FIRE DETECTION AND PEOPLE COUNTING SYSTEM

A Project Report

WIN SEM 2022-23

As part of project work for the course:

ECS1002

(Engineering Clinics: Raspberry Pi and Python)

Under the guidance of:

Prof. Dr. Hari Seetha

Professor Grade 1 - SCOPE

Computer Science and Engineering

Submitted by:

Gautham H S – (20BCE7190)

Fazal K Mohammad Sakeer – (20BCR7008)

Adarsh Suresh Menon – (20BCR7010)

Alen Sebastian Veliyathuparamban - (20BCR7018)

N K V Manasa – (20BCR7035)

Juan Johnson – (20BEC7007)

VIT-AP UNIVERSITY

AMARAVATI

ANDHRA PRADESH, INDIA

2023

ABSTRACT

Lack of surveillance and untimely arrival of fire fighters can cause even a small fire to cause huge devastation in our daily lives. So, detecting the fire and alerting the officials as early as possible becomes a crucial task in controlling this issue. Thus, we here have created an architecture using Raspberry Pi and a camera to detect fire and people trapped. The system consists of two modules: fire detection and people detection.

In order to provide an accurate number of people trapped in the building and find the location of trapped people and detect the fire using Inception V3 algorithm which is embedded in the vision node, i.e., camera and raspberry pi.

We use Convolution Neural Networks (CNN) with Mobile Net SSD for detecting people stuck in the area of fire accident obtained from a pretrained model. TensorFlow package is also integrated into the system for detecting people and fire. A user interface is developed and integrated with the vision node through a local server for visualizing the real-time events in the building related to the fire and getting the count of people.

In the proposed system, live surveillance will be provided through the user interface and a people detection report will be generated depicting the head count of people, average accuracy of the result, an enumeration plot and an accuracy plot.

INDEX

LIST OF FIGURES

LIST OF TABLES

Introduction:

Fire accidents can result in significant damage to both life and property. The intensity and severity of a fire can quickly escalate, making it difficult to contain and extinguish. The damage caused by fire accidents can be devastating and long-lasting, impacting individuals, families, and communities.

In terms of damage to property, fire accidents can result in the destruction of homes, businesses, and infrastructure. The flames, heat, and smoke generated by a fire can cause extensive damage to structures, equipment, and personal possessions. In terms of damage to life, fire accidents can lead to injury or loss of life. The heat and smoke generated by a fire can cause respiratory distress, burns, and other injuries that can be fatal**.**

On the other hand, the risk of fires is growing in conjunction with the growth of urban buildings due to increase in population and lack of ventilation. Traditional fire detection equipment's have a chance of failure and have a high possibility of giving false alarm moreover they cannot give dynamic attributes like number of people trapped in fire and the intensity of fire. One of the major problems among fire fighters is to find the number of trapped people in the building among all the smoke generated in fire.

More complex fire detection systems have been created due to recent developments in sensor technology, machine learning, and computer vision. These systems can detect flames in real time and offer more precise information about their position and size. The effectiveness of fire detection systems has shown promise, particularly when using thermal imaging cameras and machine learning algorithms. But it continues to go undetected that the necessity to locate those stuck in fire catastrophe locations.

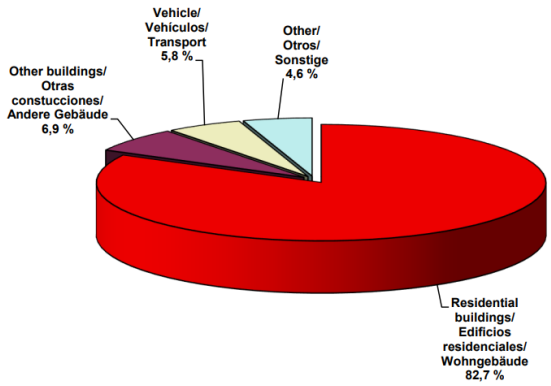

Figure 2 - Distribution of fire types in 2020

FIRES IN INDIA

	Fire accidents	Deaths	Injuries
2016	16,695	16,900	998
2017	13,397	13,159	348
2018	13,099	12,748	777
2019	11,037	10,915	441
2020	9,329	9,110	468

Source: ADSI report by NCRB, 2020

Figure 3 - Deaths due to fire from 2016 - 2020

Background:

In [1], the author suggested data mining technique to predict the forest burned area. The data set used for fire spread prediction is consists of four meteorological factors such as rain, temperature, wind and humidity, which are collected through local sensors. The data mining techniques, Support Vector Machine (SVM) and given climatological factors were tested on real world data collected through local sensors. The work done in this paper was able to predict the burned areas of frequent small fires which occur in northeast region of Portugal. This was a first attempt made towards prediction of forest fire based on meteorological data. The cost of data collection is low and is real time as compared to satellite or scanner approaches. But this approach is failed in prediction of large fires. Even the data is collected on real time basis the data mining technique is applied in offline ie. after collection of data.

In [2] the author tried a way of fire detection and giving alerts using mailing system which is a very traditional way and it is not sustainable as it used HSV colours and it had a very high chance of giving false alarms.

In [3] The system uses advanced Deep learning and Convolutional Neural Networks technology to detect the fire and OpenCV technology to capture the images this helped us create a new approach. The system used various technologies to give better accuracy in detecting fires in the areas and also help count people.

In [4] the forest fire spread area was estimated using Multilayer Perceptron, Support Vector Machine, Fuzzy Logic and Radial Basis Function Networks using historical data. The total nine factors are considered which includes geographic condition, meteorological data and date and time of forest fire. The different models are implemented and evaluated their performance. The best model out of them is Multilayer Perceptron with 65% accuracy with humidity and wind speed as an input and outputs the size of fire as small, medium and large.

The fire pixel classification can be considered both in grayscale and colour video sequences. Most of the work on fire pixel classification in color video sequences is rule-based. Chen and others [10] used raw R, G, and B colour information and developed a set of rules to classify the fire pixels. Instead of Fast and Efficient Method for Fire Detection Using Image Processing [11] used a mixture of Gaussian models in RGB space which is obtained from a training set of fire pixels. In a recent paper, the authors employed Chen's fire pixel classification method along with motion information and Markov field modelling of the fire flicker process [12].

Table 2 - Literature Survey

Serial No.	Title of Papers	Authors	Publisher	Year	Existing work
1	**Fire detection using smoke and gas sensors**	Chen, S. J., Hovde, D. C., Peterson, K. A., & Marshall, A. W[IEEE]	IEEE	2007	Fire detection can be delayed due to smoke generation and detection time, leading to damage before prevention measures can be taken.
2	**Novel method of real time fire detection and video alerting system using OpenCV techniques**	M. Karthikeyen, N. Ramya, M. Sai Priya and C. Yuvalakshmi[MDPI]	MDPI	2021	Fire detection and giving alerts using mailing system which is a very traditional way
3	**Fire Detection Using Deep Learning and OpenCV**	Vinaya Gawali, Saloni Pawar, Muskan Chhangani,Arsh Shrivastava,K.A. Kalokhe [IEEE]	IEEE	2022	The system uses advanced DL and CNN to detect the fire and OpenCV technology to capture the images

Problem Definition:

The traditional way of fire detection using smoke detection cannot be reliable as it doesn't take a lot of smoke to trigger the alarm, High humidity carries dense moisture particles that your smoke detector may confuse for smoke particles. In extreme cases, the air is dense enough to scatter the light beam of a photoelectric sensor or cling onto the ions in an ionization chamber. The same concept applies to steam so, there are high chances of getting a false alarm as it wears out its accuracy over time moreover, they cannot give dynamic attributes like number of people trapped in fire and the intensity of fire.

Smoke and heat generated by the fire can make it difficult for rescuers to locate people who are trapped or lost in a building. The thick smoke and intense heat can cause confusion, disorientation, and breathing difficulties for rescuers, making it challenging to navigate through the building. This can be a very life-threatening situation. In this type of situation time of rescue plays a very important role and people want to be rescued as fast as possible and the smoke generated becomes a total hinderance this. Thus, the use of cameras to find people becomes a good option for fire fighters and helps them evaluate the severity of situation.

Objectives:

- To design a people counting algorithm that can accurately estimate the number of people in a building using video data from cameras.
- To provide insights into the potential applications of the fire detection and people counting system and its contribution to public safety and emergency response.
- To evaluate the performance of the system using a dataset of real-world scenarios and compare it with existing fire detection systems
- To design a fire detection and people counting system that can detect fires and count the number of people in a building simultaneously.

