A project report on

Fire Recognition in Forests Using Deep Learning

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

MASTER OF TECHNOLOGY
IN
INTEGRATED SOFTWARE ENGINEERING

by

G. Sai Sheshank (17MIS7099)

Under the Guidance of

DR. Prabha Selvaraj

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

SEPTEMBER 2021

CERTIFICATE

This is to certify that the Capstone Project work titled "**FIRE RECOGNITION IN FORESTS USING DEEP LEARINING**" that is being submitted by **G. SAI SHESHANK (17MIS7099)** is in partial fulfillment of the requirements for the award of **Master of Technology (Integrated 5 Year) software engineering**, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Prabha Selvaraj

Guide

The thesis is satisfactory

Approved by

PROGRAM CHAIR

M. Tech. SE

DEAN

School Of Computer Science and Engineering

ACKNOWLEDGEMENTS

It is my pleasure to express with deep sense of gratitude to Prof.Prabha Selvaraj, VIT-AP, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my Endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of coding.

I would like to express my gratitude Dr. G. Viswanathan, MR. G. V. Selvam, MS. Kadhambari S. Viswanathan, Dr. D. Subhakar, and Dr. Hari Seetha, Dean of School of Computer Science & Engineering, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Reeja S R, Associate Professor, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: AMARAVATHI

Date: 29.08.2021 G. Sai Sheshank

ABSTRACT

This project is aimed at the areas where the fire combustion takes place to save the wildlife animals and people situated around. A huge forest fire has a significant impact on the ecology, deciding the future of the environment for decades. Forest fire prevention, as well as sufficient preparedness for efficient firefighting, is one of today's most critical jobs. To accomplish so, you'll need a thorough understanding of artificial intelligence, deep learning techniques (CNN) to visualize/recognize through the images we provide the model to forecast, we can determine if the woodlands are burning or not.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLESs

LIST OF ACRONYMS

OpenCV: Open Source Computer Vision Library

AI: Artificial Intelligence

ANN: Artificial Neural Networks

CNN: Convolution Neural Networks

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Forest fires are a huge environmental problem that cause economic and ecological harm as well as putting human lives in danger. Every year, Across the United States, about 1,00,000 wildfires are raging, 9 million acres of land have been burnt due to severe wildfires. In a sparsely inhabited forest area, it is difficult to predict and detect forest fires, and it is even more difficult to do so using conventional means such as normal cameras or a video-based approach. Satellites can be a valuable source of data both before and after the fire because of their dependability and effectiveness. Forest fires have destroyed 3.5 to 4.5 million km2 of land. We'll train the model so that it can tell if a forest has fire or not if you give it an image or a photograph of one. It will issue an alert if it recognizes the fire image. This enables us to quickly identify forest fires while being at home.

1.1OVERVIEW

Our country's greatest asset is its environment. Forests provide us with a variety of tangible items as well as environmental services. They are beneficial to both industry and rural economic development. Simultaneously, when a forest is on fire, it generates a lot of carbon dioxide, which contributes to climate change and global warming. As a result, the forest fire must be identified sooner rather than later. Forest fires, for example, have lately erupted in Australia. A total of 18,636,079 hectares (46,050,750 acres) of forest land had been burned. So, because early detection of forest fire combustion is difficult due to remoteness, inaccessibility, harsh weather, and a shortage of border employees, we designed a model to identify Fired Forest Combustion. Anyway, doing it manually would need a lot of human power, that's why we're using the CNN (Convolutional Neural Network) Model, a DL Techniques, to anticipate forest fires.

1.2CHALLENGES

- This model should be able to distinguish and reliably identify forest fires. If we give photos, it should be able to tell whether they are with or without fire. The method is now being refined in order to provide real-time automated fire detection. For example, using mobile devices, drones, Arduino sensors, and so on…
- Should be supported in any kind of weather situations.
- Manually searching the forest for the source of the fire is quite tough, thus we must implement in the form of software.

1.4 PROJECT STATEMENT

We are able to satisfy the client's major objectives through this project. The existing system does not detect fires unless we place/have sensors to recognize it on its own by them, therefore humans must waste their time going around the forest looking for them. However, a new technology that we are developing will be able to anticipate fires using CNN images (analyzing visual imagery). This can be quite time consuming and tedious, causing consumers to avoid using these applications for lengthy periods of time. As a result, we are able to meet the goal of our project by employing deep learning techniques such as CNN.

1.5 OBJECTIVES

Our project's major goal is to reduce forest fires and increase the protection to forest while also creating environment which is user-friendly so that everyone will be able to use easily. They should also be able to use the software from any location and at any time.

1.6 SCOPE OF THE PROJECT

Forest Combustion Recognition's goal is to protect the environment (plants and trees), wildlife, and tribal people who live near the forest. It can function regardless of the weather and can be utilized wherever and whenever we wish. This project is completed with the use of artificial intelligence (AI) and deep learning techniques.

1.7ORGANIZATION OF THE REPORT

The remaining chapters of the project report are described as follows:

- Chapter 2 contains information about the methodology, proposed system, software, and hardware details.
- Chapter 3 discusses the outcomes of the project after it has been implemented.
- Chapter 4 concludes the this report.
- Chapter 5 has codes of all.
- Chapter 6 show the references.

CHAPTER 2

2.1 INTRODUCTION

This project's major goal is to eliminate physical labour. Because the approach utilised for our project is CNN, someone with a good understanding of python and deep learning techniques, particularly convolution neural networks, is required to construct this type of project. Using this technique, we were able to achieve/predict two scenarios: "with fire" and "without fire."

2.2 LITERATURE SURVEY

2.2.1 EXISTING PROBLEM

There are many extant systems for forest combustion recognition these days, each of which differs from the others in that some of them fail to detect whether a fire is there or not. Some systems are built using IoT applications, and we must keep that hardware in the forest. Few systems may be destroyed due to weather circumstances, such as when it rains. As a result, detection will be tough. Then our model is born, and it works no matter what the weather is like.

2.2.2 PROPOSED ANSWER

The suggested Fire Prediction for Forests model is based on Convolution Neural Networks that have been integrated into a warning system. The proposed system's development approach is divided into two modules:

- Fire Recognition: Recognition of the areas which were affected.
- Management of Fire: Forest fire remedial techniques are all about determining where the fire started in the forest.

We attempted to identify photos of forest with fire and forest without fire in our suggested approach so that convolution neural networks might be used to further classify the images (CNNs). Because CNNs can handle enormous amounts of data and estimate features automatically, they've been used to recognize forest fires. The working GUI for this technique has been chosen from the dataset of forest of train and test. We divide a sample of a few photographs (around 100) into train and test groups, and the model successfully predicts them.

2.3 DATASET

Images of various resolutions and sizes were gathered from a variety of sources, including photos taken with a cell phone and a hand-held camera, as well as images found on the internet. These photos were rescaled to 255 255 in order to enhance the size of our dataset, resulting in a total of 200 images. The information was divided into two groups:

1) Forests Caught Fire and 2) Forests With No Fire.

A total of 140 photos were used as training data: 70 photographs of a forest with fire and 70 images of a forest without fire (70 images). A total of 30% of the testing data (60 photos out of 200) was chosen for the validation set.

Fig 1 : Forests Caught fire

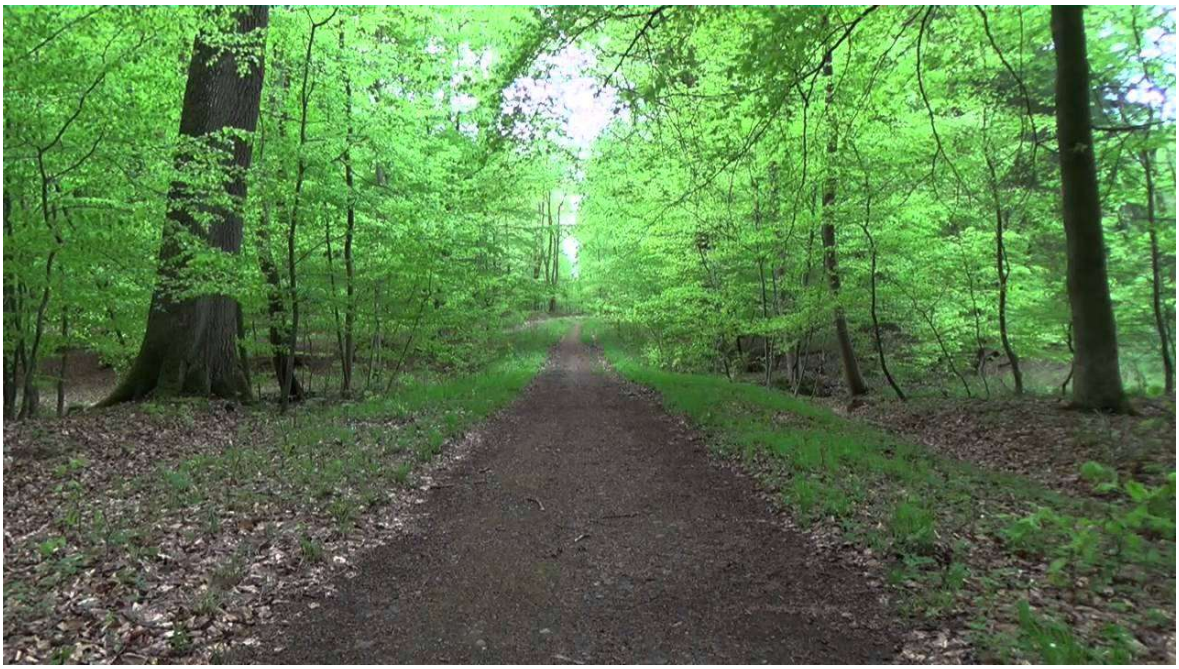

Fig 2 : Forests With No fire

2.4 WORKING METHODOLOGY

Convolutional Neural Networks are a Deep Learning approach that we applied here (CNN). We chose this method because it effectively completes the task and also serves as a key component of a conventional Artificial Neural Network (ANN) with pre-processing. The CNN (Convolutional Neural Network) is a combination of Neural Networks and Convolutional Layers. Dense Layer, Convolutional Layer, Input layer, and Pooling Layer are the layers that make up any Neural Network used for image processing.

