Retraction Notice

The Editor-in-Chief and the publisher have retracted this article, which was submitted as part of a guest-edited special section. An investigation uncovered evidence of systematic manipulation of the publication process, including compromised peer review. The Editor and publisher no longer have confidence in the results and conclusions of the article.

LL either did not respond directly or could not be reached.

Support vector machine parallelized remote sensing image classification algorithm based on big data

Li Liao[*]
Chongqing City Vocational College, Chongqing, China

Abstract. With the development of big data technology, machine learning classification methods have been widely used in the classification and recognition of remote sensing images. For remote sensing big data, how to quickly and efficiently use machine learning classification algorithms to classify remote sensing images is an urgent problem. It is a general term for the theory, method, technology, and activities of obtaining valuable information based on massive remote sensing data sets, synthesizing auxiliary data from other sources, and using big data thinking and means. The purpose of this paper is to study the support vector machine (SVM) parallelized remote sensing image classification algorithm based on big data. We propose a parallel nesting of GPU in MPI multiprocesses based on the big data framework, which can more effectively improve the calculation processing speed and build a high-performance SVM parallel computing framework based on the big data framework. The optimization problem of SVM considers both empirical risk and structural risk minimization and requires maximum edge distance when constructing hyperplane decision boundaries, so there is ample space between the interval boundaries to accommodate the test samples. Based on this framework, we improve the machine learning SVM algorithm and realize the high-performance parallel computing of SVM classification algorithm on this platform. It is an efficient hybrid parallel mode to nest GPU in MPI multiprocess in parallel. When the number of nodes is 2, 4, and 6, the speedup of the SVM classification algorithm is 1.52, 2.24, and 2.55. © *2022 SPIE and IS&T* [DOI: 10.1117/1.JEI.31.6.062005]

Keywords: big data technology; support vector machine; remote sensing image; classification algorithm; machine learning; image processing.

Paper 220378SS received Apr. 14, 2022; accepted for publication Jul. 5, 2022; published online Aug. 2, 2022.

1 Introduction

In recent years, with the continuous development of artificial intelligence, in the field of remote sensing, machine learning classification methods have been widely used in remote sensing image classification and recognition. However, with the advent of the era of big data, remote sensing image data have grown exponentially, showing obvious "big data" characteristics. The number of bands acquired on remote sensing images differs due to different sensors, and their values are jointly represented by the values of corresponding location points in different bands; none of the remote sensing images are lossy compressed. In addition, with the continuous improvement and improvement of remote sensing observation technology, the spectral resolution of remote sensing images obtained by humans through remote sensing monitoring platforms has continued to increase, and with the increasingly convenient collection of remote sensing data, remote sensing technology has entered a new era of remote sensing big data. Traditional remote sensing image classification methods generally assume that the data are normally distributed, obtain distribution parameters from training samples, and then classify unknown pixels. But for remote sensing data, the assumption of normal distribution is not valid, especially when the features of the ground are more complicated. The normal distribution is one of the most important probability distributions. The normal curve is bell-shaped, with low ends, high middle, and symmetrical left and right because its curve is bell-shaped, so people often call

*Address all correspondence to Li Liao, liaoli@cqcvc.edu.cn

1017-9909/2022/$28.00 © 2022 SPIE and IS&T

it a bell curve. Nonparametric classification does not have high requirements on the data and does not need to assume the assumption of a normal distribution. It builds statistical models through learning, also known as machine learning methods. With the rapid development of remote sensing big data, and machine learning classification methods are a solution to find the optimal solution through continuous iteration, therefore, how to use machine learning algorithms to quickly and efficiently improve the classification speed of remote sensing images is an urgent problem.

Nowadays, the era of big data of remote sensing has arrived. The traditional serial computing methods can no longer meet the needs of large data calculations. Large-scale distributed parallel computing systems provide the possibility for rapid processing of massive remote sensing big data. The emergence of clusters has brought new opportunities for the powerful use of parallel computing. Given that clusters have fast computing and processing capabilities, under this trend, large-scale machine learning has become possible, and people are now entering the era of machine learning. In today's intelligent era, science and technology for processing big data continue to emerge, and the concepts of Hadoop and Spark have emerged at the historic moment. This distributed parallel computing technology provides a very efficient and practical solution for current big data storage and processing program. Distributed parallel computing offers near-infinite scalability, which of course depends on the external environment, but in general distributed parallel computing does offer the highest level of scalability. Among these technologies, Spark is a distributed memory-based computing method. Compared with Hadoop's MapReduce-based algorithm, Spark has a stronger speed performance advantage and can maintain high reliability and fault tolerance in multiple iterative algorithms. Machine learning is a method that requires continuous iterative optimization. Therefore, the parallel processing technology based on the Spark framework is very suitable for application in machine learning algorithms. At the same time, with the rapid development of science and technology, computer hardware for processing big data has entered a rapid update. Using MPI's parallel mode can both divide complex problems into multiple subprocesses for parallel processing, that is, divide complex problems into clusters of several subproblems composed of multiple computers for overall calculation. On the one hand, it can greatly reduce the complexity of designing parallel programs, and on the other hand, it can effectively reduce the calculation time.

Meng and his team believe that Apache Spark is a popular open source platform for large-scale data processing and is well suited for iterative machine learning tasks. They introduced Spark's open source distributed machine learning library, MLlib. MLlib provides effective features for a variety of learning settings and includes some basic statistics, optimization, and linear algebra primitives. The MLlib provided with Spark supports multiple languages and provides a high-level API that leverages Spark's rich ecosystem to simplify the development of end-to-end machine learning pipelines. MLlib has experienced rapid growth due to its vibrant open source community (including 140 contributors) and includes extensive documentation to support further growth and get users started quickly.[1] Shrivastava et al. found that uncertainty in electricity prices made it difficult for power market participants to make accurate forecasts. Forecast intervals (PIs) are statistical tools that quantify the uncertainty associated with forecasts by estimating the range of future electricity prices. Predictions are estimates or approximations and contain some uncertainty, which arises from errors in the model itself and noise in the input data, and the prediction interval is a quantification of prediction uncertainty that provides upper and lower probability bounds for the estimation of the outcome variable. The cost of generating PI by traditional methods based on neural networks (NNs) is that it is computationally intensive and assumptions about data distribution are suspect. In this work, they proposed a technology that does not suffer from the above limitations, which can generate high-quality PI in a short time. The proposed method uses support vector machines (SVM) to directly generate upper and lower limits for future electricity prices. By using particle swarm optimization (PSO) technology to minimize the modified objective function based on PI, the best model parameters can be obtained.[2] The PSO algorithm is population-based and moves the individuals in the population to good areas based on their fitness to the environment. But it treats each individual as a point in the search space, flying at a certain speed in the search space, which is dynamically adjusted according to its own flight experience and the flight experience of its companions. Chenghe and his team believe that remote sensing image scene classification plays an important role in a wide range of applications, so it has received great attention. In the past few years, great efforts have been made to develop various data sets or to propose various methods for scene classification from remote sensing images. However, a systematic review of the literature on datasets and methods for scene classification is still lacking. In addition, almost all existing datasets have many limitations, including the small scale of scene categories and the number of images, the lack of image variation and diversity, and the saturation of accuracy. These limitations severely limit the development of new methods, especially methods based on deep learning.[3]

