IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 20, 2017, accepted August 25, 2017, date of publication August 30, 2017, date of current version September 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2747399

A Deep Normalization and Convolutional Neural Network for Image Smoke Detection

ZHIJIAN YIN[1], BOYANG WAN[1], FEINIU YUAN[2], (Senior Member, IEEE), XUE XIA[2], AND JINTING SHI[2,3]

[1]Department of Communications and Electronics, Jiangxi Science and Technology Normal University, Nanchang 330013, China
[2]School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330032, China
[3]Vocational School of Teachers and Technology, Jiangxi Agricultural University, Nanchang 330045, China

Corresponding author: Feiniu Yuan (e-mail: yfn@ustc.edu)

This work was supported in part by the Natural Science Foundation of China under Grant 61363038, in part by the Cultivated Talent Program for Young Scientists of Jiangxi Province under Grant 20142BCB23014, and in part by the Science Technology Application Project of Jiangxi Province under Grant KJLD12066 and Grant GJJ150459.

ABSTRACT It is a challenging task to recognize smoke from images due to large variance of smoke color, texture, and shapes. There are smoke detection methods that have been proposed, but most of them are based on hand-crafted features. To improve the performance of smoke detection, we propose a novel deep normalization and convolutional neural network (DNCNN) with 14 layers to implement automatic feature extraction and classification. In DNCNN, traditional convolutional layers are replaced with normalization and convolutional layers to accelerate the training process and boost the performance of smoke detection. To reduce overfitting caused by imbalanced and insufficient training samples, we generate more training samples from original training data sets by using a variety of data enhancement techniques. Experimental results show that our method achieved very low false alarm rates below 0.60% with detection rates above 96.37% on our smoke data sets.

INDEX TERMS Deep neural networks, deep learning, smoke detection, image classification.

I. INTRODUCTION

Smoke detection is an important and effective way of avoiding damages caused by fire. Traditional fire detection methods generally use point based sensors, which are based on smoke particle sampling, atmosphere temperature sampling, and relative humidity sampling [1]. Traditional fire sensors have been widely used because of cheapness, simplicity and accuracy. However, these sensors have some inherent shortcomings that are difficult to overcome. These sensors need to directly sample combustion products for analyses of particles, temperature or humidity. Therefore, these sensors must be installed near places where fire is ignited. This limits these sensors to be applied in small or indoor spaces. Moreover, it may take a long time to transfer combustion products, like smoke particles, to sensors, so it leads to slow response.

In order to overcome the above mentioned shortcomings of traditional fire detection methods, image based fire detection has been widely explored. By analyzing fire cases, we find that smoke often spreads faster than flame. Hence, smoke detection provides earlier fire alarms than flame detection.

A lot of algorithms have been proposed for image smoke detection in the past decades. Toreyin *et al.* [2] proposed an algorithm based on features of flicker, motion, edge blurring and color for smoke detection. Gubbi *et al.* [3] proposed a video smoke detection method by handcrafting features, which consists of geometric mean, standard deviation, skewness, kurtosis, arithmetic mean and entropy over every subband of wavelet transformed images. Yuan [4] proposed several smoke detection methods, which are a fast accumulative motion orientation model based on integral image, histograms of Local Binary Pattern (LBP) and Local Binary Pattern Variance (LBPV) based on pyramids [5], shape-invariant features on multi-scale partitions with AdaBoost [6] and high-order local ternary patterns with locality preserving projection [7].

Most of existing algorithms are based on the framework of handcrafted features. However, it is very difficult for these existing algorithms to achieve low false alarm rates without decreasing detection rates. The main reason is that smoke color, texture and shapes vary hugely. In addition, smoke blurs visual scenes, thus leading to unstable features. It is a complicated and expensive task to handcraft discriminative features for smoke detection.

In recent years, deep convolutional neural networks (CNNs) have demonstrated excellent performance on generic visual recognition tasks [8], and object detection

and image classification [9], [10]. However, little researches on deep convolutional neural networks (CNNs) for smoke detection have been reported.

In this paper, we propose a deep normalization and convolutional neural network (DNCNN) for smoke detection. Unlike traditional handcrafted methods, DNCNN completes both feature extraction and smoke recognition at the same time, so it is an end-to-end method for early fire alarms. In DNCNN, we specially construct a deep neural network to learn features directly from raw pixels of smoke and non-smoke images, so none of handcrafted features are involved.

This paper is distinguished by the following main contributions:

- An end-to-end network is proposed for smoke detection. Our method has better performance and less learnable parameters than several classical Deep CNNs, such as AlexNet [8], ZF-Net [11], and VGG16 [12].
- We replace traditional convolutional layers with normalization and convolutional layers to accelerate the convergence speed of training and improve performance at the same time.
- Insufficient training samples and imbalance of negative and positives samples would lead to overfitting. To solve this issue, a variety of data enhancement techniques are used to generate a large and balanced training set, which helps our network to further improve performance.

The rest of this paper is organized as follows. Section II reviews deep convolutional neural networks. In Section III, our DNCNN is presented in detail. Experiments and results are demonstrated in Section IV. Finally, conclusions are given in Section V.

II. RELATED WORK

A. DEEP CONVOLUTIONAL NEURAL NETWORKS

Hubel and Wiesel [13] revealed the similar structure between the hierarchical model of human visual system and Convolutional Neural Networks (CNNs). Fukushima *et al.* [14] proposed the concept of ''neocognitron'', which is a fundamental work of CNNs. Afterward, LeCun *et al.* [15] reached a milestone of CNNs by proposing LeNet-5 that is a pioneering convolutional neural network. LeNet-5 was applied to recognize hand-written numbers and achieved excellent performance. CNNs are able to deal with high resolution images when more convolutional layers are applied. However, more convolutional layers also require more computational consumption. Therefore, we need to make tradeoffs between the number of convolutional layers and computational efficiency.

