JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Real Time, Low Latency Fire Detection And Alert System

[1]Prof. Shiburaj Pappu,[2]Siddhant Poojary,[3]Robin Singh

[1]Shiburaj@eng.rizvi.edu.in,[2]sidrizvi@eng.rizvi.edu.in,[3] Rjjaatsingh@eng.rizvi.edu.in

Department Of Computer Engineering,

Rizvi College Of Engineering, Mumbai , India - 40050

Abstract : Image Processing is a form of processing media with the input of video or image and transforming the input into output with certain technique. In this project, we will create a real time fire detection system with the help of camera using machine learning, computer vision and apply the concepts of image processing and python automation in order to get the real-time detection results via mobile notification techniques. If there is a fire detected by the camera, then the footage will be processed using image processing and then the resultant data will be forwarded to the concerned personnel.

Fire detection with low –latency using color information has many applications in computer vision and other domains. The light parameter and the color of the flame helps in detecting fire. Our model has many advantages over traditional methods of fire detection, such as simplicity, practicality and cost efficient.

***IndexTerms* – Fire Detection, Computer Vision, Machine Learning, Image Processing, Deep Learning, Automation Techniques.**

I. INTRODUCTION

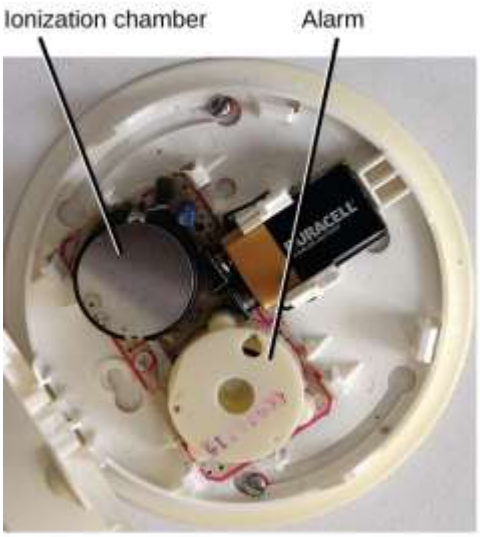
The purpose of this project is to develop a low cost and low maintenance fire detection system which uses cameras to detect fire using machine learning/ computer vision in real-time and notify officials via mobile notification techniques. Initially, we focused on innovating and developing algorithms to reduce the use of sensor based systems such as photoelectric/ionization detectors that are currently used in the markets. We realized that existing systems are more prone to false alarms because of being insensitive to smoldering fires. We also explored other detectors, but they were more expensive and required more current to operate. Furthermore, we were interested in developing a system that is low in cost, requires minimal maintenance, and can detect fire with low latency. As a result, we chose to work on machine learning and computer vision based algorithms that would make our solution cheap and low in maintenance, as we don't need expensive detectors.

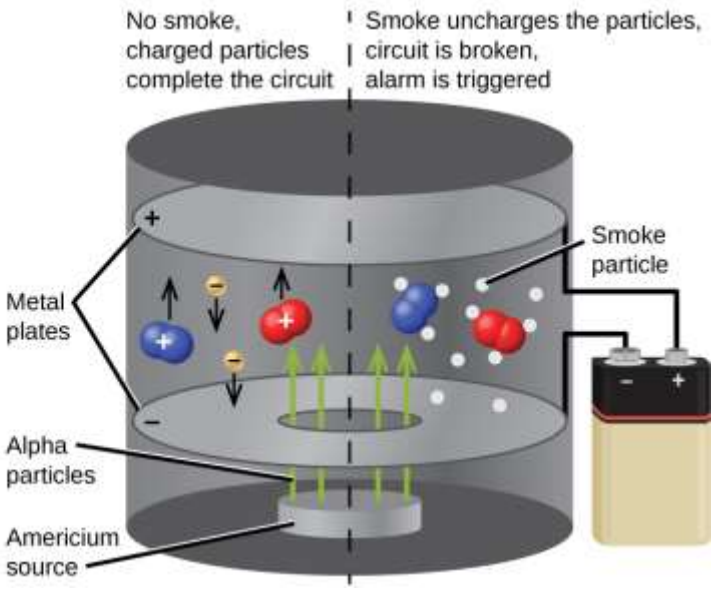
II. EXISTING METHODS

Currently, most fire detection systems rely on built-in sensors, which are primarily dependent of their reliability and spatial dispersion. The photoelectric detector and the ionization detector are the two most common types of fire detection systems used in markets such as schools, universities, offices, stadiums etc. These are typically installed at floor (or ceiling) level heights. These contain individual sensors that are not lined up, resulting in unpredictability and asynchronous alarm behavior and actions.

- Ionization Smoke Alarm

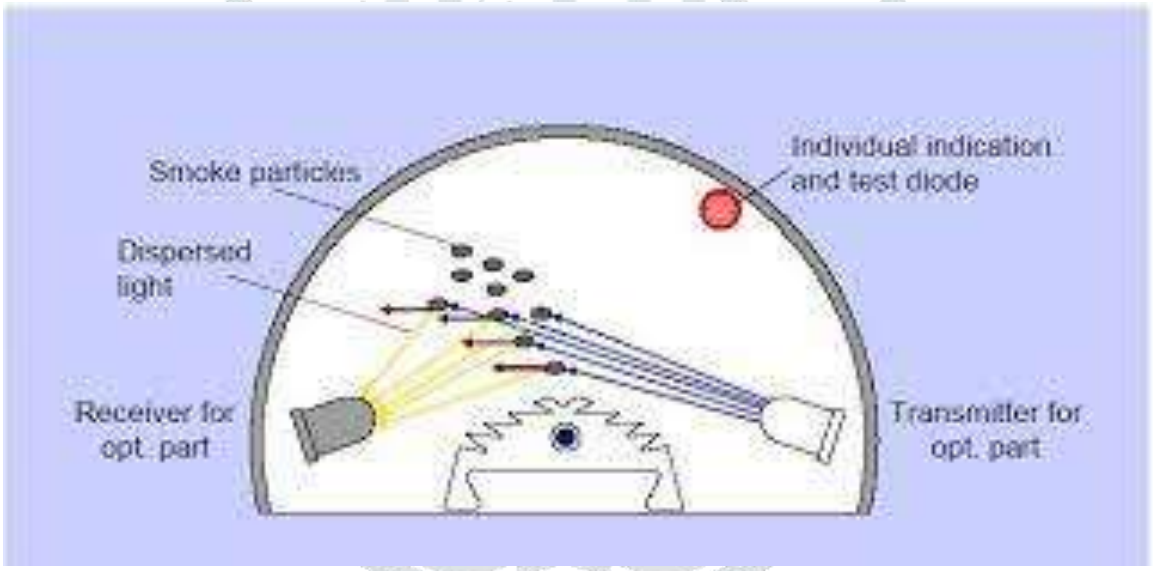
  An alpha particle producing a radioactive source (americium -241), a smoke chamber, and charged detector plates comprise the ionization smoke detector. Because of the alpha source, the air within the smoke chamber becomes ionized and conductive. As Smoke particles enter the smoke chamber, they cling to the ionized air molecules, making the air in the chamber less conductive. The alarm is triggered when the air conductivity within the chamber falls below a predetermined level.





