Automatic wildfire monitoring system based on deep learning

Yingshu Peng\textsuperscript{ab} and Yi Wang\textsuperscript{c}

\textsuperscript{a}Lushan Botanical Garden, Chinese Academy of Sciences, Jiangxi, PR China; \textsuperscript{b}College of Forestry, Nanjing Forestry University, Nanjing, PR China; \textsuperscript{c}Smart City Research Institute, Jiangsu Wiscom Technology Company Limited, Nanjing, PR China

\textbf{ABSTRACT}

Fire detection based on computer vision technology can avoid many flaws in conventional methods. However, existing methods fail to achieve a good trade-off in accuracy, model size, speed, and cost. This paper presents a high-performance forest fire recognition algorithm to solve the current problems in forest fire monitoring. Firstly, visual saliency areas in motion images are extracted to improve detection efficiency. Secondly, transfer learning techniques are employed to improve the generalization performance of the constructed deep learning classification model. Finally, fire detection is realized based on C++ deployment algorithms. Compared with the existing forest fire detection methods, the proposed method has higher classification accuracy and speed, with a more comprehensive application range and lower cost. The performance of our method can meet the accuracy and speed requirements of real-time fire detection, and it can be deployed and practiced on multiple platforms.

\textbf{Introduction}

Uncontrolled forest fires will seriously harm the local communities and threaten the stability of the ecosystem (X. Wang et al., 2017). Early fire detection can substantially reduce the damage caused by fires. According to physical features of smoke and flame, researchers have developed a variety of fire contact alarm sensors, such as smoke sensors, temperature sensors and particle sensors (Allison et al., 2021). These sensors are inexpensive and easy to deploy. However, contact sensors are only suitable for closed spaces and it takes a while to trigger an alarm which may result in missing the best time to extinguish the fire. Therefore, this method is not suitable for large-scale forest environments. Compared with traditional physical sensors, computer vision technology has many advantages in fire monitoring and early warning. It can be used to monitor forest fires in real time and quickly provide the location and intensity of the fire. Vision fire detection techniques can be divided into vision flame detection and vision smoke detection. The color of the smoke is not obvious and has a fuzzy character. In addition, regular cameras fail to work in the dark or in the presence of smoke. Therefore, vision-based flame detection systems are more popular in forest environments. (Jan et al., 2016; Shi et al., 2020; Xie et al., 2020).

The main methods of vision flame detection are dynamic fire detection based on forest inspection (Hinkley & Zajkowski, 2011) and on-site monitoring with fixed cameras (Burnett & Wing, 2018). It is an effective way to prevent fire by patrolling forests with unmanned aerial vehicles (UAVs). UAVs also have many shortcomings, such as high price, insufficient endurance, being easily affected by weather, and the need for professional staff to operate and maintain, which leads to this method not widely applied in practice (Allison et al., 2016; Berie & Burud, 2018; Merino et al., 2012). Scenarios monitoring methods based on fixed cameras have the characteristics of low cost, fast response speed and low personnel requirement, can be applied to forest environment on a large scale.

Currently, there are mainly three categories of vision-based flame detection systems. (1) hand-designing complex fire features and then utilizing a traditional machine learning algorithm for image classification (Barmpoutis et al., 2020). (2) automatic feature extraction and classification based on a deep learning network (Gaur et al., 2020). (3) flame object detection and recognition based on deep learning (Li & Zhao, 2020). Each of these methods has its own advantages and limitations.

Manual design of fire features mainly depends on the expert’s engineering experience, for example, according to features of flame movement, color, texture and frequency, to identify whether there is a flame at the scene. These features are highly targeted and interpretable, but show poor scalability. Moreover, when the environment changes, the manually extracted features will fail, resulting in lower overall accuracy. A plethora of studies have already demonstrated that fire detection algorithms based on deep neural networks can alleviate these problems well (Geetha et al., 2021). Deep learning automatically
extracts features from large amounts of data and classifies them, avoiding cumbersome pre-processing and getting a lot of features that cannot be manually defined. However, this method is extremely susceptible to static targets and causes more false positives (Barmpoutis et al., 2020). Fire object detection based on deep learning can achieve a high recognition rate (Li & Zhao, 2020). However, compared to the image classification method, this method requires a large number of manually labeled samples, which are costly, have a high false alarm rate and slow speed.

In a word, some studies on improving the speed and accuracy of fire detection have achieved excellent performance in particular scenarios (Li & Zhao, 2020). So far as we know, there is no method that can be accurately, economically and quickly applied to a complex forest environment. The paper analyses the present situation of computer vision detection algorithms. The following problems are found in existing research and applications.

1. Whether it is the application of manual design features or deep learning methods to forest fires, the size of the training dataset generally is small (no more than 50,000). Moreover, most studies use the flame image dataset to verify the algorithm effect, but rarely use the fire dynamic video, which results in the validity of the algorithm that needs further validation (Geetha et al., 2021).

2. Generally speaking, forest fires relatively seldom happen in practical situations. However, the existing fire detection algorithm development process often focuses only on the fire recognition ability increased, while ignoring the false alarm rate and generalization ability of the algorithm, which causes the algorithm to be eventually deprecated by users due to a large number of false positives.

3. Some algorithms rarely care about model size and inference speed (Li et al., 2022). Existing detection algorithms are mostly based on Python language development. Python applications are cumbersome, hard to maintain, and slow to run, which makes it very hard to deploy in actual environments (Marowka, 2018). In a realistic environment, many cameras will monitor the forest environment at the same time. Therefore, the slow speed of model inference will allow the requirement of hardware equipment to be further higher, which leads to the deployment cost of algorithms increasing greatly.

