REPORT

EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

**INTRODUCTION**

**PROJECT OVERVIEW**

Recent advancements in technology have overwhelmingly shaped society, the economy, and the environment. With the help of the various state-of-art technologies such as IoT, blockchain, AI, geospatial mapping, and so on, leading to the fifth industrial revolution, which focuses more on solving climate goals in line with the revolution. New requirements in the ecological environment arise due to the expeditious development of society. Among the various natural disasters, fire hazard seems to own the characteristics of spreading, and also, it becomes very challenging to control, and thus, it results in heavy destruction that might be irrevocable. Over the past few years, there is a tremendous increase in the count, occurrence, and severity of wildfires across the world that has created a great impact on the economy and ecosystem of the country. There are various techniques such as watchtowers, spotter planes, infrared, aerial patrols, and automatic detection systems to detect fire events

**PURPOSE**

It has an inadequate resolution, and hence, the data pertinent to the corresponding area would be taken as an average, and it is restricted to a particular pixel that results in the detection of small fires. The predominant limitation is that the satellites cover only a limited area and require a pre-processing time before the resurvey of the same region. The other limitations such as the shortage of real-time data and inadequate precision are inapt for persistent monitoring. There is a need for the infrastructure in advance if WSNs are deployed. There is more chance for the destruction of the sensors during the fire, and this might lead to more expensive restoration of the sensors. Several factors such as the static nature of the sensors, their coverage, difficulty in maintenance, the deficit in power independence, and non-scalability are the reasons for the sensor networks to limit their efficiency. The sovereignty, less cost, autonomous, and flexibility make the UAV technology the best choice for fire management efforts in the wildland. There are researchers who put more effort into focusing on the development of frameworks.

**LITERATURE SURVEY**

**EXISTING PROBLEM**

From the various inferences, it has been understood that most of the researchers have worked to increase the accuracy, and the area coverage was until 1,500 meters to the maximum. To provide more accuracy and precision, the 3D modeling of data is required, and further visualization of forest fire images could be made very easy for interpretation. The objective of this work is to deploy an efficient and robust detection fire in the early stage. Hence, a deep learning model is required so that the boundary region could be extended, and the 3D modeling images must be considered for the prediction process to augment the accuracy. The contributions in this work are as follows:

(i)An efficient and robust 3D modeling is used to augment the accuracy of the detection.

(ii)A deep learning technique YOLOv4 is combined with the Otsu method along with LiDAR. The key objective of utilizing the Otsu method is to repeat all the values of the threshold and evaluate the extent of the background and foreground pixels. The objective is to determine the threshold by examining the region of the spread, and it should be minimum.

(iii)Traditional methodologies are found to be difficult for performing the sampling since the constraints are bound to the relative position. Hence, the orientation of the images is required, and that is obtained by computing the distance between the tree and other entities with the help of LiDAR describes the proposed methodology.

**REFERENCE**

1. Surapong Surit, Watchara Chatwiriya [8] proposed a method to detect fire by smoke detection in video. This approach is based on digital image processing approach with static and dynamic characteristic analysis. The proposed method is composed of following steps, the first is to detect the area of change in the current input frame in comparison with the background image, the second step is to locate regions of interest (ROIs) by connected component algorithm, the area of ROI is calculated by convex hull algorithm and segments the area of change from image, the third step is to calculate static and dynamic characteristics, using this result we decide whether the object detected is the smoke or not. The result shows that this method accurately detects fire smoke. Surapong Surit, Watchara Chatwiriya [8] proposed a method to detect fire by smoke detection in video. This approach is based on digital image processing approach with static and dynamic characteristic analysis. The proposed method is composed of following steps, the first is to detect the area of change in the current input frame in comparison with the background image, the second step is to locate regions of interest (ROIs) by connected component algorithm, the area of ROI is calculated by convex hull algorithm and segments the area of change from image, the third step is to calculate static and dynamic characteristics, using this result we decide whether the object detected is the smoke or not. The result shows that this method accurately detects fire smoke.

