Smart Fire Alarm

Akshay Kokane Anubhav Nigam

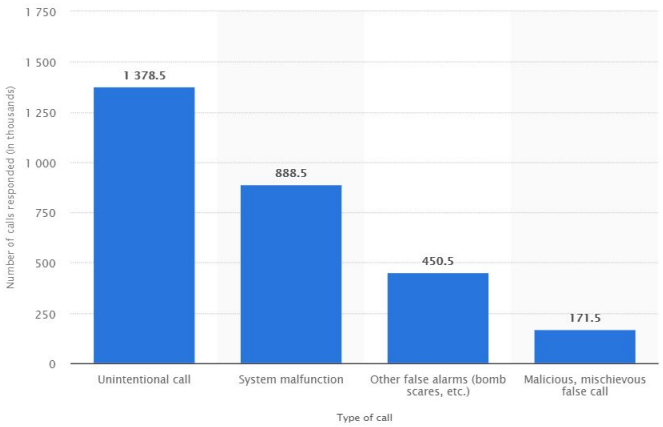
I. Abstract

Fires are very common things happening in the world. Every 24 seconds, a fire department in the United States responds to a fire somewhere in the nation. Safety and security have always been the prime concern of mankind. Society still depends on smoke alarms for the timely detection of fire. Though detecting smoke might be a good indicator of potential fire, a machine learning-based approach might bring significant improvements to this existing technology. Sensors usually have a short range of detection. Smoke alarms also face the problem of false detections, which results in authorities taking the necessary mitigating steps without the actual occurrence of the event. This causes wastage of resources, time and energy. Even though fire alarms could be found in almost all commercial buildings nowadays it still remains out of reach of the civilians due to the cost associated with these systems. So, we came up with a cost-effective and reliable approach for fire detection. Our aim was to build a reliable and efficient smart fire detection system and to alert the residents, responsible authorities as early as possible so that the casualties could be minimized. With so many advancements in the fields of machine learning, computer vision, cloud infrastructure, it provided us with an opportunity to leverage these techniques to address the aforementioned concerns. To tap the potentials of cloud infrastructures and machine learning GPUs to detect the fire and implement the fire detection system using a computer vision approach where we used two steps approach to detect fire with the help of IOT device and deep learning models. At first stage we kept a threshold to detect fire at Raspberry Pi with continuous stream of live video taken through Pi Cam. Once the possibility of fire reaches a well defined threshold, frame is captured and images taken it fed for second stage of detection thus reducing the false alarm rate. Once our second stage heavy weight model confirms fire, user is notified along with the images of fire with the help of Amazon web services. This system of fire detection ensures high performance and efficient detection of fire while keeping the false alarm rate low.

II. Introduction

There has been alarmingly rapid increase in the loss of life and property due to fire disaster. Fire in its most common form can result in conflagration, which has the potential to cause physical damage through burning. If we talk about home fires, it just takes two minutes for a fire to become life-threatening and in five minutes, a residence can be engulfed in flames. Statistics show that in every 24 seconds, a fire department in the United States responds to a fire somewhere in the nation. But these fires are very much preventable. Fire accidents are the most commonly occurring disasters nowadays. In order to mitigate the number of fire accidents, a large number of methods have been proposed for early fire detection to reduce the damage caused by such accidents.

Apart from the problem of early fire detection, present fire alarm systems also prove to be inefficient in terms of the false triggering of the alarm systems.



This statistic shows the total number of false alarms responded to by fire departments in the United States in 2018. In 2018, U.S. fire departments responded to a total of 2,889,000 false alarms. Malicious false calls increased by 22% from 2017, accounting for 171,500 of all false calls. Present fire detection methods are typically based on physical sensors like thermal detectors, smoke detectors, and flame detectors. However, these sensor-based detection systems are not very reliable for fire detection. For instance, quite often there are cases of false triggering with smoke detectors, as it does not possess the capability to differentiate between fire and smoke. This can also be supported by the stats provided by US Fire Department. While thermal detectors and flame detectors requires sufficient level of fire initiation for a clear detection, which leads to a long detection delay causing irreparable damages. An alternative, which could lead to the enhancement of robustness and reliability in the present fire detection systems, is the visual fire detection approach.

