CONVOLUTIONAL NEURAL NETWORKS BASED FIRE DETECTION IN SURVEILLANCE VIDEOS

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

STERIN JOLLY

Department of Computer Science & Engineering
Mar Athanasius College Of Engineering Kothamangalam

CONVOLUTIONAL NEURAL NETWORKS BASED FIRE DETECTION IN SURVEILLANCE VIDEOS

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

STERIN JOLLY

KNOWLEDGE IS POWER

Department of Computer Science & Engineering
Mar Athanasius College Of Engineering Kothamangalam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **Convolutional Neural Networks Based Fire Detection In surveillance Videos** *submitted by* ***Mr STERIN JOLLY*** *, Reg. No.* ***MAC15CS058****, towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer science and Engineering from APJ Abdul Kalam Technological University for December 2018 is a bonafide record of the seminar carried out by him under our supervision and guidance.*

....	....	....
Prof. Joby George	**Prof. Neethu Subash**	**Dr. Surekha Mariam Varghese**
Faculty Guide	***Faculty Guide***	***Head of the Department***

Date: Dept. Seal

ACKNOWLEDGEMENT

First and foremost, I sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the seminar.

I express my sincere gratitude and thanks to Dr. Solly George T, Principal and Dr.Surekha Mariam Varghese, Head of the Department for providing the necessary facilities and their encouragement and support

I owe special thanks to the staff in charge Mr.Joby George, Mrs.Neethu Subash and Mrs. Joby for their corrections, suggestions and sincere efforts to coordinate the seminar under a right schedule

I express my sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar.

Finally I would like to acknowledge the heart felt efforts, comments, criticisms, cooperation and tremendous support given to me by my dear friends during the preparation of the seminar and also during the presentation without whose support this would have been all the more difficult to accomplish.

ABSTRACT

The recent advances in embedded processing have enabled the vision based systems to detect fire during surveillance using convolutional neural networks (CNNs). However, such methods generally need more computational time and memory, restricting its implementation in surveillance networks. In this research paper, we propose a cost-effective fire detection CNN architecture for surveillance videos. The model is inspired from GoogleNet architecture, considering its reasonable computational complexity and suitability for the intended problem compared to other computationally expensive networks such as AlexNet. To balance the efficiency and accuracy, the model is fine-tuned considering the nature of the target problem and fire data. Experimental results on benchmark fire datasets reveal the effectiveness of the proposed framework and validate its suitability for fire detection in CCTV surveillance systems compared to state-of-the-art methods.

Contents

List of Figures

List of abbreviations

DL	:	Deep Learning
CNN	:	Convolutional Neural Network
RGB	:	Red Green Blue
HSI	:	Hue Saturation Intensity
CCTV	:	Closed Circuit Television
NFPA	:	National Fire Protection Association
MNIST	:	Modified National Institute of Standards and Technology database
ILSVRC	:	ImageNet Large Scale Visual Recognition Challenge

Introduction

The increased embedded processing capabilities of smart devices have resulted in smarter surveillance, providing a number of useful applications in different domains such as e-health, autonomous driving, and event monitoring. During surveillance, different abnormal events can occur such as fire, accidents, disaster, medical emergency, fight, and flood about which getting early information is important. This can greatly minimize the chances of big disasters and can control an abnormal event on time with comparatively minimum possible loss. Among such abnormal events, fire is one of the commonly happening events, whose detection at early stages during surveillance can avoid home fires and fire disasters .

Besides other fatal factors of home fires, physical disability is the secondly ranked factor which affected 15 percentage of the home fire victims. According to NFPA report 2015, a total of 1345500 fires occurred in only US, resulted in 14.3 billion loss, 15700 civilian fire injuries, and 3280 civilian fire fatalities. In addition, a civilian fire injury and death occurred every 33.5 minutes and 160 minutes, respectively. Among the fire deaths, 78 percentage occurred only due to home fires.

One of the main reasons is the delayed escape for disabled people as the traditional fire alarming systems need strong fires or close proximity, failing to generate an alarm on time for such people. This necessitates the existence of effective fire alarming systems for surveillance. To date, most of the fire alarming systems are developed based on vision sensors, considering its affordable cost and installation. As a result, majority of the research is conducted for fire detection using cameras.

Considering the limitations of traditional hand-engineering methods, we extensively studied deep learning (DL) architectures for this problem and propose a cost-effeuctive CNN framework for flame detection in CCTV surveillance videos. Our framework avoids the tedious and time consuming process of feature engineering and automatically learns rich features from raw fire data.

Inspired from transfer learning strategies, we trained and fine-tuned a model with architecture similar to GoogleNet for fire detection, which successfully dominated traditional fire detection schemes.

The proposed framework balances the fire detection accuracy and computational complexity as well as reduces the number of false warnings compared to state-of-the-art fire detection schemes.

Existing methods

The available literature dictates that flame detection using visible light camera is the generally used fire detection method, which has three categories including pixel-level, blob-level, and patch-level methods. The pixel-level methods are fast due to usage of pixel-wise features such as colors and flickers, however, their performance is not attractive as such methods can be easily biased. Compared to pixel-level methods, blob-level flame detection methods show better performance as such methods consider blob-level candidates for features extraction to detect flame. The major problem with such methods is the difficulty in training their classifiers due to numerous shapes of fire blobs. Patch-level algorithms are developed to improve the performance of previous two categories of flame detection algorithms, however, such methods result in many outliers, affecting their accuracy.

2.1 Red green blue color model

The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

The main purpose of the RGB color model is for the sensing, representation and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before the electronic age, the RGB color model already had a solid theory behind it, based in human perception of colors.

RGB is a device-dependent color model: different devices detect or reproduce a given RGB value differently, since the color elements (such as phosphors or dyes) and their response to the individual R, G, and B levels vary from manufacturer to manufacturer, or even in the same device over time. Thus an RGB value does not define the same color across devices without some kind of color management.

