Computers and Electronics in Agriculture 167 (2019) 105029

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Real-time forest smoke detection using hand-designed features and deep learning

Yingshu Peng[a,*], Yi Wang[b]

[a] College of Forestry, Nanjing Forestry University, Nanjing 210037, PR China
[b] Jiangsu Wiscom Technology Co. Ltd, Nanjing 210021, PR China

ARTICLE INFO

Keywords:
Forest fire
Smoke detection
Deep learning
Feature extraction
Image processing

ABSTRACT

In this paper, a fast and accurate video smoke detection algorithm is proposed to extract a powerful feature representation of forest fire. Because of the irregular shape, the variation of colors and hard-to-depict texture of smoke, it is incredibly challenging to detect smoke quickly and accurately in a complex environment. We proposes a smoke detection algorithm that combines manual features and deep learning features, an approach that differs from those of existing detection methods. First, a manual design algorithm is used to extract the suspected smoke area, which is then employed as input to the improved very small deep neural network SqueezeNet model to achieve smoke detection. Extensive detection experiments are conducted, and the results showed that the proposed method is able to differentiate smoke from other challenging objects. Compared with the existing forest fire algorithms, the proposed method has higher classification accuracy and speed, with a more comprehensive application range and lower requirements for hardware devices, and these features are worthy of promotion.

1. Introduction

Forest fire is a sudden and destructive natural disaster, which is a sudden occurrence that is difficult to control. It causes air pollution and leads to the loss of plant and animal resources. It is estimated that by 2030, the fire will destroy about half of the world's forests (Luo et al., 2013). Therefore, rapid detection of forest fire can reduce damage to ecosystems, infrastructure, and casualties. Fire detection can be subdivided into flame detection and smoke detection. Obviously, smoke is always the first sign to be seen in the case of a fire, and it has a wider range of motion than flames. Thus, smoke detection is a more efficient clue as an early fire warning.

Smoke has three main characteristics: physical properties, visual properties, and dynamic properties. Traditional forest smoke detection is based on the physical properties of smoke and requires the collection of smoke particles, temperature and humidity to trigger an alarm. However, it can only be applied to closed spaces, and the temperature and humidity of the forest vary greatly with the weather, which is often prone to false positives (Tang and Ge, 2002). The other two characteristics can be observed through a camera, so computer vision-based smoke detection technology has emerged as the times have required. Compared with the customary physical sensor smoke detector, computer vision technology has many advantages. For example, it can be used in an outdoor environment and can provide the position and intensity of smoke generation for early smoke detection.

At present, computer vision-based smoke detection algorithms focus mainly on two aspects: (1) hand-designing complex smoke features and then employing a traditional machine learning algorithm for image classification; and (2) automatic feature extraction and classification based on a deep learning network. As shown in Fig. 1 (a), the work of the smoke detection algorithm based on manually designed complex features mainly follows the object detection framework (Toreyin, 2006; Tian, 2011; Calderara, 2011), which is roughly divided into three steps: image preprocessing, feature extraction and classification. Each frame of the input video is subdivided into several blocks. The principle of this method is exemplified by building the specific complex smoke features that are manually designed and employed to classify the block into smoke and non-smoke with machine learning classifiers such as Support Vector Machine (SVM), AdaBoost, and Random Forest. However, hand-designed smoke features mostly rely on the prior knowledge of the designer. These features are highly targeted and interpretable but exhibit poor scalability. Moreover, they do not use big data to describe the sophisticated features of smoke, resulting in lower overall accuracy and implementations that can only be used in simple environments.

Smoke detection algorithms based on deep neural networks can address this problem. As shown in Fig. 1(b), deep learning can

* Corresponding author.
E-mail addresses: yspeng@njfu.edu.cn (Y. Peng), ywang4@wiscom.com.cn (Y. Wang).

https://doi.org/10.1016/j.compag.2019.105029
Received 24 July 2019; Received in revised form 21 September 2019; Accepted 29 September 2019

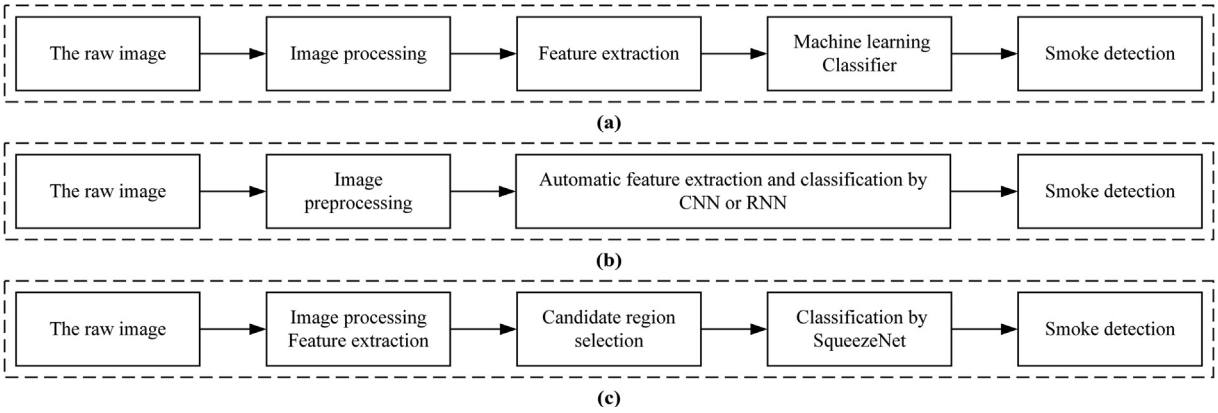

Fig. 1. Comparison of three smoke detection technologies, (a) traditional smoke image feature extractors, (b) deep neural network for image smoke detection, (c) the proposed method.

automatically extract features from extensive data and classify them, avoiding complex preprocessing and obtaining many features that cannot be manually designed. Due to the complex structure and the massive number of network parameters of a deep neural network, smoke detection based on deep learning can significantly improve detection accuracy. Nevertheless, it has slow speed and high hardware requirements, which implies an increase in the operating costs of the detection algorithm. Furthermore, deep learning relies entirely on data to build models, and the false-positive rate will increase in environments that differ from the training data set. There are many interfering factors such as clouds, fog, haze, and plants, in actual situation. Therefore, regardless of whether the detection algorithm is based on hand designed features or deep learning, it is challenging to detect smoke quickly, accurately, and economically in complex environments. This article proposes a new method to address this problem, which implements smoke detection by extracting concise features and building an incredibly smaller neural network.

Deep learning has been used for smoke detection (Zhang et al., 2018; Yin et al., 2017; Frizzi et al., 2016; Yin et al., 2018). For example, Faster R- convolutional neural network (CNN), an object detection algorithm, can monitor smoke from forest fires in the field, which improved the smoke alarm rate; however, the false alarm rate was high, and the detection time was extended (Zhang et al., 2018). A deep normalized convolutional neural network (DNCNN) for smoke image detection was proposed in (Yin et al., 2017). The network improved the convolutional layer in a traditional CNN with a batch-normalized convolutional layer, which effectively solved the problem of overfitting and improved the detection precision. However, the method used many network layers, resulting in slow detection speed and implementations in only simple scenarios. The sliding window was used to sample the video frame, and then the CNN was used for classification (Frizzi et al., 2016). The method increased the complexity of the algorithm and reduced the scope of smoke detection such that the network was capable of detecting only the local area rather than the whole smoke object, and while the overall detection speed of the algorithm was improved, the possibility of a false alarm was increased. In addition to CNNs, the recurrent neural networks (RNNs) has been used to detect a smoke video to avoid a false alarm (Yin et al., 2018). Compared with other deep neural networks, the RNNs achieve a higher accuracy of smoke detection but also incur a longer detection time and higher hardware requirements.

In summary, several studies of the speed and accuracy of smoke detection improvement have achieved excellent performance in a particular scenario. However, there is currently no method that can be applied accurately and quickly to or used widely in complex environments.

Forest fires have complex, irregular motion characteristics. The texture characteristics of the smoke caused by wind and burning materials are unstable. Color of the smoke is complex and variable, and it is easily confused with the colors of trees, clouds and sky. Therefore, monitoring the dynamic characteristics of forest fire smoke and achieving forest fire alarms is the most reasonable and accurate. The purpose of this paper is to design a practical algorithm that monitors a complex natural scene in real time and detects ae smoke in a timely manner in case of a fire. Thus, to meet the requirements mentioned above, we integrate manual features and deep learning features to enhance the applicability of the algorithm comprehensively, obtain more useful features and design a smaller deep learning network model to improve the detection accuracy and speed.

Our method can detect forest smoke quickly, accurately, and economically in various natural environments. The main structure is similar to Fig. 1(c). Specifically, (1) we build the motion model based on the motion characteristics of smoke through literature analysis to reduce the false alarm rate. The Gaussian mixture model algorithm detects the motion block in the image, and each motion block is morphologically processed separately; (2) Simple and effective smoke features are extracted to distinguish smoke from other moving objects with similar characteristics to improve the applicability of the model; (3) The small SqueezeNet CNN is optimized to improve detection accuracy and detection speed, and reduce the hardware requirement.

2. Related work

2.1. SqueezeNet neural network

Recently, the main research direction regarding deep CNNs has been to improve the accuracy and the number of network layers. Existing deep CNNs have a large volume and a large number of layers, which leads to slow prediction speed and high hardware requirements. However, for the problem of smoke classification, there are only two classification categories: smoke and no smoke; consequently, the network does not need a large number of layers. Therefore, because of these factors, smoke detection algorithms based on deep learning are very difficult to apply in real environments.

Deep learning can achieve a high classification accuracy with a small number of network layers, whereas excessive network layers will lead to overfitting and low accuracy. Thus, it is necessary to develop a very small deep learning network with high precision and a small network architecture. SqueezeNet is one type of very small deep CNN, which uses only one-fiftieth the number of parameters of AlexNet but achieves the same accuracy as AlexNet on ImageNet (Iandola et al., 2016). SqueezeNet has a faster training speed and test speed, has low hardware configuration requirements and can be widely deployed in different environments than traditional deep learning networks.

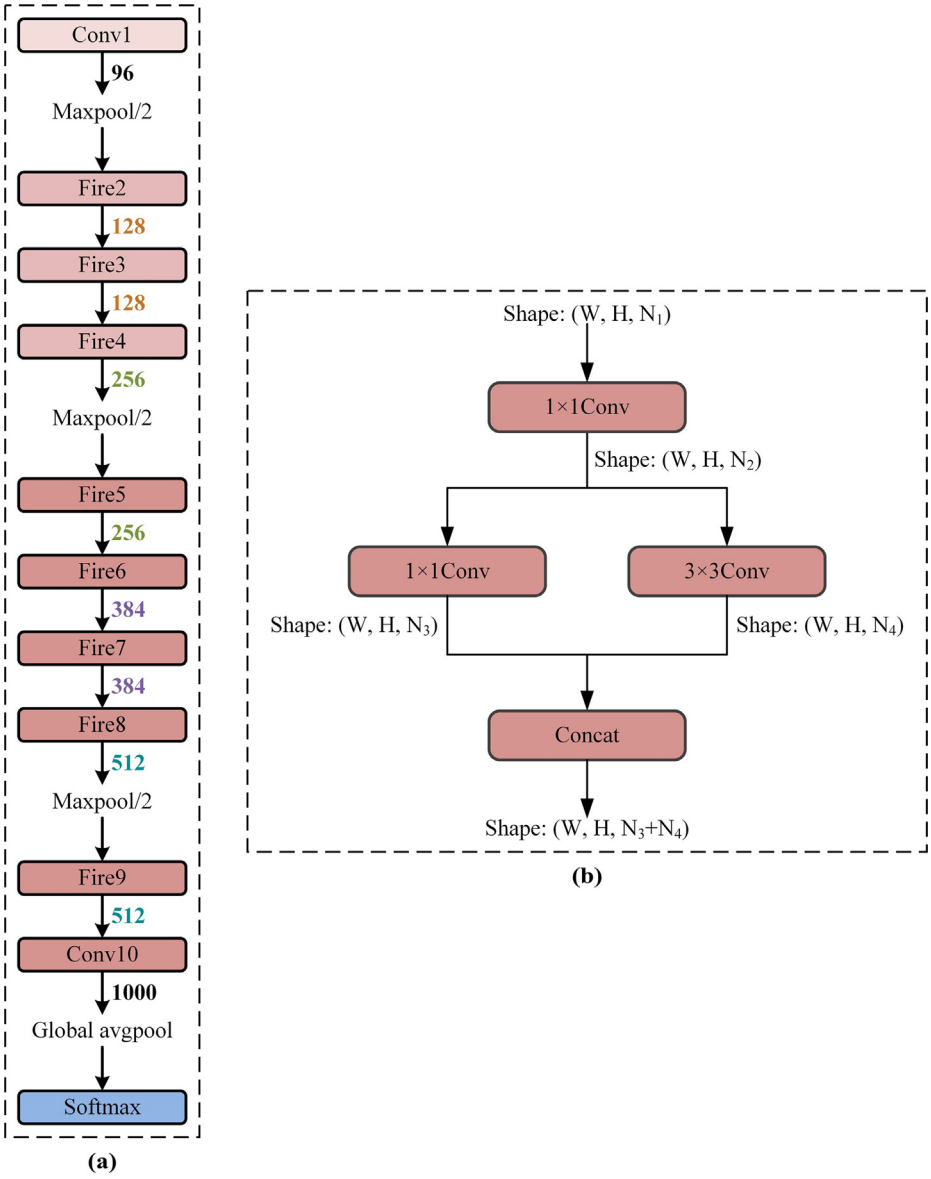

Fig. 2. SqueezeNet network overall structure, (a) SqueezeNet basic structure, (b) SqueezeNet fire module structure.