With the advent of large-scale datasets and the development of computational resources, the architectures of CNNs have been become very deeper. Deep CNNs have achieved great success in computer vision tasks. Alex *et al.* [8] proposed AlexNet, which achieved excellent performance in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. So far, algorithms based on deep CNNs [12], [16], [17] have taken the dominant place in pattern recognition.

B. BATCH NORMALIZATION

Mini-batch stochastic gradient descent (SGD) has been widely used in the training of deep CNNs. The training efficiency of deep CNNs is severely reduced by internal covariate shift [18], which is the changes of internal input distributions during training. Batch normalization [18] is proposed to alleviate internal covariance shift by transforming internal inputs with a scale and shift step before nonlinear activation. Mean and variance of each activation are calculated to normalize features by equations (1) and (2).

$$\bar{x}_f = \frac{1}{m}\sum_{i=1}^{m} x_{i,f} \tag{1}$$

$$\sigma_f^2 = \frac{1}{m}\sum_{i=1}^{m} (x_{i,f} - \bar{x}_f)^2 \tag{2}$$

where m is the size of a mini-batch, and $x_{i,f}$ is the fth feature of the ith sample in the mini-batch.

Using mini-batch mean and variance, we can normalize each feature as follows:

$$\hat{x}_f = \frac{x_f - \bar{x}_f}{\sqrt{\sigma_k^2 + \xi}} \tag{3}$$

where ξ is a small positive constant to improve numerical stability.

However, the normalization of input features reduces the representation capability of the inputs. To solve this problem, batch normalization introduces two learnable parameters γ_f and β_f for each feature f. Batch normalization is defined by transforming normalized features with a scale and shift step:

$$BN(x_f) = \gamma_f \hat{x}_f + \beta_f \tag{4}$$

The network can recover the distribution of the original inputs by the scale and shift step. Batch normalization has been widely used for CNN-based image classification.

III. APPROACH

A. NETWORK ARCHITECTURE

The network architecture plays a key role in determining the performance of CNN. We construct a novel network by sharing some basic architectures proposed in the classical ZF-Net [11] for smoke detection. Then we modify our network inspired by the idea proposed in [12]. Also, we replace convolutional layers with normalization and convolutional layers. Finally, we propose a deep normalization and convolutional neural network, which is abbreviated to DNCNN. Our DNCNN has fourteen layers, as illustrated in Fig. 1. In DNCNN, the first eight layers are normalization and convolutional layers alternated with three pooling layers for feature extraction, and the remaining three layers are fully connected layers for classification.

1) NORMALIZATION AND CONVOLUTIONAL LAYERS

In the lth layer, the parameter N^l represents the number of feature maps, so each feature map can be expressed as

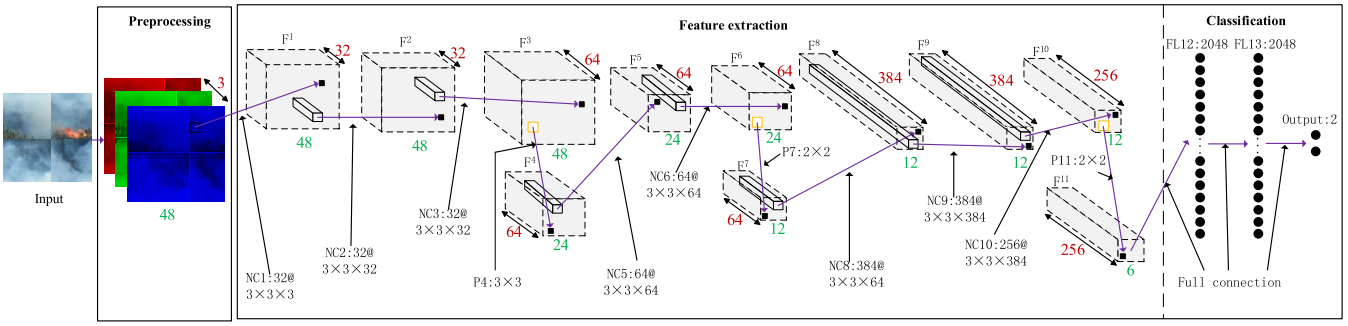

FIGURE 1. The architecture of the deep normalization and convolutional neural network classification system for smoke detection. Each cuboid, which is drawn in dotted lines, denotes a stack of feature maps. F^i stands for a stack of feature maps in the ith layer. Red numbers along depth directions denote the number of feature maps. The shapes of all feature maps are square. Green numbers under cuboids denote the widths and heights of feature maps. The ith normalization and convolutional layer is abbreviated to NCi, and similarly the ith max-pooling layer to Pi. For example, NC8:384@3×3×64 means that this is the 8th normalization and convolutional layer with 384 filters of size 3×3×64, P4: 3×3 means this is the 4th max-pooling layer with the kernel size of 3×3, and FL11:2048 means that this is the 11th fully-connected layer with 2048 neurons.

$\mathbf{F}_j^l (j = 1, 2, \ldots, N^l)$. Generally, convolutional layers are regarded as feature abstract layers. The jth feature map $\mathbf{F}_j^l$ in the lth layer associates all the feature maps $\mathbf{F}_i^{l-1} (i = 1, 2, \ldots, N^{l-1})$ in the (l-1)th layer by filters $\mathbf{W}_{ij}^l$ and bias b_j. Every filter with a bias serves as a feature extractor to extract features by convolving itself with the input feature maps. To obtain $\mathbf{F}_j^l$, each input feature map $\mathbf{F}_i^{l-1} (i = 1, 2, \ldots, N^{l-1})$ is convolved with the corresponding filter $\mathbf{W}_{ij}^l$, and the results are summed up and finally added with the bias b_j. Finally, a nonlinear activation function $\varepsilon(\cdot)$ is used to model the nonlinearity for the neural network. Mathematically, the feature maps of the lth layer can be computed as follows:

$$\mathbf{F}_j^l = \varepsilon\left(\sum_{i=1}^{N^{l-1}} \mathbf{F}_i^{l-1} * \mathbf{W}_{ij}^l + b_j^l\right), \quad j = 1, 2, \ldots, N^l \tag{5}$$

where * denotes the convolution operation.

