Computers and Geosciences 127 (2019) 111–122

Contents lists available at ScienceDirect

Computers and Geosciences

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

Design and implementation of a parallel geographically weighted *k*-nearest neighbor classifier

Yingxia Pu[a,b,c,*], Xinyi Zhao[a], Guangqing Chi[d], Shuhe Zhao[a,b,c], Jiechen Wang[a,b,c], Zhibin Jin[e], Junjun Yin[f]

[a] *School of Geography and Ocean Science, Nanjing University, Nanjing, 210023, China*
[b] *Jiangsu Provincial Key Laboratory of Geographic Information Science and Technology, Nanjing, 210023, China*
[c] *Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing, 210023, China*
[d] *Department of Agricultural Economics, Sociology, and Education, Population Research Institute, and Social Science Research Institute, Pennsylvania State University, University Park, PA, 16802, USA*
[e] *Nanjing Haixing Power Technology Co. Ltd, Nanjing, 211100, China*
[f] *Social Science Research Institute and Population Research Institute, Pennsylvania State University, University Park, PA, 16802, USA*

A R T I C L E I N F O

Keywords:
Remotely sensed image classification
Parallel computing
Geostatistical models
Parallel gw*k*-NN classifier

A B S T R A C T

The development of high-performance classifiers represents an important step in improving the timeliness of remote sensing classification in the era of high spatial resolution. The geographically weighted *k*-nearest neighbors (gw*k*-NN) classifier, which incorporates spatial information into the traditional *k*-NN classifier, has demonstrated better performance in mitigating salt-and-pepper noise and misclassification. However, the integration of spatial dependence into spectral information is computationally intensive. To improve the computing performance of the gw*k*-NN classifier, this study first considered two commonly used parallel strategies—data parallelism and task parallelism—in the model training and image classification stages. Then, our implementation of the corresponding parallel algorithms was carried out by calling message passing interface (MPI) and the geospatial data abstraction library (GDAL) in the C+ + development environment on a standalone eight-core computer. Based on the performance of these two strategies, the potentiality of dual parallelism (the simultaneous exploitation of data and task parallelism) in image classification was further investigated. Our experimental results indicate that the parallel gw*k*-NN classifier can improve the classification efficiency of high-resolution remote sensing images with multiple land cover types. Specifically, the data parallelism method is more effective than the task parallelism method in both the model training and classification stages because of the minor effect of parallel overhead on the total execution time. In addition, dual parallelism can take advantage of data and task parallel strategies, as evidenced by the two largest speedups being attained under dual parallelism I (5.28 ×), which is based on the premise of task parallelism, and dual parallelism II (5.73 ×), in which the priority is given to data decomposition. Comparatively, dual parallelism II provides the best performance by overlapping computation and data transmission, which is compatible with the current trend toward multicore architectures.

1. Introduction

Advancement in sensor technology has led to remote sensing applications with extremely high computational requirements for their implementation; for example, time-critical, large-scale environmental monitoring and disaster assessment applications (Chen et al., 2015; Ma et al., 2015; Plaza and Chang, 2008; Qin et al., 2014; van Zyl, 2012). Significant efforts have recently been directed towards the incorporation of high-performance computing (HPC) technologies, such as parallel and distributed computing, into image data acquisition, classification, and visualization (Choi and Lee, 2013; Lee et al., 2011; Li et al., 2017; Plaza, 2009; Plaza et al., 2011a, 2011b). Most of the widely used classification methods (e.g., maximum likelihood classifier (MLC), *k*-nearest neighbor (*k*-NN), and support vector machine (SVM)), have been parallelized on various HPC platforms, including massively parallel multiprocessors, specialized field programmable gate arrays (FPGAs), and graphic processing units (GPUs) (Caccetta et al., 1996; Choi and Lee, 2013; Gualtieri, 2005; Jóźwik et al., 1998; Kubota et al.,

* Corresponding author. School of Geography and Ocean Science, Nanjing University, Nanjing, 210023, China.
E-mail address: yingxiapu@nju.edu.cn (Y. Pu).

https://doi.org/10.1016/j.cageo.2019.02.009
Received 28 October 2016; Received in revised form 21 February 2019; Accepted 28 February 2019
Available online 05 March 2019
0098-3004/ © 2019 Elsevier Ltd. All rights reserved.

2000; Li et al., 2017; Plaza and Chang, 2008). Despite increases in the spatial resolution of sensors, however, the accuracy of spectra-based land cover classification can be compromised by the so-called salt-and-pepper effect, as higher-resolution images are likely to have higher within-class variability (Atkinson and Naser, 2010; Chen et al., 2016; Hay et al., 1996; Kettig and Landgrebe, 1976; Li et al., 2014; Myint et al., 2011).

A number of spatio-contextual analysis techniques including texture extraction, Markov random fields (MRFs) modeling, image segmentation, and object-based image analysis have been developed to improve classification accuracy and reduce the salt-and-pepper effect in many application areas (Andekah et al., 2017; Atkinson, 2004; Blaschke, 2010; Chen et al., 2016; De Jong and van der Meer, 2004; Johnson et al., 2012; Moser et al., 2013; Qiao et al., 2011, 2015; Stuckens et al., 2000; Tso and Mather, 1999; Wang et al., 2016). Although such applications originated as computer vision, pattern recognition, and image analysis techniques, these attempts have been directed to address the problem of spatial dependence (Li et al., 2014). Geostatistics, a set of procedures for estimating the local values of properties that vary across geographic space from sample data, has been widely used to characterize inherent spatial variation among pixels since the 1990s (Atkinson and Lewis, 2000; Comber et al., 2016; Curran, 1988; Curran and Atkinson, 1998; Dungan, 1998; Herzfeld and Higginson, 1996; Ramstein and Raffy, 1989; Woodcock et al., 1998a, 1998b). In particular, best fitted geostatistical models can outperform the direct use of sample variograms in measuring texture in remotely sensed classification. Atkinson and Naser (2010) accordingly developed a geographically weighted *k*-NN (gw*k*-NN) classifier that combined geostatistical models with a supervised *k*-NN classifier, which resulted in a significant increase in accuracy while eliminating the salt-and-pepper effect. Despite these advantages, techniques that use spatio-contextual information also add computational complexity, which creates new challenges in the use of HPC technologies in remote sensing. With the goal of accelerating the computational performance of high-resolution remote sensing classification, in this study we examined the application of HPC technologies in designing and implementing a parallel version of the gw*k*-NN classifier.

A parallel algorithm is of crucial importance for a gw*k*-NN classifier to fully exploit advanced computing environments and tools. In the domain of parallel computing, the methods of data parallelism and task parallelism—in which data partitioning (decomposition) and task grouping are respectively applied—are the most commonly used parallel strategies for increasing computational speed (Nicolescu and Jonker, 2002; Ramaswamy et al., 1997; Subhlok and Vondran, 2000). At its core, data parallelism involves performing similar operations on different partitioned datasets simultaneously. Because all datasets are processed in a parallel and independent manner, data parallelism is an example of efficient peer-to-peer parallelism. Under task parallelism, which is also known as *control* or *process parallelism*, various types of processes are defined for carrying out different tasks on a given dataset using a "divide and conquer" strategy. In this manner, the sub-results produced by the respective computing processes must be transmitted to the master process for integrating into a final result in a master-slave configuration (Chen et al., 2015; Nicolescu and Jonker, 2002). As one of the most widely used parallel computing libraries, MPI has been around since the early 1990s and spurred the development of the parallel software industry as well as large-scale parallel applications. MPI is suitable for cluster environments, and is also useful in single node environments with multiple many-core CPUs (Kinghorn, 2015; Lusk and Gropp, 1995). In this study, we implemented the proposed parallel gw*k*-NN classifier in the C++ programming language using calls to MPI and GDAL. The data parallelism and task parallelism methods have their own characteristics in our parallel gw*k*-NN classifier; the appropriate design and implementation of parallel algorithms could effectively decrease the computation time during the entire classification.

In addition to the two conventional parallel strategies, we further employed a dual parallelism method to improve the parallel performance by combining data parallelism and task parallelism and synthesizing their respective characteristics during the image classification stage. Ramaswamy et al. (1997) were the first to propose a framework for exploiting data parallelism and task parallelism simultaneously on distributed memory machines, and they obtained speedup improvements. However, the combination of data parallelism and task parallelism can be rather complex, and choosing an optimal strategy to make better use of computing resources still needs to be explored in parallelizing the gwk-NN classifier. The purpose of this research is to consider the general parallelization of a gw*k*-NN classifier based on different parallel strategies in the model training and classification stages, and further evaluate their parallel performance based on a number of evaluation criteria.

The remainder of this paper is organized as follows. Section 2 outlines in detail the design of the parallel gw*k*-NN classifier for model training and image classification. Section 3 provides an example of land cover classification, by applying the parallel gw*k*-NN classifier to a SPOT-5 image of Nanjing, China, to highlight the performance of the proposed techniques. In Section 4, we discuss the effects of different parallel strategies and various influencing factors on the performance of the parallel gw*k*-NN classifier. Finally, Section 5 presents conclusions that can serve as guidelines for future improvements.