To solve the above problems, this paper puts forward a new application solution for fire detection in the real forest environment. Based on image processing technology, fusing handicraft features and deep features, and using transfer learning (Weiss et al., 2016) to classify flame images, so as to realize flame detection. In addition, we comprehensively evaluated the performance of the algorithm and the practical engineering application effect, and verified the effectiveness of the proposed method. The detection method can alleviate the problem of false positives by image processing technology, and further improve the accuracy of deep learning image classification. The deep learning classification algorithm we used has the advantages of fast detection speed and high recognition accuracy.

Related work

Fire detection method based on physical sensor

At present, the current sensor-based has been applied to fire monitoring and provides early warning information for monitoring systems. These fire alarm sensors mainly include gas sensors, temperature sensors, optical sensors, infrared sensors, and so on (Zhang et al., 2021). Li et al. proposed a distributed fiber temperature sensor and a temperature early warning model for fire detection, whose experimental results demonstrate that their sensor can forecast the temperature trend in advance of 46s (Li et al., 2019). A distributed feedback laser-based carbon monoxide sensor has been developed for early warning detection of fire was proposed by Qiu et al (Qiu et al., 2019). In their experiments, the alarm response times were reported: 110 s for foam and A4 paper; 250 s for polyvinyl chloride, bee wood and cotton rope. Obviously, sensor-based fire detection is not suitable for rapid-fire alarms and remote detection.

Fire detection method based on handcrafted feature

Fire detection algorithms using various handicraft features have been extensively studied. These features are mainly based on the flame’s physical properties, including color, motion, texture, flickering frequency, wavelet information and so on (Shahid et al., 2022). Fire detection algorithms based on color, motion and texture features have been widely reported in many works, but in reality, many objects may have similar characteristics to flames (Geetha et al., 2021). Therefore, further measures need to be taken to exclude non-fired objects from fire detection. Many probabilistic models are based on flame flickering frequency and wavelet information, and heuristic rules are used to discriminate between fires objects and non-fire objects (Nguyen et al., 2021). However, these algorithms are difficult to apply in practice because it is difficult to determine the optimal threshold for classification.

Fire detection method based on deep learning

Over the years, a large number of studies have been conducted employing deep learning to tackle fire detection tasks. These studies are divided into two
main categories: fire image classification and fire object detection. Recent works of fire image classification mainly use various CNN models to classify suspicious regions (P. Wang et al., 2021). The advantages of using CNNs to identify fires in images are relatively high accuracy and recognition speed. One limitation of CNN-based methods is that their accuracy can be compromised when fire areas are small, which is an inherent limitation of CNN architecture due to their fixed size receptive fields (Antioquia et al., 2019). Hence, this method needs to be combined with other technologies to further improve accuracy. Fire detection algorithms based on deep learning object detection have been extensively studied. They commonly fine-tune classic object detection models to achieve better performance (Zhao et al., 2022). These methods can achieve good results in specific data sets. Nevertheless, fires have an arbitrary shape, size, and even location on the image, making it harder to recognize. Moreover, the object detection algorithm requires more label data to train the model, which brings more complexity. Therefore, it is very costly to apply an object detection algorithm to fire detection.

The proposed method

Steps of the proposed method

This paper proposes a high-performance flame fast detection method suitable for the forest environment. The proposed method combines image processing technology and deep learning technology to monitor the fire. The main steps of this method include three parts: motion detection, visual saliency detection (VSD), and flame image classification based on transfer learning. This method has high computational efficiency and quite good detection results, and can be easily extended to other applications. The flowchart of the proposed method is shown in Figure 1 and the general procedures are summarized as follows:

Step 1: Capture images from device cameras for further processing frame by frame.

Step 2: A fast background subtraction algorithm is provided to extract moving objects. Then, the minimum rectangle bounding moving image that contains all moving objects is obtained for subsequent processing.

Step 3: Three salient region images are further extracted from moving images using VSD technology.

Step 4: Transfer learning is used to train models, and the best performance network among them is selected to classify images. When at least two salient region images are classified as flame images, it is determined that flame targets exist in the current detection.

Step 5: Present an algorithm deployment solution based on C++ language and NCNN deployment framework, and implement engineering application of fire detection system.

Step 6: The performance of our system is evaluated using realistic videos.

Motion detection

In previous studies, researchers used several hand-designing image features such as color, texture and shape to realize fire recognition (Barmoputis et al., 2020). Nevertheless, hand-crafted features are often not robust to variations, such as different lighting conditions, cluttered background and occlusions (J. Chen et al., 2010; Dimitropoulos et al., 2015). Generally, the flame has obvious features of continuous motion which are not affected by the change of environment. Thus, this paper employs an improved algorithm based on flame motion features to filter non-moving areas. We will find moving objects from the input image and determine whether they are flames. Deep learning has a high false alarm rate for recognizing static objects similar to flame, but it can alleviate this problem and greatly improve accuracy when combined with a motion detection algorithm (Machado et al., 2021).

The main motion detection algorithms include the frame difference method, the background subtraction method and the optical flow method (Lin et al., 2017; Barmoputis et al., 2014). The frame difference method cannot completely extract motion regions, and the optical flow method is too complex and time-consuming for real-time processing, while the background subtraction method can completely acquire motion regions with less computation. The motion background subtraction method is the most commonly used and effective method in practice. It mainly uses various methods to fit models to obtain a background image, and then compare the pixels of a detected image with a background model and consider those that differ from the background model as moving objects (Barnich & Droogenbroeck, 2011). Commonly used motion detection algorithms include K-nearest neighbors (KNN) model and background subtraction based on Gaussian mixture model (MOG2; Zivkovic & Heijden, 2006). However, the aforementioned background subtraction algorithms need to process each frame to build a complex background model, which will consume a lot of time.