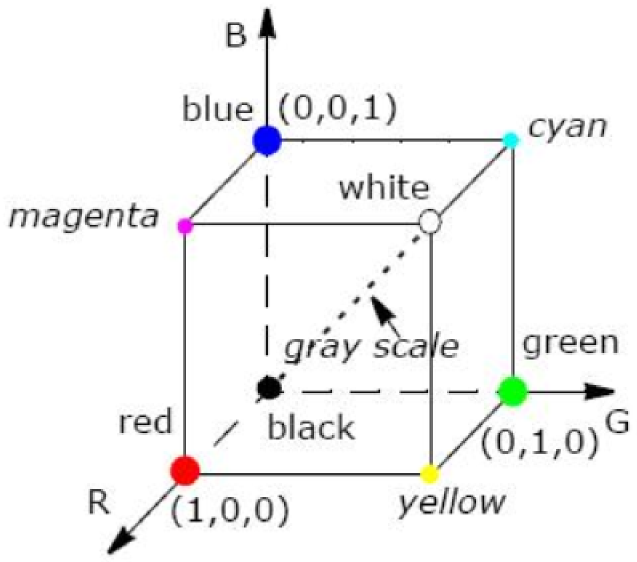


Fig. 2.1: RGB color model

In the Fig.2.1, the three primary colours, that is Red, Green and Blue are shown at the three corners of the cube. The secondary colors cyan, magenta and yellow are shown at the other three corners of the cube.

- Black is at the origin and white is at the corner furthest from the origin. In the RGB model, the gray scale (points of equal RGB values) extends from black to white along the line joining these two points.
- The different colors in this model are points on or inside the cube, and are defined by vectors extending from the origin. For convenience all color values have been normalized so the cube shown in the figure above is a unit cube.
- All the values of R,G and B are assumed to be in the range [0,1].

Typical RGB input devices are color TV and video cameras, image scanners, and digital cameras. Typical RGB output devices are TV sets of various technologies (CRT, LCD, plasma, OLED, quantum dots, etc.), computer and mobile phone displays, video projectors, multicolor LED displays and large screens.

The intensity of red color is obtained and checked againist a threshold intensity value to find out whether a fire occured or not.

2.2 Hue saturation intensity color model

The HSI color space is very important and attractive(phn phi) color model for image processingapplications because it represents color s similarly how the human eye senses colors.The HSI color model represents every color with three components: hue(sac thai) (H),saturation(bao hoa) (S), intensity (I)(cuong do). Hue is the name of a distinct color of the spectrum(quang ph)red, green, yellow,orange, blue, and so on. It is the particular wavelength frequency.

Saturation refers to the amount of white light (or gray paint) mixed with the hue. Pastelsare less saturated colors. Both of these samples below have a hue we would call ”blue” but their saturation is different.

Intensity refers to the intensity of light present. When light is at its fullest intensity, colorswill become bright, at its least intensity, colors become dim. Unlike saturation, there isn’tnecessarily ”less” of the color – it is just not as intense. You might think of value as beinga bit like the dimmer switch on your dining room light or the brightness knob on your computer’s monitor. Turn up the switch, and the value grows brighter.

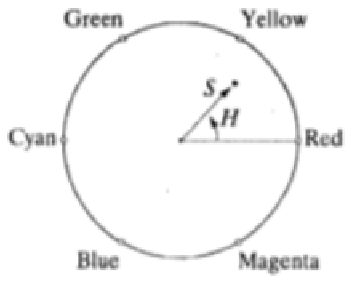


Fig. 2.2: HSI color model

Fig 2.2 shows the HSI model. The angle from the red axis gives the Hue, the length of the vector is the saturation and the intensity is given by the position of the plane on the vertical intensity axis.

The proposed framework

Majority of the research since the last decade is focused on traditional features extraction methods for flame detection. The major issues with such methods is their time consuming process of features engineering and their low performance for flame detection. Such methods also generate high number of false alarms especially in surveillance with shadows, varying lightings, and fire-colored objects. To cope with such issues, we extensively studied and explored deep learning architectures for early flame detection. Motivated by the recent improvements in embedded processing capabilities and potential of deep features, we investigated numerous CNNs to improve the flame detection accuracy and minimize the false warnings rate.

The main contributions of this article are summarized as follows:

Considering the limitations of traditional hand-engineering methods, we extensively studied deep learning (DL) architectures for this problem and propose a cost-effective CNN framework[1] for flame detection in CCTV surveillance videos. Our framework avoids the tedious and time consuming process of feature engineering and automatically learns rich features from raw fire data.

Inspired from transfer learning strategies, we trained and fine-tuned a model with architecture similar to GoogleNet [2][3] for fire detection, which successfully dominated traditional fire detection schemes.

The proposed framework balances the fire detection accuracy and computational complexity as well as reduces the number of false warnings compared to state-of-the-art fire detection schemes. Hence, our scheme is more suitable for early flame detection during surveillance to avoid huge fire disasters.

An overview of proposed framework for flame detection in CCTV surveillance networks is given in Fig.3.1.

