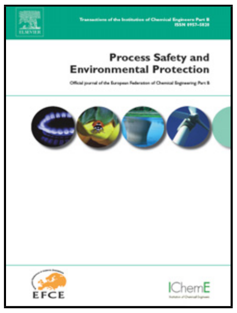


Title: An intelligent fire detection approach through cameras based on computer vision methods

Authors: Hao Wu, Deyang Wu, Jinsong Zhao

PII: S0957-5820(18)31452-6
DOI: https://doi.org/10.1016/j.psep.2019.05.016
Reference: PSEP 1770

To appear in: *Process Safety and Environment Protection*

Received date: 25 December 2018
Accepted date: 10 May 2019

An intelligent fire detection approach through cameras based on computer vision methods

Hao Wu, Deyang Wu, Jinsong Zhao*

State Key Laboratory of Chemical Engineering,
Department of Chemical Engineering,
Tsinghua University, Beijing 100084, China
Beijing Key Laboratory of Industrial Big Data System and Application,
Tsinghua University, Beijing 100084, China

*Corresponding author at: Department of Chemical Engineering, Tsinghua University, Beijing, 100084, China.
E-mail address: jinsongzhao@tsinghua.edu.cn (Jinsong Zhao)

Highlights

- An intelligent fire detection approach through cameras based on computer vision methods is proposed.
- A motion detection method based on background subtraction is used for reducing computations.
- An object detection model and an image classification model are constructed and applied for fire detection.
- The fire detection rate achieves 98.4% and the false alarm rate achieves 99.9% based on the fire image dataset.
- The detection time for one frame achieves 27.4ms.

Abstract

Fire that is one of the most serious accidents in petroleum and chemical factories, may lead to considerable production losses, equipment damages and casualties. Traditional fire detection was done by operators through video cameras in petroleum and chemical facilities. However, it is an unrealistic job for the operator in a large chemical facility to find out the fire in time because there may be hundreds of video cameras installed and the operator may have multiple tasks during his/her shift. With the rapid development of computer vision, intelligent fire detection has received extensive attention from academia and industry. In this paper, we present a novel intelligent fire detection approach through video cameras for preventing fire hazards from going out of control in chemical factories and other high-fire-risk industries. The approach includes three steps: motion detection, fire detection and region classification. At first, moving objects are detected through cameras by a background subtraction method. Then the frame with moving objects is determined by a fire detection model which can output fire regions and their locations. Since false fire regions (some objects similar with fire) may be generated, a region classification model is used to identify whether it is a fire region or not. Once fire appears in any camera, the approach can detect it and output the coordinates of the fire region. Simultaneously, instant messages will be immediately sent to safety supervisors as a fire alarm. The approach can meet the needs of real-time fire detection on the precision and the speed. Its industrial deployment will help detect fire at the very early stage, facilitate the emergency management and therefore significantly contribute to loss prevention.

Keywords: Fire detection, loss prevention, computer vision, convolutional neural networks.

1. Introduction

Fire in petroleum and chemical factories may lead to considerable production losses, equipment damages and casualties. Table 1 lists the 20 largest industry losses since 1988 and half of the incidents had fire (Nolan, 2018). Fire detection at the early stage can effectively prevent the spread of fire and minimize the damage caused by fire. In indoor environments, smoke detectors and flame detectors are widely used for fire alarms. However, these traditional physical sensors have a number of limitations. They require a close proximity to fire sources so that they cannot work for the outdoor scenes (Shi et al., 2017). Besides, they generally require the fire to burn for a while to produce large amount of smoke and then trigger the alarm (Zhang et al., 2016). Moreover, they cannot provide information about fire location and fire size (Frizzi et al., 2016). In practical scenarios, there are hundreds of video cameras installed in a large factory and only a few monitoring screens in a control room. This means that only several cameras can be displayed simultaneously and the other cameras have to take turns to be displayed. In order to display all of the cameras, it may take dozens of minutes. During this period, fire accidents may occur and be missed by the human operator who is in charge of watching the monitoring screens. In addition, the human operators often have other tasks such as monitoring the DCS control systems. Therefore, there is an urgent demand to develop a new approach to automatically detect the fire hazards.

With the rapid development of digital camera technology and computer vision technology, intelligent video fire detection methods have been proposed and applied in some industries. The development of video fire detection has gone through two stages. At the beginning, fire detection was done by using color model and hand-designed features (Chen et al., 2004; Homg et al., 2005; Marbach et al., 2006; Töreyin et al., 2006; Ebert, 2009; Celik, 2010; Narwani and Selva, 2016). They focused on color and shape characteristics of flames. Conventional video fire detection methods addressed the problem by extracting a multi-dimensional feature vector and classifying the feature vector into "fire" or "non-fire" class. However, since the discriminative feature extractors were hand-designed, these methods were not robust enough and were difficult to meet the needs of different scenarios.

In the famous ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition in 2012, AlexNet (Krizhevsky et al., 2012), the model that used a deep convolutional neural network (CNN) won the champion. Since then, CNN has brought about a revolution in computer vision and pattern recognition. CNN has the ability to perform feature extraction and classification within one network. CNN can also replace the hand-designed features and learn the complete characteristics of objects. To achieve better detection performance, some convolutional neural network (CNN) based video fire detection methods have been proposed over the past three years. Frizzi et al. (2016) proposed a CNN for video fire and smoke detection. The results showed that CNN based method achieved better performance than some relevant conventional video fire detection methods. Zhang et al. (2016) proposed a CNN based method for forest fire detection. The detection was operated in a cascaded fashion. At first, the frame from a camera is tested by the global image-level classifier, if fire is detected, the fine grained patch classifier is followed to detect the precise location of fire patches. Wang et al. (2017) developed a novel approach based on CNN and support vector machine (SVM). They replaced the fully connected layer with SVM for the classification task. It showed better performance than the pure CNN. Maksymiv et al. (2017) presented a cascaded approach based on AdaBoost, local binary patterns (LBP) and CNN for detection.

Despite their excellent performance for fire detection, there were still some limitations during the practical application. The aforementioned methods utilized CNN as a classifier so that they can only detect whether fire exists in an image or not. It is almost impossible to know the fire location and it is difficult to detect a small fire in a large image from a camera because most of the pixels are the background. Some of them (Maksymiv et al., 2017; Wang et al., 2017) had to use conventional hand-designed feature extractors to extract the regions of interest (ROI), then detect the fire in the ROI. However, this violates the purpose of replacing the hand-designed feature extractors with CNN, and the approach doesn't learn the complete characteristics of the flame. Some of them (Frizzi et al., 2016; Zhang et al., 2016) utilized a slide window method to overcome this problem. The raw image is divided into many patches and each patch is detected by CNN. The patch size is fixed and usually less than the fire size. As a result, patch classifier also cannot learn the complete characteristics of the flames. Besides, each of the generated ROIs or patches should be computed via the CNN network. Since the CNN computation costs lots of time and hardware resources, these methods are still not satisfactory in term of detection speed when there is not a good hardware resource. Moreover, these methods didn't consider that there are many objects similar with fire in application. Because their CNN model focused on the classification between fire images and non-fire images, the fire-like objects might lead to too many false alarms.

In this paper, we present a novel intelligent fire detection approach, which aims to achieve high detection rate, low false alarm rate and high speed. In order to reduce the CNN computation, we added a motion detection method. Only if moving objects appear, the following computations will be implemented. To replace the hand-designed feature extractors or the slide window methods for ROI generation, we used an object detection method based on CNN to generate fire regions directly. Then we focused on the classification between fire images and fire-like images to avoid the false alarm problem. The rest of this paper is organized as follows: Section 2 describes the proposed approach. Section 3 shows the experimental results and finally section 4 summaries this paper.

