THE RED ELEMENTAL

A project on Fire Detection and Localisation

GUIDE: **Dr Durga Prasad**

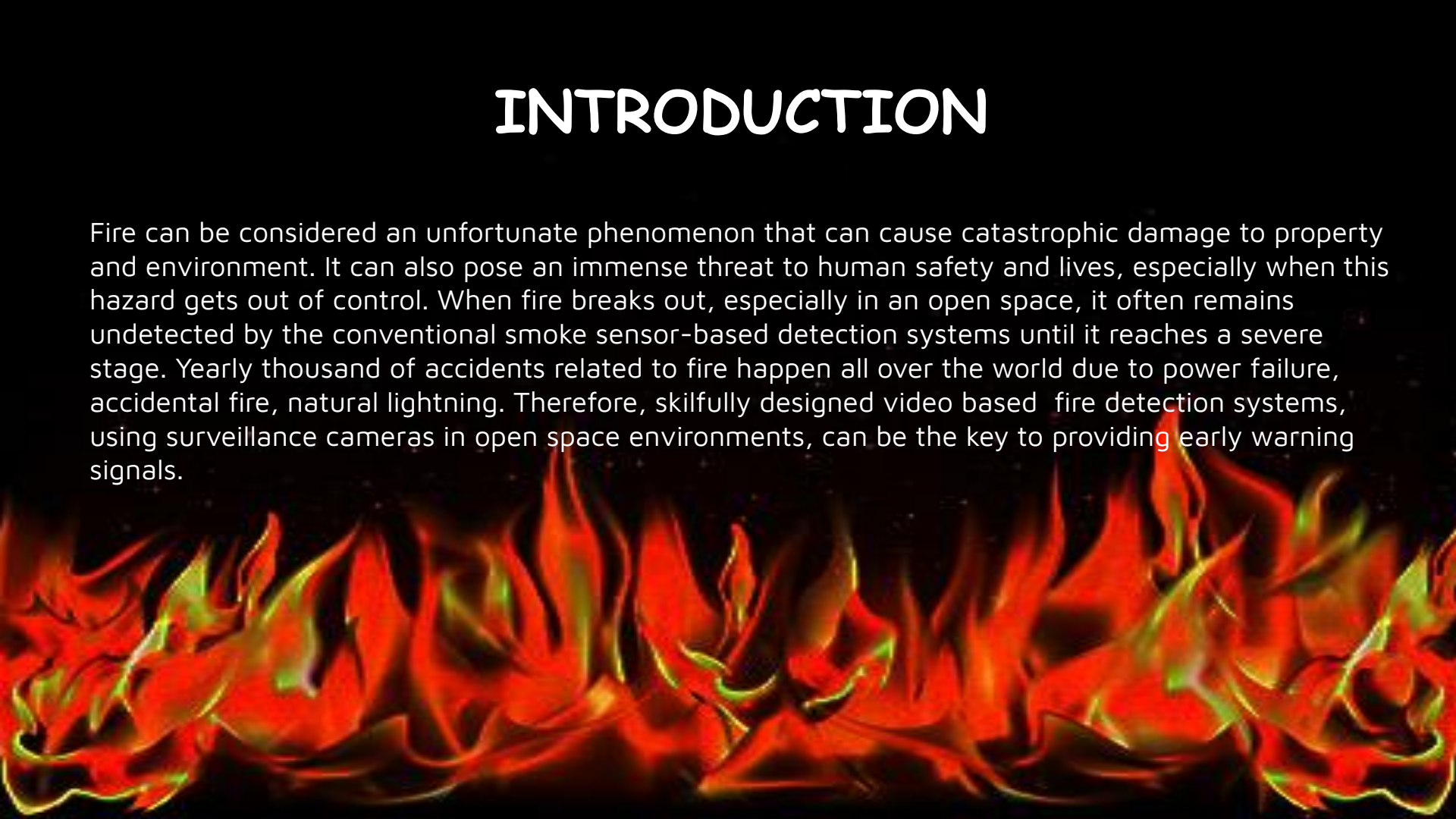
TEAM MEMBERS

19BAI10022 Ayushree Ghoshal
19BAI10025 Jigyasa Bisht
19BAI10054 Vanshika Sabhani
19BAI10077 Rashmi Rawat

Project Exhibition II

INTRODUCTION

Fire can be considered an unfortunate phenomenon that can cause catastrophic damage to property and environment. It can also pose an immense threat to human safety and lives, especially when this hazard gets out of control. When fire breaks out, especially in an open space, it often remains undetected by the conventional smoke sensor-based detection systems until it reaches a severe stage. Yearly thousand of accidents related to fire happen all over the world due to power failure, accidental fire, natural lightning. Therefore, skilfully designed video based fire detection systems, using surveillance cameras in open space environments, can be the key to providing early warning signals.



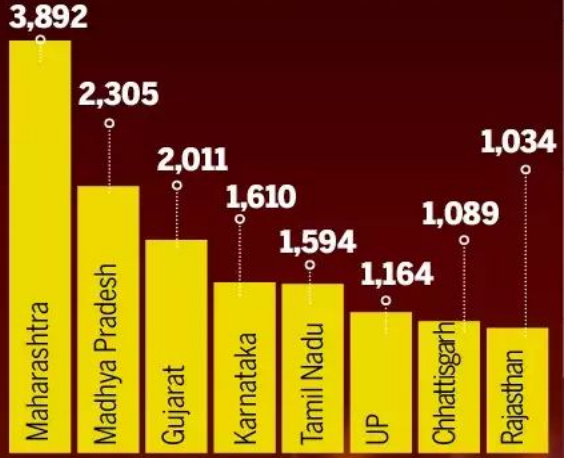
PROBLEM:

- Fire incidents are common across India due to the country's poor record for workplace safety standards and negligence in the enforcement of building regulations.
- **A total of 1,13,961 people lost their lives due to fire accidents from 2010 to 2014, which is an average of 62 deaths a day.**
- **India in 2015, 1,193 persons injured and 17,700 killed.**
- **Third position in the National Risk Survey Report.**
- **Australian wildfires of 2019-2020 were the deadliest forest fires.**
- **Half a billion animals killed, 20 million acres burnt, 100 million $ in damages.**
- Could be prevented by deploying fire detection systems but the existing system has lot of shortcomings.

