# Smart Surveillance System using Deep Learning and RaspberryPi

2021 8th International Conference on Smart Computing and Communications (ICSCC) | 978-1-7281-9687-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSCC51209.2021.9528194

Kshitij Patel
*B.Tech in Computer Science and Engineering*
*Pandit Deendayal Petroleum(Energy) University*
Gandhinagar, India
patelka1206@outlook.com

Meet Patel
*B.Tech in Computer Engineering*
*Sarvajanik College of Engineering and Technology*
Surat, India
patelmit640@gmail.com

***Abstract*—Today, in the technological era of the 21st century, CCTV cameras have been proven to be very fruitful in our daily lives. From monitoring the baby in the bassinet to prevent some crimes, CCTV camera has become of vital importance. We as humans, always try to make things perfect around us. Using this article, we also have attempted to present our perspective to make these CCTV cameras more perfect. We have made an effort to enhance regular CCTV cameras using the vast field of deep learning and IoT. We have attempted to accomplish our goal by providing a protoStype for the smart surveillance system. We have tried to upgrade the regular CCTV cameras with some customized deep learning models developed by us. In this modified version, we have given the CCTV cameras the ability to detect fire and weapons. Also, we have tried to fulfil an ad-hoc requirement of Face Mask Detection considering the current situation of COVID19. For fulfilling our objective, we have provided an outline combining IoT (RaspberryPi) to deep learning using AWS EC2 Cloud Architecture. To make the surveillance system user-friendly, we have also taken account of the client-side interface. Considering all the above applications, we have successfully provided an archetype in this paper.**

***Keywords—Machine Learning, Computer Vision, Artificial Intelligence, Deep Learning, Internet of Things(IOT), RaspberryPi, Smart Surveillance System***

## I. Introduction

In the 21st century, CCTV cameras have proven to be such lifesaver [12]. These CCTV cameras today have various features i.e. object detection, person detection, motion detection etc. Most of these cameras have a common approach that is:

- Input stream is taken by Camera
- Input stream saved on memory card or uploaded on cloud
- User can access input stream using Wi-Fi network

These cameras may be cost effective for daily use but these cameras with some advancement using IOT and Machine Learning can help us prevent hazardous accidents in our daily lives [1][2]. We encounter numerous cases of fire hazard in our surroundings. Considering these fire hazards, we thought of a camera that can detect fire in its sight. To achieve this purpose, a deep learning based model can be deployed on the AWS EC2 cloud architecture [7]. If fire is detected then the user will be notified using a mobile application. For preventing fire to make any deleterious results, a timer will be set after which the Fire Safety department will be notified.

Today all the crimes we encounter around us have us to something in common and that's weapons. This thought led the idea of weapon detection using CCTV cameras. For weapon detection, setup can be maintained on RaspberryPi itself using OpenCV modules. For further extension, user will be notified with an image taken by RaspberryPI using AWS Cloud. Same as before, here the nearby police station will be notified after some definite time.

Considering the recent scenario of COVID19, a face mask and social distancing is mandatory. This thought led us to a deep learning model deployed on AWS Cloud, that can detect face mask on a person's face. Input can be given to the model using RaspberryPi. User can be notified by an image when the person has not worn a mask through the mobile application. This can be further extended in numerous applications and be helpful in constraining the spread of COVID19.

## II. Related Work

Many similar papers already exist for the advancement in the surveillance system. Among these papers, many have tried to address the issues related to fire hazards, crime detection, etc. We have tried to mention some of these papers here and compare them with our approach.

The first of these different approaches is the idea of the smart home security system presented by N. Surantha and R. Wicakono[16]. They proposed a security system having various features like object detection (using Histogram of Gradient), human detection, human movement detection (using PIR sensor), and intruder detection. For intruder detection, they have developed a machine learning model using Support Vector Machine, which has an accuracy of 89%. This intruder detection may help prevent crime on some level but this will not be more accurate when the weapon will be included in the crime.