The CouNT (CNT) algorithm is a very quick and simple background subtractor (Zeevi, 2020), which is used to extract the background from an image in our experiments. The principle of CNT is straightforward, as it mimics the human perception of background, that is, if something in the visual field does not change for a certain period of time, the human
brain regards it as part of the background environment (Karbowiak & Bobulski, 2022). Specifically for individual pixels, the CNT counts the number of frames whose pixel value remains unchanged. If the invariant counts exceed a special threshold, the pixel is a background point, otherwise, it is a moving point. In our experiment, the threshold is set to 60. We only need to traverse the image and perform some simple addition and subtraction operations, thereby quickly locating moving objects. More importantly, these operations are not time-consuming, so CNT with a small-time required to process each frame makes our background subtraction greatly desirable for the broader problem of real-time flame detection. Finally, the minimum rectangle bounding image that contains all moving objects is obtained for the next processing step.

**Visual saliency detection**

Essentially, salience is what is most distinctive in an image or scene, enabling humane eye-brain connections to focus rapidly on the most important regions. The VSD technology, an intelligent technology that simulates human vision and extracts saliency regions from images, is applied to many aspects of computer vision and image processing, including some of the popular applications such as object detection, robotics, and so on (Cong et al., 2019). The flame is visually striking because of its color and irregular movement. Therefore, we introduce VSD technology that can mainly detect flames in the image and naturally ignore background areas. On the one hand, the color and texture features of flames are very different from those of background and other moving objects. This
leads to VSD technology can easily regard flames as important areas and reduce flame under-reporting. On the other hand, if the moving region image is fed directly into the deep learning network, the areas with smaller flames in the moving region image are usually ignored. Using VSD technology can further narrow detected areas to effectively reduce the loss caused by changing image size.

Commonly used VSD algorithms are mainly divided into three categories in terms of their technical categories (Cheng et al., 2014):

1. Motion saliency: this algorithm usually depends on detecting a video frame by frame. The motion saliency algorithm processes each image from the video and continuously tracks moving objects, and these moving objects are generally considered salient. However, image interference and object occlusion may cause the loss of salient objects.

2. Static saliency: this kind of algorithm relies on image features and statistical information to locate salient areas in the image. Although the algorithm has the advantage of speed, it cannot capture deep image features to get good performance.

3. Objectness: this algorithm uses various models to generate bounding boxes that may contain salient objects, which can effectively and quickly detect the most salient objects.

For this research, BING: Binarized normed gradients for objectness estimation at 300fps (Objectness BING), an effective saliency detection algorithm, is used to get salient regions from images (Cheng et al., 2014). This method can accurately locate salient objects and give the corresponding marking frame of target suggestion areas. The ObjectnessBING algorithm has a good detection speed. Even if the image size is increased, the speed of the algorithm increases linearly at most which is acceptable (Hosang et al., 2016). However, there may be tens of thousands of suggested areas obtained by the ObjectnessBING algorithm that need to be screened. In order to reduce the workload of the following steps, only the top three regions of visual saliency are extracted for subsequent detection.

For detection purposes, VSD technology is consistent with the general deep learning object detection method, both extracting specific objects. However, the deep learning object detection method requires a large amount of labeled data to achieve high accuracy and faces the problem of slow detection speeds (Wu et al., 2021), which increases the cost of applying a deep learning detection model to flame detection.

The ObjectnessBING algorithm is a simple but efficient VSD algorithm, which is a two-stage cascade classifier in essence (R. Wang et al., 2019). In the first stage, for each image in the training set, different sizes of sliding windows are used to traverse the images. Each window is adjusted to a fixed size of $8 \times 8$, and then extracts the norm of the gradient (NG) of each window as a simple 64 dimensions feature. A linear scoring model used to calculate the possibility of a window containing the object can be defined as:

$$
\begin{align*}
    s_i &= (w, g_i) \\
    l &= (i, x, y)
\end{align*}
$$

where $i$ and $(x, y)$ is size and top left corner coordinates of a window, respectively. $g_i$ represents the NG feature of a window. The $s_i$ denotes the probability that a window contains an object, that is, the higher the value, the greater the probability of containing an object. The “<·>” and $w$ represent the inner product and the weight parameter, respectively.

The NG features of windows containing the real object and that of windows containing random sampling backgrounds are set as positive and negative samples, respectively. So, we can train linear support vector machines (SVM) to get $w$, and finally create the scoring model.

Considering that different sizes of windows have different possibilities of containing object instances, and some windows of tiny sizes are less likely to contain objects. For example, a small window of $2 \times 2$ is unlikely to contain an object, even if its classifier output score is very high. Therefore, a calibrator for updating window scores is trained for each size candidate window in the second stage, and is defined as follows.

$$
\omega_l = \nu_l \times s_l + t_l
$$

where $\nu_l$ and $t_l$ represent the calibration coefficients for different sizes of windows. We train other SVM models to learn parameters $\nu_l$ and $t_l$.

For applying ObjectnessBING in our experiments, we first train the two cascading SVMs in ObjectnessBING with images containing flame objects. Once getting a trained BING model, it can be used to generate flame candidate regions during the detection process, and use non-maximal suppression to choose a small number of proposals from these regions. Finally, to balance accuracy and speed, only the three proposal regions with the highest scores are used, and they are serially fed into our fire detection model.

**Flame image classification based on transfer learning**

The size of the training data is an important factor that limits the performance of fire detection models. Although the best way to overcome these problems is to require massive amounts of labeled data for training models, it is time-consuming and expensive. The image classification algorithm is easier to create a large number of image samples than object detection
algorithms that need to manually mark a large number of images. Therefore, we focus on convolution neural network (CNN)-based flame image classification in this study. Specifically, we train a lightweight deep learning image classification model to accomplish flame image detection. To further improve detection accuracy, we apply transfer learning to migrate the knowledge learned from the ImageNet dataset to our fire dataset. Fine-tuning pre-trained models instead of starting from scratch can improve training efficiency (Zhang et al., 2019).