1.13 lakh Indians killed in fire incidents from 2010 to 2014

YEAR: TOTAL DEATHS	
2010	24,414
2011	24,576
2012	23,281
2013	22,177
2014	19,513

STATE-WISE BREAK-UP FOR 2014



DEADLY FIRE INCIDENTS

2013 Kolkata market fire 19 dead, 19 injured

2012 Sivakasi factory explosion, TN | 54 dead, 78 injured

2012 Delhi-Chennai TN Express burns 35 dead, 25 injured

2006 Meerut 100 dead

2010 Kolkata 42 dead

2011 AMRI hospital, Kolkata 90 dead

2005 Bihar 35 dead, 50 injured

2004 Kumbakonam, TN 94 dead

2004 Srirangam, TN 50 dead, 40 injured

1997 Uphaar Cinema, Delhi 59 dead, 103 injured

2001 Erwadi, TN 28 dead

2002 Agra factory fire 40 dead

1997 Baripada, Odisha | 200 dead

1995 Dabwali, Haryana | 300 dead

AIM OF PROJECT

THE AIM OF OUR PROJECT IS **TO DETECT FIRE USING SURVEILLANCE CAMERAS AND PLAY AN ALARM.**
OUR PROJECT WILL ALSO SEND AN EMAIL TO THE ADMINISTRATOR.

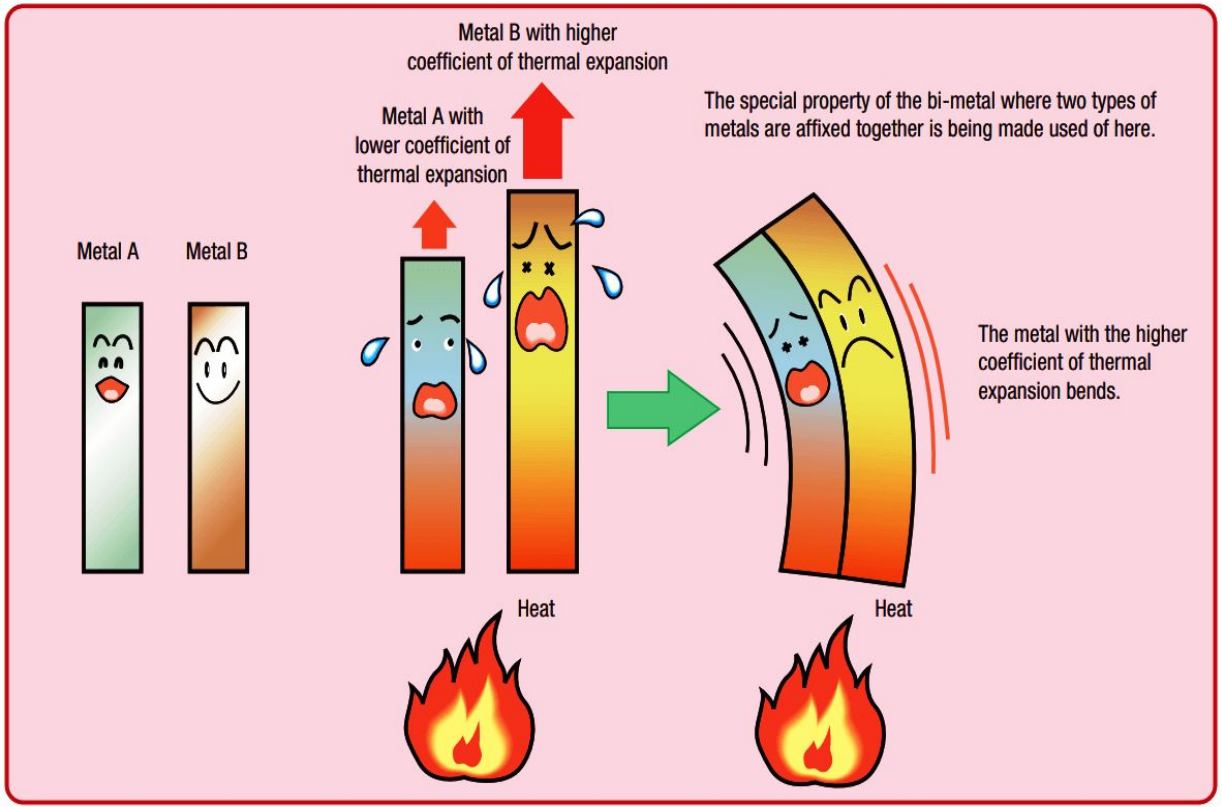
This camera-based fire monitoring system can monitor the specified area in real time through video processing. The project aimed to detect fire by using the image processing technology that will alert people by early detection of fire. When a fire is detected based on the video, it plays alarm and sends an email to the administrator, that's the working of our project.

Existing work with limitations

Heat detectors

- Do not sense particles of combustion.
- Designed to alarm only when heat on their sensors increase at a predetermined **rate** or reaches predetermined **level**.
- Heat detectors are designed to **protect** property, **not** life.
- **Blockage** of the heat flow to the detector due to objects.
- Heat detectors are generally **slower** to detect fires.
- Heat detectors **cannot** detect smolder fires which is leading cause of death in such accidents.

Mechanism of Spot Type Fixed Temperature Heat Detector

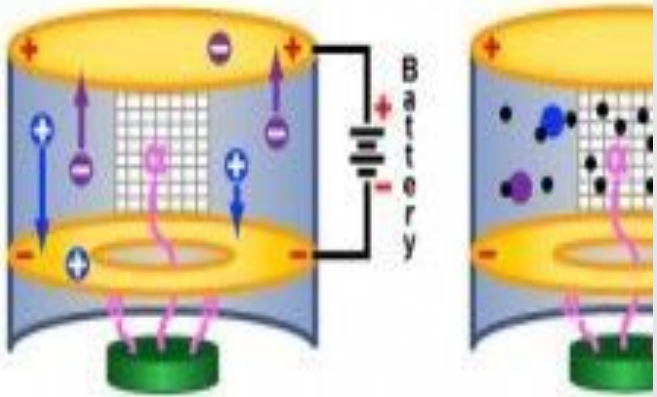


Existing work with limitations

Smoke detectors

- Are plagued by false alarms.
- Cannot be used in construction, cooking, steamy, humid, dusty, sanding environments.
- Cannot detect fires that release less smoke like LPG fires.
- A combination of smoke and heat detectors is required for effective detection.
- Requires separate infrastructure for setup which is costly.

