Final Project Report- CSE1901

SMART AI FOREST FIRE DETECTION

Faculty- Dr. M. Premalatha

1. Team members

1. Pradhuman singh 19BCE1548
2. Akarsh Reddy. T 19BCE1552
3. Magesh Rakupathi 19BCE1554

2. Code of Academic Integrity

I the undersigned solemnly declare that the project is based on my own work carried out during the course of our study under the supervision of. I assert the statements made and conclusions drawn are an outcome of my research work.

I further certify that the work contained in the report is original and has been done by all the members of the team. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.

We have followed the guidelines provided by the faculty in writing the report. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

3. Motivation

In recent times, we have heard about a lot of disasters which involves forest fire. These fires are getting really common so our team has decided to come up with a solution. In this Smart Fire Detection Project, we will be using OpenCV to detect fire and code the program in Python. We will use datasets available on the internet to train our model and when it detects fire, it will alert the nearby user and guards so that early action can be taken. This software will help prevent spread of fire at a place when it’s no longer under human attention.

4. Literature Survey

Design of intelligent fire alarm system based on GSM network

Author: Chun-yuan Lian

Changzhou Institute of Technology, School of Electronic Information & Electric Engineering, China

Review:

An intelligent fire alarm system based on GSM network is designed in order to solve the problem of complex cabling, misdeclaration and missing alarm of traditional fire alarm systems. MSP430F149 is adopted as main control chip, and GSM module TC35I is used for remote alarming and data exchanging.

Result: this design has characters of real-time and good reliability, and will be widely used.

Design and Realization of Fire Alarm by Determining Probability Based on Multi-sensor Integrated

Authors: Xia Huanxiong, Sun Shuwen, Yao Yiwu, Guo Wenzeng, Zhang Shanshan, Wang Jie (College of Mechanical Engineering & Applied Electronics Technology, Beijing Univ. of Technology, Beijing 100124, China)

Due to the shortage of detecting characteristic parameters, a single detection signal based fire alarm usually causes leak-check, error and solidified conditions and parameters. In order to reduce such a situation, we design an intellectualized fire alarm system where C8051F040 MCU collects temperature, smoke and light intensity sensor signals, and uses continuous probability models of a comprehensive analysis and processing.

Result: The alarm can accurately and quickly respond to the probability of the fire process, and give early warning, alarm signals and act related firefighting equipment.

Fire Detection using Artificial Intelligence for Fire-Fighting Robots

Authors: Sreesruthi Ramasubramanian (School of Engineering and Physical Sciences, Heriot-Watt University, Dubai, United Arab Emirates), Senthil Arumugam Muthukumaraswamy (School of Engineering and Physical Sciences, Heriot-Watt University, Dubai, United Arab Emirates), A. Sasikala (Department of Electrical and Electronics Engineering, Srisairam Institute of Technology, Chennai, India)

Fire-fighting robots are used in indoor environments to detect fires and extinguish them. Sensors such as flame sensors are currently used to detect fire in fire-fighting robots. The disadvantage of using sensors is that fire beyond a threshold distance cannot be detected. Using artificial intelligence techniques, fire can be detected in a wider range. Haar Cascade Classifier is a machine-learning algorithm that was initially used for object detection. The results obtained using Haar Cascade Classifier were not very accurate, especially when multiple fires had to be detected. Transfer learning from a pre-trained YOLOv3 model was then used to train the model for fire detection to improve accuracy. The benefits and drawbacks of using deep learning for object detection over machine learning are highlighted. The algorithm used to obtain the target location the robot must move to use bounding box coordinates is also discussed in this paper.

5. Methodology

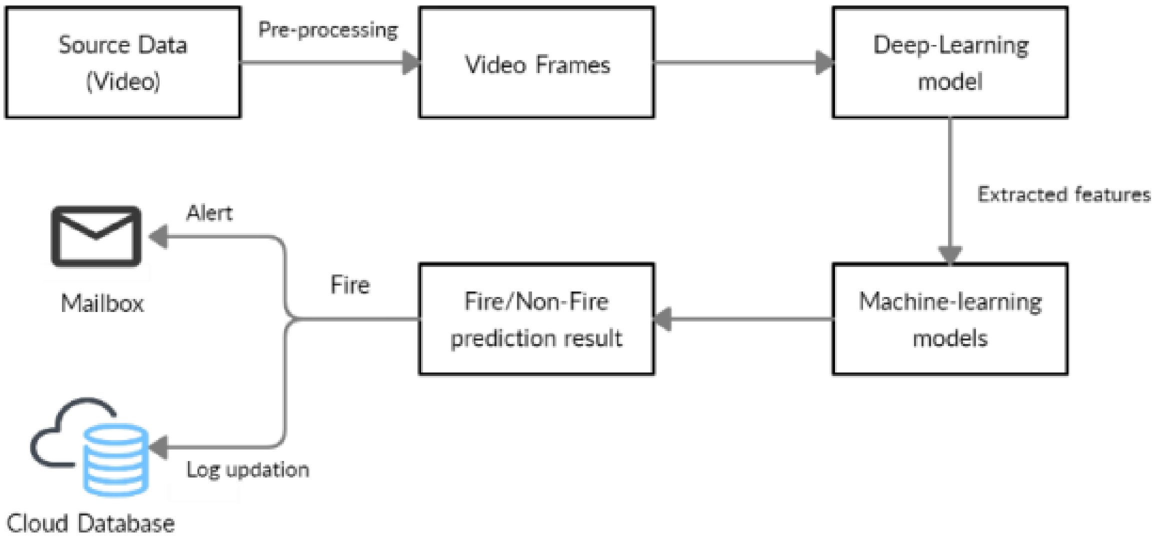
Creating the customized CNN architecture

