IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 2, 2020, accepted February 25, 2020, date of publication March 3, 2020, date of current version March 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2978090

Cloud Shape Classification System Based on Multi-Channel CNN and Improved FDM

MENGYANG ZHAO[1], CHORNG HWA CHANG[1], WENBIN XIE[2,3], ZHOU XIE[3], AND JINYONG HU[4]

[1]Electrical and Computer Engineering Department, Tufts University, Medford, MA 02155, USA
[2]Hunan Meteorological Station, China Meteorological Administration, Changsha 410000, China
[3]Yipai Weiye Co., Ltd., Beijing 100190, China
[4]Computer Science Department, Tufts University, Medford, MA 02155, USA

Corresponding author: Mengyang Zhao (Mengyang.Zhao@tufts.edu)

This work was supported in part by the Tufts University, China Meteorological Administration and Yipai Weiye Co., Ltd.

ABSTRACT This paper presents a new meteorological photo classification system based on the Multi-channel Convolutional Neural Network (CNN) and improved Frame Difference Method (FDM). This system can work in an embedded system with limited computational resources and categorize cloud observation photos captured by ground cameras. We propose the improved FDM extractor to detect and extract cloud-like objects from large photos into small images. Then, these small images are sent to a Multi-channel CNN image classifier. We construct the classifier and train it on the photo-set that we established. After combining the extractor and classifier to form the classification system, the images can be classified into three different types of clouds, namely, cumulus, cirrus and stratus, based on their meteorological features. The testing phase uses 200 actual photos of real scenes as the experimental data. The results show that the classification accuracy can reach 94%, which indicates that the system has a competitive classification ability. Moreover, the time cost and computational resource consumption for image recognition are greatly reduced. By using this system, meteorologists can lighten their workload of processing meteorological data.

INDEX TERMS Image recognition, FDM, CNN, edge computing, meteorological observation.

I. INTRODUCTION

Weather is relevant to everyone's daily life. Cloud-type classification is a very crucial tool in weather prediction, especially in aviation forecasting and raindrop prediction [3], [29]. Meteorologists need to get accurate classification information in a wide range of areas to predict the weather. The primary method to obtain this information is to classify observation photos. However, classifying such large numbers of photos is a burdensome job, even to experienced meteorologists. Therefore, the manual processing speed of cloud classification is always very slow. However, using computer-aided processing and AI technology can significantly accelerate the processing speed.

According to meteorological classification criteria [2], clouds can be classified based on cloud height and cloud shape. For the height classification process, measurement technology has greatly matured. Laser ranging technology has been able to control the error within 1% [4]. However, the technology for cloud shape recognition is still quite immature. Based on the shape [2], clouds can be divided into three large categories: Cumulus, Cirrus and Stratus [1], [2]. Figure 1 shows the typical shapes of these three kinds of clouds. It is very difficult to distinguish between their shape using traditional image processing methods. But with the development of AI technology, it is possible to use this technology for cloud shape classification. However, the accuracy of the traditional AI algorithm is not ideal. For example, the KNN algorithm in paper [5] and the SVM algorithm in paper [7], have general error rates of more than 15%. If using the current popular deep learning model, such as

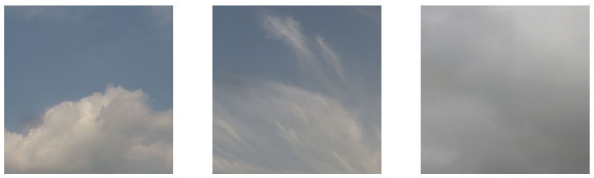


FIGURE 1. Typical shapes of three clouds.

The associate editor coordinating the review of this manuscript and approving it for publication was Mouloud Denai.

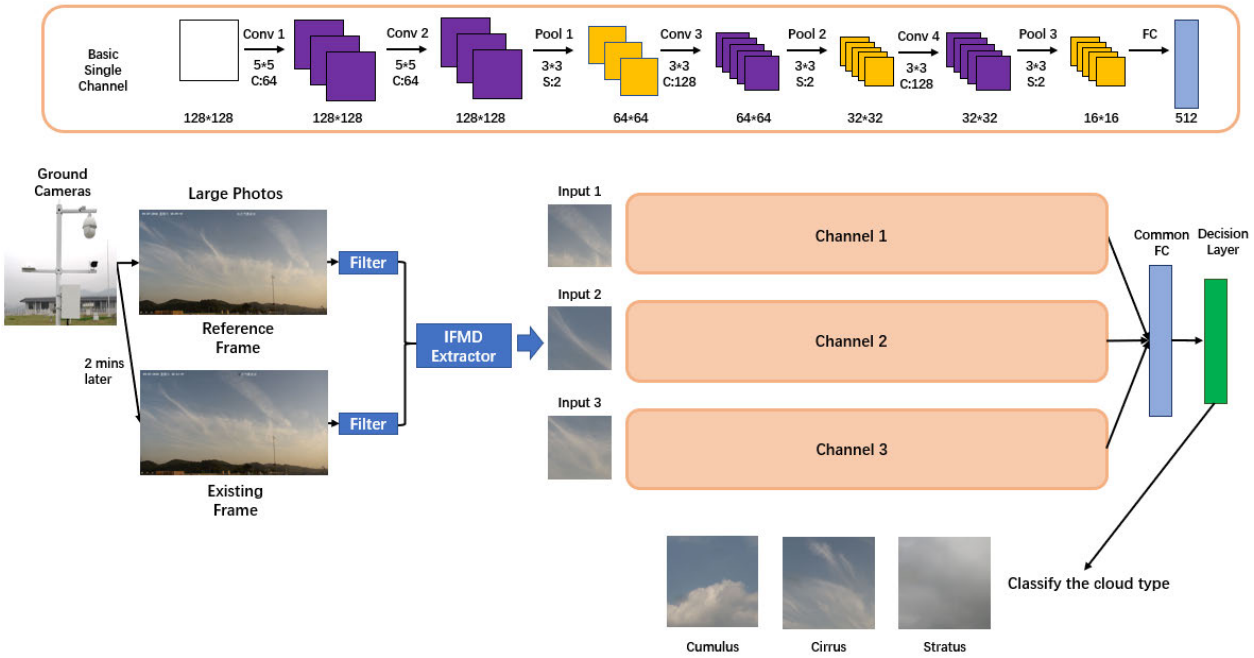


FIGURE 2. Brief System Diagram. The whole system is composed of the IFDM extractor and the Multi-channel CNN classifier. The details of the CNN classifier's network architecture are shown in Table 3.

VGG-16 [24] or Faster R-CNN [25], the consumption of computational resources could be very high [6].

In this paper, we implement an accurate cloud classification system that can work in an embedded system with limited computational resources. It processes the observation photos captured by meteorological cameras. These photos are very large and contain a large number of clouds. Before the clouds are extracted and classified, we call them cloud-like objects. Our system can extract cloud-like objects from photos and categorize them based on meteorological standards [2]. The whole system can be divided into two main parts: the Improved FDM(IFDM) extractor and the Multi-channel CNN classifier. A brief system diagram is shown in Figure 2.

The IFDM extractor uses improved FDM [8] to extract cloud-like objects from large photos. The camera takes pictures of the same region of the sky every two minutes, which will produce a large photo with 3840∗2160 pixels. This large photo will contain many clusters of clouds and other background objects. The extractor will try to extract cloud-like objects from these large photos. The traditional FDM [8] extractor relies on the difference between two frames to detect moving clouds from the static sky background. These differences are marked as the ''target block''. Normally, the extractor can extract moving objects based on these target blocks with very low computational cost. However, the slow-moving speed of typical clouds causes the differences between the two frames to be very slight. Therefore, many interfering objects are also be extracted. We proposed the object judgment and location control algorithm to select the target blocks and avoid interference objects. Through the above method, the extractor can accurately extract cloud-like objects and send them to the classifier.