There have been so many advancements in the field of machine learning, computer vision, cloud infrastructure, it provides us with an opportunity to leverage these techniques to address the early fire detection. Various deep learning models have been able to comfortably surpass the human level performance in specific computer vision applications like image classification. The visual-based fire detection approach also has been able to take advantage of these technological

improvements. Visual based fire detection systems have many advantages over hardware-based alarm systems in terms of cost, accuracy, robustness, and reliability. With better performance, visual based fire detection can significantly improve the false alarm rate as well as early detection of fire. This approach is more cost effective and can significantly decrease the loss of life and property. At present there are many models that have been trained to detect fire as early as possible. These models takes the leverage of deep learning based approaches. Therefore, in this work, our aim is to introduce an advanced fire and smoke detection unit, which is reliable, reduces the false triggering problem and incorporates various state of the art technological concepts like Convolutional Neural Networks (CNN) and the Internet of Things (IoT). In order to get the best fire detection performance while maintaining a significantly good frame rate on the Raspberry Pi 3B (1.2GHz Broadcom BCM2837 64bit CPU, 4 GB RAM computer]) during real-time fire detection, we have taken advantage of already present fire detection light weight model and incorporated one more better performant fire detection model on cloud and built a system where user/concerned department can be notified of fire along with potential fire images on smartphones. Users can also be notified on smart speakers like Google Home or Amazon's Echo. Two steps fire detection, first model deployed on Raspberry Pi and second model deployed on cloud, has shown better results in terms of false alarm reduction.

III. DATASET

We did not go at any depth for the dataset as their were already pre trained models on various datasets which we could have used. But for the reference we used two pre-trained model which was based on an annotated dataset used by the authors of the research paper based on fire detection [1] and one private dataset from the owner of GitHub repository[2]. For the model which we used on the Raspberry-Pi, author trained his light weight fire detection model using a self created diverse dataset with images randomly sampled from their self shot fire and non fire videos. The final train dataset consists of a total of 1,124 fire images and 1,301 non-fire images. Although the dataset may appear to be small, it is extremely diverse. While for testing the model they used Foggia's dataset which is open source and is available at [3]. Their complete test dataset consists of 46 fire videos (19,094 frames) with 16 non-fire videos (6,747 frames) and additional 160 challenging non-fire images. To make their test dataset diverse, out of all the frames extracted from this dataset, they randomly sampled few images from each video to form their final test dataset to be used in this work. While they also tested their model with previously held out samples. Another model which we used on our cloud end was FireNet which was open source and was available to be used for research purposes. FireNet is a real-time fire detection project containing an annotated dataset, pre-trained models and inference codes, all created to ensure that machine learning systems can be trained to detect fires instantly and eliminate false alerts. This was part of DeepQuest AI's to train machine learning systems to perceive, understand and act accordingly in solving problems in any environment they are deployed. It contained an annotated dataset of 502 images split into 412 images for training and 90 images for validation. This dataset is one of the Foggia's datasets [3].