2. Methods

2.1. The gwk-NN classifier

Atkinson and Naser (2010) presented a gw*k*-NN classifier based on the first law of geography. In their classifier, spatial information is incorporated into the model training and image classification stages by replacing the conditional probability in conventional *k*-NN classifiers with fitted geostatistical models that describe the spatial distribution features of ground objects. The specific algorithm is summarized in the following subsections.

2.1.1. Training geostatistical models

1) *Sample acquisition.* Given *K* land cover classes of the study area determined in advance, the *K* datasets of training samples are acquired, with each dataset containing multiple image pixels with spatial location and spectral information as well as their land cover class.
2) *Calculation of class-conditional probability.* The probability $p_{k,k}(\boldsymbol{h})$ of a pixel *i* attaching to land cover class *k*, given any pixel *j* at a distance lag $\boldsymbol{h}$ falling into the same class *k*, is calculated by

$$p_{k,k}(\boldsymbol{h}) = \frac{\sum_{i=1}^{N} I(z_{\boldsymbol{h}} = k|z_i = k)}{\sum_{i=1}^{N} I(z_i = k)}, \tag{1}$$

where $z_{\boldsymbol{h}}$ represents the class of pixel *j*, which is at lag $\boldsymbol{h}$ from pixel *i*. If pixels *i* and *j* both belong to class *k*, the indicator function $I(z_{\boldsymbol{h}} = k|z_i = k)$ takes the value of one; otherwise, it takes the value of zero. *N* is the total number of training samples; correspondingly, $\sum_{i=1}^{N} I(z_i = k)$ is the number of samples in class *k*. By adjusting the step length, the lag $\boldsymbol{h}$ can traverse the entire image.

3) *Model fitting.* For each class *k*, different geostatistical models are employed to fit the class-conditional probability $p_{k,k}(\boldsymbol{h})$, with the optimal one chosen in terms of the minimized residual sum of squares. In this study, exponential and Gaussian models were selected as candidate geostatistical models. The exponential model used to represent $p_{k,k}(\boldsymbol{h})$ is given as follows:

$$p_{k,k}(\theta, \boldsymbol{h}) = 1 - c\{1 - exp(-\boldsymbol{h}/r)\}, \tag{2}$$

where $\boldsymbol{h}$ is the same as described above, *c* is the partial sill

$(0 < c < 1)$, r is the distance parameter of the exponential model, and $\theta = \{c, r\}$ is the set of model parameters. The effective range of the exponential model, within which 95% of the variation can be found, is $3r$.

The Gaussian model used to represent $p_{k,k}(\boldsymbol{h})$ is given by the following:

$$p_{k,k}(\theta, \boldsymbol{h}) = 1 - c\{1 - exp(-3(\boldsymbol{h}/a)^2)\}, \tag{3}$$

where $\boldsymbol{h}$ and c are the same as described above, a is the range of the Gaussian model, and $\theta = \{c, a\}$ is the set of model parameters.

2.1.2. Classifying image pixels

1) *Data preparation.* All pixels of the remotely sensed images and relevant information, including the number of bands, image coordinates of the starting point, and spatial resolution are extracted for classification.
2) *Pixel classification.* A gw*k*-NN classifier considering spatial distribution and dependence is produced using the optimal model for each land cover class acquired in the model training stage. This results in a total of K gw*k*-NN classifiers corresponding to K land cover classes. The probability that any pixel i belongs to class k is as follows:

$$p(z_i = k) = \frac{\sum_{j=1}^{J} p_{k,k}(\boldsymbol{h})\cdot\omega_{ij}\cdot I(z_j = k)}{\sum_{k'=1}^{K}\sum_{j=1}^{J} p_{k,k'}(\boldsymbol{h})\cdot\omega_{ij}\cdot I(z_j = k')}, \quad \omega_{ij} = \frac{1}{d_{ij}^{p}}, \tag{4}$$

where $p_{k,k}(\boldsymbol{h})$ is the geographical weighting function of the selected geostatistical model for class k; ω_{ij} is a weight corresponding to the distance d_{ij} between pixels i and j ($j = 1, 2, \ldots, J$), in which p is a user-defined parameter (here, $p = 2$); $I(z_j = k)$ denotes the membership degree of the neighboring pixel j belonging to class k, as discussed previously (Maselli et al., 2005); and k' is an additional class index similar to k in all K classes. The final land cover class allocated to the object pixel i is the maximum of $p(z_i = k)$ from K classes produced by the gw*k*-NN classifier.

3) *Result writing.* The classification results of the individual pixels are written to a hard disk and their respective classes are used to draw the final classification map.

2.2. Parallel design

Because the model training of the gw*k*-NN classifier is independent of the subsequent image classification, both stages can be parallelized with different parallel strategies to optimize the overall classification efficiency.

2.2.1. Data parallelism and task parallelism for model training

In fitting K classes of sample data with M models, it is necessary to carry out $M \times K$ training tasks to select the optimal model for each class; in the case of sequential execution, it may take a considerable amount of time. A flowchart of training different geostatistical models for different land cover classes is shown in Fig. 1.

The data parallelism method is suitable for the model training stage, as each class is independently trained by different geostatistical models. A flowchart for this strategy is presented in Fig. 2(a). In this stage, n activated parallel processes access sample datasets corresponding to different land cover classes by simultaneously reading metadata information from the data source. Each process then independently trains the accessed dataset with candidate geostatistical models in sequence and selects the optimal model for each unique land cover class. That is, different processes take charge of different sample datasets, while the geostatistical models to be employed are identical. Moreover, if n is less than K, the process that completes the computation of the assigned class first will continue to access and train the remaining classes until all are traversed.

The task parallelism method can also be applied to train different geostatistical models with different processes. As the models do not need to communicate with each other, it is not necessary to determine an order of precedence relationship among the geostatistical models. Accordingly, each of the n processes simultaneously undertakes to train one or more models for all the classes, depending on the size relationship between the number of activated parallel processes n and that of geostatistical models M, similar to the data parallelism method introduced previously. The training results produced by the respective computing processes are then sent to the master process (process 0) to determine the optimal fitting model for each class, as shown in Fig. 2(b). In the case of sufficient computing processes, different models are trained by all the datasets in parallel. This procedure contrasts with the process allocation under data parallelism, in which the datasets of different classes for training all the models are performed in parallel while the models are trained by each process in a sequential manner.

2.2.2. Data parallelism and task parallelism for image classification

After finishing model training for each land cover class, the chosen optimal models are further combined with *k*-NN classifiers to calculate the probabilities of each pixel within an image belonging to each land cover class.

During the classification stage, the data parallelism method partitions the whole image into multiple datasets and assigns them to different processes for classification in parallel. That is, all processes in charge of multiple gw*k*-NN classifiers synchronously execute classification on different partitioned datasets. The classification results produced by each process are then integrated into a final result for the entire image. The image classification under data parallelism is shown schematically in Fig. 3.

The task parallelism method is also suitable for image classification, because gw*k*-NN classifiers of different classes are independent. Similar to the model training, the number of classifiers managed by each computing process depends on the size relationship between the number of processes (n) and that of gw*k*-NN classifiers (K). If the number of processes is sufficient to execute all of the classifiers ($n > K$), the respective processes will classify the entire image by different classifiers simultaneously; otherwise, some processes will execute multiple classifiers sequentially (Fig. 4). On receiving the sub-results from each of the computing processes, the master process (process 0) produces the final classification result. In theory, the task parallelism strategy can exponentially reduce classification time.

3. Experimental study

3.1. Study area and data preparation

The study area is located in the eastern part of Bagua Zhou Island in Nanjing, Jiangsu province, China (Fig. 5). This island is a sandbar silted by the Yangtze River and situated 4 km downstream of the Nanjing Yangtze River Bridge. It is geographically separated from the Nanjing urban area to the south and the Luhe District to the north by the Yangtze River. The ground objects in this area are typical and have a high spatial continuity, which makes it easy to collect sample data and to verify the land cover classes.

Remotely sensed data used in the experiment were extracted from the fused SPOT-5 image in 2002 (Fig. 6). In this study, Brovey transform, a well-known color-normalized fusion technique, was applied to produce a higher spatial resolution image by merging low-resolution multispectral (MS) bands with a higher spatial resolution panchromatic (PAN) band (Gharbia et al., 2014; Pohl and Genderen, 2016; Vrabel, 1996). Here, each SPOT-5 MS band with a 10 m resolution was multiplied by a ratio of a PAN band with a 2.5 m resolution divided by the sum of the MS bands. The output PAN-sharpening image has the spatial resolution of the PAN image and the spectral characteristics of the MS image, which is more useful for image classification and other

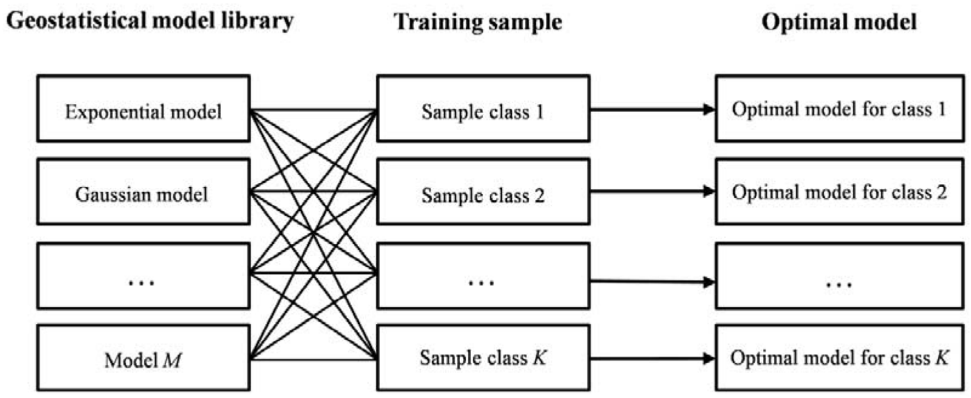

Fig. 1. Schematic of the model training stage.

applications. The size of the fused SPOT-5 image was 321 rows × 361 columns. Based on visual interpretation of the SPOT image and an additional on-the-spot field survey, the land cover in the study area was divided into five classes: bare land, farmland, grassland, road, and water. Prior to image classification, 330 training samples were selected to fit geostatistical models for different land cover classes and 205 test samples were subsequently used to assess the classification accuracy of the parallel gw*k*-NN classifier (Table 1).