The second is the idea[18] of a low-cost smart security camera presented by W. F. Abaya, J. Basa, et al. Their objective of a security camera contains various features like night vision, human detection, and smoke detection, and in addition, they also notify the user by sending captured images via email. They have extended the proposal of smoke detection to prevent fire hazards and human detection to prevent crime. They claim accuracy of 83.56% and 83.33% for human detection and smoke detection respectively. They have used the RaspberryPi to integrate software and hardware components of the system. Comparing to their proposal, our model has an accuracy of 90.31% for fire detection. In addition, we have presented a better approach for crime prevention.

The third is the idea[17] of a smart surveillance monitoring system implemented by S. Prasad, P. Mahalakshmi, et al. They have implemented a CCTV monitoring system in which they are transmitting the images from RaspberryPi to mobile web application using the 3G dongle. They have implemented features related to intruder detection for the prevention of crime. Comparing their system, our proposal has more flexibility as it is implemented using AWS Cloud architecture. Through AWS Cloud architecture, we can improve our model for better accuracy in future and also it gives us flexibility of adding solution for any immediate situation (like Covid19).

From the various literature mentioned above, we can say that our prototype for smart surveillance system is unique and our deep learning model are much more efficient for fire detection and weapon detection. Also as mentioned, AWS Cloud architecture has given us better elasticity for easy future upgradation of our model.

## III. System Design

### A. System Architecture

As we can see in Fig. 1, the system is centrally connected through AWS Database. Input stream from the RaspberryPi will be stored in the AWS Database (DynamoDB or S3). Deep learning model situated at AWS Sagemaker will have input from the database and reflect the changes in the database. Mobile applications situated at user's end will subscribe to AWS database. If any model detects positive activity, it will notify the user.

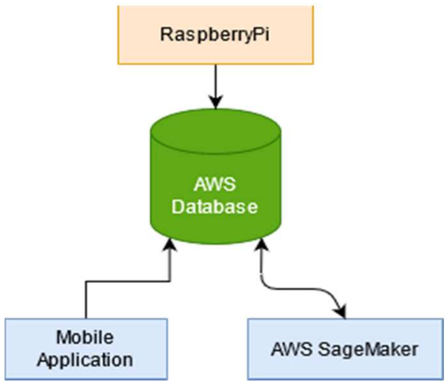


Fig. 2. Basic System Architecture

### B. Logical Flow of the System

Logical flow of the diagram is mentioned in Fig. 2. This system consists of 8 important components, which are elaborated below.

1. The first aspect of the system is to authenticate the raspberry pi on AWS EC2 Cloud Architecture. User is given a public key and a private key through which Raspberry Pi is authenticated using AWS IOT Core SDK on AWS cloud [10]. After successful authentication, Raspberry PI will be able to communicate with AWS through MQTT broker.
2. After successful authentication, the data will be sent from raspberry pi to aws database. The video data from the surrounding is received by the Raspberry Pi camera module as frames of images. These frames are then preprocessed using OpenCV modules which are there on RaspberryPi. After preprocessing they are prepared to be given as input to the deep learning situated at AWS cloud.
3. The preprocessed frames by OpenCV are then sent to a database located on AWS Cloud using MQTT broker. There are 2 options available for choosing Databases DynamoDB and S3 [9]. While S3 databases have proven to be more efficient while working with image data, either of the 2 can be chosen according to requirement. Database on the cloud will be updated continuously using the data from the RaspberryPi. So these databases play a vital role in the whole system.
4. AWS SageMaker [8] is the facility for utilizing and deploying pretrained deep learning models on AWS EC2 cloud architecture. Custom models deployed on AWS SageMaker will continuously check images from the database and will update accordingly to the database.
5. We have developed customized models for detection of fire and facemask. For developing customized models, AWS provides facility of AWS Docker Container [14]. Customized models created using Docker containers will be deployed on AWS SageMaker for runtime tasks.
6. AWS Buckets are the storage devices for our deep learning models. These buckets will contain datasets for training and testing the customized models developed using AWS Docker Container.