Images from training dataset

IV. APPROACH

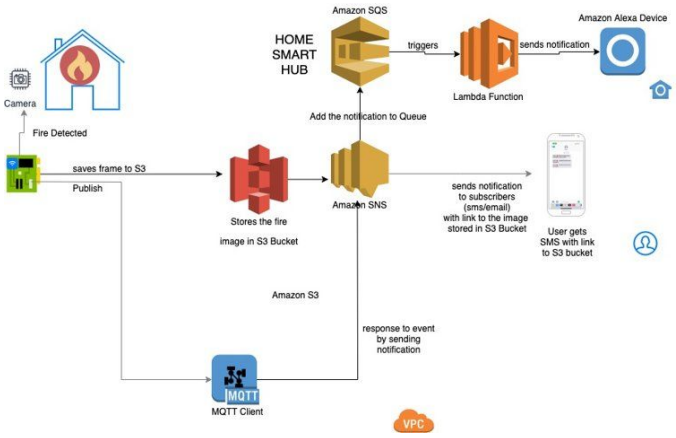
Our approach is to implement the fire detection technique in two steps. For the first step, we used a pre trained model, a light-weight model deployed on raspberry-pi, to detect the fire as a precursor to the second heavy weight model. Once this model would pass the inference as positive result we would send the data to our second heavy weight model deployed on cloud to confirm the occurrence of the fire. This approach was preferred as light weight fire detection model deployed on Raspberry Pi has the biggest advantage of its small size on disk (~7.45 MB) which is attributed to the shallow network, and consequently the small number of trainable parameters (646,818). On the downside, this model was less accurate and was detecting fire even when there was no fire which resulted in false alarm for fire. But this model detected fire almost every time when there was a fire. Hence, a threshold was set to freeze the frame to collect the image and this image of suspected fire was further sent to our second heavy weight

model for confirmation. This model was preferred because of its lightweight and easy deployment on IOT devices.

Our second heavy weight model deployed on cloud is a YOLOv3 detection model on the images and perform detection in images and videos using a pre-trained model. This was a better performant heavy weight model. Hence suspected fire image coming from Raspberry Pi is further feed as an input to this model to double check the detection of fire. This was done because of the following reason. 1) Light weight model deployed on Raspberry Pi was less reliable and gave more false alarm. 2) Heavy weight model could not be deployed on Raspberry Pi because of less computational power. 3) In order to reduce the latency for detecting fire, a threshold was set on the first model and only suspected image gathered from it was further tested on heavy weight model to confirm fire.

Once the fire was confirmed by model, user will be notified using Amazon web services (Amazon SNS) on his smartphone along with the image link stored at Amazon S3 bucket. Further user will be notified on smart speakers such as Amazon Echo or Google Home.

V. ARCHITECTURE



The complete architecture diagram of the proposed network is shown in Fig. 3, from where it can be seen that Smart Fire System consists of three major layers. 1) IOT device 2) Amazon Web Service 3) Smart devices to receive notification.

For the first layer, we are using Raspberry Pi Model B+ quad-core Cortex-A72 (ARM v8) 64-bit SoC @1.5GHz 4GB LPDDR4-2400 SDRAM) and Raspberry Pi Mini Camera (Video Module 5 Megapixels 1080p Sensor OV5647 Webcam). A light weight pre- trained deep learning model is deployed on Raspberry Pi. With Pi Cam, live video of the area is streamed to the model. Our second layer consist of the Amazon web services which includes S3 bucket for storing images captured from first light weight model deployed on Raspberry Pi, Amazon Simple Notification Service which uses MQTT protocol and sends notification to users. Our third layer consists of smart devices to inform users about the fire. These smart devices include smartphones, smart speakers like Amazon Echo and Google Home.

VI. IMPLEMENTATION

The implementation of this project was divided into two phases. In the first phase, using a camera which is connected to the raspberry pi we captured real time videos and send it to the model stored in raspberry pi for fire detection. For this step data pre-processing was required, first of all input frames are resized. We feed this input stream to model, once the fire prediction reaches 99%, frame is uploaded where fire was detected. This frame is sent to Amazon S3 bucket while message is sent with the image name to second model deployed on Amazon server for second phase of the project using MQTT protocol. Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is cost effective place to store data. While we have MQTT Client subscriber which is always listening to an event from Raspberry Pi. Once the images are saved on S3 instance we send these images to our YOLOv3 model stored in an Amazon EC2 instance to confirm the results. Message is received along with image URL stored on Amazon S3 bucket. This image is uploaded from bucket as input feed to model to get prediction. If our model gives positive results we send notifications to the user using Amazon SNS service. Amazon Simple Notification Service is publish/subscribe messaging paradigm. In such a system, publishers and receivers of messages are decoupled and unaware of each other's existence. The receivers (also known as subscribers) express interest in certain topics. The senders (publishers) can send a message to a topic. The message will then be immediately delivered or pushed to all of the subscribers to the topic. In our case topic or message is sent to smartphones and smart speakers who are subscribed to this service. The notification would be sent on mobile of the user with link of images stored in S3 bucket, Amazon Echo using the lambda function and also on google home depending on the availability of these.