2. P. Piccinini, S. Calderara, and R. Cucchiara [2] proposed a method based on the wavelet model and a color model of the smoke. The proposed method exploits two features: the variation of energy in wavelet model and a color model of the smoke. Smoke is detected based on the decrease of energy ratio in wavelet domain between background and current. The deviation of the current pixel color is measured by the color model. Bayesian classifier is used to combine these two features to detect smoke

3. Osman Gunay and Habiboglu [4] proposed a system based on Covariance Descriptors, Color Models, and SVM Classifier. This system uses video data. Spatio-temporal Covariance Matrix (2011) [13] is used in this system which divides the video data into temporal blocks and computes covariance features. The fire is detected using this feature. SVM Classifier is used to filer fire and fire-like regions. This system supports only for clear data not for blur data

. 4. Hamed Adab [6] proposed another system which is based on Indexing. GIS techniques and remote sensing [10] provides further assistance. The indexing may be structural fire index, Fire risk index, Hybrid fire index. Depending on the geographical condition of the area the indexing differs. Validations of indices are based on hot spot data. Structural fire indices show static information and it does not change over short time span and used to predict the risk in advance. Fire risk index changes as the vegetation or climate changes. Hybrid index is a combination of Structure and Fire index. The disadvantage of this indexing is that way of combining

5. Celik (2007) [3] proposed a generic model for fire and smoke detection without the use of sensors [15]. Fuzzy based approach is used in this system. Color models such as YCbCr, HSV are used for fire and smoke detection. The fire is detected using YCbCr color model samples because it distinguishes luminance and chrominance. Y, Cb, Cr color channels are separated from RGB input image. A pixel is more likely a fire pixel if intensity of Y channel is greater than channel Cb and Cr.

6. Cheng (2011) [5] proposed a fire detection system based on Neural Network; here neural network is used in detection information for temperature, CO concentration, and smoke density to determine probability of three representative fire conditions. RBF neuron structure is used, the information regarding temperature, CO concentration, and smoke density are collected and data fusion is used to generate fire signal decision. The detectors have continuous analog outputs, when detection limit is exceeded the hardware circuit sends a local fire indication to fusion center, this force the system detectors to generate final decision. Single-sensor detector is used to generate the final decision.

7. Paulo Vinicius Koerich Borges [11] proposed a fire detection method based on probabilistic method and classification. Computer vision based approach is used in this approach. Though this approach is used surveillance it is also used to automatic video classification for retrieval of fire catastrophes in databases of newscast content. There are large variations in fire and background characteristics depending on the video instance. The proposed method observes the frame-to-frame changes of low-level features describing potential fire regions. These features include color, area size, surface coarseness, boundary roughness, and skewness within estimated fire regions. Bayes classifier [12] is used for fire recognition. In addition, apriori [12] knowledge of fire events captured in videos is used to significantly improve the results.

8. Akshata & Bhosale [7] proposed another method where Local Binary Pattern acts as a base for fire detection and Wavelet Decomposition is used to detect fire. Pixel level analysis is required in this method. This method uses YCbCr color model to detect fire. Detection is based on three phase. The first phase involves segmentation of image using LBP. LBP is a texture operator whose value is computed using image ’s center and neighboring pixel values. Further accuracy is improved using Wavelet Transform and complicated data is classified using this approach. 2D Discrete Wavelet Transform is used for decomposition in this system. 2 images should be used as input and the sub bands of every image are compared with the other, if sub bands are equal the images are same else different.

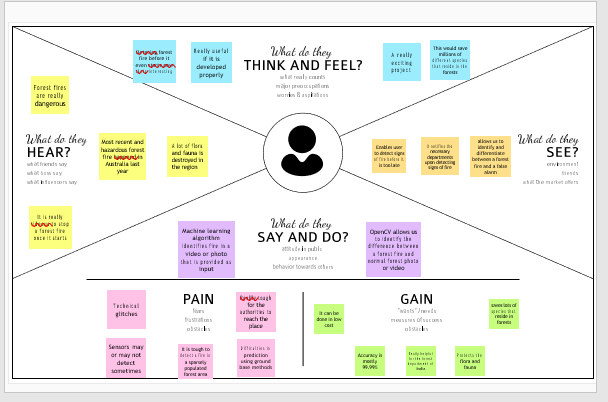
**PROBLEM STATEMENT DEFINITION**

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.