The research contents of this paper include: (1) build a high-performance parallel computing platform based on the Big Data framework. Implemented on this platform, multiple processes are started through MPI between nodes to achieve parallel computing. Further parallelization is achieved by calling GPUs in MPI-enabled multiprocesses. (2) Research on remote sensing image classification method based on SVM and introduce the parallel computing strategy and principle of the classification algorithm in detail. Finally, the algorithm was improved to enable parallel computing for big data processing using MPI and GPU. (3) Using Landsat8 remote sensing image as experimental data, analyze the classification accuracy and calculation efficiency of high-performance parallel SVM classification algorithm under different experimental conditions.

2 Proposed Method

2.1 *Classification Algorithm of Remote Sensing Image*

2.1.1 *Remote sensing data*

Remote sensing image refers to the image that describes the features in digital form.[4–6] Remote sensing images are films or photos that record the size of electromagnetic waves of various features, mainly divided into aerial images and satellite photos. There are many types of images, visually, images are divided into visible images and invisible images. From the lightness and darkness of the image and the continuity in space coordinates, it can be divided into analog images and digital images. Simulated images are also called optical images, which are visible images. The spatial coordinates and brightness of the simulated images change continuously. A/D conversion of analog images can be converted into digital images. A digital image refers to an image stored, processed, and used by a computer. The spatial coordinates and lightness and darkness of a digital image are discontinuous. It is actually a two-dimensional matrix of some points with a certain value arranged in rows and columns since it only exists in digital form, it can only be seen if it is displayed or printed in gray or color.[7,8] Analog images are composed of continuous dots, whereas digital images are composed of discrete dots. Digital images are images obtained by digitizing analog images with pixels as the basic elements. The objects we usually process are digital images. After processing the digital image, we can obtain the analog image we need by D/A conversion. Figure 1 shows the structure associated with the preprocessing of remote sensing images.

2.1.2 *Basic data processing flow of remote sensing image classification algorithm*

Data preprocessing. The purpose of data preprocessing for remote sensing images is to highlight the target to be detected, improve the ability of remote sensing image interpretation, and locate the area to be studied. There are many methods for preprocessing remote sensing image data, common methods include image enhancement and filtering, and geometric correction, radiation correction, image registration.[9,10] Among them, geometric correction and radiation correction methods are very important, because they will affect the image classification results in a wide range. The classification method of remote sensing images needs to analyze different feature areas in remote sensing images of different phases, usually, the "overlap" method is

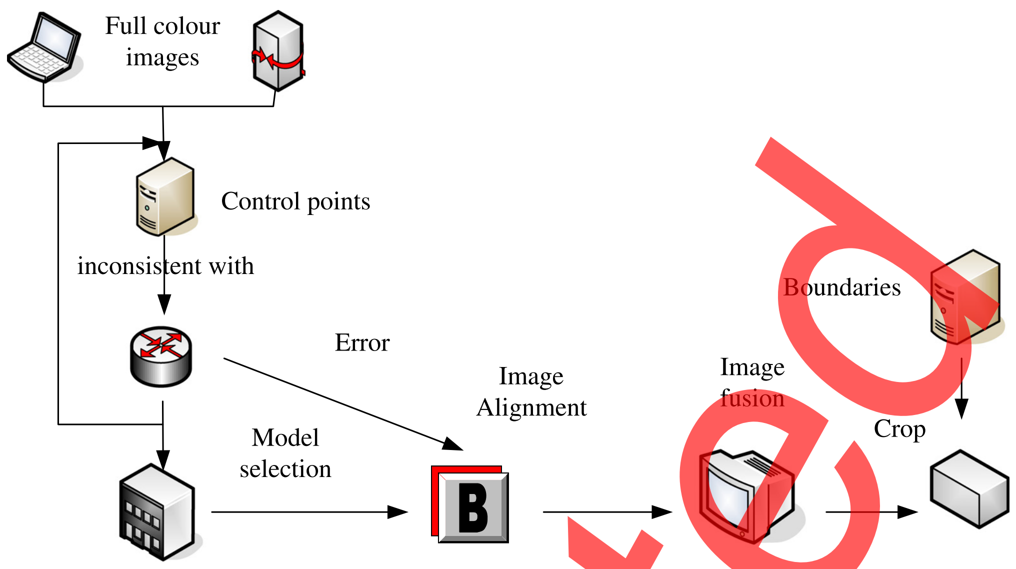

Fig. 1 Structure related to preprocessing of remote sensing images.

used to overlap the same feature areas in remote sensing images of different phases, it is necessary to have accurate position reference and gray standard for the multitemporal remote sensing images to be detected, geometric correction, and radiation correction can solve such problems.[11,12]

Image classification. Image classification is the core and key of the entire remote sensing image classification algorithm. It is necessary to choose an appropriate image classification method according to the needs of the actual project. Remote sensing image classification is to classify each image element in the image into different categories according to its spectral brightness, spatial structure characteristics, or other information in different bands according to some rules or algorithms. Extract information from the remote sensing images to be classified, and output the data in the form of data tables or graphs after analysis as a result, the process can be divided into three steps.[13–15] First, image feature extraction refers to finding the feature data of the ground cover area in the remote sensing image of the remote sensing image to be detected, and using the gray value of the ground cover area as the feature data. Then, the image feature analysis is used to compare the extracted image features to determine the feature information represented by each area in the remote sensing image. Finally, the remote sensing image classification method uses a decision classification method to distinguish detection features from different features and uses different values to represent different features and outputs classified images.[16–19] Figure 2 shows the image processing correlation.

Classification result processing. After obtaining the preliminary classification map, we can find that there may be some misjudgment areas by comparing the original images, which affect the results of image classification. The remote sensing image classification method needs to process the preliminary classification map. This step is usually processed by mathematical morphology.

Fig. 2 Image processing related situations.

2.2 *Parallel Computing Technology*

2.2.1 *Yarn resource management system*

In response to various limitations such as poor scalability, poor reliability, low resource utilization, and inability to support multiple computing frameworks in MRvl, Apache upgraded Hadoop from MRvl to RV2. MRv2 proposed a new concept Yarn. Yarn is an abstraction of resource management functions in MRvl. Yarn is a lightweight and elastic computing platform. Its birth not only solves the problem of low resource utilization, high operation and maintenance costs, and poor data sharing in the traditional model of "one computing framework, one cluster," but also enables multiple frameworks, both can be deployed to a common cluster, which facilitates resource management and task scheduling.[20,21] Yarn has many benefits, such as high resource utilization, low operation and maintenance costs, and data sharing.