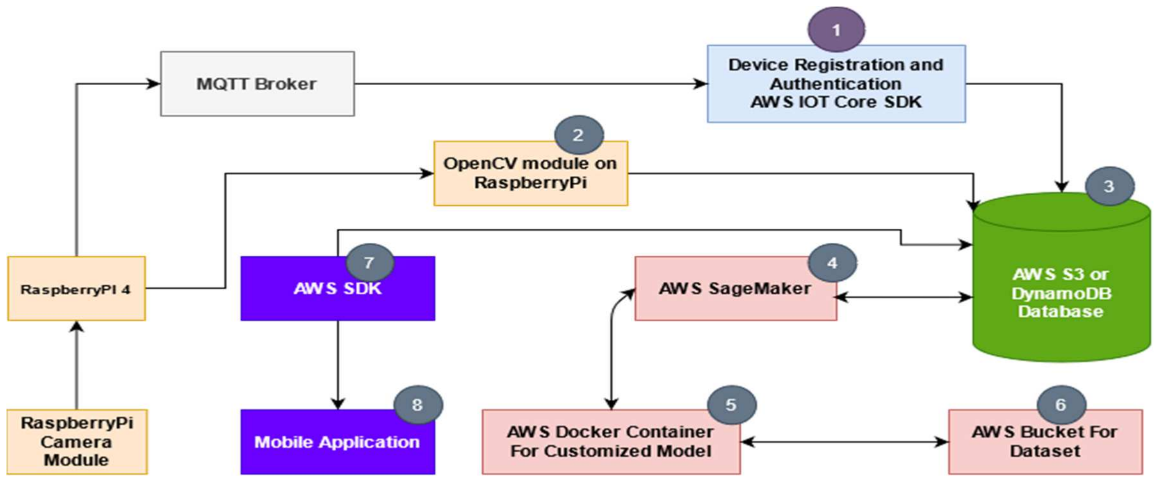

1
Device Registration and Authentication AWS IOT Core SDK
MQTT Broker
2
OpenCV module on RaspberryPi
3
AWS S3 or DynamoDB Database
RaspberryPI 4
7
AWS SDK
4
AWS SageMaker
RaspberryPi Camera Module
8
Mobile Application
5
AWS Docker Container For Customized Model
6
AWS Bucket For Dataset

Fig. 1. Logical Flow of the System

7. AWS SDK [15] is the facility provided by AWS for connecting our whole system with a mobile application. AWS SDK will provide us for implementing an interface on different platforms having different Operating Systems.
8. A mobile application is a client-side interface that will be subscribed to an AWS database. This interface will notify the user when models positively detect something. Also, this application will send the user images taken by the RaspberryPi camera module.

## IV. METHODOLOGY

After getting the glance of the overall system design, in this section structure and implementation of models for different tasks is elucidated.

### A. *Convolution Neural Network*

The Computational Neural Network (CNN) is a class of deep learning classification and the heart of the emerging field of computer vision. They are widely used to interpret and analyze information from image data. CNN fulfills these objectives by detecting a pattern in the images used for training the deep learning model. There are five required components for building a CNN model:

1. The Input Layer as in the Fig. 3 is the first layer during model development which takes the input of image data in the form of an array. The array here consists of the form [height, width, dimension], where height and width are of the images. Here dimension, differs based on type of images used for training. For instance, for black and white images, the dimension value is 1 and for RGB images the value is 3.
2. The computational layer (Convolution layer 1,2, etc. in Fig. 3) is the core of the building block of CNN, where the majority of the computations occur. It consists of 3 components: input data (as matrix), a filter(kernel) matrix, and a feature map. The kernel is applied to an equivalent portion of the image and a dot multiplication is calculated between that input submatrix and the kernel. The output of this dot product is then mapped into a feature matrix. Here the kernel starts computing dot product with an equal submatrix starting from upper left corner and then shifts by a stride (a predefined value) and it continues until it covers the entire input image. Here the dimension of feature map will be [width, height, number of kernels].
3. The next layer is Activation Function layer. Its purpose is to break linearity in the features. This layer applies element wise activation function to the output of convolution layer. The most common activation function is RELU (REctified Linear Unit), mentioned in Fig. 4.