Once the two parameters γ_f and β_f are learned, we use equations (1), (2), (3), and (4) to perform batch normalization for each feature f.

2) POOLING LAYERS

Normalization and convolutional layers are used to extract features. But the dimensions of the extracted features are too high for classification layers due to limited video memory. In addition, high dimensional features increase the risk of over-fitting and computational time.

To address this problem, a pooling layer, which generally connects to a convolutional layer, has been used to reduce the resolution of convolutional feature maps. Max-pooling or average-pooling are often used in deep learning. The former selects the maximum activation value over a pooling region, while the latter uses the average activation value. Overlapping-pooling has been proposed in [8]. The overlapping-pooling method allows pooling regions to be overlapped, so more information can be extracted from feature maps. The maxing-pooling and overlapping-pooling techniques are applied in our network.

3) CLASSIFICATION LAYERS

Classification layers generally contain one or more fully-connected layers at the end of deep neural networks. Similarly, our network contains three fully-connected layers at the end of the neural network. In most of deep neural networks, fully-connected layers often account for most of learnable parameters. However, fully-connected layers are prone to be overfitting. To overcome this issue, we used the dropout technique [19] to reduce overfitting in the first two fully-connected layers. As shown in Fig. 1, the first fully-connected layer FL12 takes all feature maps of F^{11} as the input neurons. The last fully-connected layer is the final output layer, which takes the output of FL13 as input and is parameterized by a weight matrix $\mathbf{W}^{14}$ and a bias vector $\boldsymbol{b}^{14}$.

$$\mathbf{F}^{14} = \mathbf{W}^{14}\mathbf{F}^{13} + \mathbf{b}^{14} \tag{6}$$

The output layer contains two neurons, which output the probabilities of two classes, i.e., $\widehat{\mathbf{y}} = \left[\widehat{y}_1, \widehat{y}_2\right]^T$. The probability for the jth class is computed by the softmax function, defined as follows:

$$\widehat{y}_j = \frac{\exp\left(\mathbf{F}_j^{14}\right)}{\sum_{i=1}^{2} \exp(\mathbf{F}_i^{14})}, \quad j = 1, 2 \tag{7}$$

where $\widehat{y}_j$ is the output probability of the jth neuron.

B. IMAGE NORMALIZATION

In order to obtain discriminative internal features that can lead to excellent performance, we adopt an image preprocessing method to reduce variance of images and enhance inherent characteristics of images.

Smoke images are captured under varying illumination. So we adopt the min-max normalization method [20] to remove influence of illumination. The normalization is computed pixel by pixel and is defined as follow:

$$x_n = \frac{x_r - x_{\min}}{x_{\max} - x_{\min}} \tag{8}$$

where x_r denotes the intensity value of a pixel, x_n is the normalized intensity value, $x_{\max}$ and $x_{\min}$ denotes the minimum and maximum intensity values of an image, respectively.

C. DATA AUGMENTATION

Deep CNNs often contain millions of parameters to be learned. High recognition accuracy cannot be achieved unless a large number of training images are provided. Another problem is the imbalanced distribution of training images. The number of smoke images is far less than that of non-smoke images in our training dataset. It is labor-intensive and exhausting to generate plenty of training images by capturing or collecting smoke pictures. Instead, we produce more training data from the training set by using data augmentation techniques [8].

FIGURE 2. An example of data augmentation. a) Original images in the training set; b) Corresponding new images generated by horizontal flip; c) Corresponding images by vertical flip; d) Corresponding images rotated by 90 degrees.

There are a variety of image processing methods for data augmentation, including horizontal flip, vertical flip, rotation, and so on. Fig. 2 shows some new images generated from original images through data augmentation techniques.

In this paper, we tried two strategies of data augmentation and compared the two strategies with each other. The first one is that we use all images of the training set to do data augmentation. In other words, we increased the number of the training samples without changing the proportion of smoke samples to non-smoke ones. We mixed newly-generated samples and original samples to obtain a new, larger but imbalanced training set. The second one is that we only used smoke images to generate new positive samples, so a relatively small but balanced training set was created.

Our DNCNN does not achieve good performance on the training set generated by the first strategy. However, the performance of DNCNN is significantly improved when the second strategy is applied. Detailed experimental results are presented in Section IV-C.

D. NETWORK DESIGN

There are many factors affecting performance in the design of deep neural networks. Generally, the design process of a network is to specify a set of hyper-parameters for the network.

Specifically, we need to determine all the values of model-relevant hyper-parameters [21], which are the number and size of filter kernels for convolutional layers, the size of pooling regions for pooling layers, the values of strides, the number of neurons for fully-connected layers, the probabilities for dropout, and activation functions.

We use a trial-and-error method to obtain optimized the hyper-parameters for our network. In particular, we concern about the activation functions in hidden layers. There are three commonly used activation functions, which are sigmoid function $\varepsilon(x) = \sigma(x)$ [22], hyperbolic tangent function $\varepsilon(x) = \tanh(x)$, and Rectifier Linear Unit (ReLU) function $\varepsilon(x) = \max(x,0)$ [8]. The performance provided by the three activation functions has been evaluated. Detailed experimental results are presented in Section IV-B.