SqueezeNet implements a smaller and more efficient structure, mainly in terms of the following three aspects:

(1) The 3 × 3 convolution (Conv) kernel is replaced with a smaller 1 × 1 convolution kernel to reduce network parameters.
(2) The designed fire layer in SqueezeNet is used to minimize the parameters and model volume.
(3) The pooling layer is reduced. SqueezeNet only has three max-pooling layers and one global pooling layer.

Therefore, SqueezeNet has a larger output layer that retains more information and provides higher classification accuracy.

The network structure diagram of SqueezeNet and the structure of the fire layer are depicted in Fig. 2. We use Conv and Concat as abbreviations for convolution and concatenate, respectively. In addition, we set the input format of the data as width (W), height (H), channel (N).

The overall structure of SqueezeNet is shown in Fig. 2(a), SqueezeNet consists of several fire layers, and the SqueezeNet network uses a global pooling layer and a dropout layer after the fire 9 module to avoid overfitting, finally classifying with a softmax classifier. The network does not need to add a full connection layer due to the small number of pooling layers in the network and the backward position of the pooling layer, which significantly reduces the volume of SqueezeNet and decreases the difficulty of parameter adjustment. Fig. 2(b) shows the specific structure of the fire layer, the fire layer uses a 1 × 1 convolution layer for input compression. The network is then expanded using a combination of 1 × 1 and 3 × 3 convolutions to limit the number of input channels, speed up the detection, and reduce the model size.

2.2. Depthwise separable convolution

Although SqueezeNet reduces the model volume and accelerates the detection speed through the fire layers, the performance of the traditional convolution layer in SqueezeNet has limitations. As a result, the accuracy of the model is no better than that of a CNN. It is essential to optimize the SqueezeNet network. We use a depthwise separable convolution instead of a standard convolution to optimize the SqueezeNet network. Depthwise separable convolution can make the model lighter (fewer training weight parameters), faster (fewer floating-point operations), and more accurate.

The working principle of standard convolution extracts different

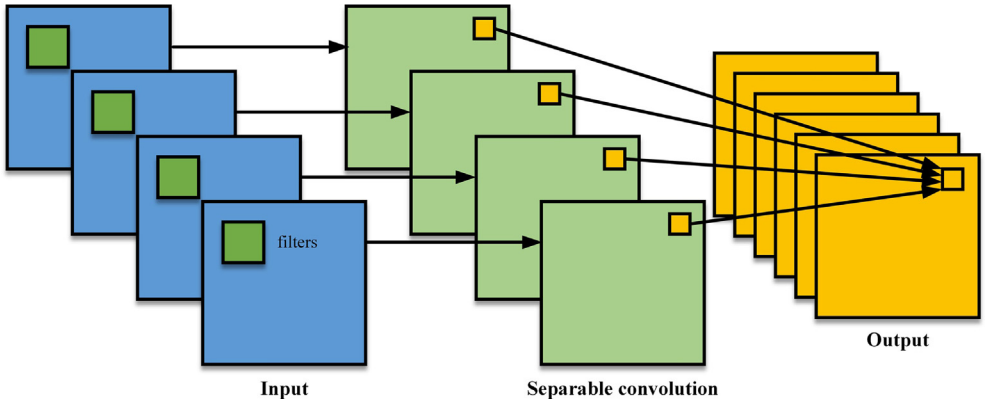

Fig. 3. Depthwise separable convolution structure.

features according to different convolution kernels by processing the input channel and the convolution region at the same time, and then outputs a characteristic image. The depthwise separable convolutions process the convolution region first and then merge the channel. The structure of the depthwise separable convolutions is specifically displayed in Fig. 3. The main working principle is equivalent to performing spatial convolution on each channel of the input. First, each channel is convolved with a convolution kernel, and then the 1×1 convolution kernel is used to perform the feature fusion channel to obtain the output. In contrast to standard convolution, the depthwise separable convolution enables separation of the channel and convolution regions, and the input feature map and the output feature map are connected one-to-one through a convolution operation. The feature extraction, feature fusion and separation can effectively reduce computational complexity and model capacity, achieve faster convolution calculation, and obtain more in-depth features (Chollet, 2017).

For an $N \times N$ standard conventional layer, the number of input channels is C_1, and the number of output channels is C_2, in the case of not including biases. The standard convolutional layer requires $C_2 \times C_1 \times N \times N$ parameters. However, the depthwise separation convolution layer requires only $C_1 \times N \times N + C_2 \times C_1 \times 1 \times 1$ parameters to achieve the same effect. The parameters of the depthwise separable convolution layer are significantly reduced. With the larger convolution kernel, the parameters used in the depthwise separable convolutions are much smaller than those used in ordinary convolution, and the effect on model optimization is more prominent.

2.3. Batch normalization (BN)

The training of a deep neural networks is a complicated process. When a layer of data changes slightly, the following hidden layer output will change drastically, resulting in a brand new data distribution. The deep learning network needs to re-adjust the learning rate and other parameters to adapt to the new data distribution, which causes the reduction of training speed and accuracy. An effective method to address this problem is to add a batch normalization layer in SqueezeNet. The core principle of the batch normalization layer is to unify the scattered data and normalize the output data of each layer such that the batch-normalized features have a distribution with a mean value of zero and a variance of one. More detailed information about batch normalization is presented in (Ioffe and Szegedy, 2015). The role of batch normalization as follows:

1. Stabilize the distribution of input data of each layer in the network and accelerate the training speed of the model.
2. Simplify the adjustment process of model parameters to make network learning more stable.
3. Alleviate the problem of gradient disappearance, which has a regularization effect on the neural network to some extent.

In summary, it is necessary to add the batch normalization layer in SqueezeNet.

3. Proposed smoke detection algorithm architecture

This paper proposes a method for the rapid and practical detection of smoke. The method comprises two modules: (1) the extraction of a suspected area of smoke based on image processing technology; and (2) the establishment of a rapid detection model for smoke employing SqueezeNet.

3.1. Steps of the proposed method

The paper proposes a smoke detection method based on manual feature extraction and deep learning, as shown in Fig. 4, and the specific steps are as follows:

Step 1: Obtain the video from the camera and extract the image frame-by-frame.
Step 2: Establish a background model to perform motion detection on the image.
Step 3: Perform noise processing and feature extraction on the moving block to identify the suspected area of the smoke.
Step 4: Design SqueezeNet networks with different structures and choose the structure of SqueezeNet with the best performance in the validation set.
Step 5: Test the performance of the algorithm through video analysis.
Step 2 and Step 3 yield the feature extraction module, and Step 4 yields a deep learning classification module. In this paper, we detect the real-time video data stream, and carry out background modelling based on the first few frames of video, and then monitor the smoke frame-by-frame.

3.2. Extractions of suspected smoke areas

Manually designed smoke features used in the reported literature are incredibly applicable to specific scene detection. However, complex features can cause the results of algorithms to be susceptible to environmental impacts. Moreover, the false negative rate of detection is extremely high in complex environments and cannot be applied to different scenarios. The direct use of the deep learning network for classification or target recognition will increase the amount of computations and the false positive rate and reduce the detection accuracy.

Therefore, our detection algorithm principle mainly implements deep learning, while manually designed features are employed secondarily. Manually designed features are practical and straightforward for obtaining the suspected smoke area, which can reduce false positives and improve the accuracy. The technical process is shown in Fig. 5.

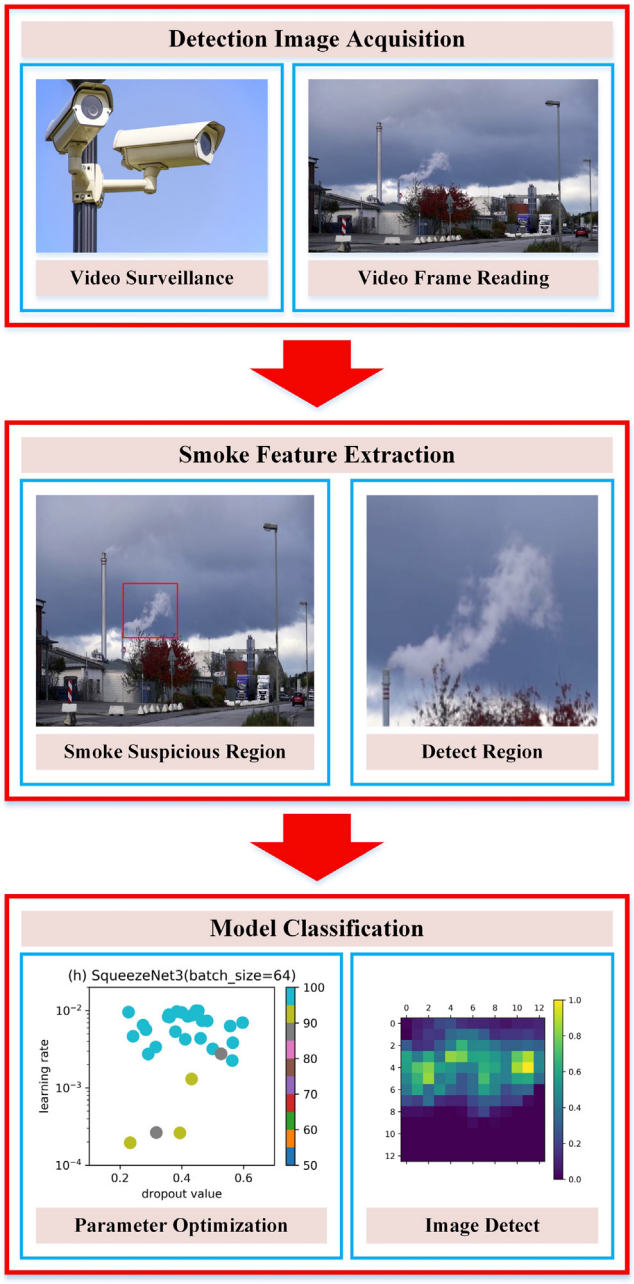

Fig. 4. Steps of our proposed method.

3.2.1. Motion detection

In general, smoke spreads slowly before the flame. Moving target detection can segment moving objects from the image, which is the premise of subsequent image processing. The current types of motion detection algorithms include the frame difference method, background subtraction methods, and the optical flow method. The frame difference method cannot completely extract the motion region, and the optical flow method has high computational complexity, while the background subtraction method can completely retrieve the motion region with less computation.