Although most images in the ImageNet dataset have nothing to do with flames, our model trained on this dataset can extract more general image features, which are helpful to identify edges, textures, shapes and object composition. These similar features are also effective in identifying flames. In this article, we use a common technology in transfer learning: fine-tuning. The fine-tuned models usually get higher accuracy in the same training period. As shown in Figure 2, fine-tuning includes the following steps:

1. We use a pretrained deep learning model trained on the ImageNet dataset which is the source domain.
2. Establish a new fire detection neural network model, which replicates all the model structures and parameters except the output layer in the pre-trained model.
3. Add an output layer whose output size is 2 to the fire model, and initialize the model parameters of this layer randomly.
4. The output layer is trained from scratch, and parameters of remaining layers are fine-tuned through pre-trained model parameters.

Normally, hardware performance is insufficient in the outdoor environment, so we want to get a lightweight deep learning model with a good accuracy-speed trade-off. For this purpose, we selected several common deep learning models and compared their performance to choose the best one. These models are SqueezeNet1.1, MobileNetV1-0.25, MobileNetV1-1.0, MobileNetV2-0.25, MobileNetV2-1.0, AlexNet, VGG16, ResNet18, MobileNetV3-Large and MobileNetV3-Small (Guo et al., 2020). In the process of model training, we first crop a random region with a random size and aspect ratio from the image, and then scale this region to a fixed resolution of 224 × 224. Subsequently, apply random transformation in this region, such as flipping, shifting, normalization, and rotation. All models were trained using the Adam optimization scheme and early stopping based on validation set performance (Kingma & Ba, 2017). During model inference, we scale both the height and width of an image to 224 pixels as input and normalize this image. The standard for judging the fire alarm is that in the detection of three salient region images, at least two images are predicted as the flame image.

**Algorithm engineering**

When applying the algorithm to a real scene, we should not only consider the accuracy of the flame

![Figure 2. Schematic diagram of transfer learning in our approach.](image-url)
detection algorithm, but also whether it can effectively solve practical problems. Due to the characteristics of Python itself, the fire detection algorithm based on Python language has some advantages like short development cycle and relatively easy to implement. However, lots of shortcomings also exist in the development process, such as high code coupling degree, low system reusability and slow execution speed. By comparison, C++ language possesses huge potential in actual production due to its advantages, such as concise and efficient, code readable, efficient, and can be developed in collaboration with different systems or even different computer languages. Consequently, we deploy algorithms in a C++ language environment and an experimental platform and provide a comprehensive analysis of their performance. The motion detection, VSD technology and other image pre-processing algorithms proposed in this research are all implemented based on OpenCV and SVM. OpenCV has multiple language interfaces such as C++ or Python, and SVM can also be implemented by the A Library for Support Vector Machines (LIBSVM; Chang & Lin, 2011) library in C++ language.

For deep learning model deployment, the general practice is to design and train a model on the Python language platform, and then migrate the model to the C++ language environment. One is the implementation of C++ API based on the training framework. The power of this approach is that it can completely migrate the model from the Python version to the C++ version. However, this method requires a lot of time to compile the training framework and the deployment process is very complicated.

Another way is to use inference frameworks dedicated to deep learning deployment, such as TensorRT (NVIDIA/TensorRT, 2021), (Openvinotoolkit/openvino, 2021, Tencent/ncnn, 2021), and so on. Various inference frameworks emphasize different application aspects. For example, TensorRT only supports NVIDIA series graphics boards; OpenVINO is mainly aimed at Intel series CPUs. In addition, these two deployment frameworks require a number of dependent libraries and are primarily used for deep learning deployments under the server. By comparison, the NCNN is an optimized high-performance deep learning network inference framework. It has the advantages of completely open source, pure C++ implementation, cross-platform, less dependency on third-party libraries, support for CPU and GPU inference, and provide model encryption. Therefore, considering the difficulty of deployment and platform requirements, we used the NCNN framework to deploy models. The flowchart of the proposed method is shown in Figure 3 and the general procedure was summarized as follows:

1. **Model training**
   We used MxNet (T. Chen et al., 2015; Python) to train deep learning models in the Ubuntu18.04 system. The trained model was saved as a parameter file with a json suffix and a network structure file with a .json suffix.

2. **Model transformation**
   The trained model obtained in MxNet cannot be directly on NCNN, so we must translate MxNet models into a proprietary model that can be used in NCNN. Specifically, the NCNN library and Vulcan SDK tools (used to drive GPU) need to be compiled. Then, we use the model transformation tool shipped with NCNN to transform the model parameter file and model network file of MxNet into a NCNN parameter file with .bin suffix and a NCNN network structure file with .param suffix.

3. **Model inference**
   Developed the model inference program using C++ API of NCNN and accomplished the task of image analysis in our research. The specific steps of the program include (a) data input conversion, (b) loading model, (c) setting the input layer and output layer of the model, (d) inference, and (e) data output.

**Results and discussion**

In order to verify the feasibility of the algorithm, the performance of each module of the algorithm was tested. The experimental equipment is divided into a training environment and a testing environment. In the training environment, training deep learning models based on the Python 3.7 environment under Ubuntu Linux 18.08. The hardware platform used is an Intel (R) Core (TM) i9-9900 K CPU at 3.60 GHz, and an NVIDIA RTX 2080Ti GPU, with 64 GB memory.