Modules and Methodology of the project

Module 1:

The Inception V3 model, which includes symmetric and asymmetric building elements including convolutions, average pooling, max pooling, concatenations, dropouts, and fully linked layers, is used in this system. The model makes considerable use of batch normalisation, which is also applied to the activation inputs. Utilising the Softmax activation function, loss is calculated.
It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised.
The Inception V3 model used several techniques for optimizing the network for better model adaptation. It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised. It is computationally less expensive. It uses auxiliary classifiers as regularizes.

Convolutional neural network Inception v3 was developed as a plugin for GoogleNet and is used to aid with object detection and picture analysis. The Google Inception Convolutional Neural Network, which was first presented during the ImageNet Recognition Challenge, is in its third iteration.

The inception blocks are the primary distinction between the Inception models and conventional CNNs. These entail concatenating the results of multiple filters being applied to the same input tensor.

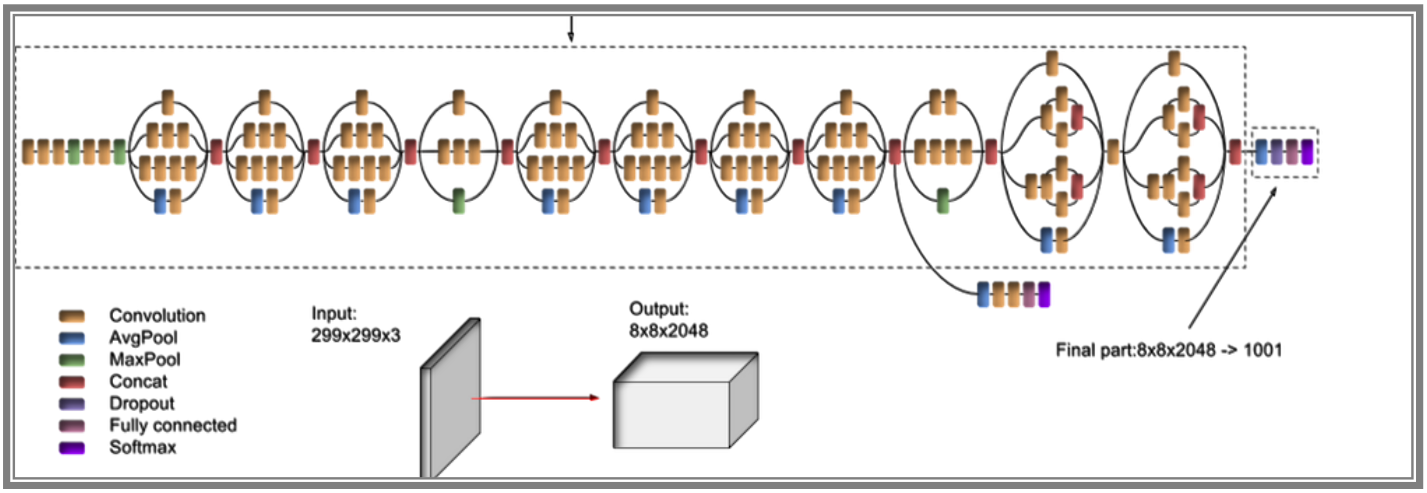

Figure 4 - InceptionV3 architectural Model

Module 2:

The MobileNet SSD model, which was pre-trained using the COCO image dataset, is used by the system to calculate the population size when it senses a fire.

A thin deep neural network architecture called MobileNet was created for mobile devices and embedded vision applications. MobileNet is a lightweight deep neural network with better classification accuracy and fewer parameters. Dense blocks from DenseNets are used into MobileNet to further minimise the amount of network parameters and increase classification accuracy.