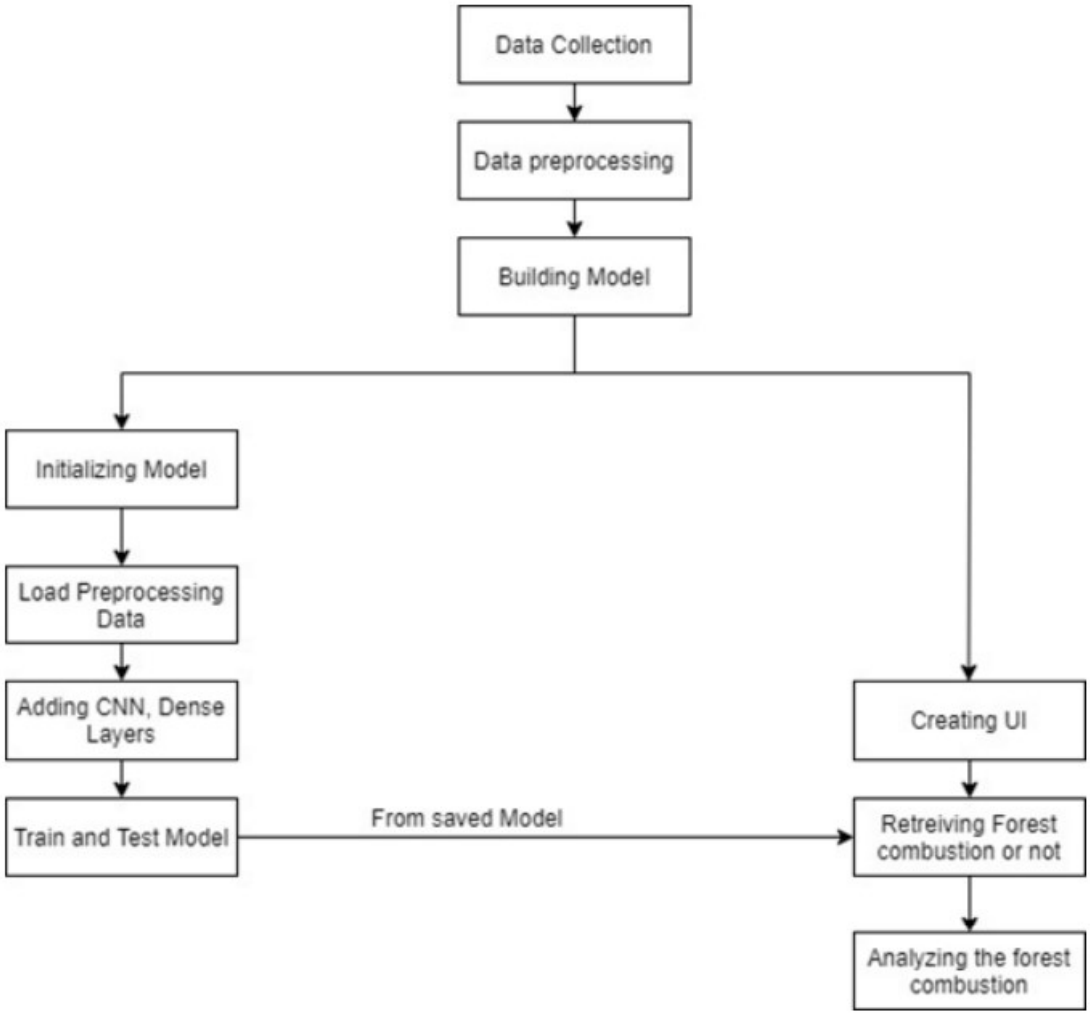

Fig 3: Flowchart

1.4.1 IMPORTING THE BUILDING LIBRARIES

Importing the libraries, we'll need for our model. Where we'll continue to add layers like Flatten, Max Pooling, Convolution, and Dense. We'll use Keras packages to defines our DLNN (Deep Learning Neural Networks). To define the network architecture, we import the Sequential, Dense, Dropout, and Activation packages. For saving and retrieving our model, we utilize the load model package.

```
In [1]: #importing the libraries
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Flatten
        from keras.models import load_model
        import numpy as np
        import cv2
        from skimage.transform import resize
```

```
Using TensorFlow backend.
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) /
'(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) /
'(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning: Passing (type,
1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) /
'(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
```

Fig 4: Imported Libraries

1.4.2 INITIALIZE THE MODEL

Keras models can be built in two ways: sequential and functional. For our model, we're going to use sequential. For most issues, the sequential API allows you to build models layer by layer. It has limitations in that it does not allow you to design models with several inputs and outputs or layers.

```
In [2]: model =Sequential()
```

```
WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:74: The name tf.get_d
efault_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

Fig5: Model Initialization

1.4.3 ADDING CNN AND DENSE LAYERS

```
In [3]: model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'))

        WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:517: The name tf.plac
        eholder is deprecated. Please use tf.compat.v1.placeholder instead.

        WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4138: The name tf.ran
        dom_uniform is deprecated. Please use tf.random.uniform instead.

In [4]: model.add(MaxPooling2D(pool_size=(2,2)))

        WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3976: The name tf.nn.
        max_pool is deprecated. Please use tf.nn.max_pool2d instead.

In [5]: model.add(Flatten())
```

Fig6: Convolution, Maxpooling, Flatten Layers

Convolution: Convolution is a technique for identifying certain features in an image. Convolution can assist the machine learn certain qualities of a picture through sharpening, blurring, noise reduction, edge detection, and more.

Pooling: Because a complex image can be overly huge, it must be shrunk. Pooling is used to reduce the size of an image without losing details or patterns.

Flattening: A 2-D matrix of features is flattened into a vector of features that can be input into a neural network or classifier.

The regular deeply linked neural network layer is the dense layer. It is the most popular and often utilized layer. The following operation is performed on the input by the dense layer, and the output is returned. We'll use this to add hidden and output layers.

The arguments we used in dense layer are as follows

- input_shape represent the shape of input data.
- The keyword arguments used for passing initializations to layers will depend on the layer it is init.
- units represent the no of nodes in the layer.
- activation represents the activation function.
- Output_dim represents how many outputs needed.

```
In [6]: model.add(Dense(output_dim=128,init='uniform',activation='relu'))

C:\Users\mahit\anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `Dense` call to the Keras 2 AP
I: `Dense(activation="relu", units=128, kernel_initializer="uniform")`
  """Entry point for launching an IPython kernel.

In [7]: model.add(Dense(output_dim=2,activation='softmax',init='uniform'))

C:\Users\mahit\anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `Dense` call to the Keras 2 AP
I: `Dense(activation="softmax", units=2, kernel_initializer="uniform")`
  """Entry point for launching an IPython kernel.
```

Fig7: Applying dense layers

Softmax Activation: When building a multi-class classifier in neural networks, the **softmax activation function** is employed to solve the challenge of allocating an instance to one of more than two possible classes.

Rectified Linear Unit (ReLU) Activation: ReLU stands for Rectified Linear Unit. The key advantage of this function over other activation functions is that it does not simultaneously stimulate all of the neurons. If we are unsure about which activation function to utilise, ReLU is a decent option.

```
In [11]: model.compile(loss = 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\optimizers.py:790: The name tf.train.Optimizer is d
eprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3295: The name tf.log
is deprecated. Please use tf.math.log instead.
```

Fig8: Compile

Optimizer: Adam is used to provide an optimization algorithm that can handle on noisy problems. This also increases the accuracy.

Loss: We have two different types of cross entropy one is binary and the other is categorical. Binary **cross-entropy** is for binary classification, whereas **categorical cross entropy** is for multi-class classification.

Metrics: Metrics is a function just to know the performance of the model.

1.4.4 IMAGE PREPROCESSING

```
In [8]: from keras.preprocessing.image import ImageDataGenerator
        train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        test_datagen=ImageDataGenerator(rescale=1./255)
```

Fig9: Image Preprocessing

Importing the necessary Libraries and then resizing the images to a specific size

Arguments:

- rescale: Factor for rescaling. None is the default value. If None or 0 is specified, no rescaling is performed; otherwise, the data is multiplied by the value specified.

- shear intensity: Shear range Intensity. Angle of Shear measured in degrees in a counter-clockwise direction.
- zoom range: The zooming range of image.

1.4.5 IMPORT DATASET

Importing the dataset from our target folder, which comprises two classes of images: forest with fire and forest without fire, stored in two separate folders that have been consolidated into one.

```
In [8]:  from keras.preprocessing.image import ImageDataGenerator
         train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
         test_datagen=ImageDataGenerator(rescale=1./255)

In [9]:  x_train = train_datagen.flow_from_directory(r'C:\Users\mahit\OneDrive\Desktop\AI intern proj1\forest combustion\trainset',tar
         x_test = test_datagen.flow_from_directory(r'C:\Users\mahit\OneDrive\Desktop\AI intern proj1\forest combustion\testset',target

         Found 140 images belonging to 2 classes.
         Found 60 images belonging to 2 classes.