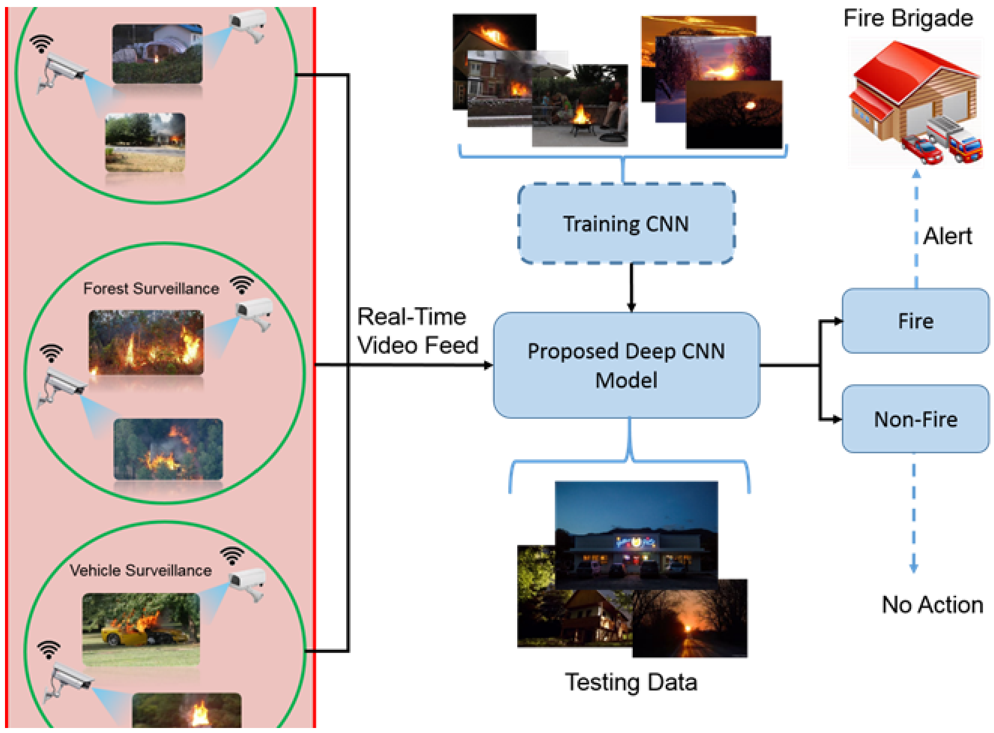


Fig. 3.1: Early flame detection in surveillance videos using deep CNN

3.1 Convolutional neural network

A convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons[4] designed to require minimal preprocessing.CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered.

3.1.1 Convolutional neural network layers

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution. This only has significance for the indices in the matrix, and thus which weights are placed at which index.

Convolutional layer

The convolution layer is the main building block of a convolutional neural network.The convolution layer comprises of a set of independent filters. Each filter is independently convolved with the image and the result is a set of feature maps.

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input as in Fig.3.2.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

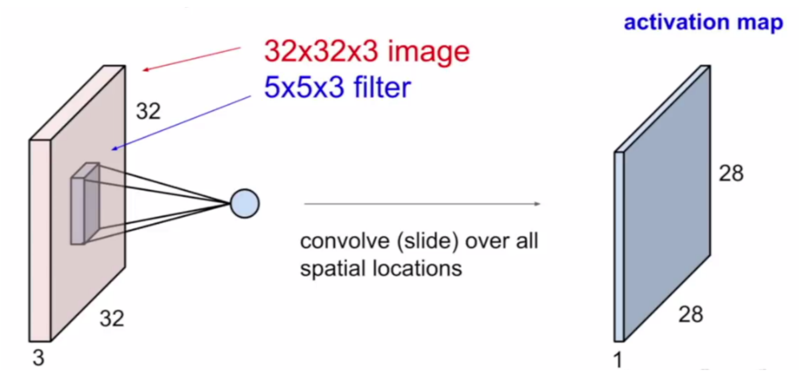


Fig. 3.2: Convolutional layer.

The main task of the convolutional layer is to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map. As a result of convolution in neuronal networks, the image is split into perceptrons, creating local receptive fields and finally compressing the perceptrons in feature maps of size m2 m3. Thus, this map stores the information where the feature occurs in the image and how well it corresponds to the filter. Hence, each filter is trained spatial in regard to the position in the volume it is applied to.

Pooling layer

A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer.

- **Max pooling**

  Max pooling uses the maximum value from each of a cluster of neurons at the prior layer as shown in Fig.3.3.Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation.

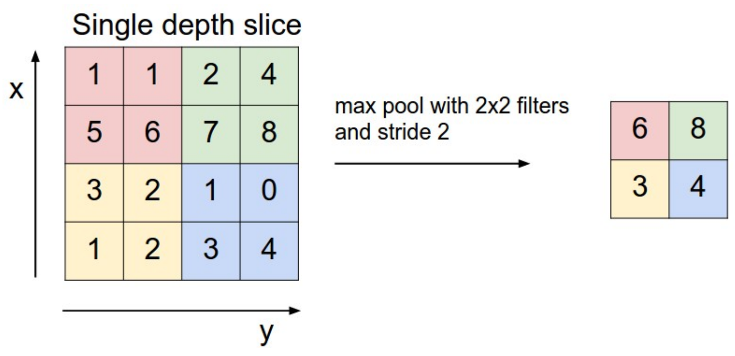


Fig. 3.3: Pooling layer.

- **Average pooling**

  average pooling uses the average value from each of a cluster of neurons at the prior layer.An average pooling layer performs down-sampling by dividing the input into rect-angular pooling regions and computing the average values of each region.

- **Sum pooling**

  Sum of all elements in the feature map call as sum pooling.

Fully connected layer

Fully connected layers connect every neuron in one layer to every neuron in another layer as shown in Fig.3.4. It is in principle the same as the traditional multi-layer perceptron neural network.After feature extraction we need to classify the data into various classes, this can be done using a fully connected neural network. In place of fully connected layers, we can also use a conventional classifier. But we generally end up adding FC layers to make the model end-to-end trainable.

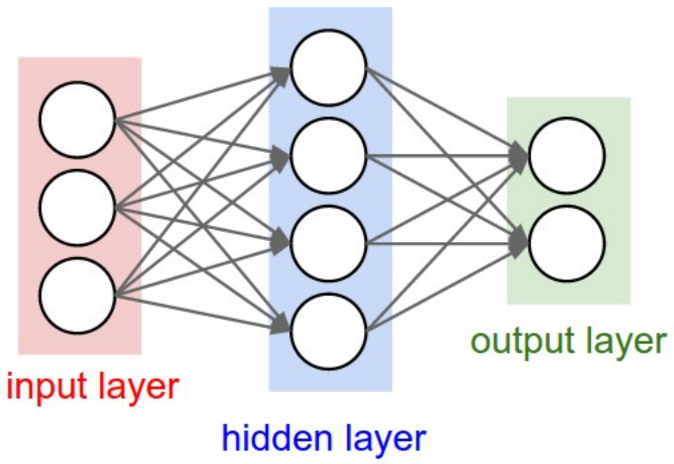


Fig. 3.4: Fully connected layer

Softmax activation function

The softmax function is a generalization of the logistic function that ”squashes” a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values, where each entry is in the interval (0, 1), and all the entries add up to 1.