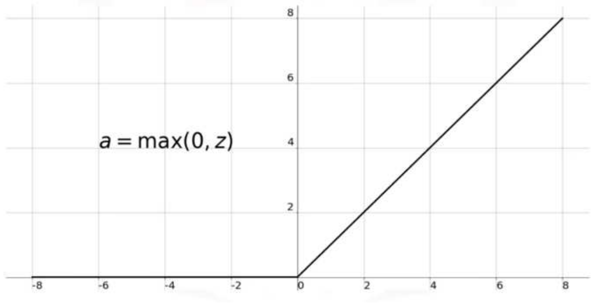


Fig. 4. RELU Activation Function

4. The next layer is Pooling Layer or Downsampling layer, which is used to perform dimensionality reduction, reducing the number of parameter in the input. It is used to make computation fast, reduce memory and also prevent model from overfitting. Here, n*n filter is moved over feature map, the filter then applies aggregation function to values in the respective field, populating it into the output matrix. The most widely used aggregation functions are Max Pooling and Average Pooling. In each of the case, respectively maximum value and average of the filter values are mapped into output value. The dimension of the output matrix is [height/n, width/n, number of convolution filters]. With every convolution, there is an activation layer and pooling layer associated in the CNN architecture.
5. The last component is Fully Connected Layer. Output of the last pooling layer which may be multidimensional matrix, is first converted into a unidimensional matrix. Now this is given to input to fully connected component, which is just a simple Artificial Neural Network. Its main purpose is to minimize the loss using backpropagation algorithm

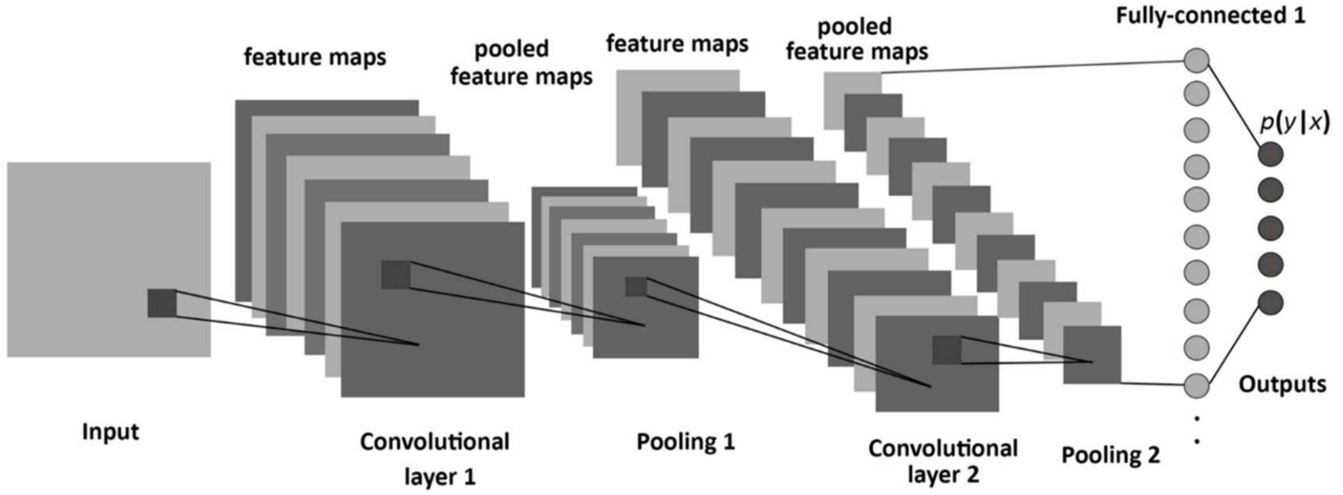


Fig. 3. Architecture of Multi CNN model

and maximize the probability of the detected object.