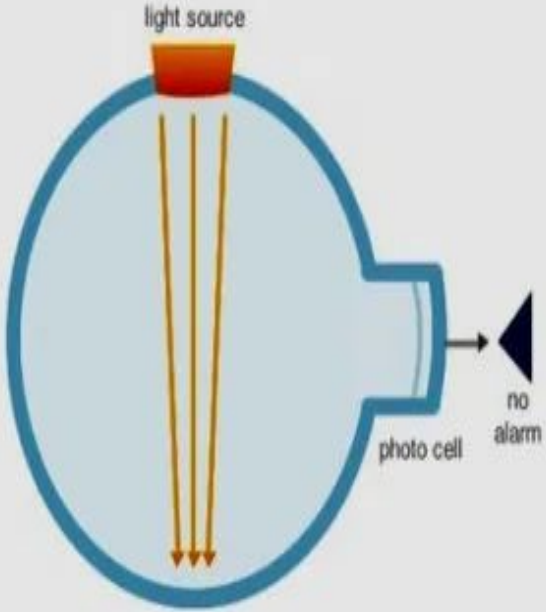
Ionization Smoke Alarm

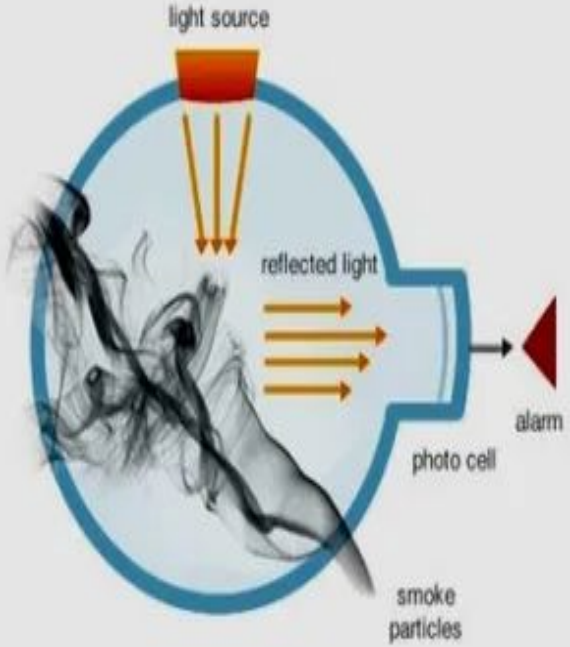


Alpha particles knock electrons free from the air molecules which then flow to the positve plate creating a small current

Smoke particles e
and attach to th
them neutral, disr
current, thus ini

Photoelectric Smoke Detector





Proposed work and methodology

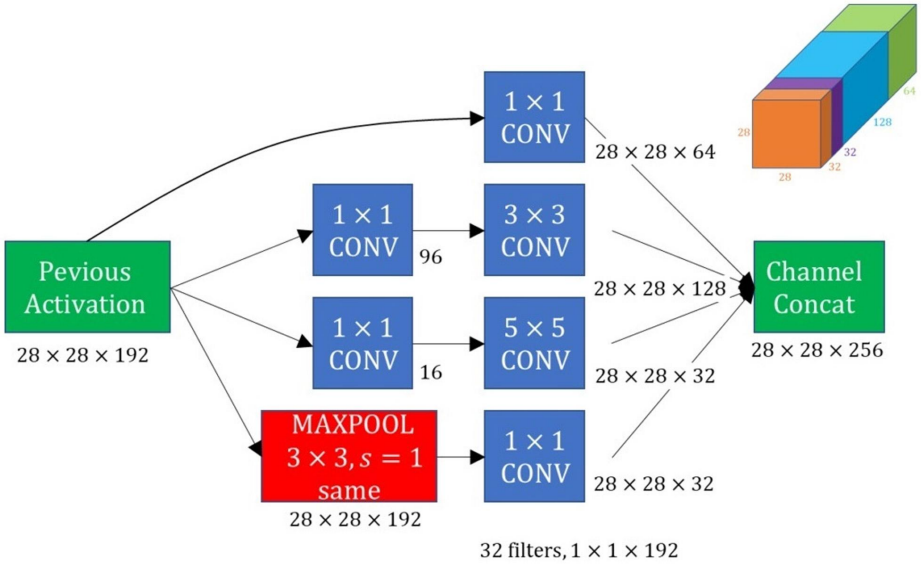
In this Deep Learning fire detection model, to make Implementation process easier and efficient we are planning using the concept of Transfer Learning (Re Use of pretrained model for our problem in simpler terms as it uses knowledge gain while solving problems and applying it to different related problems. The method is also known as representation learning that is used to train the machine learning classification model

Eg: Knowledge gained to detect car by the machine can also be used to train and detect truck with some additional features added to train the model. This transfer of knowledge that helps to identify truck from the car detection trained model is transfer knowledge.

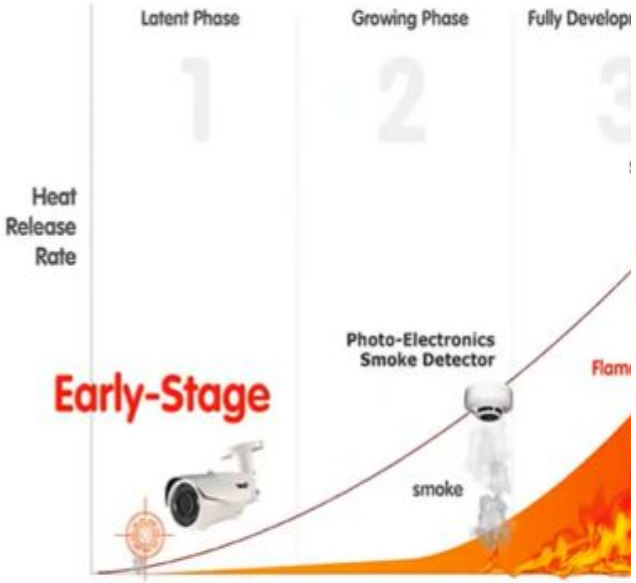
In the **initial state** we require to train our model and then divide it into frames for easy conversion of the training dataset to arrays. The **Training stage** requires preprocessing of the data. Lets know what

Pre processing

An **Inception Module** is an image model block that aims to approximate an optimal local sparse structure in a CNN. Put simply, it allows for us to use multiple types of filter size, instead of being restricted to a single filter size, in a single image block,



OLD



OUR MODEL

Training the model through various images & videos

Deep Learning

Pre Processing
Conversion of image in array
Resizing & coloring

ML Algorithms

Testing the efficiency of the system

The existing system lacked the monitoring of fire through camera.

Novelty

Our model confirms and adds :

- Accuracy , robustness, reliability
- NO manual interface
- NO dedicated Infrastructure and installments
- Easy to Build and Alter
- Requires few resources to train

The model detect fire using **visual inputs** and thus can be **easily integrated** with any surveillance infrastructure.