The Multi-channel CNN classifier uses a spatial multi-channel CNN image classification model [10], [14]. Each channel can process a small image from different parts of the large photo and extract the image's features. The channel is composed of a nine layers CNN model and is fused to a common fully connected layer. At the top of the whole model, one output layer will output the classification result. In the training step, the classification model is trained by the observation photo-set established by us. The cross entropy between the idle output and the model output is used as the loss function of the model. We use the Adam optimizer [9] to optimize the parameters in the model to reduce the loss of the output. After training, the classifier can classify small images into three categories: Cumulus, Cirrus and Stratus. Finally, we combine the IFDM extractor with the Multi-channel CNN classifier to form the classification system. The testing results show the classification accuracy of it can reach 94%, which indicates that the whole classification system has a good classification ability. Moreover, this system performs well in a resource-limited computing environment.

Our contributions can be summarized as follows:

(1) We propose a new size judgment and location control algorithm to improve the FDM [8]. By using this algorithm, our extractor can detect and extract cloud-like objects from large meteorological photos with less resource consumption.

(2) We construct a spatial Multi-channel CNN [10] classifier and use it to classify the image with the cloud-like object. The model achieves a competitive classification accuracy.

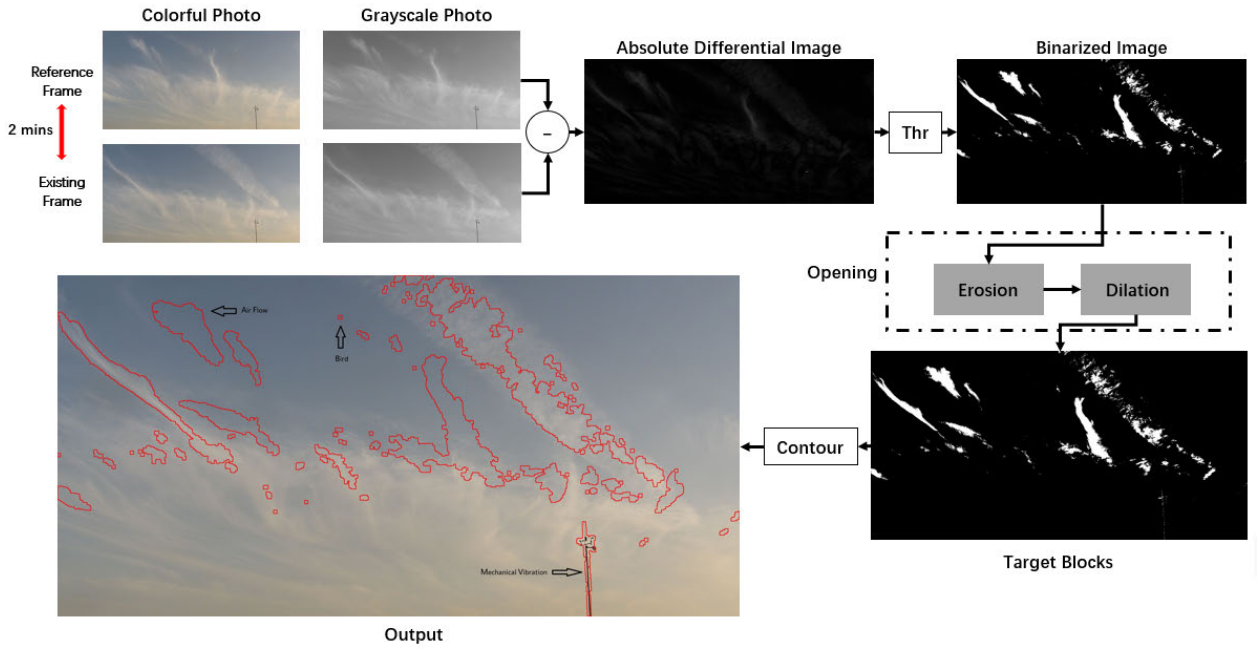


FIGURE 3. Flow Chart of FDM. The algorithm uses the differences between two frames to detect moving objects. In the figure, the red contour is the extracted objects' outline. It extracts a lot of interference blocks.

(3) We build a cloud classification system based on the above two methods on a Jetson TX2 [20] embedded platform. This system can accurately extract and classify cloud-like objects in a hardware environment with limited computational resources.

(4) We establish an image set that contains observation photos from various locations and times. The photos were manually classified by meteorologists from the China Meteorological Administration.

In the next few sections, we will give a detailed introduction to the whole system. In Section 2, we will discuss the related background of our research. Then, in Section 3, we will introduce the new methods proposed by us. In Section 4, experiments on training and testing the classifier are presented. In Section 5, we will analyze the experiment result and system performance. Finally, Section 6 provides a brief conclusion.

II. RELATED WORK

Our research is mostly based on the Frame Difference Method [8] and Convolutional Neural Network [10], [14]. We will discuss these methods and analyze their limitations.

A. FRAME DIFFERENCE METHOD

1) THEORY BEHIND THE FDM

The objective of the FDM [8] approach is to detect moving objects from the difference between the existing frame and the reference frame. This method adopts pixel differences to extract the contours of moving objects, which is a common method for motion detection. It contains four major steps: Transformation, Difference, Binarizing and Opening. Figure 3 shows the steps of processing a meteorological photo with the traditional FDM [8].

a: TRANSFORMATION TO GRAYSCALE PICTURE

To obtain the absolute differential image, the color image needs to be transformed to a grayscale image. Suppose P_i is the ith pixel of a frame, and CH represents different color channels of that pixel. Then the transformation from the color picture to a grayscale picture can be expressed as (1):

$$P_i = 0.299 \cdot CH_{red} + 0.587CH_{green} + 0.114 \cdot CH_{blue} \quad (1)$$

b: DIFFERENCE OF TWO CONSECUTIVE FRAMES

Let F_k be the k frames in an image sequence. Then, F_{k+1} is the $k+1$ frames in this sequence. F_k is called the referenced frame, and F_{k+1} is called the existing frame. Therefore, the absolute differential image is defined in (2) as follows:

$$F_{d(k,k+1)} = |F_k - F_{k+1}| \quad (2)$$

c: BINARIZING BY THRESHOLD

Next, we can binarize the grayscale picture by using a threshold. Normally, we set the threshold to 20/255. Then, the binarization can be expressed as (3):

$$P_{i\_b} = Thr_{\frac{20}{255}}(P_{i\_gray}) \quad (3)$$

After these three steps, the difference between two frames will be transformed to some nonzero blocks of different sizes.

d: OPENING OPERATION

Opening serves as the basic workhorse for morphological noise removal in image processing. It can be divided

into two fundamental operations: erosion and dilation [11]. Through these two operations, the small nonzero block will be removed, and the large nonzero block will expand. The remaining nonzero block is called the ''target block''. According to the above steps, the nonzero block is the difference between the two frames. Therefore, ideally, the contour of the target blocks should be the contour of the moving objects. So, the contours of the moving objects are finally extracted by the FDM.

2) LIMITATION OF THE FDM

Although the FDM provides a fast and effective method for moving object detection with low computational resources, it is unable to handle objects moving at a slow speed [8]. However, most cloud-like objects move very slowly. Therefore, the result is that the FDM will eventually extract the high-speed moving interfering objects, such as birds, airflow and mechanical vibrations, as target blocks from the image. These interfering blocks will eventually result in a sharp decline in the accuracy of the extractor and the whole system.

The result of Figure 3 shows a very good example of the effect of using the traditional FDM to extract a cloud-like object. The red contour is the object extracted by this algorithm. As the Figure 3 shows, although the FDM successfully extracts cloud-like objects, a large number of interfering blocks, as mentioned above, have also been extracted at the same time. There, the result shows many extracted objects in the processed picture. So finding an improvement in this algorithm to overcome these limitations is indispensable. We will introduce improved method in a later section.