Therefore, this paper selects a motion detection algorithm built on a Gaussian mixture model (GMM) that is developed from a single Gaussian model. Since the moving detection algorithm based on GMM is a background subtraction method for moving target detection with a relatively stable background, which is very useful for extracting slowly moving objects (Manchanda and Sharma, 2016). The principle of Gaussian mixture modelling is as follows:

(1) Establishment of a single Gaussian model

The principle of the single Gaussian model is to assume that the probability of each pixel value in the obtained gray image obeys the Gaussian distribution. A Gaussian model is established for each pixel, in which the mean value and the variance of the pixels are preserved. We assume that the mean value, variance and standard deviation of the single Gaussian model of the pixel (x, y) in the t-th frame image are u (x, y, t), σ^2 (x, y, t), and σ (x, y, t), respectively. The process of motion detection by a single Gaussian model includes initialization of the model, updating of parameters, and detection.

(a) Model initialization

Model initialization is performed using the following formula:

$$\begin{cases} u(x, y, 0) = I(x, y, 0) \\ \sigma^2(x, y, 0) = std\_init^2 \\ \sigma(x, y, 0) = std\_init \end{cases} \quad (1)$$

where I $(x, y, 0)$ represents the value of the pixel (x, y) of the first image in the video image sequence, *std_init* is the initial constant, and *std_init* = 20, based on our practical experience.

(b) Updating parameters and detection

When reading a new picture, if the distance between the pixel value of a point (x, y) in time t and the mean of the background Gaussian model of that in time $t-1$ is less than λ times the standard deviation of the background Gaussian model of that in time $t-1$, the point is the background. Otherwise, it is in the foreground. If we suppose the result of foreground detection is *output* and the pixel value at the position of the point (x, y) at the time t is expressed as *output* (x, y, t), the calculation formula of *output* (x, y, t) is as follows:

$$output(x, y, t) = \begin{cases} 0, |I(x, y, t) - u(x, y, t-1)| < \lambda \times \sigma(x, y, t-1) \\ 1, otherwise \end{cases} \quad (2)$$

where λ is the initial constant.

The updating formula of the model is as follows:

$$\begin{cases} u(x, y, t) = (1-\alpha) \times u(x, y, t-1) + \alpha \times u(x, y, t) \\ \sigma^2(x, y, t) = (1-\alpha) \times \sigma^2(x, y, t-1) + \alpha \times [I(x, y, t) - u(x, y, t)]^2 \\ \sigma(x, y, t) = \sqrt{\sigma^2(x, y, t)} \end{cases} \quad (3)$$

In the above, α is a constant indicating the update rate, and α can make the model have good robustness when the background changes slowly.

(2) Establishment of a Gaussian mixture model

The single Gaussian model can only describe the simple moving environment and can easily make mistakes when the complex background is manifested in a multimodal form. However, a Gaussian mixture model can address this problem, as it is robust to multimodal form of the background. The principle of the hybrid GMM is to use multiple single Gaussian models as the background model of a pixel for motion detection. The necessary steps of the GMM algorithm are as follows:

(a) Define the pixel model

Each pixel is described by K single Gaussian background models:

$$P_{(p)} = \{[w_i(x, y, t), u_i(x, y, t), \sigma_i(x, y, t)^2]\}, \quad i = 1, 2, \ldots\ldots, K \quad (4)$$

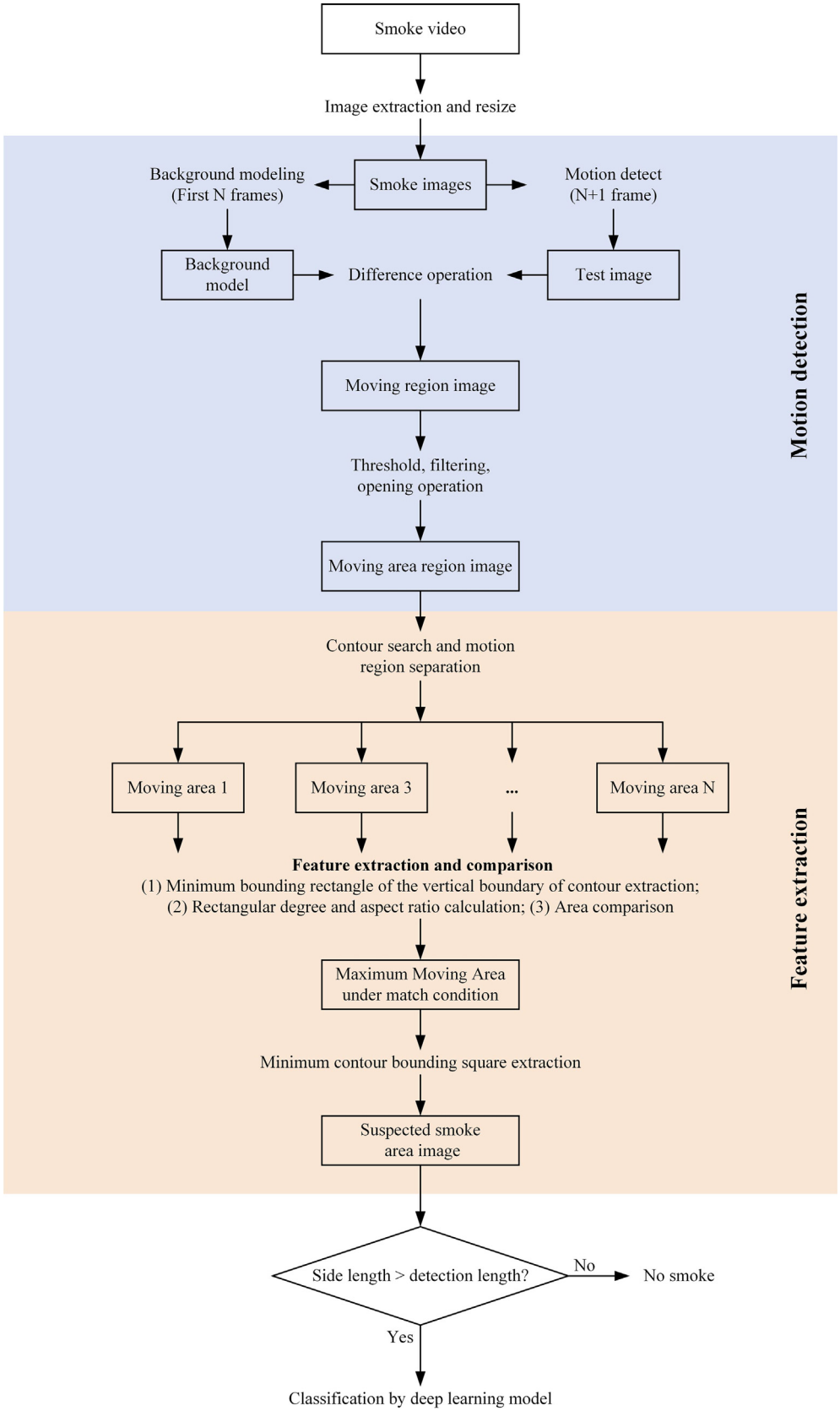

Fig. 5. Feature extraction process.

w_i (x, y, t) represents the weight of each model and satisfies the following:

$$\sum_{i=1}^{K} w_i(x, y, t) = 1 \tag{5}$$

(b) Update parameters and perform foreground detection

Step 1:

For a single Gaussian model i, if the pixel value of the newly read image at point (x, y) satisfies the following:

$$|I(x, y, t) - u_i(x, y, t)| \leqslant \lambda \times \sigma_i(x, y, t) \tag{6}$$

the point is judged to be in the background and Step 2 is performed. If not, the point is judged to be the foreground and Step 3 is performed.

Step 2:

To modify the weight of the single model matching the new pixel, the weight increment is as follows:

$$dw = \alpha \times (1 - w_i(x, y, t-1)) \tag{7}$$

and the new weight is expressed as follows:

$$\begin{aligned} w_i(x, y, t) &= w_i(x, y, t-1) + dw \\ &= w_i(x, y, t-1) + \alpha \times (1 - w_i(x, y, t-1)) \end{aligned} \tag{8}$$

At the same time, the mean and variance of the single Gaussian model matching the new pixel are corrected. After completing Step 2, the process directly transfers to Step 4.

Step 3:

If the new pixel does not match any of the single models, a new single model is added. We defined the weight of the newly added single Gaussian model as a small value (0.001 in the experiment). The mean value is the new pixel value, and the variance yields a large value (20 in this study). If the current number of single models has reached the maximum amount allowed (for absolute precision and speed, there are up to five single Gaussian models in the mixed Gaussian model), the single Gaussian model with the least importance in the current multi-model set is removed. Since the background model has the characteristics of significant weight (high frequency of background occurrence) and a small variance (small pixel value change), the importance is calculated in this article by the following formula:

$$sort\_key = \frac{w_i(x, y, t)}{\sigma_i(x, y, t)} \tag{9}$$

Step 4:

Every single Gaussian model is sorted in descending degree of importance under the assumption that the weights of the first N single models can satisfy the following:

$$\sum_{i=1}^{N} w_i(x, y, t) > T \tag{10}$$

Therefore, the n single models are used as background models and other models are deleted. According to our actual experience, T = 0.7 in this paper. Since the GMM requires setting set too many parameters, this paper adopts the adaptive method of model parameters in the mixed Gaussian model in (Zhivkovic, 2004) to reduce the computational complexity and improve the detection speed. The binary motion image is supplied by the above steps, where the pixel value of the point judged as the moving region (foreground) is 255, and the pixel value of the pixel point in the non-moving area (background) is 0.

However, due to environmental factors, there are obvious maximum or minimum regions in the moving binary image, and these regions are not moving areas but rather image noise, which will affect the detection accuracy. The random noise can be eliminated by image median filtering. Then, the image is processed by the morphological operation to remove small isolated dots as well as some minor disturbances such as leaves and the swaying of the vegetation, while also smoothing the contours of moving objects. Finally, morphological dilation is used to expand the motion detection area (Gonzalez and Woods, 2007).

3.2.2. Feature extraction

Usually, some distractors with a larger motion range, such as trees, pedestrians, vehicles, birds, clouds, and fog, will increase the difficulty of distinguishing smoke from distractors. Therefore, extraction of appropriate smoke features can enable the classifier to make a correct and rapid classification, which is extremely important for the whole smoke detection system.

When designing a smoke detection system, the feature extraction algorithm is required to be accurate and fast. To select features, we mainly consider that the following four conditions are met: (1) good distinction; (2) less computation to meet the requirements of real-time processing; (3) low complexity (more intricate features are more sensitive to environmental changes); and (4) small amount of storage requirement for the calculation.

Among the image features commonly used by traditional smoke feature extraction algorithms, color features and texture features are easily affected by light, and the calculation of frequency features and area change features is cumbersome (Tung and Kim, 2011).

We compare various smoke characteristics and select the aspect ratio feature and rectangularity feature of the motion area to extract the suspected area of smoke. The calculation of the aspect ratio feature and the rectangularity feature is simple and representative. The aspect ratio of most moving objects is within a fixed range, while the shape of the smoke is irregular with low rectangularity. Smoke can be distinguished from people, leaves, and vehicles through the aspect ratio feature. In addition, other moving objects with regular shapes can be distinguished by the rectangularity feature of the moving area.

The main steps of feature extraction are as follows:

(1) A plurality of motion regions may appear in the obtained moving image. The paper extracts the contours of each motion area in the image and then obtains the minimum enclosing rectangle of the vertical boundary of each motion area contour.
(2) The aspect ratio and the rectangularity of the minimum enclosing rectangle are calculated, and the calculation formula is as follows:

$$Rectangular\ \ degree = S_{area}/S_{rect} \tag{11}$$

$$Aspect\ \ ratio = w_{rect}/h_{rect} \tag{12}$$

where $Sarea$ is the area of the moving area, which is calculated as the ratio of the number of pixels in the moving area to the total pixels of the detected image. S_{rect} is the area of the smallest circumscribed rectangle in the motion area. The w_{rect} and h_{rect} are the width and height of the smallest circumscribed rectangle of the motion region, respectively.

