EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

A PROJECT REPORT

Submittedby

GOKULRAJ R (19EUEC046)
HARI PRIYAA VT(19EUEC047)
HARI SHRUTHI TK(19EUEC048)
HARI PRASATH AS(19EUEC049)

inpartialfulfillmentfortheawardofthedegreeof

BACHELOR OFENGINEERING

in

ELECTRONICS AND COMMUNICATIONENGINEERING

SRIKRISHNACOLLEGEOFENGINEERINGANDTECHNOLOGY
(An Autonomous Institution, Affiliated to Anna University Chennai - 600

025)NOVEMBER2022

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

SRI KRISHNA INSTITUTIONS COIMBATORE

BONAFIDECERTIFICATE

Certified that this project report titled **“EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES”**is the bonafide work of**Gokulraj R(19EUEC046) ,Hari Priyaa V T(19EUEC047),Hari Shruthi T K (19EUEC048) , HARIPRASATH A S(19EUEC049)** who carried out theprojectworkundermysupervision.

SIGNATURE **SIGNATURE**

Dr.S.SASIPRIYAM.E.,Ph.D., **Mr.C.VIVESVARAN M.E.,**

HEADOFTHEDEPARTMENT **SUPERVISOR**

Department of Electronics and Communication
Engineering
SriKrishnaCollegeofEngineeringandTechnology
,Kuniamuthur,Coimbatore

Submitted for the Project viva-voIce examination held on__________

INTERNALEXAMINER **EXTERNALEXAMINER**

ABSTRACT

Forest fires have been and still are serious problem for the European Union and for all other countries in Europe. In the year 2000, the EU has established the European Forest Fire Information system (EFFIS) [1], which will soon become part of the European Emergency Management Service, maintained by the Copernicus Earth Observation Programme [2]. This system provides valuable near real-time and also historical data on the forest fires in Europe, the Middle East and North Africa. Currently EFFIS is being used and supported with data by 25 EU member states and by numerous other countries. According to the annual report of EFFIS for 2016 [3], more than 54,000 forest fires have occurred all around Europe and they have led to nearly 376 thousand hectares of burnt areas. If we compare these values to the average values from the EEFIS reports for the period 2006-2015, the number of forest fires have decreased by 13327 or by the nearly 20%. This decease can be explained with the more severe actions and sanctions towards the arsonists and with the introduction of more advanced technical solutions for early detection of the fires. Even though their number is decreasing, the forest fire continue to be extremely devastating forest fire form 2018, which took place in the Attica region of Greece and led to more than 90 fatalities and to more than200 injured people, as well as to the destruction to thousands of buildings [4].

The most important factors in the fight against the forest fires include the earliest possible detection of the fire event, the proper categorization of the fire and fast response from the fire services. Several different types of forest fires are known, including ground fires, surface fires and crown/tree fires [5]. Each of these types of forest fires is specific and the proper counteractions against it must be considered and implemented to successfully fight it. Over the years the detection of forest fires has been conducted in different ways, ranging from the use of forest outposts to fully automated solutions.

TABLEOFCONTENTS

LISTOFFIGURES

1.INTRODUCTION

1.1 OVERVIEW

Forest fires have been and still are serious problem for the European Union and for all other countries in Europe. In the year 2000, the EU has established the European Forest Fire Information system (EFFIS) [1], which will soon become part of the European Emergency Management Service, maintained by the Copernicus Earth Observation Programme [2]. This system provides valuable near real-time and also historical data on the forest fires in Europe, the Middle East and North Africa. Currently EFFIS is being used and supported with data by 25 EU member states and by numerous other countries. According to the annual report of EFFIS for 2016 [3], more than 54,000 forest fires have occurred all around Europe and they have led to nearly 376 thousand hectares of burnt areas. If we compare these values to the average values from the EEFIS reports for the period 2006-2015, the number of forest fires have decreased by 13327 or by the nearly 20%. This decease can be explained with the more severe actions and sanctions towards the arsonists and with the introduction of more advanced technical solutions for early detection of the fires. Even though their number is decreasing, the forest fire continue to be extremely devastating forest fire form 2018, which took place in the Attica region of Greece and led to more than 90 fatalities and to more than200 injured people, as well as to the destruction to thousands of buildings [4].

The most important factors in the fight against the forest fires include the earliest possible detection of the fire event, the proper categorization of the fire and fast response from the fire services. Several different types of forest fires are known, including ground fires, surface fires and crown/tree fires [5]. Each of these types of forest fires is specific and the proper counteractions against it must be considered and implemented to successfully fight it. Over the years the detection of forest fires has been conducted in different ways, ranging from the use of forest outposts to fully automated solutions.

2.LITERATURE SURVEY

2.1 EXISTING SYSTEM

The existing system a novel system for detecting fire using Convolution Neural Networks (CNN). Detection of fire can be extremely difficult using existing methods of smoke sensors installed in the buildings. They are slow and cost inefficient due to their primitive design and technology. This paper critically analyzes the scope of Artificial intelligence for detection and sending alerts with video from CCTV footages. This project uses self-built dataset containing video frames with fire. The data is then pre processed and use the CNN to build a machine learning model. The test set of the dataset is given as input for validating the algorithm and experiments are noted. The project focus on building cost efficient and highly accurate machine that can be used in almost any use case of fire detection.

Existing system they are used Robots for Extinguish fire robot is interfaced with several sensors

- Unmanned Ariel Vehicle (UAV) – For monitoring Forest fire.
- Unmanned Ground Vehicle (UGV) – For extinguish fire.
- Both Vehicles are communicating through Radio Frequency Communication if UAV is detecting fire means it will send to UGV.
- UGV will come to that spot and extinguish fire.

2.2 REFERENCES

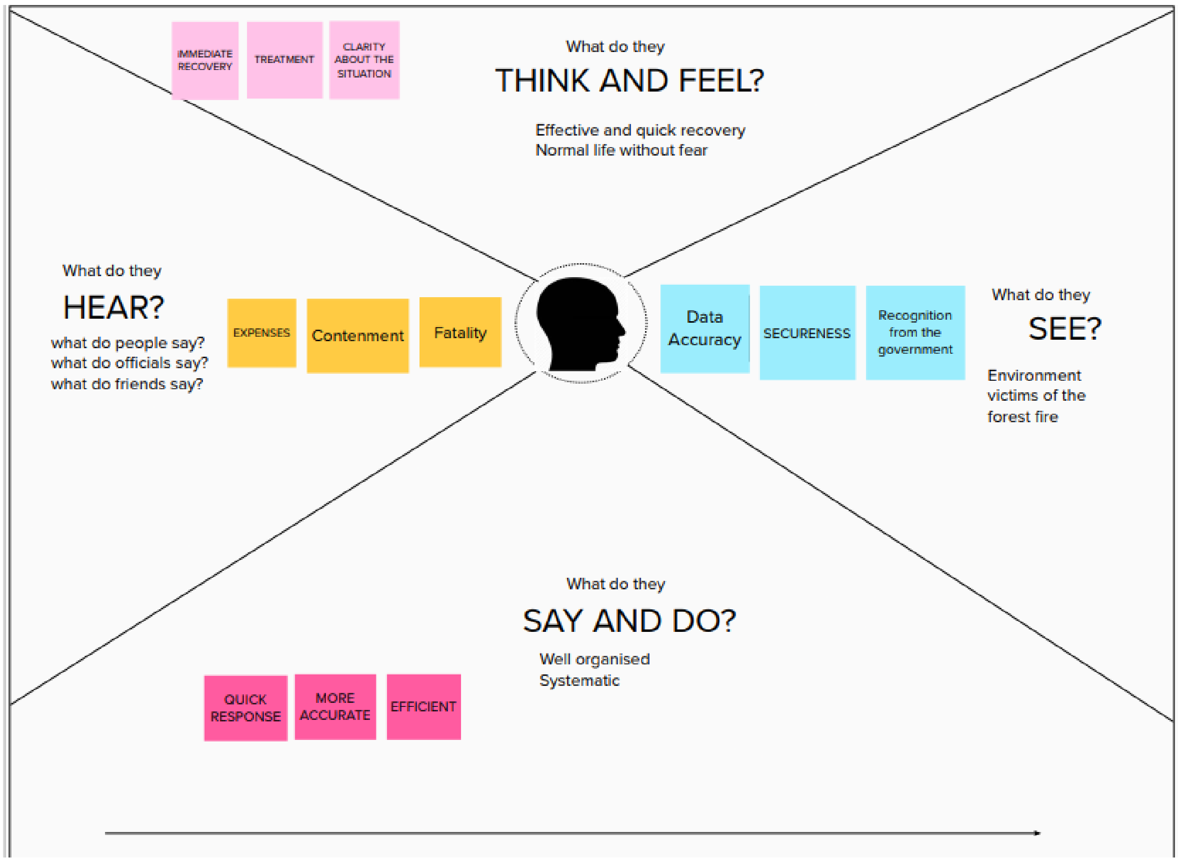
Burned areas in forest fires were predicted using estimation methods as the Multi layer Perceptron (MLP), SVM, Radial Basis Function (RBF) networks and fuzzy logic. The results indicate that MLP gives more accurate results. Available and Reliable Storage for an Incompletely Trusted Environment (FARSIGHT) simulator was used to predict forest fires spread in the Euro-Mediterranean countries. The outputs of FARSIGHT were obtained by two models, custom fuel model and standard fuel model. The experimental results showed that the accuracy of the custom fuel model was better than the standard fuel model. An intelligent system called geometric semantic genetic programming to predict burned areas. The results obtained using that intelligent systems were better than using standard genetic programming. A novel system called forecast to predict the spread of forest fires in the future. The forecast is a system that combines Artificial Intelligence (AI) and Geographic Information Systems (GIS). The forecast obtained more accurate results when compared to other random prediction models. A machine-learning algorithm based on Wireless Sensor Networks (WSN) to predict forest fires. A fire prediction tool called Disjunctive Normal Form (DNF) model to predict forest fires. The results obtained from the DNF model were compared with other machine learning models as naive Bayes, decision tree, SVM, RBF, and polynomial kernel functions. The DNF model gave the highest average accuracy with 97.8% among the other machine learning models. An algorithm that depends on SVM to predict forest fires. SVM used two class predictions of fire risk. The results demonstrated that the accuracy of SVM was approximately 96%. ANN model was used to predict the size of burned areas of forest fires in southern Spain. ANN was used in two stages: classifying forest fires size and evaluation of the burned surface areas. The results mentioned that the process of prediction was over 60%, prediction can reach more than 70% in some central areas. A probabilistic model was used to predict forest fires. There were three steps to design the probabilistic model. In step 1, the probabilistic model of forest fires was built from data of weather forecast and historical satellite. In step 2, the prediction of forest fires was produced using the data of the weather forecast as an input in the model of forest fires. In step 3, the warnings of forest fires were transported on different levels based on the need of the user. Machine learning models to predict the size of forest fires at the time of their inflammation. Decision trees, random forests, and MLP models were used in the process of prediction. The decision tree model predicted that 40% of the inflammation led to a large number of fires, and this per cent is about 75% of the total burned area. Random forests and MLP models were tested, but they did not perform the accuracy as the decision tree model. Different machine learning models to predict forest fires in Slovenia. Logistic regression, decision tree, random forests, bagging, and boosting of decision tree models were used to predict forest fires in Slovenia. These models were applied to these three data sets: Kras region, Primorska region, and continental Slovenia. From the experimental results, the bagging decision tree model obtained the best accuracy for all the data sets. Semi-parametric models were used predict forest fires.