In [10]: print(x_train.class_indices)

         {'forestwithfire': 0, 'forestwithoutfire': 1}
```

Fig10: Imported Dataset

Arguments:

- directory: place of the x_test or x_train
- target_size: This is the size that all photos identified will be scaled to. The output size is same to the input size.
- Batch_size: The size of the data batches (default: 32).
- Class_mode: The type of returned label arrays is determined by this parameter. One of “categorical”, “binary”, “sparse”, “input”, or None.

1.4.6 TESTING AND TRAINING THE MODEL

Here we are training the model by fit_generator and later going to save the model. The arguments are as follows

- generator: The neural network is trained using a generator sequence. (x_train).
- steps_per_epochs: The total number of steps (batches of samples) to yield from the generator before declaring one epoch ended and initiating the next. It's commonly calculated by dividing your dataset's number of samples by the batch size.
- epochs: The number of epochs is the total number of epochs. An epoch is a complete cycle of predictions made by a neural network.

- Validation_data: A generator sequence for testing and evaluating the neural network's predictions (x test).
- Validation_steps: The total number of steps (sample batches) that the validation data generator will produce before stopping at the conclusion of each epoch.

```
In [12]: model.fit_generator(x_train,steps_per_epoch=5,epochs=100,validation_data=x_test,validation_steps=2)

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_
support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:986: The name tf.a
ssign_add is deprecated. Please use tf.compat.v1.assign_add instead.

Epoch 1/100
5/5 [------------------------------] - 3s 692ms/step - loss: 0.6939 - acc: 0.5745 - val_loss: 0.4192 - val_acc: 0.9333
Epoch 2/100
5/5 [==============================] - 2s 383ms/step - loss: 0.4425 - acc: 0.8076 - val_loss: 0.3078 - val_acc: 0.9000
Epoch 3/100
5/5 [==============================] - 2s 418ms/step - loss: 0.3016 - acc: 0.9424 - val_loss: 0.2572 - val_acc: 0.8833
Epoch 4/100
5/5 [==============================] - 2s 368ms/step - loss: 0.2329 - acc: 0.9017 - val_loss: 0.2828 - val_acc: 0.8500
Epoch 5/100
5/5 [==============================] - 2s 375ms/step - loss: 0.1769 - acc: 0.9337 - val_loss: 0.1520 - val_acc: 0.9667
Epoch 6/100
5/5 [==============================] - 2s 371ms/step - loss: 0.1660 - acc: 0.9401 - val_loss: 0.1307 - val_acc: 0.9333
Epoch 7/100
5/5 [==============================] - 2s 379ms/step - loss: 0.1411 - acc: 0.9552 - val_loss: 0.1282 - val_acc: 0.9500
Epoch 8/100
5/5 [==============================] - 2s 378ms/step - loss: 0.1355 - acc: 0.9337 - val_loss: 0.2268 - val_acc: 0.9167
Epoch 9/100
5/5 [==============================] - 3s 518ms/step - loss: 0.1267 - acc: 0.9209 - val_loss: 0.1328 - val_acc: 0.9500
Epoch 10/100
5/5 [==============================] - 2s 366ms/step - loss: 0.1152 - acc: 0.9529 - val_loss: 0.1303 - val_acc: 0.9333
Epoch 11/100
5/5 [==============================] - 2s 362ms/step - loss: 0.1093 - acc: 0.9401 - val_loss: 0.1279 - val_acc: 0.9333
Epoch 12/100
5/5 [==============================] - 2s 395ms/step - loss: 0.0777 - acc: 0.9808 - val_loss: 0.1557 - val_acc: 0.9333
Epoch 13/100
5/5 [==============================] - 2s 383ms/step - loss: 0.0672 - acc: 0.9744 - val_loss: 0.2113 - val_acc: 0.9000
Epoch 14/100
5/5 [==============================] - 2s 363ms/step - loss: 0.0853 - acc: 0.9465 - val_loss: 0.1479 - val_acc: 0.9500
Epoch 15/100
5/5 [==============================] - 2s 370ms/step - loss: 0.0725 - acc: 0.9680 - val_loss: 0.1751 - val_acc: 0.9333
Epoch 16/100
Epoch 17/100
5/5 [==============================] - 3s 545ms/step - loss: 0.0850 - acc: 0.9442 - val_loss: 0.1549 - val_acc: 0.9333
Epoch 18/100
5/5 [==============================] - 2s 485ms/step - loss: 0.0435 - acc: 0.9872 - val_loss: 0.1829 - val_acc: 0.9333
Epoch 19/100
5/5 [==============================] - 2s 437ms/step - loss: 0.0669 - acc: 0.9680 - val_loss: 0.1622 - val_acc: 0.9333
Epoch 20/100
5/5 [==============================] - 2s 446ms/step - loss: 0.0417 - acc: 0.9872 - val_loss: 0.1832 - val_acc: 0.9167
Epoch 21/100
5/5 [==============================] - 2s 461ms/step - loss: 0.0464 - acc: 1.0000 - val_loss: 0.2405 - val_acc: 0.9167
Epoch 22/100
5/5 [==============================] - 2s 475ms/step - loss: 0.0477 - acc: 0.9872 - val_loss: 0.2401 - val_acc: 0.9167
Epoch 23/100
5/5 [==============================] - 3s 514ms/step - loss: 0.0648 - acc: 0.9657 - val_loss: 0.1507 - val_acc: 0.9167
Epoch 24/100
5/5 [==============================] - 3s 578ms/step - loss: 0.0740 - acc: 0.9616 - val_loss: 0.3794 - val_acc: 0.9000
Epoch 25/100
5/5 [==============================] - 3s 543ms/step - loss: 0.0483 - acc: 0.9808 - val_loss: 0.3627 - val_acc: 0.8833
```

```
Epoch 26/100
5/5 [==============================] - 2s 453ms/step - loss: 0.1871 - acc: 0.9187 - val_loss: 0.2418 - val_acc: 0.9333
Epoch 27/100
5/5 [==============================] - 3s 592ms/step - loss: 0.1846 - acc: 0.9104 - val_loss: 0.9841 - val_acc: 0.7667
Epoch 28/100
5/5 [==============================] - 3s 533ms/step - loss: 0.2246 - acc: 0.9017 - val_loss: 0.5859 - val_acc: 0.7833
Epoch 29/100
5/5 [==============================] - 3s 620ms/step - loss: 0.2026 - acc: 0.9017 - val_loss: 0.3482 - val_acc: 0.9333
Epoch 30/100
5/5 [==============================] - 3s 579ms/step - loss: 0.0760 - acc: 0.9680 - val_loss: 0.2718 - val_acc: 0.9167
Epoch 31/100
5/5 [==============================] - 3s 524ms/step - loss: 0.0892 - acc: 0.9552 - val_loss: 0.1858 - val_acc: 0.9333
Epoch 32/100
5/5 [==============================] - 3s 533ms/step - loss: 0.1455 - acc: 0.9465 - val_loss: 0.1519 - val_acc: 0.9500
Epoch 33/100
5/5 [==============================] - 3s 571ms/step - loss: 0.1152 - acc: 0.9424 - val_loss: 0.2013 - val_acc: 0.9167
Epoch 34/100
5/5 [==============================] - 3s 573ms/step - loss: 0.1046 - acc: 0.9657 - val_loss: 0.2678 - val_acc: 0.9167
Epoch 35/100
```

```
Epoch 36/100
5/5 [==============================] - 3s 581ms/step - loss: 0.0568 - acc: 0.9808 - val_loss: 0.2137 - val_acc: 0.9333
Epoch 37/100
5/5 [==============================] - 3s 596ms/step - loss: 0.0591 - acc: 0.9744 - val_loss: 0.1337 - val_acc: 0.9500
Epoch 38/100
5/5 [==============================] - 3s 609ms/step - loss: 0.0216 - acc: 0.9936 - val_loss: 0.1666 - val_acc: 0.9333
Epoch 39/100
5/5 [==============================] - 5s 935ms/step - loss: 0.0277 - acc: 0.9936 - val_loss: 0.1373 - val_acc: 0.9500
Epoch 40/100
5/5 [==============================] - 4s 836ms/step - loss: 0.0151 - acc: 1.0000 - val_loss: 0.1557 - val_acc: 0.9333
Epoch 41/100
5/5 [==============================] - 3s 692ms/step - loss: 0.0159 - acc: 1.0000 - val_loss: 0.1290 - val_acc: 0.9333
Epoch 42/100
5/5 [==============================] - 4s 771ms/step - loss: 0.0217 - acc: 0.9936 - val_loss: 0.1287 - val_acc: 0.9333
Epoch 43/100
5/5 [==============================] - 3s 695ms/step - loss: 0.0138 - acc: 1.0000 - val_loss: 0.1299 - val_acc: 0.9500
Epoch 44/100
5/5 [==============================] - 4s 789ms/step - loss: 0.0137 - acc: 1.0000 - val_loss: 0.1302 - val_acc: 0.9500
Epoch 45/100
```

```
Epoch 56/100
5/5 [==============================] - 5s 960ms/step - loss: 0.0198 - acc: 0.9849 - val_loss: 0.1610 - val_acc: 0.9333
Epoch 57/100
5/5 [==============================] - 5s 982ms/step - loss: 0.0383 - acc: 0.9872 - val_loss: 0.1304 - val_acc: 0.9500
Epoch 58/100
5/5 [==============================] - 5s 936ms/step - loss: 0.0294 - acc: 0.9872 - val_loss: 0.1265 - val_acc: 0.9500
Epoch 59/100
5/5 [==============================] - 4s 785ms/step - loss: 0.0162 - acc: 0.9936 - val_loss: 0.1779 - val_acc: 0.9333
Epoch 60/100
5/5 [==============================] - 4s 745ms/step - loss: 0.0073 - acc: 1.0000 - val_loss: 0.1526 - val_acc: 0.9500
Epoch 61/100
5/5 [==============================] - 4s 725ms/step - loss: 0.0126 - acc: 1.0000 - val_loss: 0.1347 - val_acc: 0.9500
Epoch 62/100
5/5 [==============================] - 5s 909ms/step - loss: 0.0074 - acc: 1.0000 - val_loss: 0.1281 - val_acc: 0.9500
Epoch 63/100
5/5 [==============================] - 4s 745ms/step - loss: 0.0044 - acc: 1.0000 - val_loss: 0.1341 - val_acc: 0.9500
Epoch 64/100
5/5 [==============================] - 4s 803ms/step - loss: 0.0041 - acc: 1.0000 - val_loss: 0.1483 - val_acc: 0.9500
Epoch 65/100
```