The softmax function is often used in the final layer of a neural network-based classifier. Such networks are commonly trained under a log loss (or cross-entropy) regime, giving a non-linear variant of multinomial logistic regression.Softmax is implemented through a neural network layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer.

Weights

Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represents some feature of the input (e.g., a particular shape)[5]. A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

The weights are learnt such that the loss function is minimized similar to an MLP. Therefore weights are learnt to extract features from the original image which help the network in correct prediction. The weight matrix behaves like a filter in an image extracting particular information from the original image matrix. A weight combination might be extracting edges, while another one might a particular color, while another one might just blur the unwanted noise.

3.2 Output dimensions

Three hyperparameter would control the size of output volume.

- **Number of filters**

  The depth of the output volume will be equal to the number of filter applied. Remember how we had stacked the output from each filter to form an activation map. The depth of the activation map will be equal to the number of filters.

- **Stride**

  Having a stride of one the image matrix is traversed , move across and down a single pixel. With higher stride values, we move large number of pixels at a time and hence produce smaller output volumes.

- **Zero padding**

  It helps to preserve the size of the input image. If a single zero padding is added, a single stride filter movement would retain the size of the original image.

3.3 Applications of convolutional neural networks

3.3.1 Video analysis

Compared to image data domains, there is relatively little work on applying CNNs to video classification. Video is more complex than images since it has another (temporal) dimension. However, some extensions of CNNs into the video domain have been explored. One approach is to treat space and time as equivalent dimensions of the input and perform convolutions in both time and space. Another way is to fuse the features of two convolutional neural networks[6], one for the spatial and one for the temporal stream.Unsupervised learning schemes for training spatio-temporal features have been introduced, based on Convolutional Gated Restricted Boltzmann Machines and Independent Subspace Analysis.

3.3.2 Image recognition

CNNs are often used in image recognition systems. In 2012 an error rate of 0.23 percent on the MNIST database was reported.Another paper on using CNN for image classification reported that the learning process was ”surprisingly fast”; in the same paper, the best published results as of 2011 were achieved in the MNIST database and the NORB database[7].

Subsequently, a similar CNN called AlexNet won the ImageNet Large Scale Visual Recognition Challenge 2012.

When applied to facial recognition, CNNs achieved a large decrease in error rate.Another paper reported a 97.6 percent recognition rate on ”5,600 still images of more than 10 subjects”. CNNs were used to assess video quality in an objective way after manual training; the resulting system had a very low root mean square error.

The ImageNet Large Scale Visual Recognition Challenge(ILSVRC) is a benchmark in object classification and detection, with millions of images and hundreds of object classes. In the ILSVRC 2014, a large-scale visual recognition challenge, almost every highly ranked team used CNN as their basic framework. The winner GoogLeNet (the foundation of DeepDream) increased the mean average precision of object detection to 0.439329, and reduced classification error to 0.06656, the best result to date. Its network applied more than 30 layers. That performance of convolutional neural networks on the ImageNet tests was close to that of humans.The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble with images that have been distorted with filters, an increasingly common phenomenon with modern digital cameras. By contrast, those kinds of images rarely trouble humans. Humans, however, tend to have trouble with other issues. For example, they are not good at classifying objects into fine-grained categories such as the particular breed of dog or species of bird, whereas convolutional neural networks handle this.

3.3.3 Drug discovery

CNNs have been used in drug discovery. Predicting the interaction between molecules and biological proteins can identify potential treatments. In 2015, Atomwise introduced AtomNet, the first deep learning neural network for structure-based rational drug design.The system trains directly on 3-dimensional representations of chemical interactions. Similar to how image recognition networks learn to compose smaller, spatially proximate features into larger, complex structures, AtomNet discovers chemical features, such as aromaticity, sp3 carbons and hydrogen bonding. Subsequently, AtomNet was used to predict novel candidate biomolecules for multiple disease targets, most notably treatments for the Ebola virus and multiple sclerosis.

3.3.4 Go

CNNs have been used in computer Go. In December 2014, Clark and Storkey published a paper showing that a CNN trained by supervised learning from a database of human professional games could outperform GNU Go and win some games against Monte Carlo tree search Fuego 1.1 in a fraction of the time it took Fuego to play.Later it was announced that a large 12-layer convolutional neural network had correctly predicted the professional move in 55 percentage of positions, equalling the accuracy of a 6 dan human player. When the trained convolutional network was used directly to play games of Go, without any search, it beat the traditional search program GNU Go in 97 percentage of games, and matched the performance of the Monte Carlo tree search program Fuego simulating ten thousand playouts (about a million positions) per move.

3.4 Deep convolutional neural network architectures

3.4.1 AlexNet

AlexNet competed in the ImageNet Large Scale Visual Recognition Challenge in 2012. The network achieved a top-5 error of 15.3 percentage, more than 10.8 percentage[8] points lower than that of the runner up. The original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of GPUs during training. As of 2018, the Alexnet paper has been cited over 30,000 times.

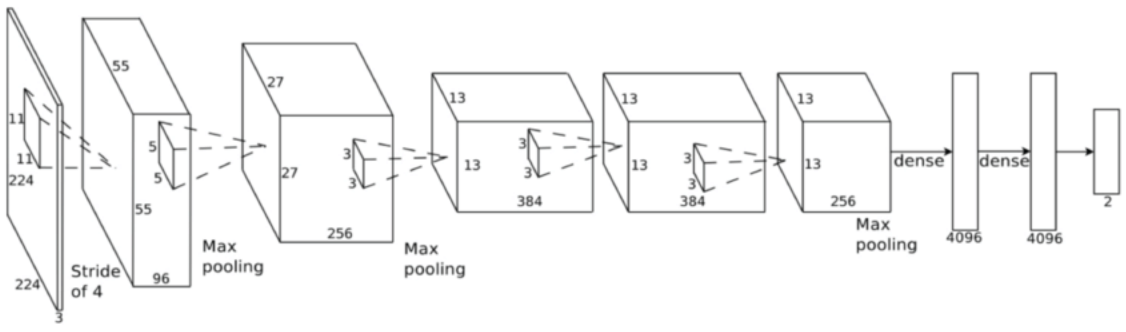


Fig. 3.5: Layers in AlexNet.