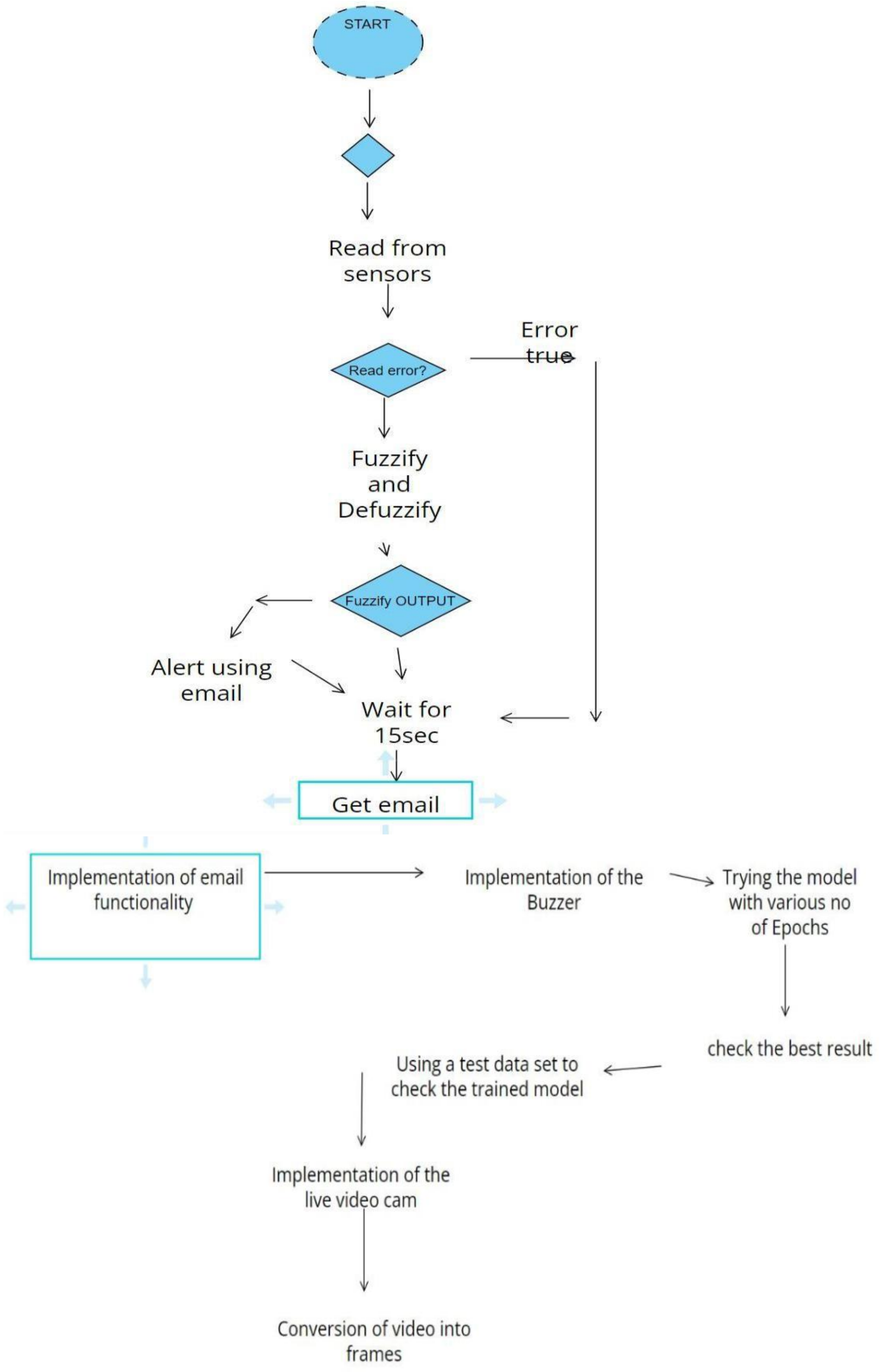
We are going to use TensorFlow API Keras for building our model. Let's first create our ImageDataGenerator for labelling our data. [1] and [2] datasets are used here for training. Finally, we will have 980 images for training and 239 images for validation. We are going to use data augmentation as well. We will use Adam as an optimizer with a learning rate of 0.0001. After training for 50 epochs, we get the training accuracy of 96.83 and validation accuracy of 94.98. The training and validation loss is 0.09 and 0.13 respectively.

Description

This smart fire detection system automatically detects the fire in real time and alert us thus limiting its further spread. we will train the system with positive and negative images thus all these data sets will help the system learn and analyse in real time that weather the image or the video they are feed with has a fire in it or not.

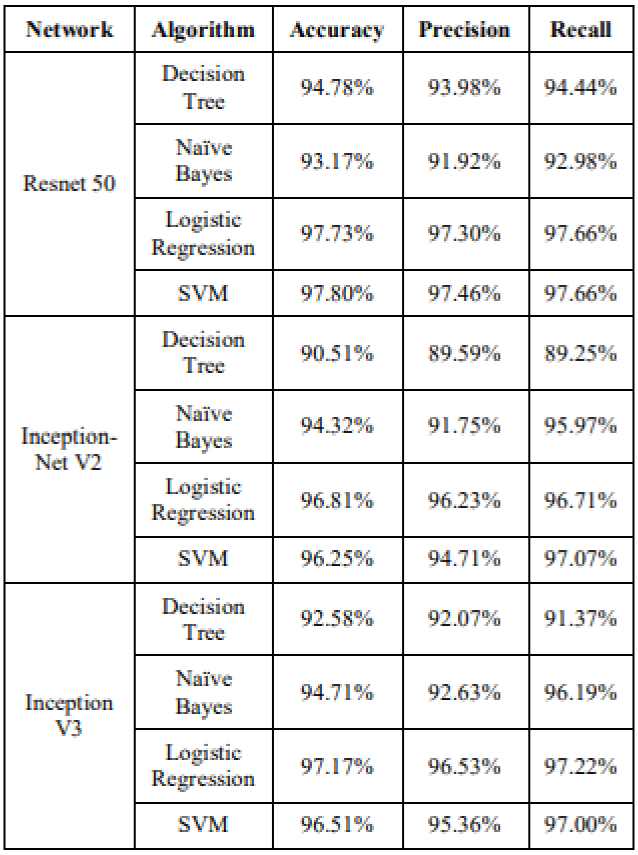
The advanced time of gadgets has seen a ton of improvement. Be that as it may, the advancement of electrical and electronic gadgets has prompted a developing pace of fire mishaps basically because of the indiscretion. The primary issue in the fire mishaps is that they are not having the option to recognize the fire as ahead of schedule as could really be expected. When we distinguish the fire it becomes hard to securely empty all or even lessen the harm caused. Our aim is to detect the fire as early as possible. The main objective is to detect the fire as early as possible. This can be attained by capturing the fire and motions of the spreading fire by using CCTV system and processing it to produce an alert. The processed image is compared with the pre fed image using ML and checked for fire accident. If the processed image matches the pre fed image then an alert message along with a buzzer is enabled. In simple word, this fire monitoring system is an addition to the existing CCTV cameras to detect the fire effectively.