```
Epoch 66/100
5/5 [==============================] - 5s 950ms/step - loss: 0.0042 - acc: 1.0000 - val_loss: 0.1382 - val_acc: 0.9500
Epoch 67/100
5/5 [==============================] - 4s 743ms/step - loss: 0.0043 - acc: 1.0000 - val_loss: 0.1506 - val_acc: 0.9500
Epoch 68/100
5/5 [==============================] - 4s 748ms/step - loss: 0.0038 - acc: 1.0000 - val_loss: 0.1445 - val_acc: 0.9500
Epoch 69/100
5/5 [==============================] - 4s 710ms/step - loss: 0.0041 - acc: 1.0000 - val_loss: 0.1490 - val_acc: 0.9500
Epoch 70/100
5/5 [==============================] - 4s 843ms/step - loss: 0.0026 - acc: 1.0000 - val_loss: 0.1678 - val_acc: 0.9500
Epoch 71/100
5/5 [==============================] - 4s 702ms/step - loss: 0.0044 - acc: 1.0000 - val_loss: 0.1512 - val_acc: 0.9500
Epoch 72/100
5/5 [==============================] - 5s 1s/step - loss: 0.0021 - acc: 1.0000 - val_loss: 0.1537 - val_acc: 0.9500
Epoch 73/100
5/5 [==============================] - 5s 1s/step - loss: 0.0025 - acc: 1.0000 - val_loss: 0.1562 - val_acc: 0.9500
Epoch 74/100
5/5 [==============================] - 5s 941ms/step - loss: 0.0021 - acc: 1.0000 - val_loss: 0.1583 - val_acc: 0.9500
Epoch 75/100
```

```
In [12]: model.fit_generator(x_train,steps_per_epoch=5,epochs=100,validation_data=x_test,validation_steps=2)
Epoch 93/100
5/5 [==============================] - 5s 932ms/step - loss: 0.0021 - acc: 1.0000 - val_loss: 0.1786 - val_acc: 0.9500
Epoch 94/100
5/5 [==============================] - 6s 1s/step - loss: 0.0019 - acc: 1.0000 - val_loss: 0.1867 - val_acc: 0.9500
Epoch 95/100
5/5 [==============================] - 4s 821ms/step - loss: 0.0015 - acc: 1.0000 - val_loss: 0.1798 - val_acc: 0.9500
Epoch 96/100
5/5 [==============================] - 9s 2s/step - loss: 8.1228e-04 - acc: 1.0000 - val_loss: 0.1781 - val_acc: 0.9500
Epoch 97/100
5/5 [==============================] - 8s 2s/step - loss: 0.0010 - acc: 1.0000 - val_loss: 0.1772 - val_acc: 0.9500
Epoch 98/100
5/5 [==============================] - 5s 931ms/step - loss: 0.0017 - acc: 1.0000 - val_loss: 0.1846 - val_acc: 0.9500
Epoch 99/100
5/5 [==============================] - 4s 737ms/step - loss: 0.0011 - acc: 1.0000 - val_loss: 0.1900 - val_acc: 0.9500
Epoch 100/100
5/5 [==============================] - 3s 578ms/step - loss: 7.8182e-04 - acc: 1.0000 - val_loss: 0.1814 - val_acc: 0.950
0
Out[12]: <keras.callbacks.History at 0x269bf183808>
```

Accuracy when we train and test the model we got 95%

Fig11: Training and Evaluating the model

Next Here we need to optimize and save the model (.h5 – regressor file)

```
In [13]: model.save("forest1.h5")
```

HARDWARE AND SOFTWARE DESIGNING

2.5.1 Hardware

1) A laptop / computer
2) 8-GB RAM
3) 32-GB ROM
4) 64 Bit

2.5.2 Software

Front End:

1) HTML

Back End:

Python – We use following GUI's

1) (Anaconda Navigator -> Jupyter Notebook, Spyder) it is a desktop GUI
2) Anaconda Prompt

CHAPTER – 3

RESULTS AND DISCUSSIONS

3.1 EXPERIMENTAL RESULTS

When we feed it as a picture, it accurately does the prediction and tells whether it is a forest caught fire or a forest with no fire, and it will also alarm if it recognizes when the image as forest caught fire.

```
In [1]: from keras.models import load_model
        from keras.preprocessing import image

        import numpy as np
        import cv2
```

```
Using TensorFlow backend.
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1) or '1
type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1) or '1
```

```
In [2]: model = load_model("forest1.h5")
```

```
WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:517: The name tf.plac
eholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4138: The name tf.ran
dom_uniform is deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3976: The name tf.nn.
max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:174: The name tf.get_
default_session is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:181: The name tf.Conf
igProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:186: The name tf.Sess
ion is deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\keras\optimizers.py:790: The name tf.train.Optimizer is d
eprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\ops\math_grad.py:1250: add_dispatch_sup
port.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
In [3]: img = image.load_img(r'C:\Users\mahit\OneDrive\Desktop\AI intern proj1\withfire.jpg',target_size = (64,64))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis = 0)
```

```
In [4]: x.shape
```

```
Out[4]: (1, 64, 64, 3)
```

```
In [5]: pred = model.predict_classes(x)
```

```
In [6]: pred
```

```
Out[6]: array([0], dtype=int64)
```

```
In [7]: x1 = image.img_to_array(img)
```

```
In [8]: x1.shape
```

```
Out[8]: (64, 64, 3)
```

Fig12: loading model and image

Result of alert alarm in jupyter Notebook

It will forecast as follows: when we insert an image with fire, we will obtain a fire alarm in our system; otherwise, it will not generate any sound in the system.

In [14]:
```
from keras.models import load_model
import numpy as np
import cv2
model =load_model('forest1.h5')
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
from skimage.transform import resize
def detect(frame):
    try:
        img= resize(frame,(64,64))
        img = np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img =img/255.0
        prediction =model.predict(img)
        print (prediction)
        prediction_class = model.predict_classes(img)
        print(prediction_class)
        return prediction_class
    except AttributeError:
            print("shape not found")
frame= cv2.imread(r'C:\Users\mahit\OneDrive\Desktop\AI intern proj1\withoutfire.jpg')
data= detect(frame)
data
from playsound import playsound
if(data[0]==0):
    playsound(r'C:\Users\mahit\Downloads\Tornado_Siren.mp3')
#elif(data[0]==1):
   # playsound(r'C:\Users\DELL\Downloads\Annoying_Alarm_Clock-UncleKornicob-1.mp3')
```

```
[[2.7503056e-06 9.9999726e-01]]
[1]
```

Fig13: Without fire- Alarm off

In [*]:
```
from keras.models import load_model
import numpy as np
import cv2
model =load_model('forest1.h5')
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
from skimage.transform import resize
def detect(frame):
    try:
        img= resize(frame,(64,64))
        img = np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img =img/255.0
        prediction =model.predict(img)
        print (prediction)
        prediction_class = model.predict_classes(img)
        print(prediction_class)
        return prediction_class
    except AttributeError:
            print("shape not found")
frame= cv2.imread(r'C:\Users\mahit\OneDrive\Desktop\AI intern proj1\withfire.jpg')
data= detect(frame)
data
from playsound import playsound
if(data[0]==0):
    playsound(r'C:\Users\mahit\Downloads\Tornado_Siren.mp3')
#elif(data[0]==1):
   # playsound(r'C:\Users\DELL\Downloads\Annoying_Alarm_Clock-UncleKornicob-1.mp3')
```

```
[[0.6241354  0.37586457]]
[0]
```

Fig14: With fire -Alarm on

In loading the UI Page Anaconda Prompt is Used. Connects to the Server (localhost:5000)

Anaconda Prompt (anaconda3) - python app.py

```
(base) C:\Users\mahit>cd C:\Users\mahit\OneDrive\Desktop\AI intern proj1\flask codes\spyder