In AlexNet as we can notice the first layer has filter of size 11x11 and the second layer has 5x5 filter as shown in Fig.3.5. And there is max pooling after every convolutional layer. Some of the highlights in AlexNet Architecture:

- It uses ReLU activation function instead Sigmoid or Tanh functions.
- It speed by more than 5 times faster with same accuracy.
- It uses Dropout instead of regularisation to deal with overfitting. But the training time is doubled with dropout ratio of 0.5

More data and bigger model with 7 hidden layers, 650K units and 60M parameters.

3.4.2 GoogleNet

The winner of ILSVRC 2014 and the GoogLeNet architecture is also known as Inception Module. It goes deeper in parallel paths with different receptive field sizes and it achieved a top-5 error rate with of 6.67 percentage. It is a parallel combination of 1x1, 3x3, and 5x5 convolutional filters. The main point was that 1x1 convolutions reduce the number of parameters. The larger convolutions are more computationally expensive, so a 11 convolution reducing the dimensionality of its feature map, passing the resulting feature map through a ReLu, and then doing the larger convolution (in this case, 55 or 33). The 11 convolution is key because it will be used to reduce the dimensionality of its feature map.

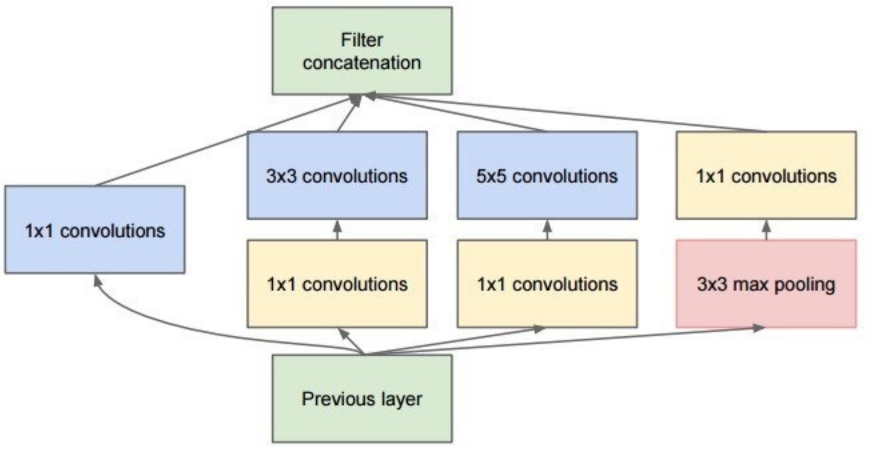


Fig. 3.6: Layers in GoogleNet Inception.

Fig.3.6 shows the GoogleNet Inception module architecture. This architecture consists of 22 layer in deep. It reduces the number of parameters from 60 million (AlexNet) to 4 million.but it doesnt have the additional 11 convolutional layers before the large convolutions (33 and 55 convolutions are considered large). Deep neural networks are computationally expensive. To make it cheaper, the we limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. Though adding an extra operation may seem counterintuitive, 1x1 convolutions are far more cheaper than 5x5 convolutions, and the reduced number of input channels also help. However, the 1x1 convolution is introduced after the max pooling layer, rather than before.

3.5 Convolutional neural network architecture

CNN is a deep learning framework which is inspired from the mechanism of visual perception of living creatures. Since the first well-known DL architecture LeNet for hand-written digits classification, it has shown promising results for combating different problems including action recognition,pose estimation, image classification, visual saliency detection, object tracking, image segmentation, scene labeling, object localization, indexing and retrieval, and speech processing. Among these application domains, CNNs have extensively been used in image classification, achieving encouraging classification accuracy over large-scale datasets compared to hand-engineered features based methods. The reason is their potential of learning rich features from raw data as well as classifier learning. CNNs generally consist of three main operations as illustrated in Fig.3.7.

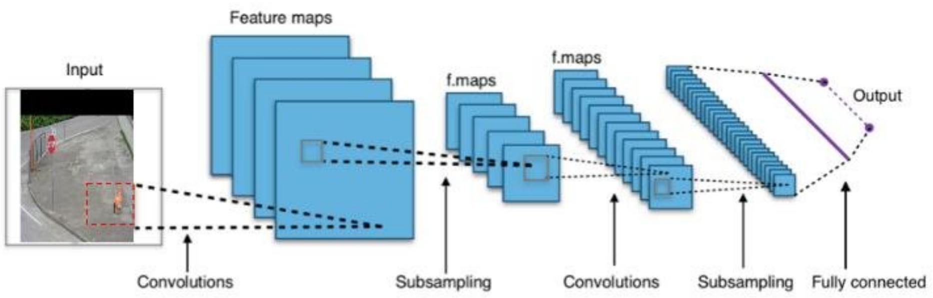


Fig. 3.7: Main operations of a typical CNN architecture.

In convolution operation, several kernels of different sizes are applied on the input data to generate feature maps. These features maps are input to the next operation known as subsampling or pooling where maximum activations are selected from them within small neighborhood. These operations are important for reducing the dimension of feature vectors and achieving translation invariance up to certain degree. Another important layer of the CNN pipeline is fully connected layer, where high-level abstractions are modeled from the input data. Among these three main operations, the convolution and fully connected layers contain neurons whose weights are learnt and adjusted for better representation of the input data during training process.

For the intended classification problem, we used a model similar to GoogleNet [2] with amendments as per our problem. The inspirational reasons of using GoogleNet compared to other models such as AlexNet include its better classification accuracy, small sized model, and suitability of implementation on FPGAs and other hardware architectures having memory constraints.

The intended architecture consists of 100 layers with 2 main convolutions, 4 max pooling, one average pooling, and 7 inception modules as given in Fig.3.8.