(3) The largest moving region of S_{area} after feature extraction is selected as the suspected smoke region. The input image shape of the deep learning model is usually square, so we first extract the minimum circumscribed circle of the contour of the motion region, and then obtain the minimum circumscribed square of the motion region based on the minimum circumscribed circle. Finally, the minimum circumscribed square is used as the input image of the deep neural network. Owing to the part of the minimum circumscribed circle possibly being beyond the boundary of the original image, changing the size of the suspected smoke detection image will often make the image fuzzy and affect the classification results. Furthermore, the moving object is usually located in the center of the minimum circumscribed square from the current step, and the edge of the image is the static background area, which is consistent with the requirements of smoke training images and helps to improve the detection accuracy. Therefore, we still keep the original image for processing.

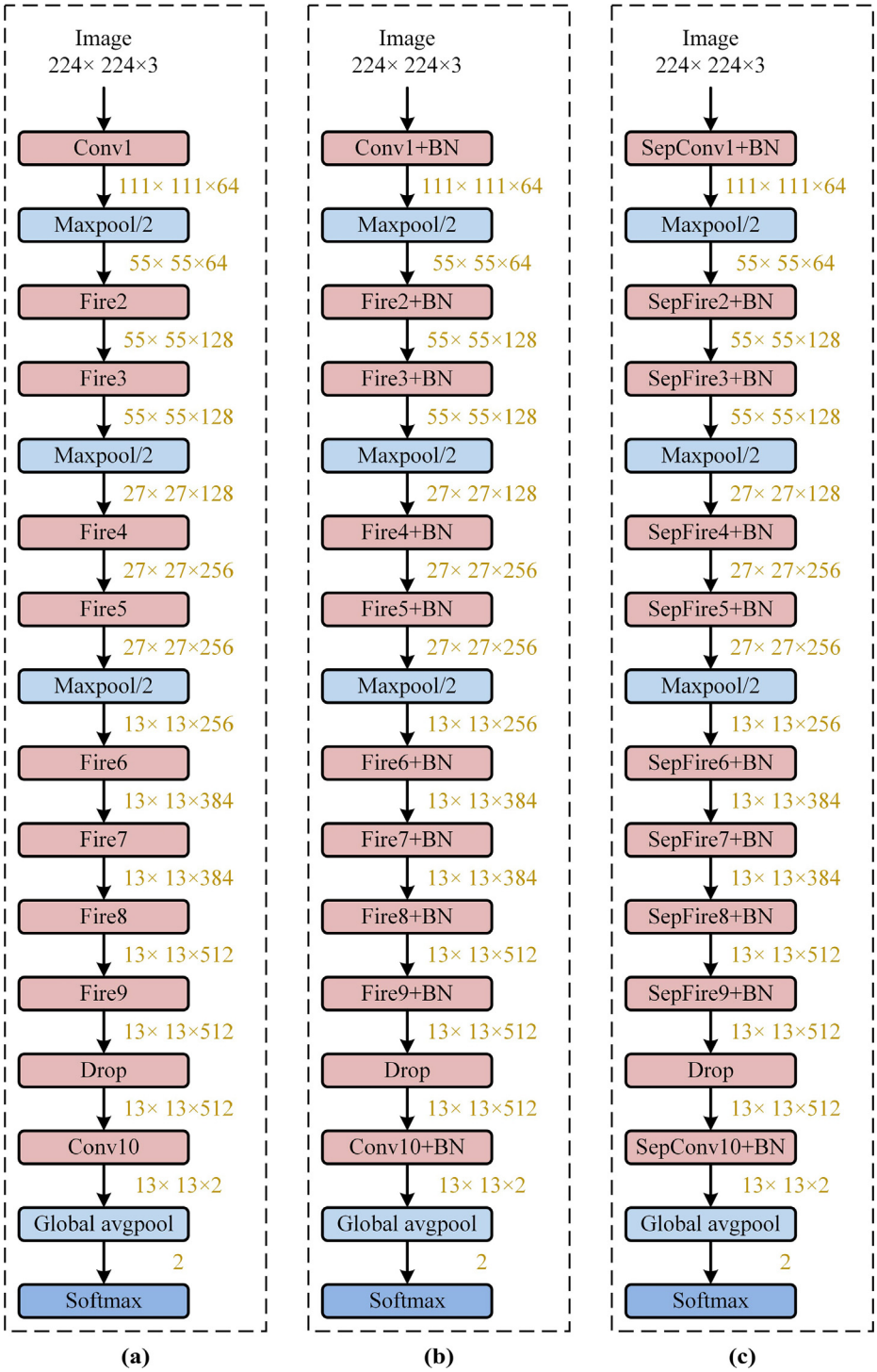

Fig. 6. SqueezeNet's experimental structure, (a) SqueezeNet1, (b) SqueezeNet2, (c) SqueezeNet3.

(4) If the size of the image from the previous step is smaller than the minimum input size of the deep learning network, it is regarded as smokeless in the image. If not, the deep learning network is used to classify the image to detect smoke in the next step.

3.3. Deep learning image classification model establishment

We obtained images of suspected smoke areas based on previous research work. Our smoke classifier is based on the structure of SqueezeNet to identify smoke and non-smoke. In this paper, three different SqueezeNet models are constructed, namely, SqueezeNet1, SqueezeNet2, and SqueezeNet3, and the structures of the three SqueezeNet models are shown in Fig. 6(a–c) in sequence.

Fig. 6 (a) shows the original SqueezeNet network (SqueezeNet1). Fig. 6(b) represents the SqueezeNet network constructed with batch normalization layers (SqueezeNet2), in which batch normalization layers are added between the convolutional layer and the convolutional layer activation function layer. Fig. 6(c) shows that based on the SqueezeNet2 model, a depthwise separation convolution layer is used instead of the convolution layer to construct the SqueezeNet network (SqueezeNet3). Fig. 6(c) shows the SqueezeNet network (SqueezeNet3) based on the SqueezeNet2 model using a depthwise separable

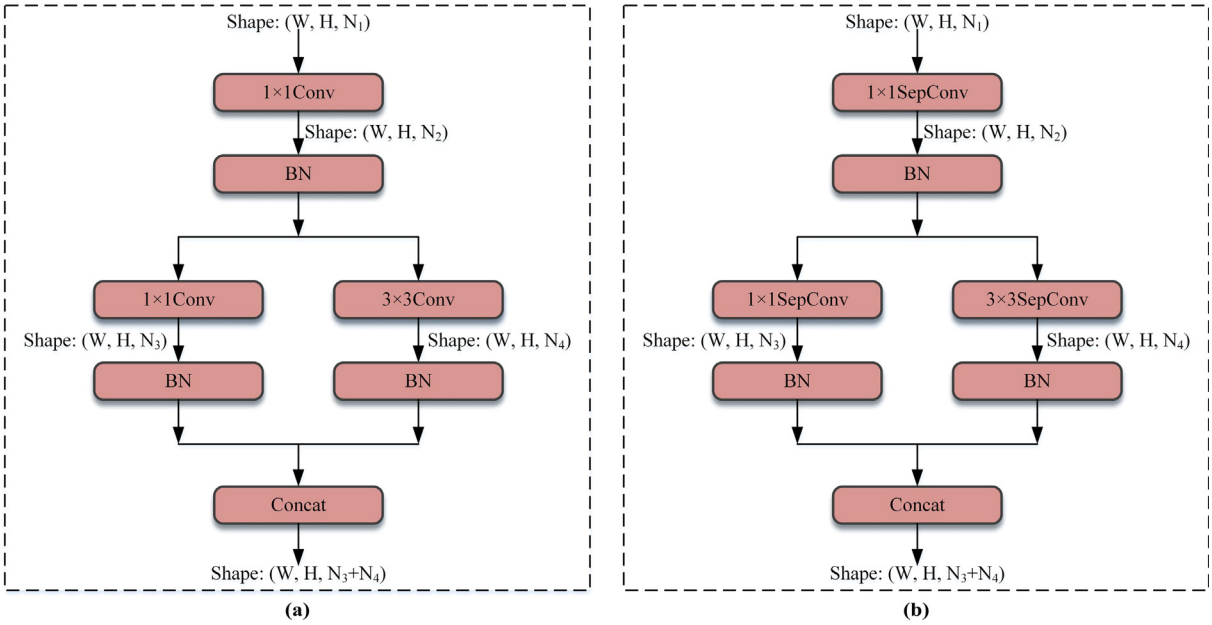

Fig. 7. Fire layer of SqueezeNet2 and SqueezeNet3, (a) SqueezeNet2 fire layer structure, (b) SqueezeNet3 fire layer structure.

convolution instead of a convolutional layer. The fire layer of the SqueezeNet2 and SqueezeNet3 models built is shown in Fig. 7. The activation function of all built models is the ReLU function, and the classifier is the softmax function. Here, "Sep" is the abbreviation of separate. As shown in Figs. 6 and 7, we use SepConv and BN to represent depthwise separable convolution layers and batch normalization layer described in Section 2, respectively. Besides, Fire + BN, Conv + BN, SepConv + BN indicates that batch normalization layer is added behind convolution layer, and SepFire means replacing Conv in the fire layer with SepConv.

3.4. Working principle of smoke detection application

In an actual deployment environment, the flow of the smoke detection program is shown in Fig. 8. Before deploying the smoke detection algorithm, we will transform the deep learning model. It builds by tf.keras (Keras module of TensorFlow). The model completed in the training of keras was transformed into the PB model under TensorFlow.

The smoke detection program includes mainly five modules, which are camera, program terminal, motion detection, image processing and image classification. The main functions of each module are as follows:

(a) Cameras

We mainly obtain real-time video data from various cameras based on network protocols, and the result is a YUV data stream.

(b) Program terminal

Program terminal is mainly used to manage the basic work of program operation, which is:

(1) Convert YUV video stream data into RGB images with the unified resolution of D1 (704 × 576).
(2) Call TensorFlow c + API, and read the TensorFlow model from the local disk.
(3) Store smoke alarm information and alarm image.
(4) Accelerate smoke detection with implementing parallel detection of the algorithm, according to the hardware equipment and the number of cameras.

(c) Motion detection.

We used 25 frames of the camera in real time to create the motion model. Reset and update it in the wee hours of the day. Smoke detection is not carried out when motion model is established. In addition, the program updates the motion model in real time based on the input image of the camera and outputs the motion binary image.

(d) Image process.

Through the image processing module, the image is preliminarily processed, and the non-smoke area is quickly eliminated, so as to obtain the input image of the deep learning model.

(e) Image classification.

The program resets the incoming image resolution to 224 × 224, and enters the image into the deep learning interface for image classification. At the end of the Image classification, the program returns the results to the program terminal, ending the current smoke detection process.

4. Results and analysis

The smoke detection algorithm was analyzed experimentally to verify the feasibility. The equipment used in the experiment is split into a training environment and a test environment. The training environment is only used to train the deep learning model. The training of SqueezeNet and other deep learning models is based on Python 3.5 and tf.keras under Ubuntu Linux 18.04. The hardware platform used is an Intel (R) Core (TM) i7-5820 K CPU at 3.30 GHz, and an NVIDIA GTX TitanX GPU, with 64 GB memory.