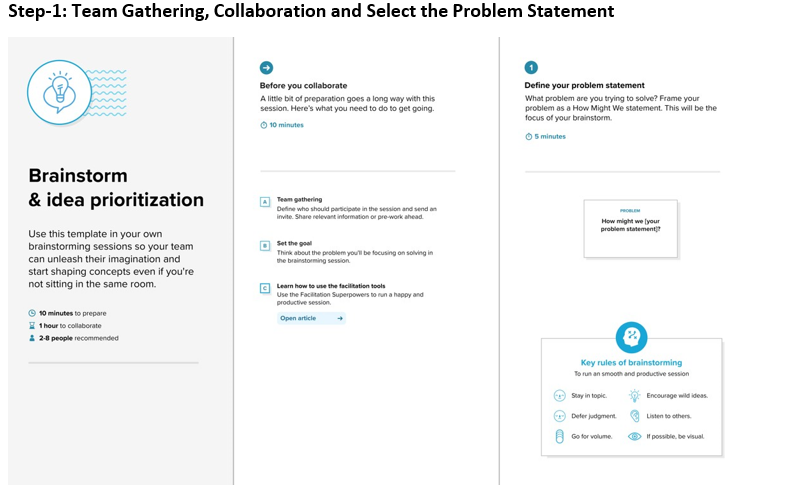
**IDEATION AND PROPOSED SOLUTION**

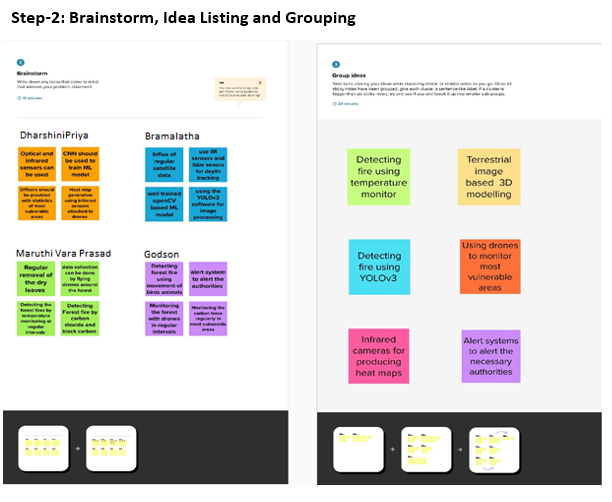
**EMPATHY MAP CANVAS**

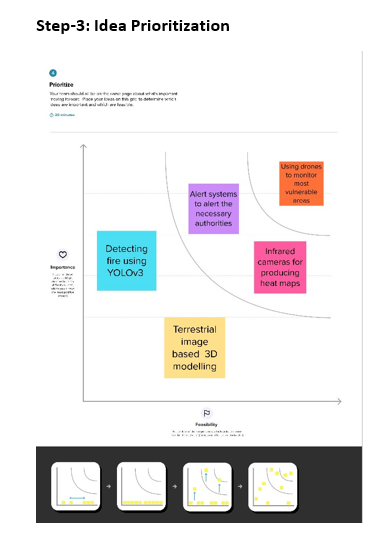


**IDEATION AND BRAINSTROMING**

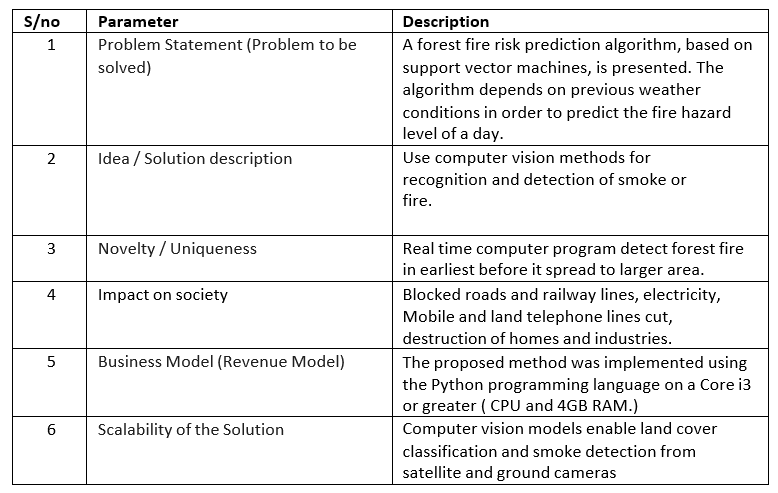
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.