B. DEEP LEARNING AND MULTI-CHANNEL CNN

1) DEVELOPMENT OF DEEP LEARNING AND CNNS

Deep learning can be traced back to 1989 when LeCun applied the BP algorithm to a multilayer neural network [12]. LeNet-5 [13], a basic CNN model, was proposed by LeCun in 1998. It uses convolutional layers to extract image features. The accuracy of using it to recognize a specific image dataset is much higher than that of the traditional machine learning method. However, this model does not have enough layers. Therefore, its performance in practical problems is not good. If the number of network layers is increased, the gradient disappearing problem during the training step will also cause poor performance. Then, in 2012, Alex proposed the AlexNet [14] architecture. The architecture adopted the ReLU activation function instead of the traditional Sigmod function, which can successfully avoid the problem of gradient disappeared in the large neural network training. The performance using AlexNet to recognize images is much better than traditional methods. However, if the image contains some interfering objects, the CNN's recognition rate will still be poor. In 2013, the R-CNN [15] was proposed by KR Girshick et.al. It uses a region selection algorithm to extract target objects from large photos for recognition. Although it will consume many computational resources, it can greatly reduce the impact of interference on recognition and greatly improve the recognition accuracy. With these research findings from recent years, using a CNN become the best choice for extracting features and recognizing images.

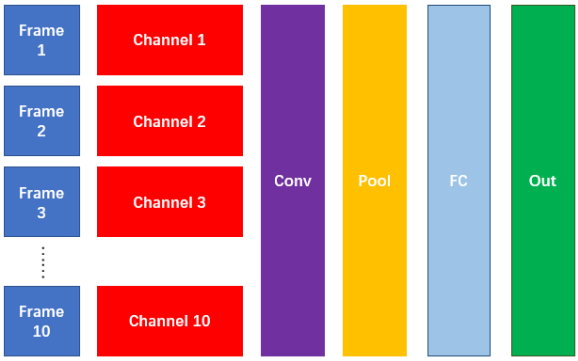


FIGURE 4. Time-domain Multi-channel CNN.

2) MULTI-CHANNEL CNN

Most traditional CNN architectures are designed to recognize static pictures. The use of CNNs in video recognition tasks has received some attention in recent years. Using a multi-channel CNN architecture to process videos can achieve a good performance. J Yue-Hei Ng et al. proposed using a multi-channel CNN for video classification [16]. They built a multi-channel architecture and used a time-domain convolutional layer to extract the time-domain feature of the video. The whole network can be simplified and is shown in Figure 4.

The network contains an extra time-domain convolutional layer before feature pooling across frames. The convolutional layer consists of 256 kernels of size 3×3 across 10 frames with a frame stride of 5. This model aims to capture the local relationships between frames. The advantage of this network is that the time-domain features, such as moving objects and frame differences, can be extracted from the video and recognized. Therefore, it achieves a better performance than the traditional CNN for video recognition. However, these algorithms require large numbers of layers to achieve a good performance, which requires plenty of computational resources. Therefore, it cannot work well in an embedded system.

3) OBJECT RECOGNITION

Inspired by a spatial multi-channel CNN model [10]. We assign each channel to a specific region in a frame. However, determining the location of that region is an important problem. In recent research, many algorithms have focused on object recognition and region selection. Among them, Faster R-CNN [15], [25] provides a good method. It uses Region Proposal Networks to generate accurate proposal regions of the target objects. And it adopts Regions of Interest (ROI) pooling to select the proposal regions. It has been

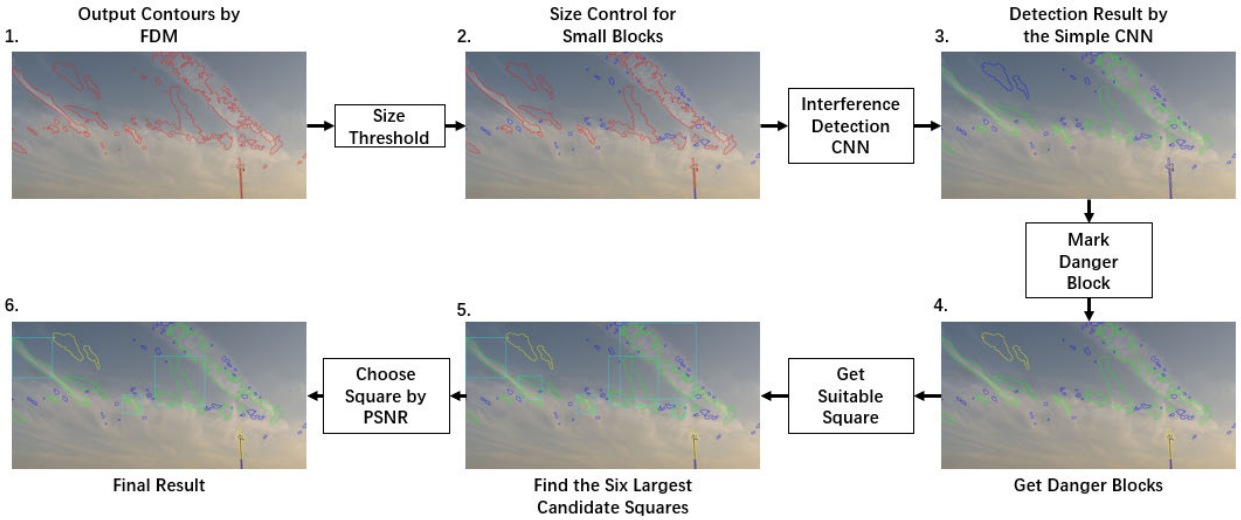


FIGURE 5. Flow Chart of the IFDM Extractor. Red contour: blocks that await judgment; blue contour: interfering blocks; yellow contour: dangerous blocks; light-blue squares: candidate or result.

widely used in object recognition tasks, such as those in medical research [30] and industry application [31].

However, R-CNNs also have thier own weaknesses. Therefore, for different application scenarios, corresponding improvements and optimization are indispensable. For example, in remote sensing and satellite images, the objects are always very small and have a very blurred boundary. A special augmentation network [33] for object detection or a judgment algorithm [32] for extraction is critical. Moreover, in medical image object detection, the object often has common characteristics: they are similar in size and shape. Li *et al.* [30] proposed using the ANCF algorithm for domain adaptation and a BN-IN Net to improve network stability.

Ideally, we can use Faster R-CNN [25] to make the region selection. However, in our system, the network needs to work in a hardware environment with limited computational resources. Therefore, we have to improve the FDM algorithm so that it only needs few resources to determine the locations of the cloud objects.

III. PROPOSED METHOD

In this section, we will present our proposed methods. First, we will introduce the improvement in the IFDM. Then we will present our network architecture of Multi-channel CNN model.

A. IMPROVED FDM

The IFDM extractor is mainly improved by two parts: object judgment and location control. The object judgment step determines whether a target block is a cloud-like object. The location control step selects the three best cloud-like objects and finds boundary boxes to contain them. The system flow chart is shown in Figure 5.

1) OBJECT JUDGMENT

The FDM provides a method to detect the contours of cloud-like objects with low computational resource but many interfering objects might also be extracted (see step 1 in Figure 5). Therefore, our goal is to find a method to distinguish between cloud-like objects and interfering blocks. To achieve this goal, we first analyze the object extracted by the FDM. We then use the FDM to extract 1189 blocks from 50 frames(25 pairs of existing and reference frames). We found more than 80% of the blocks are interfering blocks. We also found that the size of the cloud-like objects is very different from the size of the interfering blocks. Therefore, we calculate statistics on the size distribution of the block. The results are shown in Table 1.

TABLE 1. Size distribution of the blocks. The numbers in parentheses are the numbers of the blocks that are too small or ambiguous to be judged by people.

Size(pixels)	Cloud-like Objects	Interfering Blocks	**Total**
<1000	0	572(479)	572
1000-4000	0	276(144)	276
4000-9000	98	79	177
>9000	118	46	164
Total	216	973	1189

In most cases, the size of the interfering objects is very small. Therefore, if we adopt a suitable size threshold of approximately 4000 pixels, most of the interference will be filtered. The classification system will save considerable computing resources to judge these small interfering blocks (see step 2 in Figure 5). In the remaining blocks, there are still some large interfering blocks. Inspired by the selective search [17] and R-CNN [15] algorithms, we decided to use a CNN classifier to distinguish large interfering blocks. Because large interfering blocks are usually caused by mechanical vibrations or airflow, their appearance is very different from that of clouds. Therefore, only using a simple CNN classifier

TABLE 2. Simple CNN classifier architecture for interference detection. 'C' is the number of convolution kernels. 'S' is the stride.