❖ Photoelectric Smoke Alarm

These detectors are typically more sensitive to smoldering fires – that is, fires that begin with a long period of smoldering. A photoelectric sensor and a light source are used to power photoelectric alarms. As smoke enters the chamber and crosses the path of the light beam, the smoke particles scatter the light, directing it toward the sensor and triggering the alarm.



III. Limitations Of Existing System

❖ Cost And Maintenance

The main limitation of current systems is that they are expensive to build and maintain. As a result, most people prefer to avoid such systems for their storage and working environments. Inadequate maintenance is the most common cause of a fire system not working. As a result, a fire system should be tested once a week to ensure that all sensors and transmitters are operational. Despite being built to last a long time, these systems can fail at any time.

❖ Delay in detection

In the event of a fire, speed is the most important factor in determining whether lives and property can be saved in a timely manner. Current fire detection systems rely on smoke reaching the sensors to send out a response. Such a waste of valuable time can result in a fire that has already gotten out of hand and is destroying property on a large scale.

❖ Large are coverage

For a high precision fire detection system , these sensors must be disturbed densely. Due to the requirement of a regular distribution of sensors in close proximity, coverage of large areas in a sensor-based fire detection system for an outdoor environment is impractical.

❖ Fire location

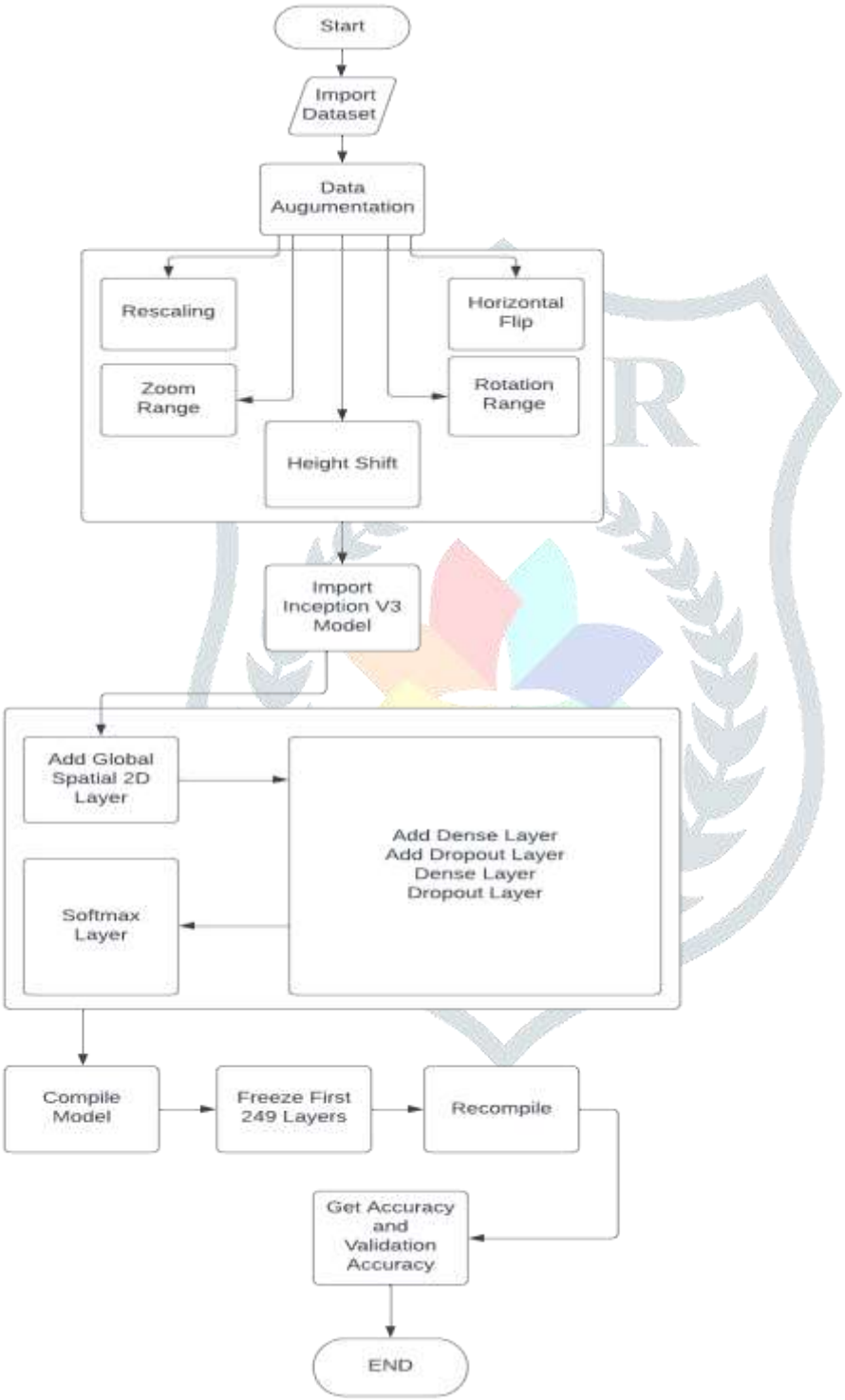
Along with fire detection, one should be able to pinpoint the precise location of the fire. If there is a large crowd in a large area(Hall, Stadium etc) and a fire breaks out at one end of the area, it should be of great interest that the officials are alerted of the location of the fire, which can be of great help to them in sending out the crowd from the other end of the place, ensuring no one by mistake goes near the place of fire.