## B. *Implementation for Fire detection*

Fire detection using this system is elaborated using the diagram mentioned in Fig. 5. As mentioned above, Raspberry Pi will take input from the surroundings. After pre-processing it using OpenCV, the image will be stored on AWS Database, later this image will be given to model as input and it will detect the fire in the image. Also there will be a global variable whose value will be changed when fire will be detected. The mobile application will send an image from the database to the user when the value of the variable is change.

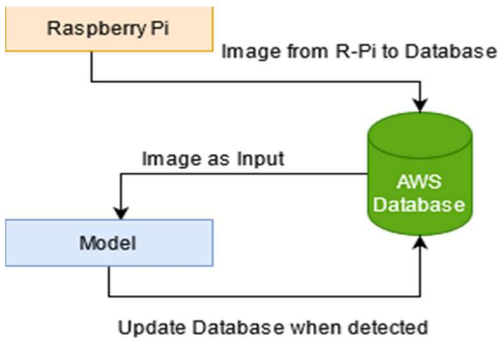


Fig. 5. Logical Implementation for Fire Detection

After considering the writings [3],[4], we have implemented a customized model along with a pretrained InceptionV3 model. We have trained our model on around 1374 images[13][22]. The structure of the model is mentioned in Fig. 6.

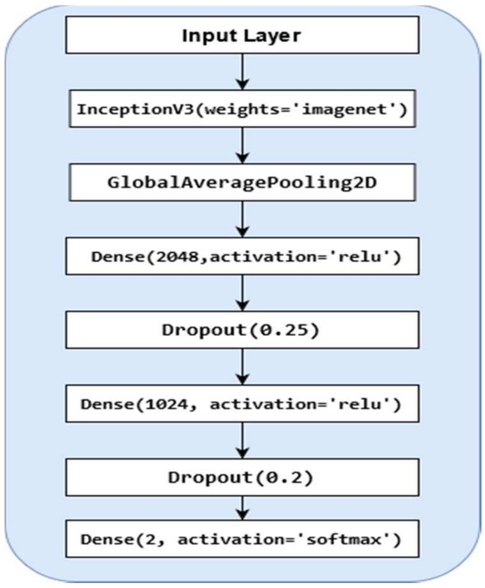

Input Layer
InceptionV3(weights='imagenet')
GlobalAveragePooling2D
Dense(2048,activation='relu')
Dropout(0.25)
Dense(1024, activation='relu')
Dropout(0.2)
Dense(2, activation='softmax')

Fig. 6. Fire Detection Model Structure

## C. *Implementation of Weapon Detection*

For weapon detection, we have developed an OpenCV cascade that can detect weapons[11]. We have trained this cascade using 1200 images[19]. Weapon detection will be done on RaspberryPi itself. In this system, video stream will be taken as frames by RaspberryPi. This frame will then be pre-processed using OpenCV for the deep learning model. At this time, weapon detection will also happen simultaneously. If a weapon is detected then the value of the corresponding variable in the database will be changed. This change in database will prompt the mobile application to notify user with image from the RaspberryPi.

## D. *Implementation of Face Mask Detection*

Considering the current situation of COVID19, we also like to include this special ad-hoc feature into our smart surveillance system. This application can be extended in many directions by integrating it with smart door lock in shops and malls, schools etc.

After consideration of some references [5][6], we have implemented a customized model along with a pretrained MobileNetV2 base using deep learning. We have trained this model on 3000 images[20][21]. The structure of our model is elaborated in the Fig. 7.