Layer	Configuration
Input Layer	64*64*3
Conv1	5*5 C: 20
Pool1	2*2 S: 2
Conv2	3*3 C: 40
Pool2	2*2 S: 2
FC	256
Output Layer	2

can distinguish them. We build a simple CNN similar to LeNet-5 [13] to recognize interfering and cloud-like objects. The architecture of this CNN network is shown in Table 2. It is a very simple model and consumes very few computational resources. Additionally, because we set the size threshold, most of the interference does not need to be distinguished through this method. Therefore, compared with the front-end region judgment in the R-CNN [15], this object judgment method only needs a small number of resources. Although its versatility is not so strong, it is enough to distinguish cloud-like objects (see step 3 in Figure 5).

Through the combination of the size threshold and simple CNN classifier, the object judgment procedure can accurately distinguish cloud-like object from interfering objects. The blocks with type information will be sent to the location control.

2) LOCATION CONTROL ALGORITHM

The shapes of the cloud-like object vary. However, the input layer of Multi-channel CNN is fixed(128∗128). Therefore, we need to find a suitable square boundary box to contain the cloud-like objects. Moreover, in this boundary box, we do not want to include interfering blocks that impact the classification. Large interfering blocks will greatly affect the recognition of cloud-like objects. Many small interfering blocks may also affect the subsequent classification. Therefore, the core idea of this algorithm is to find a suitable box to avoid large interfering blocks and retain as few small interfering blocks as possible. The algorithm is shown in Algorithm 1.

The method of avoiding large-block interference is relatively simple. First, it marks an interfering block with more than 9000 pixels as a ''dangerous block'' because this size could be as large as a cloud-like object (see step 4 in Figure 5). Since we want to extract three cloud-like objects, it marks the 6 largest cloud-like objects as ''candidate objects''. The reason that it keeps 6 objects is because these candidate objects might be discarded in the next step. Of course, it can keep more candidates, but that will also consume more computational resources. In the experiment, we found that keeping six candidates in this step can ensure that it generates three cloud-like objects in the final results. Therefore, we chose only six candidate objects. Then, it creates the minimum enclosing rectangle for each candidate object. If there is a dangerous block in the rectangle, it will change the boundary until there is no dangerous area. Then, the inscribed square of the rectangle will become a suitable boundary. If the area of this square

Algorithm 1 Location Control Algorithm

Input: *Cloud_Set*: The contour set of cloud-like objects; *Interf_Set*: The contour set of interfering objects; *Photos*: The large meteorological photo
Output: *Best_Images*: The three best images;

```
1: //Mark the "Dangerous_Block"
2: for (Object.Size > 9000) ∈ Interf_Set do
3:     Mark Object as Dangerous_Block
4: end for
5: //Find out the best squares by area size
6: for Largest six Object ∈ Cloud_Set do
7:     Obtain the enclosing rectangle Rect of Object;
8:     if Rect ∩ Dangerous_Block then
9:         Move Rect.Boundary;
10:    end if
11:    Obtain the inscribed square Square of Rect;
12:    if Square.Size < 0.5*Rect.Size then
13:        Discard Square
14:    else
15:        Append Square to Square_Set
16:    end if
17: end for
18: //Calculate the PSNR of the square's region
19: for Square.Size ∈ Square_Set do
20:    Calculate the PSNR of Photos in the Square region
21: end for
22: //Sort the squares by the PSNR
23: Sort Square_Set by Square.PSNR
24: if Square_Set.Length < 3 then
25:    Copy largest Square in Square_Set
26: end if
27: Split Photos by Square_Set send to Best_Images
```

is reduced by more than 50% compared to the rectangle, we will discard the candidate object. Generally, six candidate objects are sufficient to meet the requirements. Therefore, we will obtain a square set of candidate objects. This part of the algorithm is shown in lines 1 to 17 in Algorithm 1. The results are shown in step 5 of Figure 5.

After finding out squares without a dangerous block, we will try to select three squares with the least ''amount'' of interference as the final output of the IFDM. However, how do we measure the ''amount'' of interference? Just use the number of interfering blocks is obviously not a good method. Some squares may contain many very small interfering blocks In this research, we adopt the Peak Signal-to-Noise Ratio (PSNR) as the measurement method [18]. The *PSNR* is the ratio between the maximum possible power of a signal and the power of corrupting noise. The PSNR is defined as in (4):

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE_I}\right) \tag{4}$$

Here, *MAX* is the maximum valid value for a pixel. In the original formula, *MSE* is the mean squared error between

TABLE 3. Multi-channel CNN architecture 'C' is the number of number of convolution kernels. 'S' is the stride.

Layer	Channel 1	Channel 2	Channel 3
Input Layer	128*128*3	128*128*3	128*128*3
Conv1	5*5 C: 64	5*5 C: 64	5*5 C: 64
Conv2	5*5 C: 64	5*5 C: 64	5*5 C: 64
Pool1	3*3 S: 2	3*3 S: 2	3*3 S: 2
Conv3	3*3 C: 128	3*3 C: 128	3*3 C: 128
Pool2	3*3 S: 2	3*3 S: 2	3*3 S: 2
Conv4	3*3 C: 128	3*3 C: 128	3*3 C: 128
Pool3	3*3 S: 2	3*3 S: 2	3*3 S: 2
Channel FC	512	512	512
Fusion FC	$(\lambda_1 \oplus 512) \oplus (\lambda_2 \oplus 512) \oplus (\lambda_3 \oplus 512)$		
FC1	1024		
Output Layer	3		

the original image and the image after transmission. In our algorithm, we define MSE_I as the mean squared difference between the pixel $P_{(x,y,c)}$ in the interfering block B and the square's average pixel value V_{Avg}. Then, MSE_I can be defined as in (5):

$$MSE_I = \frac{1}{i * j * c} \sum_{x,y,c \in B} (P_{(x,y,c)} - V_{Avg})^2 \tag{5}$$

Though the PSNR [13], we can measure the ''amount'' of interference. Then, we pick the three squares with the least ''amount'' of interference as the output. In a special scenario, if the first few steps discard too many candidate objects, two or three identical squares will be generated. Then, the IFDM extractor splits large photos into three small images by these square boundaries. The second part of the algorithm is shown in lines 18 to 27 in Algorithm 1. The results are shown in step 6 of Figure 5.

B. MULTI-CHANNEL CNN

We will introduce the network architecture and the channel weighted fusion.

1) NETWORK ARCHITECTURE

We constructed a Multi-channel CNN model to classify the images output by the IFDM extractor. The architecture of the network is shown in Table 3. It is composed of three channels, and each channel is a basic nine-layer convolutional neural network. The network architecture of each channel is very similar to that of AlexNet [14]. The first two convolutional layers adopt a 5 × 5 convolution kernel, and the last two convolutional layers adopt a 3 × 3 convolution kernel. The network uses maximum pooling for each pooling layer to downsample the feature images. The ReLU function is used as the activation function. A 512-node fully connected layer outputs the extracted feature vectors from each channel. Then, these vectors will fuse to a common fully connected layer with their channel's weight. Then, another 1024-node fully connected layer is follows. At the end of the model, a three-node fully connected layer will output the type of cloud.

When the model is running, it needs three images as input. In the training step, the three images are randomly selected from the same cloud type in the training set. The size of each input image will be recorded for weight fusion. Then, it will be resized to 128*128 and sent into one channel. The three images will be assigned randomly to different channels as input, and the channel's weight will adjust based on the input image's size. Then, these three images will go through the channel and the feature vector will be output at the end of the channel. After the feature vector is generated by the fully connected layer of each channel, the three feature vectors are weighted and fused. The fused large feature vector contains the features of these three images. In the end, it will cooperate with the last two fully connected layers and output the prediction vector. This prediction vector has three nodes, which represent different cloud types of large photos by one-hot encoding.