2. Method

2.1 Computer vision

Recently, with the rapid development of deep learning (DL) and artificial intelligence (AI), computer vision has drawn significant attention from academia and industry, and has achieved spectacular success. During this period, the emergence of deep CNN was the most important event. CNN was first proposed in the late 1980s (LeCun et al., 1989), to replace the traditional hand-designed feature descriptors such as scale invariant feature transform (SIFT), histogram of oriented gradient (HOG) and local binary pattern (LBP). In 2012, a CNN architecture named AlexNet (Krizhevsky et al., 2012) won the champion of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition and achieved half the error rate of the second-best model. Since then, CNN has brought about a revolution in computer vision and pattern recognition (LeCun et al., 2015). Now CNN has become the dominant method in computer vision and even machine learning tasks.

In computer vision domain, there are three main tasks, which are image classification, object detection, instance segmentation. Among them, image classification has been studied completely with the ImageNet dataset. Image classification aims to identify the classes of the images (see Fig. 1(a)). In this task, the performance of computers using CNN based methods has surpassed humans. Object detection aims to detect and locate objects in images (see Fig. 1(b)). This means that the model will output the labels of objects and their coordinates. Besides, instance segmentation frames different instances from one image with object detection method, and then uses semantic segmentation method to mark each pixel in different instance areas (see Fig. 1(c)).

In our approach, image classification and object detection methods are combined to deal with fire detection. Image classification relies on a CNN model, the input and the output of which are images and predictive labels respectively. For our approach as a binary classification issue, if we input a fire image to CNN, the network will output a 2-length vector like [0.1, 0.9]. "0.1" means the probability of non-fire in the input image and "0.9" means the probability of fire. In this way, CNN can predict the labels of images. The basic architecture of CNN is a stack of several layers, such as convolutional layers, pooling layers, batch normalization and fully connected layers. Recently, some modules with greater performance have been presented by researchers, such as inception (Szegedy et al., 2015, 2016), residual (He et al., 2016) and xception modules (Chollet, 2017). However, most of frames from cameras include lots of background, which may interfere the label prediction. Therefore, we utilized object detection to detect the fire regions in frames directly. There are two categories of methods for object detection: two-stage methods and one-stage methods. Two-stage methods are based on region proposal, including R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2017), etc. These methods divide the object detection into two parts. At first, they generate many region proposals where objects may exist. Then each region proposal will be input into a CNN model, to predict the labels of region proposals and adjust their coordinates. One-stage methods regard the detection task as a regression problem, including YOLO (You only look one) (Redmon et al., 2016, 2017), SSD (Liu et al., 2016), etc. These methods will output the class probabilities and coordinates of objects directly. Because the two-stage methods generate many region proposals and each of them needs to be input to the network, the detection will take up lots of time and computing resources. One-stage methods can achieve faster detection speed but the accuracy is a little lower than two-stage methods. In our fire detection task, considering the real-time speed, we select YOLO method for the object detection.

2.2 Framework

Fig. 2 shows the framework of the proposed intelligent fire detection approach. The approach has two key steps, which are fire detection step based on object detection method and region classification step based on image classification method. Before the application, we should train two CNN networks for these two steps. At the application stage, videos from surveillance cameras are captured and processed by the background subtraction method, which can detect moving objects from static scenes. If moving objects appear (fire is a moving object), the current frame will be processed by the trained fire detection network. The network will directly predict the probabilities and coordinates of fire regions, which are ROIs. However, in this step, the network may generate some false ROIs such as red/orange/yellow clothes, helmets, lights, etc. We continue to use the trained region classification network to identify whether each of the generated ROIs is a fire region or not. In this way, once fire appears in the current frame, the approach can find the fire region. Simultaneously, the frame where the fire region is localized will be immediately sent to safety supervisors as a fire alarm. In the following part, the background subtraction, fire detection and region classification will be described in details.

2.3 Background subtraction

Video streams are captured from outdoor surveillance cameras by a computing server. Background subtraction aims to eliminate most of static image frames and retrench the resources of image computation, because the networks are computed on GPUs to accelerate the speed. The early methods for background subtraction were based on Gaussian Mixture Model (GMM) (Power and Schoonees, 2002; Hayman and Eklundh, 2003). Besides, to improve the adaptation speed of GMM, the Adaptive Mixture of Gaussian method was developed by KaewTraKulPong and Bowden in 2002. A study showed that KNN based method has better performance on simple static scenes (Zivkovic and Van Der Heijden, 2006). This method is more suitable for our fire detection task in outdoor static scenes. The video stream capture and the background subtraction are implemented by OpenCV, which is an open source computer vision library. Fig. 3 illustrates the whole background subtraction process. First, to eliminate the noise of images, Gaussian blur is applied on the original frames from one video stream (see Fig. 3(a)). After that, KNN based background subtraction is implemented on Fig. 3(a). We can capture the moving object (such as fire and humans), which is the white regions in Fig. 3(b). But we can find that there are some grey regions (in the green circle) in Fig. 3(b), so we binarize this image and mark the moving object with white color in Fig. 3(c). Finally, closing operation of morphology is used to combine little discrete regions into a complete region (in the red circle). In this way, the whole moving object is detected in Fig. 3(d).

In the fire detection approach, we will calculate the size of white regions in the whole frame. We will set a threshold n and if the size of white regions (moving objects) exceeds the threshold, the next detection steps will be implemented.

2.4 Fire detection

The fire detection model is one of the key steps in our approach. This model will process the frames with moving objects from the background subtraction step and generate the ROIs where fire may exist. The fire detection model is developed by YOLO method. YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. It trains a single neural network like CNN to predict bounding boxes and class probabilities directly from images in one evaluation. It is designed for multi-object detection, however, fire detection is a single-object (fire) detection task.

To input the frame into the CNN, the image should be resized to 416 × 416 pixels. YOLO divides the input image into an $S \times S$ grid (see Fig. 4). If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. The network of YOLO predicts three parts for k bounding boxes (the bounding boxes are with different width and height) in each grid cell: coordinates, confidence scores and conditional class probabilities.

Instead of predicting coordinates of the bounding boxes directly, YOLO uses k hand-picked priors (named anchor boxes) to predict k bounding boxes. According to the paper of YOLO, predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn (Redmon and Farhadi, 2017). The network predicts the offsets of 4 coordinates and a confidence score for each bounding box: t_x, t_y, t_w, t_h, and t_o (see Fig. 5). If the cell is offset from the top left corner of the image by $(c_x,\ c_y)$ and the bounding box prior (anchor box) has width and height $(p_w,\ p_h)$, then the bounding box predictions correspond to Equation (1)~(5), where σ is the logistic activation. This bounds the predicted bounding box centers within the corresponding grid cell.

$$b_x = \sigma(t_x) + c_x \tag{1}$$

$$b_y = \sigma(t_y) + c_y \tag{2}$$

$$b_w = p_w e^{t_w} \tag{3}$$

$$b_h = p_h e^{t_h} \tag{4}$$

$$confidence = \sigma(t_o) \tag{5}$$

The confidence score reflects how confident the model is that the box contains an object.

$$confidence = Pr(Object) \times IOU_{pred}^{truth} \tag{6}$$

where $Pr(Object)$ represents whether objects exist in that box. If no object, the confidence score should be 0. Otherwise, the confidence score is equal to the intersection over union (IOU) between the predicted box and the ground truth box (the region where objects really exist).