The comparison of the spectral heterogeneities among different bands in the training sample data is shown in Fig. 7. Based on a discernible spectral variation within different classes and a relatively small spectral overlap, we selected the red and green bands for classification in the gw*k*-NN classifier.

3.2. Implementation of the parallel gwk-NN classifier

The parallel gw*k*-NN classifier was implemented in the C++ programming language using calls to MPI and GDAL on a standalone Intel Xeon E5-2620V4 eight-core computer. MPI as a portable message-passing standard is widely used in parallel computing architectures and implementation of parallel programs (Lusk and Gropp, 1995; MPICH, 2012). GDAL is a translator library for raster and vector geospatial data formats that supports the acquisition of the raster data source and information about its driver, projection, and resolution, as well as other relevant information (GDAL, 2015). In our experiment, MPICH2 1.5, a high-performance and widely portable implementation of the MPI standard, and GDAL 1.11.2 were used to implement parallelization of the gw*k*-NN classifier.

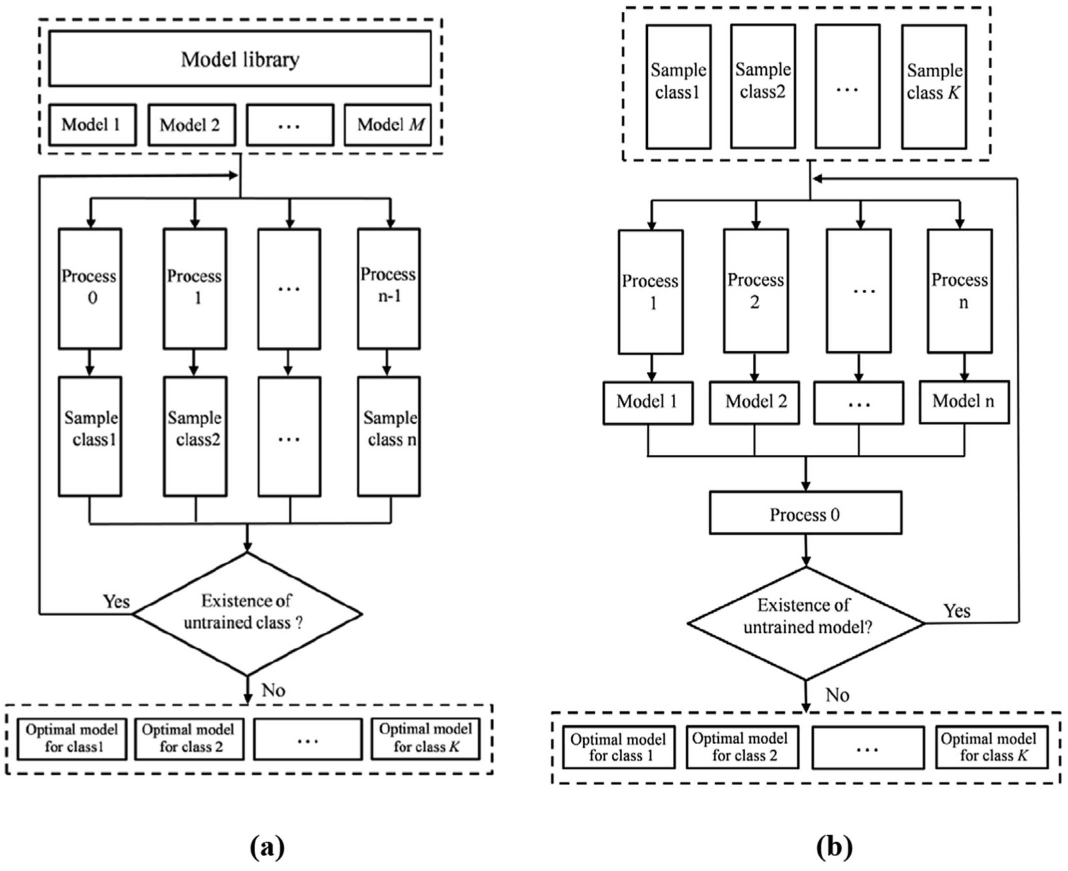

Fig. 2. Flowchart of different parallel strategies in the model training stage: (a) data parallelism and (b) task parallelism.

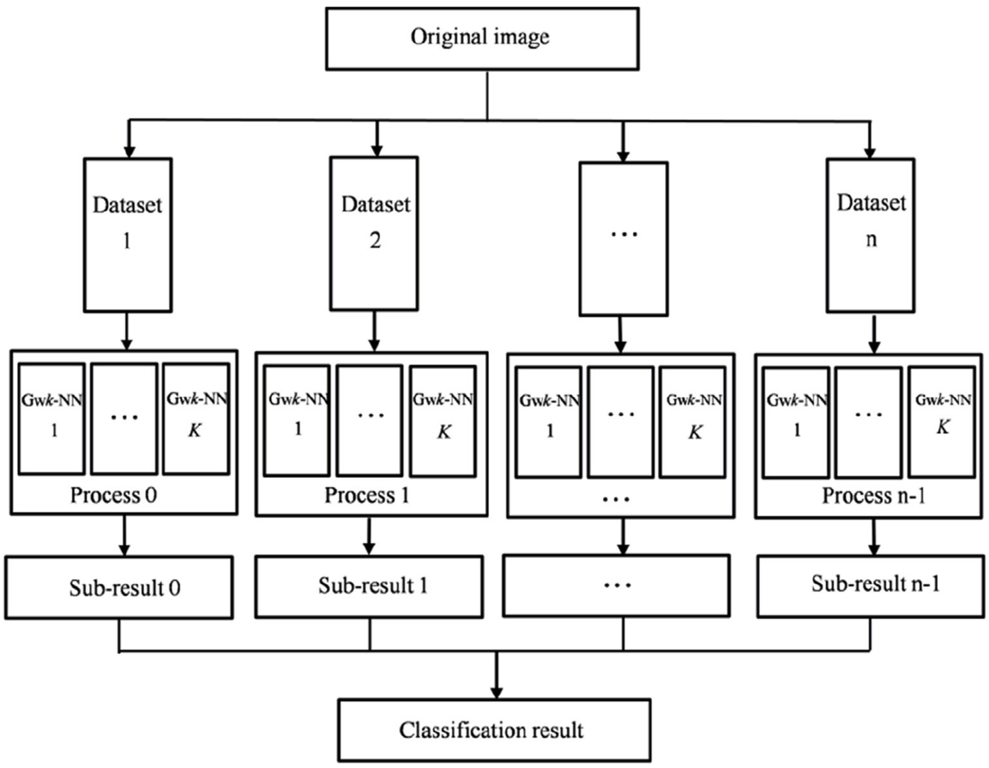

Fig. 3. Flowchart of data parallelism in the image classification stage.

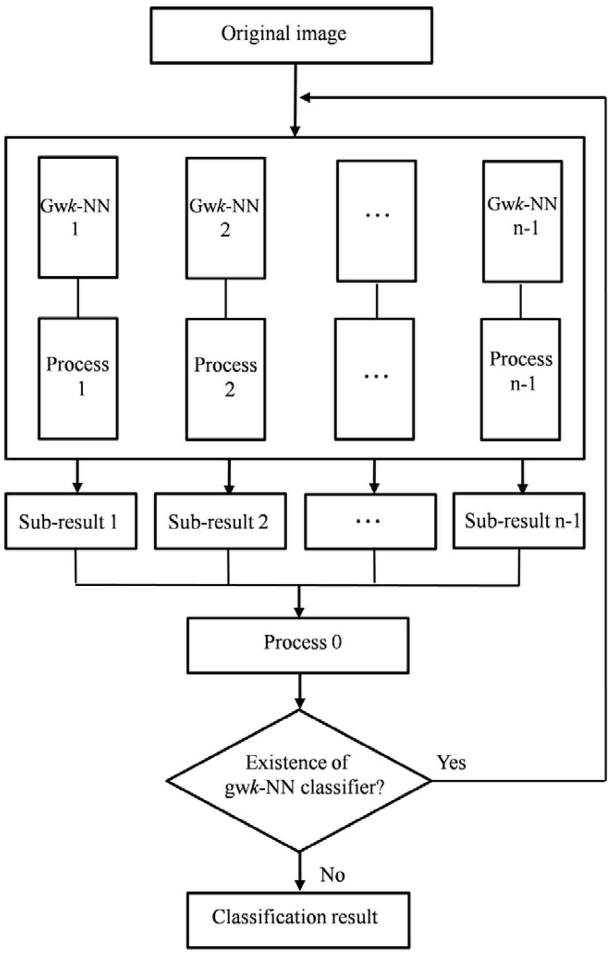

Fig. 4. Flowchart of task parallelism in the image classification stage.