(base) C:\Users\mahit\OneDrive\Desktop\AI intern proj1\flask codes\spyder>python app.py
Using TensorFlow backend.
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing
t will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\mahit\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing
t will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
```

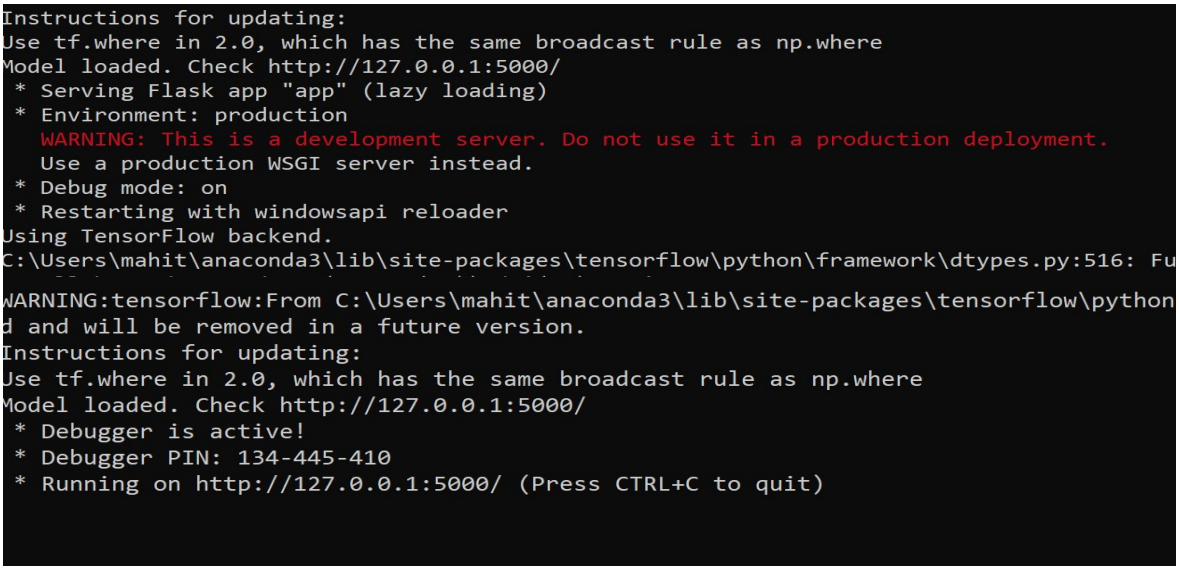

Fig15: Anaconda Prompt

3.2 BUILDING USER INTERFACE FOR THE MODEL:

A flak model was used to design a user interface. Flask is a Python-based web application framework. It features a number of modules that make it easy for a web developer to construct apps without having to worry about protocol management, thread management, and other such concerns.

Flask provides us with a number of options for constructing web apps, as well as the tools and libraries we'll need to get started. There are two html pages utilised, one for the basis and the other for the index. The purpose of these HTML pages is to provide a user interface between the user and the software.

We move on to the local host after running the flask application and are able to insert the picture from the destination. The picture we capture will be saved in the upload folder. All files must be saved in a single folder.

Screenshots of what steps we have done during the model building include:
Data Collection – Create Train and Test Folders

Flask folder -
Steps to be followed:
Create a main folder (spyder)
In that main folder create template folder and in that place .html file (base.html)
And create two folders static (contain css and jss) and uploads (images of prediction)