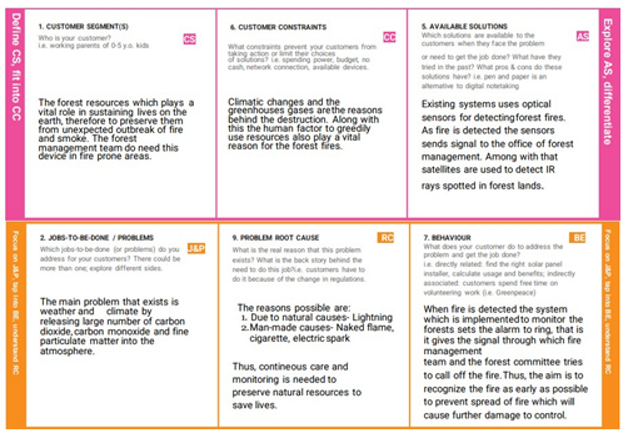


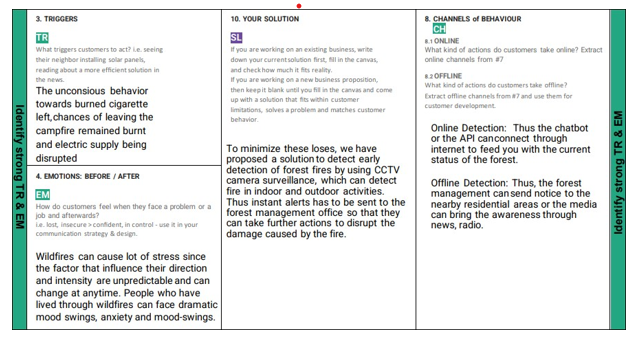


**PROPOSED SOLUTION**



**PROBLEM SOLUTION FIT**





**REQUIREMENT ANALYSIS**

**FUNCTIONAL REQUIREMENT**

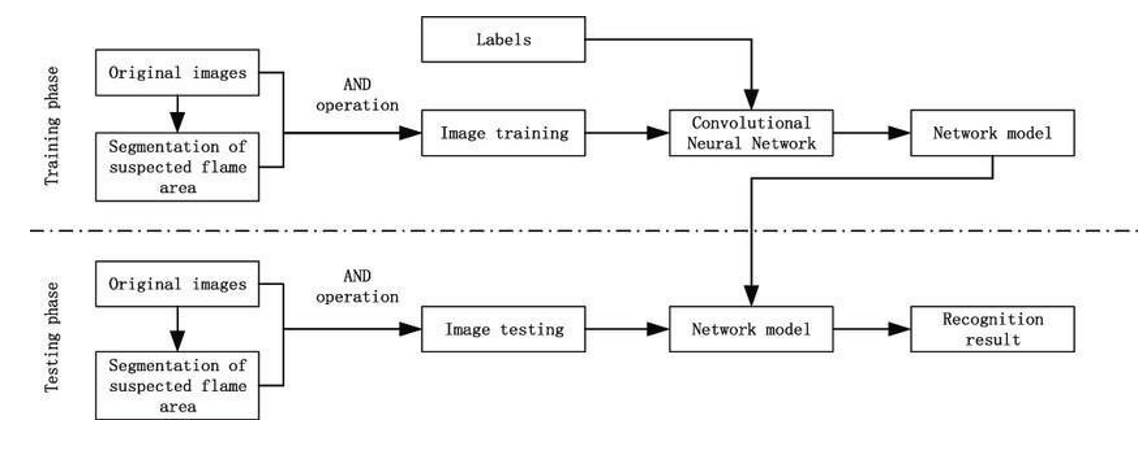
 High-resolution static cameras

**NON-FUNCTIONAL REQUIREMENT**

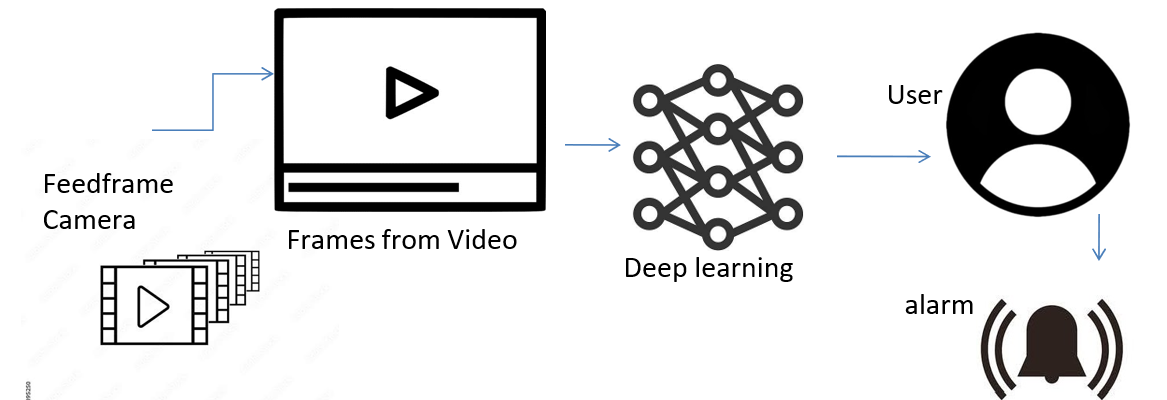
Train CNN model

**PROJECT DESIGN**

**DATA FLOW DIAGRAM**

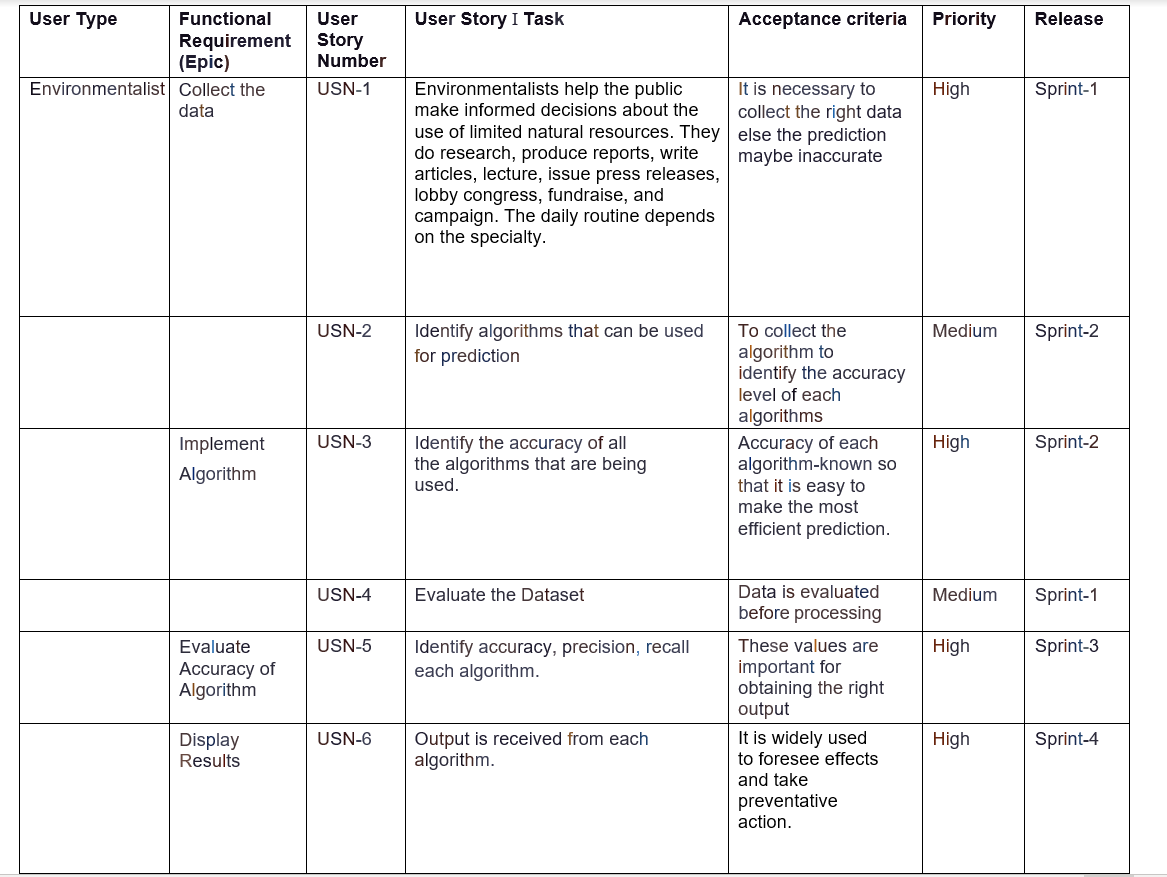


**SOLUTION AND TECHNICAL ARCHITECTURE**





**USER STORIES**



**PROJECT PLANNING AND SCHEDULING**

**SPRINT PLANNING AND ESTIMATION**

**Sprint 1:**

* Dataset – test set – image processing
* Testing in train – set
* Image processing - It is the method to perform some operation on an image in order to get an enhanced image

**Sprint 2:**

* Model building -> Add CNN layer, dense layer and model build predict
* Video analysis -> Capability of automatically analysis video to detect and determine the temporal and spatial events

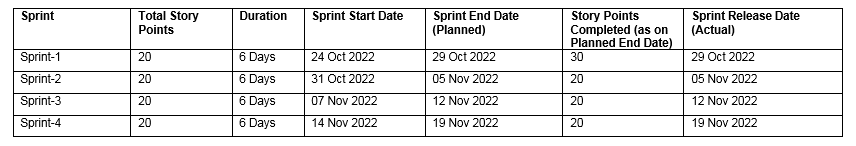
**Sprint 3:**

* Train CNN model on IBM
* Train image classification model
* It defines a set of target classes and train the model to recognize them using labelled example photos

**Sprint 4:**

* The exact location will be predicted and sent along with the alert message

**SPRINT DELIVERY SCHEDULE**



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

**AV = Sprint duration/Velocity**

**= 20/6**

**= 3**

**CODING AND SOLUTIONING**

**FEATURE 1**

**from** google.colab **import** drive

drive**.**mount('/content/drive/')