For the multi-object task, each bounding box also predicts C conditional class probabilities, $Pr(Class_i|Object)$. To obtain the class-specific confidence scores for each box, the conditional class probabilities are multiplied by the confidence score:

$$\begin{aligned} Pr(Class_i|Object) \times confidence &= Pr(Class_i|Object) \times Pr(Object) \times IOU_{pred}^{truth} \\ &= Pr(Class_i) \times IOU_{pred}^{truth} \end{aligned} \tag{7}$$

These scores reflect both the probability of that class appearing in the box and how well the predicted box fits the object. The predictions of the network for each image are encoded as an $S \times S \times k \times (4 + 1 + C)$ tensor. For fire detection that is a single-class detection task, $C = 1$.

For training our network of YOLO, a loss function for fire detection is defined as follows:

$$\begin{aligned} \text{loss} = {} & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{k} \mathbb{1}_{ij}^{obj} (2 - w_{ij} h_{ij}) \left[\left(x_{ij} - \hat{x}_{ij}\right)^2 + \left(y_{ij} - \hat{y}_{ij}\right)^2 + \left(w_{ij} - \widehat{w}_{ij}\right)^2 + \right. \\ & \left. \left(h_{ij} - \hat{h}_{ij}\right)^2 \right] + \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{k} \mathbb{1}_{ij}^{obj} (c_{ij} - \hat{c}_{ij})^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{k} \mathbb{1}_{ij}^{noobj} (c_{ij} - \hat{c}_{ij})^2 + \\ & \lambda_{class} \sum_{i=0}^{S^2} \sum_{j=0}^{k} \mathbb{1}_{ij}^{obj} (p_{ij}(fire) - \hat{p}_{ij}(fire))^2 \end{aligned} \tag{8}$$

where $\mathbb{1}_{ij}^{obj}$ denotes that objects appears in the jth bounding box in cell i and $\mathbb{1}_{ij}^{noobj}$ denotes that there is no object. The first item denotes the error of coordinates between the predicted bounding boxes and the ground truth boxes, if object exists. (x, y, w, h) denote the coordinates of the predicted bounding boxes and $(\hat{x}, \hat{y}, \hat{w}, \hat{h})$ denote the coordinates of the ground truth boxes. The second and the third items denote the error of confidence scores of bounding boxes with object or non-object. If objects (fire) exist in the ground truth box, $\hat{c}_{ij} = IOU_{pred}^{truth}, \hat{p}_{ij}(fire) = 1$; otherwise $\hat{c}_{ij} = 0, \hat{p}_{ij}(fire) = 0$. And the last item denotes the classification error (the conditional class probabilities) if an object is present in that grid cell.

However, the magnitudes of coordinate error and classification error may be different. And most of the grid cells in one image do not contain any object. To remedy these issues, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that do not contain objects. We set with $\lambda_{coord} = 2$, $\lambda_{class} = 1$, $\lambda_{noobj} = 0.2$, $\lambda_{class} = 1$. Using this loss function, we can train the network with mini-batch gradient descent method.

After the training, we can apply the network to detect the fire regions for the captured frames. We should set a threshold $t \in (0, 1)$ and the network will predict the probabilities of each bounding box in each grid cell. If fire appears in the current frame, the network will output the fire region whose probability $c_{ij} \times p_{ij}(\text{fire}) > t$. However, because the network only learnt the knowledge of fire characteristics in this step, it may generate "fire regions" with fire-like objects, such as red/orange/yellow clothes, helmet, lights, etc. So we just obtain the fire region proposals and they need to be distinguished strictly in the next step.

2.5 Region classification

In this step, we design a CNN model to distinguish the real fire regions (positive) from the fire-like regions (negative), which is a typical classification task. Since the input size of CNN model is fixed, the generated ROIs with different width and height need to be resized to 256×256 pixels. Then each of the resized regions will be computed through a single CNN model. Due to the excellent performances on ImageNet classification task, in the following experiments, we chose Inception-V3 (Szegedy et al., 2016), ResNet-50 (He et al., 2016) and Xception (Chollet, 2017) as the CNN architectures to compare their performances on our region classification task.

Inception modules mainly increased the width of the network while keeping the computational budget constant (see Fig. 6(a)). Residual module overcame the training problem of deeper neural networks, and made it possible to train the network with 100 or even 1000 layers (see Fig. 6(b)). Xception interpreted the Inception modules are an intermediate step in-between regular convolution and depthwise separable convolution. So it replaced the Inception modules with depthwise separable convolutions and achieved greater performance on ImageNet dataset (see Fig. 6(c)).

The output of the CNN for each resized region is a 2-length array $Z = [Z_1, Z_2]$. Then using softmax function to transform it into the probabilities of fire and fire-like classes. The softmax function also named normalized exponential function, transforms a K-dimensional vector Z of arbitrary real values into a K-dimensional vector $\sigma(Z)$ of real values in the range of (0,1) that add up to 1:

$$\sigma(Z)_j = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_k}}, j = 1,2, \dots, K \qquad (9)$$

The output is an array $\sigma(Z) = [\sigma(Z)_1, \sigma(Z)_2]$ and $\sigma(Z)_1 + \sigma(Z)_2 = 1$. $\sigma(Z)_1$ denotes the probability of fire and $\sigma(Z)_2$ denotes the probability of fire-like. Therefore, if $\sigma(Z)_1 > 0.5$, the generated region is a fire region; otherwise, it's not a fire region.

3. Experimental results

3.1 Fire detection

3.1.1 Dataset

Images that contains fire regions, were collected from some fire image datasets (like ImageNet) which were set up by previous researchers and some image search engines such as Google and Baidu. Furthermore, via the internet, we augmented our dataset with the images of chemical plant fire and other scenarios. The whole fire image dataset contains 5075 images with various width and height. The proportion of the training set is 80%, that is, 4060 images as the training set and 1015 images as the testing set. It should be noted that each image may contain one or several fire regions and each fire region is annotated by a bounding box for describing its location (see Fig. 7).

3.1.2 Network

The networks for this step are shown in Fig. 8. For comparison, we selected two networks with different network size, Detection Network I and II, to validate the detection performance for our task. The input of the networks is 416×416 with 3 channels (RGB). The two networks are both composed of convolutional layers with batch normalization and maxpooling layers. The output of the networks is 13×13×30, which means that the input image is divided into a 13×13 grid and the networks predicts t_x, t_y, t_w, t_h, t_o and $p(fire)$ for 5 anchor boxes.

As mentioned in section 2.4, because the networks predict the offsets of coordinates relative to anchors, widths and heights of anchors are vital and prespecified in each grid cell for prediction. If better anchor boxes for the network to start, it is easier for the network to predict. Here K-means clustering was implemented on the bounding boxes of the training set to find appropriate anchor boxes. Because we want the anchor boxes to lead to good

IOU scores, we define the distance metric of the K-means algorithm as follows:

$$d(truth, anchor) = 1 - IOU_{anchor}^{truth} \quad (10)$$

Fig. 9 shows the average IOU of clustering box dimensions with different k. It demonstrates that the average IOU increases with the increase of k. However, the model complexity will also increase. Therefore, at the model testing phase, we need to consider the tradeoff between model complexity and recall rate. Here we select $k = 5$. Table 2 lists widths and heights of anchor boxes with $k = 5$.