2.3 PROBLEM STATEMENT

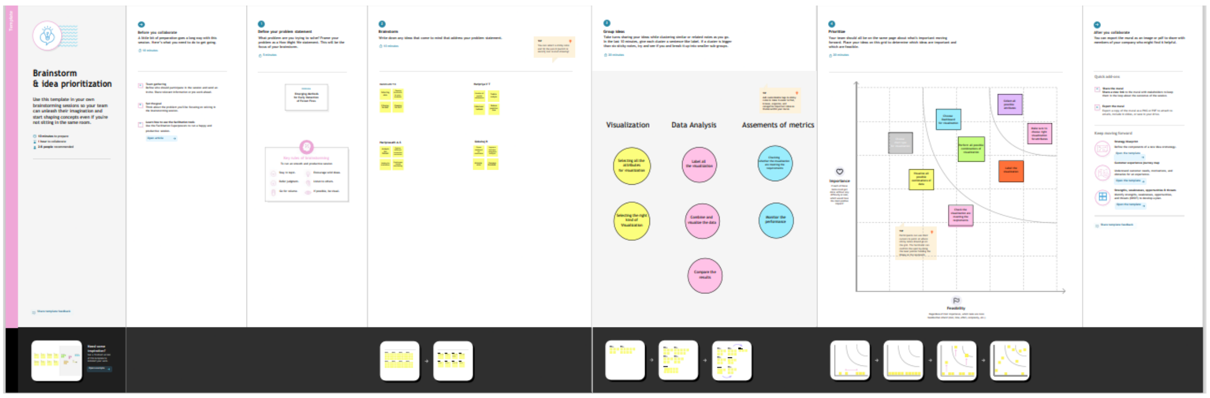
Forest fires are a major environmental issue, creating economic and ecological damage while endangering human lives. There are typically about 100,000 wildfires in the United States every year. Over 9 million acres of land have been destroyed due to treacherous wildfires. It is difficult to predict and 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 AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



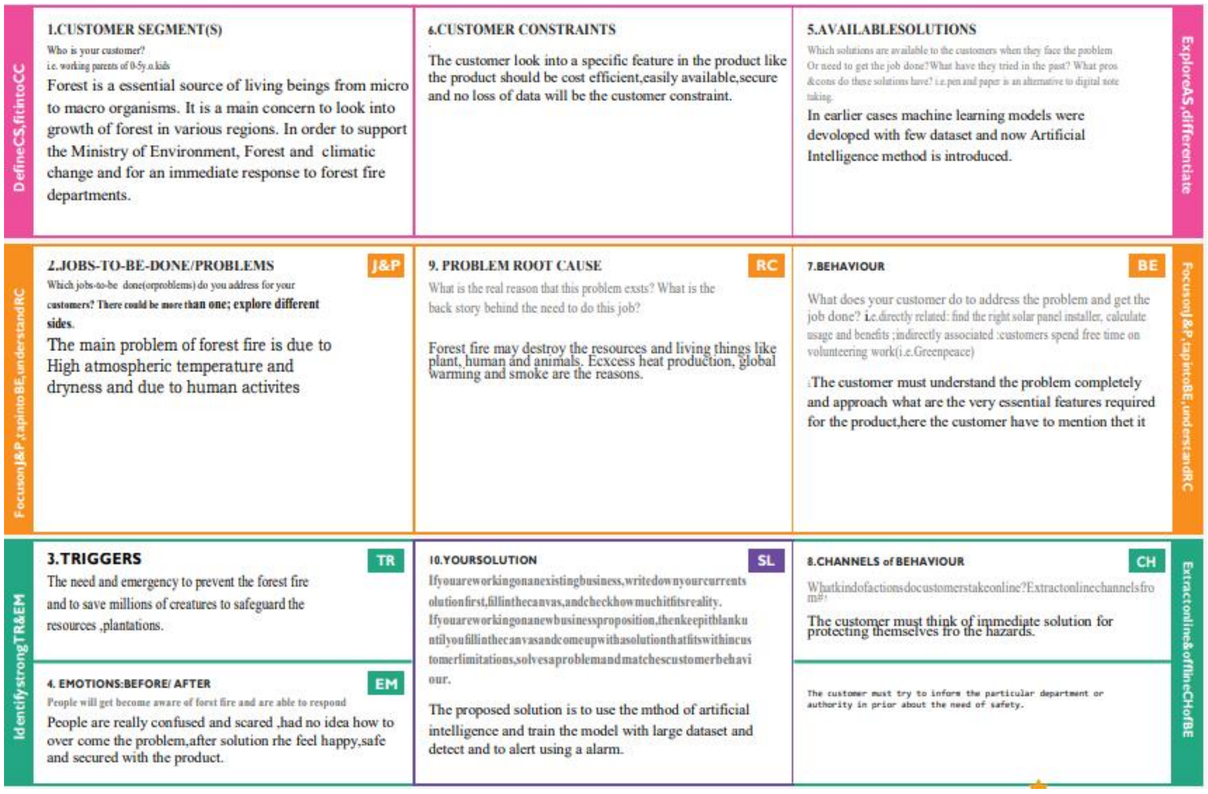
3.2 IDEATION AND BRAINSTORMING



3.3 PROPOSED SOLUTION

S/no	Parameter	Description
1	Problem Statement (Problem to be solved)	A forest fire risk prediction algorithm, based on support vector machines, is presented. The algorithm depends on previous weather conditions in order to predict the fire hazard level of a day.
2	Idea / Solution description	Use computer vision methods for recognition and detection of smoke or fire.
3	Novelty / Uniqueness	Real time computer program detect forest fire in earliest before it spread to larger area.
4	Impact on society	Blocked roads and railway lines, electricity, mobile and land telephone lines cut, destruction of homes and industries.
5	Business Model (Revenue Model)	The proposed method was implemented using the Python programming language on a Core i3 or greater (CPU and 4GB RAM.)
6	Scalability of the Solution	Computer vision models enable land cover classification and smoke detection from satellite and ground cameras

3.4 PROPOSED SOLUTION FIT



DefineCS,fitintoCC

1.CUSTOMER SEGMENT(S)

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

Forest is a essential source of living beings from micro to macro organisms. It is a main concern to look into growth of forest in various regions. In order to support the Ministry of Environment, Forest and climatic change and for an immediate response to forest fire departments.

6.CUSTOMER CONSTRAINTS

The customer look into a specific feature in the product like the product should be cost efficient,easily available,secure and no loss of data will be the customer constraint.

5.AVAILABLESOLUTIONS

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 note taking.

In earlier cases machine learning models were devoloped with few dataset and now Artificial Intelligence method is introduced.

ExploreAS,differentiate

FocusonJ&P,tapintoBE,understandRC

2.JOBS-TO-BE-DONE/PROBLEMS J&P

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

The main problem of forest fire is due to High atmospheric temperature and dryness and due to human activites

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exsts? What is the back story behind the need to do this job?

Forest fire may destroy the resources and living things like plant, human and animals. Ecxcess heat production, global warming and smoke are the reasons.

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)

The customer must understand the problem completely and approach what are the very essential features required for the product,here the customer have to mention thet it

IdentifystrongTR&EM

3.TRIGGERS TR

The need and emergency to prevent the forest fire and to save millions of creatures to safeguard the resources ,plantations.

4. EMOTIONS:BEFORE/ AFTER EM

People will get become aware of forst fire and are able to respond

People are really confused and scared ,had no idea how to over come the problem,after solution rhe feel happy,safe and secured with the product.

10.YOURSOLUTION SL

Ifyouareworkingonanexistingbusiness,writedownyourcurrentsolutionfirst,fillinthecanvas,andcheckhowmuchitfitsreality. Ifyouareworkingonanewbusinessproposition,thenkeepitblankuntilyoufillinthecanvasandcomeupwithasolutionthatfitswithincustomerlimitations,solvesaproblemandmatchescustomerbehaviour.

The proposed solution is to use the mthod of artificial intelligence and train the model with large dataset and detect and to alert using a alarm.

8.CHANNELS of BEHAVIOUR CH

Whatkindofactionsdocustomerstakeonline?Extractonlinechannelsfrom#7

The customer must think of immediate solution for protecting themselves fro the hazards.

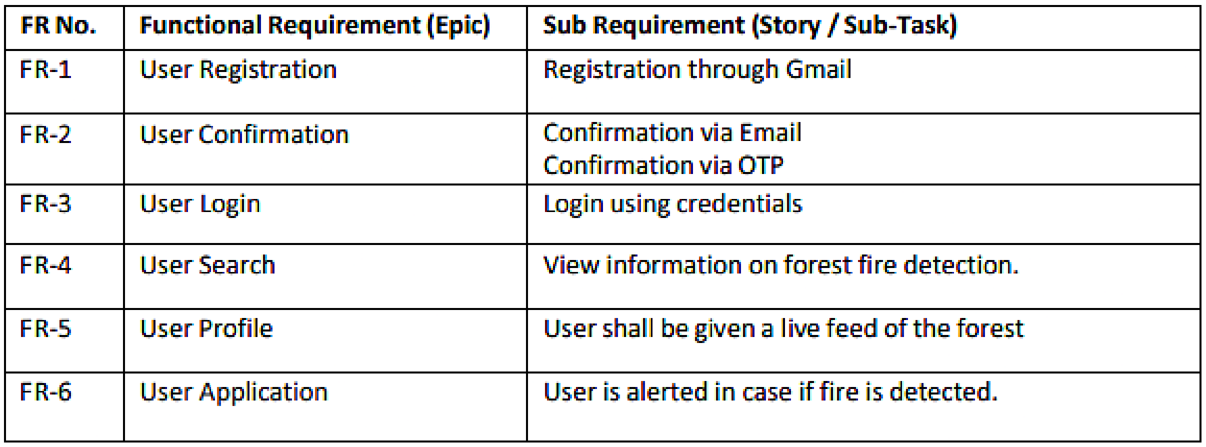
The customer must try to inform the particular department or authority in prior about the need of safety.

Extractonline&offlineCHofBE

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

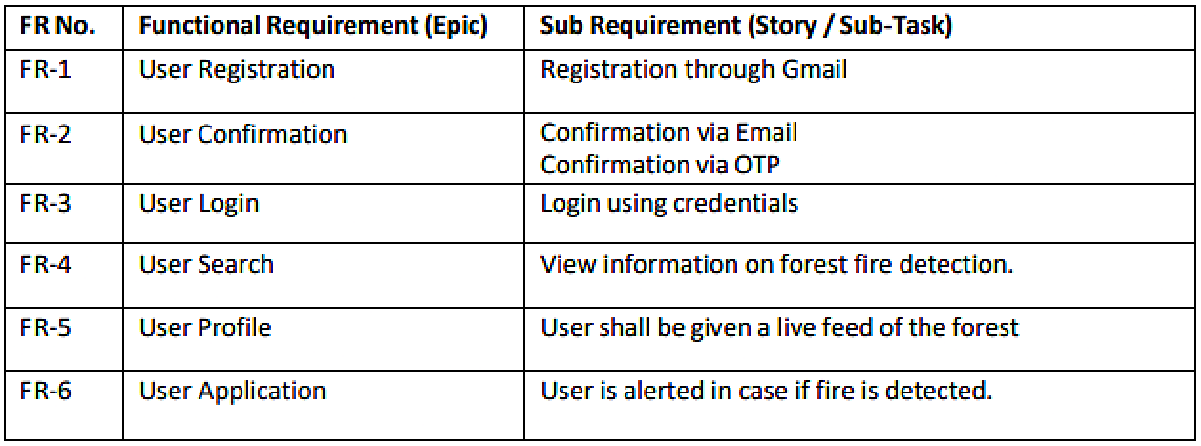
The user must register with the email,Confirmation via Email Confirmation via OTP ,Login using credentials,View information on forest fire detection,User shall be given a live feed of the forest ,User is alerted in case if fire is detected.



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login using credentials
FR-4	User Search	View information on forest fire detection.
FR-5	User Profile	User shall be given a live feed of the forest
FR-6	User Application	User is alerted in case if fire is detected.

