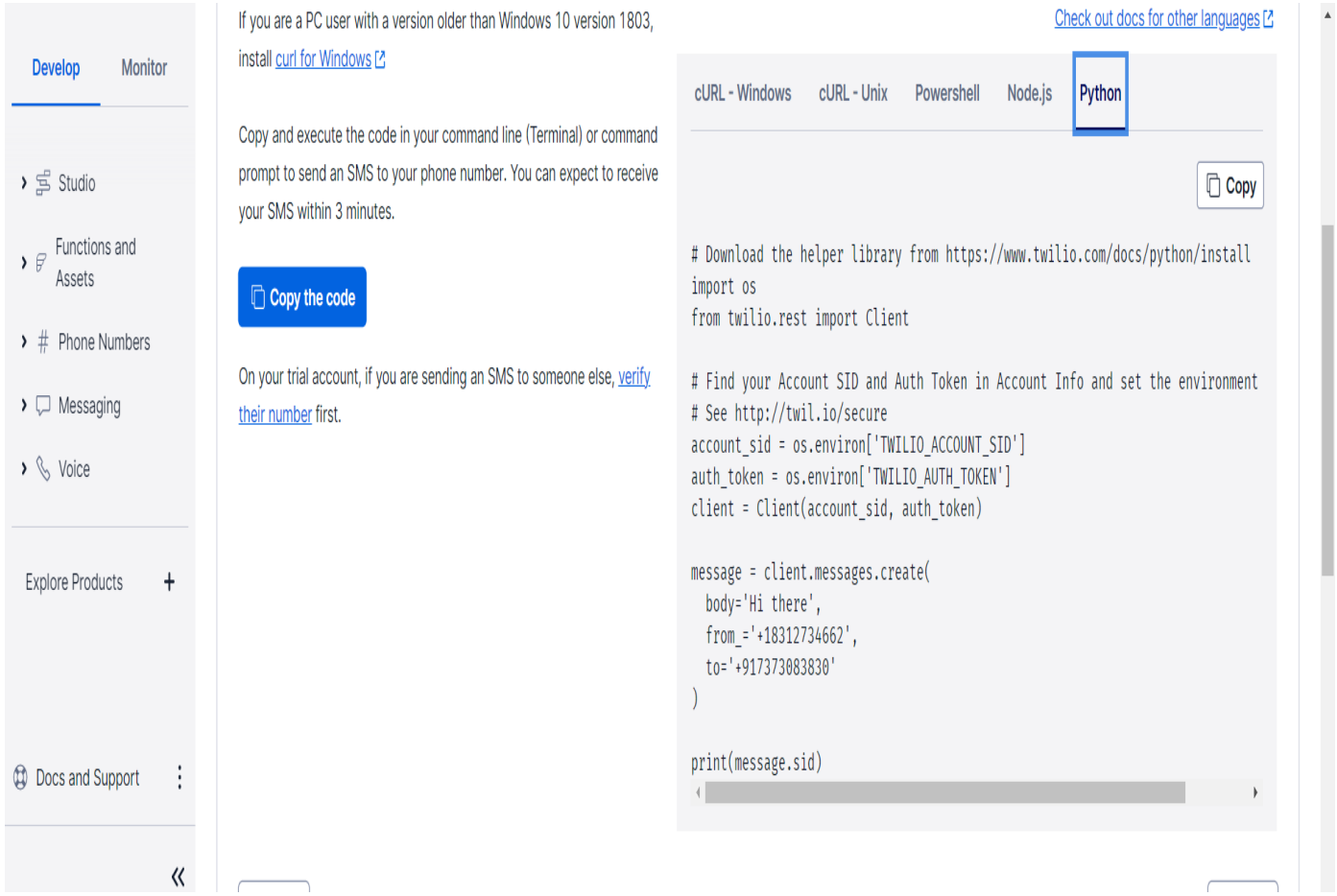
```
[ ]  #import load_model from keras.model
     from keras.models import load_model
     #import image class from keras
     import tensorflow as tf
     from tensorflow.keras.preprocessing import image
     #import numpy
     import numpy as np
     #import cv2
     import cv2
```

```
[ ]  model = load_model("/content/drive/MyDrive/forest1.h5")
```

```
     def predictImage(filename):
       img1 = image.load_img(filename,target_size=(128,128))
       Y = image.img_to_array(img1)
       X = np.expand_dims(Y,axis=0)
       val = model.predict(X)
       print(val)
       if val == 1:
         print(" fire")
       elif val == 0:
           print("no fire")
```

```
[ ]  predictImage("/content/drive/MyDrive/Dataset/Dataset/test_set/with fire/19464620_401.jpg")
```

```
1/1 [==============================] - 0s 82ms/step
[[1.]]
 fire
```



```
[ ]  #import opencv librariy
     import cv2
     #import numpy
     import numpy as np
     #import image function from keras
     from keras.preprocessing import image
     #import load_model from keras
     from keras.models import load_model
     #import client from twilio API
     from twilio.rest import Client
     #imort playsound package
     from playsound import playsound

     #load the saved model
     model = load_model(r'forest1.h5')
     #define video
     video = cv2.VideoCapture(0)
     #define the features
     name = ['forest','with forest']

     account_sid = 'AC6981743c2fda3548359b3cba6459b343'
     auth_token = '07335faf2a5a6aa032080639bd3f4190'
     client = Client(account_sid, auth_token)

     message = client.messages \
          .create(
               body='Stay Alert!!! Forest Fire is Detected...',
               from_='+18312734662',
               to='+917373083830'
           )
```

```
[ ]  #import opencv library
     import cv2
     #import numpy
     import numpy as np
     #import images and load_model function from keras
     from keras_preprocessing import image
     from keras.models import load_model
     #import client from twilio API
     from  twilio.rest import Client
     #import playsound package
     from playsound import playsound

     #load the saved model
     model = load_model(r'forest1.h5')
     video = cv2.VideoCapture(0)
     name = ['forest','with fire']

     while(1):


         success, frame=video.read()
         cv2.imwrite("image.jpg",frame)
         img=image.load_img("image.jpg",target_size=(128,128,3))
         x=image.img_to_array(img)
         x=np.expand_dims(x,axis=0)
         pred=model.predict(x)
         p=pred[0]
         print(pred)
         ##cv2.putText(frame,"predicted class= "+str(name[p]), (100,100),
           ##          cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)
         pred=model.predict(x)
```

```
[ ]         ##cv2.putText(frame,"predicted class= "+str(name[p]), (100,100),
              ##            cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)
            pred=model.predict(x)


            if pred[0]==1:


               account_sid = 'AC6981743c2fda3548359b3cba6459b343'
               auth_token = '07335faf2a5a6aa032080639bd3f4190'
               client = Client(account_sid, auth_token)
               message=client.messages\
               .create(
               body='Stay Alert!!! Forest Fire is Detected...',
               from_='+18312734662',to='+917373083830')

               print(message.sid)
               print('Fire Detected')
               print('SMS sent')
               playsound(r'"/content/drive/MyDrive/Dataset/alarm_buzzer.mp3"')

            else:
                print("No Danger")

                cv2.imshow("image",frame)

            if cv2.waitKey(1) & 0xFF ==ord('a'):
                break
        video.release()
        cv2.destroyAllWindows()
```

LINK

Project File GitHub Link: https://github.com/IBM-EPBL/IBM-Project-7244-1658850766