START
Read from sensors
Read error?
Error true
Fuzzify and Defuzzify
Fuzzify OUTPUT
Alert using email
Wait for 15sec
Get email
Implementation of email functionality
Implementation of the Buzzer
Trying the model with various no of Epochs
check the best result
Using a test data set to check the trained model
Implementation of the live video cam
Conversion of video into frames

6.comparision

All the possible feature-extractor and classifier combinations were evaluated using stratified K-fold validation. Table 1 shows the obtained performance metric values for different combinations of deep learning networks and classifier algorithms. The best performance was observed with ResNet50 as the deep learning feature extraction model and Support Vector Machine as the ML classifier and the Accuracy, Precision and Recall values for this combination was 97.8%,97.46% and 97.66% respectively. Hence this would be used in our application.



Network	Algorithm	Accuracy	Precision	Recall
Resnet 50	Decision Tree	94.78%	93.98%	94.44%
	Naïve Bayes	93.17%	91.92%	92.98%
	Logistic Regression	97.73%	97.30%	97.66%
	SVM	97.80%	97.46%	97.66%
Inception-Net V2	Decision Tree	90.51%	89.59%	89.25%
	Naïve Bayes	94.32%	91.75%	95.97%
	Logistic Regression	96.81%	96.23%	96.71%
	SVM	96.25%	94.71%	97.07%
Inception V3	Decision Tree	92.58%	92.07%	91.37%
	Naïve Bayes	94.71%	92.63%	96.19%
	Logistic Regression	97.17%	96.53%	97.22%
	SVM	96.51%	95.36%	97.00%

7.Solution of the problem

The aim of our project was to develop to an application capable of detecting fire in videos and images, as well as live videos which is robust and works in any environment. In this regard, we have experimented with various deep learning models and classification models. An email alert feature has also been incorporated to our application to provide real time alerts to the concerned stakeholders along with a logging system, which is implemented using Firebase. The application performed exceptionally well during testing. It was able to identify fires in all of the twelve test fire videos but misclassified some instances of non-fire videos.

8. Impact of the project

The purpose of this document is to present a detailed description of the Smart Fire Detection System. It will explain the purpose and features of the system and what the system will do. This document is intended for both the stakeholders and the developers of the system. Smart Fire Detection project will be applicable at various places such as institutions, homes, factories, etc. It will be more efficient and easier way to help prevent big fires causing loss of lives and huge damages. Notifications of potential fires which user can easily access according to his rights makes it better as compared to the traditional ways of fire surveillance.

9. Roles and Contribution

We as a team will divide the work in 3 parts: -

1. Data pre-processing and model building (Pradhuman)

In data

2.Python code for alarm system and email notification (Akarsh)

3.Data collection and datasets collection (Magesh)

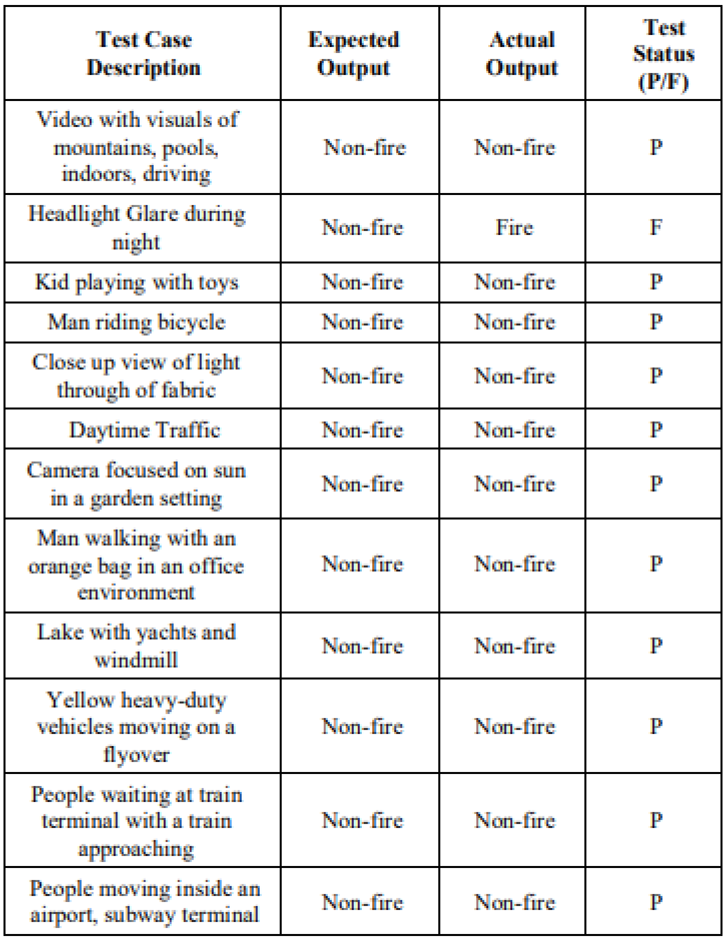
Implementation in first 30 days of the project

1. To detect fire and for the machine to study we have collected good images (without fire) and bad images (with fire) sample.

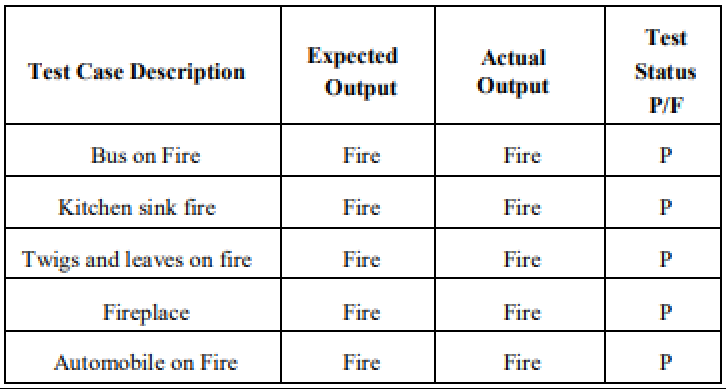
2. We started writing the python code for the project to establish and to develop a frontend.