2) WEIGHT FUSION

We found in our experiment that, considering the input images of three channels, the different sizes of input can take different weights in the classification model. Ye, H. et al. proposed a method [18], and the weight fusion for this network is very simple. First, we built three nodes, and the input of these nodes is the size value of their corresponding images. We use ReLU as the activation function. The output of each channel's node will be the λ vector of this channel. Therefore, the λ vector is determined by the input image size of each channel. The λ_n vector can be expressed as (6):

$$\lambda_n = ReLU(Size_n) \tag{6}$$

We concatenate the each λ_n vector with their corresponding channel's feature vector to get the channel feature vector $feat_n$. In the end, concatenate the channel feature vectors ch_feat_n together and got the new feature vector. The feature fusion can be expressed as (7) and (8):

$$ch\_feat_n = \lambda_n \oplus feat_n \tag{7}$$

$$fused\_feat = ch\_feat_1 \oplus ch\_feat_2 \oplus ch\_feat_3 \tag{8}$$

where $\oplus$ represents the concatenate operation. The channel feature vector ch_feat_n is generated by λ_n concatenated with $feat_n$. The three channel feature vectors are concatenated to generate $fused\_feat$.

When we finish the fusion procedure, $fused\_feat$ is the common feature with the weight of these three images. We call it the fusion FC layer. Then, the next layer, FC1, will fully connect with this new fusion FC layer.

IV. EXPERIMENT

In this section, we present the experimental details. Preparation work, training procedure, and testing process will be introduced.

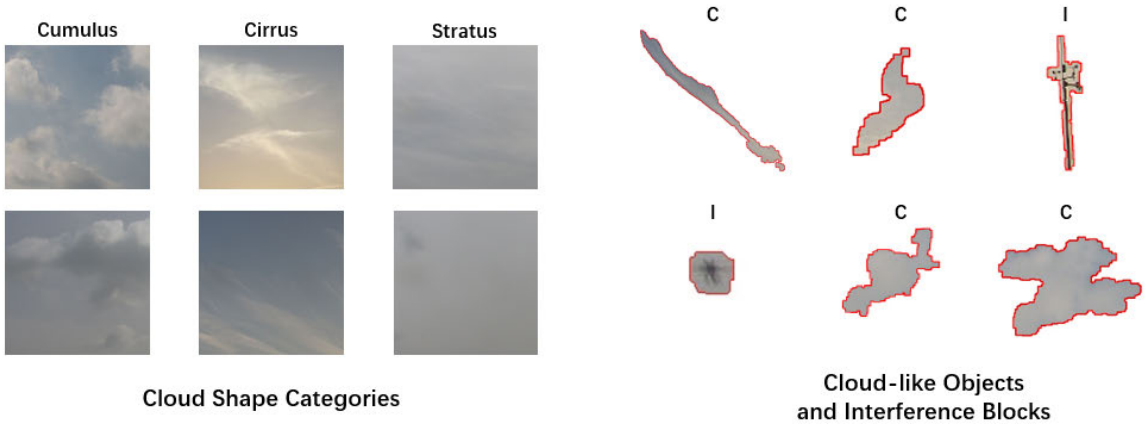


FIGURE 6. The Photoset Established by Us. The left side is the training set in Table 5, and the right side is the training set in Table 4.

A. PREPARATION WORK

1) SYSTEM IMPLEMENTATION

System implementation is mainly composed of the processing hardware and observation unit. A picture of the whole system is shown in Figure 7. The processing hardware is based on the NVIDIA Jetson TX2 platform [20], which has two CPUs: one is a Dual-Core NVIDIA Denver 2 64-Bit CPU, and the other is a Quad-Core ARM Cortex-A57 MPCore. The GPU is a 256-core NVIDIA Pascal GPU architecture with 256 NVIDIA CUDA cores. The board memory is 8 GB 128-bit LPDDR4 memory. The GPU memory and board memory are shared. We can see that it lacks the computational resources for a large-scale CNN architecture. In terms of software, we adopt Ubuntu 14.04 as the operating system. To save computational resources, most of our tests closed the GUI. For the hardware driver and GPU acceleration [26] driver, we installed JetPack 3.2 (CUDA9.0+CUDNN5). We use TensorFlow 1.9.0 and the Python programming language (version 3.6).

The observation unit mainly uses meteorological cameras to observe the sky. The spherical camera, which is on the top of the vertical pole, is mainly responsible for capturing cloud-shape photos. It can rotate to take photos in different directions. The camera rotates one full circle every two minutes, and 8 photos of different directions will be captured every 45 degrees. Then, the collected photos are sent back to the processing hardware through the Gigabit network. The average power consumption of the whole system is 100 W, and the peak power consumption is less than 150 W. Due to the low power consumption, solar cells can be used for the power supply. Therefore, it can be deployed to remote areas without electricity power supply.

2) IMAGES DATASET

The above section introduces the two different CNN models in our classification system. To train this model, we established a meteorological photoset. The photos are collected from the meteorological stations in Beijing, Changsha, and Hohhot at different times. All the cloud-type photos are classified by professional meteorological observers. Since the system does not need to work in poor weather conditions, we manually removed the images with poor image quality. Some of the images are shown in Figure 6.

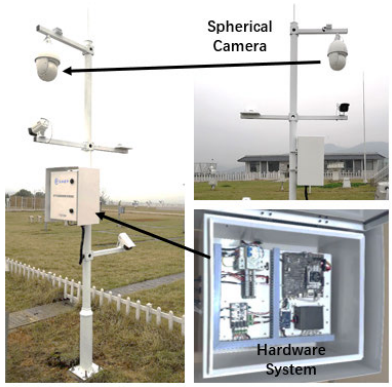


FIGURE 7. Our Observation and Classification System. This paper mainly uses the spherical camera and Jetson TX2 embedded platform.

The photo set mainly contains two parts. The first part is a The photoset mainly contains two parts. The first part is a cloud-like object and interfering block image set. The training set contains 466 images of cloud-like objects or interfering blocks larger than 4000 pixels. We manually categorized each image into the corresponding group. By using the training set, we can train the first interference detection CNN model. The testing set has 100 pairs of observation frames. We can use them to evaluate the extraction performance. The object type distribution is shown in Table 4. The second part of the photoset contains the observation photos of clouds. In the training set, we manually split the observation photos into 712 small squares, which were classified based on their shape by meteorologists. Moreover, we collected 100 large frame pairs (200 photos) to build a testing set. The distribution of the categories in this part is shown in Table 5.

TABLE 4. Interference and cloud object image dataset. The number in parentheses is the number of frame pairs.

Objects	Interfering blocks	Cloud-like objects	**Total**
Training set	269	197	466
Testing set	-	-	100(50)

TABLE 5. Distribution of the cloud-shape categories. The numbers in brackets are the numbers of frame pairs.

Categorises	Cumulus	Cirrus	Stratus	**Total**
Training set	283	218	206	712
Testing set	78(39)	62(31)	60(30)	200(100)
Whole set	366	280	266	912

B. TRAINING PROCEDURE

1) IFDM EXTRACTOR

To use the IFDM to extract cloud-like objects, we need to determine the suitable size threshold and train the CNN model for interference detection. The size threshold is only a prefilter for small interfering blocks. We have an interference detection CNN in the next step, which can accurately identify the interfering objects. Therefore, we want to keep as many cloud-like objects as possible. By analyzing the 1189 objects of different sizes from Table 1, we know that the minimum size of a cloud-like object is 4816 pixels. Therefore, we set 4815 as the size threshold for the IFDM extractor. Through this size threshold, 856 of 973 interfering blocks or ambiguous objects are filtered out. This filtering procedure greatly reduces the workload of the CNN interference detection model in the next step, which can save considerable computational resources.