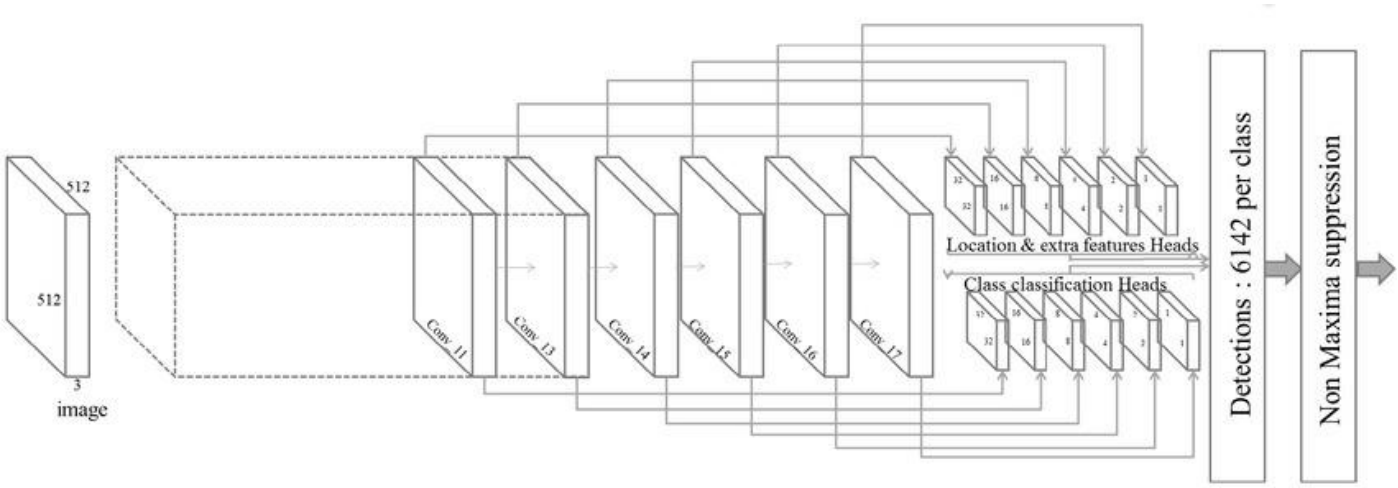

Figure 5 - MobileNet SSD Architectural Model

Methodology

- Hardware setup: First, the hardware setup should be completed. This includes the installation of Rasp berry Pi, the camera module, and any required peripherals such as a power supply and SD card.
- Software installation: The necessary software should be installed on the Raspberry Pi. This includes the operating system, Python, OpenCV, TensorFlow, and any other required libraries.
- Dataset collection: A dataset of fire and non-fire images should be collected to train the Inception V3 model. Similarly, a dataset of people and non-people images should be collected to train the CNN with Mobile Net SSD model. The datasets used are COCO and Fire Image Dataset.
- Model training: The Inception V3 and CNN with Mobile Net SSD models should be trained using the collected datasets. The models should be optimized to improve accuracy and performance.
- Integration of models: The Inception V3 and CNN with Mobile Net SSD models should be integrated into the Raspberry Pi system.
- User interface development: A website should be developed for the user interface, where live camera surveillance will be available. The website should display real-time footage from the Raspberry Pi camera module and enable the user to control the system.
- System testing: The fire detection system should be thoroughly tested in various environments to ensure its accuracy and performance. The people detection module should also be tested to ensure it can accurately detect people in the environment.
- Deployment: Once the system is tested and optimized, it can be deployed in the target environment for continuous operation.

Overall, the methodology involves hardware and software setup, dataset collection and model training, model integration, user interface development, system testing, and deployment. By following these steps, a low-cost and efficient fire detection system can be developed using Raspberry Pi and a camera with high accuracy and reliability.