3.1.3 Result

In the detection task, detection accuracy and speed should be considered simultaneously. Usually, we use Average Precision (AP) to evaluate the detection accuracy and use Frames Per Second (FPS) to evaluate the detection speed.

In a pattern recognition task, we need to define the true positives and the false positives (see Table 3). If $\text{IOU} > 0.5$ between the predicted box and the ground truth box, the predicted box is a "true positive (TP)"; otherwise, it is a "false positive (FP)". Since every part of the image where we didn't predict an object is considered a negative, measuring "True negatives (TN)" is a bit futile. So we only measure "False negatives (FN)" as the objects that the model has missed out (see Fig. 10). The predicted bounding boxes are red and the ground truth boxes are blue.

Besides, precision and recall are two vital measures of the algorithm evaluation. In a classification task, the precision for a class is the number of true positives divided by the total number of elements predicted and labeled as belonging to the positive class. Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class. Usually, there is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other. Therefore, average precision (AP) is usually used to address this issue. By computing precision and recall with different threshold, a precision-recall (P-R) curve can be plotted, plotting precision $P(R)$ as a function of recall R. AP computes the average value of $P(R)$ over the interval from $R = 0$ to $R = 1$ (see Fig. 11). According to the PASCAL VOC Challenge, the interval is 0.1 and the method of AP computation is called 11-point interpolated average precision (Everingham et al., 2010). The definitions of precision, recall, AP and accuracy are as follows:

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \quad (11)$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \quad (12)$$

$$AP = \int_0^1 P(R)dR \quad (13)$$

$$Accuracy = \frac{true\ positive + true\ negative}{true\ positive + false\ positive + true\ negative + false\ nagetive} \quad (14)$$

The other measure, FPS, means how much frames the model can detect at one second. The experiments were done on a computer server with a NVIDIA 1080 GPU.

At the training phase, mini-batch gradient descent is used to train the networks. The idea of the algorithm is to update the weights and biases of neurons after each batch computation, the aim of which is to minimize the training loss defined in Equation (8). We set the batch size to 128, the momentum to 0.9, and the decay to 0.0005. The learning rate schedule is as follows: For 1~10000 steps, we start the training with a learning rate of 10^{-3}. Then we decreased the learning rate with 10^{-4} for 10001~15000, 10^{-5} for 15001~20000. We set the threshold $t = 0.1 \sim 0.5$. The code was downloaded and modified from https://github.com/pjreddie/darknet.git.

After training, we computed AP and FPS of the two networks to compare their testing performance. The experimental results are listed in Table 4.

Furthermore, Table 5 shows the detection performance more directly. We added about 22931 negative images without fire (this dataset will be described in the next part) to the testing set (1015 positive images with fire). All the images were detected by the trained networks. Fig. 12 shows some instances of detection results. From these results, we can find that the trained networks may generate some false alarms by detecting red/yellow/orange objects similar with fire. Actually, other fire detection methods also faced this problem, that is "false alarm problem". To overcome it, we added a network to distinguish whether the generated region is a real fire region or not. Because this step tried to detect fire regions as much as possible, recall is more important than precision. Therefore, in order to achieve high recall rate, we decreased the detection threshold with $t = 0.2$.

3.2 Region classification

3.2.1 Dataset

To overcome the false alarm problem, we added the region classification step. Because all of the false alarms were generated from things that looks like a flame in color, we considered many objects that may appear in cameras, such as red/orange/yellow clothes, vehicles, signboards, helmets, flowers, umbrellas, lights, sun, moon, etc. We expanded our dataset with a lot of fire-like images as negative images. Then all of the images, 5075 fire images and 22931 non-fire images, were detected by using the above trained detection model and generated a lot of regions, 6848 fire regions and 19543 fire-like regions (see Fig. 13).

3.2.2 Network

The networks for this step are shown in Fig. 14. Also for comparison, we used three networks, Inception-V3, ResNet-50 and Xception. The input of the networks is a region image generated from the fire detection step. The region will be resized to 256×256 with 3 channels (RGB) for network computation. The output of the networks is different from the previous step. In this step, the output is a 2-length vector for each region. They denote the probabilities of fire and fire-like.

3.2.3 Result

In the region classification task, we also use the confusion matrix to evaluate the classification accuracy and use Frames Per Second (FPS) to evaluate the detection speed. After training, we listed the results of the three different networks (Inception-V3, ResNet-50 and Xception) to compare their testing performance. The experimental results are listed in Table 6.

In the end, we combined the fire detection step and the region classification step. Considering the accuracy and the speed, we chose Darknet Network II and Xception as the final networks. The whole image dataset includes 5075 fire images and 22931 non-fire images. After the first step, the network detected 5011 fire images and 9288 non-fire images where fire may exist. After the second step, 4992 fire images were detected correctly and 31 non-images were detected to have fire incorrectly. The final result is listed in Table 7. The time for detecting one frame only costs 27.4ms and the FPS can achieve 36. Besides, these two detection steps are implemented only if moving objects exist after background subtraction step. Therefore, the speed can meet the need of industrial application totally.

4. Conclusions

In this article, we present an intelligent fire detection approach through cameras based on computer vision methods. First, videos from surveillance cameras are processed frame by frame through a background subtraction method. If moving objects appear, the frame will be detected whether fire exists or not by the trained networks. Once fire appears, the networks will output the coordinates of the fire region and mark it with a white rectangle box. Simultaneously, the frame where the fire region is localized will be immediately sent to safety supervisors as a fire alarm. Compared with other CNN based classification approaches, our approach mainly has three improvements:

(1) To reduce the CNN computation, we added a motion detection method based on background subtraction. Only if moving objects appear, the following CNN computations will be implemented.
(2) To replace the hand-designed feature extractors or the slide window methods for ROI generation, we used an object detection method based on YOLO to generate fire regions directly.
(3) To avoid the false alarm problem, we considered carefully and focused on the classification between fire images and fire-like images using CNN.

The proposed method is developed for practical application in chemical factories and other high-fire-risk industries. The performance of our approach can meet the needs of real-time fire detection on the precision and the speed. Its industrial deployment will help detect fire at the very early stage, facilitate the emergency management and therefore significantly contribute to loss prevention.

References

Celik, T., 2010. Fast and efficient method for fire detection using image processing. ETRI J. https://doi.org/10.4218/etrij.10.0109.0695

Chen, T.H., Wu, P.H., Chiou, Y.C., 2004. An early fire-detection method based on image processing, in: Proceedings - International Conference on Image Processing, ICIP. https://doi.org/10.1109/ICIP.2004.1421401

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. https://doi.org/10.1109/CVPR.2017.195

Ebert, J., 2009. Computer vision based method for fire detection in color videos. Int. J. imaging.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. https://doi.org/10.1007/s11263-009-0275-4

Frizzi, S., Kaabi, R., Bouchouicha, M., Ginoux, J.M., Moreau, E., Fnaiech, F., 2016. Convolutional neural network for video fire and smoke detection, in: IECON Proceedings (Industrial Electronics Conference). https://doi.org/10.1109/IECON.2016.7793196

Girshick, R., Donahue, J., Darrell, T., Berkeley, U.C., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. https://doi.org/10.1016/j.nima.2015.05.028

Girshick, R., 2015. Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision. https://doi.org/10.1109/ICCV.2015.169

Hayman, Eklundh, 2003. Statistical background subtraction for a mobile observer, in: Proceedings Ninth IEEE International Conference on Computer Vision. https://doi.org/10.1109/ICCV.2003.1238315