Mounted at /content/drive/

!unzip "/content/drive/MyDrive/dataset1.zip"

importing Required libraries

**import** keras

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

Define parameters and arguments for ImageDataGeneratior

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)

Applying ImageDataGenerator

x\_test **=** test\_datagen**.**flow\_from\_directory(r'/content/Dataset/Dataset/test\_set',

target\_size**=**(128,128),

batch\_size**=**32,

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

Found 121 images belonging to 2 classes.

x\_train **=** train\_datagen**.**flow\_from\_directory(r'/content/Dataset/Dataset/train\_set',

target\_size**=**(128,128),

batch\_size**=**32,

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

Found 436 images belonging to 2 classes.

**FEATURE 2**

**from** google.colab **import** drive

drive**.**mount('/content/drive/')

Mounted at /content/drive/

!unzip "/content/drive/MyDrive/dataset1.zip"

importing Required libraries

**import** keras

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

Define parameters and arguments for ImageDataGeneratior

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)

Applying ImageDataGenerator

x\_test **=** test\_datagen**.**flow\_from\_directory(r'/content/Dataset/Dataset/test\_set',

target\_size**=**(128,128),

batch\_size**=**32,

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

Found 121 images belonging to 2 classes.

x\_train **=** train\_datagen**.**flow\_from\_directory(r'/content/Dataset/Dataset/train\_set',

target\_size**=**(128,128),

batch\_size**=**32,

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

Found 436 images belonging to 2 classes.

Model building

*#import model building libraries*

*#to define linear initialisation import sequential*

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

*#to add layer import Dense*

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

*#to create convolution kernel import convolution2D*

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

*#import Maxpooling layer*

**from** keras.layers **import** MaxPool2D

*#import flatten layer*

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

**import** warnings

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

*#initialize the model*

model**=**Sequential()

**from** keras.layers.pooling.max\_pooling2d **import** MaxPooling2D

*#add convolution 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())