3. We also have stated OpenCV code so as to detect fire and identify it properly.

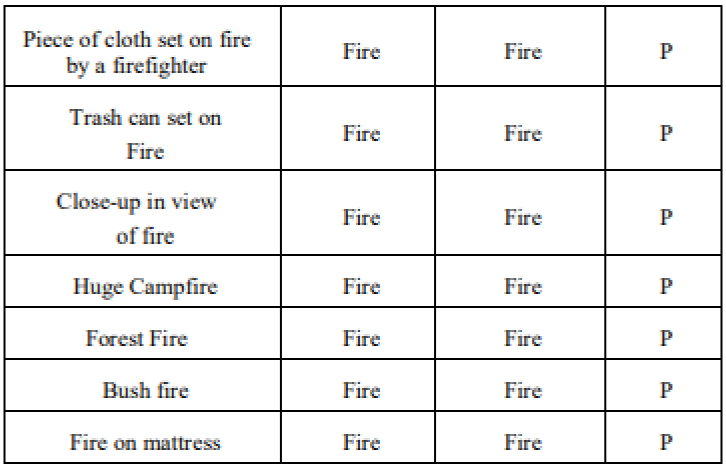
Details of all the experiments conducted so far



Test Case Description	Expected Output	Actual Output	Test Status (P/F)
Video with visuals of mountains, pools, indoors, driving	Non-fire	Non-fire	P
Headlight Glare during night	Non-fire	Fire	F
Kid playing with toys	Non-fire	Non-fire	P
Man riding bicycle	Non-fire	Non-fire	P
Close up view of light through of fabric	Non-fire	Non-fire	P
Daytime Traffic	Non-fire	Non-fire	P
Camera focused on sun in a garden setting	Non-fire	Non-fire	P
Man walking with an orange bag in an office environment	Non-fire	Non-fire	P
Lake with yachts and windmill	Non-fire	Non-fire	P
Yellow heavy-duty vehicles moving on a flyover	Non-fire	Non-fire	P
People waiting at train terminal with a train approaching	Non-fire	Non-fire	P
People moving inside an airport, subway terminal	Non-fire	Non-fire	P



Test Case Description	Expected Output	Actual Output	Test Status P/F
Bus on Fire	Fire	Fire	P
Kitchen sink fire	Fire	Fire	P
Twigs and leaves on fire	Fire	Fire	P
Fireplace	Fire	Fire	P
Automobile on Fire	Fire	Fire	P



Piece of cloth set on fire by a firefighter	Fire	Fire	P
Trash can set on Fire	Fire	Fire	P
Close-up in view of fire	Fire	Fire	P
Huge Campfire	Fire	Fire	P
Forest Fire	Fire	Fire	P
Bush fire	Fire	Fire	P
Fire on mattress	Fire	Fire	P

Details of individual contributions of team members

TEAM MEBERS	WEEK 1	WEEK 2	WEEK 3
PRADHUMAN SINGH 19BCE1548	INTIALCODE OF THE FRONT END	DID REASERCH ANOUT DIFFERENT MODELS	FINISHED THE FRONT IMPLEMENTATION AND MODEL
AKARSH REDDY 19BCE1552	WORKED ON PROCESSING THE IMAGES	STUDED VARIOUS PAPERS TO PASS THE VIDEO INPUT	IMPLEMENTATION OF ALARM SYSTEM AND EMAIL NOTIFICATION
MAGESH RAKUPATHI 19BCE1554	COLLCETED SAMPLE IMAGES	RESEARCH FOR VIDEO TO FRAMES FOR PROCSSING	IMPLEMENTATION OF VIDEO TO FRAMES FOR PROCSSING

10. Other Information

10.1 Abstarct

The advanced time of gadgets has seen a ton of improvement. Be that as it may, the advancement of electrical and electronic gadgets has prompted a developing pace of fire mishaps basically because of the indiscretion. The primary issue in the fire mishaps is that they are not having the option to recognize the fire as ahead of schedule as could really be expected. When we distinguish the fire it becomes hard to securely empty all or even lessen the harm caused. Our aim is to detect the fire as early as possible. The main objective is to detect the fire as early as possible. This can be attained by capturing the fire and motions of the spreading fire by using CCTV system and processing it to produce an alert. The processed image is compared with the pre fed image using ML and checked for fire accident. If the processed image matches the pre fed image then an alert message along with a buzzer is enabled. In

simple word, this fire monitoring system is an addition to the existing CCTV cameras to detect the fire effectively.

10.2 Personal Observation

This fire detector will help us to prevent forest fire. And its spread.

10.3 Challenges faced during the project

1.we faced difficulty while detecting the fire point or spot it use to detect any bright object as fire so we had to train it with more no of datasets.

2.we had problem connecting our code with the live webcam and had to go though and read about various directories.

3.overfitting of the data

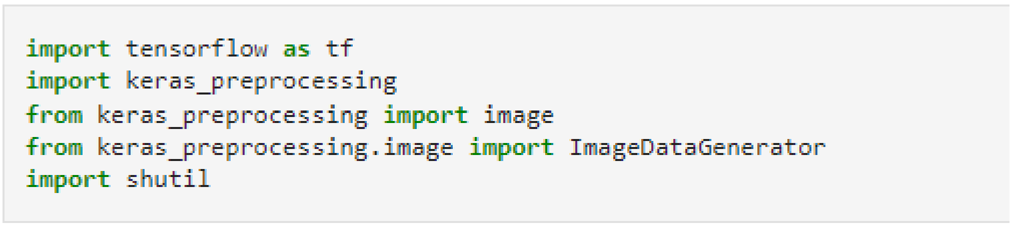
10.4 Scope of converting the project to a product/research paper

This project has a lot of capabilities as per us. This project enables us to detect the fire much faster than the traditional fire detector and can be a life saviour in many cases. Usually in case of a traditional fire detector, it detects the fire when the damage is done and nothing could be done about it but our fire detector on the other hand doesn't detect the fire by sensing the smoke but detects the fire by the colour gradient.

It can be converted into a product and can be installed at places where fire can spread easily thus alarming us before the onset of vast forest fire .