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR.2016.90

Homg, W., Peng, J., Chen, C., 2005. A New Image-Based Real-Time Flame Detection. IEEE Int. Conf. Netw. https://doi.org/10.1109/ICNSC.2005.1461169

KaewTraKulPong, P., Bowden, R., 2002. An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection, in: Video-Based Surveillance Systems. https://doi.org/10.1007/978-1-4615-0913-4_11

Krizhevsky, A., Hinton, G.E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. Neural Inf. Process. Syst. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., 1989. Backpropagation Applied to Handwritten Zip Code Recognition. Neural Comput. https://doi.org/10.1162/neco.1989.1.4.541

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature. https://doi.org/10.1038/nature14539

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: Single shot multibox detector, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-319-46448-0_2

Maksymiv, O., Rak, T., Peleshko, D., 2017. Real-time fire detection method combining AdaBoost, LBP and convolutional neural network in video sequence, in: 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics, CADSM 2017 - Proceedings. https://doi.org/10.1109/CADSM.2017.7916148

Marbach, G., Loepfe, M., Brupbacher, T., 2006. An image processing technique for fire detection in video images. Fire Saf. J. https://doi.org/10.1016/j.firesaf.2006.02.001

Narwani, S., Selva Kumar, A., 2016. Real-time fire detection for video surveillance applications using a combination of experts based on color, shape and motion. Int. J. Control Theory Appl. https://doi.org/10.1109/TCSVT.2015.2392531

Nolan, D.P., 2018. Handbook of Fire and Explosion Protection Engineering Principles for Oil, Gas, Chemical and Related Facilities: 4th Edition. https://doi.org/10.1016/C2017-0-04314-8

Power, P.W., Schoonees, J. a, 2002. Understanding Background Mixture Models for Foreground Segmentation Understanding Background Mixture Models for Foreground Segmentation. Image Vis. Comput. https://doi.org/10.1.1.9.5222

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. (YOLO) You Only Look Once, in: Proceedings - 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR.2016.91

Redmon, J., Farhadi, A., 2017. YOLO9000: Better, faster, stronger, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. https://doi.org/10.1109/CVPR.2017.690

Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. https://doi.org/10.1109/TPAMI.2016.2577031

Shi L., Long F., Lin C., Zhao Y., 2017. Video-Based Fire Detection with Saliency Detection and Convolutional Neural Networks. In: Cong F., Leung A., Wei Q. (eds) Advances in Neural Networks - ISNN 2017. ISNN 2017. Lecture Notes in Computer Science, vol 10262. Springer, Cham. https://doi.org/10.1007/978-3-319-59081-3_36

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the

IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy, C., Vanhoucke, V., Shlens, J., Wojna, Z., 2016. Rethinking the Inception Architecture for Computer Vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR.2016.308

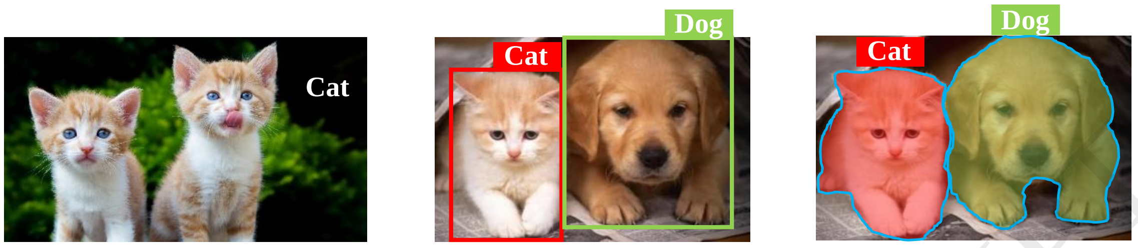
Töreyin, B.U., Dedeoğlu, Y., Güdükbay, U., Çetin, A.E., 2006. Computer vision based method for real-time fire and flame detection. Pattern Recognit. Lett. https://doi.org/10.1016/j.patrec.2005.06.015

Wang, Z., Wang, Z., Zhang, H., Guo, X., 2017. A novel fire detection approach based on CNN-SVM using tensorflow, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-319-63315-2_60

Zhang, Q., Xu, J., Xu, L., Guo, H., 2016. Deep Convolutional Neural Networks for Forest Fire Detection. 2016 Int. Forum Manag. Educ. Inf. Technol. Appl. 568–575. https://doi.org/10.2991/ifmeita-16.2016.105

Zivkovic, Z., Van Der Heijden, F., 2006. Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recognit. Lett. https://doi.org/10.1016/j.patrec.2005.11.005

Fig. 1 Three tasks in computer vision domain.



(a) Image classification. (b) Object detection. (c) Instance segmentation.

Fig. 2 Framework of the intelligent video fire detection approach.

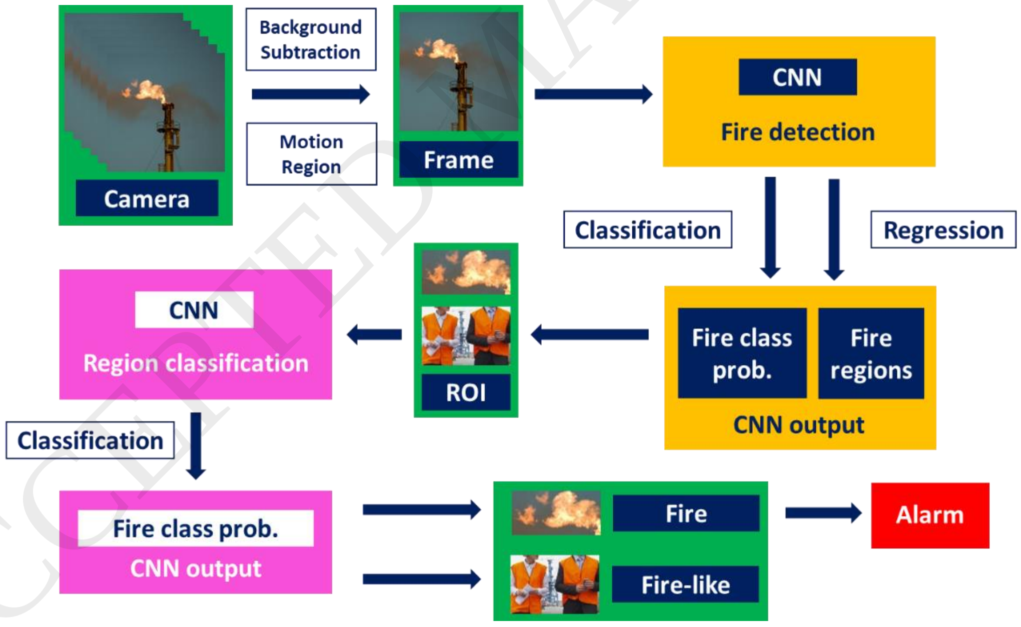
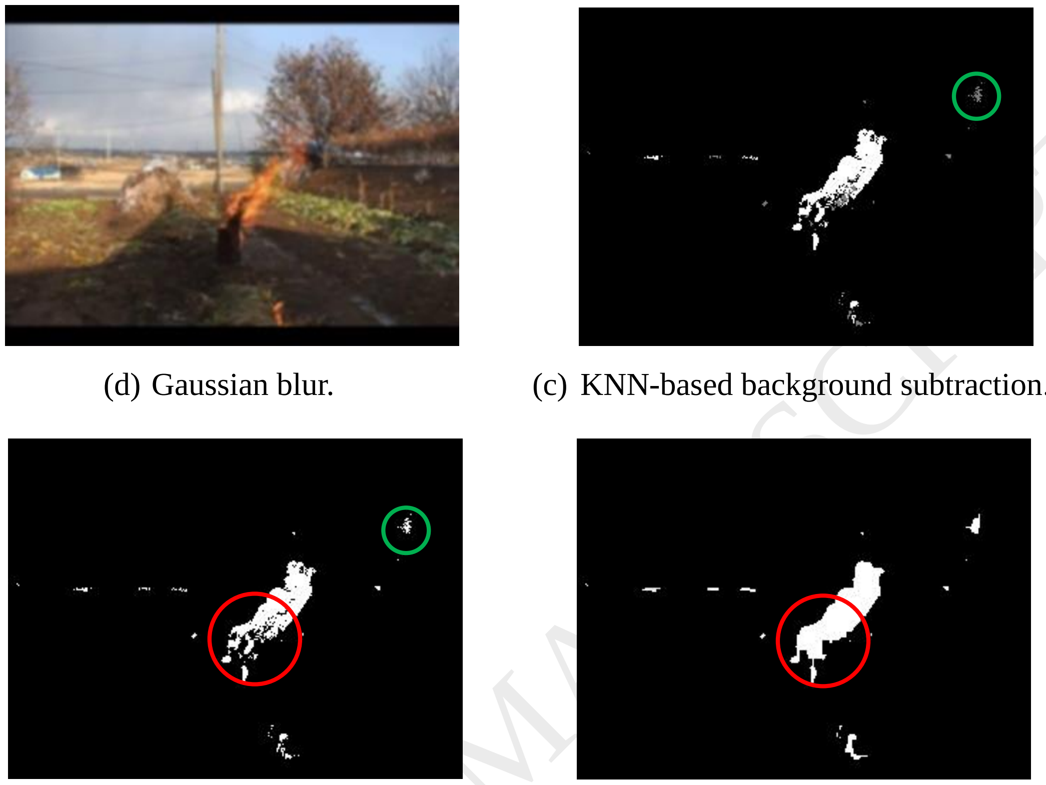