The test environment is used to test the accuracy of the feature extraction algorithm and the performance of the deep learning model. We realized all image processing and feature extraction operations based on C + +11 and OpenCV3.4 in Windows 7 and deployed the deep learning model through the TensorFlow C + + API. The hardware platform used is an Intel (R) Core (TM) i5-6300HQ CPU at 2.30 GHz with 12 GB memory, and all development work is based on the Visual Studio 2017 IDE. We can thoroughly verify the performance of our algorithm in the GPU-free environment, and precisely determine the

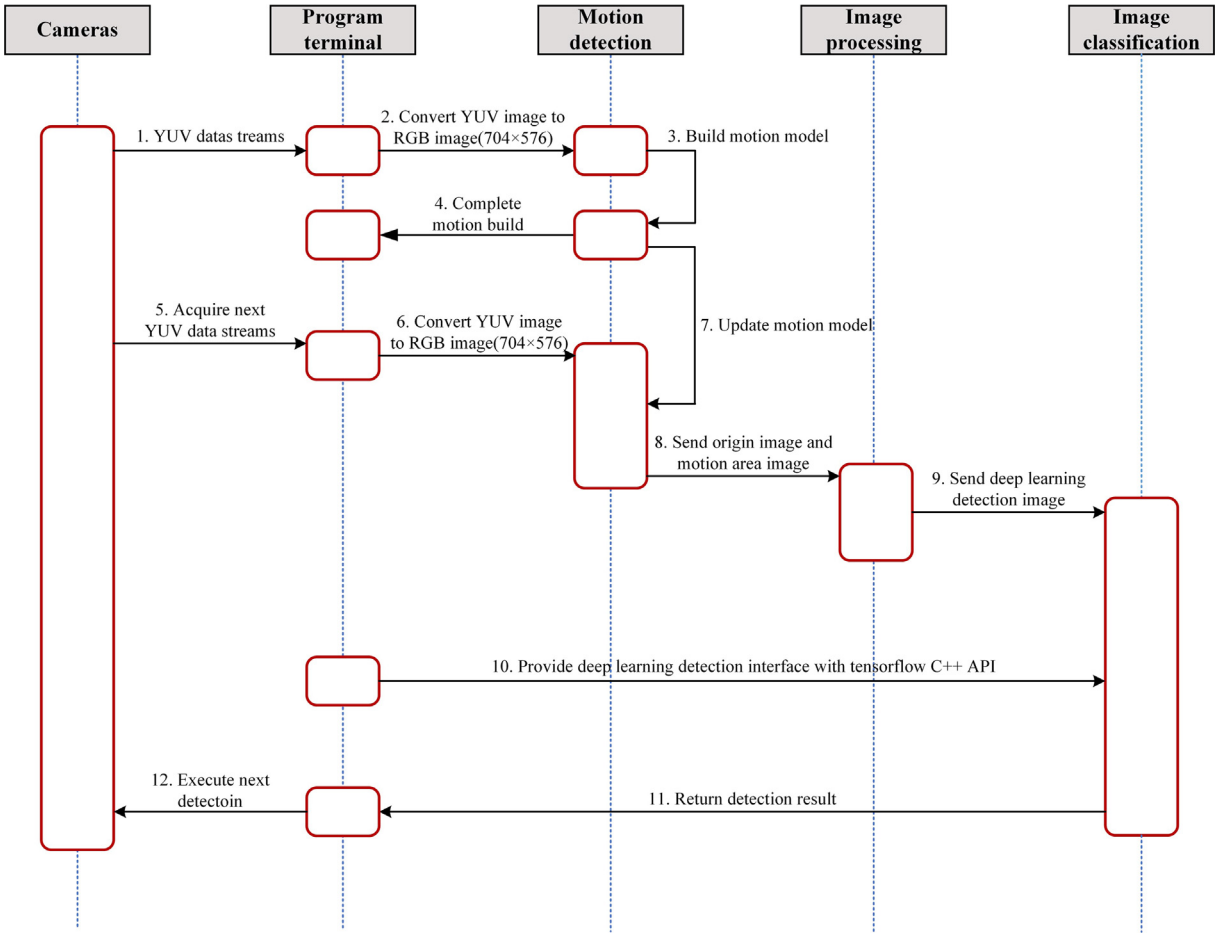

Fig. 8. Sequence diagram of smoke detection.

accuracy, speed, and requirements of the hardware configuration platform.

4.1. Feature extraction

We obtain areas of suspected smoke by manually extracting simple and effective features. Common video image formats in video surveillance are the CIF format (352 × 288), D1 format (704 × 576), 720p format (1280 × 720), and 1080p format (1920 × 1080) (Vasudev et al., 2003). The images used in this paper are reset to D1 resolution (704 × 576), which can guarantee detection accuracy, speed, and applicability.

In this article, videos with image resolutions of 320 × 240, 1280 × 720 and 1920 × 1080 are detected. The first 25 frames of the video are read for modelling, image processing and feature extraction are performed from the 26th frame. There are three different video feature extraction results (a)–(c) in Fig. 8. The results of each step are as follows:

(1) The first step is to obtain the original image, and then the image is scaled to D1 format (704 × 576) for background modelling.
(2) The second step is motion detection, and the result is a motion binary image extracted by the motion detection. The white area is the motion area, and the black area is the background area.
(3) The third step is to perform median filtering on the motion binary image, and the image opening operation and image dilation operation are performed to obtain the motion image after filtering.
(4) The fourth step is to extract the contour of the moving area and obtain the results of each moving area block. The moving area image is obtained, in which the red rectangle is the minimum enclosing rectangle of the moving area block.
(5) The fifth step is to select the region with the largest area as the suspected moving region by rectangularity and the aspect ratio. The results are shown in the Fig. 9 of the suspected smoke area image. To match the input square image required by deep learning, we extract the minimum enclosing rectangle (yellow area), and the minimum circumscribed circle (purple area) of the maximum smoke area is obtained. Then, on the basis of these, we obtain the minimum enclosing square (pink area) of the moving area.
(6) The sixth step is to extract the pink maximum square region from the D1 original image as the deep learning detection image.

From the image processing results of three different resolutions, the detection area of smoke can be significantly reduced by feature extraction of the image motion area, and the non-smoke blocks can be quickly excluded. Further, the motion area in the smoke detection image obtained through the feature extraction stage is in the middle of the image. In this manner, the background information and the shape information of moving objects can be retained, and the false positive rate can be reduced to improve the detection accuracy of the model.

4.2. SqueezeNet model establishment

4.2.1. Data set

Some of the image data used in the deep learning model for the classification of smoke are shown in Fig. 10, where the smoke image is from internet copyright-free websites and public datasets. The selected image contains smoke images of different sizes and colors in different

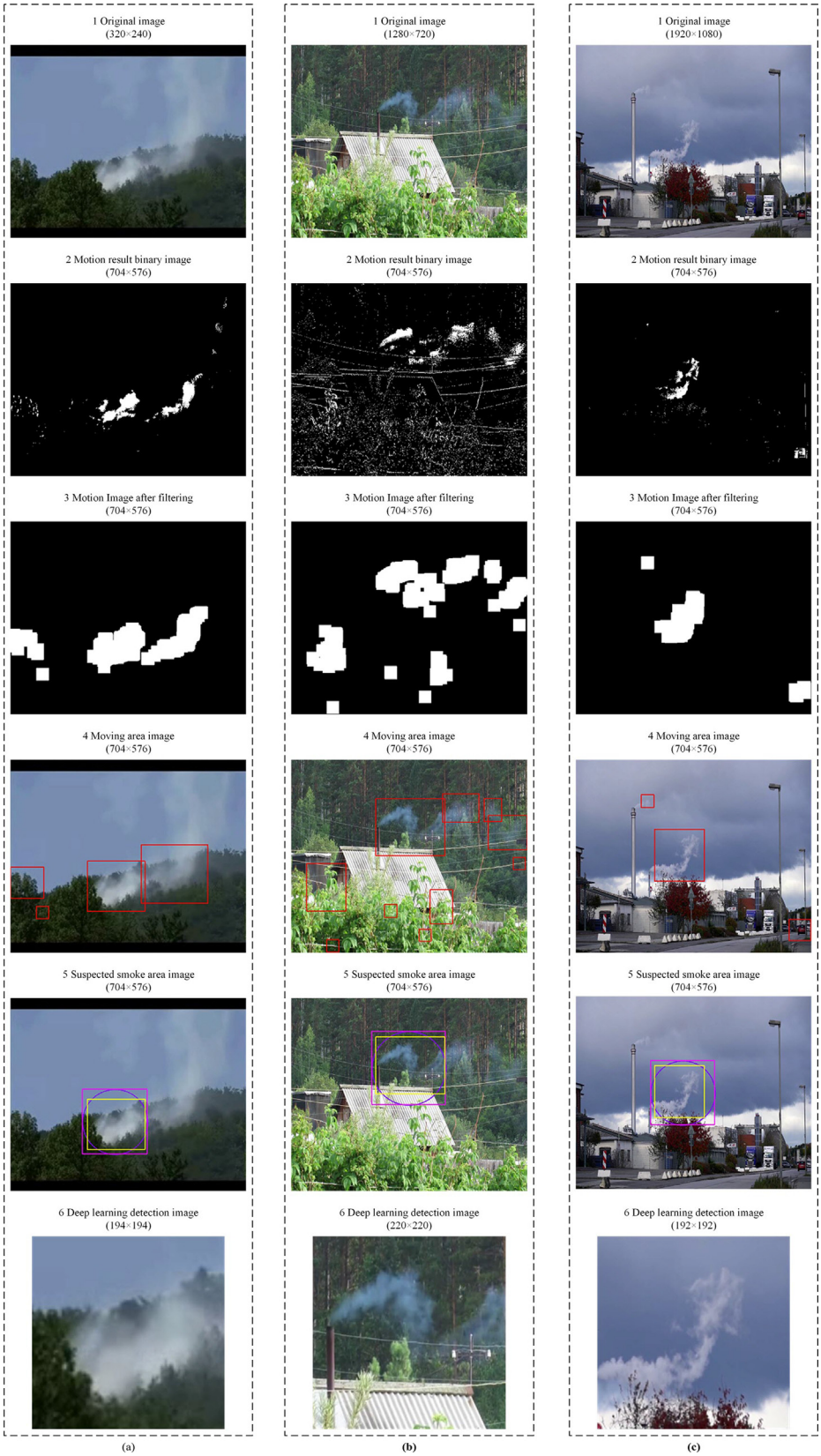

Fig. 9. Feature extraction results, (a) extraction results of D1 resolution video, (b) extraction results of 720p resolution video, (c) extraction results of 1080p resolution video.

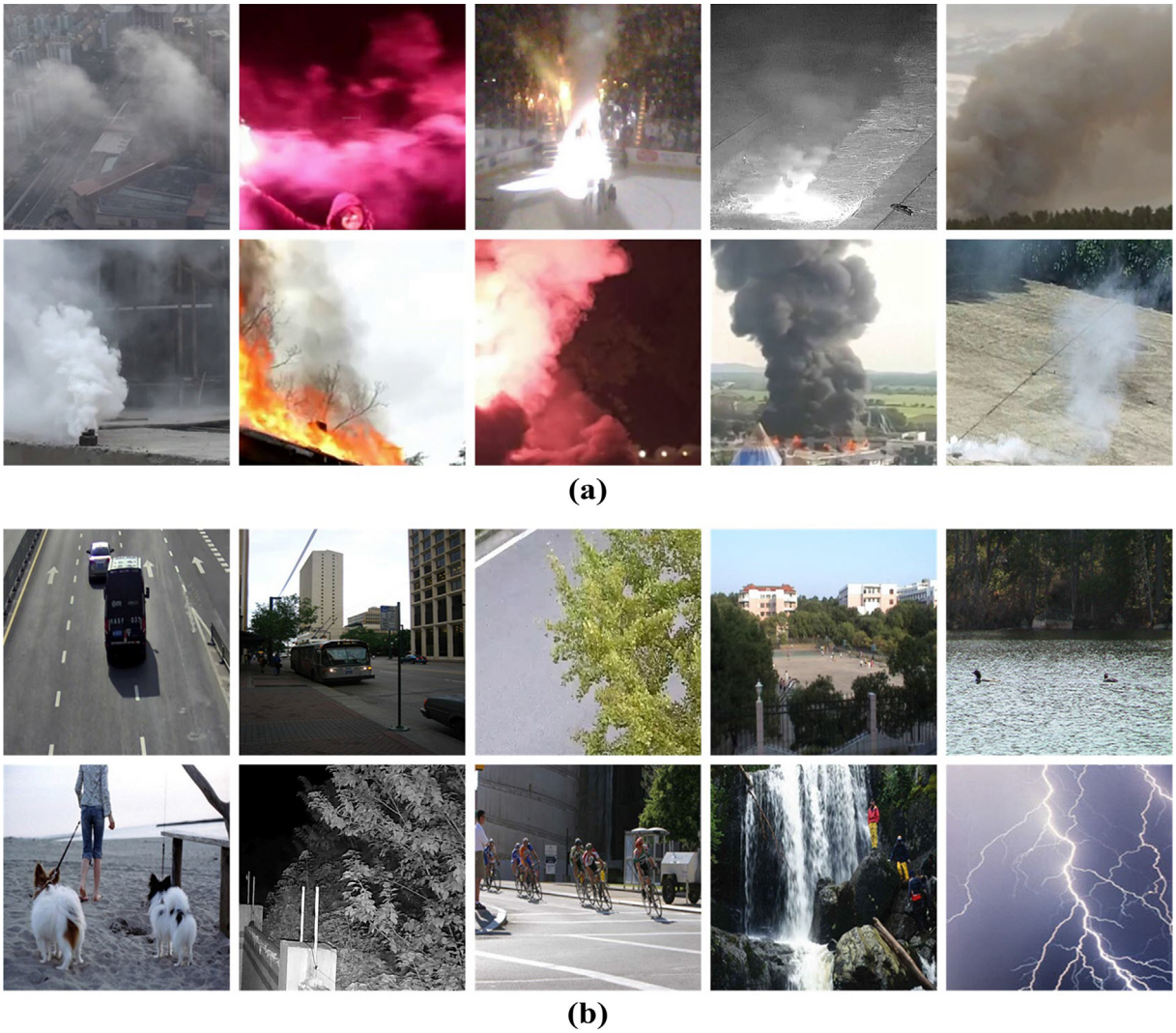

Fig. 10. Images from the training set of deep learning, (a) smoke images, (b) non-smoke images.