- Can be easily duped
  Because an ion detector looks for small combustible particles in the air, it can be fooled by chemical or paint particles in the air. Dust, steam , and even spider webs can fool the photoelectric detector, which needs to "see" the fire. Despite the fact that both offer protection against undetected fires, ion detectors have a higher incidence of nuisance alarms.

IV. THE PROPOSED FRAMEWORK

We propose a system which automatically detects the presence of fire based on the algorithm described subsequently.

- Model Flowchart



- Webcam
  A webcam is installed at locations where inventory has to be monitored. It is connected to the computer via a USB 2.0 cable, where image processing is done in OpenCV. It takes the snapshot from the captured video, where further processing such as Image Acquisition, Dataset Augmentation etc. Furthermore, it has to be positioned properly to capture the entire area under observation.

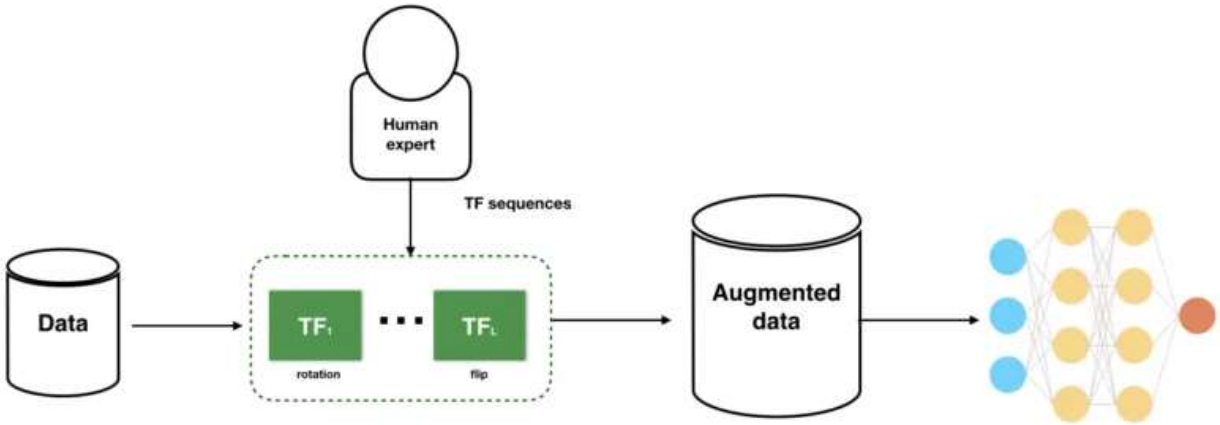
❖ Data Augmentation

The performance of most ML models, particularly deep learning models, is determined by the quality, quantity, and relevance of training data. However, one of the most common challenges in implementing machine learning in the venture is a lack of data. This is due to the fact that gathering such data can be pricey and time-consuming in many case scenarios.

Applications can use data augmentation to reduce their reliance on training data collection and preparation, allowing them to build more accurate machine learning models more quickly. Data augmentation is a set of techniques for artificially increasing data volume by generating new data points from existing data. Making minor changes to data or using deep learning models to generate new data points are examples of this.

Why it is important? The answer to this question is Machine learning applications, particularly in the deep learning domain, continue to diversify and expand at a rapid pace. Data augmentation techniques could be a useful tool in combating the challenges that the artificial world faces. Moreover, Data augmentation is useful for improving the performance and outcomes of machine learning models by generating new and diverse examples for training datasets. When a machine learning model's dataset is rich and sufficient, the model performs better and more accurately. Cleaning data is one of the steps in creating a data model, and it is required for high accuracy models. However, if cleaning reduces the representability of the data, the model will be unable to provide accurate predictions for real-world inputs. Data augmentation techniques improve the robustness of machine learning models by introducing variations that the model may encounter in the real world.

- How does it work?

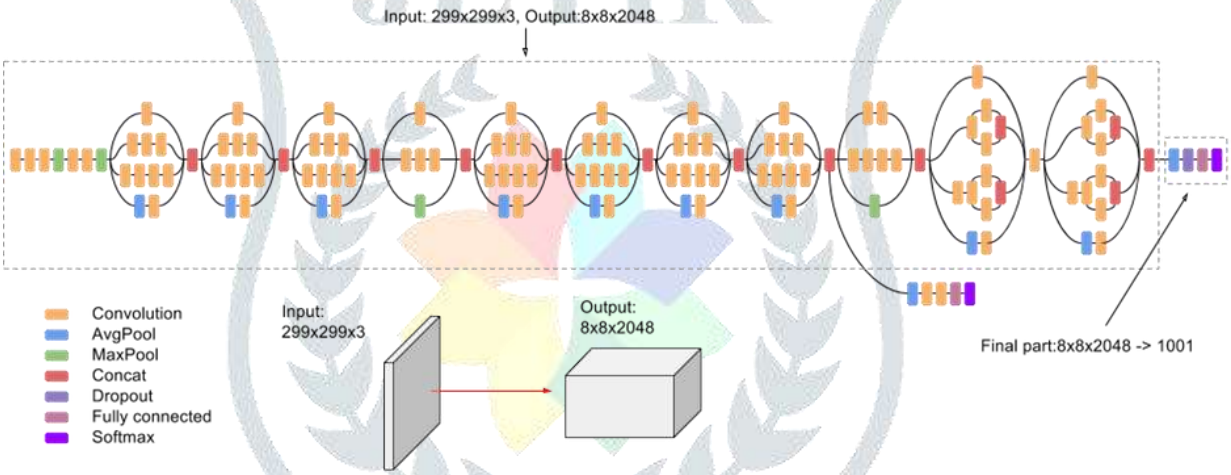


Making simple changes to visual data is popular for data augmentation. GANs (generative adversarial networks) are also used to generate new synthetic data. The following are examples of traditional image processing activities for data augmentation:

- Padding
- Random Rotating
- Re-scaling,
- Vertical and horizontal flipping
- Translation (image is moved along X, Y direction)
- Cropping
- Zooming
- Darkening & Brightening/Color Modification
- Gray scaling
- Changing Contrast
- Adding Noise
- Random Erasing

In this project, we have used five different image processing techniques to make our data more robust and increase prediction accuracy such as Rescaling, Zoom Range, Rotational Change, Height Shift, and Horizontal Flip.

❖ Inception V3 Model



Above diagram has been taken from official documentation

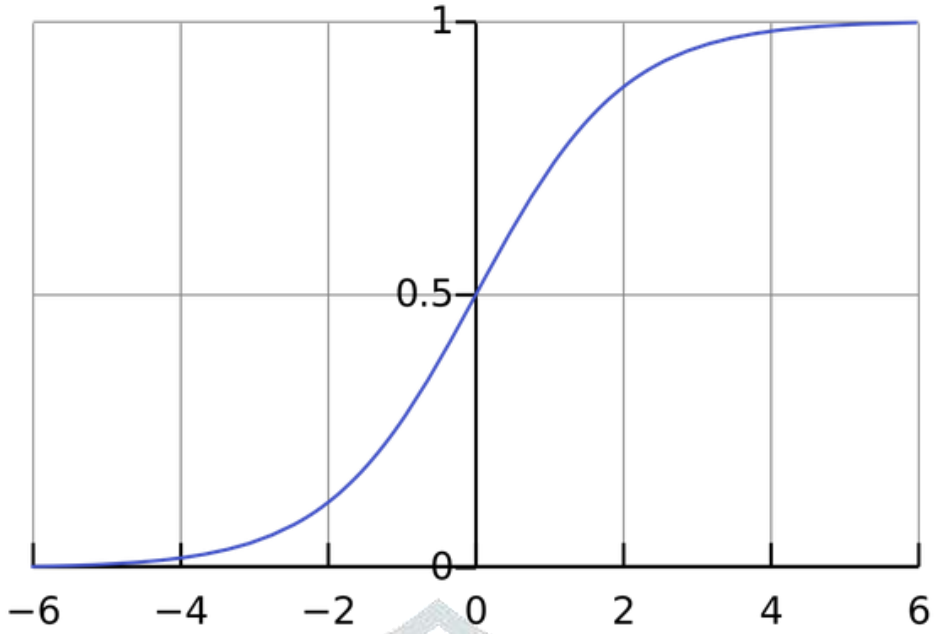
The above diagram depicts a high-level diagram of the model:

Inception v3 is a popular image recognition model that has been shown to achieve more than 78.1 percent accuracy on the ImageNet dataset. The model is the culmination of many ideas developed over time by various researchers. It is based on Szegedy et aloriginal .'s paper, "Rethinking the Inception Architecture for Computer Vision."

Convolutions, average pooling, max pooling, concatinations, dropouts, and fully connected layers are among the symmetric and asymmetric building blocks used in the model. Batch normalisation is applied to activation inputs and is used extensively throughout the model. Softmax is used to compute loss.

- Softmax Layer

  A neural network's activation function is an essential component. A neural network is nothing more than a simple linear regression model in the absence of an activation function. This means that the activation function gives the neural network non-linearity.



It is primarily used to normalise the output of neural networks so that it falls between zero and one. It is used to represent the network output's certainty "probability."

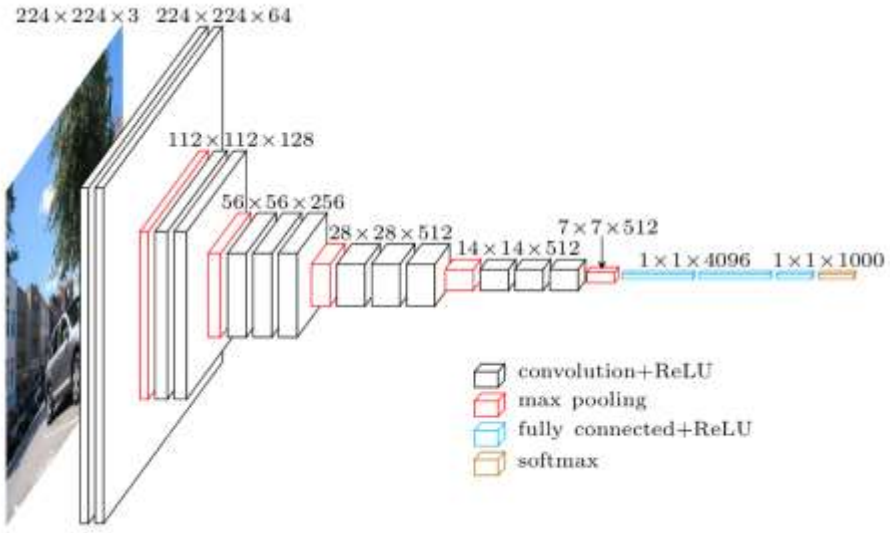
Normalization is calculated by dividing the exp value of the investigated output by the sum of the exp values of all possible outputs.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

- Global Spatial 2d Layer

A common choice for convolutional neural network (CNN) architecture for image classification tasks is repeated blocks of convolution and max pooling layers, followed by two or more densely connected layers. The final dense layer contains a softmax activation function as well as a node for each possible object category.

Consider the VGG-16 model architecture, which is depicted in the figure below.