3.2.1. *Parallel implementation in the model training stage*

The pseudocode for fitting geostatistical models in the data and task parallel schemes is shown in Table 2. K is the number of land cover classes, M represents the number of geostatistical models, and np signifies the number of processes executed in parallel. The functionality of the pseudocode corresponds to the corresponding descriptions given in Section 2.2.1.

Based on the empirical class-conditional probability plot for each land cover class, we selected exponential and Gaussian models as candidates to fit the spatial distribution features of the training samples. Other kinds of geostatistical models, such as spherical and power models, were excluded because of a mismatch between empirical probability plots and theoretical geostatistical models. Table 3 shows the selected optimal model for each land cover class with the corresponding parameter estimates; Fig. 8 further depicts the graph of the selected geostatistical models with their specific functions. The determination of the optimal model for each land cover class is not affected by the parallel strategies adopted, as the parallel strategies have no impact on the core classification scheme of the classifiers.

The exponential model is a much better fit to the spatial distribution features of bare land, grassland, and road; the Gaussian model is more suitable for characterizing farmland and water. Additionally, farmland extends over geographic space with a range of 16.5 m (the spatial resolution is 2.5 m) and shows the strongest spatial dependence among five land cover classes, with a remarkably high class-conditional probability of more than 50%. Comparatively, the class-conditional probability of water is rather low, generally less than 30%. Over the whole region, farmland covers the largest area with the widest spatial distribution, which also enhances its spatial correlation to a certain degree.

3.2.2. *Parallel implementation in the image classification stage*

The pseudocode for classifying an image in the data and task parallel schemes is shown in Table 4. K, M, and np are the same as mentioned previously. The functionality of the pseudocode relates to the

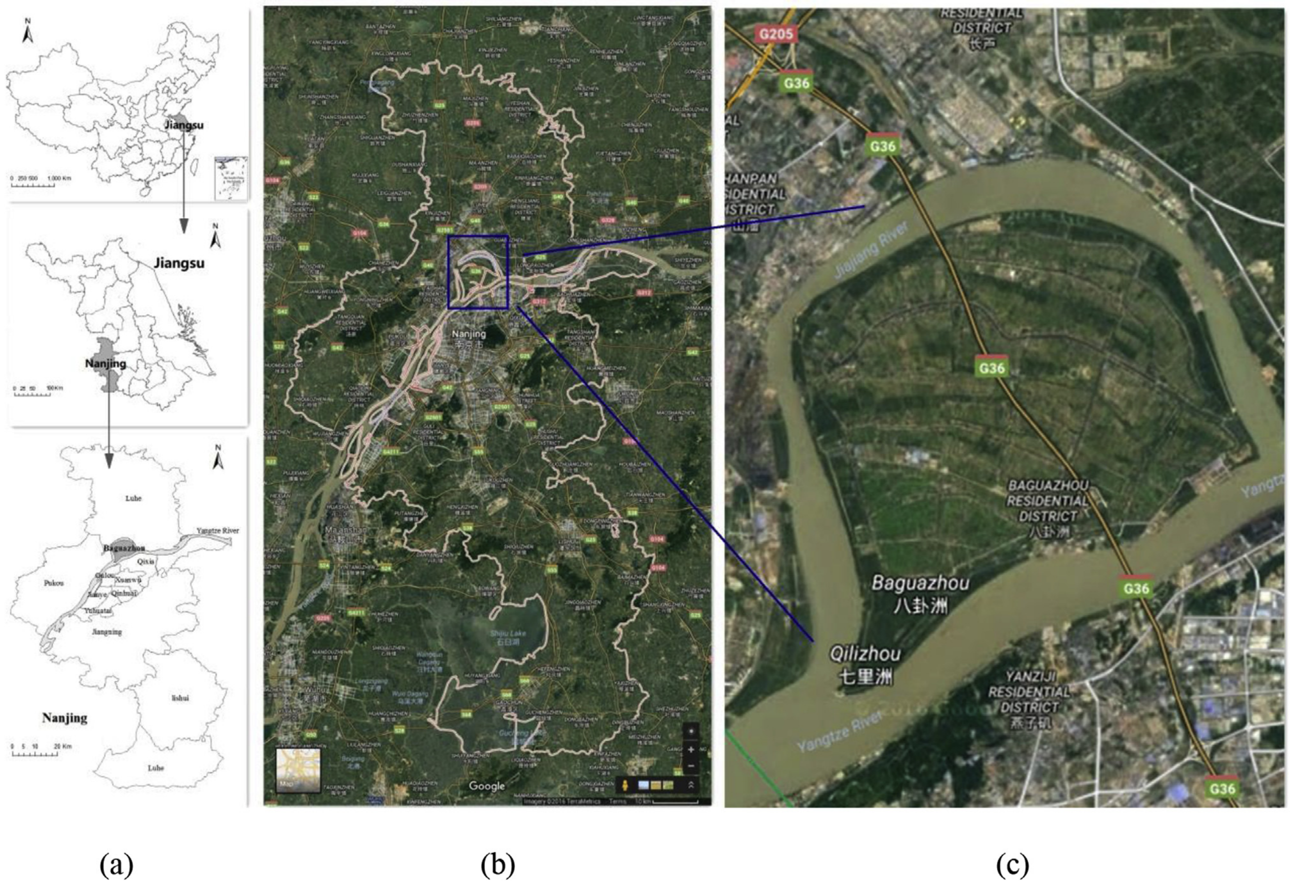

Fig. 5. Location map of the study area: (a) geographical locations at national, provincial, and city scales; (b) and (c) satellite images at local scales.

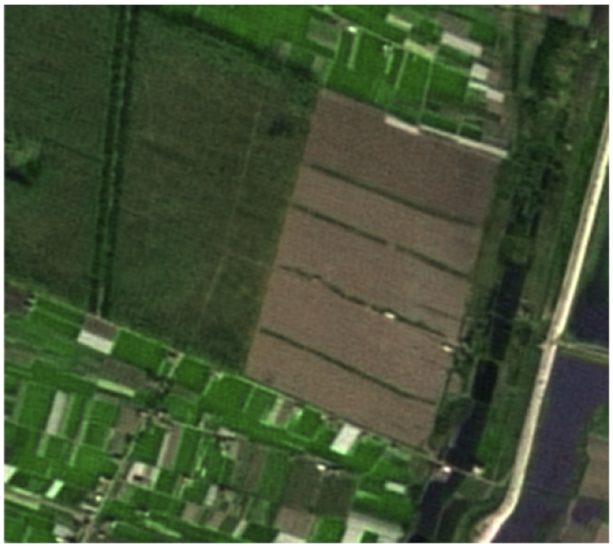

Fig. 6. SPOT-5 image of the study area.

Table 1
Training samples and test data.

Class code	Class name	Number of training samples	Number of test data
C1	Bare land	60	40
C2	Farmland	110	60
C3	Grassland	100	60
C4	Road	20	15
C5	Water	40	30

corresponding descriptions given in Section 2.2.2.

Based on different parallel schemes, the parallel gw*k*-NN classifiers were implemented to classify the whole image with the optimal model for each class. As a benchmark for comparison, we also ran the conventional *k*-NN classifier. Fig. 9 shows the classification results produced by these two classifiers. It is seen that the results from the gw*k*-NN approach are much smoother due to the consideration of geographical weights and spatial distribution features of the ground objects. In addition, the salt-and-pepper noise present in the conventional *k*-NN classification results is greatly alleviated.

The confusion matrix of the parallel gw*k*-NN classifier is summarized in Table 5. We can see that the overall accuracy and the accuracies of individual categories have been greatly improved by considering spatial dependence, and accordingly mitigating the problems of the same object producing varying spectra and different objects producing the same spectrum. Specifically, the overall accuracy for all land cover classes reaches 98.54%, and both the user's and producer's accuracy for water is up to 100%. However, the producer's accuracy for road class is the lowest (93.33%), which is as a result of one test sample being misclassified. The cause of this may be the relatively small coverage area and the small size of the training and test samples. Despite this limitation in the study area, the gw*k*-NN classifier is still adequate for land cover classification, as proven by the high classification accuracy.