In the test environment, performance assessment experiments of algorithm major modules, that is, motion detection, VSD, and flame image deep learning classification, were first carried out under Python 3.7 software platform. Then, we deployed the whole algorithm platform in the C++11 language environment, including using OpenCV4.4 contrib library and LIBSVM library to implement all image processing algorithms and using NCNN library to execute deep learning models. Finally, the whole system is tested through real wild-fire videos, which assure the consistency between algorithm system functions and requirement detection functions. The hardware platform used in the test environment is Core (TM) i7 7700 CPU 3.60 GHz, an NVIDIA RTX 1060 GPU and 32 G memory.
Experimental datasets

Experimental image datasets
A large number of flame images and non-flame images are acquired by several public datasets (Kim & Lee, 2019; Jadon et al., 2019) and web crawlers. Some images used in the deep learning model for fire classification are shown in Figure 4. The selected image contains fire images of different sizes and colors in various conditions. Among all the fire images, the flame region accounts for one-fifth to four-fifths of the total image area. In this way, the edge of the image can be guaranteed to contain the background region. The selected non-fire images all contain moving objects, which are common moving objects in the forest, for example, trees, animals, etc.

According to the above criteria, a total of 50,000 fire images and 50,000 non-fire images were acquired, and a fire image dataset was created based on these images. In order to avoid the particularity and contingency of the diagnosis results, the random selection strategy of training samples was adopted in this study. Specifically, random samples of each condition were used for training, validation, and testing. The fire image dataset was divided into a training set, a validation set, and a test set according to a ratio of 6:2:2. The training set was used to train the model, the validation set was used to adjust the parameters and select the model, and the test set was used to evaluate the performance of the model as a whole.

Experimental video datasets
After the algorithm was deployed in the C++ environment, we collected some real scene videos from the Internet to test the generalization ability of our methods. None of the scenes in these video datasets appear in the training sets. The test video dataset contains 27 videos, of which the first 11 videos contain fire targets and the remaining 16 videos have no flames. (Dimitropoulos et al., 2015). These videos with different resolutions and scenes can ensure that the test results are convincing.

Image pre-processing experiment and analysis

Motion detection
All input images of the datasets were reset to a resolution of 704 x 576 to ensure the accuracy, speed and applicability of the algorithm. Before actual detection, our motion detection algorithm will read 100 frames in advance for background modeling. To
evaluate the performance of the CNT algorithm, we compared the performance and speed of CNT with several other background subtraction algorithms (GMG, KNN, LSBP, MOG, MOG2; Godbehere et al., 2012; Table 1, Figure 5).

Our goal is to extract moving regions from the original image, and then use them as a new image for further analysis. As shown in Figure 5, we compared the pixels of a detected image with a background model and considered those that differ from the background model as moving objects. The white area shows where moving objects are certain, and the black area means background regions. Due to the complex environment, there are distinct image noise regions in the moving binary image, which will affect detection accuracy. These random noises will be eliminated by the image opening operation. So, the image is processed by a morphological operation to remove small isolated dots as well as some minor disturbances such as leaves and animals, while also smoothing contours of moving objects. Finally, since multiple moving areas may appear in the detected image, we extract the minimum bounding rectangle of these moving regions as the output image of the next step.

The commonly used methods MOG and MOG2 are not sensitive to small-amplitude movements, resulting in inaccurate results for these methods. Other motion detection algorithms can also accurately detect motion regions, and among them, KNN and CNT algorithms.

**Table 1. Speed comparison of different motion detection algorithms.**

| Method | Speed/ ms |
|--------|-----------|
| CNT    | 6.22      |
| GMG    | 70.11     |
| GSOC   | 60.71     |
| KNN    | 34.50     |
| LSBP   | 100.09    |
| MOG    | 20.81     |
| MOG2   | 22.11     |
Rapid detection of fires can reduce the environmental damage caused by wildfires, but there is a big gap between different background subtraction algorithms in terms of speed. The detection time of KNN is 7 times that of the CNT algorithm. The time-consuming procedure of the CNT algorithm is much lower than other background modeling algorithms (Table 1). These experimental results demonstrate the superiority of the CNT algorithm to other competing approaches.

**Visual saliency detection**
The image of the moving region may be large, whose suspected flame objects perhaps just cover a small part of the whole image. We can further use the VSD technology to obtain suspected flame areas from mov-
ing region images to aid the deep learning model classification and improve detection accuracy. In this research, we apply ObjectnessBING to extract the three most visually significant images from the moving region image. These three images are then passed as input to a deep learning model for image classification.

Figure 6 shows the top three significant box regions of ObjectnessBING detection results, where darker red box color indicates higher significance. We can easily recognize that objects in the red box are fire and part of the background, while the objects in the dark red box is fire and smoke, and the blue box focuses on people. The above outcomes are virtually the same as the human vision impression, which fully proves the effectiveness of our proposed method.

To verify the strong feature extraction ability of the proposed method, ObjectnessBING is compared in this case study with two other commonly used visual saliency detection algorithms (Spectral residual (Hou & Zhang, 2007) and Fine grained (Montabone & Soto, 2010)). As shown in Figure 6, the ObjectnessBING algorithm can detect objects more accurately and quickly compared with other comparison algorithms. This ObjectnessBING takes a mere 30.57 ms inference time for a single image, hence it meets the real-time requirements for saliency detection by a large margin.

Deep learning classification experiment and analysis

Comparison of transfer learning and model training

In this section, we presented flame image classification results using multiple lightweight deep learning models. For each type of model, we select the model with the best accuracy on the validation dataset for further performance testing on the test dataset. The two groups of results exhibited Figure 7, corresponding to training from scratch represented by blue dots, the

![Recognition effects of different visual saliency detection techniques.](image)
Figure 7. Accuracy of different models on validation datasets.