4.2 NON FUNCTIONAL REQUIEMENT

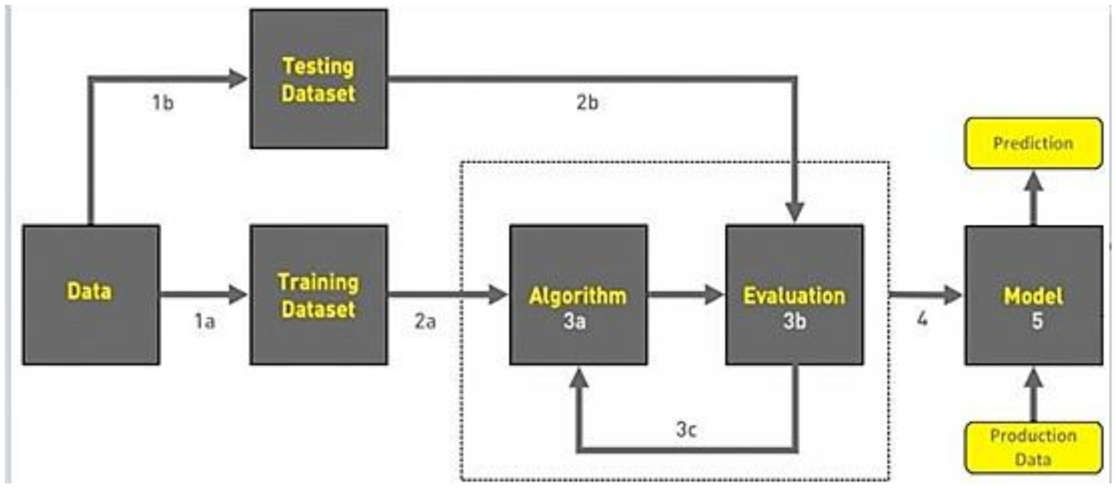
Registration through Gmail,Confirmation via Email Confirmation via OTP ,Login using credentials,View information on forest fire detection,User shall be given a live feed of the forest ,User is alerted in case if fire is detected..

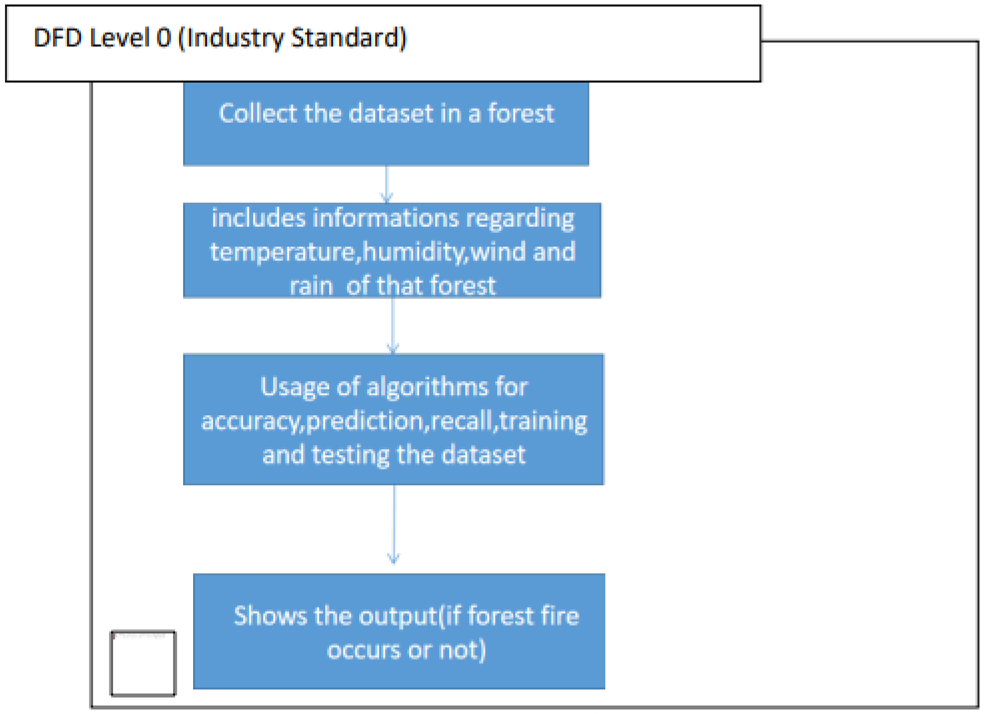


FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login using credentials
FR-4	User Search	View information on forest fire detection.
FR-5	User Profile	User shall be given a live feed of the forest
FR-6	User Application	User is alerted in case if fire is detected.

5.PROJECT DESIGN

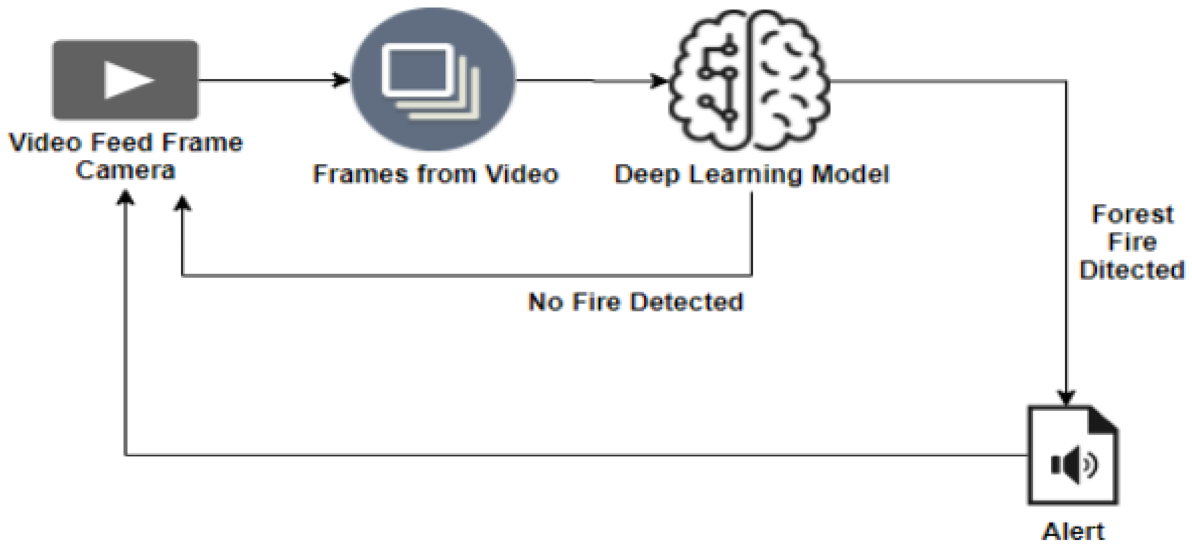
5.1 DATA FLOW DIAGRAMS



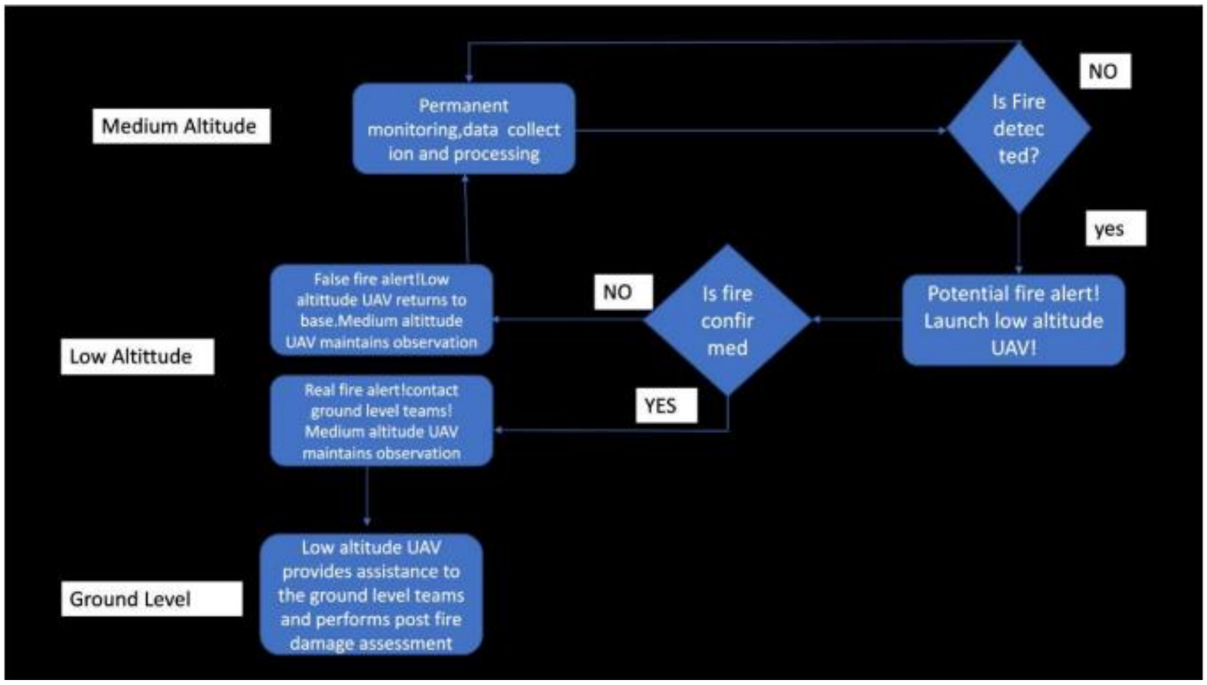

DFD Level 0 (Industry Standard)
Collect the dataset in a forest
includes informations regarding temperature,humidity,wind and rain of that forest
Usage of algorithms for accuracy,prediction,recall,training and testing the dataset
Shows the output(if forest fire occurs or not)

1. COLLECT DATA
2. EVALUATE DATA SET
3. IMPLEMENT ALGORITHMS
4. EVALUATE THE ACCURACYOF EACH ALGORITHMS
5. DISPLAY RESULTS

5.2 SOLUTION AND TECHNICAL ARCHITECTURE



WORK FLOW


Medium Altitude
Permanent monitoring,data collect ion and processing
Is Fire detec ted?
NO
yes
Low Altittude
False fire alert!Low altittude UAV returns to base.Medium altittude UAV maintains observation
NO
Is fire confir med
Potential fire alert! Launch low altitude UAV!
Real fire alert!contact ground level teams! Medium altitude UAV maintains observation
YES
Ground Level
Low altitude UAV provides assistance to the ground level teams and performs post fire damage assessment

5.3 USER STORIES

User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story/ Task**	**Acceptancecriteria**	**Priority**	**Release**
Environmentalist	Collect the data	USN-1	As an Environmentalist,it is necessary to collect the data of the forest which includes temperature,humidity,wind and rain of the forest	It is necessary to collect the right data else the prediction may become wrong	High	Sprint-1
		USN-2	Identify algorithms that can be used for prediction	To collect the algorithm to identify the accuracy level of each algorithms	Medium	Sprint-2
		USN-3	Identify the accuracy of each algorithms	Accuracy of each algorithm- calculated so that it is easy to obtain the most accurate output	High	Sprint-2
		USN-4	Evaluate the Dataset	Data is evaluated before processing	Medium	Sprint-1
		USN-5	Identify accuracy,precision,recall of each algorithms	These values are important for obtaining the right output	High	Sprint-3
		USN-6	Outputs from each algorithm are obtained	It is highly used to predict the effect and to take precautionary measures.	High	Sprint-4

6.PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

User Type	Functional Requirement(Epic)	User Story Number	User Story/ Task	Acceptance criteria	Priority	Release
Environmentalist	Collect the data	USN-1	As an Environmentalist,it is necessary to collect the data of the forest which includes temperature ,humidity,wind and rain of the forest	It is necessary to collect the right data else the prediction may become wrong	High	Sprint-1
		USN-2	Identify algorithms that can be used for prediction	To collect the algorithm to identify the accuracy level of each algorithms	Medium	Sprint-2
		USN-3	Identify the accuracy of each algorithms	Accuracy of each algorithm-calculated so that it is easy to obtain the most accurate output	High	Sprint-2
		USN-4	Evaluate the Dataset	Data is evaluated before processing	Medium	Sprint-1
		USN-5	Identify accuracy,precision,recall of each algorithms	[illegible]	High	Sprint-3
		USN-6	Outputs from each algorithm are obtained	[illegible]	High	Sprint-4