**DATABASE SCHEMA**

|  |
| --- |
| Importing The ImageDataGenerator Library |
|  |  |
|  | !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: tensorboard<2.10,>=2.9 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.1) |
|  | Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.0) |
|  | Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.2.0) |
|  | Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0) |
|  | Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from tensorflow) (21.3) |
|  | Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0) |
|  | Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.0) |
|  | Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.21.6) |
|  | Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (14.0.6) |
|  | Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.4.0) |
|  | Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.1.0) |
|  | Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0) |
|  | Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.19.6) |
|  | Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from tensorflow) (57.4.0) |
|  | Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2) |
|  | Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.6.3) |
|  | Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (4.1.1) |
|  | Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.14.1) |
|  | Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.3.0) |
|  | Requirement already satisfied: flatbuffers<2,>=1.12 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.12) |
|  | 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: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.50.0) |
|  | Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.7/dist-packages (from astunparse>=1.6.0->tensorflow) (0.38.3) |
|  | Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py>=2.9.0->tensorflow) (1.5.2) |
|  | Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (3.4.1) |
|  | Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (0.6.1) |
|  | Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.8.1) |
|  | Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.23.0) |
|  | Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (0.4.6) |
|  | Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.0.1) |
|  | Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.14.1) |
|  | Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (4.9) |
|  | Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (0.2.8) |
|  | Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (5.2.0) |
|  | Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow) (1.3.1) |
|  | Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (4.13.0) |
|  | Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (3.10.0) |
|  | Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (0.4.8) |
|  | Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (1.24.3) |
|  | Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (3.0.4) |
|  | Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2.10) |
|  | Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2022.9.24) |
|  | Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow) (3.2.2) |
|  | Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->tensorflow) (3.0.9) |
|  | Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/ |
|  | Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (4.6.0.66) |
|  | Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python) (1.21.6) |
|  | Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/ |
|  | Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.7/dist-packages (4.6.0.66) |
|  | Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-contrib-python) (1.21.6) |
|  | Define the parameters/arguments for ImageDataGenerator class |
|  |  |
|  | train=ImageDataGenerator(rescale=1./255, |
|  | shear\_range=0.2, |
|  | rotation\_range=180, |
|  | zoom\_range=0.2, |
|  | horizontal\_flip=True) |
|  | train = ImageDataGenerator(rescale=1/255) |
|  | test = ImageDataGenerator(rescale=1/255) |
|  | Applying ImageDataGenerator functionality to trainset |
|  |  |
|  | train\_dataset = train.flow\_from\_directory("/content/drive/MyDrive/Dataset/train\_set", |
|  | target\_size=(128,128), |
|  | batch\_size = 32, |
|  | class\_mode = 'binary' ) |
|  | Found 436 images belonging to 2 classes. |
|  | Applying ImageDataGenerator functionality to testset |
|  |  |
|  | test\_dataset = test.flow\_from\_directory("/content/drive/MyDrive/Dataset/test\_set", |
|  | target\_size=(128,128), |
|  | batch\_size = 32, |
|  | class\_mode = 'binary' ) |
|  | Found 121 images belonging to 2 classes. |
|  | test\_dataset.class\_indices |
|  | {'forest': 0, 'with fire': 1} |
|  | Import model building libraries |
|  |  |
|  | #to define linear initialisation import sequential |
|  | from keras.models import Sequential |
|  | #to add layer import Dense |
|  | from keras.layers import Dense |
|  | #to create convolution kernel import convolution2D |
|  | from keras.layers import Convolution2D |
|  | #import Maxpooling layer |
|  | from keras.layers import MaxPooling2D |
|  | #import flatten layer |
|  | from keras.layers import Flatten |
|  | import warnings |
|  | warnings.filterwarnings('ignore') |
|  | Initializing the model |
|  |  |
|  | model =Sequential() |
|  | Add CNN Layer |
|  |  |
|  | model.add(Convolution2D(32,(3,3),input\_shape=(128,128,3),activation='relu')) |
|  | model.add(MaxPooling2D(pool\_size=(2,2))) |
|  | model.add(Convolution2D(32,(3,3),activation='relu')) |
|  | model.add(MaxPooling2D(pool\_size=(2,2))) |
|  | model.add(Convolution2D(32,(3,3),activation='relu')) |
|  | model.add(MaxPooling2D(pool\_size=(2,2))) |
|  | model.add(Convolution2D(32,(3,3),activation='relu')) |
|  | model.add(MaxPooling2D(pool\_size=(2,2))) |
|  | model.add(Flatten()) |
|  | Add Hidden Layer |
|  |  |
|  | model.add(Dense(150,activation='relu')) |
|  |  |
|  | model.add(Dense(1,activation='sigmoid')) |
|  | Configure the learning process |
|  |  |
|  | model.compile(loss = 'binary\_crossentropy', |
|  | optimizer = "adam", |
|  | metrics = ["accuracy"]) |
|  | Train the model |
|  |  |
|  | model.fit\_generator(x\_train,steps\_per\_epoch=14,epochs=5,validation\_data=x\_test,validation\_steps=4) |
|  | Epoch 1/5 |
|  | 14/14 [==============================] - 33s 2s/step - loss: 0.5697 - accuracy: 0.7018 - val\_loss: 0.2470 - val\_accuracy: 0.9421 |
|  | Epoch 2/5 |
|  | 14/14 [==============================] - 36s 3s/step - loss: 0.3486 - accuracy: 0.8280 - val\_loss: 0.1461 - val\_accuracy: 0.9752 |
|  | Epoch 3/5 |
|  | 14/14 [==============================] - 30s 2s/step - loss: 0.2088 - accuracy: 0.9060 - val\_loss: 0.0464 - val\_accuracy: 0.9917 |
|  | Epoch 4/5 |
|  | 14/14 [==============================] - 34s 2s/step - loss: 0.1883 - accuracy: 0.9128 - val\_loss: 0.0730 - val\_accuracy: 0.9669 |
|  | Epoch 5/5 |
|  | 14/14 [==============================] - 30s 2s/step - loss: 0.1682 - accuracy: 0.9220 - val\_loss: 0.0353 - val\_accuracy: 1.0000 |
|  | Save The Model |
|  |  |
|  | model.save("/content/drive/MyDrive/archive(1)/forest1.h5") |
|  | Predictions |
|  |  |
|  | predictions = model.predict(test\_dataset) |
|  | predictions = np.round(predictions) |
|  | 4/4 [==============================] - 6s 1s/step |
|  | predictions |

**TESTING**

**TEST CASES**

In the previous stage, the fire region candidates were extracted. However, these regions may not belong to fire objects, which results in misdetection. Hence, the classification stage is required to verify whether the region is a fire or a non-fire object. Two machine learning algorithms were implemented and compared in the experiment, such as support vector machines and random forest.

The support vector machine (SVM) is one of the most popular machine learning algorithms for binary classification , as it is relatively straightforward and simple but effective. Technically, the SVM finds the optimal hyperplane that distinguishes the two separated classes by maximizing their distance. This hyperplane is located in the middle of the two classes. For example, in fire detection problems, these two classes are defined as fire and non-fire.