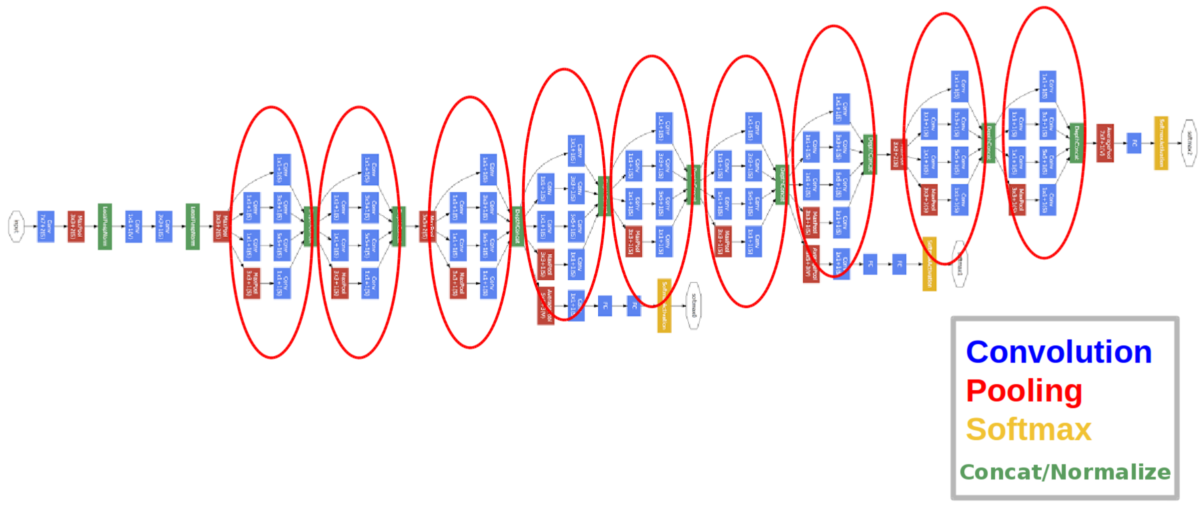


Fig. 3.8: Architectural overview of the proposed deep CNN.

3.6 Fire detection using deep convolutional neural networks

It is highly agreed among the research community that deep learning architectures automatically learn deep features from raw data, yet some effort is required to train different models with different settings for obtaining the optimal solution of the target problem. For this purpose, we trained numerous models with different parameter settings depending upon the collected training data, its quality, and problems nature.

The method applied transfer learning strategy which tends to solve complex problems by applying the previously learned knowledge. As a result, we successfully improved the flame detection accuracy up to 6 percentage from 88.41 percentage to 94.43 percentage by running the fine-tuning process for 10 epochs.

After several experiments on benchmark datasets[9], we finalized an optimal architecture, having the potential to detect flame in both indoor and outdoor surveillance videos with promising accuracy. For getting inference from the target model, the test image is given as an input and passed through its architecture. The output is probabilities for two classes i.e., fire and non-fire. The maximum probability score between the two classes is taken as the final label of a given test image. To illustrate this procedure, several images from benchmark datasets with their probability scores are given in Fig.3.9.

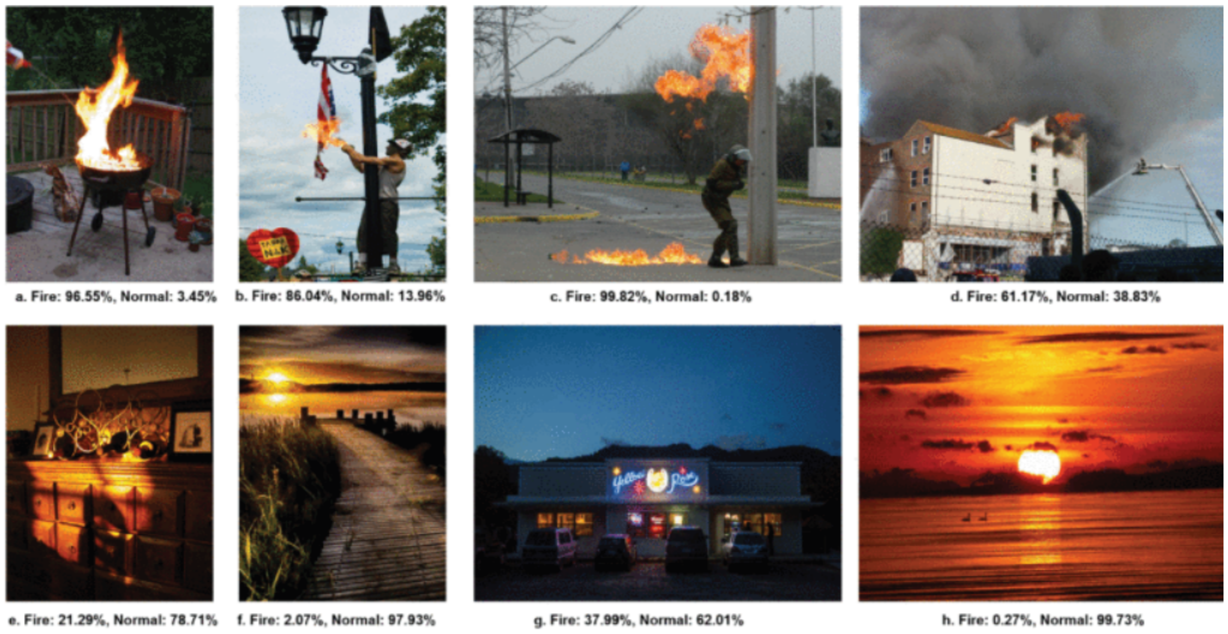

a. Fire: 96.55%, Normal: 3.45%
b. Fire: 86.04%, Normal: 13.96%
c. Fire: 99.82%, Normal: 0.18%
d. Fire: 61.17%, Normal: 38.83%
e. Fire: 21.29%, Normal: 78.71%
f. Fire: 2.07%, Normal: 97.93%
g. Fire: 37.99%, Normal: 62.01%
h. Fire: 0.27%, Normal: 99.73%

Fig. 3.9: Probability scores and predicted labels produced by the proposed deep CNN framework for different images from benchmark datasets.