2.2.2 *Spark platform*

Spark's core technology, RDD, is an abbreviation for elastic distributed data sets. RDD is a parallel fault-tolerant data set. Spark, including its upper layer architecture, is based on RDD. RDD can be understood as a collection of data elements distributed on the data that contains many of the same data type. It hermitically stores the data in a distributed manner, making the operation of RDD as easy as operating on a local data set, without concern, the distribution of many data behind the cluster.[22,23] For the data structure RDD, Spark defines two major methods: transformation and action. Since data are implicitly distributed in a distributed environment in Spark, the operation of RDD is mainly on the operation of elements, without focusing on the communication between elements, so communication in a distributed environment is implicitly implemented. Because Spark is based on Scala1251 is a functional programming language.[24,25] Therefore, transferring functions to RDD is very natural for data manipulation. In contrast to MPI's model of passing data to processes that manipulate data, in Spark, the running model is to pass operations on data to data. Therefore, the Spark model can be summarized as "data do not move code."

2.2.3 *MPI*

MPI uses clusters to implement multinode parallel computing, by dividing the task process, the purpose of parallel data processing is achieved.[26] MPI is a cross-language communication protocol for writing parallel computers, supporting peer-to-peer and broadcast, and mainly specifying how to exploit its features in various implementations. MPI is a more traditional parallel method used in grid computing. It is mainly used in cluster parallel computing using MPI for low communication efficiency requirements and small amount of communication data.[27,28] Compared with MPI, OpenMP is a parallel processing mode that runs on a process. It divides a process into multiple threads and realizes the simultaneous execution of threads to achieve visual parallel processing. Parallel visualization usually includes three parallel processing modes. Data parallelism divides data into multiple subsets and then executes programs in parallel to process different subsets of data at the subset granularity. MPI provides portability, standardization, performance, and functionality, including point-to-point messaging and collective operations, all in a range of user-specified groups of processes. MPI provides a library of entity set writing, debugging, testing, and performance distribution. The MPI library is commonly used for clustered systems in parallel programming because it is a messaging programming language.

2.2.4 *GPU*

Currently, NVIDIA's CUDA Tookit is widely used in GPU program development. It includes a compiler that can extend C++ language to C language for GPU kernels development. The CUDA programming model fully focuses on data parallelism operations and provides lightweight programming abstractions. Allows programmers to express the kernel with a single thread of execution and can also be extended to a dozen threads, allowing them to cooperate with each other, share resources, and further expand to thousands of threads running on GPU devices.[29,30] Because CUDA uses language extensions, the work of packaging and unpacking GPU kernel parameters and specifying various runtime kernel startup parameters is mostly performed by the CUDA compiler, which makes CUDA code easier to read.[31,32]

2.3 *Support Vector Machines and Parallelization*

2.3.1 *Support vector machine*

SVM is a new generation of machine learning algorithms based on statistical learning theory, dual programming theory classification, and function estimation methods. It is based on the principle of VC dimension and structural risk minimization and is mainly used to process samples. A limited number of text classification problems. The SVM algorithm can be summarized as the solution process of the quadratic programming problem during the training phase. Its main purpose is to find the global optimal solution. SVM is an applicable small sample learning method with a solid theoretical foundation. It basically does not involve probability measures and the law of large numbers, etc., and also simplifies the usual problems of classification and regression.

2.3.2 *Basic principles of support vector machines*

When dealing with two types of problems, the most commonly used method of SVMs is to establish an optimal classification surface in the sample space, so the two types of samples can be effectively separated, and at the same time, ensure that the distance between the two types of samples is the farthest. For the nonlinear separable problem, the SVM uses a mapping function to map the sample space vector into a high-dimensional space, making it linearly separable, and constructs an optimal hyperplane in this high-dimensional space. Let H be the classification line of the two types of samples, and H_1 and H_2 are the sample points with the smallest distance from the classification line in the two types of samples, and parallel to the classification line. The training sample points on the H_1 and H_2 straight lines are SVMs of two types of samples. The distance between the two straight lines H_1 and H_2 is called the classification interval.

The equation for defining a linear classification surface is

$$g(x) = w^T x + b, \tag{1}$$

where w is the weight coefficient vector and b is the classification threshold.

Normalize the discriminant function so both types of sample data meet $|g(x)| \geq 1$, where the equation is as follows:

$$y(w^T x_i + b_i) - 1 \geq 0, \quad i = 1,2,\ldots,I. \tag{2}$$

y_i is defined as the class label of the sample. The classification surface with the largest classification interval between the two classes is the optimal classification hyperplane. Therefore, to maximize the classification interval value, the Lagrange multiplier a_i is introduced and converted into the following format:

$$w(a) = \sum_{i=1}^{1} a_i = -\frac{1}{2}\sum_{i=1}^{i}\sum_{j=1}^{i} a_i a_j y_i y_j (x_i^T x_j). \tag{3}$$

After solving it twice, the following format is obtained:

$$w = \sum_{i=1}^{1} a_i y_i x_i, \tag{4}$$

$$b = \frac{1}{2}(w^T x(1) + w^T x(-1)). \tag{5}$$

a_i is defined as the support vector, and $x\ (1)$ and $x\ (1)$ are the support vector of the first type of sample and the support vector of the second type of sample, respectively. The final decision function is

$$f(x) = \text{sign}\left(\sum_{sv} y_i a_i (x^T x_i) + b\right). \tag{6}$$

Substitute the sample points of the class into the decision function. A value of 1 belongs to the class, otherwise it does not belong to the class.

2.3.3 *Parallel algorithm*

A parallel algorithm is a method and procedure for solving a problem jointly using multiple processors. It is executed by first decomposing a given problem into a number of subproblems that are as independent as possible from each other and then solving it simultaneously using multiple computers so as to finally find the solution to the original problem. Common parallel decomposition algorithms include parallel constraint distribution, parallel variable distribution, parallel gradient distribution, and parallel variable transformation. They decompose the original problem into a series of smaller subproblems and assign them to multiple processors to solve independently and in parallel. Each processor can use the existing mature serial algorithms with faster convergence speed. After the independent solution is completed, a synchronization step is performed to update the current data and assign new tasks to many processors until the termination condition is met. This solution process is easy to implement in a distributed computing environment, which can greatly improve the efficiency of the algorithm and reduce the storage load of a single processor.

The cascaded parallel SVM is a parallel algorithm based on incremental algorithms and multiclass learning. It first divides the learning sample set into multiple subsets and then distributes these sample subsets to multiple processor nodes. The SVMs are trained in parallel, and then their training results are merged and continued to be optimized until all support vectors for the entire training set are obtained. In this parallel mechanism, there is very little communication between different processors, which can achieve high parallel efficiency. Moreover, this method is simple in thought and easy to implement and is widely used to solve large-scale SVM problems.