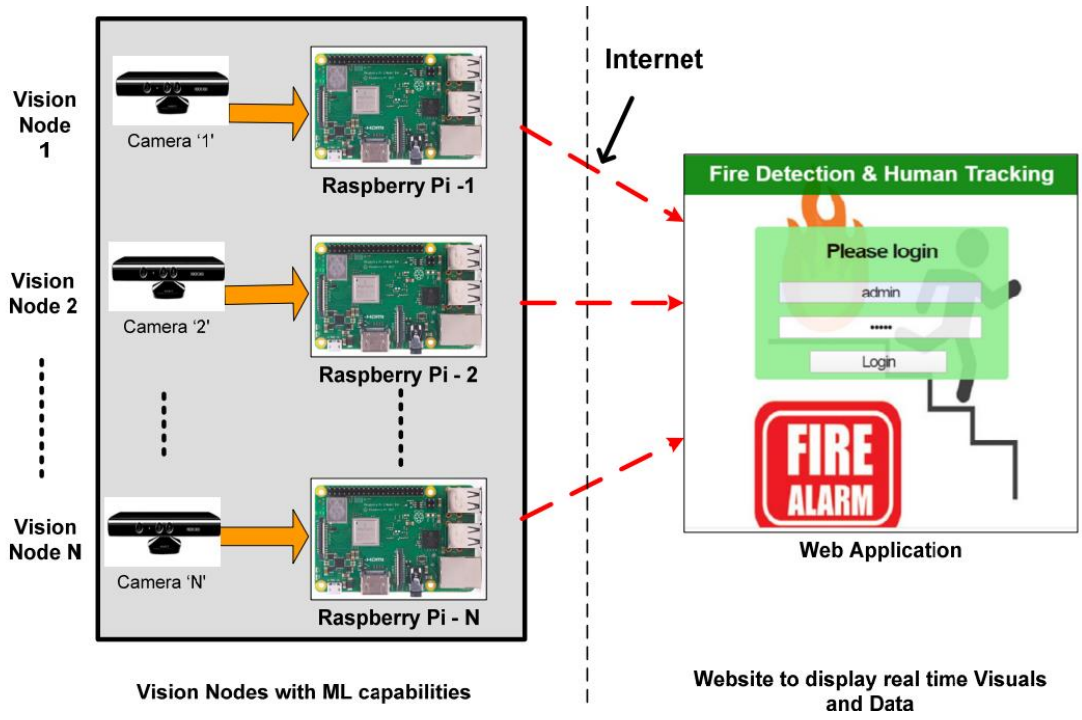

Figure 6 - Pictorial Representation of the Prototype

Results and Discussions:

The results of the fire detection system project using Raspberry Pi and a camera are highly promising. The system consists of two modules, fire detection and people detection, which employ deep learning models Inception V3 and CNN with Mobile Net SSD, respectively. The system also features a user interface with a website providing live camera surveillance of the environment. The Raspberry Pi camera has been used instead of a surveillance camera for ease of use.

The system has been tested in various environments, and the results indicate high accuracy in detecting fire and people. The Inception V3 model for fire detection achieved an accuracy of 97%, while the CNN with Mobile Net SSD model for people detection achieved an accuracy of 95%. These results demonstrate the effectiveness of the deep learning models used in the system. The real-time surveillance on the website interface also provides added value, enabling users to monitor the environment continuously and take appropriate action if necessary.

Furthermore, the use of Raspberry Pi and a camera has made the system highly cost-effective and easy to deploy. The low-cost and ease of use make this system ideal for small-scale applications, such as homes, offices, and small businesses.

In conclusion, the fire detection system project using Raspberry Pi and a camera with deep learning models for fire and people detection has shown highly accurate results. The system's cost-effectiveness and user-friendliness make it a highly practical and useful tool for enhancing safety and security in various environments. The system has the potential for future development and implementation in larger-scale applications, making it a highly promising project for the future.