E. NETWORK TRAINING

In order to capture discriminative features for smoke detection, we construct DNCNN, which contains about 20 million learnable parameters. It is necessary to appropriately set training parameters for our network in order to achieve fast convergence.

The loss function of our network is defined as the cross-entropy between the output probability vector $\widehat{\mathbf{y}} = \left[\widehat{y}_1, \widehat{y}_2\right]^T$ and the class label vector $\mathbf{y} = [y_1, y_2]^T$. One-hot encoding is adopted to specify label vectors for all samples. The cross-entropy is defined as follows:

$$C(\mathbf{y}, \widehat{\mathbf{y}}) = -\sum_{j=1}^{2} y_j \log(\widehat{y}_j) \tag{9}$$

IV. EXPERIMENTS AND RESULTS

We used Tensorflow [24] and Keras [25] to construct and train the proposed DNCNN. To facilitate comparisons, we also implemented other deep CNNs using Tensorflow and Keras. All experiments were carried out in the Ubuntu 16.04 operation system running on a PC with Inter(R) Xeon(R) E3-1230 v3 CPU 3.30GHz and an Nvidia GTX1060 GPU. It takes about 45 seconds to train DNCNN one epoch on GPU.

TABLE 1. image data sets for training, validation and testing.

Datasets	Number of smoke images	Number of non-smoke images	Total number of images	Purposes for
Set1	552	831	1383	Testing
Set2	688	817	1505	Testing
Set3	2201	8511	10712	Training
Set4	2254	8363	10617	Validation

Table 1 shows four data sets named as Set1, Set2, Set3 and Set4. Set1 has 1383 images including 552 smoke images and 831 non-smoke images. Set2 has 1505 images that are 688 smoke images and 817 non-smoke images, respectively. Set3 contains 10712 images including 2201 smoke images

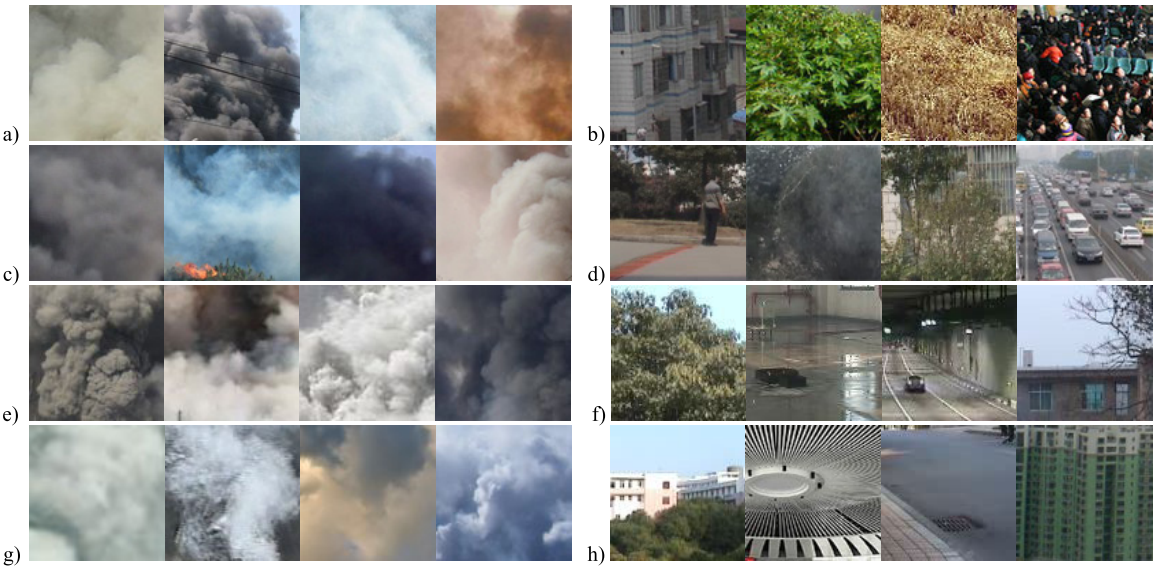

FIGURE 3. Images from the four data sets. a) Smoke images and b) non-smoke images from Set 1. c) Smoke images and d) non-smoke images from Set 2. e) Smoke images and f) non-smoke images from Set 3. g) Smoke images and h) non-smoke images from Set 4.

and 8511 non-smoke images. In Set4, there are 10617 images consisting of 2254 smoke images and 8363 non-smoke images. We adopt the relatively large dataset Set3 for training, Set4 for validation, and other two datasets (Set1 and Set2) for testing.

As shown in Fig. 3, both intro-class and inter-class variances of smoke and non-smoke are very large. That is the main reason why it is very challenging to detect smoke from images.

A. EVALUATION METHODS

To quantitatively compare our method with existing algorithms, we used evaluation methods [7] that are Detection Rate (DR), False Alarm Rate (FAR) and Accuracy Rate (AR), defined as follows:

$$\mathrm{DR} = \frac{P_p}{Q_p} \times 100\% \tag{10}$$

$$\mathrm{FAR} = \frac{N_p}{Q_n} \times 100\% \tag{11}$$

$$\mathrm{AR} = \frac{P_P + N_n}{Q_n + Q_p} \times 100\% \tag{12}$$

where Q_p and Q_n are the numbers of positive and negative samples respectively, P_p is the number of correctly detected true positive samples, and N_p denotes the number of negative samples falsely classified as positive samples.

Actually, DR is the True Positive Rate (TPR) and FAR is False Positive Rate (FPR). Our goal is to achieve high AR, high DR, and low FAR at the same time.

B. OPTIMIZATION OF NETWORK HYPER-PARAMETERS

We initialized the weights by the glorot-uniform distribution [23]. All of learnable parameters were updated periodically using SGD [15]. The momentum trick [21] was employed to skip local minimums and speed up training, since it can smooth the directions of gradient descent and make the network converge fast. Also, learning rate decay was adopted to decrease learning rates after each training epoch. The detailed training-relevant hyper-parameters are listed in **Table 2**, and these hyper-parameters are mainly referenced from [9]. The training process terminates after correct classification rates of both training and validation sets plateaus at some epochs.