The most representative cascaded parallel SVM algorithm is CascadeSVM algorithm. It uses a binary cascade structure to learn SVM in parallel. Through continuous support vector feedback and feedback, the algorithm can quickly converge to the global optimal solution of SVM. Experiments show that the cascaded SVM algorithm and its deformation method avoid data transmission between different processors and have good parallel solution efficiency for large-scale SVM problems. Although PSVM has unavoidable defects brought by these SVM theories such as difficulty in selecting kernel functions and parameters and lack of ability to integrate with prior knowledge, it can well train large-scale, ultra-large-scale data sets. The theoretical research on PSVM is also constantly developing and perfecting, and it has been well applied in many practical fields such as network intrusion detection and financial data analysis.

2.4 *MPI-GPU Parallel Computing Design*

2.4.1 *MPI/GPU parallel computing architecture*

At present, multiple parallel computing frameworks have been proposed, all of which have advantages over other frameworks in some aspects and cannot completely replace each other. The emergence of the resource management system Yarn solves this problem. Therefore, this paper builds an MPI/GPU parallel computing framework based on Yarn. The framework of MPI/GPU parallel computing processing mechanism is shown in Fig. 3.

The bottom layer is the resource layer and the data layer. The resource layer is the foundation of the entire platform. At this layer, the resource management system Yarn is deployed. Yarn, as the lowest layer of the entire architecture, has an irreplaceable status and directly determines

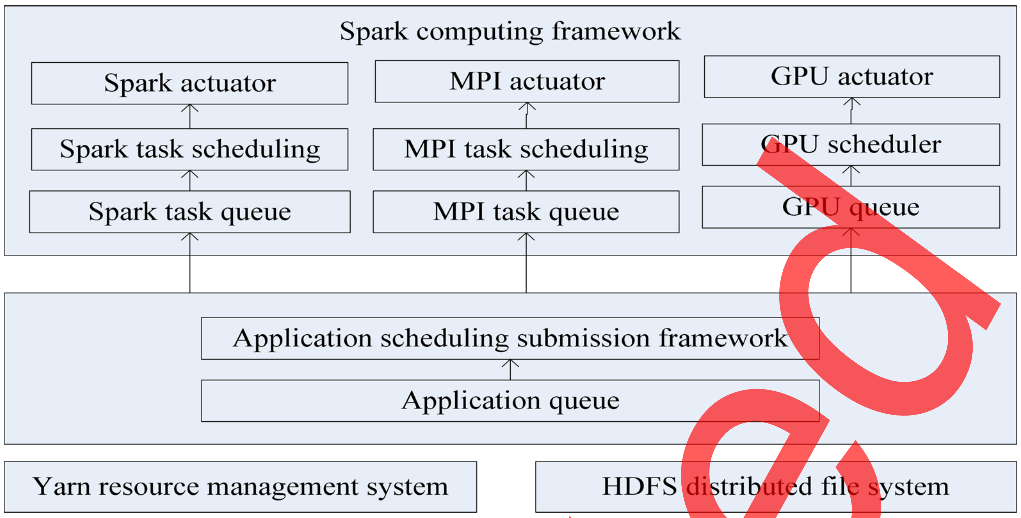

Fig. 3 Framework of MPI/GPU parallel computing processing mechanism under Spark.

the upper-level data and procedures. It is not only responsible for the resource management and allocation of the entire system but also for monitoring and managing all applications in the entire cluster.

The data layer is composed of Hadoop's distributed storage management system HDFS. HDFS is a distributed file system of Hadoop, which is mainly responsible for data storage management. It also adopts a master-slave structure, which provides file system support functions for distributed parallel computing systems. It is mainly responsible for the storage and management of data, programs, and tasks.

The middle layer is the application submission framework layer. The traditional Spark scheduling between applications does not have the explicit scheduling control but uses the natural order of task submission and adopts the first-in-first-out scheduling strategy for scheduling. This article provides a submission framework for task submission at the upper layer of Spark task submission and implements a batch control algorithm for batch applications within the submission framework. By controlling the order in which applications are submitted to Spark, the implementation of heterogeneous tasks on Spark task scheduling control.

The top layer is the Spark computing layer. This layer is mainly for data processing and multitask scheduling. It provides a unified computing service interface for CPU multicore parallel computing, MPI parallel computing, and CUDA parallel computing.

2.4.2 *Processing flow of MPI/GPU parallel computing*

First, open source software such as Hadoop, Spark, CUDA, and MPI are deployed on the cluster to build a cluster environment required for a distributed parallel computing system. Yarn and Spark application environments are created, and data and programs are uploaded to HDFS. Spark creates a SparkContest object that encapsulates the execution environment and cluster information of the Spark program. This article specifies the IP address of the master node, the name of the application, the installation path of Spark on the cluster, the job code on the local node, and the list of Jars files since. Spark deploys all the above information to Spark cluster nodes. This article uses SparkContest to obtain an RDD and read the file on HDFS. In the driver, this article reads the data from the main file. The function of the function testfile() is to obtain the KDD of the main file from HDFS. After that, this article can initialize each row of RDD. The flow of MPI/GPU parallel computing is shown in Fig. 4.

As shown in Fig. 4, the client uploads data and programs to HDFS on the master node, uses the scheduling algorithm to generate the application pending submission queue, and Spark submits the Job task to Yam. The submission framework queries the current task execution status on the Spark cluster. When there are idle resources and there are unsubmitted tasks, the tasks are submitted sequentially. Spark recognizes task types and further assigns tasks to various nodes.

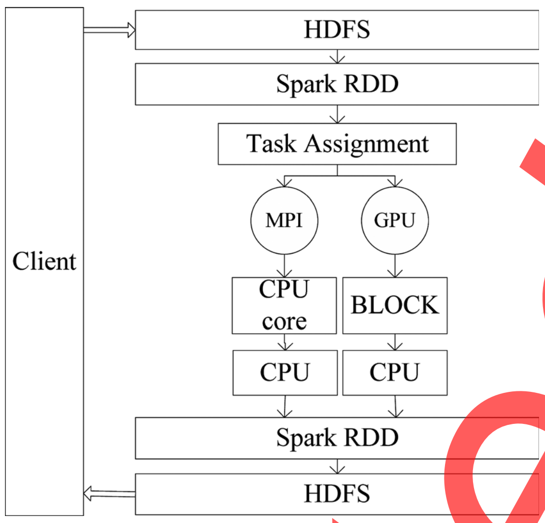

Fig. 4 MPI/GPU parallel computing process.