3.3. *Performance evaluation of the parallel gwk-NN classifier*

To evaluate the performance of the proposed parallel gw*k*-NN classifier in a multicore computing environment, we took three criteria—specifically, running time, speedup, and efficiency—into account (Czech, 2016). Running time (T), or the worst-case parallel running time, is recorded from the start of executing an algorithm by the first

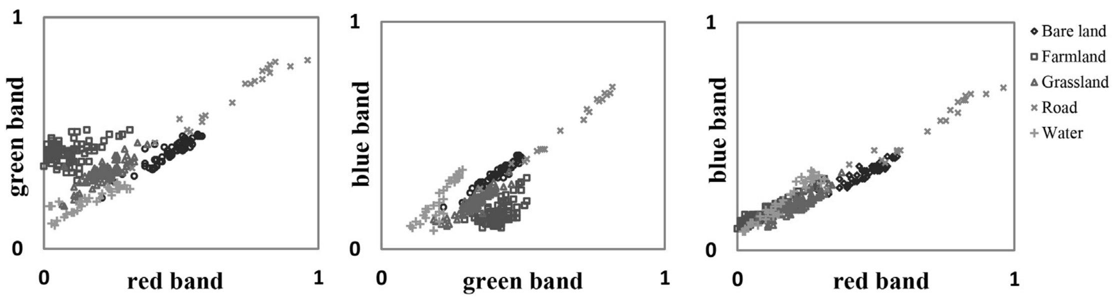

Fig. 7. Scatter plots of spectral heterogeneity of training samples (red, green, and blue bands). (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

Table 2
Data and task parallel implementation in the model training stage.

Data parallelism	Task parallelism
if $np \geq K$	**if** $np \geq M + 1$
for training samples of class i ($i = 0, 1, \ldots, K - 1$) **do**	**for** geostatistical model i ($i = 0, 1, \ldots, M - 1$) **do**
//The classes are fitted in parallel	//The models are called in parallel
$O_i \leftarrow$ Optimal model for class i from M models	$R_i \leftarrow$ Model i fitting to K classes by process $i + 1$
end for	*SendMessage*(R_i, 0)//Send results to process 0
else	**end for**
$i = 0, 1, \ldots, np - 1$	**else**
while $i \leq K - 1$ **do**	$i = 0, 1, \ldots, np - 2$
for training samples of class i **do**	$cp \leftarrow i + 1$//cp is the current process with model i
$O_i \leftarrow$ Optimal model for class i from M models	**while** $i \leq M - 1$ **do**
end for	**for** geostatistical model i **do**
$i \leftarrow i + np$//The class to be fitted in the next loop	$R_i \leftarrow$Model i fitting to K classes by process cp
end while	*SendMessage*(R_i, 0)
end if	**end for**
return optimal model set O	$i \leftarrow i + np - 1$//The models to be used in the next loop
	end while
	end if
	$R \leftarrow$ *ReceiveMessage*(R_i, *any*)//Process 0 receives results
	$O \leftarrow$ Gather optimal models for all classes from results R
	return optimal model set O

Table 3
Fitted parameters of the optimal models for different land cover classes.

Class code	Class name	Model	Sill (c)	Range
C1	Bare land	Exponential	0.6161	1.8000
C2	Farmland	Gaussian	0.4871	6.6000
C3	Grassland	Exponential	0.5891	3.6000
C4	Road	Exponential	0.6711	0.6000
C5	Water	Gaussian	0.7241	1.6000

process until the end of execution by all processes. Given the problem size, the less the time costs, the better the performance is. Speedup (S) measures the benefit achieved by parallel computation, defined as the ratio of the execution time of the best sequential algorithm with a single process to the execution time with a number of processes in a parallel computing environment. The speedup achieved is typically less than the number of processes because of unbalanced loads or computation overhead such as communication between different processors and synchronization of their operation. Efficiency (E), defined as the ratio of speedup (S) to the number of processors, measures processor utilization. The efficiency value of one means that none of the processes is idle (Grama et al., 1993; Orii, 2010).

Theoretically, the performance of the parallel gw*k*-NN classifier can be measured by overall speedup (the ratio of the total time of sequential execution involved in the model training and image classification to that of parallel execution), because both stages are independently executed. In practice, however, the overall speedup might not effectively reflect the improvement in performance because the execution time of the model training and classification differs significantly and there may be no comparability between them. Therefore, we present the parallel performance in the model training and classification stages, respectively. Moreover, to ensure that multiple processes can simultaneously read the different datasets to be allocated, each process first calculates the starting position in the corresponding dataset and then operates the reading task.

3.3.1. Model training

The specific parallel performance results under data parallelism and task parallelism in the model training stage are listed in Table 6.

Under data parallelism, the training time continuously decreases as the number of processes increases to four, which is close to but not equal to the total number of land cover classes (five). Ideally, optimal performance would occur when each process trains the corresponding sample class, i.e., all datasets are trained in parallel and the total training time is determined by the maximum elapsed time among the five processes. In practice, however, the training time of four processes is the shortest, meaning that the optimal allocation and scheduling are achieved in advance. As the number of processes increases, the idle processes cost CPU resources and affect the overall performance.

By contrast, the parallel performance under task parallelism does not conform to our prior expectations. If there are only two processes, one will engage in model training for the overall sample dataset while the other will serve as the master process. In this case, extra time might be required to transfer messages between these two processes, leading to a decrease in parallel efficiency. However, increasing the number of processes to three and then four results in a slight decrease in training time initially, then a significant increase (from four upward), which in some cases exceeds the sequential execution time. It is probably not

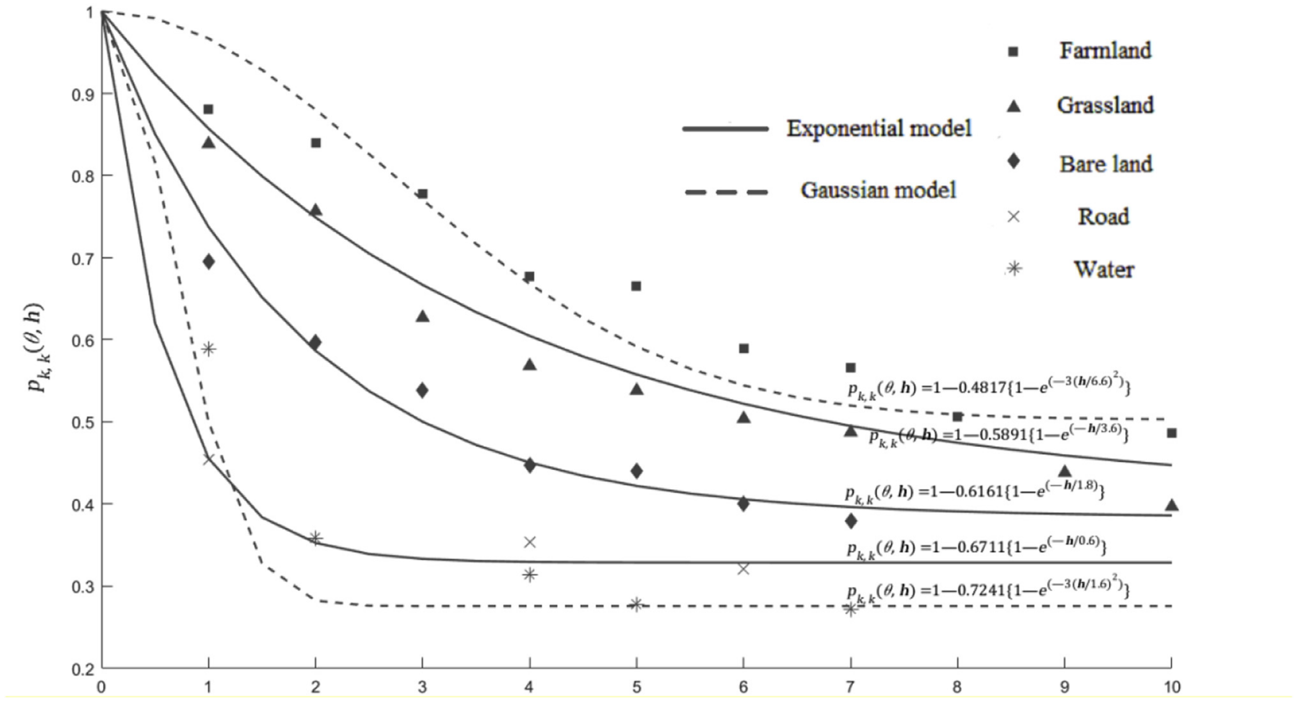

Fig. 8. Graph of the optimal models for different land cover classes.

necessary to involve additional processes to support the task parallelism method because only two geostatistical models are fitted in this stage. Thus, the data parallelism method is seen to be more suitable than the task parallelism method for training models in this study.

3.3.2. Image classification

After the model training, the data parallelism and task parallelism methods were applied to parallelize the gw*k*-NN classifiers during the image classification stage. Table 7 shows the corresponding classification time, speedups, and parallel efficiencies achieved using a limited number of processors under two parallel schemes. The classification time is significantly reduced as the number of processes increases, indicating that both strategies can play an effective role in this stage. However, it is noteworthy that the parallel performance under data parallelism is much better than that under task parallelism.