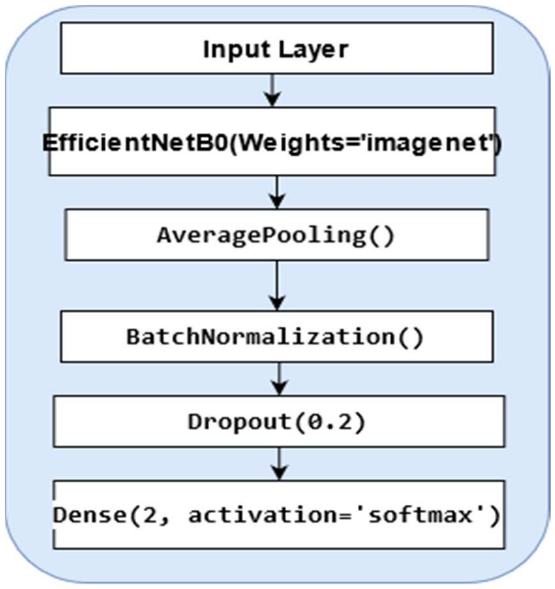

Input Layer
EfficientNetB0(Weights='imagenet')
AveragePooling()
BatchNormalization()
Dropout(0.2)
Dense(2, activation='softmax')

Fig. 7. Face Mask Detection Model Structure

## V. RESULTS AND ANALYSIS

In this section, we have mentioned the accuracy of our above-mentioned models and also, we have included some real time predictions made by above mentioned deep learning models.

## A. *Verification of Deep Learning Models*

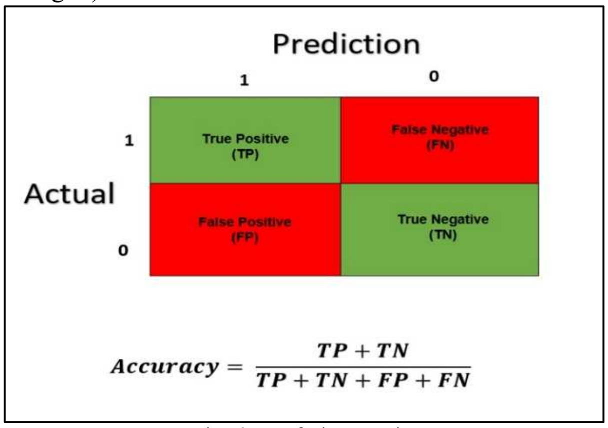
To measure how our model works on testing data, we have decided to use "Accuracy" metric. Accuracy of any model can be calculated from the confusion matrix.

Confusion matrix also known as Error matrix, is used to describe the performance of classification model. In confusion matrix there are basically 4 terminologies, which are:

- *True Positive* – Number of positive predictions done efficiently by the model. For e.g., number of instances when fire is there in the picture and model detects it.
- *True Negative* – Number of negative predictions made efficiently by the model. For e.g., number of instances when fire is not there in the picture and model does not detect it.
- *False Positive* (Type - 1 error) – Number of negative instances which are predicted positively by the model. For e.g., number of occurrences when fire is not there in the picture and model detects fire in it.

- *False Negative* (Type – 2 error) – Number of positive instances which are predicted negatively by the model. For e.g., number of occurrences when fire is there in the picture but model fails to detect it.

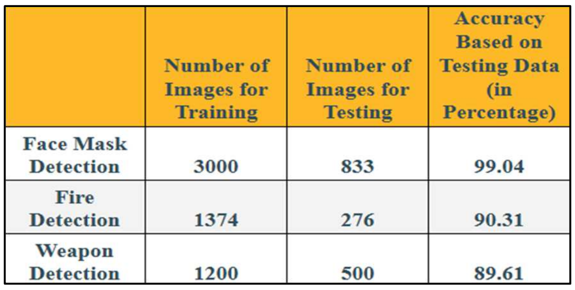
From these confusion matrix, one can compute various metrics for verification of the model like accuracy, recall, precision and f1-score. Here we have decided to use accuracy as our verification metric. Accuracy is the ratio of the number of instances predicted efficiently by the model to total number of instances (Formula is mentioned in Fig. 8).