At the same time call the corresponding MPI program or GPU program. Finally, the operation results are passed back to the master node, and the master node is uploaded to HDFS. There are many schedules in the Spark framework, both between tasks and between applications. If you want to improve the overall operating efficiency by improving the scheduling mechanism, you can improve the scheduling mechanism inside the Spark task, the scheduling mechanism of the application "Application." In this experiment, to reduce the complexity of the experiment, this paper directly designs the greedy algorithm divided by time based on the original Application scheduling mechanism of the Spark framework. The order in which the Spark framework executes tasks is its submission order, which is the FIFO algorithm. According to theoretical analysis, you can change the order of task execution. The task scheduling framework improved by the scheduling algorithm in this paper can be abstracted into the following steps: first, submit the task description and generate a task description object in the Application queue. Second, use the scheduling algorithm for the task description objects in the Application queue to obtain a dispatched Application pending submission queue. Third, according to the resources of the Spark framework, when there are idle resources and there are uncommitted tasks in the Application pending queue, tasks are submitted in the order in the pending queue. When the resources on the Spark cluster are full, the task submission is suspended and waiting when resources on the Spark cluster are idle, continue to submit.

3 Experiments

3.1 *Data Collection*

The data used in the experiment are Landsat8 image data, the data range is a certain city, and the bands used are eight bands of visible light and near-infrared, each band has $3910 * 4020$ pixels. Through image analysis, five land cover types were identified: construction land, agricultural land, water body, wetland vegetation, and tidal flat. Here, we do not consider the difference between the construction land in the old and new urban areas, agricultural land with or without crops (including forest land), and clear or turbid water.

3.2 *Experimental Environment*

First, you need to configure various development environments before conducting experiments. This includes setting up a Spark21 cluster, configuring MPI and CUDA environments, and installing a specific development package pycuda for the development of the python language. Finally, for the reading and writing of remote sensing image data, the GeoPySpark development package is installed. The configuration of a single-node environment is shown in Table 1.

Table 1 Configuration of a single node environment.

CPU	Intel Xeon	Operating system	Hadoop	Spark	JDK
RAM	12GB	CentOS 7	2.7.4	Spark-2.1.0-bin-hadoop2.7	1.8.0

Table 2 Environment configuration of cluster nodes.

Node	CPU	GPU	RAM (GB)	IP
Slave1	Intel Xeon	GeoForce GTX 1080Ti	32	58.198.182.20
Slave2	Intel Xeon	GeoForce GTX 1080Ti	32	58.198.182.21
Slave3	Intel Xeon	GeoForce GTX 1080Ti	32	58.198.182.22
Slave4	Intel Xeon	GeoForce GTX 1080Ti	32	58.198.182.23
Slave5	Intel Xeon	GeoForce GTX 1080Ti	32	58.198.182.24
Slave6	Intel Xeon	GeoForce GTX 1080Ti	32	58.198.182.25

In the multinode test scenario, the experimental environment configuration of each cluster node is shown in Table 2.

3.3 *Experimental Scheme*

This article has designed the following four experimental schemes:

(1) The first experimental scheme is to compare the classification accuracy of a single-machine serial environment and MPI-GPU parallel in a single-node Spark environment.
(2) The second experimental scheme is to compare the classification speed and classification accuracy in the case of a single-node Spark environment with CPU serial, MPI multicore, CUDA parallel, and MPI-GPU.
(3) The third experimental scheme is to compare the classification speed and classification accuracy of MPI-GPU under different number of processes in a single node Spark environment.
(4) The fourth experimental scheme is to compare the classification speed and acceleration ratio of MPI-GPU under different nodes in a multinode environment.

4 Discussion

4.1 *Analysis of the Results of Experiment Schemes 1 and 2*

4.1.1 *Comparative analysis of classification confusion matrix and accuracy under serial/parallel conditions*

In the serial environment based on Spark, when calculating the confusion matrix, the sample categories are divided into five categories: construction land (including old urban construction land, new urban construction land), agricultural land (including crop agricultural land, cropless agricultural land), water bodies (including clear water bodies, turbid water bodies), wetlands, and tidal flats. The classification confusion matrix and its accuracy are shown in Fig. 5.

As shown in Fig. 5, for construction land and agricultural land under serial conditions, user accuracy and producer accuracy are both higher. The lowest user accuracy is wetland vegetation, which is 86.37%; the lowest producer accuracy is tidal flats. For land, it is 89.29%. The overall classification accuracy is 93.76%, and the kappa coefficient is 0.93.

In a Spark-based parallel environment (i.e., MPI-GPU environment), the classification confusion matrix and its accuracy are shown in Fig. 6.

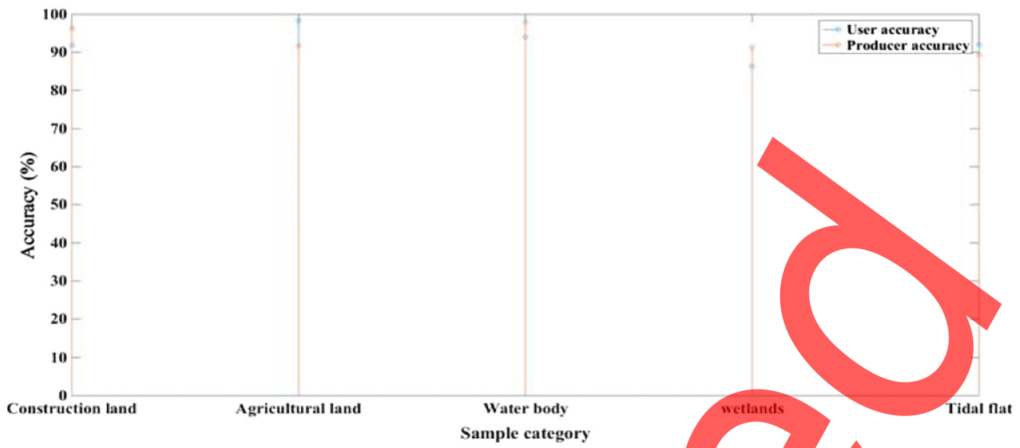

Fig. 5 Classification confusion matrix and its accuracy under serial conditions.