Fig. 3 Background subtraction process for video streams.



(d) Gaussian blur. (c) KNN-based background subtraction.

(b) Binarization. (a) Closing.

Fig. 4 Grid partition and prediction of YOLO.

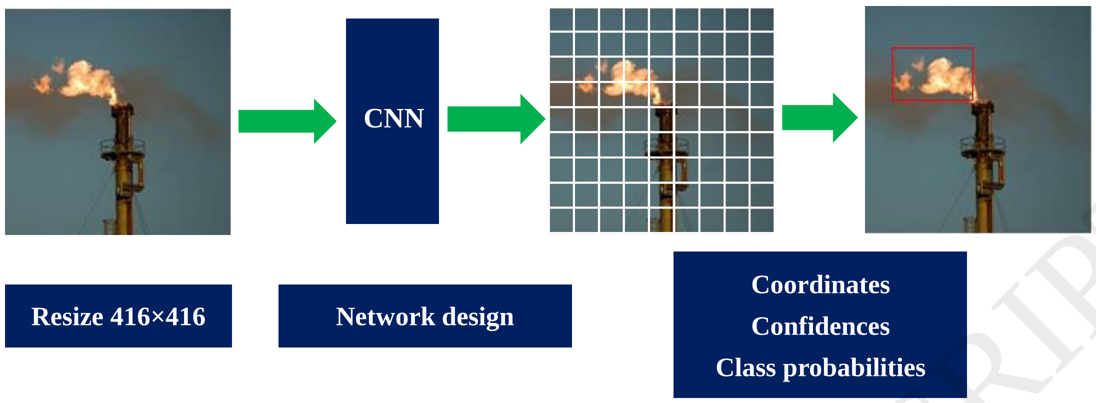


Fig. 5 Location predictions of bounding boxes with anchor boxes (Redmon and Farhadi, 2017).

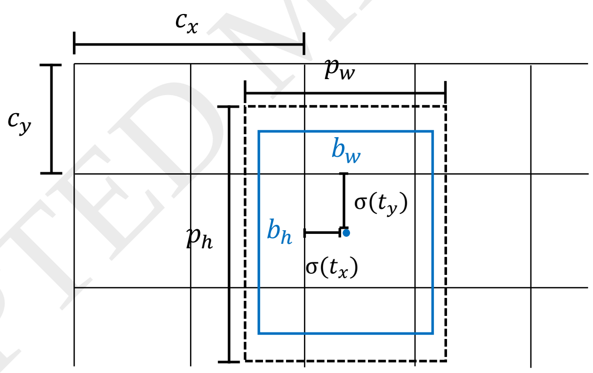
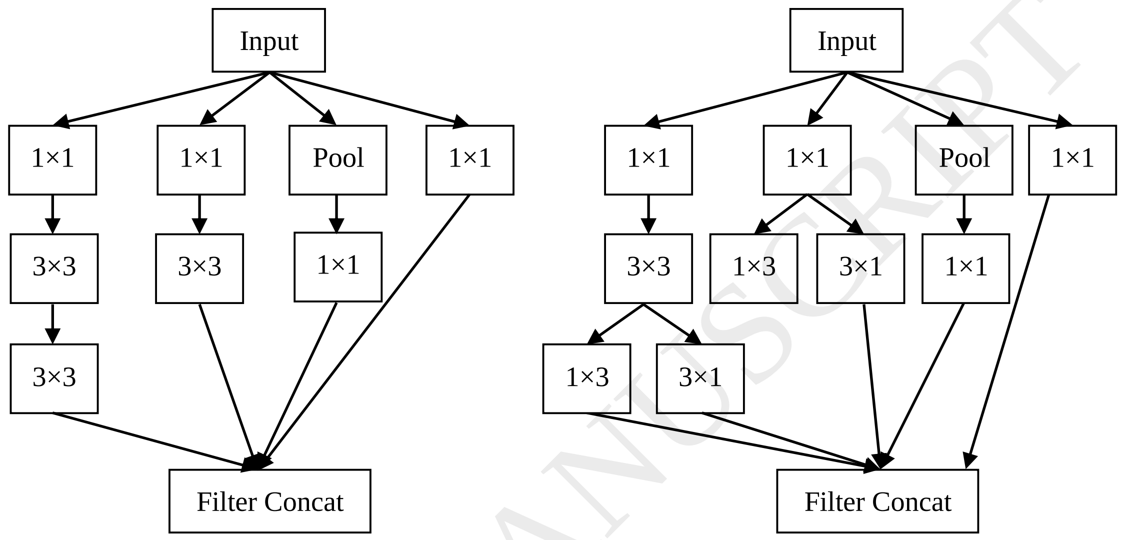
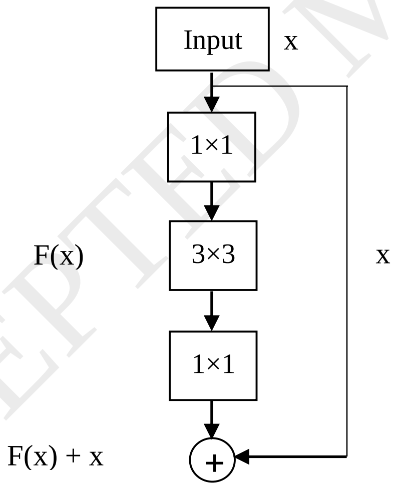
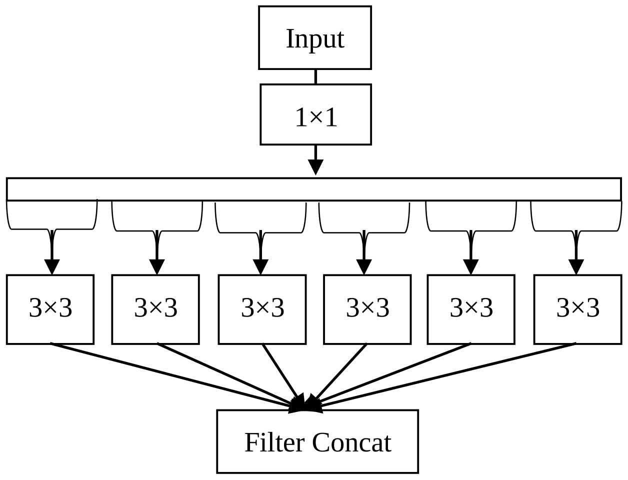


Fig. 6. Inception, residule and xception modules.