$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Fig. 8. Confusion Matrix

Accuracies of our deep learning models are mentioned below in Fig. 9.



| | Number of Images for Training | Number of Images for Testing | Accuracy Based on Testing Data (in Percentage) |
|---|---|---|---|
| Face Mask Detection | 3000 | 833 | 99.04 |
| Fire Detection | 1374 | 276 | 90.31 |
| Weapon Detection | 1200 | 500 | 89.61 |

Fig. 9. Accuracy Of Our Deep Learning Models

## B. Real Time Detection

Here are the examples or real time detection of weapons mentioned in Fig. 10. As We can see there are no weapons in the 1st image so "Not Found" is written in the left bottom corner. While there are guns detected in the 2nd image so "Gun Detected" is written in the left bottom corner.

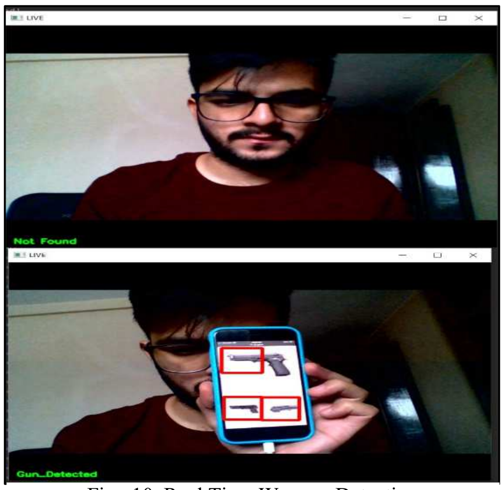


Fig. 10. Real Time Weapon Detection

Here is the real time demo of fire detection mentioned in Fig. 11.

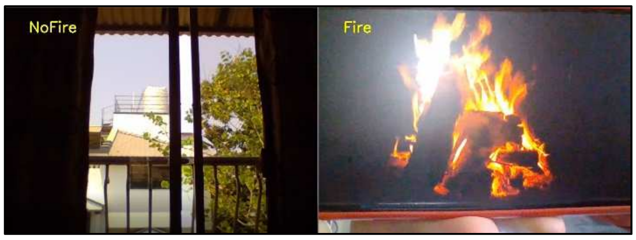


Fig. 11. Real Time Fire Detection

Here is the real-time face mask detection demo mentioned in Fig. 12. As one can see, there is no mask detected in the left photo while in the right photo, the model has detected it successfully.

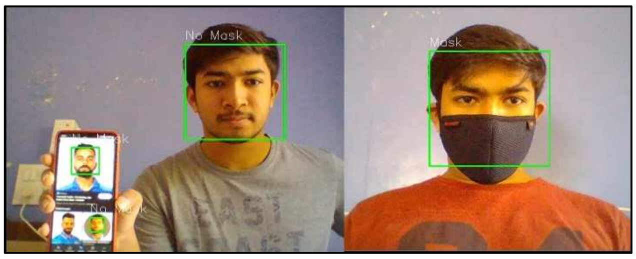


Fig. 12. Real Time Face Mask Detection

## VI. CONCLUSION AND FUTURE WORKS

Throughout this paper, our perspective has been to enhance the capabilities of our current surveillance system and make it more beneficial. Though this advanced surveillance system may cost a little more than regular CCTV cameras, it can surely outweigh the commonplace CCTV cameras in functionality.

Also, our sole purpose of using AWS Cloud architecture is to give more flexibility for future advancements. One can easily add more features i.e. object detection, family member detection (for home), etc. One can also easily upgrade existing deep learning models for fire, facemask detection without interrupting the whole system. As we have used AWS cloud, one can also enhance the surveillance system using Amazon voice assistant Alexa. In addition, one can easily implement any ad-hoc requirement using AWS Cloud architecture (As we have done for Face Mask Detection). So, we have successfully presented a prototype that can be extended in many directions as needed.