11. Code and Output

CODE

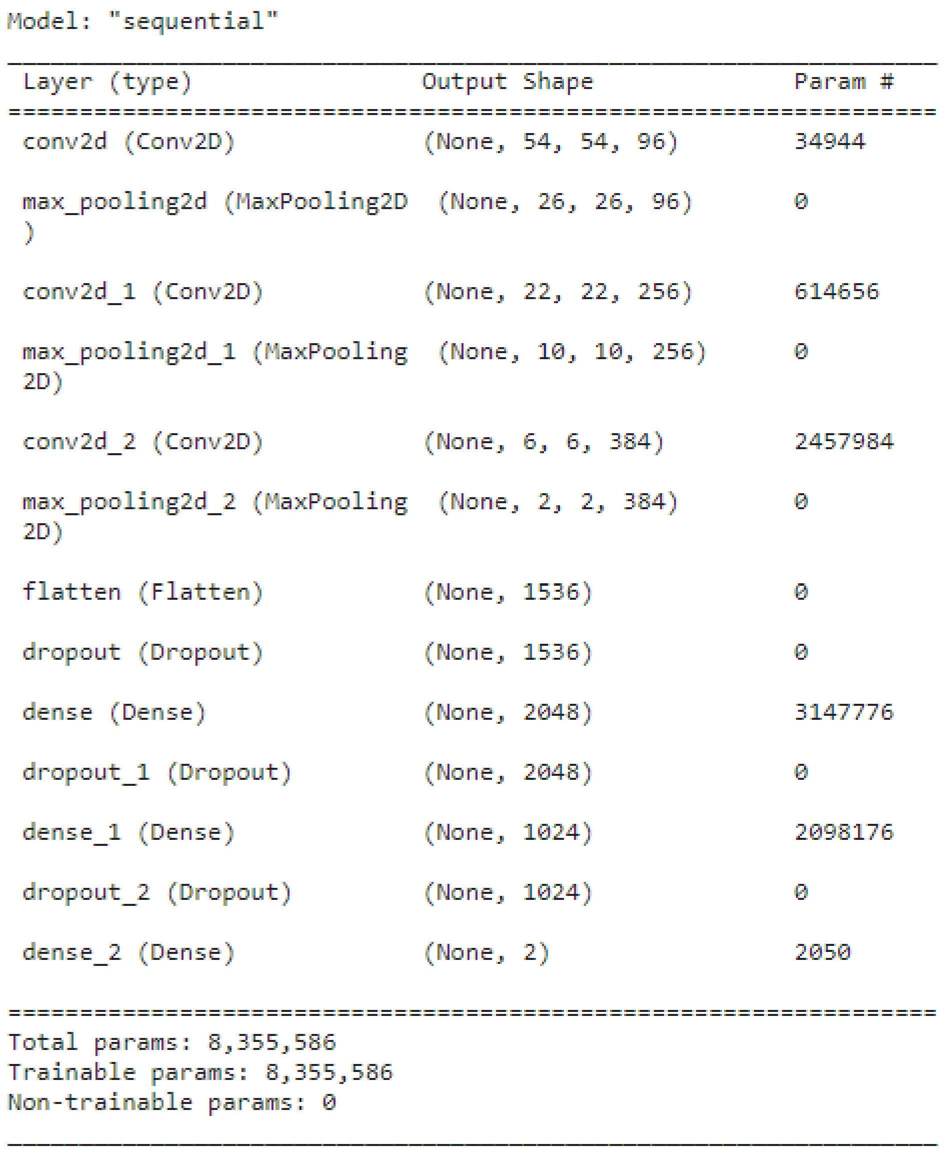


```
import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator
import shutil
```

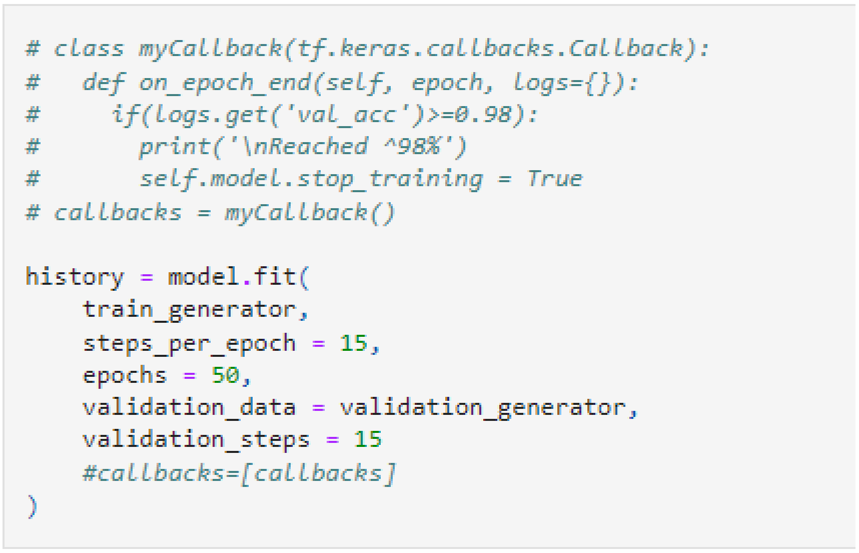
```
TRAINING_DIR = "Datasets 1-2\Training"

training_datagen = ImageDataGenerator(rescale = 1./255,
                                      horizontal_flip=True,
                                                                     rotation_range=30,
                                                                     height_shift_range=0.2,
                                      fill_mode='nearest')

VALIDATION_DIR = "Datasets 1-2\Validation"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
        TRAINING_DIR,
        target_size=(224,224),
        class_mode='categorical',
  batch_size = 64
)

validation_generator = validation_datagen.flow_from_directory(
        VALIDATION_DIR,
        target_size=(224,224),
        class_mode='categorical',
  batch_size= 16
)
```

```
Found 980 images belonging to 2 classes.
Found 239 images belonging to 2 classes.
```