6.2 Sprint Delivery Schedule

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date(Planned)**		**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	19	1/11/2022
Sprint-3	20	6Days	07 Nov 2022	12 Nov 2022	19	12/11/2022
Sprint-4	20	6Days	14 Nov 2022	19 Nov 2022	20	15/11/2022

7.CODING AND SOLUTIONING

CONVOLUTIONAL NEURAL NETWORK:

Convolution al neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:Convolutional layer, Pooling layer, Fully-connected (FC) layer. The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let’s assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution.The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

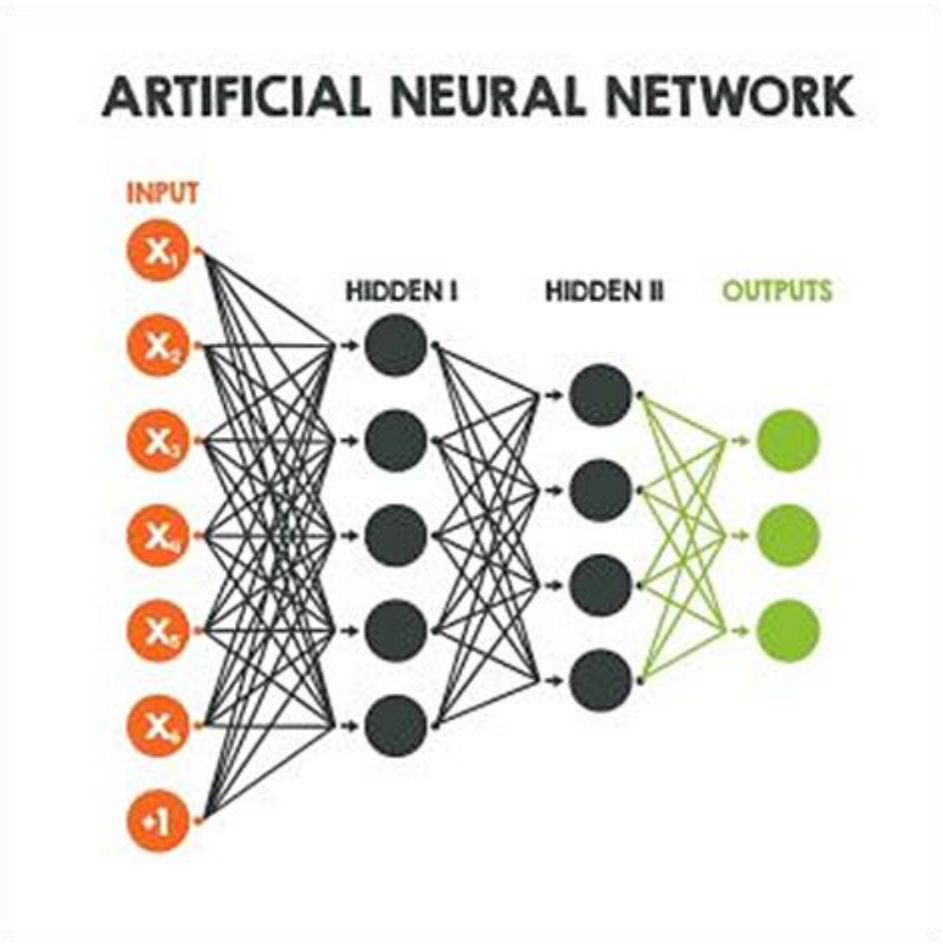
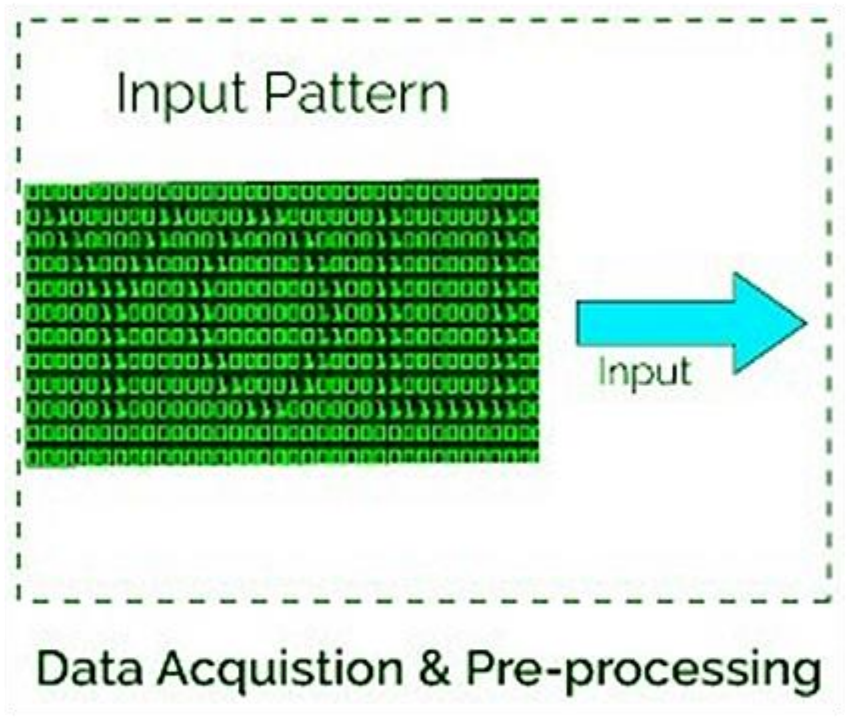


IMAGE PREPROCESSING:

Read the picture files (stored in data folder).Decode the JPEG content to RGB grids of pixels with channels. Convert these into floating-point tensors for input to neural nets.Rescale the pixel values (between 0 and 255) to the [0, 1] interval (as training neural networks with this range gets efficient.

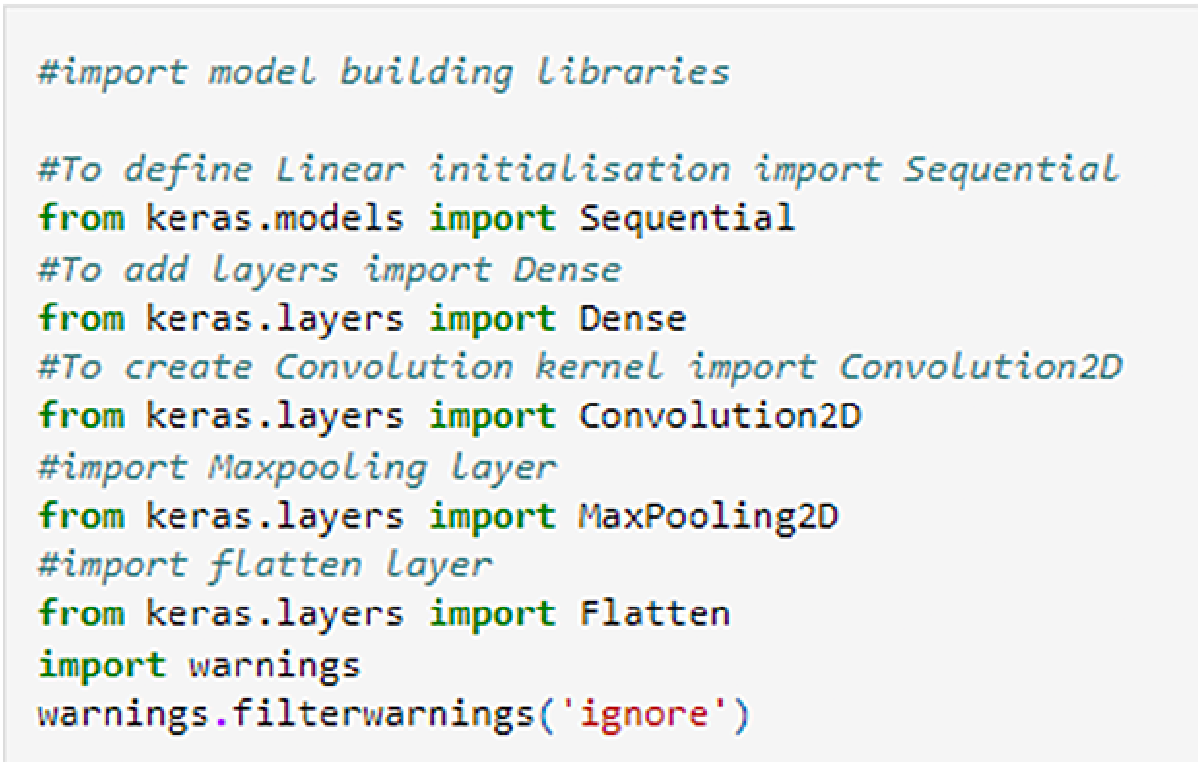
It may seem a bit fussy, but Keras has utilities to take over this whole algorithm and do the heavy lifting for you. Keras has a module with image-processing helping tools, located at keras.preprocessing.image. It contains the class *ImageDataGenerator*, which lets you quickly set up Python generators that can automatically turn image files on disk into batches of preprocessed tensors.



4.4 MODEL BUILDING:

MODEL BUILDING LIBRARIES:

These are importing some important libraries and the modules that are required for building the convolutional model. The Conv2D layer is the convolutional layer required to creating a convolution kernel that is convolved with the layer input to produce a tensor of outputs.



```
#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')
```

ADD CNN LAYERS

Filters

The primary purpose of convolution is to find features in the image using a feature detector. Then put them into a feature map, which preserves distinct features of images.

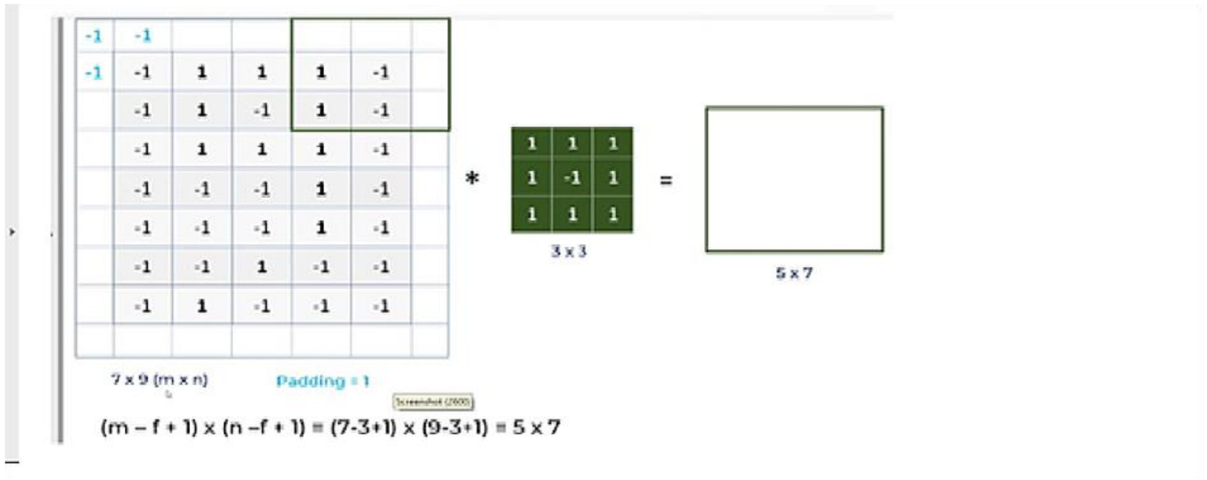
Feature detector which is known as a filter also is initialized randomly and then after a lot of iteration, filter matrix parameter selected which will be best for separating images. For instance, animals' eye, nose, etc. will be considered as a feature which is used for classifying images using filter or feature detectors. Here we are using 16 features.

Kernel_size

Kernel_size refers to filter matrix size. Here we are using a 2*2 filter size.

Padding

FIG 5



Let’s discuss what is problem with CNN and how the padding operation will solve the problem.

a. For a gray scale (n x n) image and (f x f) filter/kernel, the dimensions of the image resulting from a convolution operation is (n – f + 1) x (n – f + 1).