(a) Some inception modules (Szegedy et al., 2016).





(a) Residual module (He et al., 2016).

(c) Xception module (Chollet, 2017).

Fig. 7 Fire detection dataset.

Fig. 8 Detection Network I and II for fire detection.

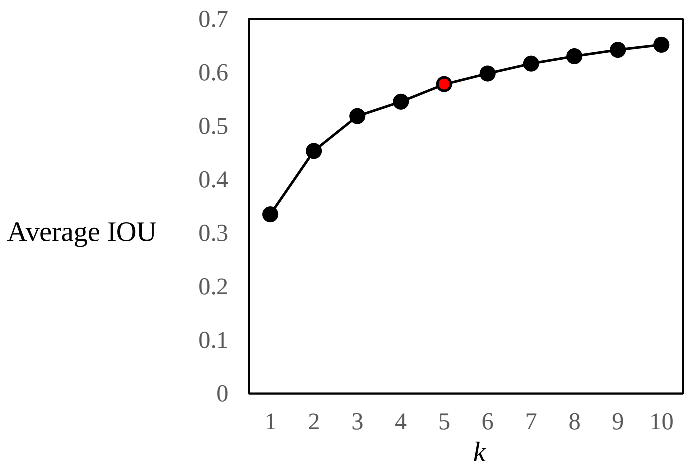
Detection Network I

Type	Filters	Size/Stride	Output
Convolutional	16	3×3	416×416
Maxpool		2×2 / 2	208×208
Convolutional	32	3×3	208×208
Maxpool		2×2 / 2	104×104
Convolutional	64	3×3	104×104
Maxpool		2×2 / 2	52×52
Convolutional	128	3×3	52×52
Maxpool		2×2 / 2	26×26
Convolutional	256	3×3	26×26
Maxpool		2×2 / 2	13×13
Convolutional	512	3×3	13×13
Maxpool		2×2 / 1	13×13
Convolutional	1024	3×3	13×13
Convolutional	1024	3×3	13×13
Convolutional	30	1×1	13×13

Detection Network II

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	416×416
Maxpool		2×2 / 2	208×208
Convolutional	64	3×3	208×208
Maxpool		2×2 / 2	104×104
Convolutional	128/64/128	3×3	104×104
Maxpool		2×2 / 2	52×52
Convolutional	256/128/256	3×3	52×52
Maxpool		2×2 / 2	26×26
Convolutional	512/256/512/256/512	3×3	26×26
Maxpool		2×2 / 2	13×13
Convolutional	1024/512/1024/512/1024	3×3	13×13
Convolutional	1024	3×3	13×13
Convolutional	1024	3×3	13×13
Convolutional	64	1×1	26×26
Reorg	256	/ 2	13×13
Concat	1024+256		13×13
Convolutional	1024	3×3	13×13
Convolutional	30	1×1	13×13

Fig. 9 Clustering box dimensions on fire image dataset.



Average IOU
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
k

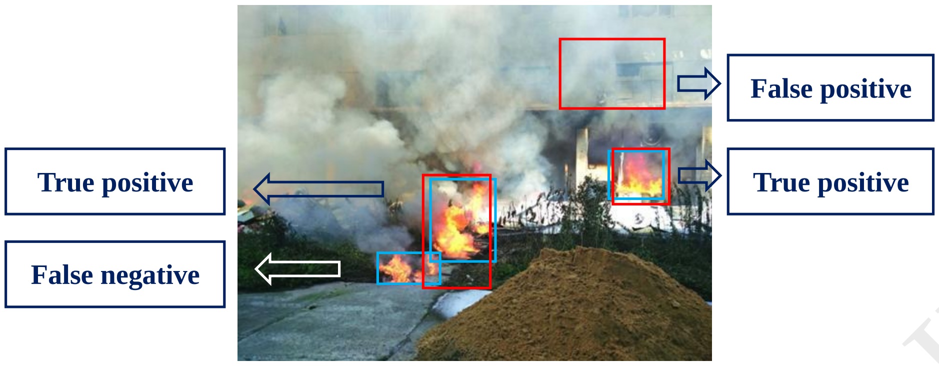


Fig. 10 The definition of TP, FP, FN.

Fig. 11 The definition of *AP*.

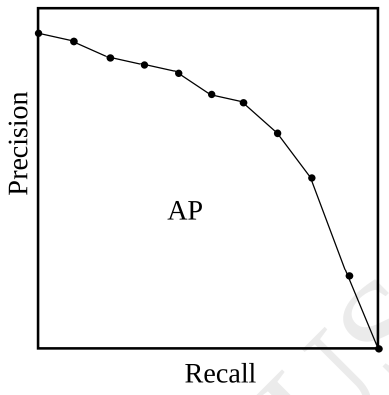
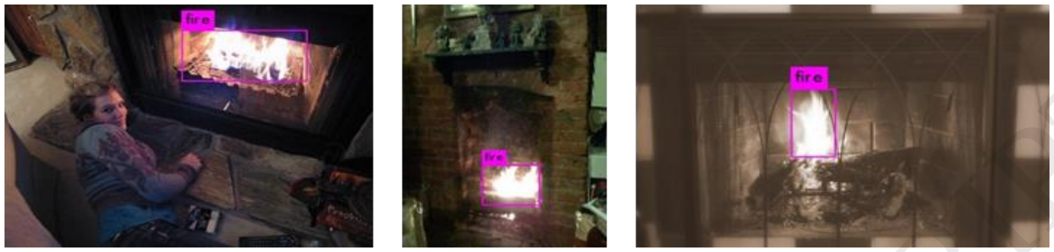
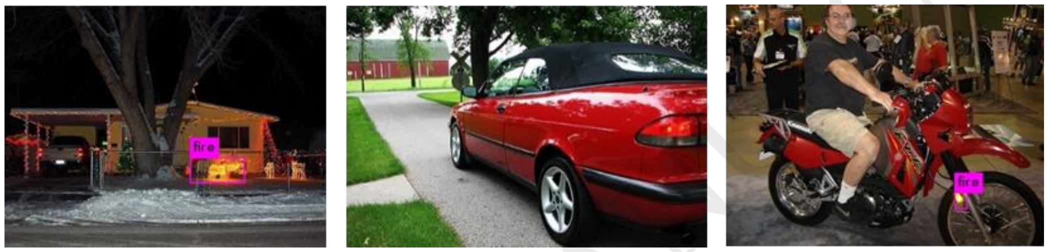


Fig. 12 Detection results.