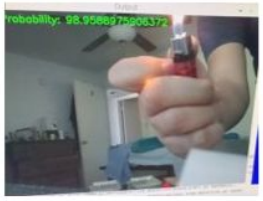
For getting notification on Amazon Echo, there is a way to build the cloud-based service for a custom Alexa skill which is to use AWS Lambda, an Amazon Web Services offering that runs your code only when it's needed and scales automatically, so there is no need to provision or continuously run servers. We can upload the code for our Alexa skill to detect fire using a Lambda function and Lambda does the rest, executing it in response to Alexa voice interactions and automatically managing the compute resources for us. Once the fire is detected, notification is added to the queue to Amazon SQS.

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables us to decouple and scale microservices, distributed systems, and serverless applications. SQS eliminates the complexity and overhead associated with managing and operating message oriented middleware, and empowers developers to focus on differentiating work. It triggers the Lambda Function where we have already defined a skill for notifying user through Alexa. This part is still under development as current functionality of Alexa does not support notifying user without any voice input. Even for defining Using these notifications user can respond to the situation even if they are not on the site of the occurrence of the event and inform the concerned authorities to take the necessary steps to mitigate the fire timely.

VII. RESULTS

The aim of this work is to present a method that can be smoothly deployed to an embedded device in order to finally build a complete fire detection unit. While using an additional model at Amazon cloud to verify the results obtained from model deployed at Raspberry Pi. By using this type of system can ensure high performance in detecting fire and helps in reducing false positive alarm at great extent. This gives an extra cushion for users to be sure, they are notified for fire along with additional link to check the images of the predicted fire and necessary steps could be taken as early as possible. Even the concerned authority can be informed along with the images of fire at earliest. This visual based fire detection approach based on deep machine learning can significantly decrease the false alarm rate and hence it can become cost effective and fire department be more productive, Therefore, it becomes inevitable to use a test dataset that includes images that are often encountered in real-world fire emergencies with an image quality that is commonly obtained with a camera attached.

Therefore, to reflect the performance of our trained model in most realistic situations, we compiled the test dataset from our own videos shot in challenging real world fire and non-fire environment. Let's see some expected and actual results obtained by our system.