After importing the VGG 16 model, we will have the following output

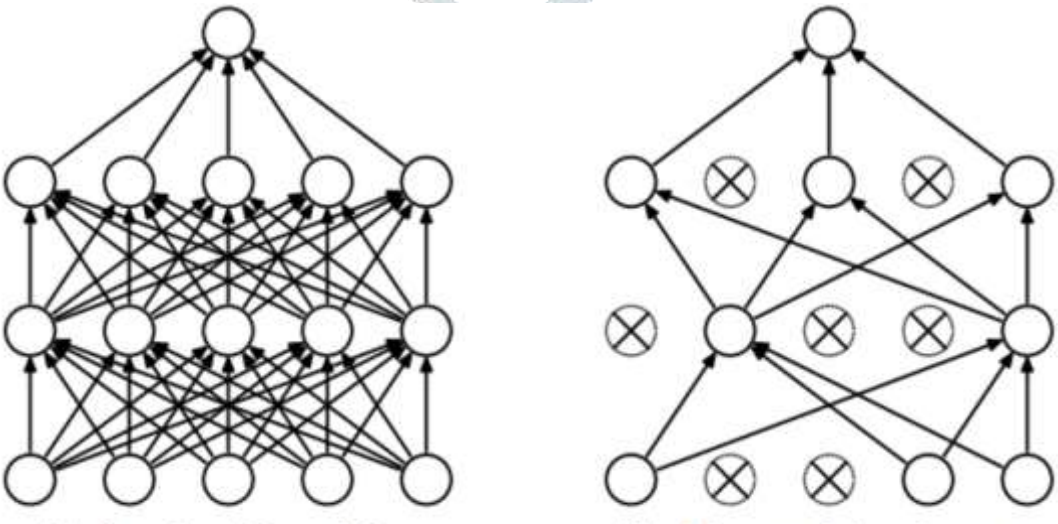
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
block1_conv1 (Convolution2D)	(None, 224, 224, 64)	1792	input_1[0][0]
block1_conv2 (Convolution2D)	(None, 224, 224, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block1_conv2[0][0]
block2_conv1 (Convolution2D)	(None, 112, 112, 128)	73856	block1_pool[0][0]
block2_conv2 (Convolution2D)	(None, 112, 112, 128)	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block2_conv2[0][0]
block3_conv1 (Convolution2D)	(None, 56, 56, 256)	295168	block2_pool[0][0]
block3_conv2 (Convolution2D)	(None, 56, 56, 256)	590080	block3_conv1[0][0]
block3_conv3 (Convolution2D)	(None, 56, 56, 256)	590080	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0	block3_conv3[0][0]
block4_conv1 (Convolution2D)	(None, 28, 28, 512)	1180160	block3_pool[0][0]
block4_conv2 (Convolution2D)	(None, 28, 28, 512)	2359808	block4_conv1[0][0]
block4_conv3 (Convolution2D)	(None, 28, 28, 512)	2359808	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0	block4_conv3[0][0]
block5_conv1 (Convolution2D)	(None, 14, 14, 512)	2359808	block4_pool[0][0]
block5_conv2 (Convolution2D)	(None, 14, 14, 512)	2359808	block5_conv1[0][0]
block5_conv3 (Convolution2D)	(None, 14, 14, 512)	2359808	block5_conv2[0][0]
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0	block5_conv3[0][0]
flatten (Flatten)	(None, 25088)	0	block5_pool[0][0]
fc1 (Dense)	(None, 4096)	102764544	flatten[0][0]
fc2 (Dense)	(None, 4096)	16781312	fc1[0][0]
predictions (Dense)	(None, 1000)	4097000	fc2[0][0]

Total params: 138357544

There are five blocks of (two to three) convolution layers, each followed by a max pooling layer. The final max pooling layer is flattened, and three densely connected layers follow. Take note that the fully connected layers contain the majority of the model's parameters!

As you might expect, an architecture like this runs the risk of over fitting to the training dataset. Dropout layers are used in practise to avoid over fitting.

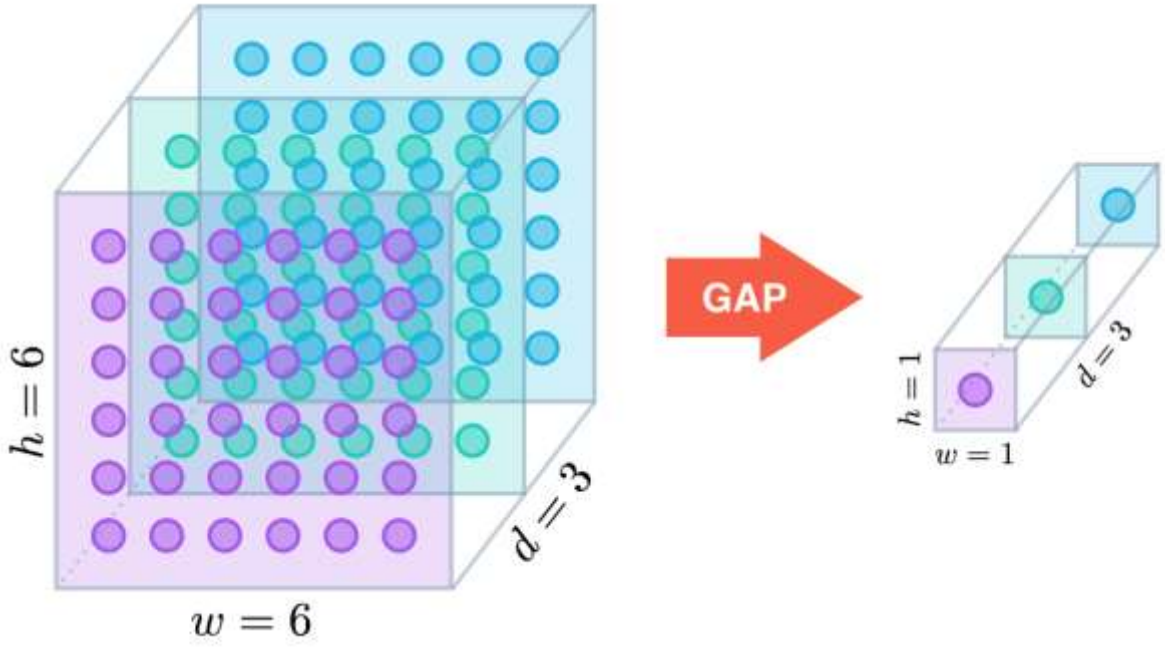
To avoid over fitting, we have used dense layer and dropout layer just after implementing GAP layers (Global Average Pooling). As shown in the diagram below, a dropout layer ignores a set of neurons (randomly). This is typically used to keep the net from over fitting. In a neural network, the dense layer is just a normal fully connected layer.