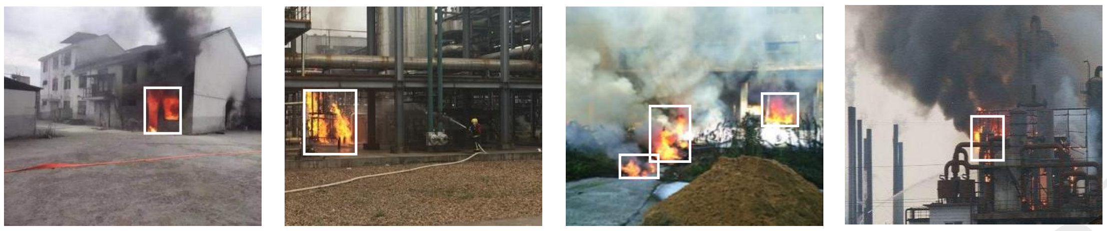


(b) Positive images with fire.

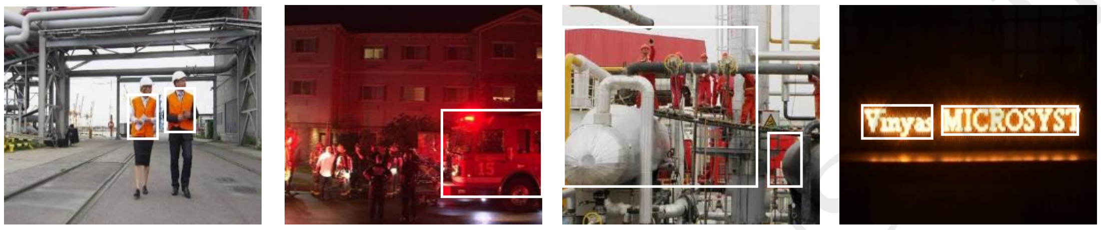


(a) Negative images without fire.

Fig. 13 Region classification dataset.



(b) Positive images and fire regions.



(a) Negative images and fire-like regions.

Fig. 14. Inception-V3, ResNet-50 and Xception architectures.

Inception-V3

Type	Filters	Size/Stride	Output
Convolutional	32	3×3 / 2	127×127
Convolutional	32	3×3 / 1	125×125
Convolutional	64	3×3 / 1	125×125
Maxpool		3×3 / 2	62×62
Convolutional	80	1×1 / 1	62×62
Convolutional	192	3×3 / 1	60×60
Maxpool		3×3 / 2	29×29
Inception × 3	256/288/288		29×29
Inception × 5	768/768/768/768/768		14×14
Inception × 2	1280/2048/2048		6×6
Average Pool			2048
Dense			2

ResNet-50

Type	Filters	Size/Stride	Output
Convolutional	64	7×7 / 2	128×128
Maxpool		3×3 / 2	64×64
Residual × 3	64/64/256		64×64
Residual × 4	128/128/512		32×32
Residual × 6	256/256/1024		16×16
Residual × 3	512/512/2048		8×8
Average Pool			2048
Dense			2

Xception

Type	Filters	Size/Stride	Output
Convolutional	32	3×3 / 2	127×127
Convolutional	64	3×3 / 1	125×125
Block	128		63×63
Block	256		32×32
Block	728		16×16
Block × 8	728		16×16
Block	1024		8×8
SeparableConv.	1536		8×8
SeparableConv.	2048		8×8
Average Pool			2048
Dense			2

Table 1 The 20 largest industry losses since 1988 (Nolan, 2018).

Date	Industry segment	Incident	Location	Property loss[a]
04.20.10	Production	Blow-out, explosion, **fire**, and pollution, vessel destruction	Gulf of Mexico, United States	>15, 000
07.07.88	Production	Explosion and **fire**, platform destruction	North Sea, United Kingdom	1600
10.23.89	Petrochemical	Vapor cloud explosion and **fire**	Pasadena, TX, United States	1300
04.01.15	Production	**Fire** on offshore complex	Bay of Campeche, Mexico	1000
03.19.89	Production	Explosion and **fire**	Gulf of Mexico, United States	750
09.12.08	Refining	Hurricane	Texas, United States	750
06.04.09	Production	Collision	North Sea, Norway	750
08.23.91	Production	Structural failure, sinking, destruction of platform hull	Sleipner North Sea, Norway	720
05.15.01	Production	Explosion, **fire** and vessel sinking	Campos Basin, Brazil	710
09.28.98	Gas processing	Vapor cloud explosion	Victoria, Australia	680
04.15.03	Production	Riot	Escravos, Nigeria	650
04.28.88	Production	**Fire**	Campos Basin, Brazil	640
09.21.01	Petrochemical	Explosion	Toulouse, France	610
06.25.00	Gas processing	Vapor cloud explosion	Mina Al-Ahmadi, Kuwait	600
05.04.88	Petrochemical	Explosion	Nevada, United States	580
01.19.04	Gas processing	Explosion and **fire**	Skikda, Algeria	580
05.05.88	Refining	Vapor cloud explosion	Louisiana, United States	560
11.01.92	Production	Mechanical damage	Northwest Shelf, Australia	470
12.25.97	Gas processing	Explosion and **fire**	Sarawak, Malaysia	430
07.27.05	Production	Explosion and **fire**	Mumbai High, India	430

[a] US dollars (MM), adjusted for inflation.

Table 2. Widths and heights of anchor boxes by using K-means clustering.

Anchor	W*	H*
1	0.238	0.292
2	0.754	0.819
3	0.325	0.572
4	0.094	0.117
5	0.597	0.407

W*, H* denote width and height relative to the image respectively.

Table 3. Confusion matrix.

	Actual Class		
Predicted Class		1	0
	1	True Positive	False Positive
	0	False Negative	True Negative

Table 4. AP and FPS with two networks.

	Size	AP				FPS
Threshold *t*		0.5	0.3	0.2	0.1	
Detection Network I	60M	0.563	0.714	0.753	0.781	151
Detection Network II	192M	0.604	0.750	0.794	0.826	73

Table 5. Details of fire detection for fire and non-fire images.

(a) Detection Network I.

	Actual Class				
Predicted Class	Size (60M)	Fire		Non-fire	
	Fire	87.4%	94.8%*	16.3%	29.7%*
	Non-fire	12.6%	5.2%*	83.7%	70.3%*

(b) Detection Network II.

	Actual Class				
Predicted Class	Size (192M)	Fire		Non-fire	
	Fire	95.8%	98.7%*	34.4%	40.5%*
	Non-fire	4.2%	1.3%*	65.6%	59.5%*

*The values without * denote $t = 0.3$ and the values with * denote $t = 0.2$.

Table 6. Details of region classification for fire and fire-like regions.

(a) Inception-V3.

	Actual Class		
Predicted Class	Size (83.9M) FPS (34)	Fire	Fire-like
	Fire	95.3%	0.9%
	Fire-like	4.7%	99.1%

(b) ResNet-50.

	Actual Class		
Predicted Class	Size (90.4M) FPS (55)	Fire	Fire-like
	Fire	96.2%	1.7%
	Fire-like	3.8%	98.3%

(c) Xception.

	Actual Class		
Predicted Class	Size (79.9M) FPS (73)	Fire	Fire-like
	Fire	96.6%	1.2%
	Fire-like	3.4%	98.8%

Table 7. Final result of Darknet Network II and Xception.

	Actual Class		
Predicted Class	FPS 36	Fire	Non-fire
	Fire	4992 (98.4%)	31 (0.1%)
	Non-fire	83 (1.6%)	22900 (99.9%)