The size of input image[10][11] is 2242243 pixels on which 64 kernels of size 77 are applied with stride 2, resulting in 64 feature maps of size 112112 . Then, a max pooling with kernel size 33 and stride 2 is used to filter out maximum activations from previous 64 feature maps. Next, another convolution with filter size 33 and stride 1 is applied, resulting in 192 feature maps of size 5656 . This is followed by another max pooling layer with kernel size 33 and stride 2, filtering discriminative rich features from less important ones. Next, the pipeline contains two inception layers. The motivational reason of such inception modulus assisted architecture is to avoid uncontrollable increase in the computational complexity and networks flexibility to significantly increase the number of units at each stage.

To achieve this, dimensionality reduction mechanism is applied before computation-hungry convolutions of patches with larger size. The approach used here is to add 11 convolutions for reducing the dimensions, which in turn minimizes the computations. Such mechanism is used in each inception module for dimensionality reduction. Next, the architecture contains a max pooling layer of kernel size 33 with stride 2, followed by four inception modules. Next, another max pooling layer of same specification is added, followed by two more inception layers.

Then, an average pooling layer with stride 1 and filter size 77[12]is introduced in the pipeline, followed by a dropout layer to avoid overfitting. At this stage, we modified the architecture according to our classification problem by keeping the number of output classes to 2 i.e., fire and non-fire.

3.7 Performance evaluation

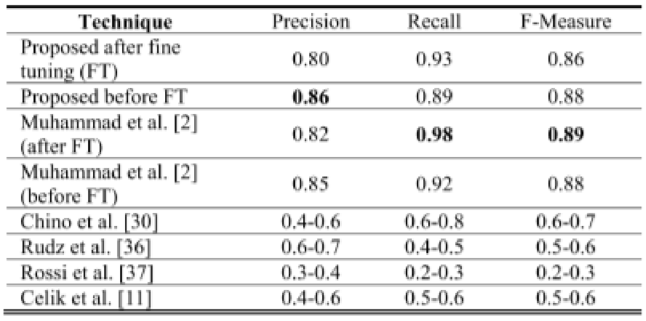
Experiments from different perspectives using images and videos from different sources were analysed.All experiments are performed using NVidia GeForce GTX[13] TITAN X with 12 GB onboard memory and deep learning framework and Ubuntu OS installed on Intel Core i5 CPU with 64 GB RAM.he total number of images used in experiments is 68457, out of which 62690 frames are taken from Dataset1 and remaining from other sources

3.7.1 Performance on dataset1

Dataset1 containing 31 videos which cover different environments. This dataset has 14 fire videos and 17 normal videos without fire. The dataset is challenging as well as larger in size, making it a better option for experiments. The dataset has been made challenging for both color-based and motion-based fire detection methods by capturing videos of fire-like objects and mountains with smoke and clouds.

The results are compared with other flame detection methods, which are carefully selected using a selection criteria, reflecting the features used for fire detection, time, and dataset. The best results are reported by among the existing recent methods by achieving an accuracy of 93.55 percent with 11.67 percent false alarms. The score of false alarms is still high and needs further improvement.

Fig.3.10 shows the prediction accuracy values of various models, which are compared with the proposed model.

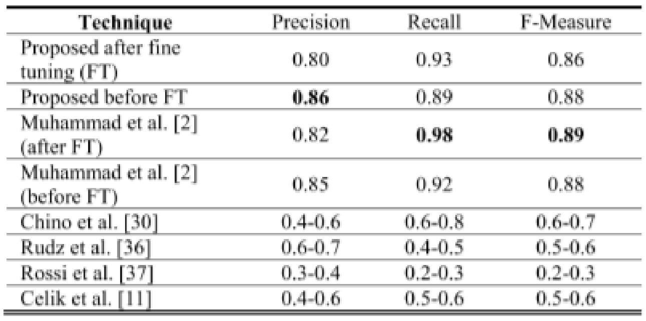


Technique	Precision	Recall	F-Measure
Proposed after fine tuning (FT)	0.80	0.93	0.86
Proposed before FT	**0.86**	0.89	0.88
Muhammad et al. [2] (after FT)	0.82	**0.98**	**0.89**
Muhammad et al. [2] (before FT)	0.85	0.92	0.88
Chino et al. [30]	0.4-0.6	0.6-0.8	0.6-0.7
Rudz et al. [36]	0.6-0.7	0.4-0.5	0.5-0.6
Rossi et al. [37]	0.3-0.4	0.2-0.3	0.2-0.3
Celik et al. [11]	0.4-0.6	0.5-0.6	0.5-0.6

Fig. 3.10: Comparison with different fire detection methods.

3.7.2 Performance on dataset2

Dataset2 contain 226 images out of which 119 images belong to fire class and 107 images belong to non-fire class. The dataset is small but very challenging as it contains red-colored and fire-colored objects, fire-like sunlight scenarios, and fire-colored lightings in different buildings.The results are compared with five methods including both hand-crafted features based methods and deep learning based method.Fig.3.11 shows the comparsion of dataset2 with different models.



Technique	Precision	Recall	F-Measure
Proposed after fine tuning (FT)	0.80	0.93	0.86
Proposed before FT	**0.86**	0.89	0.88
Muhammad et al. [2] (after FT)	0.82	**0.98**	**0.89**
Muhammad et al. [2] (before FT)	0.85	0.92	0.88
Chino et al. [30]	0.4-0.6	0.6-0.8	0.6-0.7
Rudz et al. [36]	0.6-0.7	0.4-0.5	0.5-0.6
Rossi et al. [37]	0.3-0.4	0.2-0.3	0.2-0.3
Celik et al. [11]	0.4-0.6	0.5-0.6	0.5-0.6

Fig. 3.11: Results of Dataset2 for the proposed method and other fire detection methods.

3.7.3 Effect on the performance against different attacks

The effect on performance of our method against different attacks such as noise, cropping, and rotation were tested. Two test images were considered: one from fire class and second from normal class.