TABLE 2. Training-relevant hyper-parameters.

Hyper-parameters	Initial learning rate	Mini-batch size	Momentum coefficient	Learning rate decay coefficient
Value	0.01	96	0.9	0.01

We conducted experiments to demonstrate the importance of hyper-parameters for network design. We used the original training set without data augmentation for training and testing. The final optimized hyper-parameters are summarized in **Table 3**. To see the influence of hyper-parameters, we changed only one or two parameters of the final hyper-parameters at each time and performed experiments with the changed hyper-parameters. **Table 4** gives evaluation results on the test sets. Variants of network hyper-parameters are assigned empirically and evaluated on a large number of experiments.

As we can see from **Table 4**, the hyper-parameters of our network significantly influence the performance of our network. We carefully tuned all the hyper-parameters until excellent performance was obtained. For our DBCNN, with the hyper-parameters described in **Table 2** and **Table 3**, we can achieve AR of 97.83%, DR of 95.28% and FAR of 0.48% on Set1, and AR of 98.08%, DR of 96.36%, FAR of 0.48% on Set2. In summary, overlapped max-pooling achieves

TABLE 3. Model-relevant hyper-parameters.

Layer	Layer Type	Hyper-parameters
Input	Input	Image size: 48×48
NC1	Normalization and convolution	Filer size k1×k1: 3×3
		Filer number n1: 32
		Stride: s1×s1: 1×1
		Activation function: ReLU
NC2	Normalization and convolution	Filer size k2×k2: 3×3
		Filer number n2: 32
		Stride: s2×s2: 1×1
		Activation function: ReLU
NC3	Normalization and Convolution	Filer size k3×k3: 3×3
		Filer number n3: 64
		Stride: s3×s3: 1×1
		Activation function: ReLU
P4	Pooling	Pooling region size k4×k4: 3×3
		Stride: s4×s4: 2×2
		Pooling method: max-pooling
NC5	Normalization and convolution	Filer size k5×k5: 3×3
		Filer number n5: 64
		Stride:s5×s5: 1×1
		Activation function: ReLU
NC6	Normalization and convolution	Filer size k6×k6: 3×3
		Filer number n6: 64
		Stride: s6×s6: 1×1
		Activation function: ReLU
P7	Pooling	Pooling region size k7×k7: 2×2
		Stride: s7×s7: 2×2
		Pooling method: max-pooling
NC8	Normalization and convolution	Filer size k8×k8: 3×3
		Filer number n8: 384
		Stride: s8×s8: 1×1
		Activation function: ReLU
NC9	Normalization and convolution	Filer size k9×k9: 3×3
		Filer number n9: 384
		Stride:s9×s9: 1×1
		Activation function: ReLU
NC10	Normalization and convolution	Filer size k10×k10: 3×3
		Filer number n10 : 256
		Stride:s10×s10: 1×1
		Activation function: ReLU
P11	Pooling	Pooling region size k11×k11: 2×2
		Stride:s11×s11: 2×2
		Pooling method: max-pooling
FL12	Fully-connected	Neurons number n12: 2048
		Dropout probability p: 0.5
		Activation function: ReLU
FL13	Fully-connected	Neurons number n13: 2048
		Dropout probability p: 0.5
		Activation function: ReLU
Output	Output	Neurons number n14: 2

better performance than non-overlapped max-pooling in bottom layers. Besides, appropriately reducing the number of neurons in fully-connected layers not only shortens the convergence time but also improves the recognition ability for smoke detection.

C. EXPERIMENTS WITH DATA AUGMENTATION

To get more training samples and explore a right strategy for combination of new and original training samples, we used three image processing methods, which are horizontal flip, vertical flip and central rotation, to generate three new images from each original image. Then we put new and original images together to create two new training datasets in two different ways.

TABLE 4. Evaluation of performance with different hyper-parameters.

Network Design	Set1			Set2		
	AR(%)	DR(%)	FAR(%)	AR(%)	DR(%)	FAR(%)
k4=2	97.32	94.20	0.60	97.07	94.18	0.49
k7=k11=3	97.10	93.65	0.60	97.87	95.93	0.49
n12=n13=4096	96.39	94.20	2.16	96.75	95.05	1.83
$\varepsilon(x)=\sigma(x)$	97.78	95.10	0.84	97.33	94.91	0.61
$\varepsilon(x)=\tanh(x)$	96.67	93.47	1.20	**98.33**	**97.38**	0.85
hyper-parameters in **Table 3**	**97.83**	**95.28**	**0.48**	98.08	96.36	**0.48**

TABLE 5. Original and augmented datasets.

Datasets	Number of smoke images	Number of non-smoke images	Total number of images
Set3	2201	8511	10712
SetL	8804	34044	42848
SetB	8804	8511	17315

TABLE 6. Experiments with data augmentation.

Datasets	Set1			Set2		
	AR(%)	DR(%)	FAR(%)	AR(%)	DR(%)	FAR(%)
Set3	97.83	95.28	**0.48**	98.08	96.36	0.48
SetL	97.57	94.74	0.77	98.33	96.65	**0.24**
SetB	**98.19**	**96.37**	0.60	**98.53**	**97.52**	0.61

We combined original images with the newly generated ones in two different ways. Thus we constructed two new training sets named SetB and SetL. SetB consists of original images and part of newly generated smoke images that are almost balanced, while SetL contains original images and all of newly generated smoke images. Therefore, SetL is still imbalanced since the original Set3 is imbalanced. Details about the two augmented datasets are presented in **Table 5**. We used these three different datasets, i.e., Set3, SetB and SetL, to train our DNCNN with the training hyper-parameters described in **Table 3**. The testing results are listed in **Table 6**.