After finding the suitable threshold, we can train the CNN interference detection model. We use the training set in Table 5 for training. During training, to avoid losing any training samples, we temporarily set the threshold to 0. We resize all the images in the training set to 64*64 and feed them into the CNN model. After running the model, the final FC layer outputs the predicted value. Then, the loss function is calculated. In this model, the loss function is the cross entropy between the predicted type and the labeled type. The Adam optimizer [9] is adopted to optimize the network parameters of the model. We set the learning rate to 1-e4. Moreover, we adopted dropout [21] layer with a probability of 0.25 to prevent network overfitting in the last convolutional layer. When the training step is complete, we combine the size threshold and the CNN interference detection model to build an IFDM extractor.

2) MULTI-CHANNEL CNN

Before training, we need to build a training batch. First, each image was resized, and all the images were scaled to a size of 128*128 pixels, and their original size was recorded. Then, we randomly selected three images of the same cloud type to form a training group. In each iteration of training, we fed a training group into the model. For each training batch, one batch size with different types of training groups was obtained.

After building the training batch, we can start to train the CNN model. For one training step, three images in the same training group are simultaneously fed into the three channels of the CNN model for training. Meanwhile, their size data are fed into the fusion layer. Similar to the CNN model in the IFDM, the cross-entropy loss function is calculated in the output FC layer. The Adam optimizer [9] is adopted to optimize the network parameters of the model. We add a dropout [21] layer with a probability of 0.5 at the last convolutional layer of each channel and another dropout [21] layer with a probability of 0.25 at the FC1 layer. In training, the batch size was 128, the learning rate was 1-e4, and the number of iterations was 10,000.

C. TESTING

We will introduce the testing method and process in this subsection. The analysis and discussion will be presented in Section 5.

1) CLASSIFICATION PERFORMANCE

We evaluated our proposed method with two testing approaches: the IFDM extractor for extraction ability and the whole classification system for classify ability.

For the interference detection part, we tested the ability of the IFDM extractor to extract cloud-like objects. In addition, we used traditional image or video extract algorithms, namely, the Canny edge detector (Canny) [22], the double-frame FDM (D-FDM) [8], the triple-frame FDM (T-FDM) [8] and the FDM with edge detection (FDM-ED) [23], to compare with our method. We use these algorithms to extract objects from the video frame from the test set in Table 4.

In terms of the classification ability, we combined the IFDM extractor with the Multi-channel CNN classifier and tested the classification ability of the whole system. We used 200 images in the testing set from Table 5 to test the classification accuracy. Moreover, we also used traditional CNN models, namely, LeNet-5 [13], AlexNet [14] VGG-16 [24], BNInception [35], ResNet-152 [34] and Faster R-CNN [25], and a traditional classification algorithm, SVM [7], to perform the comparison testing.

2) COMPUTATIONAL RESOURCE CONSUMPTION

Because the classification system in this paper needs to work in the embedded system, its computational resource consumption is very important. We evaluate the whole system from two aspects: memory usage and time cost. We store the testing frames in Table 5 on the hard disk before testing. After starting the classification program, we record the memory occupied by the program every 10 seconds. We will also record how long the program takes to classify each image. In addition, we also test the resource consumption of other classification algorithms, namely, LeNet-5 [13], AlexNet [14] VGG-16 [24] and Faster R-CNN [25].

TABLE 6. Accuracy of the extraction. The size threshold is 4815 pixels.

Method	Size>4815	Size>9000
Canny [22]	71.12%	77.37%
D-FDM [8]	63.53%	72.35%
T-FDM [8]	84.43%	89.32%
FDM-ED [23]	92.00%	97.67%
IFDM	**97.33%**	**98.78%**

V. ANALYSIS AND DISCUSSION

In this section, we analyze the experimental results from the testing step. Then, we will perform an overall evaluation of our system. We also evaluate and discuss the performance under different conditions.

A. RESULT ANALYSIS

1) IFDM EXTRACTOR

In this system, the main function of the extractor is to extract the cloud-like objects in a large photo. To quantitatively evaluate the proposed model, we use the accuracy rate as the measurement standard. We regard the extracted cloud-like objects as the correct extraction, and other interfering blocks are regarded as the incorrect extraction. Then, the extraction accuracy is as follows (9):

$$Accuracy = \frac{Cloud\_Like\ Objects}{Total\ Number\ of\ Extracted\_Objects} \tag{9}$$

We manually identify the output objects of the IFDM extractor and record the actual types of each object. Then, we can obtain the accuracy of the extraction. We also tested the Canny [22], D-FDM [8], T-FDM [8] and FDM-ED [23] algorithms in the same way. The results are shown in Table 6. Because we have set the size threshold for IFDM, to make a fair comparison, we calculate statistics on the objects above the threshold. Compared with other detection algorithms, the measuring accuracy of the IFDM is obviously higher. Since most of the cloud-like objects extracted by the extractor are larger than 9000 pixels, we also calculate statistics on all the extracted objects larger than 9000 pixels. We can see that the accuracy of all the algorithms has been improved, and the accuracy difference between the IFDM and other algorithms is being reduced. However, the accuracy of the IFDM is still the highest.

2) CLASSIFICATION SYSTEM

We combine the IFDM extractor with the Multi-channel CNN classifier to form our classification system. We use Sensitivity Precision and Accuracy to evaluate the system. The accuracy can be calculated in terms of positive and negative classes as (10):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

where TP (True Positives) is the number of correctly classified instances of the class of the specific type, TN (True Negatives) is the number of correctly classified instances of the rest of the classes, FP (False Positives) is the number of misclassified instances of the rest of the classes and FN (False Negatives) is the number of misclassified instances of the class of the specific type.

And the Sensitivity and Precision can be expressed as (11),(12):

$$Sensitivity = \frac{TP}{TP + FN} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

In addition, we use f-score in the evaluation. In this paper we use Macro-F1, which can be expressed as (13):

$$F1\_Score = 2 \times \frac{Sensitivity \times Precision}{Sensitivity + Precision} \tag{13}$$

After we finished testing and obtained the output data, we created a confusion matrix. The matrix is shown in Table 8 Then, we can obtain TP, TN, FP, FN by comparing the testing output predicted by our classification system with the actual category. We can calculate the sensitivity, precision, and accuracy. We also perform the same test on LeNet-5 [13], AlexNet [14] VGG-16 [24], BNInception [35], ResNet-152 [34] and Faster R-CNN [25] and then obtain the data in the same way. The results are shown in Table 7.

Based on the above experiments and data, our classification system has a very good classification ability. Through Figure 8, we can intuitively understand the costs of these models. Compared with most mainstream CNN image classification models, our system has an obviously advantageous accuracy rate. Although our system has the same accuracy as Faster R-CNN, our model system has better sensitivity, precision and F1-score. Therefore, our classification system has very competitive performance.

3) RESOURCE CONSUMPTION

By testing the image in Table 5 and recording the memory and recognition speed, we can obtain the memory cost and time cost of each classification model. Then, we calculate the average and standard deviation of each cost. The results are shown in Table 9.

From the above results, we can see that the resource consumption of our classification system is also not bad. Although it consumes slightly more resources than the other CNN models, the cost difference is not very large. We can see that the advantage of our classification system in resource consumption is obvious compared with Faster R-CNN [25]. Compared with Faster R-CNN [25], our system only needs approximately 1/3 of the memory and 1/4 of the time to complete the task. Therefore, our system has a very competitive performance in computational resource consumption.

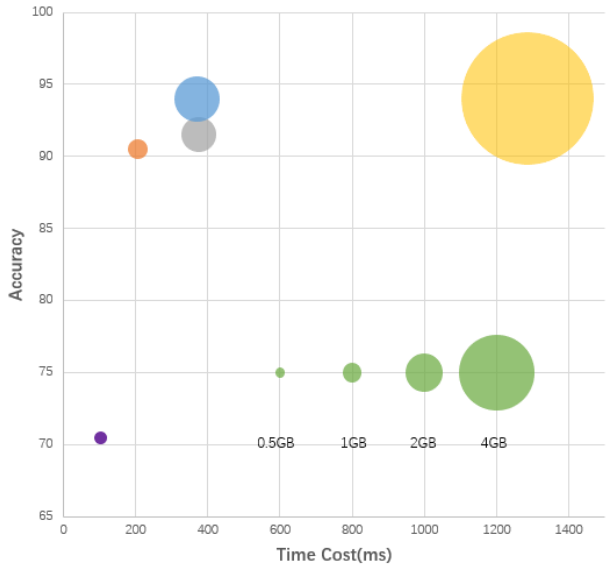
B. PERFORMANCE DISCUSSION

1) OVERALL EVALUATION

Through the analysis in the previous section, we can see that our classification system is very competitive across different

TABLE 7. Comparison of the proposed method with the other methods.