Expected : Fire Detected but below threshold

Actual : Fire Detected above threshold

Expected : Fire Detected above threshold

Actual : Fire Detected above threshold

Expected : Fire Detected but below threshold

Actual : Fire Detected but below threshold

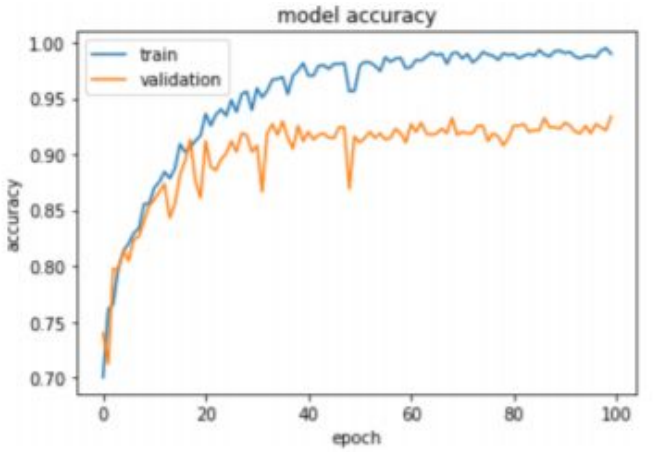
Expected : No Fire Detected

Actual : No Fire Detected

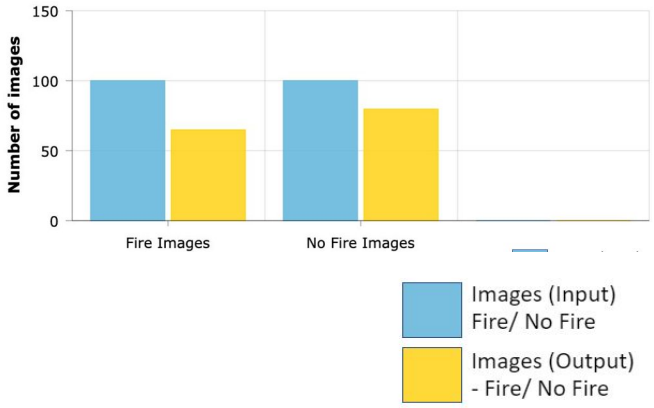
Above four images shows our few test results. Image 1 over here expected a fire detection which should be below threshold as we just used an image with lighter flame which is not going to cause any fire. Still our system predicts it as fire and it was above threshold. Since the area covered by lighter flame is still high and its image will be stored in Amazon S3 bucket and user will be informed about this classified fire. But with an option for user to look at image which will give clear view. Similarly for Image 2 and 3, Fire detected above threshold and fire detected below threshold was expected respectively. Our model predicted the accurate results for these cases. In Image 4, which clearly shows that there are no signs of fire occurrence and this can be seconded by our system's model prediction.

Training and testing model for detecting fire plays an important role in our fire detection system. Since we are continuously providing video feed to our light weight model deployed at Raspberry Pi, accuracy of model should be high to ensure that we are accurately determining fire. Below are some test results for model deployed at edge server end.

Metrices	Our dataset (%)	Foggia's dataset (%)
Accuracy	**93.91**	**96.53**
False Positives	1.95	1.23
False Negatives	4.13	2.25
Recall	94	97.46
Precision	97	95.54
F-measure	95	96.49



While accuracy of the pre trained model at cloud was also checked to ensure better result. We supplied 100 random images of fire to this heavy weight model, out of which 67 were detected of fire. Then again 100 random images of no fire were fed to model, out of which 83 were accurately predicted. Since these results are good for reducing false alarm rate but model needs to be trained more for better accuracy.



VIII. CONCLUSION

The two step process ensures the credibility of our model in detecting fires. The model was successfully deployed on Amazon Cloud Service in conjunction with the model deployed locally. The system informs the user of the fire well within time, to take the desired mitigating action and alleviate the casualties caused by it. Our model also decreases the occurrence of the false alarms thereby helping the concerned authorities in decreasing their workload.

IX. FUTURE WORK

This project could be extended to inform the concerned authority directly apart from informing the users. This would ensure that the user would not have to inform the authorities on their own and our service would do this for them. Since in cases of emergencies a single second saved could save many lives. This could be a good field to work on to improve this project.

Alexa and Google home currently do not have any notifications service available using which any notification could be sent to the user. There are some state of the art carrying out this but those could not be trusted. Both Amazon and Google are working on providing such services, once these are released it could be used to send the additional notifications to the users.

Various pre trained models could be used as trial purpose for the first phase of the project as an experiment to observe the results respectively. A study could be done on them and a better pre trained model might be deployed for the first phase of the project.

By integrating thermal imaging with machine learning approach, we can further reduce false positive alarms. While buzzers can also be used, connected to pi which will ring once the fire is detected.

X. REFERENCES

1)https://github.com/OlafenwaMoses/FireNET

2)https://github.com/arpit-jadon/FireNet-LightWeight-Network-for-Fire-Detection

3)https://mivia.unisa.it/datasets/video-analysis-datasets/fire-detection-dataset/

4)https://github.com/OlafenwaMoses/FireNET/releases

5)https://aws.amazon.com/blogs/aws/