Specifically, under data parallelism, an image can be divided into multiple strip-sized datasets, each of which is then to be classified by

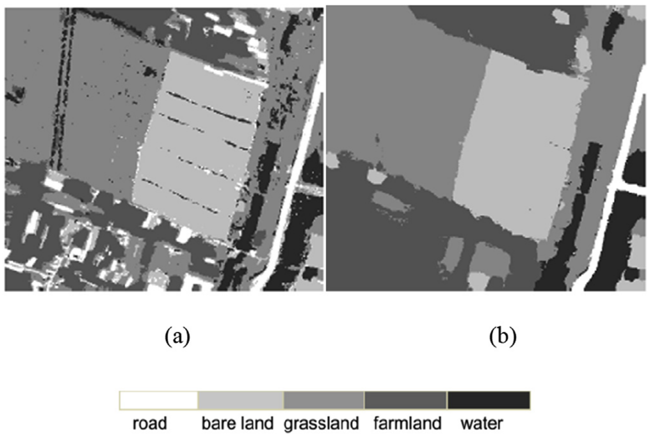

Fig. 9. Classification results: (a) *k*-NN classifier and (b) gw*k*-NN classifier.

Table 4
Data and task parallel implementation in the image classification stage.

Data parallelism	Task parallelism
//Divide the image into N datasets	**if** $np \geq K + 1$
for dataset i ($i = 0, 1, \ldots, np - 1$) **do**	**for** gw*k*-NN classifier i ($i = 0, 1, \ldots, K - 1$) **do**
//The datasets are classified in parallel	//The classifiers are employed in parallel
if $i\ != np - 1$	$S_i \leftarrow$Classify the image by classifier i (process $i + 1$)
//Calculate the number of pixels in divided direction	*SendMessage*(S_i, 0)//Send the sub result to process 0
$p\_ysize \leftarrow RasterYsize/N$	**end for**
$S_i \leftarrow$ Classify dataset i by K gw*k*-NN classifiers	**else**
else	$i = 0, 1, \ldots, np - 2$
$ysize \leftarrow RasterYsize/N * (N - 1)$	$cp \leftarrow i + 1$//cp is the current process for classifier i
$p\_ysize \leftarrow RasterYsize - ysize$	**while** $i \leq K - 1$ **do**
$S_i \leftarrow$ Classify dataset i by K gw*k*-NN classifiers	**for** gw*k*-NN classifier i **do**
end if	$S_i \leftarrow$ Classify the image by classifier i (process cp)
end for	*SendMessage*(S_i, 0)//Send the result to process 0
return final classification result S	**end for** $i \leftarrow i + np - 1$//The classifiers in the next loop
	end while
	end if
	$S \leftarrow$*ReceiveMessage*(S_i, *any*)//Process 0 receive the results
	return final classification result S

Table 5
Classification accuracies of the parallel gw*k*-NN classifiers.

Land cover classes	Classified results					Total
	Bare land	Farmland	Grassland	Road	Water	
Bare land	40	0	0	0	0	40
Farmland	0	59	1	0	0	60
Grassland	1	0	59	0	0	60
Road	0	0	1	14	0	15
Water	0	0	0	0	30	30
Total	41	59	61	14	30	205
User's accuracy (%)	97.56	100	96.72	100	100	
Producer's accuracy (%)	100	98.33	98.33	93.33	100	
Overall accuracy (%)	98.54			Kappa coefficient		0.9808

Table 6
Parallel performance under data and task parallelism in the model training stage.

No. of processes	Data parallelism			Task parallelism		
	Training time (s)	Speedup	Efficiency	Training time (s)	Speedup	Efficiency
1	97.40	1.00	1.00	93.00	1.00	1.00
2	77.10	1.26	0.63	94.23	0.99	0.50
3	56.14	1.73	0.58	86.36	1.08	0.36
4	41.54	2.34	0.59	87.46	1.06	0.27
5	45.65	2.13	0.43	94.81	0.98	0.20
6	49.37	1.97	0.33	102.78	0.90	0.15
7	51.23	1.90	0.27	109.90	0.85	0.12
8	51.44	1.89	0.24	112.15	0.83	0.10

one of different processes assigned in parallel. The size of each dataset decreases as the number of processes grows, thereby reducing the overall classification time. Although the speedup of the parallel gw*k*-NN classifier increases continuously, the parallel efficiency continues to decrease. In contrast, the classification time under task parallelism tends to first decline and then rise as the number of processes increases, with maximum parallel performance at five processes. It is worth noting that the computing processes used in this case are sufficient to attain an optimal schedule for five gw*k*-NN classifiers, as is the case in the model training stage under data parallelism. The shortest classification time is achieved with the highest speedup at five processes; beyond five processes more time is required to pass messages between master and slave processes, in turn decreasing the efficiency. Overall, the parallel efficiency first rises and then declines with fluctuation. The specific values of parallel efficiency under task parallelism are significantly smaller than those under data parallelism.

To take full advantage of both data and task parallel strategies, we implemented a dual parallelism scheme in the classification stage. The implementation of dual parallelism is more complicated than that of either pure data or task parallelism as it can entail various process allocation methods. Based on the preceding results, in which optimal performance under task parallelism was obtained with four computing processes, the processes under dual parallelism were divided into multiple groups of four processes when possible. When there were no more than four processes, the parallel mode would involve pure task parallelism, as the number of processes would then be insufficient to implement the data parallelism method. As the number of processes increased, they were grouped in advance to classify different partitioned datasets of the overall image in parallel. Notably, the process receiving the sub-results in each group would also execute the gw*k*-NN classifiers to avoid idle processes. For instance, six operating processes could be separated into two groups of four and two processes, respectively. These two groups would classify the two halves of an image independently. From a general perspective, different groups of processes in classifying separate datasets is an implementation of data parallelism in nature, while from a local perspective the classification of one dataset by a group of processes is an implementation of task parallelism. Table 8 shows the performance results produced by the parallel gw*k*-NN classifier under dual parallelism.

From the table, we can see that the dual parallelism method achieved further improvements in performance compared with the task parallelism method in the case of optimal performance with four computing processes, excluding the master process. In general, the total classification time under dual parallelism declined significantly as the number of processes increased. At the maximum number (eight) of processes in this experiment, the classification time was reduced to the lowest value (5.38 s). In addition, the parallel efficiency began to increase after five processes, and reached 0.66 with eight processes. Overall, the dual parallelism method is feasible and effective in image classification by making better use of multiple processes in sharing datasets and multiple gw*k*-NN classifiers.

4. Discussion

The number of processes, the computing environment, and the objects to be classified all have a marked impact on performance. Given a set of objects and the computing environment, the most suitable number of processes and optimal performance can be experimentally determined by using an enumeration method. The speedup under different parallel strategies is a function of the number of parallel processes, which is illustrated in Fig. 10. Among different strategies, the performance under dual parallelism varies greatly in terms of process schedules. Dual parallelism I is based on the premise of task parallelism, while dual parallelism II, which takes data decomposition as a prerequisite and then further implements task parallelism on different datasets, is also considered as an alternative strategy. To analyze the performance variation under different parallel strategies and their respective impacting factors, we further calculated the Karp-Flatt metric,

Table 7
Parallel performance under data and task parallelism in the image classification stage.

No. of processes	Data parallelism			Task parallelism		
	Classification time (s)	Speedup	Efficiency	Classification time (s)	Speedup	Efficiency
1	28.73	1.00	1.00	28.68	1.00	1.00
2	14.36	2.00	1.00	29.44	0.97	0.49
3	10.18	2.82	0.94	17.78	1.61	0.54
4	8.26	3.47	0.87	13.07	2.19	0.55
5	7.35	3.90	0.78	11.38	2.52	0.50
6	6.52	4.41	0.74	11.54	2.48	0.41
7	5.85	4.91	0.70	13.18	2.17	0.31
8	5.54	5.19	0.65	14.05	2.04	0.26

Table 8
Parallel performance under dual parallelism in the image classification stage.

No. of processes	Classification time (s)	Speedup	Efficiency
1	28.38	1	1.00
2	18.42	1.54	0.77
3	14.13	2.01	0.67
4	11.08	2.56	0.64
5	13.62	2.08	0.42
6	10.17	2.79	0.47
7	7.34	3.87	0.55
8	5.38	5.28	0.66

an experimentally determined sequential fraction of the total operation that cannot be parallelized regardless of what is done (e.g., reading data, setting up calculations, control logic, and storing results). The smaller a serial fraction is, the better an algorithm could be recognized (Karp and Flatt, 1990).

Under data parallelism, the increase in speedup tends to flatten as the number of processes increases. With a smaller number of processes, the data partitioning significantly reduces the size of the dataset allocated to each process, resulting in a rapid drop in time consumption. However, the benefit through parallel computation is further limited by insufficient degree of concurrency, computation overhead, or synchronization of operations. Finally, the parallel performance (speedup) stabilized at approximately five processes in the experiment. To minimize the cost of parallel computation, we must optimize the parallel algorithm by reducing redundant computations, avoiding idle processes, and/or overlapping communication and computation. Alternatively, as indicated by the Amdahl effect, we can increase the size of the image to keep speedup and efficiency growing with a fixed number of processes (Czech, 2016).

Under task parallelism, there is an initial rise in speedup followed by a decline. Peak performance occurs at five processes, which may be related to the number of land cover classes. The overall performance improvement under task parallelism is not as significant as that under data parallelism, with a peak speedup of only half that achieved in the latter case. As the number of processes surpasses five, the gap between data parallelism and task parallelism begins to expand. The reason for the unsatisfactory performance under task parallelism may be attributed to frequent communication between master and slave processes, as indicated by the relatively higher Karp-Flatt metric.