On the other hand, random forest is a widely known technique to develop classification models [[**29**](https://www.mdpi.com/2571-6255/5/1/23/htm#B29-fire-05-00023)]. The random forest algorithm works by constructing several randomized decision trees, called predictors. First, each predictor is generated by using training sample data. Then, these predictors are aggregated for the final decision based on majority voting [[**30**](https://www.mdpi.com/2571-6255/5/1/23/htm#B30-fire-05-00023)]. For instance, suppose we generated a Random Forest model with three decision trees and would classify unknown data. The first and the second decision trees classified the data as fire class, while the last decision tree assigned the same data as a non-fire class. Therefore, the data are classified as a fire class in the final decision using majority voting.

**USER ACCEPTANCE TESTING**

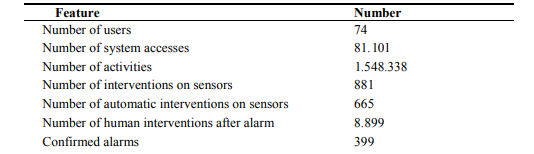
The features have been detected in a convolutional layer,their exact location becomes of less importance; what matters more is their relative positions compared to other features. To help the reader understand, in a handwritten number recognition system, if we understand our input image is made of a roughly horizontal segment in the upper left area, a corner in the upper right area, a roughly vertical segment stretching to the lower area then we can deduce the image corresponds to the number seven. Knowing the accurate position of all of these features is not only useless, but could also lead to not generalizing as well to other types of instances of the number seven since their exact position could vary. So, in order to reduce but not completely nullify the precision with which the position of different features is recorded in the feature map, a possible approach (the one taken here) is to reduce the spatial resolution of the feature map: this is the purpose of what are called subsampling layers (also sometimes referred to as pooling layers) in a convolutional network. Each of them conducts a local averaging and a subsampling, which has the effect of making the result more resilient to shifts and distortions. Also, in a subsampling layer, contrarily to a convolutional layer, the contiguous receptive fields of contiguous units do not overlap.

That is why convolutional and subsampling layers are generally alternated in CNNs: at each occurrence of a combination of convolutional and subsampling layer, the number of feature maps increases and the spatial resolution decreases. And thus, an important amount of invariance to geometric transformation of the input is achieved by compensating the progressive reduction of spatial resolution with a progressive increase of the richness in the representation ( i.e. the number of feature maps). CNNs have, since then, been the most used feature extractor for image tasks.

**RESULTS**

**PERFORMANCE METRICS**

Some numbers that illustrate the results of the forest fire observer network are presented. The numbers are collected after the first season of intelligent forest fire monitoring and surveillance system. The system was implemented on 43 locations in forests.



The table shows that system has been used intensively by users in this period and produced over million and a half activity logs. The majority of the sensors problems have been solved automatically, while only 25% of the problems required intervention. Human intervention after alarm describes situations when human operator inspected some event in the environment by manually checking the sensor and the number describes the number of true detections of the phenomenon.

**ADVANTAGES**

It can also **gauge temperature, humidity, and air pressure to create a climate map of the forest**. This map provides the means to assess the risk of fire.

**DISADVANTAGES**

In recent history and even the present day, several forest fire detection methods have been implemented, such as watchtowers, satellite image processing methods, optical sensors, and digital camera-based methods, although there are many drawbacks, such as **inefficiency, power consumption, latency, accuracy etc…**

#### CONCLUSION

A real-time and reliable fire detection method for an early warning system is required so that an immediate response to an incident can be made effective. In this study, methods based on colour probabilities and motion features were successfully implemented to achieve this goal. The proposed method exploits the characteristics of the colour of fire by developing a probability model using a multiple Gaussian. On the other hand, other fire characteristics, namely, dynamic fire movement model with motion features based on moment invariants, were also applied. The experiment found that the processing time required on average reached 21.70 FPS with a relatively high true positive rate of 89.92%. These results indicate that the proposed method is suitable for a real-time early warning system. Nonetheless, one of the greatest challenges in implementing the module is physically installing the camera, which may be very difficult. Therefore, it will remain a challenge for our further research

**FUTURE SCOPE**

In future, we are planning to install smart water tank system in dense forest where reachability of resources and firefighters is difficult. In addition to that we will be updating the system with more features and reliability. We will also include a high pitch sound system that will keep away the animals from the site of fire.

**APPENDIX**

**GITHUB &PROJECT DEMO LINK**

<https://drive.google.com/file/d/1-6_hPzW10R3PZTlG4SONYBhmcbtvoaWp/view?usp=sharing>

https://github.com/IBM-EPBL/IBM-Project-35319-1660283426.git