environments. Among all the smoke images modelled, the smoke area accounts for 1/5–4/5 of the total image area. In this manner, the image edge part can be guaranteed to contain the background region, which is consistent with the characteristics of the suspected smoke region image extracted in the previous step. Selected non-smoke images were all images containing moving objects, which were common moving objects that occur in daily life, such as people, animals, trees, and vehicles. The area of the moving object in the non-smoke image is 1/5–4/5 that of the total image area.

According to the above criteria, a total of 25,000 smoke images and 25,000 non-smoke images were acquired, and a smoke image data set was created based on these images. The smoke image data set is divided into a training set, a verification set, and a test set according to a ratio of 6:2:2. The training set is used to train the model, the validation set is used to adjust the parameters and select the model, and the test set is used to evaluate the performance of the model as a whole.

4.2.2. Selection of suitable models

To choose the best SqueezeNet structure while avoiding its special and accidental nature and to reduce the interference of human parameter adjustment, the network is trained without manual parameter adjustment by randomly setting the dropout value (the proportion of the discarded layers in training) and initial learning rate. The dropout value is between 0.2 and 0.6. Because there is no batch normalization layer in SqueezeNet1, a large initial learning rate will lead to no convergence, so the initial learning rate range of SqueezeNet1 is 10^{-6}–10^{-4}. However, SqueezeNet2 and SqueezeNet3 have batch normalization layers, so we can choose a larger learning rate to start training; therefore, SqueezeNet2 and SqueezeNet3 have an initial learning rate in the range of 10^{-4}–10^{-2}. The initial learning rate and dropout values are randomly obtained under batch sizes of 32, 64, and 128, and the Adam optimization algorithm (Kingma and Adam, 2014) is used to train the network. Each network has 30 different initial combination parameters for training under different training batches. The maximum number of training sessions for each combined parameter network is 10,000. When the verification set loss is unchanged for ten consecutive training sessions, the training ends early. The accuracy and detailed information on the verification set under different conditions are presented in Fig. 11 and Table 1, respectively:

According to Fig. 11 and Table 1, the maximum and average accuracy of SqueezeNet2 and SqueezeNet3 under processing conditions with different batch sizes are better than that of SqueezeNet1. These findings indicate that after the addition of batch normalization, the training model with a larger learning rate can speed up the convergence of the model and the gradient disappearance problem of the deep network can be avoided. The training results of SqueezeNet2 and SqueezeNet3 under different initial conditions are not very different, so the batch normalization layer can reduce the dependence on parameter initialization methods. Furthermore, when depthwise separable convolution layers are added in SqueezeNet3, the average and maximum

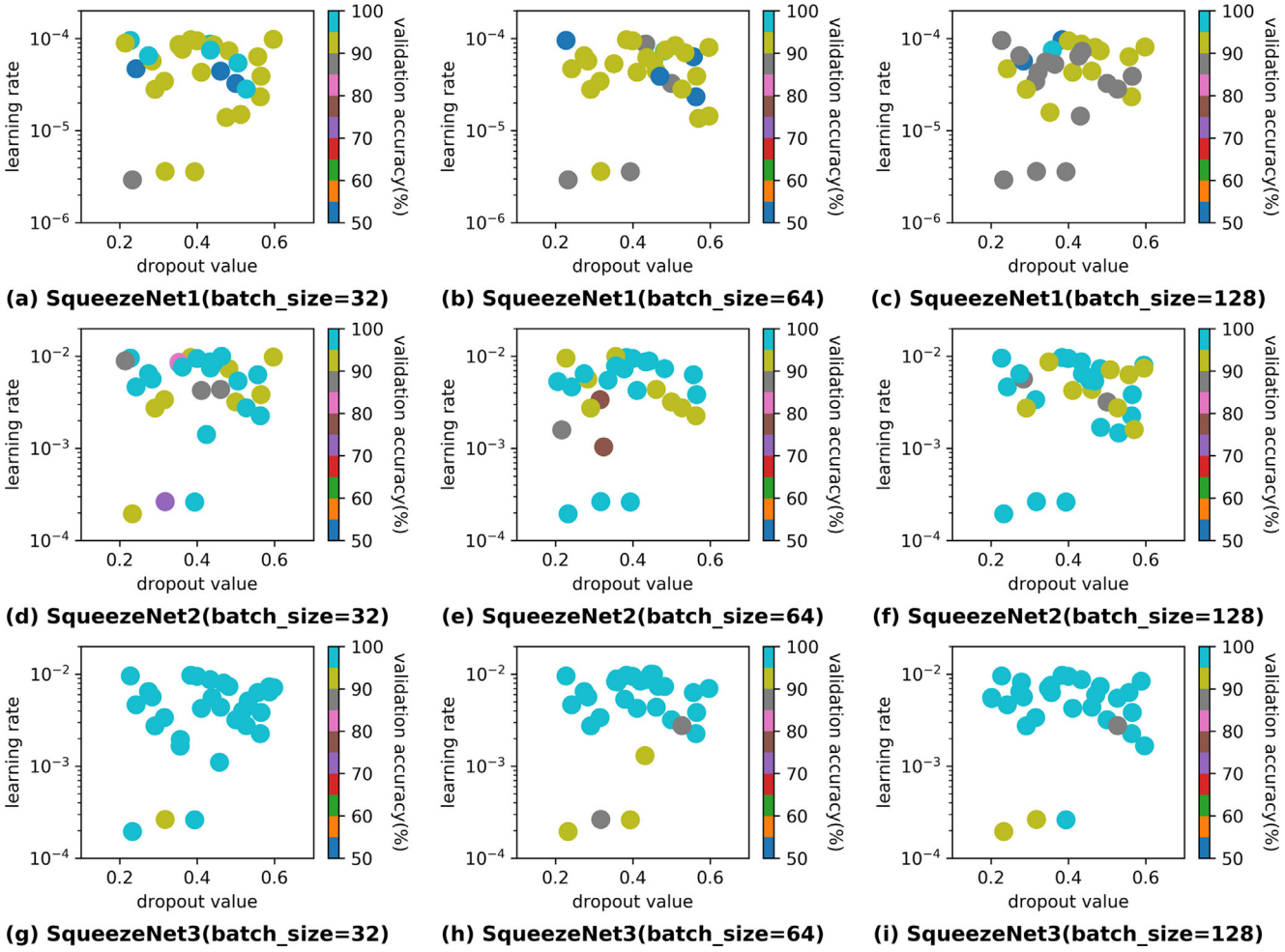

Fig. 11. The validation accuracy of three SqueezeNets under different conditions.

Table 1
Accuracy of different SqueezeNets under validation set.

Models	Batchsize	Maximum accuracy (%)	Average accuracy (%)	Minimum accuracy (%)
Squeezenet1	32	95.73	89.07	50.00
	64	95.41	86.53	50.00
	128	95.02	87.60	50.00
Squeezenet2	32	98.14	93.45	70.87
	64	98.02	94.11	76.15
	128	98.46	95.16	85.47
Squeezenet3	32	98.85	97.51	94.75
	64	98.85	96.32	88.45
	128	98.56	96.85	88.72

accuracy of the model is higher than that of SqueezeNet2, which shows that the performance of depthwise separable convolution layers is better than that of the ordinary convolution layer. Therefore, SqueezeNet3 was chosen as the smoke classification and detection model in this study.

4.2.3. Model comparison

To verify the superiority of the SqueezeNet3 model with the batch normalization layer and depthwise separable convolution layer, some other small models and classical models are compared. These small models are Xception (Chollet, 2017), MobileNet (Howard et al., 2017), MobileNetV2 (Sandler and Howard, 2018), ShuffleNet (Zhang et al., 2018), and ShuffleNetV2 (Ma et al., 2018), while the classic models are AlexNet (Krizhevsky et al., 2012) and VGG16 (Simonyan and Zisserman, 2014).

The SqueezeNet1, SqueezeNet2, and SqueezeNet3 models with the highest verification set under different conditions in Section 4.2.2 were selected as the detection model. The deep learning model for comparison was trained with the training set, and the best model is obtained based on the verification set adjustment parameters. Finally, the classification performance, calculation accuracy, recall rate, precision rate, and F_1 score of different methods based on the unbalanced data set were quantitatively studied by comparing each model with the test set. The calculation formulas are as follows:

$$\begin{aligned} Accuray &= (TP + TN)/(TP + TN + FP + FN) \\ Precision &= TP/(TP + FP) \\ Recall &= TP/(TP + FN) \\ F_1 &= 2 \times Precision \times Recall/(Precision + Recall) \end{aligned} \tag{13}$$

where TP is the number of actual smoke samples predicted to be smoke; FP is the number of actual non-smoke samples predicted as smoke; FN is the number of actual smoke samples predicted as non-smoke, and TN is the number of actual non-smoke samples predicted as non-smoke.

The higher the accuracy is, the fewer the false alarms, and the higher the recall rate is, the fewer the missing alarms. In addition to smoke detection accuracy, the detection speed is also a critical indicator. We calculated the average test time (forward propagation time), the number of parameters and the model size of various models 1000 times for a single image under the test set. Our goal is to select the model with the highest F_1 score and accuracy, and the fastest detection speed based on the comprehensive indicators. Table 2 lists the accuracy and F_1 score of the various models in the test set, as well as the recall

Table 2
Results of different models under the test set.

Models	Precision (%)	Recall (%)	Accuracy (%)	F1
SqueezeNet1	97.920	93.775	95.803	95.710
SqueezeNet2	96.480	96.067	96.273	96.265
SqueezeNet3	97.950	96.313	97.124	97.100
Xception	94.140	96.494	95.303	95.360
MobileNet	94.430	99.536	96.916	96.995
MobileNetV2	99.000	96.585	97.778	97.750
ShuffleNet	94.430	98.447	96.396	96.470
ShuffleNetV2	98.990	97.326	98.151	98.135
AlexNet	94.270	96.066	95.160	95.205
VGG16	96.500	98.550	97.514	97.540

Table 3
Comparison of prediction speed and total parameters of different models.

Models	Average prediction time (ms)	Total parameters	Model sizes (MB)
SqueezeNet1	35.09	723,522	2.90
SqueezeNet2	92.12	735,306	3.05
SqueezeNet3	78.49	248,101	1.20
Xception	928.40	20,871,721	79.70
MobileNet	389.05	3,229,889	12.40
MobileNetV2	260.46	2,587,201	10.00
ShuffleNet	235.32	938,713	4.65
ShuffleNetV2	180.64	1,266,617	5.06
AlexNet	89.19	59,369,729	226.00
VGG16	733.56	134,264,641	512.00

rate and accuracy. Table 3 lists the average prediction time, total parameters and models sizes of 1000 test images.