Method	Sensitivity			Precision			Accuracy	F1_Score
	Cumulus	Cirrus	Stratus	Cumulus	Cirrus	Stratus		
SVM [7]	53.85%	50.00%	53.33%	50.60%	57.41%	50.79%	52.50%	0.5255
LetNet-5 [13]	70.51%	67.74%	73.33%	69.62%	72.41%	69.84%	70.50%	0.7054
AlexNet [14]	89.74%	91.93%	90.00%	90.91%	90.48%	90.00%	90.50%	0.9051
VGG-16 [24]	89.74%	93.55%	91.67%	90.91%	92.06%	91.67%	91.50%	0.9160
BNInception [35]	88.46%	91.94%	93.33%	**94.74%**	93.75%	93.33%	90.00%	0.90028
ResNet152 [34]	92.31%	**96.77%**	88.33%	93.51%	90.91%	92.98%	92.50%	0.92417
Faster R-CNN [25]	**92.31%**	**96.77%**	93.33%	**94.74%**	93.75%	93.33%	**94.00%**	0.94026
Our System	**92.31%**	93.55%	**96.67%**	92.31%	**96.67%**	**93.55%**	**94.00%**	**0.94157**



Data Sheet

	Method	Time Cost	Accuracy	Memory Cost
	LetNet5	102ms	70.50%	0.67GB
	AlexNet	206ms	90.50%	1.02GB
	VGG-16	376ms	91.50%	1.87GB
	Faster R-CNN	1284ms	94%	7.06GB
	Our System	369ms	94%	2.44GB

FIGURE 8. The ball chart of overall performance. Y-axis: Accuracy X-axis: Time cost ball size: memory usage.

TABLE 8. Confusion matrix of our system's output. The data outside the brackets is our system's output, and the data in the brackets is from Faster R-CNN.

		Prediction		
		Cumulus	Cirrus	Stratus
Actual	Cumulus	72(72)	2(2)	4(4)
	Cirrus	4(2)	58(60)	0(0)
	Stratus	2(2)	0(2)	58(56)

TABLE 9. Computational resource consumption.

Method	Memory Cost		Time Cost	
	Average(GB)	Standard Deviation	Average(ms)	Standard Deviation
LetNet-5 [13]	0.67	0.04	102	5.0
AlexNet [14]	1.02	0.05	206	3.2
VGG-16 [24]	1.87	0.05	376	3.0
Faster R-CNN [25]	7.06	0.06	1284	12.5
Our system	2.44	0.08	369	8.9

aspects. Our system can quickly make accurate classifications of cloud shape and consume relatively fewer computational resources. A ball chart is shown in Figure 8, which intuitively shows the performance of the system across different aspects.

From this plot, we can be seen that our system has the highest classification accuracy. Its accuracy is the same as that of Faster R-CNN [25], which is higher than that of the other CNN classification models. Moreover, when using the same test platform, the time cost for recognizing one photo by our system is similar to that of VGG-16 [24] and AlexNet [14]. However, it is shorter than that of Faster R-CNN. Moreover, the memory usage is only slightly larger than that of VGG-16 and much smaller than that of R-CNN. Based on the above evaluation and discussion, it is obvious that our system achieves a good overall performance. It has more advantages than the other classification models under the condition of low computational resources.

2) DISCUSSION OF THE MISCLASSIFIED RESULTS

Due to the efficient extraction by the IFDM and the accurate classification by the multichannel CNN, our system can achieve a very good overall performance. However, we want

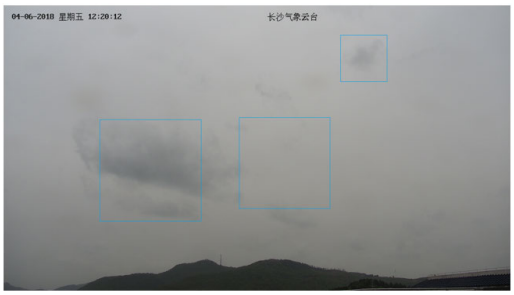


FIGURE 9. An example of an incorrect extraction. The blue square is the extracted image.

TABLE 10. Accuracy of the different poor observation conditions.

Weather Condition	Rain	Strong winds	Foggy	Sunset
Accuracy	90.32%	92.86%	80.00%	91.67%
Sensitivity	90.15%	83.33%	76.67%	90.00%
Precision	92.34%	95.83%	89.74%	93.33%
F1_Score	0.9111	0.8782	0.7912	0.9121

to know which part of our classification system can be improved. Therefore, we analyzed several incorrectly classified photos.

Through analysis, we found that if there are two kinds of clouds in the photo, our classification system is prone to misjudgment. The reason is because the IFDM mainly relies on the difference between two frames when extracting cloud-like objects. However, faster-moving clouds have larger differences. If there are two kinds of clouds in the photo and the minority clouds move faster than the majority clouds, the IFDM could extract the faster-moving cloud-like objects. Therefore, the multichannel CNN classifier classifies the photo based on the minority cloud-like objects. The output will be incorrect based on these objects.

Figure 9 shows a good example of this type of error. This picture should be classified as a stratus cloud. However, a cumulus cloud is passing by at a high speed. The high-speed cumulus cloud generates larger differences between the two frames. The FDM produces an incorrect candidate region based on this large difference, while the PSNR of this candidate region is just relatively small. In the end, the extractor extracts the cumulus cloud with a smaller PSNR instead of the stratus cloud, which causes an incorrect classification. Fortunately, this situation is very rare, so it has little effect on the accuracy. When we accumulate enough data after the system trial execution, we will study this problem and try to avoid it in the future.

3) POOR OBSERVATION CONDITION

Although the system does not need to work under poor observation conditions, we use photos of severe weather conditions to explore the impact of poor observation conditions on the system performance [27]. Some images under poor observation conditions are shown in Figure 10.

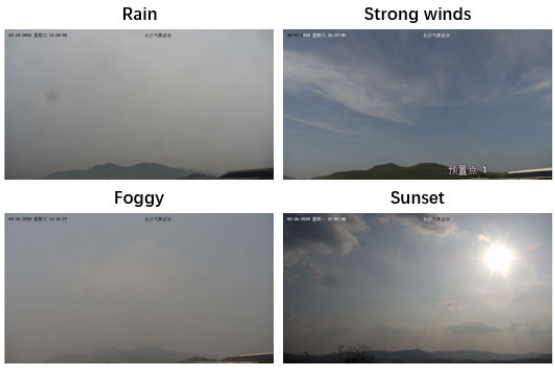


FIGURE 10. Different poor observation conditions. Rain, strong winds, foggy and sunset are the four main poor observation conditions.

In the experiment, we found that the classification performance of our system decreased when testing with these observation photos. The results are shown in Table 10. However, despite the foggy condition, most of them only decrease by approximately 2-4%, and the accuracy performance is still not poor. The main reason for the sharp decline in the accuracy on foggy days is that the camera cannot clearly capture the clouds in the sky. Even for meteorologists, it is very difficult to recognize cloud shapes under this condition. Due to the limited dataset, we have not found an effective method to improve the accuracy. Therefore, this might be a limitation of the classification system.

Fortunately, the system can know the observation conditions accurately through other sensors (visibility perception cameras). Thus, the classification system can avoid working in foggy conditions and producing the wrong cloud-type data. Therefore, the limitation will not have a great impact on the actual observations. In the future, with the accumulation of data, we hope we can solve the bad weather problem by training a recognition model for foggy conditions.