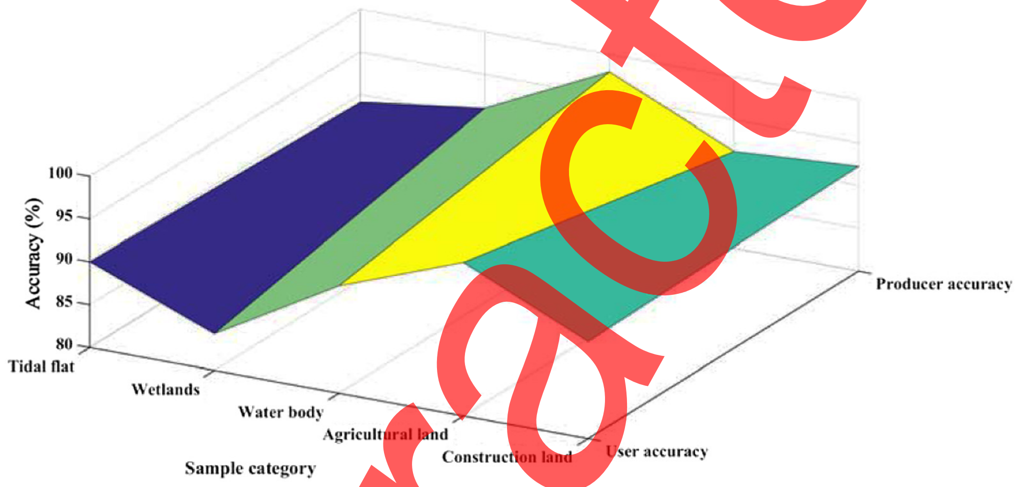

Fig. 6 Classification confusion matrix and its accuracy under parallel conditions.

As shown in Fig. 6, the overall classification accuracy under parallel conditions is 93.71%, and the kappa coefficient is 0.92. The highest user accuracy is for construction land, which is 97.88%, the lowest is wetland (84.26%); the lowest for producers is tidal flat, 89.10%, and the highest is water body (97.94%). Although the classification accuracy is reduced compared to the serial condition, the reduction is not large enough to meet the classification requirements.

4.1.2 *Comparative analysis of processing efficiency of support vector machine classification algorithms*

In the single-node Spark environment, the processing efficiency of the SVM classification algorithm in the case of CPU serial, MPI multicore, CUDA parallel, and MPI-CUDA is compared. Because the CPU serial environment does not process the training and classification process of the SVM algorithm in parallel, it takes the longest time. The processing efficiency comparison of SVM classification algorithms is shown in Table 3 and Fig. 7.

As shown in Table 3 and Fig. 7, the multicore environment based on MPI consumes only time better than CPU serial. GPU-based SVM classification uses the multithreaded characteristics of the graphics card to achieve parallel training of sample data, which takes only one-fifth of the time of the traditional CPU serial. Compared with the traditional CPU serial efficiency up to 6.4 times and better than MPI multicore and CUDA parallel classification efficiency. However, since the MPI multicore and the MPI-GPU framework proposed in this paper are integrated and optimized local classifiers of each subprocess to obtain the final classification model, the final

Table 3 Comparative analysis of processing efficiency of SVM classification algorithms.

CPU serial		MPI multicore		GPU parallel		MPI-GPU	
Time (s)	Accuracy (%)	Time (s)	Accuracy (%)	Time (s)	Accuracy (%)	Time (s)	Accuracy (%)
5752	93.76	1215	92.71	1023	93.76	817	92.71

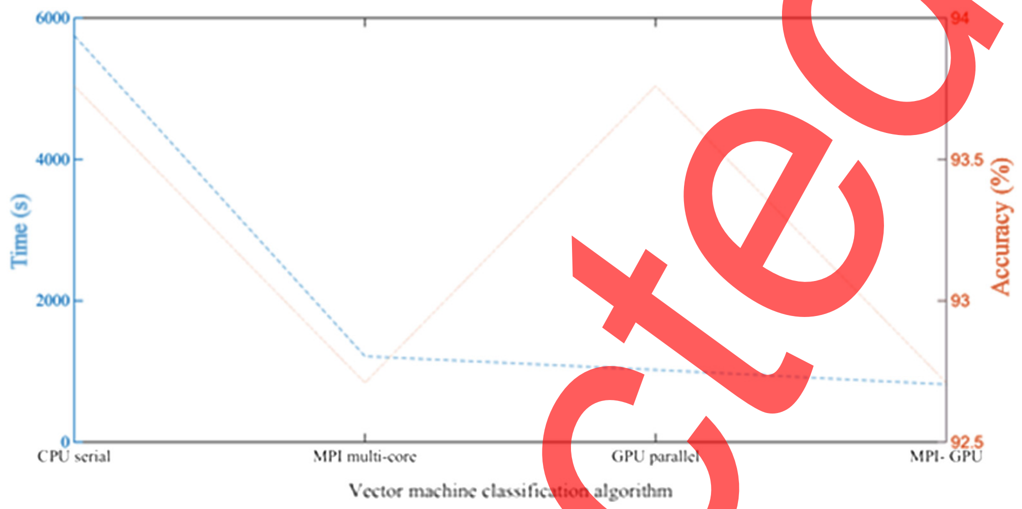

Fig. 7 Comparative analysis of processing efficiency of SVM classification algorithms.

classification model is not necessarily the global optimal classification model. So here in both environments, the classification accuracy of SVMs has decreased, from 93.76% to 92.71% but still within an acceptable accuracy range.

4.2 *Analysis of Results of Experiments 3 and 4*

4.2.1 *Time-consuming situation and classification accuracy of SVM parallelization algorithm under different number of processes*

Since MPI-GPU will affect the segmentation of training samples and the training of local classifiers when different numbers of secondary processes are turned on, the experiments were performed in a single-node environment by enabling different numbers of secondary processes to compare and analyze MPI-GPU-based SVM classification efficiency and accuracy. Multicore SVM classification algorithm is based on MPI. Although the accuracy of remote sensing image classification has decreased with the increase of secondary processes, its calculation speed has gradually increased and finally stabilized. The classification accuracy will increase as the number of secondary processes increases. The time-consuming situation and classification accuracy of MPI-GPU-based SVM under different numbers of secondary processes are shown in Table 4 and Fig. 8.

In Table 4 and Fig. 8, it can be seen that as the number of secondary processes increases, the classification accuracy of SVMs based on MPI-CUDA will decrease, from 94.64% (with 1 secondary process enabled) to 92.42% (with 10 enabled). When the number of secondary processes exceeds 8, the classification accuracy of SVM tends to be stable. When the number of secondary processes is 2, its time consumption is reduced by 43.8% than when the number of secondary processes is 1. When the number of secondary processes exceeds 9, the classification time consumption also tends to be stable. Although the classification accuracy will decrease with the increase of the number of secondary processes, the classification speed will gradually increase and eventually stabilize.

Table 4 MPI-GPU-based SVM time consumption and classification accuracy under different number of secondary processes.