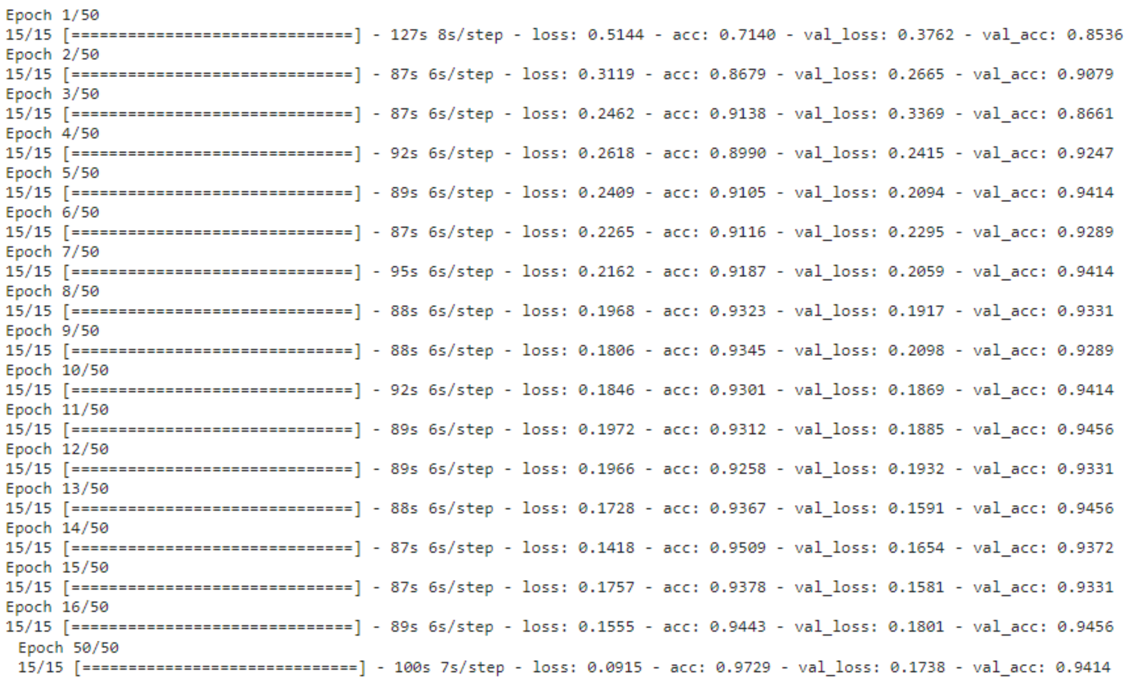
```
from tensorflow.keras.optimizers import RMSprop,Adam
model = tf.keras.models.Sequential([
          tf.keras.layers.Conv2D(96, (11,11), strides=(4,4), activation='relu', input_shape=(224, 224, 3)),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(256, (5,5), activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(384, (5,5), activation='relu'),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2048, activation='relu'),
          tf.keras.layers.Dropout(0.25),
          tf.keras.layers.Dense(1024, activation='relu'),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(2, activation='softmax')
])
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=0.0001),
              metrics=['acc'])
model.summary()
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 96)        34944

 max_pooling2d (MaxPooling2D  (None, 26, 26, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 22, 22, 256)       614656

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 6, 6, 384)         2457984

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 384)        0
 2D)

 flatten (Flatten)           (None, 1536)              0

 dropout (Dropout)           (None, 1536)              0

 dense (Dense)               (None, 2048)              3147776

 dropout_1 (Dropout)         (None, 2048)              0

 dense_1 (Dense)             (None, 1024)              2098176

 dropout_2 (Dropout)         (None, 1024)              0

 dense_2 (Dense)             (None, 2)                 2050

=================================================================
Total params: 8,355,586
Trainable params: 8,355,586
Non-trainable params: 0
_________________________________________________________________
```



```python
# class myCallback(tf.keras.callbacks.Callback):
#   def on_epoch_end(self, epoch, logs={}):
#     if(logs.get('val_acc')>=0.98):
#       print('\nReached ^98%')
#       self.model.stop_training = True
# callbacks = myCallback()

history = model.fit(
    train_generator,
    steps_per_epoch = 15,
    epochs = 50,
    validation_data = validation_generator,
    validation_steps = 15
    #callbacks=[callbacks]
)
```

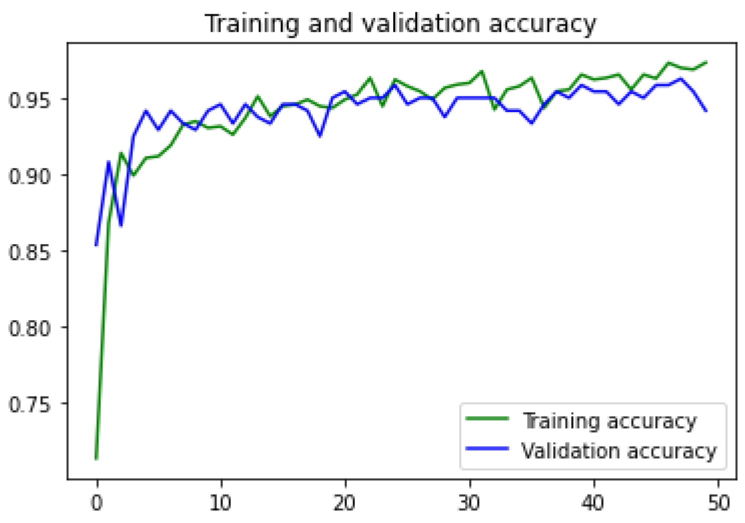


```
Epoch 1/50
15/15 [==============================] - 127s 8s/step - loss: 0.5144 - acc: 0.7140 - val_loss: 0.3762 - val_acc: 0.8536
Epoch 2/50
15/15 [==============================] - 87s 6s/step - loss: 0.3119 - acc: 0.8679 - val_loss: 0.2665 - val_acc: 0.9079
Epoch 3/50
15/15 [==============================] - 87s 6s/step - loss: 0.2462 - acc: 0.9138 - val_loss: 0.3369 - val_acc: 0.8661
Epoch 4/50
15/15 [==============================] - 92s 6s/step - loss: 0.2618 - acc: 0.8990 - val_loss: 0.2415 - val_acc: 0.9247
Epoch 5/50
15/15 [==============================] - 89s 6s/step - loss: 0.2409 - acc: 0.9105 - val_loss: 0.2094 - val_acc: 0.9414
Epoch 6/50
15/15 [==============================] - 87s 6s/step - loss: 0.2265 - acc: 0.9116 - val_loss: 0.2295 - val_acc: 0.9289
Epoch 7/50
15/15 [==============================] - 95s 6s/step - loss: 0.2162 - acc: 0.9187 - val_loss: 0.2059 - val_acc: 0.9414
Epoch 8/50
15/15 [==============================] - 88s 6s/step - loss: 0.1968 - acc: 0.9323 - val_loss: 0.1917 - val_acc: 0.9331
Epoch 9/50
15/15 [==============================] - 88s 6s/step - loss: 0.1806 - acc: 0.9345 - val_loss: 0.2098 - val_acc: 0.9289
Epoch 10/50
15/15 [==============================] - 92s 6s/step - loss: 0.1846 - acc: 0.9301 - val_loss: 0.1869 - val_acc: 0.9414
Epoch 11/50
15/15 [==============================] - 89s 6s/step - loss: 0.1972 - acc: 0.9312 - val_loss: 0.1885 - val_acc: 0.9456
Epoch 12/50
15/15 [==============================] - 89s 6s/step - loss: 0.1966 - acc: 0.9258 - val_loss: 0.1932 - val_acc: 0.9331
Epoch 13/50
15/15 [==============================] - 88s 6s/step - loss: 0.1728 - acc: 0.9367 - val_loss: 0.1591 - val_acc: 0.9456
Epoch 14/50
15/15 [==============================] - 87s 6s/step - loss: 0.1418 - acc: 0.9509 - val_loss: 0.1654 - val_acc: 0.9372
Epoch 15/50
15/15 [==============================] - 87s 6s/step - loss: 0.1757 - acc: 0.9378 - val_loss: 0.1581 - val_acc: 0.9331
Epoch 16/50
15/15 [==============================] - 89s 6s/step - loss: 0.1555 - acc: 0.9443 - val_loss: 0.1801 - val_acc: 0.9456
 Epoch 50/50
 15/15 [==============================] - 100s 7s/step - loss: 0.0915 - acc: 0.9729 - val_loss: 0.1738 - val_acc: 0.9414
```

```
model.save('firemodelfinal.h5')
```