Real time usage

★ **Early Fire Detection and warning generation-**

The earlier the fire gets detected, it will play an alarm and will send an email to the administrator, the faster the firefighters will be informed and further,they'll work to stop it. Early detection can enable you to avoid serious damage or destruction,so it is of extreme importance.

★ Placement Flexibility-

A camera can be placed just about anywhere we want.

★ **24/7 Monitoring-** It provide constant protection no matter where we are. This advantage exemplify the many uses of this system.

- **Easy & Affordable-**Once installed, fire detectors are easy to use. They automatically sense hazardous conditions without our doing anything.
- **Low Maintenance-** Despite their importance, this system require little maintenance.
- **Decreased Risk of Fire Damage-** Property damage can require large investments and a great deal of time to repair. We can also avoid damage to nearby properties when we install this system. Prevention is possible because a fire alarm can enable firefighters to respond and put out the fire before it gets out of control.

Hardware Requirements

Recommended hardware specifications are:

- CPU - Intel Core i7 7th Generation
- GPU - NVIDIA GeForce GTX 940 or higher.
- RAM - 16 GB memory and up for some heavy lifting and multi tasking
- Stable internet connection (minimum 5mbps download)
- Operating System
  - Windows 10 Home, Pro
  - Linux distros, Ubuntu and its derivatives

Software Requirements

The application is based upon the python language, which is high-level programming language easy to use, and with the python, the system uses the openCV library for all the image processing systems.

.

Python

Python is considered as the general-purpose, high-level programming language. With the python, the readability of code is enhanced, compared to another language like C, C++, java.

OpenCV

OpenCV is an open-source package and machine learning library, which is designed for real-time computer vision applications. OpenCV is the cross-platform library that supports many programming languages like Python, Java, C++, C, etc.

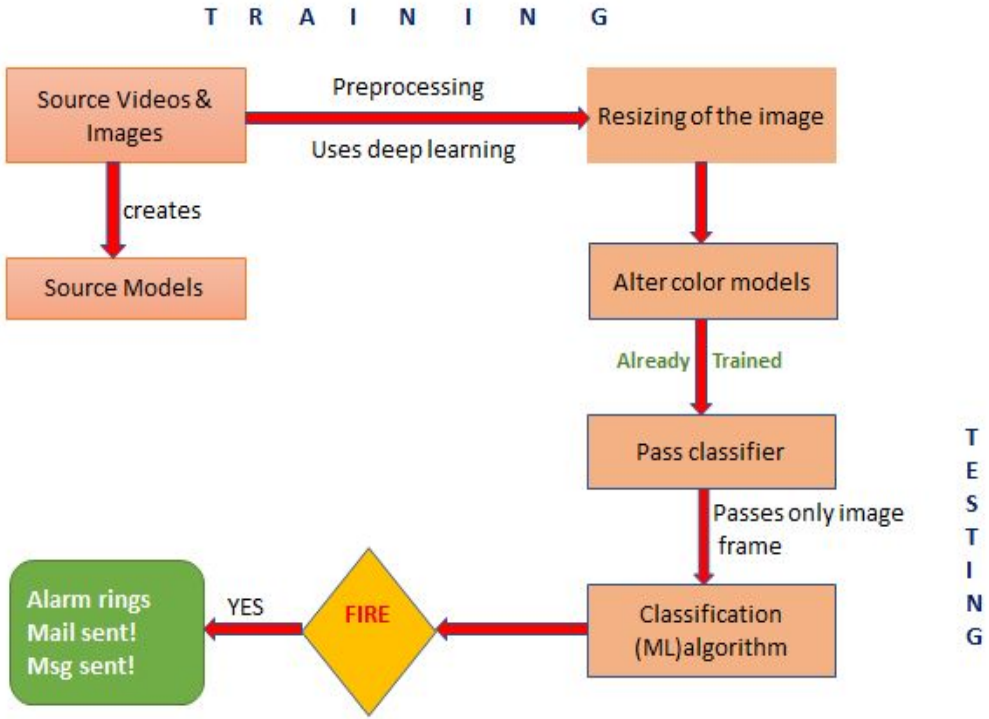
TensorFlow

1. ***Easy model building***: TensorFlow offers multiple levels of abstraction so that we can choose the right one for our needs..

2. ***Robust ML production anywhere*:** TensorFlow has always provided a direct path to production.

3. ***Powerful experimentation for research***: TensorFlow gives us the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies.

Overall System Architecture



First Review

Literature Review

- T. H. Chen in his work "**An early fire-detection method based on image processing**" examined the dynamic behavior of fires using RGB and HSI color models and proposed a decision rule-assisted fire detection approach, which uses the irregular properties of fire for detection. Their approach is based on frame-to-frame differences, and hence cannot distinguish between fire and fire-colored moving regions
- G. Marbach works "**An image processing technique for fire detection in video images**" investigated the YUV color space using motion information to classify pixels into fire and non-fire components.
- B. U. Töreyin project "**Computer vision based method for real-time fire and flame detection**" used temporal and spatial wavelet analysis to determine fire and non-fire regions. Their approach uses many heuristic thresholds, which greatly restricts its real world implementation.
- D. Han and B. Lee, "**Development of early tunnel fire detection algorithm using the image processing**" compared normal frames with their color information for tunnel fire detection; this method is suitable only for static fires, as it is based on numerous parameters.
- T. Çelik and H. Demirel, "**Fire detection in video sequences using a generic color mode**" explored the YCbCr color space and presented a pixel classification method for flames.
- P. V. K. Borges and E. Izquierdo, "**A probabilistic approach for vision based fire detection in videos**"utilized the low-level features including color, skewness, and roughness in combination with a Bayes classifier for fire recognition.

After many researches and study going on the flame detection and smoke detection, some algorithms are specially designed for this purpose. These algorithms designed till now on fire detection through videos are, statistical color model, spatioTemporal Flame Modelling and Dynamic Texture Analysis and now a day's optical mass flow estimators is getting attention. The Scientists Kosmas Dimitropoulos, Panagiotis Barmpoutis and Nikos Grammalid are focuses on different modelling algorithms used in flame detection.

Large numbers of techniques have been developed for the fire detection due to the number of vision based algorithms proposed in various literature surveys. Conventional methods of fire detection have been practically replaced with Video-based smoke detection methods due to various advantages over conventional methods like such as early fire detection, speedy response, non-presence of spatial limits, also information of the fire progress can be achieved due to live video and in fire investigation vision based is capable to provide forensic evidences.

Module description

LIBRARIES REQUIRED: In this module we see the inclusion and significance of various neural network open-source models and libraries used.