(a) Standard Neural Net (b) After applying dropout.

In the last few years, experts have turned to global average pooling (GAP) layers to minimize over fitting by reducing the total number of parameters in the model. Similar to max pooling layers, GAP layers are used to reduce the spatial dimensions of a three-dimensional tensor. However, GAP layers perform a more extreme type of

dimensionality reduction, where a tensor with dimensions h × w × d is reduced in size to have dimensions 1×1×d. GAP layers reduce each h × w feature map to a single number by simply taking the average of all h x w values.



Moreover, CNNs with GAP layers that have been trained for a classification task (a.k.a. GAP-CNNs) can also be used for object localization. That is, a GAP-CNN not only tells us what object is in the image, but it also tells us where the object is in the image, with no additional work on our part. The localization is represented as a heat map (also known as a class activation map), with a colour-coding scheme identifying regions that are relatively important for the GAP-CNN to perform the object identification task.

- Layer Freezing
  - In the context of neural networks, freezing a layer means controlling how the weights are updated. When a layer is frozen, the weights cannot be changed any further. This technique, as obvious as it may sound, reduces computational time for training while sacrificing little accuracy. Techniques such as Dropout and Stochastic depth have already demonstrated how to train networks efficiently without having to train every layer. Freezing a layer is another technique for accelerating neural network training by gradually freezing hidden layers.

  - For example, during transfer learning, the first layer of the network is frozen while the end layers are allowed to change. This means that if a machine learning model is trained to detect objects, running an image through it during the first epoch and then running the same image through it again during the second epoch yields the same value through that layer.

  - In other words, consider a two-layer network. The first layer is frozen, but the second layer is not. When we run 100 epochs, we perform the same computation through the first layer for each of the 100 epochs. The same images are run through the same layers without the weights being updated. This means that the inputs to the first layer are the same for each epoch (the images). The first layer's weights are the same, as are the first layer's outputs (images * weights + bias). By default, the pre-trained part is frozen, and only the last layers are trained, with the size of the change in weights in a layer determined by the learning rate.

V. METHODOLOGY

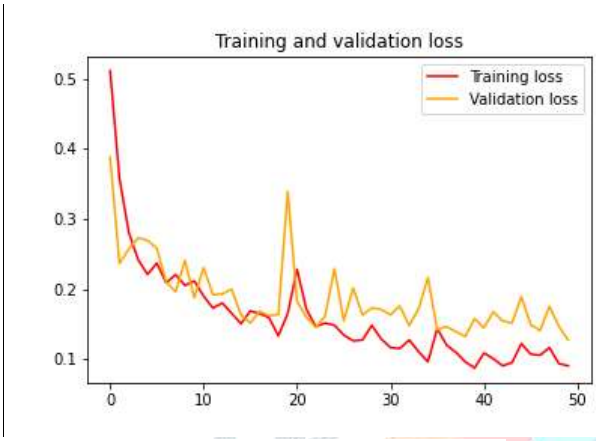
The model is divided in two parts

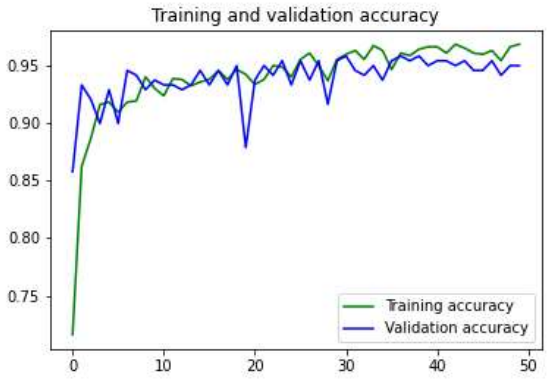
1. Creating a model having Dataset augmentation techniques and Inception V3 Model together
2. Importing the created model and performing Image acquisition techniques followed by if-else conditions and finally sending the alert to the officials.

The first step is to amass images for the problem statement. The dataset is split into two sections: fire and non-fire. The positive samples are images of actual fire. Images with false positives contain objects that appear to be fire but are not. False positives are easier to collect. As a result, we need to collect a wide range of images to aid in fire detection. For building our model, we used the TensorFlow API Keras. Kaggle datasets are used for training in this case. Finally, we have 1800 training images and 200 testing images. As a preprocessing step, we also used data augmentation. Rescaling, Zoom Range, Rotational Change, Height Shift, and Horizontal Flip are the five data augmentation techniques used.

We developed our CNN model. Three Conv2D-MaxPooling2D layer pairs are followed by three Dense layers in the model. To address the issue of overfitting, we will also include dropout layers. The final layer is the softmax layer, which will provide us with the probability distribution for both the Fire and Nonfire classes.

With a learning rate of 0.0001, we used Adam as an optimizer. After 50 epochs of training, we obtain a training accuracy of 96.83 and a validation accuracy of 94.98. The training and validation losses are respectively 0.09 and 0.13.





Following that, we ran our model against any image to see if it could correctly guess it. We chose three images for testing: a fire image, a non-fire image, and a photo of myself with fire-like colours and shades.