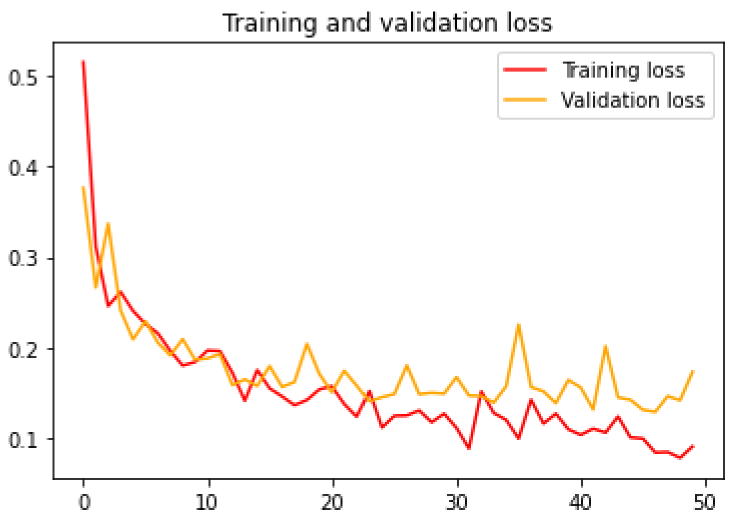
```
model=tf.keras.models.load_model('firemodel.h5')
history=model
```

```
%matplotlib inline
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'g', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()
plt.show()

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'orange', label='Validation loss')
plt.title('Training and validation loss')

plt.legend(loc=0)
plt.figure()
plt.show()
```



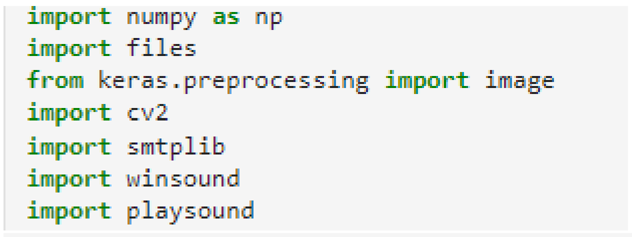
Training and validation accuracy
Training accuracy
Validation accuracy
0.95
0.90
0.85
0.80
0.75
0
10
20
30
40
50

```
<Figure size 432x288 with 0 Axes>
```

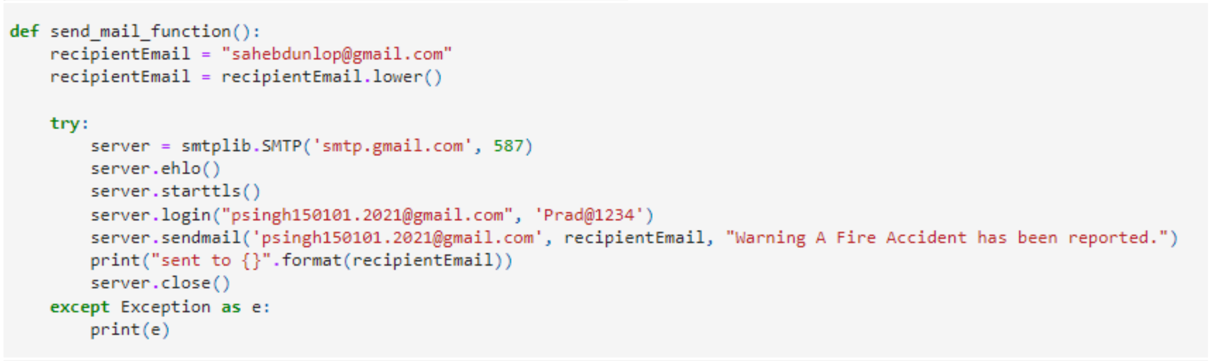


Training and validation loss
Training loss
Validation loss
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50