Model-based deep transfer learning represented by red dots. The vertical axis represents different initial training learning rates, and the horizontal axis represents validation accuracy. If there is no result at a certain learning rate, it means that training based on this learning rate will cause models to converge too slowly or not at all. From the analysis of Figure 7, it can be concluded that: (1) there is little difference in validation accuracy between different models that are trained in our fire dataset; (2) fine-tuning a pre-trained model is more effective than training from scratch in our dataset classification experiments. All the comparison results show the effectiveness of transfer learning to our fire image tasks.

**Model selection**

To get the best performance model, we selected models with the best performance on the validation set and presented the test results for those models (Figure 8). The blue line indicates the inference speed of models and the red column indicates model accuracy. The left vertical axis represents model type and the right vertical axis represents model size.

It can be clearly found that the lowest accuracy happens in MobileNetV3-Small (98.32%), and the best accuracy happens in MobileNetV2-1.0 (99.34%) (Figure 8). However, the inference speed of MobileNet series models is much slower than other models in our GPU environment. Because it is mainly designed for mobile terminals (Sandler et al., 2018). Since MobileNet models have completely different performance on specific platforms or operating systems, we give up using the MobileNet series model in the deployed environment due to model stability considerations. Overall, ResNet18 also has excellent performance in model size and model inference speed, and its inference time is only 5.690 ms.
Moreover, to further study the performance and the generalization ability of these models, we measure the precision, recall, accuracy score and F1 score of the models. The calculation formula is as follows:

\[
\begin{align*}
\text{Precision} &= \frac{TP}{TP + FP} \\
\text{Recall} &= \frac{TP}{TP + FN} \\
F_1 &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\end{align*}
\]

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

Figure 9 shows the precision, recall and F-score of the results. Wildfires typically rarely occur in forest environments, but once they occur, they can be very serious. Consequently, we hope that the classification model has ideal accuracy and a high false positive rate. The accuracy, recall and F1 score of the ResNet18 model all reach 99.28%, proving that it can more effectively balance the detection accuracy and false positive rate compared to other models in our tasks. Combined, we believe ResNet18 is the most suitable model for actual deployment in our tasks.

**Deep learning C++ deployment**

Finally, we deployed deep learning models through NCNN deployment framework and C++ language, and the average inference speed of these models on the NCNN deployment framework is listed in Table 2.

Table 2 displays the actual performance of model deployment through NCNN. Simply from Table 2 and Figure 8, model inference using the NCNN C++ framework is generally faster than using Python on GPU and NCNN also achieves good results in single/multi-threaded CPU inference models. The single inference time of the ResNet18 model under NCNN/GPU was 4.912 ms.

Our model format needs to be converted to the NCNN-specific model format before it can be deployed into the NCNN framework. This process led to a decrease in accuracy, but model accuracy dipped by less than 0.2% and the accuracy of the ResNet18 has no change after model conversion.

**Overall test**

We combined all algorithms to form a complete fire detection system and deployed it for real-time C++ applications. We conducted experiments on a real-world video dataset to test all aspects of our algorithms. During the system performance test process, the first sample 100 continuous frames from a video to background modelling and then sub-sample one frame to detection for every 8 frames.

As shown in Table 3, for each video, we give the detection times of the earliest triggered alarm and report the fire alarm times and detection times. The detection times of the earliest triggered alarm refer to how much detection are required to trigger an alarm after a fire occurs. The reason for setting up this evaluation index is to assess the sensitivity of our algorithm. The number of detection represents the number of times the flame detection algorithm is...
executed on the current video. The number of fire alarms indicate the number of times the flame alarm is triggered for the current video.

We tested a total of 27 videos, 11 with flames and 16 without flames. For 16 non-fire videos, there were only 15 false alarms in 1,329 detection, with a false positive rate of 1.12% (Table 3). The detection results were also very satisfying in fire videos with different resolutions, and all flame objects were well detected. Our algorithm does not always detect flame objects every time, this is because flame objects do not appear in every frame of fire video. Overall, our fire detection system not only ensures detection sensitivity, but also greatly reduced false positives in non-fire videos. These experiment results reveal that our system was robust in various scenarios and has extremely practical value.

Conclusions

We presented an intelligent fire detection approach through cameras based on computer vision methods and deep learning technology, and the deep learning model was successfully deployed based on C++ programming language. First, videos from surveillance cameras were processed frame by frame through a background subtraction method to extract moving
region images. Following an area containing moving objects, we used the VSD algorithm in this area to further extract significant regions. A deep learning network was then used to determine whether these salient regions contain flame objects. Compared with other image classification algorithms, our approach mainly has the following improvements.

1. To reduce the fire false alarm, we added a motion detection method based on background subtraction. When moving objects appear, the VSD algorithm will generate three significant region sub-images from a moving area image.

2. We used large amounts of image data to fine-tune our deep learning network and determine whether the fire was detected in current detection combined with the classification results of the three significant images. This process improved model training efficiency and detection accuracy.

3. An algorithm engineering deployment scheme based on C++ language and NCNN deep learning inference framework was proposed, which has good application effects.

The proposed method was developed for practical application in forests and other high-fire risk scenarios. Our approach can meet the accuracy and speed requirements of real-time fire detection.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

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) and Special Project of “Lushan Plants” (2022ZWZX02);Priority Academic Program Development of Jiangsu Higher Education Institutions;

**References**

Allison, R. S., Johnston, J. M., Craig, G., & Jennings, S. (2016). Airborne optical and thermal remote sensing for wildfire detection and monitoring. Sensors, 16(8), 1310. https://doi.org/10.3390/s16081310

Allison, R. S., Johnston, J. M., & Wooster, M. J. (2021). Sensors for fire and smoke monitoring. Sensors, 21(16), 5402. https://doi.org/10.3390/s21165402

Antioquia, A. M. C., Stanley Tan, D., Azcarraga, A., & Hua, K.-L. (2019). Single-fusion detector: towards faster multi-scale object detection. 2019 IEEE International Conference on Image Processing (ICIP), 76–80. https://doi.org/10.1109/ICIP.2019.8802913

Barmpoutis, P., Dimitropoulos, K., & Grammalidis, N. (2014). Smoke detection using spatio-temporal analysis, motion modeling and dynamic texture recognition. 2014 22nd European Signal Processing Conference (EUSIPCO), 1078–1082.