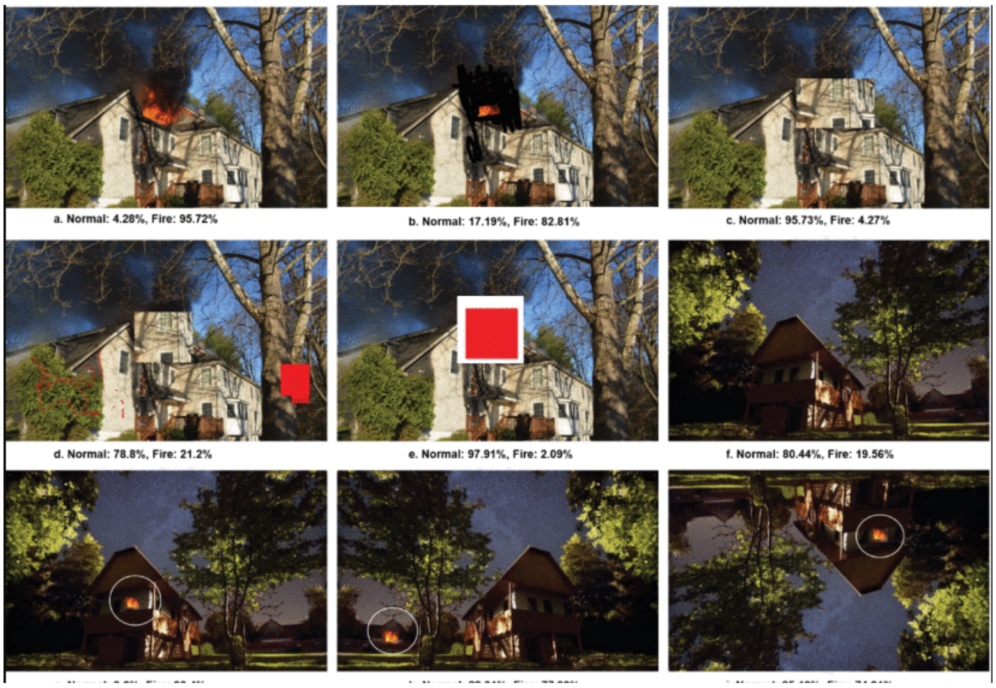

a. Normal: 4.28%, Fire: 95.72%
b. Normal: 17.19%, Fire: 82.81%
c. Normal: 95.73%, Fire: 4.27%
d. Normal: 78.8%, Fire: 21.2%
e. Normal: 97.91%, Fire: 2.09%
f. Normal: 80.44%, Fire: 19.56%

Fig. 3.12: Effect on fire detection accuracy for proposed method against different attacks.

Better performance was observed with different types of fire and non fire images as shown in Fig 3.12:

- The fire region in the image is distorted and the resultant image is passed through our method. CNN method still assigned it the label fire with accuracy 82.81 percent.
- The fire region is blocked and proposed method successfully predicted it as normal. To show the effect on performance against images with fire-colored regions.
- Red-colored boxes were placed on different parts of the image and these were identified as non fire images.

These tests indicate that the proposed algorithm can detect fire even if the video frames are effected by noise or the amount of fire is small and at a reasonable distance, in real-world surveillance systems, thus, validating its better performance.

Conclusion

The recent improved processing capabilities of smart devices have shown promising results in surveillance systems for identification of different abnormal events i.e., fire, accidents, and other emergencies. Fire is one of the dangerous events which can result in great losses if it is not controlled on time. This necessitates the importance of developing early fire detection systems. Therefore, in this research article, we propose a cost-effective fire detection CNN architecture for surveillance videos. The model is inspired from GoogleNet architecture and is fine-tuned with special focus on computational complexity and detection accuracy. Through experiments, it is proved that the proposed architecture dominates the existing hand-crafted features based fire detection methods as well as the AlexNet architecture based fire detection method.

REFERENCES

[1] K. Muhammad, R. Hamza, J. Ahmad, J. Lloret, H. H. Ge Wang, and S. W. Baik, secure surveillance framework for IoT systems using probabilistic image encryption, IEEE Trans. Ind. Inform.

[2] K. Muhammad, J. Ahmad, and S. W. Baik, Early fire detection using convolutional neural networks during surveillance for effective disaster management, Neurocomputing, vol. 288, pp. 3042, May 2018

[3] J. Choi and J. Y. Choi, An integrated framework for 24-hours fire detection, in Proc. Eur. Conf. Comput. Vis., 2016, pp. 463479

[4] H. J. G. Haynes. (2016). Fire Loss in the United States During 2015.

[5] C.-B. Liu and N. Ahuja, Vision based fire detection, in Proc. 17th Int. Conf. Pattern Recognit. (ICPR), Aug. 2004, pp. 134137.

[6] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou, An early fire-detection method based on image processing, in Proc. Int. Conf. Image Process. (ICIP), Oct. 2004, pp. 17071710.

[7] B. U. Treyin, Y. Dedeolu, U. Gdkbay, and A. E. etin, Computer vision based method for real-time fire and flame detection, Pattern Recognit. Lett., vol. 27, pp. 4958, Jan. 2006

[8] J. Choi and J. Y. Choi, Patch-based fire detection with online outlier learning, in Proc. 12th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS), Aug. 2015, pp. 16.

[9] G. Marbach, M. Loepfe, and T. Brupbacher, An image processing technique for fire detection in video images, Fire Safety J., vol. 41, no. 4, pp. 285289, 2006

[10] D. Han and B. Lee, Development of early tunnel fire detection algorithm using the image processing, in Proc. Int. Symp. Vis. Comput., 2006, pp. 3948

[11] T. elik and H. Demirel, Fire detection in video sequences using a generic color model, Fire Safety J., vol. 44, no. 2, pp. 147158, 2009.

[12] P. V. K. Borges and E. Izquierdo, A Probabilistic approach for visionbased fire detection in videos, IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 5, pp. 721731, May 2010.

[13] A. Rafiee, R. Dianat, M. Jamshidi, R. Tavakoli, and S. Abbaspour, Fire and smoke detection using wavelet analysis and disorder characteristics, in Proc. 3rd Int. Conf. Comput. Res. Develop. (ICCRD), Mar. 2011, pp. 262265.