.py file (app.py) should be saved in that main folder
.h5 file (forest1.h5) which is downloaded from jupyter notebook should also be saved in main folder.

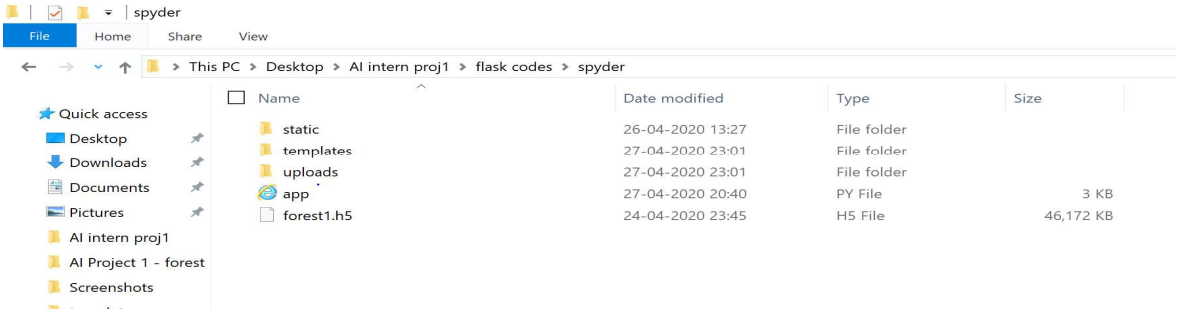

Fig16: Folder of Flask model

Static is where we keep some of the CSS and JS files that we want to use in our web app. The index and base html files are stored in templates. When the user takes a picture, the upload will be saved. The following code will be stored in the app, which connects the html files and stores the model. Building is the.h5 file in which we saved our Python model.

app.py

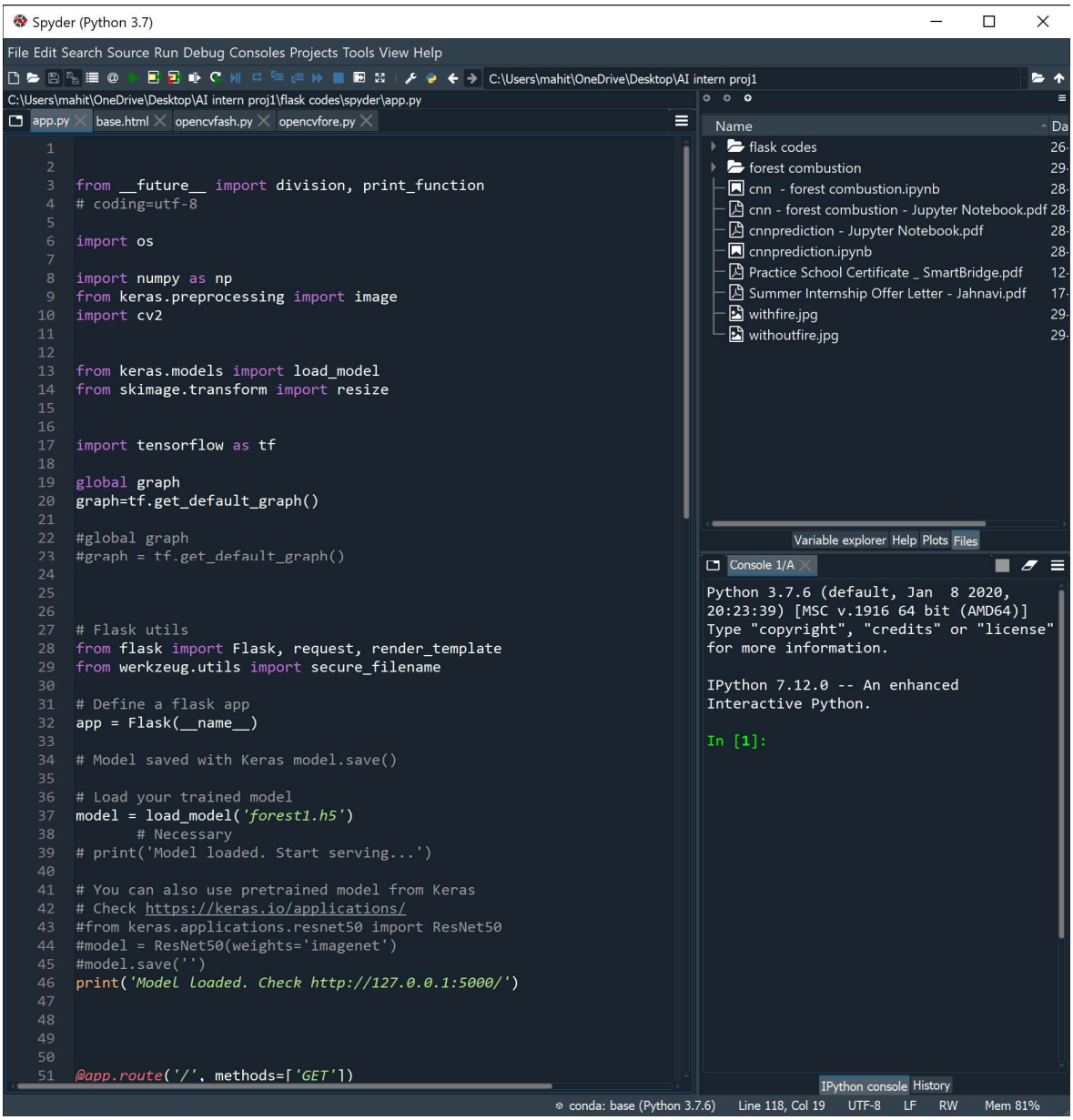

Fig17: Flask Code

base.html

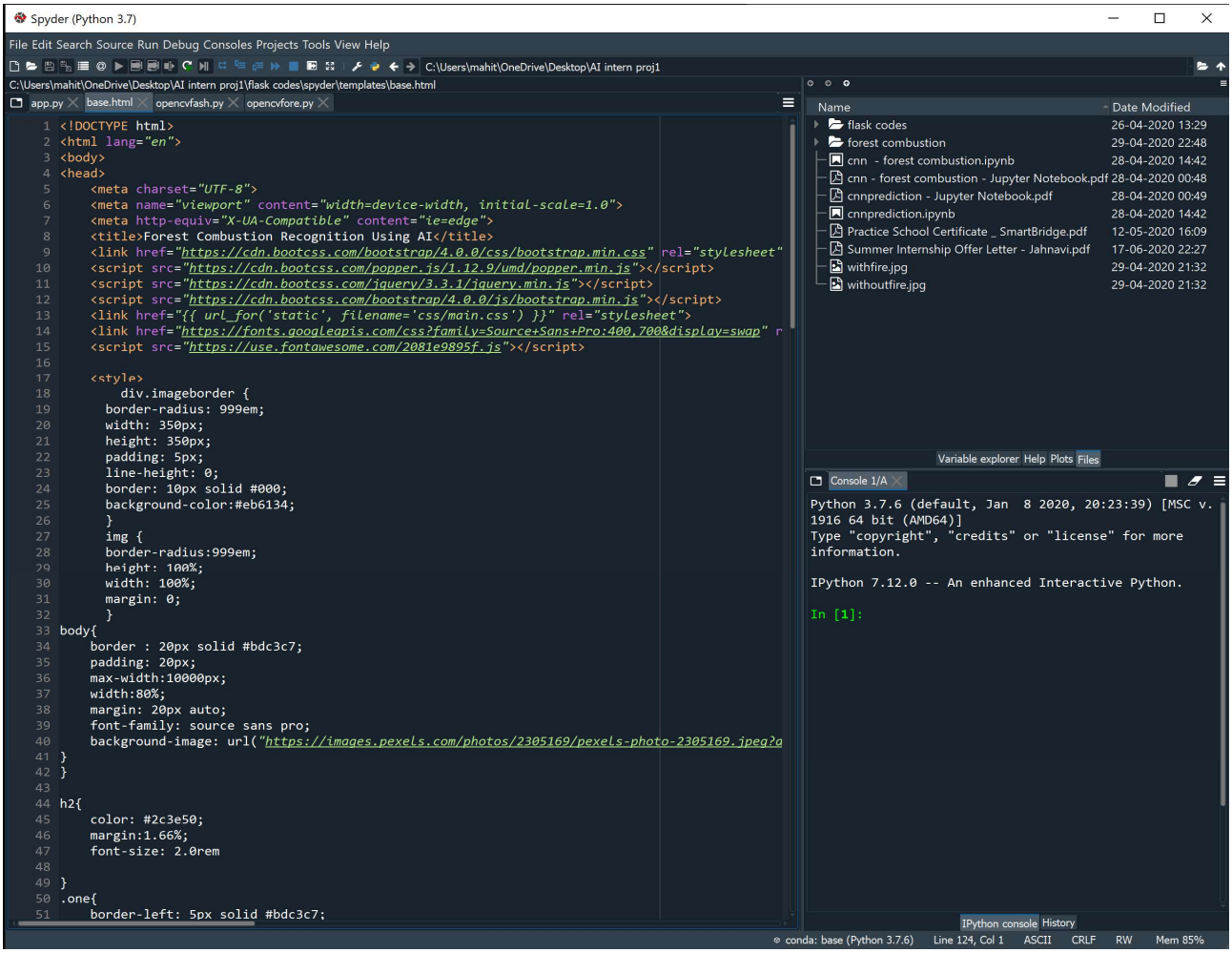

Fig 18: html code

3.3 PREDICTION:

As a result, when we choose an image to upload, the predict button appears, and when we click it, the model predicts whether the image is of a forest caught fire or with no fire. Below image is the user interface that was created by combining Python and HTML.

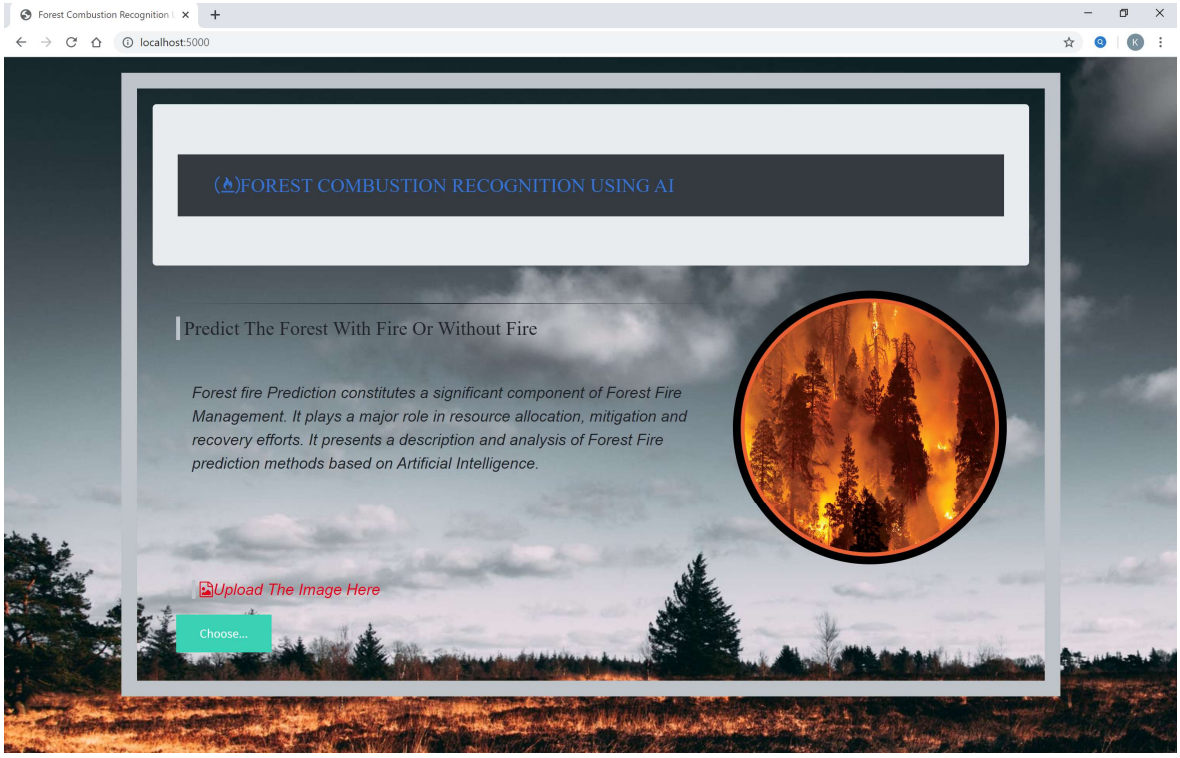

Fig19: Developed UI

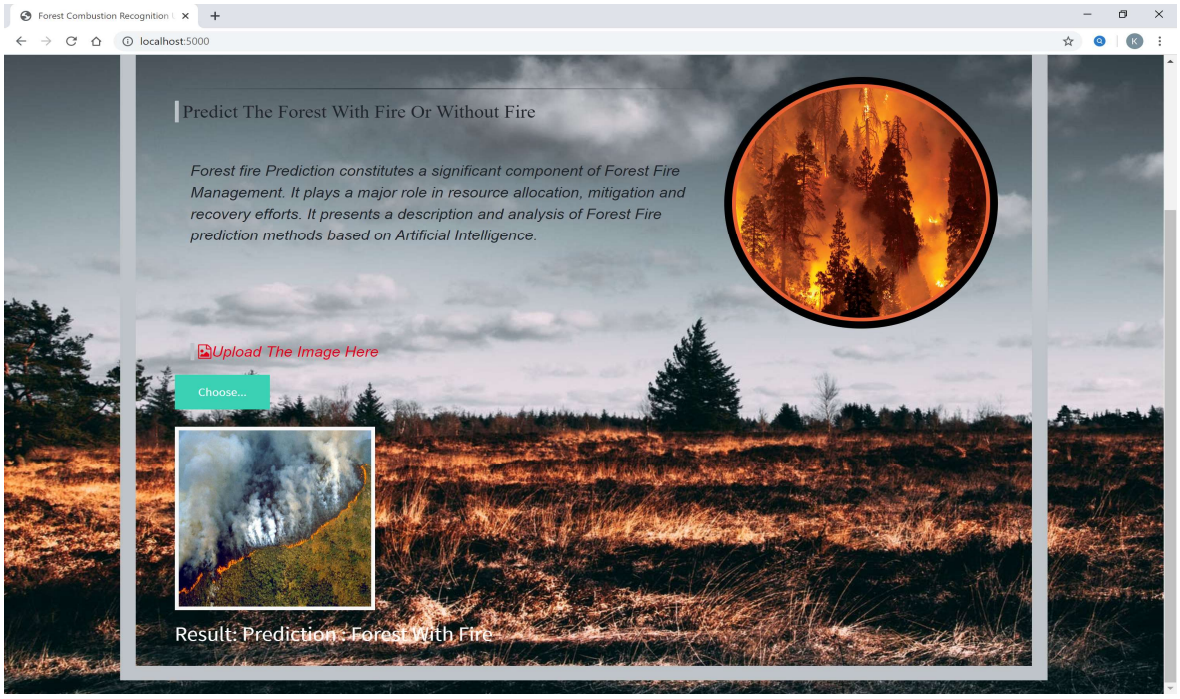

Fig20: Prediction of Forest With Fire

Fig21: Prediction of Forest Without Fire

The predicted outcome is displayed below the image

OpenCV

OpenCV (Open Source Computer Vision Library) is a free computer vision and machine learning software library. OpenCV was established to provide a standard infrastructure for computer vision applications and to aid in the faster integration of machine perception into commercial goods.
We can use OpenCV to open the camera and use it to forecast whether or not the forest is on fire.

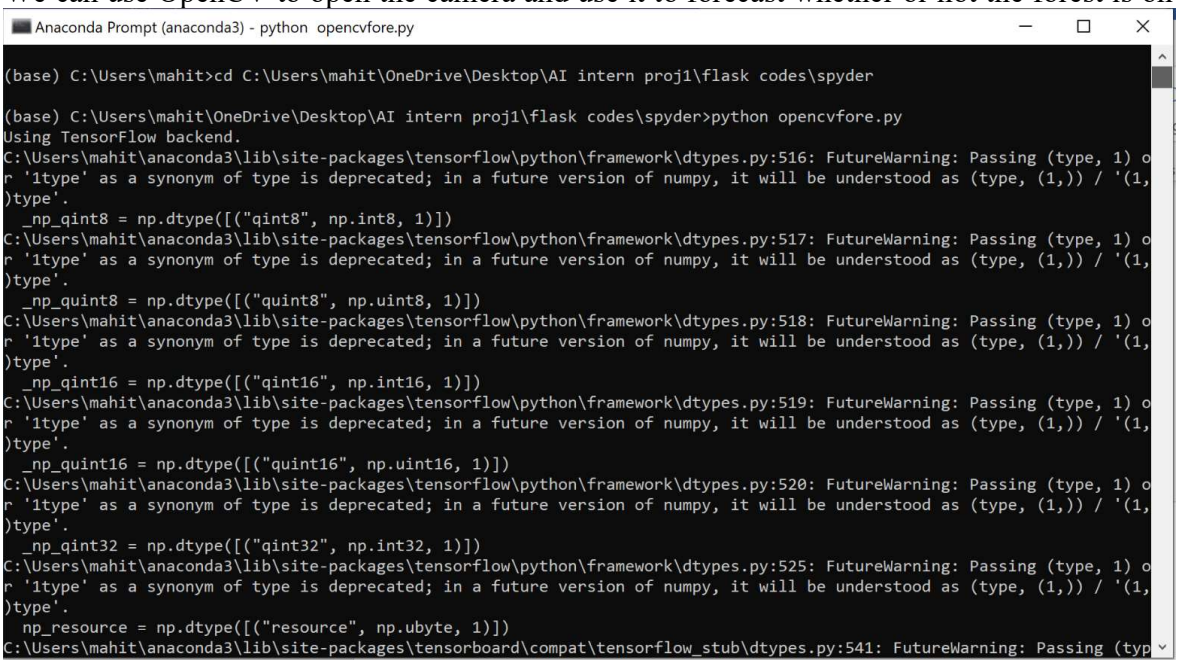