Here, in the below image we can see that our previously created model is incorrectly classifying our images. The model is 52 percent certain that the image contains fire. This is due to the dataset on which it was trained. The dataset that teaches a model about indoor fires contains very few images. As a result, because the model only knows about outdoor fires, it makes an error when presented with an indoor fire-like shaded image. Another reason is that our model is not complex enough to learn complex fire features. Next, we'll take a standard InceptionV3 model and customise it. A complex model has the ability to learn complex features from images.

```
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]))
```

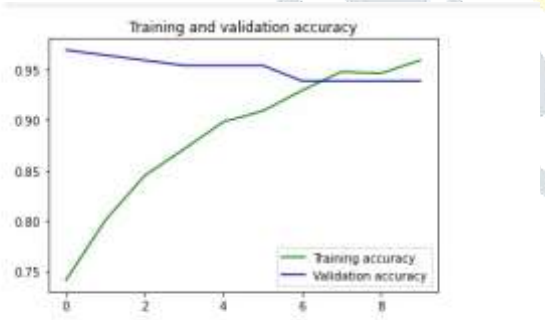
```
Choose Files 3 files
• fire.jpg(image/jpeg) - 6720 bytes, last modified: 7/1/2020 - 100% done
• me.jpg(image/jpeg) - 86686 bytes, last modified: 7/5/2020 - 100% done
• nonfire.jpg(image/jpeg) - 10472 bytes, last modified: 7/1/2020 - 100% done
Saving fire.jpg to fire (3).jpg
Saving me.jpg to me (3).jpg
Saving nonfire.jpg to nonfire (3).jpg
True 0.9651741
True 0.5265175
False 0.994662
```

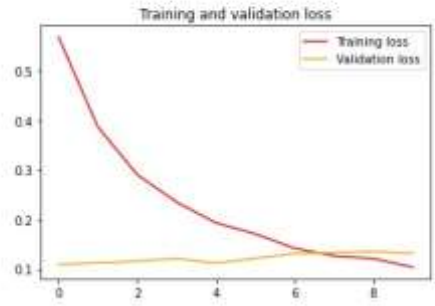
This time, we'll use a different dataset , one that includes both outdoor and indoor fire images. I trained our previous CNN model on this dataset, and it over fitted because it couldn't handle the larger dataset and learn complex features from the images. Let us begin by developing the ImageDataGenerator for our customized InceptionV3. The dataset contains three classes, but we will only use two in this article. It includes 1800 training images and 200 validation images. In addition, I added 8 images of my living room to the dataset to add some noise.

We can use data augmentation techniques to improve training accuracy even further. We added two new data augmentation techniques this time: horizontal flipping and zooming. We used the Keras API to import our InceptionV3 model. Our layers were added to the top of the InceptionV3 model. To ensure that our model does not overfit, we added a global spatial average pooling layer, followed by two dense layers and two dropout layers. Finally, for two classes, we added a softmax activated dense layer.

Following that, we will train only the layers that we added and randomly initialised. RMSprop will be used as an optimizer in this case. After training our top layers for 20 epochs, we will freeze the first 249 layers of the models and train the remaining layers, which will be the top two inception blocks. We used SGD as an optimizer with a learning rate of 0.0001 in this case.

After ten epochs of training, we obtained a training accuracy of 98.04 and a validation accuracy of 96.43. The losses for training and validation are 0.063 and 0.118, respectively.





We tested our model again for the same images, and this time it correctly predicted all three. My image is 96 percent certain that it does not contain any fire.

```
from google.colab import files
from keras.preprocessing import image

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]))
```

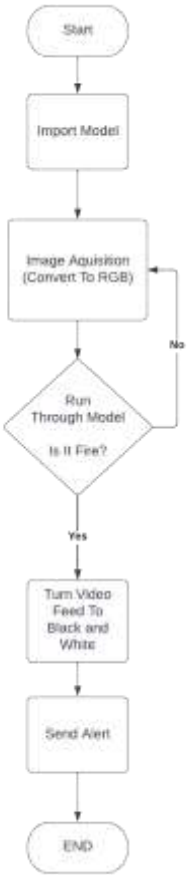
Choose Files 3 files

- **fire.jpg**(image/jpeg) - 6720 bytes, last modified: 7/1/2020 - 100% done
- **me.jpg**(image/jpeg) - 86686 bytes, last modified: 7/5/2020 - 100% done
- **nonfire.jpg**(image/jpeg) - 10472 bytes, last modified: 7/1/2020 - 100% done

```
Saving fire.jpg to fire.jpg
Saving me.jpg to me.jpg
Saving nonfire.jpg to nonfire.jpg
True 0.9448784
False 0.9697459
```

Our model is ready for testing with real-world scenarios. We used OpenCV to gain access to our webcam and predict whether or not each frame contained fire. If a frame contains fire, we want to change the colour of that frame to B&W and, finally, we will notify the authorities if the fire is detected by sending emails or using any mobile notification technique.

❖ Flowchart



VI. RESULT

The goal of our work was to create a robust application that detects fire in live video feeds and works in any environment. In this regard, we implemented and used the Inception V3 Model for detecting fires in conjunction with data augmentation techniques because it provided the best performance and accuracy, such as 96 percent. In order to provide real-time alerts to the concerned stakeholders, we have also included an email alert feature in our application. The graphical user interface (GUI) provides a user-friendly experience and enables users with non-technical backgrounds to use the application. During testing, the application performed admirably. It correctly identified fires in all of the images and videos but incorrectly classified some non-fire videos and images.

In comparison to existing hardware solutions, our application is less expensive, more robust, and reliable, and provides high performance without the need for a dedicated infrastructure. Our model outperforms existing software solutions that make extensive use of feature engineering due to the use of machine learning, computer vision, and deep learning algorithms.