VI. CONCLUSION

In the present research, we implement a photoclassification system for cloud shapes. The classification system can work in a hardware environment with limited computational resources, and the classification accuracy of the system can reach 94%. Although the performance for inclement weather needs to be improved, the overall classification accuracy of the system output is satisfactory. Compared with several recognition algorithms (LeNet-5, AlexNet, VGG-16 and Faster R-CNN), our classification system has a very competitive accuracy and consumes much fewer computational resources: 1/3 of the memory usage and 1/4 of the processing time than Faster R-CNN. With this classification system, meteorologists can greatly reduce the time for categorizing weather photos, which will improve the efficiency of meteorological statistics. With comprehensive statistical data, meteorologists can provide better weather predictions. The China Meteorological Administration has deployed three systems for trial execution.

However, some limitations of the system still need to be improved. The extractor and the classification ability can be improved under poor observation conditions. Due to the limited data, the testing phase is not very comprehensive. In the future, we can try some new classification models or make further improvements to the extractor based on feedback and new data. In addition, with these new data, we can also test the whole classification system more comprehensively. In the end, we hope our research will be helpful to the application of machine learning in meteorological observation and edge computing.

REFERENCES

[1] C. E. Duchon and M. S. O'Malley, "Estimating cloud type from pyranometer observations," *J. Appl. Meteorol.*, vol. 38, no. 1, pp. 132–141, Jan. 1999.

[2] T. Letian, "China cloud map (new edition)," (in Chinese), *Meteorology*, vol. 9, no. 7, p. 34, 1983.

[3] H. R. Pruppacher and R. L. Pitter, "A semi-empirical determination of the shape of cloud and rain drops," *J. Atmos. Sci.*, vol. 28, no. 1, pp. 86–94, Jan. 1971.

[4] J. L. Gaumet, J. C. Heinrich, M. Cluzeau, P. Pierrard, and J. Prieur, "Cloud-base height measurements with a single-pulse erbium-glass laser ceilometer," *J. Atmos. Ocean. Technol.*, vol. 15, no. 1, pp. 37–45, Feb. 1998.

[5] A. Heinle, A. Macke, and A. Srivastav, "Automatic cloud classification of whole sky images," *Atmos. Meas. Techn.*, vol. 3, no. 3, pp. 557–567, May 2010.

[6] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[7] M. R. Azimi-Sadjadi and S. A. Zekavat, "Cloud classification using support vector machines," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. Taking Pulse Planet, Role Remote Sens. Manag. Environ. (IGARSS)*, vol. 2, Jul. 2000, pp. 669–671.

[8] N. Singla, "Motion detection based on frame difference method," *Int. J. Inf. Comput. Technol.*, vol. 4, no. 15, pp. 1559–1565, 2014.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[10] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based CNN with improved triplet loss function," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1335–1344.

[11] N. Jawas, and N. Suciati, "Image inpainting using erosion and dilation operation," *Int. J. Adv. Sci. Technol.*, vol. 51, pp. 127–134, Feb. 2013.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[13] 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.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[16] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4694–4702.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.

[18] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 2366–2369.

[19] H. Ye, Z. Wu, R. W. Zhao, X. Wang, Y. G. Jiang, and X. Xue, "Evaluating two-stream CNN for video classification," in *Proc. 5th ACM Int. Conf. Multimedia Retr.*, Jun. 2015, pp. 435–442.

[20] *NVIDIA Jetson TX2 SERIES Technical Specifications*. Accessed: Nov. 25, 2019. [Online]. Available: https://developer.nvidia.com/embedded/develop/hardware

[21] 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.

[22] Y. Luo and R. Duraiswami, "Canny edge detection on NVIDIA CUDA," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2008, pp. 1–8.

[23] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Proc. 4th Int. Conf. Image Graph. (ICIG)*, Aug. 2007, pp. 519–523.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[26] T. H. Wong, A. K. Qin, S. Wang, and Y. Shi, "cuSaDE: A CUDA-based parallel self-adaptive differential evolution algorithm," in *Proc. 18th Asia Pacific Symp. Intell. Evol. Syst.*, vol. 2, 2015, pp. 375–388.

[27] Z. Wang, W. Zheng, C. Song, Z. Zhang, J. Lian, S. Yue, and S. Ji, "Air quality measurement based on double-channel convolutional neural network ensemble learning," *IEEE Access*, vol. 7, pp. 145067–145081, 2019.

[28] P. Nousi, E. Patsiouras, A. Tefas, and I. Pitas, "Convolutional neural networks for visual information analysis with limited computing resources," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 321–325.

[29] R. P. Lawson, L. J. Angus, and A. J. Heymsfield, "Cloud particle measurements in thunderstorm anvils and possible weather threat to aviation," *J. Aircr.*, vol. 35, no. 1, pp. 113–121, Jan. 1998.

[30] Z. Li, M. Dong, S. Wen, X. Hu, P. Zhou, and Z. Zeng, "CLU-CNNs: Object detection for medical images," *Neurocomputing*, vol. 350, pp. 53–59, Jul. 2019.

[31] A. Urbonas, V. Raudonis, R. Maskeliūnas, and R. Damaševičius, "Automated identification of wood veneer surface defects using faster region-based convolutional neural network with data augmentation and transfer learning," *Appl. Sci.*, vol. 9, no. 22, p. 4898, Nov. 2019.

[32] F. Gao, T. Huang, J. Sun, J. Wang, A. Hussain, and E. Yang, "A new algorithm for SAR image target recognition based on an improved deep convolutional neural network," *Cognit. Comput.*, vol. 11, no. 6, pp. 809–824, Jun. 2018.

[33] G. Chen, C. Li, W. Wei, W. Jing, M. Woźniak, T. Blažauskas, and R. Damaševičius, "Fully convolutional neural network with augmented atrous spatial pyramid pool and fully connected fusion path for high resolution remote sensing image segmentation," *Appl. Sci.*, vol. 9, no. 9, p. 1816, May 2019.

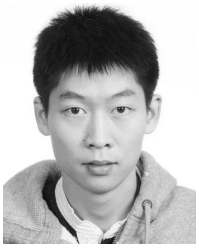
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

MENGYANG ZHAO received the B.S. degree in electronic and information engineering from Sichuan University, China, in 2017. He is currently a graduate student with the Electrical and Computer Engineering Department, Tufts University. He was a network image transmission engineer at MoJo Data Co., Ltd., from 2017 to 2018. In addition, he received the CCIE certification, in 2018. His current research interests include image processing, pattern recognition, image transmission, and edge computing.

CHORNG HWA CHANG received the B.S. degree from National Cheng Kung University (NCKU), the master's degree in CS from Montana State University, and the Ph.D. degree in electrical and computer engineering from Drexel University. He is currently an Associate Professor with the Electrical and Computer Engineering Department, Tufts University. He is the Director of the Tufts Wireless Laboratory and the Program Director of computer engineering. His research interests include computer architecture, parallel processing, computer networking, wireless networks, and edge computing.

WENBIN XIE received the master's degree in atmospheric observation and research from the Chinese Academy of Meteorological Sciences, Beijing, China, in 1991. From 1991 to 1996, he worked at Hunan Meteorological Station and was responsible for meteorological and atmospheric observations and weather data collection. He has served as the Director of the Hunan Meteorological Station, from 1995 to 1996. He is currently a Senior Weather Research Consultant at Yipai Weiye Co., Ltd. He won the Lifetime Honor Award from the China Meteorological Society, in 2018.



ZHOU XIE received the first B.S. degree in computer engineering and the second B.S. degree in electrical engineering from the University of California, Irvine, CA, USA, in 2009 and 2010, respectively. He is currently an Algorithm Development Engineer at Yipai Weiye Co., Ltd. He has experiences in software and algorithm development. His implements algorithms and test their performance.

JINYONG HU received the B.E. degree in Internet-of-Things engineering from the Beijing University of Posts and Telecommunications, China, in 2018. From 2017 to 2018, he was a Research Assistant at the Artificial Intelligent Laboratory, Computer Network Information Center, Chinese Academy of Sciences. He is currently a graduate student in computer science at Tufts University, Medford. He improves the CNN model and assists the testing.