Number of processes	1	2	3	4	5	6	7	8	9	10
Time (s)	4503	2526	1758	1244	917	804	727	691	656	643
Accuracy (%)	93.64	93.41	93.17	92.94	93.71	92.24	92.01	91.66	91.14	91.42

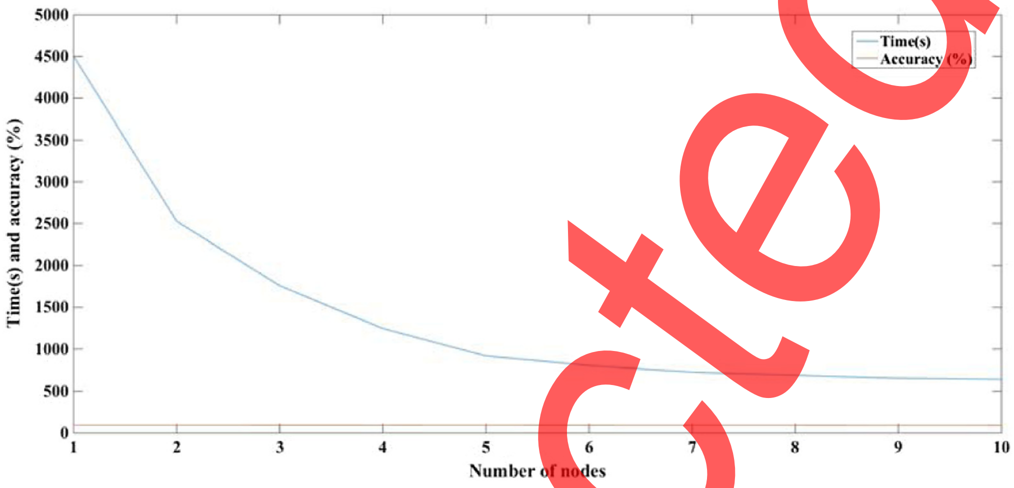

Fig. 8 MPI-GPU-based SVM time consumption and classification accuracy under different number of secondary processes.

4.2.2 *Analysis of the time-consuming situation and speedup of SVM parallelization algorithms under different nodes*

In the Spark framework cluster environment, data transmission and task execution under multiple nodes can be realized. Therefore, this study analyzes the processing efficiency of the SVM classification algorithm based on the MPI-GPU parallel framework under different numbers of nodes. Considering that the computer configuration of different nodes is different, in this experiment, five subprocesses are started uniformly under each node. The speedup ratio is widely used in a cluster environment to analyze the ratio of the time-consuming time under the condition of parallel processing and the condition of single processor. Therefore, this study selects this indicator to explore the effect of different node numbers on the classification efficiency of SVMs in MPI-GPU parallel framework. The acceleration ratio (S_v) is defined as follows:

$$S_v = T_s / T_p. \tag{7}$$

Among them, p represents the number of computing nodes, T_s represents the time consumed by a single processor, and T_p represents the time consumed by parallel processing. S_v is a real number greater than or equal to 1. The larger the S_v value, the faster the processing speed. The time consumption and acceleration of the SVM parallelization algorithm under different nodes are shown in Table 5 and Fig. 9.

As shown in Table 5 and Fig. 9, under different numbers of nodes, the SVM classification algorithm based on the MPI-GPU parallel framework has a faster processing speed as the number of nodes increases, and the time consumption gradually decreases, but the amplitude gradually decreases. Since the number of secondary processes opened by each node number is 5, when the number of nodes is 2, it means that the classification process is performed by a total of 10 secondary processes at the same time, which takes 572 s, which is better than starting under a single node time consuming for 10 secondary processes. From the aspect of acceleration ratio, when the number of nodes is 2, 4, and 6, respectively, the corresponding acceleration ratio is 1.62, 2.34, 2.65.

Table 5 Time-consuming situation and speedup of SVM parallelization algorithm under different nodes.

Number of nodes	1	2	3	4	5	6
Time (s)	917	565	447	386	379	340
Speedup ratio	1	1.52	1.93	2.24	2.41	2.55

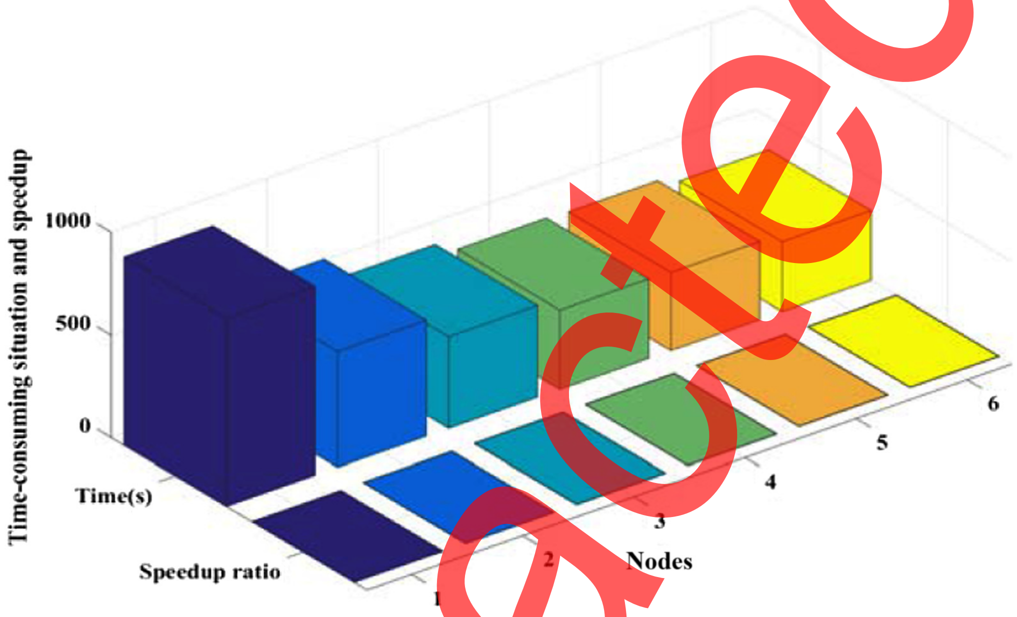

Fig. 9 Time-consuming situation and speedup of SVM parallelization algorithm under different nodes.

5 Conclusions

(1) In this paper, a high-performance parallel computing processing platform based on big data environment is constructed, and high-performance parallelism of SVM classification algorithm is implemented under this platform. Experiments prove that the high-performance parallel computing framework for big data proposed in this paper is feasible and reliable. Nested GPUs in MPI multiprocesses are an efficient hybrid parallel mode. Although the classification accuracy of remote sensing images is degraded, it does improve the computational efficiency and reduce the computation time.

(2) This paper studies the two parallel computing technologies based on message passing interface MPI and CUDA. Aiming at the characteristics of these two parallel technologies, MPI is a coarse-grained parallel programming model, and CUDA-based GPU parallelism is typically fine-grained. The parallel programming model proposes a hybrid parallel mode in which CUDA is nested in MPI multiprocess in parallel.

(3) This paper analyzes the calculation time of the SVM classification algorithm under different experimental scenarios. Experiments show that in the single-node MPI-GPU environment, although the accuracy of remote sensing image classification gradually decreases with the increase of the number of secondary processes, when the number of secondary processes exceeds 8, the classification accuracy of SVMs tends to be stable. With the increase of the number of nodes, the calculation time of the SVM classification algorithm gradually decreases, and the acceleration ratio gradually increases.