Barmpoutis, P., Papaioannou, P., Dimitropoulos, K., & Grammalidis, N. (2020). A review on early forest fire detection systems using optical remote sensing. Sensors, 20(22), 6442. https://doi.org/10.3390/s20226442

Barnich, O., & Droogenbroeck, M. V. (2011). ViBe: A universal background subtraction algorithm for video sequences. IEEE Transactions on Image Processing, 20(6), 1709–1724. https://doi.org/10.1109/TIP.2010.2101613
Berie, H. T., & Burud, I. (2018). Application of unmanned aerial vehicles in earth resources monitoring: Focus on evaluating potentials for forest monitoring in Ethiopia. European Journal of Remote Sensing, 51(1), 326–335. https://doi.org/10.1080/22797254.2018.1432993

Burnett, J. D., & Wing, M. G. (2018). A low-cost near-infrared digital camera for fire detection and monitoring. International Journal of Remote Sensing, 39(3), 741–753. https://doi.org/10.1080/01431161.2017.1385109

Chang, C. C., & Lin, C. J. (2011). LIBSVM: A library for support vector machines, 2(3), 1–27. https://doi.org/10.1145/1961189.1961199

Cheng, M., Zhang, Z., Lin, W., & Torr, P. (2014). BING: binarized normed gradients for objectness estimation at 300fps. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 3286–3293. https://doi.org/10.1109/CVPR.2014.414

Chen, J., He, Y., & Wang, J. (2010). Multi-feature fusion based fast video flame detection. Building and Environment, 45(5), 1113–1122. https://doi.org/10.1016/j.buildenv.2009.10.017

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., & Zhang, Z. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems (arXiv:1512.01274). https://arxiv.org/10.1109/TCSVT.2014.2339592

Cong, R., Lei, J., Fu, H., Cheng, M., Lin, W., & Huang, Q. (2019). Review of visual saliency detection with comprehensive information. IEEE Transactions on Circuits and Systems for Video Technology, 29(10), 2941–2959. https://doi.org/10.1109/TCSVT.2018.2870832

Dimitropoulos, K., Barmpoutis, P., & Grammalidis, N. (2015). Spatio-temporal flame modeling and dynamic texture analysis for automatic video-based fire detection. IEEE Transactions on Circuits and Systems for Video Technology, 25(2), 339–351. https://doi.org/10.1109/TCSVT.2014.2339592

Gaur, A., Singh, A., Kumar, A., Kumar, A., & Kapoor, K. (2020). Video flame and smoke based fire detection algorithms: A literature review. Fire Technology, 56(5), 1943–1980. https://doi.org/10.1007/s10694-020-00986-y

Geetha, S., Abhishek, C. S., & Akshayanat, C. S. (2021). Machine vision based fire detection techniques: A survey. Fire Technology, 57(2), 591–623. https://doi.org/10.1007/s10694-020-01064-z

Godbehere, A. B., Matsukawa, A., & Goldberg, K. (2012). Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. 2012 American Control Conference (ACC), 4305–4312. https://doi.org/10.1109/ACC.2012.6315174

Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., Zhang, A., Zhang, H., Zhang, Z., Zhang, Z., Zheng, S., & Zhu, Y. (2020). GluonCV and GluonNLP: Deep learning in computer vision and natural language processing. ArXiv, 1907.04433 [cs, Stat]. http://arxiv.org/abs/1907.04433

Hinkley, E. A., & Zajkowski, T. (2011). USDA forest service–NASA: Unmanned aerial systems demonstrations – Pushing the leading edge in fire mapping. Geocarto International, 26(2), 103–111. https://doi.org/10.1080/10106049.2011.555823

Hosang, J., Benenson, R., Dollar, P., & Schiele, B. (2016). What makes for effective detection proposals? IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(4), 814–830. https://doi.org/10.1109/TPAMI.2015.2465908

Hou, X., & Zhang, L. (2007). Saliency detection: A spectral residual approach. 2007 IEEE Conference on Computer Vision and Pattern Recognition, 1–8. https://doi.org/10.1109/CVPR.2007.383267

Jadon, A., Omama, M., Varshney, A., Ansari, M. S., & Sharma, R. (2019). FireNet: A Specialized Lightweight Fire & Smoke Detection Model for Real-Time IoT Applications. https://doi.org/10.48550/arXiv.1905.11922

Jan, M. A., Nanda, P., He, X., & Liu, R. P. (2016). A Sybil attack detection scheme for a forest wildfire monitoring application. Future Generation Computer Systems, 80, 613–626. https://doi.org/10.1016/j.future.2016.05.034

Karbowiak, L., & Bobulski, J. (2022). Background segmentation in difficult weather conditions. PeerJ Computer Science, 8, e962. https://doi.org/10.7717/peerj-cs.962

Kim, B., & Lee, J. (2019). A video-based fire detection using deep learning models. Applied Sciences, 9(14), 2862. https://doi.org/10.3390/app9142862

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization (arXiv:1412.6980). https://arxiv.org/10.1109/TCSVT.2014.2339592

Lin, G., Zhang, Y., Zhang, Q., Jia, Y., Xu, G., & Wang, J. (2017). Smoke detection in video sequences based on dynamic texture using volume local binary patterns. KSII Transactions on Internet and Information Systems (TIIS), 11(11), 5522–5536. https://doi.org/10.3837/tiis.2017.11.019