RAW DATA PREPROCESSING: Here we download a dataset and perform filtering and cleaning of images

CREATION OF THE CNN ARCHITECTURE: The model is trained using MaxPooling, Convolution layers and the results are then mapped and validated.

CREATION OF INCEPTION V3 MODEL: This model is created to train the dataset to analyse complex features and increase the accuracy to minimize the overfitting.

OUTPUT GENERATION: Analysis of the testing dataset along with real time testing.

MODULE 1: LIBRARIES REQUIRED

```
import tensorflow as tf
```

Tensor flow is an open-source library for deep learning applications.It supports traditional machine learning and huge numeric computations thus open sourced by google. Data is fed in the form of multi dimensional array of high dimensions (TENSOR) as they can handle a huge dataset. As the mechanism of execution is mostly graphical it is much easier to execute across a cluster of computers by using GPUs.

```
import keras_preprocessing
```

```
from keras_preprocessing import image
```

```
from keras_preprocessing.image import ImageDataGenerator
```

The Keras libraries are used for data pre-processing and data augmentation. The libraries help is to work with image data and resizing of the imported images of the dataset.

```
import shutil
```

This is used for high level operation on files and for removal and copy of file and directories.

MODULE 2: RAW DATA PREPROCESSING

Datasets containing the images of fire and non-fire are downloaded and fed into the model for preprocessing

We have used the ImageDataGenerator for labeling the dataset for training. We need to ensure same size, height and rotation to enable the machine easy and hassle-free learning. We apply 3 data augmentation techniques — horizontal flipping, rotation, and height shifting.

Finally, we have 980 images for training and 239 images for validation of the function to be accurate.

```
TRAINING_DIR = "/content/drive/MyDrive/fire-detection-master/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 = "/content/drive/MyDrive/fire-detection-master/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)
```

MODULE 3: CREATION OF THE CNN ARCHITECTURE

We use an optimizer to help us to change the attributes and adjust the weights to improve the learning rate of the model. We use ADAMs optimizer instead of the classical gradient descent to update weights of the training data. ADAM has the perks of AdaGrad and RMSProp helping it to maintain the learning rate of each weight unlike the classical method where the learning rate is not stored during training set.

We have three Conv2D, MaxPooling2D and Dense layers. To avoid overfitting Dropout layers are also added. At the last layer give the Probability distribution for both fire and non-fire.

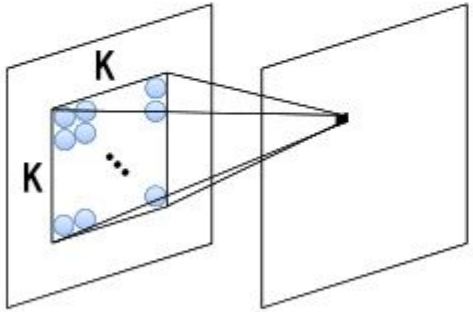
```
from tensorflow.keras.optimizers import RMSprop,Adam
model = tf.keras.models.Sequential([
          tf.keras.layers.Conv2D(96, (11,11), strides=(4,4), activation='relu', input_shape=(224, 224, 3)),
          tf.keras.layers.MaxPooling2D(pool_size = (3,3), strides=(2,2)),
          tf.keras.layers.Conv2D(256, (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(lr=0.0001), metrics=['acc'])
model.summary()
```

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 (MaxPooling2	(None, 10, 10, 256)	0
conv2d_2 (Conv2D)	(None, 6, 6, 384)	2457984
max_pooling2d_2 (MaxPooling2	(None, 2, 2, 384)	0
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

Convolution Layer

The convolution neural network enables us to work with two-dimensional image data. It can be used for one as well as three-dimensional data as well.

Centre of the CNN we have the convolution layer that deals with the multiplication of 2d array weight (filter /kernel) to the input data. The filter is smaller than the input layer and is basically the dot product of 2d weight array to the input data. This helps us to detect a special feature in the input images. This is known as translation invariance i.e., can check whether feature is present rather than where is it present.



$$z_{ij} = y_{ij} + \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} x_{i+m,j+n} w_{mn}$$

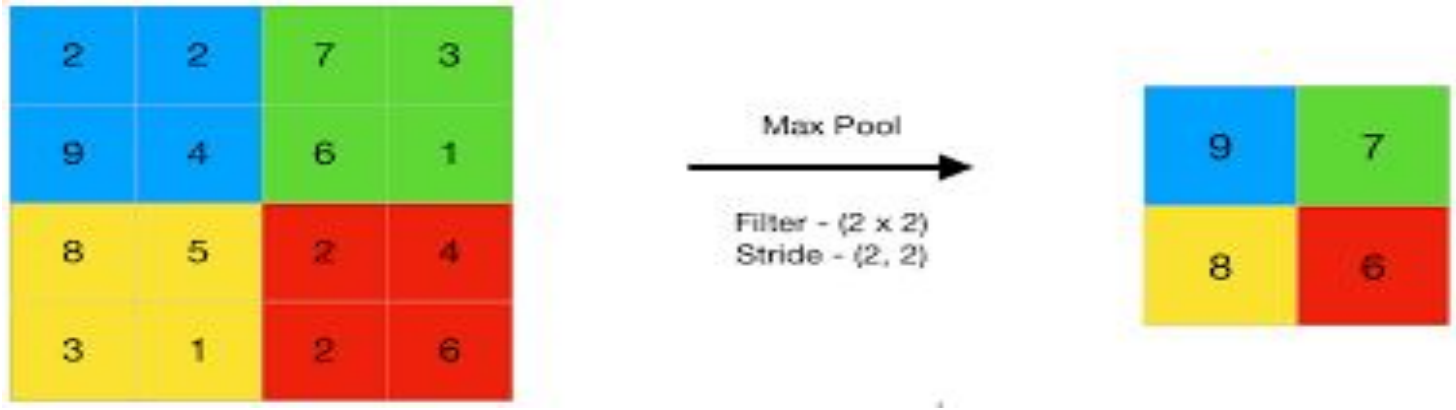
x_{ij}: Input signal
y_{ij} : Bias
w_{mn}: Weight
K: Kernel size
z_{ij}: Output signal

Max Pooling Layers

Pooling layers basically summarise as whether the feature is present in the feature map created by the convolution layer. They are of two types.

Average pooling and MaxPooling2D layer.