❖ Non-fire videos testing results

Test Case Description	Expected Outcome	Actual output	Test Status (P/F)
Video with visuals of pools,indoor	Non-fire	Non fire	P
Traffic	Non-fire	Non-fire	P
Headlight glare during night	Non fire	Fire	F
Camera focused on sun	Non fire	Non fire	P
Inside an airport	Non fire	Non fire	P
Shopping Mall	Non fire	Non fire	P
Kid going to school with a orange bag	Non fire	Non fire	P
Snow mountain	Non fire	Non fire	P

❖ Fire videos testing results

Test case description	Expected Output	Actual output	Test status (P/F)
Burning forest	Fire	Fire	P
Fireworks	Fire	Fire	P
Campfire	Fire	Fire	P
Bush Fire	Fire	Fire	P

Fire on car	Fire	Fire	P
Leaves on fire	Fire	Fire	P
Gas station explosion	Fire	Fire	P

VII. Conclusion

This project proposed a fire detection algorithm that is devoid of sensors, as are most fire detection systems. The goal of this project was to develop a system that could detect fire as early as possible from a live video feed and send an alert to an official without the need for human intervention. The System is designed to detect fires while they are still small and have not grown to enormous proportions. Furthermore, the hardware is minimal and already present in places, saving capital. It also saves money by eliminating the need for expensive temperature and heat sensors, among other things. The system has proven to be effective at detecting fire based on the results. This system is a fusion of various fire detection algorithms.

VIII. Future Work

Future research may concentrate on making this system weatherproof, combining smoke detection with fire detection, and achieving lower latency by optimizing the system. Furthermore, this system can be embedded on a drone or another UAV for security purposes, which will be very useful in detecting forest fires, and it can also be used in military applications, such as land and sea rescue operations.

References

[1]. National Risk Survey Report - Pinkerton, FICCI (2018).
[2]. Janku P., Kominkova Oplatkova Z., Dulik T., Snopek P. and Liba J. 2018. “Fire Detection in Video Stream by Using Simple Artificial Neural”. Network. MENDEL. 24, 2 (Dec. 2018), 55–60.
[3]. Shen, D., Chen, X., Nguyen, M., & Yan, W. Q. (2018). “Flame detection using deep learning”. 2018 4th International Conference on Control, Automation and Robotics (ICCAR).
[4]. Li, C., & Bai, Y. (2018). “Fire Flame Image Detection Based on Transfer Learning”. 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS).
[5]. K. Muhammad, J. Ahmad, I. Mehmood, S. Rho and S. W. Baik, “Convolutional Neural Networks Based Fire Detection in Surveillance Videos," in IEEE Access, vol. 6, pp. 18174-18183, 2018.
[6]. S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.
[7]. [Zhang, Qingjie & Xu, Jiaolong & Xu, Liang & Guo, Haifeng. (2016). “Deep Convolutional Neural Networks for Forest Fire Detection”.
[8]. K. Muhammad, J. Ahmad, Z. Lv, P. Bellavista, P. Yang and S. W. Baik, “Efficient Deep CNN-Based Fire Detection and Localization in Video Surveillance Applications,” in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 7, pp. 1419-1434, July 2019.
[9]. Z. Jiao et al., “A Deep Learning Based Forest Fire Detection Approach Using UAV and YOLOv3,” 2019 1st International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, 2019.
[10].Faming Gong ,1 Chuantao Li,1 Wenjuan Gong ,1 Xin Li,1 Xiangbing Yuan,2 Yuhui Ma and Tao Song. “A RealTime Fire Detection Method from Video with Multifeature Fusion”. Hindawi, Computational Intelligence and Neuroscience, Volume 2019.
[11].L. Shao, F. Zhu and X. Li, “Transfer Learning for Visual Categorization: A Survey,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 5, pp. 1019-1034, May 2015.
[12].S. Mohd Razmi, N. Saad and V. S. Asirvadam, “Visionbased flame detection: Motion detection & fire analysis,” 2010 IEEE Student Conference on Research and Development (SCOReD), Putrajaya, 2010, pp. 187-191.
[13].T. Qiu, Y. Yan and G. Lu, “A new edge detection algorithm for flame image processing,” 2011 IEEE International Instrumentation and Measurement Technology Conference, Binjiang, 2011, pp. 1-4.
[14].M. Ligang, C. Yanjun and W. Aizhong, “Flame region detection using color and motion features in video sequences,” The 26th Chinese Control and Decision Conference (2014 CCDC), Changsha, 2014, pp. 3005- 3009.
[15].Toulouse, Tom & Rossi, Lucile & Celik, Turgay & Akhloufi, Moulay. (2015). “Automatic fire pixel detection using image processing: A comparative analysis of Rulebased and Machine Learning-based methods. Signal, Image and Video Processing”.
[16].K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.
[17].S. Wilson, S. P. Varghese, G. A. Nikhil, I. Manolekshmi and P. G. Raji, “A Comprehensive Study on Fire Detection,” 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), Tiruchengode, 2018, pp. 242- 246.
[18].X. Wu, X. Lu and H. Leung, “An adaptive threshold deep learning method for fire and smoke detection,” 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, 2017, pp. 1954-1959.

[19].Chenebert, Audrey & Breckon, Toby & Gaszczak, Anna. (2011). "A nontemporal texture driven approach to realtime fire detection".
[20].Seebamrungsat, Jareerat et al. "Fire detection in the buildings using image processing." 2014 Third ICT International Student Project Conference (ICTISPC) (2014): 95-98.
[21] OpenCV official website http://www.opencv.org
[22] Sonali K. M., Poojashree M., Shilpashree A., Sindhu N., Rekha K. S.,"Automated Fire Detection Surveillance System", CSE, NIE, Mysuru,International Journal of Current Trends in Engineering & Research (IJCTER) e-ISSN 2455– 1392 Volume 2 Issue 6, June 2016 pp. 49 – 52.
[23] P. KadewTraKuPong, R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection" 2001
[24] Turgay Celik, Hasan Demirel, Huseyin Ozkaramanli, and Mustafa Uyguroglu, "Fire detection using statistical color model in video sequences," J. Vis. Comun. Image Represent., vol. 18, pp. 176–185, April 2007
[25] Rafael C. Gonzalez and Richard E. Woods: "Digital Image Processing", 2009
[26] Global Average Pooling https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/