So for instances, a 5*7 image and 3*3 filter kernel size, the output result after convolution operation would be a size of 3*5. Thus, the image shrinks every time after the convolutional operation

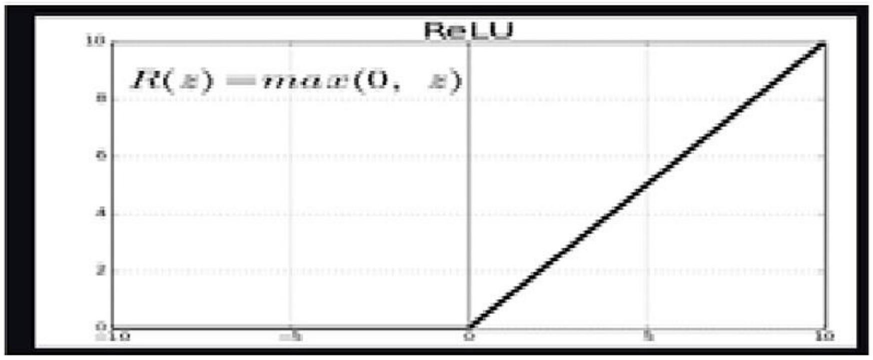
b. Pixels, located on corners are contributed very little compared to middle pixels.

So, then to mitigate these problems, padding operation is done. Padding is a simple process of adding layers with 0 or -1 to input images so to avoid above mentioned problems.

Here we are using Padding = Same arguments, which depicts that output images have the same dimensions as input images.

.

Activation Function – Relu



Since images are non-linear, to bring non-linearity, the relu activation function is applied after the convolutional operation.

Relu stands for Rectified linear activation function. Relu function will output the input directly if it is positive, otherwise, it will output zero.

Input shape

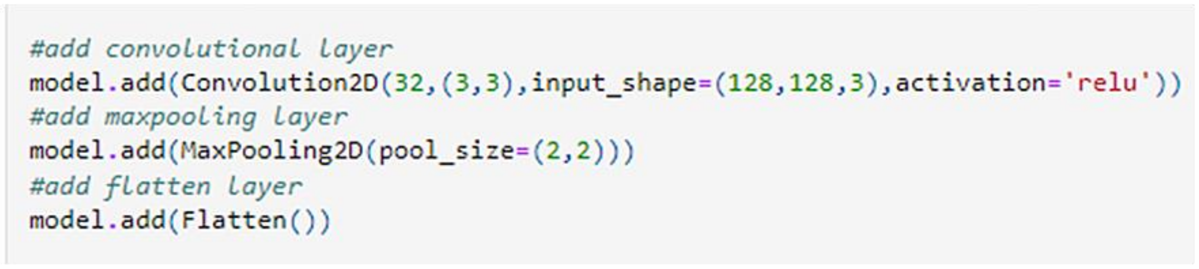
This argument shows image size – 224*224*3. Since the images in RGB format so, the third dimension of the image is 3.

Pooling Operation

Python Code :

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

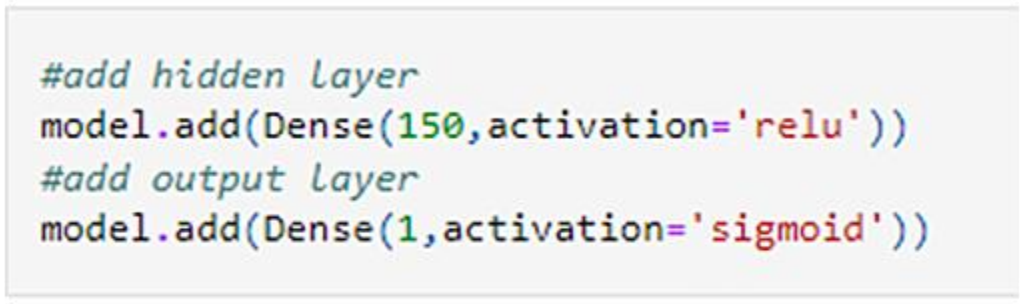
We need to apply the pooling operation after initializing CNN. Pooling is an operation of down sampling of the image. The pooling layer is used to reduce the dimensions of the feature maps. Thus, the Pooling layer reduces the number of parameters to learn and reduces computation in the neural network.



```
#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 DENSE LAYER

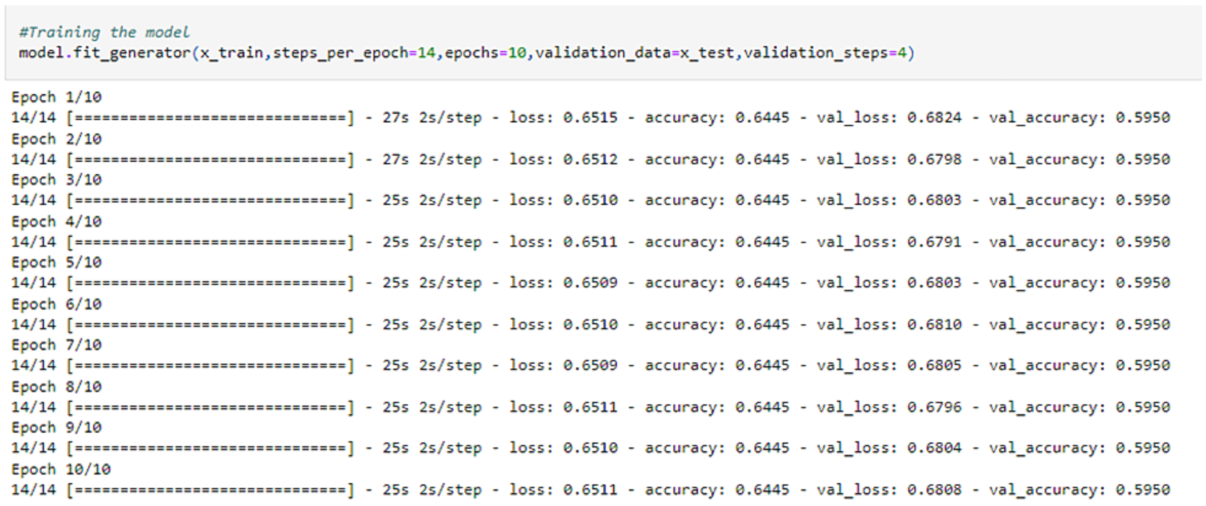
Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True). These are all attributes of Dense.



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

TRAINING THE MODEL:

he training step generates 3 datasets. 1) accuracy of the trained model, 2) the trained model, downloadable as a zip file, and 3) the trained model weights, downloadable as an hdf5 file. These files are needed for prediction in the next step.

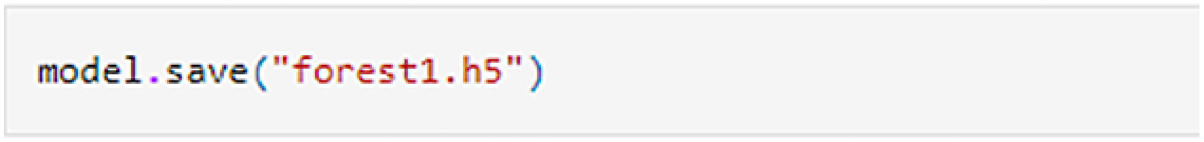


```
#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_steps=4)
```

```
Epoch 1/10
14/14 [==============================] - 27s 2s/step - loss: 0.6515 - accuracy: 0.6445 - val_loss: 0.6824 - val_accuracy: 0.5950
Epoch 2/10
14/14 [==============================] - 27s 2s/step - loss: 0.6512 - accuracy: 0.6445 - val_loss: 0.6798 - val_accuracy: 0.5950
Epoch 3/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accuracy: 0.6445 - val_loss: 0.6803 - val_accuracy: 0.5950
Epoch 4/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accuracy: 0.6445 - val_loss: 0.6791 - val_accuracy: 0.5950
Epoch 5/10
14/14 [==============================] - 25s 2s/step - loss: 0.6509 - accuracy: 0.6445 - val_loss: 0.6803 - val_accuracy: 0.5950
Epoch 6/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accuracy: 0.6445 - val_loss: 0.6810 - val_accuracy: 0.5950
Epoch 7/10
14/14 [==============================] - 25s 2s/step - loss: 0.6509 - accuracy: 0.6445 - val_loss: 0.6805 - val_accuracy: 0.5950
Epoch 8/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accuracy: 0.6445 - val_loss: 0.6796 - val_accuracy: 0.5950
Epoch 9/10
14/14 [==============================] - 25s 2s/step - loss: 0.6510 - accuracy: 0.6445 - val_loss: 0.6804 - val_accuracy: 0.5950
Epoch 10/10
14/14 [==============================] - 25s 2s/step - loss: 0.6511 - accuracy: 0.6445 - val_loss: 0.6808 - val_accuracy: 0.5950
```

SAVE THE MODEL:

Save your model by calling the *save()* function on the model and specifying the filename. The example below demonstrates this by first fitting a model, evaluating it, and saving it to the file *model.h5*.



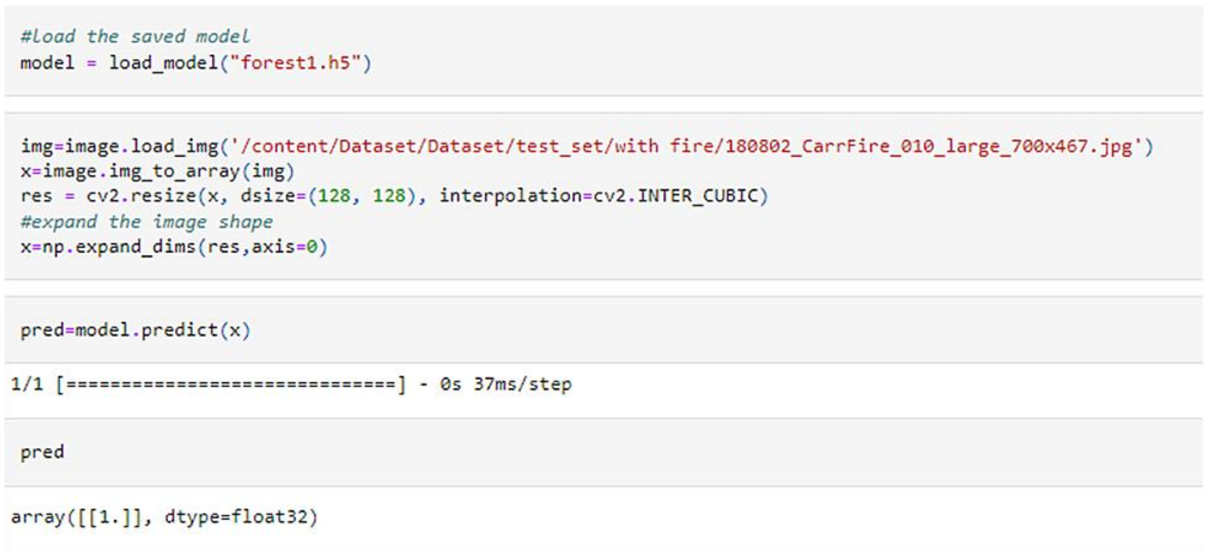
```
model.save("forest1.h5")
```

PREDICTION OF THE MODEL:

There are the following six steps to determine what object does the image contains?

- Load an image.
- Resize it to a predefined size such as 224 x 224 pixels.
- Scale the value of the pixels to the range [0, 255].
- Select a pre-trained model.

- Run the pre-trained model.
- Display the results



```
#load the saved model
model = load_model("forest1.h5")
```

```
img=image.load_img('/content/Dataset/Dataset/test_set/with fire/180802_CarrFire_010_large_700x467.jpg')
x=image.img_to_array(img)
res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)
```

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