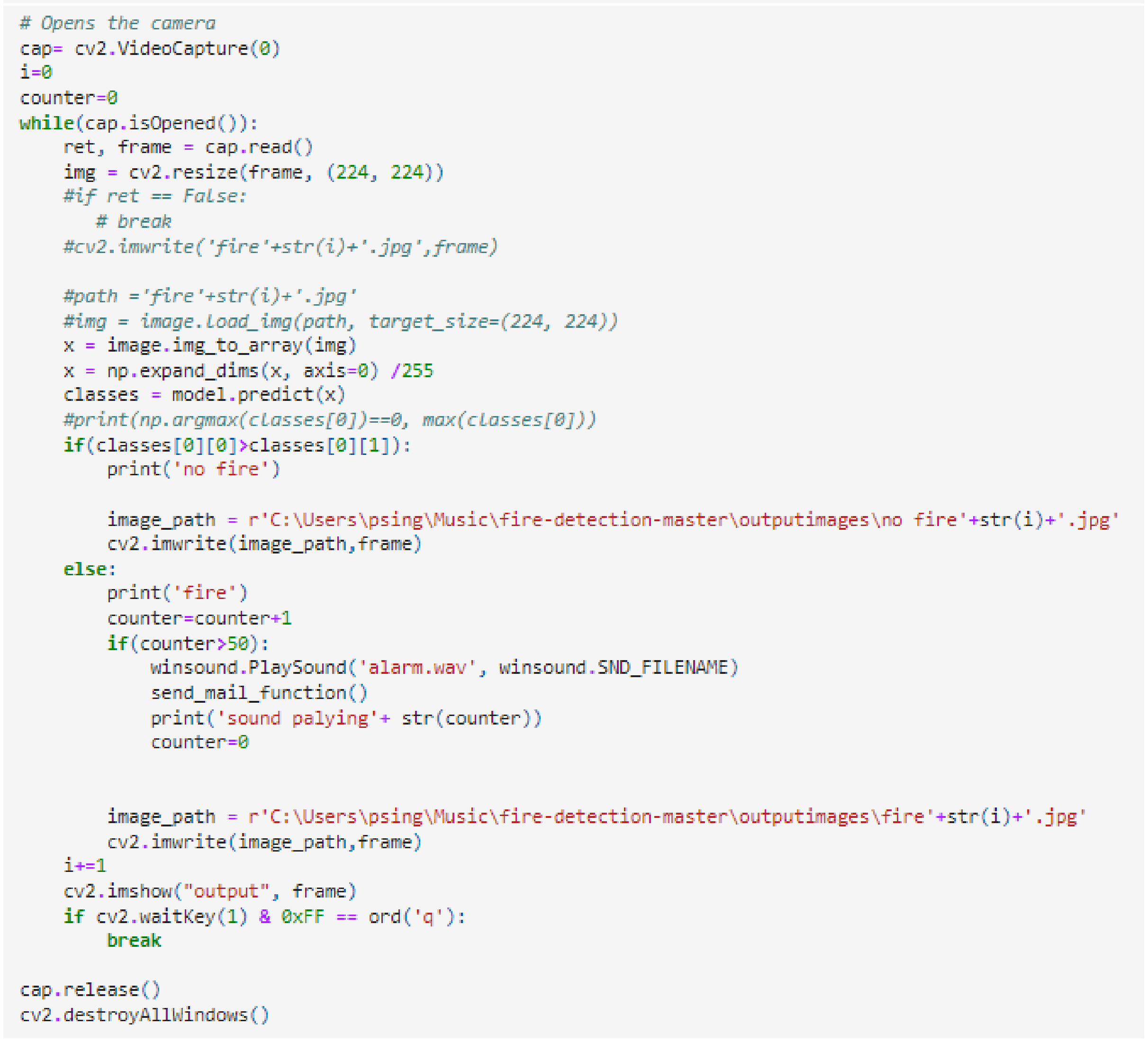
```
<Figure size 432x288 with 0 Axes>
```



```
import numpy as np
import files
from keras.preprocessing import image
import cv2
import smtplib
import winsound
import playsound
```



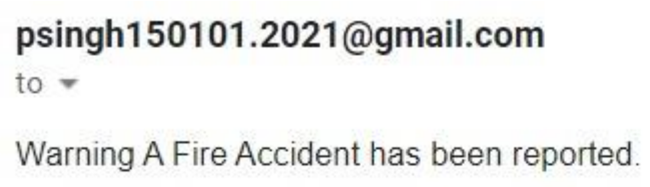
```
def send_mail_function():
    recipientEmail = "sahebdunlop@gmail.com"
    recipientEmail = recipientEmail.lower()

    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.ehlo()
        server.starttls()
        server.login("psingh150101.2021@gmail.com", 'Prad@1234')
        server.sendmail('psingh150101.2021@gmail.com', recipientEmail, "Warning A Fire Accident has been reported.")
        print("sent to {}".format(recipientEmail))
        server.close()
    except Exception as e:
        print(e)
```



```
# Opens the camera
cap= cv2.VideoCapture(0)
i=0
counter=0
while(cap.isOpened()):
    ret, frame = cap.read()
    img = cv2.resize(frame, (224, 224))
    #if ret == False:
       # break
    #cv2.imwrite('fire'+str(i)+'.jpg',frame)

    #path ='fire'+str(i)+'.jpg'
    #img = image.load_img(path, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0) /255
    classes = model.predict(x)
    #print(np.argmax(classes[0])==0, max(classes[0]))
    if(classes[0][0]>classes[0][1]):
        print('no fire')

        image_path = r'C:\Users\psing\Music\fire-detection-master\outputimages\no fire'+str(i)+'.jpg'
        cv2.imwrite(image_path,frame)
    else:
        print('fire')
        counter=counter+1
        if(counter>50):
            winsound.PlaySound('alarm.wav', winsound.SND_FILENAME)
            send_mail_function()
            print('sound palying'+ str(counter))
            counter=0


        image_path = r'C:\Users\psing\Music\fire-detection-master\outputimages\fire'+str(i)+'.jpg'
        cv2.imwrite(image_path,frame)
    i+=1
    cv2.imshow("output", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT

```
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
no fire
fire
fire
fire
fire
fire
fire
fire
fire
fire
fire
fire
```


psingh150101.2021@gmail.com
to
Warning A Fire Accident has been reported.

12. Refrences

Gomes, P., Santana, P., & Barata, J. (2014). A vision-based approach to fire detection. International Journal of Advanced Robotic Systems, 11(9), 149.

Arul, A., Prakaash, R. H., Raja, R. G., Nandhalal, V., & Kumar, N. S. (2021, May). Fire Detection System Using Machine Learning. In Journal of Physics: Conference Series (Vol. 1916, No. 1, p. 012209). IOP Publishing.

Abdel-Zaher, M., Hisham, M., Yousri, R., & Darweesh, M. S. (2021, July). Light-weight convolutional neural network for fire detection. In 2021 International Conference on Electronic Engineering (ICEEM) (pp. 1-5). IEEE.