Fig 22 : anaconda prompt – OpenCV

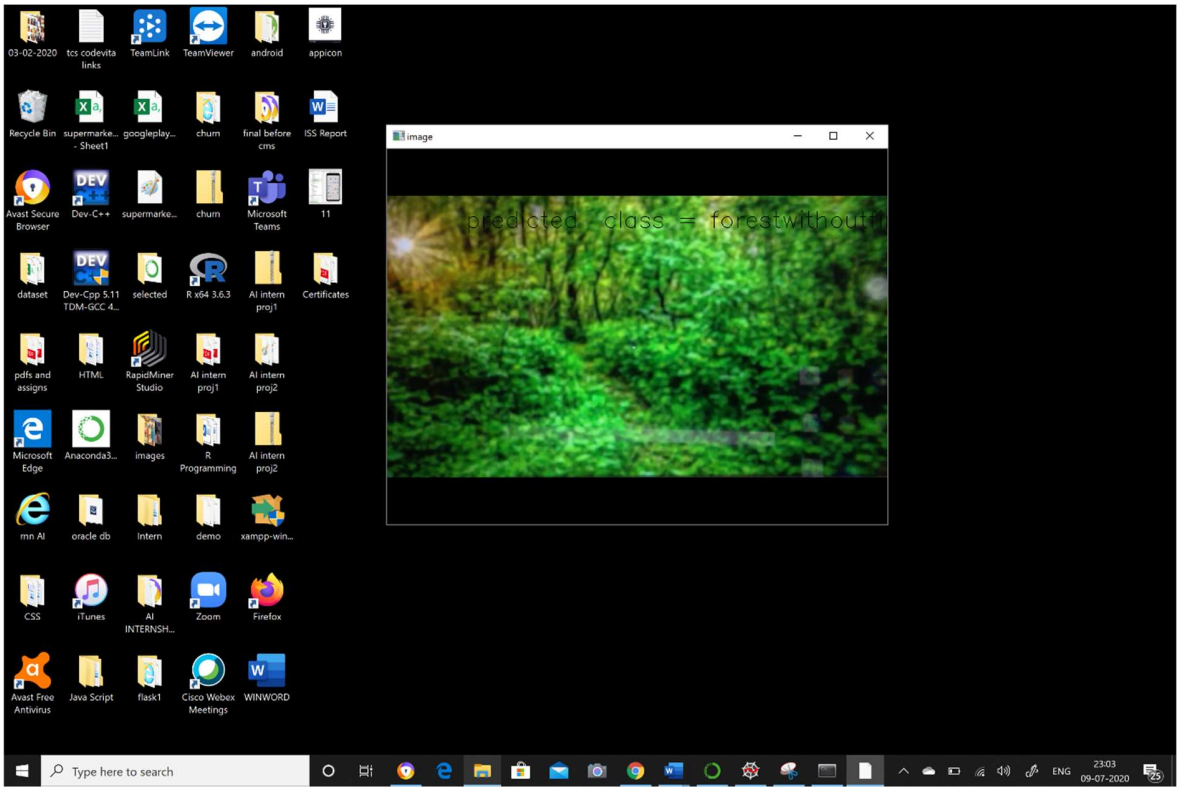

Fig 23 : predicted class – without fire

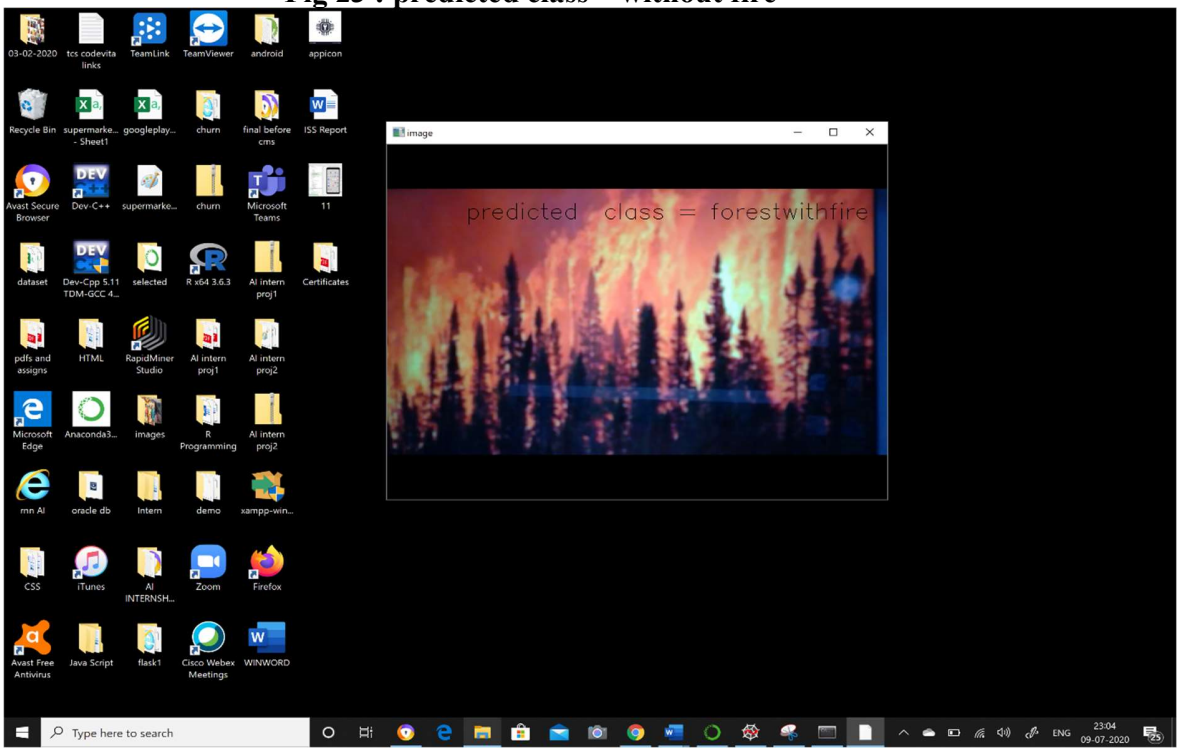

Fig 24 : predicted class – with fire

3.4 Block diagram

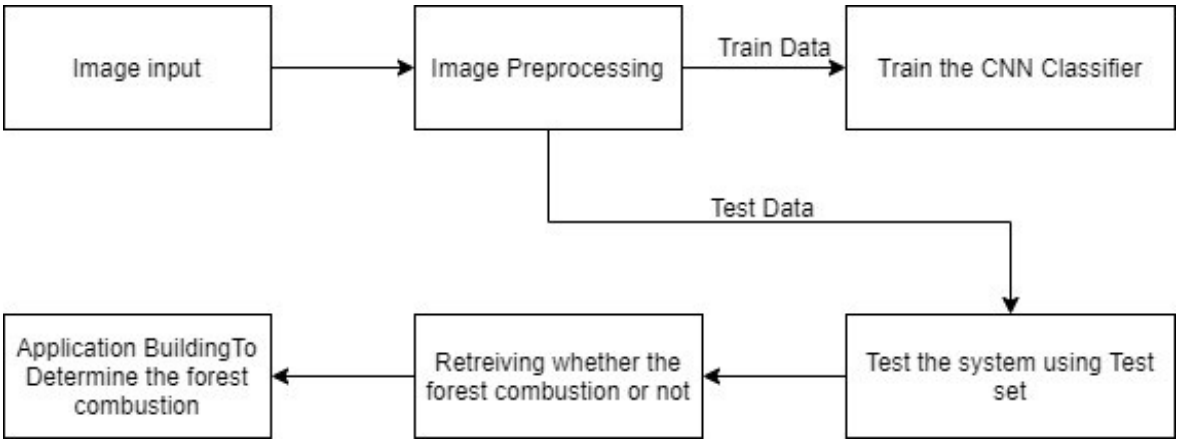

Fig 25 : block diagram

CHAPTER-4

4.1 CONCLUSION

We used AI, deep learning techniques (CNN), and Python 3 to create this project. Using image processing techniques, this study provides an effective forest fire detection approach. Train and test models are used in the method. It will train our model and then test whether or not there is a fire. To detect the fire, five fire detection criteria are used. The proposed algorithm's performance is evaluated using a data set consisting of diverse forest photographs gathered from the Internet, 100 of which were actual fire pictures while the other 100 were not. The findings reveal that the proposed method has a high detection rate. These findings suggest that the proposed technology is accurate and suitable for application in automatic forest fire detection systems.

4.2 FUTURE SCOPE

The most important requirements for an autonomous early forest combustion recognition system, as well as its core modules and methodologies for risk assessment and wildfire management, combustion and smoke detection, have all been met using photos. We trained and tested the model, which had a 95% accuracy and produced an alarm sound in Jupyter Notebook. We want to improve this UI tied to the page of html by adding an alert to our system i.e., alarm and making our model run faster and more precisely.

APPENDIX

CODE

CNN CODE (Jupyter notebook)