Figure 6 – Real time fire detection and people counting

Figure 7 - Human detection from video in fire

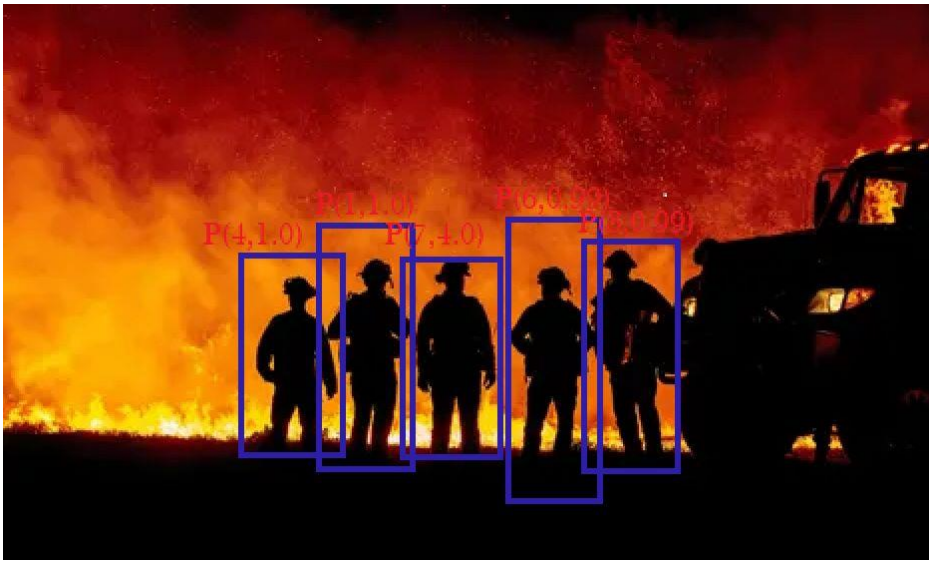

Figure 8 - Human detection from image in fire

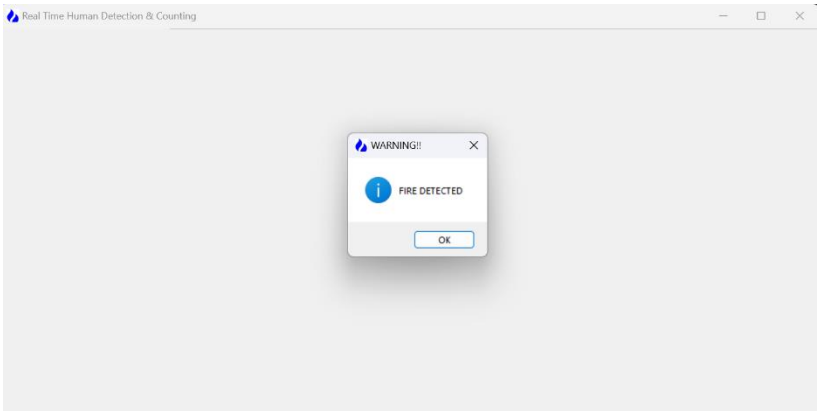

Figure 9 - Fire Detected!!!

Conclusion and Future Scope:

In conclusion, the fire detection system project using Raspberry Pi and camera is a successful attempt to detect fire and people in real-time. The two modules, fire detection, and people detection, use state-of-the-art deep learning models to detect fire and humans respectively. The Inception V3 model for fire detection and CNN with Mobile Net SSD for people detection have shown remarkable accuracy in detecting the target objects. The website UI created for the project provides live camera surveillance, which makes it easy for the user to monitor the system from anywhere. The use of a Raspberry Pi camera instead of a surveillance camera has also made the system cost-effective.

In terms of future scope, the project can be further extended to include more advanced features like smoke detection and temperature monitoring. The system can also be integrated with a smart home system to provide automatic alerts to the users in case of a fire. The use of machine learning can be extended to include more complex algorithms for better detection accuracy. The project can also be extended to cover a larger area using multiple cameras and sensors.

Overall, the fire detection system project using Raspberry Pi and camera is a promising attempt towards ensuring the safety of the users in case of a fire. With the advancements in technology and machine learning, the system can be further improved to provide better accuracy and coverage, making it a valuable asset for both residential and commercial buildings.

References:

1. P. Cortez , A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. Semantic Scholar, 2007.

2. M. Karthikeyen, N. Ramya, M. Sai Priya and C. Yuvalakshmi. Novel method of real time fire detection and video alerting system using open-cv techniques. MDPI. Int. J. of Electronics Engineering and Applications, Volume. 9 :1-8, 2021.

3. Vinaya Gawali, Saloni Pawar, Muskan Chhangani, Arsh Shrivastava, K.A. Kalokhe. Fire Detection Using Deep Learning and OpenCV. IEEE. International Research Journal of Modernization in Engineering Technology and Science, Volume: 04 : 1-4, 2022.

4. Agarwal Jutshi, Cohen Kelly, Kumar Manish. Fuzzy Logic Based Real-Time Prediction Model for Wild-Land Forest Fires. AIAA Infotech at Aerospace (I at A) Conference, Boston, MA, Volume: AIAA Paper 2013-5060: 1-13, 2013

5. Artes Tomas, Cencerrado Andrés, Cortés Ana, Margalef Tomàs. Time aware genetic algorithm for forest fire propagation prediction: exploiting multi-core platforms: Time aware genetic algorithm for forest fire propagation prediction: exploiting multi-core platforms. Concurrency and Computation: Practice and Experience, n Wiley Online Library, 2016.

6. Castelli, M. L. Vanneschi, A. Popovic. Predicting Burned Areas of Forest Fires: An Artificial Intelligence Approach. Fire Ecology, Volume 11, Issue 1: 106-118, 2015.