```
1/1 [==============================] - 0s 37ms/step
```

```
pred
```

```
array([[1.]], dtype=float32)
```

VIDEO ANALYSIS

OpenCV for Video Processing

OpenCV is an open-source library that provides us with the tools to perform almost any kind of image and video processing.

Task 1: Capture Video from Camera

Often, we have to capture the live stream with a camera. OpenCV provides a very simple interface to this. Let’s capture a video from the camera (I am using the in-built webcam of my laptop), convert it into grayscale video, and display it.

To capture a video, you need to create a **Video Capture** object. Its argument can be either the device index or the name of a video file. The device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So I simply pass 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But in the end, don’t forget to release the capture. To read web cam will see the code.

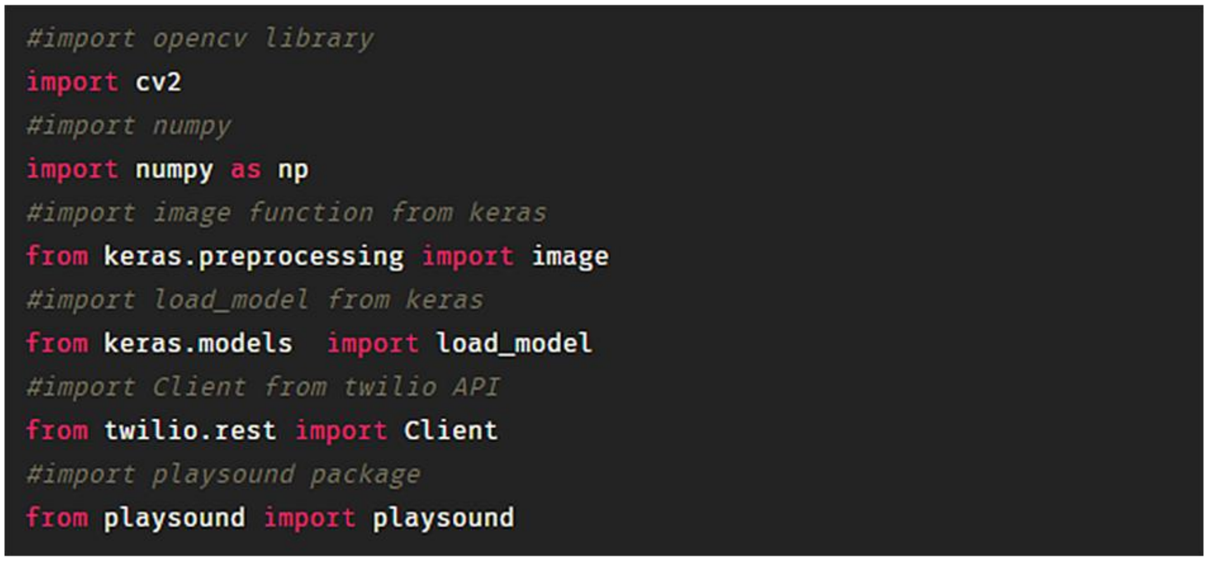
OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- **Core functionality** (**core**) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- **Image Processing** (**imgproc**) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- **Video Analysis** (**video**) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- **Camera Calibration and 3D Reconstruction** (**calib3d**) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- **2D Features Framework** (**features2d**) - salient feature detectors, descriptors, and descriptor matchers.
- **Object Detection** (**objdetect**) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- **High-level GUI** (**highgui**) - an easy-to-use interface to simple UI capabilities.
- **Video I/O** (**videoio**) - an easy-to-use interface to video capturing and video codecs.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

Task 2: Importing the required libraries.

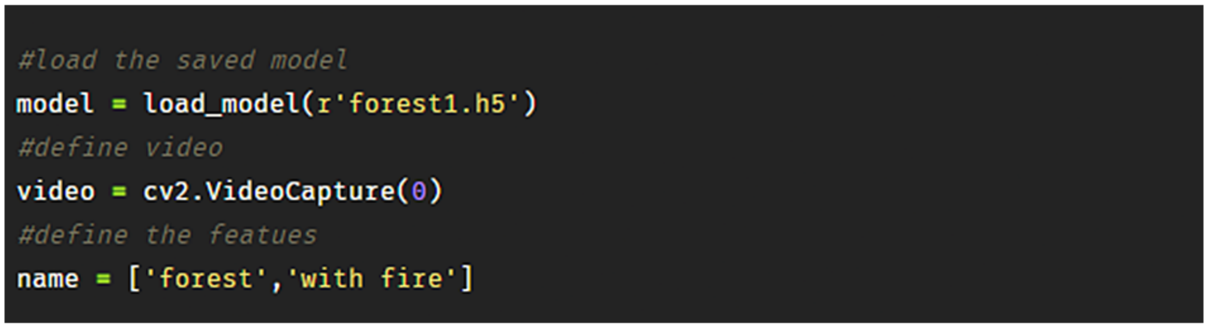
Install Twilio library, run the below command in anaconda prompt,

“pip install twilio”.



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

Task 3: Loading our saved model file using load_model from Keras library



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

TWILIO SERVICE:

Twilio is a customer engagement platform used by hundreds of thousands of businesses and more than ten million developers worldwide to build unique, personalized experiences for their customers.Twilio Engage is a new omnichannel marketing and growth platform that lets businesses of all sizes use the same kinds of tools, data integrations, analytics, and channels to build personalized campaigns that the digital giants use. Twilio Engage is the latest in the suite of data management and analytics solutions offered by Twilio Segment.

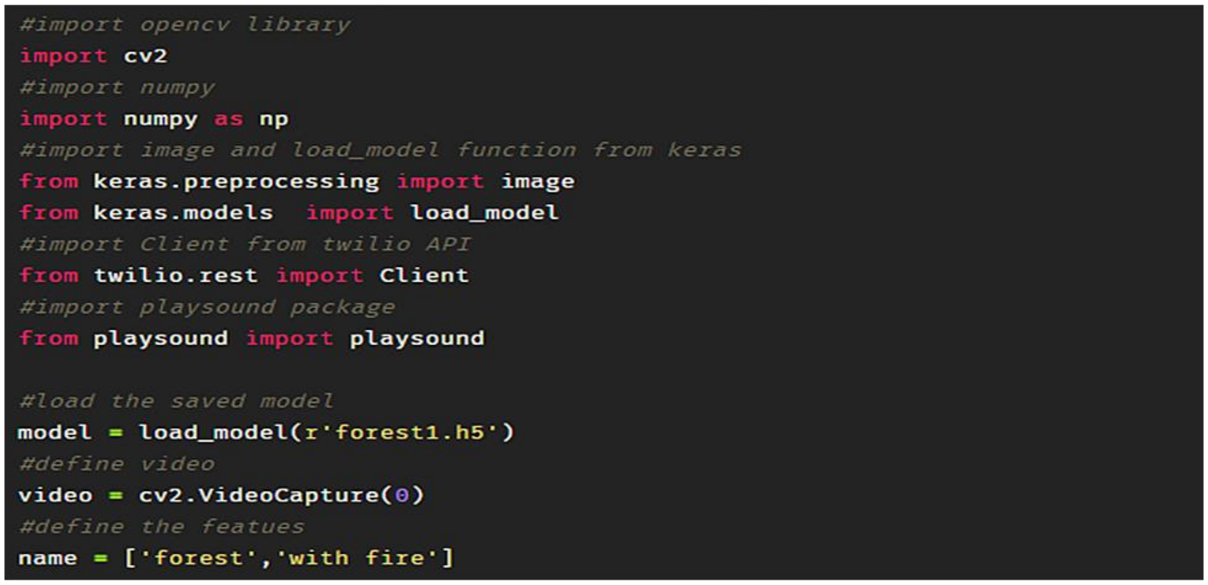
SENDING ALERT MESSAGE:

To play an alerting sound we need to install **"playsound"** library.

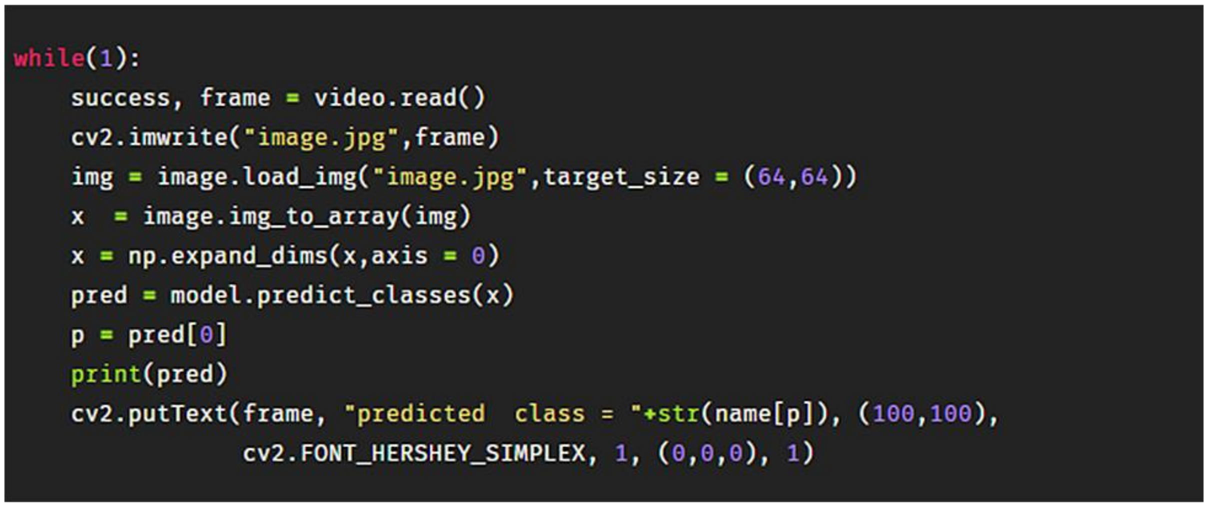
To install this library, open anaconda prompt and execute the below command.

Type **“pip install playsound”** click enter.

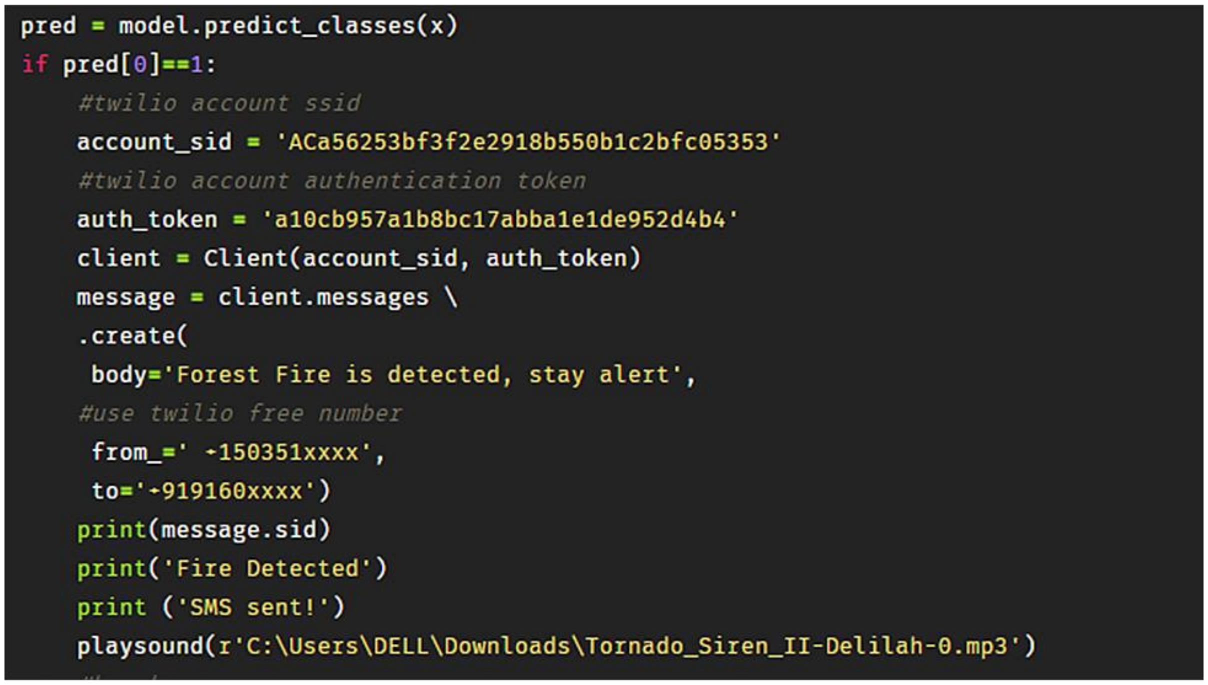
Combining all codes



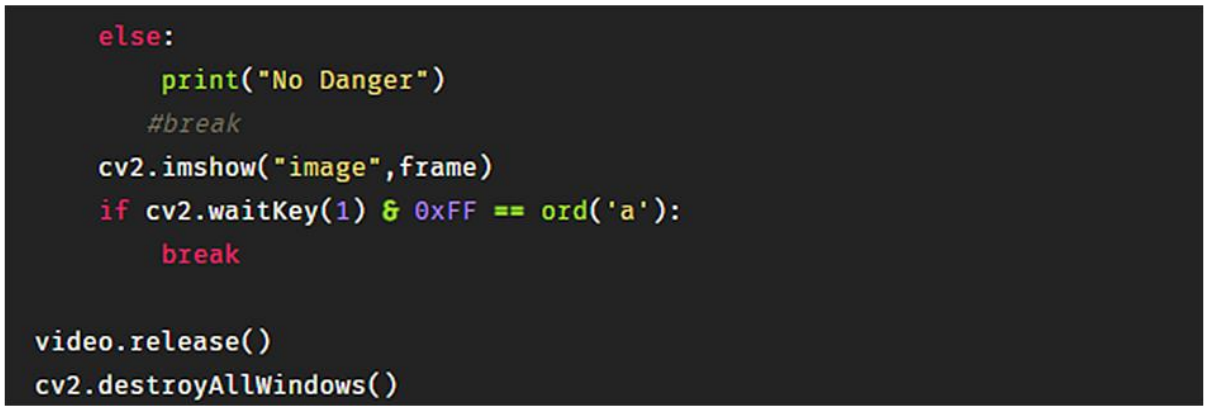
```
#import opencv library
import cv2
#import numpy
import numpy as np
#import image and load_model function from keras
from keras.preprocessing import image
from keras.models  import load_model
#import Client from twilio API
from twilio.rest import Client
#import playsound package
from playsound import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