```
#importing the libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.models import load_model
import numpy as np
import cv2
from skimage.transform import resize
model = Sequential()
model.add(Convolution2D(32,(3,3), input_shape = (64,64,3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(units = 128, init ='uniform', activation ='relu'))
model.add(Dense(units = 128, init ='uniform', activation ='relu'))
model.add(Dense(output_dim = 2, activation = 'softmax', init ='uniform'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
from keras.preprocessing.image import  ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2,
horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
x_train = train_datagen.flow_from_directory(r'C:\Users\mahit\OneDrive\Desktop\AI intern
proj1\forest combustion\trainset',target_size = (64,64),batch_size = 32,class_mode = 'categorical')
x_test = test_datagen.flow_from_directory(r'C:\Users\mahit\OneDrive\Desktop\AI intern
proj1\forest combustion\testset',target_size = (64,64),batch_size = 32,class_mode = 'categorical')
print(x_train.class_indices)
```

```
model.compile(loss= 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])
model.fit_generator(x_train,steps_per_epoch=5,epochs=100,validation_data=x_test,validation_steps=2)
model.save("forest1.h5")
```

app.py code

```
from __future__ import division, print_function
# coding=utf-8
import os
import numpy as np
from keras.preprocessing import image
import cv2
from keras.models import load_model
from skimage.transform import resize
import tensorflow as tf
global graph
graph=tf.get_default_graph()
#global graph
#graph = tf.get_default_graph()
# Flask utils
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer

# Define a flask app
app = Flask(__name__)
# Model saved with Keras model.save()
# Load your trained model
model = load_model('forest1.h5')
      # Necessary
# print('Model loaded. Start serving...')
# You can also use pretrained model from Keras
# Check https://keras.io/applications/
#from keras.applications.resnet50 import ResNet50
#model = ResNet50(weights='imagenet')
#model.save('')
```

```
print('Model loaded. Check http://127.0.0.1:5000/')
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('base.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(64, 64))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        with graph.as_default():
            preds = model.predict_classes(x)
            #preds = [0]
            #type(preds)
            #print(preds[0])
            #print("prediction",preds)
  index = ['Forest With Fire','Forest Without Fire']
        #text = index[0]
        #print(text)
        text = "Prediction : "+index[preds[0]]
       # ImageNet Decode
        return text
from keras.models import load_model
model =load_model('forest1.h5')
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
def detect(frame):
    try:
        img= resize(frame,(64,64))
        img = np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img =img/255.0
        prediction =model.predict(img)
        print (prediction)
        prediction_class = model.predict_classes(img)
        print(prediction_class)
        return prediction_class
    except AttributeError:
        print("shape not found")
if __name__ == '__main__':
    app.run(debug = True,threaded = False)
    frame= cv2.imread(r'C:\Users\mahit\OneDrive\Desktop\AI intern proj1\withfire.jpg')
data= detect(frame)
data
from playsound import playsound
if(data[0]==0):
    playsound(r'C:\Users\mahit\Downloads\Tornado_Siren.mp3')
#elif(data[0]==1):
    #playsound(r'C:\Users\DELL\Downloads\Annoying_Alarm_Clock-UncleKornicob-1.mp3')
```

base.html

```
<!DOCTYPE html>
<html lang="en">
<body>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Forest Combustion Recognition Using AI</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:400,700&display=swap"
rel="stylesheet" >
    <script src="https://use.fontawesome.com/2081e9895f.js"></script>
<style>
                div.imageborder {
    border-radius: 999em;
    width: 350px;
    height: 350px;
    padding: 5px;
    line-height: 0;
    border: 10px solid #000;
    background-color:#eb6134;
    }
    img {
    border-radius:999em;
    height: 100%;
```

```
width: 100%;
     margin: 0;
     }
body{
          border : 20px solid #bdc3c7;
   padding: 20px;
   max-width:10000px;
   width:80%;
   margin: 20px auto;
   font-family: source sans pro;
   background-image: url("https://images.pexels.com/photos/2305169/pexels-photo-
2305169.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940");
}
}

h2{
          color: #2c3e50;
          margin:1.66%;
          font-size: 2.0rem

}
.one{
          border-left: 5px solid #bdc3c7;
          font-size: 25px;
          <!--color: #ffbf00;-->
          color: #000000;
   font-family: Aharoni;
          padding-left: 5px;
}
.date{
          color: #3498db;
          margin:1.66%;
          letter-spacing: 0.2rem;
```

```
}
p{
        margin:1.66%;
}
hr{
   border: 0;
   height: 1px;
   background-image: linear-gradient(to right, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.75), rgba(0, 0, 0, 0));
}
 post{
        margin:20px;
}
h1{
        color: #3474db;
        margin:1.66%;
        font-size: 1.3rem;
        font-family:Arial Black Web Safe Font;
        text-align: right;
}
h1 two{
        color: #3474db;
        font-size: 25px;
        font-family:Arial Black Web Safe Font;
        text-align: right;
}
 post{
   font-size: 20px;
   font-style: oblique;
        font-family: sans-serif
        margin:20px;
}

#result {
```

```
        color:  #FFFFFF;
}
p.big {
  line-height: 1.8;
}
 .lg
 {
        color: #3474db;
 }
 .three
 {
        color: #e6001a;
 }
 .upload-labe one
 {
        color: #e6001a;
 }
        </style>
</head>
 <!--<source src=r"C:\Users\mahit\Downloads\Tornado_Siren.mp3" type="audio/mp3">-->
   <div class="jumbotron">
   <nav class="navbar navbar-expand-lg navbar-light bg-dark">
      <div class="container">
         <h1 class="two" ><i class="fa fa-free-code-camp fa lg" aria-hidden="true"></i>FOREST
COMBUSTION RECOGNITION USING AI
         </h1>
      </div>
       </div>
   </nav>
   <div class="container">
      <div id="content" style="margin-top:2em">
                  <div class="container"
 <div class="row">
```

```
                    <div class="col-sm-8 bd" >
                <!--<div><h3 class="one"> Forest Combustion Recognition </h3></div>-->
                    <br>
                <h3 class="one"> Predict The Forest With Fire Or Without Fire </h3>
                    <br>
 <p class="post">Forest fire Prediction constitutes a significant component of Forest Fire
Management. It plays a major role in resource allocation, mitigation and recovery efforts. It
presents a description and analysis of Forest Fire prediction methods based on Artificial
Intelligence.</p>
                    </div>
                    <div class="imageborder" >
<imgsrc="https://magazine.columbia.edu/sites/default/files/styles/wysiwyg_full_width_image/pu
blic/2018-09/Wild-fires.jpg?itok=pBKjTId4"height="350" width="350" alt="iceland" >
                    </div>
                    <div class="col-sm-5">
                    <div>

<h4 class="one post three" ><i class="fa fa-file-image-o" aria-hidden="true">
                        </i>Upload The Image Here </h4>

<form action = "http://localhost:5000/" id="upload-file" method="post" enctype="multipart/form-
data">
<label for="imageUpload" class="upload-label">
                            Choose...
                        </label>
        <input type="file" name="image" id="imageUpload" accept=".png, .jpg, .jpeg">

                    </div>
                    <br>
                </form>
                    </div>
                    <div class="image-section" style="display:none;">
                        <div class="img-preview">
```

```
                                    <div id="imagePreview">
                                    </div>
                                </div>
                                <div>
<button type="button" class="btn btn-info btn-lg " id="btn-predict">Predict!</button>
                                    <!--<audio
src=r"C:\Users\mahit\Downloads\Tornado_Siren.mp3" controls>-->
                                    </audio>
                                </div>
                            </div>
                            <div class="loader" style="display:none;"></div>
                            <h3>
                            <span id="result"> </span>
                            </h3></div>
                            </div>
                        </div>
                    </div>
                    </div>
    </div>
</body>
<footer>
    <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>
</footer>
</body>
</html>
```

Opencvfore.py

```
import cv2
#import facevec
import numpy as np
from keras.preprocessing import image
from keras.models  import load_model
model = load_model('forest1.h5')
video = cv2.VideoCapture(0)
name = ["forestwithfire","forestwithoutfire"]
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)
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('a'):
        break
video.release()
cv2.destroyAllWindows()
```

REFERENCES

Anon, 1995. Report of the Committee of Enquiry into Forest Fires in Himachal Pradesh and Uttar Pradesh in 1995, Government of India.

Bahuguna, V.K. and Upadhyay, A., 2002. Forest fires in India: Policy initiatives for community participation; International Forestry Review, 4(2).

FAO, 2003.Wildland Fire Management Terminology. FAO Forestry Paper 70. FAO, Rome.

https://www.researchgate.net/publication/261272818_Artificial_intelligence_for_forest_fire_prediction

https://ieeexplore.ieee.org/document/5695809/metrics#metrics

BIODATA

Name : Sai Sheshank Gaddam
Mobile Number : 9133499615
E-mail : saisheshank.gaddam@vitap.ac.in
Permanaent Address : 1 – 2 – 155/1, Tagore Raod, Gunj, Bhongir, Yadadri Bhongri Distrct, Telangana, India.

PPT's – Drive Link

https://drive.google.com/drive/folders/1Pkg-eMQJx1hE2zQrMGLXBuTxPhMLigeQ?usp=sharing

Video Link –

https://drive.google.com/drive/folders/1_ZosdFV9DHnB-IaSgVPcbmHhLtvEhYK8?usp=sharing

Poster - https://docs.google.com/presentation/d/1ptN7HgkfsiaWP36fiG2X6vw50_2EwwqN/edit?usp=sharing&ouid=104324311414189456238&rtpof=true&sd=true