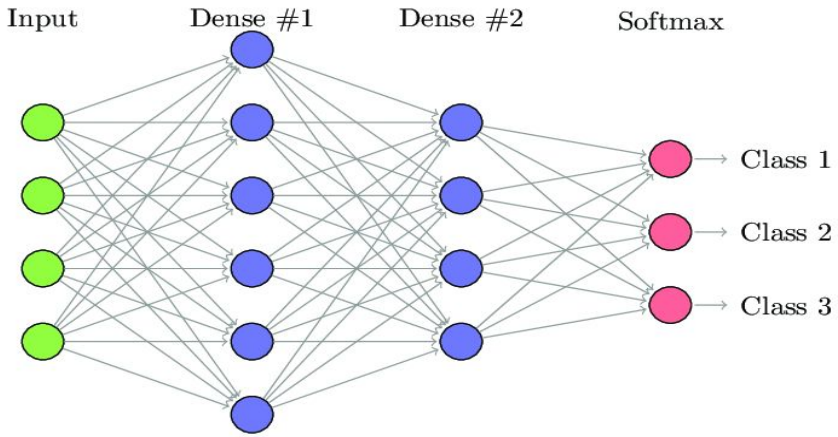
They tell us about the presence of a feature or the most dominant feature. The MaxPooling2D works on finding the maximum value of every patch of the feature map and then the combined result highlights the most dominant feature and presence of a feature. It is better than average pooling for image classification tasks.

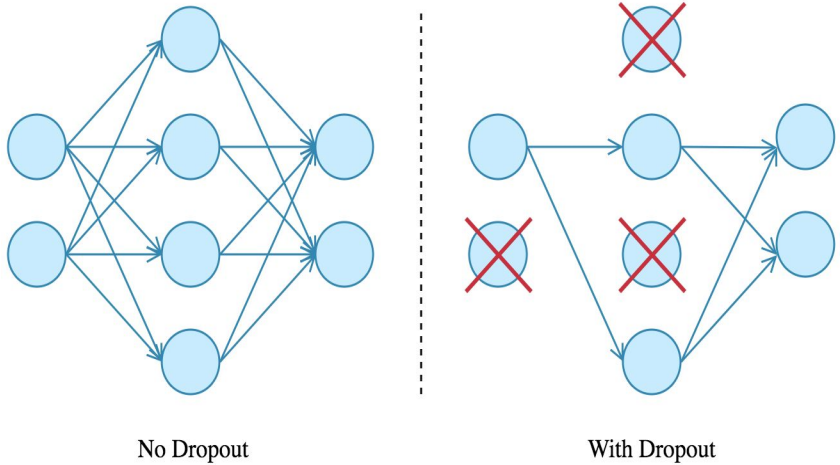


Dense Layer and Dropout Layer

Dense Layers are strongly connected neural network that means all neurons are connected to others neurons in next layer. It helps us to learn features from the previous neurons and so on. The bias value (for optimisation) is added to the dot product of input and kernel values.

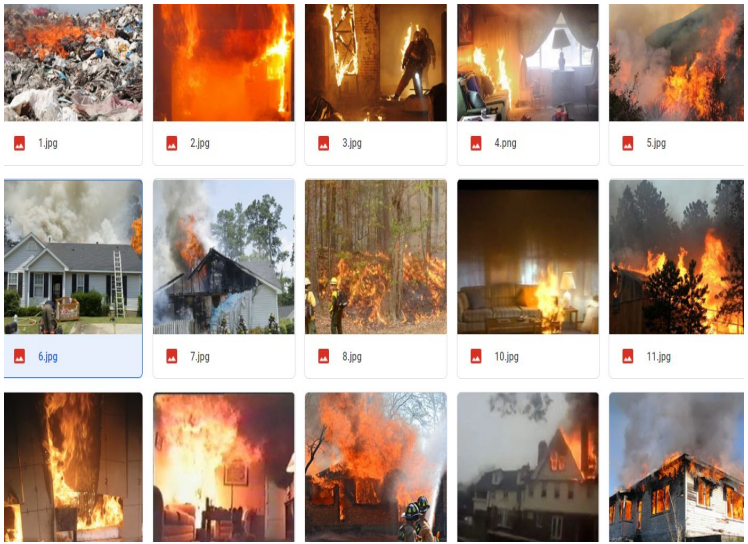
All these might cause a case of overfitting (high variance) thus we now require the use of dropout layers. They are used to drop the hidden layer or the network layer which is not required in the dataset and causing problem of overfitting.