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



```
while(1):
    success, frame = video.read()
    cv2.imwrite("image.jpg",frame)
    img = image.load_img("image.jpg",target_size = (64,64))
    x  = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)
    pred = model.predict_classes(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_classes(x)
if pred[0]==1:
    #twilio account ssid
    account_sid = 'ACa56253bf3f2e2918b550b1c2bfc05353'
    #twilio account authentication token
    auth_token = 'a10cb957a1b8bc17abba1e1de952d4b4'
    client = Client(account_sid, auth_token)
    message = client.messages \
    .create(
     body='Forest Fire is detected, stay alert',
    #use twilio free number
     from_=' +150351xxxx',
     to='+919160xxxx')
    print(message.sid)
    print('Fire Detected')
    print ('SMS sent!')
    playsound(r'C:\Users\DELL\Downloads\Tornado_Siren_II-Delilah-0.mp3')
```

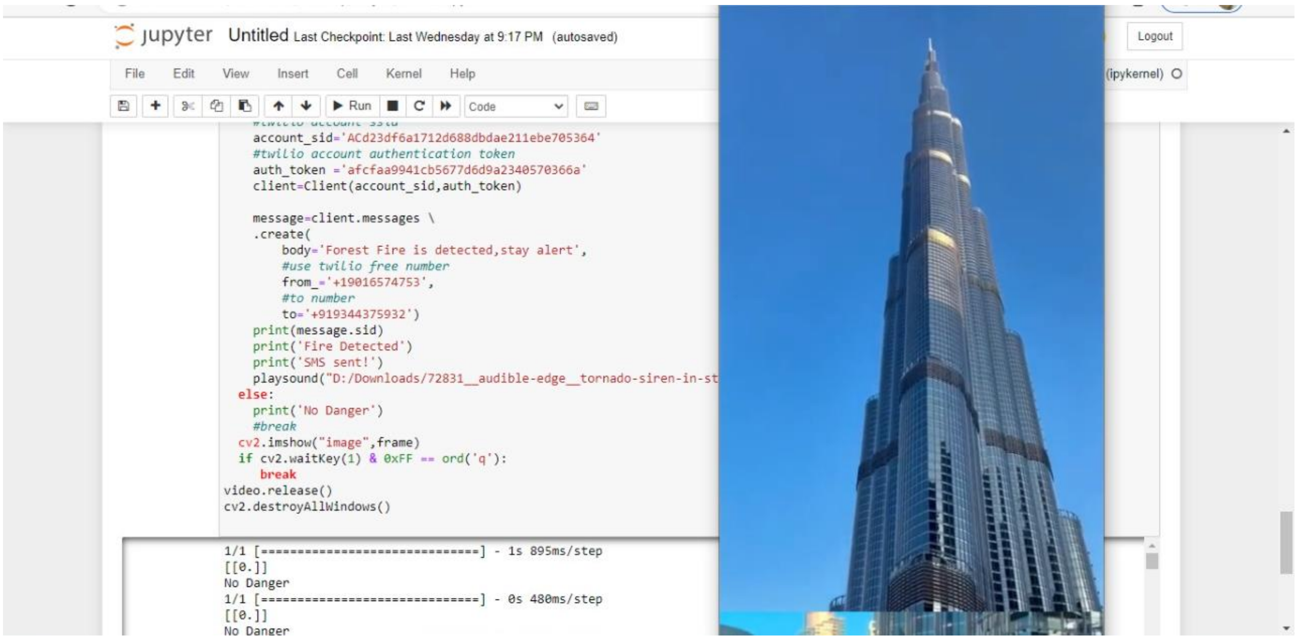


```
    else:
        print("No Danger")
       #break
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('a'):
        break

video.release()
cv2.destroyAllWindows()
```

8.TESTING

8.1 TEST CASE

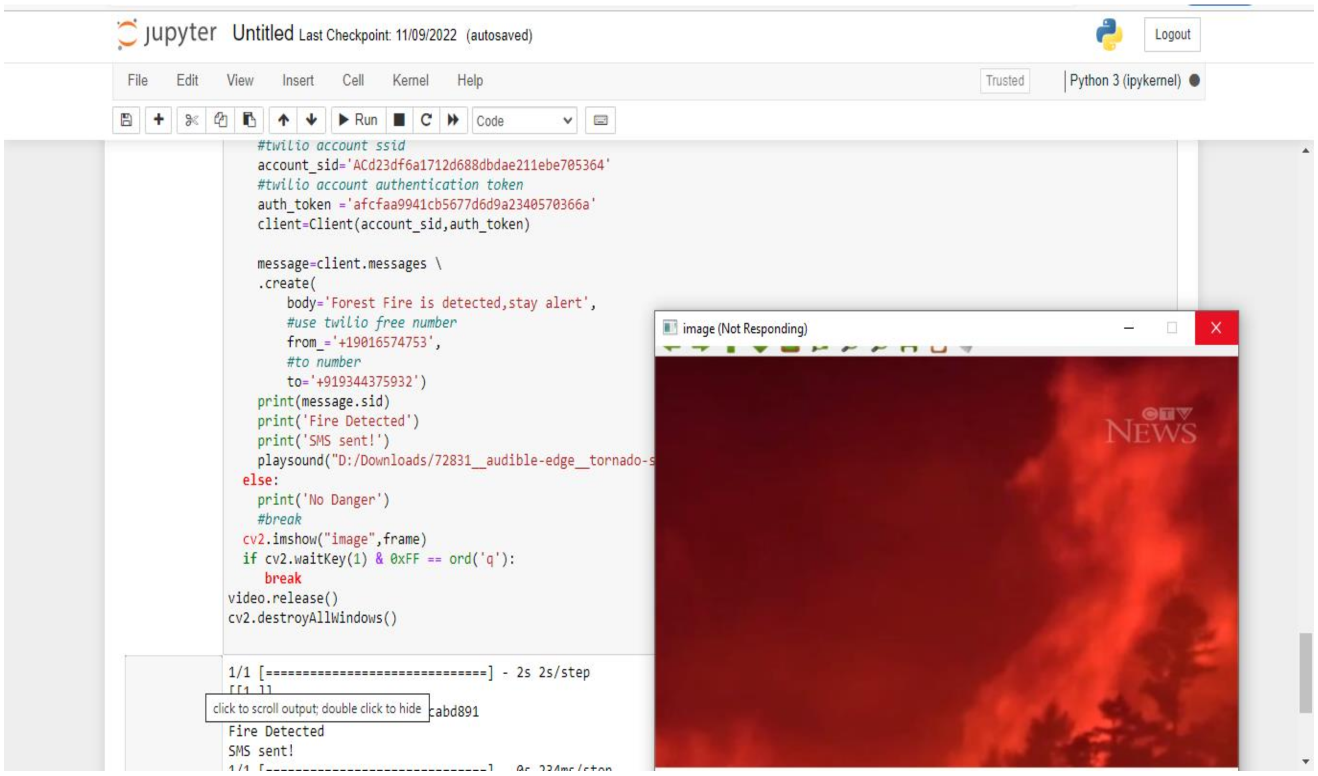
WITHOUT FIRE :


jupyter
Untitled Last Checkpoint: Last Wednesday at 9:17 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
(ipykernel)
Run
Code

```
      account_sid='ACd23df6a1712d688dbdae211ebe705364'
      #twilio account authentication token
      auth_token ='afcfaa9941cb5677d6d9a2340570366a'
      client=Client(account_sid,auth_token)

      message=client.messages \
      .create(
          body='Forest Fire is detected,stay alert',
          #use twilio free number
          from_='+19016574753',
          #to number
          to='+919344375932')
      print(message.sid)
      print('Fire Detected')
      print('SMS sent!')
      playsound("D:/Downloads/72831__audible-edge__tornado-siren-in-st
    else:
      print('No Danger')
      #break
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
       break
  video.release()
  cv2.destroyAllWindows()
```

```
1/1 [==============================] - 1s 895ms/step
[[0.]]
No Danger
1/1 [==============================] - 0s 480ms/step
[[0.]]
No Danger
```

WITH FIRE:


jupyter
Untitled Last Checkpoint: 11/09/2022 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code
image (Not Responding)
CTV NEWS
click to scroll output; double click to hide

```
      #twilio account ssid
      account_sid='ACd23df6a1712d688dbdae211ebe705364'
      #twilio account authentication token
      auth_token ='afcfaa9941cb5677d6d9a2340570366a'
      client=Client(account_sid,auth_token)

      message=client.messages \
      .create(
          body='Forest Fire is detected,stay alert',
          #use twilio free number
          from_='+19016574753',
          #to number
          to='+919344375932')
      print(message.sid)
      print('Fire Detected')
      print('SMS sent!')
      playsound("D:/Downloads/72831__audible-edge__tornado-s
    else:
      print('No Danger')
      #break
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
       break
  video.release()
  cv2.destroyAllWindows()
```

```
1/1 [==============================] - 2s 2s/step
[[1.]]
                                       cabd891
Fire Detected
SMS sent!
1/1 [==============================] - 0s 234ms/step
```

9.RESULTS

Along with knowing the presence of fire in the frame, information about the exact location of the fire is also very significant. The specific fire location information will speed up the process of extinguishing the fire. Therefore, in addition to the frame-based evaluation, it is also crucial to conduct an intersection over union (IoU)-based evaluation. However, to the best of our knowledge, most of the state of the art methods do not conduct an evaluation based on IoU. The IoU-based evaluation is more challenging than frame-based evaluation because of the large number of annotations required for each video. However, this study proposes a new annotation based on the region to be evaluated on an IoU rate. This evaluation compares the location of the detected fire with the actual location of the fire.

The IoU can be calculated as a ratio between the intersection area of the detected fire and the ground truth and the union area of the detected fire and the ground truth.
The proposed method was also evaluated based on the computational time. The faster the computation time used, the more potential the technique is implemented for real-time surveillance. To the best of our knowledge, there was no other research evaluating fire detection based on the IoU rate. Therefore, due to limited resources in implementation, we only compared our proposed method with the R-CNN deep learning method. Table 7 shows the comparison of the proposed method and R-CNN based on the IoU and computation time.