Data augmentation techniques can improve performance. Our DNCNN learned on SetB achieved better performance than that on Set3. When we learned DNCNN on SetB instead of Set3, AR increases from 97.83% to 98.19% on Set1 and from 98.08% to 98.53% on Set2, DR increases from 95.28% to 96.37% on Set1 and from 96.36% to 97.52% on Set2, and FARs increase only about 0.1% on Set1 and Set2. The experimental results indicate that overfitting can be reduced when a larger and balanced training set is provided, and the balance of training sets is also an important factor for improving effectiveness of data augmentation.

D. COMPARISON EXPERIMENTS

1) COMPARISONS WITH CLASSICAL DEEP CNNs

We compared our DNCNN with several classical deep CNNs, which have achieved excellent performance on generic visual

TABLE 7. Comparisons with classical deep CNNs.

Deep CNNs	Input image size	Set1			Set2			Learnable parameters
		AR(%)	DR(%)	FAR(%)	AR(%)	DR(%)	FAR(%)	
AlexNet	227	97.18	93.29	**0.24**	98.04	96.07	**0.12**	60 Million
ZF-Net	224	97.18	93.29	0.24	97.01	94.33	0.73	60M
VGG16	224	97.48	**96.19**	0.60	97.48	95.05	0.48	120M
DNCNN (without BN)	48	94.86	90.21	2.04	95.94	93.02	1.59	20M
DNCNN	48	**97.83**	95.28	0.48	**98.08**	**96.36**	0.48	20M

recognition tasks. The classical deep CNNs for comparisons include AlexNet [8], ZF-Net [11], and VGG16 [12]. We basically used the training-relevant hyper-parameters described in **Table 2** but the size of mini-batch was set to 32. We adjusted the size of input images in order to adapt to different network architectures. For the sake of fair comparisons, the training set without data augmentation was used for all the compared algorithms. To gain insight into our network, our DNCNN without using batch normalization (BN) was also tested to evaluate effects of BN layers. In addition, all the networks are trained from scratch.

The experimental results are presented in **Table 7**. On Set1 and Set2, our network achieved the highest ARs among the four networks, and obtained higher DRs than other methods except for VGG16 on Set1. Although our method obtained lower DR than VGG16 on Set1, FAR of our network was reversely lower than VGG16. AlexNet achieved slightly lower FARs than our network on Set1 and Set2, but ARs and DRs of our network were obviously higher than AlexNet. ZF-Net obtained lower FARs on Set1 and Set2, and other testing results were worse than our network. Our network obviously became worse when batch normalization layers were removed. Hence, batch normalization layers play an important role in our network.

In summary, Our DNCNN has better performance than AlexNet, ZF-Net, and VGG16 for smoke detection, and batch normalization really helps improve the generalization ability of our network. In addition, we also find that our DNCNN has less learnable parameters than other three deep CNNs, as shown in **Table 7**.

The training processes of different networks are illustrated in Fig. 4. As shown in Fig. 4, compared to the other three deep CNNs, our DNCNN consumes extremely few iterations to converge. The training accuracy of our DNCNN reaches 100% after about 35 epochs, and then it becomes stable. Our network has similar training performance when it does not use BN. Meanwhile, the training accuracies of other three deep CNNs are not higher than 95%, and they become stable after about 200 epochs. We also observe the same situation in the validation accuracy curves. This means that our DNCNN consumes less computing resources and has fast convergence to capture recognition ability for smoke detection.

From Fig. 4, we can easily see that there is a gap between validation accuracy lines of original DNCNN and DNCNN

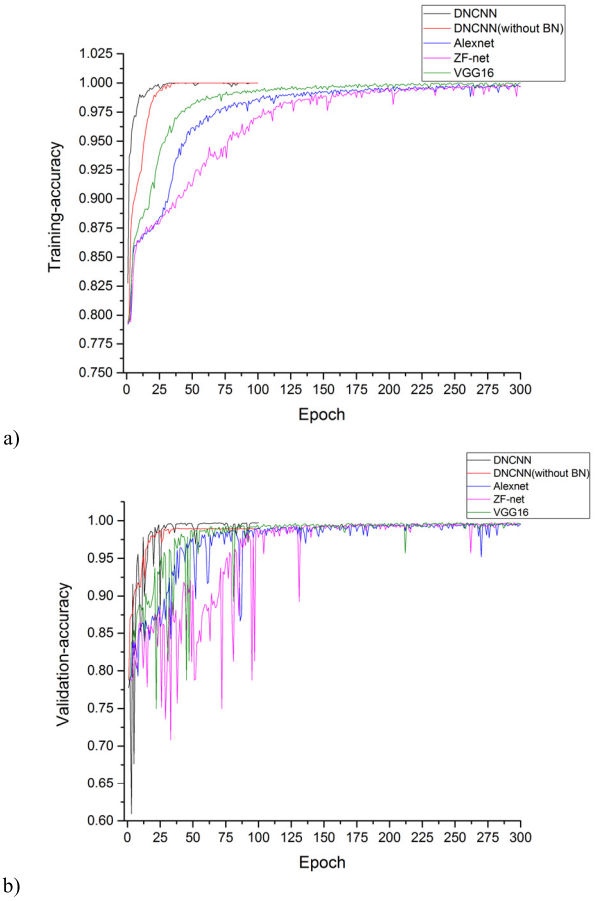

FIGURE 4. Training and validation accuracy curves of DNCNN, DNCNN (without BN), AlexNet, ZF-Net and VGG16. a) The training accuracy curves. b) The validation accuracy curves.

without BN. The reason is that DNCNN without BN brings the problem of overfitting, while the original DNCNN avoids this problem. Since BN is one of the components in our method, we only involve the original DNCNN in the coming comparison experiments.