Acknowledgments

This work was supported by Chongqing Teaching Science Planning Project (Research on Fast Recognition of Online Learning Expressions-2020-GX-398) and Yongchuan District Natural Science Foundation Project (Research on Personalized Video Recommendation for Online Vocational Education-2021yc-jckx20025). All authors contributed equally to this work.

References

1. X. Meng, J. Bradley, and B. Yavuz, "MLlib: machine learning in Apache spark," *J. Mach. Learn. Res.* **17**(1), 1235–1241 (2015).
2. N. A. Shrivastava, A. Khosravi, and B. K. Panigrahi, "Prediction interval estimation of electricity prices using PSO-tuned support vector machines," *IEEE Trans. Ind. Inf.* **11**(2), 322–331 (2015).
3. G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: benchmark and state of the art," *Proc. IEEE* **105**(10), 1865–1883 (2017).
4. M. Zaharia, R. S. Xin, and P. Wendell, "Apache spark: a unified engine for big data processing," *Commun. ACM* **59**(11), 56–65 (2016).
5. H. Lu et al., "A survey of semantic construction and application of satellite remote sensing images and data," *J. Organizational End User Comput.* **33**(6), 1–20 (2021).
6. X. Wang et al., "Remote sensing monitoring method based on BDS-based maritime joint positioning model," *Comput. Model. Eng. Sci.* **127**(2), 801–818 (2021).
7. S. Gopalani et al., "Comparing Apache spark and map reduce with performance analysis using K-means," *Int. J. Comput. Appl.* **113**(1), 8–11 (2015).
8. J. Merlin, B. A. Evans, and N. Dehvari, "Could burning fat start with a brite spark? Pharmacological and nutritional ways to promote thermogenesis," *Mol. Nutrition Food Res.* **60**(1), 18–42 (2015).
9. E. Govea-Alcaide, I. F. Machado, and R. F. Jardim, "10 to 25-fold increase in the transport superconducting critical current density of spark-plasma sintered Bi-2223 superconductors," *J. Appl. Phys.* **117**(4), 043903 (2015).
10. R. Xin et al., "Scaling spark in the real world: performance and usability," *Proc. VLDB Endowment* **8**(12), 1840–1843 (2015).
11. D. T. Bui et al., "Spatial prediction models for shallow landslide hazards: a comparative assessment of the efficacy of support vector machines, artificial neural networks, kernel logistic regression, and logistic model tree," *Landslides* **13**(2), 361–378 (2016).
12. H. Hong, B. Pradhan, and M. N. Jebur, "Spatial prediction of landslide hazard at the Luxi area (China) using support vector machines," *Environ. Earth Sci.* **75**(1), 40 (2016).
13. A. Abdiansah and R. Wardoyo, "Time complexity analysis of support vector machines (SVM) in LibSVM," *Int. J. Comput. Appl.* **128**(3), 28–34 (2015).
14. Y. You, J. Demmel, and R. Vuduc, "Design and implementation of a communication-optimal classifier for distributed kernel support vector machines," *IEEE Trans. Parallel Distrib. Syst.* **28**(4), 974–988 (2016).
15. M. Abdolmaleky et al., "Red-green-blue multi-channel quantum representation of digital images," *Optik* **128**, 121–132 (2017).
16. P. F. Shan, "Image segmentation method based on K-mean algorithm," *EURASIP J. Image Video Process.* **2018**, 81 (2018).
17. P. Shan and X. Lai, "Influence of CT scanning parameters on rock and soil images," *J. Vis. Commun. Image Represent.* **58**(1), 642–650 (2019).
18. T. Tanino, R. Kawachi, and M. Akao, "Performance evaluation of multiobjective multiclass support vector machines maximizing geometric margins," *Numer. Algebra Control Optim.* **1**(1), 151–169 (2017).
19. S. Guo et al., "Sensor multi-fault diagnosis with improved support vector machines," *IEEE Trans. Autom. Sci. Eng.* **14**(2), 797–808 (2017).
20. J.-Y. Gotoh and S. Uryasev, "Support vector machines based on convex risk functions and general norms," *Ann. Oper. Res.* **249**(1–2), 1–28 (2017).

21. K. Tan, J. Zhang, and Q. Du, "GPU parallel implementation of support vector machines for hyperspectral image classification," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **8**(10), 4647–4656 (2015).
22. T. Singh, F. Di Troia, and V. A. Corrado, "Support vector machines and malware detection," *J. Comput. Virol. Hack. Tech.* **41**(10), 1–10 (2016).
23. B. Demir and L. Bruzzone, "Hashing-based scalable remote sensing image search and retrieval in large archives," *IEEE Trans. Geosci. Remote Sens.* **54**(2), 892–904 (2015).
24. Ö. Gürsoy and Ş. Kaya, "Detecting of lithological units by using terrestrial spectral data and remote sensing image," *J. Indian Soc. Remote Sens.* **45**(2), 259–269 (2017).
25. S. Rajendran et al., "MapReduce-based big data classification model using feature subset selection and hyperparameter tuned deep belief network," *Sci. Rep.* **11**, 24138 (2021).
26. O. I. Khalaf and G. M. Abdulsahib, "Optimized dynamic storage of data (ODSD) in IoT based on blockchain for wireless sensor networks," *Peer-to-Peer Netw. Appl.* **14**(5), 2858–2873 (2021).
27. L. Huanjun et al., "Soil organic matter content inversion model with remote sensing image in field scale of blacksoil area," *Trans. Chin. Soc. Agric. Eng.* **34**(1), 127–133 (2018).
28. L. Zhang, A. Li, and Z. Zhang, "Global and local saliency analysis for the extraction of residential areas in high-spatial-resolution remote sensing image," *IEEE Trans. Geosci. Remote Sens.* **54**(7), 3750–3763 (2016).
29. C. Liu, L. Hong, and J. Chen, "Fusion of pixel-based and multi-scale region-based features for the classification of high-resolution remote sensing image," *J. Remote Sens.* **19**(2), 228–239 (2015).
30. S. Liu, Y. Zhu, and L. Xue, "Remote sensing image super-resolution method using sparse representation and classified texture patches," *Geomat. Inf. Sci. Wuhan Univ.* **40**(5), 578–582 (2015).
31. L. Jia, M. Li, and P. Zhang, "Remote-sensing image change detection with fusion of multiple wavelet kernels," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **9**(8), 3405–3418 (2016).
32. L. Ma, B. Du, and H. Chen, "Region-of-interest detection via superpixel-to-pixel saliency analysis for remote sensing image," *IEEE Geosci. Remote Sens. Lett.* **13**(12), 1752–1756 (2016).

Li Liao received her Master of Engineering from the University of Electronic Science and Technology in China. Now, she works in Chongqing City Vocational College. Her research interests include artificial intelligence, data mining, and computer vision.