The dual parallelism method can further improve performance to a great extent by combining data and task parallel strategies. Under dual parallelism I, priority is given to the assignment of processes to different classifiers owing to the fixed number of land cover classes. As such, the key to dual parallelism I is adding processes to execute data parallelism on the premise that task parallelism is fulfilled, indicating that dual parallelism I indeed follows the task parallelism of four processes. The speedup pattern shown in Fig. 10 suggests that this type of parallel strategy further improves the performance of task parallelism, except in the case of five processes, in which one half of the image is classified sequentially by one process. In addition, the improvement in speedup beyond five processes is more significant than that under pure data parallelism. Overall, the dual parallelism I method is effective in improving the classification efficiency, characterized by much smaller serial fractions than those under pure task and even pure data parallel strategies.

Surprisingly, the dual parallelism II method achieves much better performance than the dual parallelism I method. When the number of processes is four or below, this method first assigns different processes to different partitioned datasets, which makes full use of the data parallel strategy; subsequently, more processes are employed to achieve task parallelism. We can see from Fig. 10 that the pattern of speedup under dual parallelism II is very similar to that of data parallelism prior to four processes, after which it is steady until the maximum of 5.73 achieved with eight processes. Additionally, this method is particularly effective in situations in which the task parallelism is exactly achieved by two processes in each dataset (*np* is equal to eight). The lower serial fraction ratio may provide an explanation for its better performance under dual parallelism II.

Finally, the hardware of the computing environment is of equal importance to parallel performance. In general, multicore processors support the execution of more processes parallel. If a processor cannot accommodate the parallel execution of a very large number of processes, it will result in the nonstop switching of current processes to satisfy the parallel requirement, which in turn affects overall efficiency. In addition, increasing the number of land cover classes requires the addition of gw*k*-NN classifiers, which demands that more processes be executed in parallel. Moreover, the optimal number of processes for achieving peak performance will also be affected by the characteristics of the image. Accordingly, it is important to consider both the computing environment and the classification objects in optimizing the performance of the parallel gw*k*-NN classifier.

5. Conclusion

With the advancement of sensor and computer technologies in data

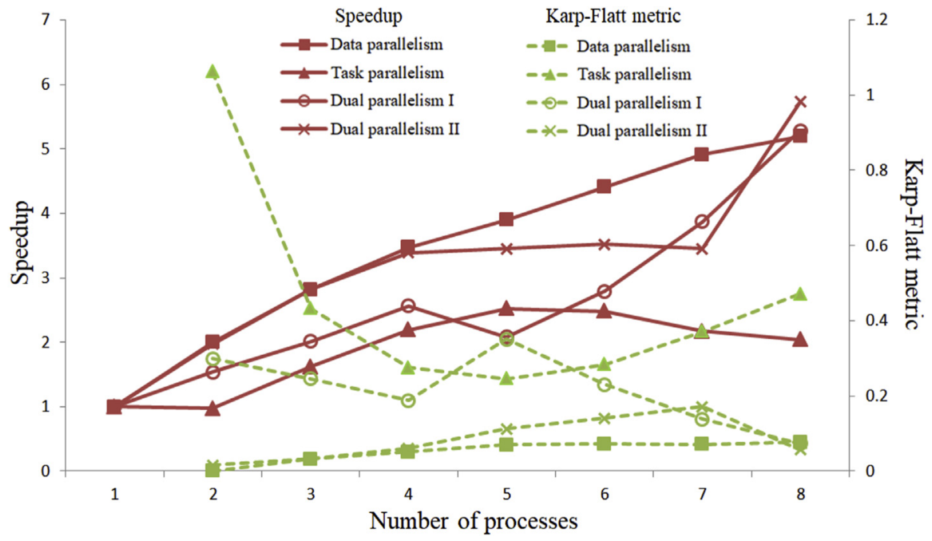

Fig. 10. Speedups and Karp-Flatt metrics under different parallel strategies in the image classification stage.

collection and manipulation, the rapid growth in high-resolution remotely sensed imaging has led to computational demands in many time-critical remote sensing applications. To take full advantage of HPC technologies, we designed a parallel gw*k*-NN classifier in the model training and image classification stages and further implemented it by using calls to MPI and GDAL in the C++ development environment on an eight-core CPU. Moreover, the parallel performance of different parallel strategies (data parallelism and task parallelism, as well as their combination—dual parallelism) was evaluated in terms of running time, speedup, and efficiency, as well as their limiting factors.

Our preliminary experiment demonstrated that the proposed parallel gw*k*-NN classifier can improve the efficiency of high-resolution remotely sensed images with multiple land cover types. In general, the data parallelism method was effective in both the model training and image classification stages, while the dual parallelism method could exploit the potentialities of data parallelism and task parallelism in image classification. Specifically, the two largest speedups were achieved under dual parallelism I and II with eight processes, as opposed to those under pure data or task parallel strategies.

The parallel performance of multiple gw*k*-NN classifiers is limited by inherently sequential computation and communication overhead. Given the fixed size of an image, the parallel overhead under data parallelism tends to grow with the increasing number of processes, but it does not constitute a major part in the total time of parallel computation. However, frequent communication between master and slave processes under task parallelism leads to unsatisfactory performance. The dual parallelism method can reduce the execution time and improve parallel efficiency by overlapping computation and data transmission under certain allocation and scheduling conditions. In particular, dual parallelism II with data decomposition priority achieves the best performance with eight processes.

Although the proposed parallel gw*k*-NN classifier can be significantly improved in terms of performance under different parallel strategies, we only tested the parallel algorithm on a standalone eight-core computer. Further studies may involve the following: (1) use of a parallel cluster environment to enhance the performance of the parallel gw*k*-NN classifier, (2) introduction of additional geostatistical models for multiple land cover classes to achieve more accurate classification, (3) use of remotely sensed data from larger areas to test the scalability of the parallel gw*k*-NN classifier, and (4) consideration of dynamic load balancing to generate more balanced distribution of loads across processes.

Computer code availability

The parallel gwk-NN classifier is implemented by using calls to MPI and GDAL in the C++ development environment on an eight-core CPU and the implementation code for the classifier is available at Github:

https://github.com/zxy919781142/Design-and-implementation-of-a-parallel-geographically-weighted-k-nearest-neighbor-classifier.

The different parallelism methods applied to parallelize the classifier in the model training and image classification stages are supplied in the website.

Authorship statement

Yingxia PU proposed the research idea, designed the parallel strategies, and drafted the manuscript. Xinyi ZHAO designed some of the strategies, wrote the codes, analyzed the results, and drafted the manuscript. Guangqing CHI advised on research design and framed and revised the manuscript. Shuhe ZHAO provided the experimental data. Jiechen WANG provided beneficial suggestions about parallel algorithms. Zhibing JIN conducted the field surveys and provided beneficial suggestions about data interpretation. Junjun YIN revised the manuscript.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China (Grant #41771417), Priority Academic Program Development of Jiangsu Higher Education Institutions, the U.S. National Science Foundation (Awards #1541136 and #1823633), the U.S. Department of Transportation (Awards #DTRT12GUTC14-201307 and #DTRT12GUTC14-201308), and the Eunice Kennedy Shriver National Institute of Child Health and Human Development (Award #P2C HD041025).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cageo.2019.02.009.

Note: Overall accuracy = total number of correct pixels (i.e., the sum of the main diagonal)/total number of pixels; User's accuracy = total number of correct pixels in a category/total number of pixels classified in that category; Producer's accuracy = total number of correct pixels in a category/total number of pixels of that category as derived from the reference data (Congalton, 1991).

References

Andekah, Z.A., Naderan, M., Akbarizadeh, G., 2017. Semi-supervised Hyperspectral Image Classification Using Spatial-Spectral Features and Superpixel-Based Sparse Codes. 2017 Iranian Conference on Electrical Engineering. ICEE, Tehran, pp. 2229–2234.

Atkinson, P.M., 2004. Spatially weighted supervised classification for remote sensing. Int. J. Appl. Earth Obs. Geoinf. 5 (4), 277–291.

Atkinson, P.M., Lewis, P., 2000. Geostatistical classification for remote sensing: an introduction. Comput. Geosci. 26 (4), 361–371.

Atkinson, P.M., Naser, D.K., 2010. A geostatistically weighted k-NN classifier for remotely sensed imagery. Geogr. Anal. 42 (2), 204–225.

Blaschke, T., 2010. Object based image analysis for remote sensing. ISPRS J. Photogrammetry Remote Sens. 65 (1), 2–16.

Caccetta, P.A., Campbell, N.A., West, G.A., 1996. A massively parallel implementation of an image classifier. In: Proceedings of the 16th Asian Conference on Remote Sensing, pp. 203–307.

Chen, L., Ma, Y., Liu, P., Wei, J., Jie, W., He, J., 2015. A review of parallel computing for large-scale remote sensing image mosaicking. Clust. Comput. 18, 517–529.

Chen, Y., Ge, Y., Jia, Y., 2016. Integrating object boundary in super-resolution land-cover mapping. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 10 (1), 219–230.