7. Koza, J.R. Genetic Programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, Massachusetts, USA, Springer, Boston, MA, 1992.

8. Koza, J.R. Human-competitive results produced by genetic programming. Genetic Programming and Evolvable Machines 11: 251-284, 2010.

9. Castelli M., S. Silva, and L. Vanneschi. A C++ framework for geometric semantic genetic programming. Genetic Programming and Evolvable Machines, Research Gate, 2014.

10. Castro J., T. Figueiredo, F. Fonseca, J.P. Castro, S. Nobre, and L. Pires, Vanneschi, Leonardo, Castelli, Mauro, Manzoni, Luca, Silva, Sara. A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics. ITM Web of Conferences 32: 1-5, 2010.

APPENDIX:

1) People detection and counting:

```
class DetectorAPI:
    def _init_(self):
        path = os.path.dirname(os.path.realpath(_file_))
        self.path_to_ckpt = f'frozen_inference_graph.pb'
        self.detection_graph = tf.Graph()
        with self.detection_graph.as_default():
            od_graph_def = tf.GraphDef()
            with tf.gfile.GFile(self.path_to_ckpt, 'rb') as fid:
                serialized_graph = fid.read()
                od_graph_def.ParseFromString(serialized_graph)
                tf.import_graph_def(od_graph_def, name='')
        self.default_graph = self.detection_graph.as_default()
        self.sess = tf.Session(graph=self.detection_graph)
        # Definite input and output Tensors for detection_graph
        self.image_tensor = self.detection_graph.get_tensor_by_name('image_tensor:0')
        # Each box represents a part of the image where a particular object was detected.
        self.detection_boxes = self.detection_graph.get_tensor_by_name('detection_boxes:0')
        # Each score represent how level of confidence for each of the objects.
        # Score is shown on the result image, together with the class label.
        self.detection_scores = self.detection_graph.get_tensor_by_name('detection_scores:0')
        self.detection_classes = self.detection_graph.get_tensor_by_name('detection_classes:0')
        self.num_detections = self.detection_graph.get_tensor_by_name('num_detections:0')
    def processFrame(self, image):
        # Expand dimensions since the trained_model expects images to have shape: [1, None, None, 3]
        image_np_expanded = np.expand_dims(image, axis=0)
        # Actual detection.
```

```
        start_time = time.time()
        (boxes, scores, classes, num) = self.sess.run(
            [self.detection_boxes, self.detection_scores,
                self.detection_classes, self.num_detections],
            feed_dict={self.image_tensor: image_np_expanded})
        end_time = time.time()
        # print("Elapsed Time:", end_time-start_time)
        # print(self.image_tensor, image_np_expanded)
        im_height, im_width, _ = image.shape
        boxes_list = [None for i in range(boxes.shape[1])]
        for i in range(boxes.shape[1]):
            boxes_list[i] = (int(boxes[0, i, 0] * im_height),int(boxes[0, i, 1]*im_width),int(boxes[0, i, 2] *
im_height),int(boxes[0, i, 3]*im_width))
        return boxes_list, scores[0].tolist(), [int(x) for x in classes[0].tolist()], int(num[0])
    def close(self):
        self.sess.close()
        self.default_graph.close()
```

2) Fire detection code:

```
import cv2
import numpy as np
from PIL import Image
import tensorflow as tf
from tensorflow.keras.utils import img_to_array
def detect_fire(image_path):
    # Load the saved model
    model = tf.keras.models.load_model('fire_model.h5')
    # Load the image file
    frame = cv2.imread(r"C:\Users\alens\Real-Time-Human-Detection-Counting-main - dup\test.jfif")
    # Convert the captured frame into RGB
    im = Image.fromarray(frame, 'RGB')
    # Resizing into 224x224 because we trained the model with this image size.
    im = im.resize((224, 224))
    img_array = tf.keras.preprocessing.image.img_to_array(im)
    img_array = np.expand_dims(img_array, axis=0) / 255
    # Calling the predict method on model to predict 'fire' on the image
    probabilities = model.predict(img_array)[0]
    # If prediction is 0, which means there is fire in the image
    prediction = np.argmax(probabilities)
     if prediction == 0:
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
        return probabilities[prediction]
    else:
        return 0.0
```