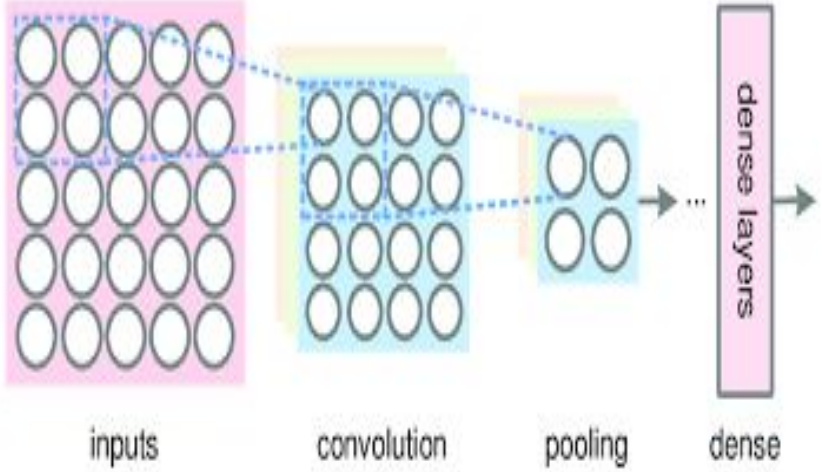




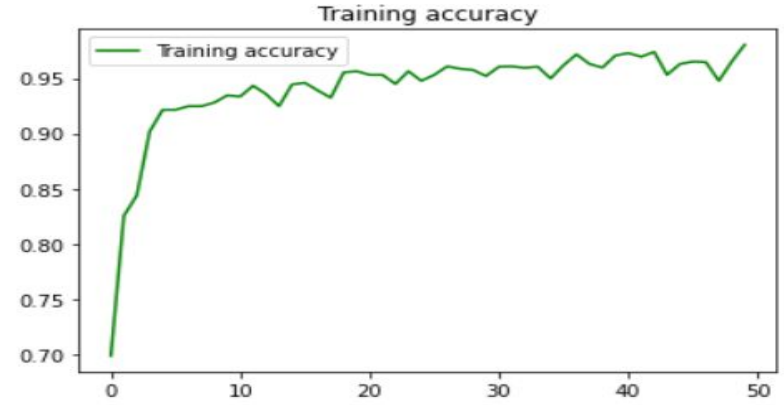
Snapshots of the Working of Modules

Dataset containing images are preprocessed and created in form of arrays

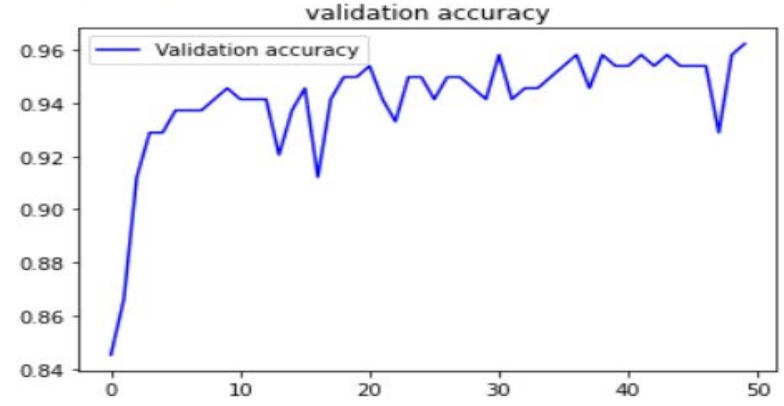




Graphical representation of the training and Validation Set



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

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.

Testing of the Module

For testing on any random image we use the code

```
uploaded = files.upload()
for fn in uploaded.keys():
  path = '/content/' + fn
  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]))
```

True 0.876

false

On testing it on any other random image we get output

The output as follows:

In this picture the output is true but accuracy is 0.756517

This machine is not sure so we need analysis of

more complex features.

Inception V3 Model

As we see that the accuracy of the previous model was around 75% thus to increase the image accuracy we decided to include the Inception V3 Model. This model is widely used as it is attains 78.1% accuracy on ImageNet dataset. As we have incorporated the dataset from image net thus this model suit us.

WORKING : get the dataset of training and validation

Import the InceptionV3 model and all the layers (AveragePooling2D, Input, Dropout)

Creating customized InceptionV3 model

```
[ ]  TRAINING_DIR = '/content/drive/MyDrive/Dataset 3/Train'

     training_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.15, horizontal_flip=True, fill_mode='nearest')

     VALIDATION_DIR = '/content/drive/MyDrive/Dataset 3/Test'
     validation_datagen = ImageDataGenerator(rescale = 1./255)

     train_generator = training_datagen.flow_from_directory(TRAINING_DIR, target_size=(224,224), shuffle = True, class_mode='categorical',  batch_size = 128)

     validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR, target_size=(224,224), class_mode='categorical', shuffle = True, batch_size= 14)
```

```
Found 1801 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
```

```
[ ]  from tensorflow.keras.applications.inception_v3 import InceptionV3
     from tensorflow.keras.preprocessing import image
     from tensorflow.keras.models import Model
     from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Input, Dropout
```

```
input_tensor = Input(shape=(224, 224, 3))

base_model = InceptionV3(input_tensor=input_tensor, weights='imagenet', include_top=False)

# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# add a fully-connected layer
x = Dense(2048, activation='relu')(x)
x = Dropout(0.25)(x)
# output layer
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(2, activation='softmax')(x)
# model
model = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])
```

Shape - size of image matrices and rgb layer

Weights

Include_top

Relu Activation

Softmax Activation

RmsProp Optimizer

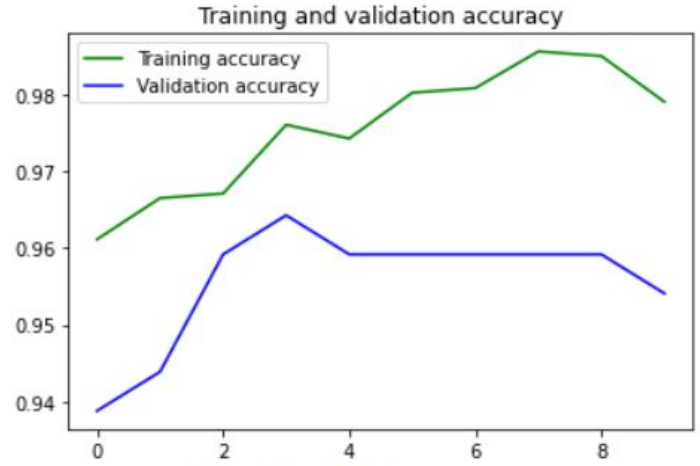
Categorical_Crossentropy Loss

Metrics

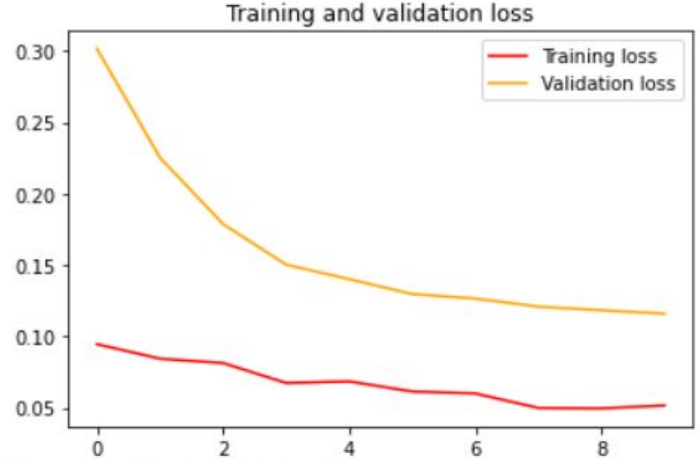
```
class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('val_loss')<0.1100 and logs.get('loss')<0.1100):
      print('\n\n Reached The Destination!')
      self.model.stop_training = True
callbacks = myCallback()

history = model.fit(
    train_generator,steps_per_epoch = 14,epochs = 20,validation_data = validation_generator, validation_steps = 14)
```

A callback is a powerful tool to customize the behavior of a Keras model during training, evaluation, or inference.

[ ]



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

The graph below shows the training and validation set accuracy graph along with the loss occurred in the epoch cycle.

We see that Training accuracy has reached the level of 0.98

Thus we see that the Inception model has yielded desired results and trained well.

Validating accuracy has reached the level of 0.96

```
for layer in model.layers[:249]:
   layer.trainable = False
for layer in model.layers[249:]:
   layer.trainable = True

from tensorflow.keras.optimizers import SGD
model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['acc'])


class myCallback(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('val_loss')<=0.1099 and logs.get('loss')<=0.1099):
      print('\n\n Reached The Destination!')
      self.model.stop_training = True
callbacks = myCallback()

history = model.fit(train_generator, steps_per_epoch = 14, epochs = 10, validation_data = validation_generator, validation_steps = 14)
print(len(base_model.layers))
```

to train the top 2 inception blocks, freeze the first 249 layers and unfreeze the test,then recompiling the model or these three modifications to take effect.