Choi, S.W., Lee, C.H., 2013. A FPGA-based parallel semi-naive Bayes classifier implementation. IEICE Electron. Express 10 (19), 1–7.

Comber, A.J., Harris, P., Tsutsumida, N., 2016. Improving land cover classification using input variables derived from a geographically weighted principal components analysis. ISPRS J. Photogrammetry Remote Sens. 119, 347–360.

Congalton, R.G., 1991. A review of assessing the accuracy of classifications of remotely sensed data. Remote Sens. Environ. 37 (1), 35–46.

Curran, P.J., 1988. The semivariogram in remote sensing: an introduction. Remote Sens. Environ. 24 (3), 493–507.

Curran, P.J., Atkinson, P.M., 1998. Geostatistics and remote sensing. Prog. Phys. Geogr. 22 (1), 61–78.

Czech, Z.J., 2016. Introduction to Parallel Computing. Cambridge University Press, Cambridge, UK, pp. 66.

De Jong, S.M., van der Meer, F.D., 2004. Remote Sensing Image Analysis: Including the Spatial Domain. Springer, Dordrecht, The Netherlands, pp. 359.

Dungan, J., 1998. Spatial prediction of vegetation quantities using ground and image data. Int. J. Remote Sens. 19 (2), 267–285.

GDAL, 2015. Geospatial Data Abstraction Library, version 1.11.2. Open Source Geospatial Foundation accessed date: June 2015. http://www.gdal.org.

Gharbia, R., Baz, A.H.E., Hassanien, A.E., Tolba, M.F., 2014. Remote sensing image fusion approach based on Brovey and Wavelets transforms. In: Kömer, P., Abraham, A., Snášel, V. (Eds.), Proceedings of the Fifth Intern. Conf. On Innov. in Bio-Inspired Comput. and Appl. IBICA 2014, Advances in Intelligent Systems and Computing 303. Springer International Publishing, Switzerland, pp. 311–321.

Grama, A.Y., Gupta, A., Kumar, V., 1993. Iso efficiency: measuring the scalability of parallel algorithms and architectures. IEEE Parallel Distrib. Technol. 1 (3), 12–21.

Gualtieri, J.A., 2005. A parallel processing algorithm for remote sensing classification. In: Proc. Airborne Earth Science Workshop, Pasadena, CA, pp. 1–14.

Hay, G.J., Niemann, K.O., Mclean, G., 1996. An object-specific image-texture analysis of H-resolution forest imagery. Remote Sens. Environ. 55, 108–122.

Herzfeld, U.C., Higginson, C.A., 1996. Automated geostatistical seafloor classification-principles, parameters, feature vectors, and discrimination criteria. Comput. Geosci. 22 (1), 35–52.

Johnson, B., Tateishi, R., Xie, Z., 2012. Using geographically weighted variables for image classification. Remote Sens. Lett. 3 (6), 491–499.

Jóźwik, A., Serpico, S., Roli, F., 1998. A parallel network of modified 1-NN and k-NN classifiers– application to remote-sensing image classification. Pattern Recogn. Lett. 19 (1), 57–62.
Karp, A.H., Flatt, H.P., 1990. Measuring parallel processor performance. Commun. ACM 33 (5), 539–543.
Kettig, R.L., Landgrebe, D.A., 1976. Classification of multispectral image data by extraction and classification of homogeneous objects. IEEE Trans. Geosci. Electron. 14 (1), 19–26.
Kinghorn, D., 2015. 5 Ways of Parallel Programming. http://www.pugetsystems.com/labs/hpc/5-Ways-of-Parallel-Programming-652.
Kubota, K., Nakase, A., Sakai, H., Oyanagi, S., 2000. Parallelization of decision tree algorithm and its performance evaluation. In: Proceeding of the Fourth International Conference/exhibition on High Performance Computing in the Asia-Pacific Region, Beijing, China, vol. 2. pp. 574–579.
Lee, C.A., Gasster, S.D., Plaza, A., Chang, C., Huang, B., 2011. Recent developments in high performance computing for remote sensing: a review. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 4 (3), 508–527.
Li, M., Zang, S., Zhang, B., Li, S., Wu, C., 2014. A review of remote sensing image classification techniques: the role of spatio-contextual information. Eur. J. Remote Sens. 47, 389–411.
Li, W., Fu, H., You, Y., Yu, L., Fang, J., 2017. Parallel multiclass support vector machine for remote sensing data classification on multicore and many-core architectures. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. 10 (10), 4387–4398.
Lusk, E., Gropp, W., 1995. The MPI message-passing interface standard: overview and status. Adv. Parallel Comput. 10 (6), 265–269.
Ma, Y., Wu, H., Wang, L., Huang, B., Ranjan, R., Zomaya, A., Jie, W., 2015. Remote sensing big data computing: challenges and opportunities. Future Gener. Comput. Syst. 51, 47–60.
Maselli, F., Chirici, G., Bottai, L., Corona, P., Marchetti, M., 2005. Estimation of Mediterranean forest attributes by the application of k-NN procedures to multitemporal Landsat ETM+ images. Int. J. Remote Sens. 26 (17), 3781–3796.
MPICH, 2012. Message-passing Interface, Version 1.5. Message Passing Interface Forum. http://www.mpich.org, Accessed date: 1 June 2013.
Moser, G., Serpico, S.B., Benediktsson, J.A., 2013. Land-cover mapping by Markov modeling of spatio-contextual information in very-high-resolution remote sensing images. In: Proceedings of the IEEE, vol. 101. pp. 631–651 (3).
Myint, S.W., Gober, P., Brazel, A., Grossman-Clarke, S., Weng, Q., 2011. Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery. Remote Sens. Environ. 115 (5), 1145–1161.
Nicolescu, C., Jonker, P.P., 2002. A data and task parallel image processing environment. Parallel Comput. 28 (7), 945–965.
Orii, S., 2010. Metrics for evaluation of parallel efficiency toward highly parallel processing. Parallel Comput. 36 (1), 16–25.
Plaza, A., 2009. Special issue on architectures and techniques for real-time processing of remotely sensed images. J. R. Time Imag. Process. 4 (3), 191–193.
Plaza, A., Chang, C., 2008. High Performance Computing in Remote Sensing. Chapman & Hall/CRC, Boca Raton, London, New York, pp. 451.
Plaza, A., Du, Q., Chang, Y., King, R.L., 2011a. High performance computing for hyperspectral remote sensing. IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. 4 (3), 528–544.
Plaza, A., Plaza, J., Paz, A., Sanchez, S., 2011b. Parallel hyperspectral image and signal processing. IEEE Signal Process. Mag. 28, 119–126.
Pohl, C., Genderen, J.V., 2016. Remote Sensing Image Fusion: A Practical Guide. CRC Press, Taylor & Francis Group, pp. 115.
Qiao, C., Shen, Z., Wu, N., Hu, X., Luo, J., 2011. Remote sensing image classification method supported by spatial adjacency. J. Remote Sens. 12, 88–99.
Qiao, C., Wang, J., Shang, J., Daneshfar, B., 2015. Spatial relationship-assisted classification from high-resolution remote sensing imagery. Int. J. Digit. Earth 8 (9), 710–726.
Qin, C., Zhan, L., Zhu, A., Zhou, C., 2014. A strategy for raster-based geocomputation under different parallel computing platforms. Int. J. Geogr. Inf. Sci. 28 (11), 2127–2144.
Ramstein, G., Raffy, M., 1989. Analysis of the structure of radiometric remotely sensed images. Int. J. Remote Sens. 10 (6), 1049–1073.
Ramaswamy, S., Sapatnekar, S., Banerjee, P., 1997. A framework for exploiting task and data parallelism on distributed memory multicomputers. IEEE Trans. Parallel Distrib. Syst. 8 (11), 1098–1116.
Stuckens, J., Coppin, P.R., Bauer, M.E., 2000. Integrating contextual information with per-pixel classification for improved land cover classification. Remote Sens. Environ. 71 (3), 282–296.
Subhlok, J., Vondran, G., 2000. Optimal use of mixed task and data parallelism for pipelined computations. J. Parallel Distrib. Comput. 60 (3), 297–319.
Tso, B.C.K., Mather, P.M., 1999. Classification of multisource remote sensing imagery using a genetic algorithm and Markov random fields. IEEE Trans. Geosci. Remote Sens. 37 (3), 1255–1260.
van Zyl, J., 2012. Application of satellite remote sensing data to the monitoring of global resources. In: 2012 IEEE Technology Time Machine Symposium (TTM), Dresden, Germany, 1 − 1..
Vrabel, J., 1996. Multispectral imagery band sharpening study. Photogramm. Rem. Sens. 62 (9), 1075–1083.
Wang, L., Shi, C., Diao, C., Ji, W., Yin, D., 2016. A survey of methods incorporating spatial information in image classification and spectral unmixing. Int. J. Remote Sens. 37 (6), 3870–3910.
Woodcock, C.E., Strahler, A.H., Jupp, D.L.B., 1998a. The use of variograms in remote sensing I: scene models and simulated images. Remote Sens. Environ. 25 (3), 323–348.
Woodcock, C.E., Strahler, A.H., Jupp, D.L.B., 1998b. The use of variograms in remote sensing II: real digital images. Remote Sens. Environ. 25 (3), 349–379.