Tables 2 and 3 show that SqueezeNet2 with the additional batch normalization layers is more accurate than SqueezeNet1, but SqueezeNet2 is slower than SqueezeNet1 at runtime. The addition of the batch normalization layer yields increased precision, but greatly reduced speed. SqueezeNet3 not only has the additional batch normalization layer but also the additional deep separable convolution layer. Compared with SqueezeNet2, SqueezeNet3 exhibits not only significantly improved accuracy but also improved recall rate, precision rate, and F_1 score. Furthermore, the detection speed is also accelerated. Moreover, the model parameters of SqueezeNet3 are considerably smaller than those of SqueezeNet1 and SqueezeNet2, and fewer model parameters make SqueezeNet3 more practical and useful for deployment in a realistic environment.

Compared with other models, SqueezeNet1 exhibits an accuracy close to that of AlexNet, and the accuracy of SqueezeNet2 is close to that of ShuffleNet and MobileNet. For SqueezeNet3, the accuracy rate (97.124%) is second only to ShuffleNetV2 (98.151%), VGG16 (97.514%) and MobileNetV2 (97.778%), but the prediction time and model size of SqueezeNet3 are much smaller than those of ShuffleNetV2, VGG16 and MobileNetV2. At the same time, the SqueezeNet3 model achieves the smallest volume in the model used. Thus, SqueezeNet3 is more suitable for smoke detection than other models in terms of accuracy, prediction time, and model size.

Finally, we use the method of (Selvaraju et al., 2017) to visualize the results of the SqueezeNet3 model (CAM, class activation map). The class-activated thermogram is a two-dimensional fractional grid associated with a particular output category, which indicates how important each location is to that category. The thermal performance map can serve to visualize the classification performance of SqueezeNet3. The thermodynamic value is standardized in the 0–1 range (Fig. 12). For images implemented in the smoke classification SqueezeNet3, CAM visualization can generate “smoke” and “non-smoke” thermographs, indicating the similarity of different parts of the image to “smoke” and “non-smoke” respectively. The class activation heat map is superimposed on the original image (Fig. 12). The smoke detection effect of the SqueezeNet3 model is quite consistent with the actual smoke area, and the non-smoke area is mainly determined in the non-smoke area (house). The detection results show that the smoke area intensity is much higher than that of the non-smoke area and that the image is classified as smoke, which indicates that SqueezeNet3 can extract deep-level features of the image and has good classification ability.

4.3. Video testing

4.3.1. Test video dataset

In this paper, we test the performance of the algorithm with the video data by using the video data set of scenes that were not employed in the training set. The fire smoke video currently available for research is low resolution, and the resolution of most videos is smaller than D1. Therefore, we selected the smoke video dataset from the State Key Laboratory of Fire Science at the University of Science and Technology of China (Lin et al., 2017). The video data set was captured by shooting a set of 1920 × 1080 smoke and non-smoke videos in an indoor room using a high-definition video network camera. The data set is made up of training sets and test sets, which are used for video testing. The dataset test video is shown in Fig. 13.

A detailed description of each video in this article is provided in Table 4. The video data set was created with a high-definition video network camera to capture 15 sets of videos with a resolution of 1920 × 1080, which is consistent with the live real-time camera monitoring image and has a typical representative meaning. The first ten groups are smoke videos, and the last five groups are non-smoke videos.

4.3.2. Video detection accuracy test

Each video in the test set is detected separately, where the first 25 frames of video are used for background model building, and the remaining frames are tested after updating the background model. The specific detection accuracy of the test video is shown in Fig. 14. The average accuracy of the first ten groups of the smoke video is 89.41%. The average accuracy of the last five groups of the non-smoke video is 97.74%, and the overall accuracy is 91.81%.

Although the results show that our model has high detection accuracy, in the actual smoke test, it is still lower than that reported in a previous study (Lin et al., 2017), which uses an SVM to classify and identify smoke by extracting the texture features of smoke. However, since the training data set adopted in the former study (Lin et al., 2017) is consistent with the test data set environment, it is unfair to compare the results because the test difficulty is significantly reduced. Therefore, this method can only be used in specific situations and does not apply to complex environments that are exposed to elements in nature.

The motion detection algorithm used in this paper needs to establish a smoke-free background motion model before detection. Moreover, since some test smoke videos have smoke from the first frame and there is no change in the direction of smoke movement in an indoor environment, the precision of motion detection is reduced. Furthermore, given that our model mainly solves the problem of smoke monitoring in a complex natural environment and indoor smoke monitoring is not the focus of this paper, indoor smoke images are not added to the data set, resulting in low accuracy test results of indoor smoke detection.

The first ten videos are divided into smoke motion detection videos. However, little smoke was generated in the first 750 frames (1/4 of the detected frames) of video 8. However, the proposed the algorithm indicated that there was smoke at the start of the video, which was wrong and led to a decrease in the accuracy of the overall test results. Video 11 is less accurate than other smokeless videos because of the flame in the video. The motion characteristics of fire and smoke contours are similar, and our detection model does not restrict the color of smoke. Yellow smoke is also included in the training set, which leads to the misjudgment of flame as smoke. However, when a fire occurs, our model can trigger an alarm in time through the dynamic changes of the

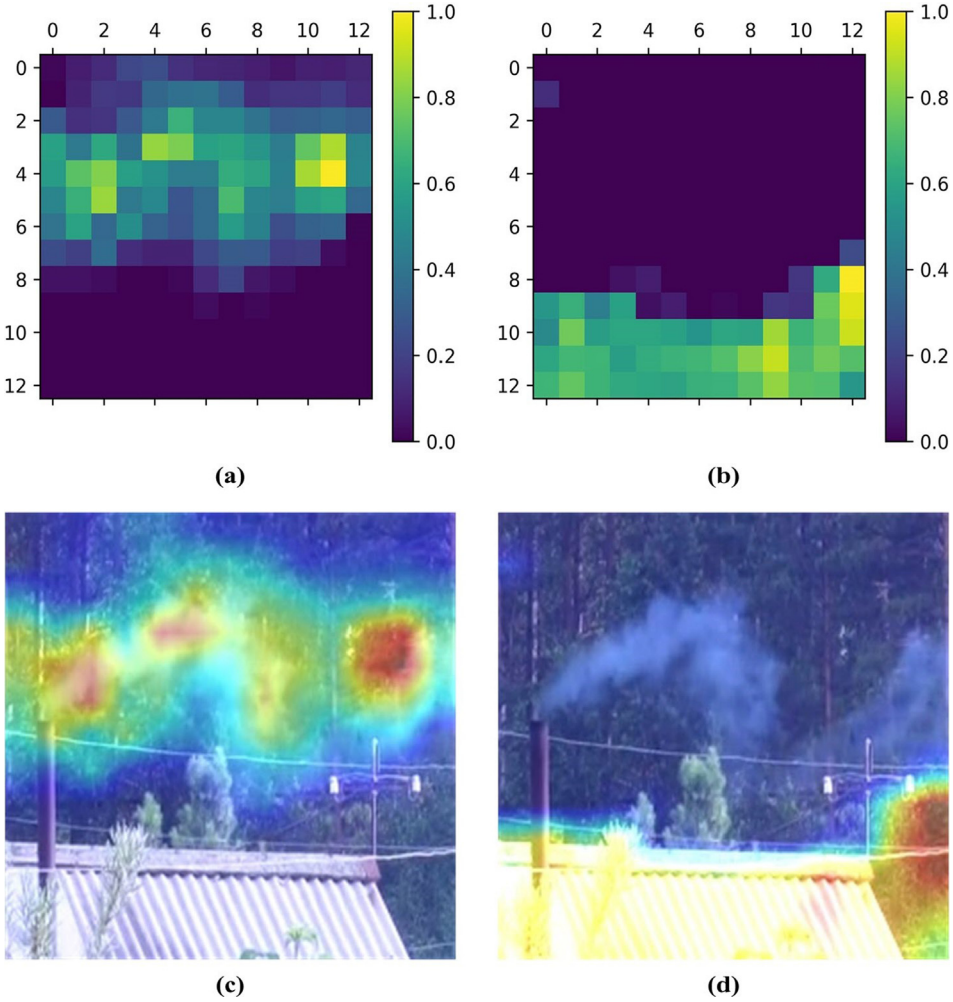

Fig. 12. Thermogram of smoke detection deep learning model, (a) smoke heatmap image, (b) detected image with superimposed smoke heatmap, (c) non-smoke heat map image, (d) detected image with superimposed non-smoke heatmap.

flame and can doubly monitor the site environment, which is helpful in practice.

In summary, in the case of a complex environment and a vast difference between the training set and the test set, the proposed model still achieves high accuracy (91.81%). In particular, the detection of the non-smoke video has achieved good results (97.74%). Therefore, the proposed method not only guarantees the detection of smoke sensitivity but also reduces the false detection of non-smoke videos, which proves the effectiveness of the algorithm. Furthermore, the method can be effectively applied to various scenes and has high practical value.

The above is the smoke detection efficiency based on a single image. However, in actual tests, detection is usually based on the results of detection in multiple images to comprehensively judge whether there is smoke in the scene and ensure accuracy. Multiple detections can avoid the low accuracy of video 8 (Table 5). The detection method in Table 5 indicates that the occurrence of the frame A in the detection of the N frame video determines that the current detection result is smoke, which is expressed in A/N. When A/N is less than 0.5, the smoke detection accuracy and the smoke detection sensitivity can be remarkably improved. When A/N is greater than 0.5, the false alarm rate can be reduced, but the smoke detection sensitivity will also be significantly reduced (Table 5).

As indicated in Table 5, models 1/2 and 1/3 have the highest detection accuracy, and the specific accuracy for the 15 videos in the test video set is reported in Fig. 15. The results show that the detection methods of modes 1/2 and 1/3 can reduce the false negative rate and the false positive rate, and further improve the accuracy of smoke recognition. Compared to a single smoke detection image, although the number of detections is increased in detections of multiple images, the accuracy of the detection can be ensured because the algorithm has low hardware requirements and a fast detection speed.

4.3.3. Video test performance test

In case of fire, quick and accurate identification of smoke can speed up fighting it and reduce losses. Therefore, the detection speed is an essential indicator of smoke detection. However, there are often multiple cameras to monitor the occurrence of fire and smoke in real environments. Our detection algorithm requires the ability to process multiple videos simultaneously. Generally, the CPU of a computer has multiple computing cores, each of which can perform a single task, or multiple cores that can perform a task together. Accordingly, we tested the performance of the algorithm in three different cases: one video processed by a single CPU kernel, one video processed by multiple CPU cores, and numerous videos processed by multiple CPU cores (Wang et al., 2016).

Since the CPU of the test environment consists of four cores, the test environment can only process four videos at a time. Most of the frames of test videos 2, 3, 6, and 7 enter the various stages of the algorithm detection operation. Hence, we select these four representative videos for the algorithm performance test and take the average value of each frame to evaluate the performance of the algorithm. The leading evaluation indicators are the speed of each detection module and the total

Fig. 13. Image from the test video set.

Table 4
Specific information on test set videos.

Videos	Total frame numbers	Content
1	1501	White smoke plume
2	1332	
3	951	
4	1501	
5	1681	
6	961	Diffuse white smoke
7	2251	
8	3001	Thin black smoke
9	1441	Black smoke
10	721	
11	1896	Walking pedestrians
12	1501	
13	471	
14	961	Swing curtains
15	1441	Lighting changes

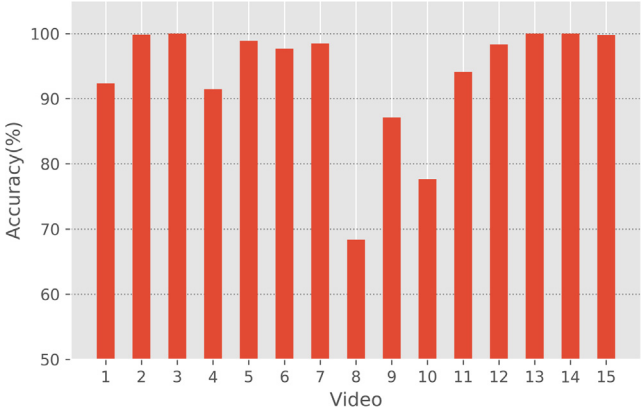

Fig. 14. Accuracy with the test set.