2) COMPARISONS WITH TRADITIONAL ALGORITHMS

To demonstrate the advantages of deep learning based smoke detection method, we compared the proposed DNCNN with traditional smoke detection methods consisting of hand-crafted feature extraction and classification. Smoke can be

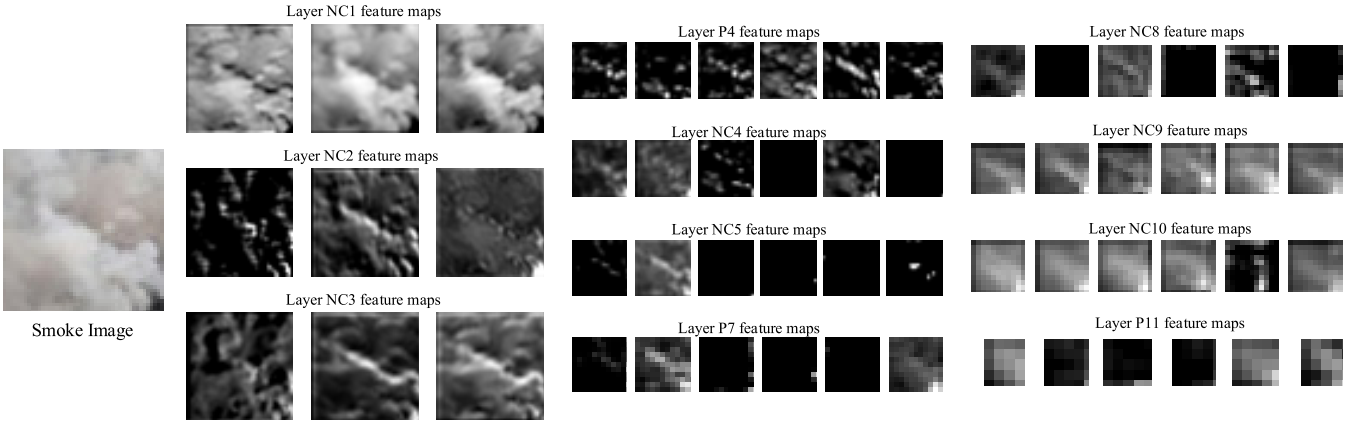

FIGURE 5. Visualization of feature maps in convolutional and pooling layers learned by our DNCNN on SetB.

TABLE 8. Comparisons with traditional algorithms.

Algorithms	Set1			Set2		
	AR(%)	DR(%)	FAR(%)	AR(%)	DR(%)	FAR(%)
DNCNN	**98.19**	96.37	**0.60**	**98.53**	97.52	**0.61**
HLTPMC [7]	96.46	**98.00**	4.57	98.48	**99.56**	2.44
MCLBP [26]	97.47	96.37	1.80	97.41	96.65	1.95

TABLE 9. Evaluation of different classification layers.

Number of fully connected layers	Set1			Set2			Dimension of feature
	AR(%)	DR(%)	FAR(%)	AR(%)	DR(%)	FAR(%)	
3	**97.83**	**95.28**	**0.48**	**98.08**	**96.36**	**0.48**	**2048**
2	97.46	94.74	0.72	96.87	93.75	0.48	2048
1	60.08	0.00	0.00	54.28	0.00	0.00	6*6*256

regarded as a specific kind of texture, and it has been proved that texture feature is discriminative in representing smoke. Support Vector Machine (SVM) is one of the most widely used classifiers. So two texture descriptor based methods, HLTPMC [7] and MCLBP [26], are involved in our comparison experiments.

Traditional algorithms often do not use data augmentation. However, data augmentation is usually a part of deep learning-based algorithms, so our DNCNN also uses data augmentation to generate the balanced training set SetB for training. The testing results on Set1 and Set2 are presented in **Table 8**.

Our algorithm achieved slightly lower DRs but higher ARs than HLTPMC. However, our method obtained far lower FARs than HLTPMC. It is very important for smoke detection methods to achieve very low FARs with acceptable DRs. Our method obviously outperformed MCLBP with respect to DRs, FARs and ARs.

Compared to traditional algorithms based on handcrafted features, our deep learning-based algorithm apparently achieves high detection rates and low false alarm rates.

E. VISUALIZATION AND ABLATION

1) VISUALIZATION OF DNCNN FEATURES

To gain insight into the behavior of deep CNNs, visualization is a common practice. In order to understand features generated in DNCNN, we visualized the feature maps learned by our network.

Fig. 5 shows the visualization results of feature maps generated from a smoke image. As we can see, feature maps in layer NC1 have textures that are similar to the smoke image, so the filters in the first layer behave like texture detectors. We also observe similar results in the second and third layers. However, we find the feature maps in the two layers have more and more pixels that have zero values. The reason is that the ReLU activation function has the property of sparsity. As the size of the feature maps decreases in higher layers, features become more and more abstract, and they cannot be defined exactly and explained explicitly.

2) ABLATION ANALYSIS OF OUR NETWORK

As shown in Fig. 1, there are three fully connected layers in our network. The last fully connected layer is used for classification, while the other two are used to extract feature vectors from feature maps obtained by preceding layers.

To study effectiveness of different components of our network, we analyzed the results on the original smoke datasets for each of the three fully connected layers.

The experimental results are presented in Table 9. The DNCNN with three fully connected layers achieves the best performance, but the network performance gets slightly worse when one of the first two fully connected layers is removed. Even worse, the network does not converge when only the last fully connected layer is retained. The reason is that without the first two fully connected layers, the dimension of features that are fed to the softmax classifier is so high that back propagation is dismissed. Therefore, although convolutional layers provide representational power, fully connected layers are also indispensable for our network.