10.ADVANTAGES AND DISADVANNTAGES

10.1 ADVANTAGES

Continuous monitoring of forest fires: all year round, on the whole or part of the territory, day and night.Time saving: real-time visualization of the disaster, immediate transmission of alarms, precise localization of the source of the fire thanks to a triangulation system. Thanks to the cameras and especially the doubt-removal camera, verification and confirmation is much faster than an emergency call.Elimination of human risks: no more isolated men on watch towers.Long-term monitoring: data storage enables continuous improvement of forest fire monitoring and detection.

10.2DISADVANTAGES

However, applying machine learning techniques to fire detection systems has many limitations, such as the limited amount of energy, the energy required for data processing, the short range of communication and limited computations, the complexity of ML algorithms when executing on sensor nodes, and the difficulty of being distributed on every sensor node

11.CONCLUSION

In this model, the process of the forest fire image recognition algorithm based on CNN is presented. Its main feature is that the flame image is employed for training and testing. Then, AlexNet model is introduced, and an adaptive pooling method combined with color features is proposed for the problem that the traditional pooling method in CNN may weaken the image features in some cases. The effects of learning rate, batch size, and other parameters on the performance of CNN are analyzed based on experiments, and the optimal parameters are determined. Candidate flame area is extracted based on color feature; thus, the image feature of non-flame area in the hidden layer is reduced, and the feature, such as shape and texture, is enhanced. The information loss of image are avoided as adaptive pooling is adopted, and the rate of flame recognition in which fire area is segmentation than that of original image is adopted without segmentation. It is shown that the proposed algorithm has high recognition rate and is feasible. In this paper, the pooling of CNN is modified and applied on forest image recognition, recognition rate and consuming time will be developed deeply and compared with other algorithms in future.

12.FUTURESCOPE

Evolution emerges in the processing, computation, and algorithms. This strives many researchers to pay attention in many domains where they work in the processing of surveillance video streams so that abnormal or unusual actions could be detected. The usage of UAVs is recommended in the detection of forest fire due to the high mobility and ensures the coverage areas at various altitudes and locations at a low cost. Hence, an efficient and scalable UAV is used for detection. This work aims in developing the 3D model for the captured scene. YOLOv4 tiny network is deployed to detect the fire. The accuracy of the detection rate achieved through this model is 91%. The proposed model outperforms the other existing techniques in terms of detecting in the early stage. However, this model is sensitive to the forest with dense fogs and clouds. This is because smoke appears as the same as fog, and the model may misclassify the fog as smoke. As our future works, focus to meet practical detection and meet the necessity of early detection including the generation of the mixed reality model of the forest fire area that gives more information, and prevention analysis will be made easy. The 3D modeling techniques presented in this paper can also be extended to various natural disaster prediction models.

13.APPENDIX

SOURCE CODE

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

#Define the parameters/arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zo
om_range=0.2,horizontal_flip=True)

test_datagen=ImageDataGenerator(rescale=1./255)

#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory("d:/Downloads/archive
(1)/Dataset/Dataset/train_set",target_size=(128,128),
                                        batch_size=32,class_mode='binary')

#Applying ImageDataGenerator functionality to testset
x_test=test_datagen.flow_from_directory("d:/Downloads/archive
(1)/Dataset/Dataset/test_set",target_size=(128,128),
                                      batch_size=32,class_mode='binary')

#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
#importMaxpooling 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'))
#addmaxpooling 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"])

from tensorflow.keras.models import Model

#Training the model
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validati
on_steps=4)

model.save("forest1.h5")

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

#load the saved model
model = load_model("forest1.h5")

img=image.load_img("d:/Downloads/archive (1)/Dataset/Dataset/test_set/with
fire/180802_CarrFire_010_large_700x467.jpg")
x=image.img_to_array(img)
res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
#expand the image shape
x=np.expand_dims(res,axis=0)

pred=model.predict(x)

pred

pip install twilio
pip install playsound

#importopencv library
import cv2
#importnumpy
import numpy as np
#import image function from keras
from tensorflow.keras.preprocessing import image
#importload_model from keras
from keras.models import load_model
#import Client from twilio API
from twilio.rest import Client
#importplaysound package
from playsound import playsound

#load the saved model
model=load_model('forest1.h5')
#define video
video=cv2.VideoCapture("d:/Downloads/FORESTFIRE.mp4")
#define the features
name=['forest','with fire']
```

```
#load the saved model
model=load_model('forest1.h5')
#define video
video=cv2.VideoCapture("d:/Downloads/FORESTFIRE.mp4")
#define the features
```

```
name=['forest','with fire']
while(1):
  success,frame=video.read()
  cv2.imwrite("image.jpg",frame)
  img=image.load_img("image.jpg",target_size=(64,64))
  x=image.img_to_array(img)
  res = cv2.resize(x, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
  x=np.expand_dims(res,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))
  if pred[0]==1:
    #twilio account ssid
    account_sid='ACd23df6a1712d688dbdae211ebe705364'
    #twilio account authentication token
    auth_token ='afcfaa9941cb5677d6d9a2340570366a'
    client=Client(account_sid,auth_token)

    message=client.messages \
    .create(
        body='Forest Fire is detected,stay alert',
        #usetwilio free number
        from_='+19016574753',
        #to number
        to='+919344375932')
    print(message.sid)
    print('Fire Detected')
    print('SMS sent!')
    playsound("D:/Downloads/72831__audible-edge__tornado-siren-in-streamwood-
il.mp3")
  else:
    print('No Danger')
    #break
  cv2.imshow("image",frame)
  if cv2.waitKey(1) & 0xFF == ord('q'):
     break
video.release()
cv2.destroyAllWindows()
```

TRAIN CNN ON IBM CLOUD

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

#Define the parameters/arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,rotation_range=180,zo
om_range=0.2,horizontal_flip=True)

test_datagen=ImageDataGenerator(rescale=1./255)

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='CyfbVlhpb88HCt21fx4u1-RwsYc8kWTEnhyO199kN_R5',
    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 = 'forestfiredetection-donotdelete-pr-u0rtatnl4rnifa'
object_key = 'ff.zip'

streaming_body_1 = 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_1.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)

pwd

import os
filenames=os.listdir('/home/wsuser/work/Dataset/Dataset/train_set')

#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory('/home/wsuser/work/Dataset/Dataset/train_set',
target_size=(128,128),batch_size=32,class_mode='binary')

#Applying ImageDataGenerator functionality to testset
x_test=test_datagen.flow_from_directory('/home/wsuser/work/Dataset/Dataset/test_set',targ
et_size=(128,128),batch_size=32,class_mode='binary')

#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
#importMaxpooling 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'))
#addmaxpooling 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,validati
on_steps=4)

model.save("forest1.h5")

!tar -zcvf forest-fire-model_new.tgz forest1.h5
```

```
ls

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

# Replace the credentials that you got from Watson Machine Learning service
from ibm_watson_machine_learning import APIClient
wml_credentials={
                "url":"https://us-south.ml.cloud.ibm.com",
                "apikey":"Z4BMC3kx1thzvwrAjgVHVNJ9ohlU8eHZyg5cRoJ2YqhY"
                }
client=APIClient(wml_credentials)

client=APIClient(wml_credentials)

def guid_from_space_name(client,space_name):
    space=client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"] ==
space_name)['metadata']['id'])

space_uid = guid_from_space_name(client,'forest_fire_project')
print("Space UID = " + space_uid)

client.set.default_space(space_uid)

client.software_specifications.list()

software_spec_uid=client.software_specifications.get_uid_by_name("tensorflow_rt22.1-
py3.9")
```

```
software_spec_uid

model_details=client.repository.store_model(model='forest-fire-
model_new.tgz',meta_props={
    client.repository.ModelMetaNames.NAME:"CNN",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid,
    client.repository.ModelMetaNames.TYPE:"tensorflow_2.7"}
                                          )
model_id=client.repository.get_model_uid(model_details)

model_id

client.repository.download(model_id,'my_model.tar.gz')
```

REFERENCES

[1] Z. Abrams, A. Goel and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In Proc. of International Symposium on Information Processing in Sensor Networks (IPSN’04), Berkeley, CA, April 2004, pp. 424–432.
[2] Aerovision Web Page. http://www.aerovision-uav.com.
[3] N. Ahmed, S. Kanhere and S. Jha. Probabilistic coverage in wireless sensor networks. In Proc. of IEEE Conference on Local Computer Networks (LCN’05), Sydney, Australia, November 2005, pp. 672–681.
[4] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam and E. Cayirci. A survey on sensor networks. IEEE Communications Magazine 40(8) (2002), 102–114.
[5] M.E. Alexander and W.J. De Groot. Fire behavior in Jack Pine stands as related to the Canadian Forest Fire Weather Index System. Technical report, Canadian Forest Service,

Northern Forestry Centre, Edmonton, Alberta, 1988.

[6] AVHRR Web Page. http://noaasis.noaa.gov/noaasis/ml/avhrr.html.

[7] B.C. Ministry of Forests and Range. Fire Review Summary for Okanagan Mountain Fire (K50628).

[8] B.C. Ministry of Forests and Range Web Page. http://www.for.gov.bc.ca.

[9] E. Breejen, M. Breuers, F. Cremer, R.A.W Kemp, M. Roos, K. Schutte and J.S. Vries. Autonomous forest fire detection. In Proc. of Third International Conference on Forest Fire Research and Fourteenth Conference on Fire and Forest Meteorology, Luso, Portugal, November 1998, pp. 2003–2012.

[10] H. Bronnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. Discrete and Computational Geometry 14(4) (1995), 463–479.

[11] Canadian Forest Fire Danger Rating System (CFFDRS)Web Page. http://www.nofc.forestry. ca/fire.

[12] Canadian Forest Service (CFS) Web Page. http://www.nrcan.gc.ca/cfs.

[13] Q. Cao, T. Yan, T. Abdelzaher and J. Stankovic. Analysis of target detection performance for wireless sensor networks. In Proc. of International Conference on Distributed Computing in Sensor Networks, Marina Del Rey, CA, June 2005, pp. 276–292.

[14] M. Cardei and J. Wu. Coverage in wireless sensor networks. In M. Ilyas and I. Mahgoub, editors, Handbook of Sensor Networks. CRC Press, 2004.

[15] M. Cardei and J.Wu. Energy-efficient coverage problems in wireless ad hoc sensor networks. Elsevier Computer Communications 29(4) (2006), 413–420.

[16] Z. Chaczko and F. Ahmad. Wireless sensor network based system for fire endangered areas. In Proc. of Third International Conference on Information Technology and Applications (ICITA’05), Sydney, Australia, July 2005.

[17] K. Chakrabarty, S. Iyengar, H. Qi and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. IEEE Transactions on Computers 51(12) (2002), 1448–1453.

[18] Crossbow Inc. Web Page. http://www.xbow.com/.

[19] F. Dai and J. Wu. An extended localized algorithm for connected dominating sets formation in ad hoc wireless networks. IEEE Transactions on Parallel and Distributed Systems, 15(10) (2004), 908–920.

[20] W. J. de Groot. Interpreting the Canadian Forest Fire Weather Index (FWI) System. In Proc. of Fourth Central Region Fire Weather Committee Scientific and Technical Seminar, Edmonton, Canada, 1998.

[21] D. M. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. In Proc. of SPIE Symposium on Smart Structures and Materials, San Diego, CA, March 2005, pp. 477–484.

[22] Fire Watch Web Page. http://www.fire-watch.de/.

[23] J. Fleming and R. G. Robertson. Fire Management Tech Tips: The Osborne Fire Finder. Technical Report 0351 1311-SDTDC, USDA Forest Service, October 2003.

[24] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NPCompleteness. W. H. Freeman, 1979.

[25] H. Gupta, Z. Zhou, S. Das and Q. Gu. Connected sensor cover: self-organization of sensor networks for efficient query execution. IEEE/ACM Transactions on Networking 14(1) (2006), 55–67.

IBM REPOSITARY LINK:

IBM-EPBL/IBM-Project-16533-1659616656