Li, J., Yan, B., Zhang, M., Zhang, J., Jin, B., Wang, Y., & Wang, D. (2019). Long-range raman distributed fiber temperature sensor with early warning model for fire detection and prevention. IEEE Sensors Journal, 19(10), 3711–3717. https://doi.org/10.1109/JSEN.2019.2895735

Li, P., & Zhao, W. (2020). Image fire detection algorithms based on convolutional neural networks. Case Studies in Thermal Engineering, 19, 100625. https://doi.org/10.1016/j.cset.2020.100625

Li, J., Zhou, G., Chen, A., Wang, Y., Jiang, J., Hu, Y., & Lu, C. (2022). Adaptive linear feature-reuse network for rapid forest fire smoke detection model. Ecological Informatics, 68, 101584. https://doi.org/10.1016/j.ecoi.2022.101584

Machado, P., Oikonomou, A., Ferreira, J. F., & Mcginnity, T. M. (2021). HSMD: An object motion detection algorithm using a hybrid spiking neural network architecture. IEEE Access, 9, 125258–125268. https://doi.org/10.1109/ACCESS.2021.3111005

Marowka, A. (2018). Python accelerators for high-performance computing. The Journal of Supercomputing, 74(4), 1449–1460. https://doi.org/10.1007/s11227-017-2213-5

Merino, L., Caballero, F., Martínez-de-Dios, J. R., Maza, I., & Ollero, A. (2012). An unmanned aircraft system for automatic forest fire monitoring and measurement. Journal of Intelligent & Robotic Systems, 65(1), 533–548. https://doi.org/10.1007/s10846-011-9560-x

Montabone, S., & Soto, A. (2010). Human detection using a mobile platform and novel features derived from a visual saliency mechanism. Image & Vision Computing, 28(3), 391–402. https://doi.org/10.1016/j.imva.2009.06.006

Nguyen, M. D., Vu, H. N., Pham, D. C., Choi, B., & Ro, S. (2021). Multistage real-time fire detection using convolutional neural networks and long short-term memory networks. IEEE Access, 9, 146667–146679. https://doi.org/10.1109/ACCESS.2021.3122346

NVIDIA/TensorRT. (2021). [C++]. NVIDIA Corporation. https://github.com/NVIDIA/TensorRT (Original work published 2019)
OpenVINO. (2021). [C++]. OpenVINO Toolkit. https://github.com/openvino-toolkit/openvino

Shahid, M., Virtusio, J. J., Wu, Y.-H., Chen, Y.-Y., Tanveer, M., Muhammad, K., & Hua, K.-L. (2022). Spatio-temporal self-attention network for fire detection and segmentation in video surveillance. IEEE Access, 10, 1259–1275. https://doi.org/10.1109/ACCESS.2021.3132787

Shi, J., Wang, W., Gao, Y., & Yu, N. (2020). Optimal placement and intelligent smoke detection algorithm for wildfire-monitoring cameras. IEEE Access, 8, 72326–72339. https://doi.org/10.1109/ACCESS.2020.2987991

Tencent/ncnn. (2021). [C++]. Tencent. https://github.com/Tencent/ncnn

Wang, X., Wang, W. M., Cantin, A. S., Parisien, M. A., Anderson, K., Moore, B., & Flannigan, M. D. (2017). cfdr: An R package for the Canadian forest fire danger rating system. Ecological Processes, 6(1), 5. https://doi.org/10.1186/s13717-017-0070-z

Wang, R., Xu, J., & Han, T. X. (2019). Object instance detection with pruned AlexNet and extended training data. Signal Processing: Image Communication, 70, 145–156. https://doi.org/10.1016/j.image.2018.09.013

Wang, P., Zhang, J., & Zhu, H. (2021). Fire detection in video surveillance using superpixel-based region proposal and ESE-ShuffleNet. Multimedia Tools and Applications, 1–28. https://doi.org/10.1007/s11402-021-11261-9

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big Data, 3(1), 9. https://doi.org/10.1186/s40537-016-0043-6

Wu, R., Guo, X., Du, J., & Li, J. (2021). Accelerating neural network inference on FPGA-based platforms—A survey. Electronics, 10(9), 1025. https://doi.org/10.3390/electronics10091025

Xie, Y., Zhu, J., Cao, Y., Zhang, Y., Feng, D., Zhang, Y., & Chen, M. (2020). Efficient video fire detection exploiting motion-flicker-based dynamic features and deep static features. IEEE Access, 8, 81904–81917. https://doi.org/10.1109/ACCESS.2020.2991338

Zeevi, S. (2020). Sagi-z/BackgroundSubtractorCNT [C++]. Accessed December 2020. https://github.com/sagi-z/BackgroundSubtractorCNT (Original work published 2016)

Zhang, J., Li, W., Ogunbona, P., & Xu, D. (2019). Recent advances in transfer learning for cross-dataset visual recognition: A problem-oriented perspective. ACM Computing Surveys, 52(1), 7: 1–7:38. https://doi.org/10.1145/3291124

Zhang, J., Zhu, H., Wang, P., & Ling, X. (2021). ATT squeeze U-Net: A lightweight network for forest fire detection and recognition. IEEE Access, 9, 10858–10870. https://doi.org/10.1109/ACCESS.2021.3050628

Zhao, L., Zhi, L., Zhao, C., & Zheng, W. (2022). Fire-YOLO: A small target object detection method for fire inspection. Sustainability, 14(9), 4930. https://doi.org/10.3390/su14094930

Zivkovic, Z., & Heijden, F. V. D. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recognition Letters, 27(7), 773–780. https://doi.org/10.1016/j.patrec.2005.11.005