V. CONCLUSION

To improve performance, we propose a deep normalization and convolutional neural network (DNCNN) for smoke detection. Our network uses batch normalization to speed up the training process and boost accuracy of smoke detection. Unlike algorithms based on handcrafted features, our DNCNN can automatically extract features for smoke detection. Experimental results show that our DNCNN achieves high detection rates and low false alarm rates at the same time. Moreover, we also validate the importance and effectiveness of data augmentation for our DNCNN, especially when the training samples are insufficient and imbalanced.

Our future work will focus on recent breakthroughs of deep learning to improve deep learning based smoke detection methods. For example, GANs [27] have achieved great success in image generation, so we will investigate data augmentation by GANs for smoke detection. Also, we will explore better network architectures to extract features for smoke detection.

REFERENCES

[1] F. Yuan, "Rotation and scale invariant local binary pattern based on high order directional derivatives for texture classification," *Digit. Signal Process.*, vol. 26, pp. 142–152, Mar. 2014.
[2] B. U. Töreyin, Y. Dedeoğlu, and A. E. Çetin, "Wavelet based real-time smoke detection in video," in *Proc. 13th Eur. Signal Process. Conf.*, Sep. 2005, pp. 1–4.
[3] J. Gubbi, S. Marusic, and M. Palaniswami, "Smoke detection in video using wavelets and support vector machines," *Fire Safety J.*, vol. 44, no. 8, pp. 1110–1115, Nov. 2009.
[4] F. Yuan, "A fast accumulative motion orientation model based on integral image for video smoke detection," *Pattern Recognit. Lett.*, vol. 29, no. 7, pp. 925–932, 2008.
[5] F. Yuan, "Video-based smoke detection with histogram sequence of LBP and LBPV pyramids," *Fire Safety J.*, vol. 46, no. 3, pp. 132–139, Apr. 2011.
[6] F. Yuan, "A double mapping framework for extraction of shape-invariant features based on multi-scale partitions with AdaBoost for video smoke detection," *Pattern Recognit.*, vol. 45, no. 12, pp. 4326–4336, 2012.
[7] F. Yuan, J. Shi, X. Xia, Y. Fang, Z. Fang, and T. Mei, "High-order local ternary patterns with locality preserving projection for smoke detection and image classification," *Inf. Sci.*, vol. 372, pp. 225–240, Dec. 2016.
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. 2012, pp. 1097–1105.
[9] Z. Gao, L. Wang, L. Zhou, and J. Zhang, "HEp-2 cell image classification with deep convolutional neural networks," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 2, pp. 416–428, Feb. 2017.
[10] J. Geng, H. Wang, J. Fan, and X. Ma, "Deep supervised and contractive neural network for SAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2442–2459, Apr. 2017.
[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[12] K. Simonyan and A. Zisserman. (Sep. 2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[13] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, 1962.
[14] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition Cooperation Neural Nets*. Berlin, Germany: Springer-Verlag, 1982, pp. 267–285.
[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[16] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[18] S. Ioffe and C. Szegedy. (Feb. 2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167
[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2006.
[21] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*. Berlin, Germany: Springer, 2012.
[22] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *Proc. Int. Workshop Artif. Neura*, 1995, pp. 195–201.
[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, Mar. 2010.
[24] M. Abadi *et al.* (Mar. 2016). "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available: https://arxiv.org/abs/1603.04467
[25] (2015). F. Chollet. *Keras*. [Online]. Available: http://keras.io
[26] S. R. Dubey, S. K. Singh, and R. K. Singh, "Multichannel decoded local binary patterns for content-based image retrieval," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4018–4032, Sep. 2016.
[27] I. Goodfellow *et al.*, "Generative adversarial nets," in *Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

ZHIJIAN YIN received the M.S. degree from Jiangxi Science and Technology Normal University in 1991. He is currently a Professor with the Department of communications and electronics, Jiangxi Science and Technology Normal University. His research interests include data mining, pattern recognition, and machine learning.

BOYANG WAN received the B.E. degree in electronic and information engineering from the Polytechnic Institute, Jiangxi Science and Technology Normal University, Nanchang, China, in 2012, where he is currently pursuing the M.S. degree in signal and information processing with the School of Communications and Electronics. His research interests include image processing, deep learning, and pattern recognition.

FEINIU YUAN received the B.Eng. and M.E. degrees in mechanical engineering from the Hefei University of Technology, Hefei, China, in 1998 and 2001, respectively, and the Ph.D. degree in pattern recognition and intelligence system from the University of Science and Technology of China (USTC), Hefei, in 2004. From 2004 to 2006, he held a post-doctoral position with the State Key Laboratory of Fire Science, USTC. From 2010 to 2012, he was a Senior Research Fellow with Singapore Bioimaging Consortium, Agency for Science, Technology and Research, Singapore. He is currently a Professor and a Ph.D. Supervisor with the Jiangxi University of Finance and Economics. His research interests include 3D modeling, image processing, and pattern recognition.

XUE XIA received the B.E. degree in film & TV arts and technology and the M.E. degree in communication and information engineering from Shanghai University, Shanghai, in 2011 and 2014, respectively. She is currently pursuing the Ph.D. degree with the School of Information Technology, Jiangxi University of Finance and Economics, Nanchang, China. Her research interests include 3D display technology, image processing, and pattern recognition.

JINTING SHI received the B.E. degree in computer science and technology from Jiangxi Normal University, Nanchang, China, in 2003, and the M.S. degree in computer science and technology from Jiangxi Agricultural University, Nanchang, in 2008. She is currently pursuing the Ph.D. degree with the School of Information Technology, Jiangxi University of Finance and Economics, Nanchang, China. Her research interests include image processing and pattern recognition.