We have used SGD optimizer because it is much faster and takes less no. of epochs.

we train our model again (this time fine-tuning the top 2 inception blocks alongside the top Dense layers

Testing of the Module after Inception V3 model

For testing on any random image we use the code

```
uploaded = files.upload()
for fn in uploaded.keys():
  path = '/content/' + fn
  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]))
```

true 0.956781

false

On testing it on any other random image we get output

The output as follows:

In this picture the output is true and accuracy is **0.966517**

The need of inception model is fulfilled.

Alarm and Email

```
def alert():
    mixer.init()
    alert = mixer.Sound('/content/WhatsApp Audio 2021-05-03 at 11.40.17 PM.mpeg')
    alert.play()
    time.sleep(0.1)
    alert.play()
```

As soon as the value comes out to be true the next step an alarm is generated and that is followed by an email from the sender to receiver. Thus our system is able to predict fire and generate warning .

Now for future work could include the generation of a device that would be able to track the events and warn us of fire.

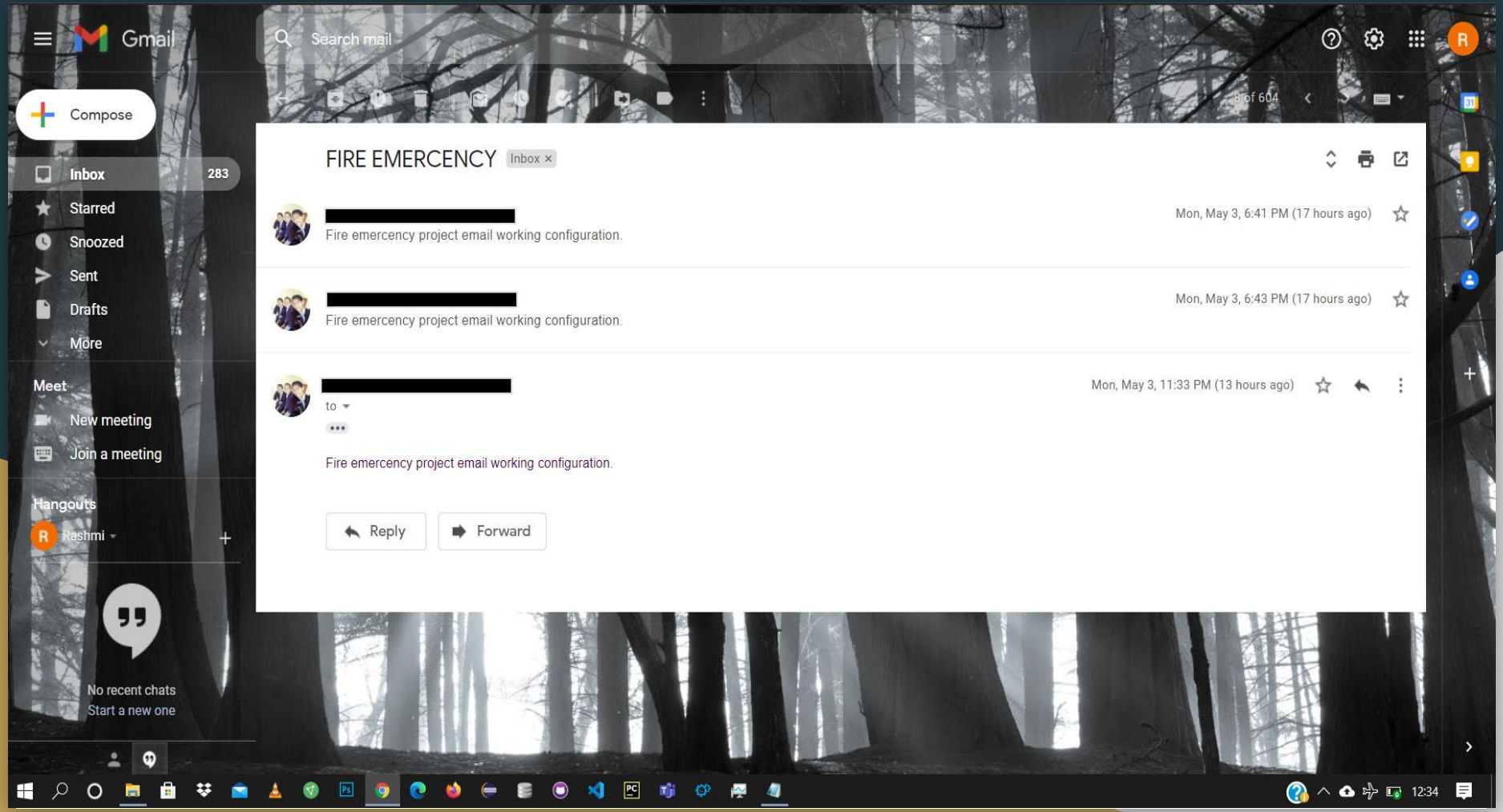
E mail

```
[ ]  import smtplib
     from email.mime.image import MIMEImage
     from email.mime.text import MIMEText
     from email.mime.multipart import MIMEMultipart
     import datetime
```

```
[ ]  import ssl
     print(ssl.OPENSSL_VERSION)
```

```
     OpenSSL 1.1.1  11 Sep 2018
```

```
[1]  if (flag ==1):
       port = 587  # For starttls
       smtp_server = "smtp.gmail.com"
       sender_email = input("Type sender email press enter:")
       receiver_email = input("Type receiver email press enter:")
       password = input("Type your password and press enter:")
       message = """\
       Subject: FIRE EMERCENCY

       Fire emercency project email working configuration."""

       context = ssl.create_default_context()
       with smtplib.SMTP(smtp_server, port) as server:
            server.ehlo()  # Can be omitted
            server.starttls(context=context)
            server.ehlo()  # Can be omitted
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message)
            print("Mail Sent")
```


Gmail
Search mail
Compose
Inbox 283
Starred
Snoozed
Sent
Drafts
More
Meet
New meeting
Join a meeting
Hangouts
Rashmi
No recent chats
Start a new one
8 of 604
FIRE EMERCENCY
Inbox ×
[redacted]
Fire emercency project email working configuration.
Mon, May 3, 6:41 PM (17 hours ago)
[redacted]
Fire emercency project email working configuration.
Mon, May 3, 6:43 PM (17 hours ago)
[redacted]
Mon, May 3, 11:33 PM (13 hours ago)
to
Fire emercency project email working configuration.
Reply
Forward
12:34

THANK YOU !