## REFERENCES

[1] S. D. T. Kelly, N. K. Suryadevara and S. C. Mukhopadhyay, "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes," in *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3846-3853, Oct. 2013, doi: 10.1109/JSEN.2013.2263379.

[2] B. Rao, R. Sudheer, "Surveillance Camera using IOT and Raspberry Pi", Aug. 2019.

[3] M. Khan, J. Ahmed, SW. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management", Dec. 2017, doi: 10.1016/j.neucom.2017.04.083

[4] P. Foggia, A. Saggese, M. Vento, "Real-Time Fire Detection for Video-Surveillance Applications Using a Combination of Experts

Based on Color, Shape, and Motion", Jan. 2015, doi: 10.1109/TCSVT.2015.2392531

[5] GJ. Chowdary, NS. Punn, SK. Sonbhadra, S. Agarwal, "Face Mask Detection Using Transfer Learning of InceptionV3", Jan. 03 2021.

[6] I. Ahmed, M. Ahmed, JJPC. Rodrigues, G. Jeon, S. Dinf, "A deep learning-based social distance monitoring framework for COVID-19",Nov. 2020, doi: 0.1016/j.scs.2020.102571

[7] Amazon, What is Amazon EC2?, Accessed On: Feb. 26 2021 Available: https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/concepts.html .

[8] Amazon, What is Amazon SageMaker?, Accessed On: Mar. 1 2021.Available: https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html .

[9] Amazon, What is Amazon S3?, Accessed On: Mar. 1 2021.Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html .

[10] Amazon, What is Amazon IOT Core?, Accessed On: Feb. 22 2021.Available: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

[11] Skyra18, Gun Detection using OpenCV, Accessed On: Jan. 24 2021, Available : https://www.geeksforgeeks.org/gun-detection-using-python-opencv/

[12] MPJ. Ashby, "The Value of CCTV Surveillance Cameras as an Investigative Tool: An Empirical Analysis", Sept. 2017, doi: 10.1007/s10610-017-9341-6

[13] Fire Detection Dataset. 28 Oct. 2019, Accessed On: Jan. 2021, kaggle.com/atulyakumar98/test-dataset.

[14] Amazon, How To Deploy Docker Containers?, Accessed On: Feb. 16 2021.Available: https://docs.aws.amazon.com/sagemaker/latest/dg/docker-containers.html

[15] Amazon, Using the AWS Mobile SDKs for IOS and Android, Accessed On: Feb. 14 2021.Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-mobile-sdks.html

[16] N. Surantha, R. Wicaksono, "Design of Smart Home Security System using Object Recognition and PIR Sensor", Aug. 2019, doi: 10.1016/j.procs.2018.08.198

[17] S. Prasad, P. Mahalakshmi, AJC. Sunder, R. Swathi, "Smart Surveillance Monitoring System Using Raspberry PI and PIR Sensor", Aug. 2019, doi: 10.5120/ijca2018916889

[18] W. F. Abaya, J. Basa, M. Sy, A. C. Abad and E. P. Dadios, "Low cost smart security camera with night vision capability using Raspberry Pi and OpenCV," *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2014, pp. 1-6, doi: 10.1109/HNICEM.2014.7016253.

[19] Weapon_detection_dataset. 2 Jan. 2020, Accessed On: Jan. 2021, kaggle.com/abhishek4273/gun-detection-dataset.

[20] Face Mask Detection. 22 May 2020, Accessed On: Jan. 2021, kaggle.com/andrewmvd/face-mask-detection.

[21] Face Mask Detection Dataset. 31 July 2020, Accessed On: Jan. 2021, kaggle.com/omkargurav/face-mask-dataset.

[22] FIRE Dataset. 25 Feb. 2020, Accessed On: Jan. 2021, kaggle.com/phylake1337/fire-dataset.