Table 5
Average accuracy of different detection modes.

Detection modes	Smoke videos (%)	Non-smoke videos (%)	All videos (%)
1/2	90.298	96.126	91.985
2/2	70.179	98.828	78.468
1/3	95.446	95.408	95.435
2/3	80.453	97.411	85.359
3/3	65.122	99.170	74.972

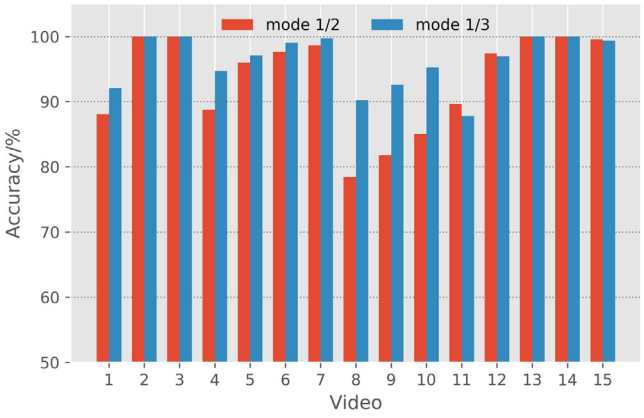

Fig. 15. Test video set accuracy in detection mode of 1/2 and 1/3.

Table 6
Performance test of detection algorithm.

Detection modes	Single video/ Single core	Single video/ four cores	Four videos/four cores
Motion modeling/ms	29.15	11.99	39.30
Image process/ms	11.90	9.60	14.10
Deep learning/ms	23.30	10.50	32.50
Others/ms	8.15	4.36	15.35
Total time/ms	72.50	36.45	101.25
Memory/MB	121.00	122.40	422.70

memory occupancy of the algorithm. The performance results of the algorithm in three different cases are reported in Table 6:

In terms of a single core of the CPU to detect only one video at a time, the single image detection of the entire process requires 72.5 ms. After each frame of the image is read in, it takes a small amount of time to update the background model. Therefore, we use C++ to run the image processing algorithm and the deep learning detection model in the actual environment, which can greatly save the image processing time and the deep learning model detection time. Although the feature extraction will increase the detection time, it is essential, and a negligible impact on the whole algorithm time and memory. Additionally, the initial loading of the deep learning model and the initialization of the background model also take a small amount of time.

When multiple CPUs process a video together, the detection time can be reduced by half compared to a single core, and the memory occupancy does not increase much. When the four cores of the CPU process four videos, this method requires only 28.75 ms more than a single core to process a video, and the memory increases by merely 301.7 MB.

The CPU core of the calculation determines the number of simultaneous processing tasks on the computer CPU. Because the algorithm in this paper detects faster and has less memory, we can perform video detection tasks that exceed the number of CPU cores at one time. The CPU will only execute the same amount of video tasks as the number of CPU cores at the same time. Other video detection tasks will be executed successively in a short time under the scheduling mechanism, which can make the CPU run effectively and avoid the

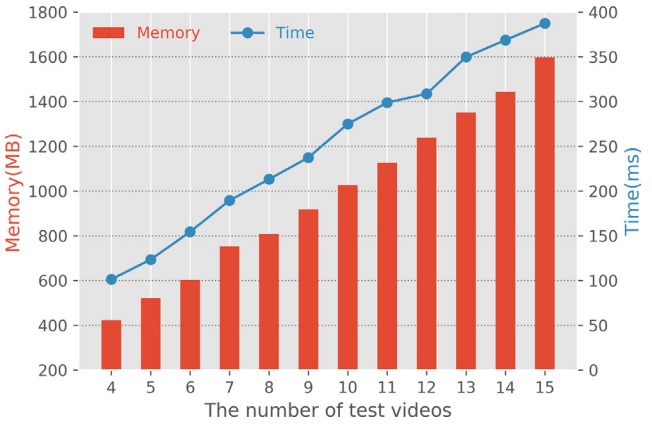

Fig. 16. Performance result of multi-video detection.

repeated loading of the deep learning model and other large modules, thus improving the detection efficiency.

The test time and the memory usage consumed by executing the detection of four videos to 15 videos at the same time are shown in Fig. 16. As the number of video detection tasks increased, the detection time and memory usage of the algorithm increased, but the detection time and memory occupation rate that increased each time are generally lower than those when utilizing a single CPU core to process a single video. As show in Fig. 16, when 15 video tasks were processed simultaneously by four CPU core, the detection time was 387.45 ms, and the working memory was 1597.44 MB. However, 15 video tasks were processed one video at a time by a single CPU core, which used a total of $72.5 \times 15 = 1087.5$ ms. Multiple cores treated one video at a time to handle 15 video tasks in turn, requiring a total of $36.45 \times 15 = 546.75$ ms. Therefore, this method can significantly reduce the detection time and substantially improve the detection speed of smoke. Moreover, it only needs approximately 1.56 GB of memory, and an ordinary computer without a GPU can support simultaneous processing of the algorithm, which also proves the small size and universal applicability of our method.

5. Conclusion

This paper proposes a new video forest fire detection method based on image feature extraction and a small deep neural network model. First, detected images from surveillance cameras are processed frame by frame through the background model with hand-designed features. Then, we add a SqueezeNet network optimized by deep separation convolution layers and batch normalization layers to realize the fast classification of images and enhance the robustness of the detection algorithm. Compared with other smoke detection approaches, our approach has mainly three improvements:

(1) We use an optimized SqueezeNet network as the smoke detection model to achieve fast and accurate smoke detection. The detection model achieved 97.124% accuracy on the test set, and the detection speed and model size in the environment of a CPU and Python were 78.49 ms and 1.20 MB, respectively, a performance that is far superior to other CNN models.
(2) We combine simple and effective hand-designed smoke features and a deep learning model to reduce the computational expense and false alarm rate. The smoke detection rate reached 89.41%, and the non-smoke detection rate reached 97.74% for a video data set with scenes that did not appear in the training set.
(3) The deployed detection algorithm is written in the C++ programming language, and its running speed and accuracy can meet the needs of real-time smoke detection. The entire detection time of a single frame was only 36.45 ms with a CPU and multithreading,

and our algorithm could detect 15 images in parallel at the cost of 387.45 ms and 1.56 GB memory.

The method can be applied to real-time monitoring of forest smoke, effectively preventing and detecting forest fires in a timely manner. Our future work will concentrate on improving the motion algorithms and the selection of smoke features. In addition, we will explore a more accurate and smaller network architecture to improve the smoke recognition performance.

Declaration of Competing Interest

The authors declared that there is no conflict of interest.

Acknowledgments

This work was supported by the Doctorate Fellowship Foundation of Nanjing Forestry University (163010550), and the Priority Academic Program Development of Jiangsu High Education Institutions (PAPD).

References

Calderara, S., Piccinini, P., Cucchiara, R., 2011. Vision based smoke detection system using image energy and color information. Mach. Vision Appl. 22, 705–719. https://doi.org/10.1007/s00138-010-0272-1.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE conference on comput. Vision Pattern Recognition, USA, Hawaii, pp. 1251–1258.

Frizzi, S., Kaabi, R., Bouchouicha, M., Ginoux, J.M., Moreau, E., 2016. Convolutional neural network for video fire and smoke detection. IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society. IEEE, Florence, Italy.

Gonzalez, R.C., Woods, R.E., 2007. Digital Image Processing, third ed. Prentice-Hall, Inc., pp. 1.

Howard, A.G., Zhu, M., Chen, B., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: efficient convolutional neural networks for mobile vision applications. Preprint ArXi 1704.04861. in press.

Iandola, F.N., Han, S., Moskewicz, M.W., Khalid, A., William, J.D., Kurt, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size. ArXiv 1602.07360. in press.

Ioffe, S., Szegedy, C., 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. ArXiv Preprint ArXiv. 1502.03167.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. Neur. Inf. Process. Syst. 141, 1097–1105.

Kingma, D.P., Adam, Ba.J., 2014. A method for stochastic optimization. ArXiv Preprint ArXiv:1412.6980.

Lin, G., Zhang, Y., Zhang, Q., Zhang, J., Jia, Y., Xu, G., Wang, J., 2017. Smoke detection in video sequences based on dynamic texture using volume local binary patterns. KSII Trans. Internet Inf. Syst. 11, 5522–5536. https://doi.org/10.3837/tiis.2017.11.019.

Luo, S., Jiang, Y.Z., 2013. State-of-art of video based smoke detection algorithms. J. Image. Graphics 10, 1225–1236.

Ma, N., Zhang, X., Zheng, H., Sun, J., 2018. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. Eur. Conf. Comput. Vision 122–138.

Manchanda, S., Sharma, S., 2016. Analysis of computer vision based techniques for motion detection. Cloud System Big Data Engineering. IEEE, Noida, India.

Sandler, M.B., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L., 2018. MobileNetV2: inverted residuals and linear bottlenecks. Comput. Vision Patt. Recog. 4510–4520.

Selvaraju, R.R., Cogswell, M., Das, A., Abhishek, R.V., Parikh, D., Dhruv, B., 2017. Grad-CAM: visual explanations from deep networks via gradient-based localization. Int. Conf. Comput. Vision 618–626.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. ArXiv Preprint ArXiv. 1409.1556.

Tang, Y., Ge, L., 2002. Development of scattering-style infrared smog photoelectric detector. J. Sichuan Univ. Eng. Sci. Ed. 4, 117–120.

Tian, H., Li, W., Ogunbona, P., Nguyen, D., Zhan, C., 2011. Smoke detection in videos using non-redundant local binary pattern-based features. IEEE 13th Int. Workshop Multi. Sig. Process. https://doi.org/10.1109/MMSP.2011.6093844.

Toreyin, B.U., Dedeoglu, Y., Cetin, A.E., 2006. Contour based smoke detection in video using wavelets. 14th European Signal Processing Conference, Florence, Italy, pp. 15023869.

Tung, T.X., Kim, J., 2011. An effective four-stage smoke-detection algorithm using video images for early fire-alarm systems. Fire Safety J. 46, 276–282.

Vasudev, B., Hsing, T.R., Tescher, A.G., Ebrahimi, T., 2003. Image and video communications and processing 2003. Int. Soc. Opt. Photon. 5022, 700–707.

Wang, L., Hsiao, Y., Xie, X., Lee, S., 2016. An outdoor intelligent healthcare monitoring device for the elderly. IEEE Trans. Consumer Electron. 62, 128–135. https://doi.org/10.1109/TCE.2016.7514671.

Yin, M., Lang, C., Li, Z., Feng, S., Wang, T., 2018. Recurrent convolutional network for video-based smoke detection. Multi. Tools Appl. 8, 1–20.

Yin, Z., Wan, B., Yuan, F., Xia, X., Shi, J., 2017. A deep normalization and convolutional neural network for image smoke detection. IEEE Access 5, 18429–18438. https://doi.org/10.1109/ACCESS.2017.2747399.

Zhang, Q., Lin, G., Zhang, Y., Xu, G., Wang, J., 2018a. Wildland forest fire smoke detection based on faster R-CNN using synthetic smoke images. Procedia Eng. 211, 441–446. https://doi.org/10.1016/j.proeng.2017.12.034.

Zhang, X., Zhou, X., Lin, M., Sun, J., 2018b. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. Comput. Vision Patt. Recog. 6848–6856.

Zhivkov, Z., 2004. Improved adaptive Gaussian mixture model for background subtraction. In: International Conference on Pattern Recognition. Cambridge, UK, pp. 28–31.